Enterprise Computing

Band 3
Praktische Übungen unter z/OS

Prof. Dr.-Ing. Wilhelm G. Spruth
Prof. Dr. Martin Bogdan

spruth@informatik.uni-leipzig.de
bogdan@informatik.uni-leipzig.de

September 2013.

Die Vorderseite des Buches zeigt einen zEC12 Rechner mit angeschlossener zEnterprise Blade Center Extension (zBX). Die Darstellung entstand aus 2 Abbildungen in dem IBM Redbook : IBM zEnterprise EC12 Technical Guide, August 2012, SG24-8049-00.

Vorwort

Die Studenten unserer e-Learning Vorlesung investieren einen erheblichen Teil ihres Lernaufwandes (etwa 50 %) in die Bearbeitung praktischer Übungen auf dem z/OS Rechner des Lehrstuhls Technische Informatik.

Wir betrachten praktische Übungen als ein Kernelement für ein erfolgreiches e-Learning. Während bei vielen anderen Informatik Disziplinen praktische Übungen auf dem häuslichen PC durchgeführt werden können, ist bei einer Mainframe Ausbildung ein Zugriiff auf einen z/OS Server erforderlich.

Die Übungen werden teilweise im Wintersemester, teilweise im darauffolgenden Sommersemester durchgeführt. Nicht alle Übungen werden in jedem Vorlesungszyklus behandelt. Gelegentlich bieten wir den Studenten an, wahlweise die eine oder die andere Übung durchzuführen.

Unser Problem mit den praktischen Übungen besteht darin, dass wir für die Betreuung der Studenten zusätzliche Hilfskräfte einsetzen müssen, deren Finanzierung für uns regelmäßig ein Problem darstellt. Um diesen Betreuungsaufwand so klein wie möglich zu halten, enthalten die Übungstexte (Tutorials) sehr detaillierte Screen by Screen Anweisungen. Der Nachteil ist, es verleitet manche Übungsteilnehmer dazu, mechanisch Cut und Paste durchzuführen, ohne den Hintergrund verstanden zu haben. Wir kompensieren dies, indem wir in der mündlichen Prüfung dieses Verständnis spezifisch hinterfragen. Auf der anderen Seite eignen sich die Tutorialtexte damit gut zum Selbststudium.

Unsere praktischen Übungen können auch auf älteren Versionen von z/OS durchgeführt werden. Wir benutzen in der Regel z/OS Version 1.8. Eine Umstellung auf eine modernere z/OS Version bringt in fast allen Fällen keine Vorteile. Die beiden RMI Tutorials in Band3, Abschnitt 8 laufen unter zLinux.

Neben dem Inhalt von Band 3 existieren eine Reihe weiterer praktischer Übungen, welche wir für längere Zeit nicht benutzt haben, die aber bei Bedarf reaktiviert werden könnten. Sie können auf Wunsch „as is“ zur Verfügung gestellt werden.

Inhaltsverzeichnis

Seite

6. EJB 3.0

7. Java Communication

8. JCICS 3.0

9. Web Services

Weiterführende Information

1. Einführung

1.a Dateiverwaltung unter TSO

Tutorials sind Anweisungen, mit deren Hilfe Sie praktische Übungsaufgaben auf unserem Mainframe Rechner durchführen können.

Tutorial 1a beschreibt, wie sie auf den Rechner zugreifen können.

In dieser Aufgabe lernen Sie kennen, wie man eine Verbindung zu einem z/OS-Rechner herstellen kann, wie man Dateien anlegt sowie wie man Dateien mit Daten füllen oder den Dateiinhalt mittels des ISPF-Editors verändern kann. Sie lernen ebenfalls das Dateiformat unter TSO kennen, das sich signifikant von den Dateiformaten unter Windows oder UNIX / LINUX unterscheidet.

Hinweis: Dieses Tutorial wurde unter Verwendung der Benutzer-ID "PRAK25" erstellt. In allen Dateinamen müssen Sie "PRAK025" durch ihre eigene Benutzer-ID ersetzen.

Aufgabe: ***Arbeiten Sie sich anhand des nachfolgenden Tutorials in TSO /ISPF ein.***

Inhalt

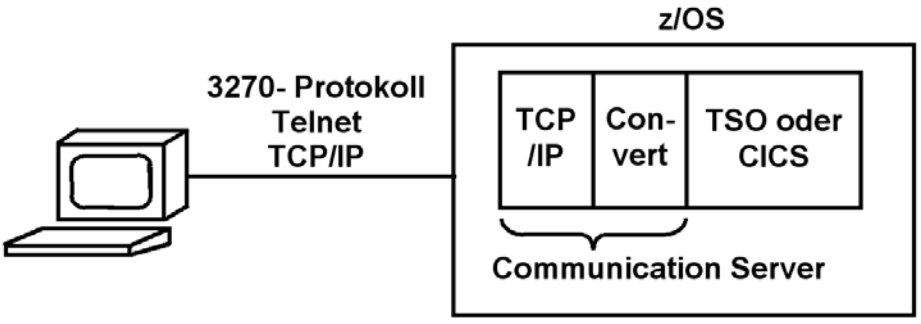
1. Voraussetzungen

Das Arbeiten mit einem z/OS Rechner ist ein wesentlicher und wichtiger Bestandteil der Mainframe Ausbildung. Hierfür stehen uns die Rechner jedi.informatik.uni-leipzig.de sowie hobbit.cs.uni-tuebingen zur Verfügung. Sie erhalten für diesen Rechner eine Benutzerkennung für die Dauer Ihrer Ausbildung. Sie können auf den Rechner von zu Hause mit Ihrem Windows oder Linux PC zugreifen.

Auf beiden Rechnern jedi.informatik.uni-leipzig.de läuft (unter anderem) das Betriebssystem z/OS Version 1.8 mit zahlreichen weiteren Komponenten. Ein Zugriff auf z/OS erfolgt in der Regel über eins von mehreren z/OS Subsystemen. Beispiele für hierfür geeignete Subsysteme sind TSO, CICS, WebSphere, DB2 und andere. Jedes Subsystem hat seinen eigenen Zugriffsmechanismus. Im ersten Teil des Lehrgangs greifen wir auf das TSO Subsystem zu.

Um mit Hilfe eines Arbeitsplatzrechners auf einen Server zugreifen zu können braucht man eine Zugriffskomponente, die allgemein als Client bezeichnet wird. Einige Beispiele für Klientensoftware sind der Browser, der Thunderbird Mail Klient, der Telnet Client (z.B. putty) und der FTP Client (z.B. ws_ftple).

Ein Zugriff auf z/OS kann über eine graphische Benutzerschnittstelle (Graphical User Interface, GUI) oder eine Kommandozeilenschnittstelle (Command line user interface, CUI) erfolgen. z/OS unterstützt all diese Möglichkeiten. Am weitesten gebräuchlich ist jedoch die Benutzung eines Kommandozeilenschnittstelle mit Hilfe eines 3270 Klienten. Dieser kommuniziert über das 3270-Übertragungsprotokoll mit dem z/OS-"Communication Server“ Subsystem des z/OS Betriebssystems.



TSO (Time Sharing Option) ist das Standard z/OS Subsystem für remote Klientenzugriffe. Die TSO Kommandozeilenschnittstelle ist vergleichbar mit der Linux bash shell oder der Windows DOS Shell. Eine Shell ist ein Interpreter, der Kommandos entsprechend seiner eigenen Syntax interaktiv oder selbständig ausführt. Die Semantik (nicht aber die Syntax) aller Kommandozeilen-Shells haben viele Ähnlichkeiten. Dabei unterscheidet sich die Syntax der TSO Shell sehr von der Syntax der Linux Shell oder der Syntax der Windows Shell.

Ein TSO Client benutzt das 3270 Übertragungsprotokoll um mit dem z/OS "Communication Server" Subsystem und dann mit TSO zu kommunizieren. Das 3270 Übertragungsprotokoll setzt auf dem Telnet Protokoll auf, und verwendet ebenfalls den TCP-Port 23 auf der Serverseite. Der 3270-Klient wird allgemein als "3270 Emulator" bezeichnet.

Für den Rechner Zugriff müssen Sie einen 3270 Emulator auf Ihrem PC installieren. Weit mehr als ein Dutzend Firmen vertreiben 3270 Emulatoren für Windows oder Linux Plattformen.

1.1 Windows

Unter Windows empfehlen wir den kostenlosen "Quick32Freeware" Emulator . Dies ist eine funktionsmäßig eingeschränkte Version des Quick3270 Emulators, die für Ausbildungszwecke ausreicht. Der Emulator liegt unter http://www.informatik.uni-leipzig.de/cs/support/index.html zum Download bereit. Die kommerzielle Version ist erhältlich unter http://www.dn-computing.com/Quick3270.htm.

Von der Northern Illinois University
http://www.cs.niu.edu/compresource/compresource.html
können Sie den QWS3270 Emulator herunterladen:
QWS3270: MVS Terminal Program for Win95/98/NT/XP

Eine andere kostenlose Alternative ist der wc3270 Emulator. Der Emulator liegt ebenfalls unter http://www.informatik.uni-leipzig.de/cs/support/index.html zum Download bereit. Das Manual ist verfügbar unter http://x3270.bgp.nu/download.html.

Von der Firma IBM werden u.a. die Produkte "Personal Communication" und "Host on Demand" (HOD) angeboten. Der Zugriff über das HOD ist auf unseren Server derzeitig nicht möglich.

Andere Firmen bieten Ihnen eine kostenlose 30 oder 60 Tage Demo Version zum Herunterladen an. Wir haben gute Erfahrungen mit dem Vista Emulator der Fa. Tombrennansoftware gemacht, siehe www.tombrennansoftware.com

1.2 Linux

Unter Linux / Unix empfehlen wir den x3270 (oder für Kommandozeile c3270) Emulator. Beide sind Open Source und sind noch auf der jeweiligen Zielplattform zu kompilieren.

Download under
http://x3270.sourceforge.net/
http://x3270.bgp.nu/download.html
http://www.geocities.com/SiliconValley/Peaks/7814/

x3270 ist auch bei gängigen Linux Distributionen enthalten (z.B. SuSE Linux). Danach einfach x3270 ausführen und mit 139.18.4.30 verbinden.

1.3 Andere

Anweisungen für die x3270 Installation unter Mac OS X ist zu finden unter http://planetmvs.com/mvsintosh/x3270.html

Nutzer von MacOs können auf das Programm tn3270 zurückgreifen. Dieses ist auf der Seite http://www.brown.edu/cis/tn3270/index.html der Brown Universität erhältlich.

Zahlreiche weitere Hilfen für X3270 sind im Internet unter Google verfügbar.

Manche 3270 Emulatoren verwenden das Telnet Protokoll nur für die erste Verbindungsaufnahme. Danach wird ein Java Applet geladen; alle weitere Communication findet innerhalb eines Web Browsers statt. Das 3270 Protokoll setzt dann auf dem HTTP Protokoll auf. Das Applet wird entweder lokal gespeichert, oder wird bei Bedarf herunter geladen.

Welchen 3270 Emulator Sie benutzen ist weitestgehend Geschmacksache; es kann sein, dass wenn sie für ein Unternehmen arbeiten, dort ein bestimmter 3270 Emulator bevorzugt wird.

1.4 Firewall

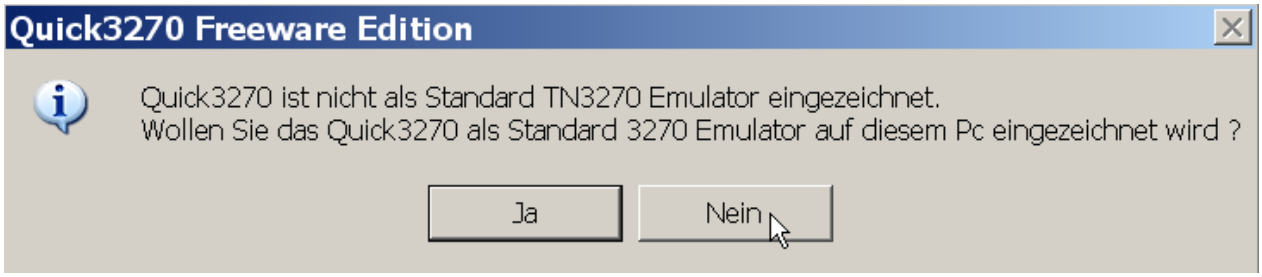
Unser Rechner ist durch keinen externen Firewall geschützt. z/OS verfügt über eine eingebaute Firewall Funktionalität, die Bestandteil des Communication Subsystems und des z/OS Security Servers ist. Dies hat den großen Vorteil, dass Sie sich von zu Hause mit Ihrem PC problemlos über das Internet in unseren z/OS Rechner einloggen können.

Wenn Sie sich von Ihrem Arbeitsplatz an der Hochschule oder in einem Unternehmen aus einloggen, haben Sie evtl. Probleme mit dem dortigen Firewall. Hierauf haben wir keinen Einfluss; dieses Problem müssen Sie mit dem Administrator des dortigen Firewalls direkt lösen.

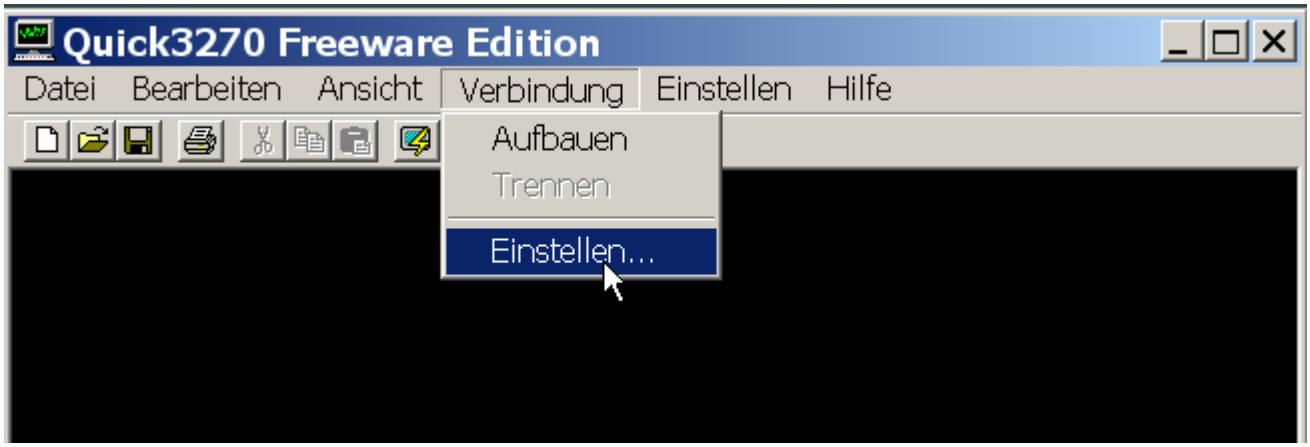
Der nächste Abschnitt beschreibt die Installation von Quick32Freeware. Wenn Sie einen anderen 3270 Emulator bevorzugen können sie diesen Abschnitt überspringen.

2. Installation von Quick3270 Freeware Edition unter Windows XP

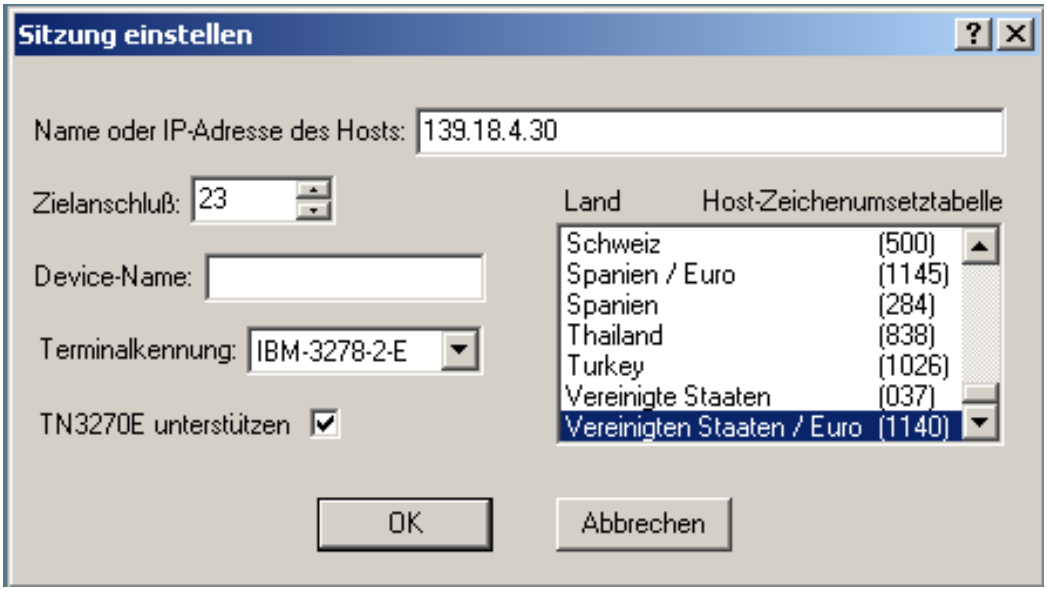
Download Quick32 Freeware Edition. Start install.exe und folgen Sie den Instructions. Legen Sie den ICON auf Ihren Desktop. Start Quick32Freeware.



Vorschlag ist „Nein“. Kann später immer noch erfolgen. Evtl. wollen Sie früher oder später auf einen anderen 3270 Emulator umsteigen.



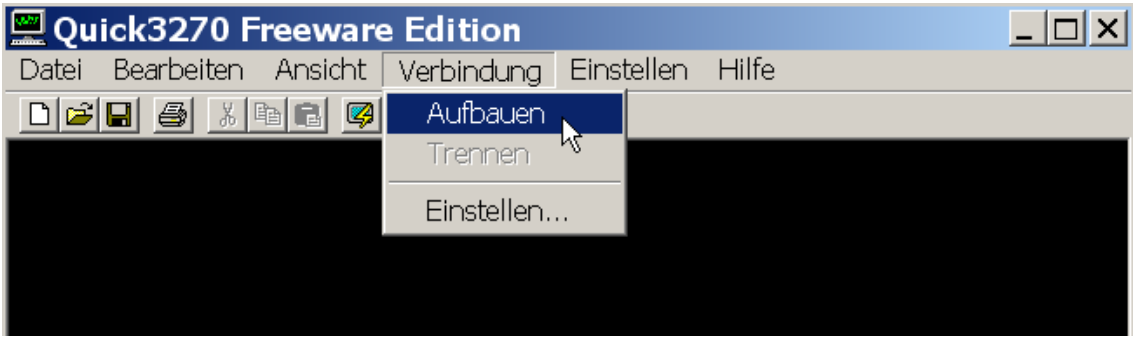
Auf Verbindung – Einstellen klicken



Einstellungen übernehmen wie angegeben. 139.18.4.30 ist die IP Adresse von leia.informatik.uni-leipzig.de . Das 3270 Protokoll ist ein modifiziertes Telnet Protokoll. Deshalb

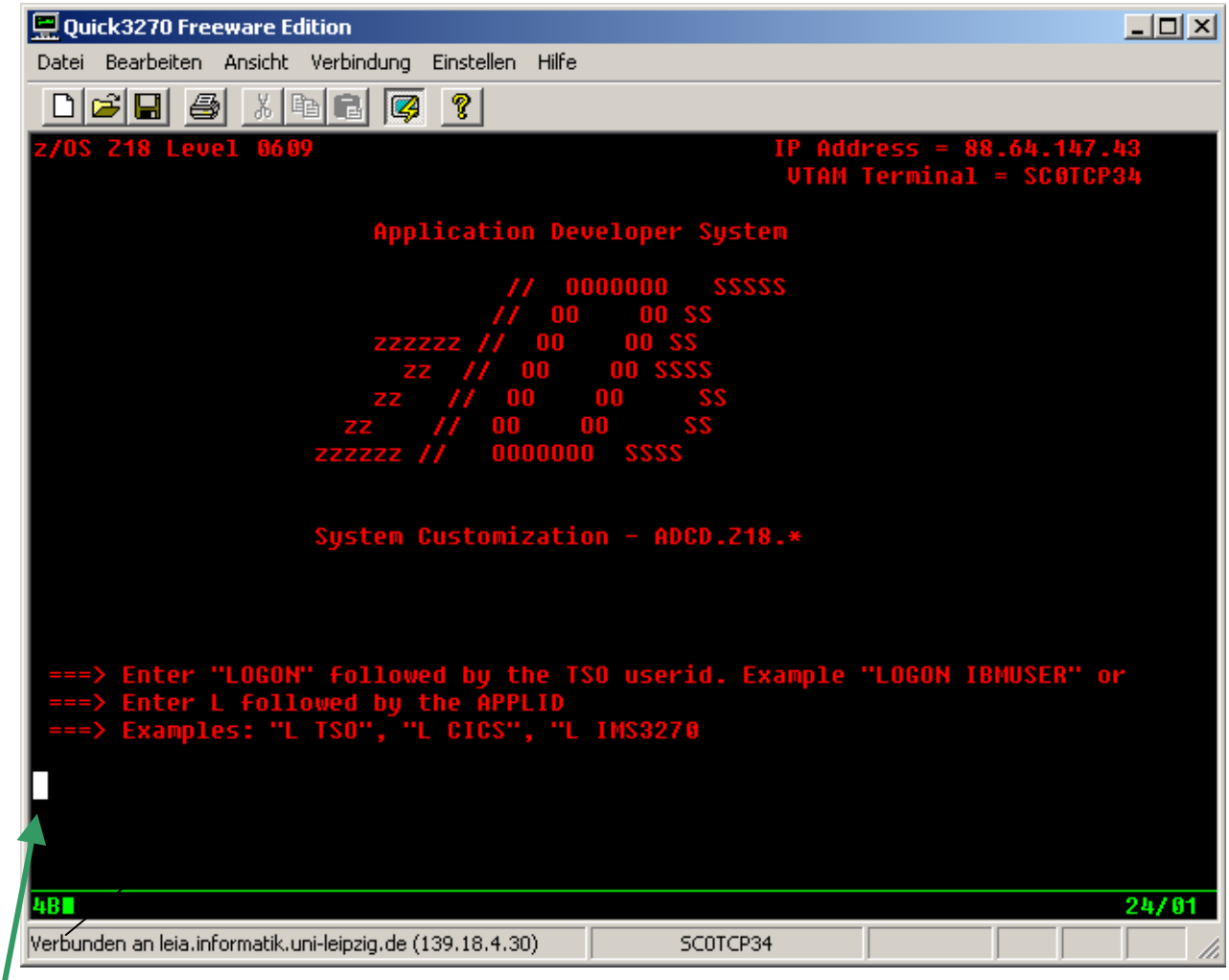
benutzt es standardmäßig Port 23 (auf unserem Tübinger Rechner benutzen wir Port 423). Als Host Zeichenumsetzungstabelle Vereinigte Staaten / Euro (1140) wählen.

OK

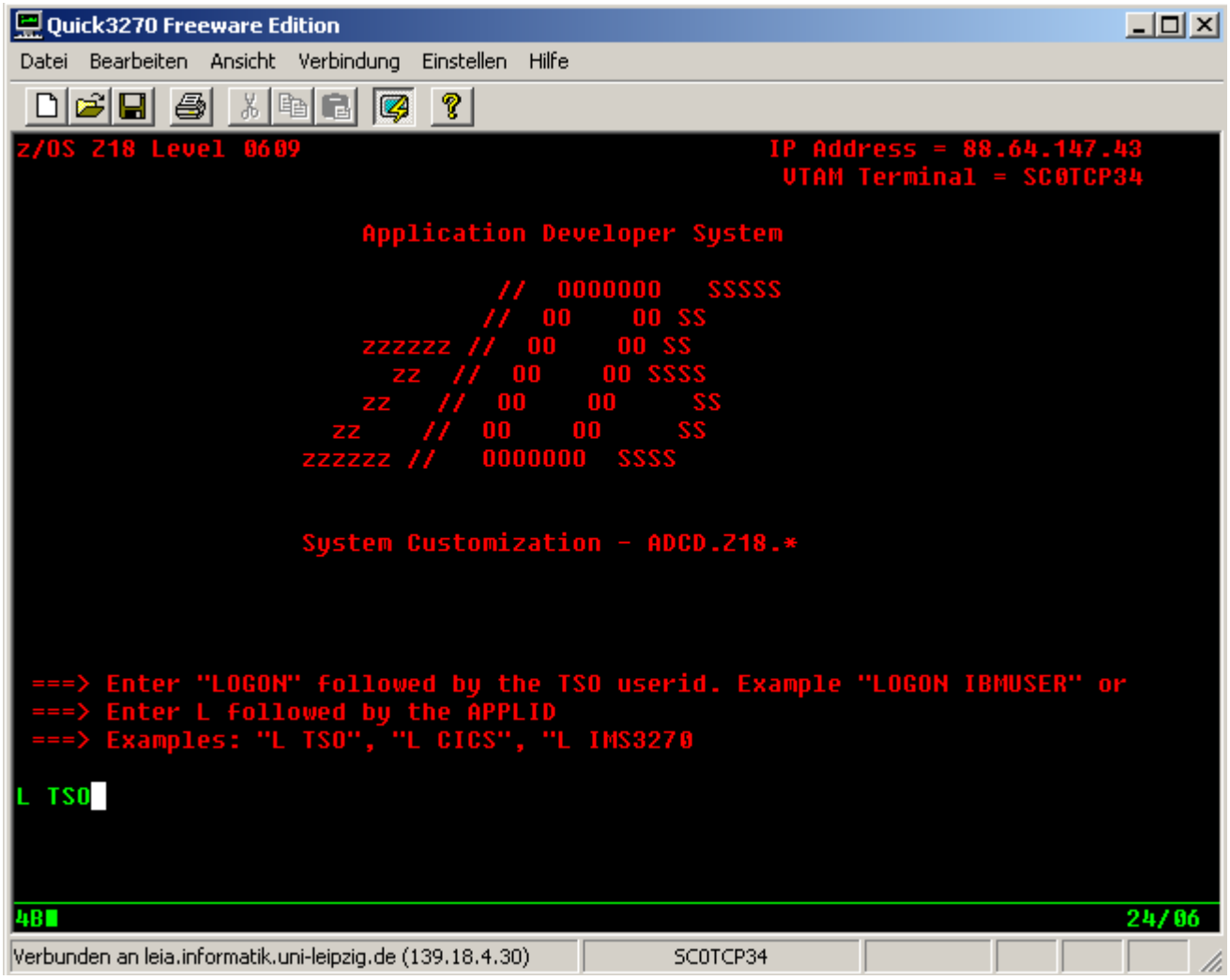


Die Eingabetaste ist - abweichend von der sonst üblichen Norm - bei z/OS standardmäßig die Strg-Taste (!). Allerdings erkennen heutige 3270-Emulatoren meist auch die Enter-Taste als die Eingabe-Taste.

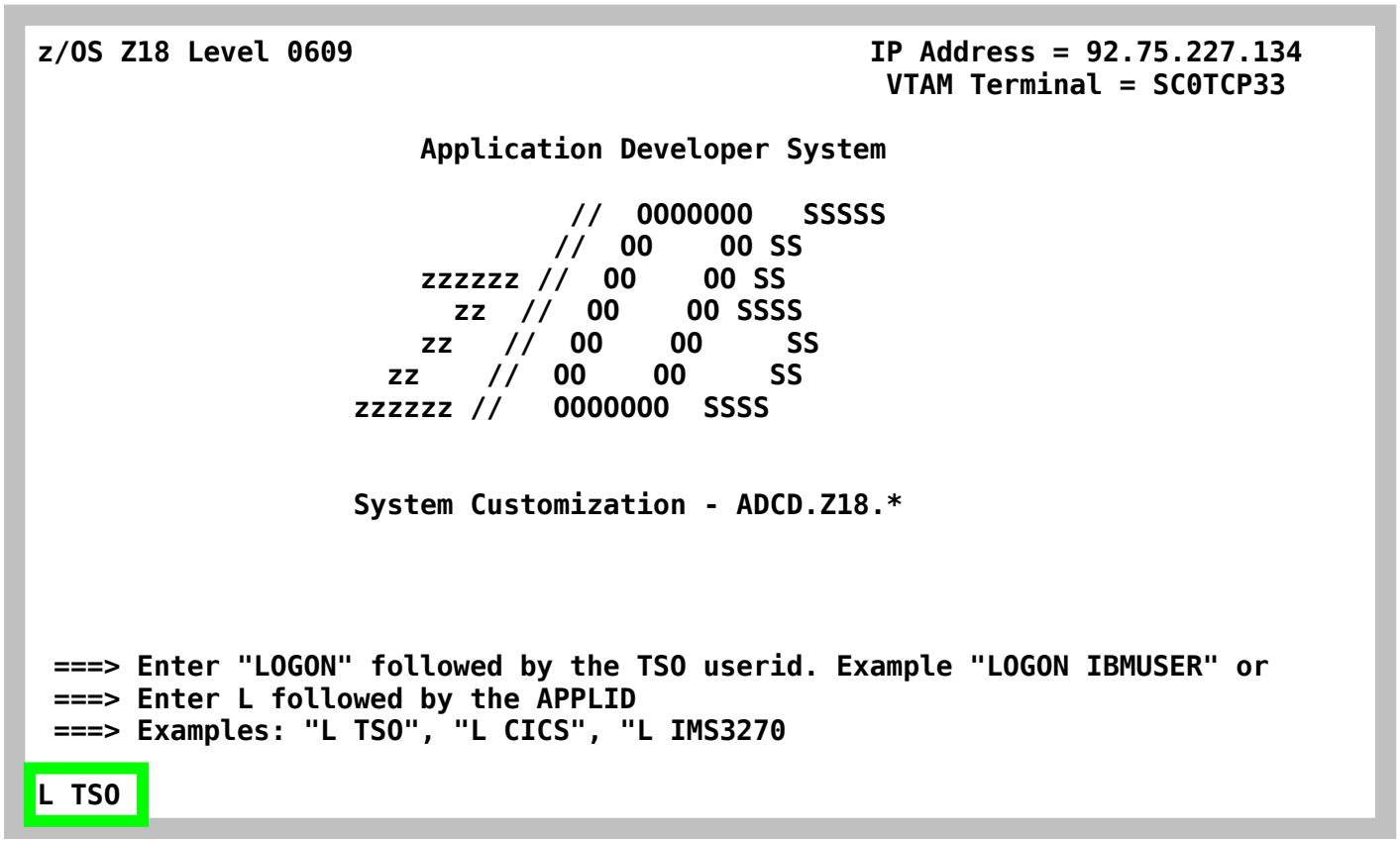
3. Einloggen auf den z/OS-Rechner 139.18.4.30 unter Port 23



Das weiße Rechteck links unten markiert die Cursor Position.



Geben Sie „L TSO“ (oder LOGON TSO) ein.



Ab hier benutzen wir die oben dargestellten Screenshots.

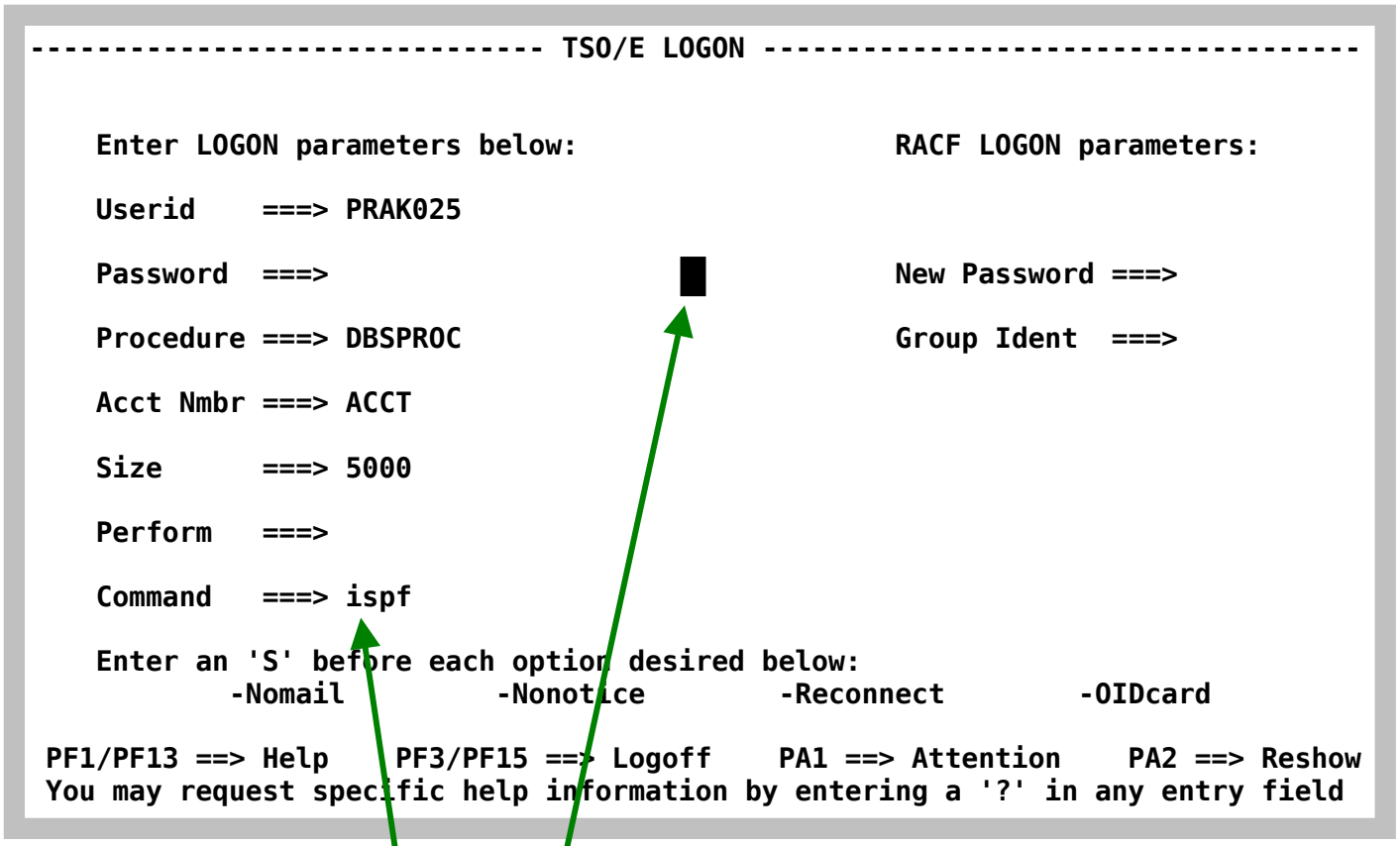
An Stelle von L TSO könnten Sie sich hier mit L CICS in das CICS Subsystem, oder mit L IMS3270 in das IMS Subsystem einloggen.

An dieser Stelle könnten Sie auch kleine Buchstaben, also l tso, verwenden. TSO ist ähnlich wie Windows und anders als Linux nicht Case sensitive. An manchen Stellen akzeptiert TSO aber nur Großbuchstaben; im Anfang ist es besser, auf die Verwendung von Kleinbuchstaben zu verzichten.

Verlassen einer Endlosschleife

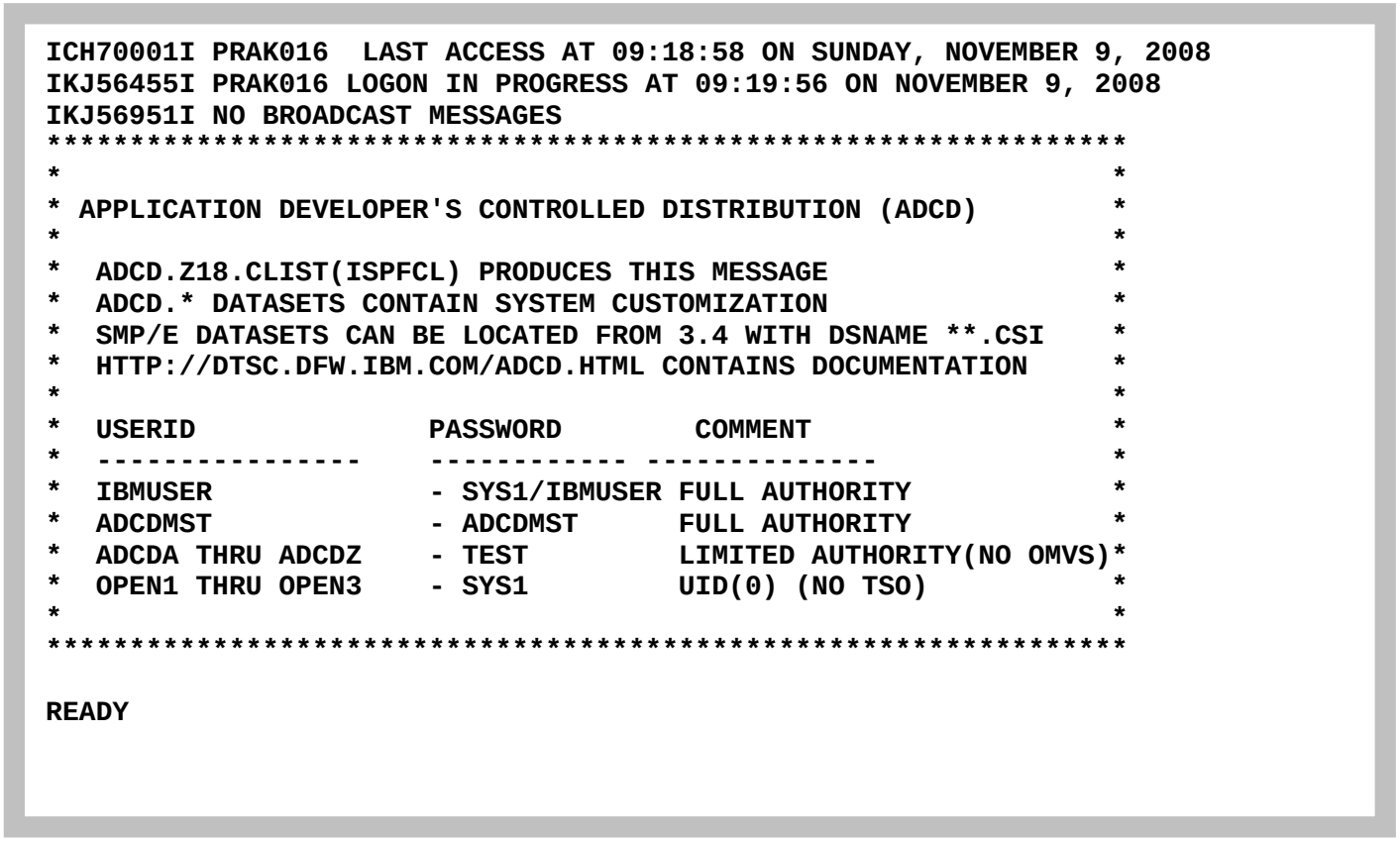
Wenn man sich unter z/OS in einer Endlosschleife befindet, aus der man nicht herauskommt, z.B. weil nach jeder nur denkbaren Eingabe immer wieder die gleiche Aufforderung "IKJ56700A ENTER DATA SET NAME –", ausgegeben wird, drückt man die PA1 Taste. Dies führt zum Verlassen der Endlosschleife.

Der ursprüngliche IBM 3270 Terminal hatte eine andere Tastaturbelegung als der heutige PC. Ein 3270 Emulator bildet die 3270 Taten auf der heutigen PC Tastatur ab. Leider ist die Tastenbelegung von einem 3270 Emulator zum nächsten unterschiedlich. Beim Quick32Freeware Emulator wird die PA1 Taste durch die „Bild up „ (Bild↑) Taste emuliert.



Geben Sie Ihre User ID und Ihr Passwort ein und drücken Sie die Enter Taste. Bei der Eingabe des Passwortes bewegt sich der Cursor nach rechts, es wird aber auf dem Bildschirm nichts wiedergegeben.

Geben Sie als Command „ispf“ ein.



Der Rechner sendet eine Message. Die Bearbeitung des komplexen Logon-Vorganges dauert einige Sekunden. Während dieser Zeit wird in der Mitte kurz die Nachricht "X SYSTEM" sichtbar. Nach kurzer Zeit ist die Nachricht "X SYSTEM" verschwunden. Im Normalfall erscheint die Nachricht "READY" im TSO. READY bedeutet, dass TSO mit der Abarbeitung des vorher abgeschickten Kommandos fertig und für die Eingabe eines weiteren TSO-Kommandos bereit ist (z/OS liebt Großbuchstaben).

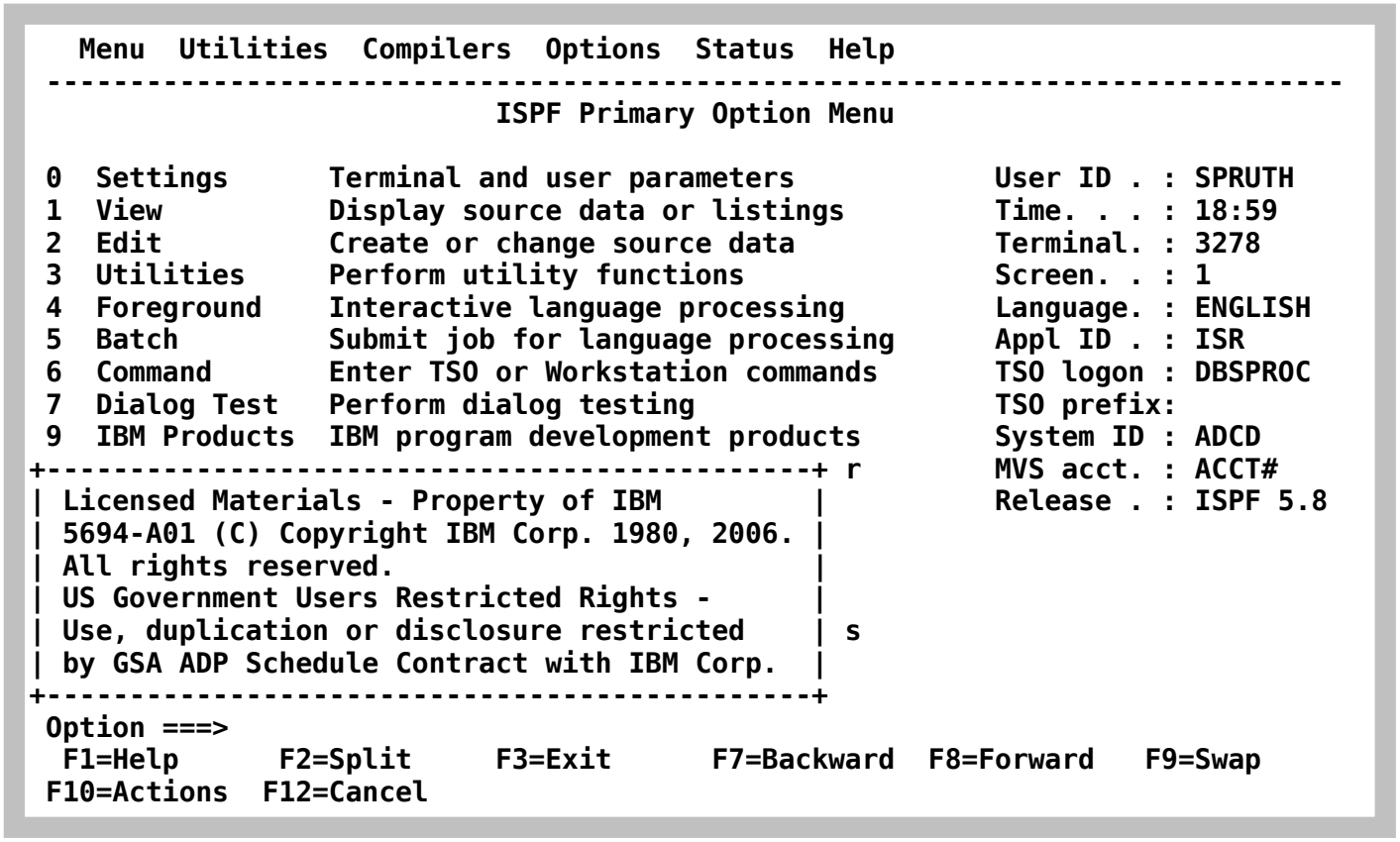
```
ICH70001I PRAK003  LAST ACCESS AT 20:48:42 ON MONDAY, MARCH 31, 2008
IKJ56455I PRAK003 LOGON IN PROGRESS AT 19:41:44 ON SEPTEMBER 3, 2008
IKJ56951I NO BROADCAST MESSAGES
*****************************************************************
*                                                               *
* APPLICATION DEVELOPER'S CONTROLLED DISTRIBUTION (ADCD)        *
*                                                               *
*  ADCD.Z18.CLIST(ISPFCL) PRODUCES THIS MESSAGE                 *
*  ADCD.* DATASETS CONTAIN SYSTEM CUSTOMIZATION                 *
*  SMP/E DATASETS CAN BE LOCATED FROM 3.4 WITH DSNAME **.CSI    *
*  HTTP://DTSC.DFW.IBM.COM/ADCD.HTML CONTAINS DOCUMENTATION     *
*                                                               *
*  USERID              PASSWORD        COMMENT                  *
*  ----------------    ------------ -------------               *
*  IBMUSER             - SYS1/IBMUSER FULL AUTHORITY            *
*  ADCDMST             - ADCDMST      FULL AUTHORITY            *
*  ADCDA THRU ADCDZ    - TEST         LIMITED AUTHORITY(NO OMVS)*
*  OPEN1 THRU OPEN3    - SYS1         UID(0) (NO TSO)           *
*                                                               *
*****************************************************************

 READY
ISPF
```

Sie wollen das ISPF Subsystem benutzen und geben deshalb „ISPF“ gefolgt von Enter ein. Wenn Sie auf dem letzten Bildschirm bereits als Command „ispf“ angegeben haben, ist das hier nicht mehr erforderlich. Die Ausgabe „running“ deutet an, dass die Bearbeitung etwas länger als die normale Zeit in Anspruch nimmt. Das System erwartet nach der Ausgabe von "*", dass der Benutzer etwas unternimmt. In diesem Fall wird die Eingabetaste betätigt. Die Eingabe von ISPF und Enter führt uns zum „ISPF Primary Option Menu“ .**

```
ISPF
***
```

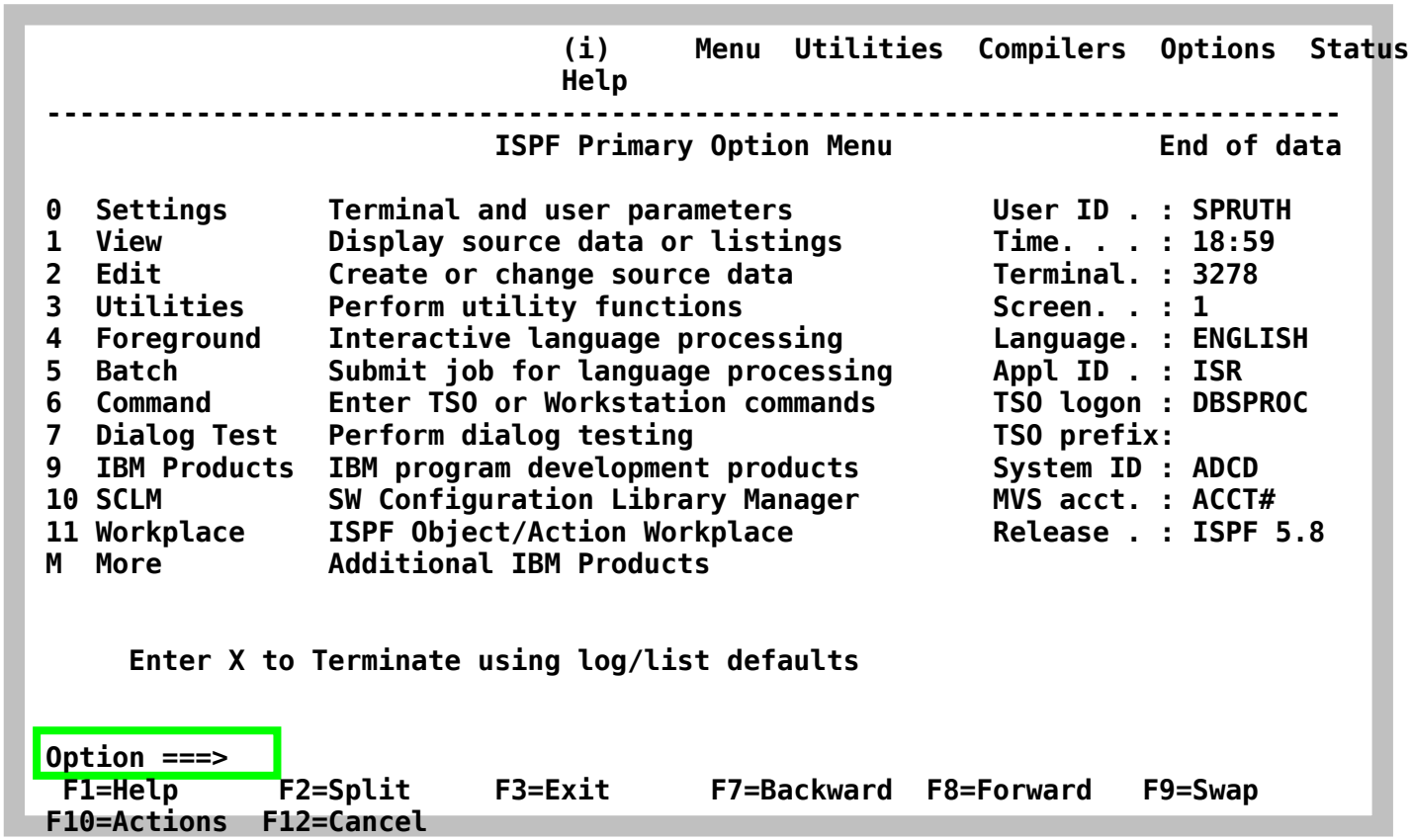
Je nach Einstellung können die Bildschirmwiedergabe (Panel oder Menu unter TSO genannt) unterschiedlich aussehen. Sieht Ihr Panel so aus, drücken Sie nochmals .Enter.

Anhang B zeigt, wie Sie beim Logon erreichen können dass das TSO Kommando „ISPF“ automatisch ausgeführt wird.



Es erscheint das ISPF Primary Options Menu. ISPF ist ein Subsystem des TSO Subsystems, also streng genommen ein Sub-Subsystem. Es bietet etwas mehr Komfort als eine reinrassige Kommandozeilen-Oberfläche. Natürlich ist es möglich, unter TSO auch eine reine Kommandozeilen-Oberfläche zu benutzen. In der Praxis wird fast ausschließlich ISPF benutzt und fast alle TSO Funktionen sind auch unter ISPF vorhanden.

Drücken Sie die F8 Taste, und ....



Der Bildschirm sieht so aus. Das Primary Options Menu ist der Ausgangspunkt für alle Ihre ISPF Aktivitäten.

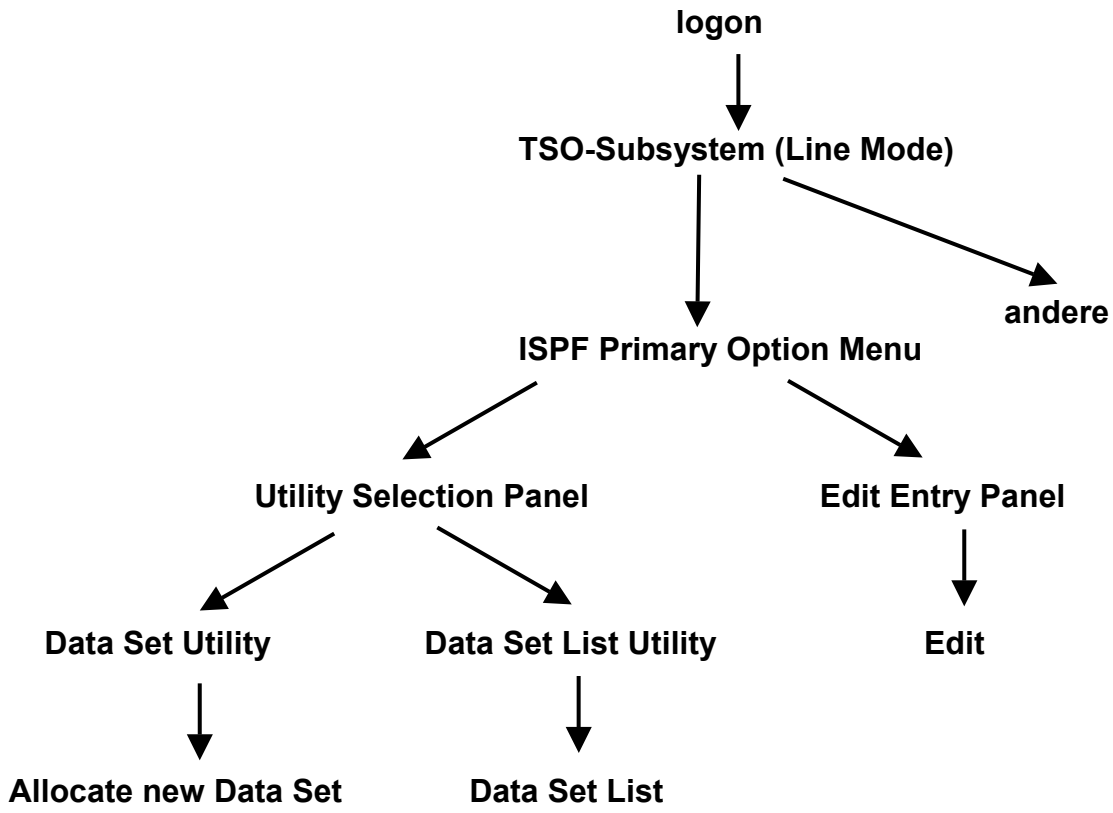
Eine derartige Bildschirmwiedergabe wird auch als "Screen" oder auch als "Panel" bezeichnet. Besonders der Ausdruck "Panel" ist sehr gebräuchlich.

TSO bietet viele Möglichkeiten der Nutzung, zu viele für den Neuling.

Für die unterschiedlichen Nutzungen stehen ISPF Subsysteme zur Verfügung. Einige von ihnen können von diesem Bildschirm aus aufgerufen werden, indem die in der linken Spalte stehende Buchstabenkombination in die zweitoberste Zeile (Kommandozeile) hinter "OPTION ===>" eingegeben wird. Andere Funktionsaufrufe, z.B. TSO-Zeilen-Kommandos, können ebenfalls hier eingegeben werden (dem eigentlichen Kommando muss "TSO" vorangestellt werden, damit es als TSO-Kommando erkannt und entsprechend behandelt wird); sie werden durch den TSO-Kommando-Interpreter abgearbeitet.

Alle Bildschirme in diesem Tutorial werden im 3270-Format dargestellt. Ein 3270-Bildschirm besteht aus 24 Zeilen mit je 80 alphanumerischen Zeichenpositionen. An Stelle von hart-verdrahteten 3270-Endgeräten (Terminals) werden heute meistens PC's eingesetzt, auf denen eine als "3270-Emulator" bezeichnete Software-Komponente den 3270-Bildschirm darstellt.

Häufig verfügt der 3270-Emulator über eine als "Screen Scraper" bezeichnete zusätzliche Komponente, die den 24 x 80 Zeichen-Bildschirminhalt in eine modern und gefällig aussehende Darstellung übersetzt. Beispielsweise könnte ein Pushbutton in einer Zeile die entsprechende Funktion aufrufen. Grafische Gestaltungselemente können die in dem 3270- Datenstrom enthaltene Information benutzerfreundlich darstellen.



2. Funktionen unter TSO und ISPF

4. Erstellen eines Datasets (Allocate)

Wenn sie sich das erste Mal unter TSO einloggen ist Ihre allererste Aufgabe die Zuordung (allocate) von Data Sets.

Ihre Frage sollte sein: Was soll das ? „Das habe ich unter Windows noch nie gemacht“.

Dies ist falsch !!! Wenn Sie unter Windows eine neue Datei erstmalig anlegen, müssen Sie z. B. entscheiten, ob sie unter C: oder D: gespeichert wird. Wenn Ihr Rechner über 24 Festplatten oder Partitionen verfügt, z.B.

C:, D:, E:, ...Y, Z: ,

wird die Verwaltung schon etwas schwieriger. Bei der Auswahl kann auch sein, dass einige der Festplatten (welche ?) mit FAT32 und andere mit NTFS formatiert sind, dass sie unterschiedliche Kapazitäten und/oder Performance Eigenschaften haben , und dass dies für Ihre Entscheidung relevant sein mag. Stellen Sie sich vor, Ihr Rechner hat eine normale Festplatte und zusätzlich eine besonders schnelle, aber kleine, Solid State Disk.

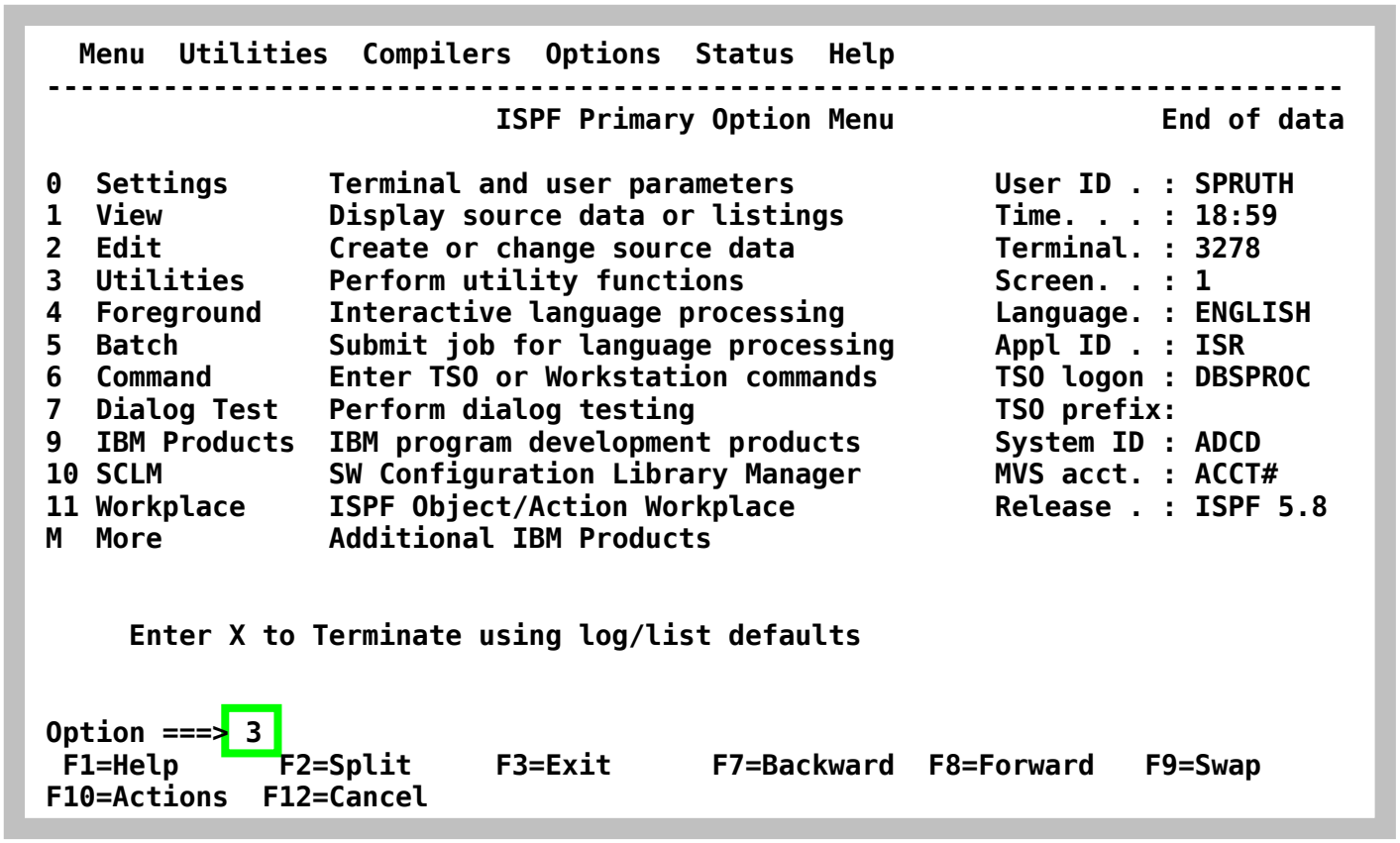
Was machen Sie wenn Ihr z/OS Rechner über 50 000 Festplatten verfügt, dazu 1500 Solid State Disks ? Wenn die 50 000 Platten unterschiedliche Speichervolumen und Zugriffszeiten haben und für unterschiedliche File Systeme formatiert sind ? Wenn für die Datenbestände unterschiedliche Backup Strategien existieren ?

Sie lösen das Problem, indem Sie sorgfältig spezifizieren, wo die Datei gespeichert wird, usw.

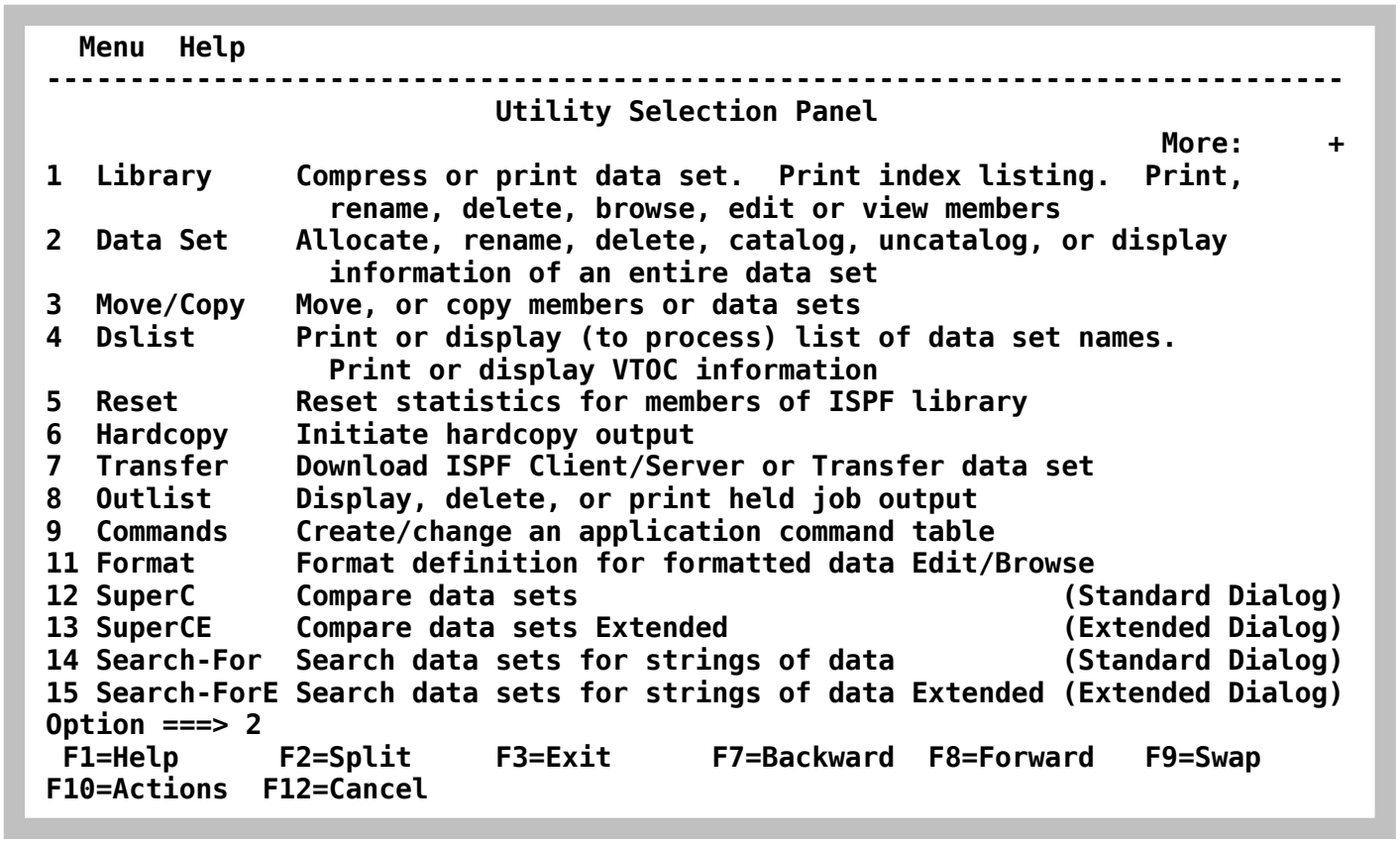
Der Administrator eines großen Servers hat beim Umgang mit den Daten täglich viele Entscheidungen zu treffen. Zusätzlich zum Umgang mit Themen, die nur die Daten oder die Anwendung betreffen, muss er die Speicherverwaltungsmaßnahmen der Installationen kennen. Er muss sich außerdem mit den Themen auseinandersetzen, die Format, Bearbeitung und Positionierung der Daten betreffen:

- **Welchen Wert soll die Blockgröße haben? Wie viel Speicherplatz ist erforderlich?**
- **Welcher Einheitentyp soll verwendet werden? Sollen die Daten in den Cache geschrieben werden? Soll eine Fehlerbehandlung durchgeführt werden?**
- **Wie oft soll eine Sicherung oder Migration durchgeführt werden? Soll die Sicherung/Migration erhalten bleiben oder (wann ?) gelöscht werden?**
- **Welche Datenträger stehen für die Dateipositionierung zur Verfügung?**

z/OS erwartet, dass alle Files (Data Sets, Dateien) bezüglich maximaler Größe, Format und anderer Eigenschaften definiert werden, ehe man sie benutzt. Diese Definitionen erwartet es (anders als bei Unix oder Windows) vom Benutzer. Man verwendet für die Verwaltung eine z/OS Komponente „Storage Management System“ (SMS). . Wenn Sie Platz für eine neue Datei brauchen, melden Sie dies bei SMS an. Dieser Vorgang wird als Allocation bezeichnet.



Wir rufen die Utility-Funktion auf, indem wir eine "3" auf der Kommandozeile (hinter "Option ===> ") eingeben. Anschließend betätigen wir die Eingabetaste.

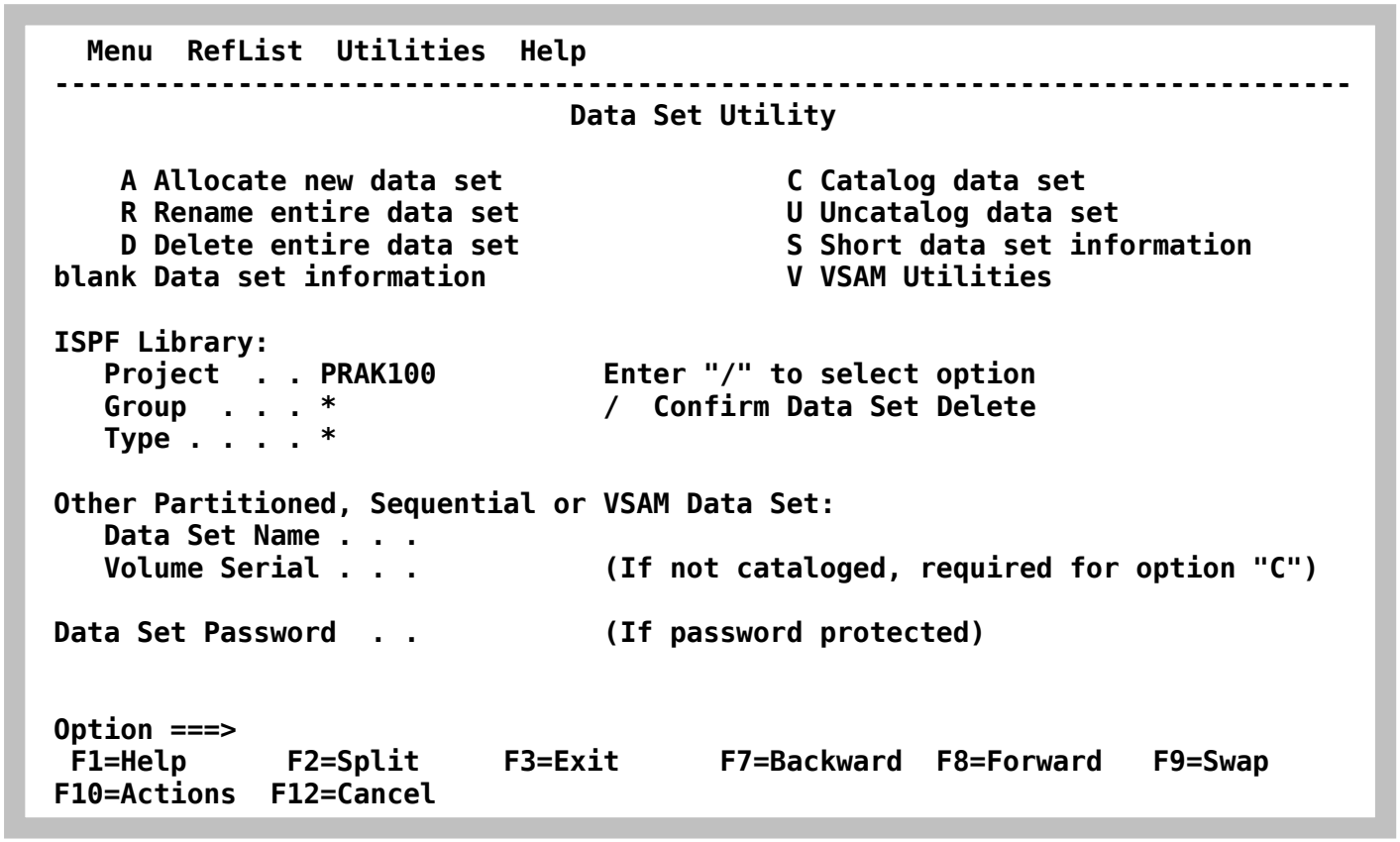


Der "Utility Selection Panel"-Bildschirm erscheint

z/OS verwendet im Gegensatz zu Windows oder Linux größtenteils strukturierte „Data Sets“. Es existieren mehrere Typen von Data Sets. Beispiele für solche Typen sind QSAM, BSAM, BDAM, VSAM und PDS. Auf einer bestimmten Festplatte sind nur Data Sets des gleichen Typs untergebracht. Wenn Sie einen neuen Data Set mittels Allocate anlegen, müssen Sie entscheiden, zu welchen Typ er gehören soll.

Für unseren ersten Versuch (und die meisten folgenden unserer praktischen Übungen) verwenden wir den sogenannten "Partitioned Data Set" (PDS)-Typ.

Das Anlegen eines Partitioned Dataset geschieht mit Hilfe der "Data Set Utility". Wir geben eine "2" auf der Kommandozeile ein und betätigen anschließend die Eingabetaste .



Der Data Set Utility Bildschirm erscheint.

ISPF erwartet den Namen des Datasets, der angelegt werden soll.

Dataset-Namen bestehen häufig aus 3 Feldern und haben das Format xxx.yyy.zzz, wobei xxx, yyy und zzz jeweils Zeichenfolgen mit einer maximalen Länge bis zu 8 Zeichen darstellen. Die drei Felder (oft auch „Qualifier“ genannt) werden mit "Project", "Group" und "Type" bezeichnet und durch Punkte voneinander getrennt.

Es ist üblich, für den "Project"-Teil des Dataset-Namens die Benutzer-ID zu wählen (hier "PRAK025"). Für den z/OS-Rechner der Universität Leipzig (und die meisten anderen z/OS Installationen) ist dies zwingend erforderlich. Der Project-Teil des Dataset-Namens wird auch als „High Level Qualifier“ bezeichnet.

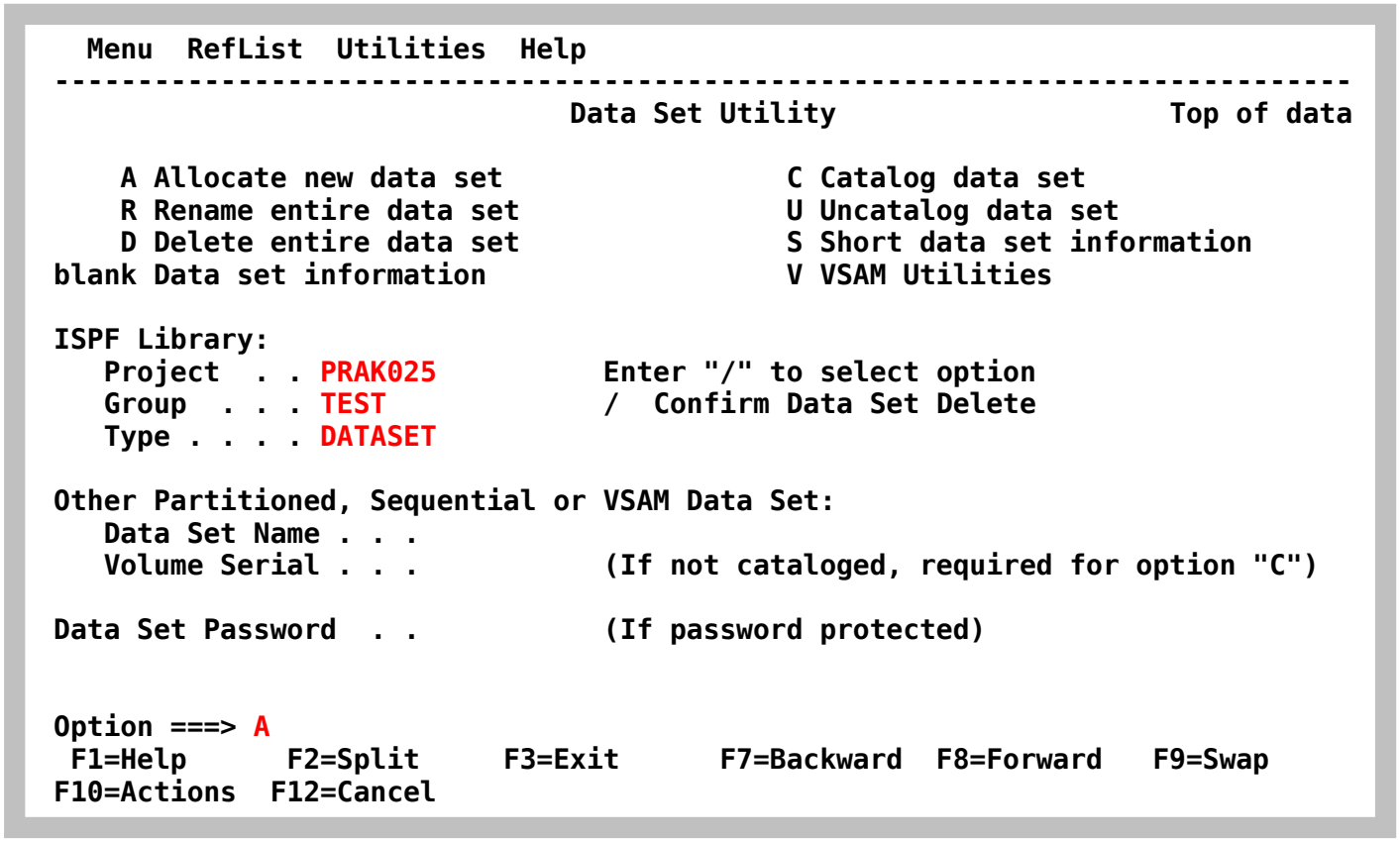
Beispiel für einen Data Set Namen:

xxx.yyy.zzz

Hierbei ist xxx der High Level Qualifier.

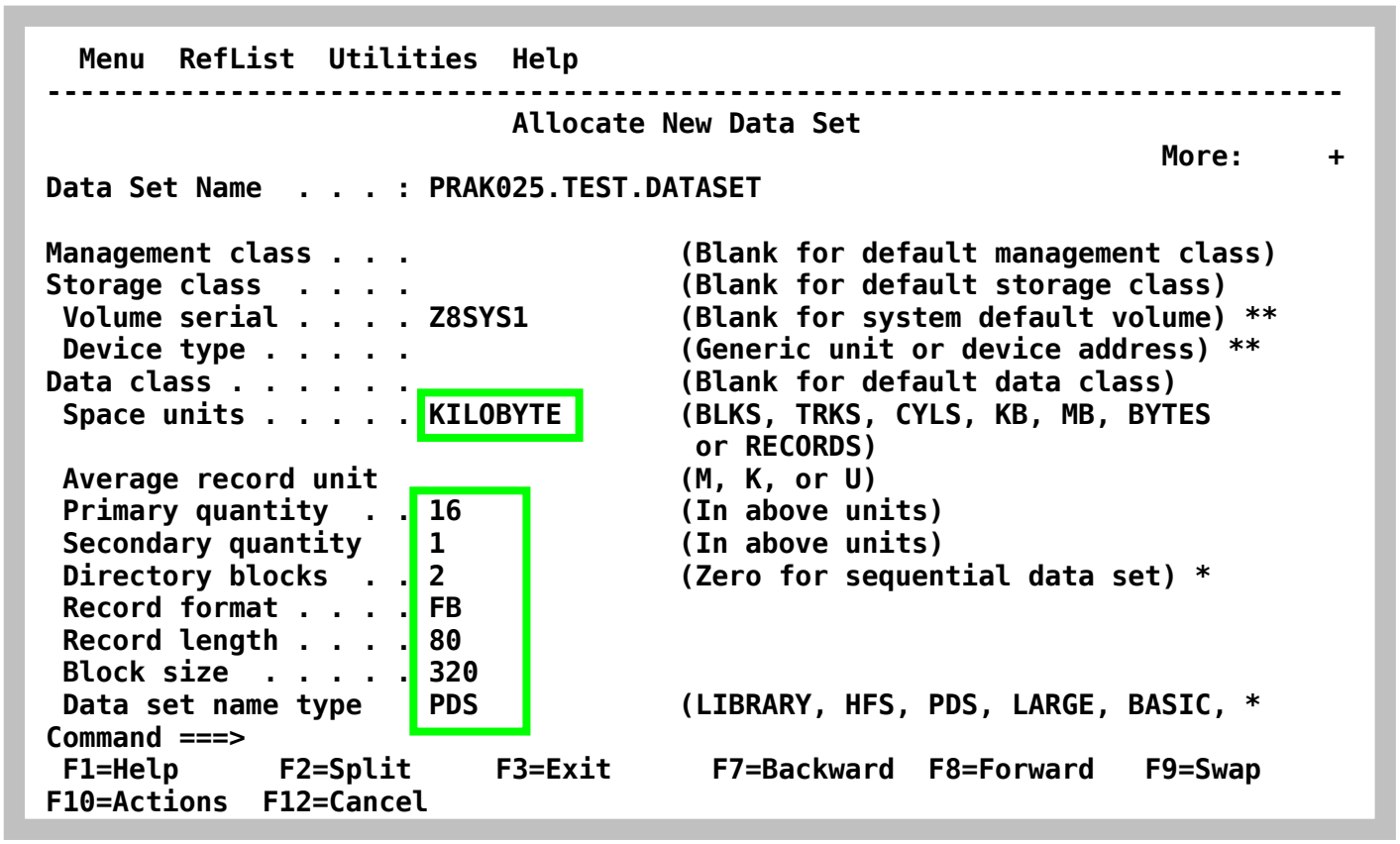
Wir möchten einen Dataset anlegen, den wir anschließend mit Daten beschreiben wollen. Er soll "PRAK025.TEST.DATASET" heißen. Bitte verwenden sie beim Arbeiten mit z/OS immer Ihre User ID an Stelle von PRAK025. Wir geben die drei Namensbestandteile in die dafür vorgesehenen Felder ein (siehe unten).

Um den Dataset "PRAK025.TEST.DATASET" nun anzulegen (allocate), geben wir "A" in die Kommandozeile ein und betätigen die Eingabetaste .



Eingabe “A” im Data Set Utility Bildschirm

z/OS ist ein sehr flexibles System. Die Konsequenz ist, dass der Benutzer mehr Entscheidungen selber treffen muss als dies bei anderen Systemen der Fall ist. Es ist klar, dass TSO bzw. ISPF für unsere Übungsaufgaben hoffnungslos überdimensioniert ist. Schließlich haben wir es hier mit einem vielseitigen Großrechner zu tun, der in der Regel in einer sehr komplexen Systemumgebung eingesetzt wird.



Im „Allocate new Data Set“ Panel muss definiert werden, wie groß unser Dataset sein soll. Wir legen fest, dass alle Größenangaben (Space units) in KILOBYTE (KB) erfolgen sollen. Alternativen wären Blöcke (BLKS), Records, Festplattenspuren (TRKS) oder Festplattenzylinder (CYLS) usw. Wir legen eine Dataset Größe (Primary quantity) von 16 Kilobytes fest und erlauben einen Überlauf (Secondary quantity) von einem weiteren Kilobyte; insgesamt wurden also 17 Kilobytes (16 plus 1 KByte Überlauf) an Festplattenplatz reserviert.

Für Datasets stehen viele alternative Arten zur Verfügung. Der von uns gewünschte Typ "Partitioned Data Set" wird durch eine "2" in der Zeile "Directory blocks" (sowie durch "PDS" in der untersten Zeile "Data set name type") gekennzeichnet.. Es kann auch ein Wert größer als 2 gewählt werden (eine 0 würde einen Typ „Sequential Data Set spezifizieren).

Wir geben für das "Record Format" "FB" (für "Fixed Block") an, für die "Record length" 80 Bytes und für die "Block Size" 27920 Bytes an. Wichtig: Die "Block Size" muss ein ganzzahliges Vielfaches der "Record Length" sein (80 x 4 = 320) !

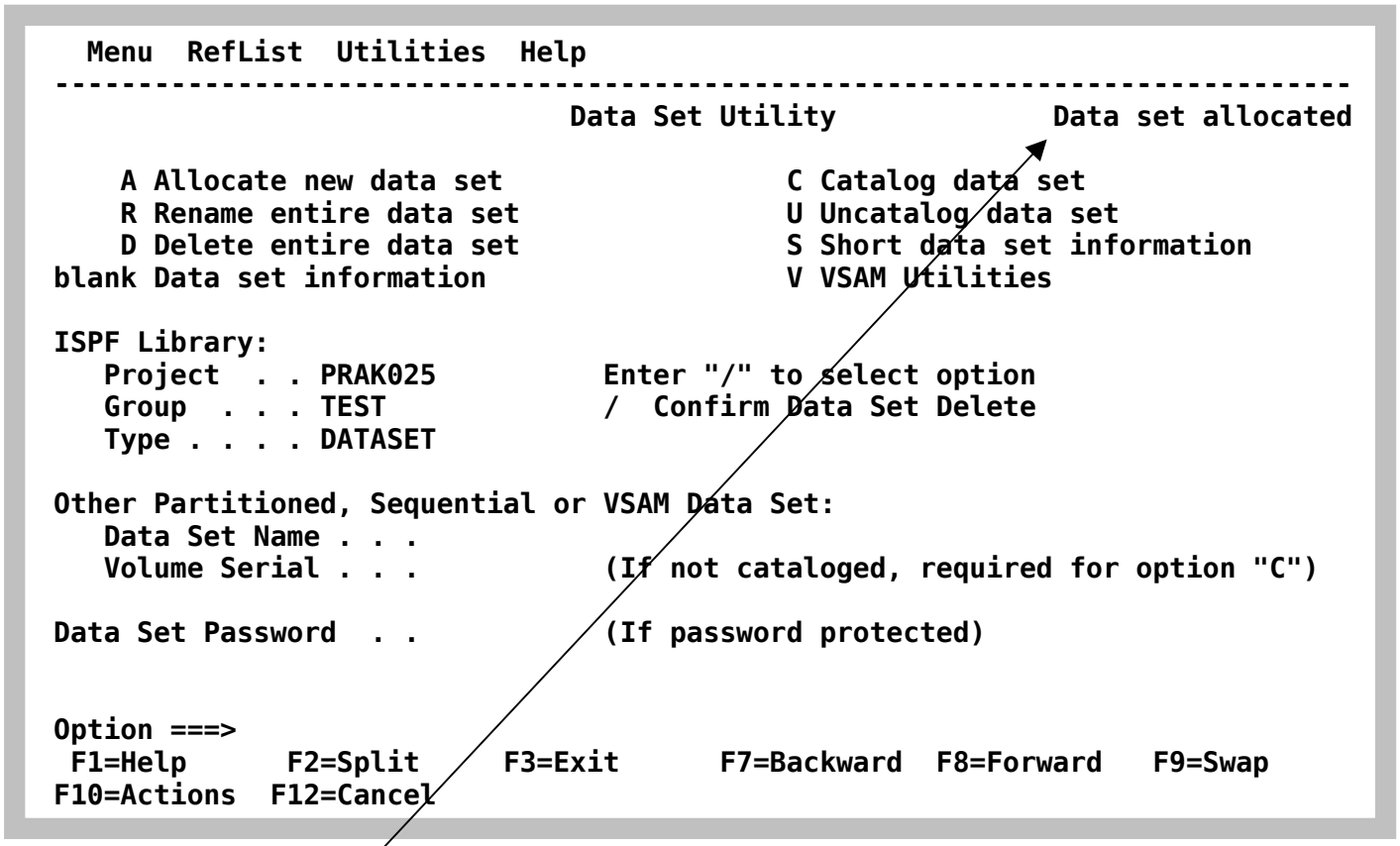
Die Hintergründe für diese Vorgehensweise werden in Band 1, Abschnitt 5.3.4 auf Seite 5-33 erläutert.

Für die restlichen Felder nimmt ISPF Default-Werte an. Es kann sein, dass die Default Werte im „Allocate new Data Set“ Panel nicht angezeigt werden.

```
  Menu  RefList  Utilities  Help
--------------------------------------------------------------------------------
                           Allocate New Data Set
                                                                        More:     +
Data Set Name  . . . : PRAK025.TEST.DATASET

Management class . . .                 (Blank for default management class)
Storage class  . . . .                 (Blank for default storage class)
 Volume serial . . . . Z8SYS1          (Blank for system default volume) **
 Device type . . . . .                 (Generic unit or device address) **
Data class . . . . . .                 (Blank for default data class)
 Space units . . . . . BloCK           (BLKS, TRKS, CYLS, KB, MB, BYTES
                                        or RECORDS)
 Average record unit                   (M, K, or U)
 Primary quantity  . . 1               (In above units)
 Secondary quantity    16               (In above units)
 Directory blocks  . . 20               (Zero for sequential data set) *
 Record format . . . . FB
 Record length . . . . 80
 Block size  . . . . . 27920
 Data set name type    PDS             (LIBRARY, HFS, PDS, LARGE, BASIC, *
Command ===>
 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
F10=Actions  F12=Cancel
```

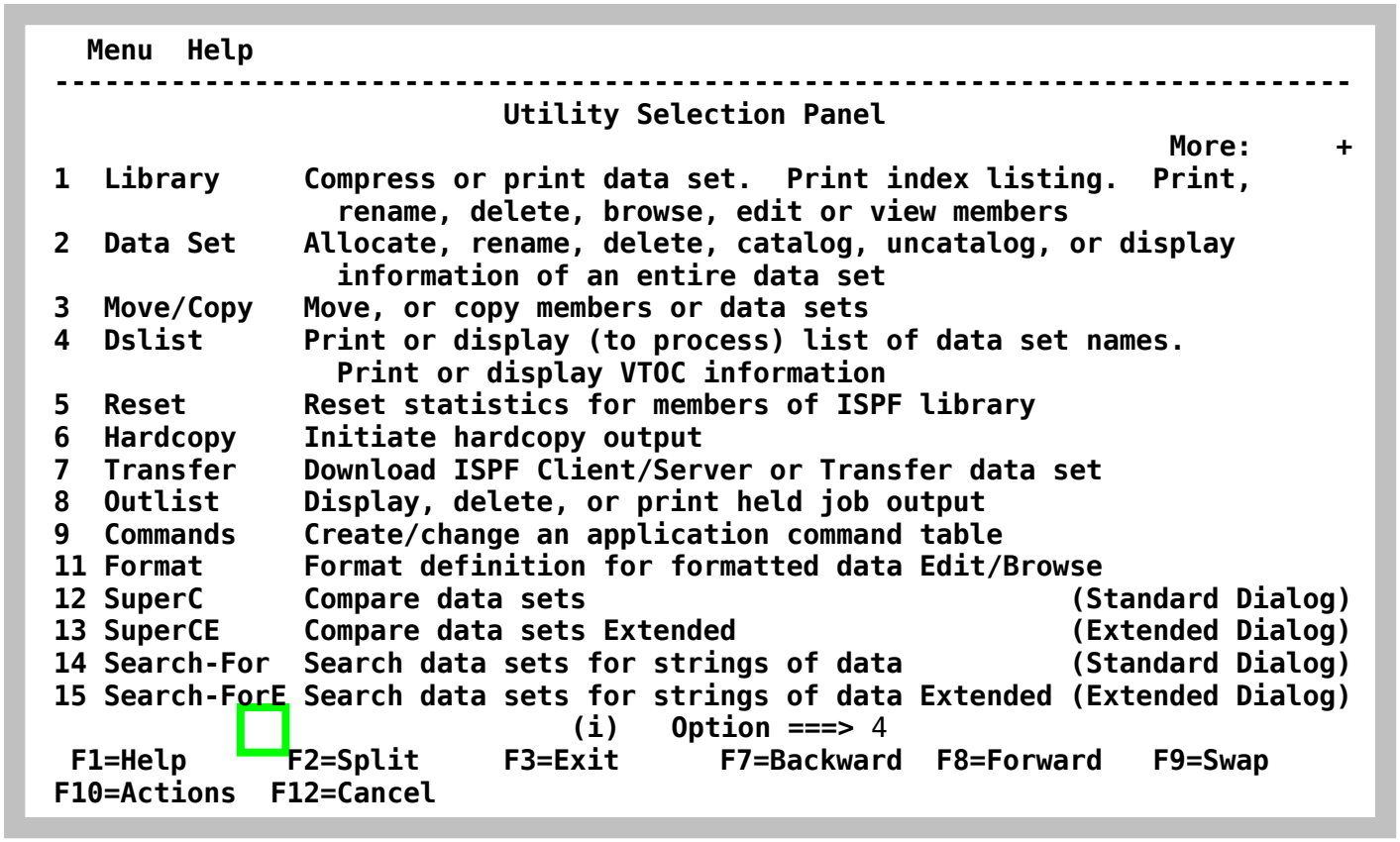
Im Vorlesungsscript gehen wir im Thema Input/Output Teil 3 auf die Ermittelung einer optimalen Block Size ein. Eine Block Größe von 320 Bytes ist ausreichend für unsere Übungsaufgabe, aber sehr ungewöhnlich in der Praxis. Ein mehr professionelles Allocate würde z.B. wie hier dargestellt aussehen.

Enter Taste eingeben

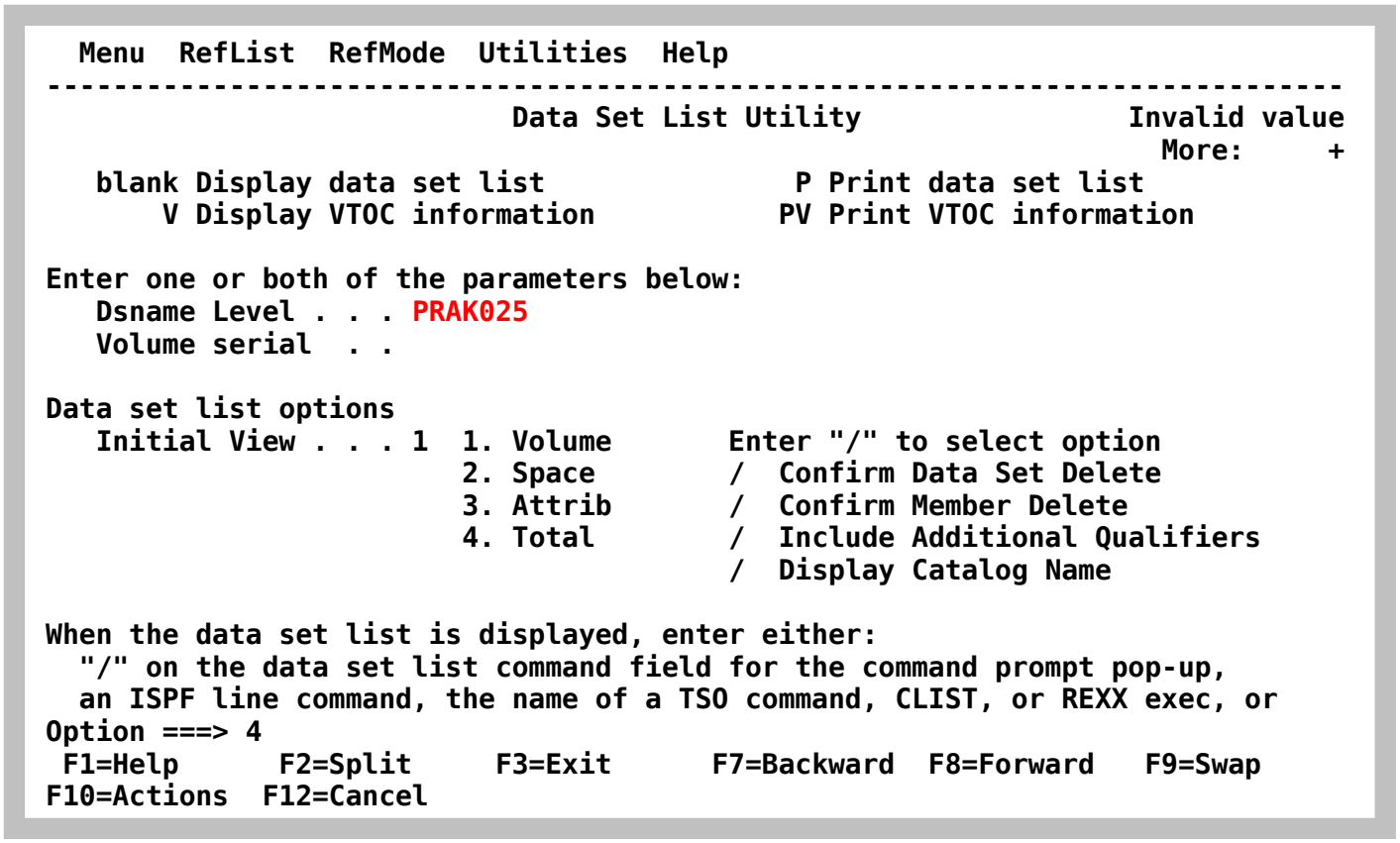


Es erscheint wieder der "Data Set Utility"-Bildschirm. In der rechten oberen Ecke ist die Meldung "Data set allocated" zu sehen. Unser Dataset "PRAK025.TEST.DATASET" wurde damit angelegt und ist nun in der Lage, mehrere Files (Members) aufzunehmen.

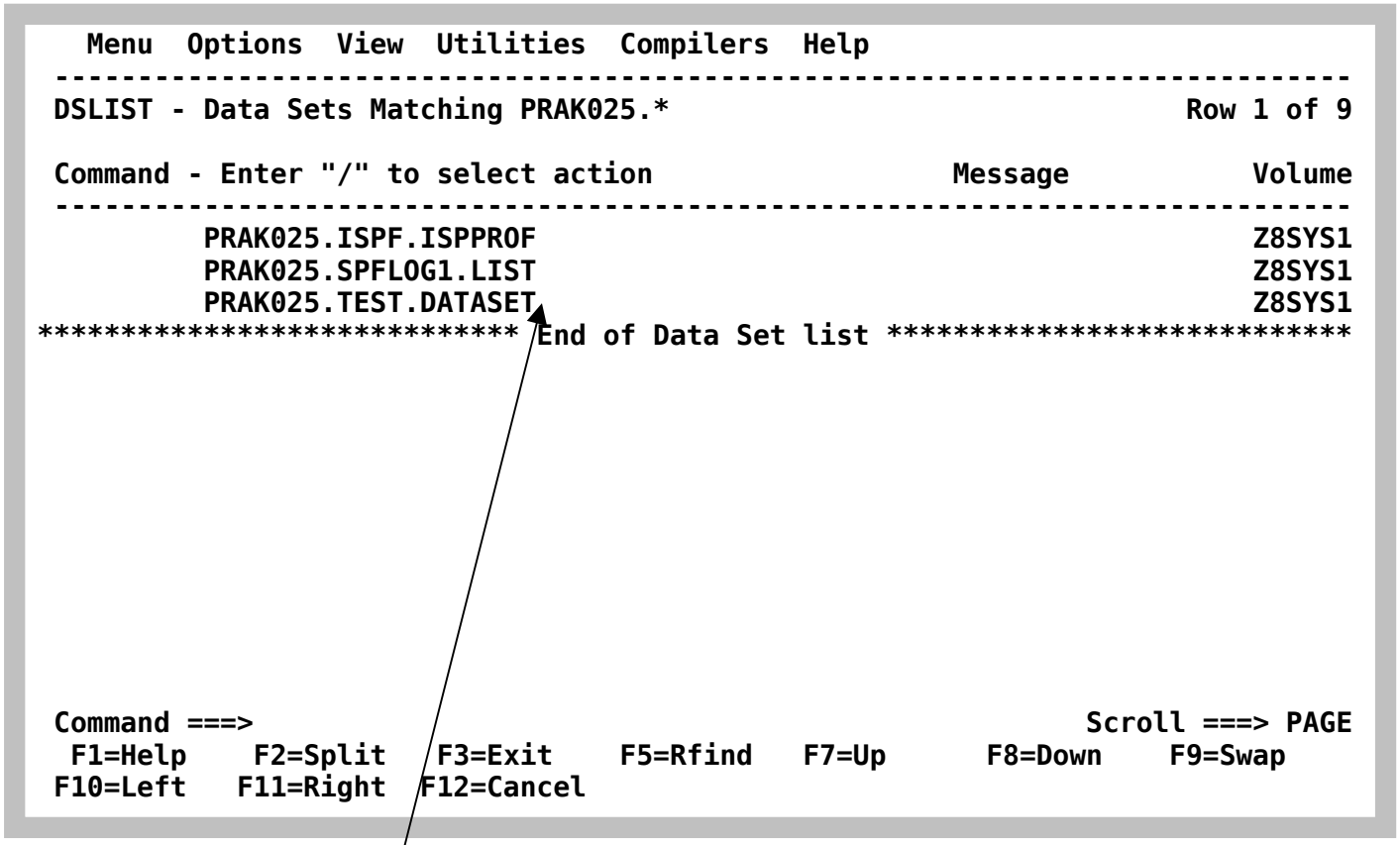
F3 Taste betätigen



Dies bringt uns zurück zum "Utility Selection Panel"-Bildschirm. Wir wollen uns nun das Ergebnis unserer Arbeit ansehen. Dies geschieht mit dem Dslist ("Data Set List")- Kommando. Wir geben eine "4" in die Kommandozeile ein und betätigen die Eingabetaste.

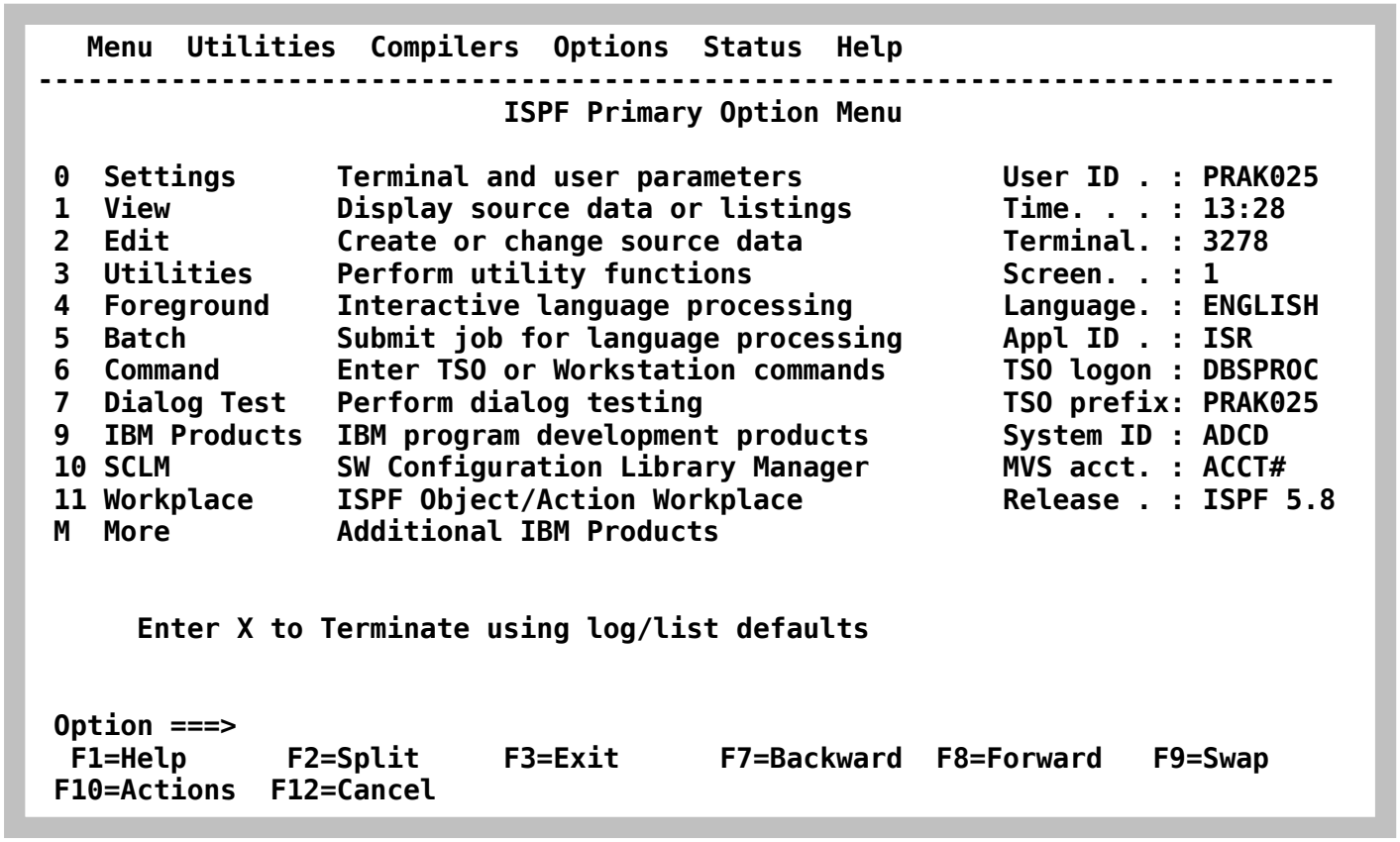


Das "Data Set List Utility" kann für viele Informationsabfragen benutzt werden. Der High Level Qualifier des Data Set Namens (hier PRAK025) wird als Default Wert vorgeschlagen. Wir betätigen die Eingabetaste.



Dies ist das Ergebnis: Beim erstmaligen Einloggen hat TSO selbständig und standardmäßig die beiden Datasets "PRAK025.ISPF.ISPPROF" und "PRAK025.SPFLOG1.LIST" angelegt. Der Dataset "PRAK025.TEST.DATASET" ist von uns angelegt worden.

Dreimaliges Betätigen der F3-Taste bringt uns zurück zum "ISPF Primary Option Menu" Screen.



<u>Aufgabe:</u> Legen Sie in Ihrer Group "TEST" einen neuen Partitioned Dataset an (mit den gleichen Parametern wie im Tutorial). Bezeichnen Sie den „Type“statt mit „DATASET“diesmal mit "CNTL" . Verifizieren Sie, dass alles richtig geklappt hat. Danach mit wiederholter Betätigung der F3 Taste zurück in das Primary Option Menu.

5. Logoff Prozess

```
(b)                                                Specify Disposition of Log Data
Set
                                                                          More:     +
 Log Data Set (PRAK025.SPFLOG1.LIST) Disposition:
 Process Option . . . . 3  1. Print data set and delete
                           2. Delete data set without printing
                           3. Keep data set - Same
                              (allocate same data set in next session)
                           4. Keep data set - New
                              (allocate new data set in next session)
 Batch SYSOUT class . .
 Local printer ID or
 writer-name  . . . . .
 Local SYSOUT class . .

 List Data Set Options not available

 Press ENTER key to complete ISPF termination.
 Enter END command to return to the primary option menu.

 Job statement information:  (Required for system printer)
  ===>
 Command ===>
  F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
 F12=Cancel
```

Im ISPF Primary Option Menu die F3 Taste betätigen. Es erscheint das hier gezeigte Panel. Eine „3“ auf der Cursor Position in der Zeile „Process Option“ eingeben.

Die Eingabe der Ziffer 3 wählt die Option „ Keep data set - Same (allocate same data set in next session) “ aus. TSO will wissen, was mit den drei Partitioned Datasets geschehen soll, die wir in dieser Sitzung angelegt haben.

Dies ist ein sehr wichtiger Schritt !!!!!!!!! Sie sollten nie Ihre TSO Session ohne diesen Schritt beenden. Wenn sie es trotzdem tun, kann dies interessante Konsequenzen haben !!!!!!!

Die Eingabe der Ziffer 3 bedeutet, dass alle Daten, die Sie in Ihrer TSO Session erarbeitet haben, permanent gespeichert werden und in Ihrer nächsten Session wieder verfügbar sind. Es wäre auch ok, eine der anderen Optionen 1, 2 oder 4 auszuwählen. Dies wird bei Ihren Sitzungen mit TSO nur sehr selten oder nie vorkommen. Ohne diese (oder eine andere) Eingabe wird Ihnen das Logoff verweigert. TSO hat es gar nicht gerne, wenn Sie sich ohne ordnungsgemäßes Logoff verabschieden.

Es ist sehr wichtig, dass Sie den hier beschriebenen Logoff Prozess bei jeder TSO Sitzung strikt einhalten.

Logoff ist ein Kommando der TSO Shell. Andere Betriebssysteme benutzen häufig die Bezeichnung (Syntax) Logout

Wir geben in das Eingabefeld auf der "Process option"-Zeile eine "3" ein, gefolgt von der Eingabetaste.

In manchen Fällen ist hier ein shortcut eingebaut. Wenn Sie im ISPF Primary Options Menue die F3 Taste betätigen, erfolgt automatisch ein Logoff. Sie brauchen hier nichts weiteres zu tun.

```
 PRAK025.SPFLOG1.LIST has been kept.
READY
```

Nach Eingabe der Ziffer 3 und der Betätigung der Enter Taste erscheint der hier gezeigte Bildschirm. Wir haben das ISPF-Subsystem verlassen und sind zurück im TSO-Linemode-Subsystem .

Die Meldung

```
" PRAK025.SPFLOG1.LIST has been kept.
 READY"
```

erscheint. Wir erinnern uns: "PRAK025.SPFLOG1.LIST" war einer der beiden Datasets, die das System für uns angelegt hatte.

Wir geben "logoff" auf der Cursor Position ein und betätigen die Eingabetaste. Je nach verwendetem 3270 Emulator erscheint der Welcome-Bildschirm des z/OS-Rechners wieder, oder der Bildschirm wird schwarz (leer) . Wir können die Verbindung jetzt trennen, indem wir den 3270 Emulator schließen..

Das war es. Sie haben Ihre erste TSO Session erfolgreich abgeschlossen.

3. Anhang

Quick3270 Tastaturbelegung

Die durch den 3270 Emulator wiedergegebene Bildschirmdarstellung geht auf den ursprünglich 1971 herausgebrachten IBM 3278 Bildschirmterminal zurück. Dieses Gerät hatte für eine Reihe von Tasten eine andere Tastaturbelegung als es heute beim PC üblich ist. Jeder 3270 Emulator verfügt deshalb über eine „Keyboard MAP“ Funktion, welche diese Tasten dem Benutzer verfügbar macht.

Quick3270 verwendet die folgende standardmäßige Tastaturbelegung:

3270 Tastatur	Neubelegbar	PC - Taste
CURSOR UP	Nein	Nach Oben
CURSOR DOWN	Nein	Nach Unten
CURSOR LEFT	Nein	Nach Links
CURSOR RIGHT	Nein	Nach Rechts
FAST CURSOR LEFT	Ja	Alt + Nach Links
FAST CURSOR RIGHT	Ja	Alt + Nach Rechts
BACK WORD	Ja	Strg + Left Arrow
FORWARD WORD	Ja	Strg + Right Arrow
INSERT	Ja	Einfg
DELETE (Löschtaste)	Ja	Löschen / Entfernen
ERASE TO END OF FIELD	Ja	Ende
ERASE INPUT	Ja	Alt + Ende
HOME	Ja	Pos1
TAB	Ja	Tab
BACKTAB	Ja	Umschalt + Tab
ENTER	Ja	Eingabetaste
NEW LINE (Neue-Zeile-Taste)	Ja	Strg + Eingabetaste oder
	Nein	Rechte Strg-Taste
RESET (Grundstellungstaste)	Ja	Esc
CLEAR	Ja	Pause
ATTENTION	Ja	Umschalt + Esc
SYSTEM REQUEST	Ja	Alt + F11
FIELDMARK	Ja	Umschalt + Pos1
DUP	Ja	Umschalt + Ende
Cent Buchstabe ¢	Nein	Alt + C
PA1	Ja	Bild Nach Oben
PA2	Ja	Bild Nach Unten
PA3	Ja	Umschalt + Bild Nach Unten
PF1 - PF12	Ja	F1 - F12
PF13 - PF24	Ja	Umschalt + F1 - Umschalt + F12

Häufig verfügt ein 3270 Emulator über eine Einrichtung, welche es ermöglicht, diese Abbildung (Mapping) individuellen Wünschen anzupassen.

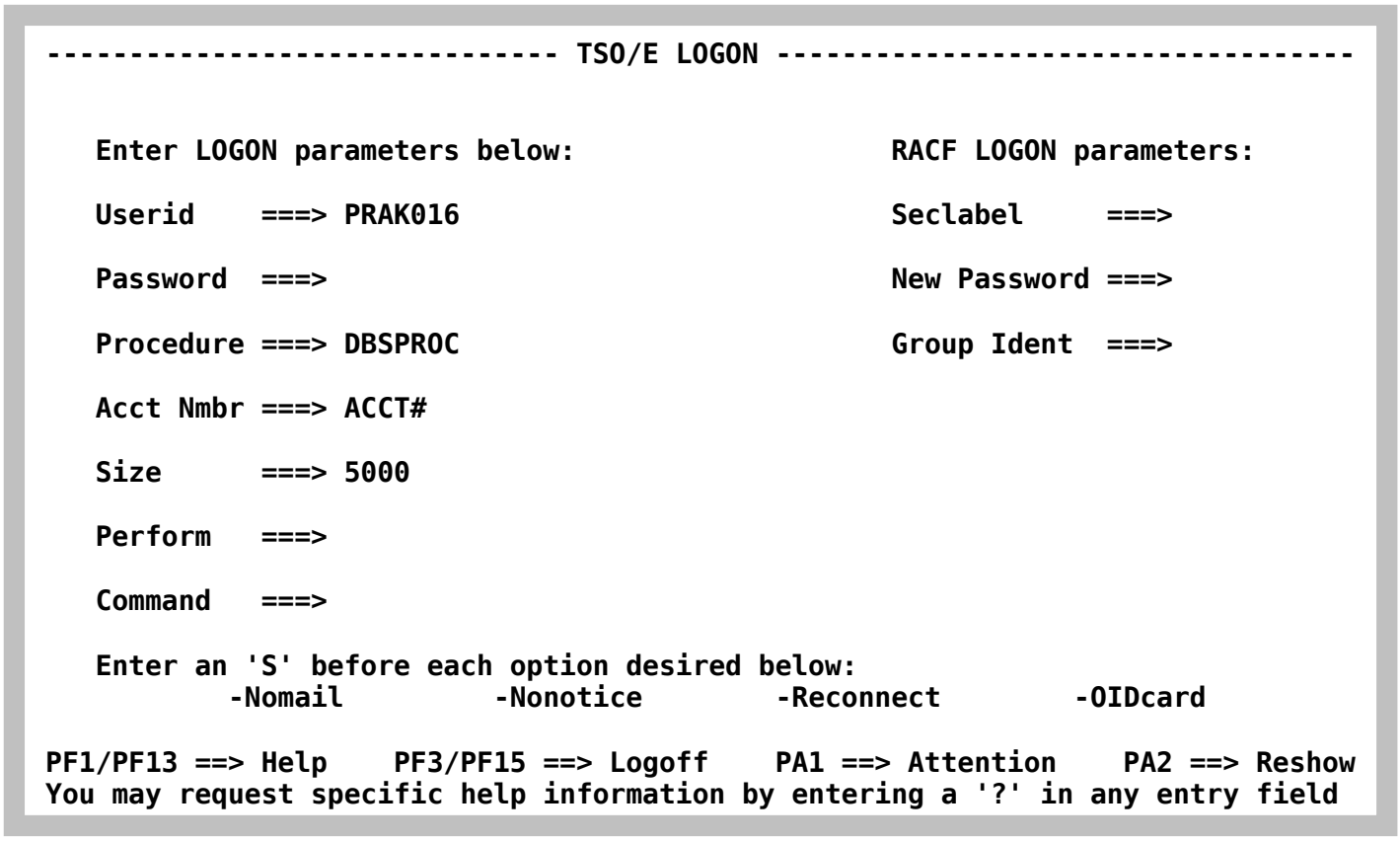
Um bei Quick3270 die Tastaturbelegung zu verändern, wählen Sie Einstellen dann Tastaturbelegung.

Mit diese Option können Sie die Tastaturbelegung einstellen. Die Änderungen werden in der Sitzung Konfigurationsdatei gespeichert. Es ist auch möglich die Konfiguration in eine Tastaturbelegungsdatei zu speichern. Somit ist es möglich die Tastaturbelegung in einer andere Sitzung wiederzuverwenden.

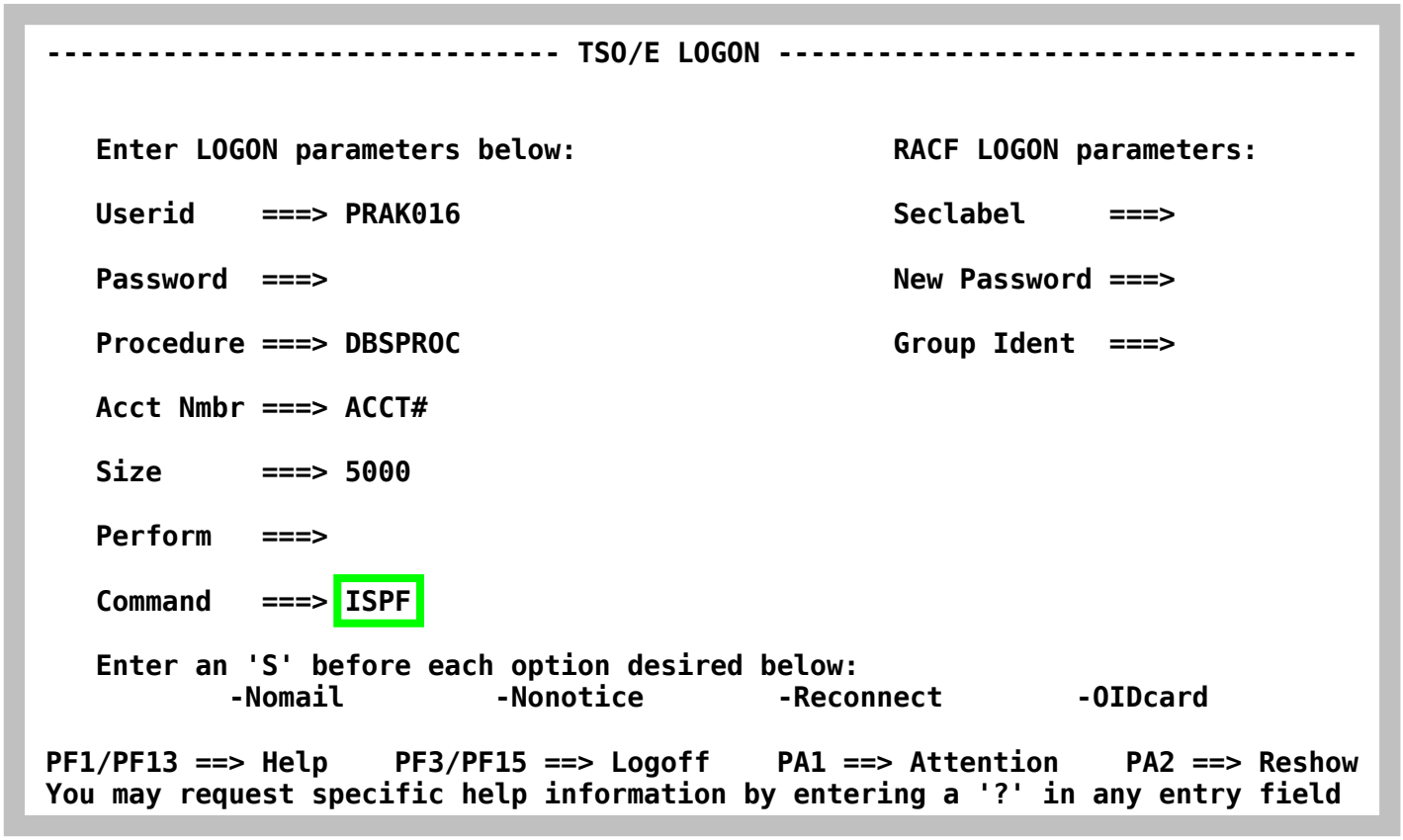
Einige Windows Funktionstasten können nicht für 3270 Funktionen wiederverwendet werden (Alt+Tab, Alt+F4, Alt+Esc...).
Achten Sie darauf, dass nicht eine Tastenkombination für mehrere 3270 Funktionen verwenden werden. Diese Version von Quick3270 prüft nicht die Gültigkeit der Tastaturbelegung.

14) Anhang B

Diese Anweisung beschreibt, wie Sie sicherstellen können, dass sie beim Einloggen sofort in das ISPF Subsystem kommen.

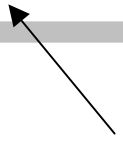


Das nächste Mal, wenn sie sich einloggen, und der Logon Screen erscheint, dann...



geben Sie zusätzlich zu Ihrem Passwort auf der Zeile `Command ===>` den Wert ISPF wie gezeigt ein. Nachdem Sie Enter gedrückt haben, wird das TSO Kommando „ISPF“ automatisch ausgeführt, und.....

```
(a)                          ICH70001I PRAK016  LAST ACCESS AT 09:19:56 ON SUNDAY,
NOVEMBER 9, 2008
 IKJ56455I PRAK016 LOGON IN PROGRESS AT 09:55:22 ON NOVEMBER 9, 2008
 IKJ56951I NO BROADCAST MESSAGES
 ********************************************************************
 *                                                                  *
 * APPLICATION DEVELOPER'S CONTROLLED DISTRIBUTION (ADCD)           *
 *                                                                  *
 *  ADCD.Z18.CLIST(ISPFCL) PRODUCES THIS MESSAGE                    *
 *  ADCD.* DATASETS CONTAIN SYSTEM CUSTOMIZATION                    *
 *  SMP/E DATASETS CAN BE LOCATED FROM 3.4 WITH DSNAME **.CSI       *
 *  HTTP://DTSC.DFW.IBM.COM/ADCD.HTML CONTAINS DOCUMENTATION        *
 *                                                                  *
 *  USERID              PASSWORD        COMMENT                     *
 *  ----------------    ------------ --------------                 *
 *  IBMUSER             - SYS1/IBMUSER FULL AUTHORITY               *
 *  ADCDMST             - ADCDMST      FULL AUTHORITY               *
 *  ADCDA THRU ADCDZ    - TEST         LIMITED AUTHORITY(NO OMVS)*
 *  OPEN1 THRU OPEN3    - SYS1         UID(0) (NO TSO)              *
 *                                                                  *
 ********************************************************************

 ISPF
 ***
```



die Anzeige „*“zeigt an, dass das ISPF Kommando ausgeführt wurde. Nochmals Enter, und Sie landen im ISPF Primary Options Menue.**

1.b ISPF Editor

In diesem Tutorial erwerben Sie Grundkenntnisse für den Umgang mit dem ISPF Editor.

Inhalt

Z/

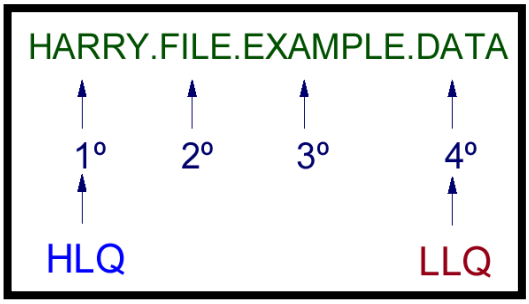
1. Datasets

1.1 Dataset-Namen

Jeder Dataset benötigt einen Namen, mit dem er gespeichert und verarbeitet werden kann.

Regeln:

- **Ein Name besteht aus mehreren Bestandteilen (Qualifiern), die jeweils durch einen Punkt miteinander verbunden sind.**
- **Der erste Qualifier des Namens ist der „High Level Qualifier“ (HLQ).**
- **Der letzte Qualifier des Namens ist der „Low Level Qualifier“ (LLQ).**
- **Jeder Qualifier besteht aus 1 - 8 Zeichen, wobei Buchstaben, Ziffern und die drei Zeichen #, § und $ erlaubt sind. Der Qualifier darf nicht mit einer Ziffer beginnen.**
- **Die Gesamtlänge des Namens ist maximal 44 Zeichen , dabei zählen die Punkte mit.**
- **Für die Namen der Member in einem Partitioned Dataset (siehe unten) gelten die gleichen Regeln wie für einen Qualifier.**



Gültige Dataset-Namen sind:

USER1.TEST.DATA
SYS1.PARMLIB
MAX.PROGRAM.VERSION1

Ungültige Dataset-Namen sind:

USER3-X.BEISPIEL.LIST	**(Bindestrich nicht erlaubt)**
8TEST.LISTE	**(Ziffer am Anfang)**
EMIL.TESTPROGRAMM1	**(Qualifier > 8 Zeichen)**

P390A.A.B.C.D.E.F.G.H.I.J.K.L.M.N.O.P.Q.R.S

ist ein gültiger Data Set Name (Großbuchstaben, nicht mehr als 44 Bytes, keine Spezialzeichen) aber ist nicht sehr sinnvoll.

Es ist üblich, dass Datasets 2, 3 oder 4 Qualifier haben, wobei 3 Qualifier überwiegen. Beim Anlegen (allocate) eines Data Sets ist es meistens erforderlich, dass der HLQ identisch mit Ihrer User ID ist.

1.2 Partitioned Data Set (PDS)

Ein Partitioned Dataset (PDS) ist eine Art Mini-Unix File-System. Es verfügt über ein einfaches Inhaltsverzeichnis (Directory) und Platz für mehrere Dateien, die als „Member“ bezeichnet werden. Jeder einzelne Member des Datasets kann ausgewählt, geändert oder gelöscht werden, ohne die anderen Member zu beeinflussen. Alle Member befinden sich auf der gleichen Ebene; es besteht keine hierarchische Beziehung der Member zueinander.

Ein "Member", wird in sequentieller Reihenfolge geschrieben, und es wird jedem Member ein NAME im PDS Inhaltsverzeichnis zugeordnet.

Ein Member ist entweder eine strukturlose Zeichenkette (Byte Stream) ähnlich einer Unix File, oder kann alternativ aus einzelnen Records bestehen.

Das Inhaltsverzeichnis (Directory) ist ein Index, der vom Betriebssystem verwendet wird, um ein Member innerhalb eines PDS zu lokalisieren. Alle Einträge (Namen der Member) im Inhaltsverzeichnis sind in alphabetisch aufsteigende Reihenfolge angeordnet. Die einzelnen Member können jedoch in jeder beliebigen Reihenfolge innerhalb des Partitioned Dataset gespeichert werden.

Ein Member in einem Partitioned Dataset (PDS) wird folgendermaßen angesprochen:

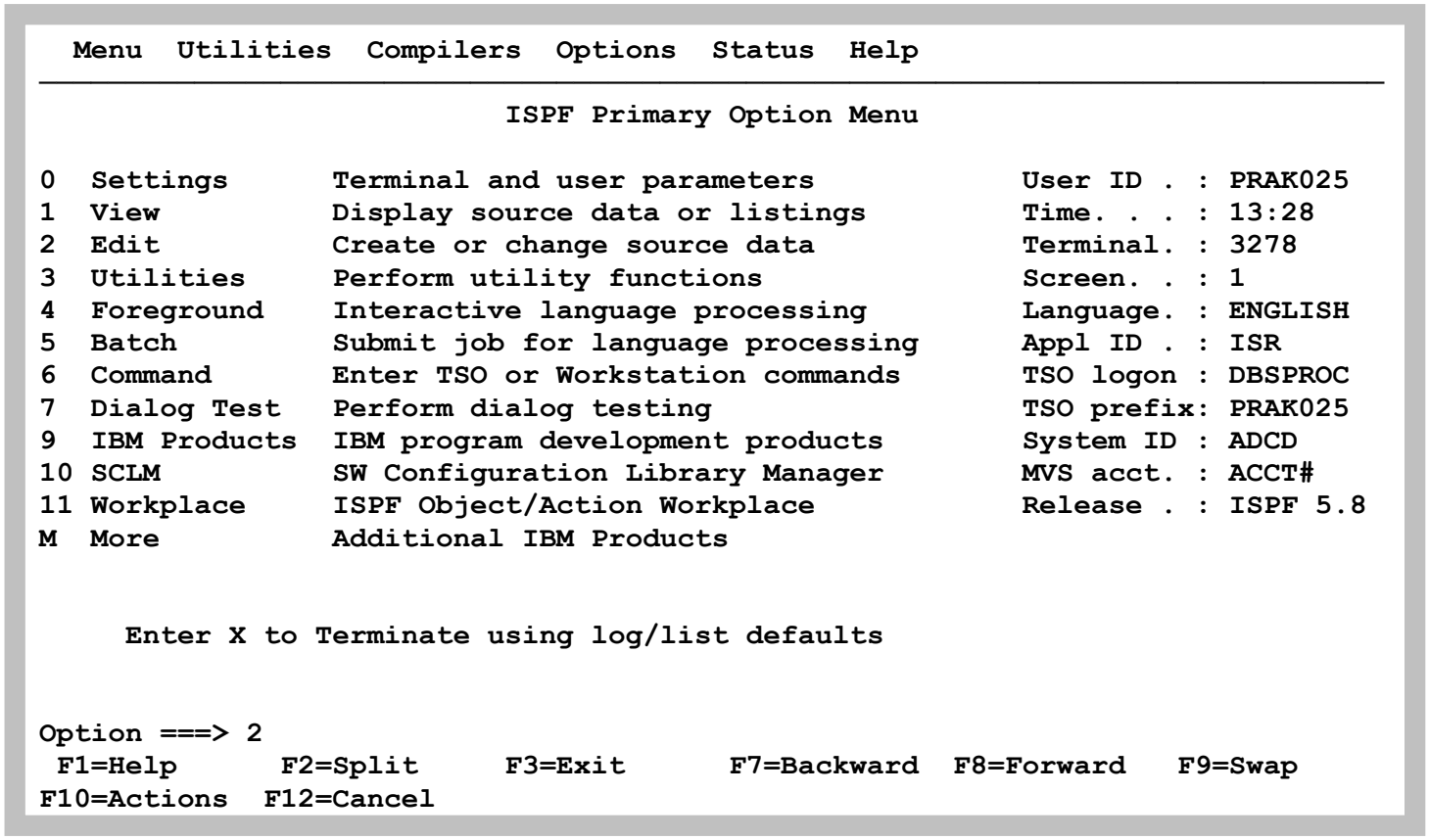
USER1.PROGRAM.LIBRARY(PROG1)

Wobei USER1 der HLQ, Library der LLQ und PROG1 der Member Name ist.

Die offizielle Bezeichnung ist Partitioned Data Set extended (PDSe), wobei PDSe eine Weiterentwicklung des ursprünglichen Partitioned Data Set (PDS) ist. Letzterer ist jedoch fast ausgestorben, weshalb man heute unter PDS fast immer einen PDSe versteht.

In den praktischen Übungen dieses Lehrgangs werden wir fast ausschließlich mit Partitioned Data Sets arbeiten.

Häufig besteht ein neu entwickeltes Cobol Programm aus mehreren Unterprogrammen U1, U2 und U3 die auch einzeln übersetzt werden. Jetzt kann man z.B. alle Quellprogramme in einem einzigen PDS Data Set als USER1,TEST,COBOL(U1), USER1,TEST,COBOL(U2) und, USER1,TEST,COBOL(U3), abspeichern, und die entsprechenden übersetzten Programme in einem zweiten PDS Data Set als USER1,TEST,OBJECT(U1), USER1,TEST, OBJECT(U2) und USER1,TEST, OBJECT (U3), abspeichern

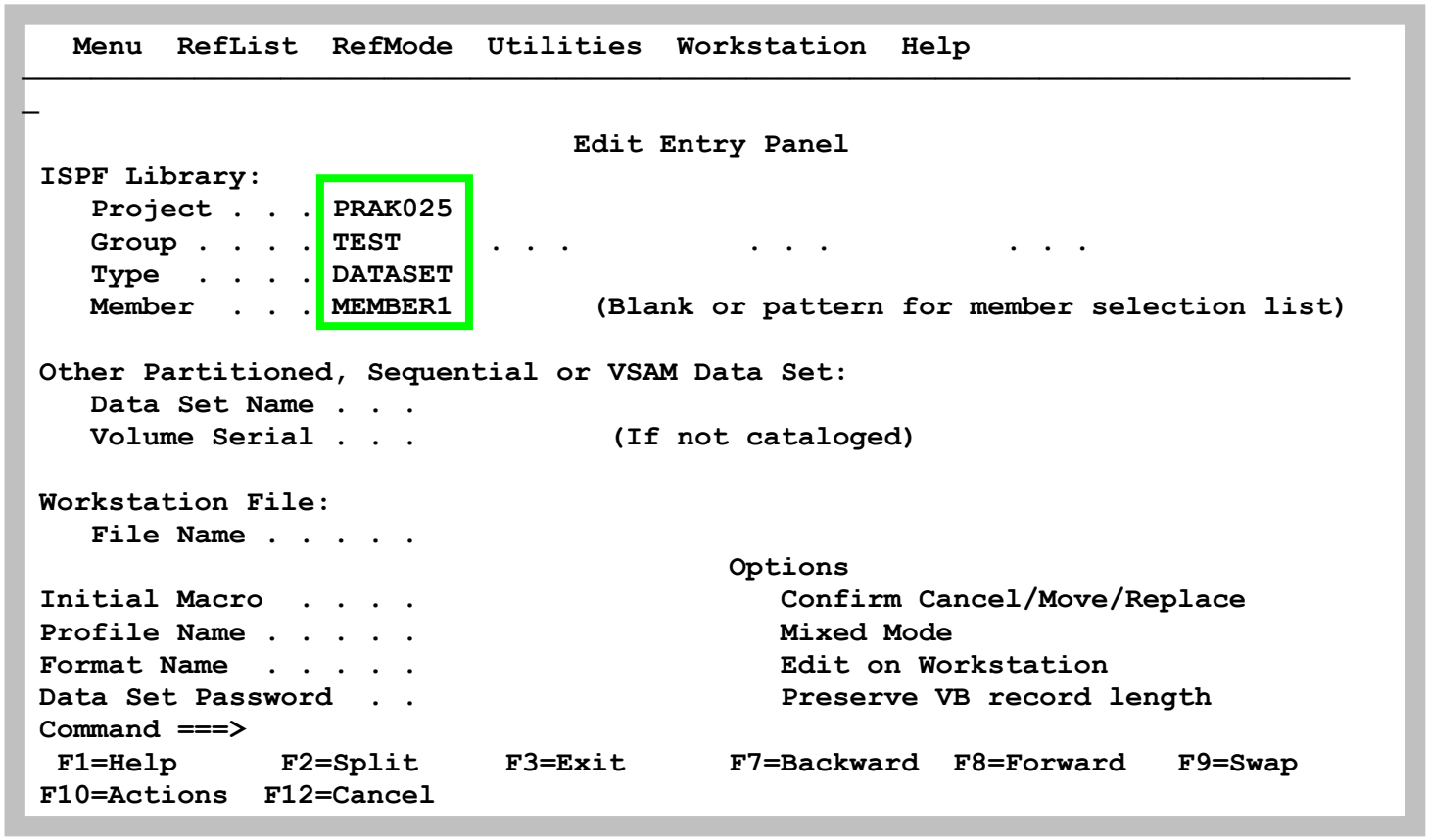


Wir wollen nun unserem angelegten Partitional Dataset einen Member zuordnen und diesen mit Daten füllen. Dies geschieht mit Hilfe der "Edit"-Funktion. Im ISPF Primary Options Menue geben wir eine "2" in die Kommandozeile ein und betätigen die Eingabetaste.

```
  Menu  RefList  RefMode  Utilities  Workstation  Help
──────────────────────────────────────────────────────────────────────────────
                              Edit Entry Panel

ISPF Library:
   Project . . . PRAK025
   Group . . . . TEST     . . .          . . .          . . .
   Type  . . . . DATASET
   Member  . . .                 (Blank or pattern for member selection list)

Other Partitioned, Sequential or VSAM Data Set:
   Data Set Name . . .
   Volume Serial . . .           (If not cataloged)

Workstation File:
   File Name . . . . .
                                       Options
Initial Macro  . . . .                    Confirm Cancel/Move/Replace
Profile Name . . . . .                    Mixed Mode
Format Name  . . . . .                    Edit on Workstation
Data Set Password  . .                    Preserve VB record length
Command ===>
 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
F10=Actions  F12=Cancel
```

Der "Edit Entry Panel"-Bildschirm erscheint und fordert uns auf, den Namen des zu editierenden Programms einzugeben.

Zuvor müssen wir dem zu erstellenden Member einen Namen geben, wir wählen "MEMBER1". Der volle Name dieses Members ist "PRAK025.TEST.DATASET(MEMBER1)".

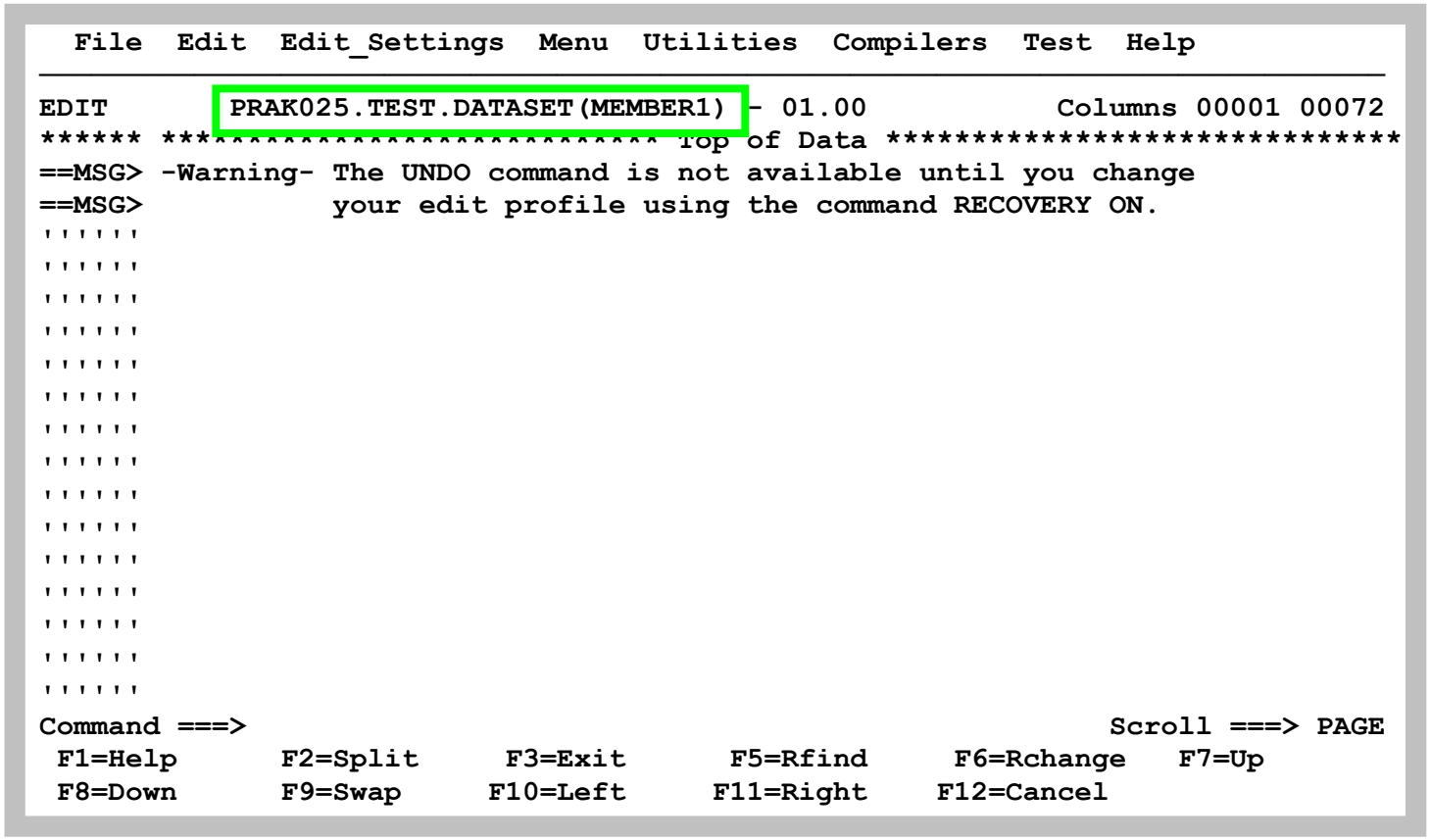


Die drei Teile des Dataset-Namens sind jetzt in die Felder "Project", "Group" und "Type" einzutragen. Häufig stehen in diesen Feldern schon Werte drin. Korrekte Werte können unverändert übernommen werden, falsche Werte müssen mit den richtigen überschrieben werden.

Der zu erstellende Member soll den Namen "MEMBER1" erhalten. Dieser Name ist in das Feld "Member" einzutragen.

Wir geben für "*Project*" die Benutzer-ID, für "*Group*" den Wert "TEST", für das Feld "*Type*" den Wert "DATASET" sowie für das Feld "*Member*" den Wert "MEMBER1" ein.

und betätigen anschließend die Eingabetaste



Das Betätigen der Eingabetaste legt diesen Member an. Der ISPF-Editor wird aufgerufen und öffnet ein Fenster zum Editieren des noch leeren Members .

Es wird "PRAK025.TEST.DATASET(MEMBER1" angezeigt. Dies ist die Bestätigung dafür, dass im Dataset PRAKT20.ISPF.TEST1 der Member MEMBER1 erfolgreich angelegt wurde.

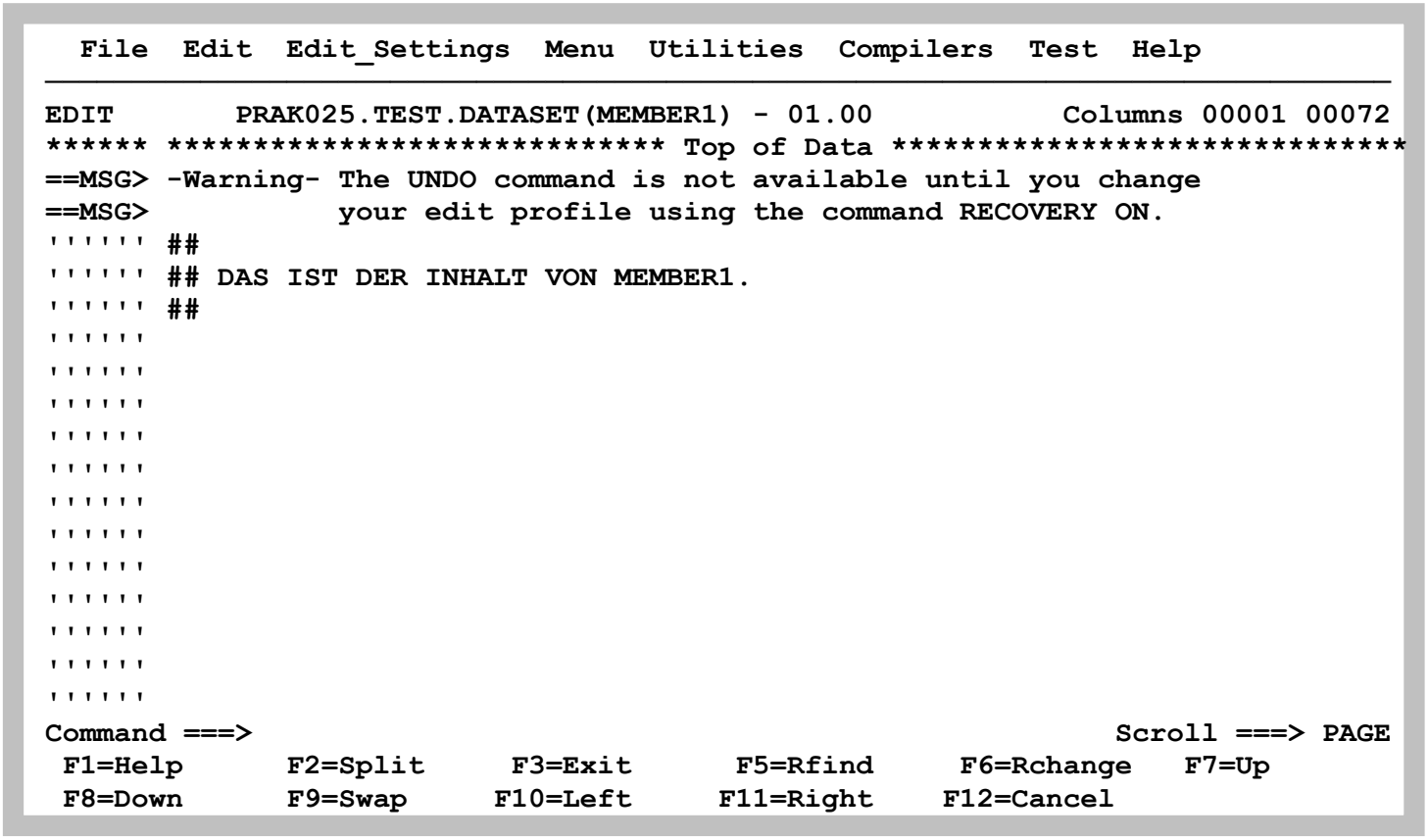
2. Benutzung des ISPF Editors

Von der Philosophie her ist der ISPF Editor dem Unix-vi-Editor ähnlich.

Für Unix existieren viele Editoren. Der vi-Editor ist sehr leistungsfähig, kryptisch, unlogisch, nicht intuitiv und schwer zu erlernen. Jedes Unix-System verfügt über einen vi-Editor. Experten schwören auf ihn. Für den Neuling ist er ein Horror.

Der ISPF-Editor ist auch sehr leistungsfähig, kryptisch, unlogisch, nicht intuitiv und schwer zu erlernen. Jedes z/OS-System verfügt über einen ISPF-Editor. Experten schwören auf ihn. Für den Neuling ist er gleichfalls ein Horror.

Sorry, sie haben keine Wahl. Sie müssen den Umgang mit dem ISPF Editor lernen. Für den Anfang benutzen Sie wie gewohnt die notwendigen alphanumerischen Tasten für die Texteingabe.



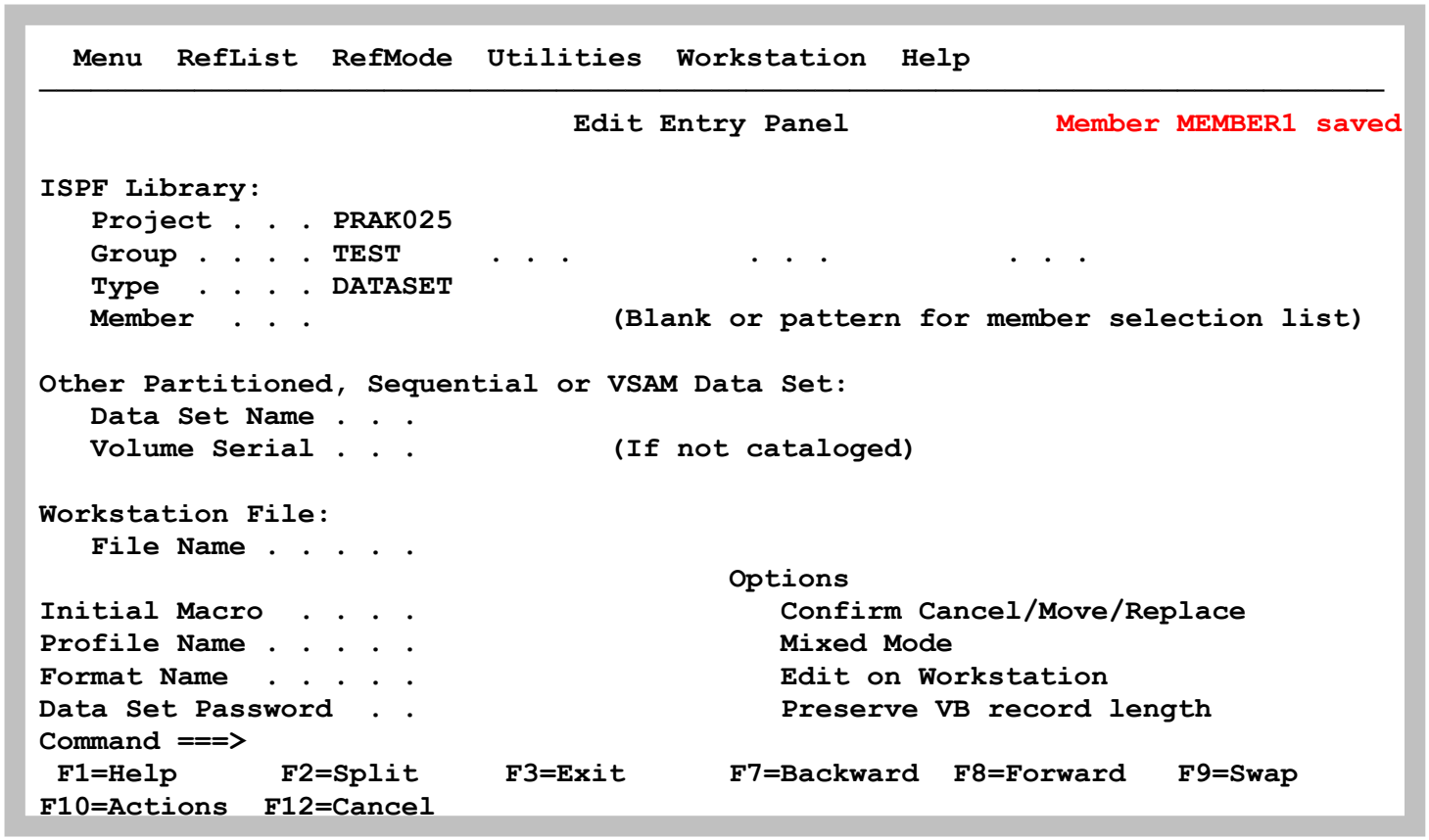
Wir denken uns einen kurzen Text aus und füllen den Member "MEMBER1" mit diesem Text. Durch Betätigen der F3-Taste kehren wir zum vorherigen Bildschirm zurück. Unser Dataset wird dadurch automatisch gespeichert (saved).

Beim Niederschreiben von längeren Texten empfiehlt es sich, den Stand der Arbeit in bestimmten Zeitabständen auf die Festplatten zu sichern. Dazu gibt man auf der Kommandozeile (hinter dem `===>` Symbol) einfach "`SAVE`" ein. Ein erfolgreiches Sichern wird mit

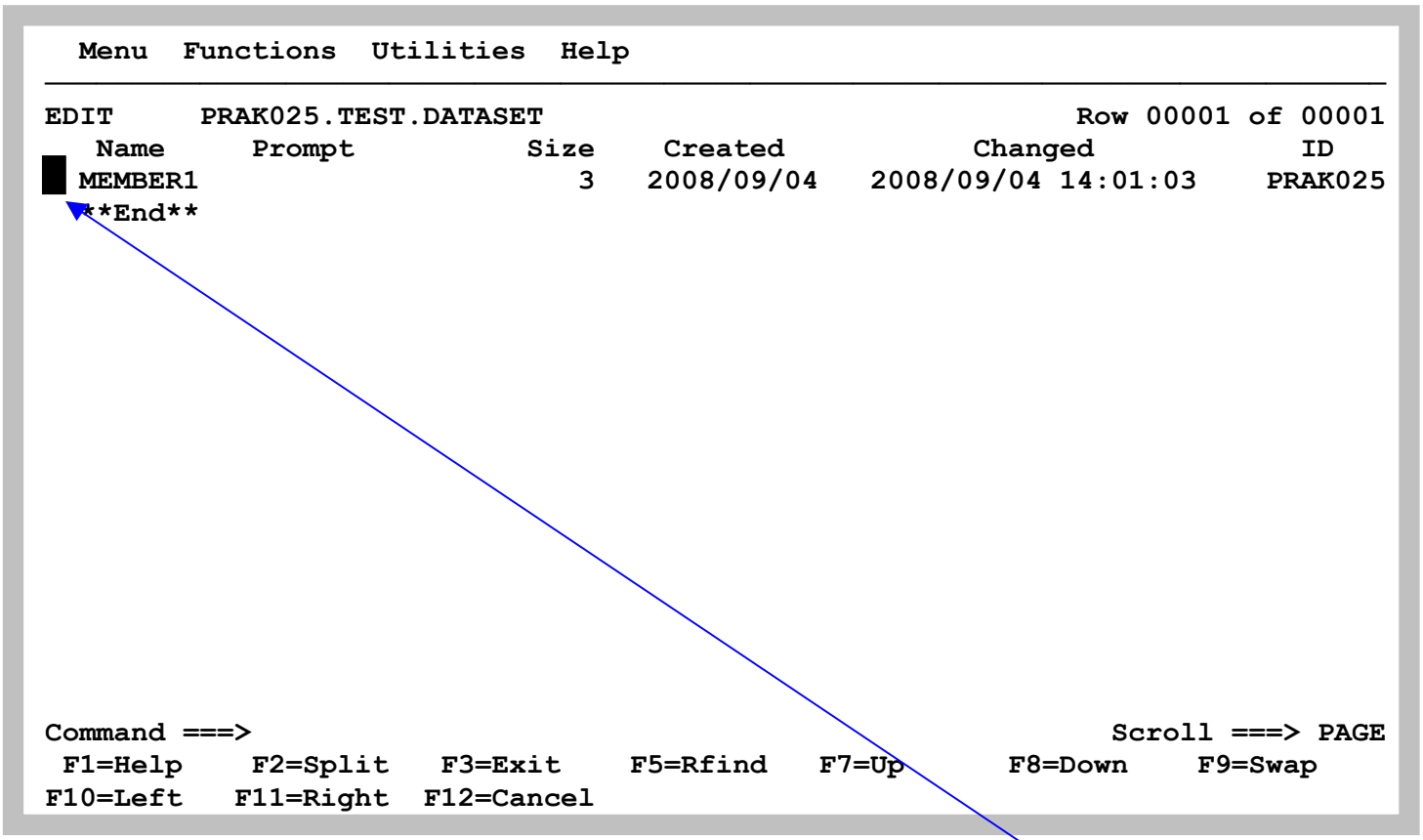
"`Member MEMBER1 saved`" rechts oben im Panel quittiert.

Alternativ dazu kann man den Stand der Arbeit auch über die Action Bar abspeichern. Dazu stellt man den Cursor mit Hilfe der Maus in die erste Zeile des Panels, und zwar dort auf "`File`". Ein anschließendes Drücken der Eingabetaste öffnet ein Pull Down-Menü. Hier gibt man eine "`1`" für "`Save`" ein und schließt diese Aktion mit der Eingabetaste ab.

Den geöffneten Member verläßt man mit F3 (Exit). Noch nicht auf Festplatte gesicherte Änderungen werden automatisch gespeichert. Ein zweites Drücken von F3 verlässt auch den Edit Entry Panel; damit wird der ISPF-Editor geschlossen.



Rechts oben erscheint die Meldung, dass unser Member abgespeichert wurde: "Member MEMBER1 saved" .
Betätigen Sie die Eingabetaste

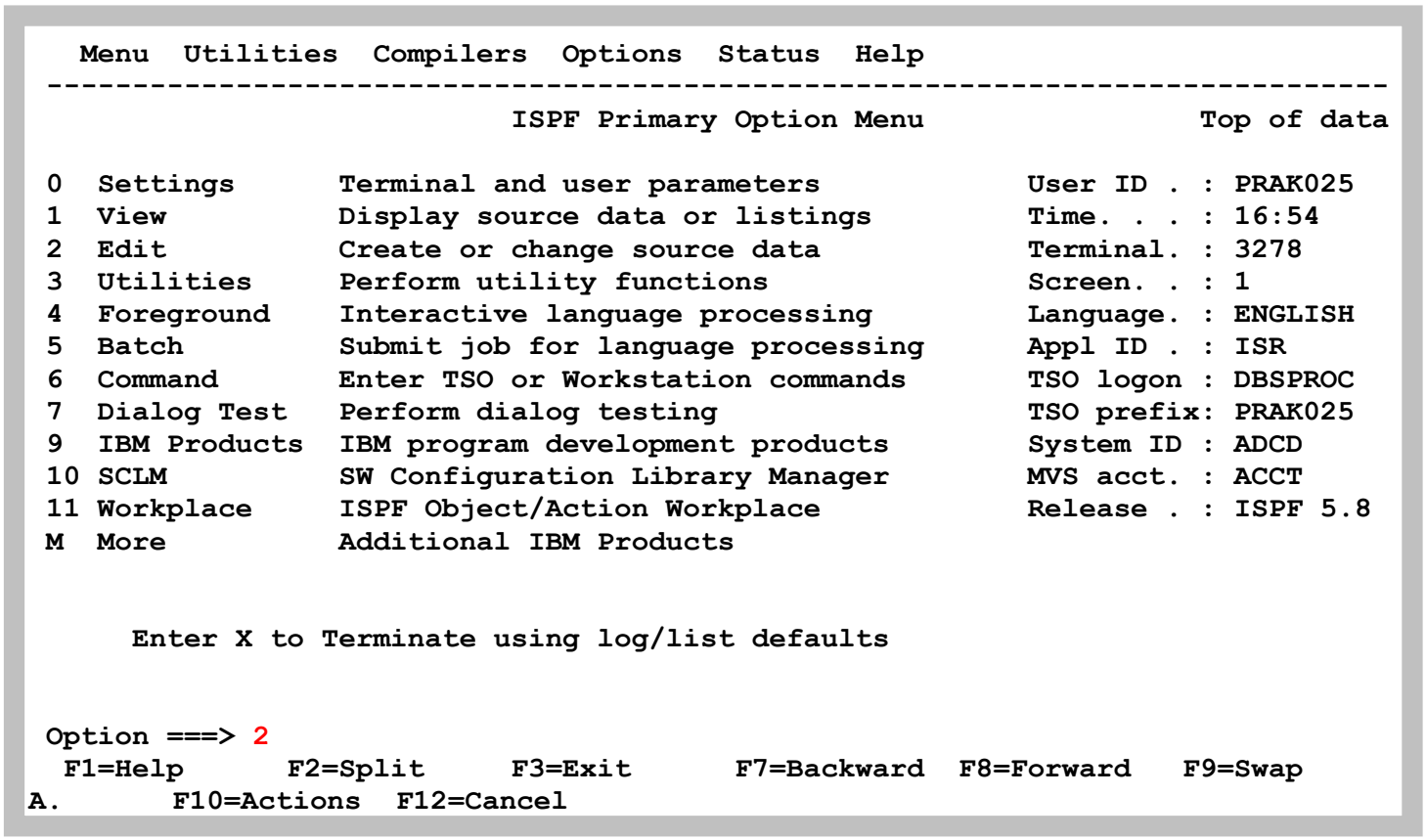


Zu sehen sind alle Members unseres Datasets PRAK025.TEST.DATASET .Im Augenblick hat der Dataset nur einen Member mit dem Namen MEMBER1. Durch Plazieren des Cursors auf den Punkt links neben dem Member-Namen sowie Betätigung der Eingabetaste rufen wir den ISPF-Editor noch einmal auf.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       PRAK025.TEST.DATASET(MEMBER1) - 01.00           Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -CAUTION- Profile changed to CAPS ON (from CAPS OFF) because the
==MSG>           data does not contain any lower case characters.
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 ##
000200 ## DAS IST DER INHALT VON MEMBER1.
000300 ##
****** **************************** Bottom of Data ****************************









Command ===>                                                 Scroll ===> PAGE
 F1=Help       F2=Split      F3=Exit       F5=Rfind      F6=Rchange    F7=Up
 F8=Down       F9=Swap      F10=Left      F11=Right     F12=Cancel
```

Der Editor hat unseren Text mit Zeilennummern versehen . Wir könnten ihn jetzt abändern, brauchen dazu aber Kenntnisse des ISPF-Editors. Diese lernen Sie im nächsten Abschnitt kennen..

3. Nutzung der Line Command Area



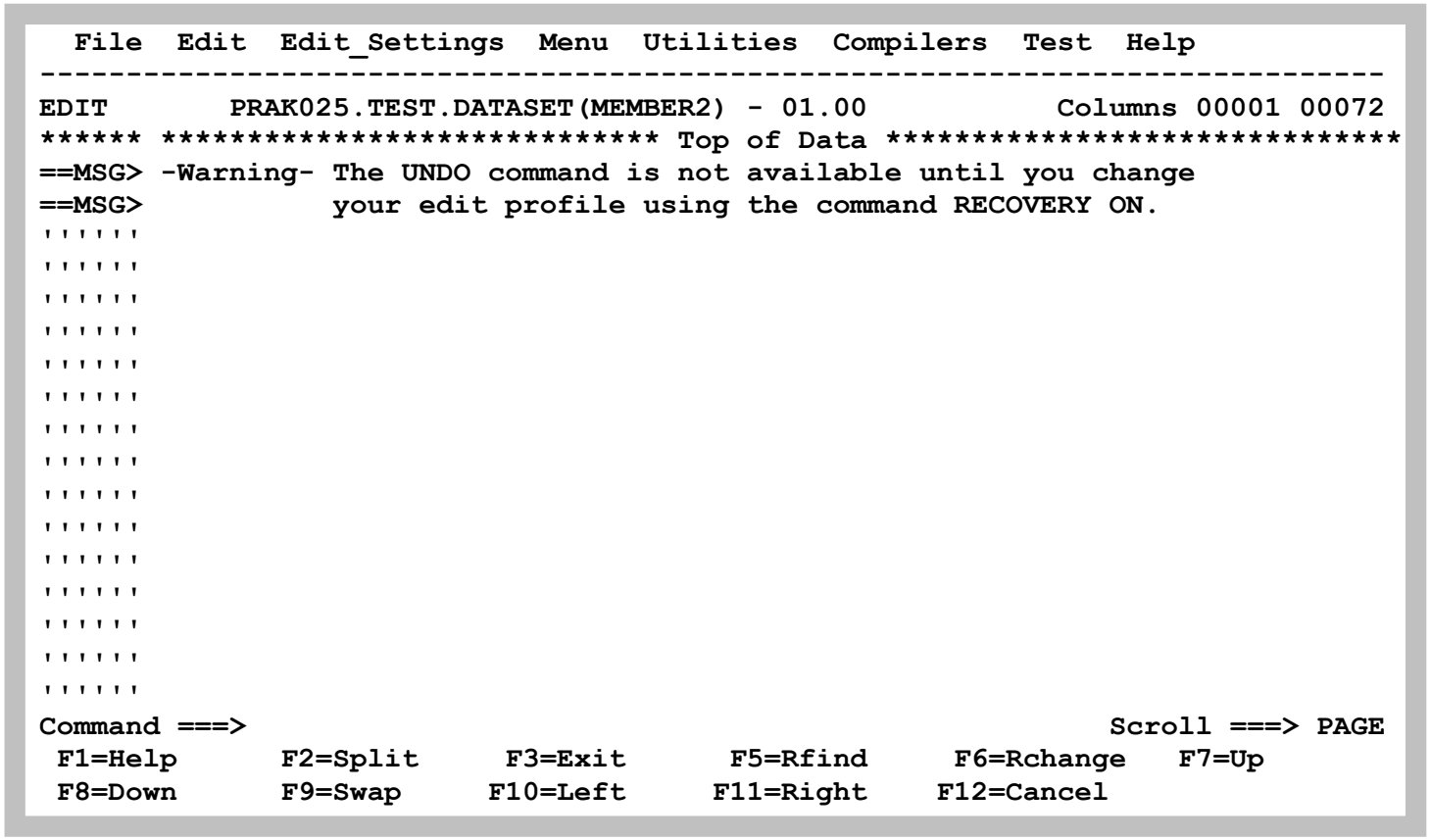
Im ISPF Primary Options Menue bringt uns eine 2 zum Edit Panel

```
B.         Menu  RefList  RefMode  Utilities  Workstation  Help
 ------------------------------------------------------------------------------
                                Edit Entry Panel

 ISPF Library:
    Project . . . PRAK025
    Group . . . . TEST     . . .           . . .           . . .
    Type  . . . . DATASET
    Member  . . .                  (Blank or pattern for member selection list)

 Other Partitioned, Sequential or VSAM Data Set:
    Data Set Name . . .
    Volume Serial . . .            (If not cataloged)

 Workstation File:
    File Name . . . . .
                                          Options
 Initial Macro  . . . .                      Confirm Cancel/Move/Replace
 Profile Name . . . . .                      Mixed Mode
 Format Name  . . . . .                      Edit on Workstation
 Data Set Password  . .                      Preserve VB record length
 Command ===>
  F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
 F10=Actions  F12=Cancel
```

Der vorher bereits angelegte Partitioned Data Set wird angezeigt.

```
  Menu  RefList  RefMode  Utilities  Workstation  Help
--------------------------------------------------------------------------------
                              Edit Entry Panel

 ISPF Library:
C.         Project . . . PRAK025
    Group . . . . TEST     . . .          . . .          . . .
    Type  . . . . DATASET
    Member  . . . Member2         (Blank or pattern for member selection list)

 Other Partitioned, Sequential or VSAM Data Set:
    Data Set Name . . .
    Volume Serial . . .           (If not cataloged)

 Workstation File:
    File Name . . . . .
                                        Options
 Initial Macro  . . . .                    Confirm Cancel/Move/Replace
 Profile Name . . . . .                    Mixed Mode
 Format Name  . . . . .                    Edit on Workstation
 Data Set Password  . .                    Preserve VB record length
 Command ===>
  F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
F10=Actions  F12=Cancel
```

Wir fügen einen neuen Member mit dem Namen “Member2” zu. Enter

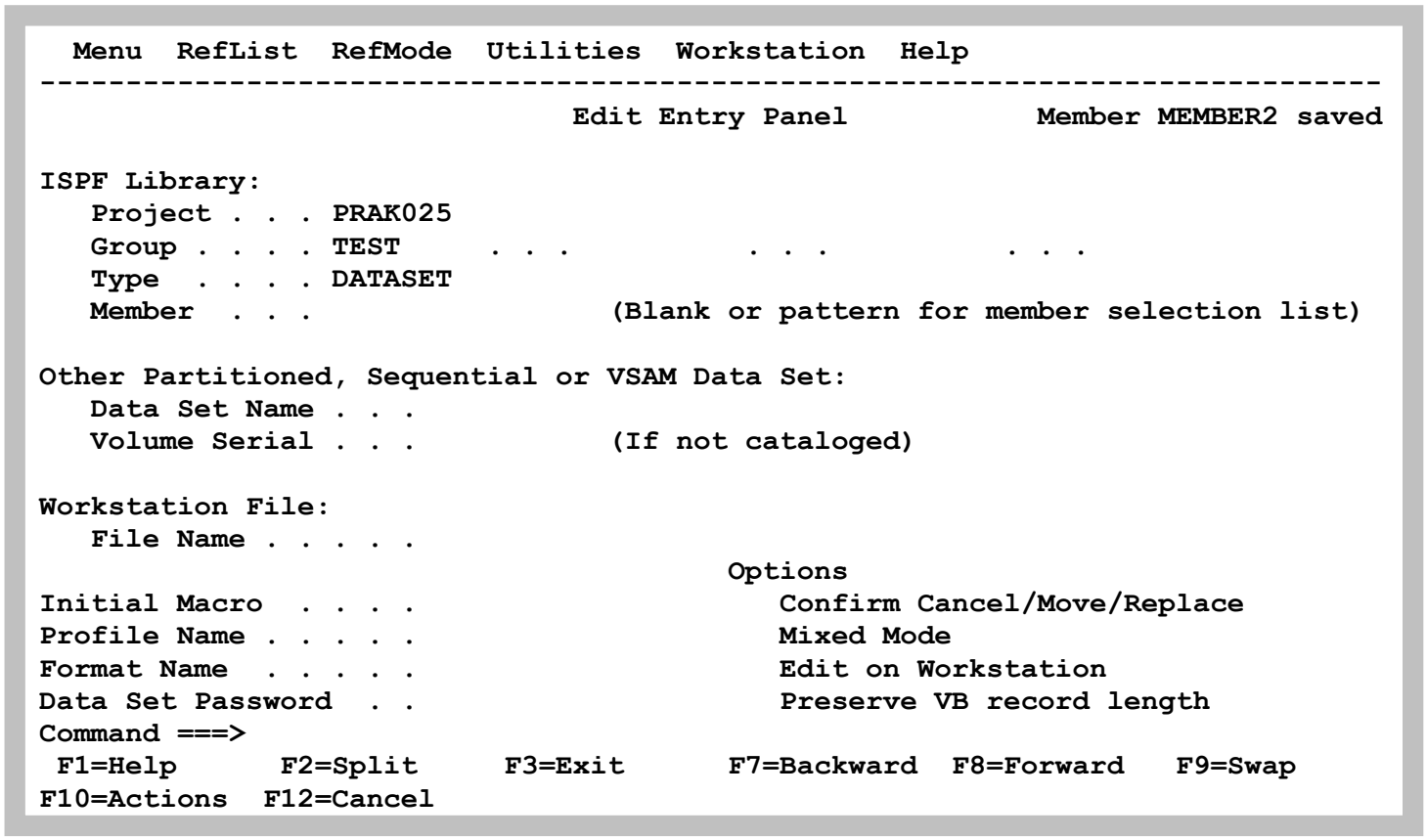


Der ISPF Editpr öffnet ein Fenster, in dem wir Daten eingeben.

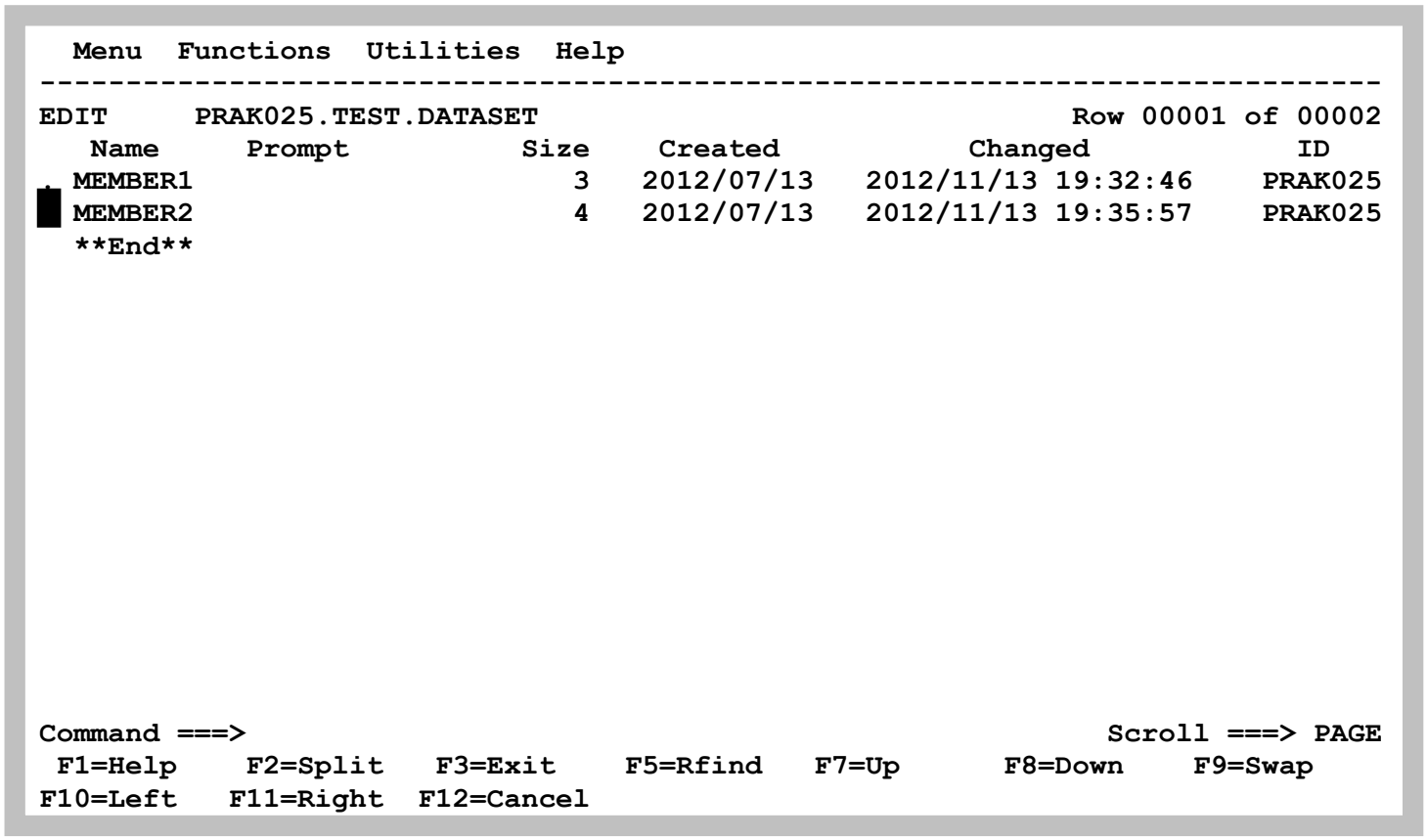
```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
--------------------------------------------------------------------------------
EDIT       PRAK025.TEST.DATASET(MEMBER2) - 01.00           Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
'''''' Ich weiß nicht was soll es bedeuten
'''''' Daß ich so traurig bin;
'''''' Ein Märchen aus alten Zeiten,
'''''' Das kommt mir nicht aus dem Sinn.
'''''' Die Luft ist kühl und es dunkelt,
'''''' Und ruhig fließt der Rhein;
'''''' Der Gipfel des Berges funkelt
'''''' Im Abendsonnenschein.
''''''
''''''
''''''
''''''
''''''
''''''
''''''
Command ===>                                                 Scroll ===> PAGE
 F1=Help       F2=Split      F3=Exit       F5=Rfind      F6=Rchange    F7=Up
 F8=Down       F9=Swap      F10=Left      F11=Right     F12=Cancel
```

Wir geben als Übungstext die beiden ersten Strophen des Gedichtes „Die Lore-Ley“ ein, das wohl bekannteste Gedivht von Heinrich Heine, 1824 veröffentlicht. Buchstabe ß in der alten Rechtschreibung.

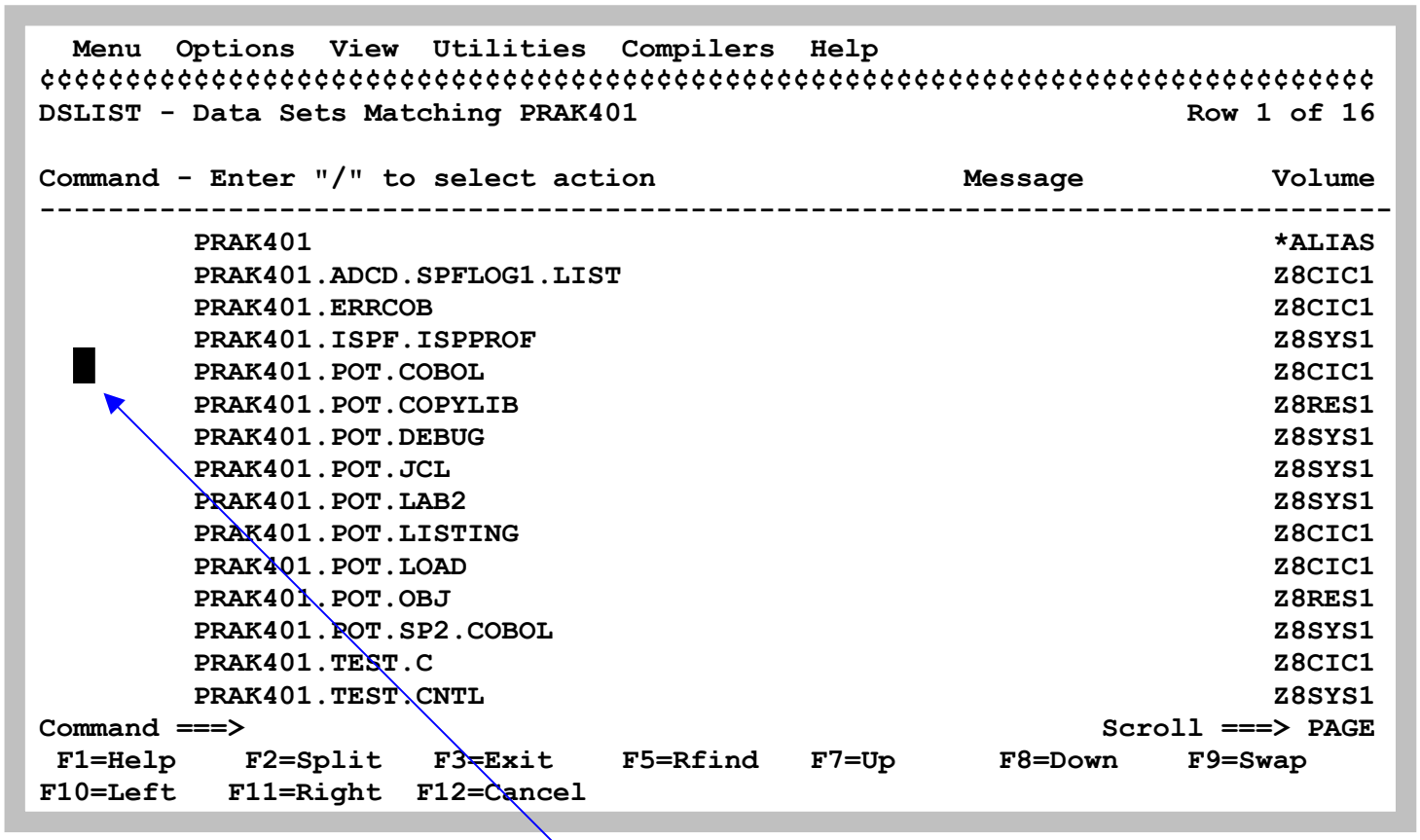
F3



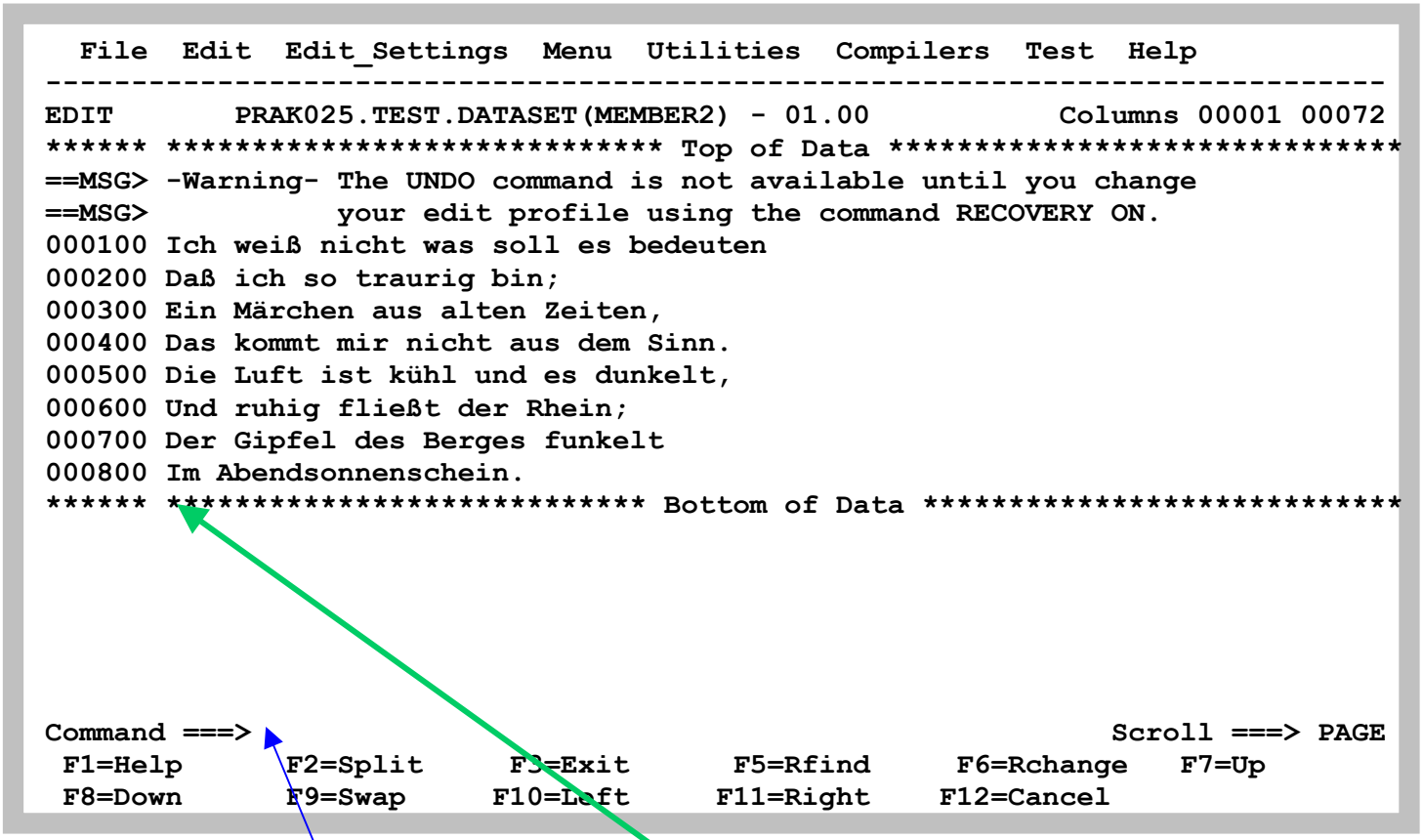
F3 Taste. Damit wird unser Input automatisch gespeichert.



Unser PDS Data Set hat im Augenblick zwei Member mit dem Namen Member1 und Member2. Wir bewegen den Cursor vor den Namen des Members und geben Enter ein.



Normalerweise würde ein PDS Data Set viele Member enthalten, die oft gar nicht alle auf einen einzigen Screen passen, und mit den F8 bzw. F7 Scroll Operationen angeschaut werden. Da ist es sehr praktisch, wenn man den Cursor vor den Data Set Namen setzen kann, den man für den ISPF Editor öffnen möchte, z.B. hier PRAK401.POT.COBOL



Der ISPF Editor hat unsere Eingabe automatisch mit Zeilennummern versehen. Bei manchen Eingaben, z.B. Cobol Quell Text ist das zwingend erforderlich; hier ist es eine nette Feature. Man bezeichnet diesen Bereich als die „Line Command Area“, und kann ihn benutzen, um dort (an Stelle der Command Line) Kommandos einzugeben.

In dem ISPF Editor existieren zwei unterschiedliche Arten von Kommandos:

- **primary commands**
- **line commands (auch als prefix commands bezeichnet)**

Primary Commands werden in der Command Input Area (hinter dem ==> command prompt Symbol) eingegeben. Sie gelten für den gesamzen PDS Member.

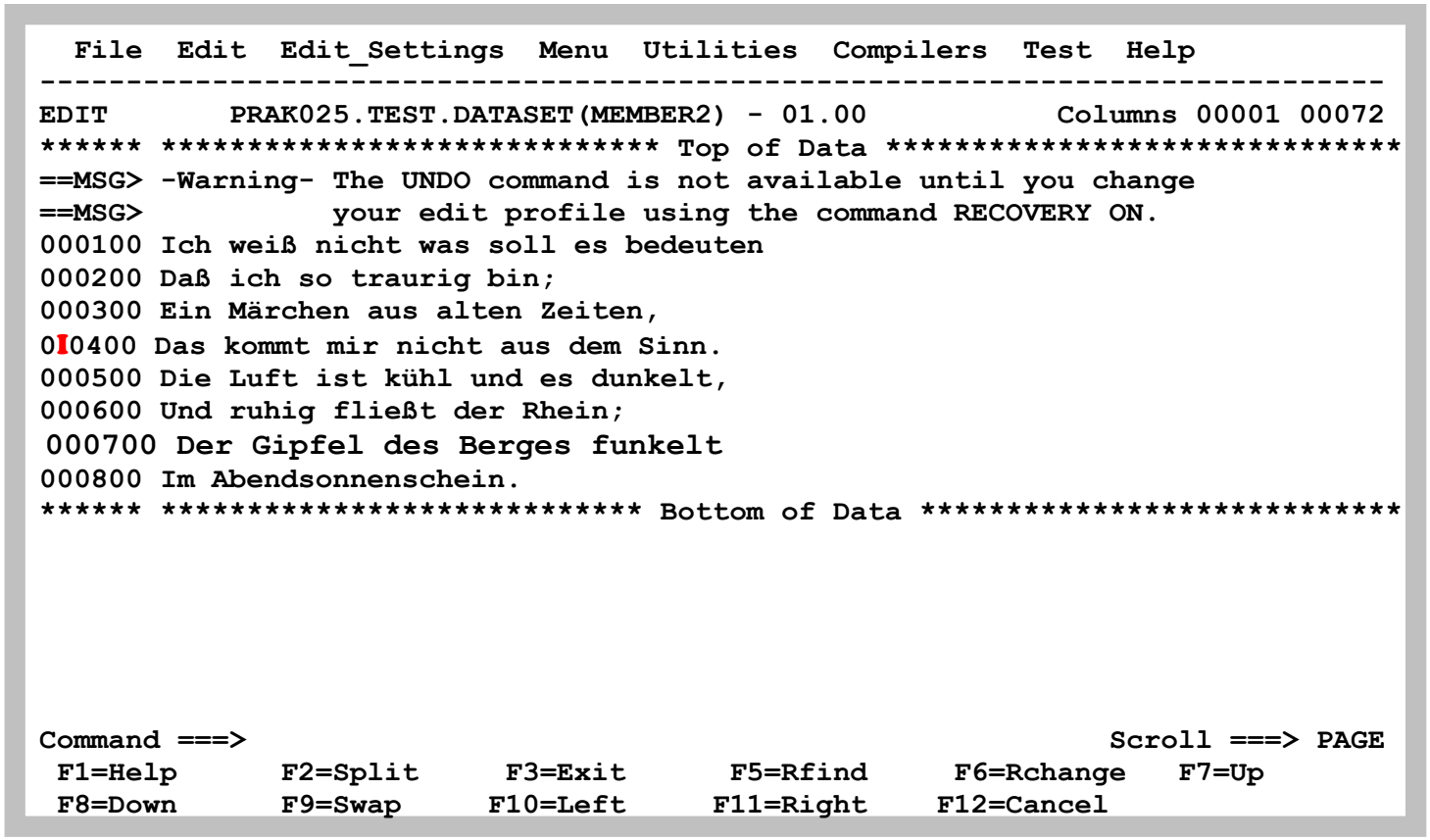
Line commands (prefix commands) überschreiben eineZeilennummer in der Line Command Area. Sie beziehen sich auf einzelne Zeilen.

Leerzeilen können mit dem I Command (insert) eingefügt werden. Es wird auf der Zeile in der Line Command Area eingegeben, wo die Leerzeile entstehen soll. Wenn wir beispielsweise die Zeilennummer 400 mit "I" überschreiben, wird hinter der Zeile000400 eine Leerzeile eingefügt, und alle folgenden Zeilen rutschen um eins nach unten.

In die Leerzeile können jetzt Daten eingegeben werden. Wenn Sie Enter drücken entfernt der ISPF Editor alle Leerzeilen, welche keine Daten enthalten.

Es existieren 2 Arten des "I" Commands

I	**erzeugt eine Leerzeile, folgend der Eingabezeile**
In	**erzeugt n Leerzeilen, folgend der Eingabezeile**



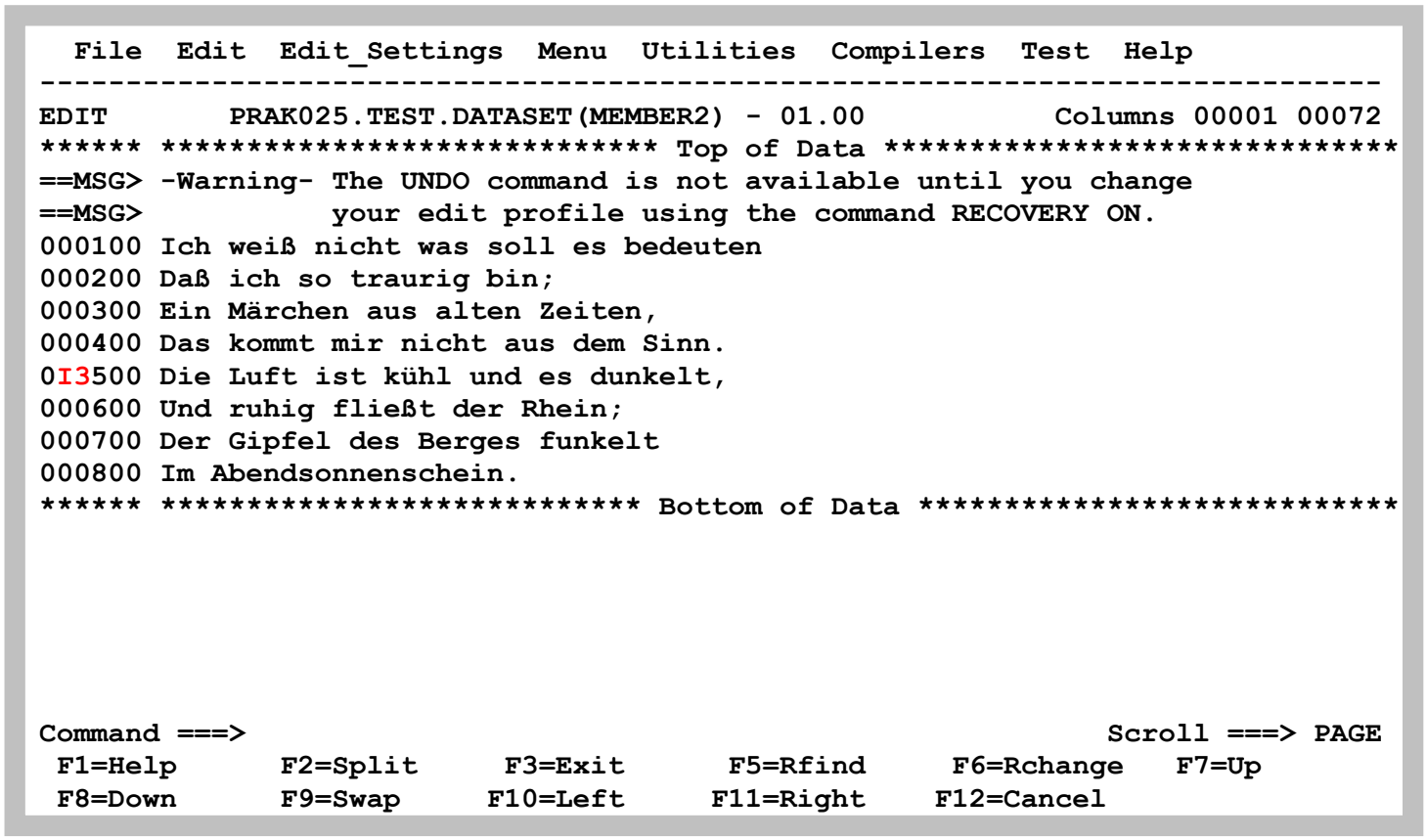
Eingabe von `I` erzeugt

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------
EDIT       PRAK025.TEST.DATASET(MEMBER2) - 01.00           Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 Ich weiß nicht was soll es bedeuten
000200 Daß ich so traurig bin;
000300 Ein Märchen aus alten Zeiten,
000400 Das kommt mir nicht aus dem Sinn.
''''''
000500 Die Luft ist kühl und es dunkelt,
000600 Und ruhig fließt der Rhein;
000700 Der Gipfel des Berges funkelt
000800 Im Abendsonnenschein.
****** **************************** Bottom of Data ****************************




Command ===>                                                 Scroll ===> PAGE
 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

eine Leerzeile.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
--------------------------------------------------------------------------------
EDIT       PRAK025.TEST.DATASET(MEMBER2) - 01.00            Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 Ich weiß nicht was soll es bedeuten
000200 Daß ich so traurig bin;
000300 Ein Märchen aus alten Zeiten,
000400 Das kommt mir nicht aus dem Sinn.
'D''''
000500 Die Luft ist kühl und es dunkelt,
000600 Und ruhig fließt der Rhein;
000700 Der Gipfel des Berges funkelt
000800 Im Abendsonnenschein.
****** **************************** Bottom of Data ****************************




 Command ===>                                                Scroll ===> PAGE
  F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
D.        F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Eingabe von D löscht eine Zeile (in diesem Fall unsere gerade erzeugte Leerzeile).



Eingabe von `I3` erzeugt 3 Leerzeilen

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
--------------------------------------------------------------------------------
EDIT       PRAK025.TEST.DATASET(MEMBER2) - 01.00            Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 Ich weiß nicht was soll es bedeuten
000200 Daß ich so traurig bin;
000300 Ein Märchen aus alten Zeiten,
000400 Das kommt mir nicht aus dem Sinn.
000500 Die Luft ist kühl und es dunkelt,
''''''
''''''
''''''
000600 Und ruhig fließt der Rhein;
000700 Der Gipfel des Berges funkelt
000800 Im Abendsonnenschein.
****** **************************** Bottom of Data ****************************



Command ===>                                                 Scroll ===> PAGE
 F1=Help       F2=Split      F3=Exit       F5=Rfind      F6=Rchange    F7=Up
 F8=Down       F9=Swap      F10=Left      F11=Right     F12=Cancel
```

Das sieht dann so aus.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       PRAK025.TEST.DATASET(MEMBER2) - 01.01           Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 Ich weiß nicht was soll es bedeuten
000200 Daß ich so traurig bin;
000300 Ein Märchen aus alten Zeiten,
000400 Das kommt mir nicht aus dem Sinn.
0D3500 Die Luft ist kühl und es dunkelt,
000600 Und ruhig fließt der Rhein;
000700 Der Gipfel des Berges funkelt
000800 Im Abendsonnenschein.
****** **************************** Bottom of Data ****************************




 Command ===>                                                Scroll ===> PAGE
  F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
E.        F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Eingabe von D3 löscht 3 Zeilen

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
--------------------------------------------------------------------------------
EDIT       PRAK025.TEST.DATASET(MEMBER2) - 01.02            Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 Ich weiß nicht was soll es bedeuten
000200 Daß ich so traurig bin;
000300 Ein Märchen aus alten Zeiten,
000400 Das kommt mir nicht aus dem Sinn.
000800 Im Abendsonnenschein.
****** **************************** Bottom of Data ****************************









Command ===>                                                 Scroll ===> PAGE
 F1=Help       F2=Split      F3=Exit       F5=Rfind      F6=Rchange    F7=Up
 F8=Down       F9=Swap      F10=Left      F11=Right     F12=Cancel
```

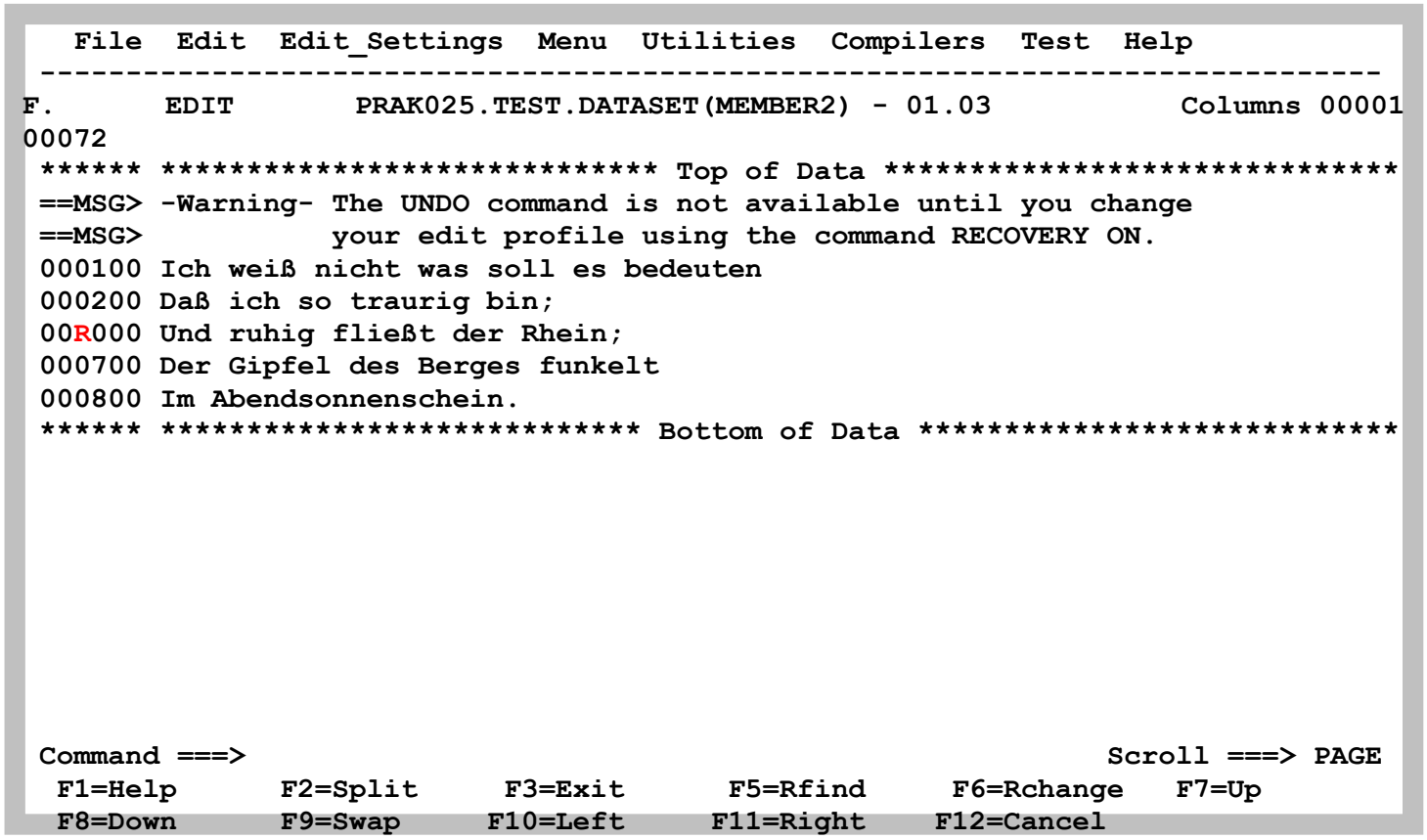
Und das Gedicht von Heine wurde gekürzt.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------
EDIT       PRAK025.TEST.DATASET(MEMBER2) - 01.02            Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 Ich weiß nicht was soll es bedeuten
000200 Daß ich so traurig bin;
00DD00 Ein Märchen aus alten Zeiten,
000400 Das kommt mir nicht aus dem Sinn.
00DD00 Die Luft ist kühl und es dunkelt,
000600 Und ruhig fließt der Rhein;
000700 Der Gipfel des Berges funkelt
000800 Im Abendsonnenschein.
****** **************************** Bottom of Data ****************************




Command ===>                                                Scroll ===> PAGE
 F1=Help       F2=Split      F3=Exit       F5=Rfind      F6=Rchange    F7=Up
 F8=Down       F9=Swap      F10=Left      F11=Right     F12=Cancel
```

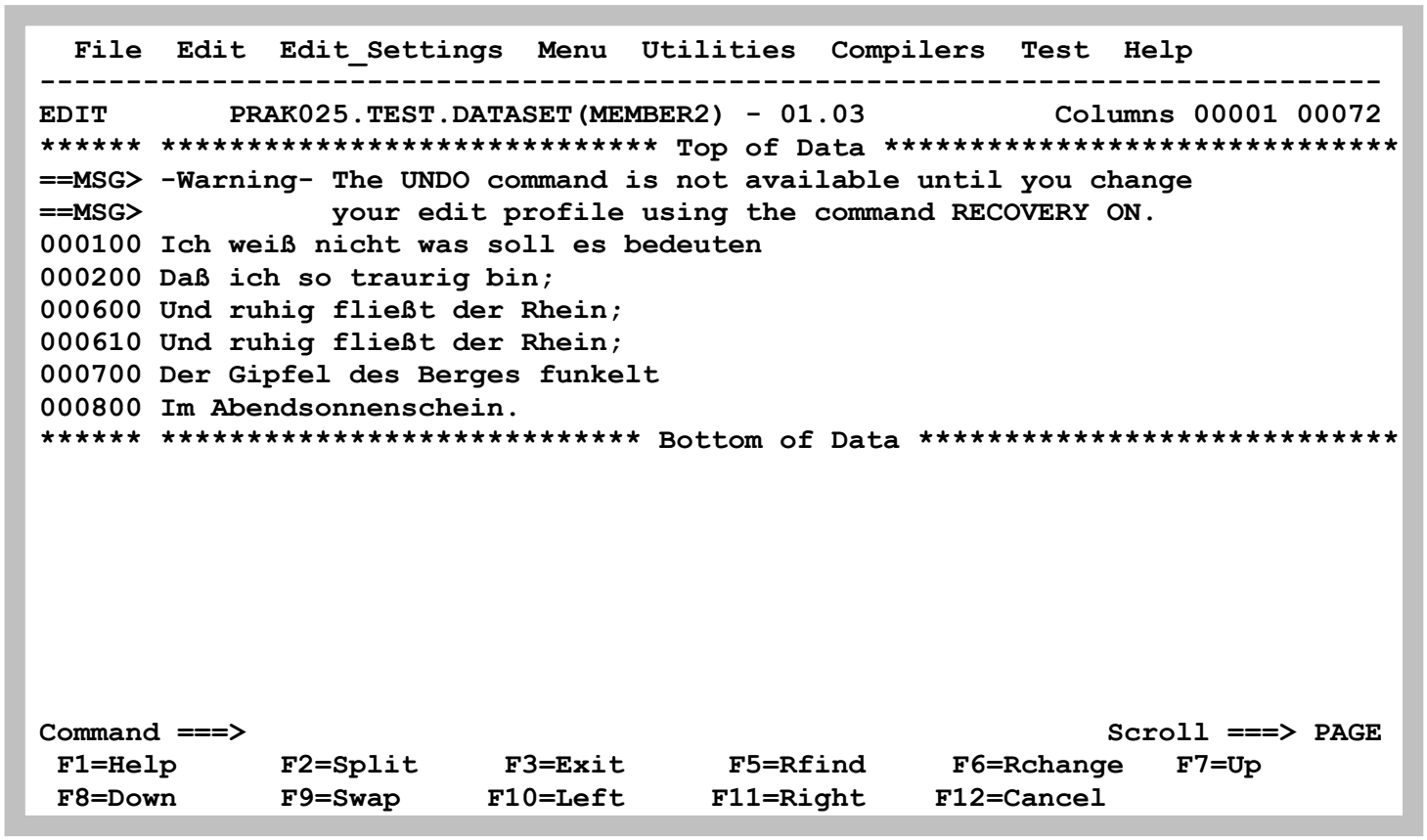
Alternativ hätten wir das gleiche mit 2 DD Kommandos erreicht.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------
EDIT       PRAK025.TEST.DATASET(MEMBER2) - 01.03           Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 Ich weiß nicht was soll es bedeuten
000200 Daß ich so traurig bin;
000600 Und ruhig fließt der Rhein;
000700 Der Gipfel des Berges funkelt
000800 Im Abendsonnenschein.
****** **************************** Bottom of Data ****************************




Command ===>                                                 Scroll ===> PAGE
 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

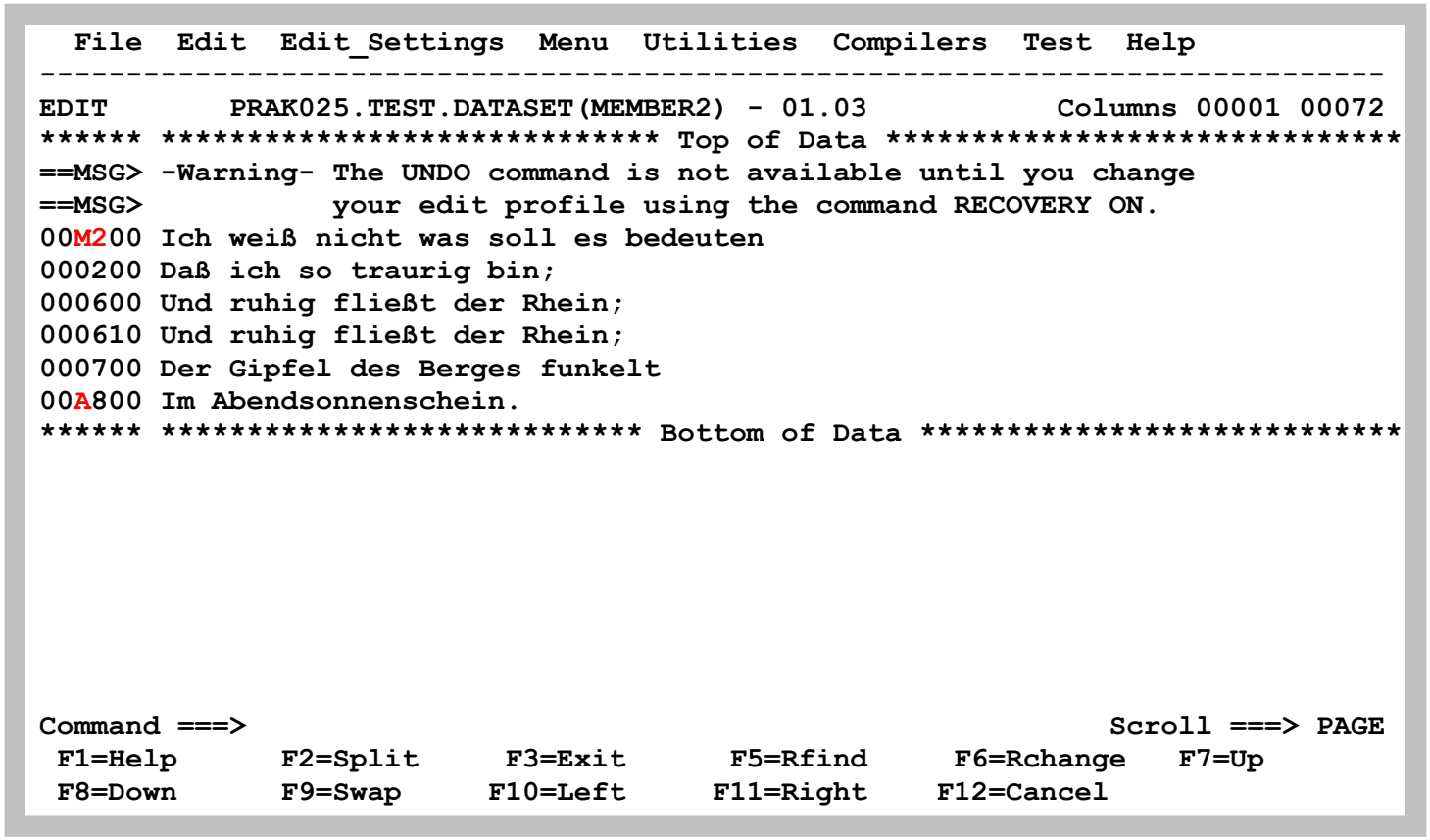
Und hier ist das Ergebnis.



Das R Kommando replicates eine Zeile.


File Edit Edit_Settings Menu Utilities Compilers Test Help
EDIT PRAK025.TEST.DATASET(MEMBER2) - 01.03 Columns 00001 00072
****** ****************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG> your edit profile using the command RECOVERY ON.
000100 Ich weiß nicht was soll es bedeuten
000200 Daß ich so traurig bin;
000600 Und ruhig fließt der Rhein;
000610 Und ruhig fließt der Rhein;
000700 Der Gipfel des Berges funkelt
000800 Im Abendsonnenschein.
****** ***************************** Bottom of Data ******************************
Command ===> Scroll ===> PAGE
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel

Siehe hier.



Mit M2 können wir 2 Zeilen zu der mit A markierten Zeile bewegen (move).

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------
EDIT       PRAK025.TEST.DATASET(MEMBER2) - 01.04           Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000600 Und ruhig fließt der Rhein;
000610 Und ruhig fließt der Rhein;
000700 Der Gipfel des Berges funkelt
000800 Im Abendsonnenschein.
000900 Ich weiß nicht was soll es bedeuten
001000 Daß ich so traurig bin;
****** **************************** Bottom of Data ****************************




Command ===>                                                 Scroll ===> PAGE
 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Hier das Ergebnis.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------
EDIT       PRAK025.TEST.DATASET(MEMBER2) - 01.04            Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
00C200 Und ruhig fließt der Rhein;
000610 Und ruhig fließt der Rhein;
000700 Der Gipfel des Berges funkelt
000800 Im Abendsonnenschein.
000900 Ich weiß nicht was soll es bedeuten
00A000 Daß ich so traurig bin;
****** **************************** Bottom of Data ****************************




Command ===>                                                Scroll ===> PAGE
 F1=Help       F2=Split      F3=Exit       F5=Rfind      F6=Rchange    F7=Up
 F8=Down       F9=Swap      F10=Left      F11=Right     F12=Cancel
```

Mit C2 können wir 2 Zeilen Kopieren (copy).

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------
EDIT       PRAK025.TEST.DATASET(MEMBER2) - 01.04           Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000600 Und ruhig fließt der Rhein;
000610 Und ruhig fließt der Rhein;
000700 Der Gipfel des Berges funkelt
000800 Im Abendsonnenschein.
000900 Ich weiß nicht was soll es bedeuten
001000 Daß ich so traurig bin;
001100 Und ruhig fließt der Rhein;
001200 Und ruhig fließt der Rhein;
****** **************************** Bottom of Data ****************************




Command ===>                                                Scroll ===> PAGE
 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Ist passiert.

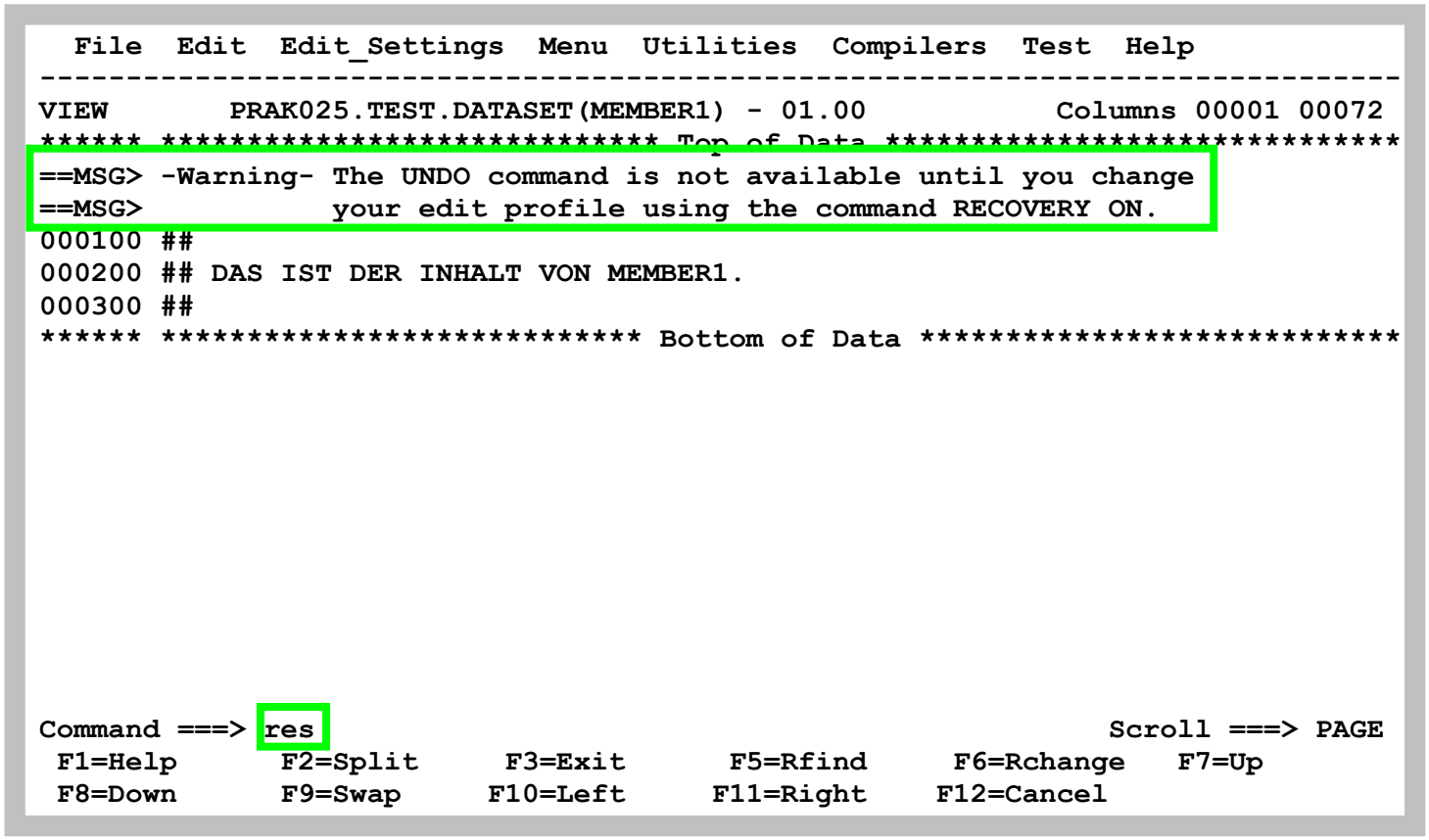
Experimentieren Sie etwas mit diesen Möglichkeiten. Es ist recht schnell erlernt.

4. ISPF Edit Panel - Line Command Übersicht

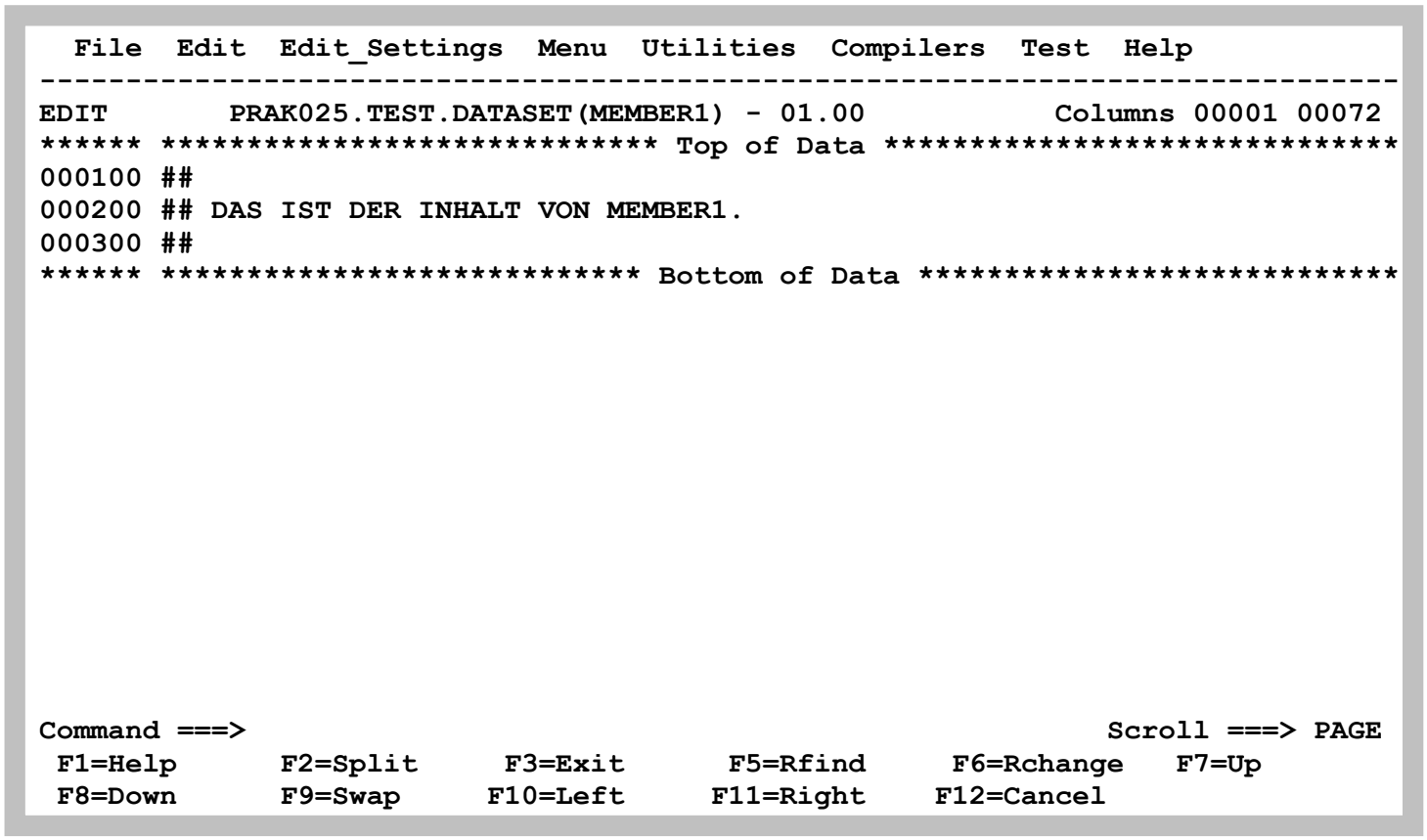
Command	Description
I	Insert lines
D	Delete lines
R	Repeat lines
C	Copy lines
M	Move lines
A	After line
B	Before line
(	Shift right columns
<	Shift right data
)	Shift left columns
>	Shift left data
X	Exclude lines

5. Nützliche Tips

5.1 Text verbergen



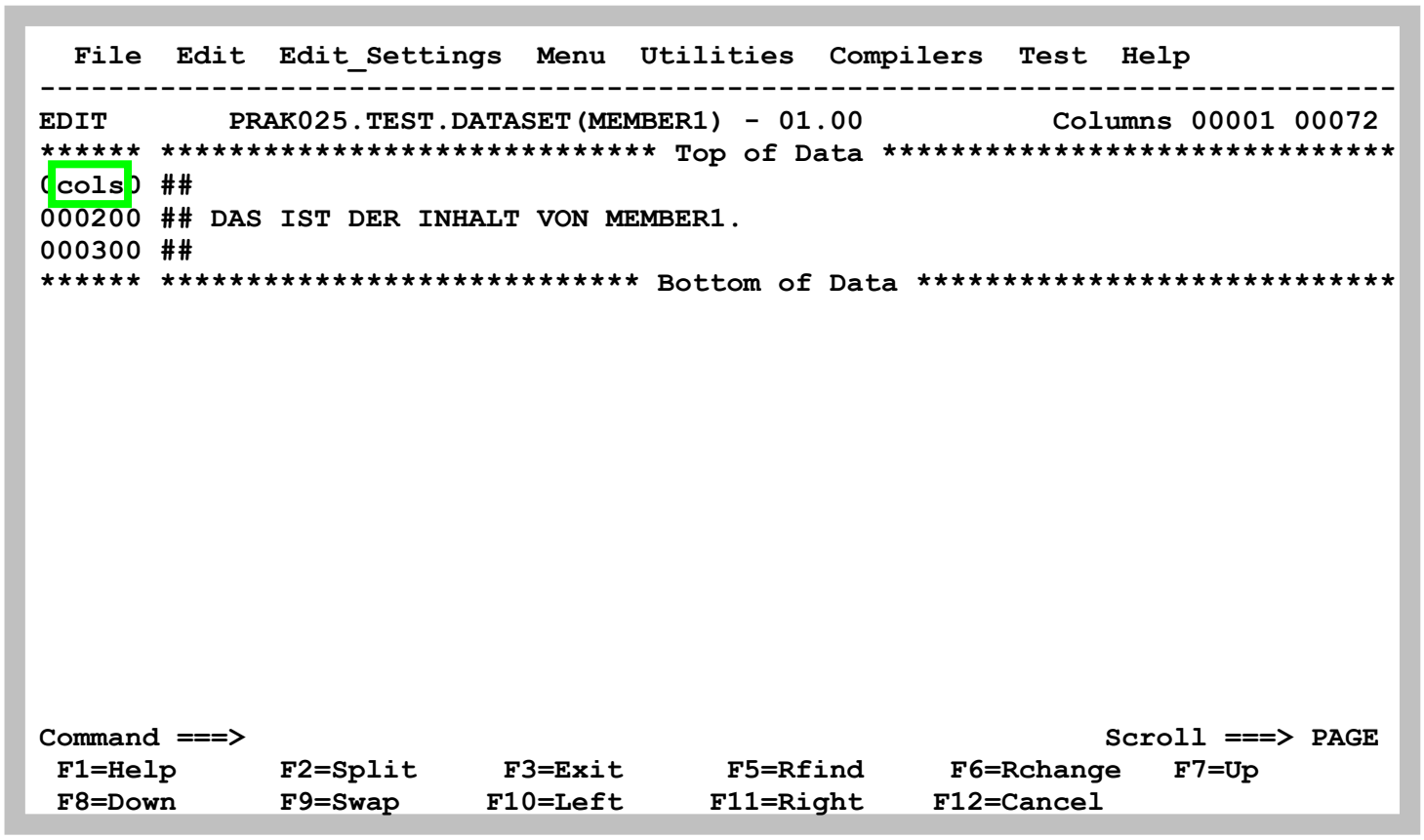
Öffnen Sie den Dataset `PRAK025.TEST.DATASET(MEMBER1)` z. B mit dem ISPF Editor. Sie sehen oben zwei Zeilen
mit ==MSG>. Die Messages in diesen Zeilen sind für Sie evtl. nicht so sehr interessant- Sie können diese Zeilen entfernen, indem Sie einfach in die Command Zeile "`res`" schreiben.



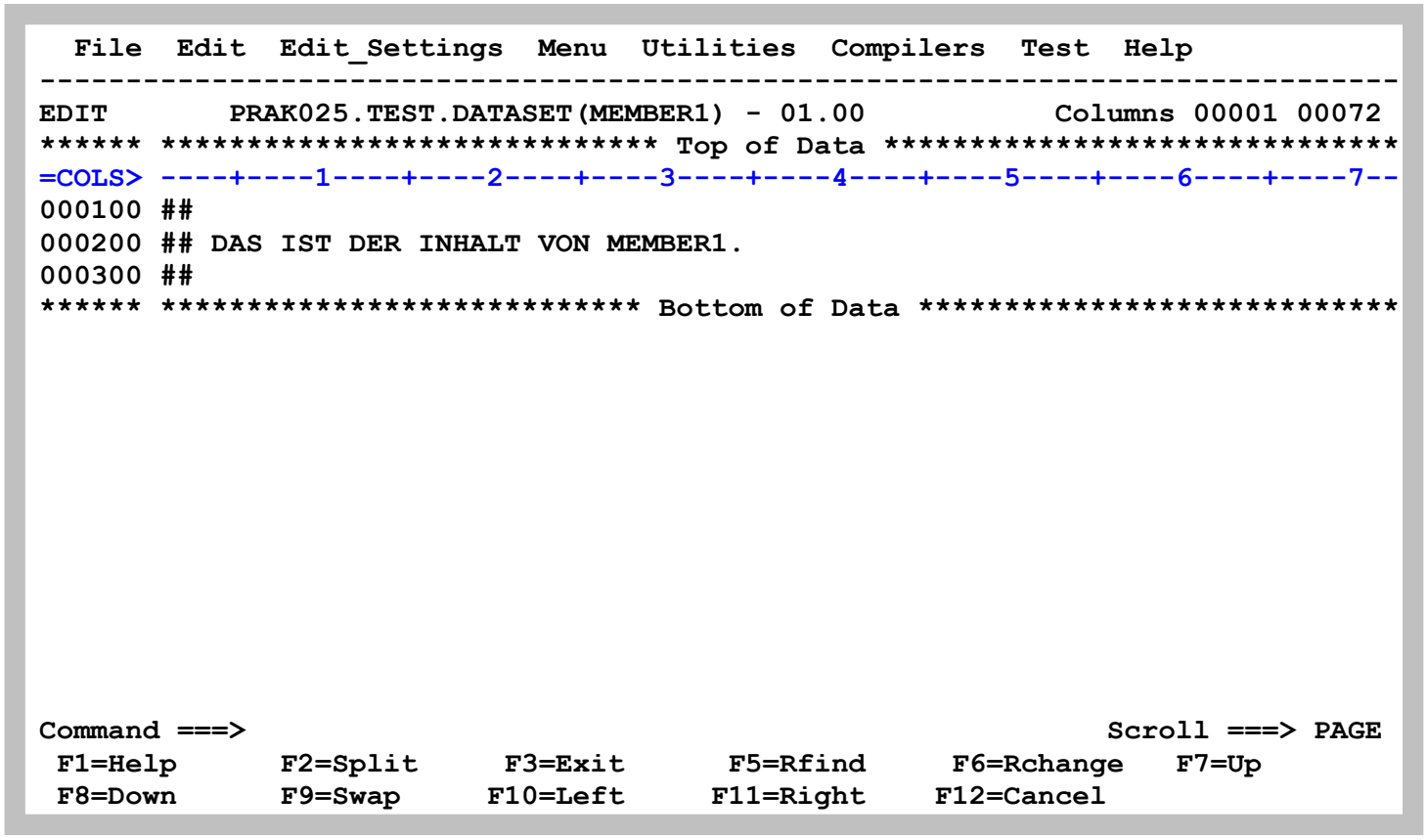
Damit sind die Message Zeilen verschwunden. Es können da nämlich manchmal auch mehr ==MSG> lines stehen und die Übersicht geht verloren.

5.2 Spalten anzeigen

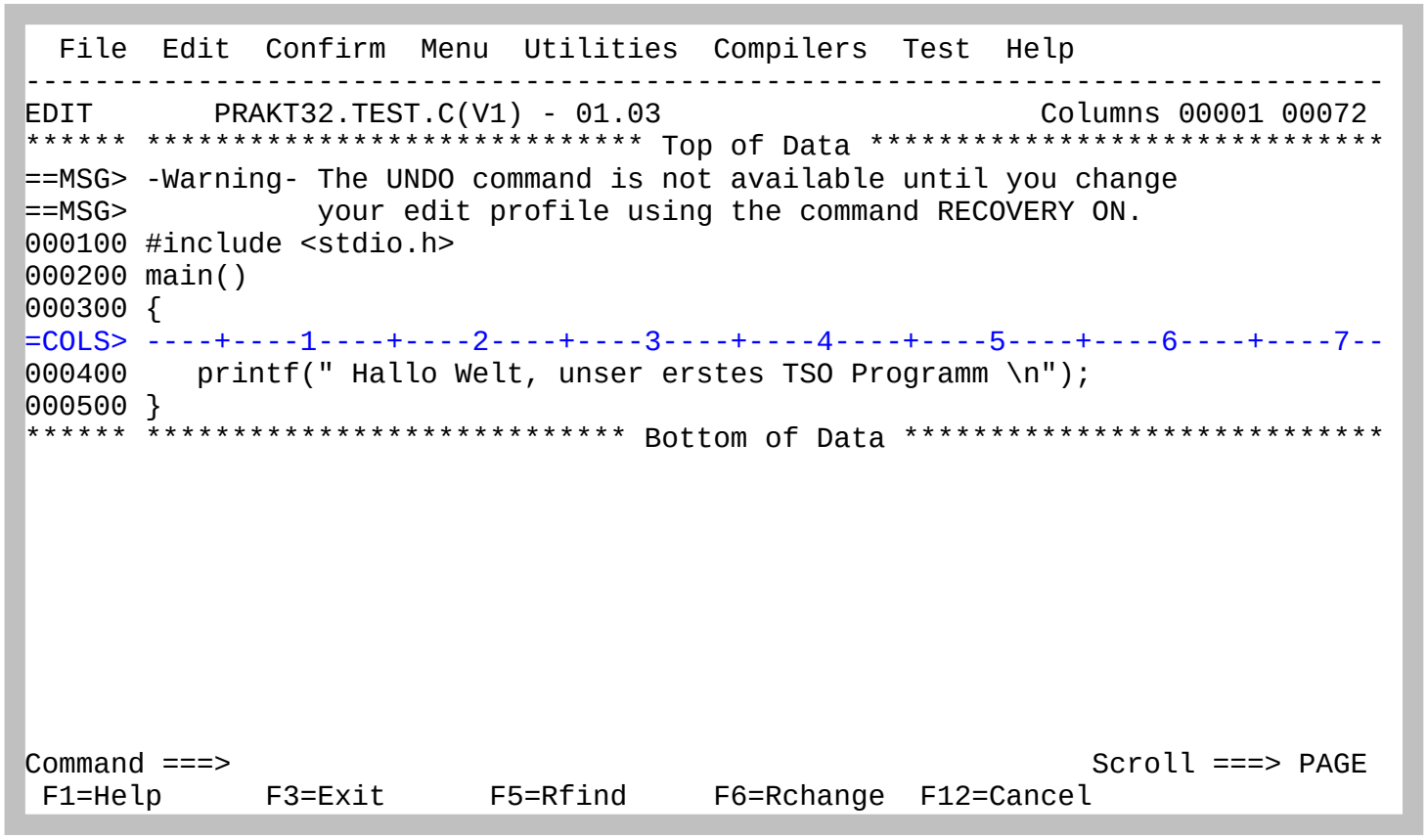
Tip: Um Kleinbuchstaben eingeben zu können: Auf der Command Line „CAPS OFF“ eingeben



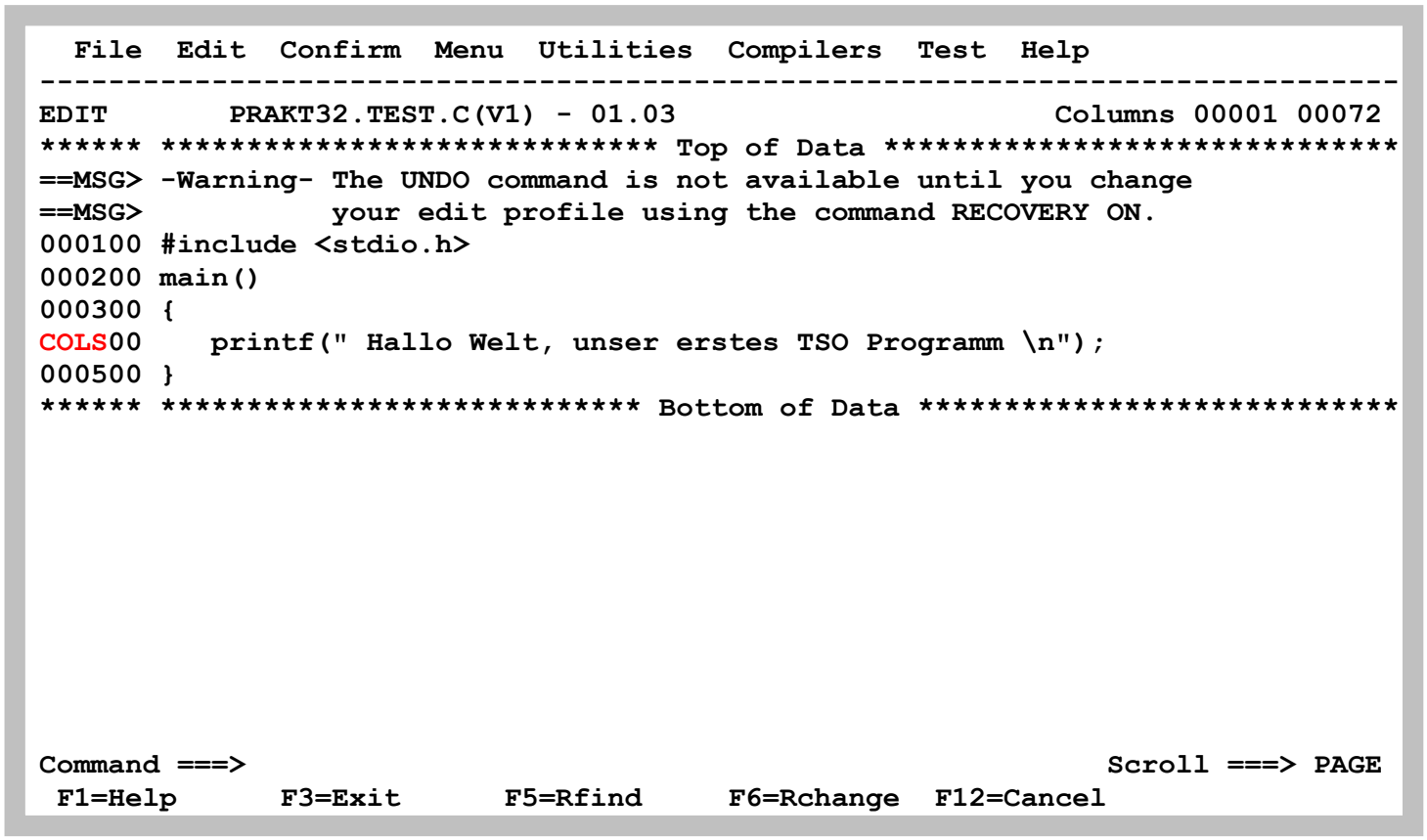
Schreiben Sie in die Zeile mit der Zeilen Nummer „000100“ den Wert "cols"; .....



das sieht dann so aus. Es werden die Spaltennummern angezeigt 1=10, 2=20, und so weiter.



Sie könnten auch cols in eine beliebige andere Zeile eintragen. Das würde dann wie hier gezeigt aussehen. Nützlich, wenn Sie die Spaltenausrichtung für eine spezifische Zeile sehen wollen.

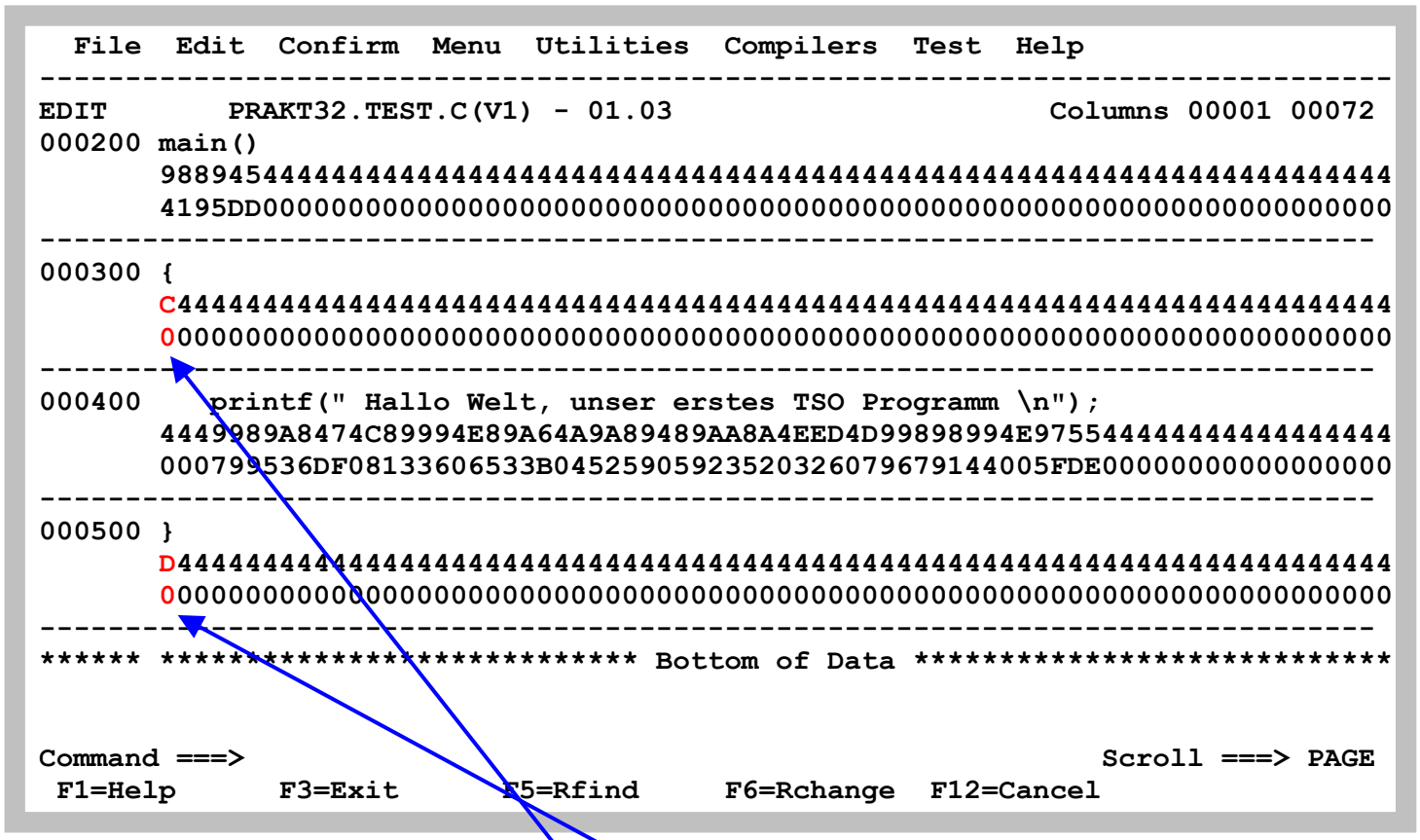


Beispiel für Programm Code

```
  File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       PRAKT32.TEST.C(V1) - 01.03                Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 #include <stdio.h>
000200 main()
000300 {
=COLS> ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000400    printf(" Hallo Welt, unser erstes TSO Programm \n");
000500 }
****** **************************** Bottom of Data ****************************




Command ===>                                              Scroll ===> PAGE
 F1=Help      F3=Exit      F5=Rfind     F6=Rchange   F12=Cancel
```

Bei F3 und erneutem Aufruf ist das Lineal verschwunden

5.3 Hex Darstellung



Rechteckige und geschweifte Klammern: Auf der Command Line „HEX ON“ eingeben. Mit „HEX OFF zurück.

Dies sind die korrekten Representationen: { = C0 ; } = D0 ; [= AD ;] = BD

Die Darstellung ist je nach verwendeter Code Page und je nach verwendetem Emulator anders.

Empfohlen: Code Page 1047. Funktioniert mit dem VISTA 3270 Emulator, aber nicht immer mit anderen 3270 Emulatoren

Aufgabe: Erstellen Sie einen Member "V1" (Version 1) Ihres Datasets "TEST.CNTL".
Füllen Sie diesen mit den 6 Zeilen des unten stehenden JCL-Scriptes. Erstellen
Sie mit Hilfe der Tastenkombination "ALT / Druck" einen Screenshot Ihres Panels
(JCL-Script mit Zeilennummern) und schicken Sie diesen per E-Mail an Ihren Tutor.
Achten Sie darauf, dass das Bild nicht mehr als 250 KByte Speicherplatz belegt. Sehr
gut ist das JPEG-Format, das mit weniger als 100 KByte auskommt.

JCL-Script:

```
//PRAK025C JOB (),CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1),NOTIFY=&SYSUID,
//          TIME=1440,REGION=0K
//PROCLIB  JCLLIB ORDER=CBC.SCCNPRC
//CCL      EXEC PROC=EDCCB,
//          INFILE='PRAK025.TEST.C(V1)',
//          OUTFILE='PRAK025.TEST.LOAD(V1),DISP=SHR'
```

6. Literatur

Für Interessenten gibt es eine umfangreiche Dokumentation von IBM unter

http://publibfp.boulder.ibm.com/epubs/pdf/ispzem50.pdf

oder alternativ

http://www.informatik.uni-leipzig.de/cs/Literature/Textbooks/ISPFedit.pdf

Noch mehr dazu unter dem Link

http://www-306.ibm.com/software/awdtools/ispf/library

Einführungsdokumente outside IBM

http://docweb.cns.ufl.edu/docs/d0089/d0089.html
http://eits.uga.edu/ExpressionEngine/files/documents/TSO_and_ISPF.pdf

Anhang A -- COMMON EDIT "PREFIX" COMMANDS (line commands)

To try out some of the following prefix commands, position the cursor in the line number (prefix) area on the left side of the screen while you are in SPF option 2 (EDIT), type the command and press <enter>.

I	**Insert one line.**
I3	**Insert 3 lines.**
D	**Delete one line.**
D3	**Delete 3 lines.**
DD	**Delete a group of lines ending with the next line also marked with DD.**
C	**Copy one line before (B) or after (A) another line marked with an A or a B line command.**
C2	**Copy 2 lines (this line and the next line) before or after another line marked with an A or a B line command.**
CC	**Copy a group of lines beginning with this line and ending with the next line also Marked with CC before or after another line marked with an A or a B line command.**
M	**Same as C, except it moves the line rather than copying it.**
M2	**Same as C2, except it moves the lines instead of copying them.**
MM	**Same as CC, except it moves the group of lines.**
R	**Repeat a line.**
R4	**Repeat a line 4 times.**
RR	**Repeat a series of lines ending with the next line marked with RR.**
RR2	**Same as RR, but repeat the series of lines twice.**
TE	**Set up the screen for text entry. Press "new line" to move to the next line. Press <enter> to end text entry mode.**
TS	**Split the line at the point where the cursor is positioned when <enter> is pressed. Flow the text TF evenly to the line length of the data set.**
TF70	**Same as TF, but end each line at column 70 or before.**

COLS	**Display a line indicating column positions. Delete the COLS line with the D prefix area command.**
BNDS	**Display and allow changes to bounds.**
TABS	**Display and allow changes to tab settings.**
)10	**Shift the data on the line to the right 10 columns.**
(10	**Shift the data on the line to the left 10 columns.**
))5	**Shift the data on the group of lines ending in the next line marked with)) to the right 5columns.**
((5	**Shift the data on the group of lines ending in the next line marked with ((to the left 5 columns.**
>5	**Shift the data - after the first blank - on the line to the right 5 columns.**
<5	**Shift the data - after the first blank - on the line to the left 5 columns.**
>>5	**Shift the data - after the first blank - on the group of lines ending in the next line marked with a >> to the right 5 columns.**
<<5	**Shift the data - after the first blank - on the group of lines ending in the next line marked with a << to the left 5 columns.**

41.

Anhang B -- COMMON EDIT "PRIMARY" COMMANDS

The following list of commands are entered on the primary COMMAND line while in EDIT. (Some of the commands may also be used on the COMMAND line elsewhere in SPF, such as in BROWSE.)

SAVE Save the data set (or data set member) and remain in EDIT mode.

CANCEL Exit EDIT, but do not save any changes made to the data set during the current EDIT session. (Note: You are actually editing a copy of the data set.)

PROFILE Display the profile, or characteristics, of this data set. Once displayed, the characteristics may be changed by typing over them.

CAPS OFF Change the profile for this data set so that subsequent characters are not converted to uppercase. The profile CAPS ON/OFF is automatically set depending on the ontents of a new data set.

NULLS ON Modify the profile for this data set so that the end of every line is filled with nulls instead of blanks. This makes it much easier to insert characters within lines. Especially useful if data set includes text.

HEX ON/OFF Change the profile for this data set to display/not display the hexadecimal characters for each character as well.

RESET Remove extraneous information lines, such as the three lines of profile display or a column positions line.

FIND string Find and display the next occurrence of the string. Surrounding apostrophes are necessary if the string contains any blanks or certain special characters (such as * or "). As is, the FIND does not distinguish capital letters (i.e., 'Xxx', 'X)X', and 'xxx' are the considered the same string). The FIND can be repeated by pressing the RFIND (repeat find) key (PF5).

F c'Xxx' Find the next occurrence of the string 'Xxx'. Specifying a string as a character string enables FIND to distinguish between uppercase and lowercase letters (i.e., 'Xxx', 'XXX', 'xxx' are not considered the same string).

F xxx WORD Find the next occurrence of the string 'xxx' that is not part of another word.

F xxx PREV Find the previous occurrence of the string 'xxx'.

F xxx 1 7 Find the next occurrence of the string 'xxx' in columns 1 through 7. The string must be completely contained within these columns.

CHANGE xxx yyy Change the next occurrence of the string 'xxx' to 'yyy'. Special characters are handled the same as for FIND. The CHANGE can be repeated by pressing the RCHANGE (repeat change) key (PF6).

C xx yy ALL Change all occurrences of 'xx' to 'yy'. Be careful when doing this. (You might want to SAVE the data set before using this command.)

COPY mem Copy a member of the same partitioned data set into this member. Use an A or a B line command (in prefix area) to indicate where the member should be placed.

COPY Copy a part of another member within the same partitioned data set or copy all or part of a member of another partitioned data set or copy all or part of a sequential data set. When you press <enter> a "copy from" menu will be displayed for you to enter the name of the data set and possibly the member you want to copy and, optionally, the lines you want to copy from that data set or member. First, use an A or a B line command to indicate where the copied lines should be placed.

CREATE mem Create a new member of this partitioned data set. Use the C (or CC) line command to indicate what to copy into the other member.

CREATE Create a new member of this partitioned data set, copy into an existing member of any partitioned data set, or copy into a sequential data set. Use the C (or CC) line command to indicate what to copy into the other member or data set. When you press <enter> a menu will be displayed for you to enter the name of the data set and/or member you want to copy into. (The data set must already exist.)

SUBMIT Submit a background job. The data set must contain appropriate Job Control Language. You will want to have NOTIFY=logonid on your JOB statement so that you will be notified when your job ends. You may also want to have MSGCLASS=S on your JOB statement. MSGCLASS=S will cause the output of your job to be placed in the held output queue so that you can view it from IOF (SPF option I)

TABS ALL/OFF Turns on/off hardware tabbing so that the tabs you set via the TABS line command will/will not be in effect.

1.c ISPF Subsystem

Version 13, Juli 2012

ISPF steht für Interactive System Productivity Facility. Es ist ein auf das TSO (Time Sharing Option) aufgesetztes Subsystem. Die allermeisten Funktionen der TSO Shell können auch unter ISPF ausgeführt werden. Als Folge benutzt fast jedermann ISPF, so lange das möglich ist. Die direkte Ausführung von TSO Commands erfolgt nur in Ausnahmefällen.

ISPF unterscheidet keine Groß- und Kleinbuchstaben. Alle einzugebenden Parameter, Dataset-Namen oder ISPF-Kommandos können groß- oder kleingeschrieben werden. Sogar eine Mischung aus beidem führt zu keinem Fehler. Beispiele für zulässige Eingaben:

- **"a" oder "A" (allocate)**
- **"HELP", "help" oder "heLP" (Hilfefunktion aufrufen)**
- **"PRAK025.TEST.NEW", "prak025. test.new" oder "PrAk025. TeST.nEW" . Dies zeigt den Namen von ein- und demselben Dataset.**

Inhalt

1. ISPF Einführung

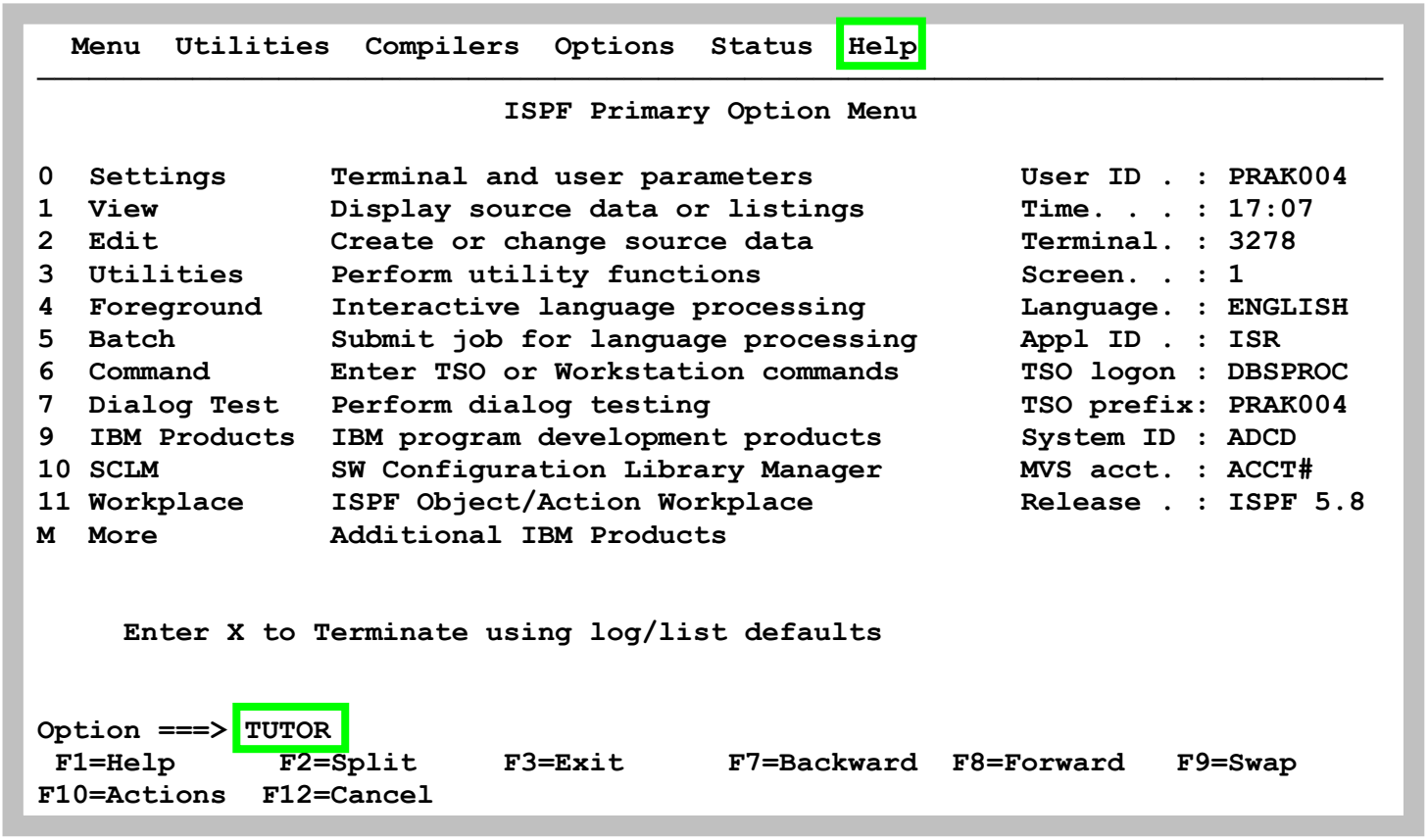
1.1. ISPF Tutorial

Sehr nützlich für die Arbeit mit ISPF sind dessen Hilfefunktionen.

Vorteilhaft ist das Aufrufen passender Hilfen zu den verschiedenen ISPF-Funktionen aus den entsprechenden ISPF-Panels heraus. Möchte man in einem beliebigen Panel die passende Hilfe öffnen, so ist von diesem Panel aus auf der Kommandozeile "HELP" einzugeben oder alternativ dazu einfach die Funktionstaste F1 zu betätigen. Dass diese Funktionstaste mit "Help" belegt ist, darauf wird in den meisten Panels auch dadurch hingewiesen, dass in der linken unteren Ecke "`F1=Help`" steht.

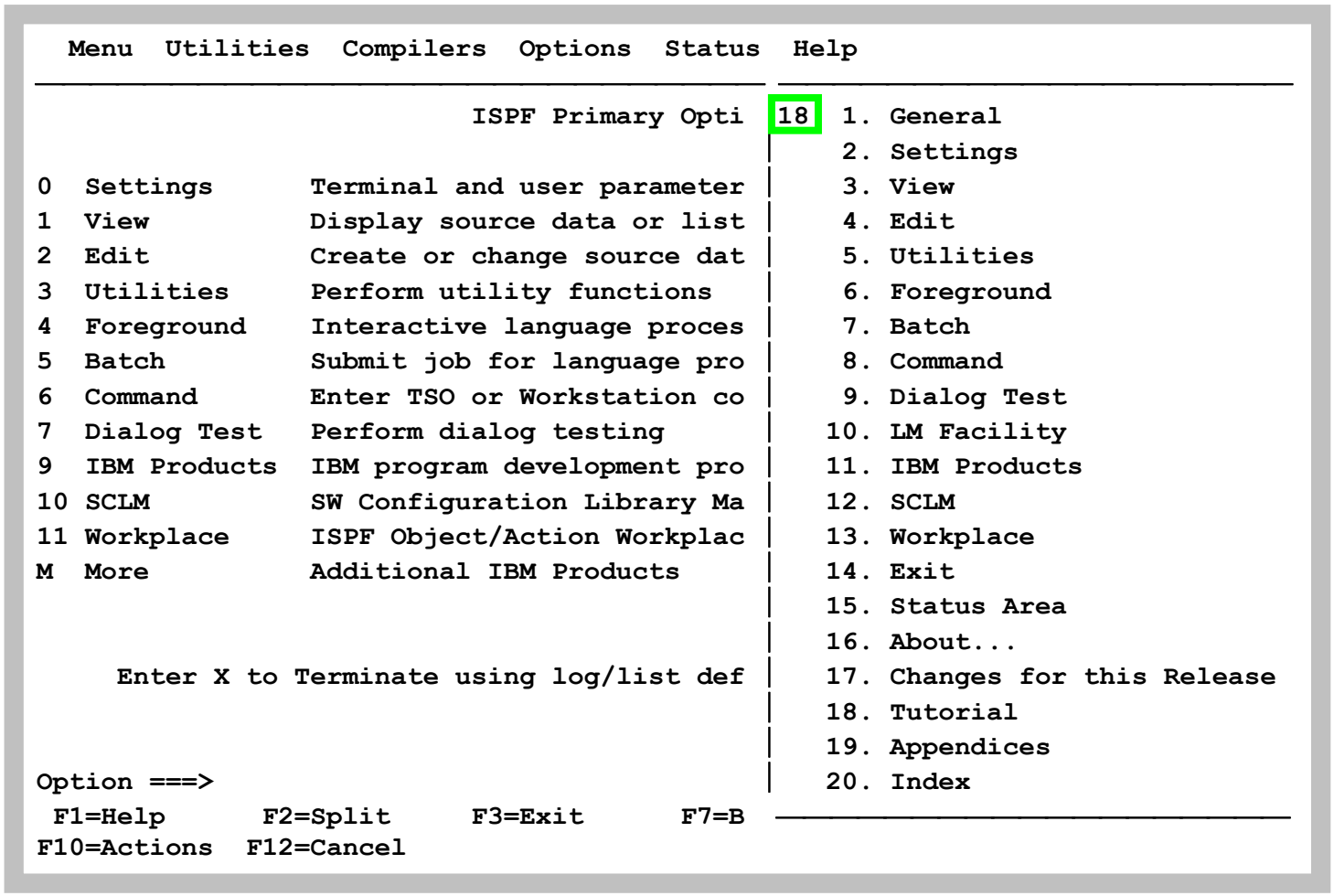
Eine Kommandozeile erkennt man an "`Command ===>`" oder an "`Option ===>`".

Als Panel bezeichnet man das standardmäßig schwarz aussehende Mainframe-Text-Fenster mit seinen 24 (oder 32) Zeilen sowie 80 Spalten.

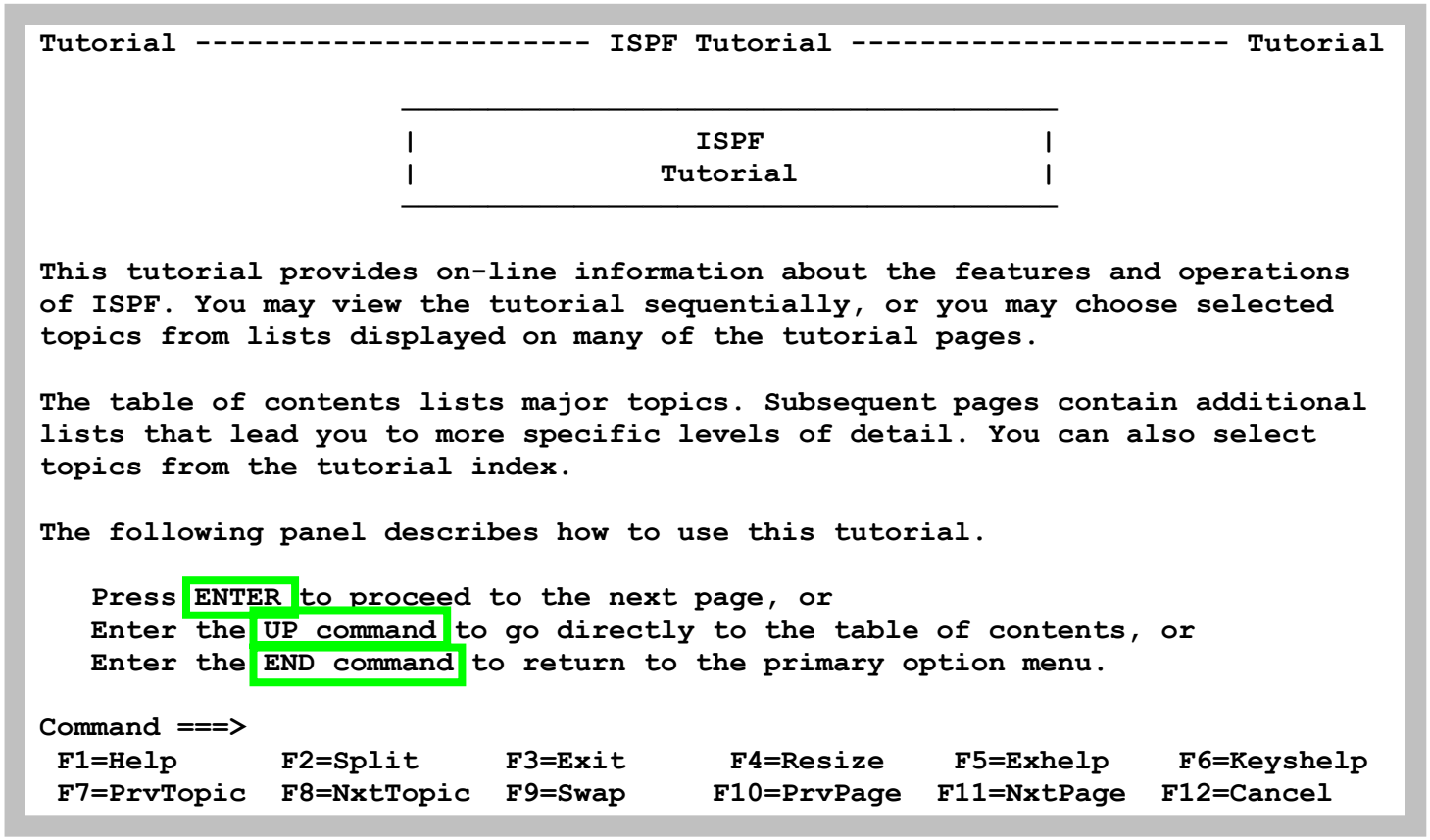


Zu der Hilfefunktion gehört ein Tutorial, welches Bestandteil von ISPF ist. Wie kann man das ISPF Tutorial aufrufen ?

1. **In die Kommandozeile "TUTOR" oder "tutor", gefolgt von der Eingabetaste, eingeben, oder**
2. **Den Cursor in die erste Zeile genau auf "Help" stellen, mittels Eingabetaste das Pull Down Menü öffnen, nun "18" eintippen, Eingabetaste betätigen.**



Beide Alternativen bewirken den Aufruf des ISPF internen Tutorials.



Gezeigt ist der ISPF Tutorial Eingangsscreen. Die Enter Taste blättert zur nächsten Seite.

```
TUTORIAL --------- MOVE/COPY UTILITY - "FROM" DATA SET PANEL -------- TUTORIAL

  To perform a move or copy operation, fill in the following fields on the
  first move/copy utility panel and press the ENTER key:

    - Enter the move/copy option in the option field:
        - C to copy              - CP to copy and print
        - M to move              - MP to move and print
        - L to copy and LMF lock - LP to copy, LMF lock, and print
        - P to LMF promote       - PP to LMF promote and print

    - Enter the "from" library information in the appropriate fields.
    - If the "from" data set is partitioned, enter a member name as follows:
          - to move,copy or promote a single member, enter the member name.
          - to move, copy or promote all members,  enter * (asterisk).
          - to request a member selection list, leave member name blank or
            specify a pattern.

The following topics will be presented only if selected by number:
    1 - How to enter the library or data set information
    2 - Member name patterns
OPTION  ===>
 F1=Help      F3=Exit      F5=Exhelp    F6=Keyshelp  F7=PrvTopic  F8=NxtTopic
F10=PrvPage  F11=NxtPage  F12=Cancel
```

Die letzten beiden Zeilen des Panels weisen den Nutzer auf hier aktive Funktionstasten und deren Belegung hin. Der Anwender kann beispielsweise F3 betätigen, um so das Hilfe-Tutorial zu verlassen (="Exit"). Er kann aber auch F11 drücken, um so zur nächsten Tutorial-Seite zu blättern (="NxtPage") oder F10 betätigen, um eine Seite zurückzublättern (="PrvPage").

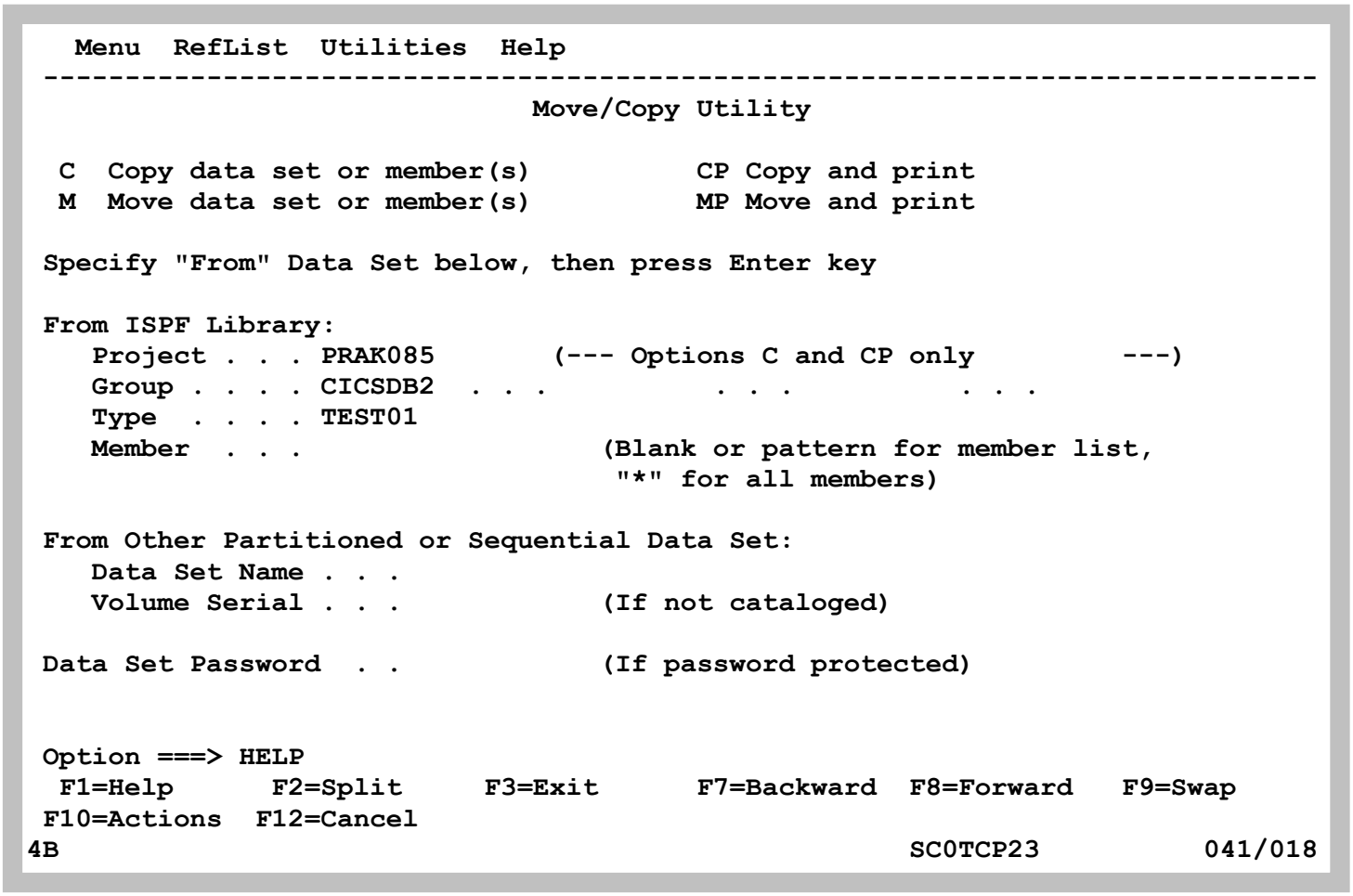
<u>Aufgabe:</u> Gehen Sie das eingebaute ISPF Tutorial durch.

1.2 Benutzung der ISPF-Hilfe

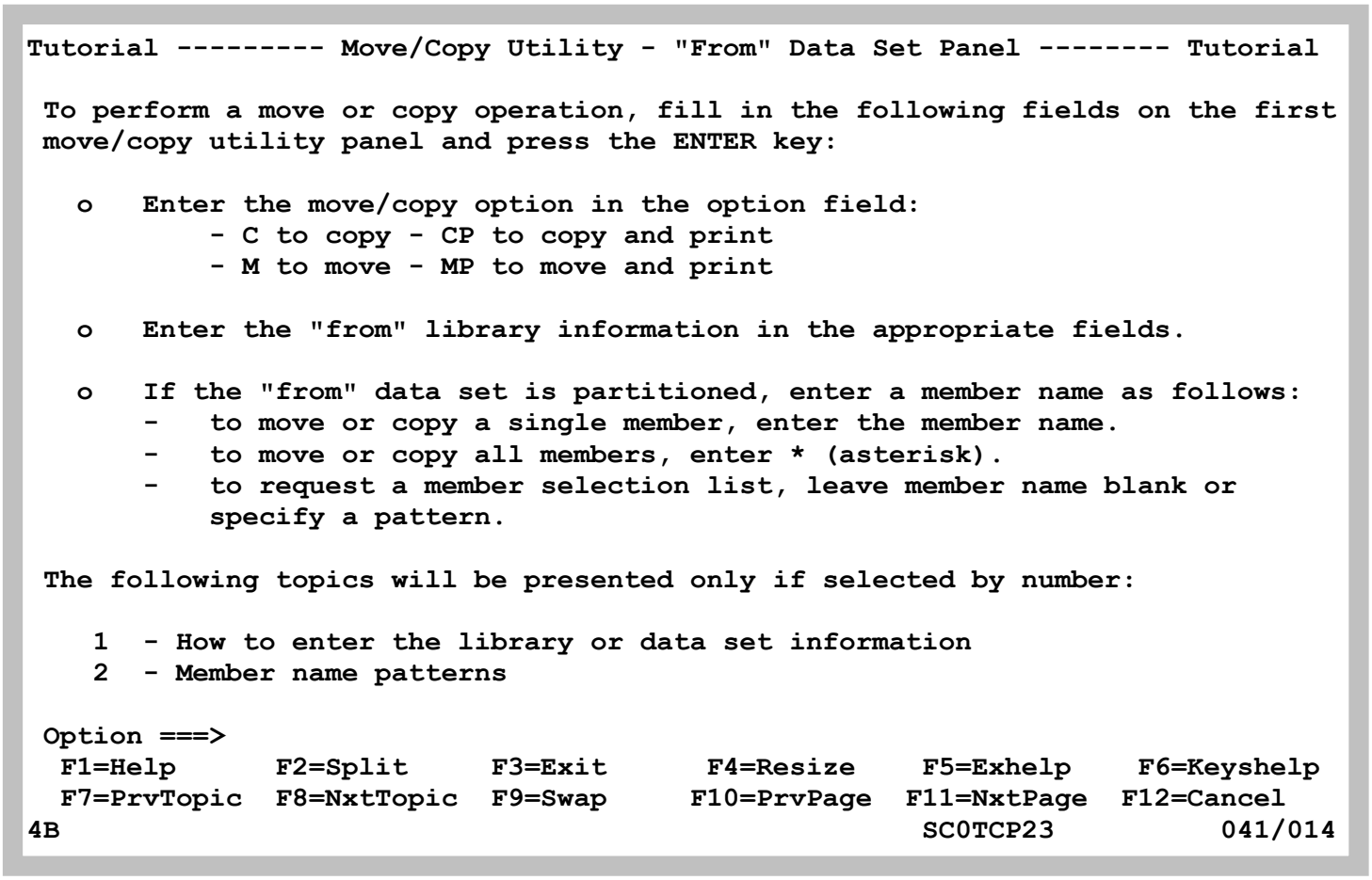
Noch wichtiger als eine allgemeine Beschäftigung mit dem ISPF-Tutorial ist das Aufrufen passender Hilfen zu den verschiedenen ISPF-Funktionen aus den entsprechenden ISPF-Panels heraus. Dazu gibt man auf der Kommando-Zeile groß oder klein "HELP" ein oder man betätigt einfach die Funktionstaste F1 (siehe Abschnitt 1.3 "Benutzung der Tasten F1 bis F12").

Als Panel bezeichnet man das standardmäßig schwarz aussehende Mainframe-Text-Fenster mit seinen 24 (oder 32) Zeilen sowie 80 Spalten.

Möchte man z.B. zu dem in der folgenden Abbildung gezeigten Panel die passende Hilfe öffnen, so ist von diesem Panel aus auf der Kommandozeile "HELP" einzugeben oder alternativ dazu einfach die Funktionstaste F1 zu betätigen. Dass diese Funktionstaste mit "Help" belegt ist, darauf wird man auch durch das in der Abbildung 5 gezeigte Panel hingewiesen. In der linken unteren Ecke steht nämlich "F1=Help".



Aufruf der zu einem Panel passenden Hilfe. Es öffnet sich genau die zum Panel passende Hilfe



Zu dem Panel aus de vorhergehenden Abbildung passende Hilfe

Die letzten beiden Zeilen des Panels in der obigen Abbildung weisen den Nutzer auf hier aktive Funktionstasten und deren Belegung hin. Der Anwender kann beispielsweise F3 betätigen, um so das Hilfe-Tutorial zu verlassen (="`Exit`"). Er kann aber auch F11 drücken, um so zur nächsten Tutorial-Seite zu blättern (="`NxtPage`") oder F10 betätigen, um eine Seite zurückzublättern (="`PrvPage`")..

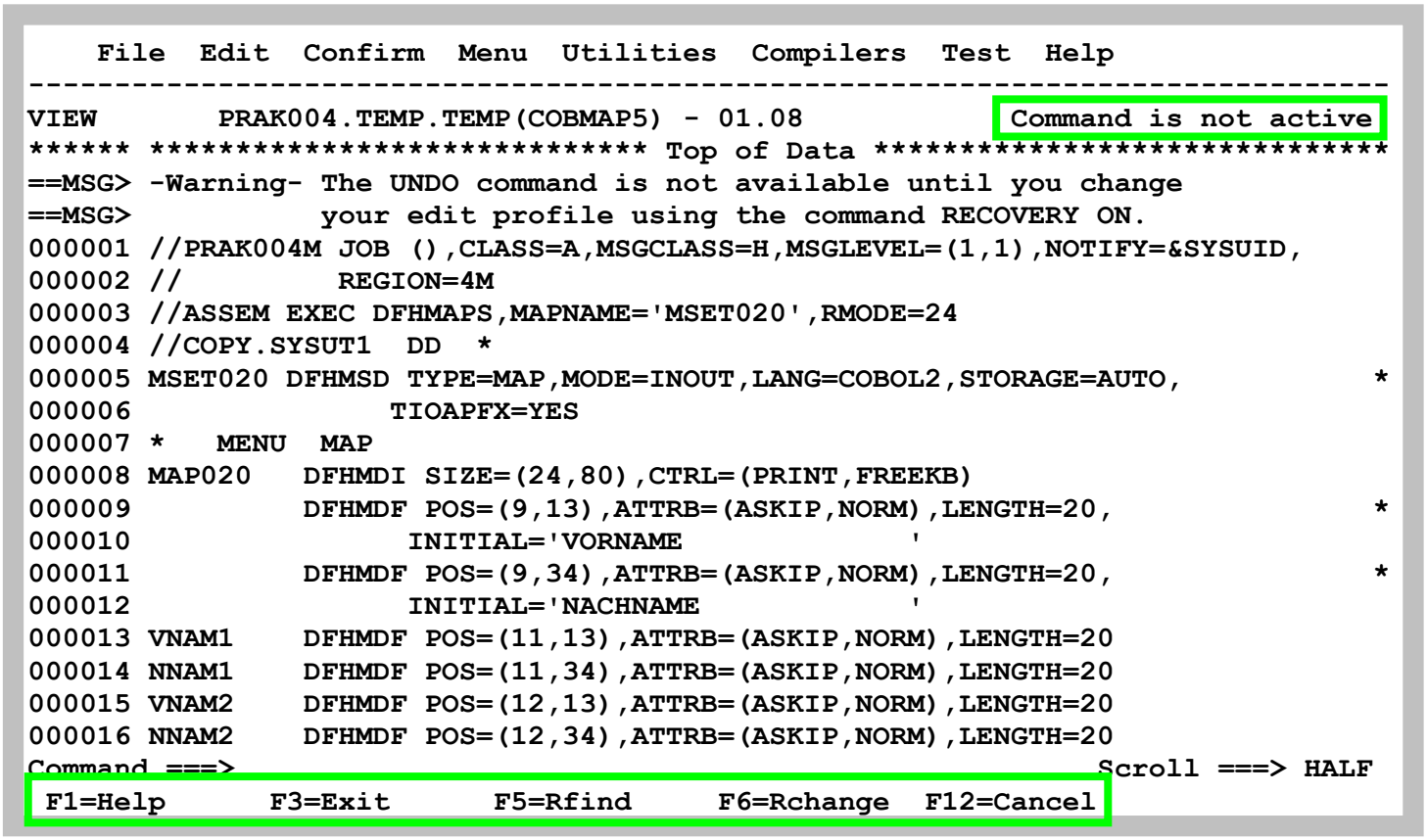
1.3. Benutzung der Tasten F1 bis F12

Heutige Tastaturen haben sogenannte Funktionstasten (Function Keys) F1 bis F12. Bei der Betätigung einer solchen Taste wird ein bestimmtes ISPF-Kommando ausgeführt. Der Benutzer kann die Funktionstasten umprogrammieren, also einer jeden Taste ein neues ISPF-Kommando zuweisen. Doch ist dies meist nicht notwendig. Systemseitig sind die Funktionstasten F1 bis F12 oft mit den folgenden ISPF-Kommandos belegt:

Taste Kommando ISPF-Beschreibung der Funktionsweise des ISPF-Kommandos

F 1	**HELP**	**Die Hilfe-Funktion wird aufgerufen. Es erscheint ein Hilfetext zum Panel, von dem aus HELP aufgerufen wurde.**
F 2	**SPLIT**	**Das aktive Panel wird in zwei voneinander unabhängige Panels aufgeteilt.**
F 3	**END**	**Beenden der aktiven Funktion und Rückkehr in das nächst höhere Panel.**
F 4	**RETURN**	**Beenden der aktiven Funktion und Sprung ins ISPF Primary Option-Menü.**
F 5	**RFIND**	**Repeat FIND. Es wird ein Find-Kommando wiederholt, also z.B. die nächste Zeichenkette "exec" im angezeigten Text gesucht.**
F 6	**RCHANGE**	**Repeat CHANGE. Es wird das Change-Kommando wiederholt, also z.B. die nächste Zeichenkette "020" gesucht und durch "127" ersetzt.**
F 7	**UP**	**Scrolling eines Textes nach oben.**
F 8	**DOWN**	**Scrolling eines Textes nach unten.**
F 9	**SWAP**	**Nachdem (z.B. durch Betätigung von F2) aus einem Panel zwei voneinander unabhängige Panels erzeugt wurden, kann man per F9 zwischen den beiden Panels wechseln.**
F 10	**LEFT**	**Scrolling eines Textes nach links.**
F 11	**RIGHT**	**Scrolling eines Textes nach rechts.**
F 12	**RETRIEVE**	**Anzeige des vorigen Panels, um in diesem eventuell fehlerhafte Eingabewerte korrigieren zu können und um anschließend per Eingabetaste die zuletzt ausgeführte Funktion zu wiederholen.**

Nicht in jedem Panel sind alle Funktionstasten benutzbar. Sollte einmal eine Funktionstaste nicht benutzbar sein, erscheint links oben im Panel "`Command is not active`" (siehe unten).



Auf einige der gerade benutzbaren Funktionstasten wird in der letzten Zeile oder den letzten Zeilen eines Panels hingewiesen. Doch gibt es häufig auch benutzbare Funktionstasten, auf die dort nicht hingewiesen wird.

3. Die wichtigsten ISPF-Kommandos, die standardmäßig per Funktionstaste aufgerufen werden können

F2 (SPLIT) und F9 (SWAP)

Möchte man mit zwei voneinander unabhängigen Panels arbeiten, so stellt man den Cursor in die Zeile, oberhalb derer das erste und unterhalb derer das zweite Panel entstehen soll. Die Taste F2 teilt den Screen entsprechend. Mit der Taste F9 kann man nun beliebig oft zwischen den beiden Panels wechseln.

Möchte man beide Panel in maximaler Größe nutzen, dann ist der Cursor auf die erste Zeile des Panels zu stellen und anschließend F2 zu betätigen. Es erscheint ein neues unabhängiges Panel. Ein Wechsel zwischen diesem und dem alten Panel ist auch hier jederzeit per Taste F9 möglich.

F5 (RFIND)

Möchte man in einem (z.B. im ISPF-Editor) geöffneten Text oder in einer angezeigten Logfile eine bestimmte Zeichenkette *mehrfach* finden, ist F5 sehr nützlich.

Man gibt in die Kommandozeile "find exec" oder abgekürzt "`f exec`" ein. Das anschließende Betätigen der Eingabetaste findet die erste Zeichenkette "`exec`" im Text. An der Fundstelle steht der Cursor.

Möchte man das nächste "`exec`" im Text finden, reicht es jetzt, die Taste F5 zu betätigen. Der Cursor steht anschließend über dem zweiten gefundenen "`exec`" und rechts oben steht im Panel "`CHARS 'EXEC' FOUND`". Nach jedem F5 wird das nächste "`exec`" gesucht und der Cursor zeigt auf die Fundstelle. Erst wenn sich kein "`exec`" mehr findet, erscheint rechts oben im Panel
`*BOTTOM OF DATA REACHED*.`

F6 (RCHANGE)

Die Anwendung der Taste F6 erfolgt ähnlich der Anwendung von F5. F6 wiederholt einen Zeichenketten-Ersetzungsvorgang (Repeat CHANGE).

Z.B. kann die erste gefundene Zeichenkette "`127`" durch "`020`" mittels "`change 127 020`" oder kurz "`c 127 020`" ersetzt werden. Der Cursor steht anschließend an der Ersetzungsstelle und rechts oben erscheint im Panel die Meldung "`CHARS '127' changed`".

Um die zweite Zeichenkette "`127`" durch "`020`" zu ersetzen, reicht nun die Taste F6 aus! Wieder steht der Cursor neben der ersetzten Zeichenkette. So lässt sich mit jedem F6-Tastendruck eine "`127`" ersetzen. Wird keine "`127`" mehr gefunden, wird rechts oben im Panel `*Bottom of data reached*` ausgegeben.

F7, F8, F10, F11 (Scrolling)

Mit diesen Tasten ist ein Scrollen in einem Text, der größer als ein Panel ist, möglich: F7 scrolled hoch, F8 herunter, F10 nach links und F11 nach rechts.

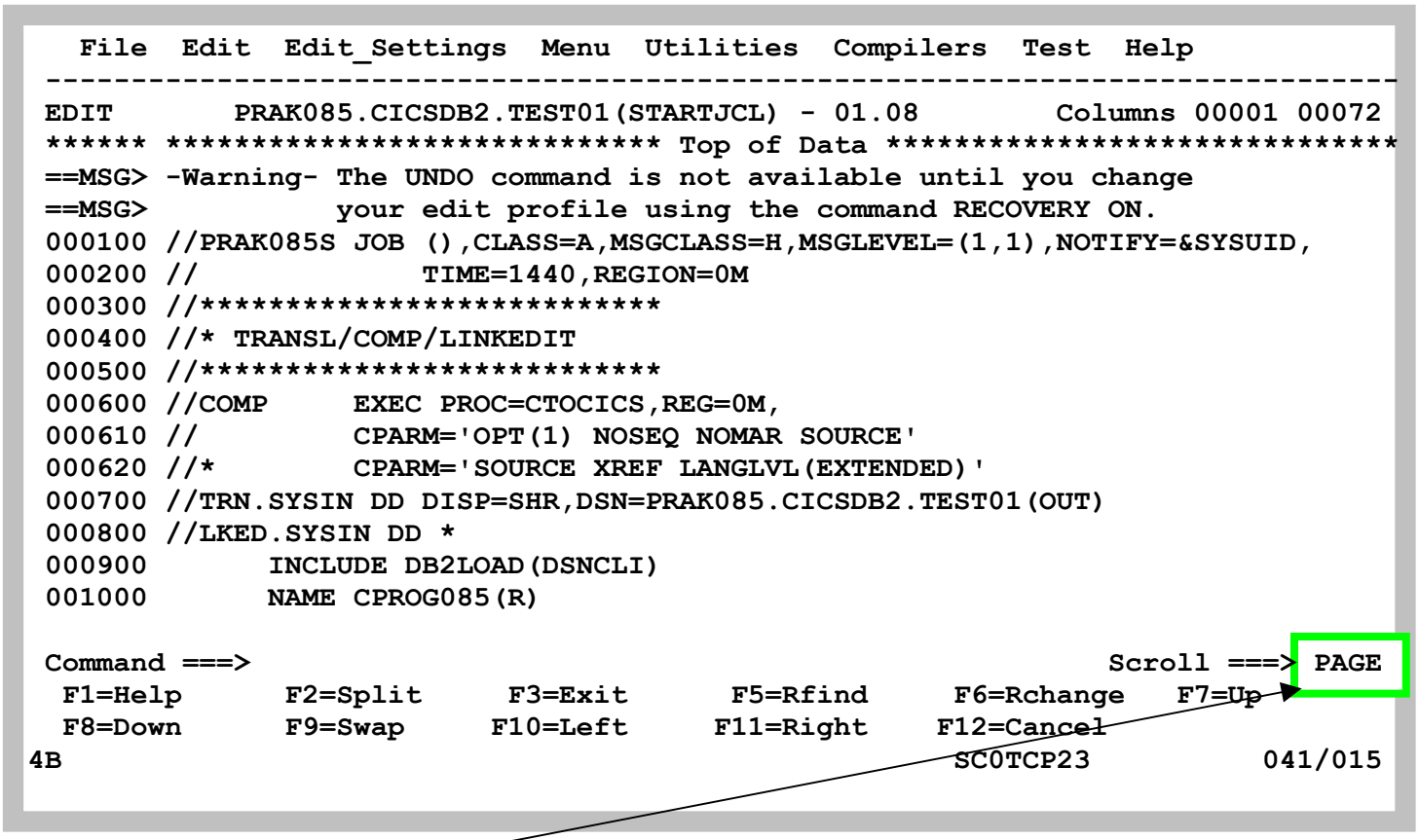
Folgende Werte sind möglich:

PAGE = Ein Tastendruck ersetzt die komplette angezeigte Seite.

HALF = Die halbe Seite des Textes wird hinausgescrolled, eine neue halbe Seite Text erscheint.

DATA = Fast die ganze alte Seite wird hinausgescrolled, lediglich eine alte Zeile / alte Spalte bleibt nach dem Tastendruck noch auf dem Panel sichtbar.

<zahl> = <zahl> steht für eine konkrete Zahl, die ebenfalls in das Feld "Scroll" eingetragen werden kann. Um <zahl> Zeilen oder Spalten wird dann der Text pro Tastendruck gescrolled. Eine "3" in diesem Feld bewirkt z.B., dass drei alte Zeilen oder Spalten heraus und drei neue Zeilen oder Spalten hineingescrolled werden.



Rechts unten befindet sich in den meisten ISPF-Panels ein Feld "`Scroll`" . Hier läßt sich einstellen, um wieviele Zeilen oder Spalten je Tastendruck gescrollt werden soll.

Folgende Werte sind möglich:

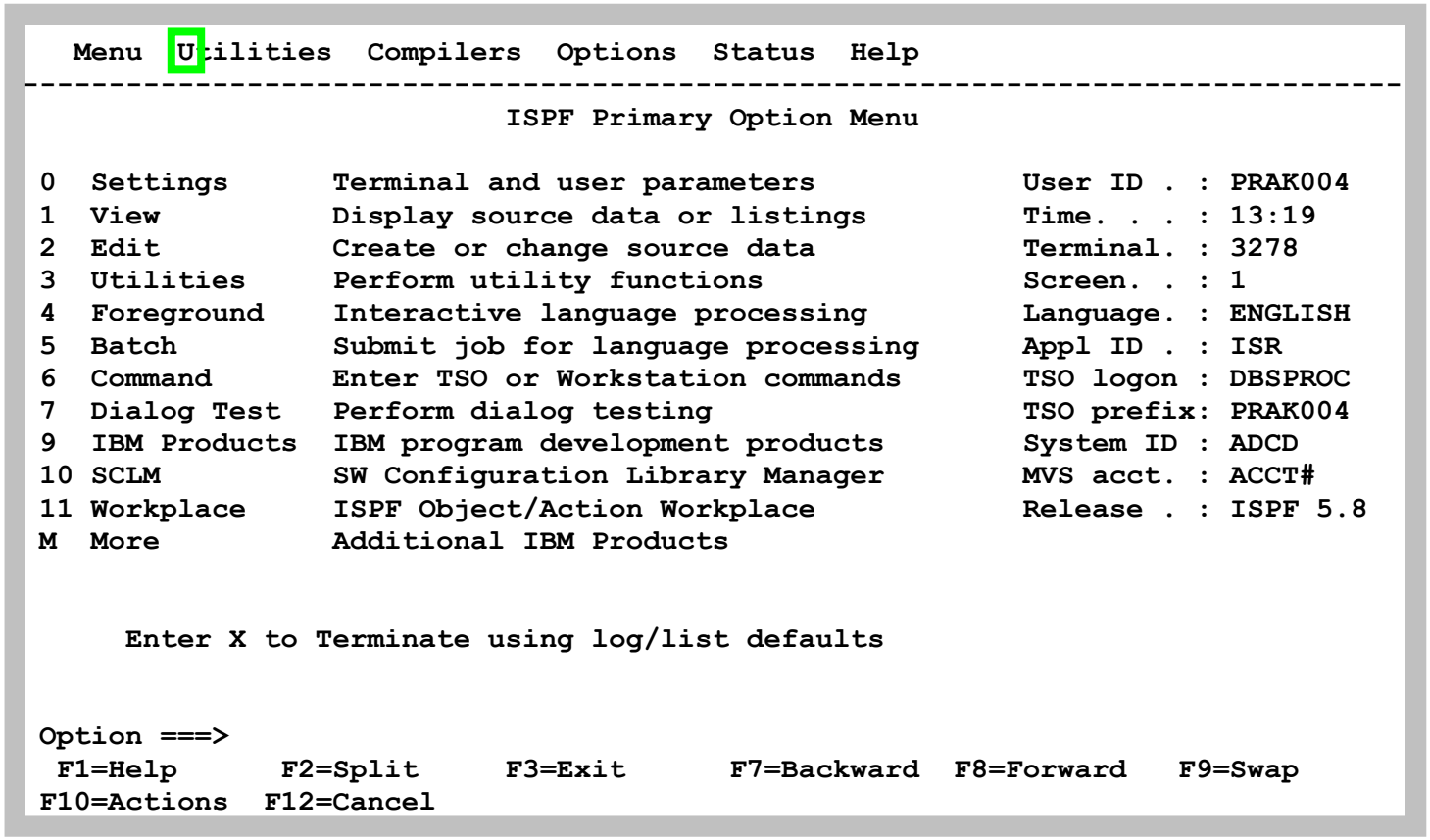
PAGE = Ein Tastendruck ersetzt die komplette angezeigte Seite.

HALF = Die halbe Seite des Textes wird hinausgescrollt, eine neue halbe Seite Text erscheint.

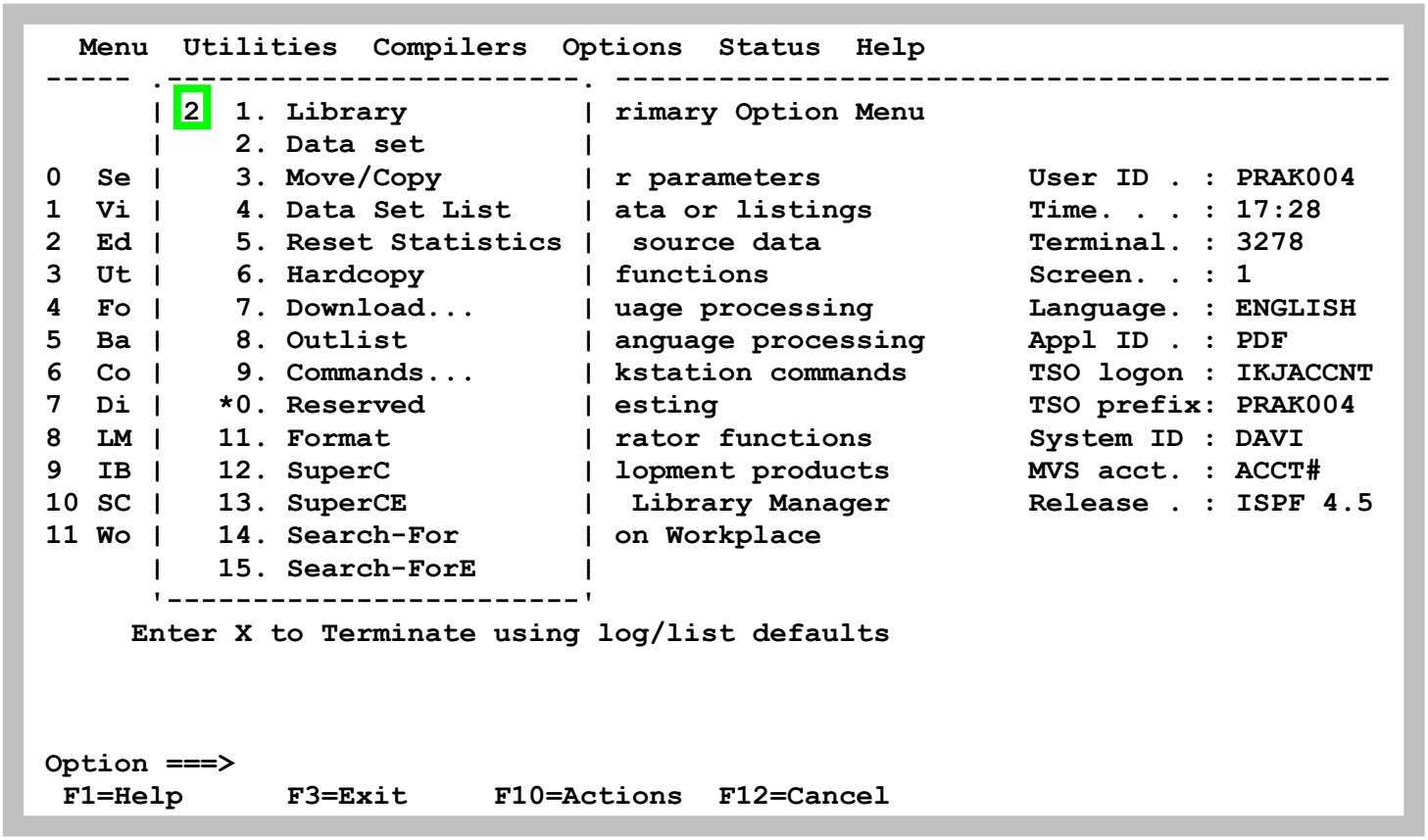
DATA = Fast die ganze alte Seite wird hinausgescrollt, lediglich eine alte Zeile / alte Spalte bleibt nach dem Tastendruck noch auf dem Panel sichtbar.

<zahl> = <zahl> steht für eine konkrete Zahl, die ebenfalls in das Feld "Scroll" eingetragen werden kann. Um <zahl> Zeilen oder Spalten wird dann der Text pro Tastendruck gescrollt. Eine "3" in diesem Feld bewirkt z.B., dass drei alte Zeilen oder Spalten heraus und drei neue Zeilen oder Spalten hineingescrollt werden.

1.4. Alternative Navigation



Es gibt noch einen alternativen Weg, wie man vom Hauptpanel des ISPF (ISPF Primary Option-Menü) ins Data Set Utility gelangen kann: Dabei kann man die erste Zeile des ISPF Primary Option Menu-Panels, die Action Bar, benutzen. Dazu stellt man den Cursor mit der Maus auf "Utilities" in diese Action Bar. Die Eingabetaste öffnet ein Pull Down-Menü.



Hier gibt man eine "2", gefolgt von der Eingabetaste, ein

Dies ist ein Beispiel dafür, dass es im ISPF oft viele Wege gibt, um in ein bestimmtes Tool / Panel zu gelangen.

2. Data Set List Utitility

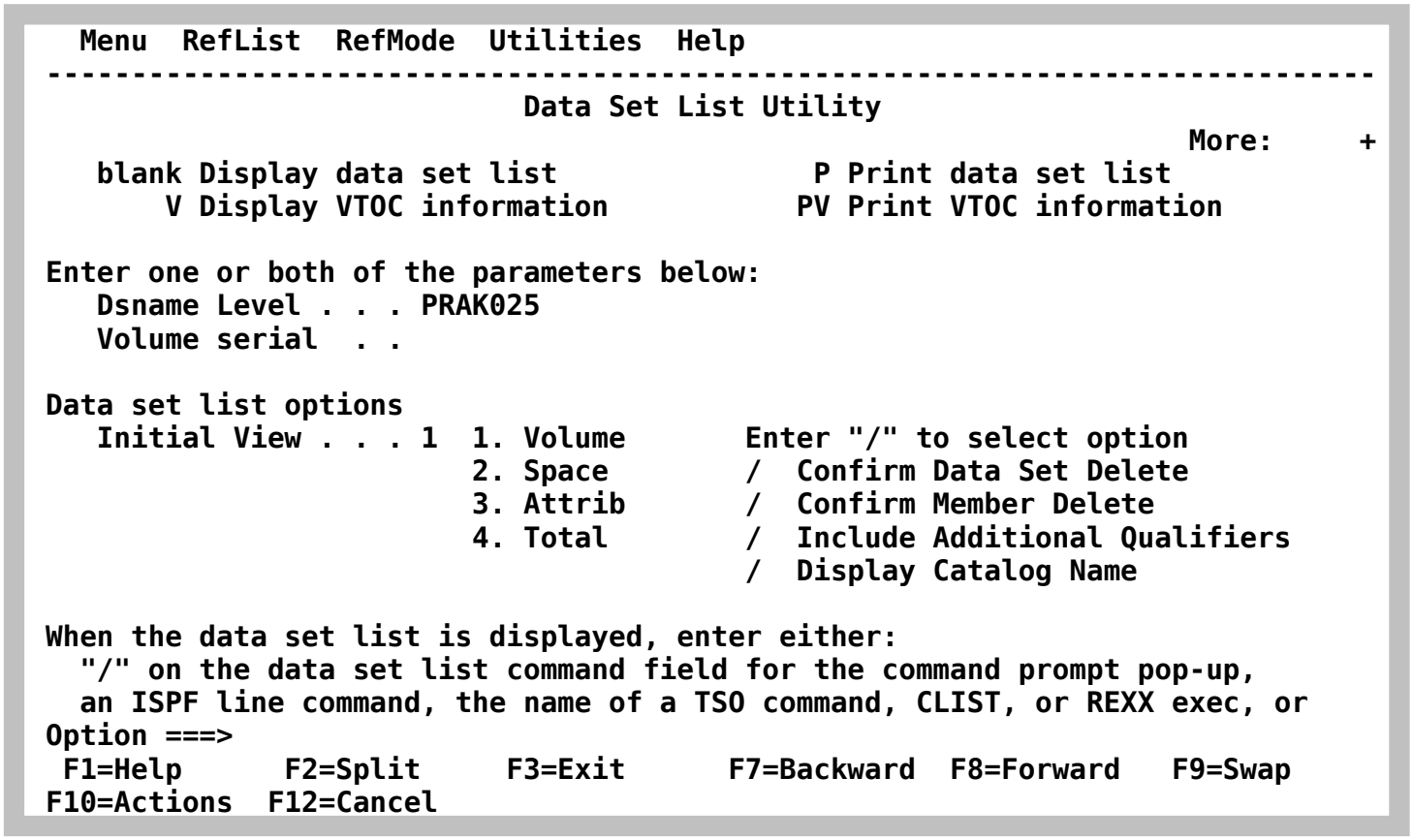
Die Data Set List Utility ist nützlich, um

1. **sich eine Liste von Datasets anzeigen zu lassen,**
2. **sich die existierenden Member eines Datasets anzeigen zu lassen,**
3. **Member zum Zwecke der Ansicht oder Modifikation zu öffnen,**
4. **Member zu kopieren oder zu verschieben,**
5. **sich die Eigenschaften eines Datasets anzusehen.**

Um das Dataset List Utility vom ISPF Primary Option-Menü aus zu starten, wählt man vom ISPF Primary Option-Menü zuerst "3" (Utilities) und im sich anschließend öffnenden Utility Selection Panel "4" (Dslist) aus. Abkürzend kann man auch hier auf der Kommandozeile des ISPF Primary Option-Menüs "3 . 4" eingeben.

<u>Aufgabe:</u> Sie haben unter Ihrer User ID PRAK025 einen Data Set PRAK025.TEST.DATASET erzeugt und hierfür einen Member MEMBER1 und MEMBER2 erzeugt. Als Vorbereitung für die folgende Übung legen wir einen neuen Data Set PRAK025.TEST.NEW an (allocate). ErstellenSie hierfür einen Member MEMBER3 und füllen ihn mit irgendwelchen Daten. Ersetzen sie überall PRAK025 durch ihre eigene User ID.

2.1 Eine Liste von Datasets anzeigen



Haupt Panel der Data Set List Utility

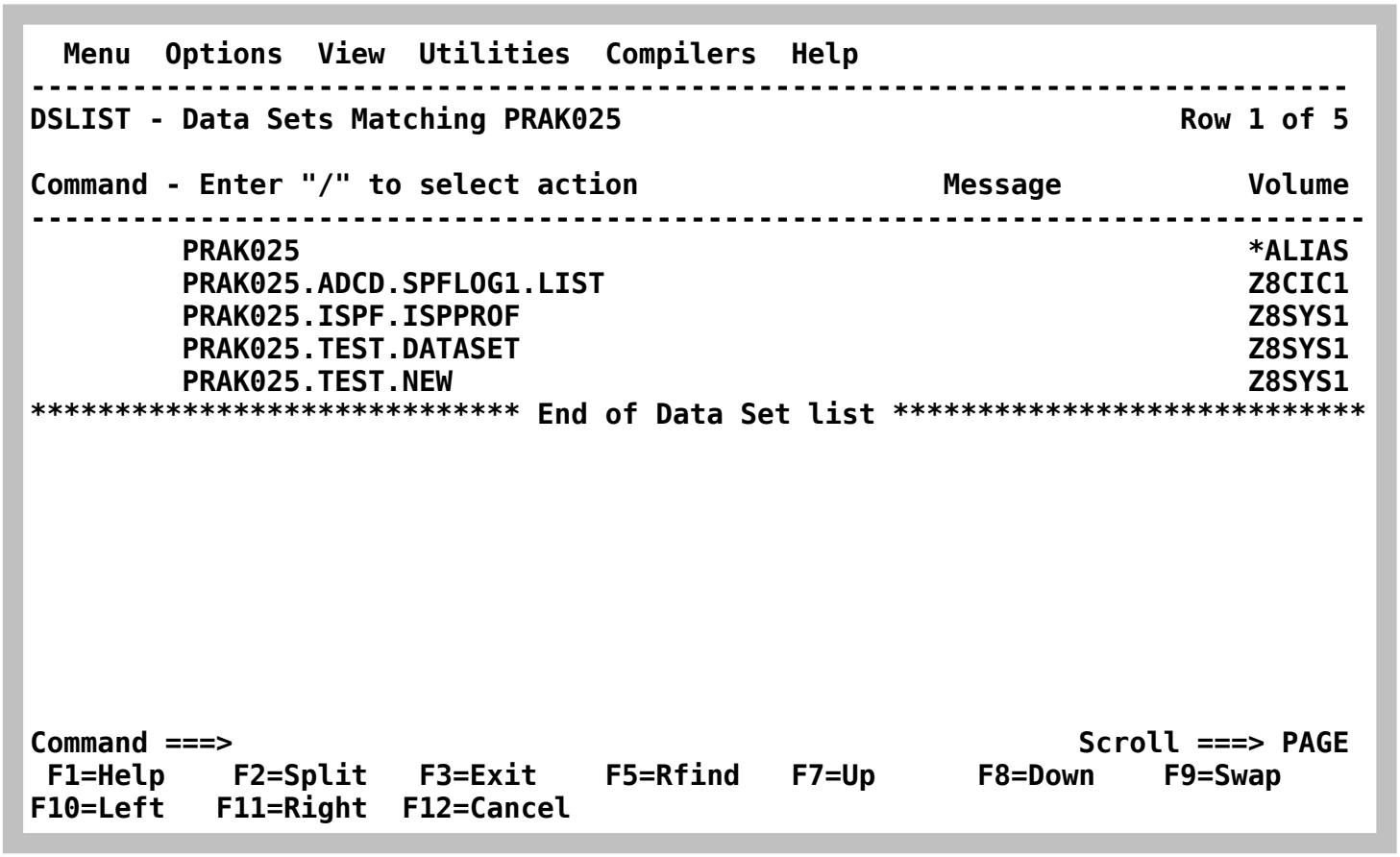
Die Dataset List Utility meldet sich mit dem hier gezeigten Panel. Hier sind eine ganze Reihe von Funktionen möglich.

Um nun eine Liste aller eigenen Datasets oder eine Liste einer bestimmten Auswahl seiner eigenen Datasets anzeigen zu können, ist ein passender Wert in das Feld "`Dsname Level`" einzutragen.

Möchten Sie alle Ihre Datasets Ihres PRAKxxx-Accounts anzeigen lassen, so ist in das Feld "`Dsname Level`" der erste Teil Ihrer Dataset-Namen, z.B. "`PRAK025`", einzutragen (siehe oben).

Möchten Sie einen Teil Ihrer Datasets anzeigen, z.B. alle "PRAK025.TEST"-Datasets, so ist in das Feld "`Dsname Level`" der Wert "`PRAK025.TEST`" einzutragen.

Enter



Die beiden Data Sets `PRAK025.ADCD.SPFLOG1.LIST` und `PRAK025.ISPF.ISPPROF` hatte das System bei der Erstellung Ihres Accounts automatisch angelegt. `PRAK025.TEST.DATASET` und `PRAK025.TEST.NEW` wurde von Ihnen angelegt (allocated).

2.2 Die existierenden Member eines Datasets anzeigen

Möchte man die Member des Datasets `PRAK025.TEST.NEW` anzeigen, so platziert man den Cursor links neben den Data Set Namen und gibt ein „V“ (für View) ein. Eine Liste der Member von `PRAK025.TEST.NEW erscheint.`

In diesem Fall haben wir bisher nur einen Member “MEMBER3” angelegt.

2.3 Member zur Ansicht oder Modifikation öffnen

Möchte man einen Member, der lesbaren Text enthält, zur Ansicht öffnen, so ist in der Member-Liste dieser Member auszuwählen. Dazu gibt man links neben den Member-Namen "`V`" (view), gefolgt von der Eingabetaste, ein (siehe oben). Der so geöffnete Member läßt sich nicht modifizieren.

Soll eine Modifikation erlaubt sein, dann ist in der Member-Liste links neben den Member anstatt von "`V`" der Buchstabe "`e`" (edit) einzugeben. Die Eingabetaste öffnet den nun modifizierbaren Text.

```
  Menu  Functions  Confirm  Utilities  Help
------------------------------------------------------------------------------
VIEW              PRAK025.TEST.NEW                        Row 00001 of 00001
           Name     Prompt       Size   Created          Changed          ID
___V_____ MEMBER3                   4  2012/11/13  2012/11/13 21:08:40  PRAK025
          **End**




Command ===>                                                  Scroll ===> PAGE
 F1=Help    F2=Split   F3=Exit    F5=Rfind   F7=Up      F8=Down    F9=Swap
F10=Left   F11=Right  F12=Cancel
```

2.4 Member kopieren

2.4.1 Einen Member von einem Dataset in einen anderen kopieren

Hierzu öffnen wir die Member Liste von PRAK025.TEST.DATASET

Möchte man einen Member aus der Liste der Member kopieren, so ist "C" (copy) links neben dessen Member-Namen einzutragen

```
  Menu  Functions  Confirm  Utilities  Help
------------------------------------------------------------------------------
VIEW              PRAK025.TEST.DATASET                      Row 00001 of 00002
           Name     Prompt        Size   Created          Changed          ID
___C_____ MEMBER1                    3  2012/11/13  2012/11/13 19:32:46  PRAK025
__________ MEMBER2                   4  2012/11/13  2012/11/13 19:35:57  PRAK025
           **End**



Command ===>                                                 Scroll ===> PAGE
 F1=Help     F2=Split    F3=Exit     F5=Rfind    F7=Up       F8=Down     F9=Swap
F10=Left    F11=Right   F12=Cancel
```

Die Eingabetaste führt in das Panel, in welchem das Ziel des Kopiervorganges festgelegt wird.

```
  RefList  Help
--------------------------------------------------------------------------------
                               COPY Entry Panel
                                                                      More:     +
CURRENT from data set: 'PRAK025.TEST.DATASET(MEMBER1)'

To Library                          Options:
   Project . . . PRAK025               Enter "/" to select option
   Group . . . . TEST                     Replace like-named members
   Type  . . . . NEW                   /  Process member aliases

To Other Data Set Name
   Data Set Name . . .
   Volume Serial . . .              (If not cataloged)

NEW member name  . . .              (Blank unless member to be renamed)

Options
   Sequential Disposition           Pack Option          SCLM Setting
   2  1. Mod                        1  1. Default        3  1. SCLM
      2. Old                           2. Pack              2. Non-SCLM
Command ===>
 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
F12=Cancel
```

Soll die Member-Kopie noch einen anderen Namen erhalten als das Member-Original, so ist in das Feld "NEW member name" noch der andere Name einzutragen. Soll der Name der Kopie zum Namen des Originals identisch sein, so kann das Feld "NEW member name" leer bleiben.

Die Eingabetaste schließt den Kopier-Vorgang ab. Als Bestätigung erscheint "*Copied". Überprüfen Sie, dass PRAK025.TEST.NEW nun einen zusätzlichen Member enthält.

2.4.2 Eine Kopie eines Members innerhalb eines Datasets erstellen

Wir möchten den Member "MEMBER3" innerhalb des Datasets PRAK025.TEST.NEW kopieren. Der neue Member soll MEMBER5 heißen.

Unter Nutzung des Data Set List-Utilitys lassen wir uns den Member des Datasets PRAK025.TEST.NEW auflisten. Wir stellen den Cursor links neben "MEMBER3", tippen ein "c" und schließen mit der Eingabetaste ab.

Im sich öffnenden "COPY Entry Panel" müssen in die "To Library"-Felder wieder die drei Teile des Ziel-Datasets eingetragen werden. Wir tragen ins Project-Feld "PRAK025", ins Group-Feld "TEST" sowie ins Type-Feld NEW" ein. Diesmal muss ebenfalls der neue Member-Name, also "MEMBER5", in das "NEW member name"-Feld eingetragen werden. Die Eingabetaste startet den Kopiervorgang, dessen Ende wird wieder mit "*Copied" quittiert.

Um uns nun die beiden Member des Datasets PRAK025.TEST.NEW anzusehen, müssen wir dieses Panel auf folgendem Weg aktualisieren:

Funktionstaste F3 → Eingabetaste → "v" (="Member List" auswählen) → Eingabetaste

2.4.3 Mehrere Member aus dem gleichen Dataset an das gleiche Ziel kopieren

Nach einem erfolgreichen Kopieren eines Members unterstützt das Data Set List-Utility, dass weitere Member aus der gleichen Quelle an das gleiche Ziel kopiert werden können, ohne dass das Ziel noch einmal explizit angegeben werden muss.

Möchten wir im direkten Anschluss an obigen Kopiervorgang einen weiteren Member kopieren, so ist in die Liste der Member wieder ein "C" links neben den Member Namen einzutragen und anschließend die Eingabetaste zu betätigen. Das Ziel des Kopiervorganges wird nicht noch einmal abgefragt (!), der Member wird statt dessen sofort kopiert.

Nicht nur ein Member, sondern mehrere Member lassen sich auf diese Weise in einem Vorgang kopieren. Wollen wir drei Member in einem Vorgang kopieren, so ist das "c" links neben alle drei Member-Namen einzutragen und anschließend die Eingabetaste zu betätigen. Alle drei Member werden anschließend kopiert.

2.4.4 Verschieben von Membern

Das Verschieben von Membern lässt sich völlig analog zu den gerade vorgestellten Varianten des Kopierens von Membern durchführen. Es ist lediglich ein "m" (move) anstatt des "c" (copy) zu verwenden (siehe unten).

2.5 Member und ganze Datasets löschen

2.5.1 Member löschen

Das Data Set List Utility kann natürlich auch verwendet werden, um Member oder ganze Datasets (einschließlich mehrerer Member) zu löschen.

Wir behandeln zunächst den Fall „Member löschen“.

Um einen oder mehrere Member eines Datasets löschen zu können, zeigt man sich wieder eine Member-Liste an, die den oder die zu löschenden Member enthält.

Links neben jedem Member, der gelöscht werden soll, trägt man ein "`d`" (=delete) ein. Dieser Buchstabe kann, wie so oft im ISPF, groß oder kleingeschrieben werden (siehe unten).

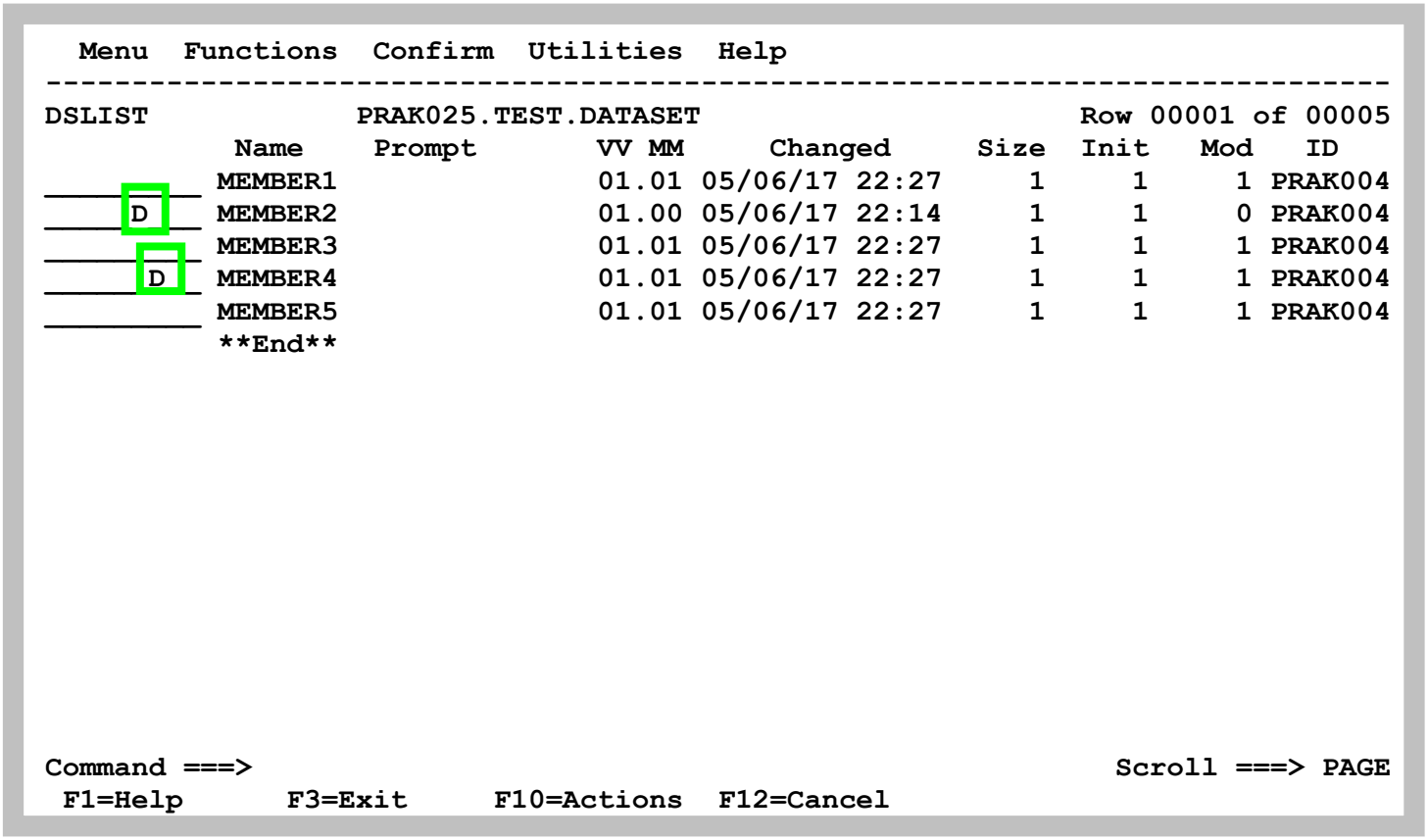
Die Eingabetaste beginnt den Löschvorgang. Doch muss man das endgültige Löschen eines jeden Members standardmäßig noch einmal per Eingabetaste bestätigen. Man kann diese Bestätigung ausstellen, indem man "`Set member delete confirmation off`" durch Eintrag eines "`/`" markiert. Letzteres erhöht die Gefahr von Datenverlust und sollte wohlüberlegt eingesetzt werden. Diese Bestätigung lässt sich mittels Eingabe von "`confirm`" auf der Kommandozeile wieder einschalten.

ISPF kennt noch eine alternative Möglichkeit, Member zu löschen: Die über das Library Utility. Diese wird hier nicht näher behandelt.

<u>Aufgabe:</u> ***Löschen Sie auf dem gerade beschriebenen Weg die beiden Member MEMBER3 und MEMBER5 des Datasets PRAK025.TEST.DATASET.***

Nun hat sich Ihr Dataset "PRAK025.TEST.DATASET" seit Ihrem ersten Screenshot stark verändert. Erzeugen Sie erneut eine Liste aller Member dieses Datasets und fertigen Sie einen zweiten Screenshot von dieser Liste an.

Sie sollten keine Daten Ihrer Arbeit löschen, damit Ihr Tutor sich Ihre Arbeit anschauen kann (Ausnahme: Ihre letzte Aufgabe dieses Tutorials "Löschen Sie den Dataset ...").


Menu Functions Confirm Utilities Help
DSLIST PRAK025.TEST.DATASET Row 00001 of 00005
Name Prompt VV MM Changed Size Init Mod ID
MEMBER1 01.01 05/06/17 22:27 1 1 1 PRAK004
D MEMBER2 01.00 05/06/17 22:14 1 1 0 PRAK004
MEMBER3 01.01 05/06/17 22:27 1 1 1 PRAK004
D MEMBER4 01.01 05/06/17 22:27 1 1 1 PRAK004
MEMBER5 01.01 05/06/17 22:27 1 1 1 PRAK004
End
Command ===>
Scroll ===> PAGE
F1=Help F3=Exit F10=Actions F12=Cancel

2.5.2 Ganze Datasets löschen

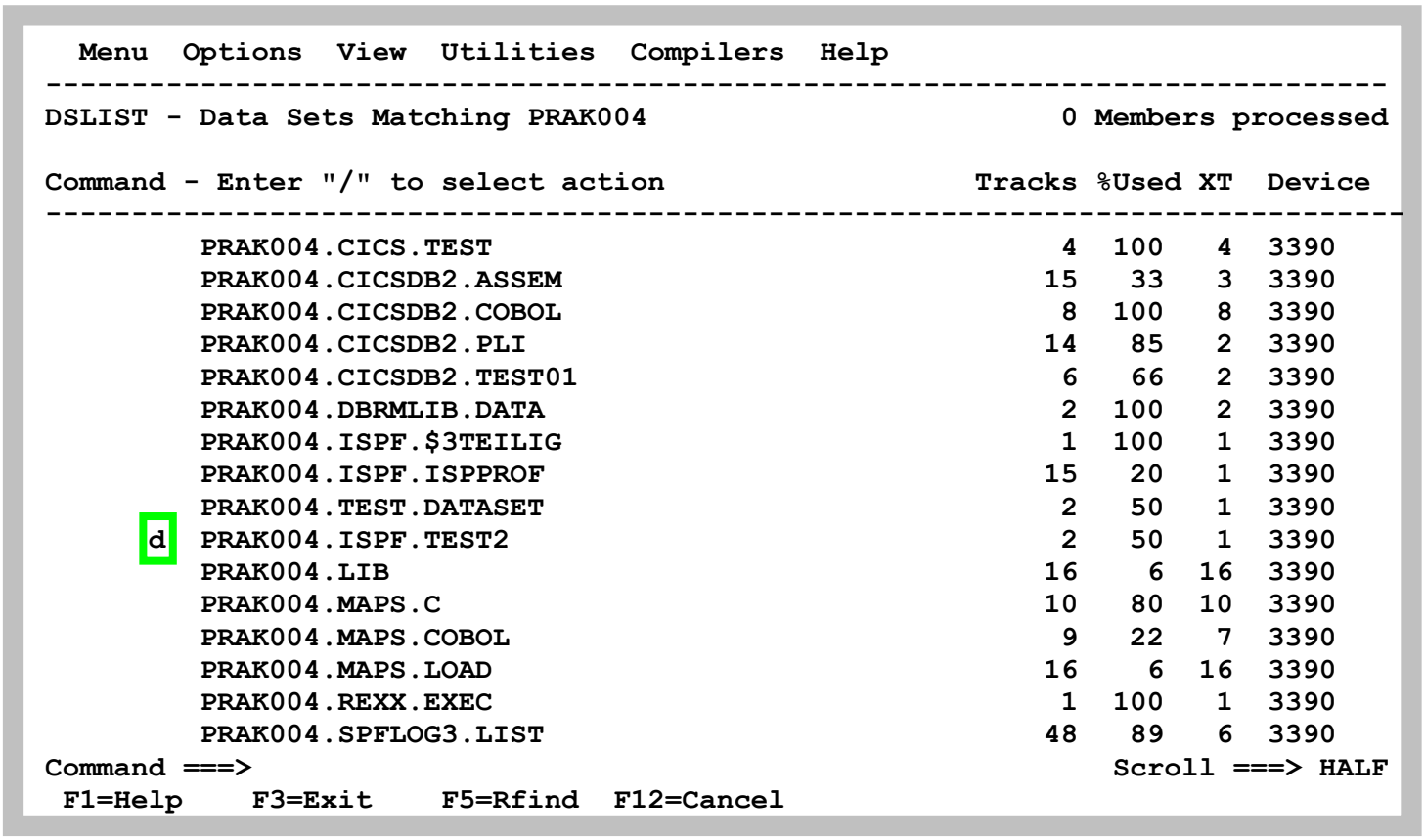
Diese Option sollte man sehr sorgfältig einsetzen, weil mit einem ganzen Dataset sämtliche Member einschließlich derem Inhalt gelöscht werden. Der per Allocate reservierte Plattenspeicherplatz wird wieder freigegeben.

Öffnen Sie ein Panel mit einer Dataset-Liste, die den zu löschenden Dataset enthält. Das Der Abschnitt "2.1 Eine Liste von Datasets anzeigen" behandelt dies.

Tragen Sie neben dem Dataset, den Sie löschen möchten, "`d`" (delete) ein (siehe unten).

Betätigen Sie die Eingabetaste.

Anschließend werden Sie aufgefordert, Ihre Löschanforderung durch erneute Betätigung der Eingabetaste zu bekräftigen:
"`Press ENTER key to confirm the delete request`". Tut man das, wird der Dataset einschließlich aller seiner Member gelöscht. Danach erscheint in der rechten oberen Panel-Ecke "`Data set deleted`".


Menu Options View Utilities Compilers Help
DSLIST - Data Sets Matching PRAK004 0 Members processed
Command - Enter "/" to select action Tracks %Used XT Device
PRAK004.CICS.TEST 4 100 4 3390
PRAK004.CICSDB2.ASSEM 15 33 3 3390
PRAK004.CICSDB2.COBOL 8 100 8 3390
PRAK004.CICSDB2.PLI 14 85 2 3390
PRAK004.CICSDB2.TEST01 6 66 2 3390
PRAK004.DBRMLIB.DATA 2 100 2 3390
PRAK004.ISPF.$3TEILIG 1 100 1 3390
PRAK004.ISPF.ISPPROF 15 20 1 3390
PRAK004.TEST.DATASET 2 50 1 3390
d PRAK004.ISPF.TEST2 2 50 1 3390
PRAK004.LIB 16 6 16 3390
PRAK004.MAPS.C 10 80 10 3390
PRAK004.MAPS.COBOL 9 22 7 3390
PRAK004.MAPS.LOAD 16 6 16 3390
PRAK004.REXX.EXEC 1 100 1 3390
PRAK004.SPFLOG3.LIST 48 89 6 3390
Command ===> Scroll ===> HALF
F1=Help F3=Exit F5=Rfind F12=Cancel

2.6 Die Eigenschaften von Datasets sich anzeigen lassen

Beim Anlegen (allocate) eines Datasets wurde dieser mit bestimmten Eigenschaften angelegt. Beispiele

Typ des anzulegenden Datasets, z.B. "PDS"
Record Frmat den Parameter, z.B. FB (Fixed Blocks)
Record length
Block size
usw.

Es existieren zwei unterschiedliche Alternativen, um diese Eigenschaften mittels der Dataset List Utility anzuzeigen:

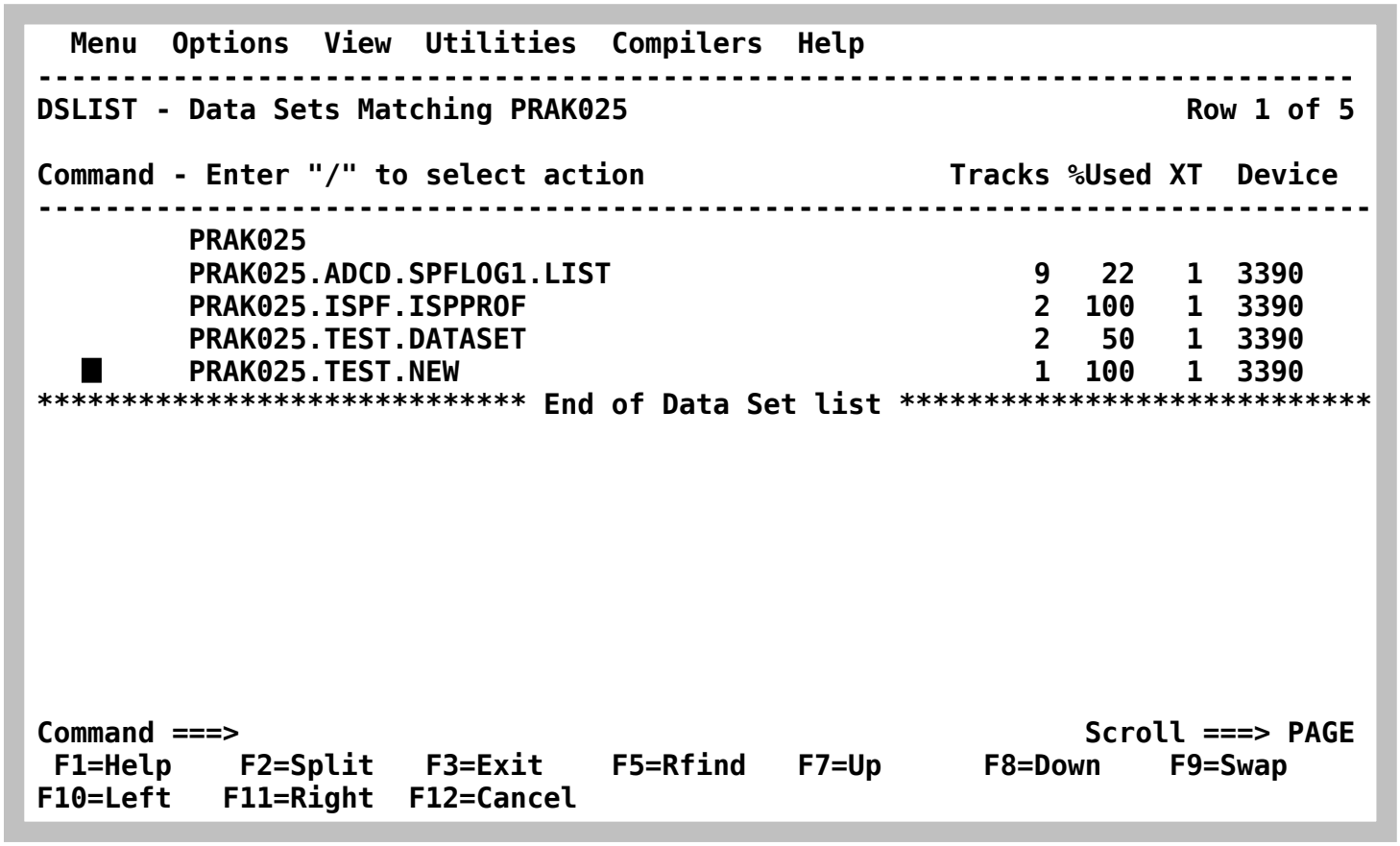
2.6.1 Alternative 1

Im Data Set List Utility Panel kann eine Dataset-Liste in 4 verschiedenen Varianten ausgegeben werden. Je nach gewünschter Variante kann man eine Zahl von 1 bis 4 in das Feld "`Initial View`" eingeben. Der Default Wert ist 1. Mögliche Varianten:

1. **Zum jeweiligen Dataset die Festplatte (Volume) anzeigen, auf der er sich befindet.**
2. **Zum Dataset den zugeordneten (allocate) Festplattenspeicherplatz (in Spuren / Tracks) anzeigen**
3. **Zum Dataset u.a. Rekord-Format, Rekord-Größe und Blockgröße anzeigen**
4. **Umfassende Angaben zum Dataset anzeigen, einschließlich der Punkte 1. bis 3.**

```
  Menu  RefList  RefMode  Utilities  Help
--------------------------------------------------------------------------------
                            Data Set List Utility
                                                                        More:     +
   blank Display data set list               P Print data set list
       V Display VTOC information           PV Print VTOC information

Enter one or both of the parameters below:
   Dsname Level . . . PRAK025
   Volume serial  . .

Data set list options
   Initial View . . . 2  1. Volume       Enter "/" to select option
                         2. Space        /  Confirm Data Set Delete
                         3. Attrib       /  Confirm Member Delete
                         4. Total        /  Include Additional Qualifiers
                                         /  Display Catalog Name

When the data set list is displayed, enter either:
  "/" on the data set list command field for the command prompt pop-up,
  an ISPF line command, the name of a TSO command, CLIST, or REXX exec, or
Option ===>
 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
F10=Actions  F12=Cancel
```

Geben wir beispielsweise eine „2“ ein. Eingabetaste.

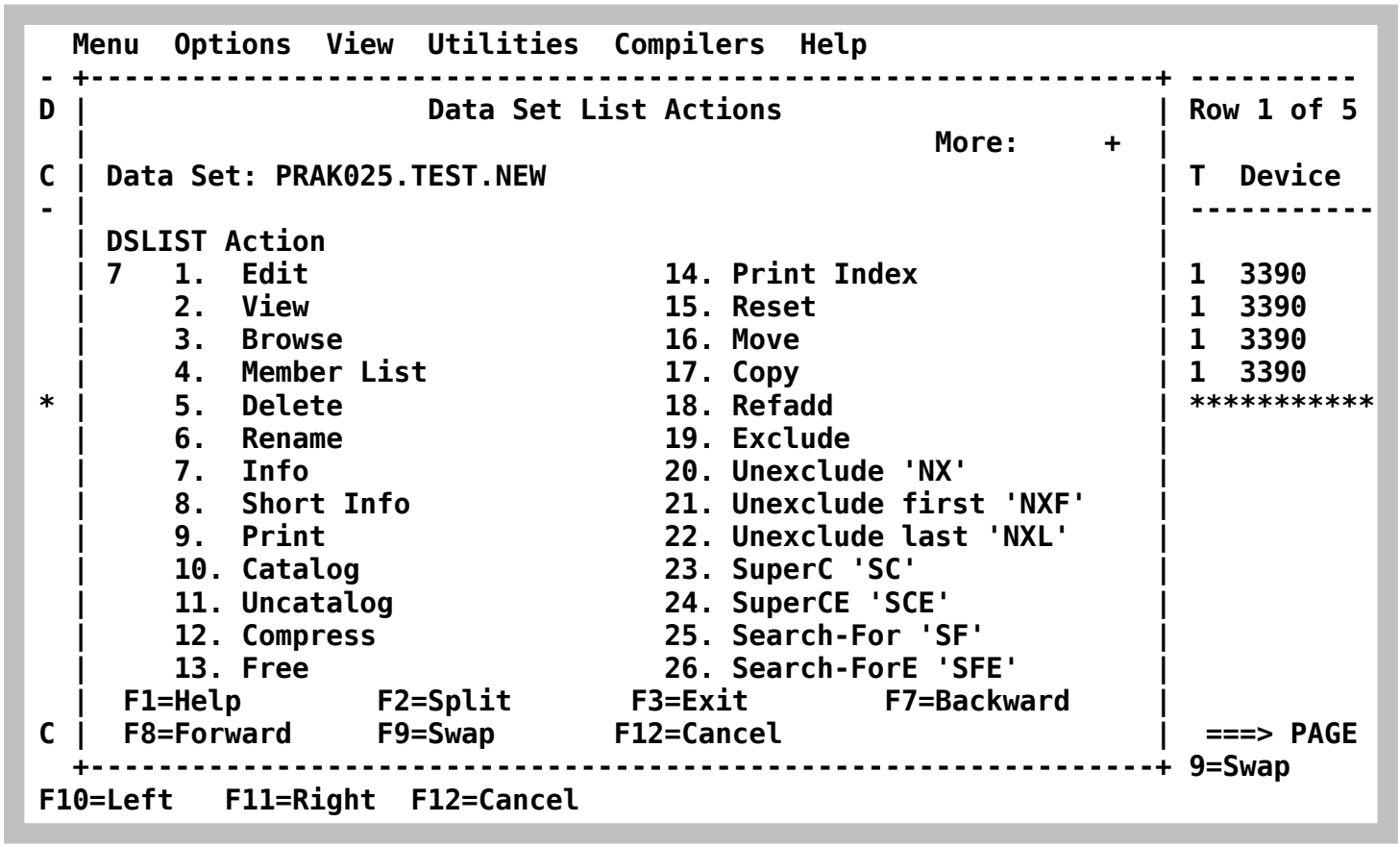


Es erscheint die Liste der gewünschten Datasets auf dem Bildschirm (siehe unten). Passen nicht alle Datasets auf das Panel, so kann man mit den Funktionstasten F7 und F8 nach oben sowie nach unten scrollen.

Eine auf dem Bildschirm angezeigte Dataset-Liste kann durch Druck der Tasten F10 oder F11 in eine andere Variante umgewandelt werden.

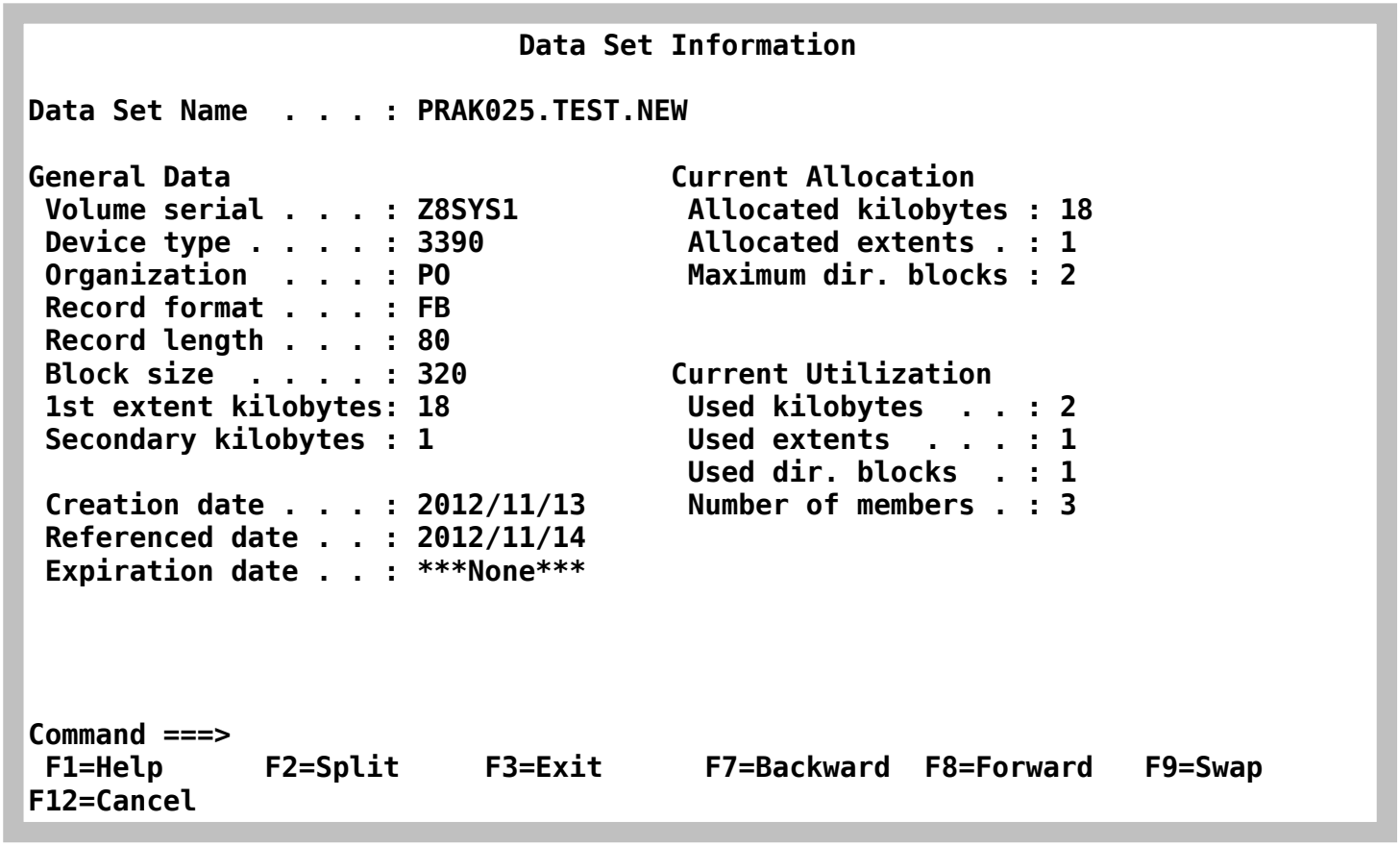
2.6.2 Alternative 2

Man plaziert man den Cursor links neben den entsprechenden Dataset und betätigt die Eingabetaste.



Im sich geöffneten "`Data Set List Actions`"-Panel wählt man "7" (="Info") aus. Die Eingabetaste erzeugt ein Panel mit den gewünschten Eigenschaften.

In einem konkreten Beispiel wird der Cursor links neben "`Edit`" gestellt, mit "7" der Menüpunkt "`Info`" ausgewählt und mit der Eingabetaste das "`Data Set Information`"-Panel geöffnet.



Eigenschaften des Datasets "PRAK004.TEST.NEW"

.....welches die gewünschten Informationen enthält.

2.7 Auf einen Dataset ein Compress anwenden

Arbeitet man mit Partitioned Datasets und werden sehr häufig neue Member anlegt und gelöscht, wird unter Umständen ein Komprimieren (Compress) der Datasets erforderlich.

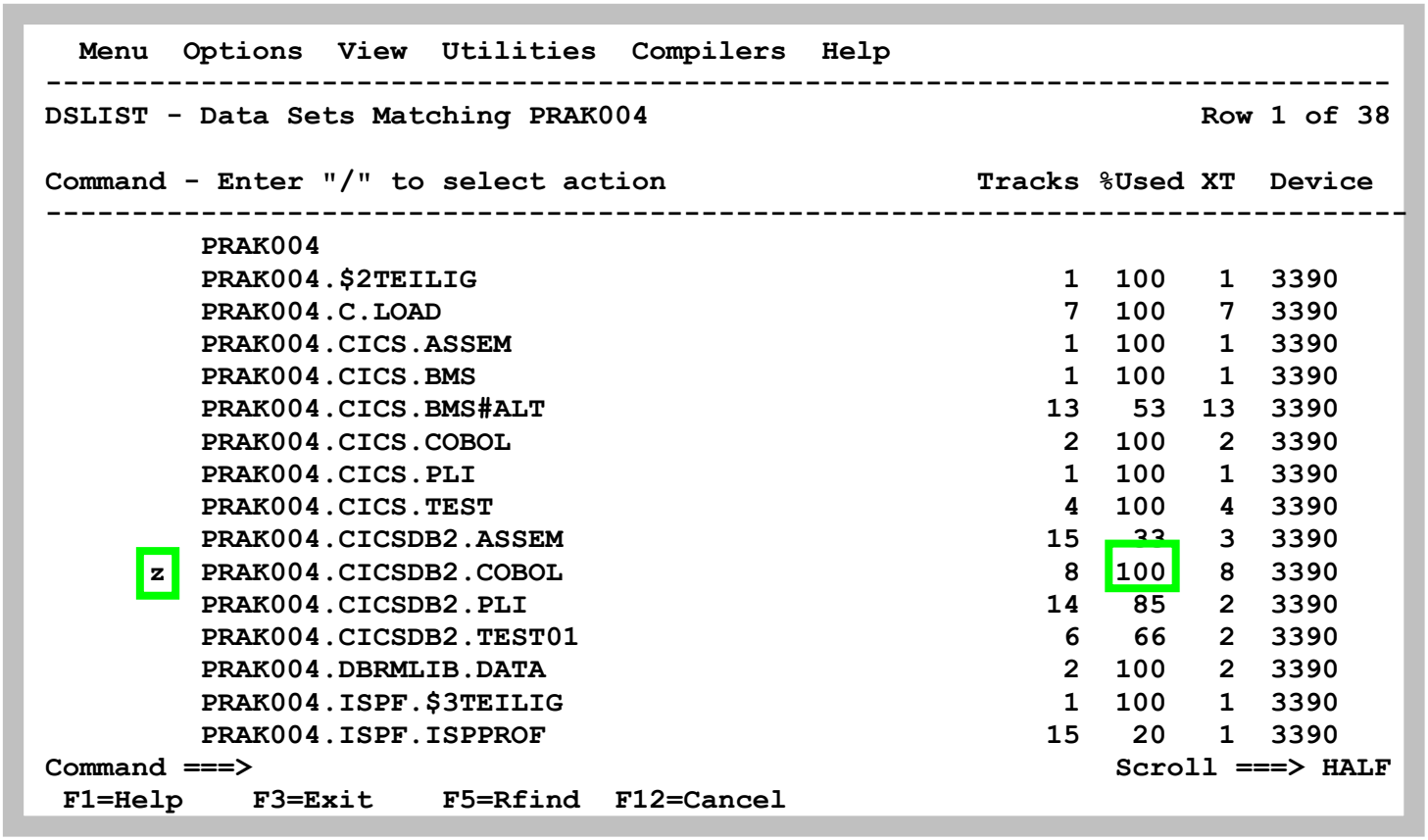
Der Grund ist, dass beim Löschen von Membern deren ehemaliger Festplatten-Speicherplatz nicht automatisch zur Wiederverwendung freigegeben wird. Diese Speicherplatzfreigabe muß man explizit durch einen Compress herbeiführen.

Man sollte die Notwendigkeit eines Compresses in den folgenden Fällen prüfen:

1. **Es läßt sich kein neuer Member anlegen**
2. **Ein Member läßt sich nicht mehr editieren**
3. **Die Ausführung eines JCL-Scriptes erzeugt die folgende Fehlermeldung:**
   09.27.54 JOB15798 $HASP165 PRAK025B ENDED AT N1 - ABENDED SE37 U0000 CN(INTERNAL) *

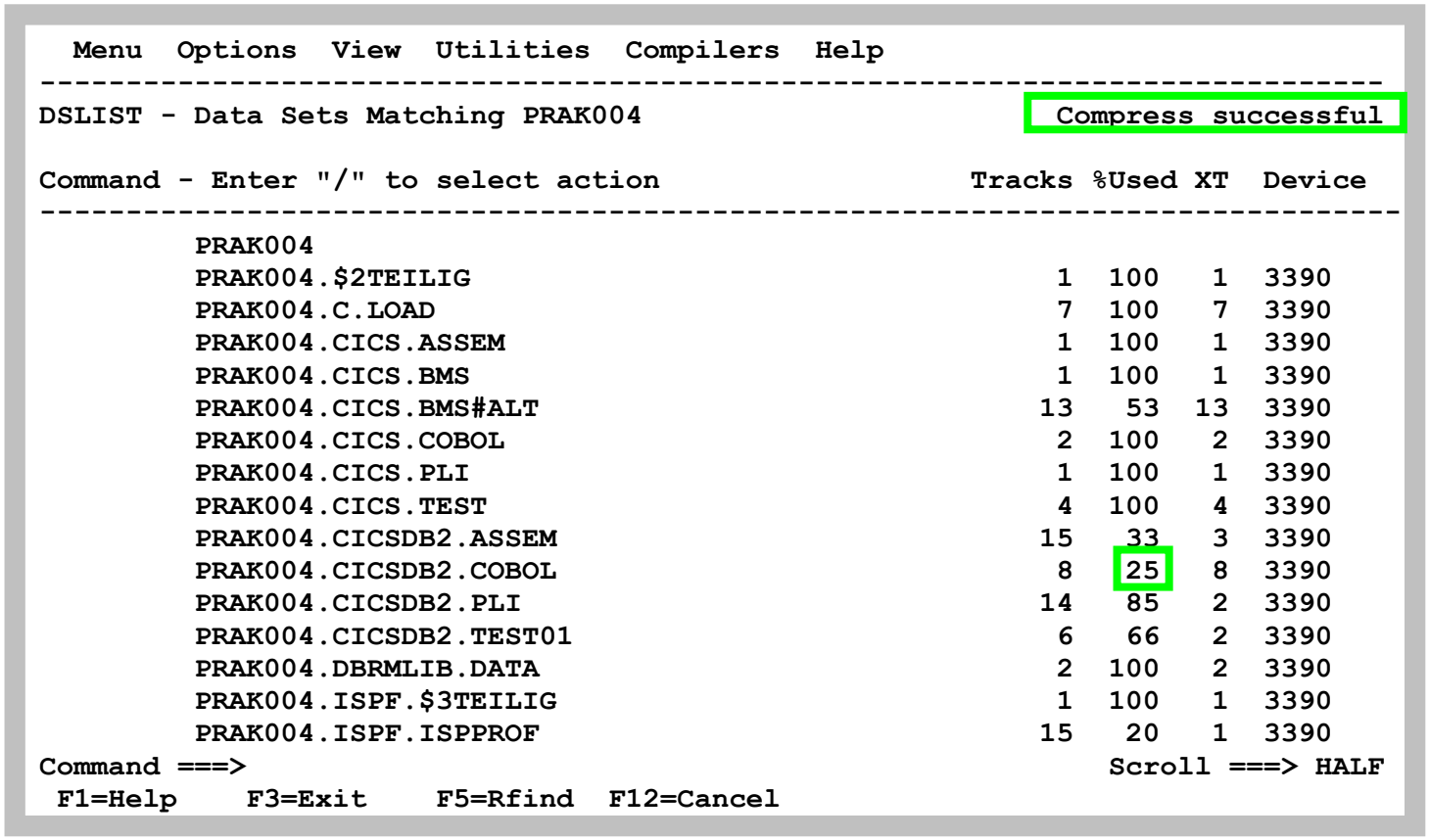
Um einen Compress auf einen Dataset anzuwenden, ist zuerst ein Panel mit einer Dataset-Liste, die diesen Dataset enthält, zu erstellen.

Einen Hinweis auf eine eventuell notwendige oder sinnvolle Komprimierung liefert auch eine Angabe von "%Used"="100" in der Dataset-Liste.



Dies gilt insbesondere dann, wenn die Anzahl der angelegten Tracks erheblich größer ist als 1. Dies ist beispielsweise bei dem in dem hier dargestellten Partitioned Dataset "PRAK004.CICSDB2.COBOL" der Fall.

Den zu komprimierenden Dataset wählt man durch Eingabe eines "Z" aus, welches man links neben den Dataset-Namen einträgt, aus. Eine anschließende Betätigung der Eingabetaste startet die Komprimierung.



Eine erfolgreiches Ende der Komprimierung wird durch "`Compress successful`" bestätigt.

Wie effektiv die Komprimierung war, kann man an der Veränderung des Wertes "`%Used`" ablesen. Vor der Komprimierung betrug dieser 100% (siehe oben). Die Komprimierung veränderte diesen Wert auf 25%. Die 100% bedeuteten, dass alle 8 für den Dataset reservierten Tracks benutzt waren. Die 25% bedeuten, dass 6 von 8 Tracks vom Dataset "`PRAK004.CICSDB2.COBOL`" frei wurden und deshalb für neue Member verfügbar sind.

3. Literatur

Ein sehr empfehlendes Lehrbuch ist:

**Michael Teuffel :TSO Time Sharing Option im Betriebssystem z/OS MVS.
Oldenbourg, ISBN-10: 3486255606, ISBN-13: 978-3486255607**

Den IBM TSO/E User's Guide finden Sie unter

http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/IKJ4C210/CCONTENTS

Das IBM TSO Referenz Handbuch können Sie herunterladen unter

http://www.informatik.uni-leipzig.de/cs/Literature/Textbooks/TSOreference.pdf

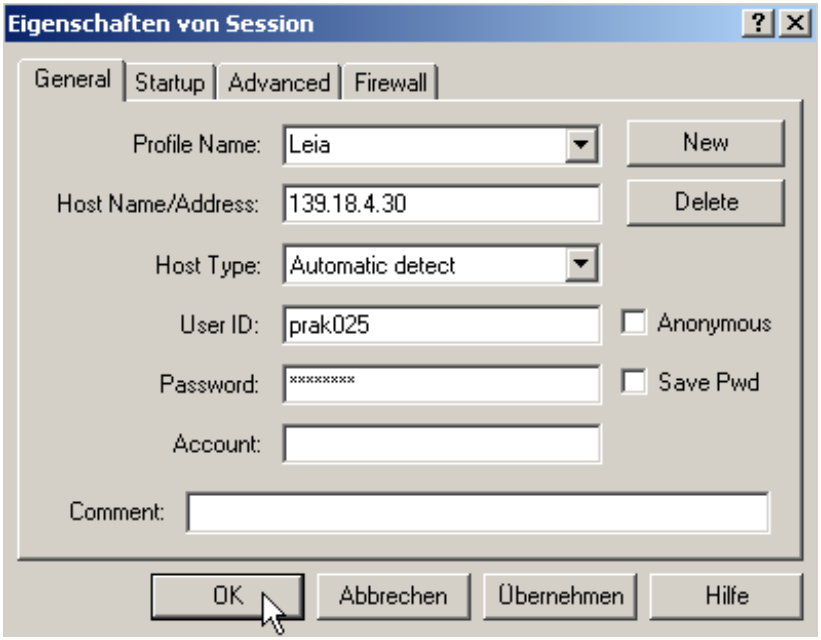
Die Western Illinois University ist eine der wenigen USA Universitäten, die über einen eigenen Mainframe Rechner für Ausbildungszwecke verfügen. Ein ausführliches und sehr empfehlenswertes TSO Tutorial finden Sie unter

http://mvs.wiu.edu/stumvs/TSO/tso.html

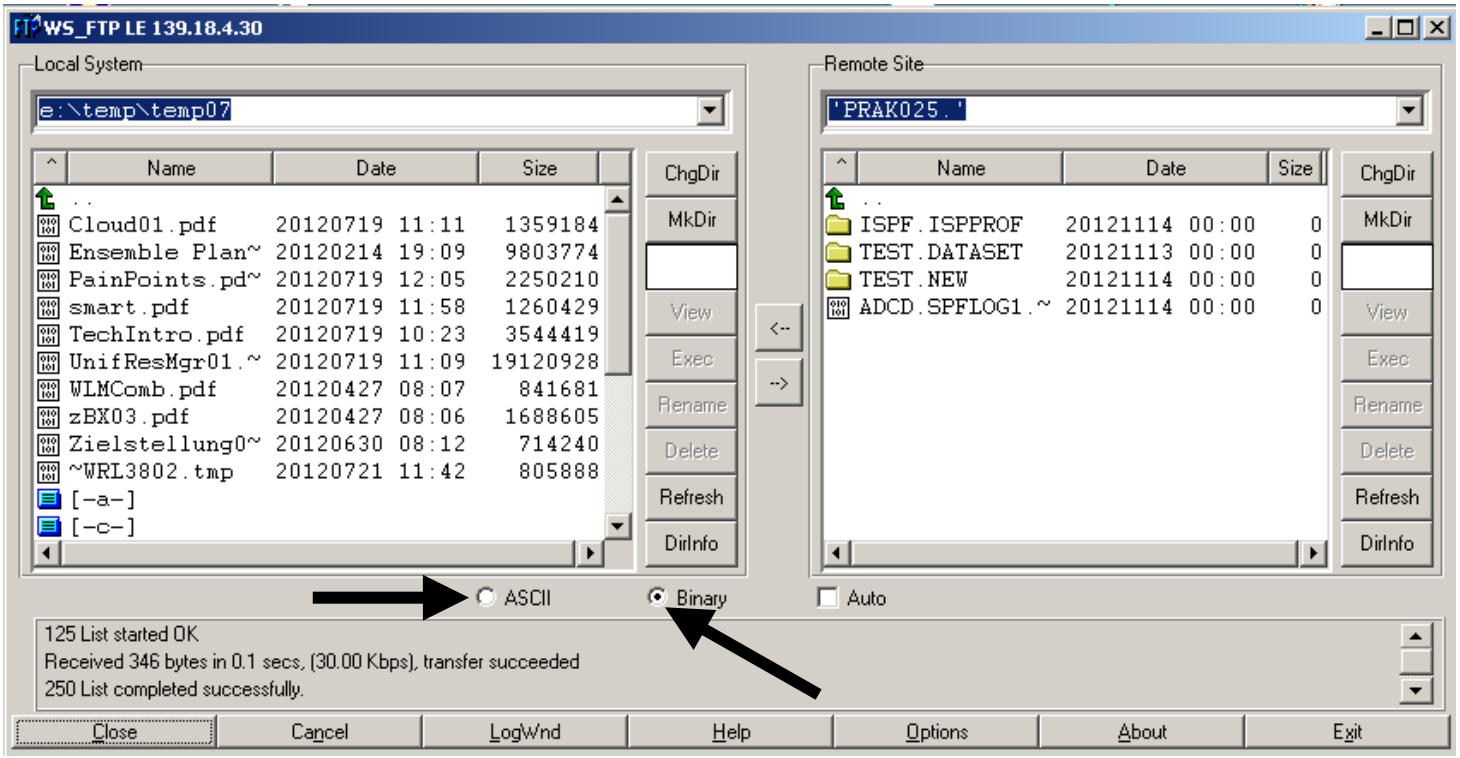
4. Download mit FTP

Vielfach ist es nützlich, Daten zwischen ihrem PC und Ihrem z/OS User Account auszutauschen. Hierzu gibt es mehrere Möglichkeiten. Sie können z.B. unter Windows mit Start → Ausführen ein FTP Programm im Zeilenmodus aufrufen.

Besonders einfach ist an dieser Stelle die Nutzung von `ws_ftple`. Hiermit können Sie problemlos Daten zwischen Ihrer TSO User ID und Ihrem PC transferieren. Sie können `ws_ftple` im Internet kostenlos herunterladen, z.B. unter http://www.wsftple.com/download.aspx.



Beim Aufruf von ws_ftple fragt Sie das Session Panel nach Ihrer Identifikation. Hie Sie loggen sich ein wie hier gezeigt.



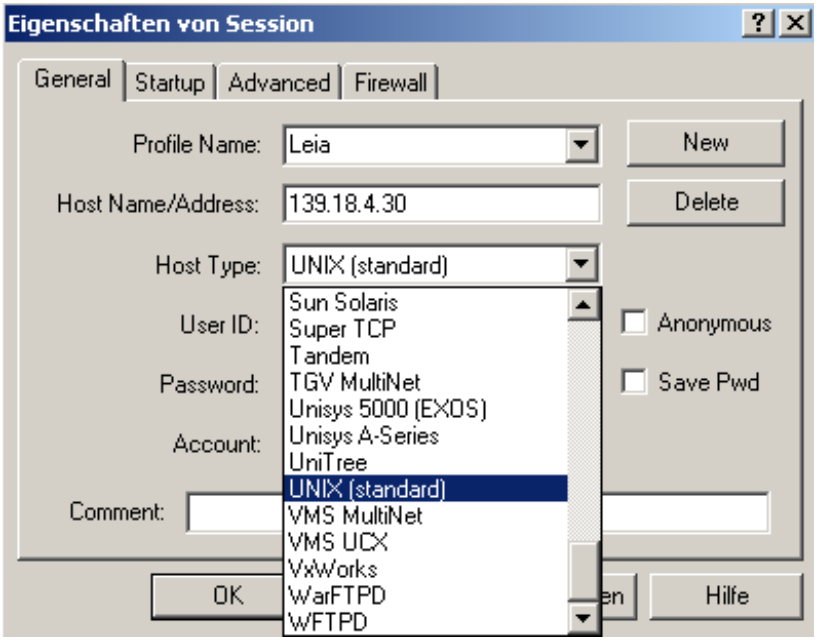
Im rechten Fenster erscheint die gleiche Information wie im DLIST Panel. Spezifisch werden die beiden angelegten Datasets PRAK025.TEST.DATASET und PRAK025.TEST.NEW sichtbar. Im linken Fenster erscheint irgend ein Verzeichnis auf Ihrem Windows PC. Sie können von dort aus zu einem Verzeichnis Ihrer Wahl navigieren,

Jetzt kann mit Drag and Drop von dem linken Fenster in das rechte Fenster kopiert werden und umgekehrt. Sie können z.B. auf diese Art ein Backup Ihrer z/OS Datasets auf Ihrem Arbeitsplatzrechner erstellen.

Jetzt ist eine Übertragung von und nach TSO möglich.

Vorsicht: z/OS stellt Daten im EBCDIC Format dar. Für die Übertragung von ASCII Daten (z.B. Textfiles) den ASCII Radio Button aktivieren. Dann funktioniert die Konvertierung automatisch. Für die Übertragung von Binärdaten (Beispiel pdf Files) den Binary Radio Button aktivieren.

Bei manchen File Typen kann die Entscheidung ASCII – Binary schwierig werden. Hier hilft im Zweifelsfall probieren.



Wenn Sie Daten in das WebSphere Web Application Server Subsystem übertragen, gelten andere Regeln. WebSphere läuft unter z/OS Unix System Services, und hier verhält sich z/OS wie ein Unix System. Spezifisch benutzt WebSphere (und andere Unix System Services Anwendungen) ein Unix kompatibles Hierarchisches Unix File System an Stelle von normalen z/OS Data Sets

Deshalb im Session Panel an Stelle von „Automatic Detect“ die Einstellung „Unix Standard“ wählen.

Anhang A - Common TSO Commands

ALLOCATE (ALLOC) Allocate a new data set or link an existing data set to a particular ddname (or file name) to be read by a program.
Example: ALLOC DS(NEW.DATA) NEW UNIT(PERM) SPACE(10) BLOCK(6320) Ex2. ALLOCATE FILE(FT08F001) DATASET('CSLSU.DATA')

ALLOW Grant ACF2 (security) authorization for another logonid to use your department and project numbers.
Example: ALLOW CSLSU

BLKSIZE Determine optimal blocksizes for disk data sets. In the example below, 80 represents the data set's record length.
Example: BLKSIZE 80

CANCEL Cancel one of your background jobs. Example: CANCEL CSLSU1

COMPARE Compare the contents of two data sets. List any differences.
Example: COMPARE 'CSLSU.VERSIONI"CSLSU.VERSION2'

COMPRESS (C) Compress a partitioned data set.
Example: C 'CSLSU.PDS'

CONC Concatenate your CLIST library with SYS2.CMDPROC (the system CLIST library) or a user specified CLIST library.
Example: CONC

COPY Copy a sequential data set, partitioned data set (PDS), or PDS member to a new or existing data set or PDS member.
Example: COPY OLD.DATA NEW.DATA

DSAT Display data set attributes.Example: DSAT 'CSLSU' ; Example: DSAT CLIST

DUSER (DU) Display free space on the USEROx volumes.
Example: DUSER

DVOL Display free space on a disk volume. Example: DVOL USER08

EDIT (E) Edit a data set using the TSO Editor. (This is normally used only when SPF is not available on a terminal.)
Example: EDIT (BEGIN) CLIST

FREE Free an allocated file. Example: FREE DATASET('CSLSU.DATA')

FREEALL Free all dynamically allocated files except for excluded (FA) ddname list.
Example: FREEALL

HELP (H) Obtain online help information for a TSO command. Example: HELP DSAT

IOF Facility for viewing and printing "held" job output. Example: IOF CSLSU1

LIST (L) List a data set (or PDS member) at your terminal. Example: LIST 'CSLSU.PGM.FORT'

MANUAL Display list of printable documentation and, optionally, print a manual. (From SPF, use option L.DOC) Example: MANUAL

MEMBERS (MEM) Display names of members of a partitioned data set. Example: MEMBERS PDS.DATA

MMDEL Delete a member(s) from a PDS. Example: MMDEL CLIST (MEMI,MEM2)

OQ Display job status information. Example: OQ CSLSU1 JOBNAME

OSDEL Delete a disk data set. Example: OSDEL PGM.DATA

PERMIT Grant ACF2 (security) authorization for another logonid to access your data set(s). Example: PERMIT CSLSU

PLOTDD Allocate files for Benson plotting. Example: PLOTDD

PRINTDA Print or punch a disk data set. Example: PRINTDA PGM.DATA

PRINTOUT Move held job output to a disk data set and (optionally) print. Omit NOPRINT to print output.
(PO) Omit KEEP to have job output purged from held output queue after printing. Example: PRINTOUT CSLSUI NOPRINT KEEP

REL Release some of the unused tracks in a data set. In the following example, 10 tracks are released.
Example: REL PDS.DATA 10

RLSE Release all unused space in a disk data set. Example: RLSE 'CSLSU.PGM.FORT'

RENAME Rename a disk data set.
(REN) Example: RENAME OLD.NAME NEW.NAME

SEARCH Search a PDS for a specified string. Example: SEARCH 'CSLSU.PGM.FORT"VECTOR'

SEND Send a message to another logonid. Example: SEND 'MESSAGE' USER(CSLSU)
(SE)

SPF Invoke SPF. Example: SPF

TSOSORT Sort a data set from TSO. Example: TSOSORT FROM.DATA TO.DATA'FIELDS=(1,4,CH,A)'

TSOUSER Specify TSO user identification. Statement usually placed in BEGIN member of user's CLIST to denote print destination.
Example: TSOUSER 'CSLSU - STUBBS'

USA Grant ACF2 (security) authorization to SNCC User Services to use your department and project numbers.
(Not required for classwork logonids.) Example: USA

USERS Display logonids of users logged on to TSO. Example: USERS

USP Grant ACF2 (security) authorization to SNCC User Services to access your data sets.
(Not required for classwork logonids.) Example: USP

Erstellen, Kompilieren und Ausführen eines COBOL-Programms

In dieser Aufgabe wiederholen wir das Anlegen von Datasets (Allocate) sowie das Füllen mit Daten unter Verwendung des ISPF-Editors, Sie lernen kennen, wie man ein COBOL-Programm unter z/OS schreibt, kompiliert und ausführt.

Hinweis: Dieses Tutorial wurde unter Verwendung der Benutzer-ID "PRAK085" erstellt. In allen Dateinamen müssen Sie "PRAK085" durch ihre eigene Benutzer-ID ersetzen.

Aufgabe: ***Arbeiten Sie nachfolgendes Tutorial durch.***

1. Einführung

1.1 Entwicklungsumgebung

In diesem Tutorial werden Sie ihr erstes Hello World Programm auf dem Mainframe entwickeln und ausführen. Wir benutzen hierfür die Programmiersprache Cobol (COmmon Business Oriented Language).

Auf einem Einzelplatzrechner würden Sie vermutlich die gleiche Umgebung sowohl für die Entwicklung als auch für die Ausführung eines neuen Programms benutzen, also z. B. unter Benutzung einer bash oder korn Shell unter Linux, oder der DOS Eingabeaufforderung unter Windows. Auf einem Mainframe kann man ebenfalls Entwicklung und Ausführung beides unter TSO plus ISPF durchführen.

Ein Mainframe wird aber hauptsächlich als Server für die Ausführung von interaktiven- oder Stapelverarbeitungs-Prozessen benutzt. Deshalb werden die Umgebungen für Entwicklung und Ausführung fast immer getrennt. Die Ausführungsumgebungen für interaktive Anwendungen sind z.B. CICS, DB2, IMS Stored Procedures oder WebSphere Java Servlet und Java Enterprise Bean Prozesse. Die Ausführungsumgebung für Stapelverarbeitungsprozesse ist fast immer das Job Entry Subsystem JES.

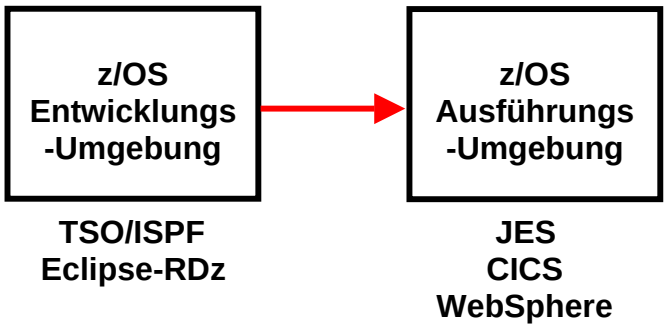


Abb. 1.1 : Entwicklungs- und Ausführungsumgebung

Die Entwicklung und anschließende Ausführung eines neuen Programms besteht damit aus 3 Schritten:

Schritt 1: Sie benutzen die Entwicklungsumgebung um Ihr neues Programm

- **zu editieren,**
- **zu übersetzen (compile),**
- **zu debuggen, und**
- **in einer Library mit anderen von Ihnen entwickelten Programmen abzuspeichern, sowie**
- **um die von Ihrem neuen Anwendungsprogramm benutzten Daten zu erstellen (oder importieren und ebenfalls in einem Data Set abzuspeichern.**

Schritt 2: Danach importieren Sie ihr neues Programm in eine Ausführungsumgebung. Ausführungsumgebungen unter z/OS sind z.B. JES (Job Entry Subsystem), CICS, DB2 Stored Procedures, IMS-DC, IMS Stored Procedures oder WebSphere, oder in seltenen Fällen TSO.

Als letzten Schritt 3 sind Sie dann in der Lage, ihr neues Programm auszuführen, indem Sie z.B. eine neue CICS Transaktion ausführen, ein (JCL oder REXX) Script für einen Stapelverarbeitungsprozess aufrufen, oder ein Programm mittels TSO und ISPF aufrufen.

Als Entwicklungsumgebung für neue Mainframe Anwendung wird heute häufig Eclipse mit der „Rational Developer für System z“ (RDz) Erweiterung eingesetzt. Wir werden dies in einem späteren Tutorial einsetzen. Aus didaktischen Gründen verwenden wir in diesem Tutorial den klassischen Ansatz, bei dem als Entwicklungsumgebung TSO und ISPF benutzt werden.

1.2 Übersetzen (Kompilieren) eines Programms

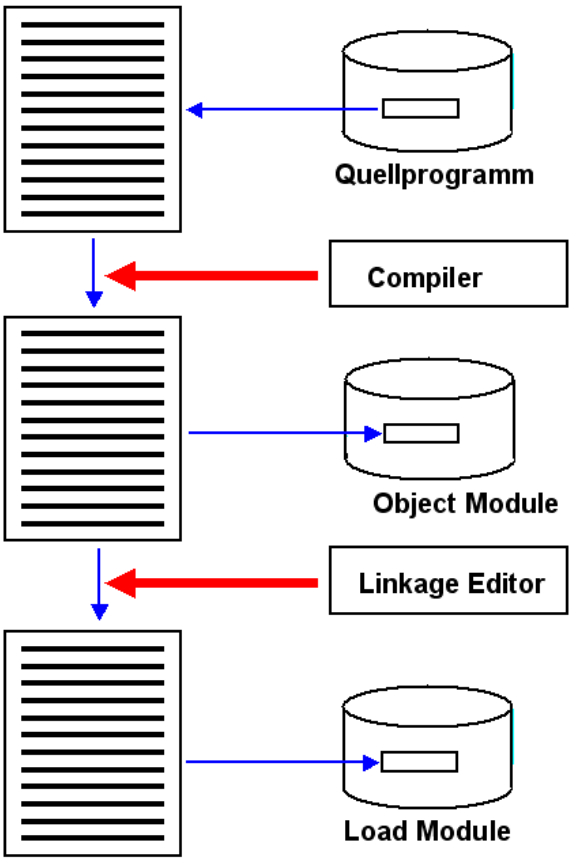


Abb. 1.2 : Übersetzungsvorgang

Ein neues Programm wird als Quellprogramm (Quelltext) mittels eines Editors in einer Programmiersprache wie z.B. Cobol, PLI, C/C++ oder Java erstellt und in einem PDSe Data Set abgespeichert. Ein PDSe Data Set stellt in vielen Fällen eine Programm-Library dar, wobei die Member einzelne Programme speichern.

Ein Compiler übersetzt das Quellprogramm in ein „Object Module“, welches ebenfalls als Member in einem PDSe Data Set gespeichert wird. Das Object Module besteht aus lauter ausführbaren System z Maschinenbefehlen. Im Prinzip könnte das Object Module in den Hauptspeicher geladen werden und dann aufgerufen und ausgeführt werden.

In der Praxis findet das jedoch so gut wie nie statt. Der Grund ist, dass ein größeres Anwendungsprogramm in der Regel aus mehreren Programm-Teilen besteht, die einzeln übersetzt, und als getrennte Object Module in den Membern eines PDSe Data Sets gespeichert werden. Die einzelnen Member werden von einem „Linkage Editor“ zu einem einzigen ausführbaren Programm, dem Load Module zusammengefügt. Eine Aufgabe des Linkage Editors ist es, die (symbolischen) Referenzen der Object Module untereinander aufzulösen und durch Hauptspeicheradressen zu ersetzen.

Weiterhin werden vom Linkage Editor vorgefertigte System Routinen eingebunden. Ein Beispiel hierfür sind die „Access Methoden“ bei der Verwendung eines VSAM Data Sets. Ein anderes Beispiel sind „Language Environment“ (LE) Routinen. Wenn Sie z.B. in einem Cobol, PLI oder C/C++ eine SQRT (Quadratwurzel) Routine aufrufen, wird diese vom Compiler nicht mit übersetzt. Statt dessen bindet der Linkage Editor eine bereits vorhandene Quadratwurzel Routine (die aus lauter Maschinenbefehlen besteht), in das Load Module ein.

Beim Aufruf des neuen Programms wird das Load Module in den Hauptspeicher geladen und dann ausgeführt.

Die Definitionen für all dies werden in einem Script in der JCL (Job Control Language) Sprache zusammengefasst und abgespeichert. Eine make File unter Linux hat eine vergleichbare Funktion.

1.3 Ausführen eines Programms.

Wenn ein Quellprogramms unter TSO/ISPF entwickelt wird, erfolgt die Übersetzung typischerweise mittels eines JES (Job Entry Subsystem) Stapelberarbeitungsprozesses. Der Auftrag hierfür wird mittels eines JCL Scripts erstellt; das Linken mehrer Object Module zu einem Load Modul durch den Linkage Editor erfordert ein weiteres JCL Script. Die Ausführung eines Load Modules ist ein davon getrennter Vorgang. Falls das Load Module in einer CICS, IMS oder WebSphere Ausführungsumgebung gestartet werden soll, ist es erforderlich, das Load Module vorher in dieser Ausführungsumgebung zu installieren.

Die Übersetzung des Quellprogramms und das Linken unter JES ist besonders einfach. Hierzu wird das entsprechende JCL Script mit Hilfe des ISPF Kommandos „submit“ (abgekürzt „sub“) an JES übergeben. JES interpretiert die einzelnen Anweisungen des JCL Scriptes und führt es aus.

Das vorliegende Tutorial würde eigentlich zwei JCL Scripts benötigen, je eines für die Compile und Linkage Edit Schritte. Unser primitives Hello World Programm ermöglicht eine Vereinfachung. Wir erstellen nur ein einziges JCL Script, welches aus der System Library ein weiteres JCL Script IGYWCL aufruft. IGYWCL bewirkt die Schritte

1. **Übersetzen eines Cobol Programms,**
2. **Link Edit**

in einem Schritt. Siehe hierzu Abschnitt 4.3 .

2. Einrichten der Entwicklungsumgebung

In dem hier vorliegenden Tutorial verwenden wir als Entwicklungsumgebung TSO und ISPF. Die Ausführungsumgebung für die Übersetzung ist JES. Spezifisch verwenden wir für die Erstellung des Cobol Quellprogramms den ISPF Editor.

2.1 Anlegen der erforderlichen Data Sets.

Wir benötigen in diesem Tutorial 3 Data Sets:

- **Einen Data Set *PRAK085.TEST.COB* für die Aufnahmen des Quelltextes unseres Cobol Programms**
- **Einen Data Set *PRAK085.TEST.LOAD* für die Aufnahmen Object Codes unseres übersetzten Cobol Programms**
- **Einen Data Set *PRAK085.TEST.CNTL* für die Aufnahmen eines in der JCL (Job Control Language) Sprache geschriebenen Scriptes für die Ablaufsteuerung. Diesen Data Set haben Sie möglicherweise in dem Tutorial 1a bereits angelegt.**

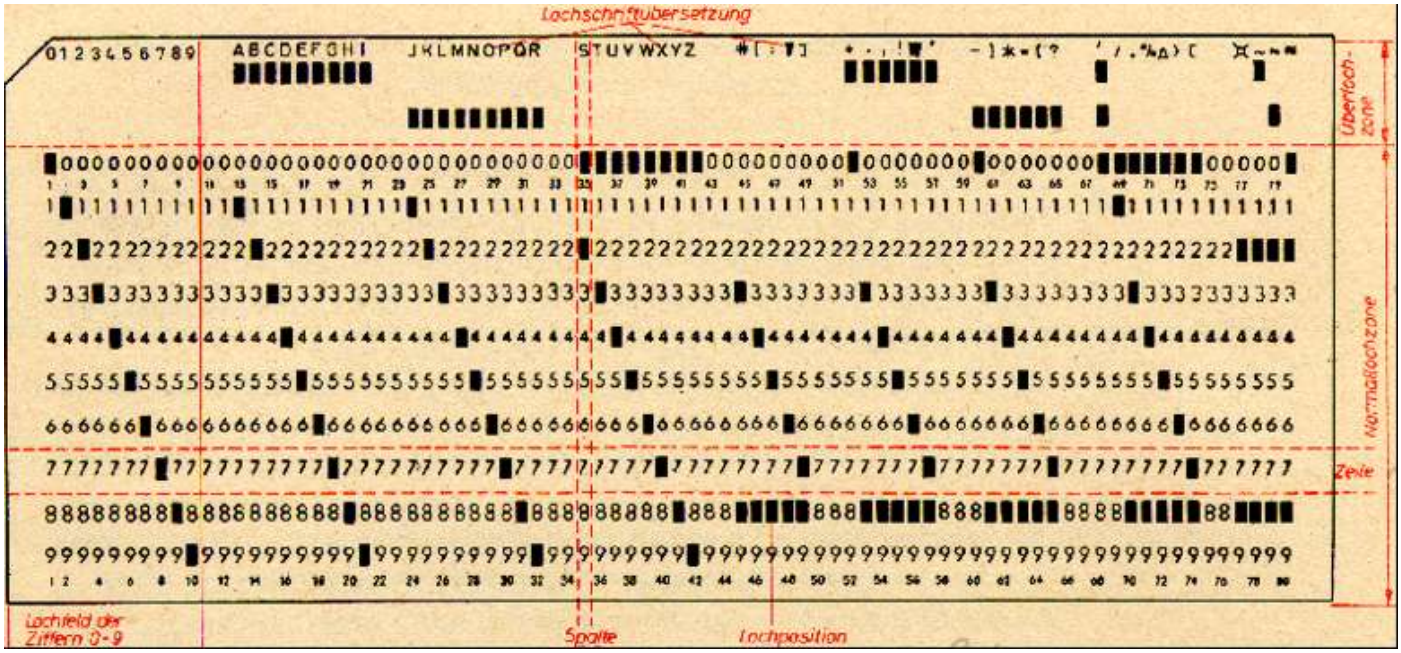
Aufgabe: *Legen Sie zwei Datasets PRAK085.TEST.COB und PRAK085.TEST.CNTL ("PRAK085" durch Ihre Benutzer-ID ersetzen) an. Verwenden Sie die gleichen Parameter wie im Tutorial zur Aufgabe 1.*

Legen Sie den Dataset PRAK085.TEST.LOAD ("PRAK085" durch Ihre Benutzer-ID ersetzen) an, welcher die ausführbare Datei nach dem Kompilieren aufnehmen soll. Verwenden Sie wieder die Ihnen bekannten Parameter, mit einem Unterschied: Statt im Dateiformat (Record format) "Fixed Block" soll dieser Dataset im Dateiformat "Undefined" erstellt werden. Dazu ist an der dafür vorgesehenen Stelle ein "U" als Parameter anzugeben.

Verifizieren Sie, dass diese drei Datasets existieren.

2.2 Spaltenabhängigkeit

Bei der Benutzung des ISPF Editors tritt ein als „Spaltenabhängigkeit bezeichnetes Phänomen auf. Ohne Wissen der historischen Entwicklung ist es unmöglich, dies zu verstehen.



1 Abb. 2.1 IBM Lochkarte

Die Entwicklung der maschinellen Informationsverarbeitung begann mit der Erfindung der Lochkarte durch Herrn Herrmann Hollerith, dem Gründer der Firma IBM. Abb. 2.1 zeigt das Format einer Lochkarte, etwa 19 x 8 cm groß, wie sie 1928 von IBM eingeführt und standardisiert wurde, und bis etwa 1970 in Gebrauch war. Der Vorläufer mit einem etwas anderen Format wurde von Herrmann Hollerith 1890 bei der Volkszählung in den USA und wenig später auch in Deutschland eingesetzt.

Die „IBM Lochkarte“ besteht aus einem Stück biegsamer Karton, in welches Löcher gestanzt werden. Sie hat insgesamt 80 Spalten. In jede Spalte konnte an einer oder an zwei von 12 Stellen jeweils ein Loch gestanzt werden. Die Position konnte in einem Lochkartenleser abgefühlt werden. Damit war es möglich, mittels der sog. BCD Kodierung (Binary Coded Decimal) in jeder Lochkarte 80 alphanumerische Zeichen zu speichern.

Zahlreiche Abbildungen von Lochkarten sind zu finden unter http://www.google.de/search?q=punched+card&hl=de&lr=&as_qdr=all&prmd=imvns&tbm=isch&tbo=u&source=univ&sa=X&ei=wiqlUMu4FobV4QSk0oHICA&ved=0CCoQsAQ&biw=1516&bih=963.

Bis zur Mitte des 20. Jahrhunderts war die Lochkarte das dominierende Medium zur Speicherung von relativ großen digitalen Datenmengen. Ein Lochkartenstapel mit 2000 Karten, etwa 35 cm hoch, hat eine Speicherkapazität von etwa 160 000 Bytes.

Cobol, Fortran und JCL Programme speicherten jeweils ein Statement pro Lochkarte. Hat das Statement eine Länge von < 80 Bytes, bleibt der Rest des Platzes auf der Lochkarte ungenutzt. Ist die Länge > 80 Bytes, benutzt man zwei oder mehr Lochkarten für ein Statement. Die Kennzeichnung von 2 oder mehr Karten pro Statement geschieht durch eine Lochung in einer spezifischen Spalte.

Cobol entstand 1960. Um das Parsing eines Statements zu erleichtern, bestand die Konvention, dass die Label in Spalte 7 und die Befehle in Spalte 12 beginnen müssen. Diese Spaltenabhängigkeit wird auch von der heutigen Version des ISPF Editors und dem dazugehörigen Cobol Compiler vorgeschrieben. Bei anderen Sprachen, z.B. JCL, bestehen ähnliche Einschränkungen. In anderen Cobol Entwicklungsumgebungen und deren Compilern bestehen diese Einschränkungen nicht.

Der heute von Mainframes verwendete Extended Binary Coded Decimal Interchange Code (EBCDIC, ausgesprochen ebsidik) ist eine 1:1 Umsetzung der BCD Lochkartenstanzungen in einen 8 Bit Code.

3. Erstellen des Quelltextes des COBOL-Programms

Unser Hello World Tutorial ist für die vier Programmiersprachen C/C++, Cobol, PL/1 und Assembler verfügbar. In dem hier vorliegenden Tutorial wird mit Cobol Quellcode gearbeitet.

Cobol ist die auf Mainframes am weitesten verbreitete Programmiersprache. Auf anderen Plattformen wird Cobol ebenfalls, aber nicht so häufig wie C/C++ oder Java eingesetzt. Eine signifikante Anzahl neuer Anwendungen wird jedes Jahr in Cobol entwickelt.

Es existieren zahlreiche Cobol Compiler von unterschiedlichen Herstellern. Der wichtigste ist der „z/OS Enterprise Cobol Compiler“, den wir hier benutzen.

Zwei ebenfalls weit verbreitete Cobol Compiler sind:

- **Microfocus Cobol Compiler: http://www.microfocus.com/mcro/cobol/index.aspx**
- **Fujitsu Cobol Compiler: http://www.netcobol.com/products/COBOL-Compilers-and-Development-Tools/overview**

Zwei kostenlose Open Source Cobol Compiler sind:

- **Tiny COBOL: http://tiny-Cobol.sourceforge.net/**
- **OpenCOBOL: http://www.opencobol.org/**

Ein COBOL-Programm ist in Teile (DIVISION), Kapitel (SECTION) und Abschnitte (PARAGRAPH) gegliedert.. Die vier DIVISIONs sind in ihrer festgelegten Reihenfolge:

- **Identification Division mit dem Programmnamen und Kommentaren,**
- **Environment Division, wo Schnittstellen zum Betriebs- und Dateisystem definiert werden,**
- **Data Division mit der Definition der Programmvariablen und Datenstrukturen und**
- **Procedure Division mit dem eigentlichen Code prozeduralen Anweisungen.**

Die ENVIRONMENT und DATA DIVISIONs können unter Umständen ganz entfallen.

Hieraus folgt eine strikte Trennung von Datendeklaration und prozeduralen Anweisungen, durch die sich COBOL auszeichnet. In der Procedure Division kann man nur Variablen benutzen, die vorher in der Data Division deklariert worden sind.

Cobol ist case insensitive. Historisch haben viele Cobol Programme nur Großbuchstaben verwendet. Cobol gilt als eine sehr produktive Sprache für die Entwicklung und anschließende Wartung von Programmen.

Wir benötigen insgesamt 3 Datasets. Der erste Dataset soll den Quellkode des COBOL-Programms aufnehmen, der zweite die ausführbare Datei. Einen dritten Dataset "PRAK085.TEST.CNTL" nimmt das JCL-Script auf und wird zum Kompilieren benutzt.

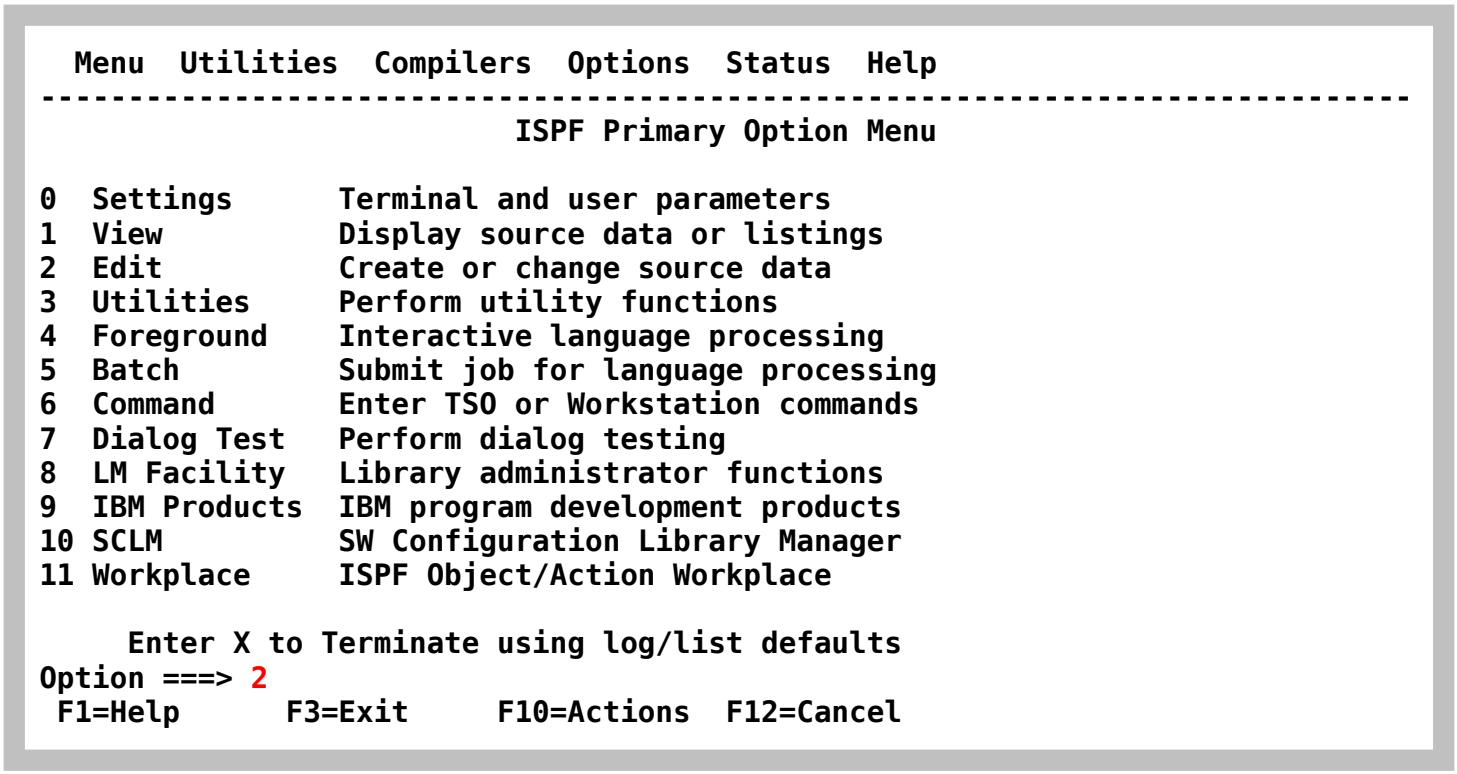


Abb. 3.1: "ISPF Primary Option Bildschirm"

Wir haben bisher die Utilities-Funktion benutzt, um unsere Entwicklungsumgebung anzulegen. Hierzu haben wir drei Partitioned Datasets angelegt. Jetzt wollen wir diesen Speicherplatz benutzen, um ein Programm zu schreiben, zu übersetzen und auszuführen. Dies geschieht mit Hilfe der "Edit"-Funktion. Wie in Abb. 3.1 dargestellt, geben wir eine "2" in die Kommandozeile des "ISPF Primary Option Menu" ein und betätigen die Eingabetaste.

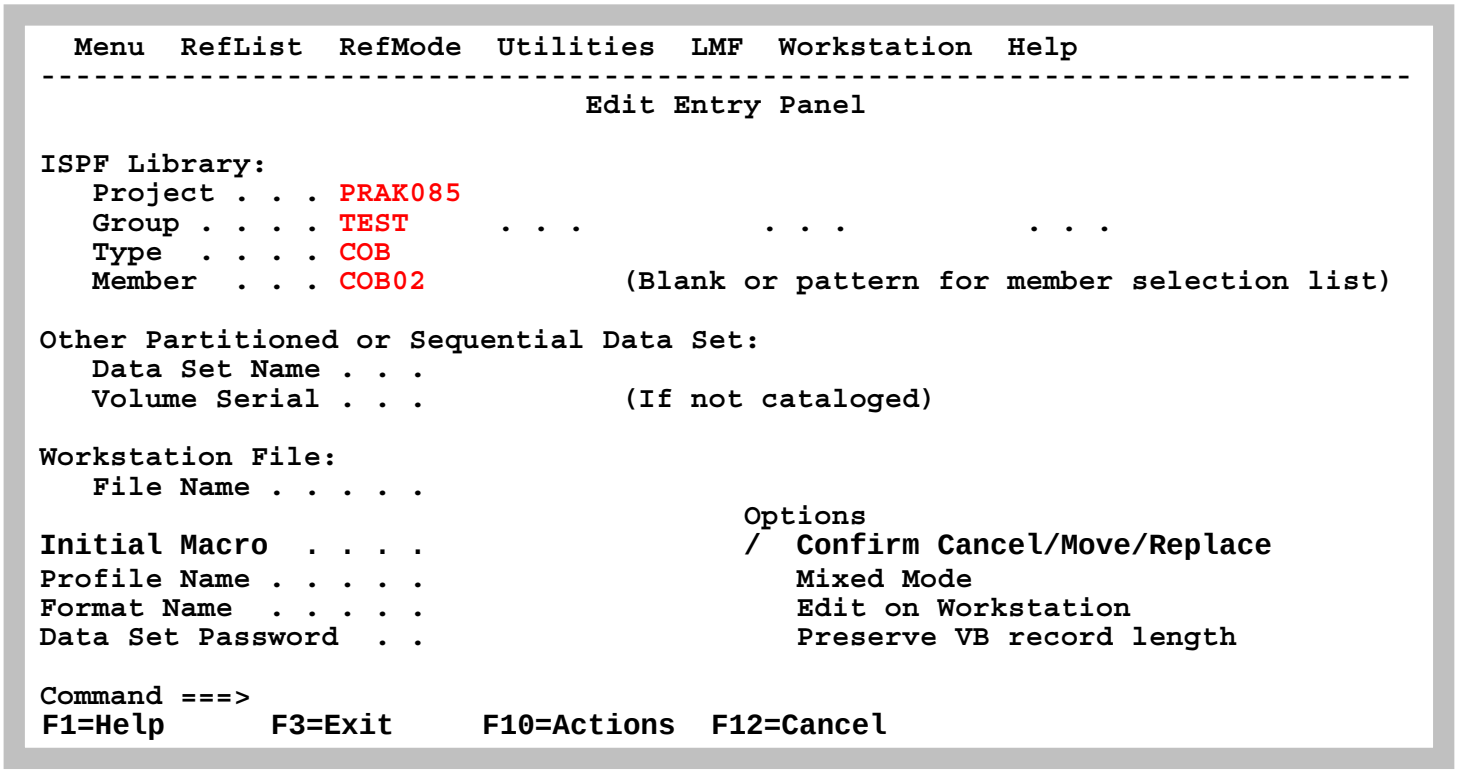


Abb. 3.2 "Edit Entry"-Bildschirm

Wir wollen zuerst das Cobol Quellprogramm mit Hilfe des ISPF-Editors erstellen. Der "Edit Entry"-Bildschirm fordert uns auf, den Namen des zu editierenden Programms einzugeben (s. Abb. 3.2).

Unser Quellprogramm soll als eine (von potentiell mehreren) File in dem für Quellprogramme von uns vorgesehenen Partitioned Dataset PRAK085.TEST.COB gespeichert werden. Files innerhalb eines Partitioned Datasets werden als Members bezeichnet. Zur Unterscheidung brauchen die einzelnen Members einen Namen.

Wir bezeichnen unseren Member als COB02. Der volle Name dieses Members ist PRAK085.TEST.COB.(COB02). Wir geben diese Werte in die dafür vorgesehenen Felder des "Edit Entry"-Bildschirmes ein. Es ist also wie in Abb. 3.2 gezeigt, ihre Benutzer-ID ins Feld "*Project*", "TEST" ins Feld "*Group*", "COB" ins Feld "*Type*" sowie "COB02" ins Feld "*Member*" einzutragen. Anschließend betätigen Sie die Eingabetaste.

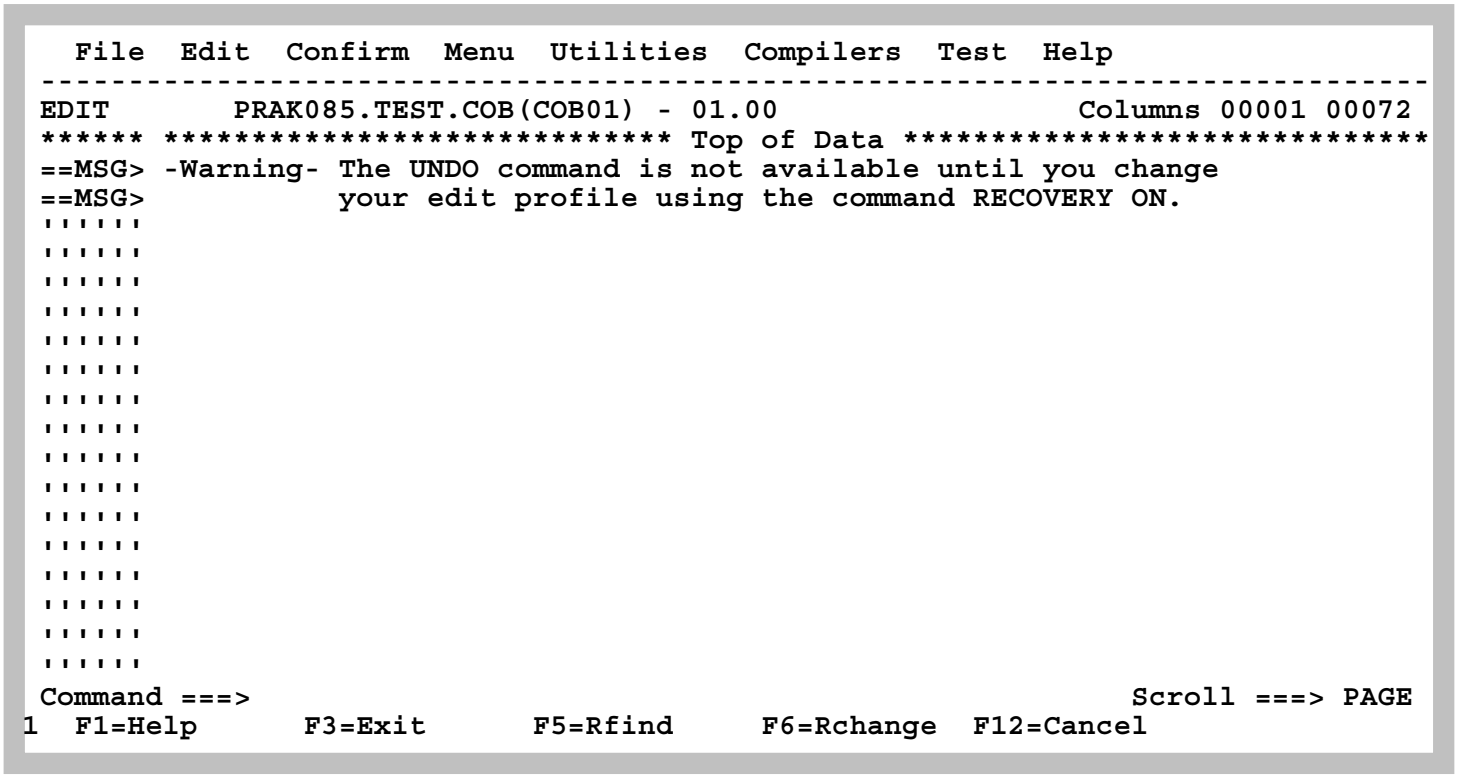


Abb. 3.3: ISPF-Editor

Abb. 3.3 zeigt die Oberfläche des ISPF-Editors. Für die Eingabe des Programmtextes benutzen wir nur die erforderlichen alphanumerischen Tasten. Wir verwenden keine Tasten zur Steuerung außer der DEL-Taste. Wir bewegen den Cursor mit Hilfe der Pfeiltasten.

Zusätzliche ISPF Editor Funktionen sind in dem Tutorial 1.b , ISPF Editor, beschrieben.

An dieser Stelle empfehlen wir Ihnen dringend, aus dem Tutorial 1b, ISPF Editor, die kurzen Abschnitte

- **5.1 Text verbergen,**
- **5.2 Spalten anzeigen**
- **5.3 Hex Darstellung**

sich anzusehen und/oder zu wiederholen-

```
000100         IDENTIFICATION DIVISION.
000200         PROGRAM-ID. COB02.
000300         ENVIRONMENT DIVISION.
000400         INPUT-OUTPUT SECTION.
000410
000500         FILE-CONTROL.
000600             SELECT PRINTOUT
000700                    ASSIGN TO SYSPRINT.
000800         DATA DIVISION.
000900         FILE SECTION.
001000         FD      PRINTOUT
001100                 RECORD CONTAINS 80 CHARACTERS
001200                 RECORDING MODE F
001300                 BLOCK CONTAINS 0 RECORDS
001400                 LABEL RECORDS ARE OMITTED.
001500         01  PRINTREC         PIC X(80).
001600         WORKING-STORAGE SECTION.
001610
001700         LINKAGE SECTION.
001800         PROCEDURE DIVISION.
001900         anfang.
002000             OPEN OUTPUT PRINTOUT.
002100             move 'Hallo Welt, unser erstes TSO-PROGRAMM in COBOL' to prin
002200        -    trec.
002300             WRITE PRINTREC.
002400             CLOSE PRINTOUT.
002500             GOBACK.
002600
```

Minuszeichen beachten, der Name der Variablen ist „printrec“

Abb. 3.4

Jedes Cobol Programm besteht aus 4 „Sections“ : Identtification Division, Environment Division, Data Division und Procedure Division. Die Divisions können wiederum aus mehreren Sections bestehen.

Wir erstellen das in Abb, 3.4 gezeigte Cobol Programm. Zum besseren Verständnis sind die Divisions und Sections farbig markiert.

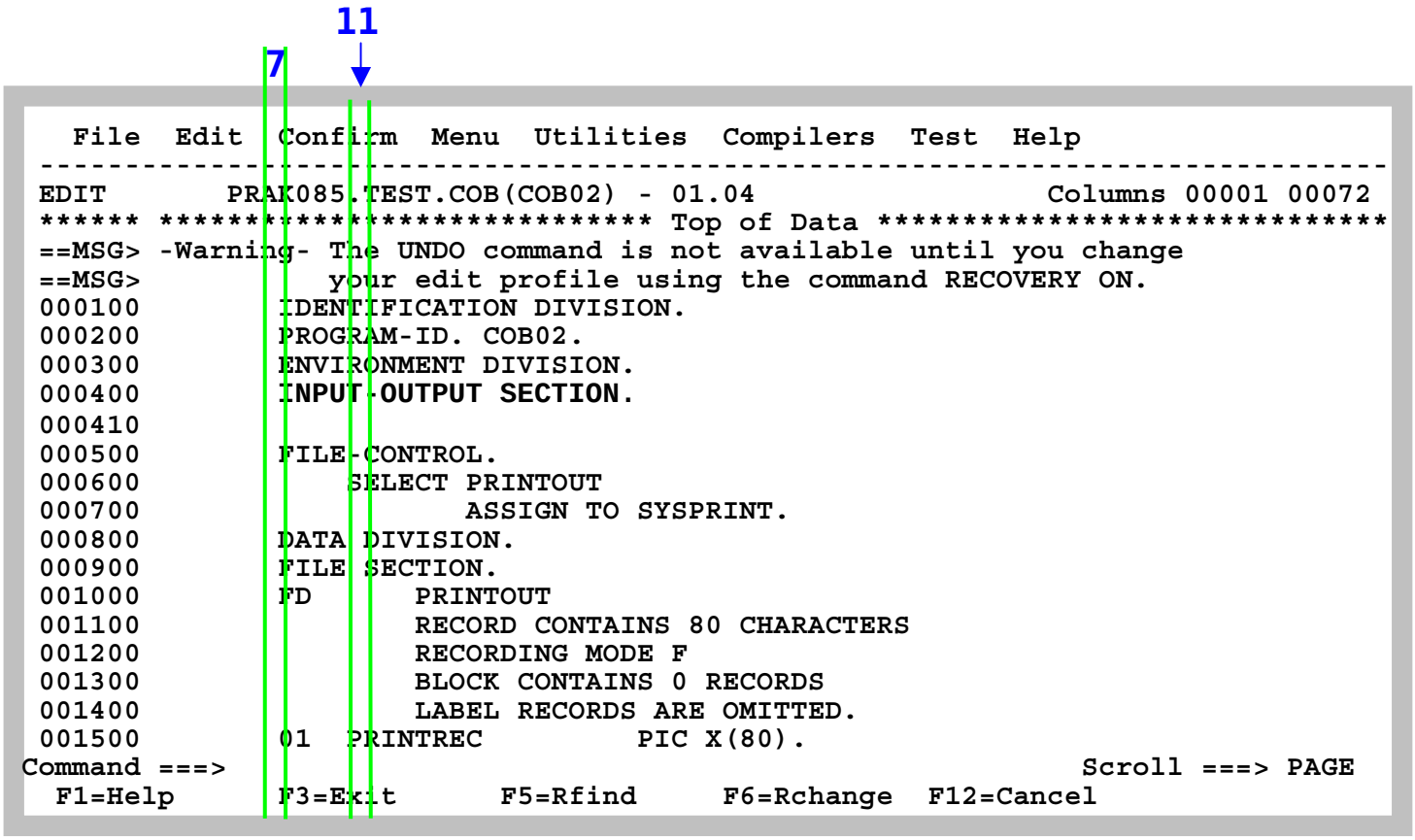


Abb. 3.5: ISPF-Editor mit COBOL-Programm (1/2)

Die Spalten 7 und 12 sind in Abb. 3.5 mit grünen Linien markiert.

Cobol Quellprogramme sind unter ISPF genauso spaltenabhängig wie JCL Scripts. Spezifisch sind:

Spalten	**1-6**	**Numerierung**
Spalte	**7**	**Zeilenkennzeichen**
		blank: Leerzeichen
		- Folgezeile
		*** Kommentar**
		/ Kommentar +Seitenwechsel
Spalte	**8-11**	**Bereich A (Überschriften = Label)**
Spalte	**12-72**	**Bereich B (Befehle)**
Spalte	**73-80**	**frei**

Hier ist vielleicht das COLS Kommando nützlich, siehe Tutorial 1b, Abschnitt 5.2 .

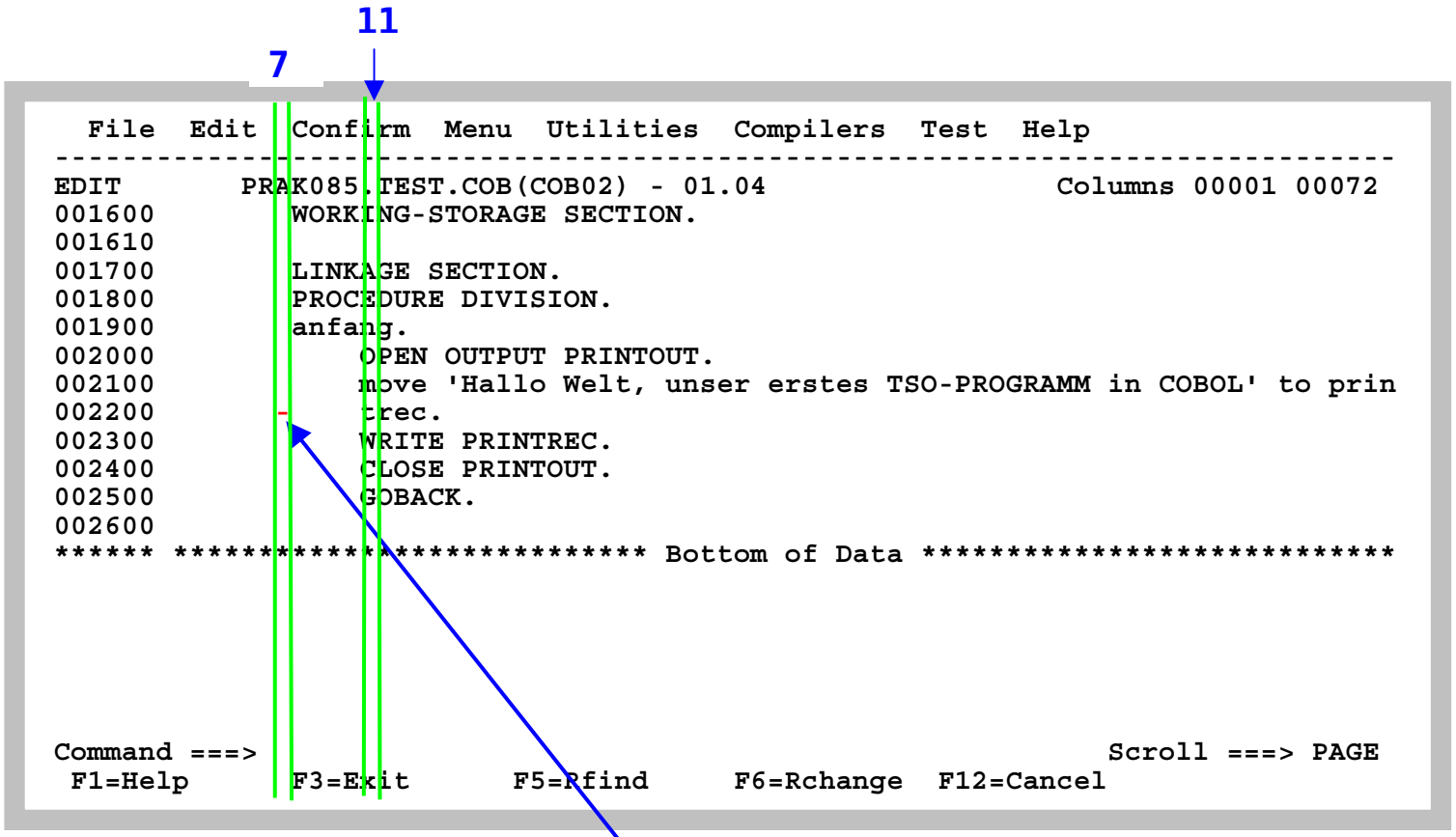


Abb. 3.6: ISPF-Editor mit COBOL-Programm, Fortsetzung (2/2)

Abbildung 3.6 ist die Fortsetzung von Abb. 3.5. Die Spalten 7 und 12 sind in Abbildung 3.6 mit grünen Linien markiert. Beachten sie in Zeile 002200 das Minuszeichen in Spalte 7, welches eine Fortsetzung von Zeile 002100 markiert !!!

Abb. 3.5 und Abb. 3.6 zeigen die beiden Teile des Programms im ISPF Fenster. Durch Betätigen der F3-Taste kehren wir zum vorherigen Bildschirm zurück. Unser Programm wird automatisch abgespeichert (saved).

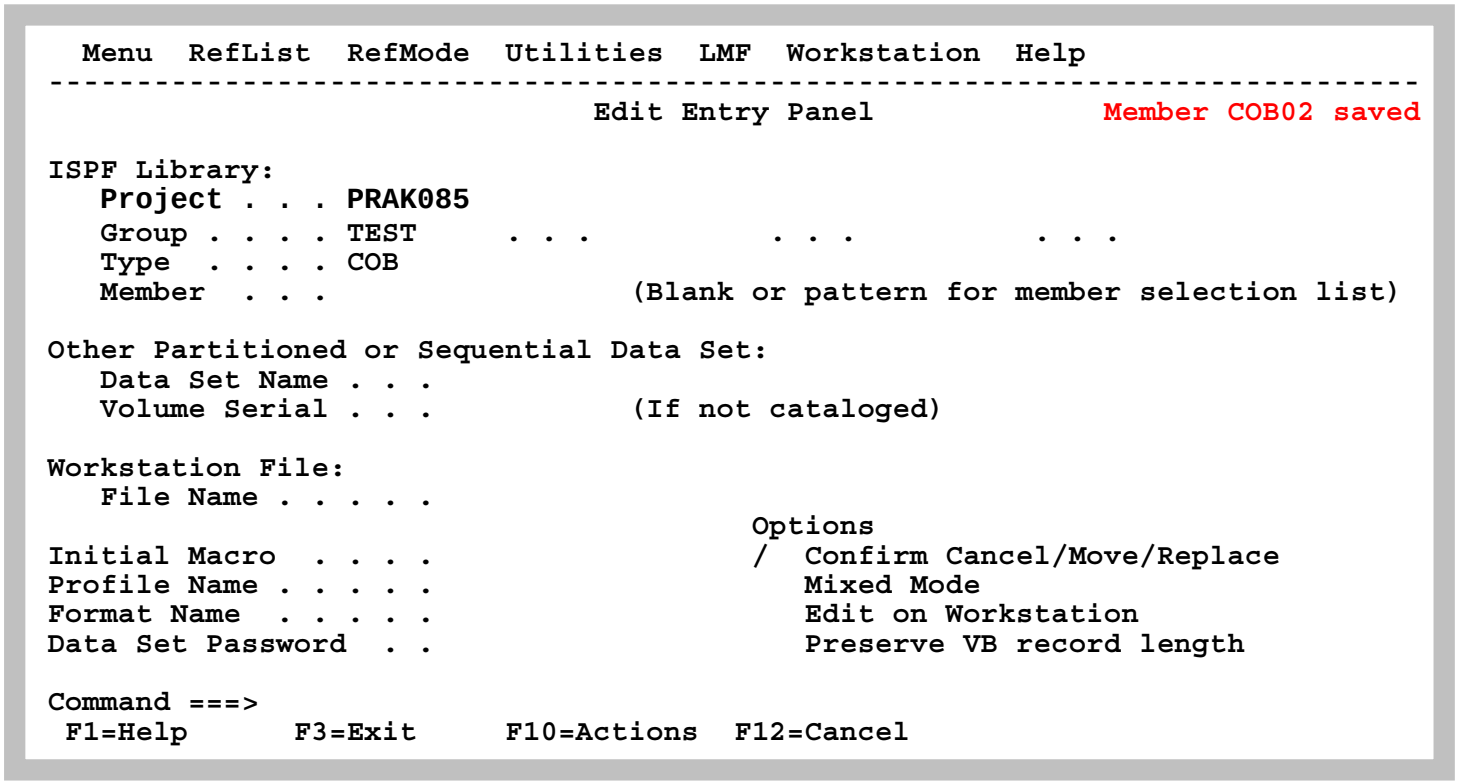


Abb. 3.7: "Edit Entry Panel"-Bildschirm

Rechts oben erscheint die Meldung, dass unser Member abgespeichert wurde (Abb. 3.7).

Durch erneutes Eingeben des Member-Namens sowie durch Bestätigen mit der Eingabetaste lässt sich unser Member nochmals aufrufen und bei Bedarf modifizieren.

Literatur zum Thema Cobol Programmierung:

Tutorial Mainframe
http://www.mainframegurukul.com/tutorials/programming/cobol/cobol-tutorial.html

S/360 Tutprial
http://www.mainframes360.com/2009/08/cobol-tutorial-introduction-writing.html

1 COBOL Programming Course
http://www.csis.ul.ie/cobol/course/Default.htm

4. Erstellen und Ausführung des JCL-Scriptes

4.1 Job Control Language

Scriptsprachen sind Programmiersprachen, deren Ziel es ist, Anweisungsfolgen oder kleinere Anwendungen zu realisieren. Programme, die in Skriptsprachen geschrieben sind, werden auch Skripte oder Scripts genannt. Microsoft verwendet häufig die Bezeichnung Makro. Scripte werden vielfach nicht von einem Compiler in maschinenlesbaren Code übersetzt, sondern zur Laufzeit von einem Interpreter ausgeführt.

Unter z/OS dominieren die beiden Scriptsprachen JCL (Job Control Language) und REXX.

REXX ist eine recht normale Scriptsprache, vergleichbar mit Perl, PHP, Python, Ruby oder Tcl.

JCL ist dagegen – sagen wir – anders. Für viele Neulinge ist es ein Kulturschock.

JCL enstammt dem Lochkartenzeitalter. Es hat eine ungewöhnliche Syntax und ähnliche Spaltenabhängigkeiten wie Cobol unter dem ISPF Editor.

Die positiven Eigenschaften sind: JCL ist sehr mächtig, und ist, nachdem man über die Anfangsschwierigkeiten hinweg ist, sehr leicht erlernbar. Programmierer, die sich einmal mit JCL auseinandergesetzt haben, werden sehr schnell zu gläubigen Bekennern. Für alle von IBM unterstützen Mainframe Sprachen existieren immer zwei Handbücher: User's Guide und Reference Manual. Die JCL User's Guide und Reference Manual sind im Umfang deutlich kürzer als die äquivalenten Dokumente für andere Mainframe Sprachen.

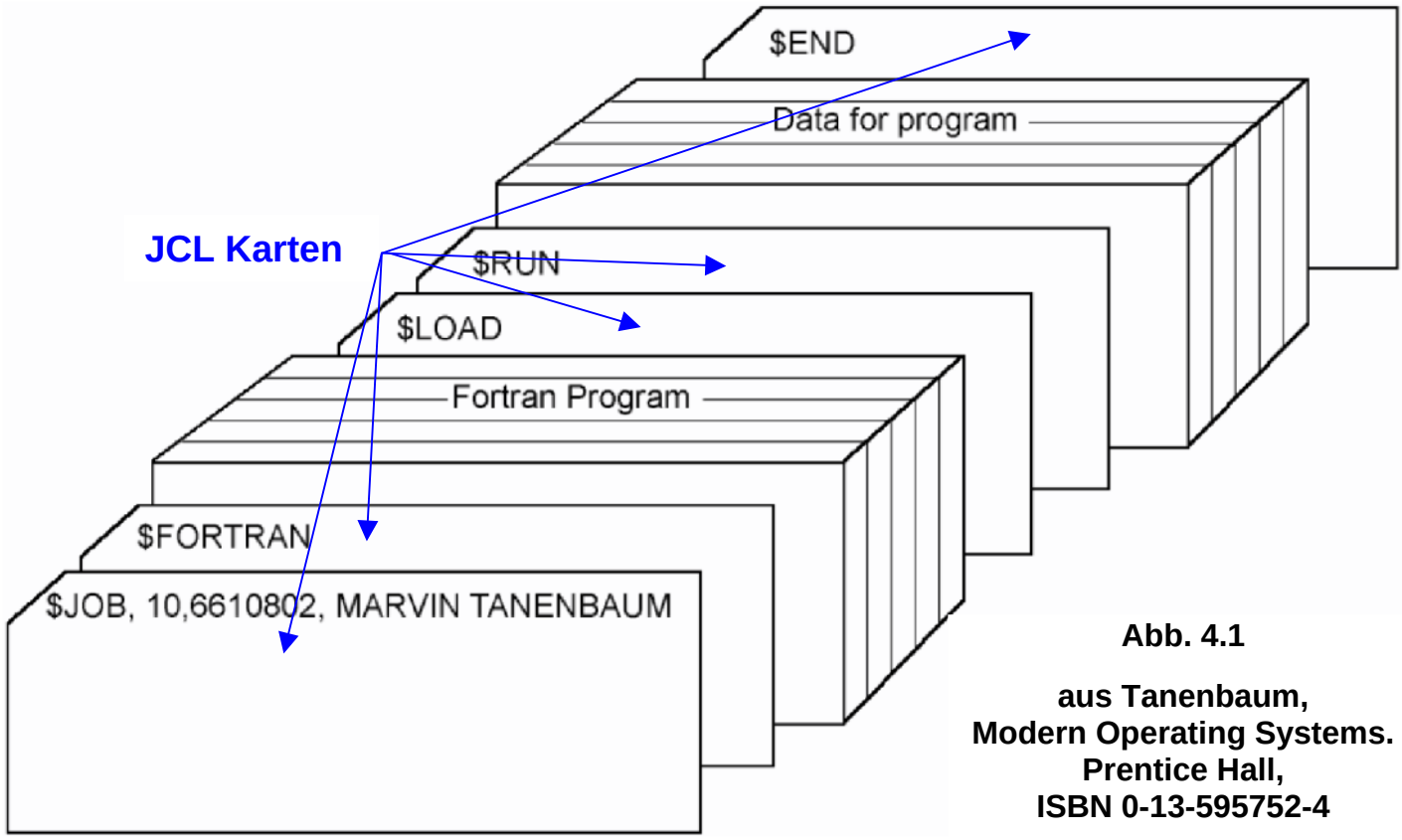


Abb. 4.1

**aus Tanenbaum,
Modern Operating Systems.
Prentice Hall,
ISBN 0-13-595752-4**

Gezeigt ist die Implementierung eines FORTRAN Programms etwa Anno 1960. Es besteht aus einer Reihe von JCL Karten, ein JCL Statement pro Karte. Zwischen den JCL Karten befindet sich das FORTRAN Programm Karten Deck, jeweils eine Karte pro FORTRAN Statement, sowie ein weiteres Karten Deck mit den von dem vom FORTRAN Programm zu verarbeitenden Daten. Das gesamte Kartendeck wurde von dem Lochkartenleser des Rechners in den Hauptspeicher eingelesen und verarbeitet. Die Ausgabe erfolgte typischerweise mittels eines angeschlossenen Druckers.

Später wurde es üblich, das FORTRAN Programm und die Daten auf Magnetband (und noch später auf einem Plattenspeicher) zu speichern, und das Programmdeck und das Datendeck durch zwei Library Call Lochkarten zu ersetzen. Noch später speicherte man dann das JCL Script ebenfalls in einer Library auf dem Plattenspeicher.

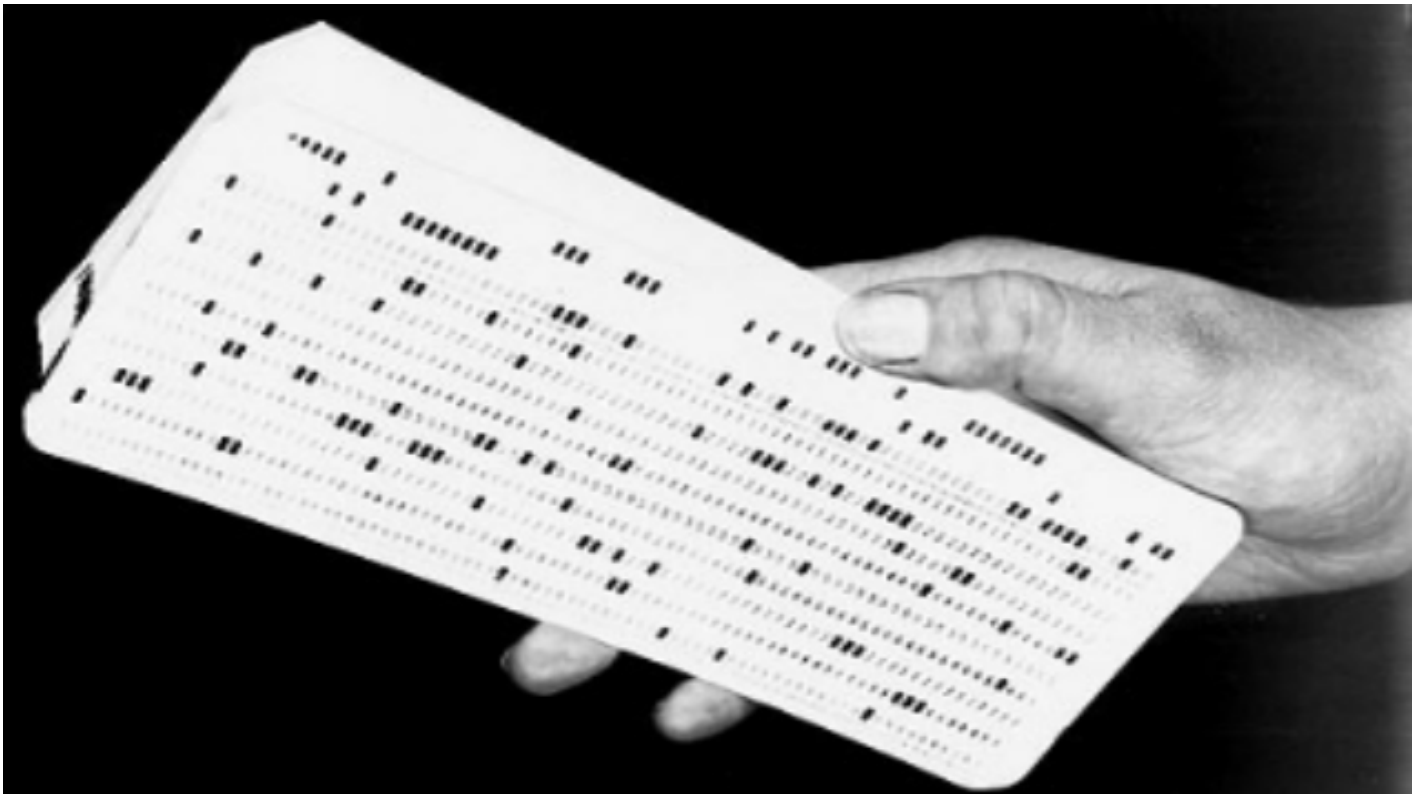


Abb. 4.2 Kartendeck

Hier hält ein Programmierer in der Hand ein Deck mit etwa 100 Lochkarten, die ein Programm enthalten.

4.2 JCL Format

Ein JCL Script besteht aus einzelnen JCL Statements. Ein JCL Statement passte früher auf eine Lochkarte mit 80 Spalten, und an der Länge von 80 Bytes hat sich bis heute nichts geändert.

Wie auch bei anderen Programmiersprachen besteht ein JCL Statement aus genau fünf) Feldern.

/ /	Name	Operation	Parameter	Kommentar

Abb. 4.2
Felder eines JCL Statements

1. **// (zweimal Slash) in Spalte 1 und 2**
2. **Label Feld (Name), bis zu 8 Zeichen lang, beginnt in Spalte 3**
3. **Statement Type (auszuführende Operation)**
4. **Parameter**
5. **Kommentar**

Um die Logik zum Parsen eines Statements zu vereinfachen, führte man einige Konventionen ein:

- **Jedes Feld beginnt immer an einer bestimmten Spalte der Lochkarte. Das vereinfachte damals das Parsen des Statements. Diese Spaltenabhängigkeit existiert auch heute noch !!! Wenn Sie dagegen verstoßen, gibt es eine Fehlermeldung. Warnung: Dies ist mit großem Abstand der häufigste Programmierfehler beim Bearbeiten unserer ersten Mainframe Tutorials.**
- **Jedes JCL Statement beginnt mit den beiden Zeichen / / (forward slashes). Damit war es möglich, die JCL Statements leicht von den Lochkarten des FORTRAN Decks und des Datendecks zu unterscheiden. Die beiden Slashes befinden sich in Spalte 1 und 2.**
- **Das Namensfeld (Label) Feld beginnt in Spalte 3, die maximale Länge ist 8 Zeichen.**
- **Das Operation Feld folgt dem Namensfeld, getrennt durch ein Leerzeichen. (früher musste es in Spalte 12 anfangen, heute nicht mehr erforderlich).**
- **Das Parameter Feld (Operand Feld) folgt dem Operation Feld, getrennt durch ein (oder mehr) Leerzeichen.**
- **Alles was hinter dem Operand Feld folgt, ist ein Kommentar. Zwischen Operand und Kommentar muss sich (mindestens) ein Leerzeichen befinden.**

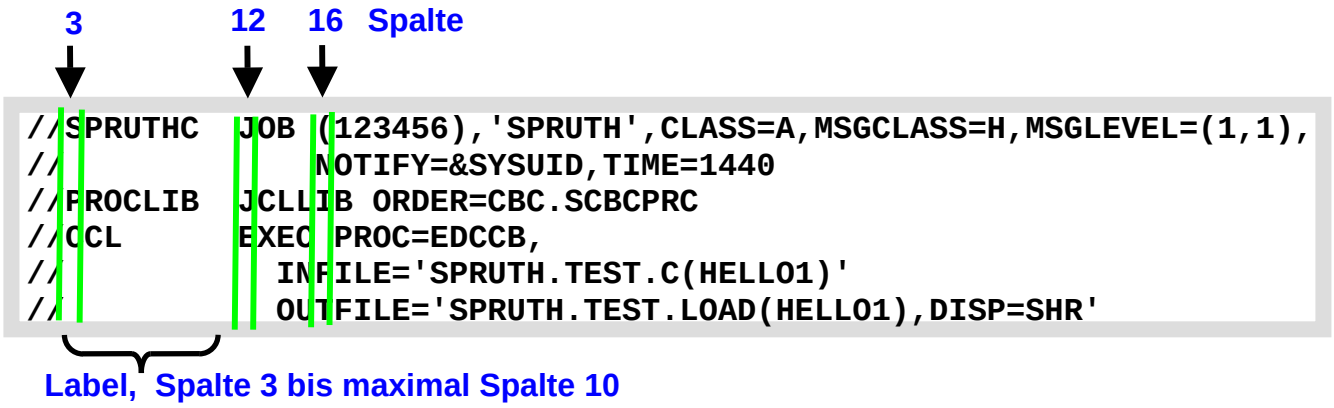


Abb. 4.3 Ein einfaches JCL Script

Unser JCL-Script macht einen reichlich kryptischen Eindruck. JCL-Scripte werden dadurch gekennzeichnet, dass alle Zeilen in Spalten 1 und 2 mit "//" beginnen.

Spalten 3 bis 10 enthalten normalerweise einen Namen (Label), der zwischen 1 und 8 Zeichen lang sein kann.

Beginnend in Spalte 12 wird die Operation spezifiziert. Einige Beispiele:

- **JOB Statement markiert den Anfang eines Jobs**
- **EXEC Statement bezeichnet Prozedur, die ausgeführt werden soll**
- **PROC Statement gibt die Adresse einer Procedur an**
- **DD Statement bezeichnet die zu benutzenden Dateien**

Auf die Operation folgt die Parameterliste in dem Parameterfeld (Operandenfeld). Jeder Parameter ist von dem folgenden Parameter durch ein Komma getrennt. Kein Leerzeichen zwischen den einzelnen Parametern.

Die Spalten 72 – 80 werden von JCL nicht berücksichtigt. Wenn Ihre Parameterliste über Spalte 71 hinausgeht, dann

1. **unterbrechen sie die Parameterliste nach dem letzten vollständigen Parameter, einschließlich des Kommas,**
2. **kodieren Sie / / in den Spalten 1 und 2 der folgenden Zeile**
3. **kodieren Sie ein Leerzeichen in Spalte 3 der folgenden Zeile. Wenn die Spalte 3 alles andere als ein Leerzeichen oder ein Asterik enthält, interpretiert JCL diese Zeile als ein neues JCL Statement.**
4. **Vollenden Sie die Parameter Liste. Die Parameter Liste soll frühestens in Spalte 4 und spätestens in Spalte 16 anfangen.**

An die Parameterliste schließt sich das Kommentarfeld an, getrennt von der Parameterliste durch ein Leerzeichen. Ebenso ist eine Zeile mit den Symbolen //* in Spalten 1, 2 und 3 ein Kommentar.

Achten Sie darauf, dass Ihr JCL Script die richtigen Spalten benutzt !!!

Unser Beispiel in Abb. 4.3 enthält drei Operationen JOB, JCLLIB und EXEC. Die Parameterliste des Job Statements erstreckt sich bis auf die zweite Zeile (Leerzeichen in Spalte 3, Fortsetzung der Parameterliste in Spalte 16. Ebenso hat das EXEC Statement eine Fortsetzung über 2 weitere Zeilen.

Beispiel für einen JCL Befehl:

```
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)
```

RECFM, FB, LRECL und BLKSIZE sind Schlüsselwörter der JCL Sprache.

Der Befehl besagt, dass die hiermit angesprochene Datei (bzw. deren Data Control Block, DCB) ein Fixed Block (FB) Record Format (RECFM) hat (alle Datensätze haben die gleiche Länge), dessen Länge (Logical Record Length LRECL) 80 Bytes beträgt, und dass für die Übertragung vom/zum Hauptspeicher jeweils 5 Datensätze zu einem Block von (Blocksize BLKSIZE) 400 Bytes zusammengefasst werden.

4.3 DD Statement

In einem JCL Script definieren Data Definition (DD) Anweisungen die Data Sets, welche ein Programm oder eine Prozedur bei der Ausführung verwendet. Sie müssen ein DD-Statement für jeden Data Set kodieren, der während eines Auftrags Schritt verwendet oder erstellt wird.

Die Reihenfolge der DD-Statements innerhalb eines Auftrags Schrittes ist in der Regel nicht signifikant.

Diese JCL Beispiel veranschaulicht das Format eines DD-Anweisung

```
//PAY DD DSN=HLQ.PAYDS,DISP=NEW VENDOR PAYROLL
```

Das Operations Feld enthält den Eintrag DD (für Data Definition).

Das Namens-Feld enthält einen ein-bis acht-stelligen Namen (hier PAY), als ddname bekannt. Dieser kennzeichnet die DD-Anweisung, so dass andere JCL, Programme, Verfahren, oder das Betriebssystem darauf verweisen können. Der ddname in dieses DD-Anweisung ist PAY.

Das Parameter-Feld enthält zwei Keyword-Parameter:

- **DSN, welches den echten Namen eines Datensatzes identifiziert,**
- **DISP, welches HLQ.PAYDS als einen neuen Datensatz identifiziert, den das System erstellen muss, wenn dieser Job zur Verarbeitung übermittelt wird.**

Das Kommentar-Feld enthält den Eintrag VENDOR PAYROLL.

Ein DD Statement kann einen Data Set sehr umfangreich beschreiben. Es kann die folgenden Angaben enthalten:

- **Der Name, den das Programm verwendet, um den Datensatz beziehen, als ddname bekannt**
- **Der eigentliche Name des Datensatzes und seine Lokation**
- **Physische Eigenschaften des Datensatzes, wie z.B. das Record Format**
- **Die Anfangs-und Endzustand des Datensatzes, als Disposition bezeichnet**

Sie können auch DD Statements benutzen um I/O-Geräte anzufordern, oder um Speicherplatz für neue Datensätze bereitzustellen (allocate).

4.3 Erstellen und Ausführung des JCL-Scriptes

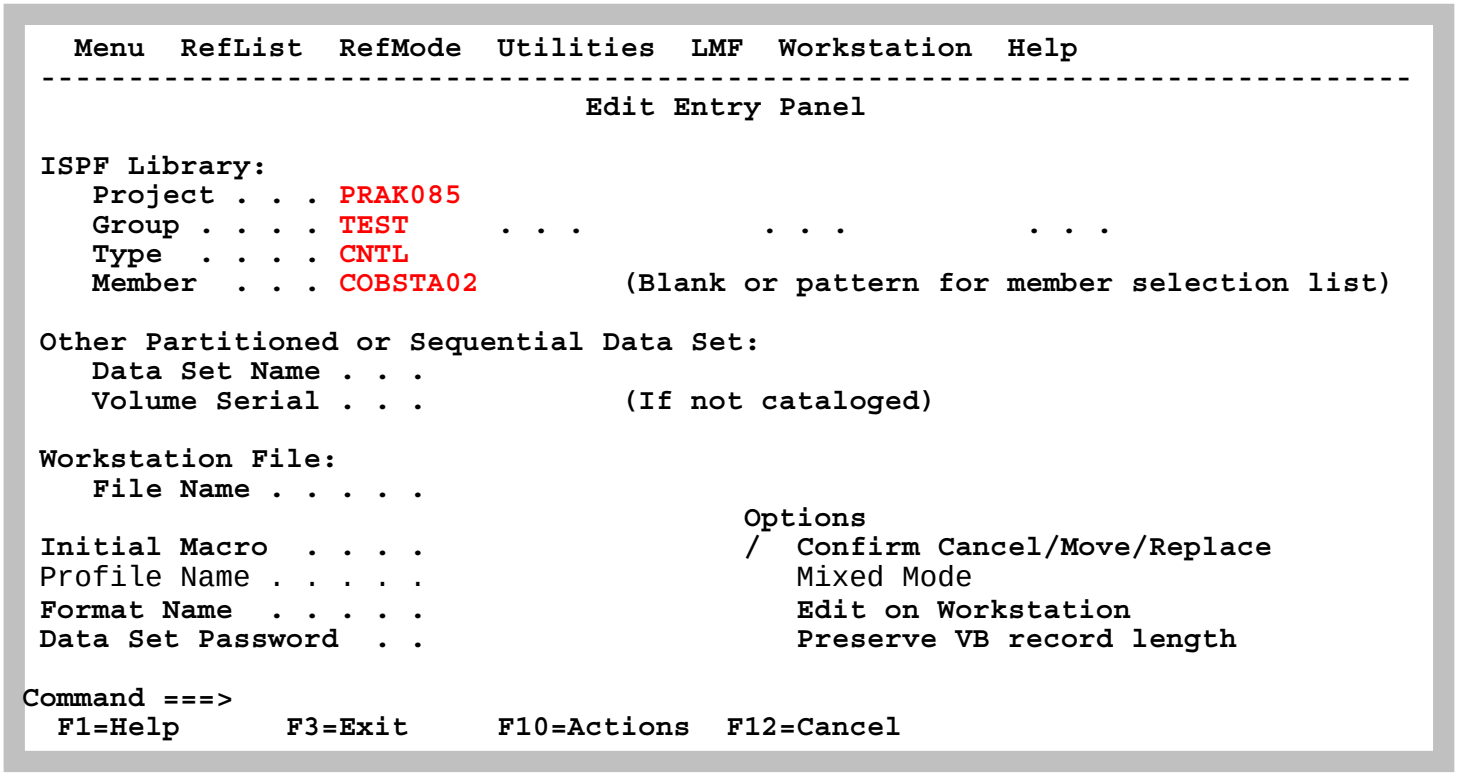


Abb. 4.4: "Edit Entry Panel"-Bildschirm

Wir legen alle "JCL Scripts" als Member in dem von uns dafür vorgesehenen Partitioned Dataset PRAK085.TEST.CNTL ab. Wie bereits erläutert, wollen wir die zwei Schritte Compile sowie Link Edit jedoch mit einem einzigen JCL Script durchführen.

Die hierfür verwendete Scriptsprache ist die "z/OS Job Control Language" (JCL).

Wir geben als Typ "CNTL" sowie als Member "COBSTA02" ein und betätigen die Eingabetaste (s. Abb. 4.4).

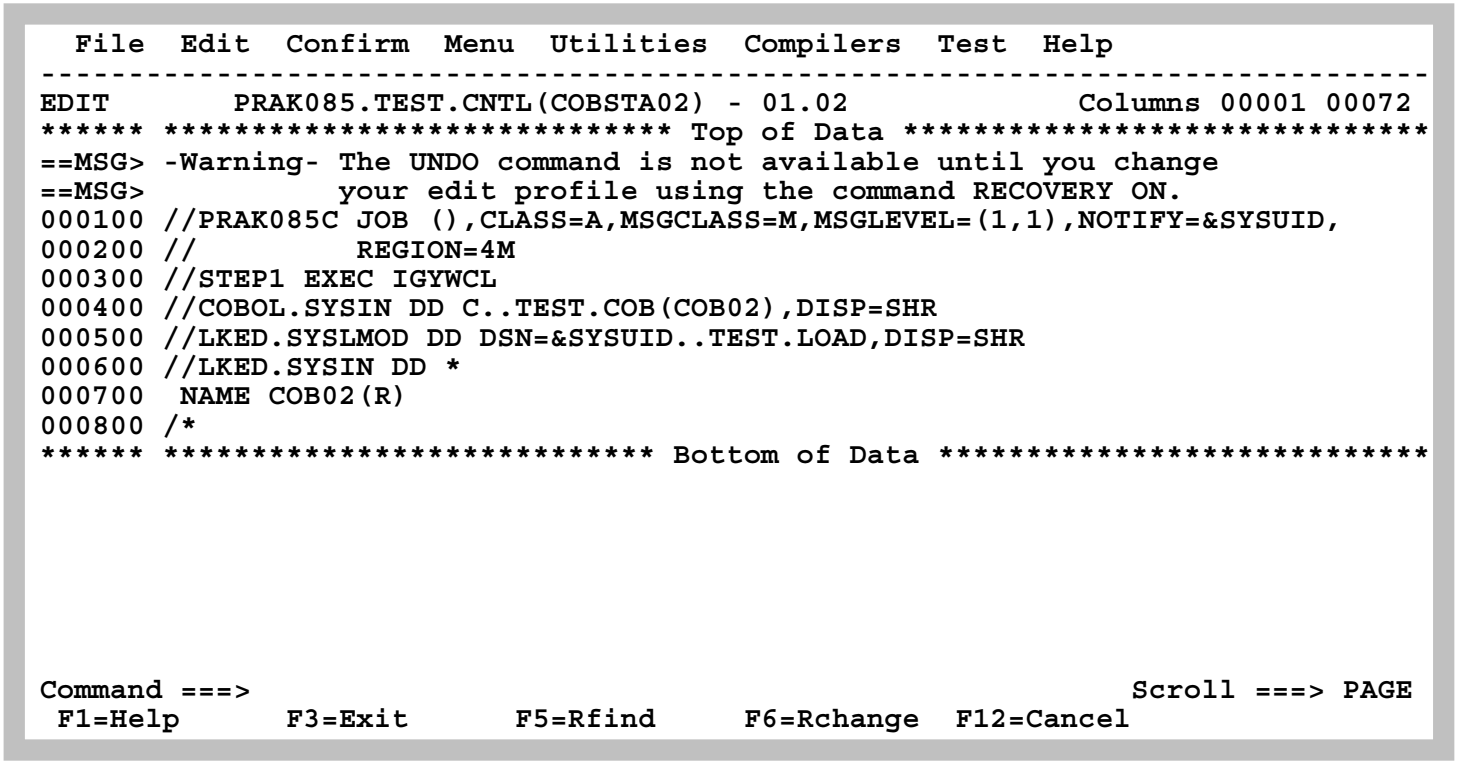


Abb.4.5: JCL-Script
Erstellen Sie das in Abb. 4.5 gezeigte JCL-Script.

Aufgabe: ***Erstellen Sie das in Abb. 4.5 gezeigte JCL-Script.***

Das erste Statement in einem JCL-Script ist immer ein "JOB" Statement (Zeile 000100 in Abb. 4.5). Es enthält eine Reihe von Dispositionsparametern, die von dem "z/OS Job Entry Subsystem" ausgewertet werden. Es ist üblich, als Label für das Job-Statement die TSO-Benutzer-ID (hier "PRAK085") plus einen angehängten Buchstaben (hier C) zu verwenden. Das hierfür vorgesehene Feld hat eine Länge von maximal 8 Zeichen. Aus diesem Grund haben TSO-Benutzer-ID's eine maximale Länge von 7 Zeichen.

Das zweite Statement (Zeile 000300 in Abb. 4.5) unseres Scripts ist ein EXEC Statement. Es enthält die Anweisung, die Prozedur "IGYWCL" abzuarbeiten. " IGYWCL ist ein von z/OS zur Verfügung gestelltes Script (Catalogued Procedure), welches

- **den COBOL-Compiler aufruft,**
- **anschließend den Linkage-Editor aufruft,**
- **den zu übersetzenden Quelltext als Member eines Partitioned Datasets (in unserem Fall) mit dem Namen PRAK085.TEST.COB(COB02) erwartet.**
- **das erstellte Maschinenprogramm (in unserem Fall) unter PRAK085.TEST.LOAD abspeichert.**

Es existiert eine große Anzahl derartiger vorgefertigter Scripte, die zusammen mit z/OS ausgeliefert werden. Der Systemadministrator stellt sie in "JCL Libraries" (JCLLIB) zusammen. z/OS ist ein sehr großes und sehr flexibles System. Es existieren häufig mehrere JCL Libraries. Was, wie und wo ist von einer Installation zur nächsten oft verschieden und wird vom Systemadministrator verwaltet.

Wenn Sie unter z/OS einen Cobol Compiler installieren, gehört sowie eine ganze Reihe weiterer Prozeduren mit zum Lieferumfang. Das Handbuch „Enterprise COBOL for z/OS Programming Guide Version 4 Release 1, SC23-8529-00, December 2007“ enthält in Kapitel 14 auf Seite 249 eine Beschreibung dieser Prozeduren.
Siehe http://www.cedix.de/VorlesMirror/Band3/ProgGuidezOS.pdf

Das JCL Script entnimmt den High Level Qualifier des Cobol Programms PRAK085.TEST.COB(COB02) der Zeile 000100 des JCL Scripts. Der Rest des Namens wird in Zeile 000400 angegeben. Zeile 000500 definiert den File Namen, unter dem das übersetzte Programm abgespeichert wird PRAK085.TEST.LOAD(COB02). Es wird unterstellt, dass der Name des Members unverändert bleibt. IGYWCL benutzt hierfür den Namen SYSLMOD.

```
//IGYWCL PROC  LNGPRFX='IGY.V4R1M0',SYSLBLK=3200,
//             LIBPRFX='CEE',
//             PGMLIB='&&GOSET',GOPGM=GO
//*
//*  CALLER MUST SUPPLY //COBOL.SYSIN DD . . .
//*
//COBOL  EXEC PGM=IGYCRCTL,REGION=2048K
//STEPLIB  DD  DSNAME=&LNGPRFX..SIGYCOMP,                (1)
//             DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSLIN   DD  DSNAME=&&LOADSET,UNIT=SYSDA,
//             DISP=(MOD,PASS),SPACE=(TRK,(3,3)),
//             DCB=(BLKSIZE=&SYSLBLK)
//SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT2   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT3   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT4   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT5   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))              (2)
//SYSUT6   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT7   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//LKED   EXEC PGM=HEWL,COND=(8,LT,COBOL),REGION=1024K
//SYSLIB   DD  DSNAME=&LIBPRFX..SCEELKED,                (3)
//             DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSLIN   DD  DSNAME=&&LOADSET,DISP=(OLD,DELETE)
//         DD  DDNAME=SYSIN
//SYSLMOD  DD  DSNAME=&PGMLIB(&GOPGM),
//             SPACE=(TRK,(10,10,1)),
//             UNIT=SYSDA,DISP=(MOD,PASS)
//SYSUT1   DD  UNIT=SYSDA,SPACE=(TRK,(10,10))
```

Abb. 4.6 : Library Procedur `IGYWCL`

Dargestellt ist die Compile and link-edit Prozedur IGYWCL

(1)
STEPLIB can be installation-dependent.

(2)
SYSUT5 is needed only if the LIB option is used.

(3)
SYSLIB can be installation-dependent.

Unter anderem enthält IGYWCL (Abb. 4.6) die Namen COBOL.SYSIN, SYSLMOD und SYSIN, die in unserem JCL Script auftauchen (Abb. 4.5). Der Parameter LKED (linked) besagt, dass eine Verknüpfung besteht.

http://pic.dhe.ibm.com/infocenter/pdthelp/v1r1/index.jsp?topic=%2Fcom.ibm.entcobol.doc_4.1%2FPGandLR%2Fref%2Frpmvs06.htm

Als Alternative stellt z/OS eine IGYWCLG cataloged procedure zur Verfügung. Diese bewirkt COBOL compile, link, and go, also die unmittelbare Ausführung des übersetzten Programms.

Siehe
https://publib.boulder.ibm.com/infocenter/zos/basics/index.jsp?topic=/com.ibm.zos.zappldev/zappldev_116.htm

```
  File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
--------------------------------------------------------------------------------
EDIT       PRAK085.TEST.CNTL(COBSTA02) - 01.02            Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 //PRAK085C JOB (),CLASS=A,MSGCLASS=M,MSGLEVEL=(1,1),NOTIFY=&SYSUID,
000200 //         REGION=4M
000300 //STEP1 EXEC IGYWCL
000400 //COBOL.SYSIN DD DSN=&SYSUID..TEST.COB(COB02),DISP=SHR
000500 //LKED.SYSLMOD DD DSN=&SYSUID..TEST.LOAD,DISP=SHR
000600 //LKED.SYSIN DD *
000700  NAME COB02(R)
000800 /*
****** **************************** Bottom of Data ****************************




Command ===> SUB                                          Scroll ===> PAGE
 F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
```

Abb. 4.7: Ausführung des JCL-Scriptes

Beachten Sie die beiden DD Statements in Zeile 000400 und 000500.

Unser Compile- und Link-Script kann nun ausgeführt werden. Wir geben, wie in Abb. 4.7 gezeigt, auf der Kommandozeile "SUB" (für Submit) ein und betätigen die Eingabetaste.

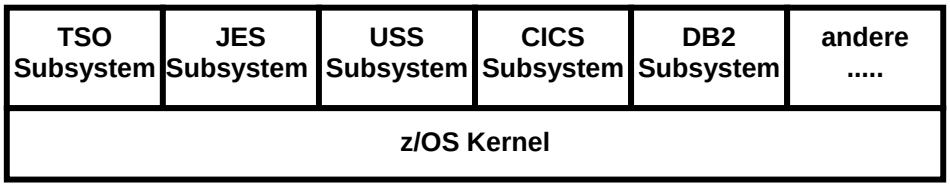


Abb. 4.8 z/OS Subsysteme

Das "Job Entry Subsystem" (JES) des z/OS-Betriebssystems dient dazu, Stapelverarbeitungsaufträge (Jobs) auf die einzelnen CPU's zu verteilen und der Reihe nach abzuarbeiten. Jobs werden dem "JES"-Subsystem in der Form von JCL-Scripten zugeführt, wobei deren erstes JCL-Statement ein JOB-Statement sein muss. PRAK085.TEST.CNTL(COBSTA02*)* ist ein derartiges Script. Das Kommando "SUB" (Submit) bewirkt, dass PRAK085.TEST.CNTL(COBSTA02*)* in die Warteschlange der von JES abzuarbeitenden Aufträge eingereiht wird.

z/OS gestattet es grundsätzlich, Programme entweder interaktiv im Vordergrund (unter TSO) oder als Stapelverarbeitungsprozesse durch JES im Hintergrund abzuarbeiten. Ersteres garantiert bessere Antwortzeiten, letzteres führt zu einem besseren Durchsatz.

Was wir hier machen, ist ein Quellprogramm zu übersetzen. Das kann mehr Zeit in Anspruch nehmen. Es ist deshalb üblich, die Durchführung nicht im Vordergrund durch TSO, sondern im Hintergrund auszuführen. Die Durchführung im Vordergrund hat den Vorteil, dass das übersetzte Programm direkt aufgerufen, und das Ergebnis direkt auf dem Bildschirm wiedergegeben werden kann. Die Durchführung im Hintergrund bewirkt, dass das übersetzte Programm in einem Data Set, nämlich `PRAK085.TEST.LOAD(COB02)`, abgespeichert wird. Es muss dann noch irgendwie, z.B. durch ein TSO Kommando, aufgerufen werden.

```
  File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       PRAK085.TEST.CNTL(COBSTA02) - 01.02            Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 //PRAK085C JOB (),CLASS=A,MSGCLASS=M,MSGLEVEL=(1,1),NOTIFY=&SYSUID,
000200 //          REGION=4M
000300 //STEP1 EXEC IGYWCL
000400 //COBOL.SYSIN DD DSN=&SYSUID..TEST.COB(COB02),DISP=SHR
000500 //LKED.SYSLMOD DD DSN=&SYSUID..TEST.LOAD,DISP=SHR
000600 //LKED.SYSIN DD *
000700  NAME COB02(R)
000800 /*
****** **************************** Bottom of Data ****************************




IKJ56250I JOB PRAK085C(JOB03979) SUBMITTED
***
```

Abb. 4.9
Meldung "JOB PRAK085C(JOB03979) SUBMITTED"

Der JCL-Kommando-Interpreter überprüft die Syntax des Scripts. Falls er keinen Fehler findet, übergibt (submitted) er den Job zur Abarbeitung an das JES-Subsystem. Die Meldung oberhalb der Kommandozeile besagt, dass dies hier der Fall ist (siehe Abb. 4.9). Der Job erhält die Nummer 03979. Diese Nummer kann z.B. vom Systemadministrator benutzt werden, um den Status der Verarbeitung dieses Jobs abzufragen. Es ist eine gute Idee, wenn Sie sich die Job Nummer (hier 03979) auf einem Stück Papier notieren.

Wir warten einige Sekunden und betätigen anschließend die Eingabetaste. Erscheint keine Meldung, hat JES das JCL-Script noch nicht endgültig abgearbeitet. Wir warten erneut einige Sekunden und Betätigen die Eingabetaste; wir wiederholen dies notfalls mehrfach, bis eine Statusmeldung, so ähnlich wie in Abb. 4.10 dargestellt ist, ausgegeben wird.

http://www.mainframes360.com/2010/02/submitting-job.html oder
http://www.cedix.de/VorlesMirror/Band3/submit.pdf
enthält eine Beschreibung, was beim Submit eines Jobs passiert.

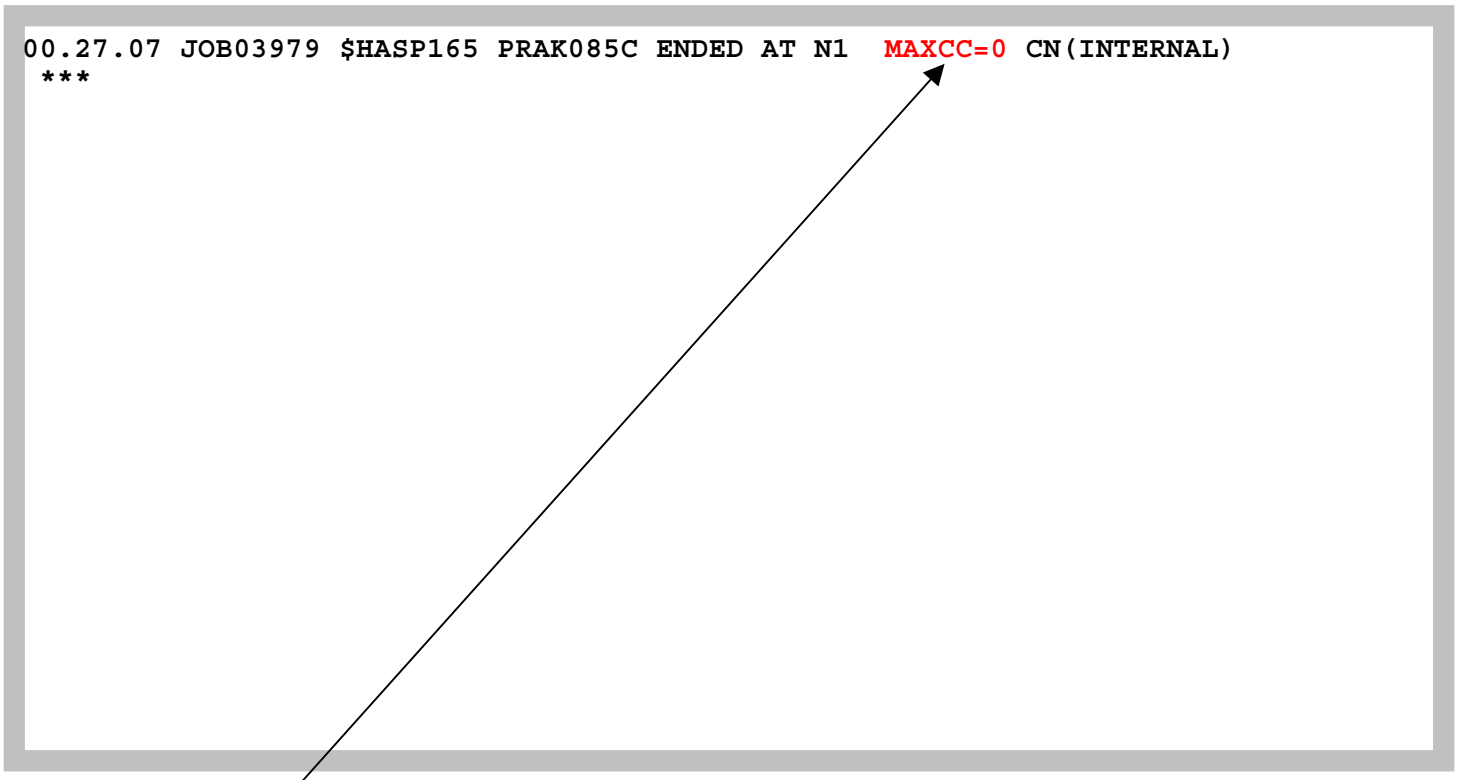


Abb. 4.10: Statusmeldung nach Abarbeitung des JCL-Scriptes

"MAXCC-0" ist eine Erfolgsmeldung: Die Übersetzung ist erfolgreich durchgeführt worden. "MAXCC-4" ist ebenfalls OK, alles andere besagt, dass ein Fehler aufgetreten ist. In diesem Fall hilft SDSF, siehe Abschnitt 6.

Das übersetzte Programm ist nun ausführungsfertig in dem File `PRAK085.TEST.LOAD(COB02)` abgespeichert.

5. Ausführung des COBOL-Programms

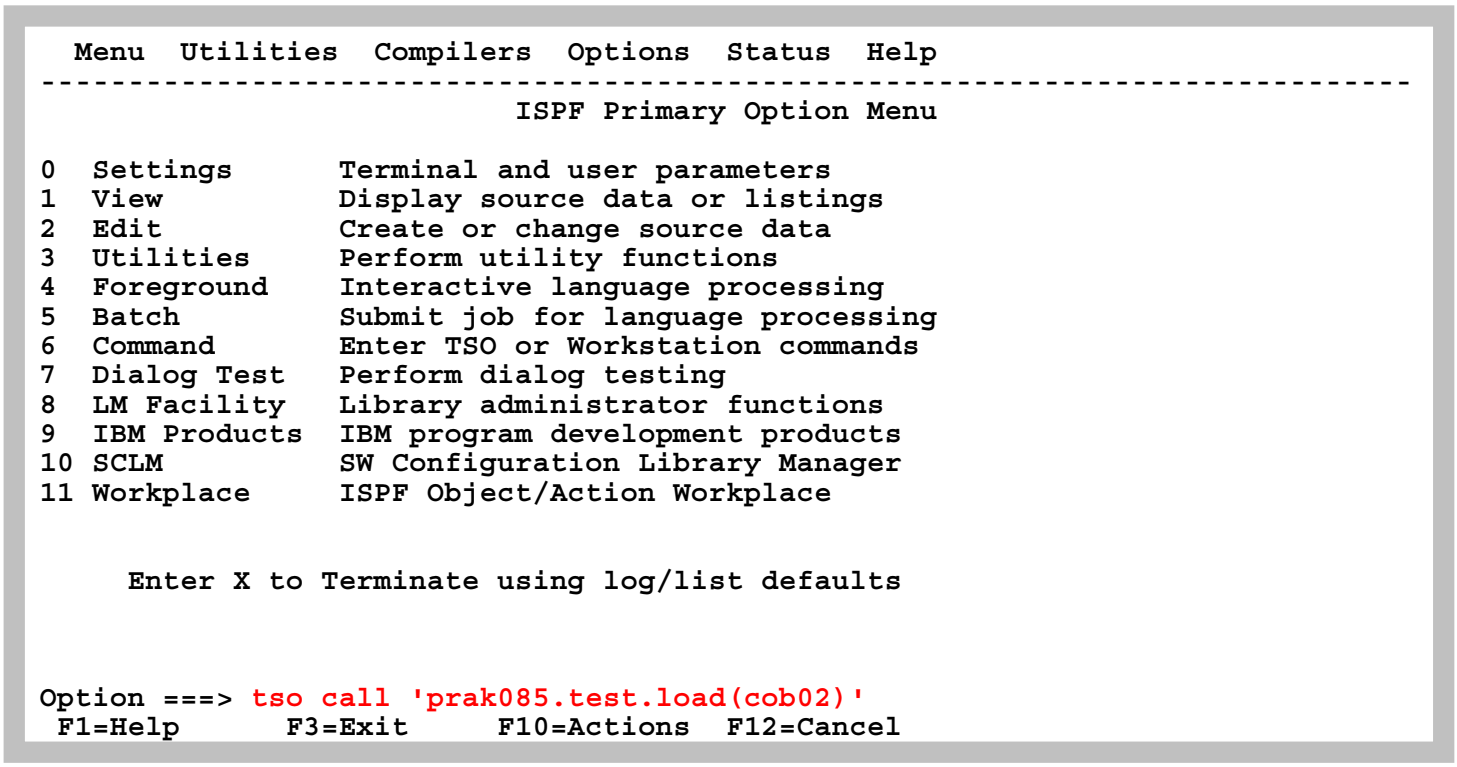


Abb. 5.1: "ISPF Primary Option Menu"-Bildschirm

Wir sind nun soweit, dass unser Programm ausgeführt werden kann. Durch mehrfaches Betätigen der F3-Taste kehren wir in das "ISPF Primary Option Menu" zurück (siehe Abb. 5.1).

Auf der Kommandozeile geben wir den Befehl

tso call 'prak085.test.load(cob02)'

ein und betätigen die Eingabetaste. "prak085.test.load(cob02)" enthält das vom Compiler erzeugte Maschinenprogramm. "call" ist ein TSO-Kommando und ruft ein Programm auf.

Wir sind aber im ISPF-Subsystem und nicht im TSO-Subsystem. "tso call" an Stelle von "call" bewirkt, dass der "call"-Befehl auch innerhalb des ISPF-Subsystems ausgeführt werden kann.

Dieses und die folgenden Tutorials enthalten eine Reihe von Selbst-Test Fragen.

Wir empfehlen, dass Sie die Selbst-Test Fragen bearbeiten. Sie sollen Ihnen die Gewissheit zu geben, dass Sie verstanden haben, was Sie in dem Tutorial gemacht haben. Es kann sein, dass derartige Fragen in der mündlichen Prüfung auftauchen.

Selbst-Test

- **Wurde das Compile und Link unseres Cobol Programms unter dem JES oger dem ISPF oder dem TSO Subsystem durchgeführt ?**
- **Wurde das Ausführen unseres Cobol Programms unter dem JES oger dem ISPF oder dem TSO Subsystem durchgeführt ?**

Wichtiger Hinweis:

Achten Sie darauf, dass Sie bei dem Befehl "tso call 'prakt20.test.load(cob02)'" die richtigen Hochkommas verwenden. Das korrekte Hochkomma steht auf den meisten Tastaturen über dem Zeichen "#".

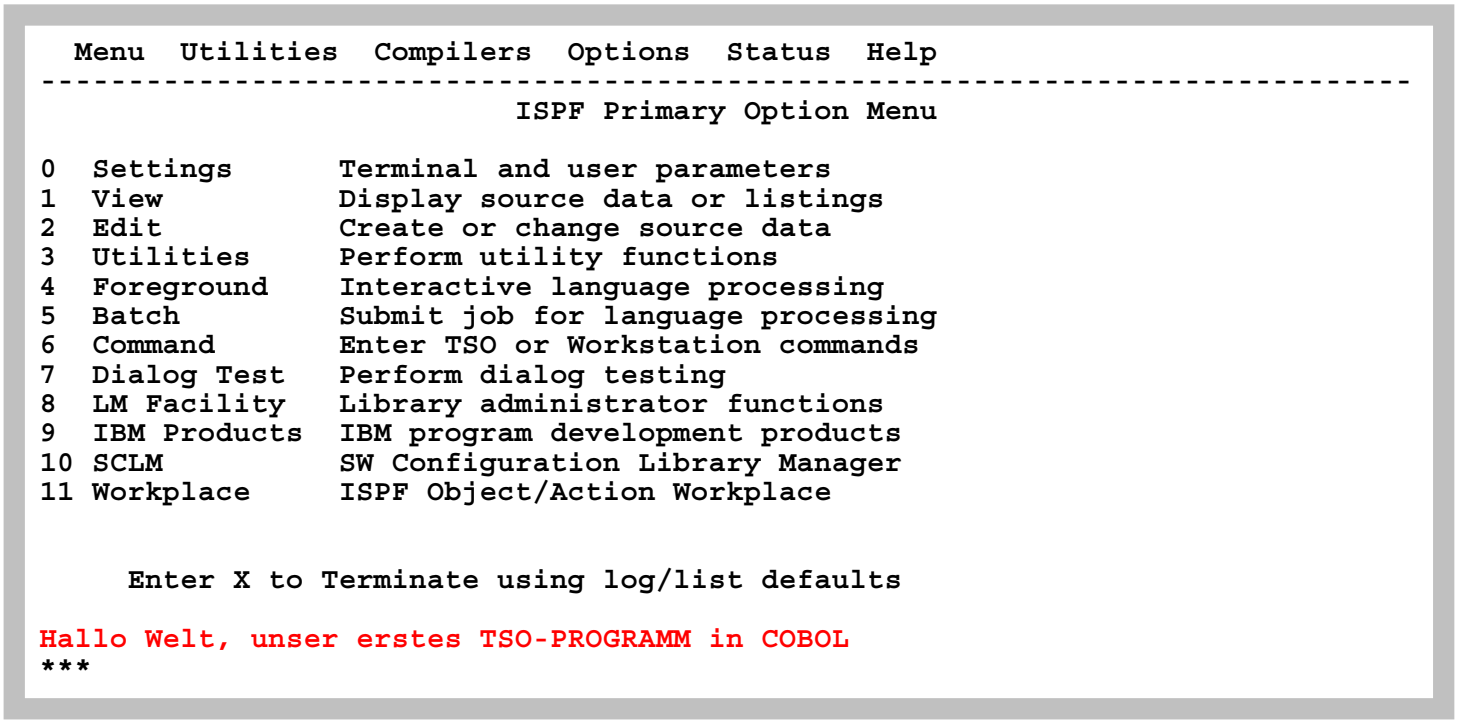


Abb. 5.2: Ausgabe unseres COBOL-Programms

Abb. 5.2 zeigt: Oberhalb der Kommandozeile erscheint die Ausgabe unseres COBOL-Programms. Wir hätten die Ausgabe etwas schöner gestalten können, indem wir `Hallo Welt, unser erstes TSO-PROGRAMM in COBOL` auf einem leeren Bildschirm schön zentriert in der Mitte ausgegeben hätten (fällt Ihnen eine primitive Lösung ein, wie man dies quick und dirty erreichen könnte ?).

Wir nehmen an, Ihnen fallen jetzt viele Möglichkeiten ein, ein aussagefähigeres COBOL-Programm zu schreiben. Sie können ein neues Quellprogramm PRAK085.TEST.COB(COBn) schreiben und hierfür ein neues JCL-Script PRAK085.TEST.CNTL(COBSTAn) erzeugen, was sich von PRAK085.TEST.CNTL(COB02) durch andere INFILE- und OUTFILE-Parameter unterscheidet. Dies resultiert in zusätzlichen Members in unseren drei Partitioned Datasets.

Aufgabe: ***Verfassen Sie ein eigenes funktionsfähiges COBOL-Programm (keine Modifikation des vorgegebenen Hallo-Welt-Programms) und legen Sie den Quellkode in PRAK085.TEST.COB(COBn) ab (n sei eine Ziffer Ihrer Wahl). Das angepasste JCL-Script legen Sie bitte in PRAK085.TEST.CNTL(COBSTAn) ab ("PRAK085" ist bei beiden Datasets durch Ihre Benutzer-ID zu ersetzen). Erstellen Sie je einen Print-Screen von Ihrem ISPF-Fenster mit dem Quellcode Ihres Programms sowie von Ihrem ISPF-Fenster mit der Ausgabe Ihres COBOL-Programms. Erzeugen Sie ebenfalls einen Print-Screen von dem ISPF-Fenster, der das von Ihnen modifizierte JCL-Script enthält. Schicken Sie die drei Print-Screens an Ihren Betreuer.***

Selbst-Test

- **Versuchen Sie doch, einmal ihr eigenes Cobol Programm zu schreiben. Sie finden reichlich Vorlagen hierfür im Internet, z.B. unter http://www.infogoal.com/cbd/cbdprj.htm**

Literatur zum Thema JCL:

Coding JCL Statements
http://www.informatik.uni-leipzig.de/cs/Literature/Textbooks/jcl/CodeJCL.pdf

M.Winkler: „MVS/ESA JCL“. Oldenbourg, 3. Auflage, 1999

z/OS JCL Reference SA22-7597-10 Eleventh Edition, April 2006
http://www.informatik.uni-leipzig.de/cs/Literature/Textbooks/jcl/MvsJclReference.pdf

z/OS JCL Users Guide
http://www.informatik.uni-leipzig.de/cs/Literature/Textbooks/jcl/MvsJclUserGuide.pdf

URLs

http://www.csis.ul.ie/cobol/

http://www.cobol-workshop.de/

http://www.mainframegurukul.com/tutorials/programming/cobol/cobol-tutorial.html

http://www.freeprogrammingresources.com/coboltutr.html

Unix
http://scis.acast.nova.edu/~jclevin/COURSE/COBOL/cobol.html

Enterprise
http://theamericanprogrammer.com/programming/manuals.cobol.shtml

System Display and Search Facility (SDSF)

Falls Sie die Meldung MAXCC=0 (oder MAXCC=4) in Abb. 4.10 erhalten haben, könnten Sie eigentlich diesen Abschnitt überspringen. Sie werden die Information in Abschnitt 6 dennoch früher oder später brauchen, deswegen arbeiten Sie ihn bitte durch. Deshalb

Aufgabe: ***Erstellen Sie die am Ende von Abschnitt 6 gewünschten Screenshots***

Die System Display and Search Facility (SDSF) ist eine Sammlung von Werkzeugen, welche bei der Fehlersuche behilflich sein können. Diese Sammlung ist sehr mächtig, aber nicht sehr intuitiv. Das Software Paket „Rational Developer for System z“ (RDz) bietet modernere Alternativen zu SDSF. Wir werden uns dies in einem späteren Tutorial ansehen.

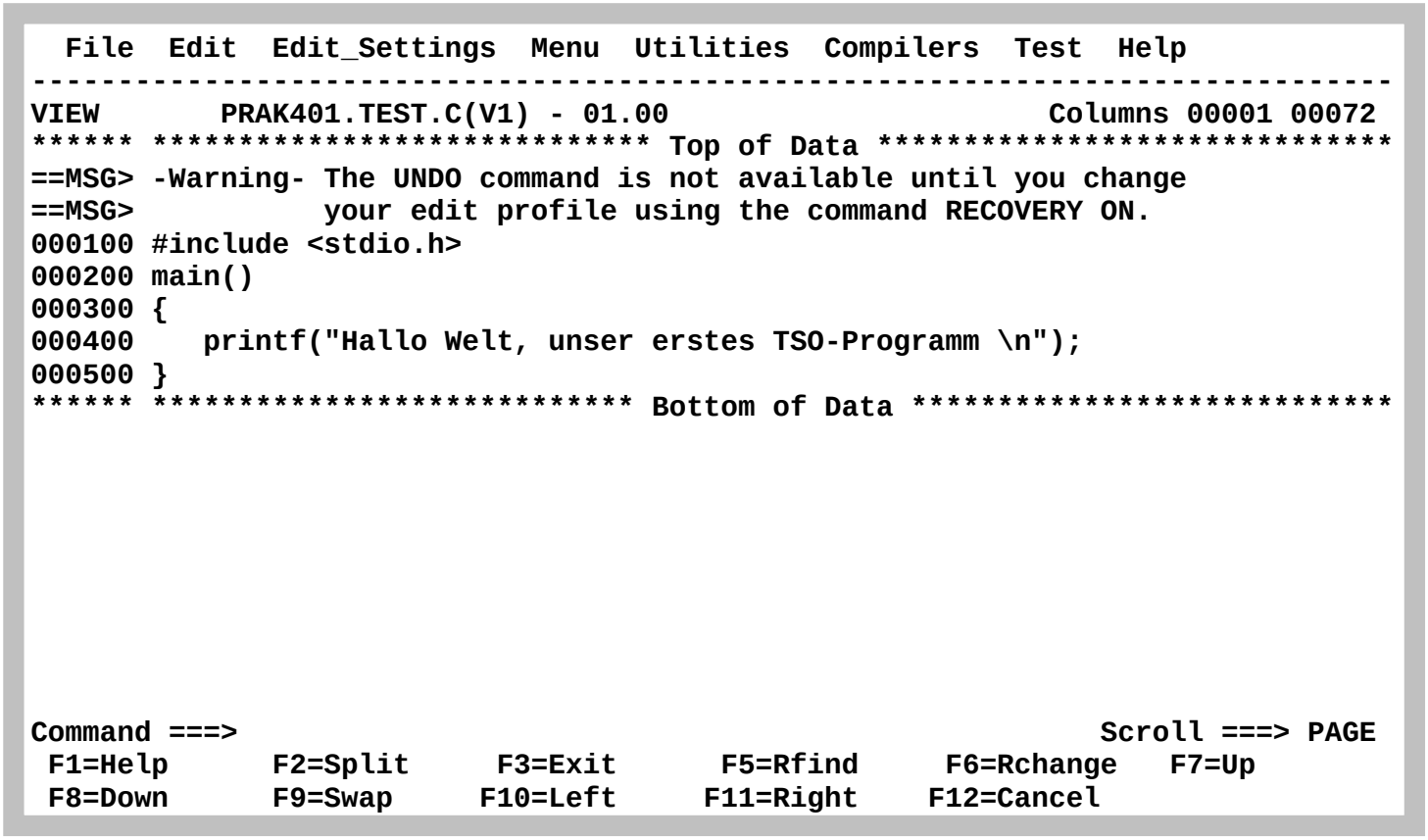


Abb. 6.1

Angenommen, wir haben dieses C Programm,

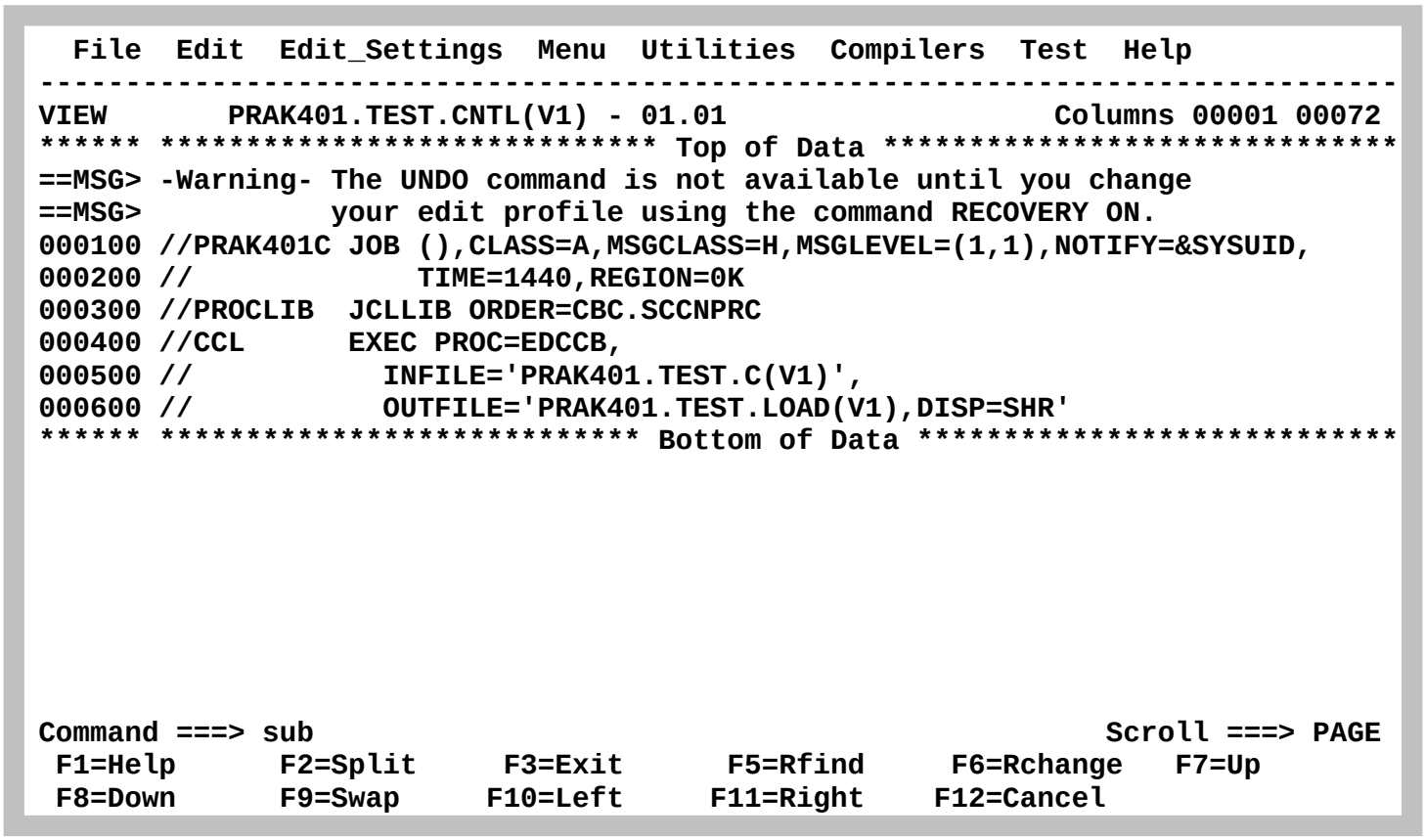


Abb. 6.2

und das dazugehörige JCL Script erzeugt. Wir geben das sub Kommando ein-

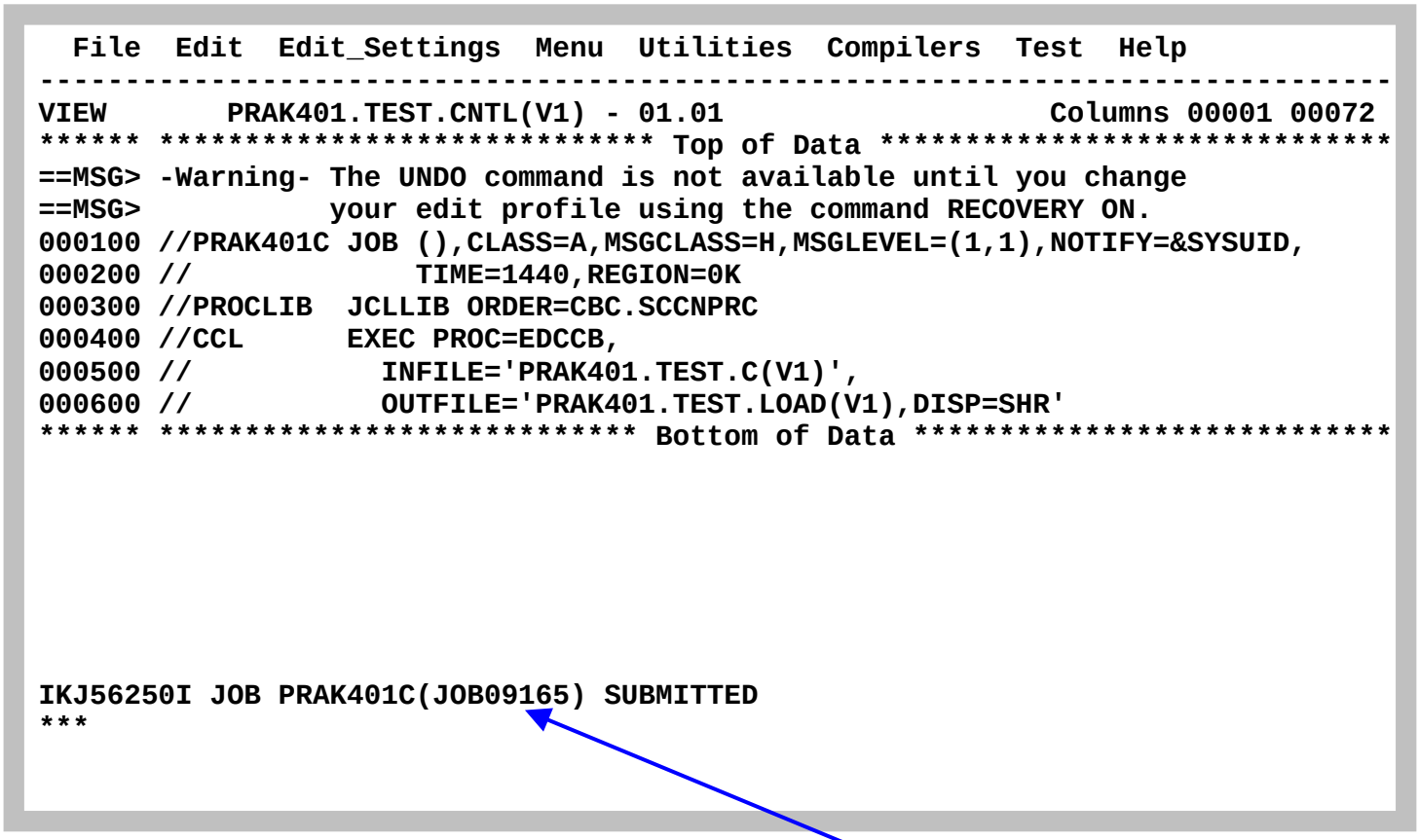


Abb. 6.3

JES hat gerade den Job akzeptiert, und wird ihn unter der Job Nummer 09165 ausführen. Diese Job Nummer notieren wir uns auf einem Stück Papier

```
09.18.07 JOB09165 $HASP165 PRAK401C ENDED AT N1  MAXCC=0 CN(INTERNAL)
 ***
```

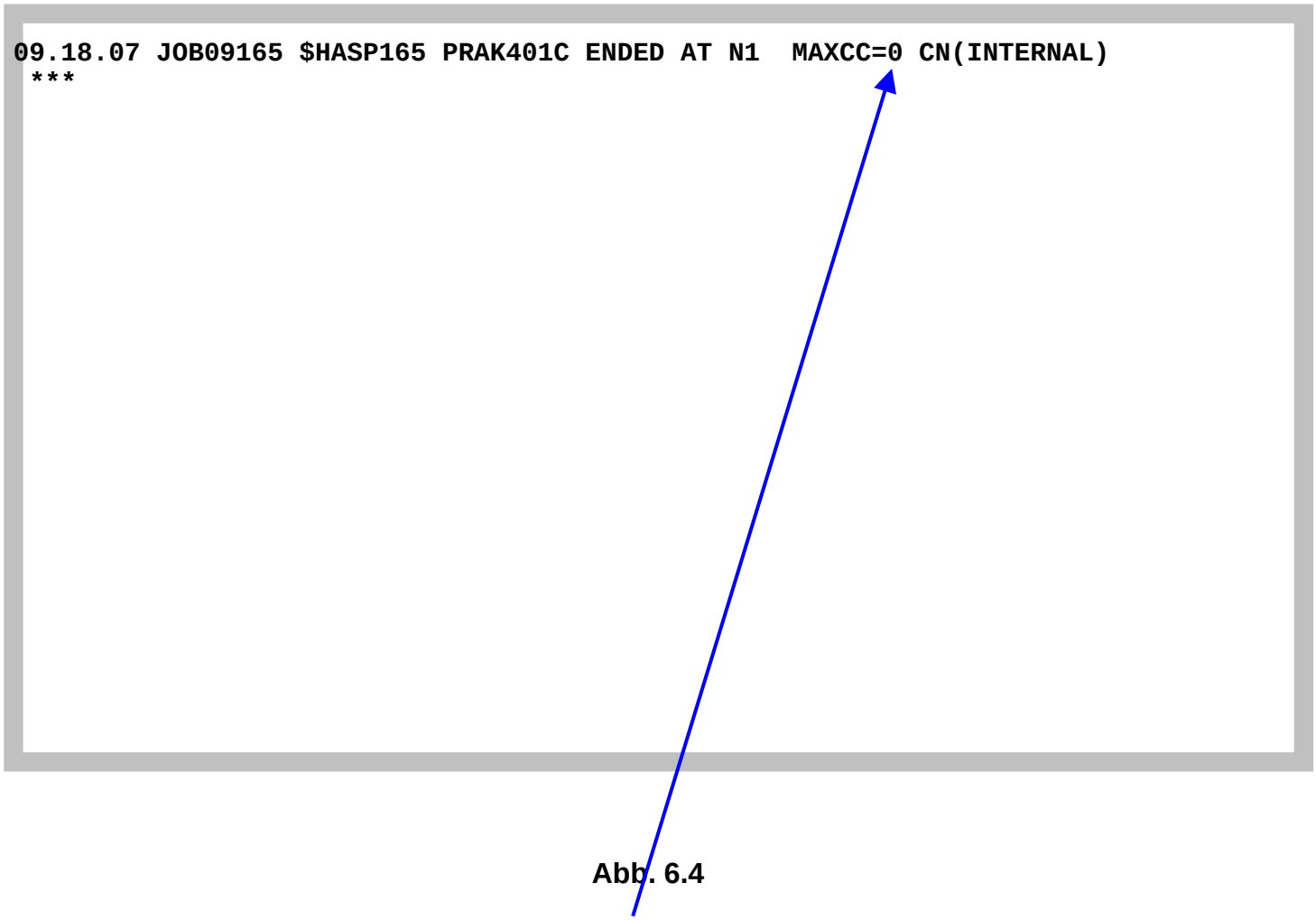


Abb. 6.4

Es sei angenommen, alles ist ok, deshalb MAXCC= 0

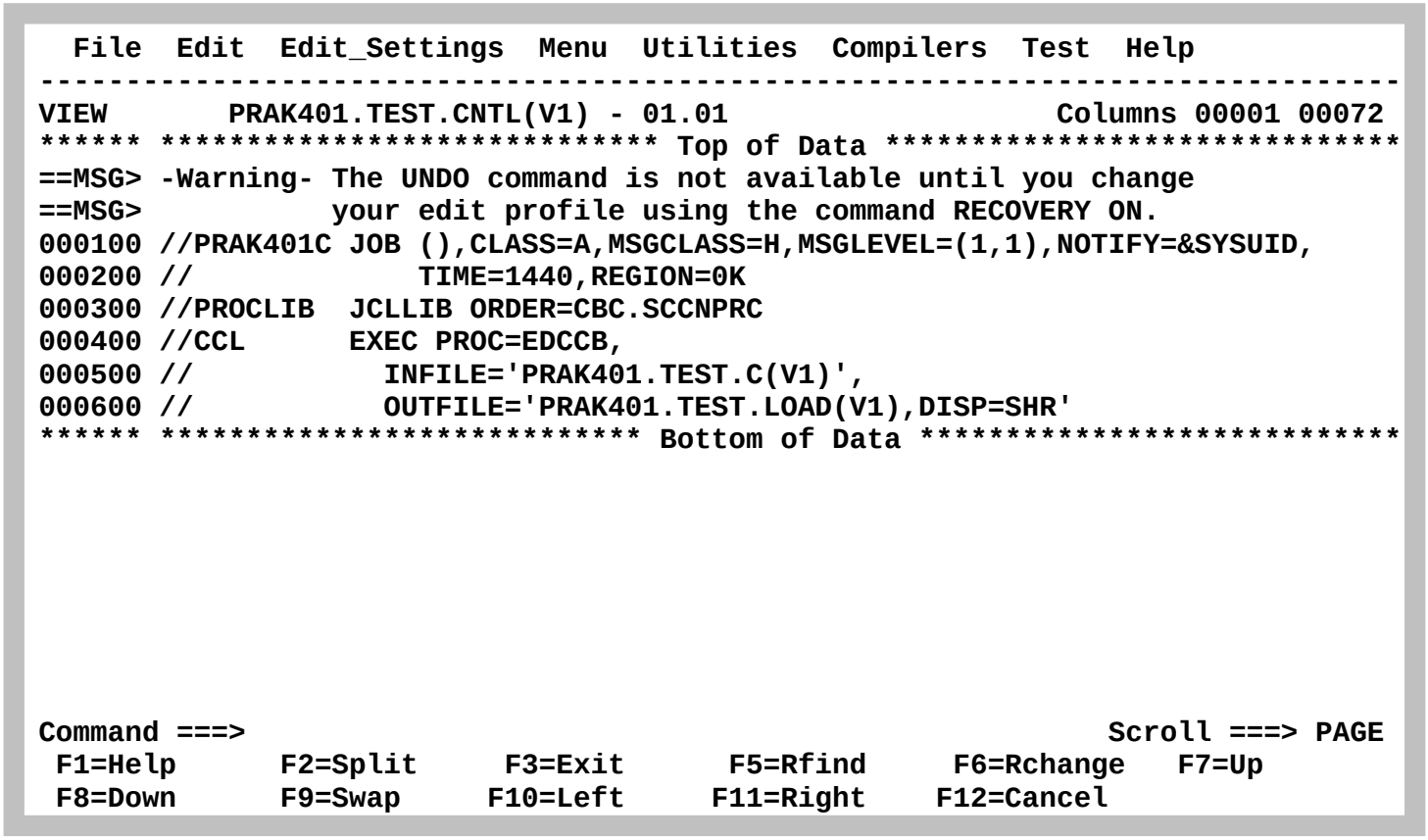


Abb. 6.5

Enter bringt uns zurück.

Jetzt 4 mal die F3 Taste betätigen, um zum ISPF Primary Option Menu zurück zu gelangen.

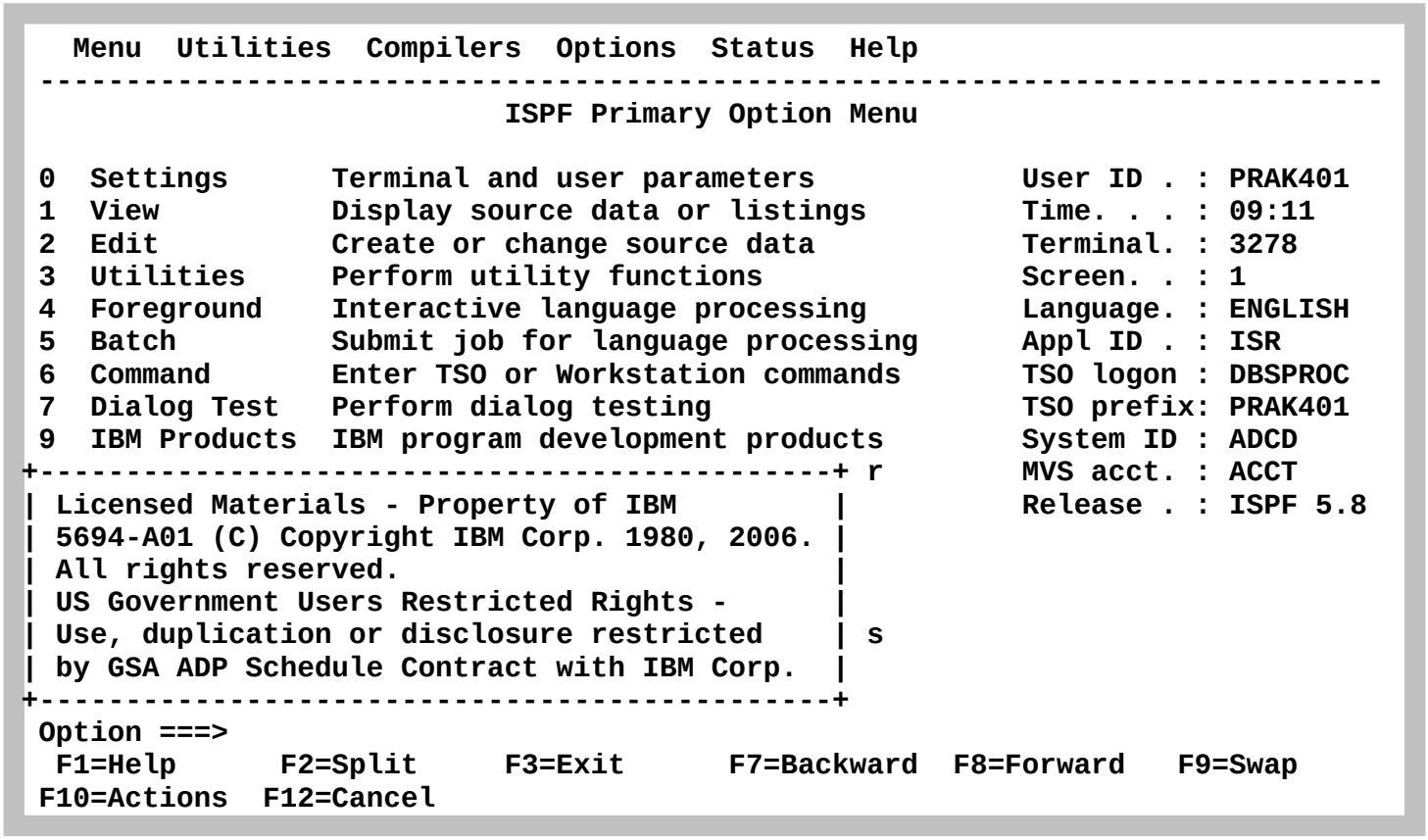


Abb. 6.6

Mittels der F8 Taste entfernen wir die uninteressante Meldung „Licensed Materials usw. .....“,

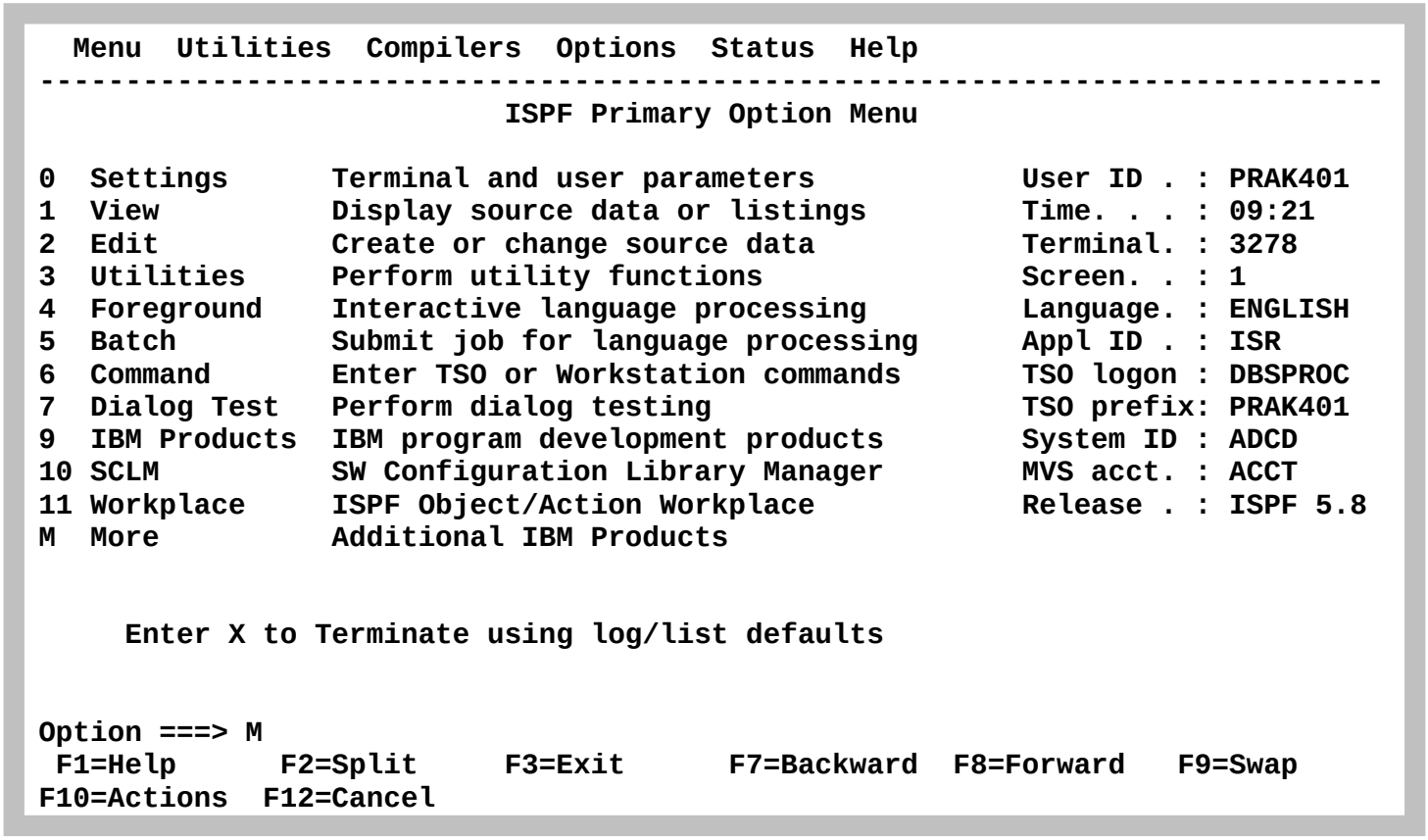


Abb. 6.7

und geben M auf der Command Line ein. Damit blättern wir zum Folgescreen (aus Platzgründen braucht das ISPF Primary Option Menue zwei Panels).

(Alternativ, Eingabe des TSO SDSF Commands von der Command line ===> ruft ebenfalls SDSF auf.)

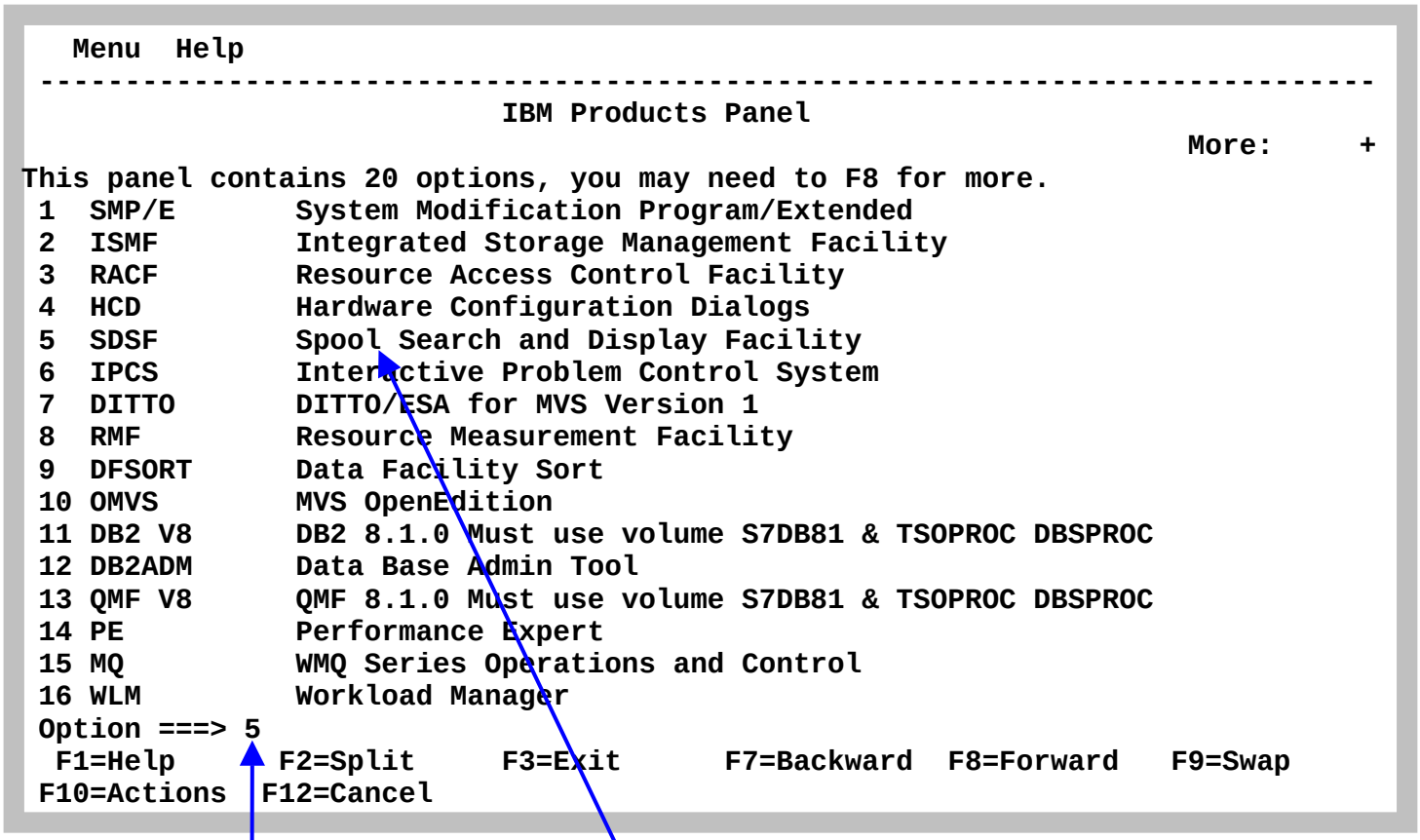


Abb. 6.8

Toll, was es unter ISPF da so alles an Möglichkeiten gibt. 3 RACF benutzen Sie z.B als Systemadministrator, um Sicherheitseinstellungen zu verwalten, oder eine neue User ID anzulegen. 16 WLM (Work Load Manager) ist eine hervorragende Einrichtung, um das Leistungsverhalten eines Rechners zu optimieren. Wir werden uns WLM später anschauen. Hier geben wir eine „5“ ein, um die „System Display and Search“ (SDSF) Facility aufzurufen.

JES speichert eine Reihe von Zwischenergebnissen in temporären Files ab. Ganz früher wurden diese Data Sets auf Magnetband geschrieben (Spool) um dann bei Bedarf offline ausgedruckt zu werden. Deshalb werden diese Files als Spool Files bezeichnet.

Die Spool Files enthalten auch Status Information über den fortschritt bei der Ausführung unseres Jobs. Das kann vor allem im Fehlerfall interessant sein.

Wir geben eine 5 ein, um das SDSF Subsystem aufzurufen, und ...

gelangen in das SDSF Primary Option Menu, einem ISPF Subsystem. Es bedeuten:

DA **Display Active zur Überwachung der Jobs die im System ausgeführt werden.**
I **Input Queue zur Überwachung der Jobs, die zur Ausführung bereit stehen.**
H **Held output queue, um die Ergebnisse der Job-Ausführung zu überprüfen**

```
  Display  Filter  View  Print  Options  Help
------------------------------------------------------------------------------
HQX7730 ----------------  SDSF PRIMARY OPTION MENU  -------------------------

DA    Active users
I     Input queue
O     Output queue
H     Held output queue
ST    Status of jobs

SE    Scheduling environments

END   Exit SDSF

Licensed Materials - Property of IBM

5694-A01 (C) Copyright IBM Corp. 1981, 2006. All rights reserved.
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.


COMMAND INPUT ===> ST                                    SCROLL ===> PAGE
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=IFIND     F6=BOOK
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Abb. 6.9

Wir geben ST auf der Kommandozeile ein, um uns nach dem Status unseres Jobs zu erkundigen.

Die ersten beiden Spalten werden als NP (Non Protected) bezeichnet. In diese Spalten kann man Kommandos eingeben, vor allem die „?“ und „S“ Kommandos.

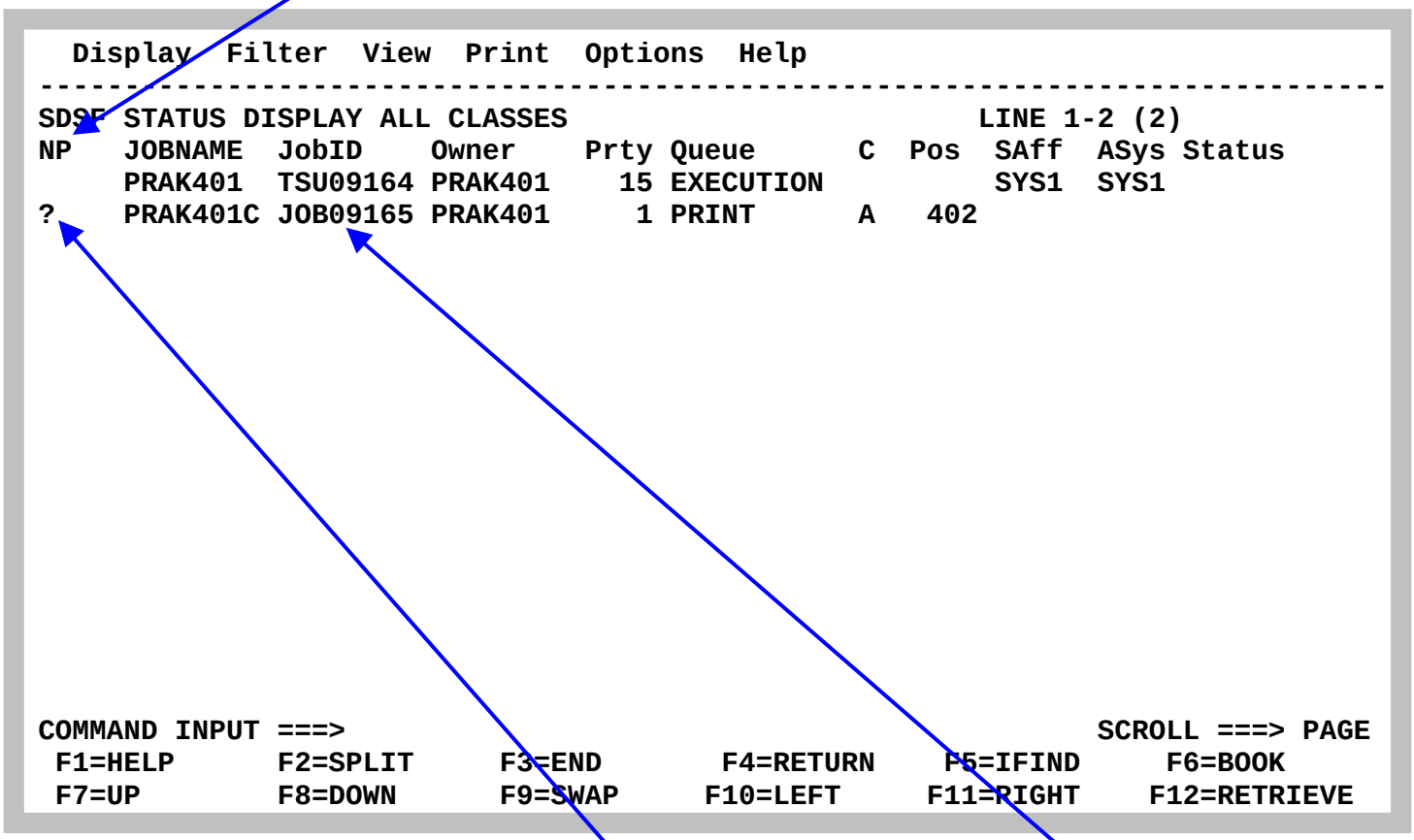


Abb. 6.10

In diesem Beispiel ist der Screen ungewöhnlich leer. Meistens befinden sich hier zahlreiche Einträge für unterschiedliche Jobs. Aber wir finden hier den Eintrag für unseren Job 09165. Wir geben ein Fragezeichen (?) in diese Zeile in der NP Spalte ein, um nachzuschauen.

Dies bewirkt, dass die bei der Ausführung unseres Jobs angelegten Spool Files angezeigt werden.

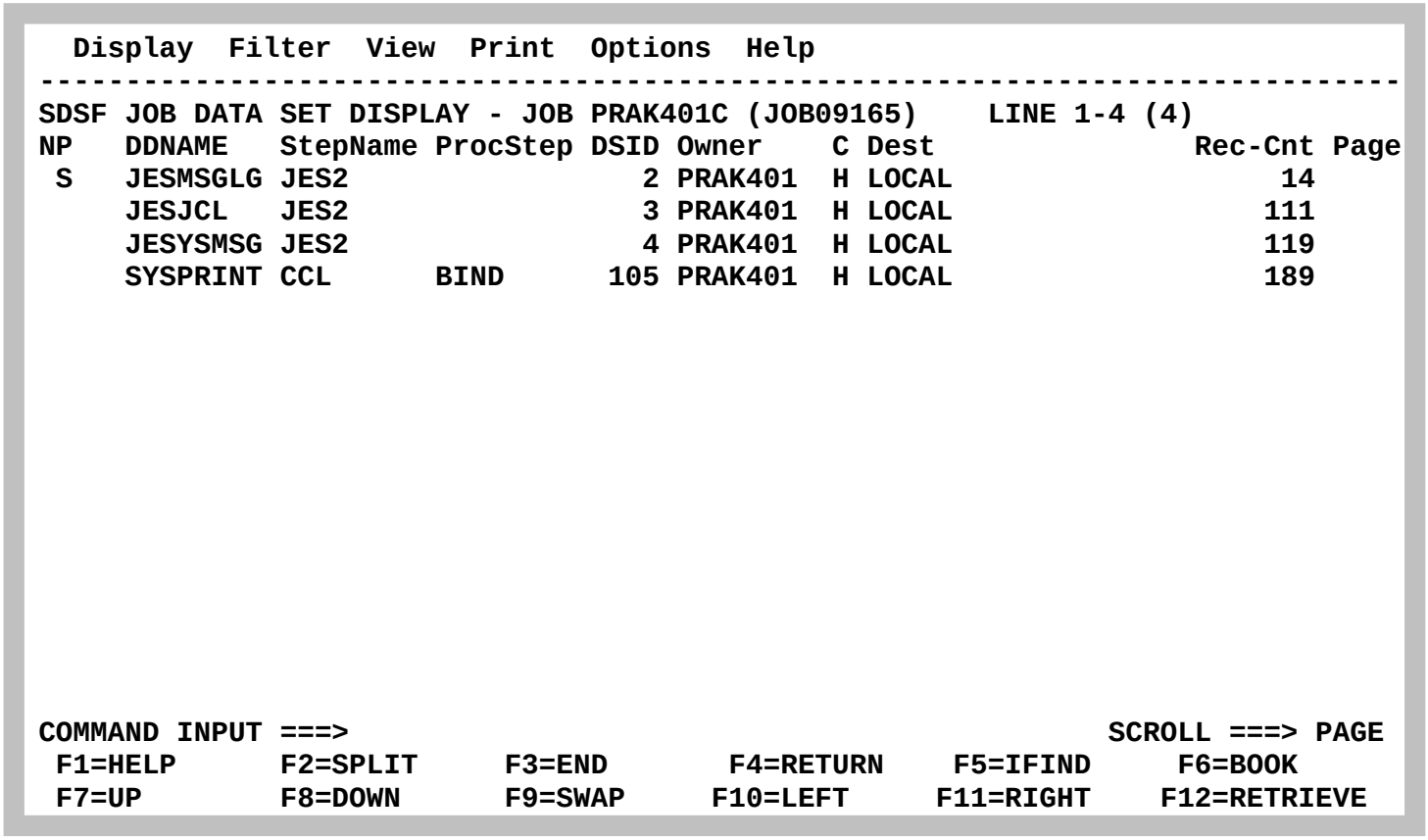


Abb. 6.11

Wiedergegeben werden drei Data Set Namen (dd names): JES2 messages log file, JES2 JCL file, und JES2 system Messages File unseres Jobs 09165.

JES notiert alle Aktionen für alle Jobs in einer Log File JESMSGLG. Ein S in der NP Spalte zeigt jeweils den Inhalt einer der Files an.

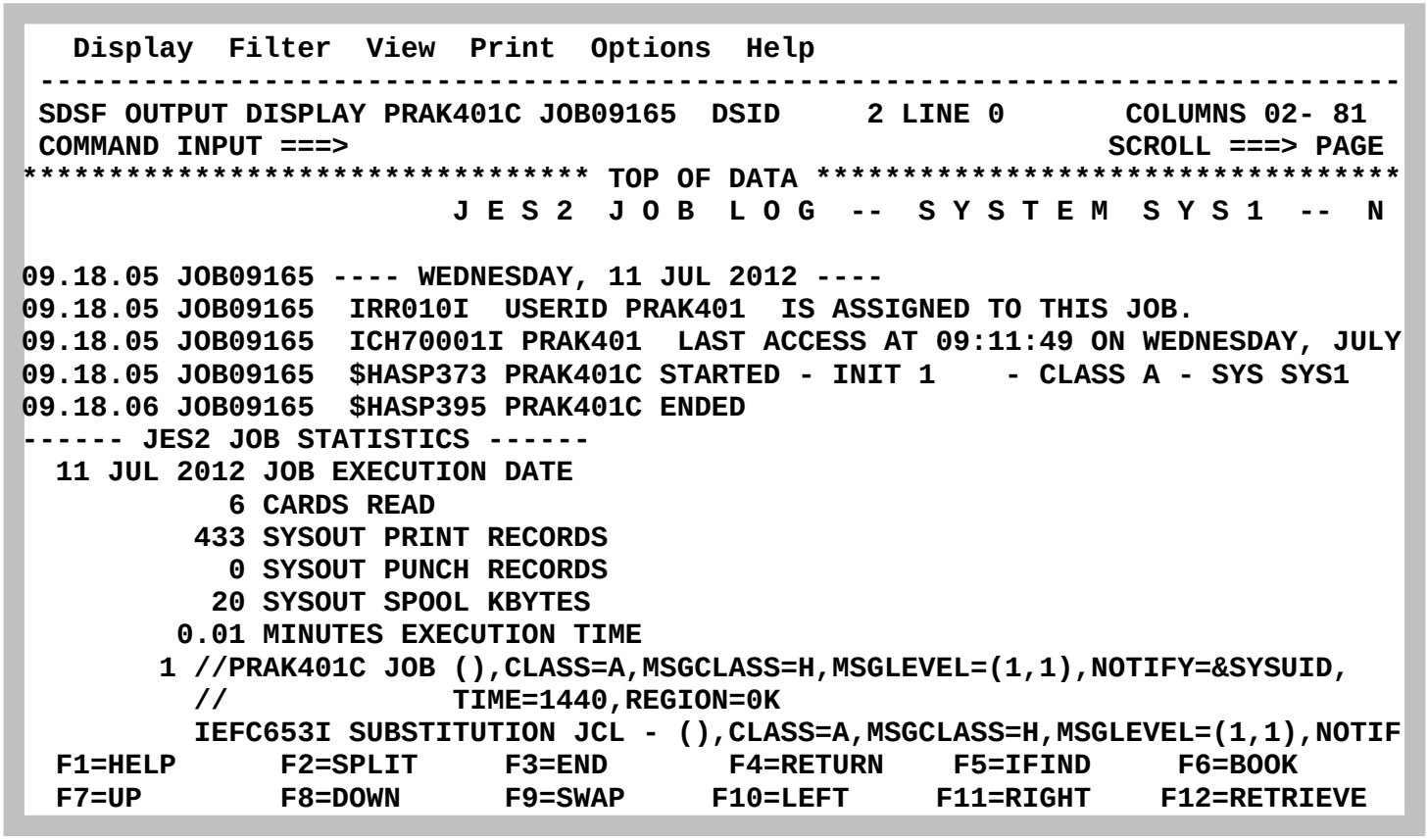


Abb. 6.12

Dies ist der erste von vielen Screens, die man sich mit F8 und F7 blättern und ansehen kann.

Mit F3 Zurück zum vorherigen Panel.

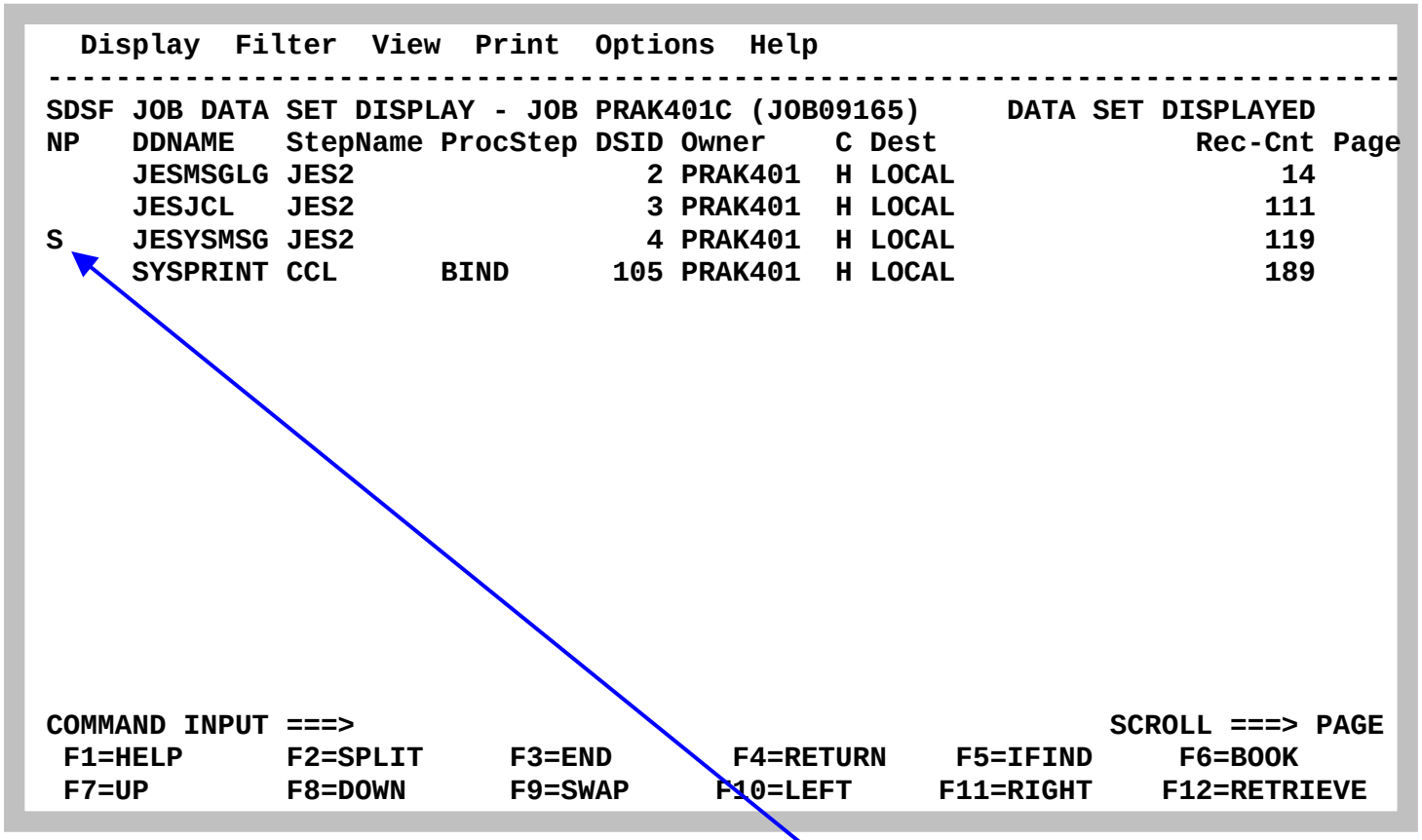


Abb. 6.13

Schauen wir uns den Inhalt von JESYSMSG an, in dem wir ein „S“ eingeben.

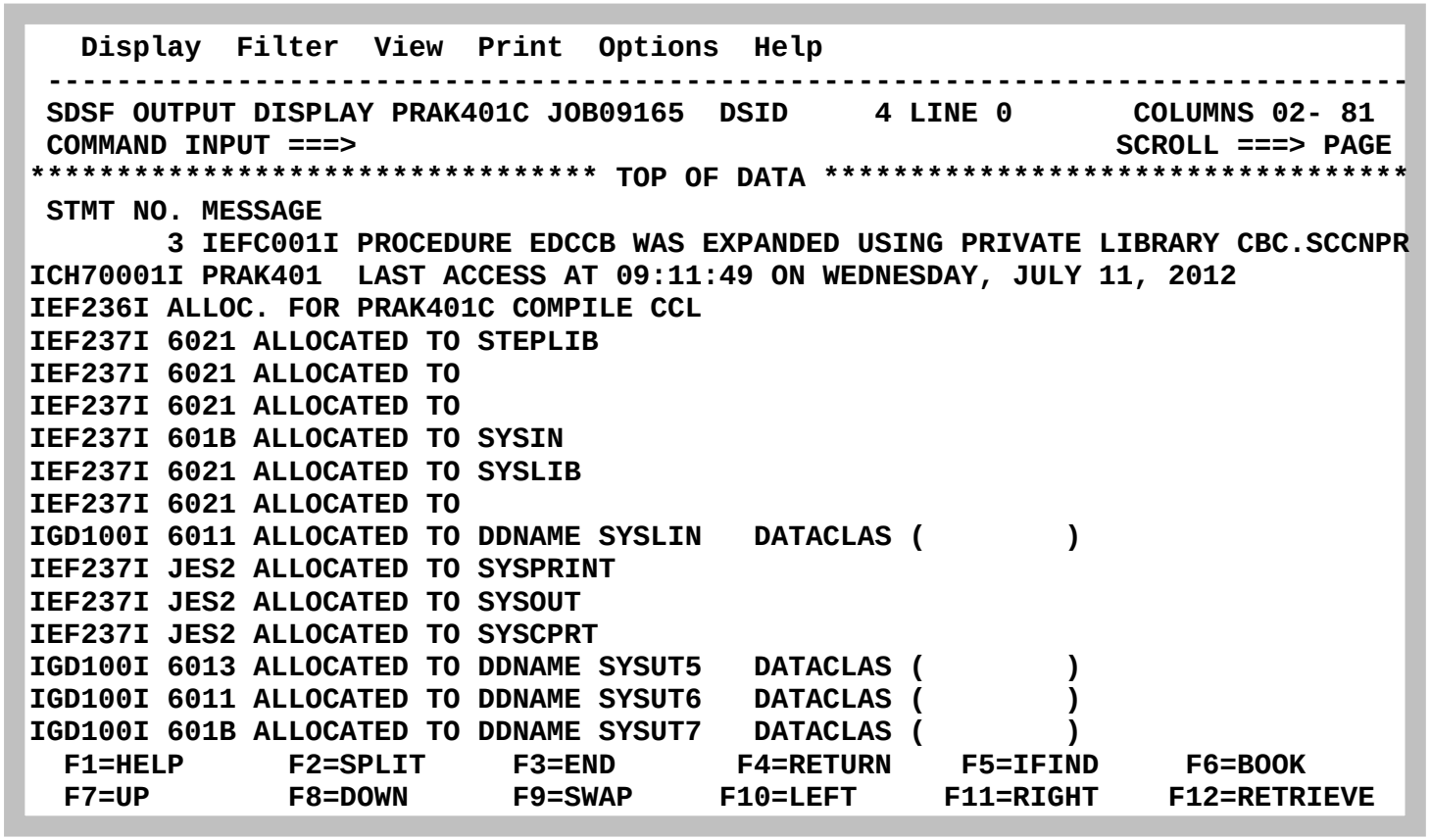


Abb. 6.14

Das JESYSMSG Listing enthält Memory Allocation und Cleanup Messages. ALLOC sagt, welche Devices und wieviel Hauptspeicherplatz für den Job Step allocated wurde. Sie erinnern sich, ein Job Step führt ein Programm aus. Es informiert ebenfalls über die von dem Job Step verbrauchte CPU Zeit.

Mit F3 zurück zum SDSF Status Display all Classes Panel.

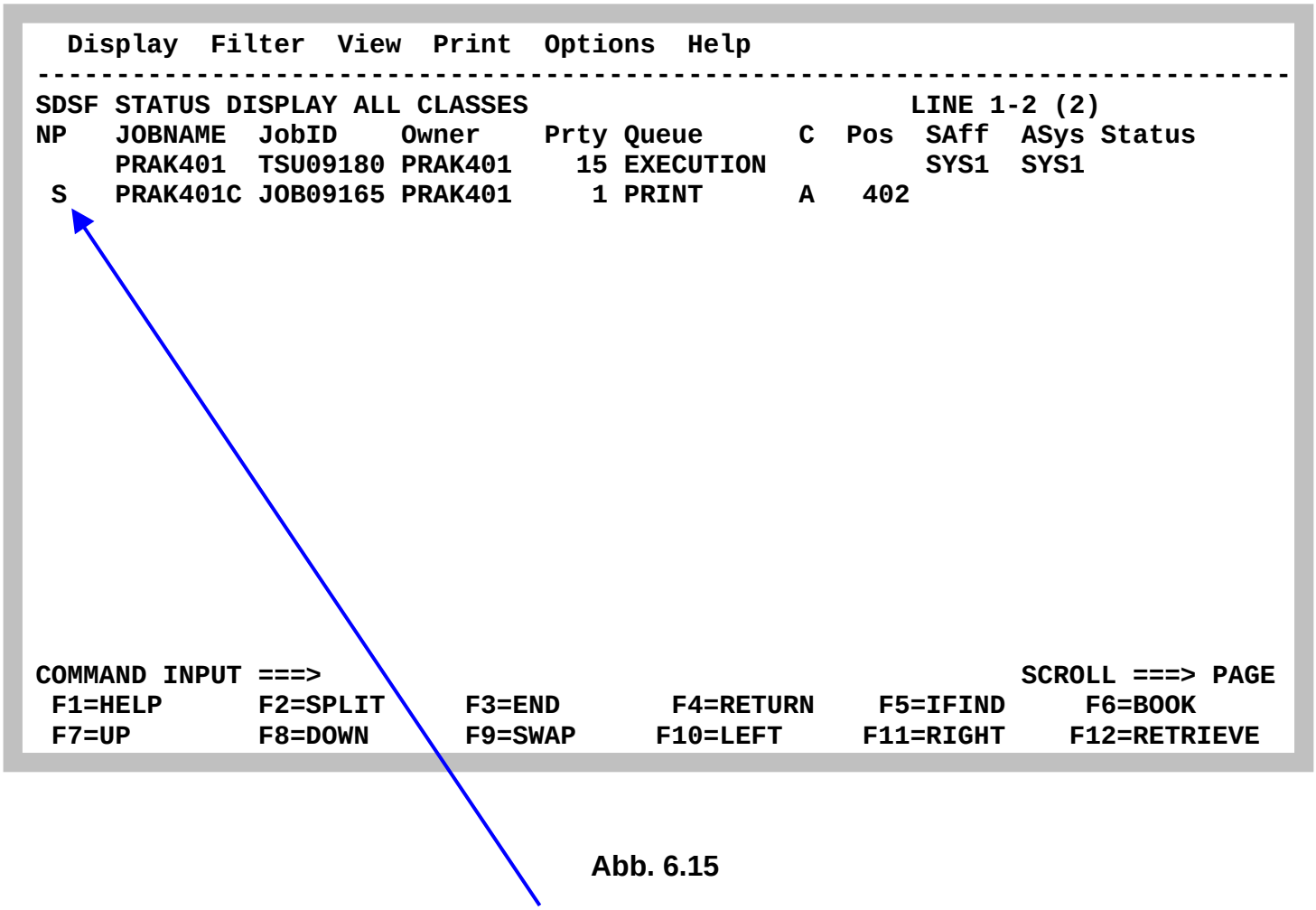


Abb. 6.15

Um eine Zusammenfassung aller Log Files zu sehen, die bei der Ausführung von Job 09165 erzeugt wurden, geben wir ein “S” an Stelle des Fragezeichens (?) in der NP Spalte ein.

```
  Display  Filter  View  Print  Options  Help
 ------------------------------------------------------------------------------
 SDSF OUTPUT DISPLAY PRAK401C JOB09165  DSID     2 LINE 0       COLUMNS 02- 81
 COMMAND INPUT ===>                                           SCROLL ===> PAGE
********************************* TOP OF DATA ***********************************
                        J E S 2  J O B  L O G  --  S Y S T E M  S Y S 1  --  N

09.18.05 JOB09165 ---- WEDNESDAY, 11 JUL 2012 ----
09.18.05 JOB09165  IRR010I  USERID PRAK401  IS ASSIGNED TO THIS JOB.
09.18.05 JOB09165  ICH70001I PRAK401  LAST ACCESS AT 09:11:49 ON WEDNESDAY, JULY
09.18.05 JOB09165  $HASP373 PRAK401C STARTED - INIT 1    - CLASS A - SYS SYS1
09.18.06 JOB09165  $HASP395 PRAK401C ENDED
------ JES2 JOB STATISTICS ------
  11 JUL 2012 JOB EXECUTION DATE
           6 CARDS READ
         433 SYSOUT PRINT RECORDS
           0 SYSOUT PUNCH RECORDS
          20 SYSOUT SPOOL KBYTES
        0.01 MINUTES EXECUTION TIME
       1 //PRAK401C JOB (),CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1),NOTIFY=&SYSUID,
         //             TIME=1440,REGION=0K
         IEFC653I SUBSTITUTION JCL - (),CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1),NOTIF
  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=IFIND     F6=BOOK
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Abb. 6.16

Dies gibt eine Zusammenfassung der Files von Job09165 wieder. Der Screen Inhalt ist mit Abb. 6.12 identisch.

Dies ist der erste von vielen Screens, die man sich mit F8 und F7 blättern und ansehen kann. Interessant sind die letzten Screens

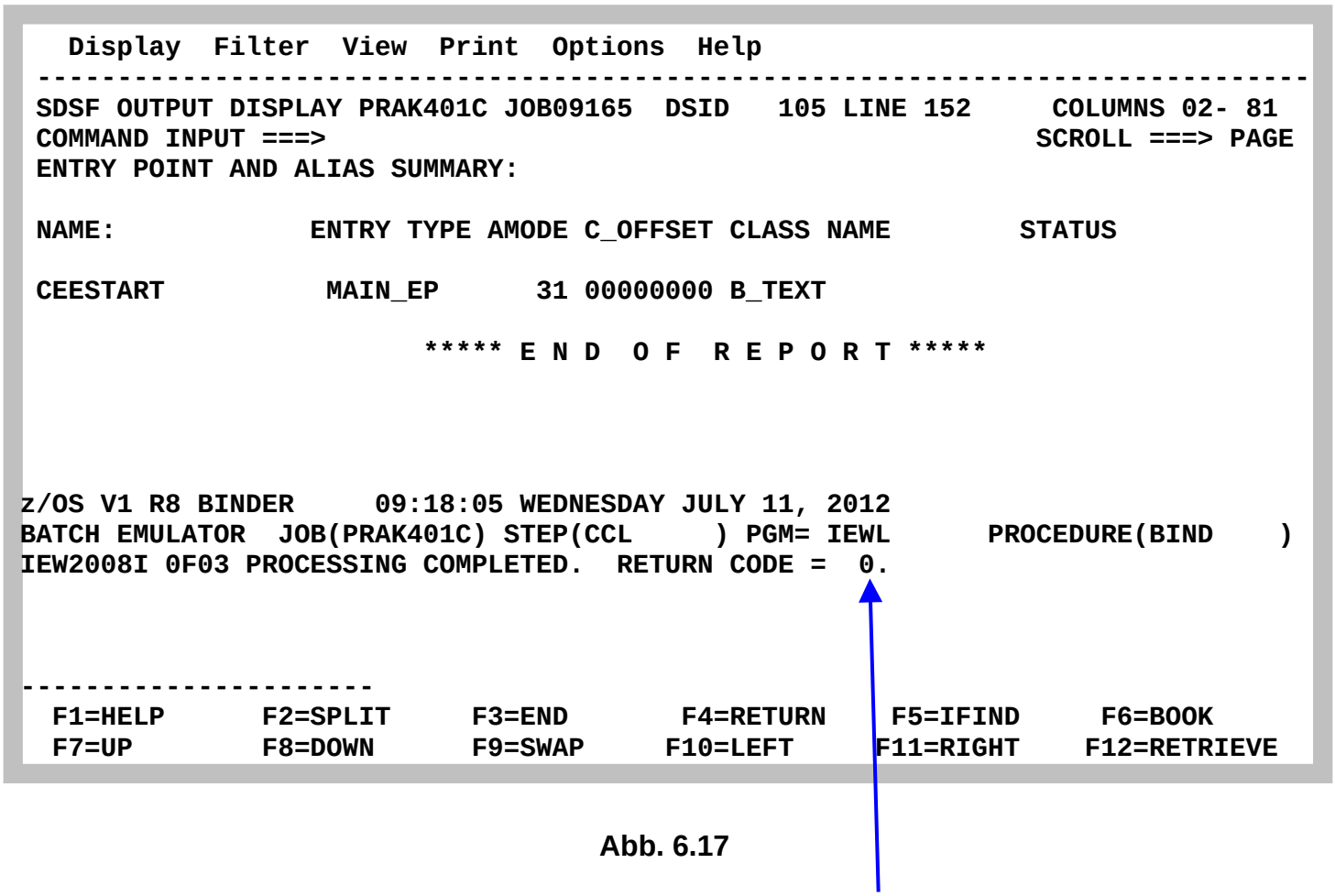


Abb. 6.17

Hier wird bestätigt, dass der Return Code (MAXCC) den Wert 0 hat.

```
  Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
SDSF OUTPUT DISPLAY PRAK401C JOB09165  DSID   105 LINE 170     COLUMNS 02- 81
COMMAND INPUT ===>                                            SCROLL ===> PAGE
MESSAGE SUMMARY REPORT
----------------------
TERMINAL MESSAGES      (SEVERITY = 16)
NONE

SEVERE MESSAGES        (SEVERITY = 12)
NONE

ERROR MESSAGES         (SEVERITY = 08)
NONE

WARNING MESSAGES       (SEVERITY = 04)
NONE

INFORMATIONAL MESSAGES (SEVERITY = 00)
2008  2278


 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=IFIND     F6=BOOK
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Abb. 6.18

Das darauf folgende Panel beagt, die Fehler Messages sind alle ok

Im Fehlerfall würden die hier wiedergegebenen Files Tips und Anmerkungen zu dem aufgetretenen Fehler enthalten. Je nach Fehlerursache kann das sehr unterschiedlich sein.

<u>Aufgabe:</u> Bauen Sie in Ihr Cobol Programm oder in ihr JCL Script einen Fehler ein, und zeigen Sie in einem Screenshot wie SDSF hierfür Hilfe leistet.

Literatur zum Thema SDSF

ABCs of z/OS System Programming Volume 1, Abschnitt 4.5.1, S. 173 ff.
http://www.informatik.uni-leipzig.de/cs/Literature/Textbooks/ispf/ABCs01.pdf

Chip Wood: SDSF for New Users.
http://www.cedix.de/VorlesMirror/Band3/sdsf03.pdf

Ebenfalls Information über SDSF

http://www.cedix.de/VorlesMirror/Band3/sdsf01.pdf

http://www.cedix.de/VorlesMirror/Band3/sdsf02.pdf

Anhang A : Beispiel für ein weiteres Cobol Programm

Source Code

```
000100 ID DIVISION.
000200 PROGRAM-ID.  ACCEPT1.
000300 DATA DIVISION.
000400 WORKING-STORAGE  SECTION.
000500 01  WS-FIRST-NUMBER      PIC 9(3).
000600 01  WS-SECOND-NUMBER     PIC 9(3).
000700 01  WS-TOTAL             PIC ZZZ9.
000800*
000900 PROCEDURE DIVISION.
001000 0000-MAINLINE.
001100     DISPLAY 'ENTER A NUMBER: '.
001200     ACCEPT WS-FIRST-NUMBER.
001300*
001400     DISPLAY 'ANOTHER NUMBER: '.
001500     ACCEPT WS-SECOND-NUMBER.
001600*
001700     COMPUTE WS-TOTAL = WS-FIRST-NUMBER + WS-SECOND-
NUMBER.
001800     DISPLAY 'THE TOTAL IS: ', WS-TOTAL.
001900     STOP RUN.
```

Sample Run

```
ENTER A NUMBER: 7
ANOTHER NUMBER: 7
THE TOTAL IS: 14
```

Abb. A.1 Beispiel für ein weiteres Cobol Programm

Anhang B: Cobol Fixed Format -Spaltenabhängigkeit

In fixed format, the COBOL source record is divided into the following areas:

Columns 1 - 6	**Sequence number**
Column 7	**Indicator area**
Column 8 - 11	**Area A**
Columns 12 - 72	**Area B**

COBOL source record can extend up to a maximum of 80 columns in length. The contents of columns 73 to 80 are ignored by the COBOL system.

Sequence Number

A sequence number of six digits can be used to identify each source program line.

If the first character position of the sequence number field contains an asterisk, or any nonprinting control character (less than the character SPACE in the ASCII collating sequence), then the line is treated as comment and is not output to the listing file or device. This facility allows an output listing file to be used as a source file to a subsequent compilation.

Indicator Area

An asterisk (*) in the indicator area marks the line as documentary comment only. Such a comment line can appear anywhere in the program after the IDENTIFICATION DIVISION header. Any characters from the ASCII character set can be included in area A and area B of the line.

A comment line can appear before the IDENTIFICATION DIVISION header.

A "D" or "d" in the indicator area represents a debugging line. Areas A and B can contain any valid COBOL sentence.

A "–" in the indicator area represents a continuation of the previous line without spaces or the continuation of a nonnumeric literal (see the section Continuation of Lines in the chapter Language Fundamentals).

A "$" in the indicator area indicates a special line for setting directives or conditional compilation.

Areas A and B

Section names and paragraph names begin in area A and are followed by a period and a space.

Anhang C: Frequently asked Questions

What happens if you:

1. **capitalize IDENTIFICATION DIVISION? Nothing… cobol is not case sensitive**

2. **delete the period after identification division? Compiler error… line2: Warning: '.' expected, PROGRAM-ID found**

3. **put an * in column 7 on the first statement (identification division)? Nothing… weird, huh?!**

4. **change the program-id from prelab to sample? Nothing… Remember that the program-id does not have to match the file name. I'll point out in a later question where this is used.**

5. **put the program-id name, prelab. (don't forget to include the period) on the next line starting in column 10? Nothing…**

6. **put the program-id name, prelab. (don't forget to include the period) on the next line starting in column 12? Nothing…**

7. **move data division to start in column 11? Nothing…**

8. **move data division to start in column 12? Compiler error… line12: Warning: Must start in Area A**

9. **delete the period on the line before the data division i.e. after the second organization is line sequential? Compiler error… line 12: warning: Area A should be blank, line 12: numeric literal expected, DIVISION found, line 14: syntax error**

10. **comment out the 3 statements environment division, input-output section and file-control by putting an * in column 7 of each statement? Nothing… weird, huh?!**

11. **comment out the file section statement? 6 compiler errors with the 7th one saying "Can't recover, good bye!"**

NOTE: Delete the * on the accept omitted statement (before the stop run statement) for the next question only.

12. **in the file section, change the pic x(20) on 05 employee-name-in to pic x(18)? Run error occurs 5 times… non-numeric data in numeric field at 000083 in PRELAB. The 000083 is a memory location which will probably change each time the program is run. The value is a hex value (binary = base 2, octal = base 8, decimal = base 10, hexadecimal = base 16). The PRELAB is the program-id variable name! The change also causes not just a run error, but a logic error as well! Notice that the output file has incorrect data…**

13. **change the 77 level number in the working-storage section to an 01 level number? nothing**

14. **change the non-numeric literal on the value clause for the are-there-more-records variable to "no"? a logic error occurs where in the output file, the page and column headers as well as the underline appear, but no data**

15. in the file section, put a $ in front of the 9.99 on the pic clause for rate-out (i.e. pic $9.99)? a $ appears on the data under the $/hr column; notice the earnings data was moved over one position

16. delete the period on the weekly-wages-out pic clause (i.e. pic 99999 instead of 999.99)? in the output file, leading zeroes and no decimal values appear on the data

17. delete the period before the working-storage section statement? Compiler errors… line 30: Warning: Area A should be blank; line 30: syntax error scanning WORKING-STORAGE; line 31: illegal level number

18. in the file section, change the variable name hours-worked-in to hrs-wrk-in? compiler error - line 65: undefined data item: HOURS-WORKED-IN. Notice error on use of variable name, not declaration.

19. in the file section, change the variable name weekly-wages-out to wkly-wgs-out? Compiler error - Line 69: same as above with WEEKLY-WAGES-OUT

20. change the variable name employee-data on the read statement to employee-record? Compiler error - Line 55 or 71: no SELECT for file: EMPLOYEE-RECORD

21. change the variable name hours-out in 200-wage-routine paragraph to hrs-out? Compiler error – line 66: undefined data item: HRS-OUT

22. change 05 levels on employee-record in the file section to 01 levels? Compiler error – line 17: warning: must start in Area A, line 17: warning: no space in item: EMPLOYEE –RECORD i.e. no pic clause; same two errors for line 18

23. change the double quotes to single quotes on the select statement? nothing

24. change the double quotes to single quotes for all the "no" non-numeric literals? nothing

25. change "no" to "NO" on the perform statement (in paragraph 100-main-module)? Run error… box hard to see… press enter, maximize the window, and re-run the program. File error 46 on prelabin.dat. COBOL error at 00008C in prelab <OK>. Also, notice the output file has extra/duplicate last data line. Why is this an error?! Because non-numeric literal are case sensitive!!! i.e. "no" is not the same as "NO".

26. comment out the 100-main-module paragraph name? compiler error line 46: identifier expected, OPEN found; line 46: missing PARA/SECTION; line 46: Can't recover, good-bye!

27. comment out the 1st line in the 100-main-module paragraph (open input employee-data)? Compiler error – line 47: verb expected, OUTPUT found. What if comment out the entire OPEN statement i.e. open line and next line as well?! You get a run error… box says file error 48.02 on prelabout.dat. COBOL error at 000010 in prelab <OK>.

28. delete all the periods under the 100-main-module paragraph? Compiler error – line 62: warning: missing period.

29. delete all the periods under the 100-main-module paragraph except on the stop run statement? Logic error! Output file has up to the first data line under page and column headers with underline.

30. correct the change you made to the above question and forget to save before recompiling? You get the same error since prelab.cob was not actually updated; just the screen version was updated, not the stored version.

31. change the line before the read statement to after advancing 0 lines (instead of 1 lines)? It looks like the column headers are deleted. Actually, this underlines the column headers but since the screen cannot put two characters in one location/space, the dashes overwrote the column header info. If you print the output, you will see the underlined column headers without advancing the print head one line (to be discussed in more detail when talk about the WRITE statement).

32. change the line before the read statement to after advancing 1 line (deleting the s off the word lines)? nothing

33. comment out the close statement? nothing

34. put employee-data on the close statement on a different line, starting column 12 or after? nothing

35. comment out the stop run statement? Same as #27! Why?! Because without a stop run, the program continues to run statements in sequential order including over the paragraph name and to the first statement in the paragraph, etc. and you can't write to a file that you have already closed!

Other:

36. Notice that the 3rd line and the 11th line are blank; there is no * in the 7th column to denote a comment. Why is this okay?! COBOL ignores "white space"!

37. Notice that the non-numeric literal "Employee Name" has a length of 13, but the pic clause has a length of 23. Why is this okay? Because the data is stored to the left so allows for open positions to the right.

38. Notice that there are no variable names on the 05 levels under both page-header and column-headers. Why is this okay? Because I don't need to access those memory locations

39. List the first 20 statements executed by the program. Open, write, write, write, read, perform, move, move, move, move, multiply, write, read, perform, move, move, move, move, multiply, write

40. Delete the * on the accept omitted statement. Add the following display statement between the first read statement and the perform statement. Also add the following display statement at the end of the 2nd paragraph (i.e. the last sequential statement for the program). What output do you see on the screen? Employee record is (data line information for each record) except the last (i.e.5th) record is repeated, so there are 6 output lines to the screen

What happens if you delete the period after the second read statement? Only two lines of output appear; one for the first record, one for the last record.

http://www.cse.ohio-state.edu/~reeves/CSE314/Labs/Lab1PartBHWSolutions.htm

see also http://www.cse.ohio-state.edu/~reeves/CSE314/Labs/

Anhang D : Coding JCL Statements

Identifier field
The identifier field indicates to the system that a statement is a JCL statement rather than data. The identifier field consists of the following:

- **Columns 1 and 2 of all JCL statements, except the delimiter statement, contain // v Columns 1 and 2 of the delimiter statement contain either /* or two other characters designated in a DLM parameter to be the delimiter**
- **Columns 1, 2, and 3 of a JCL comment statement contain //***

Name field
The name field identifies a particular statement so that other statements and the system can refer to it. For JCL statements, code the name as follows:

- **The name must begin in column 3. v The name is 1 through 8 alphanumeric or national ($, #, @) characters. See Table 4-2 on page 4-3 for the character sets.**
- **The first character must be an alphabetic or national ($, #, @).**
- **The name must be followed by at least one blank.**

Operation field
The operation field specifies the type of statement, or, for the command statement, the command. Code the operation field as follows:

- **The operation field consists of the characters in the syntax box for the statement. v The operation follows the name field.**
- **The operation must be preceded and followed by at least one blank.**

Parameter, or operand field
The parameter field, also sometimes referred to as the operand field, contains parameters separated by commas. Code the parameter field as follows:

- **The parameter field follows the operation field.**
- **The parameter field must be preceded by at least one blank.**

Comments field
The comments field contains any information you deem helpful when you code the control statement. Code the comments field as follows:

- **The comments field follows the parameter field.**
- **The comments field must be preceded by at least one blank.**

Code the identifier field beginning in column 1 and the name field immediately after the identifier, with no intervening blanks. Code the operation, parameter, and comments fields in free form. Free form means that the fields need not begin in a particular column. Between fields leave at least one blank; the blank serves as the delimiter between fields.

Exceeding 71 columns
Do not code fields, except on the comment statement, past column 71. If the total length of the fields would exceed 71 columns, continue the fields onto one or more following statements

Continuing the Parameter Field

1. **Interrupt the field after a complete parameter or subparameter, including the comma that follows it, at or before column 71.**
2. **Code // in columns 1 and 2 of the following statement.**
3. **Code a blank character in column 3 of the following statement. If column 3 contains anything but a blank or an asterisk, the system assumes the following statement is a new statement. The system issues an error message indicating that no continuation is found and fails the job.**
4. **Continue the interrupted parameter or field beginning in any column from 4 through 16.**

z/OSMVS JCL Reference SA22-7597-10 Eleventh Edition, April 2006 , chapter 3, http://www-01.ibm.com/support/docview.wss?uid=pub1sa22759712

Anhang E : Cobol Key Words

#+Here is a list of COBOL keywords that can start a line and end with the . character:

ACCEPT
ACQUIRE
ADD
ALTER
CALL
CANCEL
CLOSE
COMMIT
COMPUTE
DELETE
DISPLAY
DIVIDE
DROP
EXIT
EXIT PROGRAM
GO TO
GOBACK
INITIALIZE
INSPECT
MERGE
MOVE
MULTIPLY
OPEN
PERFORM
READ
RELEASE
RETURN
REWRITE
ROLLBACK
SET
SORT
START
STOP
STOP RUN
STRING
SUBTRACT
UNSTRING
WRITE

Some of these statements also have an implicit terminator:

END-ACCEPT
END-ADD
END-CALL
END-COMPUTE
END-DELETE
END-DIVIDE
END-EVALUATE
END-IF
END-MULTIPLY
END-PERFORM
END-READ
END-RETURN
END-REWRITE
END-SEARCH
END-START
END-STRING
END-SUBTRACT
END-UNSTRING
END-WRITE

REXX Scripting Language

PHP, Python, PHP, Perl und Tcl/Tk sind weit verbreitete Scripting Languages. Unter z/OS existiert vor allem das sehr populäre JCL, was aber im Vergleich zu den erwähnten Scripting Languages den Eindruck erweckt, auf einem anderen Planeten erfunden worden zu sein. Neben JCL existieren unter z/OS „normale“ Scripting Languages:

- **PHP (ein regulärer Port von PHP Version 5.1.2),**
- **CLIST (verfügbar nur unter z/OS und mit schrumpfender Bedeutung) sowie**
- **REXX (Restructured Extended Executor).**

REXX wurde Anfang der 1980er Jahre von Mike Cowlishaw entwickelt, und ist eine von IBM bevorzugte Scripting Language. REXX ist auf alle IBM Betriebssystemen seit vielen Jahren verfügbar. Dies gilt auch für nicht-Mainframe Betriebssysteme wie AIX und i5/OS. Es bestehen Ähnlichkeiten mit Pearl; wie Pearl kann REXX gut für CGI Programmieraufgaben eingesetzt werden. REXX kann und wird an Stelle von Cobol oder Java für die Programmierung von CICS Anwendungen eingesetzt.

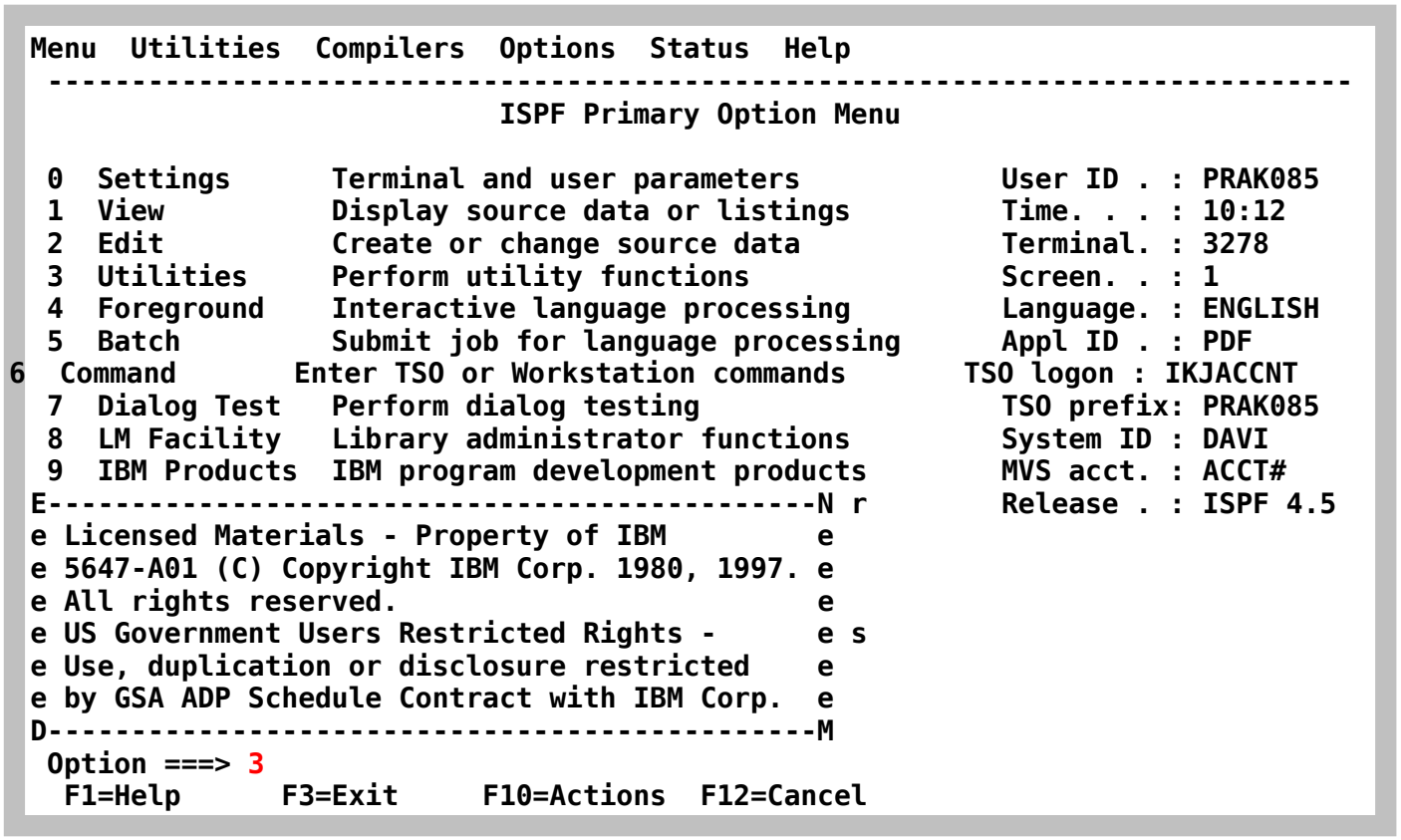
Eine ganze Reihe von open source/public domain REXX Versionen, sowie kommerzielle Versions existieren für praktisch alle bestehenden Betriebssysteme einschließlich Android, Windows CE, Symbian Epoc und Palm OS. Beispiele für Open Source Versionen sind der Regina REXX Interpreter (http://regina-rexx.sourceforge.net/) oder Open Object Rexx (http://www.oorexx.org/) . Microsoft hat seit Jahren einen REXX Interpreter mit ihren SDKs ausgeliefert.

The American National Standard (ANSI Standard) for Rexx is called "Programming Language - REXX", and its number is X3.274-1996.

The Rexx Language Association (RexxLA) is an independent organization dedicated to promoting the use and understanding of the Rexx programming language. http://www.rexxla.org/.

Das folgende Tutorial beschreibt die Erzeugung und Nutzung eines einfachen Hallo Welt REXX Programms unter TSO.

Aufgabe: *Beschäftigen Sie sich mit diesem Tutorial. Schreiben Sie, so wie hier beschrieben, die beiden REXX-Programme; also dieses, das nur "Hallo Welt" ausgibt und jenes, das zwei Zahlen dividiert.*

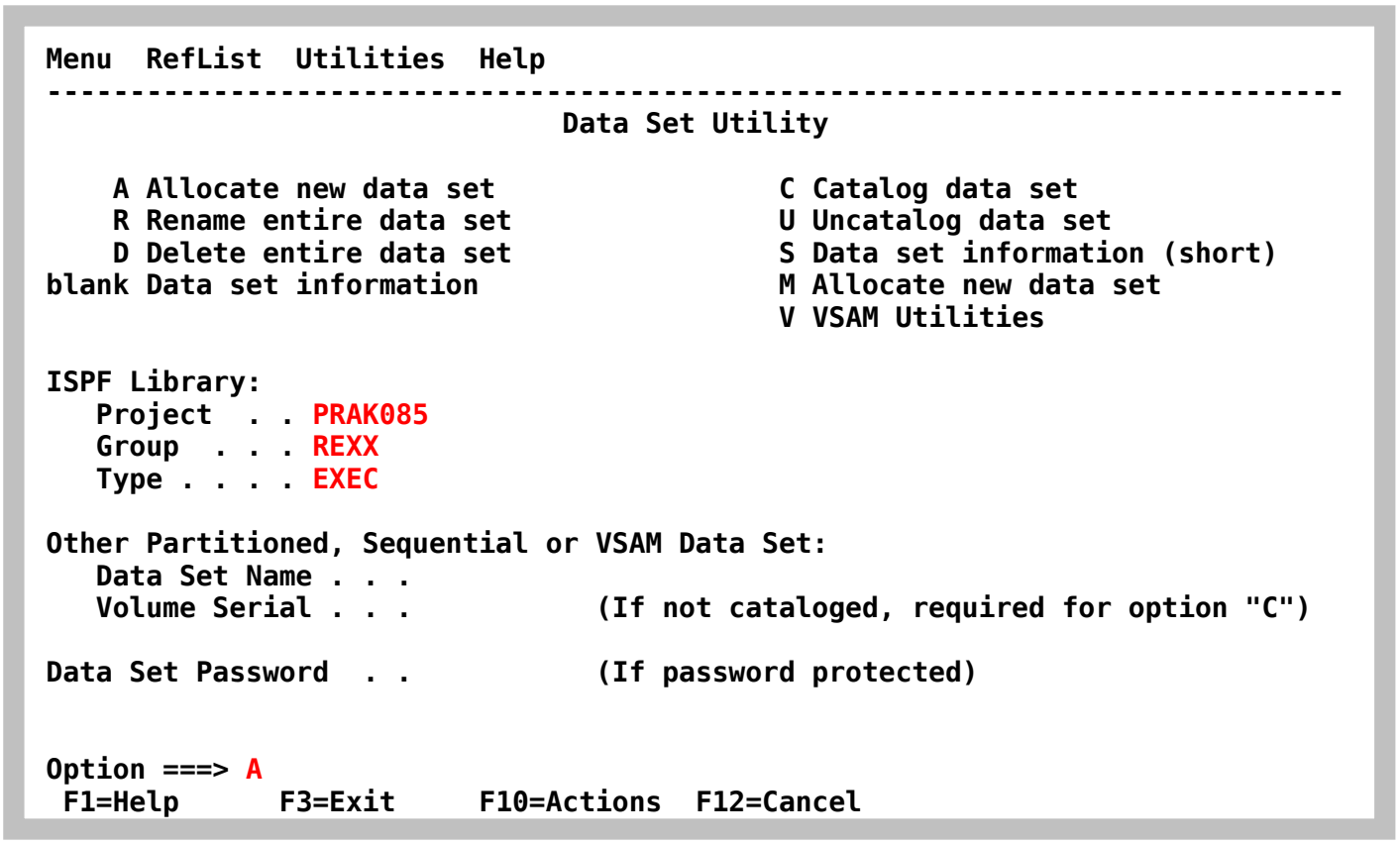


Vom ISPF Primary Option Menue aus einen Partitioned Data Set für die Aufnahme von REXX Scripten anlegen

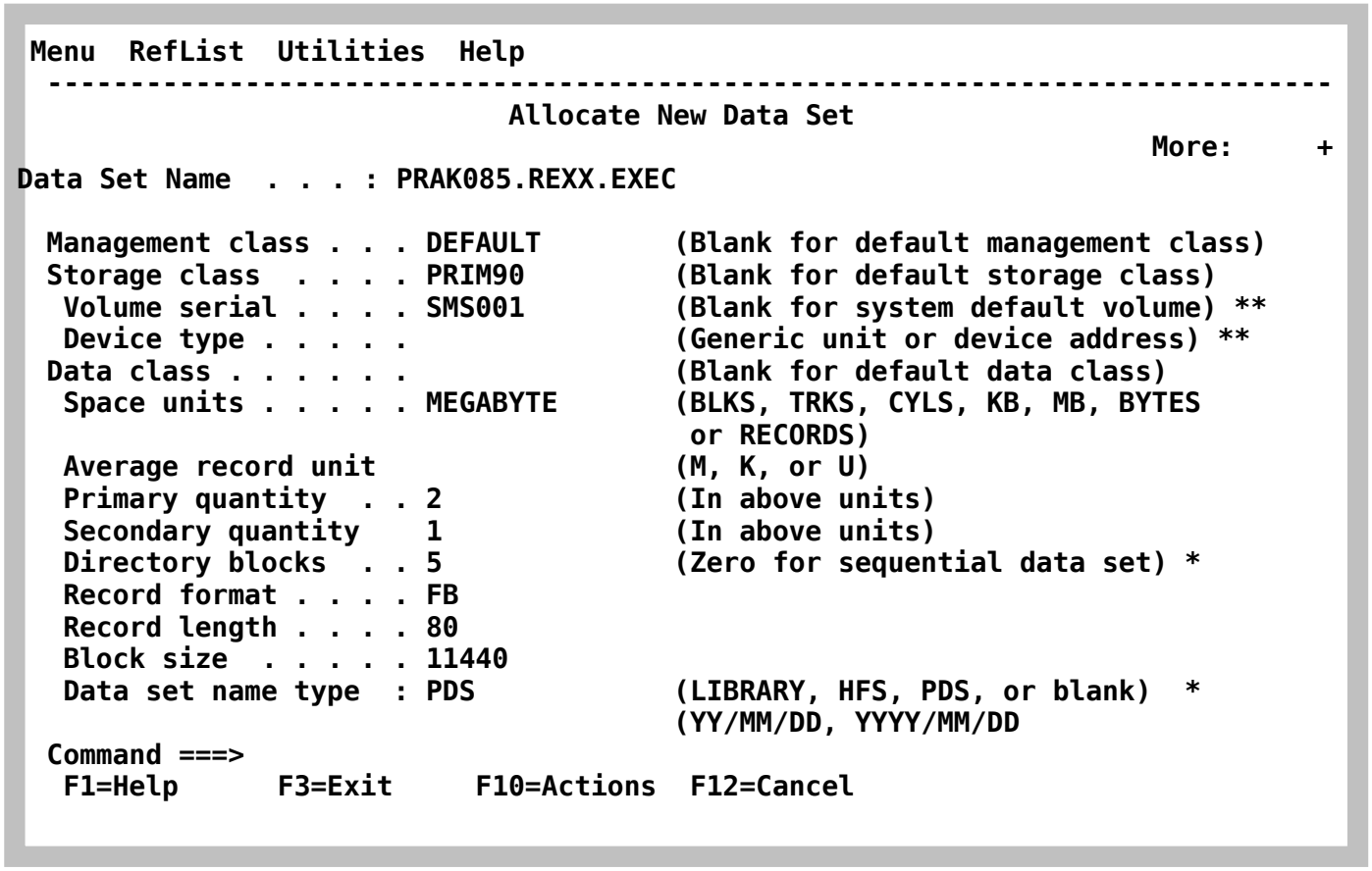
Enter

```
Menu  Help
 ------------------------------------------------------------------------------
                            Utility Selection Panel

 1  Library     Compress or print data set.  Print index listing.  Print,
                  rename, delete, browse, edit or view members
 2  Data Set    Allocate, rename, delete, catalog, uncatalog, or display
                  information of an entire data set
 3  Move/Copy   Move, copy, or promote members or data sets
 4  Dslist      Print or display (to process) list of data set names.
                  Print or display VTOC information
 5  Reset       Reset statistics for members of ISPF library
 6  Hardcopy    Initiate hardcopy output
 7  Download    Download ISPF C/S, VA for ISPF, transfer map, or data set.
 8  Outlist     Display, delete, or print held job output
 9  Commands    Create/change an application command table
 *  Reserved    This option reserved for future expansion.
 11 Format      Format definition for formatted data Edit/Browse
 12 SuperC      Compare data sets                              (Standard Dialog)
 13 SuperCE     Compare data sets Extended                     (Extended Dialog)
 14 Search-For  Search data sets for strings of data           (Standard Dialog)
 15 Search-ForE Search data sets for strings of data Extended (Extended Dialog)
 Option ===> 2
  F1=Help      F3=Exit     F10=Actions  F12=Cancel
```

Enter

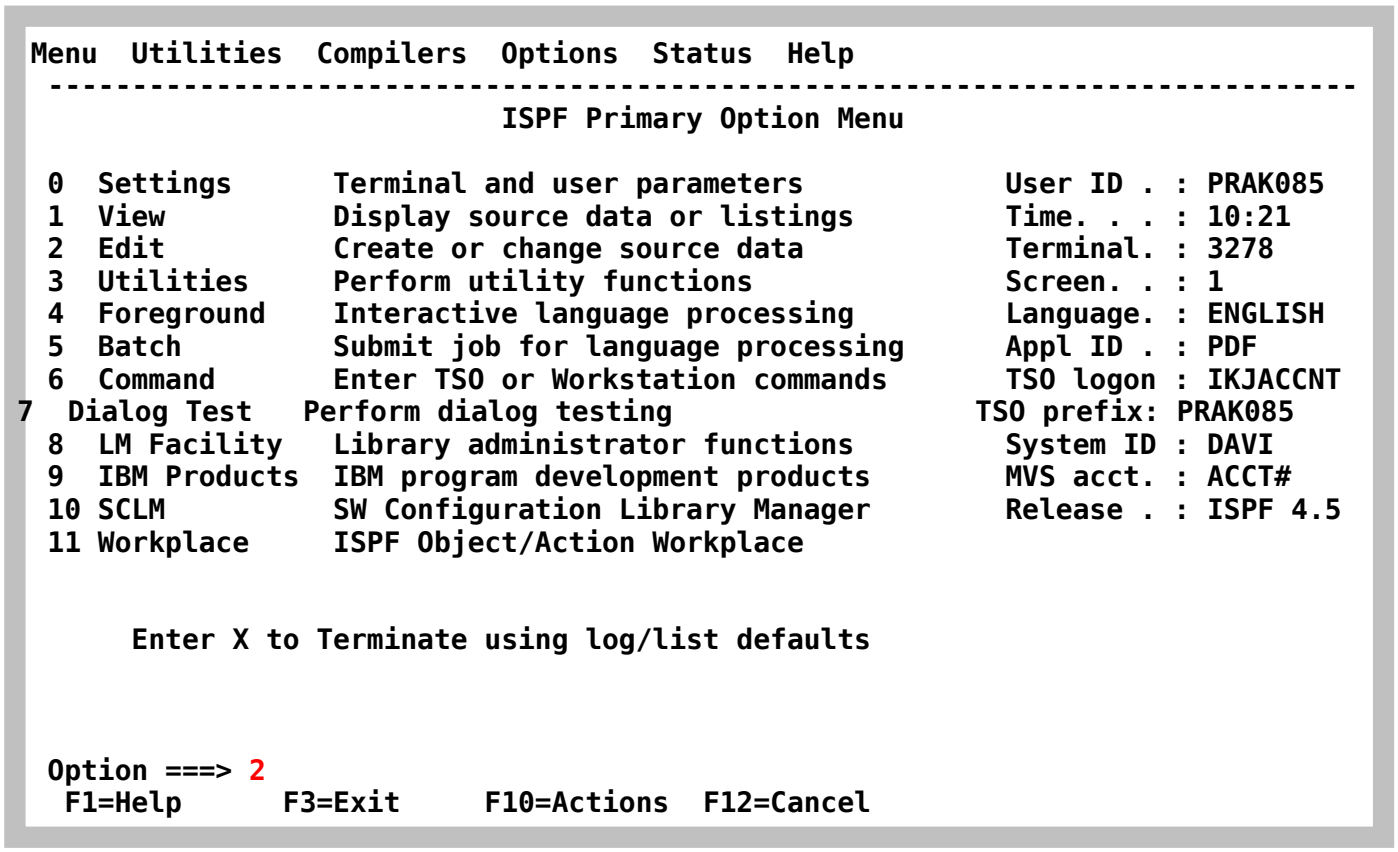


Wir benennen den neuen PDS „PRAK085.REXX.EXEC . Allocate, Enter

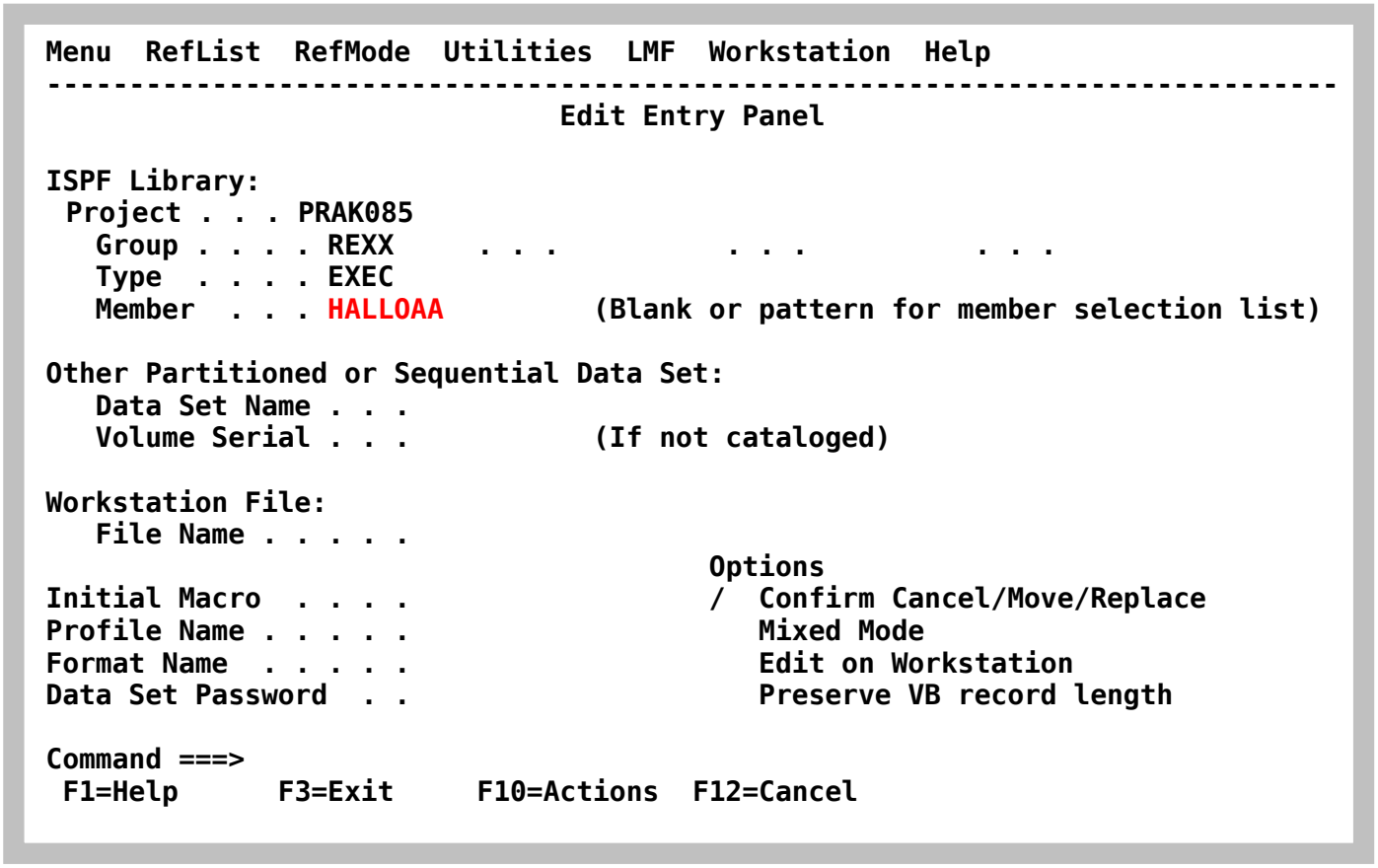


Die übliche Struktur eines Partitioned Data --ets wählen

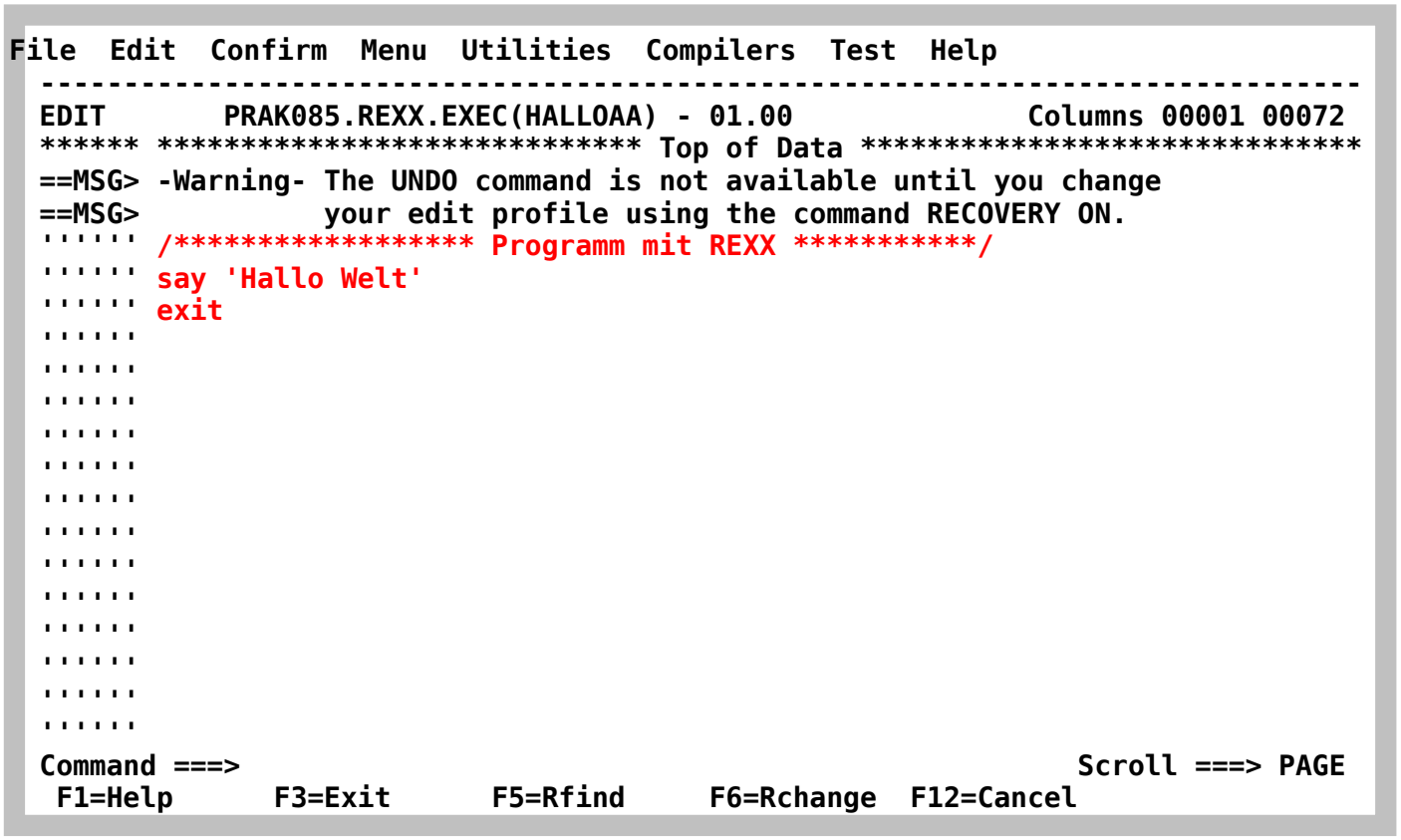
drei mal F3 Taste drücken



In den Editor um das REXX Programm zu schreiben. Enter



Wir schreiben unser REXX Programm in den Member HALLOAA



So sieht das Programmm aus.

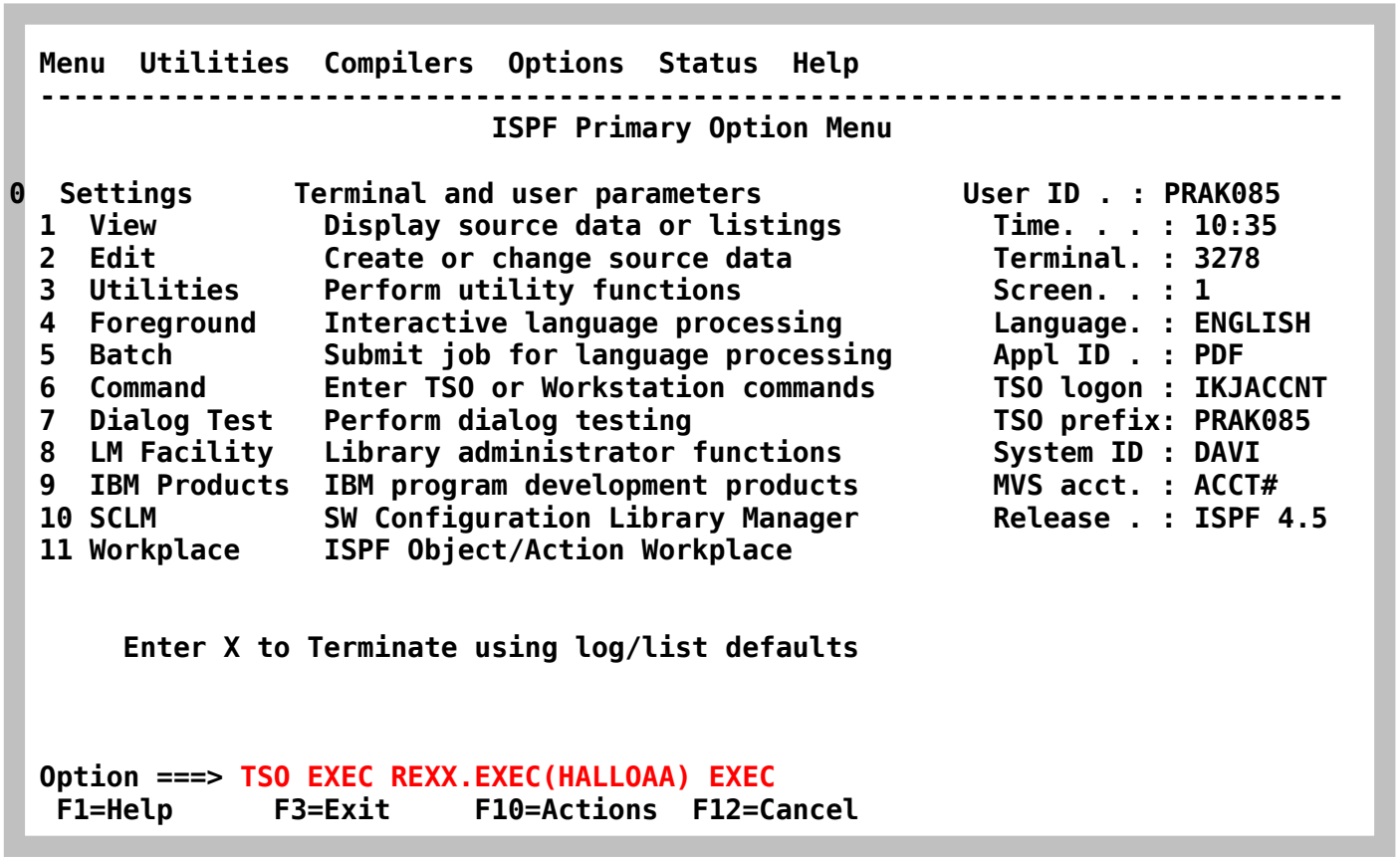
Ein z/OS REXX Programm beginnt grundsätzlich mit einer Kommentar Zeile mit beliebigem Inhalt. Es ist guter Programmierstiel, es mit einem „exit“ Statement abzuschließen, obwohl das nicht immer erforderlich ist. Für Plattformen wie Linux oder Windows können leicht andersartige Regeln gelten.

An Stelle eines Schüsselwortes wie print, write oder put verwendet REXX das Schlüsselwort „say“

F3 und nochmal aufrufen um das Ergebnis zu kontrollieren

```
File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
 --------------------------------------------------------------------------------s
EDIT       PRAK085.REXX.EXEC(HALLOAA) - 01.00              Columns 00001 00072
 ****** ***************************** Top of Data ******************************
 ==MSG> -Warning- The UNDO command is not available until you change
 ==MSG>           your edit profile using the command RECOVERY ON.
 000100 /****************** Programm mit REXX ***********/
 000200 say 'Hallo Welt'
 000300 exit
 ****** **************************** Bottom of Data ****************************
 Command ===>                                                  Scroll ===> PAGE
  F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
```

Sieht ok aus. Zwei mal F3 betätigen



Im ISPF Primary Option Menue den Befehl “TSO EXEC REXX.EXEC(HALLOAA) EXEC“ eingeben. Enter

Wir brauchen kein JCL Script, um PRAK085.REXX.EXEC(HALLOAA) auszuführen, da es hierbei um ein Script handelt, was ohne Compile direkt ausgeführt werden kann.

```
Menu  Utilities  Compilers  Options  Status  Help
 ------------------------------------------------------------------------------
                           ISPF Primary Option Menu

 0  Settings      Terminal and user parameters            User ID . : PRAK085
 1  View          Display source data or listings         Time. . . : 10:35
 2  Edit          Create or change source data            Terminal. : 3278
 3  Utilities     Perform utility functions               Screen. . : 1
 4  Foreground    Interactive language processing         Language. : ENGLISH
 5  Batch         Submit job for language processing      Appl ID . : PDF
 6  Command       Enter TSO or Workstation commands       TSO logon : IKJACCNT
 7  Dialog Test   Perform dialog testing                  TSO prefix: PRAK085
 8  LM Facility   Library administrator functions         System ID : DAVI
 9  IBM Products  IBM program development products        MVS acct. : ACCT#
 10 SCLM          SW Configuration Library Manager        Release . : ISPF 4.5
 11 Workplace     ISPF Object/Action Workplace


      Enter X to Terminate using log/list defaults

 Hallo Welt
 ***
```

Und hier ist das Ergebnis. Einfach, ist es nicht?

Die Ausgabe * weist darauf hin, dass TSO eine Bestätigung möchte. Diese wird durch Betätigen der Eingabetaste erreicht.**

Wie unter der weiterführenden Literatur zu finden kann man mit REXX sehr viele Dinge machen. Hier finden Sie ein Beispiel, wie Sie einen Bruch berechnen:

```
File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
 ------------------------------------------------------------------------------s
 EDIT       PRAK085.REXX.EXEC(MEXEC) - 01.00               Columns 00001 00072
 ****** ***************************** Top of Data ******************************
 ==MSG> -Warning- The UNDO command is not available until you change
 ==MSG>           your edit profile using the command RECOVERY ON.
 000001 /******REXX***************************************************/
 000002 /*     EXEC für Division zweier Zahlen                        */
 000003 /*************************************************************/
 000004 say 'Zaehler :'
 000005 pull z
 000006 say 'Nenner :'
 000007 pull n
 000008 if n=0 then say 'Division durch 0 nicht erlaubt'
 000009 else do
 000010   say z 'geteilt durch' n 'ist' z/n
 000011   say 'oder'
 000012   say z 'geteilt durch' n 'ist' z%n 'Rest' z//n
 000013 end
 ****** **************************** Bottom of Data ****************************


 Command ===>                                                  Scroll ===> PAGE
  F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
```

Hierfür legt man im Editor einfach einen weiteren Member neben HALLOAA an, zum Beispiel MEXEC. Die restlichen Schritte erfolgen wie bereits gehabt.

<u>Aufgabe:</u> ***Schreiben Sie ein drittes REXX-Programm. Dieses muss als Bildschirmausgabe sowie den Namen des Programm-Autors oder – wenn die Aufgaben von mehreren Teilnehmern gemeinsam gelöst werden – die Namen aller Programm-Autoren liefern.***

<u>Aufgabe:</u> ***Erstellen Sie mit Hilfe der Tastenkombination "ALT-Druck" zwei Screenshots (einen, der den Programm-Code und einen, der die Bildschirmausgabe Ihres dritten REXX-Programms enthält). Schicken Sie beide Screenshots Ihrem Betreuer per E-Mail zu. Achten Sie darauf, dass ein Screenshot nicht mehr als 250 KByte Speicherplatz belegt. Sehr gut ist das JPEG-Format, das mit weniger als 90 KByte auskommt. Löschen Sie nichts von Ihren Daten! Ihr Tutor möchte sich diese anschauen sowie Ihre REXX-Programme aufrufen.***

Einige Möglichkeiten für derartige weitere REXX Programme:

1. Rao-Zahlen

Es gibt Zahlen für die gilt, daß $a^2 + b^2$ = a||b ist. Mit a||b ist gemeint, dass a und b hintereinander als eine Zahl aufgeschrieben wird. So wird aus a = 53 und b = 23 die Zahl 5323 . Diesen Vorgang nennt man Konkatenation. Ein Beispiel für eine Rao-Zahl ist 1233, da $12^2 + 33^2$ = 1233 ergibt.

Schreibe in REXX ein Programm, das die Rao-Zahlen bis 1.000.000 ausgibt.

2. Weitere REXX Beispiele sind zu finden unter
http://www.rexxinfo.org/html/rexxinfo2.html#Rexx-Tools .

Kostenlose REXX Lehrbücher im Internet

Introductory Rexx Tutorial
http://www.rexxla.org/About_Rexx/rexxtut.html

REXX Tutorial
http://users.comlab.ox.ac.uk/ian.collier/Rexx/info.html

REXX
http://de.wikibooks.org/wiki/REXX

Einführung in die Programmiersprache REXX
http://lptp7.gm.fh-koeln.de/schulung/LUTZ/Rexx12/REXX.PDF

Kilowatt Software
http://www.kilowattsoftware.com/tutorial/Rexx/

Rexx – das Schweitzer Taschenmesser
http://www-lehre.informatik.uni-osnabrueck.de/~mpeussne/rexx/

Jim Barry's Rexx Tutor
http://www.kyla.co.uk/other/rexx1.htm

IBM Manuals and Books

SC28-1882 TSO/E V2R1.1 REXX User's Guide
SC28-1883 TSO/E Version 2 REXX/MVS Reference
SC23-3803 Using REXX to Access OpenEdition MVS Services
SC31-8231 TME 10 NetView for OS/390 V1R2 Customization: Using REXX
SC28-1905 OS/390 Using REXX and OS/390 UNIX System Services
SC28-1975 OS/390 V2R10.0 TSO/E REXX Reference
SC28-1974 OS/390 V2R9.0-R10.0 TSO/E REXX User's Guide
ZB35-5100 The REXX Language, 2nd Ed. -- Cowlishaw (see below)
SC26-4358 SAA CPI: Procedures Language Reference (Level 1)
SC24-5549 SAA CPI: REXX Level 2 Reference
G511-1430 IBM REXX Compiler and Library/370: Introducing the Next Step in REXX
SH19-8160 REXX/370 (Compiler and Library/370): User's Guide and Reference
SK2T1402 REXX/370 Compiler and Library V1R2.0 Online Product Library
LY19-6264 IBM REXX Compiler and Library/370: Diagnosis Guide
SB20-0020 The REXX Handbook -- Ed. Goldberg & Smith (see below)
SR23-8926 The NetRexx Language -- M. F. Cowlishaw
SG24-2216 Creating Java Applications Using NetRexx -- U. Wahli et al
SH21-0482 REXX Development System for CICS/ESA and REXX Runtime Facility for CICS/ESA , Guide and Reference
GG24-1615 Using REXX in Practice: EXEC2 to REXX Conversion Experiences
SC23-3803 Using REXX to Access OpenEdition MVS Services
SC23-3501 OS/390 Internet BonusPak II: Web Programming

Bücher

The REXX Language -- M. F. Cowlishaw
In English: ISBN 0-13-780735-X Prentice-Hall, 1985
ISBN 0-13-780651-5 (Second edition), 1990
In German: ISBN 3-446-15195-8 Carl Hanser Verlag, 1988
ISBN 0-13-780784-8 P-H International, 1988
In Japanese: ISBN 4-7649-0136-6 Kindai-kagaku-sha, 1988

Modern Programming Using REXX -- Robert P. O'Hara and David R. Gomberg
In English: ISBN 0-13-597311-2 Prentice-Hall, 1985
ISBN 0-13-579329-5 (Second edition), 1988
(From REXXPress, REXXPress@wcf.com)

REXX In the TSO Environment -- Gabriel F. Gargiulo
ISBN 0-89435-354-3, QED Information Systems Inc.,
Order #CC3543; 320pp, 1990
Revised edition:
ISBN 0-89435-418-3, QED Information Systems Inc.,
471pp, 1993

The REXX Handbook -- Edited by Gabe Goldberg and Phil Smith III
ISBN 0-07-023682-8, 672pp, McGraw Hill, 1991

Practical Usage of REXX -- Anthony S. Rudd
ISBN 0-13-682790-X, Ellis Horwood (Simon & Schuster), 1990

Using ARexx on the Amiga -- Chris Zamara and Nick Sullivan
ISBN 1-55755-114-6, 424pp+diskette, Abacus, 1991

Programming in REXX -- Charles Daney
ISBN 0-07-015305-1, 300pp, McGraw Hill, 1992

The ARexx Cookbook -- Merrill Callaway
ISBN 0-9632773-0-8, 221pp, Whitestone, 1992
(Companion diskette: ISBN 0-9632773-1-6)

Command Language Cookbook -- Hallett German
ISBN 0-442-00801-5, 352pp, Van Nostrand Reinhold, 1992

REXX--Advanced Techniques for Programmers -- Peter C. Kiesel
ISBN 0-07-034600-3, 239pp, McGraw Hill, 1993

REXX Tools and Techniques -- Barry K. Nirmal
ISBN 0-89435-417-5, 264pp, QED, 1993

REXX in der Praxis -- Peter Kees
ISBN 3-486-22666-5, 279pp, Oldenbourg, 1993

VX-Rexx for OS/2 (Programmer's Guide and Reference) 2.0
ISBN 1-55094-074-0 Watcom International Corp., 724pp, 1993

REXX Reference Summary Handbook (OS/2) -- Dick Goran
ISBN 0-9639854-0-X C F S Nevada, Inc, 102pp, 1993.

ISBN 0-9639854-1-8 (second edition), 148pp, 1994.
ISBN 0-9639854-2-6 (third edition, for Warp), 208pp, 1995.
ISBN 0-9639854-3-4 (fourth edition, for Warp 4), 270pp, 1997.

OS/2 2.1 REXX Handbook: Basics, Applications, and Tips -- Hallett German
ISBN 0442-01734-0, 459pp, Van Nostrand Reinhold, 1993

Proceedings of the REXX Symposium for Developers and Users
SLAC Report-422, 247pp, May 18-20, 1993

Mastering OS/2 REXX -- Gabriel F. Gargiulo
ISBN 0-471-51901-4, 417pp, Wiley-QED, 1994

Application Development Using OS/2 REXX -- Anthony S. Rudd
ISBN 0-471-60691-X, 416pp, Wiley-QED, 1994

Writing VX-Rexx Programs -- Ronny Richardson
ISBN 0-07911-911-5, 336pp+CD-ROM, McGraw-Hill, 1994

Teach Yourself REXX in 21 Days -- William F. Schindler & Esther Schindler
ISBN 0-672-30529-1, 527pp, SAMS, 1994

Writing OS/2 REXX Programs -- Ronny Richardson
ISBN 0-07052-372-X, 390pp, McGraw-Hill, 1994

REXX Procedursprak--hur du programmerar din PC med OS/2 -- Bengt Kynning
ISBN 91-44-48541-7, 300pp, Studentlitteratur (Sweden), 1994

The REXX Cookbook -- Merrill Callaway
ISBN 0-9632773-4-0, 319pp, Whitestone, 1995
ISBN 0-9632773-5-9 The REXX files (Companion disk)

Yasashii Rexx Nyumon (Introduction to Easy Rexx) -- Fumito Yamada
ISBN 4-88648-434-4, Tokyo: Daiwa Art/Di Art, diskette, 1995

ISPF/REXX Development for Experienced Programmers -- Lou Marco
ISBN 1-878956-50-7, 201pp, diskette, CBM Books, Ft. Washington, PA, 1995

REXX ni Yoru OS/2 Nyumon (Introduction to OS/2 via REXX) -- Naohiko Yamashita
ISBN 4-7898-3562-6 Tokyo: CQ Shuppan, 1996

REXX Jiyu Jizai (Handy with REXX) -- Takanori Seki
ISBN 4-7819-0810-1 Tokyo: Science-sha, 1996

Object REXX by Example -- Gwen L. Veneskey et al.
ISBN 0-9652329-0-5, 331pp+disk, Aviar Inc. (+1-412-488-9730), 1996

Practical Usage of MVS REXX -- Anthony S. Rudd
ISBN 3-540-19952-7, 310pp, Springer-Verlag, 1996

Object REXX for OS/2 Warp: Rexx Bytes Objects Now -- Trevor Turton et al.
ISBN 0-13-273467-2, 320pp+disk, Prentice-Hall, 1997

Das REXX Lexikon -- Gerd Leibrock
ISBN 3-486-23672-5, 598pp, Oldenbourg Verlag, 1997

The NetRexx Language -- M. F. Cowlishaw
ISBN 0-13-806332-X, 197pp, Prentice-Hall, 1997

Object Oriented Programming with REXX -- Tom Ender
ISBN 0-471-11844-3, 253 pp, Wiley Computer Publishing, diskette, 1997

Object REXX for Windows NT and Windows 95 -- Ueli Wahli et al
ISBN 0-13-858028-6, 490pp+disk, Prentice-Hall, 1997

| Down to Earth REXX -- William F. (Bill) Schindler
| ISBN 0-9677590-0-5, 507pp, Perfect Niche Software, 2000

| Practical Usage of TSO REXX -- Anthony S. Rudd
ISBN 1-85233-261-1, 324pp, Springer-Verlag, 2000

RDz Cobol Tutorial 01
RDz Einführung

Dank an Frau Isabel Arnold, die in Hamburg im August 2006 eine IBM Training Session veranstaltete. Hier wurde das Material in dieser und in den beiden folgenden Tutorials vorgeführt. Die ursprüngliche Version des Materials wurde von Reginaldo W. Barosa, IBM Executive IT Specialist, erstellt.

„Rational Developer für System z“ (RDz) ist eine auf Eclipse basierende moderne Entwicklungsumgebung, welche unter anderem ein universelles Werkzeug für die Anwendungsentwicklung auf Mainframes darstellt. RDz wird in immer mehr Unternehmen für die Anwendungsentwicklung eingesetzt.

Nahezu alles, was unter Nutzung eines 3270-Emulators auf einem Mainframe möglich ist, ist auch unter RDz möglich; z.B. Datasets anlegen (allocate), Datasets mit einem Editor bearbeiten, Jobs in Form von JCL-Scripten mittels "SUB" ausführen, Output-Jobs ansehen, Member kopieren etc.

Dieses Tutorial demonstriert anhand von Beispielen den grundlegenden Umgang mit RDz. Details in den RDz Tutorials 2 – 4.

Voraussetzungen für die Durchführung dieses Tutorials:

- Internetfähiger Windows PC
- PRAK(xxx)-Login auf Leia oder Hobbit
- Login auf unseren virtuellem RDz-Server mit installiertem RDz.

Wir verwenden für die verschiedenen Tutorial Texte 3 unterschiedliche RDz Versionen, die über unterschiedliche IP Adressen gestartet werden:

- 139.18.8.211 für Version 6.0 (WSED)
- 139.18.8.212 für Version 7.0 (WDz)
- 139.18.8.214 für Version 7.5 (RDz)

} Der Einfachheit halber bezeichnen wir alle Versionen als RDz, obwohl dies nicht den IBM Bezeichnungen entspricht.

Der Text für dieses Tutorial basiert auf 139.18.8.212 für Version 7.0 (WDz), kann aber von Ihnen auch unter 139.18.8.211 für Version 6.0 oder 139.18.8.214 für Version 7.5 (RDz) durchgeführt werden. Die ebenfalls verfügbaren Tutorials in der Programmiersprache PL/1 verwenden 139.18.8.214 für Version 7.5 (RDz). Zu beachten ist lediglich, dass unterschiedliche Portnummern für JES und MVS Files angegeben werden müssen. Wir planen alle Tutorials nach RDz 8.0 zu konvertieren – really soon now -. Die Unterschiede sind jedoch vernachlässigbar.

Inhalt

Dieses Tutorial und die folgenden Tutorials sind installiert auf dem:
z/OS 1.8 System hobbit.cs.informatik.uni-tuebingen.de, oder 134.2.205.54, der Abteilung Technische Informatik der Universität Tübingen, und
z/OS 1.8 System leia.informatik.uni-leipzig.de, oder 139.18.4.30, der Abteilung Technische Informatik der Universität Leipzig

Alle RDz Installationen wurden durchgeführt von Isabel Arnold, Uwe Denneler, Elisabeth Puritscher und Mr. Martin Benjamin Storz, und unterstützt von Andreas Hermelink (alle IBM). Lokale Unterstützung ist verfügbar von Andreas Nagel (Universität Tübingen) sowie (Frank Güttler (Universität Leipzig).

Dieses Tutorial verwendet die folgenden Konventionen

1k bedeutet 1 Klick mit der linken Maustaste
2k bedeutet 2 Klick mit der linken Maustaste
1kr bedeutet 1 Klick mit der rechten Maustaste

1. Übersicht

Software wird normalerweise in einer Entwicklungsumgebung erzeugt und anschließend in einer unterschiedlichen Produktionsumgebung ausgeführt. Traditionelle z/OS Entwicklungsumgebungen sind TSO mit ISPF, sowie CMS unter dem z/VM Betriebssystem. Typische z/OS Produktionsumgebungen sind JES, CICS, IMS, DB2 Stored Procedures und der WebSphere Web Application Server (WAS).

Ein Mainframe Rechner verfügt normalerweise über mehreren Logische Partitionen (LPARs), von denen eine als Entwicklungsumgebung und eine oder mehrere parallel als Produktionsumgebungen dienen.

Die Übernahme einer neu entwickelten Anwendung von der Entwicklungsumgebung in die Produktionsumgebung ist in der Regel ein sehr komplexer und aufwendiger Prozess. Normalerweise wird eine Test Umgebung zwischengeschaltet, die in einer weiteren getrennten LPAR untergebracht ist.. Während die Entwicklungsumgebung sehr viel anders als die Produktionsumgebung sein kann (schließlich soll Code nur entwickelt und debugged, nicht aber unter praxisnahen Bedingungen ausgeführt werden), versucht eine Testumgebung eine Produktionsumgebung möglichst naturgetreu abzubilden. Eine neu entwickelte Mainframe Anwendung wird zunächst in der Testumgebung auf Herz und Nieren überprüft, ehe sie in der Produktionsumgebung installiert und für den täglichen Einsatz freigegeben wird.

Häufig besteht eine neue z/OS Anwendung aus einem Teil, der auf dem Mainframe, und einem weiteren Teil, der auf einem Linux oder Windows Vorrechner läuft. Sehr typisch ist z.B. eine CICS Anwendung auf dem Mainframe, deren graphische Präsentation auf einem Linux/Windows Server mit Hilfe des WebSphere Application Servers implementiert wird.

Bis etwa zum Jahre 2000 wurden neue Mainframe Anwendungen fast ausschließlich mit Hilfe einer Entwicklungsumgebung implementiert, die ebenfalls auf dem Mainframe lief. Seitdem gewinnt zunehmend eine Alternative an Bedeutung, bei der die Entwicklungsumgebung auf einem Windows Server läuft. Diese Entwicklungsumgebung wurde seit 2000 von IBM unter unterschiedlichen Namen vertrieben, z.B. Websphere Studio Enterprise Developer (WSED) oder WebSphere Developer for System z (WDz). Die derzeitige Produktbezeichnung ist „Rational Developer for System z“ (RDz). In der Praxis ist RDz eine sehr evolutionäre Weiterentwicklung von WSED, und RDz eine ebenso evolutionäre Weiterentwicklung von WDz. Die Versionen bedingen unterschiedliche Installationen auf der Workstation und der z/OS Host Seite. Bei den vorliegenden Tutorials werden die Funktionsunterschiede jedoch nicht sichtbar.

Bei der Benutzung von RDz wird der Quellcode (COBOL, PL/I, C/C++, Java oder Assembler).auf einem Windows Rechner erzeugt, kompiliert und ausgetestet. Der gleiche Code kann auch auf dem Mainframe kompiliert und ausgeführt werden. Dieses parallele Vorgehen wird von RDz hervorragend unterstützt.

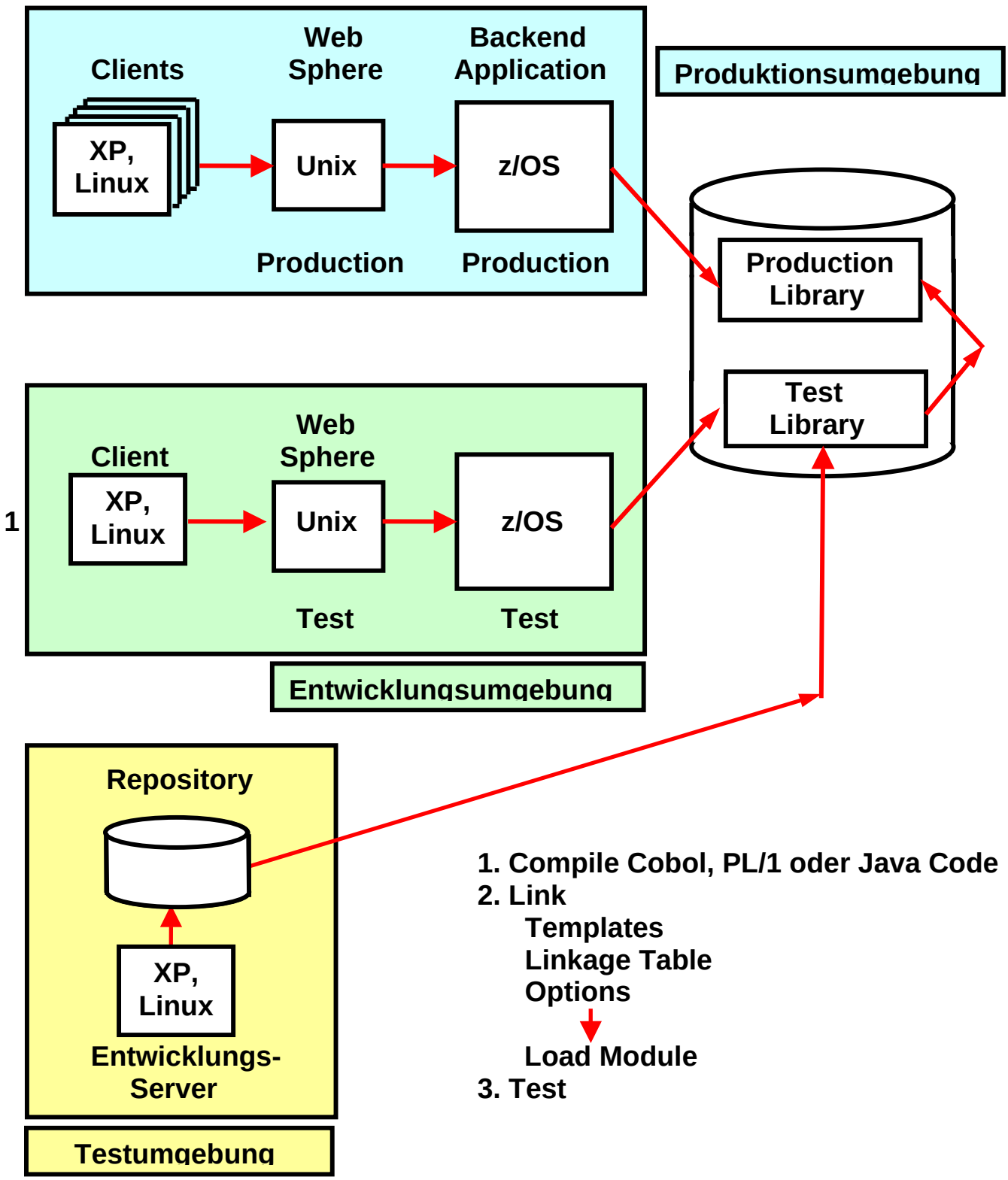


Abb. 1.1 Überführung von der Entwicklung in die Produktion

2. Integrierte Entwicklungsumgebung

Eine integrierte Entwicklungsumgebung (Abkürzung IDE, von engl. integrated Development Environment) ist eine Sammlung von Anwendungsprogrammen, mit denen die Aufgaben der Softwareentwicklung möglichst ohne Medienbrüche bearbeitet werden können.

Integrierte Entwicklungsumgebungen verfügen in der Regel über folgende Komponenten:

- **Texteditor**
- **Compiler bzw. Interpreter**
- **Linker**
- **Debugger**
- **Quelltextformatierungsfunktion**

Umfangreichere integrierte Entwicklungsumgebungen enthalten oft weitere hilfreiche Komponenten wie Versionsverwaltung, Projektmanagement, UML-Modellierung oder die Möglichkeit der einfachen Erstellung von grafischen Benutzeroberflächen (GUI).

In erster Linie sind integrierte Entwicklungsumgebungen hilfreiche Werkzeuge, die dem Softwareentwickler häufig wiederkehrende Aufgaben abnehmen, einen schnellen Zugriff auf wichtige Funktionen bieten, mit denen die Arbeits-(zwischen)-ergebnisse verwaltet und in spätere Bearbeitungsfunktionen direkt überführt werden können. Der Entwickler wird dadurch von formalen Arbeiten entlastet und kann sich ganz auf seine eigentliche Aufgabe, die Softwareentwicklung/Programmierung konzentrieren.

IDEs gibt es für nahezu alle Programmiersprachen und Plattformen. Oft wird nur eine Programmiersprache unterstützt. Es gibt aber auch IDE's, die mehrere Programmiersprachen unter einer gemeinsamen Benutzeroberfläche zusammenfassen.

Integrierte Entwicklungsumgebungen kamen in der ersten Hälfte der 1980er Jahre auf und lösten die damals übliche Praxis ab, Editor, Compiler, Linker und Debugger als vier getrennte Produkte anzubieten, die vom Benutzer über die Kommandozeile ausgeführt wurden. Eine der ersten erfolgreichen IDEs war Turbo Pascal.

Während die ersten IDEs noch textbasiert arbeiteten, ging der Trend vor allem bei den großen Anbietern ab 1990 zunehmend hin zu visuellen Programmierumgebungen.

http://de.wikipedia.org/wiki/Integrierte_Entwicklungsumgebung

2.1. Die wichtigsten IDE's

Der Oracle Developer enthält als wichtigste Komponente den JDeveloper. Der JDeveloper ist eine kostenlose Integrierte Entwicklungsumgebung (IDE) von Oracle, und unterstützt Sprachen wie Java, XML, SQL and PL/SQL, HTML, JavaScript, BPEL and PHP.

Visual Studio ist eine von dem Unternehmen Microsoft angebotene, integrierte Entwicklungsumgebung. Die wichtigsten unterstützten Sprachen sind VB.NET (Visual Basic .NET), C, C++ und C# .

NetBeans ist eine Open-Source Entwicklungsumgebung, die komplett in der Programmiersprache Java geschrieben wurde und auf der NetBeans Plattform läuft. Die NetBeans IDE wurde hauptsächlich für die Programmiersprache Java entwickelt, unterstützt jedoch auch C, C++ und dynamische Programmiersprachen.

Eclipse ist ein quelloffenes Programmierwerkzeug zur Entwicklung von Software verschiedenster Art. Ursprünglich wurde Eclipse als integrierte Entwicklungsumgebung für die Programmiersprache Java genutzt, aber mittlerweile wird es wegen seiner Erweiterbarkeit auch für viele andere Entwicklungsaufgaben eingesetzt. Für Eclipse gibt es eine Vielzahl sowohl quelloffener als auch kommerzieller Erweiterungen.

Das SAP NetWeaver Developer Studio (NWDS) von SAP ist eine auf Eclipse basierende integrierte Entwicklungsumgebung. Die IDE integriert sowohl Java-Technologien (JSE, JEE, XML, ...) als auch SAP-Technologien (Web Dynpro, Java Dictionary, ...). Es kommen viele Plug-ins des Web Toolkit Projekts zum Einsatz und die IDE wurde mit proprietären Erweiterungen von SAP ergänzt. Damit können Web-Dynpro- und JEE-Applikationen als Java-EE-konforme Anwendungen erstellt werden, welche auf SAPs Java EE NetWeaver Application Server zum Einsatz kommen.

Eclipse wurde ursprünglich von IBM entwickelt, dann aber als Open Source der Allgemeinheit zur Verfügung gestellt. Heute bieten zahlreiche Unternehmen proprietäre Erweiterungen (plugins) für Eclipse an, darunter Borland, HP, IBM und SAP.

IBM vermarktet proprietäre Eclipse Erweiterungen unter dem Namen Rational, spezifisch den Rational Application Developer (RAD). Ähnlich wie das SAP NetWeaver Developer Studio in erster Linie für Neuentwicklungen für den SAP Java EE NetWeaver Application Server eingesetzt wird, benutzt man RAD für Neuentwicklungen für den IBM WebSphere Application Server.

Eine Sonderstellung nimmt der IBM Rational Developer for System z (RDz) ein. RDz hat viele Gemeinsamkeiten mit RAD und basiert wie dieses auf Eclipse. RDz ist heute die führende IDE für die Entwicklung neuer z/OS Anwendungen und löst zunehmend TSO und ISPF in dieser Rolle ab. Allerdings wird die Benutzung von RDz wesentlich erleichtert, wenn Grundkenntnisse in TSO und ISPF vorhanden sind.

Selbst-Test

- **Muss man für die Entwicklung neuer z/OS Anwendung eine IDE benutzen ?**
- **Kann man an stelle von RDz auch Eclipse oder NetBeans oder NWDS benutzen ?**

2.2. Eclipse

Eclipse ist ein Open-Source-Framework zur Entwicklung von Software nahezu aller Art. Die bekannteste Verwendung ist die Nutzung als Entwicklungsumgebung (IDE) für die Programmiersprache Java. Aber auch für die Entwicklung von Rich-Client-Applikationen auf Basis der Eclipse Rich Client Platform (RCP) wird es zunehmend häufiger eingesetzt. Eclipse ist nicht auf Java festgelegt und wird aufgrund seiner offenen Plug-in-basierten Struktur mittlerweile für sehr unterschiedliche Entwicklungsaufgaben eingesetzt. Für Eclipse existieren eine Vielzahl von Plug-ins sowohl von Opensource-Projekten als auch kommerziellen Anbietern. Es existieren Plugins für weitere Programmiersprachen, unter anderem für C, C++, Perl, PHP, Ruby und Python.

Eclipse basiert auf der IDE „Visual Age for Java 4.0“, die von IBM entwickelt wurde. Im November 2001 wurde der Quellcode für das Programm freigegeben und seitdem kontinuierlich als Open Source Projekt erweitert.

Das von IBM geleitete Eclipse Konsortium umfasst über 80 Mitglieder neben IBM sind unter anderem Borland, MERANT, QNX Software Systems, Rational Software, Red Hat, SuSE, TogetherSoft und Webgain vertreten. Das Konsortium gründete im Januar 2004 die rechtlich unabhängige Eclipse Foundation, die seitdem für die Entwicklung von Eclipse verantwortlich ist. Etwa die Hälfte der derzeitig am Eclipse-Basisframework arbeitenden Entwickler werden weiterhin von IBM bezahlt.

Eclipse basiert auf einem Prinzip, bei dem sich alles als Plugin integrieren lässt.

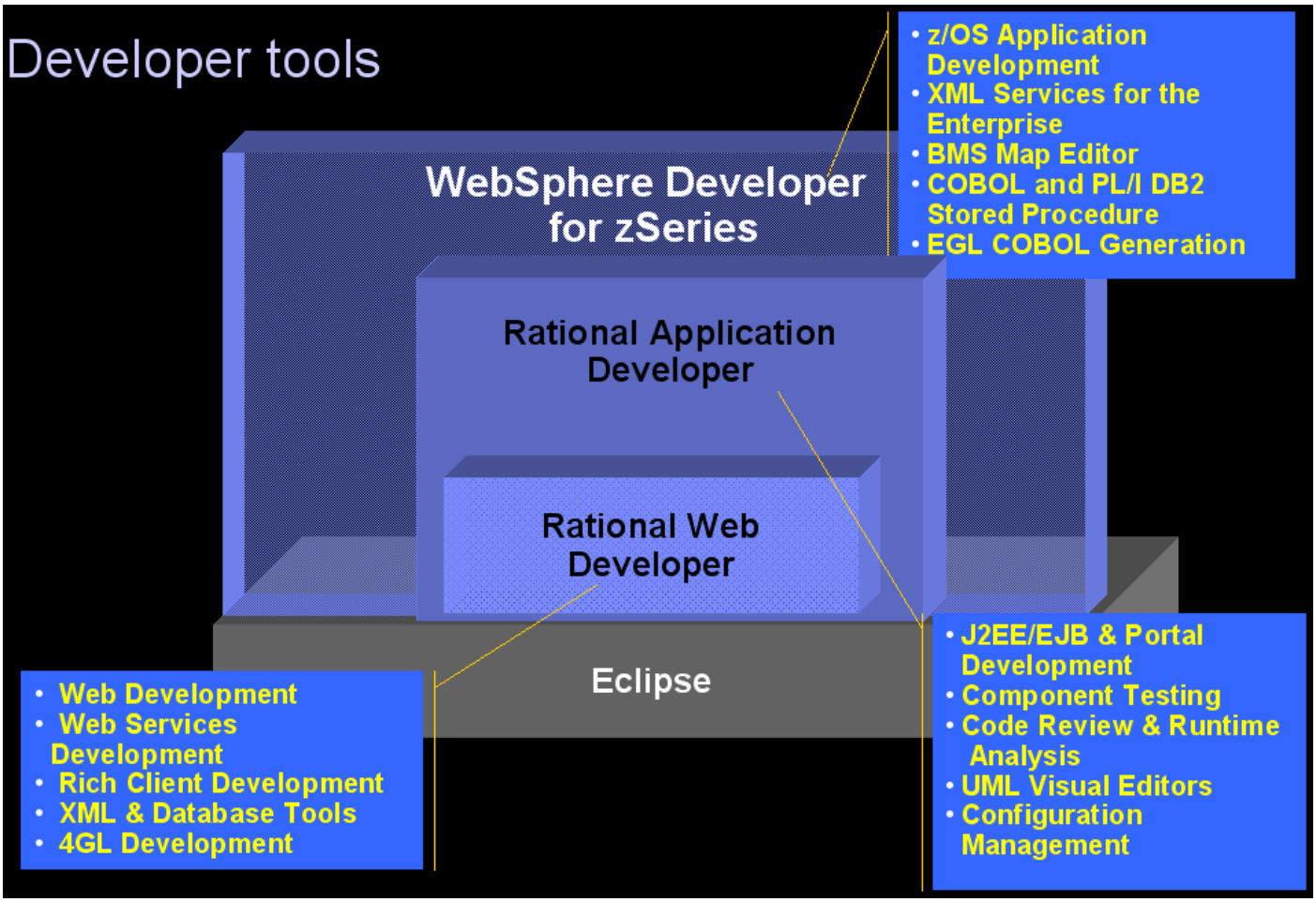
Seit Version 3.0 ist Eclipse selbst nur der Kern, der die einzelnen Plug-ins lädt, die dann die eigentliche Funktionalität zur Verfügung stellen. Diese Funktionalität basiert auf dem OSGi-Standard. Sowohl Eclipse als auch die Plugins sind vollständig in Java implementiert. Als GUI-Framework zur Erstellung der grafischen Oberfläche wurde SWT verwendet. Zur Darstellung der GUI-Komponenten basiert SWT ähnlich wie AWT auf den nativen GUI-Komponenten des jeweiligen Betriebssystems. Eclipse ist daher nicht platformunabhängig, wird aber für 14 verschiedene Systeme und Architekturen bereitgestellt. Die Plug-ins lassen sich durch den Download direkt in Eclipse von einem Update-Server oder durch einfaches Entpacken installieren.

Das frei verfügbare Eclipse SDK umfasst die Eclipse Platform, Werkzeuge zur Java-Entwicklung (Java Development Tools JDT) und die Umgebung zur Entwicklung von Eclipse-Plug-ins (Plug-in Development Environment PDE).

Die Eclipse Open Source Community versteht sich als Gemeinschaft, die eine kostenlose, erweiterbare Entwicklungsumgebung und ein Framework für die Laufzeit und Anwendungsentwicklung zur Verfügung stellt, welches die Entwicklung, Betreuung und Wartung eines Softwareprojekts über dessen gesamte Lebensdauer begleitet. Die Community stellt über 60 verschiedene Open Source Projekte bereit, die in sieben Kategorien unterteilt werden:

- **Enterprise Development**
- **Embedded und Device Development**
- **Rich Client Platform**
- **Rich Internet Applications**
- **Application Framework**
- **Application Lifecycle Management (ALM)**
- **Service Oriented Architecture (SOA)**

2.3 Rational Developer



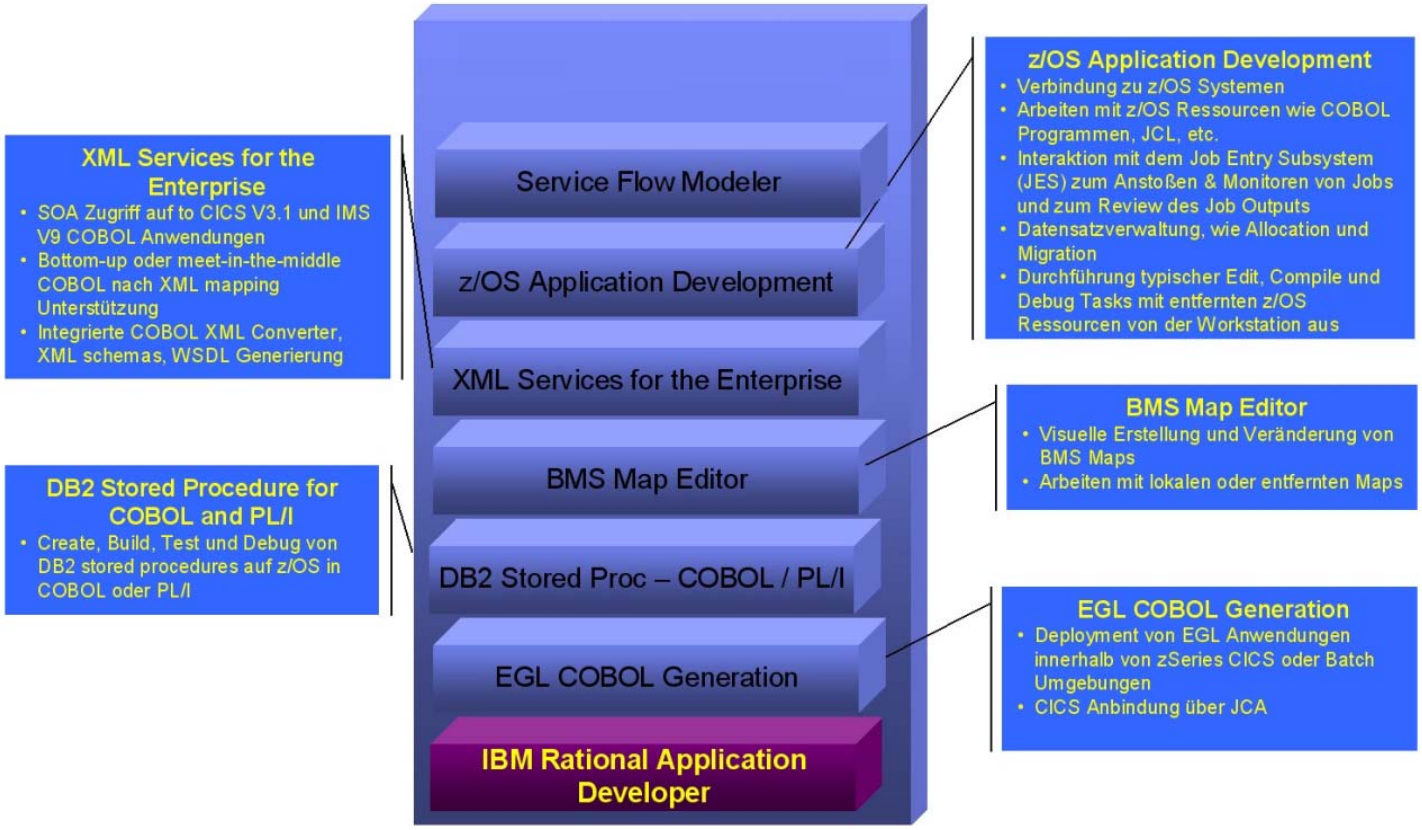
Von IBM existieren eine ganze Reihe von Eclipse Erweiterungen, die als Plugins in Eclipse integriert werden können. Die wichtigsten sind:

- **Rational Web Developer Plugin**
- **Rational Application Developer Plugin, welches das Rational Web Developer Plugin enthält.**
- **WebSphere Developer for System z Plugin, heute als Rational Developer for System z bezeichnet. WDz bzw. RDz enthalten das Rational Application Developer Plugin.**

Das von IBM bereitgestellte Plugin Rational Application Developer (RAD) unterstützt

- **die Entwicklung von HTML, XML, Java, JSP, Servlets, Enterprise JavaBeans und Webservices.**
- **eine integrierter Testumgebung, Debugger, Codegeneratoren für Tests**
- **integrierte Build-Prozesse.**

2.4 Rational Developer for System z (RDz)



Das von IBM bereitgestellte Plugin Rational Developer for System z (RDz) bietet zusätzlich zu der RAD Funktionalität Einrichtungen

- **zur integrierten Entwicklung von Großrechneranwendungen, z.B. die Entwicklung von COBOL, PL/I, C/C++, Java und Assembler Programmen**
- **für Entwicklungen für die Transaktionsmonitore IMS und CICS,**
- **für die Entwicklung von DB2 Stored Procedures.**

Sie können mit RDz auch Anwendungen entwickeln, die später unter Windows oder anderen Betriebssystemen laufen. Im Prinzip würde dafür das Basis Eclipse System ausreichen. RDz bietet den Vorteil, dass z.B. für die Kompilierung einer Windows Anwendung der IBM Enterprise Cobol oder Enterprise PL/I Compiler verwendet wird. Eclipse stellt hier Compiler zur Verfügung, die nicht vollständig kompatibel mit den IBM Enterprise Compilern sind.

Mit RDz ist es möglich, sich auf ein z/OS System einzuloggen und mit den auf dem z/OS System liegenden Daten zu arbeiten, als ob es Dateien der Workstation wären. Dies wird durch das sog. z/OS File System Mapping ermöglicht, bei dem auf der Workstation zusätzlich zu einzelnen Dateien auch spezielle Member dieser z/OS Datasets gemappt werden können. Das gestattet es mit Datasets zu arbeiten, die Member unterschiedlichen Typs beinhalten. Die Daten werden von der Workstation gemäß ihrer Mappingkriterien behandelt. Von der Workstation aus können auch neue Datasets und Member angelegt und bearbeitet werden.

Der bei der Bearbeitung der Dateien verwendete Editor hat volle ISPF (Interactive System Productivity Facility) Funktionalität und ist durch eine Vielzahl weiterer Module auf dem Stand aktueller Entwicklungsumgebungen. Hervorzuheben sind hierbei ein Content Assist für COBOL und PL/I, der bei Syntaxvervollständigung automatisch alle Ressourcen integriert, auf die ein Entwickler Zugriff hat. Weitere Funktionen sind lokale und remote Syntaxprüfung sowie die in Eclipse integrierten Werkzeuge Compare With... und Replace with Local History.

Ein weiterer Teil von RDz ist die Interaktion mit dem Job Entry System (JES), bei dem Jobs an den Großrechner übermittelt, überwacht und deren Ergebnisse betrachtet werden können. Die dabei verwendeten Job Control Language-Files (JCL-Files) für compile, link-edit und run können automatisiert aus dem vorhandenen Quellcode generiert und an das entsprechende z/OS System zur Ausführung gesandt werden.

Generell sind alle typischen Editier-, Kompilier- und Debugfunktionen lokal und auf dem remote z/OS System von der Workstation aus verfügbar.

Die meisten Funktionen können auch im Offline-Modus verwendet werden. Hier ist als Voraussetzung nötig, entsprechende Projekte und Daten offline verfügbar zu machen, wofür eine Routine in RDz bereit steht.

Um Änderungen lokal und offline testen zu können, ist für die Testumgebung der Workstation ein CICS Transaktionsserver in RDz integriert, der CICS-Anweisungen lokal übersetzt und dadurch ein Testen ermöglicht.

Ein weiterer Teil von WebSphere Developer for System z ist die Erstellung und Bearbeitung des IBM Basic Mapping Supports (BMS). Der dabei verwendete Editor ermöglicht die Bearbeitung der BMS Maps über ein Drag & Drop-Verfahren. Dargestellt und bearbeitet werden können die BMS Maps in einem Design View, bei dem direkt in eine angezeigte BMS Map Teile eingefügt werden können. Es besteht die Möglichkeit, neue Map Sets zu erstellen oder Bestehende zu importieren. Alle Arbeiten können lokal oder remote ausgeführt, sowie fertige Maps exportiert werden.

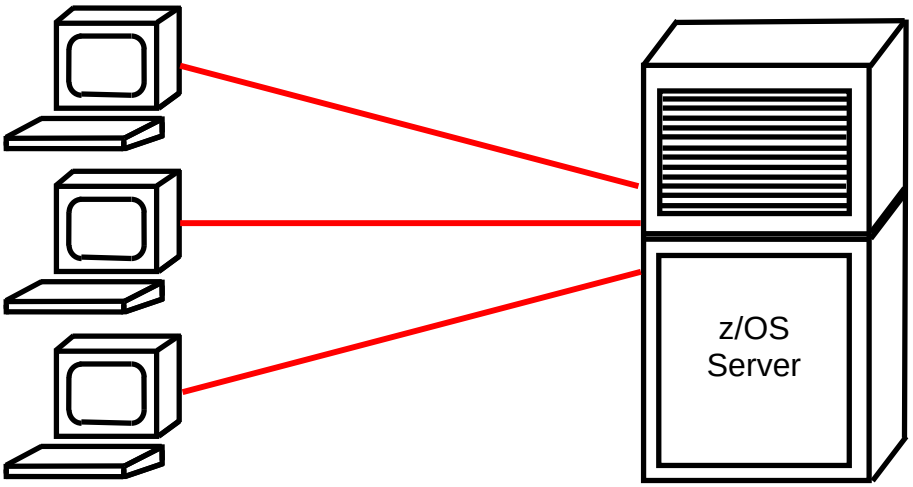
Mit RDz können COBOL und PL/1 Stored Procedures unter z/OS erstellt, getestet und im Bedarfsfall auf der Workstation debugged werden. Für die automatisierte Generierung der SQL Definitionen und der COBOL und PL/1 Stored Procedure Programme steht ein Wizard als Teil von RDz zur Verfügung.

Selbst-Test

- **Ist RDz Open Source ?**
- **Kann man mit RDz auch CICS Anwendungen entwickeln ?**
- **Kann man mit RDz auch Windows Anwendungen entwickeln ?**

3. Einloggen in den remote Windows XP-Server

3.1 Benutzung des Remote RDz Servers

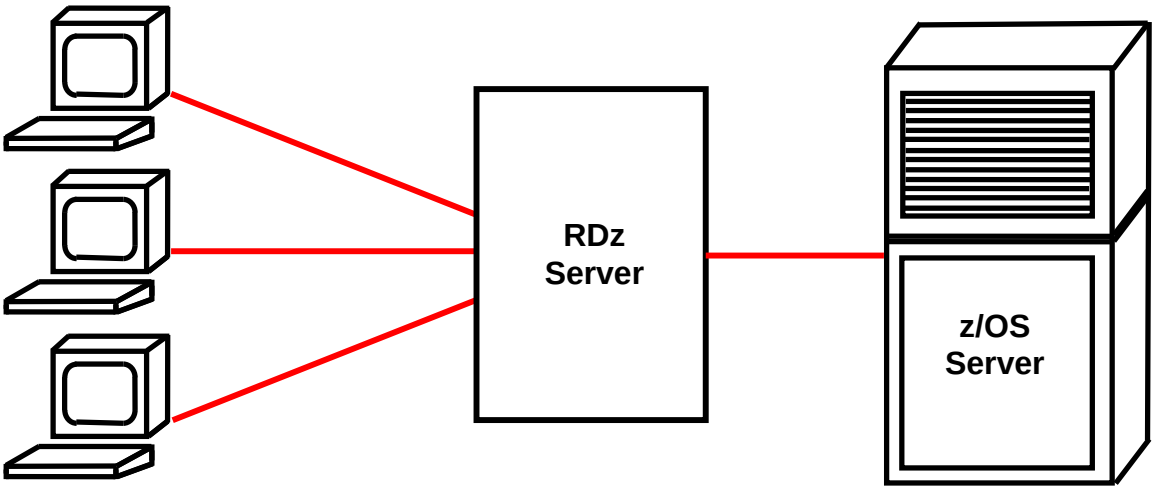


PC's mit RDz installiert

Abbildung 3.1

Abb. 1.1 zeigt eine Standard RDz Konfiguration. Auf allen teilnehmenden Arbeitsplatzrechnern ist Eclipse mit dem RDz Plugin installiert.

Eclipse belegt auf dem Plattenspeicher Ihres Arbeitsplatzrechners wenige 100 MByte Speicherplatz, Das RDz Plugin benötigt weitere 8 GByte.



PC's mit Remote Desktop

Abbildung 3.2

Um Ihnen die Arbeit der Installation zu ersparen verwenden wir die in Abb. 3.2 gezeigte Konfiguration. Jeder Arbeitsplatzrechner enthält lediglich die normale Windows Remote Desktop-Verbindung. Mit dieser erstellen wir eine Remotedesktop-Verbindung zu dem RDz Server des Institutes für Informatik. RDz läuft für alle Benutzer auf diesem Server.

Der RDz Server ist natürlich ein Performance Bottleneck. Für unsere praktischen Übungen hat er sich bisher aber als ausreichend bewährt.

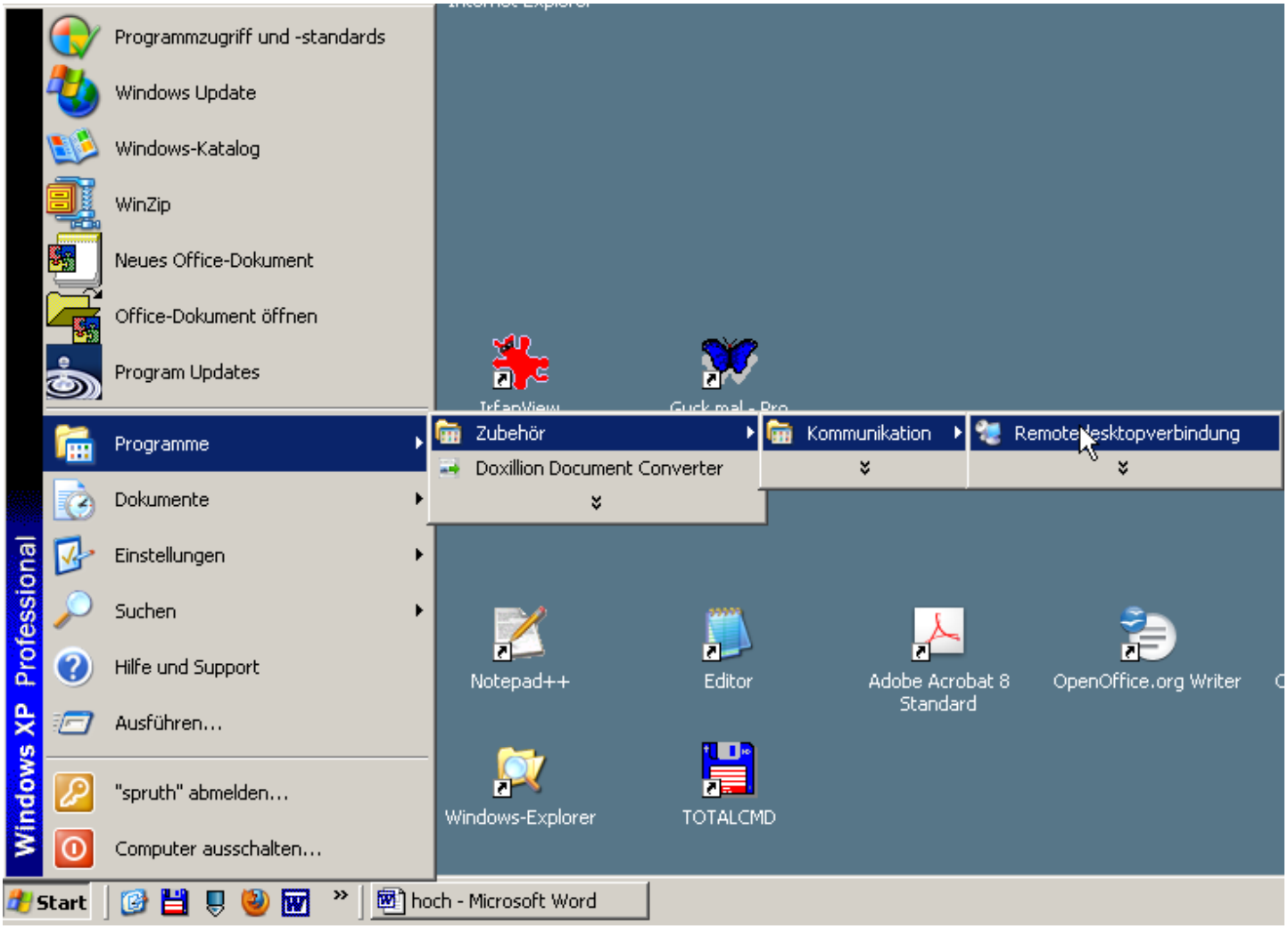


Abbildung 3.3

Start → Alle Programme → Zubehör → Kommunikation → Remotedesktopverbindung

Abbildung 3.4

"Verbinden" anklicken

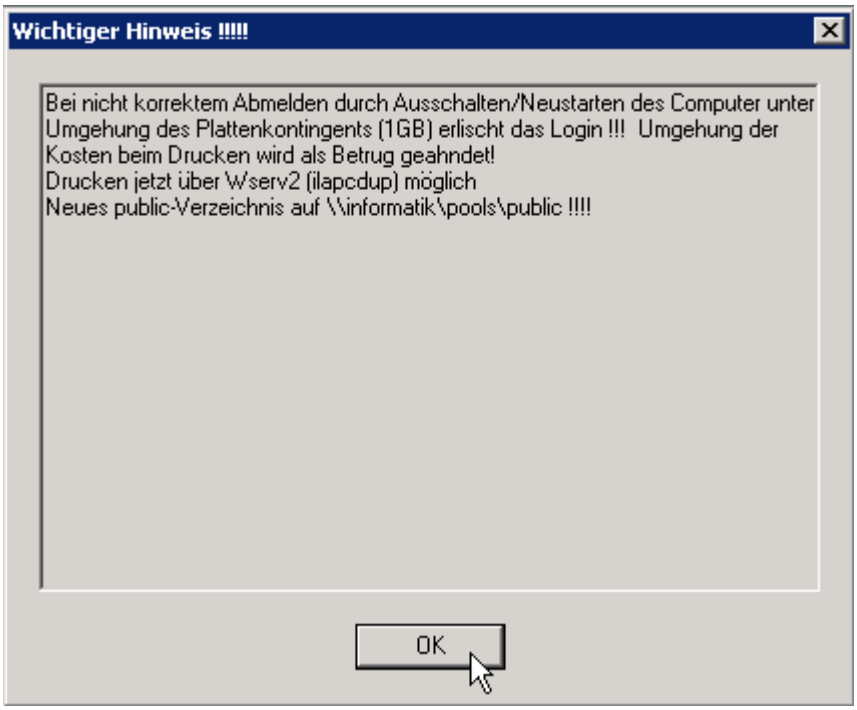


Abbildung 3.5
"OK" anklicken

Abbildung 3.6

Benutzername "IBMP<Nr>" und Passwort eingeben. Die von uns vergebenen Benutzernahmen haben typischerweise das Format "IBMP<Nr>", wobei <Nr> eine zweistellige Ziffer ist.

Dieser Benutzername ist ein Login auf einem entfernten Windows RDz Server.
Das Passwort lässt sich nicht ändern.

Nach ca. ½ Minute erscheint die Windows-Oberfläche der entfernten Maschine.

Selbst-Test

- **Was ist der Vor- und/oder Nachteil, einen RDz Server zu benutzen ?**
- **Würde man in praktischen Entwicklungsprojekten auch einen RDz Server benutzen ?**

3.2 RDz starten

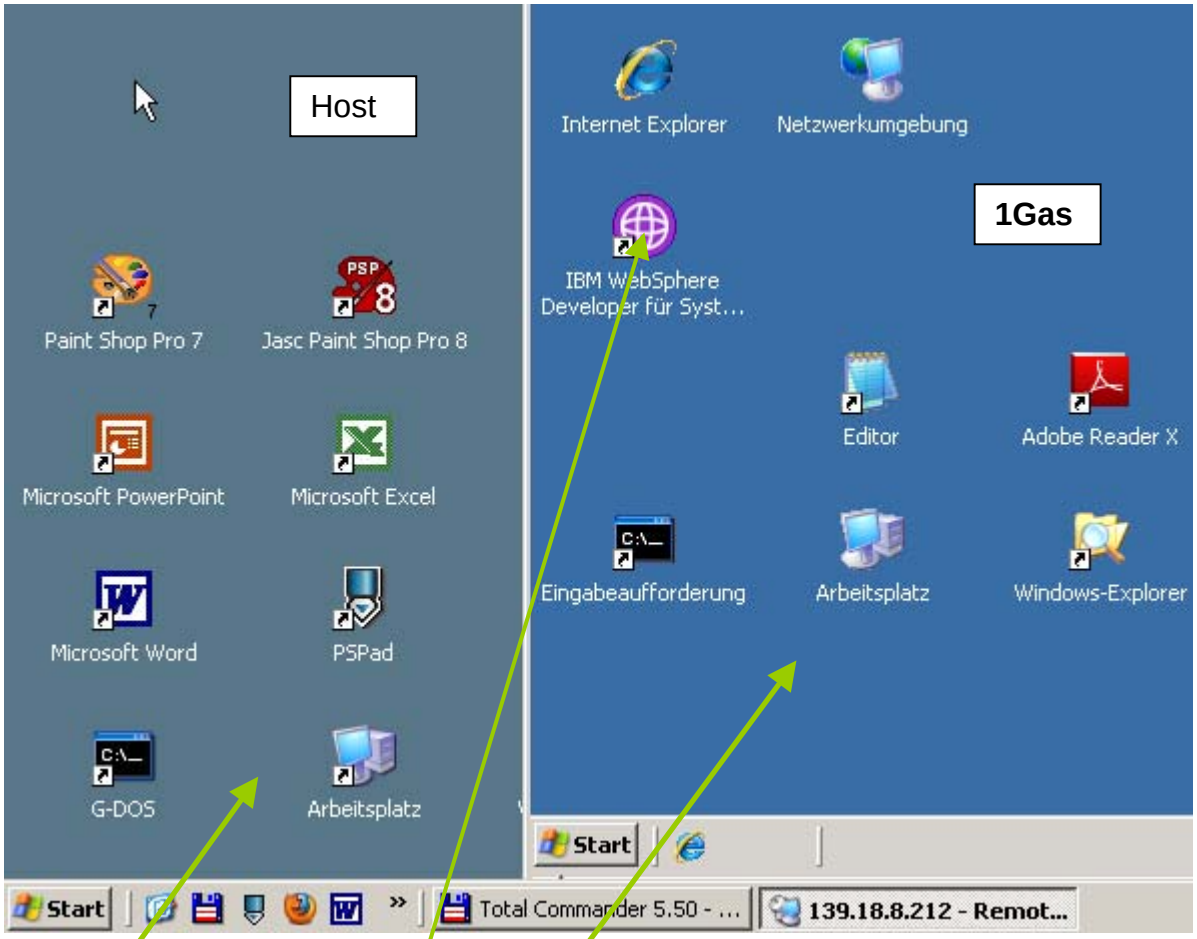


Abbildung 3.7

Es öffnet sich ein Prozess in einem getrennten Fenster, welches den entfernten Windows XP Server 139.18.8.212 repräsentiert, in Zukunft als „Gast“ bezeichnet. Dieses Fenster ist auf dem Desktop des PCs abgebildet, auf dem Sie die Remote Desktopverbindung aufgebaut haben (in Zukunft als Host bezeichnet). Durch anklicken mit der Maus auf dem Hostfenster oder dem Gastfenster können wir beide Fenster bedienen.

2k auf das „WebSphere Developer for System z“ Icon (RDz). Wenn es sich dort noch nicht befindet, ...

Start

→ Alle Programme → IBM Rational → IBM Rational Application Developer V6.0 → Rational Application Developer

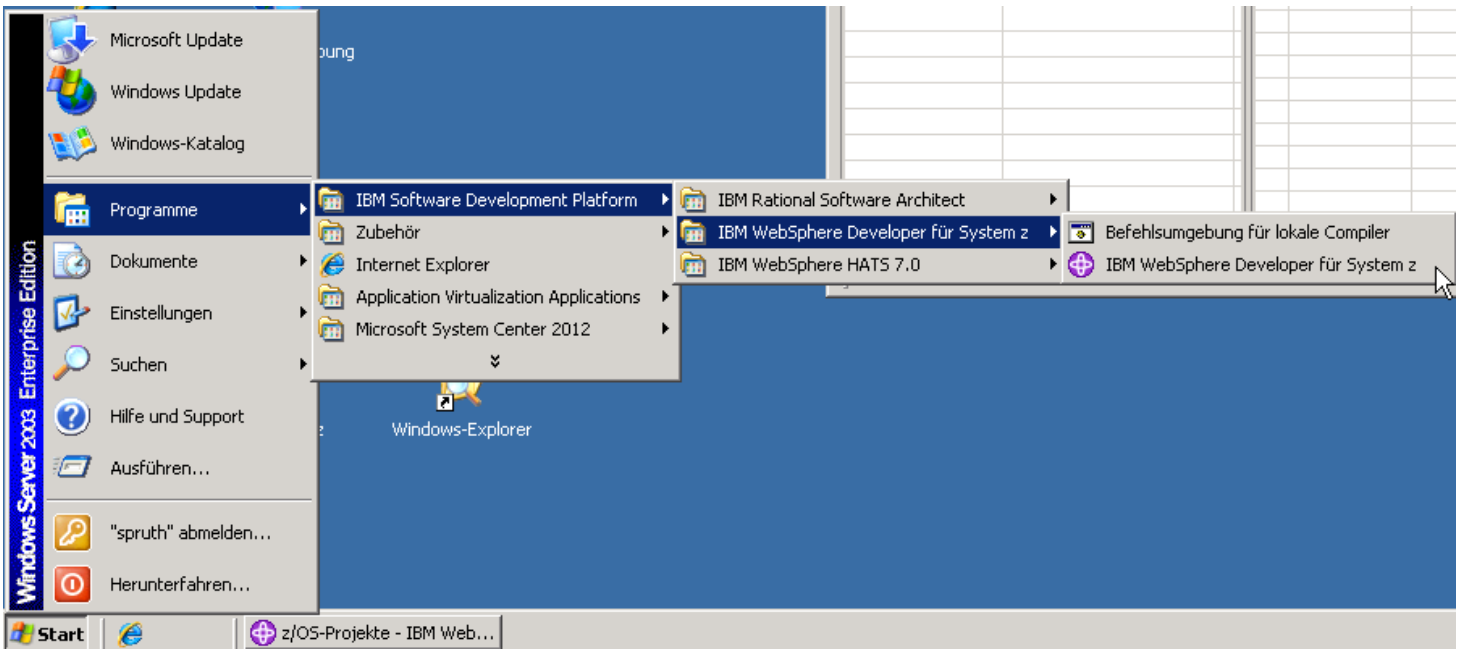


Abbildung 3.8

Der Ladevorgang des auf Eclipse basierenden Tools dauert ca. 1 Minute.

Anschließend soll ein Defaultwert für den Arbeitsbereich festgelegt werden:

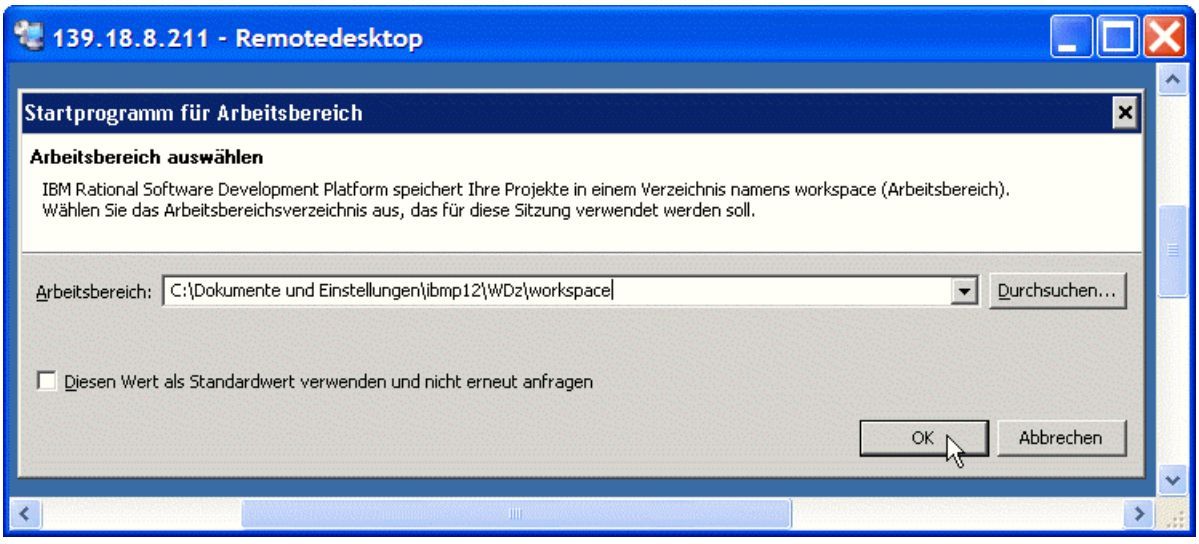


Abbildung 3.9

Es wird das Verzeichnis
C:\Dokumente und Einstellungen\ibmp12\WDz\ibmp12\WDz\Workspace
vorgeschlagen. Da auf dem remote Windows Server für jeden Benutzer eine eigene getrennte Festplatte mit der Bezeichnung z:\ eingerichtet wird, schlagen wir vor, dass Sie statt „C:\Dokumente und Einstellungen\ibmp12\WDz\ibmp12\WDz\Workspace“ den Wert „Z:\Workspace01“ benutzen.

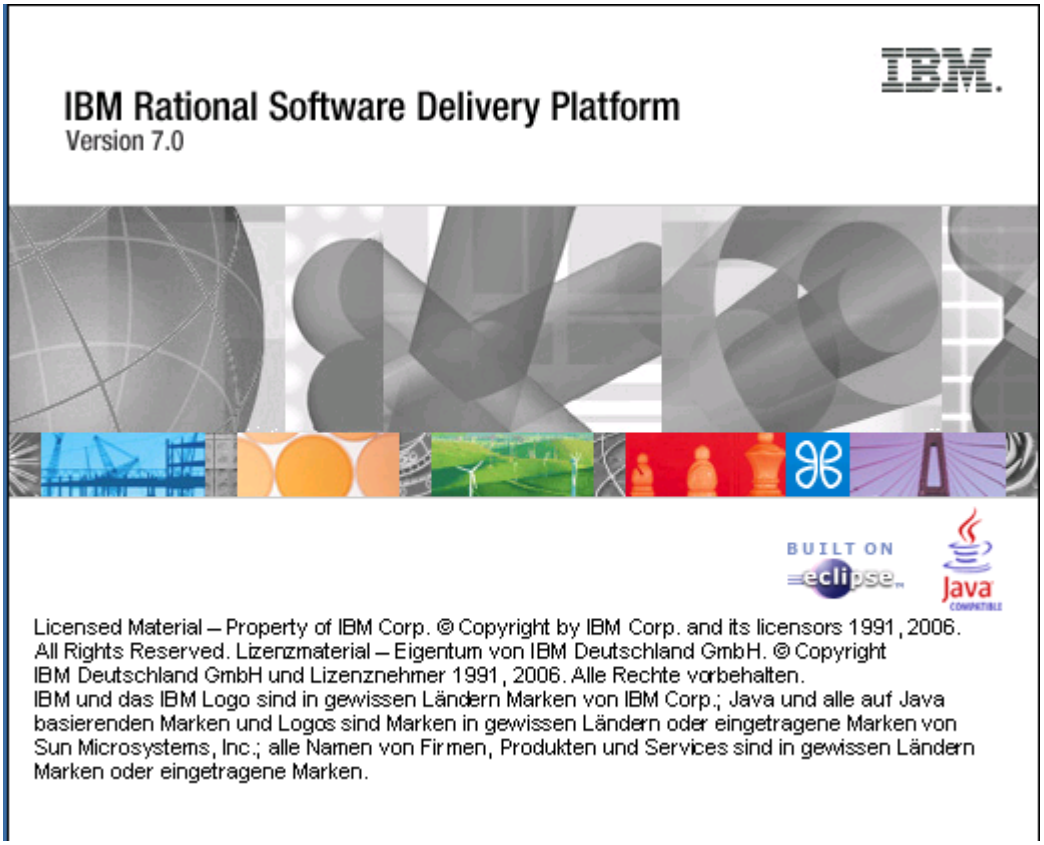


Abbildung 3.10

Der Welcome Screen erscheint. Bitte warten, dauert ca. 1 Minute. Dann erscheint

Evtl. erscheint ein spezielles Fenster.

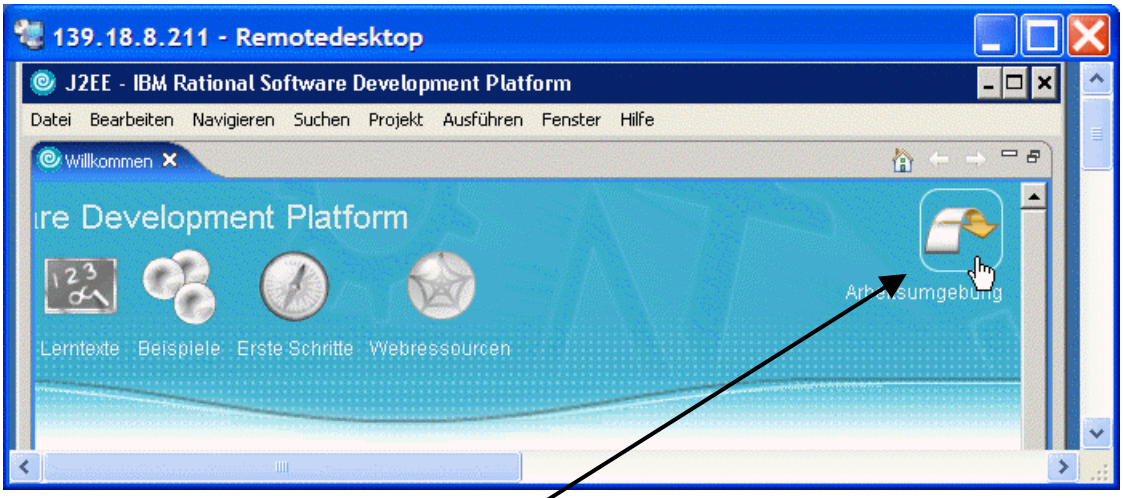


Abbildung 3.11

Beim wiederholten Einloggen erscheint dieses Fenster wahrscheinlich nicht mehr.

Klick auf "Arbeitsumgebung". Es erscheint der RDz Desktop.

3.3 RDz Desktop

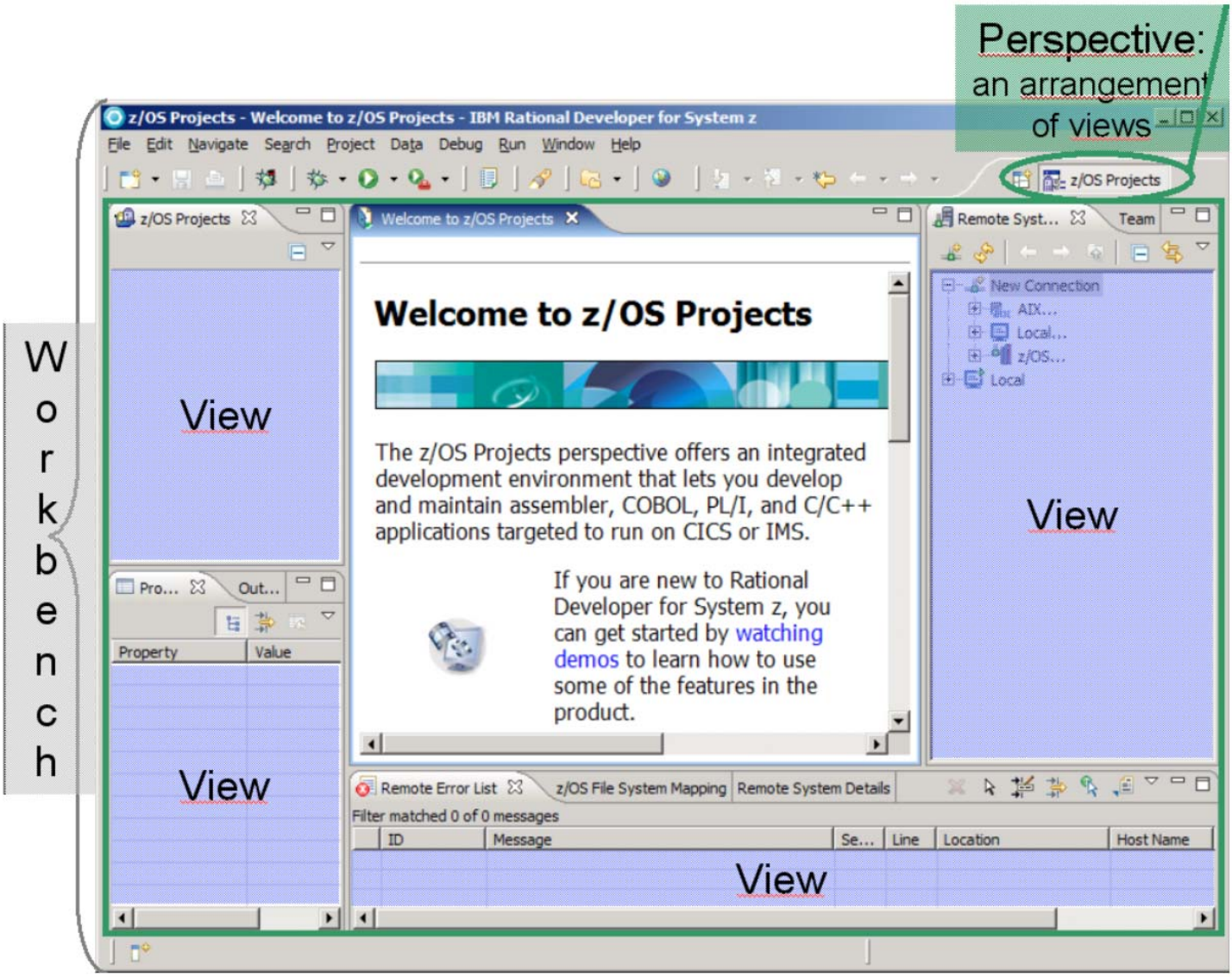


Abbildung 3.12

Eine RDz Oberfläche ist eine spezielle Art einer Eclipse Oberfläche. Programmierer mit Eclipse Vorkenntnissen haben in der Regel nur geringe Anlaufschwierigkeiten, sich in der RDz Oberfläche zurechtzufinden.

Eine Eclipse Oberfläche wird als Workbench bezeichnet. Sie besteht aus einer Reihe von Fenstern, von Eclipse als „Views“ bezeichnet. Für unterschiedliche Entwicklungsaufgaben existieren unterschiedliche Anordnungen der Views. Diese unterschiedlichen Anordnungen werden von Eclipse als „Perspectives“ bezeichnet. Es existieren spezielle Perspectives beispielsweise für XML, Java oder HTML Entwicklungen. Es ist einfach, in einer Eclipse Umgebung zwischen unterschiedlichen Perspectives hin und her zu schalten.

Für uns von besonderer Wichtigkeit ist die „z/OS Projects“. Perspective

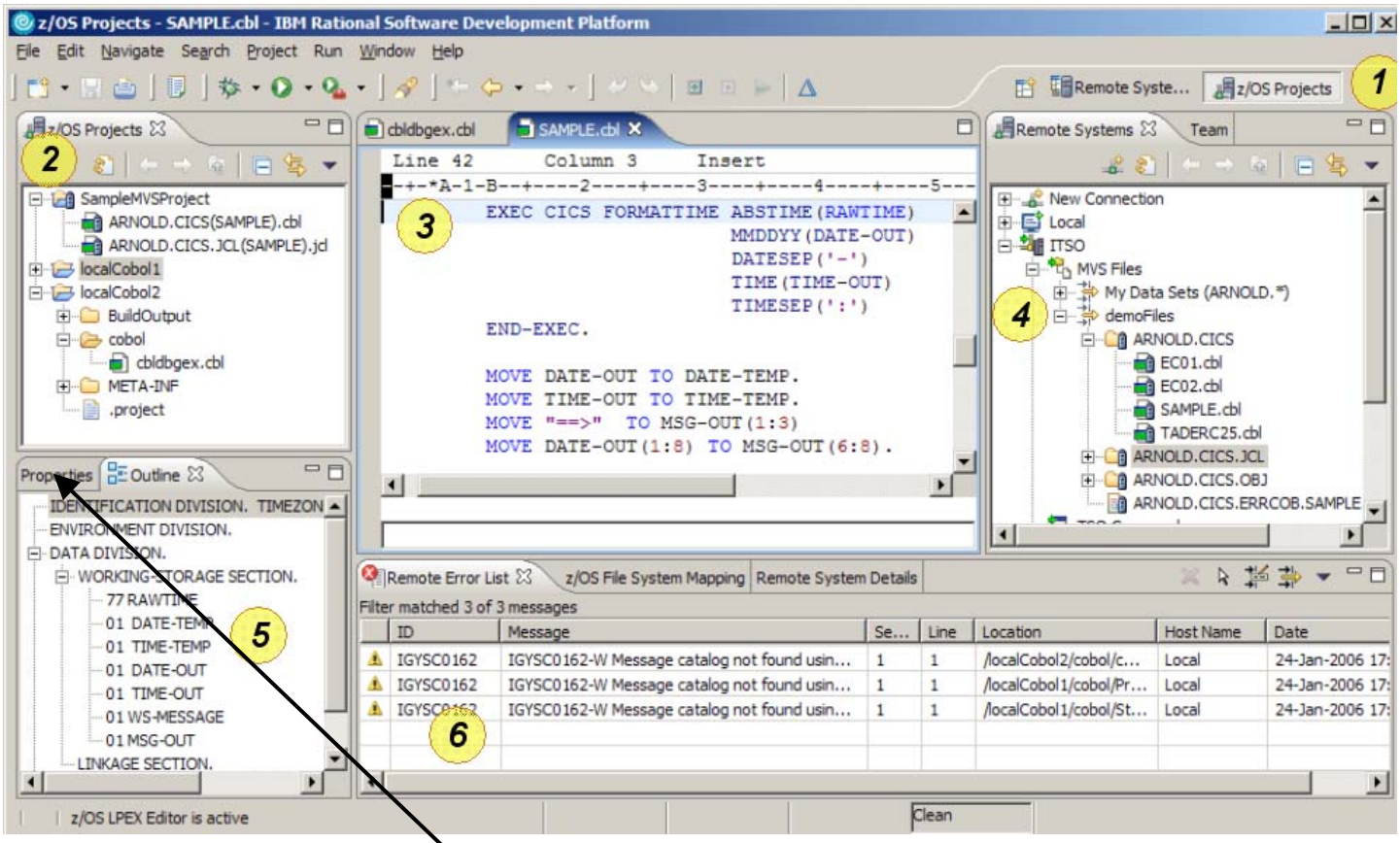


Abbildung 3.13

Dargestellt ist die exemplarische z/OS Projects Perspective. Sie besteht aus 6 Views:

1. **Short-cut Icons zu anderen Perspektiven,**
2. **z/OS Projects view (local view),**
3. **Editor,**
4. **Remote System view,**
5. **Outline view,**
6. **Tasks view**

Weil der Bildschirm nie groß genug ist, werden einige der Views wie Registrierkarten von anderen Views überlagert und verdeckt. Mit Hilfe von Tabs kann man einen verdeckten View sichtbar machen.

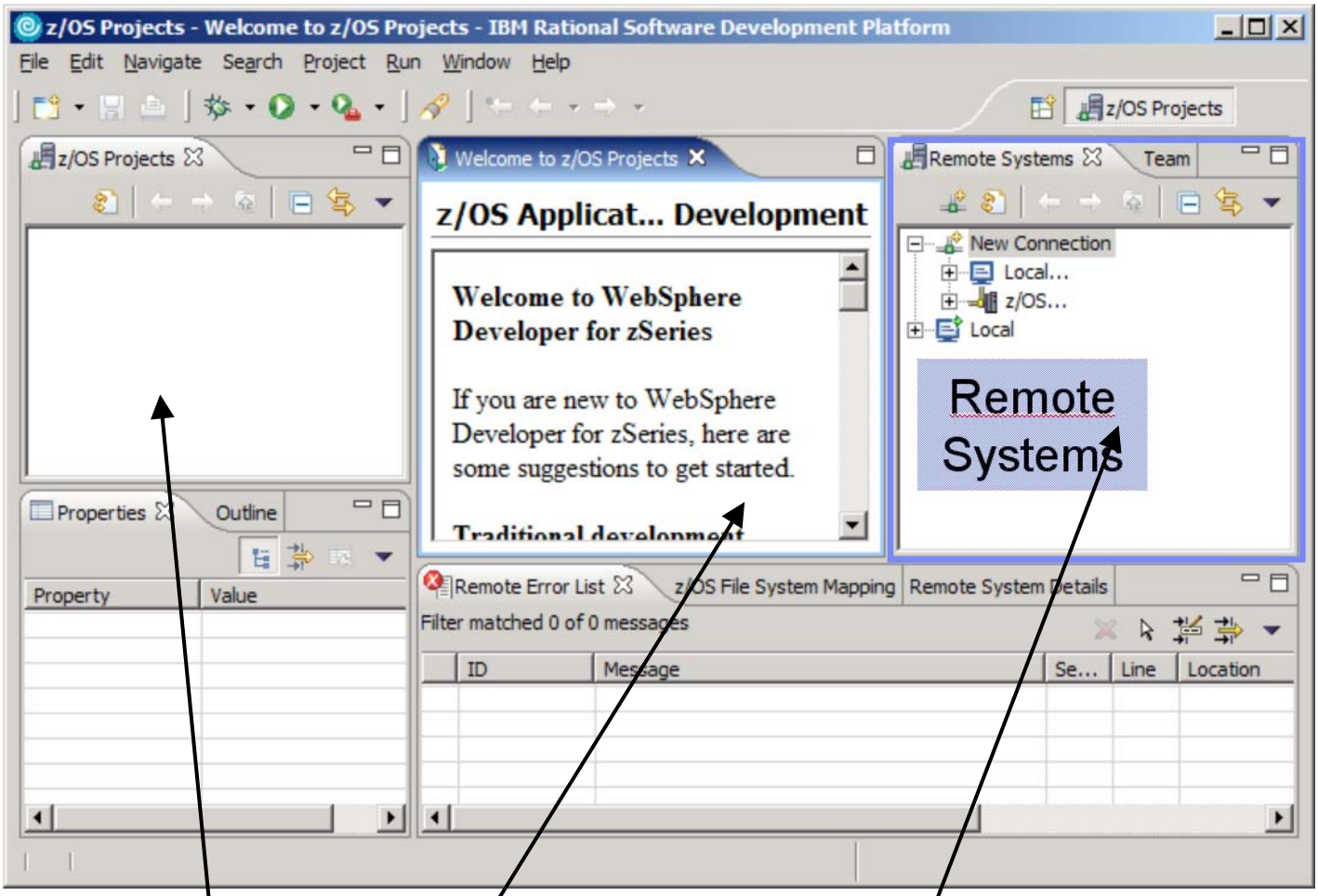


Abbildung 3.14

Für die Entwicklung von Mainframe Anwendungen ist der Remote Systems View (in der Standard „z/OS Projects“ Perspektive immer rechts oben angeordnet) von besonderem Interesse. Der Remote Systems Viel enthält typischerweise etwas ähnliches wie das ISPF DLIST Panel, und ermölicht einen Zugriff auf alle Data Sets, welche Sie unter Ihrer Prakxxx User ID auf unserem Mainframe Rechner angelegt haben.

Das z/OS Projects View enthält normalerweise Cobol Files , die Sie unter z/OS gerade bearbeiten. Diese Files sind unter RDz als Windows Files gespeichert. Sie können derartige Files mit der Maus und Drag und Drop zwischen den z/OS Projects View und dem Remote Systems View hin- und herziehen und auf diese Art kopieren.

Das mittlere Fenster enthält normalerweise einen Editor, mit dem Sie Ihren Cobol Quelltext (oder Ihr JCL Script) bearbeiten.

Insgesamt besteht ein überraschend großes Maß an Flexibilität.

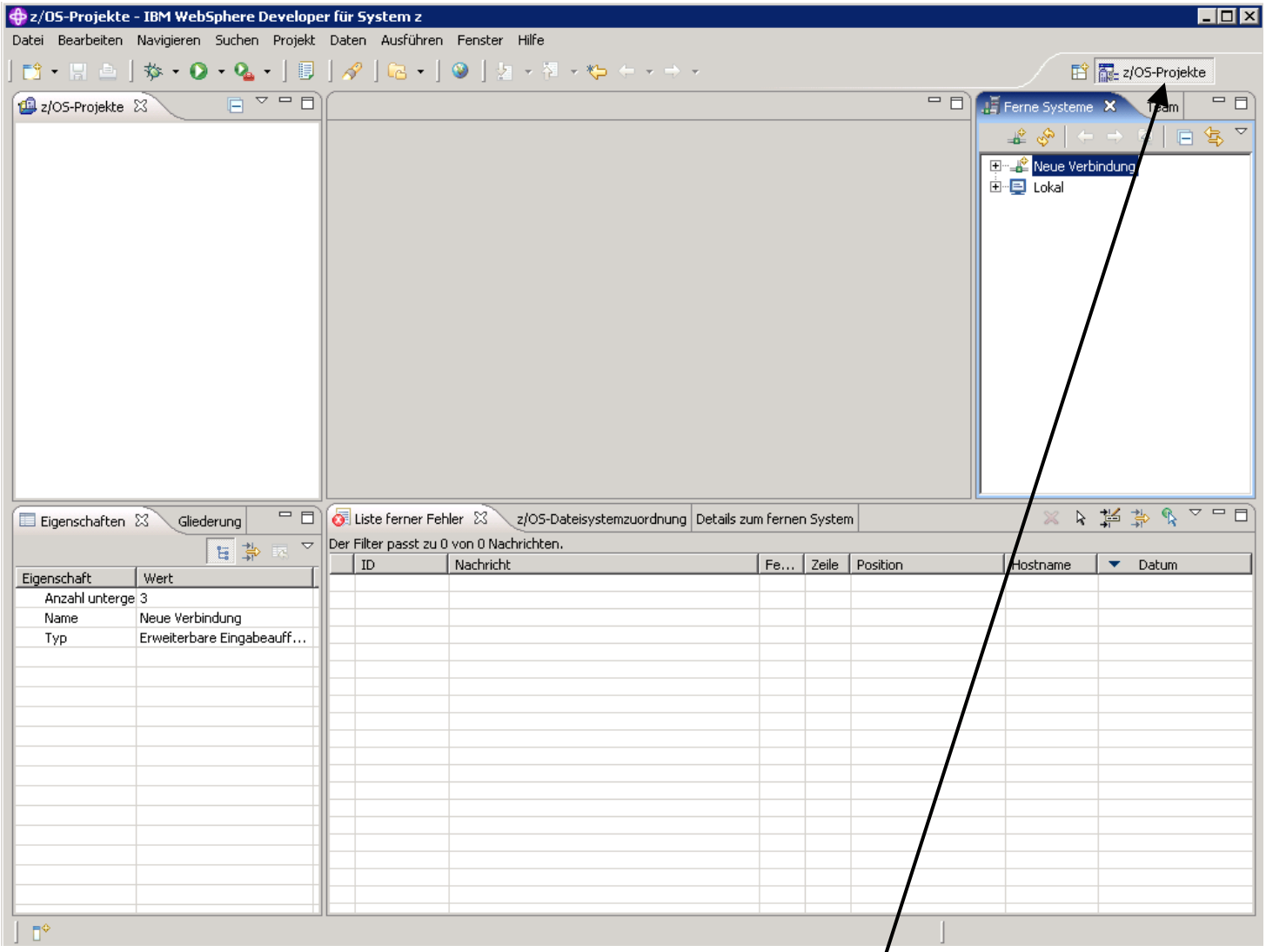


Abbildung 3.15

Normalerweise ist die z/OS Projects Perspective beim Starten von RDz bereits enabled. Falls nicht, müssen Sie das Enable nachholen. Hierzu .....

Selbst-Test

- **Was ist der Unterschied zwischen dem z/OS Projects View und dem Remote Systems View ?**

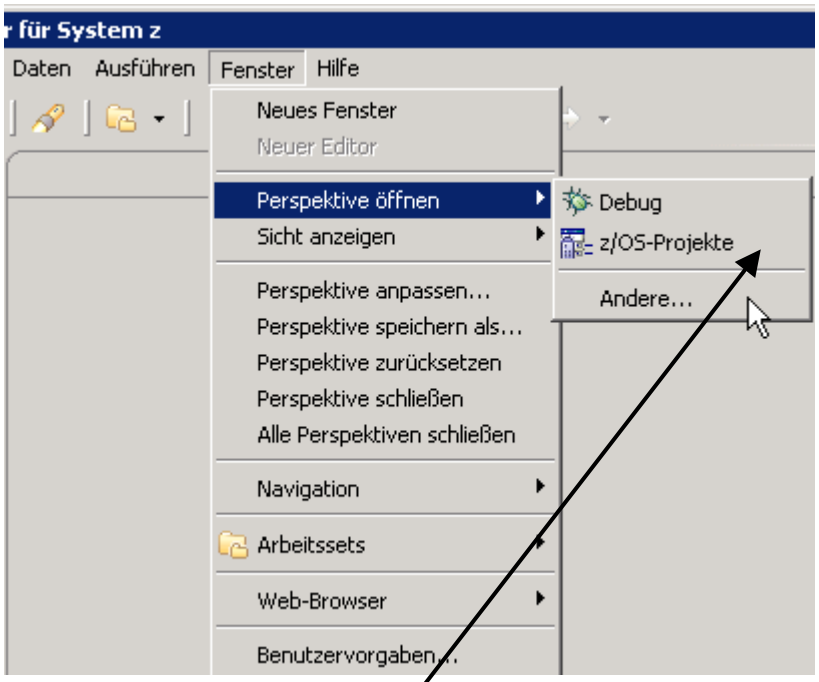


Abbildung 3.16

Fenster → Perspektive öffnen → z/OS Projekte, 1k .

4. Perspektiven und Views

4.1 Views

Workbench Tools sind in kleinen Bereichen angeordnet, die sich wie Registrierkarten verhalten und als“Views” bezeichnet werden. Views sind kleine Fenster, welche Datei oder Projekt Information wiedergeben oder einen Zugriff auf RDz Funktionen ermöglichen.
Die sich nach dem Starten von RDz geöffnete Arbeitsumgebung kombiniert mehrere verschiedenartige Views. Eine Überlagerung mehrerer Views ist möglich; nur der oberste View ist dann zu sehen. Durch 1k auf den Tab einer Registrierkarte (View) kann diese sichtbar gemacht werden.

Die Kombination aller angezeigten Views (Art und Position der Views) heißt Perspektive.

Es können sogar mehrere Perspektiven gleichzeitig geöffnet sein. In einem geöffneten RDz-Fenster wird jedoch genau eine der geöffneten Perspektiven angezeigt.

Die z/OS Projects Perspective besteht aus einer großen Anzahl von Views. Von diesen müssen Sie nur eine Untermenge kennen, um mit RDz produktiv arbeiten zu können. Sies sind spezifisch:

- **Remote Systems View**
- **z/OS Projects View**
- **COBOL Source Editor**
- **Properties View**
- **Outline View**
- **Remote Error List View**
- **Perform Hierarchy View**

14.1.1 Schließen eines Views

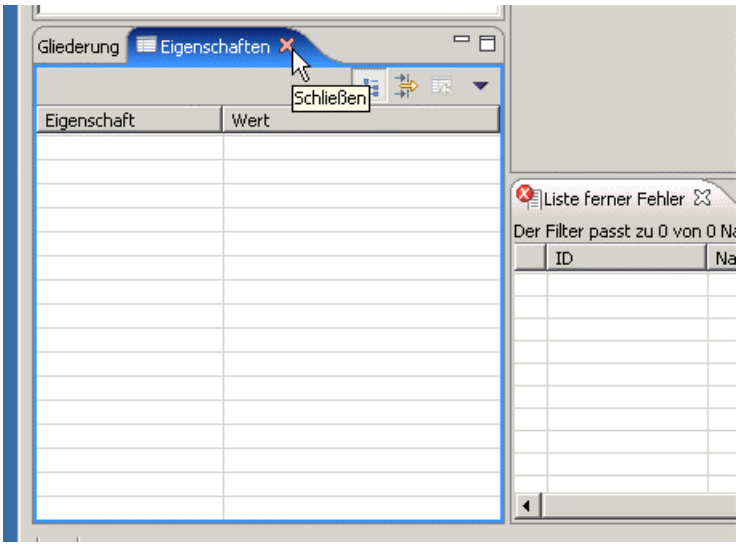


Abbildung 4.1

Eine View kann durch Klick auf das Kreuz der jeweiligen Registrierkarte geschlossen werden:

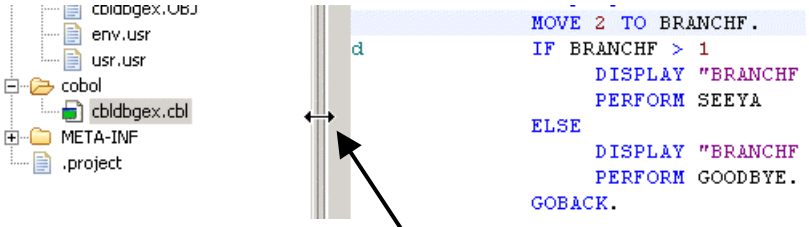


Abbildung 4.2

4.1.2 Verändern der Breite und Höhe von Views

Views kann man auch in Breite und Höhe seinen Bedürfnissen anpassen. Dazu sind einfach die jeweiligen Ränder entsprechend zu ziehen.

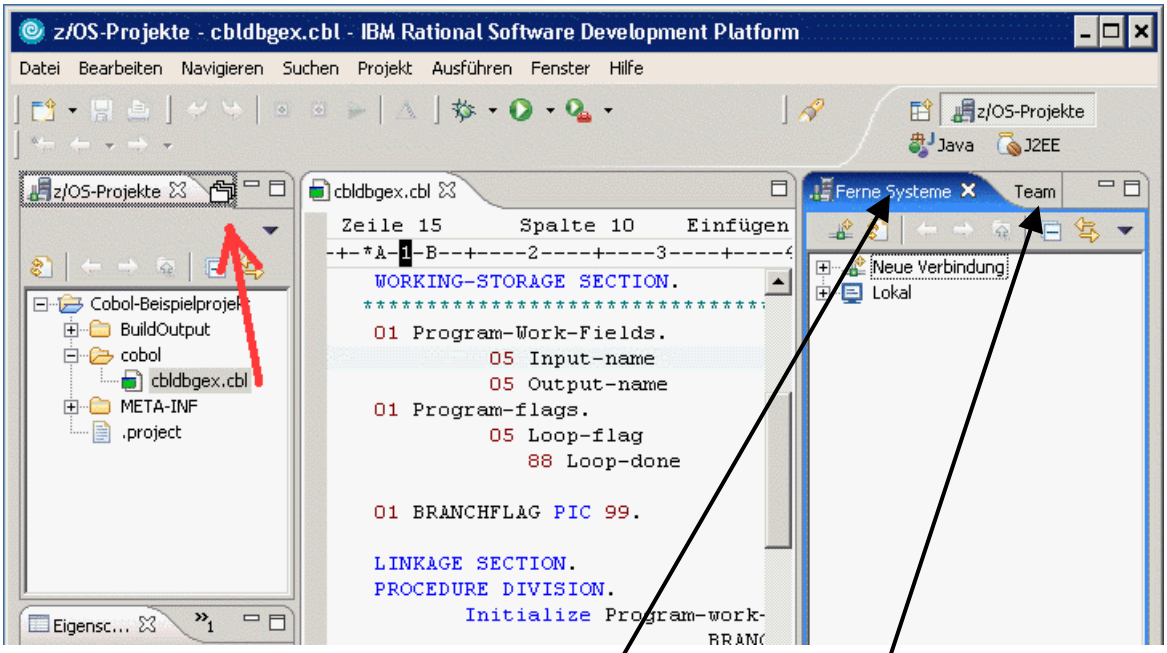


Abbildung 4.3

4.2. Vergrößern des Cobol-Programm-View

Werden die Views "Ferne Systeme" (remote Systems) und "Team" gerade nicht benötigt, können diese per drag and drop auf die linke Seite – unter die "z/OS-Projekte"-View – gezogen werden. So erhält man mehr Platz für das Editieren des Cobol-Programm-Codes.

Die Views können nur einzeln hinübergezogen werden. Um z.B. die View "Ferne Systeme" (remote Systems) hinüberzuziehen, klickt man auf "Ferne Systeme", hält den Klick und zieht. Der Cursor verwandelt sich in einen kurzen dicken schwarzen Pfeil und an einer möglichen Ablagestelle weiter zu . Wird die Maustaste losgelassen, wird der View eingefügt.

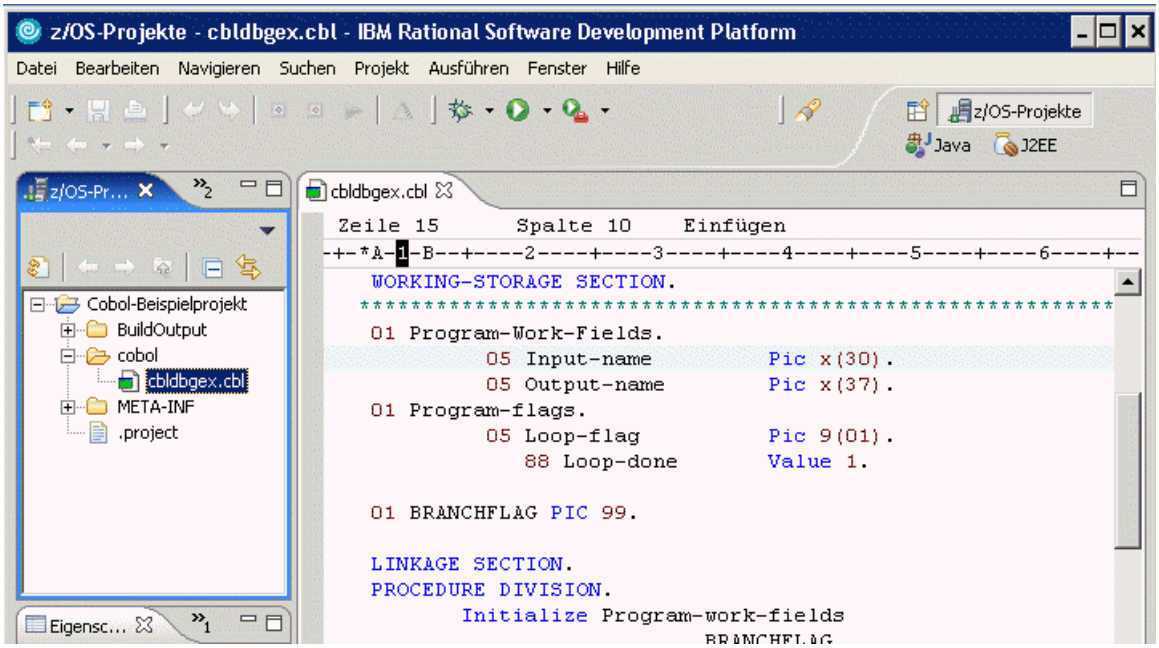


Abbildung 4.4

Dies ist das Ergebnis: Im Gegensatz zu der Abbildung 2.3 ist der Platz für das Editieren des Cobol-Programms in Abb. 2.4 größer geworden.

4.3. Zurücksetzen einer Perspektive in ihren Default-Zustand

Möchte man zum Standard-Aussehen einer Perspektive zurück, also u.a.

- **Manuell geschlossene Views wieder öffnen**
- **Views auf ihre Standardbreiten und -höhen zurücksetzen,**

dann ist das wie folgt möglich:

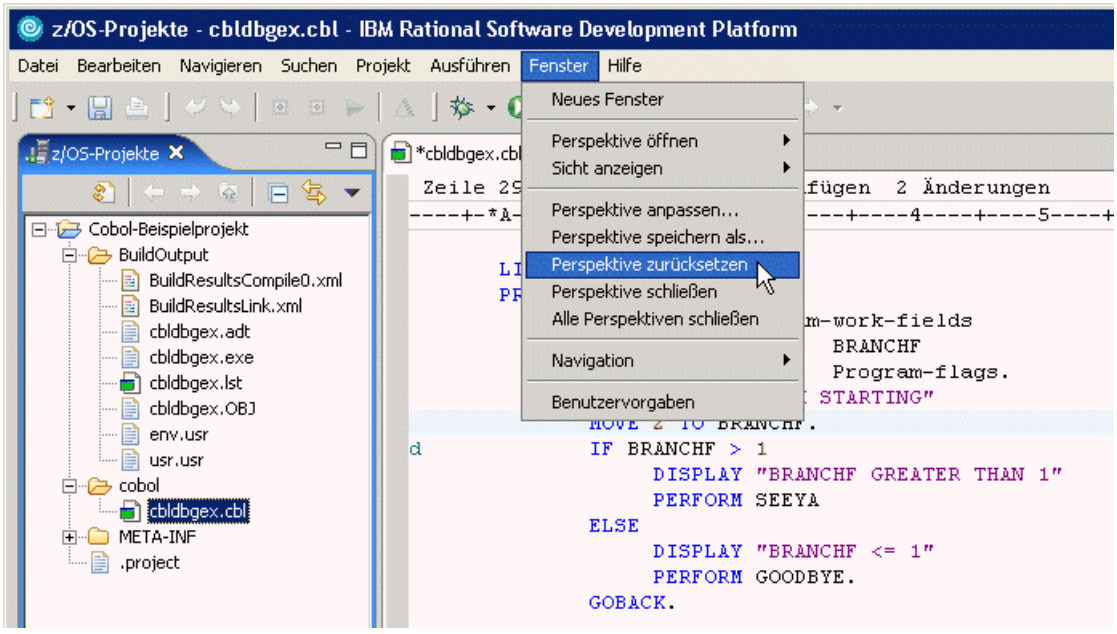


Abbildung 4.5

4.4. Wechseln zwischen verschiedenen geöffneten Perspektiven

Es ist möglich, mehrere Perspektiven zu öffnen. Doch nur eine wird jeweils in einem RDz-Fenster angezeigt.

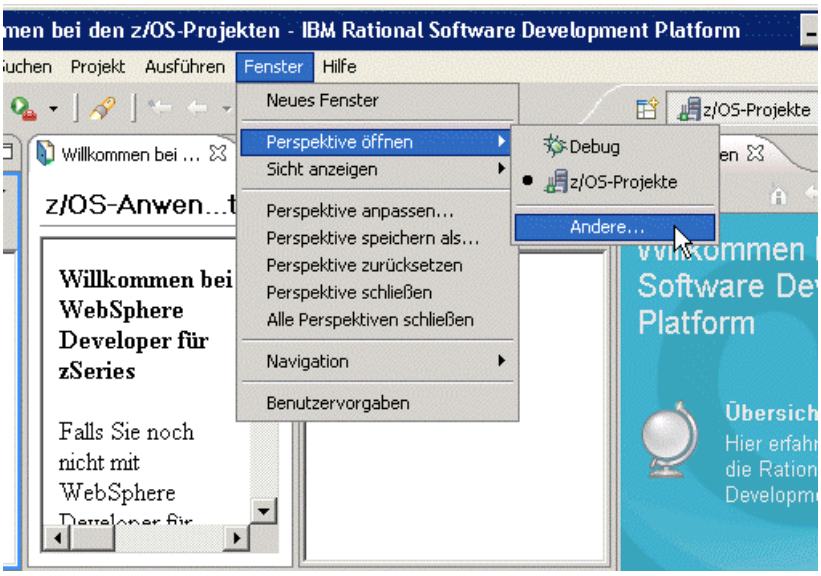


Abbildung 4.6

Neben der Perspektive "z/OS-Projekte" gibt es eine Reihe weiterer Perspektiven.
Benötigt man neben dieser Perspektive z.B. noch die "Java"-Perspektive, so kann diese ebenfalls geöffnet werden:

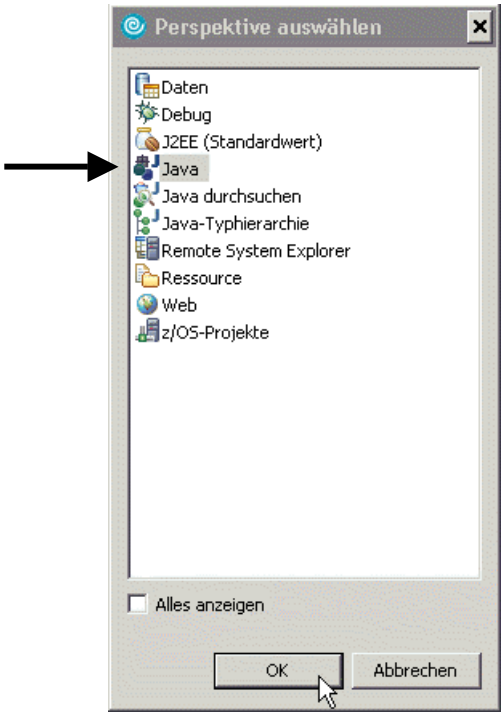


Abbildung 4.7

Klick auf Java

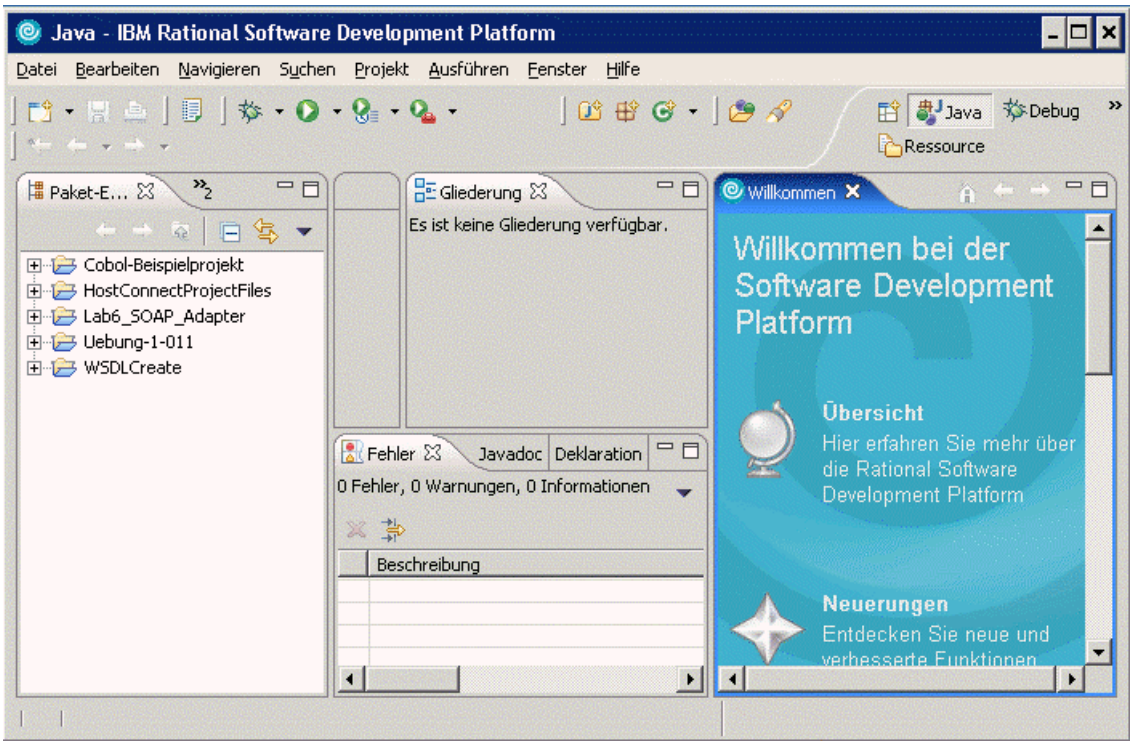


Abbildung 4.8

Nach einigen Sekunden erscheint die neue Perspektive, die "Java"-Perspektive.

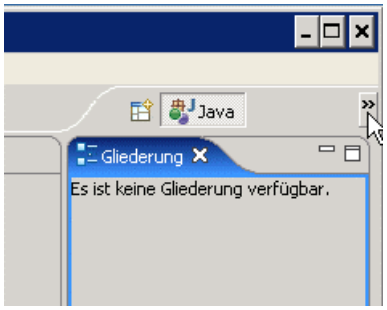


Abbildung 4.9

**Es ist nun möglich, zwischen beiden geöffneten Perspektiven zu wechseln.
Dazu ist, wie Abbildung 2.9 zeigt, auf das entsprechende Feld zu klicken.**

Geöffnete Perspektiven lassen sich auch wieder schließen:

**"Fenster → Perspektive schließen" schließt die gerade angezeigte Perspektive.
"Fenster → Alle Perspektiven schließen" schließt alle zur Zeit geöffneten Perspektiven.**

Selbst-Test

- **Müssen Sie unbedingt die z/OS Perspektive für die Entwicklung von z/OS Anwendungen benutzen ?**

5. RDz beenden und Ausloggen

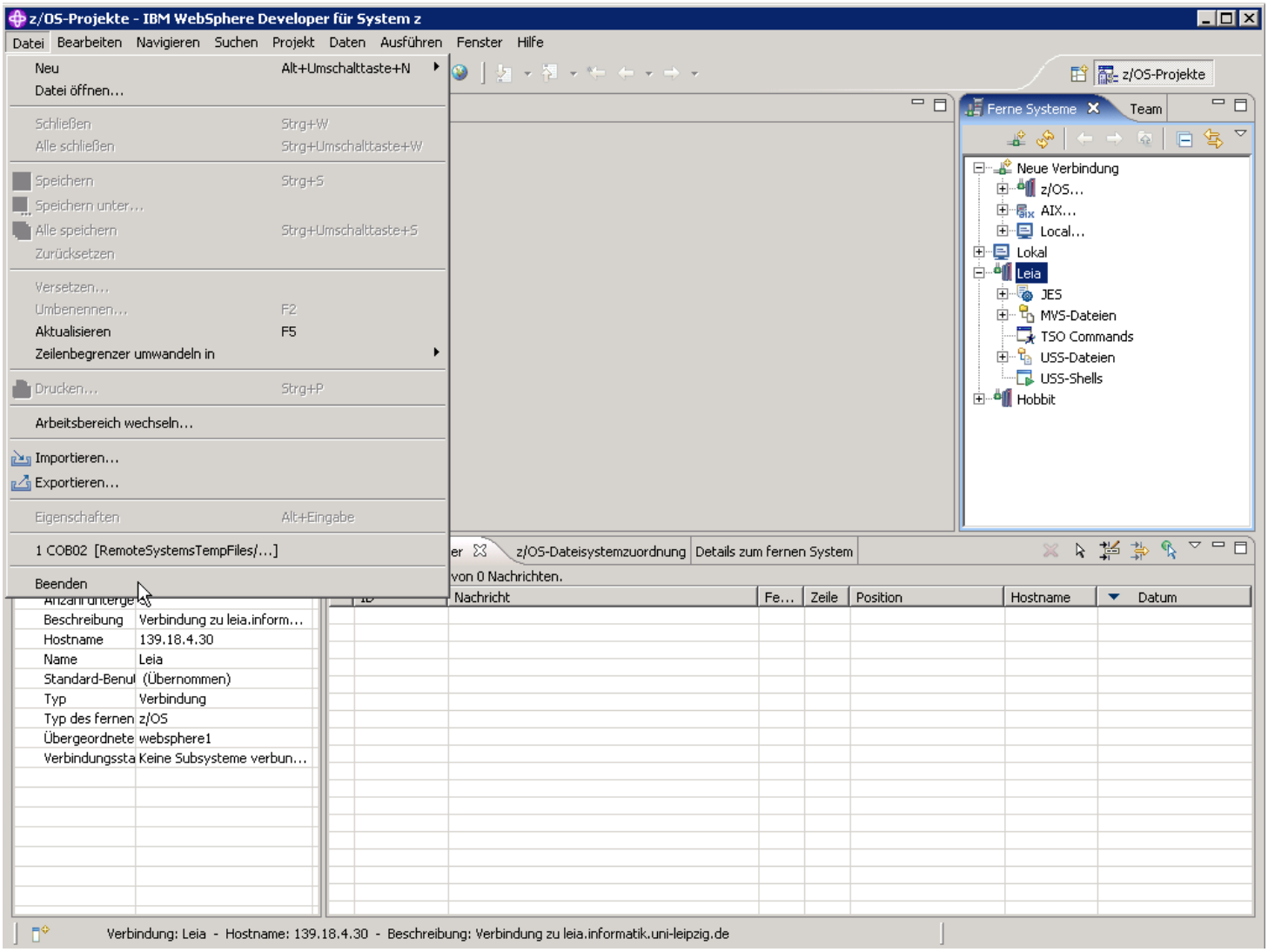
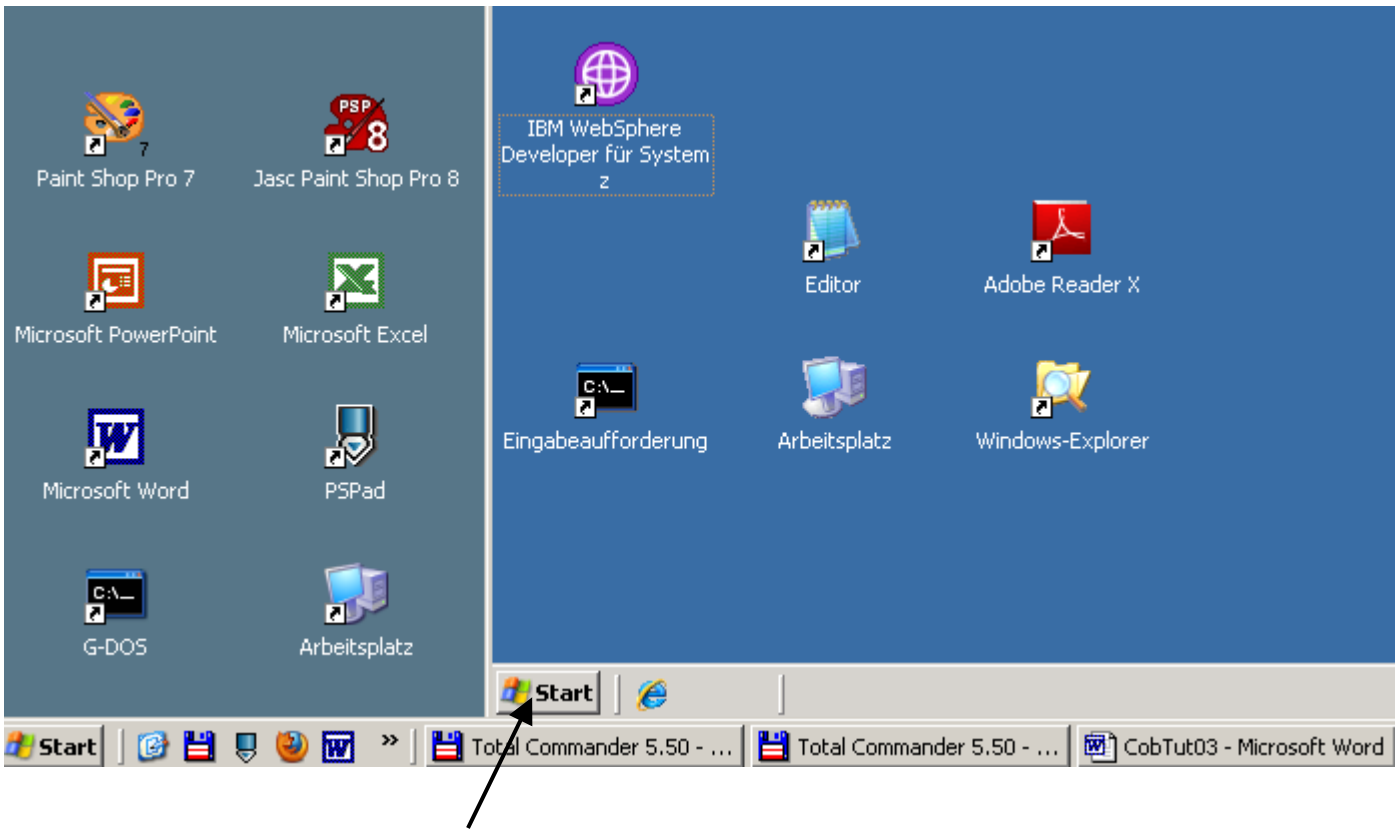


Abbildung 5.1

RDz wird wie folgt beendet: Datei → Beenden



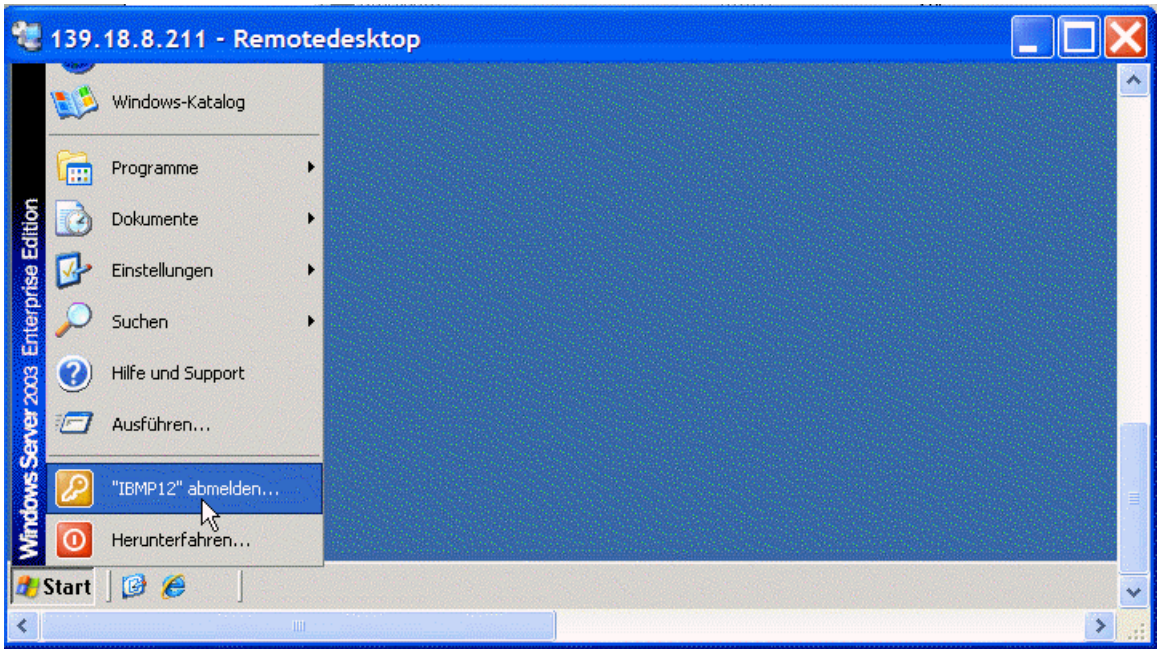


Abbildung 5.2

Nun ist noch ein Abmelden vom virtuellen entfernten Windows-Server erforderlich:

START → "IBMP<xx> abmelden..."
Es ist noch die Frage zu beantworten, ob man sich wirklich abmelden möchte. Hierzu ein Klick auf den Button "Abmelden".

Das Fenster mit der Remote Desktop Verbindung wird geschlossen.

Das war es, Sie haben RDz Tutorial 01 erfolgreich abgeschlossen. In dem nächsten Tutorial werden wir unter Windows ein Cobol Programm entwickeln und austesten.

Ende

RDz Cobol Tutorial 02
Local COBOL

Dieses Tutorial ist der zweite Teil einer Reihe. Es wird Sie durch die Schritte zur Verwendung des z/OS Application Development Komponente von WebSphere Developer for System z Version 7.0 mit lokalen Systemen führen. In diesem Tutorial werden wir mit einem Sample COBOL-Programm zu arbeiten, einschließlich der Bearbeitung, Kompilierung und Debugging eines COBOL-Programms. Nach Beendigung des Tutorials sollten Sie mit den grundlegenden RDz Workstation Einrichtungen vertraut sein, und sollte Kenntnisse erworbenen haben, um einfache COBOL-Anwendungen mit RDz entwickeln.

Dieses Tutorial (und die folgenden Tutorials) verwenden RDz Version 7.0. Es läuft in einer VMWare Player virtuellen Maschine auf unserem RDz Server. Es wird davon ausgegangen, dass Sie bereits eine Verbindung zu unserem RDz Server hergestellt haben (siehe Tutorial01).

Es existieren mehrere Cobol Dialekte. "IBM COBOL for Windows" (die Cobol Compiler Komponente von RDz) ist zu 100,00 % compatible mit dem „IBM Enterprise COBOL“ Compiler für z/OS (siehe http://www-01.ibm.com/software/awdtools/cobol/zos/). Es existieren zahlreiche Unterschiede zu den Micro Focus und Fujitsu (aka Alchemy) COBOL Compilern, die häufig in Windows Umgebungen eingesetzt werden.

Übersicht:

In dem Abschnitt 2 laden Sie in Ihr Projekt ein Cobol Beispiel Programm, welches das Schreiben eines eigenen Cobol Programms erspart.

Abschnitt 3 erläutert die vielen Verbesserungen und Erweiterungen, die RDz gegenüber dem ursprünglichen TSO ISPF Editor anbietet. Wenn Sie nicht wollen, müssen Sie diese nicht benutzen. Die Erweiterungen machen aus ISPF einen modernen, State of the Art Editor.

Abschnitt 5 erläutert die modernen Debug Funktionen, welche die TSO Funktionen deutlich erweitern und ergänzen.

Die ursprüngliche Version dieses Tutorials wurde von Frau Isabel Arnold in einer z / OS Sommer-Klasse an der Universität Hamburg vorgestellt und von Herrn Niels Michaelsen übernommen. Sie wurde geändert und angepasst durch Herrn Matthias Beyerle an der Fakultät für Informatik der Universität Tübingen.

Dieses Tutorium (und die folgenden Tutorials) sind gehostet auf dem z/OS 1.8 System leia.informatik.uni-leipzig.de oder 139.18.4.30 der Abteilung Technische Informatik der Universität Leipzig und hobbit.cs.informatik.uni-tuebingen.de oder 134.2.205.54 der Abteilung Technische Informatik der Universität Tübingen. Die RDz Installation auf diesem System wurde von Herrn Uwe Denneler, Elisabeth Puritscher und Herr Martin Benjamin Storz von IBM durchgeführt und unterstützt von Frau Martina Koederitz, Herb Kircher und Herr Andresa Hermelink, ebenfalls IBM. Lokale Unterstützung erfolgte durch Herrn Frank Güttler an der Universität Leipzig, sowie Andreas Nagel an der Universität Tübingen.

Dieses Tutorial verwendet die folgenden Konventionen

1k bedeutet 1 Klick mit der linken Maustaste
2k bedeutet 2 Klick mit der linken Maustaste
1kr bedeutet 1 Klick mit der rechten Maustaste

Selbst-Test

- **Findet der Compile Vorgang in diesem Tutorial auf der Workstation, auf dem RDz Server, oder auf dem z/OS System statt ?**
- **Ist der erzeugte Maschinencode (Binaries) sowohl unter Windows als auch unter z/OS lauffähig ? Nur unter z/OS ? Nur unter Windows ?**

Aufgabe: ***Erstellen Sie ein einfaches Cobol Beispiel Programm, übersetzen (compile sie es und führen es aus. Die Durcharbeitung von Abschnitt 3 ist eine Option, macht Sie aber mit moderner Software Entwicklung vertraut.***

Aufgabe: ***Als Option machen Sie sich mit den in Abschnitt 5 beschriebenen RDz Test/Debug Funktionen vertraut.***

1. Starting WDz

Erstellen Sie eine Remote Desk Top Verbindung zu unserem RDz Server, wie bereits im letzten Tutorial dargestellt.

Abb. 1.1

Starten RDz, wenn es nicht bereits ausgeführt wird

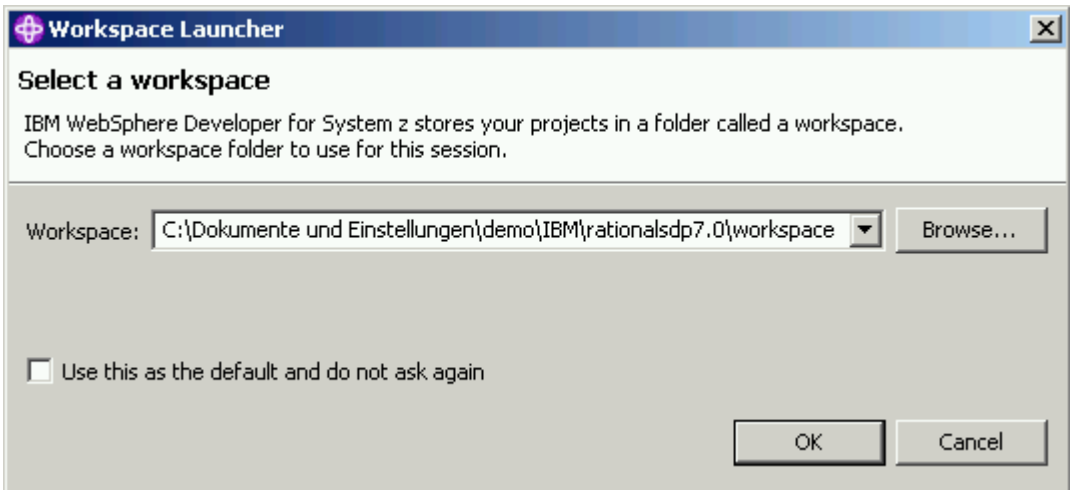


Abb. 1.2

Da auf dem remote Windows Server für jeden Benutzer eine eigene Festplatte z:\ eingerichtet wird, schlagen wir vor, dass sie statt des vorgeschlagenen Verzeichnisses den Wert „Z:\Workspace01“ benutzen.

1k auf OK, das Starten dauert ca. 1 Minute

Die Standard-Perspektive nach der Installation von RDz ist die z/OS Projects Perspektive. Um es zu öffnen machen Sie folgendes: Von der z/OS Projects Perspektive, wählen Sie in der Menüleiste Datei?-- New?--Beispiel ... Sie werden die z / OS Projects Perspektive zum Erstellen und Bearbeiten der Projekte und Projekt-Dateien auf Ihrer Workstation verwenden, auf Ihrem z/OS-System, oder in einer verteilten Umgebung. Beachten Sie, dass Perspektiven viele Ansichten (Views) haben. Um eine Ansicht zu schließen, klicken Sie auf die Schaltfläche Schließen mit einem X in der rechten oberen Ecke der Registrierkarte.

2. Laden eines Local COBOL Sample Programms

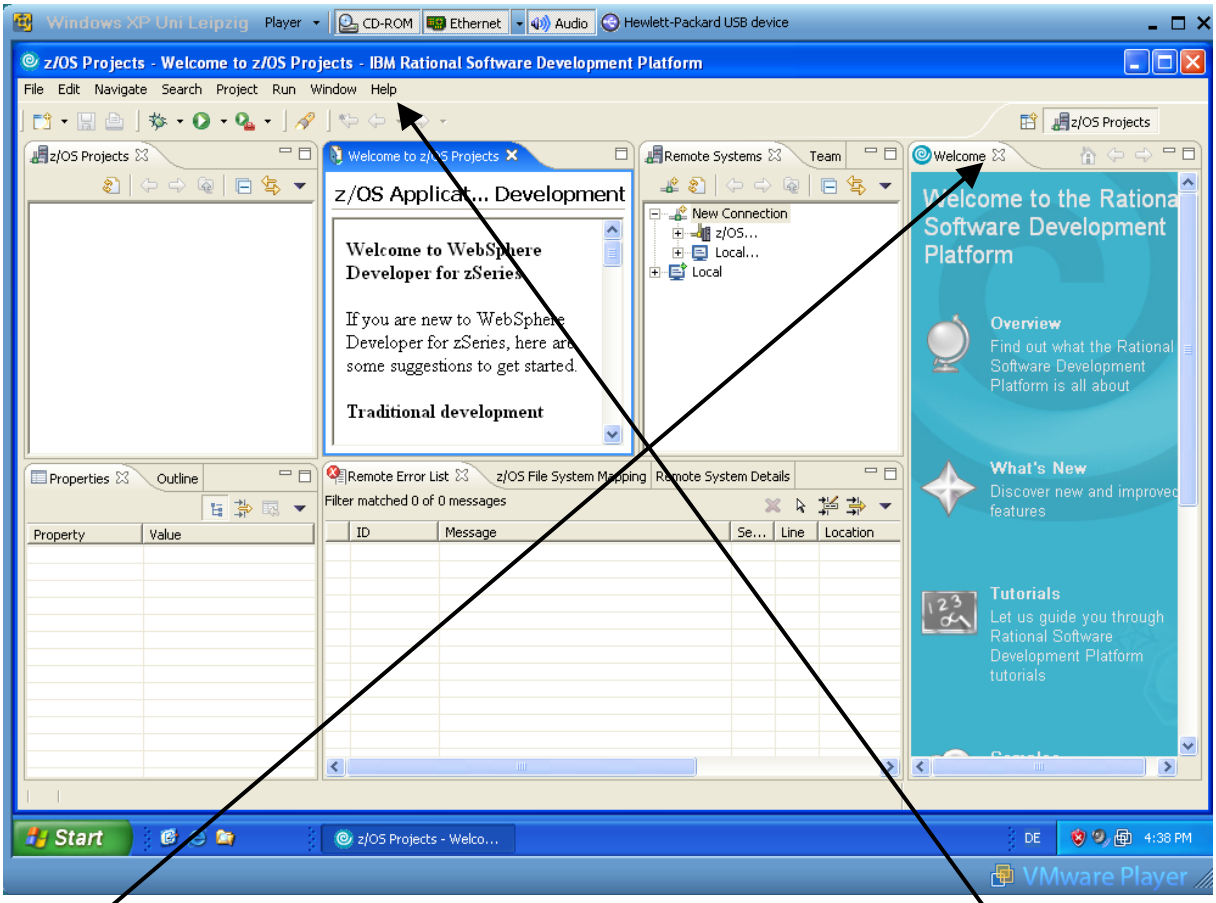


Abb. 2.1

Dies ist der RDz Welcome Screen, wie er beim erstmaligen Login erscheint.

1k auf dem Kreuz neben Willkommen auf das rechte Fenster zu schließen. Mit „Hilfe“ können Sie jederzeit neu zu erstellen.

Die Standard-Perspektive nach der Installation von RDz ist die z/OS Projects Perspektive. Sie benutzen diese zum Erstellen und Bearbeiten von Projekten und Projekt-Dateien auf Ihrer Workstation, auf Ihrem z/OS-System, oder auch in einer verteilten Umgebung. Beachten Sie, dass Perspektiven viele Views haben.

Um einen View zu schließen, klicken Sie auf die Schaltfläche Schließen mit einem X in der rechten oberen Ecke der Registrierkarte. Um es zu öffnen, klicken Sie auf Window ? Show View? Andere, und wählen Sie den gewünschte View, den Sie öffnen möchten.

Als ersten Schritt laden Sie nun ein Beispiel COBOL-Programm in Ihrem lokalen Workspace.

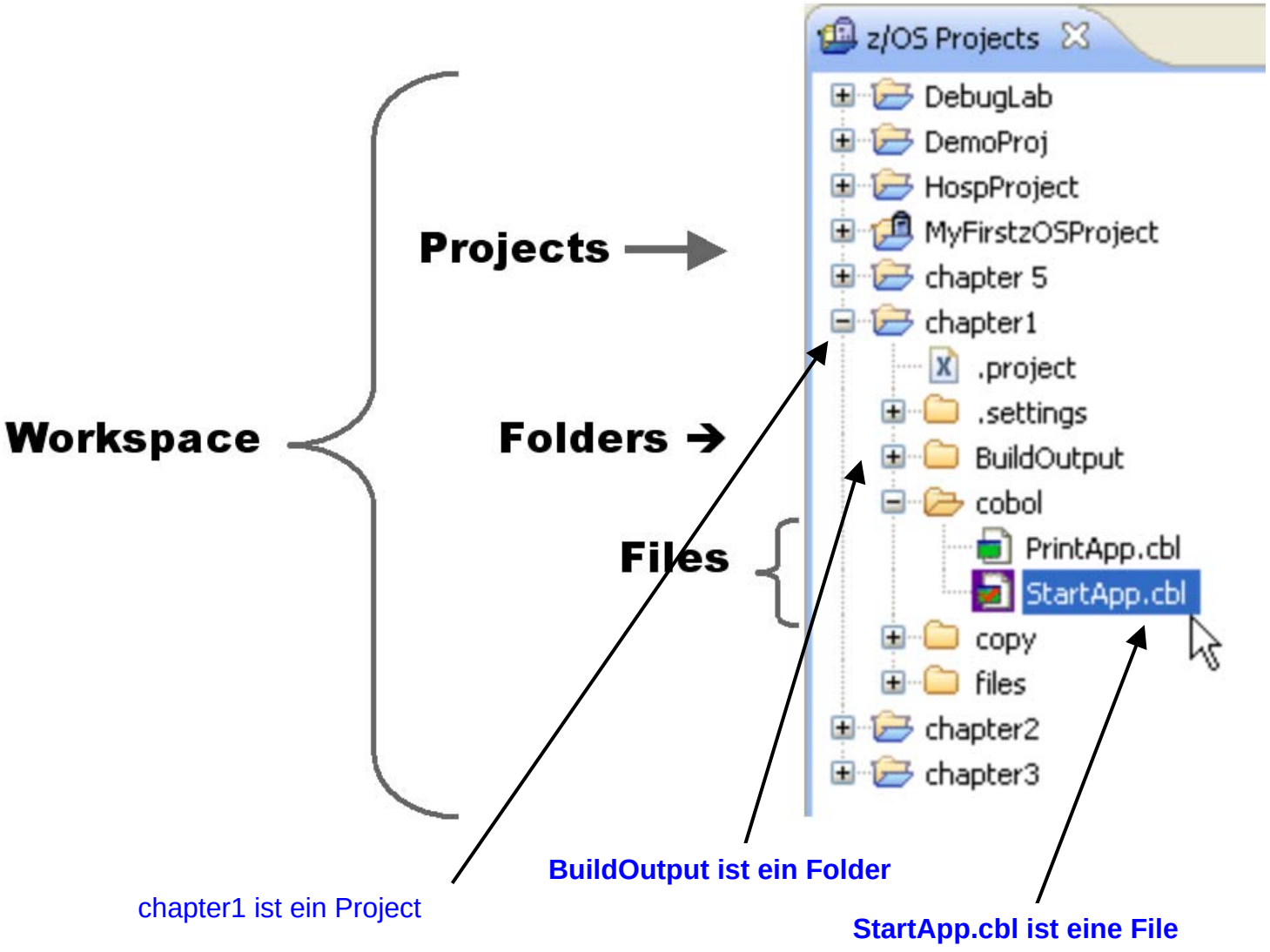
2.1 Workspace, Projects und Files

Wir werden mit unterschiedlichen COBOL Resourcen arbeiten: Pogramme, Copybooks, Compiler Listings, ausführbaren Programmen (executables), Daten Files usw.

RDz speichert, organisiert und managed die zu Ihren Projekten gehörigen Resourcen innerhalb eines Workspaces. Workspace Resourcen sind organisiert in:

- **Projekt(e)**
- **Folder**
- **Files.**

Nachdem Sie RDz gestartet und einen Workspace selektiert haben, haben Sie Zugriff auf alle Files und Folders innerhalb des Workspace. Hierzu ist der Workspace typischerweise in ein oder in mehrere Projekte organisiert:



Was ist ein Projekt, und was sind die Bestandteile eines Projektes ?

Workspace Projekte organisieren und verwalten zueinander gehörige Anwendungs-Resourcen. Projekte innerhalb eines Workspace können organisiert werden:

- **Batch Anwendung .. vs.. Online Anwendung**
- **Line of Business Anwendungen: Accounts Payable, Inventory, Claims, Manufacturing Part Assembly, usw.**
- **Common – oder shared Projekte, die Daten oder Record Definitionen enthalten können, welche von mehreren unterschiedlichen Anwendungen gemeinsam genutzt werden.**

Es existieren eine Reihe von spezifischen RDz Projektarten, mit denen wir uns später noch beschäftigen werden. Beispiele sind z/OS Local Projects, MVS Subprojects, usw.

Projekte enthalten Build Configuration Files, auch als Property Files bezeichnet. Diese spezifizieren Generierungs-Optionen für Ihre Projekt Komponenten ihres Projektes.

Projekte können aus mehreren Foldern bestehen.

z/OS COBOL Projekte enthalten typischerweise die folgenden high-level Folder:

- **Cobol,**
- **Build Output,**
- **copy**

Was befindet sich in dem Cobol Folder (Directory) ?

\cobol ist der Default Folder (der highest level folder) innerhalb eines Projektes, unter dem alle COBOL Ressourcen (Quell Programme) organisiert sind.

Diese COBOL Resourcen können z.B. Sub-Folders einschließen:

```
cobol
    batch
        program1.cbl
        program2.cbl
        …
    online
        program3.cbl
        …
```

oder enthalten alle COBOL Quellprogramm Files:

```
cobol
    program1.cbl
    program2.cbl
    …
```

Was befindet sich in dem BuildOutput Folder (Directory) ?

\BuildOutput\ ist der Default Folder (der highest level folder) innerhalb eines Projektes, unter dem alle übersetzten (compiled) COBOL Programme organisiert sind.

Diese Resourcen enthalten:

- **.OBJ – object modules**
- **.exe – COBOL executables – können ausgeführt oder debugged werden**
- **.adt – an internal-system file used by RDz when you do source-level debugging produced by a Compiler Directive)**
- **.lst – listing file**
  - **- Shows highest error condition**
  - **- Sorted XRef of COBOL variables**
  - **- Other program info**

Der Folder copy enthält Copy Files, auch als Copybooks bezeichnet. Ein Copybook ist eine Bezeichnung für eine Dateibeschreibung oder eine Routine, die mehrfach in einem Cobol Programm und dessen Subroutines in identischer Form benötigt wird. Ein copy Statement bewirkt etwas ähnliches wie ein include Statement in C/C++. Ein Beispiel Programm finden Sie unter http://www.csis.ul.ie/cobol/course/Copy.htm. Es ist auf den beiden folgenden Seiten wiedergegeben.

Source Text

```
      $SET SOURCEFORMAT"FREE"
IDENTIFICATION DIVISION.
PROGRAM-ID. COPYEG1.
AUTHOR.  Michael Coughlan.

ENVIRONMENT DIVISION.
FILE-CONTROL.
    SELECT StudentFile ASSIGN TO "STUDENTS.DAT"
    ORGANIZATION IS LINE SEQUENTIAL.

DATA DIVISION.
FILE SECTION.
FD StudentFile.
COPY COPYFILE1.

PROCEDURE DIVISION.
BeginProg.
   OPEN INPUT StudentFile
   READ StudentFile
      AT END SET EndOfSF TO TRUE
   END-READ
   PERFORM UNTIL EndOfSF
      DISPLAY StudentNumber SPACE StudentName SPACE
              CourseCode SPACE FeesOwed SPACE AmountPaid
      READ StudentFile
         AT END SET EndOfSF TO TRUE
      END-READ
   END-PERFORM
   STOP RUN.
```

Text in COPYFILE1

```
01  StudentRec.
    88  EndOfSF      VALUE HIGH-VALUES.
    02  StudentNumber               PIC 9(7).
    02  StudentName                 PIC X(60).
    02  CourseCode                  PIC X(4).
    02  FeesOwed                    PIC 9(4).
    02  AmountPaid                  PIC 9(4)V99.
```

Text in COPYFILE1

```
01  StudentRec.
    88  EndOfSF      VALUE HIGH-VALUES.
    02  StudentNumber                PIC 9(7).
    02  StudentName                  PIC X(60).
    02  CourseCode                   PIC X(4).
    02  FeesOwed                     PIC 9(4).
    02  AmountPaid                   PIC 9(4)V99.
```

Source Text after processing

```
      $SET SOURCEFORMAT"FREE"
IDENTIFICATION DIVISION.
PROGRAM-ID. COPYEG1.
AUTHOR.  Michael Coughlan.

ENVIRONMENT DIVISION.
FILE-CONTROL.
    SELECT StudentFile ASSIGN TO "STUDENTS.DAT"
    ORGANIZATION IS LINE SEQUENTIAL.

DATA DIVISION.
FILE SECTION.
FD StudentFile.
COPY CopyFile1.
01  StudentRec.
    88  EndOfSF      VALUE HIGH-VALUES.
    02  StudentNumber                PIC 9(7).
    02  StudentName                  PIC X(60).
    02  CourseCode                   PIC X(4).
    02  FeesOwed                     PIC 9(4).
    02  AmountPaid                   PIC 9(4)V99.

PROCEDURE DIVISION.
BeginProg.
   OPEN INPUT StudentFile
   READ StudentFile
      AT END SET EndOfSF TO TRUE
   END-READ
   PERFORM UNTIL EndOfSF
      DISPLAY StudentNumber SPACE StudentName SPACE
              CourseCode SPACE FeesOwed SPACE AmountPaid
      READ StudentFile
         AT END SET EndOfSF TO TRUE
      END-READ
   END-PERFORM
   STOP RUN.
```

Selbst-Test

- Muss ein Cobol Programm unbedingt Copybooks enthalten ?

2.2 Laden eines Beispielprogramms

Wenn erforderlich: "z/OS-Projekte"-Perspektive öffnen
Von der z/OS Projects Perspective, wählen Sie File → New → Example at the menu bar.

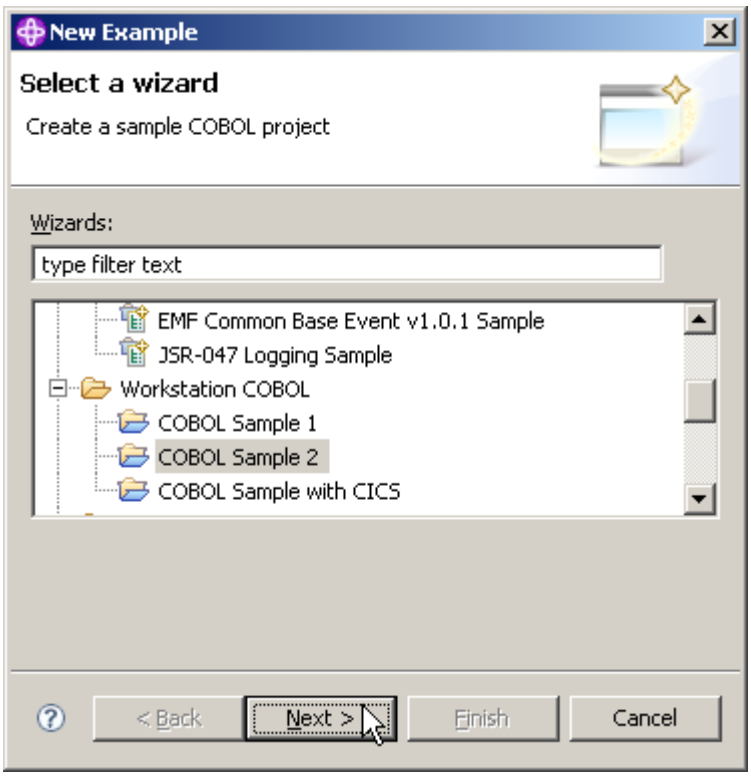


Abb. 2.2

"Workstation COBOL" erweitern

In dem Select a wizard panel, selektieren Sie Workstation COBOL - COBOL Sample 2.
1k auf Next.

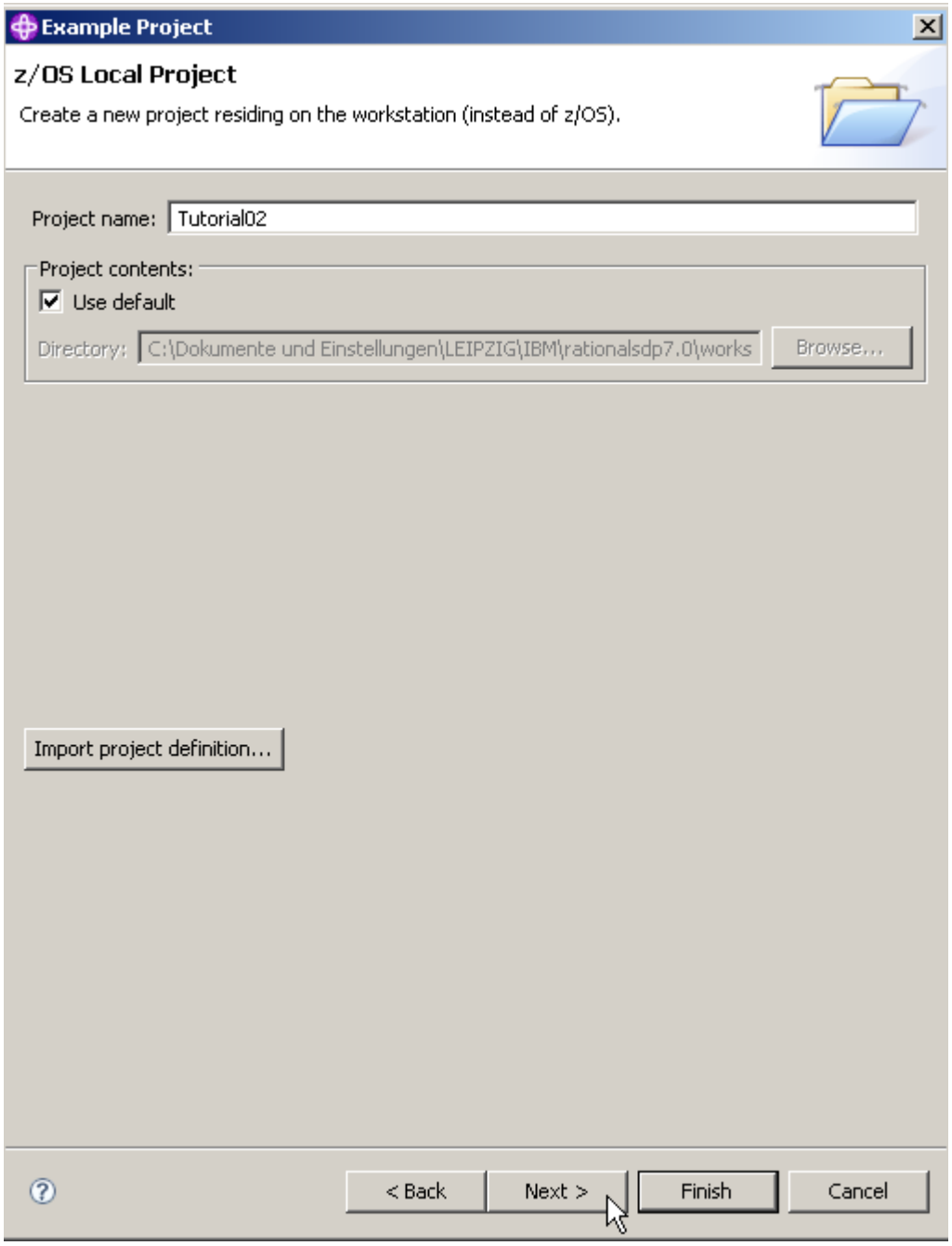


Abb. 2.3

Im z/OS Local Project Panel, geben Sie Tutorial02 als den Projekt Namen ein, setzen Sie ein Häkchen auf Use default, und click Next.

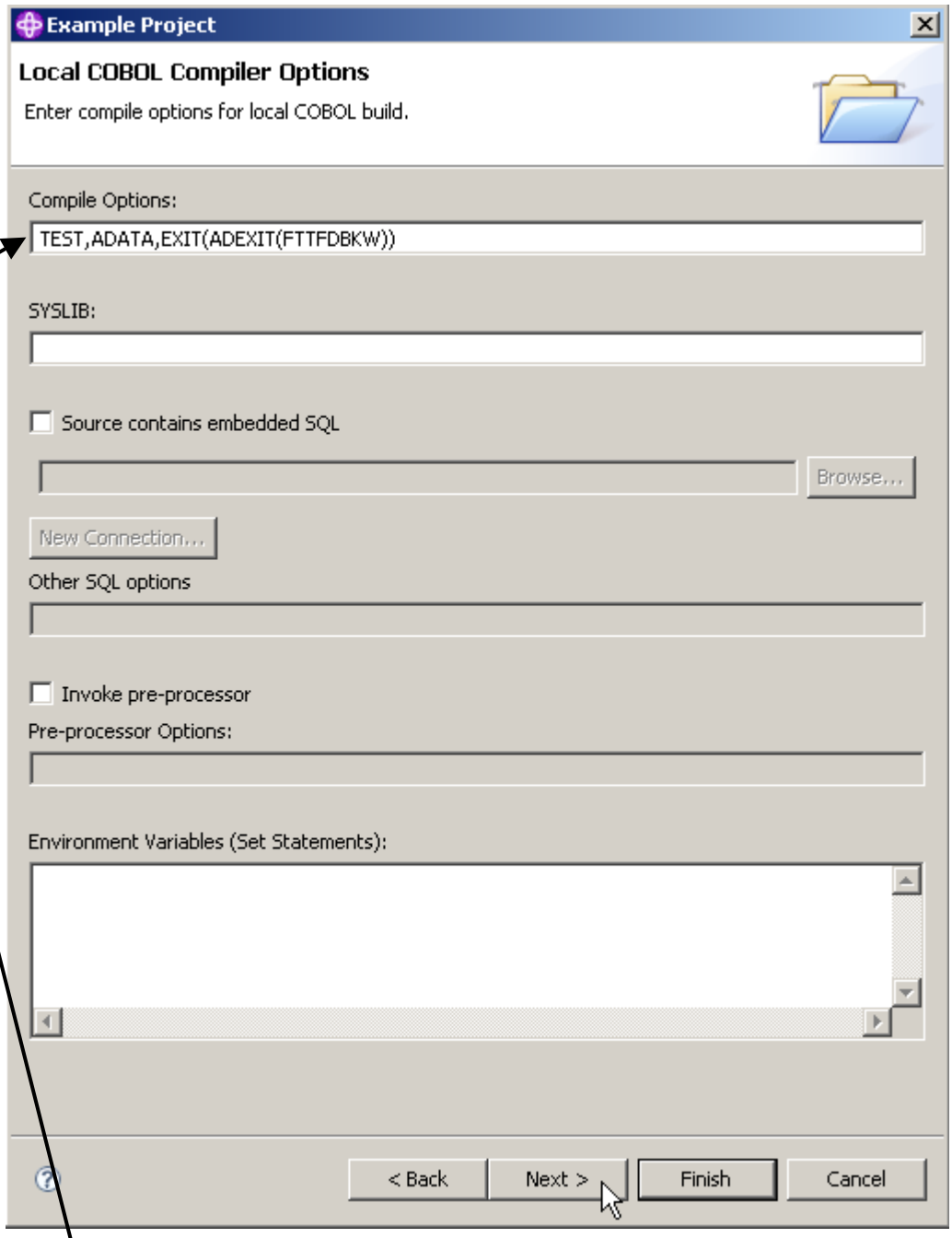


Abb. 2.4

Im Local COBOL Compiler Options Panel verifizieren Sie, dass TEST unter Compile Options spezifiziert ist. Lassen Sie die Defaults für den Rest unverändert.

Beachten Sie die default Local COBOL Compiler Options für Ihr COBOL Programm. Die TEST Compiler Option beinhaltet Sub-Optionen für die Definition von debugging Funktionen.

Verifizieren Sie, dass TEST spezifiziert ist unter Compile Options. Der Screen sollte aussehen wie dargestellt. Click Next.

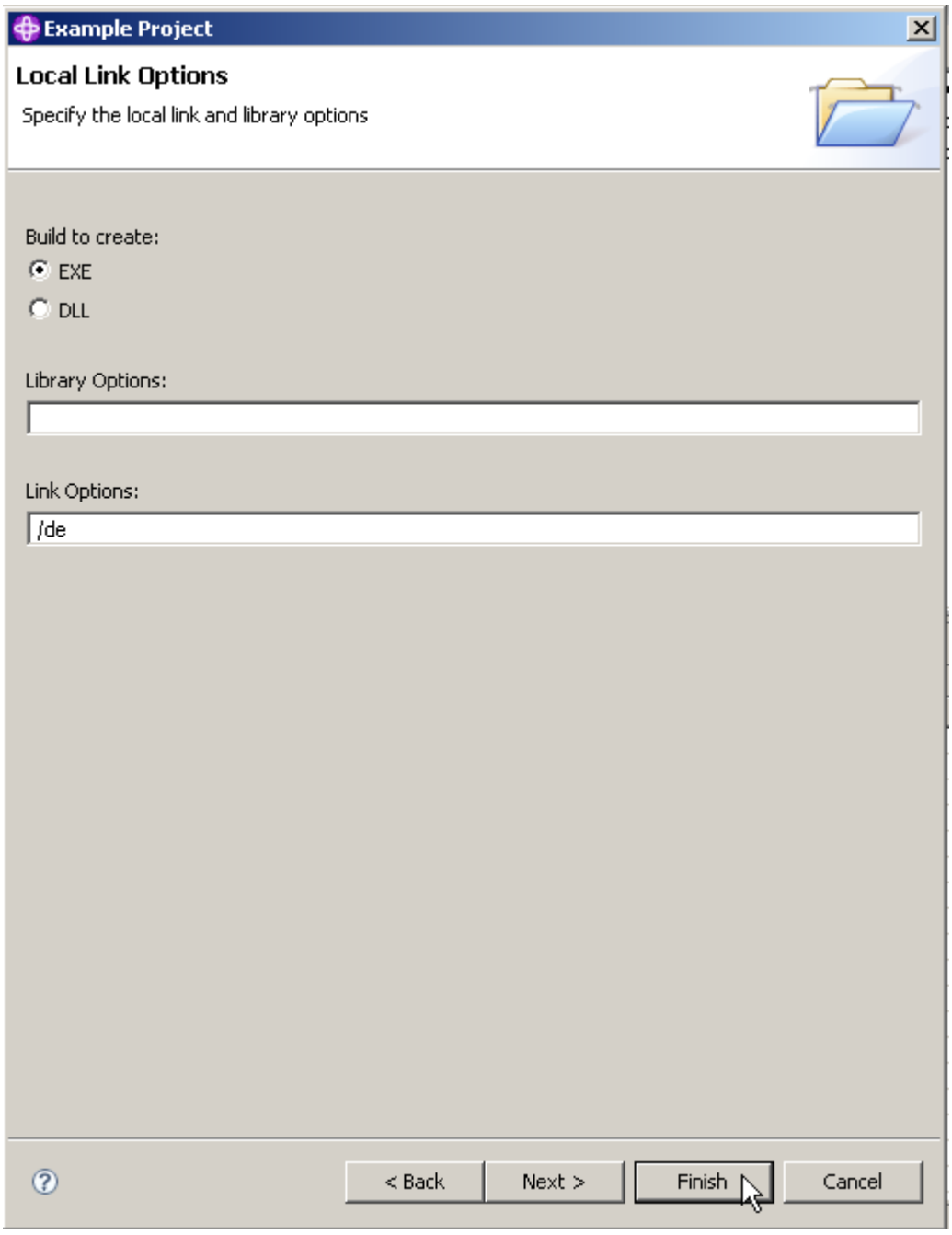


Abb. 2.5

Auf dem lokalen Link-Optionen-Panel sind die Link-Parameter festzulegen. Die Option für Build sollte EXE sein. Dies bedeutet, für ein COBOL-Programm aus diesem Projekt wird eine exe-Datei anstelle einer dll-Datei generiert.

Beachten Sie auch die Option /de. Dies bedeutet, Ihr Programm wird Debugging-Informationen enthalten. Klicken Sie auf Finish, um den Vorgang abzuschließen.

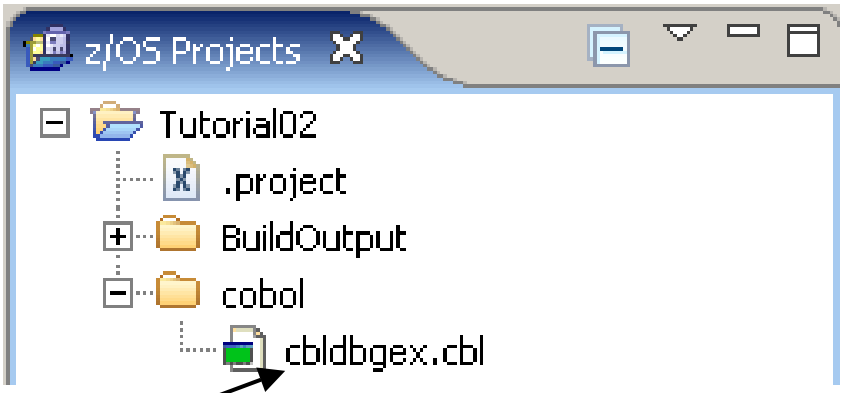


Abb. 2.6

Nach einigen Sekunden ist das Ergebnis links oben in der z/OS-Perspektive sichtbar Das cbldbgex.cbl Cobol source Program wurde geladen.

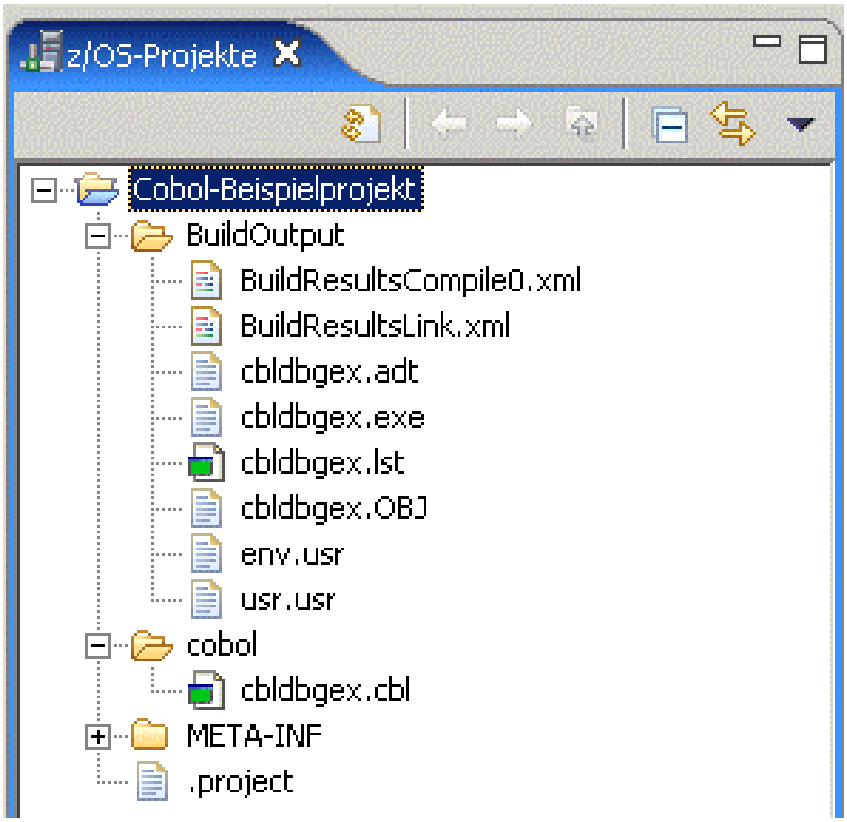


Abb. 2.7

Nach Erweiterung von "BuildOutput" und "cobol" sieht das Ergebnis wie folgt aus:

Beispiel-Cobol-Programm-Code: "cbldbgex.cbl"
Daraus erzeugte ausführbare EXE-Datei: "cbldbgex.exe"

3. Arbeiten mit einem localen COBOL Programm

In diesem Kapitel erfahren Sie, wie Sie Quellcode bearbeiten und wie Sie eine Syntaxprüfung mit dem RDz COBOL-Editor durchführen.

3.1 Bearbeiten eines COBOL-Quellprogramms

Stellen Sie sicher, dass die z / OS Projects Perspektive geöffnet ist (Window → open Perspective → z/OS Projects).

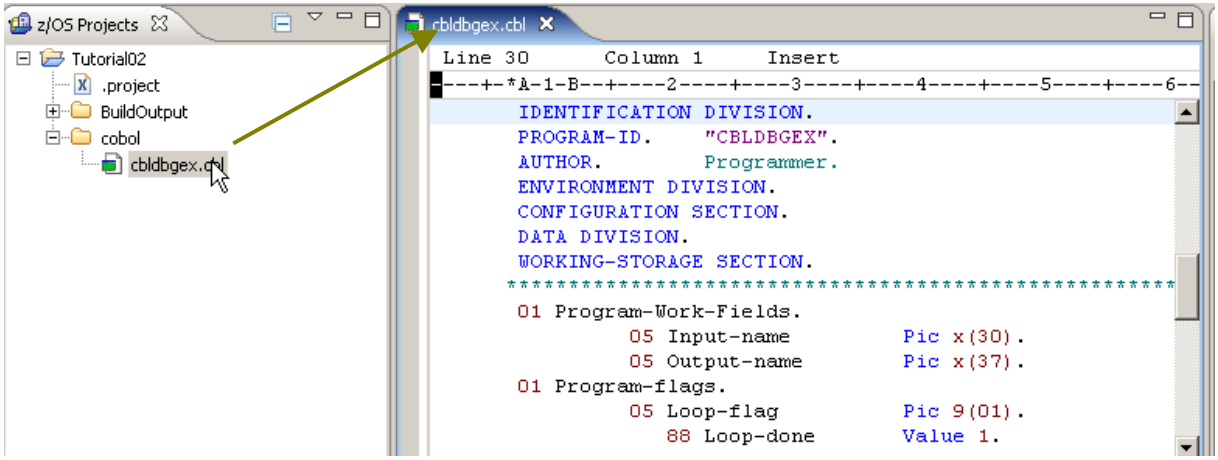


Abb. 3.1.1

Expandieren Sie den Cobol Folder. 2k auf cbldbgex.cbl. Sie sollten den Cobol Quell Code in dem mitleren Fenster sehen.

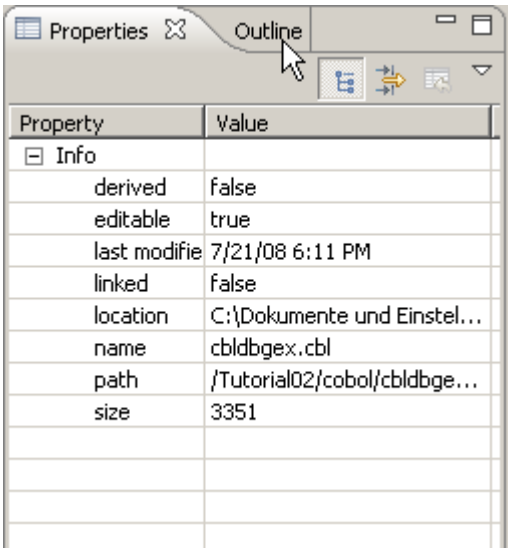


Abb. 3.1.2

Im linken unteren Fenster (unterer linker Teil des RDz Fensters) befindet sich der View (Registrierkarte) „Properties" (Eigenschaften), unter ihm verdeckt der View "Outline“ (Gliederung). Um letzteren View anzuzeigen, Klick auf die Registrierkarte „Outline“ (Gliederung).

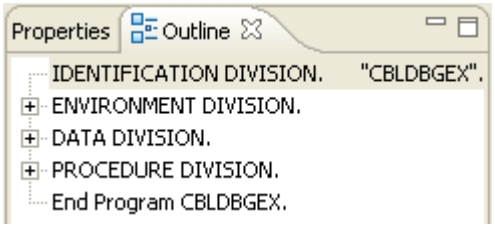


Abb. 3.1.3

Die Gliederungsansicht zeigt einen Überblick über die Strukturierung der Datei, die derzeit in dem Editor-Bereich geöffnet ist. Der Inhalt der Gliederungsansicht ist Editor-spezifisch.

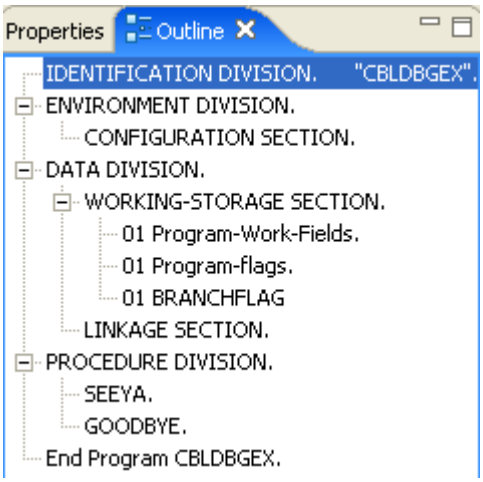


Abb. 3.1.4

Expandiere die + Zeichen und betrachte die COBOL Struktur.

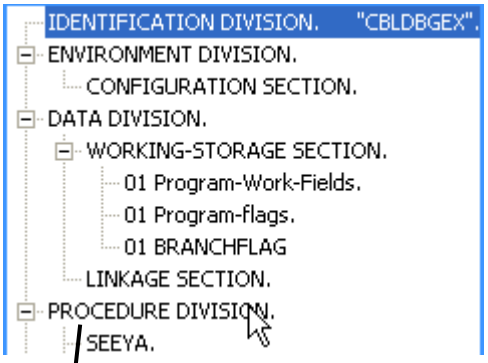


Abb. 3.1.5

Klick auf " PROCEDURE DIVISION" in der Gliederung bewirkt, dass genau dieser Teil des Cobol-Programm-Codes in der zentralen View angezeigt wird.

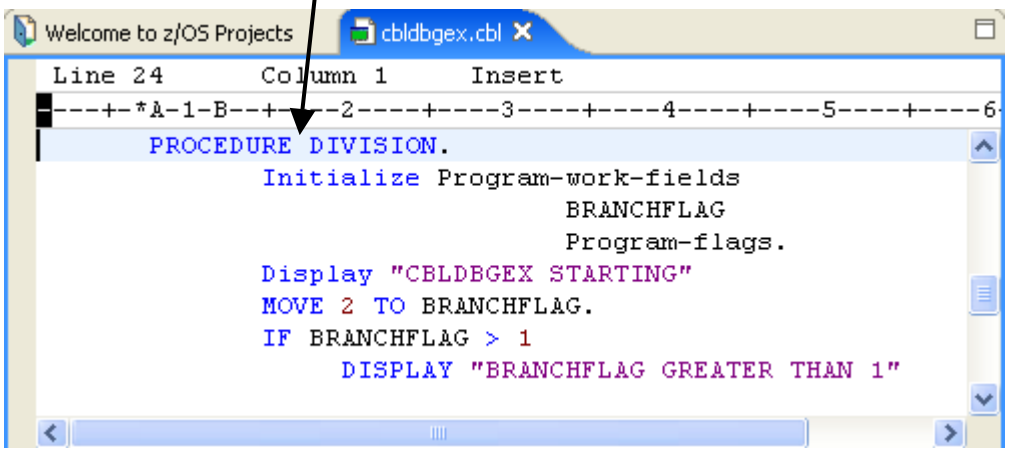


Abb. 3.1.6

Der Editor zeigt die PROCEDURE DIVISION des COBOL Programms. Klick auf "ENVIRONMENT DIVISION" in der Gliederung würde bewirken, dass genau dieser Teil des Cobol-Programm-Codes in dem zentralen View angezeigt wird.

3.2 LPEX und ISPF Editor

Der in RDz integrierte Editor heißt LPEX-Editor. LPEX (Live Parsing Extensible Editor) ist ein erweiterbarer und leistungsfähiger Texteditor. Zu den Funktionen zählen Syntaxhervorhebung und Programmierbarkeit.

Die Tastenkombination Strg+F ermöglicht nun das Suchen (und Ersetzen) von Strings im Cobol-Programm-Code.

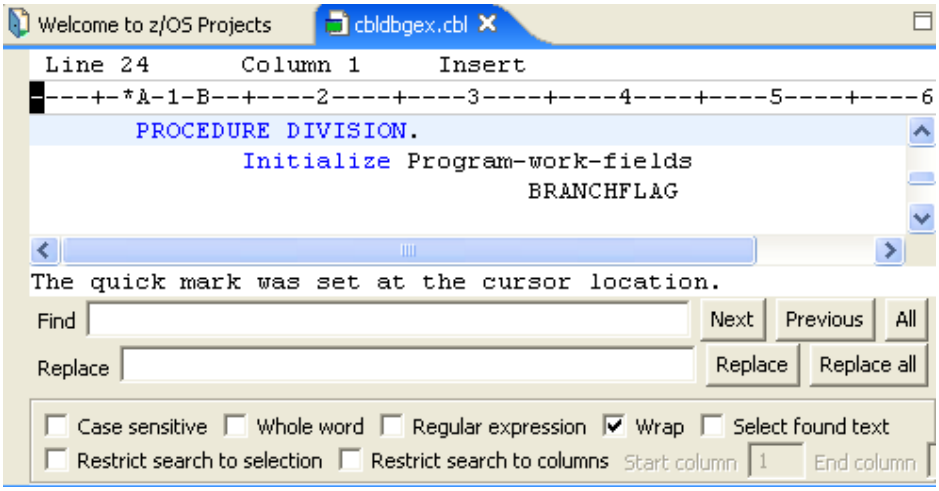


Abb. 3.2.1

Dies aktiviert die Search Facility. Soll zum Beispiel nach "Branchflag" gesucht werden:
In die Zeile "Find (Suchen)" den String "BRANCHFLAG" eingeben. (Ein Abschließen mit der Eingabetaste ist nicht erforderlich).

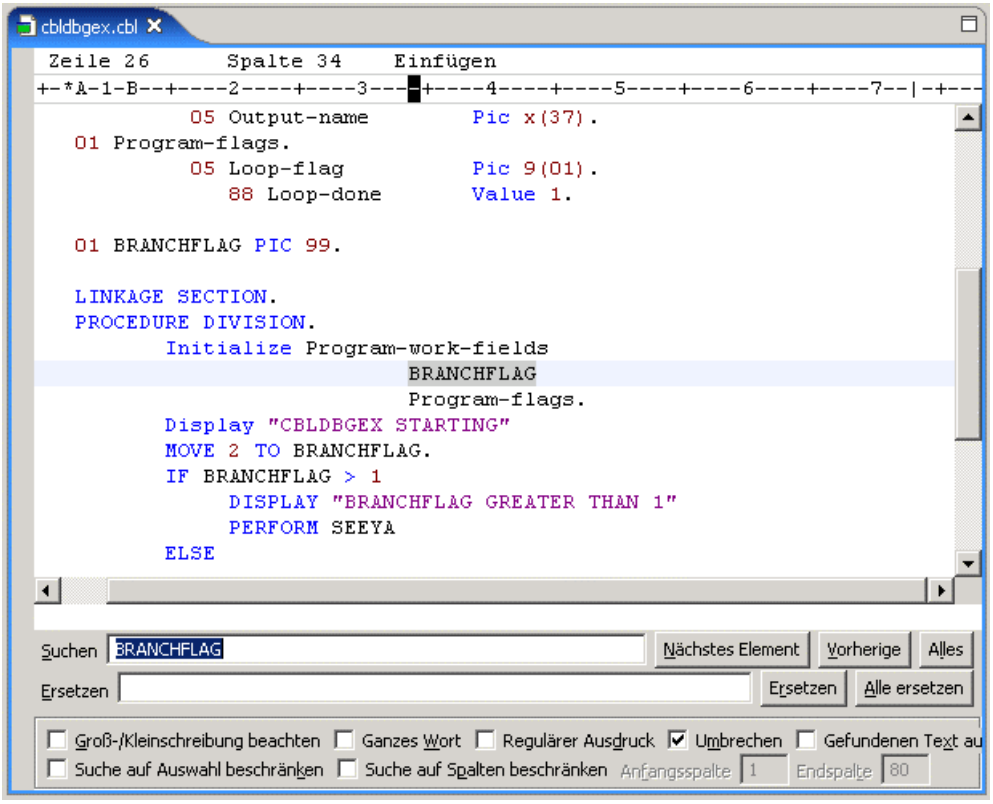


Abb. 3.2.2

Es ist intuitiv sehr einfach, diese Suchen- und Ersetzen-Funktionalität zu verwenden.

Probieren Sie doch mal:

- **Nach dem nächsten String "BRANCHFLAG" suchen,**
- **nach dem vorigen String "BRANCHFLAG" suchen,**
- **Alle vorhandenen "BRANCHFLAG" z.B. durch "BRANCHF" ersetzen, etc.**

Das vom LPEX Editor benutzte Default Profil ist lpex. Es definiert die Tasten Belegung. Der LPEX Editor kann die Tastenbelegung von anderen Editoren emulieren, u.a. dem

- **ISPF-Editor,**
- **Xedit-Editor (Standard CMS Editor für das z/VM Betriebssystem),**
- **Emacs-Editor,**
- **vi-Editor (von POSIX standardisierter Texteditor für Unix und Linux),**
- **etc.**

Standardmäßig ist das lpex Profil eingestellt. Ist man die Funktionsweise des ISPF-Editors gewöhnt, so lässt sich beispielsweise die ISPF-Editor-Emulation wie folgt aktivieren:

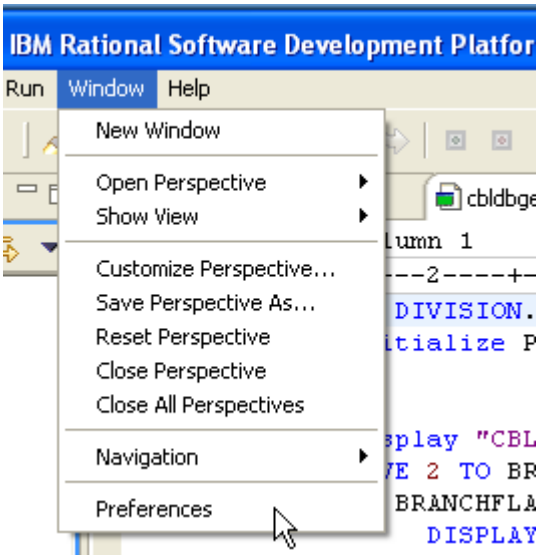


Abb. 3.2.3

Open the Preferences Dialog. 1k auf Window → Preferences.

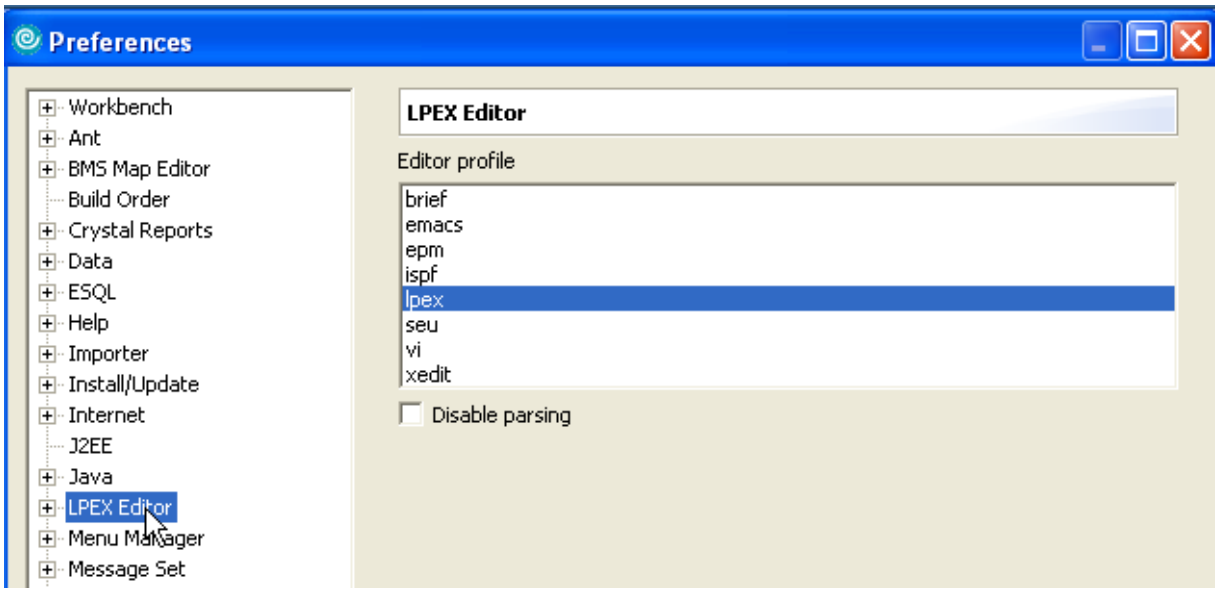


Abb. 3.2.4

1k auf LPEX Editor, und

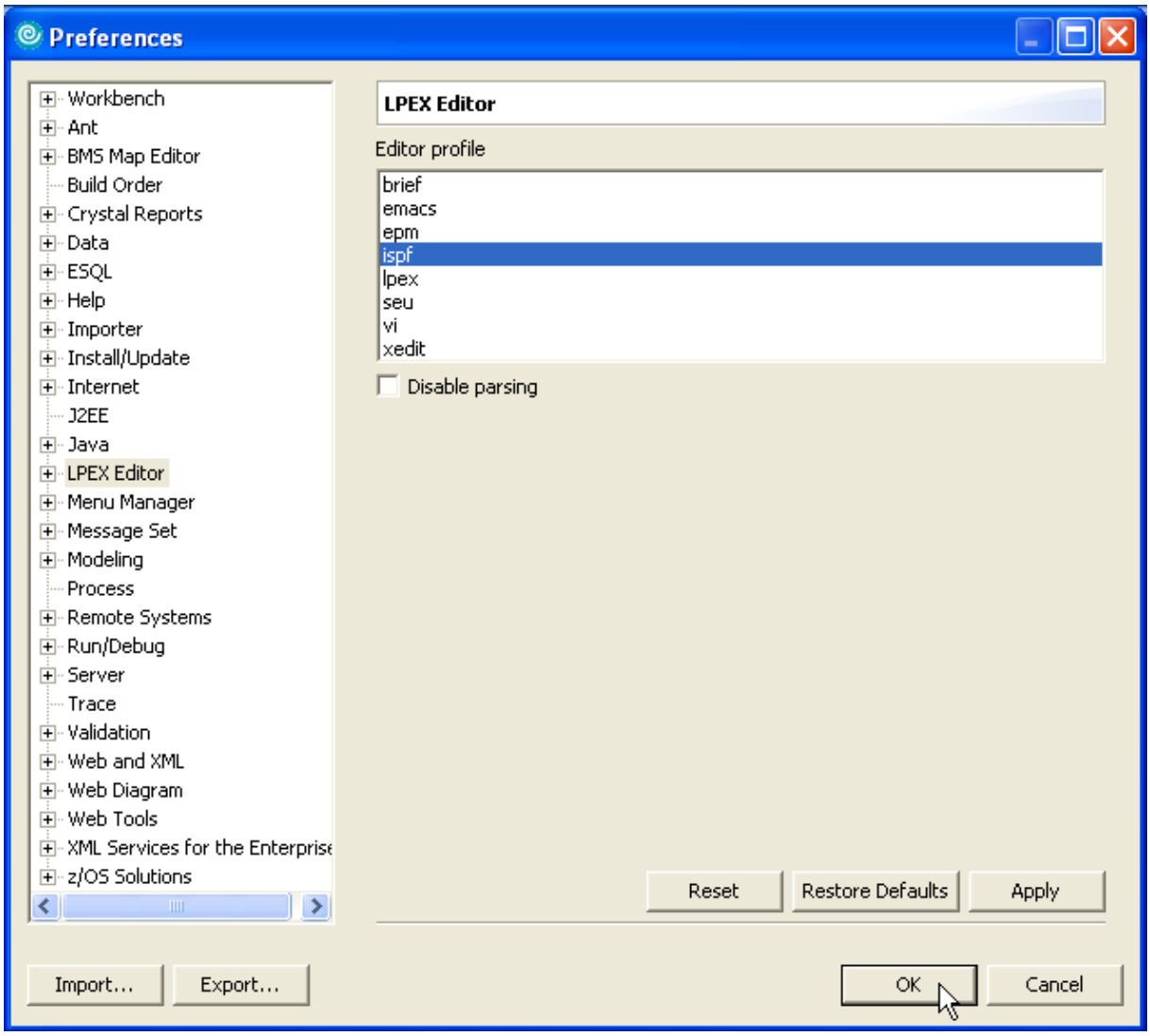


Abb. 3.2.5

1k on ispf, 1k on OK. Dies schließt den Preferences Dialog.

Selbst-Test

- **Ist es unbedingt erforderlich, das Editor Profile von Ipex auf ispf zu ändern, oder könnte man Cobol Programme auch mit dem Ipex Profile erstellen ?**

3.3 Erstellen von Tabs für den Editor

Da wir mit COBOL arbeiten, ist es eine gute Idee, einige TABS für eine einfache Navigation im Editor zu erstellen. Öffnen Sie den Dialog Preferences. 1k auf Window → Preferences.

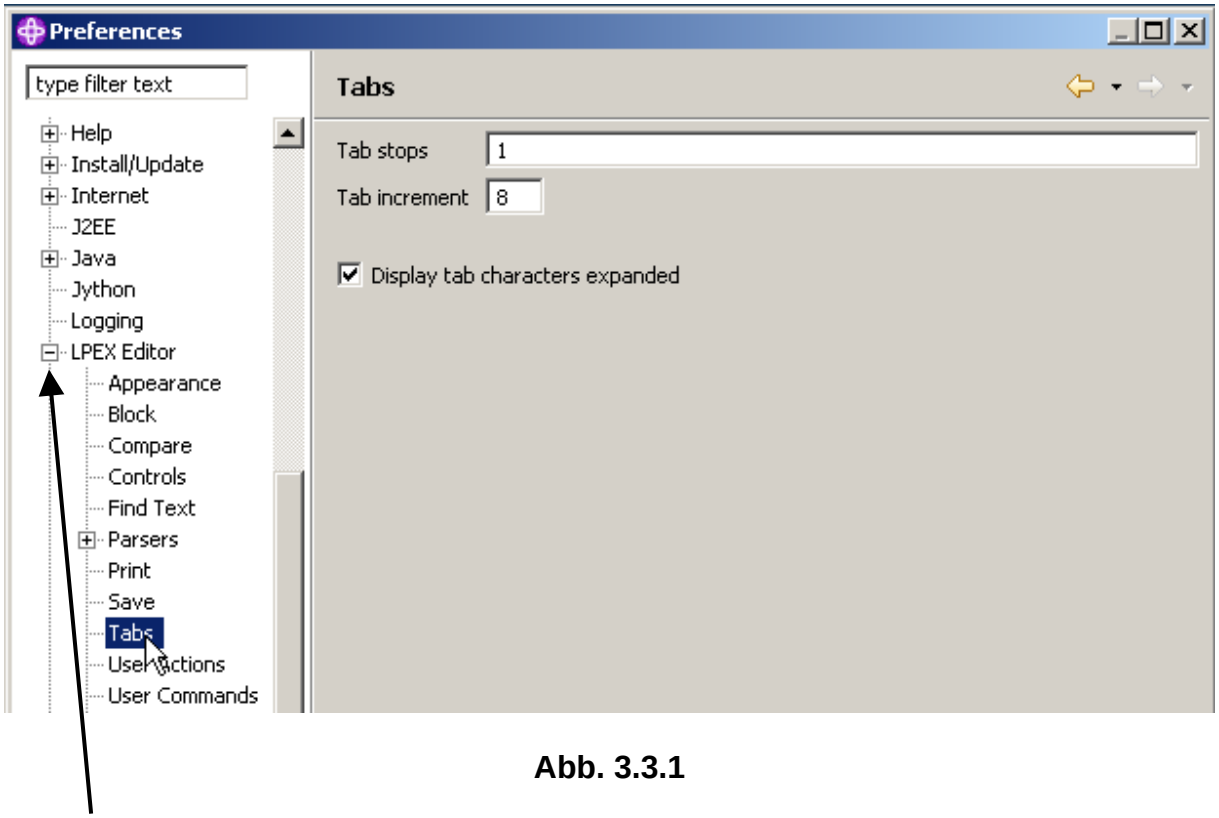


Abb. 3.3.1

Expand LPEX Editor, click on Tabs

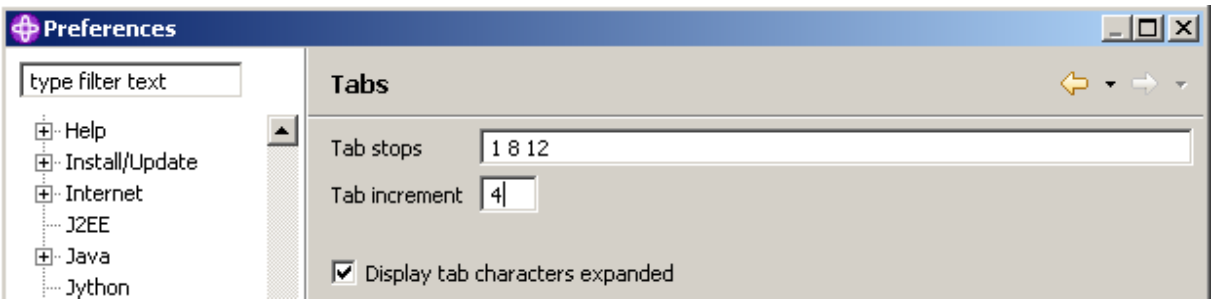


Abb. 3.3.2

Click on Tab stops und enter 1 8 12 als Tab Stops. Ändere das Tab increment zu 4 (statt 8). Jedes Mal wenn die Tab Taste gedrückt wird, wird der Cursor auf Positions 1, 8 und 12 bewegt. Das Increment 4 bewirkt weiterhin Stops auf 16, 20, 24 usw. Click OK.

3.4 Using Prefix Commands

Sie sind jetzt in der Lage, mit Präfixbefehlen im Präfix Bereich zu arbeiten, wie d zu löschen, m zu bewegen, etc. Sie können auch Tastenkombinationen für Befehle, z.B. (Strg + L) für Locate, (Strg + F) für Find, etc. eingeben. Siehe Tutorial 1c für Einzelheiten.

Beispiel, um nach Zeile 25 zu bewegen, hit Ctrl + L,

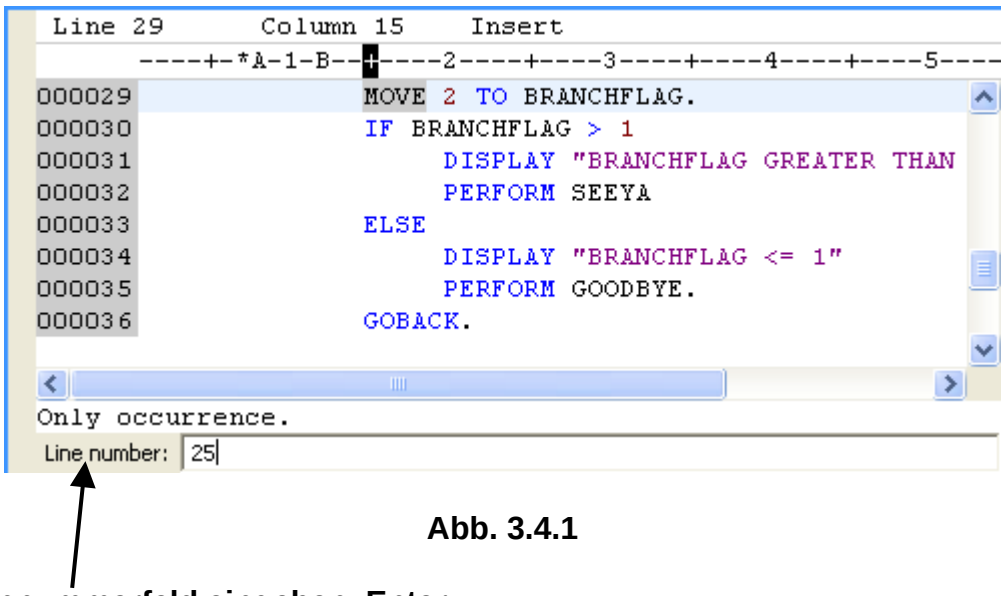


Abb. 3.4.1

25 in das Zeilennummerfeld eingeben, Enter.

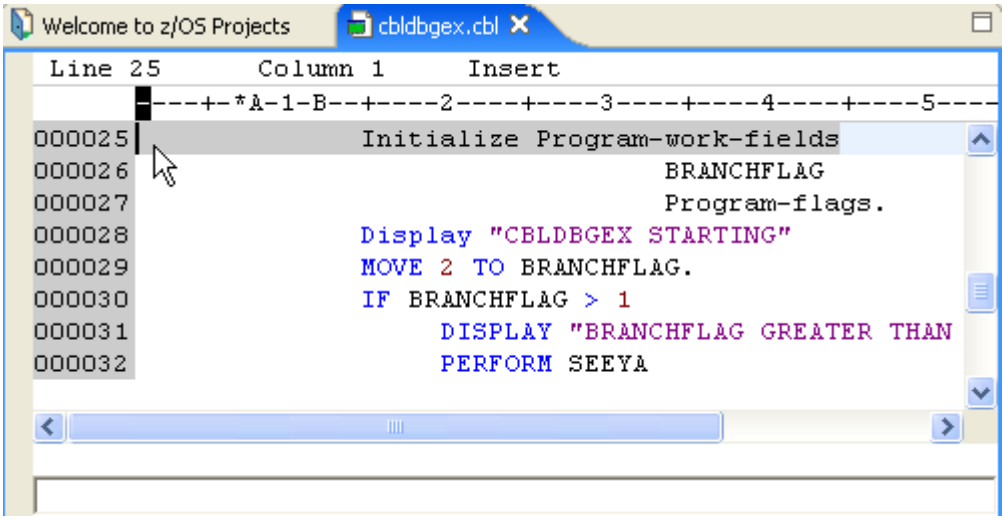


Abb. 3.4.2

Dies ist das Ergebnis

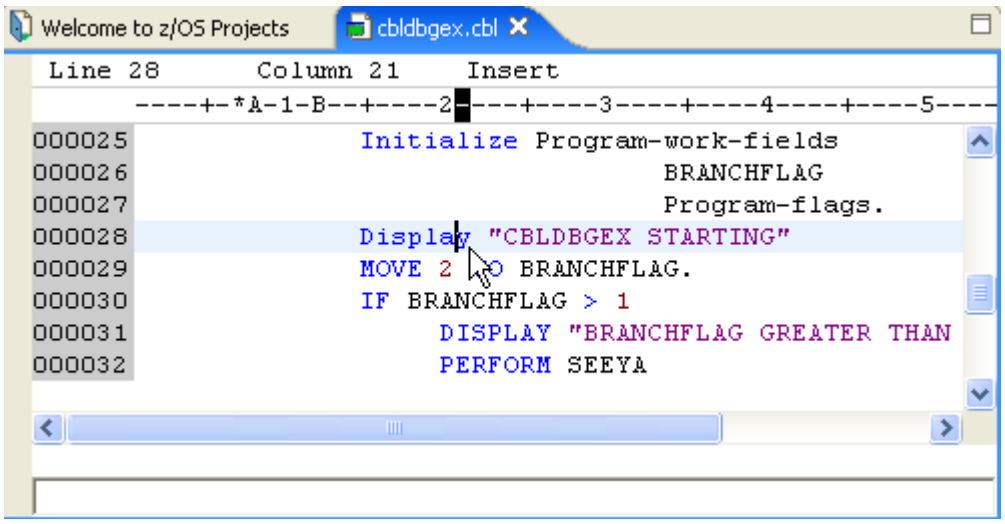


Abb. 3.4.3

Nach Zeile 28 bewegen, den Tab Key benutzen um den Cursor auf das Wort “Display” zu setzen. (Anmerkung: Shift und Tab bewegt den Cursor zurück).

Selbst-Test

- **Können Prefix Commands auch mit dem Ipex Profile benutzt werden ?**

Tipp: Wenn die Tabs beim ersten Mal nicht arbeiten, wechseln sie von ISPF nach LPEX zurück, und dann wieder nach ISPF.

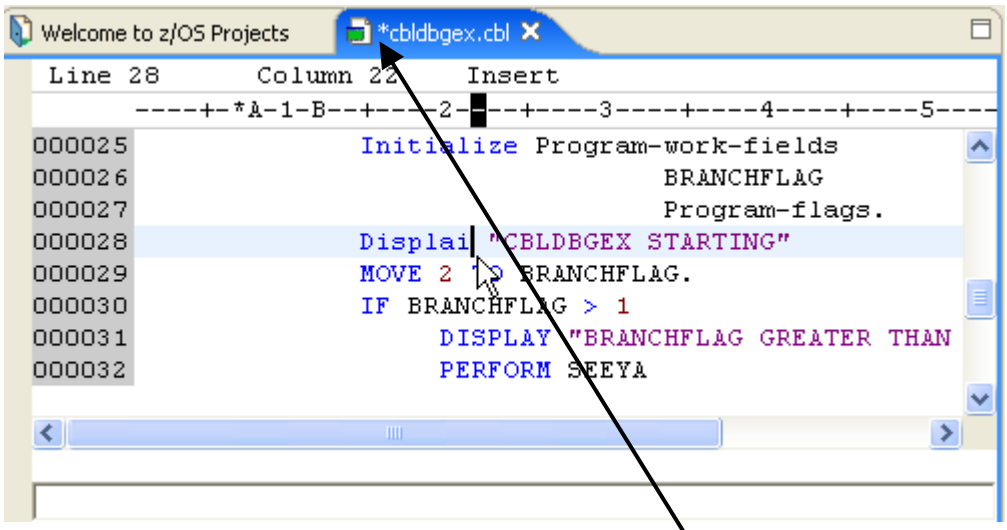


Abb. 3.4.4

Ändern Sie das Wort Display in Displai. Beachten Sie: da wir über einen Smart Editor verfügen, und Displai kein gültige COBOL Statement ist, wechselt die Farbe von blau auf schwarz .

Achten Sie auf den Stern * auf der linken Seite des Dateinamens 

Er zeigt an, dass die Datei geändert wurde. Das Sternchen verschwindet, sobald die Datei gespeichert wurde.

Drücken Sie STRG + S, um die Änderung zu speichern. Zu diesem Zeitpunkt wird kein Fehler festgestellt, da das Programm compiliert werden muss, um Fehler zu finden.

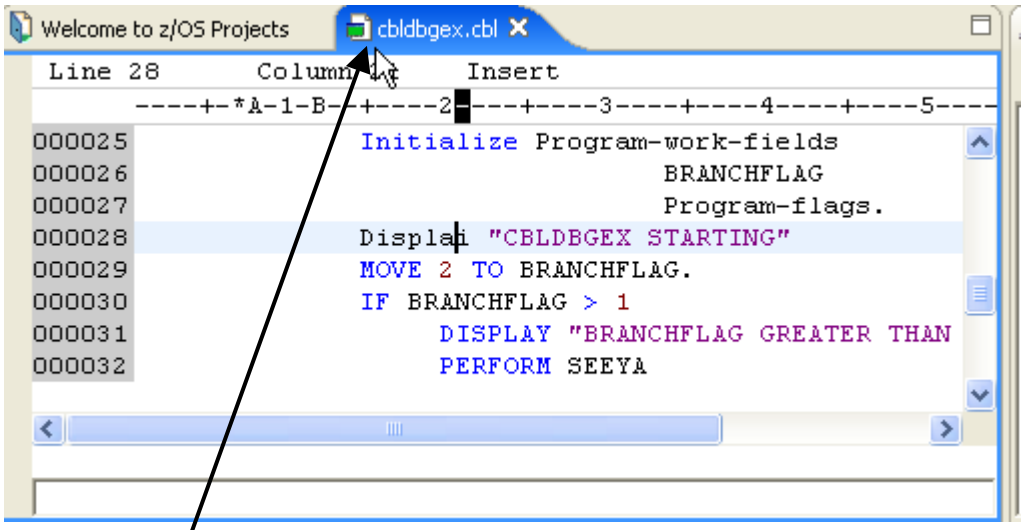


Abb. 3.4.5

Beachten Sie, dass der * links vom Dateinamen verschwunden ist.

Machen Sie eine weitere Änderung. Gehen Sie nach Zeile 33 , geben Sie ein “d” in linken grauen Area (der „Line Command Area“, siehe Tutorial 1c) ein.

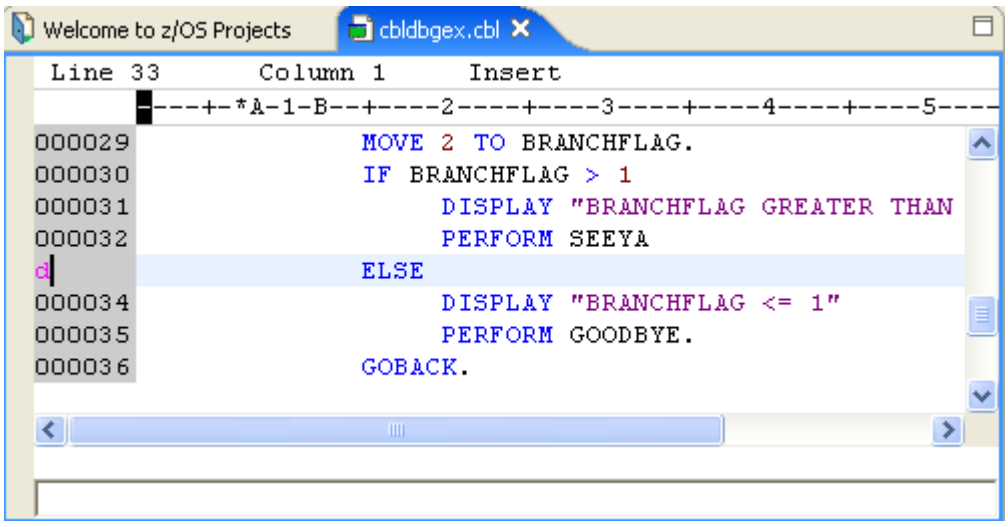


Abb. 3.4.6

Drücken Sie Enter, um die Zeile zu löschen.

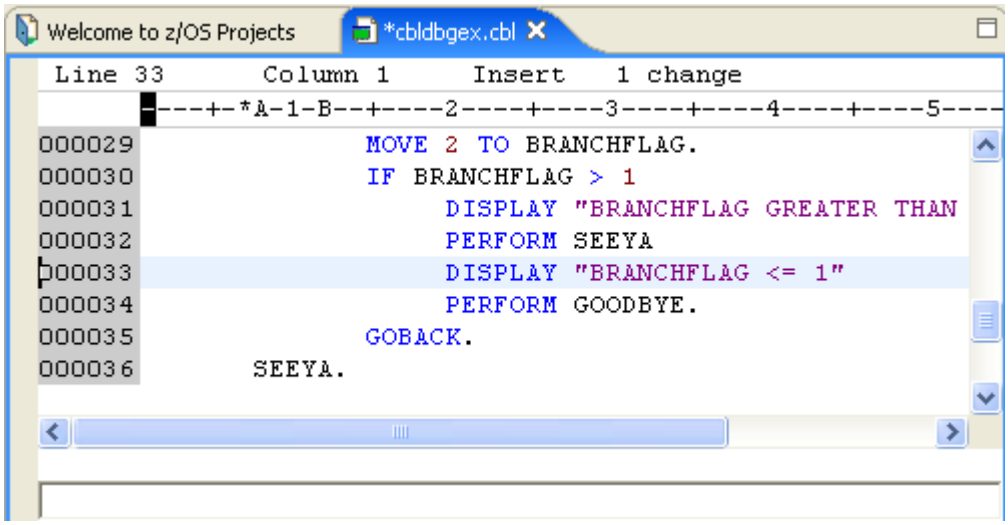


Abb. 3.4.7

Drücken Sie CTRL + S um es zu retten (to save it).

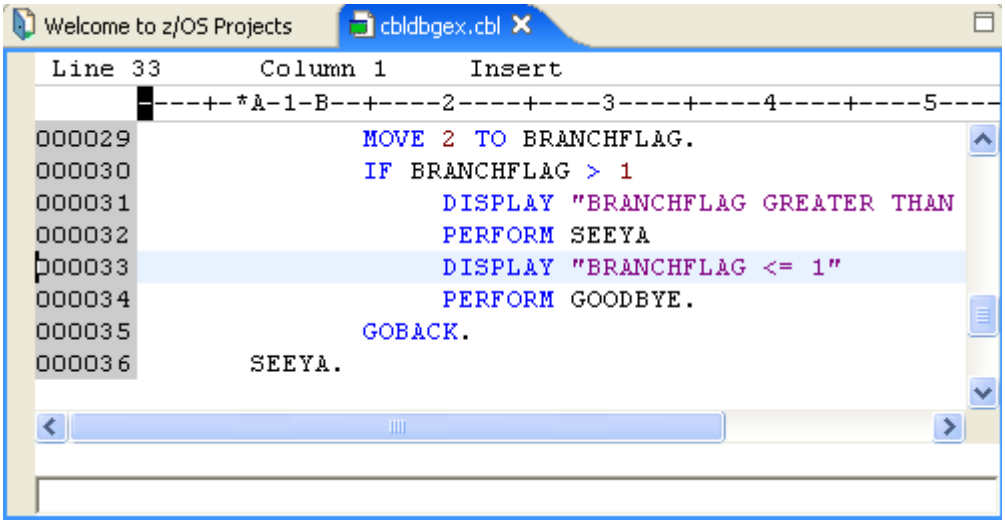


Abb. 3.4.8

Dies ist das Ergebnis.

Sie können jederzeit das Editor Fenster vergrößern oder verkleinern. Gelegentlich werden sie mehr Raum brauchen um das ganze Programm zu sehen.

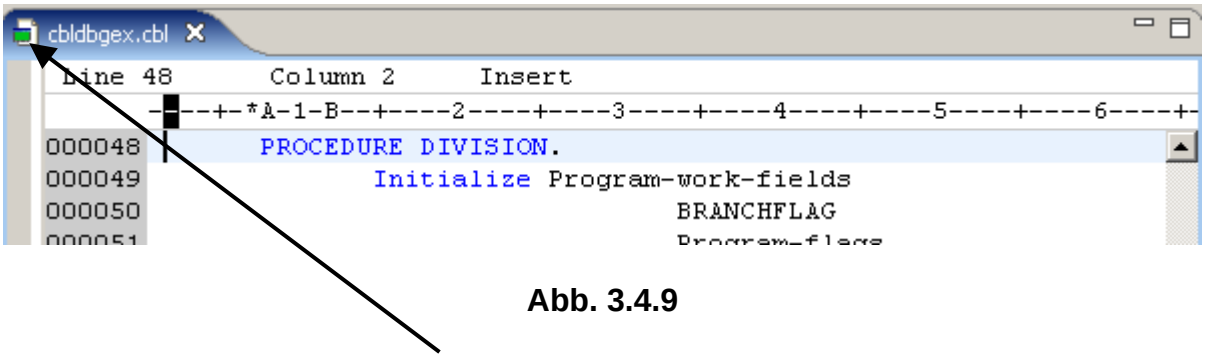


Abb. 3.4.9

Am Einfachsten geht dies mit einem 2k (Double Click) an dieser Stelle. Dies maximiert die Größe des Fensters. Ein weiteres 2k stellt die ursprüngliche Fenstergröße wieder her.

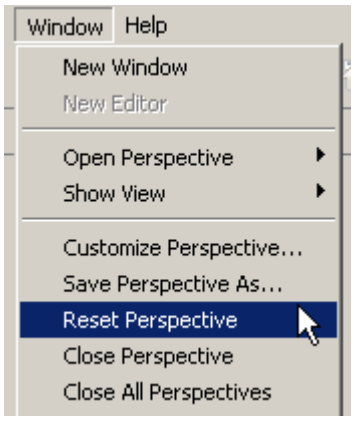


Abb. 3.4.10

Wenn Sie die Größe des Editor Fensters geändert haben, irgendwelche Views geschlossen haben, usw. und Sie wollen zum Originalzustand der Perspective zurückkehren, selektieren Sie Window → Reset Perspective.

3.5 Benutzung lokaler Datei Versionen

Die lokale History einer Datei wird automatisch zwischengespeichert, wenn Sie diese erstellen oder ändern. Jedes Mal, wenn Sie eine Datei bearbeiten und speichern, wird eine Kopie der alten Version zwischengespeichert. Damit können Sie jederzeit Ihren aktuellen Datei Zustand mit einen früheren Zustand vergleichen, oder die Datei durch einem früheren Zustand ersetzen. Jeder Status in der lokalen History wird durch das Datum und die Uhrzeit der Datei-Speicherung identifiziert. Um eine aktuelle Version durch eine vorherige zu ersetzen (z.B. die Zeile, die Sie vorher gelöscht haben, wiederherzustellen) gehen Sie so vor:

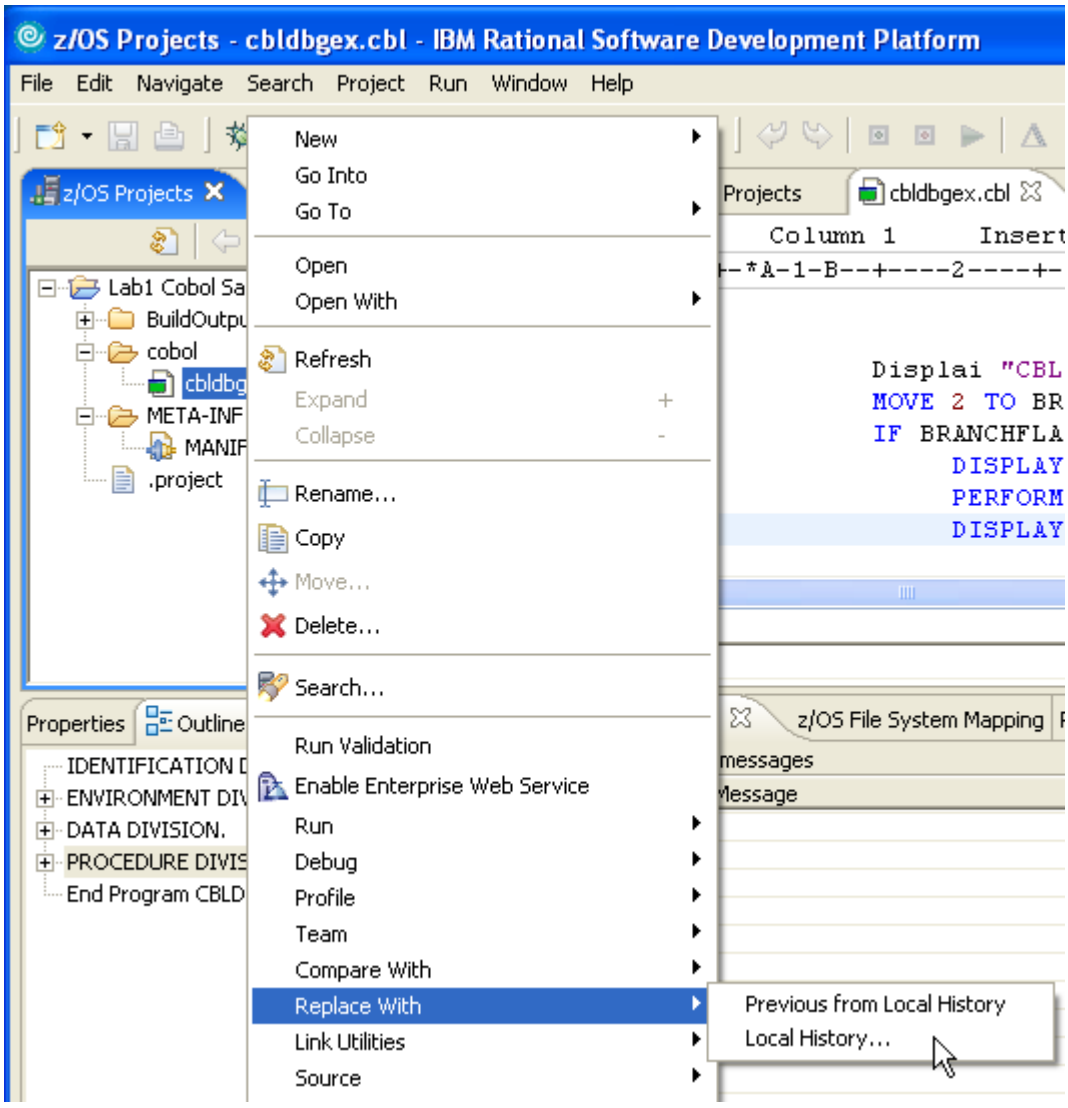


Abb. 3.5.1

Im z/OS Projects View, 1kr auf cbldbgex.cbl. Von dem pop-up Menu, selektiere “Replace with! → Local History

1k

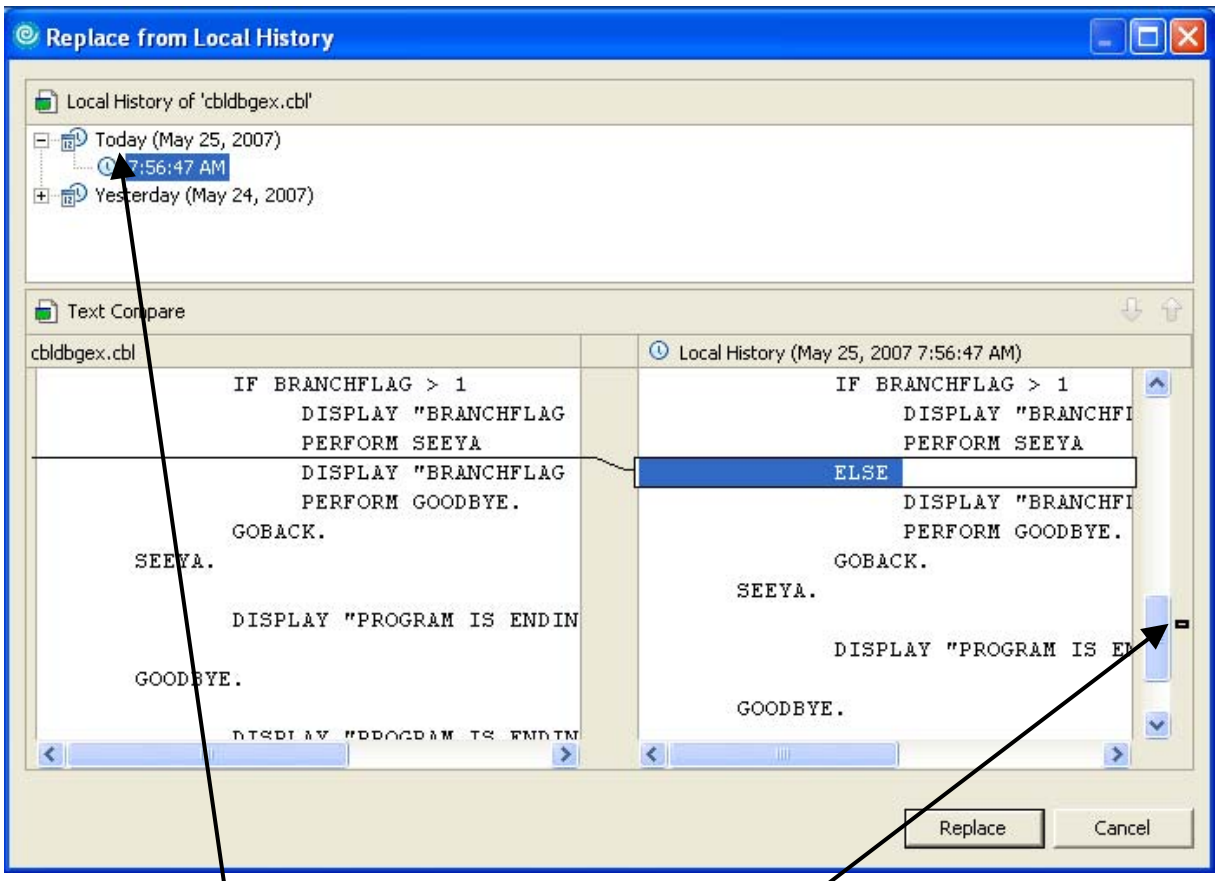


Abb. 3.5.2

In diesem Fall sind Datum und Uhrzeit unterschiedlich.

Das „Replace from Local History" Fenster wird geöffnet. Auf der linken Seite sehen Sie die aktuelle Version unseres Programms. Auf der rechten Seite befindet sich die vorherige Version.

Beachten Sie das kleine Zeichen auf der rechten Seite. Dies ist wichtig, wenn wir vielfache Änderungen vorgenommen hatten.

Nur auf diese Marke klicken, um zur nächsten Änderung zu gelangen. Beachten Sie, dass in diesem Beispiel nichts geschieht, da Sie nur eine Veränderung vorgenommen hatten.

Von der Local History, wählen Sie die Zeit, die Sie ersetzen möchten. In diesem Beispiel, wählen Sie den jüngsten Eintrag.

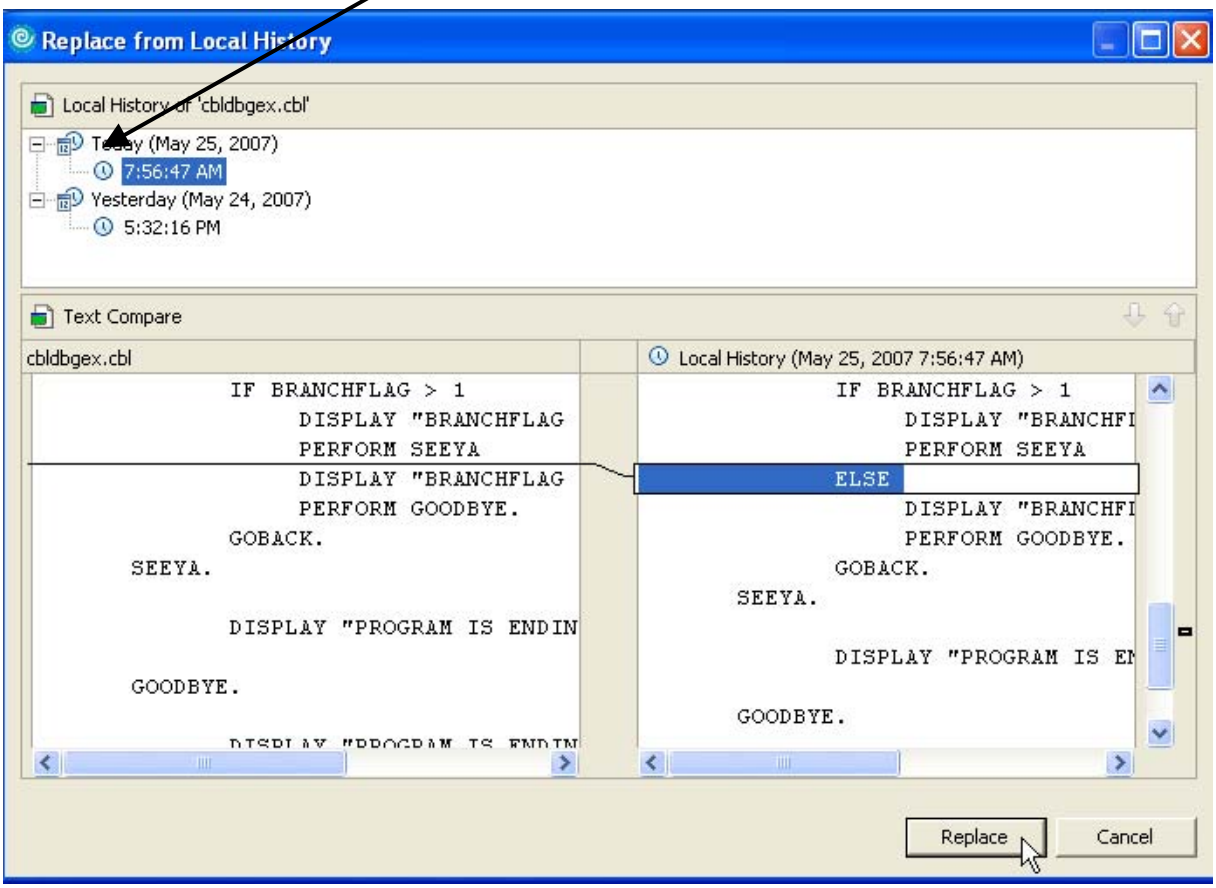


Abb. 3.5.3

Dann click Replace.

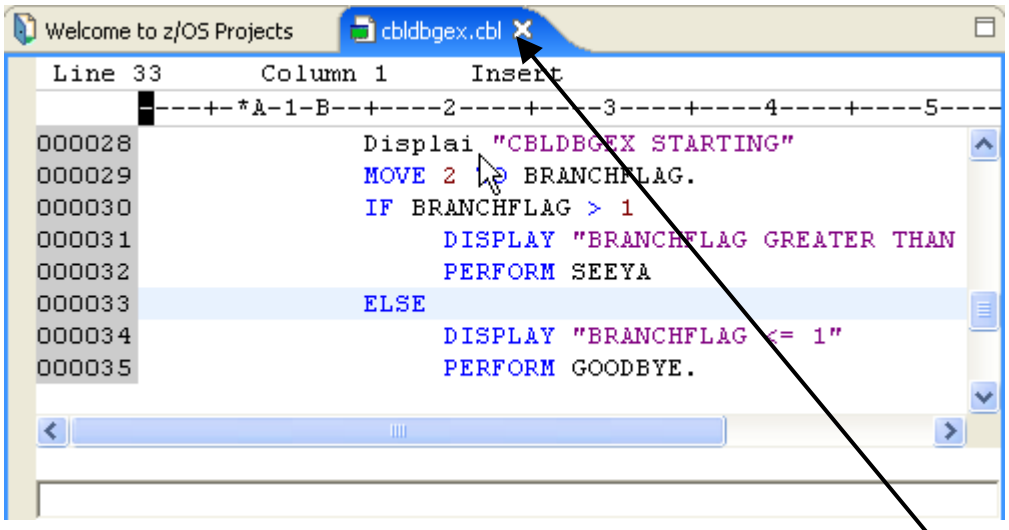


Abb. 3.5.4

Zeile 33 ist wiederhergestellt, die Änderung “Displai“ ist noch vorhanden.

Den Editor schließen durch ein Klick auf das X neben dem File Namen.

3.6 Überprüfung der COBOL-Quelldatei Syntax

In dem z/OS-Projekte View, highlight cbldbgex.cbl, 1kr (klicken Sie die rechte Maustaste), und wählen Sie Local Syntax Check Check aus dem Pop-Up-Menü:

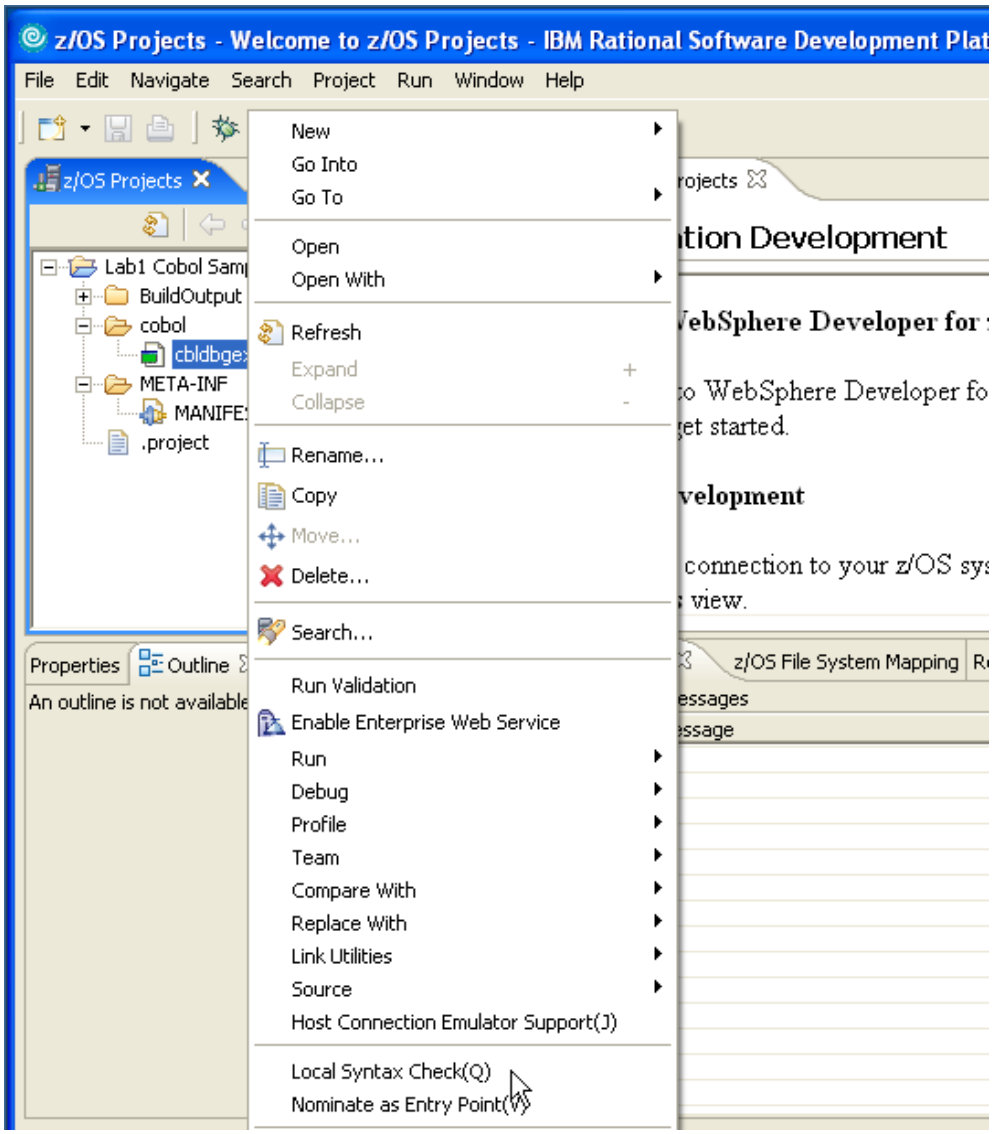


Abb. 3.6.1

Click auf Local Syntax Check.

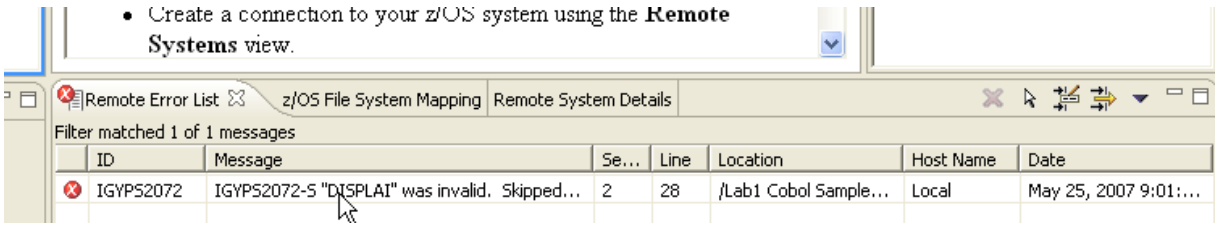


Abb. 3.6.2

Schauen Sie nach unten. Das Tab “Remote Error List” zeigt die Compile Fehler an.

Doppelklicken Sie auf einen Fehler. Das bringt Sie zu dem Editor an die Stelle, wo das fehlerhafte Statement entdeckt wurde.

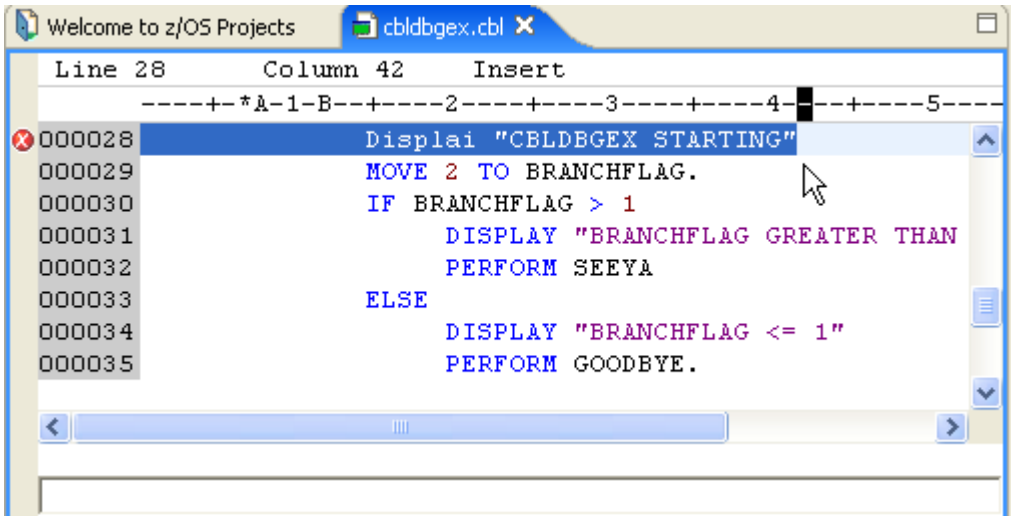


Abb. 3.6.3

Die fehlerhafte Zeile ist highlighted.

Ändern Sie Displai in Display.

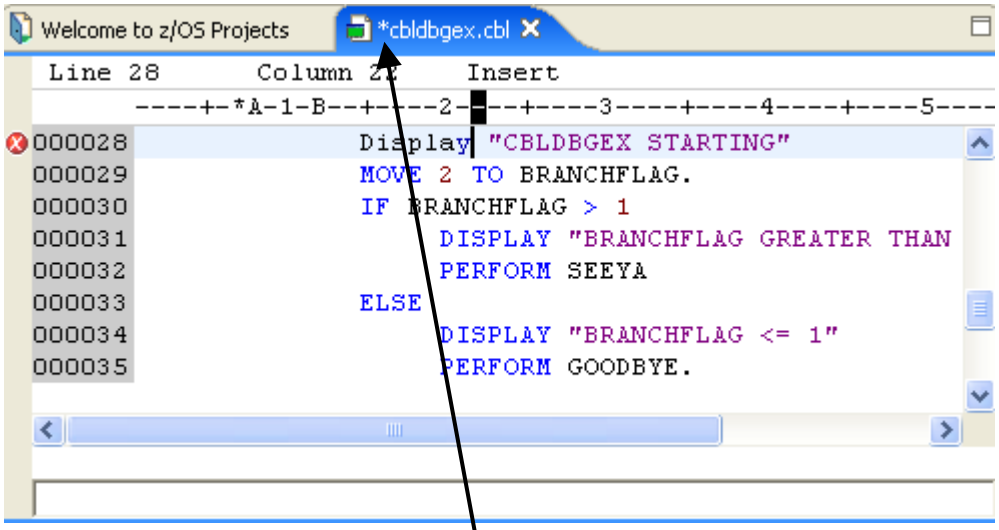


Abb. 3.6.4

Save Änderungen mit (Ctrl + S), Das Sternchen * verschwindet.

Führen Sie einen weiteren Lokalen Syntax Check durch. Sie sollten keine Fehler in der Remote Error List nach dem Ausführen des Local Syntax Check mehr sehen.

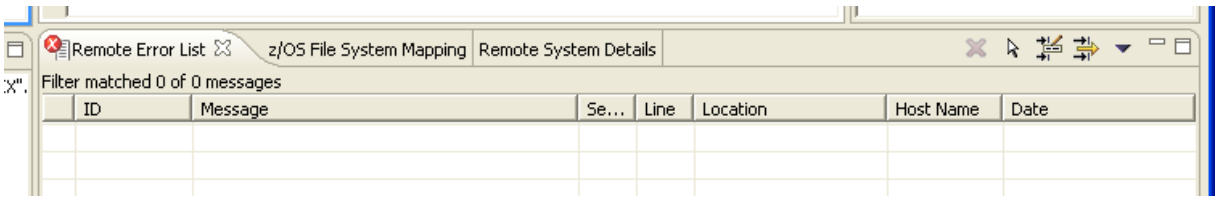


Abb. 3.6.5

Die Compilierung ist jetzt sauber.

3.7 Fenster vergrößern/verkleinern

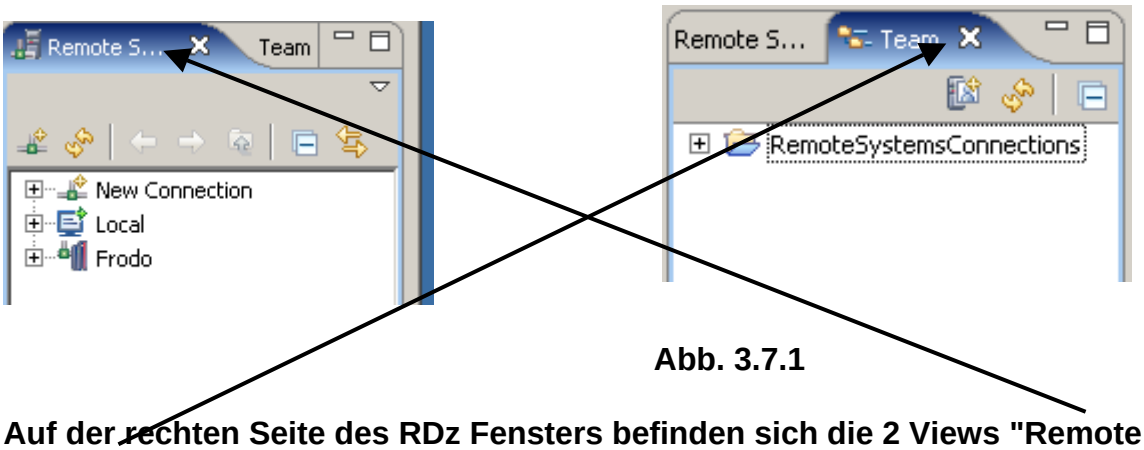


Abb. 3.7.1

Auf der rechten Seite des RDz Fensters befinden sich die 2 Views "Remote Systems" und "Team" jeweils überlagert. Sie können jeden der beiden Views durch einen Klick auf die entsprechende Registrierkarte (Tab) öffnen.

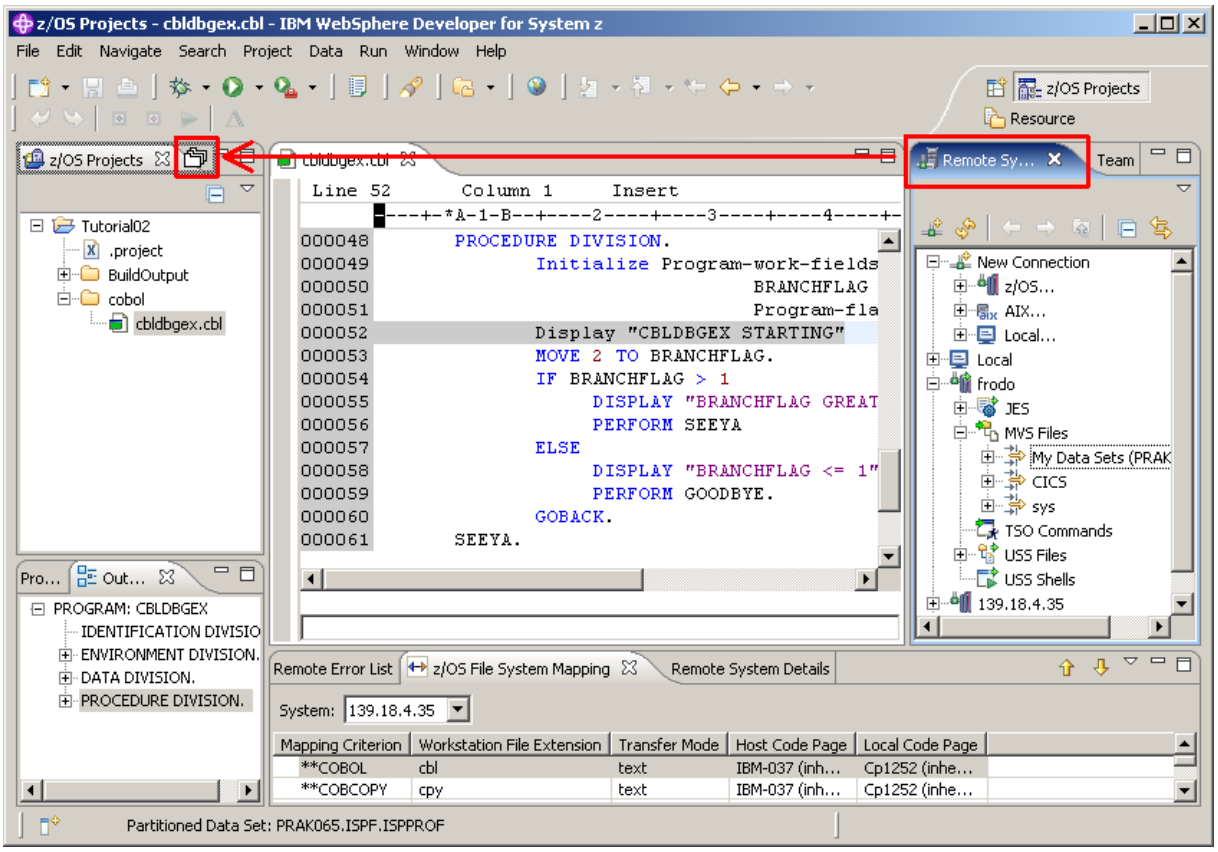


Abb. 3.7.2

In diesem Tutorial werden Sie die Ansichten Remote Systems und Team nicht benutzen. Sie können beide Views auf der linken Seite des RDz Fensters bewegen, um mehr Platz für die COBOL-Bearbeitung zu gewinnen. Hierzu Drag & Drop beide Views auf der Oberseite des z/OS Projects View, wie unten dargestellt.

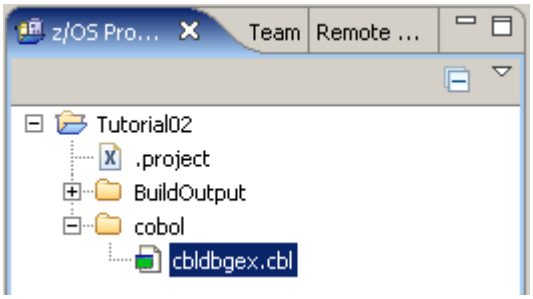


Abb. 3.7.3

Beachten Sie, dass der Maus Zeiger sich zu ändert, ehe Sie den Drop durchführen. Sie müssen dies für jeden der beiden Views getrennt tun.

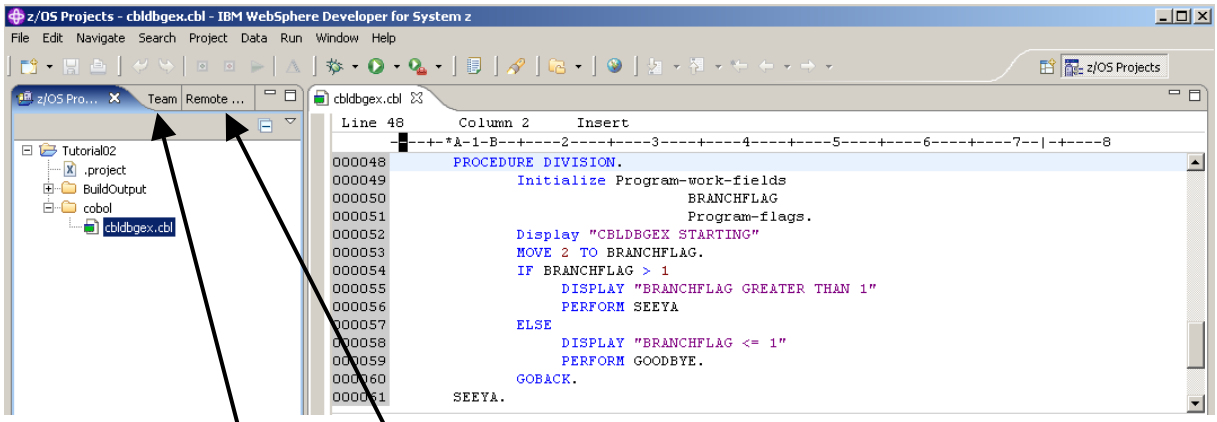


Abb. 3.7.4

Dies sind die Tabs für Team und für Remote System. Das Editor Fenster hat jetzt mehr Platz.

Je nach Bildschirmauflösung können Sie jetzt 1k auf und dann den z/OS Projects tab selektieren.

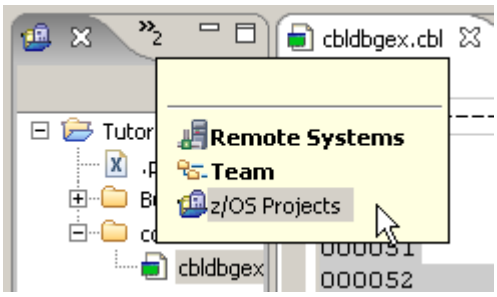


Abb. 3.7.5

Klick auf z/OS Projects. Ansonsten 1k auf das z/OS Projects Tab.

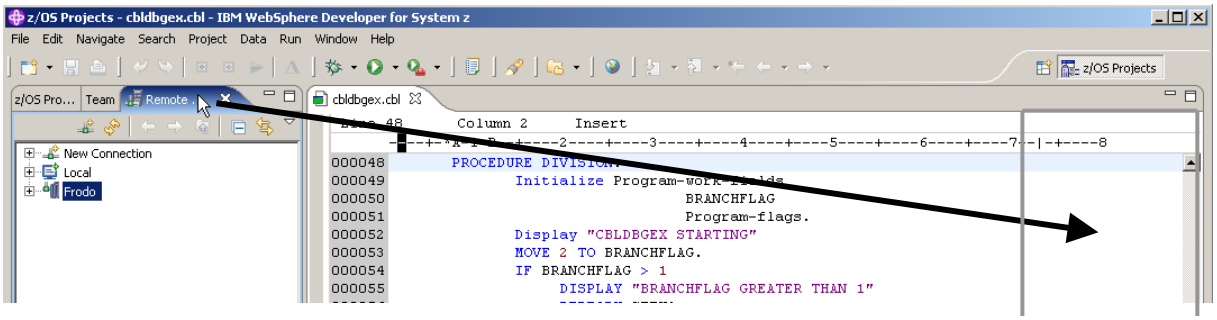


Abb. 3.7.6

Um das Fenster wiederherzustellen, einfach per Drag & Drop die Registrierkarte an den ungefähren Ort, wo Sie wollen, dass das Fenster wiedergegeben wird. Beachten Sie, ein Rechteck erscheint, um die Position anzugeben.

3.8 Das Cobol Quellprogramm ändern

Einer der attraktiven Eigenschaften des COBOL-Editor ist die Fähigkeit, Fehler vor der Kompilierung zu finden und Entwicklern zu helfen, Code zu schreiben. Lassen Sie uns diese Funktionen hier näher ansehen. Angenommen, wir wollen etwas Logik zu unserem Programm hinzufügen.

Ein Beispiel ist, wenn wir zu dem COBOL Quell-Programm Statements wie: DISPLAY, ACCEPT, MOVE und IF hinzufügen möchten. Hier sind einige Beispiele:

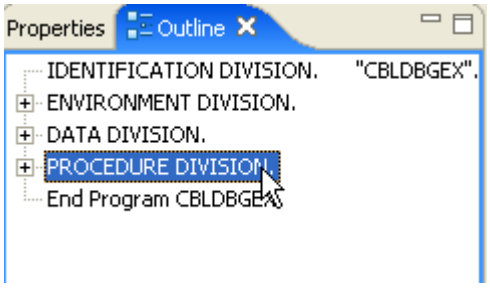


Abb. 3.8.1

Um COBOL Anweisungen unmittelbar nachdem der PROCEDURE DIVISION hinzuzufügen, verwenden Sie den Outline View. Klicken Sie auf die PROCEDURE DIVISION, um den Cursor auf dieser Zeile zu positionieren.

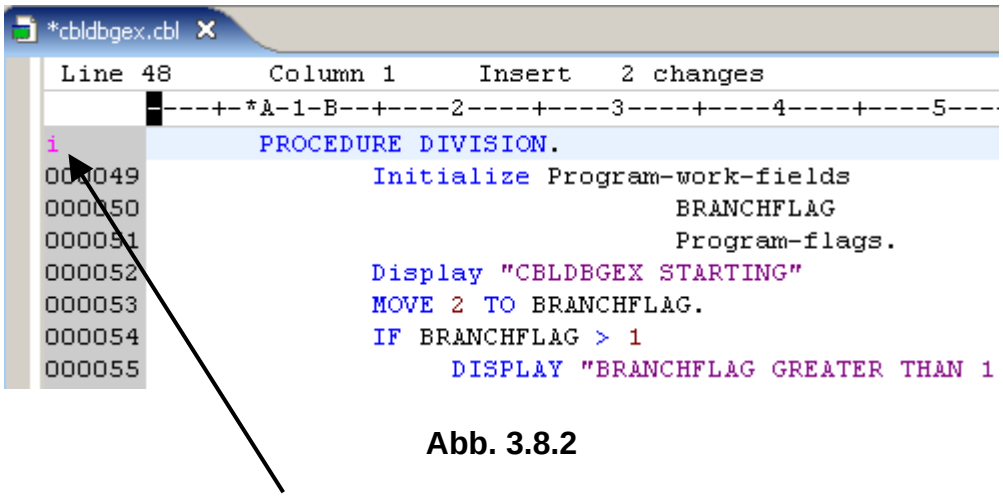


Abb. 3.8.2

Dann das Command “i” in den grauen Bereich (Line Command Area) für diese Zeile eingeben.

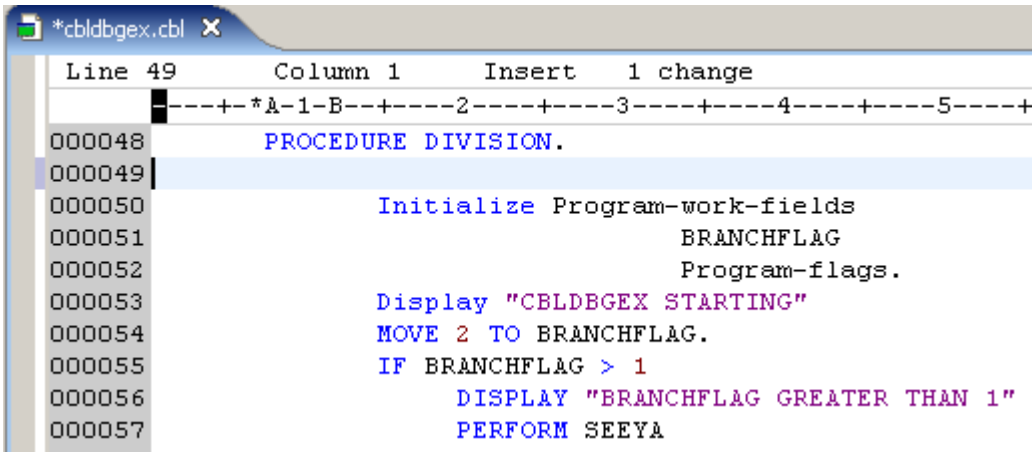


Abb. 3.8.3

Enter, um eine Leerzeile zu erzeugen. Alternativ könnten Sie CTRL + ENTER benutzen um eine Leerzeile zu erzeugen.

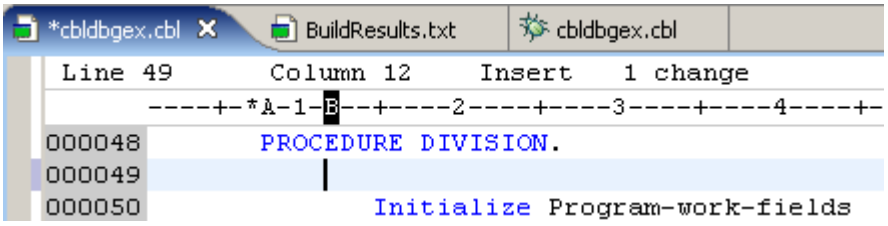


Abb. 3.8.4

Zweimal Tab Taste bewegt den Cursor auf Position 12.

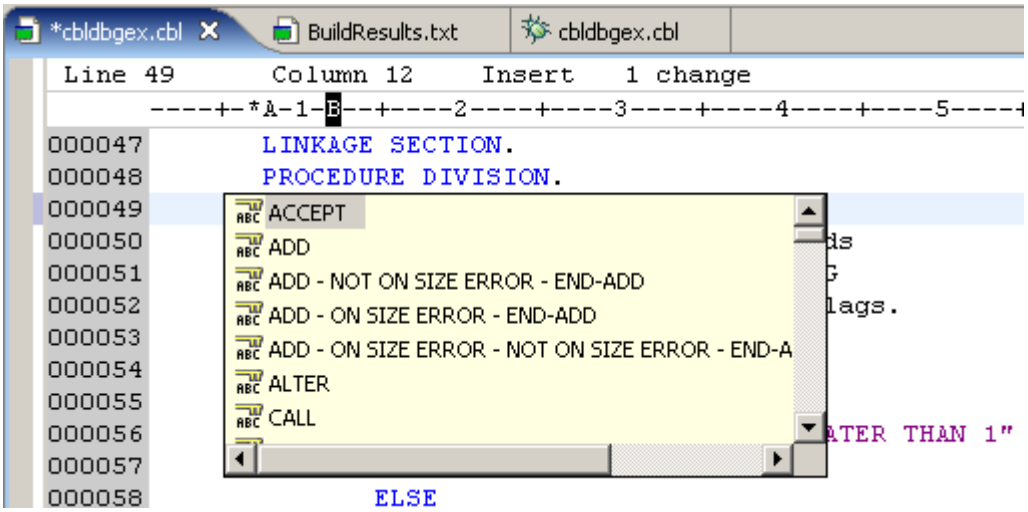


Abb. 3.8.5

Benutzen wir eine nette Eigenschaft: „Code Assist“. Mit zweimal Tab den Cursor nach Spalte 12 bewegen. CTRL + SPACE eingeben. Sie sehen eine mögliche Liste aller COBOL Statements.

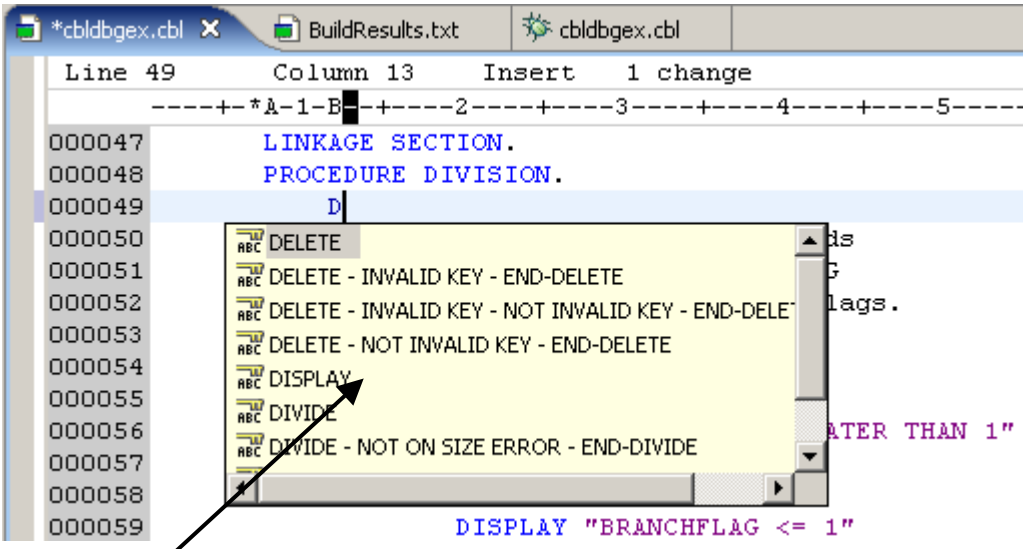


Abb. 3.8.6

Buchstaben D eingeben. Angezeigt wird eine Liste aller Cobol Statement, die mit D anfangen. 2k auf DISPLAY.

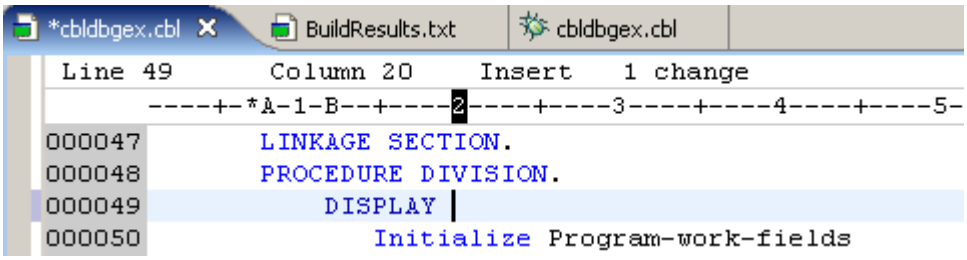


Abb. 3.8.7

Eingeben “Enter name or Q to quit”, Leerzeichen eingeben,

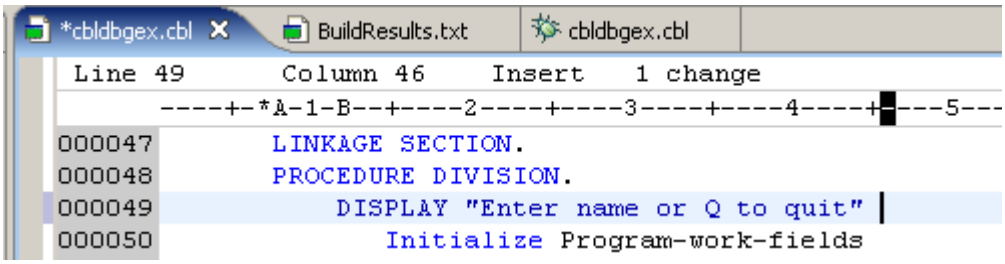


Abb. 3.8.8

wieder CTRL + SPACE,

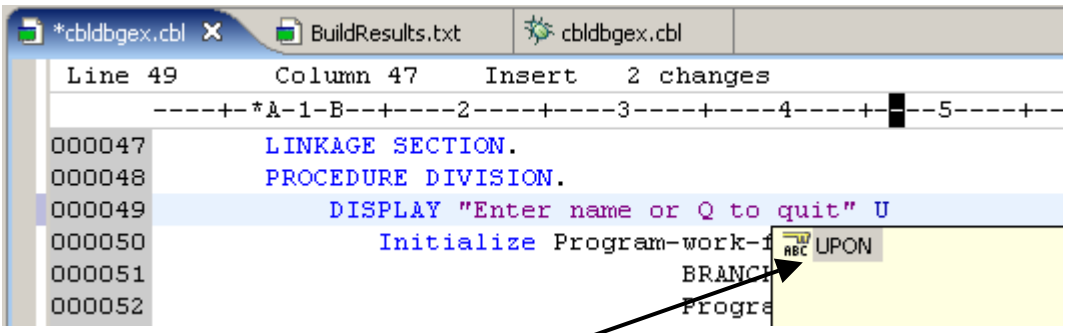


Abb. 3.8.9

ein „U“ eingeben, und 2k auf UPON

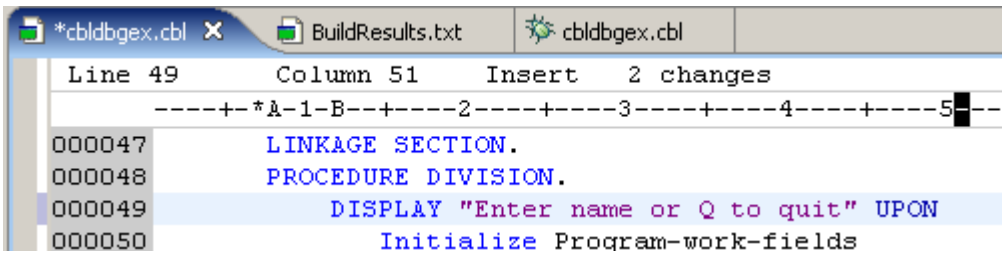


Abb. 3.8.10

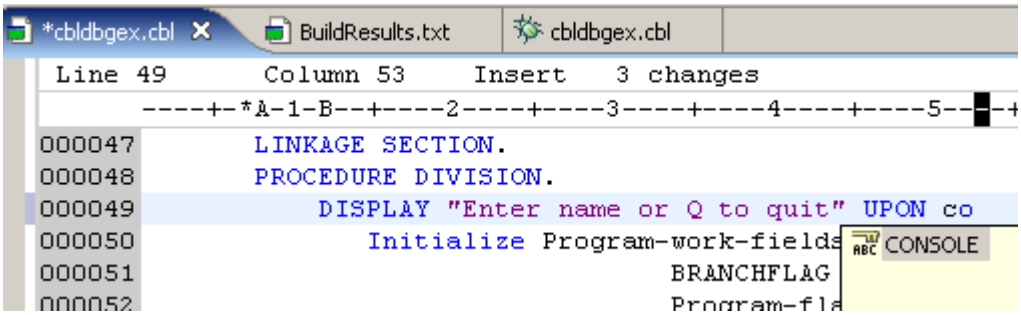


Abb. 3.8.11

nochmals CTRL + SPACE, , Eingabe “co”, und 2k auf CONSOLE

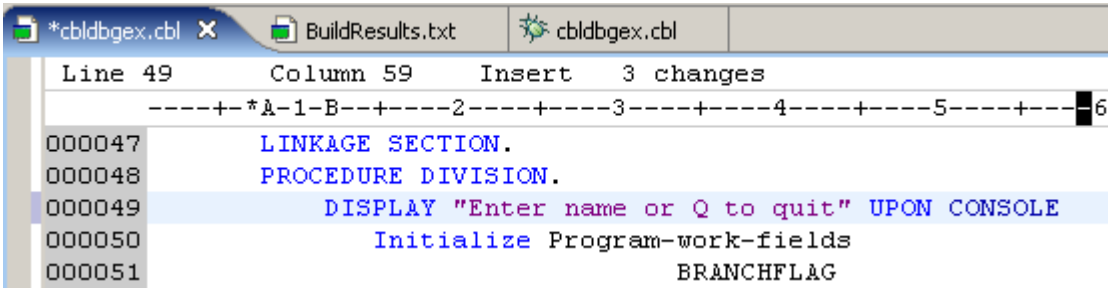


Abb. 3.8.12

Die Verwendung von STRG + SPACE ist sehr hilfreich, wenn wir Daten-Namen verwenden möchten. Zum Beispiel, wenn Sie den Befehl ACCEPT verwenden, um Daten von der Konsole in einen Daten-Namen zu kopieren, wird eine Liste der möglichen Daten-Namen gezeigt, welche diesen Wert aufnehmen könnten. In einem kleinen Programm mag diese Funktion nicht so

wichtig sein, aber in einem großen Programm, mit vielen Daten Namen, kann diese Fähigkeit sehr nützlich sein.

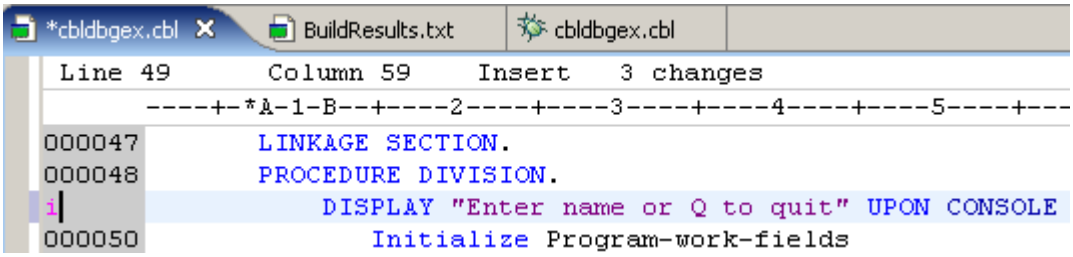


Abb. 3.8.13

Type “i “in den grauen Bereich auf der linken Seite.

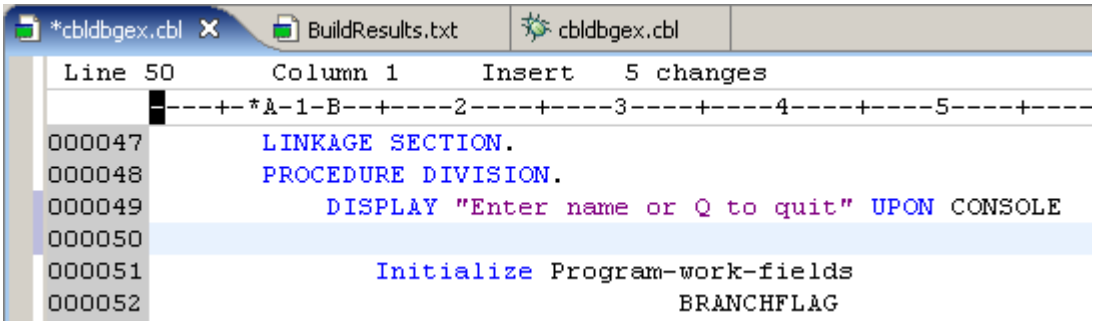


Abb. 3.8.14

Enter, um eine Leerzeile zu erzeugen. (Dies könnte mit CTRL + ENTER geschehen, wenn wir den LPEX _Editor an Stelle des ISPF Editors benutzen würden).

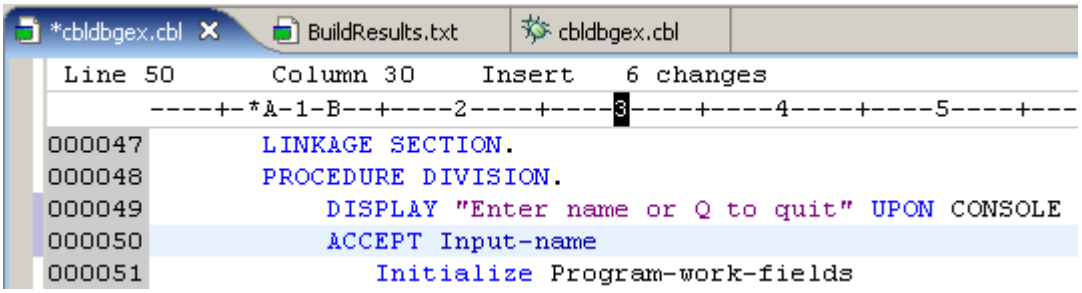


Abb. 3.8.15

2 x Tab, Type ein ACCEPT Statement. Benutzen Sie CTRL + Space, um den Input-Namen einer Variablen hinzuzufügen.

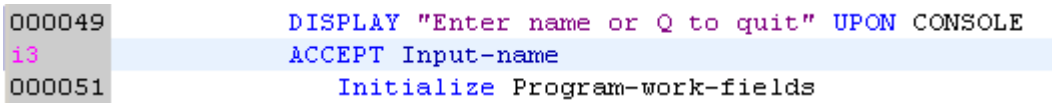


Abb. 3.8.16

Type i3 auf Zeile 50

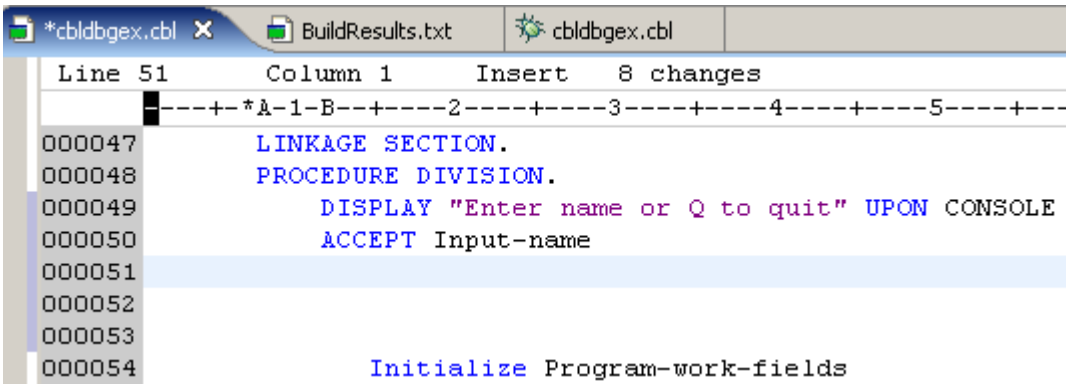


Abb. 3.8.17

Enter drücken um 3 Leerzeilen einzufügen.

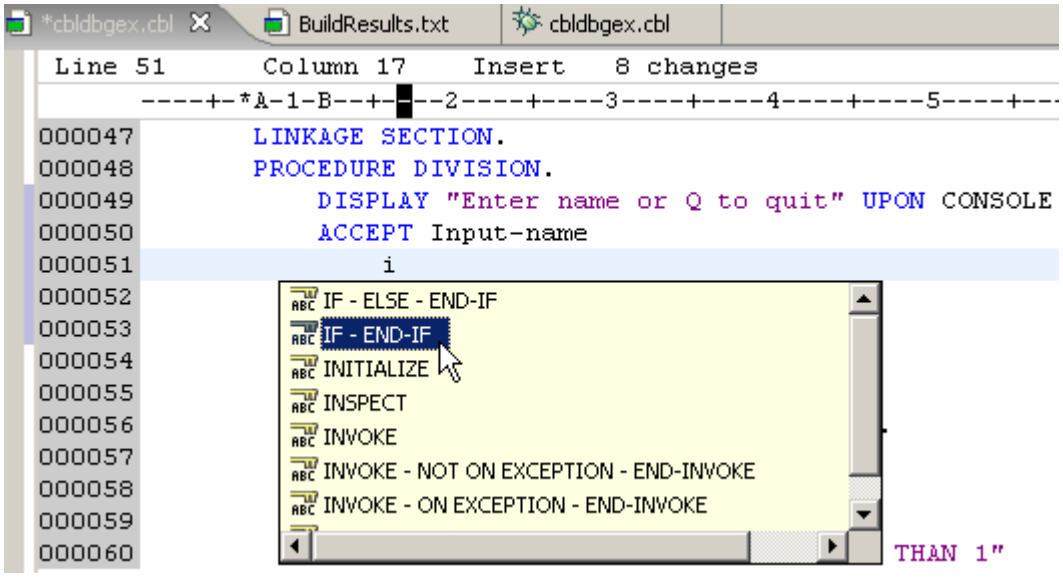


Abb. 3.8.18

Ein IF – END-IF Statement einfügen durch Eingabe eines “i”; anschließend “IF - END-IF” von dem Content Assist Dropdown Menu selektieren.

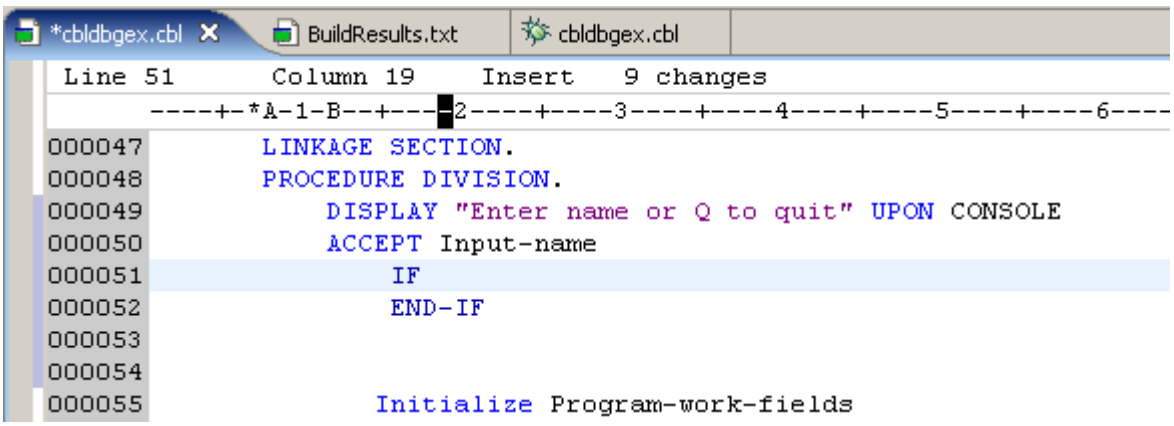


Abb. 3.8.19

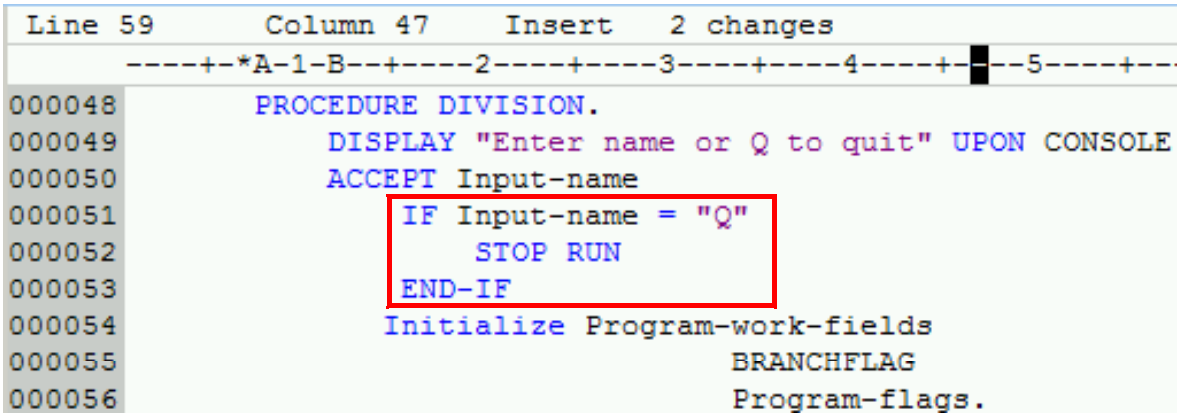


Abb. 3.8.20

Statement vervollständigen. Neuen Text in Zeilen 51 - 53 eingeben, wie hier dargestellt.

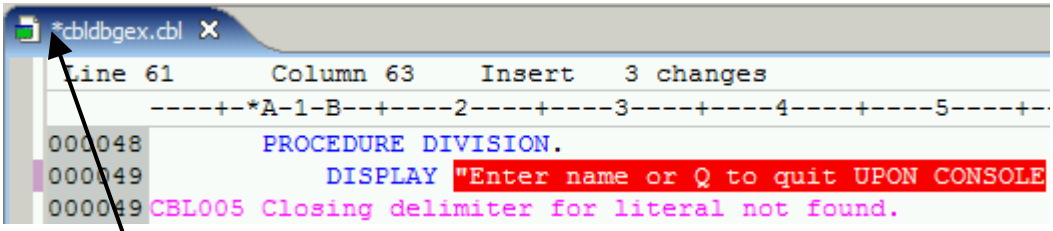


Abb. 3.8.21

Weil der COBOL-Editor ein Smart-Editor ist, können einige Fehler beim Kodieren gefunden werden. Zum Beispiel, wenn wir die ersten Anführungszeichen (") Im Statement DISPLAY beseitigen und drücken die Eingabetaste (oder bewegen den Cursor auf einer andere Zeile), erhalten wir eine Fehler Meldung wie oben angezeigt.

Korrigieren Sie den Fehler, und speichern Sie mit CTRL + S.

Beachten Sie, das Sternchen * auf der linken Seite des Titels verschwindet nach der Speicherung. 

Machen Sie noch einen lokale Syntax Check, um die Syntax wieder zu überprüfen.

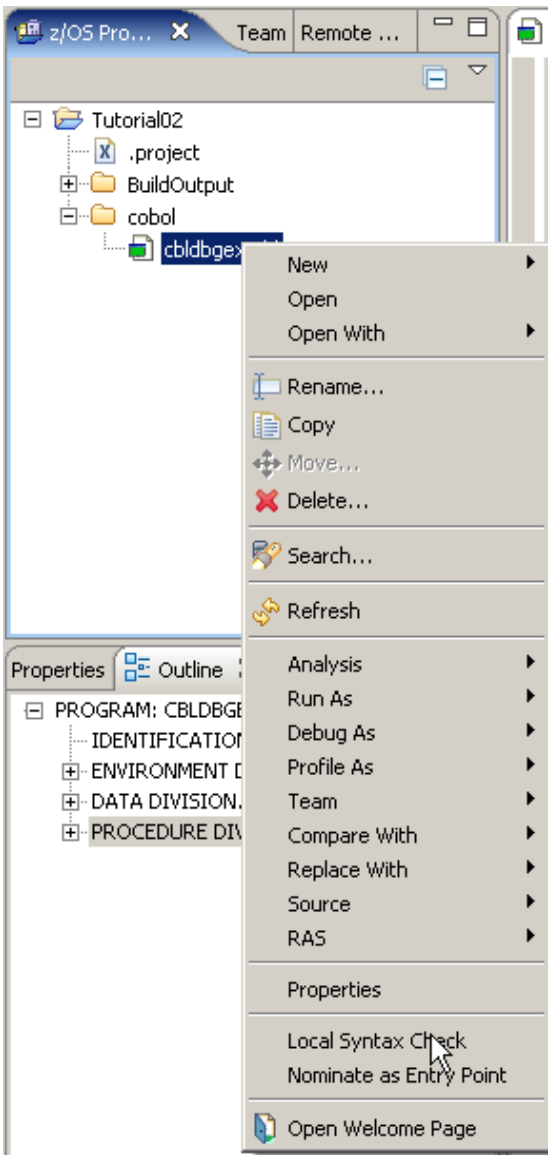


Abb. 3.8.22

1kr auf cbldbgex.cbl. Selektiere Local Syntax Check, 1k.

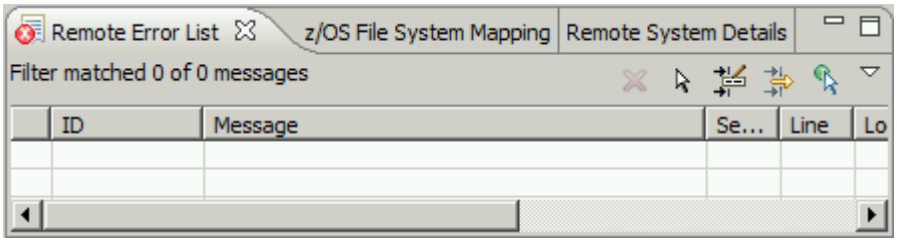


Abb. 3.8.23

Klicken Sie in die Ansicht Remote Error List. Es sollten keine Fehler in dieser Ansicht aufgeführt sein. Wenn Sie Fehler haben, korrigieren Sie diese und überprüfen Sie die Syntax wieder.

Hinweis: Falls Sie Fehler haben und Zeit für die Korrektur nicht verlieren wollen, laden Sie einfach den Code mit den genannten Änderungen aus dem Ordner c: \ wdz Tutorials \ Ressourcen \ Tut02solution. Verwenden Sie einfach File → Import → File system → Next → und selektieren sie die Datei cbldbgex.cbl

Ein weitere nette Hilfe bei der Codierung des Programms ist die Help Built-in Funktion.

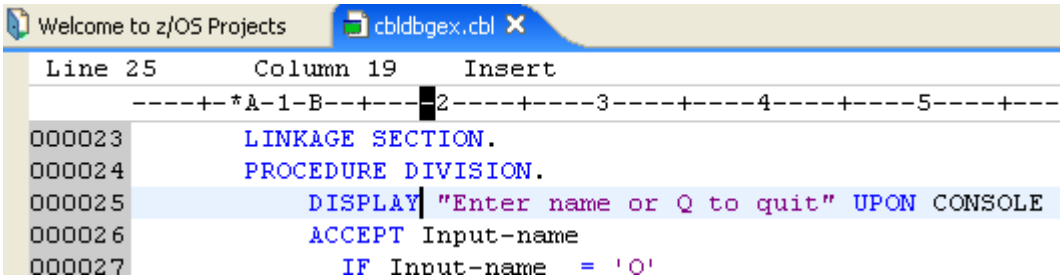


Abb. 3.8.24

Bewegen sie den Cursor zu dem Statement DISPLAY und drücken Sie F1. Ein Help Fenster wird geöffnet und zeigt das selektierte Statement. Dies kann für jedes beliebige Statement benutzt werden.

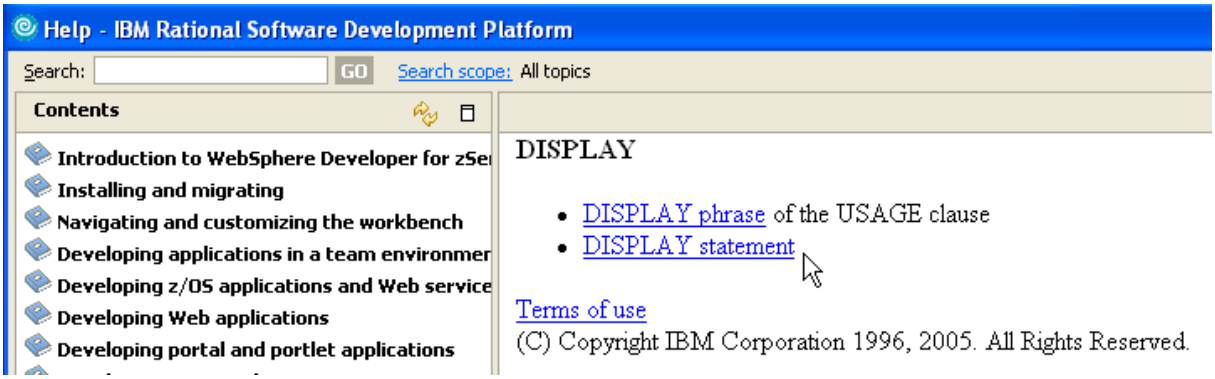


Abb. 3.8.25

Nach einiger Zeit wird das Help Fenster geöffnet. In unserer VMWare Umgebung kann dies etwas länger dauern.

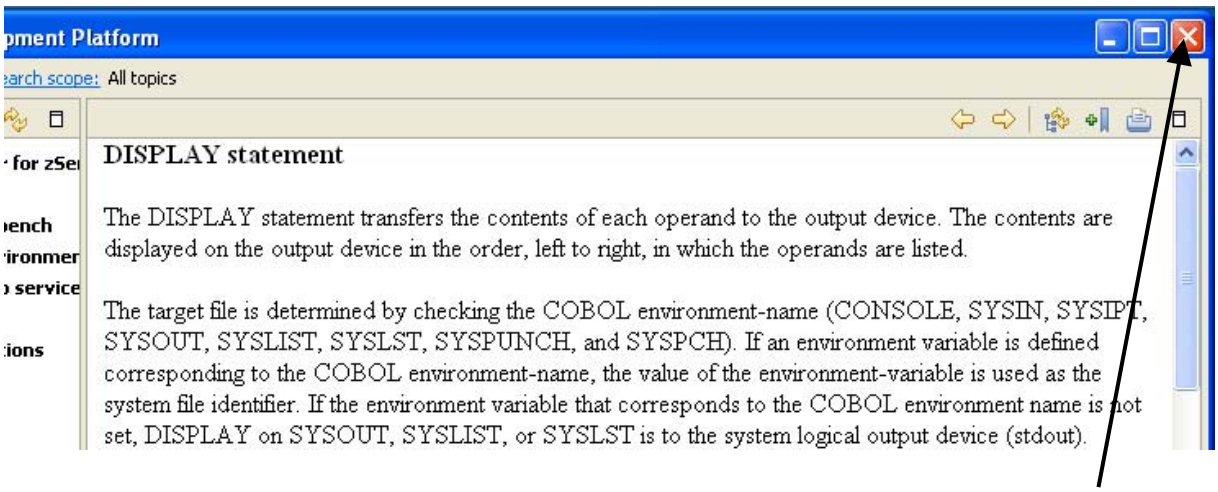


Abb. 3.8.26

1k auf das DISPLAY Statement produziert mehr Information. Das Help Window wieder schließen

4. Kompilieren, Linken und Ausführen des lokalen COBOL-Programms

Nachdem Sie Ihre Quellcode fertigt gestellt haben, führen Sie nun Ihr Programm aus. Es wird in der lokalen COBOL-Laufzeitumgebung (d.h. unter Windows) compiled und ausgeführt. Darüber hinaus werden Sie den Debugger verwenden, um den Code zur Laufzeit Step für Step auszuführen.

4.1. Erstellen eines ausführbaren COBOL-Programms

Vor der Übersetzung und dem Erstellen der ausführbaren Datei, löschen Sie zunächst evtl. von früher her vorhandenen ausführbaren Code.

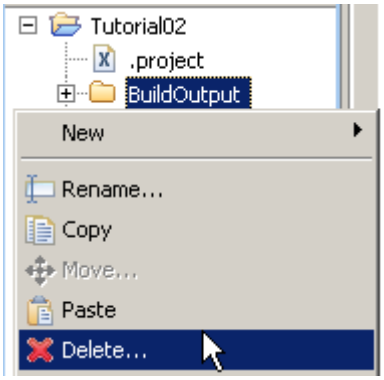


Abb. 4.1.1

Der Folder „BuildOutput“ enthält allen übersetzten und ausführbaren Code. BuildOutput selektieren, 1kr. Das Context Menu zum Löschen benutzen.

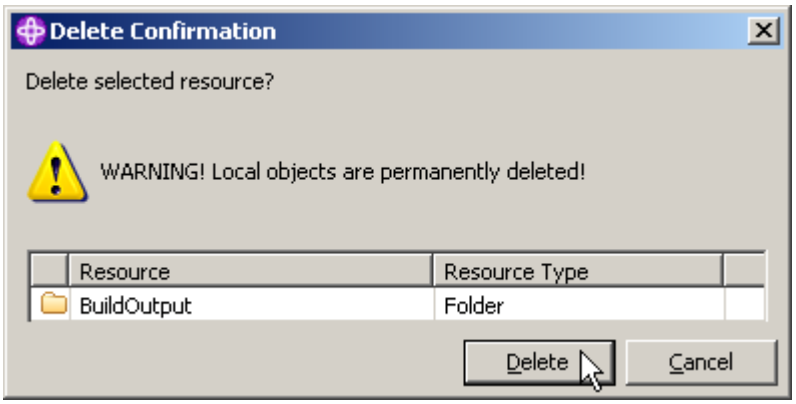


Abb. 4.1.2

Das Delete Confirmation Window erscheint. 1k Click auf Delete

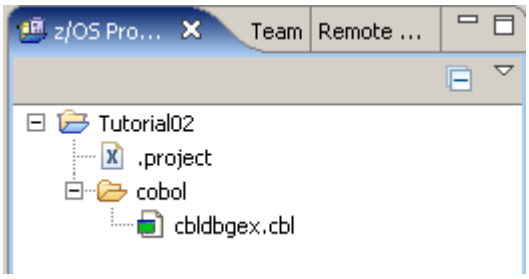


Abb. 4.1.3

Hinweis: Wir löschen den BuildOutput Ordner nur um zu zeigen, dass er neu erstellt wird. Im Augenblick wäre dieser Schritt vermutlich nicht wirklich notwendig.

To Build the program:

Wir benutzen den z/OS Projects View. Expandiere COBOL. 1kr auf cbldbgex.cbl, und

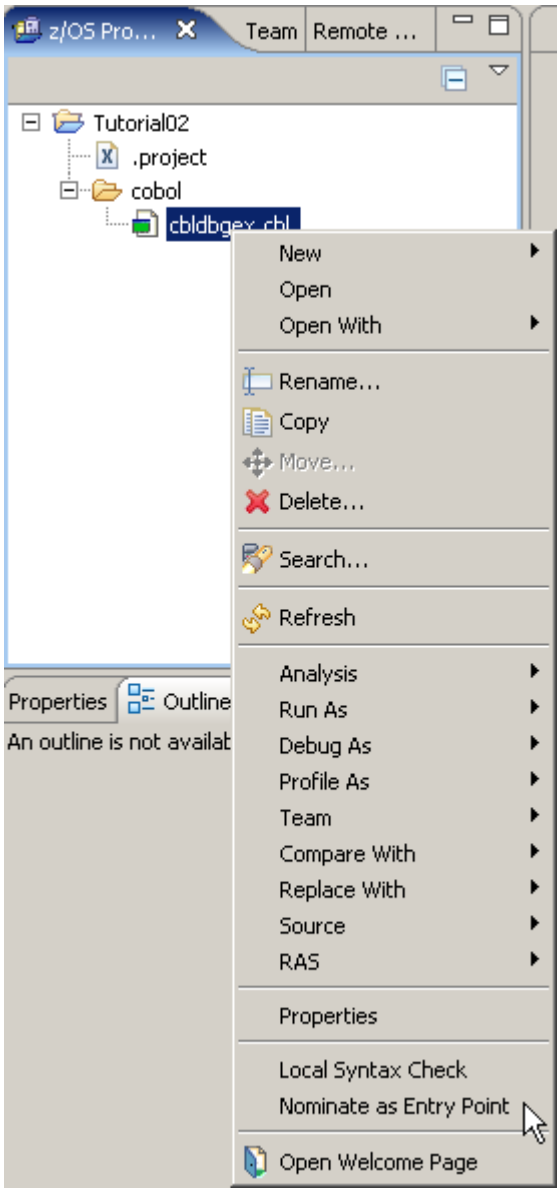


Abb. 4.1.4

selektiere “Nominate as Entry Point”. 1k.

Wenn Sie das nicht tun, und haben mehrere Programme im selben Projekt, wird eine DLL mit dem Projekt-Namen erzeugt. Das ist hier nicht der Fall, aber es ist besser, diesen Schritt immer auszuführen.

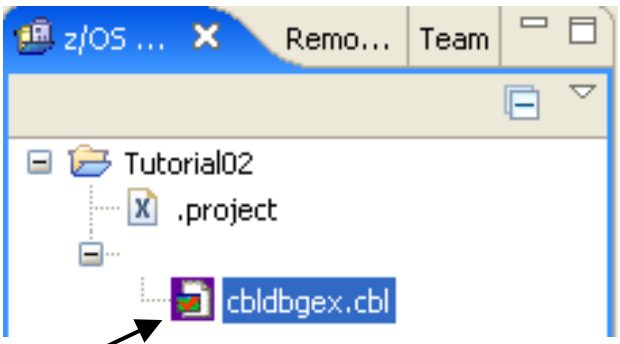


Abb. 4.1.5

Ein rotes Häkchen erscheint in dem Icon für das Programm, sobald dieses als Einstiegspunkt (Entry Point) nominiert worden ist.

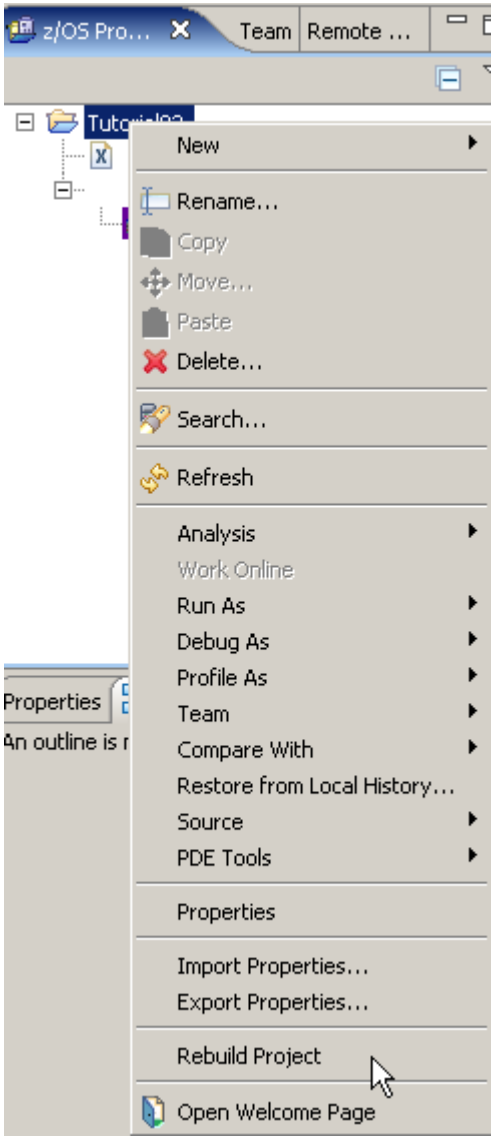


Abb. 4.1.6

Um das Programm zu erstellen, 1kr auf Projekt Tutorial02. Selektiere Rebuild Project, 1k.

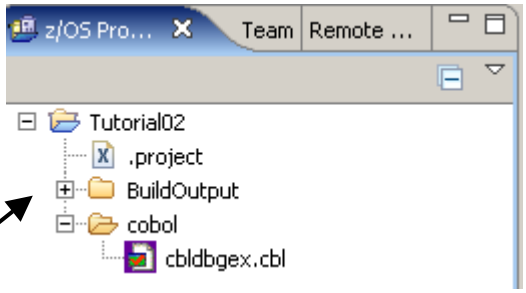


Abb. 4.1.7

Beachten Sie, BuildOutput wird neu erstellt.

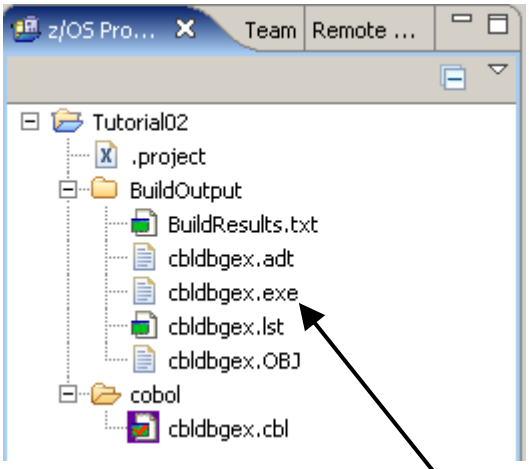


Abb. 4.1.8

Expandiere BuildOutput um den Ordner Inhalt anzuzeigen. Eine exe-Datei wurde erstellt. Optional kann eine DLL erstellt werden (dies ist z.B. für CICS TS Programme erforderlich).

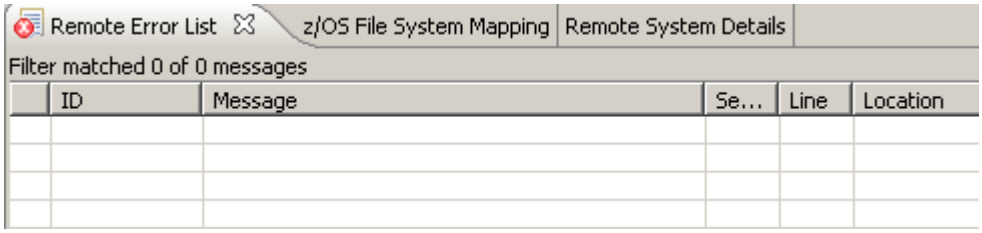


Abb. 4.1.9

Es sollten keine Fehler im Remote Error List View aufgeführt werden.

4.2. Überprüfen Sie die lokale COBOL Compilation und Link Edit

Der COBOL-Compiler speichert die Ausgabe im BuildOutput Ordner.

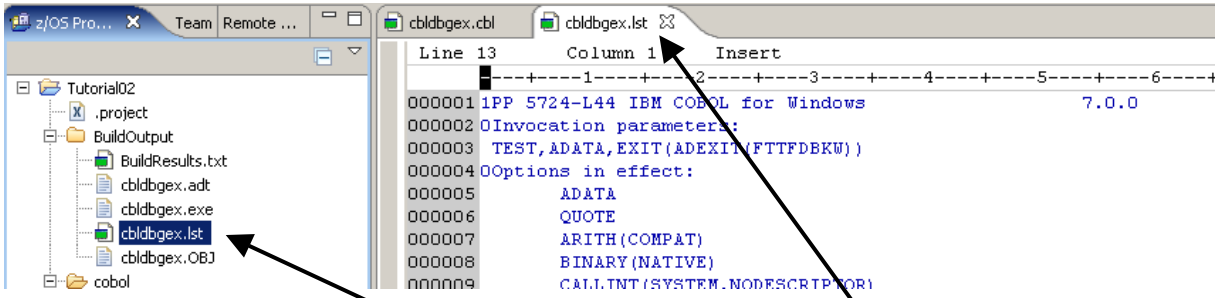


Abb. 4.2.1

Um das compile Listing anzusehen, 2k auf cbldbgex.lst. Ein weiterer Tab erscheint.

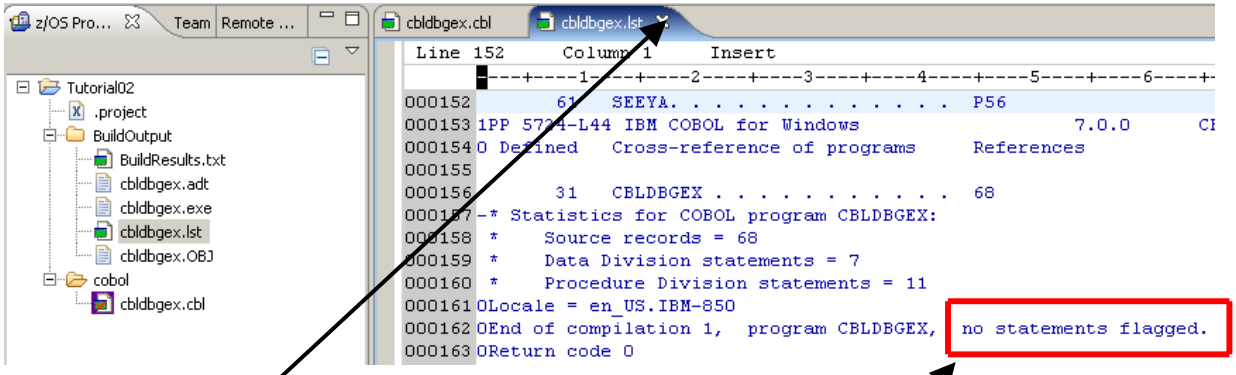


Abb. 4.2.2

Scroll down zum Ende des Listing. Es werden keine Compile Fehler angezeigt. Benutzen Sie die Ctl + End Tasten-Kombination “to go to the end”.

Schließen Sie den Editor.

4.3. Erstellen einer Run Launch Konfiguration, und Ausführen des COBOL-Programms.

Da wir dieses Programms viele Male ausführen wollen, es ist eine gute Idee, eine Debug-Launch-Konfiguration zu erstellen. Dies erleichtert das Ausführen und Debuggen eines spezifischen Programms. Der folgende Schritt erstellt eine Launch Konfiguration, welche das kompilierte COBOL-Programm lädt:

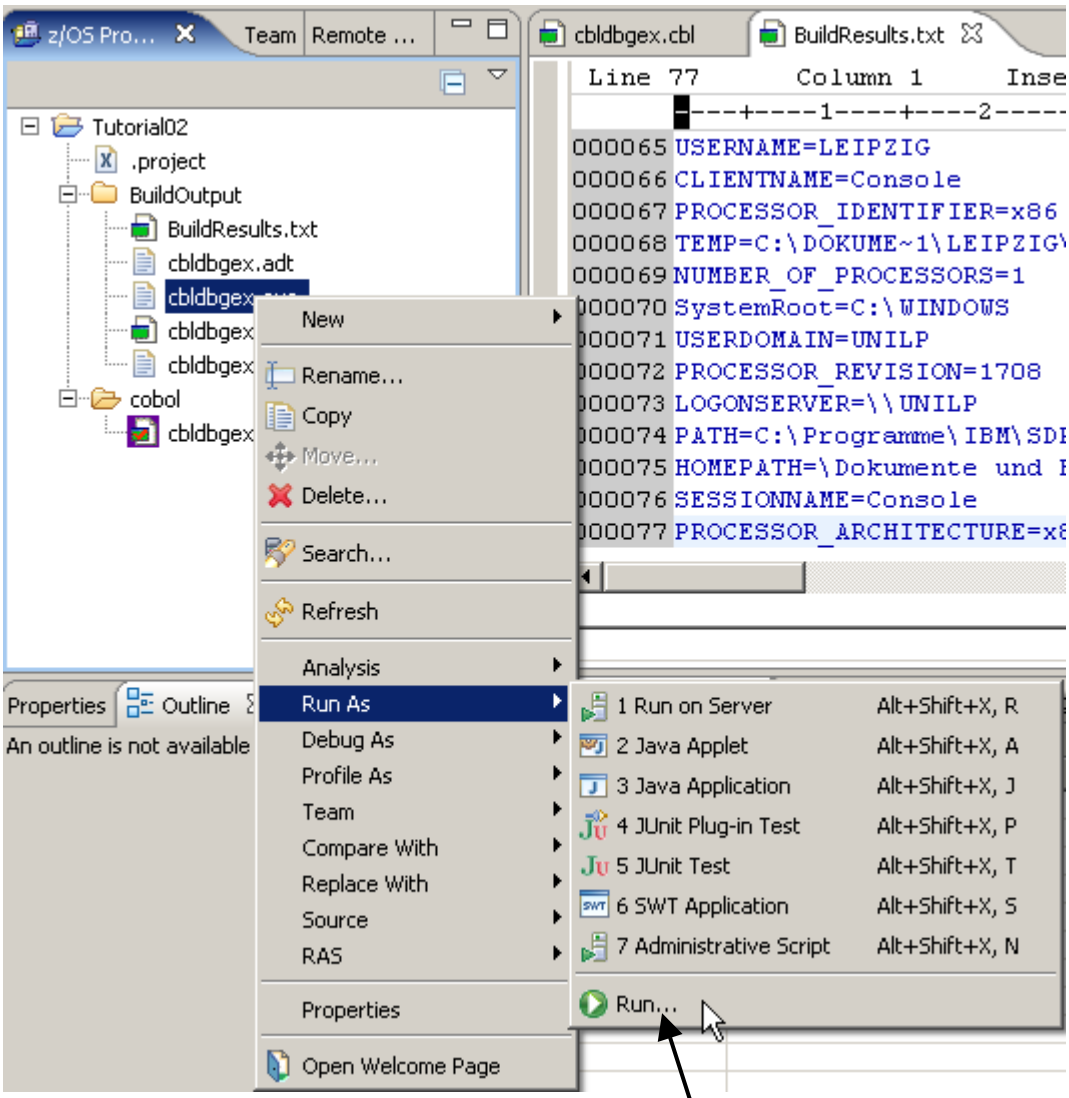


Abb. 4.3.1

1kr auf cbldbgex.exe. Selektiere Run As → Run… (<u>nicht</u> Run on Server)

1k

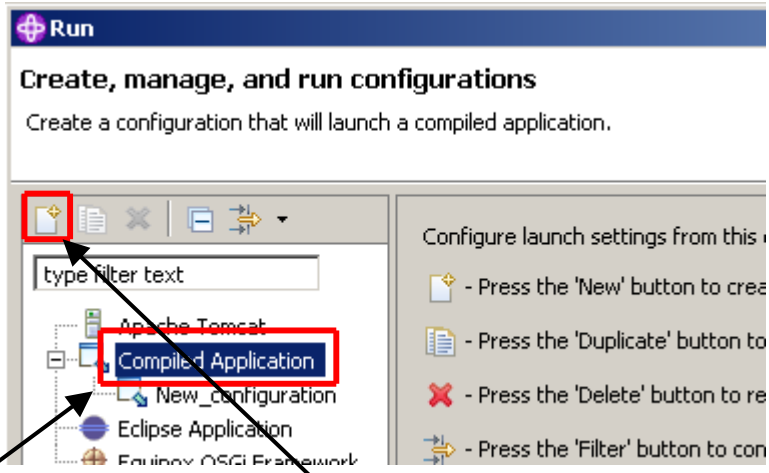


Abb. 4.3.2

Selectiere Compiled Application, 1k auf den New Button.

Als Folge erscheinen die Launch Configuration Tabs und Eingabefelder auf der rechten Seite der Dialog Box.

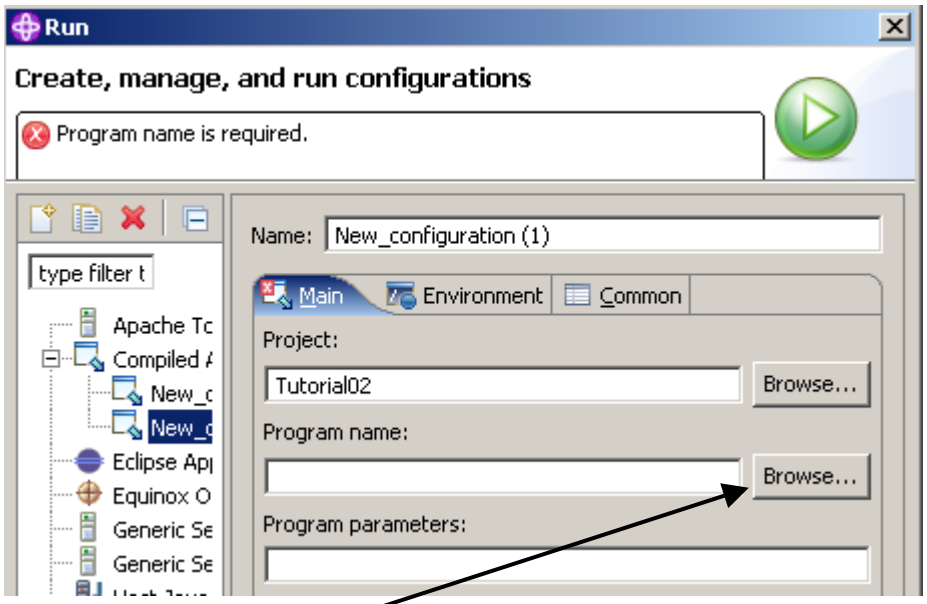


Abb. 4.3.3

Benutzen Sie den Browse Button neben Program name, um cbldbgex.exe im BuildOutput Verzeichnis zu finden, und …

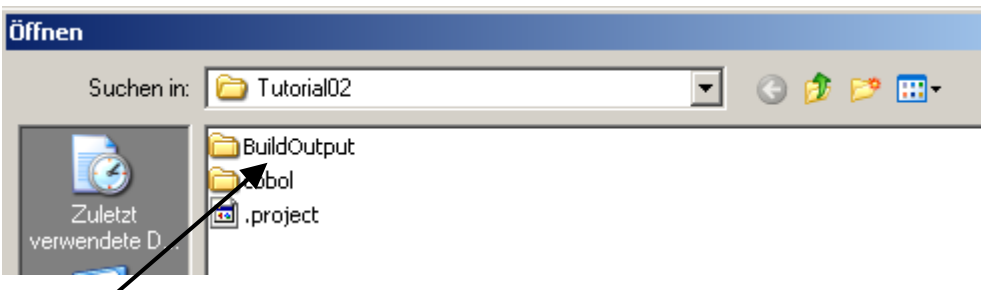


Abb. 4.3.4

1k auf BuildOutput Folder um ihn zu öffnen, 2k .

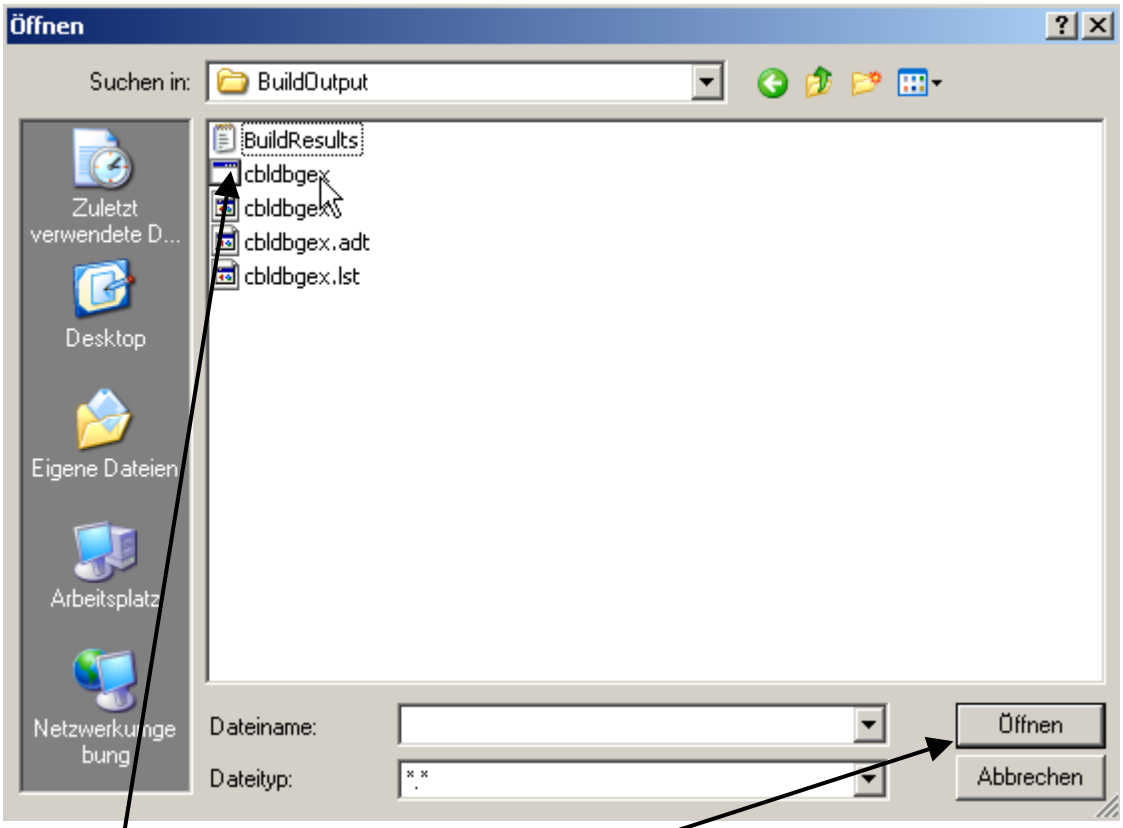


Abb. 4.3.5

Selektiere cbldbgex.exe. 1k auf open.

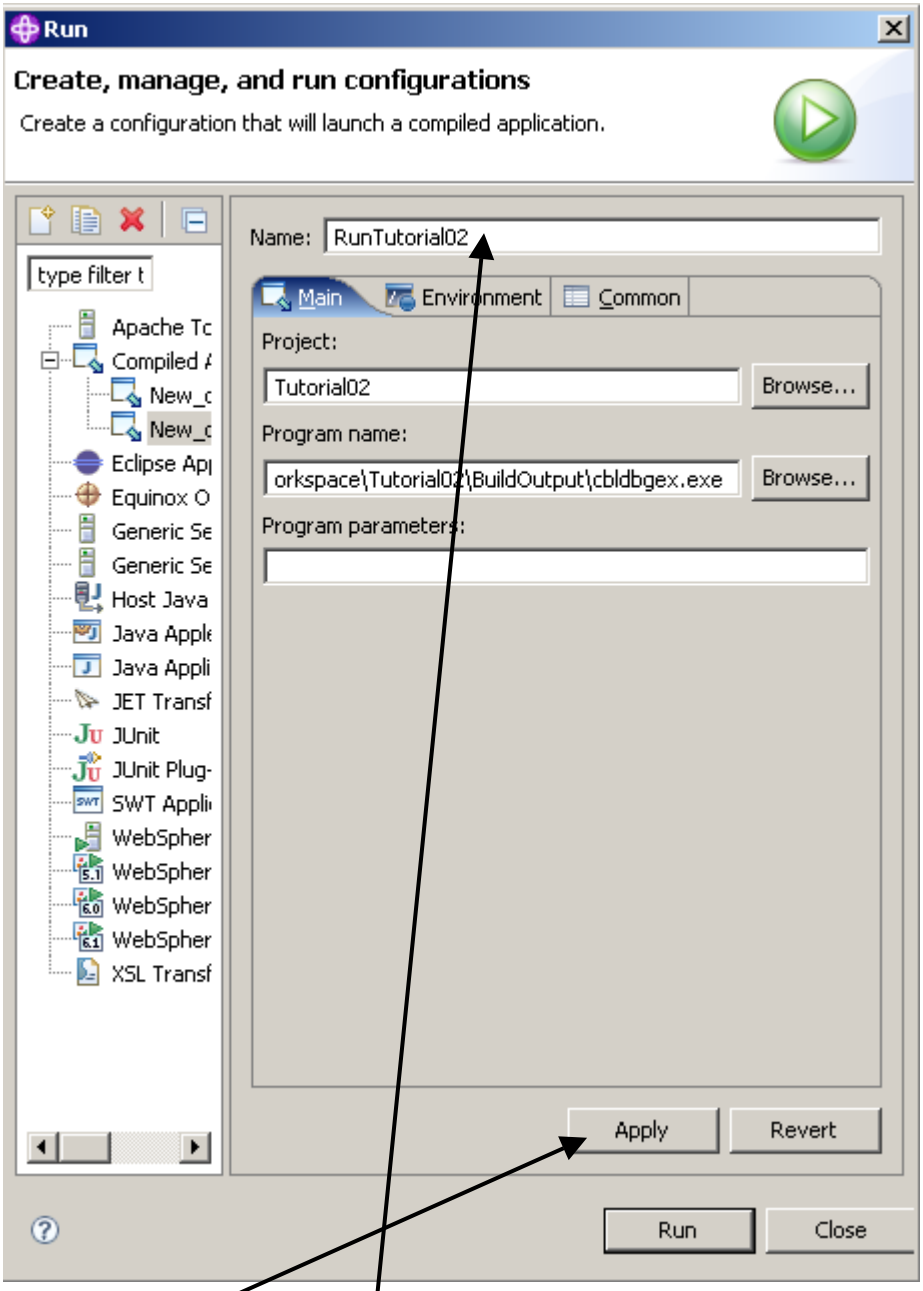


Abb. 4.3.6

In den Name Feld, einen Namen wie z.B. „RunTutorial02“ eingeben.

1k auf „Apply“ rettet (saves) die Änderungen. Jetzt…

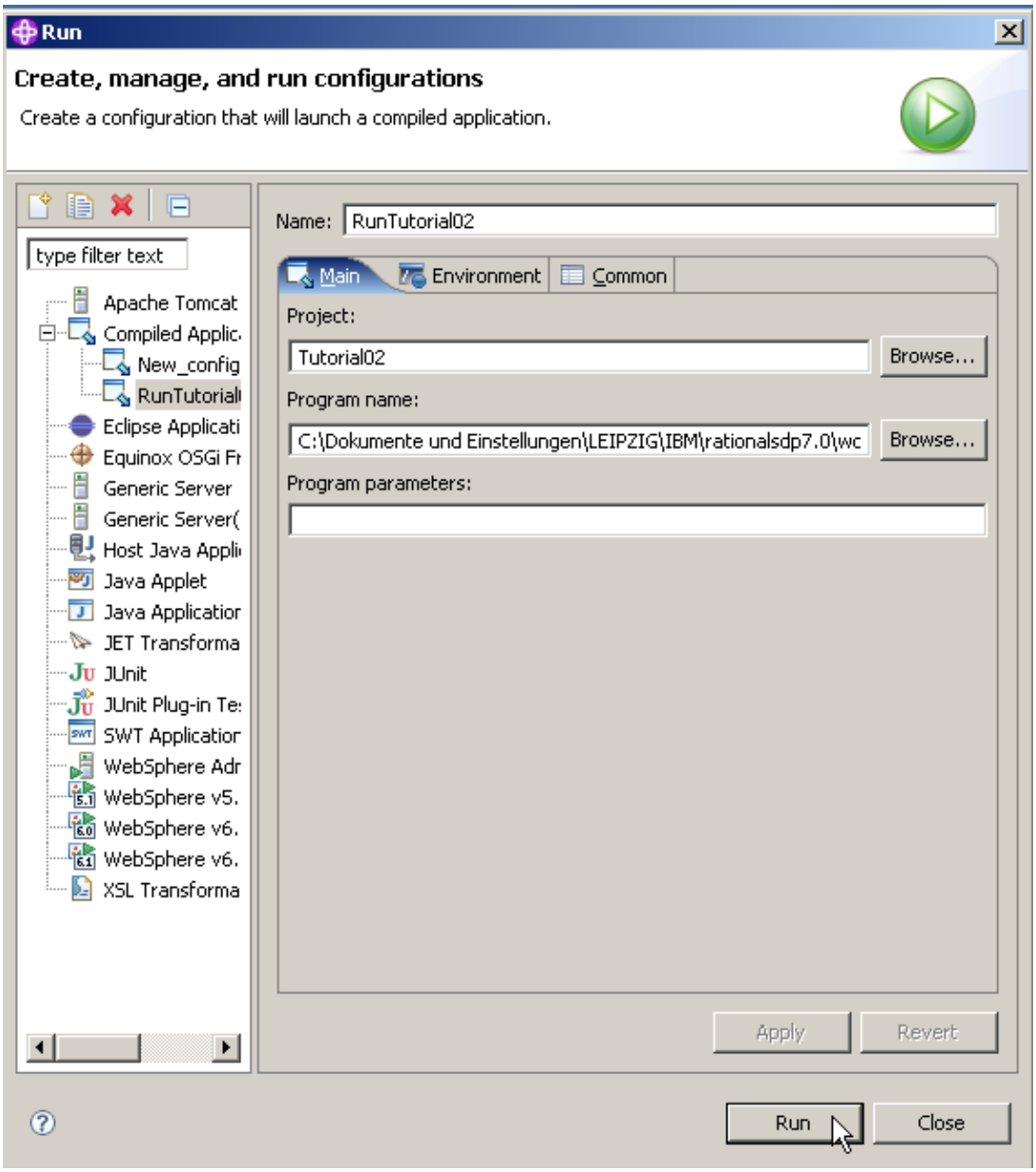


Abb. 4.3.7

Einen Namen wie RunTutorial02 eingeben. 1k auf Run.

Die Ausführung beginnt und normalerweise würden Sie diesen Dialog in einem neu geöffneten Konsole Fenster sehen:

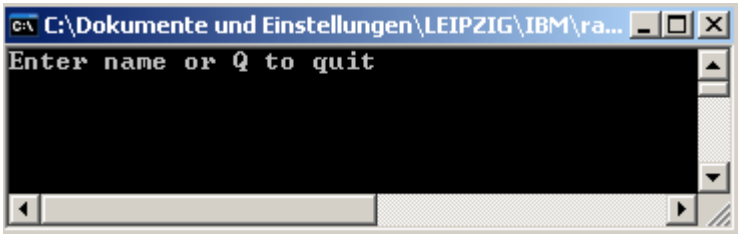


Abb. 4.3.8

Einen Namen eingeben, und …..

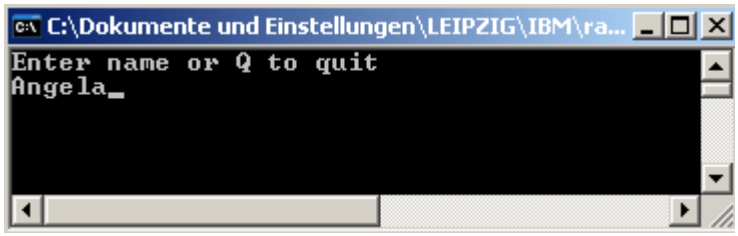


Abb. 4.3.9

Enter eingeben.

Eine Antwort wird zwar korrekt wiedergegeben, aber das Konsole Fenster wird wieder geschlossen, ehe Sie Zeit haben, etwas zu sehen.

Aufgabe: *Erweitern Sie das Cobol Beispiel Programm, so dass das in Abb. 4.3.9 dargestellte Fenster permanent sichtbar bleibt, und z.B. durch Enter geschlossen wird.*

Aufgabe: *Senden Sie einen entsprechenden Screen shot an Ihren Betreuer.*

Aufgabe: *Als Option machen Sie sich mit den in Abschnitt 5 beschriebenen RDz Test/Debug Funktionen vertraut.*

Selbst-Test

- **Warum ist es sinnvoll, eine Debug-Launch-Konfiguration zu erstellen ?**

5. Testing/Debugging des COBOL Programms

5.1 Debug Perspektive

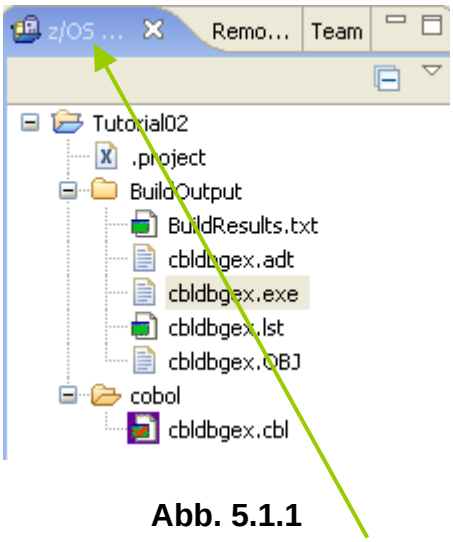


Abb. 5.1.1

Zur Erinnerung: Eclipse und RDz benutzen das Konzept einer Perspective. Derzeitig benutzen wir die z/OS Projects Perspective. Dies wird in dem lokalen Fenster angezeigt.

Eine Perspective definiert die Benutzerschnittstelle (GUI), mit der Sie arbeiten.

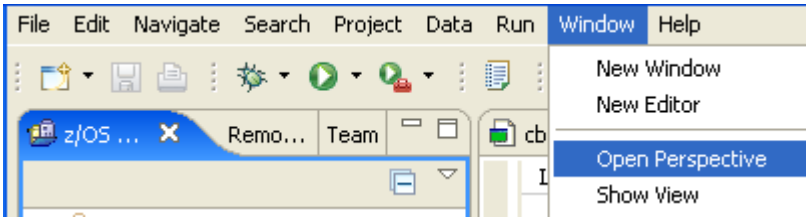


Abb. 5.1.2

Sie können jederzeit die gerade benutzte Perspektive wechseln: 1k auf Window → open Perspective. Am wichtigsten ist: Wenn Sie den Überblick verloren haben, können Sie immer zu der jetzt vertrauten z / OS Projects Perspektive zurückzukehren, einfach durch 1k Window → Open Perspective.

OK, fangen wir mit dem Testen und Debuggen unseres Cobol-Programms an. Hierfür werden wir schon bald in die Debug-Perspektive wechseln, und später, wenn wir fertig sind, zurück zu der z/OS Projects Perspektive.

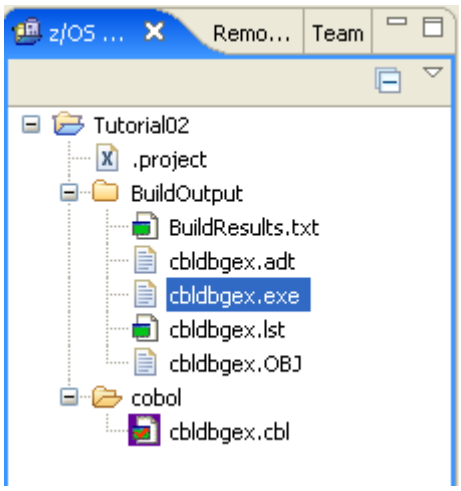


Abb. 5.1.3

Da wir die Launch Configuration bereits erstellt haben (siehe Abschnitt 4.3), 1kr auf cbldbgex.exe, und....

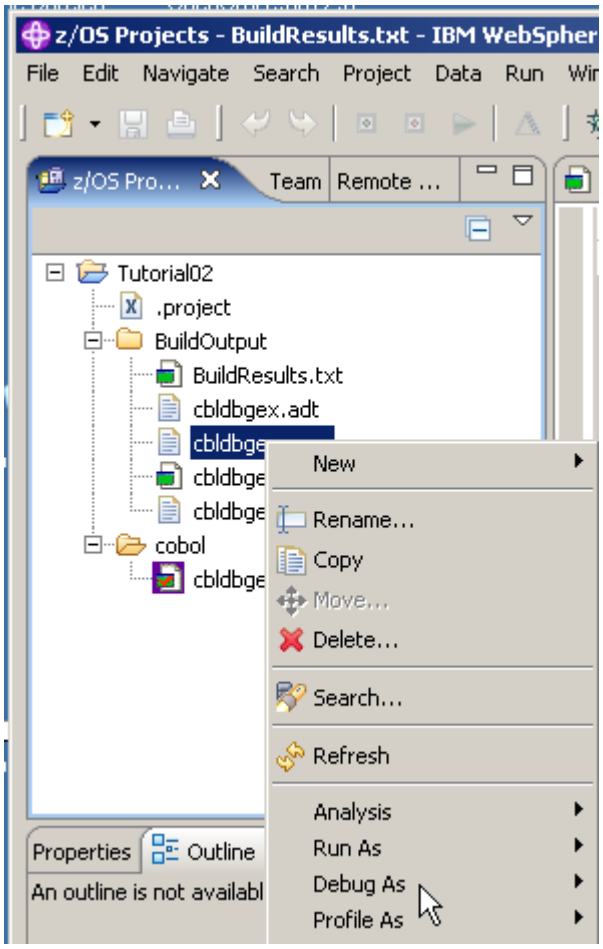


Abb. 5.1.4

selectiere “Debug As” und dann Debug.

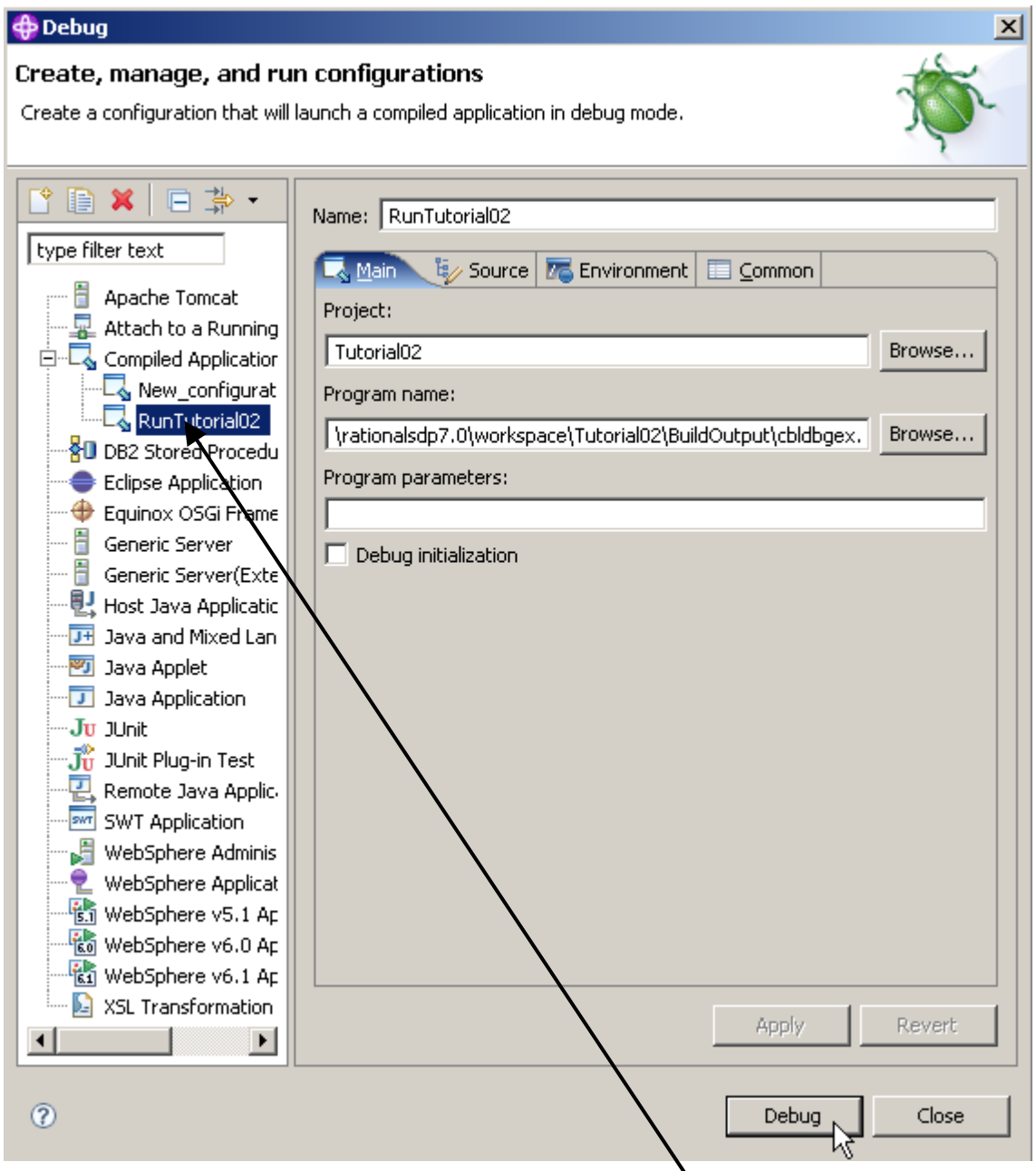


Abb. 5.1.5

Wenn das Debug Window geöffnet wird, selektieren Sie die in Abschnitt 4.3 erstellte Launch Configuration RunTutorial02 und click Debug.

In dem folgenden Dialog Fenster wird gefragt, ob Sie die Debug-Perspektive wechseln wollen. Derzeitig arbeiten sie mit der z/OS Projects Perspektive. Das Umschalten der Perspektiven ist ein normaler Vorgang, wenn Sie RDz benutzen. Sie können immer wieder zur z/OS Projects Perspektive zurückkehren,, indem Sie auf Window → open perspective → z/OS Projects Perspective klicken.

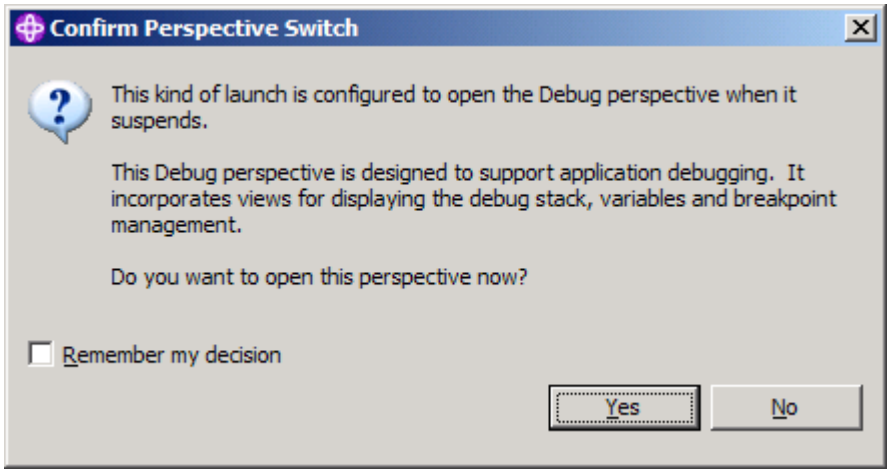


Abb. 5.1.6

Click Yes. Dies schaltet von der z/OS Projects Perspective auf die Debug Perspective um.

Beachten Sie, das obige Dialog Fenster ist möglicherweise hinter dem Konsole/DOS Fenster versteckt. Möglicherweise müssen Sie das Konsole-Fenster minimieren, um den Dialog zu sehen.

Die Debugger *P*erspective wird geöffnet und das Debug beginnt.

5.2 Debug Steps

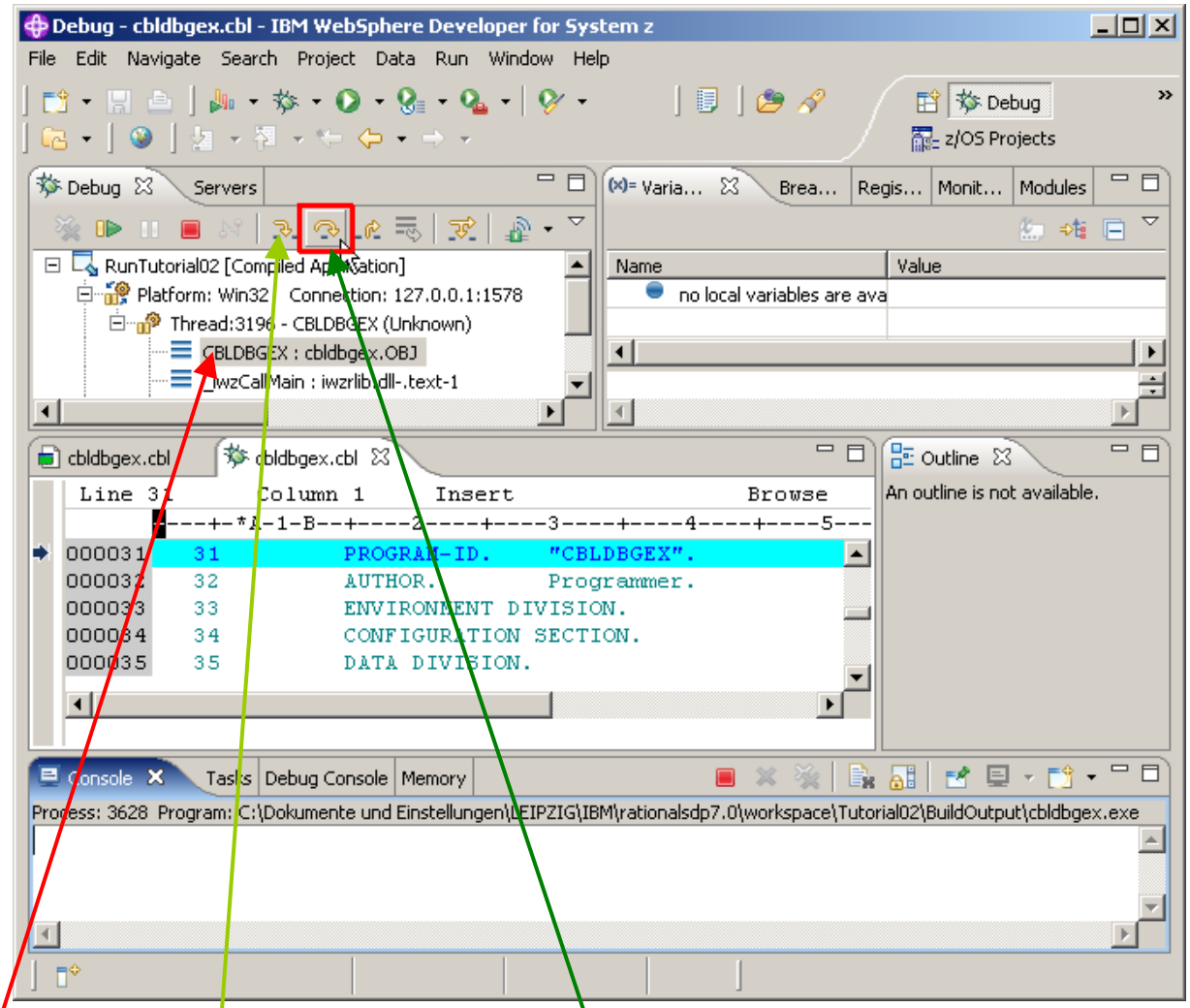


Abb. 5.2.1

Dieser Eintrag sollte hervorgehoben sein.

In der Debug-Ansicht, click auf das Step Over Icon (oder alternativ hit F6 zwei mal). (Verwenden Sie kein Step Into)., Wenn Sie während dieser Schritte etwas falsch gemacht haben, schließen Sie die Debug-Perspektive und die gerade bearbeitete Datei wieder, und fangen Sie mit Abschnitt „5. Testing/Debugging des COBOL Programms“ erneut von vorne an.

Denken Sie daran, Sie sind jetzt in der Debug-Perspektive. Wenn Sie zu Abschnitt „5. Testing/Debugging des COBOL Programms“ zurückkehren wollen, click auf Window – open Perspective - z/OS project.

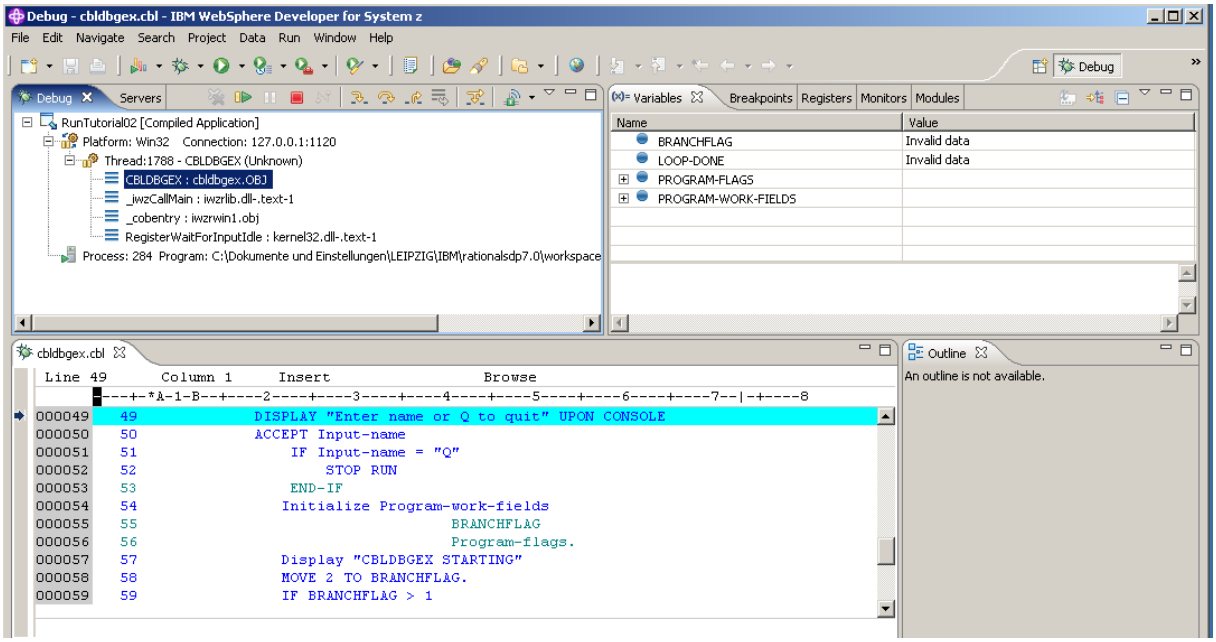


Abb. 5.2.2

Nach einem click auf das Step Over icon ( ) , startet die Ausführung des COBOL Programms. Sie sehen etwas ähnliches wie den hier gezeigten Screen.

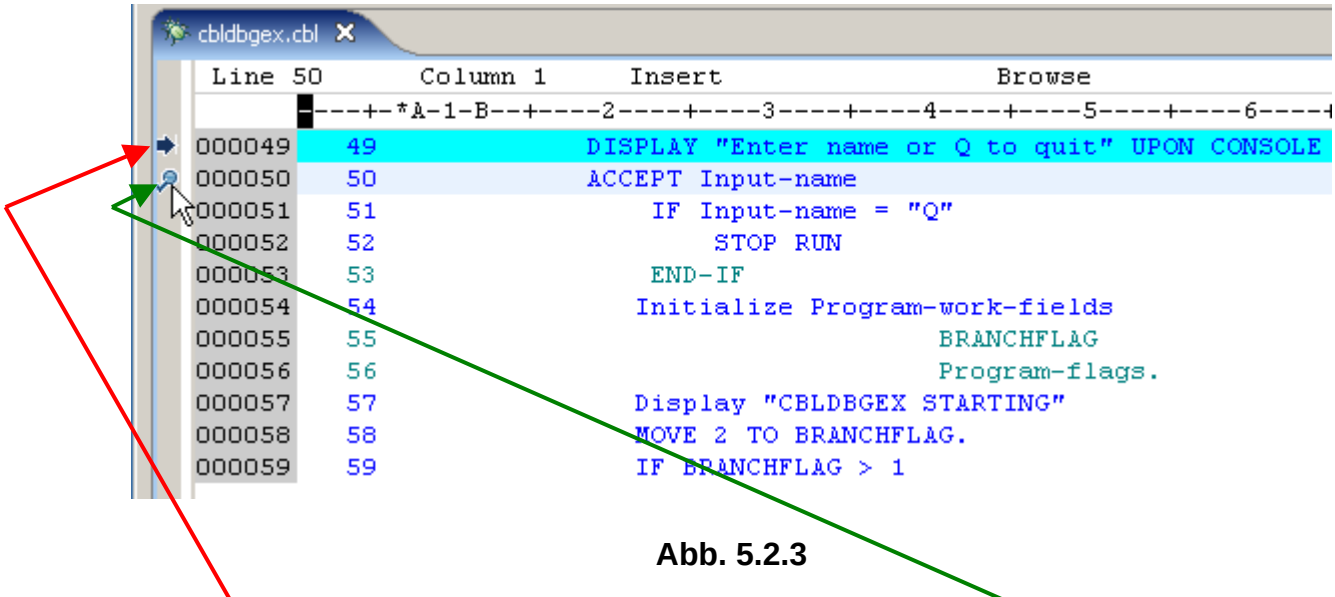


Abb. 5.2.3

Dieser kleine Pfeil zeigt an, dass die Programmausführung nach der Ausführung dieser Zeile gestoppt wurde.

Fügen Sie einen Breakpoint in das ACCEPT Statement ein. Hierzu bewegen Sie den Mauszeiger auf den grauen Bereich vor der Zeilennummer, und klicken Sie zwei mal. Ein kleiner Kreis zeigt einen Haltepunkt (Breakpoint) in Zeile Nr. 000050 an.

Ein Breakpoint bewirkt, dass die Ausführung eines Programms an der Stelle unterbrochen wird, wo der Haltepunkt gesetzt wurde.

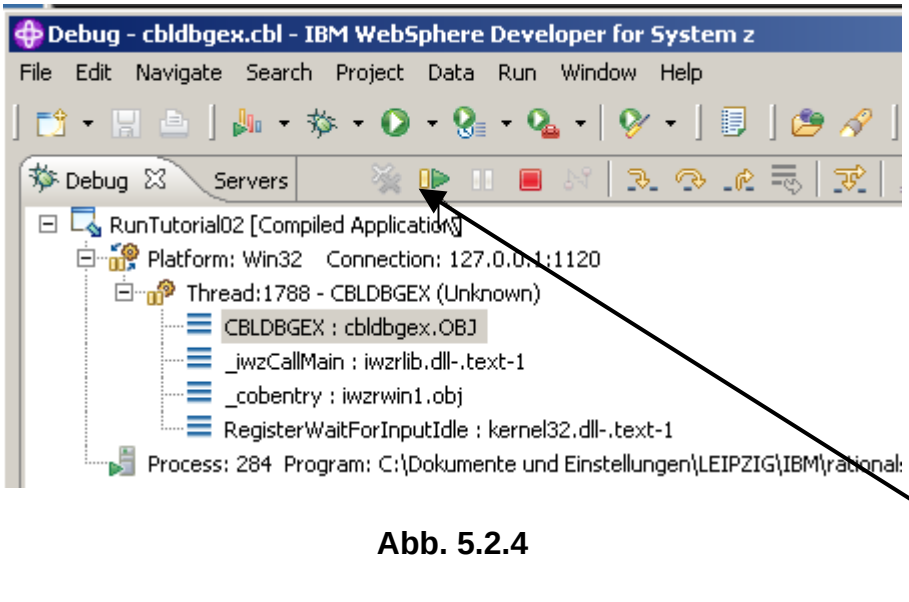


Abb. 5.2.4

Restarten Sie das Programm bis der Breakpoint gefunden wird. Hierzu klick auf das Resume Icon ( ), oder drücke F8.

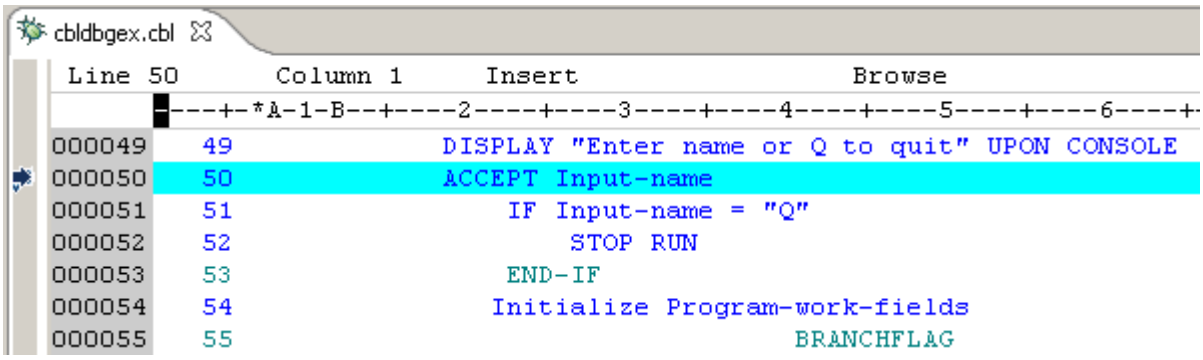


Abb. 5.2.5

Die Programm-Ausführung wird an der ACCEPT-Anweisung in Zeile 000050 stoppen.

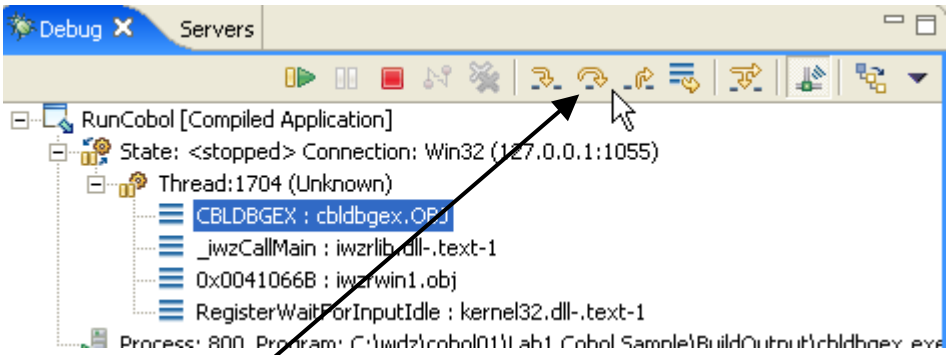


Abb. 5.2.6

Mit dem Step Over Icon (oder F6) ein Statement ausführen..

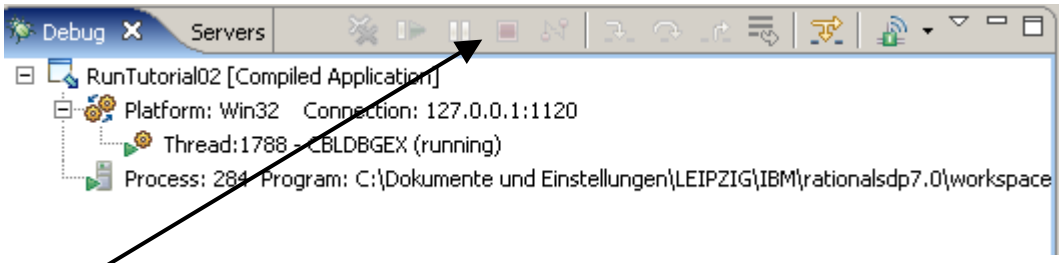


Abb. 5.2.7

Die Icons sind nun deaktiviert (die Symbole sind hellgrau), da Sie zu dem Konsole Fenster gehen müssen, um eine Antwort einzugeben. Wir warten auf die Eingabe in der Konsole.

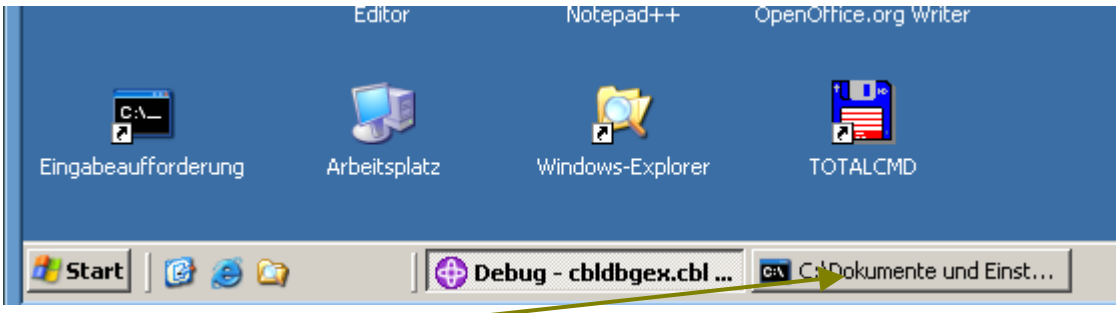


Abb. 5.2.8

A Konsole Window erscheint. Es kann minimiert oder versteckt unter dem RDz Window sein. Klicken Sie auf die Registrierkarte, um es zu öffnen.

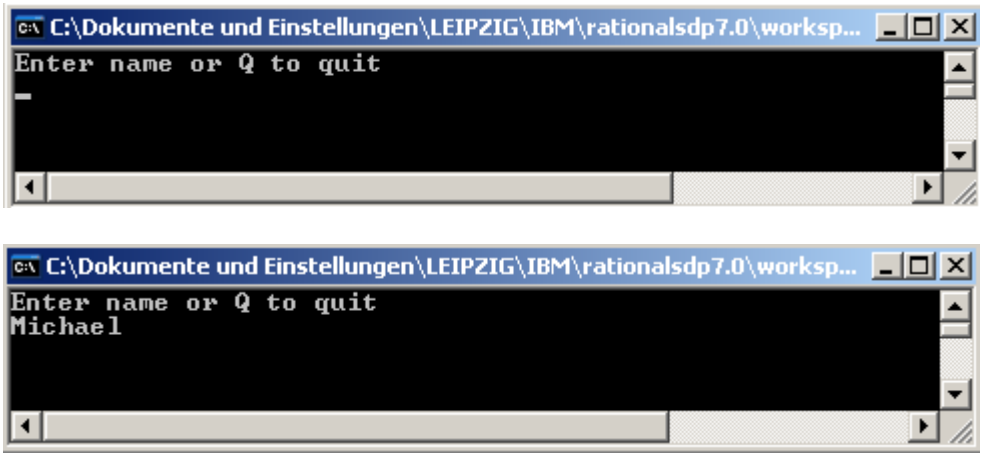


Abb. 5.2.9

Stellen Sie das Konsole-Fenster wieder her, geben Sie einen Namen ein und drücken Sie Enter, und minimieren Sie das Konsole-Fenster (nicht schließen), um mit debug fortzufahren.

5.3 Variablen Werte

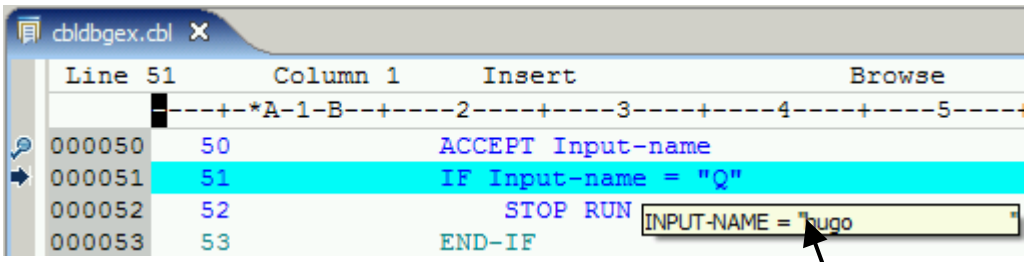


Abb. 5.3.1

Um den Wert einer Variablen zu überwachen (z.B. die Variable INPUT-NAME), bewegen Sie den Maus Zeiger auf den Data Namen, und warten Sie 5 – 10 Sekunden. Der Wert wird wie hier gezeigt wiedergegeben.

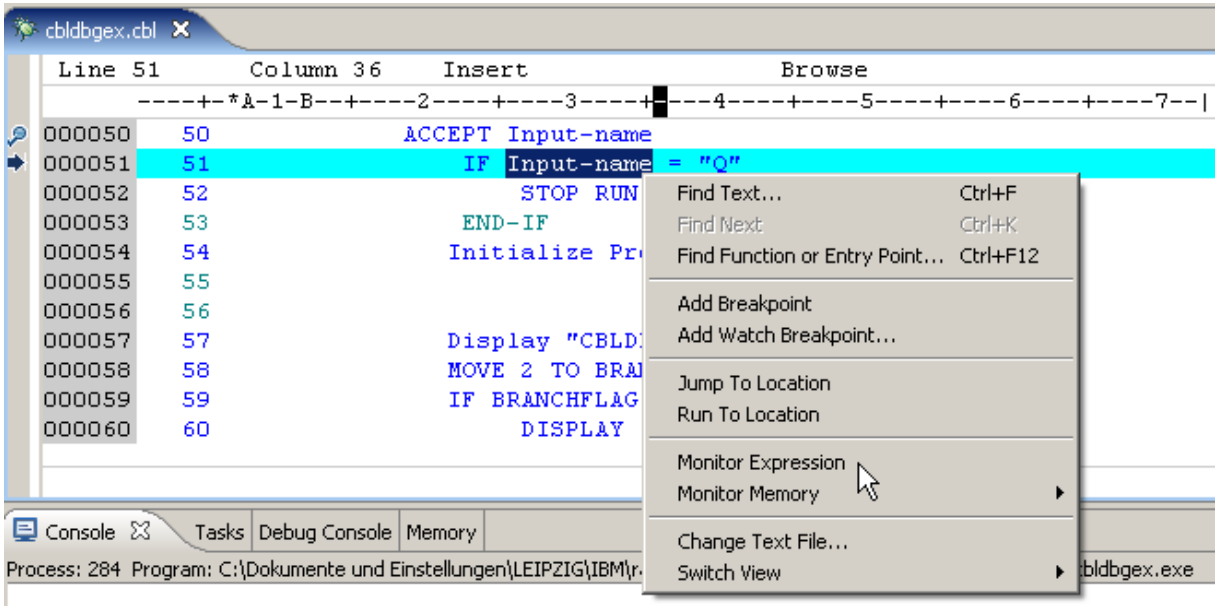


Abb. 5.3.2

Sie können das Feld “Input-name” selektieren, 1kr. Im Kontext menu click “Monitor Expression”.

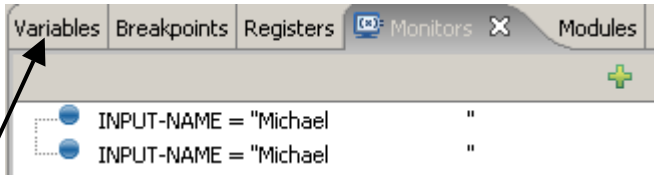


Abb. 5.3.3

In der rechts oberen Ecke wird der Wert der Variablen wiedergegeben. (Wenn Sie dies 2 mal machen, wird der Wert 2 mal wiedergegeben).

1k auf den “Variables” Tab um zum Fenster zurückzukehren.

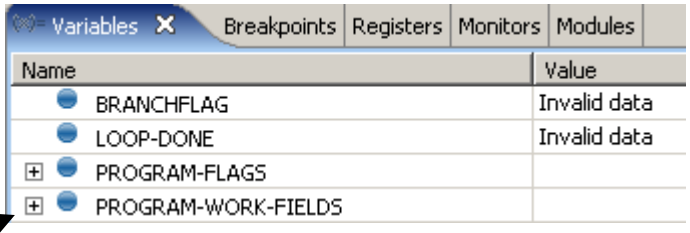


Abb. 5.3.4

Expandiere Program-Work-Fields

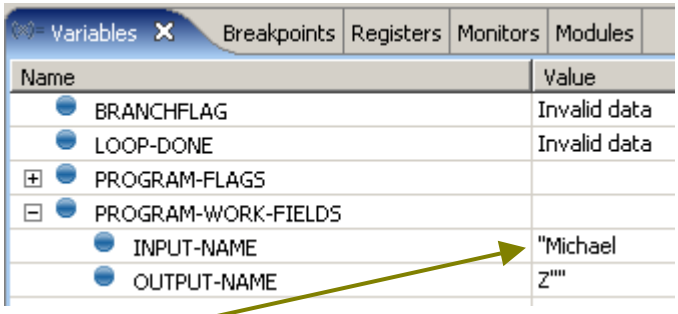


Abb. 5.3.5

Der Wert der Variablen wird dargestellt.

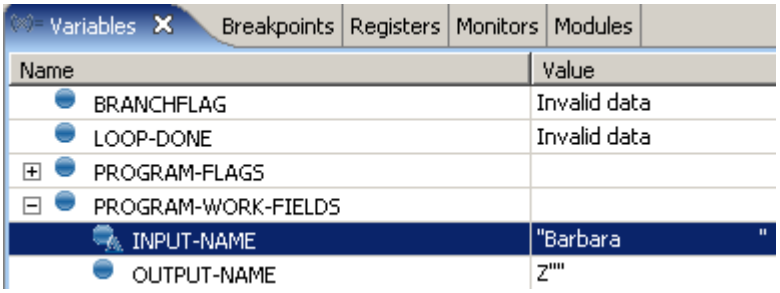


Abb. 5.3.6

Klick auf Input –Name und überschreibe den Wert mit “Barbara”. Wenn Sie jetzt den Mauszeiger auf das Input-Field in Zeile 000051 bewegen, wird der geänderte Wert angegeben.

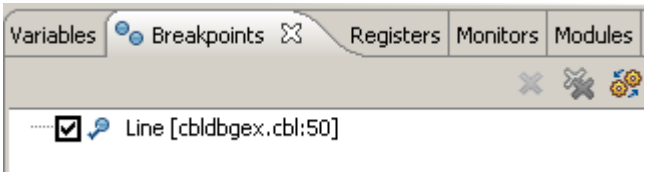


Abb. 5.3.7

In der rechts oberen Ecke, 1k auf den Breakpoints Tab um alle Breakpoints in dem Programm zu sehen.

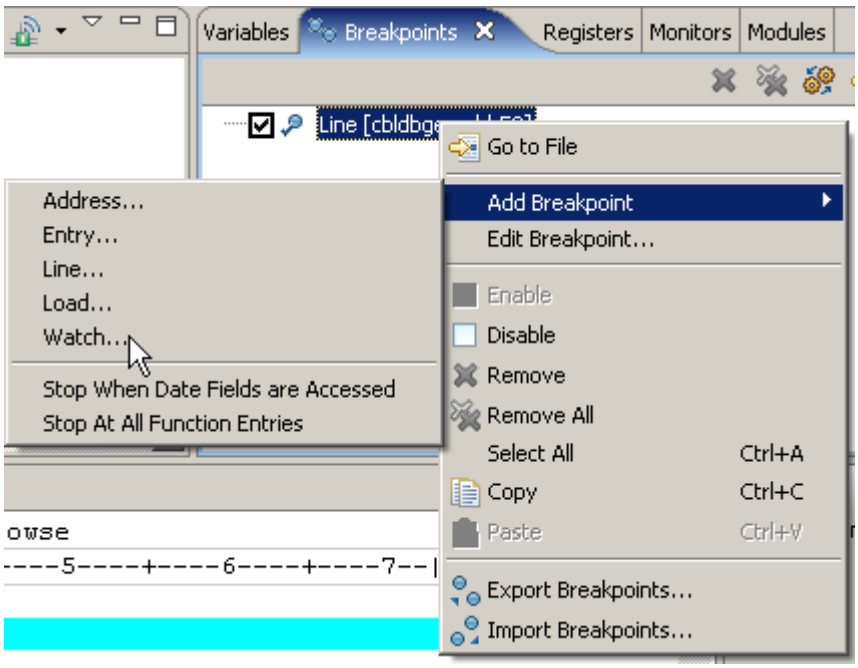


Abb. 5.3.8

Wir fügen nun einen Watchpoint hinzu.. 1kr auf den Breakpoint View. Selektiere Add Breakpoint → Watch…, 1k

5.4 Watchpoint

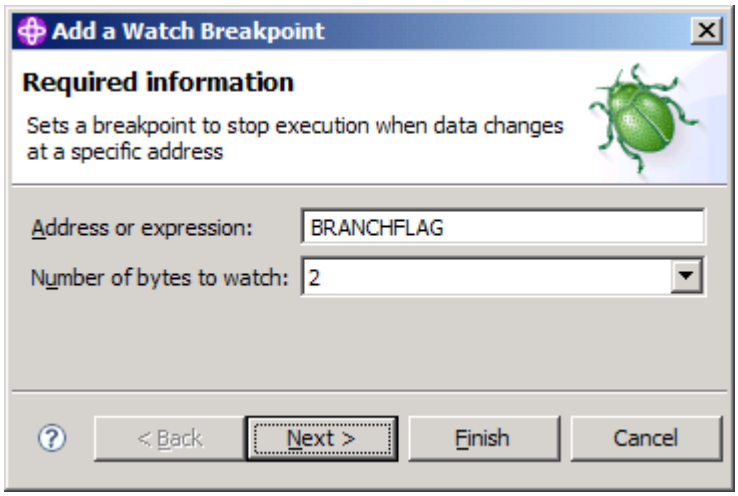


Abb. 5.4.1

BRANCHFLAG in das “Address or expression” Feld eingeben, sowie eine “2” in “Number of Bytes to watch”. Wir wünschen einen Stop an diesem Feld, wenn es geändert wird.
1k on Next.

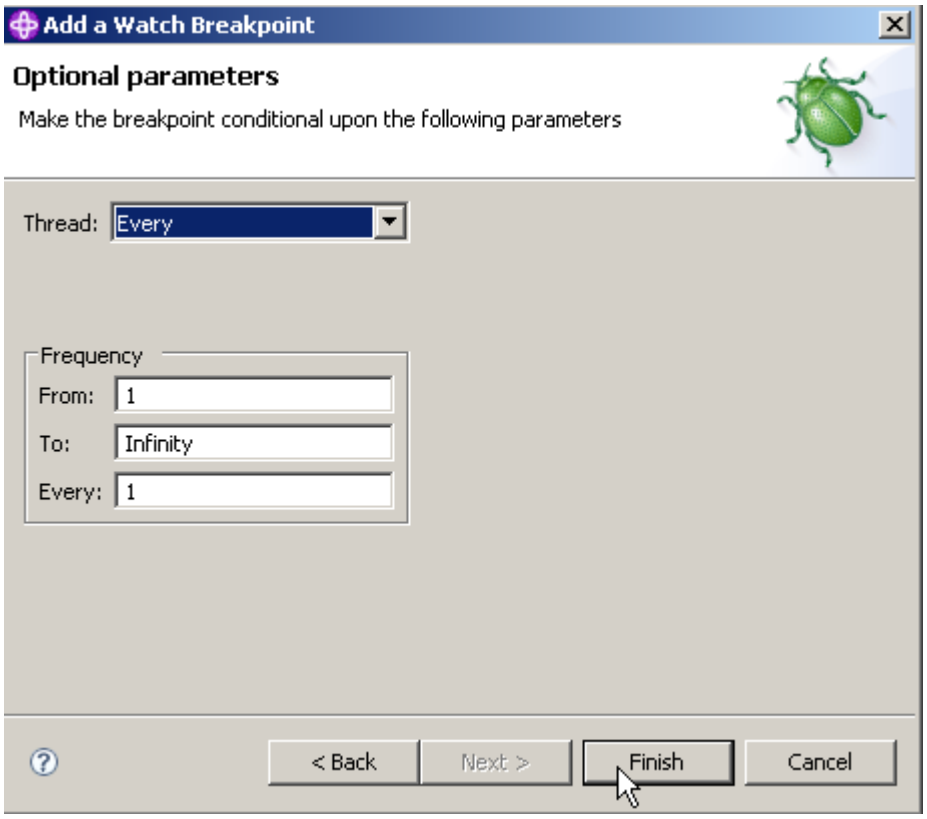


Abb. 5.4.2

Die Default Werte übernehmen, 1k auf Finish.

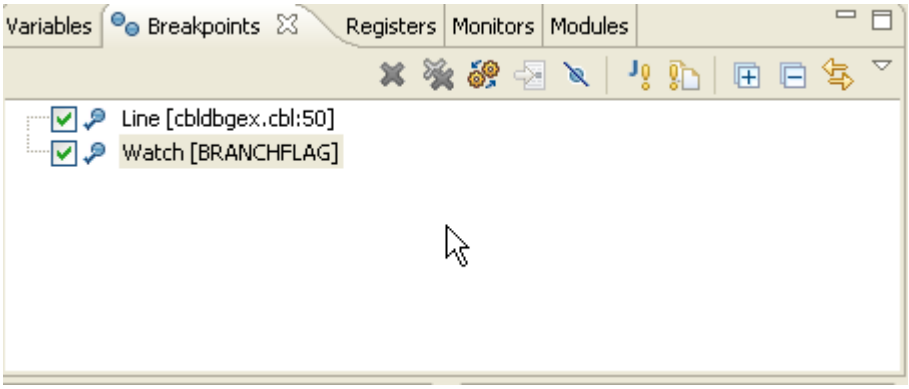


Abb. 5.4.3

Entfernen Sie den Breakpoint (an Stelle von entfernen könnten Sie ihn auch disablen). Öffnen Sie den Breakpoints View, 1kr innerhalb des Fensters, und …

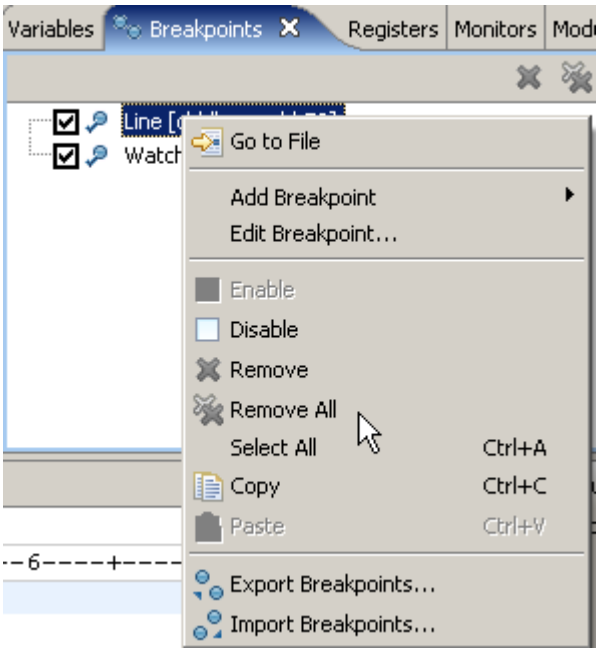


Abb. 5.4.4

selektiere “ Remove All” von dem Kontext Menu (1kr)

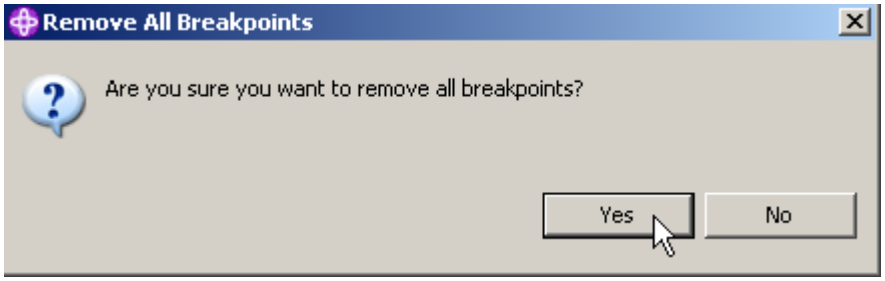


Abb. 5.4.5

Dies löscht alle Breakpoints.

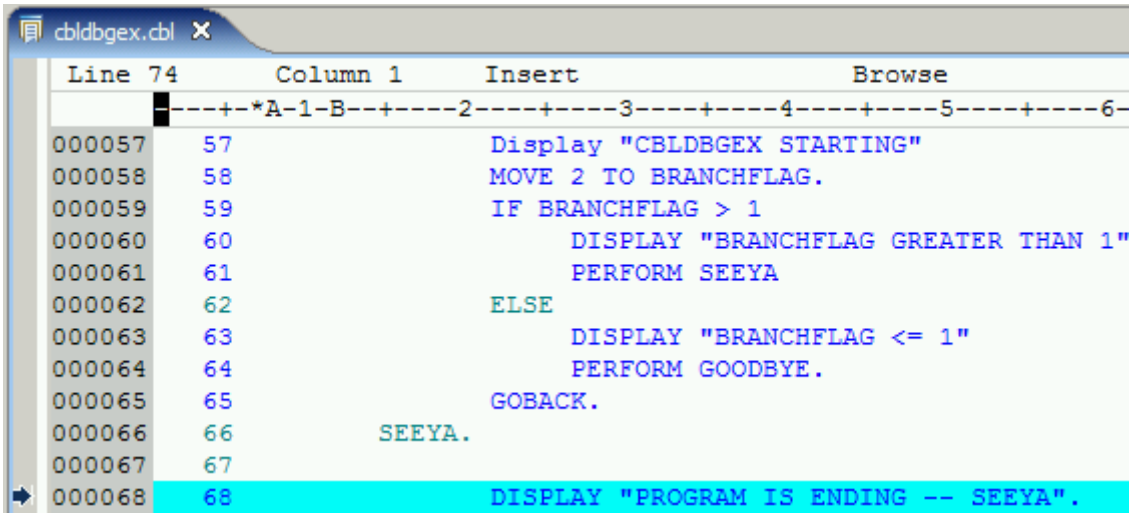


Abb. 5.4.6

In dem Debug Window, klicken Sie wiederholt Step Over (F6) bis Sie Zeile 68 erreichen:

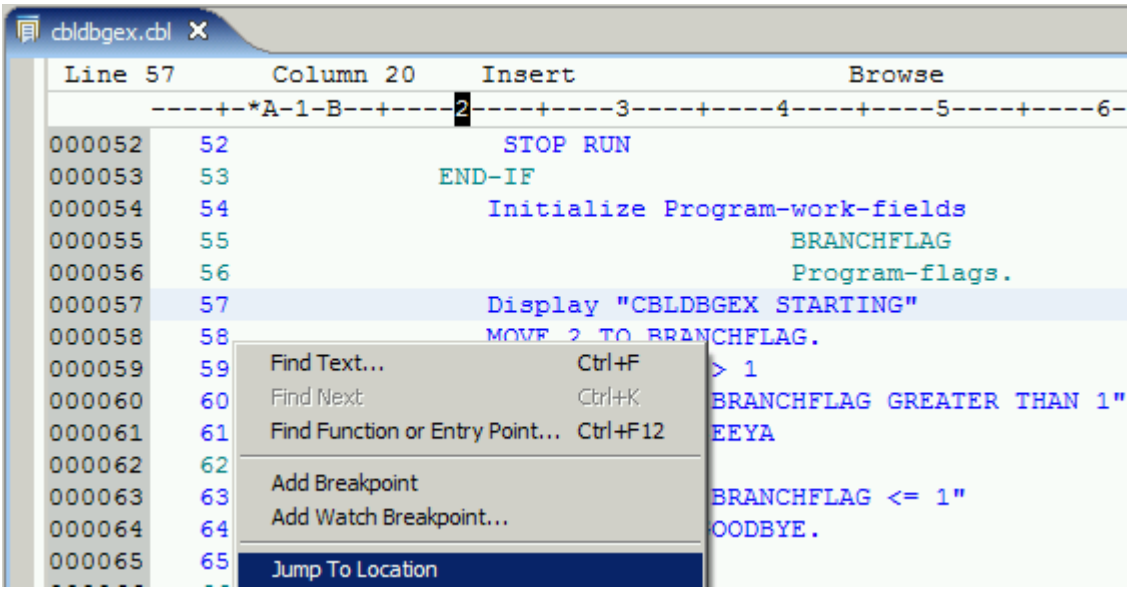


Abb. 5.4.7

Scroll zurück zu Zeile 58. Im Kontext Menu selektieren Sie “Jump To Location”. Dies bewirkt dass die Programm Ausführung zurück zu Zeile 58 bewegt wird.

Spielen Sie mit Debug, um vertrauter zu werden.

Wenn Sie das Programm weiter ausführen wollen, benutzen sie den Resume Icon ( ).

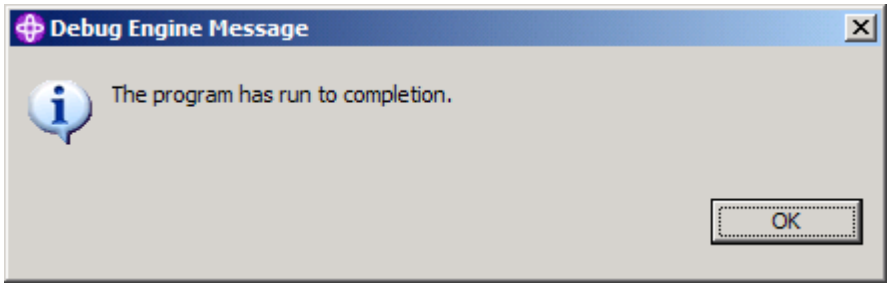


Abb. 5.4.8

Eine Nachricht zeigt das Programm ende an.

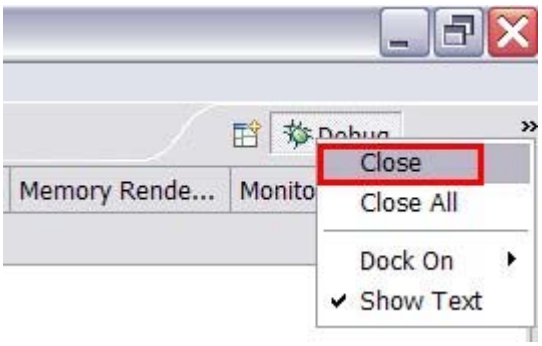


Abb. 5.4.9

Click OK um die Debug Perspective zu schließen (Upper Right).

Selbst-Test

- **Wodurch unterscheiden sich die RDz Test/Debug Funktionen von denen, die unter TSO und ISPF verfügbar sind ? Es gibt zahlreiche Personen, die lieber mit TSO als mit RDz arbeiten.**

Und das war es

Sie können die gleichen Debug Funktionen für CICS, IMS oder DB2 Programme benutzen, welche unter z/OS laufen. Hierzu ist es erforderlich, den Interactive Debugger auf diesen Platformen zu installieren.

Herzlichen Glückwunsch, Sie haben Tutorial 02 erfolgreich abgeschlossen. Bis hierhin mag das alles interessant sein, erläutert aber noch nicht das Schreiben von Cobol Programmen unter z/OS. Wir werden dies in dem nächsten Tutorial nachholen.

Ende

RDz Tutorial 03
Remote COBOL

Dieser Text wurde ohne den RDZ Server mit einer Konfiguration erstellt, bei der RDz unmittelbar auf der Workstation installiert war. Als Folge können bei der Benutzung des RDz Servers einige Referenzen auf die Workstation anders aussehen. Siehe z.B. Abb. 2.1.3, 2.2.3, 3.3.2, 4.3.10, usw.

Übersicht

In diesem Tutorial werden Sie unter Benutzung von RDz eine Cobol Anwendung auf einem System z Mainframe entwickeln. Sie definieren eine remote z/OS-Verbindung, setzen ein MVS-Projekt auf und führen ein remote Bearbeiten, Kompilieren und Debuggen durch.

Das Tutorial besteht aus den folgenden Teilaufgaben:

1. Vorbereitungen für die RDz Remote-Entwicklung

Bevor wir anfangen zu programmieren, müssen einige Vorbereitungen durchgeführt werden.

Zunächst werden Sie eine Verbindung zu einem Remote-System herstellen, welches für das Arbeiten mit RDz eingerichtet wurde. Sie werden lernen, wie Sie Ihre Daten mithilfe von Filtern organisieren, und wie Sie Data Sets mit RDz zuordnen. Sie werden ein JCL Script ausführen, um benötigte Data Sets zu erstellen.

Details zum Einloggen in den Remote RDz Server und zum Starten von RDz sind im RDz-Tutorial 01 Abschnitt 3 beschrieben.

1.1 Verbindung zu einem Remote-System definieren

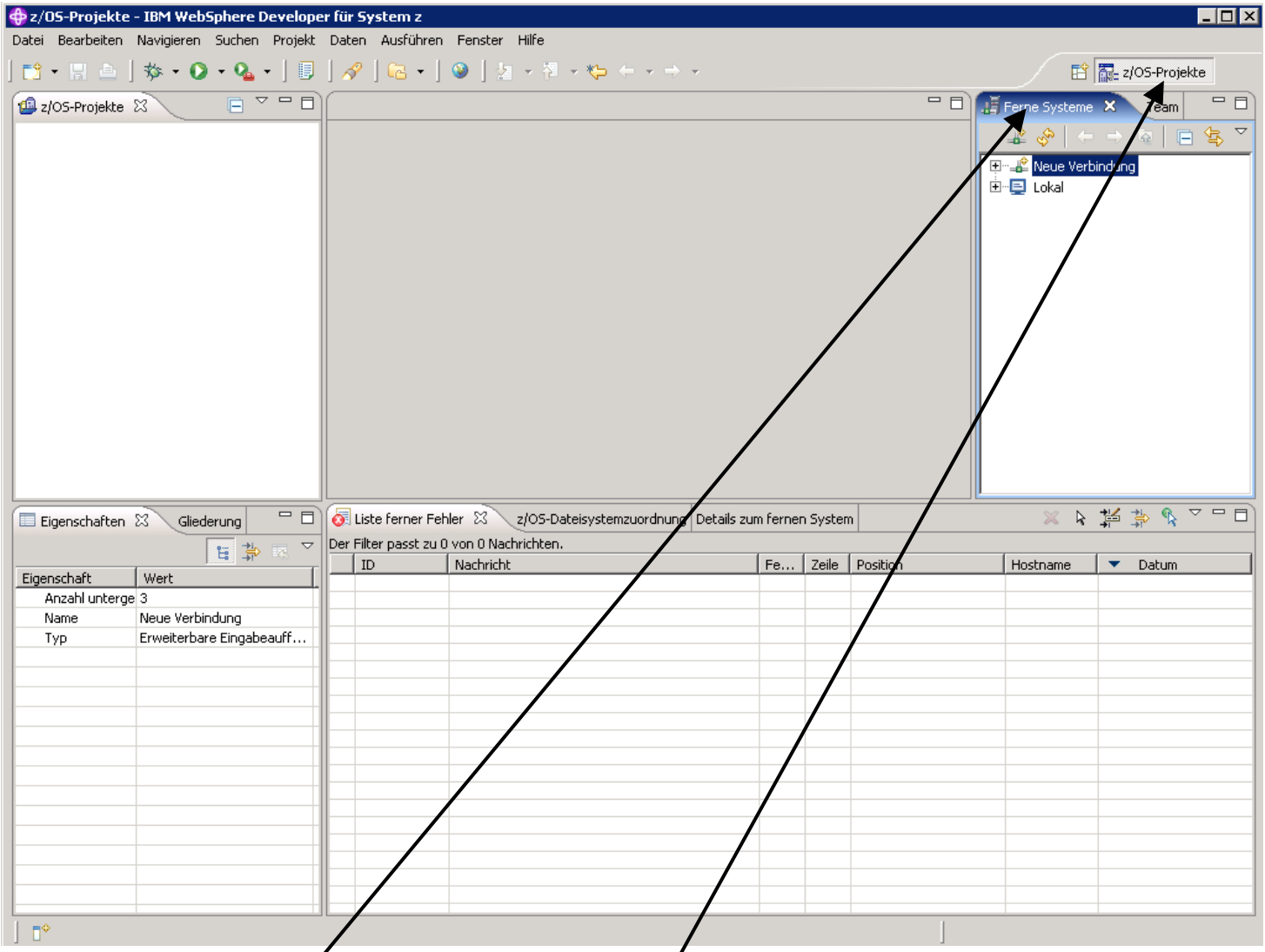


Abbildung 1.1

Stellen Sie sicher, dass die z/OS Projekte Perspective eingeschaltet ist.

Der Remote Systems View zeigt alle bestehenden Verbindungen zu entfernten-Systemen an.

Verbindungen sind persistente System Connection Objekte. Sie enthalten die erforderlichen Informationen, um auf einen bestimmten Remote-Host zuzugreifen. Der Remote Systems View ermöglicht Pop-up-Menu Aktionen für das Umbenennen, Kopieren, Löschen und neu Anordnen von bestehenden Verbindungen.

Verbindungen enthalten Attribute oder Daten, die zwischen Workbench Sitzungen gespeichert werden. Diese Attribute sind der Verbindungsname des Remote-Systems, eine optionale Beschreibung, und eine Benutzer-ID, die beim Verbindungsaufbau standardmäßig von jedem untergeordneten z/OS Subsystem als Default verwendet wird. Darunter sind alle Verbindungen als Dateien in einer Eclipse-Projekt namens RemoteSystemsConnections untergebracht. Der Benutzer kann sie für eine Team-Unterstützung aktivieren. Hiermit können Verbindungen von einem Team gemeinsam genutzt (shared) werden.

In diesem Beispiel benutzt der Mainframe Host entweder

- **den Namen frodo, und die IP Adresse 134.2.205.54, Port 423, oder**
- **den Namen leia und die IP Adresse 139.18.4.30, Port 23.**

Die hier verwendete Benutzer-ID ist prak032. Bitte ersetzen Sie alle Verweise mit dem von Ihnen benutzten Mainframe-Host und Ihrer Benutzer-ID.

Im Tutorial verwendete Parameter

	Wert	Kommentar
User-ID	prak<xxx>	Im Tutorial benutzte User ID
Passwort	????????	Erhalten sie vom Betreuer
IP Adresse	134.2.205.54, port 423 139.18.4.30, port 23	Telnet 3270-Emulator Telnet 3270-Emulator
Ports	139.18.4.30:4035 139.18.4.30:4035 139.18.4.30: 6715	MVS Port Entfernter Dämon JES Job-Monitor
Verknüpfungsbibliotheken	CEE.SCEELKED	Für den Verknüpfungsschritt erforderliche Bibliothek

Jeder Übungsteilnehmer erhält vom Betreuer eine User-ID mit einer ihn eindeutig zugeordneten Nummer. Im Tutorial ist prak<xxx> bzw. prak032 überall durch diese Nummer zu ersetzen.

Wenn Sie nicht unseren RDz Server benutzen, sondern eine Kopie von RDz 7.0 auf Ihrer Workstation (z.B. unter VMWare) installiert haben, stellen Sie sicher, dass auf Ihrer Workstation der Firewall sowohl auf Ihrem VMware Host und auf Ihrem Gastcomputer deaktiviert ist. Zumindest müssen alle Verbindungen von/zu 134.2.205.54 oder 139.18.4.30 offen sein.

1.2 Remote-Verbindung zu einen Mainframe herstellen

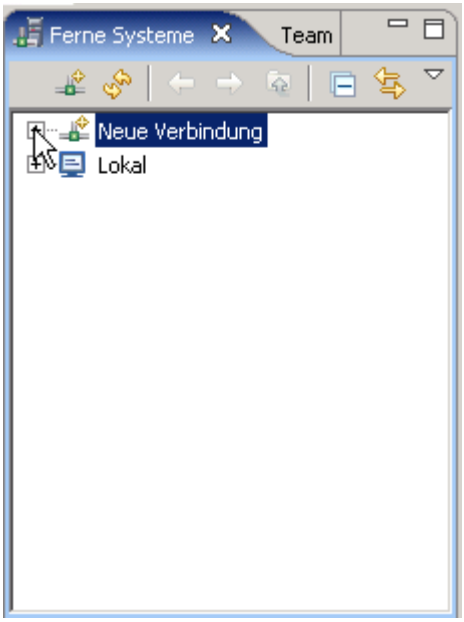


Abbildung 1.2

In der View "Ferne Systeme“ (remote Systems) "Neue Verbindung" erweitern.

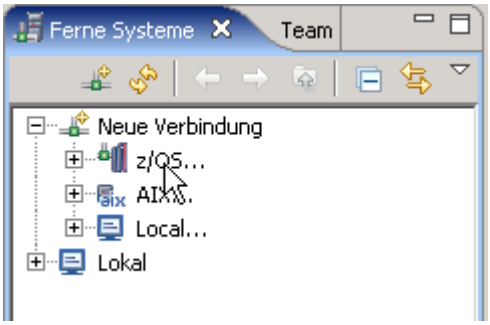


Abbildung 1.3

Rechtsklick auf "z/OS..." und

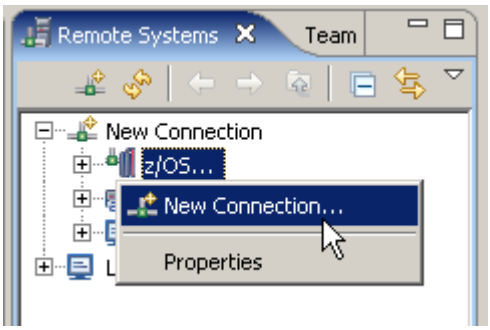


Abbildung 1.4

"Neue Verbindung" auswählen

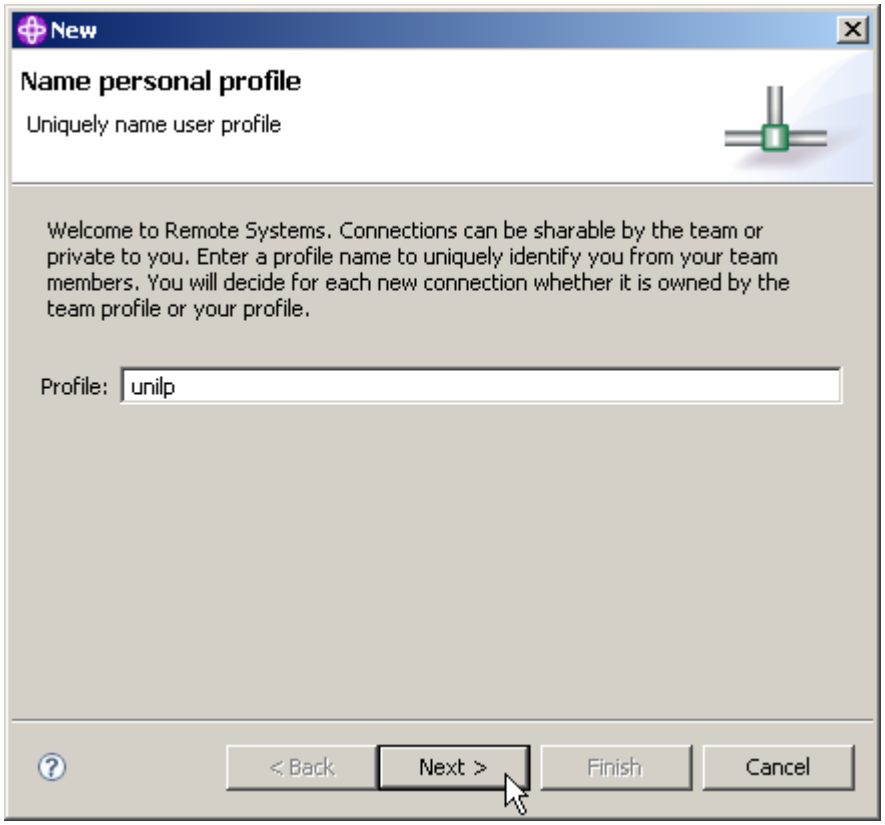


Abbildung 1.5

Wenn dies der erste Versuch ist eine Verbindung zu erstellen, werden Sie aufgefordert, ein Profil zu erstellen. Nennen Sie Ihr Profil „unilp“und klicken Sie auf Weiter.

Hinweis: Wenn der Bildschirm nicht angezeigt wird gibt es bereits ein vorhandenes Profil, und Sie können diesen Schritt ignorieren.

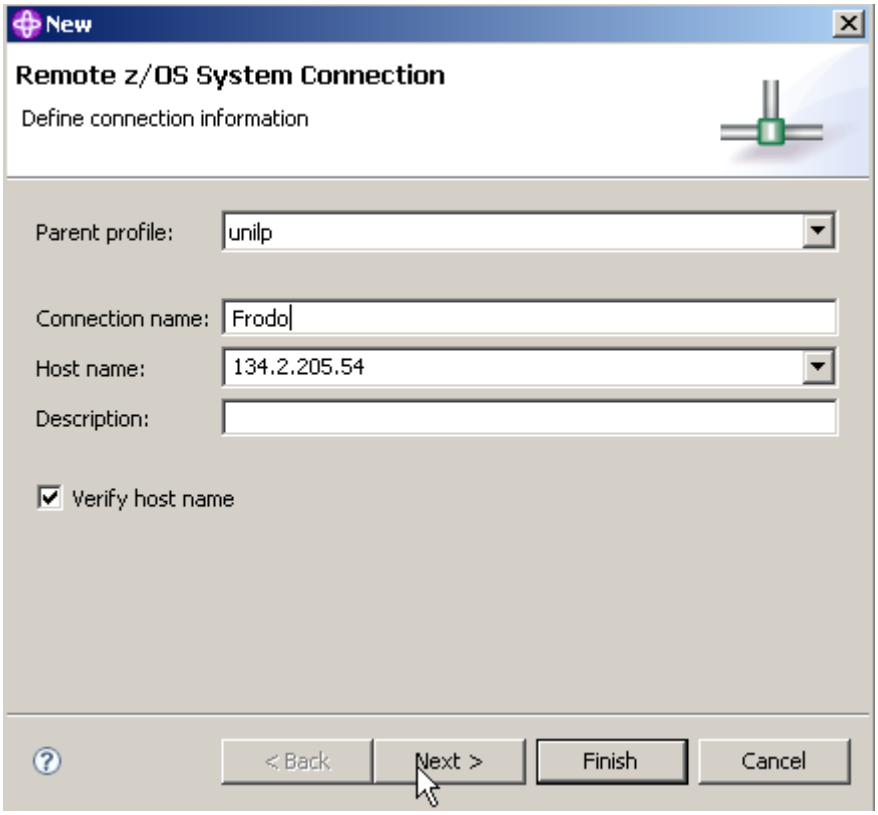


Abbildung 1.6

Die anzulegende Verbindung benötigt einen eindeutigen Namen, z.B. Frodo.

Als Hostnamen die IP-Adresse "134.2.205.54” eintragen.

Als "Beschreibung" kann ein beliebiger Text eingetragen werde, , z.B. "Verbindung zu frodo.informatik.uni-tuebingen.de“

Wird ein Häkchen vor "Hostnamen prüfen" gesetzt, wird die Korrektheit des Hostnamens geprüft.

Frodo braucht Port 423 statt Port 23

Alternativ können Sie auch mit dem Sytem leia.informatik.uni-leipzig.de oder 139.18.4.30 unter Port 23 arbeiten.

Klick auf "Weiter"

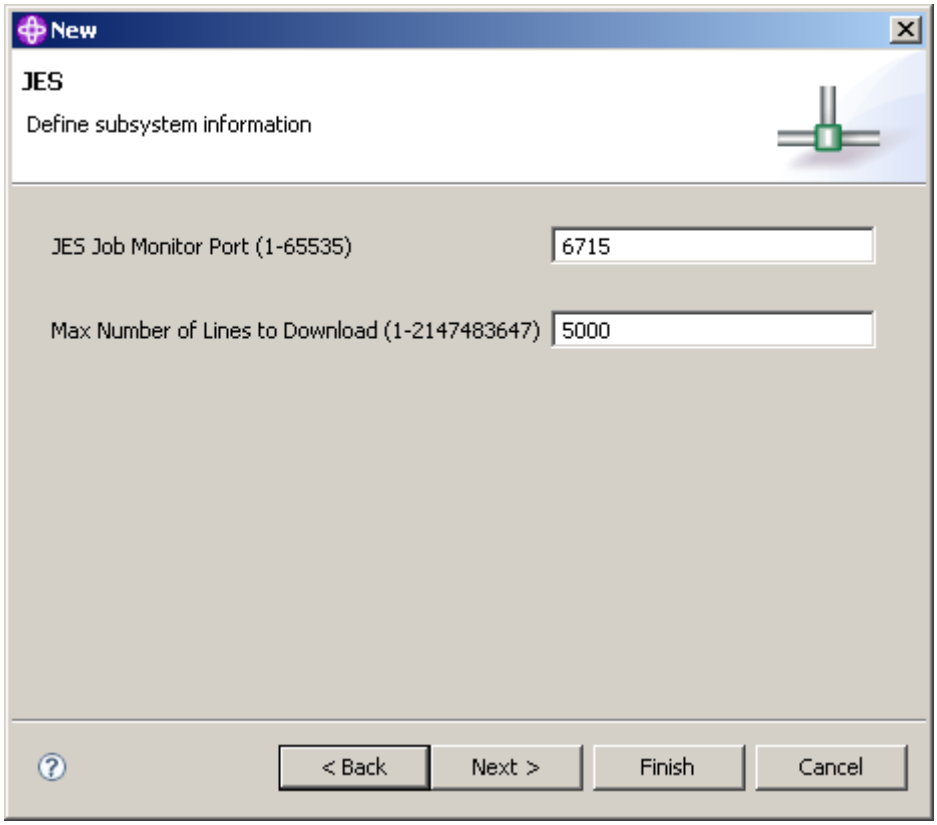


Abbildung 1.7

Im JES properties Panel, Port = 6715 eingeben. Klick auf "Weiter"

Wenn Sie einen anderen z/OS Server als frodo oder leia verwenden, werden diese Portnummern unterschiedlich sein.

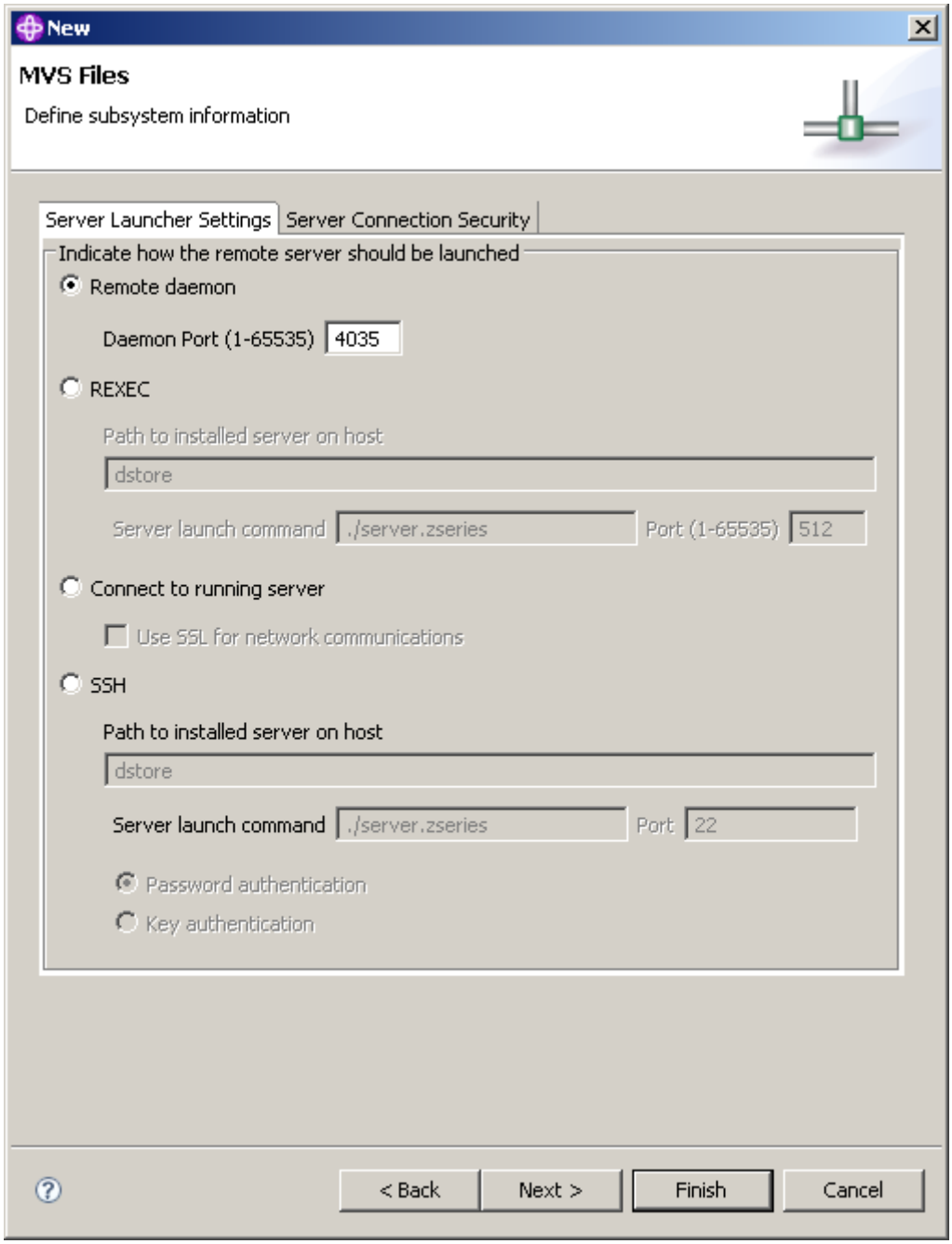


Abbildung 1.8

Im MVS Files Panel, den Defaults Port 4035 unverändert lassen. Klick auf "Weiter"

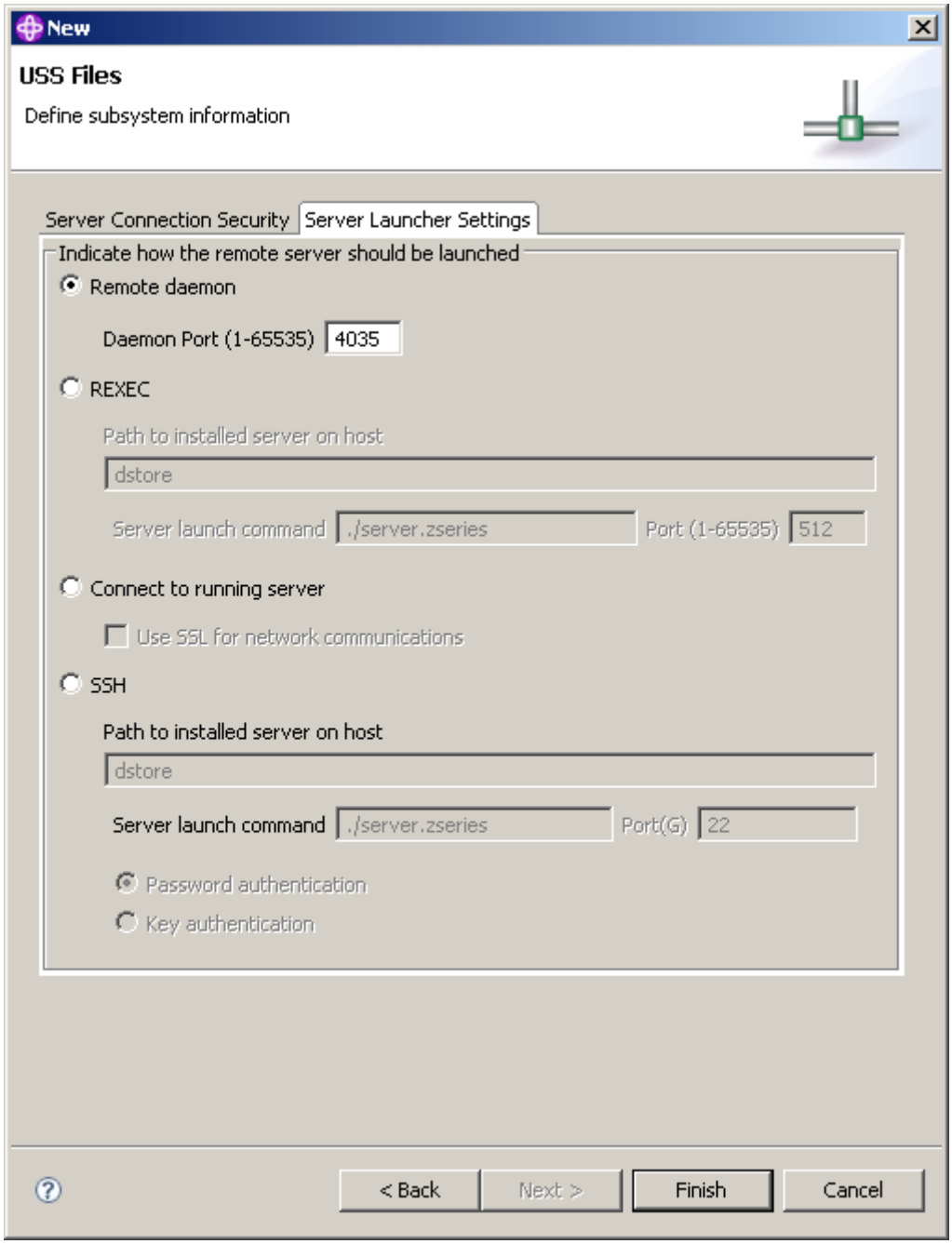


Abbildung 1.9

Im USS Files Panel, 1k auf Server Launch Settings. Verifizieren Sie Daemon Port 4035, lassen Sie die Default Werte unverändert, 1k auf Finish. Dies erstellt die neue z/OS-Verbindung und fügt sie in der Remote Systems Perspektive ein.

Selbst-Test

- **Warum müssen wir eine Verbindung erstellen, ehe wir uns einloggen ?**
- **Es muss zunächst eine Verbindung zu dem von Ihnen benutzten z/OS System aufgebaut werden. Diese benutzt standardmäßig Port 23 (Port423 im Falle von Frodo).? Wozu braucht man dann die extra Ports für JES, MVS Files und USS ?**

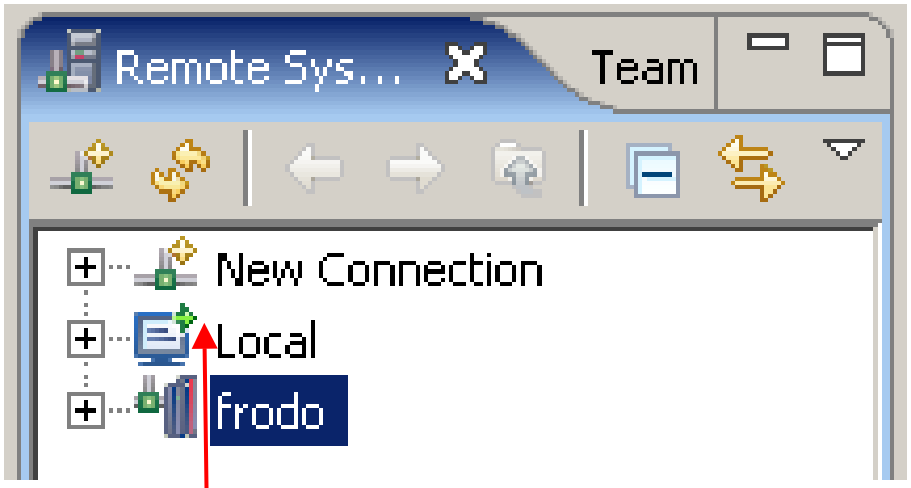


Abbildung 1.10

Nun ist in dem "Remote System"-View die Ressource "frodo“ hinzugekommen.

Beachten Sie: “Local” bezieht sich auf die RDz Funktionen, die unter Windows verfügbar sind. „frodo“ bezieht sich auf die RDz Funktionen, die unter dem z/OS System 134.2.205.54 potentiell verfügbar sind. Auf die „Local“ Funktionen kann unmittelbar zugegriffen werden. Dies wird durch einen kleinen grünen Pfeil gekennzeichnet. Bei „frodo“ fehlt der kleine grüne Pfeil, weil zunächst noch ein Logon zu frodo vorgenommen werden muss.

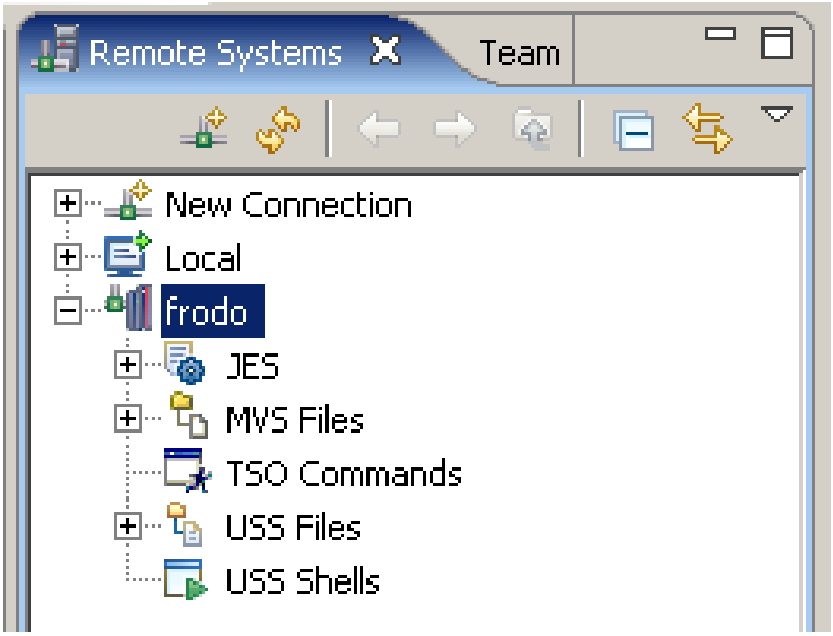


Abbildung 1.11

Expandieren Sie frodo. Es werden die die Ressourcen JES, MVS Dateien, TSO Commands, USS Dateien und USS Shells angezeigt. Diese angelegten Resourcen sind zwar verfügbar, aber noch sind die Verbindungen zu diesen Ressourcen inaktiv.

Diese Verbindung stellen wir als Nächstes her, indem wir uns über RDz auf Frodo mit unserer User ID und unserem Passwort einloggen.

21.3 Aktivierung einer Verbindungs-Ressource

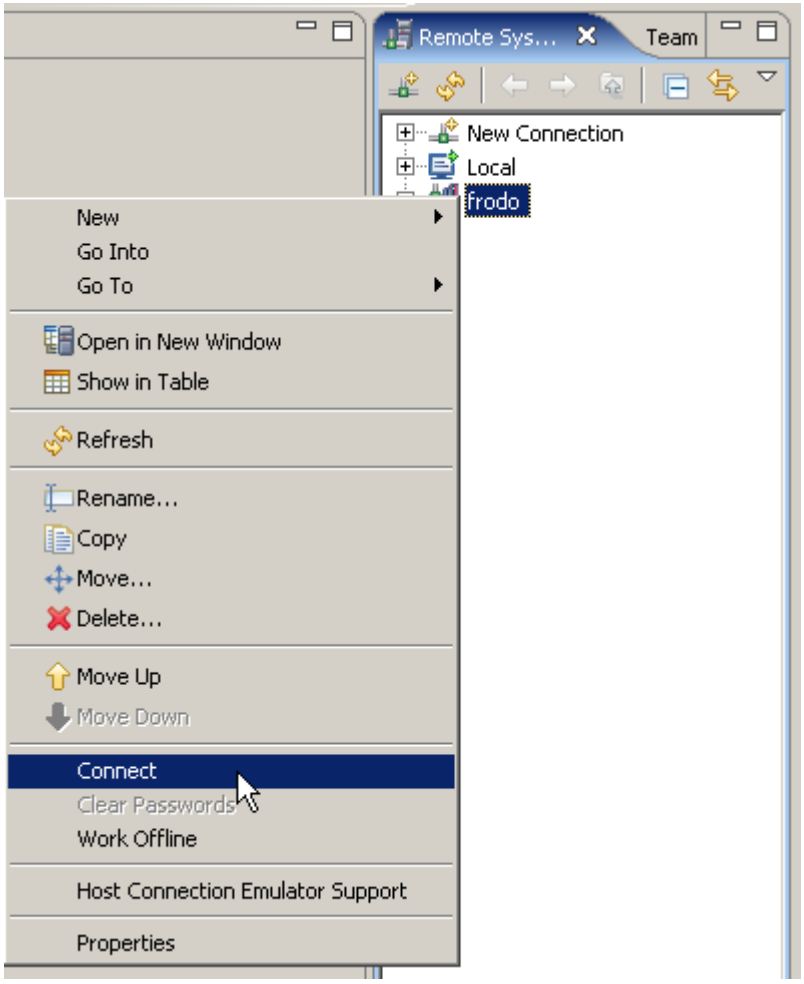


Abbildung 1.12

Rechte Maustaste auf "frodo“. Im geöffneten Popup-Fenster Klick auf "Verbindung herstellen".

Abbildung 1.13

Nun sind Login-Name und Passwort des Mainframes einzugeben, auf den man mittels der angelegten Verbindungsressource zugreifen möchte. Beide können auf Wunsch gespeichert werden.

Das Passwort darf kein Sonderzeichen enthalten.

Click OK um die Verbindung zu frodo herzustellen.

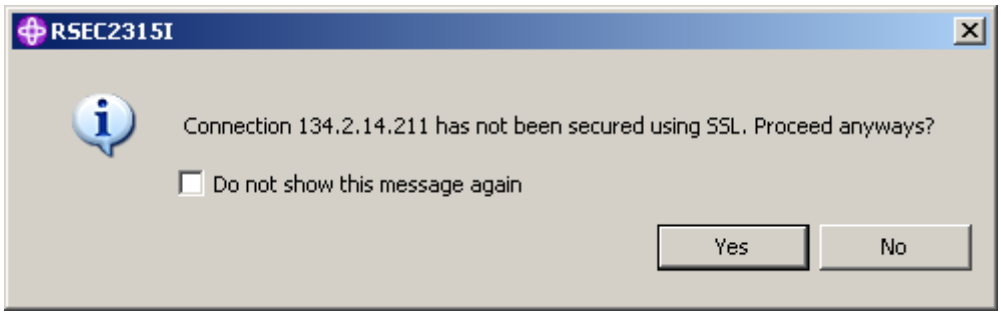


Abbildung 1.14

Natürlich.

Nach etwa 15 Sekunden sind alle Verbindungen aufgebaut.

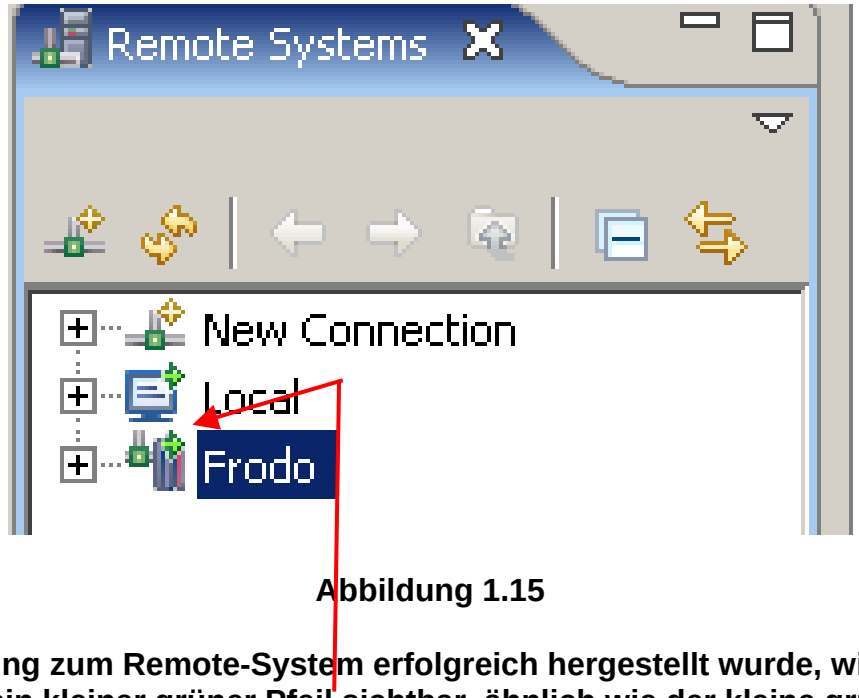


Abbildung 1.15

Wenn die Verbindung zum Remote-System erfolgreich hergestellt wurde, wird neben dem Symbol für Frodo ein kleiner grüner Pfeil sichtbar, ähnlich wie der kleine grüne Pfeil neben dem Symbol für Local.

Wichtig: Sie sind nun mit Ihrem Benutzer-ID auf dem z/OS Rechner verbunden. Wenn Sie die RDz Session beenden wollen, müssen Sie vorher ein z/OS Logout durchführen! Das Verfahren ist das Gleiche wie bei der Verbindung mit dem System. Der kleine grüne Pfeil wird dann wieder verschwinden.

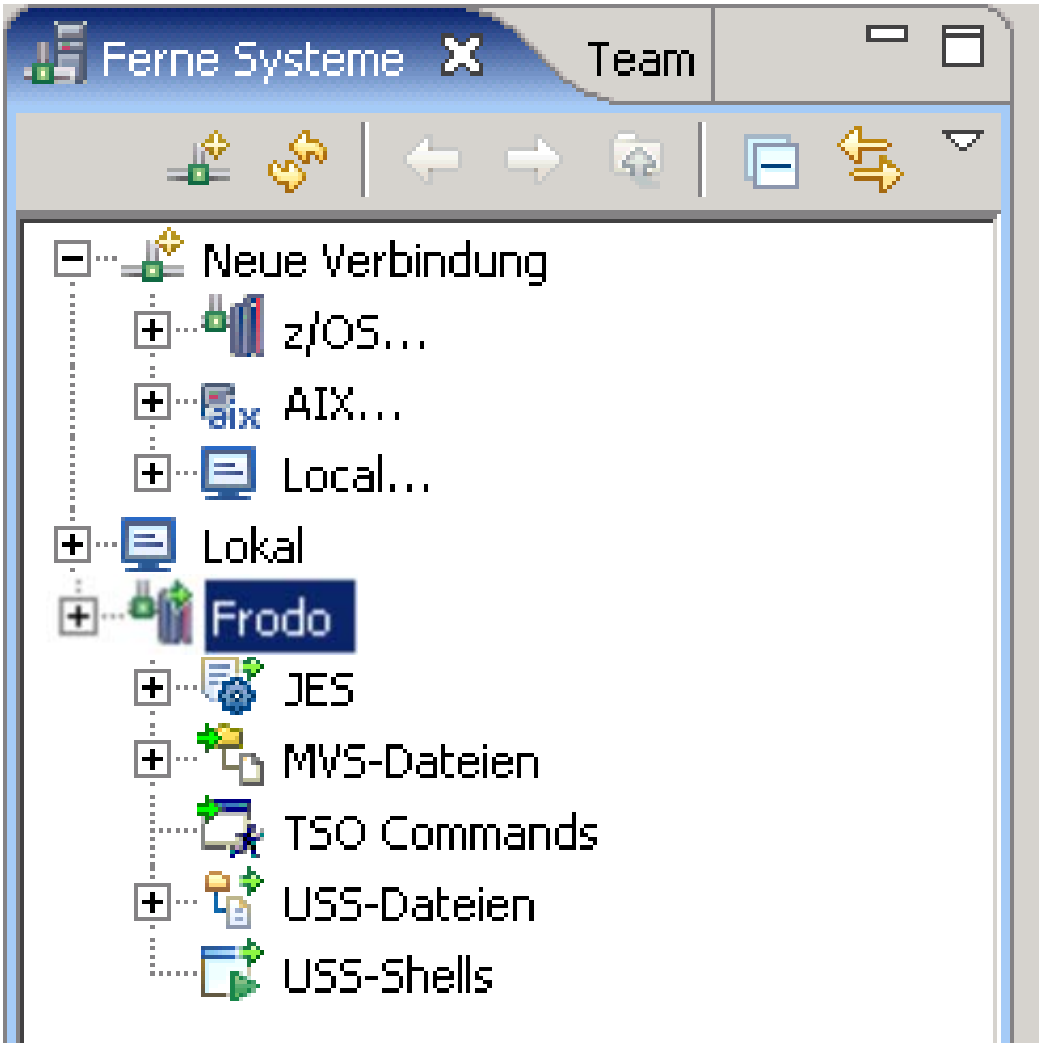


Abbildung 1.16

Wir hoffen, Sie haben gute Augen. Im Unterschied zu Abbildung 1.11 sind die kleinen grünen Pfeile hinzugekommen. Sie bedeuten, dass sowohl der Rechner frodo, als auch seine Resourcen (spezifisch JES, MVS Dateien, TSO Commands, USS Dateien und USS Shells) mit ihrem Workspace verbunden sind. Sowohl frodo, als auch alle Resourcen unter frodo, müssen den grünen Pfeil aufweisen, damit man damit arbeiten kann.

Wenn ein grüner Pfeil fehlt (z.B. bei JES) haben Sie ein Problem. Im Falle von JES kann die Ursache sein, dass Sie in Abb. 1.7 die falsche Port Nr. eingegeben haben. Wenn Sie das Problem nicht lösen können, wenden Sie sich an Ihren Betreuer.

Sie können unseren RDz Server benutzen, um mit unterschiedlichen Mainframe Rechnern zu arbeiten. Dies kann z.B. 139.18.4.30, eine zusätzliche unterschiedliche LPAR (virtuelle Maschine) von jedi.informatik.uni-leipzig.de sein, die wie Leia benannt haben.

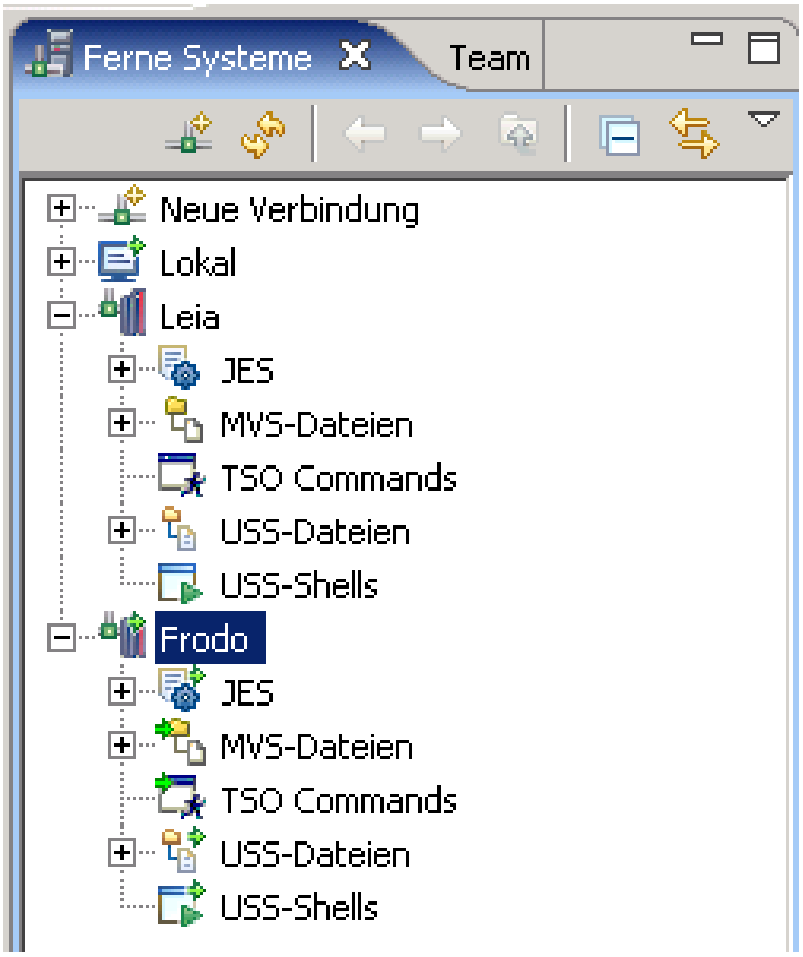


Abbildung 1.17

Sie sehen in Abb. 1.17 die Verfügbarkeit der z/OS Resourcen sowohl unter Frodo, als auch unter Leia. Sie können aber in jedem Augenblick mit Ihrer Benutzerkennung nur mit einem der beiden Rechner verbunden sein, in diesem Beispiel mit Frodo.

31.4 Auflistung der Datasets

Nun können wir uns mit einer PRAK User ID in Leia einloggen. Die kleinen grünen Pfeile sind nun unter den Leia Resourcen sichtbar.

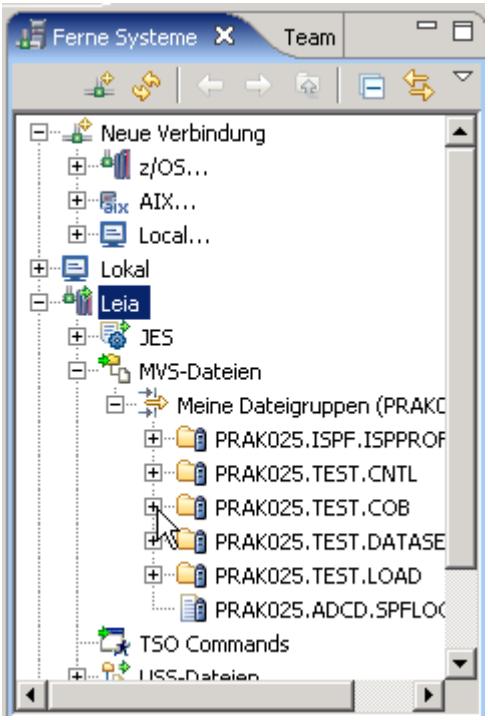


Abbildung 1.19

Die Erweiterung von "MVS-Dateien" und "Meine Dateigruppen..." macht – falls vorhanden – die dort vorhandenen Datasets sichtbar. Abb. 1.19 – 1.21 zeigt Data Sets, die ein Benutzer mit der Kennung PRAK025 auf Leia in der Vergangenheit angelegt hat.

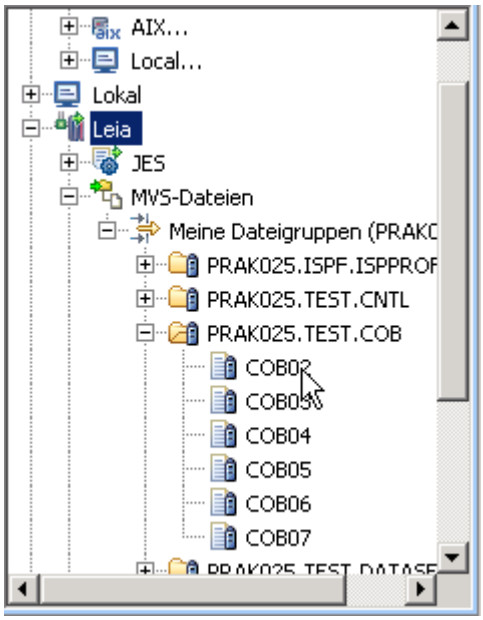


Abbildung 1.20

Der Partitioned Dataset PRAK025.TEST.COB lässt sich erweitern, es werden seine Member sichtbar.

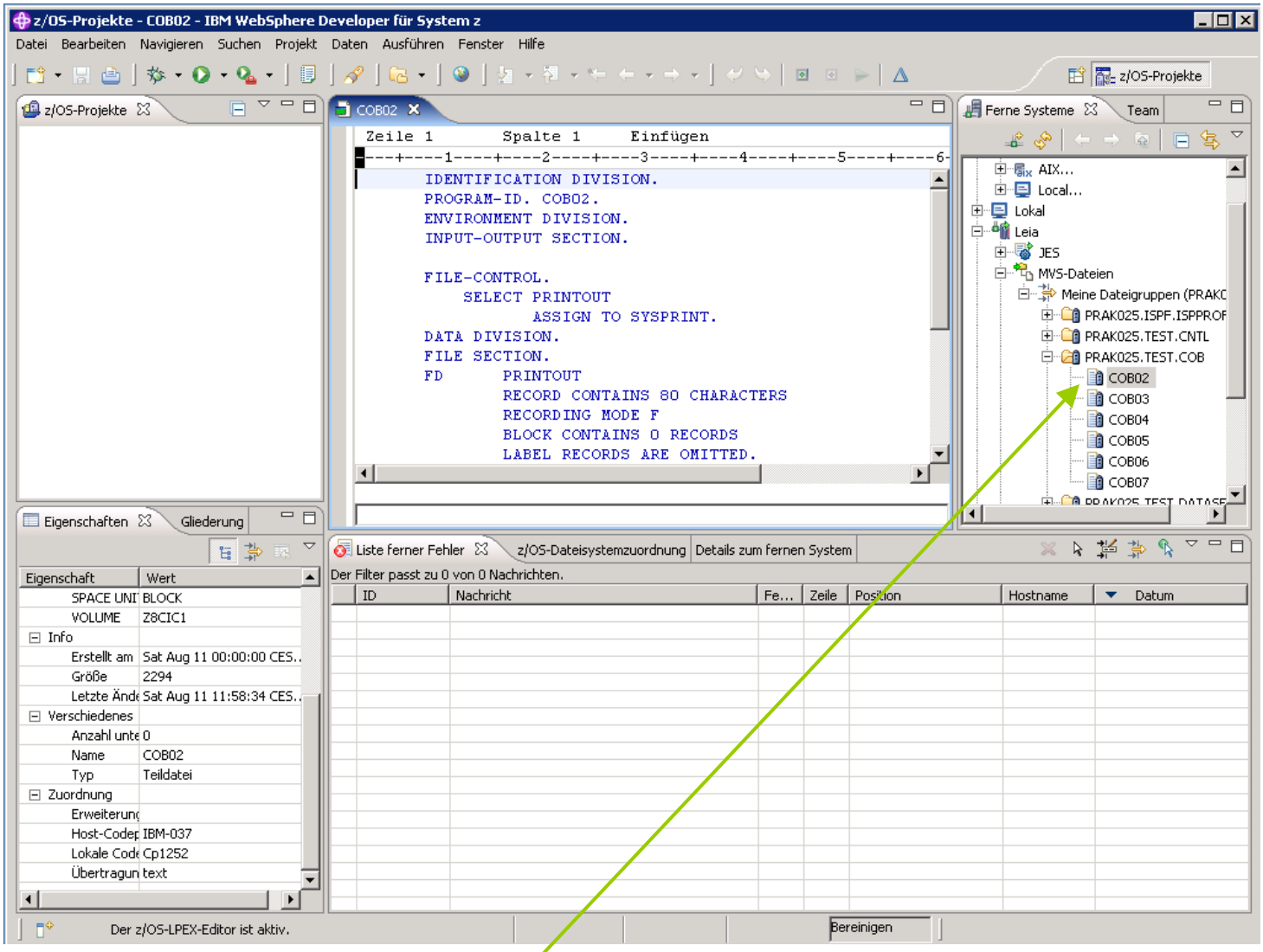


Abbildung 1.21

Ein Doppelklick auf einen Member öffnet diesen im LPEX-Editor im mittleren Fenster, falls der Member lesbare Information enthält.

41.5 Auflistung der JES-Jobs

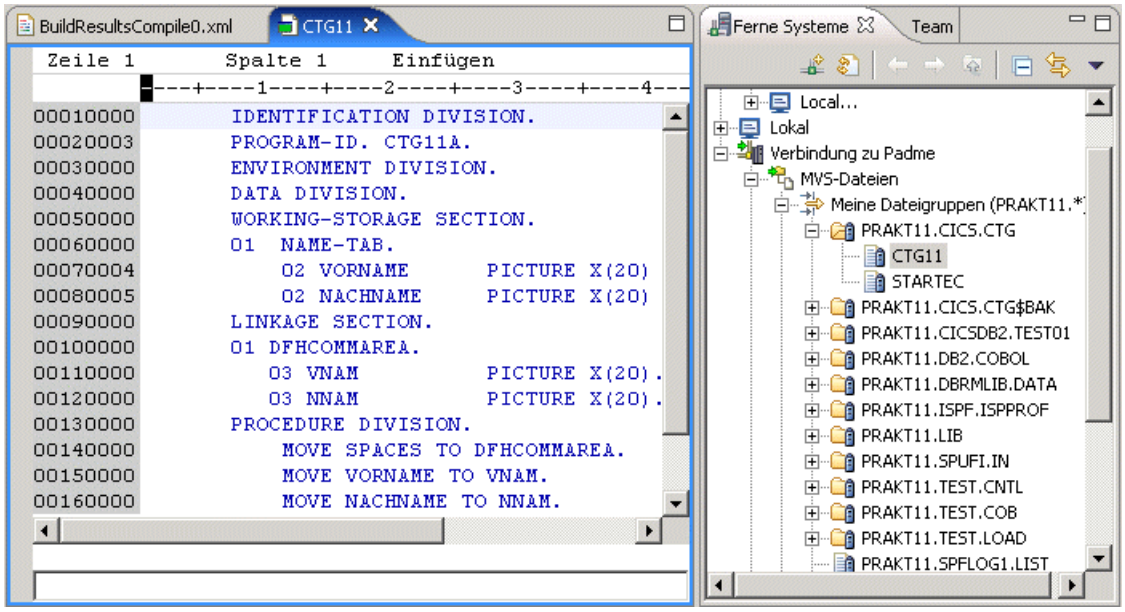


Abbildung 1.22

Abbildung 1.22 zeigt, dass im Mainframe-Account "PRAKT11" unter anderem der PDS "PRAKT11.CICS.CTG" mit seinen beiden Membern "CTG11" und "STARTEC" angelegt ist. In der Editor-View wird der Inhalt des Members "CTG11" (Cobol-Programm-Code) angezeigt.

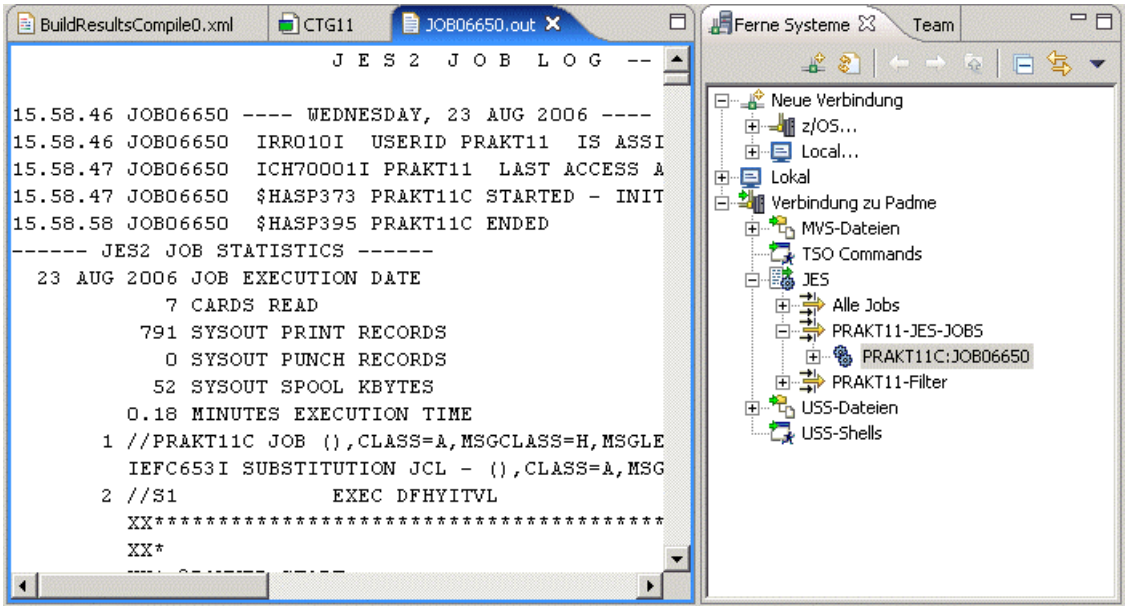


Abbildung 1.23

Die Expansion von "JES" und „PRAKT11-JES-JOBS" macht – falls vorhanden – die von JES erzeugten Jobs sichtbar. Das sind genau die Jobs, die auch mit einem 3270-Emulator im SDSF aufgerufen werden können. Ein Doppelklick auf einen Job öffnet im LPEX-Editor seinen Inhalt, falls dieser lesbaren Text enthält.

Das Beispiel in Abbildung 1.23 zeigt, dass es unter dem Login PRAKT11 nur einen einzigen JES-Job gibt, welcher den Jobnamen "PRAKT11C" und die Job-Nummer "06650" trägt. Der linke Teil der Abbildung enthält den lesbaren Output des Jobs.

Selbst-Test

- **Was ist der Unterschied zu den ISPF Dlist und Edit Funktionen ?**
- **Welche zusätzlichen Funktionen bietet RDz, die in ISPF nicht vorhanden sind ?**

51.6 Auflistung der Unix System Services Dateien

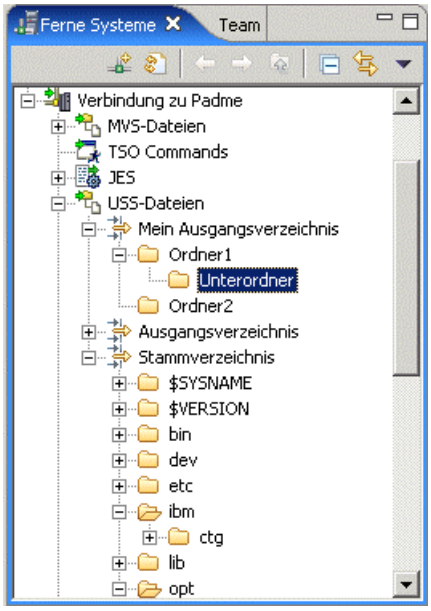


Abbildung 1.24

Wenn Sie "USS-Dateien" und "Mein Ausgangsverzeichnis" expandieren, werden – falls vorhanden – die eigenen USS-Ordner und -Dateien anzuzeigen.

Wenn Sie "Stammverzeichnis" expandieren, wird die Verzeichnisstruktur unter der Wurzel sichtbar gemacht.

61.7 Deaktivierung der Verbindung zum Mainframe

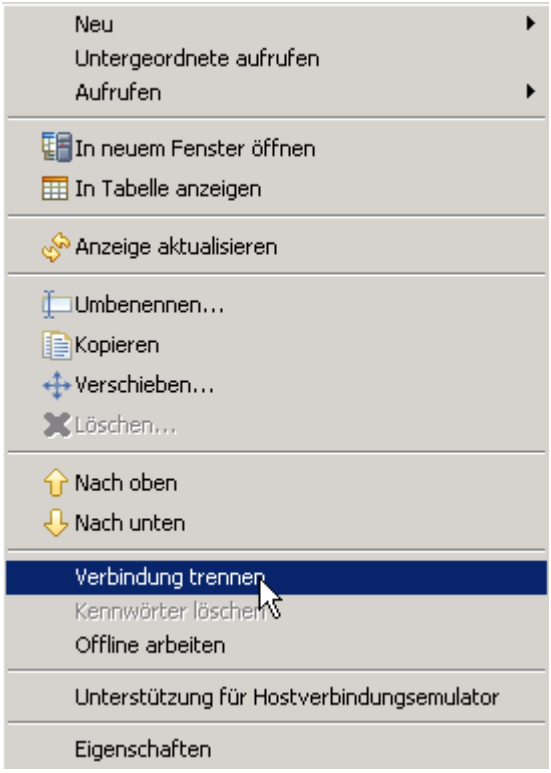


Abbildung 1.25

Rechtsklick auf den Namen der Verbindungsressource. Im sich öffnenden Fenster Klick auf "Verbindung trennen".

Sämtliche kleinen hellgrünen Pfeile verschwinden. Damit ist die Verbindung zum Mainframe geschlossen. An dieser Stelle können Sie jetzt - wenn sie wollen – die Sitzung mit RDz beenden.

Auf jeden Fall sollten Sie die Verbindung mit einem entfernten z/OS System trennen, ehe Sie eine RDz Sitzung beenden.

2. Parameter einstellen

2.1 z/OS System-Einstellungen

Wenn Sie zum ersten Malein neues z / OS-System installieren, spezifizieren Sie eine Reihe von Installations-Parametern. Beispiele hierfür sind die Lokatition der System-Libraries, die Konfigurationen Ihrer CICS oder DB2-Subsysteme, etc. Diese Daten werden als Systemeinstellungen (System Settings) bezeichnet.

Weiter benötigen wir die erforderlichen Einstellungen für z/OS Assets (z.B. z/OS Datasets) auf dem entfernten (remote) Host. Beispiele sind die richtigen COBOL Bibliotheken, CICS und DB2-Einstellungen usw. Normalerweise ist dies nur einmal für jeden neuen Workspace erforderlich.

RDz ordnet diesem Satz von System-Eigenschaften eine Reihe von Default Werten zu. Wenn Sie COBOL-Anwendungen in der Workbench entwickeln möchten, können Sie die Eigenschaften auf der Compileroptionen Registerkarte dieser Seite setzen. Sie überschreiben die JCL-Prozeduren, die sonst für compile, link, und die Ausführung verwendet werden würden.

Der String <HLQ> wird automatisch mit dem High Level Qualifier (Ihrer User ID) des Data Set Namens ersetzt. Später, je nach Verwendung von CICS, DB2 und IMS, müssen Sie möglicherweise zusätzliche Eigenschaften auf den übrigen Registerkarten (Tabs) einstellen.

Da diese Aufgabe nur einmal pro Installation getan werden muss, beschließen wir, die bereits definierten Eigenschaften für unsere Umgebung zu importieren.

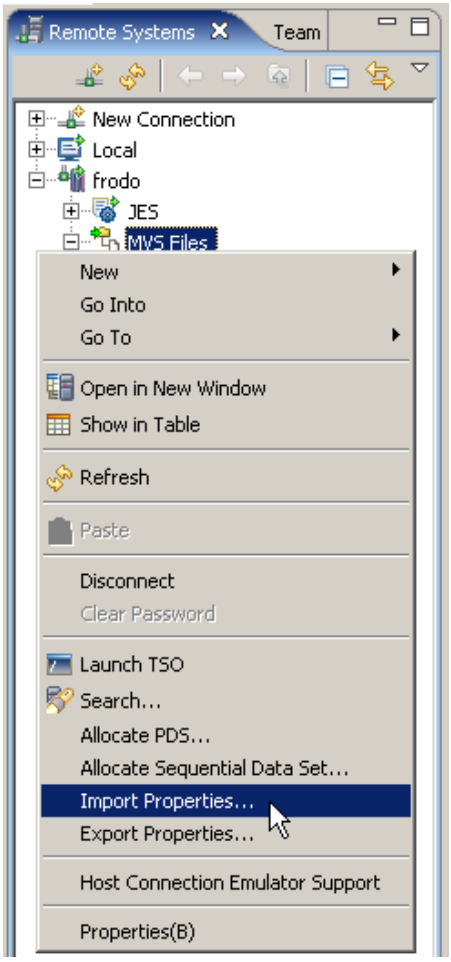


Abbildung 2.1.1

Im Remote Systems View, expandiere Frodo. 1kr auf MVS File, selektiere Import Properties.

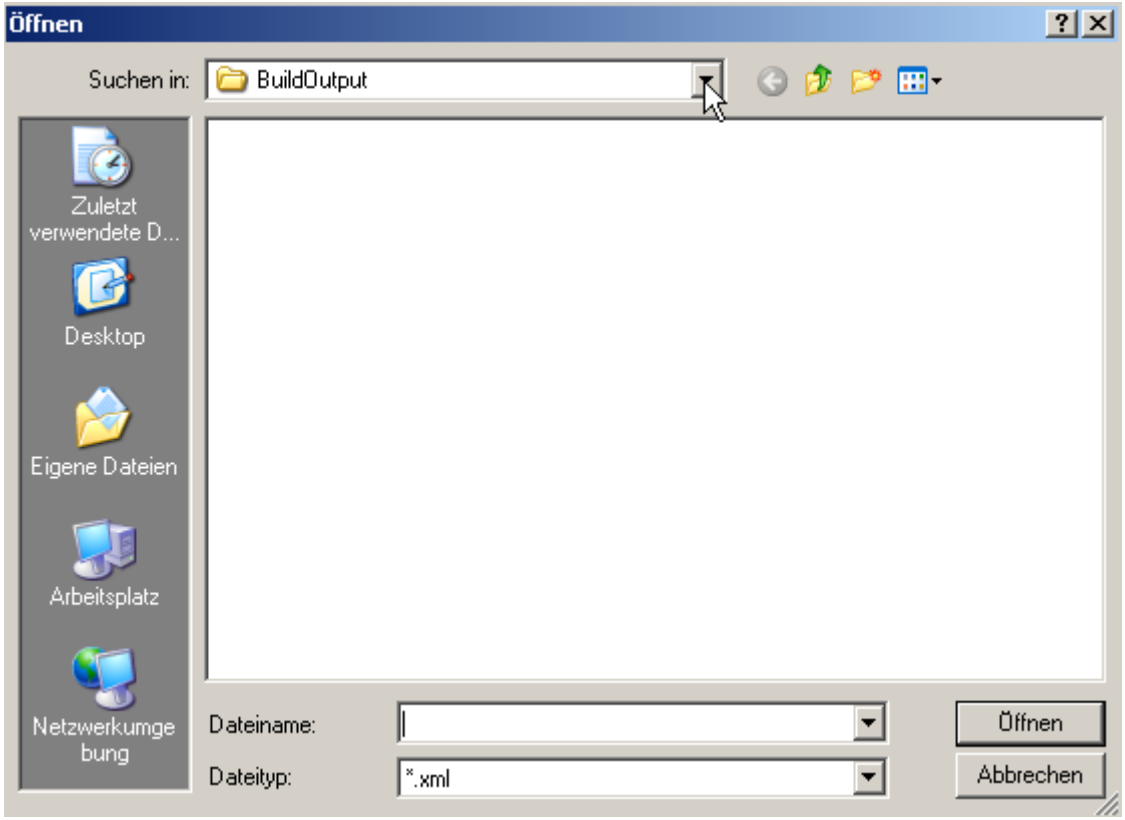


Abbildung 2.1.2

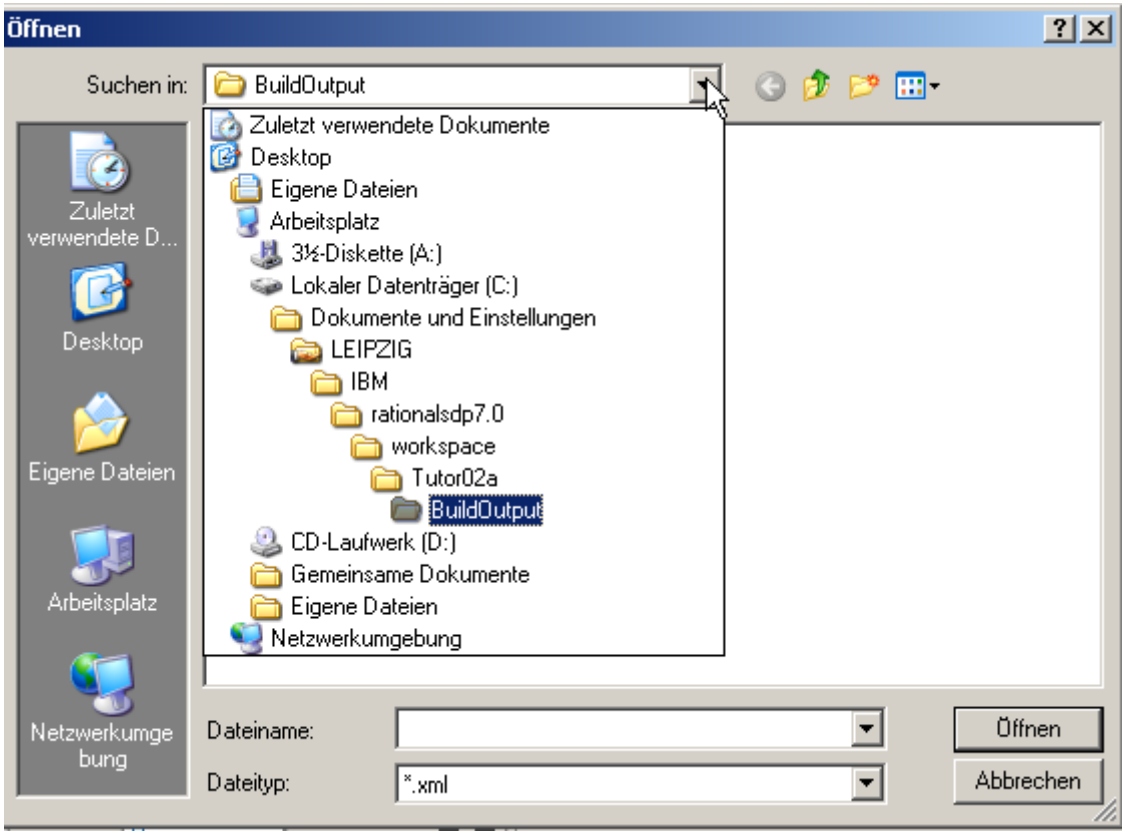


Abbildung 2.1.3

Auf dem RDz Server. Navigiere nach C:\Tutorials\Tutorial03\Resources\MVS_Files_Properties.xml , und dann …

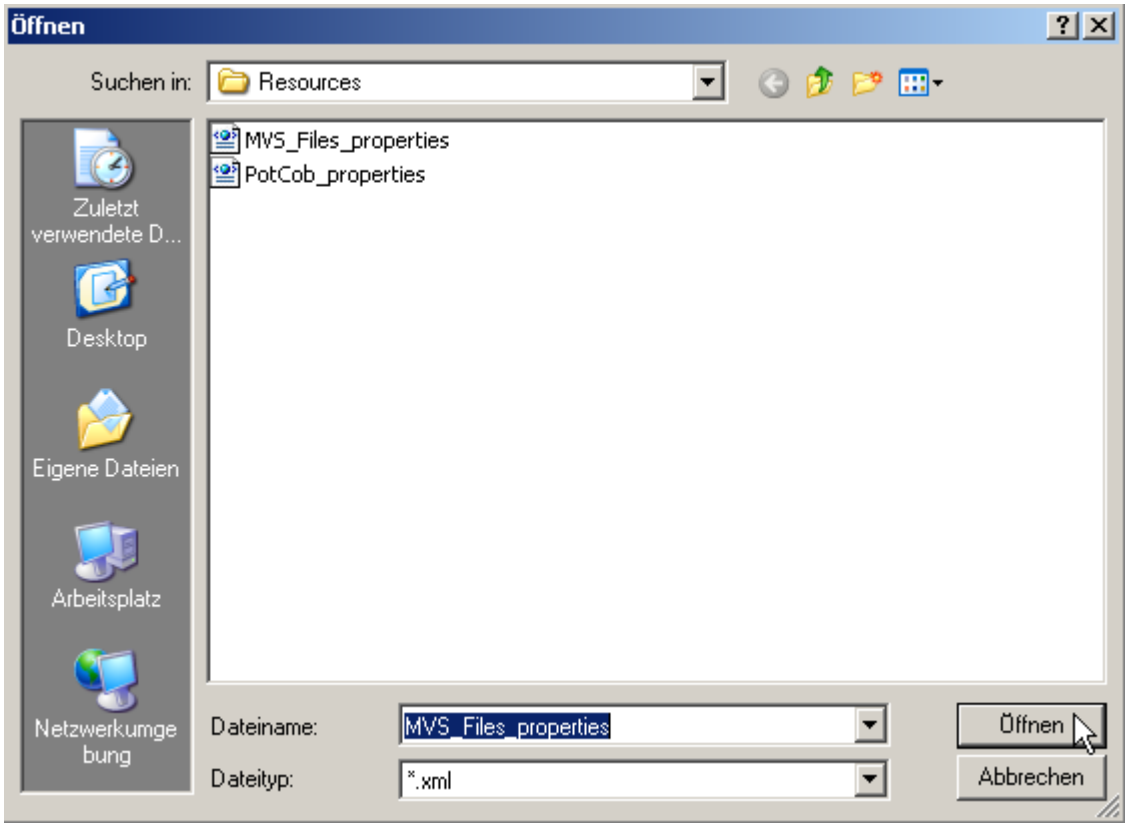


Abbildung 2.1.4

2k um es zu selektieren.
(zum selektieren können sie es auch öffnen. Dazu 1k auf die „Öffnen“ Schaltfläche).

Öffnen.

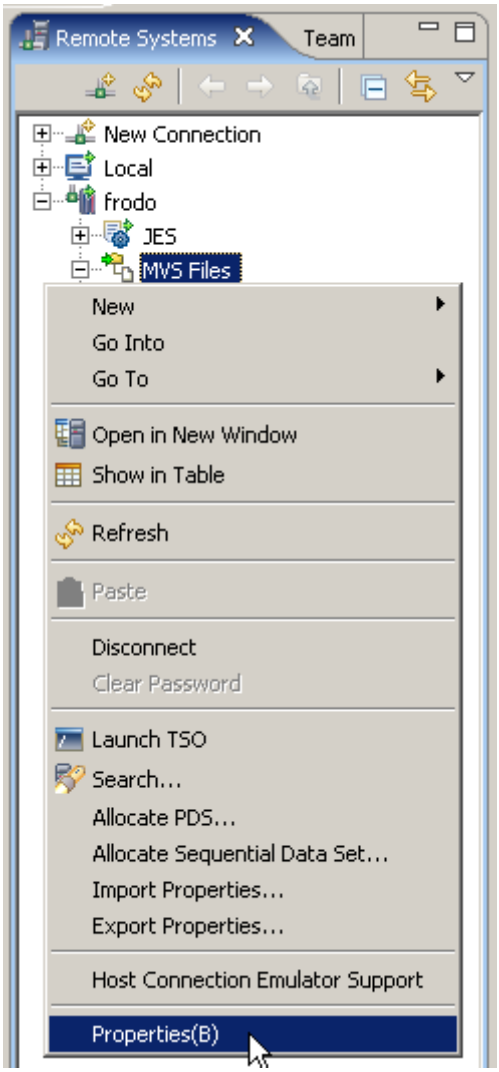


Abbildung 2.1.5

Nochmals 1kr auf MVS Files. Diesmal Properties selektieren.

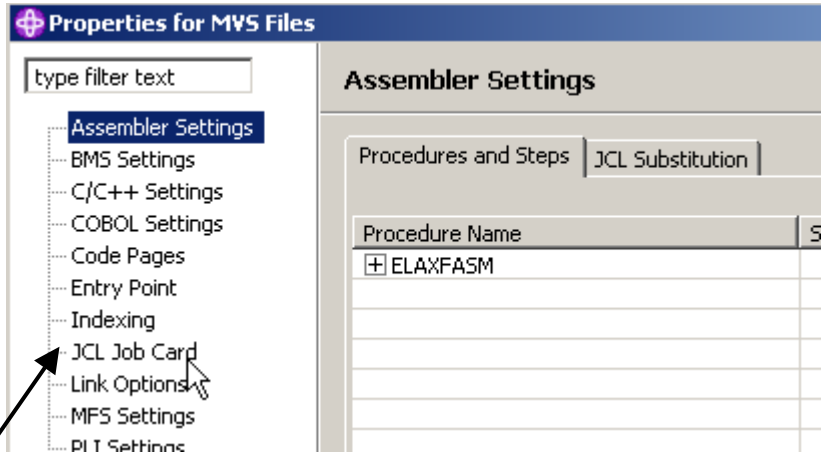


Abbildung 2.1.6

Die JCL Job Card muss geändert werden.

In dem “Properties for MVS Files” Fenster, 1k auf JCL Job Card

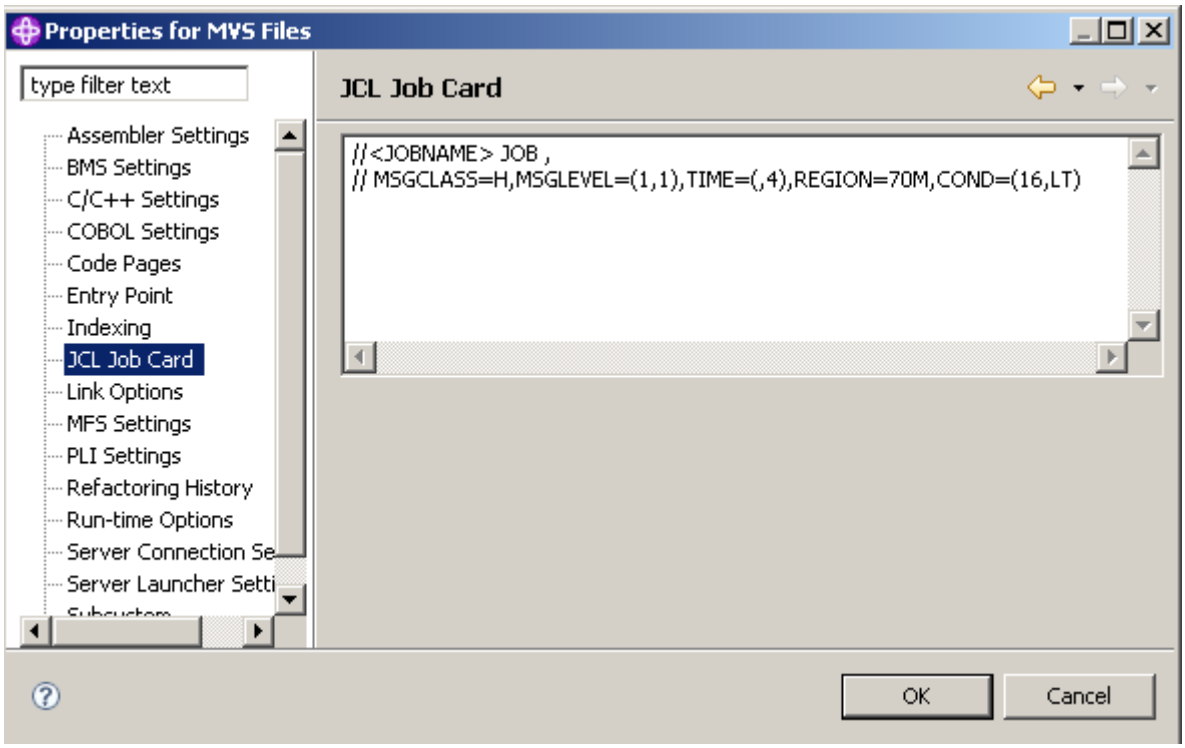


Abbildung 2.1.7

<JOBNAME> muss durch Ihre User ID ersetzt werden, z.B. prak032 in diesem Beispiel.

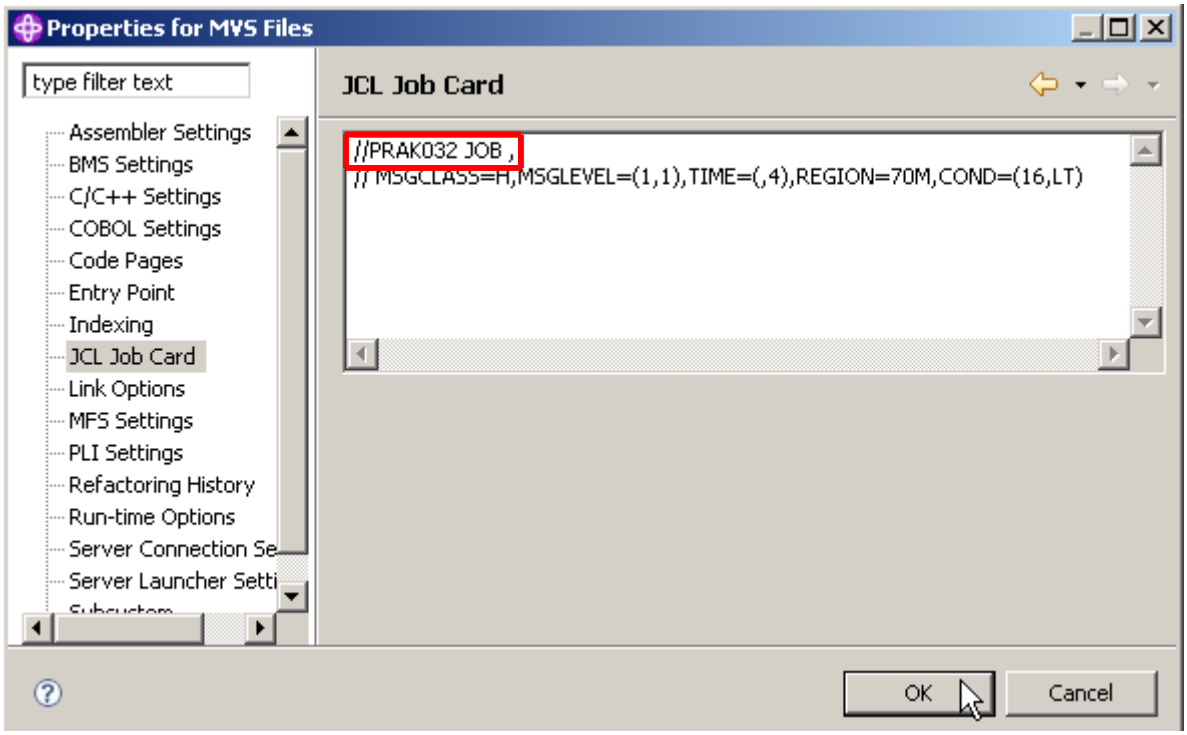


Abbildung 2.1.8

Der Rest ist ok.

Click OK to save. Das „Properties for MVS“ Fenster schließen.

Selbst-Test

- **Rekapitulieren Sie nochmals: Warum sind die Schritte in Abschnitt 2.1. erforderlich ?**
- **Lesen Sie ggf. die einführenden Sätze zu Beginn von Abschnitt 2.1 nochmals durch.**

2.2 Benötigte z/OS Datasets erstellen

Bisher haben wir mit einem Remote-System verbunden. Wir haben die z/OS System-Einstellungen erfolgreich eingerichtet.

Jetzt erstellen wir (allocate) insgesamt 9 Datensätze, die für die Entwicklung von Cobol Programmen benötigt werden. Wir könnten dies manuell tun, so wie bisher. Stattdessen verwenden wir ein JCL Script, dass alle 9 Datensätze erstellt.

Wir modifizieren einen bereits existierenden JCL Script Prototypen und submit diesen zu z/OS um die 9 Datasets zu erstellen. Sie finden den Prototypen dieses JCL-Scriptes auf Ihrer virtuellen Machine unter C: .

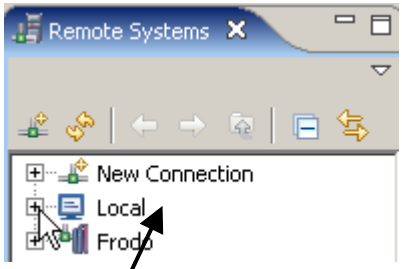


Abbildung 2.2.1

Denken Sie daran, RDz besteht aus einer Komponente auf Ihrer Workstation (Local) und einer Komponente auf Ihren Mainframe (Frodo in diesem Beispiel).

Die Dateien im Verzeichnis C: \ der virtuellen Maschine auf Ihrer Workstation sind in der in der Remote Systems Sicht unter Local angezeigt.

Unter Benutzung der Remote Systems Sicht, 1k auf das + Zeichen vor „Local“ um es zu erweitern (nicht das "Local ..." das Sie finden, wenn Sie New Connection expandieren !!).

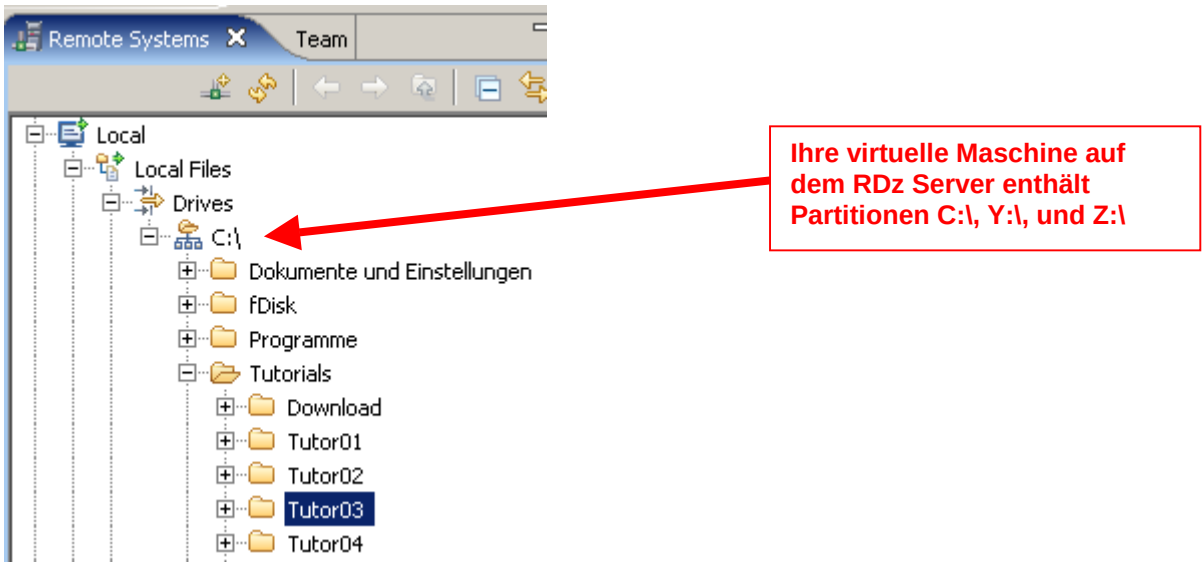


Abbildung 2.2.2

Navigieren sie nach C:\WDz_Tutorials\Tutor03 auf Ihrer Virtuellen Maschine.

Navigieren sie zum Directory “Local Files\Drives\C\Tutorials\Tutorial03”.

Wenn das Directory nicht da ist, besorgen sie sich den Inhalt von Ihrem Betreuer und kopieren sie es nach C:\ von Ihrer virtuellen Maschine.

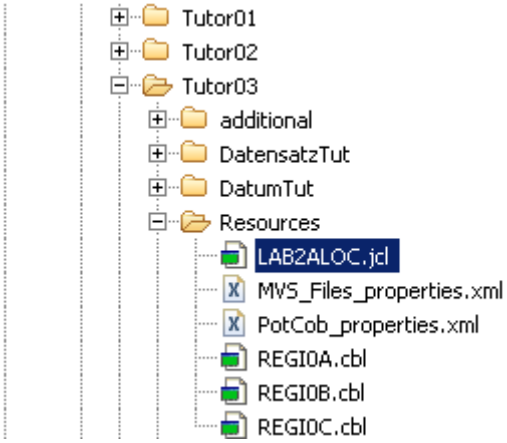


Abbildung 2.2.3

Jetzt Tutorl03, und die File LAB2ALOC.jcl in dem Directory Resources suchen.

2k auf LAB2ALOC.jcl um die file im editor Fenster zu öffnen.

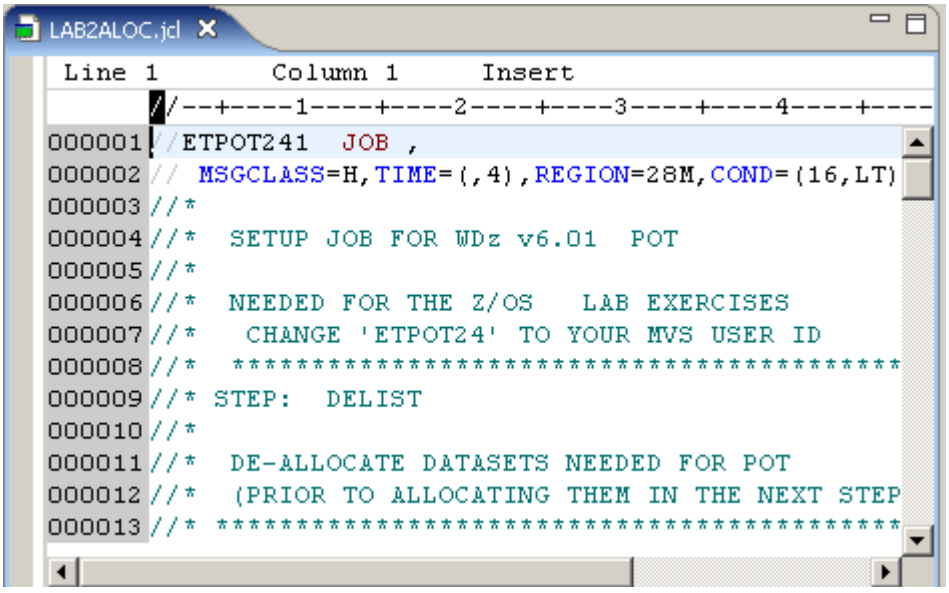


Abbildung 2.2.4

Das dargestellte Prototype JCL Script benutzt die UserID “ETPOT24”. ETPOT24 taucht in dem Script häufig auf. Alle Instanzen von ETPOT24 müssen durch Ihre eigene UserID ersetzt werden.

Die folgenden Schritte zeigen auf, wie ETPOT24 durch Ihre eigene User ID ersetzt wird.

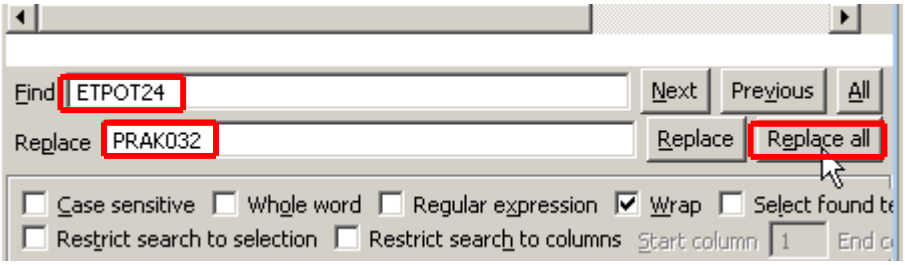


Abbildung 2.2.5

Die Search Facility CTRL+ F öffnen. In dem “Find” Feld ETPOT24 eingeben. In dem Replace Feld Ihre UserID eingeben(in diesem Beispiel ist die UserID = PRAK032). Es ist wichtig, Großbuchstaben zu benutzen. 1k auf „Replace all”.

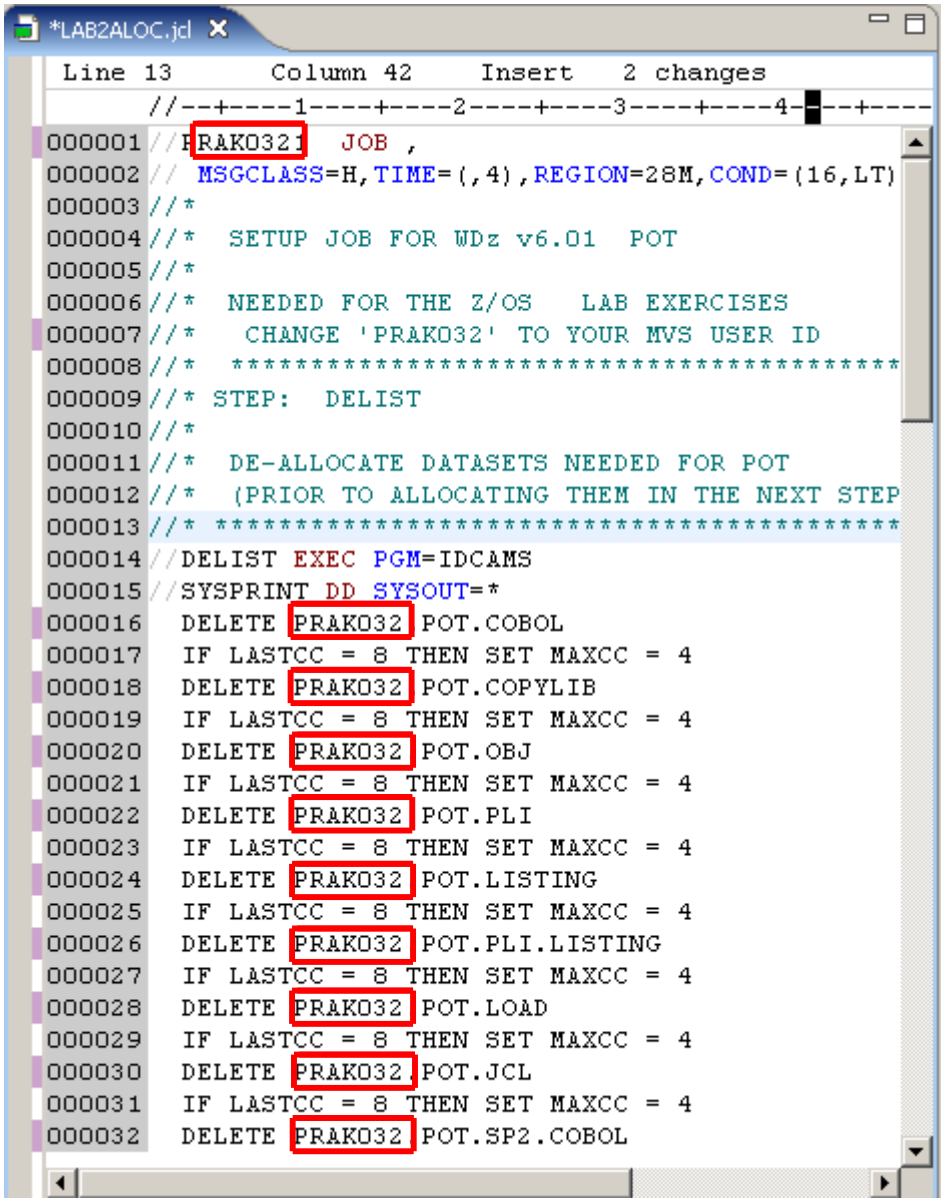


Abbildung 2.2.6

Das Ergebnis sollte ähnlich wie hier dargestellt aussehen.

Scroll down im Editor Fenster um die Statements anzusehen, welche das Allocate der Data Sets bewirken.

1k innerhalb des Editor Fensters, um die find/replace Function zu verbergen. Die erforderlichen Änderungen sind geschehen. Um die permanent zu machen, CTRL + S eingeben.

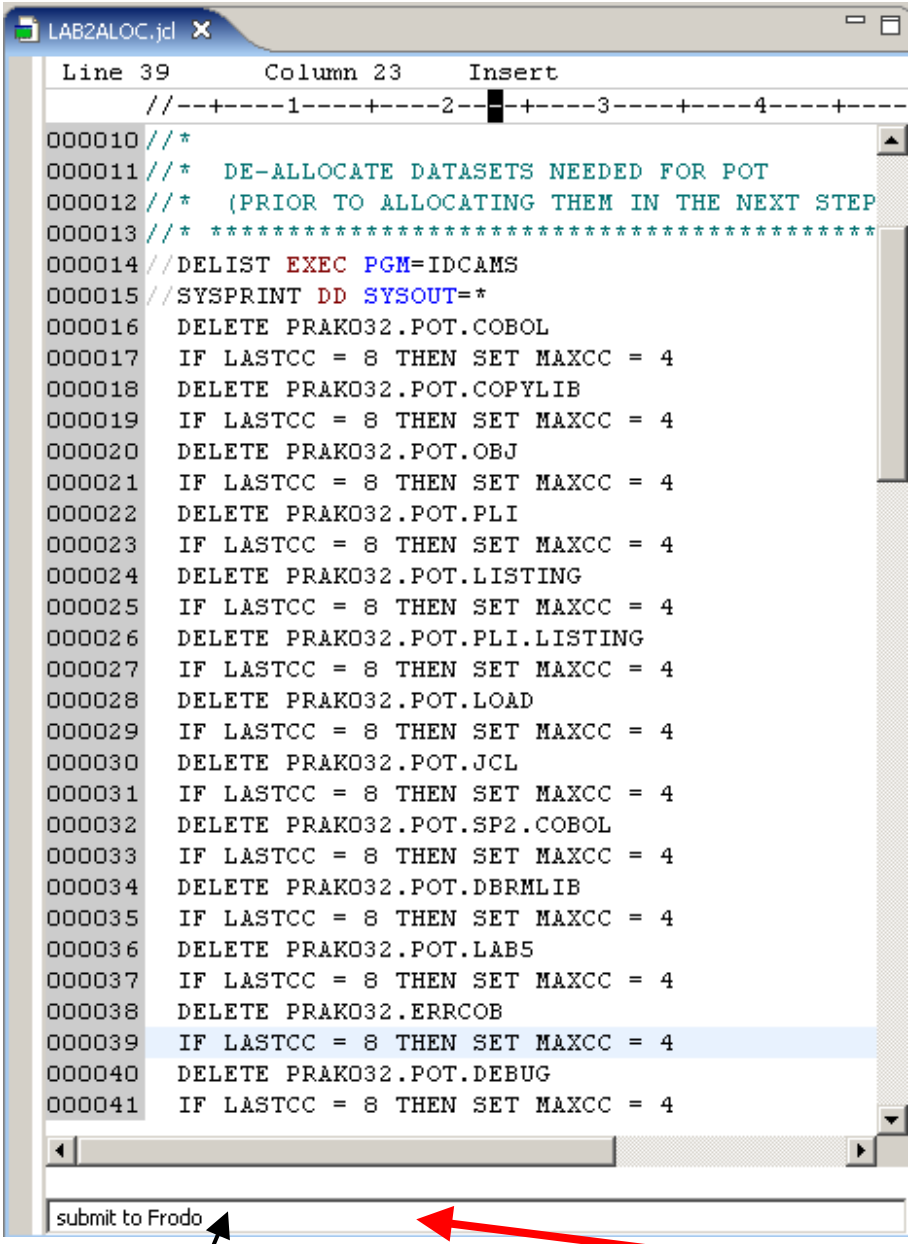


Abbildung 2.2.7

Sie haben das lokale JCL-Script auf Ihrer Workstation geändert.

Wir werden jetzt das Skript an Ihre Benutzerkennung auf dem fernen z/OS-Server senden. Eigentlich ist dies das gleiche wie mit TSO ein JCL-Script zu generieren und das „Sub“-Kommando einzugeben.

Denken Sie daran, das JCL-Script ist auf Ihrer Workstation gespeichert. Sie könnten das JC-Skript auf Ihr remote z/OS-System kopieren und dort ausführen (submit) es dort. Stattdessen können Sie auf Ihrer lokalen Workstation das Command „Submit to Frodo“ eingeben, was zu dem gleichen Ergebnis führt.

Hierzu selektieren sie das LAB2ALOC.jcl Editor-Fenster, und drücken Sie die Esc-Taste. Der Cursor wird auf der Kommandozeile verschoben.

„Submit to Frodo“ auf der Kommandozeile eingeben, und drücken Sie Enter.

(Hinweis: Stellen Sie vorher sicher, dass Sie an Frodo angeschlossen sind) !

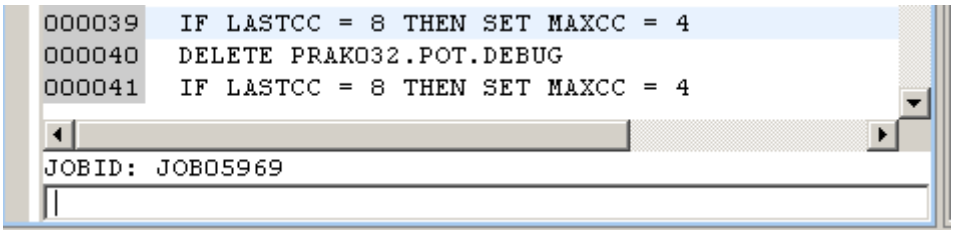


Abbildung 2.2.8

Wurde der Hob erfolgreich submitted, erhalten Sie eine JOBID (in diesem Beispiel ist die JOBID = JOB05969).

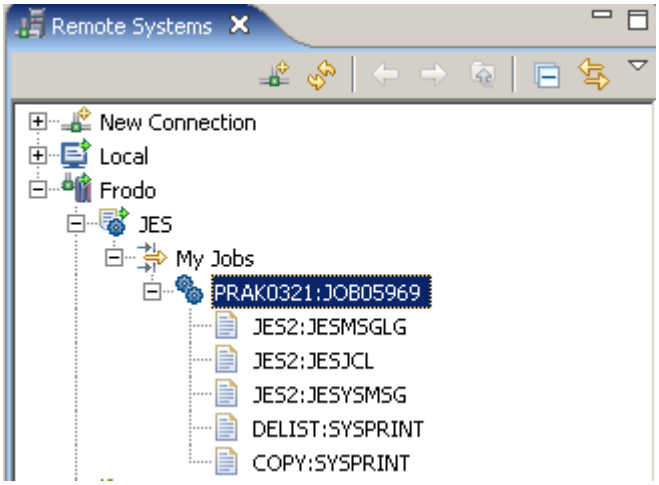


Abbildung 2.2.9

Sie haben den Job erfolgreich submitted. Dennoch sollten sie verifizieren, dass keine Fehler aufgetreten sind.

Hierfür, im Remote Systems View, expandiere JES →My Jobs. Dort finden Sie die gerade generierte JOBID (in diesem Beispiel JOB05969). Um dies sichtbar zu machen, kann es sein, dass Sie für Frodo ein disconnect und reconnect durchführen müssen.

2k auf diesen Eintrag um den Editor zu öffnen.

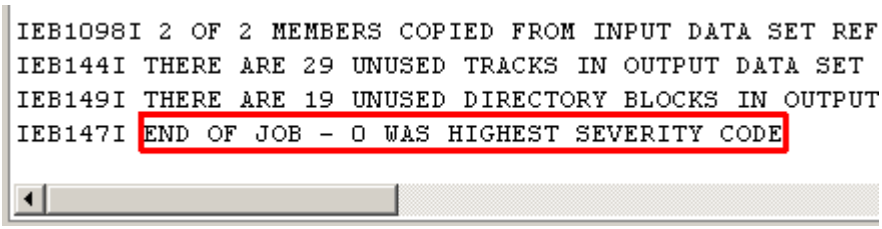


Abbildung 2.2.10

Scroll down zum Ende. Wenn die Ausführung erfolgreich war, sollte die Botschaft wie oben dargestellt aussehen.

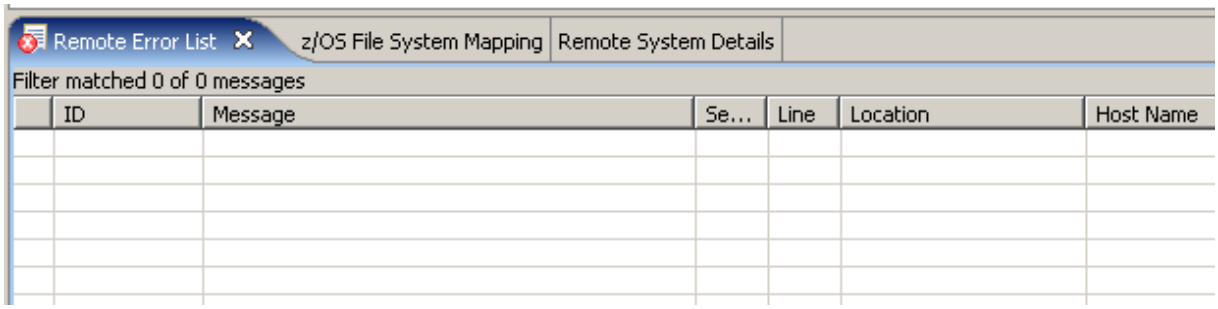


Abbildung 2.2.11

Ebenso sollte die Remote Error List leer sein, und bis zum Ende von RDz Tutorial 03 leer bleiben.

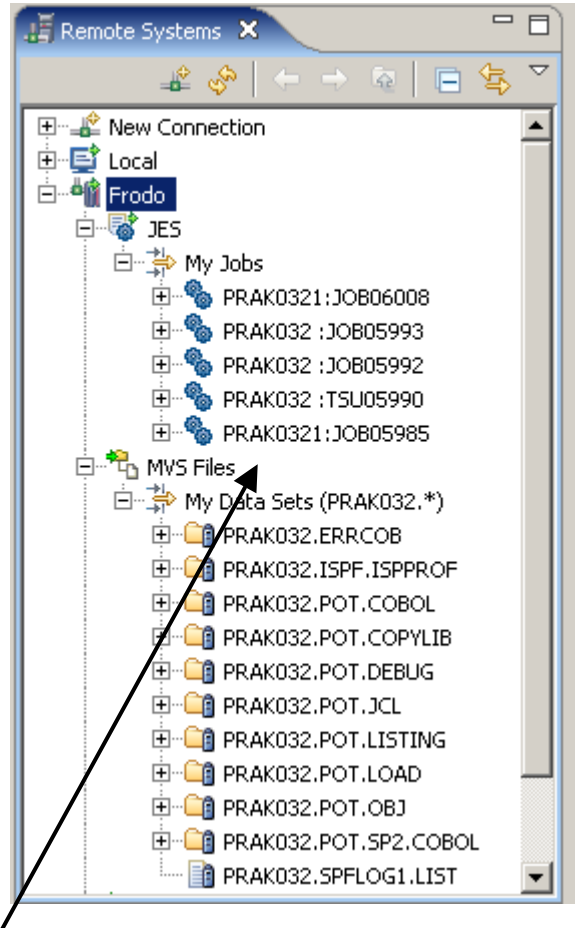


Abbildung 2.2.12

Im Remote Systems View enthält "My Data Sets" jetzt 11 Data Sets, 9 von ihnen neu. Das lokale JCL Script LAB2ALOC.jcl wurde auf dem z/OS-Host mit Hilfe des „Submit to Frodo“ Commands ausgeführt. Es generiert die 9 neuen Datensätze. Vielleicht möchten Sie einen Blick auf LAB2ALOC.jcl im Edit-Fenster werfen, und die entsprechenden Zuweisungen betrachten.

Es kann sein, dass Sie weitere Datensätze sehen, wenn Sie diese in einer früheren Sitzung mit Ihrem Benutzer-ID generiert haben.

Die unter MVS Files angezeigte Information ist identisch mit derjenigen, welche Sie sehen, wenn Sie sich mit Ihrem 3270 Emulator unter TSO und ISPF einloggen, und das Data Set List Utility Panel (3,4) aufrufen.

Selbst-Test

- **Sie hätten das gleiche Ergebnis auch erreichen können, wenn sie die benötigten 9 Data Sets wie bisher zu Fuß angelegt hätten. Schauen Sie sich den Abschnitt 2.2 nochmals an. Ist dies wirklich eine Arbeitserleichterung ?**
- **Sieht das anders aus, wenn sie häufig neue Cobol Programme schreiben ? Müssen die 9 Data Sets dann jedes Mal neu angelegt werden ? Wann ja und wann nein ?**

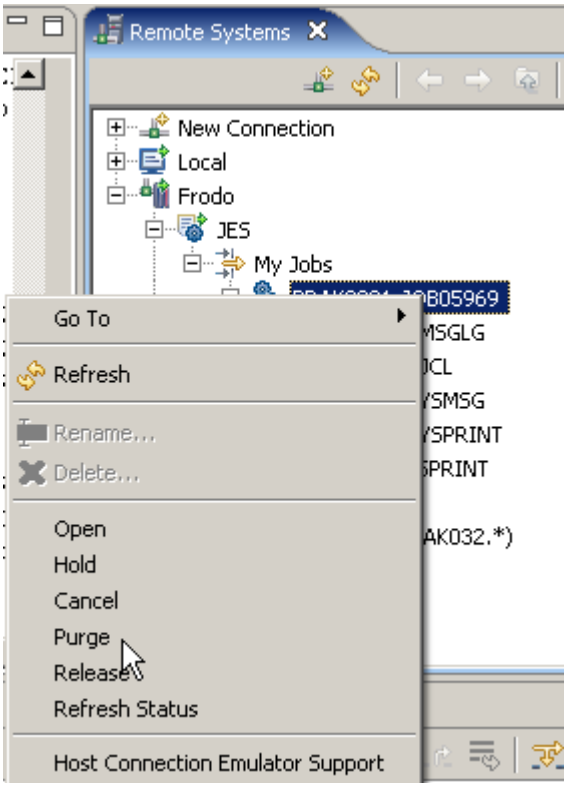


Abbildung 2.2.13

Wenn Sie einen Job Submitten, werden die Zwischenergebnisse in temporären Data Sets, den sog. Spool Files zwischengespeichert. Diese Spool Files für die Jobs laufen mit der Zeit voll. Deshalb ist es eine gute Idee, den Inhalt von Zeit zu Zeit zu leeren (purge). Um diese Aufgabe zu erfüllen, 1kr auf einen Job, und selektieren Sie Purge aus dem Kontext-Menü.

Bei einem längeren arbeiten mit RDz ist dies absolut notwendig. Bitte jedoch nicht gerade jetzt. Sie wollen die Jobs nicht löschen, die Sie gerade erstellt haben.

72.3 Host Code Page einstellen

Bei der Benutzung von Windows kennen Sie den Ärger, die richtige Code Page einzustellen. Bei der Verwendung der falschen Code Page werden z.B. die deutschen Umlaute oder das ß durch merkwürdige Buchstabenkombinationen dargestellt.

z/OS ist lange ohne Code Pages ausgekommen. Der Grund ist, der EBCDIC Code ist ein 8 Bit Code, und stellt die Sonderzeichen der wichtigsten europäischen Alphabete automatisch richtig dar. Der ASCII Code ist im Gegensatz dazu ein 7 Bit Code.

In der 2.Hälfte der 90er Jahre führte aber auch z/OS Code Pages ein. Die Standard z/OS Code Page is IBM-037, Local Codepage CP1252.

Wir empfehlen, bei der Benutzung von RDz eine andere Code Page zu benutzen. Wenn Sie die Code Page ändern wollen, gehen Sie in den Remote System View, und dann :

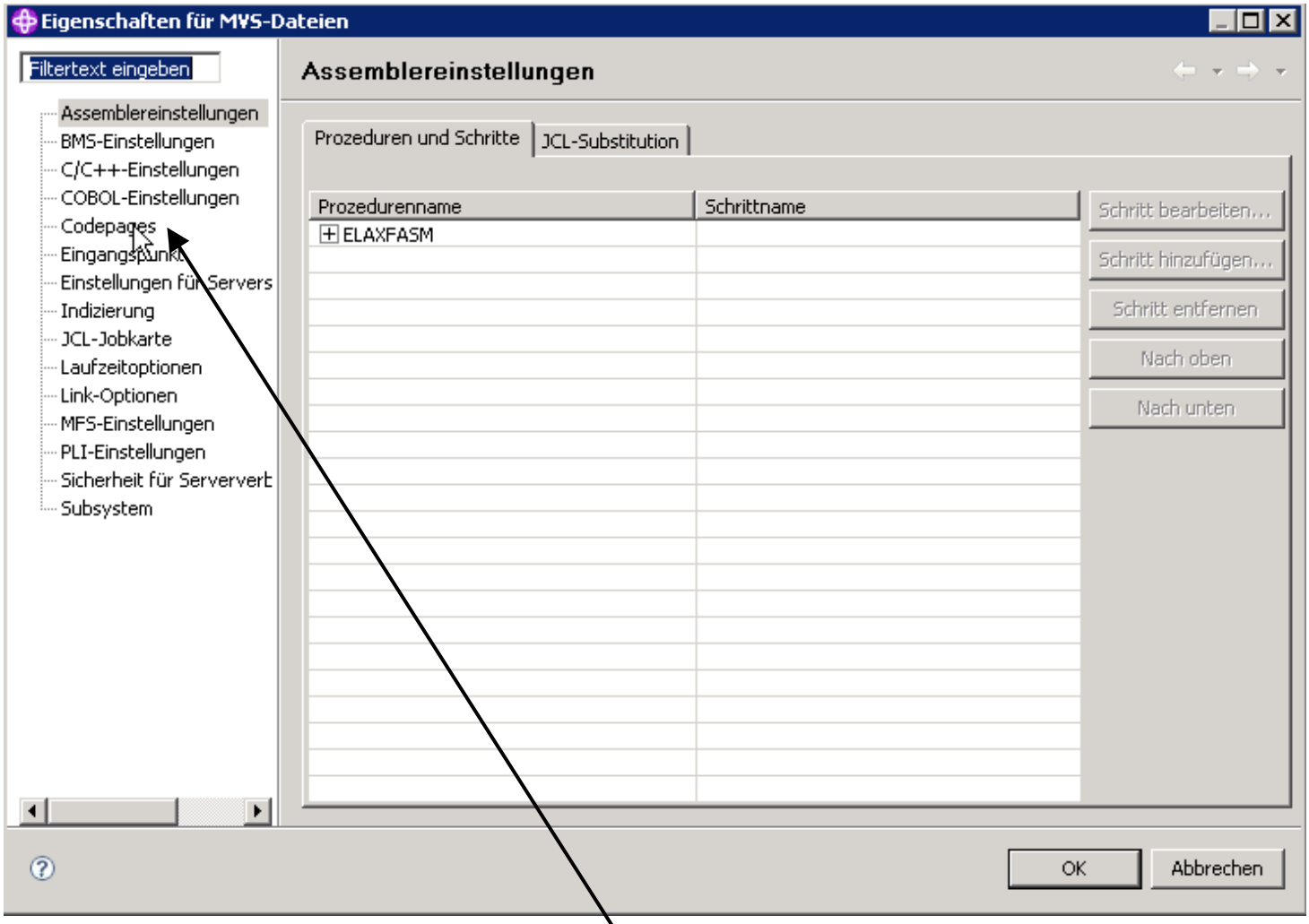


Abbildung 2.3.1

Frodo → MVS Files 1kr → Properties → 1k auf Code Pages

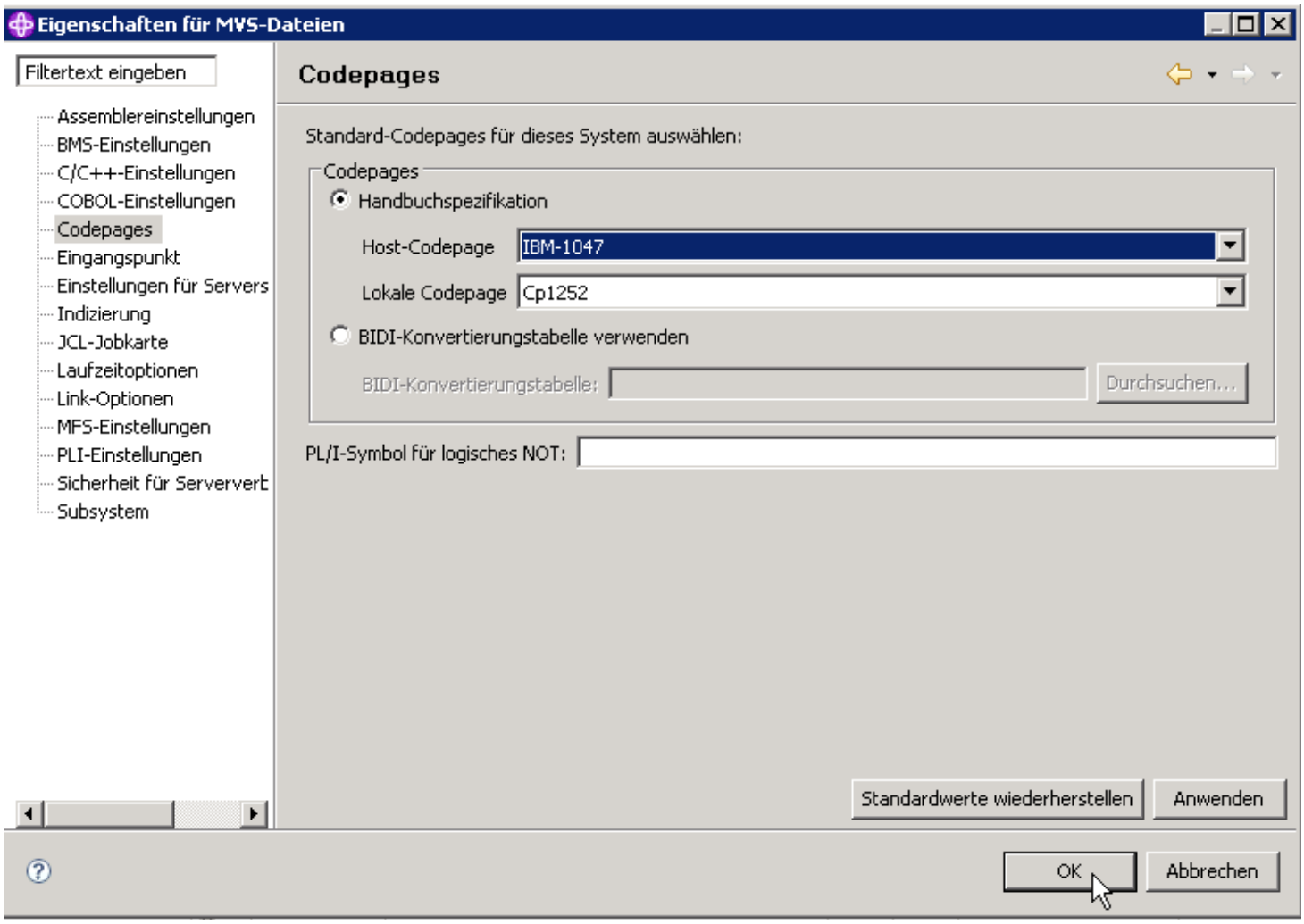


Abbildung 2.3.2

Jetzt selektiere IBM.1047, Local Codepage CP1252, 1k auf ok. Damit werden Sie hoffentlich einigen Schwierigkeiten aus dem Wege gehen.

Nichts ist vollkommen. Ein berüchtigtes Beispiel ist die Darstellung von rechteckigen Klammern beim Programmieren in C/C++. In Problemfällen hilft es, die Benutzung unterschiedlicher z/OS Code Pages auszuprobieren.

2.4. Emulation des 3270 Green Screens

RDz besitzt auch einen integrierten 3270-Emulator, den Sie statt Ihres normalen 3270 Emulators (z.B. Quick32 Freeware Edition). Nachfolgend wird dessen Aufruf demonstriert:

- **Rechtsclick auf die Remote-Verbindung, die mit Hilfe eines 3270-Emulators aufgebaut werden soll.**
- **"Unterstützung für Hostverbindungsemulator" auswählen**

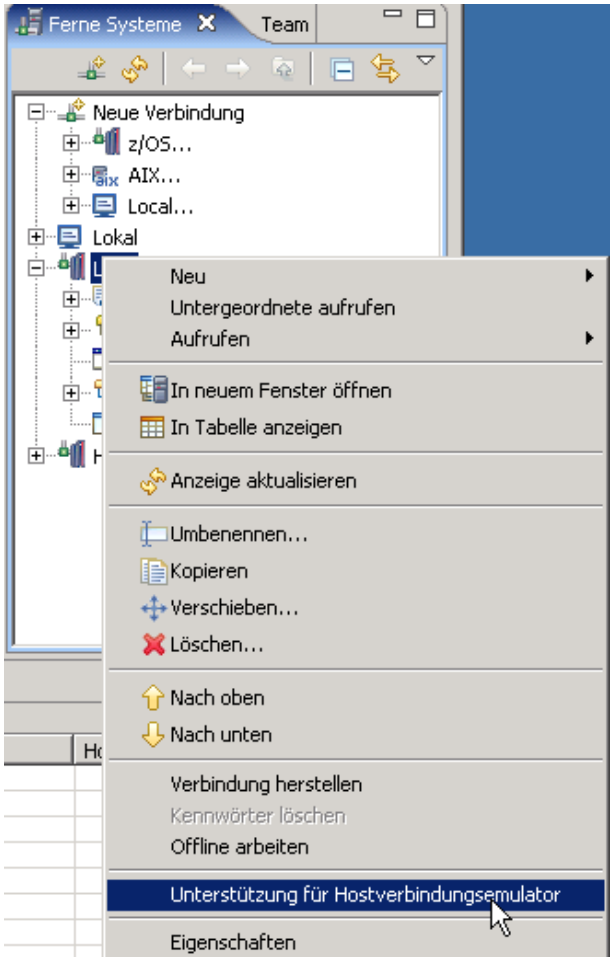


Abbildung 2.4.1

Es wird die Verbindung via eines 3270-Emulators aufgebaut. Der bekannte Green Screen erscheint im Zentrum der RDz-Arbeitsfläche.

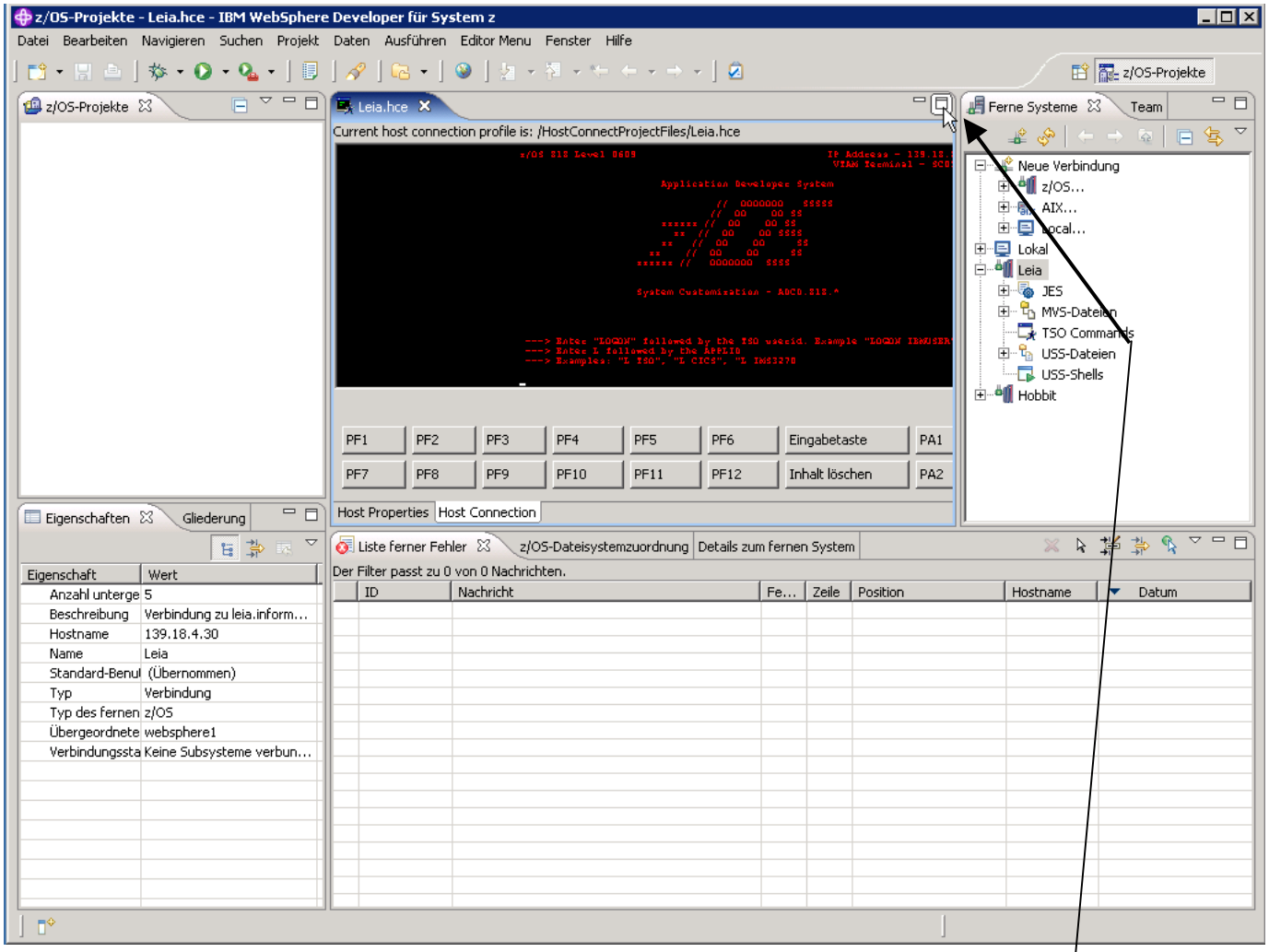


Abbildung 2.4.2

Um den 3270-Emulator auf die ganze RDz-Arbeitsfläche zu vergrößern, auf die Schaltfläche "Maximieren" klicken (siehe Abbildung 7.2).

Mit diesem integrierten 3270-Emulator kann man die gewohnten Aktionen ausführen, z.B.

- **Einloggen und Arbeiten im TSO**
- **Einloggen und Arbeiten im CICS**

Beispielsweise kann man sich mittels "L TSO", Mainframe-Login und Passwort ins TSO Subsystem einloggen, oder mittels "L CICS", Mainframe-Login und Passwort ins CICS Subsystem.

3.0 Working with MVS files

Wir sind jetzt in der Lage, mit z/OS Assets zu arbeiten (besonders z/OS Datasets).

Wir haben 3 Cobol Quell-Programme REGIO0A,cbl, REGIO0B.cbl, und REGIO0C.cbl für Sie präpariert, welche Sie in dem lokalen Verzeichnis C:\WDz_Tutorials\Tutorial03 finden können. Wir werden diese letztendlich kompilieren, linken und ausführen.

In Abb. 2.2.12 hatten wir neu erstellte Data Sets wiedergegeben, welche mit unseren Settings für die Cobol Programmierung notwendig sind. Zusätzlich werden wir jetzt einen weiteren neuen PDS (Partition Data Set) unter Ihrer z/OS User ID erstellen. Wir werden die vorgefertigten 3 Cobol Programme REGIO0A,cbl, REGIO0B.cbl, und REGIO0C.cbl in Member dieses Partitioned Data Sets kopieren, sowie Filter benutzen um Daten zu betrachten.

3.1 Allocating a Data Set

Erinnern Sie sich, in Ihrem allerersten Tutorial, Dateiverwaltung unter TSO, war das Allocate eines Data Sets ihre erste Aufgabe. Für das Cobol RDz Tutorial machen wir das Gleiche, ausgenommen, dass wir hierfür RDz benutzen.

Im Remote Systems View, expandiere “MVS Files”.

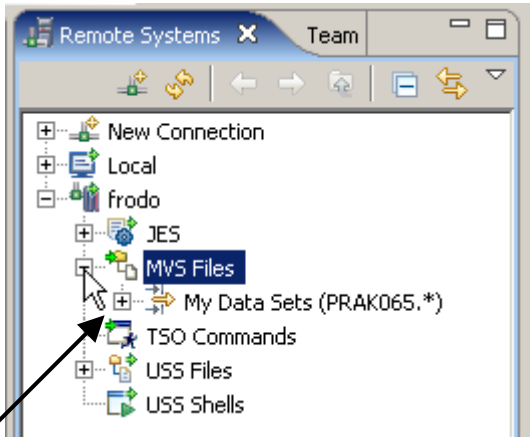
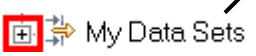


Abbildung 3.1.1

1k auf  **um alle Data Sets zu sehen. (Keine Angst, wenn bereits viele Data Sets allocated sind; wir werden ein refresh des Systems in Kürze durchführen).**

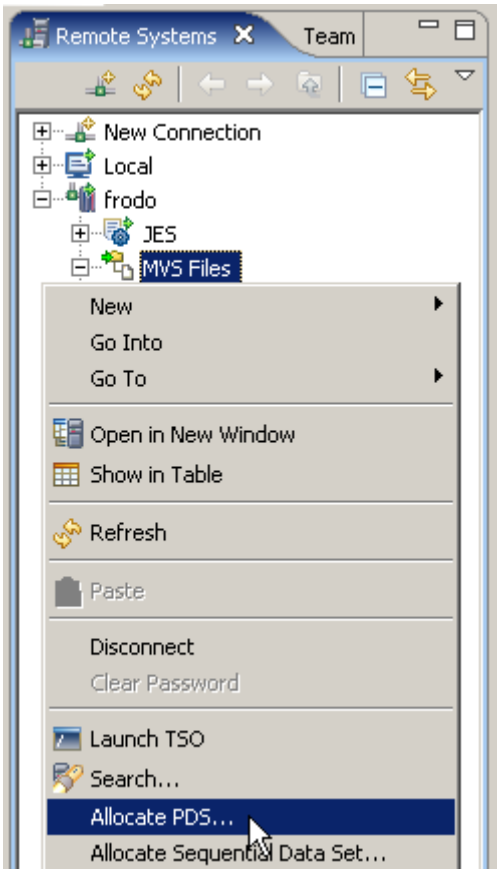


Abbildung 3.1.2

1kr auf MVS Files. Selektiere “Allocate PDS” aus dem Kontext Menu.

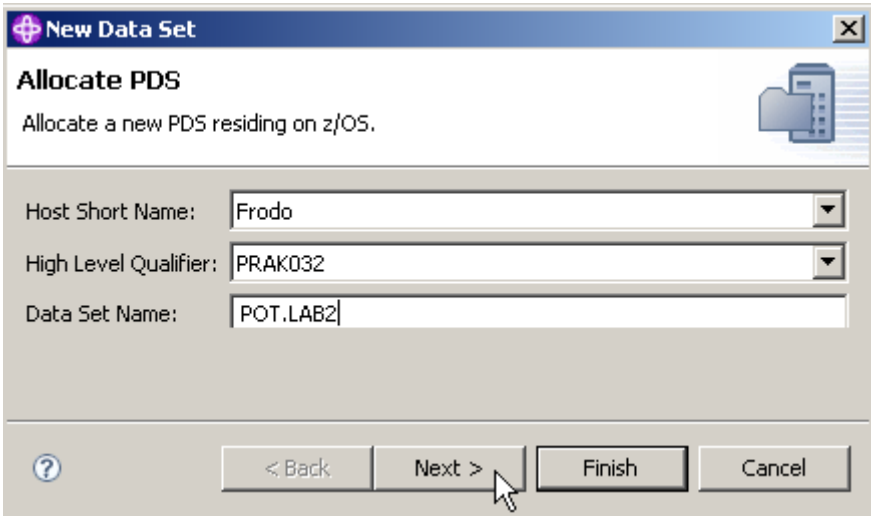


Abbildung 3.1.3

Im “Allocate PDS” Panel benutzen sie die Drop Down Liste um den Namen Frodo zu selektieren und Ihren High Level Qualifier (PRAK032 in diesem Beispiel). Geben Sie POT.LAB2 als den Data Set Name Namen ein. Hiermit soll ein Data Set allocated werden, der später Ihr Cobol Programm auf nimmt. 1k auf Next.

Die hier als Intermediate Level Qualifier gewählten Buchstaben “POT” sind eine Abkürzung für “Proof of Technology”.

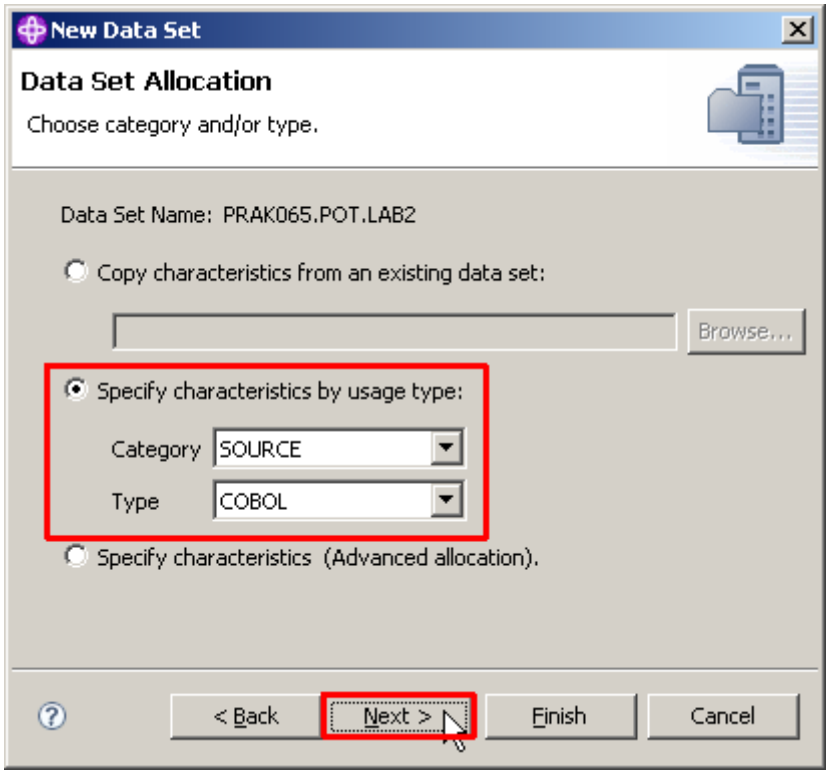


Abbildung 3.1.4

Im Data Set Allocation-Panel, selektieren Sie "Specify characteristics by usage type“. Wählen Sie SOURCE für Kategorie und COBOL für Type. 1k auf Weiter.

Dies ist ein nettes Feature, da es einen Datensatz allocated, der die erforderlichen Eigenschaften für das Speichern von COBOL Quellcode aufweist.

Die Alternativen für SOURCE (Quellcode) wäre z.B. LISTING, oder andere. Die Alternativen für COBOL wäre ASM (Assembler), C/C++, JCL, PLI und REXX.

Anmerkung: Ganz im Gegensatz zu der x86 Architektur ist die Assembler Programmierung unter z/OS überraschend einfach. Die Bedeutung von z/OS Assembler ist in den letzten Jahren allerdings sehr stark zurückgegangen. Dennoch gibt es immer wieder Fälle, in denen z/OS Assembler eine Rolle spielt. Assembler spielt unter z/OS eine bedeutendere Rolle als unter Linux, Unix und Windows.

Click next

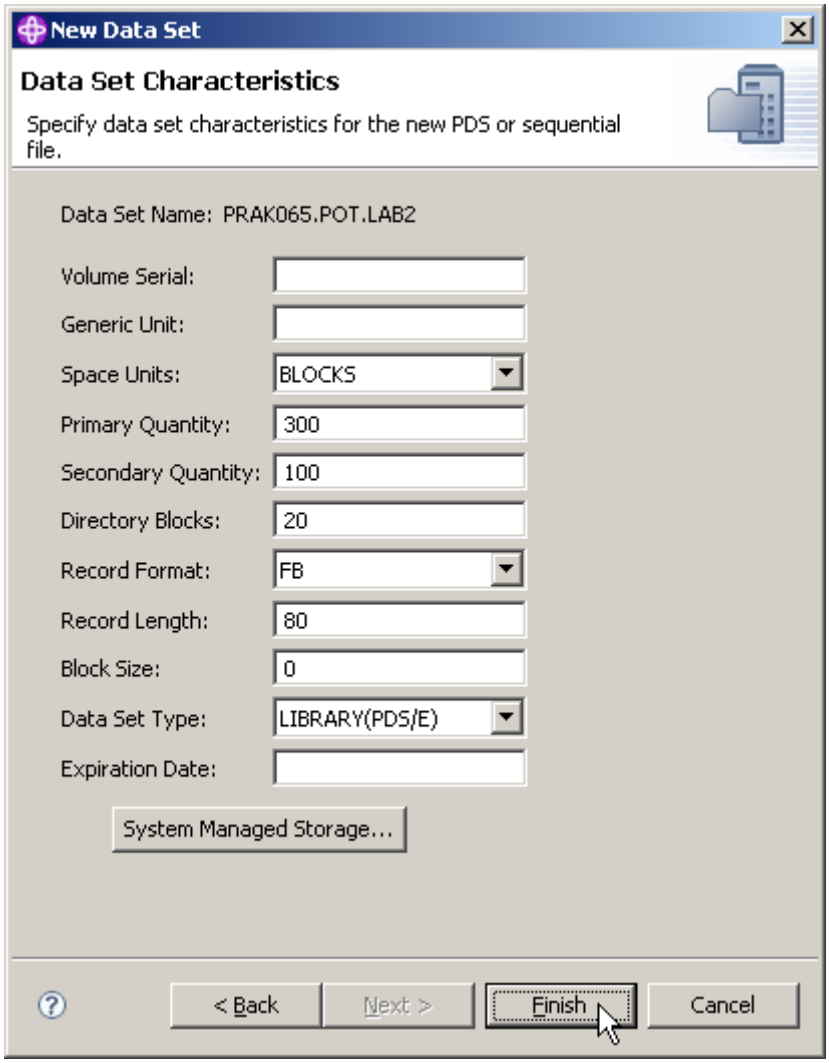


Abbildung 3.1.5

Dies sieht ähnlich wie der vertraute TSO Allocate Screen aus dem allerersten Tutorial (Dateiverwaltung unter TSO) aus, und hat die gleiche Funktion. Übernehmen sie die Default Werte und click Finish.

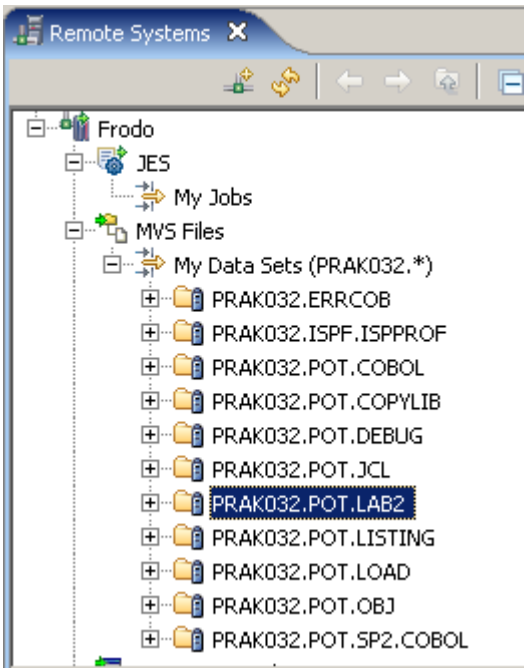


Abbildung 3.1.6

Unter der Annahme, dass Plattenspeicherplatz unter z/OS verfügbar ist, wird der Partitioned Data Set PRAK032.POT.LAB2 angelegt. Zu diesem Zeitpunkt hat er noch keine Member.

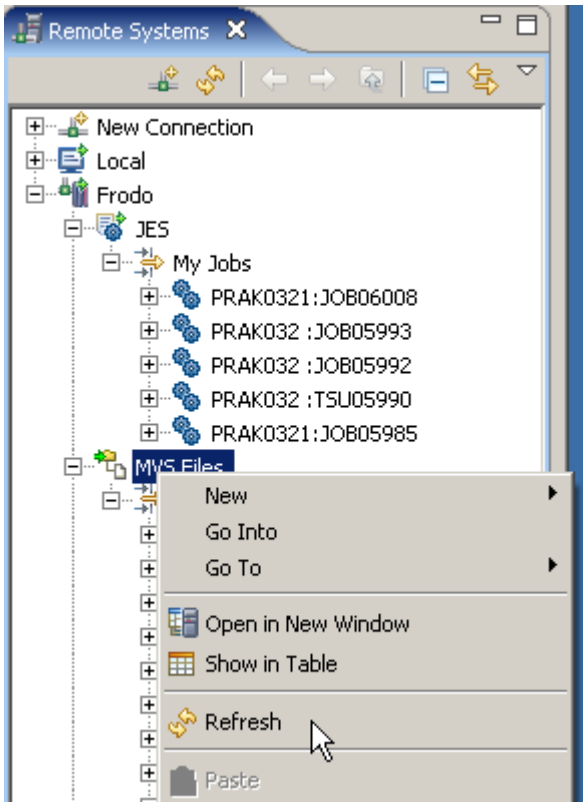


Abbildung 3.1.7

Wenn Sie ihn jetzt nicht sehen, 1kr auf MVS Files, und selektiere Refresh (disconnecting und reconnectiong Frodo würde den gleichen Effekt haben).

3.2 Mapping Data Sets

RDz speichert Identische Kopien einer File sowohl unter z/OS als auch auf der Workstation, die von RDz synchron manipuliert werden. Erfahrungsgemäß ist dies nicht ganz leicht zu verstehen, deshalb Abschnitt 3.2 sorgfältig lesen.

Es existieren unterschiedliche Konventionen für die Namensgebung von Dateien unter Windows und unter z/OS. Wir möchten eine Kopie der gleichen Datei unter beiden Betriebssystemen speichern und bearbeiten.

Das Problem ist, eine Datei Bezeichnung wie PRAK032.POT.COBOL oder PRAK032.POT.LAB2 ist kein gültiger Name für eine Windows FAT32 oder NTFS Datei.

Erinnern wir uns, bei einem Data Set Namen wie PRAK032.POT.COBOL bezeichnen wir „PRAK032“ als den High Level Qualifier (HLQ) und „COBOL“ als den Lowest Level Qualifier (LLQ).

Ein “Data Set Mapping” assoziiert den Lowest Level Qualifier in jedem MVS Data Set mit einer File-Name Extension, welche das Betriebssystem Ihrer Workstation(auf der RDz läuft), versteht. Beispielsweise auf dem Mainframe haben wir Data Sets wie PRAK032.POT.COBOL oder PRAK032.POT.JCL. Wir mappen **COBOL auf cbl, und **JCL auf jcl, usw. Hierbei repräsentieren die beiden Asterics „ ** “ die beiden Qualifier PRAK032 und POT in dem Data Set Namen.

Dies ist erforderlich, damit die Workstation weiß, mit welchen Arten von Daten sie es zu tun hat. Das Gleiche ist auch für die Member eines PDS Data Sets möglich.

Wir führen jetzt ein Mapping für die Member des gerade neu angelegten z/OS PDS Data Sets PRAK032.POT.LAB2 durch. Spezifisch generieren wir drei Member, die Cobol Quellcode enthalten, mit den Namen

- **PRAK032.POT.LAB2(REGI0A)**
- **PRAK032.POT.LAB2(REGI0B)**
- **PRAK032.POT.LAB2(REGI0C)**

Wenn wir fertig sind, können Sie diese drei Member sich im Remote Systems View ansehen.

Gleichzeitig werden wir auf der Windows Workstation ein Directory PRAK032.POT.LAB2 einrichten, und in diesem Directory drei Files mit den Namen

- **REGI0A.cbl**
- **REGI0B.cbl**
- **REGI0C.cbl**

unterbringen, welche den identischen Cobol Quellcode wie die drei Member des PDS Data Sets enthalten.

Die drei Member des PDS Data Sets erhalten das Mapping cbl. Damit wird erreicht, dass RDz auf eine für den Benutzer transparente Art wahlweise mit den Membern des Datasets auf dem z/OS System oder den Files unter Windows arbeiten kann.

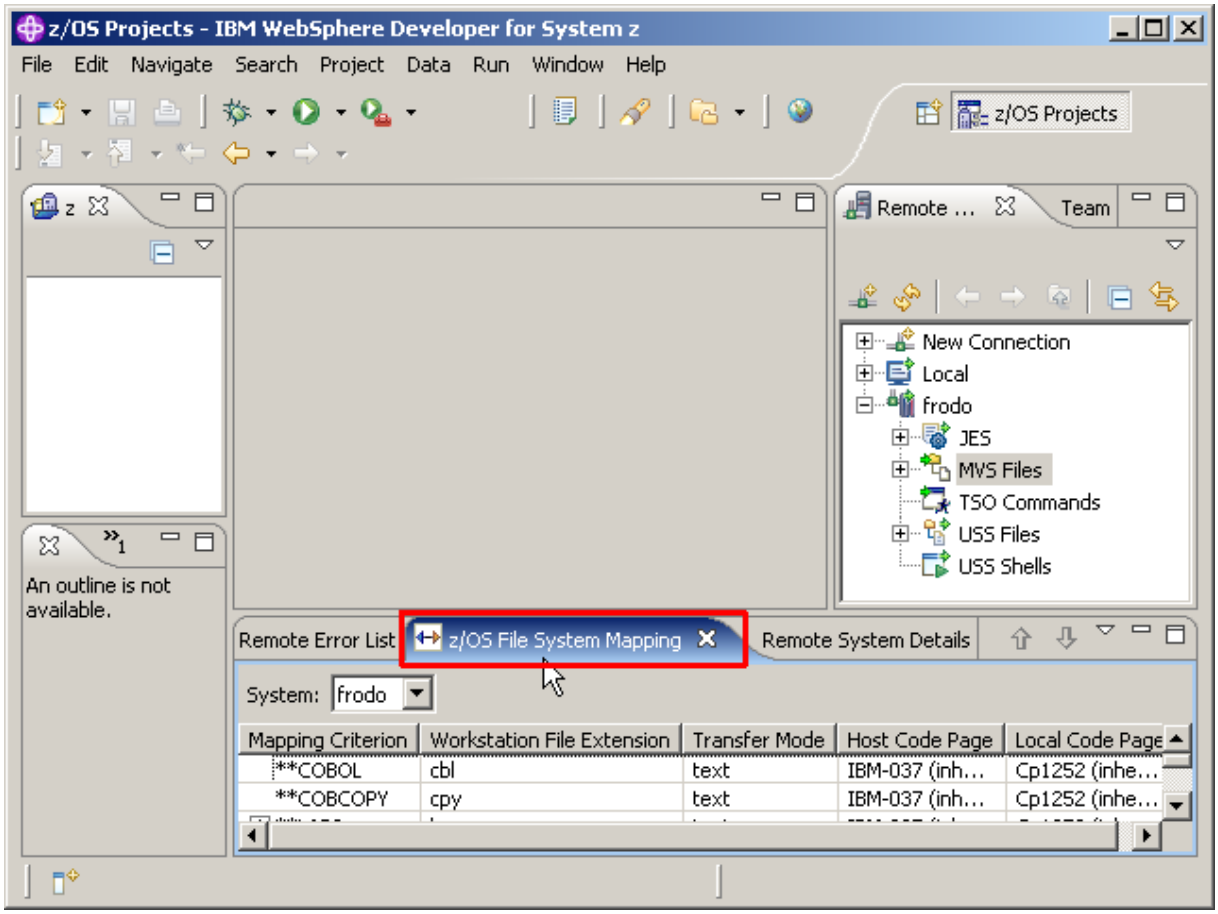


Abbildung 3.2.1

Wechseln Sie zu dem z/OS File System Mapping View mit 1k auf den Tab.

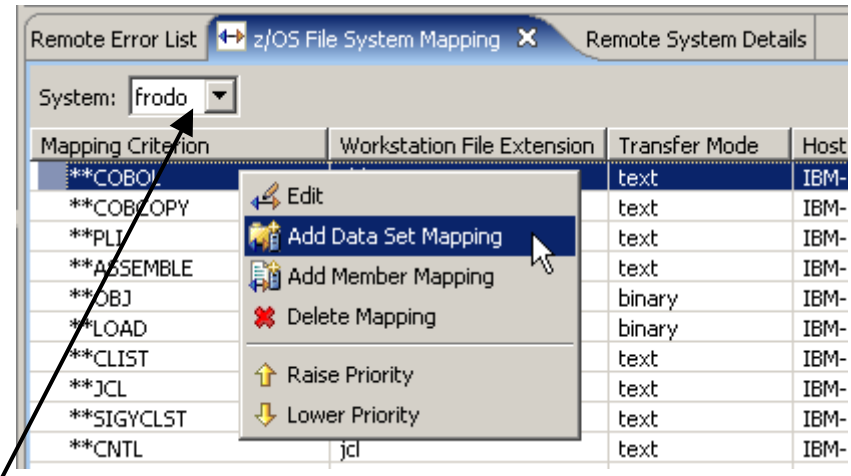


Abbildung 3.2.2

In Abb. 3.2.2 wird eine Liste von Standard mäßig vorgeschlagenen Mappings dargestellt, z.B. cbl für **Cobol Data Sets, oder cpy für **COBCOPY Data Sets (Cobol Copy Books).

1kr auf **COBOL, selektiere “Add Data Set Mapping”.

Wir wollen jetzt zu der Liste den Data Set PRAK032.POT.LAB2 hinzufügen, den wir gerade erstellt haben. “LAB2 ist dessen LLQ. Verifizieren Sie hierzu den Namen Ihrer Host Connection (Frodo in diesem Beispiel).

Wir machen dies in 2 Schritten. Im ersten Schritt werden alle Member des Data Sets PRAK032.POT.LAB2 als „bms“ abgebildet (mapped). In einem 2. Schritt wird eine Untermenge der Member von PRAK032.POT.LAB2 als “cbl” abgebildet. Dies geschieht für solche Member, deren Name mit den 4 Buchstaben REGI anfängt.

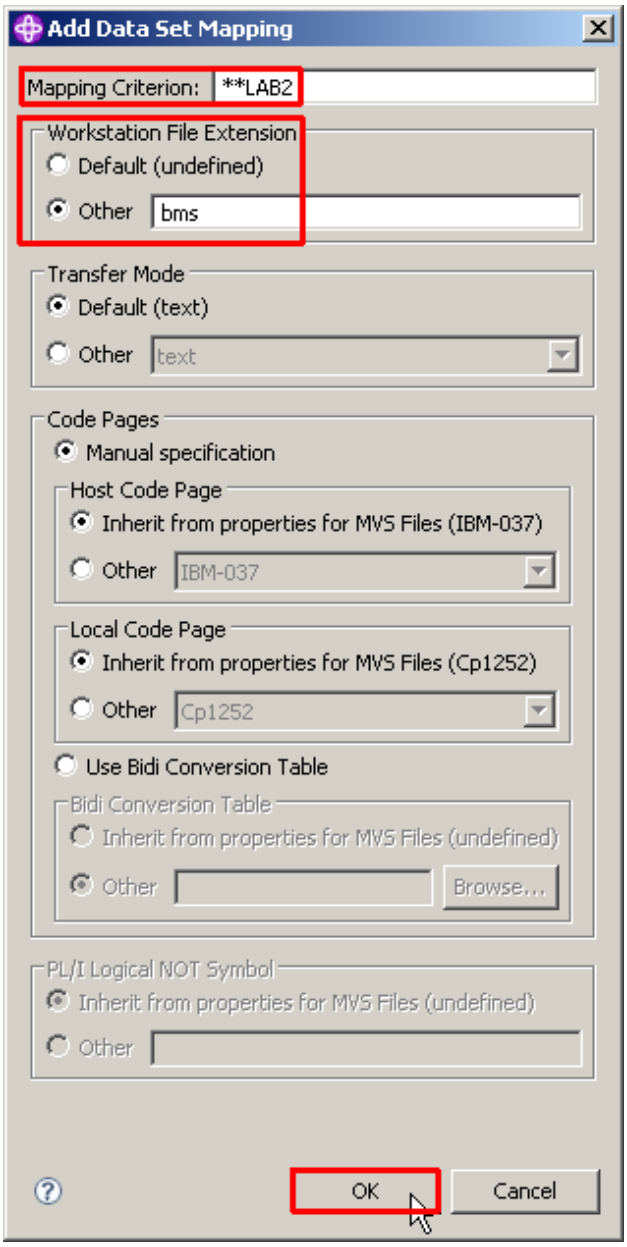


Abbildung 3.2.3

Als Mapping Criterion den Wert **LAB2 eingeben (für PRAK032.POT.LAB2), als „Workstation File Extension“ selektiere „Other“, und dann „bms“ eingeben. Alle Member dieses Data Sets werden als bms abgebildet. Die restlichen Default Werte unverändert übernehmen. 1k auf OK.

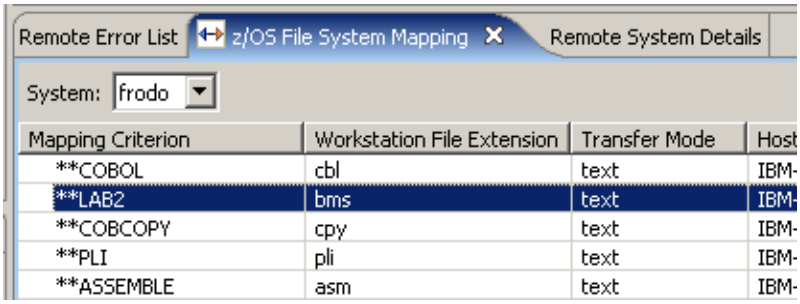


Abbildung 3.2.4

Als Ergebnis wurde **LAB2 erfolgreich zu der Liste hinzugefügt. Alle Member von **LAB2 erhalten das Mapping bms. Und damit zu Schritt 2.

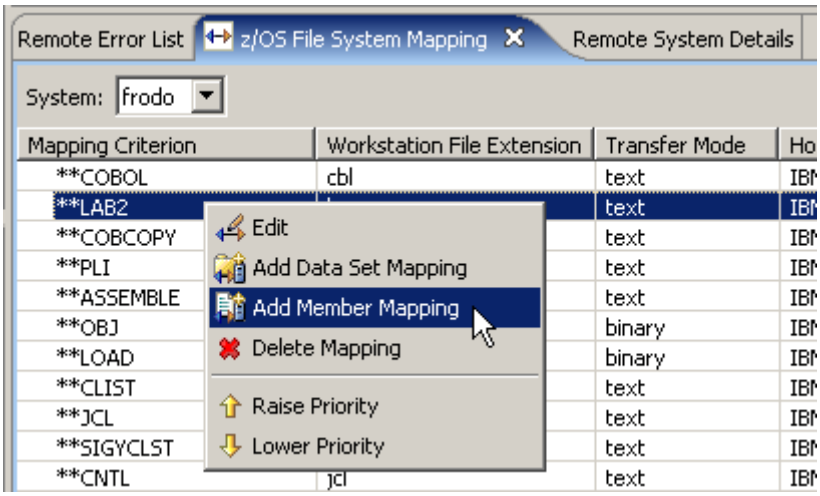


Abbildung 3.2.5

1kr auf **LAB2, selektiere Add Member Mapping (nicht Data Set Mapping wie vorher).

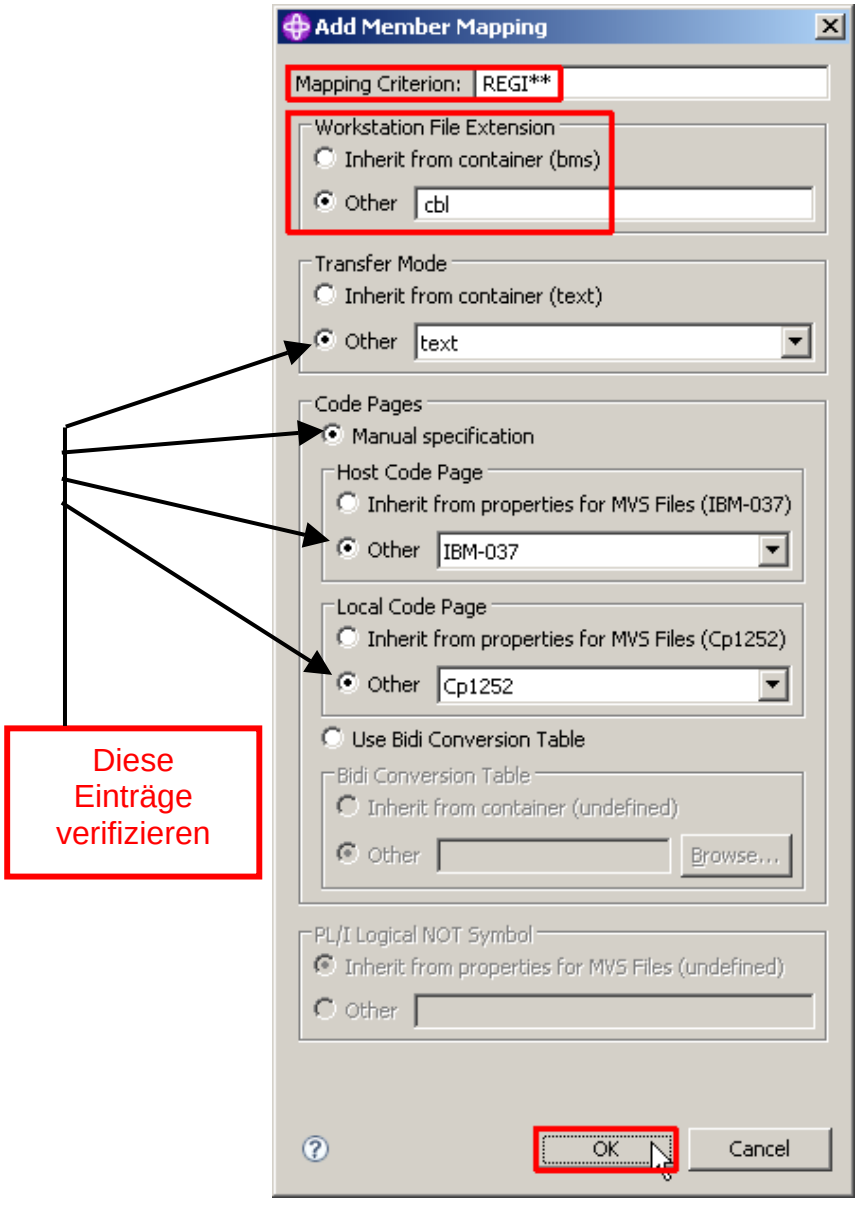


Abbildung 3.2.6

Als Mapping Criterion REGI eingeben, als Workstation File Extension selektiere Other, und cbl eingeben. Für den Rest die gezeigten Default Werte übernehmen. Click OK.**

Erklärung: Alle Member des Data Sets mit der Endung **LAB2 werden als bms abgebildet (Schritt 1, Data Set Mapping), mit Ausnahme der Member, deren Name mit REGI anfängt (in Schritt 2, Add-Member Mapping). Letztere werden auf cbl abgebildet. Die ASTERICS ** stellen die Qualifier des Data Set Namens dar.**

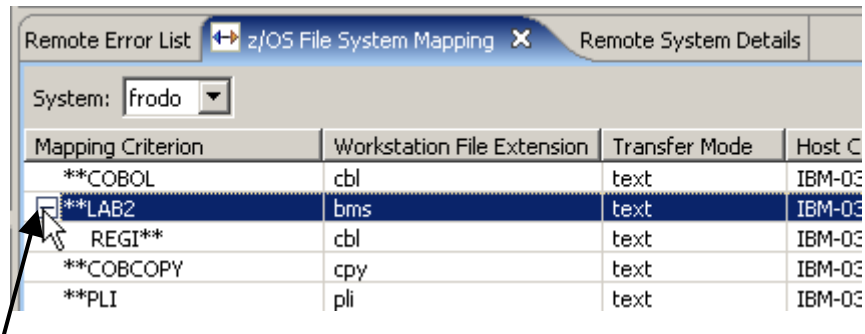


Abbildung 3.2.7

Expandiere das + von **LAB2. Sie sehen ein anderes Mapping Kriterium unter **LAB2. Damit wird jeder Member, der mit den 4 Buchstaben REGI im Partitioned Data Sets (PDS) **LAB2 anfängt, als cbl (COBOL) abgebildet. Die restlichen Member werden als bms abgebildet.**

Hiermit haben wir erreicht, dass alle Member des Data Sets PRAK032.POT.LAB2, die mit den 4 Buchstaben REGI anfangen, unter Windows als Files mit der Extension cbl bearbeitet werden können

Selbst-Test

- **Befinden sich jetzt in dem Data Set PRAK032.POT.LAB2 bereits die drei Member Regio0A, Regio0B und Regio0C ?**
- **Wenn nein, was haben wir dann erreicht ?**

3.3 Kopieren lokaler Files in den allocated z/OS Data Set PRAK032.POT.LAB2

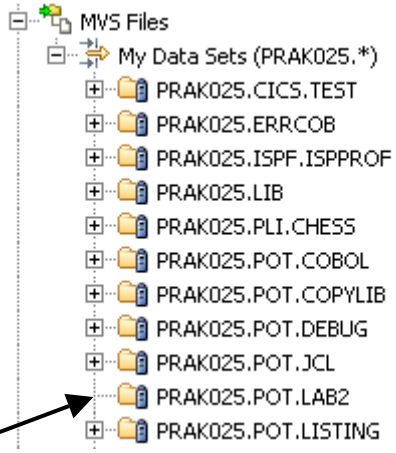


Abbildung 3.3.1

Der allocated Partitioned Data Set PRAK032.POT.LAB2 ist leer (hat keine Member).

Wir kopieren jetzt drei COBOL Programme in drei Member des Partitioned Data Set PRAK032.POT.LAB2. Diese 3 Cobol Programme befinden sich bereits in dem Windows XP Directory Ihrer virtuellen Gast Maschine unter den Namen REGIO0A.cbl, REGIO0B.cbl, und REGIO0C.cbl. Nach dem Kopieren sollen sie unter z/OS als die drei Member

- **PRAK032.POT.LAB2(REGI0A),**
- **PRAK032.POT.LAB2(REGI0B), und**
- **PRAK032.POT.LAB2(REGI0C)**

verfügbar sein.

1k auf den Remote Systems View, 1k auf das + Zeichen der Node Local Files. Navigieren Sie zu dem Verzeichnis C:\WDz_Tutorials\Tutorial03\Resources.

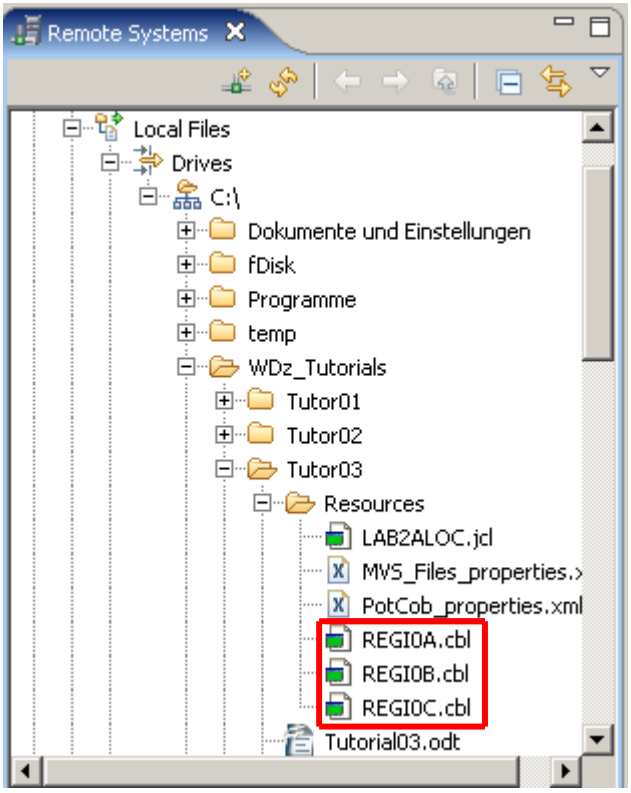


Abbildung 3.3.2

1k auf REGI0A.cbl, Shift Taste halten, und 1k auf REGI0C.cbl um alle drei cbl-Files zu markieren. Alternativ 1k auf REGI0A.cbl, den Ctr Key halten, 1k auf REGI0B.cbl und dann auf REGI0C.cbl um alle 3 cbl-Files zu markieren.

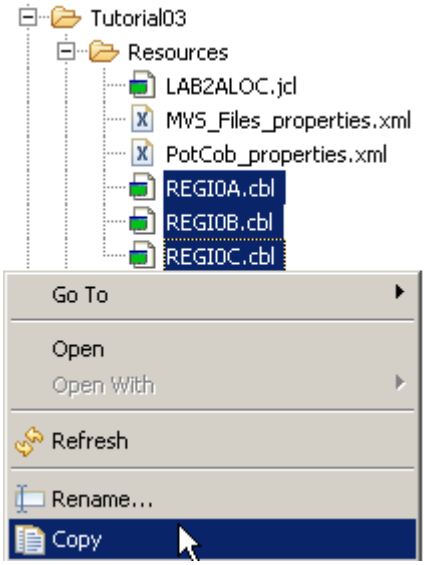


Abbildung 3.3.3

1kr auf die markierte Gruppe, select Copy. Die 3 Dateien sind nun im Clipboard.

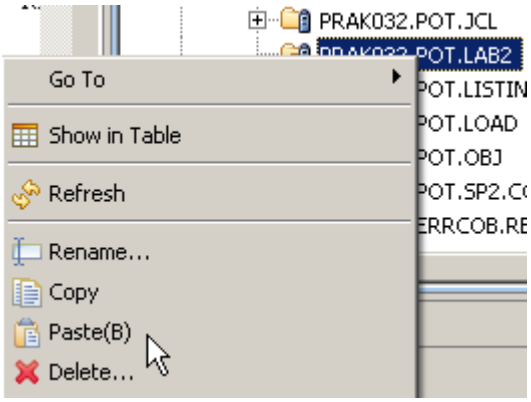


Abbildung 3.3.4

1kr auf PRAK032.POT.LAB2, selektiere Paste, und PRAK032 durch Ihre UserID ersetzen.

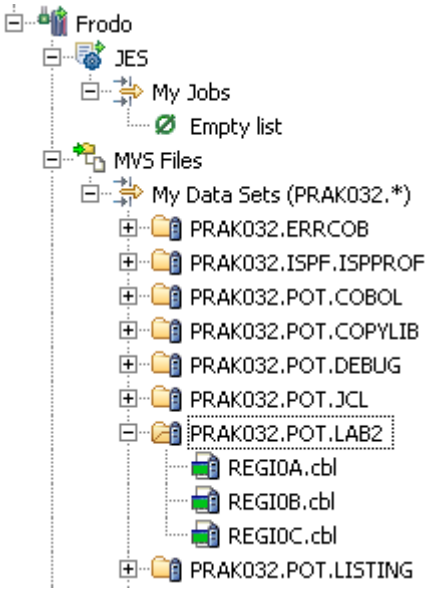


Abbildung 3.3.5

Ihr neuer PDS Data Set hat nun drei neue Members, REGI0A.cbl, REGI0B.cbl und REGI0C.cbl.

Ohne das gerade durchgeführte Member Mapping hätten die drei Files nicht die Zuordnung .cbl, die gebraucht wird, um mit den Kopien auf der Workstation synchron zu arbeiten !!!

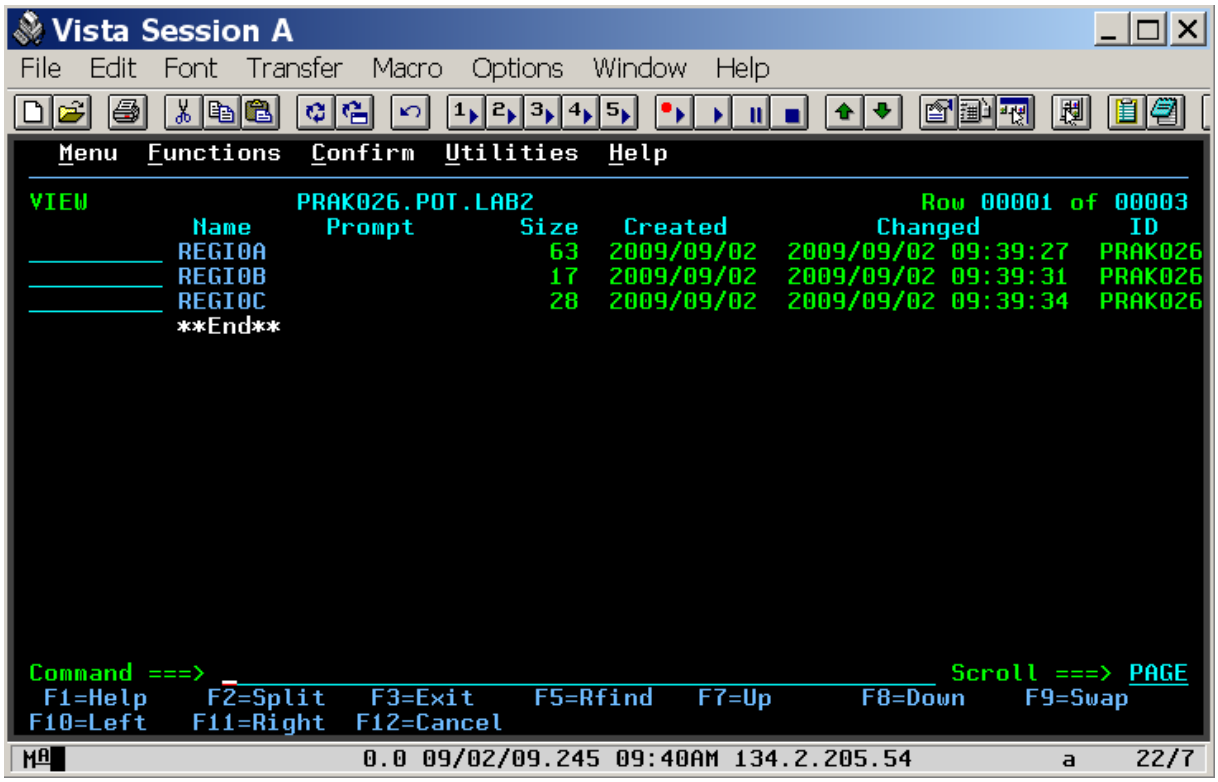


Abbildung 3.3.6

Wenn Sie unter ISPF nachsehen, PRAK032.POT.LAB2 hat nun 3 Member REGIO0A, REGIO0B, and REGIO0C . Die Zuordnung .cbl ist unter RDz, nicht aber unter ISPF sichtbar.

Selbst-Test

- **Befinden sich jetzt in dem Data Set PRAK032.POT.LAB2 bereits die drei Member Regio0A, 0B und 0C ?**
- **Wie ist das geschehen ?**

4. Arbeiten mit entfernten Data Sets

4.1 Wiederholung

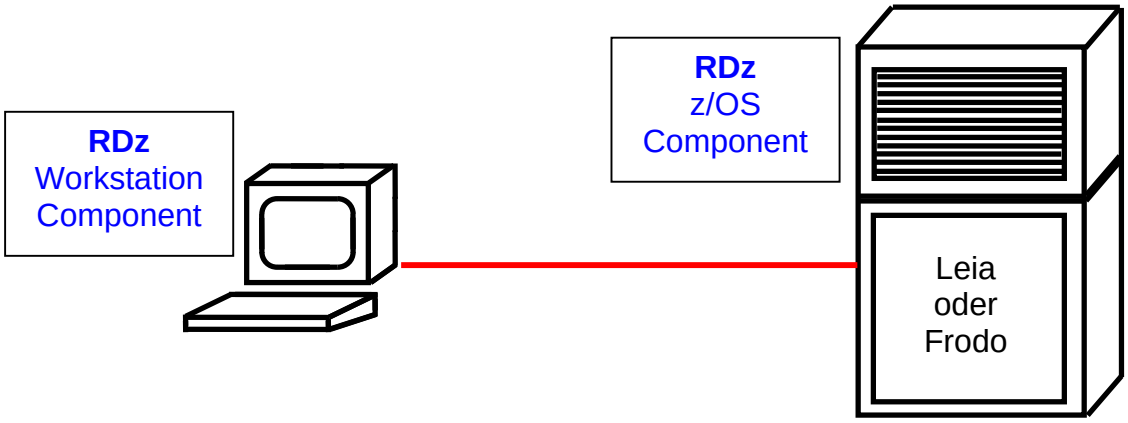


Abbildung 4.1.1

Eine übliche RDz Konfiguration besteht aus einer Workstation Komponente (installiert auf Ihrem PC), sowie einer z/OS Komponente, die auf einem Mainframe installiert ist. Die Workstation speichert Ihren Workspace.

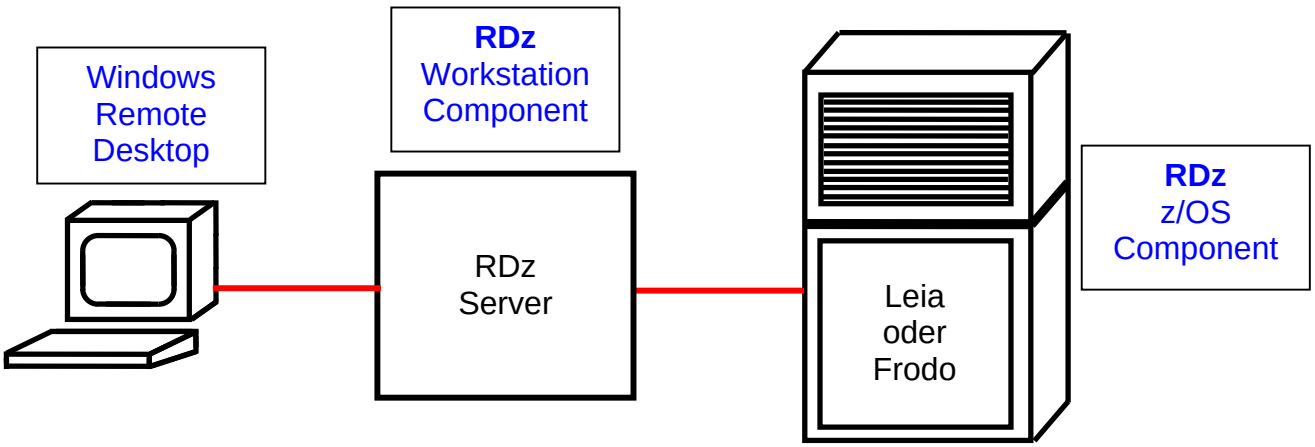


Abbildung 4.1.2

In unserem Fall benutzt Ihr PC nur den Standard Windows Remote Desktop um auf den RDz Server an der Universität Leipzig zuzugreifen. Die RDz Workstation Komponente ist auf dem Leipziger RDz Server in einer virtuellen Windows Maschine installiert. Letztere speichert auch Ihren RDz Workspace.

In beiden fällen befinden sich die Data Sets Ihres Cobol Projektes unter z/OS unter Ihrer User ID (PRAK032 in diesem Beispiel). Zusätzlich ist eine 1:1 Kopie der Data Sets in Ihrem Workspace gespeichert. Wir haben die Data Set mapping Funktion in Abschnitt 2.2 dieses Tutorials durchgeführt, um Inkompatibilitäten in der Naming Syntax für Windows Files und z/OS Data Sets aufzulösen..

In Abschnitt 2.3 kopierten wir die 3 Files REGI0A.cbl, REGI0B.cbl und REGI0C.cbl in Member des z/OS Data Sets PRAK032.POT.LAB2 von irgendwoher. In unserem Fall befand sich das "irgendwoher" ebenfalls auf dem RDz Server, aber außerhalb der RDz Workstation Komponente, und damit außerhalb Ihres Workspace. Wir werden später die drei Files ebenfalls in Ihren Workspace kopieren.

84.2 Arbeiten mit Projekten und Subprojekten

Das Arbeiten mit RDz erfolgt traditionell mit Hilfe von Projekten. Ein Projekt enthält eine Sammlung von Dateien. Sie können mehrere Projekte auf Ihrer Workstation speichern, aber Sie werden zu jedem Zeitpunkt nur mit einem einzigen Projekt arbeiten.

Sie definieren ein lokales Projekt, wenn es Ihre Absicht ist, COBOL (oder PL/I) Code auf der Workstation zu entwickeln und dort auszuführen. Sie arbeiten zum Beispiel mit einem lokalen Projekt, , wenn Ihre neue COBOL-Anwendung auf einem Windows-Workstation ausgeführt werden soll. Rational Developer für System z bietet eine Reihe von Tools an, mit deren Hilfe Sie lokale Projekte erstellen, welche Host-basierten Ressourcen benutzen. Beispiele für solche Host basierte Resourcen sind Enterprise Service Tools und der Database Application Generator. Lokale Projekte benutzen lokale Compiler und Übersetzer für die Syntaxprüfung. Sie zeigen Abhängigkeiten auf und bewirken Projekt-Builds. Rational Developer für System z bietet auch Werkzeuge für die Remote-Synchronisierung an. Dies erlaubt es, ein lokales Projekt mit einem Remote-System zu verbinden. Mit der Remote-Synchronisierung können Sie COBOL (und PL/I) -Anwendungen lokal entwickeln und das Ergebnis dann auf ein Remote-System übertragen

Sie definieren ein z/OS Projekt, wenn es Ihre Absicht ist, COBOL (oder PL/I) Code zu entwickeln, der unter z/OS gespeichert und dort ausgeführt wird. Für z/OS bedeutet dies, dass Sie Ihre Assets (vor allem z/OS-Datensätze) in einer Projektstruktur verknüpfen, nämlich in „z/OS-Projekte". z/OS-Projekte dienen in erster Linie als Werkzeug für das Organisieren und Arbeiten mit mehreren Load Modulen.

z/OS-Projekte sind Container, die ein oder mehrere Subprojekte enthalten. Subprojekte können entweder MVS Subprojekte oder Unix System Services (USS) Subprojekte sein, je nachdem, ob Sie mit PDS Membern in MVS-Data Sets, oder Dateien in einem USS hierarchischen Filesystem arbeiten. In beiden Fällen bleiben ihre Assets (z.B. z/OS Datensätze) auf dem Host, und sind nur logisch mit den Teilprojekten verknüpft.

Ein Teilprojekt enthält die Dateien und Programme für ein einziges Load-Modul. (Mit mehreren Teilprojekten, können Sie mehrere Lademodule mit einem einzigen Build-Befehl bauen). Teilprojekte werden in einem bestimmten Remote-System erstellt; die Eigenschaften dieses Systems wirken als Default Werte für Subprojekt Eigenschaften.

Dieses Kapitel führt Sie durch den Prozess der Erstellung eines Remote-Projects, und das Festlegen seiner Properties (Eigenschaften) für verschiedene Sprachen und Runtimes (Laufzeiten).

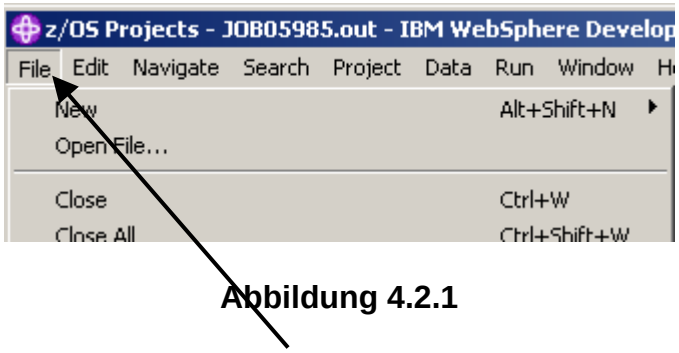


Abbildung 4.2.1

Selektiere File → New → Project… von der Menu Bar.

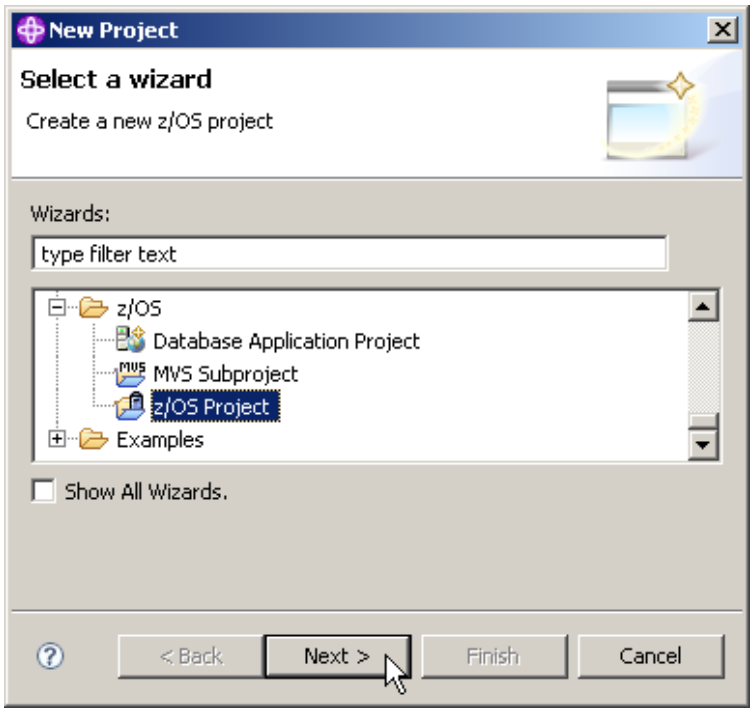


Abbildung 4.2.2

Scroll down, expandiere z/OS, selektiere z/OS Projekt, 1k auf Next.

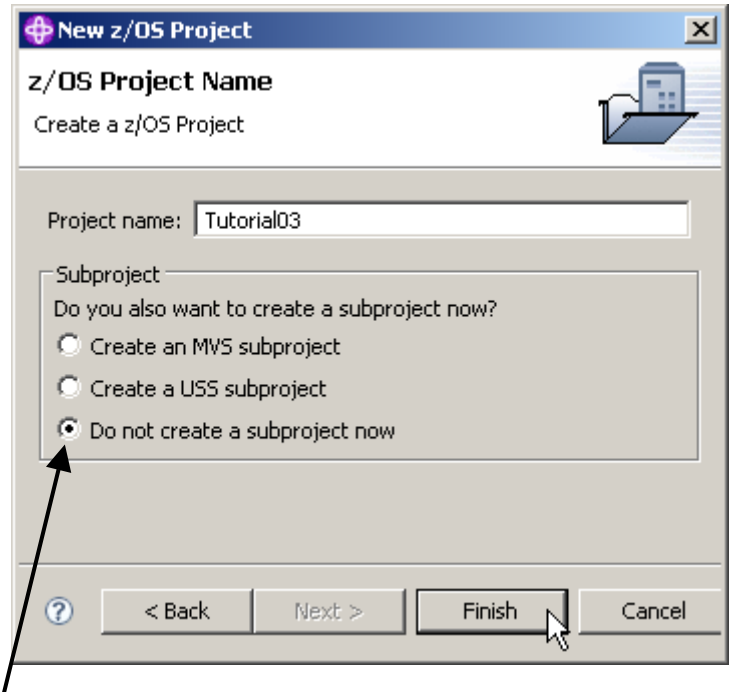


Abbildung 4.2.3

Im z/OS Project Name Panel, benennenSie das Projekt “Tutorial03”. Selektieren Sie den “Do not create a subproject now” Radio Button.

(Anmerkung: Wir werden ein Subproject später erstellen. Es könnte auch hier sofort geschehen.)

1k auf Finish. Dies erzeugt ein z/OS Patent Projekt.

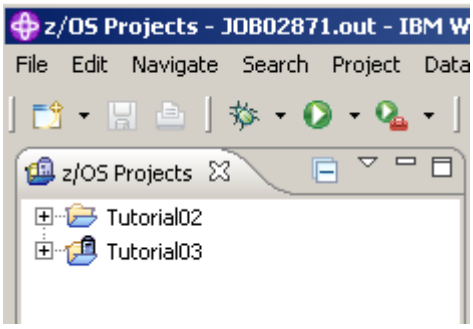


Abbildung 4.2.4

Erinnern Sie sich: Der z/OS Projects View lists lokale Projekte in Ihrem RDz Workspace. Project Tutorial02 war in dem vorhergehenden RDz Tutorial 02 “Local Cobol”erstellt worden.

4.3 Create an MVS Subproject

Zuallererst können Sie ein Mus-Subprojekt nur erstellen, wenn Sie mit dem System (Frodo in diesem Fall) verbunden sind. Ein MVS-Project befindet sich jeweils in einem von zwei Zuständen:

Im Online-Zustand ist das Projekt mit dem System verbunden, auf das sich das Projekt bezieht. Sie können die Dateien direkt ändern, die in diesem System gespeichert sind.

Im Offline-Zustand kann das Projekt nur auf Dateien auf dem Arbeitsplatzrechner zugreifen. Dies können neue Dateien oder auch Kopien von z/OS Dateien sein.

Wenn Sie sich von z/OS trennen (disconnect), können Sie die Data Sets und Member angeben, die zum Arbeitsplatzrechner übertragen werden sollen. Wenn Sie zurück zum Online-Zustand umschalten, werden die angegebenen Dateien zum Großrechner automatisch hoch geladen. Dies erfolgt mit einer Bestätigungsmeldung, welche Sie davon abhält, Resourcen unbeabsichtigt zu überschreiben.

In dem z/OS Projects View, 1kr auf Tutorial03. Selektiere New → MVS Subproject von dem Context Menü.

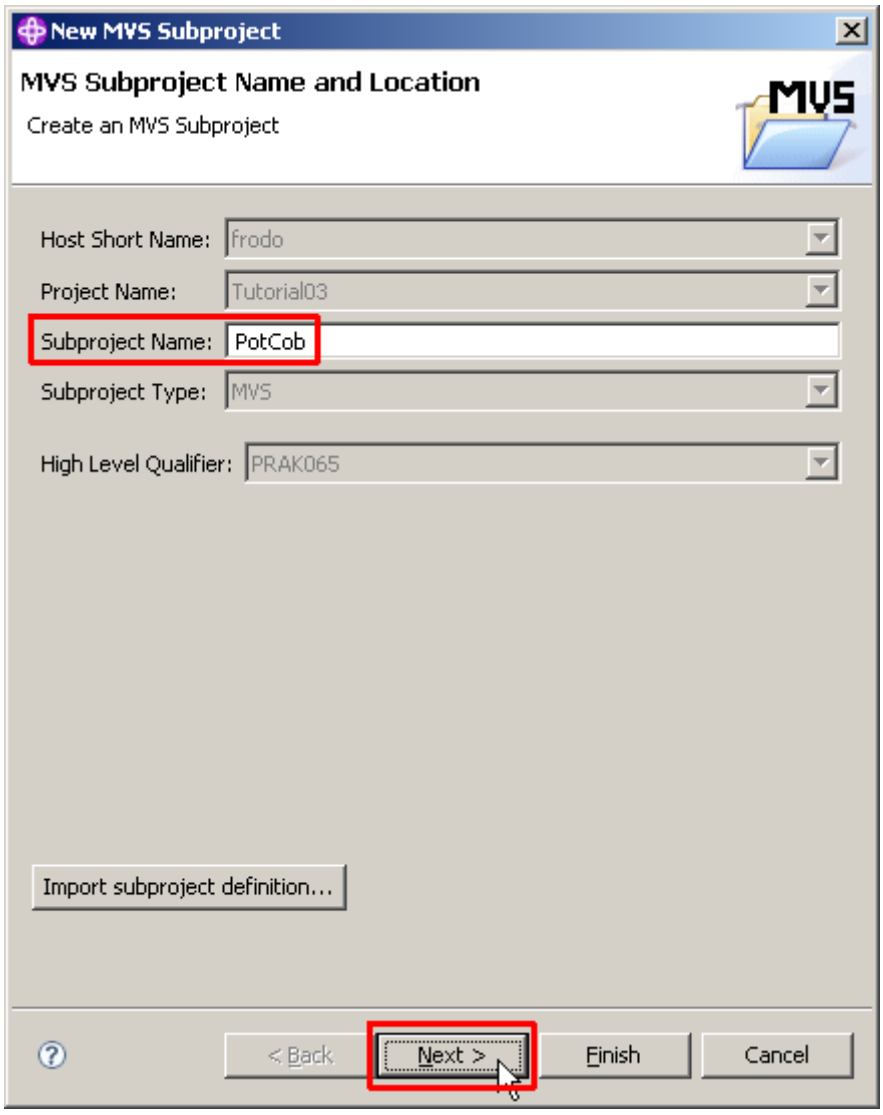


Abbildung 4.3.1

Benennen Sie Ihr MVS Subproject PotCob, und click Next..

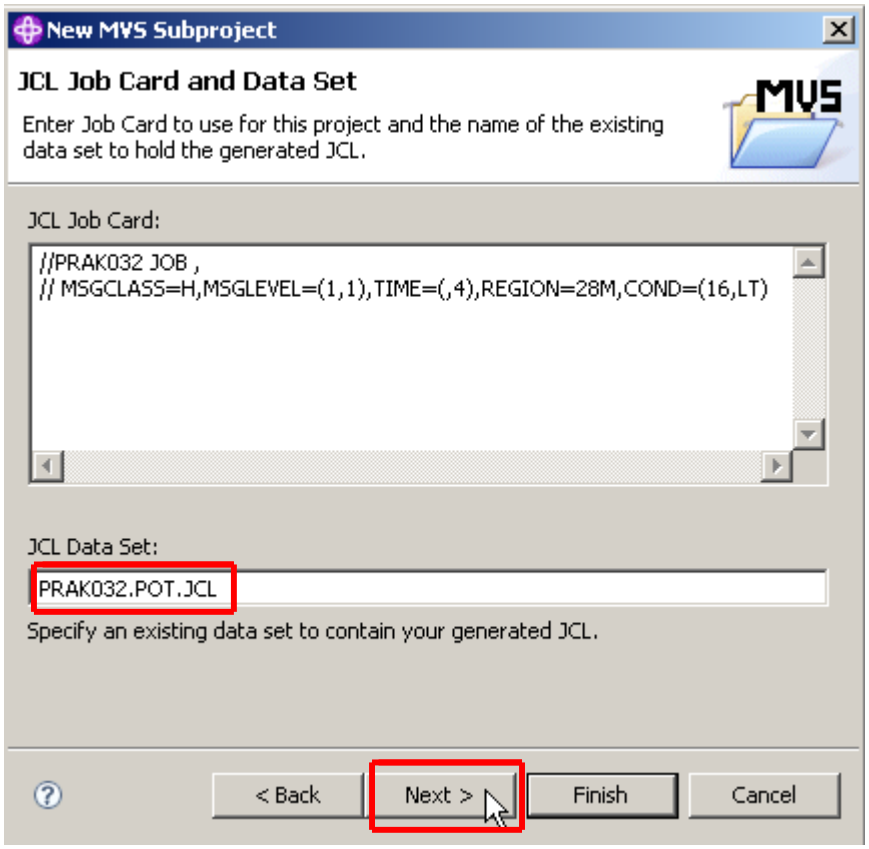


Abbildung 4.3.2

Auf dem “JCL Job Card and JCL Data Set panel”, ändere den JCL Data Set to PRAK###.POT.JCL where ### Ihre UserID ist (032 in diesem Beispiel). Nicht den Default Wert! (PRAK###.GENERTED.JCL) benutzen !!!!!

Erinnern Sie sich: Die Job Card wurde von den z/OS System Settings geerbt (inherited), die wir in Abschnitt 2.1 „z/OS System-Einstellungen“ spezifiziert hatten)

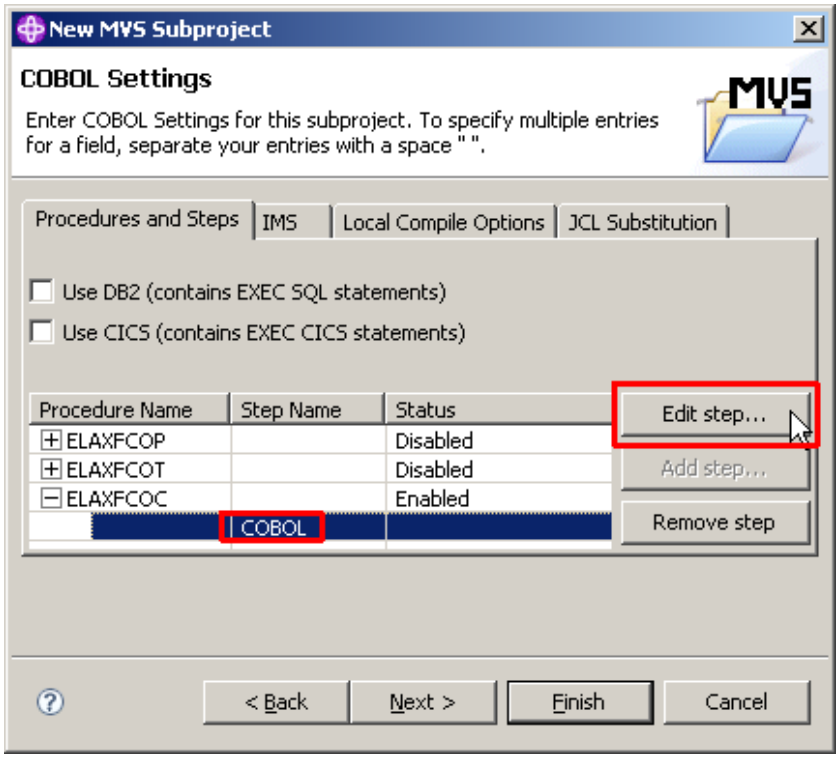


Abbildung 4.3.3

In dem COBOL Settings Panel, expandiere die ELAXFCOC Prozedur. Selektiere COBOL, 1k auf Edit Step. ELAXFCOC ist der Name einer z/OS Prozedur, welche das Kompilieren des Programms bewirkt.

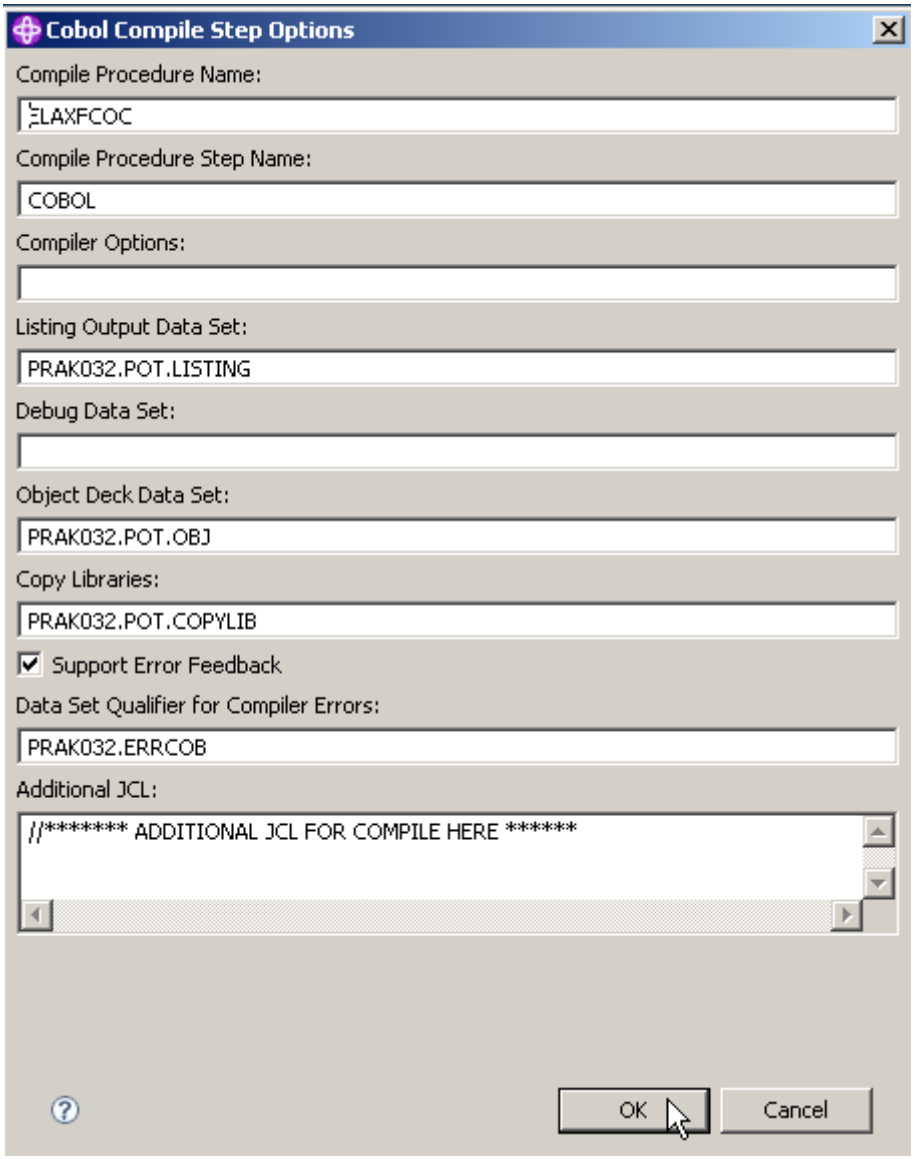


Abbildung 4.3.4

Schauen sie sich die Einträge an. Beachten Sie, die Werte wurden "inherited" von den frodo System Settings in Abschnitt 2.1.. Benutzen sie einen Text Editor, um sich die „MVS_Files_properties.xml“ File anzusehen. Sie finden dort ähnliche Werte.

Beachten Sie auch, der <HLQ> wurde automatisch auf Ihre User ID geändert (PRAK032 in diesem Beispiel). 1k auf OK.

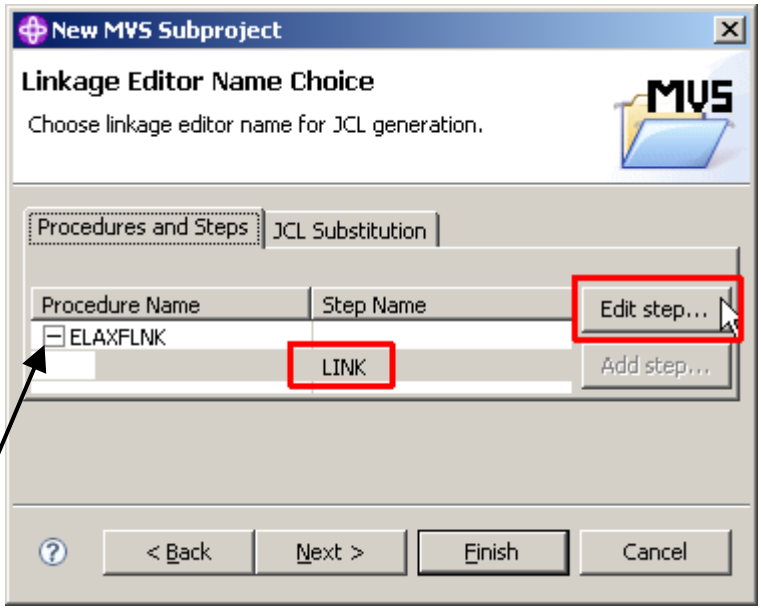


Abbildung 4.3.5

Mehrfach auf Next klicken, bis Sie zu dem „Linkage Editor Name Choice“ Panel kommen (etwa 6 mal).

Die ELAXFLNK Prozedur expandieren.

Selektiere LINK und click Edit Step.

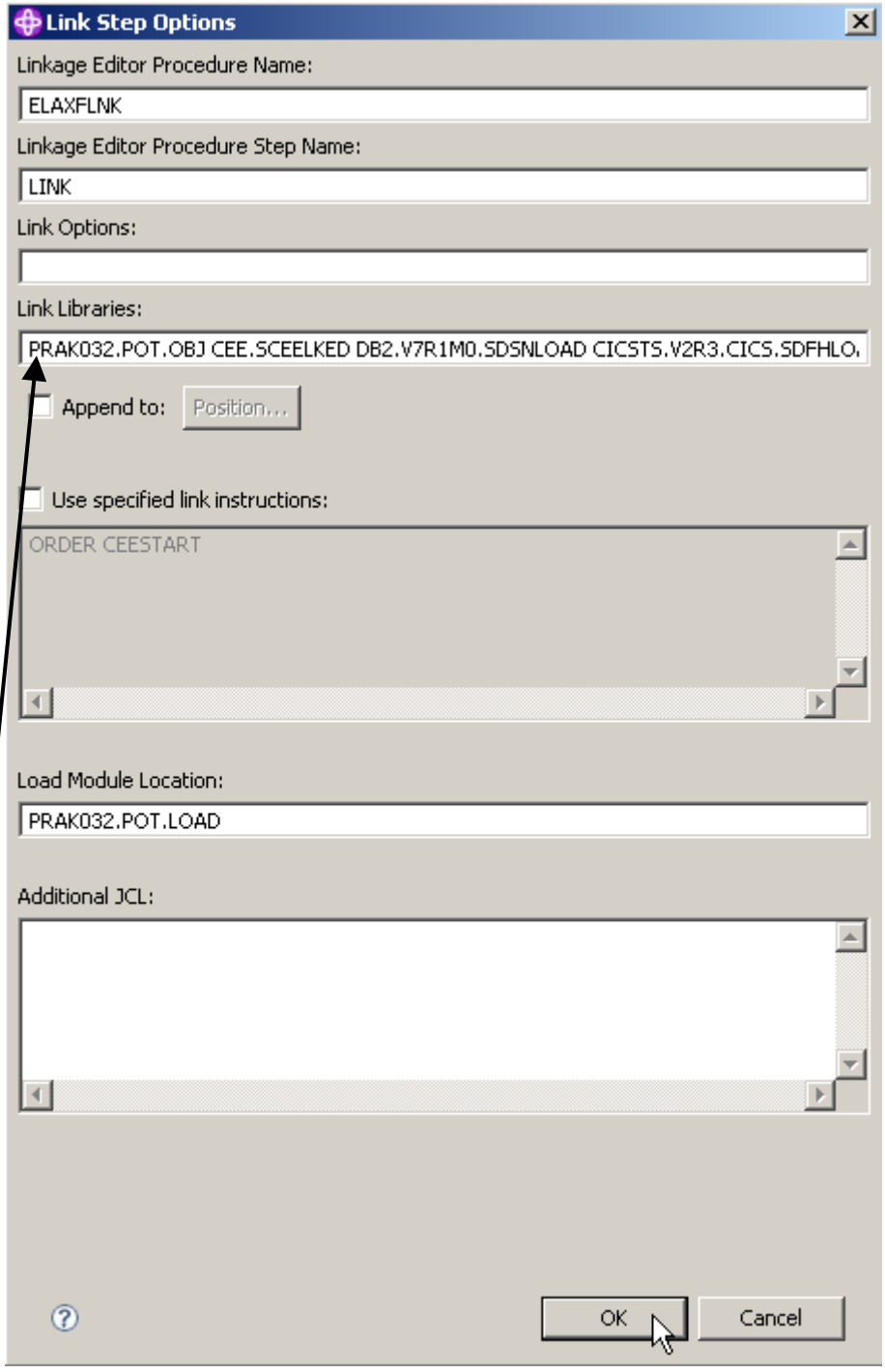


Abbildung 4.3.6

Das Link Step Options Panel erscheint. In dem Link Libraries Field sehen sie diesen Eintrag:

PRAK032.POT.OBJ CEE.SCEELKED DB2.V7R1M0.SDSNLOAD CICSTS.V2R3.CICS.SDFHLOAD

Ein potentielles Problem ist, dass nicht alle dargestellten User IDs gegen Ihre eigene User ID ausgetauscht werden. Bitte alle Änderungen sorgfältig überprüfen.

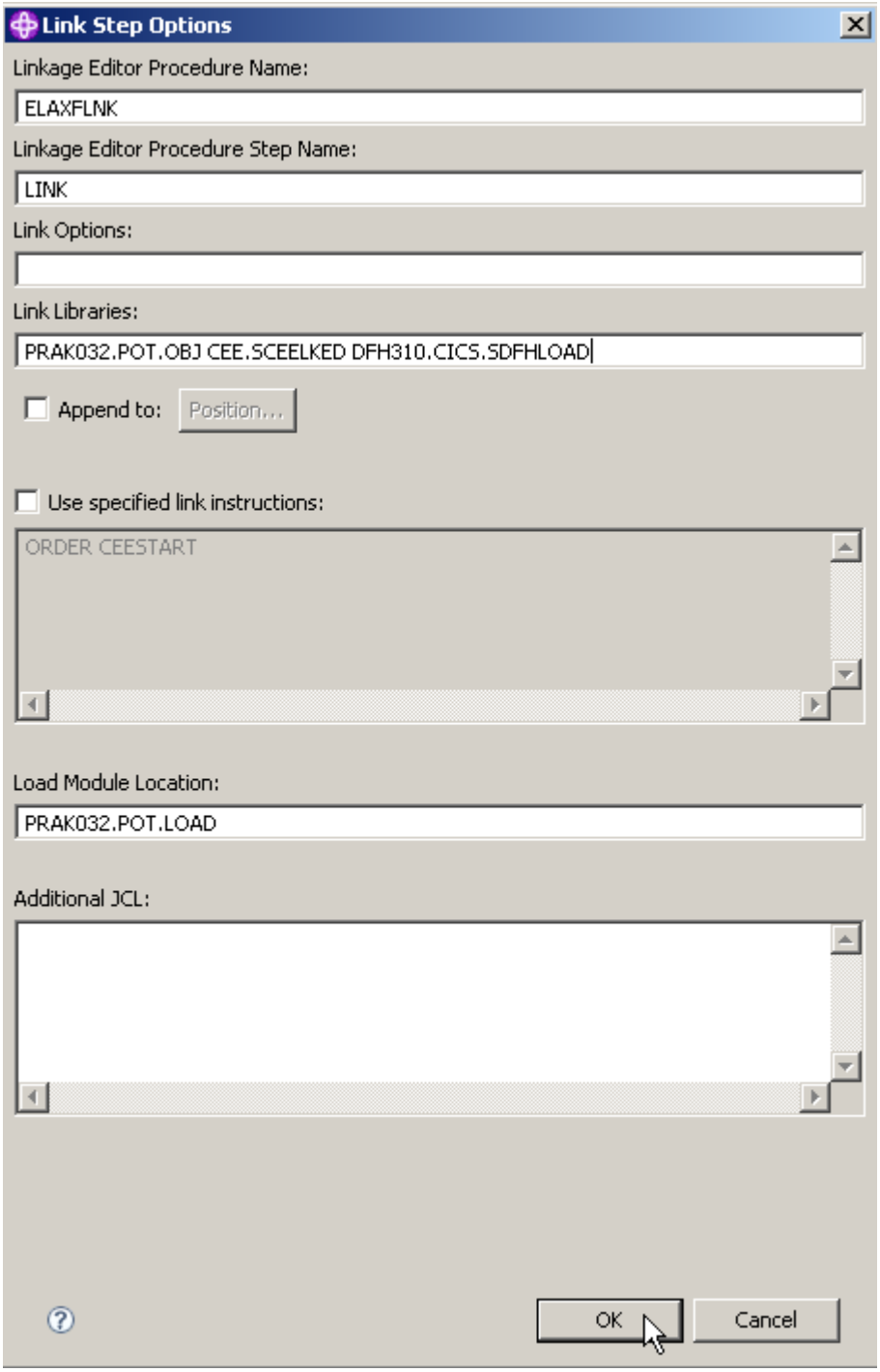


Abbildung 4.3.7

Den Link Libraries Eintrag ändern auf

PRAK032.POT.OBJ CEE.SCEELKED DFH310.CICS.SDFHLOAD

space

Die Link Libraries enthalten mehr als einen Data Set. To concatenate datasets, just add as many as you want with a space between. “CEE.SCEELKED“ und „DFH310.CICS.SDFHLOAD“ sind die Namen von zwei z/OS System Libraries, die für das Linking benötigt werden.

Click OK.

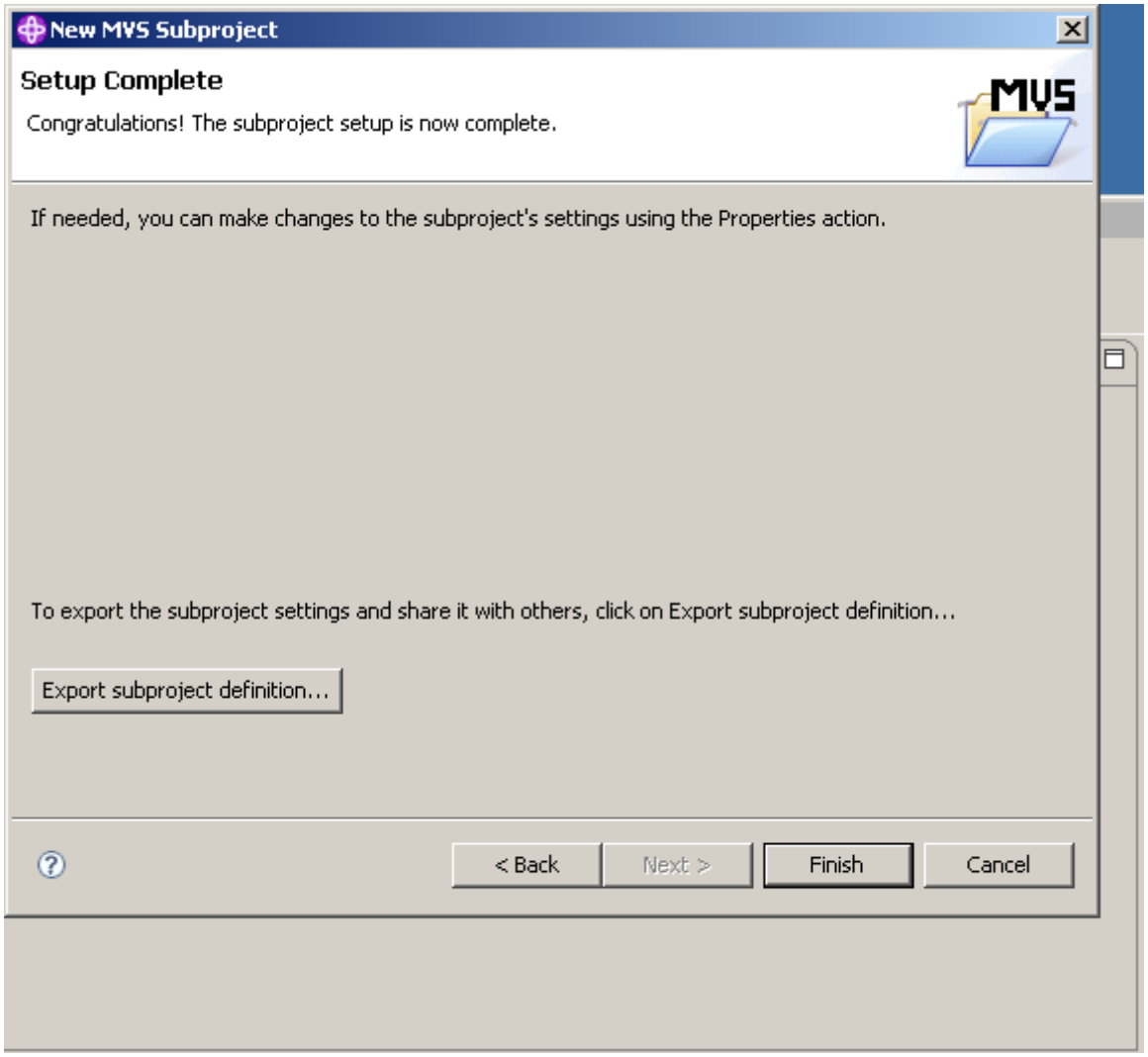


Abbildung 4.3.8

Mehrfaches Klick auf Next (etwa 3 mal) bis Sie zu dem „Setup Complete“ Panel kommen.

Anmerkung: Für zukünftigen Gebrauch können sie die Subproject Definitionen an dieser Stelle in eine xml-file exportieren.

1k auf Finish um das Fenster zu schließen.

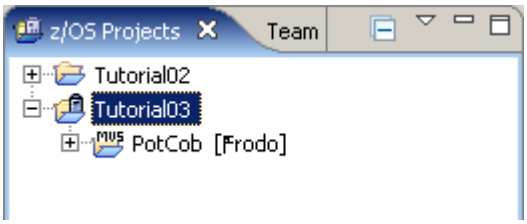


Abbildung 4.3.9

Sie sollten das neu erstellte MVS Project PotCob in Ihrem z/OS Projects View sehen. Wenn sie auf das + Zeichen klicken um das Projekt zu expandieren, werden sie finden, dass PotCob leer ist.

Ihr Workspace C:\Dokumente und Einstellungen\LEIPZIG\IBM\rationalsdp7.0\workspace

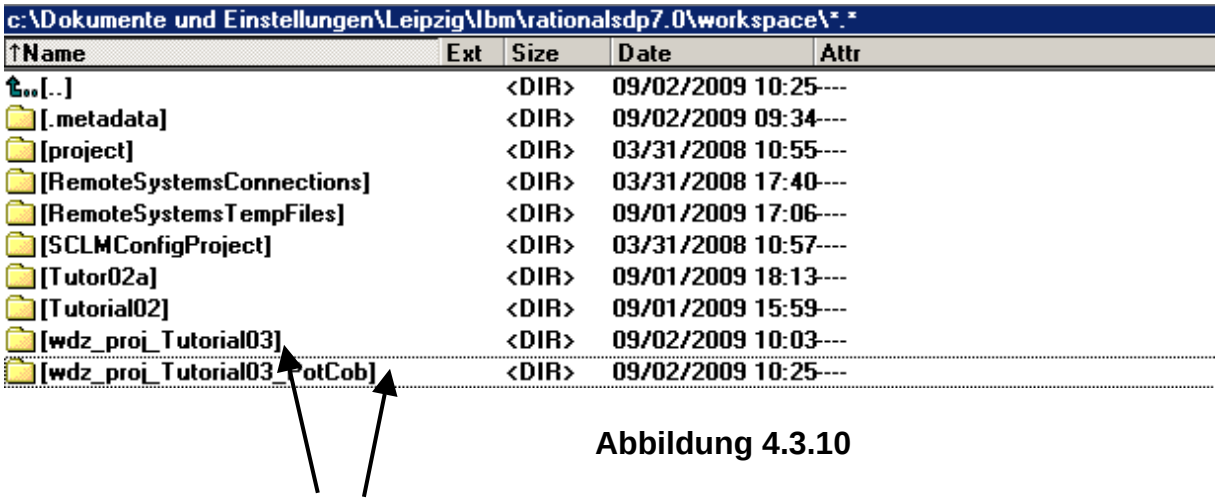


Abbildung 4.3.10

hat jetzt zwei neue Einträge. Ihr Inhalt sieht so aus:

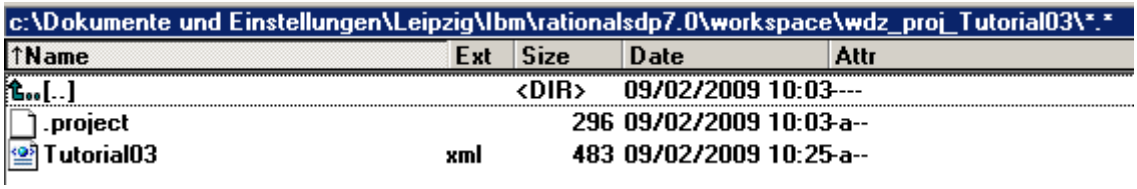


Abbildung 4.3.11

und:

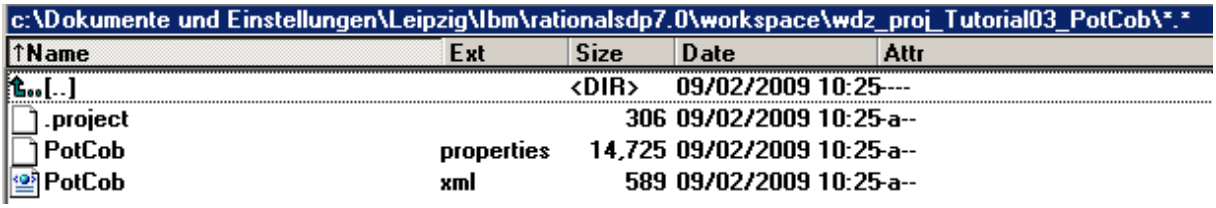


Abbildung 4.3.12

Anmerkung: Wenn Sie in den folgenden Schritten Probleme haben sollten, könnten Sie die gerade erstellte Konfiguration von der xml-file PotCob_properties.xml importieren. Sie finden diese auf der Workstation unter C:\Tutorials\Tutorial03\Resource.

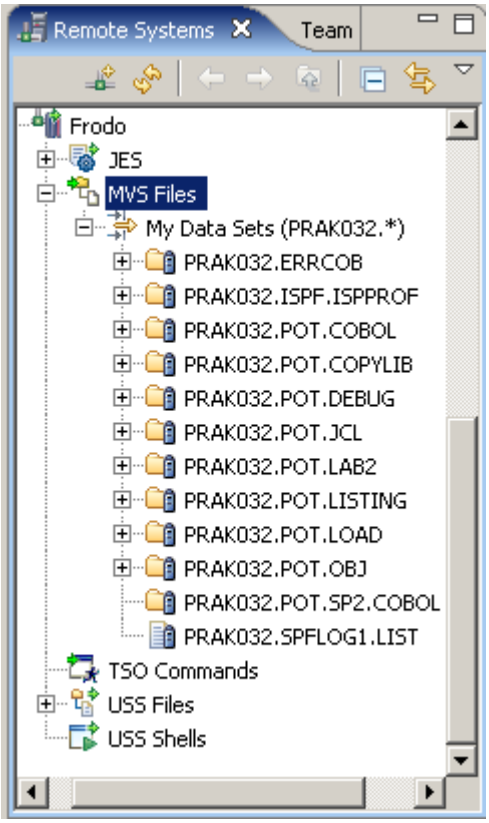


Abbildung 4.3.13

Remote Systems sollte jetzt so aussehen:

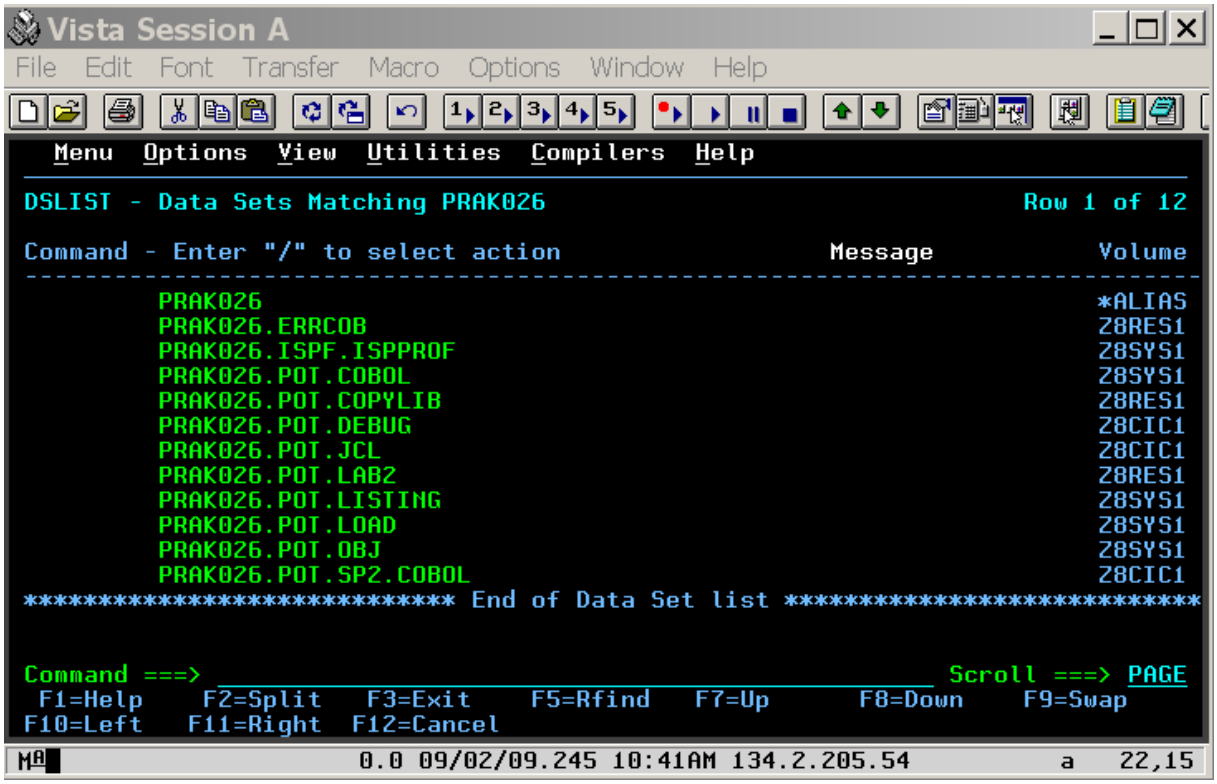


Abbildung 4.3.14

Wenn Sie einen 3270 Emulator benutzen, würde das ISPF DLIST Panel jetzt so aussehen.

4.4 Ressourcen dem Projekt hinzufügen

Wir installieren jetzt einige Data Sets unter Ihrer z/OS User ID. Hierzu fügen wir diese Ihrem Remote Projekt hinzu.

Wechseln sie zum Remote Systems View. Expandieren Sie die MVS Files unter Frodo bis Sie die Datasets sehen, welche Sie vorher allocated haben.

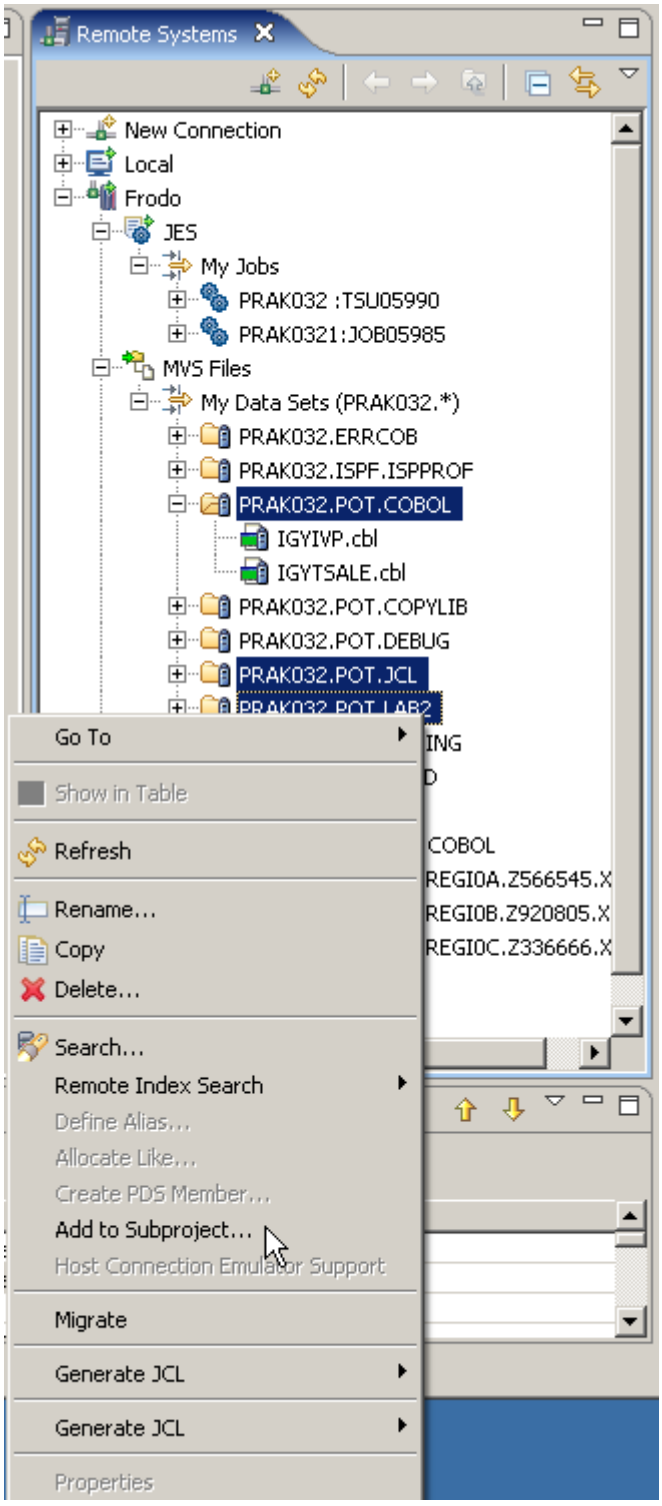


Abbildung 4.4.1

Den CTRL Key drücken (-> multiple select), markieren Sie die drei Data Sets PRAK032.POT.COBOL, PRAK032.POT.JCL und PRAK032.POT.LAB2. 1kr auf die Gruppe, selektiere „Add to Subproject".

(Ersetzen Sie 032 mit Ihrer User ID)

Anmerkung: Sie könnten das Gleiche mit Drag and Drop erreichen..

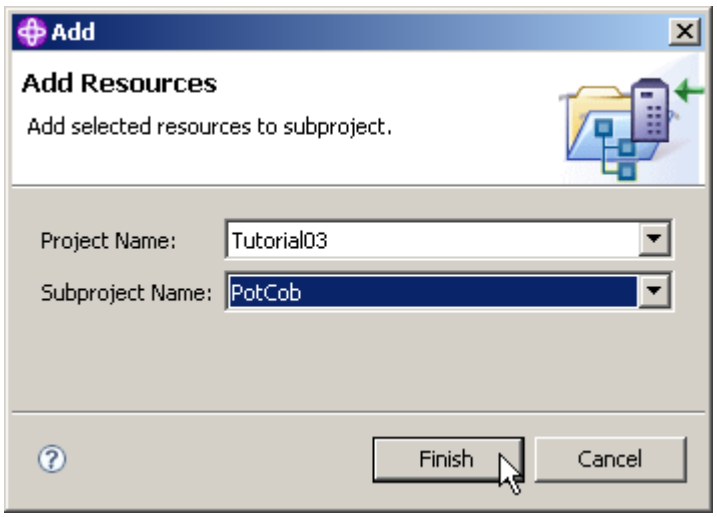


Abbildung 4.4.2

Akzeptieren Sie den Projekt Namen Tutorial03 und den Subprojekt Namen PotCob. 1k um die Data Sets hinzuzufügen.

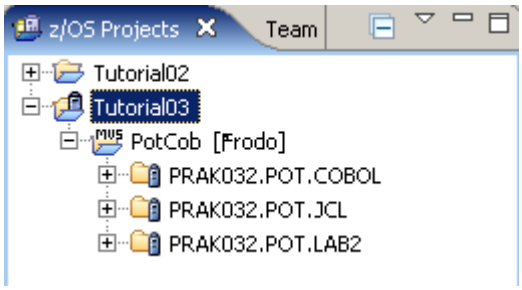


Abbildung 4.4.3

Wechseln Sie zum z/OS Projects View. Sie werden sehen, dass die drei Data Sets dem PotCob Projekt hinzugefügt wurden. Der z/OS Projects View sollte wie hier gezeigt aussehen.

Denken Sie daran: Das Arbeiten mit RDz erfolgt traditionell mit Projekten. Für z/S bedeutet dies, dass Ihre Assets (besonders z/OS-Datensätze) in eine Projektstruktur zu verknüpfen sind: in ein z/OS-Project.

Diese Projekte können entweder Unix System Services (USS) oder MVS Subprojekte sein, je nachdem, ob Sie mit Dateien eines USS hierarchischen Filesystems oder mit PDS-Membern in MVS arbeiten. In beiden Fällen bleiben Ihre Assets auf dem Mainframe-Host und werden nur logisch mit den Subprojekten verknüpft.

Selbst-Test

- **Befinden sich jetzt Kopien der in dem Data Set PRAK032.POT.LAB2 enthaltenen drei Member Regio0A, Regio0B und Regio0C in ihrem Workspace ?**
- **Von welchen z/OS Data Sets befinden sich jetzt ebenfalls Kopien in Ihrem Workspace ?**

4.5 Compile/Link/Execute des entfernten Cobol Programms.

Wir werden jetzt Ihre Programme unter z/OS kompilieren, verknüpfen (link) und ausführen. Sie verwenden hierfür eine Funktion „JCL Generation“. Wir benutzen die Properties des Remote z/OS-Projektes, um JCL nur für Compile, Compile und Link oder Compile, Link und Go zu generieren.

Wir verwenden diese Funktion nur zu Demonstrationszwecken. In der realen Welt würden Sie vermutlich Compile Verfahren einer bestehende Mainframe-Installation verwenden, oder ein Quellcode-Management-System wie SCLM (ISPF Software Configuration and Library Manager) oder den Endeavor Software Change Manager von Computer Associates benutzen (http://www.ca.com/us/products/detail/ca-endevor-software-change-manager.aspx).

In diesem Beispiel brauchen wir eine kompilierte Version von REGI0B.cbl, bevor wir mit der Kompilation von REGI0A.cbl fortfahren können. Wir fangen daher mit der Kompilation von REGI0B.cbl an.

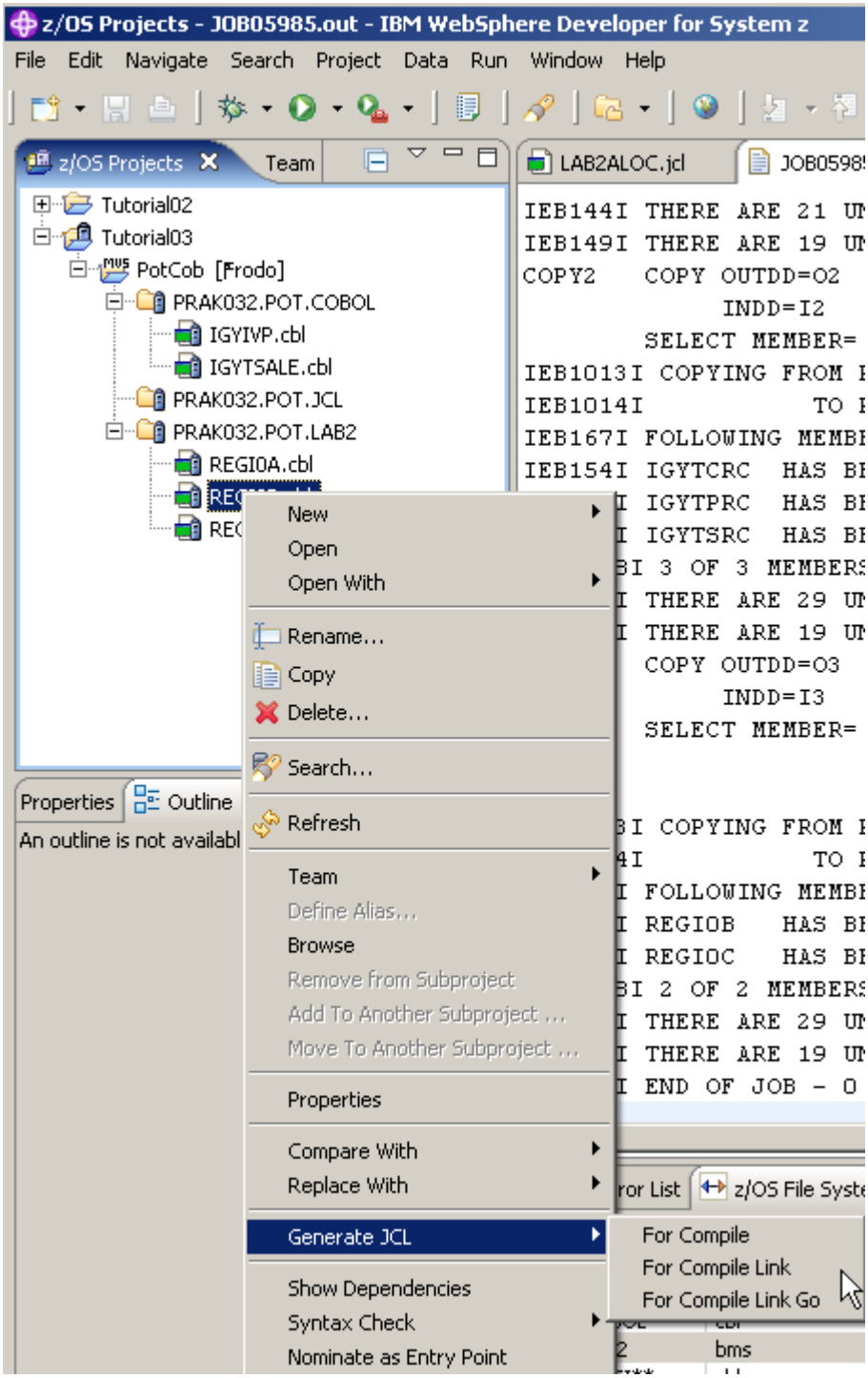


Abbildung 4.5.1

Im z/OS Projects View, expandiere PotCob und PRAK032.POT:LAB. 1kr auf REGI0B.cbl. Selektiere Generate JCL → For Compile Link. 1k

Bitte beachten, sowohl in dem z/OS Projects View als auch in dem Remote Systems View ist PRAK032.POT.JCL leer (enthält keine Member).

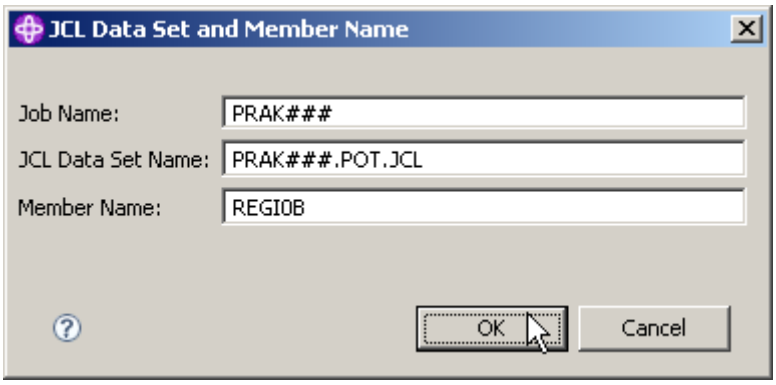


Abbildung 4.5.2

Im „JCL Data Set and Member Name“ Window hat der JCL Data Set Name den Wert, den Sie in Ihren Project Settings spezifizierten.

muss durch Ihre User ID ersetzt werden. Click OK.

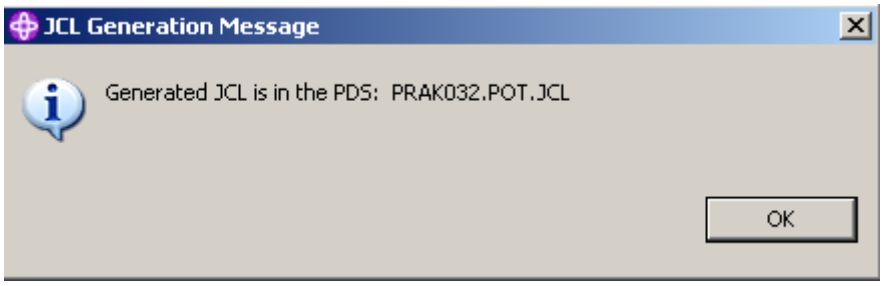


Abbildung 4.5.3

Sie sollten diese Nachricht sehen (Hier für die UserID PRAK032). Nochmals click OK.

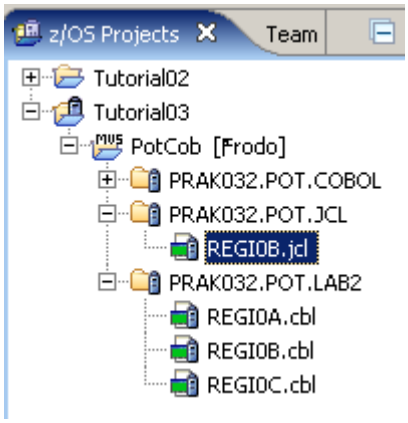


Abbildung 4.5.4

Gehen Sie zum z/OS Projects View. Sie werden sehen, dass REGI0B.jcl als Member von PRAK032.POT.JCL generiert wurde.

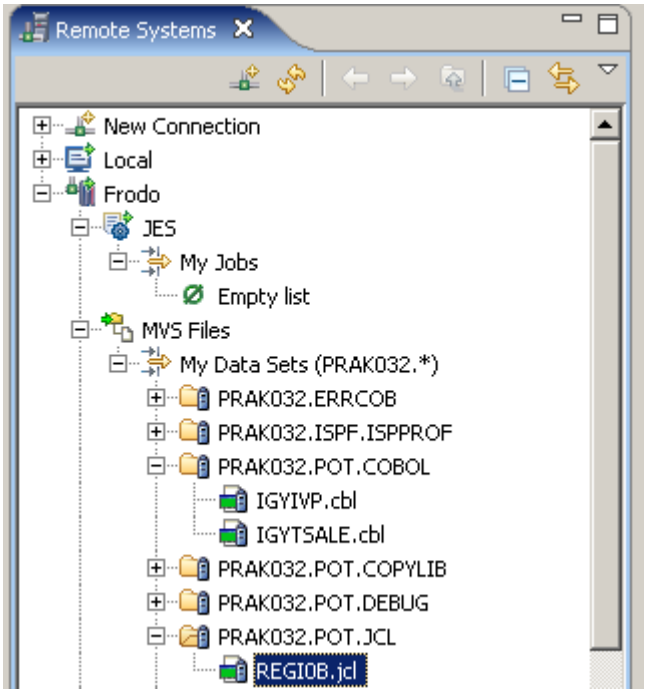


Abbildung 4.5.5

Beachten Sie, im Remote Systems View enthält Frodo einen Data Set mit dem gleichen Namen.

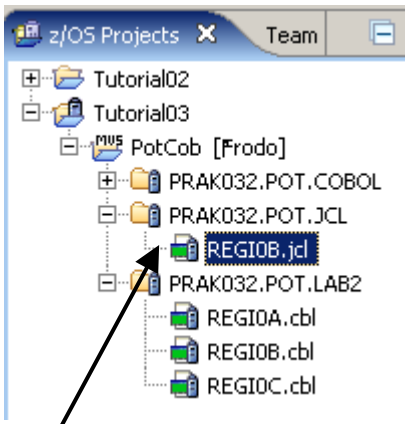


Abbildung 4.5.6

Im z/OS Projects View , 2k auf REGIO0B.jcl um die JCL File im Editor Fenster zu öffnen.

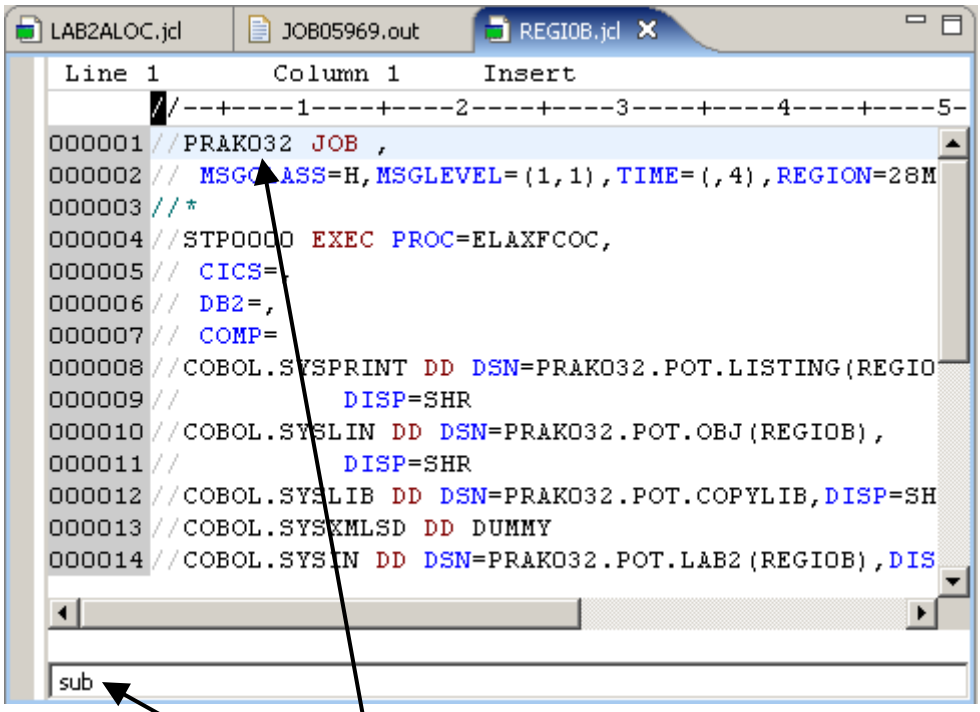


Abbildung 4.5.7

Stellen Sie sicher, der Job Name ist PRAK###, wobei ### Ihre UserID ist (PRAK032 in diesem Beispiel).

In der Command Zeile (unten im Editor Fenster), sub eingeben, und dann Enter, to submit das JCL Script. Mit der Escape Taste können Sie den Cursor auf der Command Zeile positionieren.

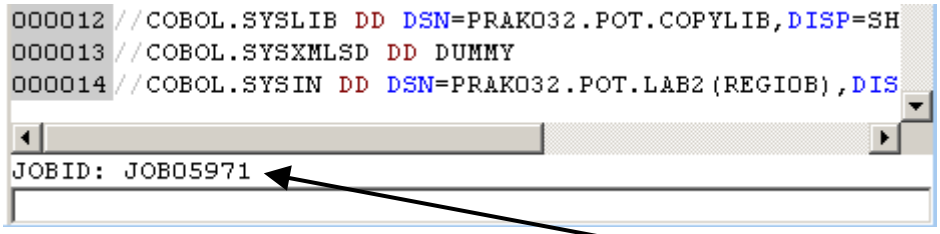


Abbildung 4.5.8

Angenommen, Sie sind mit dem Host verbunden. Sie erhalten eine Bestätigung mit der JOB ID.

Mit dieser JOB ID können nun Sie nach dem Job im JES Subsystem in dem Remote Systems Window schauen. Überprüfen Sie die erfolgreich Durchführung des Jobs. Die JOBID sollte unter “ Frodo – JES – My Jobs” aufgeführt sein.

Sie müssen möglicherweise ein Disconnect und Reconnect für Frodo ausführen, um es zu sehen.

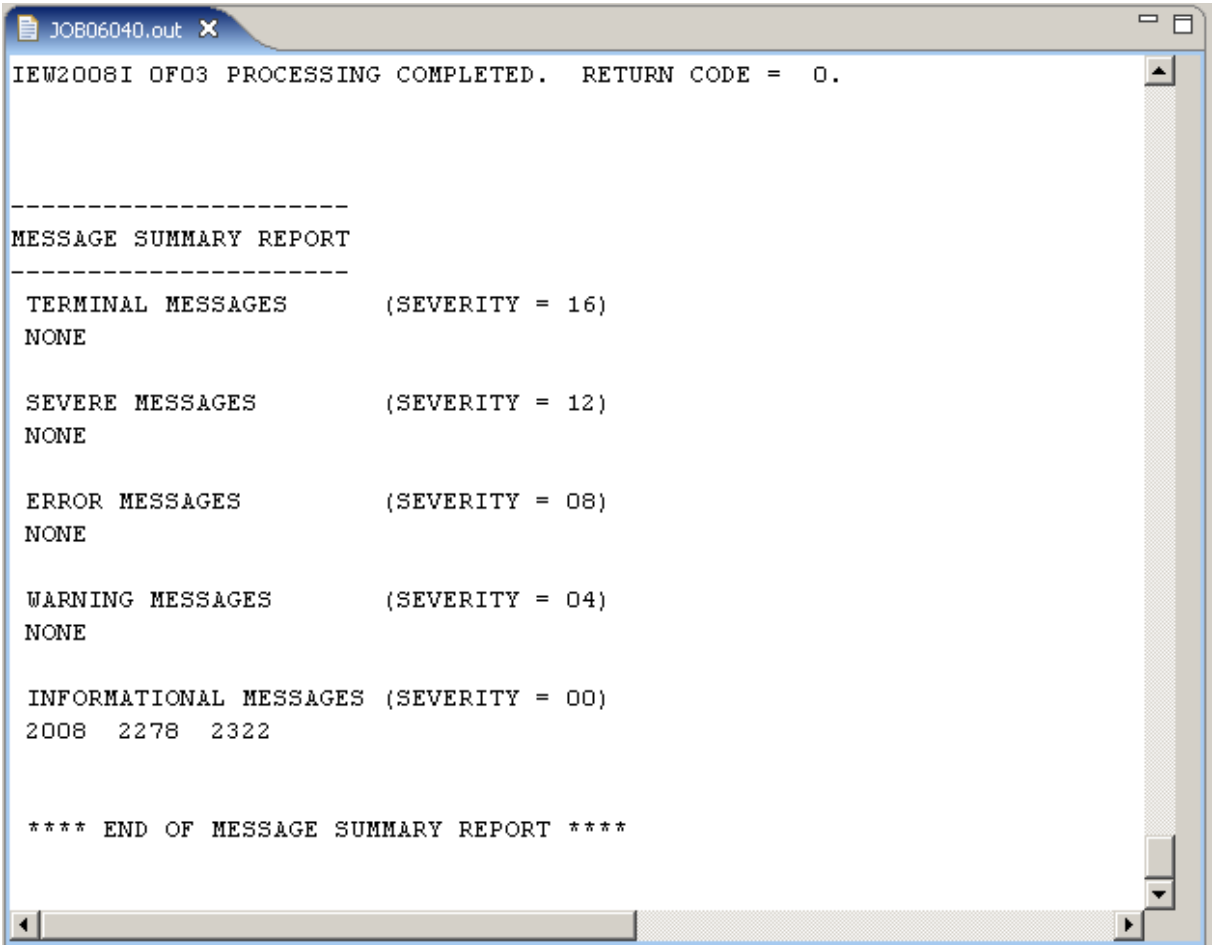


Abbildung 4.5.9

2k auf die JOB ID in dem Remote Systems View um es in dem Editor Window zu öffnen. Scroll down zum Ende. Sie sollten etwas ähnliches sehen wie in Abb. 4.5.9 dargestellt.

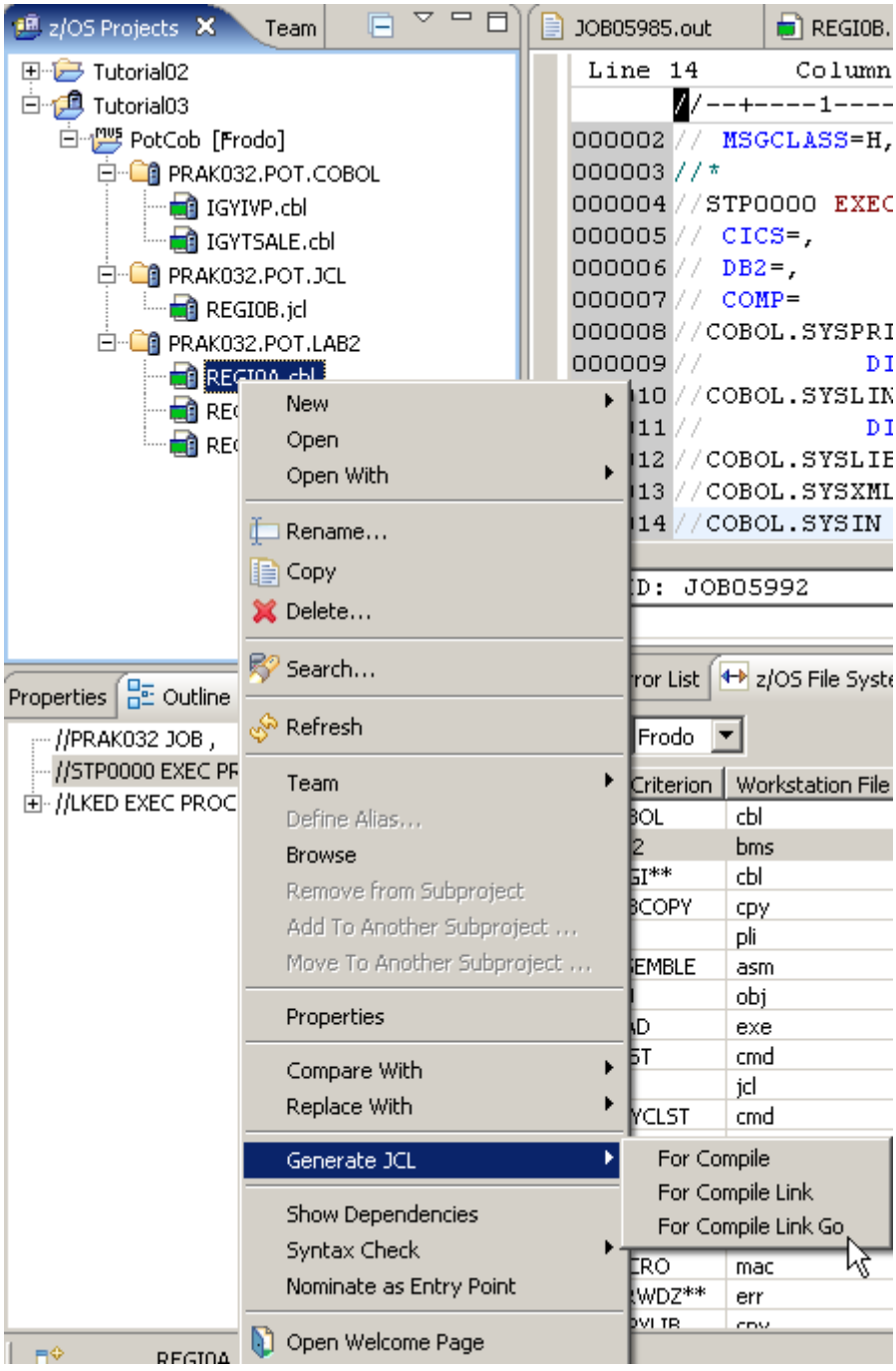


Abbildung 4.5.10

1kr auf REGI0A.cbl. Selektiere Generate JCL → For Compile Link Go.

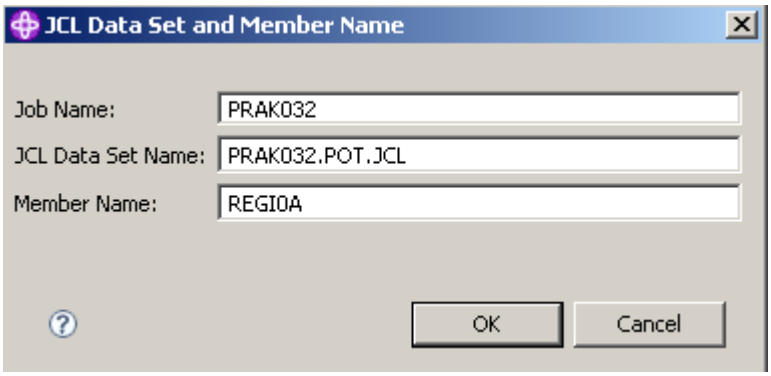


Abbildung 4.5.11

Verifizieren Sie, dass alles richtig ist. Ersetzen Sie PRAK032 durch Ihre User ID. Click OK.

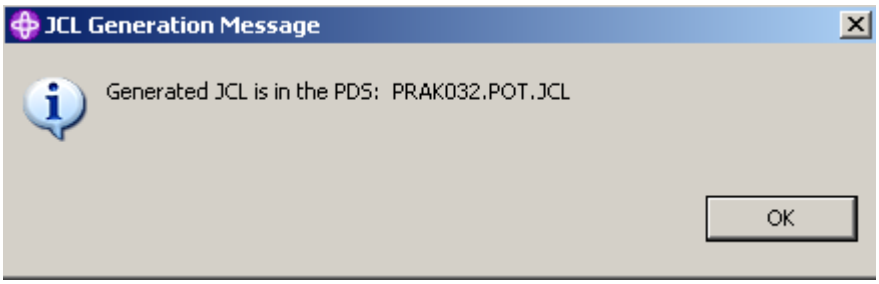


Abbildung 4.5.12

Click OK.

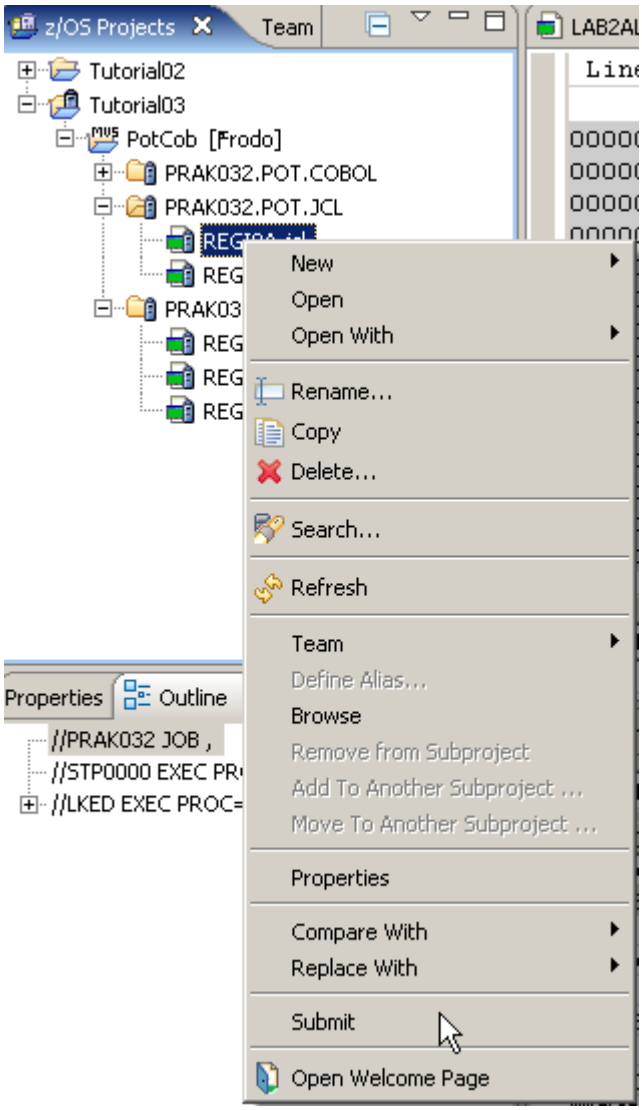


Abbildung 4.5.13

**Die neu erstellte File REGI0A.jcl befindet sich in PRAK032.POT.JCL.
1kr; selektiere Submit.**

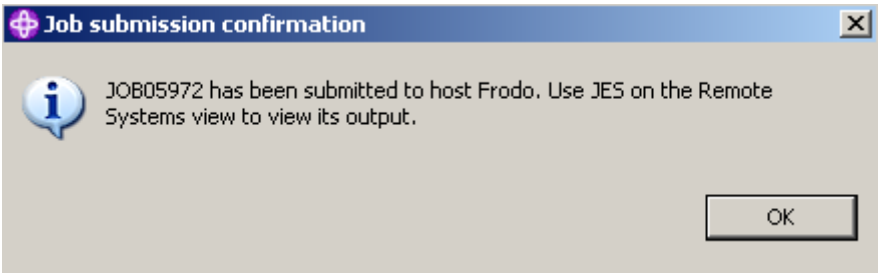


Abbildung 4.5.14

Click OK.

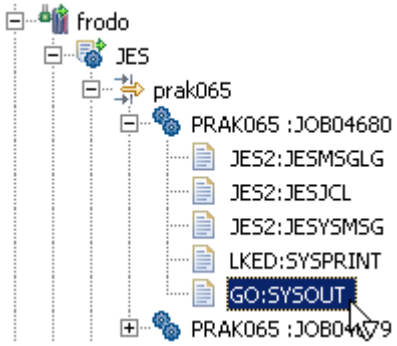


Abbildung 4.5.15

Um den Output Ihres Programms zu sehen, wechseln sie zum Remote Systems Window. Expandiere JES und My Jobs.

Schauen Sie nach der JOB ID Ihres submitted Jobs. Expandieren Sie, ….

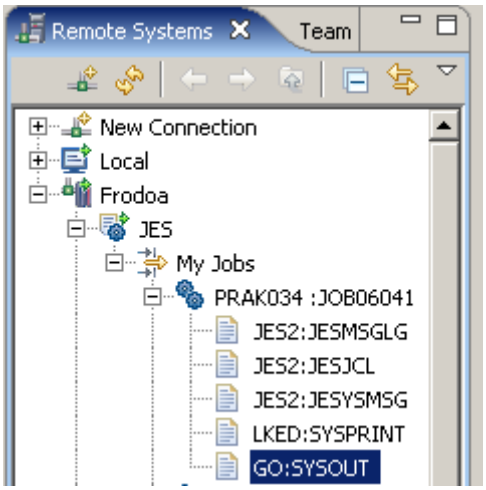


Abbildung 4.5.16

und 2k auf GO:SYSOUT. Sie müssen möglicherweise ein Disconnect und Reconnect für Frodo ausführen, um es zu sehen.

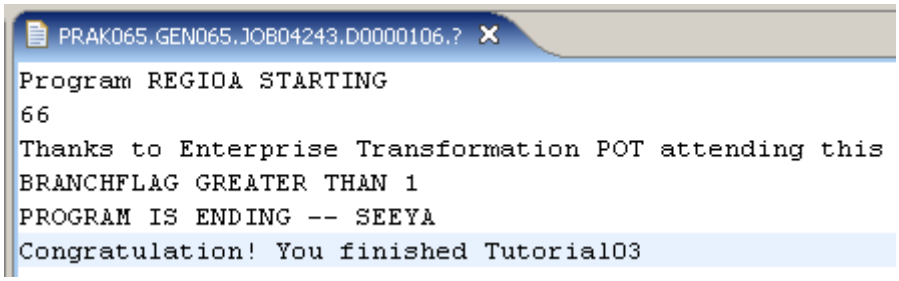


Abbildung 4.5.17

Falls Sie alles richtig gemacht haben, sollten Sie diesen Output im Editor Fenster sehen.

Selbst-Test

- **Schauen Sie sich den Quelltext der drei Programme REHIO0A, REGIO0B und REGIO0C an. Warum wurde REGIO0B als erstes Kompiliert ?**
- **Was ist mit der Kompilierung von REGIO0C ?**

Herzlichen Glückwunsch,
Sie haben RDz Tutorial03 erfolgreich abgeschlossen !

Anhang

Anhang

Für die Durchführung des Tutorials werden die folgenden Resourcen benötigt:

- **LAB2ALOC.jcl**
- **MVS_Files_properties.xml**
- **PotCob_properties.xml**
- **REGI0A.cbl**
- **REGI0B.cbl**
- **REGI0C.cbl**

Die Datei Cobresource.zip enthält diese Resources als zip Archiv. Sie kann heruntergeladen werden unter

www.cedix.de/Vorles/Band3/Resources/Cobresource.zip

Erstellen und Benutzen von VSAM-Datasets

Dies ist ein einführendes Tutorial zur Benutzung von VSAM-Datasets.
Dieses Tutorial behandelt „Key Sequential Datasets“ (KSDS).

Das Tutorial findet komplett unter TSO statt. Es werden Erfahrungen im Umgang mit ISPF (Umgang mit Datasets), dem ISPF-Editor und SDSF vorausgesetzt.

Als Beispiel dient folgendes:

Es sollen für die Universität Matrikelnummern, Studserv-Kennung sowie Vor- und Nachname in einem VSAM Dataset gespeichert werden. Dabei soll in einem Programm einmal über die Matrikel-Nummer und einmal über das Studserv-Kennung ein Zugriff auf die restlichen Daten erfolgen.

Für diesen Zweck soll zuerst ein VSAM-Dataset angelegt werden, der die Daten beinhaltet und einen Index auf die Matrikelnummer anlegt. Dieser wird danach mit den Daten der Studenten gefüllt werden.

Um auf die Daten zuzugreifen, ist ein COBOL-Programm zu schreiben. Das Programm soll über einen JCL-Script gestartet werden und dann als Batch-Job laufen.

Anschließend wird der VSAM-Dataset um einen alternativen Index – die Studserv-Kennung – erweitert und ein weiteres COBOL-Programm geschrieben (bzw. das vorhandene angepasst), das über diesen Index die Daten sucht.

Das Tutorial besteht aus den folgenden drei Teilen:

1. **Erstellen des VSAM-Datasets**
2. **Schreiben des COBOL-Programms**
3. **Erweitern um alternativen Index**

Inhalt

Für die Durchführung des Tutorials erstellen wir eine ganze Reihe von Datasets und deren Members.Dies sind spezifisch:

PRAKT20.VSAM.STUDENT	**VSAM Cluster**
PRAKT20.VSAM.STUDENT.DATA	**Data Component des VSAM Cluster**
PRAKT20.VSAM.STUDENT.INDEX	**Index Component des VSAM Cluster**
PRAKT20.VSAM.STUDENT.ALTINDEX	**VSAM Cluster**
PRAKT20.VSAM.STUDENT. ALTINDEX.DATA	**Data Component des VSAM Cluster**
PRAKT20.VSAM.STUDENT.ALTINDEX.INDEX	**Index Component des VSAM Cluster**
PRAKT20.VSAM.STUDENT.PATH	**Pfad zum VSAM Cluster**
PRAKT20. VSAM.SEQDATA	**Sequentieller Dataset für Daten Eingabe**
PRAKT20. VSAM.COBOL(STUD)	**Cobol Programm # 1**
PRAKT20. VSAM.COBOL(STUD2)	**Cobol Programm # 2**
PRAKT20.VSAM.CNTL(DEFCLUST)	**JCL Script**
PRAKT20. VSAM.CNTL(REPRO)	**JCL Script**
PRAKT20. VSAM.CNTL(COMPILE)	**JCL Script**
PRAKT20. VSAM.CNTL(RUN)	**JCL Script**
PRAKT20. VSAM.CNTL(DEFAIX)	**JCL Script**
PRAKT20. VSAM.CNTL(COMP2)	**JCL Script**
PRAKT20. VSAM.CNTL(RUN2)	**JCL Script**
PRAKT20.VSAM.LOAD(STUD)	**Load Module**
PRAKT20.VSAM.LOAD(STUD2)	**Load Module**

1. Einführung

1.1 VSAM

z/OS benutzt den Begriff Access Method um einen bestimmten Dataset Typ zu bezeichnen. Unter Linux wäre der Begriff „File System“ äquivalent. Genauso wie ein Linux System mit mehreren File Systemen arbeiten kann (extfs3, extfs4, ReiserFS), existieren auch unterschiedliche Access Methods unter z/OS. VSAM (Virtuell Storage Access Method) ist die wichtigste z/OS Access Method.

Der Begriff Virtual Storage Access Method (VSAM) bezeichnet sowohl den Dataset Typ (Organisation) als auch die Verfahren (access method) um unterschiedliche Benutzer Daten Typen zu managen und zu verwalten. VSAM verfügt über komplexere Funktionen als andere z/OS Access Methods und verwendet ein sehr spezifisches Vorgehen, um Daten auf dem Plattenspeicher zu speichern.

VSAM wird in erster Linie für Anwendungsdaten eingesetzt. Es wird nicht für die Speicherung von Quellcode, JCL, oder ausführbaren Programme benutzt. VSAM Datasets können nicht ohne weiteres mit ISPF editiert oder wiedergegeben werden. Spezielle VSAM Editoren sind jedoch von mehreren unabhängigen Herstellern erhältlich.

VSAM Records können auf 4 unterschiedliche Arten organisiert und benutzt werden: Entry-Sequenced, Key-Sequenced, Relative Record (direkt), oder linear. Hiervon ist der Key Sequence Dataset (KSDS) am gebräuchlichsten.

Die (logischen) Records in einer VSAM Datei können eine unterschiedliche Länge haben.

Ein neuer VSAM Dataset wird mit Hilfe einer z/OS System Utility „IDCAMS“ erstellt.

Literatur: IBM Redbook: “VSAM Demystified”. September 2003, SG24-6105-01, http://www.redbooks.ibm.com/redbooks/pdfs/sg246105.pdf

11.2 Blocking

In der Anfangszeit der Mainframe Entwicklung benutzte man „Basic“ Access Methods. Wenn im Anwendungsprogramm ein READ oder WRITE Befehl ausgeführt wurde, wurde durch das Betriebssystem ein einziger Record von der Festplatte gelesen oder auf die Festplatte geschrieben.

Sehr bald ging man zu „Queued“ Access Methods über. Bei der Ausführung eines READ Befehls wurde von der Festplatte immer eine Gruppe benachbarter Records gelesen und in einen entsprechend größeren Buffer Bereich des Hauptspeichers gelesen. Mit etwas Glück befand sich der gewünschte Record bei einem folgenden READ Befehl bereits im Hauptspeicher, und man konnte sich den aufwendigen Lesevorgang vom Plattenspeicher ersparen.

Hierzu fasste man mehrere Records (auch als „logische“ Records bezeichnet) zu einem Block (auch als „physische Records bezeichnet) zusammen. Beim Zugriff auf die Festplatte wurde immer ein Block an Stelle eines einzelnen Records gelesen.

In unserem Tutorial 1a haben wir erstmalig einen Data set allocated. Dort machten wir diese Angaben:

```
Primary quantity  . . 16              (In above units)
Secondary quantity    1               (In above units)
Directory blocks  . . 2               (Zero for sequential data set) *
Record format . . . . FB
Record length . . . . 80
Block size  . . . . . 320
Data set name type    PDS             (LIBRARY, HFS, PDS, LARGE, BASIC, *
```

Wir haben die (logische) Record Länge mit 80 Byte definiert, und haben eine Block (physischer Record) Größe von 320 Bytes festgelegt. Jeder Block nimmt also 4 logische Records auf, und bei einem Lese Zugriff auf den Festplattenspeicher werden immer 320 Bytes ausgelesen.

Für eine Diskussion über optimale Block Größtem siehe Band 1 Abschnitt 5.3.4.

1.3 VSAM Dataset Terms

VSAM Datasets werden andersartig als non-VSAM Datasets auf dem Plattenspeicher abgespeichert..

VSAM speichert Records in Blöcken, die als Control Intervals bezeichnet werden Ein Control Interval (CI) ist ein zusammenhängender (contiguous) Bereich auf einem DASD (disk drive), welcher sowohl Records als auch Steuerinformation speichert. Daten werden zwischen Hauptspeicher und Plattenspeicher als ganze Control Intervals bewegt. Die Größe der CIs kann von einem VSAM Dataset zum nächsten VSAM Dataset unterschiedlich sein; alle CIs innerhalb eines spezifischen VSAM Datasets haben jedoch die gleiche Länge. Eine sehr häufig benutzte Control Interval Größe ist 4 KByte, identisch mit der Größe eines 4 KByte Virtual Storage Page Frames.

Ein Control Interval besteht aus

- **mehreren logischen Records, plus einem**
- **Control Interval Definition Field (CIDF) sowie mehreren**
- **Record Definition Fields (RDFs).**

In so fern unterscheidet sich ein CI von den Blöcken (physischen Records) anderer Dataset Access Methods, bei denen ein Block lediglich eine Aneinanderreihung von (logischen) Records enthält.

Mehrere Control Intervals in einem VSAM Dataset werden in einem zusammenhängender (contiguous) Bereich auf einem DASD (disk drive) zusammengefasst, der als Control Area bezeichnet wird. Eine Control Area hat häufig die Größe eines Zylinders (15 Spuren) einer 3390 Festplatte, oder 849960 Bytes.

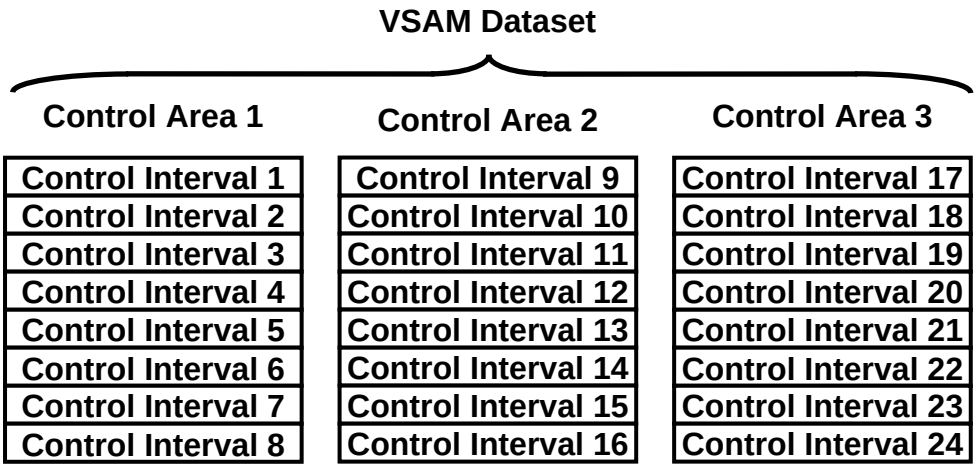


Abb.1.1 : Beispiel eines VSAM Datasets, der aus 24 Control Intervallen und 3 Control Areas besteht.

Ein VSAM Dataset kann aus vielen Control Areas bestehen. Mit einer Control Interval Größe von 4 KByte ist die maximale Größe eines VSAM Datasets von 16 TByte möglich.

1.4 Key Sequenced Dataset (KSDS)

Jeder VSAM Record besteht aus mehreren Feldern. Eines dieser Felder kann als Schlüsselfeld (Key Field) definiert werden.

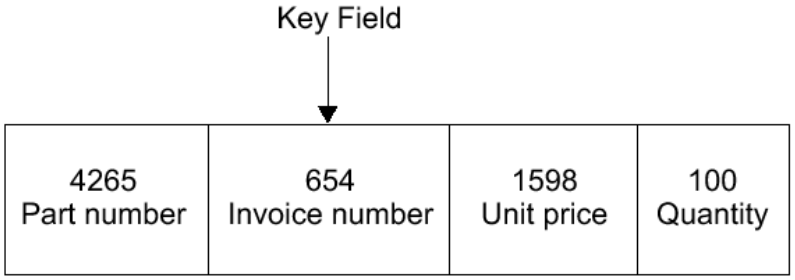


Abb. 1.2

In diesem Beispiel besteht jeder Record eines VSAM Datasets aus 4 Feldern. Die Invoice Number wird als Key Field benutzt. Für den hier gezeigten Record hat das Key Field den Wert 654.

Das Key Field muss sich in der gleichen Position in jedem Datensatz eines Key Sequenced Dataset befinden (das zweite Feld in der Abbildung oben). Jeder Datensatz Key muss eindeutig sein. Der Wert des Keys kann nachträglich nicht geändert werden; jedoch kann der gesamte Datensatz gelöscht werden.

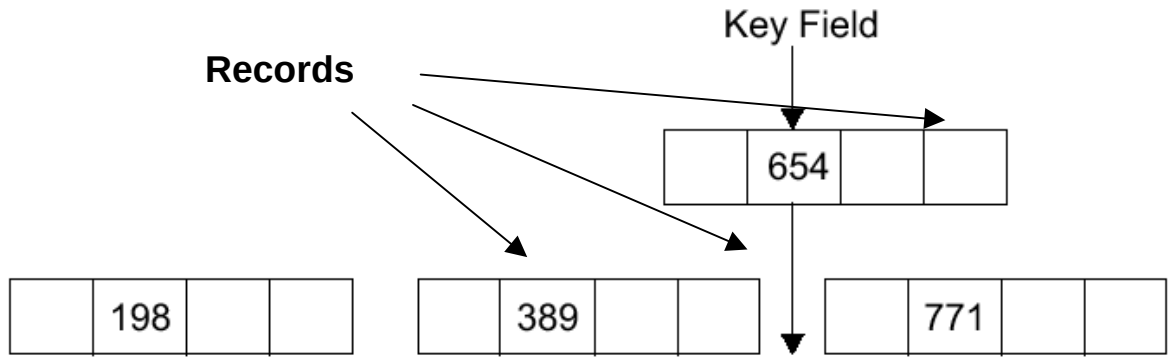
In einem Key-Sequenced Dataset sind logische Records in dem Dataset in aufsteigender Reihenfolge entsprechend der Werte in dem Key Field angeordnet. Der Key enthält einen eindeutigen Wert, z.B. eine Mitarbeiter Nummer oder eine Rechnungs Nummer. Dieser Wert bestimmt die Sortier- (collating) Position des Records in dem Dataset.

Werden neue Records erzeugt, werden diese an der richtigen Sortierstelle eingeschoben. Vorhanden Records werden verschoben um die korrekte Reihenfolge zu erhalten.

In einem KSDS können Records entsprechend ihrer Key Werte sowohl sequentiell als auch zufallsbedingt verarbeitet werden. Die Vorteile der KSDS sind:

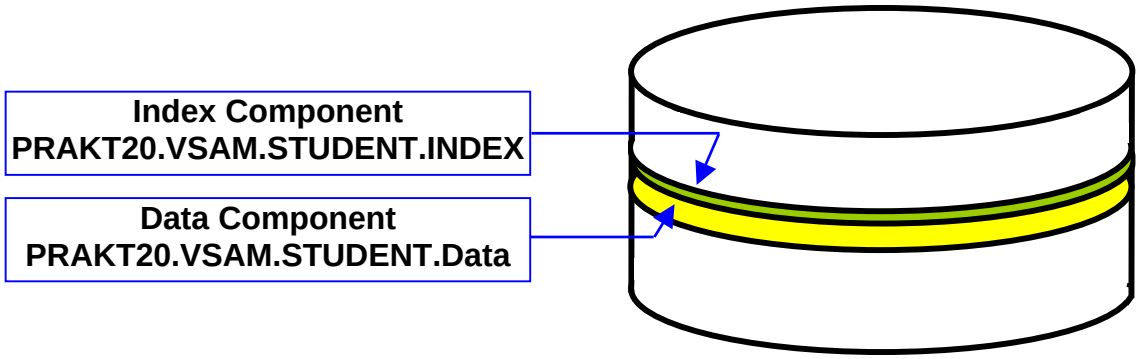
- **Eine sequentielle Verarbeitung ist für das Abrufen von Datensätzen in der sortierten Form nützlich.**
- **Eine zufällige oder direkte Verarbeitung der Rekords ist nützlich bei interaktiven Online-Anwendungen.**

Datensätze in einer KSDS werden in Key Sequence gespeichert. Das Key Field der Records bestimmt die Reihenfolge, in der die Datensätze gespeichert werden.



Wenn ein neuer Record dem Dataset hinzugefügt wird, wird es in der Sortierfolge seines Keys eingefügt. In dem gezeigten Beispiel enthält das VSAM-Dataset Control Interval drei Records mit den Key-Werten 198, 389 und 771. Ein neuer Record mit dem Key-Wert 654 wird innerhalb des CIs nach dem Record mit dem Key 389 und vor dem Record mit dem Key 771 eingefügt.

1.5 Index – und Data Component



Auf einem Plattenspeicher nimmt ein Key Sequenced Dataset (KSDS) zwei lineare Speicherbereiche ein, sogenannt Komponenten. Es gibt zwei Arten von Komponenten, die Daten-Komponente und die Index-Komponente. Die Daten Komponente ist der Teil einer VSAM-Datet, welcher die Daten Records enthält. Alle VSAM Datasets haben eine Daten-Komponente. Andere VSAM Organisationen, zum Beispiel Entry Sequenced Datasets (ESDS), haben nur die Daten-Komponente.

- **Die Datensätze (Records) der Datenkomponente werden in einem linearen Bereich auf der Plattenspeicher gespeichert, und die Index-Datensätze werden in einer zweiten linearen Raum auf dem Datenträger gespeichert.**

- **Die Index-Komponente ist eine Sammlung von Records (Index logical Records), welche**
  - **Data Keys der Records der Datenkomponente enthalten, sowie deren**
  - **Adressen (RBA, Relative Byte Address).**

- **Die Data Keys werden von dem Key-Feld in jedem Datenrecord kopiert. Mit Hilfe des Index ist VSAM der Lage, einen logischen Rekord aus dem Daten-Komponente abzurufen, wenn eine Anfrage für einen Record mit einem bestimmten Key gemacht wird.**

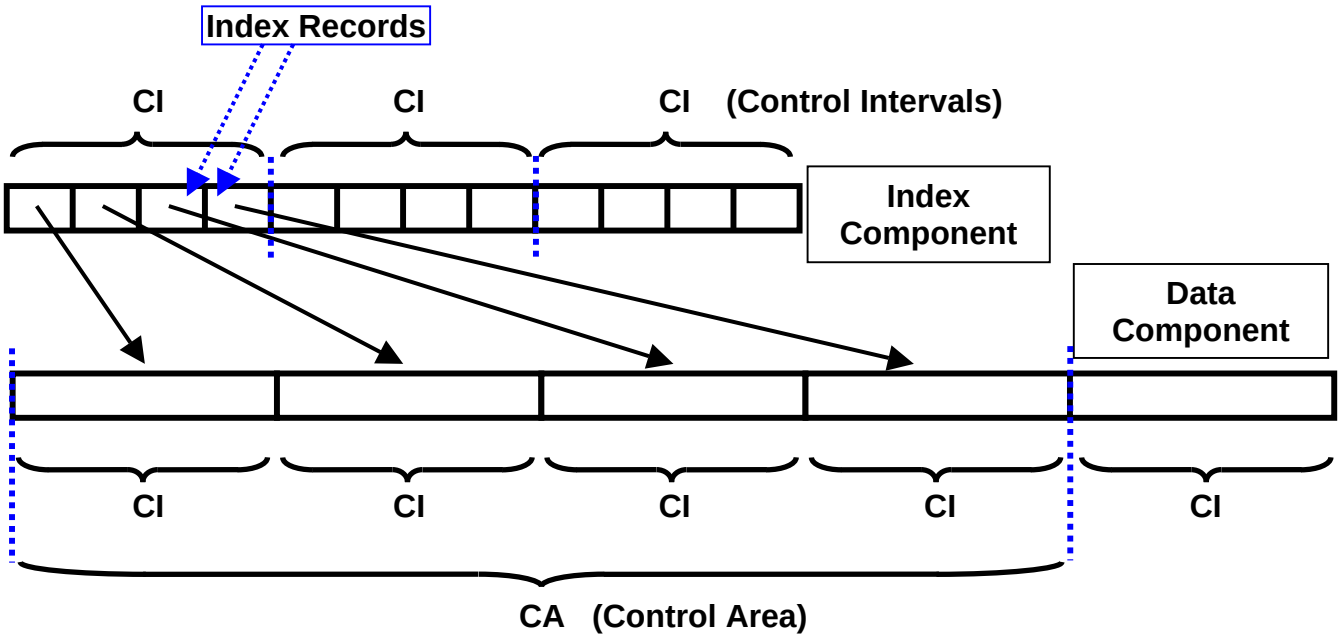
Die Datenkomponente und die Index-Komponente sind wirklich zwei unabhängige VSAM Datasets.

Ein VSAM-Cluster ist die Kombination der Daten Komponente (Dataset) und der Index-Komponente (Dataset) eines KSDS. Das Cluster Konzept vereinfacht die VSAM Verarbeitung. Es bieten eine Möglichkeit, Index und Daten-Komponenten als eine Einheit zu behandeln, und mit einem eigenen Namen zu katalogisieren. Es kann auch jede Komponente einen eigenen Namen haben. Dies ermöglicht es, die Daten Komponente getrennt von der Index-Komponente zu verarbeiten.

Jetzt kommt der entscheidende VSAM Frage: "Was in VSAM entspricht einem z/OS Dataset?" Die beste Antwort ist, es hängt von den Umständen ab. Wenn Sie zum Beispiel den Cluster-Namen in einem DD-Anweisung eines JCL-Script benutzen, dann entspricht der Cluster dem Dataset. Wenn Sie sich auf eine Komponente beziehen, dann entspricht diese dem Dataset.

(Die Bedeutung des Begriffs "Cluster", im Kontext mit VSAM, ist nicht identisch mit der Bedeutung des Begriffs "Cluster" in einer parallel Processing Configuration.)

In unserer VSAM Übungsaufgabe definieren wir einen VSAM Cluster mit dem Namen PRAKT20.VSAM.STUDENT, der aus einer Data Component PRAKT20.VSAM.STUDENT.DATA sowie einer Index Component PRAKT20.VSAM.STUDENT.INDEX besteht.



Die Index-Komponente besteht aus mehreren CIs (und wahrscheinlich auch aus mehreren CAs). Jeder CI in der Index-Komponente besteht aus mehreren Index Records, wobei jeder Index Record in der Index-Komponente auf ein CI in der Daten Komponente zeigt.

In dem oben gezeigten Beispiel enthält jedes CI in der Index-Komponente 4 Records, und jede CA in der Data Component enthält 4 CIs.

1.6 Multilevel Index Component

Ein VSAM-Index (Index Component) kann aus einer einzigen Ebene oder aus mehr als einer Ebene bestehen. Jede Stufe enthält Zeiger auf die nächst tiefere Ebene.

Die Suche in einem großen Index kann sehr zeitaufwendig sein. Ein mehrstufiger Index wird aufrechterhalten, um die Index-Suche zu verkürzen. VSAM teilt die Index Komponente in zwei Teile auf: Sequence-Set und Index-Set. Die unterste Ebene der Index Komponente wird als Sequence-Set bezeichnet. Die Pointer in der untersten Ebene zeigen direkt (mit Hilfe einer RBA) auf ein Kontroll-Intervall (CI) innerhalb einer Control Area (CA) der Daten Komponente. Der Sequence Set enthält

- **einen Index-Eintrag für jedes CI in der Daten Komponente, und damit auch**
- **ein Index CI für jedes CA in der Daten-Komponente.**

Bei kleinen VSAM Datasets würde die Index Komponente nur aus einem Sequence Set bestehen. Größere Index Sets unterhalten als Bestandteil ihrer Index Komponente eine oder mehrere Index Ebenen zusätzlich zu dem Sequence Set. Man bezeichnet die Ebenen oberhalb des Sequence Set als den Index Set. Er kann beliebig viele Ebenen enthalten. Sequence-Set und Index Set bilden zusammen die Index-Komponente eines KSDS.

Ein Eintrag in einem Index Set Datensatz enthält den höchstmöglichen Key in einem Index-Datensatz in der nächst tieferen Ebene, und einen Zeiger auf den Anfang dieses Index-Datensatzes. Die höchste Ebene des Index enthält immer einen einzelnen Index CI.

Selbst-Test

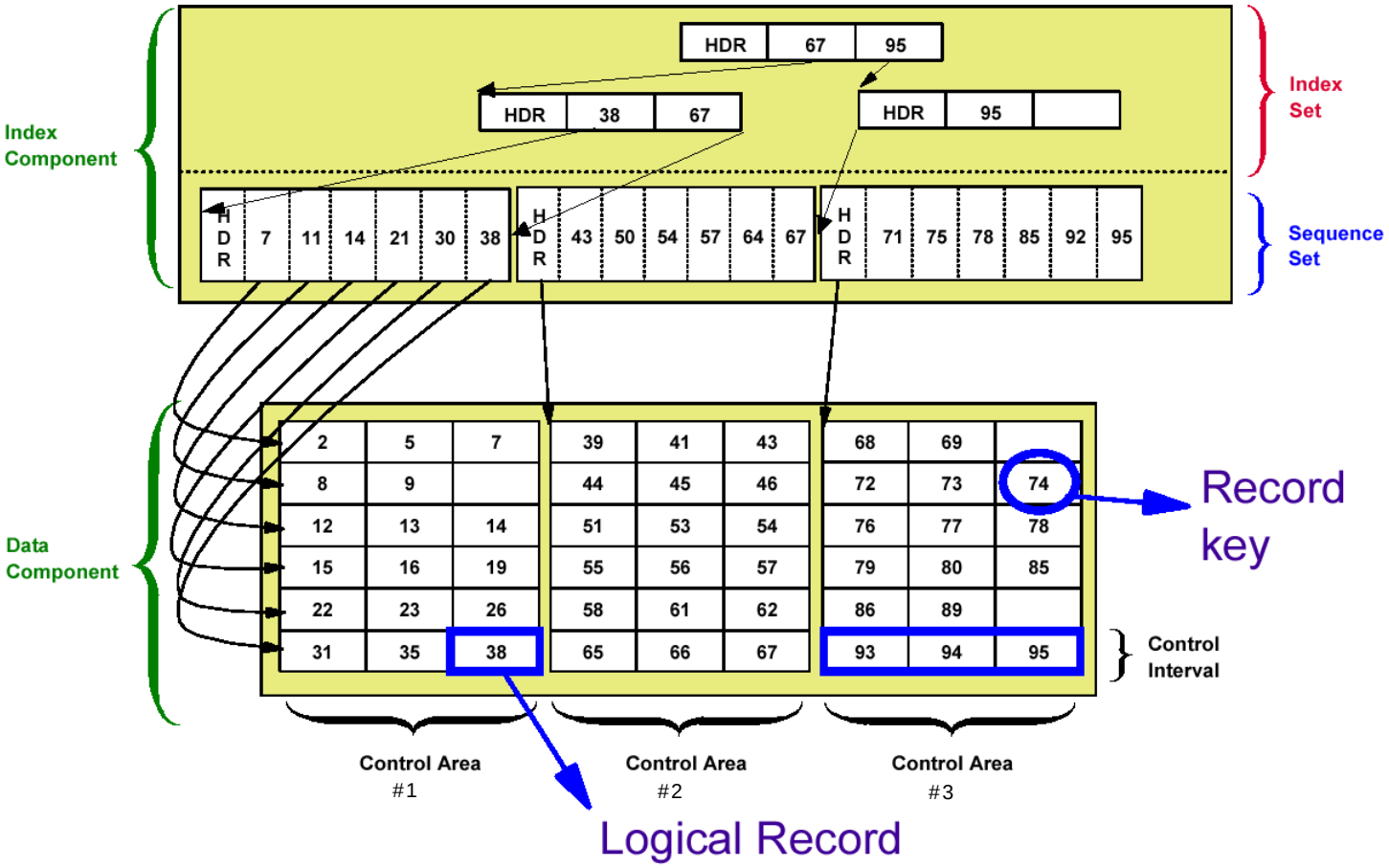
- **Sind der (die) Index Sets eigene Datasets ?**

1.7 Beispiel

Im dem unten gezeigten Beispiel, besteht die Daten Komponente eines VSAM-Key Sequenced Dataset (KSDS) aus 3 Control Areas. Jede CA besteht aus 6 Kontrollintervallen, und jedes CI speichert genau 3 Records. Es kann nützlich sein, wenn Sie die Seite mit dem Beispiel ausdrucken um den folgenden Text besser zu verstehen.

Der Sequence Set in der Index-Komponente enthält 3 CIs, eine CI für jede CA in der Daten Komponente. Jedes CI in dem Sequence Set enthält 6 Records, einen Datensatz für jedes CI in der Daten-Komponente. Jeder Sequence Set Datensatz enthält den höchsten Key in dem zugehörigen Daten-Komponente CI.

Die erste CI in der Daten-Komponente hat die Schlüssel 2, 5 und 7. Der erste Datensatz in der Sequenz-Set enthält einen Zeiger (RBA) an den Datensatz mit dem Schlüssel 7 in der Daten-Komponente.



Das zweite CI in der Daten-Komponente hat die Keys 8, 9, und einen freien Platz. Das dritte CI in der Daten-Komponente hat die Keys 12, 13 und 14. Wenn der freie Speicherplatz in dem zweiten CI in der Daten Komponente später gefüllt wird, kann ihr Key einen Wert nicht höher als 11 haben (oder es wäre an anderer Stelle platziert werden). Deshalb enthält der zweite Datensatz in dem Sequence Set einen Zeiger (RBA) auf einem potentiellen Datensatz mit dem Schlüssel 11 in der Daten-Komponente.

Für die zweite CA in der Daten-Komponente enthält der Sequence Set eine neue CI. Der erste Datensatz des CI enthält den Wert 43, weil das der Schlüssel des letzten Datensatzes in CI Nr. 1 von CA Nr. 2 in der Daten-Komponente ist.

Das letzte CI in der ersten CA der Datenkomponente hat die Keys 31, 36 und 38. Die Index Ebene unmittelbar über dem Sequenz-Set enthält einen Indexeintrag für jedes Sequence Set Controll-Intervall. Damit enthält der erste Datensatz in der ersten CI des Index Set den Wert 38. Der zweite Datensatz in dem ersten CI des Index Set enthält den Wert 67. Dies ist der höchste Key in CA 2 der Daten-Komponente.

Die unterste Ebene des Index Set enthält 2 CIs, jeweils ein CI für die beiden CAs 1 und 2 der Daten-Komponente, und ein weitere CI für CA 3 der Daten-Komponente. Diese beiden CIs werden durch eine zweite Ebene des Index Sets adressiert.

Alle Einträge werden in aufsteigender Key Reihenfolge gehalten.

Selbst-Test

- **Warum legen Sie zwei Datasets an ?**
- **Haben beide Datasets die gleiche Record Struktur ?**
- **Was bedeutet die Bezeichnung „Cluster“ ?**

2. Erstellen des VSAM-Datasets

Für das Erstellen eines VSAM-Datasets gibt es mehrere Möglichkeiten. Unter z/OS existiert ein Hilfsprogramm IDCAMS für das Anlegen von VSAM Datasets. Genauere Details der Befehle von IDCAMS kann man in „DFSMS Access Method Services for Catalogs“ (SC26-7394-03) S. 5 nachlesen.

In diesem Tutorial wird der Dataset mit einem JCL-Script unter Benutzung von IDCAMS erstellt.

Der erste Schritt bei der Generierung einer neuen Relationalen Datenbank ist die Erstellung eines Datenbank Schemas. Im Falle von VSAM muss als erster Schritt ein VSAM-Cluster angelegt werden. Weiterhin muss das (logische) Record Format definiert werden. Wir entscheiden uns für ein fixed length Record Format. Ein Record soll aus 4 Feldern bestehen:

- **Matrikelnummer 7 Zeichen (Bytes)**
- **Studserv-Kennung 8 Zeichen (Bytes)**
- **Nachname 20 Zeichen (Bytes)**
- **Vorname 20 Zeichen (Bytes)**

Damit hat ein Record 55 Zeichen. Die Daten werden in der oben angegebenen Reihenfolge im Record gespeichert.

Zum Bearbeiten der nachfolgenden Aufgaben können Sie die Datei "vsam.zip" nutzen. Diese kann unter "Datasets zum Tutorial" von der gleichen Web-Site heruntergeladen werden wie dieses Tutorial (Vorsicht: ältere Version !!!).

Bei der Anlage aller Datasets beachten Sie bitte, dass Sie für jeden Dataset nur so viel Plattenspeicher reservieren wie Sie auch benötigen. Ein oder zwei Tracks reichen in der Regel aus.

Aufgabe: *Legen Sie einen Partitioned Dataset <Ihre User-ID>.VSAM.CNTL (im FB 80-Format) an, der zukünftig alle JCL-Scripte aufnehmen soll. Beachten Sie, dass dieser eine Größe von 2 Tracks nicht überschreitet.*

Aufgabe: *Erstellen Sie im Dataset <Ihre User-ID>.VSAM.CNTL einen Member DEFCLUST, mit dessen Hilfe der VSAM-Cluster erstellt werden kann.*

Aufgabe: *Editieren Sie in den Member DEFCLUST das JCL-Script DEFCLUST (Abb. 2.1) und führen Sie es aus (sub).*

Aufgabe: *Prüfen Sie, ob der VSAM-Cluster tatsächlich angelegt wurde*

Selbst-Test

- **Aus welchen Data Sets besteht der „Cluster“ ?**

```
//PRAKT20D JOB (),NOTIFY=&SYSUID
//*
//* DEFINE VSAM CLUSTER
//*
//DEFCLS EXEC PGM=IDCAMS,REGION=4096K
//SYSPRINT DD SYSOUT=A
//SYSIN DD *

  DELETE PRAKT20.VSAM.STUDENT

  DEFINE CLUSTER ( -
         NAME(PRAKT20.VSAM.STUDENT) -
         VOL(SMS009) -
         RECORDSIZE(55 55) -
         RECORDS(10 10) -
         KEYS(7 0) -
         INDEXED ) -
       DATA ( -
         NAME(PRAKT20.VSAM.STUDENT.DATA) ) -
       INDEX ( -
         NAME(PRAKT20.VSAM.STUDENT.INDEX) )
/*
```

Das Programm IDCAMS wird aufgerufen.

Abbildung 2.1 : JCL-Script zum Erstellen des VSAM-Clusters

Die beiden Parameter bei RECORDSIZE geben die mittlere und die maximale Größe eines Records an, die in diesem Fall gleich groß sind. Wir wählen genau 55, weil ja ein Datensatz - bestehend aus Matrikel-Nr, Studserv-Kennung, Vor- und Zuname - genau 55 Zeichen lang ist.

INDEXED legt fest, dass es sich um einen KSDS handelt.
KEYS gibt an, dass der Schlüssel für den Zugriff 7 Zeichen lang ist und bei Offset 0 beginnt.

Selbst-Test

- **Was ist IDCAMS ?**
- **Benutzen Sie IDCAMS für die Erstellung von PRAKT20.VSAM.SEQDATA ?**
- **Warum arbeiten Sie mit einer Record Länge von 55 Bytes ?**

Als nächstes soll der Cluster mit Studenten-Daten gefüllt werden. Diese später in den VSAM-Cluster zu kopierenden Daten erstellen wir zunächst in einem sequentiellen Dataset.

Aufgabe: *Legen Sie diesen sequentiellen Dataset an. Er sollte 55 Bytes/Record haben (Record-Format FB) und den Namen PRAKT20.VSAM.SEQDATA erhalten. Da dieser nur sehr wenige Beispiel-Daten aufnehmen soll, reicht eine Dataset-Größe von 1 Track.*

Aufgabe: *Füllen Sie unter Nutzung des ISPF-Editors den gerade angelegten Dataset mit einigen Beispiel-Datensätzen (siehe Abbildung 2.2). Sie können einige Fantasie-Datensätze verwenden, doch einer der Datensätze muss wahrheitsgemäß Ihre persönlichen Daten enthalten: Ihr eigener Datensatz.*

```
  File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       PRAKT20.VSAM.SEQDATA                              Data set saved
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000001 1187579mai91jhzMichaelsen          Nils
000002 1207747mai02wdrMueller             Georg
000003 1207864mai03werFriedrich           Boris
000004 1213435mai03fgtHeinrich            Albrecht
000005 1309745mai03hgeFischer             Katrin
****** **************************** Bottom of Data ****************************









Command ===>                                                 Scroll ===> 0001
 F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
```

Abbildung 2.2
PRAKT20.VSAM.SEQDATA mit 5 Datensätzen füllen

Jeder Datensatz besteht aus 4 Feldern mit einer Länge von jeweis 7, 8, 20 und 20 Bytes

Selbst-Test

- **Warum schreiben Sie diese Daten in PRAKT20.VSAM.SEQDATA ?**
- **Ist PRAKT20.VSAM.SEQDATA ein VSAM Dataset ?**
- **Falls ja, warum, falls nein, warum nicht ?**

Aufgabe: *Legen Sie im Partitioned Dataset „PRAKT20.VSAM.CNTL“ den Member REPRO an, ein JCL-Script zum Kopieren der Daten aus dem sequentiellen Dataset in den VSAM-Cluster (Abbildung 2.3).*

Aufgabe: *Kopieren Sie die Studenten-Daten hinüber, indem Sie mittels SUB das JCL-Script ausführen.*

```
//PRAKT20R JOB (),NOTIFY=&SYSUID
//*
//* COPY SEQENTIAL DATASET INTO VSAM CLUSTER
//*
//DEFCLS   EXEC PGM=IDCAMS,REGION=4096K
//SEQDD    DD DSN=PRAKT20.VSAM.SEQDATA,DISP=SHR
//VSAMDD   DD DSN=PRAKT20.VSAM.STUDENT,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
  REPRO    INFILE(SEQDD) -
           OUTFILE(VSAMDD)
/*
```

Abbildung 2.3: Inhalt des Members REPRO

Selbst-Test

- **REPRO benutzt die beiden Parameter SEQDD und VSAMDD. Was bezeichnen diese ?**

3. Schreiben des COBOL-Programms

Um auf den VSAM-Dataset zuzugreifen, erstellen wir ein COBOL-Programm.

Aufgabe: *Erstellen Sie einen weiteren Partitioned Dataset <Ihre User-ID>.VSAM.COBOL, der zukünftig den Code der beiden Cobol-Programme aufnehmen soll (Record length = 80).*

Aufgabe: *Legen Sie in diesem einen Member "STUD" mit dem im folgenden angegebenen Programm-Code an.*

Es folgt der Quelltext des Cobol-Programms:

```
000100 IDENTIFICATION DIVISION.
000200*
000300* Programm zum Zugriff auf VSAM-Dataset
000400*
000500 PROGRAM-ID.  STUD.
000600/
000700 ENVIRONMENT DIVISION.
000800*--------------------
000900 CONFIGURATION SECTION.
001000 SPECIAL-NAMES.
001100 INPUT-OUTPUT SECTION.
001200 FILE-CONTROL.
001300     SELECT STUD-FILE
001400      ASSIGN TO STUDDS
001500      ORGANIZATION IS INDEXED
001600      ACCESS IS DYNAMIC
001700      RECORD KEY IS MATNR
001800      FILE STATUS IS FSTAT-CODE VSAM-CODE.
001810     SELECT MATNR-IN
001820      ASSIGN TO SYSIN
001830      ORGANIZATION IS SEQUENTIAL
001840      FILE STATUS IS MATNR-IN-CODE.
001900 DATA DIVISION.
002000*-------------
002100 FILE SECTION.
002200 FD STUD-FILE
002300     RECORD CONTAINS 55 CHARACTERS.
002400   01 STUDENT-RECORD.
002500     05 MATNR       PIC X(7).
002600     05 STUDSERV    PIC X(8).
002700     05 NNAME       PIC X(20).
002800     05 VNAME       PIC X(20).
002810 FD MATNR-IN
002820     RECORDING MODE F
002830     BLOCK 0 RECORDS
002840     RECORD 80 CHARACTERS
002850     LABEL RECORD STANDARD.
002860   01 MATNR-RECORD  PIC X(80).
002900/
003000 WORKING-STORAGE SECTION.
003100 01 STATUS-AREA.
```

```
003200   05 FSTAT-CODE PIC X(2).
003300     88 I-O-OKAY  VALUE ZEROES.
003310   05 MATNR-IN-CODE  PIC X(2).
003400   05 VSAM-CODE.
003500     10 VSAM-R15-RETURN-CODE    PIC 9(2) COMP.
003600     10 VSAM-FUNCTION-CODE      PIC 9(1) COMP.
003700     10 VSAM-FEEDBACK-CODE      PIC 9(3) COMP.
003800 01 WS-STUDENT.
003900   05 WS-MATNR     PIC X(7).
004000   05 WS-STUDSERV  PIC X(8).
004100   05 WS-NNAME     PIC X(20).
004200   05 WS-VNAME     PIC X(20).
004210 01 WS-MATNR-IN-RECORD.
004220   05 WS-MATNR-IN  PIC X(7).
004230   05 FILLER       PIC X(73).
004300/
004400 PROCEDURE DIVISION.
004500     OPEN INPUT MATNR-IN.
004600     READ MATNR-IN INTO WS-MATNR-IN-RECORD.
004700     DISPLAY "SUCHE " WS-MATNR-IN.
004800     OPEN INPUT STUD-FILE.
004900     IF FSTAT-CODE NOT = "00"
005000       DISPLAY "OPEN INPUT VSAM FILE FS-CODE: " FSTAT-CODE
005100       PERFORM VSAM-CODE-DISPLAY
005200       STOP RUN
005300     END-IF.
005310     MOVE WS-MATNR-IN TO MATNR.
005400     READ STUD-FILE RECORD KEY IS MATNR.
005500     IF FSTAT-CODE NOT = "00" AND FSTAT-CODE NOT = "02"
005600       DISPLAY "READ STUD-FILE FS-CODE: " FSTAT-CODE
005700       PERFORM VSAM-CODE-DISPLAY
005800       STOP RUN
005900     END-IF.
006000     MOVE MATNR TO WS-MATNR.
006100     MOVE STUDSERV TO WS-STUDSERV.
006200     MOVE NNAME TO WS-NNAME.
006300     MOVE VNAME TO WS-VNAME.
006310     CLOSE STUD-FILE.
006320     CLOSE MATNR-IN.
006400     DISPLAY "MATNR:    " WS-MATNR.
006500     DISPLAY "STUDSERV: " WS-STUDSERV.
006600     DISPLAY "NACHNAME: " WS-NNAME.
006700     DISPLAY "VORNAME:  " WS-VNAME.
006800     STOP RUN.
006810
006900 VSAM-CODE-DISPLAY.
007000     DISPLAY "VSAM CODE -->"
007100             " RETURN: " VSAM-R15-RETURN-CODE,
007200             " COMPONENT: " VSAM-FUNCTION-CODE,
007300             " REASON: " VSAM-FEEDBACK-CODE.
```

Abbildubg 3.1
Cobol Quelltext

Aufgabe: *Versehen sie den obigen Cobol Code großzügig mit Kommentarzeilen und senden sie das Ergebnis an Ihren Betreuer*

Aufgabe: *Erstellen Sie einen weiteren Partitioned Dataset <Ihre User-ID>.VSAM.LOAD, der zukünftig alle ausführbaren Programme aufnehmen soll (Record Format "U").*

Aufgabe: *Um das Cobol Programm zu compilieren, existiert ein Member COMPILE (Abbildung 3.2). Legen Sie auch diesen in <Ihre User-ID>.VSAM.CNTL an, modifizieren Sie ihn und wenden Sie anschließend SUB auf ihn an.*

```
//PRAKT20C JOB (),NOTIFY=&SYSUID
//*
//* COMPILIEREN DES COBOL-PROGRAMMS ZUM VSAM-ZUGRIFF
//*
//COBCOMP EXEC IGYWCL
//COBOL.SYSIN  DD DSN=PRAKT20.VSAM.COBOL(STUD),DISP=SHR
//LKED.SYSLMOD DD DSN=PRAKT20.VSAM.LOAD,DISP=SHR
//LKED.SYSIN   DD *
 NAME STUD(R)
/*
```

Abbildung 3.2: JCL-Script zum Compilieren des Cobol-Programms

Selbst-Test

- **Das JCL Script in Abb. 3.2 übersetzt das Cobol Programm und speichert den Code. Wo ?**
- **Es führt den Code aber nicht aus. Warum nicht ?**

Um das gerade compilierte Cobol-Programm ausführen zu können, ist ein weiterer Member RUN erforderlich (Abbildung 3.3).

Aufgabe: *Erstellen Sie im Dataset <Ihre User-ID>.VSAM.CNTL den Member RUN . Passen Sie ihn wieder an Ihre Bedürfnisse an.*

Aufgabe: *Schreiben Sie in den Member RUN die Matrikel-Nummer hinein, nach der im VSAM-Dataset gesucht werden soll (Abbildung 3.3, hellgrün umrahmte Zahl). Starten Sie die Programmausführung mittels SUB.*

Der „STUDDS“-DD-Entry stellt die Verbindung für das COBOL-Programm dar. In Zeile 001400 in Abb. 3.1 wird festgelegt, wo die Daten abgelegt sind.

Über SYSIN übergeben wir die Matrikelnummer 1207864 des Records, der gesucht werden soll.

```
//PRAKT20R JOB (),NOTIFY=&SYSUID
//*
//* AUSFUEHREN DES COBOL-PROGRAMMS ZUM VSAM-ZUGRIFF
//*
//RUN      EXEC PGM=STUD
//STEPLIB  DD DSN=PRAKT20.VSAM.LOAD,DISP=SHR
//STUDDS   DD DSN=PRAKT20.VSAM.STUDENT,DISP=SHR
//SYSIN    DD *
1207864
/*
```

Abbildung 3.3: Member RUN

Nach dem submit steht nun im Job Log der Eintrag des Studenten (siehe Abbildung 2.2) mit allen zu ihm vorhandenen Daten (Abbildung 3.4).

```
SUCHE 1207864
MATNR:    1207864
STUDSERV: mai03wer
NACHNAME: Friedrich
VORNAME:  Boris
```

Abbildung 3.4 : Auszug aus dem Job Log

Wie können wir uns nun die in der Abbildung 3.4 gezeigte Ausgabe unseres Cobol-Programms anschauen? Dafür starten wir SDSF und rufen den Job Status auf (ST).

Nachdem der Job gefunden wurde, schauen wir uns entweder das Job Log durch das Kommando "S" direkt an oder wir geben statt "S" ein "?" ein.

Nun werden alle Ausgaben nach DD Cards gruppiert angezeigt. Wir interessieren uns nur für SYSOUT im Step RUN und schreiben vor diese Zeile ein „S“, um den Inhalt anzuzeigen.

Weitere Details dazu können Sie unserem Tutorial "Fehlersuche in OS/390 und OS/390-Anwendungen" entnehmen.

Aufgabe: ***Sehen Sie sich das Suchergebnis wie oben beschrieben an.***

Aufgabe: ***Suchen Sie nach Ihrer eigenen Matrikel-Nummer. Sehen Sie sich auch dieses Suchergebnis an. Fertigen Sie von diesem einen Screenshot entsprechend der Abbildung 3.5 an; dieser muss mindestens die rot umrandeten Zeilen enthalten. Der Screenshot soll im JPG-Format erstellt werden und darf 90 KByte nicht überschreiten.***

Aufgabe: ***Schicken Sie diesen sowie einen später anzufertigenden zweiten Screenshot unverpackt per E-Mail an Ihren Tutor, und zwar genau eine Abgabe-E-Mail mit je einem Anhang pro Screenshot.***

```
  Display  Filter  View  Print  Options  Help
 -------------------------------------------------------------------------------
 SDSF OUTPUT DISPLAY PRAKT20R JOB04593  DSID     4 LINE 9       COLUMNS 02- 81
 COMMAND INPUT ===>                                            SCROLL ===> 1
IEF142I PRAKT20R RUN - STEP WAS EXECUTED - COND CODE 0000
IGD104I PRAKT20.VSAM.LOAD                          RETAINED,  DDNAME=STEPLIB
IGD104I PRAKT20.VSAM.STUDENT                       RETAINED,  DDNAME=STUDDS
IEF285I   PRAKT20.PRAKT20R.JOB04593.D0000101.?         SYSIN
IEF285I   PRAKT20.PRAKT20R.JOB04593.D0000102.?         SYSIN
IEF373I STEP/RUN     /START 2006017.1928
IEF374I STEP/RUN     /STOP  2006017.1928 CPU    0MIN 00.03SEC SRB    0MIN 00.00S
IEF375I  JOB/PRAKT20R/START 2006017.1928
IEF376I  JOB/PRAKT20R/STOP  2006017.1928 CPU    0MIN 00.03SEC SRB    0MIN 00.00S
SUCHE 1187579
MATNR:    1187579
STUDSERV: mai91jhz
NACHNAME: Michaelsen
VORNAME:  Nils
******************************** BOTTOM OF DATA ********************************


  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=IFIND     F6=BOOK
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Abbildung 3.5: Ergebnis der Suche nach dem eigenen Namen

Selbst-Test

- **Unser Cobol Programm stellt das Verarbeitungsergebnis lediglich wohin ?**
- **Ginge es auch anders ? Wie?**
- **Haben Sie einen Vorschlag, wie man das Ganze benutzerfreundlicher gestalten könnte ?**

Die normale Ein- und Ausgabe würde darin bestehen, ein Cobol Anwendungsprogramm zu schreiben , welches auf den VSAM Dataset zugreift, ihn bearbeitet, abfragt und/Oder modifiziert.

Ein derartiges Beispiel ist verfügbar unter http://www.mainframes360.com/2009/04/vsam-tutorial-links.html.

oder

http://www.simotime.com/cblcbl01.htm#Introduction

Weitere Tutorials

https://sites.google.com/site/cobolmaterial/vasam-tutoriall

http://www.angelfire.com/folk/anoop/Vsamcob.pdf

4. Einführung in einen alternativen Index

4.1 AIX Cluster

Ein wichtiges Feld in jedem logischen Record ist der Key. Sein Inhalt kann zum Abrufen eines spezifischen logischen Records verwendet werden. Beispiele für Keys sind z.B. Personalnummern in einer Mitarbeiterdatei oder Teilenummern in einer Stücklistendatei. Ein KSDS muss mindestens einen Primärkey verwenden; er kann zusätzliche Sekundärkeys benutzen.

In einer VSAM Key Sequenced Organisation (KSDS) muss ein Record über einen eindeutigen Primärkey fester Länge in der gleichen Position innerhalb jedes logischen Records verfügen. Primärkeys können eine Größe von mindestens einem Byte und höchstens 255 Bytes haben.

Es können mehrere weitere Keyfelder in dem gleichen logischen Record vorhanden sein. Diese werden als alternate oder sekundäre Keys bezeichnet. Im Gegensatz zu den Primärkeys, die eindeutig und identisch sein müssen, können Sekundärkeys mit einem identischen Wert in mehr als einem logischen Record auftreten.

Ein VSAM Cluster, dessen Index Component Primärkeys enthält, wird als Basis Cluster bezeichnet. Jeder Sekundärkey erfordert einen weiteren VSAM Cluster, der als „Alternate Index“ Cluster (AIX Cluster) bezeichnet wird. Eine „Sphere“ ist ein Basis VSAM-Cluster mit seinen dazugehörigen AIX Clustern. Diese damit verbundenen associated Cluster sind die alternate Indizes (AIXs) des Basis Cluster.

Der Begriff AIX ist bei der Firma IBM mehrfach belegt. Neben VSAM bezeichnet der PowerPC sein weit verbreitetes Unix Betriebssystem ebenfalls als AIX (Advanced Interactive eXecutive). Wir verwenden AIX lediglich im Zusammenhang mit VSAM als Abkürzung für Alternate Index.

Ein AIX Cluster ist ein getrennter KSDS Cluster mit einer Index-Component und einer Daten Component. Die Index Component des AIX Clusters speichert die Sekundärkeys. Die AIX-Data Component speichert keine Daten, sondern Records, die Sekundärkeys und zugeordnete Primärkeys enthalten (Zeiger auf Daten in dem Basis-Cluster mit dem gleichen sekundären Keywert). Da identische Sekundärkeys in mehr als einem logischen Record auftreten können, kann jedem Sekundärkey in der AIX Data Component eine Gruppe von Primärkeys zugeordnet sein. Die Primärkeys in dem logischen AIX Record mit dem gleichen sekundären Keywert sind in aufsteigender Reihenfolge angeordnet.

Die Primärkeys in der AIX Data Component können nun benutzt werden, um über die Index Component des VSAM Basis Clusters auf den entsprechenden logischen Record zuzugreifen.

Jedes Feld in dem Basis Cluster Record kann als Sekundärkey verwendet werden. Der gleiche Basis-Cluster kann mehrere AIX Cluster mit unterschiedlichen Sekundärkey unterhalten. Es können mehrere Primärkey Records einen identischen Sekundärkey Wert benutzen. Beispiel: Der Primärkey ist die Mitarbeiternummer, und der Sekundärkey ist der Name der Abteilung. Offensichtlich kann die gleiche Abteilung mehrere Mitarbeiter haben.

Das Access Methods Services (AMS) utility program IDCAMS ermöglicht Definiton und Erstellung von AIXs. Es kann mit dem BLDINDEX Befehl aufgerufen werden. Ein AIX kann nur nach Definition des dazugehörigen Basis-Cluster definiert werden. Der BLDINDEX Befehl bewirkt einen sequentiellen Scan des angegebenen Basis Cluster. Während des Scans werden Sekundär Keywerte und Primärkeys extrahiert. Hiermit werden die AIX Records in der Data Component des AIX Clusters erzeugt.

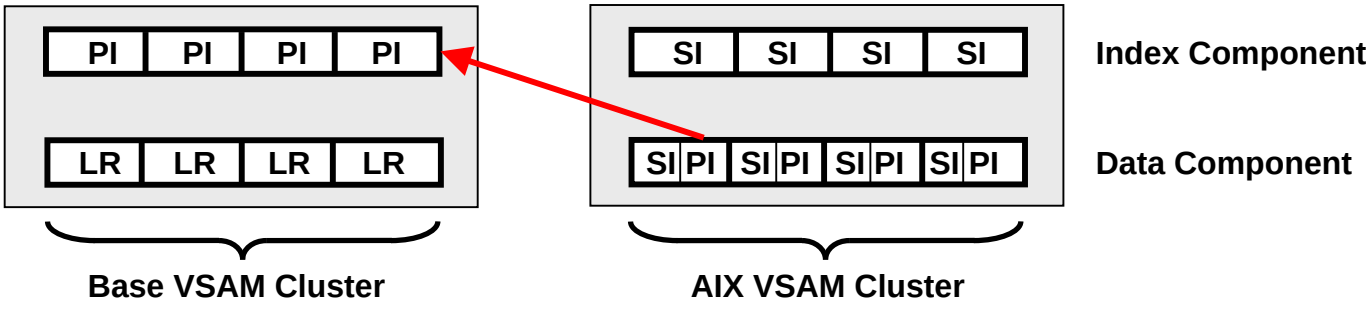


Abbildung 4.1

LR	**Logical VSAM Data Record**
PI	**Primärindex**
SI	**Sekundärindex**

4.2 Alternate Index Pfade

Vor einem Zugriff auf KSDS über eine alternate Index muss ein Pfad (Path) definiert werden. Mittels eines Pfades kann auf einen Basis-Cluster über die alternate Indizes zugegriffen werden. Ein Pfad wird mittels des AMS DEFINE PATH Befehls definiert und benannt. Mindestens ein Pfad muss für jede der alternaten Indizes definiert werden.

Der Pfadname verweist auf das Basis-Cluster und AIX-Cluster Paar. Wenn ein Anwendungsprogramm einen OPEN Befehl ausführt, werden sowohl der Base Cluster als auch der AIX Cluster geöffnet (opened).

5. Erweiterung um alternativen Index

Um nun auch über das Studserv-Kennung auf einen Record zugreifen zu können benötigt man einen alternativen Index Cluster. Der alternative Index Cluster hat seine eigene Kopie der Records und wird über einen PATH mit dem Base VSAM Cluster verbunden.

Einen alternativen Index Cluster kann man wieder mit einem geeigneten JCL-Script definieren. Es folgt geeigneter Programmcode.

```
//PRAKT20D JOB (),NOTIFY=&SYSUID
//*
//* DEFINE VSAM ALTERNATE INDEX
//*
//DEFCLS   EXEC PGM=IDCAMS,REGION=4096K
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *

  DEFINE ALTERNATEINDEX ( -
           NAME(PRAKT20.VSAM.STUDENT.ALTINDEX) -
           RELATE(PRAKT20.VSAM.STUDENT) -
           VOL(SMS009) -
           RECORDSIZE(55 55) -
           KILOBYTES(100 100) -
           KEYS(8 7) -
           UPGRADE )

  DEFINE PATH ( -
           NAME(PRAKT20.VSAM.STUDENT.PATH) -
           PATHENTRY(PRAKT20.VSAM.STUDENT.ALTINDEX) )

  BLDINDEX INDATASET(PRAKT20.VSAM.STUDENT) -
           OUTDATASET(PRAKT20.VSAM.STUDENT.ALTINDEX) -
           SORTCALL
/*
```

JCL Script zur Erstellung eines Alternate Index Cluster
Abbildung 5.1

Dieses JCL Script übernimmt verschiedene Aufgaben. Zuerst wird der alternative Index angelegt. Der alternate Key hat eine Länge von 8 Zeichen und beginnt bei Offset 7.
Damit der alternative Index funktioniert, muss ein PATH angelegt werden.
Zuletzt muss der alternative Index erstmalig erstellt werden. Das Wiederholen dieses Schrittes ist durch die Angabe von UPGRADE beim Definieren des alternativen Indexes nicht nötig.

Selbst-Test

- **Wodurch unterscheidet sich der alternative Index von dem primary Index?**
- **Was ist die Aufgabe des DEFINE PATH Kommandos ?**
- **Was ist die Aufgabe des BLDINDEX Kommandos ?**

Aufgabe: *Legen Sie ein entsprechendes JCL-Script als Member DEFAIX im PDS „PRAKT20.VSAM.CNTL“ an.*

Aufgabe: *Führen Sie dieses JCL-Script aus und kontrollieren Sie, ob der alternative Index tatsächlich angelegt wurde. Der VSAM-Cluster müsste unter anderem um weitere Komponenten PRAK<xxx>.VSAM.STUDENT.ALTINDEX.* erweitert worden sein.*

Jetzt brauchen wir ein Cobol Programm, welches den Alternate Index benutzt. Hierzu kopieren wir das ursprüngliche Cobol Programm (welches den Primärindex benutzte in einen neuen Member. Wir ändern es dann so ab, dass es den Alternate Index benutzt.

Aufgabe: *Kopieren Sie den Cobol-Quelltext in einen neuen Member STUD2.*

Aufgabe: *Modifizieren Sie den Member STUD2 entsprechend der nachfolgenden Liste.*

Hier die Liste der Änderungen in STUD2 (- heißt löschen, + heißt hinzufügen):

```
            ORGANIZATION IS INDEXED
            ACCESS IS DYNAMIC
            RECORD KEY IS MATNR
+           ALTERNATE RECORD KEY IS STUDSERV
            FILE STATUS IS FSTAT-CODE VSAM-CODE.
-          SELECT MATNR-IN
+          SELECT STUDSERV-IN
            ASSIGN TO SYSIN
            ORGANIZATION IS SEQUENTIAL
-           FILE STATUS IS MATNR-IN-CODE.
+           FILE STATUS IS STUDSERV-IN-CODE.
       DATA DIVISION.
       -------------
       FILE SECTION.
@@ -30,18 +31,18 @@
           05 STUDSERV    PIC X(8).
           05 NNAME       PIC X(20).
           05 VNAME       PIC X(20).
-      FD MATNR-IN
+      FD STUDSERV-IN
           RECORDING MODE F
           BLOCK 0 RECORDS
           RECORD 80 CHARACTERS
           LABEL RECORD STANDARD.
-        01 MATNR-RECORD  PIC X(80).
+        01 STUDSERV-RECORD  PIC X(80).
```

```
       WORKING-STORAGE SECTION.
       01 STATUS-AREA.
         05 FSTAT-CODE PIC X(2).
           88 I-O-OKAY  VALUE ZEROES.
-        05 MATNR-IN-CODE  PIC X(2).
+        05 STUDSERV-IN-CODE  PIC X(2).
         05 VSAM-CODE.
           10 VSAM-R15-RETURN-CODE    PIC 9(2) COMP.
           10 VSAM-FUNCTION-CODE      PIC 9(1) COMP.
@@ -51,22 +52,22 @@
         05 WS-STUDSERV  PIC X(8).
         05 WS-NNAME     PIC X(20).
         05 WS-VNAME     PIC X(20).
-      01 WS-MATNR-IN-RECORD.
-        05 WS-MATNR-IN  PIC X(7).
-        05 FILLER       PIC X(73).
+      01 WS-STUDSERV-IN-RECORD.
+        05 WS-STUDSERV-IN  PIC X(8).
+        05 FILLER          PIC X(72).

       PROCEDURE DIVISION.
-          OPEN INPUT MATNR-IN.
-          READ MATNR-IN INTO WS-MATNR-IN-RECORD.
-          DISPLAY "SUCHE " WS-MATNR-IN.
+          OPEN INPUT STUDSERV-IN.
+          READ STUDSERV-IN INTO WS-STUDSERV-IN-RECORD.
+          DISPLAY "SUCHE " WS-STUDSERV-IN.
           OPEN INPUT STUD-FILE.
           IF FSTAT-CODE NOT = "00"
             DISPLAY "OPEN INPUT VSAM FILE FS-CODE: " FSTAT-CODE
             PERFORM VSAM-CODE-DISPLAY
             STOP RUN
           END-IF.
-          MOVE WS-MATNR-IN TO MATNR.
-          READ STUD-FILE RECORD KEY IS MATNR.
+          MOVE WS-STUDSERV-IN TO STUDSERV.
+          READ STUD-FILE RECORD KEY IS STUDSERV.
           IF FSTAT-CODE NOT = "00" AND FSTAT-CODE NOT = "02"
             DISPLAY "READ STUD-FILE FS-CODE: " FSTAT-CODE
             PERFORM VSAM-CODE-DISPLAY
@@ -77,7 +78,7 @@
           MOVE NNAME TO WS-NNAME.
           MOVE VNAME TO WS-VNAME.
           CLOSE STUD-FILE.
-          CLOSE MATNR-IN.
+          CLOSE STUDSERV-IN.
           DISPLAY "MATNR:    " WS-MATNR.
           DISPLAY "STUDSERV: " WS-STUDSERV.
           DISPLAY "NACHNAME: " WS-NNAME.
```

Die Liste der Änderungen im Cobol-Progamm STUD sollte helfen, alle Änderungen einzubauen.

Es wird nun ein JCL-Script als Hilfsmittel benötigt, um das neue COBOL-Programm zu compilieren.

6. Arbeiten mit dem alternativen Index

Aufgabe: *Kopieren Sie den Member COMPILE im PDS „PRAKT20.VSAM.CNTL“ nach COMP2 und passen Sie diesen so an, dass Sie ihn zum Compilieren des neuen Cobol-Programms einsetzen können. Diese Anpassung muss bewirken, dass der alte ausführbare Member "STUD" in "PRAKT20.VSAM.LOAD" nicht überschrieben wird. Statt dessen soll hier ein zweiter Member "STUD2" angelegt werden.*

Aufgabe: *Compilieren Sie das neue Cobol-Programm.*

Nun bleibt uns noch, das gerade compilierte neue Cobol-Programm auszuführen. Dazu kopieren wir den Member RUN nach RUN2 und passen ihn an. Bei einem Vergleich von RUN mit RUN2 (Abbildung 7) fällt auf: In RUN2 muss noch zusätzlich etwas eingefügt werden, um den alternativen Index bekannt zu machen (über den erstellten PATH).

Nach Ausführung findet man die Daten des Studenten wie beim vorigen Cobol-Programm im Job Log des Jobs.

Aufgabe: *Erstellen Sie einen letzten neuen Member RUN2 in PRAKT20.VSAM.CNTL (siehe Abbildung 6.1) und führen Sie mit dessen Hilfe das zweite Cobol-Programm zu Test-Zwecken mehrfach aus.*

Aufgabe: *Suchen Sie wieder nach Ihrem eigenen Datensatz, diesmal über Ihr eigenes Studserv-Kennung. Fertigen Sie nach dem Muster von Abbildung 6.2 einen zweiten Screenshot an. Auch dieser soll im JPG-Format erstellt werden und darf 90 KByte nicht überschreiten und muss mindestens die rot umrandeten Zeilen enthalten.*

Aufgabe: *Schicken Sie nun beide Screenshots unverpackt und im JPG-Format an Ihren Tutor, und zwar in genau einer E-Mail mit je einem Anhang pro Screenshot.*

```
//PRAKT20R JOB (),NOTIFY=&SYSUID
//*
//* AUSFUEHREN DES COBOL-PROGRAMMS ZUM VSAM-ZUGRIFF
//* UEBER ALTERNATIVEN INDEX (STUDSERV-KUERZEL)
//*
//RUN      EXEC PGM=STUD2
//STEPLIB  DD DSN=PRAKT20.VSAM.LOAD,DISP=SHR
//STUDDS   DD DSN=PRAKT20.VSAM.STUDENT,DISP=SHR
//STUDDS1  DD DSN=PRAKT20.VSAM.STUDENT.PATH,DISP=SHR
//SYSIN    DD *
mai00aaa
/*
```

Abbildung 6.1: JCL-Script zum Ausführen des zweiten Cobol-Programms

```
  Display  Filter  View  Print  Options  Help
 ------------------------------------------------------------------------------
 SDSF OUTPUT DISPLAY PRAKT20R JOB04622  DSID     4 LINE 10      COLUMNS 02- 81
 COMMAND INPUT ===>                                           SCROLL ===> 1
IEF142I PRAKT20R RUN - STEP WAS EXECUTED - COND CODE 0000
IGD104I PRAKT20.VSAM.LOAD                          RETAINED,  DDNAME=STEPLIB
IGD104I PRAKT20.VSAM.STUDENT                       RETAINED,  DDNAME=STUDDS
IGD104I PRAKT20.VSAM.STUDENT.PATH                  RETAINED,  DDNAME=STUDDS1
IEF285I   PRAKT20.PRAKT20R.JOB04622.D0000101.?         SYSIN
IEF285I   PRAKT20.PRAKT20R.JOB04622.D0000102.?         SYSIN
IEF373I STEP/RUN     /START 2006018.1558
IEF374I STEP/RUN     /STOP  2006018.1558 CPU    0MIN 00.04SEC SRB    0MIN 00.00S
IEF375I  JOB/PRAKT20R/START 2006018.1558
IEF376I  JOB/PRAKT20R/STOP  2006018.1558 CPU    0MIN 00.04SEC SRB    0MIN 00.00S
SUCHE mai91jhz
MATNR:    1187579
STUDSERV: mai91jhz
NACHNAME: Michaelsen
VORNAME:  Nils
****************************** BOTTOM OF DATA ********************************

  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=IFIND     F6=BOOK
F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Abbildung 6.2: Ergebnis der Suche über das Studserv-Kennung

7. Tips

Tip Nr. 1

Die Matrikel-Nummern im sequentiellen Dataset müssen ausnahmslos aufsteigend geordnet sein. Wenn man das nicht beachtet, entstehen sehr schwer auffindbare Fehler.

Tip Nr. 2

Problem, dass der alternative Index, z.B. mam07abc, nicht gefunden wird:

Beim Durchsuchen der alternativen Indizes wird auf Groß- und Kleinschreibung geachtet. Möglicherweise ist in Ihrem ISPF-Editor eine Option aktiviert, die korrekt eingetippte klein geschriebene alternative Indizes fälschlicherweise in Großbuchstaben konvertiert. Wie sich dieses Problem beheben lässt, ist auch in diesem Forum in einem anderen Thread beschrieben.

Im Tutorial wurde als Aufgabe unter anderem gestellt, einen sequentiellen Dataset anzulegen und später dann auch zu editieren. Wie man das im Detail tun kann, ist im Tutorial nicht beschrieben. Doch denke ich, dass dies eine einfache Aufgabe für Sie ist, dies selbst herauszufinden.

7. Anhang

Für die Durchführung des Tutorials werden die folgenden Resourcen benötigt:

Inhalt der Datei VSAMcode.zip

CLEAN	**Delete VSAM Cluster**
COMPILE	**Compilieren des Cobol Programms, P. 8**
DEFAIX	**Define alternate Index, p.10**
DEFCLUST	**Erstellen VSAM Cluster, p.3**
STUD	**Cobol Program, P. 7**

Die Datei VSAMcode.zip enthält diese Resources als zip Archiv. Sie kann heruntergeladen werden unter

www.cedix.de/Vorles/Band3/Resources/VSAMcode.zip

CICS Cobol Tutorial 1
Cobol Hello World unter CICS

Ziel dieser Aufgabe ist es, ein "Hello World"-Programm in COBOL zu schreiben und Daten mittels CICS auf dem Bildschirm auszugeben. CICS ist ein z/OS Transaktions-Server, der Transaktionen ausführt. Transaktionen sind eine spezielle Art von Prozessen, die entweder zu 100 %, oder aber gar nicht ausgeführt werden. Auf einem Mainframe Rechner werden etwa 80 % aller Prozesse als Transaktionen ausgeführt, die meisten davon mittels des CICS Transaktionsservers.

Informationen über CICS finden Sie auch in Kapitel 8 dieses Buches.

Hinweis: Dieses Tutorial wurde unter Verwendung der Benutzer-ID "PRAK085" erstellt. In allen Dateinamen müssen Sie "PRAK085" durch ihre eigene Benutzer-ID ersetzen.

Aufgabe: ***Beschäftigen Sie sich mit diesem Tutorial und lösen Sie gewissenhaft alle kursiv geschriebenen und eingerahmten Aufgaben.***

Inhalt

Hinweise

Die CICS-Tutorien sind etwas schwerer als die vorangegangenen Tutorien. Hierzu noch ein paar Tips:

- **Nervige Fehlersuchen sind in den CICS-Tutorien normal, und Bestandteil des Lerneffektes.**
- **Falls "JCL Error" zurückgegeben wird, befindet sich der Fehler mit großer Wahrscheinlichkeit im JCL-Script und nicht im Cobol-Programm.**
- **Gehen Sie Schritt für Schritt vor; d.h. bringen Sie erst das Cobol-Programm zum Laufen, wenn das BMS-Programm fehlerfrei läuft und beginnen Sie erst mit den CICS-Installationen, sobald das Cobol-Programm fehlerfrei übersetzt wurde u.s.w.**
- **Falls der ISPF-Editor ungewollt gültigen Programmcode mit überflüssigen Zeilen-Nummern überschreibt, dann geben Sie bitte im ISPF-Editor auf der Kommandozeile "NUMBER OFF" ein (siehe http://mvs.wiu.edu/stumvs/TSO/TSOChangingYourEditProfile.html).**
- **Sie können NUR den Transaktionsnamen X< Prak-Nr.>, also z.B. X613 oder X609 für die User Ids prak613 oder prak609 verwenden. Bei anderen Transaktionsnamen wird die Fehlermeldung "`DFHAC2033 22:14:09 CICS You are not authorized to use transaction W607`" zurückgegeben.**
- **Wenn man SUB auf einen Mapset anwendet, dann werden aus den selbst gewählten Namen für BMS-Eingabefelder neue Variablen gebildet (die zukünftige Schnittstelle zwischen Mapset und Programm), welche unter Umständen verbotene Schlüsselwörter werden. Aus dem Feldnamen "C" wird z.B. "CF" und aus "I" wird z.B. "IF" gebildet, beides verbotene Schlüsselwörter.**

  Deshalb sind "C", "I" sowie "G" als mögliche Namen für Felder in einer BMS-Map verboten.

  Als Symptom entsteht "MAXCC=12". Wenn man in die Joblog-File hineinschaut, findet man z.B.'CF is a reserved word related to language'.

1. Einführung

Das Erstellen einer neuen Anwendung verwendet in der Regel eine Entwicklungsumgebung für das Schreiben und Übersetzen des neuen Programms und eine Produktionsumgebung für das Ausführen des Programms. In der letzten Übung war TSO und ISPF die Entwicklungsumgebung, JES die Ausführungsumgebung für Compile und Link, und TSO die Ausführungsumgebung für die Programmausführung. In der vorliegenden Übung verwenden wir wieder TSO und ISPF als Entwicklungsumgebung, JES als Ausführungsumgebung für Compile und Link, und CICS als Produktionsumgebung (Ausführungsumgebung für die Programmausführung).

TSO ist ein z/OS-Subsystem. CICS ist ein weiteres z/OS-Subsystem. Jedes der beiden Subsysteme hat eine eigene Benutzerschnittstelle (eine eigene Shell). Um eine CICS-Anwendung zu erstellen, müssen wir mit beiden Subsystemen arbeiten: Mit TSO, um die Anwendung zu erzeugen, und mit CICS, um die Anwendung (unter dem CICS-Subsystem) auszuführen. Da z/OS ein Multi-User-Betriebssystem ist (multisession-fähig), können wir gleichzeitig eine TSO-Session und eine CICS-Session auf unserem Arbeitsplatzrechner laufen lassen. Jede Session läuft in einem eigenen Fenster.

Wir starten einen 3270-Emulator zunächst für eine TSO-Session und loggen uns ein.
Wir öffnen den "Data Set Utility"-Screen und erzeugen (Allocate) einen neuen Partitioned Dataset: "PRAK085.CICS.COBOL". Dabei verwenden wir die in der nachstehenden Aufgabe angegebenen Parameter.

Außerdem brauchen wir noch einen Partitioned Dataset mit dem vorgegebenen Namen "PRAK085.LIB", dessen Members von der Entwicklungsumgebung während der CICS-Programmentwicklung mit Daten gefüllt werden. Verwenden Sie bei dessen Anlage ebenfalls die unten angegebenen Parameter.

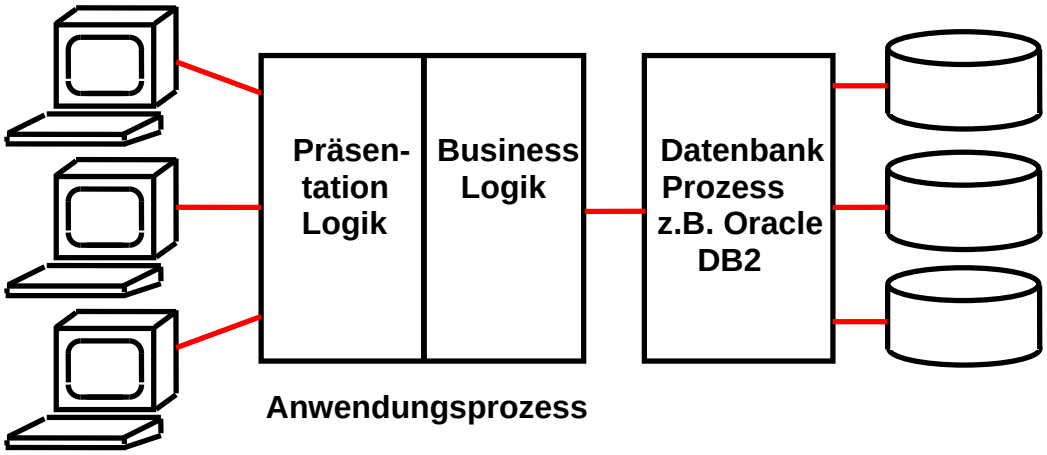
Dieser Name besteht aus einem "Project"-Begriff und einem "Group"-Begriff. Es fehlt der "Type"-Begriff. Wenn wir das Typ-Feld leer lassen, wird TSO dies nicht akzeptieren. Deshalb tragen wir den Namen 'PRAK085.LIB' (mit Hochkommas!) in die Zeile "Data Set Name" ein. Damit wird auch dieser Dataset angelegt.

Aufgabe: ***Legen Sie die Datasets "PRAK085.CICS.COBOL" , "PRAK085.LIB" an. Verwenden Sie dazu folgende Parameter:***

Space units . . . . . KILOBYTE	***Record format . . . . FB***
Primary quantity . . 16	***Record length 80***
Secondary quantity 1	***Block size 320***
Directory blocks . . 2	***Data set name type : PDS***

Unsere Anwendung besteht aus zwei Programmteilen und einem JCL-Script für die Übersetzung. Wir erstellen diese als Members in dem neuen Partitioned Dataset "PRAK085.CICS.COBOL".

Ein sauber strukturiertes CICS-Programm besteht aus zwei Teilen: Business Logic und Presentation Logic. Während die Geschäftslogik die eigentliche Arbeit verrichtet, dient die Präsentationslogik dazu, die Ergebnisse der Geschäftslogik auf eine attraktive Art auf dem Bildschirm darzustellen.



Der Server Teil einer Client/Server Anwendung besteht aus zwei Teilen: einem Anwendungsprozess und einem Datenbankprozess.

Dargestellt ist eine logische Struktur. 2-Tier, 3-Tier oder n-Tier Konfigurationen unterscheiden sich dadurch, wie diese Funktionen auf physische Server abgebildet werden.

Einige der Alternativen sind:

- **In einer 2-Tier Konfiguration befindet sich der Anwendungsprozess auf dem gleichen Rechner wie der Klient.**
- **In einer 3-Tier Konfiguration befinden sich Anwendung und Datenbank als getrennte Prozesse auf getrennten Rechnern.**
-

2. Presentation Logic

Business Logic wird in Sprachen wie C, C++, COBOL, PL/I, Java, ASSEMBLER, REXX usw. geschrieben. Für die Presentation Logic gibt es viele Alternativen. Eine sehr moderne Alternative benutzt Java Server Pages und einen Web Application Server, um den Bildschirminhalt innerhalb eines Web-Browsers darzustellen. Eine weitere sehr moderne Alternative benutzt XML, SOAP und Web Services. Die älteste (und einfachste) Alternative verwendet das CICS-BMS (Basic Mapping Support)-Subsystem und eine 3270 Bildschirmausgabe. Dies setzen wir hier ein. BMS-Programme werden in der BMS-Sprache geschrieben. In unserem Beispiel wird die Business Logic in COBOL und die Presentation Logic in BMS geschrieben.

Wir fangen mit der Presentation Logic an, rufen den "Edit Entry Panel Screen" auf und editieren ein Member "`MAPCO01`" für den neu angelegten Partitioned Dataset "PRAK085.CICS.COBOL" (s. Abbildung 1). Man beachte den Buchstaben "O" und die Ziffer "0" in diesem Membernamen (!!!)

```
 File  Edit  Confirm  Menu  Utilities  Compilers Help
--------------------------------------------------------------------------------
EDIT       PRAK085.CICS.COBOL(MAPCO01) - 01.05            Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 //PRAK085M JOB (),CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1),NOTIFY=&SYSUID,
000200 //         REGION=4M
000300 //ASSEM EXEC DFHMAPS,MAPNAME='MSET085',RMODE=24
000400 //SYSUT1  DD  *
000500 MSET085 DFHMSD TYPE=MAP,MODE=INOUT,LANG=COBOL2,STORAGE=AUTO,           *
000600                TIOAPFX=YES
000700 *       MENU MAP
000800 MAP085  DFHMDI SIZE=(24,80),CTRL=(PRINT,FREEKB)
000900         DFHMDF POS=(9,23),ATTRB=(ASKIP,NORM),LENGTH=34,               *
001000                INITIAL='WELCOME TO THE MAGIC WORLD OF CICS'
001100         DFHMDF POS=(12,33),ATTRB=(ASKIP,NORM),LENGTH=16,              *
001200                INITIAL='MEMBER PRAK085 !'
001300         DFHMSD TYPE=FINAL
001400         END
001500 /*
001600 //
Command ===>                                                 Scroll ===> PAGE
 F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
```

Abbildung 1: Das BMS-Programm

Die sehr einfache von CICS verwendete BMS Sprache besteht nur aus den drei Befehlen DFHMSD, DFHMDI und DFHMDF, mehr wird für die Beschreibung der BMS Maps nicht benötigt. Es ist nachvollziehbar, dass graphische Bildschirm-Darstellungen eine komplexere Beschreibungssprache (z.B. HTML, Java usw.) benötigen. (Es existieren auch Werkzeuge, die BMS Code direkt in HTML Code umwandeln).

Die BMS Sprache wird mit Hilfe von drei Assembler Macros implementiert, die alle mit den Buchstaben DFHM anfangen. Ein BMS Programm ist also in Wirklichkeit ein Assembler Programm.

Unser BMS-Programm verwendet 3 Befehlstypen: DFHMSD, DFHMDI und DFHMDF. Ein BMS-Bildschirm verwendet das 24 Zeilen x 80 Zeichen / Zeile 3270-Bildschirmformat. Die Ein- und Ausgabemasken werden als Felder innerhalb der 24x80-Matrix dargestellt, jeweils mit der Angabe: Zeilenadresse, Spaltenadresse und Feldlänge. Dies geschieht mit Hilfe des DFHMDF-Befehls. Der DFHMDF-Befehl in den Zeilen 000900 und 001000 definiert ein Feld, welches in Zeile 9, Spalte 23 beginnt, 34 Zeichen lang ist, und mit dem Wert "WELCOME TO THE MAGIC WORLD OF CICS" initialisiert wird. Unser Beispiel-BMS-Programm enthält 2 derartige DFHMDF-Befehle.

Der DFHMSD-Befehl (Zeile 000500) definiert einen "Mapset" mit dem Namen "MSET085". (Bitte benutzen Sie hier die letzten 3 Ziffern Ihrer User ID.) Eine Transaktion involviert in der Regel mehrere unterschiedliche Screens (Maps): Z.B. einen Screen, in dem der Benutzer zu einer Eingabe aufgefordert wird und einen weiteren Screen, welcher die Ergebnisse der Anfrage wiedergibt. Alle Maps (Screens) eines Transaktionstyps werden zu einem Mapset zusammengefasst.

Die einzelnen Maps (Screens) eines Mapsets werden durch den Befehl DFHMDI definiert und zur Kennzeichnung mit einer Label versehen. In unserem einfachen "Hello World"-Beispiel besteht der Mapset aus einer einzigen Map, die in Zeile 000800 mit der Bezeichnung "MAP085" definiert wird.

Das Member "MAPCO01" stellt in Wirklichkeit ein JCL-Script dar. Im Gegensatz zu Tutorial 2 wird das zu verarbeitende File nicht mit INFILE='xxx.yyy.zzz' angegeben. Der JCL-Befehl in Zeile 000400 "//SYSUT1 DD *" besagt, dass das zu verarbeitende File unmittelbar danach folgt (Zeilen 000500 bis 001400).

Wir geben auf der Kommandozeile den ISPF-Befehl "SUB" ein. Es wird die Prozedur "DFHMAPS" ausgeführt.

JCL findet das Member "DFHMAPS" (Zeile 000300) in der Library "ADCD.Z18.PROCLIB(DFHMAPS)". Durch die Ausführung von "DFHMAPS" werden zwei Ausgabe-Files erzeugt. Einmal wird der übersetzte BMS-Quellcode in einen Member in eine CICS-interne Library mit dem Namen "DFH310.CICS.PRAKLOAD" gestellt. Hier kann ihn die BMS-Komponente des CICS-Subsystems später finden.

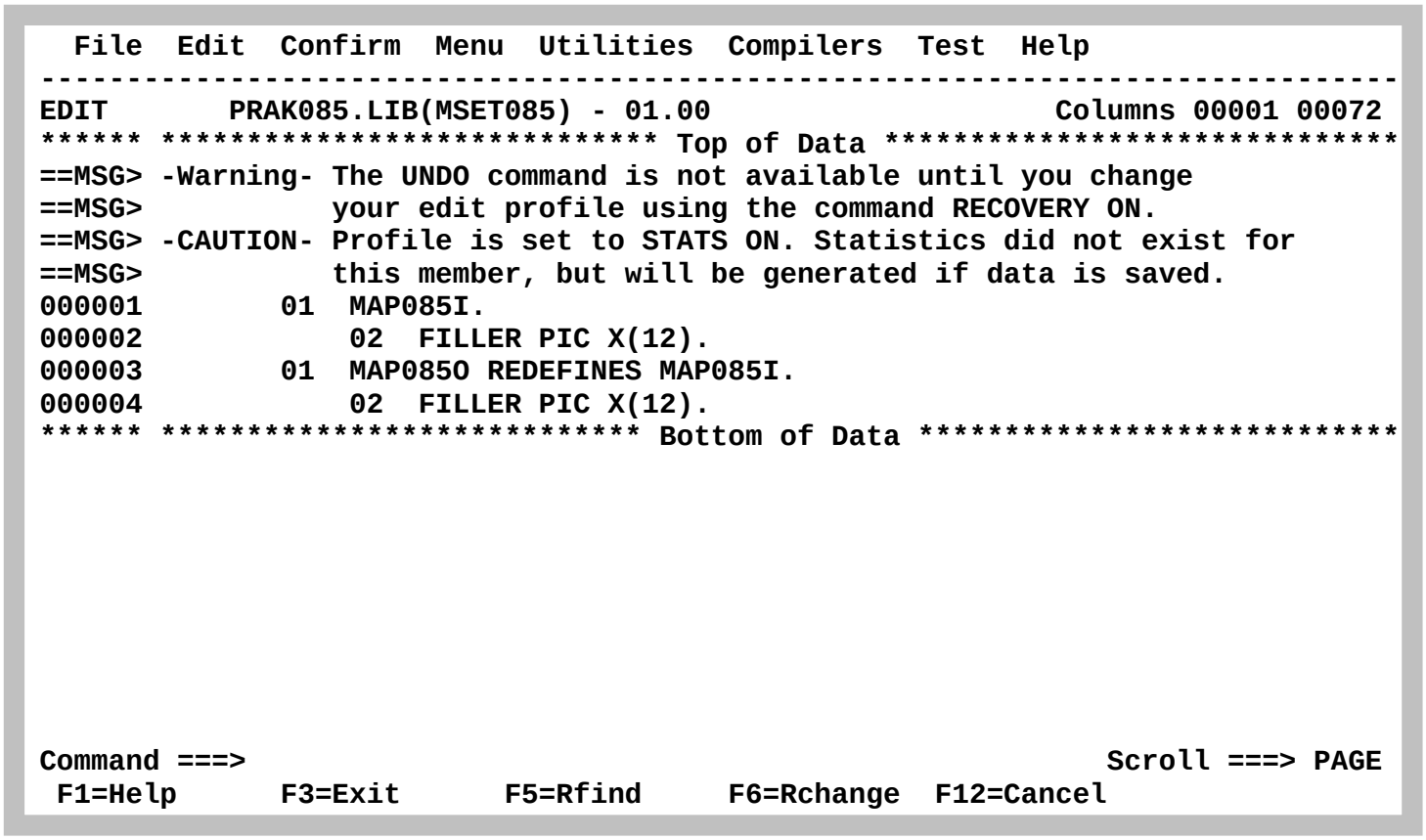


Abbildung 2: Das Member "MSET085"

Das zweite Ausgabe-File wird als Member "MSET085" in den Partitioned Dataset "PRAK085.LIB" gestellt (s. Abbildung 2). Alle Ein- und Ausgabe-Masken, die auf dem Bildschirm wiedergegeben werden sollen, werden ja bereits durch das BMS-Programm definiert. Das Business Logik-Programm bearbeitet diese Daten als COBOL-Strukturen, und diese werden von DFHMAPS während der Übersetzung von "MAPCO01" gleich miterzeugt und in "PRAK085.LIB(MSET085)" abgespeichert.

Aufgabe: ***Modifizieren Sie Ihr BMS-Programm derart, dass es den Transaktionsnamen "U<Login-Nr.>", "TUTORIAL 3 IN COBOL" sowie den Namen oder die Namen der Autoren enthält. Dieses BMS-Programm soll später einen Screen ähnlich der Abbildung 29 generieren. (Hinweis: Die Ausführung des JCL-Scriptes erfolgt fehlerfrei, wenn dieses mit der Statusmeldung "MAXCC=0" beendet wird).***

3. CICS Anwendungsprogramm

Wir rufen den Editor auf und erstellen in COBOL das Anwendungsprogramm COB085 (s. Abbildung 3). Letzteres enthält das CICS-Statement "EXEC CICS SEND MAP('MAP085') ...

Dies ist das einzige EXEC CICS Statement in unserem Cobol Programm. Normalerweise würden viele EXEC CICS Statements vorhanden sein, die vor allem etwas mit dem Zugriff auf Daten zu tun haben würden.

```
  File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       PRAK085.CICS.COBOL(COB085) - 01.03            Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100        IDENTIFICATION DIVISION.
000200        PROGRAM-ID. COB085.
000300        ENVIRONMENT DIVISION.
000400        DATA DIVISION.
000500        WORKING-STORAGE SECTION.
000600        COPY MSET085.
000700        LINKAGE SECTION.
000800        PROCEDURE DIVISION.
000900            EXEC CICS SEND MAP('MAP085')
001000                       MAPSET('MSET085')
001100                       FROM(MAP085O)
001200                       ERASE
001300            END-EXEC.
001400            GOBACK.
****** **************************** Bottom of Data ****************************

Command ===>                                                 Scroll ===> PAGE
 F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
```

Abbildung 3: Das Anwendungsprogramm "COB085"

Dieses bewirkt, dass die Map mit dem im BMS-Programm definierten Mapnamen "MAP085" aus dem Mapset "MSET085" mit Hilfe des 3270-Protokolls an den Bildschirm des Endbenutzers übertragen wird.

```
  File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
--------------------------------------------------------------------------------
EDIT       PRAK085.CICS.COBOL(COBSTA03) - 01.00            Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 //PRAK085C JOB (),CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1),NOTIFY=&SYSUID,
000200 //         REGION=4M
000300 //STEP1  EXEC COBCICS,PARM.TRN='COBOL3'
000400 //TRN.SYSIN DD DISP=SHR,DSN=&SYSUID..CICS.COBOL(COB085)
000500 //COB.SYSLIB DD DSN=&SYSUID..LIB,DISP=SHR
000600 //LKED.SYSIN DD *
000700  NAME COB085(R)
000800 /*
****** **************************** Bottom of Data ****************************




Command ===> SUB                                          Scroll ===> PAGE
 F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
```

Abbildung 4: JCL-File "COBSTA03"

Dieses Programm soll nun übersetzt werden. Dazu wird für den Partitioned Dataset "PRAK085.CICS.COBOL" ein neues Member "COBSTA03" erstellt (s. Abbildung 4). Dies ist ein JCL-File, das die Prozedur COBCICS enthält. COBCICS ruft zunächst den CICS-Precompiler auf, der alle CICS-Befehle in COBOL-Befehle übersetzt. Anschließend wird der COBOL-Compiler aufgerufen, der ein Maschinenprogramm erstellt und in eine für das CICS-Subsystem zugängliche Library stellt.

Wir geben "SUB" ein und warten, bis der Job ausgeführt wurde (s. Abbildung 4).

4. Definieren und Installieren des Programms in CICS

Wir entwickeln unsere Anwendung unter TSO. Wir wollen sie unter CICS laufen lassen. Dazu muss sie als Teil des CICS-Subsystems installiert werden. Es ist komfortabel, mit zwei z/OS-Sessions gleichzeitig zu arbeiten. Dazu kann man einfach noch ein zweites 3270 Emulator Fenster öffnen, mit dem man CICS startet.

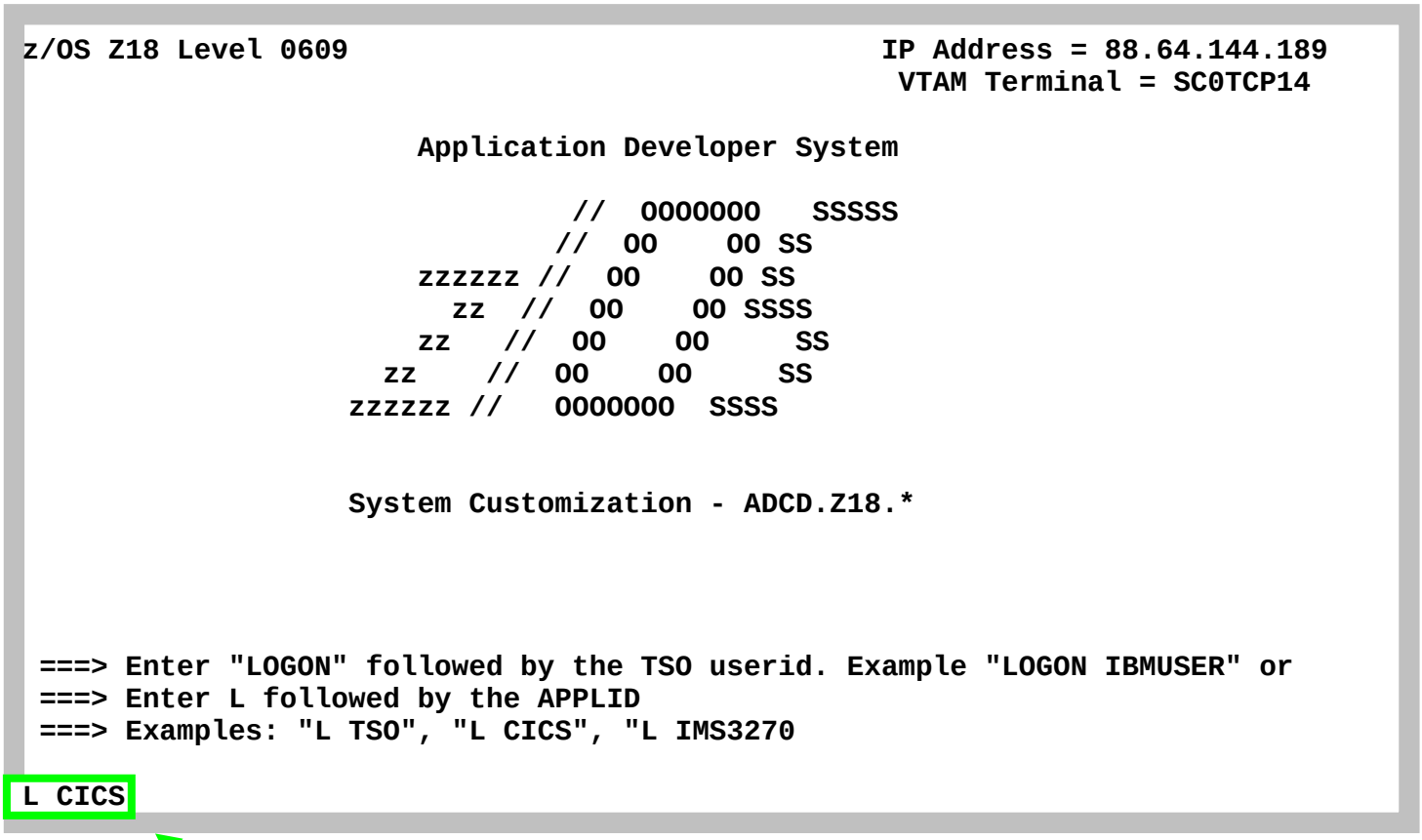


Abbildung 5: Logon-Bildschirm

Wir loggen uns mit "L CICS" (einschließlich Eingabetaste) ein und rufen damit das CICS-Subsystem auf (s. Abbildung 5). Das CICS Subsystem verwendet eine ganz andere Shell und Benutzeroberfläche als TSO.

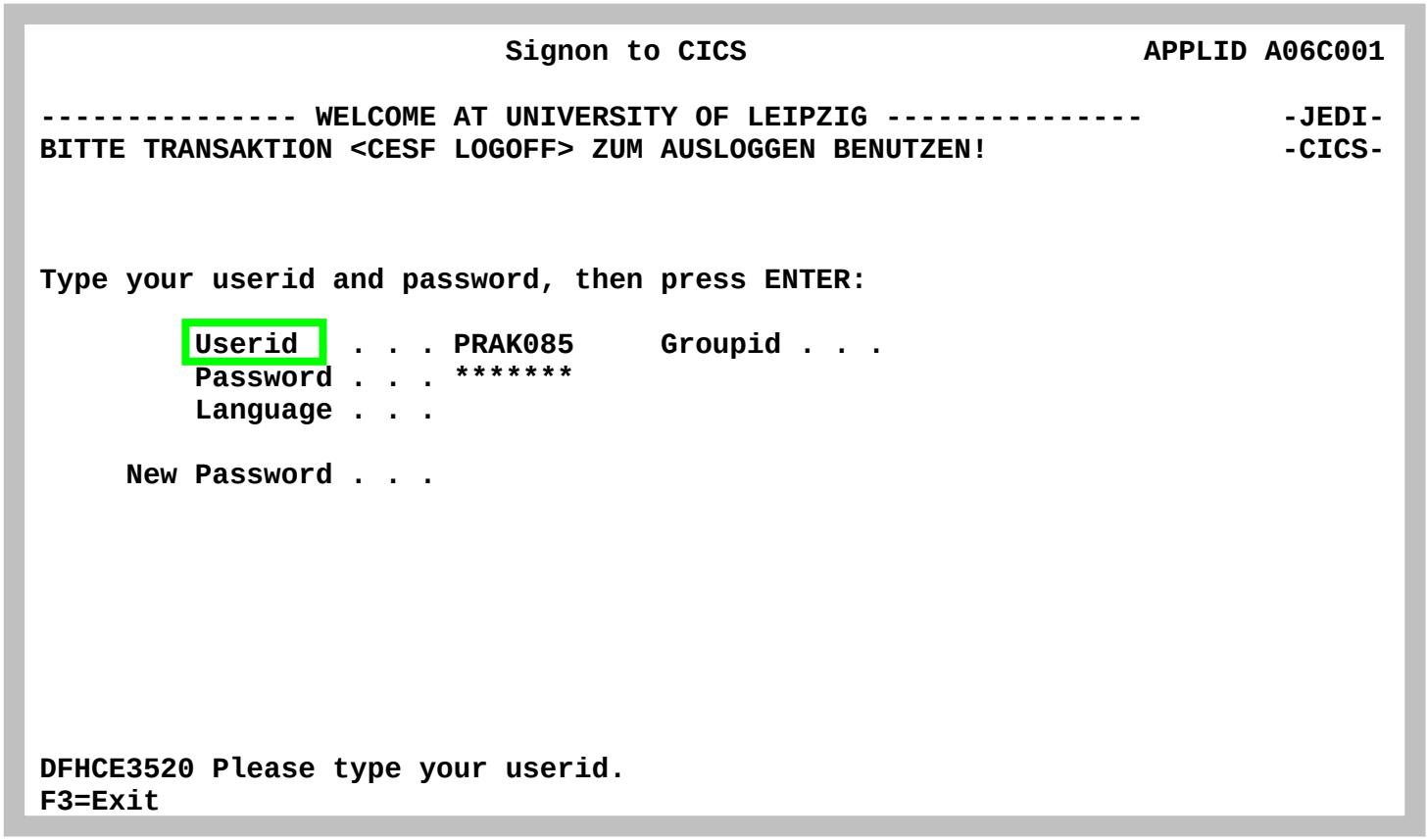


Abbildung 6: Signon to CICS-Bildschirm

Der Signon to CICS-Bildschirm erscheint (s. Abbildung 6). Hier müssen wir unseren TSO-Loginnamen sowie das entsprechende Passwort eingeben. Die Eingabetaste führt uns in den nächsten Screen (s. Abbildung 7).

Abbildung 7: Sign-on is complete

Mit der "Tab"-Taste bewegen wir den Cursor auf die unterste Zeile.

Abbildung 8: Beispielkommando "CEDA DISPLAY GROUP(*)"

CICS erwartet, dass man eine (von vielen) Transaktionen aufruft. Die unterschiedlichen Transaktionen werden normalerweise durch die Eingabe einer aus vier Zeichen bestehenden Tranaktions-ID aufgerufen.

Genauso wie eine Linux ftp shell andere Commands als die Linux bash shells benutzt, verwendet CICS ebenfalls einen anderen Command Interpreter als z.B. TSO oder z/OS Unix System Services.

Der CICS-Kommandointerpreter ist ebenfalls als Transaktion implementiert (etwas anderes als Transaktionen kann CICS nicht ausführen). Er wird mit der Transaktions-ID "CEDA" aufgerufen, gefolgt von einer Parameterliste, welche CICS-Kommandos sowie Eingabedaten enthält. Als Beispiel geben wir das Kommando "CEDA DISPLAY GROUP(*)" gefolgt von der Eingabetaste ein (siehe Abb. 9).

```
display group(*)
ENTER COMMANDS
   GROUP
   AOR2TOR
   ARTT
   ATC
   CBPS
   CEE
   CICREXX
   CICSJADP
   CICS0ADP
   CSQ
   CSQCKB
   CSQSAMP
   CTA1TCP
   C001EZA
   C001TCP
   DAVIN15
   DAVIN4
 + DAVIN8

                                                    SYSID=C001 APPLID=A06C001
   RESULTS: 1 TO 17                              TIME:  00.00.00  DATE: 01.035
 PF 1 HELP       3 END 4 TOP 5 BOT 6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Abbildung 9: Auflistung der Gruppen

Wenn unter CICS Anwendungen (Transaktionen) installiert werden, dann wird für jede Transaktion eine "Group" angelegt. In der "Group" befinden sich typischerweise Komponenten wie das Anwendungsprogramm selbst, der dazugehörige Mapset sowie ein Eintrag, der die Transaktion mit einer 4-stelligen TRID (TRansaktions-ID) verknüpft.

Es werden die ersten 17 Gruppen angezeigt (s. Abbildung 9). Durch Betätigen der F8 Taste können Sie sich weitere Gruppen ansehen.

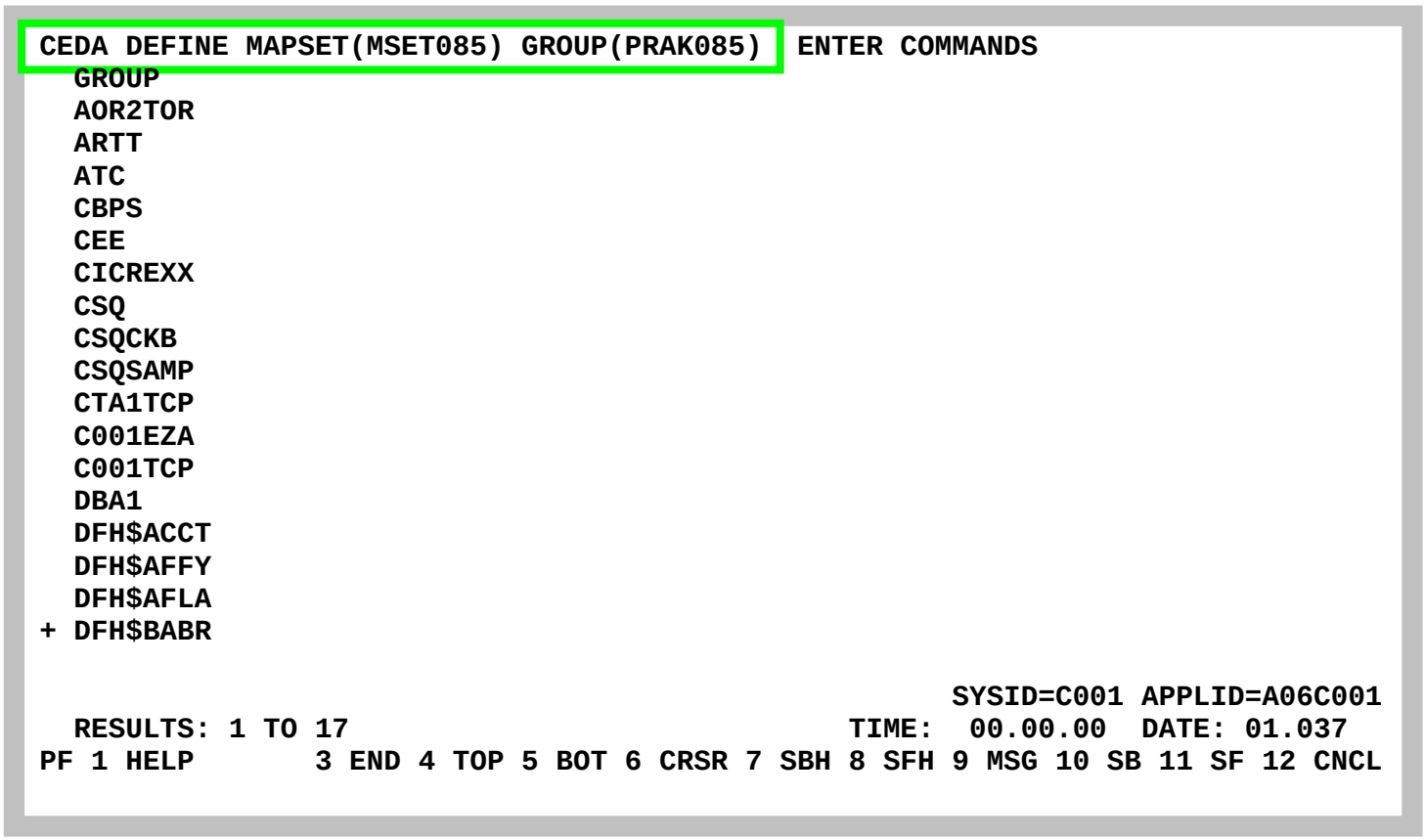


Abbildung 10: Definition des Mapsets "MSET085" und der Gruppe "PRAK085"

Wir definieren für unsere Transaktion eine eigene Gruppe "PRAK085" und den dazugehörigen Mapset als "MSET085". Hierzu überschreiben wir die oberste Zeile, die als Kommandozeile dient, mit dem CEDA-Befehl "CEDA DEFINE MAPSET(MSET085) GROUP(PRAK085)" (s. Abbildung 10, bitte Großbuchstaben benutzen!)

Um zu bestätigen drücken wir die Eingabetaste.

```
 CEDA DEFINE MAPSET(MSET085) GROUP(PRAK085)
 OVERTYPE TO MODIFY                                        CICS RELEASE = 0530
  CEDA  DEFine Mapset( MSET085   )
   Mapset         : MSET085
   Group          : PRAK085
   Description  ==>
   REsident     ==> No                 No | Yes
   USAge        ==> Normal             Normal | Transient
   USElpacopy   ==> No                 No | Yes
   Status       ==> Enabled            Enabled | Disabled
   RSl            : 00                 0-24 | Public









  I New group PRAK085 created.
                                                 SYSID=C001 APPLID=A06C001
  DEFINE SUCCESSFUL                          TIME:  00.00.00  DATE: 01.037
PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Abbildung 11: Bestätigung der Erstellung von Gruppe und Mapset

CICS teilt uns mit, dass die neue Gruppe "PRAK085" erstellt sowie der Mapset erfolgreich definiert wurde (s.
Abbildung 11**).**
Führen wir nochmals den Befehl "CEDA DISPLAY GROUP(*)" aus, so finden wir in der dargestellten Liste den Eintrag "PRAK085" (erfordert ein Durchblättern der Liste unter Benutzung der Taste F8).

Nachdem der Mapset unter CICS definiert wurde, definieren wir jetzt das COBOL-Programm "COB085".

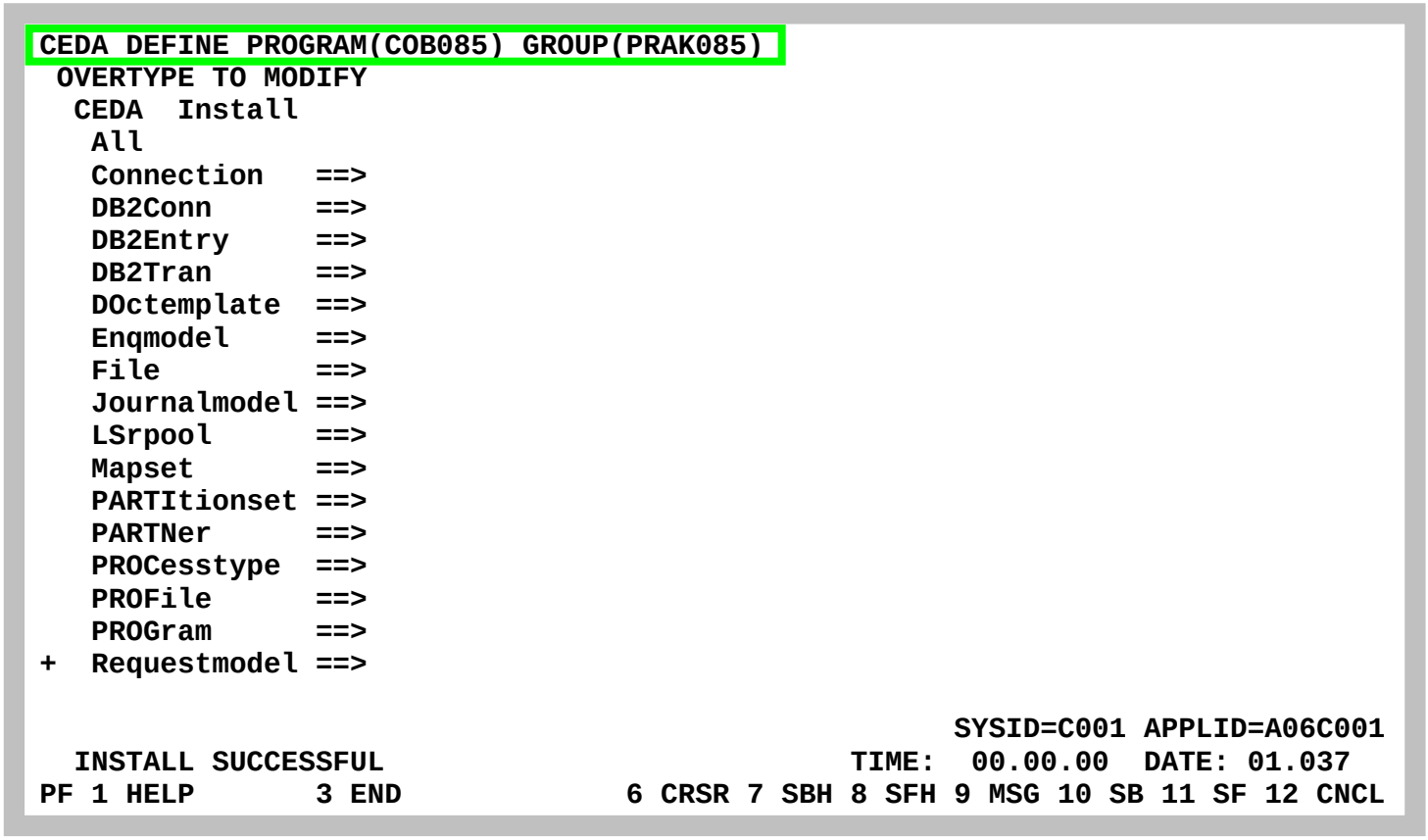


Abbildung 12: Definition des Programms "COB085"

Wir definieren für die Gruppe "PRAK085" unser Anwendungsprogramm "COB085", indem wir den entsprechenden CEDA-Befehl

"CEDA DEFINE PROGRAM(COB085) GROUP(PRAK085)"

in die oberste Zeile schreiben und anschließend die Eingabetaste betätigen (s. Abbildung 12).

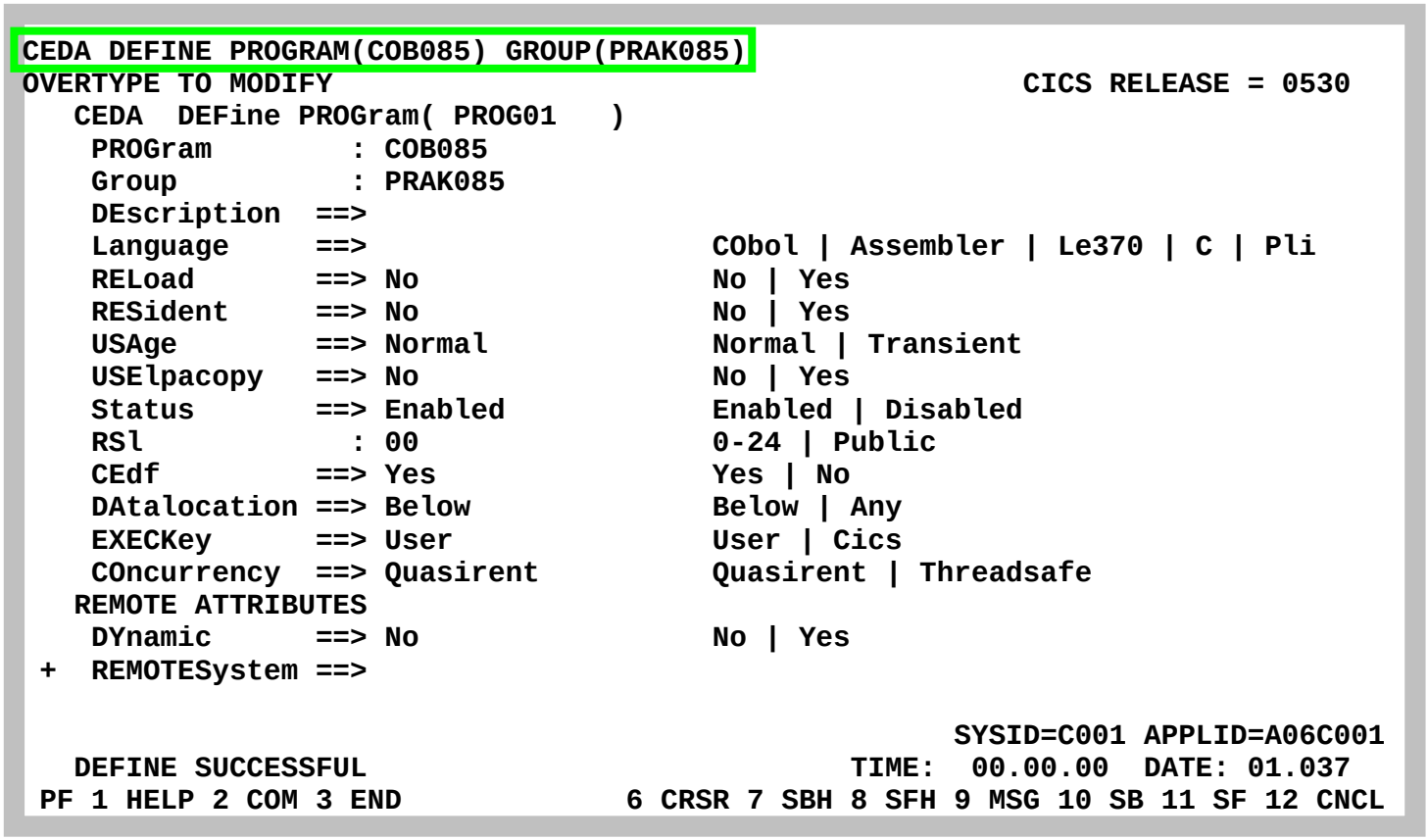


Abbildung 13: Definition von "COB085"

CEDA will einiges von uns wissen (s. Abbildung 13). Wir übernehmen alle Default-Werte und geben als Sprache "Le370" an
(s. Abbildung 14).

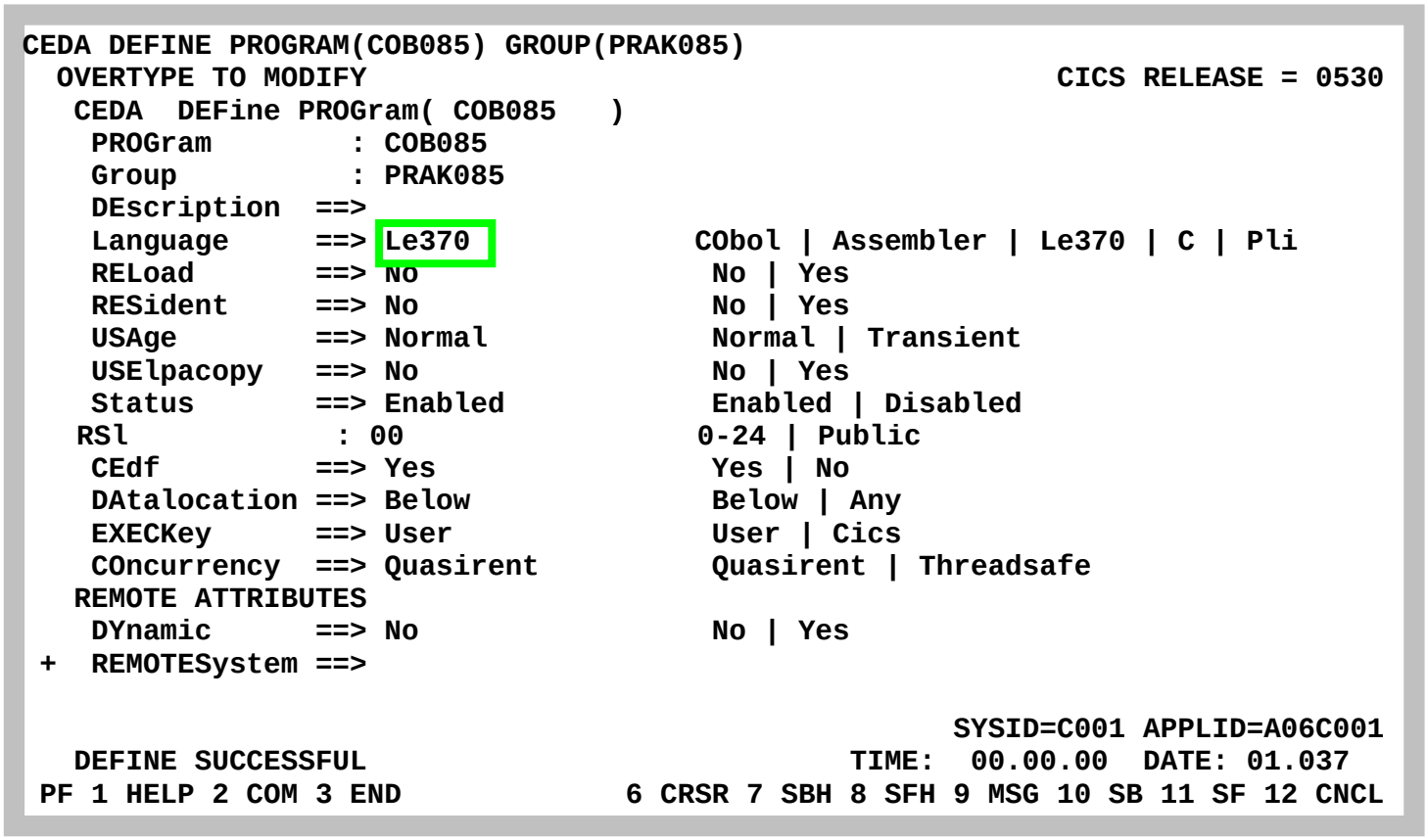


Abbildung 14: Auswahl der Sprache

Was ist denn "Le370" für eine Sprache? "Le370" ist überhaupt keine Sprache, sondern eine Laufzeitumgebung. CICS braucht an dieser Stelle in Wirklichkeit nicht die Angabe der Quellsprache unseres Anwendungsprogramms (wir haben es ja bereits übersetzt), sondern die Angabe der Laufzeitumgebung des von uns verwendeten Compilers. Alle modernen z/OS-Compiler verwenden eine gemeinsame Laufzeitumgebung, die den Namen "Le370" (oder LE390) trägt. Mit Betätigung der Eingabetaste erscheint der nächste Screen.

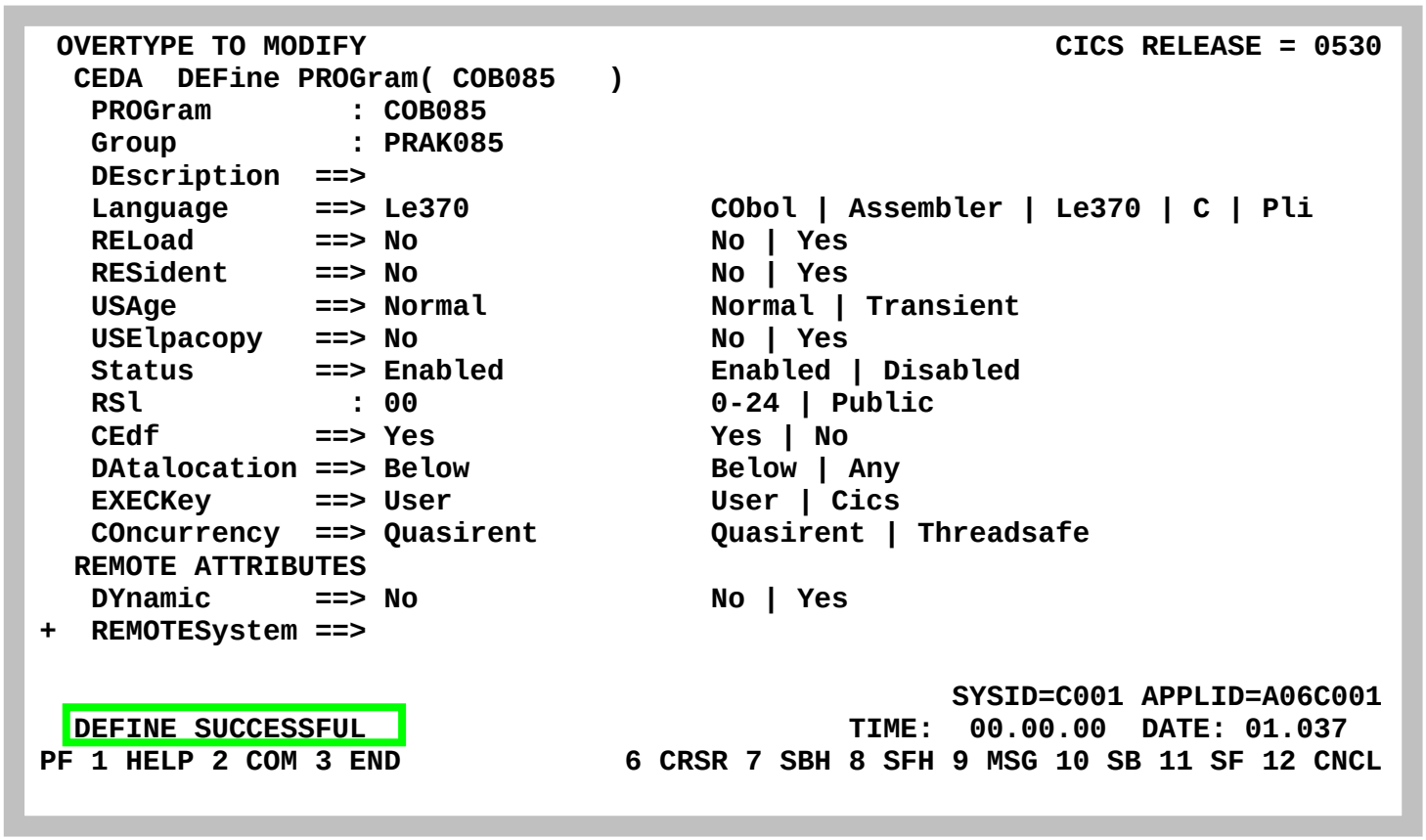


Abbildung 15: Erfolgreiche Definition

Nach dem Betätigen der Eingabetaste erscheint der Bildschirm in Abbildung 15. Wir verlassen die Definition mit der Eingabe von F3, als Ergebnis davon erscheint "SESSION ENDED" (s. Abbildung 16).

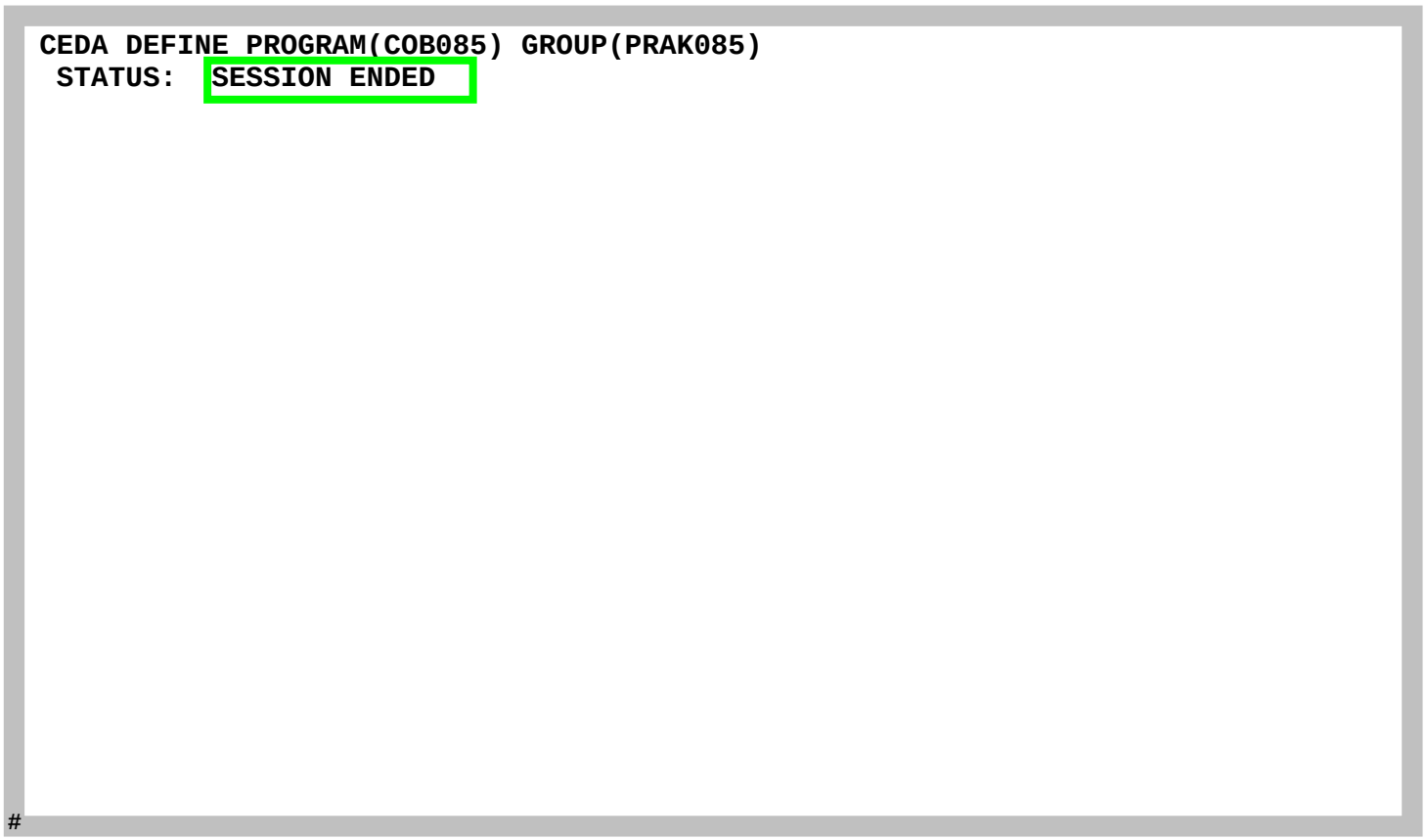


Abbildung 16: Definition wurde beendet

In diesen Bildschirm geben wir in die oberste Zeile den nächsten CEDA-Befehl, gefolgt von der Eingabetaste, ein (s. Abbildung 17).

```
 CEDA DEFINE TRANS(X085) GROUP(PRAK085)
STATUS:  SESSION ENDED
```

Abbildung 17: Definition der Transaktion X085

Unsere Transaktion soll wie alle anderen Transaktionen vom Bildschirm über eine 4-stellige Transaktions-ID aufgerufen werden. Wir wählen hierfür die ID "X085" und teilen diese Wahl mit Hilfe des "CEDA DEFINE"-Befehls mit (s. Abbildung 17). Genauso wie der Mapset "MSET085" und das Programm "COB085" wird dies Bestandteil der Gruppe "PRAK085". Abschließend betätigen wir die Eingabetaste.

```
DEFINE TRANS(X085) GROUP(PRAK085)
 OVERTYPE TO MODIFY                                        CICS RELEASE = 0530
  CEDA  DEFine TRANSaction( X085 )
   TRANSaction  ==> X085
   Group        ==> PRAK085
   DEscription  ==>
   PROGram      ==>
   TWasize      ==> 00000             0-32767
   PROFile      ==> DFHCICST
   PArtitionset ==>
   STAtus       ==> Enabled           Enabled | Disabled
   PRIMedsize     : 00000             0-65520
   TASKDATALoc  ==> Below             Below | Any
   TASKDATAKey  ==> User              User | Cics
   STOrageclear ==> No                No | Yes
   RUnaway      ==> System            System | 0 | 500-2700000
   SHutdown     ==> Disabled          Disabled | Enabled
   ISolate      ==> Yes               Yes | No
   Brexit       ==>
+ REMOTE ATTRIBUTES
  S PROGRAM OR REMOTESYSTEM MUST BE SPECIFIED.
                                                    SYSID=C001 APPLID=A06C001

PF 1 HELP 2 COM 3 END             6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Abbildung 18: Der Definitions-Screen

CEDA will mehrere Angaben von uns und schlägt eine Reihe von Default-Werten vor (s. Abbildung 18).

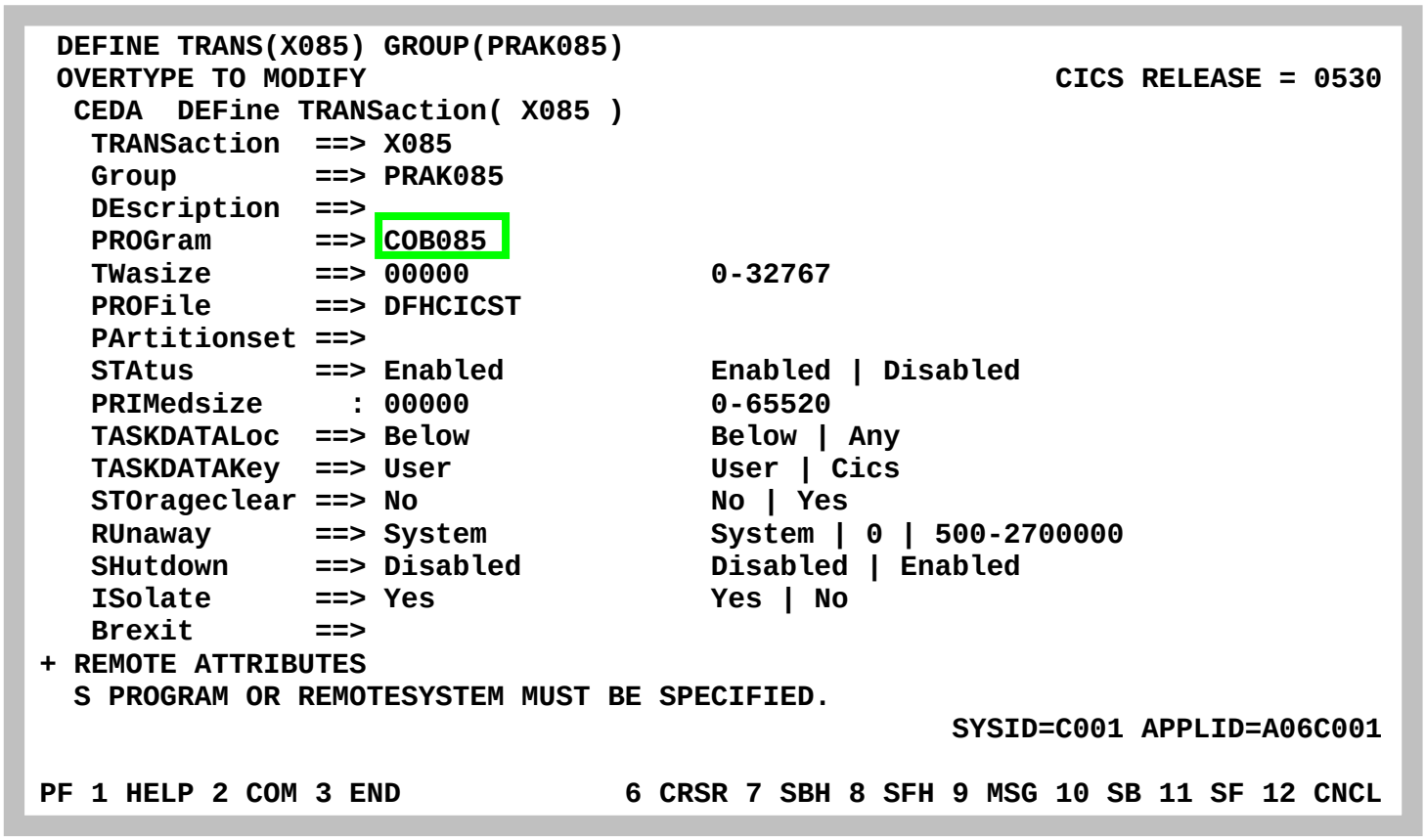


Abbildung 19: Eingabe der Parameter

Wir übernehmen alle Default-Werte und geben in die Zeile "PROGram" den Namen unseres Anwendungsprogramms, nämlich "COB085" ein und bestätigen mit der Eingabetaste (s. Abbildung 19).

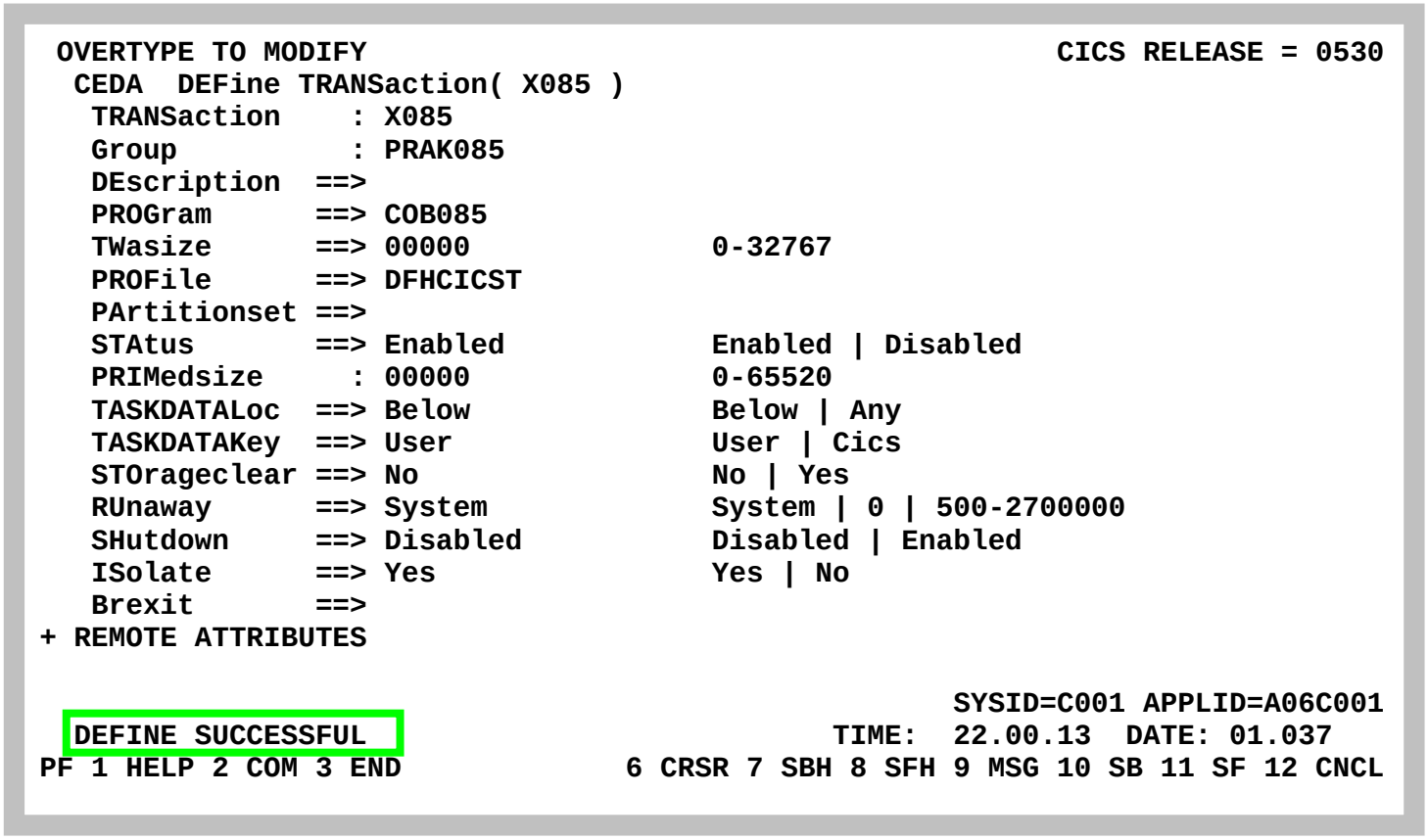


Abbildung 20: Erfolgreiche Definition

Die Meldung "DEFINE SUCCESSFUL" erscheint (s. Abbildung 20).

Wir verlassen dieses Menü mit F3.

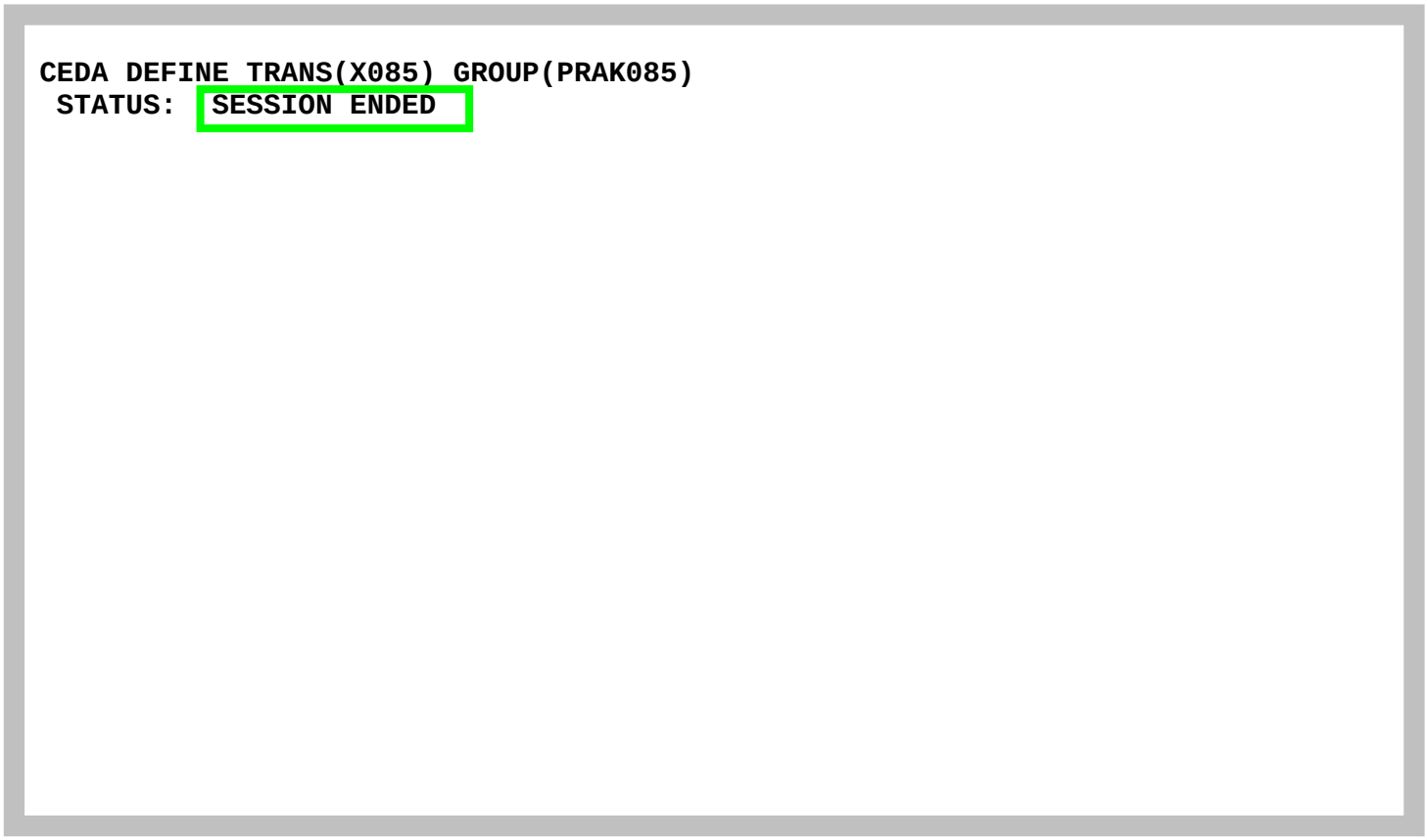


Abbildung 21: SESSION ENDED

Der Bildschirm in der Abbildung 21 erscheint. Wir haben CICS den Namen unseres Mapsets, Anwendungsprogramms und eine dazugehörige Transaktions-ID bekanntgegeben. Jetzt müssen diese drei Komponenten in die CICS-Programmbibliothek übernommen (installiert) werden.

Abbildung 22: Aufruf der Installation

Wir geben in die oberste Zeile das CEDA-INSTALL-Kommando ein und drücken die Eingabetaste (s. Abbildung 22).

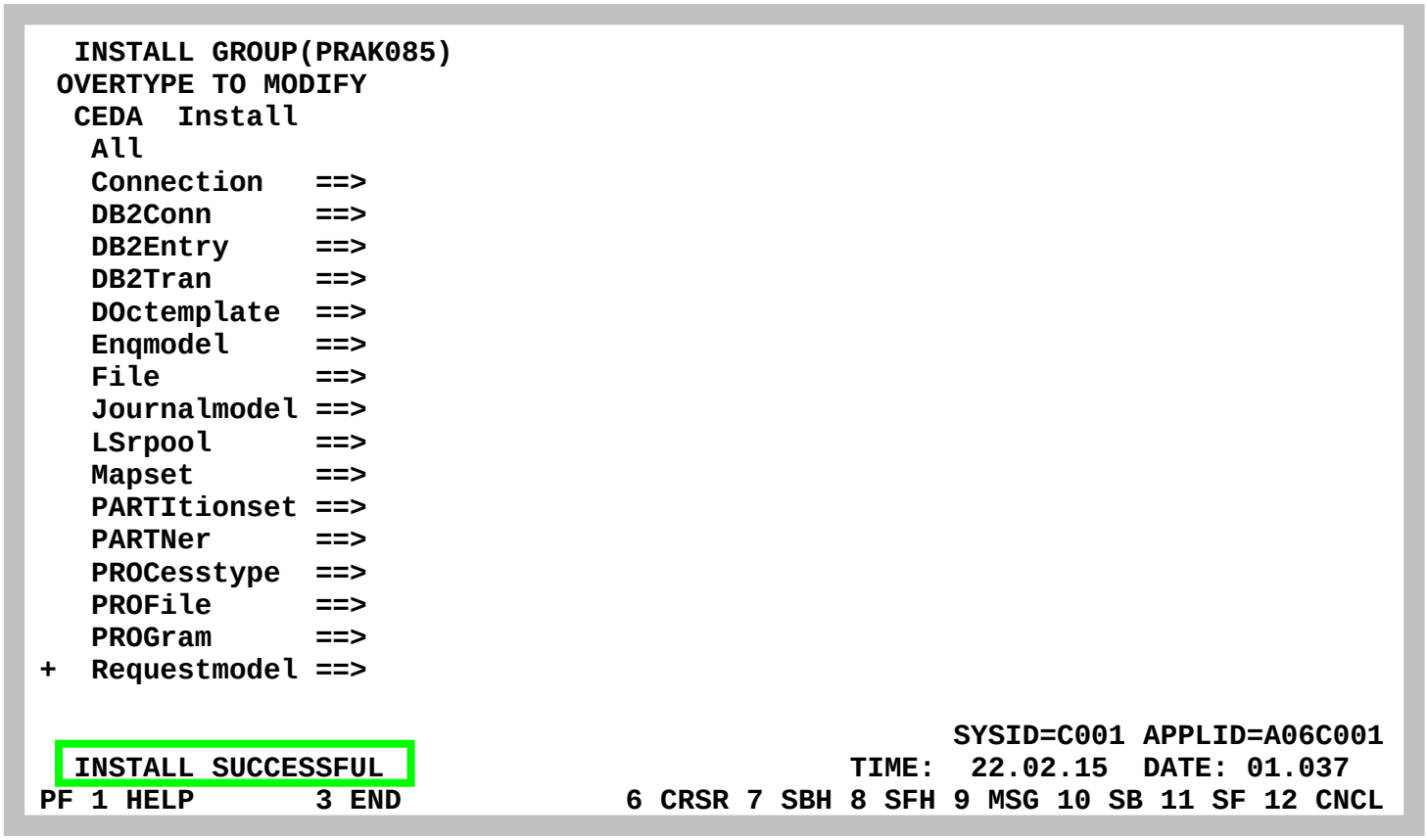


Abbildung 23: Installation war erfolgreich

CEDA bestätigt, dass die Installation erfolgreich war (s. Abbildung 23). Wir verlassen mit F3 dieses Menü.

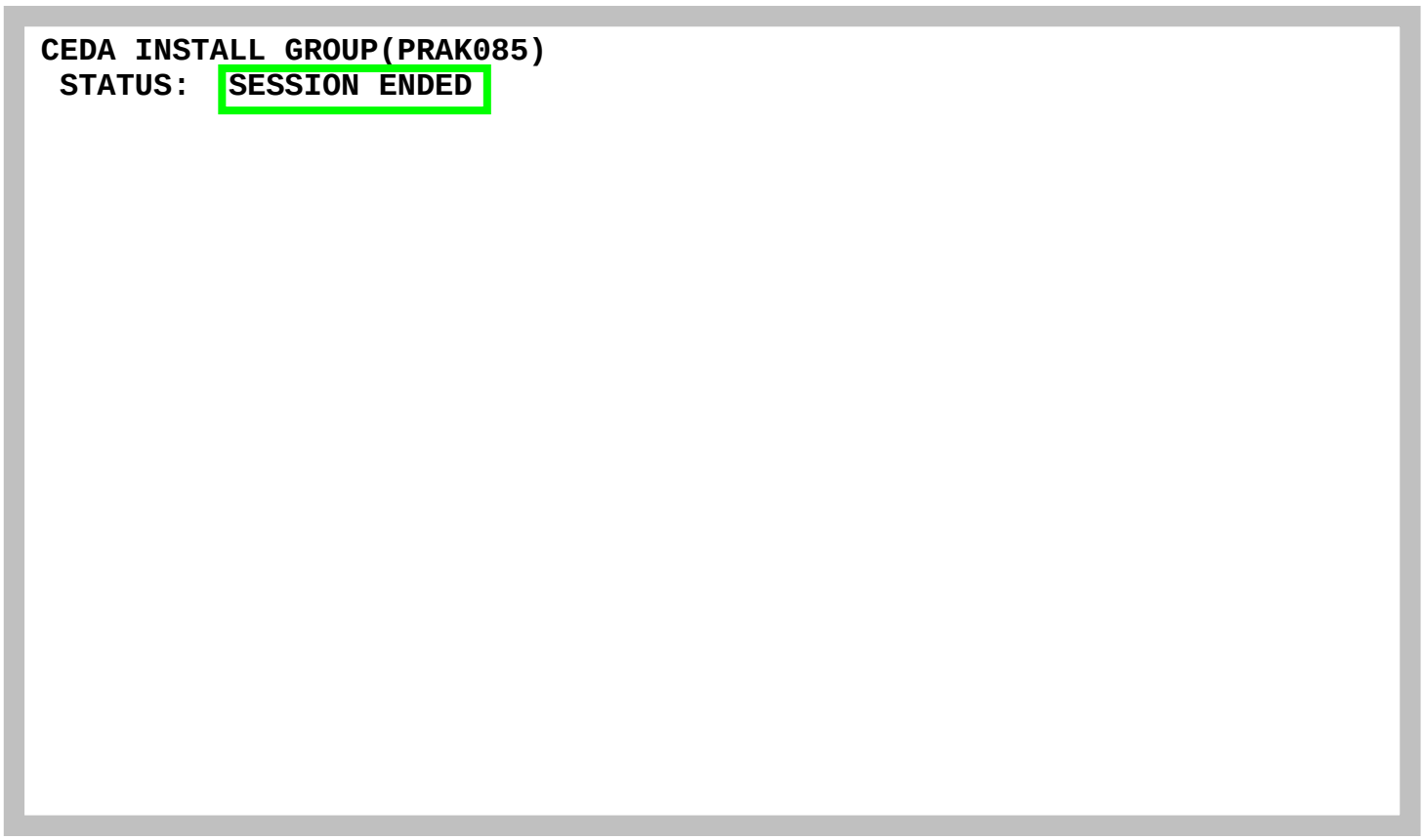


Abbildung 24: Beendete Installation

Der obige Bildschirm erscheint (s. Abbildung 24). Unsere Transaktion ist als Teil der CICS-Anwendungsbibliothek installiert worden und kann nun aufgerufen und damit ausgeführt werden. Hierzu löschen wir die oberste Zeile (die CEDA-Kommandozeile) ganz und rufen unsere Anwendung auf, indem wir dort unsere Transaktions-ID, nämlich "X085", eingeben.

5. Ausführen des Programms als CICS Transaktion

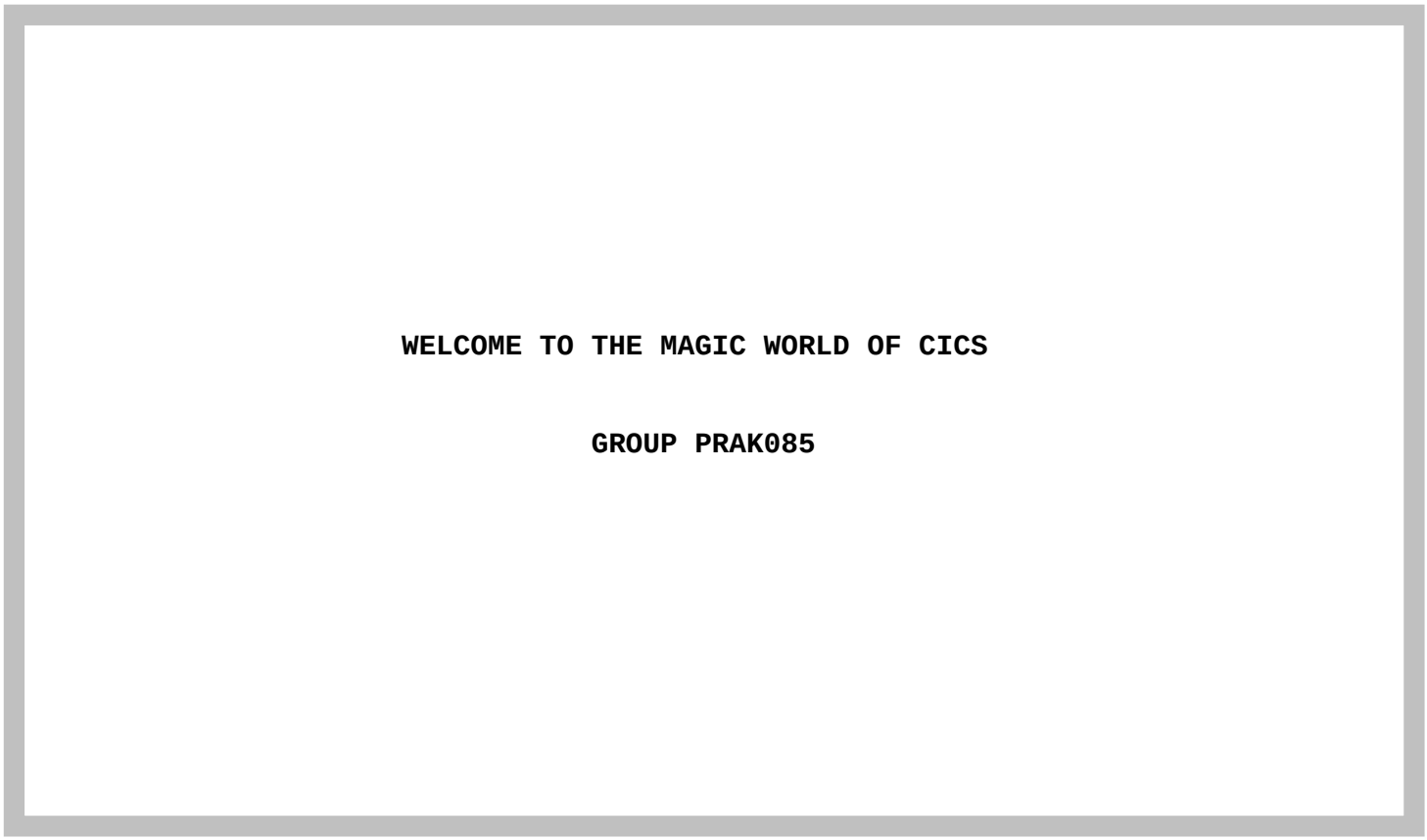


Abbildung 25: Ausgabe der Transaktion "X085" auf dem Bildschirm

Nach dem Betätigen der Eingabetaste erscheint unsere CICS-Transaktion auf dem Monitor (s. Abbildung 25).

Wir betätigen die Eingabetaste, um zum nächsten Screen zu gelangen.

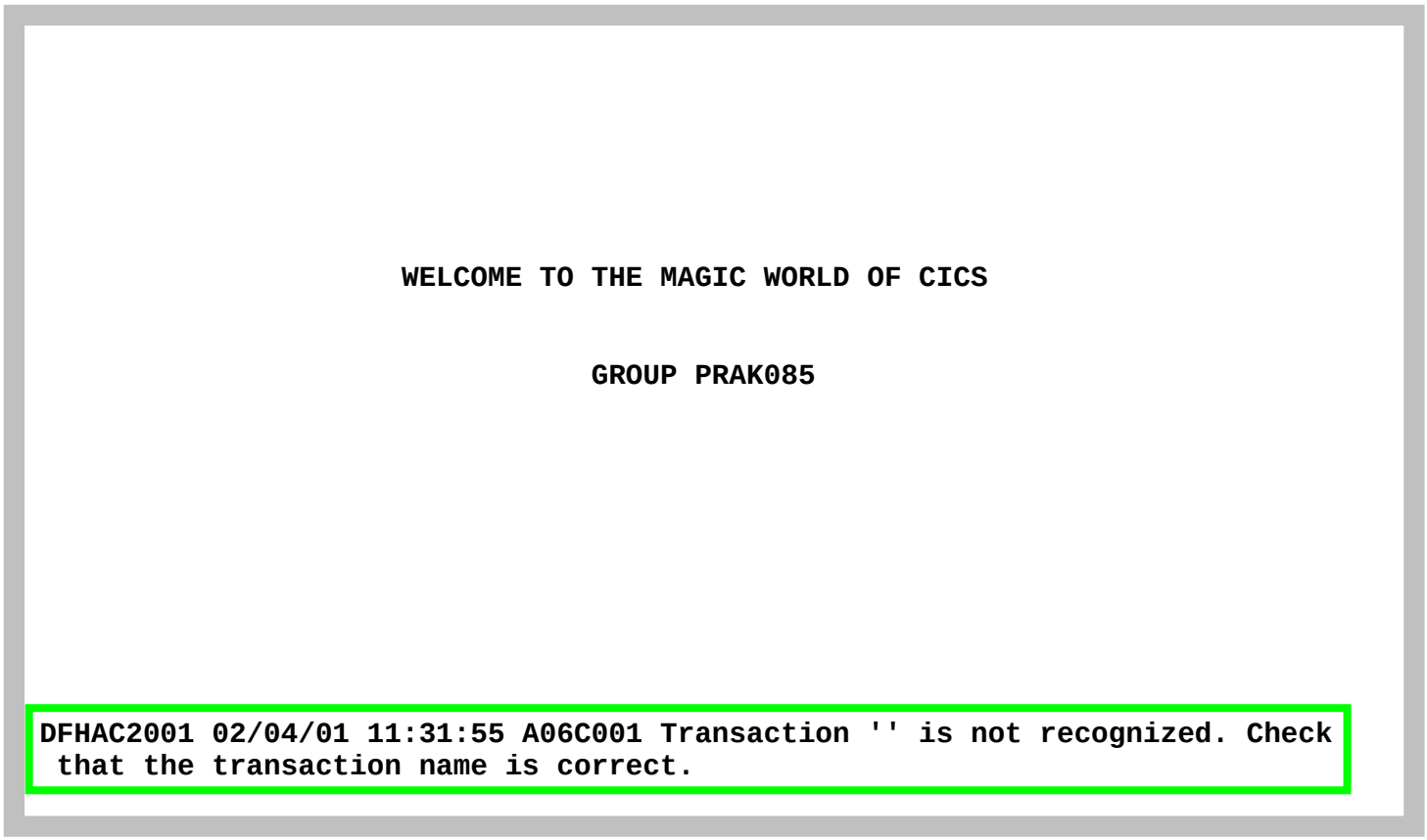


Abbildung 26: Fehlermeldung

Dies terminiert die Bildschirmausgabe unserer Transaktion und erzeugt wieder eine (belanglose) Fehlermeldung (s. Abbildung 26).

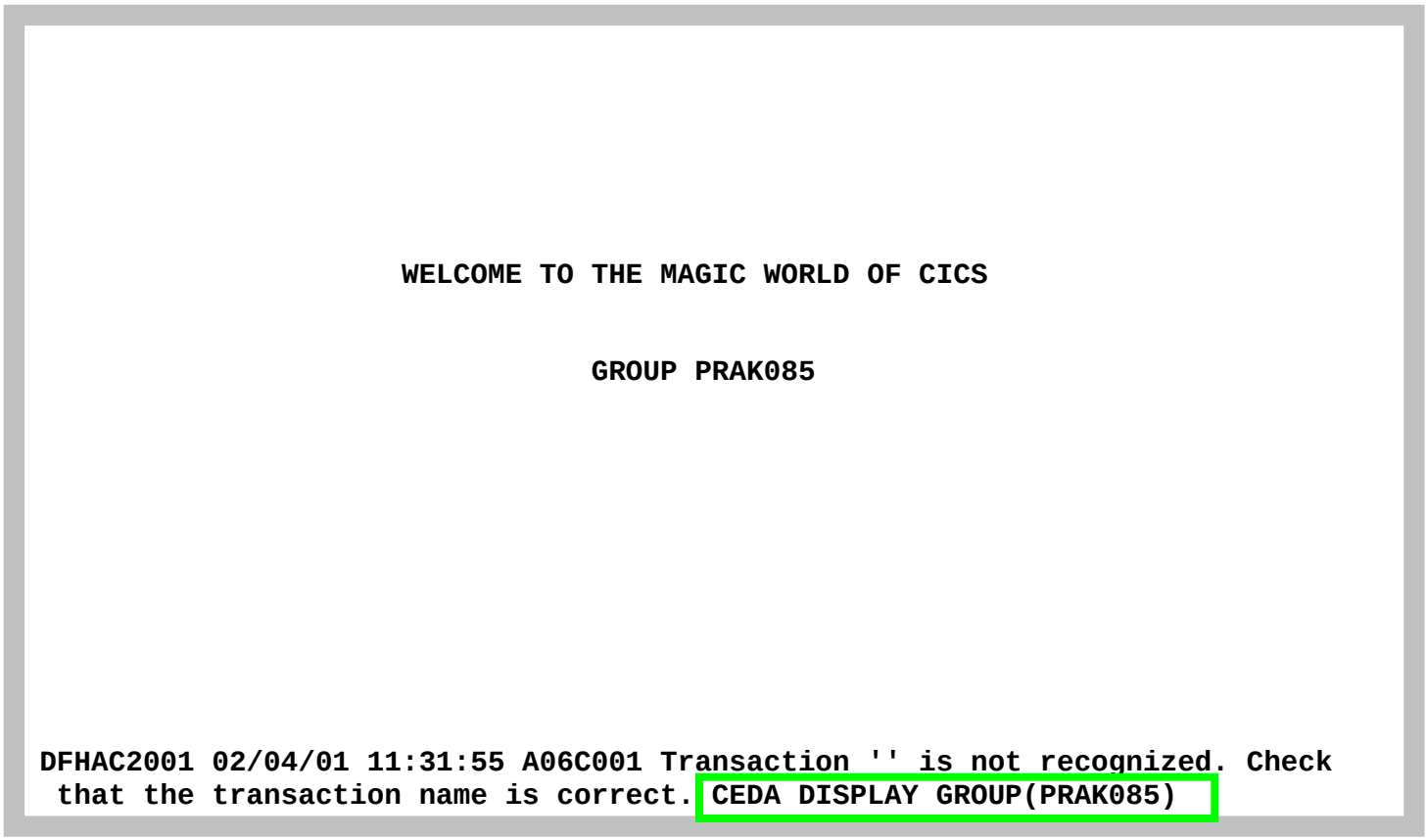


Abbildung 27: Ansicht aller gespeicherter Daten in Group "PRAK085"

Alle Bestandteile unserer Transaktion sind in der Gruppe PRAK085 gespeichert. Wir schauen sie uns an, indem wir den Befehl "CEDA DISPLAY GROUP(PRAK085)" eingeben und anschließend die Eingabetaste drücken (s. Abbildung 27).

```
 DISPLAY GROUP(PRAK085)
 ENTER COMMANDS
  NAME     TYPE          GROUP                                    DATE   TIME
  MSET085  MAPSET        PRAK085                                  01.034 24.00.00
  COB085   PROGRAM       PRAK085                                  01.034 24.00.00
  X085     TRANSACTION   PRAK085                                  01.034 24.00.00
                                                                SYSID=C001 APPLID=A06C001
   RESULTS: 1 TO 3 OF 3                          TIME:  00.00.00  DATE: 01.035
PF 1 HELP       3 END 4 TOP 5 BOT 6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Abbildung 28: Ausgabe aller Komponenten der Gruppe "PRAK085"

Die Gruppe "PRAK085" besteht aus den drei Komponenten "MSET085", "COB085" und "X085", die wir unter CICS definiert und anschließend installiert haben.

Aufgabe: ***Installieren Sie eine CICS-Transaktion "U<Login-Nr.>, die eine Ausgabe ähnlich der***

Abbildung 29 liefert und führen Sie diese dann aus. (Hinweis: Die Ausführung der beiden JCL-Scripte erfolgt fehlerfrei, wenn diese mit der Statusmeldung "MAXCC=0" beendet werden). Die Ausgabe Ihrer Transaktion muss unbedingt den Transaktionsnamen, "TUTORIAL 3 IN COBOL" sowie den Namen oder die Namen der Autoren enthalten. Benutzen Sie als CICS-Gruppennamen Ihren z/OS-Login-Namen (z.B. "PRAK085" oder "PRAK100").
Ersetzen Sie in den Bezeichnern für Ihren Mapset, Ihren Map sowie Ihr Cobol-Programm die Ziffern "085" durch die Nummer Ihres Prakt<xx> oder PRAK<xxx>-Accounts. Wenn Sie das nicht tun und mehrere Teilnehmer dieses Tutorial gleichzeitig bearbeiten, kommt es zu sehr unschönen Effekten, wie z.B. dass Sie den Screen von jemand anderem anstelle Ihres eigenen als Ergebnisscreen erhalten.

Aufgabe: ***Erzeugen Sie einen Screenshot Ihres Fensters, welches die Bildschirmausgabe von CICS enthält (Ihre Version der***

Abbildung 29). Achten Sie darauf, dass das Bild nicht mehr als 250 KByte Speicherplatz belegt. Sehr gut ist das JPEG-Format, das mit weniger als 90 KByte auskommt. Schicken Sie Ihrem Tutor dieses Bild im Bitmap- oder JPEG-Format zu. Die Daten Ihrer Arbeit dürfen nicht gelöscht werden, damit der Tutor Ihre Transaktion aufrufen kann.

Aufgabe: ***Gehen Sie vom CUSTOMPAC MASTER APPLICATION MENU aus mittels SD.ST ins SDSF. Es erscheint eine Liste mit Logfiles. Löschen Sie alle Logfiles in der Print Queue, um so Server-Ressourcen wieder freizugeben. Einen Job dürfen Sie dort allerdings nicht killen: den in der Execution Queue. Denn das ist der Job, durch den Sie jetzt gerade eingeloggt sind. Für jedes Logfile, dass Sie löschen möchten, geben Sie "p" (purge) in die Spalte NP sowie die entsprechende Zeile ein. Mit der Eingabetaste ist jetzt das Löschen eines jeden Logfiles noch einmal zu bestätigen. Möchten Sie mehrere Logfiles auf einmal im Block löschen, so ist das durch Eingabe von "//p" in die Spalte NP (erstes zu löschendes Logfile), die Eingabe von "//" in die Spalte NP (letztes zu löschendes Logfile) sowie Enter möglich. Ein bestätigendes zweites Enter löscht nun den gesamten ausgewählten Logfile-Block.***

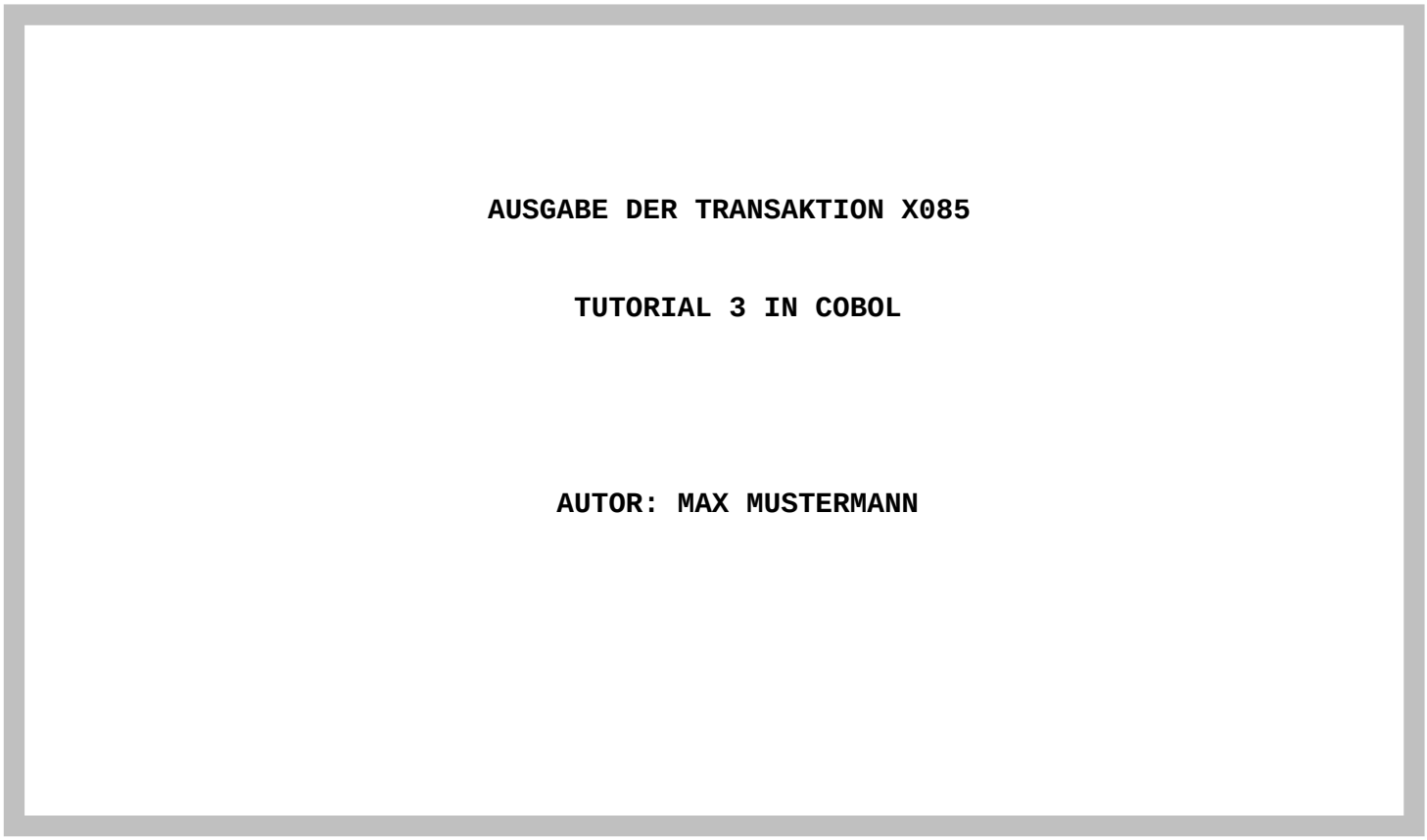


Abbildung 29: Ausgabe der Transaktion X085 der obigen Aufgabe

6. Logoff

Um uns aus CICS auszuloggen, betätigen wir so oft die Taste F3, bis die Meldung "STATUS: SESSION ENDED" ausgegeben wird. In die Zeile darüber geben wir die Logoff-Transaktion "CESF LOGOFF", gefolgt von der Eingabetaste, ein (s. Abbildung 30).

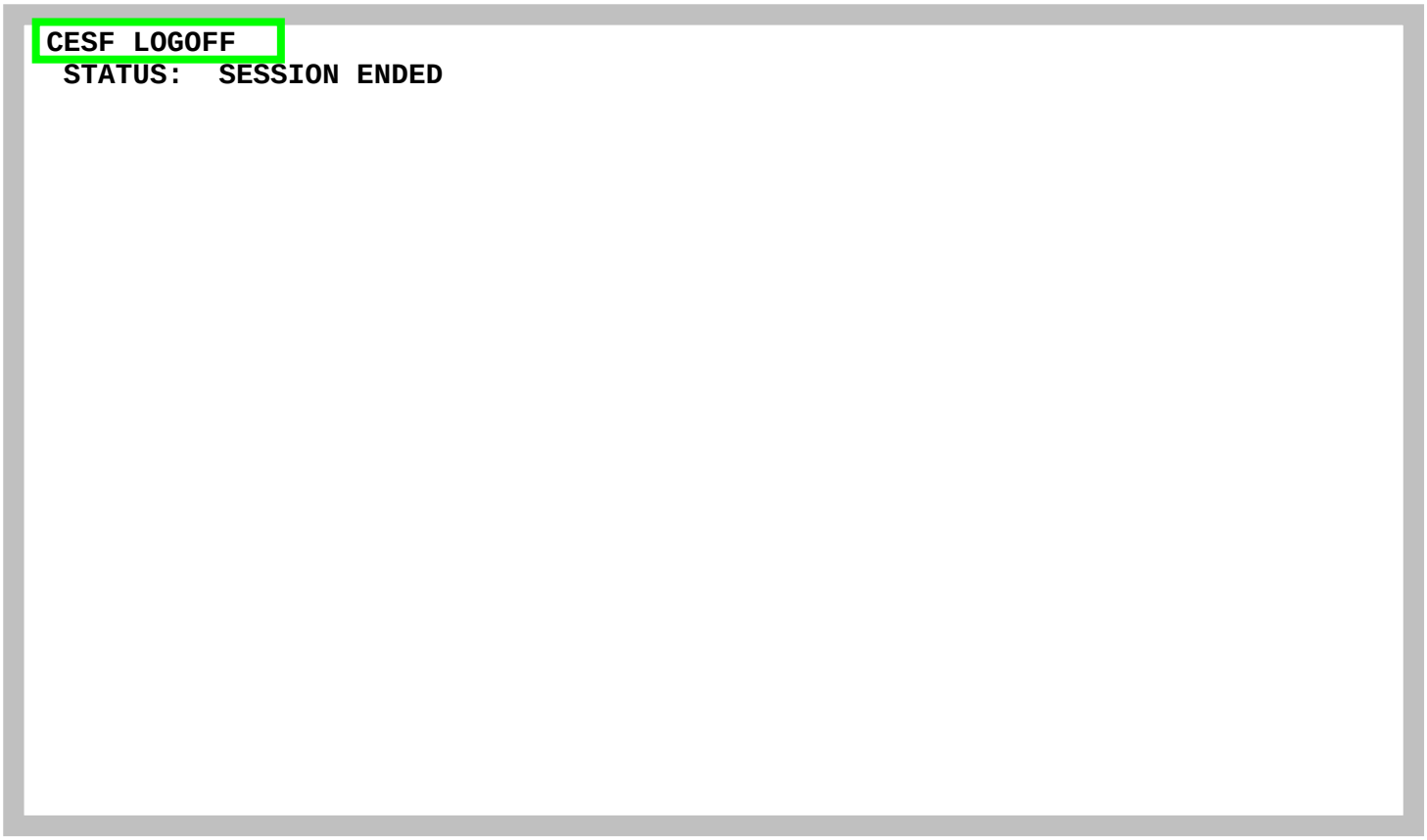


Abbildung 30: Ausloggen aus CICS

Die CICS Shell besteht neben CEDA aus aus einer Reihe weiterer Transaktionen. Wenn Sie CICS auf einem z/OS Rechner installieren, werden diese Transaktionen automatisch mit installiert. CEDA, CESSN und CESF sind wichtig für Sie; den Rest werden Sie selten brauchen.

CBAM	**Business Transaction Server browser**
CDBC	**database control menu**
CDBI	**database control inquiry**
CDBM	**database control interface**
CEBR	**temporary storage browse**
CEBT	**master terminal (alternate CICS)**
CECI	**command-level interpreter**
CECS	**command-level syntax-checking transaction**
CEDA	**resource definition online**
CEDF	**and**
CEDX	**the execution diagnostic facility**
CEMT	**master terminal**
CEOT	**terminal status**
CESF	**sign off**
CESN	**sign on**
CEST	**supervisory terminal**
CETR	**trace control**
CIND	**in-doubt testing tool**
CLER	**Language Environment run-time options**
CMAC	**messages and codes display**
CMSG	**message switching**
CRTE	**remote transactions**
CSFE	**terminal and system test**
CSPG	**page retrieval**
CWTO	**write to operator**
DSNC	**CICS DB2 transaction**

CICS Cobol Tutorial 2
Anlegen einer DB2 Datenbank

Übersicht

Ziel dieses Tutorials ist es, eine z/OS - DB2 relationale Datenbank zu erzeugen und auszulesen. In dem hier beschriebenen Schritt wird von uns die Datenbank mit einer einzigen einfachen Tabelle angelegt und anschließend mit einigen wenigen Daten gefüllt.

Die Aufgabe besteht aus den folgenden Schritten

1. **DB2**
2. **Anlegen benötigter Datasets**
3. **Einloggen ins z/OS DB2**
4. **Einstellen des Subsystem identifiers (SSIDs)**
5. **Starten von SPUFI**
6. **Überblick über die vier vorzunehmenden Definitionen**
    - **6,1 DB2**
    - **6.2 DB2 Storage Management**
    - **6.3 Definitionen**
    - **6.4 Tabellenform**
7. **Definition des Speicherplatzes für Datenbanken**
8. **Anlegen einer Datenbank**
9. **Defininition von Tablespace für DB2-Tabellen**
10. **Erstellen der Tabelle**
11. **Datensätze in die Tabelle einfügen**
12. **Ansehen sämtlicher Datensätze der Tabelle**

<u>Aufgabe:</u> ***Beschäftigen Sie sich mit diesem Tutorial und lösen Sie exakt und gewissenhaft die kursiv geschriebenen und umrahmten Aufgaben.***

1. DB2

Das Datenbankmanagementsystem DB2 existiert in zwei unterschiedlichen Implementierungen, die jedoch weitestgehend kompatibel miteinander sind:

- **DB2 for z/OS, (frühere Bezeichnungen: DB2 UDB for z/OS)**
- **DB2 UDB for Linux, UNIX and Windows**

DB2 Express für Windows und Linux ist eine kostenlose (nicht open Source) Version, welche den gleichen Code wie DB2 UDB benutzt.

DB2 Everyplace ist eine slimmed-down, embedded Version von DB2 UDB für Cellphones, PDAs und ähnliche Devices.

DB2 for z/OS verfügt über zahlreiche Funktionen, die unter DB2 UDB nicht verfügbar sind. Unterstützung für Sysplex, Coupling Facility, Workload Manager sind nur einige Beispiele.

DB2 verwaltet Daten in Tables und speichert sie in Tablespaces. DB2 unterstützt neben den Standard-SQL-Datentypen auch binäre Datentypen (Text, Töne, Bilder, Videos, XML-Daten). Zusammen mit Oracle Database und MS SQL-Server gehört DB2 zu den Datenbanksystemen mit den größten Marktanteilen.

Der Datenzugriff der Anwendungsschicht erfolgt mit SQL, das weitgehend dem ANSI-SQL Standard entspricht. Zugriffe auf gespeicherte Daten können daher aus vielen Programmiersprachen heraus mit eingebettetem SQL erfolgen. Darüber hinaus hat DB2 APIs für REXX, PL/I, COBOL, RPG, FORTRAN, C/C++, Delphi, .NET CLI, Java, andere.

Um bei der Ausführung von DB-Zugriffen optimale Performance zu erzielen, wird ein sog. Optimizer eingesetzt, ein regelbasiertes Expertensystem, welches bei der Programmvorbereitung den Zugriff auf die betreffenden Tabellen festlegt. Dies beruht unter anderem auf den sogenannten Tabellenstatistiken, die mittels des Tools „RUNSTATS“ periodisch aktualisiert werden können, berücksichtigt aber auch andere Kennwerte wie z. B. die Anzahl der CPUs, den Systemzustand, die verfügbare Speichermenge und die physische Verteilung der Daten.

2. Anlegen benötigter Datasets

Wir loggen uns als TSO-Benutzer ein und wählen aus dem "ISPF Primary Option Menu“ die Option "3" (Utilities) aus. Anschließend gehen wir mittels "2" zum "Data Set Utility" (s. Abbildung 1) .

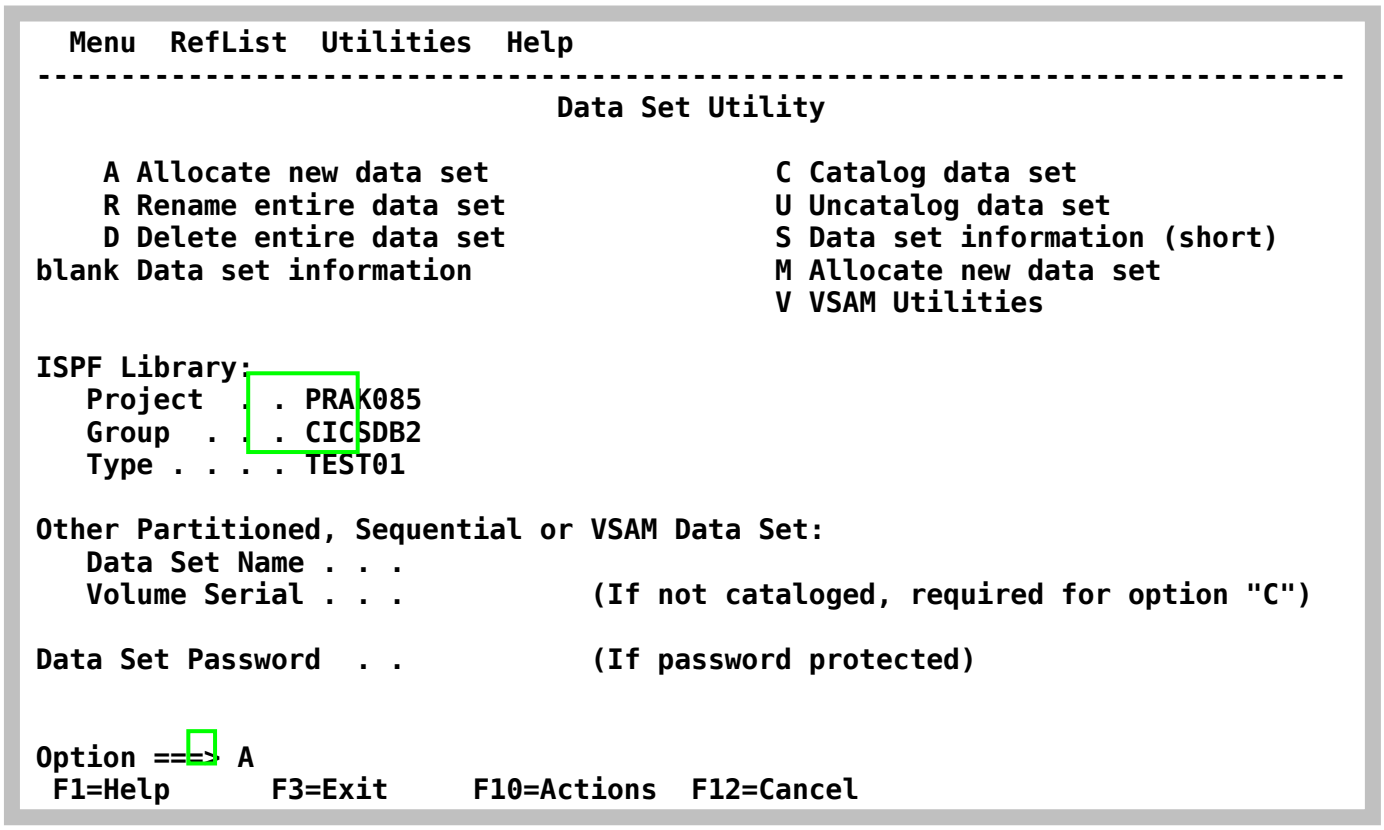


Abbildung 1: "Data Set Utility"

Wir legen für den Benutzer "PRAK085" drei neue Partitioned DataSets (PDS) an:

PRAK085.CICSDB2.TEST01 (s. Abbildung 1)
PRAK085.SPUFI.IN
PRAK085.DBRMLIB.DATA

Wir verwenden dazu die in der Abbildung 2 angegebenen Parameter.

Die Member von "PRAK085.CICSDB2.TEST01" nehmen das von uns zu erstellende C-Programm (Tutorial 5), das dazugehörige BMS-Programm und das JCL-Script auf. Der Dataset wird im später zu absolvierenden Tutorial 5 (C-Version) benötigt.

```
  Menu  RefList  Utilities  Help
--------------------------------------------------------------------------------
                            Allocate New Data Set

 Data Set Name  . . . : PRAK085.CICSDB2.TEST01

 Management class . . .                (Blank for default management class)
 Storage class  . . . .                (Blank for default storage class)
  Volume serial . . . .                (Blank for system default volume) **
  Device type . . . . .                (Generic unit or device address) **
 Data class . . . . . .                (Blank for default data class)
  Space units . . . . . KILOBYTE       (BLKS, TRKS, CYLS, KB, MB, BYTES
                                        or RECORDS)
  Average record unit                  (M, K, or U)
  Primary quantity  . . 18             (In above units)
  Secondary quantity    1              (In above units)
  Directory blocks  . . 5              (Zero for sequential data set) *
  Record format . . . . FB
  Record length . . . . 80
  Block size  . . . . . 320
  Data set name type    PDS            (LIBRARY, HFS, PDS, LARGE, BASIC, *
                                        EXTREQ, EXTPREF or blank)
  Expiration date . . .                (YY/MM/DD, YYYY/MM/DD
 Enter "/" to select option             YY.DDD, YYYY.DDD in Julian form
    Allocate Multiple Volumes           DDDD for retention period in days
                                        or blank)

 ( * Specifying LIBRARY may override zero directory block)

 ( ** Only one of these fields may be specified)

 Command ===>
  F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
 F10=Actions  F12=Cancel
4B                                                    SC0TCP11          021/028
```

Abbildung 2: Die Parameter

Die Member von "PRAK085.SPUFI.IN" nehmen DB2-Kommandos auf, deren Ausführung unsere Datenbank irgendwie modifizieren wird. Wir werden DB2-Kommandos zum Anlegen von Datenbanken, Tabellen, Einfügen von Daten etc. kennen lernen.

In dem Partioned Dataset "PRAK085.DBRMLIB.DATA" werden Zwischenergebnisse abgespeichert, welche in Precompile-, Compile- sowie Link-Prozessen anfallen werden.

Für die spätere Bearbeitung des Tutorial 5 benötigen wir außerdem noch einen Partitioned Dataset "PRAK085.LIB". Hier wird angenommen, dass er in der letzten Sitzung angelegt wurde und deshalb bereits existiert. Sollte dies nicht der Fall sein, legen wir diesen unter Nutzung der Parameter laut Abbildung 2 an (s. auch Abbildung 3).

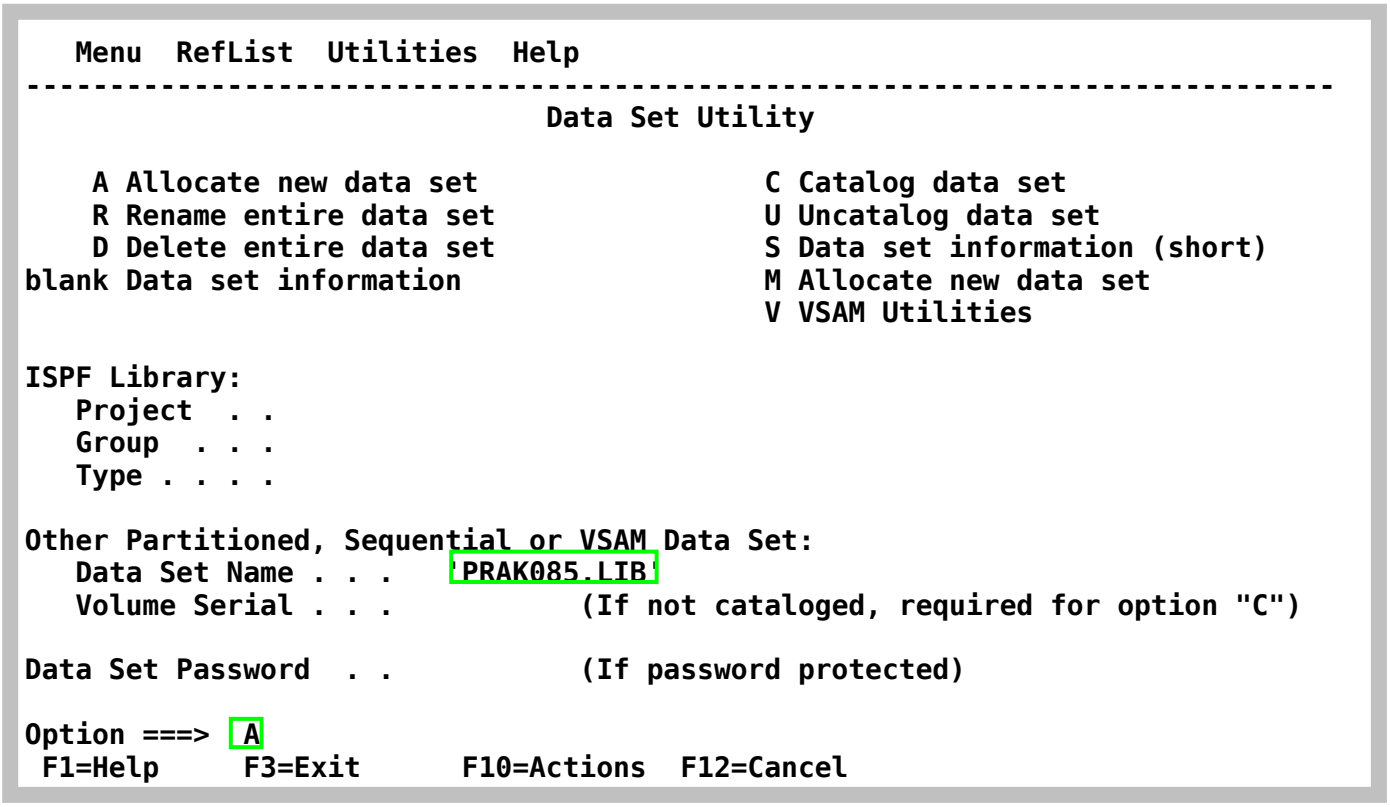


Abbildung 3: Anlegen des Datasets "PRAK085.LIB"

Der Dataset "PRAK085.LIB" soll wie der Dataset "PRAK085.DBRMLIB.DATA" Zwischenergebnisse von Precompile-, Compile- und Link-Schritten aufnehmen.

<u>Aufgabe:</u> Erstellen Sie die benötigten Datasets. Verwenden Sie dazu die in Abbildung 2 angegebenen Parameter.

Wir betätigen zweimal die F3-Taste und kehren so in das "ISPF Primary Option Menu“-Panel zurück.

3. Einloggen ins z/OS DB2

Mit der Eingabe "M" für die Option "More" und danach "11" für die Option "DB2 V8" rufen wir das z/OS-DB2-Subsystem auf, das uns das Anlegen einer neuen DB2-Datenbank sowie deren Modifikation ermöglicht.

```
DB2I PRIMARY OPTION MENU          SSID:
 COMMAND ===> d

 Select one of the following DB2 functions and press ENTER.

  1  SPUFI                   (Process SQL statements)
  2  DCLGEN                  (Generate SQL and source language declarations)
  3  PROGRAM PREPARATION     (Prepare a DB2 application program to run)
  4  PRECOMPILE              (Invoke DB2 precompiler)
  5  BIND/REBIND/FREE        (BIND, REBIND, or FREE plans or packages)
  6  RUN                     (RUN an SQL program)
  7  DB2 COMMANDS            (Issue DB2 commands)
  8  UTILITIES               (Invoke DB2 utilities)
  D  DB2I DEFAULTS           (Set global parameters)
  Q  QMF                     (Query Management Facility
  X  EXIT                    (Leave DB2I)




 PRESS:                      END to exit      HELP for more information




  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
4B                                                    SC0TCP26          002/015
```

Abbildung 4: das "DB2I PRIMARY OPTION MENU"

Es erscheint das "DB2I PRIMARY OPTION MENU"-Panel (s. Abbildung 4).

4. Einstellen des Subsystem Identifiers (SSIDs)

Ehe wir mit dem Anlegen der Datenbank beginnen, müssen wir die "Subsystem Identifier" (SSID) setzen. Dies ist nur ein einziges Mal erforderlich, und zwar beim erstmaligen Gebrauch des DB2-Subsystems. Loggt man sich wiederholt in das DB2-Subsystem ein, ist das Einstellen der SSID nicht mehr erforderlich.

Ein SSID ist eine Datenbankbezeichnung, die systemintern benutzt wird. Auf unserer Maschine "leia.informatik.uni-leipzig.de" ist dies die Bezeichnung "D931". Auf einem anderen z/OS-Rechner kann die Bezeichnung anders sein.

Wir geben "d" ein und bestätigen anschließend mit der Eingabetaste.

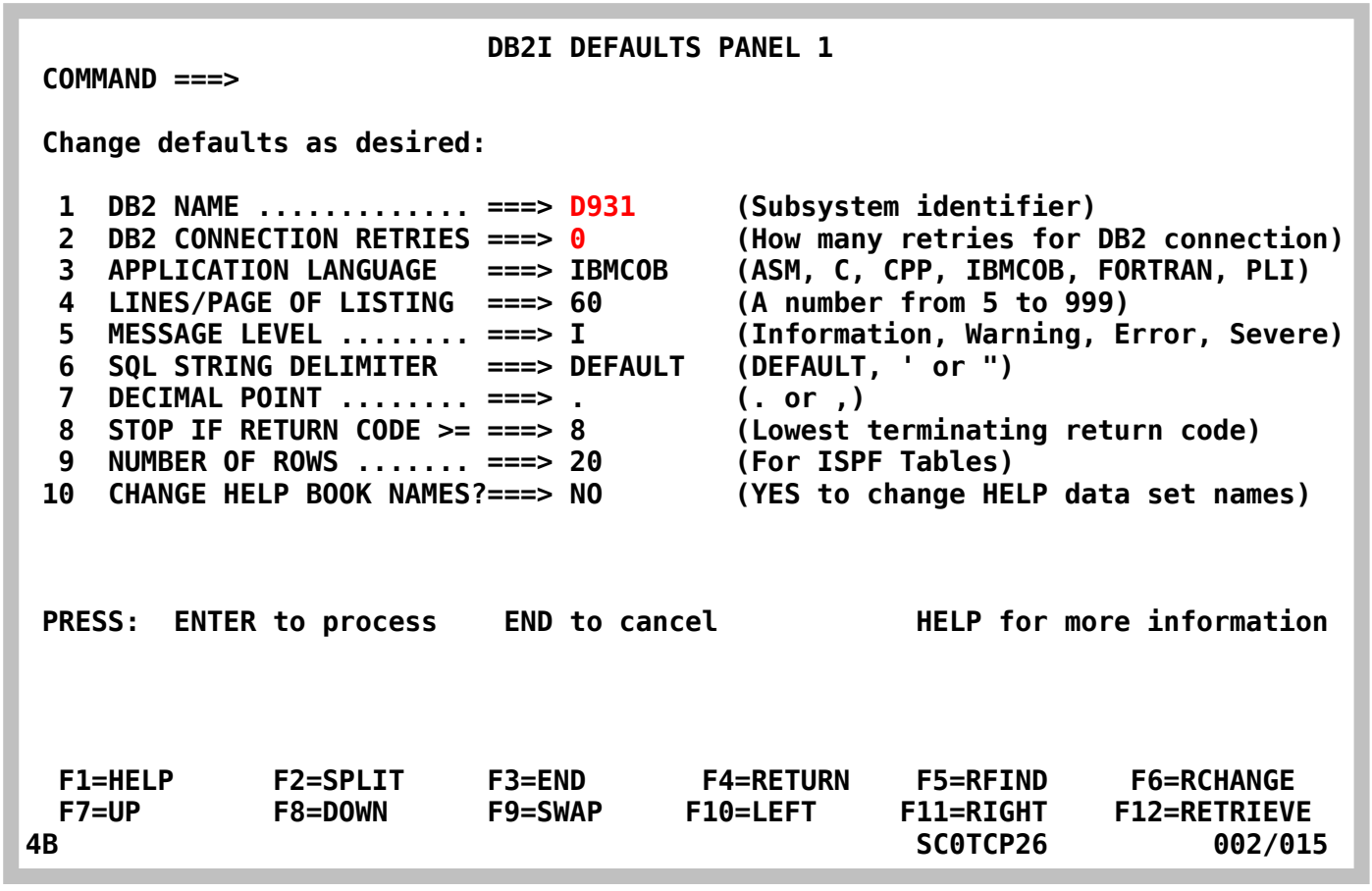


Abbildung 5: Datenbankbezeichnung

Wir geben die beiden gekennzeichneten Werte ein, wenn sie nicht schon da stehen, lassen den Rest unverändert und bestätigen mit der Eingabetaste (s. Abbildung 5).

```
 DB2I DEFAULTS PANEL 2
COMMAND ===>

Change defaults as desired:

 1  DB2I JOB STATEMENT:   (Optional if your site has a SUBMIT exit)
    ===> //PRAK085A JOB (ACCOUNT),'NAME'
    ===> //*
    ===> //*
    ===> //*

   COBOL DEFAULTS:                          (For COBOL, COB2, or IBMCOB)
 2  COBOL STRING DELIMITER ===> DEFAULT   (DEFAULT, ' or ")
 3  DBCS SYMBOL FOR DCLGEN ===> G         (G/N - Character in PIC clause)




PRESS: ENTER to process    END to cancel       HELP for more information

 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Abbildung 6: DB2I DEFAULTS PANEL 2

Das Panel in der Abbildung 6 bedarf keiner Änderungen, wir drücken lediglich die Eingabetaste.

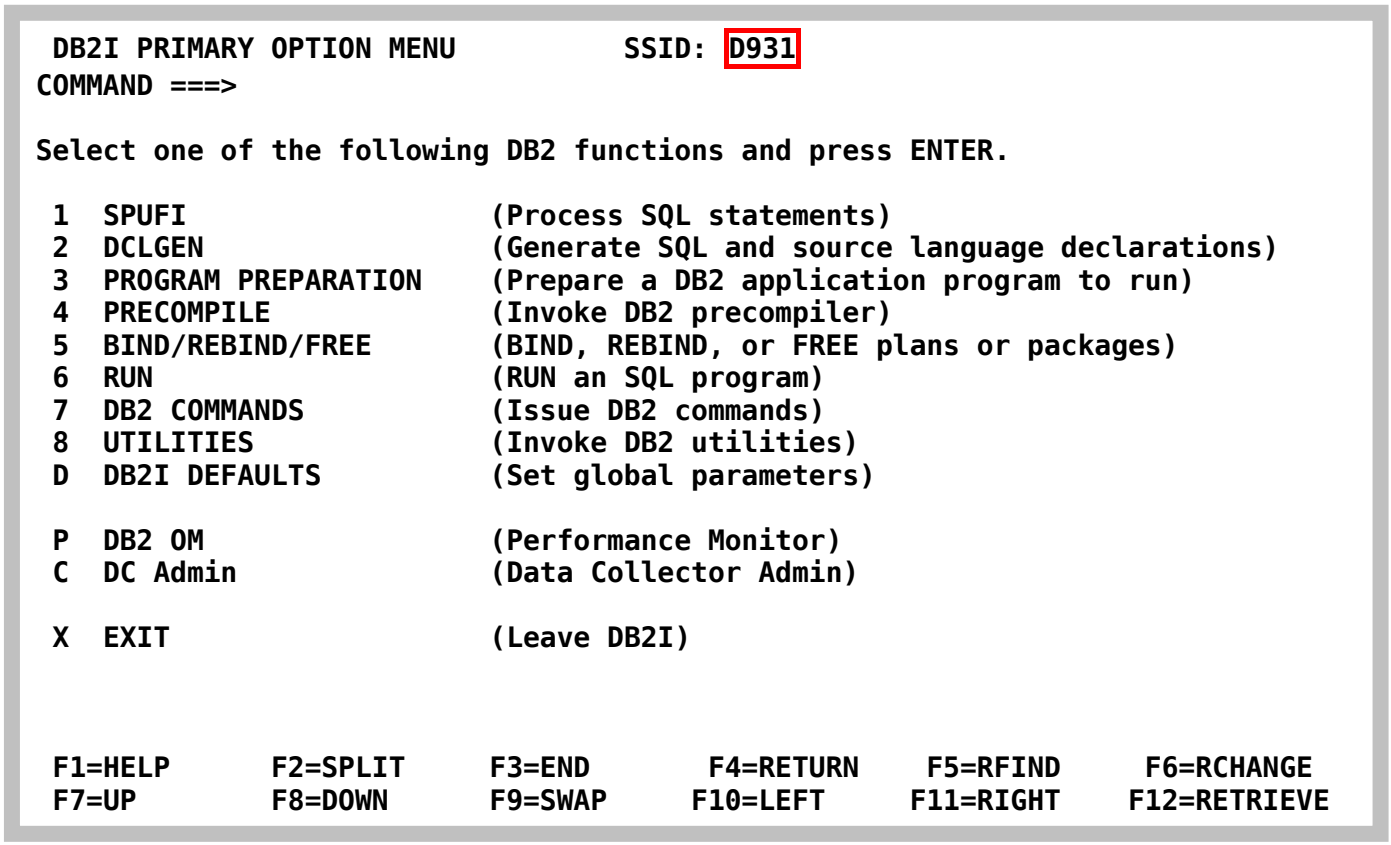


Abbildung 7: "DB2I PRIMARY OPTION MENU"

Das "DB2I PRIMARY OPTION MENU"-Panel erscheint wieder. Als SSID ist jetzt "D931" eingetragen (s. Abbildung 7).

5. Starten von SPUFI

Es existieren zahlreiche Möglichkeiten, eine DB2 Datenbank anzulegen. Wir verwenden hier das DB2-Subsystem "SPUFI".

Unter z/OS ist SPUFI (SQL Processing Using File Input) ein Werkzeug, das es ermöglicht, interaktiv Datenbankstrukturen zu ändern, SQL-Statements zu editieren, auszuführen und die Ergebnisse aufbereitet anzuzeigen. Ansonsten bezeichnet SPUFI ein SQL-Statement, welches nicht in ein Programm eingebettet, sondern über eine Datei oder eine manuelle Eingabe abgesetzt wird.

Wir geben eine "1" ein (`COMMAND ===> 1`), betätigen die Eingabetaste und erhalten den folgenden Screen.

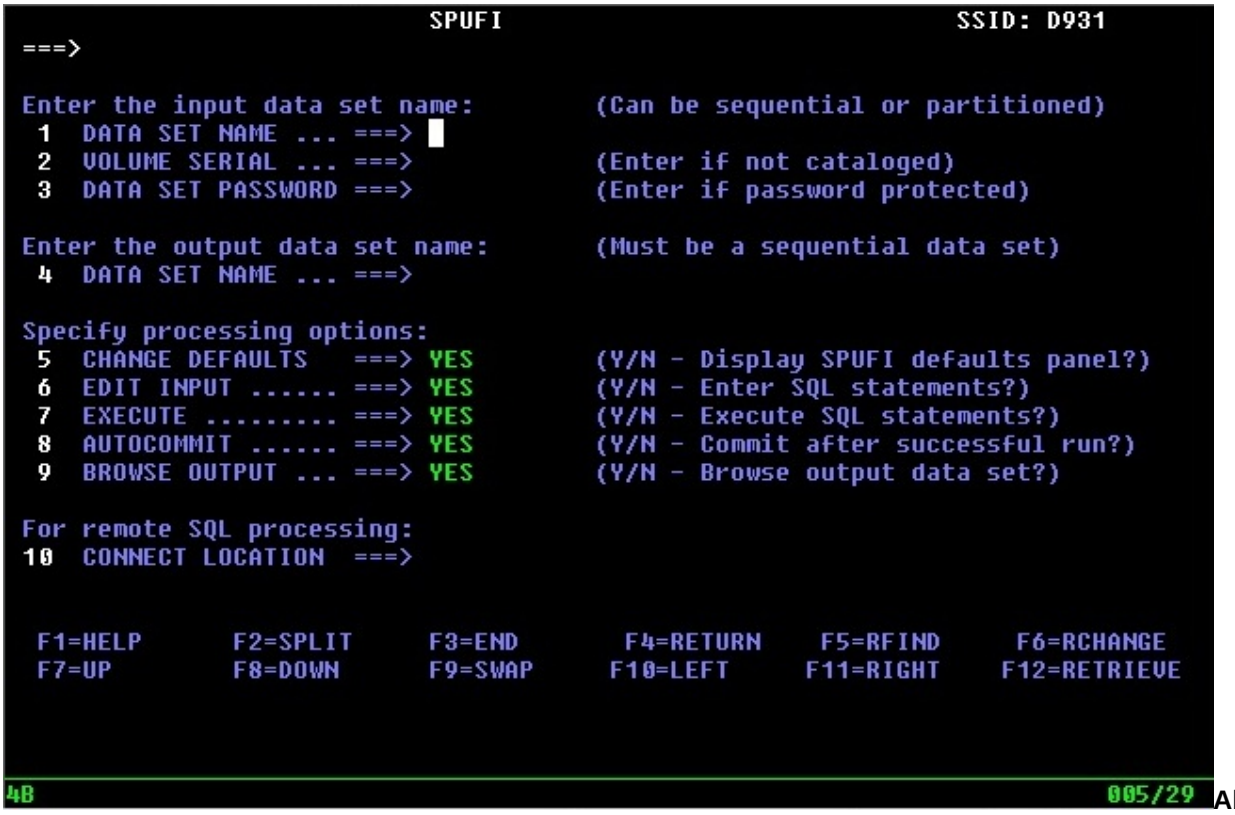
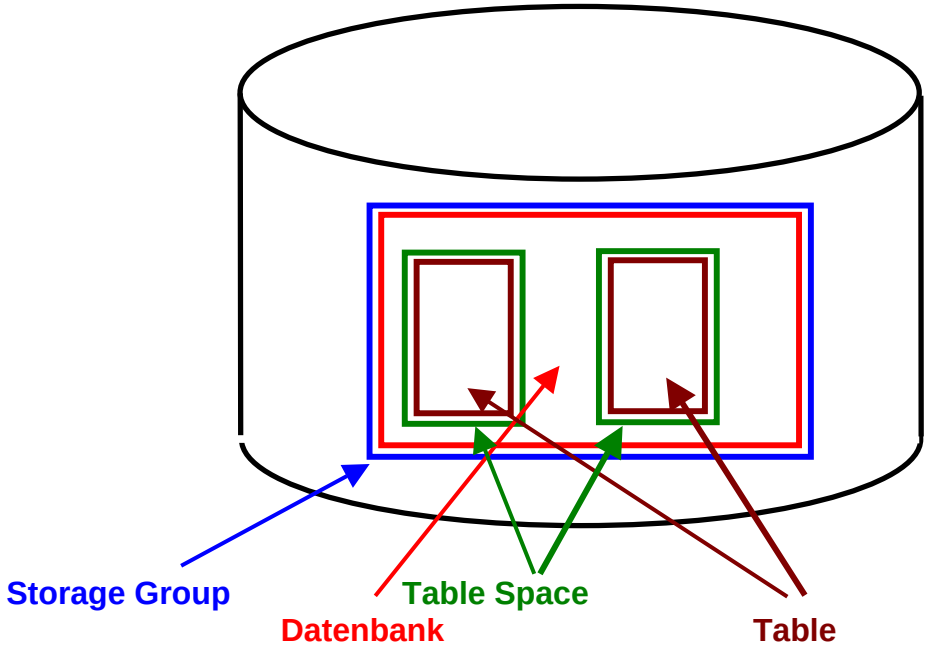


Abbildung 8: SPUFI-Panel

6. Überblick über die vier vorzunehmenden Definitionen

6.1 DB2 Spaces

Für die Anlage einer Datenbank brauchen wir Definitionen, die eine Aussage über das "was", "wie" und "wo" machen. Aus der Sicht des Benutzers besteht eine DB2 Datenbank aus mindestens einer, meistens aber mehreren Tabellen.



Auf einem Plattenspeicher wird für die Datenbank Platz in der Form einer Storage Group definiert und angelegt. Diese nimmt eine DB2 Datenbank auf.

Innerhalb der Storage Group werden eine oder mehrere Table Spaces angelegt. Jeder Table Space nimmt eine DB2-Tabelle auf.

Als Beispiel sei eine DB2 Großrechnerinstallation erwähnt, die mehr als 50 000 Tabellen oder Indices für ERP/CRM Anwendungen verwendet.

6.2 DB2 Storage Management

Storage Management System (SMS) ist die z/OS Komponente, welche für die Plattenspeicherplatz-Verwaltung zuständig ist. Data Sets werden von SMS generell in Data Classes, Storage Classes und Management Classes zusammengefasst. Data Sets mit identischen Data Classes, Storage Classes und Management Class Eigenschaften werden von SMS zu einer Storage Group zusammen gefasst. Eine Storage Group ist eine Gruppe von Plattenspeicher Volumes, die SMS-verwaltete Data Sets enthält.

Eine DB2 Storage Group ist ebenfalls eine derartige SMS Storage Group und besteht aus einzelnen Data Sets.

DB2 speichert seine Daten in VSAM Linear Data Sets (LDS). Jeder DB2 Table Space oder Index Space erfordert wenigstens einen (möglicherweise mehrere) VSAM Data Sets. DB2 benutzt den VSAM Media Manager für seine I/O Operationen. Für jede I/O Anforderung erstellt der VSAM Media Manager ein Kanalprogramm und sendet eine Request an den z/OS I/O Supervisor.

Solutions Journal: DB2 and Storage Management
http://www.craigsmullins.com/DB2%20and%20Storage%20Management.pdf

6.3 Definitionen

Im Einzelnen werden nun vier Definitionen erstellt, die in vier Members abgespeichert werden. Diese Definitionen enthalten:

1. **Art, Ort (Bereich auf einem von mehreren Plattenspeichern), Größe und Eigenschaften des Speicherplatzes, der unsere Datenbank aufnehmen soll. Dieser Speicherplatz wird als Storage Group (STOGROUP) bezeichnet und erhält einen symbolischen Namen ("SYSDEFLT" in dem vorliegenden Beispiel), siehe Abbildung 12.**

2. **Für den symbolischen Namen der Datenbank wird der Name "PRAK085" gewählt.**

   Eine Datenbank speichert (cached) normalerweise einen Teil der aktiven Daten innerhalb des Hauptspeichers temporär ab. Dieser Cache wird allgemein als "Bufferpool" bezeichnet und erhält ebenfalls einen symbolischen Namen ("BP0" in unserem Beispiel), siehe Abbildung 17.

3. **Eine relationale Datenbank besteht aus mindestens einer, meistens aber aus mehreren Tabellen (Relationen). Für jede Tabelle muss Speicherplatz (als Tablespace bezeichnet) reserviert werden. Dieser erhält ebenfalls einen symbolischen Namen (hier "TABSP085"). In unserem einfachen Beispiel legen wir nur eine einzige Tabelle an, benötigen also auch nur einen Tablespace (siehe Abbildung 20).**

4. **Schließlich muss die Tabelle selbst bezüglich ihres Namens ("TAB085"), ihrer Struktur und der Bezeichnung ihrer Felder (Spalten) definiert werden (siehe Abbildung 23).**

Für die vier Definitionen erstellen wir je einen Member. Die folgende Tabelle enthält eine Übersicht über die Membernamen, die Aufgaben und die symbolischen Namen. Hierbei unterscheiden wir zwischen

- **den symbolischen Namen der Datenbank, der Tabelle und des jeweiligen zugeordneten Speicherplatzes (rechte Seite obiger Tabelle),**
- **den Namen der Member von "PRAK085.SPUFI.IN", die diese Definitionen aufnehmen (linke Seite obiger Tabelle).**

Symbolischer Name und Membername können, müssen aber nicht identisch sein.

SPUFI interner Member Name	Aufgabe	symbolischer Name
STOGR1	Speicherplatz für unsere DB2-Datenbank anlegen	SYSDEFLT
DB1	Die Datenbank selbst anlegen	PRAK085
TABSP1	Speicherplatz für eine Tabelle anlegen	TABSP085
TAB1	Die Tabelle selbst anlegen	TAB085

Das Anlegen dieser Definitionen ist in den Abschnitten 6 bis 9 beschrieben.

Problem:

Diese Definitionen können teilweise nur mit bestimmten Privilegien erzeugt werden, über welche Ihre Accounts nicht verfügen. Im Falle des Praktikumaccounts sind Storage Group und Database vorgegeben und die Privilegien nur auf den Tablespace und den (die) Table begrenzt . Als Storage Group verwenden Sie die Default-Storage Group "SYSDEFLT".

In den Abschnitten 6, 7 und 8 wird beschrieben, wie ein User mit den entsprechenden Privilegien eine Storage Group und eine Database anlegt. Als Praktikumsteilnehmer können Sie dies nicht tun. Ihr Betreuer hat deshalb diese Schritte für Sie bereits vorgenommen; Sie finden deshalb Storage Group und Datenbank bereits vor. Schauen Sie sich die Vorgehensweise in den Abschnitten 6, 7 und 8 dennoch an.

Sie selbst setzen Ihre Arbeit mit Abschnitt 9 fort.

6.4 Tabellenform

Wir legen für unseren Partitioned Dataset "PRAK085.SPUFI.IN" nur zwei Members an:

Member Name	Aufgabe	symbolischer Name
~~STOGR1~~	~~Speicherplatz für unsere DB2-Datenbank anlegen~~	~~SYSDEFLT~~
~~DB1~~	~~Die Datenbank selbst anlegen~~	~~PRAK085~~
TABSP1	Speicherplatz für eine Tabelle anlegen	TABSP085
TAB1	Die Tabelle selbst anlegen	TAB085

Die von Ihnen anzulegende Tabelle soll über 2 Spalten verfügen und das folgende Format haben:

VNAME	NNAME
.....	.....
.....	.....
.....	.....
.....	.....

7. Definition des Speicherplatzes für Datenbanken

Wir beginnen mit der Definition des Speicherplatzes (Storage Group) für unsere Datenbank. Hierfür wird, wie bereits beschrieben, das entsprechende Privileg benötigt. Praktikumsteilnehmer besitzen dies nicht.

```
 SPUFI                                SSID: D931
===>

 Enter the input data set name:        (Can be sequential or partitioned)
  1  DATA SET NAME ... ===> SPUFI.IN(STOGR1)
  2  VOLUME SERIAL ... ===>             (Enter if not cataloged)
  3  DATA SET PASSWORD ===>             (Enter if password protected)

 Enter the output data set name:       (Must be a sequential data set)
  4  DATA SET NAME ... ===> SPUFI.OUT

 Specify processing options:
  5  CHANGE DEFAULTS   ===> YES        (Y/N - Display SPUFI defaults panel?)
  6  EDIT INPUT ...... ===> YES        (Y/N - Enter SQL statements?)
  7  EXECUTE ......... ===> YES        (Y/N - Execute SQL statements?)
  8  AUTOCOMMIT ...... ===> YES        (Y/N - Commit after successful run?)
  9  BROWSE OUTPUT ... ===> YES        (Y/N - Browse output data set?)

 For remote SQL processing:
10    CONNECT LOCATION  ===>

PRESS: ENTER to process         END to exit      HELP for more information


  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Abbildung 9: Definition für die Storage Group

Unsere Eingabe soll in dem Member "PRAK085.SPUFI.IN(STOGR1)" gespeichert werden. Diese Eingabe wird von SPUFI übersetzt.

Für die Ausgabe der Übersetzung wird ein neuer Dataset benötigt. Wir nennen ihn PRAK085.SPUFI.OUT. Er wird automatisch angelegt.

Nachdem wir die in Abbildung 9 gekennzeichneten Änderungen vorgenommen haben, drücken wir die Eingabetaste.

Was jetzt geschieht ist die Eingabe und Übersetzung eines SQL Statements. Normalerweise sind SQL Statements Bestandteil eines Anwendungsprogramms und werden als Teil des Anwendungsprogramms übersetzt und ausgeführt. SPUFI gibt uns eine Möglichkeit, ein SQL Statement auch außerhalb eines Anwendungsprogramms zu übersetzen und auszuführen.

```
    CURRENT SPUFI DEFAULTS             SSID: D931
 ===>

 Enter the following to control your SPUFI session:
  1  SQL TERMINATOR .. ===> ;          (SQL Statement Terminator)
  2  ISOLATION LEVEL   ===> RR         (RR=Repeatable Read, CS=Cursor Stability)
  3  MAX SELECT LINES  ===> 250        (Maximum number of lines to be
                                          returned from a SELECT)
  4  ALLOW SQL WARNINGS===> NO         (Continue fetching after sqlwarning)
  5  CHANGE PLAN NAMES ===> NO         (Change the plan names used by SPUFI)

 Output data set characteristics:
  6  RECORD LENGTH ... ===> 4092       (LRECL=Logical record length)
  7  BLOCK SIZE ...... ===> 4096       (Size of one block)
  8  RECORD FORMAT ... ===> VB         (RECFM=F, FB, FBA, V, VB, or VBA)
  9  DEVICE TYPE ..... ===> SYSDA      (Must be DASD unit name)

 Output format characteristics:
 10  MAX NUMERIC FIELD ===> 33         (Maximum width for numeric fields)
 11  MAX CHAR FIELD .. ===> 80         (Maximum width for character fields)
 12  COLUMN HEADING .. ===> NAMES      (NAMES, LABELS, ANY or BOTH)

 PRESS:  ENTER to process    END to exit              HELP for more information



  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
4B                                                    SC0TCP06          002/007
```

Abbildung 10: Dataset-Parameter

In diesem Screen (Abbildung 10) werden Dataset-Parameter angezeigt. Wir übernehmen hier (und auch in Zukunft für alle weiteren SPUFI Definitionen) alle Default-Werte ohne Änderung, indem wir mit der Eingabetaste bestätigen.

Es erscheint ein (leerer) Edit-Screen (s. Abbildung 11):

```
  File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       PRAK085.SPUFI.IN(STOGR1) - 01.00               Columns 00001 00072
****** ***************************** Top of Data ****************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
Command ===>                                                Scroll ===> PAGE
 F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
```

Abbildung 11: der Edit-Screen

Wir benutzen ihn, um den Speicherplatz für unsere Datenbank zu definieren (s. Abbildung 12). Die Angaben für "VOLUMES" und "VCAT" sind installationsspezifisch und müssen vom Systemadministrator vorgegeben werden.

```
  File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       PRAK085.SPUFI.IN(STOGR1) - 01.00               Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
'''''' CREATE STOGROUP SYSDEFLT
''''''    VOLUMES (SCPMV5)
''''''    VCAT DSN510;
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
Command ===>                                                 Scroll ===> PAGE
 F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
```

Abbildung 12: "SYSDEFLT"

Nach Eingabe der Zeilen (s. Abbildung 12) wird dieser Screen mit der F3-Taste verlassen.

```
 SPUFI                                SSID: D931
===>

Enter the input data set name:         (Can be sequential or partitioned)
 1  DATA SET NAME ... ===> SPUFI.IN(STOGR1)
 2  VOLUME SERIAL ... ===>             (Enter if not cataloged)
 3  DATA SET PASSWORD ===>             (Enter if password protected)

Enter the output data set name:        (Must be a sequential data set)
 4  DATA SET NAME ... ===> SPUFI.OUT

Specify processing options:
 5  CHANGE DEFAULTS   ===> *           (Y/N - Display SPUFI defaults panel?)
 6  EDIT INPUT ...... ===> *           (Y/N - Enter SQL statements?)
 7  EXECUTE ......... ===> YES         (Y/N - Execute SQL statements?)
 8  AUTOCOMMIT ...... ===> YES         (Y/N - Commit after successful run?)
 9  BROWSE OUTPUT ... ===> YES         (Y/N - Browse output data set?)

For remote SQL processing:
10  CONNECT LOCATION  ===>

 PRESS: ENTER to process    END to exit     HELP for more information

     DSNE808A EDIT SESSION HAS COMPLETED. PRESS ENTER TO CONTINUE
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Abbildung 13: Bestätigung der beendeten Edit-Session

SPUFI teilt uns mit, dass damit die Edit Session – Erstellen des Members SPUFI.IN(STOGR1) – beendet wurde (s. Abbildung 13). Nach dem Drücken der Eingabetaste wird unsere Definition übersetzt und das Ergebnis in SPUFI.OUT gestellt.

Das Ergebnis der Übersetzung wird mitgeteilt (s. Abbildung 14).

```
   Menu  Utilities  Compilers  Help
 ------------------------------------------------------------------------------
 BROWSE    PRAK085.SPUFI.OUT                        Line 00000000 Col 001 080
******************************** Top of Data ********************************
---------+---------+---------+---------+---------+---------+---------+---------+
CREATE STOGROUP SYSDEFLT                                                 00010000
   VOLUMES (SCPMV5)                                                      00020000
   VCAT DSN510;                                                          00030000
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+---------+
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 1
DSNE621I NUMBER OF INPUT RECORDS READ IS 3
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 15
******************************* Bottom of Data ********************************


 Command ===>                                                Scroll ===> PAGE
  F1=Help    F3=Exit    F5=Rfind  F12=Cancel
```

Abbildung 14: Erfolgreiche Übersetzung

Wird an dieser Stelle eine Fehlermeldung ausgegeben, dann ist der zu erstellende Speicherplatz für unsere Datenbank (Stogroup) möglicherweise bereits vorhanden. In diesem Fall gehen sie wie im Anhang beschrieben vor.

8. Anlegen einer Datenbank

```
 SPUFI                                SSID: D931
===>
DSNE361I SPUFI PROCESSING COMPLETE
Enter the input data set name:        (Can be sequential or partitioned)
 1  DATA SET NAME ... ===> SPUFI.IN(STOGR1)
 2  VOLUME SERIAL ... ===>             (Enter if not cataloged)
 3  DATA SET PASSWORD ===>             (Enter if password protected)

Enter the output data set name:       (Must be a sequential data set)
 4  DATA SET NAME ... ===> SPUFI.OUT

Specify processing options:
 5  CHANGE DEFAULTS   ===> YES        (Y/N - Display SPUFI defaults panel?)
 6  EDIT INPUT ...... ===> YES        (Y/N - Enter SQL statements?)
 7  EXECUTE ......... ===> YES        (Y/N - Execute SQL statements?)
 8  AUTOCOMMIT ...... ===> YES        (Y/N - Commit after successful run?)
 9  BROWSE OUTPUT ... ===> YES        (Y/N - Browse output data set?)

For remote SQL processing:
10  CONNECT LOCATION  ===>


 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Abbildung 15: Der SPUFI-Screen

Wir kehren zum SPUFI-Bildschirm (s. Abbildung 15) zurück.

Als nächstes wird im Speicherplatz "SYSDEFLT" eine Datenbank angelegt.

```
 SPUFI                                SSID: D931
===>
DSNE361I SPUFI PROCESSING COMPLETE
Enter the input data set name:         (Can be sequential or partitioned)
 1  DATA SET NAME ... ===> SPUFI.IN(DB1)
 2  VOLUME SERIAL ... ===>              (Enter if not cataloged)
 3  DATA SET PASSWORD ===>              (Enter if password protected)

Enter the output data set name:        (Must be a sequential data set)
 4  DATA SET NAME ... ===> SPUFI.OUT

Specify processing options:
 5  CHANGE DEFAULTS   ===> YES         (Y/N - Display SPUFI defaults panel?)
 6  EDIT INPUT ...... ===> YES         (Y/N - Enter SQL statements?)
 7  EXECUTE ......... ===> YES         (Y/N - Execute SQL statements?)
 8  AUTOCOMMIT ...... ===> YES         (Y/N - Commit after successful run?)
 9  BROWSE OUTPUT ... ===> YES         (Y/N - Browse output data set?)

For remote SQL processing:
10  CONNECT LOCATION  ===>


 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Abbildung 16: Definition für DB1

Wir geben den in Abbildung 16 gezeigten Dataset- und Member-Namen ein und drücken anschließend die Eingabetaste.

Wir übernehmen im nächsten Screen wieder alle Werte unverändert und bestätigen mit der Eingabetaste.

Wir erstellen eine Datenbank mit dem symbolischen Namen "PRAK085". Sie wird in der vorher angelegten Storage Group "SYSDEFLT" abgespeichert (s. Abbildung 17). Sie benutzt einen Hauptspeicher-Cache (Bufferpool) mit dem symbolischen Namen "BP0".

```
  File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       PRAK085.SPUFI.IN(DB1) - 01.00               Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
'''''' CREATE DATABASE PRAK085
'''''' STOGROUP SYSDEFLT
'''''' BUFFERPOOL BP0;
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
Command ===>                                                 Scroll ===> PAGE
 F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
```

Abbildung 17: Definition der Datenbank "PRAK085"

Wir verlassen diesen Bildschirm mit F3 und betätigen anschließend die Eingabetaste.

```
   Menu  Utilities  Compilers  Help
 ------------------------------------------------------------------------------
 BROWSE    PRAK085.SPUFI.OUT                          Line 00000000 Col 001 080
******************************** Top of Data ********************************
---------+---------+---------+---------+---------+---------+---------+---------+
CREATE DATABASE PRAK085                                                 00010000
STOGROUP SYSDEFLT                                                       00020000
BUFFERPOOL BP0;                                                         00030000
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+---------+
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 1
DSNE621I NUMBER OF INPUT RECORDS READ IS 3
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 15
****************************** Bottom of Data ******************************


 Command ===>                                                Scroll ===> PAGE
  F1=Help    F3=Exit    F5=Rfind  F12=Cancel
```

Abbildung 18: Erfolgsmeldung der Anlage der Datenbank

Auch diese Anlage der Datenbank war erfolgreich (s. Abbildung 18). Mit der F3-Taste kehren wir zum SPUFI-Screen zurück.

Die von uns bisher vorgenommene Reservierung von Speicherplatz für die Datenbank und das Erstellen der Datenbank wird normalerweise vom Systemadministrator vorgenommen, dem einzelnen Praktikumsbenutzer fehlen hierfür in der Regel die Zugriffsrechte. Die weiteren Schritte kann der Benutzer aber selbst vornehmen.

9. Defininition von Tablespace für DB2-Tabellen

<u>Aufgabe:</u> ***Erstellen Sie den Tablespace wie im folgenden Beispiel beschriebene. Dieser soll den Namen "TABSPxxx" erhalten, wobei "xxx" für Ihre dreistellige Nummer Ihres PRAK-Accounts steht.***

In unserem Beispiel soll die DB2-Datenbank aus einer einzigen Tabelle bestehen. Wir erstellen eine Definition des Speicherplatzes (Tablespace) für unsere Tabelle in dem Member "TABSP1" (s. Abbildung 19).

```
      SPUFI                               SSID: D931
===>
DSNE361I SPUFI PROCESSING COMPLETE
Enter the input data set name:         (Can be sequential or partitioned)
 1  DATA SET NAME ... ===> SPUFI.IN(TABSP1)
 2  VOLUME SERIAL ... ===>              (Enter if not cataloged)
 3  DATA SET PASSWORD ===>              (Enter if password protected)

Enter the output data set name:        (Must be a sequential data set)
 4  DATA SET NAME ... ===> SPUFI.OUT

Specify processing options:
 5  CHANGE DEFAULTS   ===> YES         (Y/N - Display SPUFI defaults panel?)
 6  EDIT INPUT ...... ===> YES         (Y/N - Enter SQL statements?)
 7  EXECUTE ......... ===> YES         (Y/N - Execute SQL statements?)
 8  AUTOCOMMIT ...... ===> YES         (Y/N - Commit after successful run?)
 9  BROWSE OUTPUT ... ===> YES         (Y/N - Browse output data set?)

For remote SQL processing:
10  CONNECT LOCATION  ===>


 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Abbildung 19: Member TABSP1 zum Editieren öffnen

Wir drücken zweimal die Eingabetaste, um in den "Edit Entry Panel" zu gelangen.

Wir erstellen einen Tablespace mit dem symbolischen Namen "TABSP085" (s. Abbildung 20). Die darin enthaltene Tabelle gehört zur Datenbank "PRAK085", die wiederum in der Storage Group "SYSDEFLT" untergebracht ist. Der hierzu gehörige Buffer Pool (Hauptspeicher Cache) heißt "BP0".

```
  File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       PRAK085.SPUFI.IN(TABSP1) - 01.00               Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
'''''' CREATE TABLESPACE TABSP085
''''''    IN PRAK085
''''''    USING STOGROUP SYSDEFLT
''''''      PRIQTY 60
''''''      SECQTY 60
''''''      ERASE NO
''''''    BUFFERPOOL BP0
''''''    CLOSE NO;
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
Command ===>                                                 Scroll ===> PAGE
 F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
```

Abbildung 20: Speicherplatz für Tabelle anlegen

Wir verlassen den Bildschirm mit F3 und bestätigen mit der Eingabetaste.

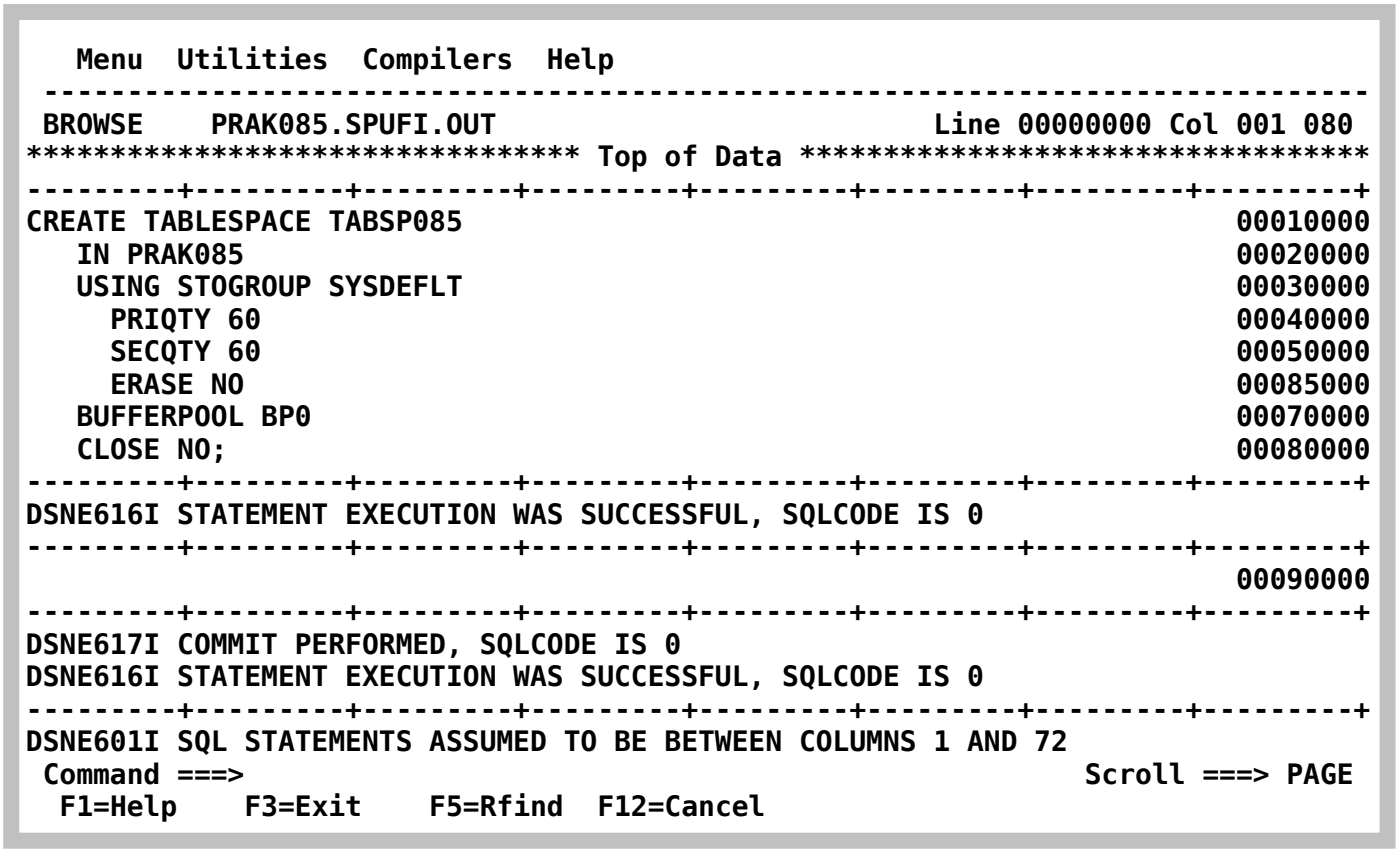


Abbildung 21: Erfolgreiche Anlage des Tablespace

Die Anlage des Tablespaces "TABSP085" war ebenfalls erfolgreich. Durch Drücken der F8-Taste (Scroll Forward) können wir den Rest der Nachricht ansehen. Mit der F7-Taste (Scroll Backward) geht es wieder zurück. Mit der F3-Taste verlassen wir den Screen.

10. Erstellen der Tabelle

Aufgabe: ***Definieren Sie Ihre Tabelle, wie im Folgenden beschrieben. Diese soll den Namen "TABxxx" erhalten, wobei "xxx" für Ihre dreistellige Nummer Ihres PRAK-Accounts steht.***

Wir sind wieder im SPUFI-Panel. In den Member "TAB1" speichern wir die Definition der Tabelle selbst (s. Abbildung 22).

```
 SPUFI                                SSID: D931
===>
DSNE361I SPUFI PROCESSING COMPLETE
Enter the input data set name:         (Can be sequential or partitioned)
 1  DATA SET NAME ... ===> SPUFI.IN(TAB1)
 2  VOLUME SERIAL ... ===>              (Enter if not cataloged)
 3  DATA SET PASSWORD ===>              (Enter if password protected)

Enter the output data set name:        (Must be a sequential data set)
 4  DATA SET NAME ... ===> SPUFI.OUT

Specify processing options:
 5  CHANGE DEFAULTS   ===> YES         (Y/N - Display SPUFI defaults panel?)
 6  EDIT INPUT ...... ===> YES         (Y/N - Enter SQL statements?)
 7  EXECUTE ......... ===> YES         (Y/N - Execute SQL statements?)
 8  AUTOCOMMIT ...... ===> YES         (Y/N - Commit after successful run?)
 9  BROWSE OUTPUT ... ===> YES         (Y/N - Browse output data set?)

For remote SQL processing:
10  CONNECT LOCATION  ===>


 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Abbildung 22: Definition für TAB1

Um weiter zu kommen, drücken wir zweimal die Eingabetaste.

Die Einträge in der Tabelle "TAB085" (s. Abbildung 23) haben zwei Spalten: "VNAME" (zur Aufnahme des Vornamens einer Person) und "NNAME" (zur Aufnahme des Nachnamens einer Person). Die Felder haben eine Länge von je 20 Zeichen. Die Tabelle ist Teil der Datenbank "PRAK085" und verwendet den Tablespace "TABSP085".

```
  File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       PRAK085.SPUFI.IN(TAB1) - 01.00               Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
'''''' CREATE TABLE TAB085
''''''    (
''''''     VNAME CHAR(20) NOT NULL,
''''''     NNAME CHAR(20) NOT NULL
''''''    )
''''''     IN PRAK085.TABSP085;
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
Command ===>                                                Scroll ===> PAGE
 F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
```

Abbildung 23: Anlegen der Tabelle "TAB085"

Wir schreiben die Definition der Tabelle und verlassen den Screen mit F3 und betätigen abschließend die Eingabetaste.

```
   Menu  Utilities  Compilers  Help
 ------------------------------------------------------------------------------
 BROWSE    PRAK085.SPUFI.OUT                        Line 00000000 Col 001 080
******************************** Top of Data ********************************
---------+---------+---------+---------+---------+---------+---------+---------+
CREATE TABLE TAB085                                                     00010000
   (                                                                    00020000
    VNAME CHAR(20) NOT NULL,                                            00030000
    NNAME CHAR(20) NOT NULL                                             00040000
   )                                                                    00050000
    IN PRAK085.TABSP085;                                                00085000
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+---------+
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 1
DSNE621I NUMBER OF INPUT RECORDS READ IS 6
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 18
 Command ===>                                          Scroll ===> PAGE
  F1=Help    F3=Exit    F5=Rfind  F12=Cancel
```

Abbildung 24: Erfolgreiche Definition

Da die Definiton erfolgreich war (s. Abbildung 24), verlassen wir den Screen mit F3.

11. Datensätze in die Tabelle einfügen

Damit haben wir eine Datenbank und Tabelle angelegt. Aber sie ist zur Zeit noch leer. Es existieren zahlreiche Software-Funktionen, um leere Datenbanken und Tabellen mit bereits existierenden Daten zu füllen. Wir machen es hier auf eine einfache Art "zu Fuß".

Aufgabe: ***Füllen Sie Ihre Tabelle, wie im Folgenden beschrieben, mit einigen Namen einschließlich Ihres eigenen Namens. Sollten Sie die Übung zu zweit absolvieren (unter Nutzung eines gemeinsamen PRAK-Accounts), dann müssen Ihre beiden Namen in die Tabelle aufgenommen werden.***

Hierzu legen wir (zusätzlich zu den bisherigen 4 Members) einen weiteren SPUFI-Member "INSERT" an (s. Abbildung 25)…

```
  SPUFI                                SSID: D931
===>

Enter the input data set name:         (Can be sequential or partitioned)
 1  DATA SET NAME ... ===> SPUFI.IN(INSERT)
 2  VOLUME SERIAL ... ===>              (Enter if not cataloged)
 3  DATA SET PASSWORD ===>              (Enter if password protected)

Enter the output data set name:        (Must be a sequential data set)
 4  DATA SET NAME ... ===> SPUFI.OUT

Specify processing options:
 5  CHANGE DEFAULTS   ===> YES         (Y/N - Display SPUFI defaults panel?)
 6  EDIT INPUT ...... ===> YES         (Y/N - Enter SQL statements?)
 7  EXECUTE ......... ===> YES         (Y/N - Execute SQL statements?)
 8  AUTOCOMMIT ...... ===> YES         (Y/N - Commit after successful run?)
 9  BROWSE OUTPUT ... ===> YES         (Y/N - Browse output data set?)

For remote SQL processing:
10  CONNECT LOCATION  ===>


 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Abbildung 25: Definition für INSERT

… und drücken zweimal die Eingabetaste.

Wir tippen drei Insert-Kommandos ein (Abbildung 26). Wir können die vorgegebenen Namen oder andere Namen verwenden.

```
   File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       PRAK085.SPUFI.IN(INSERT) - 01.00               Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
'''''' INSERT INTO TAB085
''''''     VALUES ('HANS', 'BAUER');
'''''' INSERT INTO TAB085
''''''     VALUES ('FRED', 'MAYER');
'''''' INSERT INTO TAB085
''''''     VALUES ('JORG', 'WAGNER');
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
Command ===>                                                 Scroll ===> PAGE
 F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
```

Abbildung 26: Inhalt der Datenbank

Anschließend verlassen wir mit F3 den Screen und drücken dann die Eingabetaste.

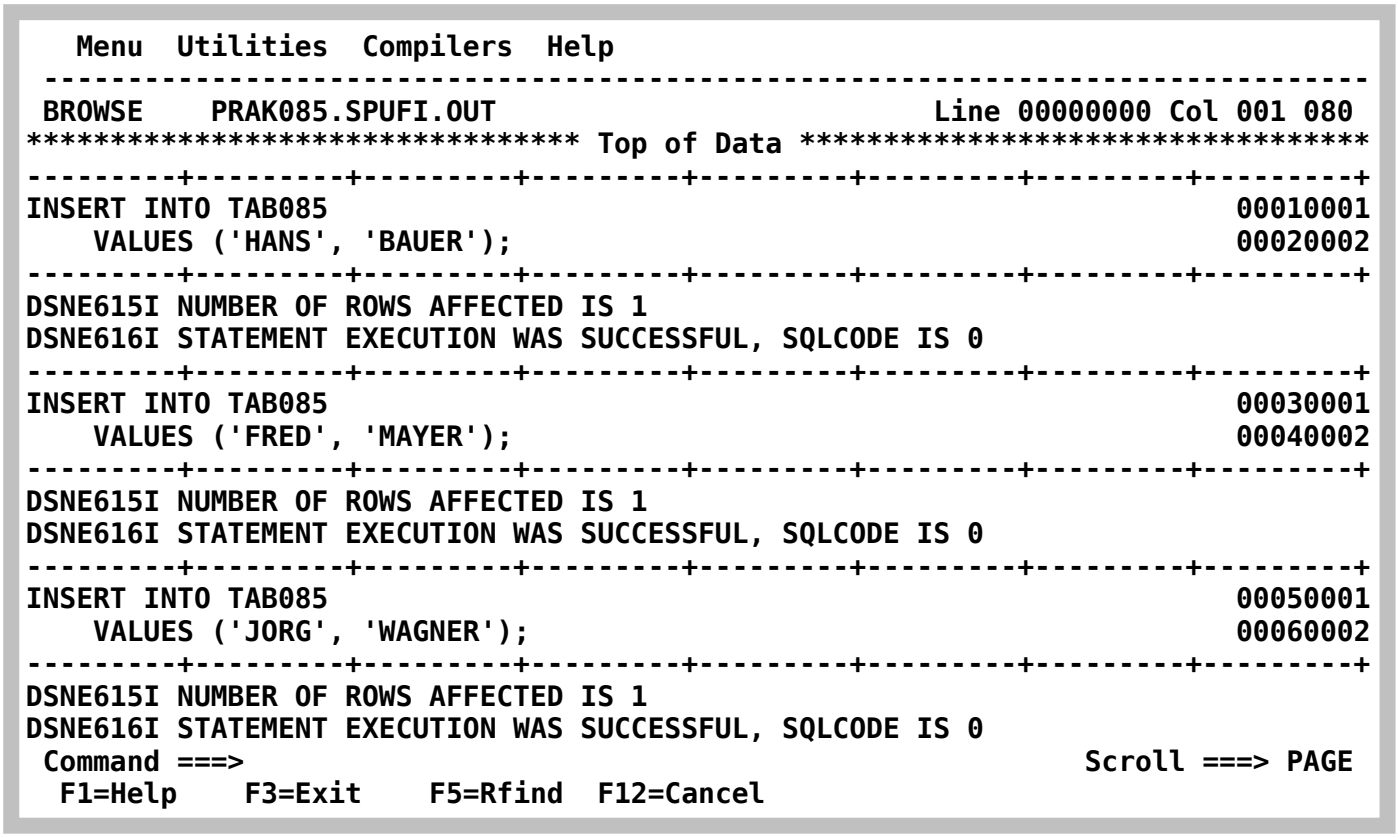


Abbildung 27: Erfolgreiche Definition

Die Eingabe war erfolgreich (s. Abbildung 27). Mit der F8-Taste sehen wir uns den Rest der Ausgabe an.

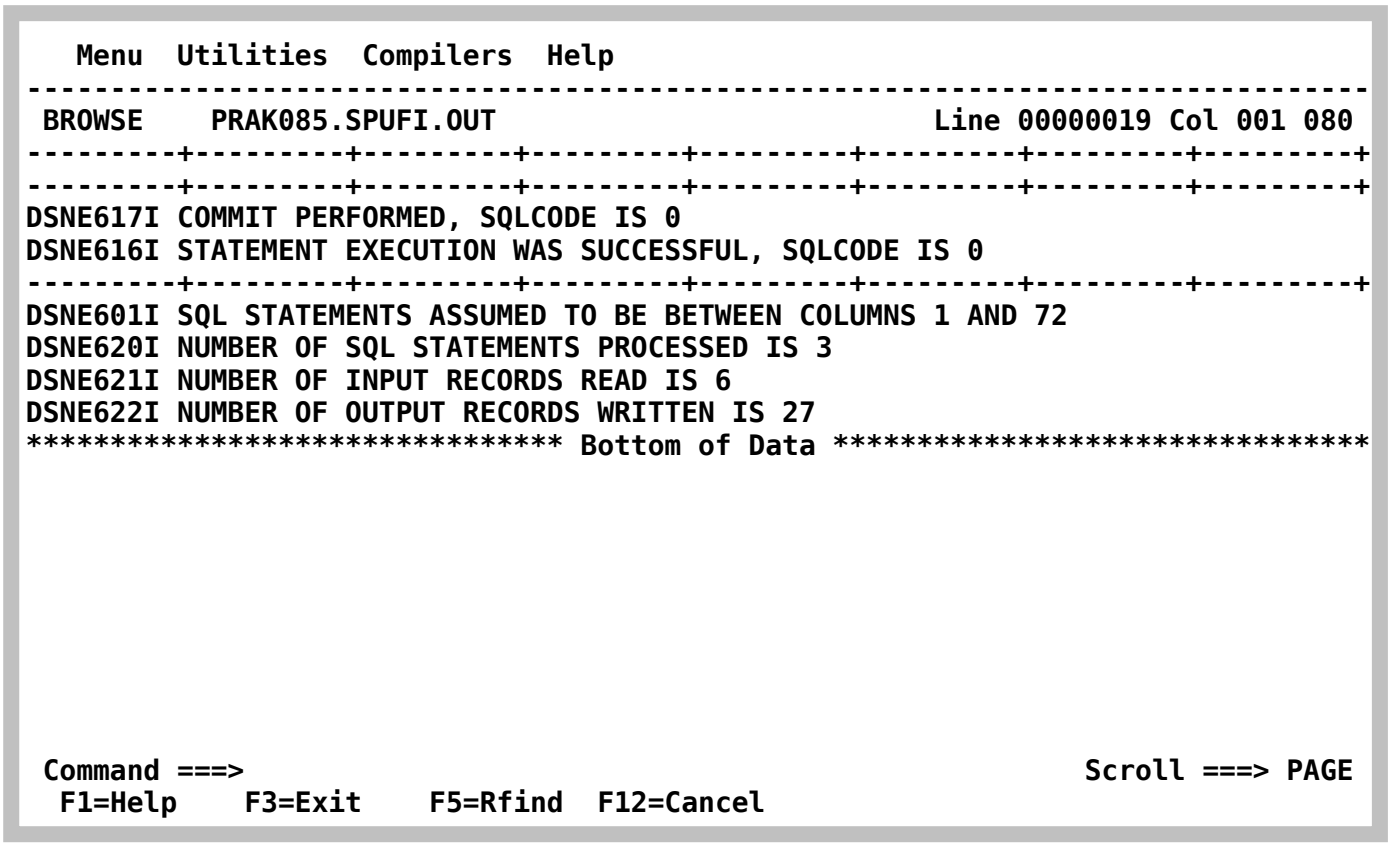


Abbildung 28: Die restliche Ausgabe

Wir haben nun unsere Datenbank mit Daten beschrieben.

Mit der F3-Taste rufen wir den SPUFI-Panel erneut auf.

12, Ansehen sämtlicher Datensätze der Tabelle

Frage: Befinden sich nun die korrekten Namen in der Datenbank? Mit SPUFI können wir uns auch den Datenbank-Inhalt ansehen.

Aufgabe:	***Selektieren Sie alle Einträge in Ihrer Tabelle, wie im folgenden Beispiel beschrieben.***

Wir erstellen noch einen weiteren SPUFI-Member "SELECT" (s. Abbildung 29) …

```
  SPUFI                                SSID: DB7G
===>
DSNE800A NO DEFAULT VALUES WERE CHANGED. PRESS ENTER TO CONTINUE
Enter the input data set name:         (Can be sequential or partitioned)
 1  DATA SET NAME ... ===> SPUFI.IN(SELECT)
 2  VOLUME SERIAL ... ===>              (Enter if not cataloged)
 3  DATA SET PASSWORD ===>              (Enter if password protected)

Enter the output data set name:        (Must be a sequential data set)
 4  DATA SET NAME ... ===> SPUFI.OUT

Specify processing options:
 5  CHANGE DEFAULTS   ===> *           (Y/N - Display SPUFI defaults panel?)
 6  EDIT INPUT ...... ===> YES         (Y/N - Enter SQL statements?)
 7  EXECUTE ......... ===> YES         (Y/N - Execute SQL statements?)
 8  AUTOCOMMIT ...... ===> YES         (Y/N - Commit after successful run?)
 9  BROWSE OUTPUT ... ===> YES         (Y/N - Browse output data set?)

For remote SQL processing:
10  CONNECT LOCATION  ===>


 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Abbildung 29: Definition für SELECT

… und drücken zweimal die Eingabetaste.

Wir erstellen eine Abfrage "SELECT" für unsere Datenbank (s. Abbildung 30), verlassen den Screen mit F3 und betätigen anschließend die Eingabetaste.

```
  File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
--------------------------------------------------------------------------------
EDIT       PRAK085.SPUFI.IN(SELECT) - 01.00               Columns 00001 00072
****** ***************************** Top of Data ********************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
'''''' SELECT * FROM PRAK085.TAB085;
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
Command ===>                                                 Scroll ===> PAGE
 F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
```

Abbildung 30: SELECT

```
   Menu  Utilities  Compilers  Help
 ------------------------------------------------------------------------------
 BROWSE    PRAK085.SPUFI.OUT                       Line 00000000 Col 001 080
******************************** Top of Data ********************************
---------+---------+---------+---------+---------+---------+---------+---------+
SELECT * FROM PRAK085.TAB085;                                          00010000
---------+---------+---------+---------+---------+---------+---------+---------+
VNAME                NNAME
---------+---------+---------+---------+---------+---------+---------+---------+
HANS                 BAUER
FRED                 MAYER
JORG                 WAGNER
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+---------+
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 1
DSNE621I NUMBER OF INPUT RECORDS READ IS 1
 Command ===>                                          Scroll ===> PAGE
  F1=Help    F3=Exit    F5=Rfind  F12=Cancel
```

Abbildung 31: Erfolgreiche Definition

Die Abfrage war erfolgreich, wie aus Abbildung 31 hervorgeht.

SPUFI gibt den Inhalt der Tabelle wieder. Die mit dem "INSERT"-Statement eingegebenen drei Vornamen und Nachnamen sind in der Tabelle gespeichert.

Wir können das Aufnehmen von Datensätzen wiederholen, indem wir den SPUFI-Screen mit "INSERT" erneut aufrufen und weitere Namen eingeben. Sie werden an den vorhandenen Datenbestand angehängt. Führen Sie es als Übung durch:

F3-Taste drücken, um den SPUFI-Screen aufzurufen, den aufzurufenden Membernamen mit "INSERT" eingeben, zweimal die Eingabetaste drücken, um zum "Edit Screen" zu gelangen, neue Daten eingeben, mit F3 und anschließend mit der Eingabetaste das Aufnehmen dieser Namen in die Tabelle durchführen.

Wenn wir uns nun mittels des SQL-Kommandos "SELECT * FROM PRAK085.TAB085;" wieder den Inhalt unserer Tabelle anschauen, könnte dieser wie in Abbildung 32 dargestellt aussehen.

```
  Menu  Utilities  Compilers  Help
------------------------------------------------------------------------------
BROWSE    PRAK085.SPUFI.OUT                        Line 00000000 Col 001 080
******************************** Top of Data ********************************
---------+---------+---------+---------+---------+---------+---------+---------+
SELECT * FROM PRAK085.TAB085;                                          00010000
---------+---------+---------+---------+---------+---------+---------+---------+
VNAME                NNAME
---------+---------+---------+---------+---------+---------+---------+---------+
HANS                 BAUER
FRED                 MAYER
JORG                 WAGNER
HEINZ                BAUER
FRITZ                MAYER
RICHARD              SCHULTE
JORG                 MEISTER
HANS                 BERG
FRITZ                MEIER
RICHARD              SCHMITZ
MARTIN               WAGNER
DSNE610I NUMBER OF ROWS DISPLAYED IS 11
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
 Command ===>                                          Scroll ===> PAGE
  F1=Help    F3=Exit    F5=Rfind  F12=Cancel
```

Abbildung 32: Datenbank mit erweitertem Inhalt

Nach Betätigen der F8-Taste erscheint der nächsten Screen der Ausgabe.

```
  Menu  Utilities  Compilers  Help
 ------------------------------------------------------------------------------
 BROWSE    PRAK085.SPUFI.OUT                          Line 00000019 Col 001 080
---------+---------+---------+---------+---------+---------+---------+---------+
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE617I COMMIT PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 1
DSNE621I NUMBER OF INPUT RECORDS READ IS 1
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 27
****************************** Bottom of Data ********************************
 Command ===>                                                Scroll ===> PAGE
  F1=Help    F3=Exit    F5=Rfind  F12=Cancel
```

Abbildung 33: Der letzte Teil der Ausgabe

Aufgabe: ***Erstellen Sie einen Screenshot entsprechend der Abbildung 31 bzw. 32, der alle Einträge Ihrer Tabelle zeigt.***

Wir haben erfolgreich eine DB2-Datenbank angelegt und mit Daten bestückt. Bitte bedenken Sie, dass wir hier Prinzipien erklären, in der Praxis jedoch meist mit anderen und weitgehend automatisierten Prozessen gearbeitet wird.

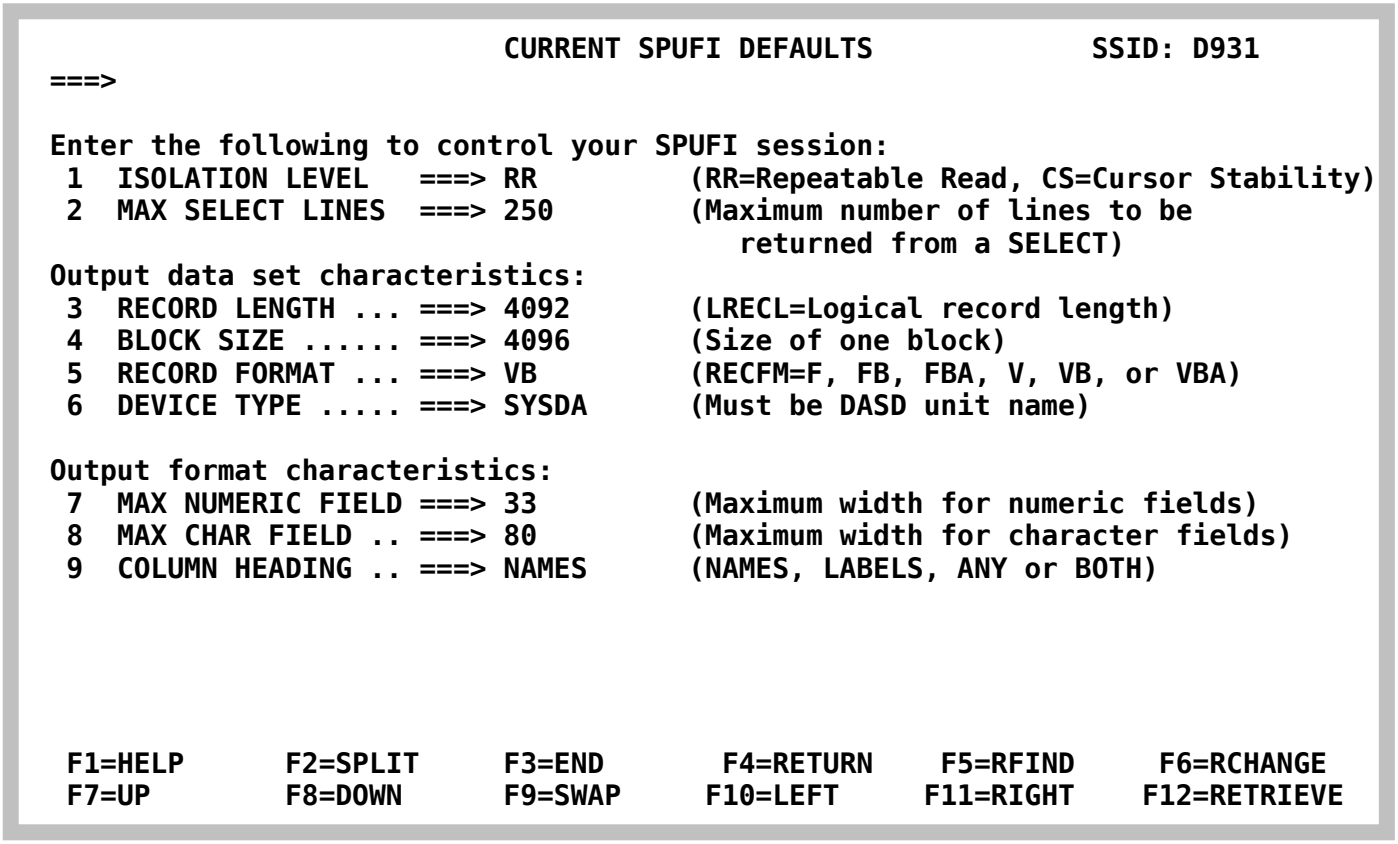
Anhang

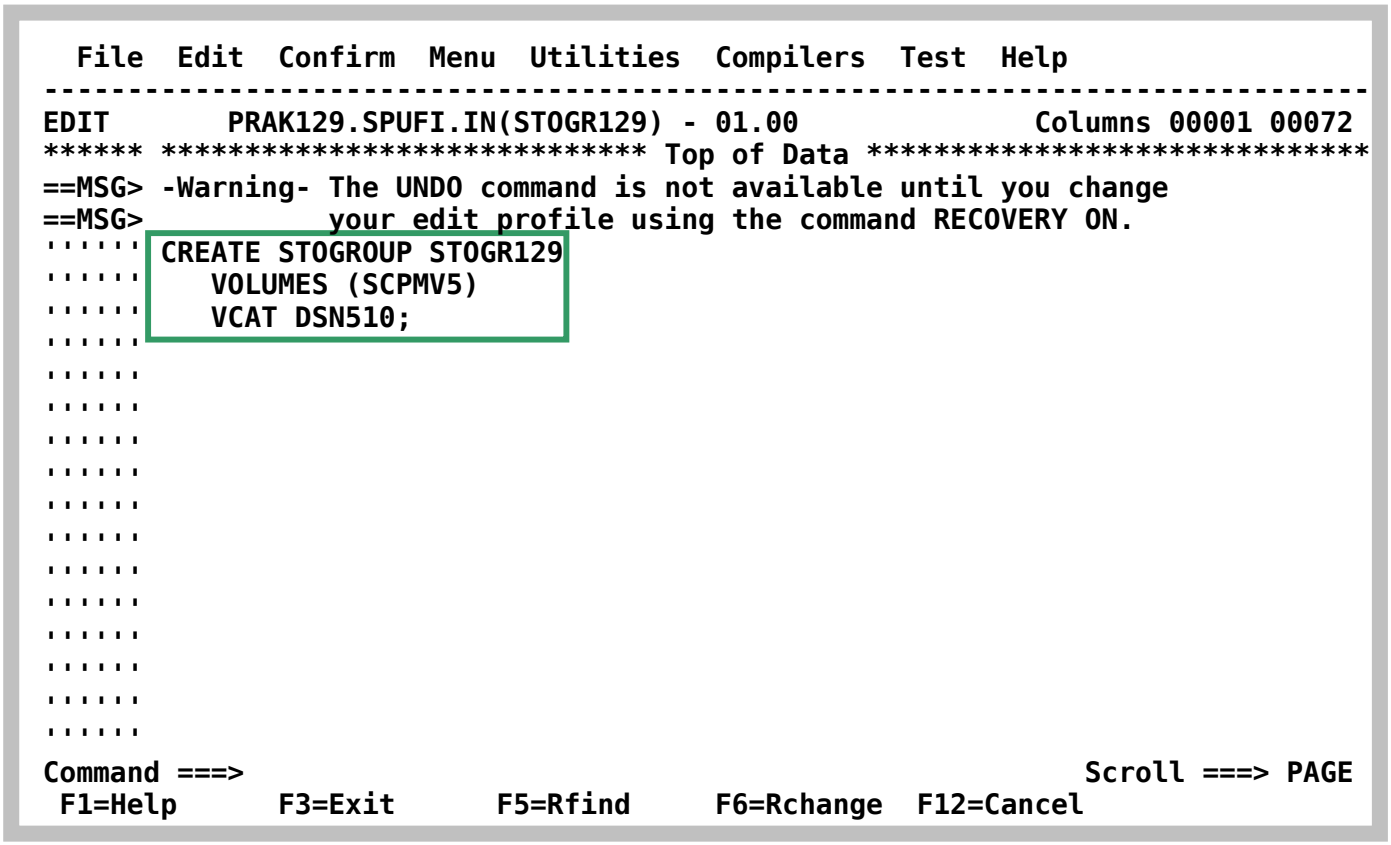
Fehlermeldung beim Anlegen der Storage Group
Löschen von alten Objekten, um Neue anlegen zu können

Dieser Anhang ist für die Praktikumsteilnehmer von theoretischem Interesse. Hier wird der Vollständigkeit halber beschrieben, wie ein privilegierter User vorgeht, wenn Fehler beim Anlegen einer Storage Group passieren. Folgende Fehlersituation: Nehmen wir an, wir sind folgendermaßen vorgegangen:

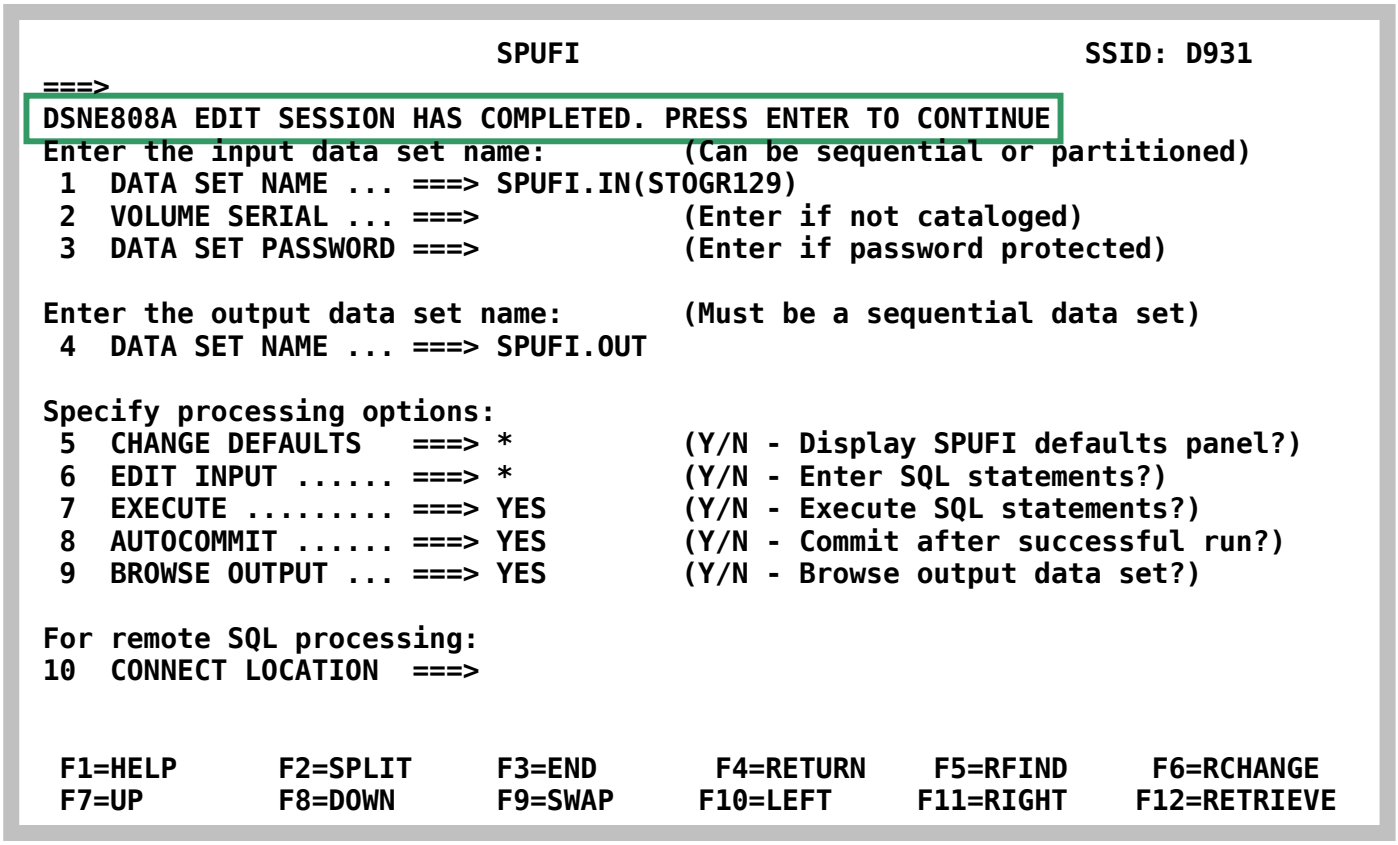
```
                          SPUFI                                SSID: D931
===>

Enter the input data set name:         (Can be sequential or partitioned)
 1  DATA SET NAME ... ===> SPUFI.IN(STOGR129)
 2  VOLUME SERIAL ... ===>              (Enter if not cataloged)
 3  DATA SET PASSWORD ===>              (Enter if password protected)

Enter the output data set name:        (Must be a sequential data set)
 4  DATA SET NAME ... ===> SPUFI.OUT

Specify processing options:
 5  CHANGE DEFAULTS   ===> YES         (Y/N - Display SPUFI defaults panel?)
 6  EDIT INPUT ...... ===> YES         (Y/N - Enter SQL statements?)
 7  EXECUTE ......... ===> YES         (Y/N - Execute SQL statements?)
 8  AUTOCOMMIT ...... ===> YES         (Y/N - Commit after successful run?)
 9  BROWSE OUTPUT ... ===> YES         (Y/N - Browse output data set?)

For remote SQL processing:
10  CONNECT LOCATION  ===>


 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Enter


CURRENT SPUFI DEFAULTS SSID: D931
===>
Enter the following to control your SPUFI session:
1 ISOLATION LEVEL ===> RR (RR=Repeatable Read, CS=Cursor Stability)
2 MAX SELECT LINES ===> 250 (Maximum number of lines to be returned from a SELECT)
Output data set characteristics:
3 RECORD LENGTH ... ===> 4092 (LRECL=Logical record length)
4 BLOCK SIZE ..... ===> 4096 (Size of one block)
5 RECORD FORMAT ... ===> VB (RECFM=F, FB, FBA, V, VB, or VBA)
6 DEVICE TYPE ..... ===> SYSDA (Must be DASD unit name)
Output format characteristics:
7 MAX NUMERIC FIELD ===> 33 (Maximum width for numeric fields)
8 MAX CHAR FIELD .. ===> 80 (Maximum width for character fields)
9 COLUMN HEADING .. ===> NAMES (NAMES, LABELS, ANY or BOTH)
F1=HELP F2=SPLIT F3=END F4=RETURN F5=RFIND F6=RCHANGE
F7=UP F8=DOWN F9=SWAP F10=LEFT F11=RIGHT F12=RETRIEVE



Wir geben diesen Text ein, dann F3



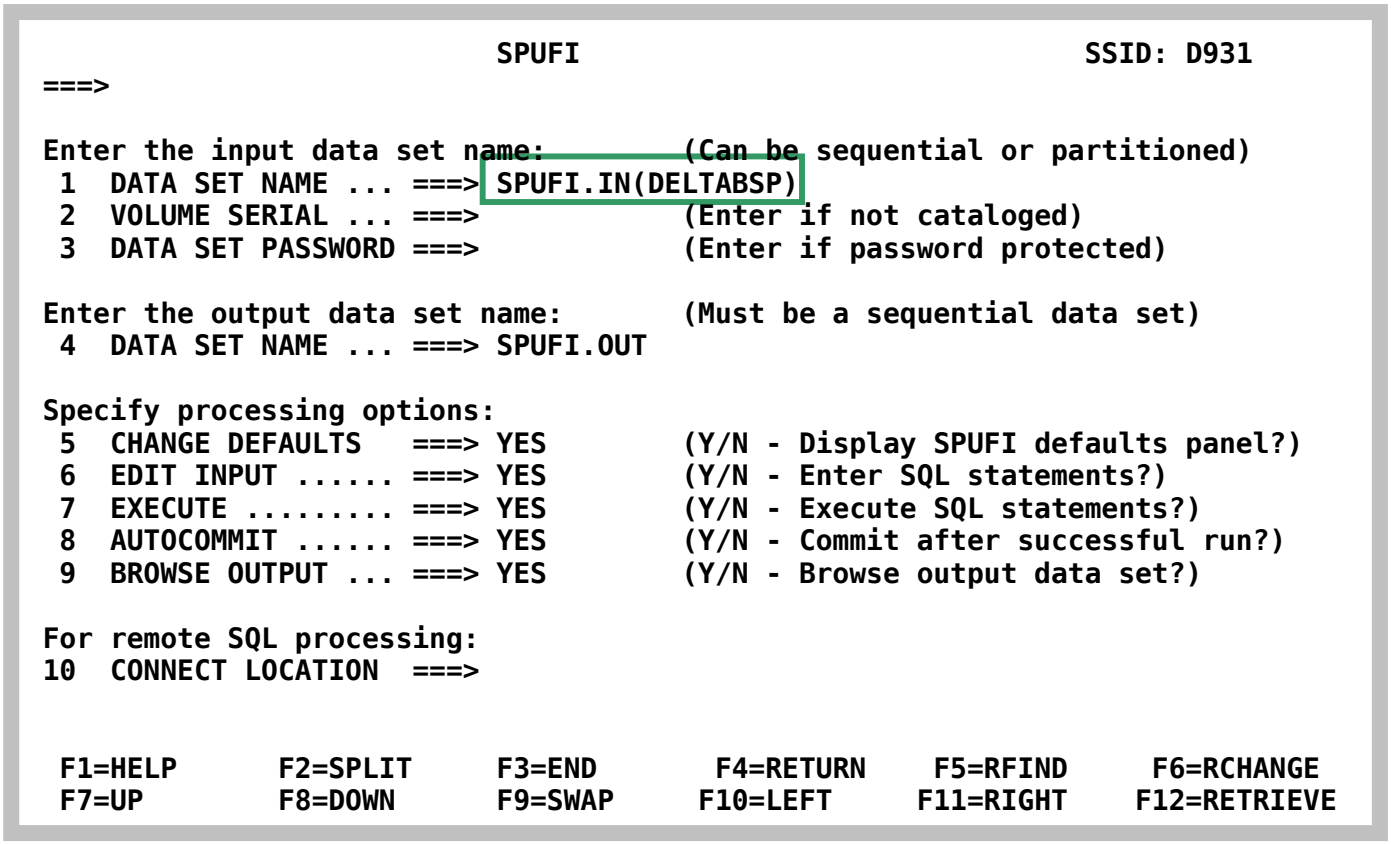
Und jetzt Enter

```
   Menu  Utilities  Compilers  Help
 ------------------------------------------------------------------------------
 BROWSE    PRAK129.SPUFI.OUT                          Line 00000000 Col 001 080
******************************** Top of Data ********************************
---------+---------+---------+---------+---------+---------+---------+---------+
CREATE STOGROUP STOGR129                                                 00010000
   VOLUMES (SCPMV5)                                                      00020000
   VCAT DSN510;                                                          00030000
---------+---------+---------+---------+---------+---------+---------+---------+
DSNT408I SQLCODE = -601, ERROR:  THE NAME OF THE OBJECT TO BE CREATED OR THE
         TARGET OF A RENAME STATEMENT IS IDENTICAL TO THE EXISTING NAME
         STOGR129 OF THE OBJECT TYPE STOGROUP
DSNT418I SQLSTATE   = 42710 SQLSTATE RETURN CODE
DSNT415I SQLERRP    = DSNXICSG SQL PROCEDURE DETECTING ERROR
DSNT416I SQLERRD    = 10  0  0  -1  0  0 SQL DIAGNOSTIC INFORMATION
DSNT416I SQLERRD    = X'0000000A'  X'00000000'  X'00000000'  X'FFFFFFFF'
         X'00000000'  X'00000000' SQL DIAGNOSTIC INFORMATION
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE618I ROLLBACK PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
 Command ===>                                          Scroll ===> PAGE
  F1=Help    F3=Exit    F5=Rfind  F12=Cancel
```

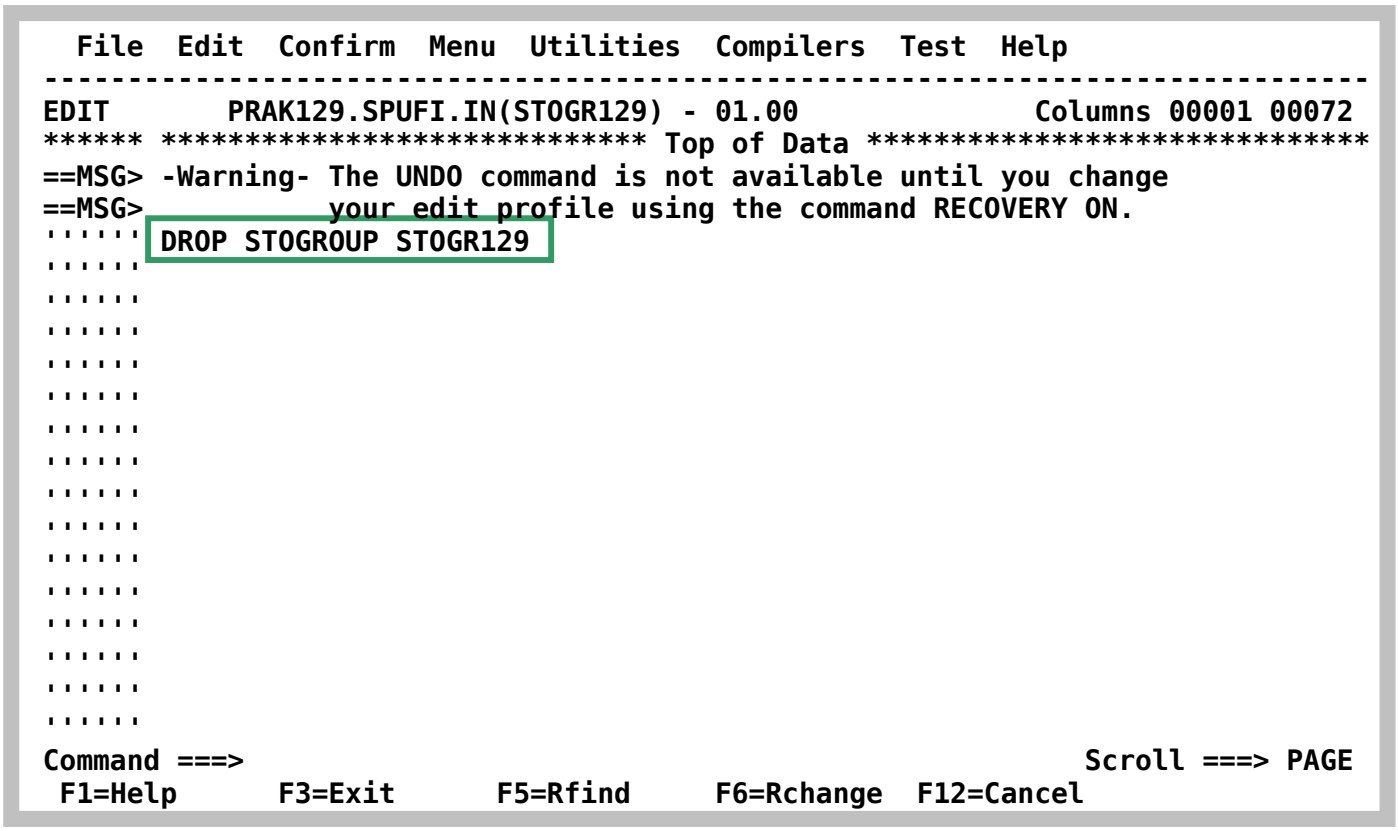
Wird an dieser Stelle eine Fehlermeldung ausgegeben, dann ist der zu erstellende Speicherplatz für unsere Datenbank (Stogroup) möglicherweise bereits vorhanden (s. Abbildung 14a). Er wurde zu einem früheren Zeitpunkt von jemand anderem schon einmal angelegt. In diesem Fall ist der alte Speicherplatz erst einmal zu entfernen. Dies ist mit dem folgenden SQL-Statement möglich:

```
DROP STOGROUP STOGR129
```

Hierzu gehen Sie folgendermaßen vor:



Im SPUFI Panel machen Sie die obenstehende Eingabe. Anstelle von "STOGR1" wählt man einen in dem Dataset "SPUFI.IN" noch nicht benutzten Member-Namen, z.B. "DELTABSP" (DELete TABle Space). Dieses Member nimmt das auszuführende SQL-Statement z.B. "DROP STOGROUP STOGR129" auf.



Mit F3 und Enter verifizieren Sie, dass dies funktioniert hat.

Die Stogroup wird aber nur gelöscht, wenn sich in ihr keine Speicherplatzreservierung für eine Tabelle befindet (Tablespace). Befindet sich eine solche in der Stogroup, wird die unten dargestellte Fehlermeldung ausgegeben sowie die Stogroup nicht gelöscht. Die Fehlermeldung gibt aber den Namen des Objektes aus, welches vor der Stogroup noch gelöscht werden muss, in diesem Beispiel der Tablespace "DB129.TABSP129". Dies geschieht durch das SQL-Statement

```
   Menu  Utilities  Compilers  Help
 ------------------------------------------------------------------------------
 BROWSE    PRAK085.SPUFI.OUT                        Line 00000000 Col 001 080
******************************** Top of Data ********************************
---------+---------+---------+---------+---------+---------+---------+---------+
DROP STOGROUP STOGR129                                                 00010000
         +         +         +         +         +         +         +         +
DSNT408I SQLCODE = -616, ERROR:  STOGROUP STOGR129 CANNOT BE DROPPED BECAUSE IT
         IS REFERENCED BY TABLESPACE DB129.TABSP129
DSNT418I SQLSTATE   = 42893 SQLSTATE RETURN CODE
DSNT415I SQLERRP    = DSNXIDSG SQL PROCEDURE DETECTING ERROR
DSNT416I SQLERRD    = 60  0  0  -1  0  0 SQL DIAGNOSTIC INFORMATION
DSNT416I SQLERRD    = X'0000003C'  X'00000000'  X'00000000'  X'FFFFFFFF'
         X'00000000'  X'00000000' SQL DIAGNOSTIC INFORMATION
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE618I ROLLBACK PERFORMED, SQLCODE IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0
---------+---------+---------+---------+---------+---------+---------+---------+
DSNE601I SQL STATEMENTS ASSUMED TO BE BETWEEN COLUMNS 1 AND 72
DSNE620I NUMBER OF SQL STATEMENTS PROCESSED IS 1
DSNE621I NUMBER OF INPUT RECORDS READ IS 1
DSNE622I NUMBER OF OUTPUT RECORDS WRITTEN IS 18
 Command ===>                                            Scroll ===> PAGE
   F1=Help    F3=Exit    F5=Rfind  F12=Cancel
```

Die Fehlermeldung gibt aber den Namen des Objektes aus, welches vor der Stogroup noch gelöscht werden muss, in diesem Beispiel der Tablespace "DB129.TABSP129". Dies geschieht durch das SQL-Statement

DROP TABLESPACE DB129.TABSP129

Weiterhin ist folgende Situation möglich. Sie haben Ihre Database angelegt, hierbei aber einen Fehler gemacht. Wenn Sie das Edit Panel erneut aufrufen, den Fehler korrigieren und dann F3 und Enter machen, erscheint eine Fehlermeldung, dass das Objekt schon existiert. In diesem Fall müssen Sie das fälschlich angelegte Objekt mit

DROP DATABASE DB129

erst eliminieren, ehe Sie die korrekte Version erneut anlegen können.

CICS Cobol Tutorial 3
Datenbankzugriff mit CICS (COBOL)

Ziel dieses Tutorials ist es, mittels einer CICS-Transaction auf die in Tutorial 4 erstellte DB2-Datenbank zuzugreifen. Unser Anwendungsprogramm soll wieder aus zwei Teilen bestehen, einem COBOL-Programm für die Business Logic und einem BMS-Programm für die Presentation Logic.

Hinweis: Tutorial 5 baut auf die erfolgreiche Bearbeitung von Tutorial 4 auf.

Unser Business Logic-Programm soll dabei SQL-Aufrufe enthalten. Diese müssen durch einen SQL-Precompiler in native DB2 API-Aufrufe übersetzt werden, ehe der COBOL-Compiler das Business Logic-Programm übersetzen kann.

1. Vorgehensweise

CICS trennt strikt Berechnungen und Datenbankzugriffe von dem Layout der Darstellungen auf Panels. Ersteres wird als "Business Logic" und letzteres als "Presentation Logic" bezeichnet.

Die Präsentationslogik wird , genauso wie in Tutorial 3, durch ein JCL Script erzeugt. Die Business Logik besteht aus einem Cobol Programm, welches auf die DB2 Datenbank zugreift.

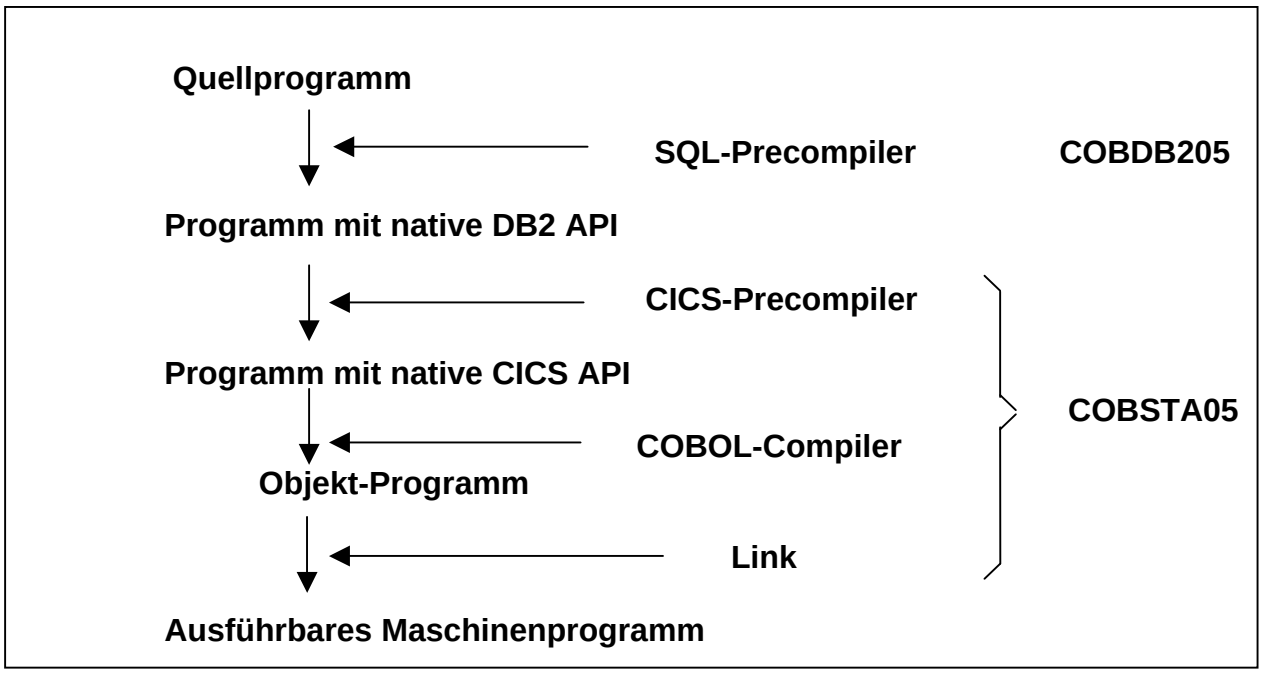


Abbildung 1:
Schritte vom COBOL-Programm mit EXEC SQL-Statements zum ausführbaren Maschinenprogramm

Wie in Abbildung 1 dargestellt, muss das fertige Cobol Quellprogramm vor der Übersetzung durch einen SQL Precompiler verarbeitet werden. Wir benutzen standardmäßig SQL für den Zugriff auf die DB2 Datenbank. Der SQL Precompiler übersetzt alle SQL Aufrufe durch die native API der DB2 Datenbank. Dieser native API Code wird ebenfalls in Cobol generiert. Das gesamte Cobol Programm wird dann wie in Tutorial 3 durch einen Compiler und einen Linkage Editor in ausführbaren Maschinencode übersetzt.

Unser Anwendungsprogramm besteht aus einem Cobol Programm und mehreren JCL Scripts. Wir bringen diese als Member in einem neuen partitioned Data Set PRAK218.CICSDB2.COB unter. Spezifisch benutzen wir diese Member:

Cobol CICS Business Logic Quellcode	**PRAK218.CICSDB2.COB(COB218)**
Cobol CICS Presentation Logic	**PRAK218.CICSDB2.COB(COBMAP5)**
JCL Script für DB2 Precompile Lauf	**PRAK218.CICSDB2.COB(COBDB205)**
JCL Script für Cobol Übersetzung	**PRAK218.CICSDB2.COB(COBSTA05)**

Der Dataset PRAK218.CICSDB2.COB muss neu angelegt werden.

In diesem wird als ersten Schritt in einem Member das auszuführende JCL-Script: PRAK218.CICSDB2.COB(COBMAP5) (s. die Abbildungen 4) angelegt, welches die Presentation Logic behandelt. Sie besteht aus genau einem Mapset "MSET218", der genau eine Map "MAP218" enthält. Ein Mapset kann aber auch mehrere Maps enthalten. Diese Map "MAP218" definiert Positionen, Länge sowie weitere Attribute der Darstellung der Daten aus der DB2-Datenbank auf dem Bildschirm.

Anschließend erstellen wir das COBOL-Programm "PRAK218.CICSDB2.COB(COB218)" (s. die Abbildung 6), welches EXEC SQL-Statements enthält.

So wie in Abbildung 1 dargestellt, führt das zweite von uns erstellte JCL-Script "PRAK218.CICSDB2.COB(COBDB205)" einen Precompiler Durchlauf aus. Alle EXEC SQL-Statements im COBOL-Programm werden durch dieses in native DB2 API-Aufrufe konvertiert.

Als drittes JCL-Script wird von uns "PRAK218.CICSDB2.COB(COBSTA05)" erstellt und ausgeführt (s. auch die Abbildung 12). Der CICS-Precompiler generiert aus dem COBOL-Programm mit native *DB2* API-Aufrufen ein COBOL-Programm mit native *CICS* API-Aufrufen. Anschließend wird der nun so entstandene COBOL-Programmcode zu einen Objekt-Programmcode übersetzt, aus welchem der Linker ein ausführbares Maschinenprogramm erzeugt (s. auch Abbildung 1).

Alle diese Schritte werden im TSO ausgeführt. TSO ist ein Subsystem von z/OS. Ein weiteres Subsystem von z/OS ist CICS. Der folgende Teil des Tutorials erläutert, über welche Schritte das übersetzte Cobol Programm innerhalb des CICS Subsystems als CICS-Transaktion mit der Transaktions-ID "X218" installiert wird. Folgende Schritte sind dazu notwendig:

1. **Definition des Mapsets mittels "CEDA DEFINE MAPSET(MSET218) GROUP(PRAK218)"**
2. **Definition des COBOL-Programmes mittels "CEDA DEFINE PROG(COB218) GROUP(PRAK218)"**
3. **Definition des Namens der Transaction-ID mittels "CEDA DEFINE TRANS(X218) GROUP(PRAK218)"**
4. **Definition unserer Datenbank und Datenbanktabelle mittels "CEDA DEFINE DB2ENTRY"**

Nach diesen Schritten sind der Mapset MSET218 mit der Map MAP218, das ausführbare Maschinenprogramm, das aus COB218 generiert wurde, die selbst definierte Transaction-ID "X218" sowie die Datenbank und Tabelle, aus der ausgelesen werden soll, dem CICS-System bekannt. Ebenfalls ist ihm bekannt, dass alle diese Komponenten der Gruppe "PRAK218" zugewiesen wurden. Diese Gruppe wird durch Schritt 1. automatisch erstellt. Doch diese *Definitionen* reichen noch nicht aus. Unser Ziel erreichen wir erst, wenn alle Komponenten auch *installiert* werden. Dies geschieht durch

5. **"CEDA INSTALL GROUP(PRAK218)"**

Nun haben wir unser Ziel erreicht. Geben wir "X218" unter CICS ein (s.

Abbildung 30), so wird unsere selbst definierte Transaktion ausgeführt, welche die Spalten "VORNAME" und "NACHNAME" aus der im Tutorial 4 angelegten DB2-Tabelle ausliest und auf unserem Bildschirm ausgibt (s.
Abbildung 31).
Warnung:

Ihr DB2ENTRY ist nur ein einziges Mal von Ihnen installierbar. Deshalb könnte z.B. Ihre zweite Anwendung von "CEDA INSTALL GROUP ..." die Fehlermeldung "INSTALL UNSUCCESSFUL" produzieren. Dieser Fehler kann von Ihnen mangels Ihrer CICS-Zugriffsrechte nicht behoben werden. Im Anhang wird dieses Problem erläutert. Informieren Sie deshalb umgehend Ihren Betreuer.

2. Anlegen des Mapsets

Die nun benötigten Datasets "PRAK218.DBRMLIB.DATA", "PRAK218.LIB" und "PRAK218.CICSDB2.COB" müssen neu angelegt werden. "PRAK218.DBRMLIB.DATA" wird während der Ausführung des SQL-Precompilers automatisch gefüllt. "PRAK218.CICSDB2.COB" nimmt die von uns zu erstellenden Quellprogramme auf.

Aufgabe: *Legen Sie den Dataset "PRAK218.CICSDB2.COB" und - wenn noch nicht vorhanden - die Datasets "PRAK218.LIB" sowie "PRAK218.DBRMLIB.DATA" an. Verwenden Sie dazu folgende Parameter:*

Space units . . . . . KILOBYTE	*Record format . . . . FB*
Primary quantity . . 16	*Record length 80*
Secondary quantity 1	*Block size 320*
Directory blocks . . 2	*Data set name type : PDS*

Falls einer der beiden Datasets "PRAK218.LIB" oder "PRAK218.DBRMLIB.DATA" auf ihrem Account schon vorhanden ist, empfiehlt sich das Löschen aller Members sowie ein anschließender Compress dieser Datasets.

Mit Hilfe des "Edit Entry Panels" erstellen wir einen neuen Member "COBMAP5" (s. Abbildung 3) und bestätigen anschließend mit der Eingabetaste.

```
   Menu  RefList  RefMode  Utilities  Workstation  Help
--------------------------------------------------------------------------------
                              Edit Entry Panel

ISPF Library:
   Project . . . PRAK218
   Group . . . . CICSDB2  . . .          . . .          . . .
   Type  . . . . COB
   Member  . . . COBMAP5         (Blank or pattern for member selection list)

Other Partitioned, Sequential or VSAM Data Set:
   Data Set Name . . .
   Volume Serial . . .           (If not cataloged)

Workstation File:
   File Name . . . . .
                                       Options
Initial Macro  . . . .                    Confirm Cancel/Move/Replace
Profile Name . . . . .                    Mixed Mode
Format Name  . . . . .                    Edit on Workstation
Data Set Password  . .                    Preserve VB record length
```

Abbildung 2: Anlegen des Members "COBMAP5"

Unsere CICS-Anwendung soll wiederum aus einem BMS-Programm (Mapset) für die "Presentation Logic" und einem COBOL-Programm für die Business Logic bestehen. Wir beginnen mit dem Mapset.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------
EDIT       PRAK218.CICSDB2.COB(COBMAP5) - 01.06            Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 //PRAK218M JOB (),CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1),NOTIFY=&SYSUID,
000200 //         REGION=4M
000300 //ASSEM EXEC DFHMAPS,MAPNAME='MSET218',RMODE=24
000400 //COPY.SYSUT1  DD  *
000500 MSET085  DFHMSD TYPE=MAP,MODE=INOUT,LANG=COBOL2,STORAGE=AUTO,          *
000510               TIOAPFX=YES
000600 *    MENU  MAP
000700 MAP218   DFHMDI SIZE=(24,80),CTRL=(PRINT,FREEKB)
001000          DFHMDF POS=(9,13),ATTRB=(ASKIP,NORM),LENGTH=20,              *
001100               INITIAL='VORNAME             '
001200          DFHMDF POS=(9,34),ATTRB=(ASKIP,NORM),LENGTH=20,              *
001210               INITIAL='NACHNAME'
001300 VNAM1    DFHMDF POS=(11,13),ATTRB=(ASKIP,NORM),LENGTH=20
001400 NNAM1    DFHMDF POS=(11,34),ATTRB=(ASKIP,NORM),LENGTH=20
001500 VNAM2    DFHMDF POS=(12,13),ATTRB=(ASKIP,NORM),LENGTH=20
001600 NNAM2    DFHMDF POS=(12,34),ATTRB=(ASKIP,NORM),LENGTH=20
001700 VNAM3    DFHMDF POS=(13,13),ATTRB=(ASKIP,NORM),LENGTH=20
001800 NNAM3    DFHMDF POS=(13,34),ATTRB=(ASKIP,NORM),LENGTH=20
001900 VNAM4    DFHMDF POS=(14,13),ATTRB=(ASKIP,NORM),LENGTH=20
002000 NNAM4    DFHMDF POS=(14,34),ATTRB=(ASKIP,NORM),LENGTH=20
002100          DFHMSD TYPE=FINAL
002200          END
002300 /*
002400 //
****** **************************** Bottom of Data ****************************
Command ===>                                                Scroll ===> PAGE
 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
===============================================================================
```

Abbildung 3: Das BMS-Programm

Dies ist das vollständige BMS-Programm nach Fertigstellung.

Die Zeilen 700 bis 1210 definieren eine Überschrift, die aus 2 Feldern besteht. Die beiden Felder werden mit den Werten VORNAME und NACHNAME initialisiert.

Die Zeilen 1300 bis 2000 definieren 8 Felder, welche die Vornamen und Nachnamen von 4 Personen aufnehmen sollen, welche wir aus unserer DB2-Datenbank auslesen.

Wir geben "SUB" auf der Kommandozeile ein. Zusätzlich zu dem übersetzten Programm wird in dem Member "PRAK218.LIB(MSET218)" ein Template für unser Business Logic-Programm abgespeichert.

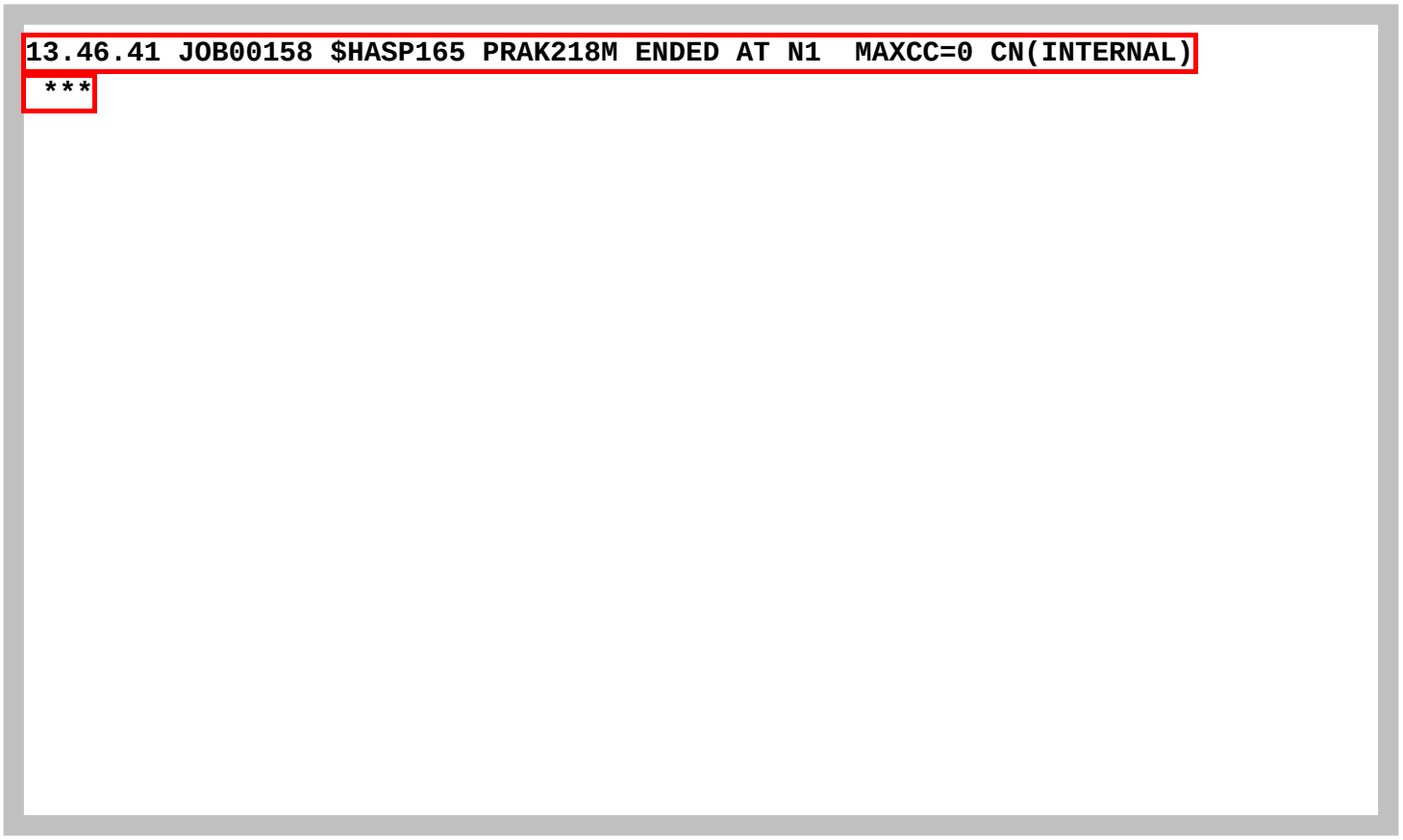


Abbildung 4: Bestätigung der Jobverarbeitung

Wir warten, bis JES unser BMS-Programm übersetzt hat (30-60 Sekunden). Durch das Betätigen der Eingabetaste erscheint der hier gezeigte Panel (s.Abbildung 7). "MAXCC=0" bestätigt, dass die Übersetzung erfolgreich war.

Die Eingabetaste bringt uns zurück zum vorhergehenden Screen.

Aufgabe: ***Legen Sie einen Member an, schreiben Sie das BMS-Programm und führen Sie es aus. Ersetzen Sie "//PRAK218M" entsprechend Ihres Mainframe-Accountnamens. Benutzen Sie MAP<Ihre Prakt-ID> als Mapnamen sowie SET<Ihre Prakt-ID> als Mapsetnamen. Haben Sie z.B. den Account PRAK162, so ist Ihr Map-Name MAP162 und Ihr Mapset-Name MSET162.***

Wir betätigen zweimal die F3-Taste, um diesen Bildschirm zu verlassen.

Als nächstes sehen wir uns die Members von "PRAK218.LIB" an.

Wir wechseln zu dem Partitioned Dataset "PRAK218.LIB". Dort existiert jetzt der während der Übersetzung erstellte Member "PRAK218.LIB(MSET5218)". Wir schauen uns "PRAK218.LIB(MSET5218)" an.

```
000001        01  MAP218I.
000002            02  FILLER PIC X(12).
000003            02  VNAM1L    COMP  PIC  S9(4).
000004            02  VNAM1F    PICTURE X.
000005            02  FILLER REDEFINES VNAM1F.
000006              03 VNAM1A    PICTURE X.
000007            02  VNAM1I  PIC X(20).
000008            02  NNAM1L    COMP  PIC  S9(4).
000009            02  NNAM1F    PICTURE X.
000010            02  FILLER REDEFINES NNAM1F.
000011              03 NNAM1A    PICTURE X.
000012            02  NNAM1I  PIC X(20).
000013            02  VNAM2L    COMP  PIC  S9(4).
000014            02  VNAM2F    PICTURE X.
000015            02  FILLER REDEFINES VNAM2F.
000016              03 VNAM2A    PICTURE X.
000017            02  VNAM2I  PIC X(20).
000018            02  NNAM2L    COMP  PIC  S9(4).
000019            02  NNAM2F    PICTURE X.
000020            02  FILLER REDEFINES NNAM2F.
000021              03 NNAM2A    PICTURE X.
000022            02  NNAM2I  PIC X(20).
000023            02  VNAM3L    COMP  PIC  S9(4).
000024            02  VNAM3F    PICTURE X.
000025            02  FILLER REDEFINES VNAM3F.
000026              03 VNAM3A    PICTURE X.
000027            02  VNAM3I  PIC X(20).
000028            02  NNAM3L    COMP  PIC  S9(4).
000029            02  NNAM3F    PICTURE X.
000030            02  FILLER REDEFINES NNAM3F.
000031              03 NNAM3A    PICTURE X.
000032            02  NNAM3I  PIC X(20).
000033            02  VNAM4L    COMP  PIC  S9(4).
000034            02  VNAM4F    PICTURE X.
000035            02  FILLER REDEFINES VNAM4F.
000036              03 VNAM4A    PICTURE X.
000037            02  VNAM4I  PIC X(20).
000038            02  NNAM4L    COMP  PIC  S9(4).
000039            02  NNAM4F    PICTURE X.
000040            02  FILLER REDEFINES NNAM4F.
000041              03 NNAM4A    PICTURE X.
000042            02  NNAM4I  PIC X(20).
000043        01  MAP218O REDEFINES MAP218I.
000044            02  FILLER PIC X(12).
000045            02  FILLER PICTURE X(3).
000046            02  VNAM1O  PIC X(20).
000047            02  FILLER PICTURE X(3).
000048            02  NNAM1O  PIC X(20).
000049            02  FILLER PICTURE X(3).
000050            02  VNAM2O  PIC X(20).
000051            02  FILLER PICTURE X(3).
000052            02  NNAM2O  PIC X(20).
000053            02  FILLER PICTURE X(3).
000054            02  VNAM3O  PIC X(20).
000055            02  FILLER PICTURE X(3).
000056            02  NNAM3O  PIC X(20).
000057            02  FILLER PICTURE X(3).
000058            02  VNAM4O  PIC X(20).
000059            02  FILLER PICTURE X(3).
000060            02  NNAM4O  PIC X(20).
```

Dies ist der Code von "PRAK218.LIB(MSET218)". Er erstreckt sich über mehrere Panels. Wir verwenden es als Vorlage (Template) für die von uns als COBOL-Programm zu erstellende Business Logic.

3. Erstellen des Business Logik Cobol Programms

Wir rufen erneut den Edit-Entry-Panel auf.

```
Menu  RefList  RefMode  Utilities  Workstation  Help
 ------------------------------------------------------------------------------
                               Edit Entry Panel

 ISPF Library:
    Project . . . PRAK218
    Group . . . . CICSDB2  . . .          . . .          . . .
    Type  . . . . COB
    Member  . . . COB218          (Blank or pattern for member selection list)

 Other Partitioned, Sequential or VSAM Data Set:
    Data Set Name . . .
    Volume Serial . . .           (If not cataloged)

 Workstation File:
    File Name . . . . .
                                          Options
 Initial Macro  . . . .                      Confirm Cancel/Move/Replace
 Profile Name . . . . .                      Mixed Mode
 Format Name  . . . . .                      Edit on Workstation
 Data Set Password  . .                      Preserve VB record length

 Command ===>
  F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
 F10=Actions  F12=Cancel
```

Abbildung 5: Anlegen des Members "CPROG218"

Wir legen ein weiteres Member "PRAK218.CICSDB2.COB(COB218)" an (s. Abbildung 8). Es soll unser Business Logic-Programm aufnehmen.

Wir bestätigen mit der Eingabetaste.

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       PRAK218.CICSDB2.COB(COB218) - 01.00            Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100        IDENTIFICATION DIVISION.
000200        PROGRAM-ID. COB218.
000300        ENVIRONMENT DIVISION.
000400        DATA DIVISION.
000500        WORKING-STORAGE SECTION.
000600            EXEC SQL
000700               INCLUDE SQLCA
000800            END-EXEC.
000900        01  NAME-TAB.
001000            02 VORNAME     PICTURE  X(20).
001100            02 NACHNAME    PICTURE  X(20).
001200        COPY MSET218.
001300        LINKAGE SECTION.
001400        PROCEDURE DIVISION.
001500            MOVE LOW-VALUES TO MAP218O.
001600            EXEC SQL
001700              DECLARE C1 CURSOR FOR
001800              SELECT VNAME,NNAME FROM PRAK218.TAB218
001900            END-EXEC.
002000
002100            EXEC SQL  OPEN C1  END-EXEC.
002200
002300            EXEC SQL  FETCH C1 INTO :VORNAME, :NACHNAME   END-EXEC.
002400            MOVE VORNAME TO VNAM1I.
002500            MOVE NACHNAME TO NNAM1I.
002600
002700            EXEC SQL  FETCH C1 INTO :VORNAME, :NACHNAME   END-EXEC.
002800            MOVE VORNAME TO VNAM2I.
002900            MOVE NACHNAME TO NNAM2I.
003000
003100            EXEC SQL  FETCH C1 INTO :VORNAME, :NACHNAME   END-EXEC.
003200            MOVE VORNAME TO VNAM3I.
003300            MOVE NACHNAME TO NNAM3I.
Command ===>                                                 Scroll ===> PAGE
 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Abbildung 6: Der erste Teil des Business Logic-Programms

Dies ist das vollständige COBOL-Programm nach Fertigstellung. Es umfasst mehrere Panels. Mit den F8- bzw. F7-Tasten „scrollen“ wir zwischen den beiden Panels hin und her.

```
File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------
EDIT       PRAK218.CICSDB2.COB(COB218) - 01.00            Columns 00001 00072
003400
003500            EXEC SQL  FETCH C1 INTO :VORNAME, :NACHNAME   END-EXEC.
003600            MOVE VORNAME TO VNAM4I.
003700            MOVE NACHNAME TO NNAM4I.
003800
003900            EXEC SQL  CLOSE C1   END-EXEC.
004000
004100            EXEC CICS SEND MAP('MAP218')
004200                           MAPSET('MSET218')
004300                           ERASE
004400            END-EXEC.
004500
004600            GOBACK.
****** **************************** Bottom of Data ****************************

Command ===>                                                 Scroll ===> PAGE
 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Abbildung 7: Der zweite Teil des Business Logic-Programms

Während der Übersetzung des Mapsets PRAK218.CICSDB2.COB(COBMAP5) wurde ein Member "MSET218" im Dataset "PRAK218.LIB" erstellt, der ein Template für unser COBOL-Programm enthält.

Zeile 1200 unseres COBOL-Programms enthält das Statement

COPY MSET218.

und Zeile 1500

MOVE LOW-VALUES TO MAP218O.

Dieses lädt das vorher angelegte Template automatisch in das COBOL-Programm.

Nach Fertigstellung des Programms kehren wir zum Edit-Entry-Panel zurück.

<u>Aufgabe:</u> ***Erstellen Sie den Member und schreiben Sie das COBOL-Programm hinein. Benutzen Sie als Membernamen COB<Ihre Prak-ID>.***

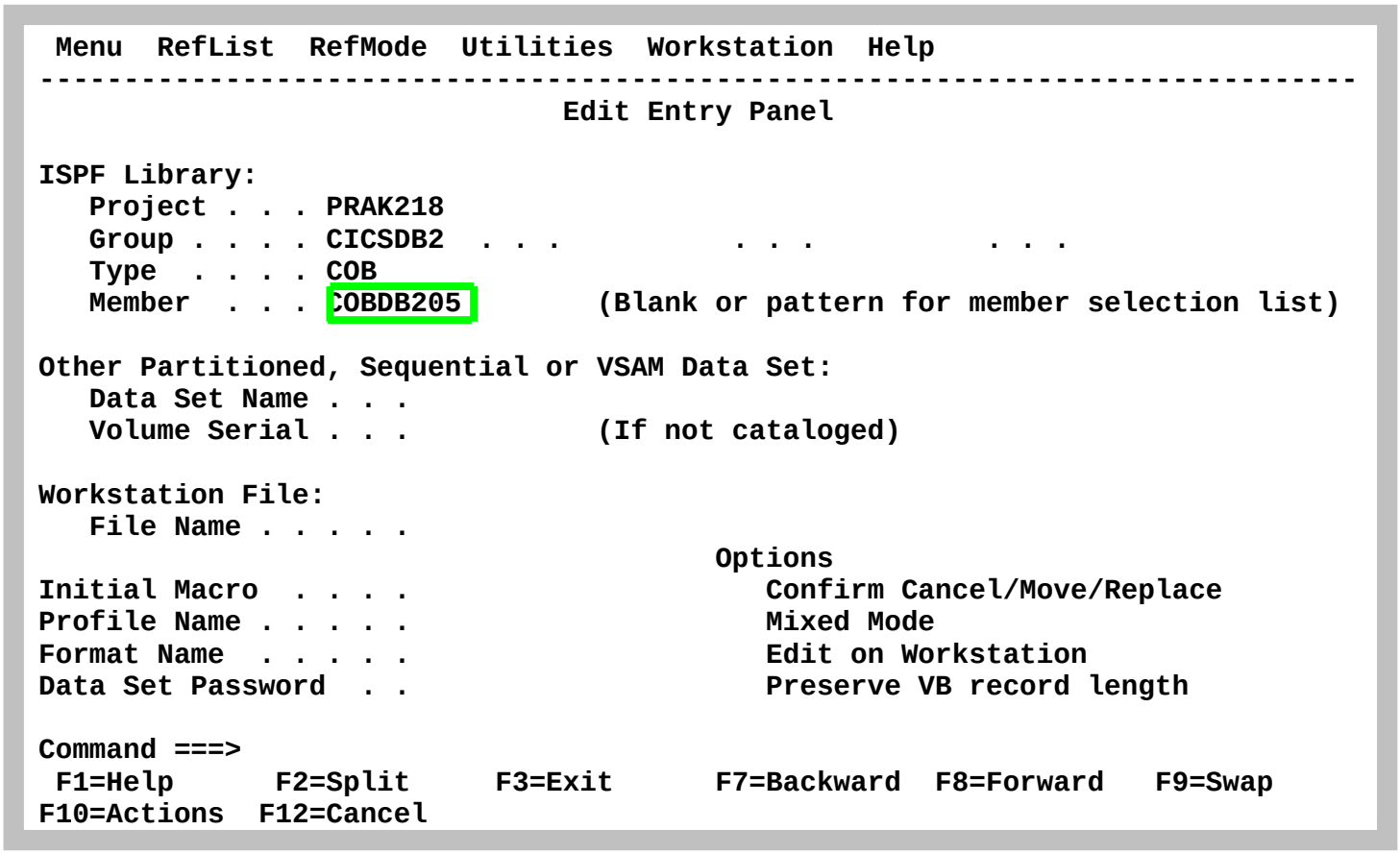


Abbildung 8: Anlegen des Members PCOMPJCL

4. Übersetzung des Cobol Programms

Ehe dieses Programm mit Hilfe des COBOL-Compilers übersetzt werden kann, sind 2 Precompiler-Läufe erforderlich. Der erste Precompiler-Lauf übersetzt alle EXEC SQL-Statements in native DB2 API-Aufrufe.

Wir erstellen ein neues Member „COBDB205" (s. Abbildung 9) zur Aufnahme eines JCL-Scripts, das den SQL-Precompiler aufruft.

```
File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
 -------------------------------------------------------------------------------
 EDIT       PRAK218.CICSDB2.COB(COBDB205) - 01.00          Columns 00001 00072
 ****** ***************************** Top of Data ******************************
 ==MSG> -Warning- The UNDO command is not available until you change
 ==MSG>           your edit profile using the command RECOVERY ON.
 000100 //PRAK218D JOB (),CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1),NOTIFY=&SYSUID,
 000200 //             REGION=4M
 000300 //PCOMP EXEC DB2COB
 000400 //PC.STEPLIB DD DISP=SHR,DSN=SYS1.DSN.V910.SDSNEXIT
 000500 //           DD DISP=SHR,DSN=SYS1.DSN.V910.SDSNLOAD
 000600 //DBRMLIB  DD  DSN=PRAK218.DBRMLIB.DATA(COB218),DISP=SHR
 000700 //SYSCIN   DD  DSN=PRAK218.CICSDB2.COB(PCOMPOUT),DISP=SHR
 000800 //SYSLIB   DD  DSN=PRAK218.CICSDB2.COB,DISP=SHR
 000900 //SYSIN    DD  DSN=PRAK218.CICSDB2.COB(COB218),DISP=SHR
 ****** **************************** Bottom of Data ****************************




 Command ===>                                                 Scroll ===> PAGE
  F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
  F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Abbildung 9: Das JCL-Script

Wir erstellen das in Abbildung 9 dargestellte JCL-Script zum Aufruf des EXEC SQL-Precompilers.

"PRAK218.DBRMLIB.DATA" ist bis jetzt noch leer. Nach der Ausführung des JCL-Scriptes hat der Precompiler einen Member "PRAK218.DBRMLIB.DATA(COB218)" angelegt.

Auf der Kommandozeile geben wir wieder den SUBMIT-Befehl "SUB" ein. Wir warten die Ausführung des JES-Jobs ab (s. Abbildung 10) und bestätigen diese dann mit der Eingabetaste.

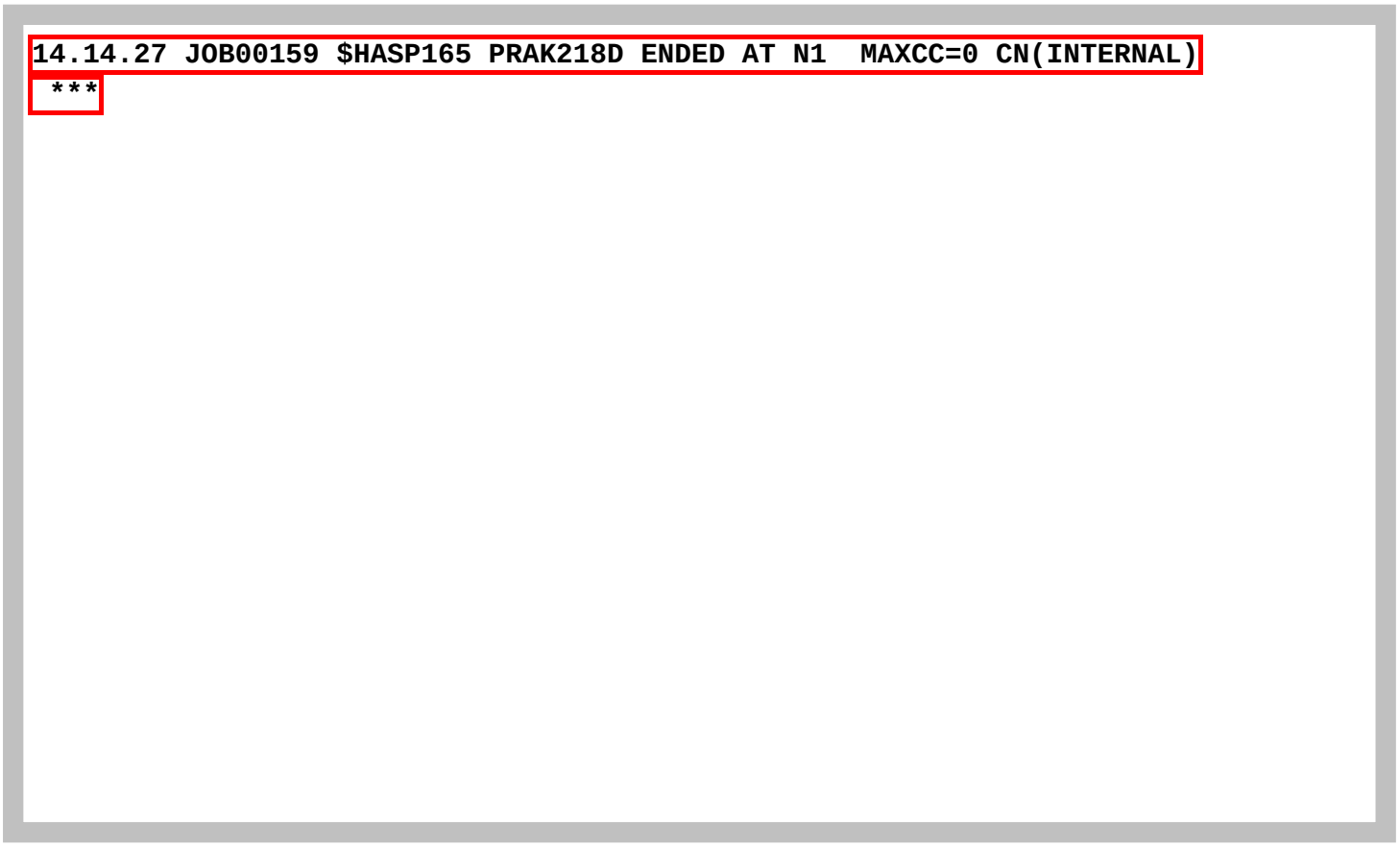


Abbildung 10: Bestätigung der Jobverarbeitung

"MAXCC=0" oder "MAXCC=4" zeigt an, dass der Befehl erfolgreich ausgeführt wurde. Wir bestätigen mit der Eingabetaste.

Aufgabe: ***Erstellen Sie einen neuen Member und schreiben Sie das JCL-Script, das den Precompiler-Aufruf enthält, hinein. Führen Sie es anschließend aus. Denken Sie daran, den Jobnamen ' PRAK218D ' wieder an Ihren Mainframe-Accountnamen anzupassen.***

Als nächstes erstellen wir einen neuen Member "COBSTA05" (s. Abbildung 11) zur Aufnahme eines JCL-Scripts, das folgende Funktionen aufruft:

- **den CICS-Precompiler**
- **den COBOL-Compiler**
- **den Linker**

Anschließend betätigen wir die Eingabetaste.

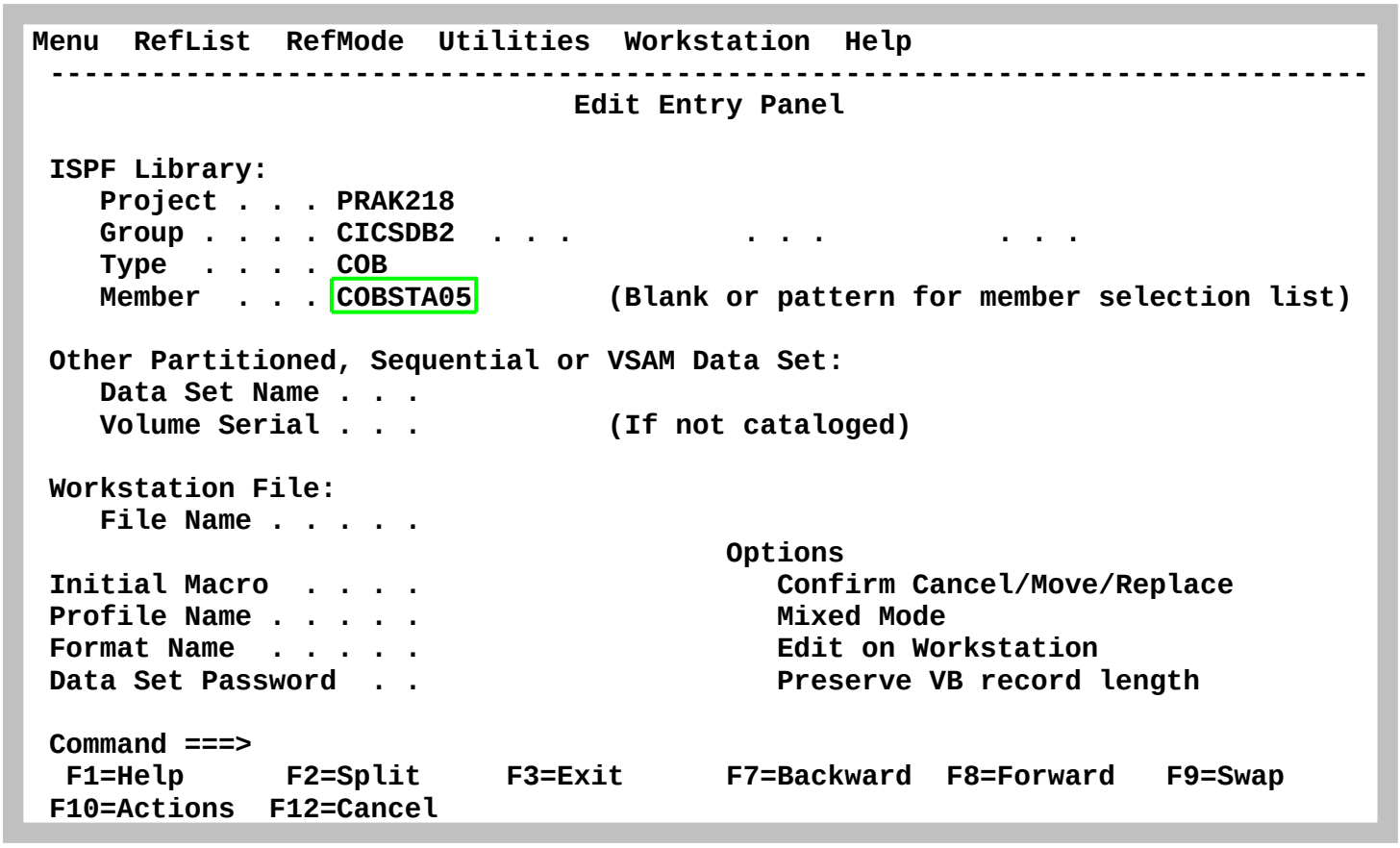


Abbildung 11: Anlegen des Members "COBSTA05"

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------
EDIT       PRAK218.CICSDB2.COB(COBSTA05) - 01.05           Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 //PRAK218C JOB (),CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1),NOTIFY=&SYSUID,
000200 //         REGION=4M,LINES=10,CARDS=1000
000300 //COMP   EXEC COBCICS,PARM.TRN='COBOL3'
000400 //TRN.SYSIN DD DISP=SHR,DSN=PRAK218.CICSDB2.COB(PCOMPOUT)
000500 //COB.SYSLIB DD DSN=PRAK218.LIB,DISP=SHR
000600 //LKED.SYSIN DD *
000700  INCLUDE SYSLIB(DSNCLI)
000800  NAME COB218(R)
001000 //BIND     EXEC PGM=IKJEFT01
001100 //STEPLIB  DD DISP=SHR,DSN=DSN931.DSN.V910.SDSNEXIT
001200 //         DD DISP=SHR,DSN=SYS1.DSN.V910.SDSNLOAD
001300 //DBRMLIB  DD DISP=OLD,DSN=PRAK218.DBRMLIB.DATA(COB218)
001400 //SYSPRINT DD SYSOUT=*
001500 //SYSTSPRT DD SYSOUT=*
001600 //SYSUDUMP DD SYSOUT=*
001700 //SYSTSIN  DD  *
001800   DSN S(D931)
001900   BIND  PLAN(ZGR218CO) MEMBER(COB218) ACTION(REP) RETAIN ISOLATION(CS)
002000   END
002200 //GRANT    EXEC PGM=IKJEFT01
002300 //STEPLIB  DD DISP=SHR,DSN=SYS1.DSN.V910.SDSNLOAD
002400 //SYSPRINT DD SYSOUT=*
002500 //SYSTSPRT DD SYSOUT=*
002600 //SYSUDUMP DD SYSOUT=*
002700 //SYSTSIN  DD *
002800   DSN SYSTEM(D931)
002900   RUN PROGRAM(DSNTIAD) PLAN(DSNTIA91) -
003000       LIBRARY('SYS1.DSN.V910.SDSNLOAD')
003100   END
003200 //SYSIN    DD *
003300 GRANT EXECUTE ON PLAN ZGR218CO TO PUBLIC
003400 /*
****** *************************** Bottom of Data ****************************
Command ===>                                                 Scroll ===> PAGE
 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Abbildung 12: Das JCL-Script (Panel #1)

Der SQL-Precompiler-Lauf hat im Dataset "PRAK218.DBRMLIB.DATA" ein Member "COB218" angelegt (Zeile 800).
Die Ausführung unseres Programms "COB218" unter CICS benötigt einen Zeiger auf die anzusprechende Datenbank-Tabelle (als im JCL-Script als "PLAN" bezeichnet). Wir geben diesem Zeiger den Namen "ZGR218CO" (Zeile 1900 und 3300).

Wir geben "SUB" auf der Kommandozeile ein, warten, bis JES den Job ausgegeben hat und bestätigen anschließend mit der Eingabetaste.

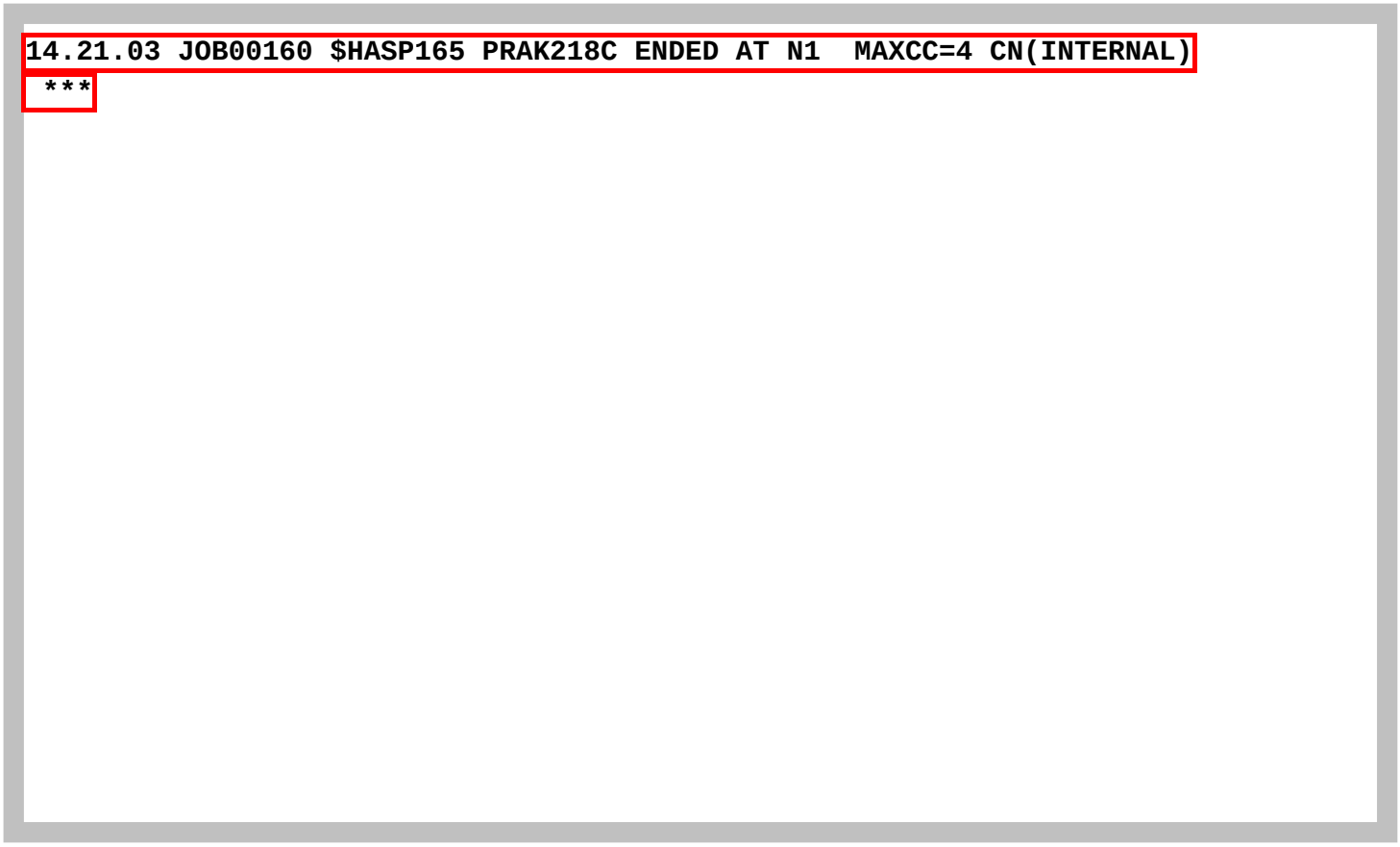


Abbildung 13: Ausgabe der Jobverarbeitung

"MAXCC=4" bedeutet, dass der Compile- und Link-Lauf erfolgreich durchgeführt wurde.

Aufgabe: *Erstellen Sie einen neuen Member, legen Sie das JCL-Script COBSTA05 an (mit an Ihren Accountnamen angepasstem Jobnamen) und führen Sie es aus. Benutzen Sie als Zeiger (Plan) den Bezeichner ZGR<Ihre Prakt-Nr>CO.*

Wir haben nun alle Programme für unsere CICS - DB2-Transaktion erstellt. Als nächsten Schritt müssen sie in dem CICS-Subsystem installiert werden. Hierzu öffnen wir eine weitere Z/OS-Session.

5. Installation im CICS Subsystem

```
TCPIP MSG10 ==> SOURCE DATA SET = SYS1.LOCAL.VTAMLST(USSTCPIP)
z/OS Z18 Level 0609                                IP Address = 134.2.212.63
                                                    VTAM Terminal = SC0TCP18

                      Application Developer System

                               //  OOOOOOO   SSSSS
                              //  OO    OO SS
                      zzzzzz //  OO    OO SS
                        zz  //  OO    OO SSSS
                      zz   //  OO    OO     SS
                    zz    //  OO    OO     SS
                  zzzzzz //   OOOOOOO  SSSS


                  System Customization - ADCD.Z18.*




 ===> Enter "LOGON" followed by the TSO userid. Example "LOGON IBMUSER" or
 ===> Enter L followed by the APPLID
 ===> Examples: "L TSO", "L CICS", "L IMS3270

L CICS
```

Abbildung 14a: Der Login-Screen

Wir loggen uns mit "L CICS" ein (s. Abbildung 14a) und bestätigen mit der Eingabetaste.

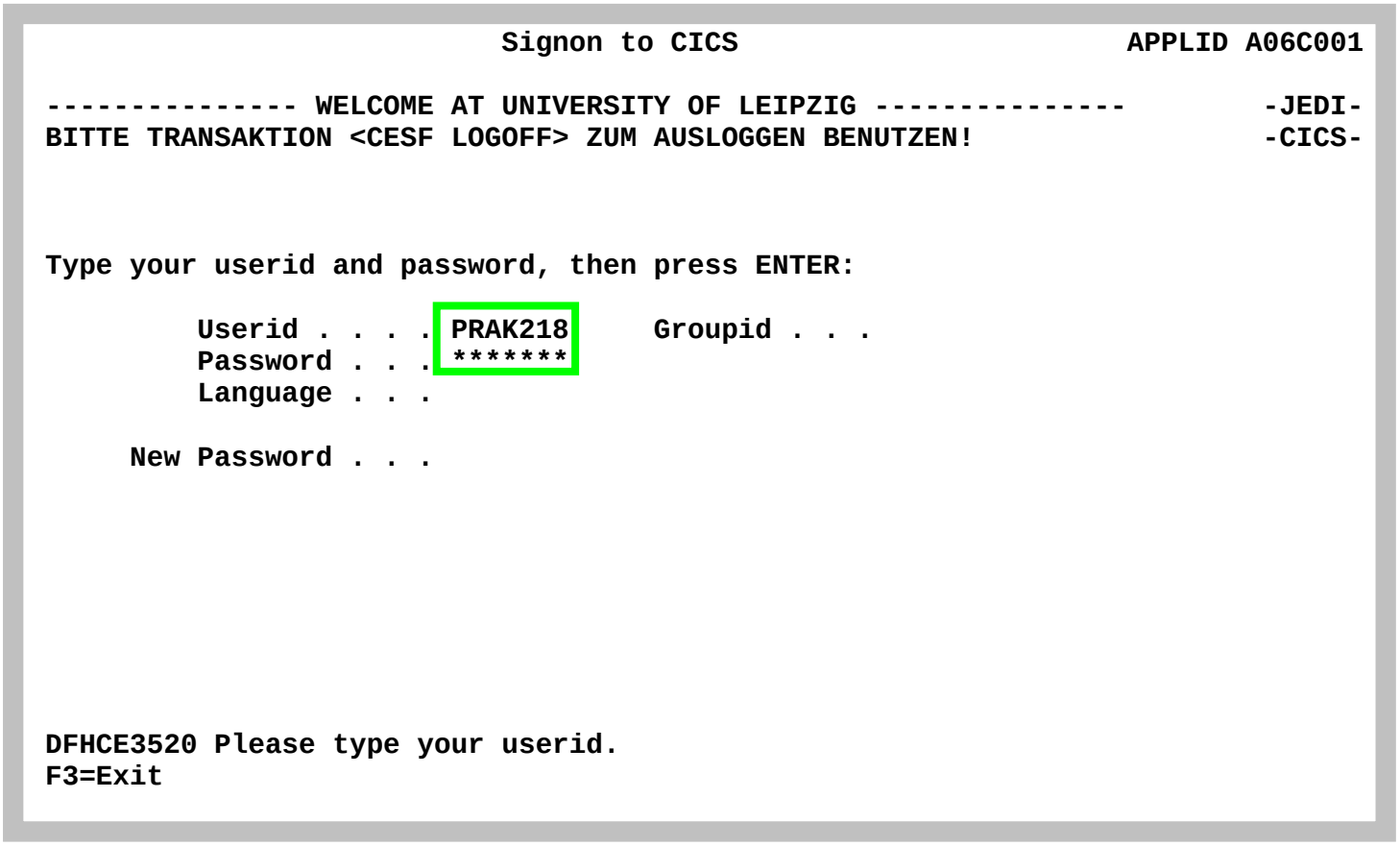


Abbildung 14b: Signon to CICS-Screen

Wir müssen uns unter CICS mit der gleichen Userid wie unter TSO einloggen (s. Abbildung 14b). Auch unser TSO-Password ist in dieses Panel einzugeben. Durch das Betätigen der Eingabetaste kommen wir in den nächsten Screen.

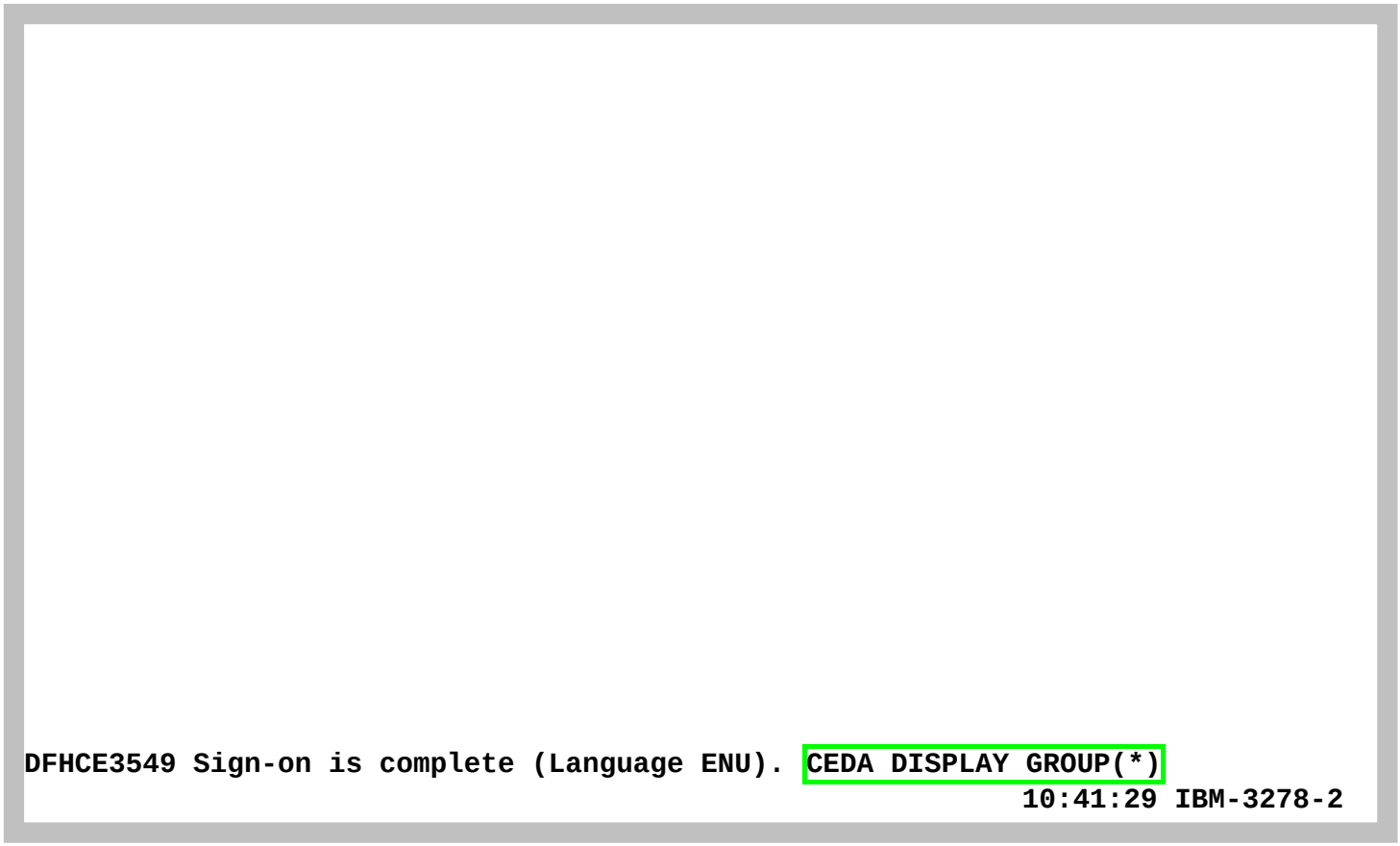


Abbildung 15: Einloggvorgang ist abgeschlossen

Wir betätigen die Tab-Taste, so dass der Cursor auf die letzte Zeile springt (Abbildung 15). Hier geben wir den "CEDA DISPLAY GROUP(*)"-Befehl ein und bestätigen anschließend mit der Eingabetaste.

Der Group-Name kann beliebig gewählt, aber immer nur einmal vergeben werden. Der Übersichtlichkeit wegen ist es sinnvoll, den Login-Namen zu verwenden. Wir haben aber bereits die Gruppe PRAK218 in Tutorial 3 verwendet. Wir löschen deshalb die Gruppe PRAK218 mit dem Befehl

CEDA DELETE ALL GROUP(PRAK218)

Und verifizieren danach mit CEDA DISPLAY GROUP(*) dass dies auch tatsächlich geschehen ist.

An dieser Stelle ist es jetzt notwendig, eine neue Gruppe anzulegen. Wir definieren zunächst unser BMS-Programm mit dem Namen "MSET218" für die neue Group "PRAK218" und betätigen anschließend dreimal die Eingabetaste.

```
 CEDA DEFINE MAPSET(MSET218) GROUP(PRAK218)
 OVERTYPE TO MODIFY                                      CICS RELEASE = 0640
  CEDA  DEFine Mapset( MSET218  )
   Mapset         : MSET218
   Group          : PRAK218
   Description  ==>
   REsident     ==> No                 No | Yes
   USAge        ==> Normal             Normal | Transient
   USElpacopy   ==> No                 No | Yes
   Status       ==> Enabled            Enabled | Disabled
   RSl            : 00                 0-24 | Public









   I New group PRAK218 created.
                                                  SYSID=CICS APPLID=CICS
   DEFINE SUCCESSFUL                          TIME:  14.27.04  DATE: 12.120
PF 1 HELP 2 COM 3 END           6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Abbildung 16: Definition des Mapsets "SET5218"

Die Definition war erfolgreich und die neue Gruppe wurde erstellt.

Als nächstes wird das COBOL-Programm definiert. Dazu drücken wir die Eingabetaste.

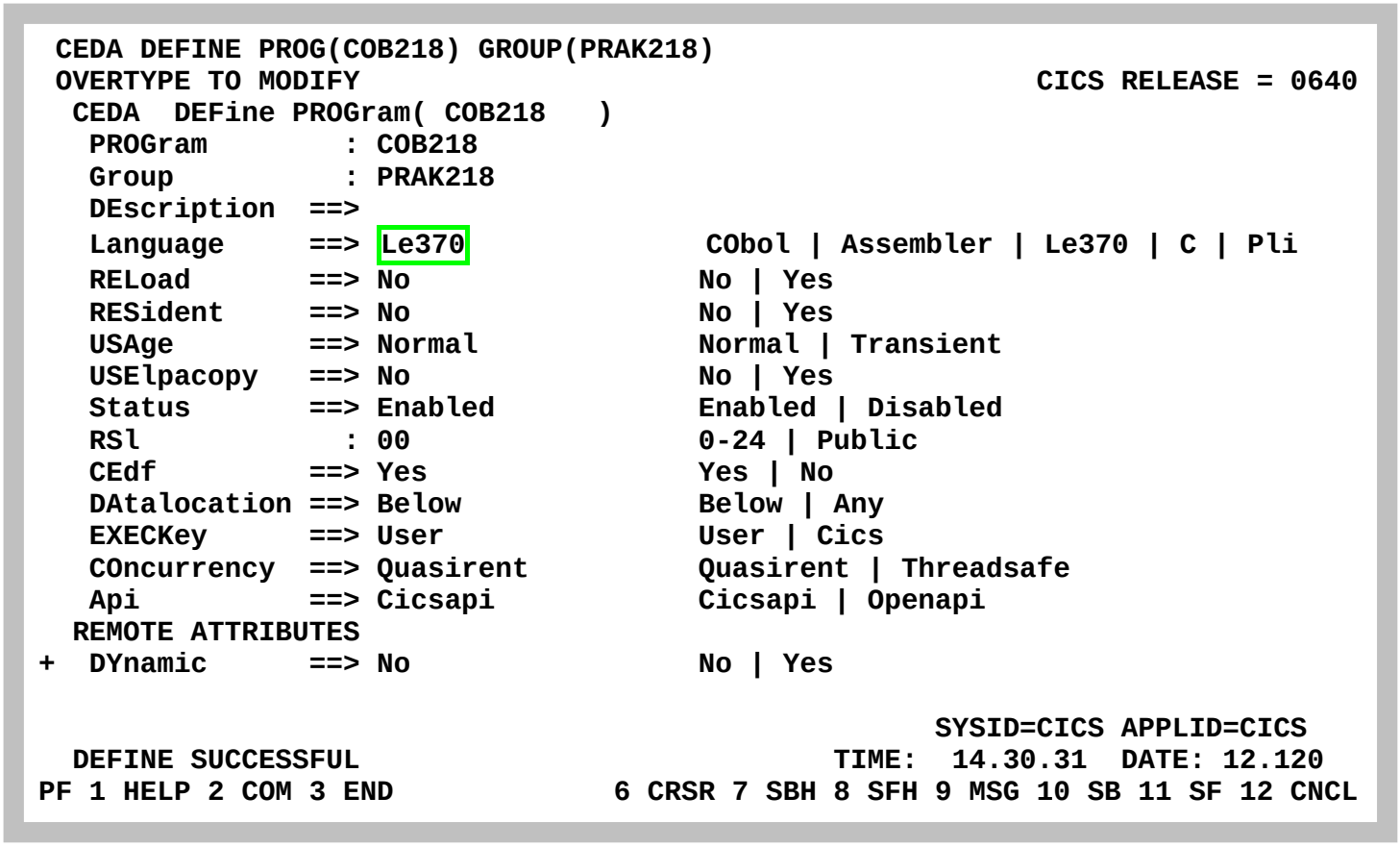


Abbildung 17: Auswahl der Parameter

Le370 wird bei Language eingeben; sie ist aber eigentlich eine Entwicklungsumgebung (s. Abbildung 17)

Auch hier bestätigen wir mit der Eingabetaste.

Die Nachricht "DEFINE SUCCESSFUL" erscheint; wir beenden diese Aktion mit Betätigung der F3-Taste.

```
CEDA DEFINE TRANS(X218) GROUP(PRAK218)
  STATUS:  SESSION ENDED
```

Abbildung 18: Sitzung beendet

Als letztes müssen wir die Bezeichnung der neuen Transaktion definieren. Wir wählen auch hierfür den Namen "X218". Der Namen der Transaktion muss immer aus vier Zeichen bestehen. Man könnte den Namen auch anders wählen. Für die Teilnehmer sind die Rechte jedoch so vergeben, dass die Transaktion nur wie unten angegeben angelegt werden kann. Wir geben das Kommando "CEDA DEFINE TRANS(X218) GROUP(PRAK218)" ein (s. Abbildung 18) und bestätigen mit der Eingabetaste.

In die Zeile "PROGram" geben wir nun "COB218" (Abbildung 19) ein und bestätigen dies mit der Eingabetaste.

```
 DEFINE TRANS(X218) GROUP(PRAK218)
 OVERTYPE TO MODIFY                                      CICS RELEASE = 0640
  CEDA  DEFine TRANSaction( X218 )
   TRANSaction  ==> X218
   Group        ==> PRAK218
   DEscription  ==>
   PROGram      ==> COB218
   TWasize      ==> 00000             0-32767
   PROFile      ==> DFHCICST
   PArtitionset ==>
   STAtus       ==> Enabled           Enabled | Disabled
   PRIMedsize     : 00000             0-65520
   TASKDATALoc  ==> Below             Below | Any
   TASKDATAKey  ==> User              User | Cics
   STOrageclear ==> No                No | Yes
   RUnaway      ==> System            System | 0 | 500-2700000
   SHutdown     ==> Disabled          Disabled | Enabled
   ISolate      ==> Yes               Yes | No
   Brexit       ==>
+ REMOTE ATTRIBUTES
  S PROGRAM OR REMOTESYSTEM MUST BE SPECIFIED.
                                                    SYSID=CICS APPLID=CICS

PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Abbildung 19: Definition der Transaktion

"DEFINE SUCCESSFUL" erscheint; also war die Definition erfolgreich, wir beenden sie mit der F3-Taste.

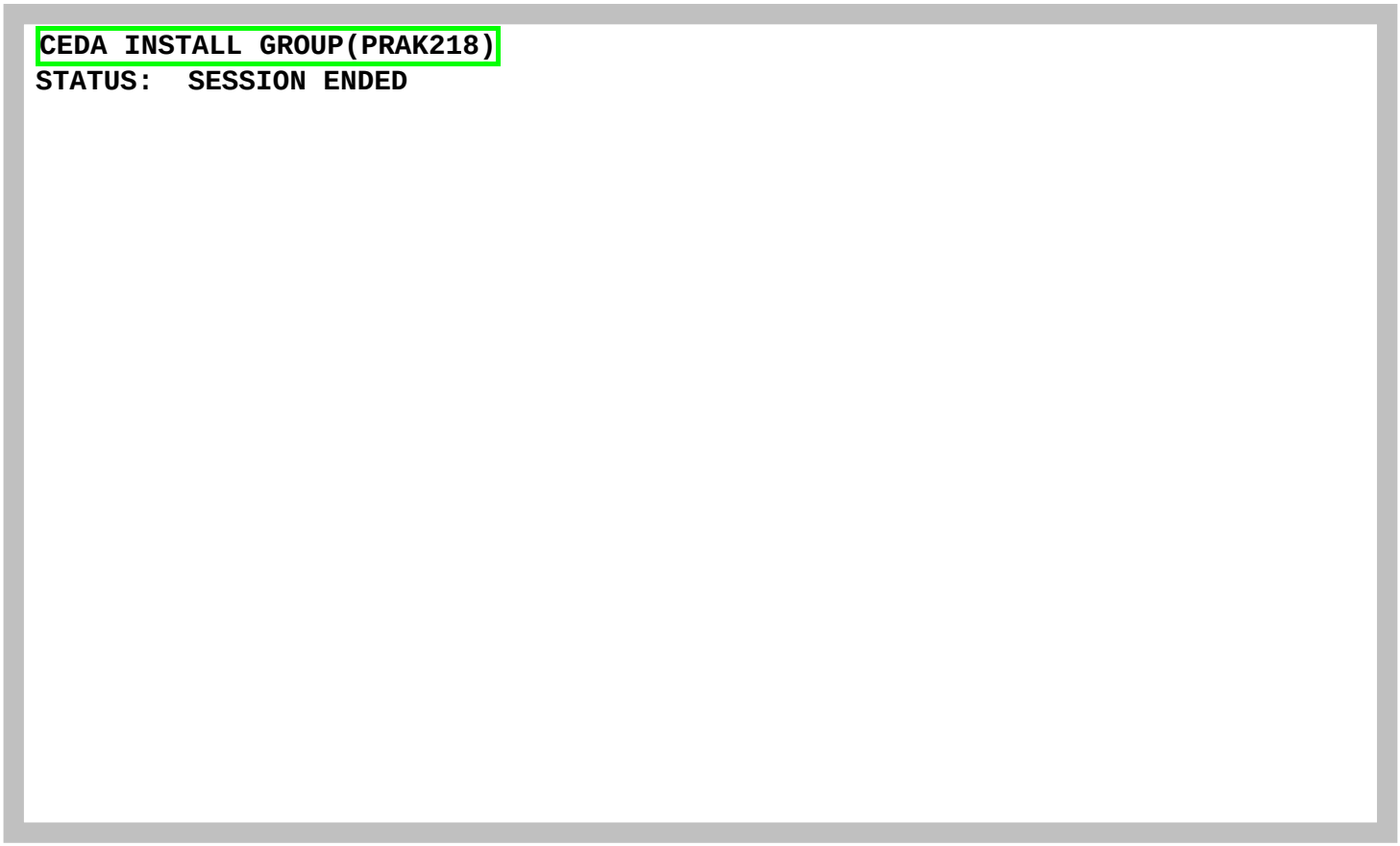


Abbildung 20: Installation der Gruppe

Nachdem die BMS-MAP, das COBOL-Programm und die Transaktionsbezeichnung definiert worden sind, wird nun alles in unserer Gruppe "PRAK218" installiert. Dazu geben wir den Befehl "CEDA INSTALL GROUP(PRAK218)" ein (s. Abbildung 21) und bestätigen mit der Eingabetaste.

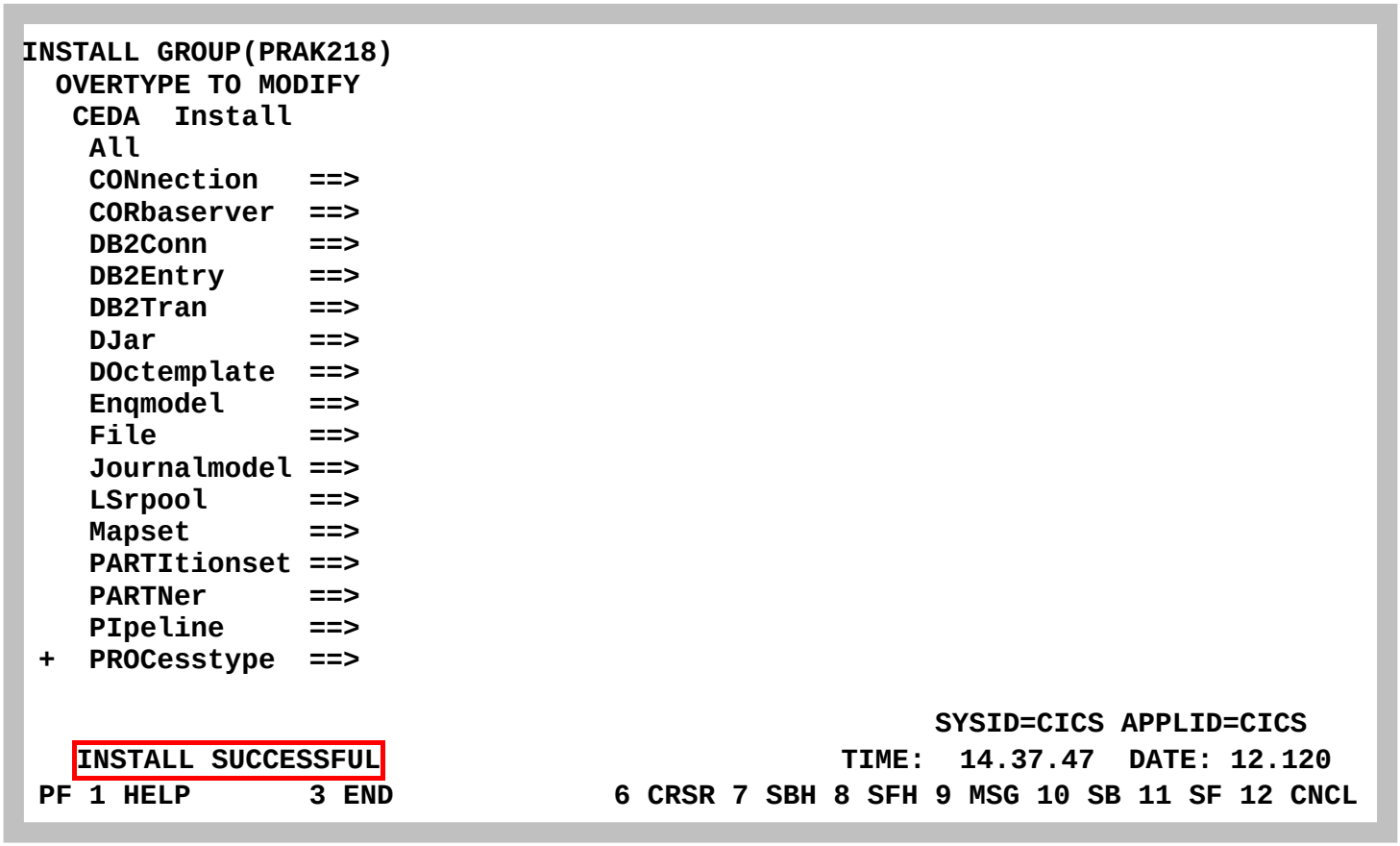


Abbildung 21: Installation war erfolgreich

Die erfolgreiche Installation der Gruppe "PRAK218" zeigt die Ausgabe "INSTALL SUCCESSFUL" (s. Abbildung 21) an. Wir beenden diese Installation, indem wir die F3-Taste drücken.

Es kann sein, dass an dieser Stelle die Meldung „Install unsuccessful“ erscheint. Eine Erläuterung dieses Fehlers finden Sie im Anhang dieses Tutorials.

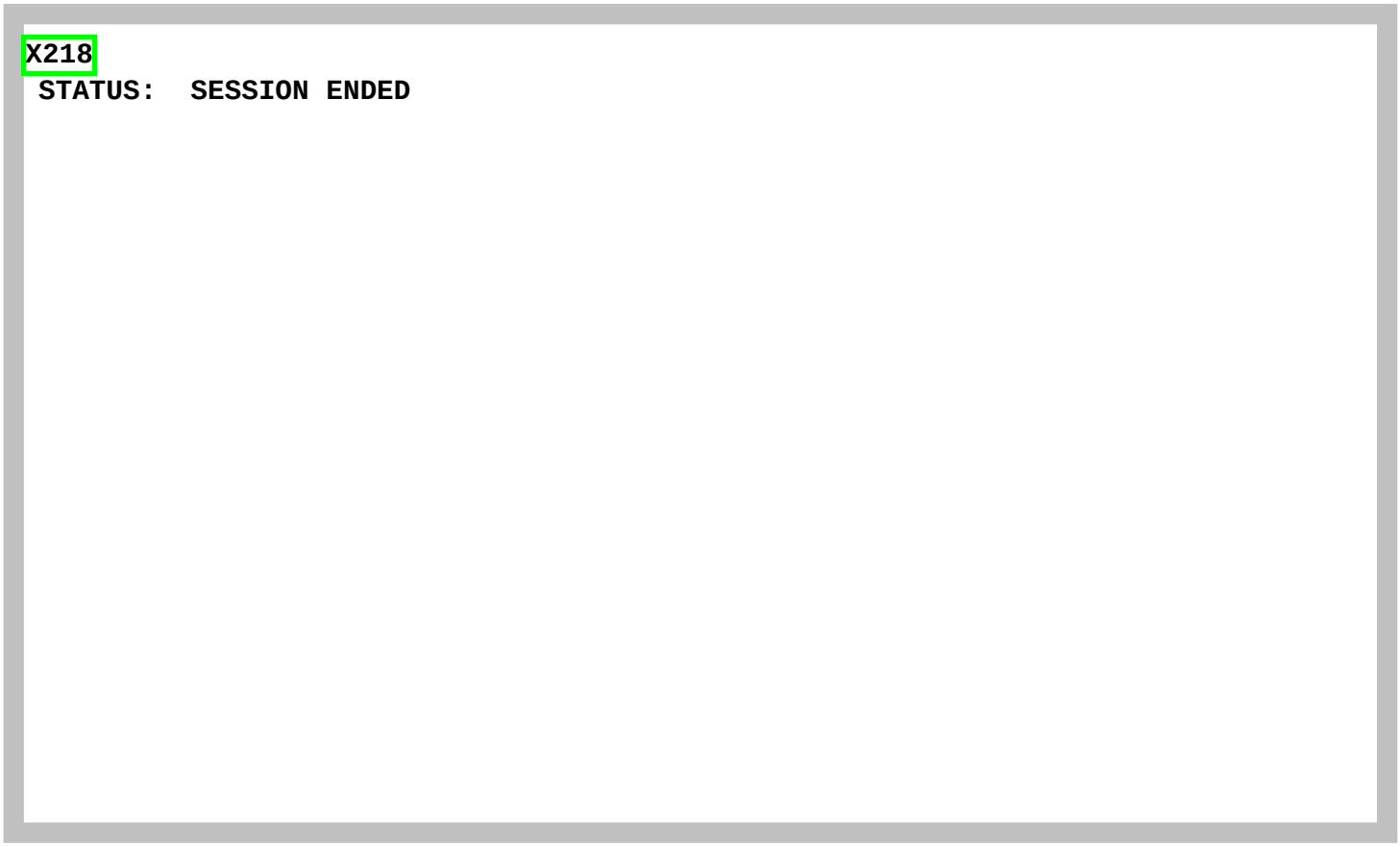


Abbildung 22: Aufruf der Transaktion

In Tutorial 3 waren wir mit der Definition und Installation unserer Transaktion fertig. Wir versuchen es einmal, indem wir unsere Transaktion mit der Bezeichnung "X218" aufrufen. Dazu tragen wir den Namen in die CICS-Kommandozeile ein (s. Abbildung 22) und bestätigen mit der Eingabetaste.

```
DFHAC2220 20:49:29 A06C001 The coordinator system has indicated that the
  current unit of work is to be backed out. Transaction X218 has been
  abnormally terminated with abend ASP3.
```

Abbildung 23: Fehlermeldung

Wir erhalten eine Fehlermeldung (s. Abbildung 23).
Manchmal erscheint auch eine andere Fehlermeldung als die in Abbildung 23 dargestellte.

Die Beschreibung zur Fehlermeldung ASP3 findet sich im CICS IBM Online-Handbuch mit dem folgenden Eintrag:

> **Explanation: The abnormal termination occurs because a remote system on which the unit of work depends fails to take a syncpoint. The transaction cannot commit its changes until all coupled systems to which function has been transmitted also commit. This may be because the syncpoint protocol for transaction to transaction has been violated by failing to be in send mode for all sessions for which syncpoint has not been received.**
> **User Response:**
>
> **Check why the remote system failed to respond to the request.**

TSO, CICS und DB2 sind unterschiedliche z/OS-Subsysteme, die in getrennten virtuellen Adressräumen laufen. Die CICS-Gruppe "PRAK218" benötigt eine Definition unserer Datenbank und Datenbanktabelle.

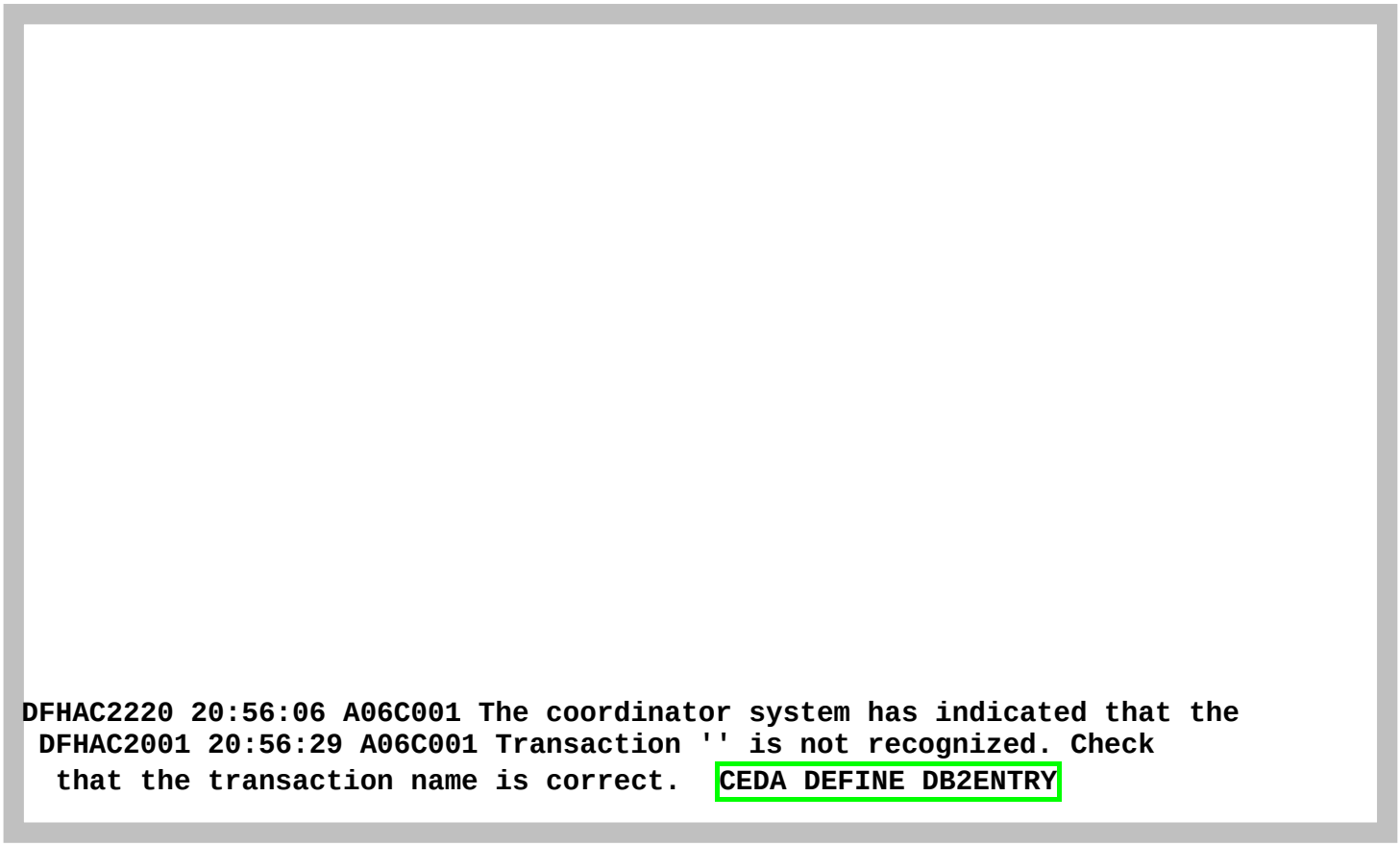


Abbildung 24: Aufruf der Definition der Datenbank

Die Definition erfolgt mit dem Kommando "CEDA DEFINE DB2ENTRY" (s. Abbildung 24).

Es kann auch sein, dass das System sich an dieser Stelle aufhängt. Resultat: keine Tastatureingabe ist möglich, und links unten erscheint ein Strich-Männchen. Drücken der PF2 Taste behebt dies Problem.

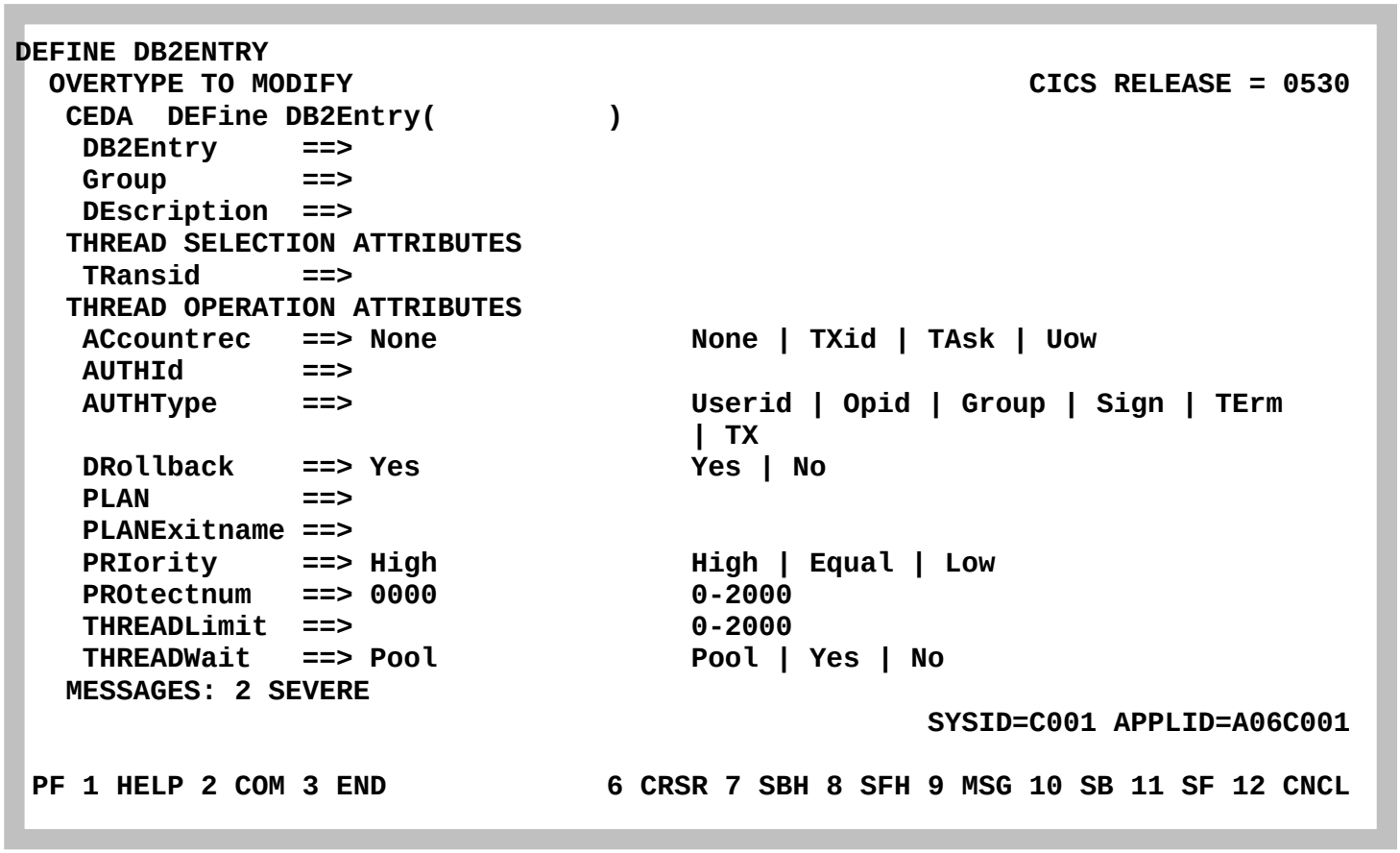


Abbildung 25: DEFINE DB2ENTRY-Panel

Nachdem wir die Eingabetaste gedrückt haben, erscheint der "DEFINE DB2ENTRY-Panel" (s. Abbildung 25). Wir müssen die fehlenden Angaben eintragen und betätigen abschließend die Eingabetaste (s. Abbildung 26).

```
define DB2ENTRY
 OVERTYPE TO MODIFY                                        CICS RELEASE = 0530
  CEDA  DEFine DB2Entry(          )
   DB2Entry     ==> X218
   Group        ==> PRAK218
   DEscription  ==>
  THREAD SELECTION ATTRIBUTES
   TRansid      ==> X218
  THREAD OPERATION ATTRIBUTES
   ACcountrec   ==> TXid             None | TXid | TAsk | Uow
   AUTHId       ==>
   AUTHType     ==> Sign             Userid | Opid | Group | Sign | TErm
                                      | TX
   DRollback    ==> Yes               Yes | No
   PLAN         ==> ZGR218CO
   PLANExitname ==>
   PRIority     ==> High              High | Equal | Low
   PROtectnum   ==> 0000              0-2000
   THREADLimit  ==> 0003             0-2000
   THREADWait   ==> Yes         Pool | Yes | No
  MESSAGES: 2 SEVERE
                                                  SYSID=C001 APPLID=A06C001

 PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Abbildung 26: Eingabe der Parameter

Wir bezeichnen den DB2-Zugriff (DB2Entry) mit dem Namen "X218". Das Ganze wird Teil der Gruppe "PRAK218". Unsere Transaction-ID (TRansid) ist "X218". Wir hatten ein JCL-Script "COBSTA05" erstellt, das unser COBOL-Programm übersetzte. In diesem Script definierten wir an zwei Stellen einen Zeiger auf unsere Datenbanktabelle (Plan) mit dem Namen "ZGR218CO". Hier wird jetzt für CICS die Verknüpfung zu der Datenbanktabelle hergestellt.

```
 OVERTYPE TO MODIFY                                          CICS RELEASE = 0530
  CEDA  DEFine DB2Entry( X218     )
   DB2Entry       : X218
   Group          : PRAK218
   DEscription  ==>
  THREAD SELECTION ATTRIBUTES
   TRansid      ==> X218
  THREAD OPERATION ATTRIBUTES
   ACcountrec   ==> TXid              None | TXid | TAsk | Uow
   AUTHId       ==>
   AUTHType     ==> Userid            Userid | Opid | Group | Sign | TErm
                                      | TX
   DRollback    ==> Yes               Yes | No
   PLAN         ==> ZGR218CO
   PLANExitname ==>
   PRIority     ==> High              High | Equal | Low
   PROtectnum   ==> 0000              0-2000
   THREADLimit  ==> 0003              0-2000
   THREADWait   ==> Yes               Pool | Yes | No

                                                  SYSID=C001 APPLID=A06C001
   DEFINE SUCCESSFUL                          TIME:  00.00.00  DATE: 01.061
PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Abbildung 27: Bestätigung der gelungenen Definition

Die Definition war erfolgreich und wird bestätigt durch die Ausgabe: "DEFINE SUCCESSFUL" (s. Abbildung 27).

Wir verlassen die Definition mit der F3-Taste.

```
CEDA INSTALL GROUP(PRAK218)
 STATUS:  SESSION ENDED
```

Abbildung 28: Installation der Gruppe

Diese Änderung muss wieder installiert werden. Dazu geben wir wieder den Befehl "CEDA INSTALL GROUP(PRAK218)" (s. Abbildung 28) ein und bestätigen mit der Eingabetaste.

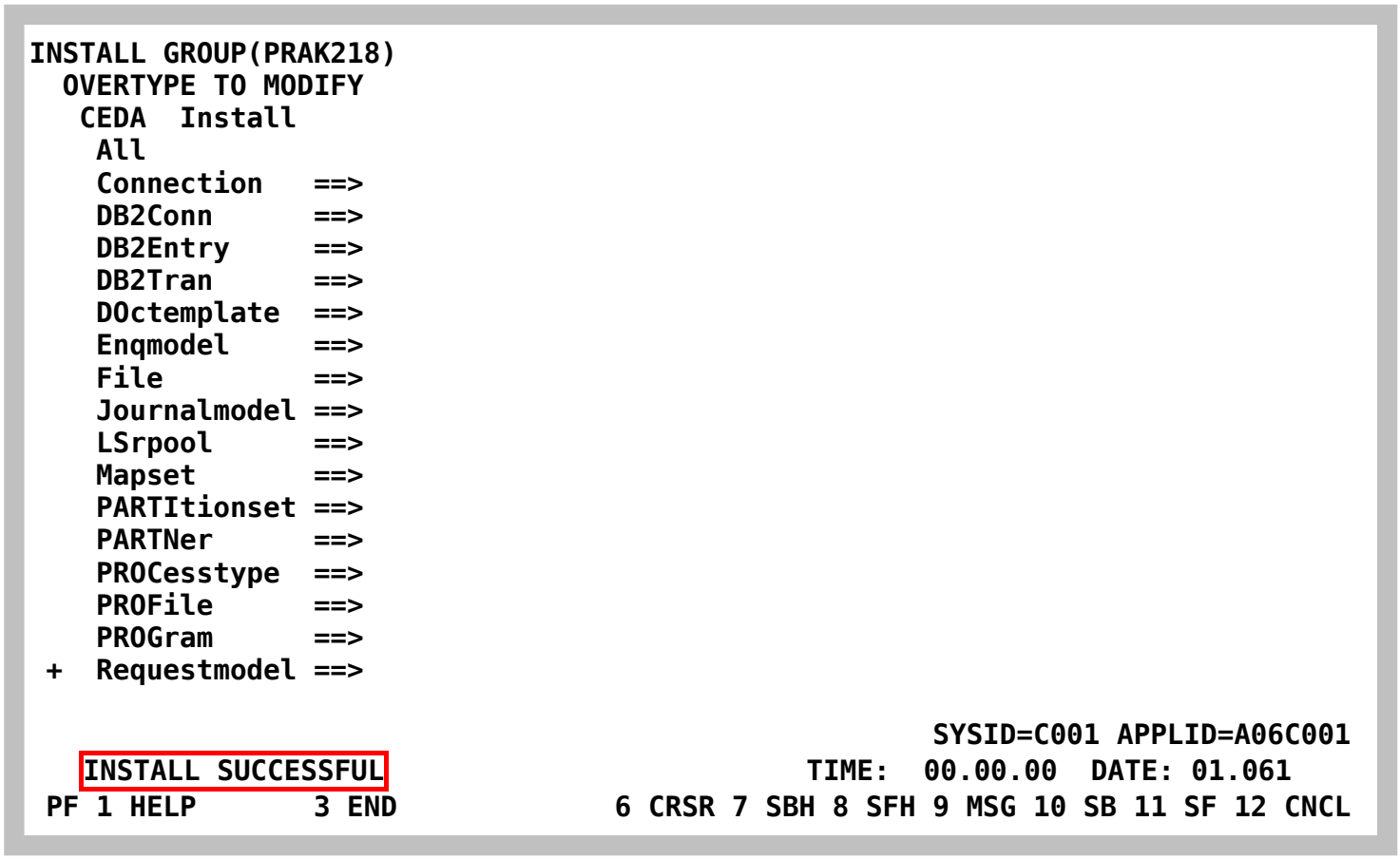


Abbildung 29: Installation der Gruppe

Die Ausgabe „INSTALL SUCCESSFUL“ in der Abbildung 29 sagt aus, dass die Installation erfolgreich war. Wir verlassen diesen Screen wieder mit F3.

6. Ausführen der Transsaktion

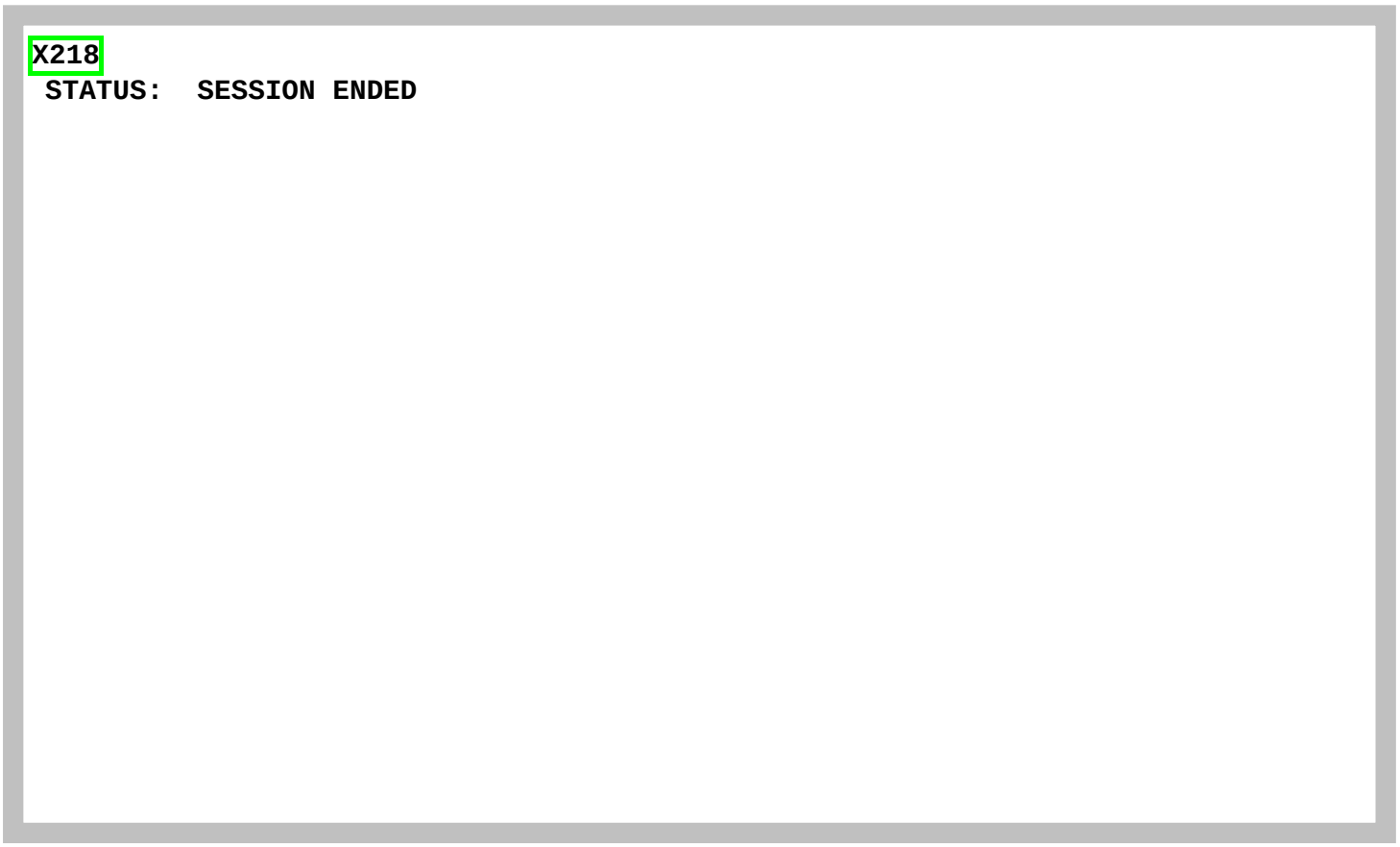


Abbildung 30: Starten der Transaktion

Wir geben den Namen unserer Transaktion "X218" ein, um diese aufzurufen (s. Abbildung 30) und bestätigen mit der Eingabetaste.

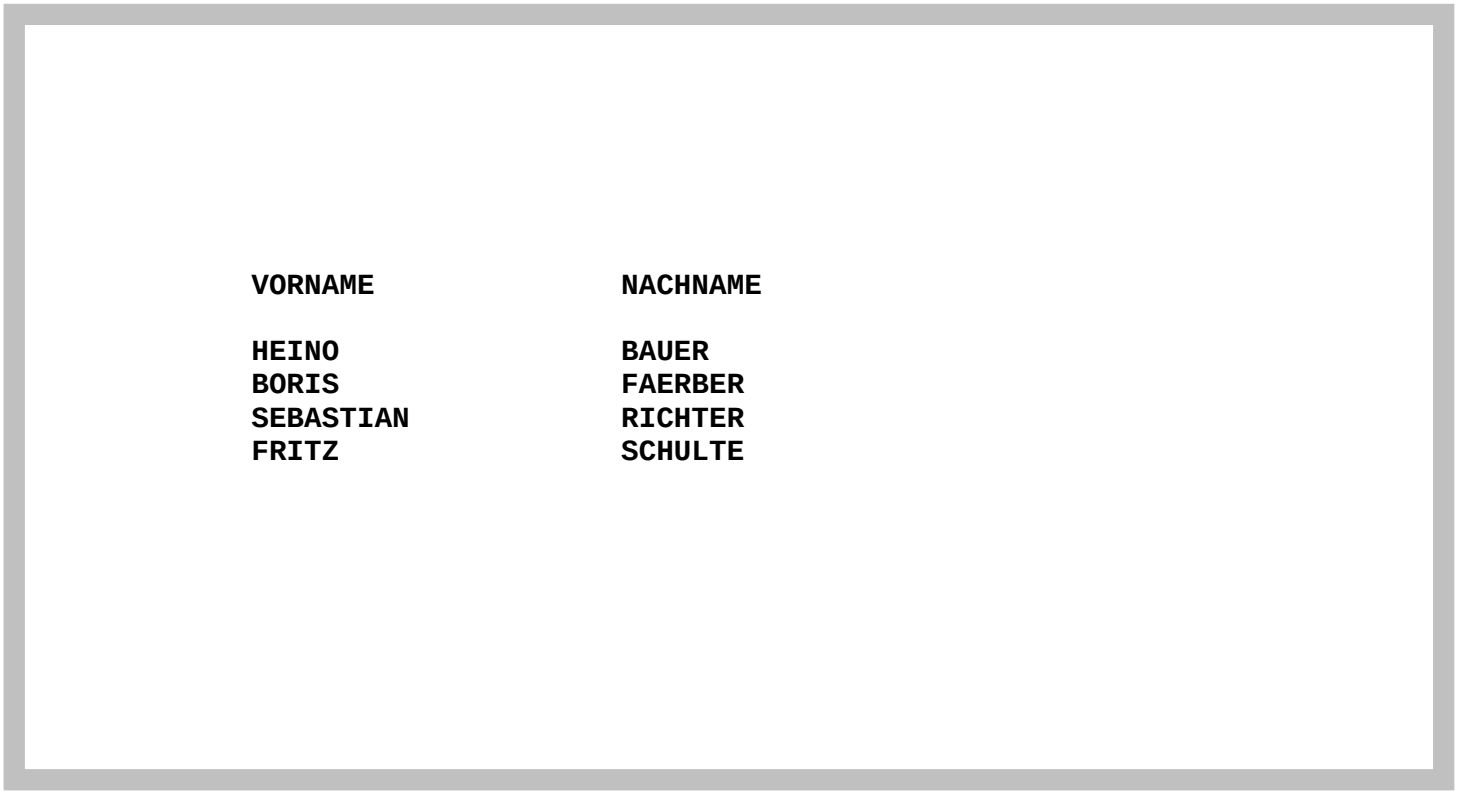


Abbildung 31: Ausgabe der Datenbanktabelle

Die korrekte Ausgabe der Datenbank erscheint auf dem Bildschirm (s. Abbildung 31).

Aufgabe: ***Bereiten Sie unter CICS die Transaktion vor, die auf die DB2-Datenbank zugreifen soll und führen Sie diese anschließend aus. Benutzen Sie dabei als CICS-Gruppen-Namen Ihren Accountnamen, also z.B. PRAK145 oder PRAK162. Die DB2-Datenbank soll Ihren Namen / Ihre Namen enthalten. Benutzen Sie als Transactions-ID "X<Ihre Prakt-ID>". Bezeichnen Sie den DB2ENTRY identisch zu Ihrer Transaction-ID.***

Erzeugen Sie einen Screenshot (unter Windows durch den Shortcut ALT-Druck) Ihrer Version der Abbildung 31 und schicken Sie diesen Ihrem Betreuer per Mail zu. Der Screenshot darf eine Größe von 250 KByte nicht überschreiten, benutzen Sie möglichst das JPG-Format, dass mit Dateigrößen unter 90 KByte auskommt. Löschen Sie nichts von Ihrer Lösung, so dass Ihr Betreuer Ihre Transaktion aufrufen kann.

Aufgabe: ***Gehen Sie vom CUSTOMPAC MASTER APPLICATION MENU aus in die System Display and Search Facility. Im erscheinenden SDSF PRIMARY OPTION MENU wählen Sie die Option ST. Löschen Sie alle angezeigten Jobs, die sich in der PRINT-Queue befinden, indem Sie links neben einen jeden Jobnamen "p" (purge) eintragen und anschließend die Eingabetaste (mehrfach) drücken. Einen Job dürfen Sie natürlich nicht löschen: Den einen, der sich in der EXECUTION-Queue befindet. Denn das ist der Job, mit dem Sie zur Zeit eingeloggt sind.***

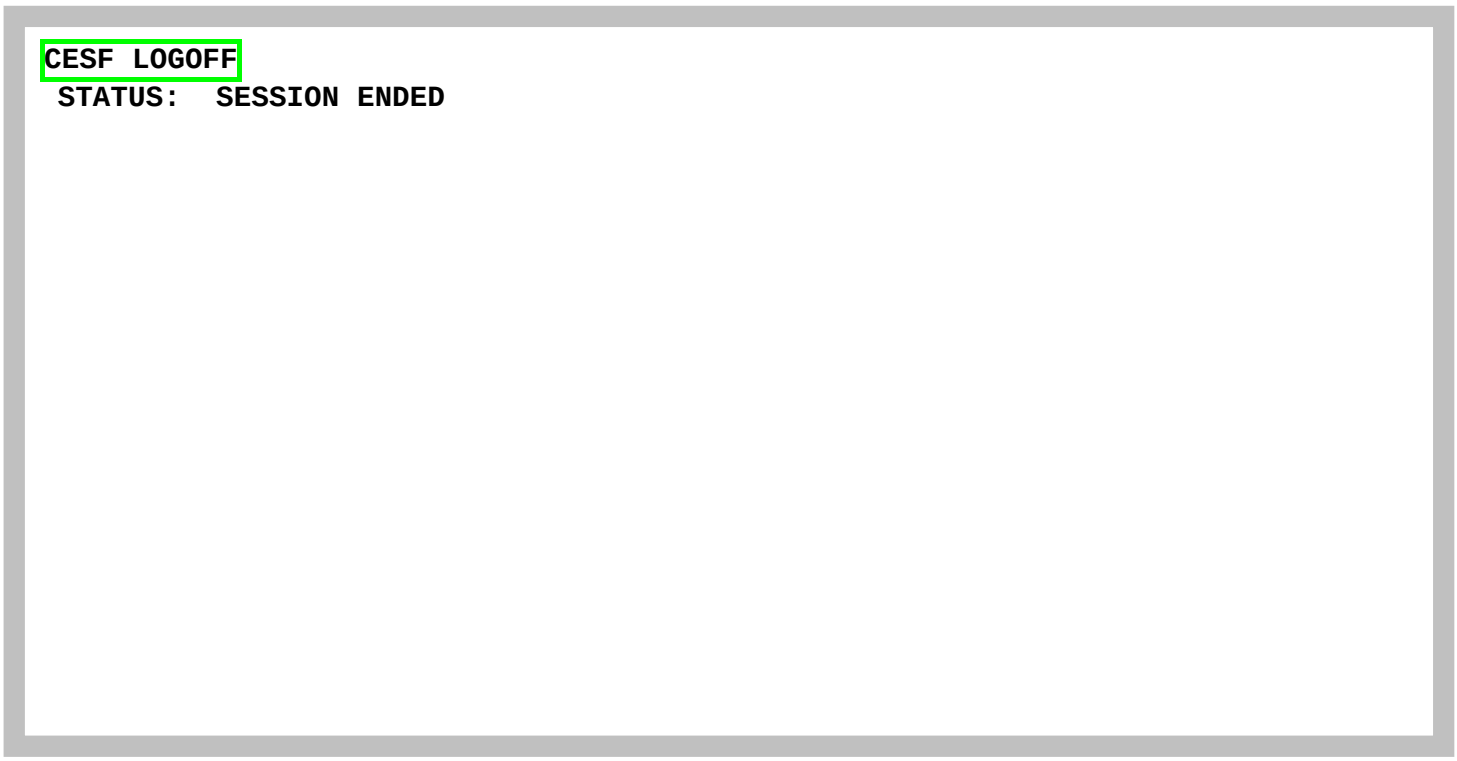


Abbildung 32: Ausloggen aus CICS

Die Ausführung unserer Transaktion (unseres COBOL-Programms) ist damit abgeschlossen – CICS erwartet jetzt die Eingabe einer neuen Transaktion. Dies könnte z.B. CEDA DISPLAY GROUP(*) sein.

Wenn wir mit unserer CICS Sitzung fertig sind und keine weitere Transaktion durch Eingabe einer TRID starten wollen, geben wir die Logoff-Transaktion "CESF LOGOFF" ein, gefolgt von der Eingabetaste, ein (s. Abbildung 32).

7. Anhang

Die Übersetzung des COBSTA05 erzeugt MAXCC=8

Dieses Problem kann aufgetreten sein, weil der in unserem JCL-Script verwendete Zeiger schon existiert.

Sehen sie hierzu im ISPF-Menu More -> SDSF -> Status of jobs nach, welchen Fehler sie erhalten. Sollte es tatsächlich daran liegen, dass der Zeiger schon existiert, so finden sie hier die Meldung:

```
READY
  DSN S(D931)
DSN
  BIND PLAN(ZGR218CO) MEMBER(COB218) ACTION(REP) RETAIN ISOLATION(CS)
DSNT210I  -D931 BIND AUTHORIZATION ERROR
          USING PRAK218 AUTHORITY
          PLAN = ZGR218
          PRIVILEGE = BIND
DSNT201I  -D931  BIND FOR PLAN ZGR218CO   NOT SUCCESSFUL
DSN
END
READY
END
```

In diesem Fall überlegen sie sich bitte einen anderen Namen für ihren Zeiger. Verwenden sie jedoch immer die Ziffern ihres Accounts.

"CEDA INSTALL GROUP ..." erzeugt den Fehler "install unsuccessful"

Dieses Problem könnte auftreten, wenn jemand mehrmals den Befehl "CEDA INSTALL GROUP ..." eingibt. Als Fehlermeldung wird INSTALL UNSUCCESSFUL zurückgegeben.

Das Problem ist, dass sich die schon einmal per "CEDA INSTALL GROUP ..." installierte DB2ENTRY-Komponente nicht so ohne weiteres überschreiben lässt.

Man muss das Überschreiben erlauben. Dies erfordert aber Administrator Rechte über die Ihre User ID nicht verfügt. Deshalb versuchen Sie bitte, das Problem zu vermeiden, indem ein mehrfaches Abarbeiten der Tutorials 5 vermieden wird.

Sollte das Problem trotzdem einmal auftreten, informieren Sie bitte jemanden mit Admin-Rechten, z. B. Ihren Betreuer.

Zu Ihrer Information:

Das Problem kann mit Administrator Rechten behoben werden durch die Eingabe:

CEMT I DB2E(<DB2E-Name>)

Dies gibt den DB2Entry mit dem Namen <DB2E-Name> auf dem Bildschirm aus.

Im konkreten Beispiel liefert

CEMT I DB2E(A060) die folgende Bildschirmausgabe:

```
I DB2E(A060)
STATUS:  RESULTS - OVERTYPE TO MODIFY
 Db2e(A060    ) Txi Sig Ena Poo Hig Pro( 0000 ) Pth(0000)
    Threadl( 0003 ) Threads(0000) Twa Plan( AS5      )
```

Der Wert "Ena" (ENAble) ist auf "Dis" (DISable) zu setzen, um ein Überschreiben zu erlauben. Dazu reicht es, wenn man den Buchstaben "E" von "Ena" mit einem "D" überschreibt sowie die Eingabetaste betätigt. Das Ergebnis dieser Aktion ist im konkreten Beispiel

```
I DB2E(A060)
STATUS:  RESULTS - OVERTYPE TO MODIFY
 Db2e(A060    ) Txi Sig Dis Poo Hig Pro( 0000 ) Pth(0000)
 NORMAL
    Threadl( 0003 ) Threads(0000) Twa Plan( AS5      )
```

Nun ist ein Überschreiben des DB2Entry-Eintrages "A060" wieder möglich und damit auch eine Neuinstallation der Gruppe PRAK218 wieder möglich:

CEDA INSTALL GROUP(PRAK218)

funktioniert fehlerfrei und gibt wieder

INSTALL SUCCESSFUL zurück.

Tutorial MQ 01
Einführung in WebSphere MQ

In dieser Aufgabe legen wir uns eine Message Queue an, schicken an diese Nachrichten und lesen diese danach wieder aus.

Hinweis: Dieses Tutorial wurde unter Verwendung der Benutzer-ID "PRAK222" erstellt. In allen Dateinamen müssen Sie "PRAK222" durch ihre eigene Benutzer-ID ersetzen. Außerdem wird oft in den Erklärungen PRAKxxx verwendet, wobei die xxx ihre Praktikumsnummer sein soll.

Bei Anweisungen, die sich auf Eingaben beziehen, sind die Anführungszeichen nicht mit einzugeben.

Übersicht

Aufgabe: ***Arbeiten Sie nachfolgendes Tutorial durch.***

1. WebSphere MQ

WebSphere MQ, auf das auch heute noch oft unter der alten Bezeichnung MQSeries verwiesen wird, zählt zur sogenannten nachrichtenbasierten Middleware (Message Oriented Middleware; MOM). Bei nachrichtenbasierter Middleware werden, wie der Name schon sagt, Nachrichten zwischen den Systemen ausgetauscht.

WebSphere MQ nutzt eine spezielle Art der Message Oriented Middleware, das Message Queuing. Das Prinzip des Message Queuing führt Warteschlangen, die Queues, ein.
In diesen Warteschlangen werden zum Beispiel ausgehende Nachrichten solange zwischengespeichert, bis der Empfänger sie anfordert und in seine Warteschlange einreiht.

Diese Technik ermöglicht eine asynchrone Kommunikation zwischen den Teilnehmern. Asynchrone Kommunikation bedeutet in dem Fall, dass, falls der Empfänger zu einem Zeitpunkt nicht zur Verfügung stehen sollte, der Sender so lange wartet, also die Nachricht in seiner Queue lässt, bis der Empfänger empfangsbereit ist. Wobei „warten“ in diesem Fall nicht bedeutet, dass der Sender währenddessen blockiert ist. Es können durchaus noch andere Aufgaben ausgeführt werden.

An dieser Stelle sollte jedoch darauf hingewiesen werden, dass mit „Sender“ und „Empfänger“ hier nicht die jeweiligen Anwendungsprogramme, sondern vielmehr die MQ-Elemente gemeint sind. So kann es zum Beispiel auch der Fall sein, dass die sendende Anwendung nicht mehr aktiv ist, aber trotzdem noch zu sendende Nachrichten in der Queue liegen. Auch in diesem Fall werden die Nachrichten sobald wie möglich versendet.

Wie hat man sich Nachrichten (Messages) überhaupt vorzustellen? Grundsätzlich besteht eine Nachricht aus zwei Teilen, dem Header (Message Descriptor) und dem eigentlichen Datenteil. Der Message Descriptor (bei WebSphere MQ als Message queuing message descriptor (MQMD) bezeichnet), enthält Informationen zu der Nachricht bezüglich des Nachrichtentyps, der Lebensdauer der Nachricht, der zuständigen Antwortwarteschlange (Reply Queue), u.a.
Im Datenteil befinden sich die entsprechenden Anwendungsdaten, welche auch Programmbefehle enthalten können. Die voreingestellte Maximalgröße einer Nachricht beträgt 4 MB, bei WebSphere MQ v6 ist eine Nachrichtengröße zwischen 0 und 100 MB möglich.

Diese Aufgabe wird von einem Queue Manager übernommen. Dem Queue Manager obliegt ebenfalls die Aufgabe, die Queues, Channels und die übrigen MQ-Objekte zu verwalten. Er ist also zwingend notwendig, um mit WebSphere MQ arbeiten zu können.

Innerhalb des Queue Managers übernehmen die Message Channel Agents (MCA) die Aufgabe, Nachrichten aus den Queues zu lesen und über die Kanäle an den Gegenpart zu senden, oder umgekehrt, über die Kanäle empfangene Nachrichten in den entsprechenden Queues abzulegen.

Die Channels (Kanäle) dienen als Verbindung zwischen den Queue Managern. Jedem Kanal ist ein eigener Message Channel Agent zugewiesen.
Ein Kanal ist hierbei eine unidirektionale Verbindung, d.h. also über denselben Kanal kann nicht gesendet und empfangen werden.

Der allgemeine Zweck, den Middleware erfüllen soll, nämlich eine plattformenabhängige Anwendungsentwicklung zu ermöglichen, erfüllt WebSphere MQ durch eine einheitliche Programmierschnittstelle (Application Programming Interface, API). Bei WebSphere MQ wird diese Schnittstelle als Message Queue Interface (MQI) bezeichnet.

Wir unterstellen, dass sie mit den Einzelheiten vertraut sind, siehe Band 1, Abschnitt 10.1.11 ff.

2. Tutorial Übersicht

In diesem Tutorial benutzen wir die MQI Interface. (Die JMS Alternative werden wir in Tutorial WebSphere und Message Diven Beans behandeln). Wir werden (indirekt) die folgenden sechs MQ-Funktionen (auch Calls genannt) benutzen. Einzelheiten in Band 1, Abschnitt 10.4.1 ff. .

MQCONN	**Stellt eine Verbindung zum Queue Manager her (Connect)**
MQDISC	**Löst die Verbindung zum Queue Manager (Disconnect**
MQOPEN	**Öffnet eine Verbindung zu einer bestimmten Queue**
MQCLOSE	**Schließt die Verbindung zu einer bestimmten Queue**
MQPUT	**Legt eine Nachricht in einer Queue ab**
MQGET	**Liest und löscht eine Nachricht aus einer Queue**

WebSphere MQ stellt mit der Installation einige Beispiele bereit, mit denen wir hier arbeiten wollen. Es handelt sich hierbei im Speziellen um die COBOL II-Beispiele zu MQPUT und MQGET.

Wir werden hierzu eine lokale Queue auf dem Queue Manager des z/Os Rechnerst-Maschine anlegen und in diese mit den zur Verfügung gestellten Programmen Nachrichten ablegen und wieder auslesen. COBOL-Kenntnisse sind hierfür nicht erforderlich, Änderungen, also die Übergabe von Parametern für die MQ-Calls, werden in einem, ebenfalls standardmäßig zur Verfügung stehenden, JCL-Skript gemacht.

Wir verwenden hier den Queue Manager CSQ6 auf hobbit (134.2.205.54).
Sollte der Queue Manager nicht existieren, sollte umgehend der Betreuer des Praktikums informiert werden.

Ein Queue Manager kann nur von entsprechend autorisiertem Administrator gestartet werden.

Zur Bearbeitung dieses Tutorials ist lediglich ein TN3270-Emulator erforderlich (z.B. x3270 für Linux-Systeme oder wc3270 für Windows-Systeme).

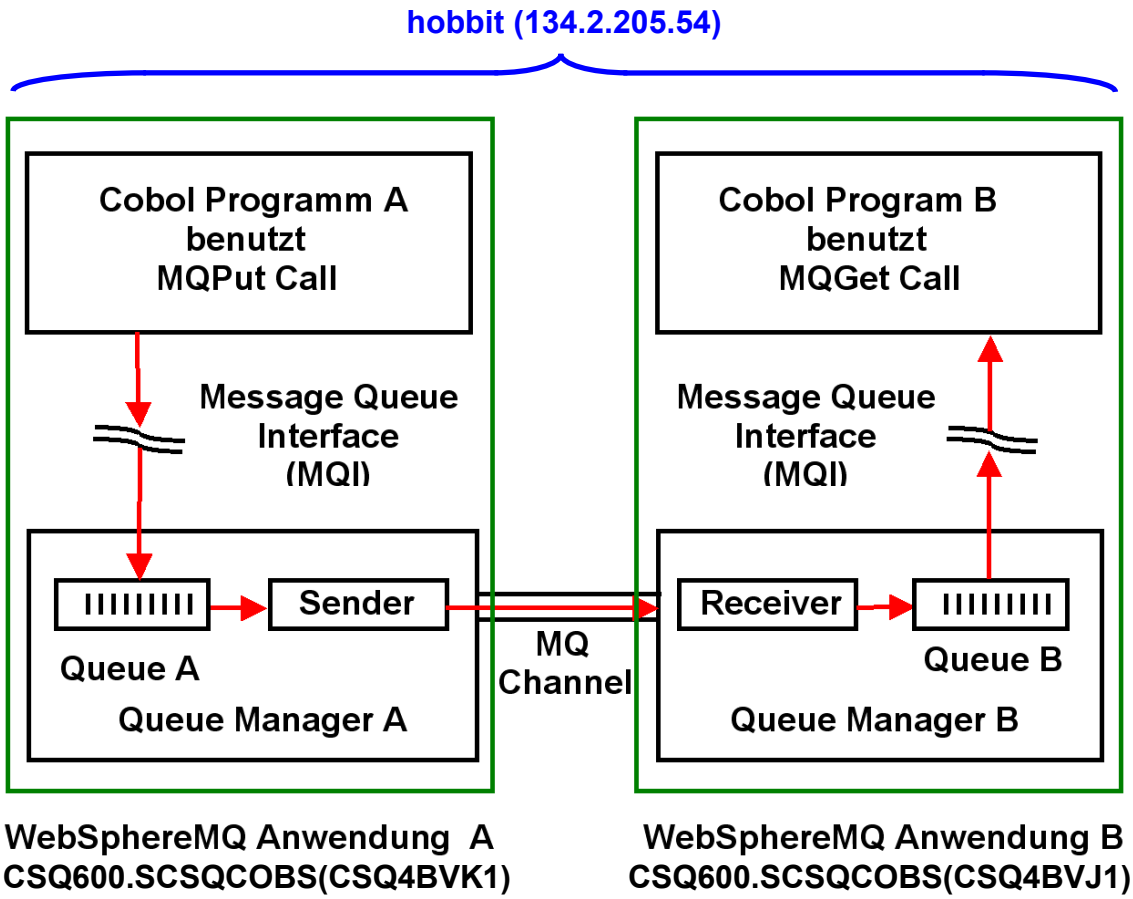


Abb. 6.1

Das von uns erstellte Tutorial erstellt zwei z/OS Anwendungen (Anwendung A und Anwendung B) mit jeweils einem Queue Manager (Queue Manager A) und (Queue Manager B) auf dem gleichen z/OS System.
Für Anwendung A erstellen wir eine Queue A (Transmission Queue, hier als Local Queue bezeichnet). Für Anwendung B erstellen wir eine Queue B (Target Queue).

Sender (MCA), Receiver (MCA und Channel werden durch eine lokale z/OS Struktur implementiert.

Wir speichern in der Transmission Queue eine einfache Nachricht bestehend aus je fünfmal dem Zeichen „Z“ (also dem String „ZZZZZ“).

Wir benutzen ein vorgefertigtes Cobol Programm A, um den Inhalt der Transmission Queue asynchron an die Target Queue zu senden. Wir benutzen ein weiteres vorgefertigtes Cobol Programm B, um den Inhalt der Target Queue auszulesen.

Programm A liegt vorgefertig als Member eines Partitioned Data Sets vor, spezifisch als CSQ600.SCSQCOBS(CSQ4BVK1). Wir können es von dort kopieren. Ebeso liegt Programm B vorgefertigt als CSQ600.SCSQCOBS(CSQ4BVJ1) vor.

Wir legen einen eigenen Partitioned Data Set PRAK222.CSQ6.USERJCL (PRAK 222 durch Ihre eigene User ID ersetzen) an. Wir kopieren Programm A in den Member MQPUT und Programm B- in den Member MQGET.

Wir erstellen ein JCL Script, welches das Programm PRAK222.CSQ6.USERJCL(MQPUT) mittels submit ausführt. Wir benutzen SDSF, um zu verifizieren, dass dies korrekt geschehen ist. Mit einem weiteren JCL Script führen wir das Programm PRAK222.CSQ6.USERJCL(MQGET) mittels submit aus. Wiederum mittels SDSF verifizieren wir, dass unsere Nachricht „ZZZZZ" korrekt eingetroffen ist.

Noch eine Anmerkung zum Schluss: zum Teil zeigen die Abbildungen in diesem Tutorial nur einen Ausschnitt des Bildschirms, davon sollten wir uns jedoch nicht verwirren lassen.

3. WebSphere MQ for z/OS

Zunächst müssen wir eine lokale Warteschlange (local queue) auf dem Server erstellen. Hierfür loggen wir uns auf dem Server (wir benutzen Time-Sharing Option, also „L TSO" oder einfach „LOGON PRAKxxx") ein und rufen das ISPF auf. Hierzu geben wir einfach nach erfolgreichem Login „ispf" ein und bestätigen mit der Eingabetaste.

Es erscheint nun das ISPF Primary Option Menu. Hier werden jedoch nur Basisfunktionen angeboten, wir benötigen für dieses Tutorial jedoch zusätzliche Optionen. Wir geben also in der Eingabezeile „m" für More (im Menü steht hierzu „Additional IBM Products") ein und bestätigen mit der Eingabetaste.

Wir sehen nun das IBM Products Panel (siehe Abbildung 2). Hier finden wir unter Punkt 15 „WMQ Series Operations and Control", genau das also, was wir jetzt benötigen.

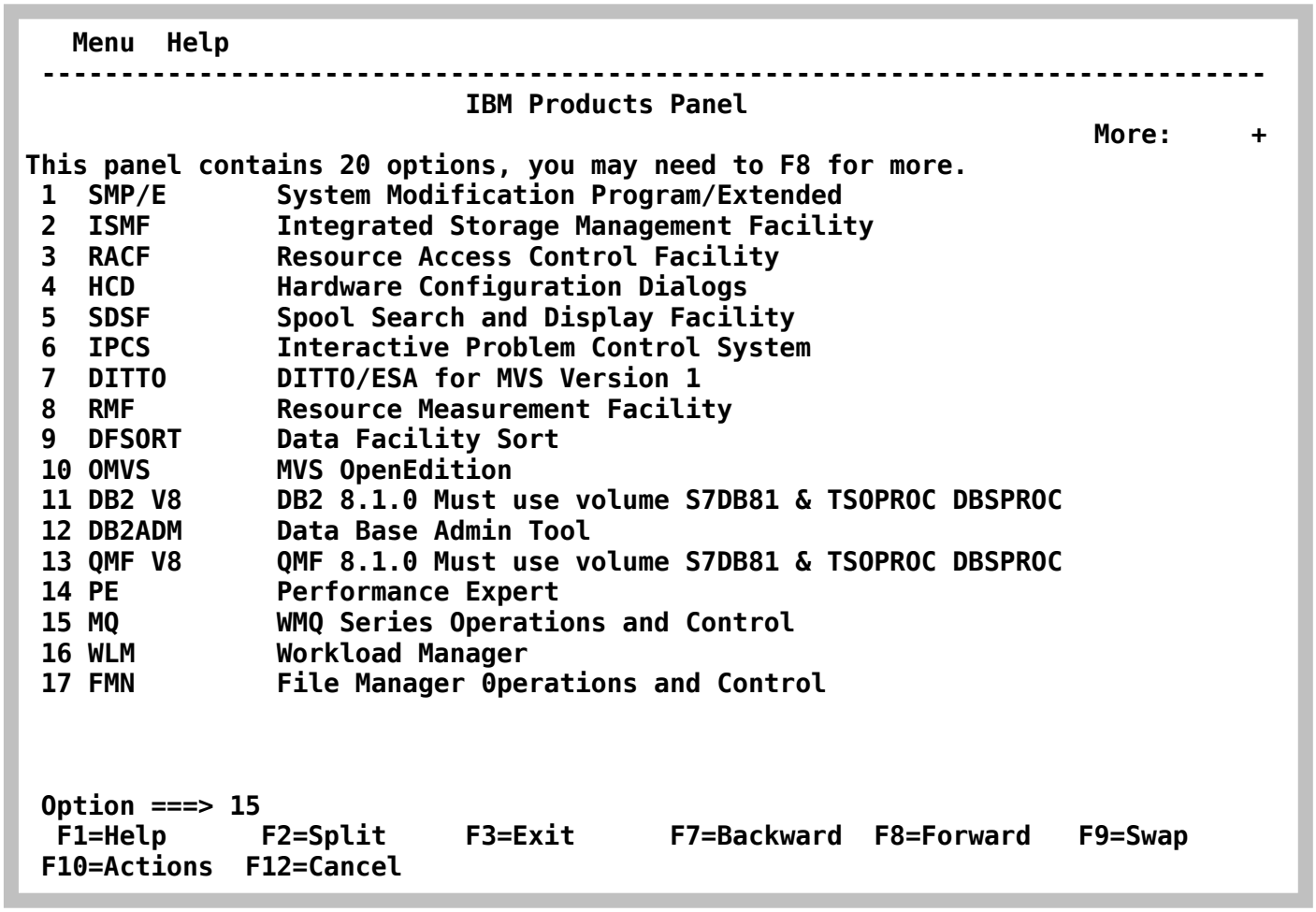


Abbildung 2: Bildschirm des IBM Products Panel

Wir geben also in der Optionszeile „15“ ein und bestätigen wieder mit der Eingabetaste.

Nun befinden wir uns im IBM WebSphere MQ for z/OS – Main Menu.

Zunächst sollte überprüft werden, ob bereits eine lokale Queue mit unserer Praktikums-ID existiert. Hierfür geben wir die Daten wie in Abbildung 3 ein. Da wir zunächst nach einer lokalen Warteschlange suchen wollen, geben wir als „Action“ 1 ein. Als „Object“ wollen wir lokale Queues anzeigen lassen, geben also QLOCAL ein. Andere mögliche Parameter wären zum Beispiel QREMOTE, SENDER oder CHANNEL. In einem anderen Tutorial werden wir einigen noch einmal begegnen. Als „Connect Name“ geben wir unseren Queue Manager CSQ6 ein und bestätigen mit der Eingabetaste

```
                    IBM WebSphere MQ for z/OS - Main Menu

Complete fields. Then press Enter.

Action  . . . . . . . . . . . 1     0. List with filter   4. Manage
                                    1. List or Display    5. Perform
                                    2. Define like        6. Start
                                    3. Alter              7. Stop
Object type . . . . . . . . . QLOCAL        +
Name  . . . . . . . . . . . . PRAK222
Disposition . . . . . . . . .    Q=Qmgr, C=Copy, P=Private, G=Group,
                                 S=Shared, A=All

Connect name  . . . . . . . . CSQ6  - local queue manager or group
Target queue manager  . . .
           - connected or remote queue manager for command input
Action queue manager  . . .        - command scope in group
Response wait time  . . . . 30    5 - 999 seconds

(C) Copyright IBM Corporation 1993,2005. All rights reserved.




Command ===>
 F1=Help      F2=Split     F3=Exit      F4=Prompt    F9=SwapNext F10=Messages
F12=Cancel
```

Abbildung 3: IBM WebSphere MQ for z/OS Hauptmenü

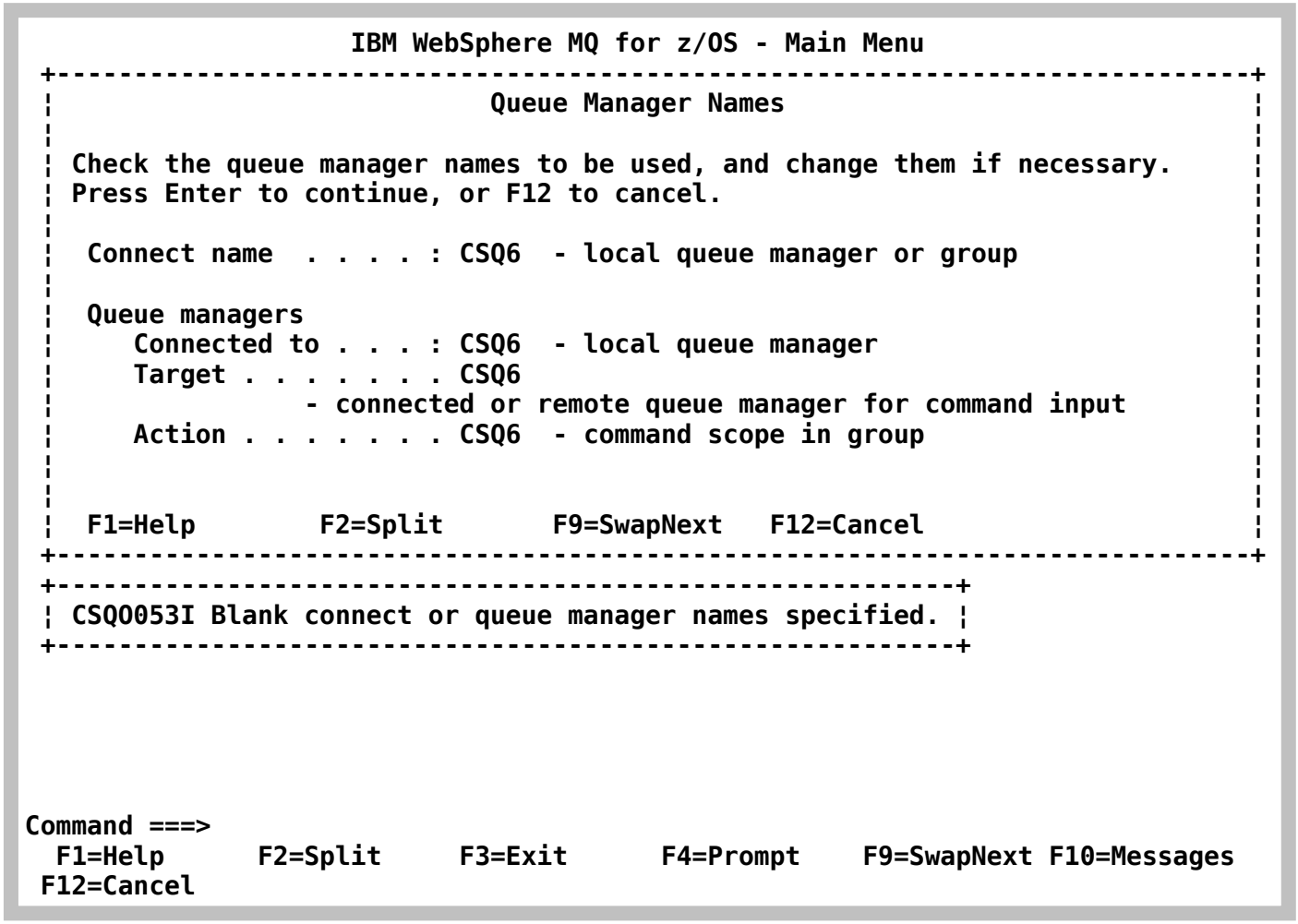


Abbildung 4: Bildschirm: Queue Manager Names

Da wir auf dem letzten Bildschirm (Abbildung 3) die Angaben bei „Target queue manager“ und „Action queue manager“ ausgelassen haben, gelangen wir mit der Meldung „`CSQO053I Blank connect or queue manager names specified.`“ auf den Bildschirm „Queue Manager Names“ (Abbildung 4). Hier geben wir überall den Namen unseres Queue Managers ein.

Wenn wir jetzt wieder im WebSphere MQ Hauptmenü landen und die Meldung „`CSQO050I No objects of type QLOCAL disposition ALL match PRAKxxx.`“ ausgegeben wird, bedeutet dies, dass noch keine Queue mit unserem Namen angelegt wurde. In diesem Fall können wir den folgenden Abschnitt „Löschen einer Local Queue“ überspringen und mit „Anlegen einer Local Queue“ fortfahren.

4. Löschen einer Local Queue

Wenn statt der obigen Meldung folgender Bildschirm (Abbildung 5) erscheint, bedeutet dies, dass bereits eine Queue mit unserem Namen existiert, die wir zunächst löschen müssen.

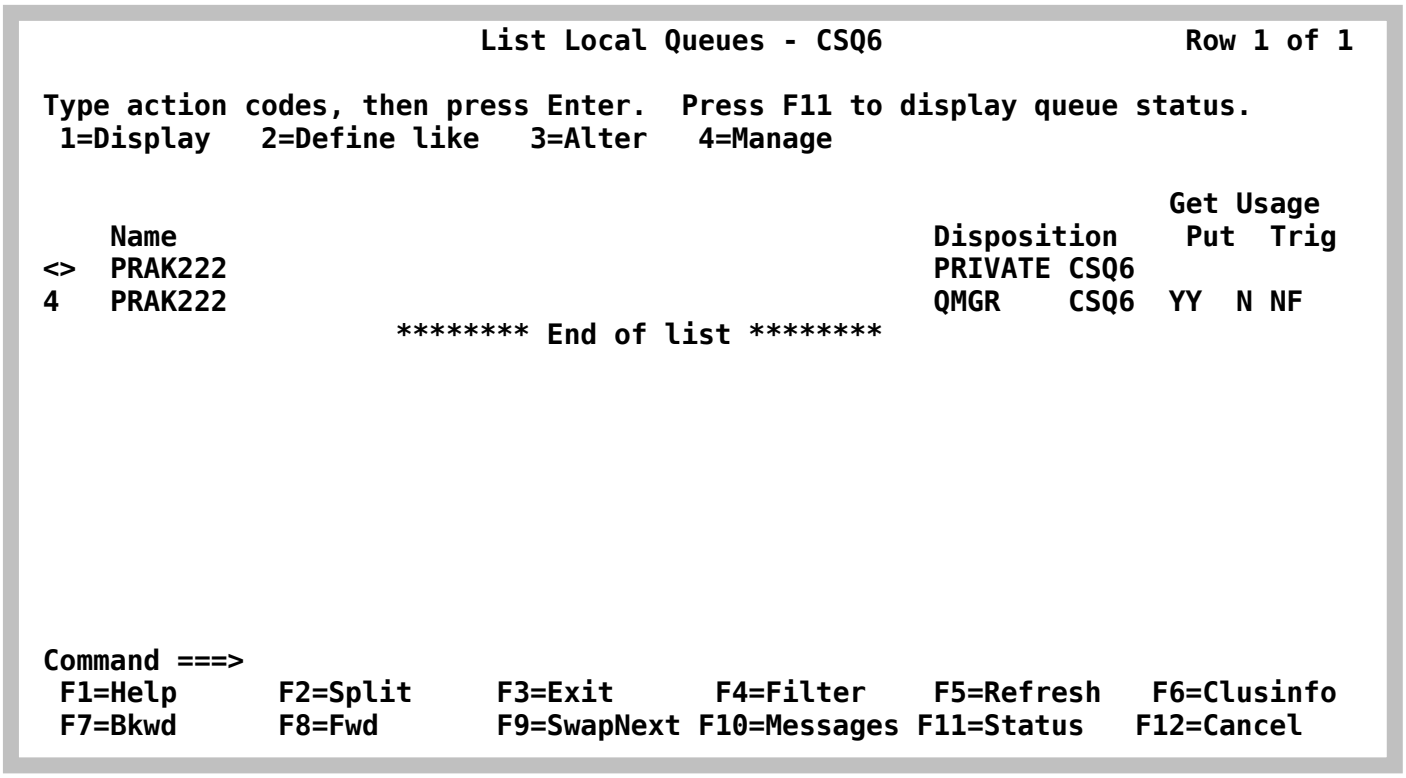


Abbildung 5: Bildschirm: Queue Manager Names

Was wir vor uns sehen, ist eine Liste unserer lokalen Warteschlangen, die wir eigentlich noch gar nicht haben sollten. Wir setzen daher an der Position vor dem Namen PRAKxxx eine 4 und bestätigen mit der Eingabetaste. So gelangen wir auf die Management-Optionen für die lokalen Queues.

Wir haben hier verschiedene Optionen (u.a. Verschieben von Nachrichten in andere Queues und alle Nachrichten löschen) und wählen die 1 für „Delete“.

Wir werden nun nach einer Bestätigung für den Löschvorgang gefragt und bestätigen mit der Eingabetaste. Es erscheint jetzt im „Manage a Local Queue“-Bildschirm die Meldung „`CSQ9022I %CSQ6 CSQMUQLC ' DELETE QLOCAL' NORMAL COMPLETION.`“. Unsere Queue ist gelöscht und wir kehren mit F3 zum Hauptmenü zurück.

5. Anlegen einer Local Queue

Im „IBM WebSphere MQ for z/OS - Main Menu“ erstellen wir jetzt eine neue lokale Warteschlange mit unserer Praktikums-ID als Name.

Wir geben hierfür die Daten wie in Abbildung 6 zu sehen ein und bestätigen mit der Eingabetaste.

```
                    IBM WebSphere MQ for z/OS - Main Menu

Complete fields. Then press Enter.

Action  . . . . . . . . . . 2     0. List with filter   4. Manage
                                  1. List or Display    5. Perform
                                  2. Define like        6. Start
                                  3. Alter              7. Stop
Object type . . . . . . . . QLOCAL        +
Name  . . . . . . . . . . .
Disposition . . . . . . . .    Q=Qmgr, C=Copy, P=Private, G=Group,
                               S=Shared, A=All

Connect name  . . . . . . . CSQ6  - local queue manager or group
Target queue manager  . . . CSQ6
          - connected or remote queue manager for command input
Action queue manager  . . . CSQ6  - command scope in group
Response wait time  . . . . 30    5 - 999 seconds

(C) Copyright IBM Corporation 1993,2005. All rights reserved.




Command ===>
 F1=Help      F2=Split     F3=Exit      F4=Prompt    F9=SwapNext F10=Messages
F12=Cancel
```

Abbildung 6: Erstellen einer Local Queue

Wir lassen hier bewusst den Namen PRAKxxx noch weg, damit eine Queue mit den Standard-Werten erstellt wird. Mit der Eingabetaste gelangen wird zu den Screens mit den Parametern für die Queue.

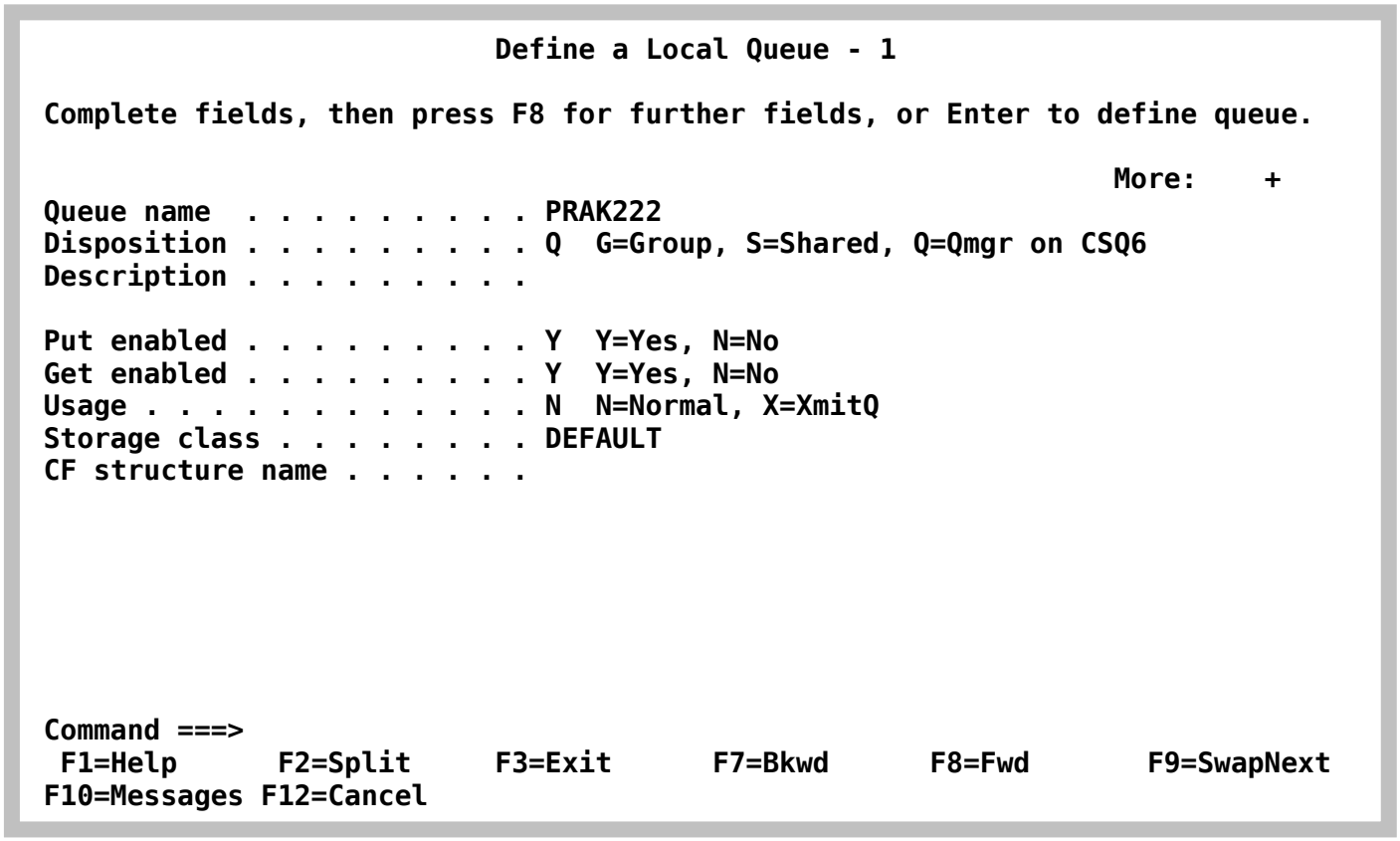


Abbildung 7: Define a Local Queue - 1

Im ersten Bildschirm, Abbildung 7, geben wir als Queue Name unsere Praktikums-ID ein und vergewissern uns, dass die sonstigen Angaben denen in der Abbildung entsprechen. So sollten zum Beispiel PUT und GET jeweils aktiv sein.

```
                          Define a Local Queue - 2

Press F7 or F8 to see other fields, or Enter to define queue.

                                                                  More:  - +
Queue name  . . . . . . . . : PRAK222
Disposition . . . . . . . . : QMGR    CSQ6

Cluster name  . . . . . . . .
Cluster namelist name . . . .
Default bind  . . . . . . . . O  O=Open, N=Notfixed
Cluster workload rank . . . . 0  0 - 9
Cluster workload priority . . 0  0 - 9
Cluster workload queue use    Q  A=Any, L=Local, Q=Qmgr









Command ===>
 F1=Help      F2=Split     F3=Exit      F7=Bkwd      F8=Fwd       F9=SwapNext
F10=Messages F12=Cancel
```

Abbildung 8: Define a Local Queue – 2

Mit F8 wechseln wir zum nächsten Definitionsbildschirm. Bei diesem Bildschirm brauchen wir keine Änderungen vorzunehmen, wir übernehmen die Daten wie in Abbildung 8 und blättern mit F8 weiter.

```
                          Define a Local Queue - 3

Press F7 or F8 to see other fields, or Enter to define queue.

                                                                  More:  - +
Queue name  . . . . . . . . : PRAK222
Disposition . . . . . . . . : QMGR    CSQ6

Default persistence . . . . . N  Y=Yes, N=No
Default priority  . . . . . . 0  0 - 9
Message delivery sequence . . P  P=Priority, F=FIFO
Permit shared access  . . . . N  Y=Yes, N=No
Default share option  . . . . E  E=Exclusive, S=Shared
Index type  . . . . . . . . . N  N=None, M=MsgId, C=CorrelId, G=GroupId,
                                 T=MsgToken

Maximum queue depth . . . . . 999999999  0 - 999999999
Maximum message length  . . . 4194304    0 - 104857600

Command ===>
 F1=Help      F2=Split     F3=Exit      F7=Bkwd      F8=Fwd       F9=SwapNext
F10=Messages F12=Cancel
```

Abbildung 9: Define a Local Queue - 3

Im dritten Definitionsbildschirm (Abbildung 9) müssen wir uns unter anderem entscheiden, ob wir persistente oder nichtpersistente Nachrichten wollen. Eine Nachricht wird dann als „persistent“ bezeichnet, wenn sie nach einem Neustart des Queue Managers wiederhergestellt werden soll. Persistente Nachrichten werden daher geloggt, um zum Beispiel im Falle eines unerwarteten Queue Manager-Ausfalls, keine Nachrichten zu verlieren.

Für unsere Zwecke ist dies aber nicht notwendig, daher deaktivieren wir persistente Nachrichten mit „N“. Des Weiteren haben wir die Möglichkeit, uns zwischen einem prioritisierten und einem FIFO-Abarbeiten der Queue zu entscheiden. Im ersten Fall wird den Nachrichten in der Warteschlange jeweils eine Priorität zugewiesen, die Nachrichten mit der höchsten Priorität werden als Erstes bearbeitet. Bei FIFO (First-In First-Out) wird die Nachricht, welche als Erste in die Schlange eingereiht wurde, auch als Erstes bearbeitet.

Wir entscheiden uns hier für die Priorität („P“).

Wir überprüfen die Daten entsprechend Abbildung 9 und blättern zum vierten Definitionsbildschirm.

```
                        Define a Local Queue - 4

Press F7 or F8 to see other fields, or Enter to define queue.

                                                                More:  - +
Queue name  . . . . . . . . : PRAK222
Disposition . . . . . . . . : QMGR    CSQ6

Trigger Definition

   Trigger type . . . . . . . F  F=First, E=Every, D=Depth, N=None
   Trigger set  . . . . . . . N  Y=Yes, N=No

   Trigger message priority   0  0 - 9
   Trigger depth  . . . . . . 1          1 - 999999999

   Initiation queue . . . . .
   Process name . . . . . . .
   Trigger data . . . . . . .




Command ===>
 F1=Help      F2=Split     F3=Exit      F7=Bkwd      F8=Fwd       F9=SwapNext
F10=Messages F12=Cancel
```

Abbildung 10: Define a Local Queue - 4

Im vierten Bildschirm (Abbildung 10) tragen wir bei „Trigger type“ ein „F“ (für First), bei „Trigger set“ „N“ (für No) ein.

Es folgen jetzt noch zwei weitere Definitionsbildschirme. Hier sind jedoch keine Änderungen erforderlich und so können wir mit der Eingabetaste bestätigen und unsere Queue damit anlegen. Wir können uns natürlich auch noch die restlichen Screens anschauen (Blättern mit F7 und F8) und erst dann die Eingabetaste betätigen.

Wir landen jetzt wieder auf dem ersten Definitionsbildschrim, auf dem die Meldung „CSQ9022I %CSQ6 CSQMAQLC ' DEFINE QLOCAL' NORMAL COMPLETION“ erscheinen sollte (Abbildung 11).

```
                        Define a Local Queue - 1

Complete fields, then press F8 for further fields, or Enter to define queue.

                                                                More:    +
Queue name  . . . . . . . . . PRAK222
Disposition . . . . . . . . . Q  G=Group, S=Shared, Q=Qmgr on CSQ6
Description . . . . . . . . .

Put enabled . . . . . . . . . Y  Y=Yes, N=No
Get enabled . . . . . . . . . Y  Y=Yes, N=No
Usage . . . . . . . . . . . . N  N=Normal, X=XmitQ
Storage class . . . . . . . . DEFAULT
CF structure name . . . . . .

      +-------------------------------------------------------------+
      ¦ CSQ9022I %CSQ6 CSQMAQLC ' DEFINE QLOCAL' NORMAL COMPLETION ¦
      +-------------------------------------------------------------+
Command ===>
 F1=Help      F2=Split     F3=Exit      F7=Bkwd      F8=Fwd       F9=SwapNext
F10=Messages F12=Cancel
```

Abbildung 11: Queue erfolgreich angelegt

Jetzt überprüfen wir noch, ob unsere Queue auch in der Liste erscheint. Hierzu gehen wir mit F3 zurück zum IBM WebSphere MQ for z/OS – Main Menu. Hier geben wir als Action wieder „1“ für „List or Display“ an und bei Name „PRAK*“ ein (Abbildung 12).

```
                    IBM WebSphere MQ for z/OS - Main Menu

Complete fields. Then press Enter.

Action  . . . . . . . . . . . 1     0. List with filter   4. Manage
                                    1. List or Display    5. Perform
                                    2. Define like        6. Start
                                    3. Alter              7. Stop
Object type . . . . . . . . . QLOCAL        +
Name  . . . . . . . . . . . . PRAK*
Disposition . . . . . . . . .    Q=Qmgr, C=Copy, P=Private, G=Group,
                                 S=Shared, A=All

Connect name  . . . . . . . . CSQ6  - local queue manager or group
Target queue manager  . . . CSQ6
           - connected or remote queue manager for command input
Action queue manager  . . . CSQ6  - command scope in group
Response wait time  . . . . 30    5 - 999 seconds

(C) Copyright IBM Corporation 1993,2005. All rights reserved.




Command ===>
 F1=Help      F2=Split     F3=Exit      F4=Prompt    F9=SwapNext F10=Messages
F12=Cancel
```

Abbildung 12: Queues ausgeben

Wir bestätigen die Eingaben mit der Eingabetaste und gelangen zum Bildschirm mit der Liste der Queues auf unserem Queue Manager CSQ6, die unseren Anforderungen (beginnen mit PRAK) entsprechen (Abbildung 13).

*** ist eine Wildcard und sorgt dafür, dass uns alle Queues, die mit PRAK beginnen, ausgegeben werden.**

Falls es notwendig sein sollte, können wir mit F7 und F8 durch die Liste blättern.

Aufgabe: *Legen Sie eine lokale Queue mit dem Namen PRAKxxx (Entsprechend Ihrer Praktikums-ID) an und überprüfen Sie, ob die Queue angelegt wurde.*

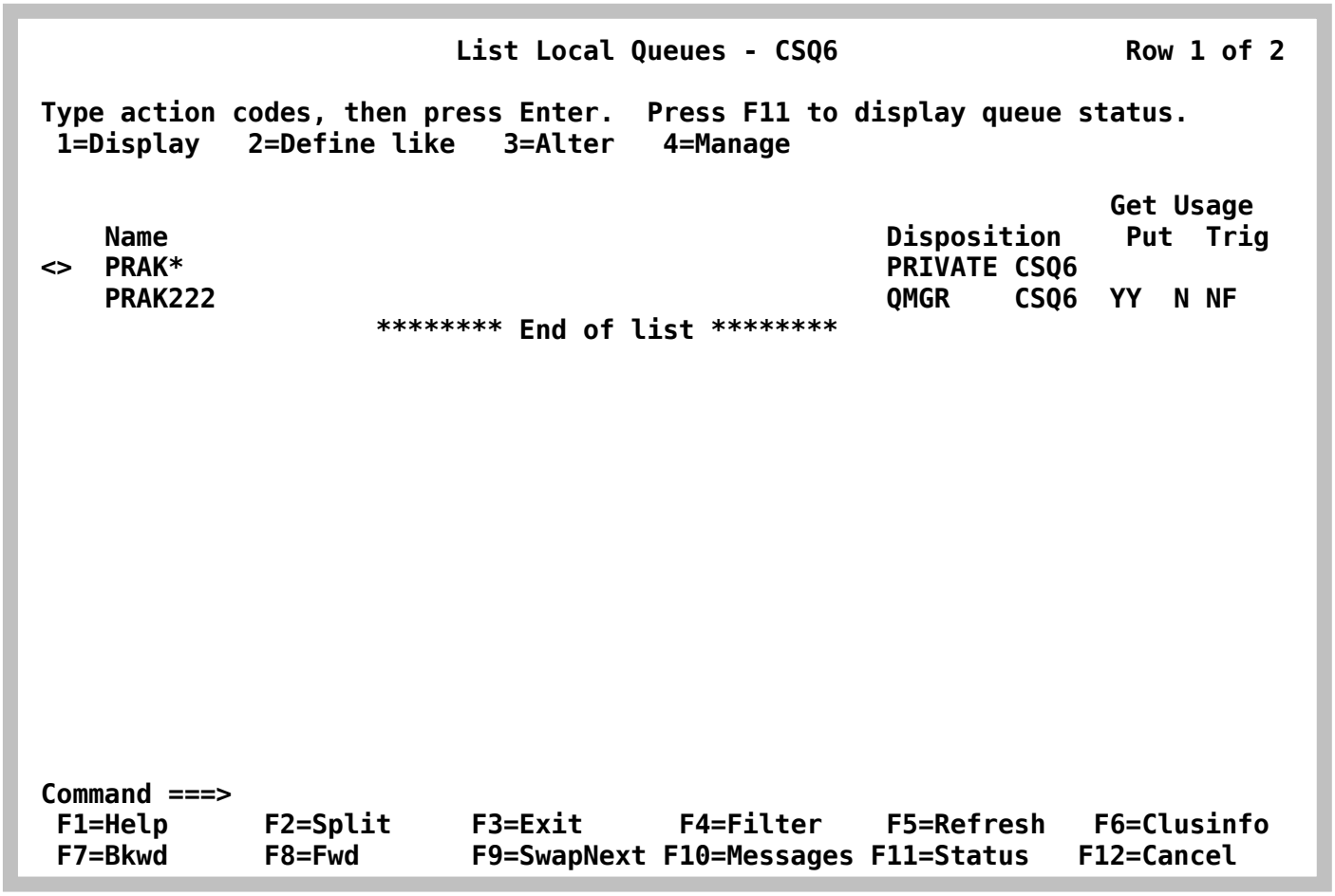


Abbildung 13: Queues ausgeben

Da wir jetzt erfolgreich eine Queue erstellt haben, können wir uns den Anwendungsprogrammen zuwenden. Hierzu wird der Quellcode in COBOL II generiert. Da nicht davon ausgegangen werden kann, dass jeder der dieses Tutorial bearbeitet, COBOL-Kenntnisse besitzt, greifen wir hierfür auf Quellcodes der IBM WebSphere MQ-Bibliotheken zurück. Diese Bibliotheken werden bereits vorkompiliert auf dem Server zur Verfügung gestellt.

Die Bibliothek für die COBOL-Quellcodes hat die Bezeichnung CSQ600.SCSQCOBS.

Um fortzufahren, gehen wir nun ins ISPF Primary Option Menu (also nicht in das IBM Product Panel, von dem aus wir WebSphere MQ aufgerufen haben).

Hier geben wir „3;4“ um direkt ins Data Set List Utility (Abbildung 14) zu gelangen.

6. COBOL Quelltext-Module für WebSphere MQ

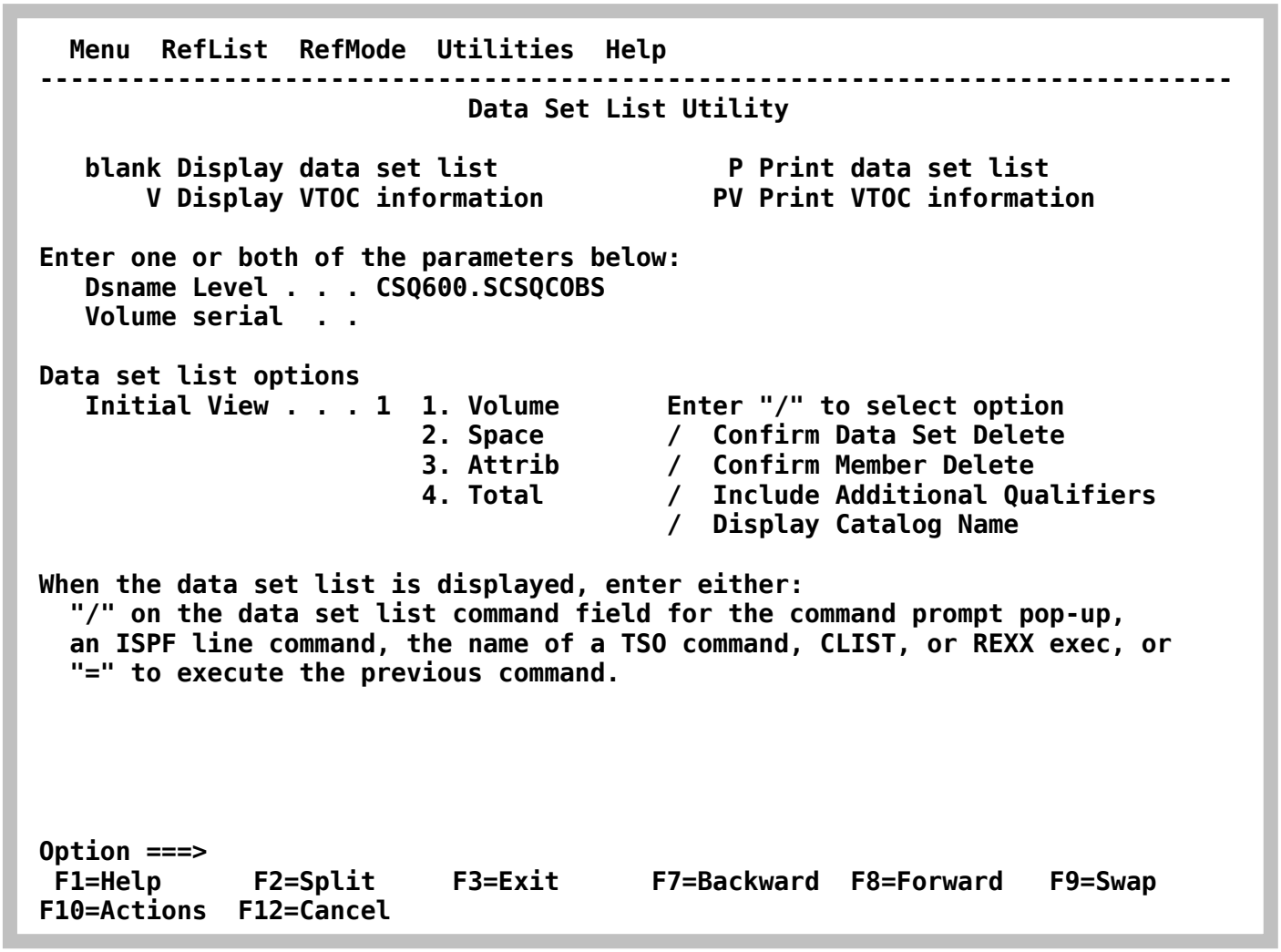


Abbildung 14: Data Set List Utility

Im Data Set List Utility geben wir bei „Dsname Level“ die Data Set-Bezeichnung CSQ600.SCSQCOBS ein. Diese Bibliothek enthält COBOL II-Quelltext-Module für WebSphere MQ, unter anderem die von uns benötigten PUT und GET.

Die „Schwester“-Bibliothek dazu ist SCSQCOBC, welche die COBOL Copy Books enthält. Man erkennt dies leicht an den Endungen der beiden Bibliotheken (S für Source, C für Copy).

Bei „Initial View“ begnügen wir uns mit „1“ für Volume.

Wir bestätigen mit der Eingabetaste und gelangen zum Bildschirm „DSLIST - Data Sets Matching CSQ600.SCSQCOBS“, auf welchem uns die Bezeichnung des Datenträgers angezeigt wird, auf dem das Data Set liegt (Abbildung 15). Diese Angabe ist für uns aber jetzt nicht von Interesse.

In der Command-Spalte (die erste Spalte, vor der Bezeichnung des DataSet) setzen wir ein „b“ (für browse) und bestätigen mit der Eingabetaste. Als Ergebnis bekommen wir eine Ausgabe aller Member des Data Set (Abbildung 16).

```
  Menu  Options  View  Utilities  Compilers  Help
------------------------------------------------------------------------------
DSLIST - Data Sets Matching CSQ600.SCSQCOBS                     Row 1 of 1

Command - Enter "/" to select action                Message          Volume
------------------------------------------------------------------------------
b        CSQ600.SCSQCOBS                                          Z8RES2
****************************** End of Data Set list ****************************
                                                                  Scroll ===> PAGE
Command ===>
 F1=Help    F2=Split   F3=Exit    F5=Rfind   F7=Up      F8=Down    F9=Swap
F10=Left   F11=Right  F12=Cancel
```

Abbildung 15: DSLIST Data Set Matching

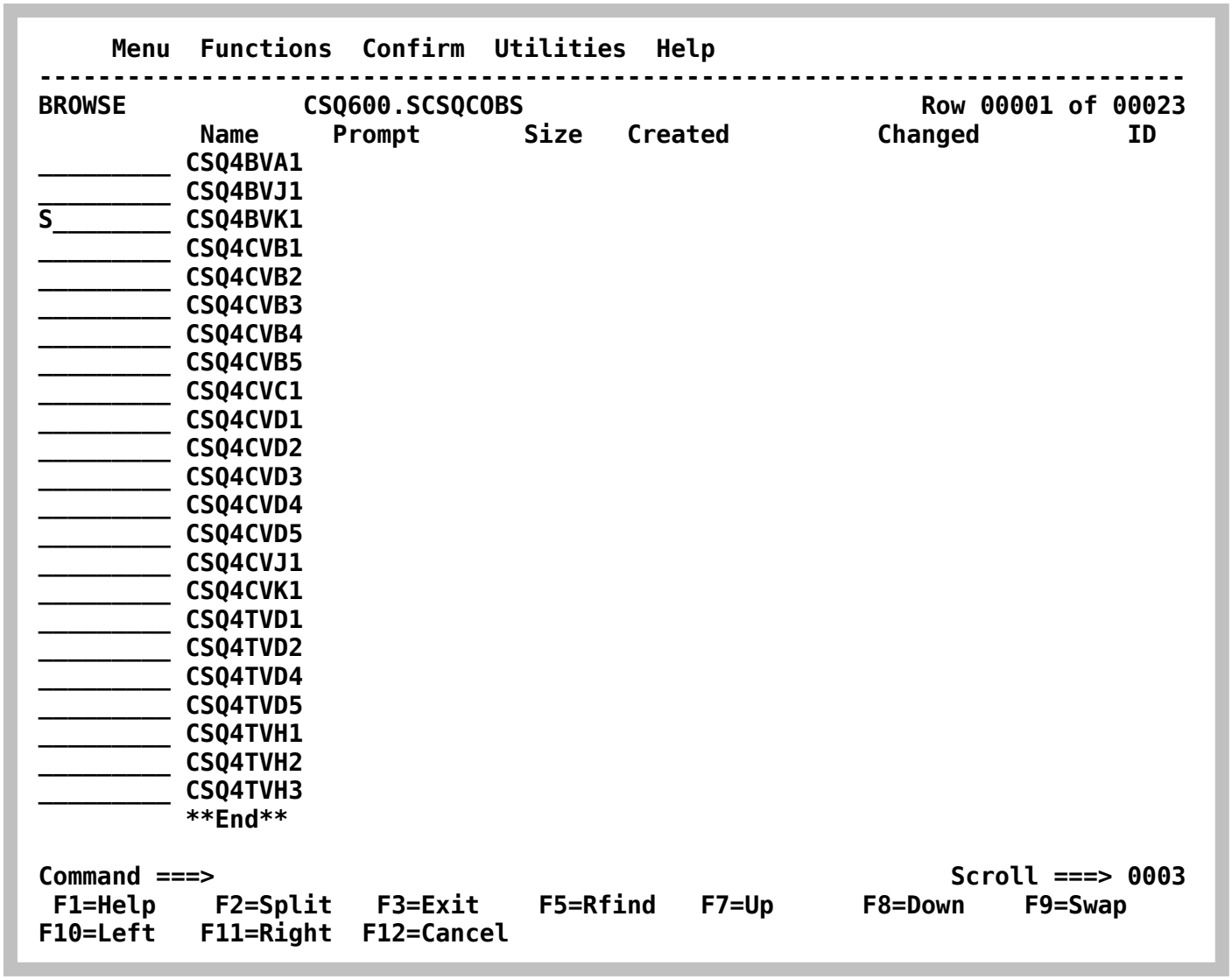


Abbildung 16: DSLIST Data Set Matching

Wenn wir in der Browse-Spalte ein „S“ (für select) eingeben, können wir uns den Quellcode der Module ansehen.

Wir schauen uns die Quellcodes der beiden Module CSQ4BVK1 (put-Modul) und CSQ4BVJ1 (get-Modul) an. Mit den Tasten F7 und F8 können wir durch den Code blättern, mit F3 kommen wir zurück zum Bildschirm aus Abbildung 15.

Im Anhang ist der Cobol Quellcode für den Member CSQ600.SCSQCOBS(CSQ4BVK1) wiedergegeben

Was uns jetzt noch fehlt, ist ein eigenes Data Set für unsere zwei zu kopierende JCL-Scripte, welche die Nachrichten zur Message Queue schicken (MQPUT) und wieder empfangen (MQGET).

Wir legen also, wie in Tutorial 1, ein Data Set mit dem Namen PRAKxxx.CSQ6.USERJCL mit den Werten aus Abbildung 17 an.

7. Anlegen und Kopieren von Datasets

```
  Menu  RefList  Utilities  Help
------------------------------------------------------------------------------
                            Allocate New Data Set

Data Set Name  . . . : PRAK222.CSQ6.USERJCL

Management class . . . DEFAULT        (Blank for default management class)
Storage class  . . . . PRIM90         (Blank for default storage class)
 Volume serial . . . . Z8IMS1         (Blank for system default volume) **
 Device type . . . . .                (Generic unit or device address) **
Data class . . . . . .                (Blank for default data class)
 Space units . . . . . TRACK          (BLKS, TRKS, CYLS, KB, MB, BYTES
                                       or RECORDS)
 Average record unit                  (M, K, or U)
 Primary quantity  . . 3              (In above units)
 Secondary quantity    1              (In above units)
 Directory blocks  . . 5              (Zero for sequential data set) *
 Record format . . . . FB
 Record length . . . . 80
 Block size  . . . . . 6160
 Data set name type    PDS            (LIBRARY, HFS, PDS, LARGE, BASIC, *
                                       EXTREQ, EXTPREF or blank)
 Expiration date . . .                (YY/MM/DD, YYYY/MM/DD
Enter "/" to select option             YY.DDD, YYYY.DDD in Julian form
   Allocate Multiple Volumes           DDDD for retention period in days
                                       or blank)

( * Specifying LIBRARY may override zero directory block)

( ** Only one of these fields may be specified)


Command ===>
 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
F10=Actions  F12=Cancel
```

Abbildung 17: Neuen Data Set anlegen

In diesem Data Set werden gleich unsere JCL-Scripte für PUT und GET abgelegt. Hierzu kopieren wir aus dem CSQ600.SCSQPROC den Member CSQ4BVJR zweimal, einmal mit der Bezeichnung MQPUT und einmal mit MQGET.

Für die beiden Skripte benötigen wir die Parameter für den Queue Manager QMGR, die lokale Warteschlange QUEUE, die Anzahl der Nachrichten MSGS, das Füllzeichen PAD, die Länge LEN und das Flag persistente/nicht-persistente Nachrichten PERS.

MQPUT wird das Modul CSQ4BVK1 aufrufen und eine Nachricht in die lokale Queue geben, MQGET wird CSQ4BVJ1 aufrufen und das Ergebnis aus der Queue auslesen.

Mit dem Move/Copy Utility können wir den Member bequem in unser Data Set kopieren. Um zum Move/Copy Utility zu gelangen, geben wir im ISPF Primary Option Menu „3;3“ ein und bestätigen mit der Eingabetaste.

Im Move/Copy Utility geben wir, wie in Abbildung 18, bei „From Other Partitioned or Sequential Data Set“ den Data Set Name (Achtung, unbedingt in Hochkomma setzen) 'CSQ600.SCSQPROC(CSQ4BVJR)' ein, als Option wählen wir C (für copy).

```
     Menu  RefList  Utilities  Help
--------------------------------------------------------------------------------
                              Move/Copy Utility

 C  Copy data set or member(s)          CP Copy and print
 M  Move data set or member(s)          MP Move and print

Specify "From" Data Set below, then press Enter key

From ISPF Library:
   Project . . .               (--- Options C and CP only         ---)
   Group . . . .          . . .          . . .          . . .
   Type  . . . .
   Member  . . .                  (Blank or pattern for member list,
                                   "*" for all members)

From Other Partitioned or Sequential Data Set:
   Data Set Name . . . 'CSQ600.SCSQPROC(CSQ4BVJR)'
   Volume Serial . . .            (If not cataloged)

Data Set Password  . .            (If password protected)



Option ===> C
 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
F10=Actions  F12=Cancel
```

Abbildung 18: Member kopieren

Als nächstes müssen wir noch die Destination angeben, also wohin kopiert werden soll. Wir geben hierfür im nächsten Bildschirm (Abbildung 19) unter „To ISPF Library“ unser zuvor erstelltes Data Set an.
Zunächst kopieren wir das Modul in den Member MQPUT indem wir alle Daten wie in der Abbildung eintragen und mit der Eingabetaste bestätigen.

Als Erfolgsmeldung sollte danach in der oberen rechten Ecke „Member CSQ4BVJR copied“ erscheinen.

```
  Menu  RefList  Utilities  Help
--------------------------------------------------------------------------------
COPY     From CSQ600.SCSQPROC(CSQ4BVJR)

Specify "To" Data Set Below

To ISPF Library:                 Options:
   Project  . . PRAK222             Enter "/" to select option
   Group  . . . CSQ6                   Replace like-named members
   Type . . . . USERJCL             /  Process member aliases
   Member . . . MQPUT     (Blank unless member is to be renamed)

To Other Partitioned or Sequential Data Set:
   Data Set Name . . .
   Volume Serial . . .           (If not cataloged)

Data Set Password  . .           (If password protected)

To Data Set Options:
   Sequential Disposition        Pack Option         SCLM Setting
   1  1. Mod                     3  1. Yes           3  1. SCLM
      2. Old                        2. No               2. Non-SCLM
                                    3. Default          3. As is









Command ===>
 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
F10=Actions  F12=Cancel
```

Abbildung 19: Member kopieren (2)

Jetzt kopieren wir den Member noch einmal, diesmal jedoch in MQGET.

<u>Aufgabe:</u> Erstellen Sie den neuen Dataset und kopieren Sie das JCL Skript zweimal (MQGET und MQPUT).

8. JCL Scripte für MQPUT und MQGET

Als nächstes werden die beiden JCL-Scripte MQPUT und MQGET entsprechend unserer Anforderungen editiert.. Wir beginnen mit MQPUT und öffnen es mit dem TSO-Editor (ISPF Primary Option Menu → „2“).

Innerhalb des Editors lässt sich wieder mit F7 und F8 blättern. Zeilen, die mit „//*“ beginnen, sind auskommentiert.

Als erste Handlung ändern wir die Job-Card (die ersten Zeilen) des JCL-Scripts .
Anstatt CSQ4BVJR geben wir unsere Praktikums-ID gefolgt von einem „P“ für Put ein. Die restlichen Angaben entnehmen wir der Abbildung 20.

```
000001 //PRAK222P JOB (),CLASS=A,MSGCLASS=M,MSGLEVEL=(1,1),NOTIFY=&SYSUID,
000002 //         REGION=4M
000003 //*********************************************************************
```

Abbildung 20: Jobcard

Werfen wir zunächst einen Blick auf den entscheidenden Teil dieses Skripts (Abbildung 21).

```
000059 //JOBLIB   DD   DSN=++THLQUAL++.SCSQLOAD,DISP=SHR
000060 //         DD   DSN=++THLQUAL++.SCSQANL++LANGLETTER++,DISP=SHR
000061 //         DD   DSN=++THLQUAL++.SCSQAUTH,DISP=SHR
000062 //*
000063 //PUTMSGS  EXEC PGM=CSQ4BVK1,REGION=1024K,
000064 //  PARM=('++QMGR++,++QUEUE++,++MSGS++,++PAD++,++LEN++,++PERS++')
000065 //*
000066 //*GETMSGS  EXEC PGM=CSQ4BVJ1,REGION=1024K,
000067 //*  PARM=('++QMGR++,++QUEUE++,++MSGS++,++GET++,++SYNC++')
000068 //*
000069 //SYSDBOUT DD   SYSOUT=*
000070 //SYSABOUT DD   SYSOUT=*
000071 //SYSPRINT DD   SYSOUT=*
000072 //SYSOUT   DD   SYSOUT=*
```

Abbildung 21: Unveränderter Inhalt der Skripte

Wir passen diese Skripte nun entsprechend unserer Anforderungen an (Abbildung 22):

```
000059 //JOBLIB   DD   DSN=CSQ600.SCSQLOAD,DISP=SHR
000060 //         DD   DSN=CSQ600.SCSQANLE,DISP=SHR
000061 //         DD   DSN=CSQ600.SCSQAUTH,DISP=SHR
000062 //*
000063 //PUTMSGS  EXEC PGM=CSQ4BVK1,REGION=1024K,
000064 //  PARM=(CSQ6,PRAK222,3,Z,5,N)
000065 //*
000066 //*GETMSGS  EXEC PGM=CSQ4BVJ1,REGION=1024K,
000067 //*  PARM=('++QMGR++,++QUEUE++,++MSGS++,++GET++,++SYNC++')
000068 //*
000069 //SYSDBOUT DD   SYSOUT=*
000070 //SYSABOUT DD   SYSOUT=*
000071 //SYSPRINT DD   SYSOUT=*
000072 //SYSOUT   DD   SYSOUT=*
```

Abbildung 22: Unser Skript MQPUT

PUTMSGS ruft hier das (COBOL-)Programm CSQ4BVK1 mit den angegebenen Parametern auf.

Neben dem Queue Manager (CSQ6) und unserer Queue geben wir an, dass 3 Nachrichten, bestehend aus je fünfmal dem Zeichen „Z“ (also dem String „ZZZZZ“) gesendet werden sollen. Mit „N“ geben wir an, dass die Nachrichten nicht-persistent sind.

Nachdem wir die Zeilen entsprechend geändert haben, übermitteln wir den Job mit dem „sub“-Kommando. Wir erhalten jetzt die Meldung `JOB PRAK222P(JOB00398) SUBMITTED.` Unser Job hat damit die Bezeichnung PRAKxxxP (Abbildung 23).

```
000058 //JOBLIB   DD   DSN=CSQ600.SCSQLOAD,DISP=SHR
000059 //         DD   DSN=CSQ600.SCSQANLE,DISP=SHR
000060 //         DD   DSN=CSQ600.SCSQAUTH,DISP=SHR
000061 //*
000062 //PUTMSGS  EXEC PGM=CSQ4BVK1,REGION=1024K,
000063 //  PARM=(CSQ6,PRAK222,3,Z,5,N)
000064 //*
000065 //*GETMSGS  EXEC PGM=CSQ4BVJ1,REGION=1024K,
000066 //*  PARM=('++QMGR++,++QUEUE++,++MSGS++,++GET++,++SYNC++')
000067 //*
000068 //SYSDBOUT DD   SYSOUT=*


IKJ56250I JOB PRAK222P(JOB00398) SUBMITTED
***
```

Abbildung 23: Der PUT-Job

```
19.32.43 JOB00398 $HASP165 PRAK222P ENDED AT N1  MAXCC=0 CN(INTERNAL)
 ***
```

Abbildung 24: Rückmeldung auf sub

Wir erhalten als Antwort eine Meldung wie in Abbildung 24. MAXCC=0 bedeutet, dass keine Fehler aufgetreten sind. Eine Ausgabe MAXCC=4 ist eine Warnung, man kann sie jedoch auch vernachlässigen. Alles andere bedeutet einen Fehler. In diesem Fall sollten wir uns das Skript noch einmal anschauen.
Sollte z.B. MAXCC=2085 sein, haben wir vermutlich die falsche Bezeichnung für unsere Queue angegeben.

Wir schreiben jetzt also Nachrichten in die Queue. Jetzt ist es Zeit, diese wieder auszulesen. Dafür wenden wir uns dem Member MQGET zu.

Auch hier passen wir wieder die erste Zeilen (die Job-Card) an. Diesmal vergeben wir als Bezeichnung des Jobs PRAKxxxG.
Die restlichen Zeilen passen wir entsprechend Abbildung 25 an.
Achtung, diesmal nehmen wir GETMSGS, nicht PUTMSGS.

```
000058 //JOBLIB   DD   DSN=CSQ600.SCSQLOAD,DISP=SHR
000059 //         DD   DSN=CSQ600.SCSQANLE,DISP=SHR
000060 //         DD   DSN=CSQ600.SCSQAUTH,DISP=SHR
000061 //*
000062 //*PUTMSGS  EXEC PGM=CSQ4BVK1,REGION=1024K,
000063 //*  PARM=('++QMGR++,++QUEUE++,++MSGS++,++PAD++,++LEN++,++PERS++')
000064 //*
000065 //GETMSGS  EXEC PGM=CSQ4BVJ1,REGION=1024K,
000066 //  PARM=(CSQ6,PRAK222,3,D,N)
000067 //*
000068 //SYSDBOUT DD   SYSOUT=*
000069 //SYSABOUT DD   SYSOUT=*
000070 //SYSPRINT DD   SYSOUT=*
000071 //SYSOUT   DD   SYSOUT=*


Command ===> sub                                          Scroll ===> PAGE
 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Abbildung 25: Unser MQGET

Wir übermitteln den Job mit „sub“ und erhalten die Bestätigung, dass der Job PRAKxxxG übertragen wurde.

Auch jetzt sollten wir im Idealfall eine Meldung mit MAXCC=0 bekommen.

Aufgabe: ***Führen Sie die Skripte MQPUT und danach MQGET je einmal aus.***

9. Benutzung von SDSF

Um zu sehen, was wir gerade gemacht haben, gehen wir zunächst durch mehrmaliges Drücken von F3 zurück ins ISPF Primary Option Menu. Von dort aus gelangen wir mit „m;5“ zur Spool Search and Display Facility, SDSF (Abbildung 26).

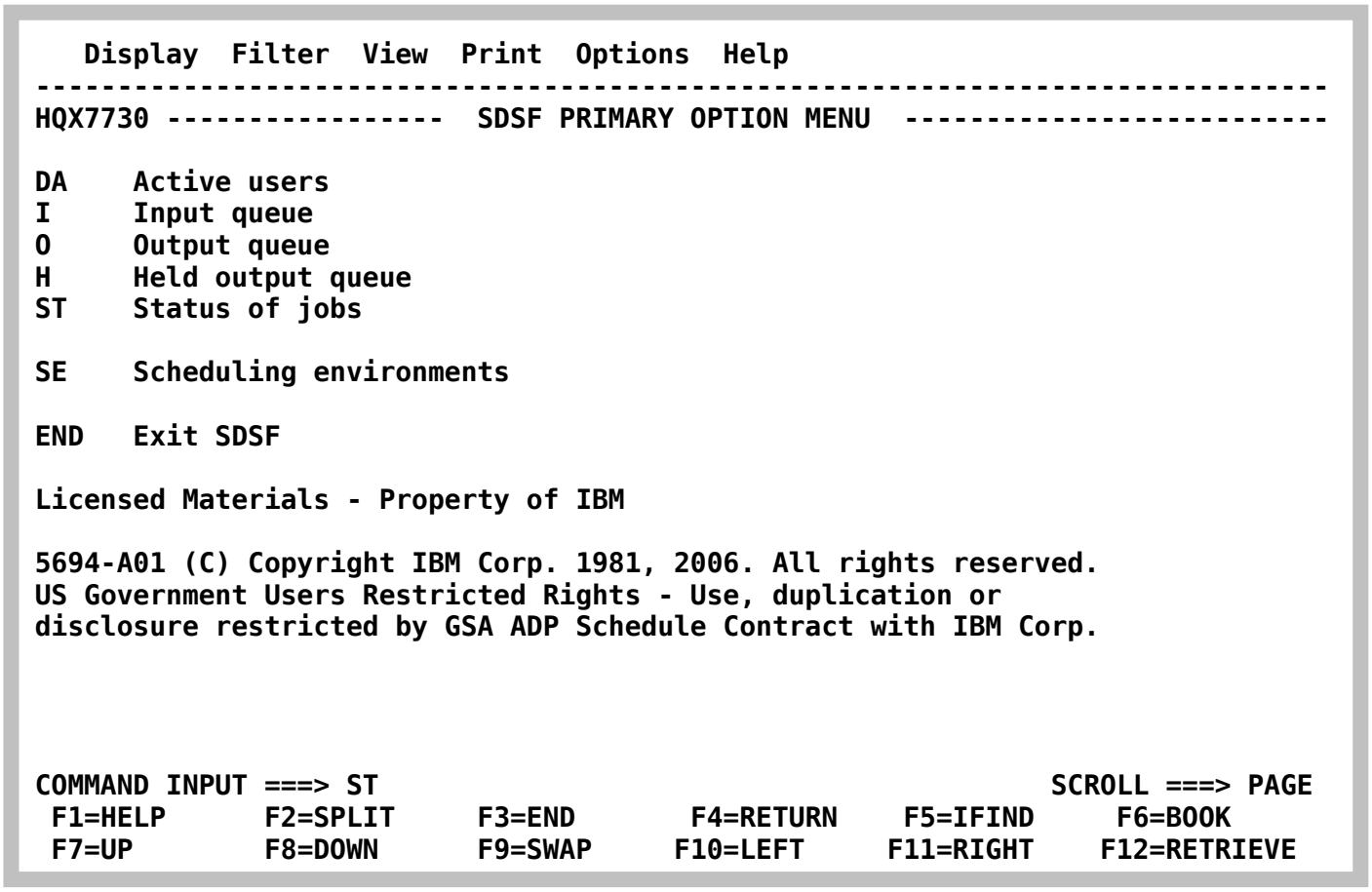


Abbildung 26: SDSF

Wir geben hier „ST“ für „Status of jobs“ ein und gelangen so zu einer Übersicht über alle unsere laufenden Jobs (Abbildung 27).

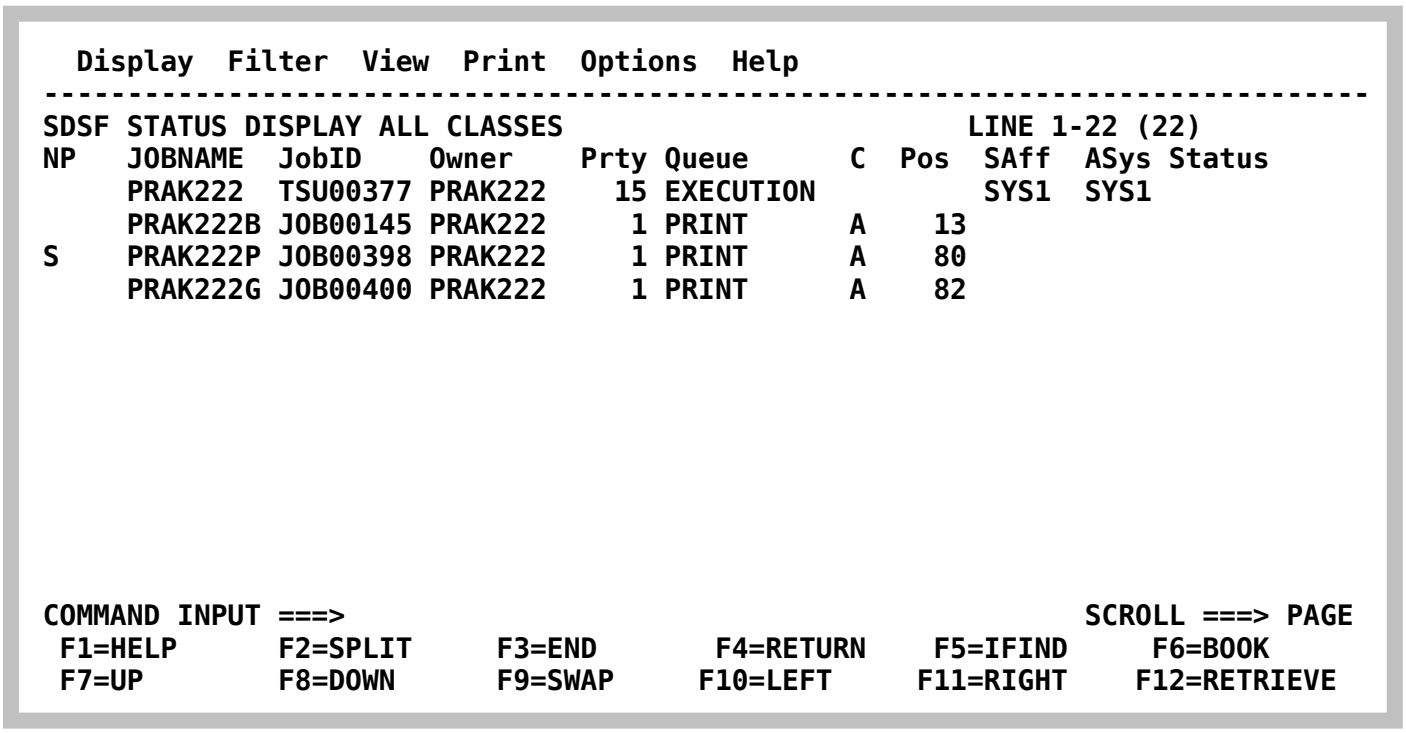


Abbildung 27: SDSF

In der ersten Spalte unseres Put-Jobs (PRAKxxxP) geben wir ein „S“ (für Select) ein und erhalten damit eine Ausgabe unseres Skriptes.

Mit F7 und F8 können wir durch die Ausgabe blättern. Wir blättern bis zum Ende der Ausgabe (Abbildung 28).

```
==========================================
PARAMETERS PASSED :
   QMGR        - CSQ6
   QNAME       - PRAK222
   NUMMSGS     - 000000003
   PADCHAR     - Z
   MSGLENGTH   - 000000005
   PERSISTENCE - N
==========================================
MQCONN SUCCESSFUL
MQOPEN SUCCESSFUL
000000003 MESSAGES PUT TO QUEUE
MQCLOSE SUCCESSFUL
MQDISC SUCCESSFUL
******************************** BOTTOM OF DATA ********************************



  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=IFIND     F6=BOOK
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Abbildung 28 Ausgabe MQPUT

Wie wir sehen, hat das Skript mit unseren Parametern MQCONN und MQOPEN ausgeführt, dann die drei Nachrichten in die Queue gelegt und den Prozess mit MQCLOSE und MQDISC erfolgreich beendet.

Mit F3 gelangen wir zurück zur Übersicht und schauen uns auf die gleiche Weise die Ausgabe von PRAKxxxG an (Abbildung 29) .

```
==========================================
PARAMETERS PASSED :
   QMGR        - CSQ6
   QNAME       - PRAK222
   NUMMSGS     - 000000003
   GET         - D
   SYNCPOINT   - N
==========================================
MQCONN SUCCESSFUL
MQOPEN SUCCESSFUL
000000000 : 000000005 : ZZZZZ
000000001 : 000000005 : ZZZZZ
000000002 : 000000005 : ZZZZZ
000000003 MESSAGES GOT FROM QUEUE
MQCLOSE SUCCESSFUL
MQDISC SUCCESSFUL
******************************** BOTTOM OF DATA ********************************
  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=IFIND     F6=BOOK
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Abbildung 29: Ausgabe MQGET

Wir sehen wieder unsere Parameter und die diversen Erfolgsmeldungen der MQ-Befehle.

Des Weiteren erkennen wir unsere drei abgelegten Nachrichten (ZZZZZ).

Aufgabe: ***Bearbeiten Sie die vorher gegebenen Aufgaben und schicken Sie die Print-Screens 28 und 29 im Bitmap- oder JPEG-Format (pro Bild maximal 250 KByte) an die Mailadresse Ihres Betreuers.***

10. Anhang

Code des vorgefertigten Cobol Programms

CSQ600.SCSQCOBS(CSQ4BVK1)

```
5 000001 CBL NODYNAM,LIB,OBJECT,RENT,RES,APOST
 000002       *                                                               *
 000003       * ------------------------------------------------------------- *
 000004        IDENTIFICATION DIVISION.
 000005       * ------------------------------------------------------------- *
 000006        PROGRAM-ID. CSQ4BVK1.
 000007       *REMARKS
 000008       ****************************************************************
 000009       * @START_COPYRIGHT@                                             *
 000010       * Statement:     Licensed Materials - Property of IBM           *
 000011       *                                                               *
 000012       *                5655-F10                                       *
 000013       *                (C) Copyright IBM Corporation. 1993, 2002      *
 000014       *                                                               *
 000015       * Status:        Version 5 Release 3                            *
 000016       * @END_COPYRIGHT@                                               *
 000017       ****************************************************************
 000018       *                   IBM WebSphere MQ for z/OS                   *
 000019       *                                                               *
 000020       *  Module Name      : CSQ4BVK1                                  *
 000021       *                                                               *
 000022       *  Environment      : z/OS Batch; COBOL II                      *
 000023       *                                                               *
 000024       *  Description      : Sample program to put a number of         *
 000025       *                     messages to a queue.                      *
 000026       *                                                               *
 000027       *  Limitation       : Maximum message length set at 65535.      *
 000028       *                                                               *
 000029       ****************************************************************
 000030       *                                                               *
 000031       *                       Program  Logic                          *
 000032       *                       --------------                          *
 000033       *                                                               *
 000034       *  main                                                         *
 000035       *  ----                                                         *
 000036       *                                                               *
 000037       *   Move parameters into corresponding variables.               *
 000038       *   If parameters are invalid then                              *
 000039       *            Call USAGE-ERROR and exit.                         *
 000040       *                                                               *
 000041       *   Display the parameters passed to the program.               *
 000042       *                                                               *
 000043       *   Connect to the queue manager.                               *
 000044       *   If connection failed then                                   *
 000045       *            Call DISPLAY-ERROR-MESSAGE and exit                *
 000046       *                                                               *
 000047       *   Set the open options.                                       *
 000048       *   Open the specified message queue (MQOPEN).                  *
 000049       *   If open failed then                                         *
 000050       *            Disconnect from queue manager                      *
 000051       *            Call DISPLAY-ERROR-MESSAGE and exit                *
```

CSQ600.SCSQCOBS(CSQ4BVK1) Teil 1

```
000052       *   Endif.                                                    *
000053       *                                                             *
000054       *   Set some message properties.                              *
000055       *   Set the put message options.                              *
000056       *   Loop while the messages are put to queue                  *
000057       *            Put message to queue (MQPUT)                     *
000058       *            If put failed                                    *
000059       *                     Call DISPLAY-ERROR-MESSAGE              *
000060       *                     Break from loop                         *
000061       *            Endif                                            *
000062       *   Endloop.                                                  *
000063       *   Display number of messages put to the queue.              *
000064       *                                                             *
000065       *   Close the message queue.                                  *
000066       *   If close failed then                                      *
000067       *            Call DISPLAY-ERROR-MESSAGE.                      *
000068       *                                                             *
000069       *   Disconnect from the queue manager.                        *
000070       *   If disconnect failed then                                 *
000071       *            Call DISPLAY-ERROR-MESSAGE.                      *
000072       *                                                             *
000073       *   Exit program.                                             *
000074       *                                                             *
000075       *-------------------------------------------------------------*
000076       *                                                             *
000077       *  USAGE-ERROR                                                *
000078       *  -----------                                                *
000079       *                                                             *
000080       *   Print message showing correct usage for program.          *
000081       *                                                             *
000082       *-------------------------------------------------------------*
000083       *                                                             *
000084       *  DISPLAY-ERROR-MESSAGE                                      *
000085       *  ---------------------                                      *
000086       *                                                             *
000087       *   Create error message and display.                         *
000088       *                                                             *
000089       ***************************************************************
000090       * ----------------------------------------------------------- *
000091        ENVIRONMENT DIVISION.
000092       * ----------------------------------------------------------- *
000093       * ----------------------------------------------------------- *
000094        DATA DIVISION.
000095       * ----------------------------------------------------------- *
000096        FILE SECTION.
000097       * ----------------------------------------------------------- *
000098        WORKING-STORAGE SECTION.
000099       * ----------------------------------------------------------- *
000100       *
000101       *    W00 - General work fields
000102       *
000103        01  W00-RETURN-CODE             PIC S9(4) BINARY VALUE ZERO.
000104        01  W00-LOOP                    PIC S9(9) BINARY VALUE 0.
000105        01  W00-NUMPUTS                 PIC S9(9) BINARY VALUE 0.
```

CSQ600.SCSQCOBS(CSQ4BVK1) Teil 2

```
000106        01  W00-ERROR-MESSAGE           PIC X(48) VALUE SPACES.
000107       *
000108       *   Parameter variables
000109       *
000110        01  W00-QMGR                    PIC X(48).
000111        01  W00-QNAME                   PIC X(48).
000112        01  W00-PADCHAR                 PIC X(1) VALUE '*'.
000113        01  W00-MSGBUFFER.
000114          02  W00-MSGBUFFER-ARRAY       PIC X(1) OCCURS 65535 TIMES.
000115        01  W00-NUMMSGS-NUM             PIC 9(4) VALUE  0.
000116        01  W00-NUMMSGS                 PIC S9(9) BINARY VALUE 1.
000117        01  W00-MSGLENGTH-NUM           PIC 9(4) VALUE 0.
000118        01  W00-MSGLENGTH               PIC S9(9) BINARY VALUE 100.
000119        01  W00-PERSISTENCE             PIC X(1) VALUE 'N'.
000120            88 PERSISTENT     VALUE 'P'.
000121            88 NOT-PERSISTENT VALUE 'N'.
000122       *
000123       *    W03 - API fields
000124       *
000125        01  W03-HCONN                   PIC S9(9) BINARY VALUE 0.
000126        01  W03-HOBJ                    PIC S9(9) BINARY VALUE 0.
000127        01  W03-OPENOPTIONS             PIC S9(9) BINARY.
000128        01  W03-COMPCODE                PIC S9(9) BINARY.
000129        01  W03-REASON                  PIC S9(9) BINARY.
000130       *
000131       *    API control blocks
000132       *
000133        01  MQM-OBJECT-DESCRIPTOR.
000134            COPY CMQODV.
000135        01  MQM-MESSAGE-DESCRIPTOR.
000136            COPY CMQMDV.
000137        01  MQM-PUT-MESSAGE-OPTIONS.
000138            COPY CMQPMOV.
000139       *
000140       *    MQV contains constants (for filling in the control blocks)
000141       *    and return codes (for testing the result of a call)
000142       *
000143        01  MQM-CONSTANTS.
000144            COPY CMQV SUPPRESS.
000145       *
000146       *
000147       * ------------------------------------------------------------- *
000148        LINKAGE SECTION.
000149       * ------------------------------------------------------------- *
000150        01  PARMDATA.
000151            05  PARM-LEN                PIC S9(03) BINARY.
000152            05  PARM-STRING             PIC X(100).
000153       *
000154            EJECT
000155       * ------------------------------------------------------------- *
000156        PROCEDURE DIVISION USING PARMDATA.
000157       * ------------------------------------------------------------- *
000158       * ------------------------------------------------------------- *
000159        A-MAIN SECTION.
```

CSQ600.SCSQCOBS(CSQ4BVK1) Teil 3

```
000160       * ------------------------------------------------------------ *
000161       *
000162       *    If no parameters passed to program then
000163       *    call USAGE-ERROR and exit
000164       *
000165            IF PARM-LEN = 0 THEN
000166               PERFORM USAGE-ERROR
000167               MOVE 8 TO W00-RETURN-CODE
000168               GO TO A-MAIN-END
000169            END-IF.
000170       *
000171       *    Move parameters into corresponding variables
000172       *
000173            UNSTRING PARM-STRING DELIMITED BY ALL ','
000174                        INTO W00-QMGR
000175                             W00-QNAME
000176                             W00-NUMMSGS-NUM
000177                             W00-PADCHAR
000178                             W00-MSGLENGTH-NUM
000179                             W00-PERSISTENCE.
000180            MOVE W00-MSGLENGTH-NUM TO W00-MSGLENGTH.
000181            MOVE W00-NUMMSGS-NUM   TO W00-NUMMSGS.
000182       *
000183       *    Display parameters to be used in the program
000184       *
000185            DISPLAY '=========================================='.
000186            DISPLAY 'PARAMETERS PASSED :'.
000187            DISPLAY '   QMGR        - ', W00-QMGR.
000188            DISPLAY '   QNAME       - ', W00-QNAME.
000189            DISPLAY '   NUMMSGS     - ', W00-NUMMSGS.
000190            DISPLAY '   PADCHAR     - ', W00-PADCHAR.
000191            DISPLAY '   MSGLENGTH   - ', W00-MSGLENGTH.
000192            DISPLAY '   PERSISTENCE - ', W00-PERSISTENCE.
000193            DISPLAY '=========================================='.
000194       *
000195       *    Setup the message buffer
000196       *
000197            PERFORM WITH TEST BEFORE VARYING W00-LOOP FROM 1 BY 1
000198                UNTIL (W00-LOOP > W00-MSGLENGTH)
000199       *
000200                MOVE W00-PADCHAR TO W00-MSGBUFFER-ARRAY(W00-LOOP)
000201       *
000202            END-PERFORM.
000203       *
000204       *
000205       *    Connect to the queue manager
000206       *
000207            CALL 'MQCONN' USING W00-QMGR
000208                                W03-HCONN
000209                                W03-COMPCODE
000210                                W03-REASON.
000211       *
000212       *    If connection failed then display error message
000213       *    and exit
```

CSQ600.SCSQCOBS(CSQ4BVK1) Teil 4

```
000214       *
000215            IF (W03-COMPCODE NOT = MQCC-OK) THEN
000216               MOVE 'MQCONN'   TO W00-ERROR-MESSAGE
000217               PERFORM DISPLAY-ERROR-MESSAGE
000218               MOVE W03-REASON TO W00-RETURN-CODE
000219               GO TO A-MAIN-END
000220            END-IF.
000221            DISPLAY 'MQCONN SUCCESSFUL'.
000222       *
000223       *
000224       *    Open the queue for output. Fail the call if the queue
000225       *    manager is quiescing.
000226       *
000227            COMPUTE W03-OPENOPTIONS = MQOO-OUTPUT +
000228                                      MQOO-FAIL-IF-QUIESCING.
000229
000230            MOVE W00-QNAME   TO MQOD-OBJECTNAME.
000231       *
000232            CALL 'MQOPEN' USING W03-HCONN
000233                                MQOD
000234                                W03-OPENOPTIONS
000235                                W03-HOBJ
000236                                W03-COMPCODE
000237                                W03-REASON.
000238       *
000239       *    If open failed then display error message
000240       *    and exit.
000241       *
000242            IF (W03-COMPCODE NOT = MQCC-OK) THEN
000243               MOVE 'MQOPEN'   TO W00-ERROR-MESSAGE
000244               PERFORM DISPLAY-ERROR-MESSAGE
000245               MOVE W03-REASON TO W00-RETURN-CODE
000246               GO TO A-MAIN-DISCONNECT
000247            END-IF.
000248            DISPLAY 'MQOPEN SUCCESSFUL'.
000249       *
000250       *    Set persistence depending on parameter passed
000251       *
000252            IF PERSISTENT THEN
000253               MOVE MQPER-PERSISTENT     TO MQMD-PERSISTENCE
000254            ELSE
000255               MOVE MQPER-NOT-PERSISTENT TO MQMD-PERSISTENCE
000256            END-IF.
000257       *
000258       *    Put string format messages
000259       *
000260            MOVE MQFMT-STRING TO MQMD-FORMAT.
000261       *
000262       *    Set the put message options to fail the call if the
000263       *    queue manager is quiescing
000264       *
000265            MOVE MQPMO-FAIL-IF-QUIESCING TO MQPMO-OPTIONS.
000266       *
000267       *    Loop until specified number of messages put to queue
```

CSQ600.SCSQCOBS(CSQ4BVK1) Teil 5

```
000268       *
000269            PERFORM WITH TEST BEFORE VARYING W00-LOOP FROM 0 BY 1
000270                UNTIL (W00-LOOP >= W00-NUMMSGS)
000271       *
000272                MOVE MQMI-NONE TO MQMD-MSGID
000273                MOVE MQCI-NONE TO MQMD-CORRELID
000274       *
000275                CALL 'MQPUT' USING W03-HCONN
000276                                   W03-HOBJ
000277                                   MQMD
000278                                   MQPMO
000279                                   W00-MSGLENGTH
000280                                   W00-MSGBUFFER
000281                                   W03-COMPCODE
000282                                   W03-REASON
000283       *
000284       *        If put failed then display error message
000285       *        and break out of loop
000286       *
000287                IF (W03-COMPCODE NOT = MQCC-OK) THEN
000288                   MOVE 'MQPUT'     TO W00-ERROR-MESSAGE
000289                   PERFORM DISPLAY-ERROR-MESSAGE
000290                   MOVE W00-NUMMSGS TO W00-LOOP
000291                   MOVE W03-REASON  TO W00-RETURN-CODE
000292                ELSE
000293                   ADD 1 TO W00-NUMPUTS
000294                END-IF
000295       *
000296            END-PERFORM.
000297       *
000298       *    Display the number of messages successfully put
000299       *    to the queue
000300       *
000301            DISPLAY W00-NUMPUTS, ' MESSAGES PUT TO QUEUE'.
000302       *
000303       *
000304       *    Close the queue
000305       *
000306            CALL 'MQCLOSE' USING W03-HCONN
000307                                 W03-HOBJ
000308                                 MQCO-NONE
000309                                 W03-COMPCODE
000310                                 W03-REASON.
000311            IF (W03-COMPCODE NOT = MQCC-OK) THEN
000312               MOVE 'MQCLOSE'  TO W00-ERROR-MESSAGE
000313               PERFORM DISPLAY-ERROR-MESSAGE
000314               MOVE W03-REASON TO W00-RETURN-CODE
000315            ELSE
000316               DISPLAY 'MQCLOSE SUCCESSFUL'
000317            END-IF.
000318       *
000319       *
000320       *
000321        A-MAIN-DISCONNECT.
```

CSQ600.SCSQCOBS(CSQ4BVK1) Teil 6

```
000322       *
000323       *    Disconnect from the queue manager
000324       *
000325            CALL 'MQDISC' USING W03-HCONN
000326                                W03-COMPCODE
000327                                W03-REASON.
000328            IF (W03-COMPCODE NOT = MQCC-OK) THEN
000329               MOVE 'MQDISC'   TO W00-ERROR-MESSAGE
000330               PERFORM DISPLAY-ERROR-MESSAGE
000331               MOVE W03-REASON TO W00-RETURN-CODE
000332            ELSE
000333               DISPLAY 'MQDISC SUCCESSFUL'
000334            END-IF.
000335       *
000336        A-MAIN-END.
000337       *
000338       *
000339            MOVE W00-RETURN-CODE TO RETURN-CODE
000340            STOP RUN.
000341       *
000342       * ------------------------------------------------------------ *
000343        USAGE-ERROR SECTION.
000344       * ------------------------------------------------------------ *
000345       *
000346            DISPLAY '================================================'.
000347            DISPLAY 'PARAMETERS FOR PROGRAM :'.
000348            DISPLAY '     QMGR        - QUEUE MANGER'.
000349            DISPLAY '     QNAME       - QUEUE NAME'.
000350            DISPLAY '     NUMMSGS     - NUMBER OF MESSAGES'.
000351            DISPLAY '     PADCHAR     - MESSAGE PADDING CHARACTER'.
000352            DISPLAY '     MSGLENGTH   - LENGTH OF MESSAGE(S)'.
000353            DISPLAY '     PERSISTENCE - PERSISTENCE OF MESSAGE(S)'.
000354            DISPLAY '================================================'.
000355       *
000356        USAGE-ERROR-END.
000357       *
000358       *    RETURN TO PERFORMING FUNCTION
000359       *
000360            EXIT.
000361       *
000362       * ------------------------------------------------------------ *
000363        DISPLAY-ERROR-MESSAGE SECTION.
000364       * ------------------------------------------------------------ *
000365       *
000366            DISPLAY '************************************************'.
000367            DISPLAY '* ', W00-ERROR-MESSAGE.
000368            DISPLAY '* COMPLETION CODE : ', W03-COMPCODE.
000369            DISPLAY '* REASON CODE     : ', W03-REASON.
000370            DISPLAY '************************************************'.
000371       *
000372        DISPLAY-ERROR-MESSAGE-END.
000373       *
000374       *    RETURN TO PERFORMING FUNCTION
000375       *
```

CSQ600.SCSQCOBS(CSQ4BVK1) Teil 7

```
000376            EXIT.
000377       *
000378       * ---------------------------------------------------------- *
000379       *                          END OF PROGRAM
000380       * ---------------------------------------------------------- *
```

CSQ600.SCSQCOBS(CSQ4BVK1) Teil 8

EJB3 Zugriff auf DB2
Installation und Konfiguration

Unsere EJB Tutorials bestehen aus drei Teilen:

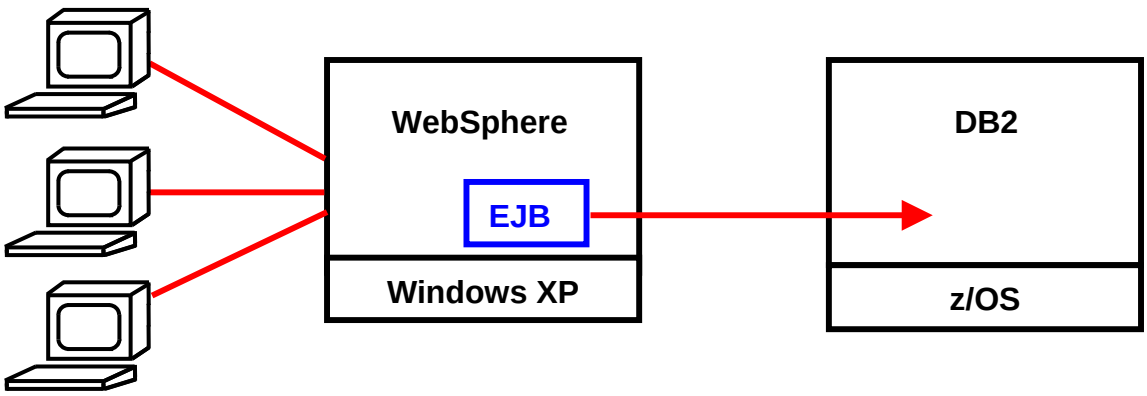
1. **Erstellen einer Test- und Entwicklungsumgebung mit WebSphere**
2. **EJB Zugriff auf z/OS DB2**
3. **Java Message driven Bean**

Dies ist Teil 1 mit folgendem Inhalt:

1. **Übersicht**
2. **Windows XP virtuelle Maschine**
3. **Konfiguration der Data Source**

1. Übersicht

Ziel von Teil 2. dieses Tutorials ist es, mittels einer Enterprise Java Bean (EJB) auf eine DB2-Datenbank unseres Rechners leia.informatik.uni-leipzig.de zuzugreifen.



Wir benutzen eine typische 3-Tier Konfiguration, mit Server-zentrischer Präsentationslogik.

Der WebSphere Application Server ist in unterschiedlichen Versionen verfügbar. Wir benutzen die einfachste Version, die auf der IBM Home Page kostenlos zur Verfügung gestellt wird, und auf einer Geronimo Engine der Apache Foundation aufbaut.

Enterprise JavaBeans (EJB) sind standardisierte Komponenten innerhalb eines Java Enterprise Edition (JEE)-Servers. Sie vereinfachen die Entwicklung komplexer mehrschichtiger verteilter Softwaresysteme mittels Java. Mit Enterprise Java Beans können wichtige Konzepte für Unternehmensanwendungen, z.B. Transaktions-, Namens- oder Sicherheitsdienste, umgesetzt werden, die für die Geschäftslogik einer Anwendung nötig sind.

Die EJB Spezifikation wurde ursprünglich in 1997 von IBM entwickelt und 1999 von Sun Microsystems als EJB 1.0 und 1.1 verbessert. Der Java Community Process brachte 2001 EJB 2.0 heraus.

Die Komplexität und die fehlende Objektorientiertheit der EJB-Technologie waren in der Vergangenheit Kritikpunkte. Entsprechend dem Feedback von einem großen Teil der J2EE Community betrachten viele Experten die Entwicklung mit früheren Versionen von JEE als unnötig komplex, obwohl alles gut funktioniert. Bei der Entwicklung mussten die Entwickler mehr Zeit verbringen, um APIs für den EJB-Container zu schreiben, als für die Implementierung der Geschäftslogik. Zum Beispiel:

- **Die EJB 2.x-Spezifikation erfordert, dass eine Session oder Entity-Bean die Remote und/oder die Home-Schnittstelle implementieren muss, welche die Schnittstelle aus dem EJB Framework-Paket erweitert (extends). Dies bewirkt eine enge Kopplung zwischen dem Entwickler-geschriebenen Code und den Interface-Klassen aus dem EJB Framework-Paket. Die Folge ist eine langweilige und wiederholte Arbeit, um mehrere unnötigen Callback-Methoden (ejbCreate (), ejbPassivate (), ejbActivate ()) zu implementiern, die keinen direkten Bezug zu der Business-Logik haben. Darüber hinaus müssen Exceptions, die durch die unnötigen Methoden verursacht werden, gehandhabt werden.**
- **Die XML-Deployment-Deskriptoren sind übermäßig verbose und komplex. Ein Großteil der Informationen in dem Deployment-Deskriptor könnten mit Standardwerten (Defaults) definiert werden.**
- **Auf Ressourcen muss durch JNDI zugegriffen werden.**
- **Das Container-Managed Persistence Modell ist kompliziert zu entwickeln und zu verwalten.**
- **Das Persistenz Mapping-Modell war nie gut in EJB2.x definiert. Dis bewirkt, dass Entity Beans nicht für alle J2EE-Container ohne Veränderungen passen.**

Diese negativen Aspekte haben die folgenden Gründe:

- **Die J2EE-Entwicklung ist zu komplex.**
- **Zu komplexe XML-basierte Konfiguration.**
- **Das Persistenz-Modell gilt heute als überholt.**

Aus diesem Grunde wurde 2006 die neue EJB 3.0-Spezifikation entwickelt, die eine deutliche Vereinfachung bringen soll. EJB 3.0 stellt einen erheblichen Bruch mit der Vergangenheit dar. Hervorzuheben ist insbesondere die Einführung eines neuen Plain Old Java Object (POJO)-basierten Programmiermodells, das die Entwicklung von J2EE-Anwendungen merklich vereinfacht. Das heißt, der Code muss nicht durch EJB-Implementierungsdetails „verschmutzt“ werden. Ein wesentliches Merkmal in EJB3 ist die breite Verwendung von Annotationen. Diese ermöglichen innovative Techniken wie Metadata Annotations, Lifecycle Interceptors, Dependency Injection, EJB Injection und Java Persistence API (JPA).

In der EJB3 Entwicklung Modell, müssen wir nur 2 einfache Schnittstellen definieren, deren Beziehung aus der Klasse hervorgeht, ohne (oder mit einer stark reduzierten) Konfiguration in einem Deployment Descriptor.

Die wichtigsten Neuerungen bei EJB3.0:

- **Annotations, die Angaben in den Deployment Deskriptoren ergänzen oder überschreiben**
- **Convention über Declarationen**
- **Verzicht auf das Homeinterface bei Session Beans**

Annotations werden als Statements in den normalen Java code eingefügt. Ein Annotation Statement beginnt mit dem Symbol @. Hier ist ein Beispiel, wie eine EJB mit Annotations aussieht:

```
public interface HelloService {
  String sayHello();
}

@Stateless
@Local(HelloService.class)
@Remote(HelloServiceRemote.class)
public class HelloServiceBean implements HelloService {
  public String sayHello() { return “Hello world”; }
}

public interface HelloServiceRemote {
  String sayHello();
}
```

Die EJB 3.1 (2009) Spezifikation und die EJB 3.2 (2012) Spezifikation brachten weitere Verbesserungen.

Eine Enterprise Application muss auf einem JEE-Anwendungsserver eingesetzt werden. Dieses Tutorial verwendet WebSphere Application Server 6.1. WebSphere Application Server 6.1 ist JEE-kompatibel und unterstützt Java Standard Edition 1.5. Mit der Installation des EJB3 Feature Packs bietet WebSphere Application Server 6.1 eine EJB 3.0 Unterstützung.

Das vorliegende Tutorium entstand aus einer Masterarbeit von Herrn Li Zhang. Sie können die Arbeit unter http://www.cedix.de/DiplArb/LiZheng.pdf herunterladen.

Java Enterprise Edition ist trotz mehr als 12-jähriger Entwicklung immer noch nicht sehr stabil. Einzelheiten unter: Doug Lyon: The Java Tree Withers. IEEE Computer, Jan. 2012, p.83-85, http://www.cedix.de/VorlesMirror/Band3/Lyon.pdf.

Als Folge implementieren die Hersteller neue JEE Spezifikationen nur mit einer nicht unerheblichen Zeitverzögerung in ihre Software Produkte. Zum Zeitpunkt der Erstellung der Masterarbeit war EJB 3.0 unter WebSphere soeben erst verfügbar geworden; EJB 3.1 kam erst deutlich später. Die zusätzlichen EJB 3.1 Erweiterungen wären allerdings für das vorliegende Tutorium auch uninteressant.

Hinweise

WinXP Login: UNILP
Passwort: unilp

WebSphere Application Server 6.1 (WAS 6.1),
Login: admin
Passwort: admin

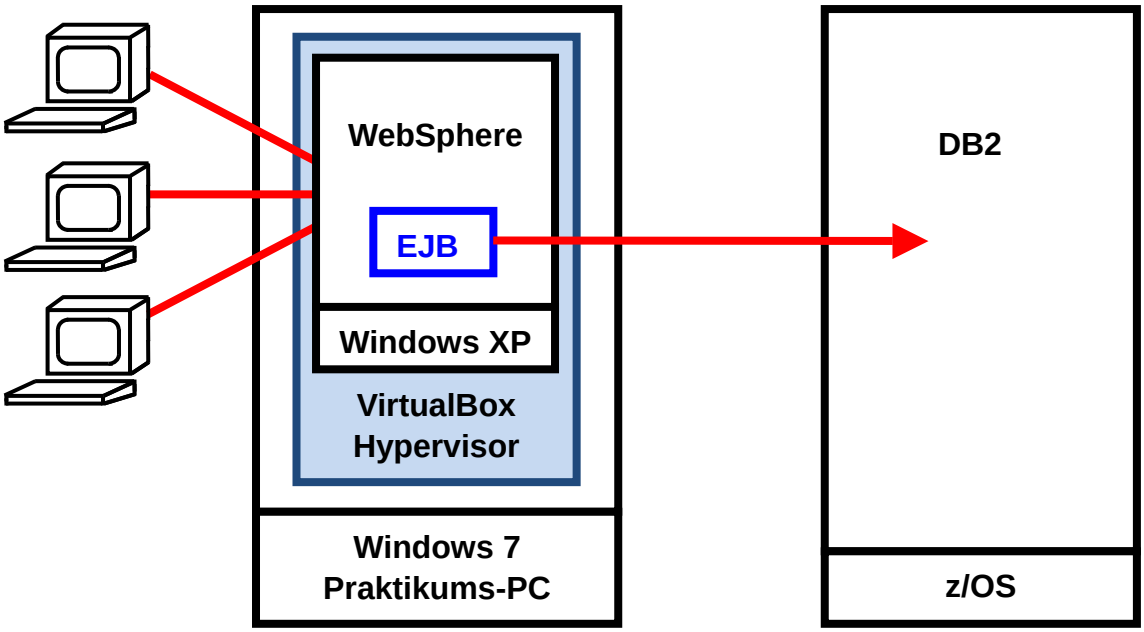
Starten/Stoppen des WAS 6.1 Server:
Programme -> IBM WebSphere -> Application Server V6.1 -> Profiles -> AppSrv01 -> Start/Stop the server

In dem hier vorliegenden Teil 1 der drei EJB Tutorials installieren wir eine Test- und Entwicklungsumgebung.

2. Windows XP virtuelle Maschine

Installation, Konfiguration, Anpassung und Einrichtung der Test- und Entwicklungsumgebung ist ein komplexer Vorgang. Wir möchten Sie nicht davon abhalten, diese Aktivität selbst vorzunehmen. Zu diesem Zweck müssen Sie die kostenlose WebSphere Application Server Community Edition (CE) aus dem Netz herunterladen (z.B. http://www-03.ibm.com/software/products/us/en/appserv-wasce) und falls erforderlich, das ebenfalls kostenlose EJB 3.0 Feature Pack. Wir empfehlen, das Ganze in einer virtuellen Maschine auf Ihrem PC zu installieren.

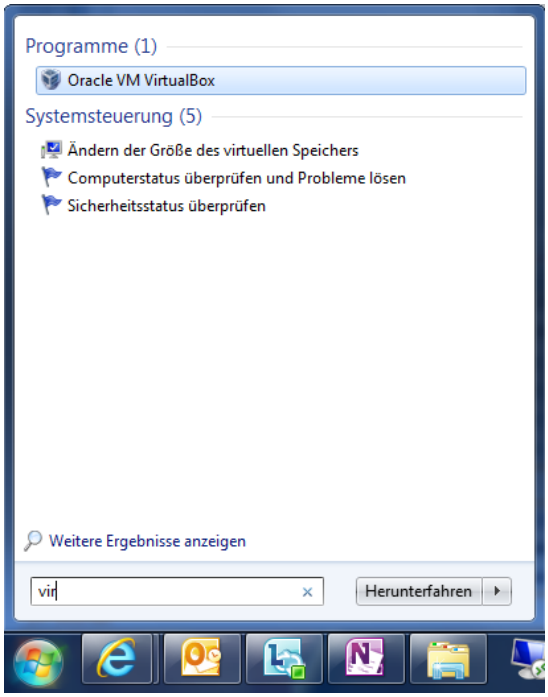
Zur Vereinfachung haben wir dies jedoch bereits für Sie vorgefertigt. Hierzu existiert eine virtuelle Maschine mit VirtualBox, in der ein Windows XP System mit der Entwicklungsumgebung einschließlich WebSphere bereits vorinstalliert ist. Diese virtuelle Maschine ist auf dem Praktikums-PC der Abteilung Technische Informatik eingerichtet und kann dort gestartet werden.



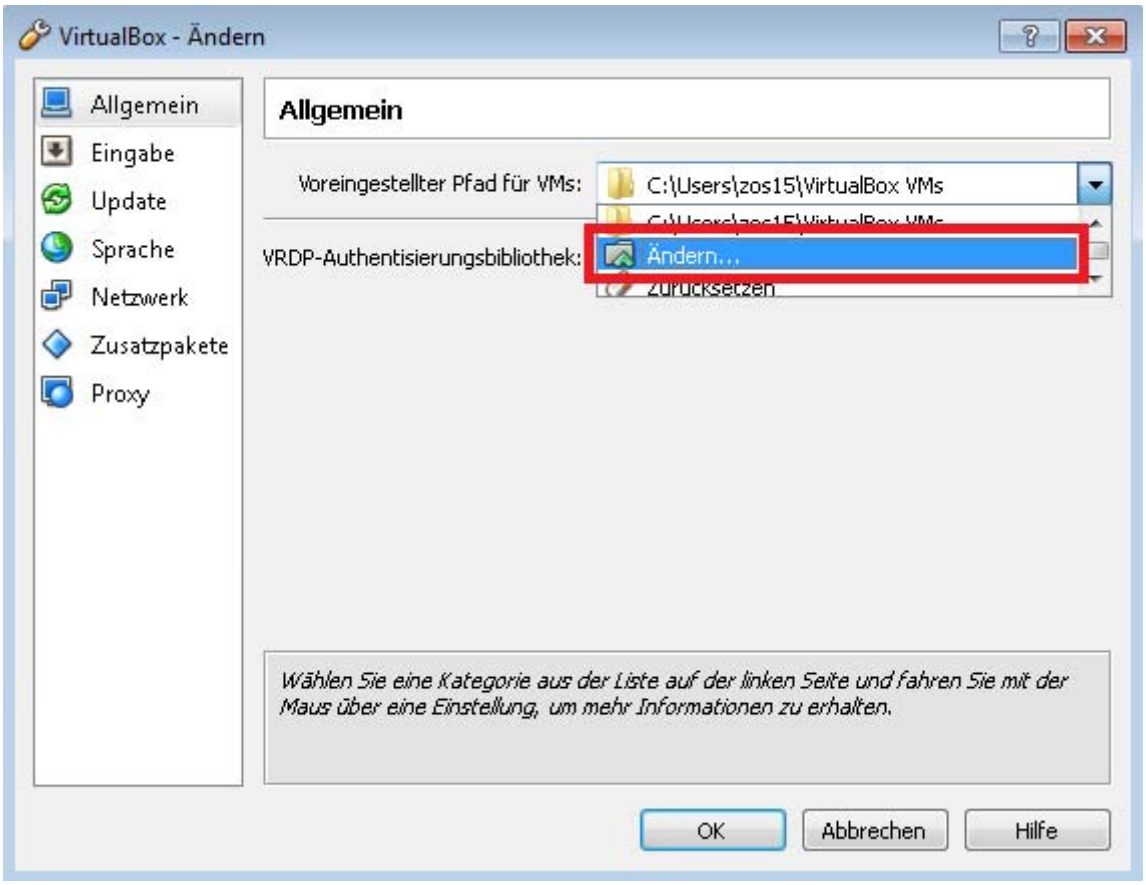
Das Praktikum kann auch auf dem eigenen Rechner durchgeführt werden, dazu ist das Virtual Disk Image (VDI) entsprechend zu kopieren.

Schritt 1 Oracle VM VirtualBox starten und konfigurieren

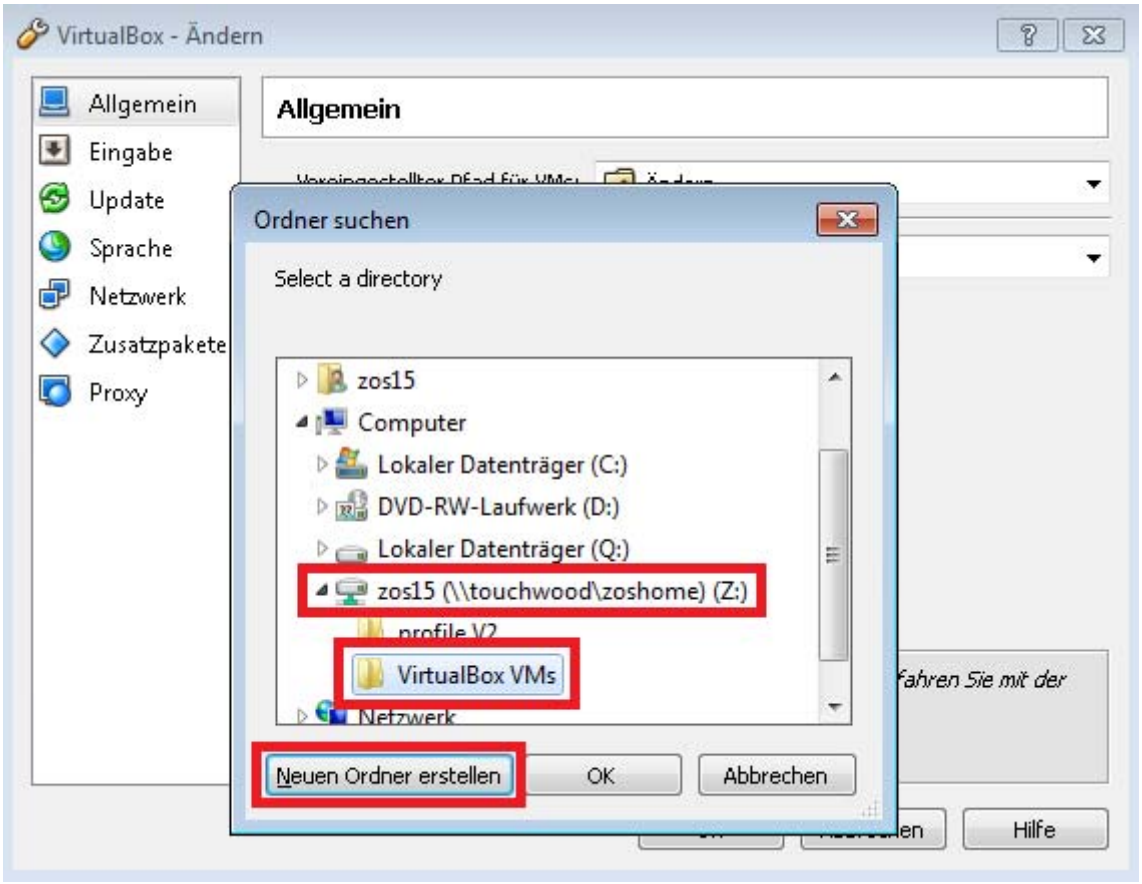
- **Starten Sie das Virtualisierungsprogramm Oracle VM VirtualBox.**



- **Datei**
- **Globale Einstellungen…**
- **Ändern…**


VirtualBox - Ändern
Allgemein
Eingabe
Update
Sprache
Netzwerk
Zusatzpakete
Proxy
Allgemein
Voreingestellter Pfad für VMs:
C:\Users\zos15\VirtualBox VMs
VRDP-Authentisierungsbibliothek:
Ändern...
Zurücksetzen
Wählen Sie eine Kategorie aus der Liste auf der linken Seite und fahren Sie mit der Maus über eine Einstellung, um mehr Informationen zu erhalten.
OK
Abbrechen
Hilfe

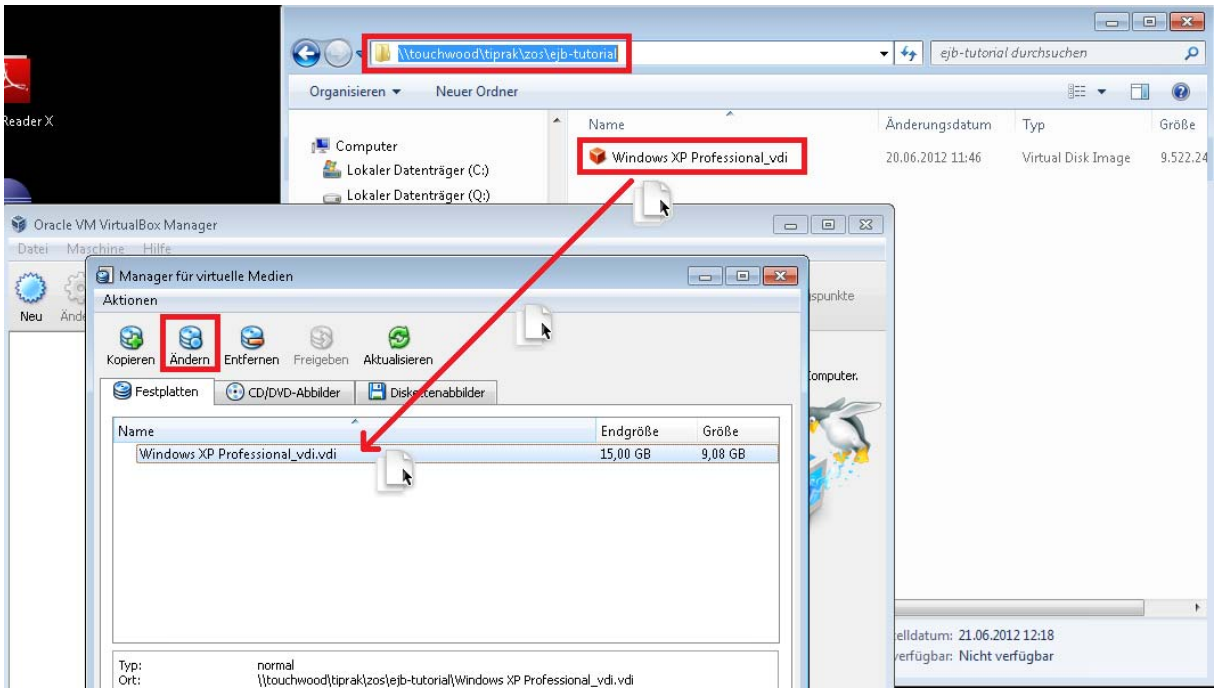
- **Laufwerk zosxx (\\touchwood\zoshome) (Z:) auswählen**
- **Neuen Ordner erstellen**
- **Namen des Ordners in VirtualBox VMs ändern**



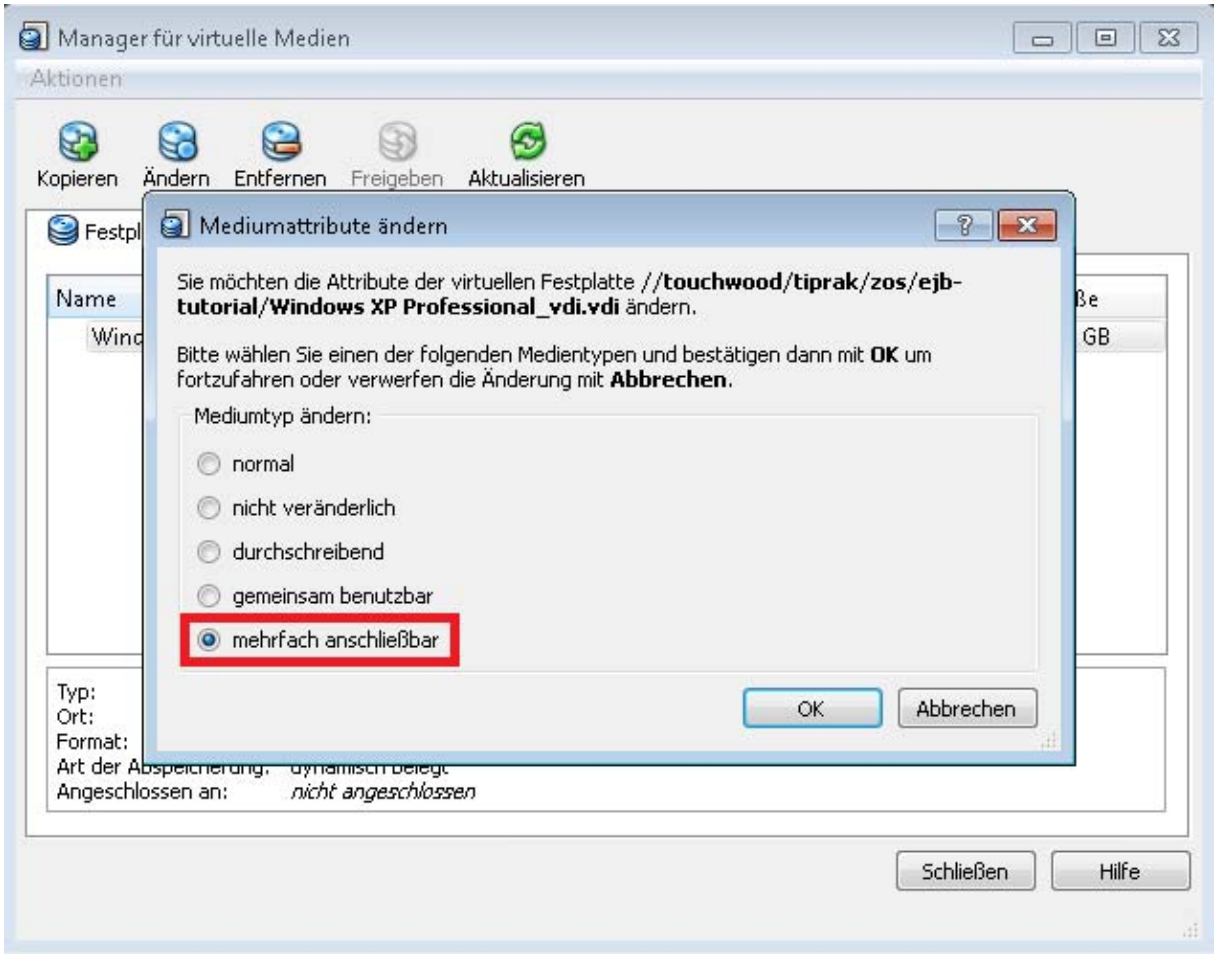
- **VirtualBox VMs (Ordner auswählen)**
- **OK**
  Alle Einstellungen und neuen Virtual Disk Image (VDI)-Dateien sowie Snapshots der VMs werden nun auf Laufwerk Z: und daher nicht im Profilordner (C:\Users\zosxx) abgelegt. Dadurch wird beim An- und Abmelden an einen Praktikums-PC der gesamte Inhalt des Ordners VirtualBox VMs nicht jedes Mal mit dem Server komplett herunter- und hochgeladen (sondern während der Sitzung automatisch synchronisiert). Die Größe des Ordners VirtualBox VMs kann während der Bearbeitung der Tutorien auf mehrere GB ansteigen!
- **OK**

Schritt 2 VM mit vorhandener VDI erzeugen

- **Datei**
- **Manager für virtuelle Medien…**
- **Windows Explorer starten**
- **In Adresszeile \\touchwood\tiprak\zos\ejb-tutorial eingeben**
- **VDI-Datei Windows XP Professional_vdi in das Fenster von Manager für virtuelle Medien hineinziehen (drag&drop)**



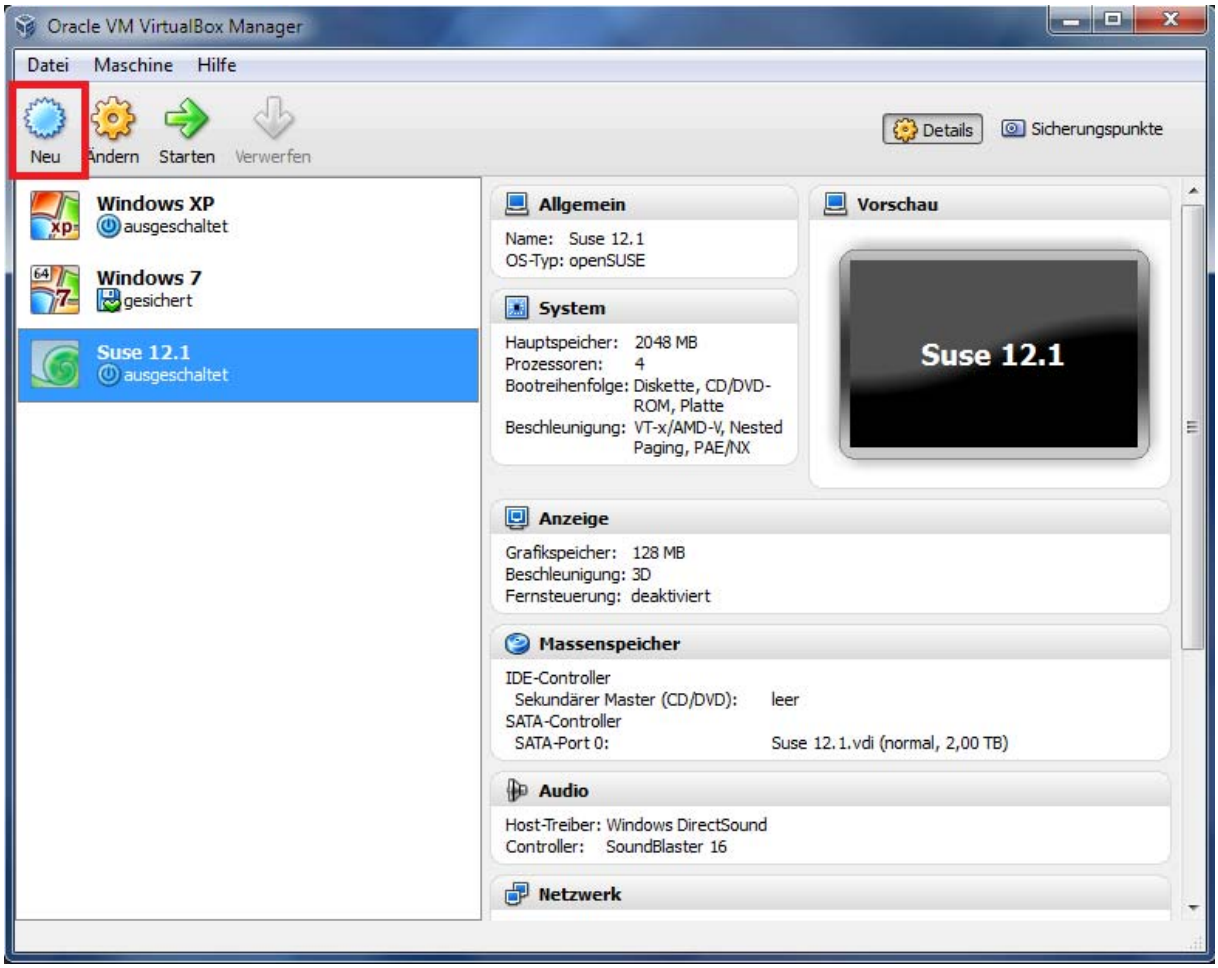
- **Ändern**
- **Medientyp mehrfach anschließbar auswählen**



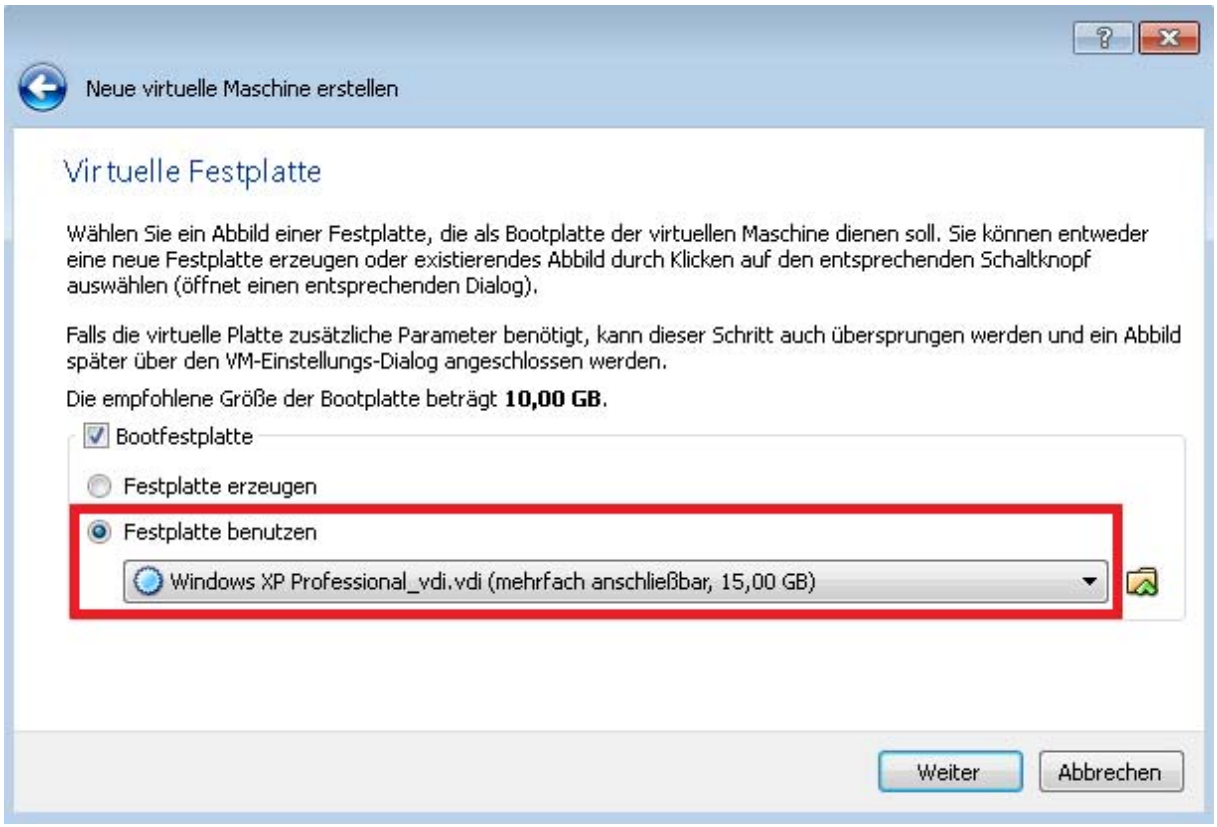
- **OK**

Das VDI ist jetzt konfiguriert. Mehrfach anschließbar bedeutet, dass alle Änderungen, die durch eine VM auf dem virtuellen Medium durchgeführt werden, nicht in dieser VDI-Datei landen, sondern in einem separaten Differenzimage abgelegt werden. In VirtualBox werden diese Differenzimages auch als Snapshots bezeichnet. Später durch angelegte Sicherungspunkte erzeugte Snapshots sind also auch jeweils ein Differenzimage. Alle Differenzimages einer VM werden im Ordner VirtualBox VMs\VM-Name abgelegt.

- **Neu**



- **Weiter**
- **Vergeben Sie der virtuellen Maschine (VM) einen Namen**
  Der Name erscheint anschließend in der obigen Liste. Da der Benutzername des Windows-Accounts in der VM unilp heißt, wird als VM-Namen unilp empfohlen.
- **Weiter**
- **Stellen Sie die Größe des Hauptspeichers auf 1024MB ein**
  Es können auch größere Werte eingestellt werden, für das Praktikum reichen 1GB RAM für die VM für ein flüssiges Arbeiten allerdings aus (die Praktikums-PCs besitzen 12GB RAM).
- **Weiter**
- **Wählen Sie „Festplatte benutzen“ aus**
- **Wählen Sie als virtuelles Plattenabbild die Datei Windows XP Professional_vdi.vdi aus**

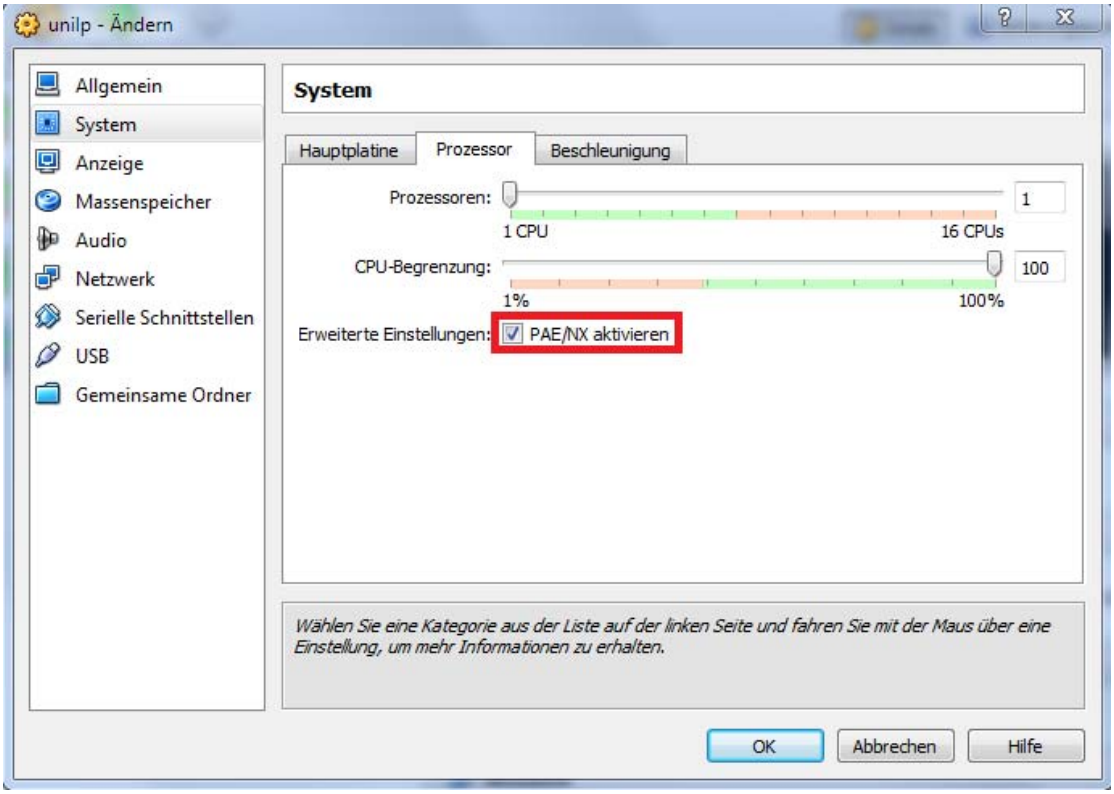


- **Weiter**
- **Erzeugen**

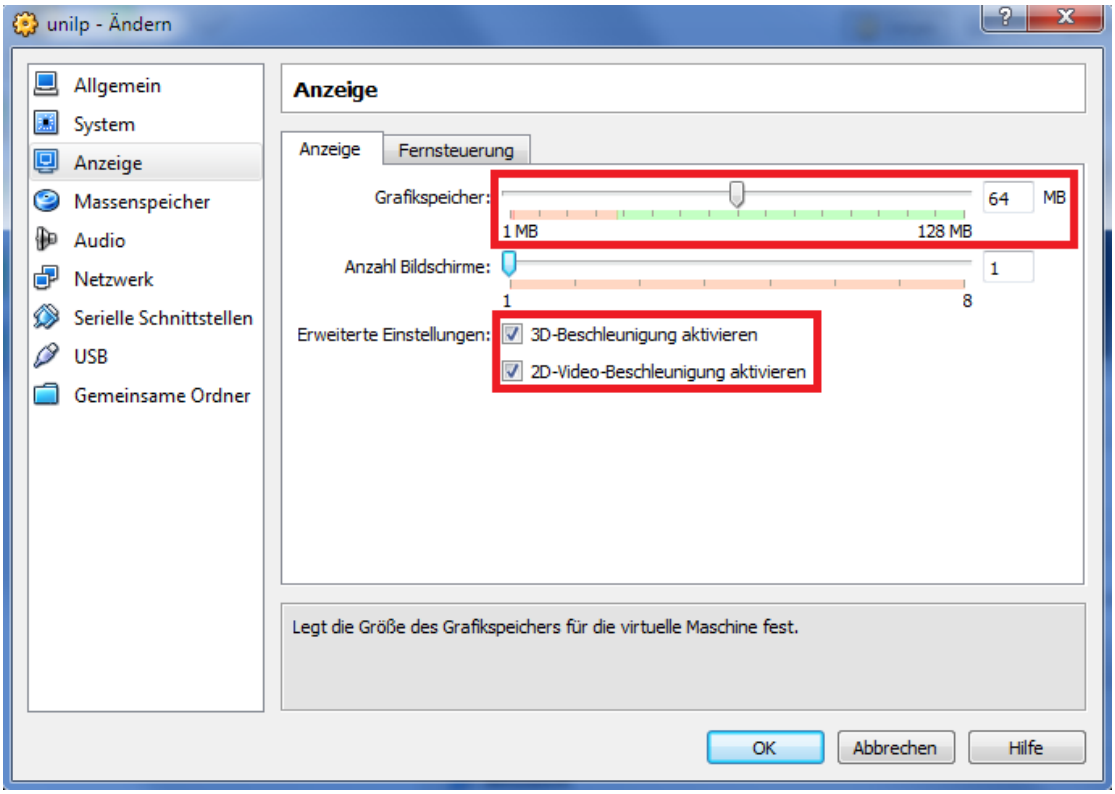
Schritt 3 Virtuelle Maschine (VM) konfigurieren

Dieser Schritt dient dazu, die VM für den Betrieb optimal zu konfigurieren. Mit diesen Einstellungen wurde die VM getestet, so dass sie auf den Praktikums-PCs schnell und einwandfrei läuft. Benutzen Sie zuhause z.B. eine Mehrbildschirmumgebung, so kann es sinnvoll sein, entsprechende Anpassungen an der Konfiguration vorzunehmen.

- **Ändern**
- **System**
- **Prozessor**
- **Häkchen bei „PAE/NX aktivieren“ setzen**
  Dadurch wird die „Physical Adress Extension“ (Physikalische Adresserweiterung) sowie das „No eXecute“-Bit (hardwareunterstützte Dateiausführungsverhinderung) aktiviert.



- **Kontrollieren, ob im Reiter Beschleunigung bereits die Häkchen bei „VT-x/AMD-V aktivieren“ sowie „Nested Paging aktivieren“ gesetzt sind**
  ➡VT-X/AMD-V ermöglicht VirtualBox, den Gast (hier Windows XP) para- anstatt vollvirtualisiert auszuführen, was zu einer deutlich höheren Performance des Gastes führt. Paravirtualisierung wird auch Hardwarevirtualisierung genannt und Vollvirtualisierung dementsprechend Softwarevirtualisierung. Mit VT-x/AMD-V können Prozesse des Gastes direkt auf der Host-Hardware arbeiten, weil der Prozessor zusätzliche Sicherheitsfunktionen (Abschottung) bereitstellt, welche verhindern, dass die Gastprozesse in Hostprozesse eingreifen können.
  ➡Nested Paging, auch bekannt als Rapid Virtualization Indexing, ist eine hardwarebasierte Virtualisierung für die Memory Management Unit (MMU) des Prozessor, um die Berechnungen in den Seitentabellen für die Adressierung des virtuellen Speichers des Gastes in Hardware durchzuführen (auch Shadow Page Tables genannt).
- **Anzeige**
- **Als Grafikspeicher 64 MB einstellen**
- **Häkchen bei „3D-Beschleunigung aktivieren“ sowie „2D-Video-Beschleunigung aktivieren“ setzen**



- **Für den Datenaustausch können Sie unter „gemeinsame Ordner“ einen Ordner des Hosts der virtuellen Maschine zur Verfügung stellen. Der verwendete Gast bietet allerdings bereits eine Dateifreigabe über das Netzwerk an.**
- **OK**

Die VM ist jetzt fertig eingerichtet. Sie können die VM jetzt starten. Das Passwort für den eingerichteten Benutzer unilp lautet unilp. Der Benutzername und Passwort für die Websphere-Administration-Backends sind identisch (unilp/unilp).

3. Konfiguration der Data Source

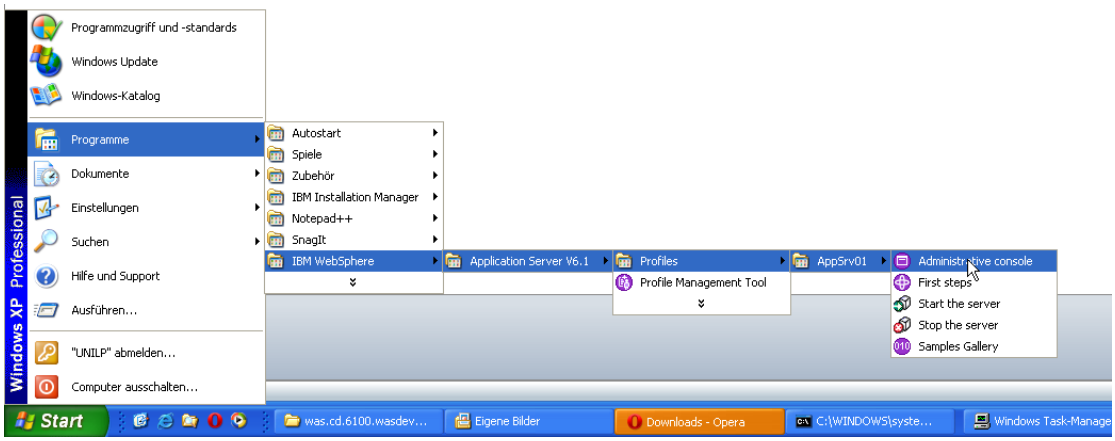
In diesem Abschnitt erläutern wir, wie Sie eine DB2 Data Source auf dem z/OS System leia.informatik.uni-leipzig.de einrichten können. Bitte besorgen Sie sich vorher einen DB2 Benutzernamen mit Passwort von Ihrem Betreuer. In diesem Text verwenden wir den Benutzernamen prak224; bitte ersetzen Sie ihn durch Ihren eigenen.

Anmerkung: Wir haben das Tutorium mit dem Internet Explorer ausgetestet. Wir können nicht garantieren, dass es mit einem anderen Browser funktioniert.

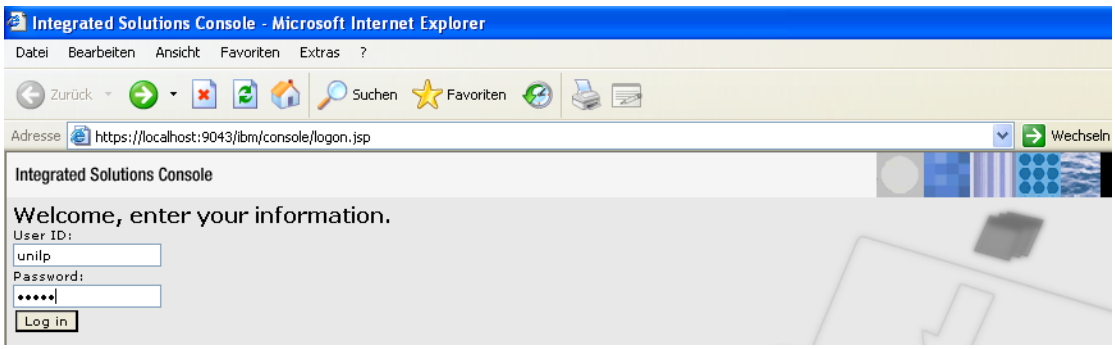
Wir verwenden die folgenden Abkürzungen

1k einmal mit der linken Maustaste klicken
2k zweimal mit der linken Maustaste klicken
1kr einmal mit der rechten Maustaste klicken

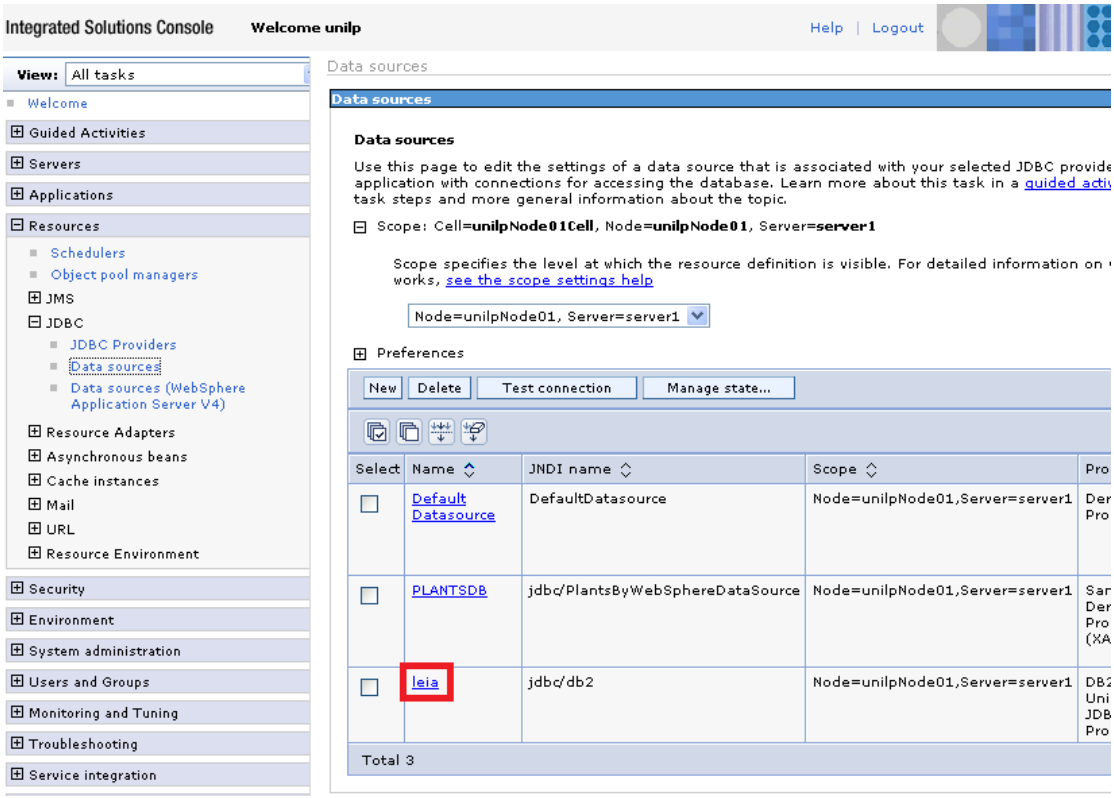
Schritt 1 1k Administive console um die Login Seite auf Ihrem Browser zu öffnen.



Schritt 2 Geben Sie Benutzername und Passwort ein, dann 1k auf denn Login Button.

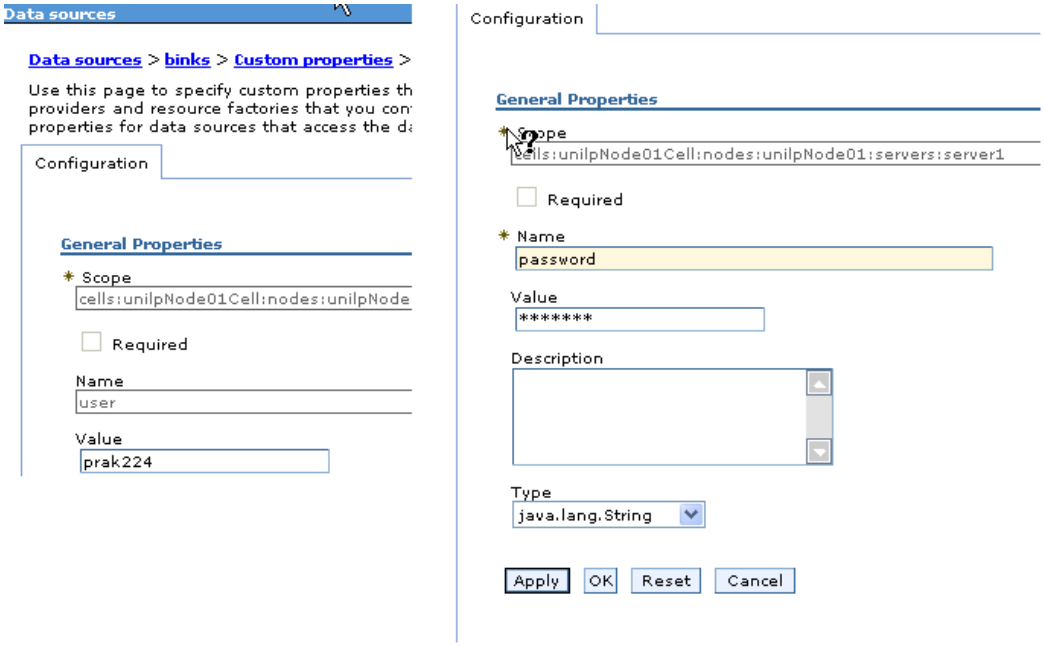


Schritt 3 In der Liste Resources → JDBC → Data Sources ist jetzt ein neuer Eintrag „leia“ zu sehen. Darauf 1k um die Configuration Page zu öffnen.

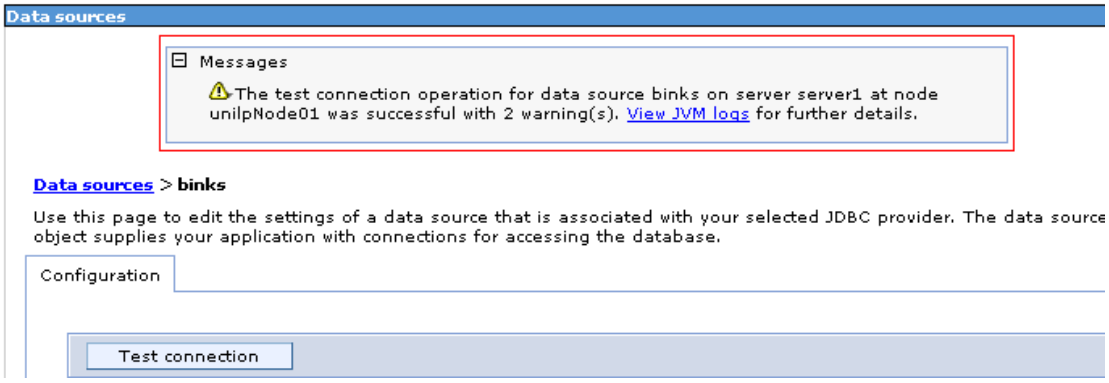


Schritt 4 Um die Verbindung zu der DB2 Data Source herzustellen, muss der Benutzername und das Passwort eingegeben werden. 1k auf Custom properties um die beiden Werte user und password eingeben.

Schritt 5 1k auf New um die Property user hinzuzufügen. In diesem Text verwenden wir den Wert „prak224“. Sie müssen Ihren eigenen Benutzernamen verwenden, den sie von dem DB2 Administrator enthalten haben. 1k auf Apply und Save. Nochmals 1k auf New um das Passwort zu setzen.



Schritt 6 Testen der Verbindung. 1k auf Data sources auf der linken Seite und dann 1k auf leia auf der rechten Seite. 1k auf Test connection. Wenn Sie alles richtig gemacht haben, bekommen Sie eine Nachricht wie in der folgenden Abbildung:



EJB3 Zugriff auf DB2
Entwicklung einer EJB3 Anwendung
für WebSphere 6.1 mit RAD 7.5

Einführung

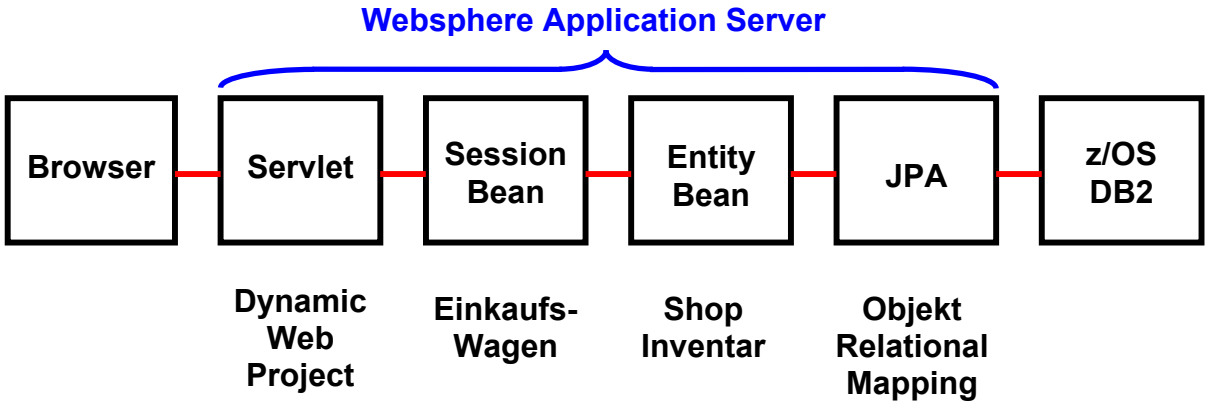
Dieses Tutorial zeigt, wie eine einfache EJB 3 Anwendung mit Rational Application Developver 7.5 (kurz RAD7.5) erstellt und auf einem WebSphere Application Server 6.1 (kurz WAS6.1) deployed werden kann. In diesem Tutorial verwenden wir die Installation aus Tutorial 1: EJB3 Zugriff auf DB2, Installation und Konfiguration. Hier wurde bereits WAS6.1 mit dem EJB3 Feature Pack und RAD7.5 installiert.

Dieses Tutorial präsentiert eine primitive EJB 3.0 Internet Shop Anwendung. Die Anwendung besteht aus einer Session Bean, die einen Einkaufswagen (Shopping Cart) darstellt. Eine Entity Bean stellt die zum Verkauf stehenden Artikel (Items) dar. Die Artikel sind in einer z/OS DB2 Datenbank gespeichert. Die Entity Bean kann mit Hilfe der EJB 3.0 Java Persistence Architektur (JPA) auf die z/OS Architektur zugreifen. Ein Servlet stellt die Verbindung mit einem Browser her.

Das Session Bean Interface für das Servlet hat vier public methods

- **Add an item to the cart**
- **Remove an item from a cart**
- **List the available items in the store**
- **Check out with the selected items**

Einfaches Internet-Shop Modell



Implementierungsübersicht

Die gesamte Anwendung besteht aus einem Enterprise Application Project „Shop“ und zwei Unterprojekten

- **JPA Project (ShopJPA)**
- **Dynamic Web Project (ShopWAR)**

Das JPA-Projekt implementiert u.A. die Schnittstelle zu der DB2 Datenbank; es enthält ein Mapping der Entity Bean Variablen auf eine DB2 Tabelle. Außerdem implementiert es

- **eine Entity Bean (ShopEJB)**
- **eine Session Bean (ShopEJBClient)**
- **eine Schnittstelle zu dem Servlet Container (ShopCart)**

Das Dynamic Web Project implementiert ein Servlet mit dem Namen ShopServlet.

Zu jedem Projekt gehört ein Deployment Deskriptor in Form einer XML-Datei.

Übersicht zur Namensgebung:

Projekt	Implementierung	Name	Deployment Deskriptor	Archivtyp
EAR Project		Shop	application.xml	EAR
JPA Project		ShopJPA	persistence.xml	JAR
	Entity Bean	ShopEJB		
	Entity Bean	ShppEJBClient		
	Interface	ShopCart		
Dyn. Web Project	Servlet	ShopWAR ShopServlet	….xml	WAR

Die Entity Bean, Session Bean und deren Schnittstelle werden in ein Java Archive (JAR) verpackt.
Die Session Bean wird in ein Web Archive (WAR) verpackt.
Beide zusammen werden in ein Enterprise Archive (EAR) verpackt.

Die Entwicklung dieser Komponenten erfolgt unter dem Rational Application Developer (RAD7.5). Das Enterprise Archive (EAR) wird dann auf einem WebSphere Application Server deployed und dort ausgeführt.

Gliederung

Dieses Tutorial ist in 2 Teile gegliedert

- **Teil 1 Konfigurieren von RAD7.5, des WebSphere Application Servers und der DB2 Datenbank**
- **Teil 2 Entwicklung einer Enterprise Anwendung**

1. Konfiguration RAD7.5

1.1 Die Perspektiven für Java EE ändern
1.2 Erstellen eines Servers
1.3 Erstellen einer Datenbank-Verbindung
1.4 Erstellen einer DB2 Tabelle und Einfügen von Daten

2. Entwicklung einer Enterprise Anwendung unter RAD7.5

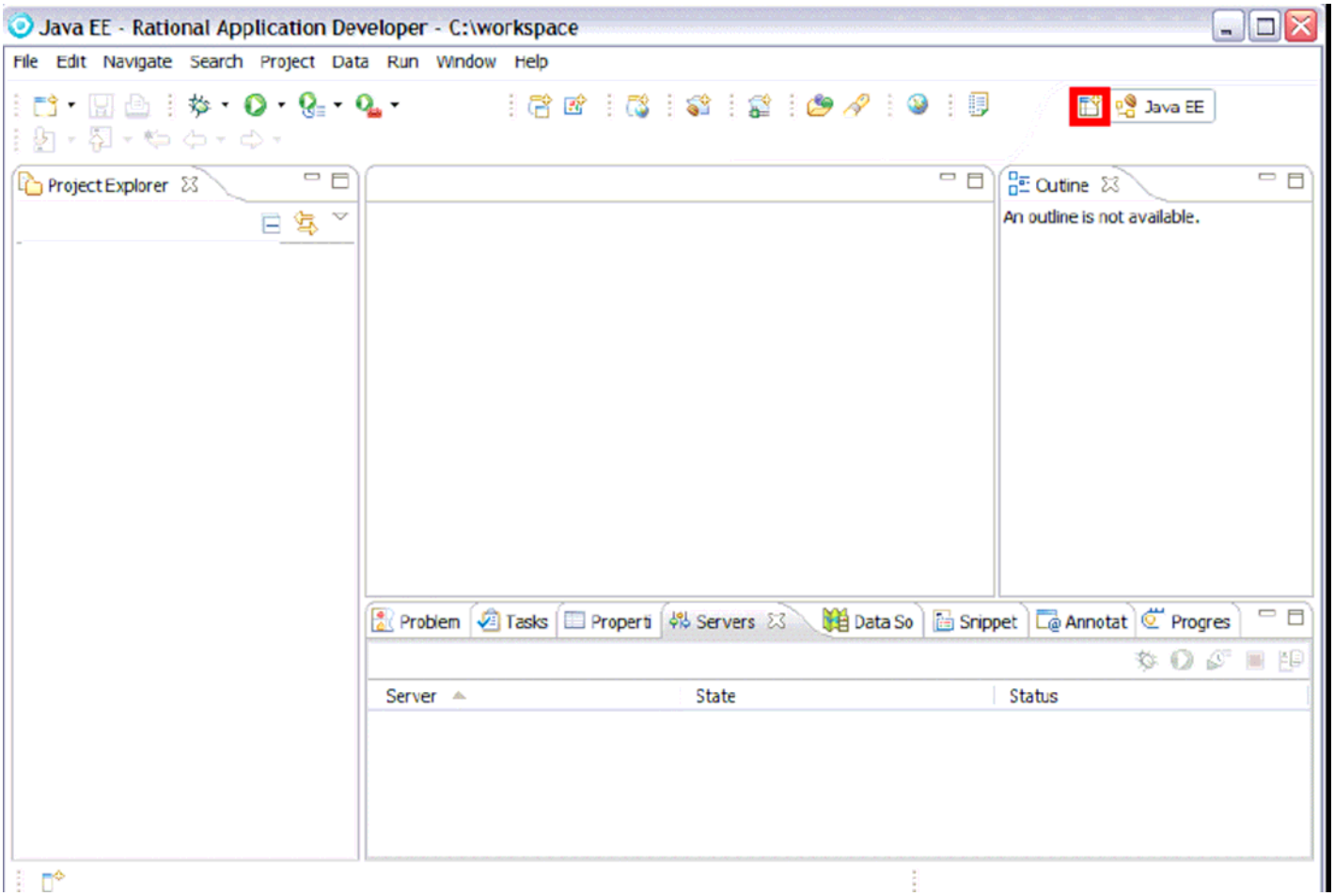
2.1 Erstellen des Enterprise Application Projectes.
2.2 Erstellen eines JPA-Projectes, erstellt auch die Entity Bean
2.3 Mapping der Laufzeitumgebung, bearbeitet Deployment descriptor
2.4 Erstellen eines EJB 3.0 Projektes für die Session Bean
2.5 Hinzufügen des Persistenz-Moduls zu dem Session EJB Modul.
2.6 Erstellen eines Business-Interface zwischen Servlet und Session Bean
2.7 Erstellen einer stateful Session Bean
2.8 Erstellen eines Dynamic Web Projektes und eines Servlets
2.9 Mapping der EJB Logical Reference. Erstellen des ShopWAR Deployment Descriptors
2.10 Das Projekt auf dem Server ausführen

1. Konfiguration von Rational Application Developer 7.5

1.1 Die Perspektive auf Java EE ändern

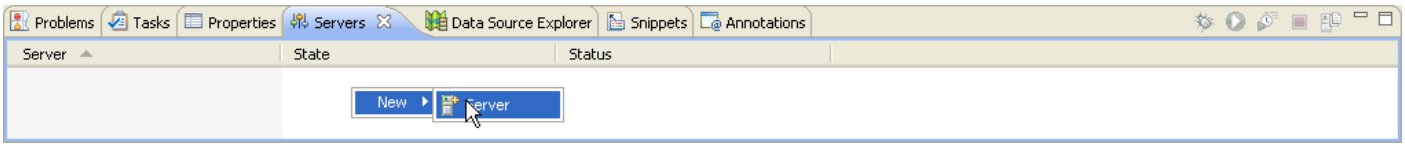
Der RAD ist ein Entwicklungstool was auf Eclipse basiert. Es bietet ein Workspace Management mit seinen Perspektiven, Editoren, Views und Plug-Ins. Wenn Sie bereits mit Eclipse gearbeitet haben, sollte es für Sie einfach sein, Enterprise-Anwendungen mit RAD zu entwerfen und zu entwickeln. Starten Sie es mit:

Start → Programme → IBM Software Delivery Platform → IBM Rational Application Developer 7.5 → IBM Rational Application Developer.

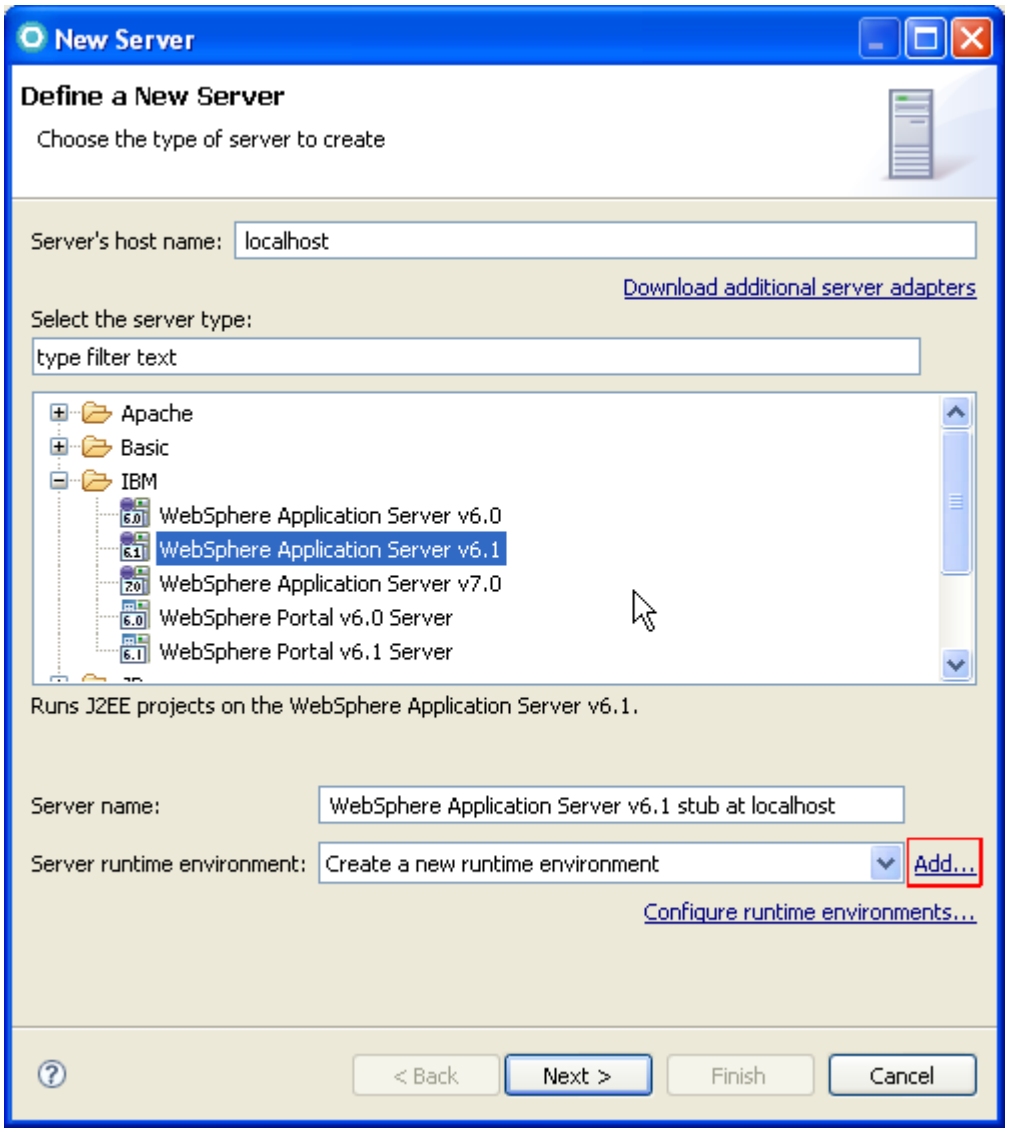


Sie können die Perspektive ändern über Window → Open Perspective oder indem Sie auf das Open Perspective Icon klicken (rot umrandet).

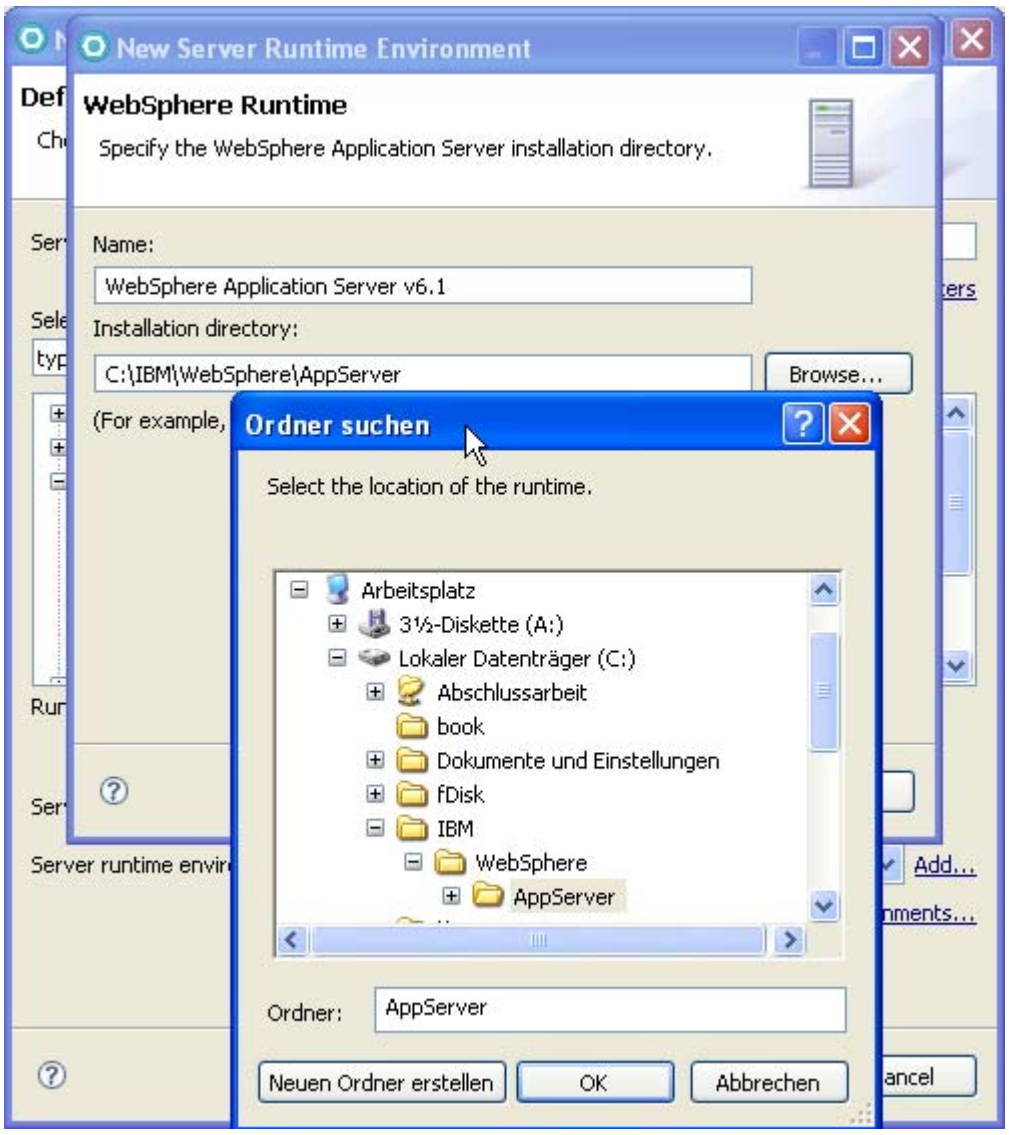
1.2 Erstellen eines Servers



Einer der wichtigsten Views der mit der JEE Perspektive verknüpft ist, ist der Servers View. Dort haben sie eine Übersicht Ihrer konfigurierten Server. 1kr irgendwo in dem Server View und dann 1k new → Server.



Wählen Sie WebSphere Application v6.1 Server und klicken auf Add um die Server-Laufzeitumgebung (Runtime Environment) zu erstellen.

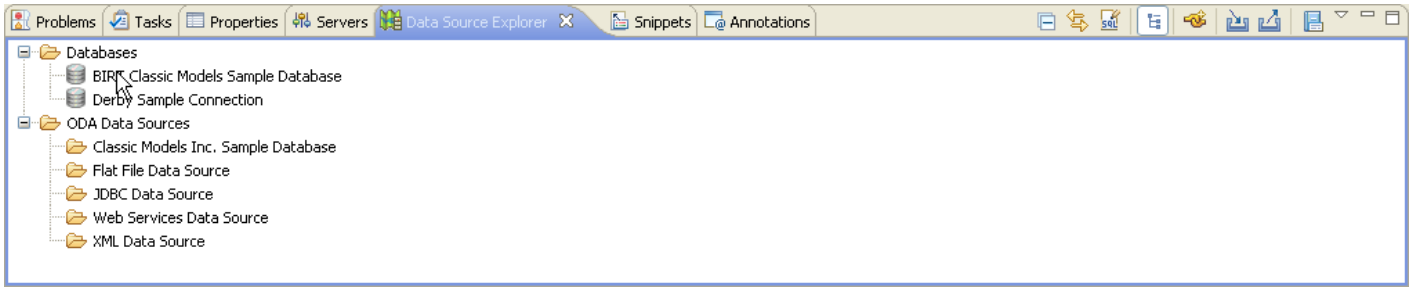


Navigieren Sie zu dem WAS 6.1 Installationsverzeichnis c:\IBM\WebSphere\AppServer und bestätigen dies.

Aufgrund der Sicherheitseinstellungen des Servers, müssen wir die Benutzer-ID und Passwort eingeben. In diesem Beispiel sind diese beide „unilp“. Klicken Sie auf Finish.

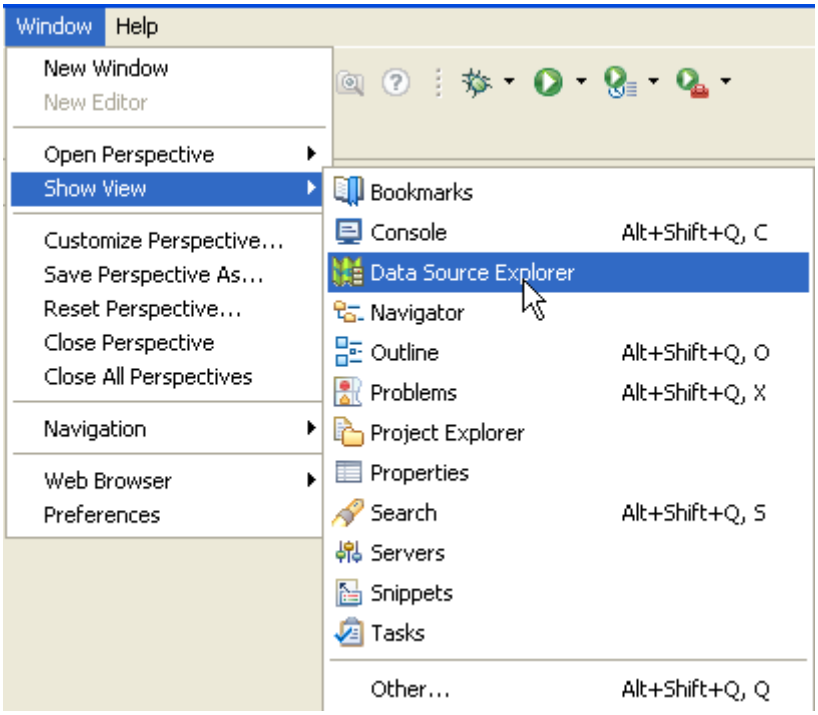
Jetzt existiert ein neuer Server-Eintrag in dem Servers-View.

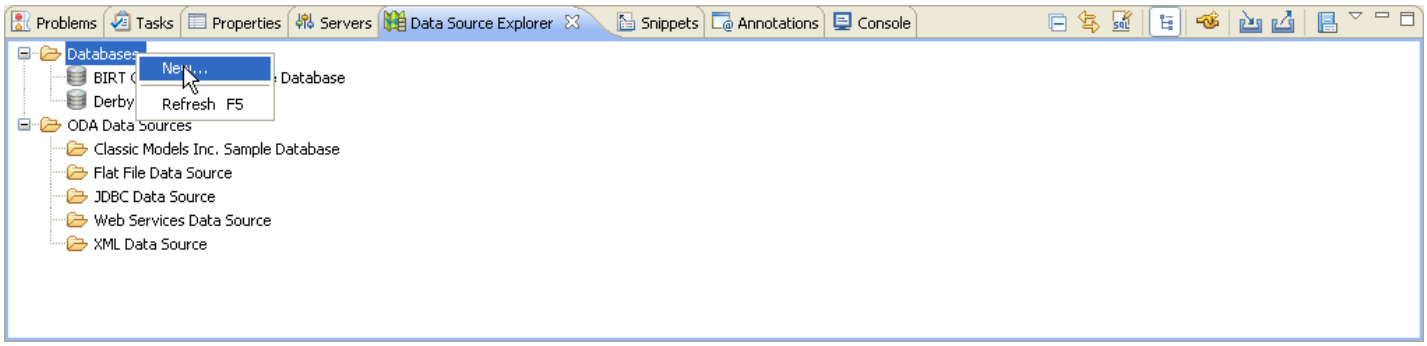
1.3 Erstellen der Datenbank-Verbindung



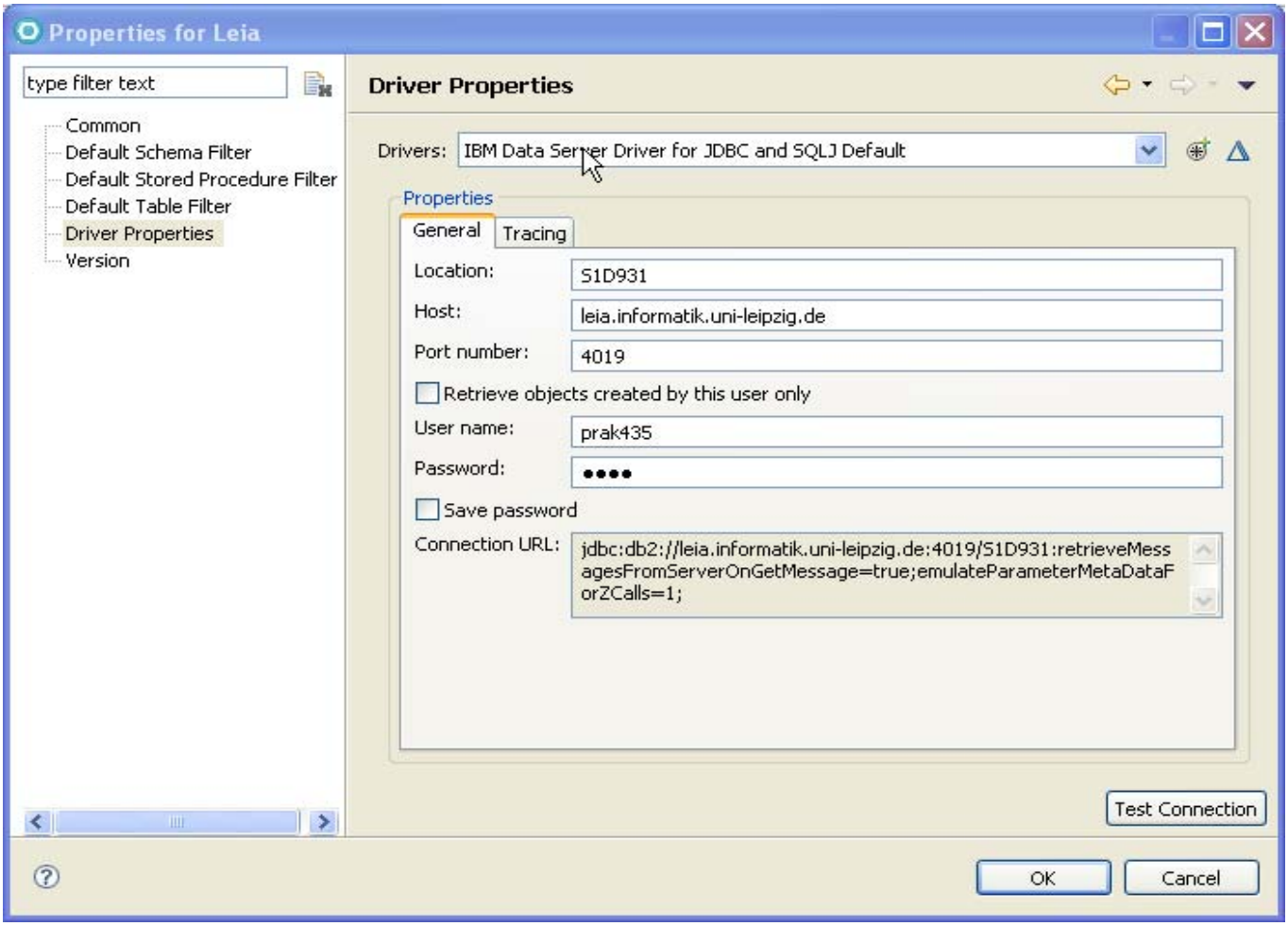
Ein weiterer wichtiger View der Java EE Perspektive ist der Data Source Explorer. Es sollte in der gleichen Leiste zu finden sein, wie der Server View.

Wenn der View nicht vorhanden ist, 1k Window->Show View-> Data Source Explorer um ihn zu öffnen.





1kr auf Databases und New wählen

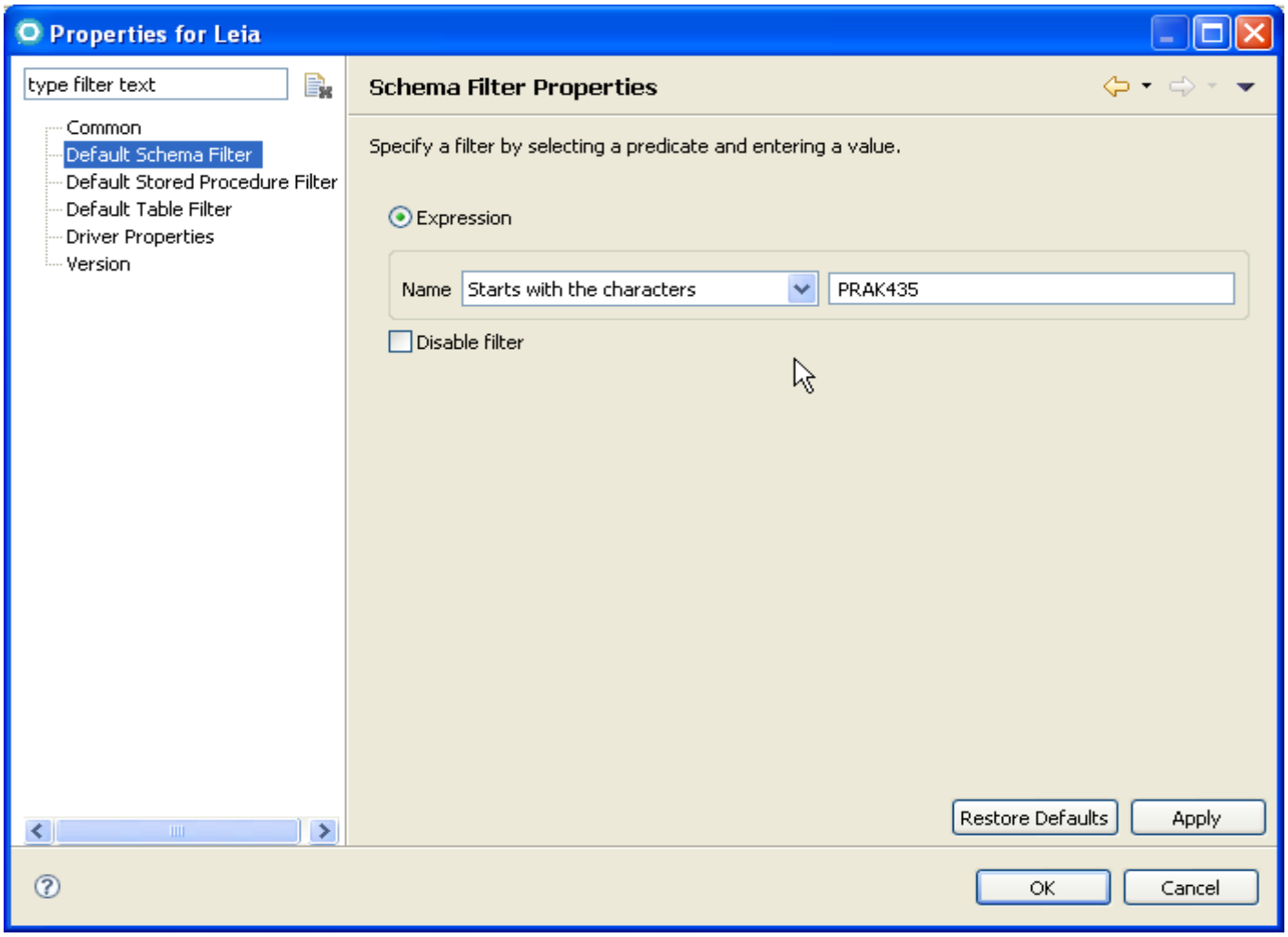


Wählen Sie nun den *DB2 for z/OS*-Datenbank-Manager.

In diesem Tutorial wird also DB2 unter z/OS verwendet. Der Name der Datenbank ist S1D931. S1D931 als Location, leia.informatik.uni-leipzig.de als Host, 4019 als Port eingeben.

Geben Sie nun noch Ihren Benutzernamen und Passwort ein. 1k auf Finish.

Jetzt wurde eine neue Datenbank-Verbindung mit dem Namen S1D931 erstellt und erscheint in dem View. Unter der Verbindung finden Sie viele Schemata, da diese Datenbank von vielen Benutzern verwendet wird. 1kr auf diese Verbindung, und klicken Sie auf Properties.



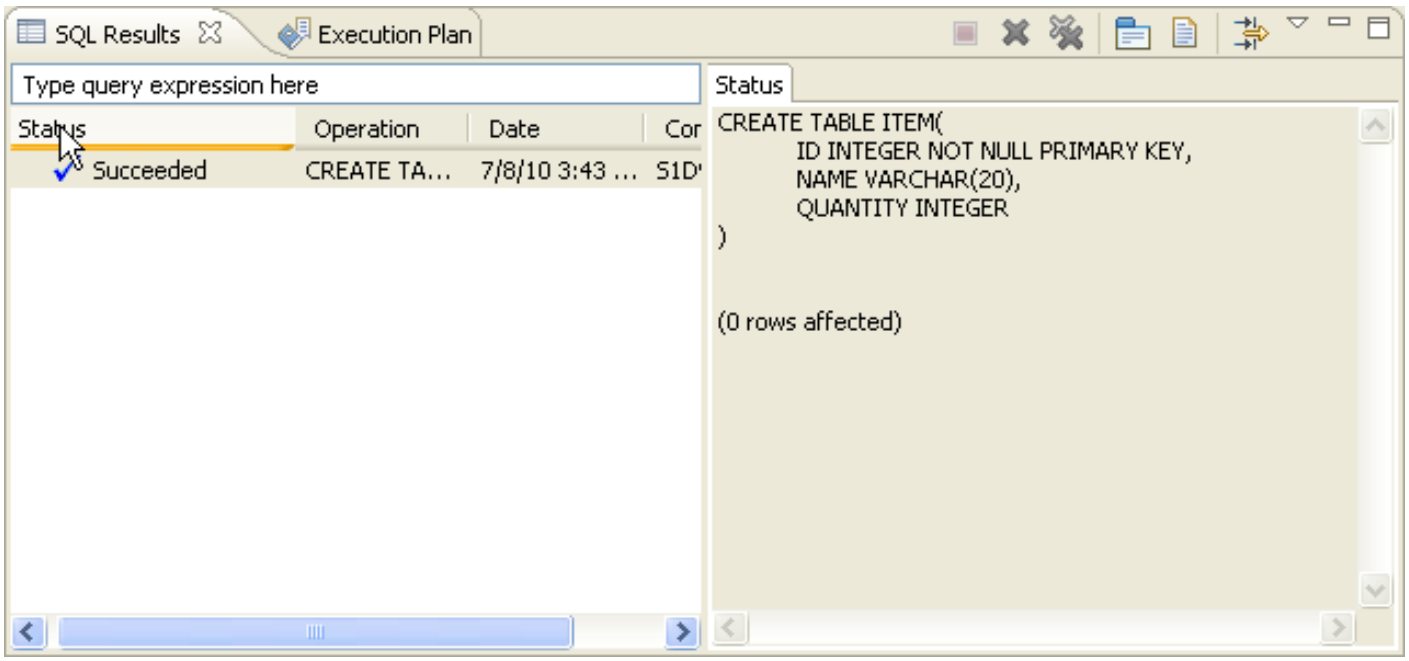
In dem nächsten Fenster wählen Sie Default Schema Filter auf der linken Seite, deaktivieren Sie Disable Filter und geben Sie Ihren Benutzernamen (case sensitive) ein. Klicken Sie auf OK. Eventuell müssen Sie den Connection View aktualisieren, dazu F5. Es gibt nur das Schema links.

1.4 Erstellen einer DB2 Tabelle und Einfügen von Daten

1kr auf S1D931, wählen Sie Open SQL Scrapbook. In den Textbereich geben folgenden Code ein, um eine neue Tabelle zu erstellen. Fügen Sie diese Daten ein:

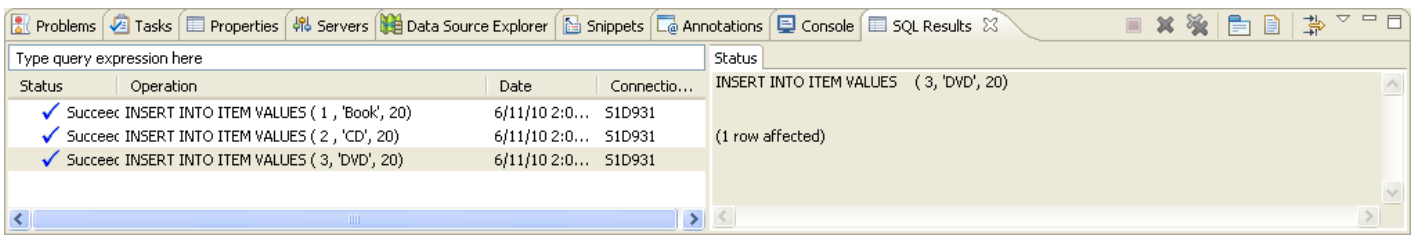
```
CREATE TABLE ITEM (
        ID INTEGER NOT NULL PRIMARY KEY,
        NAME VARCHAR (20) WITH DEFAULT NULL,
        QUANTITY INTEGER WITH DEFAULT NULL
)
```

1kr irgendwo im Text, Wählen Sie Execute all. Wenn die Tabelle erfolgreich erstellt wurde, wird der SQL Results View die folgende Meldung anzeigen:



Nun geben wir einige Daten in die neu angelegte Tabelle ein. Bitte den folgenden Code in den Text Bereich eingeben und ihn dann ausführen.

```
INSERT INTO ITEM VALUES ( 1001 , 'Book', 20);
INSERT INTO ITEM VALUES ( 1002 , 'CD', 20);
INSERT INTO ITEM VALUES ( 1003, 'DVD', 20)
```



Der SQL Results View wird Ihnen anzeigen, ob der SQL Code erfolgreich ausgeführt wurde.

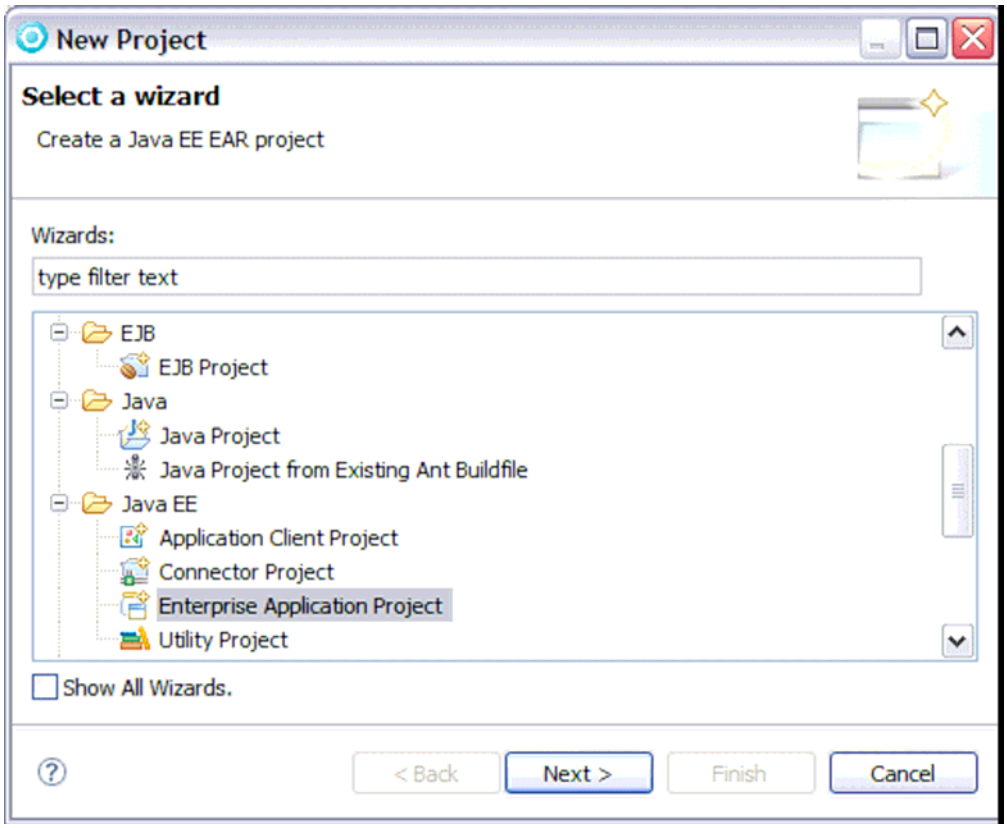
2. Entwicklung einer Enterprise Anwendung

EJB3.0 verbessert die EJB2.x Spezifikation. In EJB2.x sind alle Ressourcen miteinander durch JNDI verbunden. EJB3 vereinfacht das Modell. Die meisten Ressourcen können mit einem tag @ definiert werden.

Ein typisches EJB3 Projekt besteht aus 4 Teilen:

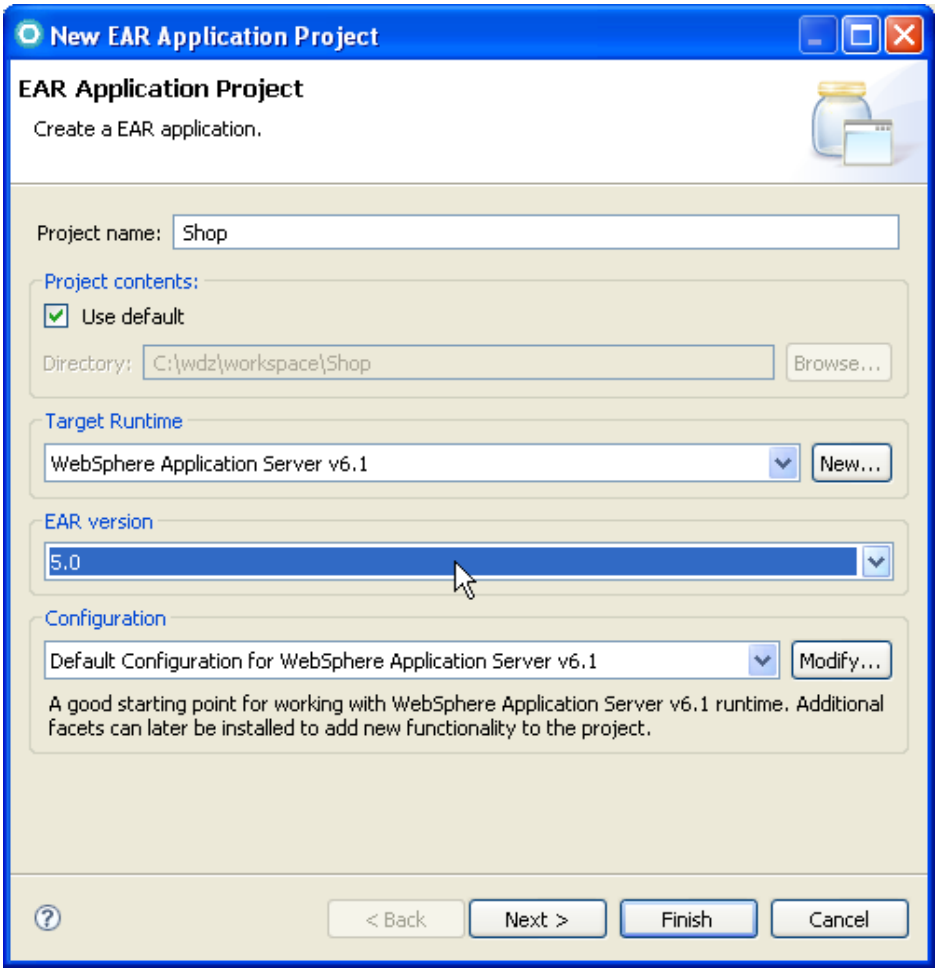
- **JPA Projekt, behält die in einer Datenbank gespeicherten Daten**
- **Client-Projekt, definiert wie ein Remote-Benutzer auf die Geschäftslogik zugreifen kann**
- **EJB-Projekt, implementiert die Schnittstellen, die in dem Client-Projekt definiert wurden**
- **Dynamic Web Project**

2.1 Erstellen des Enterprise Application-Projektes.



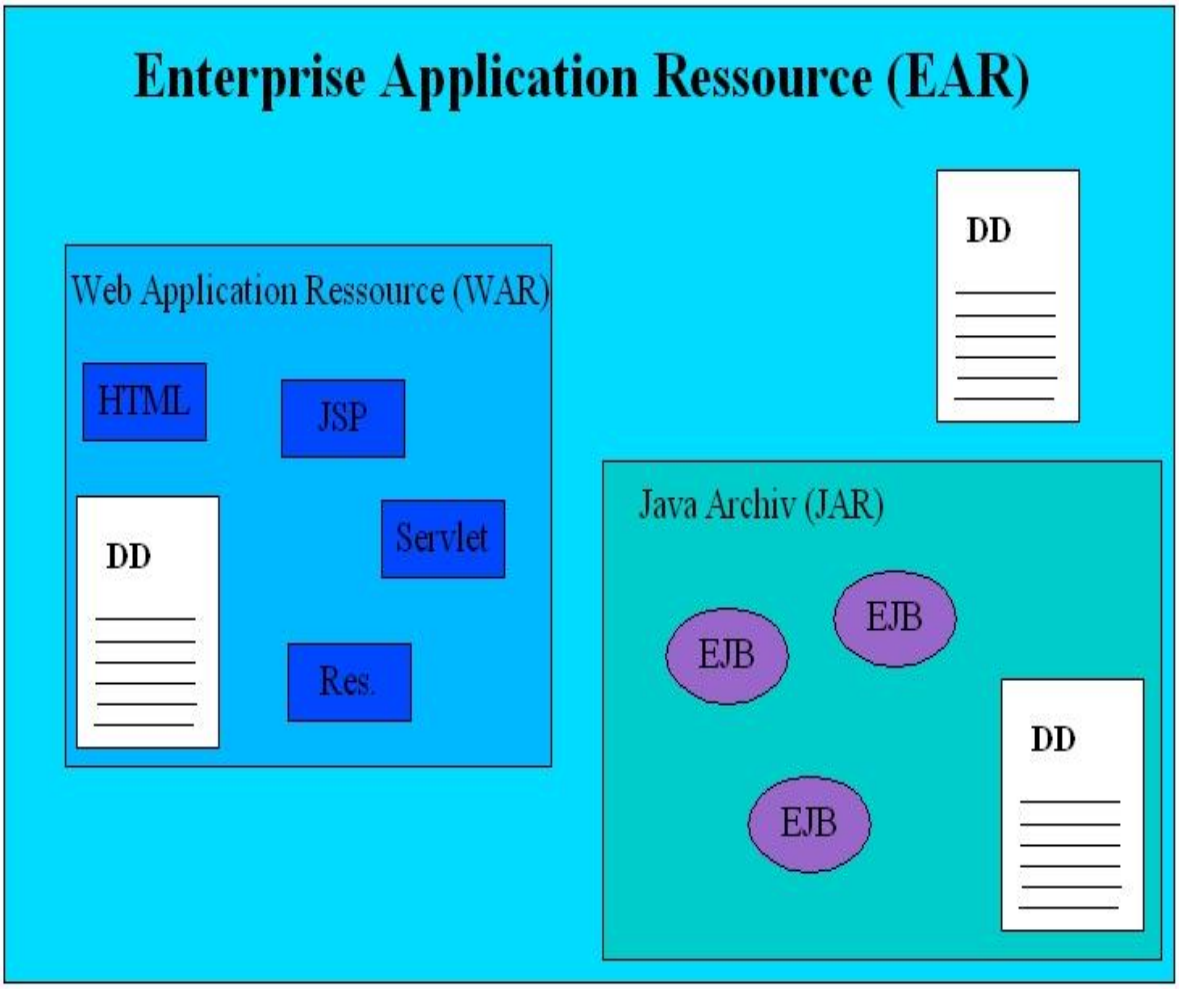
Das Enterprise Application-Projekt ist im Prinzip eine EAR (Enterprise Application Resource). Sie müssen später ein Dynamic Web Project dem EAR Project hinzufügen, und das Ganze dann exportieren.

Wählen Sie File → New → Project…
In dem New Project Wizard, wählen Sie Java EE → Enterprise Application Project



In dem EAR Application Project Panel, *Shop* als Projektnamen eingeben. Für Target Runtime *WebSphere Application Server v6.1* auswählen. Für EAR version *5.0* auswählen. Wähle "Default Configuration for WebSphere Application Server v6.1" als Configuration.

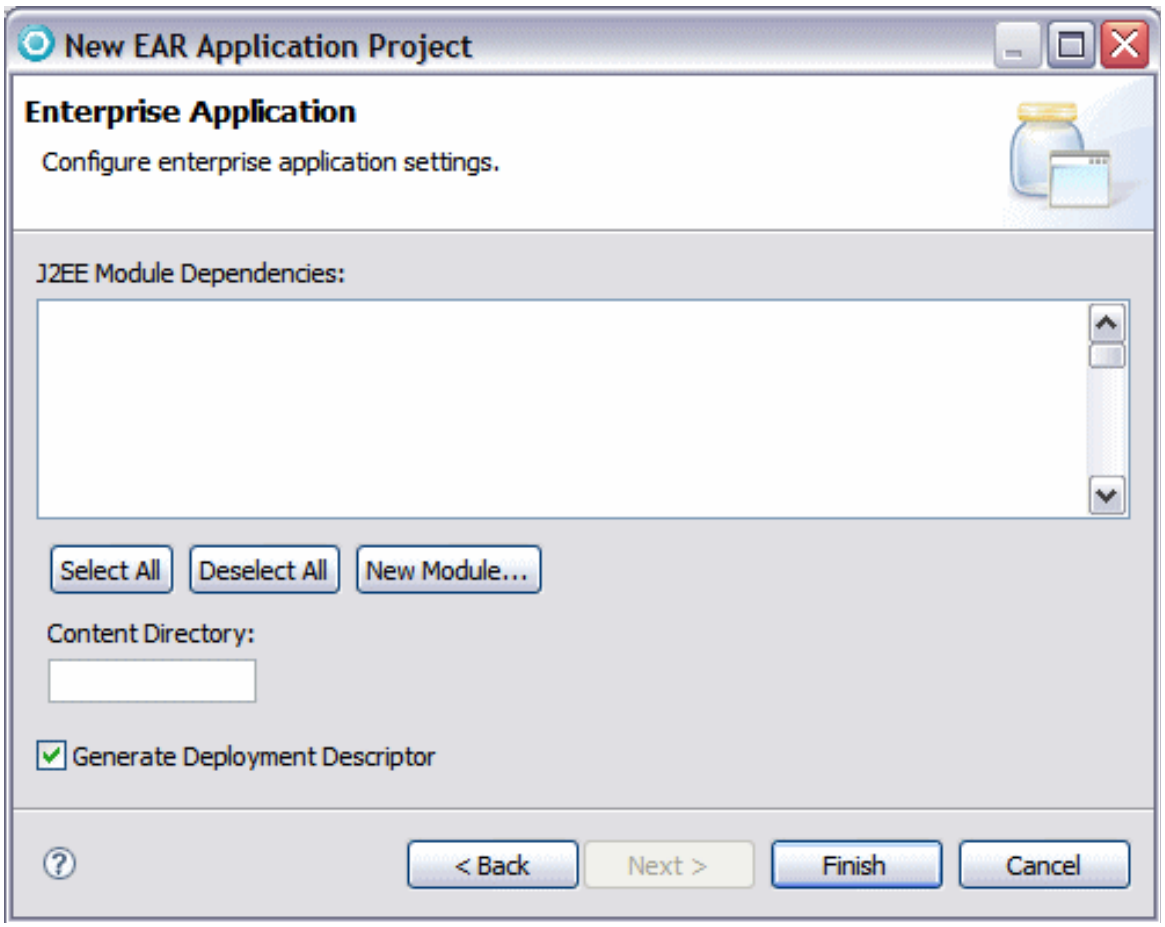
1k auf Next.



Die hier gezeigte Abbildung ist eine Kopie aus Band 2, „z/OS Internet Integration“, Abb. 17.2.7 .

Die EAR-Datei (application.xml) enthält nur den Namen der Enterprise-Anwendung und deren Bestandteile.

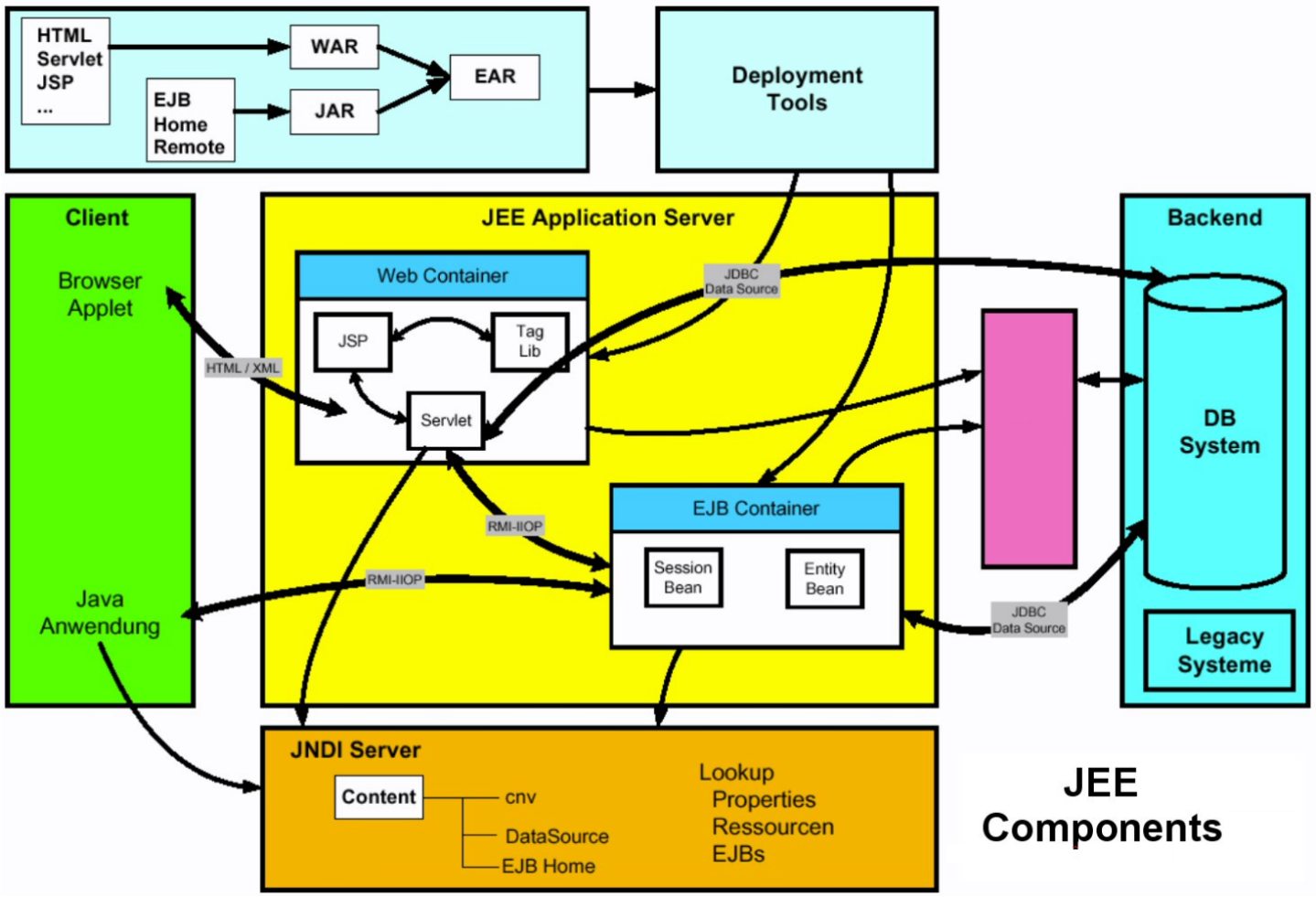
DD = Deployment Descriptor



Wählen Sie kein existierendes Modul (falls welche vorhanden sind). Generate Deployment Descriptor auswählen. 1k auf Finish.

Wenn Sie nicht bereits in der Java EE Perspektive sind, werden Sie aufgefordert, zu wechseln. Klicken Sie auf Yes.

Wundern Sie sich nicht über die Fehler (das rote Kreuz-Symbol im Shop-Projekt). Es ist sichtbar, weil die Deployment-Deskriptor-Datei (application.xml) keine Module enthält.



Die hier gezeigte Abbildung ist eine Kopie aus Band 2, „z/OS Internet Integration“, Abb. 17.2.8.Die moderne Bezeichnung für Web-Container ist „Dynamic Web Project“. Der Web-Container enthält auch statische HTML Seiten, welche Bestandteil der Anwendung sind.

JSP Tag Libraries bieten die Möglichkeit, JSPs frei von Java-Code zu halten. Dabei übernehmen Tags Aufgaben der Darstellungslogik wie bspw. Formatierungen, Iterationen über Arrays und Listen oder ähnliche Dinge. Durch die Einführung der Tag Libraries entstanden an vielen Stellen immer wieder die gleichen Tags. Daher lag es nahe, diese Bemühungen zu bündeln und eine einheitliche Bibliothek für die am häufigsten wiederkehrenden Aufgaben zu schaffen. Hierbei entstand der „JSP Standard Tag Library“ (JSTL)-Standard.

2.2 Erstellen eines JPA-Projektes

Entity Beans des EJB2.x Standards wurden in EJB 3.0 komplett überholt. Entity Beans (auch Entities/Entitäten genannt) sind reine Java-Objekte, die mit „new“ alloziiert werden können. Sie werden auf persistenten Speicher aufgesetzt (attached/detached/reattached). Entities sind nicht per remote abrufbar. (Die Definition eines Remote RMI-Objektes beinhaltet die Angabe einer Remote-Schnittstelle für das Objekt und die Erstellung einer Klasse, die diese Schnittstelle implementiert.) Auf Entities muss durch den neuen javax.persistence.EntityManager-Service zugegriffen werden. Der javax.persistence.EntityManager hat seine Wurzeln in dem populären OpenSource-Projekt „Hibernate“.

Bei den Entities handelt es sich um ganz normale Java Klassen (POJOs), die kein spezielles Interface implementieren oder von einer vorgegebenen Oberklasse erben müssen. Lediglich ein Default-Konstruktor muss vorhanden sein. Dank JPA-Annotationen kann man erkennen, dass die Objekte persistiert werden.

Möchte man EJB3-konform arbeiten, so sollten die Annotationen dort angebracht werden, wo auf die Properties zugegriffen wird, spezifisch an den Attributen für *field-access* und an den Getter-Methoden für *property-access*. Dies sollte konsistent gemacht werden und die beiden Formen sollten nicht gemischt werden. (Zwischen den beiden Formen wird über die Position der @Id-Annotation unterschieden.)

Die Annotation der Attribute hat dabei den Vorteil, dass schnell erkennbar ist, was wie persistiert wird. Dies kann jedoch bei einigen JPA-Providern zu Performance-Problemen führen. Auf der anderen Seite führen Getter-Methoden, die nicht zu persistierende Werte liefern und nicht explizit von der Persistierung ausgenommen werden zu möglicherweise schwer zu findenden Programmfehlern.

JPA, die Java Persistence API bietet im Vergleich zu JDBC ein erhöhtes Abstraktionsniveau und ermöglicht damit in kurzer Zeit Anwendungen zu entwickeln, in denen Daten dauerhaft gespeichert werden müssen.

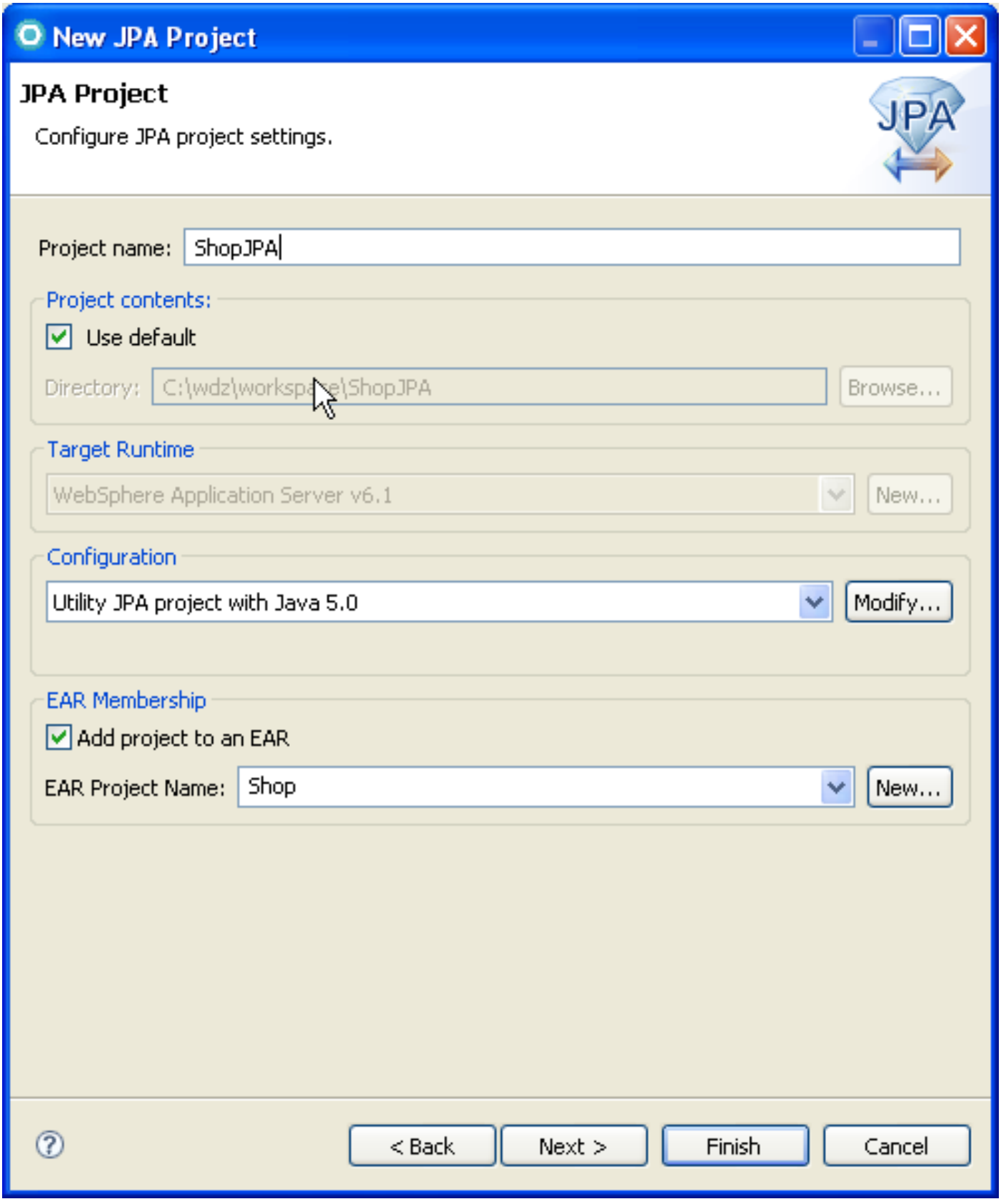
Bei dem Umgang mit Datenbanken gibt es schon sehr lange JDBC (Java Database Connectivity) als einheitliche API um die Verbindung zu Datenbanken herzustellen. Auch hier handelt es sich bereits um eine einheitliche Schnittstelle, jedoch wird darüber lediglich festgelegt, wie Verbindungen aufgebaut werden, wie eine Abfrage eingeleitet wird und wie Transaktionen manuell gesteuert werden können.

Damit bleiben zwei Probleme: Der SQL-Code muss von Hand erstellt werden, was gerade bei CRUD (Create, Update, Delete)-Anwendungen viel Produktivität kostet, und zudem ist der SQL-Code in der Regel herstellerabhängig. Ein Wechsel des Herstellers bedeutet damit, das Programm an vielen Stellen ändern zu müssen. Auch sind von Hand erstellte Abfragen nicht immer optimal: Lazy Loading von Entitäten, Caching Strategien und Transaktionssteuerung können sehr aufwendig werden. (Lazy Loading ist ein Design-Muster welches die Initialisierung eines Objekts bis zu dem Punkt verzögert, an dem die Initialisierung benötigt wird.) Dazu kommen der relativ hohe Wartungsaufwand und die Gefahr von Fehlern, die erst zur Laufzeit entdeckt werden.

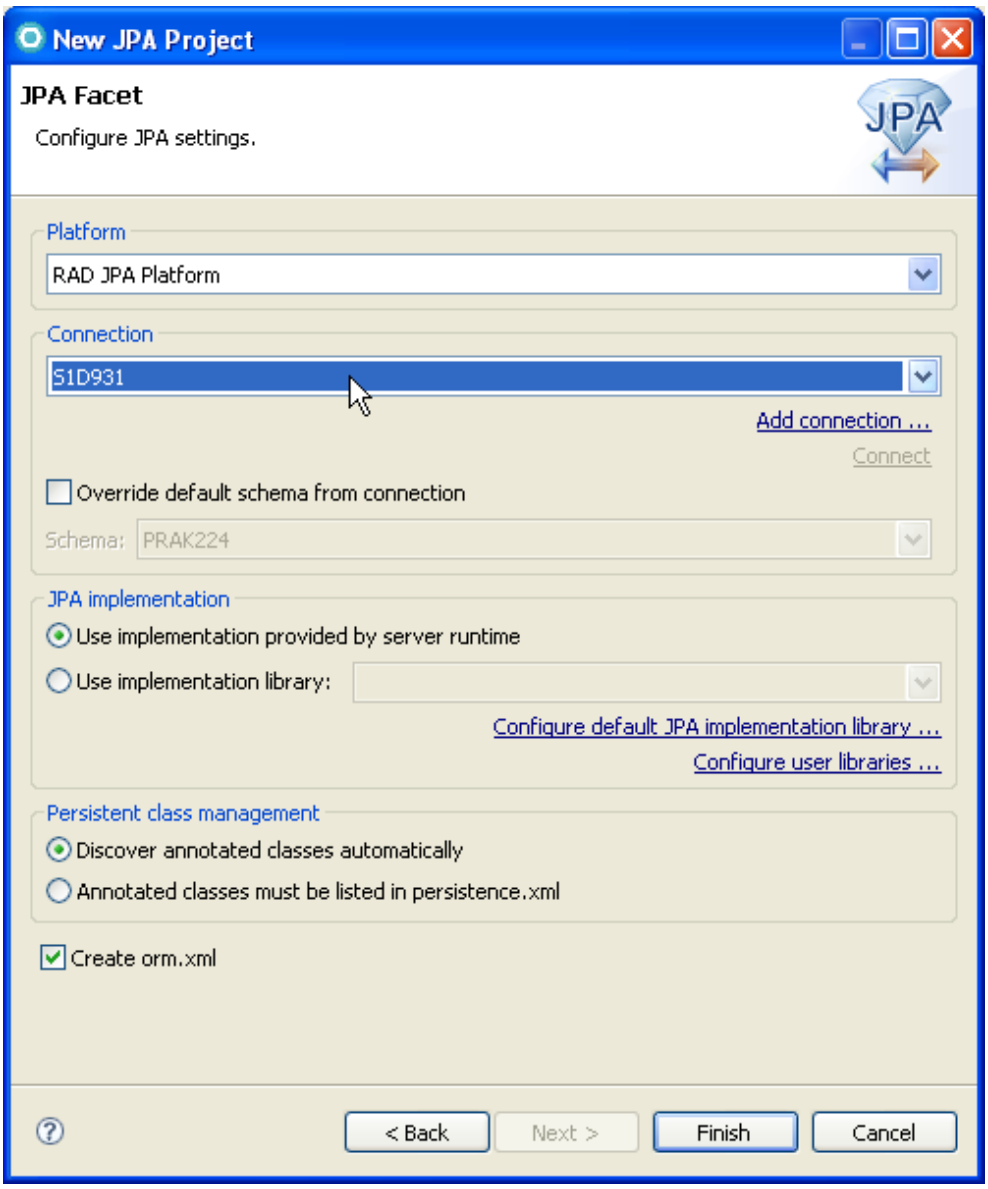
JPA selber ist lediglich eine API – diese wird durch einen sogenannten JPA Provider implementiert. Hibernate, WebSphere und EclipseLink sind Beispiele für JPA Provider, die teilweise als OpenSource zur Verfügung stehen und auch kommerziell verwendet werden können.

http://www.jug-muenster.de/jpa-tutorial-jpa-einrichtung-829

Wählen Sie File → New → Project, dann Wähle JPA → JPA Project. Klick Next.



ShopJPA als Projektnamen eingeben. Wähle Add project to an EAR, und wähle Shop als EAR Project Name. Beachten Sie die Konfiguration: Utility JPA project with Java 5.0. 1k auf Next.

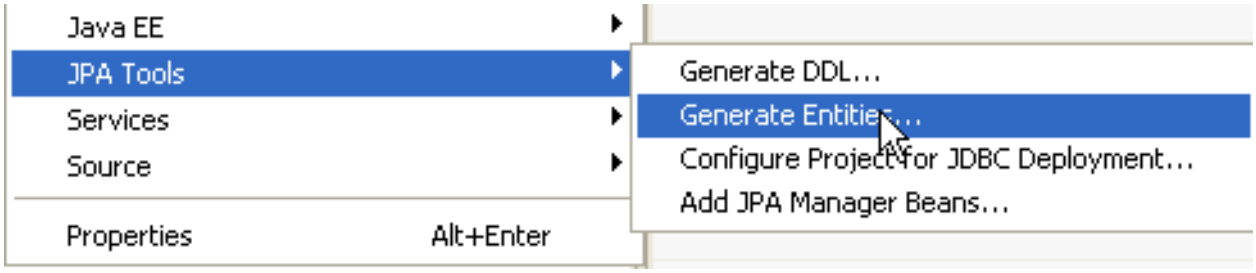


Das JPA Facet-Panel ist das wichtigste Panel in dem JPA-Projekt-Assistenten. Wir geben an, wie ein JPA-Provider die Entitäten verwaltet und ob sie in der Datei persistence.xml aufgeführt werden oder nicht.

Woher weiß der Server, welche Datenbank für das save/update/query der Entity-Objekte benutzt werden soll? Wie konfigurieren wir das zugrunde liegende Objekt-relationale Mapping? In der Datei „persistence.xml“ (normalerweise im META-INF Verzeichnis) befindet sich die Datenbankkonfiguration für den JPA-Provider. Die persistence.xml-Datei gibt Ihnen völlige Flexibilität, um den JPA-EntityManager zu konfigurieren.

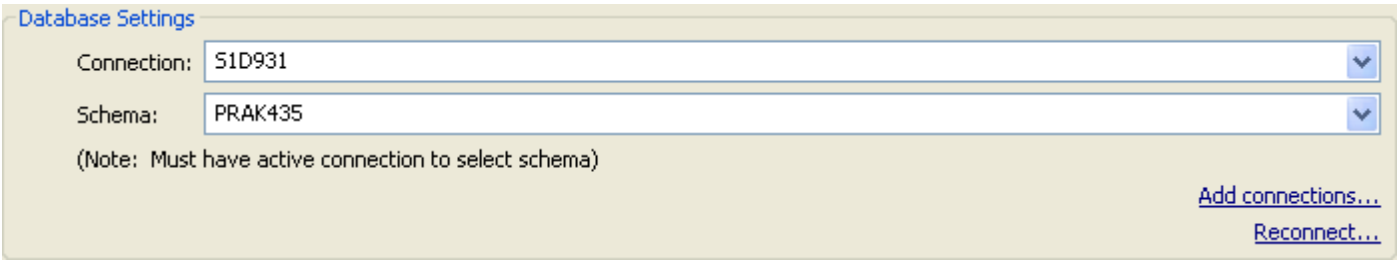
Die persistence.xml-Datei ist eine Standard JPA Konfigurationsdatei. Sie muss in dem META-INF Verzeichnis, und dort innerhalb der JAR-Datei, welche die Entity Beans enthält, vorhanden sein. Die persistence.xml-Datei muss eine Persistenz-Einheit mit einem eindeutigen Namen in dem aktuellen Classloader definieren. Das Provider-Attribut spezifiziert die zugrunde liegende Implementierung des JPA-EntityManagers.

Für Connection wählen Sie "S1D931". Dies ist die DB2 Verbindung, die wir in Teil 1 erstellt haben. 1k auf Finish.

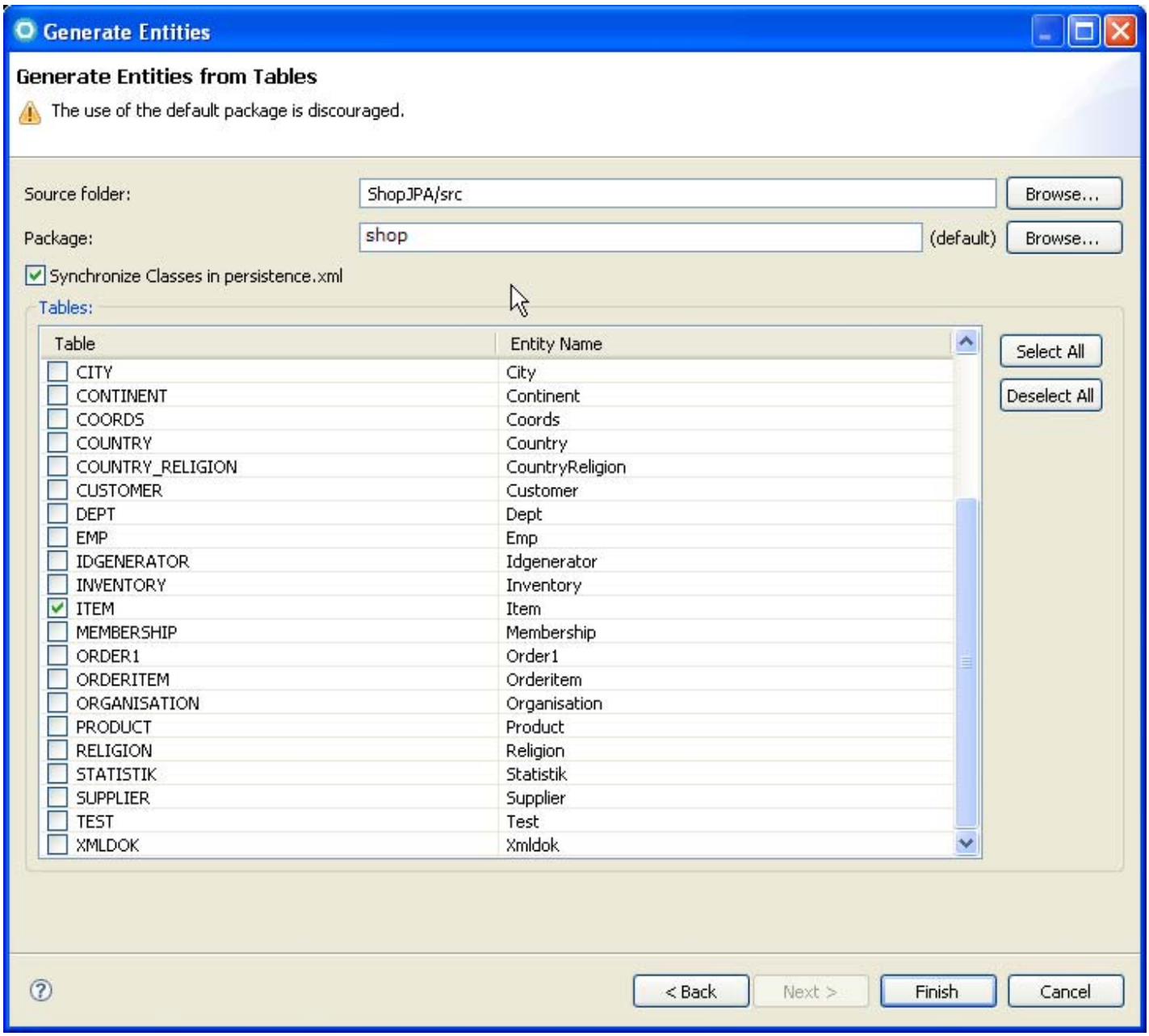


Erstellen einer JPA Entity (Entity Bean):

1kr auf ShopJPA-Project. Wählen Sie JPA Tools → Generate Entities…



Selektiere “S1D931” als Connection und ihre PRAK[xxx] als Schema. 1k auf Next.



Shop als Package eingeben. Item auswählen in dem Generate Entities Panel. 1k auf Finish.

Wir erstellen hier ganz automatisch eine Entity Bean, ohne eine spezifische Eingabe. Der Code ist in der nächsten Abbildung wiedergegeben. Wir könnten hier die vorgefertigte Bean „Item“ unverändert übernehmen. In der Praxis würde vermutlich zusätzlicher Code erforderlich sein.

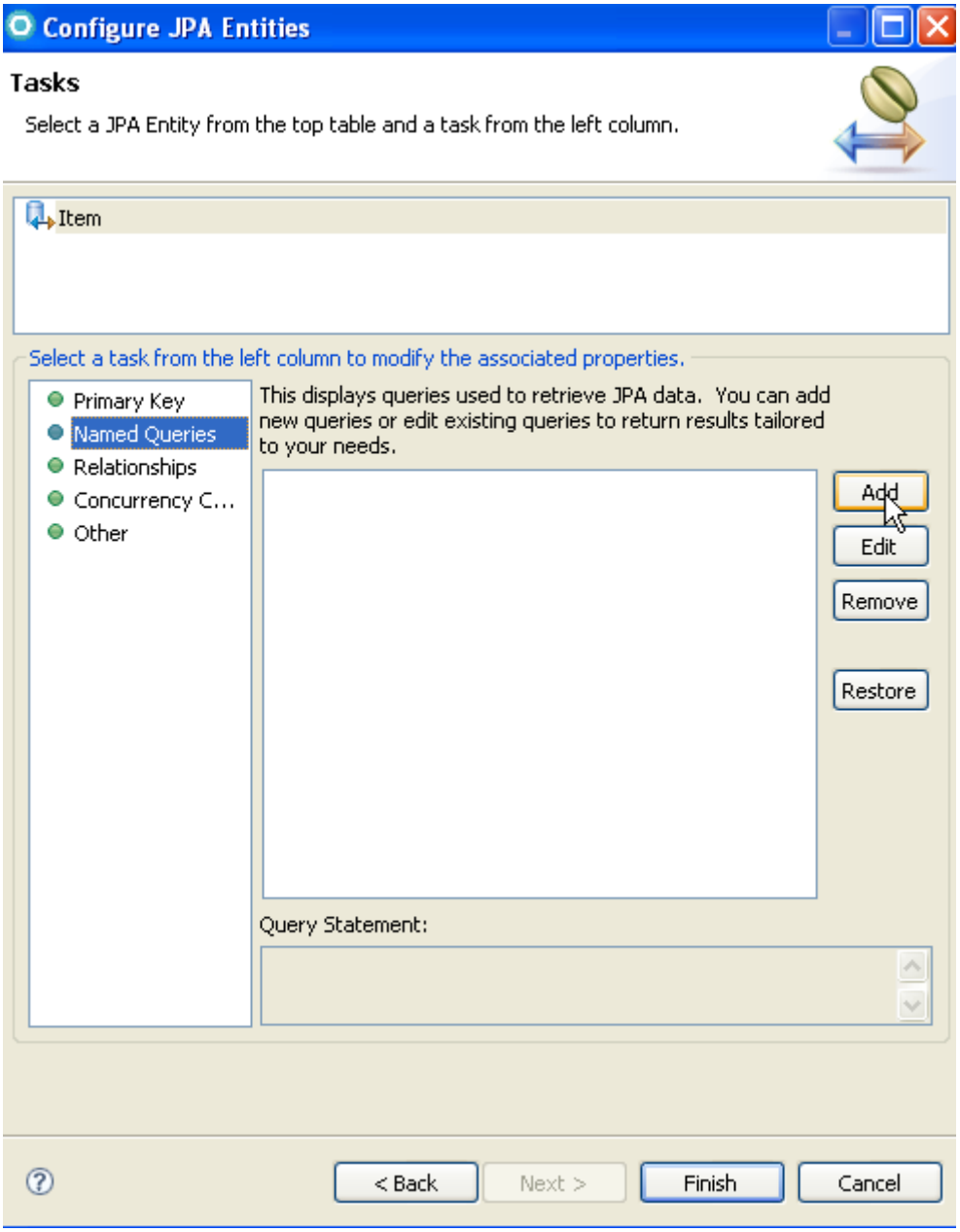
Dies ist der Quellcode (automatisch generiert):

```
package shop;

import java.io.Serializable;
import javax.persistence.Entity;
import javax.persistence.Id;

@Entity /* define an Entity Bean */
public class Item implements Serializable {
   @Id /* define the key for the bean */

   private int id;

   private String name;

   private int quantity;

   private static final long serialVersionUID = 1L;

   public Item() {  super();     }

   public int getId() {  return this.id;  }

   public void setId(int id) {  this.id = id;  }

   public String getName() {  return this.name; }

   public void setName(String name) { this.name = name;   }

   public int getQuantity() { return this.quantity; }

   public void setQuantity(int quantity) { this.quantity = quantity; }

}
```

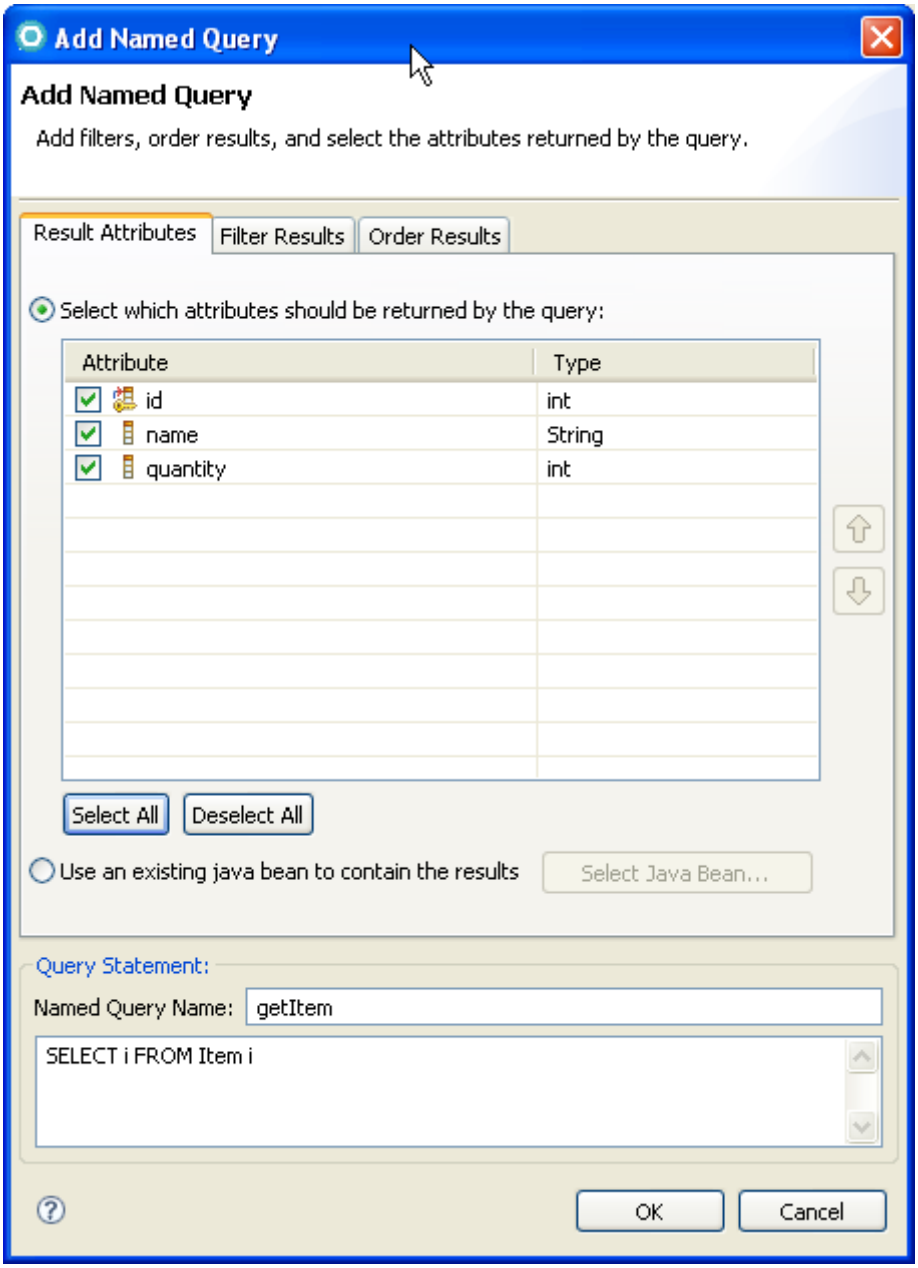
Die drei Variablen der Entity Bean

Wir fügen 2 Queries hinzu. Eine hat keine Parameter. Die andere hat den Parameter „id“.

1kr auf Item dargestellt in Project Explorer. Selektiere JPA Tools->Configure JPA Entities. Item aus der Table-Liste auswählen und 1k auf Next.



Named Queries auswählen und 1k auf Add.



1k auf OK und dann nochmal 1k auf Add.

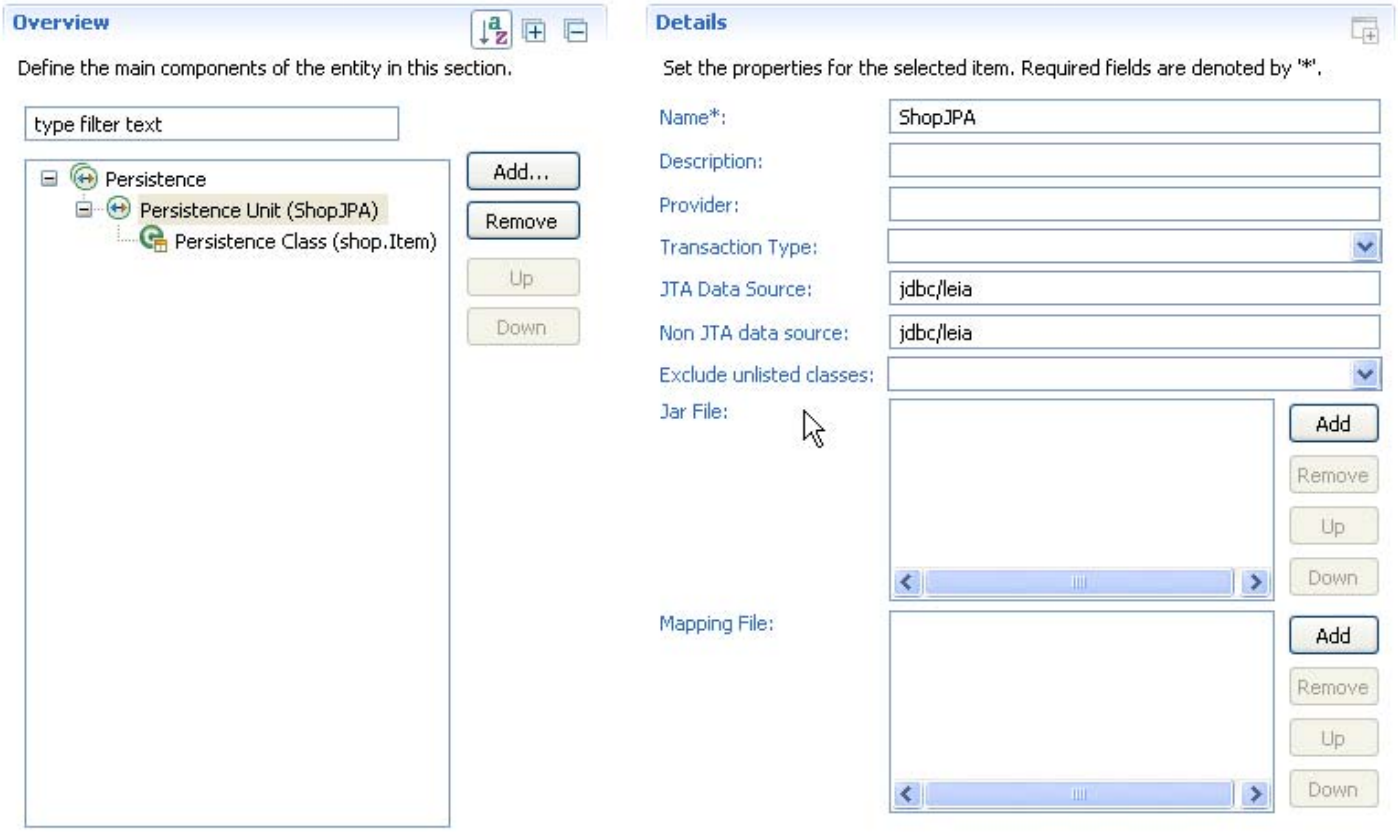
Ausgewählt sind wieder alle drei Attribute. „getItemByID“ als Named Query Name und „Select i FROM Item i where i.id =:id “ in den Text Bereich des Query Statement Panel eingeben.
1k auf OK und Finish.

Öffnen Sie den Quellcode von Item.java. Er wurde geändert. Einige neuen Zeilen Code sind in grün hier dargestellt:

```
......
@Entity
@NamedQueries(    {
   @NamedQuery(name="getItem", query = "SELECT i FROM Item i"),
   @NamedQuery(name="getItemByID", query = "SELECT i FROM Item I
      WHERE i.id = :id")
      })
public class Item implements Serializable {
......
```

2.3 Mapping der Laufzeitumgebung

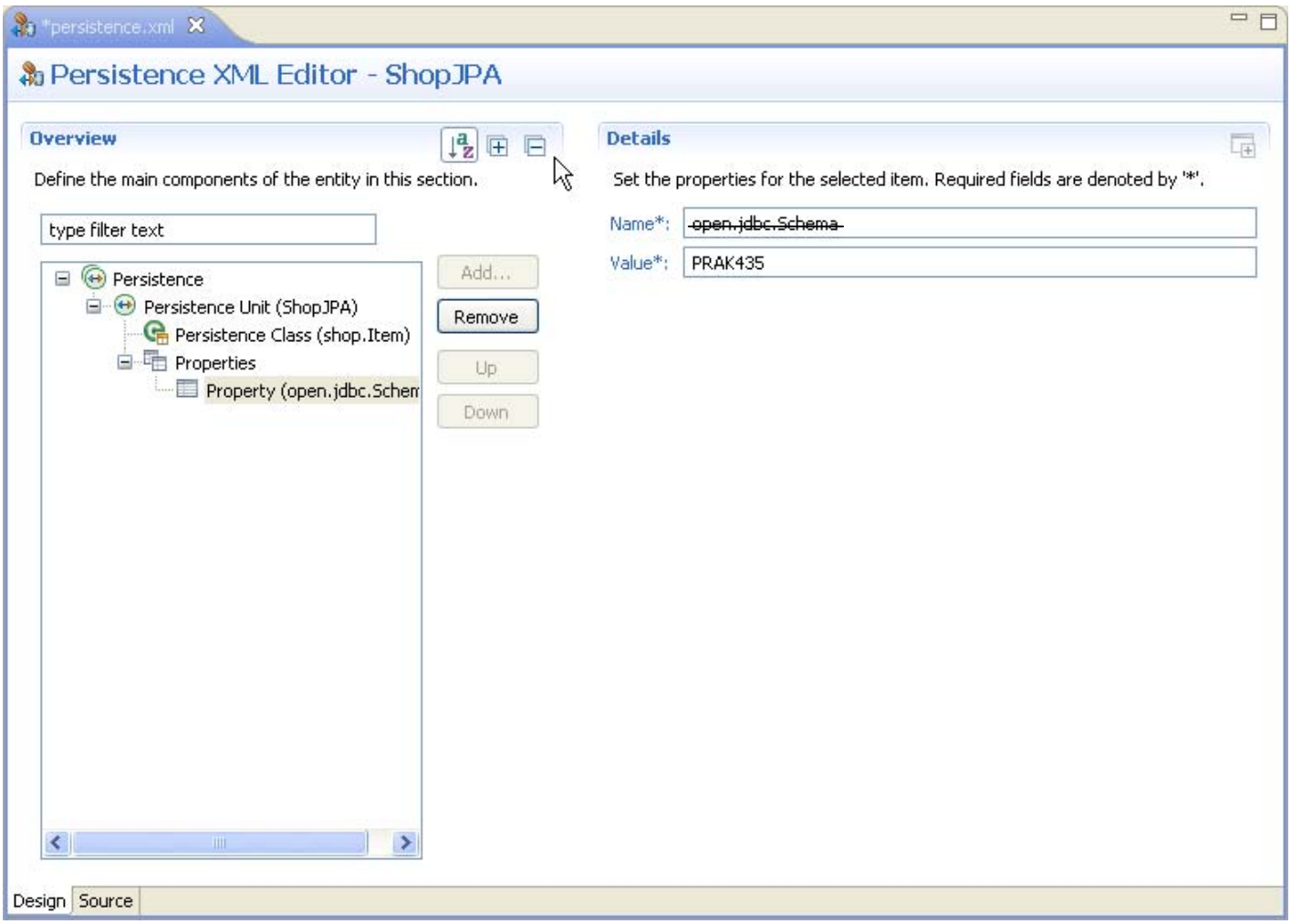
2k auf persistence.xml im ShopJPA-Project (unter META-INF) um es zu editieren.



Die Java Implementierung des X/Open XA-Standards wird als „Java Transaction Architecture“ (JTA) bezeichnet siehe Band 2, Abschnitt 19.1.

JTA als Transaction Type auswählen, sowohl für JTA Data Source als auch Non JTA Data Source „jdbc/leia“ eingeben. (JTA ist die Java Transaction Architecture. „jdbc/leia“ ist die Data Source, die wir in WAS6.1 bereits definiert haben.)

***Tipp:* Überprüfen Sie in der *Administrative console* den *JDNI name.* Ggf. müssen sie „jdbc/leia“ abändern.**



Expandiere Persistence Unit (→ Properties, falls vorhanden, sonst: Add → Properties) und füge eine neue Property mit dem Namen openjpa.jdbc.Schema für den Wert (Value) „PRAK[xxx]“ hinzu.

1k auf den Source-Tab um den Inhalt von persistence.xml anzusehen. Dieser sollte nun so aussehen:

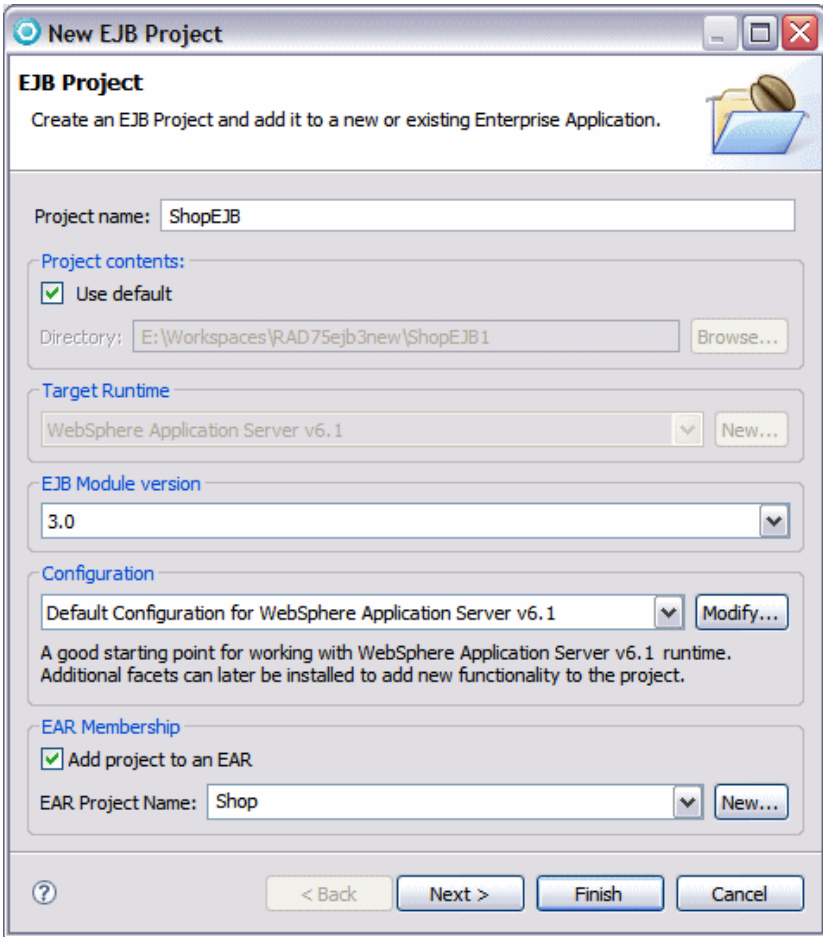
```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd">
    <persistence-unit name="ShopJPA">
         <jta-data-source>jdbc/leia</jta-data-source>
         <non-jta-data-source>jdbc/leia</non-jta-data-source>
         <class>
         shop.Item
         </class>
         <properties>
              <property name="openjpa.jdbc.Schema" value="PRAK435" />
         </properties>
    </persistence-unit>
</persistence>
```

Die Änderungen abspeichern (Strg+S).

2.4 Erstellen eines EJB 3.0 Projekes für die Session Bean

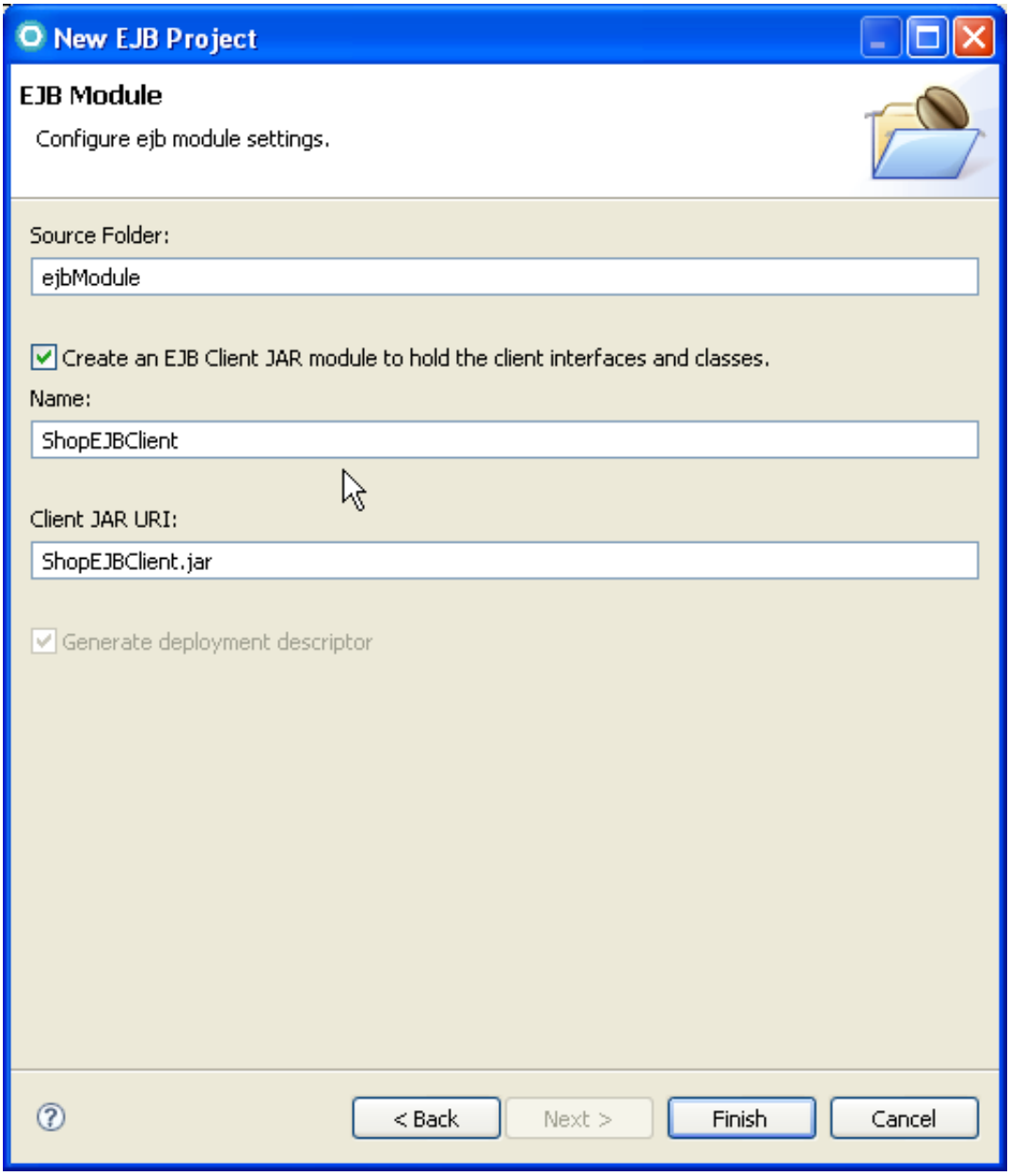
In der JavaEE-Perspektive, File → New → Project auswählen und dann EJB → EJB Project.

1k auf Next.



Eingabe von ShopEJB als Project Name.

Default Configuration for WebSphere Application Server v6.1 auswählen und das Projekt zu der Shop Enterprise Application hinzufügen (schon vorausgewählt). 1k auf Next.



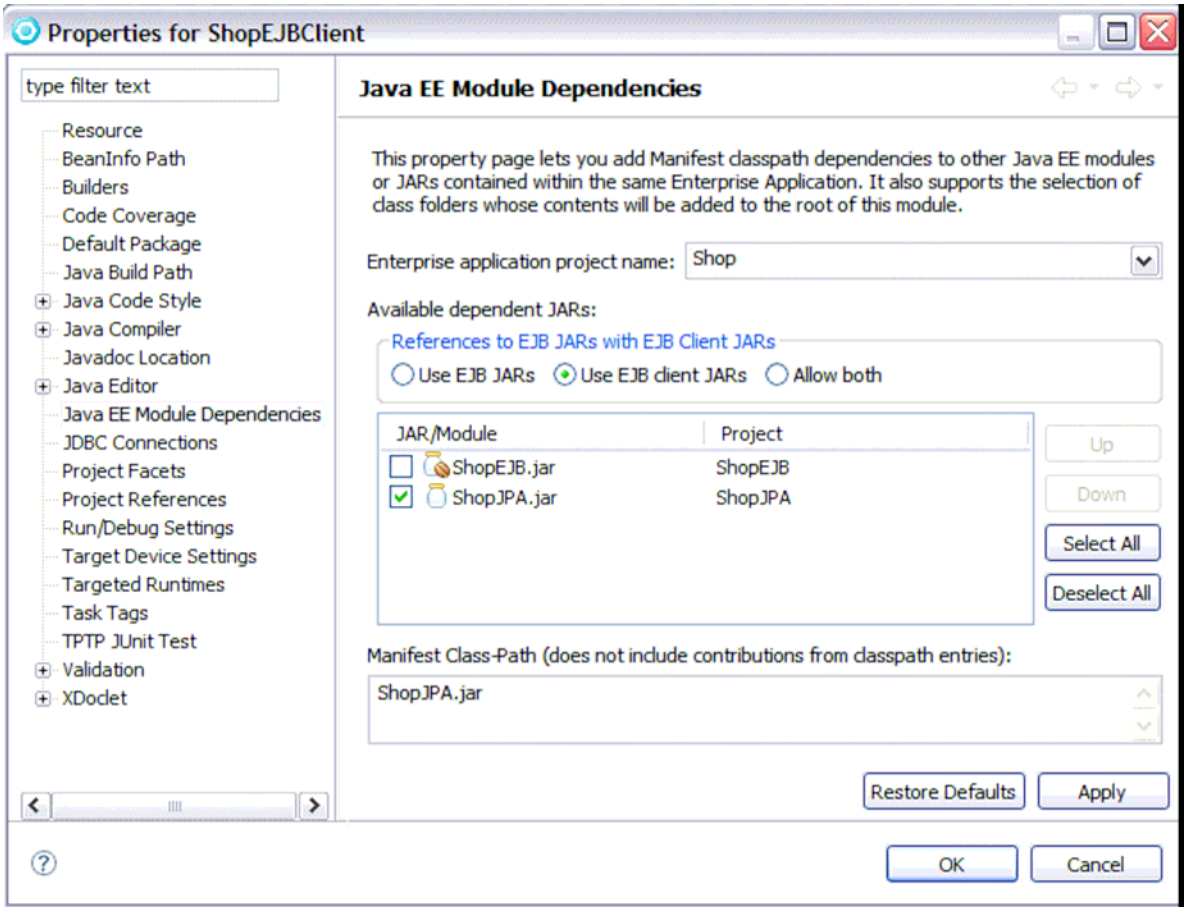
Auf das EJB3.0-Projekt wird mit Web-Clients zugegriffen, die mit dem Application Developer als Dynamic Web Projects entwickelt wurden. Das ShopEJBClient.jar enthält public (Business) Schnittstellen der EJBs. Es ist die einzige JAR-Datei, die vom Client (z. B. Web-Client) benötigt wird. Das EJB-Client-Projekt ist daher abhängig von den Client-Projekten, welche auf die EJB3.0-Bean zugreifen.

Falls noch nicht automatisch ausgewählt/-füllt:
Wählen Sie „Create an EJB Client JAR module to hold the client interfaces and classes“, und übernehmen Sie die Default Namen.

Es ist immer eine guter Design-Ansatz, das Public Interface (das Business Interface der EJB3.0-Bean) von ihrer Implementierung (der Implementierungsklasse der EJB3.0-Bean) zu trennen. Zwei Projekte mit gut definierten Zielen macht ihre Wartung einfacher.

1k auf Finish.

2.5 Hinzufügen des Persistenz-Moduls zu dem Session EJB Modul.

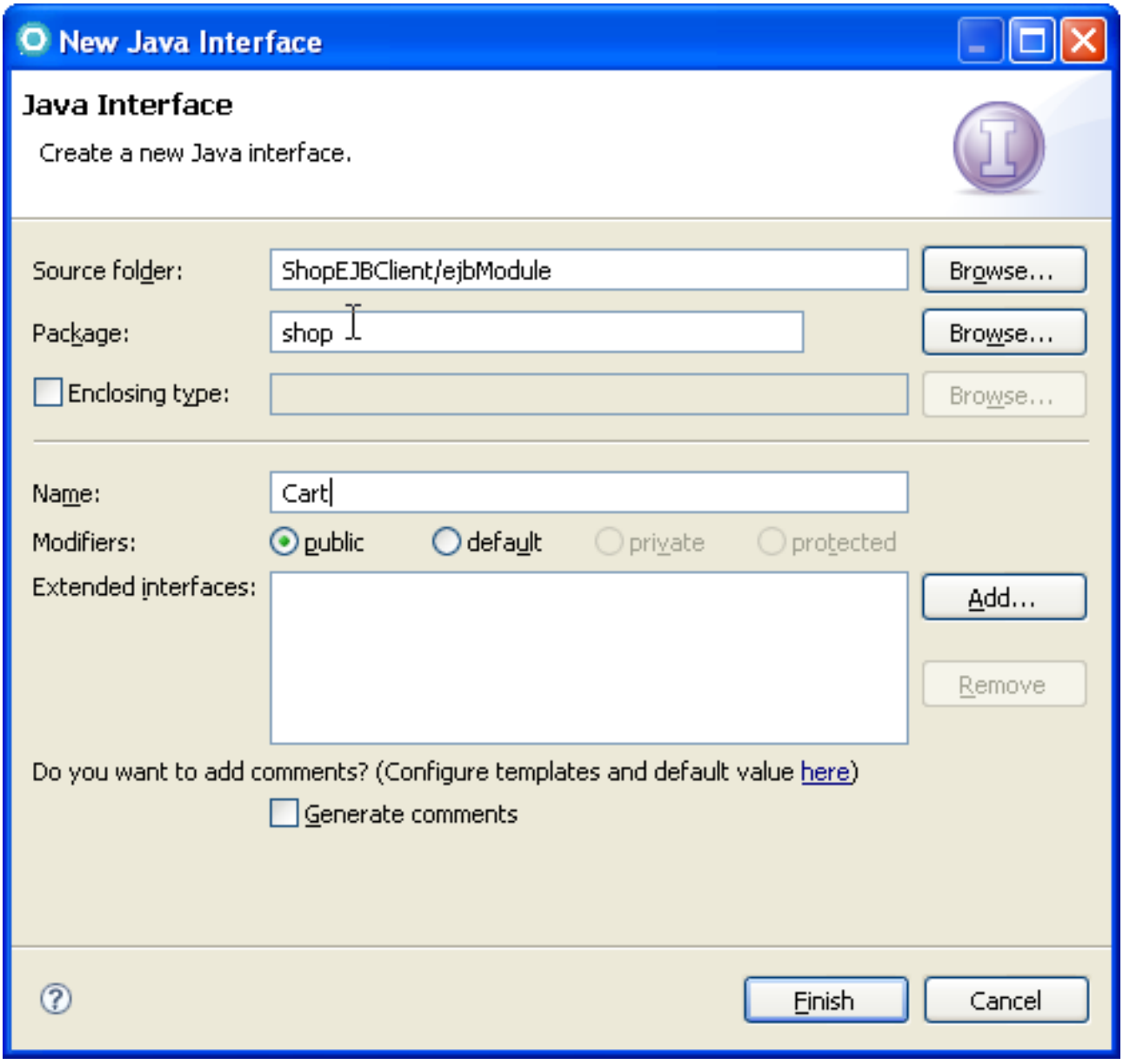


Die Session-Bean für den Warenkorb hat Zugriff auf die Artikel der Entity-Bean. Wir fügen das JPA-Modul als eine Dependency zu dem EJB-Client-Modul hinzu.

1kr auf ShopEJBClient-Projekt und 1k auf Properties.
Java EE Module Dependencies und ShopJPA.jar auswählen. 1k auf OK.

2.6 Erstellen eines Business-Interface

Eine EJB 3.0 Session Bean implementiert ein Business Interface. Wir erstellen diese Schnittstelle, ehe wir die Session-Bean erstellen.



Erstellen Sie einen Business-Interface shop.Cart in dem ShopEJBClient-Projekt (unter ejbModule).

Das Interface hat vier public methods

- **public void add(int id) Item zum Einkaufswagen hinzufügen.**
- **public void remove(int id) Item aus dem Einkaufswagen löschen.**
- **public List<Item> getItems() List Vorgandene Items auflisten.**
- **public List<Item> checkout() Mit den Items auschecken.**

Der Code ist hier dargestellt:

```
package shop;
import java.util.List;
public interface Cart {
        public List<Item> getItems();
        public List<Item> checkout();
        public void add(int id);
        public void remove( int id);
}
```

2.7 Erstellen einer Stateful Session Bean

Nach Erstellen der Interface shop.Cart erstellen wir die dazugehörige Implementation als eine Stateful Session Bean in dem ShopEJB-Projekt. Die Stateful Session Bean kann Entities in Synchronisation mit der Datenbank handhaben.

→ Fügen sie *ShopEJBClient* und *ShopJPA* projects als Dependencies zu dem ShopEJB-Projekt zu. (über die Java EE Module Dependencies in dem Properties Dialog für ShopEJB)

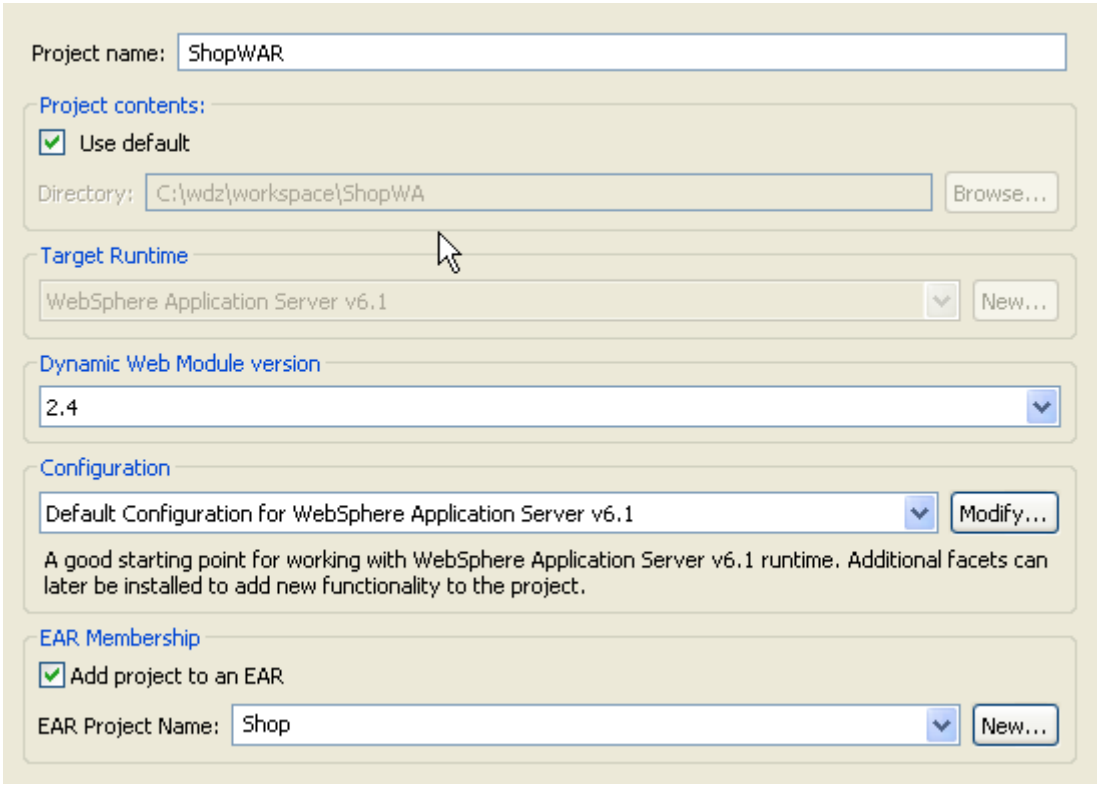
In dem ShopEJB-Projekt, erstellen Sie die Bean Klasse shop.CartBean. Diese implementiert das shop.Cart-Interface. Der Code ist hier dargestellt:

```
package shop;
import java.util.ArrayList;
import java.util.List;
import javax.ejb.Remove;
import javax.ejb.Stateful;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.persistence.PersistenceContextType;
import javax.persistence.Query;
@Stateful
public class CartBean implements Cart {

@PersistenceContext(type = PersistenceContextType.EXTENDED)
EntityManager em;
private List<Item> selectedItems = new ArrayList<Item>();

public void add(int id) {
        Query query = em.createNamedQuery("getItemByID");
        query.setParameter("id", id);
        Item item = (Item) query.getSingleResult();
        item.setQuantity(item.getQuantity()-1);
        selectedItems.add(item);
}
@Remove
public List<Item> checkout() {
        System.out.println("Checkout:");
        Item[] items = selectedItems.toArray(new Item[
selectedItems.size()]);
        for (int i=0; i< items.length; i++){
                Item item = items[i];
        System.out.println(item.getId() + " " + item.getName());
        }
        em.flush();
        return selectedItems;
}

public List<Item> getItems() {
        return em.createNamedQuery("getItem").getResultList();
}

public void remove(int id) {
        Query query = em.createNamedQuery("getItemByID");
        query.setParameter("id", id);
        Item item = (Item) query.getSingleResult();
        item.setQuantity(item.getQuantity()+1);
        selectedItems.remove(item); }
}
```

Die Stateful Session Bean verwendet einen erweiterten Persistenzkontext um die Entitäten am EntityManager angehangen zu halten. Der Persistenz-Kontext (die Menge der Entities) wird zwischen Transaktionen bewahrt, anstatt sie am Ende jeder Transaktion zu verwerfen. Die Änderungen werden in die Datenbank am Ende jeder Transaktion geschrieben. In unserem Fall geschieht dies am Ende jeder add- oder remove-Methode. Bei der Ausführung der Methode, die mit @Remove annotiert ist, wird die Stateful Bean entfernt.

In den Methoden add und remove benutzen wir die Named Query „getItemByID“. Diese haben wir in der Item Entity Bean definiert.

2.8 Erstellen eines Dynamic Web-Projektes



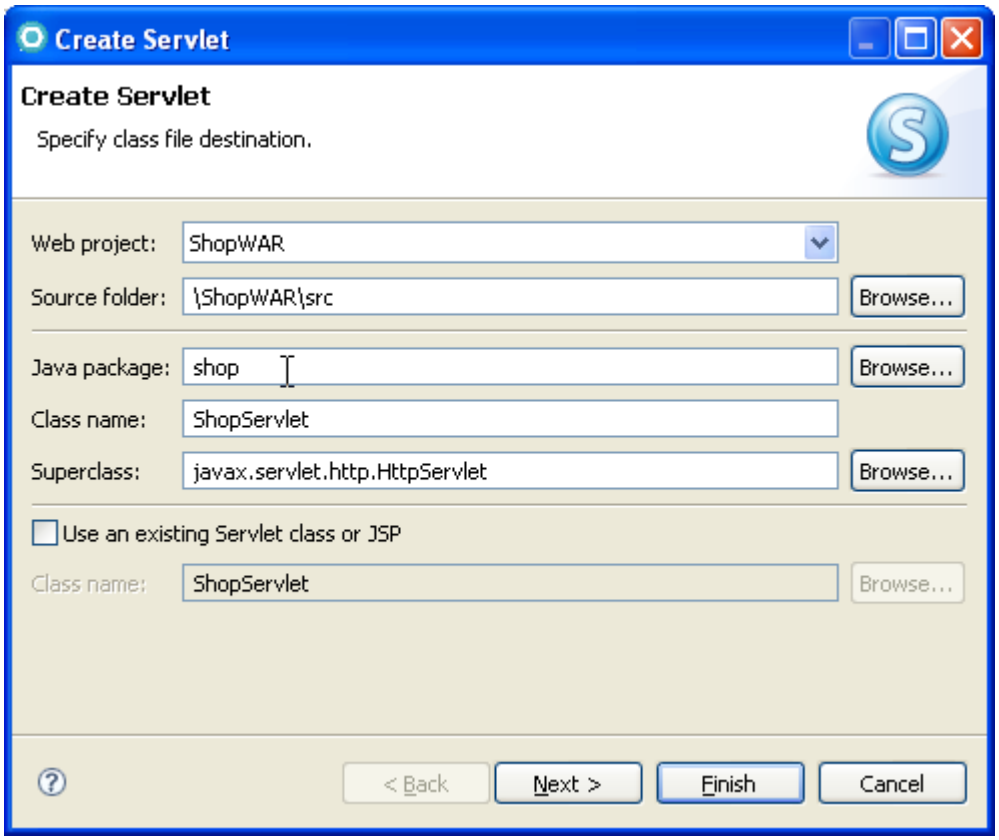
Ein Dynamic Web Project implementiert einen Web Container, der u.a. Servlets, Java Server Pages, Tag Libraries für die JSPs sowie statische HTML-Seiten enthalten kann.

Klicken Sie auf Projekt → Neu → Dynamic Web Project. Schreiben Sie "ShopWAR" als Project name. Wählen Sie "Default Configurateion für WebSphere Application Server v.6.1" für Configuration, und wählen Sie Add project to an EAR.

Klicken Sie auf Finish.

Öffnen Sie die Properties und wählen Sie ShopJPA und ShopEJBClient Projects als Java EE Module Dependencies.

Erstellen Sie ein Servlet shop.ShopServlet, indem Sie rechts klicken auf Deployment Descriptor und New → Servlet



1k auf Finish. Das ShopServlet wird nun im Editor geöffnet.

Geben Sie den folgenden Code für die Klasse ein.

```
package shop;
import java.io.IOException;
import java.util.List;

import javax.ejb.EJB;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;


public class ShopServlet extends HttpServlet {
private static final long serialVersionUID = 1L;
public static final String CART_REF_NAME = "shop.cart";
public ShopServlet() {
super();
}

protected void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    HttpSession session = request.getSession();
    Cart cart = (Cart) session.getAttribute(CART_REF_NAME);
    if (cart == null) {
        try {
            System.err.println("runs here");
            Context ctx = new InitialContext();
            cart = (Cart) ctx.lookup("java:comp/env/ejb/Cart");

            session.setAttribute(CART_REF_NAME, cart);
        } catch (NamingException ne) {
            throw new ServletException(ne);
        }
    }
    /**** the business logic ****/
    java.io.PrintWriter out = response.getWriter();
    // display the available items
    out.println("<h2>Items</h2>");
    List<Item> itemlist = cart.getItems();
    Item[] items = itemlist.toArray(new Item[itemlist.size()]);
    for (int i=0; i< items.length; i++){
        Item item = items[i];
        out.println(item.getId() + "   " + item.getQuantity() + "   "
            + item.getName()+ "<br>");
    }
```

Shop Servlet Code Teil 1

```
    // fill the shopping cart
    cart.add(1001); // add a book to cart
    cart.add(1001); // add a book to cart
    cart.add(1003); // add a dvd to cart
    cart.add(1002); // add a cd to cart
    cart.add(1002); // add a cd to cart
    cart.add(1002); // add a cd to cart
    cart.remove(1001); // remove a book from cart
    cart.remove(1002); // remove a cd from cart
    // display the remaining items
    out.println("<h2>Remaining Items</h2>");
    itemlist = cart.getItems();
    items = itemlist.toArray(new Item[itemlist.size()]);
    for (int i=0; i< items.length; i++){
            Item item = items[i];
            out.println(item.getId() + "   " + item.getQuantity()
                    + "   " + item.getName()+ "<br>");
    }
    // checkout and list the content of the shopping cart
    out.println("<h2>Checkout Cart</h2>");
    itemlist = cart.checkout();
    items = itemlist.toArray(new Item[itemlist.size()]);
    for (int i=0; i< items.length; i++){
            Item item = items[i];
            out.println(item.getId() + "   " + item.getName()+ "<br>");
    }
    // remove session data
    session.removeAttribute(CART_REF_NAME);
}

protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
    doGet(request, response);
}
}
```

Shop Servlet Code Teil 2

2.9 Mapping der EJB Logical Reference

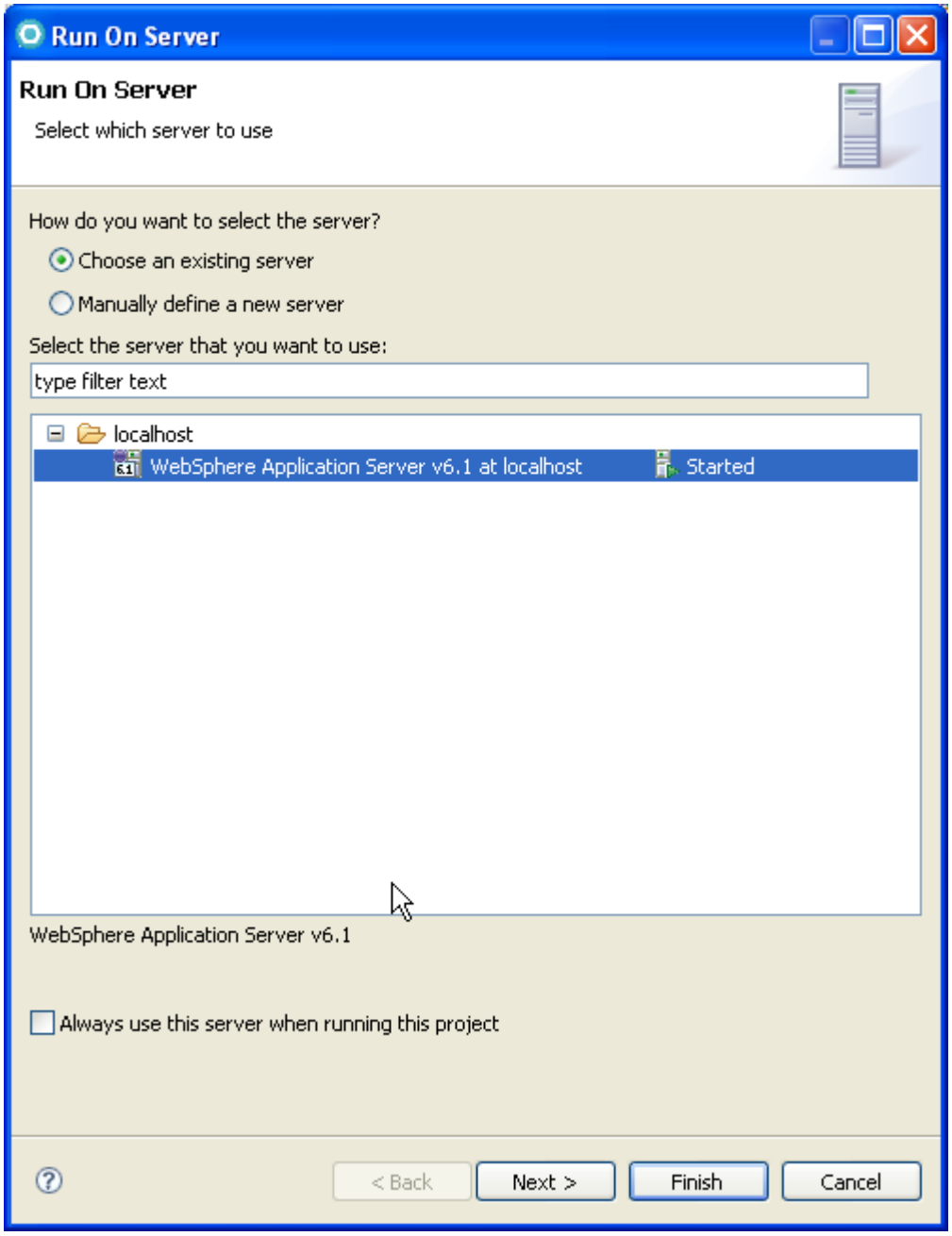
Öffnen Sie den Deployment Descriptor-Editor für ShopWAR

Wechseln Sie zu Source des Web Deployment Descriptor-Editors .Fügen Sie die ejb//Cart Reference hinzu. Weil die Web Application und die Enterprise Bean Teil der gleichen Enterprise Application sind, können wir das ejb-local-ref Element benutzen, um die Referenz zu erstellen.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4".....>
<display-name>ShopWAR</display-name>
<servlet>
<description></description>
<display-name>ShopServlet</display-name>
<servlet-name>ShopServlet</servlet-name>
<servlet-class>shop.ShopServlet</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>ShopServlet</servlet-name>
<url-pattern>/ShopServlet</url-pattern>
</servlet-mapping>
<ejb-local-ref>
        <description>
        </description>
        <ejb-ref-name>ejb/Cart</ejb-ref-name>
        <ejb-ref-type>Session</ejb-ref-type>
        <local-home></local-home>
        <local>shop.Cart</local>
        <ejb-link>CartBean</ejb-link>
</ejb-local-ref>
<welcome-file-list>
…..
</web-app>
```

2.10 Das Projekt auf dem Server ausführen

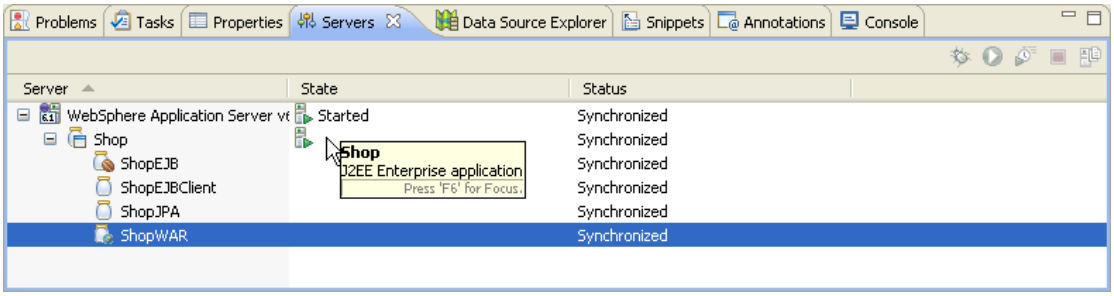
1kr auf Shop-Projekt. Wählen Sie Run As → Run on Server. wählen sie das Server Environment, das wir in Teil 1 erstellten aus.



1k auf Finish.

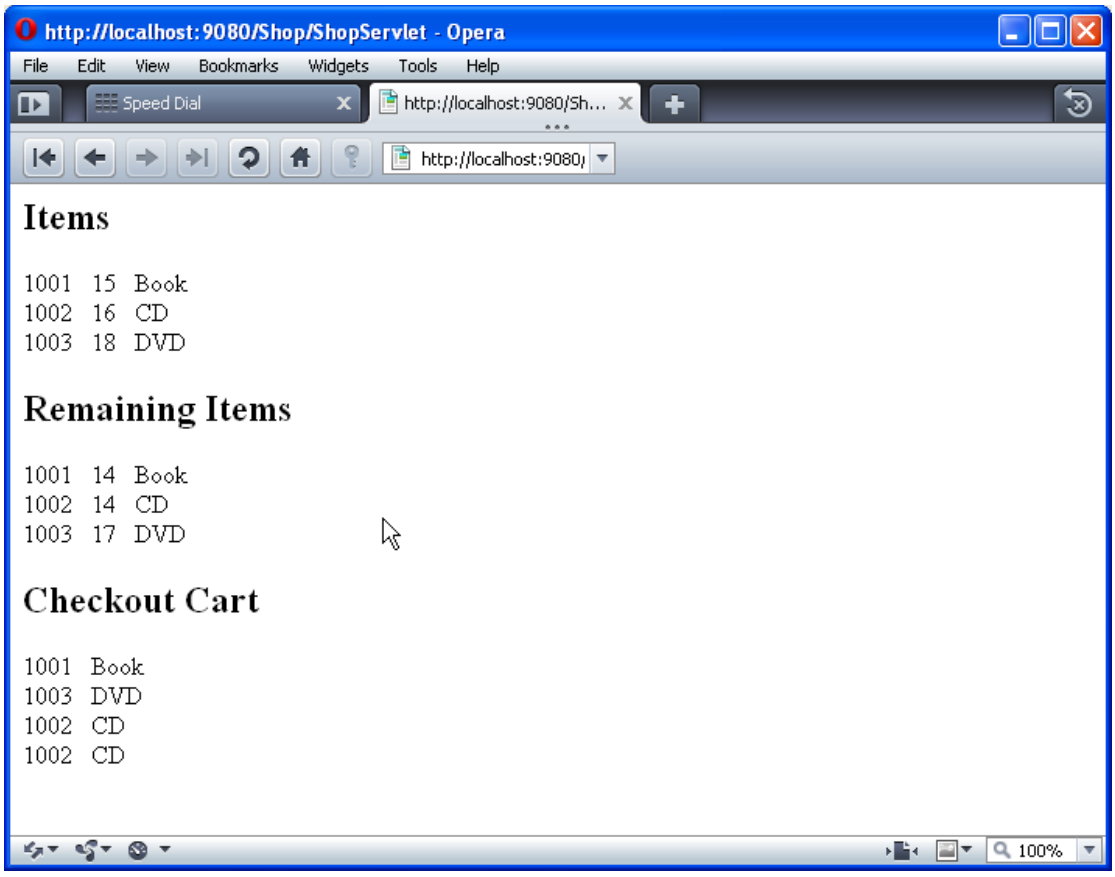
Selbst-Test

1. **Woher weiß RAD7.5, auf welchem WebApplication Server das Projekt ausgeführt werden soll?**



Warten Sie, bis alle Projekte gestartet wurden.

Öffnen sie einen Web Browser. Geben Sie “localhost:9080/ShopWAR/ShopServlet” ein. Sie sollten dieses Fenster sehen.



Herzlichen Glückwunsch! Sie haben erfolgreich eine EJB 3.0 Anwendung erstellt.

Aufgaben:

1. **Personalisieren Sie Ihr ShopServlet so, dass die Ergebnisseite Ihren Namen und Ihre PRAK[xxx]-Kennung enthält.**
2. **Erstellen Sie von dem Ergebnis im Browser ein Screenshot.**
3. **Machen Sie des Weiteren Screenshots von den RAD-Views:**
    1. **Data Source Explorer (Schemas [Filtered] ausgeklappt)**
    2. **Servers (Shop augeklappt)**
4. **Laden Sie sie in den Abgabeordner hoch.**

Viel Erfolg!

WebSphere and Message Driven Beans

1 Überblick

1.1 Test Konfiguration

Dieses Tutorial zeigt, wie eine einfache EJB Message Driven Bean Anwendung mit Rational Application Developer 7.5 (kurz RAD7.5) erstellt und auf einem WebSphere Application Server 6.1 (kurz WAS6.1) deployed werden kann. Wir verwenden die gleiche RAD7.5 und WAS6.1 Konfiguration wie im letzten EJB Tutorial.

Die in diesem Tutorial zu entwickelnde Test-Konfiguration ist in Abbildung 1.1 wiedergegeben. Sie besteht aus einem Browser, einem WebSphere Application Server6.1, der unter Windows läuft, sowie einem Konsole Fenster.

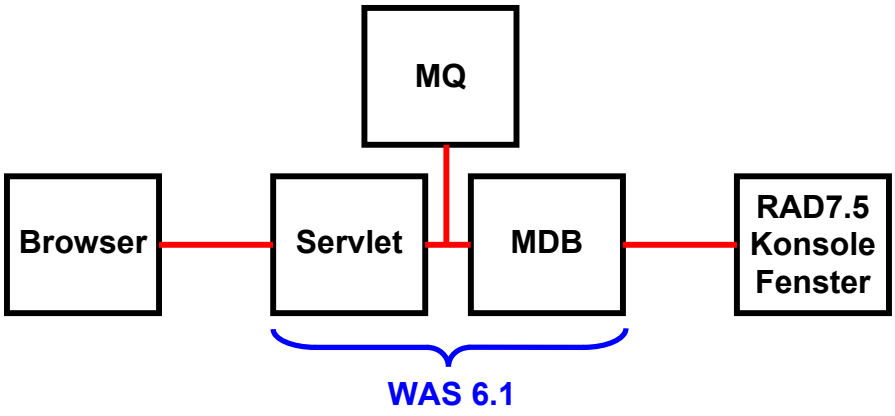


Abbildung 1.1

Der Browser übergibt einen Message Text an das Servlet. Dieses agiert als ein Klient für eine Message Driven Bean und deren Message Queue. Die MDB gibt den Text der Message in einem RAD7.5-Konsolenfenster wieder. Die integrierte Version von MQSeries verwaltet die Queue (siehe Band 1 Abschnitt 10.1.6, sowie Band 2, Abschnitt 15.4.5) .

1.2 Inhalt Übersicht

Selbst-Test

- **Könnten sie zuerst die Anwendung unter RAD 7.5 entwickeln und dann erst des WAS6.1 konfigurieren?**
- **Benutzen Sie für die Konfiguration von WAS6.1 die RAD7.5 Entwicklungsumgebung?**

2 JMS (Java Message Service)

Messaging ist ein Verfahren zur Kommunikation zwischen Softwarekomponenten oder zwischen Anwendungen. Ein Messaging-System ist eine Peer-to-Peer-Einrichtung: Ein Messaging-Client sendet Nachrichten an und/oder empfängt Nachrichten von anderen Klienten.

Messaging ermöglicht eine verteilte (distributed) Kommunikation, die lose gekoppelt ist. Eine Komponente sendet eine Nachricht an ein Ziel, und der Empfänger kann die Nachricht vom Ziel abrufen. Es ist nicht erforderlich, dass Sender und Empfänger gleichzeitig kommunizieren. Der Absender braucht nichts über den Zustand des Empfängers zu wissen, und der Empfänger muss nichts über den Absender kennen. Der Sender und der Empfänger brauchen nur zu wissen welches Nachrichtenformat benutzt, und welche Ziel Adresse verwendet wird. In dieser Hinsicht unterscheidet sich Messaging von eng gekoppelten Technologien wie Remote Method Invocation (RMI). Im Gegensatz zu E-Mail wird Messaging für die Kommunikation zwischen Software-Anwendungen oder Software-Komponenten verwendet, und implementiert eine garantierte Kommunikation unter ACID Bedingungen.

Der Java Message Service (JMS) definiert den Standard für eine zuverlässige Enterprise Messaging.
Die JMS API ist ein integraler Bestandteil der Java Enterprise Edition (JEE)-Plattform, und Anwendungsentwickler können Nachrichten mit Komponenten mit JEE APIs ("JEE Komponenten") zu verwenden.

Message Driven Beans (MDB) sind diejenigen Komponenten, die EJB-Systeme für asynchrone Kommunikation zugänglich machen. Hierzu wird der Java Message Service (JMS) verwendet. Diese Sorte von Beans wird z. B. häufig für die Kommunikation mit Legacy-Systemen genutzt. Auch für die Ausführung von klassischerweise asynchron auszuführenden Operationen, von deren Erfolg und Dauer die Performanz einer übergeordneten Anwendung nicht abhängen soll, bieten sich Message Driven Beans an.

Sinn und Zweck von Message Driven Beans ist es, Nachrichten asynchron abzuarbeiten. Dazu schickt ein Client, wie z.B. ein Session Bean oder ein Servlet, eine Nachricht in eine Queue. Die Queue übergibt die Nachricht an eine MDB und diese verarbeitet sie dann.

Der Java Message Service ist ein Satz von Java-APIs, der es ermöglicht:

- **Anwendungen zu erstellen**
- **Nachrichten zu senden**
- **zu empfangen**
- **zu lesen**

Für die Anwendung braucht man einen Provider, der die API umsetzt und somit den Dienst bereitstellt. WAS6.1 kann als ein derartiger Provider dienen und MDBs sind geeignet, die Dienstleistung zu erbringen. Die WAS6.1 Implementierung benutzt die integrierte Version von WebSphere MQ.

Das JMS API Programmiermodell besteht aus:

- **Connections (Verbindungen)**
- **Sessions**
- **Messages**
- **Message Producers**
- **Message Consumers**

Die JMS API definiert eine Reihe von Schnittstellen und die damit verbundene Semantik, die es Programmierern ermöglicht, Messaging-Komponenten in Java zu entwickeln, die eine Kommunikation mit anderen Messaging-Implementierungen ermöglichen.

Verwaltete (administered) Objekte sind vorkonfigurierte JMS-Objekte, die von einem Administrator erstellt wurden, und aus zwei Komponenten bestehen: *Connection Factories* und *Destinations.*

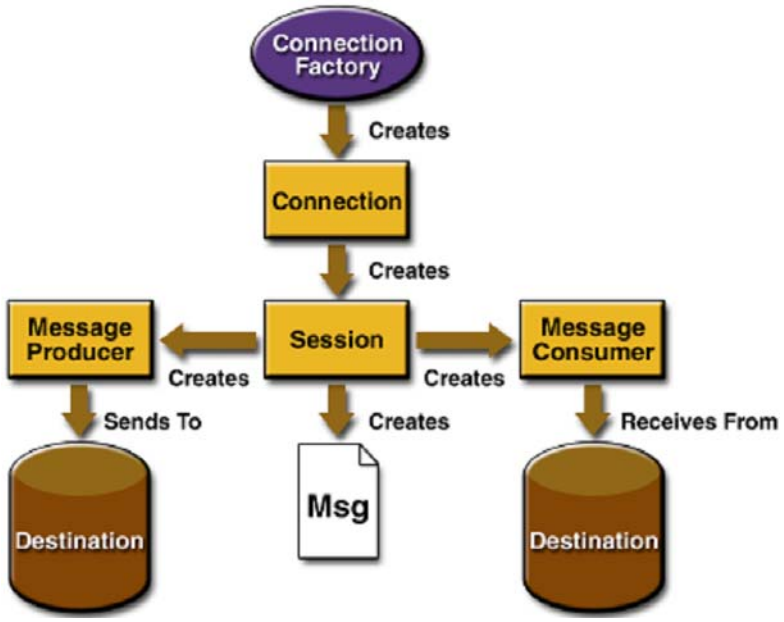
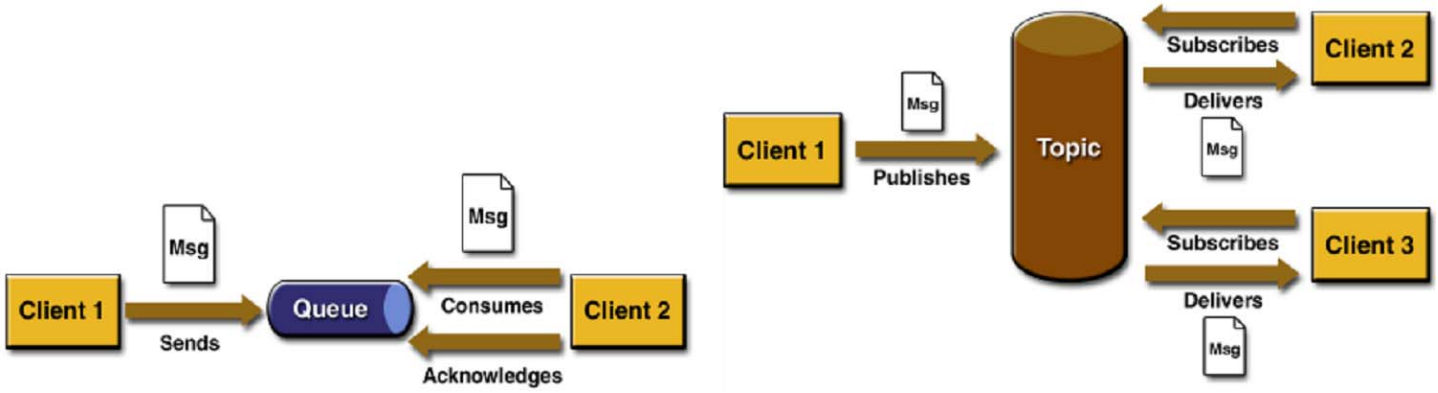


Abbildung 2.1 JMS Programming Architecture

Eine *Connection Factory* ist ein Client-Objekt, welches eine Verbindung zu einem Provider erzeugt. Sie kapselt einen Satz der Verbindungseinstellungen Konfigurationsparameter, der von einem Administrator definiert wurde. Jede Connection Factory ist eine Instanz der

- **QueueConnectionFactory oder**
- **TopicConnectionFactory Schnittstelle.**



Queue Modell **5Topic**

Abbildung 2.2 Queue / Topic Destination

Eine Destination ist ein Objekt, welches das Ziel (Target) spezifiziert, wohin ein Message Producer die Message hin liefert, und von wo der Message Consumer (Verbraucher) die Message ausliest. Es existieren zwei Arten von Zielen: Queue und Topic. Im Queue Modell muss jede Nachricht von genau einem Consumer abgeholt werden. Im Topic-Modell kann jede Nachricht von vielen Verbrauchern verarbeitet werden. Die Nachricht wird so lange im Speicher abgelegt, bis alle Verbraucher sie bekommen haben, siehe Abbildung 2.2.

Eine Connection erstellt eine virtuelle TCP/IP-Socket-Verbindung zwischen einem Client und einem Provider Service Daemon.

Eine Sitzung (Session) ist ein Single-Threaded-Context für die Erstellung und Verarbeitung von Nachrichten. Die Sitzungen werden durch eine Verbindung (Connection) erstellt.

Ein Message Producer implementiert die MessageProducer-Schnittstelle, die von einer Sitzung erstellt und verwendet wird, um Nachrichten an ein Ziel zu senden.

Ein Message Consumer ist ein Objekt, das von einer Sitzung erstellt wird, und die MessageConsumer-Schnittstelle implementiert, um Nachrichten zu empfangen, die entweder Queue oder Topic sein können.

Der Zweck einer JMS-Anwendung ist es, Nachrichten zu erstellen, die von anderen Komponenten verwendet werden können. Ein JMS-Nachricht besteht aus drei Teilen: einem Nachrichten-Kopf (Header), Eigenschaften (Properties) und einem Body. Nur der Header ist unbedingt erforderlich; die anderen zwei Teile können in einer Nachricht fehlen.

Header Field	Set By
JMSDestination	send or publish method
JMSDeliveryMode	send or publish method
JMSExpiration	send or publish method
JMSPriority	send or publish method
JMSMessageID	send or publish method
JMSTimestamp	send or publish method
JMSCorrelationID	Client
JMSReplyTo	Client
JMSType	Client
JMSRedelivered	JMS provider

Tabelle 2.1 Properties of Message Header

Ein JMS Message-Header hat eine Reihe von vordefinierten Feldern, die von Clients und Producern verwendet werden, um Nachrichten zu identifizieren und weiterzuleiten. Jedes Header-Feld verfügt über eigene Setter- und Getter-Methoden. Tabelle 2.1 zeigt alle Felder und ihre Anordnung.

Wenn wir zusätzliche Informationen in eine Nachricht für andere Komponenten setzen wollen, könnten wir Message-Properties setzen, z.B. wenn eine Eigenschaft für einen Message-Selector benötigt wird.

Message Type	Body Contains
TextMessage	A java.lang.String object
MapMessage	A set of name-value pairs, with names as String objects and values as primitive types in the Java programming language. The entries can be accessed sequentially by enumerator or randomly by name. The order of the entries is undefined.
BytesMessage	A stream of uninterpreted bytes. This message type is for literally encoding a body to match an existing message format
StreamMessage	A stream of primitive values in the Java programming language, filled and read sequentially.
ObjectMessage	A Serializable object in the Java programming language.
Message	Nothing. Composed of header fields and properties only. This message type is useful when a message body is not required.

Tabelle 2.2 JMS Message Types

Der Body enthält den Inhalt einer Nachricht. Jeder Messsage-Inhalt muss einem vordefinierten Nachrichtenformat folgen, als Message-Typ bezeichnet. Dies ermöglicht es den Software-Komponenten Daten in verschiedenen Formaten zu senden und zu empfangen. Tabelle 2.2 zeigt die möglichen Nachrichtentypen.

Wie Session Beans und Entity Beans benutzen Message Driven Beans (MDB) auch einen Deployment Descriptor (XML-Datei). Ab Version 3 können die meisten Angaben, für die zuvor der Deployment Descriptor notwendig war, mit Annotationen direkt im Java-Code implementiert werden. Dadurch kann der Deployment Descriptor entweder ganz entfallen, oder durch die Angaben in den Annotations überschrieben werden.

Selbst-Test

- **Was ist der Unterschied zwischen einem Message Producer und einem Message Provider?**
- **Benutzen JMS Messages einen Standard wie IDL?**

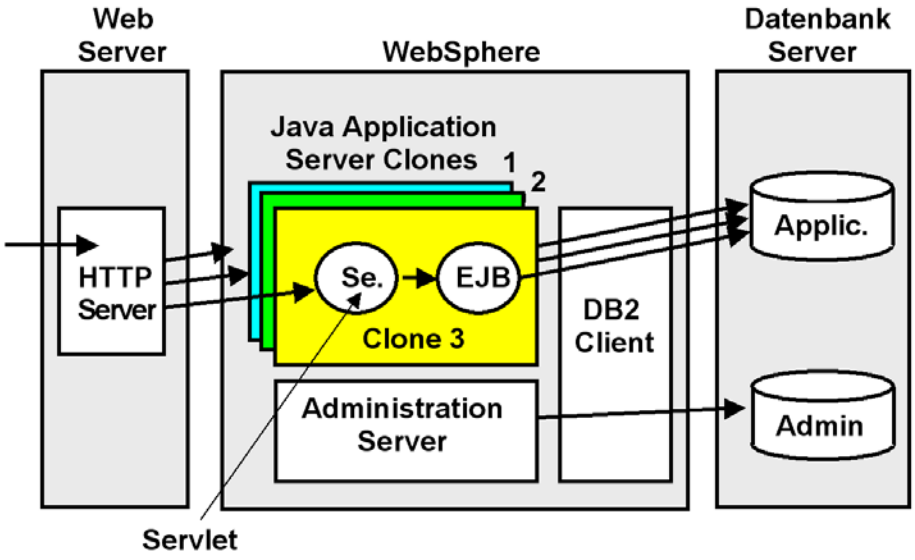
3 Entwickeln einer Message Driven Bean-Anwendung

Eine Message Driven Bean ist ein asynchroner Nachrichtenverbraucher.

In JEE wirkt eine MDB normalerweise als Verbraucher und implementiert die Listener-Schnittstelle, die nur eine Methode „onMessage()“ nach dem JMS-Standard hat. Ein MDB bietet keine Client-Schnittstelle und kann nur passiv ausgeführt werden. Die MDB ist mit einer Queue verbunden (oder einem Topic-Modell, wenn eine Nachricht von mehren MDBs empfangen werden soll), und tut nichts, bis eine Nachricht eintrifft.

WAS6.1 benutzt für die JMS Implementierung eine integrierte Version von WebSphere MQ. An dieser Stelle ist es sinnvoll, wenn Sie Kapitel 10 „WebSphere MQ“ aus Band 1 rekapitulieren.

3.1 Konfiguration in WAS6.1



Die hier gezeigte Abbildung ist eine Kopie aus Band 1, Abschnitt 10.3.

Aus Performance-Gründen muss der Application Server multiprogrammiert arbeiten. Hierzu sind mehrfache Clones des Application Servers vorhanden, die Java-Anwendungen parallel verarbeiten können. Die Clones des Java Application Servers verfügen jeder über eine eigene Java Virtuelle Maschine. Auf mehreren Clones kann die gleiche Anwendung laufen. Es ist auch möglich, dass auf den Clones unterschiedliche Anwendungen laufen, oder dass eine bestimmte Anwendung erst bei Bedarf geladen wird.

WAS6.1 beinhaltet die Möglichkeit, dass die Clones eines WAS6.1 Servers miteinander kommunizieren können, sowie auch mit logischen anderen Application Servern, die sich auf anderen physischen Rechnern befinden. Hierzu unterhält WAS6.1 einen internen Communication Bus, der als Service Integration Bus (SIB) bezeichnet wird.

Ein Service Integration Bus ist ein verwalteter Kommunikationsmechanismus eines WebSphere Application Servers, der die Serviceintegration durch synchrones oder asynchrones Messaging unterstützt. Er unterstützt Anwendungen, die nachrichtenbasierte und serviceorientierte Architekturen verwenden. Ein Service Integration Bus unterstützt eine Gruppe miteinander verbundener Server (oder Server-Cluster), die dem Bus als Member hinzugefügt wurden. Anwendungen stellen über eine Messaging-Steuerkomponente, die den Bus-Membern zugeordnet ist, eine Verbindung zum Bus her.

In unserem Tutorial ist der SIB wirklich overkill, da wir nur einen einzigen Application Server (hier als Server1 bezeichnet) benötigen. Die WebSphere Application Server Architektur ist aber so ausgelegt, dass auch sehr komplexe Anwendungen implementiert werden können. Auf Grund des großen Funktionsumfanges ist die Konfiguration eines WebSpere Web Applikation Vorgangs eine komplexere Aufgabe, als dies bei einfachen Web Application Servern wie z.B. Geronimo oder JBoss der Fall ist. Für die Konfiguration enthält WAS6.1 einen integrierten Administration Server mit entsprechender Benutzer Oberfläche in einem als Konsole bezeichnetem Fenster.

Das Ziel (Destination) eines Busses ist eine logische Adresse, die den Anwendungen als Produzent und/oder als Konsument zugeordnet werden kann. Im Falle einer Java Message Service (JMS) Application ist das Ziel eines Busses eine Warteschlange, welches für das Punkt-zu-Punkt-Messaging verwendet wird.

Ein Service Integration Bus bietet die folgenden Funktionen:

- **Eine Anwendung kann Nachrichten mit jeder anderen Anwendung austauschen. Hierfür wird eine *Destination* verwendet, an die eine Anwendung Nachrichten sendet und von der die andere Anwendung Nachrichten empfängt.**
- **Eine nachrichtenerzeugende Anwendung, d.h. ein *Erzeuger*, kann unabhängig von der Messaging-Steuerkomponente, die der Erzeuger für den Zugriff auf den Bus verwendet, Nachrichten für eine Destination erzeugen.**
- **Eine nachrichtenkonsumierende Anwendung, d.h. ein *Konsument*, kann unabhängig von der Messaging-Steuerkomponente, die der Konsument für den Zugriff auf den Bus verwendet, Nachrichten von einer Destination konsumieren (wenn diese Destination verfügbar ist).**

Zur Implementierung auf unserem WAS 6.1 öffnen Sie einen Browser für den Zugriff auf die WAS Administration Console. Geben Sie "https://localhost:9043/ibm/console/" in die Adresszeile ein. 9043 ist der Default-Port für die WAS6.1 Konsole. Wenn Sie nicht wissen, welchen Port Ihr WAS benutzt, schauen Sie in der Datei <*was-home*>*AppServerprofiles**AppSrv01**properties**portdef.props* nach, wobei <was-home> der WAS-Installationspfad ist. Für die von WAS benutzten Default Port-Nummern siehe auch http://publib.boulder.ibm.com/infocenter/wsdoc400/v6r0/index.jsp?topic=/com.ibm.websphere.iseries.doc/info/ae/ae/adrportbase.htm.**

Selbst-Test

- **Würde man einen SIB benötigen, wenn der Web Application Server nur eine einzige JVM mit je einem Servlet Container und einem EJB Container unterstützt?**
- **Können die SIBs mehrer physischen Server miteinander verbunden werden?**

3.1.1 Erstellen eines Service Integration Bus

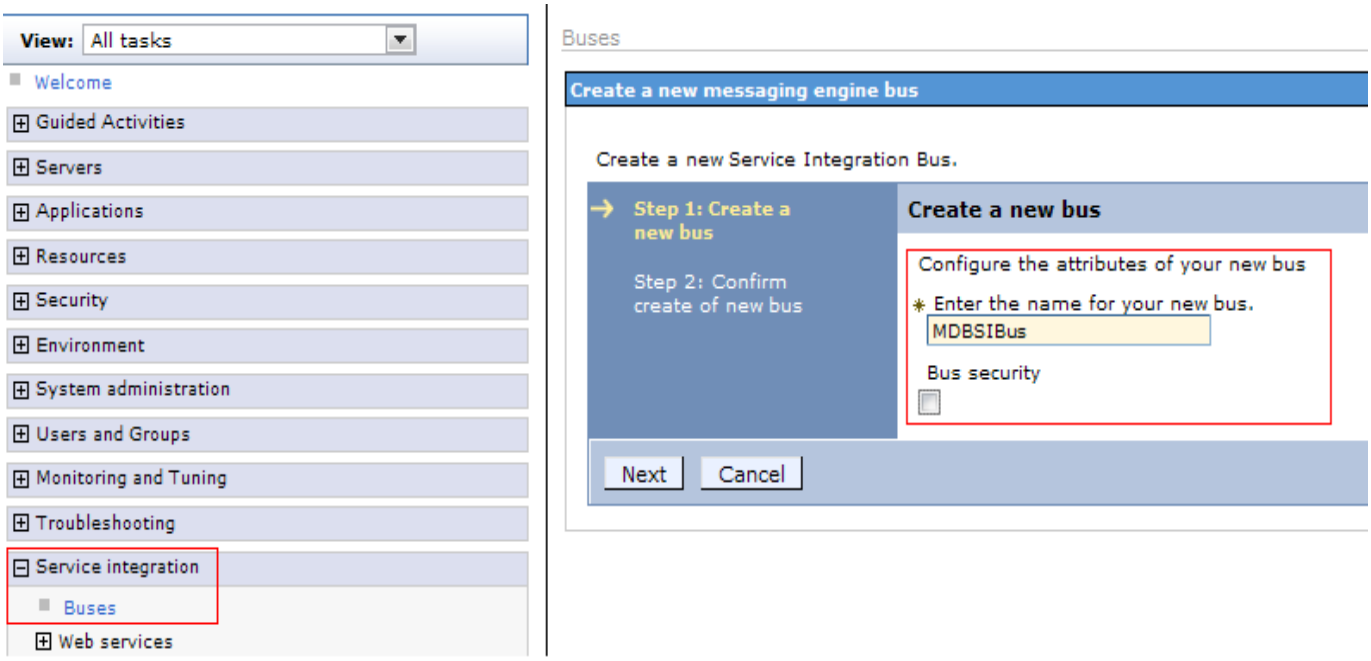


Abbildung 3.1.1-2 Erstellen eines Service Integration Bus

Wählen Sie Service-Integration → Buses und klicken Sie auf New.

Geben Sie MDBSIBus als Name ein, und *deaktivieren* Sie Bus security. (Abbildung 3.1.1-2)

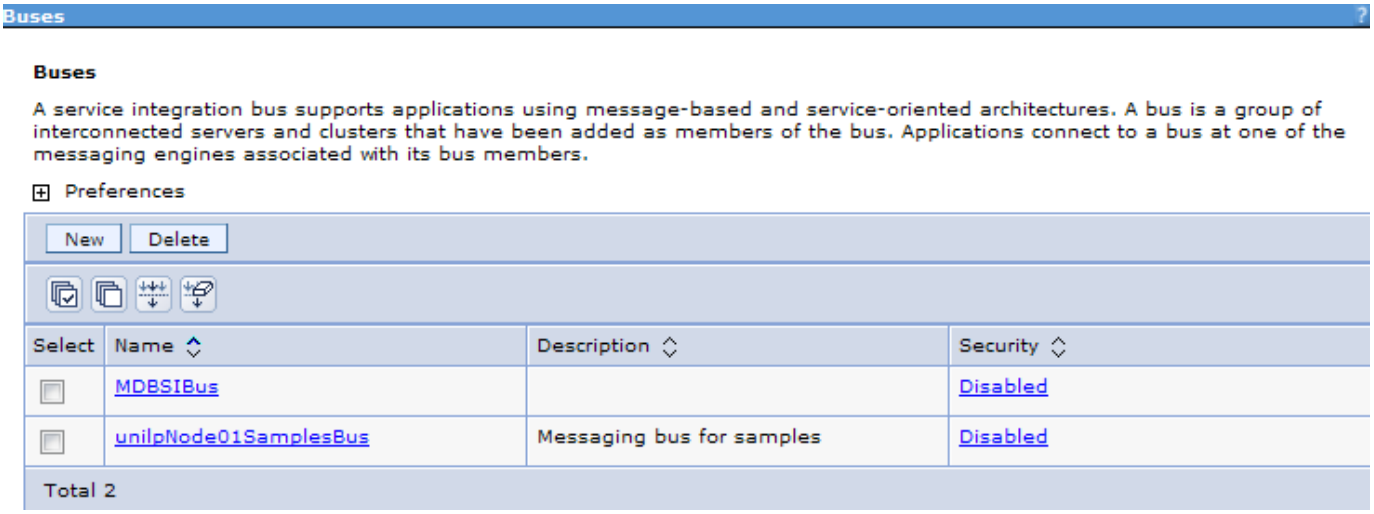


Abbildung 3.1.1-3 Erstellen eines Service Integration Bus (2)

Klicken Sie auf Next und klicken Sie dann auf Finish, um die MDBSIBus-Konfiguration zu bestätigen.

***Hinweis:* Nach jeder Änderung müssen Sie auf Save drücken, um alle Änderungen im WAS zu speichern. Bitte beachten sie das bei allen weiteren Änderungen, wir werden dies im Text nicht wiederholen.**

3.1.2 Erstellen der Messaging-Engine

Ein SIB wird benötigt, um einen (oder mehrere) Applikations-Server (oder Application Server-Cluster) zu betreiben. Für die meisten Entwicklungsaktivitäten, werden Sie einen einzigen Bus-Member für den Bus erstellen und zuweisen. Wenn ein Bus-Member hinzugefügt wird, wird eine Messaging-Engine für diese WebSphere-Instanz erstellt. Diese unterhält ihren eigenen Datenspeicher für Nachrichten.

Siehe auch:
http://www.ibm.com/developerworks/websphere/techjournal/0501_reinitz/0501_reinitz.html

http://www.ibm.com/developerworks/websphere/techjournal/0502_reinitz/0502_reinitz.html#ibm-pcon

Eine Messaging-Engine ist eine Server-Komponente, welche die Kern-Messaging-Funktion eines Service Integration Busses zur Verfügung stellt. Eine Messaging-Engine verwaltet Bus-Ressourcen und ermöglicht es Anwendungen, mit dem Bus zu kommunizieren.

Jede Messaging-Engine ist einem Server zugeordnet, der als ein Mitglied eines Busses assoziiert wurde. Wenn Sie einen Anwendungsserver als Bus-Member hinzufügen, wird eine Messaging-Engine automatisch für dieses neue Mitglied erstellt. Wenn Sie denselben Server als Mitglied mehrerer Busse hinzufügen, wird der Server mit mehreren Messaging-Engines (eine Messaging-Engine für jeden Bus) verbunden. In ihrer einfachsten Form kann ein Bus von einer einzigen Messaging-Engine realisiert werden.

Messaging-Engines haben einen Namen, der sich aus den Namen des Bus Members ergibt. Jede Messaging-Engine hat auch eine universelle eindeutige Kennung (UUID), die eine eindeutige Identität für die Messaging-Engine bietet. Wenn Sie eine Messaging-Engine löschen und neu erstellen, wird sie eine andere UUID haben. Sie wird nicht durch den Bus als die gleiche Engine erkannt werden, auch wenn sie vielleicht den gleichen Namen hat.

Auf geht's mit dem Erstellen der Messagine-Engine!

Selbst-Test

- **Wie viele Messaging-Engines benötigt ein Server?**

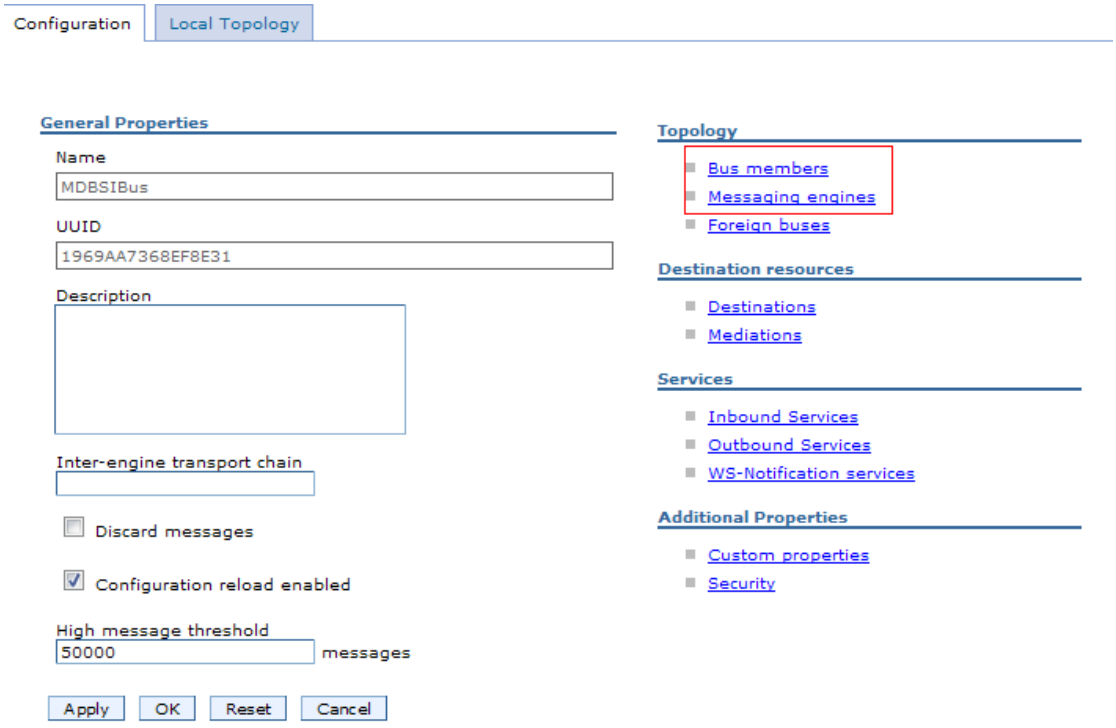


Abbildung 3.1.2-1 Erstellen einer Messaging-Engine

1k auf MDBSIBus um in die Properties Configuration Page zu gelangen (Abbildung 3.1.2-1).

1k auf Bus members in der Topology-Auswahl.

Klick Add und selektiere server1 in dem Server-DropDown-Menu. 1k auf Next.
Selektiere File store, und 1k auf Next.

Ändere die Default Log-Größe auf 20MB, die Minimum permanente und temporäre Store-Größe auf 20MB, und das Maximum auf 100MB.

1k auf Next und Finish.

Abbildung 3.1.2-2 Erstellen einer Messaging-Engine (2)

Zurück zum MDBSIBus Panel, Klick Messaging-Engines um alle verfügbaren Messaging-Engines für den MDBSIBus (Abbildung 3.1.2-2) aufzulisten.

3.1.3 Erstellen einer Destination

Messaging ermöglicht eine verteilte (distributed) Kommunikation, die lose gekoppelt ist. Eine Komponente sendet eine Nachricht an ein Ziel, und der Empfänger kann die Nachricht vom Ziel abrufen.

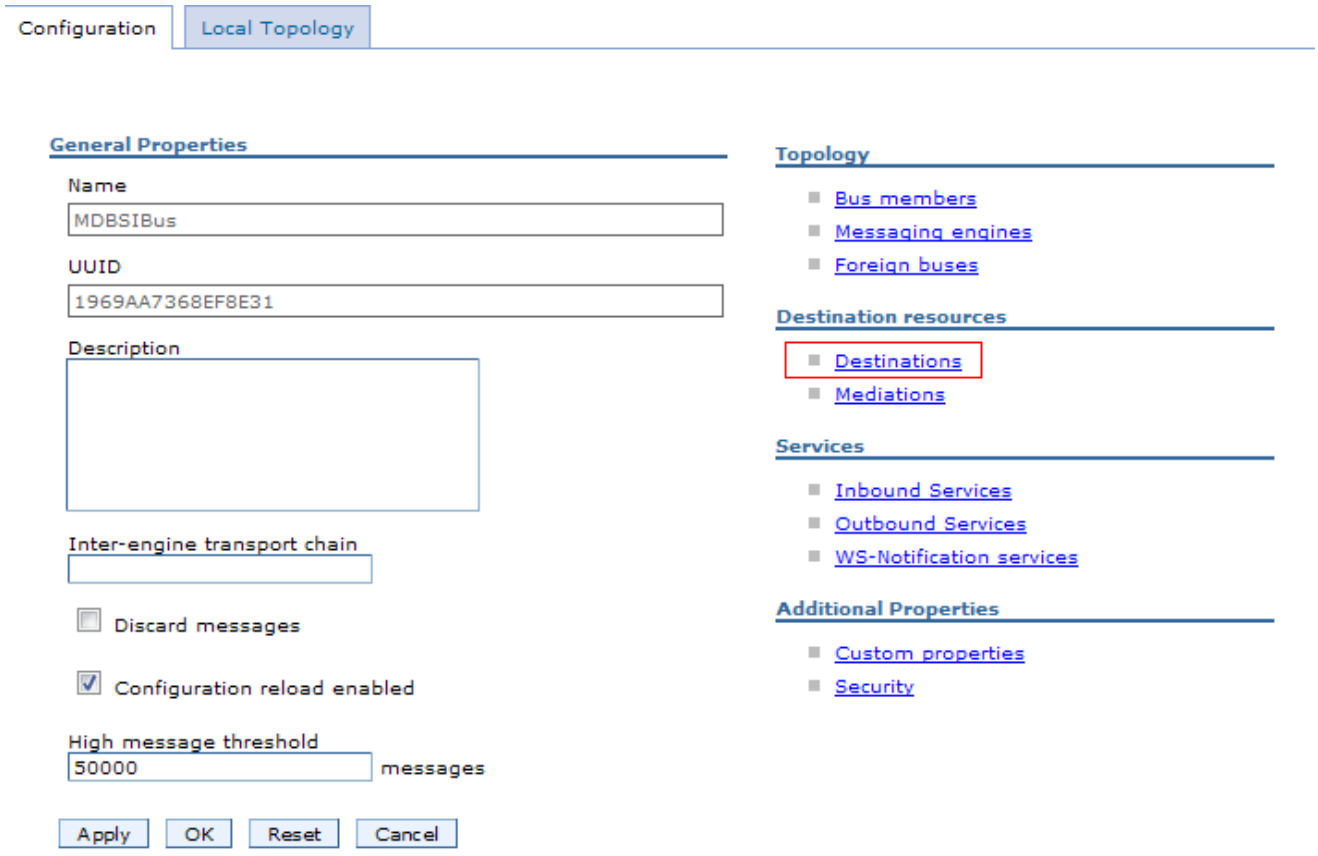


Abbildung 3.1.3-1 Ein Ziel erstellen

Zurück zum MDBSIBus-Panel, Destination auswählen (Abb. 3.1.3-1). 1k auf New.

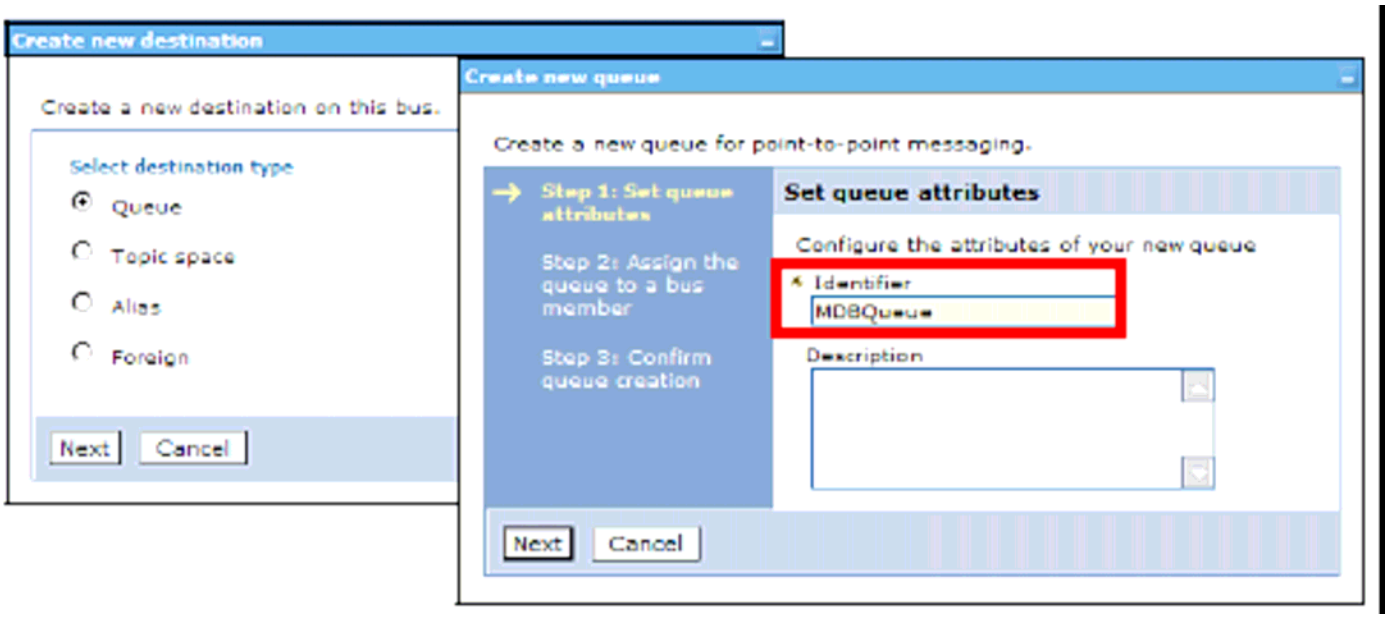


Abbildung 3.1.3-2 Ein Ziel erstellen

Selektiere Queue und klick Next (Abbildung 3.1.3-2).

MDBQueue als Identifier eingeben, und 1k auf Next.

Stellen Sie sicher, dass die Queue dem Bus-Member von MDBSIBus zugeordnet ist und 1k auf Next und Finish.

3.1.4 Konfigurieren des JMS-Provider

Nun beginnen wir mit der Konfiguration des JMS-Providers. In unserem Fall ist dies eine einzige MDB in ihrem EJB Container. Die MDB empfängt eine Nachricht über eine Queue, welche von der in WAS6.1 integrierten MQ-Komponente zur Verfügung gestellt wird. Hierzu definieren wir eine Queue Connection Factory und eine Queue mit dem gleichen Namen wie der JNDI-Name, der in dem Servlet benutzt wird. Die Queue ist Bestandteil der in WAS6.1 integrierten MQSeries-Komponente.

Zusätzlich definieren wir eine JMS-Aktivierungsspezifikationen passend zu der MDB.

Diese werden verwendet, um Message-Queues den MDB-Anwendungen zuzuordnen, welche die MDB-onMessage-Methode implementieren. Die MDB-Aktivierungsspezifikationen ersetzt die in früheren Versionen verwendeten Listener-Ports

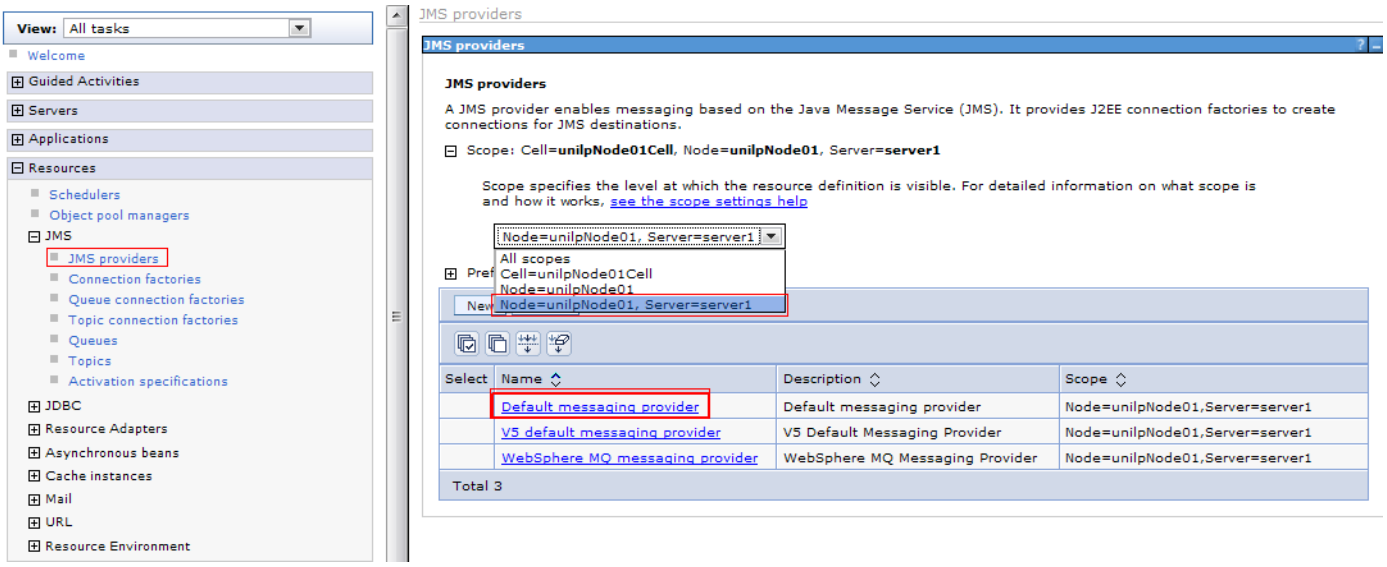


Abbildung 3.1.4-1 Selektiere *Default message provider*

Wählen Sie Ressourcen → JMS → JMS providers.

klicken Sie auf die Standard-Messaging-Provider auf der Server-Ebene. (Abbildung 4.1.4-1)

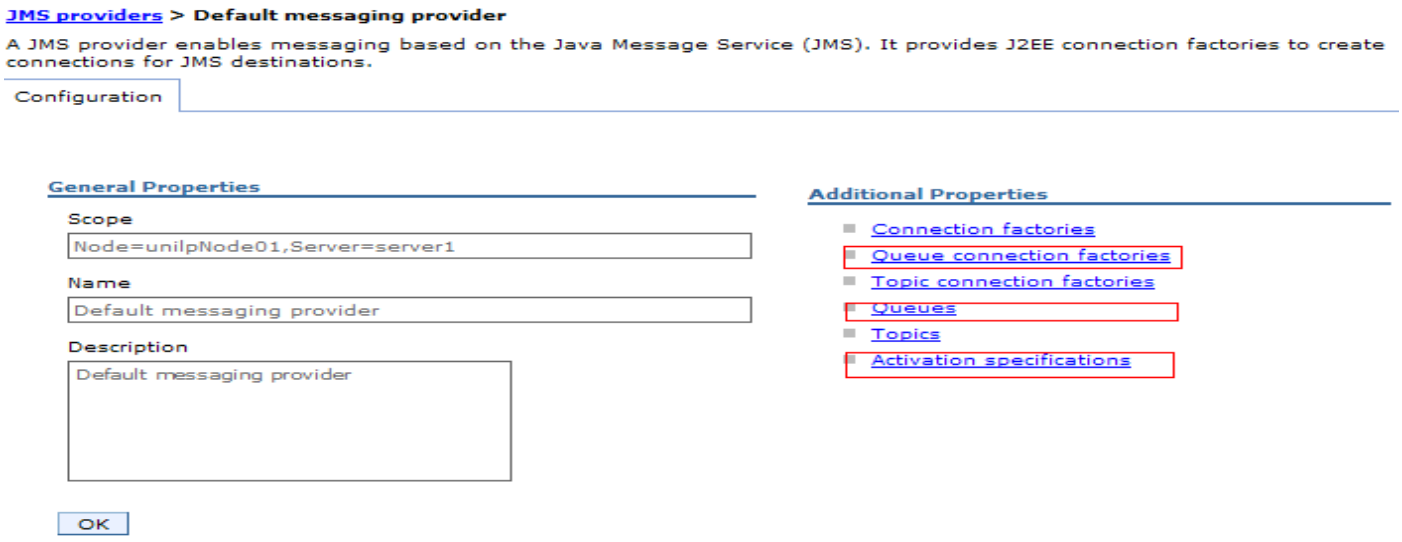


Abbildung 3.1.4-2 Properties Configuration

Im Folgenden werden Queue Connecton Factories, Queues und Activation Specifications konfiguriert (Abbildung 3.1.4.-2)

1k auf Queue Connection Factories.

Selbst-Test

- **Ist ein JMS Provider ein JMS Producer oder ein Message Consumer ?**
- **Gilt dies für dieses Tutorial oder generell ?**

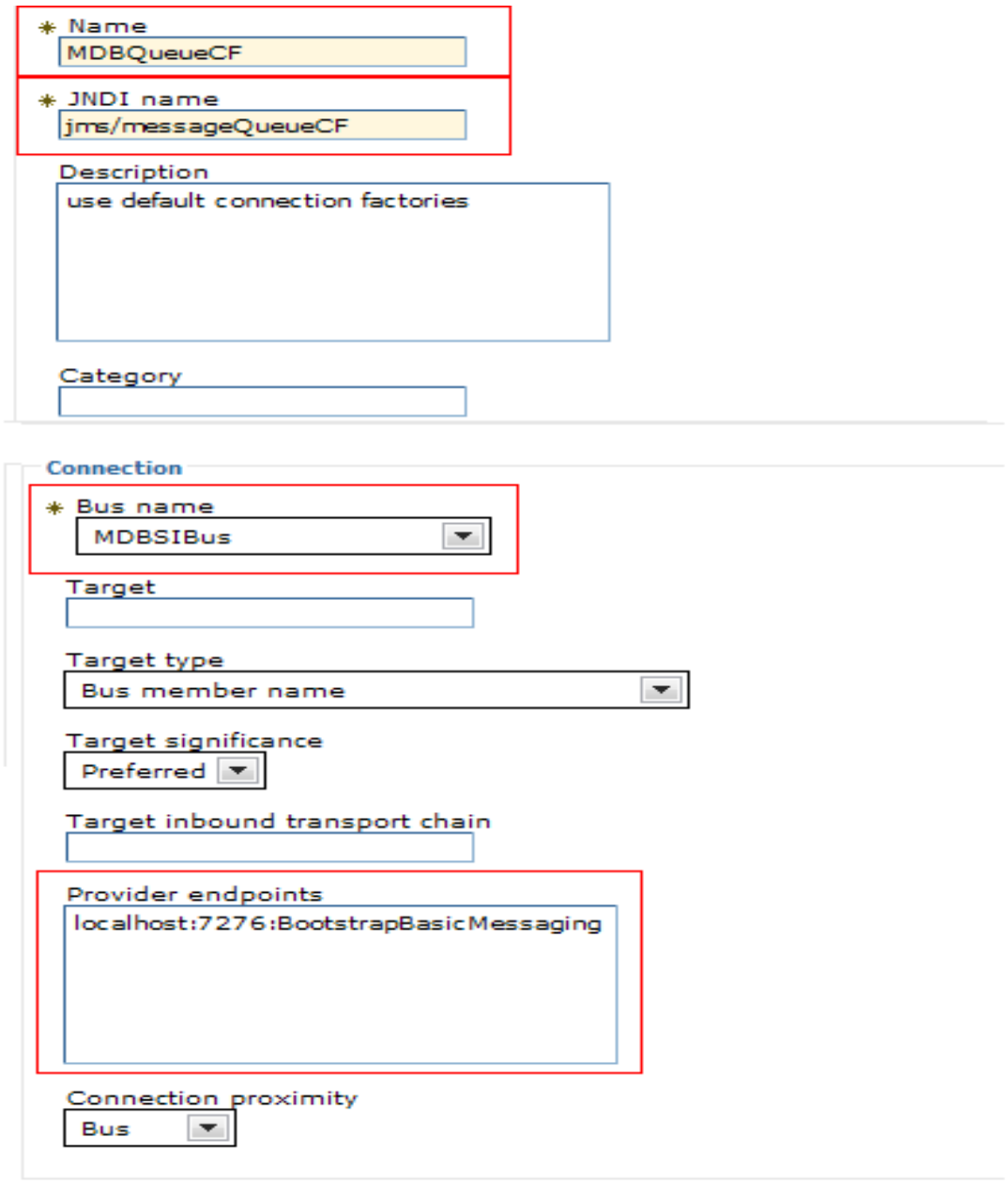


Abbildung 3.1.4-3 Queue Connection Factories Configuration

Klick New. Eingeben MDBQueueCF als Name und jms/messageQueueCF als JNDI Name. Selektiere MDBSIBus als Bus Name.

Als „Provider endpoints“: localhost:7276:BootstrapBasicMessaging eingeben, wobei 7276 der Standard-SIB_ENDPOINT_ADDRESS-Port für den Server ist. Wenn Sie nicht wissen, welchen Port Ihr WAS benutzt, siehe die Datei *<was-home>\AppServer\profiles\AppSrv01\properties\portdef.props*, wobei <was-home> der WAS Installationspfad ist.

1k auf OK (Abbildung 3.1.4-3)

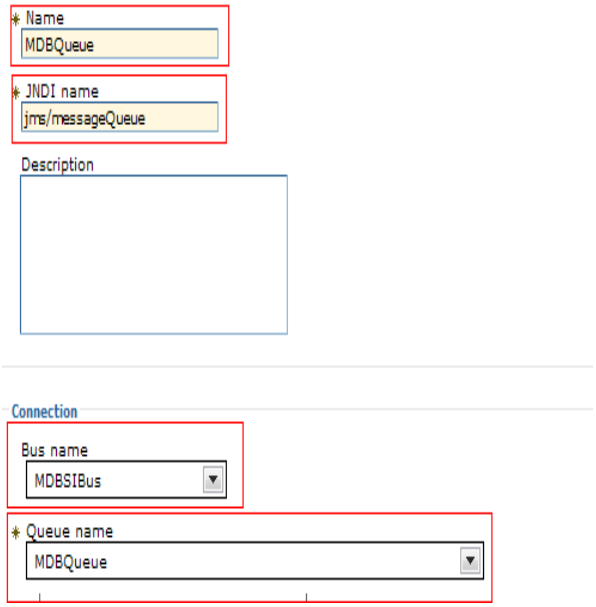


Abbildung 3.1.4-4 Queue Configuration

Zurück zu der *Default Messaging Provider*-Seite, 1k auf Queues. (Abbildung 3.1.4-4)

1k auf New. MDBQueue als Name und jms/messageQueue als JNDI name eingeben. MDBSIBus als Bus Name, MDBQueue als Queue Name auswählen und OK.

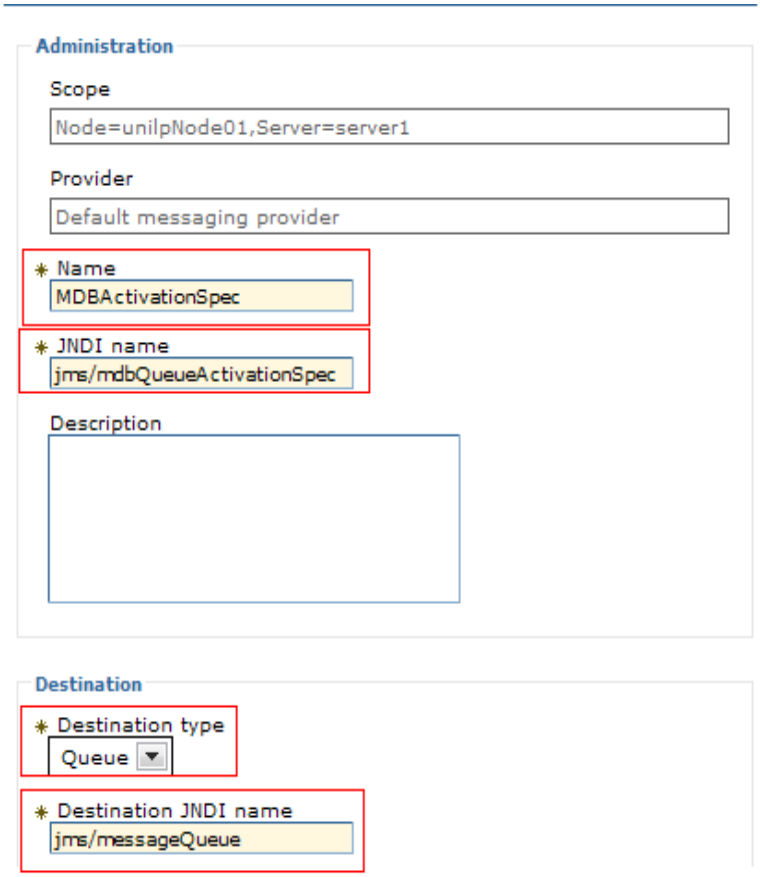


Abbildung 3.1.4-5 Activation specification Configuration

Zurück zu der *Default Messaging Provider*-Seite, 1k auf Activation specifications (Abbildung 3.1.4-5)

1k auf New. MDBActivationSpec als Name und jms/mdbQueueActivationSpec als JNDI Name eingeben. Wählen Sie Queue als Destination Type und geben Sie jms/messageQueue als Destination JNDI Name ein. MDBSIBus als Bus Name auswählen und 1k auf OK.

Den Server neu starten um die Messaging Engine zu aktivieren.

Damit ist die Konfiguration unseres WAS6.1 Application Servers abgeschlossen.

3.2 Erstellen einer Enterprise-Applikation

Wir gehen in diesem Abschnitt ähnlich wie in dem letzten Tutorial vor. Wir erstellen zunächst ein Enterprise Application-Projekt und dessen EAR Deployment-Deskriptor. Dann erstellen wir ein EJB Projekt mit dem Code der MDB und dem dazugehörigen Deployment-Deskriptor. Darauf erstellen wir ein Dynamic Web Project, welches ein Servlet enthält.

3.2.1 Erstellen eines Enterprise Application-Projektes

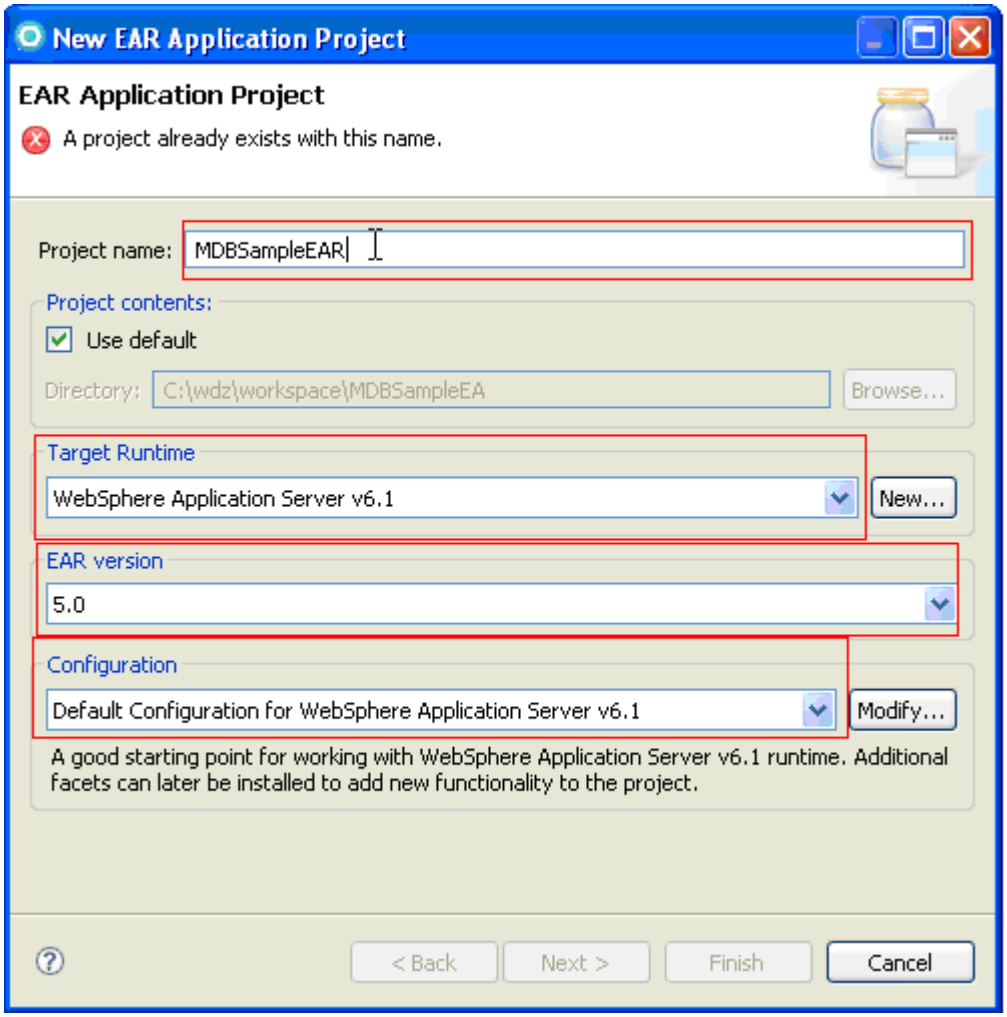


Abbildung 3.2.1-1 Erstellen eines Enterprise Application Projects

Starten Sie RAD7.5 und erstellen Sie ein Enterprise Application Projekt mit dem Namen MDBSampleEAR. Es solle ein EJB Project und ein Dynamic Web Application Projekt enthalten, (siehe Abbildung 3.2.1-1). In dem folgenden Fenster Generate Deployment Descriptor anwählen und 1k auf Finish.

3.2.2 Erstellen eines EJB Projects

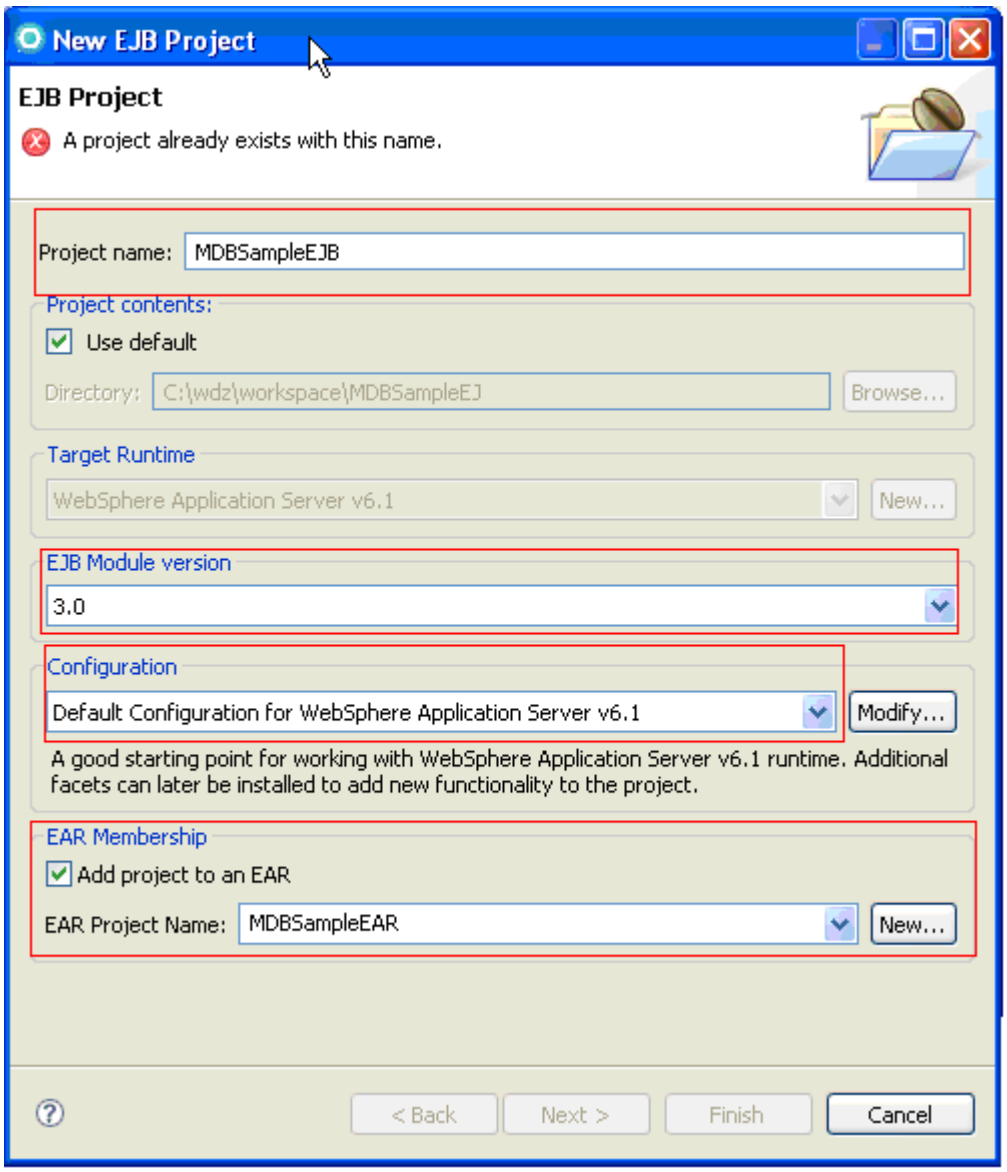


Abbildung 3.2.2-2 Creating an EJB Project

Erstellen Sie ein neues EJB-Projekt mit dem Namen MDBSampleEJB als Teil der MDBSampleEAR Enterprise Application. Wählen Sie 3.0 als die EJB Module Version (Abbildung 3.2.2-1). In dem nächsten Dialog die Option Create an EJB Client Jar module *abwählen* und dann 1k auf Finish.

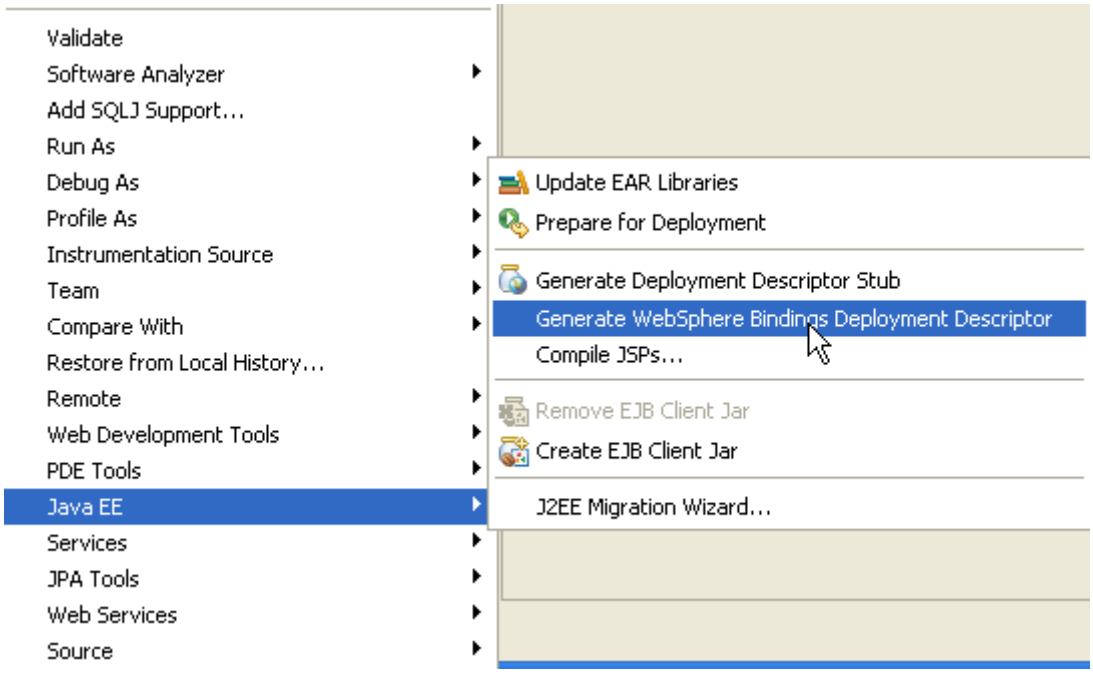


Abbildung 3.2.2-3 Generating ibm-ejb-jar-bnd.xml

1kr auf MDBSampleEJB-Projekt. Selektiere Java EE->Generate WebSphere Bindings Deployment Descriptor um die ibm-ejb-jar-bnd.xml-Datei zu erzeugen (Abbildung 3.2.2-3).

Selbst-Test

- **Stellen die Abbildungen Abbildung 3.2.2-1 bis Abbildung 3.2.2-3 RAD7.5- oder WAS6.1-Fenster dar?**

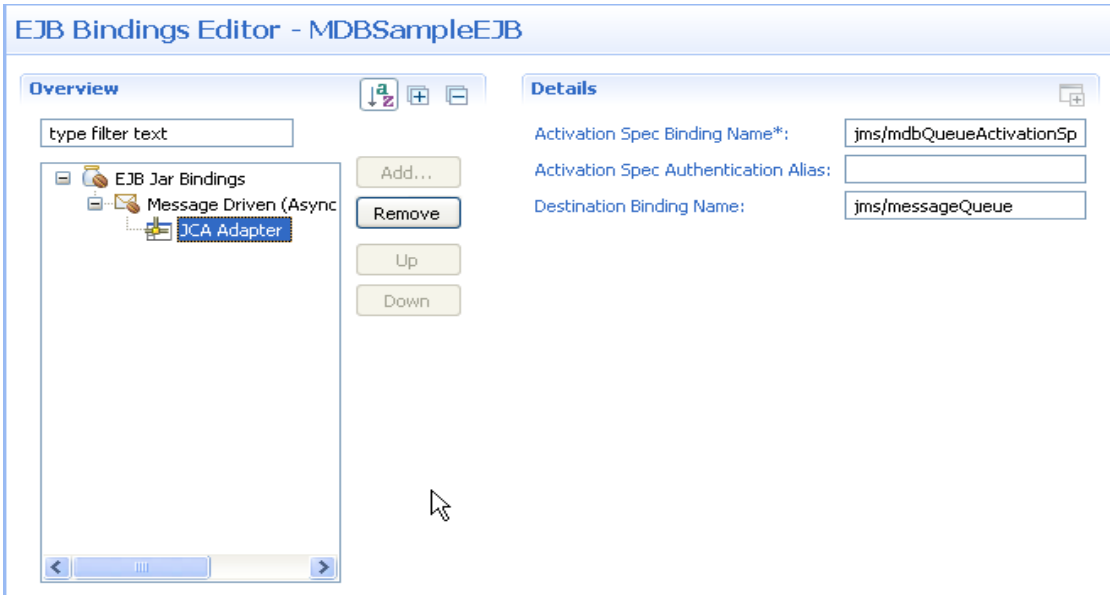


Abbildung 3.2.2-3 MDB Resource Mapping

In dem Design-Tab, können logische Ressourcen erstellt werden, welche die physischen Komponenten abbilden, die vom WAS gemanagt werden.

Klick Add um eine neue Ressource zu erstellen. Wählen Sie Message Driven und 1k auf OK. AsyncMessageConsumerBean als Name eingeben.

Die MDB anwählen, 1k auf Add und JCA Adapter auswählen. Nach dem Hinzufügen des Adapters, jms/mdbQueueActivationSpec als Activation Spec Binding Name eingeben und jms/messageQueue als Destination Binding Name (Abbildung 3.2.2-3).

Verifizieren Sie den erstellten Inhalt in dem Source-Tab. Bestätigen Sie, dass es den folgenden Quellcode entspricht:

```
<?xml version="1.0" encoding="UTF-8"?>
<ejb-jar-bnd
      xmlns="http://websphere.ibm.com/xml/ns/javaee"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://websphere.ibm.com/xml/ns/javaee

        http://websphere.ibm.com/xml/ns/javaee/ibm-ejb-jar-
bnd_1_0.xsd"
      version="1.0">

        <message-driven name="AsyncMessageConsumerBean">
              <jca-adapter activation-spec-binding-
name="jms/mdbQueueActivationSpec"
        destination-binding-name="jms/messageQueue"/>
              </message-driven>

</ejb-jar-bnd>
```

Erstellen Sie eine Bean-Klasse mit dem Namen AsyncMessageConsumerBean im MDBSampleEJB-Projekt, welche eine Nachricht von der Queue empfängt, und TextMessage handhabt.

```
import java.util.logging.Logger;
import javax.annotation.PostConstruct;
import javax.annotation.PreDestroy;
import javax.ejb.MessageDriven;
import javax.ejb.ActivationConfigProperty;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageListener;
import javax.jms.TextMessage;

/**
 * Message-Driven Bean implementation class for: AsyncMessageConsumerBean
 *
 */
@MessageDriven(activationConfig = {
  @ActivationConfigProperty(propertyName = "destinationType", propertyValue =
       "javax.jms.Queue"),
  @ActivationConfigProperty(propertyName = "destination",
      propertyValue = "jms/messageQueue") })
  // a MDB must implements the MessageListene Interface
public class AsyncMessageConsumerBean implements MessageListener {

  Logger logger = Logger.getLogger(AsyncMessageConsumerBean.class.getName());

   @PostConstruct
   void postConstruct() {
           logger.info("PostConstruct called");
   }

   @PreDestroy
   void preDestroy() {
           logger.info("PreDestroy called");
   }

   /**
    * Default constructor.
    */
   public AsyncMessageConsumerBean() {
           // TODO Auto-generated constructor stub
   }
```

```
    /**
     * @see MessageListener#onMessage(Message)
     */
    public void onMessage(Message message) {
            if (!(message instanceof TextMessage)) {
                    logger.info("received a non-TextMessage message; exiting");
                    System.err.println("received a non-TextMessage message; exiting");
                    return;
            }
            TextMessage textMessage = (TextMessage) message;
            try {
                    System.err.println("AsynchBean Received message: "
                                    + textMessage.getText());
                    } catch (JMSException e) {
                    // we can simply ignore it (print it out to the console) for now
                    e.printStackTrace();
            }
    }
}
```

3.2.3 Erstellen eines Dynamic Web Projects

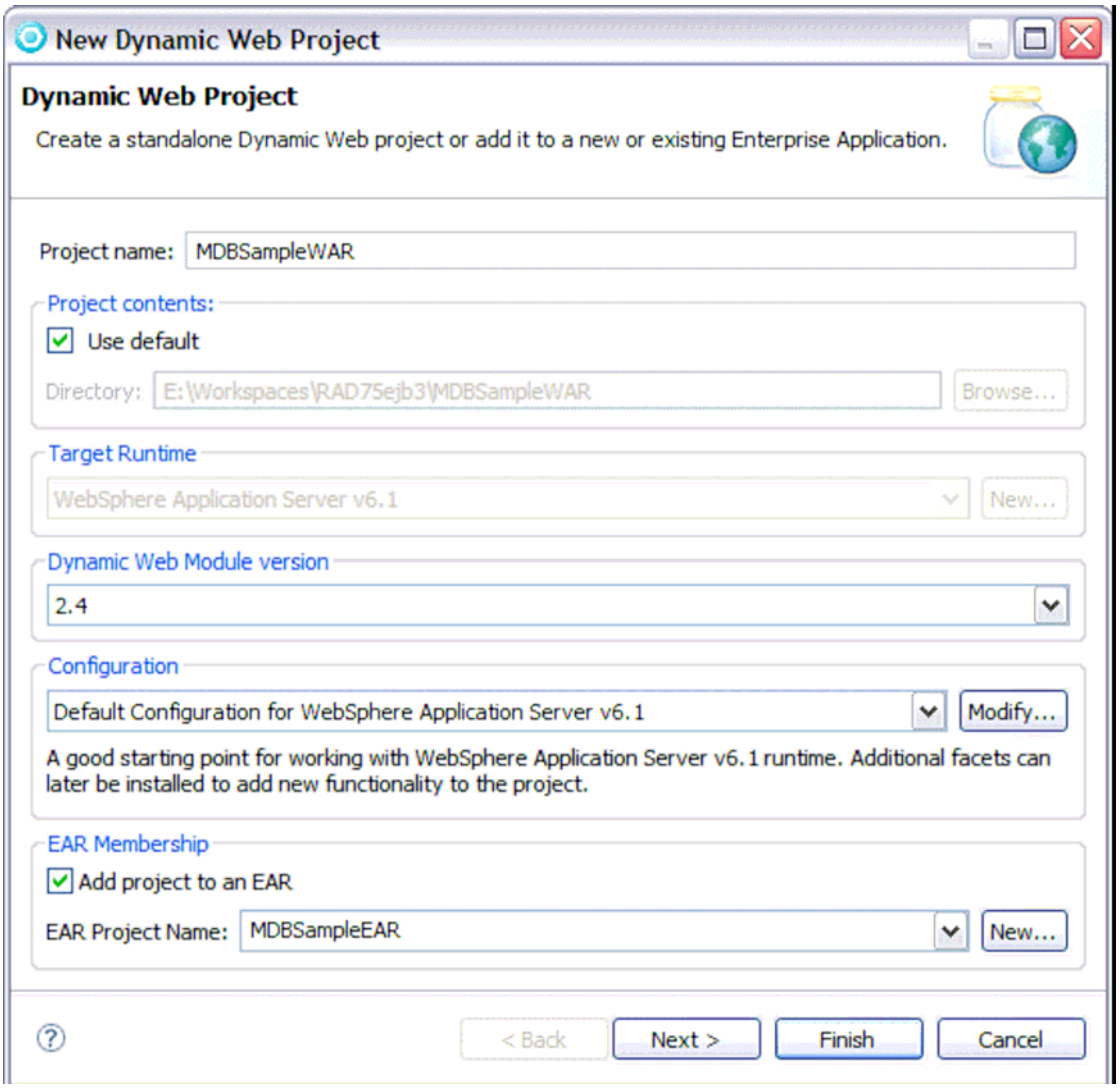


Abbildung 3.2.3-1 Erstellen eines neuen Dynamic Web Projects

Jetzt erstellen Sie ein neues Dynamic Web Project mit dem Namen MDBSampleWAR und fügen es zu der MDBSampleEAR Enterprise Application hinzu. (Abbildung 3.2.3-1).

Öffnen Sie den Web Application Deployment Descriptor.

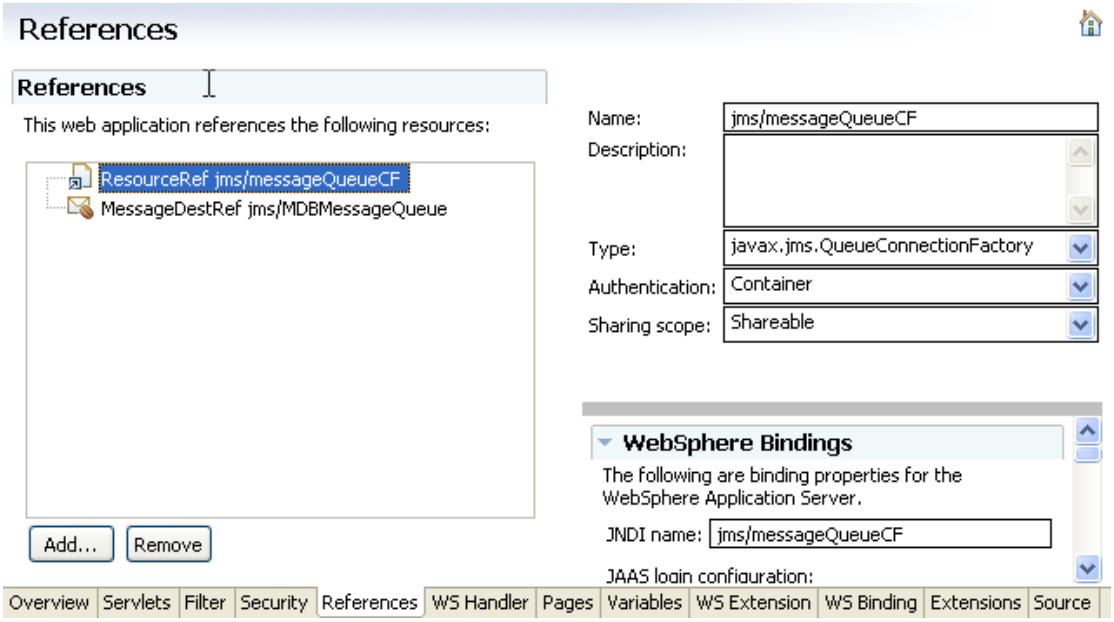


Abbildung 3.2.3-2 Adding a ResourceReference

Auf der References-Seite, 1k auf Add. Selektieren Sie Resource reference und 1k auf Next. jms/messageQueueCF als Namen eingeben. javax.jms.QueueConnectionFactory als Type wählen und Container für Authentication. 1k auf Finish.

Jetzt erscheint die ResourceRef in der Liste. Als JNDI Name jms/messageQueueCF eingeben, unter WebSphere Bindings(Abbildung 3.2.3-2).

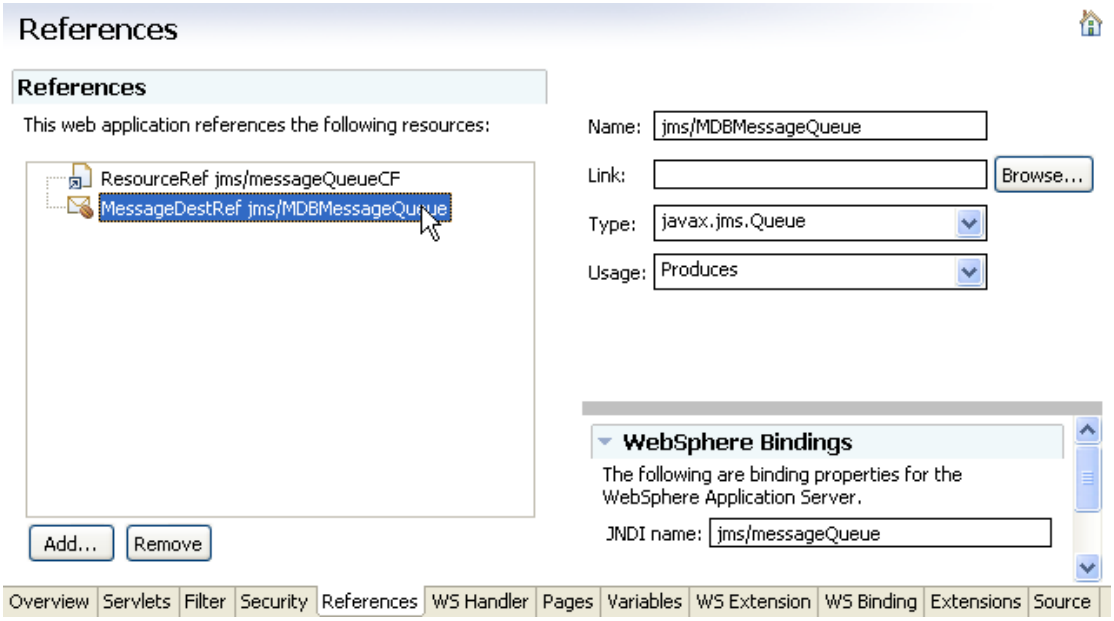


Abbildung 3.2.3-2 Adding a MessageDestinationReference

Klick nochmals Add, Message destination reference auswählen und 1k auf Next. Erstellen Sie eine neue Destination mit dem Namen jms/messageQueue und klick Next. Als Name jms/MDBMessageQueue eingeben, javax.jms.Queue als Type und Produces als usage. 1k auf Finish (Abbildung 3.2.3-3).

Hinweis: **Manchmal funktioniert das GUI-Tool nicht auf Grund von RAD Bugs. Wir müssen manuell den Code "<message-destination-ref> </ message-destination-ref>" zwischen "</ welcome-file-list>" und "</ web-app>" in dem Anwendung Deployment-Deskriptor hinzufügen, um die GUI-Konfiguration zu aktivieren.**

Die MessageDestRef erscheint in der Liste. Als JNDI Name ist jms/messageQueue einzugeben.

Speichern Sie mit Strg+S.

Erstellen Sie ein neues Servlet namens MessageCreaterServlet.java und klicken Sie auf Next. Im nächsten Fenster setzen Sie das URL-Mapping als "/MessageSender". Klicken Sie auf Finish. Dieses Servlet empfängt einen Parameter mit dem Namen "Message" von einer Web-Datei und sendet den Inhalt als TextMessage an die Queue.

Es folgt das Listing des Source Code:

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.annotation.Resource;
import javax.jms.MessageProducer;
import javax.jms.Queue;
import javax.jms.QueueConnection;
import javax.jms.QueueConnectionFactory;
import javax.jms.QueueSession;
import javax.jms.Session;
import javax.jms.TextMessage;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class MessageCreaterServlet extends HttpServlet {
   private static final long serialVersionUID = 18782112020275476 41L;
   // Using the Resource Reference defined in Web Deployment Descript
   @Resource(name = "jms/messageQueueCF")
   QueueConnectionFactory qcf;
   @Resource(name = "jms/MDBMessageQueue")
   Queue queue;

   protected void doGet(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
          performance(request, response);
   }

   protected void doPost(HttpServletRequest request, HttpServletResponse
          response) throws ServletException, IOException {
          performance(request, response);;
   }

   private void performance(HttpServletRequest request, HttpServletResponse
          response) throws ServletException, IOException{
          try {
                 QueueConnection connection = qcf.createQueueConnection();
                 QueueSession session = connection.createQueueSession(false,
                        Session.AUTO_ACKNOWLEDGE);
                 MessageProducer producer = session.createProducer(queue);
                 TextMessage txtMsg = session.createTextMessage();
                 txtMsg.setText(request.getParameter("Message"));
                 producer.send(txtMsg);
                 session.close();
                 connection.close();
                 // for testing purposes
                 PrintWriter out = response.getWriter();
                 out.println("Message is successfully sent.");
                 out.println("QueueConnectionFactory:" +qcf);
                 out.println("QueueConnection: "+ connection);
                 out.println("Queue: " + queue);
                 out.println("MessageProducer: " + producer);
                 out.println("Message Content: " + request.getParameter("Message"));
          } catch (Exception e) {
                 throw new ServletException(e);
          }
   }
}
```

Nachfolgend erstellen wir eine einfache Web Page mit dem Namen SendMessage.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<Form action=MessageSender>
<TABLE width="30%" border="0">
        <TR>
           <TD>Message</TD>
           <TD><INPUT type="text" name="Message"></TD>
        </TR>
</TABLE>
<input type=submit value="Send"/>
</Form>
</body>
</html>
```

Die Projekte in WAS ausführen und testen

3.3.1 Die Projekte unter WAS ausführen

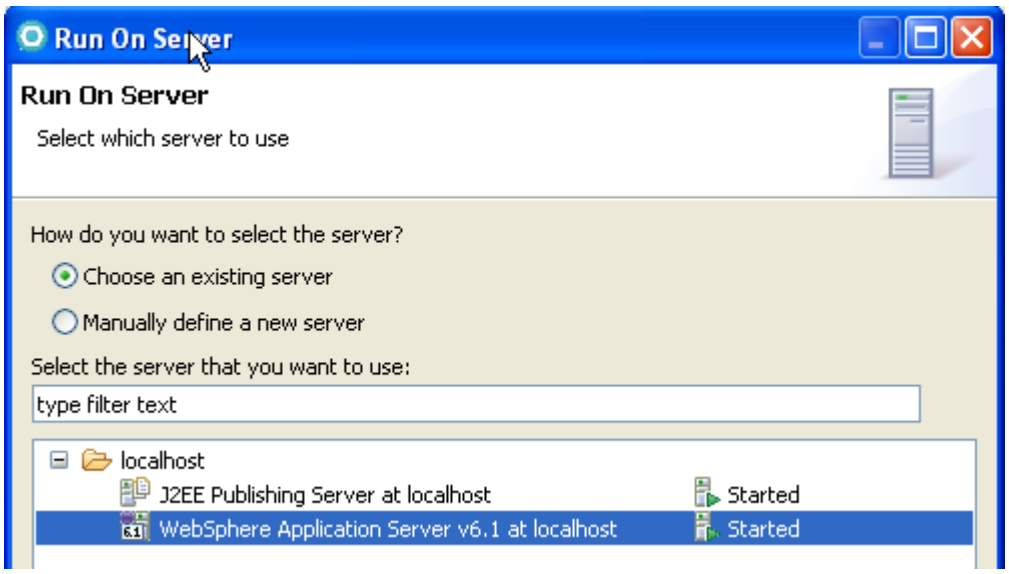


Abbildung 3.3.1

1kr auf das MDBSampleEAR-Projekt, Run As -> Run on Server auswählen, wähle WebSphere Application Server 6.1 at localhost (Abbildung 3.3.1), selektiere die beiden Module MDBSampleEJB und MDBSampleWAR, 1k auf Finish um das Projekt zu starten.

3.3.2 Testen des Projektes

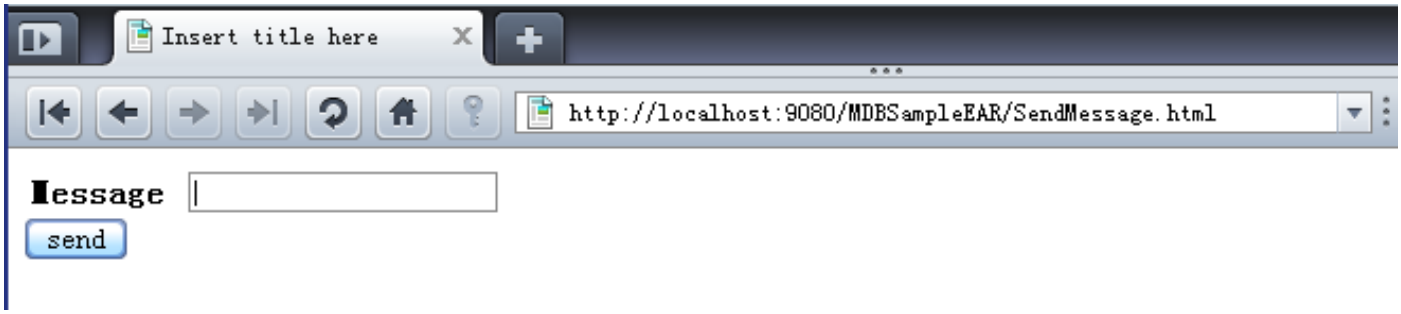


Abbildung 3.3.2-1

Öffnen Sie einen Browser und geben Sie http://localhost:9080/MDBSampleWAR/SendMessage.html in der Adresszeile ein. Die Seite sollte aussehen, wie in Abbildung 3.3.2-1 dargestellt. Geben sie in dem Text Bereich eine beliebige Nachricht ein, z.B. „This is the first message“ und klicken Send.

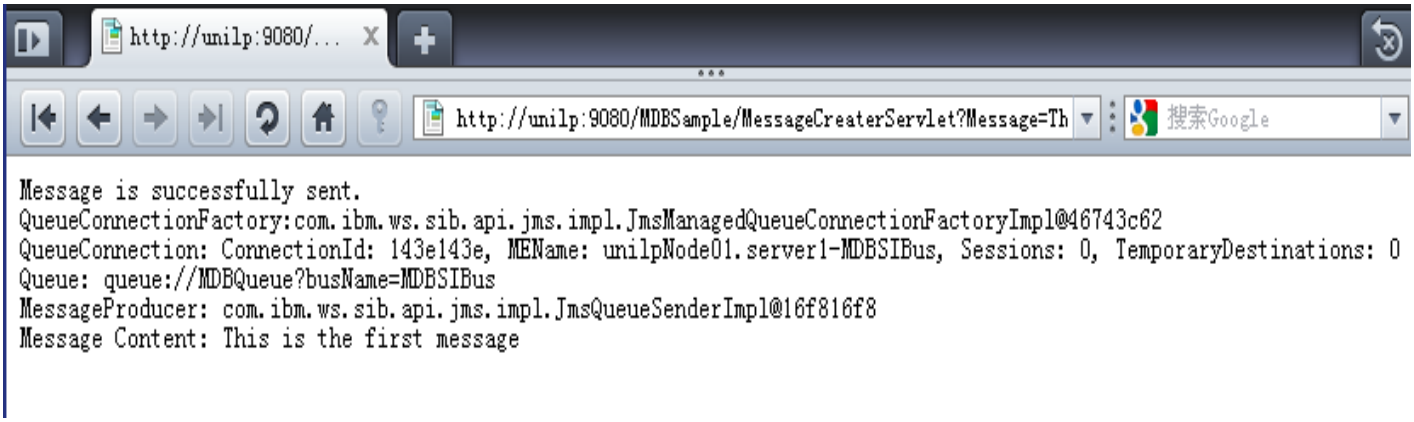


Abbildung 3.3.2-2 Ergebnis

Das Ergebnis sollte aussehen.

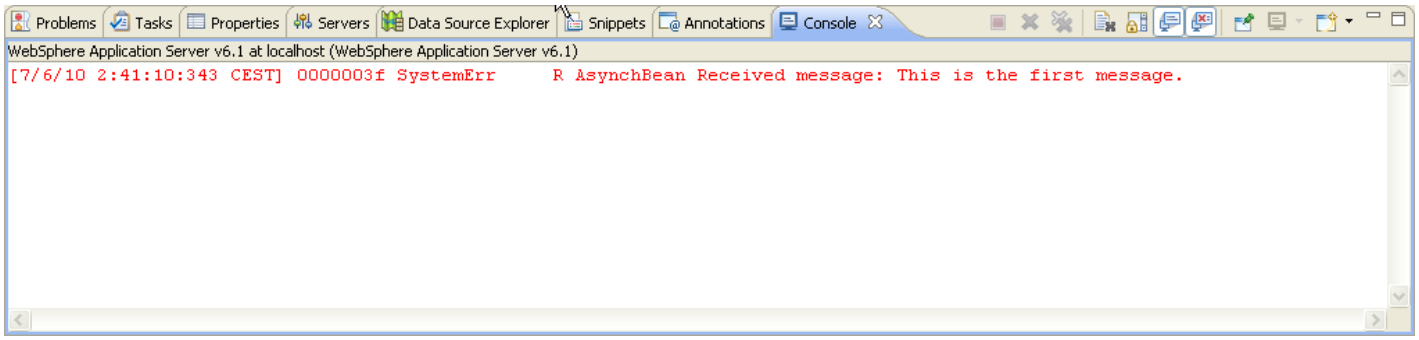


Abbildung 3.3.2-3 Messages in Console

Wechseln Sie zu RD7.5. Das Console-Fenster sollte die in Abbildung 3.3.2-3 gezeigte Nachricht wiedergeben.

Selbst-Test

- **Überlegen Sie sich einige Möglichkeiten: An Stelle nur eine Nachricht in das Eingabefeld zu schreiben, listen Sie 2 – 3 Beispiele, was die Logik in der MDB bewerkstelligen könnte.**

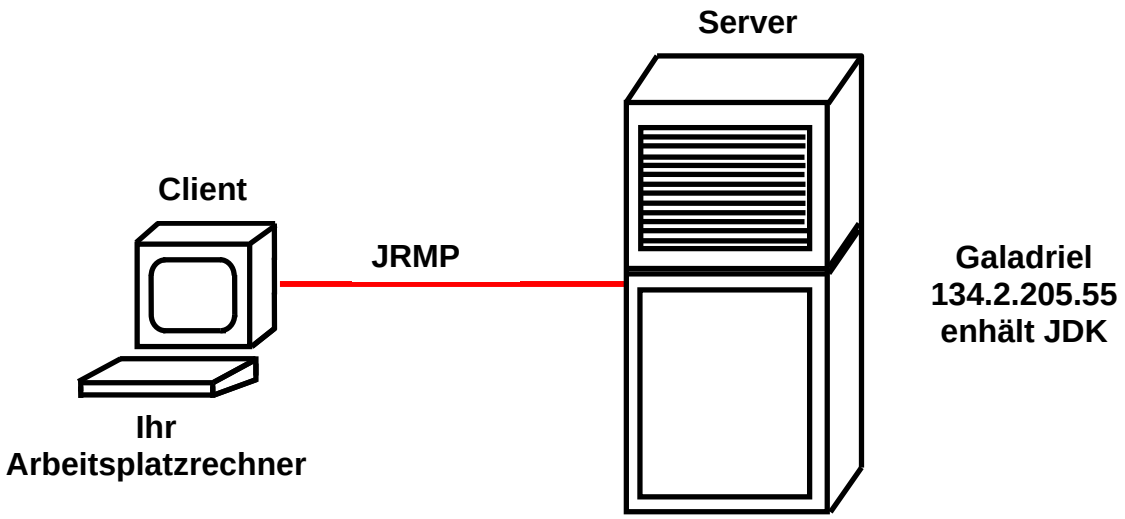
Aufgaben:

1. **Geben Sie in den Message-Text Ihren Namen und ihre PRAK[xxx]-Kennung ein.**
2. **Erstellen Sie anschließend wie in Abbildung 3.3.2-1/2 dargestellt ein Vorher-/Nachher-Screenshot mit Ihren Ergebnissen und laden Sie diese in den Abgabeordner hoch.**

JAVA Remote Method Invocation
JRMP Tutorial

In diesem Tutorial werden Sie ein Java Programm auf einem Klienten (Ihrem PC) installieren und mittels RMI einen Zugriff auf einen Java Server durchführen, der in einer zLinux LPAR auf dem z9 Mainframe in Tübingen läuft. Die zLinux LPAR hat den Namen galadriel.cs.uni-tuebingen.de (oder 134.2.205.55).

Danksagung an Herrn Robert Harbach für die Bereitstellung des Materials.



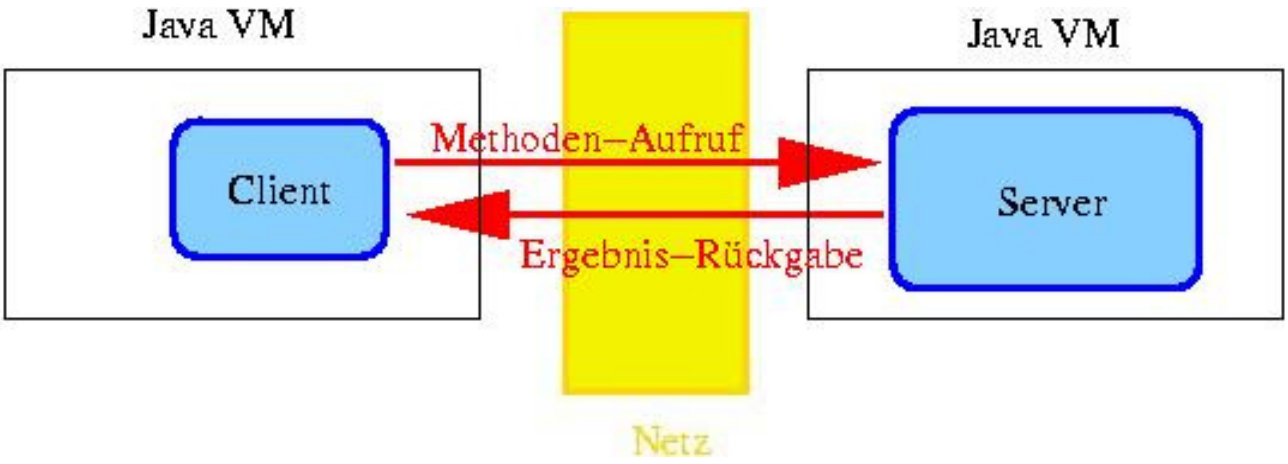
Sowohl auf dem Client als auch auf dem Server ist das Java Development Kit (JDK) installiert. Die Client-Java-Klassen kommunizieren über das Internet mit den Server-Java-Klassen über RMI. Hierfür verwenden sie das Java Remote Message Protokoll (JRMP). JRMP setzt voraus, dass sowohl die Client- als auch die Server-Anwendung in Java programmiert ist.

In einer modernen Implementierung würde man hier einen Web Application Server einsetzen. Die beiden JDKs wären Bestandteil von dessen Servlet-, JSP- und EJB-Infrastruktur. In diesem Tutorial verzichten wir auf diese zusätzliche Komplexität. Wir werden dies in einem späteren Tutorial unter Benutzung des modernen EJB 3.0 Standards nachholen.

Übersicht

1. Das RMI Programmiermodell

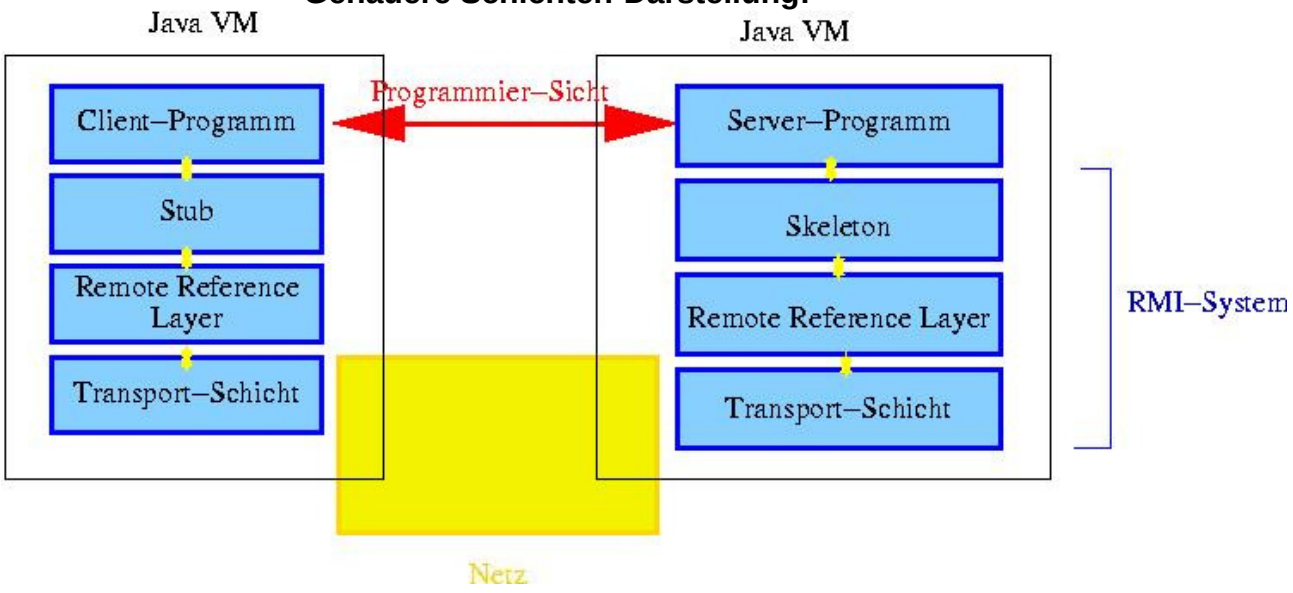
1.1 Java Remote Procedure Call



Normalerweise erfolgen Java-Methodenaufrufe innerhalb einer einzelnen JVM. Java-Objekte in einer JVM können jedoch auch RMI-Methoden von Objekten in einer entfernten JVM aufrufen. Dabei können diese JVMs auf verschiedenen Rechnern im Netz laufen. (Auch unterschiedliche JVMs auf dem gleichen Rechner kommunizieren über RMI.)

Realisierung: Erzeugen eines Stellvertreters (Client-Stub) des entfernten Objekts (Server-Skeleton) in der entsprechenden lokalen JVM. Stub und Skeleton kommunizieren miteinander.

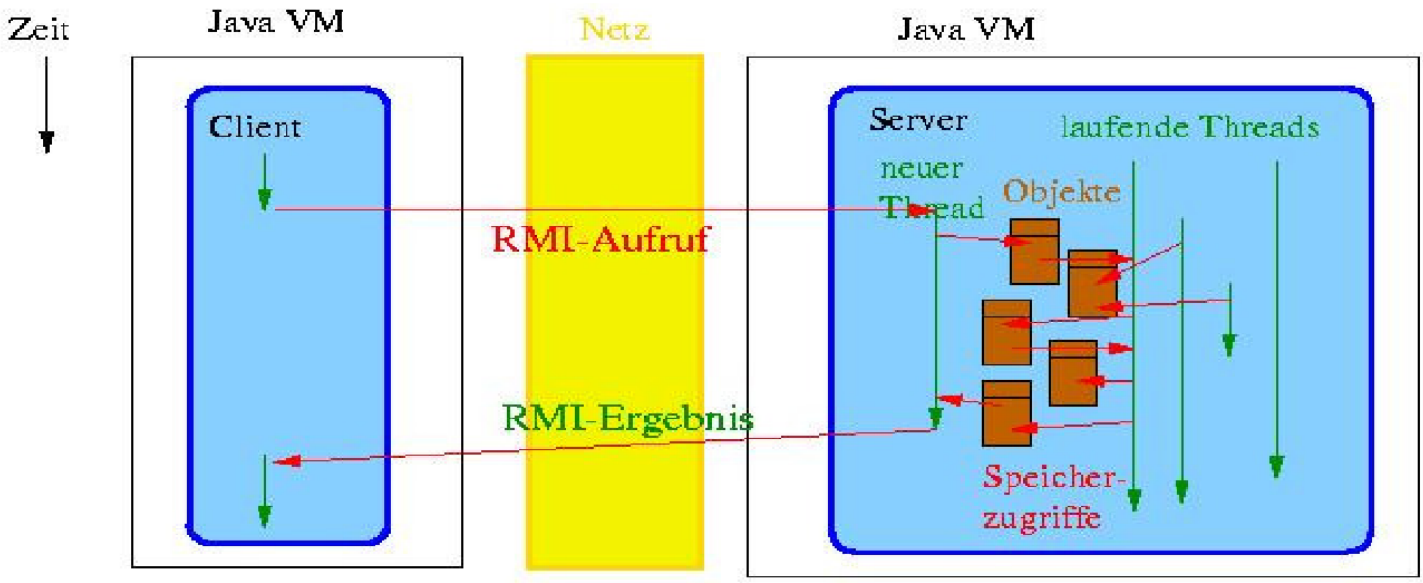
Genauere Schichten-Darstellung:



Der Remote Reference Layer muss entfernte Referenzen (Referenzen auf entfernte Objekte) auf Rechnernamen und Objekte abbilden.

1.2 Threads auf der Server-Seite

Jeder RMI-Aufruf erzeugt auf der Server-Seite (dort, wo die Funktion ausgeführt wird; Skeleton) einen neuen Thread. Das bedeutet, dass dort mehrere solcher Threads gleichzeitig laufen können, zusammen mit evtl. weiteren dauerhaft laufenden Threads des Servers (z.B. Garbage Collector, Compute-Server,...)



Man muss sich also in jedem Fall Gedanken über eine notwendige Synchronisation machen (mit synchronized-Blöcken, ggf. auch mit wait() und notify()).

2. RMI Programmierung

2.1 Übersicht

Auf dem Moodle Server wird im Script z/OS Internet Integration, im Thema 6 „RMI“ unter Teil 4 ein Beispiel für eine einfache RMI Programmierung gezeigt. Allerdings befinden sich hier Client und Server auf dem gleichen Rechner.

Wir werden ein Java-Programm auf einem Klienten (Ihrem PC) installieren und mittels RMI einen Zugriff auf einen Java-Server durchführen, der in einer zLinux LPAR auf unserem z9 Mainframe in Tübingen läuft.

Wir benötigen hierfür 4 Java-Programme, drei davon für den Server

- **Java-Server**
- **Java-Service, auch als Implementation bezeichnet**
- **Java-Interface (des Java Service)**

sowie zusätzlich ein Java-Client.

Wir stellen Ihnen diese Java-Programme vorgefertigt zur Verfügung. Sie dürfen gerne stattdessen eine modifizierte oder andere Version benutzen.

Der Java-Server läuft als Prozess auf einer zLinux LPAR. Normalerweise würden unter einem Java-Server viele Java-Services laufen. In unserem Fall ist dies jedoch nur ein einziger Service. Es ist deshalb in diesem Fall möglich, den Java-Service und den Java-Server zu einem einzigen Java Programm zu kombinieren.

Die drei Java-Dateien, die wir Ihnen zur Verfügung stellen, haben die Namen

Konto.java	**Interface**
KontoImpl.java	**Kombination von Java-Server und Java-Service (Implementation)**
Terminal.java	**Java-Client**

2.1.1 Policy

Das Java-2-Sicherheitsmodell verlangt eine so genannte Policy. Eine Policy besteht aus Regeln, die jeweils Berechtigungen gewähren. Alles, was nicht ausdrücklich in der Policy gestattet wird, ist nicht erlaubt.

Standardmäßig werden Policies in Dateien gespeichert. Die JRE versucht beim Start eines Programms Policy-Dateien auszulesen:

Wir stellen zusätzlich zu den drei Java-Dateien eine einfache Policy mit dem Namen *security.policy* zur Verfügung. Diese Policy ist am einfachsten, aber auch am unsichersten. Der Inhalt der Datei ist:

```
grant {
     permission java.security.AllPermission;
};
```

Einen Überblick über mögliche Permissions:
http://java.sun.com/j2se/1.4.2/docs/guide/security/permissions.html.

Weitere Beispiele für Policy-Files:
http://java.sun.com/j2se/1.4.2/docs/guide/security/PolicyFiles.html#Examples.

Zur Erstellung der Datei java.policy kann auch das GUI-basierte policytool aus dem JDK verwendet werden. Dokumentation:
http://java.sun.com/j2se/1.4.2/docs/tooldocs/solaris/policytool.html.

2.1.2 Interface java.rmi.Remote

Entfernte Methoden werden durch das Interface java.rmi.Remote definiert: Eine RemoteException kann z.B. auftreten, wenn das entfernte Objekt nicht mehr verfügbar ist oder wenn die Verbindung gestört ist.

2.1.3 Service Implementierung durch java.rmi.server.UnicastRemoteObject

Server-Programme implementieren das Interface Remote und erweitern `java.rmi.server.UnicastRemoteObject`, welches die wichtigsten Methoden für die Verwendung von RMI bereitstellt.

In unserem Beispiel bewirkt der Aufruf

```
public class KontoImpl extends UnicastRemoteObject implements Konto
```

die Implementierung der *Schnittstelle Konto.java*.

2.1.4 rmiregistry

Wie bekommt ein Client Zugriff auf ein Hello-Objekt?

Wir wissen: Stub für Client, Skeleton für Server.

- **Der Server muss den Skeleton unter einem (eindeutigen) Namen bei einer rmiregistry anmelden. Diese rmiregistry muss auf demselben Rechner laufen, auf dem das Remote-Objekt (Skeleton) ausgeführt wird!**
- **Der Client muss**
  1. die URL der rmiregistry kennen, von der er einen Stub des Objekts bekommen kann (z.B. rmi://galadriel.cs.uni-tuebingen.de:2012)
  2. den Namen des Objekts kennen, unter dem der Stub bei der Registry bekannt ist.

2.2 Ihre Aufgabe: Eine verteilte Anwendung mit RMI

2.2.1 Szenario

Eine Bank will ihr System komplett auf Java umstellen. Ein Server soll mindestens ein Konto halten. Von Terminals aus soll Geld eingezahlt, abgehoben und der Kontostand abgefragt werden können.

2.2.2 Code Beispiel

Quellcode Beispiele für drei Dateien

- **Konto.java,**
- **KontoImpl.java,**
- **Terminal.java,**

sowie eine Security-Policy-Datei sind im Anhang wiedergegeben:

Server-Seite:

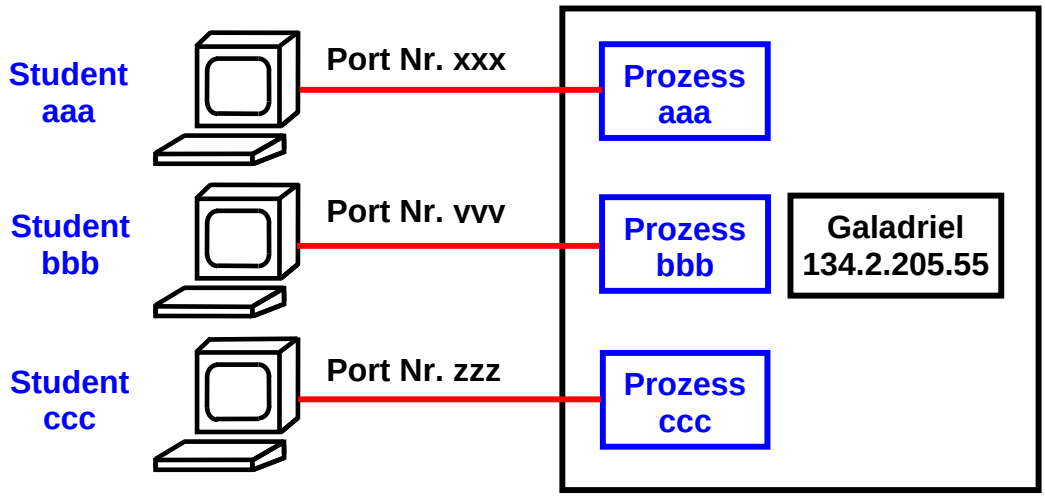
- **`interface Konto extends Remote:`**
  - **beschreibt Methoden zum Einzahlen/Abheben und Abfragen des Kontostandes.**
- **`class KontoImpl implements Konto extends UnicastRemoteObject:`**
  - **Konto zur entfernten Verwendung. KontoImpl hält das aktuelle Guthaben und implementiert die in Konto spezifizierten Methoden.**
  - **Enthält eine main()-Methode, die den RMISecurityManager erzeugt und installiert, falls er nicht schon installiert ist und das entfernte Objekt bei der rmiregistry registiert ist.**

Client-Seite:

- `class Terminal:`
  - holt sich eine entfernte Referenz auf das Konto Objekt.
  - Hält explizit nicht den Kontostand!!! (Das übernimmt die Referenz auf Konto)
  - Hat eine main()-Methode, die eine Reihe von Methodenaufrufen durchführt und eine sinnvolle und lesbare Ausgabe auf die Konsole ausgibt und so die Korrektheit Ihrer Anwendung zeigt.

2.2.3 Port Nummer

Server-Anwendungen erhalten Nachrichten über Ports. Das obige Beispiel verwendet Port Nr. 2012.



Eine Anzahl von Studenten wird gleichzeitig dieses Tutorial bearbeiten. Wir benutzen auf Galadriel aber keine Middleware wie z.B. WebSphere. Stattdessen startet jeder Benutzer auf Galadriel seinen eigenen Server-Prozess. Damit braucht jeder Benutzer für seinen Server-Prozess eine getrennte Port Nummer.

Wir schlagen vor, dass Sie eine Port Nr. im Bereich zwischen 50001 und 50999 wählen.

Um Konflikte mit anderen Benutzern zu vermeiden, schlagen wir vor, dass Sie als Port Nr. 50xxx wählen, wobei xxx die drei letzten Ziffern Ihrer prakxxx User ID sind. Wenn Sie also die User ID prak519 benutzen, wäre Ihre Port Nr. 50519.

Beachten Sie, dass die obigen Beispielprogramme Port Nr. 2012 benutzen. Sie müssen diese, und auch überall sonst in diesem Text, durch Ihre eigene Port Nr. ersetzen.

2.2.4 Vorgehen

2.2.4.1 Benötigter Quellcode

Erstellen Sie ein eigenes leeres Verzeichnis auf Ihrem Rechner, z.B. rmi, und erstellen Sie die 3 Java-Dateien:

- **Konto.java**
- **KontoImpl.java**
- **Terminal.java**

sowie die security.policy-Datei in diesem Verzeichnis.

Hinweis:

Eine rudimentäre Implementierung, welche obigen Anforderungen genügt, reicht aus. Auf PIN, Login, mehrere Kontos pro Server, Setzen/Prüfen der Kreditlinie, Zinsen usw. kann verzichtet werden.

Wenn Sie wollen, können Sie dies natürlich trotzdem implementieren.

2.2.4.2 Das Interface und die Klassen kompilieren

Sie müssen die Klassen

- **Konto.class**
- **KontoImpl.class**
- **Terminal.class**

mit Hilfe des `javac`-Compilers erzeugen. Hierfür brauchen Sie eine Java Entwicklungsumgebung. Sie können (für die primitive Aufgabenstellung ausreichend) hierfür die JDK benutzen, oder eine komfortablere Entwicklungsumgebung wie Eclipse einsetzen.

Die normale Vorgehensweise besteht darin, dass Sie eine passende Entwicklungsumgebung auf Ihrem Arbeitsplatzrechner installieren, dort den Java-Code übersetzen, Skeleton- und Stub-Klassen erzeugen, und diese – soweit sie den Server betreffen – auf diesen kopieren. Da wir nicht mit Servlets und EJBs, sondern mit einfachen Java-Klassen arbeiten, ist ein einfaches Kopieren ausreichend. (Servlets und EJBs würden eine Installation auf einem Web Application Server mittels .jar, .war und/oder .ear Files erfordern.)

Ein einfacher Vorgang? Not really.

2.2.4.3. Exkurs zum Problem der Java Kompatibilität

Java als Programmiersprache entstand in 1995, hat sich aber seitdem nicht richtig stabilisiert. Zahlreiche Probleme bleiben nur unbefriedigend gelöst, besonders im Zusammenhang mit der Isolation von Java Threads und der Zustandsänderung der JVM. Siehe hierzu die Diplomarbeit von Jens Müller, http://www-ti.informatik.uni-tuebingen.de/~spruth/DiplArb/jmueller.pdf

Java ist in mehreren Versionen verfügbar. Version 1.4 erschien 2002, Version 5 erschien 2004, Java Version 6 erschien 2006, Java Version 7 erschien 2011 und Java-Version 8 wird für 2013 erwartet. Statt der Bezeichnung Version 5, 6 oder 7 werden auch die Bezeichnungen 1.5, 1.6 und 1.7 verwendet. Im Gegensatz zu Sprachen wie C++ und Cobol ist die Kompatibilität der einzelnen Versionen untereinander problematisch.

Wegen der Kompabilitäts- und Stabilitätsprobleme führen Firmen wie IBM neuere Versionen nur sehr zögernd ein. Unser Rechner Galadriel verwendet Version 1.4. Wenn Sie heute (2011) eine JDK herunterladen (z.B. von http://www.java.com/de/download/), erhalten Sie Version 1.7.

Wenn Sie auf Ihrem Arbeitsplatzrechner unter Verwendung einer Version 1.7 JDK Klassen für den Server erzeugen, sind diese auf dem Server unter Version 1.4 nicht lauffähig. Die gute Nachricht ist, dass unter Version 1.4 erzeugte Klassen auf einem Rechner mit Version 1.7 ausführbar sind.

Für dieses Problem gibt es zwei Lösungen. Die erste Alternative besteht darin, dass Sie für die Kompilierung auf Ihrem Arbeitsplatzrechner Java Version 1.4 benutzen, und anschließend die Server Klassen nach Galadriel kopieren.

Eine zweite Alternative besteht darin, dass Sie die Kompilierung und Erstellung von Skeleton und Stub Klassen auf Galadriel mit Version 1.4 durchführen, und die für den Klienten erforderlichen Klassen dann von Galadriel auf Ihre Workstation runterladen. Diese Klassen sind dann auf Ihrem Arbeitsplatzrechner auch unter neueren Java Versionen, einschließlich Version 1.7, ausführbar.

Wir verwenden die 2. Alternative. Es steht Ihnen aber frei, mit der hier nicht beschriebenen ersten Alternative zu arbeiten. Falls Sie Schwierigkeiten haben eine passende JDK der Version 1.4 für Ihren Windows Rechner zu finden, können Sie sie unter

http://www.cedix.de/VorlesMirror/Band3/j2sdk-1_4_2_02-windows-i586-p-iftw.exe.

herunterladen. Sie wurde unter Windows XP ausgetestet.

Und wenn Sie dieses Disaster mit den verschiedenen Versionen ärgert, verstehen Sie etwas besser, warum z/OS so großen Wert auf Auf- und Abwärtskompatibilität legt.

2.2.2.4. Stubs und Skeletons mit rmic erstellen.

Mit Hilfe der Schnittstellenbeschreibung ist es möglich, Stub und Skeleton zu erstellen. Dies geschieht mit Hilfe des `rmic`-Compilers. Das Java-Interface (in diesem Beispiel Konto) Ihres Service-Implementierungsprogramms enthält alle Informationen, die der Compiler benötigt.

Den rmic-Compiler können Sie mit

```
rmic KontoImpl
```

aufrufen. Als Ergebnis sollten Sie die Dateien

KontoImpl_Skel.class
KontoImpl_Stub.class

erhalten. Die so entstandenen 5 class-Dateien

Konto.class
KontoImpl.class
Terminal.class
KontoImpl_Skel.class
KontoImpl_Stub.class

sind das Ergebnis Ihrer Programmiertätigkeit.

Als Nächstes müssen diese Dateien jetzt installiert, und dann ausgeführt werden.

3. Erstellen der Java Klassen auf dem zLinux Server

3.1 Laden der Klassen auf den zLinux Server

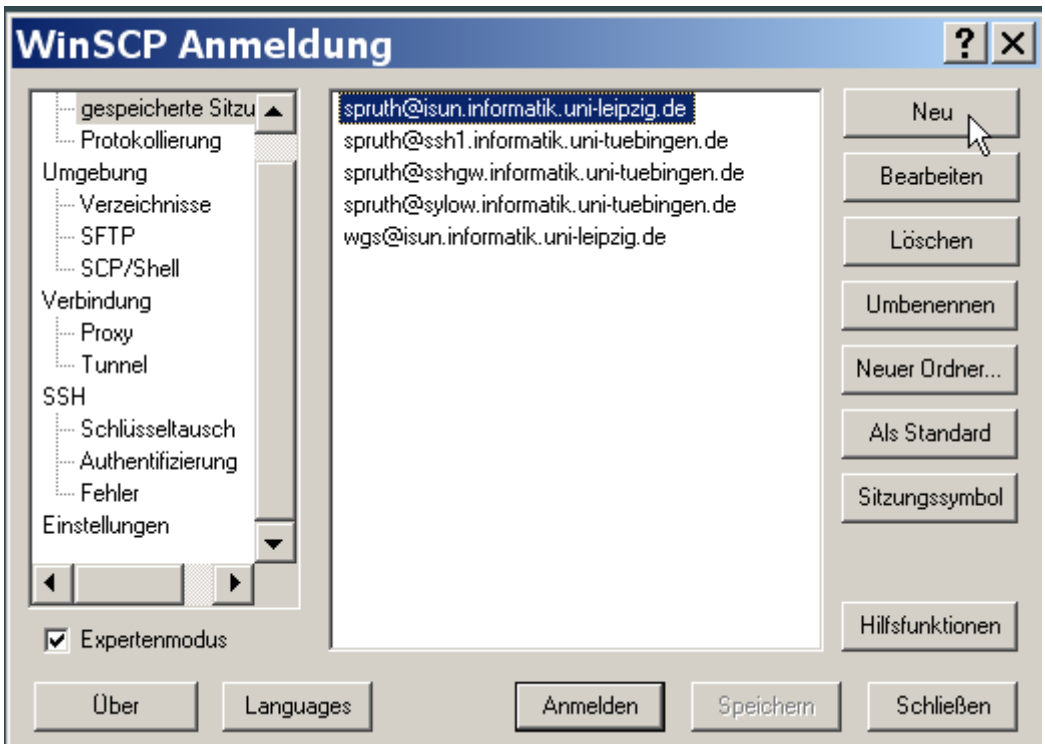
Diese 4 Dateien

- **Konto.java**
- **KontoImpl.java**
- **Terminal.java**
- **security.policy**

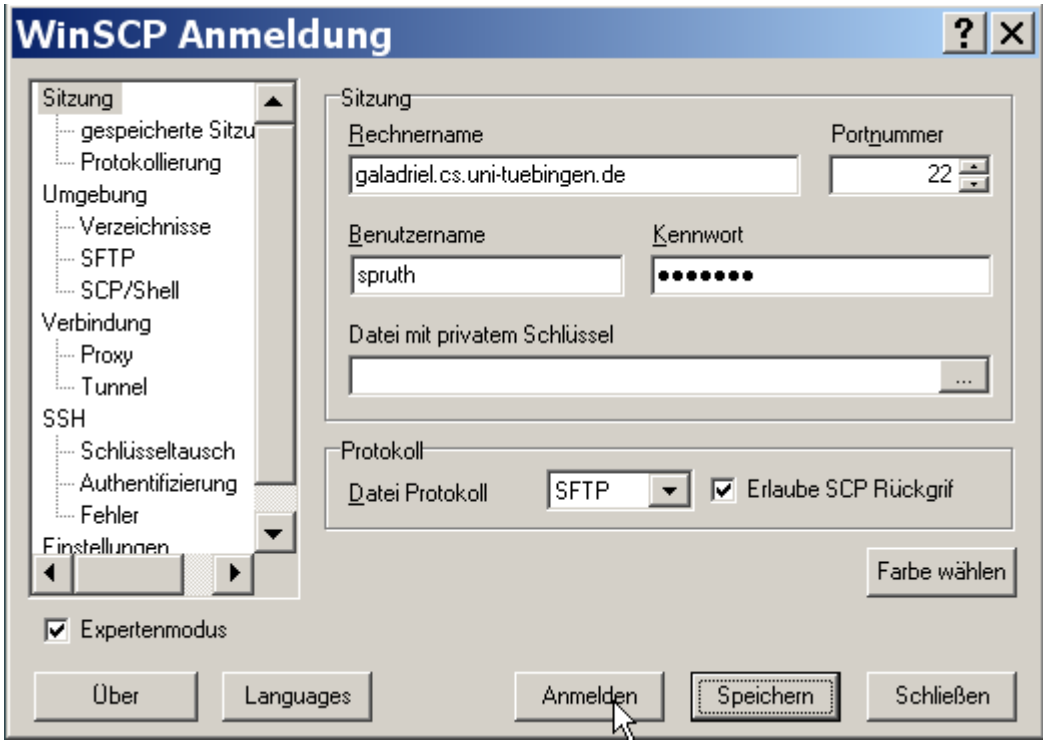
müssen nun auf dem Server galadriel.cs.uni-tuebingen.de (oder 134.2.205.55) geladen werden. Bitte bedenken sie, dass dies eine andere LPAR ist, als die von Ihnen für das letzte Tutorial verwendete LPAR leia.informatik.uni-leipzig.de.

Das Laden geschieht am Einfachsten mit WinSCP (oder FileZilla). WinSCP ist ein Open-Source- SFTP-Client für Windows. Falls noch nicht vorhanden, laden Sie sich WinSCP aus dem Internet herunter, z.B. von http://winscp.net/eng/docs/lang:de, und installieren Sie WinSCP auf ihrem Rechner.

Wir arbeiten in diesem Tutorial mit einfachen Java Klassen, keinen Servlets und EJBs. An Stelle der komplexeren Web Application Server Installation mit .jar und .ear Files genügt ein einfaches Kopieren.



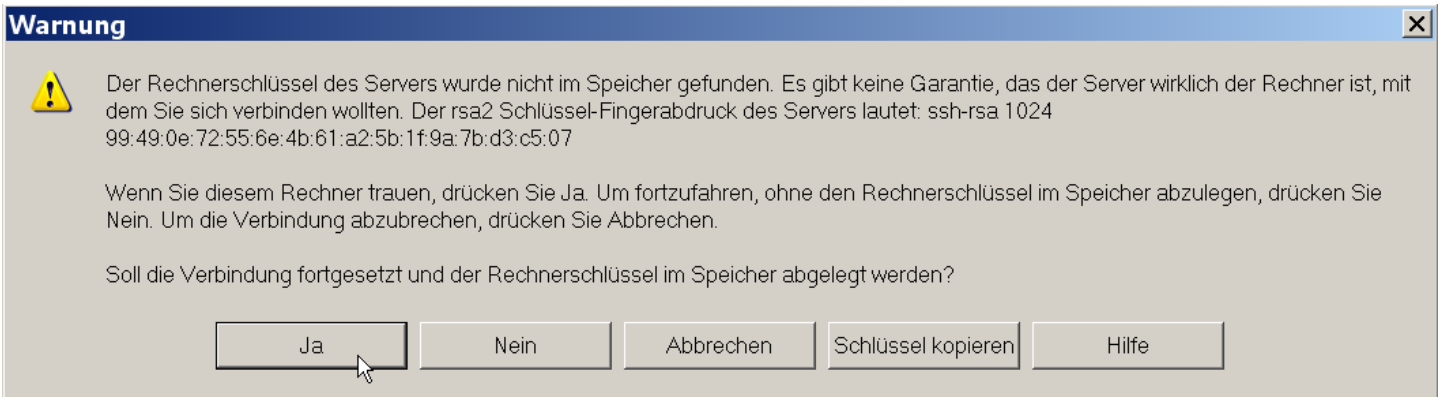
Falls sie WinSCP bereits benutzen erscheinen Ihre bisherigen Verbindungen. Sie müssen für Galadriel eine neue Verbindung erstellen. Klicken sie auf Neu oder New.



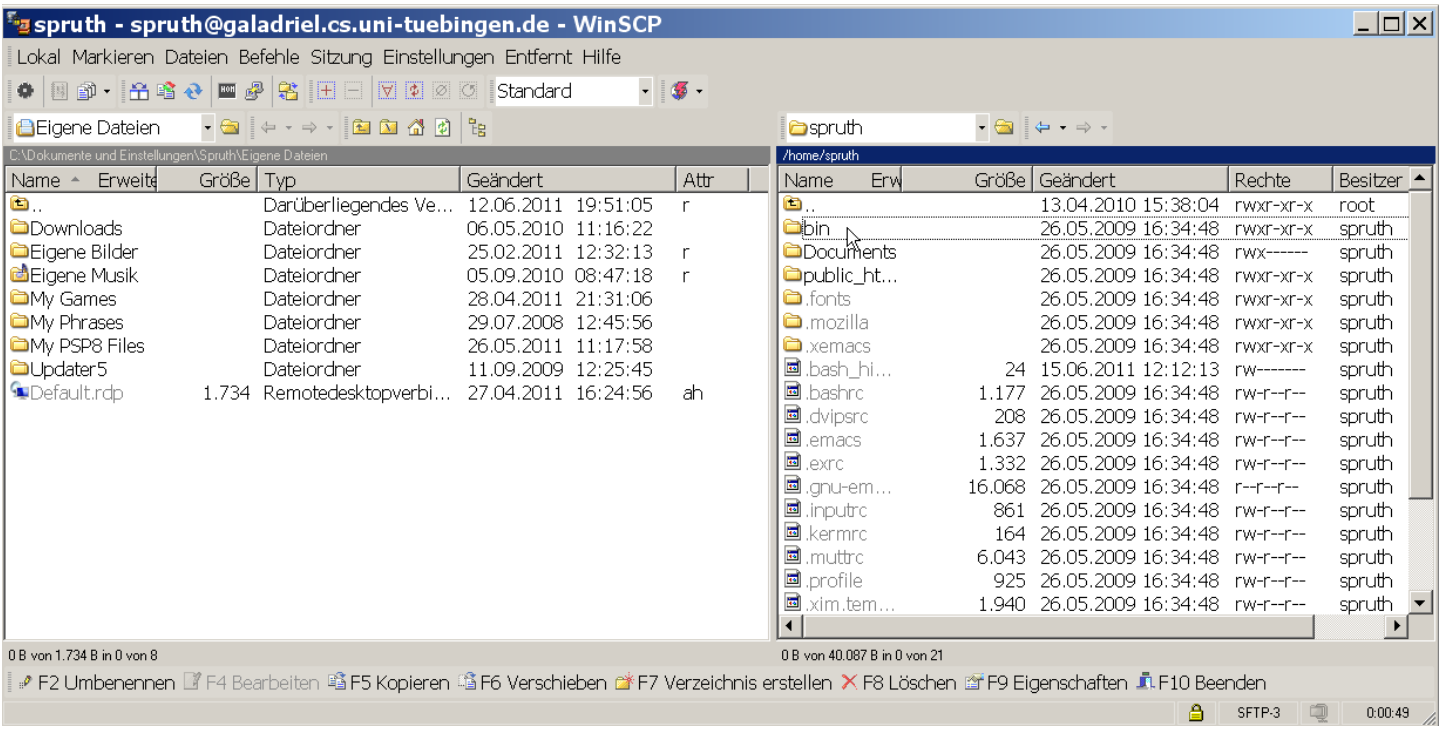
Der Rechnername ist galadriel.cs.uni-tuebingen.de, Port 22.

Wählen Sie SFTP (SSH File Transfer Protocol)

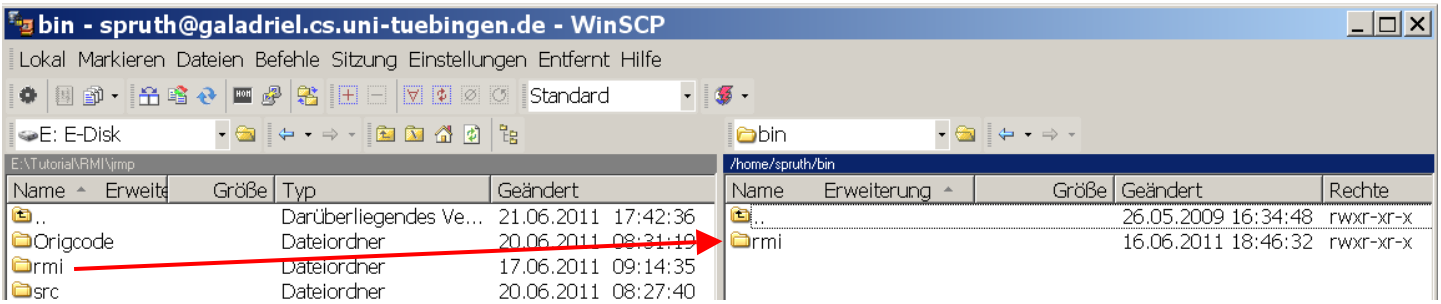
Die in der letzten Übung verwendete Verbindung war zu leia.informatik.uni-leipzig.de, einer z/OS LPAR. Wir haben für Sie eigene User IDs und Passwörter für Galadriel eingerichtet.



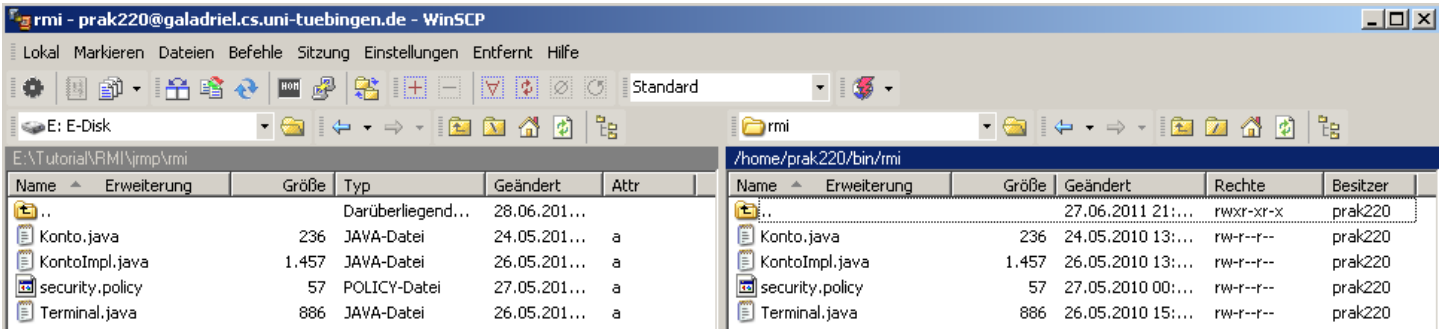
<Enter>



Im rechten Fenster sehen Sie Galadriel, im linken Fenster ihre Workstation. Navigieren Sie im rechten Fenster in das Verzeichnis bin, und im linken Fenster in das Verzeichnis, in dem Sie Ihr Verzeichnis rmi untergebracht haben.



Nun kopieren Sie ganz einfach mit *drag und drop* Ihr Verzeichnis rmi als Unterverzeichnis in das Verzeichnis bin.



Das Ergebnis sieht dann so aus.

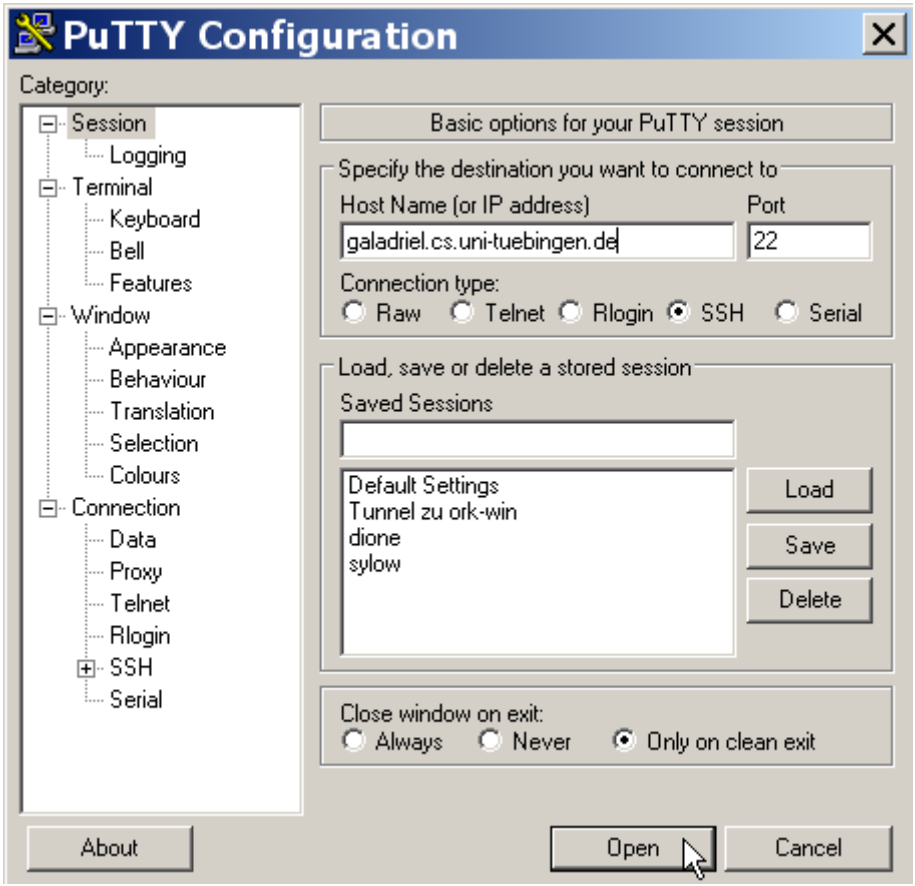
Ok, damit befindet sich der benötigte Quellcode auf Galadriel. Halten Sie das WinSCP Fenster geöffnet.

3.2 Zugriff auf zLinux

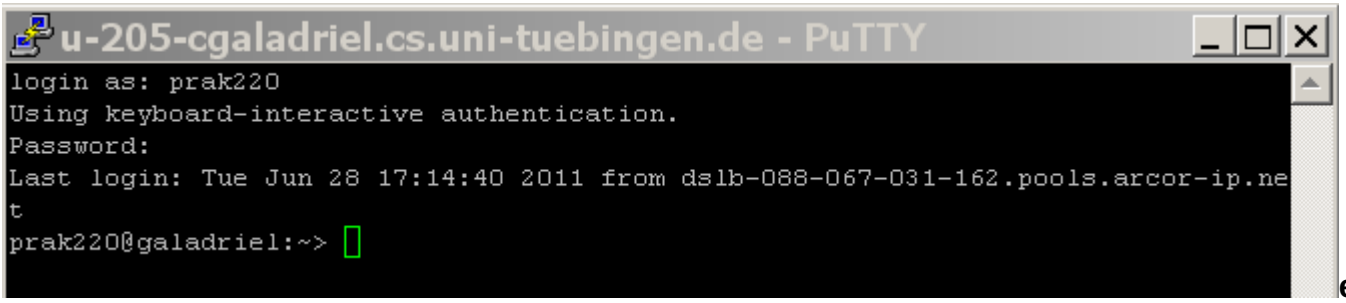
Als nächstes müssen wir den Quellcode kompilieren.

Wir brauchen analog zum 3270-Emulator einen zLinux-Klienten für den Zugriff auf Galadriel. Ähnlich wie beim 3270-Emulator gibt es hier zahlreiche Möglichkeiten. Ein populärer Klient für einen Linux Server ist PuTTY. PuTTY ist ein open source *telnet* und SSH Client für Windows.

Wenn noch nicht vorhanden, downloaden Sie PuTTY von http://www.putty.org/. Rufen Sie PuTTY auf.

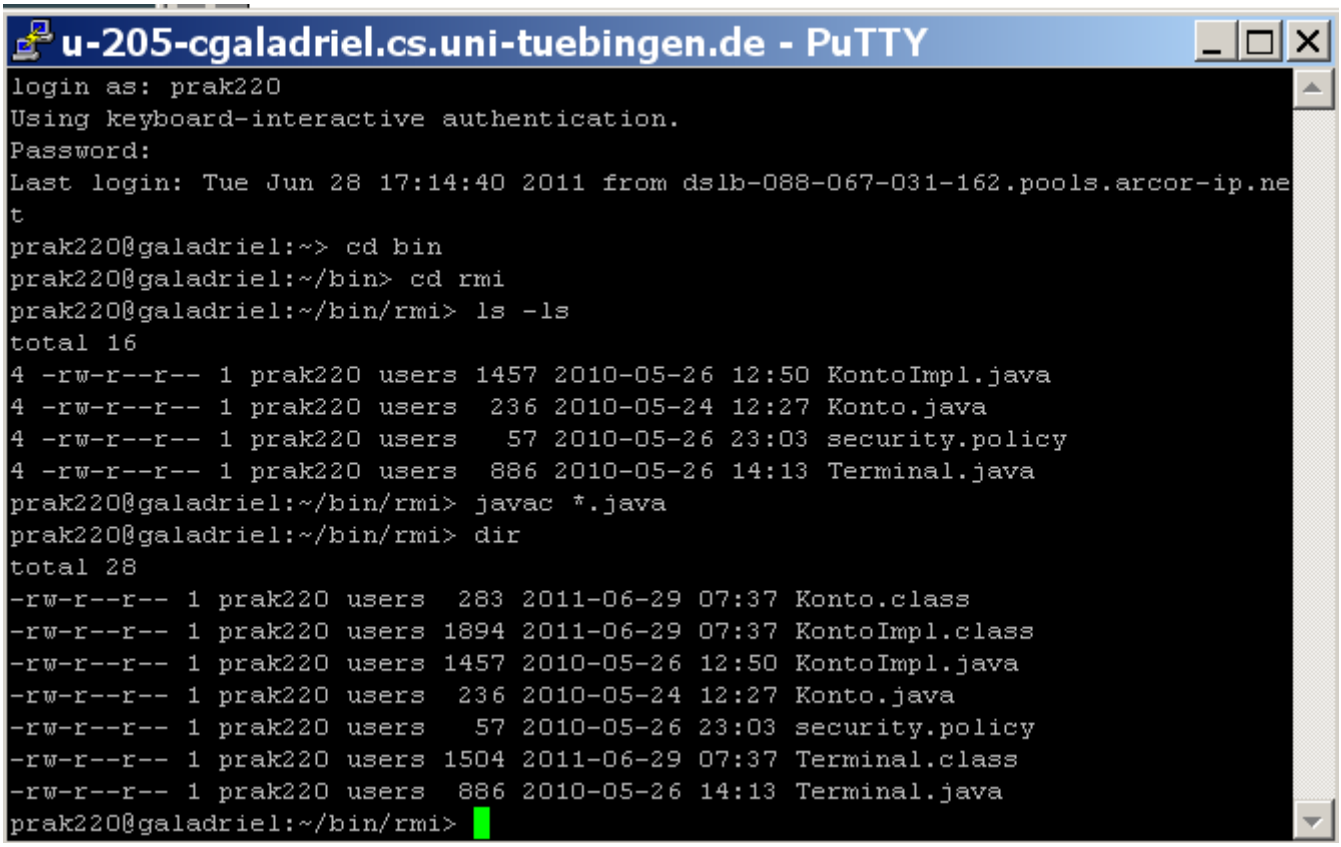


Starten Sie Putty mit der Adresse *galadriel.cs.uni-tuebingen.de, Port 22*.
PuTTY öffnet eine Command-Line-Shell.

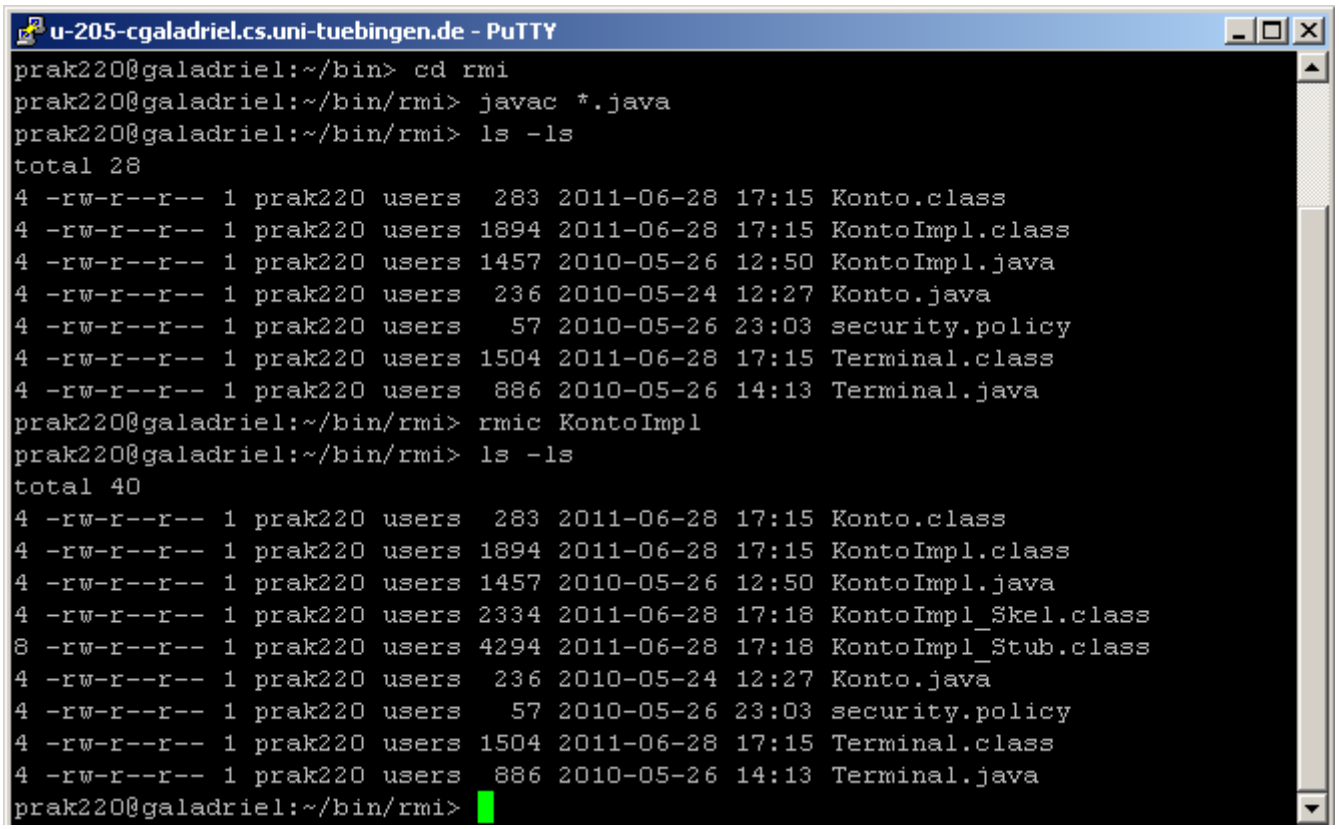


exit

Geben Sie ihre User ID und Ihr Password ein.

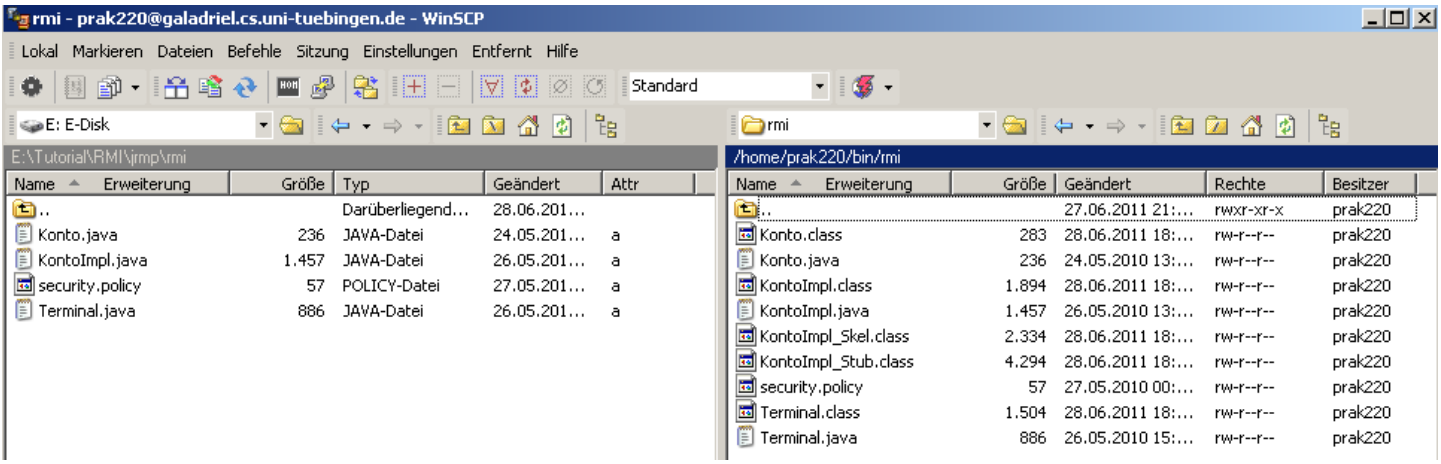


Wechseln sie in das Verzeichnis bin/rmi. Mit javac *.java kompilieren Sie alle java-Dateien.

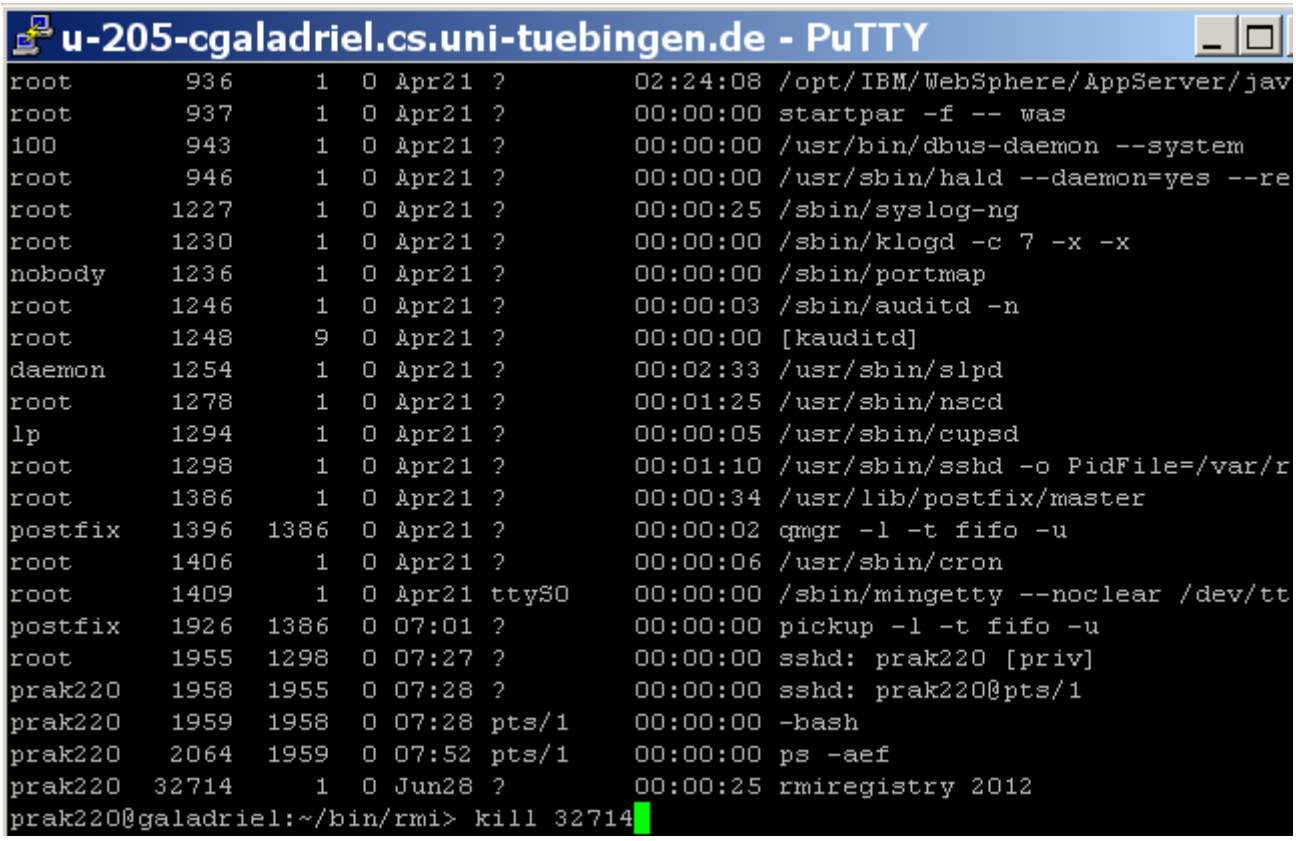


Erstellen sie mit dem Kommando `rmic KontoImpl` Skeleton- und Stub Klassen.

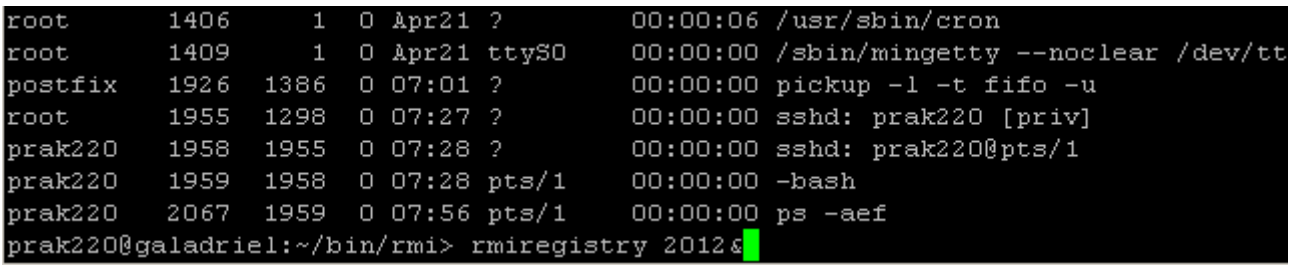
Hinweis: **Skeleton-Klassen werden nicht mehr von neueren RMI-Compiler-Versionen erstellt und benötigt. Sollte also keine KontoImpl_Skel.class nach dem Kompiliervorgang vorhanden sein ist dies kein Fehler.**



Dies ist das Ergebnis auf der Server Seite, was Sie in dem noch geöffneten WinSCP-Fenster sehen können.



Wenn bei der Ausführung des Kommandos eine Fehlermeldung erscheit, kann es sein, dass rmiregistry bereits läuft. Dies kann mit dem Kommando `ps –aef` überprüft werden. Wenn ja, notieren Sie die entsprechende PID und beenden Sie den rmiregistry-Prozess mit dem Command `kill [pid]`; in dem folgenden Beispiel also mit `kill 32714`.



Der Eintrag für den Registry Prozess ist jetzt verschwunden.

3.1.3 RMI Registry

Das“rmiregistry [port]“-Kommando startet eine Remote-Object-Registry auf dem angegebenen Port des verbundenen Servers. Der Aufruf `rmiregistry 2012&` bewirkt, dass unser Serverprogramm über port 2012 erreicht werden kann. Bitte denken sie daran, dass sie hier und bei den folgenden schritten Ihre eigene Port Nr. verwenden.

Das „&“ am Ende des Kommandos bewirkt, dass für die Ausführung ein getrennter Hintergrundprozess eingerichtet wird. Die Ziffer 23556 ist die PID (Prozess-ID) des Hintergrundprozesses.

```
root      1406     1  0 Apr21 ?        00:00:06 /usr/sbin/cron
root      1409     1  0 Apr21 ttyS0    00:00:00 /sbin/mingetty --noclear /dev/tt
postfix   1926  1386  0 07:01 ?        00:00:00 pickup -l -t fifo -u
root      1955  1298  0 07:27 ?        00:00:00 sshd: prak220 [priv]
prak220   1958  1955  0 07:28 ?        00:00:00 sshd: prak220@pts/1
prak220   1959  1958  0 07:28 pts/1    00:00:00 -bash
prak220   2067  1959  0 07:56 pts/1    00:00:00 ps -aef
prak220@galadriel:~/bin/rmi> rmiregistry 2012&
[1] 2068
prak220@galadriel:~/bin/rmi> java -Djava.security.policy=security.policy KontoIm
pl 2012
KontoObj bound to registry, port 2012.
KontoSever ready.
```

Als nächsten Schritt starten wir jetzt auf Galadriel unseren Java-Server.

Hierzu:
`java –Djava.security.policy=security.policy KontoImpl 2012`

Unter RMI läuft eine Client/Server-Verbindung grundsätzlich nur unter einer Sicherheitsvereinbarung, einer Security-Policy. welche sich normalerweise in einer Datei befindet. Mit dem Argument "-*Djava*.security.policy=security.policy" wird spezifiziert, dass die verwendete Security-Policy-Datei den (intellektuell anspruchsvollen) Namen „security.policy“ hat. (Wir hätten die Datei natürlich auch unter einem anderen Namen spezifizieren können.) Die Datei security.policy wird herangezogen, wenn eine Anwendung (hier KontoImpl) über eine Nachricht an Port 2012 aufgerufen wird.

Der Server bestätigt dies mit der Meldung „KontoServer Ready“. Damit ist der Server in der Lage, von einem beliebigen Klienten (hier Ihrer Workstation) über Port 2012 aufgerufen zu werden.

4. Ausführen des Programms

4.1 Erstellen der Client-Umgebung

In unserer Aufgabe wollen wir einen Java-Client mittels RMI mit einem Java-Server verbinden. Hierzu ist es erforderlich dass auf Ihrem Arbeitplatzrechner Java installiert ist. Im einfachsten Fall arbeitet Ihr Client ebenfalls mit einer JDK-Umgebung (an dieser Stelle existieren natürlich viele komfortable Alternativen).

Wir schlagen vor, dass Sie hierfür ein Verzeichnis c:\example einrichten, und in diesem Verzeichnis Ihr Java-Client-Programm ausführen (auch hier können sie nach Ihrer Wahl anders vorgehen). Stellen Sie sicher, dass die *class*- und *classpath*-Variablen richtig gesetzt sind, um eine Java-Programmausführung von diesem Verzeichnis zu ermöglichen. Spezifisch muss die classpath-Variable den Eintrag c:\example aufweisen.

Vielleicht ist es eine gute Idee zu verifizieren, dass Java-Programme korrekt ausgeführt werden.

Hierzu öffnen Sie auf Ihrer Workstation die Eingabeaufforderung. Dieses Fenster existiert zusätzlich zu dem bisher – immer noch offenen – PuTTy-Fenster. Produzieren sie mit einem Editor ein einfaches Test-Programm, z.B.

```
class hallo {
   public static void main(String[] args) {
      System.out.println("Hallo Welt really");
   }
}
```

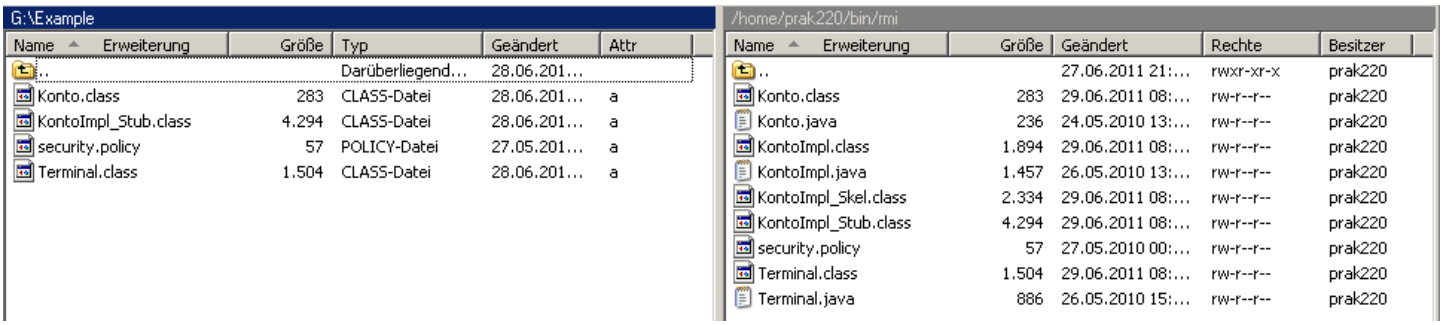
Übersetzen Sie es und führen Sie es aus.

```
C:\>md example

C:\>cd example

C:\example>javac hallo.java

C:\example>dir
 Volume in Laufwerk C: hat keine Bezeichnung.
 Volumeseriennummer: C057-A554

 Verzeichnis von C:\example

18.06.2008  08:14    <DIR>          .
18.06.2008  08:14    <DIR>          ..
18.06.2008  08:14               421 hallo.class
18.06.2008  08:12               117 hallo.java
               2 Datei(en)            538 Bytes
               2 Verzeichnis(se), 16.567.197.696 Bytes frei

C:\example>java hallo
Hallo Welt really

C:\example>_
```

Das Ergebnis sollte so aussehen. Löschen Sie wieder beide Dateien.

4.2 Download der Client Klassen

Beim Erstellen der class-Dateien auf Galadriel mittels *javac* und *rmic* hatten wir die Dateien für den Klienten gleich miterzeugt. Benutzen Sie WinSCP um die Dateien Terminal.class, Konto.class, KontoImpl_stub.class sowie security.policy in das Verzeichnis c:\example zu kopieren.



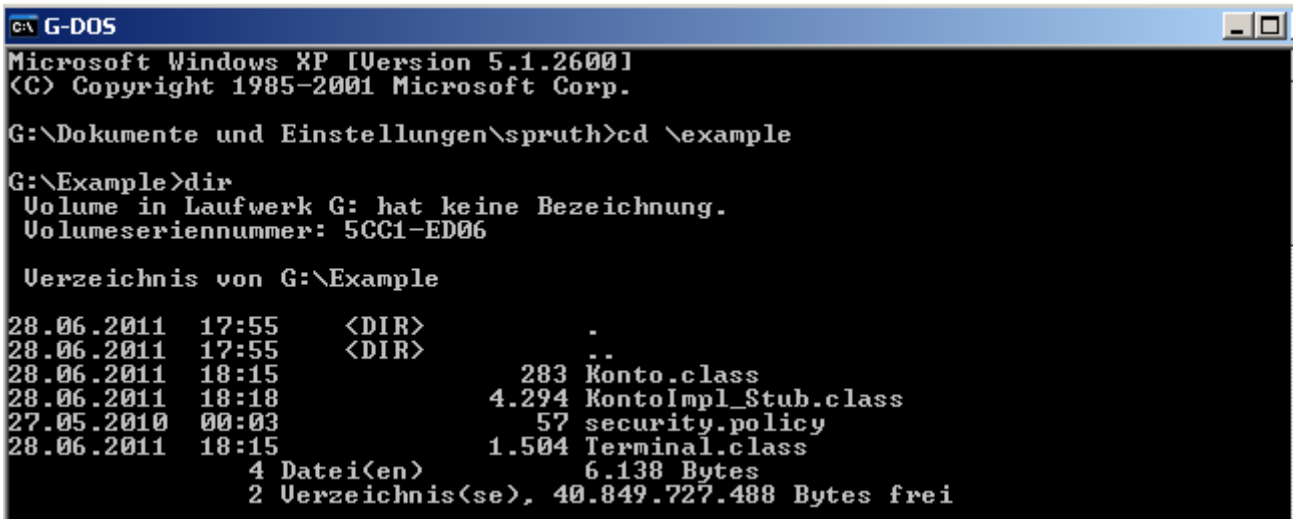
G:\Example

Name	Erweiterung	Größe	Typ	Geändert	Attr
..			Darüberliegend...	28.06.201...	
Konto.class		283	CLASS-Datei	28.06.201...	a
KontoImpl_Stub.class		4.294	CLASS-Datei	28.06.201...	a
security.policy		57	POLICY-Datei	27.05.201...	a
Terminal.class		1.504	CLASS-Datei	28.06.201...	a

/home/prak220/bin/rmi

Name	Erweiterung	Größe	Geändert	Rechte	Besitzer
..			27.06.2011 21:...	rwxr-xr-x	prak220
Konto.class		283	29.06.2011 08:...	rw-r--r--	prak220
Konto.java		236	24.05.2010 13:...	rw-r--r--	prak220
KontoImpl.class		1.894	29.06.2011 08:...	rw-r--r--	prak220
KontoImpl.java		1.457	26.05.2010 13:...	rw-r--r--	prak220
KontoImpl_Skel.class		2.334	29.06.2011 08:...	rw-r--r--	prak220
KontoImpl_Stub.class		4.294	29.06.2011 08:...	rw-r--r--	prak220
security.policy		57	27.05.2010 00:...	rw-r--r--	prak220
Terminal.class		1.504	29.06.2011 08:...	rw-r--r--	prak220
Terminal.java		886	26.05.2010 15:...	rw-r--r--	prak220

Das Ergebnis sieht dann so aus...

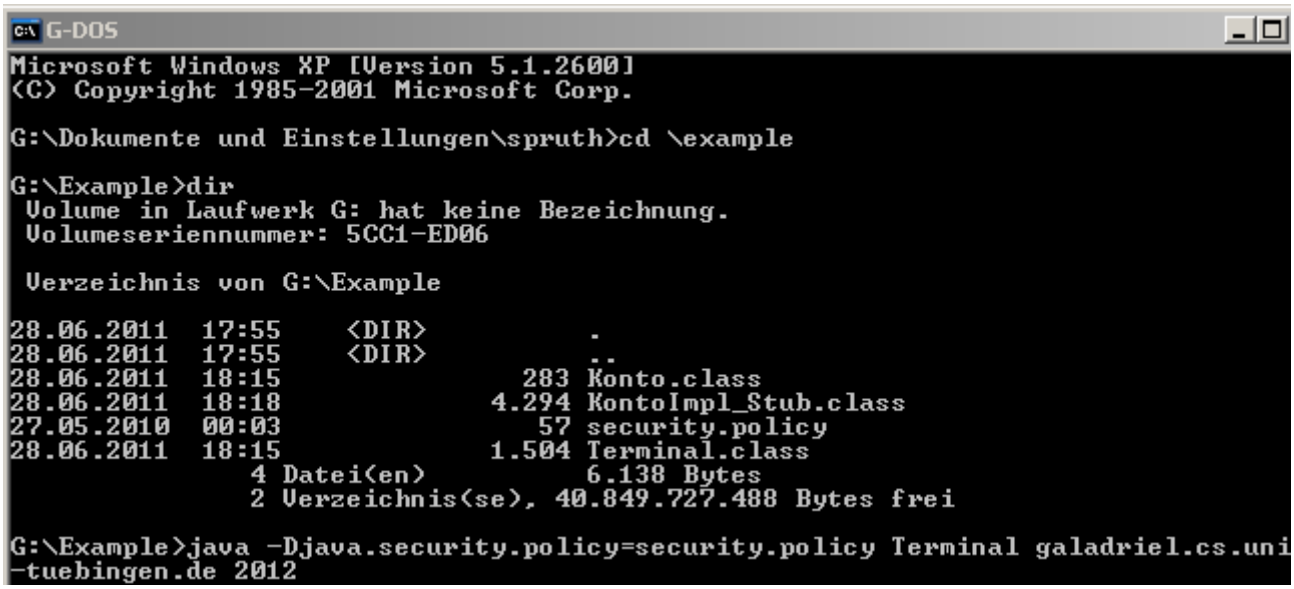


```
G-DOS
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

G:\Dokumente und Einstellungen\spruth>cd \example

G:\Example>dir
 Volume in Laufwerk G: hat keine Bezeichnung.
 Volumeseriennummer: 5CC1-ED06

 Verzeichnis von G:\Example

28.06.2011  17:55    <DIR>          .
28.06.2011  17:55    <DIR>          ..
28.06.2011  18:15               283 Konto.class
28.06.2011  18:18             4.294 KontoImpl_Stub.class
27.05.2010  00:03                57 security.policy
28.06.2011  18:15             1.504 Terminal.class
               4 Datei(en)          6.138 Bytes
               2 Verzeichnis(se), 40.849.727.488 Bytes frei
```

und im DOS-Fenster ebenso.

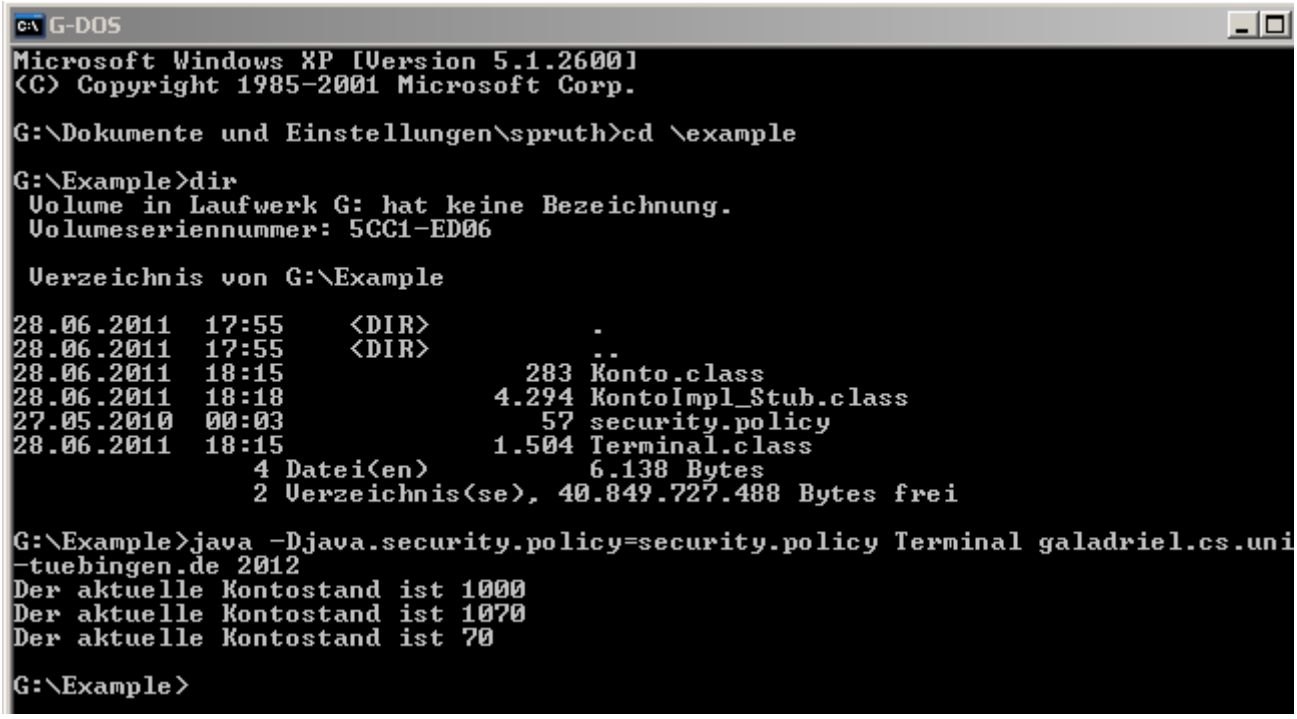
4.3 Aufruf des Servers

Sie haben nun ein ausführbares Java-RMI-Client-Programm auf ihrer Workstation, und ein dazu passendes RMI-Server-Programm auf Galadriel. Sie können nun mit dem Klienten den Server aufrufen.



Das geschieht mit dem folgenden Kommando:

java -Djava.security.policy=security.policy Terminal galadriel.cs.uni-tuebingen.de 2012



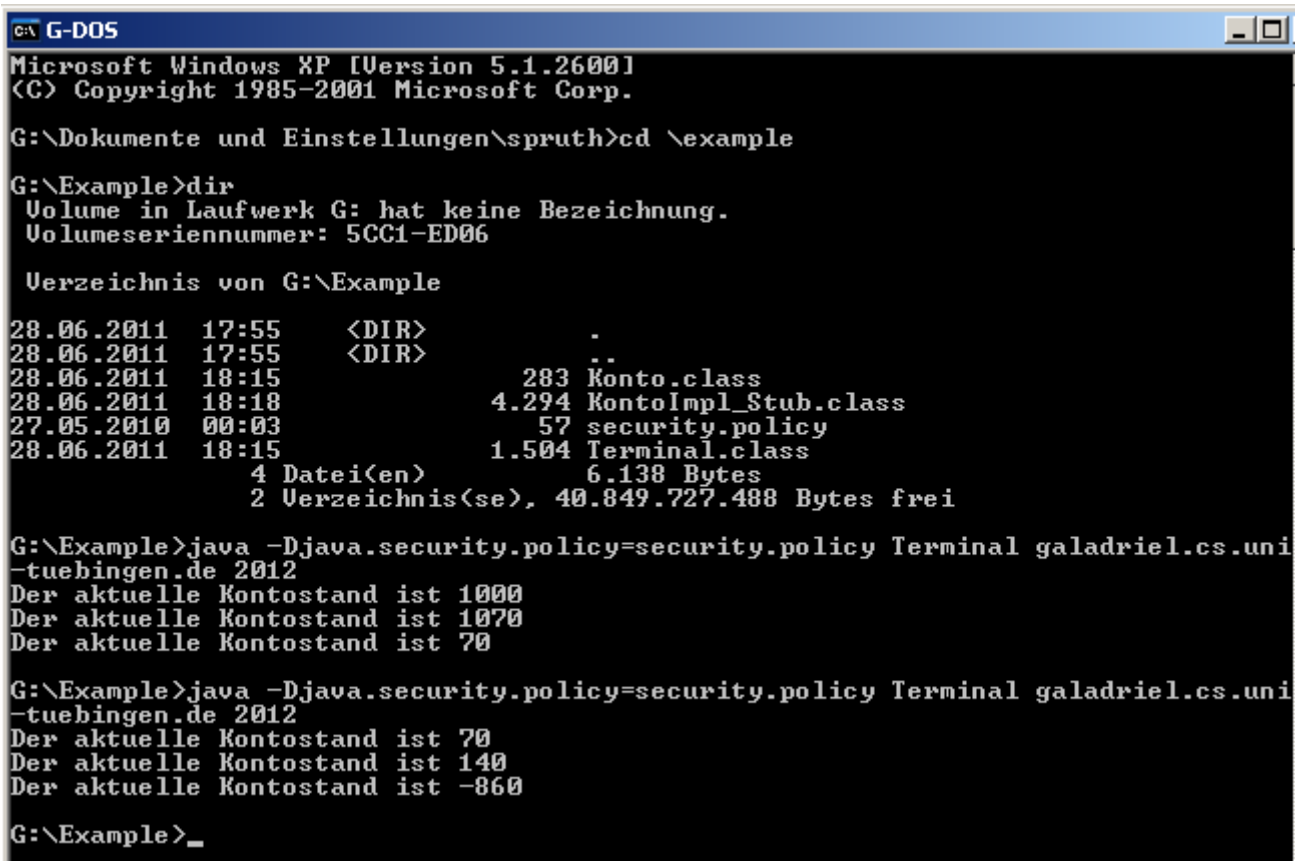
…und sie erhalten diese benutzerfreundliche Antwort.

Am besten schauen sie sich jetzt einmal den Quellcode von

- **KontoImpl.java**
- **Terminal.java**

an. Verifizieren sie, dass die gezeigte Ausgabe dem Quellcode entspricht.

Aufgabe: ***Erweitern und verbessern Sie die Quellcodes und reichen Sie wie gewohnt einen Nachweisscreenschot ein. Dieser soll ähnlich wie der Screenshot unten die Ausgaben Ihres verbesserten Servers zeigen. Unter diesen Ausgaben muss auch Ihr Vor- und Zuname mit dabei sein als Beleg dafür, dass Sie persönlich die Programmmodifikation vorgenommen haben. Sie könnten z.B. auch noch die Ausgabe des aktuellen Kontostandes durch die Angabe der jeweiligen Währung verbessern, z.B. durch folgende Ausgabe: „Der aktuelle Kontostand von Nils Michaelsen beträgt 1000 EUR“.***

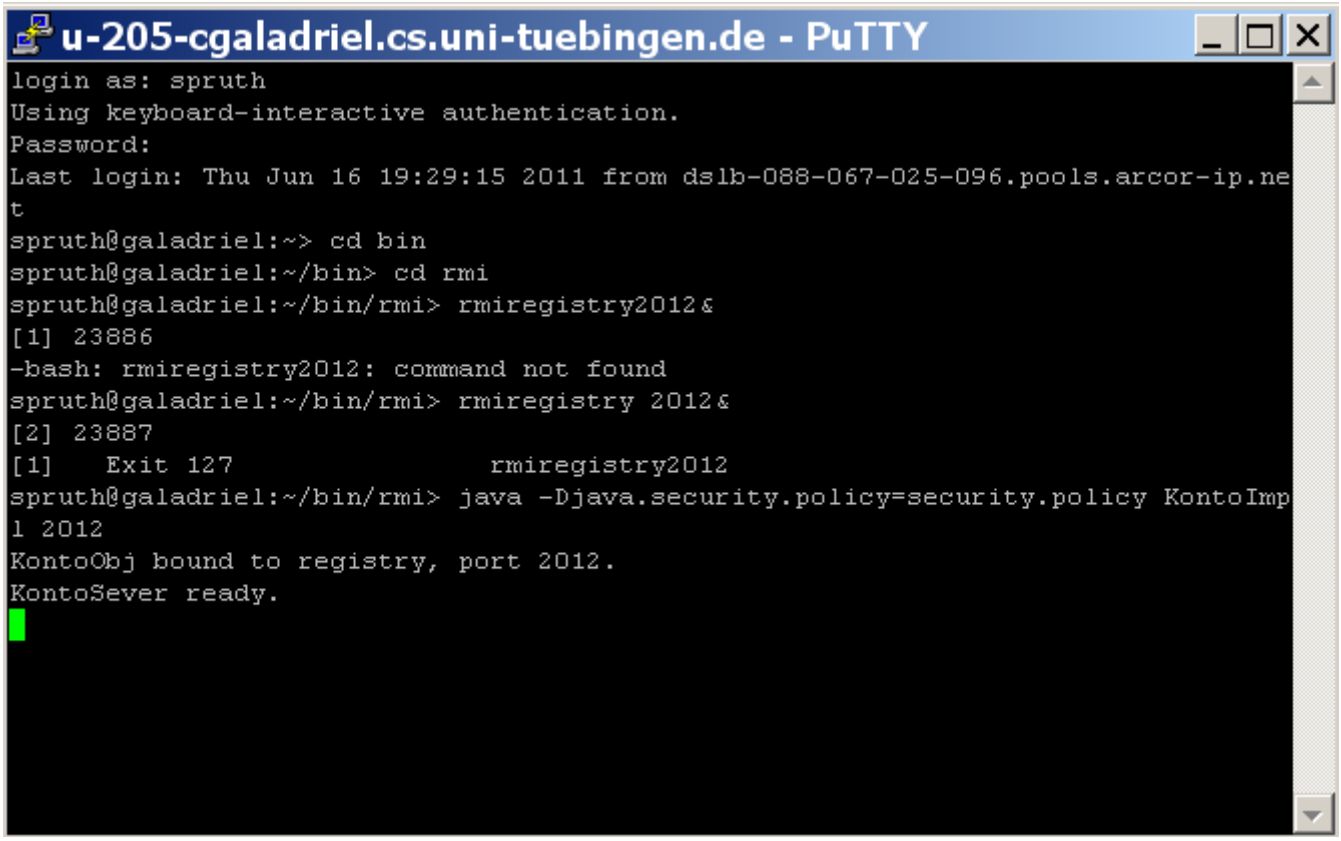


Der hier durchgeführte Methodenaufruf ist zustandslos, ähnlich wie beim Aufruf einer HTTP Seite im WWW. Da Ihr Programm die Variablen des Server-Objektes nicht verändert, erhalten Sie bei einem erneuten Aufruf das gleiche Ergebnis.

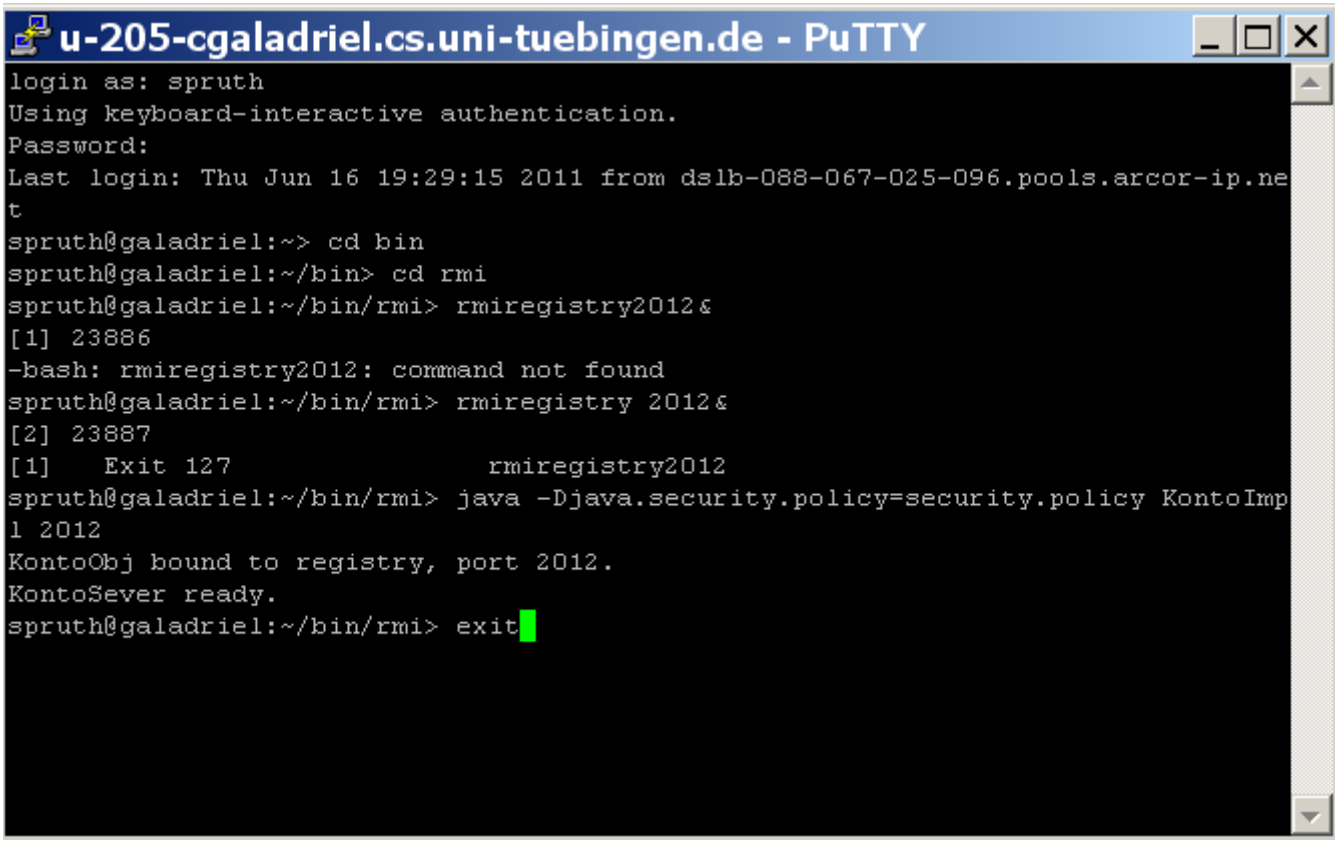
Da die JVM des Java-Servers multithreaded arbeitet, können im Prinzip die Methodenaufrufe mehrerer Teilnehmer von dem von Ihnen erstellten Server gleichzeitig verarbeitet werden. Bedenken Sie aber, dass die anderen Praktikumsteilnehmer alle ihre eigenen Server-Implementierungen mit anderen Ports benutzen.

4.4 Herunterfahren des Servers

Zum Abschluss Ihrer Sitzung sollten sie den auf Galadriel gestarteten Java-Server ordentlich herunterfahren. Gehen Sie wieder in das PuTTY-Fenster.



Auf die primitive Art können sie den Server mit Strg+C terminieren, und...



es ist eine gute Idee, an dieser Stelle den noch laufenden rmiregistry job mit `ps –aef` und `kill [pid]` zu beenden.

Danach Galadriel mit dem exit Kommando verlassen.

Das war es, - herzlichen Glückwunsch zu der erfolgreichen Durchführung des RMI-Tutorials.

Aufgabe: ***Als Abgabe wird eine funktionierende Version ihrer Anwendung inklusive Quellcode, eine Prozessliste der laufenden Prozesse wenn das Programm arbeitet und eine vollständige Prozessliste, wenn alle Programme beendet sind, sowie ein Screenshot der Ausgabe der Main-Methode der Terminal Klasse erwartet. Modifizieren Sie den Quellcode, damit die Ausgabe auf dem Bildschirm etwas aussagekräftiger und/oder benutzerfreundlicher aussieht.***

5. Fragen

5.1 Benötigt man mit der J2SE (Java 2 Standard Edition) ab der Version 1.5 ebenfalls den RMI-Compiler rmic? Warum?

- **Nein, der RMI-Compiler wird ab version 1.5 nicht mehr benötigt, da sogenannte dynamische Proxy-Objekte, die zur Laufzeit erstellt werden, die benötigen Informationen bereitstellen. Die Stub- und Skeleton-Klassen werden daher nicht mehr gebraucht.**
- **Werden jedoch ältere Java-Versionen auf Teilen des verteilten Systems verwendet, so muss man dennoch mit Hilfe von rmic die Stub- und Skeleton-Klassen erstellen.**

5.2 Was beinhaltet das Interface java.rmi.Remote? Wieso?

- **Das Interface beinhaltet keine Methoden, da es nur zu Identifikation eines Remote-Objekts dient. Ein Interface, das von java.RMI.Remote erbt, definiert die Remote-verfügbaren Methoden. Jedes Objekt, welches nun dieses Interface Implementiert, stellt daher die implementierten Methoden für Remote-Aufrufe zur Verfügung.**

6. Informationsquellen

Java RMI Home Page:
http://java.sun.com/products/jdk/rmi/index.html

Tutorial:
http://java.sun.com/j2se/1.4.2/docs/guide/rmi/getstart.doc.html

RMI Spezifikation mit ausführlicher Dokumentation:
http://java.sun.com/j2se/1.4.2/docs/guide/rmi/spec/rmiTOC.html

RMI Tutorial
http://download.oracle.com/javase/tutorial/rmi/index.html

7. Anhang
Beispielcode

6.1 Konto.java

```
import java.rmi.*;

public interface Konto extends Remote {
   public String getKontostand() throws RemoteException;
   public void einzahlung(int betrag) throws RemoteException;
   public void auszahlung(int betrag) throws RemoteException;
}
```

6.2 security.policy

```
grant {
     permission java.security.AllPermission;
};
```

6.3 KontoImpl.java

```
import java.rmi.*;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.server.*;

public class KontoImpl extends UnicastRemoteObject implements Konto {

int kontostand;

protected KontoImpl() throws RemoteException {
          this.kontostand=1000;
     }

public String getKontostand() throws RemoteException {
     return "Der aktuelle Kontostand ist "+kontostand;
}

public void auszahlung(int betrag) throws RemoteException {
     kontostand=kontostand-betrag;
}

public void einzahlung(int betrag) throws RemoteException {
     kontostand=kontostand+betrag;
}


public static void main (String[] args ) throws RemoteException {
     int port = (args.length > 0) ? Integer.parseInt(args[0]) : 2012;

     KontoImpl obj = new KontoImpl();
     String objName = "KontoObj";

     if (System.getSecurityManager() == null) {
          System.setSecurityManager (new RMISecurityManager());
     }

     Registry registry = LocateRegistry.getRegistry (port);
     boolean bound = false;

     for (int i = 0; ! bound && i < 2; i++) {
          try {
               registry.rebind (objName, obj);
               bound = true;
               System.out.println (objName+" bound to registry, port " +
                    port + ".");
          }
```

```
          catch (RemoteException e) {
               System.out.println ("Rebinding " + objName + " failed, " +
                    "retrying ...");
               registry = LocateRegistry.createRegistry (port);
               System.out.println ("Registry started on port " + port +
                    ".");
          }
     }

     System.out.println ("KontoSever ready.");
     }
}
```

6.4 Terminal.java

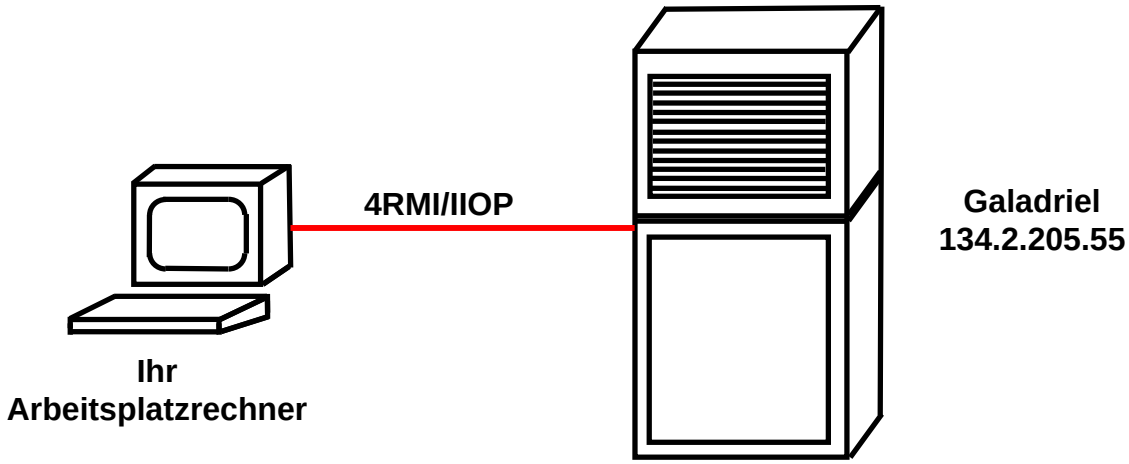
```
import java.net.MalformedURLException;
import java.rmi.*;
import java.rmi.server.*;
public class Terminal {

static public void main (String[] args) throws MalformedURLException,
RemoteException, NotBoundException{
     String host = (args.length < 1) ? "galadriel.cs.uni-tuebingen.de"
          ://"127.0.0.1" :
     args[0];
     int port = (args.length < 2) ? 2012 : Integer.parseInt(args[1]);
     try {
          if (System.getSecurityManager() == null) {
               System.setSecurityManager (new RMISecurityManager());
          }
          Konto obj = (Konto) Naming.lookup ("rmi://" + host + ":" + port
               + "/" + "KontoObj");
          System.out.println (obj.getKontostand());
          obj.einzahlung(70);
          System.out.println (obj.getKontostand());
          obj.auszahlung(1000);
          System.out.println (obj.getKontostand());
          }
          catch(Exception e) {
               System.out.println ("Client failed, caught exception " +
                    e.getMessage());
          }
     }
}
```

JAVA Remote Method Invocation
RMI/IIOP Tutorial

Dieses Tutorial stellt die Aufgabe, ein Java-Programm auf einem Client (Ihrem PC) zu installieren, und mittels RMI einen Zugriff auf einen Java-Server durchzuführen, der in einer zLinux-LPAR auf unserem z9 Mainframe in Tübingen läuft. Die zLinux-LPAR hat den Namen

galadriel.cs.uni-tuebingen.de (oder 134.2.205.55) .

Danksagung an Herrn Robert Harbach für die Bereitstellung des Materials.



Im Gegensatz zu dem JRMP-Tutorial verwenden wir hier RMI/IIOP (gesprochen „RMI over IIOP“). Wir gehen davon aus, dass Sie vor Bearbeitung dieses Tutorials vorher das JRMP-Tutorial bearbeitet haben.

Übersicht

1. Das RMI/IIOP Programmiermodell

1.1 Corba

Der Corba Remote Procedure Call wurde als Nachfolger für die in den 80er Jahren dominierenden Sun RPCs und DCE RPCs geplant und entwickelt. Corba Version 1.0 ist seit 1991 verfügbar, Corba 2.0 seit 1992 und Corba 3.0 seit 2002.

Die wichtigsten Eigenschaften von Corba sind:

- **Die Services eines Corba-Servers werden als Objekte dargestellt und über Objekt-Klassen und deren Methodenaufrufe implementiert.**
- **Corba-Klassen (Binaries) können in vielen Sprachen geschrieben werden (z.B. Cobol, C++, PLI, ADA und Java) und einwandfrei miteinander kommunizieren.**
- **Um dies zu ermöglichen, werden die Schnittstellen der Server-Objekte in einer einheitlichen Schnittstellensprache, der Corba-IDL (Interface Definition Language) beschrieben, unabhängig davon, in welcher Programmiersprache das Server-Objekt geschrieben wurde. Die IDL wird benutzt, um Server-Skeletons und evtl. Client-Stubs zu erstellen.**
- **Auf dem Client und Server wird je eine einheitliche Laufzeitumgebung benötigt, der Corba-ORB (Object Request Broker).**
- **Alle Server-Objekte werden über eine eine einheitliche Bezeichnung adressiert, der IOR (Interoperable Object Reference).**
- **Für die Kommunikation zwischen RMI-Objekten und/oder Corba-Objekten wird das IIOP *(*Internet Inter-Orb Protocol) verwendet.**

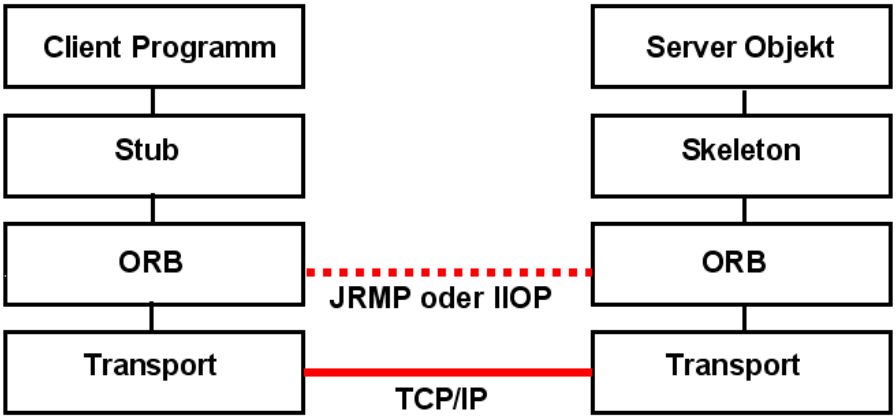
Zusätzlich existieren weitere Corba-Services, z.B. Transaction Service, Naming Service, usw.

IIOP, JRMP, Telnet, 3270, FTP, http und SOAP sind Protokolle der OSI-Layer-Schicht 5 und verwenden heute fast immer TCP/IP als Schicht-4-Übertragungsprotokoll.

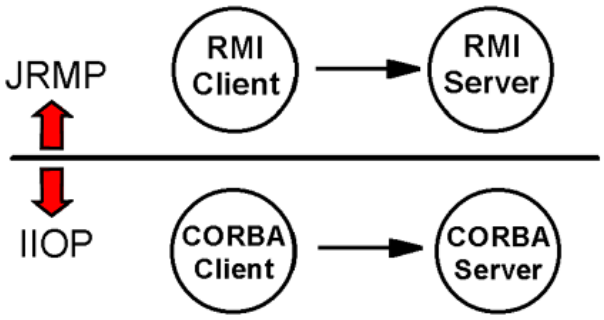
1.2 Java Remote Procedure Call

Bei der Entwicklung der Java-RMI entstand ein mit Corba sehr vergleichbares, aber inkompatibles Programmiermodell. Spezifisch wurde das proprietäre JRMP-Protokoll und eine dazu passende Laufzeitumgebung geschaffen. An Stelle der Corba-IDL benutzt man die Java-Schnittstelle, die Bestandteil von Java ist. Der Vorteil dieser Vorgehensweise ist eine etwas einfachere Programmierung.

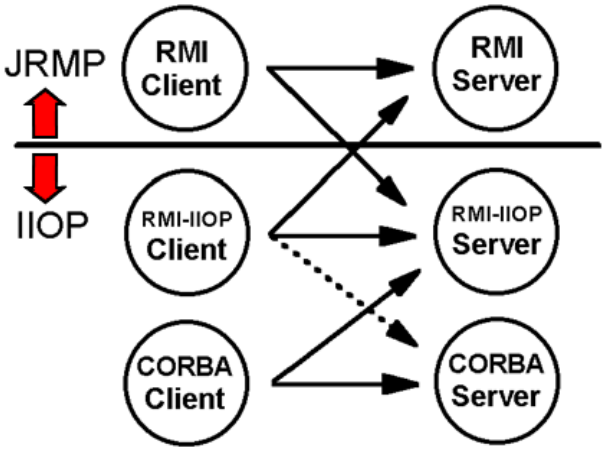
Da es Java RMI over JRMP an Interoperabilität mit anderen Sprachen mangelt, wurde zusätzlich RMI/IIOP geschaffen. Hierzu wurde die JRMP-Laufzeitumgebung durch einen Corba-ORB, und das JRMP-Protokoll durch das IIOP-Protokoll ersetzt. Damit ist es möglich, RMI-Objekte in eine Corba-Umgebung einzubinden. RMI-Objekte benutzen das IIOP-Protokoll, um mit Corba-Objekten zu kommunizieren.



RMI/IIOP bietet den Vorteil, Remote-Interfaces einfach in Java zu schreiben und diese mit den APIs von Java-RMI zu implementieren. Es ist nicht nötig, eine separate IDL-Sprache zu lernen.



JRMP hat einen Nachteil. Ohne RMI/IIOP sind RMI und Corba zwei getrennte Welten. JRMP-Clients können nur mit JRMP-Servern, und Corba-Clients nur mit Corba-Servern kommunizieren.



Die Einführung von RMI/IIOP verbessert diese Situation. Ohne Änderung von existierendem Code ist es möglich, dass JRMP-Cients mit RMI/IIOP-Server-Objekten kommunizieren, und umgekehrt. In anderen Worten, JRMP und RMI/IIOP passen nahtlos zusammen. Weiterhin können Corba-Client-Anwendungen problemlos auf RMI/IIOP-Server-Objekte zugreifen. Beim Zugriff von RMI/IIOP-Clients auf Corba-Server-Objekte existieren einige Einschränkungen, wenn es sich um älteren Code handelt.

Der Java-RMI-Standard offeriert heute zwei alternative Protokolle: JRMP und RMI/IIOP. Manche Softwareunternehmen implementieren beides in ihren RMI-Produkten. IBM hat entschieden, in ihren Java-Produkten ausschließlich RMI/IIOP einzusetzen.

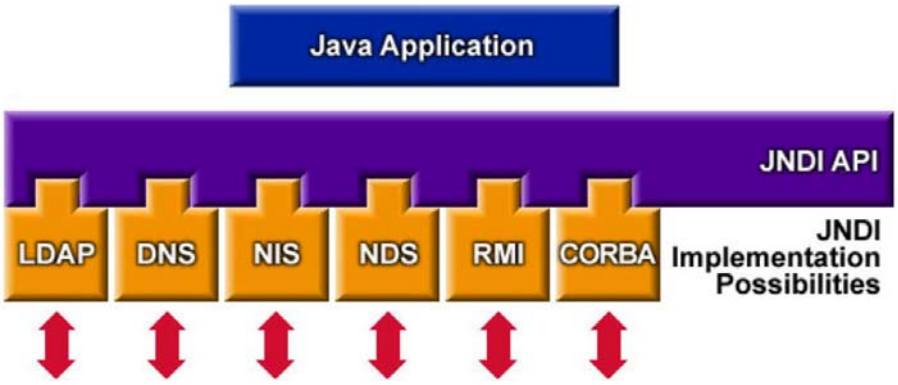
Man sollte meinen, dass der erhöhte Funktionsumfang von RMI/IIOP gegenüber JRMP eine schlechtere Performance zur Folge hat. Dies ist interessanterweise nicht unbedingt der Fall. Die Frage des Unterschieds in der Performance wird im Internet und in den einschlägigen Foren heiß und kontrovers diskutiert und hängt wohl auch von der konkreten Anwendungssituation ab. Es scheint aber so zu sein, dass in der Mehrzahl der Fälle RMI/IIOP in Bezug auf Performance besser abschneidet als JRMP.

Ein ORB (Object Request Broker) ist eine Komponente eines Betriebssystems. Er stellt ein einheitliches Klassenmodell sowie Standards für die Organisation von Klassen-Bibliotheken und die Kommunikation zwischen Objekten zur Verfügung. Alle z/OS-Java-Produkte, z.B. CICS-Java oder der WebSphere-EJB-Container enthalten deshalb einen Corba-ORB.

1.3 Corba und RMI

Es existieren einige Unterschiede in der Programmierung von JRMP- und RMI/IIOP-Anwendungen.

1.3.1 Naming



JNDI (Java Naming and Directory Interface) ist eine API, über die ein Java-Programm einen Namens- und Directory-Dienst in Anspruch nehmen kann. Der Namens- und Directory-Dienst ermöglicht einen Lookup von RMI-Objekten zwischen entfernten JVMs. Mögliche Implementierungen der JNDI-Schnittstelle sind z.B. der LDAP-Server, Corba-Common-Object-Services-Name- (COSNaming-) Server oder der rmiregistry-Server. rmiregistry ist ein Bestandteil des JDK. JRMP verwendet standardmäßig einen rmiregistry-Server.

RMI/IIOP benutzt statt dessen einen COSNaming-Server, welcher Remote-References als Interoperable-Object-References (IORs) enthält. Es existieren verschiedene COSNaming-Server. Wir verwenden den weit verbreiteten `tnameserv`, ebenfalls Bestandteil der JDK.

Das Kommando `tnameserv -ORBInitialPort` startet auf Default-Port 900 (wenn nichts anderes spezifiziert ist).

RMI erwartet aber einen RMI-Name-Server. Deshalb benötigen wir eine JNDI-nach-COSNaming-Bridge. Die `com.sun.jndi.cosnaming.CNCtxFactory`-Bridge ist ein Bestandteil der Sun-JDK, die auf Galadriel installiert ist. Andere Java-Produkte verwenden eine andere Bridge, WebSphere z.B. verwendet einen eigenen Name-Server und eine eigene Bridge.

1.3.2 `rmic`

Der rmic-Compiler erhält die zusätzliche Option `–iiop`. Damit werden Corba kompatible Stubs und Skeletons generiert. Weiterhin wird eine Corba-IDL von dem Java-Interface erzeugt.

1.3.3 Unterschiede in der Codierung

Unterschiedliche Import-Statements werden benutzt (vergleichen Sie die Code-Beispiele der JRMP- und RMI/IIOP-Tutorials).

Das Server-Objekt verwendet anstatt der `UnicastRemoteObject`- die `PortableRemoteObject`-Klasse.

1.3.4 Unterschiede in der Programm-Ausführung

Siehe Teil 4 des Tutorials. Vergleichen Sie es mit dem Teil 4 des JRMP-Tutorials.

1.3.5 Literatur

RMI-IIOP home page
http://java.sun.com/products/rmi-iiop/index.html (obsolet)

RMI-IIOP Programmer's Guide
http://docs.oracle.com/javase/7/docs/technotes/guides/rmi-iiop/rmi_iiop_pg.html

OMG Java language to IDL Mapping specification
ftp://ftp.omg.org/pub/docs/ptc/99-03-09.pdf (obsolet)
http://www.omg.org/spec/JAV2I/

2. RMI/IIOP Programmierung

2.1 Übersicht

Modifizieren Sie ihre bestehende RMI-JRMP-Anwendung so, dass sie das IIOP-Protokoll benutzt und damit Interoperabilität mit Corba-ORBs bietet. Sichern Sie am besten Ihre alte RMI/JRMP-Anwendung, bevor Sie sie modifizieren.

RMI/IIOP macht gegenüber dem JRMP-Tutorial die folgende Änderungen notwendig:

- **Ihre KontoImpl-Klasse erweitert anstatt der UnicastRemoteObject- die PortableRemoteObject-Klasse aus dem javax.rmi-Package**
- **Anstatt des rmiregistry-Name-Servers wird diesmal der Corba kompatible COSNaming-Server benutzt:**

```
> tnameserv -ORBInitialPort <port>
```

- **Ihr Server muss in der main-Methode einen Kontext für JNDI erstellen:**

```
import javax.naming.InitialContext;
InitialContext INC = new InitialContext();
```

Das Binden des Remote-Objekts funktioniert mit der rebind-Methode des InitialContext-Objekts und ersetzt die „Registry“ aus der letzten Aufgabe:

```
INC.rebind("Konto", KontoImpl_object);
```

- **Erzeugen Sie auf die gleiche Art einen JNDI-Kontext in der main-Methode des Clients. InitialContext bietet eine entsprechende „lookup“-Methode um eine Referenz von entfernten Objekten zu bekommen. Sie verwenden dann nicht mehr die lookup-Methode der Naming-Klasse.**

- **Nach dem Kompilieren Ihrer Klassen rufen Sie den RMI-Compiler mit der Option -iiop auf:**

```
> rmic -iiop KontoImpl
```

- **Der Server wird wie folgt gestartet:**

```
> java -
Djava.naming.factory.initial=com.sun.jndi.cosnaming.CNCtxFa
ctory -Djava.security.policy=<POLICYFILE> -
Djava.naming.provider.url=iiop://<SERVER:PORTNR> KontoImpl
&
```

(Beachten Sie: der Server nimmt keinen Port-Parameter mehr entgegen, ändern Sie also entpsrechend ihre main-Methode)

- **Der Client wird analog zum Server gestartet**

2.2 Ihre Aufgabe: Eine verteilte Anwendung mit RMI/IIOP

2.2.1 Szenario

Wir verwenden das gleiche Szenario wie in dem JRMP-Tutorial.

2.2.2 Code Beispiel

Quellcode-Beispiele für drei Dateien

- **KontoImpl.java,**
- **Terminal.java,**
- **Konto.java,**

sowie eine Security-Policy-Datei sind im Anhang wiedergegeben.

2.2.3 Port Nummer

Server-Anwendungen erhalten Nachrichten über Ports. Das obige Beispiel verwendet Port-Nr. 2012.

Mehrere Studenten werden gleichzeitig dieses Tutorial bearbeiten. Wir benutzen auf Galadriel aber keine Middleware wie z.B. WebSphere. Stattdessen startet jeder Benutzer auf Galadriel seinen eigenen Server. Damit braucht jeder Benutzer für seinen Server eine eigene Port-Nummer.

Wir schlagen vor, dass Sie eine Port-Nr. im Bereich zwischen 50001 und 50999 wählen.

Um Konflikte mit anderen Benutzern zu vermeiden, schlagen wir vor, dass Sie als Port-Nr. 50xxx wählen, wobei xxx die drei letzten Ziffern Ihrer prak-User-ID sind. Wenn Sie also die User-ID prak519 benutzen, wäre Ihre Port Nr. 50519.

Beachten Sie, dass die obigen Beispielprogramme Port Nr. 2012 benutzen. Sie müssen diese, und auch überall sonst in diesem Text, durch Ihre eigene Port Nr. ersetzen.

- **Bitte fahren Sie Vor Abschluss des Tutorials den von Ihnen gestarteten tnameserv-Prozess wieder herunter!**

2.2.4 Vorgehen

2.2.4.1 Benötigter Quellcode

Erstellen Sie ein eigenes leeres Verzeichnis auf Ihrem Rechner, z.B. iiop, und erstellen Sie die 3 java-Dateien

- **Konto.java**
- **KontoImpl.java**
- **Terminal.java**

in diesem Verzeichnis, sowie zusätzlich die Security-Policy-Datei .

Hinweis:

Eine rudimentäre Implementierung, welche obigen Anforderungen genügt, reicht aus. Auf PIN, Login, mehrere Kontos pro Server, Setzen/Prüfen der Kreditlinie, Zinsen usw. kann verzichtet werden.

Wenn Sie wollen, können Sie dies natürlich trotzdem implementieren.

2.2.4.2 Das Interface und die Klassen kompilieren

Sie müssen die Klassen

- **Konto.class**
- **KontoImpl.class**
- **Terminal.class**

mit Hilfe des `javac`-Compilers erzeugen. Hierfür brauchen Sie eine Java-Entwicklungsumgebung. Sie können (für die primitive Aufgabenstellung ausreichend) hierfür die JDK benutzen, oder eine komfortablere Entwicklungsumgebung wie Eclipse einsetzen.

2.2.4.3. Java Kompatibilität

Genauso wie in dem JRMP-Tutorial lösen wir das Problem der inkompatiblen Java-Versionen, indem wir das Kompilieren aller Klassen auf dem Server durchführen, und die Client-Klassen anschließend in die Workstation laden.

2.2.4.4. Stubs und Skeletons mit rmic erstellen.

Mit Hilfe der Schnittstellenbeschreibung ist es möglich, Stubs und Skeletons zu erstellen. Dies geschieht mit Hilfe des `rmic`-Compilers. Das Java-Interface (in diesem Beispiel „Konto“) Ihres Service-Implementation-Programms enthält alle Informationen, die der Compiler benötigt.

Den rmic-Compiler müssen Sie mit mit der `–iiop` Option

```
rmic –iiop KontoImpl
```

aufrufen. Als Ergebnis werden iiop kompatible Stubs und Skeletons erstellt. Es sollten die Dateien

_KontoImpl_Tie.class
_Kontol_Stub.class

zusätzlich erstellt werden. Die so entstandenen 5 class-Dateien

Konto.class
KontoImpl.class
Terminal.class
_KontoImpl_Tie.class
_Kontol_Stub.class

sind das Ergebnis.

Als nächste Schritte müssen diese Dateien jetzt erzeugt und dann ausgeführt werden.

3. Erstellen der Java-Klassen auf dem zLinux-Server

3.1 Laden der Java-Klassen auf den zLinux-Server

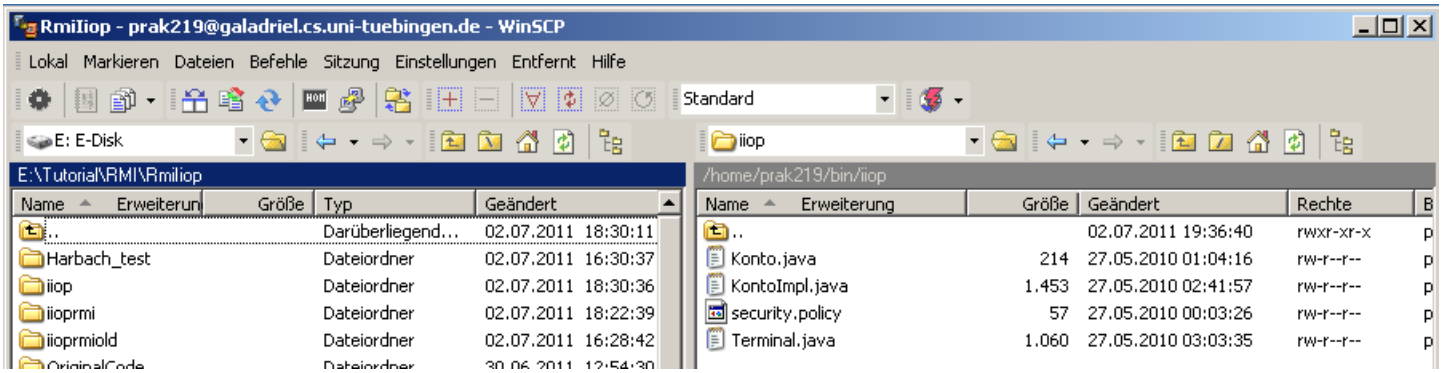
Die 4 Dateien

- **Konto.java**
- **KontoImpl.java**
- **Terminal.java**
- **security.policy**

müssen nun auf dem Server galadriel.cs.uni-tuebingen.de (oder 134.2.205.55) geladen werden. Das Laden geschieht am Einfachsten mit WinSCP.

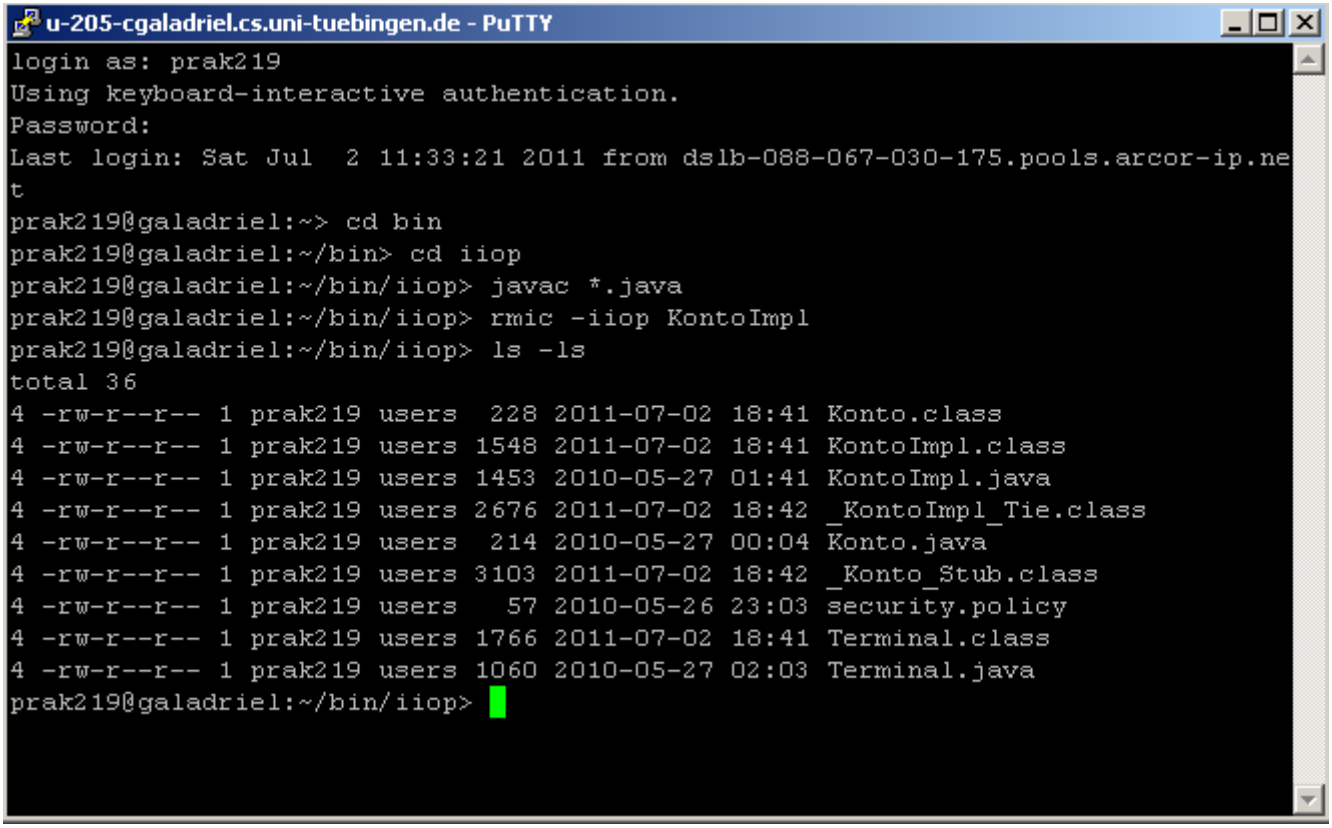
Wir arbeiten auch in diesem Tutorial mit einfachen Java-Klassen, keinen Servlets und EJBs. An Stelle der komplexeren Web-Application-Server-Installation mit jar- und ear-Files genügt ein einfaches Kopieren.

Kopieren Sie wie in dem JRMP-Tutorial das von Ihnen erstellte Verzeichnis iiop nach Galadriel.



3.2 Zugriff auf zLinux

Als nächstes müssen wir den Quellcode kompilieren. Rufen Sie wieder PuTTY auf und loggen Sie sich ein.



Java-Klassen sowie Stub und Tie werden erstellt.

```
prak219@galadriel:~/bin/iiop> tnameserv -ORBInitialPort 2012&
[1] 22629
prak219@galadriel:~/bin/iiop> Initial Naming Context:
IOR:000000000000002b49444c3a6f6d672e6f72672f436f734e616d696e672f4e616d696e67436f
6e746578744578743a312e3000000000000001000000000000007200010200000000d3133342e322e
3230352e3535000007dc000000164c4d424900000015968c45e1001000000004000000000000000
0003000000010000001800000000000001000100000001000100200001010000000000049424d0a0000
000800000001142000010000002600000000200002
TransientNameServer: setting port for initial object references to: 2012
Ready.
prak219@galadriel:~/bin/iiop>
```

Wir müssen auch hier definieren, welchen Port wir verwenden wollen. Ähnlich zur Java-Registry brauchen wir einen Eintrag in einen Corba-Name-Service. Sie müssen daher vor dem Aufruf des Servers (KontoImpl) auf Galadriel den Befehl `tnameserv -ORBInitialPort 2012&` eingeben.

`tnameserv -ORBInitialPort 2012&` ist das RMI/IIOP-Pendant zum JRMP-Registry-Aufruf `rmiregistry 2012&` aus dem JRMP-Tutorial.

Der Corba-COS- (Common Object Services) Naming-Service besteht aus einer baumartigen Ordner-Struktur für Object-References, vergleichbar mit der Ordner-Struktur für Dateien in einem Linux-Dateisystem. Der Transient-Naming-Service `tnameserv` ist eine einfache Implementierung der COSNaming-Service-Specifikation. Die Option `-ORBInitialPort` spezifiziert die Portnummer, auf der der Server Nachrichten empfängt.

Das „&“ am Ende des Command-Strings bewirkt wie im JRMP-Tutorial, dass für die Ausführung ein getrennter Hintergrundprozess gestartet wird. Die Ziffer 22629 ist die PID (Prozess-ID.

Der Begriff IOR bedeutet Interoperable Object Reference. Eine IOR beschreibt eine Objektreferenz auf ein Corba-Objekt. Der Corba-Naming-Service ist eine Möglichkeit zur Speicherung von Objekt-Referenzen unter einem Namen. IOR und Name sind vergleichbar mit DNS-IP-Adressen (z.B. 134.2.205.55) und symbolischen Namen (z.B. galadriel.cs.uni-tuebingen.de). Während aber IP-Adressen 32 Bit lang sind (128 Bit in Version 6), haben IORs typischerweise eine Länge von mehreren 100 Hex-Zeichen. In unserem Fall ist die IOR etwa 300 Zeichen lang.

IP-Adressen werden vom NIC (oder DENIC) zentral verwaltet, während IORs vom Programmierer frei vergeben werden. Die sehr langen IORs schließen die zufällige Erstellung von Synonymen praktisch aus.

Ready bedeutet, dass der Naming-Service läuft.

```
prak219@galadriel:~/bin/iiop> tnameserv -ORBInitialPort 2012&
[1] 22629
prak219@galadriel:~/bin/iiop> Initial Naming Context:
IOR:000000000000002b49444c3a6f6d672e6f72672f436f734e616d696e672f4e616d696e67436f
6e746578744578743a312e3000000000000010000000000000072000102000000000d3133342e322e
3230352e3535000007dc000000164c4d424900000015968c45e100100000000400000000000000000
0003000000010000001800000000000010001000000010001002000010100000000000049424d0a0000
0008000000011420000100000026000000020002
TransientNameServer: setting port for initial object references to: 2012
Ready.
prak219@galadriel:~/bin/iiop>
```

<Enter> bringt uns zurück zum Prompt.

```
prak219@galadriel:~/bin/iiop> tnameserv -ORBInitialPort 2012&
[1] 22629
prak219@galadriel:~/bin/iiop> Initial Naming Context:
IOR:000000000000002b49444c3a6f6d672e6f72672f436f734e616d696e672f4e616d696e67436f
6e746578744578743a312e3000000000000010000000000000072000102000000000d3133342e322e
3230352e3535000007dc000000164c4d424900000015968c45e100100000000400000000000000000
0003000000010000001800000000000010001000000010001002000010100000000000049424d0a0000
0008000000011420000100000026000000020002
TransientNameServer: setting port for initial object references to: 2012
Ready.
prak219@galadriel:~/bin/iiop> java -Djava.naming.factory.initial=com.sun.jndi.co
snaming.CNCtxFactory -Djava.security.policy=security.policy -Djava.naming.provid
er.url=iiop://galadriel.cs.uni-tuebingen.de:2012 KontoImpl
```

Als nächsten Schritt starten wir den KontoImpl-Server mit der Eingabe des folgenden Kommandos:

```
java -Djava.naming.factory.initial=com.sun.jndi.cosnaming.CNCtxFactory
     -Djava.security.policy=security.policy
     -Djava.naming.provider.url=iiop://galadriel.cs.uni-tuebingen.de:2012
     KontoImpl
```

`java KontoImpl` ruft die Klasse `KontoImpl.class` auf. Dies geschieht über drei `-Djava` Qualifier (Optionen). Djava-Qualifier werden benutzt, um Eigenschaften an die aufgerufene JVM zu übergeben. In diesem Fall sind es die Eigenschaften

- **naming.factory.initial**
- **security.policy**
- **naming.provider.url**

denen die hinter dem Gleichheitszeichen stehenden Werte übergeben werden. Bei Verwendung des Sun-JDKs ist `com.sun.jndi.cosnaming.CNCtxFactory` ein gültiger Wert für die `naming.factory.initial` Eigenschaft.

Details zu Properties finden Sie unter
http://download.oracle.com/javase/tutorial/essential/environment/properties.html .

```
prak219@galadriel:~/bin/iiop> tnameserv -ORBInitialPort 2012&
[1] 22629
prak219@galadriel:~/bin/iiop> Initial Naming Context:
IOR:000000000000002b49444c3a6f6d672e6f72672f436f734e616d696e672f4e616d696e67436f
6e746578744578743a312e3000000000000001000000000000007200010200000000d3133342e322e
3230352e3535000007dc000000164c4d424900000015968c45e1001000000004000000000000000
0003000000010000001800000000000001000100000001000100200001010000000000049424d0a0000
000800000001142000010000002600000020002
TransientNameServer: setting port for initial object references to: 2012
Ready.
prak219@galadriel:~/bin/iiop> java -Djava.naming.factory.initial=com.sun.jndi.co
snaming.CNCtxFactory -Djava.security.policy=security.policy -Djava.naming.provid
er.url=iiop://galadriel.cs.uni-tuebingen.de:2012 KontoImpl
KontoObj bound to registry
KontoServer bereit.
```

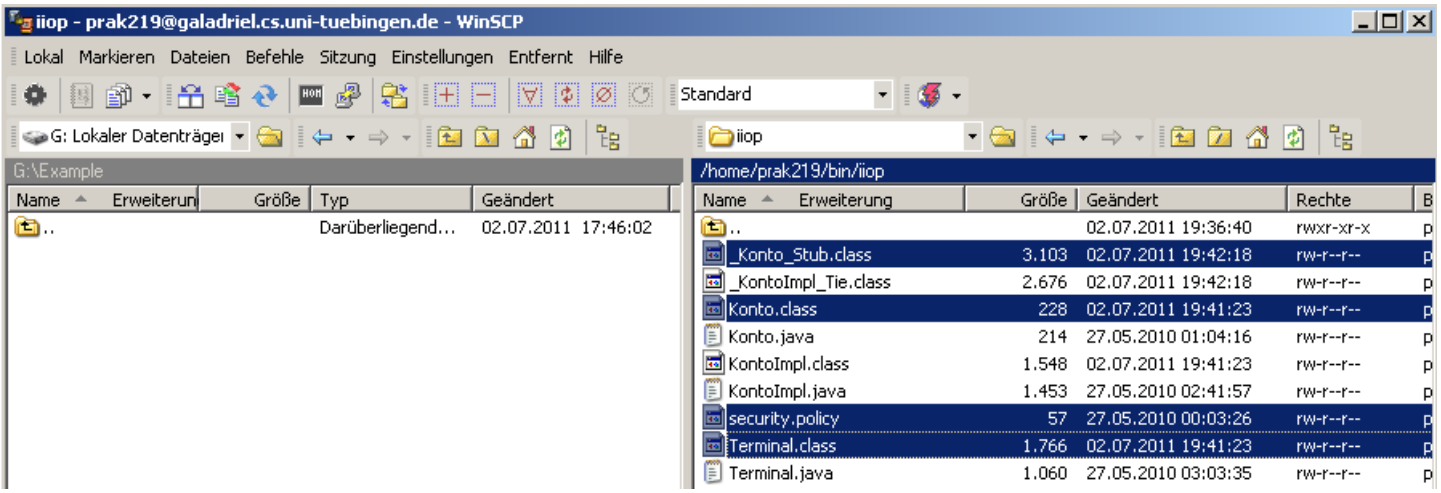
<Enter>, um das Kommando einzugeben.

Damit läuft auch unser Server und wartet auf eine Input Nachricht (über Port 2012).

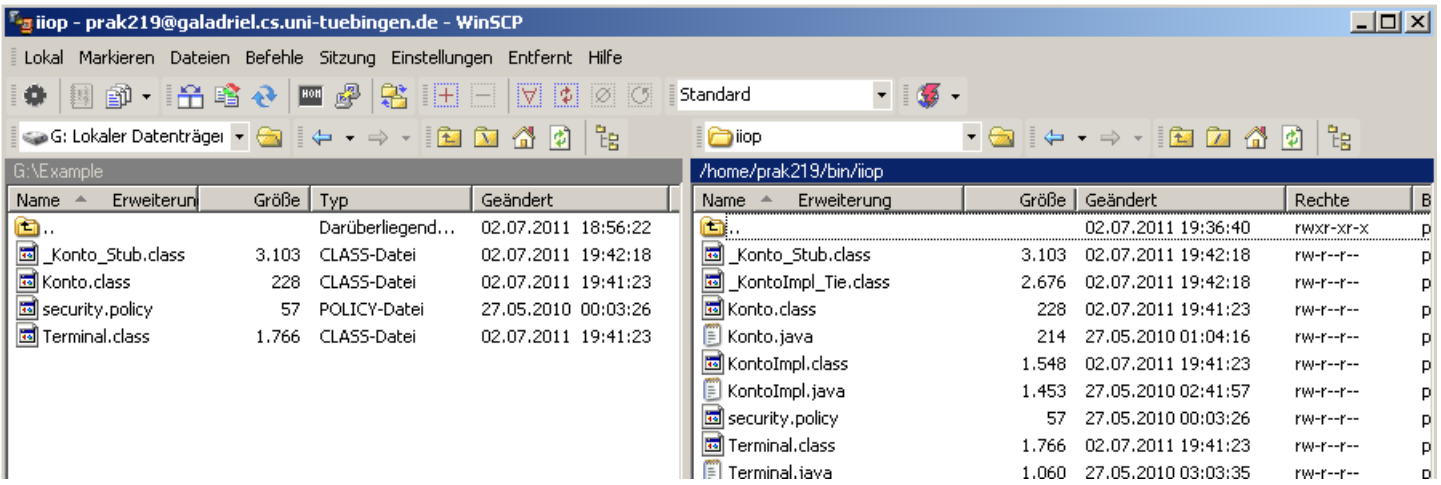
Offnen sie jetzt ein zweites Eingabeaufforderungs-Fenster.

4. Ausführen des Programms

4.1 Download der Client-Klassen



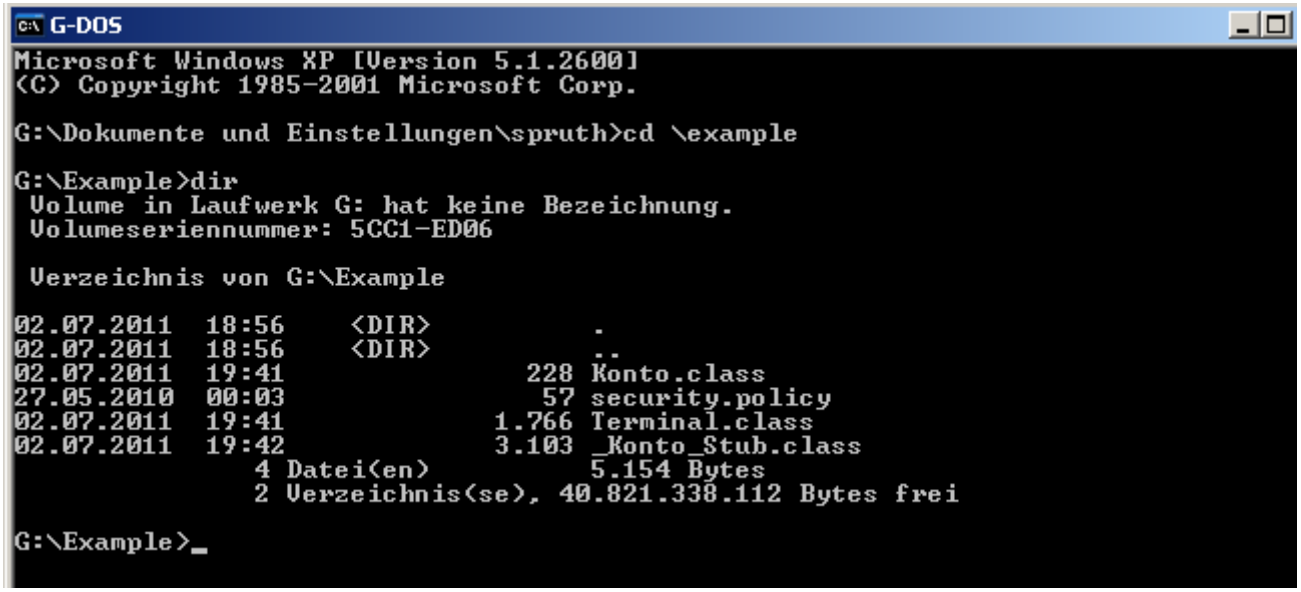
Sie haben auf dem Server Galadriel gleichzeitig auch die vom Klienten benötigten Klassen erzeugt. Kopieren Sie diese `class`-Dateien sowie Ihre `security.policy` auf Ihrem Arbeitsplatzrechner in ein Verzeichnis Ihrer Wahl, in diesem Beispiel „Example“.



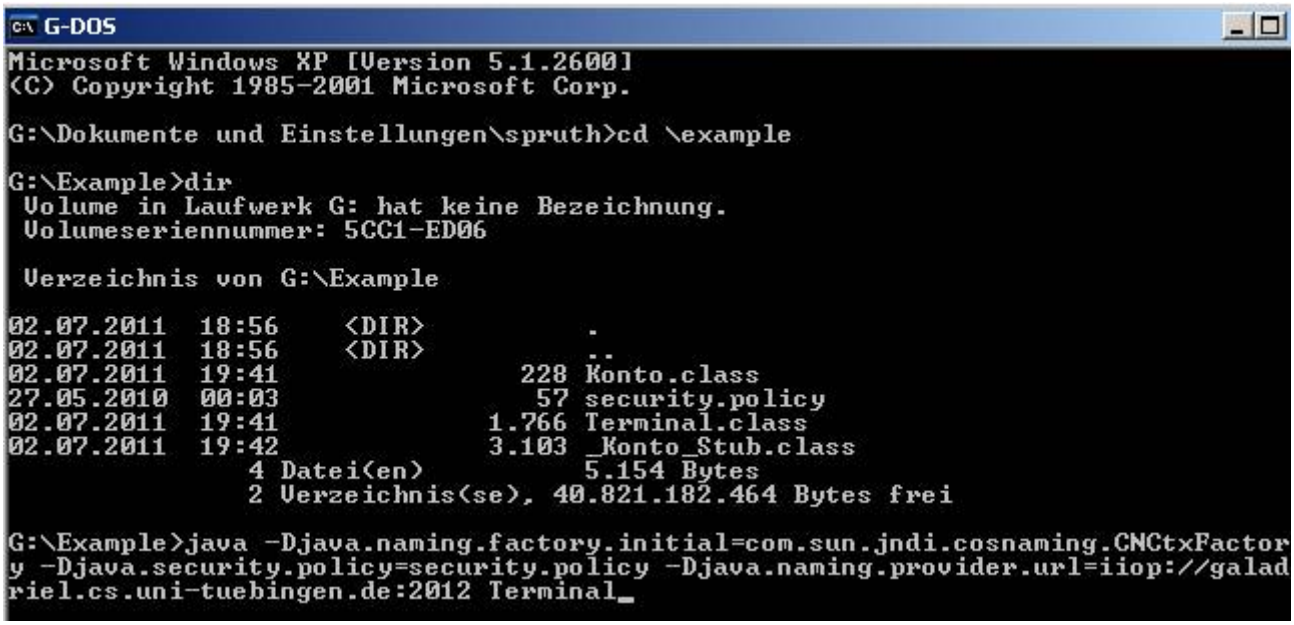
Stellen Sie sicher, dass die `classpath`-Umgebungsvariable Ihres Rechners den Namen dieses Verzeichnisses enthält.

4.2 Aufruf des Servers

Sie haben nun ein ausführbares Java-RMI-Client-Programm auf ihrer Workstation, und ein dazu passendes RMI-Server-Programm auf Galadriel. Sie können nun mit dem Client den Server ansprechen.



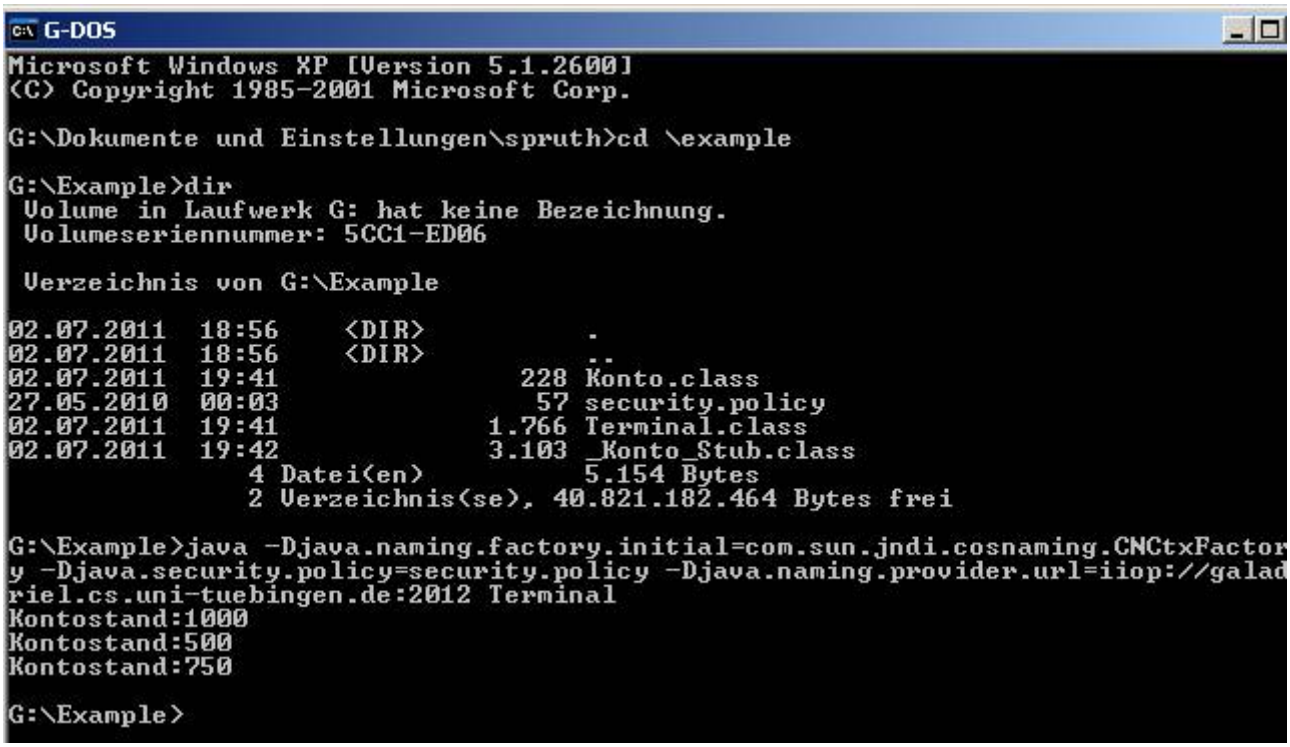
Öffnen Sie auf Ihrem Arbeitsplatz-Rechner ein zweites Eingabeauforderungs-Fenster (zusätzlich zu dem noch immer geöffneten PuTTy-Fenster). Die 4 Dateien befinden sich jetzt auf dem Client in dem Verzeichnis Example.



Mit dem Kommando

```
java -Djava.naming.factory.initial=com.sun.jndi.cosnaming.CNCtxFactory -
Djava.security.policy=security.policy -
Djava.naming.provider.url=iiop://galadriel.cs.uni- tuebingen.de:2012
Terminal
```

wird der Client gestartet.



Das Ergebnis ist hier zu sehen. Schauen Sie in Ihren Quellcode und verifizieren Sie, dass das Ergebnis Ihren Erwartungen entspricht.

Herzlichen Glückwunsch, you did it! Das Tutorial wurde erfolgreich durchgeführt.

4.3 Herunterfahren des Servers

Wichtig! Zum Abschluss ist jetzt der Server ordnungsgemäß herunterzufahren.

```
prak219@galadriel:~/bin/iiop> java -Djava.naming.factory.initial=com.sun.jndi.co
snaming.CNCtxFactory -Djava.security.policy=security.policy -Djava.naming.provid
er.url=iiop://galadriel.cs.uni-tuebingen.de:2012 KontoImpl
KontoObj bound to registry
KontoServer bereit.
```

Strg+C beendet Ihren Server.

Bitte beenden Sie am Ende wieder Ihren gestarteten tnameserv!
Hierzu ps –aef eingeben.

```
postfix  22496  1386  0 18:21 ?        00:00:00 pickup -l -t fifo -u
prak219  22629     1  0 18:45 ?        00:00:01 tnameserv -ORBInitialPort 2012
root     22804  1298  0 19:52 ?        00:00:00 sshd: prak219 [priv]
prak219  22807 22804  0 19:52 ?        00:00:00 sshd: prak219@pts/1
prak219  22808 22807  0 19:52 pts/1    00:00:00 -bash
prak219  22835 22808  0 19:52 pts/1    00:00:00 ps -aef
prak219@galadriel:~> kill 22629
```

den tnameserv Prozess mit dem kill <pid> Kommando stoppen. Die PID ist hier 22629.

```
prak219@galadriel:~/bin/iiop> exit
logout
```

Das PuTTY-Fenster kann nun geschlossen werden.

Das war es.

<u>Aufgabe:</u> ***Als Abgabe wird eine funktionierende Verison ihrer Anwendung inklusive Quellcode, eine Prozessliste der laufenden Prozesse wenn das Programm arbeitet und eine vollständige Prozessliste, wenn alle Programme beendet sind, sowie ein Screenshot der Ausgabe der Main-Methode der Terminal-Klasse erwartet. Modifizieren Sie den Quellcode, damit die Ausgabe auf dem Bildschirm etwas aussagekräftiger und/oder benutzerfreundlicher aussieht.***

5. Fragen

5.1 Nennen sie mindestens drei Unterschiede zu CORBA

- **RMI ist nur kompatibel zu Java während CORBA ein Standart für die meisten Programmiersprachen ist (solange diese IDL unterstützten).**
- **In CORBA müssen alle im Netzwerk verfügbaren Interfaces über IDL (Interface Definition Language) beschrieben sein. In Java reicht die übliche native Definition von Interfaces aus.**
- **RMI verwendet standardmäßig das JRMP (Java Remote Method Protocol); CORBA verwendet das Internet-Inter-ORB-Protokoll. Dieses ist in den meisten Fällen leistungsfähiger als JRMP.**
- **RMI-über-IIOP ist jedoch ebenfalls möglich (siehe Aufgabe RMI/IIOP).**
- **Mit JRMP können Quellcode und Objekte über das Netz übertragen werden. In CORBA können hingegen nur Datenstrukturen übertragen werden.**

5.2 Nennen sie ein Beispielszenario bei welchem der RMIClassLoader benötigt wird

- **Der RMIClassLoader wird in der Regel benötigt um Klassen aus entfernten Standorten über das Netzwerk zu laden.**
- **Wird beispielsweise eine bestimmte Klasse innerhalb eines Programms benötigt, so kann man diese (manuell) mit Hilfe der Methode *loadClass(URL,String)* in das Programm laden.**

5.3 Welche Vorteile gewinnt man durch die Nutzung von RMI-over-IIOP?

- **RMI-over-IIOP verbindet alle Vorteile von RMI und CORBA.**
- **Es wird das leistungsstärkere IIOP als Übertragunsprotokoll verwendet.**
- **Es können Quellcode und Objekte über das Netz übertragen werden.**
- **Es muss kein IDL verwendet werden um mit Objekten anderer Programmiersprachen zu kommunizieren, da das Interface nativ in Java definiert und mit Hilfe von rmic-idlin IDL überführt werden kann.**

5.4 Wofür steht JNDI? Wozu dient es?

- **Java Naming and Directory Interface.**
- **Namens- und Verzeichnisdienst über das beispielsweise Objekte innerhalb einer CORBA Anwendung gefunden werden können.**
- **In der RMI/IIOP-Aufgabe beispielsweise wird die Klasse KontoImpl mit der Option -Djava.naming.factory.initial=com.sun.jndi.cosnaming.CNCtxFactory gestartet. Dabei bedeutet der Wert com.sun.jndi.cosnaming.CNCtxFactory, dass es sich um einen CORBA-Namensdienst handelt.**

6. Anhang

6.1 KontoImpl.java

```
import java.rmi.*;
import java.rmi.server.*;

import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.rmi.PortableRemoteObject;

public class KontoImpl extends PortableRemoteObject implements Konto {

int kontostand;

protected KontoImpl() throws RemoteException {
          this.kontostand=1000;
     }

public String getKontostand() throws RemoteException {
     return "Der aktuelle Kontostand ist "+kontostand;
}

public void auszahlung(int betrag) throws RemoteException {
     kontostand=kontostand-betrag;
}

public void einzahlung(int betrag) throws RemoteException {
     kontostand=kontostand+betrag;
}

public static void main (String[] args) throws RemoteException,
NamingException {
     int port = (args.length > 0) ? Integer.parseInt(args[0]) : 2012;


     KontoImpl obj = new KontoImpl();
     String objName = "KontoObj";
     InitialContext INC = new InitialContext();
     if (System.getSecurityManager() == null) {
          System.setSecurityManager (new RMISecurityManager());
     }
     INC.rebind(objName, obj);
     System.out.println ("Konto ready.");
     }

}
```

6.2 Terminal.java

```
import java.net.MalformedURLException;
import java.rmi.*;
import java.rmi.server.*;
import javax.naming.InitialContext;
import javax.naming.NamingException;

public class Terminal {

static public void main (String[] args) throws MalformedURLException,
RemoteException, NotBoundException, NamingException {
     String host = (args.length < 1) ? "galadriel.cs.uni-tuebingen.de"
         ://"127.0.0.1" :
     args[0];
     int port = (args.length < 2) ? 2012 : Integer.parseInt(args[1]);
     try {
          if (System.getSecurityManager() == null) {
               System.setSecurityManager (new RMISecurityManager());
          }
          InitialContext INC = new InitialContext();
          Konto obj = (Konto) INC.lookup("iiop://" + host + ":" + port +
              "/" + "KontoObj");
          System.out.println (obj.getKontostand());
          obj.einzahlung(30);
          System.out.println (obj.getKontostand());
          obj.auszahlung(30);
          System.out.println (obj.getKontostand());
          }
          catch(Exception e) {
               System.out.println ("Terminal failed, caught exception " +
                   e.getMessage());
          }
     }
}
```

6.3 Konto.java

```
import java.rmi.*;

public interface Konto extends Remote {
public String getKontostand() throws RemoteException;
public void einzahlung(int betrag) throws RemoteException;
public void auszahlung(int betrag) throws RemoteException;
}
```

6.4 security.policy

```
grant {
     permission java.security.AllPermission;
};
```

CTG Tutorial
CICS Transaction Gateway

Die ursprüngliche Version dieses Tutorials entstand Nov. 2004 aus einer Diplomarbeit von Herrn Gerrit Schlüter. Erweitert mit der Primzahltestanwendung von Hanno Eichelberger und Rouven Walter, Mai 2012.

Übersicht

1. Aufgabenstellung

In der vorliegenden Übungsaufgabe werden wir eine transaktionale Anwendung entwickeln, die aus Präsentations-Logik und Business-Logik besteht. Über den Browser wird eine Zahl eingegeben, und die Anwendung ermittelt, ob es sich dabei um eine Primzahl handelt oder nicht.

Wir implementieren die Business-Logik als ein CICS-Cobol-Programm und die Präsentations-Logik als ein Java-Servlet, welches in einem WebSphere-Servlet Container läuft.

Von besonderem Interesse ist die Art der Verbindung zwischen Präsentations-Logik und Business-Logik. Wir benutzen hierfür den CICS-Intersystem-Communication-Mechanismus, welcher die ECI- oder EXCI-Schnittstelle anbietet. Um diese Verbindung nicht zu Fuß zu programmieren, verwenden wir hierfür ein vorhandenes Java-Paket, das CICS-Transaction-Gateway (CTG).

Das CICS-Transaction-Gateway ist ein sog. Connector, der die Java-Connection-Architectur (JCA) implementiert. Es existieren viele derartiger Connectoren, z. B. für die Verbindung einer Java-Präsentations-Logik mit einer Oracle-, IMS-, DB2-Stored-Procedure oder SAP-R/3-Business-Logik.

Unsere Test-Konfiguration besteht neben einem Browser auf Ihrem PC aus einem WebSphere-Web-Application-Server (WAS) und einem CICS-Transaktionsserver, wie in Abbildung 1 wiedergegeben.

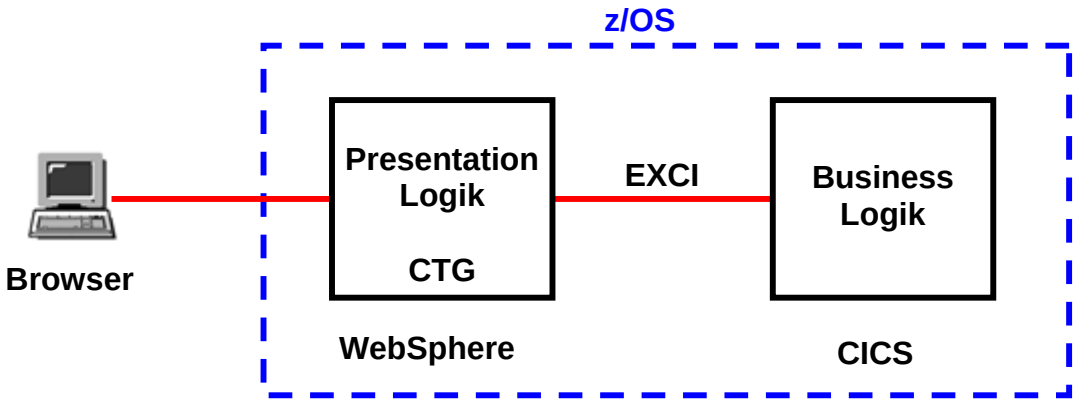


Abbildung 1: Allgemeiner Aufbau

Für eine derartige Konfiguration bestehen zwei Implementierungsalternativen. Eine integrierte Alternative ist, WebSphere und CICS in zwei getrennten Adressenräumen auf dem gleichen z/OS-System zu betreiben. In diesem Fall kann die Kommunikation zwischen beiden über eine vom z/OS-Kernel verwaltete Hauptspeicher-Queue unter Benutzung der EXCI-Schnittstelle erfolgen. Die andere „Distributed"-Alternative ist, WebSphere auf einem getrennten Linux- oder Windows-Server zu betreiben und über TCP/IP an z/OS und CICS anzuschließen. In diesem Fall benutzen wir die ECI-Schnittstelle.

Die vorliegende Übungsaufgabe benutzt die integrierte Alternative.

Selbst-Test

- **Ist das ECI-Protokoll eine CICS spezifische Implementierung des ECI-Protokolls?**
- **Würden wir auch ECI bzw. EXCI benutzen, wenn die Business-Logik genauso wie die Präsentations-Logik in Java implementiert wäre und auf einem Web-Application-Server laufen würde?**

Frage: ***Würden wir auch ECI bzw. EXCI benutzen, wenn die Business-Logik genauso wie die Präsentations-Logik in Java implementiert wäre und auf einem Web-Application-Server laufen würde?*** **Antwort:** ***Nein, wir würden RMI in diesem Fall verwenden. ECI und EXCI sind spezifisch die Schnittstellen, die für eine Kommunikation mit CICS verwendet werden.***

2. Konfiguration Details

Die in dieser Übungsaufgabe verwendete Anwendung ist recht anspruchslos: Über den Browser geben wir eine Ziffer ein, die Business-Logik ermittelt, ob es sich dabei um eine Primzahl handelt und die Präsentations-Logik teilt dem Browser das Ergebnis mit.

Das CICS-Cobol-Programm, welches die Business-Logik implementiert, erhält einen Input über die COMMAREA, ermittelt ein Ergebnis (ja/nein) und stellt es wieder in seine COMMAREA.

Die Präsentations-Logik besteht aus drei Java Server Pages (jsp), einem Servlet und einer Enterprise Java Bean (EJB). Die Java Server Page main.jsp nimmt vom Browser eine Eingabe (eine Ziffer in diesem Fall) entgegen. Eine zweite Java Server Page results.jsp liefert das Ergebnis an den Browser (Im Fehlerfall wird die Java Server Page error.jsp aufgerufen).

Die erste Java Server Page ruft über ein Form-Tag ein Servlet PrimeCTGServlet auf, welches die Präsentations-Logik enthält. Dieses benutzt eine Enterprise Java Bean CTGtesterCCIBean für die Kommunikation mit der CICS-COMMAREA mittels der EXCI-Schnittstelle. Wir verwenden EXCI an Stelle von ECI, weil WebSphere und CICS in unserem Fall unter dem gleichen z/OS-Kernel laufen. CTGtesterCCIBean enthält den CICS-Transaction-Gateway (CTG)-Code für die Kommunikation mit der CICS-COMMAREA.

Diese Zusammenhänge sind in Abb. 2 dargestellt.

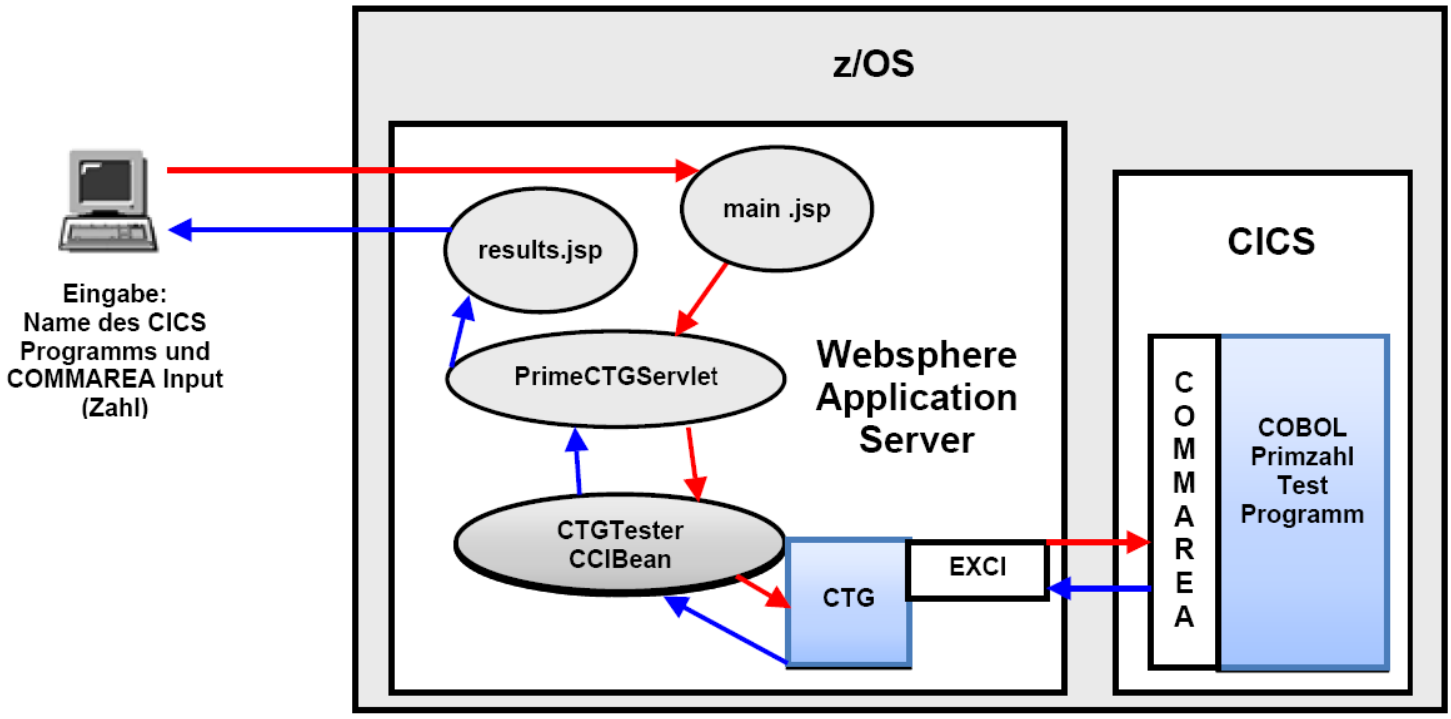


Abbildung 2: Überblick Primzahltester mit WAS, CTG und CICS

Selbst-Test

- **Läuft das CTG innerhalb des WAS-EJB-Containers?**

3. Voraussetzungen

Dieses Tutorial ist für den Tübinger Mainframe "Hobbit" (134.2.205.54) vorgesehen. An vielen Stellen taucht im Tutorial Text die Nummer "218" auf. Bitte ersetzen Sie diese durch Ihre eigene Praktikums ID. Angenommen Ihre ID ist prak238, dann ersetzen Sie zum Beispiel "PRAK218" durch "PRAK238" oder "CTG218" durch "CTG238".

4. Vorgehensweise

Zuerst erstellen und installieren wir ein CICS Cobol-Programm namens "CTG218", welches über die CICS-COMMAREA (Communications Area) eine Zahl übertragen bekommt und diese auf die Primzahleigenschaft testet. Das Ergebnis der Berechnung schreibt das Programm zurück in die COMMAREA.

Die Funktionsfähigkeit des erstellten und installierten CICS-Programms wird danach mit einem Browser und einem speziellen Servlet getestet. Der Anwender kann in einer HTML-Oberfläche eine Zahl eingeben, welche über das Servlet, eine Enterprise Bean und die COMMAREA zum CICS-Programm übertragen wird. Das Ergebnis des Primzahltests wird über denselben Weg zurückübertragen und im Browser angezeigt (siehe Abbildung 2).

CICS und WebSphere sind auf Hobbit bereits vorinstalliert. Zur Erleichterung haben wir das Servlet PrimeCTGServlet, die EJB CTGtesterCCIBean sowie die drei .jsp Pages main.jsp, results.jsp und error.jsp unter WebSphere bereits vorinstalliert; Sie brauchen also nur noch aufgerufen zu werden.

5. Erstellung des CICS-Programms

Zuerst soll das CICS Cobol Programm erstellt werden, auf welches das CICS Transaction Gateway später zugreifen soll. Der Unterschied zu den bisherigen CICS-Programmen aus der Aufgabe „Cobol Hello World unter CICS“ und „Datenbankzugriff mit CICS“ besteht darin, dass die Ausgabe nicht in Form von BMS-Maps auf dem Bildschirm erscheint. Stattdessen wird die Ausgabe in die CICS COMMAREA (Communications Area) geschrieben, die zum Datenaustausch zwischen den CICS-Programm und dem CICS Transaction Gateway verwendet wird (Abbildung 2).

Um das CICS-Programm zu erstellen, arbeiten wir sowohl mit dem TSO- als auch mit dem CICS-Subsystem. Wir verwenden am Anfang TSO, so dass wir mit dem Befehl "L TSO" eine TSO-Sitzung starten (Abbildung 2) und uns mit unseren Benutzerdaten anmelden.

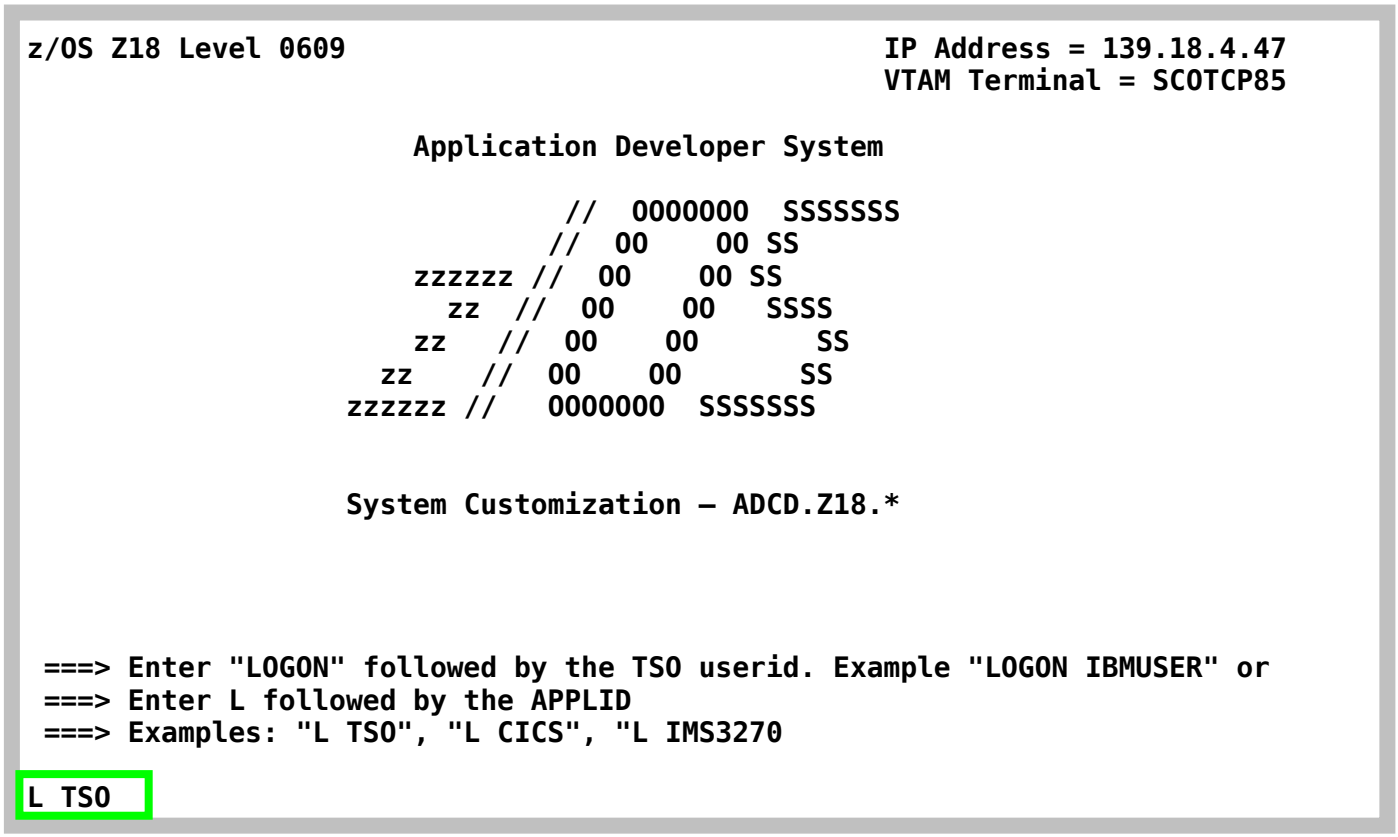


Abbildung 3: Start der TSO-Sitzung

Zuerst muss noch ein Dataset erstellt werden. Wir öffnen den "Data Set Utility" Screen und erstellen (Allocate) einen neuen Partitioned Dataset namens PRAK218.CICS.CTG. Dabei verwenden wir die in der nachstehenden Aufgabe angegebenen Parameter.

Aufgabe: ***Legen Sie den Dataset "PRAK218.CICS.CTG" an. Verwenden Sie dazu folgende Parameter:***

Space units ..... KB	***Record format .... FB***
Primary quantity .. 16	***Record length 80***
Secondary quantity 1	***Block size 320***
Directory blocks .. 2	***Data set name type: PDS***

Unsere Anwendung besteht aus einem JCL-Script, welches ein Cobol-Programm enthält. Das JCL-Script dient zur Übersetzung des Cobol-Programms. Wir erstellen dieses im neuen Partitioned Dataset "PRAK218.CICS.CTG".

In diesem Fall enthält das CICS-Programm lediglich die Businesslogik. Die Präsentationslogik wird später vom WebSphere-Application-Server in Form einer Webanwendung bereitgestellt.

5.1 Das Cobol Programm

Wir beginnen mit dem Quellcode des CICS-Cobol-Programms und editieren ein Member "CTG218" für den neu angelegten Dataset "PRAK218.CICS.CTG" (siehe Abbildung 4 und 5).

```
File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------
EDIT       PRAK218.CICS.CTG(CTG218) - 01.18               Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 //PRAK218C JOB (),CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1),NOTIFY=&SYSUID
000200 //CALLPROC EXEC PROC=DFHYITVL,
000300 // PROGLIB=DFH310.CICS.PRAKLOAD
000400 //*
000500 //TRN.SYSIN DD *
000600        IDENTIFICATION DIVISION.
000700        PROGRAM-ID. CTG218.
000800        ENVIRONMENT DIVISION.
000900        DATA DIVISION.
000901
000910       *DEKLARATION VON VERWENDETEN VARIABLEN
001000        WORKING-STORAGE SECTION.
001100        01 NAME-TAB.
001200           02 PRIME       PICTURE X(20) VALUE "PRIME".
001210           02 NPRIME      PICTURE X(20) VALUE "NPRIME".
001301           02 B           PICTURE 9(7).
001302           02 C           PICTURE 9(7).
001303           02 I           PICTURE 9(7).
001304           02 D           PICTURE 9(7).
001305
001306       *VERKNUEPFUNG ZUR COMMAREA UEBER VARIABLE NR
001310        01 COMMAREA-IN.
001320           03 NR          PICTURE X(7).
001400        LINKAGE SECTION.
001500        01 DFHCOMMAREA.
001600           03 RES         PICTURE X(20).
001700
001710       *DAS PROGRAMM
001800        PROCEDURE DIVISION.
001801
001802       *    KOPIERE INHALT DER COMMAREA IN VARIABLE NR
001810            MOVE DFHCOMMAREA TO COMMAREA-IN.
001900            MOVE SPACES TO DFHCOMMAREA.
001901
Command ===>                                                 Scroll ===> PAGE
 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Abbildung 4: Quelltext des CICS-Programms, Panel 1/2

```
   File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       PRAK218.CICS.CTG(CTG218) - 01.18               Columns 00001 00072
 001902
 001903       *    KONVERTIEREN DES COMMAREATEXT IN NUMMERISCHE ZAHL
 001904            COMPUTE B = FUNCTION NUMVAL(NR).
 001905
 001906       *    TESTEN OB ZAHL = 1
 001907            IF B = 1 PERFORM STOP-NPRIM.
 001908
 001909       *    ZAHL DURCH ALLE KLEINEREN ZAHLEN TEILEN
 001910       *    UND CHECKEN OB ERGEBNIS GERADE
 001911            DIVIDE B BY 2 GIVING C ROUNDED.
 001912            PERFORM CHECK-PARA VARYING I FROM 2 BY 1 UNTIL I > C.
 001913
 001914            MOVE PRIME TO RES.
 001915            EXEC CICS RETURN
 001916                    END-EXEC.
 001917            EXIT.
 001918
 001919       *PROC WIRD AUFGERUFEN WENN ZAHL NICHT PRIMZAHL IST
 001930        STOP-NPRIM.
 001940            MOVE NPRIME TO RES.
 001950            EXEC CICS RETURN
 001960                    END-EXEC.
 001970            EXIT.
 001971
 001972       *PROC TEILT ZAHL DURCH TEILER UND UEBERPRUEFT REST
 001980        CHECK-PARA.
 001990            DIVIDE B BY I GIVING C REMAINDER D.
 002000            IF D = 0 PERFORM STOP-NPRIM.
 002500 /*
 002600 //LKED.SYSIN DD *
 002700   NAME CTG218(R)
 002800 /*
 002900 //
****** **************************** Bottom of Data ****************************

 Command ===>                                                Scroll ===> PAGE
  F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
  F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Abbildung 5: Quelltext des CICS-Programms, Panel 2/2

Das JCL-Script ruft die Prozedur "DFHYITVL" auf, die alle Übersetzungsschritte für das CICS-Programm enthält: CICS-Übersetzer, Cobol-Compiler und Linker.

5.2 COMMAREA

Das aufrufende Programm (Java-Presentation-Logik) übergibt seine Eingabe-Daten an die COMMAREA des aufgerufenen Cobol-Programms. Das Cobol-Programm gibt sein Ergebnis über die COMMAREA an das aufrufende Präsentationslogik-Programm zurück. Das CTG implementiert die Verbindung zwischen Präsentationslogik und Business-Logik.

Im Cobol-Quelltext wird die COMMAREA durch die "DFHCOMMAREA" Angabe in der "LINKAGE SECTION" in Zeile 001500 definiert. Sie können beliebige Namen in der Working-Storage-Section benutzen, aber der Name "DFHCOMMAREA" in der Linkage-Section ist Pflicht. DFHCOMMAREA dient zum Datenaustausch zwischen dem CICS-Programm und dem CICS-Transaction-Gateway. In unserem Fall enthält die COMMAREA nur ein Feld „RES“ für die zu testende Ziffer.

In der Procedure Division kopiert das Cobol Progamm den Inhalt der COMMAREA in die Variable COMMAREA-IN (Zeile 001810), berechnet seine Antwort, und kopiert das Ergebnis zurück in die COMMAREA in das Feld „RES“ (Zeile 001914 bzw. Zeile 001940). Zeile 001900 löscht vorher den Inhalt von DFHCOMMAREA.

Hierzu muss die COMMAREA-Struktur sowohl in der Working-Storage-Section, als auch in der Linkage-Section des Cobol-Programms deklariert werden. In der Linkage-Section muss die Definition als erste Deklaration erscheinen.

Bemerkung: Source-Code-Kommentare werden in z/OS COBOL mit einem „*“ oder „/“ in Spalte 7 definiert.

Die COMMAREA-Programmierung ist recht einfach zu erstellen. Ein Nachteil ist, dass die COMMAREA in der Größe auf 32 KByte begrenzt ist. Als Alternative kann eine als „Channels and Containers“ bezeichnete Lösung eingesetzt werden. Ein Tutorial hierüber ist zu finden unter http://docweb.cns.ufl.edu/news/n0510/Channels_Containers.pdf.

Selbst-Test

- **Warum wird die COMMAREA sowohl in der Working-Storage-Section als auch in der Linkage-Section definiert?**

Aufgabe: *Geben Sie das COBOL-Programm ein und übersetzen Sie es anschließend mit dem Befehl "SUB" in der Kommandozeile.*

5.3 Übersetzung des Cobol Programms

```
14.11.21 JOB07408 $HASP165 PRAK218C ENDED AT N1  MAXCC=0 CN(INTERNAL)
***
```

Abbildung 6: Bestätigung der Jobverarbeitung

"MAXCC=0" heißt, dass der Job erfolgreich ausgeführt wurde. Das CICS-Programm wurde nun übersetzt und liegt in ausführbarer Form vor.

Nun muss das übersetzte Programm in CICS eingebunden (definiert und installiert) werden. Dazu loggen wird uns im Anmeldebildschirm mittels "L CICS" ein {Abbildung 7).

5.4 Installation des übersetzten Cobol Programms in CICS

```
z/OS Z18 Level 0609                                IP Address = 139.18.4.47
                                                   VTAM Terminal = SCOTCP85

                     Application Developer System

                              //  OOOOOOO  SSSSSSS
                             //  OO    OO SS
                     zzzzzz //  OO    OO SS
                       zz  //  OO    OO   SSSS
                     zz   //  OO    OO       SS
                   zz    //  OO    OO       SS
                 zzzzzz //   OOOOOOO  SSSSSSS


                 System Customization - ADCD.Z18.*




 ===> Enter "LOGON" followed by the TSO userid. Example "LOGON IBMUSER" or
 ===> Enter L followed by the APPLID
 ===> Examples: "L TSO", "L CICS", "L IMS3270

L CICS
```

Abbildung 7: Einloggen ins CICS

Nachdem wir unter unserem Mainframe-Login eingeloggt sind, löschen wir noch durch Betätigung der (eventuell emulierten) CLEAR-Taste das CICS-Fenster. Nun steht der Cursor in der linken oberen Ecke und wartet auf die Eingabe einer CICS-Transaktion.

Eine evtl. bereits vorhandene CICS-Group „PRAK218“ muss mit Hilfe von „CEDA DELETE ALL GROUP(PRAK218)“ gelöscht werden.

Wir definieren das Programm mit Hilfe des Befehls "CEDA DEFINE PROGRAM(CTG218) GROUP(PRAK218)" (siehe Abbildung 8).

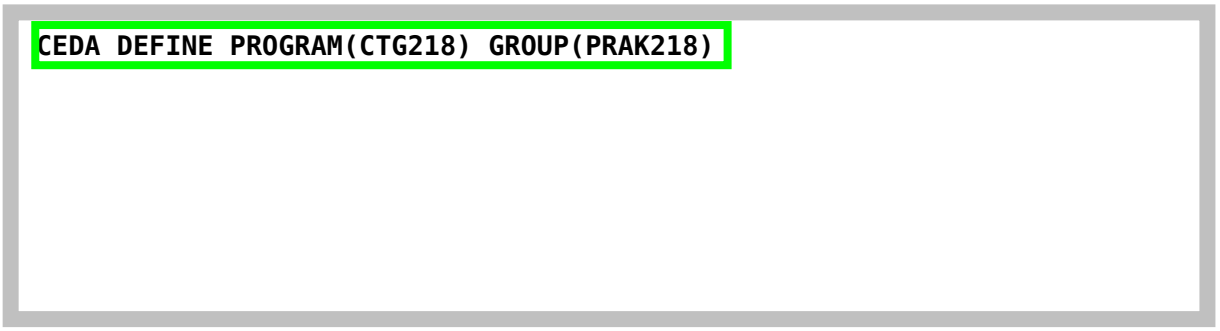


Abbildung 8: Definition des Programms

CICS möchte von uns nun einige Parameter für das Programm wissen. Wir ändern lediglich die Sprache in "Le370" und bestätigen mit der Eingabetaste (Abbildung 9)

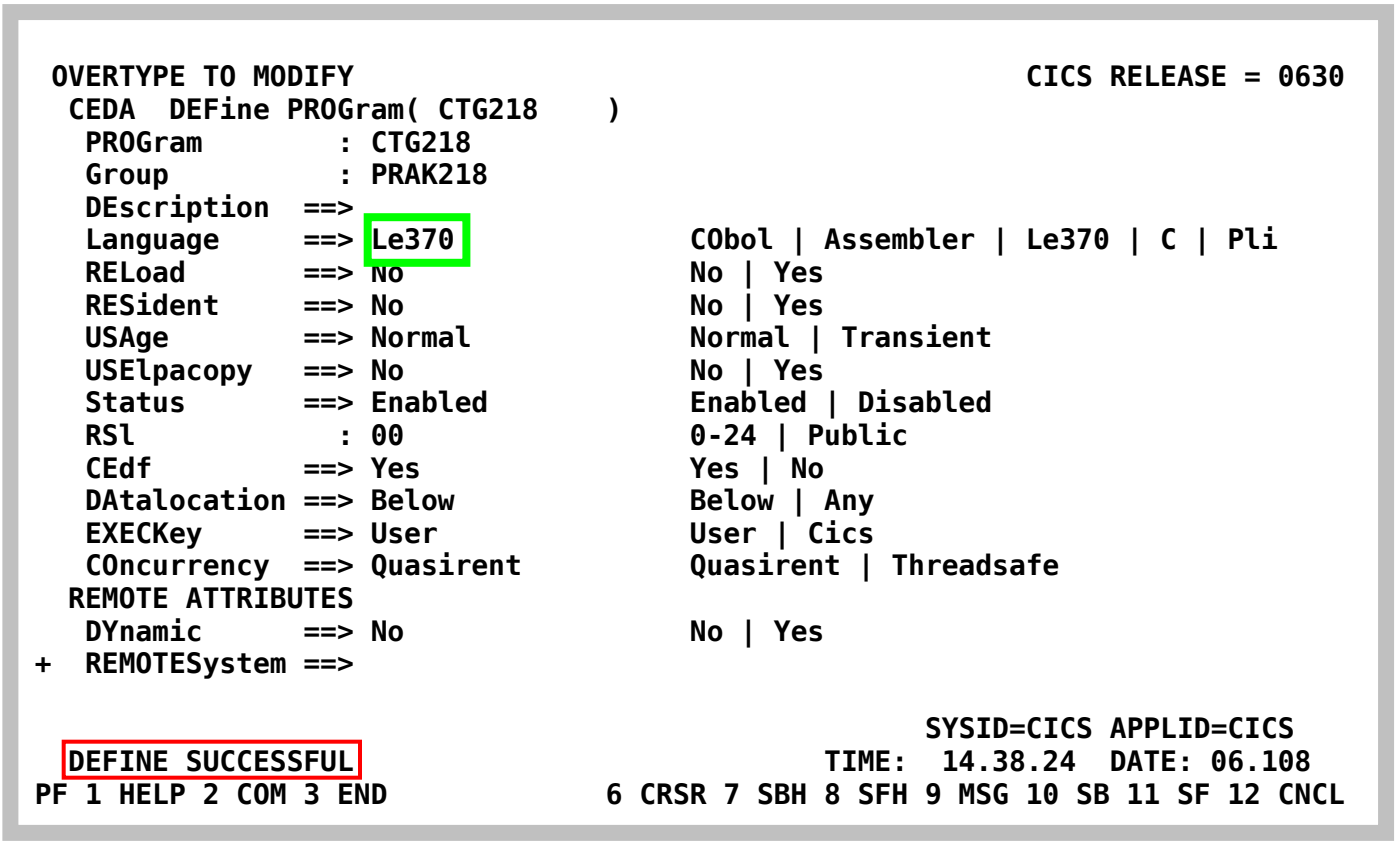


Abbildung 9: Parameter des CICS-Programms

Wir installieren daraufhin die Gruppe "PRAK218" neu, indem wir den CEDA-Befehl "CEDA INSTALL GROUP(PRAK218)" eingeben (Abbbildung 10).

.

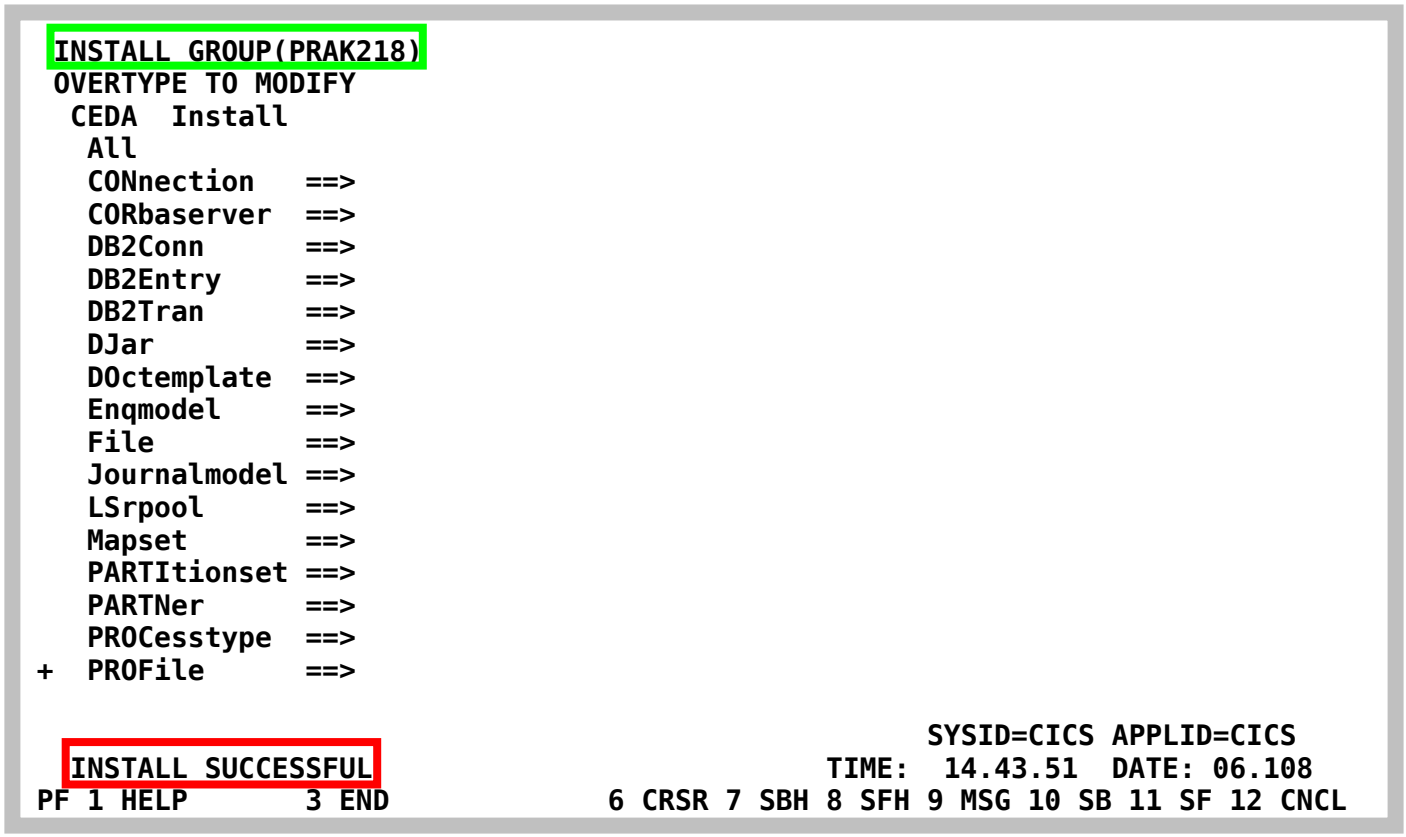


Abbildung 8: Installation der Gruppe

6. Test der Konfiguration

Nachdem nun das CICS-Primzahltestprogramm compiliert und in CICS definiert und installiert wurde, möchten wir den Aufruf von diesem über das CICS-Transaction-Gateway testen. Der Test erfolgt in einem normalen Webbrowser Ihrer Wahl.

Wir haben die beiden Java Server Pages main.jsp und results.jsp sowie das Servlet PrimeCTGServlet zu einem Web-Archiv (.war) zusammengefasst und in WebSphere im Verzeichnis PrimeCTGWeb untergebracht. Nachrichten werden über Port 9527 entgegengenommen. In den Webbrowser ist als Adresse

http://hobbit.cs.uni-tuebingen.de:9527/PrimeCTGWeb/main.jsp

einzugeben. So wird der CTG-Primzahltester "PrimeCTG" gestartet.

Nun müssen den Namen ihres CICS-Primzahltestprogramm und eine zu testende Zahl eingeben. Per Klick auf den Submit-Button wird der Test des Zugriffs auf das CICS-Programm über das CICS Transaction Gateway gestartet (Abbildung 11).

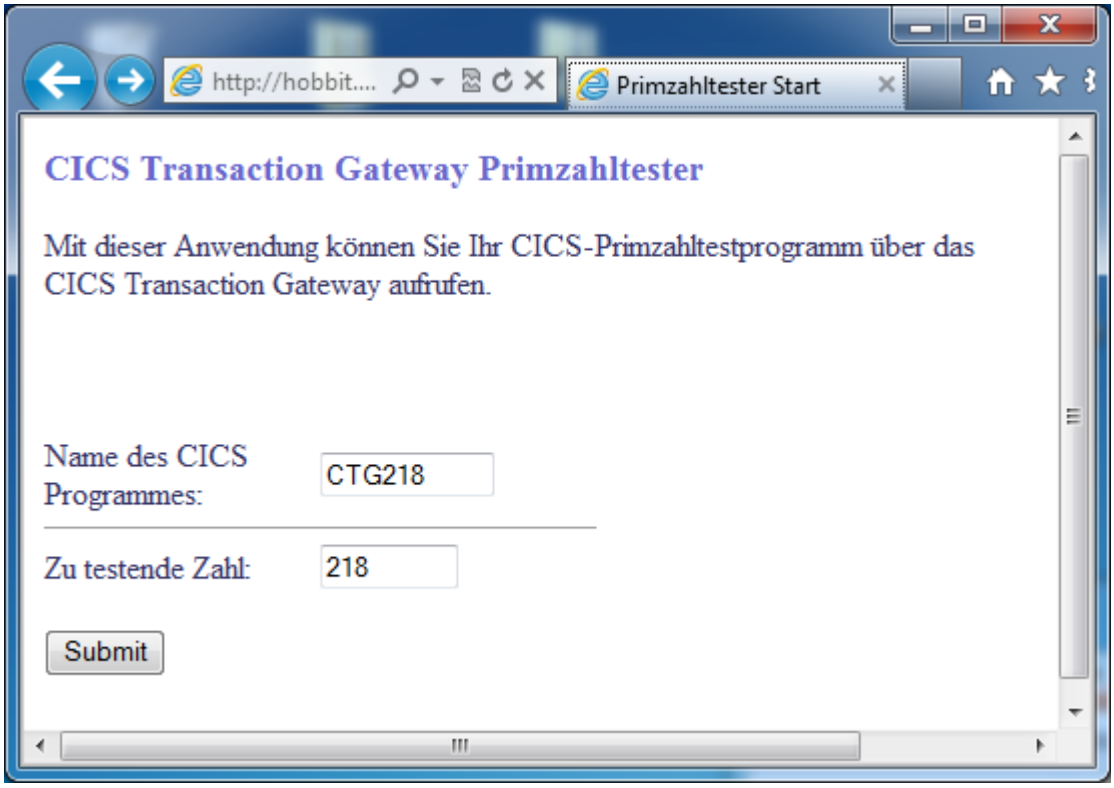


Abbildung 11: Aufruf des Primzahltest über Browser

Als Ergebnis erscheint die Information, ob die Zahl Primzahl ist (PRIME) oder nicht (NPRIME), siehe Abbildung 12.

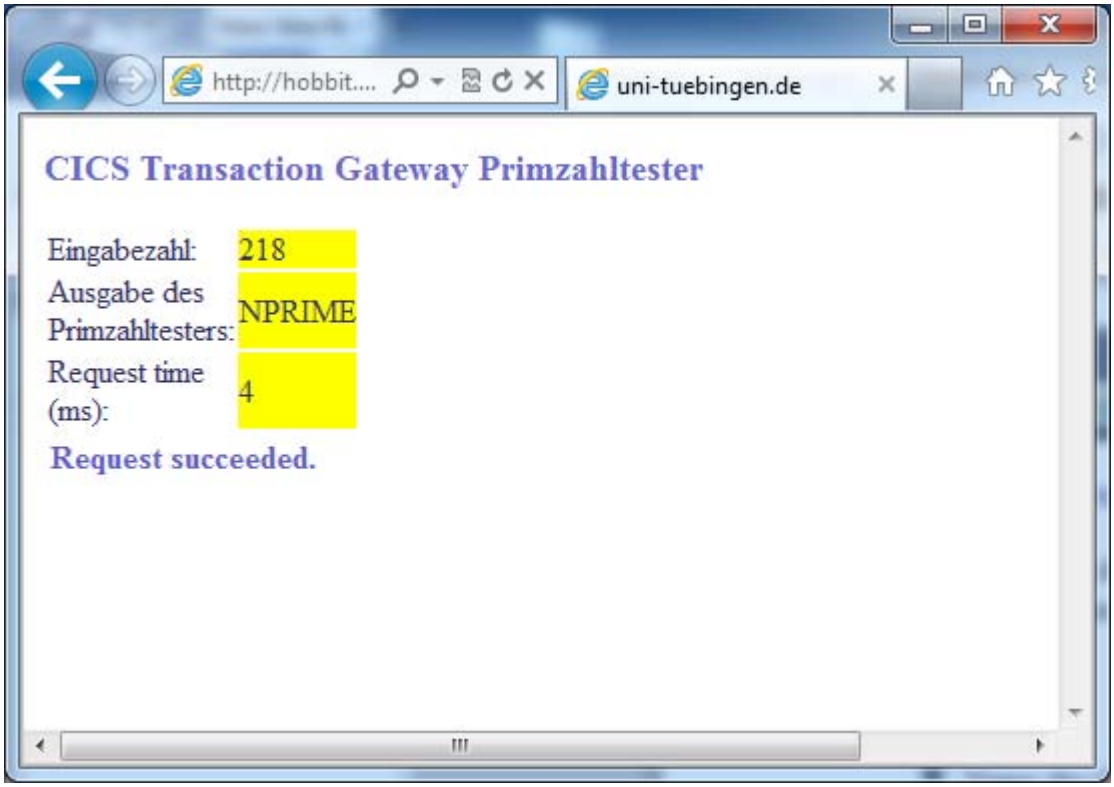


Abbildung 12: Ergebnis des Primzahltests

Aufgabe: *Starten Sie die CTG-Primzahltest-Oberfläche „PrimeCTG".*

Aufgabe: *Geben Sie den Namen Ihres CICS-Primzahltestprogramms und eine zu testende Zahl ein (möglichst den nummerischen Teil Ihrer PRAK-ID). Machen Sie einen Screenshot vom Testergebnis.*

Aufgabe: *Nachdem Ihnen ein erfolgreicher Test gelungen ist, weisen Sie dies bitte anhand eines Screenshots nach (JPEG-Format, max. Größe 99 KB). Dieser soll den gesamten Inhalt des Browser-Fensters zeigen.*

Aufgabe: *Schicken Sie diesen Screenshot im JPEG-Format an Ihren Betreuer.*

Aufgabe: *Loggen Sie sich sauber aus dem CICS aus. Dies geschieht mit dem Befehl "CESF LOGOFF". Details dazu sind in Aufgabe 2, C-Version, beschrieben. Das Schließen des 3270-Emulator-Fensters ohne ein solches sauberes Ausloggen aus CICS ist ebenfalls verboten, weil dies auch CICS zum Absturz bringen könnte, was ein Bearbeiten der CICS-Tutorien für Sie und andere bis auf weiteres unmöglich machen würde.*

7. Detail Information

Die für die Dateneingabe benutzte Java Server Page main.jsp ist in Abb. 11 dargestellt.

```
<FORM ACTION="PrimeCTGServlet" METHOD="POST">
Mit dieser Anwendung können Sie Ihr CICS-Primzahltestprogramm über das CICS
Transaction Gateway aufrufen.<P></P><TABLE border="0" cellpadding="0"
cellspacing="0">
        <TBODY>
                <TR>
                        <TD width="138">Name des CICS Programmes:</TD>
                        <TD width="138"><INPUT type="TEXT" name="funcName" size="10"
                                maxlength="6" value="ECIPROG"></TD>
                </TR>
                <TR>
                        <TD colspan="3">
                        <HR noshade size="1">
                        </TD>
                </TR>

                <TR>
                        <TD width="138">Zu testende Zahl:</TD>
                        <TD colspan="2"><INPUT type="TEXT" name="commareaInput"
                                size="7"
                            value=""></TD>
                </TR>



        </TBODY>
</TABLE><P>
<INPUT TYPE="SUBMIT" VALUE="Submit">
</FORM>
```

Abbildung 13: Form Tag Teil von main.jsp

Selbst-Test
Schauen sie sich den Code des Form Tags in Abb. 13 an:

- **Welche Variable wird mit dem Namen „funcName“ gekennzeichnet?**
- **Welche Variable wird mit dem Namen „commareaInput“ gekennzeichnet?**

Abbildung 14 zeigt einen Ausschnitt von results.jsp mit den Java Expressions.

```
<TR>
        <TD width="121">Eingabezahl:</TD>
        <TD BGCOLOR=yellow width="97">
                <%= commareaInput %></TD>
</TR>

<TR>
        <TD width="121">Ausgabe des Primzahltesters:</TD>
        <TD BGCOLOR=yellow width="97"><%= results %></TD>
</TR>

<TR>
        <TD width="121">Request time (ms):</TD>
        <TD BGCOLOR=yellow width="97"><%= requestTime %></TD>
</TR>
```

Abbildung 14: Aufruf der Java Expressions in results.jsp

Die Zeichenfolgen <%= und %> schließen eine Java Epression ein

Selbst-Test

- **Wo befindet sich der Code für die drei Java Aufrufe „commareaInput“, „results“ und „requestTime“?**

Abbildung 15 zeigt einen Ausschnitt von error.jsp.

```
<jsp:useBean class="java.lang.String" id="errorException" scope="request"/>

<BODY>
Es ist ein Fehler aufgetreten. Probieren Sie es noch einmal, oder kontaktieren Sie Ihre
Betreuer.

<P>
<TABLE BGCOLOR=red>
<TBODY>
        <TR><TH>Request failed:</TH></TR>
        <TR>
                <TD VALIGN=TOP>Exception occurred:</TD>
                <TD BGCOLOR=YELLOW><%= errorException %></TD>
        </TR>
</TBODY></TABLE></BODY>
```

Abbildung 15: Aufruf der Java Expressions in error.jsp

Abbildungen 16 und 17 zeigen einen Ausschnitt aus PrimeCTGServlet.java .

```
public void processRequest(HttpServletRequest request,HttpServletResponse response)
throws IOException, ServletException {

/**
* Konvertierung der Eingabezahl.
*/
  String number = request.getParameter("commareaInput")+"";
  int intNumber = 1;
  try{
    intNumber = Integer.parseInt(number.replaceAll("^0*",""));
  }catch(Exception e){}
  number = ""+intNumber;
  for (int i=0;i<7-(intNumber+"").length();i++)
    number = "0"+number;

/**
* Laden der Eingabezahl in ECIRequest Input.
*/
  // retrieve HTTP request parameters
  MapRequest2Details(request, eciReq1, "");
  eciReq1.commarea=number;

  ……

/**
* Aufruf der CTGTesterCCIBean mit ECIRequest Input als Parameter.
*/
  Context ic = new InitialContext();
  Object or = ic.lookup("java:comp/env/ejb/CTGTesterCCI");
  CTGTesterCCI tester = null;
  if (or != null) {
     CTGTesterCCIHome home = (CTGTesterCCIHome) PortableRemoteObject.narrow
                            (or,CTGTesterCCIHome.class);
     tester = home.create();
  }
  beforeReqTime = new Date();
  String results = tester.execute(eciReq1);
  …
```

Abbildung 16: Ausschnitt aus PrimeCTGServlet.java
Teil 1/2

```
/**
* Get Connection to CICS aus WAS Connection Factory
*/
public void ejbCreate() {
  javax.naming.Context ic=null;
  ic = new javax.naming.InitialContext();
  Connection eciConn = null;
  String cfRef = new String("java:comp/env/ECI");
  cf = (ConnectionFactory) ic.lookup(cfRef);
   ….
}


/**
* Aufruf von CICS Programm mittels CICS Connection aus Connection Factory
*/
private JavaStringRecord flowEciRequest(…){
  eciConn = getConnection(eciRD, cf);
  eciInt = getInteraction(eciConn);

  eciInt.execute(eSpec, jsr, jsr);

}
```

Abbildung 17: Ausschnitt aus PrimeCTGServlet.java
Teil 2/2

Selbst-Test
Schauen sie sich den Code des Form Tags in Abb. 13 an:

- **Welche Variable wird mit dem Namen „funcName“ gekennzeichnet?**
- **Welche Variable wird mit dem Namen „commareaInput“ gekennzeichnet?**

Der auf dem z/OS-WebSphere-Application-Server vorinstallierte Code besteht aus 7 Komponenten:

1. **main.jsp**
   ist die Java Servlet Page, die vom Browser aufgerufen wird.

2. **results.jsp**
   ist die Java Servlet Page, die an den Browser zurück gegeben wird

3. **error.jsp**
   ist die Java Servlet Page, die angezeigt wird, wenn ein Fehler aufgetreten ist

4. **PrimeCTGServlet.java**
   ist das Servlet, welches von main.jsp aufgerufen wird.

Diese 3 Komponenten wurden zu einem Web Archiv (.war) zusammengefasst, und im Verzeichnis PrimeCTGWeb installiert.

5. **CTGTesterCCIBean.java**
   Ist die EJB, welche das CICS Transaction Gateway enthält

6. **CTGTesterCCIHome.java**
   Ist die Home Interface von CTGTesterCCIBean.java

7. **CTGTesterCCI.java**
   Ist die Object Interface von CTGTesterCCIBean.java

Den vollständigen Code können Sie unter
http://www.cedix.de/Vorles/Band3/javazos/ctg/index.html

herunterladen.

Falls Sie dies interessiert, können Sie die Websphere Console unter "http://134.2.205.54:9525/ibm/console" finden, Als Benutzername können Sie einen beliebigen Werte eingeben, da das System nicht geschützt ist.

Das Tutorial ist eine Erweiterung eines Beispiels welches in dem folgenden Redbook enthalten ist:

CICS Transaction Gateway V5, The WebSphere Connector for CICS
August 2002, SG24-6133-01, Kapitel 10,
http://www.redbooks.ibm.com/redbooks/pdfs/sg246133.pdf
oder www.cedix.de/VorlesMirror/Band3/Java04.pdf

Und das war es, Sie haben erfolgreich das CTG Tutorial bearbeitet. Herzlichen Glückwunsch !

JCICS Tutorial Teil 1
CICS Integration mit Java

Dank an Herrn Stefan Huster für die Bereitstellung des Tutorials

Übersicht

1. Voraussetzungen, Einrichtung und Verwendung der Systemumgebung

In diesem Abschnitt wird die verwendete Entwicklungsumgebung besprochen. Es wird Ihnen gezeigt welche Software Sie für die Bearbeitung dieser Tutorials benötigen und welche Einstellungen Sie innerhalb dieser Programme vornehmen müssen. Am Ende dieses Abschnitts finden Sie eine Anleitung über die Einrichtung der Entwicklungsumgebung und dem An- und Abmeldeprozess unter CICS.

1.1. Verwendete Software

Die in diesem Abschnitt aufgelistete Software muss für eine erfolgreiche Bearbeitung dieser Tutorials auf Ihrem Computer installiert sein.

1.1.1 Java Development Kit

Grundvoraussetzung für die Entwicklung von Java-Anwendungen ist das Java Development Kit (JDK). Dieses enthält den notwendigen Compiler und eine entsprechende Laufzeitumgebung. Für Windows und Linux ist dieses auf der Webseite http://www.java.com/de/download/ zum kostenlosen Download erhältlich. Nutzer von MacOs benötigen die von Apple bereitgestellte Implementierung des JDK. Diese kann im Apple-Entwicklerbereich unter http://developer.apple.com/ gefunden werden. Dieses Tutorial verwendet die Java-Version 1.5.

Die Entwicklung von Java-Anwendungen erfordert theoretisch keine spezielle Entwicklungssoftware und ist mit jedem Texteditor möglich. Dennoch kann diese durch die Verwendung von integrierten Entwicklungsumgebungen (IDEs) erheblich vereinfacht werden. Jeder Programmierer hat seine eigenen Vorlieben, geht es um die Wahl der verwendeten IDE. Dieses Tutorial wurde auf Basis von Eclipse erstellt. Eclipse ist auf der Webseite http://www.eclipse.org/ für alle gängigen Betriebssysteme erhältlich. Natürlich können die hier vorgestellten Tutorials auch mit anderen IDEs wie z.B. Netbeans bearbeitet werden.

1.2 FTP-Client

Der Transfer der erstellten Programme von Ihrem Computer zum Mainframe erfolgt über das FTP-Protokoll. Hierfür benötigen Sie einen entsprechenden FTP-Client. Filezilla ist ein Beispiel für einen kostenlosen FTP-Client, der für alle gängigen Betriebssysteme unter http://www.filezilla.de/ erhältlich ist. Natürlich können Sie auch jeden anderen FTP-Client verwenden. Voraussetzung ist nur, dass dieser SFTP unterstützt.

1.3 TN3270-Emulator

IBM Mainframes verwenden für die Kommunikation 3270-Terminals, welche eine spezielle Variante des Telnet-Protokolls benutzen. Damit Sie sich von Ihrem Computer auf dem Mainframe einloggen können, müssen Sie ein solches Terminal emulieren. Diese Aufgabe übernimmt ein entsprechender TN3270-Emulator. Für Windows existiert das kostenlose Programm wc3270, welches auf der Seite http://x3270.bgp.nu/ zu finden ist. Für Linux kann der x3270-Emulator verwendet werden, der auf selben Seite veröffentlicht ist. Nutzer von MacOs können auf das Programm tn3270 zurückgreifen. Dieses ist auf der Seite http://www.brown.edu/cis/tn3270/index.html der Brown Universität erhältlich.

1.2 Einrichten der Entwicklungsumgebung

Im Folgenden gehen wir davon aus, dass Sie bereits alle oben aufgeführten Hilfsmittel installiert haben und mit ihrer grundlegenden Bedienung vertraut sind. Dieser Abschnitt beschränkt sich daher drauf, Ihnen alle notwendigen weiterführenden Programm- und Projekteinstellungen zu erklären, die für eine funktionsfähige Entwicklungsumgebung benötigt werden. IBM stellt für die Entwicklung von Java-Anwendungen unter CICS eine eigene Bibliothek bereit. Diese enthält die gesamte JCICS-API und ist Teil der Java-Installation des Mainframes. Damit Sie auf Ihrem eigenen System Java-Anwendungen für CICS entwickeln und kompilieren können, müssen Sie zuvor die entsprechende Bibliothek vom Mainframe kopieren und später in alle Java-Projekte einbinden.

Bevor wir die benötige Java-Bibliothek vom Mainframe laden können, müssen wir ein FTP-Client so einrichten, dass wir eine Verbindung zum Mainframe aufbauen können. Auf diesem Wege werden später auch die kompilierten Java-Klassen auf den Mainframe übertragen.

Einrichtung des FTP-Clients und Download der JCICS-Bibliothek

Wir erstellen hierfür eine neue FTP-Verbindung. Das Vorgehen hierfür ist abhängig von dem verwendeten FTP-Programm und daher im entsprechenden Handbuch nachzulesen. Die neue Verbindung sollte folgende Eigenschaften haben:

Protokoll SFTP
Port 22
Server-Adresse 134.2.205.54
Benutzername [Ihr Benutzername]

Im Anschluss können Sie die neu erzeugte Verbindung aufbauen. Diese öffnet Ihr Homeverzeichnis auf dem Mainframe (“/u/prak500”). Die benötigte Bibliothek finden Sie im Verzeichnis “/usr/lpp/cicsts/cicsts31/lib”. Speichern Sie aus diesem Ordner die Datei “dfjcics.jar” auf Ihrem System ab.

Einrichten der Eclipse-Umgebung.

Öffnen Sie zu Beginn eine Workspace Ihrer Wahl. An dieser Stelle empfehlen wir Ihnen, der Übersicht halber eine neue Workspace speziell für diese Reihe von Tutorials anzulegen und in diese Workspace für jedes Tutorial ein eigenes Projekt zu erzeugen. Beim Erzeugen des neuen Projekts, müssen Sie folgende Projekteinstellungen beachten:

- **Als JRE muss die Java JVM 1.5 ausgewählt werden.**
- **Als externe Bibliothek (“External JAR”) muss die zuvor gespeicherte Datei “dfjcics.jar” eingebunden werden.**

Nachdem Sie das neue Java-Projekt erstellt haben, müssen Sie noch sicherstellen, dass die Funktion “Build Automatically” in der Menüleiste unter “Project” aktiviert ist. Diese Einstellung bewirkt, dass nach jeder gespeicherten Änderung automatisch die entsprechende class-Datei erzeugt wird, die wir später auf dem Mainframe laden.

Einrichten der TN3270-Verbindung

Richten Sie innerhalb Ihres TN3270-Emulators eine neue Verbindung mit untenstehenden Eigenschaften ein. Diesen Schritt sollten Sie auf Grund Ihrer Erfahrungen aus anderen CICS-Tutorials bereits selbstständig bewerkstelligen können.

Host-Adresse 134.2.205.54
Portnummer 23
Security SSL
Connection-Type TCP/IP

1.3. Ein- und Ausloggen unter CICS

Die hier vorgestellten Tutorials verwenden CICIS 3.1. Im Folgenden wird beschrieben, wie Sie sich auf dem in Abschnitt 1.6.2 beschriebenen Mainframe unter CICS 3.1 einloggen und ausloggen können.

Starten Sie zu Beginn Ihren TN3270-Emulator und öffnen die in Abschnitt 1.2 beschriebene Verbindung. Sobald die Verbindung erfolgreich aufgebaut wurde, sollte Ihr Emulator den Begrüßungsbildschirm des Mainframes anzeigen. Dieser ist in Abbildung 1 dargestellt.

Einloggen in CICS 3.1 können Sie sich mit der Eingabe:

L CICS1

Nach der Bestätigung der Eingabe erscheint der Begrüßungsbildschirm von CICS, der in Abbildung 1 dargestellt ist. Diesen können Sie mit dem Clear-Befehl (beim wc3270 ist dies Alt+C) Ihres Emulators löschen, um Ihre eigentliche Arbeit unter CICS zu beginnen. Wie Ihnen aufgefallen sein sollte, mussten Sie während des Anmeldeprozesses kein Passwort eingeben. Das CICS 3.1-Subsystem des Mainframes der Universität Tübingen ist so konfiguriert, dass nur für die Verwendung bestimmter Befehle eine Anmeldung mit Benutzername und Passwort notwendig ist. Diese erfolgt über den Befehl:

CESN

Der Login-Befehl wird wie jede andere CICS-Anweisung auch, in die obere, linke Ecke eingetragen. Nach der Eingabe erscheint der Anmeldebildschirm mit der Eingabemaske für Ihren Benutzernamen und Ihr Passwort. Nach der erfolgreichen Anmeldung erscheint ein Bildschirm mit der Nachricht “Sign-on is complete” in der unteren linken Ecke.

Wenn Sie sich in CICS eingeloggt haben, ist es besonders wichtig, dass Sie sich vor Beendigung der Verbindung wieder ordnungsgemäß von CICS abmelden. Sollten Sie dies nicht tun, wird Ihr Account für mehrere Stunden automatisch gesperrt. Abmelden können Sie sich mit dem Befehl:

CESF LOGOFF

Dieser Befehl schließt auch automatisch Ihre Verbindung und Ihr Emulatorfenster.

Abbildung 1: CICS Begrüßungsbildschirm

2. Einführung in JCICS

In diesem Abschnitt werden Sie Ihr erstes JCICS-Programm entwickeln und auf dem Mainframe installieren. Es dient als Grundlage für alle weiterführenden Kapitel, in denen Ihnen verschiedene Techniken der Softwareintegration mit Java und XML unter CICS gezeigt werden. Am Ende dieses Kapitels werden Sie in der Lage sein, unter Anleitung JCICS-Programme auf Ihrem System zu entwickeln und diese anschließend selbstständig auf einem Mainframe unter CICS zu installieren.

2.1. Hello JCICS

In diesem Abschnitt werden Sie eine JCICS-Variante des klassischen HelloWorld-Programms erstellen. Das hier erstellte Programm dient im Wesentlichen dazu Ihnen zu zeigen, wie JCICS-Anwendungen auf dem Mainframe portiert und installiert werden.

Erstellen Sie zu Beginn in Ihrer Entwicklungsumgebung ein neues Java-Projekt “Tutorial1”. Im Anschluss legen Sie innerhalb des Projekts eine neue Java-Klasse “HelloWorld” an. Als Paketnamen sollten Sie Ihren Benutzernamen, gefolgt von “tutorial1”, verwenden. In diesem Beispiel entspricht dieser “prak500.tutorial1”.

Das Listing 1 zeigt Ihnen den gesamten Code der HelloWorld-Klasse. In den folgenden Absätzen werden wir diesen detailliert besprechen und die Unterschiede zu einem HelloWorld-Programm hervorheben, das Sie auf Ihrem eigenen System auf einer Standard-JVM ausführen würden.

Direkt zu Beginn des Programms fällt Ihnen unter (1) auf, dass das Argument der main-Methode nicht wie gewohnt eine Stringarray (String[]) ist, sondern ein Objekt vom Typ CommAreaHolder. Die Commarea, welche durch dieses Objekt gekapselt wird, ist ein speziell reservierter Speicherbereich. Dieser kann von der CICS-Laufzeitumgebung unter anderem für die Ein- und Ausgabe verwendet werden.

In Schritt (2) erzeugen und speichern wir eine Referenz auf das Task-Singelton. Dieses Objekt dient auch der Kapselung der IBM JCICS API und ermöglicht es auf Informationen der Laufzeitumgebung zuzugreifen. Die Implementierung als Singelton ist von der JCICS API vorgegeben und hat den Hintergrund, dass jedes Programm nur über eine einzige Laufzeit verfügt.

Unter CICS ist der Stream System.out nicht mit der Konsole oder einem Terminal verbunden, wie es bei einem heimischen System der Fall wäre. Die Ausgabe in das CICS-Terminalfenster wird über eine Instanz des PrintWriter-Objekts ermöglicht. Diese stellt einen Ersatz für den System.out-Stream dar und kann orthogonal zu diesem verwendet werden. Die PrintWriter-Instanz wird über eine Factory-Methode des Task-Objekts in Schritt (3) bereitgestellt.

Ein weiteres Beispiel für die vom Task-Objekt bereitgestellten Informationen sehen Sie unter Punkt (4). Dort lesen wir aus dem Task-Objekt die Namen des Benutzers und des aktuell ausgeführten Programms mit der dazugehörenden Transaktion aus. Die korrekte Ausführung der in Schritt (4) getätigten Abfrage kann jedoch nicht immer garantiert werden. Aus diesem Grund ist es notwendig, diese Statements in einem try-catch-Block einzufassen, wie Sie ihn bei Punkt (5) sehen können. Dieser soll eventuell auftretende Exceptions abfangen.

2.2. Portierung der Java-Klassen auf den Mainframe

In diesem Abschnitt werden Sie lernen, wie Sie die geschriebenen Java-Programme auf dem Mainframe portieren können. Auf Ihrem System finden Sie nach der Kompilierung für jede Java-Klasse zwei verschiedene Dateien. Eine Datei hat die Dateierweiterung .java und enthält den Quelltext der Klasse. Die zweite Datei trägt die Endung .class und enthält den kompilierten Bytecode, der später von der JVM ausgeführt wird. In welchem Ordner die class-Dateien gespeichert werden, hängt von der verwendeten Entwicklungsumgebung ab. Eclipse und Netbeans zum Beispiel erstellen hierfür einen separaten Ausgabeordner innerhalb der angegebenen Workspace.

Öffnen Sie zunächst den Ordner der verwendeten Workspace in einer Dateiverwaltung Ihrer Wahl (Explorer, Finder...). Dieser enthält für jedes angelegte Projekt einen Unterordner entsprechend des Projektnamens. Eclipse legt innerhalb des Projektordners zwei weitere Ordner an. Einer heißt ”src“ und einer ”bin“. Der bin-Ordner enthält die kompilierten Bytecodevarianten Ihrer Java-Klassen.

Für die spätere Installation der JCICS-Anwendungen müssen Sie diese auf den Mainframe laden. Die kompilierten Klassendateien lassen sich am einfachsten mit Hilfe eines FTP-Klients auf den Mainframe übertragen. Verbinden Sie sich daher über einem FTP-Klienten Ihrer Wahl mit dem Mainframe und navigieren Sie in den Ordner /u/jcicsadm/classes.

Der Ordner /u/jcicsadm/classes wurde als CLASSPATH in das JVM-Profil des Mainframes eingetragen. Sie verfügen mit Ihrem Account nicht über die erforderlichen Rechte, um Ihren eigenen Homeordner als CLASSPATH einzutragen. In den Ordner /u/jcicsadm/classes können Sie nun die gesamte Struktur des bin-Ordners laden. Diese sollte aus zwei Ordnern und einer class-Datei bestehen. Die Ordner wurden beim Kompilieren durch den Java-Compiler automatisch angelegt. Ihre Namen entsprechen den gewählten Paketnamen.

In diesem Beispiel wurde als Paketname “prak500.tutorial1” gewählt. Für diesen Paketpfad wird zuerst ein Ordner “prak500” angelegt, der selbst wiederum einen Ordner “tutorial1” enthält. In diesem befindet sich dann die class-Datei des HelloWorld-Beispiels.

Sollte Ihre Entwicklungsumgebung diese Ordnerstruktur nicht automatisch angelegt haben, ist es notwendig dies auf dem Mainframe per Hand zu tun. Auf jeden Fall sollte die von Ihnen hochgeladene class-Datei am Ende in dem Ordner /u/jcicsadm/classes/prak500/tutorial1 liegen, wobei Sie das “prak500” durch Ihren Benutzernamen ersetzen müssen.

Diese Struktur erlaubt eine saubere Trennung aller Bearbeitungen dieses Kurses. Ihre penible Einhaltung ist aus diesem Grund zwingend erforderlich.

Listing1: HelloWorld in JCICS

```
1  public class HelloWorld
2  {
3    //## (1)
4    public static void main ( CommAreaHolder cah )
5    {
6      //## (2)
7      Task task = Task.getTask();
8      //## (3)
9      PrintWriter out = task.out;
10
11     try {
12       //## (4)
13       out.println("Hello␣" + task.getUSERID() + ",␣welcome␣to␣CICS!");
14       out.println();
15       out.println("This␣is␣program␣" + task.getProgramName());
16       out.println("Transaction␣name␣is␣" + task.getTransactionName());
17     }
18     //## (5)
19     catch (InvalidRequestException e) {
20       e.printStackTrace();
21     }
22   }
23 }
```

2.3. CICS Ressource Definition

CICS verarbeitet Anwendungen mit Hilfe verschiedener Ressource-Definitionen. Für die Installation eines JCICS-Programms als Transaktionen benötigen Sie die Ressourcen:

PROGRAM
TRANSACTION

Die Ressource PROGRAM beschreibt ein unter CICS ausführbares Programm und verweist dafür auf die von Ihnen erstellte Java-Klasse. Eine Transaktion wird über die Ressource TRANSACTION erstellt. Sie kann direkt vom CICS-Transaktionsserver ausgeführt werden und arbeitet selbst wiederum mit der PROGRAM-Definition zusammen.

In diesem Abschnitt lernen Sie, wie Sie beide Ressourcen erstellen, installieren und CICS-Transaktionen ausführen. Für die Bearbeitung der folgenden Schritte ist es notwendig, dass Sie sich unter CICS eingeloggt haben.

52.3.1. Das CICS-Programm

Für die Definition, Bearbeitung, Entfernung und Installation der CICS-Ressourcen ist die eingebaute CICS-Transaktion CEDA zuständig. Für die Definition eines neuen Programms hat der Befehl folgende Syntax:

CEDA DEFINE PROGRAM(Programmname) GROUP(Gruppenname)

Den Programmnamen können Sie frei wählen. Seine maximale Länge beträgt acht Zeichen. Als Gruppennamen sollten Sie Ihren Benutzernamen angeben. In diesem Beispiel wäre dies prak500. CICS unterscheidet bei der Eingabe über das Terminal standardmäßig nicht zwischen Groß- und Kleinschreibung. Die gesamte Eingabe wird stattdessen in Großbuchstaben konvertiert. Für die Definition des HelloWorld-Programms sieht der Befehl wie folgt aus:

CEDA DEFINE PROGRAM(JHELLOW) GROUP(PRAK500)

Nachdem Sie den Befehl durch das Drücken der Entertaste bestätigt haben, öffnet sich eine Liste mit Einstellungsmöglichkeiten der Programmdefinition. Einen Ausschnitt dieser Liste sehen Sie in
Abbildung 1.Wahrscheinlich wird die Liste den Anzeigebereich Ihres Terminals überschreiten. Zum Scrollen können Sie die Tasten F7 und F8 verwenden.

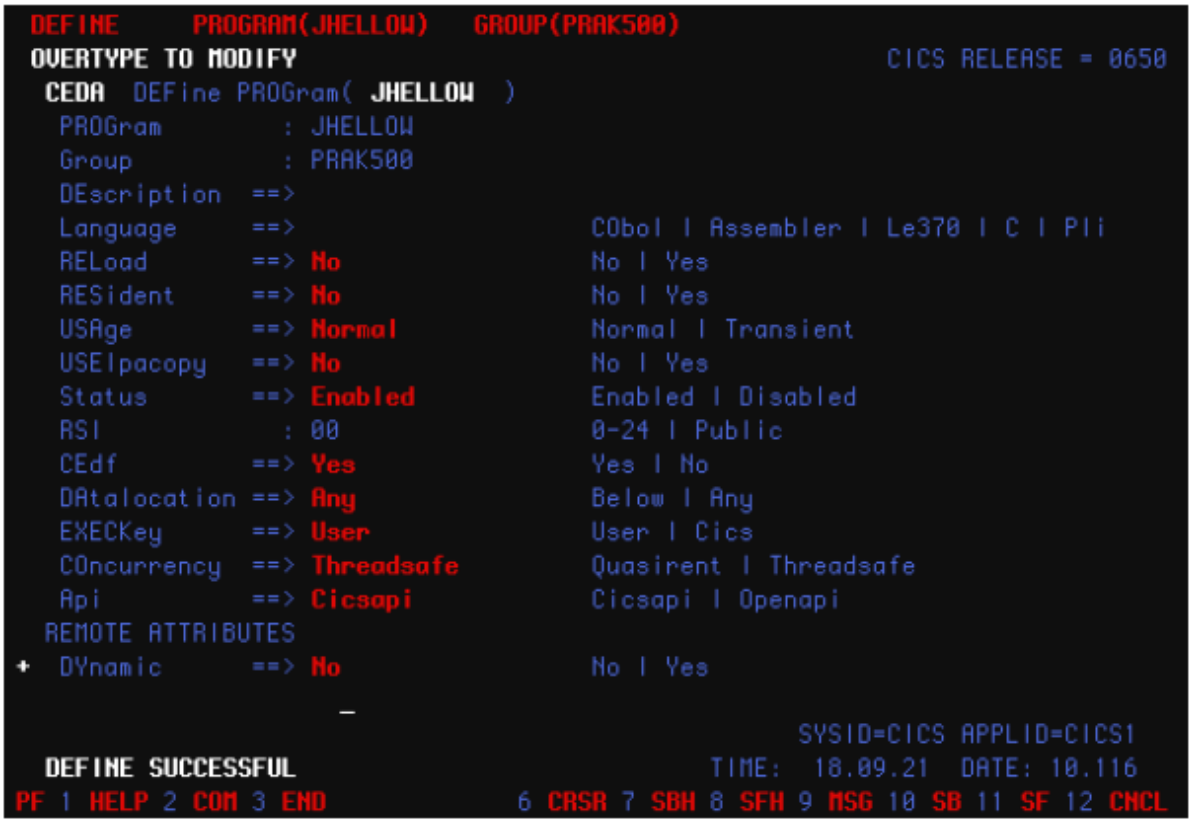


Abbildung 2: Einstellungsmöglichkeiten bei der Definition eines CICS-Programms

Die meisten Einstellungen der Definition müssen nicht geändert werden. Folgende Angaben sind jedoch abweichend von ihren Standardwerten (in Klammern) zu definieren:

Datalocation:	**Any (Below)**
Concurrency:	**Threadsafe (Quasirentrant)**
JVM:	**Yes (No)**
JVMClass:	**Pfad zur Klasse**

In das Feld JVMCLASS müssen Sie den Pfad zu der Java-Klasse angeben, die Sie für Ihre CICS-Anwendung verwenden wollen. In Projekten, die aus mehreren Klassen bestehen, reicht hier die Angabe der Klasse, welche die main-Methode enthält.

Das Feld selbst wird aktiviert, nachdem Sie die Einstellung JVM = Yes gesetzt haben. Im Unterschied zu den anderen Einstellungen wird im Feld JVMCLASS zwischen Groß- und Kleinschreibung unterschieden. Ausgangspunkt des anzugebenden Pfades ist immer der Ordner /u/jcicsadm/classes/. Dies bedeutet, als Pfad zu der HelloWorld-Klasse können Sie Benutzername.tutorial1.HelloWorld eintragen. In diesem Beispiel entspricht dies der Angabe prak500.tutorial1.HelloWorld. Die Angabe der Dateiendung .class ist nicht notwendig. Nachdem Sie alle Einstellungen getätigt haben, können Sie die Definition durch die Betätigung der Eingabetaste speichern. Im Anschluss sollte die Meldung DEFINE SUCCSESSFUL in der unteren, linken Ecke des Terminals erscheinen.

Beenden Sie die CEDA-Umgebung durch Drücken der F3-Taste. Diese agiert ähnlich zu der ESC-Taste unter Windows. Sie erlaubt es Ihnen immer zu der nächst höheren Anwendungsebene zu springen. Auf der höchsten Ebene sehen Sie nur noch die Meldung SESSION ENDED.

Nach der Definition müssen Sie das Programm noch installieren. Dazu geben Sie folgenden Befehl ein:

CEDA INSTALL PROGRAM(JHELLOW) GROUP(Gruppenname)

Im Anschluss sollte die Meldung INSTALL SUCCESSFUL erscheinen. Die Angabe Gruppenname müssen Sie natürlich wieder durch Ihren Benutzernamen ersetzen. In künftigen Installationen müssen Sie JHELLOW durch den Namen des zu installierenden Programms ersetzen.

2.3.2. Die CICS-Transaktion

Theoretisch ist es schon möglich, das oben installierte Programm manuell zu starten. Unser Ziel ist es jedoch, dieses in einer Transaktion einzubinden. Für die Definition einer Transaktion verwenden wir den CEDA-Befehl mit folgender Syntax:

CEDA DEFINE TRANSACTION(Transaktionsname) GROUP(Gruppenname)

Der Name einer Transaktion besteht genau aus vier Zeichen. Auf Grund Ihrer Benutzerrechte ist es Ihnen nur erlaubt Transaktionen auszuführen, die mit einem X beginnen. Wir wählen daher den Namen XJHW. Der Befehl zur Definition sieht daher wie folgt aus:

CEDA DEFINE TRANSACTION(XJHW) GROUP(PRAK500)

Wie bei der Definition eines Programms auch erscheint nach dem Absenden der Anweisung durch Drücken der Eingabetaste eine Liste mit Einstellungsmöglichkeiten. In dieser sind folgende Werte abweichend von ihrer Standardbelegung zu definieren:

Program: Programmname
Taskdataloc: Any

Der hier angegebene Programmname muss mit dem aus Abschnitt 2.3.1 übereinstimmen. Die Definition kann durch die Betätigung der Eingabetaste gespeichert werden. Es sollte wieder die Meldung DEFINITION SUCCESFULL in der unteren linken Ecke des Fensters erscheinen.

Verlassen Sie die Anwendungsebene durch Drücken der F3-Taste. Nach der Definition folgt wieder die Installation. Diese erfolgt durch:

CEDA INSTALL TRANSACTION (XJHW) GROUP(Gruppenname)

Den Gruppennamen müssen Sie wieder durch ihren Benutzernamen ersetzen. In späteren Installationen muss die Angabe XJHW durch den Namen der zu installierenden Transaktion ersetzt werden.

Zum Starten der Transaktion verlassen Sie ggf. alle zurzeit verwendeten Transaktionen. Löschen Sie den aktuellen Bildschirm durch den entsprechenden Befehl Ihres TN3270-Emulators. Geben Sie dann den Namen der Transaktion ein und drücken Sie die Eingabetaste. In diesem Beispiel lautet der Befehl XJHW. Die entsprechende Ausgabe ist in Abbildung 3 dargestellt.

Abbildung 3: Ausgabe der Transaktion

2.4 Zusammenfassung

JCICS-Programme bedienen sich einer anderen Systemschnittstelle als herkömmliche Java-Programme. Zum Beispiel kann die Eingabe einer main-Methode neben einer String-Array auch einen Commareaholder entgegennehmen. Die Ausgabe auf das Terminalfenster erfolgt nicht über den gewöhnlichen System.out-Befehl, sondern über einen PrintWriter. Dieser wird vom Task-Objekt zur Verfügung gestellt, das zusätzlich auch weitere Informationen der Laufzeitumgebung bereitstellt. Nach der lokalen Entwicklung eines Java-Programms müssen die Java-Class-Dateien auf den Mainframe geladen werden. Dies erfolgt am einfachsten mit Hilfe eines FTP-Klienten. Legen Sie die class-Dateien in die entsprechende Unterorder des Java-Classpath.

Für die Ausführung eines Java-Programms unter CICS ist es notwendig, ein CICS-Programm und eine CICS-Transaktion zu erstellen. Beides sind CICS-Ressourcen. Diese müssen zuerst definiert und im Anschluss installiert werden.

Aufgaben

1. Entwickeln Sie ein eigenes Hello-CICS-Programm auf Basis des in vorgestellten Codes. Erweitern Sie die Ausgabe um die Namen Ihrer Teammitglieder.

2. Informieren Sie sich im CICS 3.2 Informationscenter (http://publib.boulder.ibm.com/infocenter/cicsts/v2r3/index.jsp) über weitere Umgebungsvariablen die durch das TASK-Objekt bereitgestellt werden. Nennen Sie mindestens eine weitere und erklären Sie deren Bedeutung.

3. Informieren Sie sich im CICS Transaction Handbuch über weitere Klassen, die der Kapselung der Systemschnittstelle dienen. Nennen Sie eine und erörtern kurz deren Aufgabe.

Die Datei jcicsres01.zip enthält alle Java Resources als zip Archiv. Sie kann heruntergeladen werden unter

www.cedix.de/Vorles/Band3/Resources/jcicsres01.zip

JCICS Tutorial Teil 2
Externe Programmaufrufe und Datenaustausch unter JCICS

Dank an Herrn Stefan Huster für die Bereitstellung des Tutorials

Übersicht

1. Hintergrund

Die Hauptaufgabe der Softwareintegration ist es, die Kommunikation zwischen verschiedenen Systemen und Anwendungen zu ermöglichen. In diesem Tutorial werden Sie die CICS-internen Methoden kennenlernen, mit denen Sie eine solche Kommunikation aufbauen können. Im ersten Abschnitt werden Ihnen Befehle vorgestellt, mit denen Sie andere unter CICS installierte Programme aufrufen können. Im Anschluss erhalten Sie einen Einblick in Techniken für den Datenaustausch zwischen verschiedenen CICS-Programmen. Am Ende dieses Kapitels wird Ihnen noch der praktische Einsatz der vorgestellten Methoden und Verfahren in einem kurzen JCICS-Beispiel demonstriert.

Programme unter CICS können mit einer Vielzahl verschiedener Sprachen entwickelt werden, dazu zählen neben Java unter anderem auch Cobol, C++, oder PL/1. Die Systemschnittstelle von CICS ermöglicht es auf eine sehr einfache Weise, Kommunikationswege zwischen allen unter CICS installierten Programmen aufzubauen, unabhängig von der Sprache, mit der sie implementiert wurden. Für den Anwendungsentwickler stellt diese eine große Vereinfachung dar, da er ansonsten auf Verfahren wie zum Beispiel dem Remote Procedure Call (RPC), der Java Methode Invocation (JMI) oder der Common Object Request Broker Architecture (CORBA) zurückgreifen müsste. Das Problem dieser Techniken ist, dass sie jeweils nur von einer Teilmenge der verfügbaren Sprachen unterstützt werden.

2. Externe Programmaufrufe

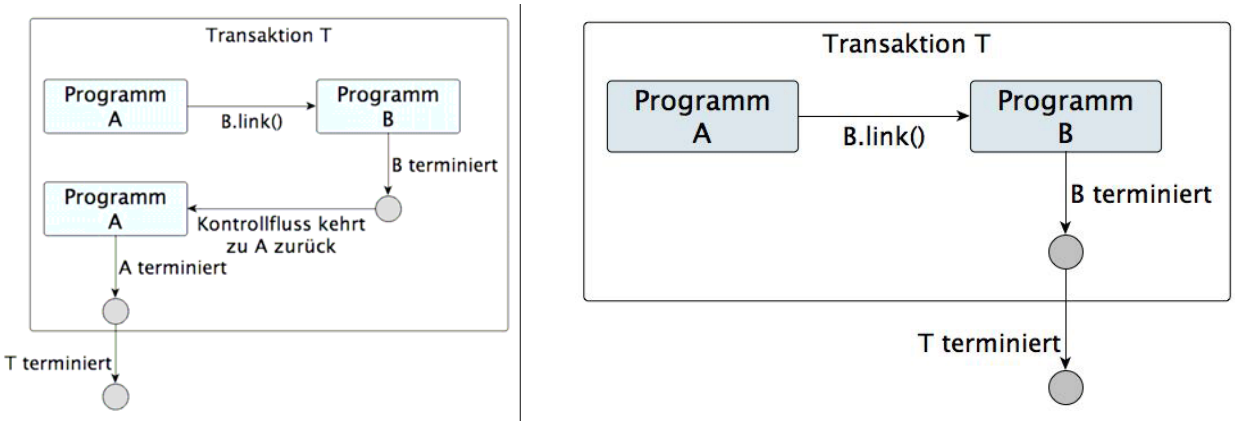
In diesem Abschnitt lernen Sie, wie Sie aus Ihrer CICS-Anwendung andere CICS-Programme aufrufen. Die Systemschnittstelle von CICS stellt Ihnen dafür zwei Befehle zur Verfügung: LINK und XCTL. Unter JCICS sind beide Befehle als Methoden der Klasse com.ibm.cics.server. Program aus der externen Bibliothek dfjcics.jar realisiert. Nach ihrer Ausführung werden die link- und die xctl-Anweisungen direkt unter CICS verarbeitet. Mit ihnen kann daher jedes CICS-Programm aufgerufen werden, unabhängig davon, in welcher Sprache es geschrieben wurde. An dieser Stelle werden Ihnen die konzeptionellen Unterschiede beider Methoden vorgestellt. Die genaue Syntax und die Verwendung unter JCICS werden Sie innerhalb des Abschnitts 3 kennenlernen.

2.1. LINK

Betrachten wir ein Szenario, bestehend aus einer Transaktion T, einem Programm A und einem Programm B. Innerhalb dieser Transaktion muss Programm A ein anderes Programm B aufrufen. Mit der abstrakten Anweisung B.link() kann das Programm A die Laufzeitumgebung auffordern, die momentane Programmausführung in der main-Methode von Programm B fortzusetzen. Terminiert Programm B, springt der Kontrollfluss zurück zu Programm A und die Programmausführung wird hinter dem Aufruf von B.link() in A fortgesetzt. Die gesamte Transaktion T terminiert demnach erst, wenn auch Programm A terminiert. Das Verfahren der externen Programmaufrufe kann natürlich transitiv fortgesetzt werden, indem auch Programm B weitere Programme aufruft.

2.2. XCTL

Die Aufgaben des xctl-Befehls sind identisch zu denen des link-Befehls. Auch die Syntax beider Anweisungen sind orthogonal zu verwenden. Der Unterschied beider Methoden liegt im Programmablauf. Betrachten wir hierfür noch einmal das Szenario aus Abschnitt 1.1. In diesem Beispiel hat Programm A ein anderes Programm B aufgerufen. Gehen wir an dieser Stelle davon aus, dass der Aufruf nicht via B.link(), sondern mit B.xctl() erfolgt ist. In diesem Fall kehrt der Kontrollfluss nach der Terminierung von B nicht mehr zu A zurück. Stattdessen terminiert die gesamte Transaktion T. Dies bedeutet, dass der xctl-Befehl die gesamte Ablaufkontrolle an das aufgerufene Programm überträgt. Den Unterschied zwischen LINK und XCTL finden Sie noch einmal in der Abbildung 1 illustriert.



(a) Der Kontrollfluss einer link-Anweisung

(b) Der Kontrollfluss einer xctl-Anweisung

Abbildung 1: Ablauf des Kontrollflusses bei LINK- und XCTL-Anweisungen

Selbst-Test

Können LINK und XCTL eingesetzt werden

- Nur zwischen 2 CICS Programmmen, die beide in Cobol geschrieben sind ?
- Nur zwischen 2 CICS Programmmen, die beide in Java geschrieben sind ?
- Auch zwischen 2 CICS Programmmen, die beide in untrschiedlichen Sprachen geschrieben sind ?

3. Datenaustausch

In diesem Abschnitt lernen Sie, wie Sie Daten zwischen verschiedenen CICS-Programmen austauschen können. Dieser Prozess spiegelt die eigentliche Kommunikation zwischen verschiedenen Softwarelösungen wieder. Diese herzustellen ist ein Ziel der Softwareintegration. Wie im vorangegangenen Abschnitt können die hier beschriebenen Verfahren dazu verwendet werden, Daten zwischen beliebigen CICS-Programmen auszutauschen, unabhängig davon, in welcher Sprache sie geschrieben wurden.

3.1. COMMAREA

Die COMMAREA ist ein speziell reservierter Speicherbereich. Jedes Programm, das z.B. über eine link-Anweisung aufgerufen wird, erhält eine Referenz auf eine übergebene COMMAREA. Auf diese Weise kann der speziell reservierte Speicher zum Austausch von Daten verwendet werden. Sowohl das aufrufende als auch das aufgerufene Programm verfügen über eine Referenz auf denselben Speicherbereich. Aus diesem Grund kann eine COMMAREA zur Ein- und zur Ausgabe verwendet werden. Die Größe des reservierten Speicherbereichs wird durch das aufrufende Programm festgelegt und kann später nicht mehr geändert werden. Ihre maximale Größe beträgt 32kB [1]. Die Übergabe der Daten erfolgt auf Byte-Ebene. Das heißt, auch unter JCICS werden Byte-Arrays ausgetauscht. Es liegt beim Anwendungsentwickler, diese in das richtige Format zu konvertieren und entsprechende Fehlerquellen zu berücksichtigen. Dieses Vorgehen ist sehr systemnah und auf einem sehr niedrigen Abstraktionslevel. Dennoch bildet diese Methode die Basis für jeden Kommunikationspfad innerhalb von CICS.

3.2. Channels und Container

Eine andere und auch etwas modernere Variante des Datenaustauschs zwischen verschiedenen CICS Programmen ist die Verwendung von „Channels und Container“. Ein Container dient der Kapselung der zu sendenden Daten. Man könnte ihn mit einem Umschlag vergleichen, der den Inhalt eines Briefes übermittelt, vergleichbar mit einem SOAP Envelope. Technisch gesehen ist ein Container nicht viel mehr als eine benannte COMMAREA. Ein Channel ist eine Verbindung zwischen zwei verschiedenen CICS-Programmen, in dem Container transportiert werden können. Der Kanal entspricht daher dem Briefträger, der einen Umschlag vom Sender zum Empfänger bringt. In Abschnitt 3 werden Sie die Verwendung von Container und Channels im praktischen Einsatz sehen.

Ein Channel bietet gegenüber der Verwendung der COMMAREA folgende Vorteile (Auszug aus [2]):

- **Channels sind nicht auf eine maximale Größe von 32kB beschränkt. Es besteht keine Limitierung bzgl. der Anzahl und Größe der übertragenen Container. Die einzige Beschränkung ist durch den physisch vorhandenen Speicher des Mainframes gegeben.**
- **Ein Channel kann Container verschiedener Typen enthalten. Dies ermöglicht, Daten strukturierter zu übertragen, vergleicht man das Vorgehen mit der COMMAREA, die nur einen einzigen Typ unterstützt.**

Channels sind ähnlch wie COMMAREA standardisiert und werden von allen unter CICS-lauffähigen Programmiersprachen unterstützt.

Auf der anderen Seite gilt es auch einige Punkte bei der Verwendung von Channels zu beachten:

- **Verwendet man Channels zur Datenübertragung zwischen CICS-Anwendungen, die auf unterschiedlichen Mainframes betrieben werden, kann der Transport größerer Datenmengen zu erheblichen Performanceverlusten führen.**
- **Für die Übertragung von Daten mit Hilfe von Channels werden mehr Systemressourcen benötigt als bei einer Übertragung via COMMAREA. Dies liegt daran, dass Container-Objekte in unterschiedlichen Speicherbereichen im System hinterlegt sein können. Dies erfordert unter Umständen häufiger langsame Lesezugriffe. Des Weiteren werden Daten innerhalb der COMMAREA nur als Referenz übergeben. Die Daten in einem Container werden kopiert.**

Um einen Container innerhalb eines Channels zu erzeugen, verwenden CICS-Programme den API-Befehl:

EXEC CICS PUT CONTAINER(Name des Containers) CHANNEL(Name des Channels)

Die Umsetzung innerhalb eines JCICS-Programms wird in Abschnitt 3 erläutert. Die Übertragung der Daten erfolgt mit Hilfe der oben beschriebenen link- oder xctl-Kommandos. Das Empfängerprogramm kann die Container über folgenden Befehl lesen:

EXEC CICS GET CONTAINER(Name des Containers)

Der Name des Channels wird automatisch impliziert, da jedes Programm zum Zeitpunkt des Aufrufs genau einen Channel zugewiesen bekommt.

Ein weiterer großer Vorteil von Channels ist die damit verbundene Erweiterungsmöglichkeit von Programmen. In Abbildung 2 sehen Sie ein Programm illustriert, welches über zwei verschiedene Channels aufgerufen werden kann. In der IBM-Literatur wird dieser Zustand als „loose coupling“ [3] bezeichnet. Damit ist gemeint, dass das Interface zum aufrufenden Klienten nicht eindeutig und fest definiert sein muss, sondern erst zur Laufzeit über die Auswahl des Channels bestimmt werden kann. Der Klient wählt einen entsprechenden Kanal des Hauptprogramms aus. Dieses interpretiert die Daten je nach Kanal anders und leitet sie unter Umständen an das entsprechende Unterprogramm weiter. In Abschnitt 3 werden Sie dieses Vorgehen in einem kleinen Beispiel im Einsatz sehen.

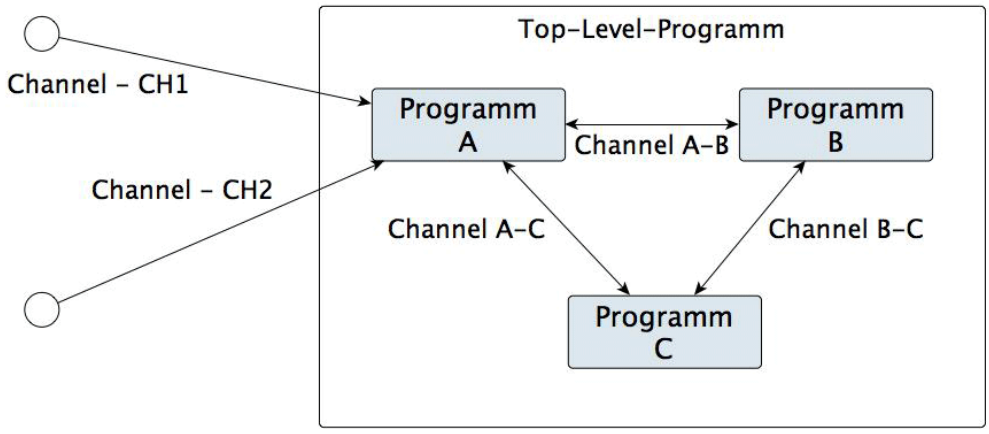


Abbildung 2: Channel Multiplexing

3.3. Channels vs. COMMAREA

Wie Sie gesehen haben, verfügen beide Verfahren über ihre Vor- und Nachteile. Die COMMAREA ist sehr leichtgewichtig und aufgrund des Datenaustauschs auf Basis von Referenzen sehr performant und zudem schnell zu implementieren. Auf der anderen Seite bieten Channels die Möglichkeit, Daten strukturiert zu übergeben und Programme sehr leicht zu erweitern. Dieser Komfort wird dafür mit einem erhöhten Verbrauch von Systemressourcen bezahlt.

Für die Entscheidung der verwendeten Technik stellt IBM in [1] drei einfache Kriterien bereit:

- **Ist es notwendig, größere Datenmengen jenseits der Grenze von 32kB zwischen Programmen zu übertragen, sind Channels und Container die eindeutig bessere Wahl.**
- **Arbeiten Sie mit bestehenden Programmen, die eine COMMAREA zur Kommunikation verwenden und nur geringe Datenmengen austauschen, lohnt sich die Umstellung auf das Channel-Konzept unter Umständen nicht.**
- **Entwickeln Sie neue Projekte, ist die Verwendung des Channel-Konzepts in der Regel die bessere Entscheidung.**

Selbst-Test

Welches der beiden Verfahren Channels vs. COMMAREA

- ist performanter ?
- gestattet die Übertragung beliebig großer Datenmengen ?
- ist einfacher zu programmieren
- hat einen größeren Funktionsumfang
- bewirkt Datenübertragung auf der Basis von Referenzen?

4. Anwendungsbeispiele

In den folgenden zwei Beispielen werden Sie sehen, wie die oben beschriebenen Techniken unter JCICS angewendet werden können.

4.1. LINK, XCTL, COMMAREA

In diesem Beispiel verwenden wir zwei Programme, von denen eins das andere einmal über die link- und einmal über die xctl-Anweisung aufruft. Zusätzlich wird dem aufgerufenen Programm über die COMMAREA eine Frage übermittelt. Das aufgerufene Programm wird dann die richtige Antwort in die COMMAREA schreiben.

Das Beispiel der link-Variante besteht aus zwei Programmen: „DoLink“ und „CallMe“. Deren Quelltexte finden Sie in den Listings 1 und 3. Wir beginnen mit „DoLink“:

Zu Beginn geben wir zur Bestätigung in (1) den Programmnamen (hier „DoLink“) des aufgerufenen Programms aus. Dies zeigt uns, dass der Kontrollfluss zurzeit von der Klasse DoLink gesteuert wird.

Als nächstes erzeugen wir in Schritt (2) eine Nachricht, die wir dem aufrufenden Programm senden möchten. Da die COMMAREA unter JCICS als Byte-Array repräsentiert wird, muss der String vor dem Senden konvertiert werden. Die Größe der COMMAREA ist damit automatisch festgelegt worden. Sie entspricht der Größe der gespeicherten Zeichenkette.

Unter Punkt (3) erzeugen wir eine neue Instanz der Programm-Klasse. Dieser Typ wird von der JCICS API bereitgestellt und dient der Abstraktion über CICS-Programme. Dem erzeugten Programm weisen wir im Anschluss einen Namen zu. Dieser darf nicht willkürlich vergeben werden.

Er muss mit dem Namen des aufzurufenden Programms übereinstimmen, der innerhalb der entsprechenden CICS-Ressource definiert wurde. Die Übereinstimmung des Klassennamens mit dem Programmnamen ist hingegen nicht vorgeschrieben. Mit dem Befehl prog.link(COMMAREA) wird schließlich das zweite Programm des Beispiels aufgerufen. Dessen Analyse folgt im Anschluss an diese Besprechung.

Das von uns aufgerufene Programm wird, wie wir später sehen werden, eine Antwort in die übergebene COMMAREA schreiben. Diese wird in Schritt (4) ausgegeben.

Sowohl der link-Befehl als auch die Abfrage des Programmnamens unter Punkt (1) sind potentielle Fehlerquellen. Es könnte zum Beispiel passieren, dass das aufgerufene Programm nicht existiert. In diesem Fall würde die link-Anweisung eine Exception werfen. Diese werden bei Punkt (5) gefangen. Genauere Informationen über die möglichen Exceptions finden Sie in [2] und [4].

Als Nächstes widmen wir uns der Betrachtung des Quelltextes des aufgerufenen Programms. In diesem Beispiel heißt dieses CALLME. Die entsprechende Java-Klasse trägt den Namen CallMe und ist in Listing 3 abgedruckt.

Wie auch zuvor beginnen wir das Programm mit einem kurzen Lebenszeichen (1). Den Sinn und Zweck der println-Anweisungen besprechen wir später, wenn wir die Ausgabe betrachten und diese mit der des xctl-Beispiels vergleichen.

Die übergebene COMMAREA wird durch den COMMAREAHolder als Parameter der main-Methode gekapselt. Dieser hat genau ein Klassenfeld namens value vom Type byte[]. Diesem Feld weisen wir in Schritt (2) die richtige Antwort auf die übertragene Frage zu.

Nachdem der Quelltext beider beteiligten Klassen besprochen wurde, wird es Zeit, einen Blick auf die Ausgabe der entsprechenden Transaktion zu werfen. Diese sehen Sie in Abbildung 3. Bereits bei der oberflächlichen Betrachtung sind zwei Punkte besonders auffällig:

1. **Es fehlt bei der Ausgabe der Transaktion in Zeile 2, 3 und 4 jeweils der Anfangsbuchstabe.**
2. **Neben der richtigen Antwort 42 [5] ist in Zeile 5 noch der Rest der Anfrage sichtbar.**

Die Erklärung der ersten Beobachtung führt uns direkt zurück zu der Frage, wieso in der Klasse CallMe so viele println-Anweisungen verwendet wurden. Die println-Anweisungen beginnen ihre Ausgabe in der ersten Zeile des Terminalfensters. In dieser Zeile steht i.d.R. jedoch der Name der Transaktion, hier XDOL. Aus diesem Grund begannen wir in der Klasse DoLink auch die Ausgabe mit einer leeren Zeile. Übergibt man den Kontrollfluss via LINK oder XCTL an ein anderes Programm, welches ebenfalls Ausgaben auf das Terminalfenster druckt, beginnt die Ausgabe wieder in der ersten Zeile. Dies liegt daran, dass dem aufgerufenen Programm eine neue Task-Umgebung zugewiesen wird. Aus diesem Grund gibt die Klasse CallMe zuerst drei leere Zeilen aus, da ansonsten die Ausgabe des DOLINK-Programms überschrieben werden würde. Die Ausgabe von println() ist jedoch nicht leer, sondern wird im Terminal als „-„ dargestellt, um jeweils das Ende einer Zeile zu markieren. Das “-„ überschreibt auch den ersten Buchstaben der Zeilen 2-4.

Der Grund für die zweite Beobachtung wurde bereits im Abschnitt 2.1 besprochen. Dort wurde festgestellt, dass die COMMAREA immer eine feste Länge hat, die vom aufrufenden Programm festgelegt wird. In diesem Beispiel entspricht die Größe der COMMAREA genau dem Platz, der durch die Anfrage “the answer to life the universe and everything” benötigt wird. Die Antwort „42" hingegen ist kürzer und füllt daher nicht die gesamte COMMAREA aus. Der nicht überschriebene Speicher bleibt unverändert. Dies führt dazu, dass ein Teil der Anfrage auch noch in der Antwort zu lesen ist. Es ist die Aufgabe des Anwendungsentwicklers, dafür zu sorgen, dass gegebene Bedingungen an den Speicherbereich der COMMAREA erfüllt werden. Eine mögliche Bedingung für dieses Beispiel wäre gewesen, dass die Antwort keinen Teil der Anfrage mehr enthalten darf. In diesem Fall hätte der nicht überschriebene Teil der COMMAREA manuell gelöscht werden müssen.

Bevor wir unsere Aufmerksamkeit auf die Ausgabe der xctl-Variante lenken, betrachten wir noch kurz den Quellcode dieser Alternative. Die entsprechende Klasse heißt DoXctl (Listing 1) und das CICS-Programm DOXCTL. Der Quelltext der Java-Klasse ist an dieser Stelle mehr der Vollständigkeit zuliebe aufgeführt, da die Änderungen verglichen zu der link-Varainte nur minimal ausfallen. Der einzige Unterschied ist unter Punkt (3) zu finden. Dort wird einfach anstelle von prog.link(COMMAREA) prog.xctl(COMMAREA) verwendet.

Die Auswirkung der xctl-Anweisung kann direkt in der Ausgabe in Abbildung 3b verfolgt werden. Vergleicht man diese mit der in Abbildung 3a gezeigter Ausgabe der DOLINK Transaktion, fällt auf, dass DOXCTL keine vierte Zeile ausgibt. Dies liegt daran, dass die entsprechende Transaktion DOXCTL nach der Ausführung von CALLME terminiert und nicht wie zuvor zum aufrufenden Programm zurückkehrt. Das heißt, die Zeilen unter Punkt (4) in der DoXctl-Klasse werden aufgrund der xctl-Anweisung nie erreicht und ausgeführt.

Ein weiterer kleiner Unterschied zu der Ausgabe von DOLINK ist das Vorhandensein der ersten Zeichen in Zeile 2 und 3. Dies hat jedoch nichts mit der xctl-Methode zu tun, sondern ist das Resultat eines vorangestellten Leerzeichens jeder println-Anweisung in DoXctl.

4.2. Channels und Container

Im zweiten Beispiel betrachten wir die Kommunikation zwischen verschiedenen CICS-Programmen unter Verwendung von Channels und Container. Des Weiteren wird Ihnen gezeigt, wie Sie über einen Channel-Multiplexer mit einem Programm verschiedene Klienten bedienen können.

```
XLNK
    his is program:TUTLINK
    y question:the answer to life the universe and
everything
    his is program CALLME
And the answer is: 42e answer to life the universe
and everything
```

(a) Ausgabe der Transaktion XDOL

```
XCTL
    This is program:TUTXCTL
    My question:the answer to life the universe and
everything
This is program CALLME
```

(b) Ausgabe der Transaktion XDOL

Abbildung 3: Ausgabe der externen Programmaufrufe

Diese kleine Anwendung besteht aus drei verschiedenen Klassen: NeedEuro, NeedDollar und MoneyConverter. Die ersten beiden Klassen erfüllen die Rolle der Klienten, während die dritte Klasse die Rolle eines Serviceproviders einnimmt. Die Idee ist, dass die Klienten jeweils entweder einen Euro- oder einen Dollar-Betrag an die Serviceklasse schicken und als Antwort den entsprechenden Betrag in der jeweils anderen Währung erhalten.

Damit der Serviceprovider beide Anfragen unterscheiden kann, verwenden die Klienten Channels mit unterschiedlichen Namen. Natürlich wären an dieser Stelle auch andere Möglichkeiten zur Unterscheidung denkbar. Streng genommen wären diese sogar allein durch unterschiedlich benannte Container möglich, jedoch würde eine solche Optimierung dieses Beispiel zunichtemachen.

Da beide Klassen bis auf ihren Namen und den Namen des verwendeten Channels identisch aufgebaut sind, begnügen wir uns hier mit der Betrachtung nur eines Quellcodes, dem der NeedEuro-Klasse in Listing 4. Den Quelltext der anderen Klasse finden Sie im Anhang.

In Schritt (1) erzeugen wir einen neuen Kanal. Dies erfolgt über eine Factory-Methode des Task- Objekts (vgl. Factory-Pattern [6]) mit dem Namen createChannel. Wir weisen dem Kanal den Namen „DollarToEuro“ zu.

Als nächstes benötigen wir einen Container, den wir über den erzeugten Kanal versenden können.

Auch dieser wird in Schritt (2) über eine Factory-Methode erstellt. Der Container erhält den Namen „DOLLAR". Container sind vom Prinzip her nichts anderes als benannte COMMAREAs. Aus diesem Grund können sie ebenfalls nur Daten als Byte-Array speichern. Dafür erstellen wir in Punkt (3) einen String mit dem zu konvertierenden Betrag als Wert. Im Anschluss wird der String über die put-Methode im Container gespeichert. Die Konvertierung in eine Byte-Array erfolgt durch die Methode automatisch.

Schritt (4) sollte Ihnen aus Abschnitt 3.1 vertraut vorkommen. Wir erzeugen eine Instanz des Program-Objects und rufen über die link-Methode das externe CICS-Programm MONCONV auf.

Dieses Mal übergeben wir der link-Anweisung jedoch keine COMMAREA, sondern den erzeugten Kanal (5).

Der Wert des Containers wird, ähnlich wie im ersten Beispiel, durch das aufgerufene Programm geändert. Der geänderte Wert wird in Schritt (6) mit der get-Methode neu ausgelesen und im Anschluss ausgegeben.

Betrachten wir als Nächstes, wie der Dollarwert durch den Serviceprovider in Euro umgerechnet wird. Den Quelltext der Klasse MoneyConverter finden Sie in Listing 6.

Zu allererst wird der Channel benötigt, mit dem dieses Programm aufgerufen wurde. Diesen erhalten wir von der Laufzeitumgebung (1), über die unter CICS durch das Task-Objekt abstrahiert wird.

Der Channel, den die Task zurückgegeben hat, kann unter Umständen auch null sein, zum Beispiel dann, wenn das Programm mit einer übergebenen COMMAREA aufgerufen wurde. Diese Möglichkeit muss überprüft werden (2), bevor mit jeglicher Verarbeitung des Channels begonnen werden kann. Im Block (3), bestehend aus Block (3-a) und (3-b), findet das Multiplexen des Channels statt. Dafür wird zuerst der Name des übergebenen Channels erfragt und anschließend mit allen unterstützten Namen verglichen. Diese Serviceklasse unterstützt zwei verschiedene Channels: „EuroToDollar“ und „DollarToEuro“. Die Verarbeitung des Channels erfolgt im Anschluss innerhalb von Block (3-a) bzw. (3-b). Dort wird der erwartete Container mit ch.getContainer() aus dem Channel extrahiert und dessen Wert in das gewünschte Format konvertiert. In diesem Fall erwarten wir einen Betrag, den es in eine andere Währung umzurechnen gilt. Der Wert ist daher numerisch und kann mit Double.valueOf() konvertiert werden.

Für die Umrechnung in die entsprechende Währung sind die Methoden unter (4-a) und (4-b) verantwortlich. Deren Ergebnis wird mit der put-Methode des Containers zurückgeschrieben und kann im Anschluss vom aufrufenden Programm gelesen werden.

Die Ausgabe beider Programme sehen Sie in Abbildung 4.

XEUR Give100 Dollars Get76.92307692307692 Euros	XUSD Give100 Euros Get130.0 Dollars
(a) Ausgabe Konvertierung Euro nach Dollar	**(b) Ausgabe Konvertierung Dollar nach Euro**

Abbildung 4: Ausgabe der beiden Klienten des Währungsrechners

Hinweis

Die Methoden LINK und XCTL können auch mit anderen Parametern aufgerufen werden, als die, die in diesem Beispiel verwendet wurden. Eine detaillierte Liste finden Sie in der IBM JCICS API [7].

Channels können auch mehr als einen Container enthalten. Wie im Abschnitt 2.1 erwähnt wurde, ist deren Anzahl nur durch die Größe des physischen Speichers beschränkt. In Fällen, in denen mehrere Container übergeben werden, kann es sehr sinnvoll sein, über diese als Liste zu iterieren. Für diesen Zweck stellt die JCICS API einen ContainerIterator bereit. Dieser implementiert das bekannte java.util.Iterator Interface. Listing 7 zeigt ein Beispiel, entnommen aus [2], für die Verwendung dieses Iterators.

5. Zusammenfassung

Die CICS API stellt zwei Methoden für den Aufruf externer CICS-Programme bereit: LINK und XCTL. Beide Methoden verfügen über eine ähnliche Schnittstelle. Die link-Methode ruft ein anderes Programm auf und kehrt nach dessen Terminierung wieder zum aufrufenden Programm zurück. Die xctl-Anweisung hingegen beendet die gesamte Transaktion, nachdem das aufgerufene Programm terminiert ist.

Für den Austausch von Daten zwischen CICS-Programmen existieren ebenfalls zwei verschiedene Verfahren: Datenaustausch über die COMMAREA und über Channels.
Die COMMAREA ist ein spezieller, geteilter Speicherbereich zweier CICS-Programme. Die Daten werden als Referenz übergeben und müssen vom Type byte[] sein. Die Größe einer COMMAREA wird einmal festgelegt und kann maximal 32kB betragen.

Ein Channel überträgt Daten, die zuvor mit einem Container gekapselt wurden. Channel und Container können benannt und somit auch unterschieden werden. Auch hier werden Daten als Bytes transportiert. CICS gibt jedoch keine Obergrenze für die Anzahl der Container und deren Größe vor. Sowohl die Methoden zum Aufruf externer Programme als auch die Verfahren zur Datenübertragung funktionieren zwischen allen unter CICS installierten Programmen.

Selbst-Test

- Erklären Sie, warum im Beispiel 4 die link-Methode verwendet wurde und nicht die xctl Anweisung ?

Aufgabe: **Entwickeln Sie einen JCICS-Taschenrechner. Dieser soll die vier Grundrechenarten unterstützen. Ein zweites CICS-Programm soll die verschiedenen Methoden des Taschenrechners aufrufen und das berechnete Ergebnis auf der Konsole ausgeben. Entscheiden Sie selbst, welches Verfahren Sie für den Programmaufruf und für die Datenübertragung verwenden**

Quellcodes

```
1 public class DoLink {
2  public static void main (CommAreaHolder cah) {
3       Task task = Task.getTask ();
4       PrintWriter out = task.out;
6       try {
7        // ## (1)
8        out.println();
9        out.println("This is program :" + task.getProgramName());
11            // ## (2)
12            String myMsg = "the answer to life the universe and
everything";
13            byte[] commArea = myMsg.getBytes();
14            out.println("My question :" + myMsg);
16            // ## (3)
17            Program prog = new Program();
18            prog.setName("CALLME");
19            prog.link(commArea);
21            // ## (4)
22            out.println();
23            String rply = new String(commArea);
24            out.println("And the answer is: " + rply);
26            // ## (5)
27           } catch (Exception e) {
28             e. printStackTrace(out);
29           }
30      }
31 }
```

Listing 1: Aufrufendes Programm mit LINK

```
1 public clas s DoXctl {
2  public static void main(CommAreaHolder cah) {
3       Task task = Task.getTask();
4       PrintWriter out = task.out;

6       try {
7        // ## (1)
8        out.println();
9        out.println(" This is program :" + task.getProgramName ());

11            // ## (2)
12            String myMsg = "the answer to life the universe and
everything";
13            byte[] commArea = myMsg.getBytes();
14            out.println("My question :" + myMsg);

16            // ## (3)
17            Program prog = new Program();
18            prog.setName("CALLME");
19            prog.xctl(commArea);

21            // ## (4)
22            out.println();
23            String rply = new String(commArea);
24            out.println("And the answer is: " + rply);

26            // ## (5)
27           } catch (Exception e) {
28            e.printStackTrace(out);
29           }
30      }
31 }
```

Listing 2: Aufrufendes Programm mit XCTL

```
1 public class CallMe {
2  public static void main (CommAreaHolder cah) {
3       PrintWriter out = Task.getTask().out;
4       out.println();
5       out.println();
6       out.println();
7       // ## (1)
8       out.println("This is program CALLME");
9       // ## (2)
10           String theAnswer = "42";
11           cah.value = theAnswer.getBytes();
12      }
13 }
```

Listing 3: Aufrufendes Programm mit XCTL

```
1 public class NeedEuro {
2  public static void main (CommAreaHolder cah) {
3       try {
4        // ## (1)
5        Channel ch = Task.getTask().createChannel("DollarToEuro");
6        // ## (2)
7        Container dollarCon = ch.createContainer("DOLLAR");
8        // ## (3)
9        String dollarStr = "100";
10            dollarCon.put(dollarStr);
11            // ## (4)
12            Program moneyConverter = new Program ();
13            moneyConverter.setName("MONCONV");
14            // ## (5)
15            moneyConverter.link(ch);
17            PrintWriter out = Task.getTask().out;
18            out.println();
19            out.println("Give " + dollarStr + " Dollars");
20            // ## (6)
21            String euroStr = new String(dollarCon.get());
22            out.println("Get " + euroStr + " Euros");
24           } catch ( Exception e) {
25            e. printStackTrace(out);
26           }
27      }
28 }
```

Listing 4: Anfrage mit Channel

```
1 public class NeedDollar {
2  public static void main ( CommAreaHolder cah ) {
3       try {
4        // ## (1)
5        Channel ch = Task.getTask().createChannel("EuroToDollar");
6        // ## (2)
7        Container euroCon = ch.createContainer("EURO");
8        // ## (3)
9        String euroStr = "100";
10            euroCon.put(euroStr);
11            // ## (4)
12            Program moneyConverter = new Program ();
13            moneyConverter.setName("MONCONV");
14            // ## (5)
15            moneyConverter.link(ch);
17            PrintWriter out = Task.getTask().out;
18            out.println();
19            out.println("Give " + euroStr + " Euros");
20            // ## (6)
21            String dollarStr = new String(euroCon.get());
22            out.println("Get " + dollarStr + " Dollars");
24           } catch (Exception e) {
25            e. printStackTrace(out);
26           }
27      }
28 }
```

Listing 5: Anfrage mit Channel 2

```
1 public class MoneyConverter {
2  public static void main(CommAreaHolder cah) {
4       PrintWriter out = Task.getTask().out;
6       // ## (1)
7       Channel ch = Task.getTask().getCurrentChannel();
9       // ## (2)
10           if ( ch != null ) {
11            String chName = ch.getName().trim();
13            try {
15             // ## (3)
16             double resValue ;
17             if ( chName.equals("EuroToDollar")) {
18              // ## (3-a)
19              Container euroRec ;
20              euroRec = ch. getContainer("EURO");
21              double reqVal = Double.valueOf(new
String(euroRec.get()));
22              resValue = euroToDollar(reqVal);
24              euroRec.put(Double.toString(resValue));
26             } else if (chName.equals("DollarToEuro")) {
27              // ## (3-b)
28              Container dollarRec = ch. getContainer("DOLLAR");
29              double reqVal = Double.valueOf(new
String(dollarRec.get()));
30              resValue = dollarToEuro(reqVal);
31              dollarRec.put(Double.toString(resValue));
32             }
33            } catch (Exception e) {
34             e.printStackTrace(out);
35            }
36           }
37      }
38      // ## (4-a)
39      private static double euroToDollar(double euro) {
40       return (euro * 1.30);
41      }
42      // ## (4-b)
43      private static double dollarToEuro(double euro) {
44       return (euro / 1.30);
45      }
47 }
```

Listing 6: Java Channel Multiplexer

```
1 Task t = Task.getTask();
2 ContainerIterator ci = t.containerIterator();
3 while (ci.hasNext()) {
4   Container custData = ci.next();
5   // Verarbeitung Container Inhalt
6 }
```

Listing 7: Beispiel für einen ContainerIterator

Die Datei jcicsres02.zip enthält alle Java Resources als zip Archiv. Sie kann heruntergeladen werden unter

www.cedix.de/Vorles/Band3/Resources/jcicsres02.zip

6. Literatur

[1] Rayns, C., Clee, G.B., Grieve, T., Taylor, J., Ge, Y.P., Li, G.Q., Zhang, Q., Wen, D.: Java Application Development for CICS. 4 edn. IBM (2009)

[2] IBM: CICS Transaction Server for z/OS Release Guide. (3.1 edn.)

[3] IBM: Java Applications in CICS. IBM. 3.2 edn. (2008)

[4] IBM: (http://publib.boulder.ibm.com/infocenter/cicsts/v3r1/index.jsp?topic=/com.ibm.cics.ts31.doc/dfhpk/com/ibm/cics/server/package-summary.html)

[5] Adams, D.: The Hitchhiker's guide to the galaxy. delrey (1979)

[6] Gamma, E., Helm, R., Johnson, R.E., Vlissides, J.: Design Patterns. Elements of Reusable Object-Oriented Software. 1 edn. Addison-Wesley (1994)

[7] Oracle: http://download.oracle.com/javaee/1.4/tutorial/doc/index.html (2010)

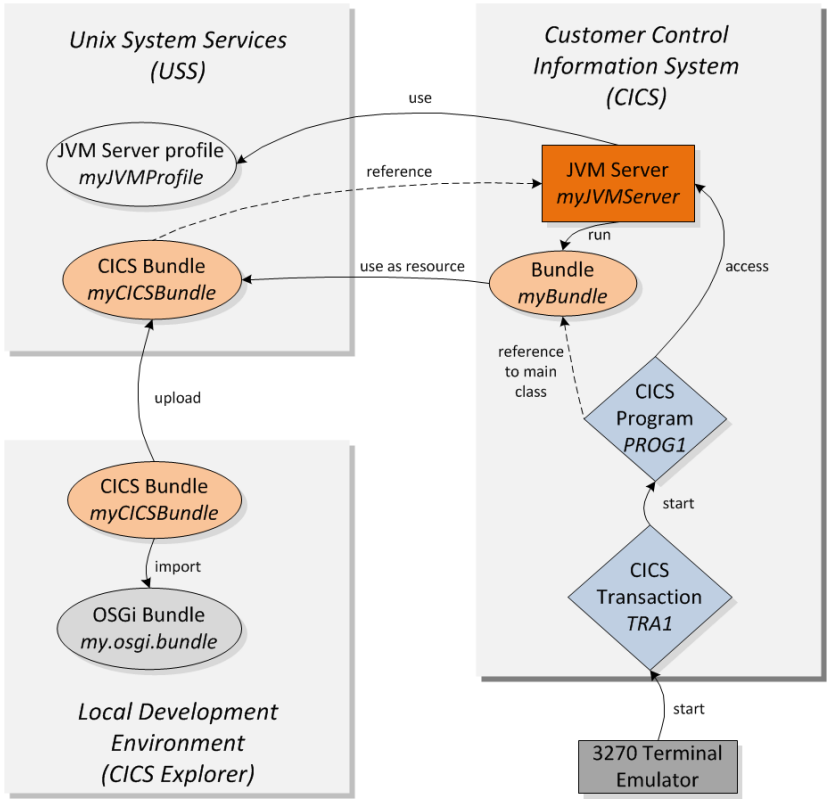
Practical Application Development with the CICS JVM Server

This tutorial practically shows the installation and configuration of the CICS Explorer as well as an example for the development of Java applications for the JVM Server.
The CICS Explorer is an interface for the local Eclipse IDE that is used for the management of CICS JVM Servers. It enables to communicate with the USS file system and with the CICS region.
In order to create a Java application for the JVM Server the following tasks have to be carried out (excluding the configurational tasks). Note that the basic procedure for this tutorial has been adapted from Transaction Server for z/OS Version 4 Release 2 - Java Applications in CICS.

1. Create a **Plug-in Project** in Eclipse that represents the OSGi bundle
2. Create a CICS bundle
3. Include the OSGi bundle into the CICS bundle and assign the JVM Server name needed for its execution
4. Export the CICS bundle into the USS file system
5. Create the following definitions in the CICS SM Perspective
   (a) JVM Server
   (b) Bundle
   (c) Program
   (d) Transaction
6. Install all definitions into the CICS region

A step by step procedure for all tasks is provided within this tutorial.

The interaction of OSGi bundles, the JVM Server, CICS programs and transactions is illustrated in the following



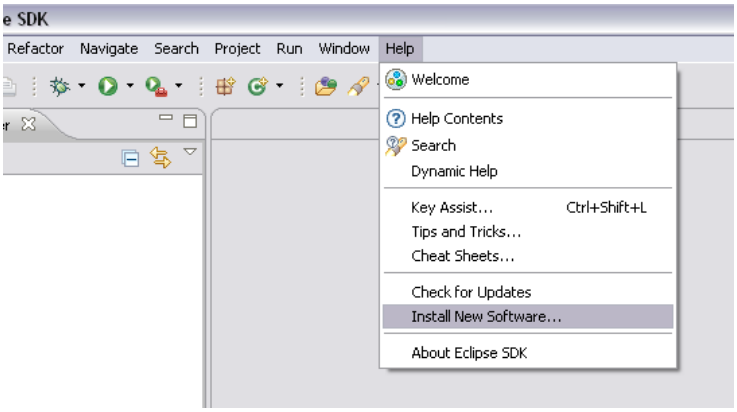
Install the CICS Explorer SDK into Eclipse

Requirements

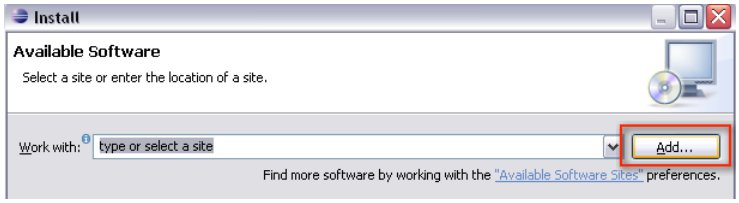
- Eclipse IDE v. 3.7 (and higher).
- CICS Explorer SDK (download from `http://www-01.ibm.com/software/htp/cics/explorer/`).

1. Start your Eclipse IDE.

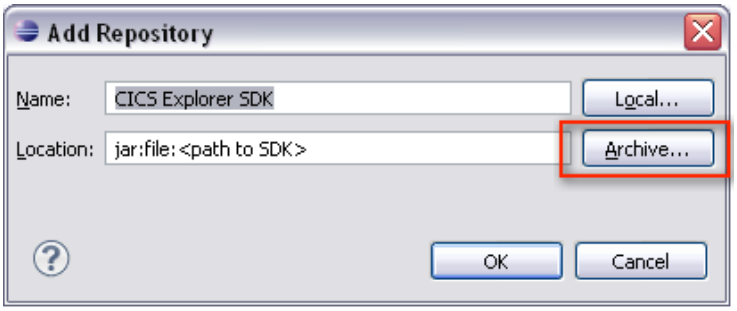
2. Go to **Window → Install New Software...**.



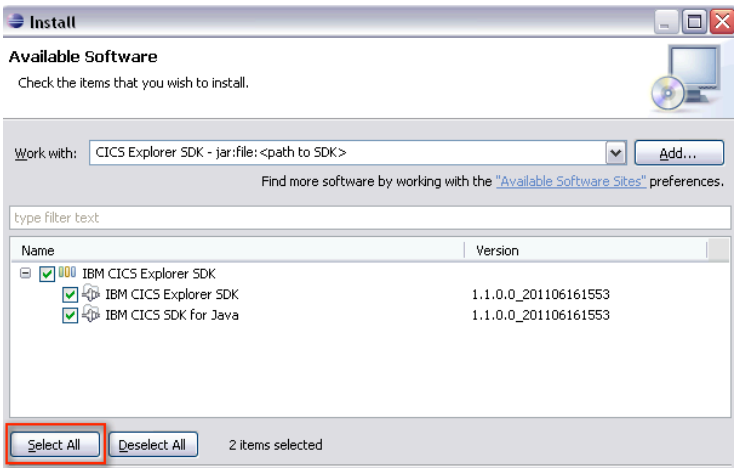
3. Click on **Add...** in the **Install** panel.



4. Name the repository, click on **Archive...** and browse to the CICS Explorer SDK *zip* file.



5. Select all options in the **Install** panel and click **Next**.

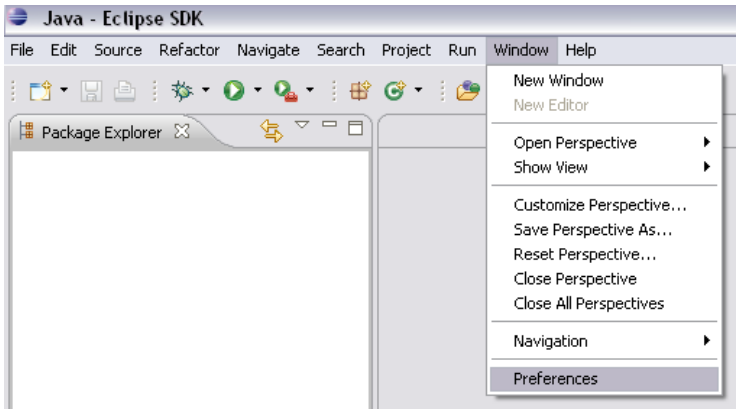


6. Accept the license agreement and finish the installation.
After restarting Eclipse several new Perspectives will be available. The ones important for this tutorial are the **CICS SM** and the **z/OS**.

Configure required Connections

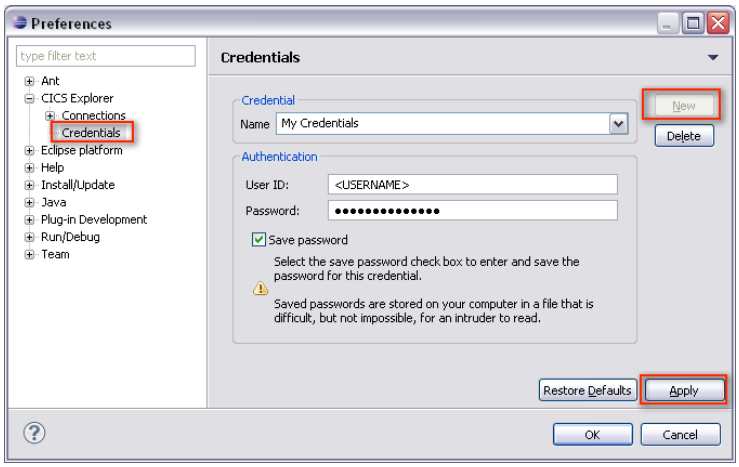
In order to run applications the connections to the CICS Management Interface and the z/OS interface need to be configured.

1. Go to **Window** → **Preferences**.

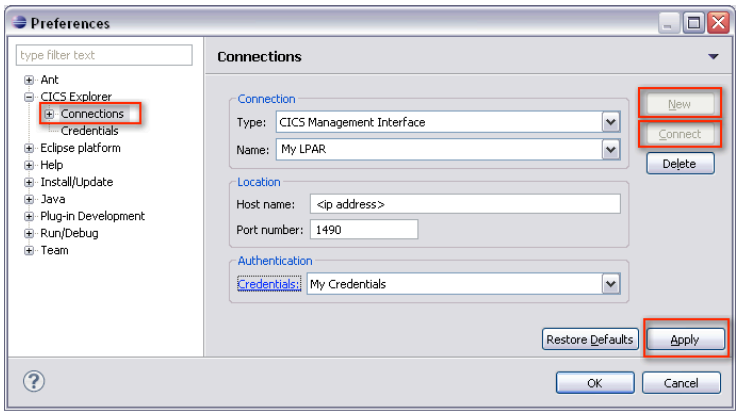


2. In the **Preferences** panel expand the **CICS Explorer**menu, click on the **Credentials** node and click on **New**.

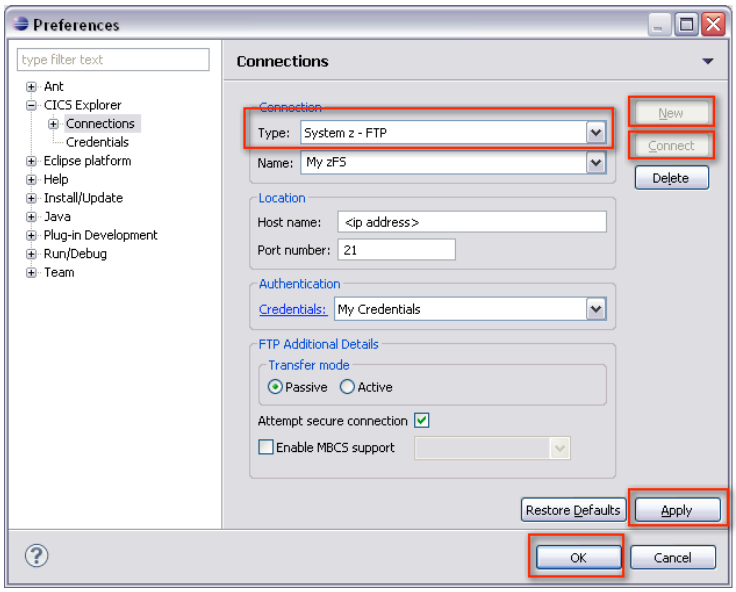
3. Name the credentials and enter your username and password and click on **Apply**. Keep the **Preferences** panel open.



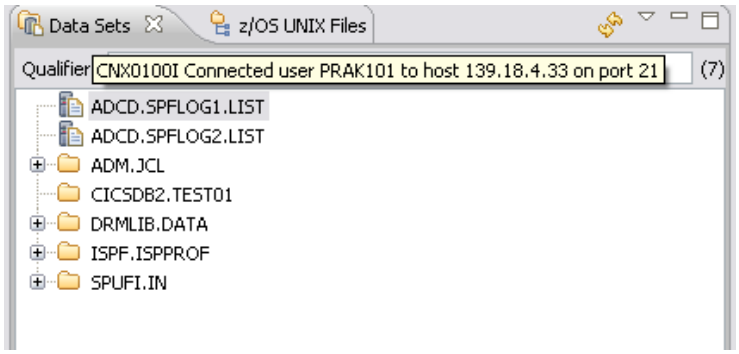
4. Click on the **Connections** node and create a new **CICS Management Interface** connection using the previously defined credentials. The default port for the CICS Management Interface is **1490**. Click on **Apply** and click on **Connect**. Keep the **Preferences** panel open.

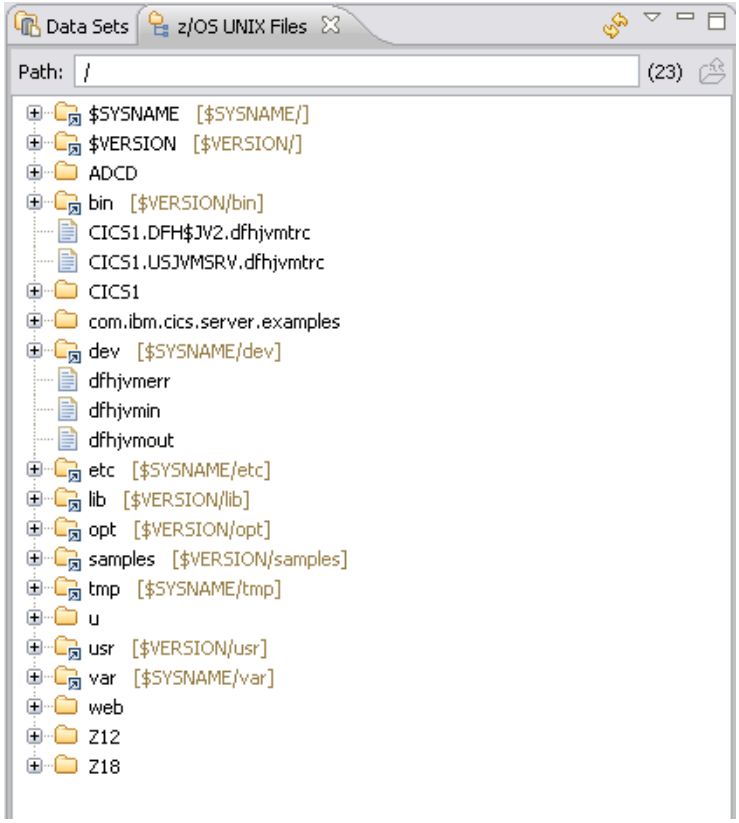


5. Choose **System z - FTP** as connection type and click on **New** to create a new FTP connection using the previously defined credentials. The default port for the **System z - FTP** is **21**. Apply the changes, click on **Connect** and click on **OK** to close the **Preferences** panel.



After successfully connecting to the System Z - FTP, you will be able to see all your z/OS and USS files within the **z/OS** Perspective.



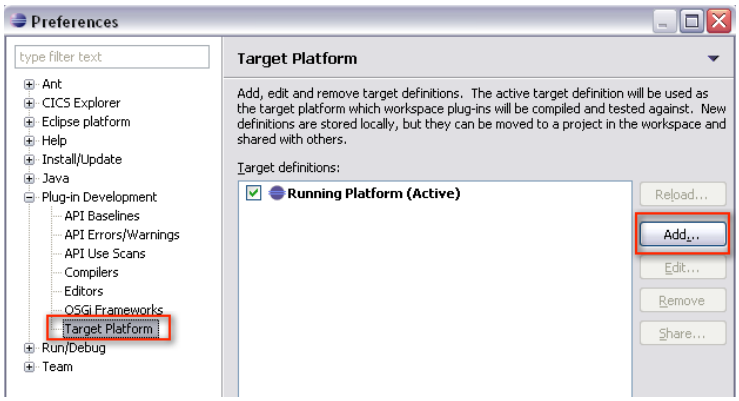


Configure the Target Platform

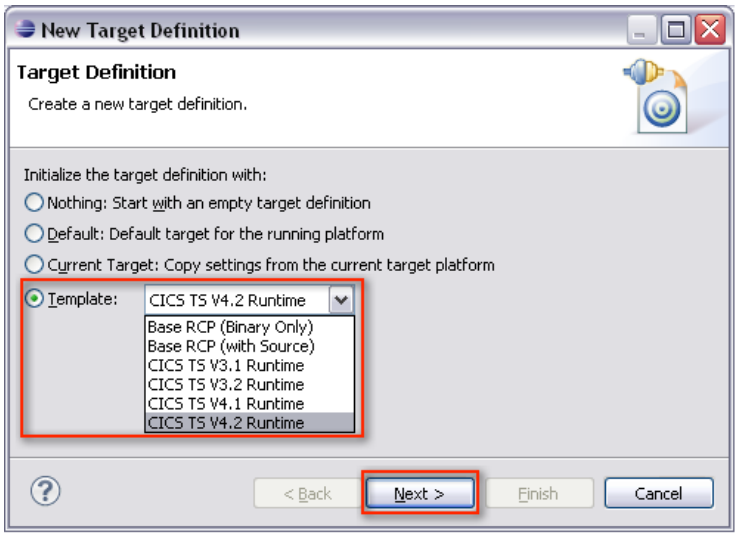
In order to develop applications for the CICS JVM Server, the target platform needs to be configured for the CICS TS 4.2 environment.

1. Go to **Window** → **Preferences**.

2. In the **Preferences** panel expand the **Plug-in Development** menu, click on the **Target Platform** node and click on **Add**.



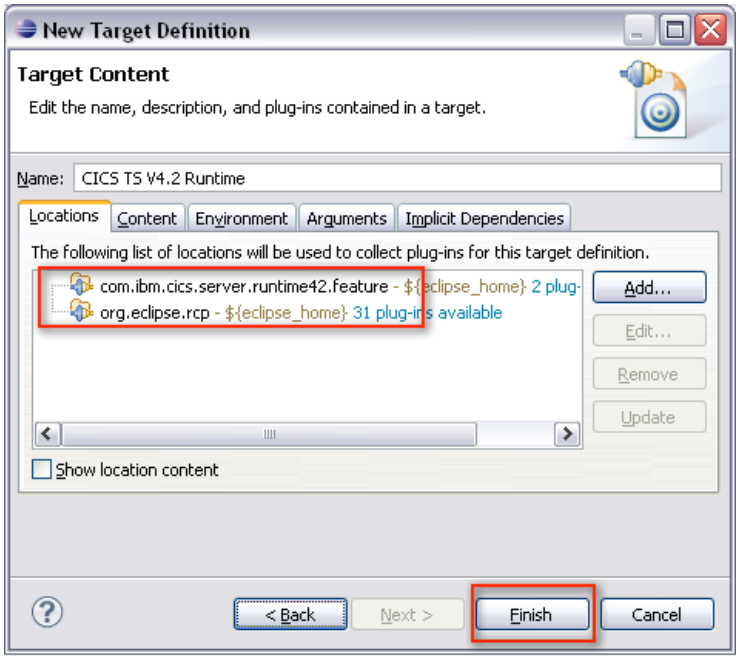
3. Choose **CICS TS 4.2 Runtime** as the template in the new window and click **Next**.



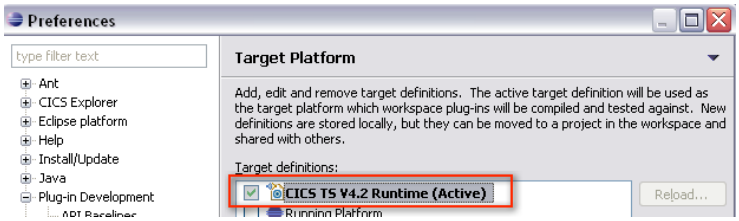
4. Review if the locations

- **com.ibm.cics.server.runtime42.feature** and
- **org.eclipse.rcp**

are included in the **Locations** tab of the **Target Content** panel and click on **Finish**.



5. Choose **CICS TS V4.2 Runtime** as the standard target platform and klick **OK**.

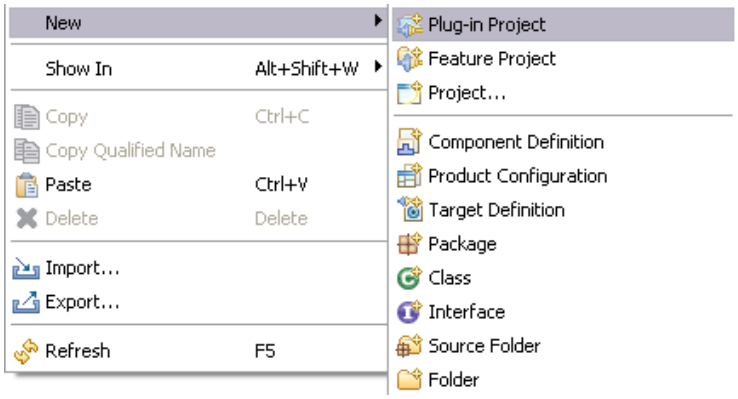


Create a HelloWorld Application

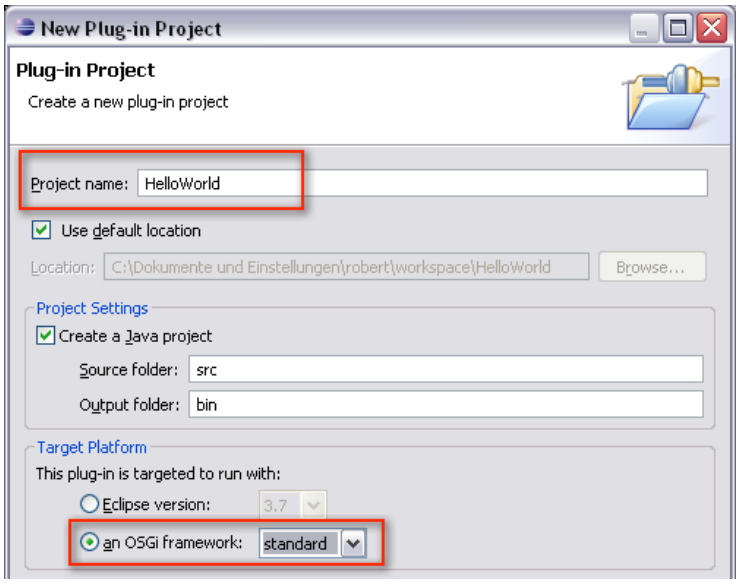
Since Eclipse is based on the OSGi Equinox Framework, OSGi bundles in Eclipse are designated as **Plug-ins**.

1. Open the **Plug-in Development** Perspective.

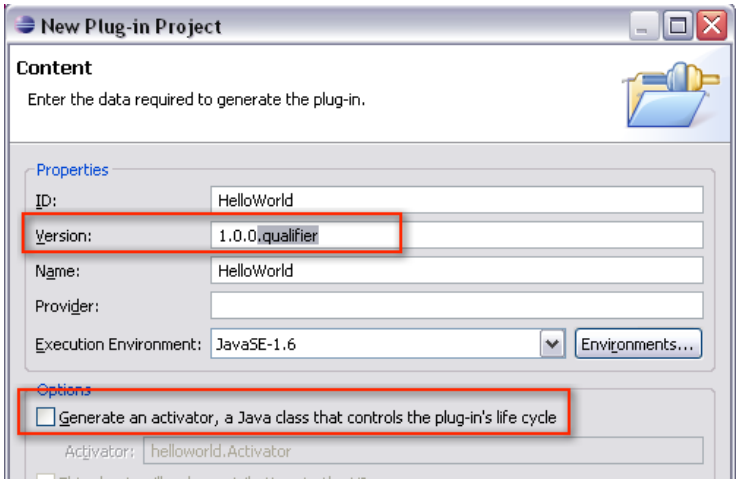
2. Create new Plug-in Project.



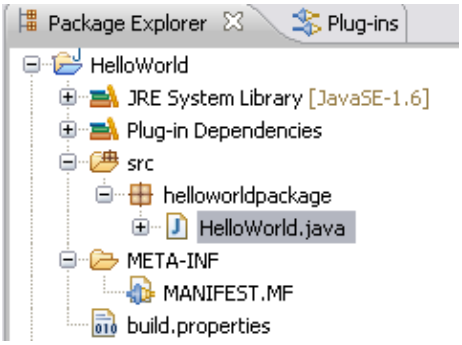
3. Enter the name of the project, choose **standard** as the target platform and click **Next**.



4. Remove the *qualifier*, uncheck the option for the generation of an activator and click **Next**.



5. Create a package **helloworldpackage** and a `HelloWorld` class in the previously created project.



6. Add the following code into the `HelloWorld` class (adapted from the **Hello CICS World** CICS Explorer Example).

```
package helloworldpackage;

import com.ibm.cics.server.*;

public class HelloWorld {

    public static void main(CommAreaHolder CAH)
    {
        Task t = Task.getTask();
        if ( t == null )
            System.err.println("HelloCICSWorld example: Can't get
                Task");
        else
                    t.out.println(" transaction started");
            t.out.println("Hello from a Java CICS application");
    }
}
```

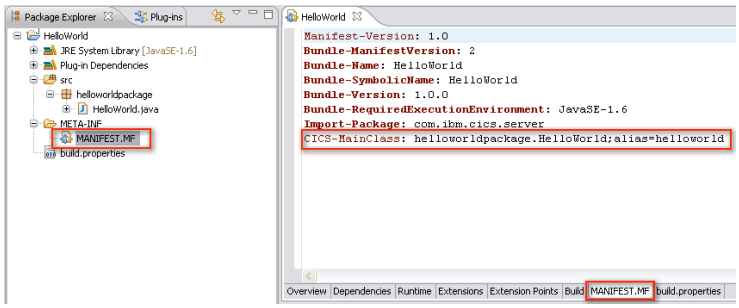
7. Open the MANIFEST.MF and add

`Import-Package: com.ibm.cics.server`

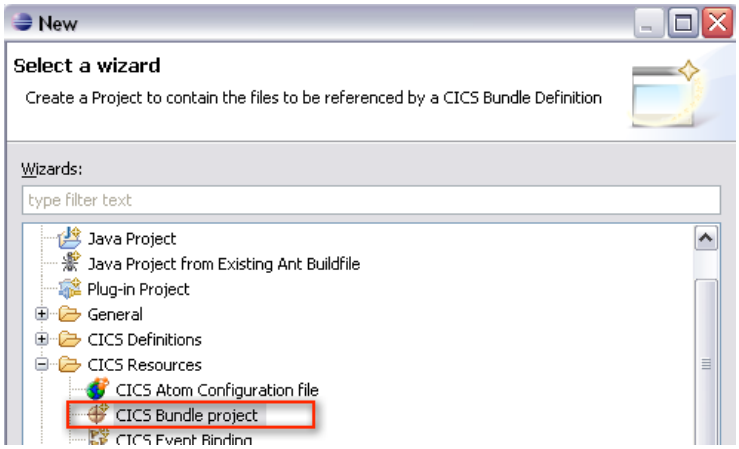
to enable the package import carried out in the `HelloWorld` class and

`CICS-MainClass: helloworldpackage.HelloWorld;alias=helloworld`

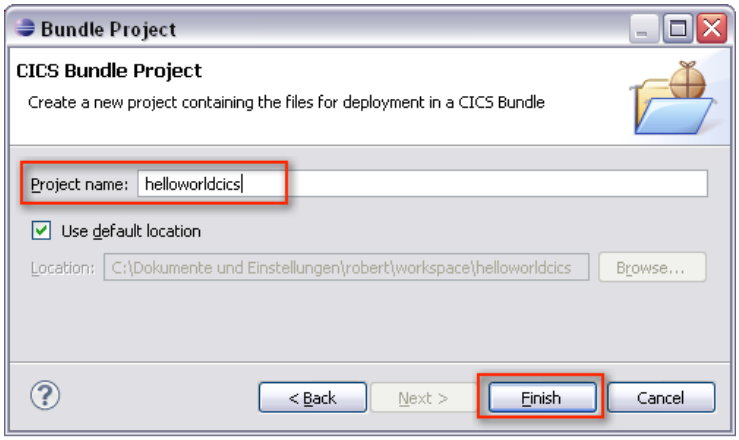
to identify the bundle and its package holding the main class. Press the key combination <**CTRL**>+**S** on your keyboard to save your changes. Note that the `alias=helloworld` statement assigns an optional alias for the main class. Also note that the last statement of the header file should include a line break (in this case a line break should appear in the CICS-MainClass line).



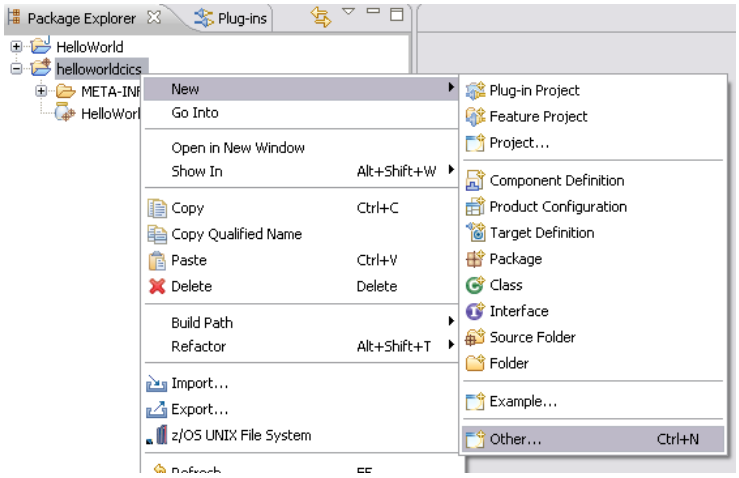
8. Go to **File** → **New** → **Other..** → **CICS Resources**, choose **CICS Bundle project** and click **Next** to create a CICS Bundle.



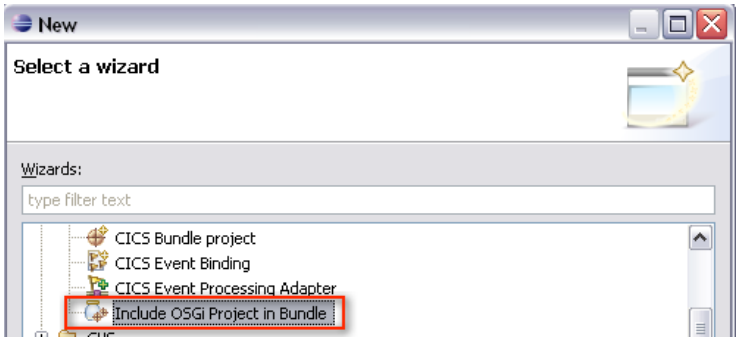
9. Name the CICS bundle and click **Finish**.



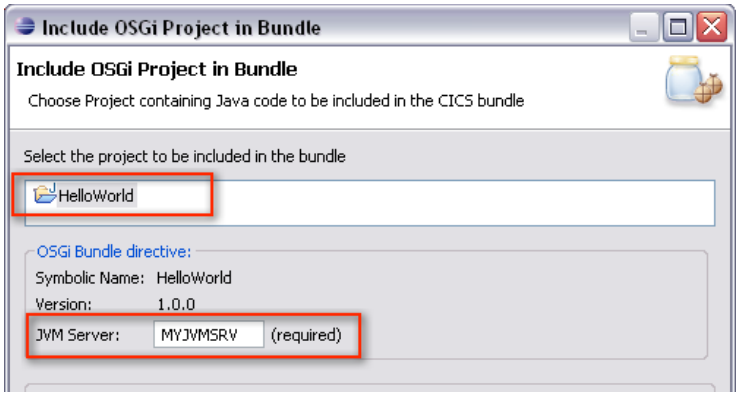
10. Perform a right-click on the previously created CICS bundle and go to **New → Other...**.



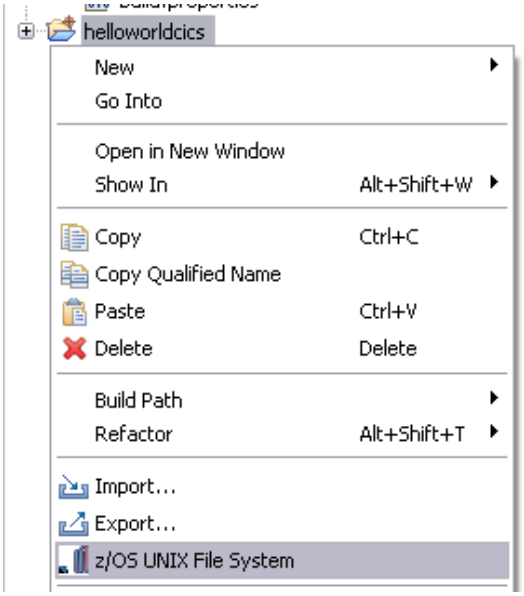
11. Choose **CICS Resources → Include OSGi Project in Bundle** and click **Next**.



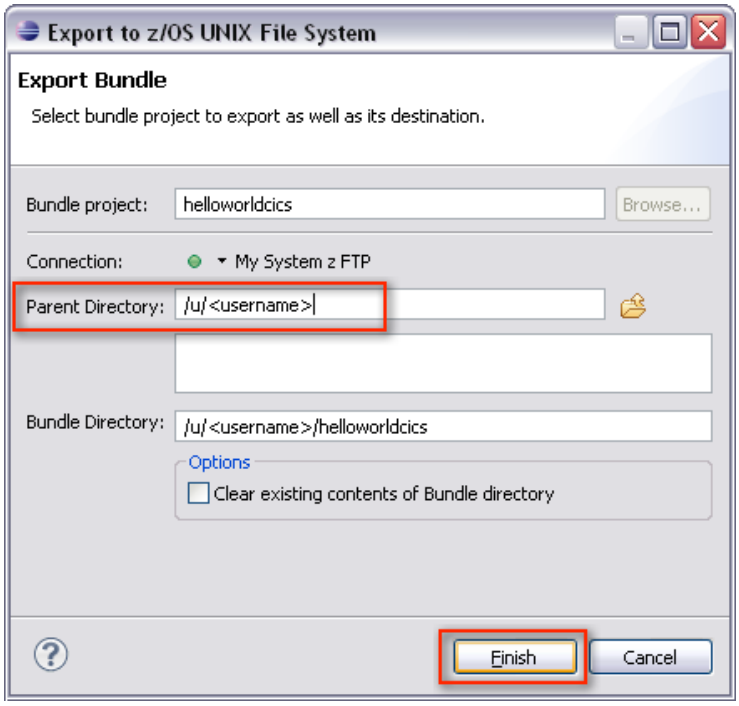
12. Choose the **HelloWorld** Plug-in project (OSGi bundle), enter the name of the JVM Server your application will use and click **Finish**. Note that a definition of a JVM Sever will be described step 18.



13. Click right on the previously created CICS bundle and go to **z/OS UNIX File System** to export the CICS Bundle into the USS file system.

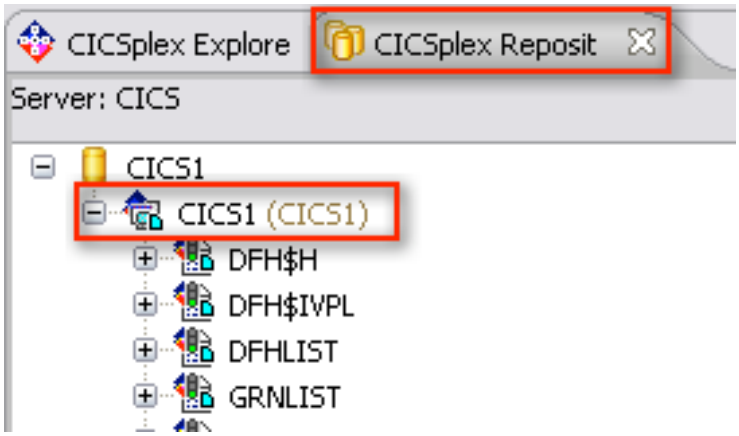


14. Choose the export directory and click **Finish**.



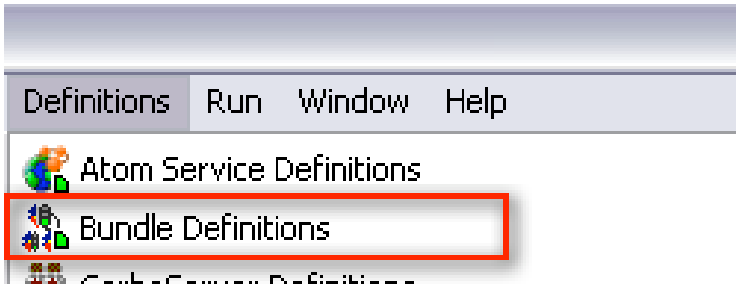
15. Open the **CICS SM** Perspective.

16. Click on the **CICSPlex Repositories** view, expand your CICS repository and select your CICS region

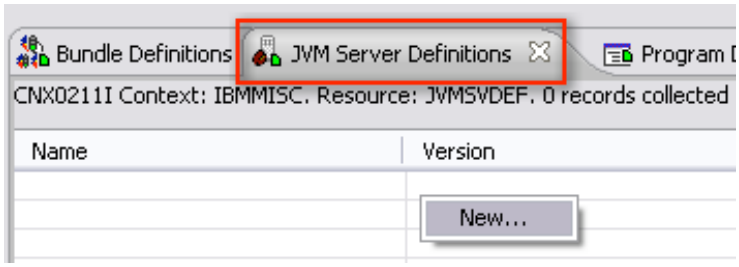


17. Since CICS requires a definition for each resource, it is mandatory to define a JVM Server, an OSGi bundle a program and a transaction.
Go to **Definitions** → **Bundle Definitions** / **JVM Server Definitions** /

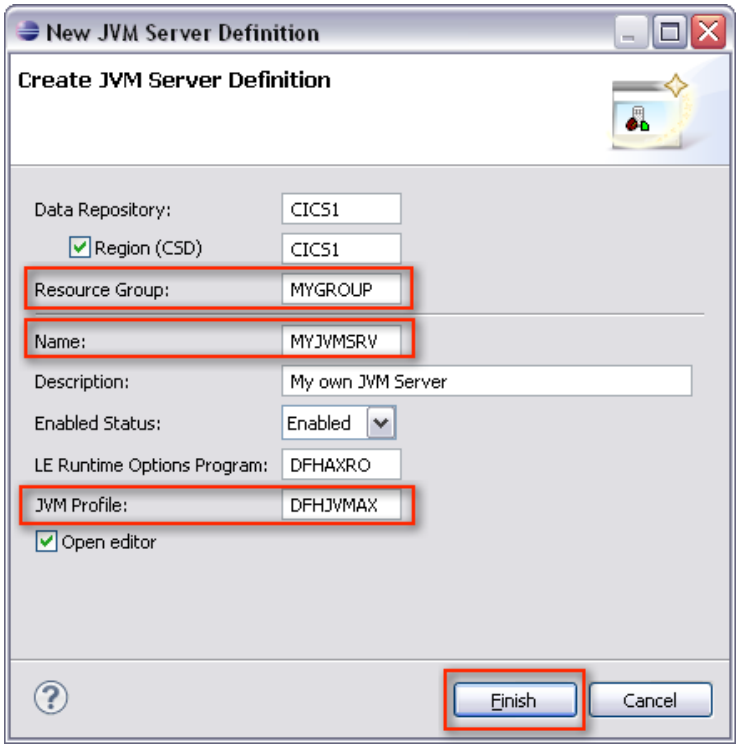
Program Definitions / **Transaction Definitions** to open all views needed for the definitions.



18. Activate the **JVM Server Definitions** view and perform a right-click → **New...** to create a new JVM Server definition. Skip this step if you want to use an existing JVM Server definition.



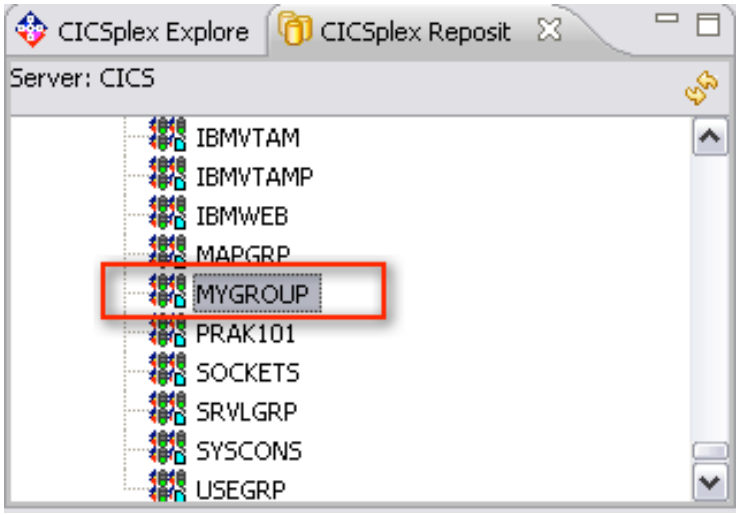
19. Assign the group (MYGROUP), define a unique name (MYJVMSRV) and assign the JVM profile (DFHJVMAX) to the JVM Server and click **Finish**.



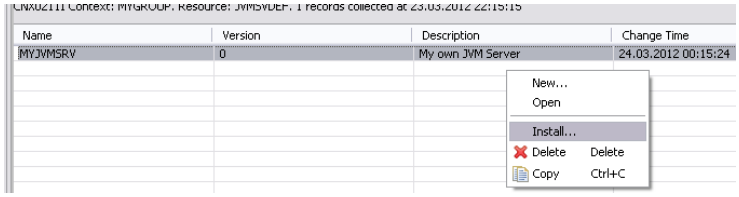
Description of options:

- **Resource Group:** Resources in CICS are grouped. Therefore, one needs to assign a group to the JVM Server. If the entered group is not defined, it will be created automatically.
- **Name:** The name is a unique identifier for the JVM Server. An OSGi bundle requires to be assigned to the JVM Server (refer step 12).
- **LE Runtime Option Program:** Assigns the program that defines the options for Language Environment. By default set to DHFAXRO. Do not alter this field.
- **JVM Profile:** The profile defines necessary parameters such as the working directory or optional startup parameters for the JVM. By default profiles are located in the USS file system directory:
  `/usr/lpp/cicsts/cicsts42/JVMProfiles`

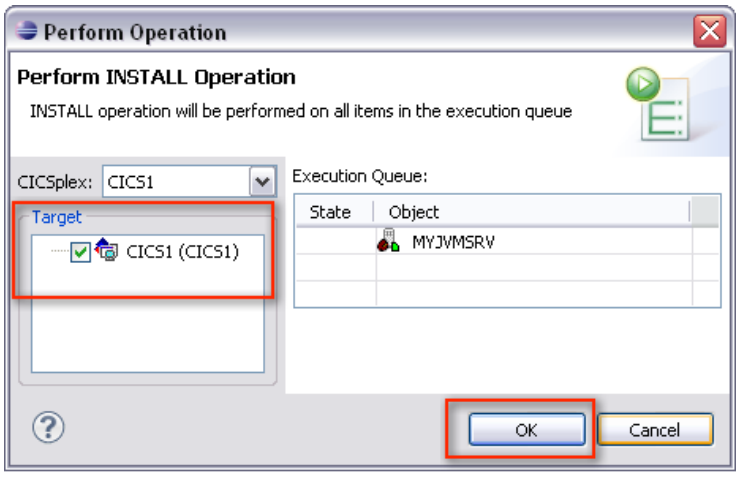
To get a user-friendlier look on your definitions, find your resource group and click in it. With it being activated, only your personal definitions will be displayed.



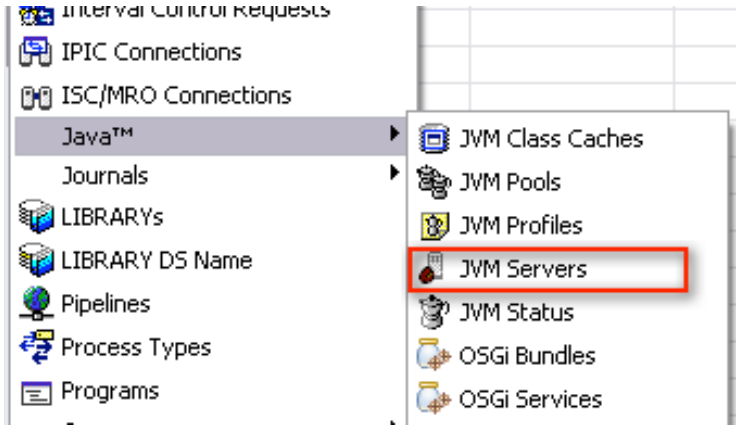
20. Perform a right-click → **Install...** on the new created JVM Server definition.



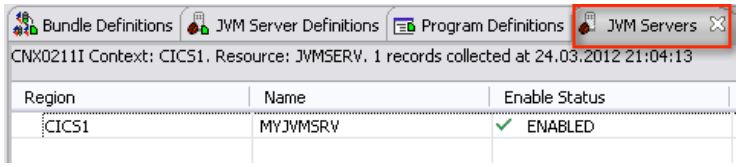
21. Choose the CICS region for the installation and click **OK**.



22. Go to **Operations** → **Java** → **JVM Servers** to open the **JVM Server** view.

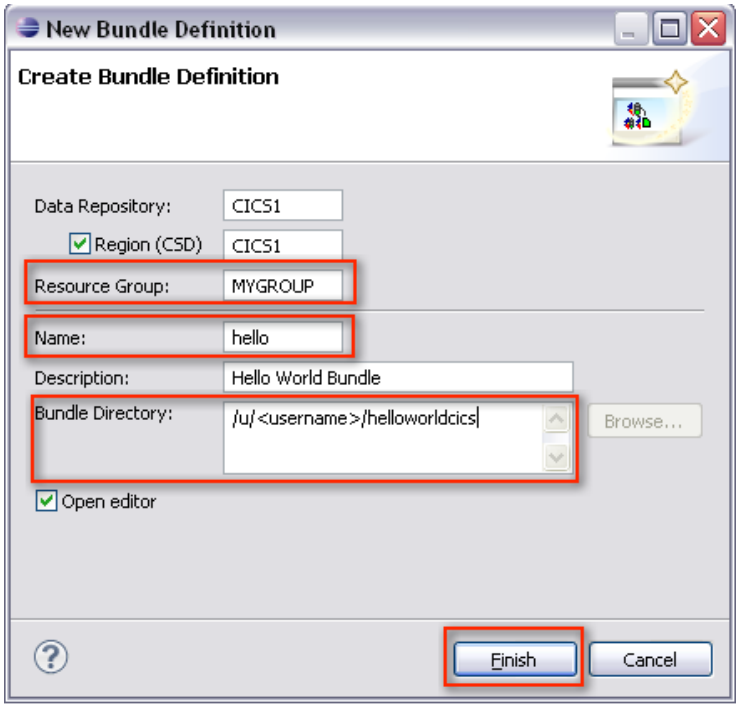


23. Activate the **JVM Servers** view, refresh the view and verify if the JVM Server installation was successful (Enable Status = ENABLED).



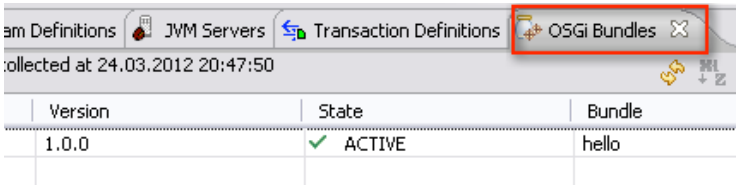
24. Activate the **Bundle Definitions** view and perform a right-click → **New...** to create a new bundle definition.

25. Assign the group (MYGROUP), define a unique name (hello) and the directory of the bundle chosen in step 14 and click on **Finish**.



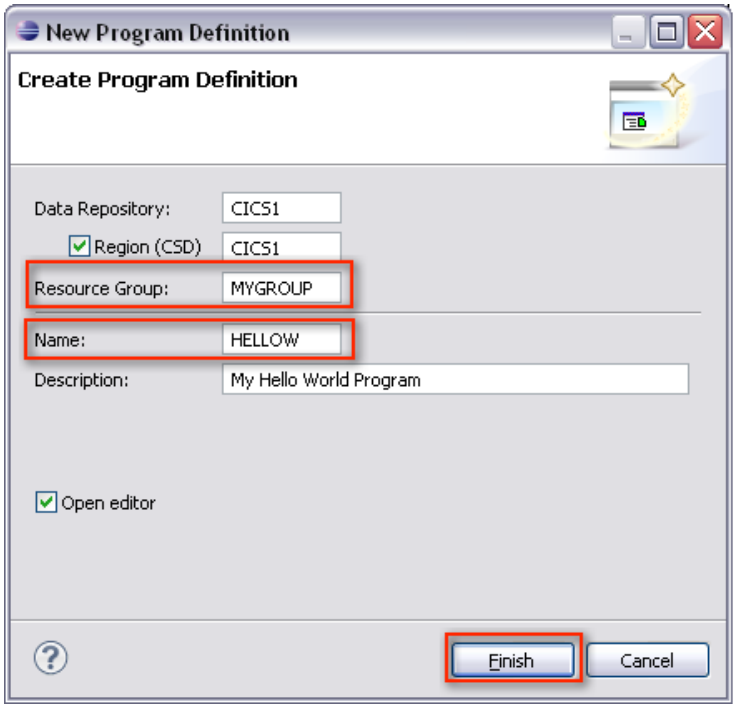
26. Perform a right-click → **Install...** on the new created OSGi bundle definition. Choose the CICS region for the installation and click **OK**.

27. Go to **Operations** → **Java** → **OSGi Bundles** to open the **OSGi Bundles** view. Activate the **OSGi Bundles** view, refresh the view and verify if the OSGi bundle installation was successful (State = Active).



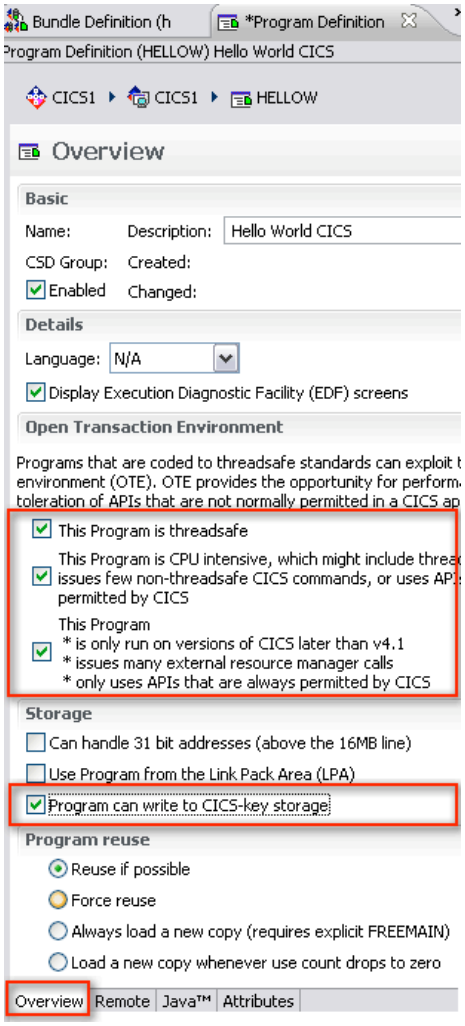
28. Activate the **Program Definitions** view and perform a right-click → **New...** to create a new program definition.

29. Assign the group (MYGROUP), define a unique name (HELLOW) to and click on **Finish**. A new view opens that enables to specify parameters for the program.

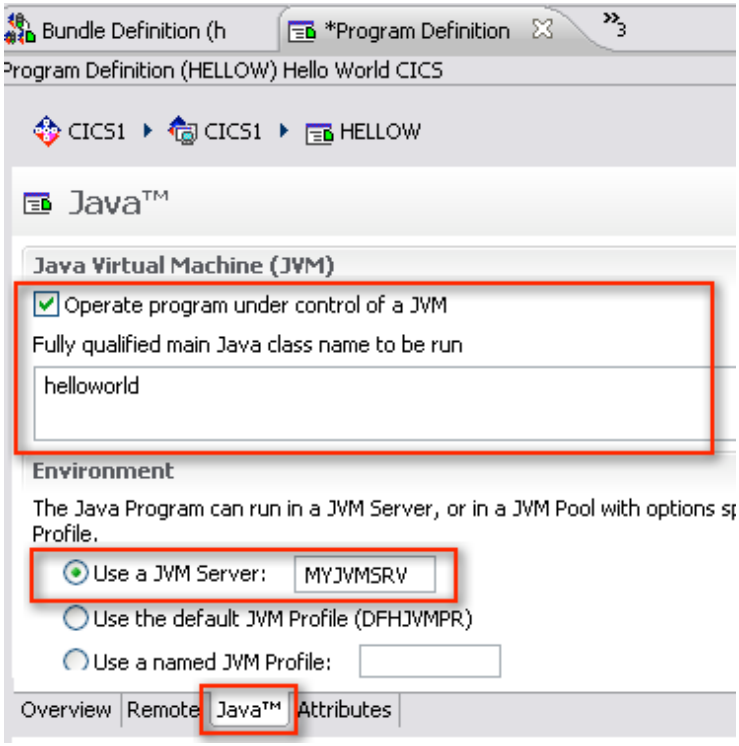


30. Activate the options for

- This program is threadsafe
  → Since the JVM Server is a multi application environment, all application are required to be threadsafe.
- This program is CPU intensive...
  → Enables higher priority scheduling.
- This program only runs on versions of CICS later that 4.1...
  → Since the JVM Server is introduced for CICS version 4.2, this option is mandatory.
- Program can write to CICS-key storage
  → Since the JVM Server uses a T8 TCB and since the storage protection key 8 is assigned to CICS, the program needs to be able to write to the CICS-key storage.



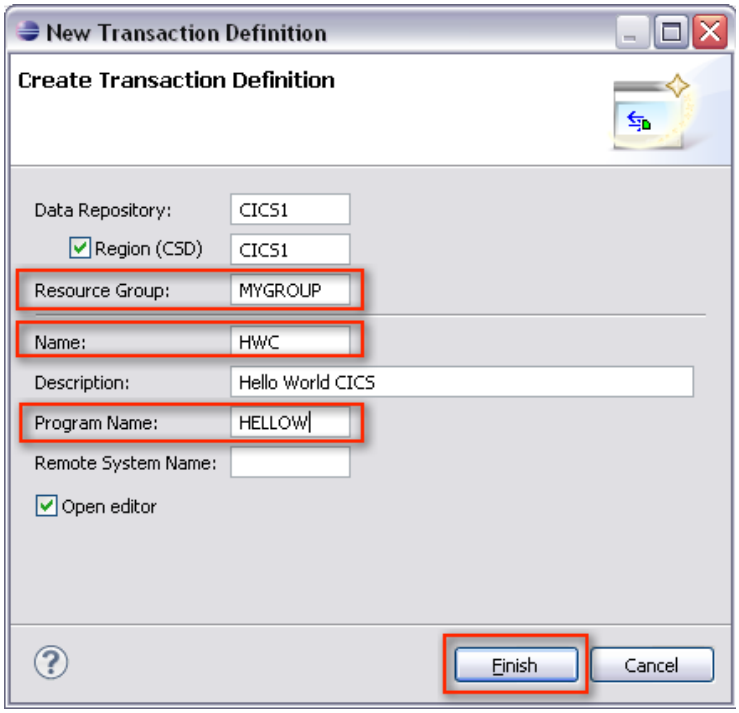
31. Activate the **Java** tab, check the option **Operate program under...**, insert the alias defined in step 7 (or the string `helloworldpackage.HelloWorld`), assign the JVM Server to the program and press the key combination **CTRL**+**S** on your keyboard to save your changes.



32. Activate the **Program Definitions** view and perform a right-click → **Install...** on the new created program definition. Choose the CICS region for the installation and click **OK**.

33. Activate the **Transaction Definitions** view and perform a right-click → **New...** to create a new transaction definition.

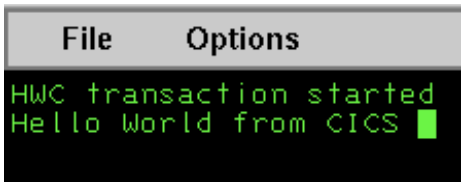
34. Assign the group (MYGROUP), define a unique name (HWC), assign the previously defined program (HELLOW) and click **Finish**.



35. Activate the **Transaction Definitions** view and perform a right-click → **Install...** on the new created program definition. Choose the CICS region for the installation and click **OK**.

36. Connect to CICS using a *3270* emulator and execute the previously installed transaction. The output should show the text

```
HWC transaction started
Hello World from CICS.
```



RDz Web Services Tutorial 01
Create a WSDL Description

Thanks to Mrs. Isabel Arnold, who conducted an IBM training session in Hamburg in August 2006, covering the material in this and the following 2 tutorials. The original Version of the material was created by Reginaldo W. Barosa, IBM Executive IT Specialist.

Content

1. Short Web Service overview

1.1 What is an XML Web Service?

The W3C defines a Web Service as a software system designed to support interoperable Machine to Machine interaction over a network. Web Services are frequently just Web APIs that can be accessed over a network, such as the Internet, and executed on a remote system hosting the requested services.

The W3C Web Service definition encompasses many different systems, but in common usage the term refers to

- **clients and servers that communicate using XML messages that follow the SOAP standard, and**
- **there is a machine readable description in the Web Services Description Language (WSDL) of the operations supported by the server.**

Some industry organizations, such as the Web Services Interoperability Organization (WS-I), mandate both SOAP and WSDL in their definition of a Web Service. Both are a prerequisite for automated client-side code generation in the mainstream Java and .NET SOAP frameworks.

XML Web Services offer the capability to describe interfaces in sufficient detail that users may create client applications that can communicate with the Web Services. These descriptions are contained in an XML-document named WSDL-document.

XML Web Services are registered. Potential users may find them using the UDDI-technology (Universal Description, Discovery and Integration) or an alternative directory service.

1.2 What is WSDL?

WSDL is an XML-based language for describing Web Services and how to access them. It is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate. Today, WSDL is used mostly in conjunction with SOAP.

WSDL is essentially an Interface Definition Language for SOAP.

The purpose of this tutorial is to illustrate the use of the WSDL editor to design a Web Services interface and produce new WSDL. There are many wizards in WDz that can be used to generate WSDL, but in this tutorial, we will start with an empty WSDL file and add appropriate information using the WSDL editor.

WSDL uses XML for describing Web Services. The description specifies

- **the characteristics of a Web Service:**
- **what the Web Service can do,**
- **where it resides, and**
- **how it is invoked.**

For an introduction into XML see
http://www.javaworld.com/javaworld/jw-04-1999/jw-04-xml.html?page=1

WSDL can be extended to allow descriptions of different bindings, regardless of what message formats or network protocols are used to communicate.

WSDL enables a service provider to specify the following characteristics of a Web Service:

- **Name of the Web Service and addressing information**
- **Protocol and encoding style to be used when accessing the public operations of the Web Service**
- **Type information: Operations, parameters, and data types comprising the interface of the Web Service, plus a name for this interface**

WSDL is a resumé for a business function implemented as a service. It describes what business functions are available, what information is to be pass to and from the business function, how the information is to get to the business function, and where the business function is located.

The business functions are represented by operations which are grouped into PortTypes. We can have as many operations as we would like in one PortType, but we would normally only specify closely related operations in a particular Port Type.

In this tutorial we will have a Port Type that represents functions a client can perform against data stored on a server (fund data). We are using a scenario of working for a mutual fund company, so the fund data is information our company has about the mutual funds it offers.

The operation (business function) we will add to our Port Type is addFund. (Optionally you can create additional WSDL files for getFund, deleteFund, and updateFund).

1.3 What is XSD

An XML Schema provides a means for defining the structure, content and semantics of XML documents. The XML Schema language is also referred to as XML Schema Definition (XSD).

We will use XSDs to describe the data layout of the information that is to be passed to and from the business operations. Our operations will receive and return messages. Messages may have one or more parts. A part may be a complex item with multiple elements. The message parts we will receive and send will point to the XSDs that describe the data. The XSDs contain multiple elements like Fund Id, Fund Name, Fund Price, etc.

The operations, messages, parts, and elements are used to generically describe our business functions, but do not tell us anything about their physical implementation. Binding information is added to the WSDL file to specify the protocol and transport used to access the operations. We are describing a Web Service. Web Services are accessed via SOAP over HTTP. WSDL can also be used to describe other types of 'services' that don't use SOAP over HTTP, but when discussing Web Services, you are discussing SOAP over HTTP.

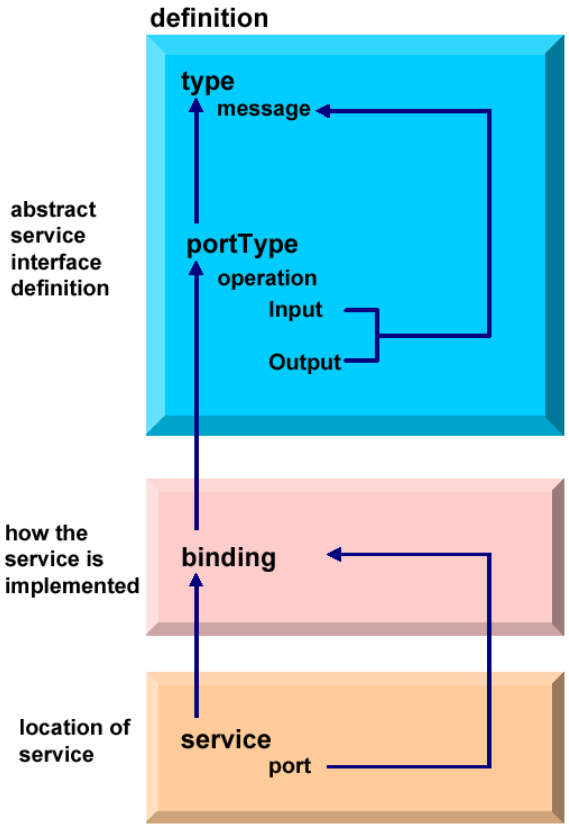
The last part of the WSDL file is the service information which details the location of the service.

1.4 WSDL Elements

A WSDL document contains the following six main elements:

1. **Port type. An abstract set of one or more operations supported by one or more ports. It includes an abstract description of an action supported by the service that defines the input and output message and optional fault message (section 2.6).**
2. **Message. An abstract, typed definition of the data being communicated. A message can have one or more typed parts (section 2.7).**
3. **Types. A container for data type definitions using some type system, usually XML Schema.**
4. **Binding. Specification of a concrete protocol and data format for a particular port type. The binding information contains the protocol name, the invocation style, a service ID, and the encoding for each operation Section 2.9).**
5. **Service. A collection of related ports. A WSDL document defines services as collections of network endpoints, or ports (section 2.10).**
6. **Port. A single endpoint, which is defined as an aggregation of a binding and a network address.**

WSDL supports the XML Schema Definition (XSD) specification.



WSDL elements and relationships

These are the elements comprising a WSDL document and the various relationships between them.

- **One WSDL document contains zero or more services. A service contains zero or more port definitions (service endpoints), and a port definition contains a specific protocol extension.**
- **The same WSDL document contains zero or more bindings. A binding is referenced by zero or more ports. The binding contains one protocol extension, where the style and transport are defined, and zero or more operations bindings. Each of these operation bindings is composed of one protocol extension, where the action and style are defined, and one to three messages bindings, where the encoding is defined.**
- **The same WSDL document contains zero or more port types. A port type is referenced by zero or more bindings. This port type contains zero or more operations, which are referenced by zero or more operations bindings.**
- **The same WSDL document contains zero or more messages. An operation usually points to an input and an output message, and optionally to some faults. A message is composed of zero or more parts.**
- **The same WSDL document contains zero or more types. A type can be referenced by zero or more parts.**
- **The same WSDL document points to zero or more XML Schemas. An XML Schema contains zero or more XSD types that define the different data types.**

1.4.1 Types

The types element encloses data type definitions used by the exchanged messages. WSDL uses XML Schema Definition (XSD) as its built-in type system:

```
<definitions ....>
   <types>
      <xsd:schema ..../>(0 or more)
   </types>
</definitions>
```

A simple types definition for example may have two schema sections: one defines the message format for the input and the other defines the message format for the output.

1.4.2 Messages

A message represents one interaction between a service requester and a service provider. If an operation is bidirectional, at least two message definitions are used in order to specify the transmissions to and from the service provider. A message consists of one or more logical parts.

```
<definitions ....>
   <message name="nmtoken">(0 or more)
      <part name="nmtoken"element="qname"(0 or 1)type="qname"(0 or 1)/>
                (0 or more)
   </message>
</definitions>
```

The abstract message definitions are used by the operation element. Multiple operations can refer to the same message definition.

1.4.3 Port types

A port type is a named set of abstract operations and the abstract messages involved:

```
<definitions ....>
   <portType name="nmtoken">
      <operation name="nmtoken"..../>(0 or more)
   </portType>
</definitions>
```

WSDL defines four types of operations that a port can support:

1. **One-way: The port receives a message. There is an input message only.**
2. **Request-response: The port receives a message and sends a correlated message. There is an input message followed by an output message.**
3. **Solicit-response: The port sends a message and receives a correlated message. There is an output message followed by an input message.**
4. **Notification: The port sends a message. There is an output message only. This type of operation could be used in a publish/subscribe scenario.**

1.4.4 Bindings

A binding contains:

- **Protocol-specific general binding data, such as the underlying transport protocol and the communication style for SOAP.**
- **Protocol extensions for operations and their messages include the URN and encoding information for SOAP, for example.**

A port references a binding. The port and binding are modelled as separate entities in order to support flexibility and location transparency. Two ports that merely differ in their network address can share the same protocol binding.

For a Web Service, the binding usually specifies SOAP. HTTP, MIME EJB, JMS, and plain Java transport bindings are other alternatives.

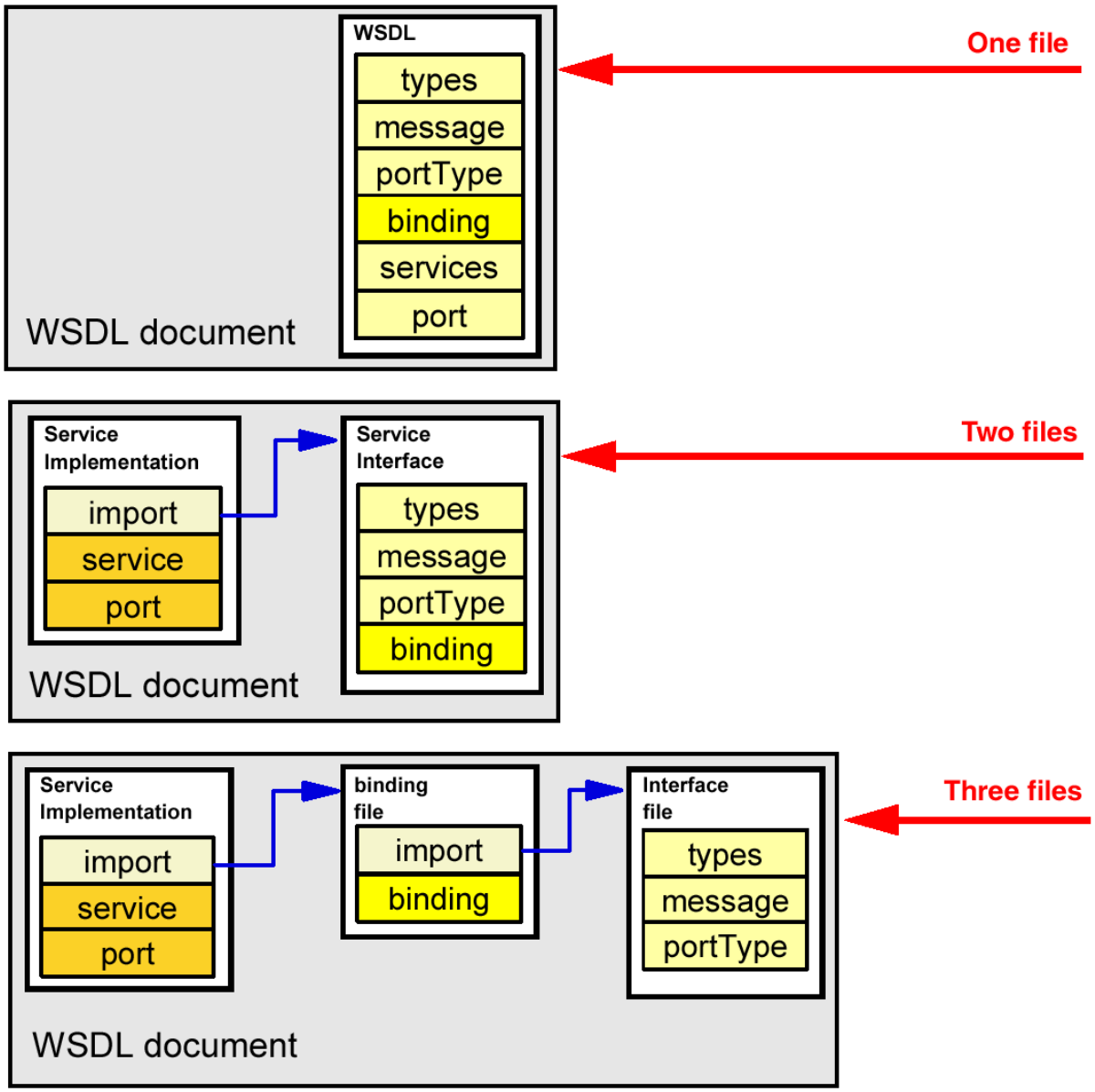
1.4.5 Port definition

A port definition describes an individual endpoint, usually an URI (for example an URL), by specifying a single address for a binding.

1.4.6 Service definition

A service definition merely bundles a set of ports together under a name,

1.5 WSDL document Overview



WSDL document Overview

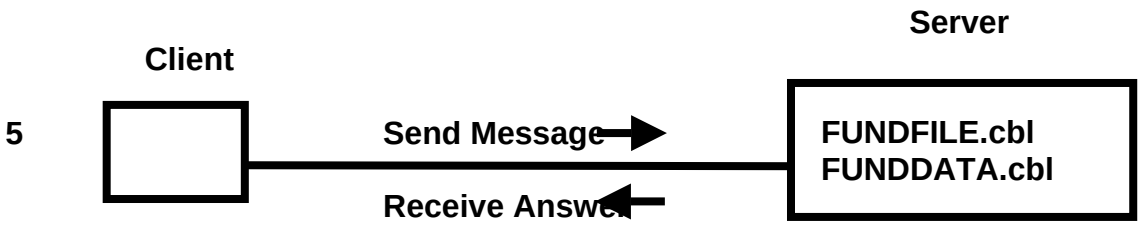
A WSDL document can be created in one or more physical files. If they are more than one, we have to connect these files using an import element. This separation of files can be convenient to hold the abstraction of concepts and to allow better maintenance.

Three files are typically used in WebSphere Developer for System z (and also in the underlying Application Developer).

2. WSDL-Editor Tutorial

Scenario

The purpose of this tutorial is to create a WSDL description, that can be used by a client to access a Webservice located on a server. In our scenario, you work for a mutual fund company. In order to make one of the existing business functions located on the server available as a Web Service, you will be describing the interface of the business function in WSDL.



Our Server contains a Web Service enabled COBOL package, including two datasets FUNDFILE.cbl and FUNDDATA.cbl. We will generate a WSDL description of a Web Service that permits the client by using this description to send a service request message to the server and receive an answer.

We will add an operation called addFund. The addFund operation will take an input and output message to be defined. The addFund operation will take all fields in the file as input, and will return the data that was added.

Tutorial Requirements:

Please note that there are often several ways to perform functions in WDz. This tutorial will present one of the ways. If you are familiar with WDz, you will notice that some of the statements are general, and not necessarily true for every situation. The main intent of this tutorial is to reinforce the discussions on SOAP and WSDL.

The following are other assumptions made in this tutorial.

Assumption 1: A VMWare image was supplied that contains WDz V7.0 and the files that are needed for this tutorial

Assumption 2: The VMWare image starts a Windows XP OS using “unilp” as login and password.

2.1 Tutorial Overview

1. Tutorial Overview (this section)

2. Initial Setup
In this part of the tutorial we will change the default line length in WDz for XML files. Although this step is not required, longer lines are required by the WSDL specification. The directions in part 1 have detailed information on why we want to extend the line length.

3. Create a Project
Our artifacts (various files and programs) are organized in WDz by projects. In this part of the tutorial we will create a simple project to hold the artifact we will use in this tutorial.

4. Create Data Schema Definitions
The information we will pass to and from our operations will ultimately go to a COBOL program. In this part of the tutorial we will generate some XSDs (XML Schema Definitions) that describe the layout of input and output of the COBOL program.

5. Create an Empty WSDL File
There are several wizards in WDz that could create WSDL for us, but to reinforce our discussions about WSDL and the parts of WSDL, we will start with an empty WSDL file and add all the information to the file that our application needs.

6. Add the addFund Operation
The business function that the WSDL will represent is to add a fund to a file. The name of this operation will be addFund.

7. Add Messages
The application data that is sent to and received from operations are called messages. Messages may have one or more parts. Each part may be a complex part with one or more elements. In this tutorial we will add messages that represent the data that will be passed to and from our addFund operation.

8. Add a fault to the addFund Operation
The usual way to indicate a problem with an operation is to return a fault. In this part of the tutorial we will add information about the fault that will be returned if a problem occurs with the addFund operation.

9. Add Binding Information to the WSDL File
After we have indicated our business operation and the information that goes to and from our operations, we need to specify how to access our business operation. WSDL Binding information is used to indicate the protocol and transport that should be used. In this tutorial we construct WSDL for a Web Service. A Web Service is SOAP over HTTP. There are also other types of services that use other protocols and transports.

10. Add Service Information
We also need to specify where the Web Service is located. The location of the Web Service is specified in the Service information.

11. (Optional) Create more WSDL files
As an optional exercise, create a WSDL file (one each) for the getFund, updateFund, and deleteFund services.

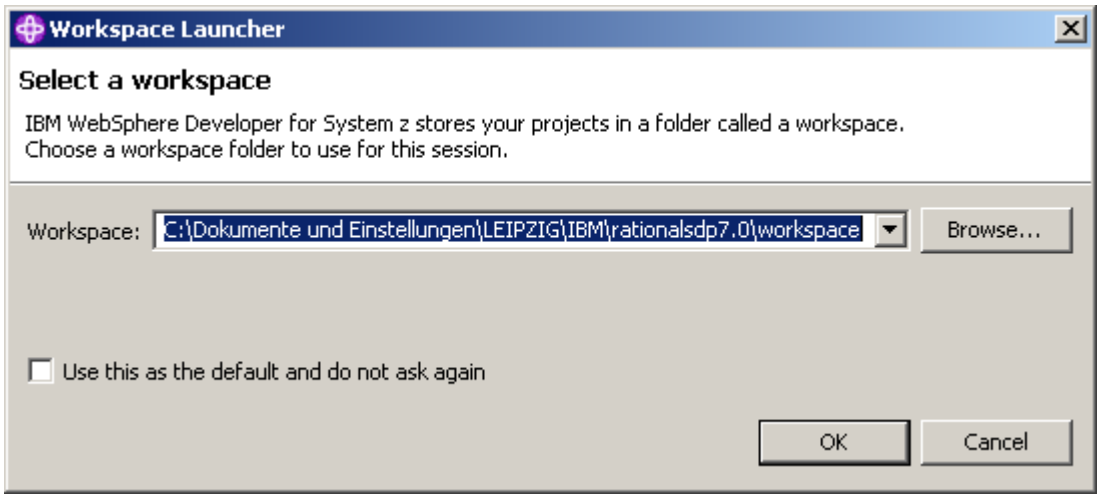
2.2 Initial Setup

The default width for the XML editor is 72 characters. Some of the lines we will be generating need to be longer than 72 characters to be compliant with the WSDL specification. To avoid syntax problems we will change the XML editor width to 100 characters. This is a one-time setup for the workspace.

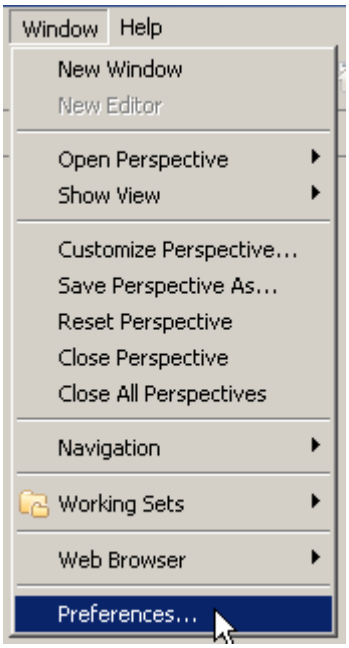
Note that if we do not change this, the wizard will still properly add code, but some lines may be broken into two or more lines where the WSDL specification does not allow lines to be broken. Although we could manually go in and correct the problem (or use short names), it is easier to change the XML editor line width.

Note also that WSDL is normally stored on a PC or in USS files on z/OS, so expanding the character width doesn't present any problems.

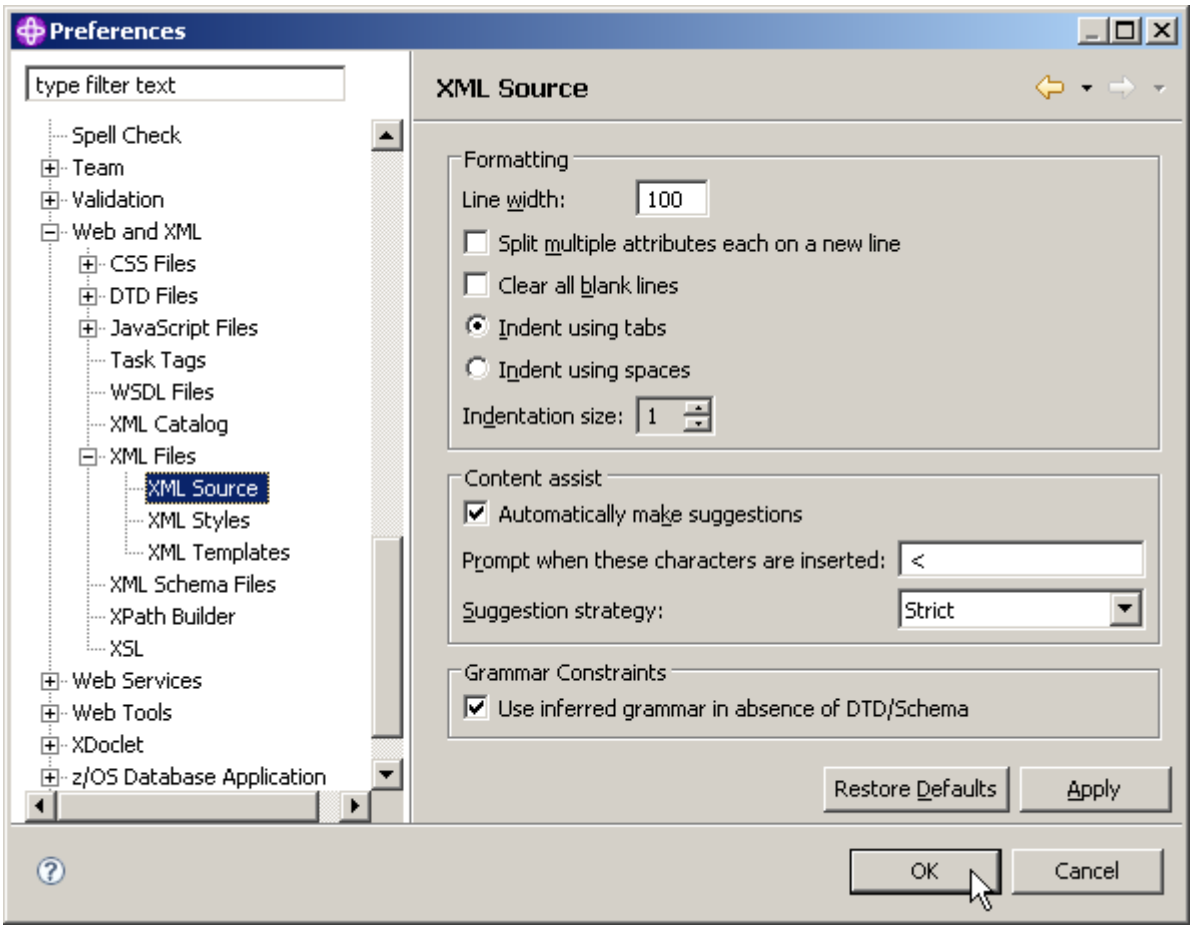
1.Start WDz if it is not already running.



2.If asked, Use the default workspace as proposed. Click OK.



3.From the menu bar, select Window → Preferences...

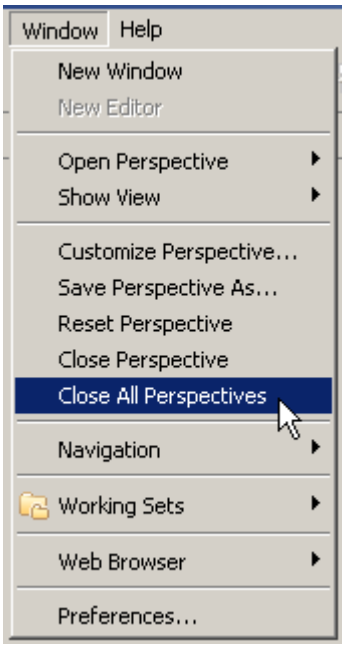


4.In the navigation on the left, Select Web and XML → XML Files → XML Source.
On the right, change the Line width to 100, then click OK.

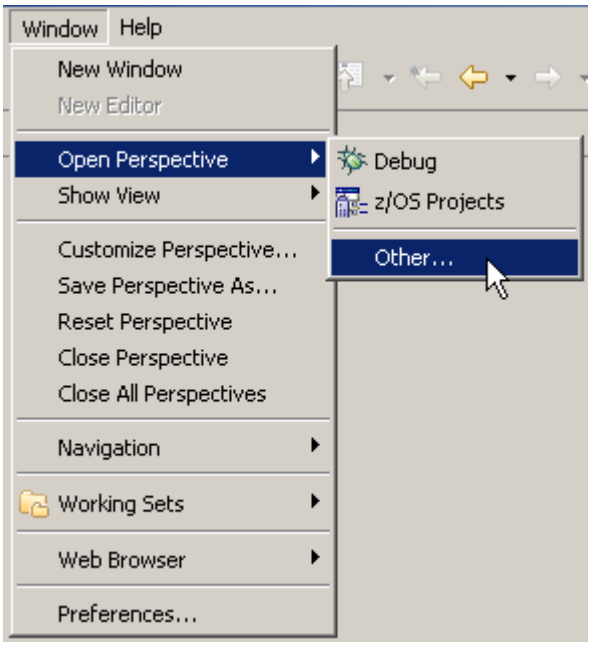
2.3 Create a Project

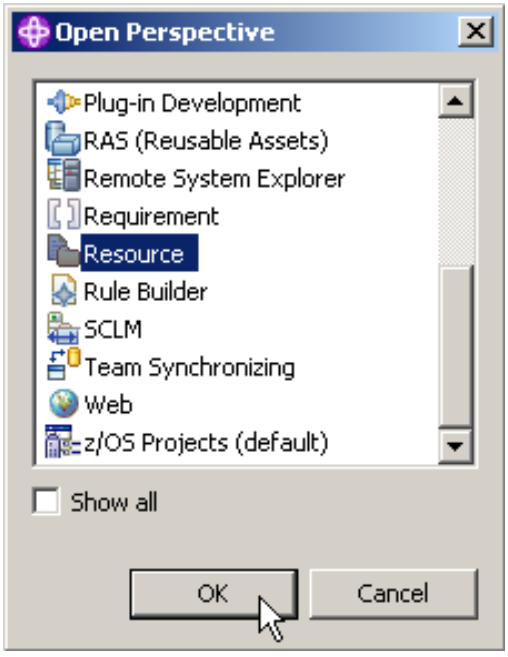
Computer artifacts are physical files that execute or are used by your software. Artifacts may include specifications, designs, code, use cases, class diagrams, etc. An artifact is one of many kinds of tangible byproduct produced during the development of software.

The various artifacts that we create in WDz are organized into projects. In this part of the exercise, we will create a simple project named WSDLCreate. We will be working from the Resource Perspective, but the functionality we will be using is available in any type of project in any type of perspective. The menus and wizards we will be using look at file extensions to determine the functionality available for this file.



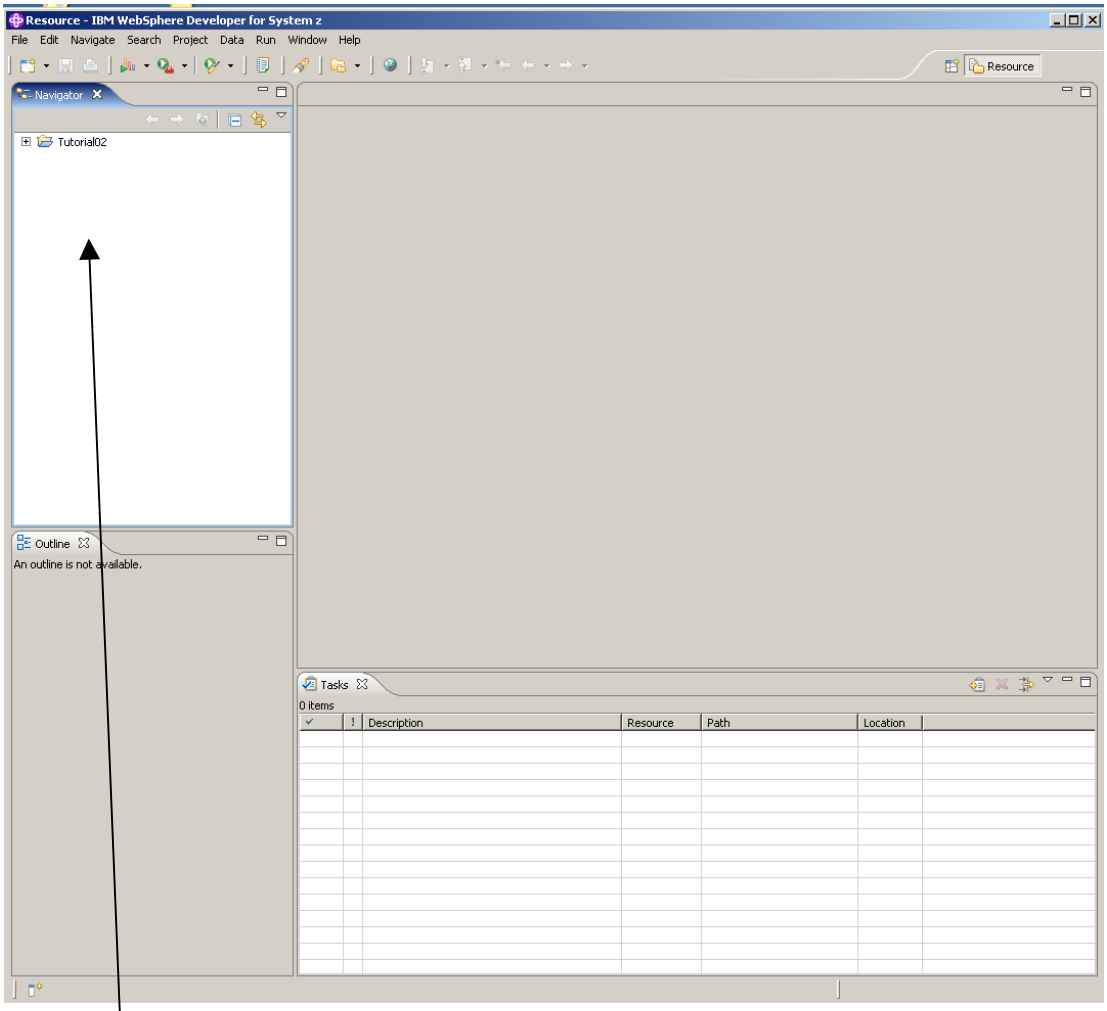
5.From the menu bar, select Window → Close All Perspectives.



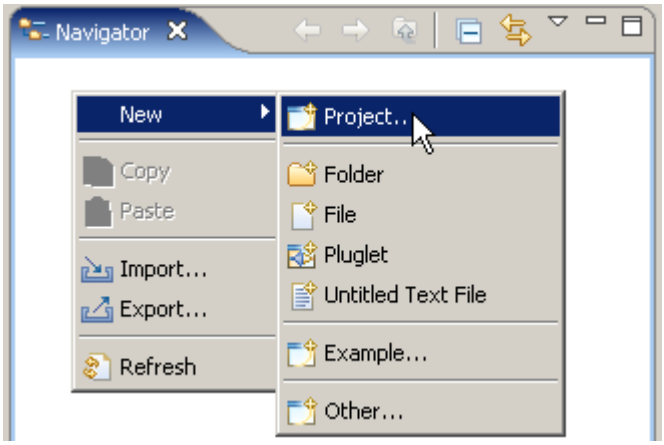


6.Then select Window → Open perspective → Other... and select Resource to open the resource perspective.

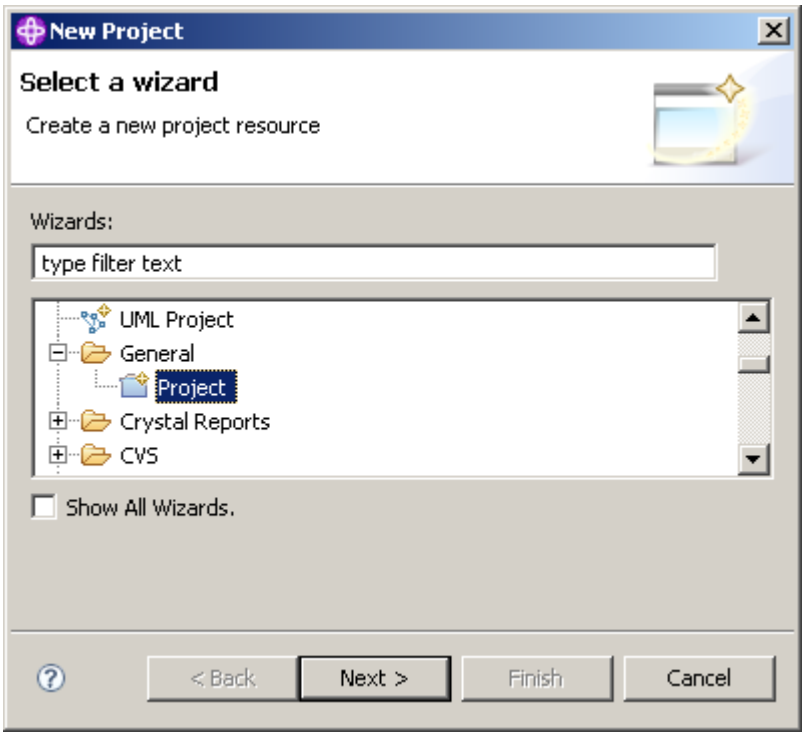
Click OK.



The screen now looks like shown above.:

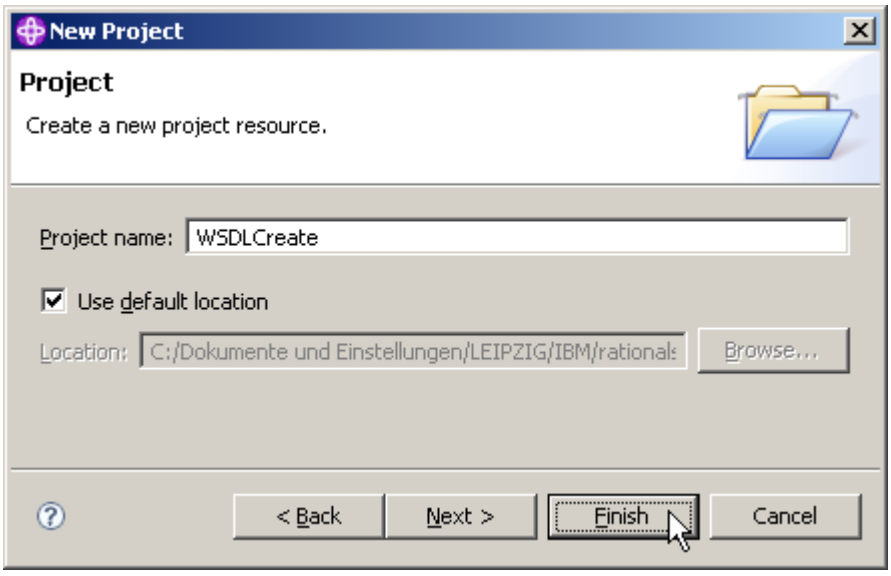


7. In the Navigator view, right-click in an open area, and from the context menu select New → Project…

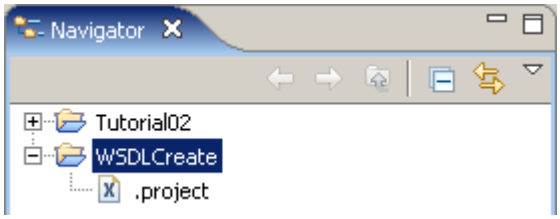


8.Expand General and select Project.

Click the Next button.



9.We decide to name our new Project WSDLCreate. On the New Project panel, specify the Project name as WSDLCreate, ensure Use default location is checked, then click the Finish button.



The Navigator view now contains a new entry WSDLCreate.
And if you expand it you will see that it is empty besides the project description file .project.

2.4 Create Data Schema Definitions

We will first import the COBOL program FUNDPROG.cbl and copybook FUNDFILE.cbl.

FUNDPROG.cbl is a regular Cobol program that is supplied with this tutorial. You may want to use a text editor to have a look at both FUNDPROG.cbl and FUNDFILE.cbl .

FUNDFILE.cbl is a Cobol copybook A copybook can be a copy of a part of any source code: working-storage, environment division, procedure division, etc. If there is source code in a COBOL program, which will be used in multiple programs, it is often put in a copybook.

A copybook is most frequently used to define the physical layout of data (either in an input or output file (file schemas), or in a temporary area (working storage)), but can also be used for sections of procedural code.

Data files, used by COBOL programs, are formatted according to record structures that can be defined in COBOL copybooks. A COBOL copybook declares the name and data types of variables that associated COBOL programs use to exchange information via the program's I/O Area (IMS) or COMMAREA (CICS). A copybook may be included in more than one COBOL program and a COBOL program can import more than one copybook. External applications passing information to the COBOL programs have to use a format declared in the imported copybook definitions.

A COBOL Copybook describes that data. The reason for using it is to allow multiple programs to reference the same structure without recoding in each program. It works like an INCLUDE statement, and it is up to the programmer to make sure the data complies with the description.

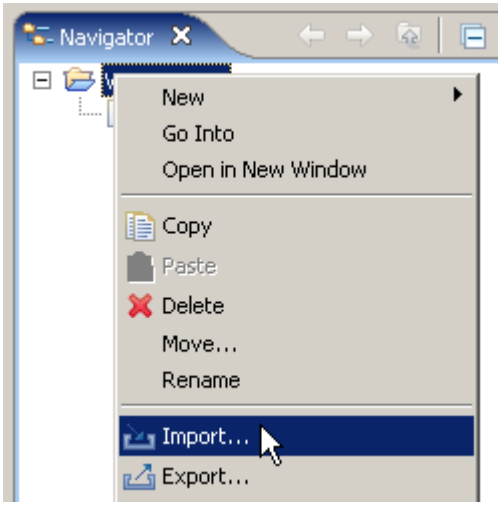
The information that will be passed to and from our Web Service will be similar to the data that is used in our target COBOL program. In this part of the exercise we will create some XSDs (XML Schema Definitions) that correspond to the Data Definitions in our COBOL program.

XML-Schema (bzw. XSD = XML-Schema-Definition) ist die moderne Möglichkeit, die Struktur von XML-Dokumenten zu beschreiben. XML-Schema bietet auch die Möglichkeit, den Inhalt von Elementen und Attributen zu beschränken, z. B. auf Zahlen, Datumsangaben oder Texte, z. B. mittels regulärer Ausdrücke. Ein Schema ist selbst ein XML-Dokument.

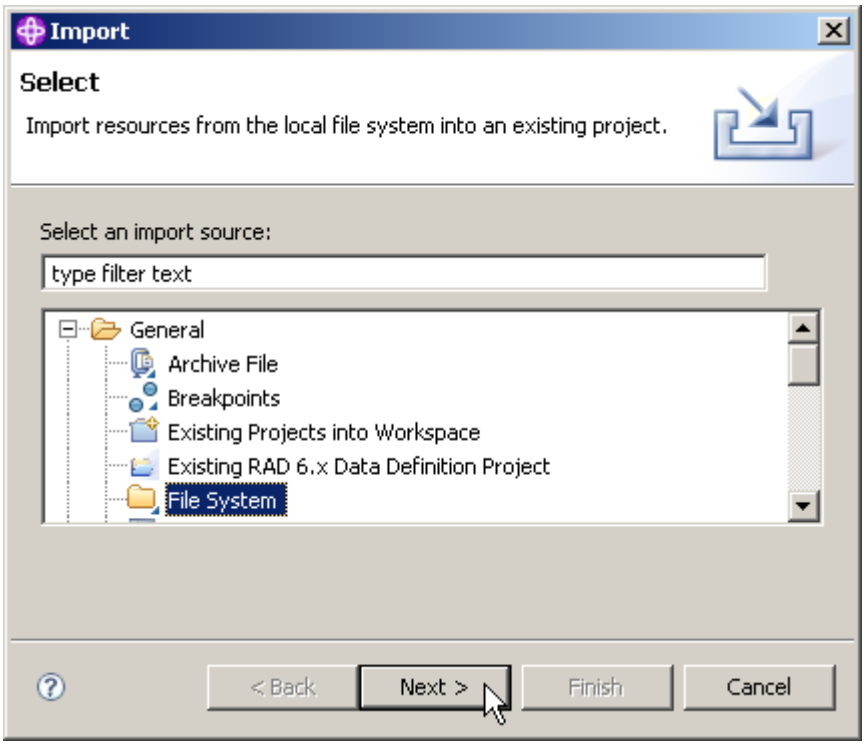
(In a later part of this tutorial, we will use the XSDs as descriptions of the messages that are passed to and from the Web Service operation.)

The WSDL editor in WDz allows you to define complex data layouts in XSD from scratch, but in this tutorial we will use an easier approach, and will generate corresponding XSDs from the existing COBOL data definitions.

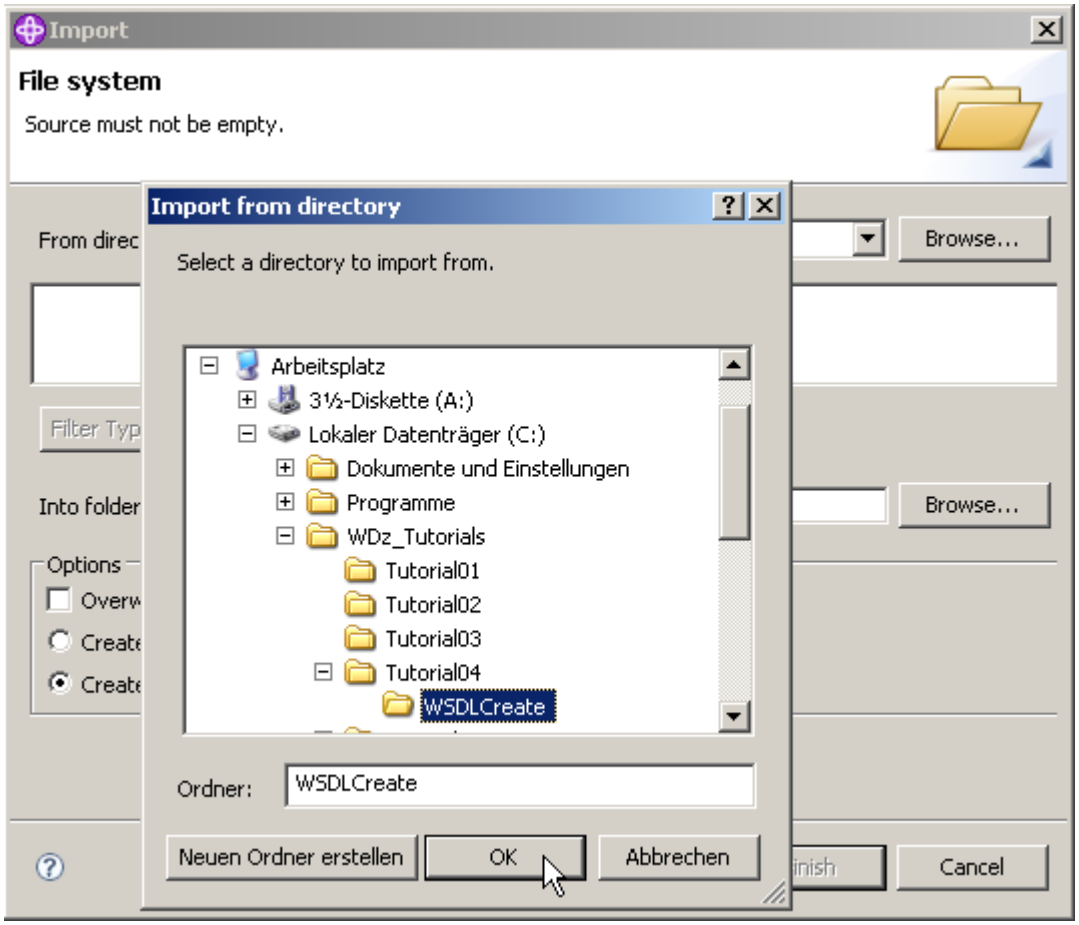
We will then use a wizard to generate the XSDs from FUNDPROG.cbl and FUNDFILE.cbl. This will save us the effort to generate the WSDL manually. However, the intent of this tutorial is to reinforce concepts discussed in the XML, SOAP, and WSDL lectures.



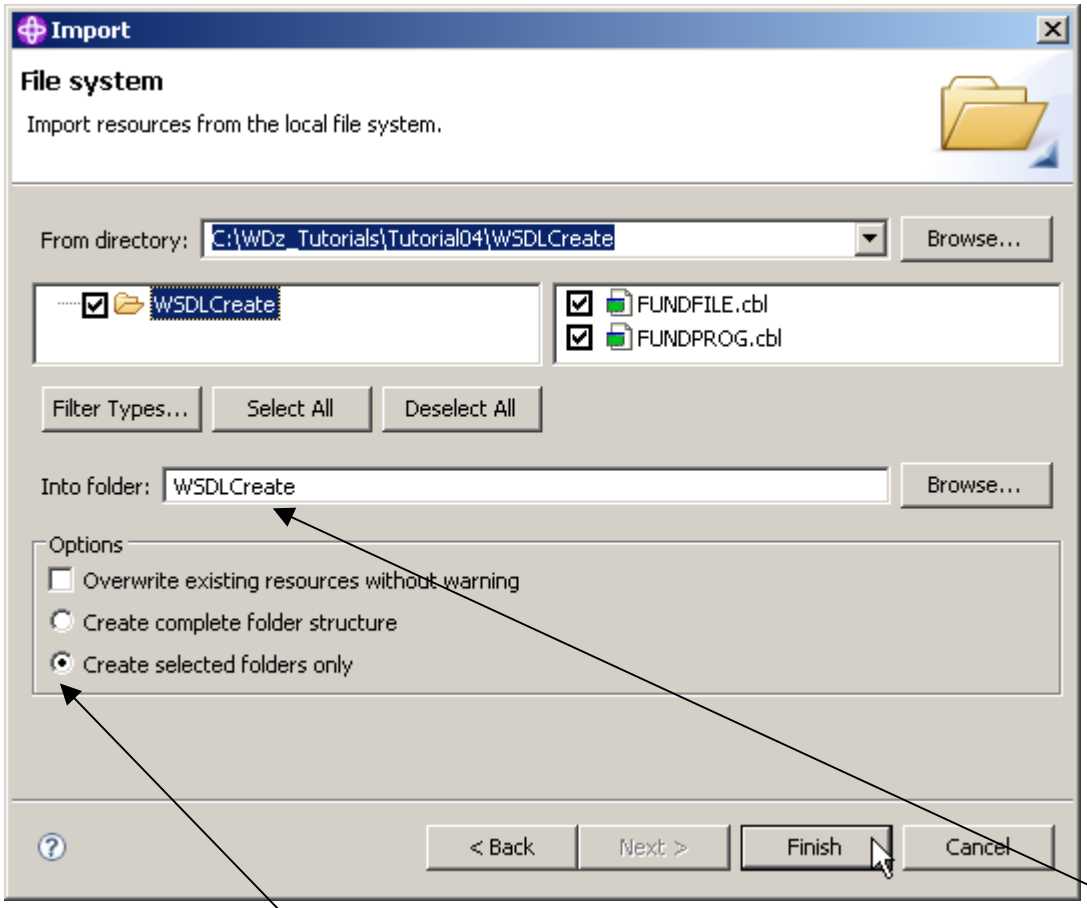
10.From the Navigator view, right-click the WSDLCreate project, and from the context menu, select Import...



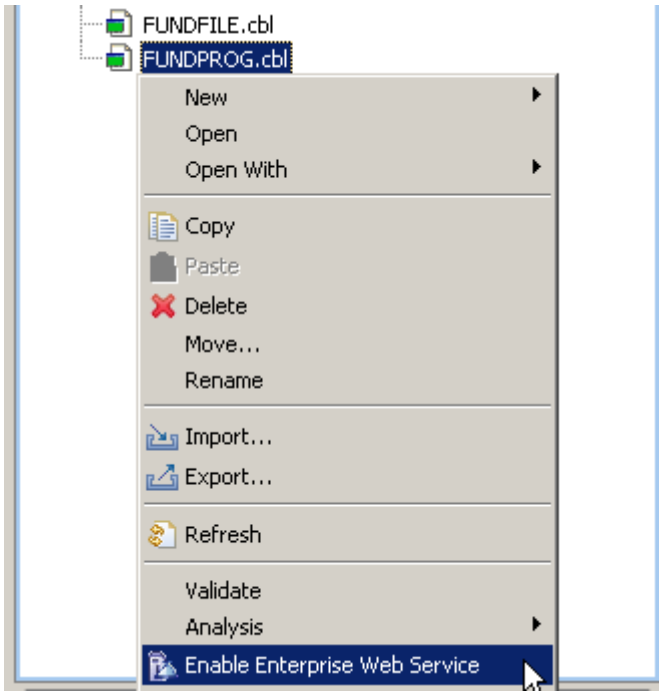
11.On the Select panel of the Import wizard, expand General and select File System. Click the Next button.



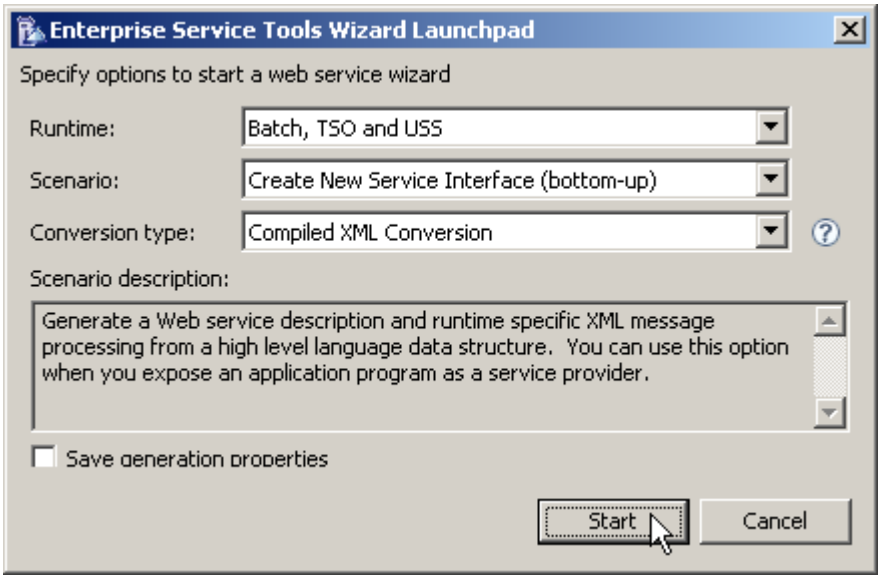
12.Click the Browse button on the top right and navigate to C:\WDz_Tutorials\Tutorial04\WSDLCreate and click OK.



13.Back on the File system page of the Import wizard, check WSDLCreate, ensure that the Into folder is WSDLCreate, ensure that Create selected folders only is selected and click the Finish button.

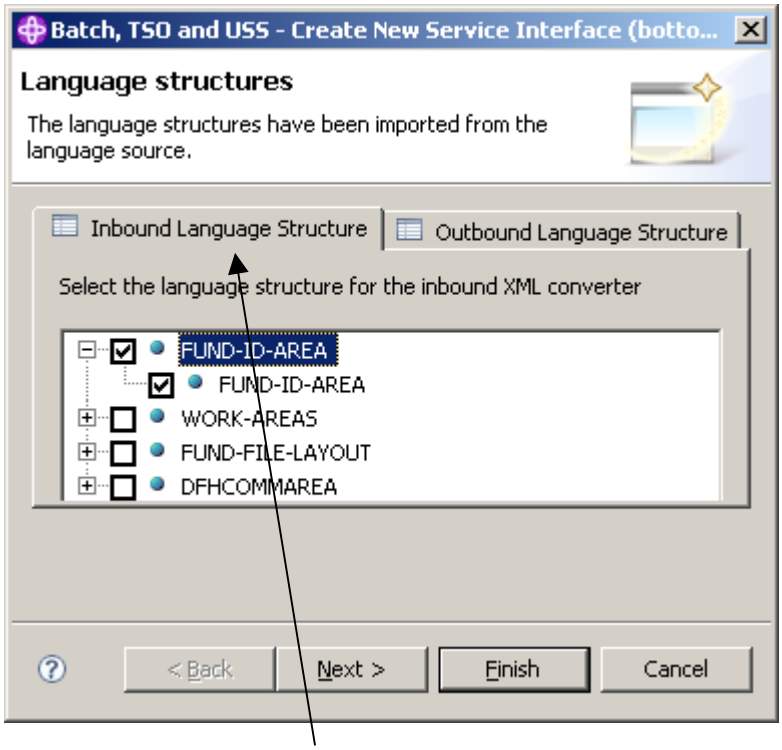


14.From the Navigator view, expand WSDLCreate, right-click on FUNDPROG.cbl and select Enable Enterprise Web Service.



15.On the Enterprise Service Tools Wizard Launchpad panel, set Batch, TSO and USS as Runtime, Create New Service Interface as Scenario and Compiled XML Conversion as Conversion Type.

Click Start.



16.On the Language structures panel, select the Inbound language structure tab, and check FUND-ID-AREA (if not already checked by default).

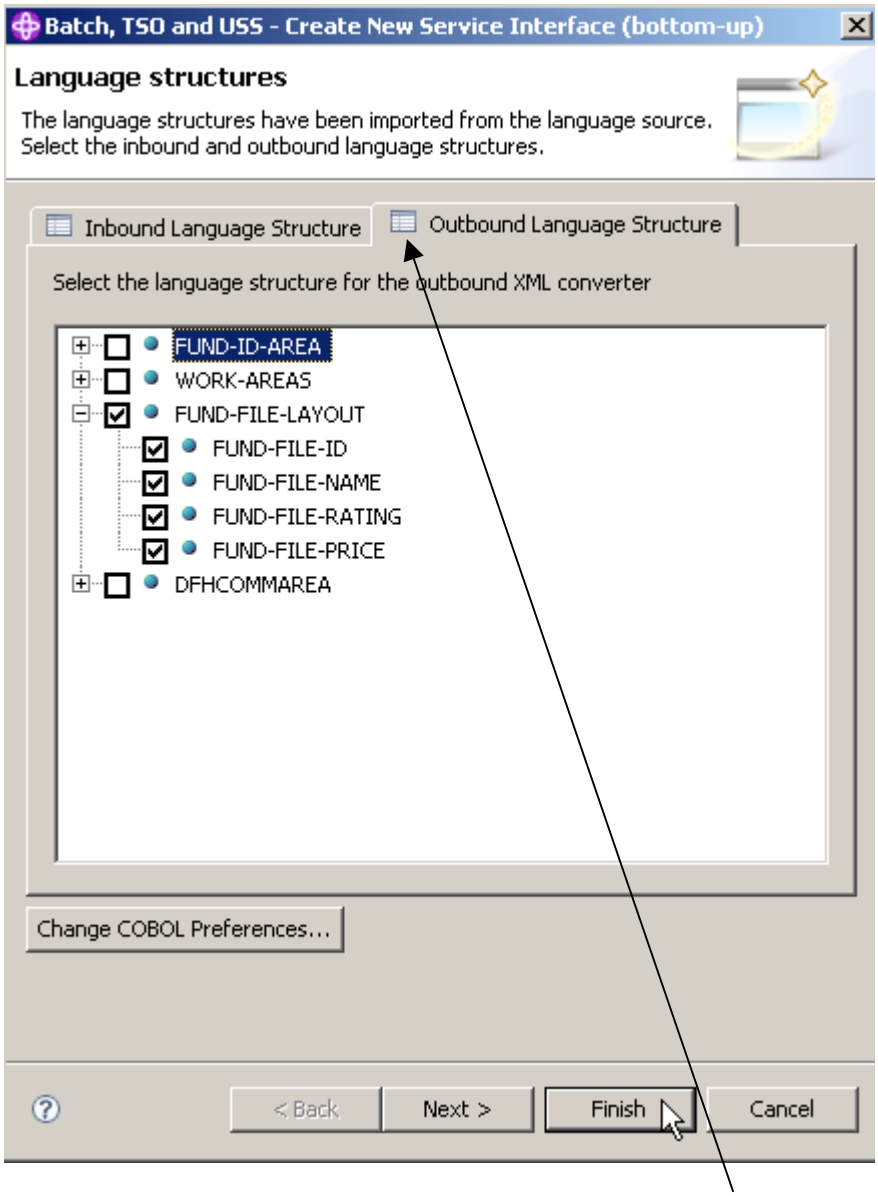
Do not yet klick on Next or Finished.

Note: The Cobol Program FUNDPROG.cbl contains as usual a Data Division. The code in the Data Division is shown on the next page.

FUND-ID-Area is the parameter in the WORKING-STORAGE-SECTION of the DATA-DIVISION of your Cobol Program FUNDPROG.cbl.

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  FUND-ID-AREA                    PIC X(8).
01  WORK-AREAS.
    05  RESP-FLD                    PIC S9(8) COMP.
    05  RESP-FLD-TWO                PIC S9(8) COMP.
    05  APP-ID                      PIC X(10) VALUE 'FUNDPROG:'.
    05  CSMT-MSG                    PIC X(121).
    05  DISPLAY-NUMBER              PIC 9999.

COPY FUNDFILE.
```

Note: Although we are going through a wizard that could generate WSDL, we are only using the wizard to generate XSDs (XML Schema Definitions) to describe a couple of data definitions in the COBOL program. FUND-ID-AREA describes fund id, which is the key to the fund file. This key would be the input message for a delete WSDL, or read WSDL operation. To use this as a message in WSDL, it must first be described in an XSD.

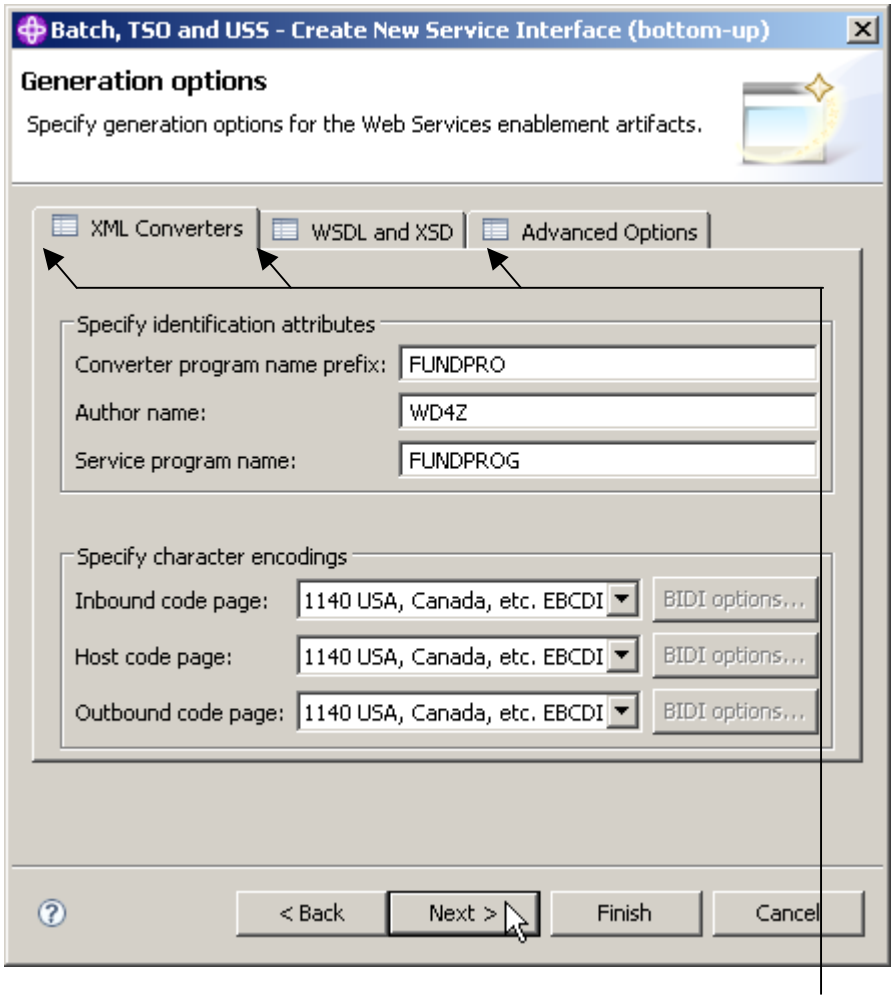


17.Still on the Language structures panel, switch to the Outbound language structure tab and check FUND-FILE-LAYOUT.

FUND-FILE-LAYOUT is the layout of the output message of our Web Service. FUNDFILE.cbl contains the following information:

```
01  FUND-FILE-LAYOUT.
    03  FUND-FILE-ID                PIC X(8).
    03  FUND-FILE-NAME              PIC X(50).
    03  FUND-FILE-RATING            PIC X.
    03  FUND-FILE-PRICE             PIC X(15).
```

Click Next (not Finish)



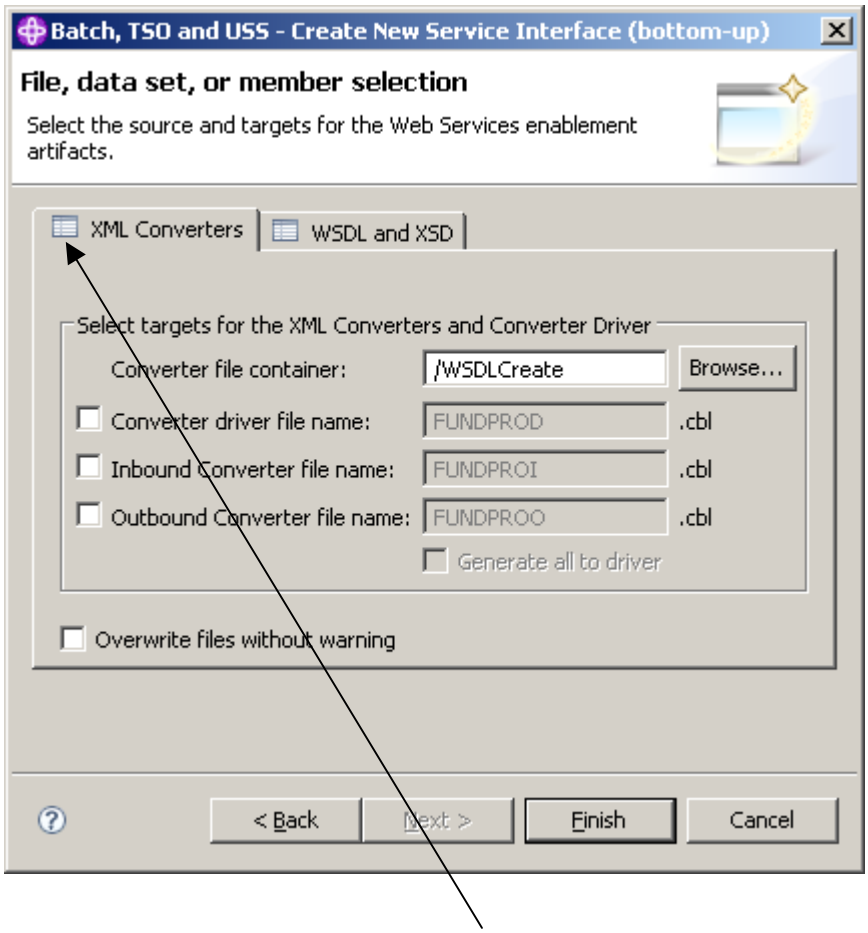
18.On the Generation options panel, feel free to have a look at the various options of the three tabs:

- **XML Converters**
- **WSDL and XSD**
- **Advanced Options**

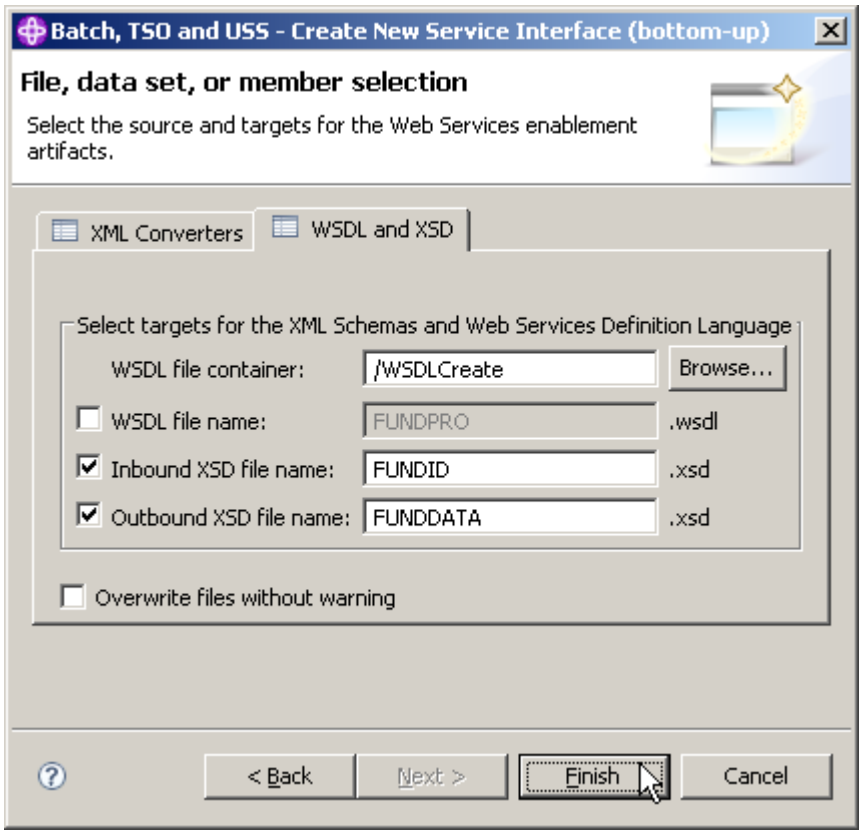
When you're finished, click Next.

Note: We will ask the wizard to generate XSDs for us, so we won't be generating a Converter program. (We will be generating a converter program in a later tutorial). From the WSDL and XSD Options tab, if we were generating WSDL, we could set the endpoint here. We could also set the Inbound and Outbound namespace names for the XSDs we will generate, but for this tutorial, the defaults are OK.

Klick next



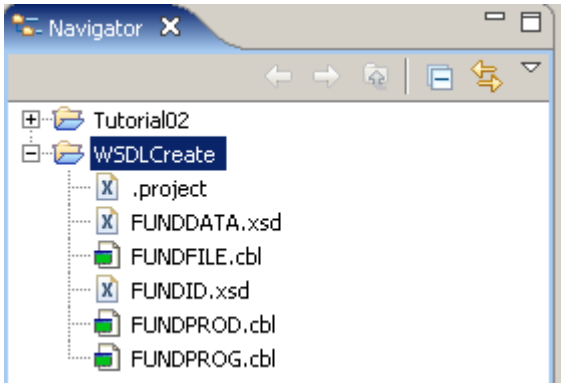
19.On the File, dataset, or member selection panel, click the XML Converters tab and uncheck all checkboxes. We are unchecking these options because we only want the wizard to generate XSDs for us.



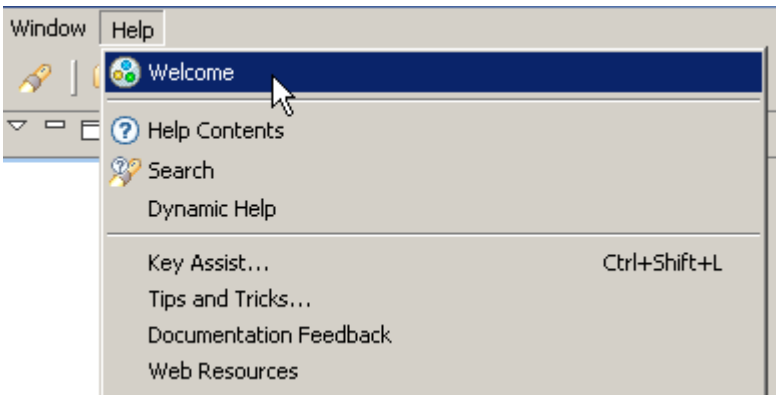
20.Still on the File, dataset, or member selection panel, switch to the WSDL and XSD tab and uncheck the box next to WSDL file name. Change

- **Inbound XSD file name to FUNDID and**
- **Outbound XSD file name FUNDDATA.**

Click the Finish button.

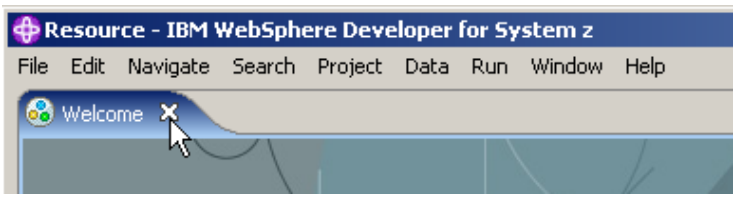


21.You should see a FUNDID.xsd and FUNDDATA.xsd file in your WSDLCreate project. Feel free to open each of the files and have a look at the content.



22.At this point, you should verify if the neccessary User Roles are enabled.
In the menu bar, click on Help ? Welcome.

23.Click on the Symbol vaguely looking like a human body in the right lower corner.
The Icon for Web Developer (advanced) must be activated. If not, activate it by clicking on it.



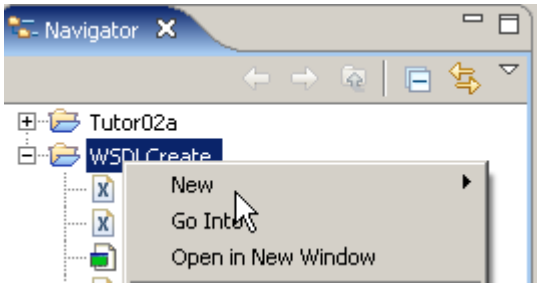
Close the Welcome window.

2.5 Create an Empty WSDL File

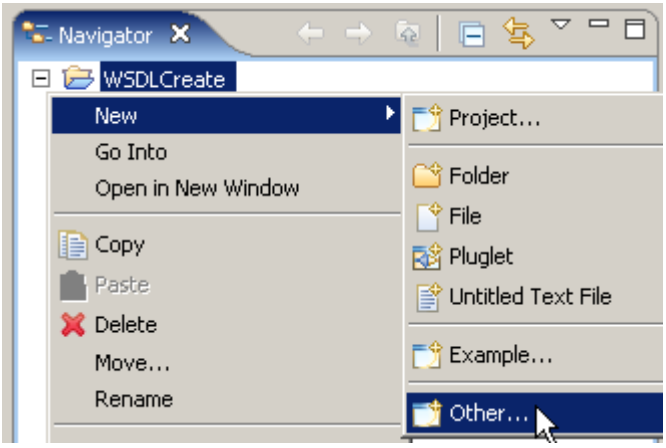
In the previous part of this tutorial, we generated some XSDs. One XSD represents the Fund file ID, and the other XSD represents the layout of the Fund file (all fields). We will use these XSDs as messages in the WSDL file that we will create.

In this part of the tutorial we will create an 'empty' WSDL file. Initially, the WSDL file will not have any operations, messages, bindings, or services. We will add those parts in a later part of this tutorial. This initial WSDL file will contain basic namespaces.

Namespaces are external locations that contain definitions for various parts of the WSDL we are defining – including the elements, attributes, and types. These definitions define various parameters including SOAP (protocol) and Envelop (data or payload).

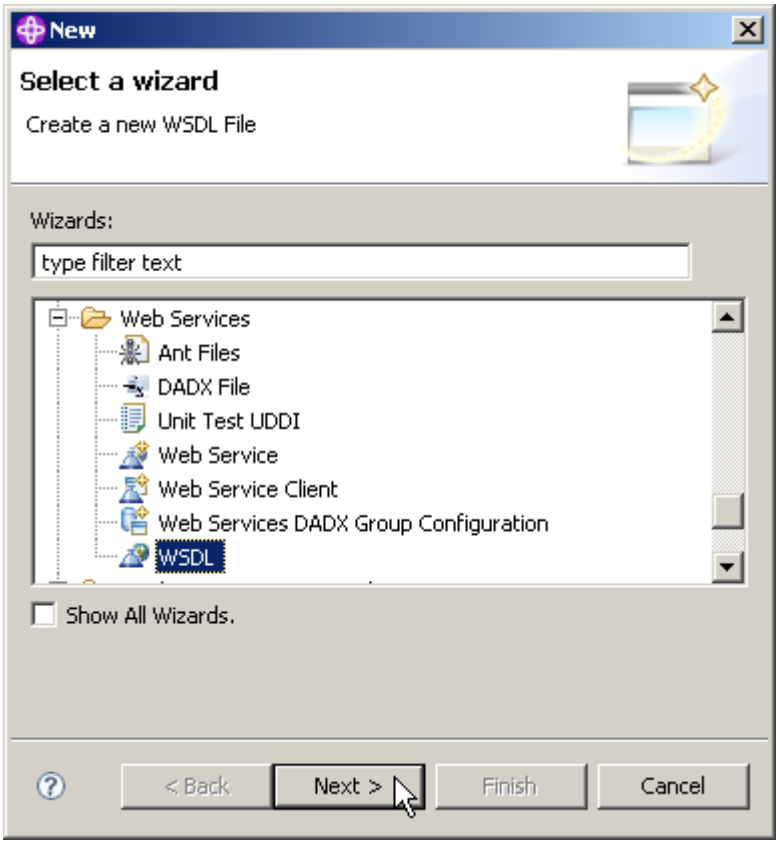


24.Right-click your WSDLCreate project.
From the context menu select New …

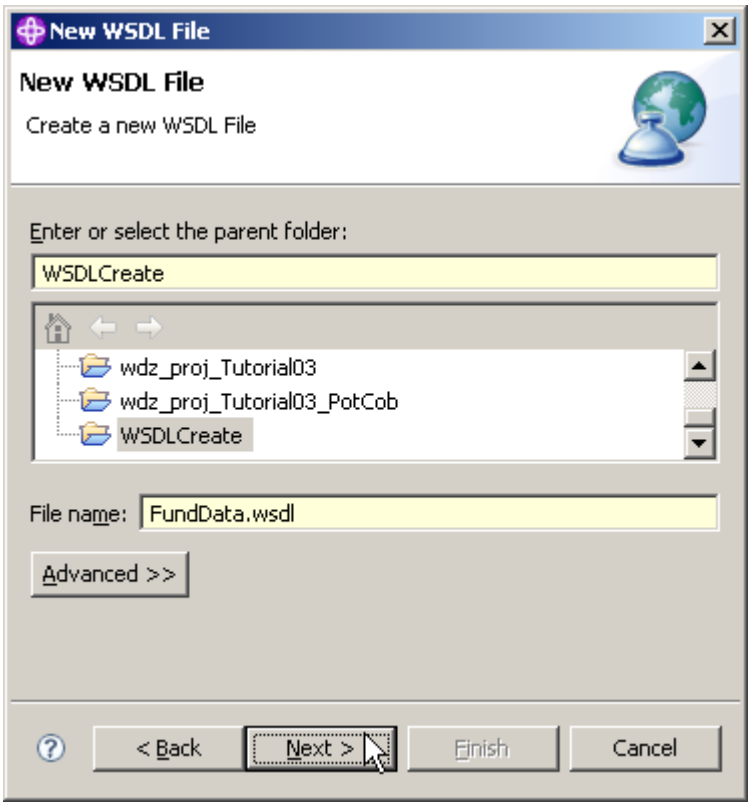


and Other...

scroll down

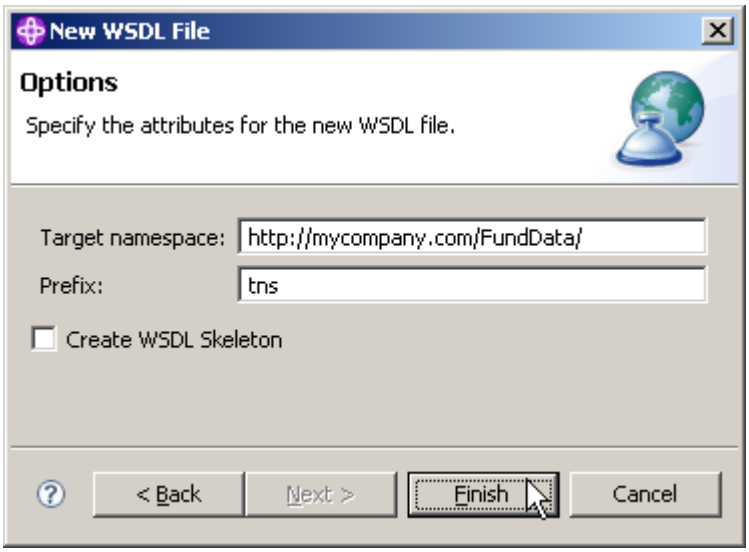


25.On the Select a wizard panel, select Web Services ? WSDL, then click the Next button.



26.On the New WSDL File panel, ensure that the parent folder is WSDLCreate, specify the File name as FundData.wsdl, and click Next.

Note: if you performed earlier WDz Tutorials, your screen may differ from the one above.



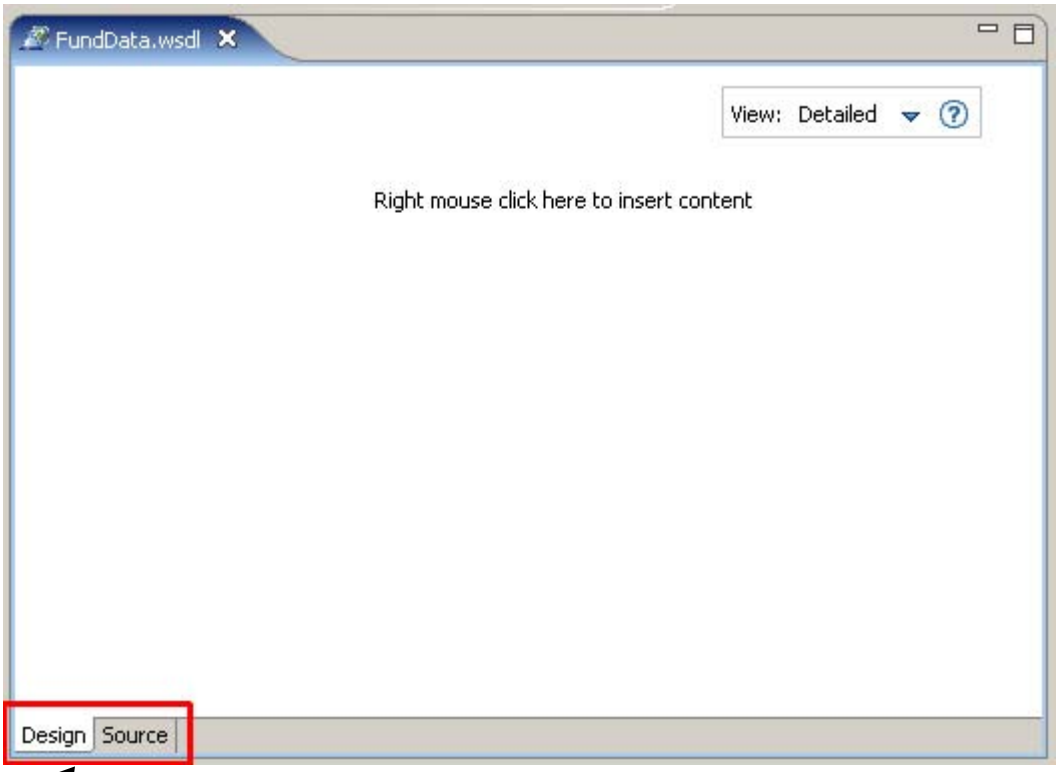
27.On the Options dialog, specify

- **http://mycompany.com/FundData/ for Target namespace,**
- **tns for Prefix,**
- **uncheck Create WSDL skeleton**

and click the Finish button.

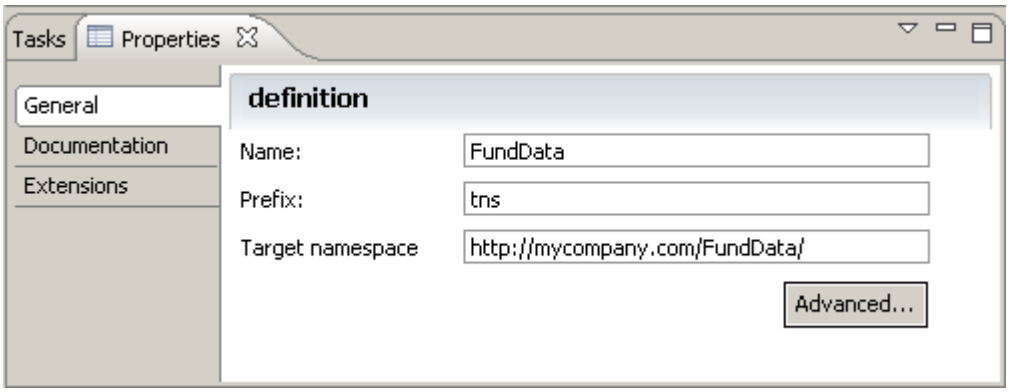
Note: We unchecked the Create WSDL Skeleton option because we want to add the operations, messages, bindings, etc, manually.

Click Finish



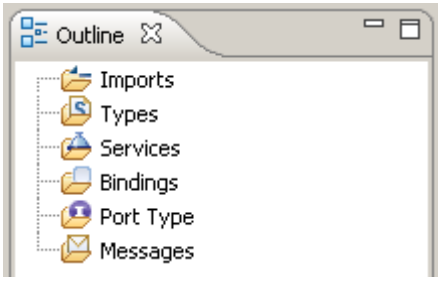
28.The FundData.wsdl file is now added in the Navigator view and opened with the WSDL-Editor

Note: This ist the WSDL-Editor window. On the bottom of the editor are two tabs. You are currently looking at the Design view, which shows a graphical representation of the WSDL file. There is also a Source view where you can view and edit the text that makes up the WSDL file.



29.It is important to note that the Properties view shown above is closely associated with the WSDL-Editor. The characteristics of items you click on in the WSDL editor will be displayed in the Properties view.

Note: If the properties view is not enabled, just open it from the menu bar (Window → Show view → Properties).



30.This is the Default Outline view situated below the Navigator view. You can use it to navigate through the different WSDL elements. It is also very useful.

2.6 Add the addFund Operation

In this part of the tutorial we will generate a PortType. We will add an operation called addFund.

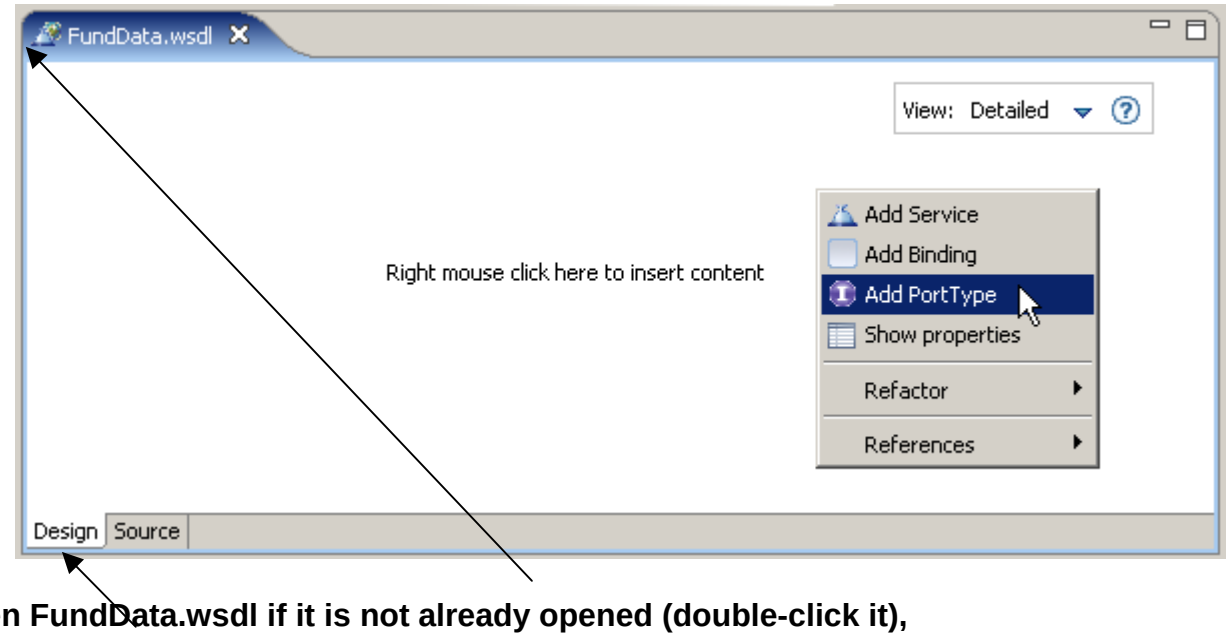
A PortType is sometimes called an "Interface". Actually, there are some similiarities with the function of an interface in RMI, Corba, or DCE. On the other hand, WSDL is also an interface, albeit much richer than an RMI or Corba Interface.

The addFund operation will take an input and output message that we will define later. The messages will point to the XSDs we created in the first part of this tutorial. The addFund operation will take all fields in the file as input, and will return the data that was added. Other design alternatives might be to return nothing, or return the ID of the Fund that was added.

It is common to have an add service return what it added. Depending on how you combine your Web Services, you will normally find that you want the output of one service to become input to the next service. By returning the added values from the add service, you can have those values flow into the next step of your business process.

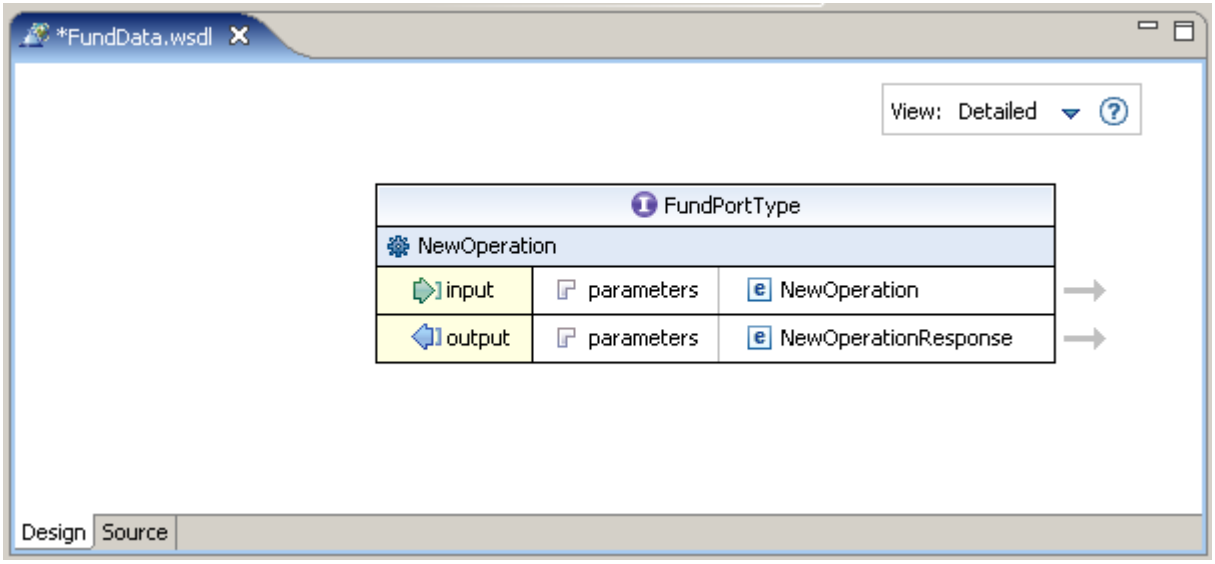
Our addFund operation is also capable of returning a fault if the operation is unsuccessful. The fault data is a string that indicates the reason why the fund could not be added.

We will first add a Port Type to the Port Types area. We will then add the addFund operation to the Port Type. The addFund operation will have input, output, and a fault. We will then associate the input, output, and fault with Messages.



31.Open FundData.wsdl if it is not already opened (double-click it), and switch to the Design view.

32.Right-click into the empty file and from the context menu, select Add PortType.

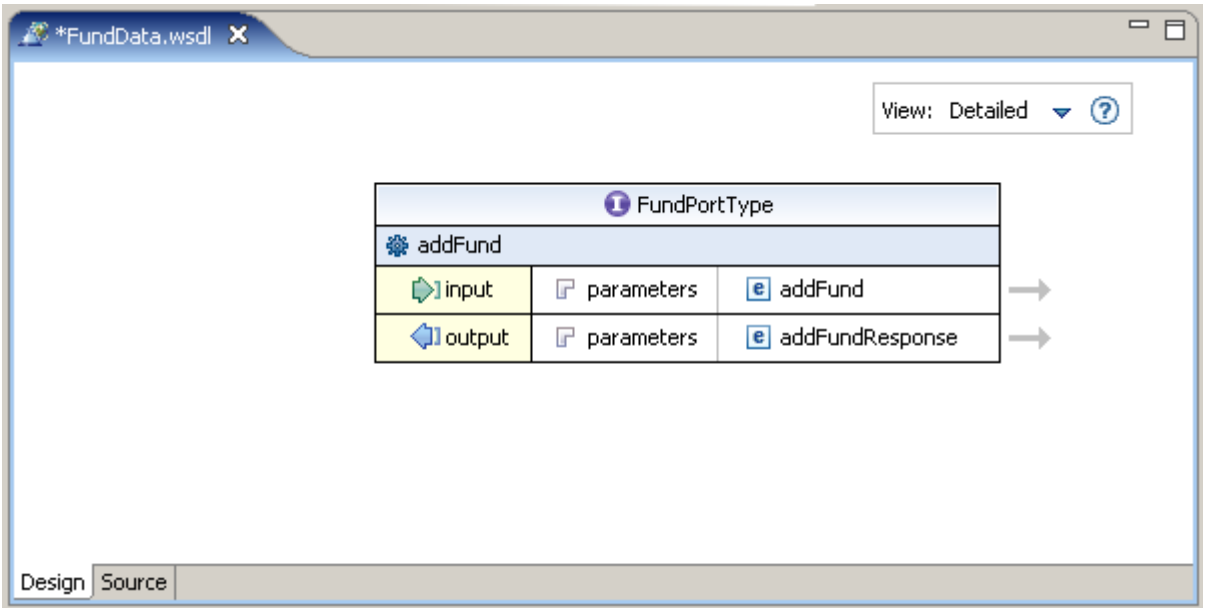


33.A new element appears, the PortType. Change the name from NewPortType to FundPortType.

Note: There are several ways to change the name of your freshly added port type. You can just click into the name to make it editable, or you can make the changes in the above mentioned Properties view, just to name two possibilities.

Klick auf NewOperation und dann auf den Pfeil, dann kann man den Namen im Properties Window ändern. direcktes Reinklicken funktioniert nicht.

As you can see, the added PortType has a NewOperation including input and output indicators.

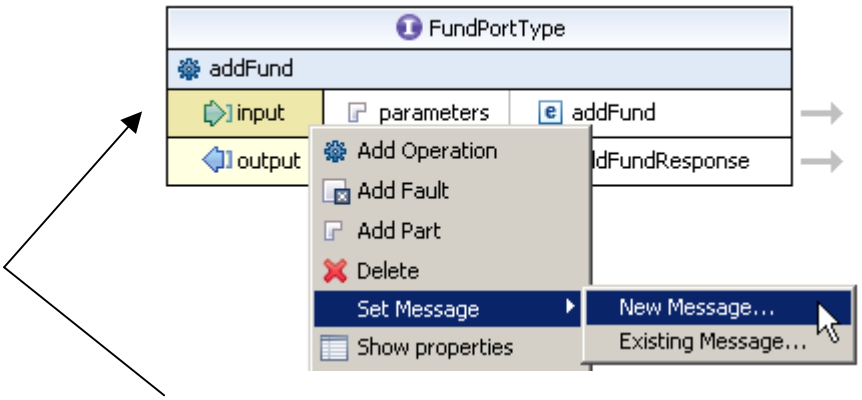


34.Rename NewOperation to addFund the same way you did with the PortType.

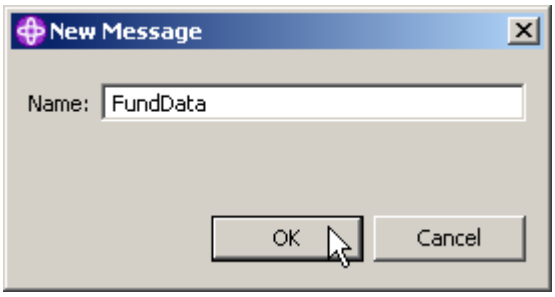
35.Save the changes with CTRL + S.

2.7 Add Messages

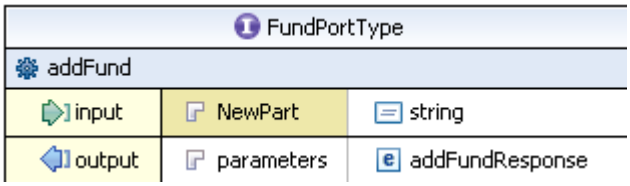
In this part of the tutorial, you will add messages to your WSDL. Messages are the application data that is passed to and from operations. The input and the output of our addFund operation will be all the fields in the Fund file. You created the message layout (XSDs) in chapter 2.4 of this exercise. The XSD that describes the data in the Fund file is in a file called FUNDDATA.xsd.



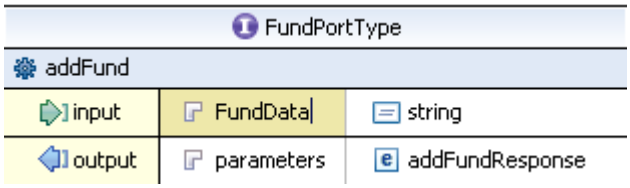
36.To add a message, right-click the input element and select Set Message → New Message...



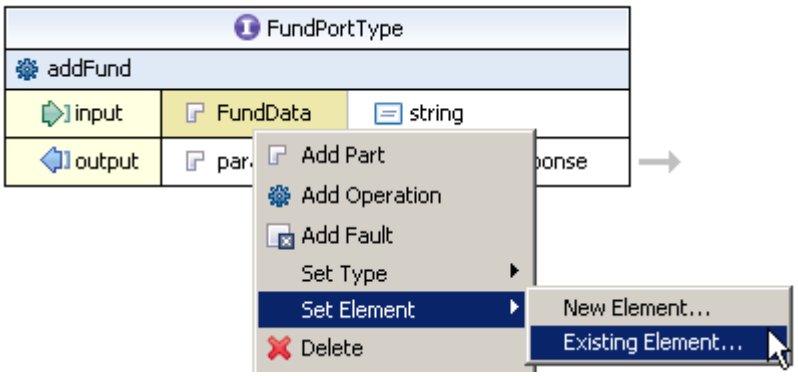
37.In the New Message dialog, specify the Name as FundData and click the OK button.



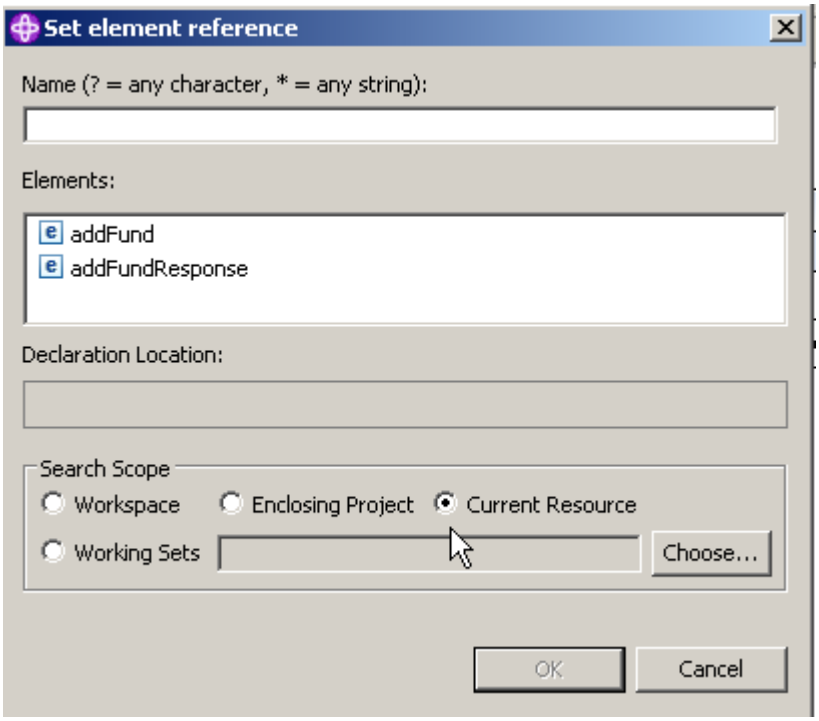
38.Click on the NewPart element next to the input element to make it editable



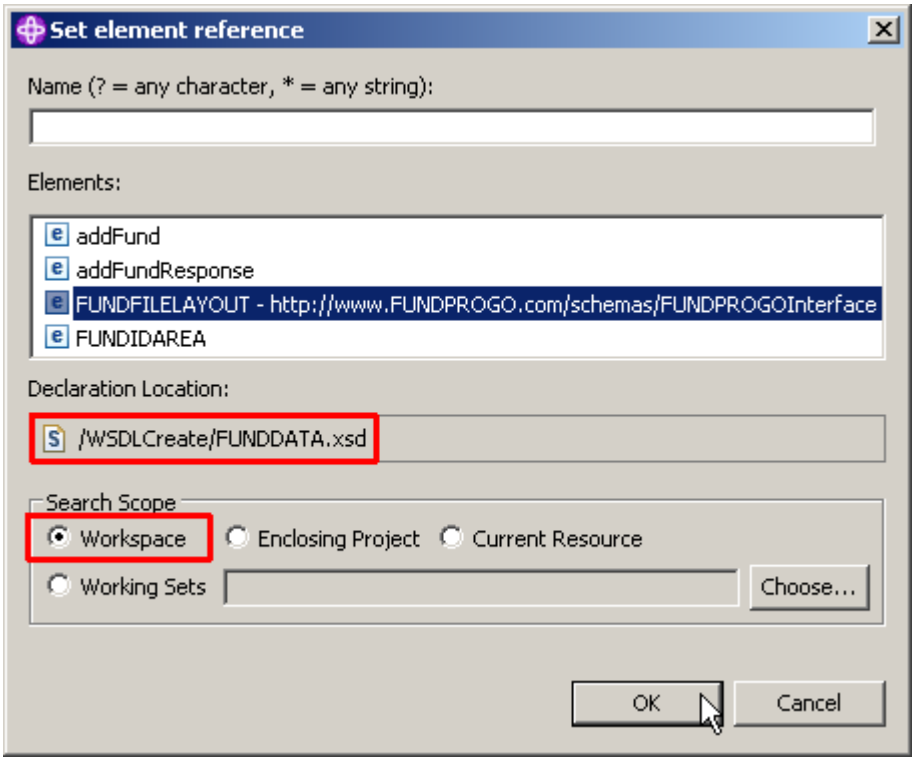
and rename it to FundData.



Now right-click on the FundData element and from the context menu, select Set Element → Existing Element...



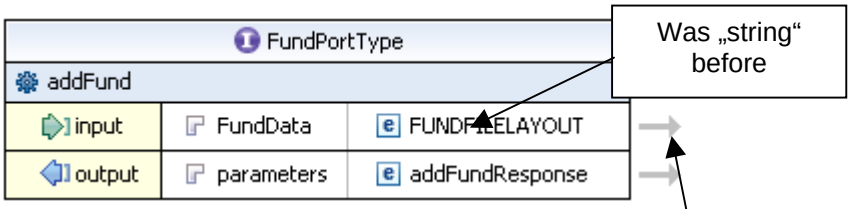
39.On the Set element reference panel, first of all, change the Search Scope from Current Resource to Workspace.



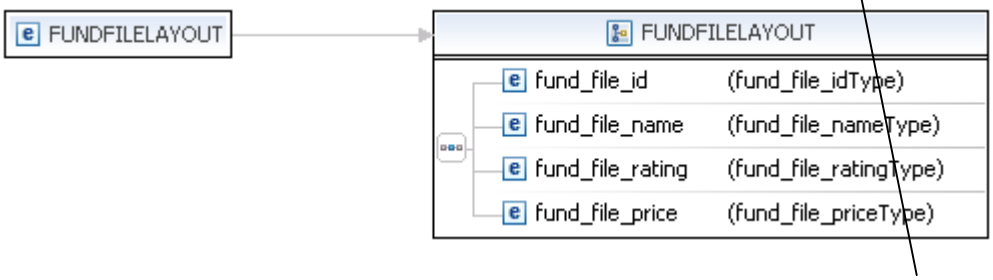
Then you will find two new elements, FUNDFILELAYOUT and FUNDIDAREA.

Choose FUNDFILELAYOUT (the text will expand) and ensure, that the Declaration Location is the /WSDLCreate/FUNDDATA.xsd as shown above.

Finally, click the OK button.

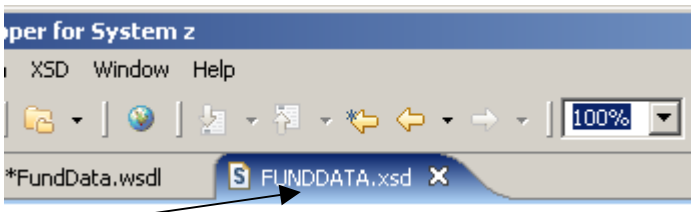


40.Your FundPortType should look like this:

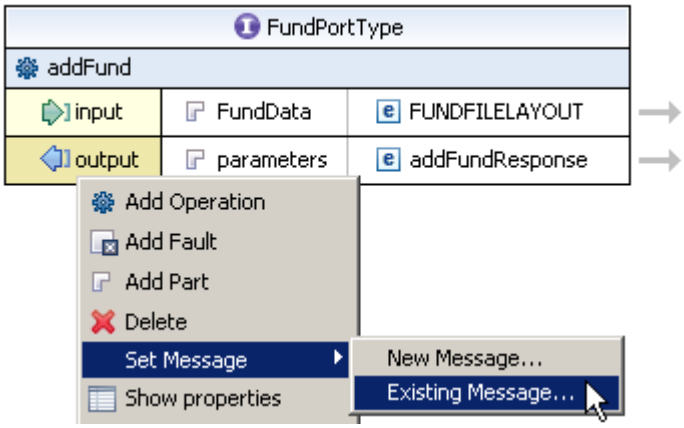


41.Let's have a look at the FUNDFILELAYOUT element. If you click on the arrow next to it, the FUNDDATA.xsd will open.

As you can see below, FUNDFILELAYOUT is a ComplexType consisting of four other elements.

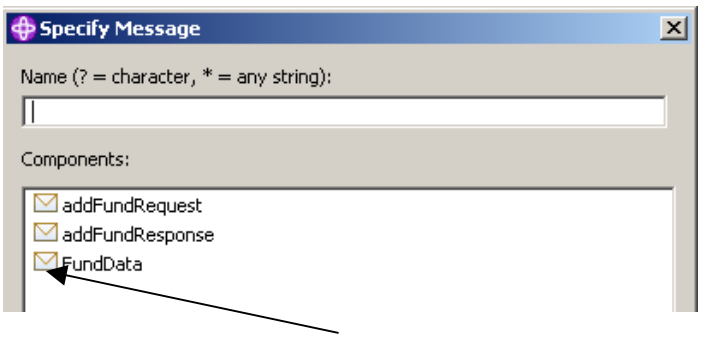


When you're finished, close the FUNDDATA.xsd.

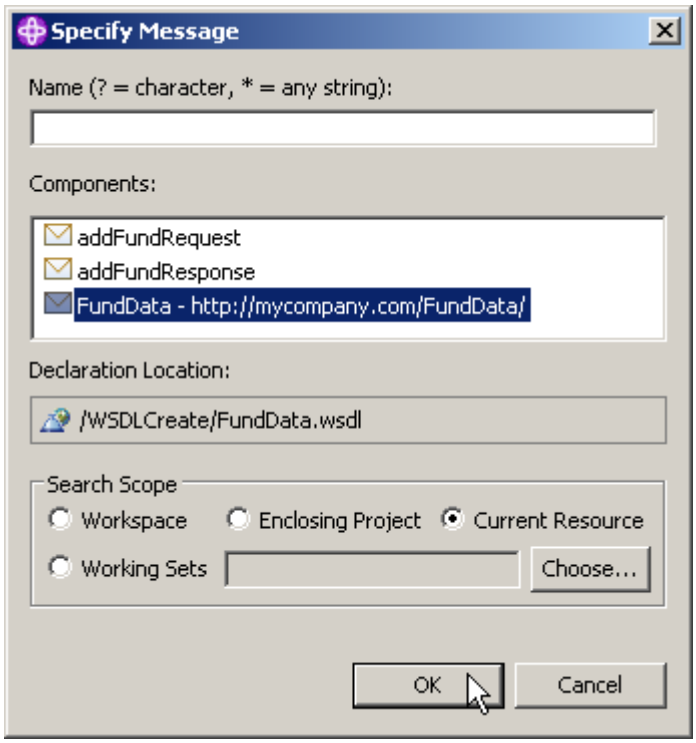


42.Now we have to configure the layout of the outgoing messages. As we use the same layout for incoming and outgoing messages, this is an easy task because we already configured everything we need.

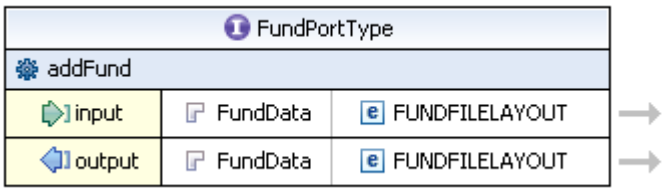
Right-click the output element and select Set Message → Existing Message...



43.In the Specify Message dialog, select the earlier created FundData Message and ….



and click OK.

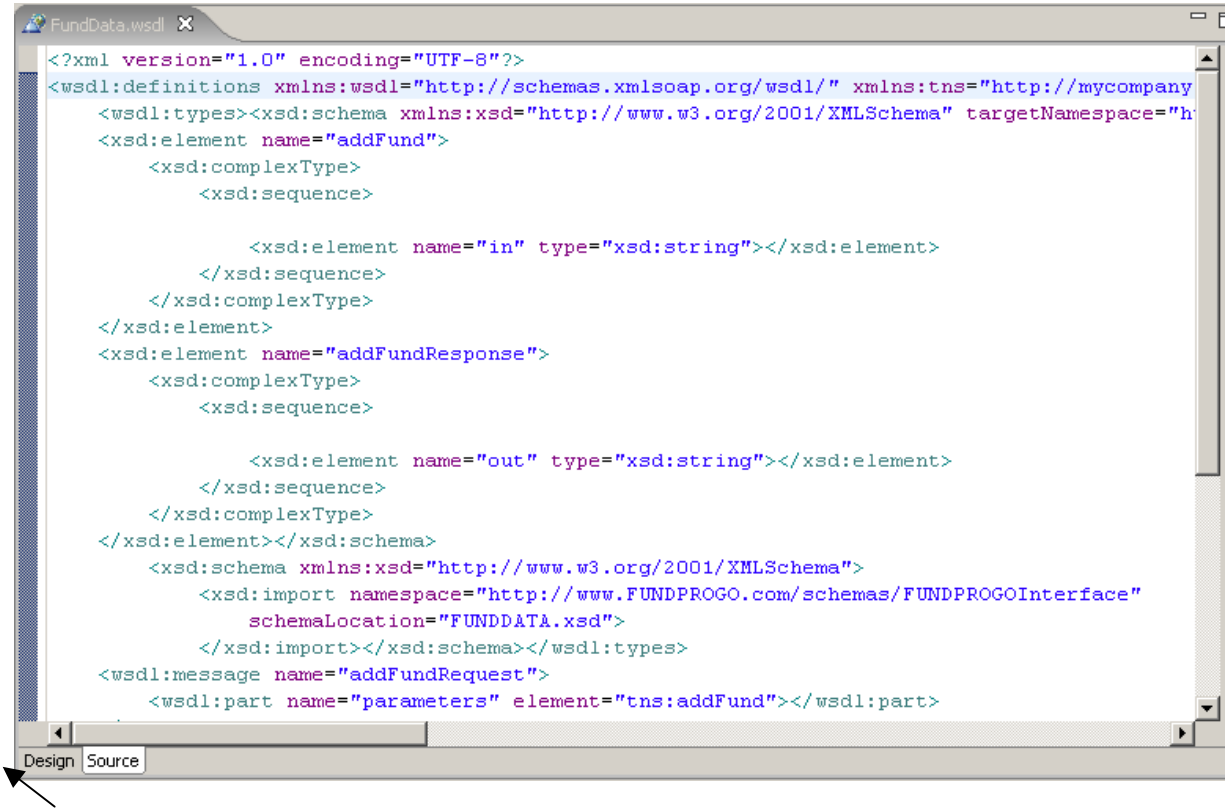


44.Your FundPortType should now look like that:

Note: You could have set the layout for the output message as well with the help of the Properties and Outline view. At that point, you will note, that you can specify a name for both, the input and the output message. If you feel it is important to you, you can do it now. You won't see a change in the Design view though (Only in the Source view!).

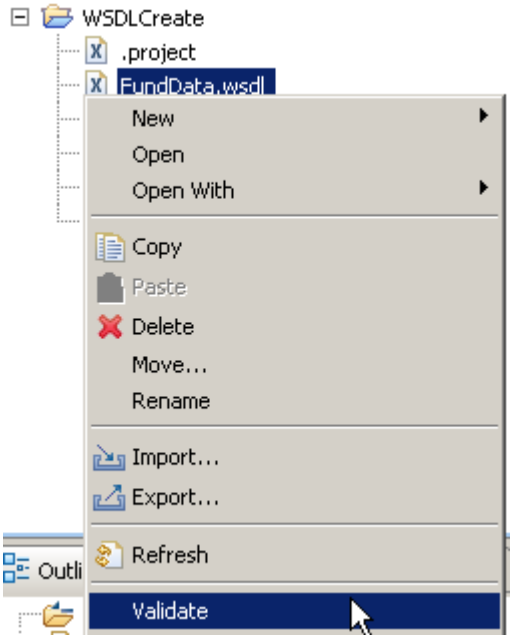
45.We finished configuring our message layout. Both, input and output messages use the same layout.

Press CTRL + S to save your work in the FundData.wsdl editor (the * next to FundData.wsdl should disappear).



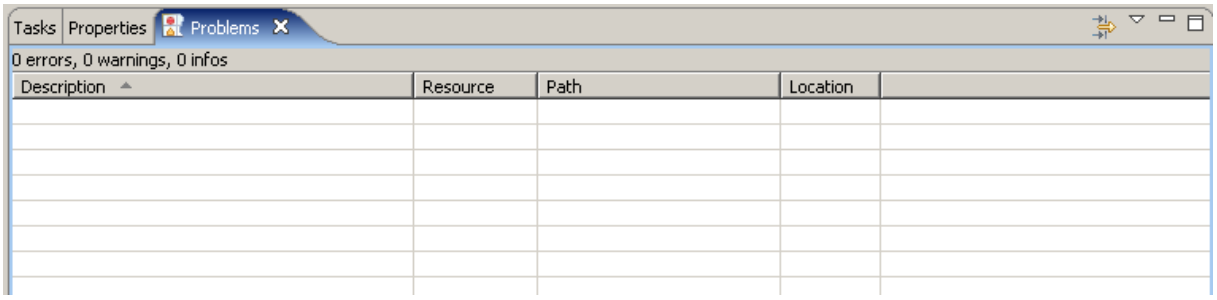
Click on Source to have a view of the generated XML code. Remember, a WSDL file consists of XML code.

If you would have to code this manually, this would be a very error prone process. One of XML's strong characteristics is, that using modern tools, you just about never have to handcode it.



46.As a final step, you should validate your WSDL file.
From the Navigator view, right-click FundData.wsdl and select Validate from the context menu.

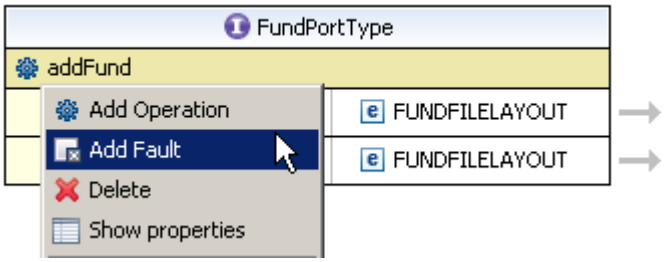
If you have a valid WSDL file, you won't get a popup or similar. In addition, there should be no errors in the Problems view. If the Problems view is not enabled, click Window → Show View → Problems on the menu bar to enable it.



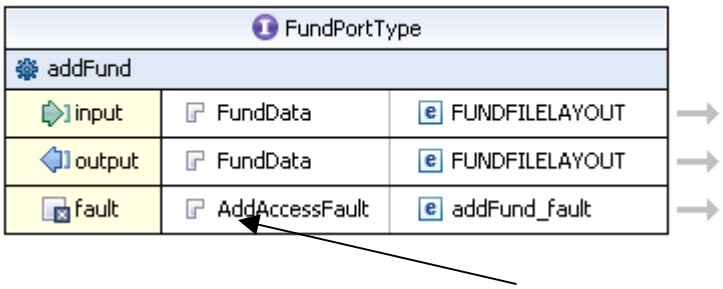
Note: You MUST have an aktive connection to the internet to validate the file because a WS-I Test Assertion Document (TAD) document is mandatory! Otherwise you will get an error like: „The WS-I Test Assertion Document (TAD)document was either not found or could not be processed“.
So check if you need to configure the use of a proxy,
for instance In Window → Preferences → Internet → Proxy Settings.

2.8 Add a fault element to the addFund Operation

If a fund cannot be added, the operation will throw a fault. In this section we will add a fault to the addFund operation.



47.Right klick on Design again to leave the Source representation. In the opened FundData.wsdl file, right-click the addFund operation, and select Add Fault.



48.You may change the name ot the fault part element from parameters to AddAccessFault.

The fault message was added with the part AddAccessFault and the element addFund_fault. The part is of type string. This a typical return type for most faults.

49.Press CTRL + S to save your WSDL file.

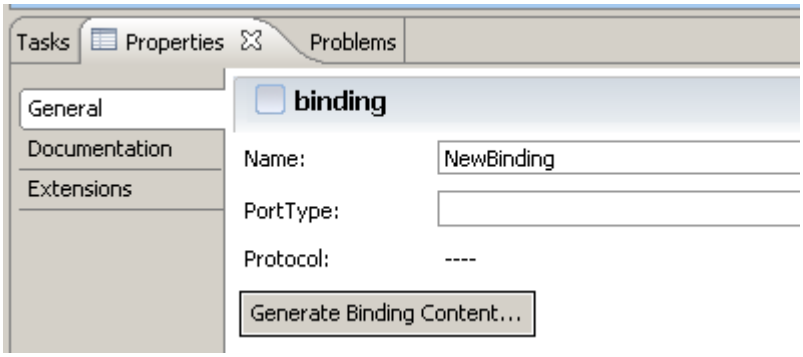
50.You should validate you edited WSDL-file again like you did above (see sectio 46).
If you get no errors, proceed to the next chapter.

2.9 Add Binding Information to the WSDL File

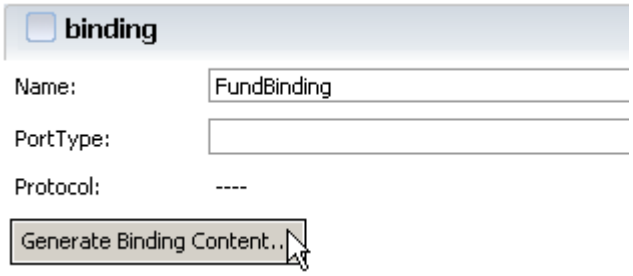
We have added a Port Type, an operation, and messages. These are generic and could apply to any service (i.e. these are transport independent). In this section, we will tell the WSDL editor that we are describing a Web Service. A Web Service is SOAP over HTTP. This means that we will be adding a SOAP Binding. In a later Part (Part 9) of this Tutorial we will add Service information that will indicate where the Web Service is located.

51.Right-click on an empty spot in your opened WSDL-file and select Add Binding from the context menu.

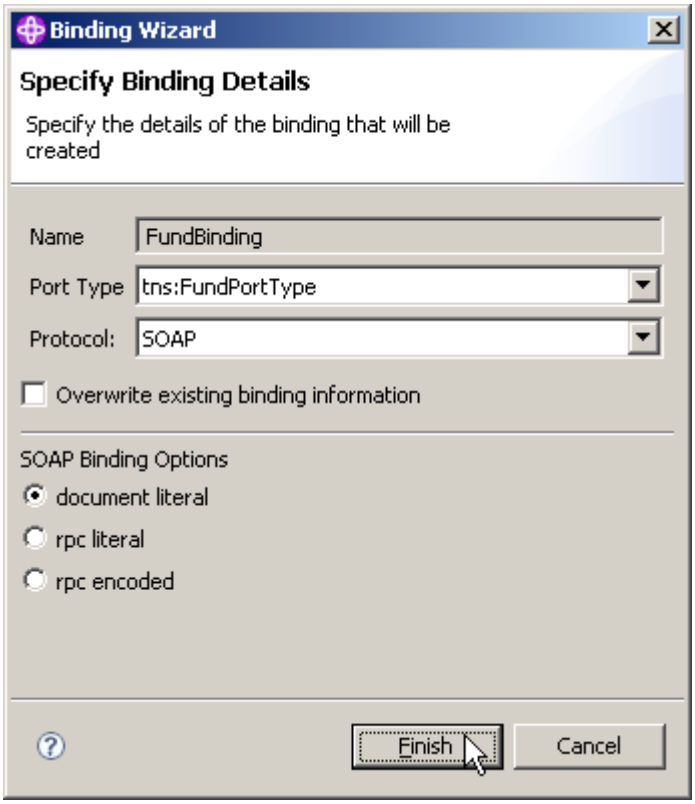
52.You will now see a Binding Element like that: 



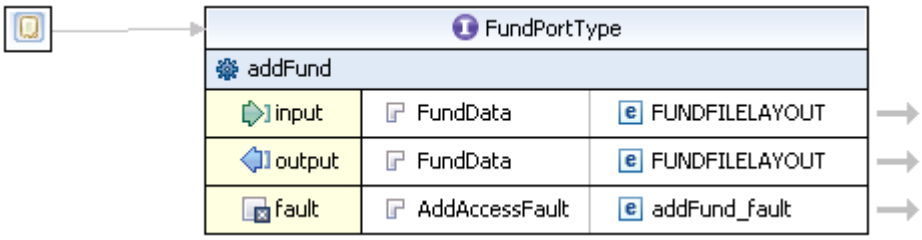
53.Click the element. In the Properties view, change the name from NewBinding



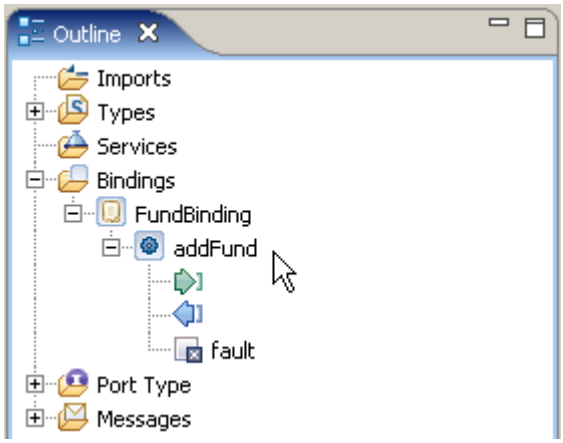
to FundBinding. Then, click the Generate Binding Content.button.



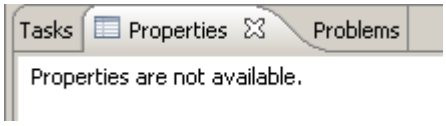
54.In the Specify Binding Details dialog, change the Port Type to tns:FundPortType and the Protocol to SOAP. Leave the defaults for the SOAP Binding Options.
Click Finish.



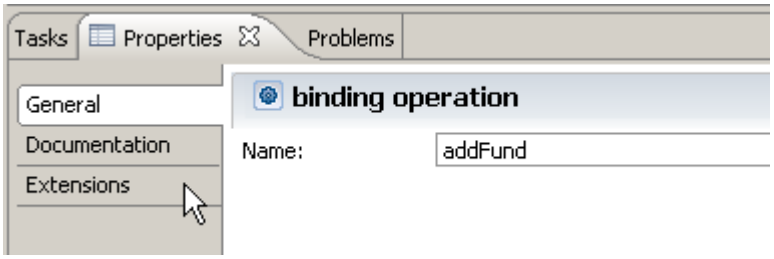
55.Your FundData.wsdl should now look similar as shown above.



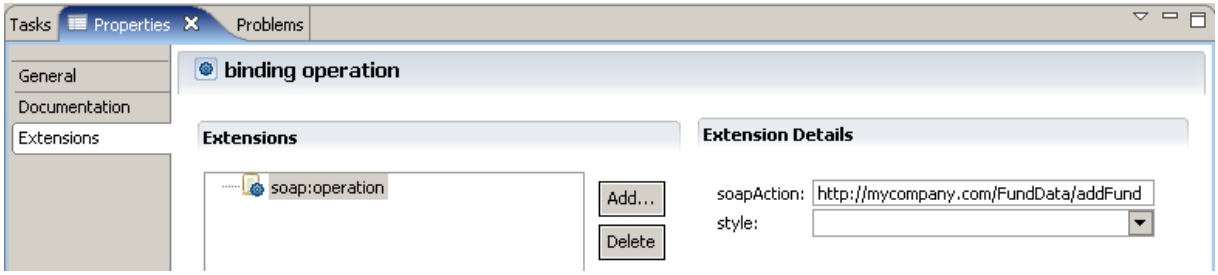
56.Fully expand the Bindings tree structure in the Outline view. You will note, that your newly created FundBinding has a binding operation named addFund. This indicates, Port Type and Binding are connected now.



At this point, the Properties View is empty



In the Outline view, click on addFund and have a look at the Properties view. Switch to Extension in the menu on the left and …



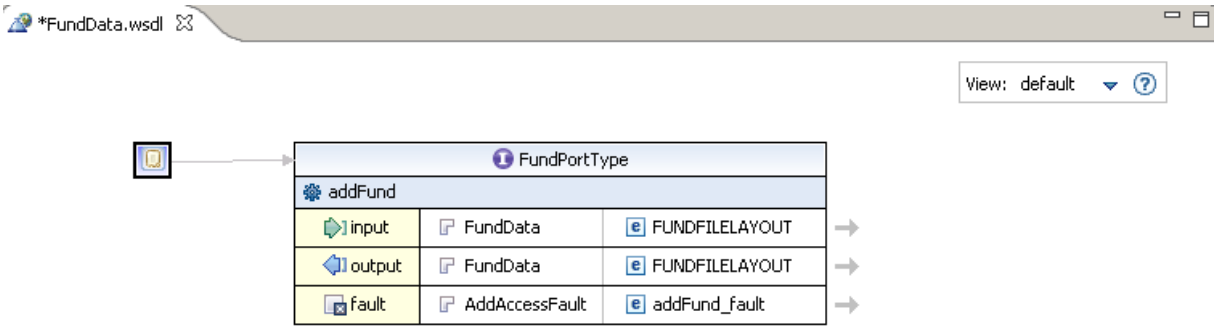
ensure that under Extension Details, soapAction is http://mycompany.com/FundData/addFund and

style is document.

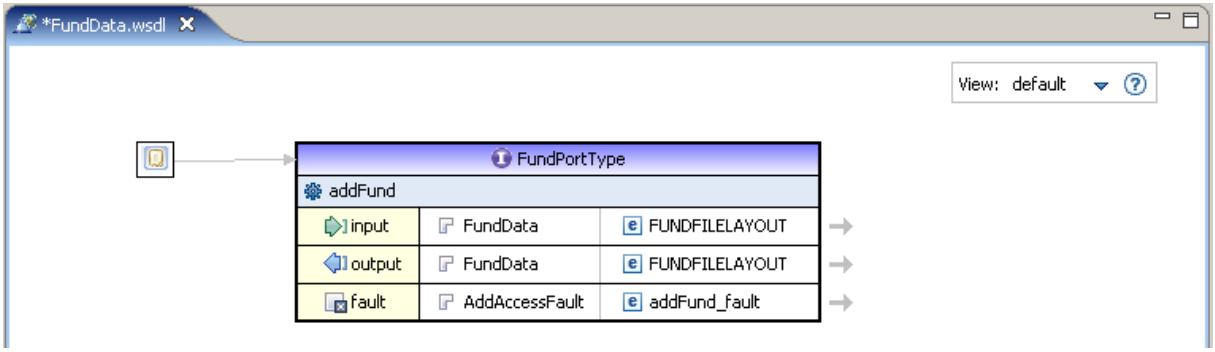
57.In the FundData.wsdl and the Properties View press CTRL + S to save your changes. Now validate your file again (section 46). If there are no errors, proceed to the next chapter.

2.10 Add Service Information

We have specified what business operation is available and the information that is to be sent and received by the operation. We specified that the information would be sent using SOAP, and now we need to specify where the Web Service resides.



58In your opened FundData.wsdl-file,

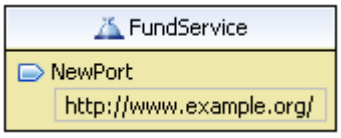


highlight FundPortType, and

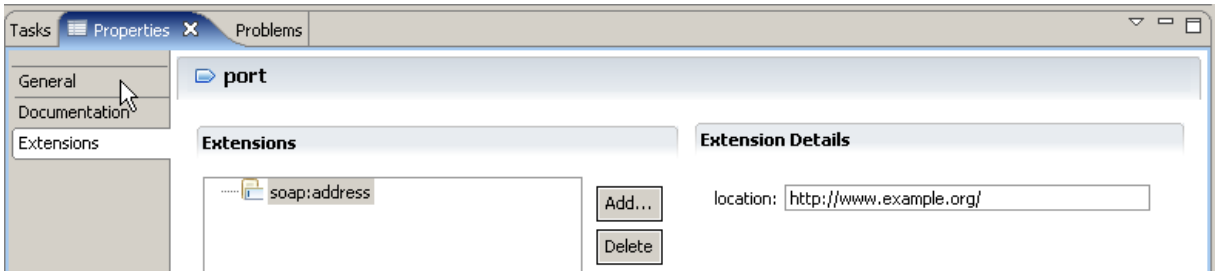
and select Add Service from the context menu.

59.A new Service Element will appear.

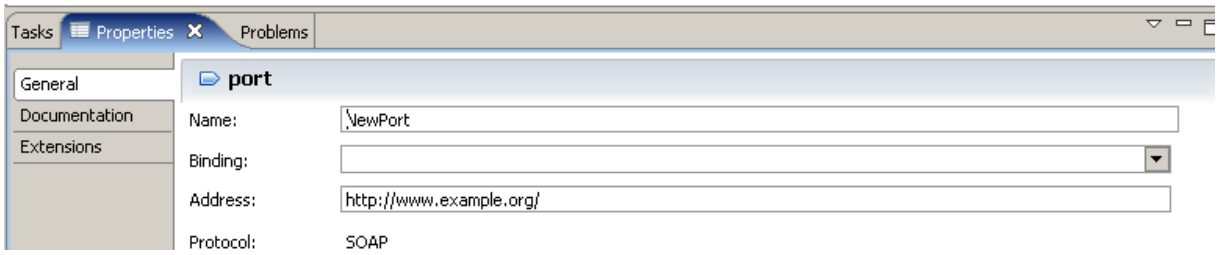
Click on the name of the new Element and change it from NewService to FundService.



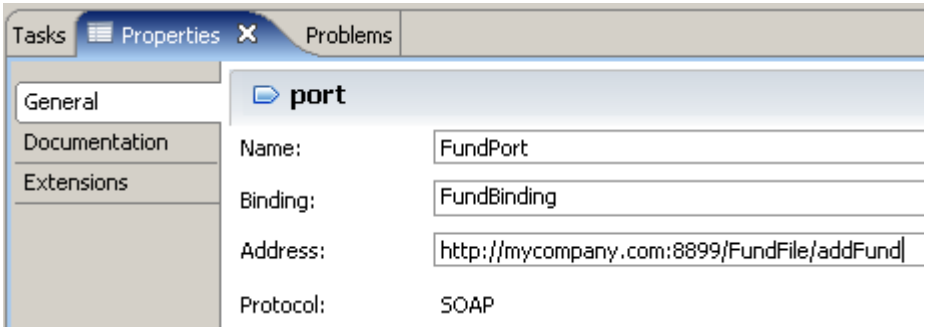
60.Click on the NewPort element. The colour changes.



In the Properties view click on General

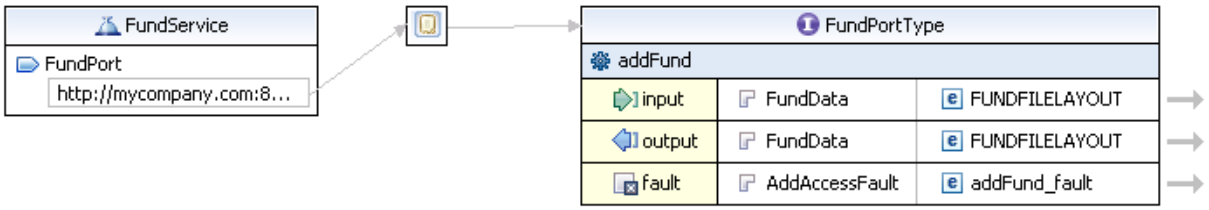


Now



specify

- **as Name: FundPort**
- **as Binding: FundBinding**
- **as Address: http://mycompany.com:8899/FundFile/addFund**



Your FundData.wsdl should now look like this:

Note: The Address is also known as the Endpoint of a Web Service.

61.Press CTRL + S to save the file and validate it a last time (see sectio 46).

62.If there are no errors, close the WSDL-file.

Now you have a fully functional WSDL document that describes a business operation (addFund), input and output messages and their layout (both point to the FUNDDATA.xsd), a fault (the service will return a fault if something goes wrong), a binding (SOAP over HTTP), and a service (which specifies the location of the endpoint).

It is a valid WSDL document, and it adheres to the WS-I basic profile 1.1. WS-I is an open industry organization chartered to provide Web Services interoperability across platforms, operating systems, and programming languages.

Congratulation, you successfully completed the Web Services Tutorial01 !

2.11 Create more WSDL Files (Optional)

This is an optional part that will give you more practice with WSDL and the WDz WSDL editor. The backend program that will be invoked supports Fund Create, Read, Update, and Delete processing. We have just created some WSDL that defines an addFund operation that will invoke Fund Create on the backend program. In this optional step, you can create a WSDL file for each of the getFund, deleteFund, and updateFund operations.

You probably also noticed that in Part 3 we made two XSDs: one that described the fund file layout (FUNDDATA.xsd), and one that described the fund id (FUNDID.xsd). For the addFund operation, both the input and the output of the operation was FUNDDATA.xsd. This means that so far, you haven't used the FUNDID.xsd. If you do this part of the tutorial, you will use the FUNDID.xsd for the input message for the getFund operation, and for the deleteFund operation.

The directions in this part of the lab are not as detailed as the directions in the early part of this lab so you may need to look back at a few earlier steps if the directions are not explicit enough.

Add WSDL files (one each) for the following operations. These operations use the FUNDID.xsd and FUNDDATA.xsd files as indicated in the table below.

Operation	input	output	fault
getFund	FundId	FundData	GetAccessFault
deleteFund	FundId	FundData	DeleteAccessFault
updateFund	FundData	FundData	UpdateAccessFault

Be sure to save and validate your WSDL files.

Resources

Webtut01.zip contains resources as a zip Archiv. Download at

www.cedix.de/Vorles/Band3/Resources/webtut01.zip

RDz Web Services Tutorial 02
Testing Web Services

Content

1. Tutorial Overview

The purpose of this exercise is to look at ways of testing Web Services. Some potential ways of testing a Web Service are

- **write a CICS program that invokes the Web Service,**
- **generate a Java program using Rational Application Developer (RAD) or WebSphere Developer for zSeries (WDz),**
- **use the debugger that comes with WDz,**
- **use the supplied Web Services Explorer that comes with RAD and WDz,**
- **use the TCP/IP Monitor that comes with RAD and WDz,**

or a combination of these.

In this tutorial we will use the Web Services Explorer and the TCP/IP Monitor. Although it is outside the scope of this tutorial, with RAD or WDz it is easy to generate a Java program to test the Web Service.

One of the major items in WSDL is the location (end-point) of the Web Service. Since WSDL is often generated by tools, the end-point location is not always correct, so you will want to verify that the end-point information is accurate. In this tutorial we will check the end-point information and will update this information to indicate the location of the Web Service that we will test.

In a later tutorial we will explore writing CICS Web Service requester programs and the use of the EXEC CICS INVOKE WEB SERVICE command.

For this tutorial, we will be invoking the supplied CICS Web Services example. We will just invoke the Web Service and see that a valid response is returned.

41.1 Scenario

In our scenario, you are a programmer at a mutual fund company. You have been given a WSDL file from a new Web Service that has been implemented on CICS TS V3.1 and you have been asked to test the service.

You will analyze the WSDL, verify the end-point information, and test the Web Service.

51.2 Tutorial Requirements

Please note that there are often several ways to perform functions in WDz. This tutorial will present one of the ways. If you are familiar with WDz, you will notice that some of the statements are general, and not necessarily true for every situation. The main intent of this tutorial is to reinforce the discussions on SOAP and WSDL.

The following are other assumptions made in this tutorial.

Assumption 1: A VMware image was supplied that contains WDz V7.0 and the files that are needed for this tutorial.
Assumption 2: The VMware image starts a Windows XP OS using "unilp" as login and password.
Assumption 3: You have some familiarity with RAD or WDz.
Assumption 4: A preconfigured CICS with the Web Services example is available and you have connectivity to the z/OS system that is running the CICS.

1.3 Section Overview

1. Tutorial Overview (this section)

2. Prerequisites

Before Starting the lab some prerequisite steps have to be performed to ensure your environment is configured properly.

3. Locate the Web Service Description (WSDL) for the Web Service
There are several ways to get the WSDL for a Web Service. In this section of the lab we will discuss the various options for obtaining WSDL and when you might use each option.

4. Analyze the WSDL for the Web Service
WSDL wasn't designed to be 'human-readable', but some understanding of WSDL will aid your understanding of the Web Service, and should anything go wrong, will greatly help in debugging.

5. Change the WSDL End-point
The endpoint of a Web Service is the actual location of the Web Service. Although the WSDL that you have been supplied properly defines the Web Service interface, the endpoint specified in the WSDL is wrong. In this part of the tutorial, you will change the endpoint of the Web Service in the WSDL that is supplied.

6. Test the Web Service
In this part of the tutorial you use the Web Services Explorer, a part of Rational Application Developer and WebSphere Developer for zSeries, to invoke the Web Services and to see if it returns results as expected.

7. Look at Web Service flows using the TCP/IP Monitor Server
When developing or testing Web Services, it is sometimes beneficial to take a look at the Web Service flows (request/response) as they appear 'on-the-wire'. In this part of the tutorial you will run a program that intercepts the flows between the service requester and the service provider.

2. Prerequisites

The WebSphere Developer for zSeries Workbench is initialized based on roles. This means that depending on your enabled roles certain features are enabled and others are disabled. You can specify your role on the Welcome screen which is displayed at the first initialization of your workspace.

1.If the Welcome screen is not displayed in your workspace retrieve it by clicking Help ? Welcome.

2.Your currently enabled roles are displayed in the lower right corner (note, that your currently enabled roles can differ from the screenshot below).

Click on the Symbol vaguely looking like a human body in the right lower corner and select the z/OS Modernization Developer (advanced) Icon. Verify your screen looks like shown above.

This role will automatically enable further associated roles like the Web Service Developer.

3.Close the Welcome window.

3. Locate the Web Service Description

WSDL (Web Services Definition Language) is an XML vocabulary that is a description of a service interface. The WSDL document contains a list of the available operations that can be invoked, a description of the information that is sent to and/or received from the service (including 'fault' information when problems occur), the protocol and transport used to access the service (e.g. SOAP over HTTP), and the location of the service.

There are several ways to get WSDL: e-mail; FTP or some other file transfer; from a web site like www.xmethods.com; using WSIL (Web Services Inspection Language), or using a UDDI server (Universal Description Discovery and Integration).

E-mail: WSDL is usually stored in a single file. This WSDL could easily be an attachment you might receive from your business partner to access their service.

FTP or other file transfer: Again, since the WSDL is simply a file, normal file transfer programs are a common way of exchanging WSDL.

Web sites: There are web sites available on the internet that offer Web Services that you can use or test with. Some of these are free, but some may charge a fee. An example of a web site that offers Web Service access is http://www.xmethods.com.

WSIL: Web Services Inspection Language is a language that allows you to query a Web Service provider site about the Web Services they offer and their interfaces. Support of this facility is not required by a Web Services provider. Although this facility may be useful in testing, it is doubtful that a production Web Service location will want to support ad hoc queries about their Web Services. Because of this, it is unlikely that you will see this facility available on a production Web Service web site.

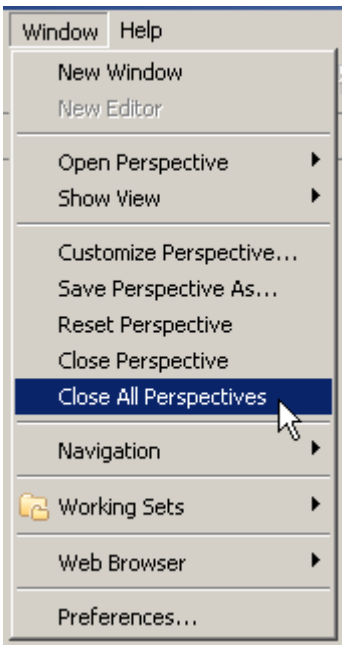
UDDI: A Universal Description, Discovery and Integration server is also known as a Business Registry. This server is similar in function to yellow pages and allows a business to register information about their available Web Services. A potential consumer of a Web Service can access a UDDI server using an architected API and request information about Web Service providers. UDDI servers allow a taxonomy or grouping, so potential consumers can ask for information on a specific Web Service provider, or can ask for a list of providers that offer a particular type of service. You could, for example, request a list of all banks that offer mortgage services. There are public UDDI servers, or you could run your own UDDI server. A UDDI server is supplied with WAS ND (WebSphere Application Server Network Deployment).

For this tutorial we have supplied the WSDL in the local PC file system.

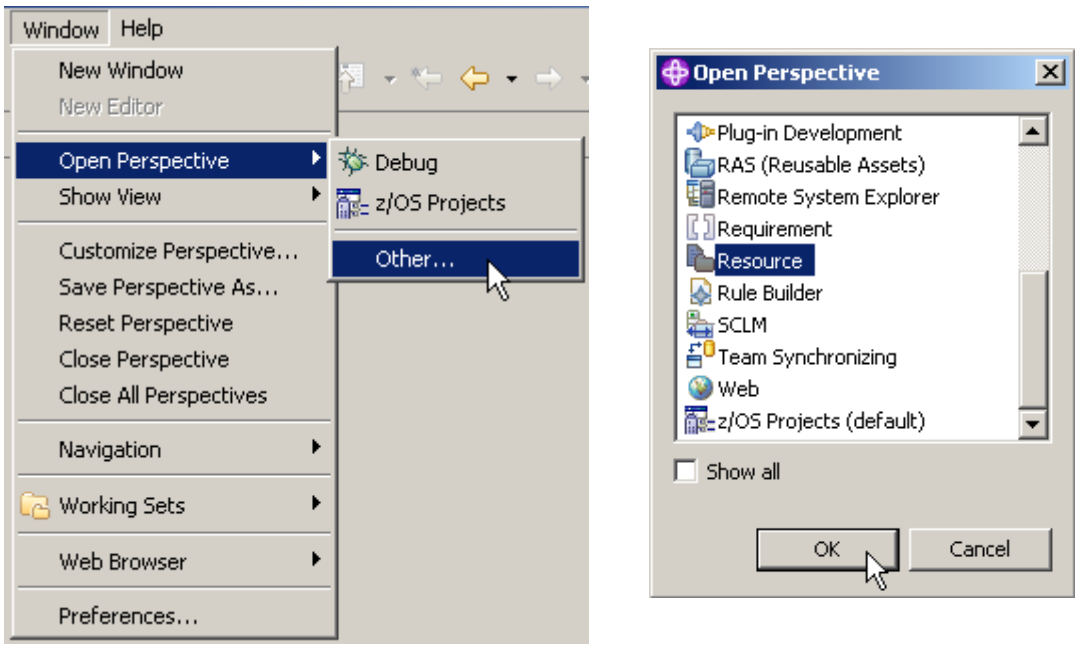
4. Create a Project

Computer artifacts are physical files that execute or are used by your software. Artifacts may include specifications, designs, code, use cases, class diagrams, etc. An artifact is one of many kinds of tangible byproduct produced during the development of software.

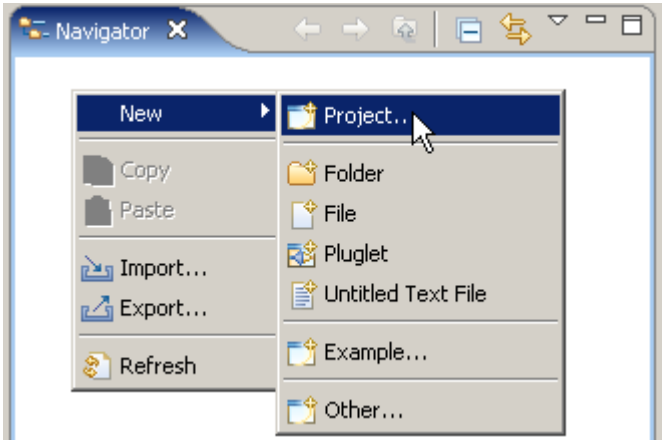
The various artifacts that we create in WDz are organized into Projects. We will create a simple project named WSDLTest. We will be working from the Resource perspective, but the functionality we will be using is available in any type of project in any type of perspective. The menus and wizards we will be using look at file extensions to determine which functionality should be offered for that file.



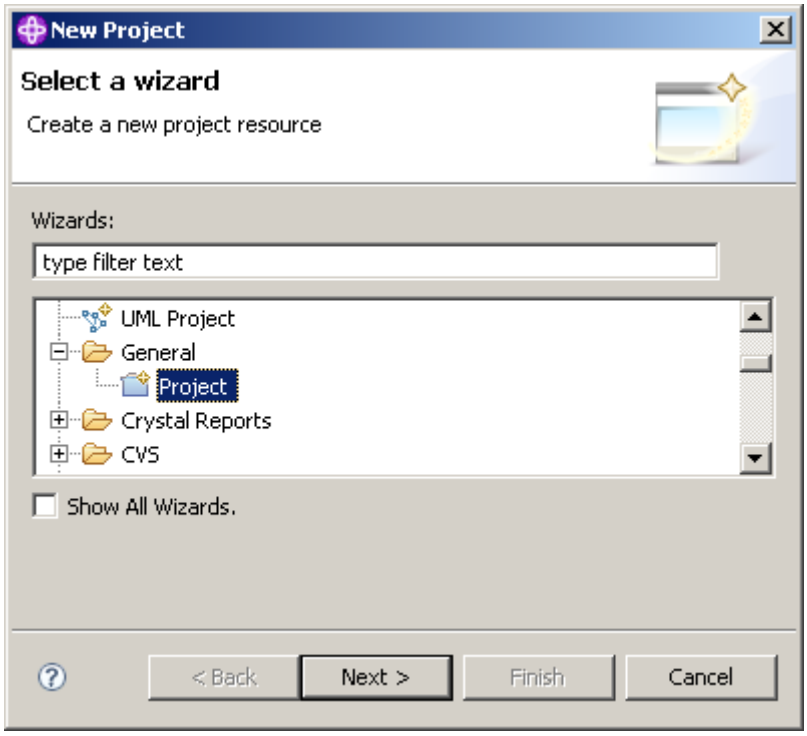
4.From the menu bar, select Window → Close All Perspectives.



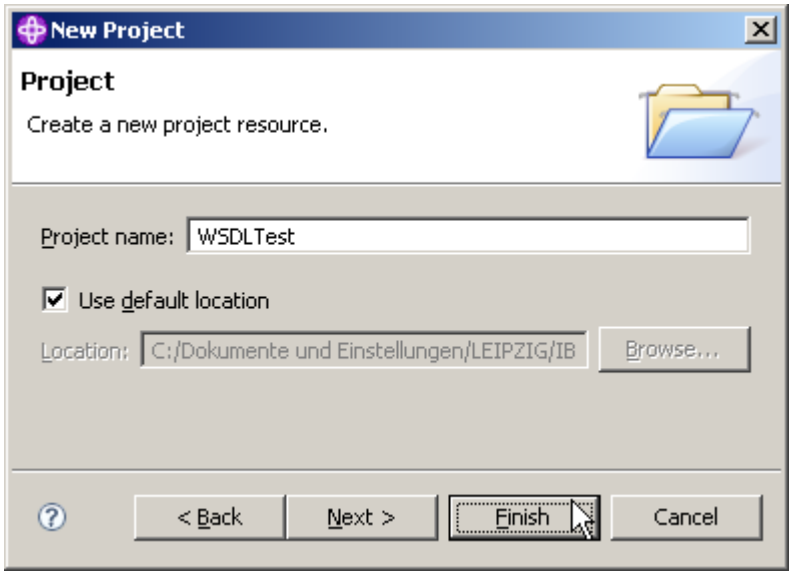
5.From the menu bar, select Window ? Open Perspective ? Other and select Resource to open the resource perspective. Click OK.



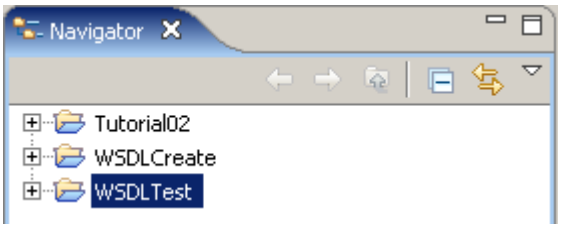
6.From the Navigator view, right-click in an open area, and from the context menu select New → Project….



7.Expand General and select Project.
Click the Next button.



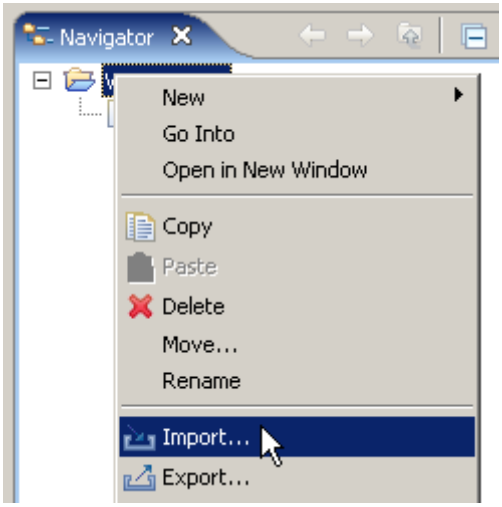
8.In the New Project dialog, specify the Project name as WSDLTest, ensure Use default location is checked, then click the Finish button.



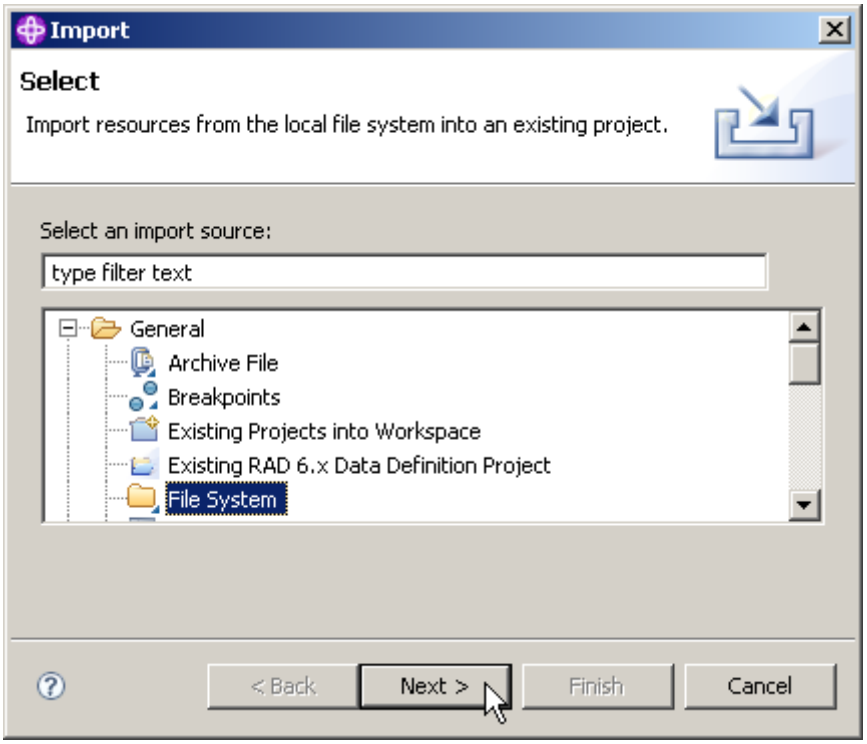
The Navigator view now contains a new entry WSDLTest.
And if you expand it you will see that it is empty besides the project description file .project.

5. Analyze the WSDL for the Web Service

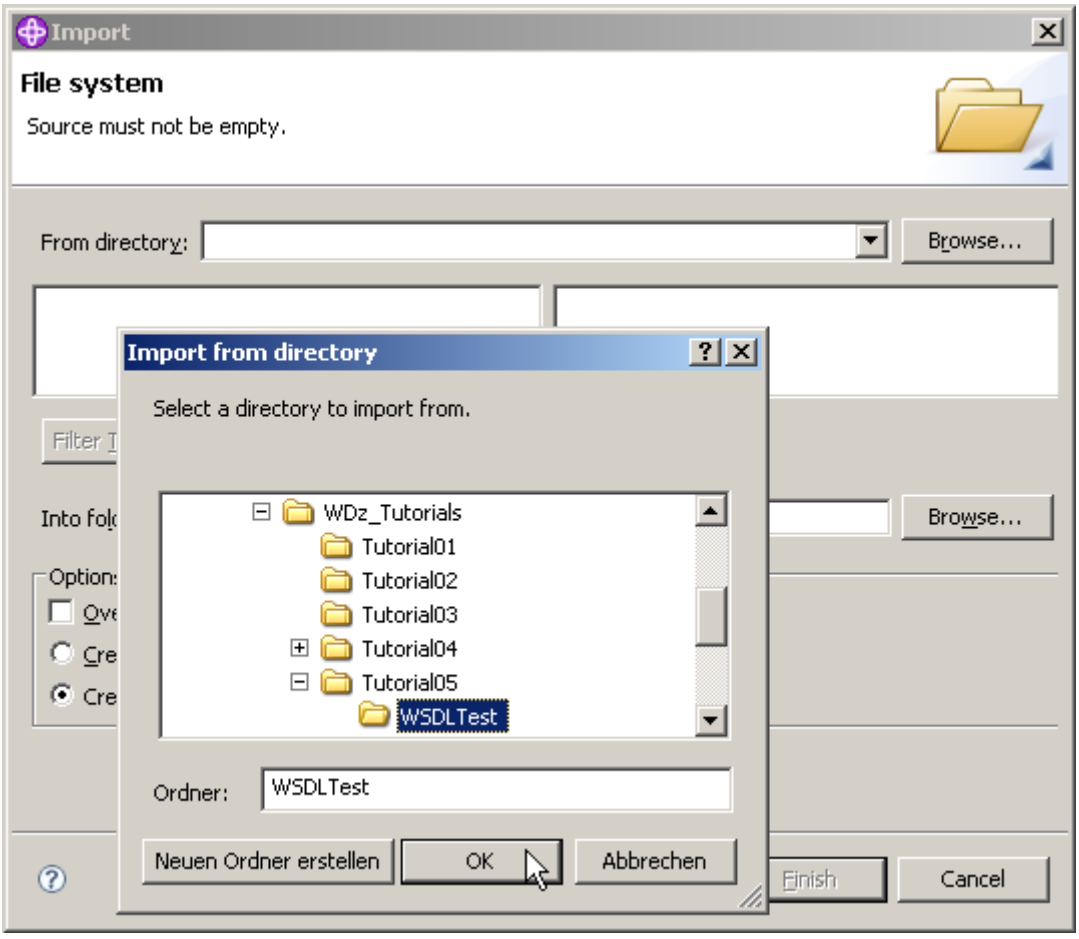
In this part of the tutorial, we will import the WSDL into our project and look at its contents.



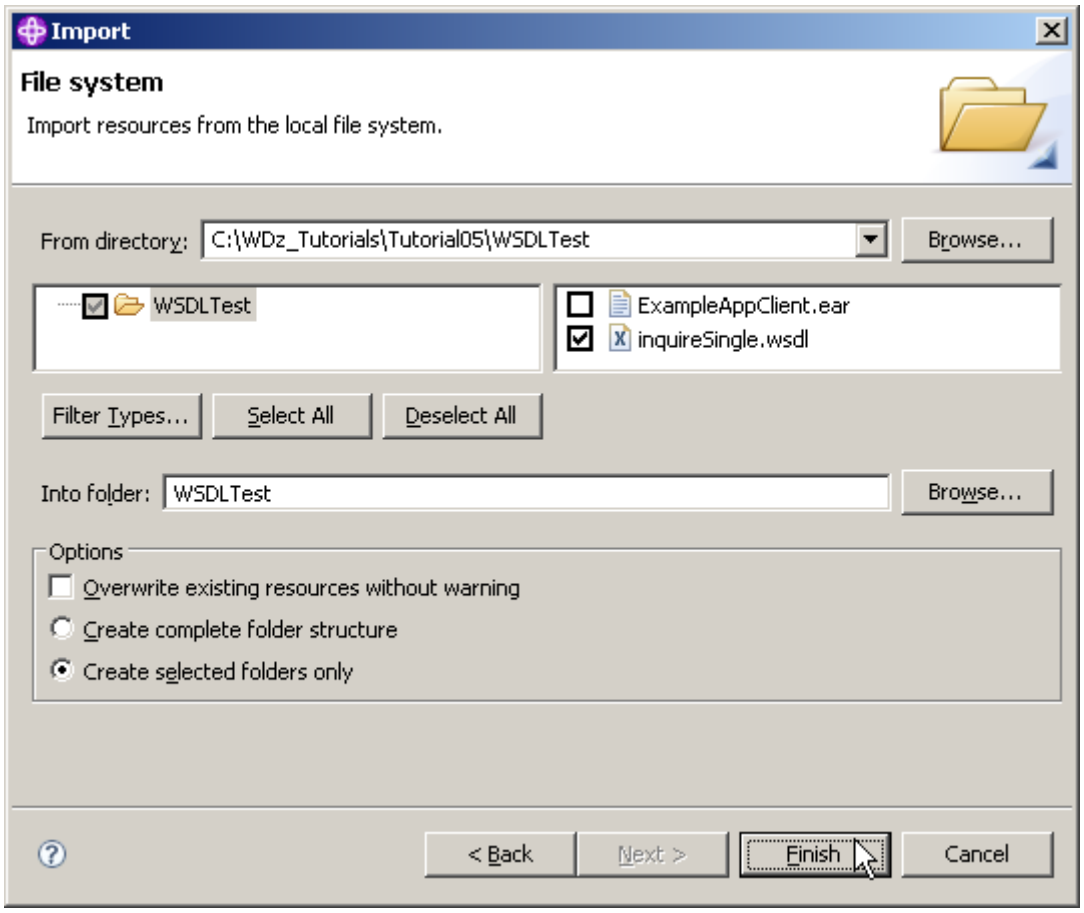
9.From the Navigator view, right-click the WSDLTest project, and from the context menu, select Import.



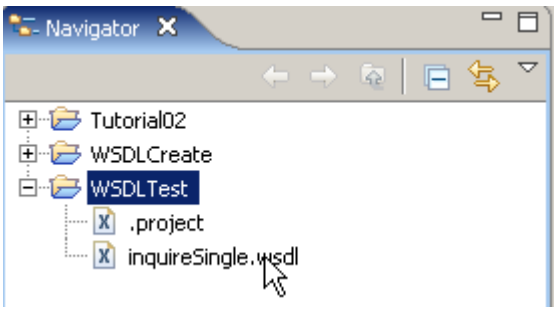
10.In the Select dialog of the Import wizard, expand General and select File System. Click the Next button.



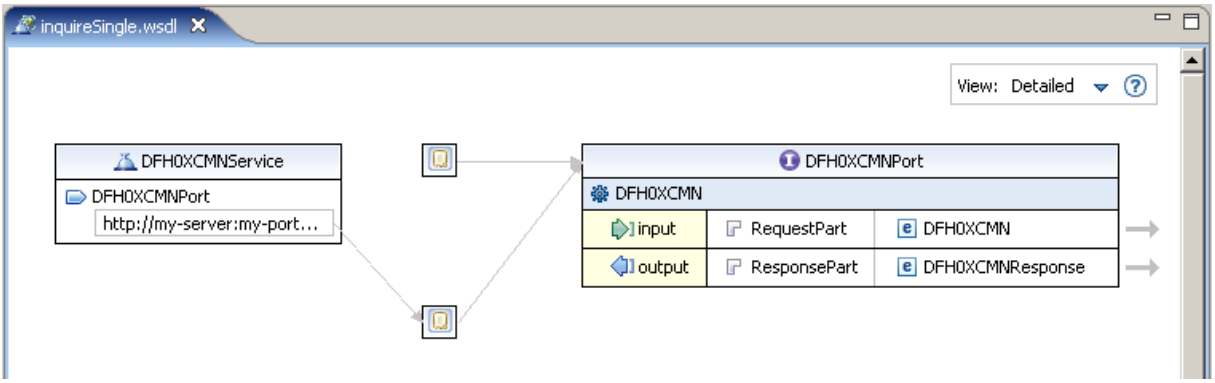
11.Click the Browse button on the top right and navigate to C:\WDz_Tutorials\WDz Tutorial05\WSDLTest and click OK.



12.Back on the File system page of the Import wizard, check the box next to InquireSingle.wsdl, ensure that the Into folder is WSDLTest, ensure that Create selected folders only is checked and click the Finish button.

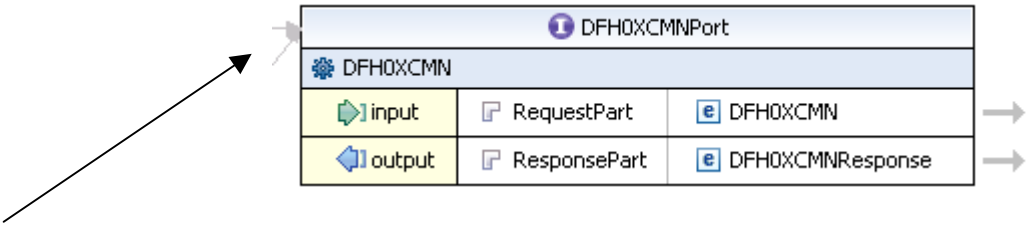


13.From the Navigator view, expand the WSDLTest project. Double-click on inquireSingle.wsdl. This will open the file in a WSDL editor, which is supplied with WDz.



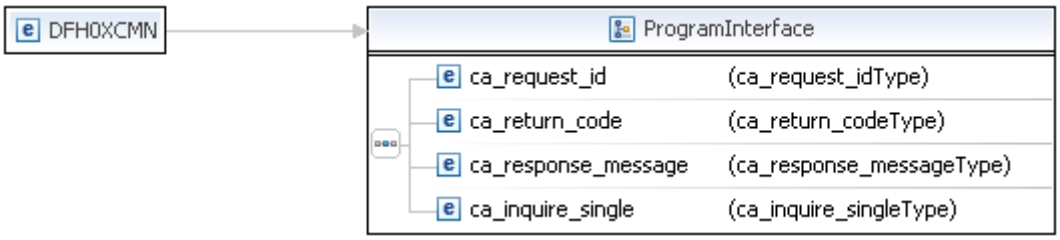
Note that inquireSingle.wsdl is part of the supplied CICS Web Service example which has already been installed on the System z mainframe. We will use the inquireSingle.wsdl file to test one of the functions in the CICS supplied Web Services example application.

We will now investigate the InquireSingle.wsdl. If you completed Tutorial04, you should recognize most of the WSDL-elements.

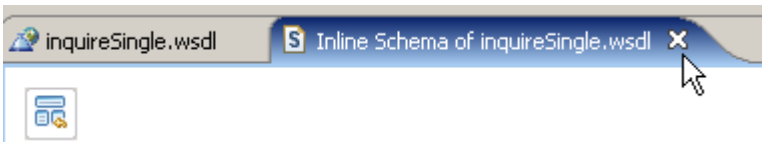


14.DFH0XCMNPort is the name of the Port Type. A Port Type is a collection of operations. In this case, we only have one operation called DFH0XCMN. The operation has an input, an output, and will not return a fault.

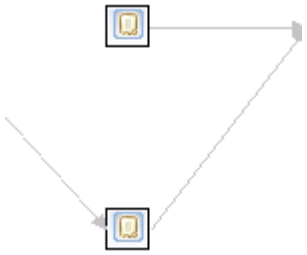
Note that the DFH0XCMNRequest message contains one part. The part is named RequestPart which refers to DFH0XCMN.



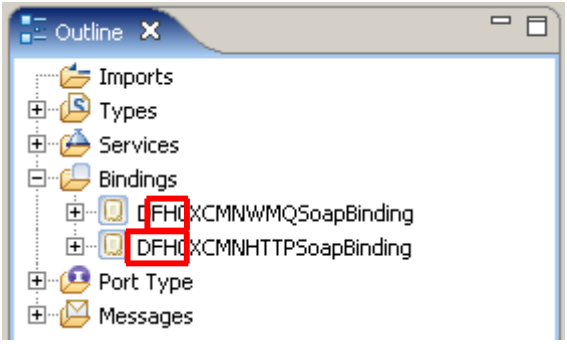
15.To see the Inline Schema of InquireSingle.wsdl, just click on the arrow on the very right of your input or output. You may have to click on DFH0XCMNPort to see the arrows.



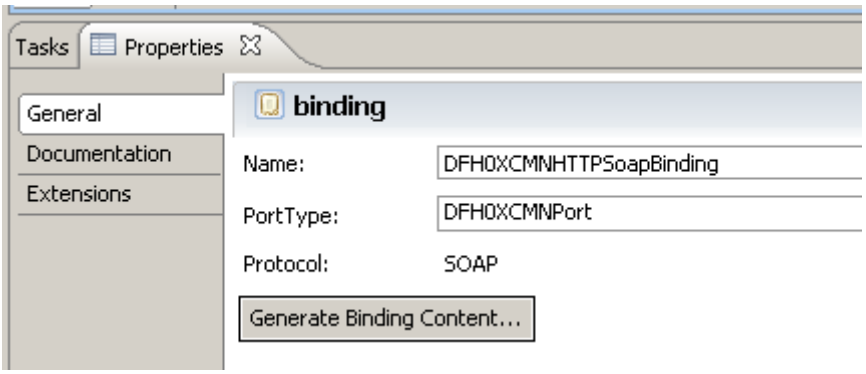
16.Close the Inline Schema and …



have a look at the bindings. Note that there are two bindings. Move the mouse over the small rectangle to see the name of the binding.

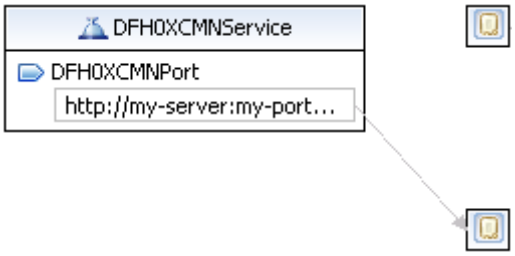


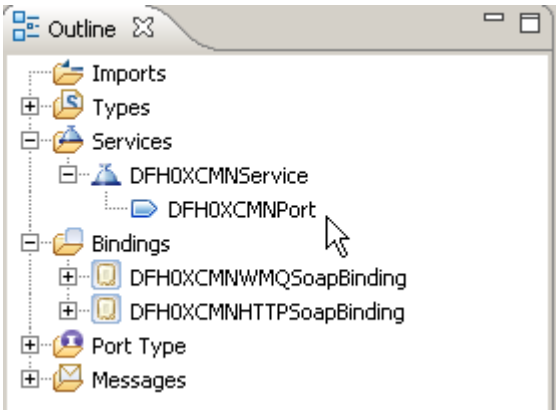
You can also see the names of the bindings in the Outline view.



You can also see the names of the bindings in the Properties view. If the Properties view is not available, go wo Window → Show View → Properties .

From the names of the bindings you can tell that one binding is for SOAP over HTTP and one is for SOAP over MQ. It shouldn't be too surprising that they are identical. Whether the SOAP is sent over HTTP or MQ, the SOAP will be exactly the same. WSDL allows you to have multiple bindings for a Port Type.



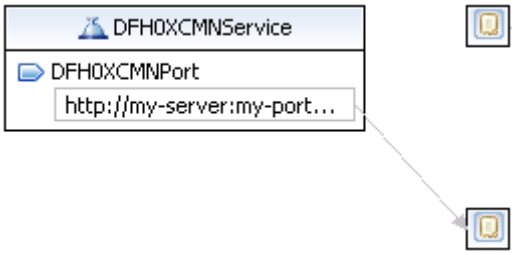


17.Finally, have a look at the Service DFH0XCMNService. A Service entry defines the location of the service.

Click on DFH0XCMNPort under DFH0XCMNService and notice (Again in the properties view) that this is the location of the Binding that defines SOAP over HTTP. Interestingly, there is no location specified for the Binding that defines SOAP over MQ (No link from Service to Binding)

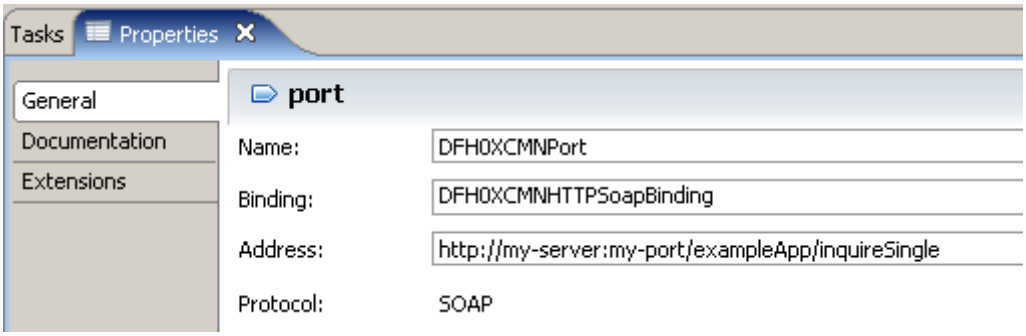
6. Change the WSDL End-point

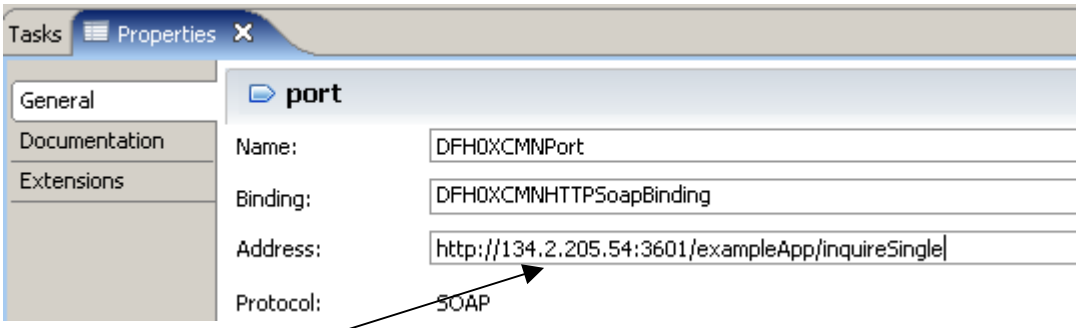
In this part of the tutorial we will change the end-point of the Web Service. The end-point is the physical location of the service.



18.We still use the WSDL editor for the file inquireSingle.wsdl in the Navigator window.
In the graphical view of the editor, in the Service element DFH0XCMNService, click on the address under DFH0XCMNPort to make the end-point editable.

(Note, that you can change it as well in the properties view)





19.The WDz package that was installed on hobbit.informatik.uni-tuebingen.de (134.2.205.54) contains a sample application that we will use. To access it, we need to change the predefined, hypothetical address of the WSDL file. Be very careful to use the correct values, and do not overlook the use of port no. 3601 .

Because it is more comfortable, we specify the end-point using the properties view. Just change the Address from http://my-server:my-port/exampleApp/inquireSingle to http://134.2.205.54/exampleApp/inquireSingle .

Why all this ? If you use a standard browser, and enter

http://134.2.205.54:3601/exampleApp/inquireSingle

you will get this response:

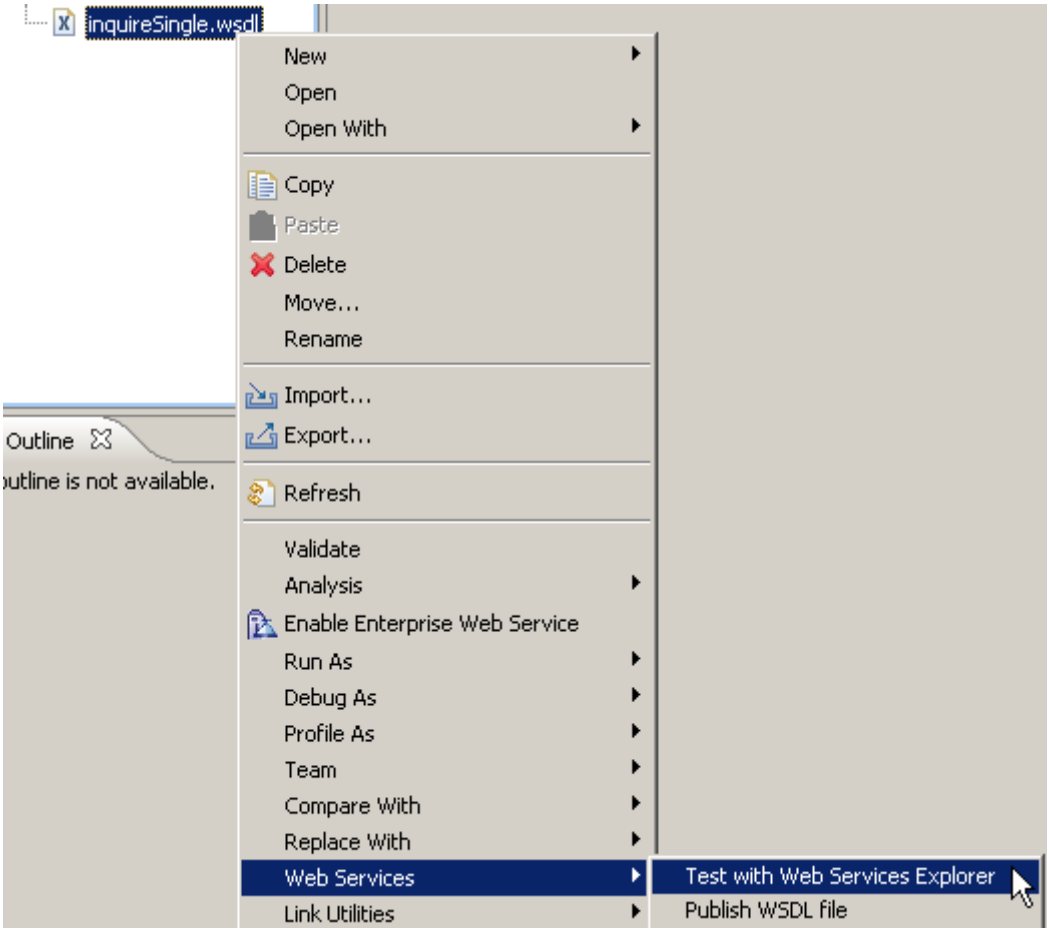
```
<SOAP-ENV:Envelope>
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>SOAP-ENV:Server</faultcode>
      <faultstring>Internal Server Error</faultstring>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

20.Don't forget to save the WSDL file (CTRL + S)

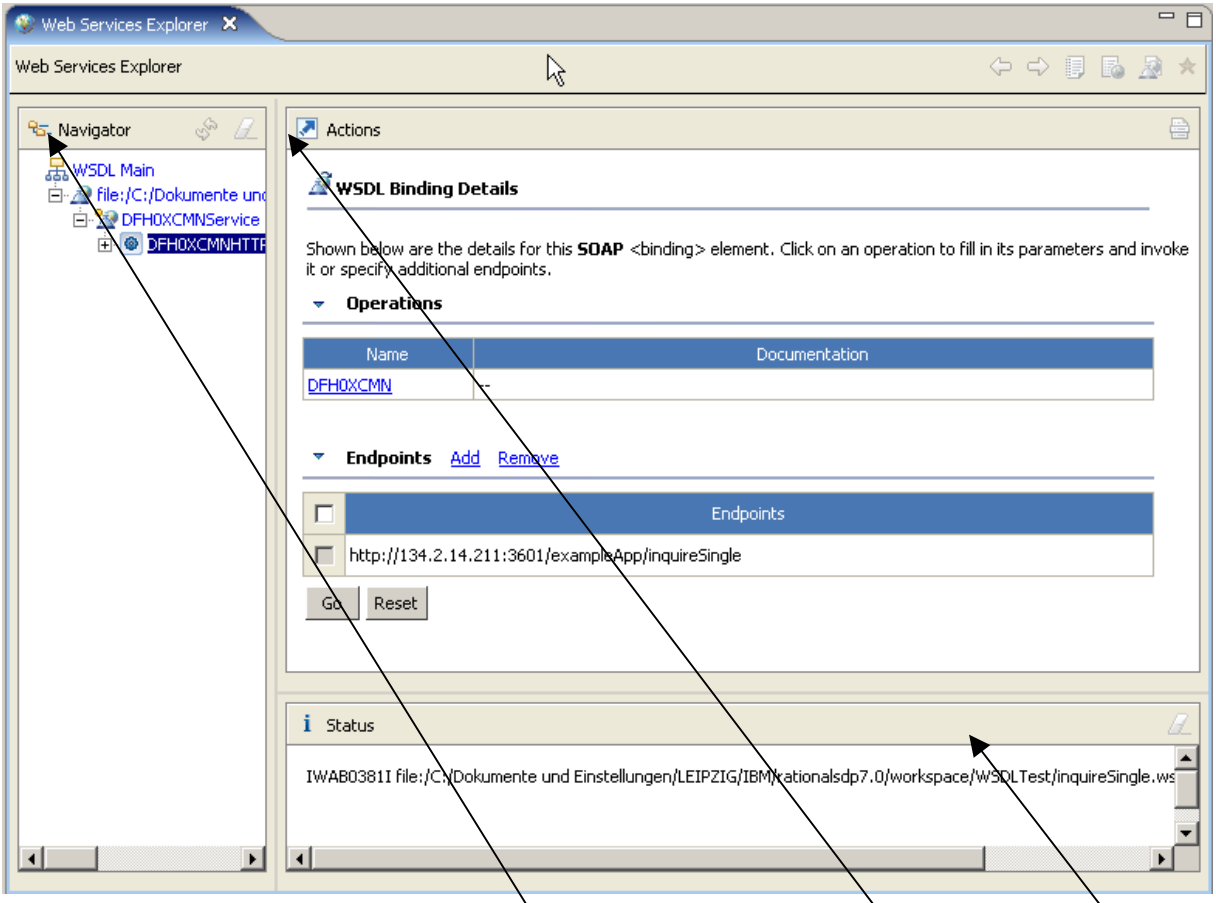
and close the WSDL editor.

7. Test the Web Service

The simplest test of a Web Service is just to invoke the Web Service and see if it returns the expected results. We will use the Web Services Explorer. Thus you don't need to write a Web Service requester (client) program.

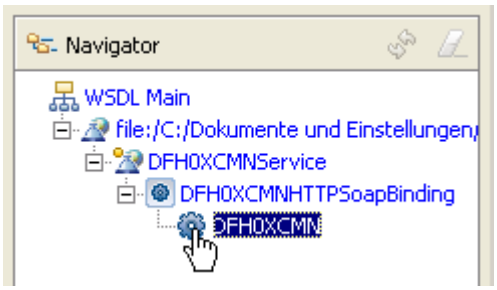


21.From the Navigator view, right-click on inquireSingle.wsdl and from the context menu, select Web Services → Test with Web Services Explorer.

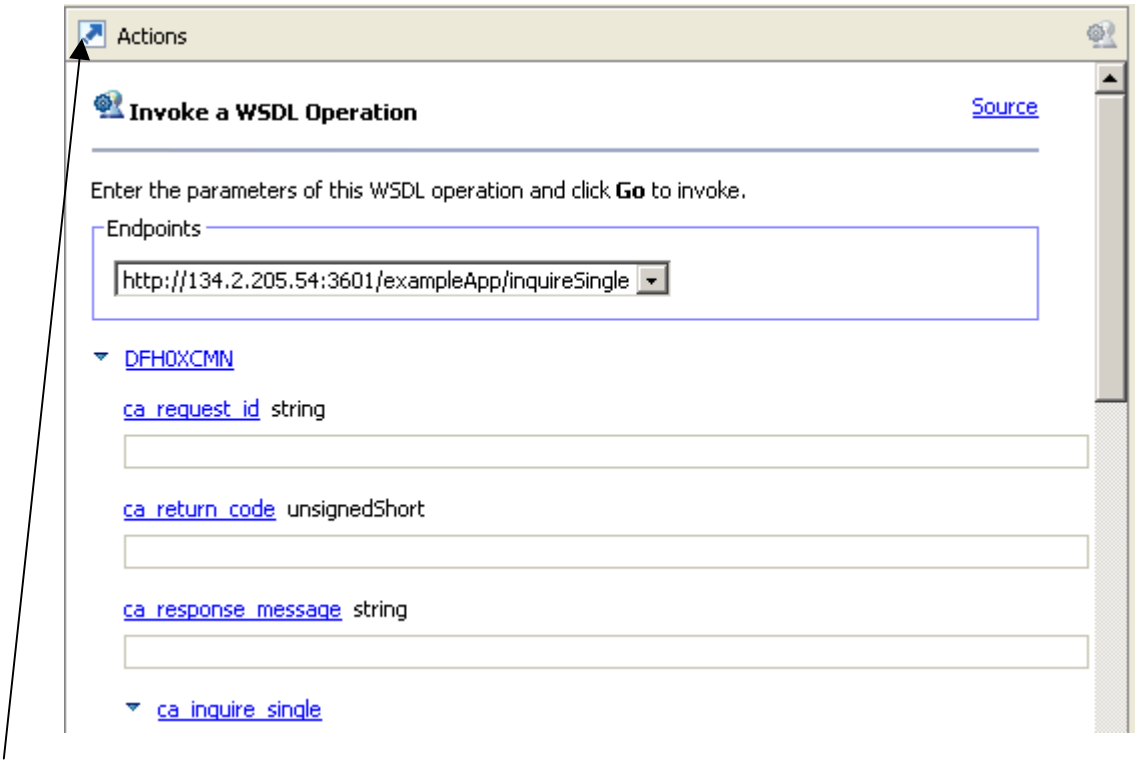


22.You will notice that a Web Browser view opens.

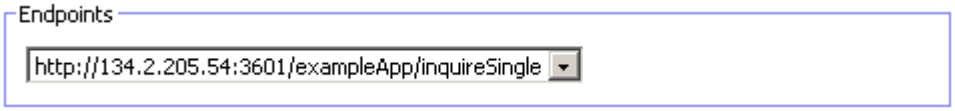
The Web Browser view consists of 3 panes: Navigator pane (left), Actions pane (right) and Status pane (bottom).



23.In the Navigator pane of the Web Services Explorer, expand DFH0XCMHNTTPSoapBinding. Select the operation DFH0XCMN (the business operation) by double clicking on it..



24.Your Actions pane should look similar to the one above:



25.From the drop down menu in the Actions pane, enter the same endpoint we specified before (if not already chosen by default).

You see in the Actions pane a number of fields labeled ca_request_id, ca_return_code, etc. You can enter a value in each one of these fields.

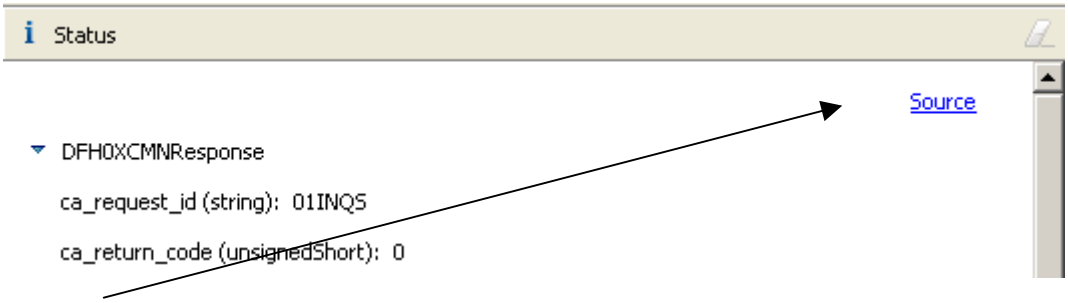
Field	Value
ca_request_id	01INQS
ca_return_code	0
ca_response_message	(space)
ca_item_ref_req	0010
filler1	0
filler2	0
ca_sngl_item_ref	0
ca_sngl_description	(space)
ca_sngl_department	0
ca_sngl_cost	(space)
in_sngl_stock	0
on_sngl_order	0

26.Type in the following values (you may need to scroll down).
These values are for a request for information on item number 0010 in our inventory.

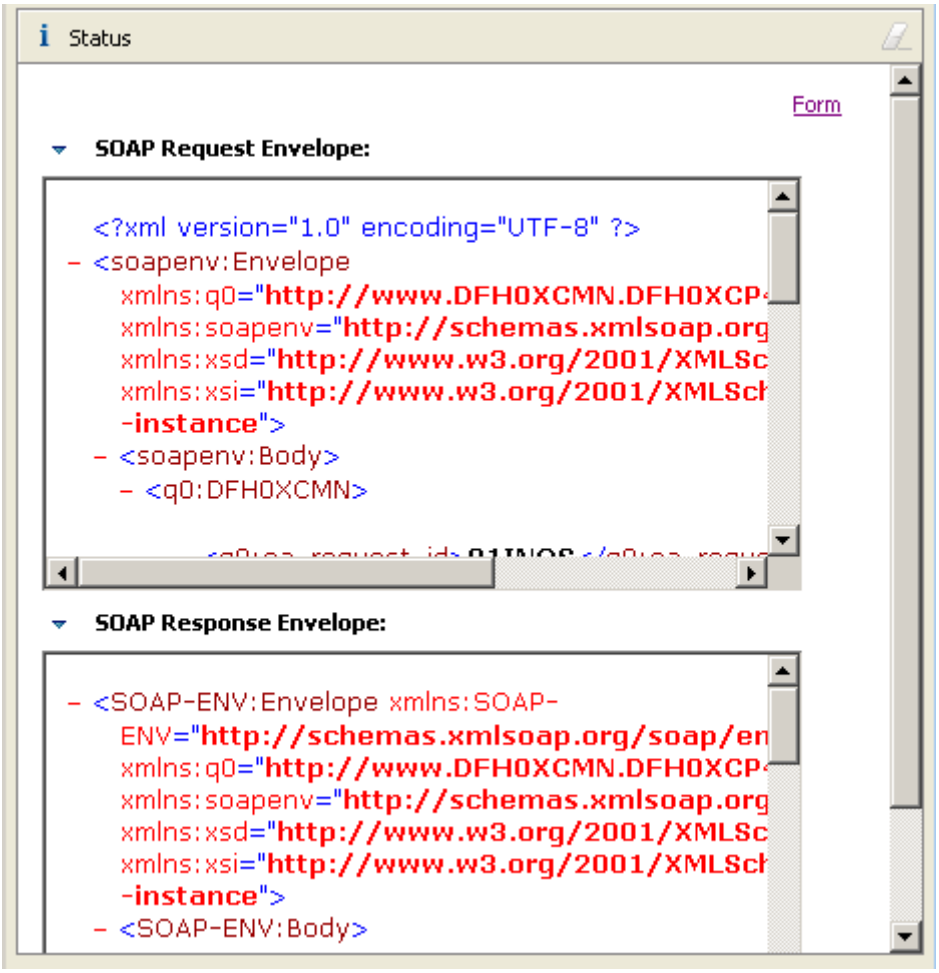
The result should look like this.

Click the Go button at the bottom of the Actions pane

27.The results from the Web Service request will be shown in the Status pane. You can double-click on the title Status to expand this pane. You should see a window similar to the above.



28.Click on Source towards the top right in the Status pane



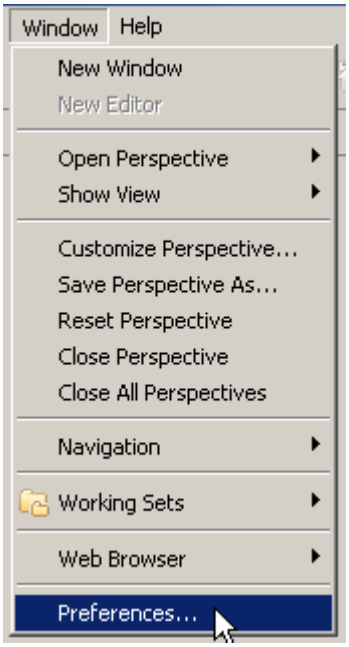
to get a detailed Source Code view of the SOAP request and response.

29.Double-click on the title Status to return to the previous Web Services Explorer view.

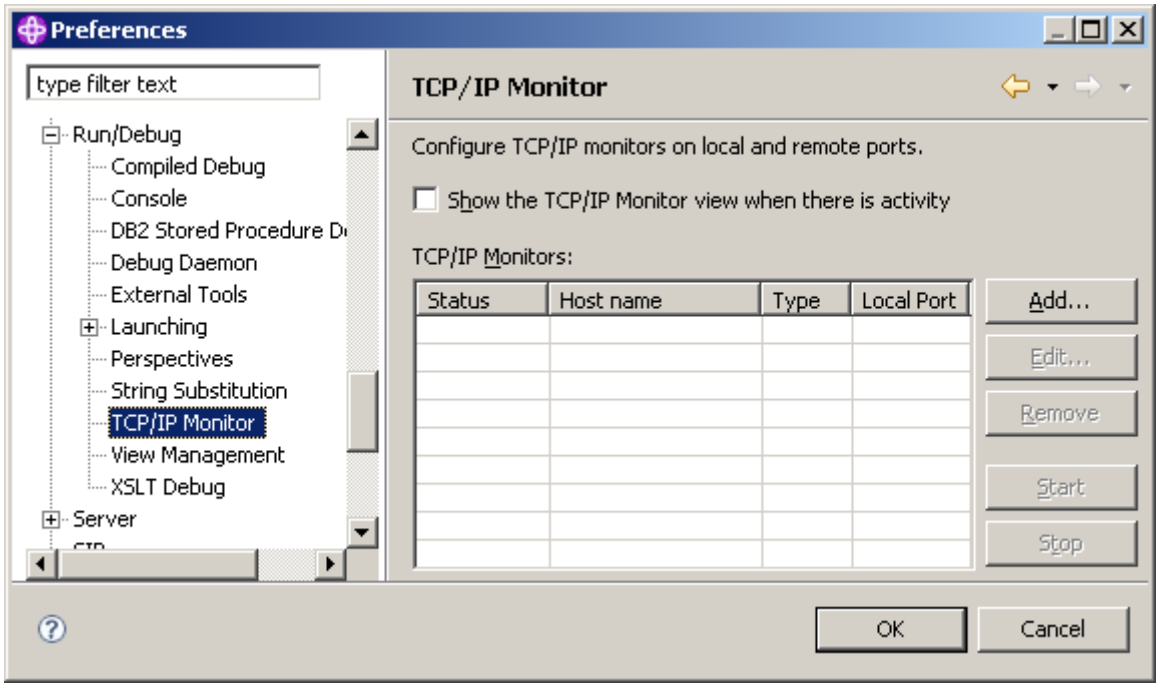
You successfully finished the Web Service test, but don't close the Web Services Explorer as you will need it again for the next chapter.

8. Look at Web Service Flows with the TCP/IP Monitor Server

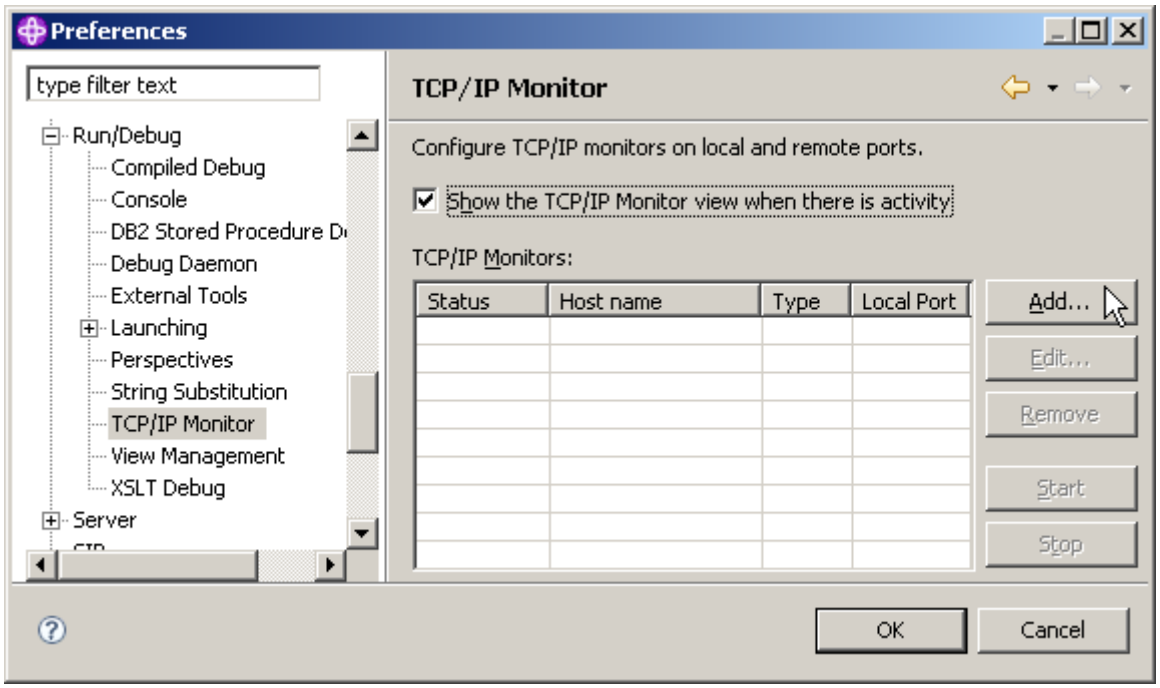
An additional way to test Web Services is to look at the Web Services flows 'on the wire'. We can do this with the TCP/IP Monitor Server that is supplied with WDz. We configure the TCP/IP Monitor Server so that it receives the Web Service request from the Web Service requester, and forwards the request to the Web Service provider. The TCP/IP Monitor Server allows us to view requests/responses as they flow back and forth between the requester and the provider.



30.From the menu bar, select Window → Preferences.



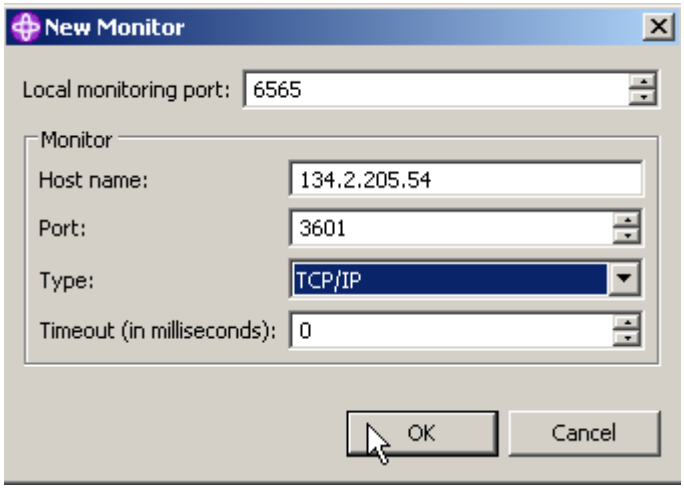
31.In the Preferences dialog, expand Run/Debug and then select TCP/IP Monitor.



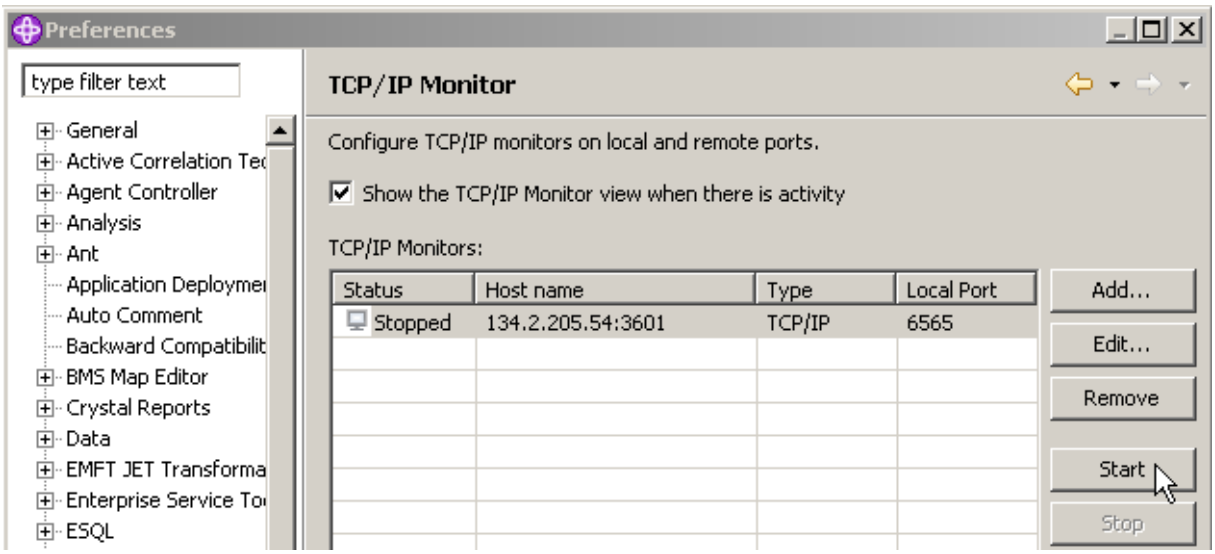
32.On the right, check Show the TCP/IP Monitor view when there is activity to display the TCP/IP Monitor view when there is activity through a TCP/IP monitoring server.
Click Add...

Field	Value
Local Monitoring Port	Specify any unused port, i.e. 6565
Host name	139.18.4.35
Port	3601
Type	TCP/IP
Timeout	0

33.On the New Monitor panel, specify the above settings and

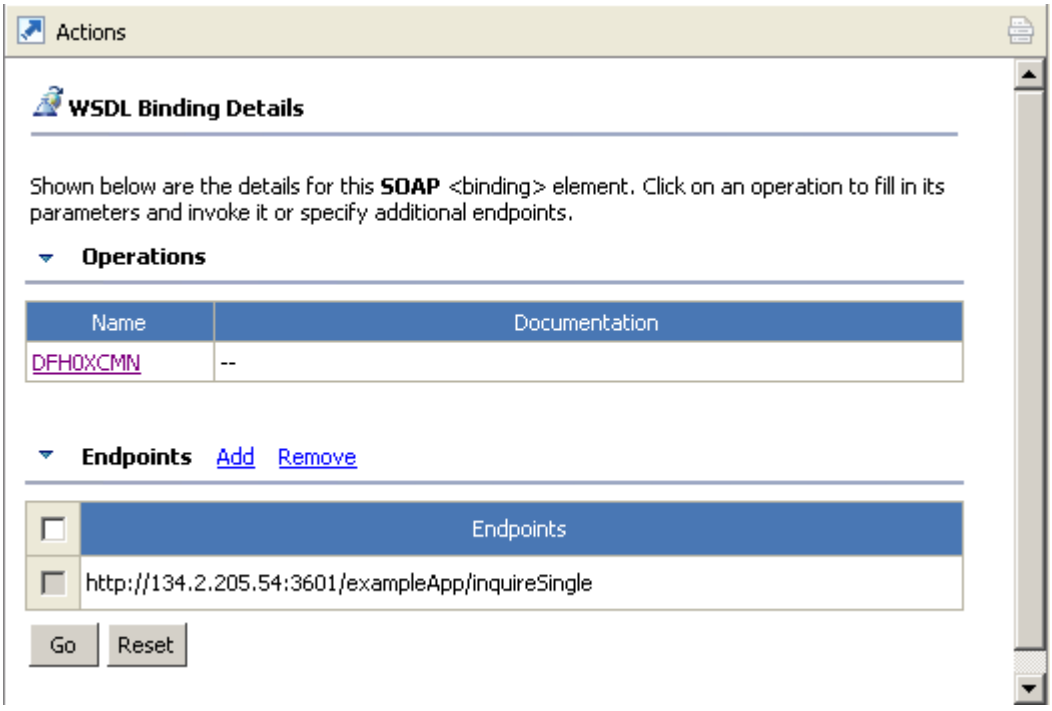


click OK.

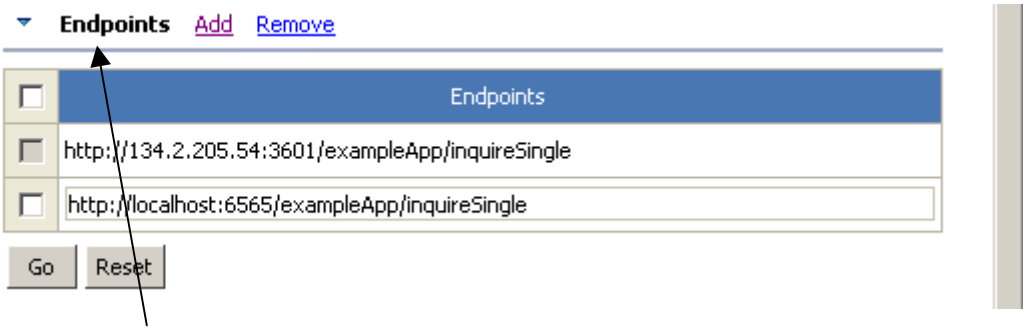


34.Click the Start button to start the TCP/IP Monitor Server.
Notice that the Status changes to Started.

35.Click the OK button to close the Preferences dialog.



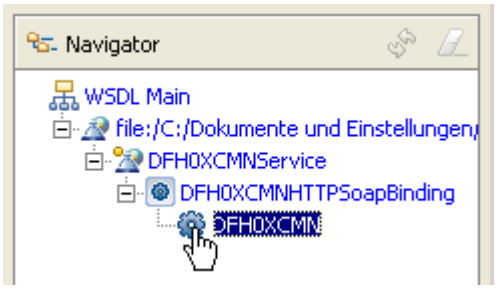
36.Back at the opened Web Services Explorer, from the Navigator pane, click on DFH0XCMNHTTPSoapBinding.



In the Actions pane, click on Add in the Endpoints section and a new Endpoint will appear. Change the Endpoint to http://localhost:6565/exampleApp/inquireSingle

Finally click Go.

This sets the endpoint to the TCP/IP Monitor. After receiving the Web Services request, the TCP/IP Monitor will then send the request to the destination we specified when we configured the TCP/IP Monitor.



37.In the Navigator Pane, click DFH0XCMN.

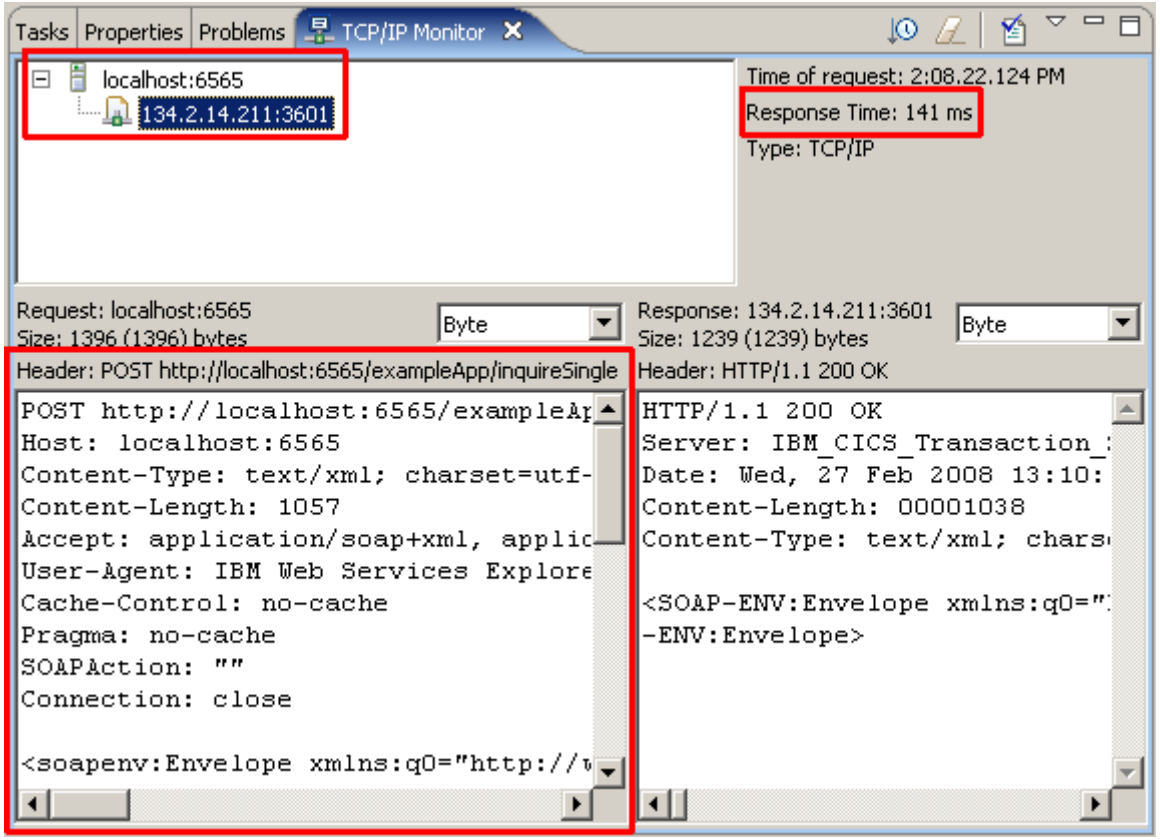
38.In the Actions pane, ensure that the Endpoint http://localhost:6565/exampleApp/inquireSingle is selected.

Field	Value
ca_request_id	01INQS
ca_return_code	0
ca_response_message	(space)
ca_item_ref_req	0010
filler1	0
filler2	0
ca_sngl_item_ref	0
ca_sngl_description	(space)
ca_sngl_department	0
ca_sngl_cost	(space)
in_sngl_stock	0
on_sngl_order	0

39.specify the above values.

This should look like shown above.

Click the Go button.

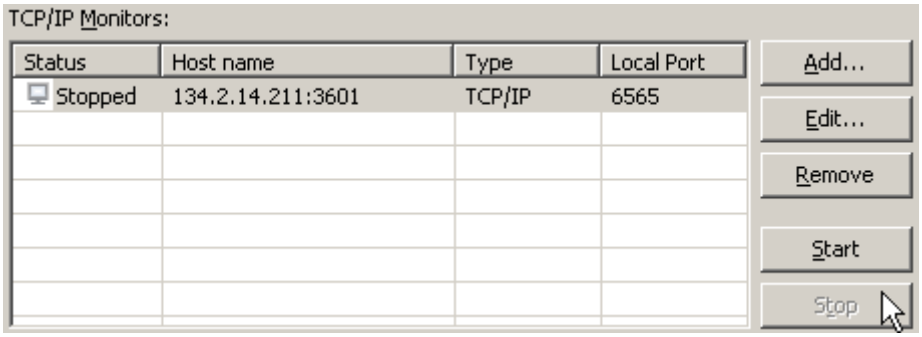


40.In addition to the information returned from the Web Service to the Web Services Explorer, you will see a TCP/IP Monitor view.
(Note that you may wish to resize the view to allow it to display more information. Double-click on the TCP/IP Monitor tab to maximize it.)

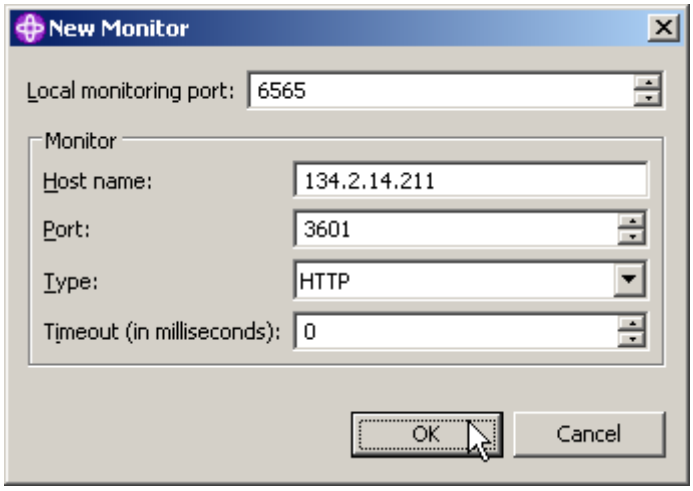
Note the response time of the request (141 ms on the screen shot above). This is the time from when the request left the TCP/IP Monitor server to the time it received the response from the request.

The TCP/IP Monitor Server shows you the information that was sent 'on the wire'. If you have multiple requests/responses, you can use the tree structure in the upper part of the TCP/IP Monitor view to select the request/response you want to analyze.

Also note that in addition to the SOAP request and response, you can see the additional HTTP information that flowed with the request and response.

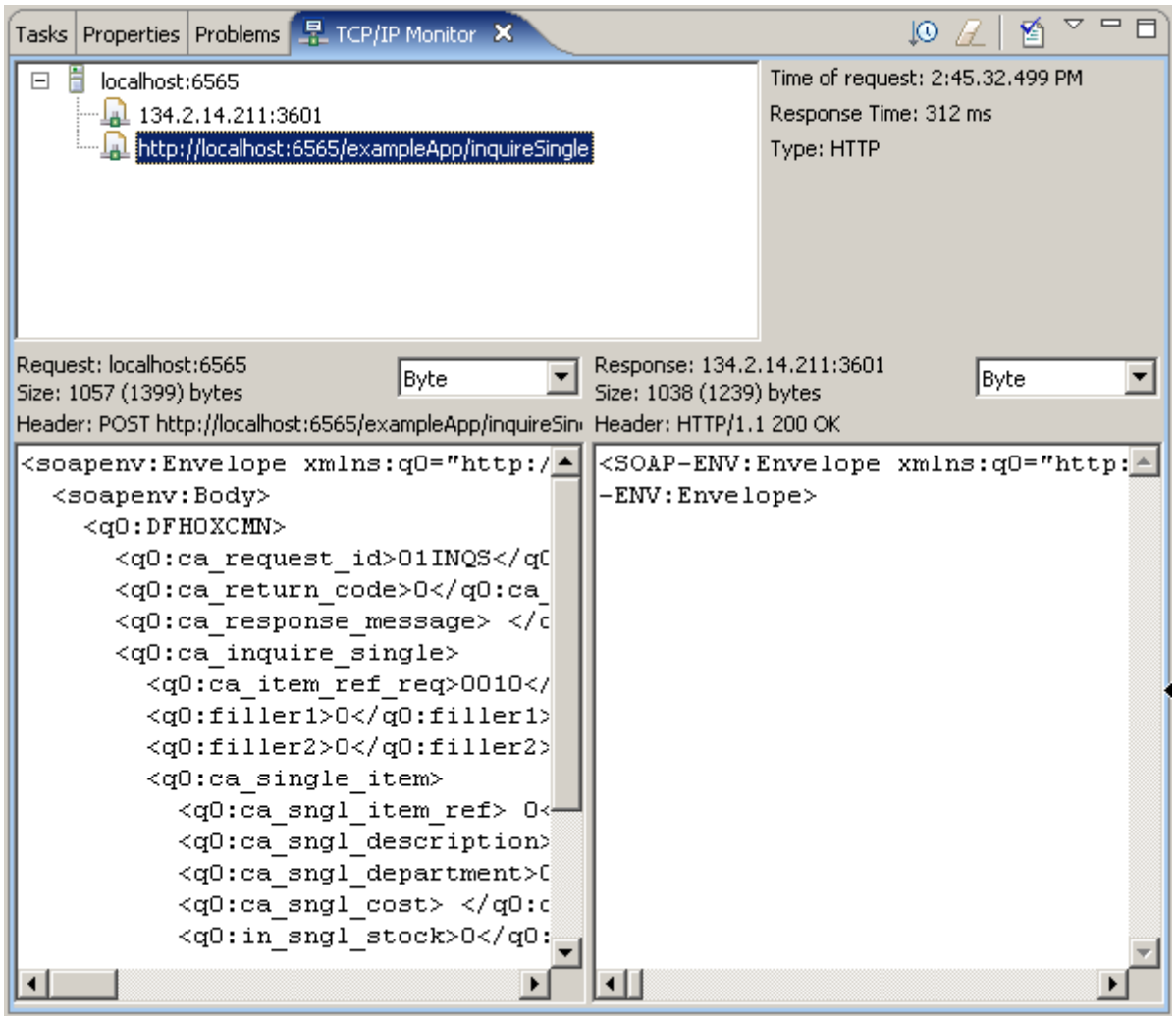


41.Now let us experiment a little bit with the TCP/IP Monitor.
Go back to the beginning of this chapter and stop the TCP/IP Monitor.

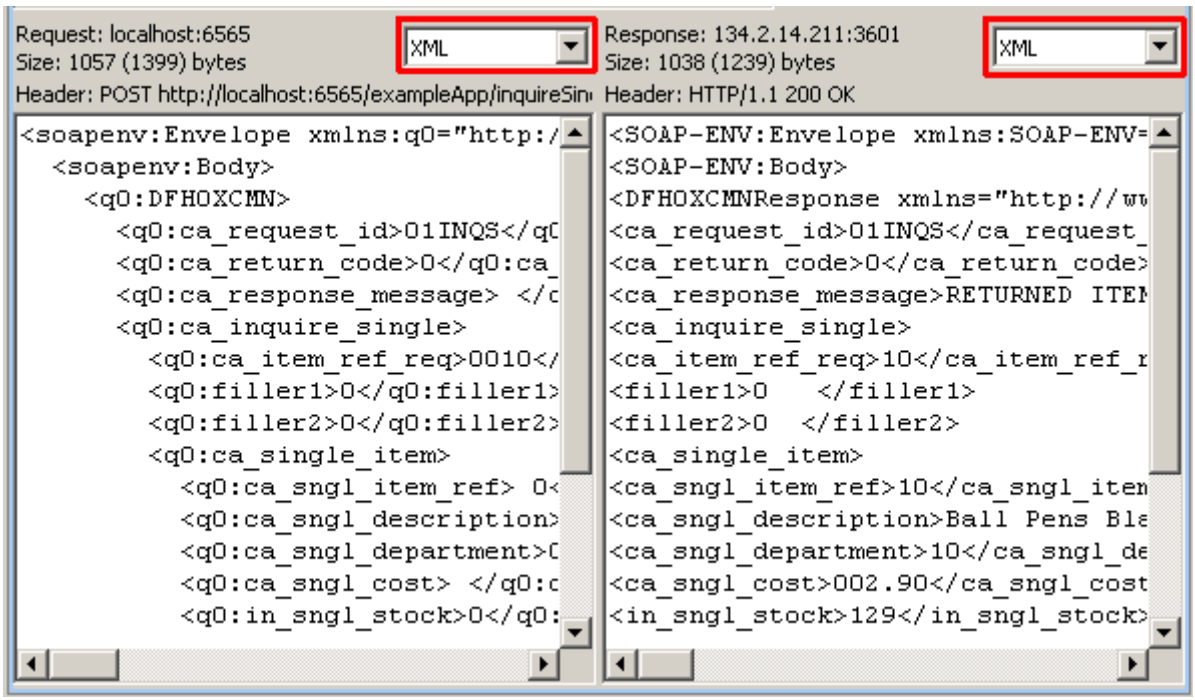


42.Click Add and enter the same values as before, but change the Type to HTTP (instead of TCP/IP).

43.Then proceed as you did above (repeat steps 34-39).



44.In the TCP/IP Monitor view you can now only see the SOAP request and response messages.



45.Now change the view to XML View to see the actual SOAP Message Request and Response in XML

46.Don't forget to stop the TCP/IP Monitor when you are finished

Congratulations, you successfully completed Web Services Tutorial 02!

Resources

Webtut02.zip contains resources as a zip Archiv. Download at

www.cedix.de/Vorles/Band3/Resources/webtut02.zip

RDz Web Services Tutorial 03
CICS Web Services Enablement

Content

1. Tutorial Overview

Within this tutorial, you will enable one function of the CICS Catalog Manager as a Web Service. You will therefore use WebSphere Developer for System z Version 7 (WDz) to generate the required components, define new CICS resources, and test it.

This method of enablement is called bottom up because you start from the bottom (i.e. your backend program) and generate the interface to the top.

1.1 Scenario

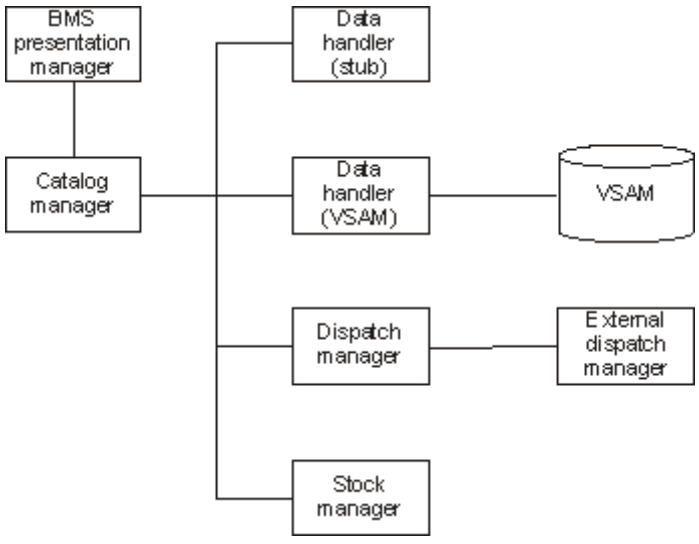
CICS TS V3.1 on your z/OS system includes a sample application called CICS Catalog Manager. This is basically a catalog ordering system for office materials whose modular design makes it perfectly suitable for modernization and reuse.

For more information about the CICS Catalog Manager read "Discover the CICS Catalog Manager". It is located on DVD and should be in your virtual machine at C:\WDz_Tutorials\Tutorial06\CatMan_Ressources.

The CICS catalog example is a working COBOL application that is designed to illustrate best practice when connecting CICS applications to external clients and servers. The example is constructed around a simple sales catalog and order processing application, in which the end user can perform these functions:

- **List the items in a catalog.**
- **Inquire on individual items in the catalog.**
- **Order items from the catalog.**

The base application has a 3270 user interface, but the modular structure, with well-defined interfaces between the components, makes it possible to add further components. In particular, the application comes with Web Service support, which is designed to illustrate how you can extend an existing application into the Web Services environment. The catalog data is stored in VSAM. You can run the transaction from a CICS® terminal using the Transaction ID (TRID) EGUI. An example, using BMS presentation logic is shown in Appendix E.



The example application has already been installed on 134.2.205.54 (hobbit). The 3270 user interface has been configured ready for you to use. The CICS System Definition (CSD) group DFH$EXBS contains the application resource definitions.

Details of the CICS catalog manager example application can be found under:
http://publib.boulder.ibm.com/infocenter/cicsts/v3r1/index.jsp?topic=/com.ibm.cics.ts31.doc/dfhxa/topics/dfhxa_t100.htm

Assume you want to modernize the CICS Catalog Manager. You decide to use CICS Web Service support for the modernization. Your decision might result from the fact, that you might want to send data larger than 32k (this disqualifies the CICS Transaction Gateway). Alternatively you may want to provide a Web Service interface to de-couple your clients from the backend so they can use the application without having to know anything about the implementation.

1.2 Tutorial Requirements

Please note that there are often several ways to perform functions in and for CICS. This tutorial will present one of the ways. If you are familiar with CICS, you will notice that some of the statements are general, and not necessarily true for every situation.

This tutorial assumes that you are familiar with the CICS Catalog Manager and WebSphere Developer for zSeries. The appendices contain additional Information:

Appendix A URIMAP resource definitions

Appendix B Using OMVS

Appendix C Build-in CICS Transactions

Appendix D CEDA—Transaction, Resource definition online

Appendix E CICS Catalog Manager (EGUI) Transaktion execution example using BMS presentation logic .

1. Tutorial Overview (this section)

2. Preparation
Create a filter to display Catalog Manager content
We need a filter to be able to access content of the Catalog Manager.

3. Generate resources for the inquireSingle Web Service provider.
You will use WDz to create all required resources to enable the inquireSingle function of the CICS Catalog Manager as Web Service.

4. Prepare HFS
In this part of the exercise we will discuss the creation of HFS directories that are needed for our resource definitions and copy the generated wsbind-file to our remote z/OS system.

5. Prepare CICS
In this part of the lab exercise we will first create a PIPELINE resource definition and reference the CICS supplied sample pipeline configuration file.
Then we need to tell CICS to listen on a TCPIP port for incoming HTML. The TCPIPService resource definition allows us to do this. You will need a TCPIPService definition for each port you want CICS to listen on for incoming Web Service requests.
And finally, we will take time to look at the URIMAP and WEBSERVICE definitions CICS has created for use when it did the SCAN and found a WSBIND file.

6. Test the Web Service
We use the Web Services Explorer in WDz in this part of the lab to verify our work.

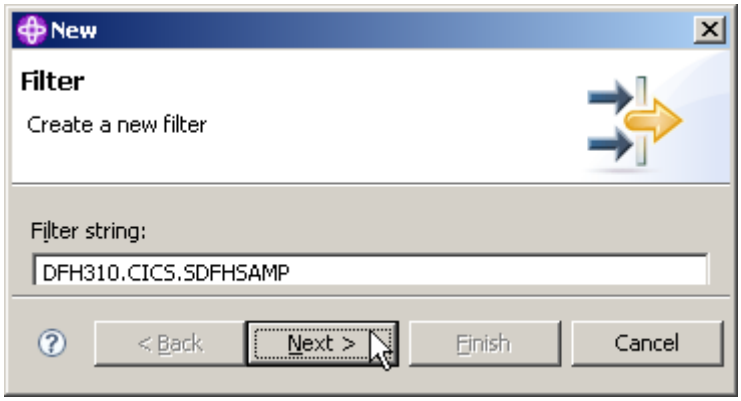
2.0 Preparation: Create Filters to display Catalog Manager Contents

To be able to access the Catalog Manager contents on the file system you will now create a filter in your workspace and specify mappings according to the occurring file types.

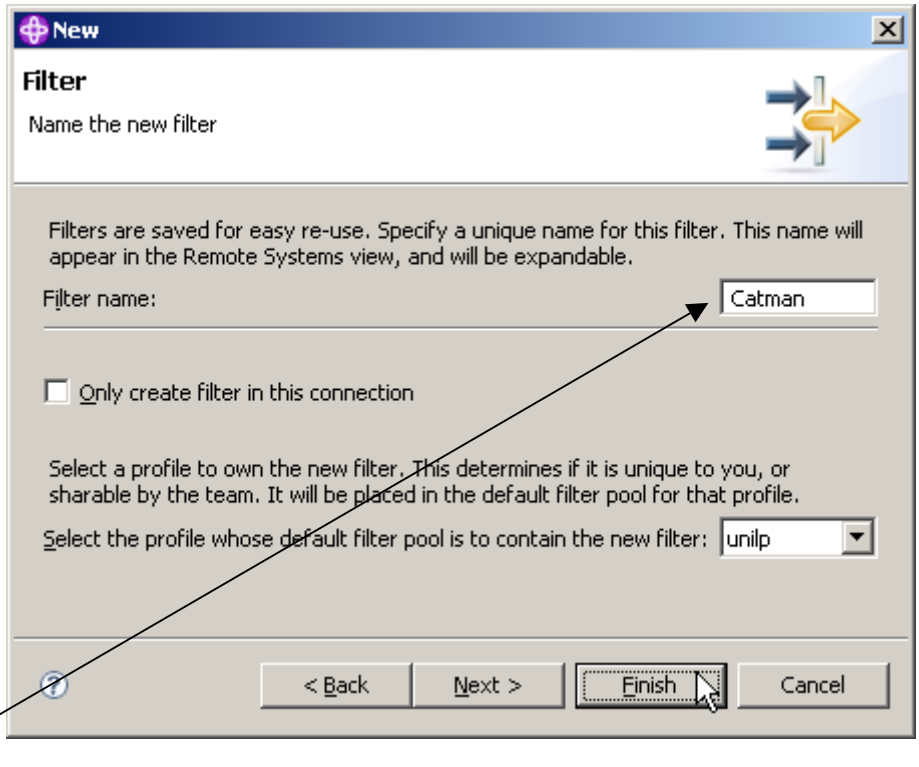
1.Open the z/OS Projects perspective: Window → Open Perspective → z/OS Projects. Connect to your host by right-clicking Frodo and selecting connect from the context menu.
Enter your UserID and your Password and click OK.

2.Expand frodo, right-click MVS Files in your Remote Systems view and select
New → Filter…

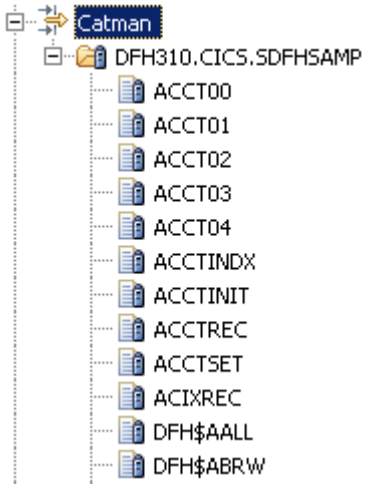
Please reaBitte lesen Sie den Text für jeden schritt sorgfältig zu ende, before you are entering data via keyboard or mouse.



3.In the New Filter panel, enter DFH310.CICS.SDFHSAMP as Filter string and click Next.

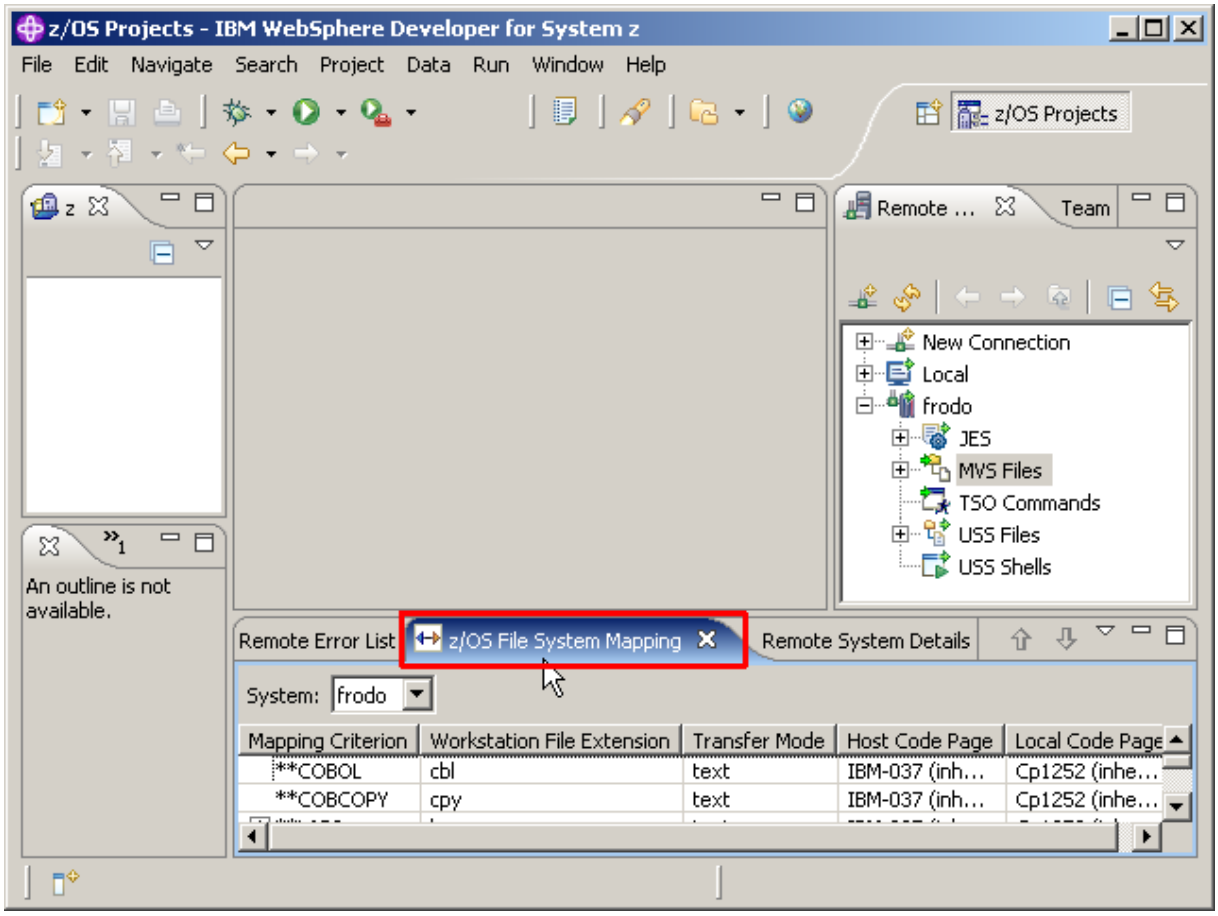


4.Insert Catman as Filter name, leave the defaults for the rest and click Finish.



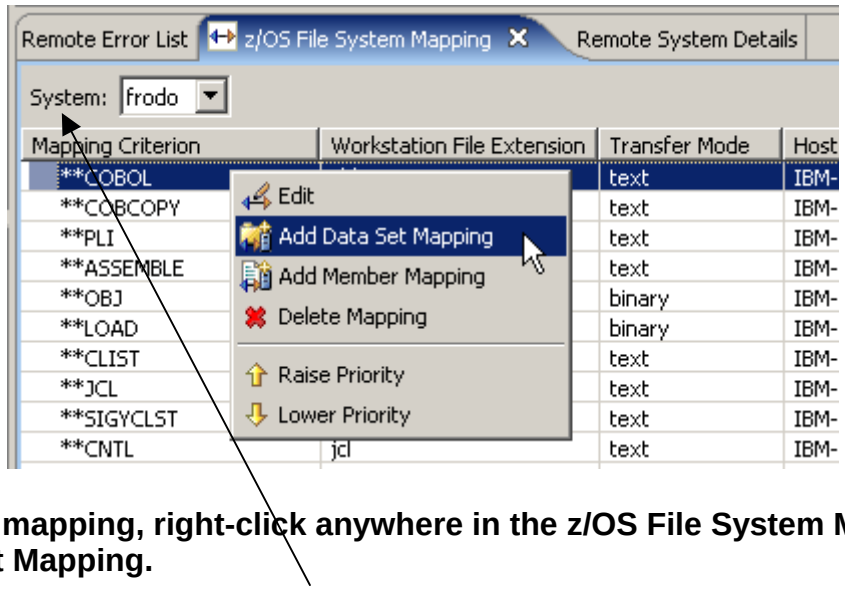
5.In the Remote Systems view expand MVS Files. Now we have Catman under MVS Files. If you expand Catman and DFH310.CICS.SDFHSAMP, wait up to a minute, and you will see many entries.

Note: This is just the beginning...If you scroll down, you'll see the rest.



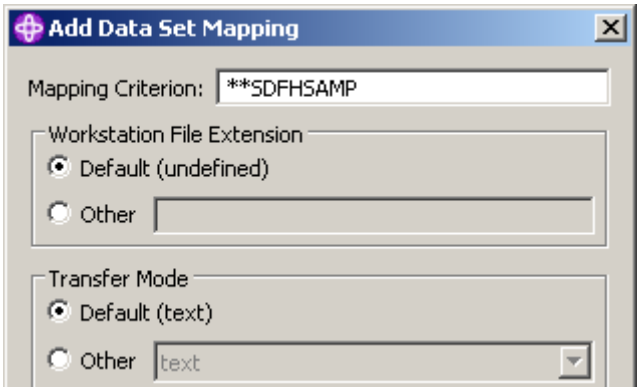
6.Switch to the z/OS File System Mapping view by clicking on the tab.

We need to add mappings to be able to work with the datasets which the Catman filter provides.

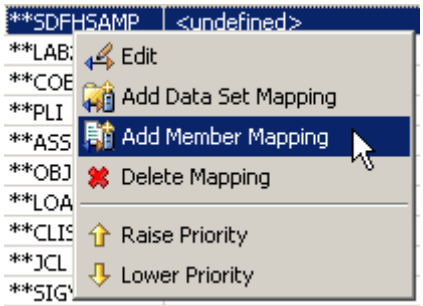


7.To add a data set mapping, right-click anywhere in the z/OS File System Mapping view and select Add Data Set Mapping.

Make sure, you are using the correct z/OS name (frodo in this example).

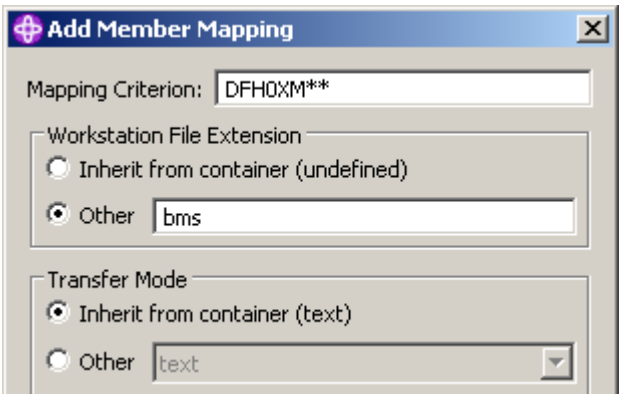


8.On the Add Data Set Mapping panel, specify **SDFHSAMP as Mapping Criterion, leave the defaults for the rest and click OK.

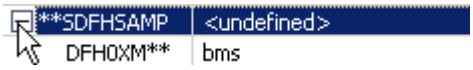


9.In the z/OS File System Mapping view, look for **SDFHSAMP, right-click it, and select Add Member Mapping.

(Note: If you make a mistake, you can click on Delete Mapping and start again).



10.On the Add Data Set Mapping panel, specify DFH0XM as Mapping Criterion, for Workstation File Extension click Other and specify it as bms.**
Click OK.



11.Expand the + on **SDFHSAMP and you will see another mapping criteria under **SDFHSAMP.
Any member starting with DFH0XM will be mapped as bms.**

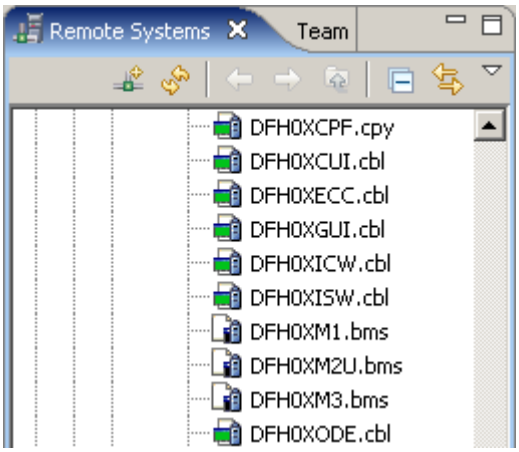
Mapping Type	Mapping Criterion	Workstation File Extension
Data Set Mapping	**SDFHSAMP	Default (undefined)
Member Mapping	DFH0XM**	bms
Member Mapping	DFH0XCP**	cpy
Member Mapping	DFH0XWC**	cpy
Member Mapping	DFH0X**	cbl

12.Now repeat step 10-11 for 3 additional members. Get the Details from the table above.

Note: The filters are prioritized, so make sure to define the most common filter for cbl as the last one.

**SDFHSAMP	<undefined>	text	IBM-037 (inherited)	Cp1252 (inherited)
DFH0XM**	bms	text (inherit...	IBM-037 (inherited)	Cp1252 (inherited)
DFH0XCP**	cpy	text (inherit...	IBM-037 (inherited)	Cp1252 (inherited)
DFH0XW...	cpy	text (inherit...	IBM-037 (inherited)	Cp1252 (inherited)
DFH0X**	cbl	text (inherit...	IBM-037 (inherited)	Cp1252 (inherited)

13. Expand **SDFHSAMP. Your filters should look like shown above



If you look at the Remote Systems view, you will see that all the members DFH0X are marked either .bms, .cpy, or .cbl.**

It may (or may not) be that you see the endings bms, .cpy, or .cbl in the Remote System View not immediately, but only after you have shut down and restarted WDz.

Of course, the files on the mainframe host have not been modified. The client (workstation) component of WDz has been notified that these members are either BMS, Copybook, or COBOL Files. The client component of WDz needs this information so that it knows what to do with them after it has downloaded them from the mainframe host.

Now all the resources we need later in this tutorial have their corresponding file extensions so we can work with them.

3. Generate resources for the inquireSingle Web Service provider

To expose an existing CICS program (the sample application called CICS Catalog Manager) as a Web Service involves the following steps:

1. **Run the CICS Web Services assistant to generate the required files**
2. **Extract the existing program interface**
3. **Run the program LS2WS (interpretative Enablement)**
4. **Customize the service location in the WSDL**

Now, let us run the CICS Web Services assistant to generate the required files

3.1 Extract the existing program interface

CICS TS V3.1 provides the Web Services assistant for generating

- **Web Services definitions (WSDL) from supplied program language structures**
- **language structures from supplied WSDL documents.**

This assistant is actually two utility programs that run in MVS batch. Both programs use the IBM Java SDK.

WDz offers two possibilities to generate the required files:

Interpretive XML Conversion

XML conversion is accomplished using generated metadata and an XML conversion component built into the runtime. While the Interpretive XML Conversion type has limited support for language structure data types and constructs, it requires fewer artifacts and does not involve compilation of XML converters. This conversion type is actually a local implementation of the z/OS batch Web Services Assistant.

Compiled XML Conversion

XML conversion is accomplished using a suite of generated high-level language (HLL) programs. A driver program interacts with the runtime and invokes bundled programs to provide conversion of XML to and from language structures. The compiled XML conversion type provides more extensive support for language structure data types and constructs than the interpretive XML conversion type, however compilation of XML Converters is required and there are more artifacts to manage.

In this step we are generating a new WSDL file from an existing program and will use the interpretive conversion of WDz. As input WDz needs to import the language copybooks that matches the program's COMMAREA for request and response as well as the program to be called.

In this step the INQUIRE SINGLE ITEM function of the central catalog manager program (DFH0XCMN) will be deployed as a Web Service. The interface to this program is a COMMAREA structure which is defined in the supplied COBOL copybook DFH0XCP1.

An excerpt from DFH0XCP1 is shown below:

```
*    Catalogue COMMAREA structure
     03 CA-REQUEST-ID            PIC X(6).
     03 CA-RETURN-CODE           PIC 9(2).
     03 CA-RESPONSE-MESSAGE      PIC X(79).
     03 CA-REQUEST-SPECIFIC      PIC X(911).
*    Fields used in Inquire Catalog
     03 CA-INQUIRE-REQUEST REDEFINES CA-REQUEST-SPECIFIC.
         05 CA-LIST-START-REF        PIC 9(4).
         05 CA-LAST-ITEM-REF         PIC 9(4).
         05 CA-ITEM-COUNT            PIC 9(3).
         05 CA-INQUIRY-RESPONSE-DATA PIC X(900).
         05 CA-CAT-ITEM  REDEFINES CA-INQUIRY-RESPONSE-DATA
                        OCCURS 15 TIMES.
             07 CA-ITEM-REF          PIC 9(4).
             07 CA-DESCRIPTION       PIC X(40).
             07 CA-DEPARTMENT        PIC 9(3).
             07 CA-COST              PIC X(6).
             07 IN-STOCK             PIC 9(4).
             07 ON-ORDER             PIC 9(3).
*    Fields used in Inquire Single
     03 CA-INQUIRE-SINGLE REDEFINES CA-REQUEST-SPECIFIC.
         05 CA-ITEM-REF-REQ          PIC 9(4).
         05 FILLER                   PIC 9(4).
         05 FILLER                   PIC 9(3).
         05 CA-SINGLE-ITEM.
             07 CA-SNGL-ITEM-REF     PIC 9(4).
             07 CA-SNGL-DESCRIPTION  PIC X(40).
             07 CA-SNGL-DEPARTMENT   PIC 9(3).
             07 CA-SNGL-COST         PIC X(6).
             07 IN-SNGL-STOCK        PIC 9(4).
             07 ON-SNGL-ORDER        PIC 9(3).
         05 FILLER                   PIC X(840).
*    Fields used in Place Order
     03 CA-ORDER-REQUEST REDEFINES CA-REQUEST-SPECIFIC.
         05 CA-USERID                PIC X(8).
         05 CA-CHARGE-DEPT           PIC X(8).
         05 CA-ITEM-REF-NUMBER       PIC 9(4).
```

Notice that the CA-REQUEST-SPECIFIC field is re-defined by fields CA-INQUIRE-REQUEST, CA-INQUIRE-SINGLE and CA-ORDER-REQUEST. In essence this copybook defines three separate interfaces for the INQUIRE CATALOG, INQUIRE SINGLE ITEM, and the PLACE ORDER functions, which are overlaid on one another.

However, the DFHLS2WS utility does not support the REDEFINES statement. So to use DFHLS2WS we need to create a new copybook that has only those elements that relate to the inquire single function. This has been done in the DFH0XCP4 copybook.

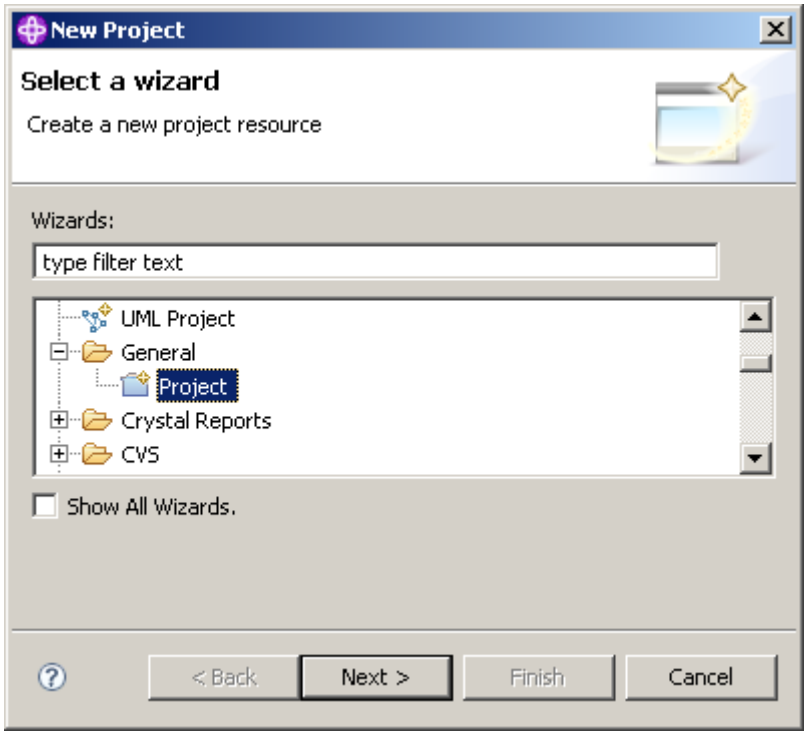
An excerpt from DFH0XCP4 is shown below (strictly speaking the “DISPLAY” keywords are not required and can be ignored):

```
*    Catalogue COMMAREA structure
     03 CA-REQUEST-ID            PIC X(6).
     03 CA-RETURN-CODE           PIC 9(2) DISPLAY.
     03 CA-RESPONSE-MESSAGE      PIC X(79).
*    Fields used in Inquire Single
     03 CA-INQUIRE-SINGLE.
         05 CA-ITEM-REF-REQ          PIC 9(4) DISPLAY.
         05 FILLER                   PIC 9(4) DISPLAY.
         05 FILLER                   PIC 9(3) DISPLAY.
         05 CA-SINGLE-ITEM.
             07 CA-SNGL-ITEM-REF     PIC 9(4) DISPLAY.
             07 CA-SNGL-DESCRIPTION  PIC X(40).
             07 CA-SNGL-DEPARTMENT   PIC 9(3) DISPLAY.
             07 CA-SNGL-COST         PIC X(6).
             07 IN-SNGL-STOCK        PIC 9(4) DISPLAY.
             07 ON-SNGL-ORDER        PIC 9(3) DISPLAY.
```

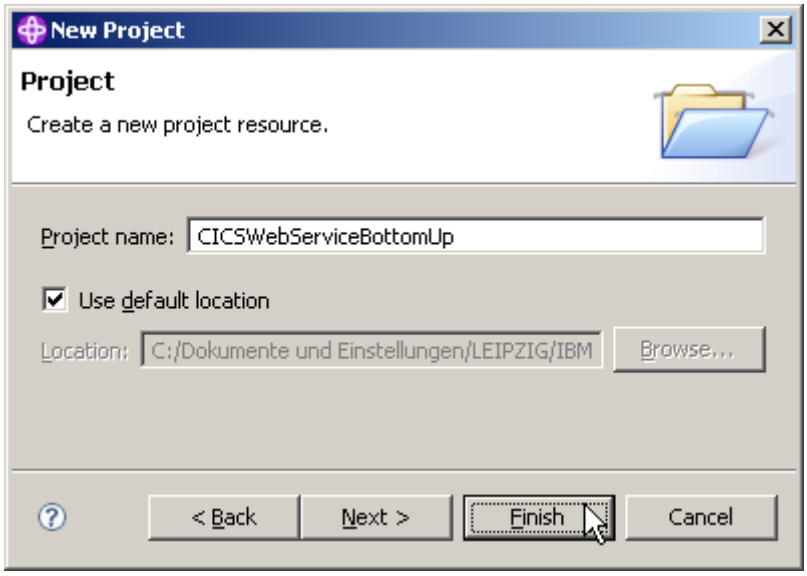
For Web Service enablement using WDz the required resources have to be transferred to your workstation.

14.First of all, verify that the z/OS Projects perspective is active. Furthermore, ensure that the navigator view is available. (If not, open it through the menu bar Window → Show view → Other...
and select General → Navigator).

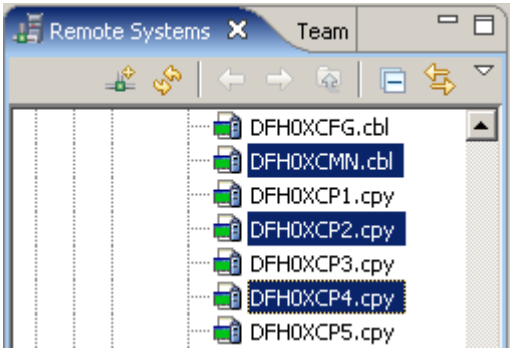
15.On the menu bar, select File → New → Project…



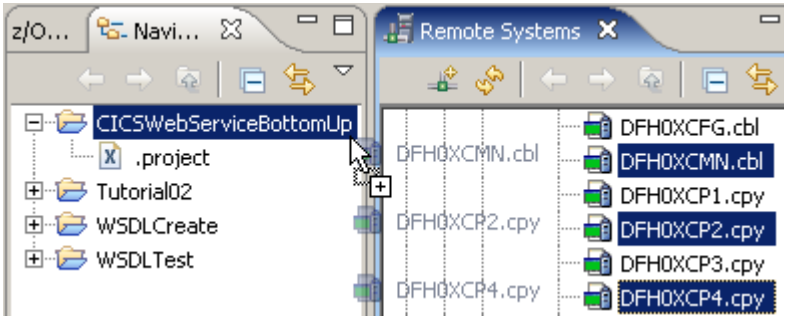
16.Expand General and select Project.
Click the Next button.



17.On the New Project panel, specify the Project name as CICSWebServiceBottomUp, ensure Use default location is checked, then click the Finish button.

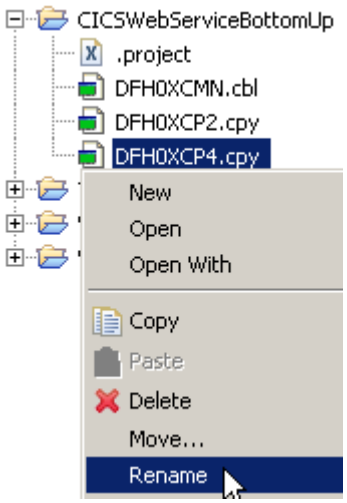


18.In the Remote Systems view, expand the Catman MVS Files filter that you created earlier. Scroll down and select DFH0XCMN.cbl, DFH0XCP2.cpy and DFH0XCP4.cpy while holding down the CTRL key.

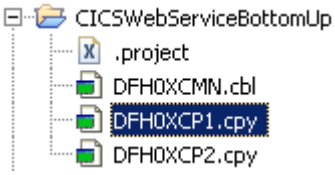


19.Copy the selected files per drag&drop to your freshly created CICSWebServiceBottomUp project.

Just drag and drop the 3 files to CICSWebServiceBottomUp in the Navigator pane.

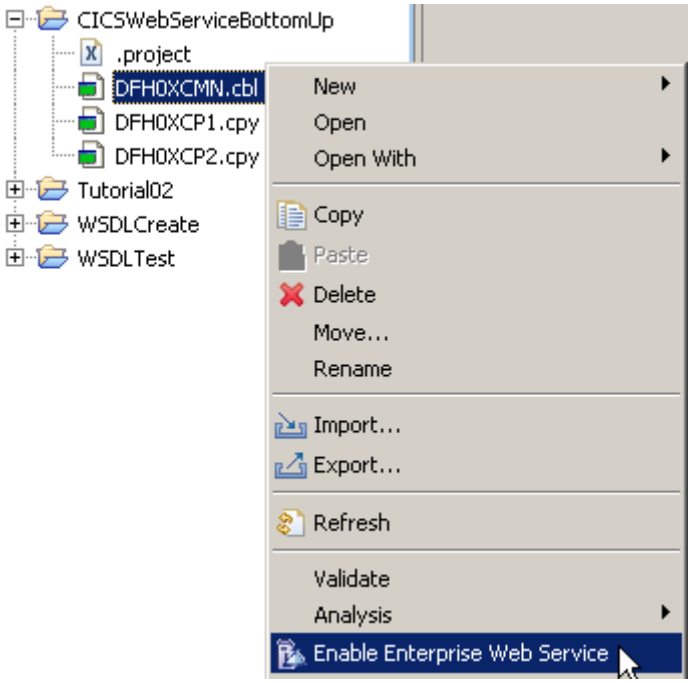


20.As DFH0XCMN expects a copybook called DFH0XCP1 (which we transformed to DFH0XCP4 because of the redefine) rename DFH0XCP4.cpy to DFH0XCP1.cpy.
Right-click on DFH0XCP4.cpy and select Rename.



21.Change the name from DFH0XCP4 to DFH0XCP1.cpy and hit Enter.

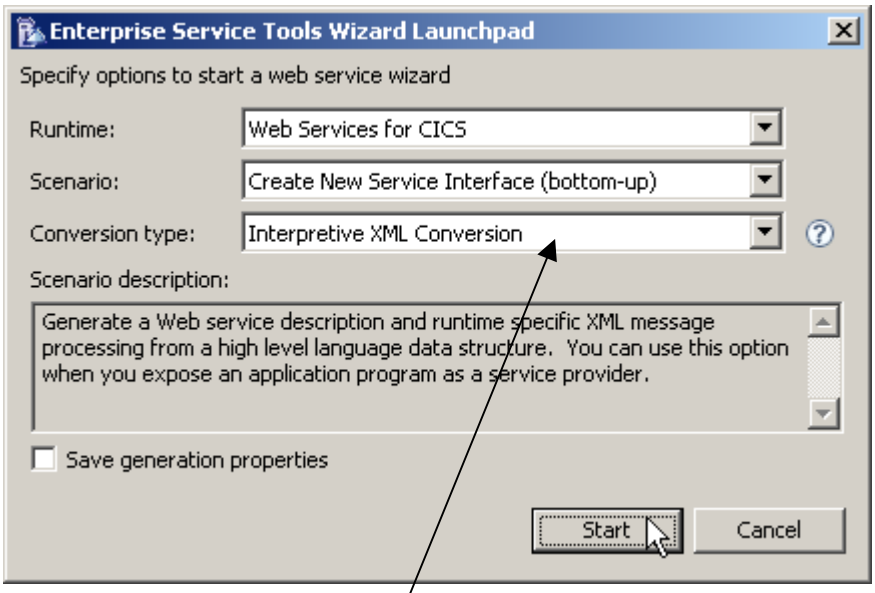
3.2 Run the interpretive conversion



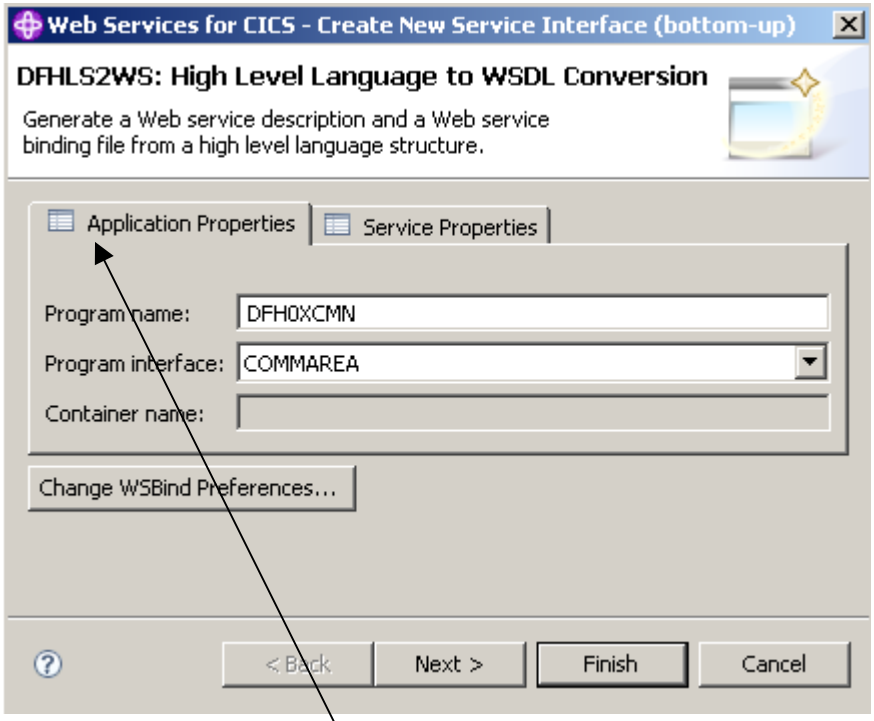
22.In the Navigator pane, right-click DFH0XCMN.cbl and select Enable Enterprise Web Service.

Option	Value
Runtime	Web Services for CICS
Scenario	Create New Service Interface (bottom-up)
Conversion Type	Interpretive XML Conversion

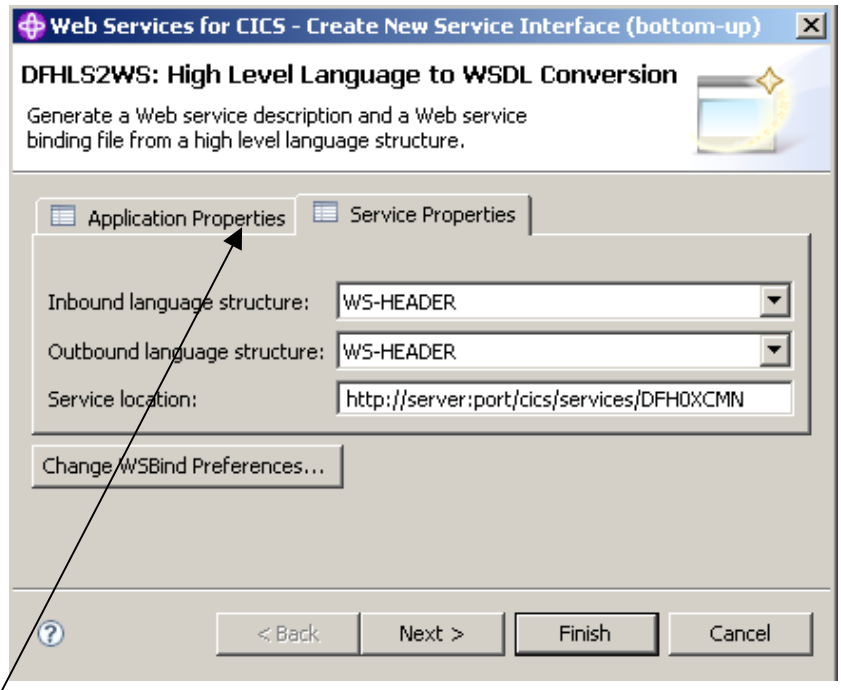
23.For the Enterprise Service Tools Wizard Launchpad panel specify the values above and …



enter values like shown above, esp. Interpretive XML Conversion. Then click Start.



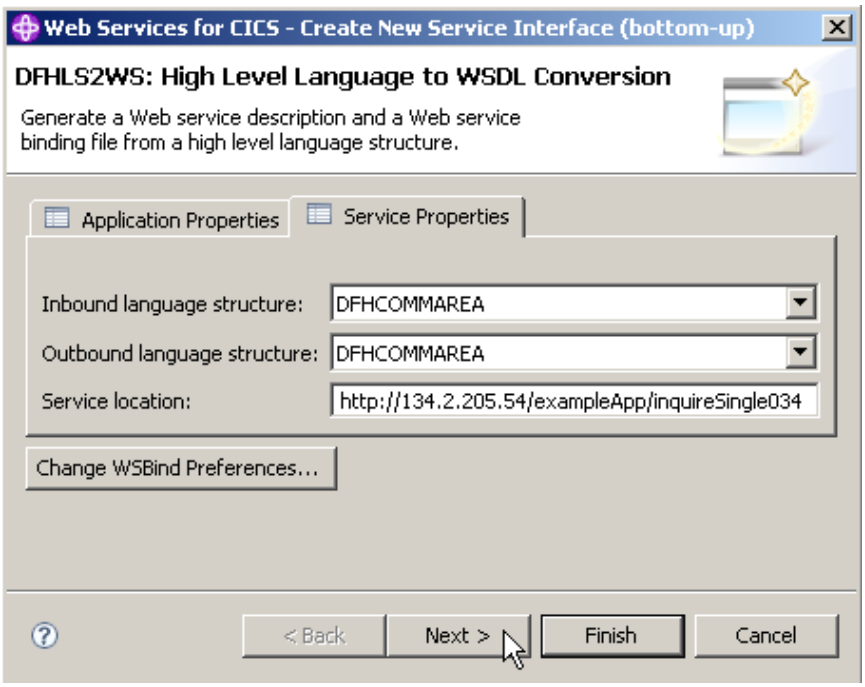
24.Now you are at the DFHLS2WS panel.
Leave the defaults for the options on the Application Properties tab.



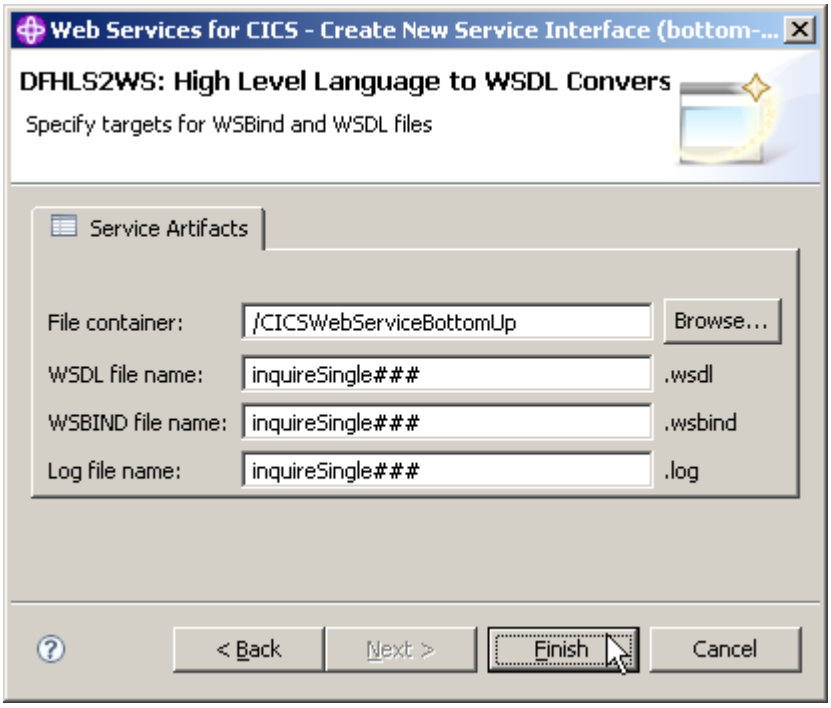
On the Service Properties tab, …

Inbound language structure	**DFHCOMMAREA**
Outbound language structure	**DFHCOMMAREA**
Service location	**http://134.2.205.54/exampleApp/inquireSingle<Team_Number>** **for example if your user-ID is PRAK034:** **http://134.2.205.54/exampleApp/inquireSingle034**

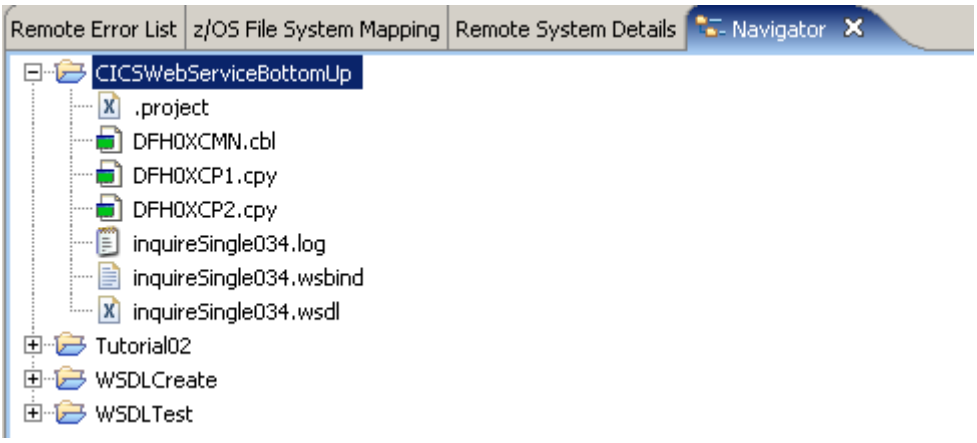
specify the values above and



click Next:



25.On the next panel, leave the default for the File container and name all service artefacts inquireSingle###
(Replace ### with your UserID).
Click Finish.



26.Your project should now look like shown above. If you are curious, you can double klick on inquireSingle034.log or inquireSingle034.wsd to examine the files.

3.3 Review the output

Review the output in your workspace, in addition to the wsbind and wsdl file a log was generated. These files fulfill the following tasks:

- **Log generated file**
- **The log file contains detailed information on the steps the job has taken to create the output files. The contents of the log file are not normally required by end users. However, it does contain a copy of the generated WSDL which can be used if the generated WSDL file is misplaced or could be used in problem determination.**

- **WSDL generated file**
- **The WSDL file defines all the information required for a client to access this Web Service. It contains an XML schema representation of the request and response and location information for the service. As generated, the only part of the WSDL file that needs to be customized before it can be published to clients is the service TCP/IP address and port.**

- **WSBIND generated file**
- **The WSBIND file contains the meta-data that CICS uses at runtime to marshal and de-marshal the request and response messages in XML into the format expected by the application program. The WSBIND file is read by CICS when a WEBSERVICE resource is installed onto a PIPELINE (it is not read on every request).**

It is important to ensure that the version of the WSBIND file matches the WSDL file used by clients. If changes are made to the language structure, the WSDL and WSBIND file need to be re-generated, and the new WSDL sent to the client.

If the PIPELINE scanning mechanism is used to install a WEBSERVICE, the resource name will be based on the name of the WSBIND file. It is suggested to keep the names of the WSBIND file and the WSDL file the same to ensure installed Web Services and clients match.

(You can optionally perform the compiled conversion. Note that in addition to WSBIND and WSDL a COBOL converter program will be generated.)

You generated all required resources, now you can go on and install them in CICS.

4. Prepare HFS

To provide a web service, CICS requires some HFS (Hierarchical File System in Unix System Services) directories where it will look for the WSBIND and other files required. This part will demonstrate how to create the required directories and place all necessary files at the right position.

4.1 Create the HFS directories

To expose a CICS program as a Web Service over HTTP, a TCPIPService and PIPELINE resources are required. The PIPELINE resource definition points to some HFS directories and also points to a pipeline configuration file.

The setup requires configuring files and directories in the Hierarchical File System (HFS).

- **Shelf directory: /var/cicsts/**
  The shelf directory is used by CICS to store the Web Service binding files that are associated with WEBSERVICE resources. Each WEBSERVICE resource is associated with a PIPELINE. The shelf directory is managed by CICS together with the PIPELINE resource. You should not modify the content in this directory. Several PIPELINEs can use the same shelf directory because CICS ensures a unique directory structure beneath the shelf directory for each PIPELINE. Since CICS writes to this directory, CICS requires appropriate security permissions.
  This directory has already been created for you.

- **Pickup directory: /tutorial06/wspickup/provider:**
  The pickup directory contains the Web Service binding files that are to be associated with the PIPELINE. When a PIPELINE is installed or in response to a PERFORM PIPELINE SCAN command, CICS searches this directory for files ending in .wsbind. For each .wsbind-file, CICS dynamically creates a WEBSERVICE and URIMAP resource, associates it with the PIPELINE and installs them ready for use.

A Uniform Resource Identifier (URI) is a compact string of characters used to identify or name a resource on the Internet.

URIMAP definitions are resource definitions that match the URIs of HTTP or Web service requests, and provide information on how to process the requests.

Message handlers and pipelines

- **A message handler is a program in which you can perform your own processing of Web service requests and responses.**
- **A pipeline is a set of message handlers that are executed in sequence.**

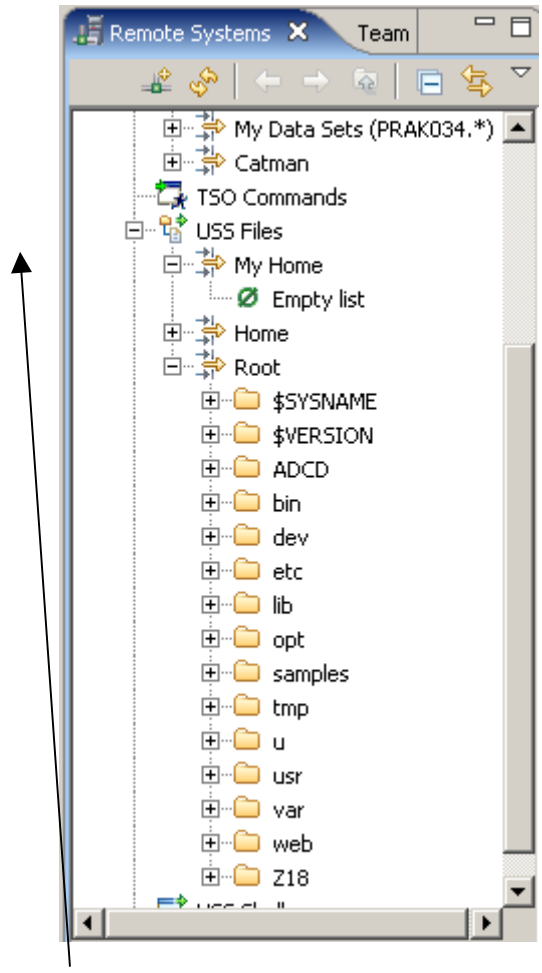
WSDL and the application data structure

A Web service description contains abstract representations of the input and output messages used by the service. CICS uses the Web service description to construct the data structures used by application programs. At run time, CICS performs the mapping between the application data structures and the messages.

The Web service binding file

The Web service binding file contains information that CICS uses to map data between input and output messages, and application data structures.

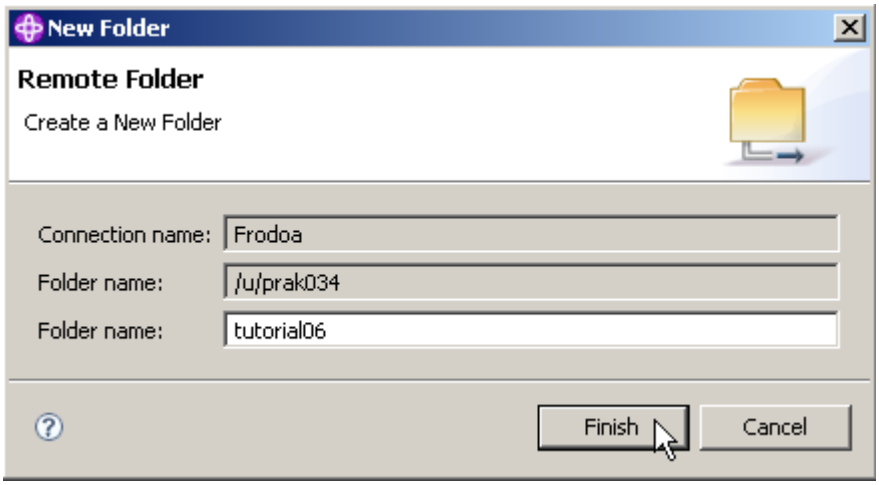
For additional Details see Appendix.



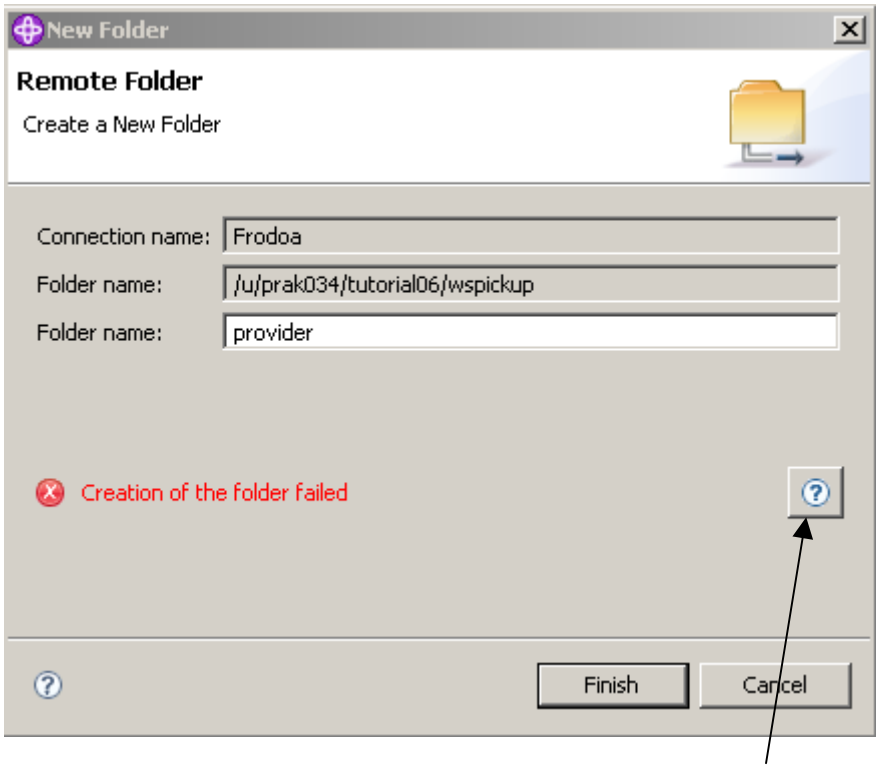
In your Remote Systems view expand USS Files and then My Home and Root.

My Home is still empty. Root shows the subdirectories usually found in a Unix system. Close both My Home and Root again.

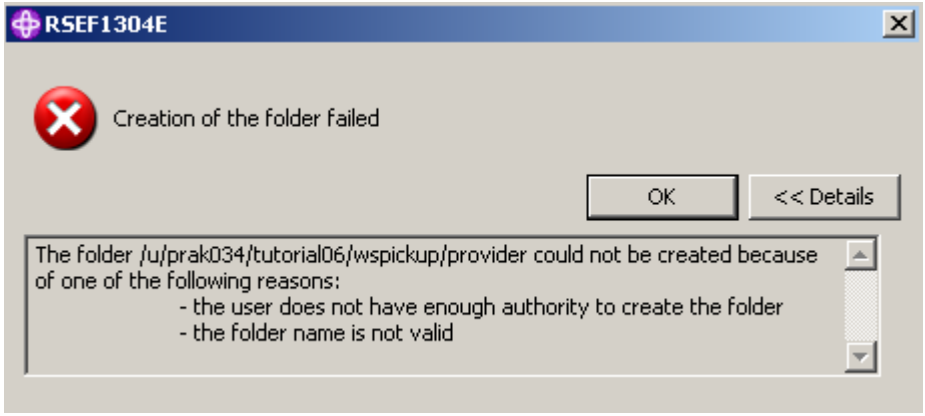
27.Create the pickup directory by right-clicking the My Home filter in the USS Files of your Remote System Connection and select New → Folder.



28.Name the folder tutorial06 and click Finish.

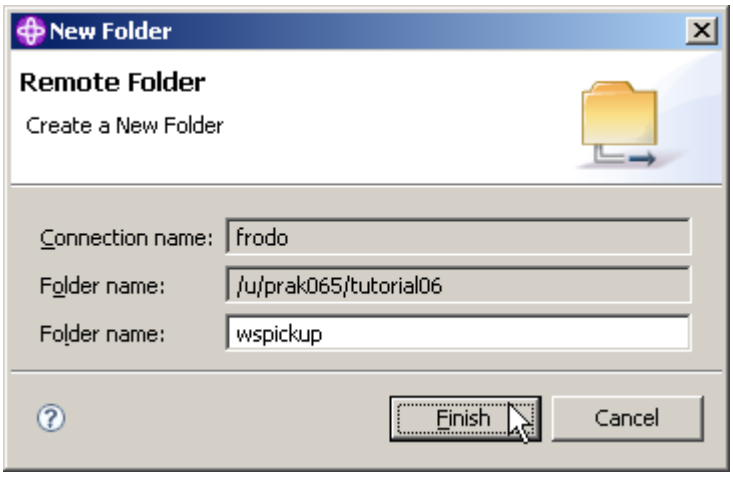


Sometimes you get an error message like shown above. Klick here for details



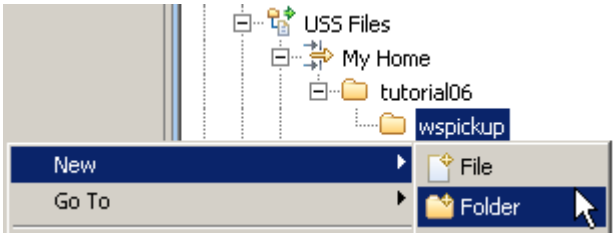
In this case, disconnect Frodo, reconnect again, and try again.

29.Assuning you are successful, right-click the tutorial06 folder to create a sub folder. Select New → Folder.



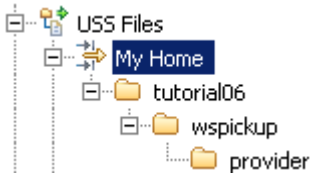
30.Name the folder wspickup and click Finish.

Note, that your Folder name will differ the one from above.



31.Right-click the wspickup folder to create a sub folder. Select New → Folder.

32.Name the folder provider and click Finish.



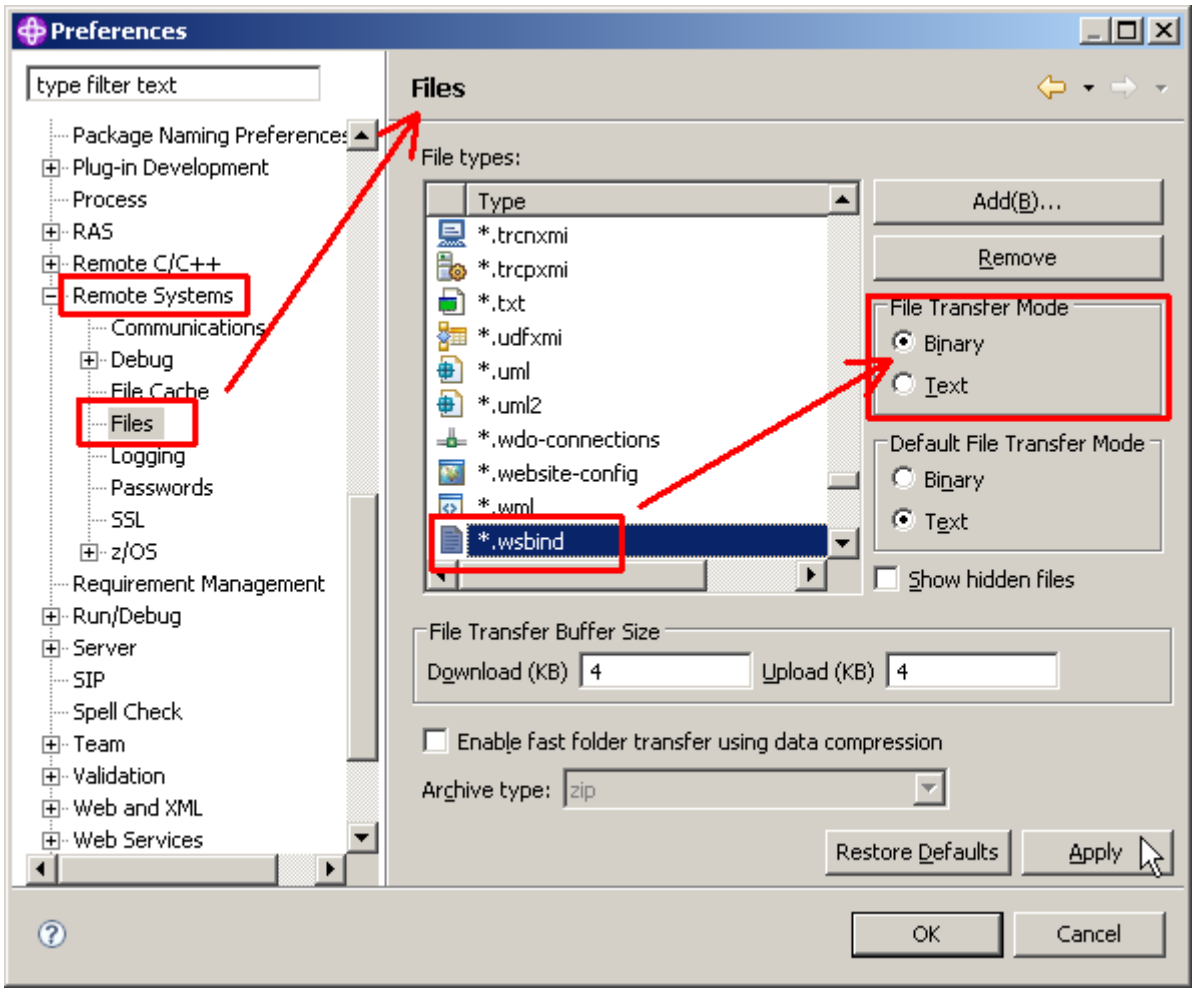
33.Your My Home filter should now look like that:

You just created the sub folders /tutorial06/wspickup/provider in your USS-Home directory.

You have added a directory called provider for the pickup directory because in this tutorial we will define a PIPELINE for use when CICS is a service provider. In a later lab, we may define a PIPELINE that contains information to be used when CICS is a Web Service requester. You will need to specify a different PIPELINE for use when CICS is a service requester and provider. Separate PIPELINE definitions are needed because a PIPELINE definition can only reference one pipeline configuration file. The pipeline configuration files must be different for processing inbound Web Service requests versus outbound service requests.

4.2 Copy WSBIND file to the Pickup Directory

In this part we will copy a WSBIND file from your workstation to the Pickup directory on z/OS. We could use several different file transfer techniques, but we will use WDz.



34.Now we have to verify that copying a .wsbind file from the workstation to z/OS will perform a binary transfer.

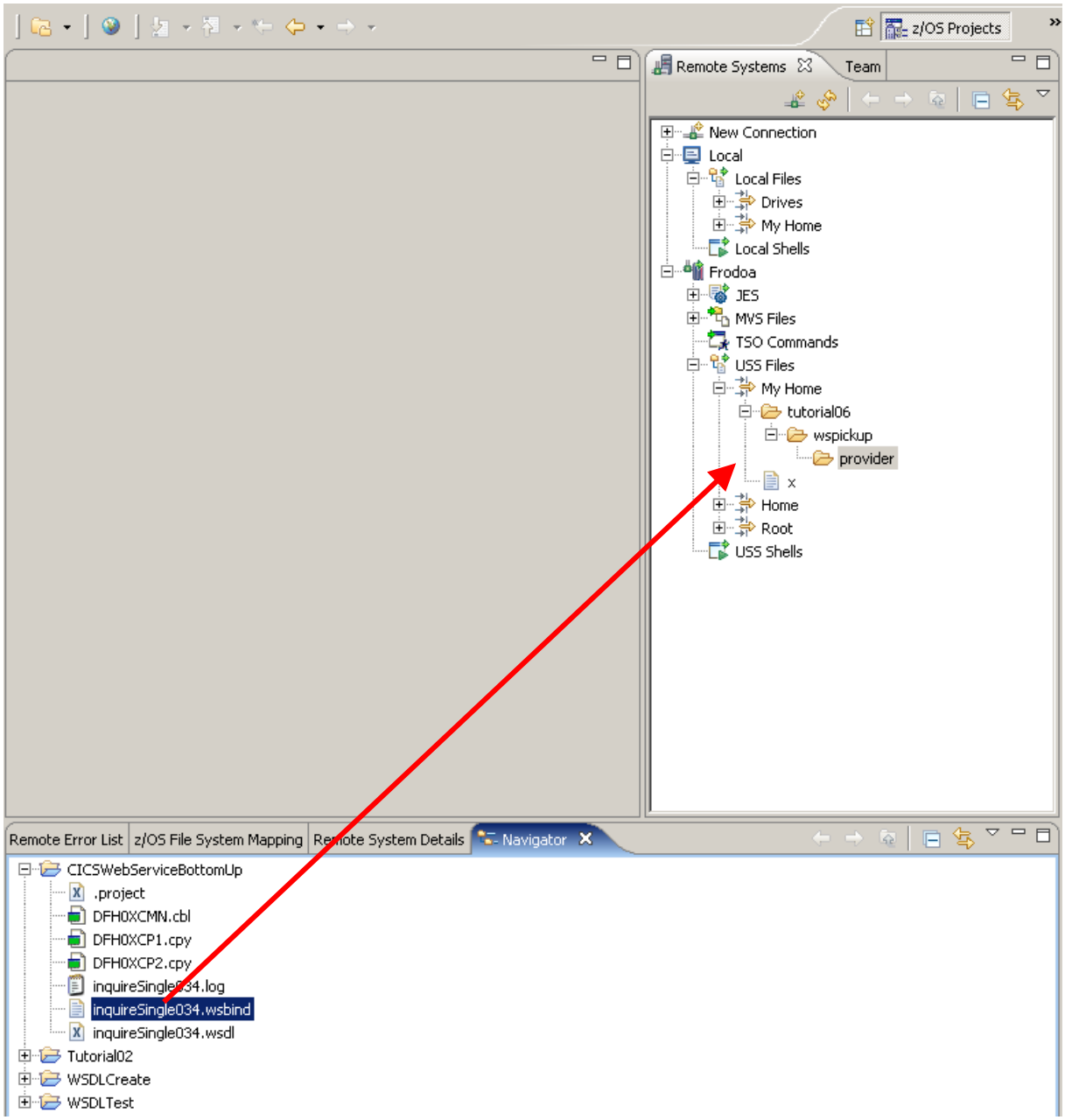
From the menu bar select Window → Preferences.

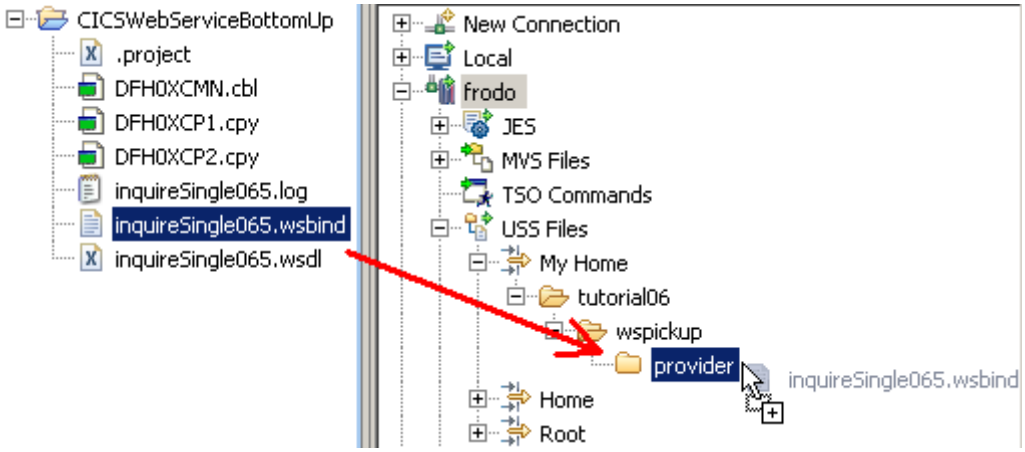
In the Preferences dialog, on the left, select Remote Systems → Files.

On the right, scroll down and click on *.wsbind.

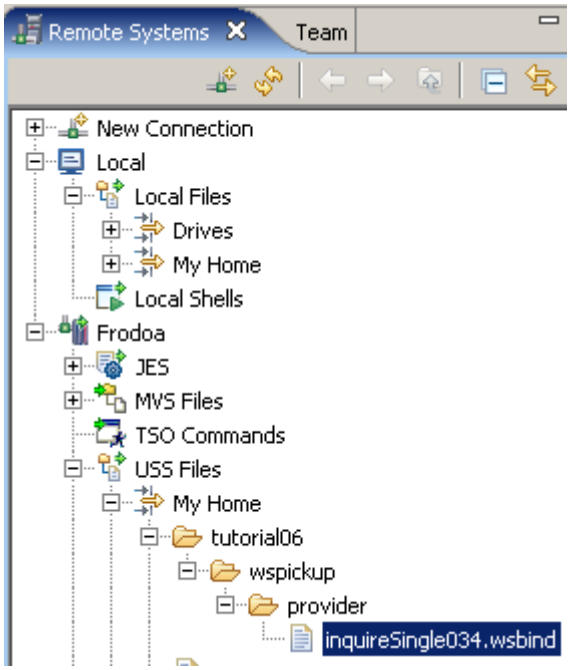
Under File Transfer Mode, click the Binary radio button.

Click the Apply button, then click the OK button.

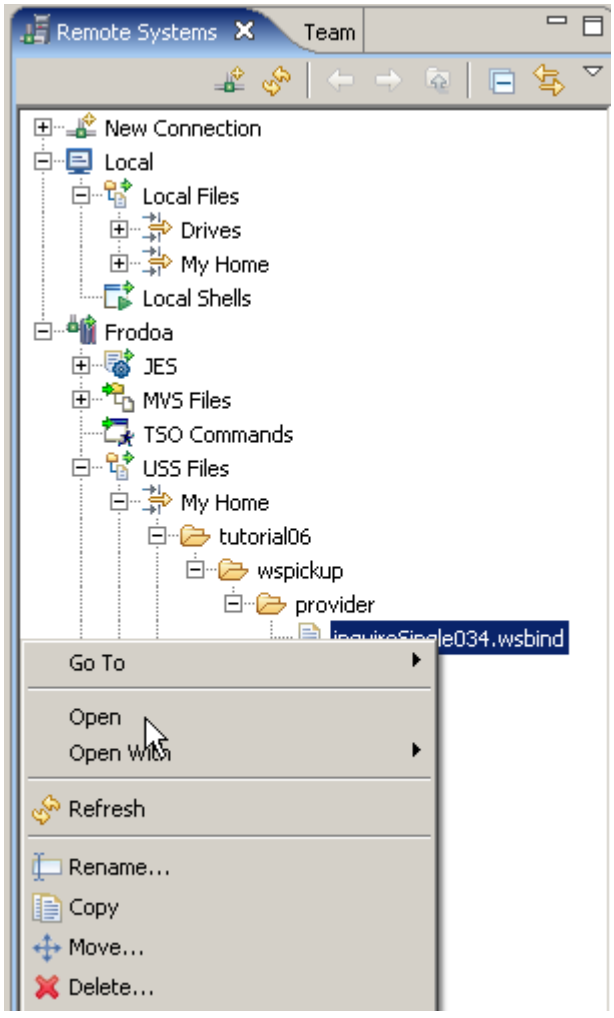




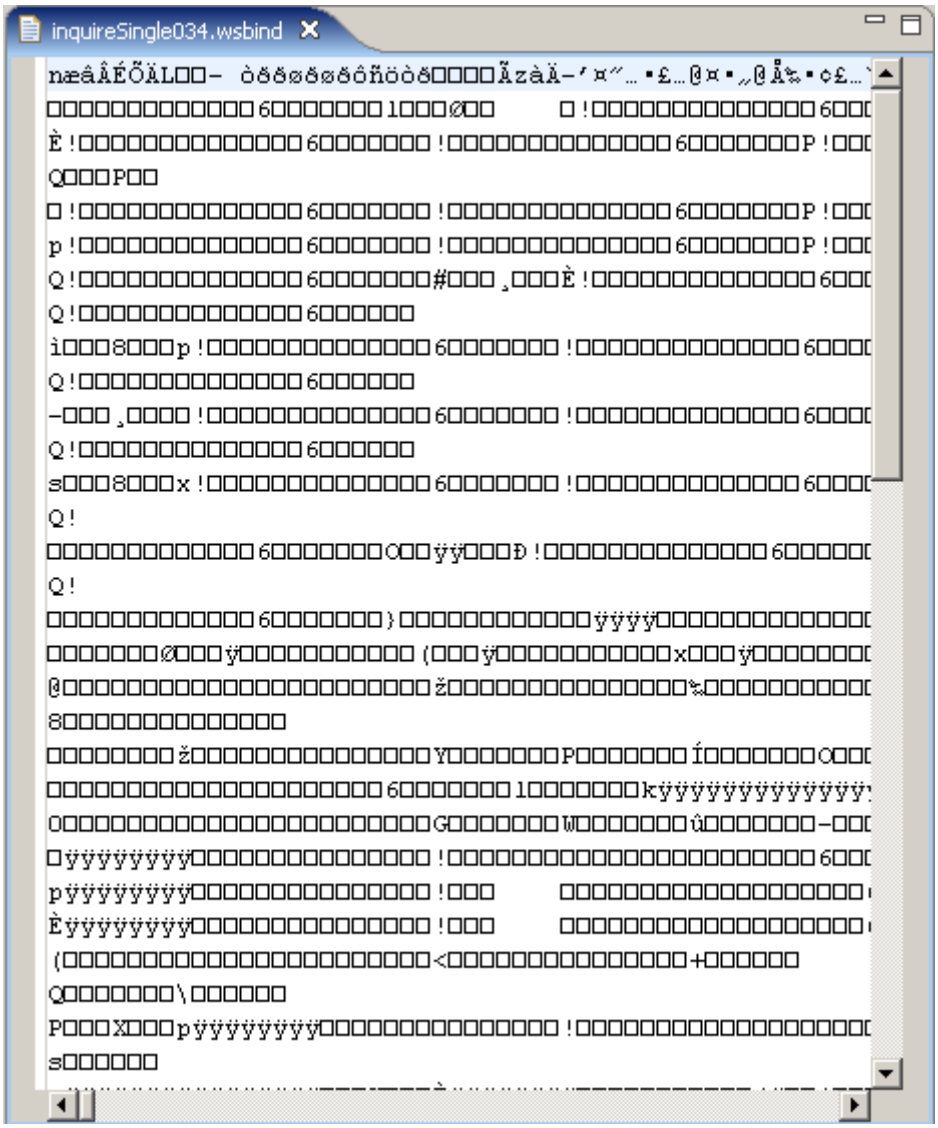
35.In the Navigator view, select the wsbind file inquireSingle###.wsbind you created earlier and drag or copy it to your Remote System USS folder /tutorial06/wspickup/provider.



36.The wsbind-file should now be on your remote system.



To verify the transfer worked, in the Remote Systems view, rightklick on inquireSingle034.wsbind and klick on Open.



You will see something unintelligable in the editor window. This is the proof, something has been copied.

You can also examine this file using TSO. For this, disconnect Frodo in the Remote Systems view.

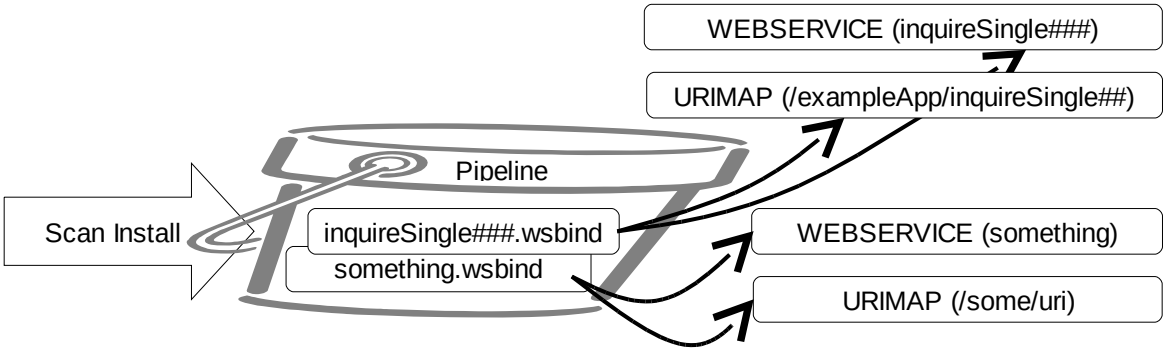
by logging in with your 3270 emulator, going to OMVS, and accessing in the directory `/u/prak034` the subdirectory `tutorial06/wspickup/provider/`.

You can use the `oedit` command to open the the file inquireSingle034.wsbind to see the text.

See Appendix B for details.

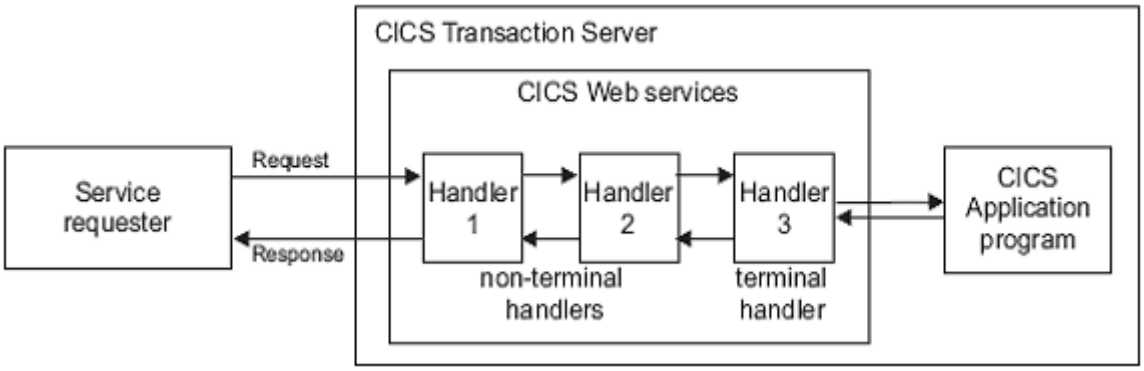
5. Prepare CICS

As a Systems Programmer, the TCPIPService and PIPELINE definition are the only definitions you need if you are using the WSBIND mechanism. A PIPELINE is associated with a pickup directory we specified earlier. When a PIPELINE is installed or in response to a PERFORM PIPELINE SCAN command, CICS searches this pickup directory for files ending with ".wsbind". For each .wsbind file, CICS dynamically creates a WEBSERVICE and URIMAP resource, associates them with the PIPELINE and installs them ready for use.



5.1 Create a PIPELINE Resource Definition

A PIPELINE tells CICS how to process a request. Multiple programs can look at and perform functions on the request body and headers. This type of program is called a message handler or a header handler. Ultimately a program will process the request. The PIPELINE definition points to a PIPELINE configuration file that specifies the name of the programs that will work on the body and headers, and will process the request



The PIPELINE definition is in XML format and details the message handlers that will act on Web service requests and responses as they pass through the pipeline. It specifies the name of the programs that will work on the body and headers, and will process the request.

The configuration of a pipeline used to handle a Web service request is specified in an XML document, known as a pipeline configuration file.The pipeline configuration file is stored in the z/OS UNIX System Services hierarchical file system (HFS), and its name is specified in the CONFIGFILE attribute of a PIPELINE resource definition. When you work with configuration files, ensure that the character set encoding is US EBCDIC (Code page 037).

There are two kinds of pipeline configuration:

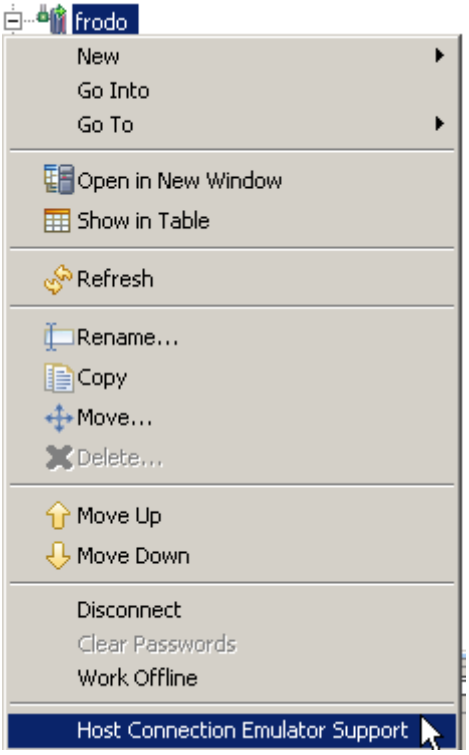
- **one describes the configuration of a service provider pipeline;**
- **the other describes a service requester pipeline.**

Each is defined by its own schema, and each has a different root element.

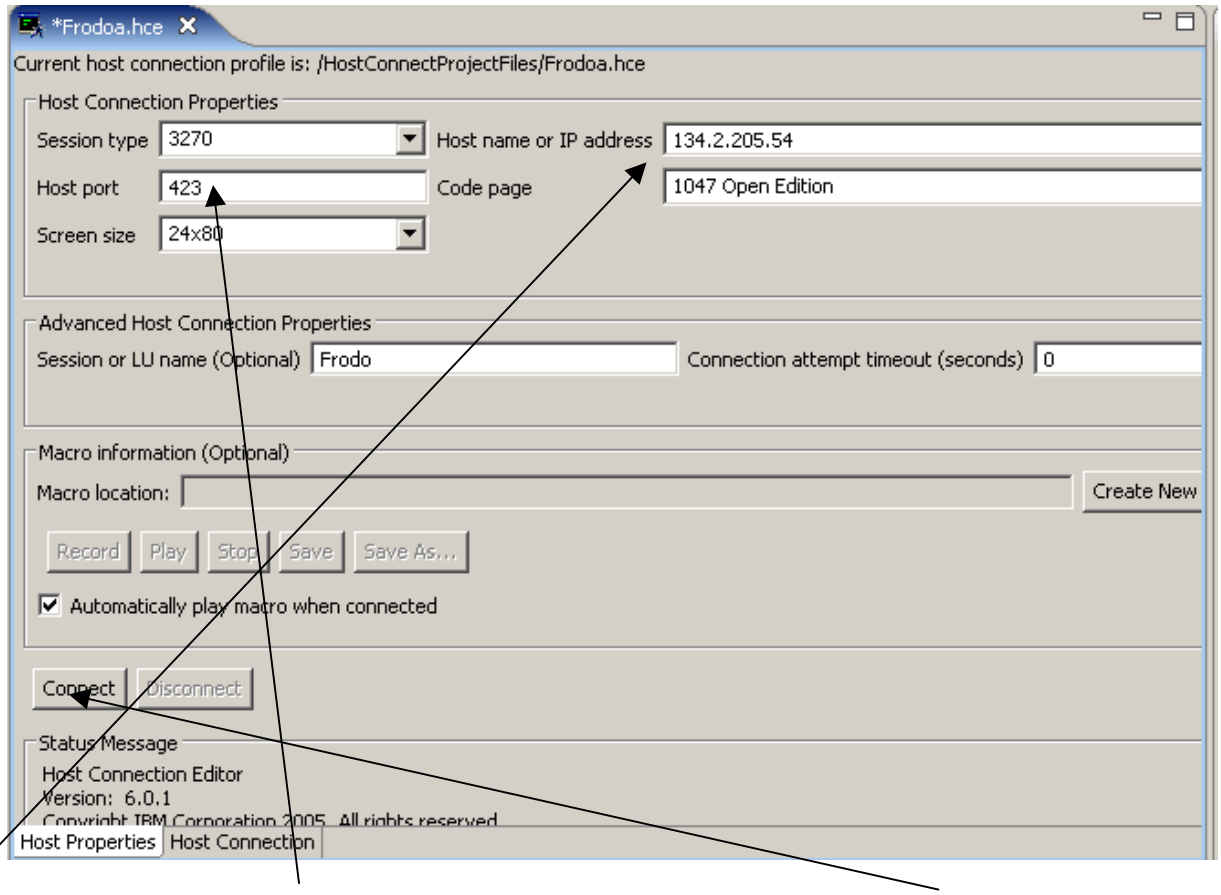
Pipeline	Schema	Root element
Service provider	`Provider.xsd`	`<provider_pipeline>`
Service requester	`Requester.xsd`	`<requester_pipeline>`

We are going to use the CICS supplied PIPELINE configuration file for a SOAP V1.1 handler.

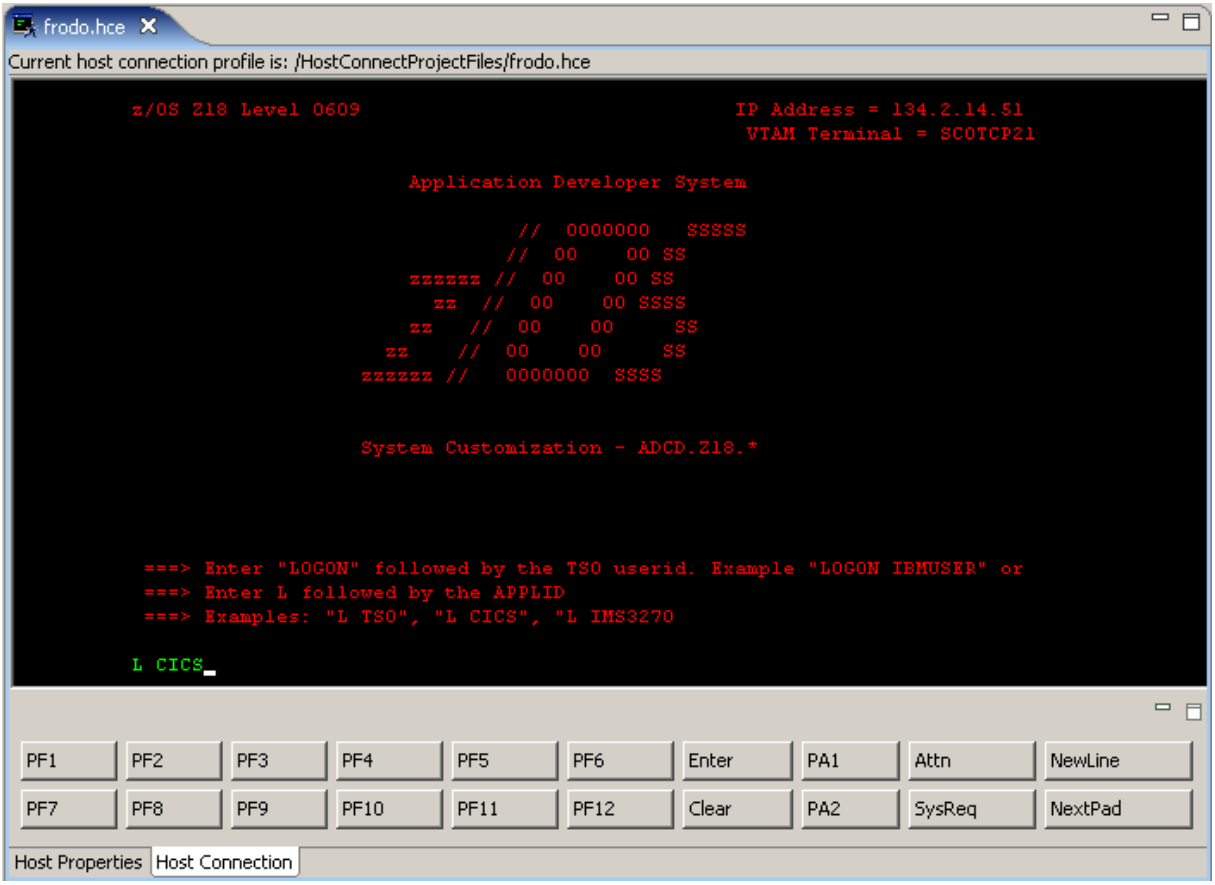
It is common to process headers and messages the same for multiple Web Services. This means that one pipeline configuration file can work with multiple Web services. This also means that as a CICS System programmer, we don’t need to make all these resource definitions and configuration files for each Web service. We can make a PIPELINE definition that will perform common header and message handling and also point to a pickup directory.



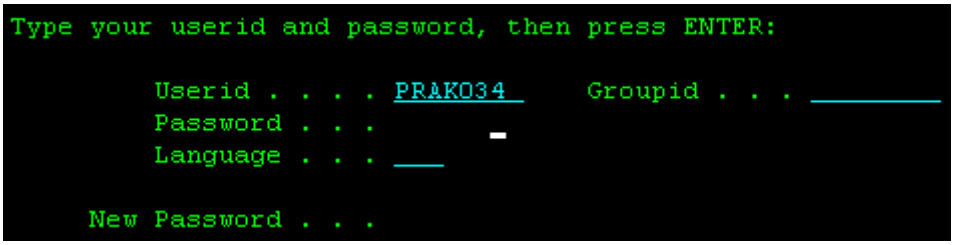
37.Right-click your Remote Systems connection frodo and select Host Connection Emulator Support to start a 3270 emulator.



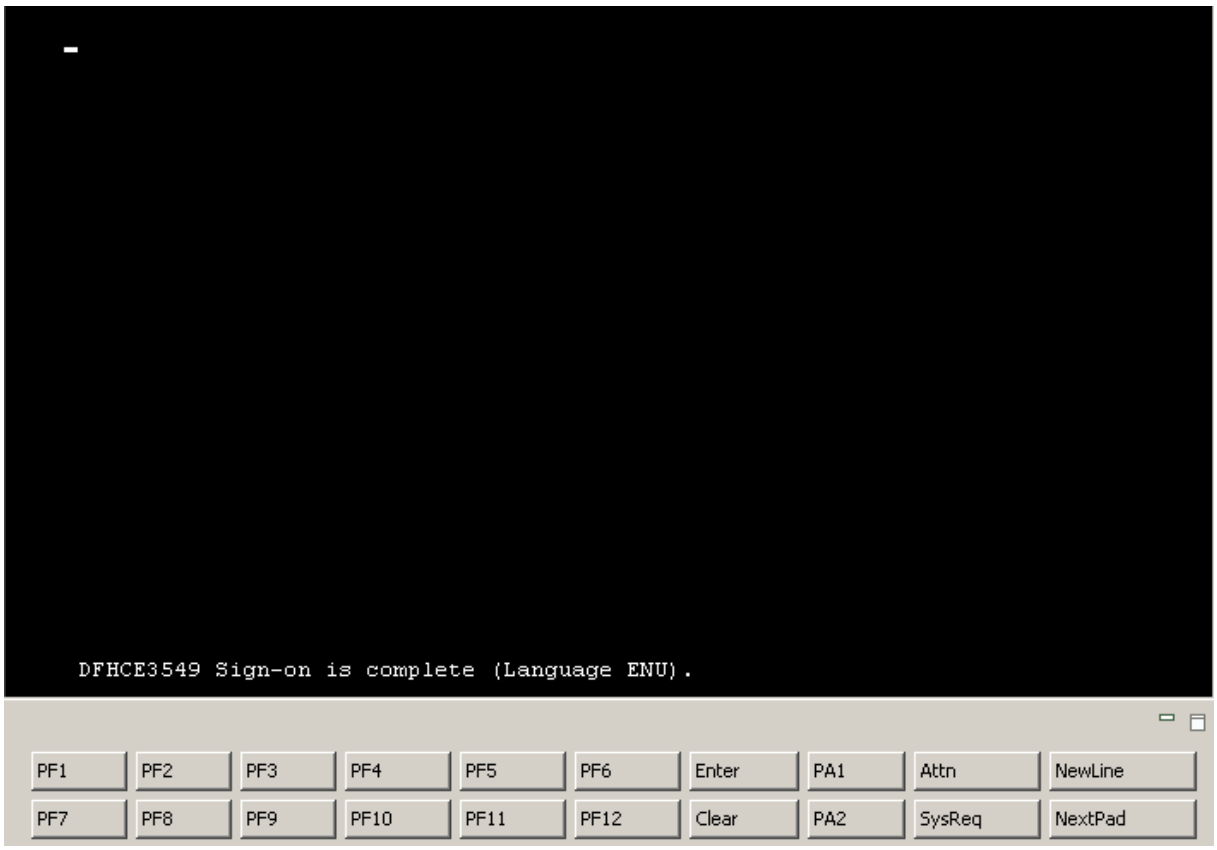
134.2.205.54 benötigt port 423 (normally this is port 23). Click on Connect .



38.Type L CICS to log in to the CICS Subsystem and press the Return key.



39.Login using your Userid and password and press the Enter key .
(Replace ### with your UserID)



The user friendly CICS Welcome screen greets you. You better know, the command line starts in the left upper corner at the cursor position.

```
CEDA DEFINE GROUP(PRAK###) PIPELINE(PIPE###)
```

```
CEDA DEFINE GROUP(PRAK034) PIPELINE(PIPE034)
```

40.When you are logged in to CICS, type (at the cursor position)
CEDA DEFINE GROUP(PRAK###) PIPELINE(PIPE###)
and press the Enter key (Replace ### with your UserID).

Attribute	Value
Pipeline	**PIPE###** **(Replace ### with your UserID)**
Group	**PRAK###** **(Replace ### with your UserID)**
Description	**Basic SOAP 1.1 provider pipeline**
Status	**Enabled**
Configfile	**/usr/lpp/cicsts/cicsts31/samples/pipelines/basicsoap11provider.xml**
Shelf	**/var/cicsts/**
WSDir	**/u/prak###/tutorial06/wspickup/provider/** **(Replace ### with your UserID)**

Basic SOAP 1.1 provider pipeline
/usr/lpp/cicsts/cicsts31/samples/pipelines/basicsoap11provider.xml
(extends to next line of 3270 screen)
/var/cicsts/
/u/prak034/tutorial06/wspickup/provider/

```
 OVERTYPE TO MODIFY                                          CICS RELEASE = 0640
  CEDA  DEFine PIpeline( PIPE034  )
   PIpeline        : PIPE034
   Group           : PRAK034
   Description  ==> BASIC SOAP 1.1 PROVIDER PIPELINE
   STatus       ==> Enabled            Enabled | Disabled
   Configfile   ==> /usr/lpp/cicsts/cicsts31/samples/pipelines/basicsoap11prov
   (Mixed Case) ==> ider.xml
                ==>
                ==>
                ==>
   SHelf        ==> /var/cicsts/
   (Mixed Case) ==>
                ==>
                ==>
                ==>
   Wsdir           : /u/prak034/tutorial06/wspickup/provider/
   (Mixed Case)    :
+                  :
  S Object already exists in this group.
                                                  SYSID=CICS APPLID=CICS
  DEFINE UNSUCCESSFUL                        TIME:  10.59.25  DATE: 08.230
PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

41.Enter the attributes shown above .
Note that you may have to scroll down using the F8 key to be able to insert values to the WSDir-field.

Press the Enter key.

```
 DEFINE GROUP(PRAK###) PIPELINE(PIPE###)
 OVERTYPE TO MODIFY                                        CICS RELEASE = 0640
  CEDA  DEFine PIpeline( PIPE065  )
   PIpeline      ==> PIPE###
   Group         ==> PRAK###
   Description   ==> Basic SOAP 1.1 provider pipeline
   STatus        ==> Enabled            Enabled | Disabled
   Configfile    ==> /usr/lpp/cicsts/cicsts31/samples/pipelines/basicsoapllprov
   (Mixed Case)  ==> ider.xml
                 ==>
                 ==>
                 ==>
   SHelf         ==> /var/cicsts/
   (Mixed Case)  ==>
                 ==>
                 ==>
                 ==>
   Wsdir           :
   (Mixed Case)    :
+                  :
  S CONFIGFILE MUST BE SPECIFIED.
                                                  SYSID=CICS APPLID=CICS

PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

(Screenshot before the scroll-down)

```
 OVERTYPE TO MODIFY OR PRESS ENTER TO EXECUTE              CICS RELEASE = 0640
  CEDA  DEFine PIpeline( PIPE###  )
+                  :
   SHelf         ==> /var/cicsts/
   (Mixed Case)  ==>
                 ==>
                 ==>
                 ==>
   Wsdir         ==> /u/prak###/tutorial06/wspickup/provider/
   (Mixed Case)  ==>
                 ==>
                 ==>
                 ==>
```

(Screenshot after the scroll-down)

Tech-Tip: The initial screen displays on half of the WSDir field. To unlock the field, you will need to scroll forward with PF8.

Since the Shelf and WSDir fields contain directory names, the entries must include a beginning and ending slash.

The fields for Configfile, Shelf, and Wsdir are 255 characters long. These 255 characters are split over multiple 3270 fields. If your entry is longer that the field on the 3270 device, just keep typing and allow the line to ‘split’ into the next 3270 field (i.e. no spaces in the middle of a name)..

```
  I New group PRAK065 created.
                                                  SYSID=CICS APPLID=CICS
  DEFINE SUCCESSFUL                          TIME:  18.58.04  DATE: 08.063
PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

42.After confirming your input by pressing the Enter key, you should get a message at the bottom of the screen, that a new group is created (Here: group PRAK065) and that the define was successful.

```
******************************** Top of Data ********************************
<?xml version="1.0" encoding="EBCDIC-CP-US"?>
<provider_pipeline xmlns="http://www.ibm.com/software/htp/cics/pipeline"
..... xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
..... xsi:schemaLocation="http://www.ibm.com/software/htp/cics/pipeline provider
  <service>
    <terminal_handler>
      <cics_soap_1.1_handler/>
    </terminal_handler>
  </service>
  <apphandler>DFHPITP</apphandler>
</provider_pipeline>
******************************** Bottom of Data ******************************
```

43.Let's have a short look at the pipeline configuration file basicsoap11provider.xml

Pipeline configuration files specify 0 or more message handlers that should be invoked as Web Service messages pass through the pipeline to the 'terminal handler'. The message handlers are invoked again as the Web Service response moves through the pipeline from the 'terminal handler' back to the service requester. "Terminal" in this case mean at the end of the pipeline (i.e. this program 'terminates' the pipeline).
For the pipeline configuration file above, there are no message handlers specified, and a CICS supplied terminal handler called cics_soap_1.1_handler will be invoked. The supplied CICS SOAP handler knows how to remove the SOAP envelope from the message and will then invoke the DFHPITP program specified in the apphandler tag. The CICS supplied SOAP handler will also add a SOAP envelope to the response from DFHPITP.

DFHPITP is a CICS supplied program that

- **locates the WSBIND file associated with the Web Service using this pipeline**
- **performs the XML to COMMAREA (or Container) conversion specified in the WSBIND file**
- **invokes the user application program specified in the WSBIND file**
- **performs COMMAREA (or Container) to XML conversionl cics**

```
CEDA INSTALL GROUP(PRAK###) PIPELINE(PIPE###)
```

44.To install the pipeline we just created, type
CEDA INSTALL GROUP(PRAK###) PIPELINE(PIPE###)
and press the Enter key (Replace ### with your UserID).

45.You should receive a message that says INSTALL SUCCESSFUL.

Note: Depending on previous labs you might get warnings telling that parts of your group have already been installed.

```
CEMT INQUIRE PIPELINE
```

46.Use the CEMT INQUIRE PIPELINE command

Note: The CEMT transaction is used to invoke all the master terminal functions. The master terminal program provides dynamic user control of the CICS system. By using this function, an operator can inquire about and change the values of parameters used by CICS, alter the status of the system resources, terminate tasks, and shut down the CICS system.

```
INQUIRE PIPELINE
STATUS:  RESULTS - OVERTYPE TO MODIFY
 Pip(DFHMASFP) Ena Con(/usr/lpp/cicsts/cicsts31/s)
    She(/var/cicsts/              ) Wsd(/u/cicsts/ws/provider/    )
 Pip(DFHMASFR) Ena Con(/usr/lpp/cicsts/cicsts31/s)
    She(/var/cicsts/              ) Wsd(/u/cicsts/ws/requester/   )
 Pip(EXPIPE01) Ena Con(/usr/lpp/cicsts/cicsts31/s)
    She(/var/cicsts/              ) Wsd(/usr/lpp/cicsts/cicsts31/s)
 Pip(EXPIPE02) Ena Con(/usr/lpp/cicsts/cicsts31/s)
    She(/var/cicsts/              ) Wsd(/usr/lpp/cicsts/cicsts31/s)
 Pip(PIPE034 ) Ena Con(/usr/lpp/cicsts/cicsts31/s)
    She(/var/cicsts/              ) Wsd(/u/prak034/tutorial06/wspi)
```

to verify the PIPELINE was installed properly (You may need to scroll down using F8).
You should see a status of Ena (Enabled) for your pipeline.

```
S Pip(PIPE034 ) Ena Con(/usr/lpp/cicsts/cicsts31/s)
     She(/var/cicsts/              ) Wsd(/u/prak034/tutorial06/wspi)
```

47.If you tab down and place an S to the left of your pipeline entry and press the enter button, you will see your pipeline's details.

```
INQUIRE PIPELINE
RESULT - OVERTYPE TO MODIFY
  Pipeline(PIPE034)
  Enablestatus( Enabled )
  Configfile(/usr/lpp/cicsts/cicsts31/samples/pipelines/basicsoap11provide)
  Configfile(r.xml)
  Shelf(/var/cicsts/)
  Wsdir(/u/prak034/tutorial06/wspickup/provider/)
```

48.The Shelf and Wsdir entries are directories and must start and end with a „/".

Note: If you add a new WSBIND file to your PIPELINE pickup directory this will not automatically be noticed by CICS. The pickup directory is searched only at PIPELINE installation (as we just did) or if you perform an explicit scan. To scan your pipeline you would have to do the following:
(just for your information, you don't have to do this now)
CEMT PERFORM PIPELINE(PIPE###) SCAN
and press the enter key.
You should see a message saying RESPONSE:NORMAL.

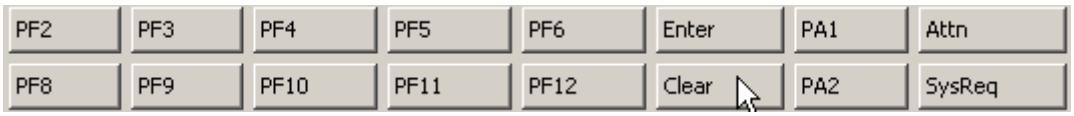
If you have difficulties, the following URL contains an overview :
http://publib.boulder.ibm.com/infocenter/cicsts/v3r1/index.jsp?topic=/com.ibm.cics.ts31.doc/dfha7/dfha71z.htm

5.2 Create a TCPIPService Resource Definition

This tells CICS to listen on a specified TCPIP port and accept HTML.
The port number CICS is listening on is defined in a TCPIPService definition. One TCPIPService can be used for several pipelines and Web Services. Therefore we can reuse the TCPIPService used in a previous tutorial (if it does exist) to tell CICS to receive the inbound HTTP requests on a specific port.

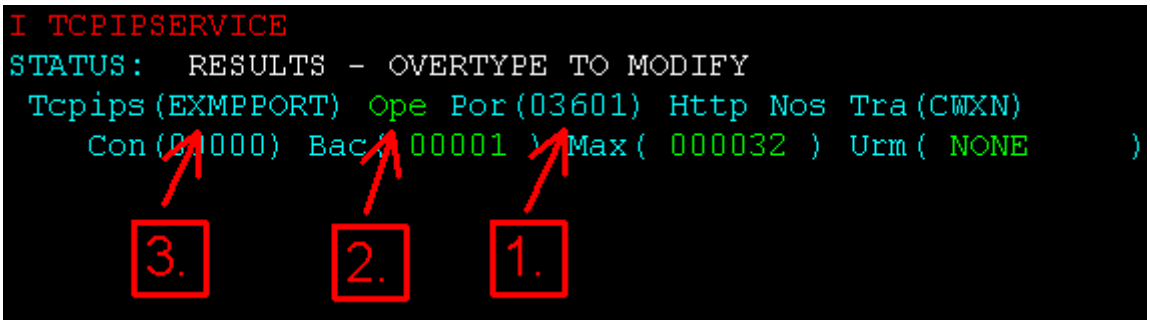
49.We need to ensure that the needed TCPIPService exists and that its status is OPEN.
To check this, press F3 to end the CEDA DEFINE procedure we did before.
You should get the message STATUS: SESSION ENDED as displayed in the screenshot below.

Wir verlassen die Definition des Programms mit der Eingabe von F3, als Ergebnis davon erscheint "SESSION ENDED".
Wir geben in die oberste Zeile den nächsten CEDA-Befehl ein



50.Now clear the screen clicking the Clear button below your 3270 screen.

51.Review the currently active service by typing CEMT I TCPIPService and …



hit Enter.

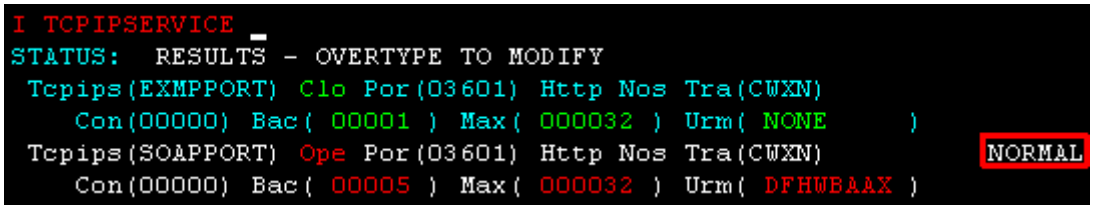
Note:

1. **Port of the TCPIPService (Here: 03601)**
2. **Status of the port (Here: Open)**
3. **Name of the TCPIPService (Here: EXMPPORT)**

We need a TCPIPService on port 3601 with status open and it should be named SOAPPORT. If there is no such TCPIPService, we need to define it (like in the screenshot above, the active TCPIPService on port 3601 is EXMPPORT).

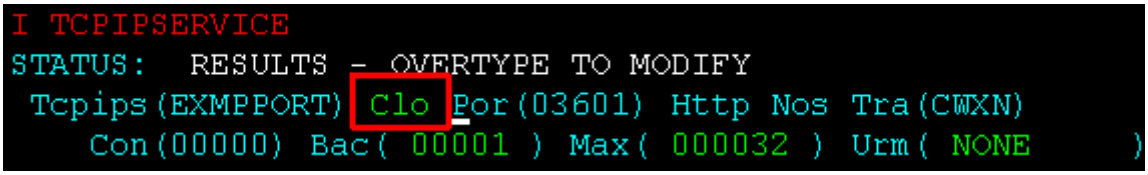
Now there are several possibilities:

Possibility # 1 : If your active (ope) TCPIPServiceon Port 3601 already is SOAPPORT, continue this tutorial with item 64.



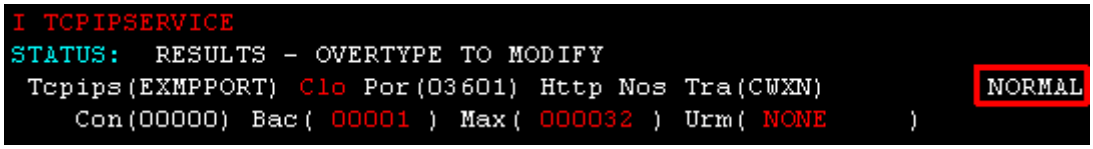
52.Possibility # 2 : If your CEMT I TCPIPSERVICE request provided a closed 'Clo' SOAPPORT TCPIPService on port 3601, open it by overtyping Clo with Ope and pressing return. You should get the message NORMAL.

53.You can ensure that the TCPIPService SOAPPORT is opened by pressing F3, followed by the enter key (The command CEMT I TCPIPSERVICE is still typed in).



54.Possibility # 3 : If your active TCPIPService on port 3601 is not SOAPPORT, we need to close the active service of port 3601 first and add our needed SOAPPORT TCPIPService.

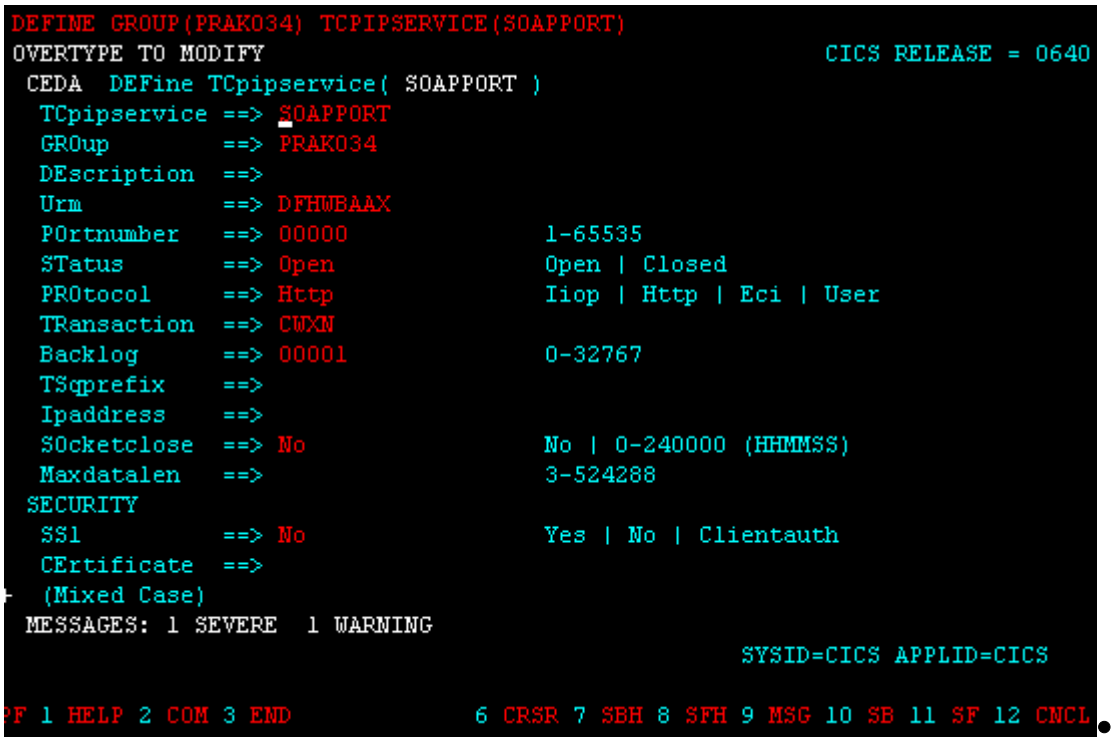
Move your cursor to the status field (to the keyword Ope) by hitting your TAB key and overtype Ope with Clo (which stands for closed)…

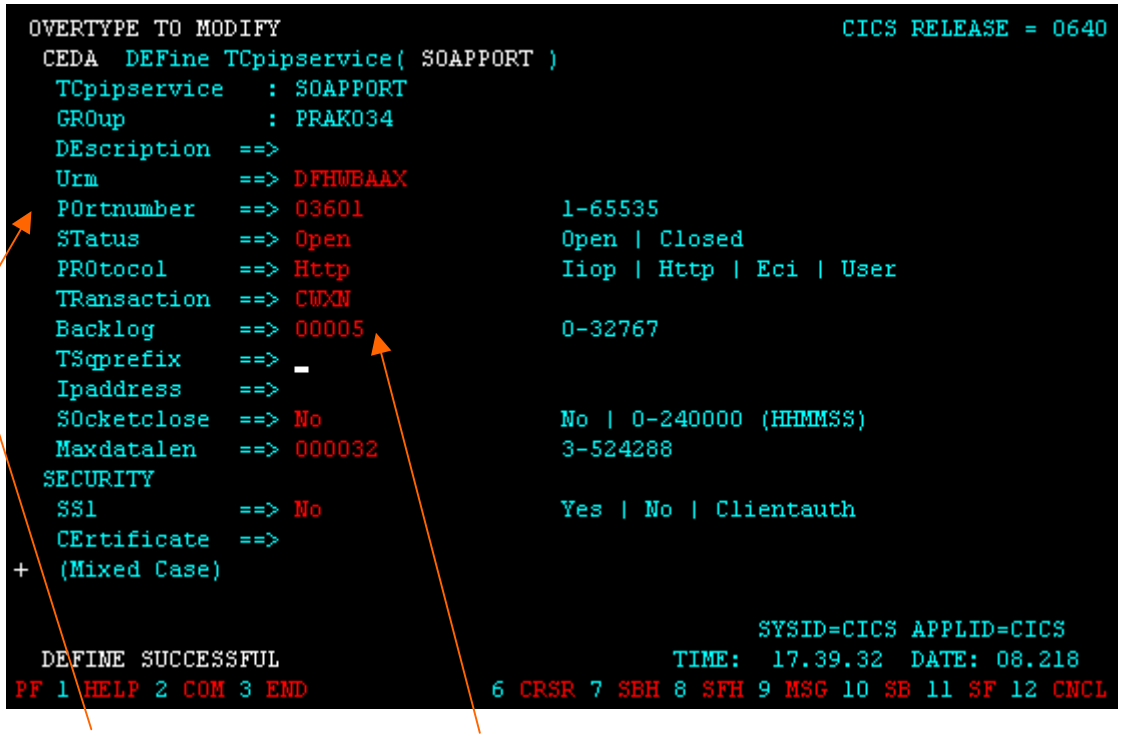


55.…and hit enter. The lines containing the EXMPPORT TCPIPService should turn white and your returned status information should be NORMAL.

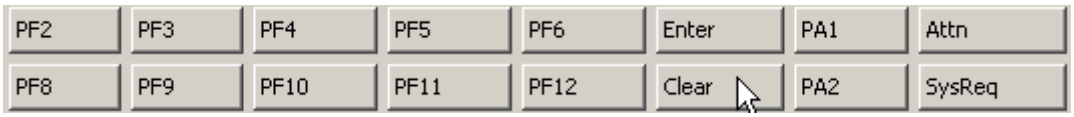
56.You can ensure that the TCPIPService EXMPPORT is closed by pressing F3, followed by the enter key (The command CEMT I TCPIPSERVICE is still typed in).

57.To add our needed TCPIPService, type in
CEDA DEFINE GROUP(PRAK###) TCPIPSERVICE(SOAPPORT)


DEFINE GROUP(PRAK034) TCPIPSERVICE(SOAPPORT)
OVERTYPE TO MODIFY
CICS RELEASE = 0640
CEDA DEFine TCpipservice(SOAPPORT)
TCpipservice ==> SOAPPORT
GROup ==> PRAK034
DEscription ==>
Urm ==> DFHWBAAX
POrtnumber ==> 00000 1-65535
STatus ==> Open Open | Closed
PROtocol ==> Http Iiop | Http | Eci | User
TRansaction ==> CWXN
Backlog ==> 00001 0-32767
TSqprefix ==>
Ipaddress ==>
SOcketclose ==> No No | 0-240000 (HHMMSS)
Maxdatalen ==> 3-524288
SECURITY
SSl ==> No Yes | No | Clientauth
CErtificate ==>
(Mixed Case)
MESSAGES: 1 SEVERE 1 WARNING
SYSID=CICS APPLID=CICS
PF 1 HELP 2 COM 3 END 6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL



58.Change Portnumber to 03601 and Backlog to 00005.
Press enter. You should get the message DEFINE SUCCESSFUL.



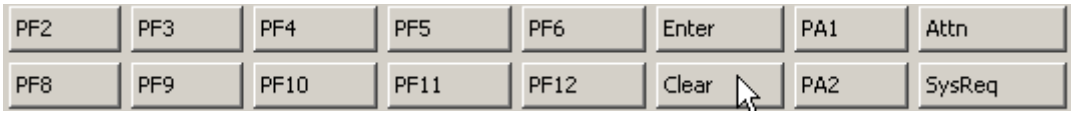
59.Press F3 to end the CEDA DEFINE procedure and Clear your screen.

```
CEDA INSTALL GROUP(PRAK###) TCPIPSERVICE(SOAPPORT)
```

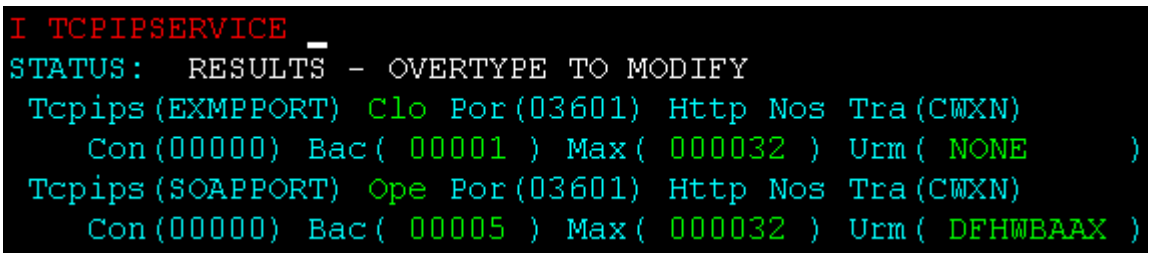
60.In order to install our newly defined TCPIPService, insert
CEDA INSTALL GROUP(PRAK###) TCPIPSERVICE(SOAPPORT)
and press Enter (Replace ### with your UserID).

CEMT I

61.You should get the message INSTALL SUCCESSFUL.



62.Press F3 to end the CEDA INSTALL procedure and Clear your screen.



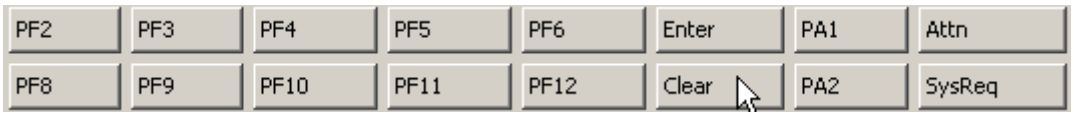
63.Let's check the TCPIPSERVICEs once again. Type
CEMT I TCPIPSERVICE
and press enter.

Our installed TCPIPService SOAPPORT on port 3601 should be displayed with Openstatus Ope.

Continue here if you skipped steps 52 or 54 till 63 because you already had SOAPPORT as TCPIPService. If you did the last steps and added SOAPPORT, also continue.

```
I TCPIPSERVICE _
RESULT - OVERTYPE TO MODIFY
  Tcpipservice(SOAPPORT)
  Openstatus( Open )
  Port(03601)
  Protocol(Http)
  Ssltype(Nossl)
  Transid(CWXN)
  Authenticate(Noauthentic)
  Connections(00000)
  Backlog( 00005 )
  Maxdatalen( 000032 )
  Urm( DFHWBAAX )
  Privacy(Notsupported)
  Ciphers()
  Ipaddress(134.2.205.54)
  Socketclose(Wait)
  Closetimeout(000000)
  Dnsgroup()
+ Dnsstatus(              )
```

64.Move your cursor to anywhere to the TCPIPService SOAPPORT entry and press enter. Your screen should look like below.



65.Press F3 to end the CEMT I TCPIPSERVICE procedure and Clear your screen.

5.3 Verify resource installation

In this part of the tutorial, you will verify that the CICS resources were installed properly.

```
CEMT I WEBSERVICE_
```

66.From the CICS session, type CEMT INQUIRE WEBSERVICE and press the enter key.

```
Webs(inquireSingle065                ) Pip(PIPE065 )
   Ins Uri($337030 ) Pro(DFHOXCMN) Com                    Dat(20080310)
```

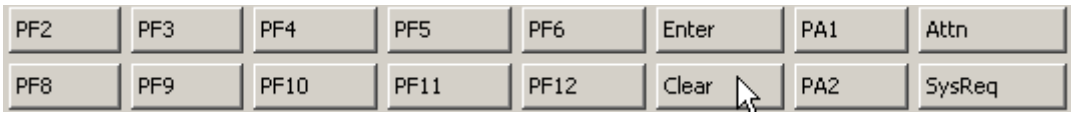
67.You may need to scroll down a few times, using the F8 key, until you find your corresponding inquireSingle### entry (Replace ### with your UserID). In this example the UserID is 065.

Note: The resource is created with a name matching the WSBIND file name and the WEBSERVICE is associated with the PIPELINE that scanned it in.

```
I WEBSERVICE
RESULT - OVERTYPE TO MODIFY
  Webservice(inquireSingle065)
  Pipeline(PIPE065)
  Validationst( Novalidation )
  State(Inservice)
  Urimap($337030)
  Program(DFHOXCMN)
  Pgminterface(Commarea)
  Container()
  Datestamp(20080310)
  Timestamp(13:37:03)
  Wsdlfile()
  Wsbind(/u/prak065/tutorial06/wspickup/provider/inquireSingle065.wsbind)
  Endpoint()
  Binding(DFHOXCMNHTTPSoapBinding)
```

68.Tab down using the TAB key and type S next to the WEBSERVICE entry, then press the enter key.
Your screen should look similar to the following:

Note: A WEBSERVICE resource definition provides more details about the Web Service. If we want CICS to do conversion between the XML that contains our application data and the COMMAREA or Container that delivers the application data to our CICS programs, the WEBSERVICE definition will point to a WSBIND file. The WSBIND file contains the directions to CICS on how to perform the conversion. If we want CICS to validate the request, which means that CICS will verify that the incoming request is valid based on the rules in the WSDL, then the WEBSERVICE definition must point to the WSDL file that CICS should use to validate the request.



69.Press F3 to end the CEMT INQUIRE procedure and Clear your screen.

```
CEMT I URIMAP
```

70.Type
CEMT INQUIRE URIMAP
and press the enter key. This will list the installed URIMAPs.

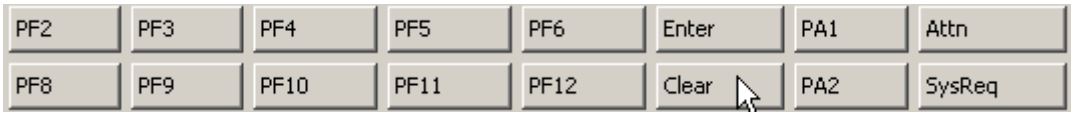
```
Uri($337030 ) Pip Ena     Http
   Host(*                        ) Path(/exampleApp/inquireSingle065    )
```

71.Again, you may need to scroll down a few times using F8 until you find your corresponding entry (Use the Path of the URIMAPs to find your entry, it should be: /exampleApp/inquireSingle###). In this example the UserID is 065)

```
I URIMAP
RESULT - OVERTYPE TO MODIFY
  Urimap($337030)
  Usage(Pipe)
  Enablestatus( Enabled )
  Analyzerstat(Noanalyzer)
  Scheme(Http)
  Redirecttype( None )
  Tcpipservice()
  Host(*)
  Path(/exampleApp/inquireSingle065)
  Transaction(CPIH)
  Converter()
  Program()
  Pipeline(PIPE065)
  Webservice(inquireSingle065)
  Userid()
  Certificate()
  Ciphers()
+ Templatename()
```

72.Tab down using the TAB key and type S next to the URIMAP entry and press the enter button. Your screen should look similar to the following:

CICS uses this URIMAP resource to match the incoming URI with a specified action. A URIMAP points to a PIPLINE definition and a WEBSERVICE definition.



73.Press F3 to end the CEMT INQUIRE procedure and Clear your screen.

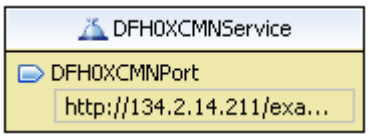
74.Sign off from CICS by typing the command CESF and close the Emulator window.

Um uns aus CICS auszuloggen, betätigen wir einmal oder mehrmals die Taste F3, bis die Meldung "STATUS: SESSION ENDED" ausgegeben wird. In die Zeile darüber geben wir die Logoff-Transaktion "CESF LOGOFF", gefolgt von der Eingabetaste, ein.

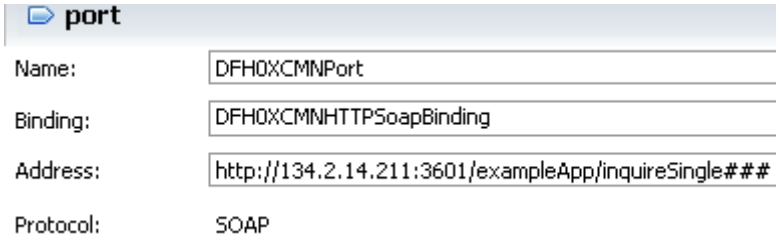
6. Test the Web Service

In this part of the tutorial we will test your newly installed Web Service. These testing steps will be similar to the testing steps you performed in an earlier tutorial.
When testing the web service using the explorer we have to define the endpoint (i.e. the location) of the web service consisting of hostname, port and uri

75.In WDz double-click on inquireSingle###.wsdl (Replace ### with your UserID).
This will open the file in a WSDL editor. Ensure you are looking at the graphical view (click the Graph tab on the bottom of the editor if needed)



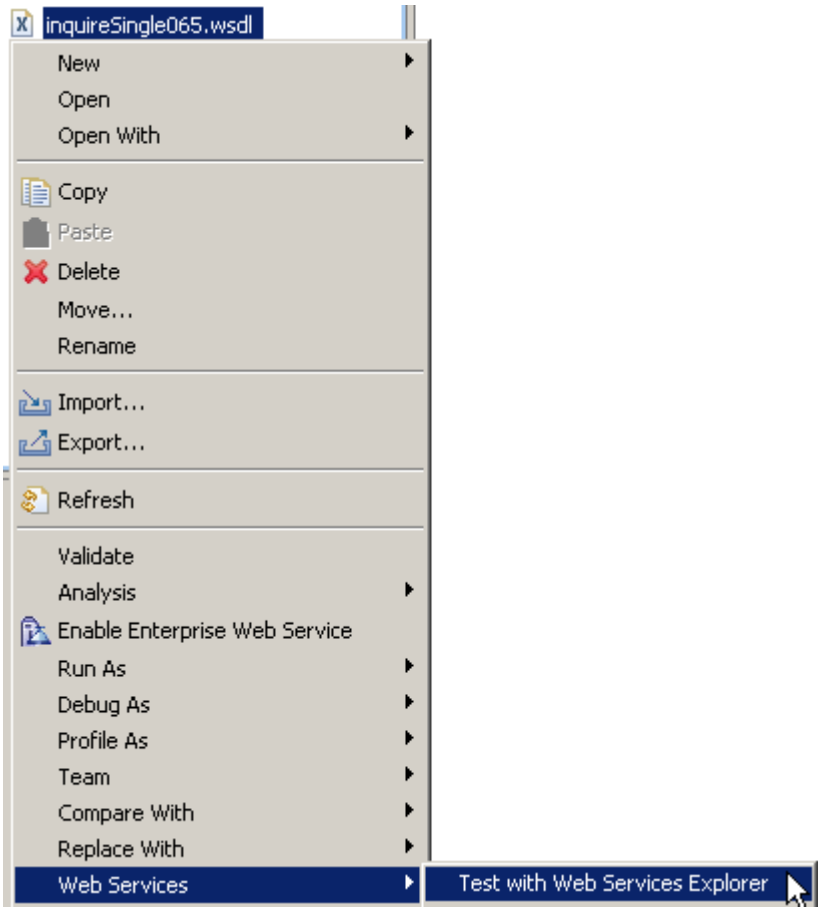
76.Click on the DFH0XCMNPort in the service element called DFH0XCMNService.



77.In the properties view below, change the address to http://134.2.14.211:3601/exampleApp/inquireSingle### (Replace ### with your UserID).

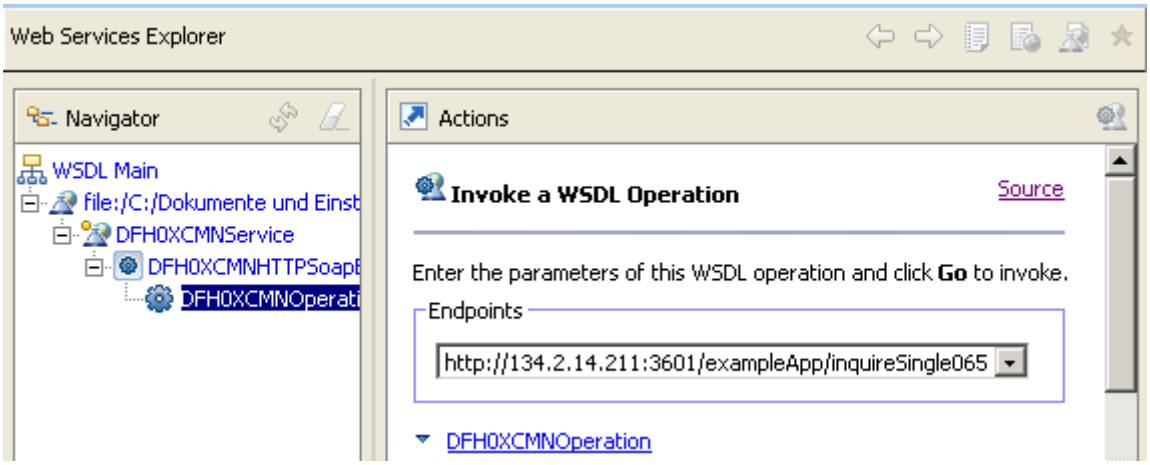
Notice that the part after host and port number (referred to as the path information) contains /exampleApp/inquireSingle###. This is the same value as seen in the PATHNAME parameter on the URIMAP resource that CICS install in response to our pipeline SCAN request.

78.Save the changes with CTRL + S and close the WSDL editor.



79.From the Navigator view, right-click on inquireSingle###.wsdl, and from the context menu select Web Services → "Test with Web Services Explorer".

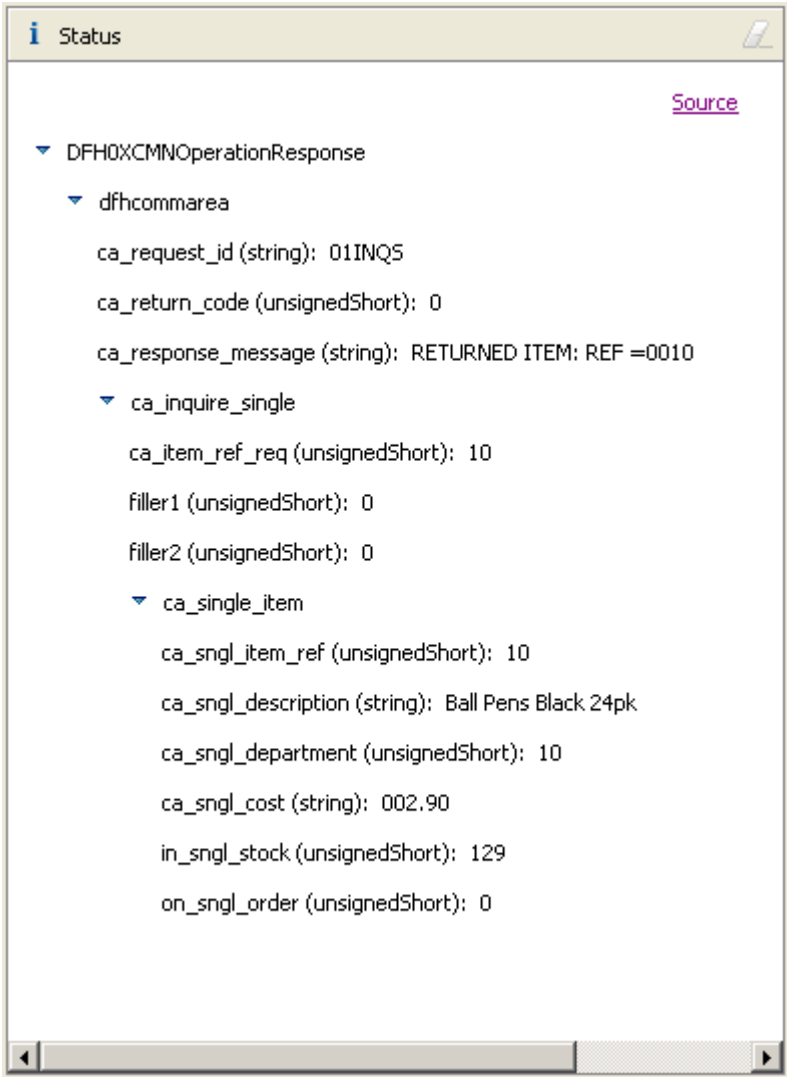
80.Verify again that the endpoint is http://134.2.14.211:3601/exampleApp/inquireSingle###
(Replace ### with your UserID)



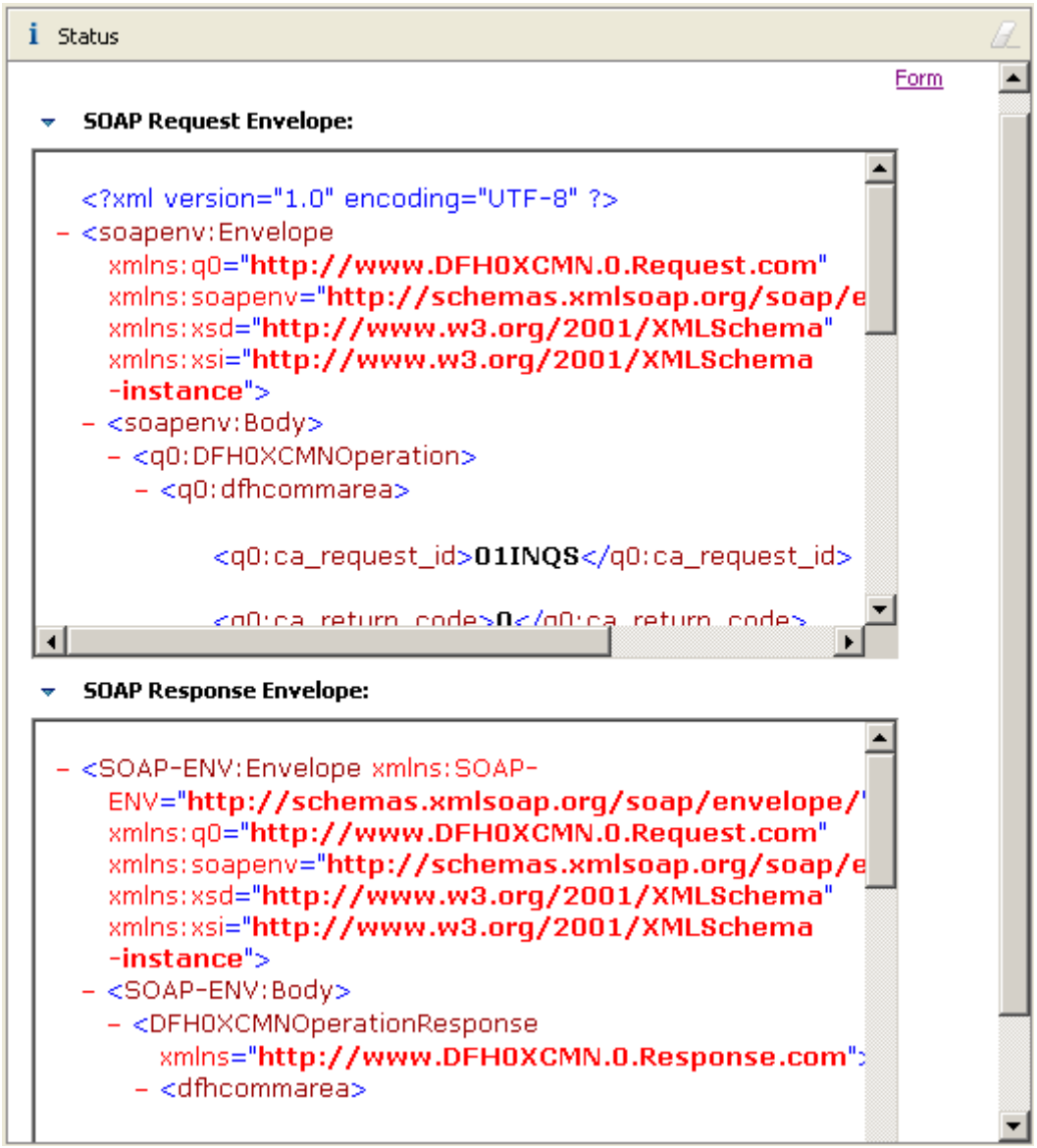
81.In the Navigator pane of the Web Services Explorer, expand DFH0XCMNHTTPSoapBinding and select DFH0XCMNOperation (this is the service operation).
You should see a window similar to the following:

Field	Value
ca_request_id	01INQS
ca_return_code	0
ca_response_message	(space)
ca_item_ref_req	0010
filler1	0
filler2	0
ca_sngl_item_ref	0
ca_sngl_description	(space)
ca_sngl_department	0
ca_sngl_cost	(space)
in_sngl_stock	0
on_sngl_order	0

82.In the Actions pane, verify that the Endpoints field shows the new endpoint and type the above values and click the Go button.



83.The results from the Web Service request will be shown in the Status pane.
You can double-click on the title Status (top of the Status pane) to expand this window. You should see a window similar to the following.



84.Click on Source towards the top right in the Status pane to get a detailed Source Code view of the SOAP request and response. Use the scroll bar to see the complete message.

85.Double-click on the title Status to return to the previous Web Services Explorer view.

86.Close the Web Services Explorer and any open editors.

Congratulation, you successfully completed Tutorial06!

Appendix A
URIMAP resource definitions

A Uniform Resource Identifier (URI) is a compact string of characters used to identify or name a resource on the Internet.

URIMAP definitions are resource definitions that match the URIs of HTTP or Web service requests, and provide information on how to process the requests.

URIMAP definitions are used to provide three different Web-related facilities in CICS®:

1. **Requests from a Web client, to CICS as an HTTP server. URIMAP definitions for requests for CICS as an HTTP server have a USAGE attribute of SERVER. These URIMAP definitions match the URLs of HTTP requests that CICS expects to receive from a Web client, and they define how CICS should provide a response to each request. You can use a URIMAP definition to tell CICS to:**

   - **Provide a static response to the HTTP request, using a document template or z/OS® UNIX® System Services HFS file.**
   - **Provide a dynamic response to the HTTP request, using an application program.**

   Planning your CICS Web support architecture for CICS as an HTTP server has planning information for CICS as an HTTP server, and Resource definition for CICS Web support has instructions for creating a URIMAP definition for these purposes.

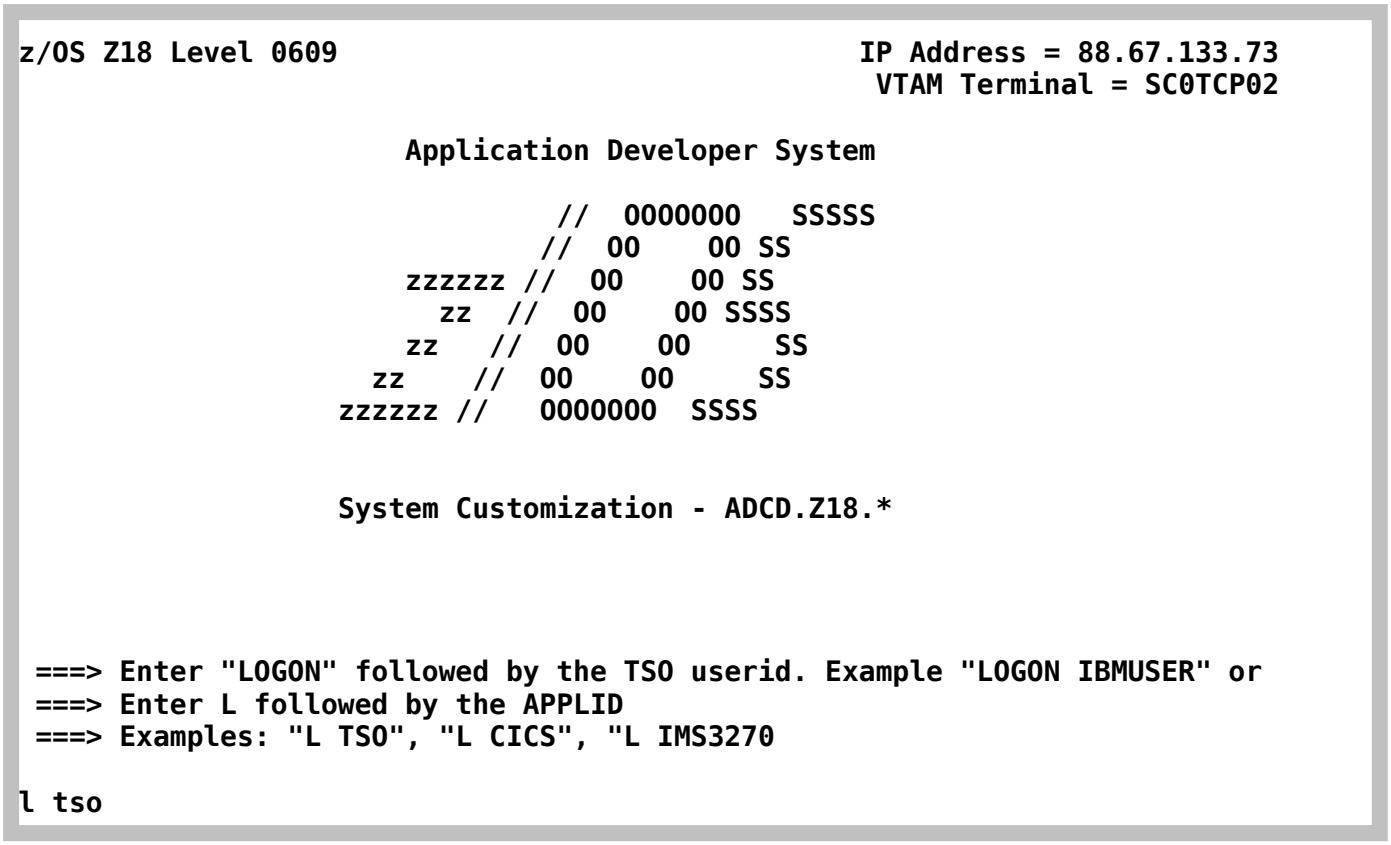
2. **Requests to a server, from CICS as an HTTP client. URIMAP definitions for requests from CICS as an HTTP client have a USAGE attribute of CLIENT. These URIMAP definitions specify URLs that are used when a user application, acting as a Web client, makes a request through CICS Web support to an HTTP server. Setting up a URIMAP definition for this purpose means that you can avoid identifying the URL in your application program. HTTP client requests from a CICS application has instructions for creating a URIMAP definition for this purpose.**

3. **Web service requests. URIMAP definitions for Web service requests have a USAGE attribute of PIPELINE. These URIMAP definitions associate a URI for an inbound Web service request (that is, a request by which a client invokes a Web service in CICS) with a PIPELINE or WEBSERVICE resource that specifies the processing to be performed. Web services - start here has further information about Web services in CICS.**

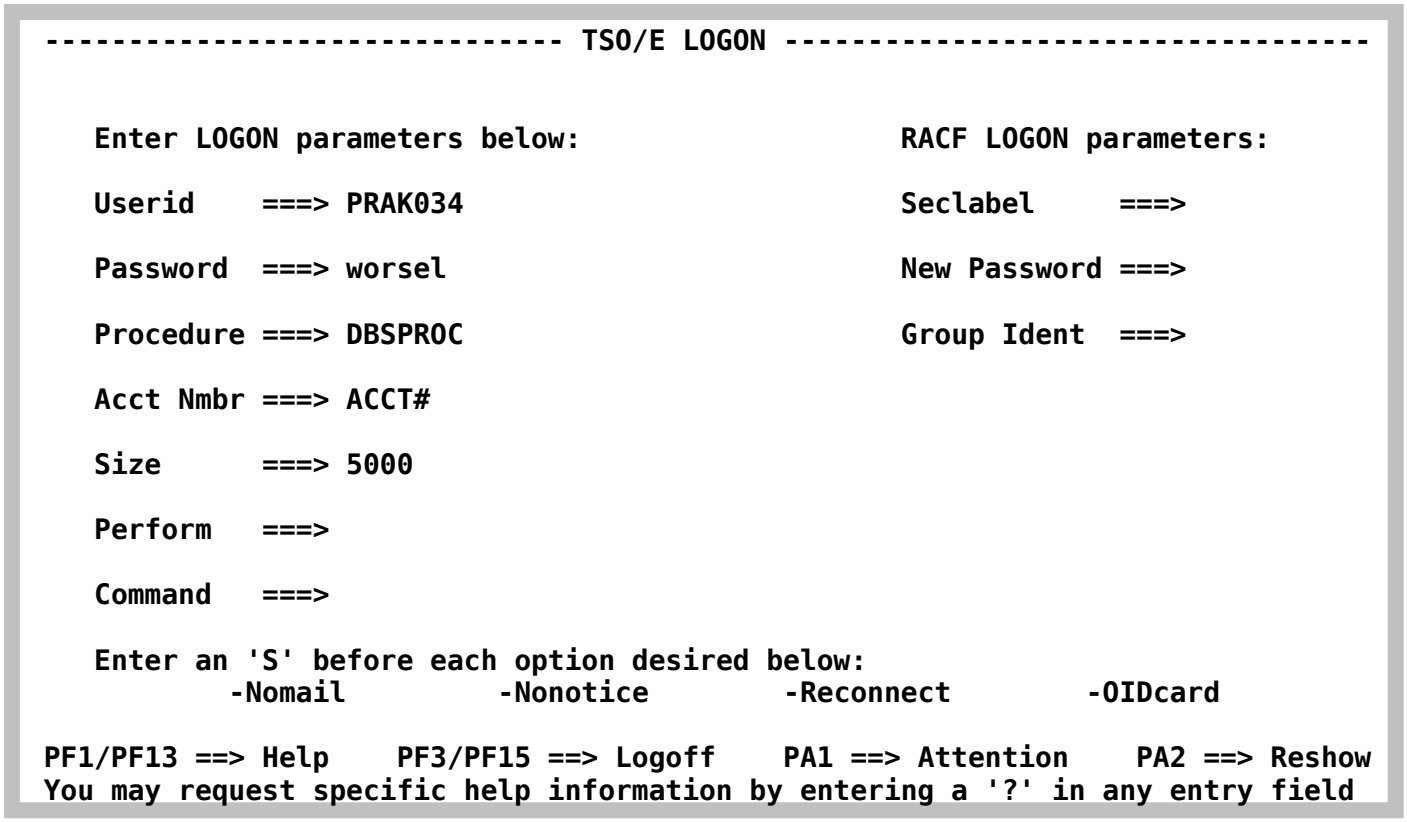
For CICS as an HTTP server, URIMAP definitions incorporate most of the functions that were previously provided by the analyzer program associated with the TCPIPSERVICE definition. An analyzer program may still be involved in the processing path if required.

http://publib.boulder.ibm.com/infocenter/cicsts/v3r1/index.jsp?topic=/com.ibm.cics.ts31.doc/dfha4/r

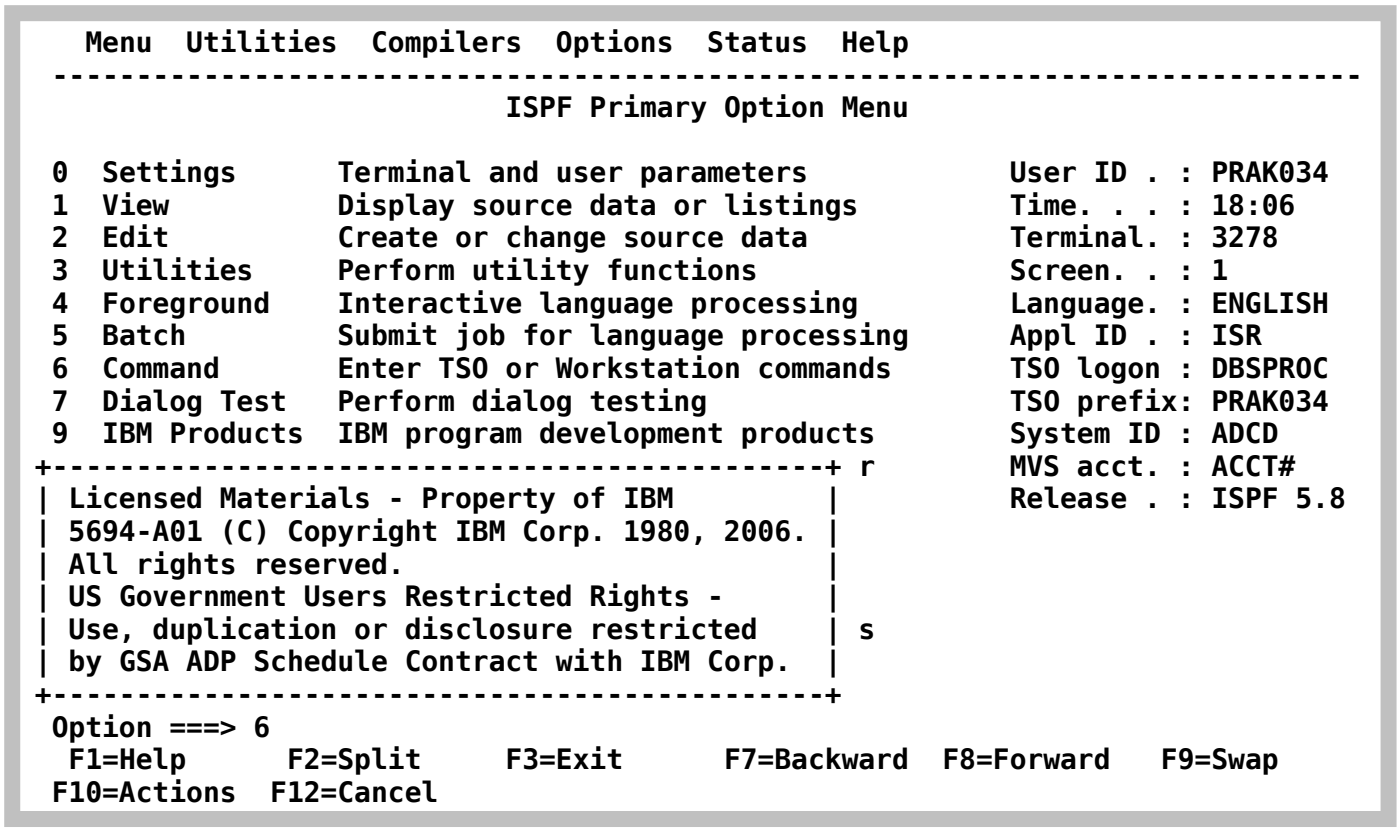
Appendix B
Using OMVS



Für die Nutzung von Unix System Services (alte Bezeichnung OMVS), close the connection in the Remote Systems view and start your 3270 emulator.



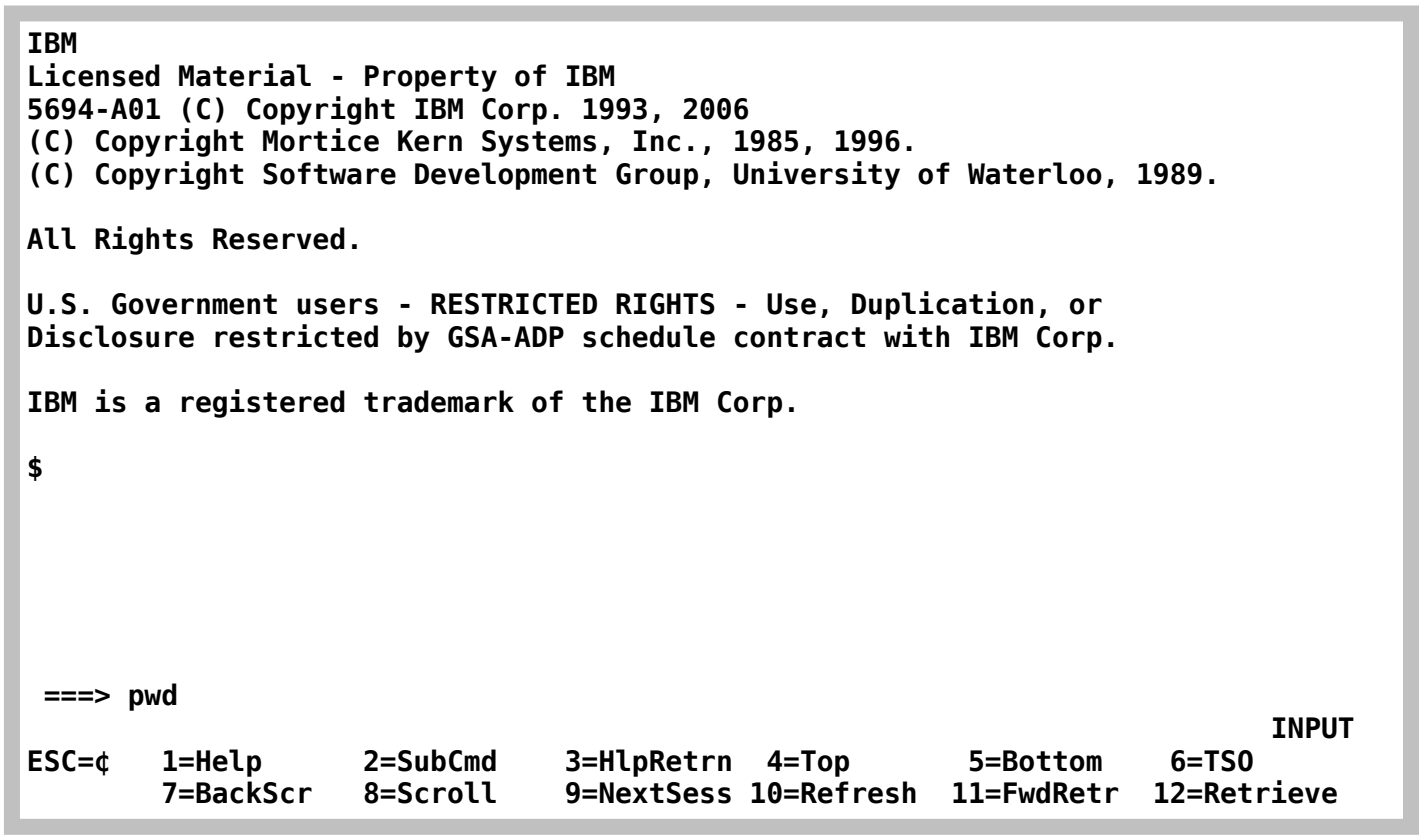
logon


Menu Utilities Compilers Options Status Help
ISPF Primary Option Menu
0 Settings Terminal and user parameters
1 View Display source data or listings
2 Edit Create or change source data
3 Utilities Perform utility functions
4 Foreground Interactive language processing
5 Batch Submit job for language processing
6 Command Enter TSO or Workstation commands
7 Dialog Test Perform dialog testing
9 IBM Products IBM program development products
User ID . : PRAK034
Time. . . : 18:06
Terminal. : 3278
Screen. . : 1
Language. : ENGLISH
Appl ID . : ISR
TSO logon : DBSPROC
TSO prefix: PRAK034
System ID : ADCD
MVS acct. : ACCT#
Release . : ISPF 5.8
Licensed Materials - Property of IBM
5694-A01 (C) Copyright IBM Corp. 1980, 2006.
All rights reserved.
US Government Users Restricted Rights -
Use, duplication or disclosure restricted
by GSA ADP Schedule Contract with IBM Corp.
Option ===> 6
F1=Help F2=Split F3=Exit F7=Backward F8=Forward F9=Swap
F10=Actions F12=Cancel

go to ISPF and enter “6” to go to the ISPF Shell. Now you can enter TSO commands.

```
  Menu  List  Mode  Functions  Utilities  Help
------------------------------------------------------------------------------
                              ISPF Command Shell
Enter TSO or Workstation commands below:

===> OMVS



Place cursor on choice and press enter to Retrieve command

=> omvs
=>
=>
=>
=>
=>
=>
=>
=>
=>

 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
F10=Actions  F12=Cancel
```

When you have used Unix System Services before, you see OMVS in the Commad List Area. Double click on it to move it to the command line.



Now you have a familiar Unix command line Screen. On the Command line you may enter regular Unix commands,

e.g. pwd, to show the present working directory

```
IBM
Licensed Material - Property of IBM
5694-A01 (C) Copyright IBM Corp. 1993, 2006
(C) Copyright Mortice Kern Systems, Inc., 1985, 1996.
(C) Copyright Software Development Group, University of Waterloo, 1989.

All Rights Reserved.

U.S. Government users - RESTRICTED RIGHTS - Use, Duplication, or
Disclosure restricted by GSA-ADP schedule contract with IBM Corp.

IBM is a registered trademark of the IBM Corp.

$ pwd
/u/prak034
$ ls -ls
total 16
  16 drwxrwxrwx   3 PRAK034  PRAKT       8192 Aug  4 15:57 tutorial06
   0 -rw-rw-rw-   1 PRAK034  PRAKT          0 Aug  8 10:21 x
$
 ===>
                                                                      INPUT
ESC=¢   1=Help      2=SubCmd    3=HlpRetrn  4=Top       5=Bottom    6=TSO
        7=BackScr   8=Scroll    9=NextSess 10=Refresh  11=FwdRetr  12=Retrieve
```

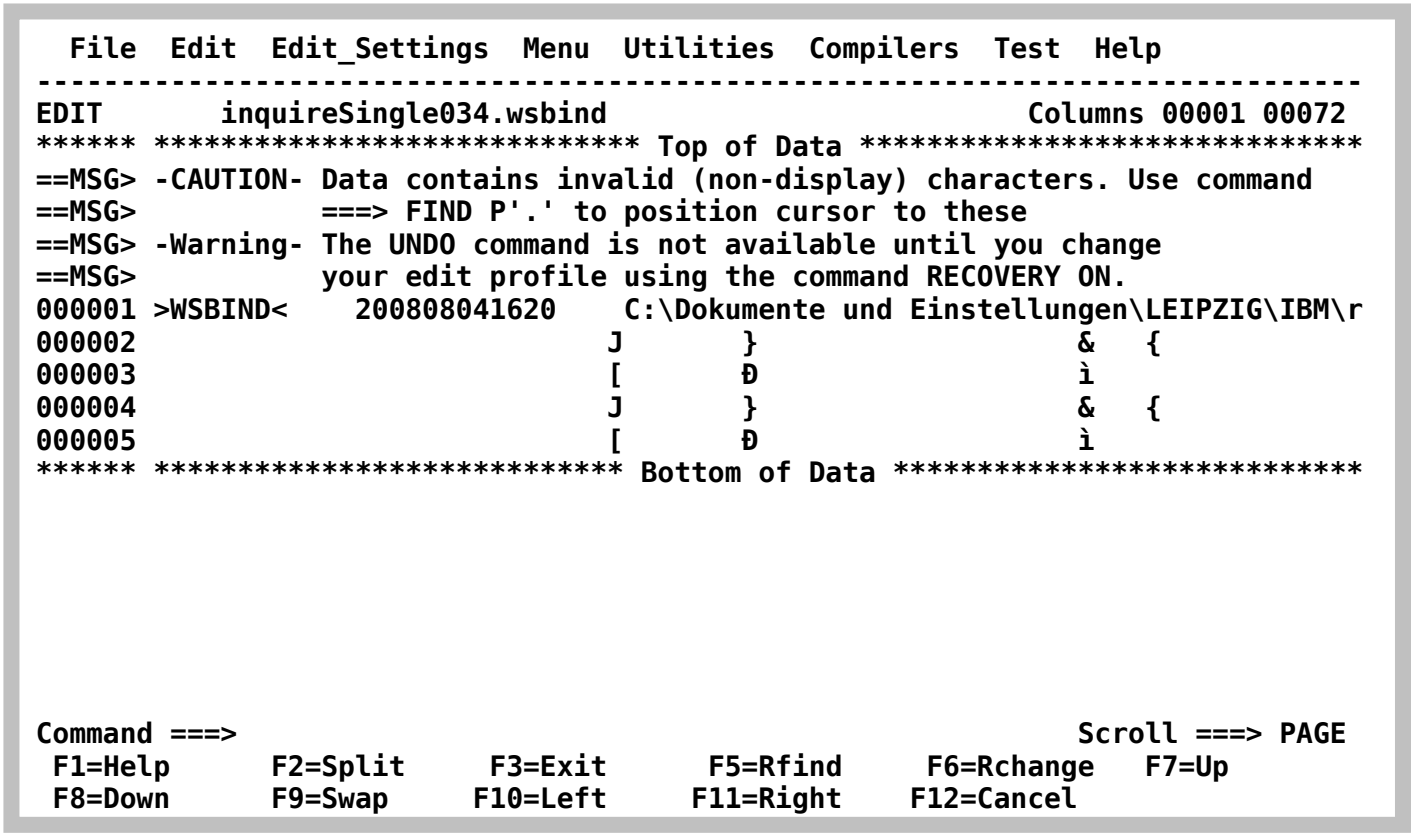
or ls –ls to show the content of the directory

```
$ cd tutorial06
$ pwd
/u/prak034/tutorial06
$ ls -ls
total 16
  16 drwxrwxrwx   3 PRAK034  PRAKT       8192 Aug  4 16:53 wspickup
$









 ===> exit
                                                                        INPUT
ESC=¢   1=Help      2=SubCmd    3=HlpRetrn  4=Top       5=Bottom    6=TSO
        7=BackScr   8=Scroll    9=NextSess 10=Refresh  11=FwdRetr  12=Retrieve
```

hit the F8 key

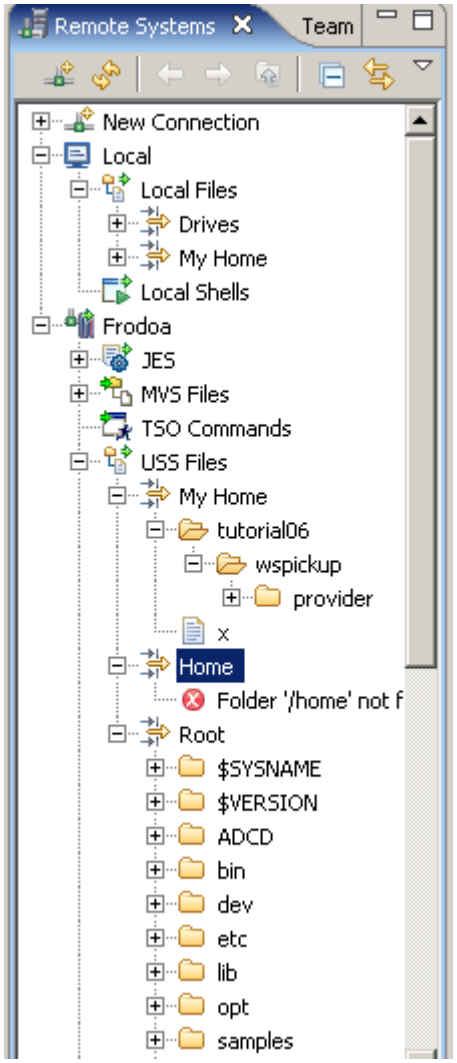
```
$ cd tutorial06
$ pwd
/u/prak034/tutorial06
$ ls -ls
total 16
  16 drwxrwxrwx   3 PRAK034  PRAKT       8192 Aug  4 16:53 wspickup
$ cd wspickup
$ ls -ls
total 16
  16 drwxrwxrwx   2 PRAK034  PRAKT       8192 Aug  4 17:06 provider
$ cd provider
$ ls -ls
total 0
   0 -rw-rw-rw-   1 PRAK034  PRAKT          0 Aug  4 17:06 inquireSingle034.wsbi
nd
$ pwd
/u/prak034/tutorial06/wspickup/provider
$
 ===>
                                                                        INPUT
ESC=¢   1=Help      2=SubCmd    3=HlpRetrn  4=Top       5=Bottom    6=TSO
        7=BackScr   8=Scroll    9=NextSess 10=Refresh  11=FwdRetr  12=Retrieve
```

repeated cd shows subdirectories. Enter the OEDIT command to call the USS ISPF editor



and you see the content of the file inquireSingle034.wsbind.

Use exit to leave the editor and again exit and enter to leave USS. repeated F3 to logoff. In the Remote Systems view, connect Frodo again.



USS files corresponds to the content of the Remote Systems View

Appendix C
Build-in CICS Transactions

CBAM - BTS browser
CDBC - database control menu
CDBI - database control inquiry
CDBM - database control interface
CEBR - temporary storage browse
CEBT - master terminal (alternate CICS)
CECI - command-level interpreter
CECS - command-level syntax-checking transaction
CEDA - resource definition online
CEDF execution diagnostic facility
CEDX also execution diagnostic facility
CEMT - master terminal
CEOT - terminal status
CESF - sign off
CESN - sign on
CEST - supervisory terminal
CETR - trace control
CIND - in-doubt testing tool
CLER - Language Environment run-time options
CMAC - messages and codes display
CMSG - message switching
CPIA - unit of work resynchronization transaction
COD0 - CICSPLex SM method-level debugging
CODB - CICSPLex SM system-level debugging
COLM - Restart CICSPLex SM management of a CICS system
COLU - CICSPLex SM online utility
COSH - Stop CICSPLex SM management of a CICS system
COVC - CICSPlex SM Web User Interface server controller
CRTE - remote transactions
CSFE - terminal and system test
CSPG - page retrieval
CWTO - write to operator
DSNC - CICS DB2 transaction

Of these, the Transactions CEDA, CEMT, and CESF are used in this tutorial.

Appendix D
CEDA—Transaction Resource definition online

Use CEDA to define resources to your CICS system while the system is running. With CEDA, you can update both the CICS System Definition (CSD) file and the running CICS system.

Start the transaction by entering CEDA on the command line of the screen. Press ENTER. You get a screen, which shows the options available within CEDA

ADd
ALter
APpend
CHeck
COpy
DEFine
DELete
DIsplay
Expand
Install
Lock
Move
REMove
RENname
UNlock
USerdefine
View

ADd
Adds a group name to a list, and creates a new list if the list did not already exist.

ALter
Changes some of the attributes of an existing resource definition.

APpend
Copies a list to another list.

CHeck
Checks that the resource definitions within a group or list are consistent with one another.

COpy
Copies one or more resource definitions.

DEFine
Creates a new resource definition on the CSD file.

DELete
Erases one or more resource definitions from the CSD file.

DIsplay
Displays one or more lists or groups on a full-screen panel.

Expand
Shows the resource definitions in one or more groups on a full-screen panel.

Install
Dynamically makes one or more resource definitions available to an active CICS system.

Lock
Assures exclusive access to a group or list by restricting update and delete access to a single operator identifier.

Move
Moves one or more resource definitions from one named group to another.

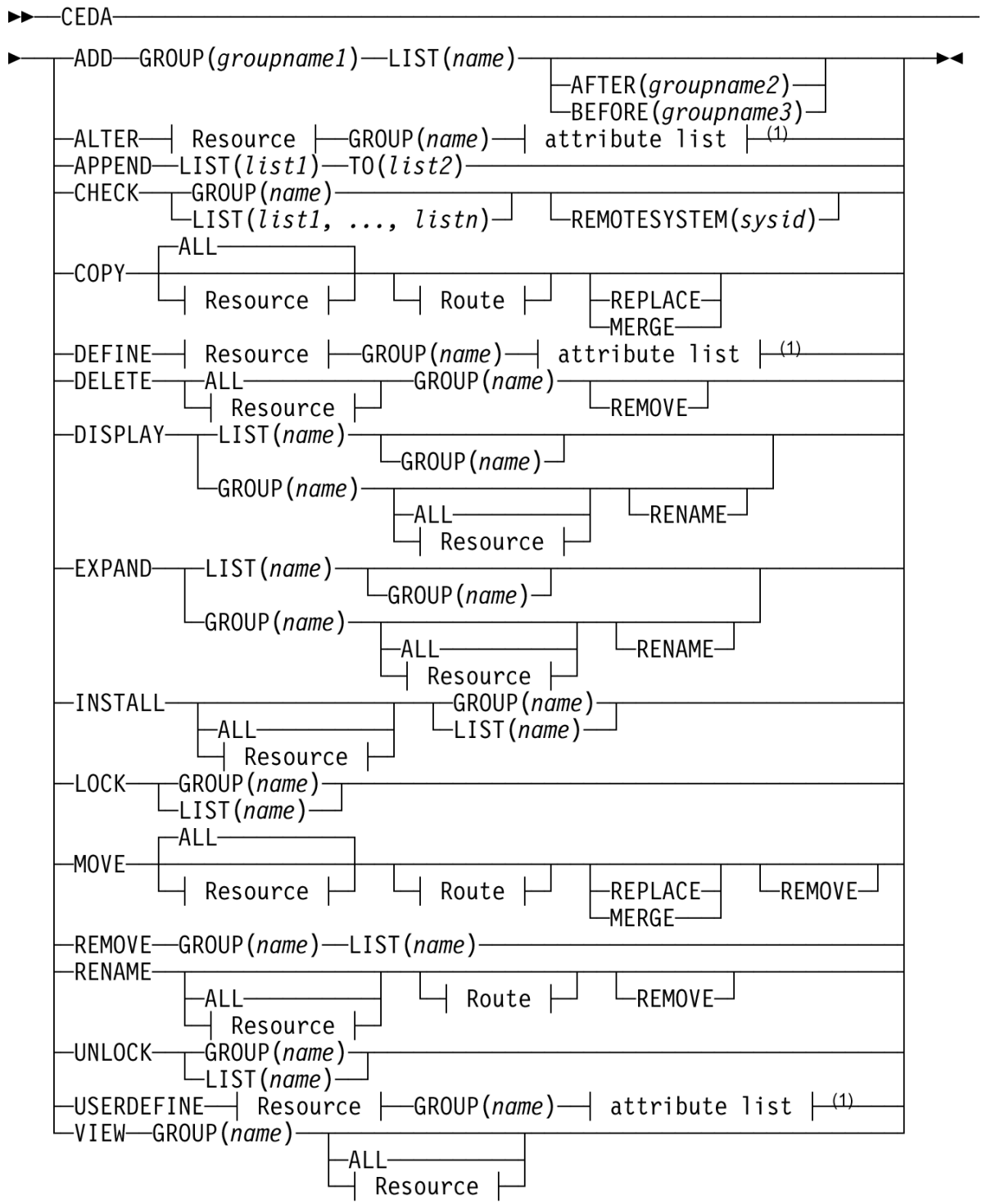
REMove
Removes a group name from a list.

RENname
Renames a resource definition.

UNlock
Removes a lock on a group or list which was previously put there by a CEDA LOCK command.

USerdefine
Instead of using CICS-supplied default values with the DEFINE command, you can set your own defaults and use the USERDEFINE command to create a new resource definition on the CSD file.

View
Lets you look at the resource definition attributes.



Note:

1 Consult the *CICS Resource Definition Guide* for information about the attributes applicable to each resource type.

http://publib.boulder.ibm.com/infocenter/cicsts/v3r1/index.jsp?topic=/com.ibm.cics.ts31.doc/dfha4/topics/dfha4_overview.htm

Appendix E

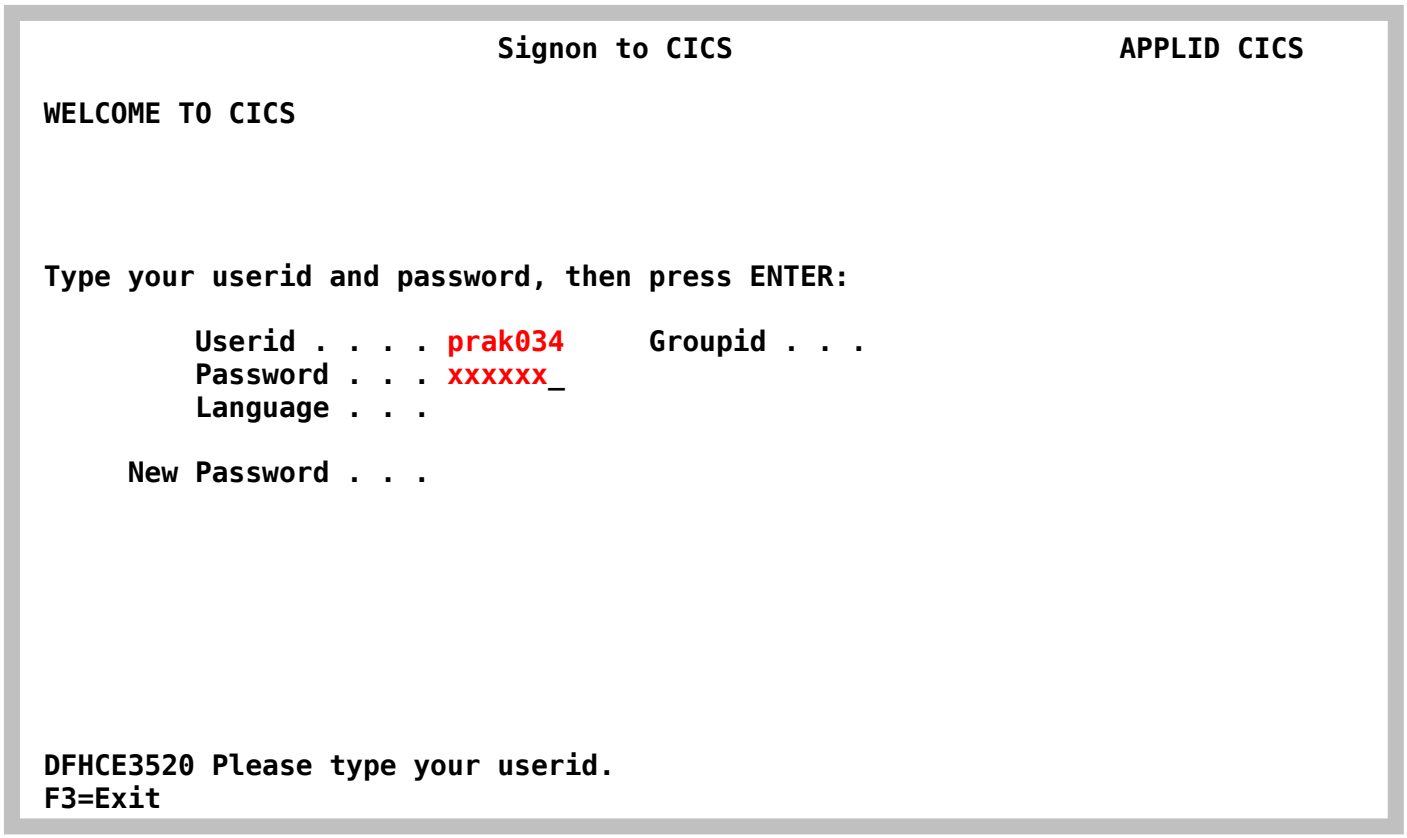
CEDA Define Parameters

```
Connection      Requestmodel
CORbaserver     Sessions
DB2Conn         TCpipservice
DB2Entry        TDqueue
DB2Tran         Terminal
DJar            TRANClass
DOctemplate     TRANSaction
Enqmodel        TSmodel
File            TYpeterm
Journalmodel    Urimap
Lsrpool         Webservice
Mapset
PARTItionset
PARTNer
PIpeline
PROCesstype
PROFile
PROGram
```

We used Mapset, PROGram, and TRANSaction in our first CICS tutorial. This Tutorial used PIpeline

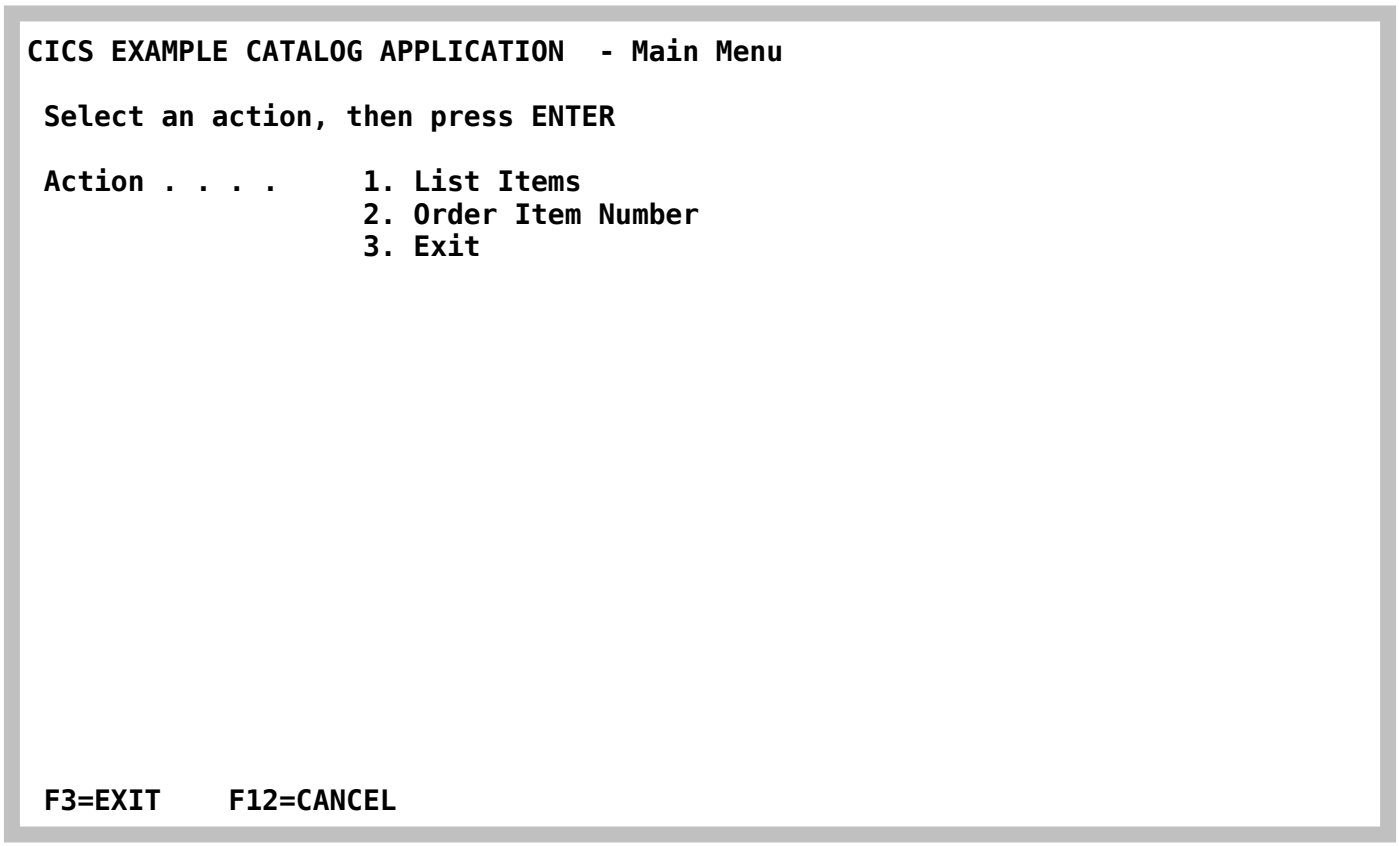
Appendix F
Ausführungsbeispiel der Transaktion unter Benutzung der BMS Präsentationslogik

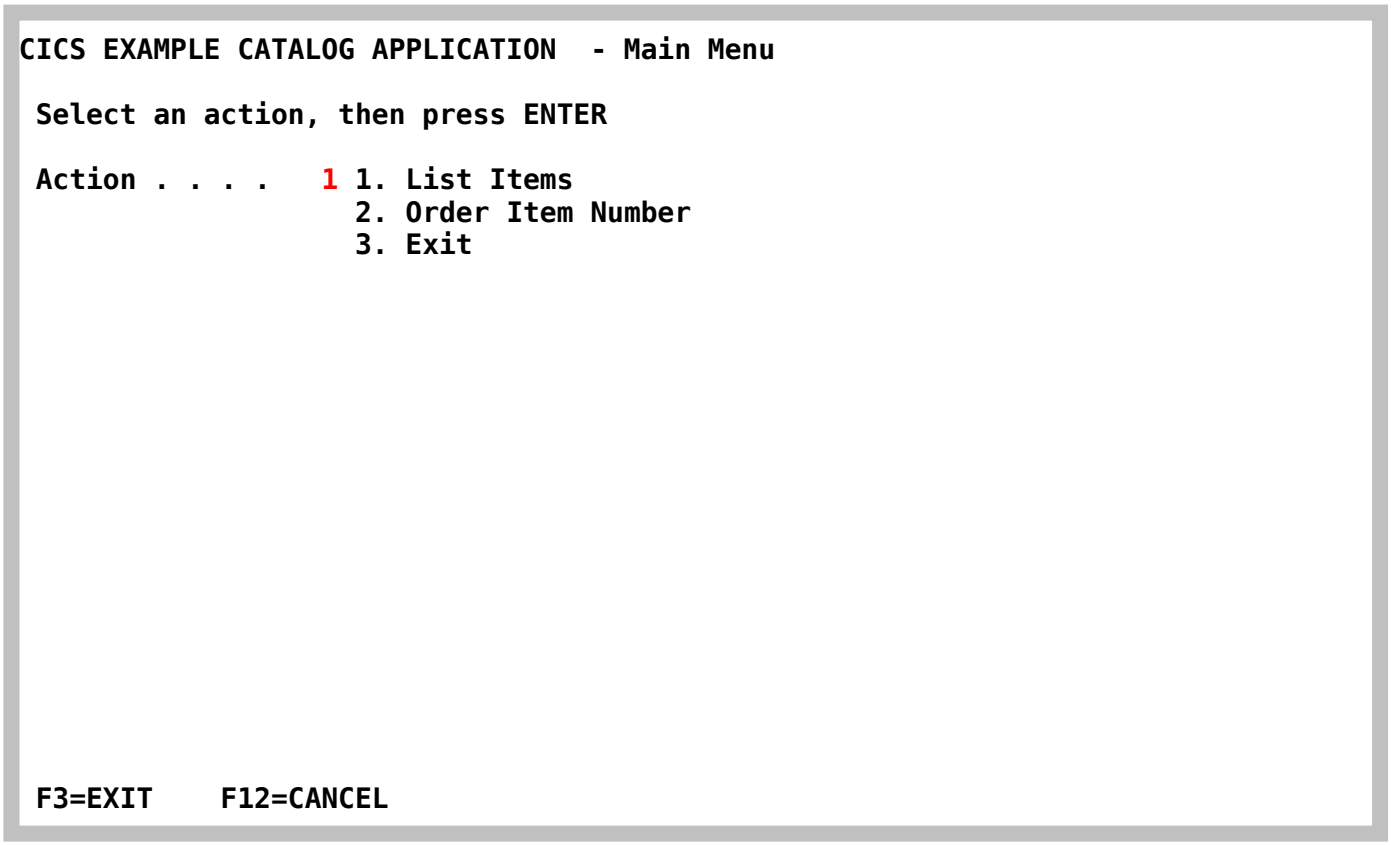
```
z/OS Z18 Level 0609                                IP Address = 212.125.104.108
                                                    VTAM Terminal = SC0TCP01

                     Application Developer System

                              //  OOOOOOO   SSSSS
                             //  OO    OO SS
                        zzzzzz //  OO    OO SS
                          zz  //  OO    OO SSSS
                         zz   //  OO    OO     SS
                       zz    //  OO    OO     SS
                     zzzzzz //   OOOOOOO  SSSS


                     System Customization - ADCD.Z18.*



 ===> Enter "LOGON" followed by the TSO userid. Example "LOGON IBMUSER" or
 ===> Enter L followed by the APPLID
 ===> Examples: "L TSO", "L CICS", "L IMS3270

l cics
```

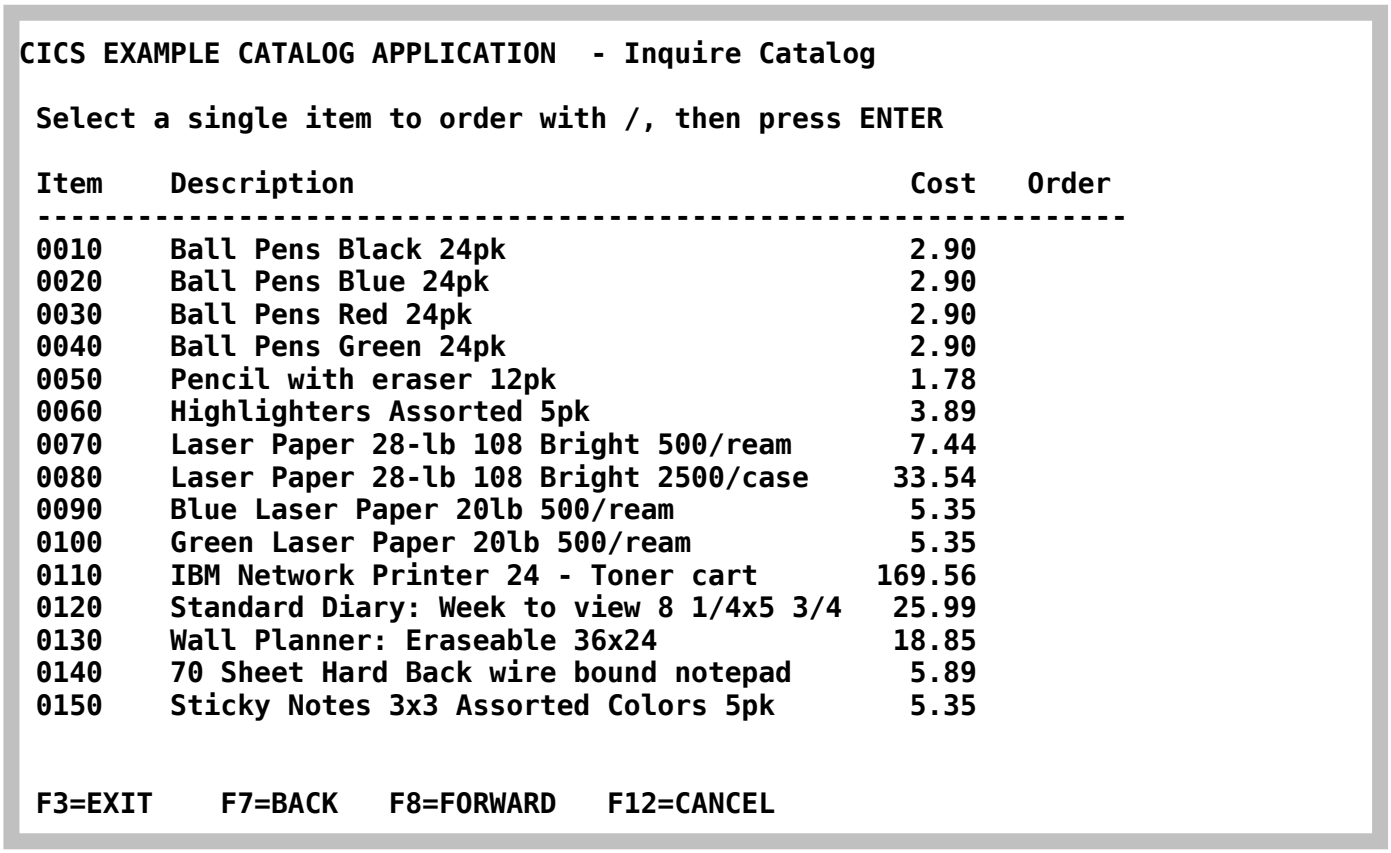

Signon to CICS APPLID CICS
WELCOME TO CICS
Type your userid and password, then press ENTER:
Userid . . . . prak034 Groupid . . .
Password . . . xxxxxx_
Language . . .
New Password . . .
DFHCE3520 Please type your userid.
F3=Exit

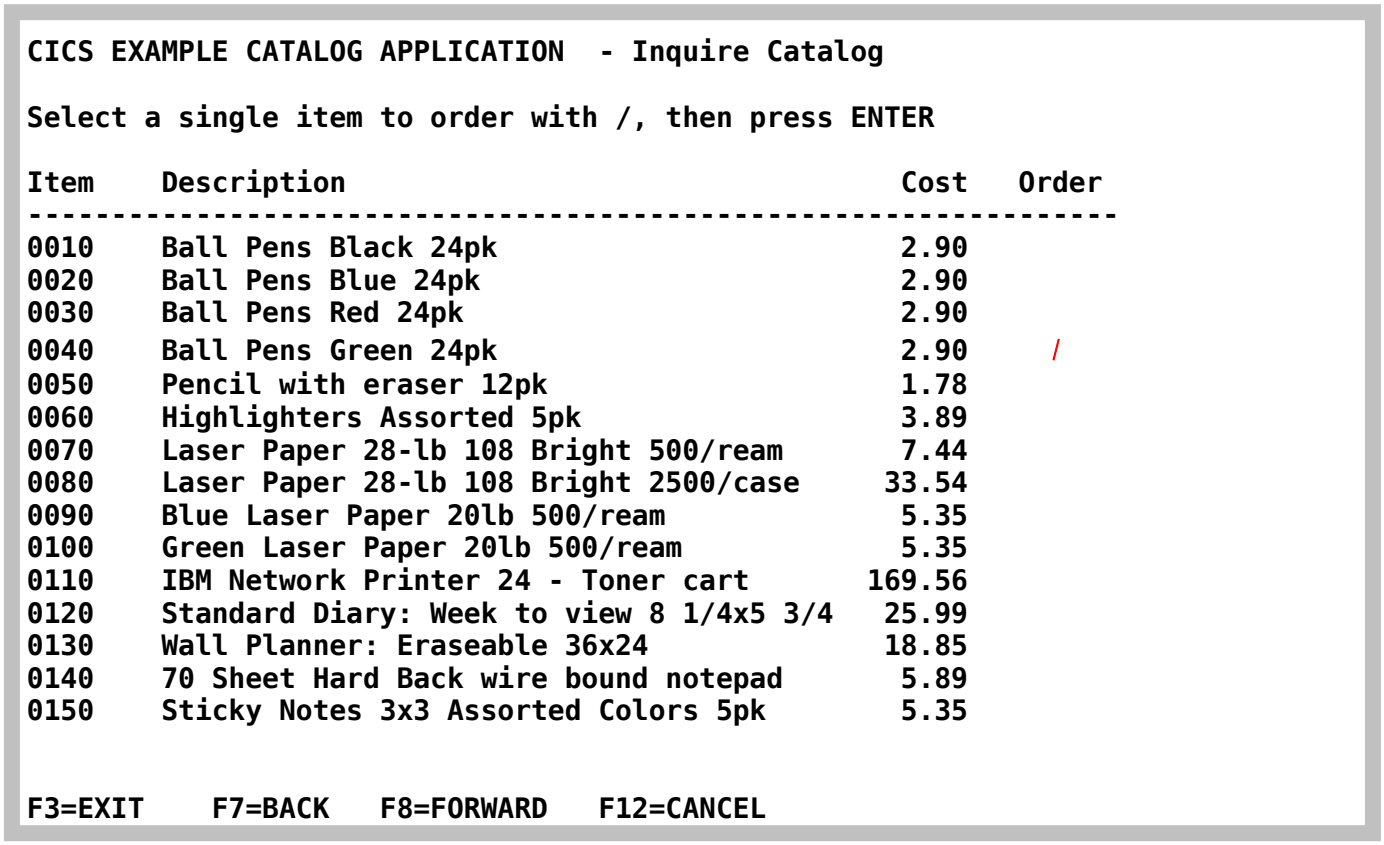
```
_


 DFHCE3549 Sign-on is complete (Language ENU).
```

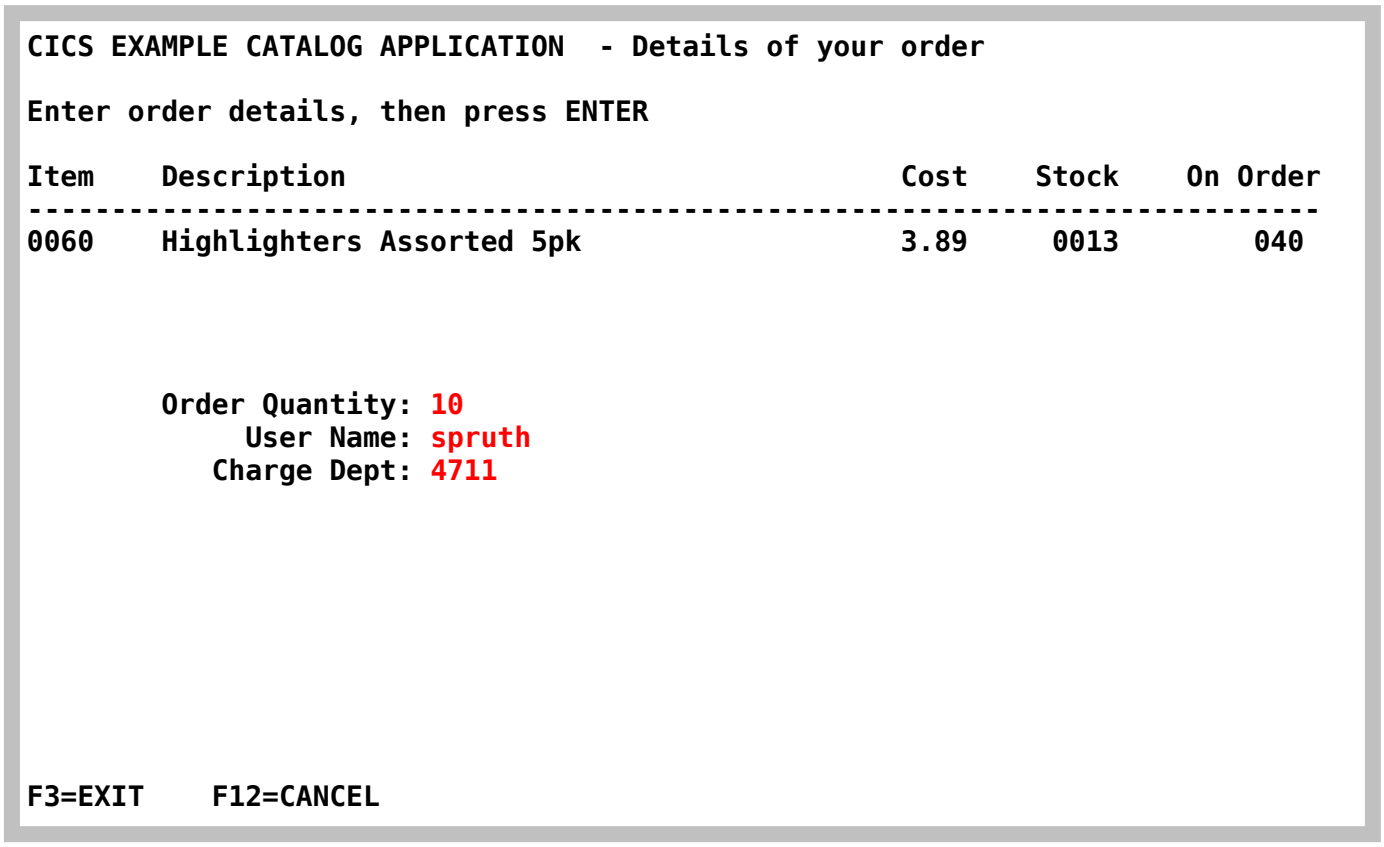

EGUI_
DFHCE3549 Sign-on is complete (Language ENU).

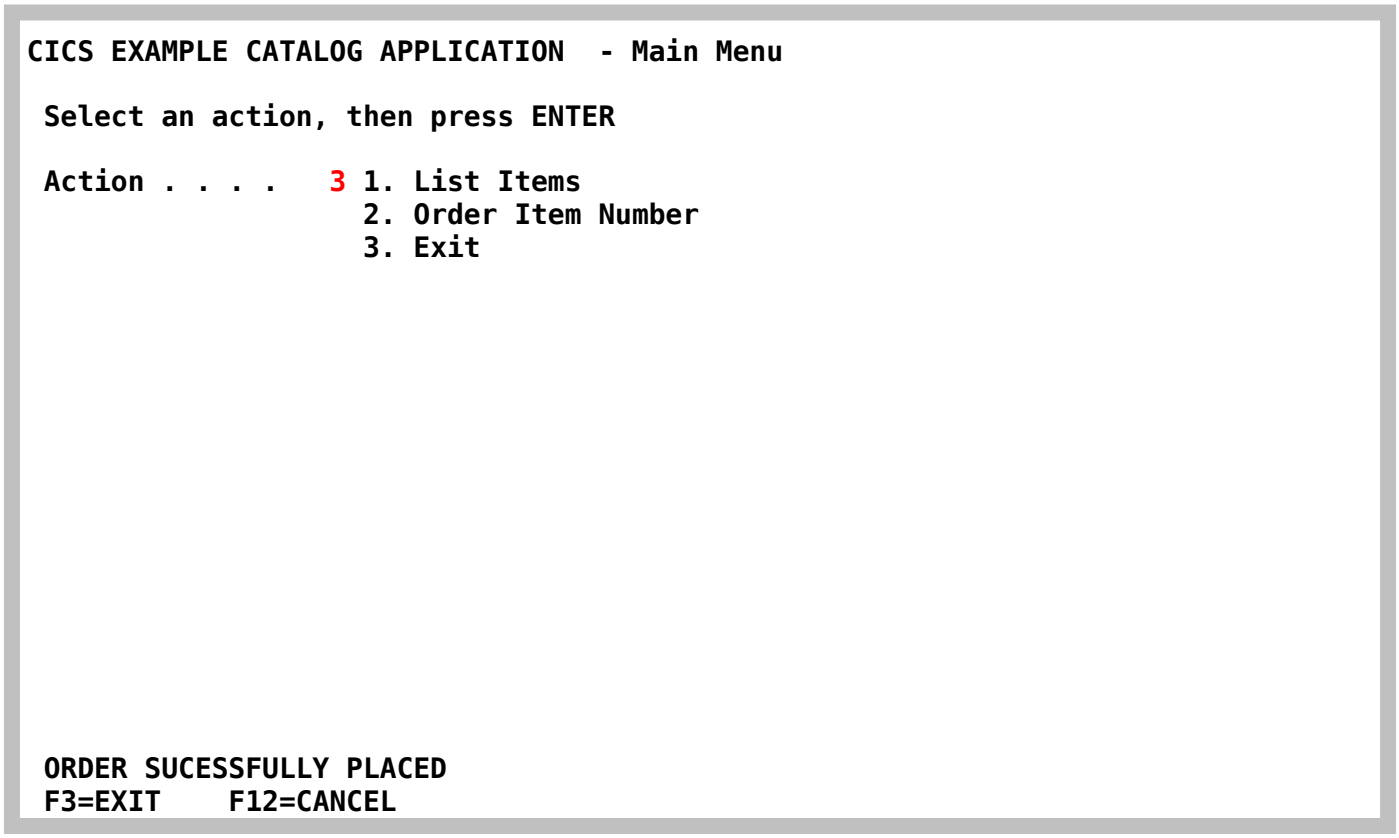

CICS EXAMPLE CATALOG APPLICATION - Main Menu
Select an action, then press ENTER
Action . . . .
1. List Items
2. Order Item Number
3. Exit
F3=EXIT
F12=CANCEL


CICS EXAMPLE CATALOG APPLICATION - Main Menu
Select an action, then press ENTER
Action . . . . 1 1. List Items
2. Order Item Number
3. Exit
F3=EXIT F12=CANCEL


CICS EXAMPLE CATALOG APPLICATION - Inquire Catalog
Select a single item to order with /, then press ENTER
Item Description Cost Order
0010 Ball Pens Black 24pk 2.90
0020 Ball Pens Blue 24pk 2.90
0030 Ball Pens Red 24pk 2.90
0040 Ball Pens Green 24pk 2.90
0050 Pencil with eraser 12pk 1.78
0060 Highlighters Assorted 5pk 3.89
0070 Laser Paper 28-lb 108 Bright 500/ream 7.44
0080 Laser Paper 28-lb 108 Bright 2500/case 33.54
0090 Blue Laser Paper 20lb 500/ream 5.35
0100 Green Laser Paper 20lb 500/ream 5.35
0110 IBM Network Printer 24 - Toner cart 169.56
0120 Standard Diary: Week to view 8 1/4x5 3/4 25.99
0130 Wall Planner: Eraseable 36x24 18.85
0140 70 Sheet Hard Back wire bound notepad 5.89
0150 Sticky Notes 3x3 Assorted Colors 5pk 5.35
F3=EXIT F7=BACK F8=FORWARD F12=CANCEL


CICS EXAMPLE CATALOG APPLICATION - Inquire Catalog
Select a single item to order with /, then press ENTER
Item Description Cost Order
0010 Ball Pens Black 24pk 2.90
0020 Ball Pens Blue 24pk 2.90
0030 Ball Pens Red 24pk 2.90
0040 Ball Pens Green 24pk 2.90 /
0050 Pencil with eraser 12pk 1.78
0060 Highlighters Assorted 5pk 3.89
0070 Laser Paper 28-lb 108 Bright 500/ream 7.44
0080 Laser Paper 28-lb 108 Bright 2500/case 33.54
0090 Blue Laser Paper 20lb 500/ream 5.35
0100 Green Laser Paper 20lb 500/ream 5.35
0110 IBM Network Printer 24 - Toner cart 169.56
0120 Standard Diary: Week to view 8 1/4x5 3/4 25.99
0130 Wall Planner: Eraseable 36x24 18.85
0140 70 Sheet Hard Back wire bound notepad 5.89
0150 Sticky Notes 3x3 Assorted Colors 5pk 5.35
F3=EXIT F7=BACK F8=FORWARD F12=CANCEL


CICS EXAMPLE CATALOG APPLICATION - Details of your order
Enter order details, then press ENTER
Item Description Cost Stock On Order
0060 Highlighters Assorted 5pk 3.89 0013 040
Order Quantity: 10
User Name: spruth
Charge Dept: 4711
F3=EXIT F12=CANCEL


CICS EXAMPLE CATALOG APPLICATION - Main Menu
Select an action, then press ENTER
Action . . . . 3 1. List Items
2. Order Item Number
3. Exit
ORDER SUCESSFULLY PLACED
F3=EXIT F12=CANCEL

```
Thank You for using the Catalog Application
```

Appendix G
Resources

Webtut03.zip contains resources as a zip Archiv. Download at

www.cedix.de/VorlesMirror/Band3/DiscoverCatalogManager.pdf

A description of the CICS catalog manager example application is available at

www.cedix.de/Vorles/Band3/Resources/webtut03.zip

and at

http://publib.boulder.ibm.com/infocenter/cicsts/v3r1/index.jsp?topic=%2Fcom.ibm.cics.ts31.doc%2Fdfhxa%2Ftopics%2Fdfhxa_t100.htm

and also in an IBM Redbook

Application Development for CICS Web Services, January 2010, SG24-7126-01
http://www.redbooks.ibm.com/redbooks/pdfs/sg247126.pdf

Schlusswort

Wir hoffen, mit der Veröffentlichung dieses e-Books einen Beitrag zur Weiterentwicklung von e-Learning und zur Verbesserung der Mainframe Ausbildung an den deuschsprachigen Hochschulen geleistet zu haben.

Wir befinden uns in einem sich schnell ändernden Umfeld. Für Verbesseungsvorschläge sind wir jederzeit sehr dankbar.

Prof. Dr. Martin Bogdan **Prof. Dr.-Ing. Wilhelm G. Spruth**