CICS Transaction Gateway-Architektur und Anwendung

Paul Herrmann, Wilhelm G. Spruth
Institut für Informatik, Universität Leipzig

Abstrakt

Die Programmiersprache Java hat sich in den letzten Jahren als ein leistungsfähiges Werkzeug für WWW-orientierte Anwendungen durchgesetzt. Mit den EJB's und dem J2EE-Standard bemüht man sich, einen objektorientierten Ansatz auch für die Geschäftslogik einzusetzen. Von dem Ziel, existierende Geschäftslogik durch neue EJB-Implementierungen zu ersetzen, sind wir jedoch noch weit entfernt. Dies gilt besonders für die größeren Unternehmen, die in der Regel z/OS mit IMS, CICS, VSAM, DB2, Oracle und Adabas auf ihrem zentralen Server einsetzen.

Vorhandene Java-Implementierungen von Internet-Anwendungen benutzen häufig Konnektoren, um mit Hilfe der Java Connector-Architektur (JCA) auf vorhandene Anwendungslogik zuzugreifen, die in der Form von Anwendungen unter CICS, IMS/DC, DB2 Stored Procedures oder Stapelverarbeitungsanwendungen auf dem z/OS Server vorliegen. Der vorliegende Beitrag erläutert, wie mit unterschiedlichen Konnektoren auf existierende CICS-Anwendungen zugegriffen werden kann. Hierbei wird besonders die Internet-Integration mit Hilfe des CICS Transaction Gateways beschrieben.

Eine erfolgreiche Integration des Internets in existierende unternehmerische IT-Strukturen ist ein wichtiges Thema für die Ausbildung des akademischen Nachwuchses. Es wird dargestellt, wie an der Universität Leipzig die Studenten mit praktische Übungen am Beispiel des CICS Transaction Gateways an dieses Ziel herangeführt werden.

Einführung

Moderne Client/Server-Systeme strukturieren ihre Anwendungen in die beiden Teile Geschäftslogik (Business Logik), auch als Backend und Präsentationslogik (Presentation Logic), auch als Frontend bezeichnet (s. Abb. 1).

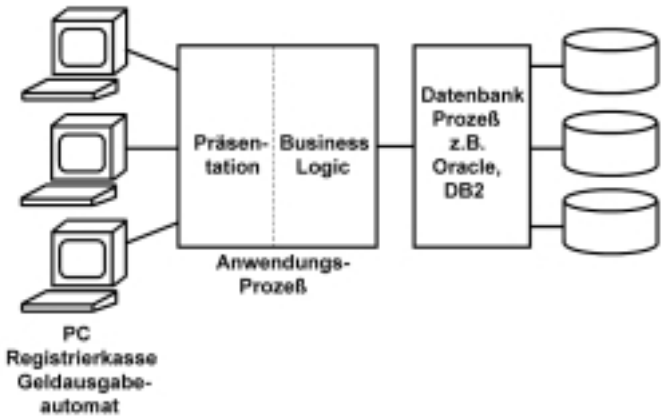


Abbildung 1: Client/Server-System

Für die Präsentationslogik haben sich Java-Technologien, besonders Servlets, Applets und Java Server Pages sowie ihre Microsoft Äquivalente als eine produktive Technologie bewährt.

Für die Business Logic wurde 1998 der J2EE-Standard und besonders der Enterprise Java Bean (EJB)-Standard entwickelt. Servlets und EJBs laufen in getrennten Servlet und EJB Containern, die wiederum in der Regel in einer gemeinsamen Java Virtuellen Maschine (JVM) untergebracht sind. EJBs benutzen für die Datenspeicherung häufig eine SQL-Datenbank. Für die Verbindung mit Klienten des World Wide Web wird ein getrennter Web Server eingesetzt, z.B. Apache. Diese Konfiguration ist in Abb. 2 wiedergegeben.

Web Application Server (WAS) implementieren Server für WWW-orientierte Anwendungen. Typischerweise sind die Anwendungen in Java geschrieben und werden als Byte Code ausgeführt.

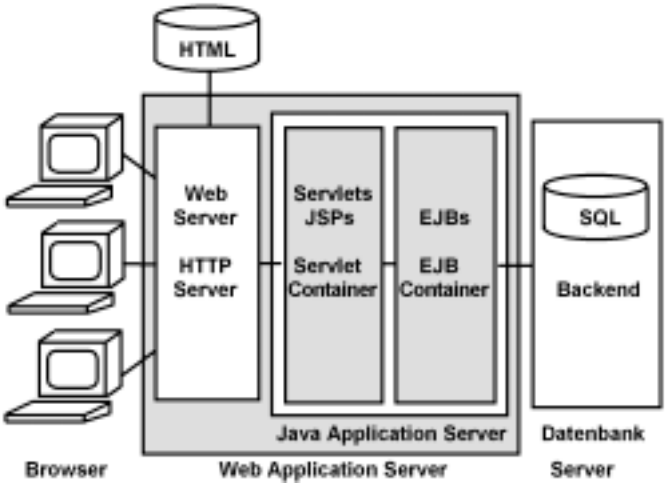


Abbildung 2: Web Application Server

Der EJB Container stellt den EJBs eine Reihe von Dienstleistungen zur Verfügung. Von besonderer Bedeutung ist hierbei der Java Transaction Service (JTS). Dieser kann durch einen Eintrag in den EJB Descriptor in Anspruch genommen werden und bewirkt eine Bean-Ausführung unter ACID-Bedingungen. Der EJB Container stellt hierfür die Funktionalität eines Transaktionsmonitors zur Verfügung. Die führenden Hersteller von Web Application Servern (BEA, IBM, Oracle und SAP) verwenden hierzu Komponenten der hauseigenen Transaktionsmonitore.

Bei der Entwicklung des J2EE-Standards ging man von einer Idealvorstellung aus, bei der z.B. die Finanzbuchhaltung einer Großbank in Zukunft ausschließlich in Java unter Verwendung von EJBs implementiert werden würde. Dass dies bis heute nur selten geschehen ist, hat vor allem vier Gründe:

1. Die Sicherstellung der ACID-Bedingungen erfordert bei unternehmenskritischen Anwendungen sehr detaillierte Java-Kenntnisse und umfangreiche Erfahrung [CZA01], [SAN04]. Abhilfe soll der kürzlich verabschiedete Java „Isolate“-Standard schaffen, der bisher aber in den existierenden WAS-Produkten noch nicht verfügbar ist [JCP04]. Eine zufriedenstellende Alternative, die „Persisten Reusable JVM“ [BEY05], ist bisher nur unter dem z/OS-Betriebssystem erhältlich.

2. In den letzten 40 Jahren entstanden zahlreiche unternehmenskritische Anwendungen, die in Programmiersprachen wie Cobol, PL/1 und ABAP/4 geschrieben sind. Nach Schätzungen stellen diese Anwendungen eine Investition von 10^{12} Dollar und 10^{7} Mannjahren dar [GRA99], [SPE99]. Es existiert weder das notwendige Geld noch die erforderliche Anzahl an Programmierern, eine Umstellung der Anwendungen vorzunehmen.

3. Darüber hinaus laufen diese sog. Legacy-Anwendungen zuverlässig und performant. Mit anderen Worten, sie haben im Durchschnitt einen 20 Jahre-Beta-Test absolviert.

4. Cobol-Programme stehen in dem Ruf, wartungsfreundlicher zu sein als die in andern Sprachen geschriebene Programme [KAP01].

Andererseits werden EJBs zunehmend in Anwendungen wie Web Services, Business Process Reengineering, Straight Through Processing [FRA03] und Service Oriented Architecture (SOA) eingesetzt.

Etwa 80 % aller Anwendungen in der Wirtschaft werden als Transaktionen ausgeführt. Sie laufen in den meisten Fällen unter Transaktionsmonitoren wie BEA Tuxedo, IBM CICS oder IMS, Oracle Stored Procedures oder SAP R/3. CICS (Customer Information Control System) der Firma IBM ist der Transaktionsmonitor, der weltweit am häufigsten eingesetzt wird. Momentan existieren ca. 30 Millionen CICS Terminals [HKS03], und die Anzahl der weltweiten CICS-Transaktionen pro Stunde entspricht etwa der Zahl der weltweiten WWW-Zugriffe im gleichen Zeitraum [HOR00].

2. Integration des World Wide Web mit existierenden Back End-Anwendungen

Die meisten Unternehmen benutzen heute mehrere Back-End Enterprise Inforationssysteme (EIS), welche Business-Transactionen verarbeiten und Geschäftsdaten in einer Datenbank verwalten. Beispiele für firmeneigene EIS sind:

- Enterprise Resource Planning (ERP)-Systeme (z.B. SAP)
- CICS- und IMS-Systeme, die Business Transactionen verarbeiten
- Unternehmensweite Datenbanken, auf die mit SQL Stored Procedures zugegriffen wird.
- Eine Kombination dieser Möglichkeiten.

Auf derartige Systeme wurde und wird häufig mit Client/Server-Verfahren zugegriffen, indem auf den Arbeitsplatzrechnern geeignete EIS-Klienten installiert werden. Hierbei eröffnet die Verbindung des Internets mit der existierenden IT-Infrastruktur zahlreiche neue Möglichkeiten. Dabei spielen moderne graphische Benutzeroberflächen (Graphical User Interface, GUI), die mit Hilfe von Java implementiert werden, eine entscheidende Rolle.

Graphische Benutzeroberflächen lassen sich komfortabel mit Hilfe von Java Applets und den äquivalenten Microsoft Dotnet-Verfahren implementieren. Hiervon wird auf Grund des relativ hohen Administrationsaufwandes zunehmend zu Gunsten Server-zentrierter Präsentationslogik Abstand genommen.

Aus dem Einsatz von Server-zentrierter Präsentationslogik ergeben sich zahlreiche weitergehende Möglichkeiten. Dies sind z.B. e-busines Strukturen wie Supply Chain Management (SMC), Customer Relation Management (CRM), Vernetzung der Warenwirtschaftssysteme unterschiedlicher Unternehmen sowie rollenbasiertes Identitätsmanagement, (um den Ende 2004 in Kraft getretenen neuen Bilanzierungsvorschriften gerecht zu werden). Daneben stehen moderne Verfahren um den Schwerpunkt Business Process Reengineering, siehe z.B. [FRA03]. Hierbei verliert Dotnet gegenüber Java an Bedeutung, weil Java für den Einsatz in heterogenen Serverlandschaften entscheidende Vorteile bietet.

Allen Fällen ist der Zugriff auf die existierenden EIS Backend-Systeme gemeinsam. Die hierbei verwendete Vorgehensweise wird am Beispiel von CICS als dem wichtigsten Backend-System erläutert.

3. Zugriffsmöglichkeiten auf CICS

Eine CICS-Anwendung besteht normalerweise aus einem Business Logic-Teil und einem Präsentations Logic-Teil. Diese beiden Teile kommunizieren über einen internen Puffer miteinander, der in der CICS-Terminologie als COMMAREA bezeichnet wird. Beispielsweise würde ein in C/C++ geschriebenes CICS Business Logic-Programm seine Ausgabedaten in Form einer Struktur in den COMMAREA-Bereich stellen. Die CICS Präsentations Logic übernimmt diese Daten und erstellt daraus eine Benutzer-freundliche Darstellung auf dem Bildschirm des Klienten.

In der Vergangenheit erfolgte die Ausgabe typischerweise in einer alpha-numerischen Darstellung auf einem Bildschirm (oder Bildschirmfenster) mit 24 Zeilen zu je 80 Zeichen pro Zeile (3270-Bildschirm). Diese Art der Präsentation verwendet eine CICS-Komponente, den „Basic Mapping Support“ (BMS). Es existieren Einrichtungen wie die EPI-Schnittstelle, die eine graphische Darstellung der BMS-Ausgabe möglich machen. Die Darstellungsmöglichkeiten sind hierbei jedoch eingeschränkt.

Es sind deshalb mehrere alternative Möglichkeiten entstanden, die eine volle graphische Funktionalität der Datendarstellung auf dem Bildschirm erlauben. Hierbei greift die Präsentations Logic mit Hilfe geeigneter Schnittstellen auf den COMMAREA-Puffer zu.

SOA- for-CICS – ein Dienst des CICS Transaction Servers – ist in der Lage, auf die COMMAREA eines CICS-Programms zuzugreifen. Damit werden CICS-Programme zu Anbietern und Benutzern eines in WSDL (Web Services Definition Language) definierten Web Services. In diesem Fall empfängt die SOAP-for-CICS-Schnittstelle eine Anfrage, verarbeitet die SOAP Header und entpackt die XML-Anfragenachricht. Letztere wird in ein COMMAREA-Format übersetzt, und es wird das CICS-Programm mit der Business Logic aufgerufen. Nachdem das CICS-Programm erfolgreich abgelaufen ist, wird eine entsprechende XML-Antwort erzeugt, die wiederum mit SOAP-Headern versehen und an den e-Business-Client zurückgegeben wird.

Eine weitere Möglichkeit zur Integration von CICS-Programmen in e-Business-Lösungen stellt der Einsatz von Enterprise JavaBeans (EJBs) dar. Hierzu enthält der CICS Transaction Server einen Corba-fähigen EJB Container, der in der Lage ist, Session Beans zu verarbeiten. Enterprise JavaBeans ermöglichen das Aufrufen von Methoden entfernter Java-Objekte über ein Netzwerk. Solche entfernten Objekte können entweder durch Suche in einem Namensdienst wie LDAP (Lightweight Directory Access Protocol) oder durch Referenzen aus anderer Quelle aufgefunden werden. Nach dem Erhalten der EJB-Anfrage wird innerhalb des CICS Transaction Server der integrierte Methodenaufruf vom Object Request Broker (ORB) dekodiert. Dieser ruft auch die entsprechende Methode der Enterprise Java Bean auf, welche wiederum die Business Logic ausführt und den Rückgabewert des darunter liegenden CICS-Programms als Ausgabe veranlasst.

Weiterhin ist ein nachrichtenbasierter Ansatz mit Hilfe von WebSphere MQ einsetzbar, einem Message Based Queuing (MBQ)- bzw. Message Oriented Middleware (MOM)-Produkt, das den einfachen Informationsaustausch über mehrere Plattformen hinweg ermöglicht. Somit werden vorhandene e-Business-Anwendungen miteinander verbunden. Wenn über das Netzwerk eine Nachricht eintrifft, startet der im CICS Transaction Server ablaufende WebSphere MQ-Trigger-Monitor den Nachrichtenadapter. Letzterer formt die Nachricht um und ruft das Business Logic-Programm auf. Die Antwort wird über die in der Nachricht definierten Antwort-Queue gesendet. Darüber hinaus ergibt sich bei WebSphere MQ die Option, die WebSphere MQ DPL Bridge zu verwenden. Diese leitet eine Nachricht von einer Eingabe-Queue über die COMMAREA an ein Business Logic-Programm weiter. Die Nachricht muss jedoch schon vom e-Business-Client so formatiert werden, dass das Business Logic-Programm sie verarbeiten kann.

Eine einfache Möglichkeit bildet die Verwendung des im CICS-System integrierten HTTP-Listeners (CICS Web Support) in Verbindung mit einem Nachrichtenadapter. Hier empfängt der HTTP-Listener eine HTTP-Anfrage und leitet sie an den Nachrichtenadapter weiter. Dieser extrahiert die HTTP-Benutzerdaten, also ein HTML-Dokument, das wiederum in ein COMMAREA-Format übersetzt wird und das Business Logic-Programm aufruft. Häufig besteht der e-Business Client aus einem Webbrowser, so dass die Antwort des Business Logic-Programms normalerweise als HTML- oder XML-Dokument formatiert wird.

TCP/IP-Sockets sind gleichfalls für die Integration in e-Business-Anwendungen geeignet. Dabei stellt der IBM-z/OS Communications Server die TCP/IP Socket-Schnittstelle zu CICS bereit. Der CICS Sockets Listener stellt nach Erhalt einer TCP/IP-Anforderung einen so genannten "Child Server" zur Verfügung, der wiederum einen Nachrichtenadapter aufruft. Dieser regelt dann die Ausführung des Business Logic-Programms. Es handelt sich hier um eine Low Level-Lösung ohne Unterstützung für Transaktionen und Sicherheit, so dass sich der Programmierer selbst darum kümmern muss.

4. Das CICS Transaction Gateway

Die Zugriffsmöglichkeit auf den CICS Transaction Server über das *CICS Transaction Gateway (CTG)* soll genauer untersucht werden. Das CTG stellt einen Konnektor (CICS ECI Ressourcen-Adapter) für den WebSpere Application Server mit Zugriff auf den CICS Transaction Server zur Verfügung und ermöglicht Webbrowsern und Netzwerkcomputern einen sicheren und einfachen Zugriff auf CICS-Anwendungen. Es sind viele verschiedene Konfigurationen möglich, bei denen Standard-Internetprotokolle zum Einsatz kommen. Das CICS Transaction Gateway kann zum Beispiel als Connektor für den IBM WebSphere Application Server (einem Web Application Server) verwendet

werden. Es wird eine Programmierschnittstelle bereit gestellt, durch die Java-Programmierer die Funktionen der J2EE-Plattform (Java 2 Enterprise Edition) nutzen können.

Die J2EE-Plattform definiert einen Standard zur Entwicklung mehrschichtiger e-Business-Anwendungen. Sie stellt standardisierte modularisierte Komponenten sowie zugehörige Dienste bereit, so dass viele Anwendungsdetails automatisch und ohne komplexe Programmierarbeit behandelt werden. Die J2EE-Architektur (s. Abbildung 3) baut auf *Java 2 Standard Edition (J2SE)* auf und nutzt zum Beispiel ebenfalls die JDBC-Bibliothek zum Datenbankzugriff, die Technologie *CORBA (Common Object Request Broker Architecture)* zur Interaktion mit vorhandenen Ressourcen sowie ein Sicherheitsmodell zum Schutz von Anwendungsdaten. Zusätzlich zu diesen grundlegenden Funktionen unterstützt die J2EE-Plattform *Enterprise JavaBeans (EJBs)*, *Java Servlets* und *Java Server Pages (JSPs)*.

Diese Server-seitigen Komponenten können nicht in einer normalen Java Virtual Machine ausgeführt werden. Die Ablaufumgebung der J2EE-Plattform ist daher der Web Application Server. Im Allgemeinen beschreibt man diesen als eine »Serverbox«, wobei diese eine höchst-skalierbare und zuverlässige Plattform implementiert, auf der komplexe e-Business-Anwendungen ausgeführt werden können. Es besteht die Möglichkeit, eine Menge von Clients gleichzeitig zu bedienen. Dabei kann es sich sowohl um Benutzer als auch um andere e-Business-Anwendungen handeln. Ein Beispiel für einen Web Application Server ist der WebSphere Application Server von IBM, der praktisch für alle Hard- und Software-Plattformen verfügbar ist.

Die entscheidende Schnittstelle zwischen Backend-Systemen wie CICS und einem Web Application Server ist die J2EE Connector Architecture (JCA). Diese definiert unter anderem Java-Schnittstellen (Konnektoren), die auf vorhandene Backend-Systeme zugreifen können. Ein Konnektor wird auch als Ressourcen-Adapter bezeichnet und ist fest im Anwendungs-Server installiert. Er kann als stehende Verbindung zum Backend-System aufgefasst werden, die von vielen Clients verwendet werden kann. Ein Bestandteil des CICS Transaction Gateway ist ein solcher Konnektor, der CICS-ECI-Ressourcen-Adapter, für den WebSphere Application Server, der den Zugriff auf den CICS Transaction Server ermöglicht.

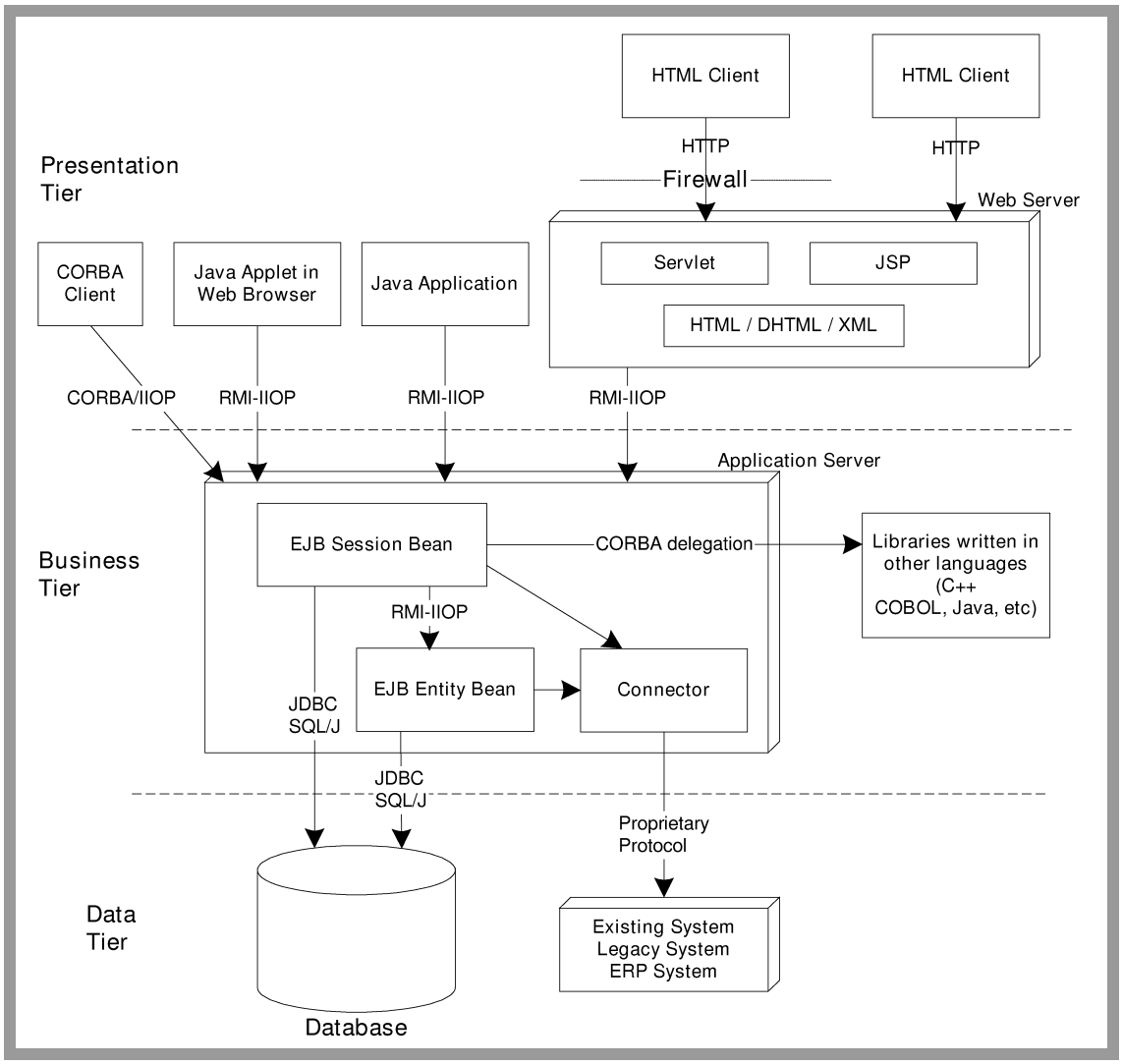


Abbildung 3: J2EE-Architektur [ROE99]

Beim CTG gibt es prinzipiell drei Möglichkeiten, den WebSphere Application Server in Verbindung mit dem CICS ECI Ressourcen-Adapter einzusetzen, um auf einen CICS-Server unter z/OS zuzugreifen:

- *Topologie 1:* Der WebSphere Application Server und das CICS Transaction Gateway befinden sich auf verschiedenen Nicht-z/OS-Rechnern, der CICS Transaction Server auf einem z/OS-Server.
- *Topologie 2:* Der WebSphere Application Server befindet sich auf einem Nicht-z/OS-Rechner, während das CICS Transaction Gateway und der CICS Transaction Server im z/OS-System installiert wurden.
- *Topologie 3:* Sowohl der WebSphere Application Server als auch das CICS Transaction Gateway und der CICS Transaction Server befinden sich im z/OS-System.

Wir beschränken uns hier auf die Topologie 3, da diese am höchsten integriert ist und somit auch die besten Eigenschaften bezüglich Sicherheit und Leistungsfähigkeit bietet. Außerdem können so die Vorteile der z/OS-Plattform in den Bereichen Sicherheit und Verfügbarkeit ausgenutzt werden. Die Abbildung 4 veranschaulicht dieses Szenario.

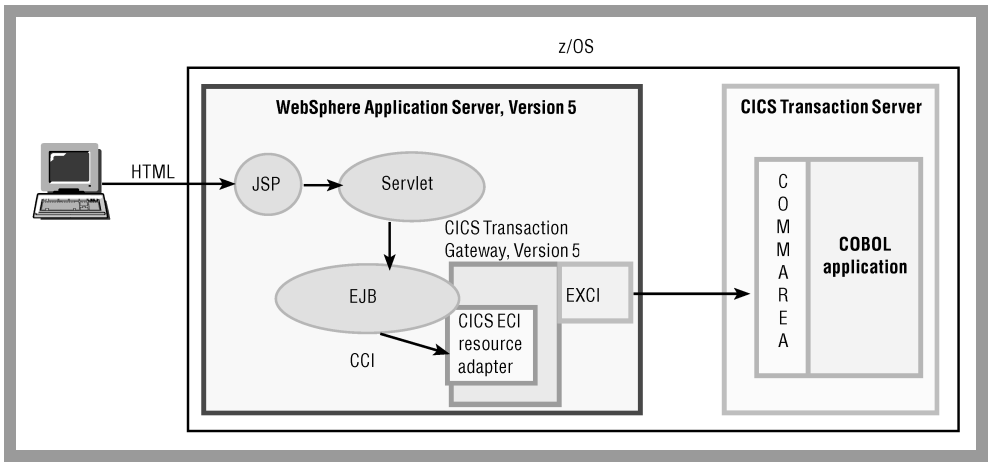


Abbildung 4: Topologie [IWC]

In der folgenden Aufzählung soll für diese Topologie der Ablauf des Zugriffs vom WebSphere Application Server auf CICS sowie der Datenfluss vom Browser des Client-PCs bis hin zu CICS und wieder zurück veranschaulicht werden:

1. Der Webserver lädt und initialisiert das Servlet. Dies kann zum Beispiel geschehen, wenn der Webbrowser oder Netzwerkcomputer eine Anfrage an das Servlet stellt (zum Beispiel über ein Formular in einer HTML- oder JSP-Datei).
2. Das Servlet sucht per JNDI nach der EJB und ruft mit den entsprechenden Parametern eine geeignete Methode der Bean auf.
3. Abbildung 5 enthält den in der Enterprise JavaBean enthaltenen Code. Diese sucht ebenfalls per JNDI nach dem im WebSphere Application Server installierten CICS-ECI-Ressourcenadapter und erhält eine Verbindung aus dessen Verbindungspool (Abbildung 5, Zeilen 1 und 4).

```
1  ConnectionFactory cf = <Lookup from JNDI namespace>
   ECIConnectionSpec cs = new ECIConnectionSpec();
   cs.setXXX();    //Set any connection specific properties

4  Connection conn = cf.getConnection( cs );
   Interaction int = conn.createInteraction();
   ECIInteractionSpec is = new ECIInteractionSpec();
   is.setXXX();    //Set any interaction specific properties

8  RecordImpl in = new RecordImpl();
   RecordImpl out = new RecordImpl();

10 int.execute( is, in, out );
   int.close();
   conn.close();
```

Abbildung 5: Aufruf des Ressourcenadapters [CGP]

4. Aus dieser Verbindung kann eine so genannte Interaktion erstellt werden (Abbildung 5, Zeile 5), die gestartet werden kann. Als Eingabewerte dieses Startbefehls werden die aus dem Servlet erhaltenen Parameter sowie eine Ein- und eine Ausgabevariable für die COMMAREA übergeben (Abbildung 5, Zeile 10).
5. Erst jetzt wird eine Verbindung zu CICS hergestellt, und es werden die entsprechenden Parameter übergeben.

6. CICS ruft das in den Parametern angegebene Programm auf, das seine Ausgabe in die COMMAREA schreibt.
7. Nach erfolgreicher Ausführung stehen in der Ausgabevariable der COMMAREA die von CICS erhaltenen Daten.
8. Diese Daten werden nun von der EJB an das Servlet weitergegeben.
9. Dieses Servlet erstellt eine HTTP-Antwort, die zurück an den Webbrowser und somit auf den Bildschirm des Benutzers gesendet wird.

Bei diesem Ablauf bleibt das verwendete CICS-Programm unverändert bestehen und kann schon viele Jahre oder gar Jahrzehnte alt sein. Das CICS-Programm merkt nicht einmal, dass es hier vom CICS Transaction Gateway und somit von einem Benutzer aus dem World Wide Web aufgerufen wird.

5. Zugriff auf weitere Backend-Systeme

Auch andere Backend-Systeme neben CICS können in e-Business-Anwendungen eingebunden werden. Ein Beispiel hierfür ist IMS (Information Management System), ein hierarchisches Datenbanksystem von IBM, das in vielen kritischen Bereichen wie Banken, Versicherungen und Verwaltungsseinrichtungen eingesetzt wird. Das dabei verwendete hierarchische Datenmodell mag veraltet erscheinen, jedoch sprechen Leistungsfähigkeit, Verfügbarkeit und die relativ geringen Kosten für IMS.

Die Struktur des Zugriffs auf IMS ähnelt der von CICS. Es gibt das relativ komplexe Produkt „IMS Connect“, das viele Bestandteile zur Verbindung mit IMS enthält und damit in etwa mit dem CICS Transaction Gateway vergleichbar ist. Auch hier existieren mehrere Topologien, die nach der Verteilung der einzelnen Komponenten untergliedert sind. Ein Bestandteil von IMS Connect ist der so genannte IMS Connector for Java (s. Abbildung 6). Letzterer ist mit dem CICS ECI-Ressourcenadapter des CICS Transaction Gateway vergleichbar und kann ebenfalls im WebSphere Application Server für z/OS installiert werden. Die Topologie ähnelt dann auch sehr der des CICS Transaction Gateway.

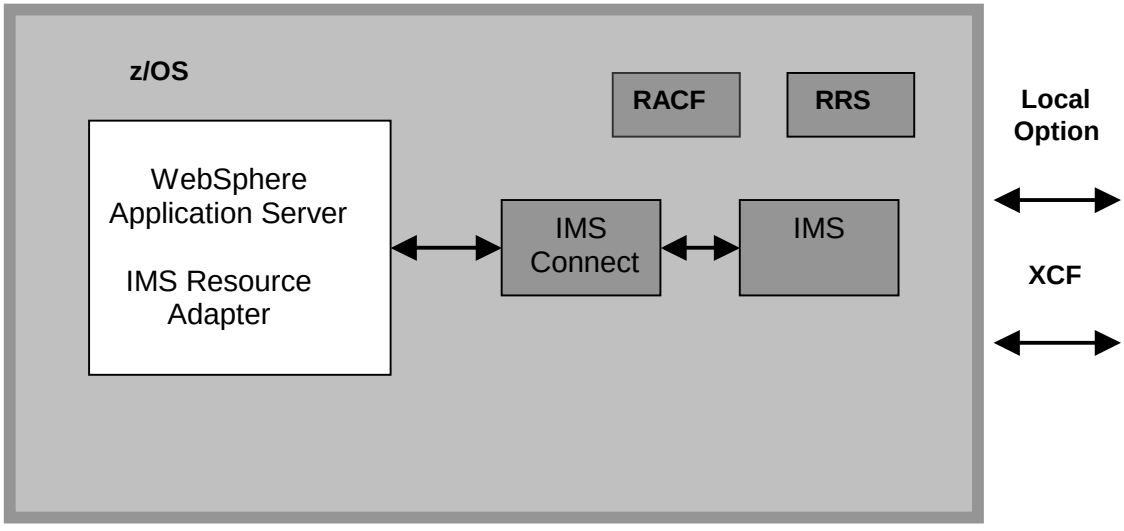
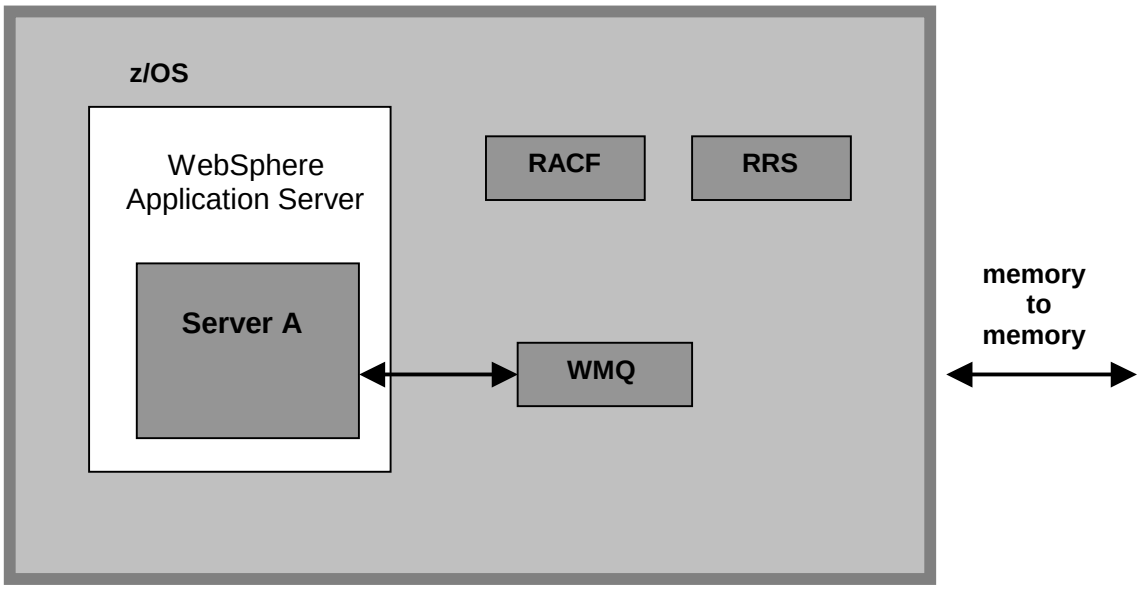


Abbildung 6: Zugriff auf IMS [ICA]

Die Vorteile dieser lokalen Konfiguration liegen wiederum in der Leistungsfähigkeit, Sicherheit und Verfügbarkeit der z/OS-Plattform.

Auch auf IBM WebSphere MQSeries, ein IBM-Produkt zum Nachrichtenaustausch, kann über den WebSphere Application Server für z/OS zugegriffen werden. Hier funktioniert der Zugriff jedoch etwas anders: WebSphere MQ stellt einen native Code zur Verfügung, durch den der WebSphere Application Server direkt mit MQ kommunizieren kann. Dabei müssen WebSphere Application Server und WebSphere MQ auf ein und demselben z/OS-Server installiert sein (s. Abbildung 7).



Abildung 7: Zugriff auf WebSphere MQ [ICA]

Als abschließendes Beispiel zum Zugriff auf ein Enterprise Information System soll IBM DB2 für z/OS dienen. Dabei handelt es sich um eines der erfolgreichsten und leistungsfähigsten relationalen Datenbankprodukte. Wieder soll der Zugriff lokal erfolgen, so dass sich der WebSphere Application Server und DB2 in ein und demselben z/OS-System befinden. Hier ist die offensichtlichste Zugriffsmöglichkeit JDBC (Java DataBase Connectivity). Im Prinzip gibt es vier verschiedene JDBC-Treiber T1 bis T4, wobei T1 und T2 einen lokalen und T3 und T4 einen Zugriff über das Netzwerk ermöglichen. Der T1-Treiber ist der älteste und wird im Allgemeinen nicht mehr verwendet. Also bleibt für diese Topologie der T2-Treiber. Dabei führt das JDBC API plattformspezifischen Code aus, um auf DB2 zuzugreifen. Der JDBC-Aufruf wird also vom T2-Treiber in SQL-Befehle umgewandelt und von der RRS Attachment Facility (RRSAF) ausgeführt. Ein Two-Phase-Commit-Protokoll ist möglich. Die Abbildung 8 veranschaulicht die Konfiguration.

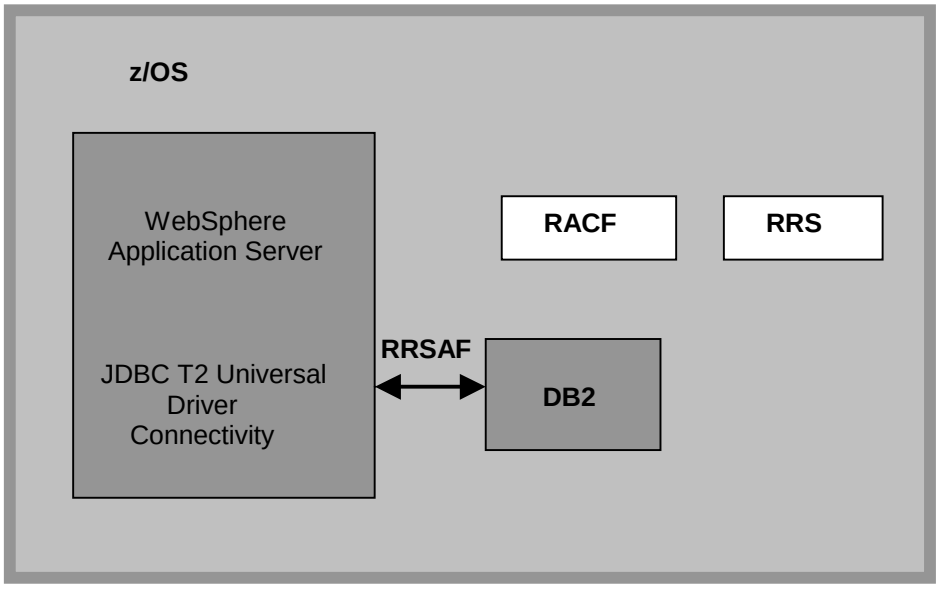


Abbildung 8: Zugriff auf DB2 [ICA]

6. Anwendungsbeispiele auf dem z/OS Rechner der Universität Leipzig

Das Institut für Informatik der Universität Leipzig unterhält einen eigenen z/OS Rechner (IP-Adresse: 139.18.4.35), der ausschließlich der studentischen Ausbildung dient. Er wird vor allem von Studenten für die Durchführung von Übungsaufgaben genutzt. Für diese Übungen existieren detailliert ausgearbeitete Anleitungen. Bei diesen sogenannten Tutorials liegt ein Schwerpunkt auf der Internet-Integration von z/OS-Anwendungen, besonders auch von CICS-Anwendungen..

Ein Anwendungs-Beispiel ist auf der Web-Seite http://jedi.informatik.uni-leipzig.de → Tutorials z/OS → Tutorial 18 (CICS Transaction Gateway) für jeden Interessenten ausführlich beschrieben und kann nach Erhalt einer User ID und eines Passwortes bearbeitet werden.

Das Ziel dieser Anwendung besteht darin, ein CICS-Programm zu schreiben, das mit dem CICS Transaction Gateway für das Web verfügbar gemacht wird. Dazu wird auf dem WebSphere Application Server für z/OS eine J2EE-Anwendung installiert, die über die Java-Klassen des CICS Transaction Gateway auf das erstellte CICS-Programm zugreift.

Zuerst wird ein CICS-Programm namens "CTG20" erstellt und installiert (s. Tutorial). Letzteres gibt keine Daten auf dem Bildschirm aus, sondern schreibt einen Namen in die COMMAREA (Communications Area).

Anschließend wird die J2EE-Anwendung (CTGTesterCCI.ear) erarbeitet, die über das CICS Transaction Gateway auf das installierte CICS-Programm zugreifen soll. Diese J2EE-Anwendung erhält mit dem Programm "Patch.exe" eine eindeutige ID.

Die veränderte J2EE-Anwendung ist auf dem WebSphere Application Server zu installieren. Dazu muss die im WebSphere Application Server V.5 neu eingeführte "Administrations-Konsole" verwendet werden, die eine einfache und stabile Oberfläche zur Verwaltung des WebSphere Application Server bietet.

Zuletzt soll das Beispiel mit einem Browser getestet werden. Der vom CICS-Programm in die COMMAREA geschriebene Name erscheint dabei auf dem Bildschirm, so dass die Verbindung zwischen CICS und dem Web hergestellt ist (s. Abbildung 4).

Tutorials 11 und 12 enthalten eine ähnlich Übungsaufgabe, bei der der CICS-Zugriff mit Hilfe eines Java Klienten, MQSeries und der MQSeries-CICS Bridge erfolgt.

SQLJ, Servlet und EJB-Zugriffe auf z/OS Anwendungen vervollständigen das Ausbildungsangebot.

Gruppen von Studenten an den folgenden Hochschulen:

- FH Bochum
- Technische Universität Chemnitz
- FH Darmstadt
- it-Akademie Bayern
- Universität Leipzig
- FH Lüneburg
- Universität Moskau
- FH Schmalkalden
- Universität Tübingen

absolvieren Übungen auf dem z/OS-Rechner am Institut für Informatik der Universität Leipzig, indem sie sich über das Internet einloggen und die erwähnten Tutorials durcharbeiten. Die Qualität der Tutorials ist ausreichend, um sie als e-Learning-Werkzeug für Studenten ohne vorherige z/OS-Kenntnisse einzusetzen.

Wir unterstützen einzelne Studenten an einigen weiteren Universitäten und Fachhochschulen, welche die Übungen ohne Anleitung durchführen. Weiterhin existieren einige teilnehmende Studenten außerhalb Deutschlands. Für die Mehrzahl der Tutorials ist eine englische Übersetzung verfügbar, die eine Nutzung außerhalb des deutschsprachigen Raums möglich macht.

7. Zusammenfassung und Ausblick

Die wachsende Integration des Internets in die existierende IT-Infrastruktur der Unternehmen ist in großem Umfang mit dem Einsatz von Java verbunden. Dabei werden die vorhandenen Back-End Enterprise Inforationssysteme (EIS) in die Java-Landschaft integriert. Dies geschieht häufig unter Verwendung von Konnektoren der Java Connection Architecture (JCA). Das CICS Transaction Gateway ist ein typisches und häufig eingesetztes Beispiel. Ähnliche Konnektoren existieren für andere Transaktionssysteme wie IMS und SAP R/3 oder für Datenbanken wie DB2 und Oracle.

Ein existierendes Problem ist die Tatsache, dass die meisten Spezialisten heute entweder über Kenntnisse und Erfahrungen in der z/OS-, CICS-, IMS- oder SAP R/3-Umgebung oder alternativ in der Java/Internet-Welt verfügen. Experten, die auf beiden Gebieten über Fachkenntnisse verfügen, bilden die Ausnahme.

An der Universität Leipzig haben wir Unterlagen entwickelt und Voraussetzungen für praktische Übungen geschaffen, die geeignet sind, Kenntnisse über die Integration des Internets mit existierenden EIS-Systemen zu vermitteln. Diese Möglichkeiten werden derzeitig von einer Reihe deutscher Hochschulen für die Ausbildung von studentischen Nachwuchskräften genutzt.

Wir glauben, dass ein ausbalanciertes und integriertes Fachwissen auf den Gebieten

- Java, Internet, XML und Webservices
- CICS, IMS, z/OS Workload Manager und z/OS Virtualisierungstechnology

eine Kernkompetenz für IT-Spezialisten in den kommenden Jahren darstellen wird.

Abkürzungen

BMS	Basic Mapping Support
CICS	Customer Information System
CTG	CICS Transaction Gateway
DPL	CICS Distributed Program Link
ECI	External Call Interface (CICS)
EPI	External Programming Interface (CICS)
EXCI	
EIS	Enterprise Information System
EJB	Enterprise Java Bean
IMS	Information Management System
J2EE	Java 2 Enterprise Edition
JCA	Java Connector Architecture
JNDI	Java Naming and Directory Interface
JSP	Java Server Page
JTS	Java Transaction Service
JVM	Java Virtual Machine
LDAP	Light Weight Directory Access Protocol
MBQ	Message based Queuing
MOM	Message oriented Middleware
ORB	Objrct Request Broker
SOA	Service Oriented Architecture
SOAP	
WSDL	Web Service Definition Language
WSED	WebSphere Studio Enterprise Developer

Literatur

[BEY05] Marc Beyerle, Joachim Franz, Wilhelm G. Spruth: *Persistent Reuseable Java Virtual Machine unter z/OS und Linux*. Inform., Forsch. Entwickl. 20(1-2): 102-111 (2005).

[CGP02] CICS Transaction Gateway: Programming Guide V5.0. IBM Form No. SC34-6141-00, Juli 2002

Grzegorz Czajkowski, Laurent Daynès: *Multitasking without Compromise: a Virtual Machine Evolution.* ACM OOPSLA'01, Tampa, FL. http://research.sun.com/projects/barcelona/papers/oopsla01.pdf.

[FRA03] Joachim Franz, Wilhelm G. Spruth: *Reengineering von Kernanwendungssystemen auf Großrechnern*. Informatik Spektrum, Band 26, Nr. 2, April 2003, S. 83-93.

[GRA99] J. Gray: How High is High Performance Transaction Processing? http://research.Microsoft.com/~Gray/Talks/

[GSl05]: Gerrit Schlüter, Untersuchungen zum CICS Transaction Gateway in der J2EE-Umgebung, Dipl.-Arbeit, Institut für Informatik, Universität Leipzig, 2005

[HKS03]: Herrmann, P., Kebschull, U., Spruth, W.G., Einführung in z/OS und OS/390, Verlag Oldenbourg, 2. Auflage, 2003

[HOR00] J. Horswill: Designing & Programming CICS Applications. O'Reilly, 2000. ISBN 1-56592-676-5

[ICA04] IBM: WebSphere for z/OS Connectivity Architectural Choices, Dezember 2004. http://www.redbooks.ibm.com/redbooks/pdfs/sg246365.pdf

[IRB04]: z/OS Version 1 Release 5 Implementation, IBM Redbook SG24-6326-00, ISBN: 0738491411, 2004

[IWC04] IBM: Integrating WebSphere Application Server and CICS using the J2EE Connector Architecture, Januar 2004. http://publibfp.boulder.ibm.com/epubs/pdf/22472180.pdf

[JCP04] Java Community Process : *JSR 121 Application Isolation API Specification*. http://www.jcp.org/en/jsr/detail?id=121.

[KAP01] Kappelmann: *Some Strategic Y2K Blessings*. IEEE Software, Vol. 17, No.2, March/April 2000, p.42.

[ROE99] E. Roman, R. Öberg: The Technical Benefits of EJB and J2EE Technologies over COM+ and Windows DNA. Artikel der Middleware Company, Dezember 1999

[SAN04] Bo Sandén: *Coping with Java Threads*. IEEE Computer, Vol. 37, Nr. 4, April 2004, p. 20.

[SPE99] http://www-3.ibm.com/developer/solutionsevent/pdfs/spector_lunchtime_keynote.pdf