

Spring 4-1-2011

Integration of MS Excel with SAP Database

Shikha Jain
Dakota State University

Follow this and additional works at: <https://scholar.dsu.edu/theses>

Recommended Citation

Jain, Shikha, "Integration of MS Excel with SAP Database" (2011). *Masters Theses*. 195.
<https://scholar.dsu.edu/theses/195>

This Thesis is brought to you for free and open access by Beadle Scholar. It has been accepted for inclusion in Masters Theses by an authorized administrator of Beadle Scholar. For more information, please contact repository@dsu.edu.

Integration of MS Excel with SAP Database

A graduate project submitted to Dakota State University in partial fulfillment of the requirements for the degree of

Master of Science
in
Information Systems
04,2011.

By
Shikha Jain

Project Committee:

Dr Shan Ronghua
Dr. Stephen Krebsbach
Dr. Mark Moran



PROJECT APPROVAL FORM

We certify that we have read this project and that, in our opinion, it is satisfactory in scope and quality as a project for the degree of Master of Science in Information Systems.

Student Name: Shikha Jain.

Master's Project Title: **Integration of MS Excel with SAP Database**

Faculty supervisor: Ranghwa Shem Date: 4/27/11

Committee member: Stephen Kules Date: 4/27/11

Committee member: Mark Moran Date: 4/27/11

ABSTRACT

The aim of this project is to empower individuals with the ability to simply launch an excel with updated and consolidated data (“Resource Time Management and Approval Process” in our case) , view and update it and upload the excel back with updated data at individual’s convenience and hence updating the SAP integrated Database Directly, therefore connecting different platforms and accessing Data from a heterogeneous combination of Systems. (The MS Excel with all its popular features can also be stored and processed at user’s convenience.)

Also the user can update the SAP database by simply loading back the updated excel sheet on the SAP database. The user just needs to load back the rows which are to be changed or just need to add the rows to be added in the SAP database (not the whole data).

Integration of MS Excel with sap is advantageous because sap has many complex business oriented logic enabled data mining capabilities, more and more business’s are going towards it. Sap reduces the redundant errors in the system as it provides real time information. It is easy to implement globally and updates need only to be done once for implementing it company wide.

DECLARATION

I hereby certify that this project constitutes my own product, that where the language of others is set forth, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions or writings of another.

I declare that the project describes original work that has not previously been presented for the award of any other degree of any institution.

Signed,

Shikha Jain.

University ID – 1875943.

TABLE OF CONTENTS

PROJECT APPROVAL FORM	ERROR! BOOKMARK NOT DEFINED.
ABSTRACT	III
DECLARATION	IV
TABLE OF CONTENTS	V
LIST OF FIGURES	VII
CHAPTER 1.....	1
INTRODUCTION	1
BACKGROUND OF THE PROBLEM	1
STATEMENT OF THE PROBLEM	1
OBJECTIVES OF THE PROJECT	2
CHAPTER 2.....	3
LITERATURE REVIEW	3
2.1 SAP	3
2.2 REMOTE FUNCTION CALL (RFC).....	4
2.3 ABAP WEB DYNPRO.....	5
2.4 WEB SERVICES.....	6
2.5 MS EXCEL SHEET	7
CHAPTER 3.....	8
SYSTEM DESIGN AND CREATION.....	8
SYSTEM DESIGN FOR CREATING DYNAMIC EXCEL SHEET	8
FUNCTION GROUP.....	11
DATABASE TABLE.....	14
TABLE TYPE.....	24
REMOTE FUNCTION MODULE	25
ABAP WEB DYNPRO OBJECT.....	35
DOWNLOAD TO EXCEL SHEET	39
UPLOAD EXCEL SHEET INTO SAP DATABASE.....	44
WEB SERVICE.....	53
CHAPTER 4.....	57

RESULTS	57
DOWNLOAD EXCEL.....	57
UPLOAD EXCEL	59
CHAPTER 5.....	63
CONCLUSIONS	63
REFERENCES	64
APPENDICES.....	65
APPENDIX A: TECHNICAL ARCHITECTURE DIAGRAM	66
APPENDIX B: SAP CODE.....	67
<i>Function Y_RFC_WS_CORP_PROJ_MEM_HOURS</i>	<i>67</i>
DOWNLOAD EXCEL METHODS.....	68
<i>ONACTIONEXPORT_TO_EXCEL.....</i>	<i>68</i>
<i>ZSUBMIT</i>	<i>70</i>
UPLOAD EXCEL SHEET METHODS	71
<i>ONACTIONUPLOAD_EXCEL.....</i>	<i>71</i>
<i>SAVE_DB</i>	<i>74</i>

LIST OF FIGURES

Figure 1. Web Dynpro Programming Model	6
Figure 2 . Web Services Architecture	7
Figure 3. Project Flow.....	9
Figure 4. Technical Architecture.....	10
Figure 5 .Function Group-Initial Screen	12
Figure 6.Function Group-Create Object	12
Figure 7.Create Function Group	13
Figure 8.Function Group-Create Object Entry.....	13
Figure 9.Function Group Created	14
Figure 10. ABAP Dictionary-Initial Screen.....	15
Figure 11.Database Table Creation.....	15
Figure 12. Maintain Table.....	16
Figure 13. Maintain Table-Delivery Tab	16
Figure 14. Defining Database Table	17
Figure 15. Database Table-Technical Settings.....	19
Figure 16. Selecting Table Maintenance Generator.....	20
Figure 17. Table Maintenance Dialog.....	21
Figure 18. Saving Settings in Table Maintenance	22
Figure 19. Maintain Table Views:Initial Screen	23
Figure 20. Data Browser-Table YTABPROJ.....	23
Figure 21.Selction to create Table Type	24
Figure 22.Creating Table Type	25
Figure 23.Selection to create Function module	26
Figure 24.Create Function Module Screen	27
Figure 25. Create Function Module-Filling up information	27
Figure 26.Function Module properties.....	28
Figure 27.Selecting Function Module as Remote.....	28
Figure 28.Default source code of a function module.....	29

Figure 29. Specifying Export Parameters of function module.....	30
Figure 30. Source code of the function module.....	32
Figure 31. Source code of function module Y_FM_PROJ_MEMBERS	33
Figure 32. Executing function Y_CORP_PROJ_MEMBERS_HOURS.....	33
Figure 33. Result screen-Y_CORP_PROJ_MEMBERS_HOURS.....	34
Figure 34. Display data from PROJTAB from results screen.....	34
Figure 35. Creating ABAP Web Dynpro Object	35
Figure 36. Create Service call from Web Dynpro Object.....	35
Figure 37. Web Dynpro wizard.....	36
Figure 38. Web Dynpro Wizard: Select Service.....	37
Figure 39. Mapping controller context with view context.....	38
Figure 40. Designing end user view.....	38
Figure 41. Context mapping for Download Excel	39
Figure 42. Method Export_To_Excel	40
Figure 43. Method Z_SUBMIT	43
Figure 44. UPLOAD EXCEL UI Element.....	45
Figure 45. Buffer string to hold uploaded excel file.....	46
Figure 46. File upload element to browse the file on the users system	47
Figure 47. UI Element UPLOAD EXCEL linked to method UPLOAD_EXCEL	47
Figure 48. Method UPLOAD_EXCEL.....	48
Figure 49. Upload Excel context in web dynpro object.....	51
Figure 50. Method SAVE_DB.....	52
Figure 51. Creating Web dynpro application.....	53
Figure 52. Activating web dynpro application.....	54
Figure 53. Executing web dynpro application	55
Figure 54. User View	56
Figure 55. User entered PROJECT ID.....	57
Figure 56. User hits DOWNLOAD EXCEL BUTTON	58
Figure 57. Generated Excel Sheet.....	59
Figure 58. Browsing SAP database table.....	60
Figure 59. Table YTABPROJ entries	60

Figure 60. Changing APPROVED column in the excel sheet..... 61

Figure 61. Browsing the excel file to be uploaded..... 61

Figure 62. Users clicks UPLOAD EXCEL button..... 62

Figure 63. Updated database table YTABPROJ 62

CHAPTER 1

INTRODUCTION

Background of the Problem

Before this project was implemented, the HR department of Infobahn had to manually verify and process the billing of the number of project hours billed per person in a particular department. They need to manually take out the entries if they wanted to calculate the numbers of man hours per project or under a particular manager and store the data in the excel sheet for reference to the higher authorities. Thus majority of the time and resources were deployed in the billing hours calculation process as it is time consuming and the employees range thousands in number. Also it was difficult for the HR group to make changes to the data entries or add new data for the said purpose. They had to send email for the same to the data group who take care of the billing data and the data storage via manual data entries for each and every week for any resource allocated to any number of projects.

Statement of the problem

The work load from the HR Department can be significantly reduced by automating the whole process of employee hours and billing management.

HR can use a web link which will show them an optional filter to get employee hours records in a consolidated and convenient MS Excel sheet, using the user friendly and popular features of MS Excel they can sort, calculate, formulate, manipulate record information and approve or Reject the time sheets of the employees in the Excel itself. All they have to do is to simply save and load the excel sheet via the same web link. All the data entries for multiple records, employees and projects will be entered in the

respective data records in the SAP Database (DB2), thus eliminating the huge resource allocation for manual data entries, approval emails and unconsolidated view for managers.

Using Excel sheet it's easy for the HR person to create the charts depending upon the data extracted from the SAP database. Charts are easy form of taking a broad picture in mind of for e.g. total number of resource hours under a particular manager or the total billing under projects.

Also it is easy for the HR people to make changes to the billing data or to add new data by simply adding it in the excel sheet and loading that excel sheet in the SAP database. It also saves a lot of time as they don't need to exchange emails and they get the confirmation right away if the data they have provided does not match to the type of columns in the database.

Moreover the end user doesn't need to have SAP installed in their system. They can perform data manipulation and see the data by just opening up a web link.

Objectives of the project

The Project Objective is accomplished in three major activities. (With activity one, being the major part of this project.)

1. The first step is to create the function modules in SAP ERP which the company is using) that can access the data from the SAP database by specifying the type of information like project id, etc. The SAP database I will be using is DB2. The tables have been created in the database which contains the information related to the project hours billed per person.
2. The second step is to create a ABAP Web Dynpro Object. This object specifies the end user layout (which will be rendered by the web service to the end user) and specifies how to connect the excel sheet to the SAP database from the end user point of view (both uploading MS Excel sheet and downloading excel sheet)
3. The last step is to create the web services from the ABAP Web Dynpro object created in step 2. Web services are standalone and executable entities which can be published, searched and called across a network. Creating web service gives a web page url to the end user which they can use to access the application independent of the system.

CHAPTER 2

LITERATURE REVIEW

2.1 SAP

SAP is an Enterprise Resource Planning package which helps to automate a companies business and enhance its information system. SAP stands for “ Systems, Applications And Products in Data Processing ”.

SAP ERP solutions offer products that cover the key areas of the business organizations. Some of them are: –

Customer Relationship Management.

Material Management.

Financials.

Human Resources.

SAP uses the concept of function modules which can be purchased, installed and run separately but all of them extract data from a common database. Each function module handles specific business task of its own but they can be linked when required. This helps in global integration of the data; updates can be done only once companywide and also provides real-time information. The modules include the utilities for marketing, sales, customer relationship management, inventory, human resources etc.

Deployment and maintenance costs of SAP systems vary depending on the size of the organization and whether the business needs full support of all the SAP modules or just few depending on the organization requirements. No doubt when SAP gets fully implemented the businesses efficiency greatly increases in terms of the enterprise resource planning. But

deploying SAP and customizing the SAP function modules according to business requirements takes lot of time and resources.

SAP client systems can communicate with each other using SAP provided security protocols. However to communicate with other systems to SAP vendor must use third party protocols.

Some of the advantages of implementing SAP are:-

- Updates on the system database needs to be done only once
- Allows easier integration of the global data like currencies, languages etc
- Reduces the redundant errors in the system as it provides real time information

SAP also has some disadvantages as mentioned below:-

- Implementing SAP has a risk of failure associated with it.
- Return on investment sometimes takes more time.
- The ERP packages of the vendor and the customer might be different and might be difficult to customize to suit each other. Also it might take lot of time and resources.

2.2 Remote function Call (RFC)

This project typically requires the use of Remote Function Call to integrate SAP with the external systems which in our case is MS Excel Sheet.

Remote function call is the SAP interface for communication between the client and server over TCP/IP protocol. Remote function calls can be inbuilt software that comes with SAP package or can be written using ABAP language using SAP ABAP workbench. RFC's provide in which external programs written in other languages or external software can use the data returned from the SAP server as well as can insert the data back into the SAP server.

Following interfaces made up the RFC interface:-

- An interface designed for calling ABAP programs
- To call a remote function module each ABAP program uses the command CALL FUNCTION FUNCTION_NAME....DESTINATION. The parameters specified in

the DESTINATION specify that the called function module is to be run in a different system than to the calling system. CALL FUNCTION command helps SAP in the RFC communication with the external system.

- RFC function modules are to be registered in the SAP system as a remote function module or else they will not be called by the external system unless specified as RFC.
- If both calling and called programs are in the SAP system one of the function modules must be registered as remote.
- If one of the programs is the non-ABAP program, it needs to be programmed in such a way that it can communicate with the other ABAP RFC program.
- To call function modules in SAP system, external programs can use RFC supported interfaces and execute them in SAP systems. Similarly ABAP programs can also use these interfaces to execute external programs

To make a function module Remote :-

- Go to transaction SE37 (Function Builder).
- Go to the Attributes tab in the ABAP Workbench
- Set the “Remote-Enabled module” flag.

2.3 ABAP Web Dynpro

ABAP Web Dynpro is the SAP standard to develop web based applications. It consists of a GUI environment and runtime environment. The transaction to create ABAP Web Dynpro Object is SE80.

The main advantages of ABAP Web Dynpro are:-

1. It is a GUI based environment which significantly reduces the implementation effort and time.
2. There is a strict separation between layout and data.
3. Components are reusable.
4. It is useful to develop user interface applications.

The Web Dynpro programming model is based on the Model View Controller model.

Model forms the interface with the back end system to access the back-end data

View makes the data available for view in the browser

Controller makes the data available to the user using View, and also processes the data on user input and presents it back to the user through the View

Following figure shows the Web Dynpro programming model

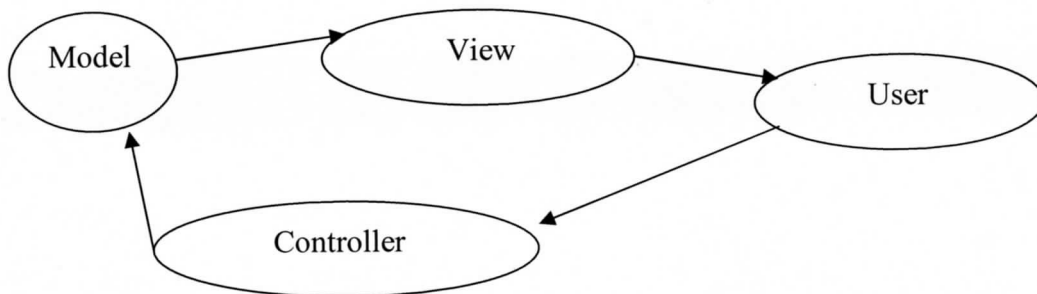


Figure 1. Web Dynpro Programming Model

2.4 Web services

An easy way to integrate functions implemented on widely differing software components is by using Web Services. Web services are based on open and accepted standards. They can combine functions into a single process even if the functions are implemented on widely different software components. Web services are standalone and executable entities which can be published, searched and called across a network.

Web services architecture involves interactions between three primary entities- service requestor, service registry service provider. These entities interact with each other through bind, publish and find requests (Ref: 1). The service provider provides access to the web service. It publishes the description of the service in the service registry. The service requestor looks up the service description in the service registry and binds itself to the

The logical view of the web services architecture is shown below

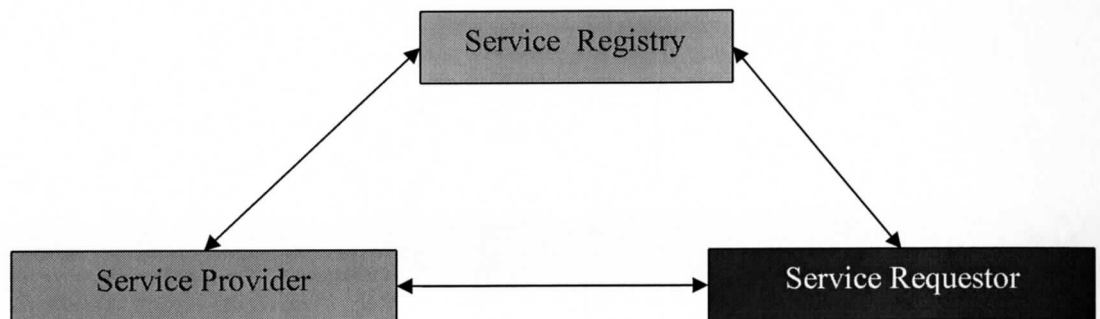


Figure 2 . Web Services Architecture

2.5 MS Excel Sheet

Microsoft Excel Sheet is a spreadsheet application which is distributed by Microsoft. It contains a grid of cells arranged in columns and rows which makes it easy to view and manipulate the data. It contains a library of inbuilt functions for arithmetical, statistical and engineering and financial needs. Moreover the user can create macros using VBA (Visual Basic for Applications) for user-defined calculations or operations.

The Excel sheet offers different options like to create charts, histograms, pivot tables etc from the data in the sheet to the user. It is useful to the user to get the broad view of the organizational data or performance etc.

Moreover Microsoft offers the flexibility to integrate MS Excel to other applications database. It helps to extract real-time data from other applications and present it to the user or update the database by loading Excel sheet into the system.

CHAPTER 3

SYSTEM DESIGN AND CREATION

System design for creating dynamic excel sheet

The systems design is to gather the latest and updated information of the number of project hours billed per person in a particular project from SAP database and show it in an Excel Sheet to the end user. The user can specify criteria such as project hours billed for a person for a particular week ending date or multiple weeks or depending upon the project how many total man hours have been billed as a whole. Additional functionality can be added as per the requirements. Also the user can update or modify the data on the excel sheet and load it back into the SAP database by just loading the updated excel sheet into SAP system.

Following technologies are used in the project:-

Application Server (SAP) - Programming Language → ABAP, ABAP Web dynpro

Web Service (SAP) - ABAP Web Service wizard

MS excel Sheet

Versions: –

MS Excel -- MS Office Suite 3.0

Sap -- ECC 6.0

SAP based its architecture on 3-tier client-server model (presentation server, application server, database server).

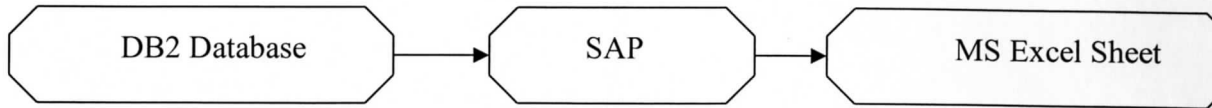


Figure 3. Project Flow

For this project we link different technologies in our multilayer architecture as shown in figure below:-

....Continued on next page

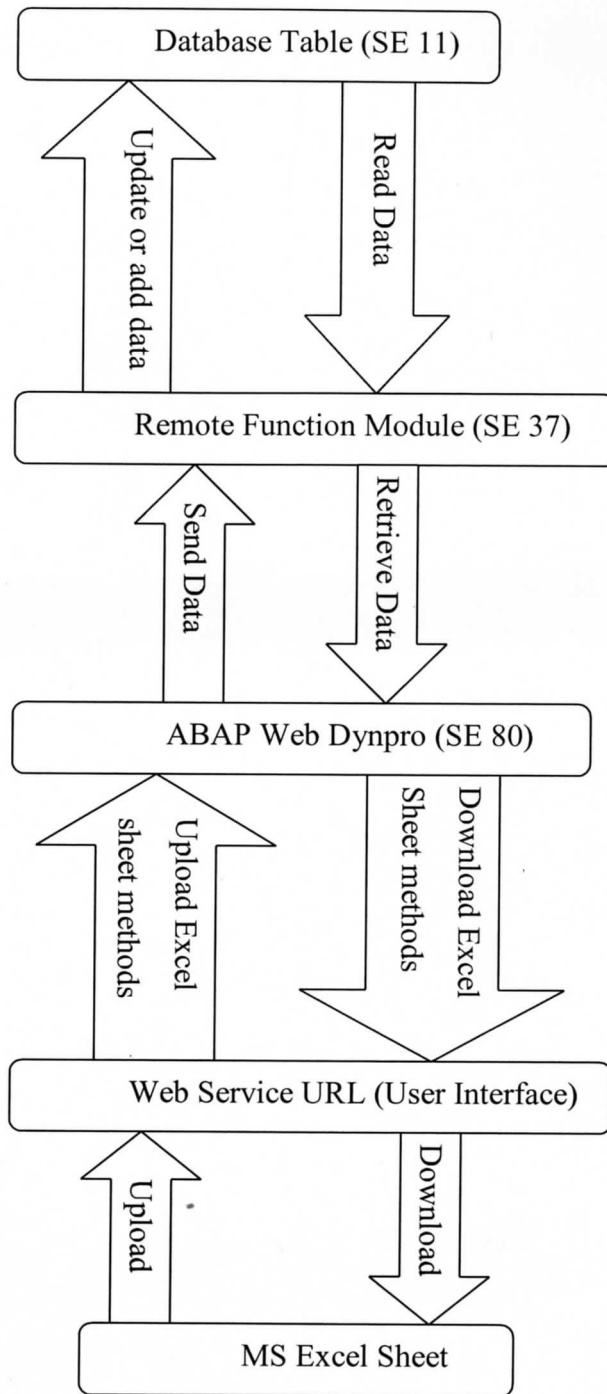


Figure 4. Technical Architecture

We use web services to integrate SAP and MS Excel. Web services are application programming interfaces (API's) that can be accessed over internet and can be executed on remote system. Since the system landscapes are heterogeneous in nature all the functions of a process cannot be implemented using the same technology and on the same component. Web services can simplify this integration. Since our priority in programming this project is the selection of data, the integration of ABAP web service is appropriate.

We create functions (in SAP System) that can access the data in SAP from DB2 database based upon requesting parameters. After which we will create a ABAP Web Dynpro Object that can specify the end user layout and also define methods which can relate the user input with the RFC's we created. Then we create a web service in SAP from Web Dynpro object using ABAP web service wizard which gives us a url which the end user can access regardless of the system.

When the user access MS Excel and tries to access the data in DB2 database, the web service will invoke the function modules in SAP system to retrieve the data in DB2 database and show it in MS Excel in the form of columns and rows.

If the user wants to update the data in the database he/she needs to upload the excel sheet lying in his/her system, which then invokes the function modules in the RFC to update the database.

The sequence of activities which needs to be performed as per the business requirements are mentioned below:-

Function Group

We need to develop function modules which can extract data from SAP Database depending upon the requirements and show it in the Excel Sheet to the end user.

To develop the function module first we need to create the Function Group. It will be a group of function modules which can extract data from the database and present it to the

Excel sheet. For the function module to extract the data from the database we also need to create the database table which contains all the relevant information according to business requirements. The database table is also part of the function group

The **function group** name which can be created using SE11 transaction of SAP is
Y_CORP_PROJ_MEMBERS_HOURS

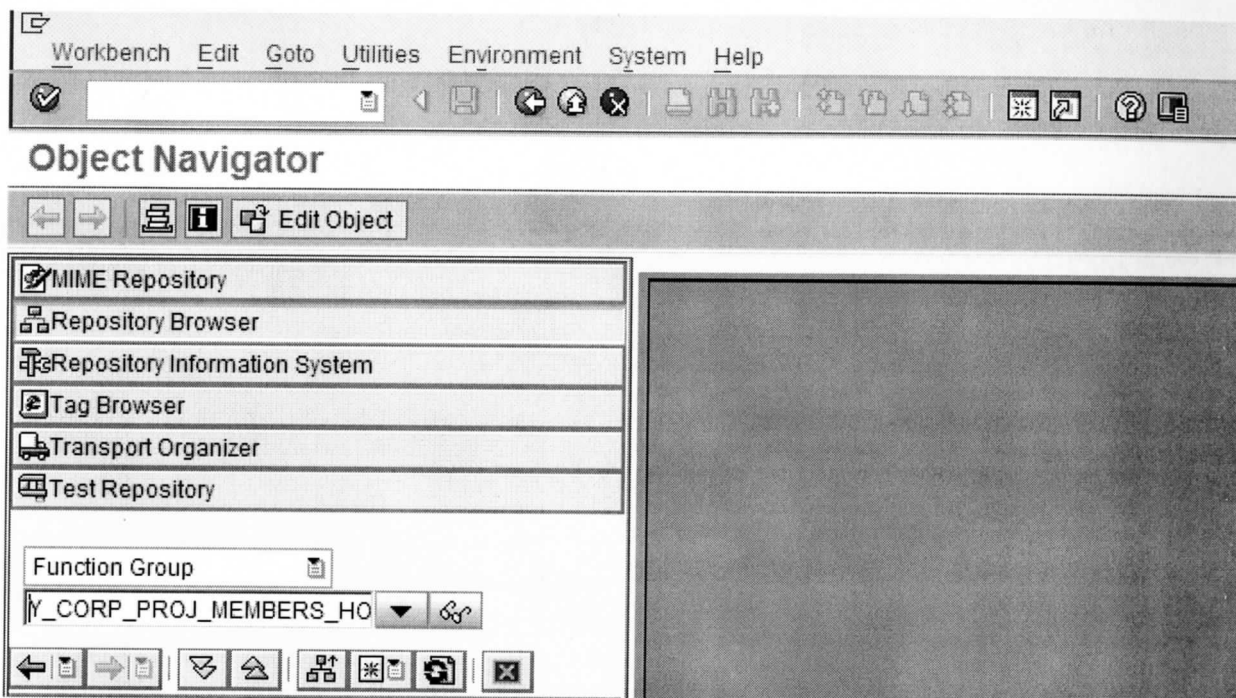


Figure 5 .Function Group-Initial Screen

Since the function group does not exist it asks to whether we want to create the object

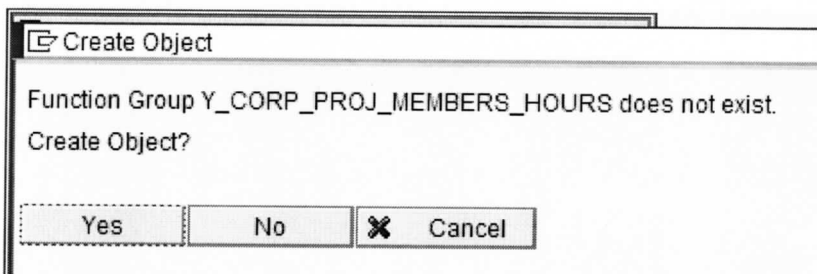
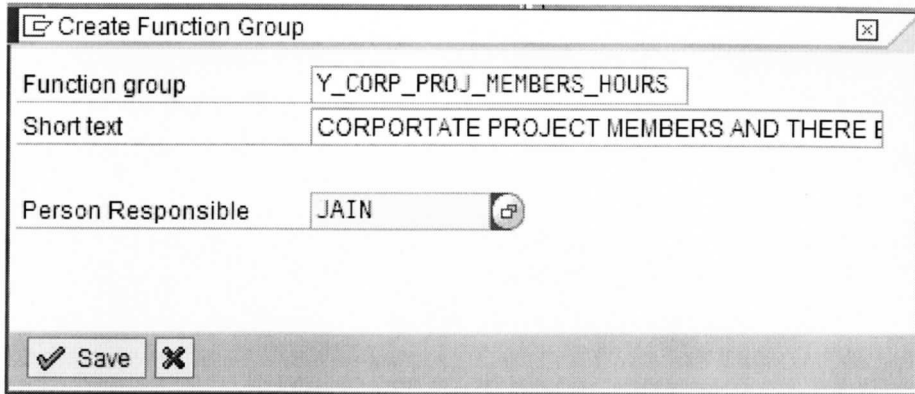


Figure 6.Function Group-Create Object

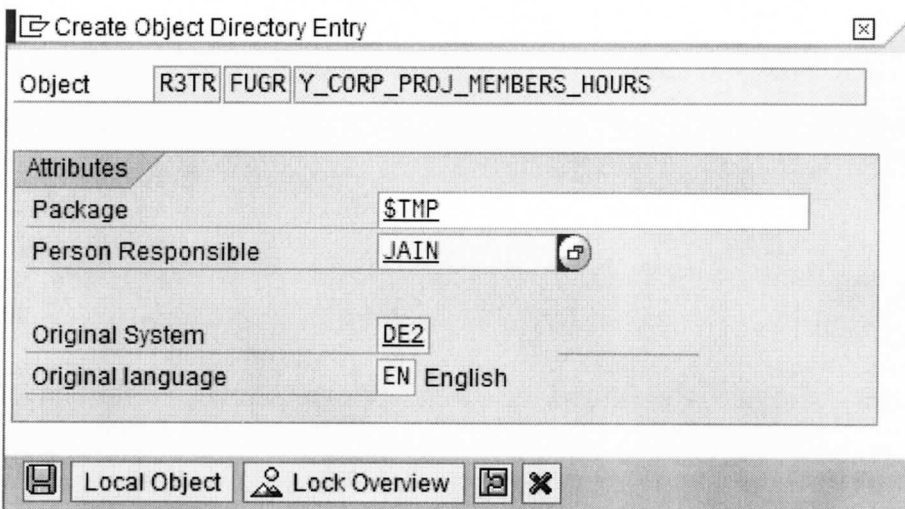


Function group	Y_CORP_PROJ_MEMBERS_HOURS
Short text	CORPORTATE PROJECT MEMBERS AND THERE B
Person Responsible	JAIN

Save X

Figure 7.Create Function Group

If we click on save window pops up



Object	R3TR FUGR Y_CORP_PROJ_MEMBERS_HOURS
Attributes	
Package	\$TMP
Person Responsible	JAIN
Original System	DE2
Original language	EN English

Local Object Lock Overview Refresh X

Figure 8.Function Group-Create Object Entry *

We can specify \$TMP as a package to which the function group belongs . \$TMP is a package local to the system. We can specify any other package depending upon the requirements. Since we are creating a local object , choose local object in the options shows in the Fig above which then takes us to the following screen

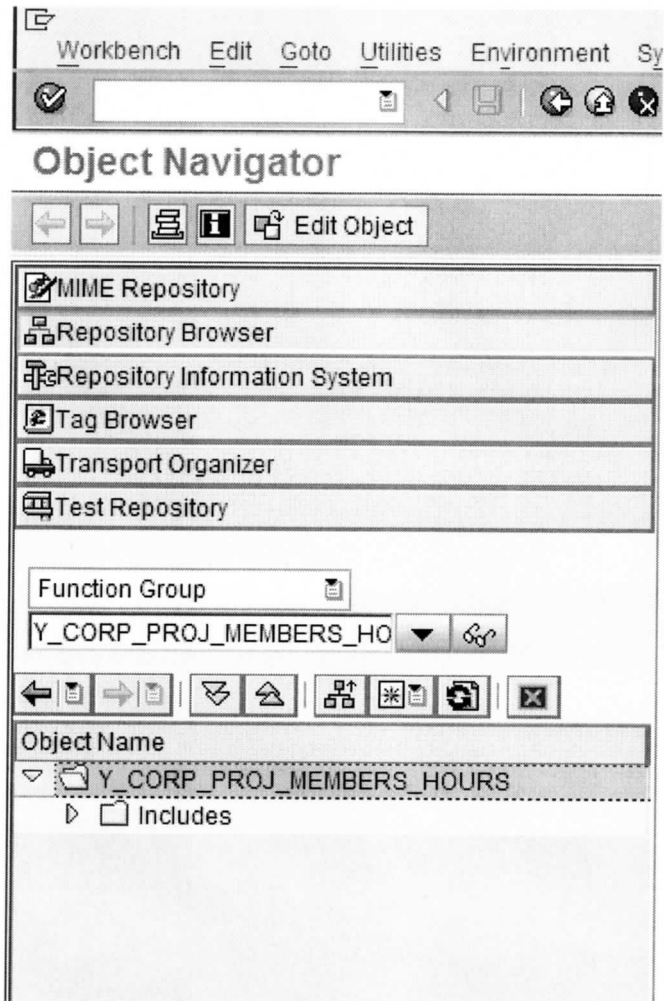


Figure 9. Function Group Created

It shows that function group Y_CORP_PROJ_MEMBERS_HOURS is created

Database table

Then we need to create the the Database table which contains the information such as PROJ_ID, EMP_ID, EMP_NAME, HOURS_BILLED, WEEK_END_DATE etc

For that we go to SE11 transaction of SAP which takes us to ABAP Dictionary initial screen

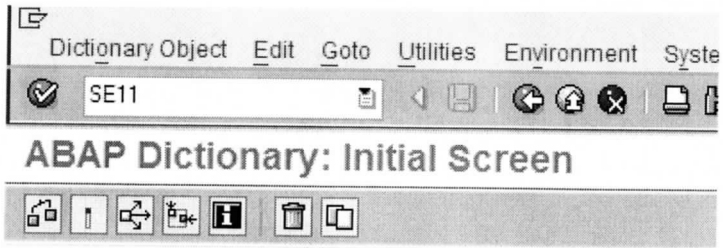


Figure 10. ABAP Dictionary-Initial Screen

ABAP dictionary is a collection of database objects where we can create objects like table, views, data type etc. For our case we create database table named YTABPROJ .

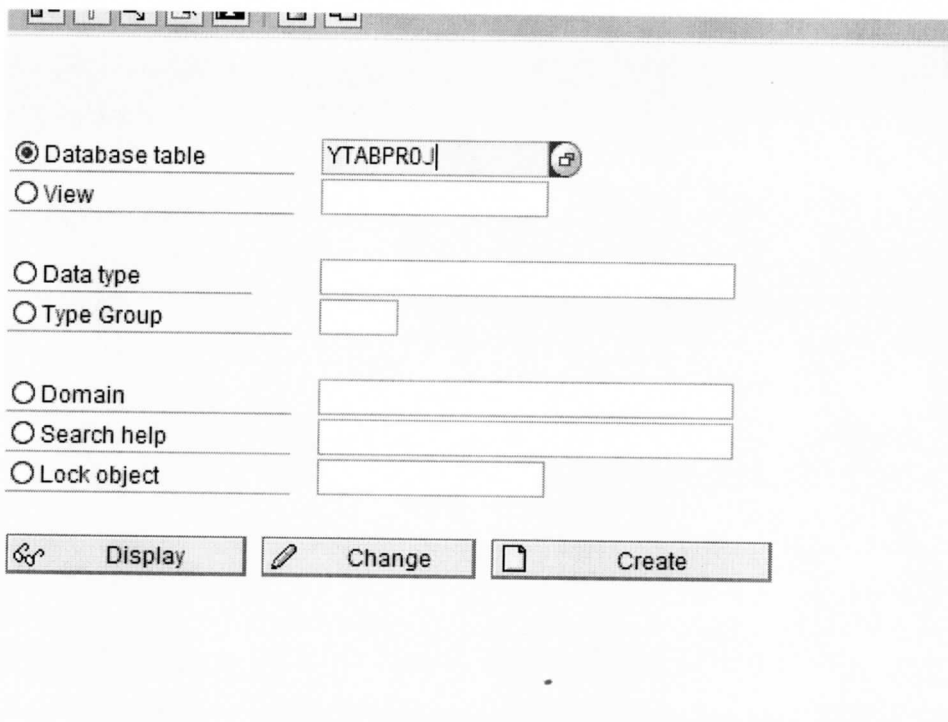


Figure 11.Database Table Creation

When we click on Create it takes us to the following screen

Table Edit Goto Utilities Extras Environment System Help

Dictionary: Maintain Table

Transp. Table YTABPROJ New(Revised)

Short Description

Attributes Delivery and Maintenance Fields Entry help/check Currency/Quantity Fields

Delivery Class A

Data Browser/Table View Maint.

- Display/Maintenance Allowed with Restrictions
- N Display/Maintenance Not Allowed
- X Display/Maintenance Allowed**
- Display/Maintenance Allowed with Restrictions

Figure 12. Maintain Table

Since we need to add columns and define the data types of those columns we choose the option Display Maintenance Allowed.

Table Edit Goto Utilities Extras Environment System Help

Dictionary: Maintain Table

Transp. Table YTABPROJ New(Revised)

Short Description Y CUSTOM TABLE FOR PROJECT ASSET NAMES & BILLING

Attributes Delivery and Maintenance Fields Entry help/check Currency/Quantity Fields

Delivery Class A

Data Browser/Table View Maint. X Display/Maintenance Allowed

Figure 13. Maintain Table-Delivery Tab

This takes to the following screen

Table Edit Goto Utilities Extras Environment System Help

Dictionary: Maintain Table

ansp. Table YTABPROJ Inactive(Revised)

Short Description Y CUSTOM TABLE FOR PROJECT ASSET NAMES & BILLING

Attributes Delivery and Maintenance Fields Entry help/check Currency/Quantity Fields

Srch Help Predefined Type 1 / 8

Field	Key	Initi	Data element	Data Ty...	Length	Decim	Short Description
CLNT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MANDT	CLNT	3		Client
PROJID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	CHAR10	CHAR	10		Character Field Length = 10
WEEK_ENDDATE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	DATUM	DATS	8		Date
EMPID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	CHAR8	CHAR	8		Character field, 8 characters long
LASTNAME	<input type="checkbox"/>	<input type="checkbox"/>	CHAR20	CHAR	20		Char 20
FIRSTNAME	<input type="checkbox"/>	<input type="checkbox"/>	CHAR20	CHAR	20		Char 20
HOURS_BILLED	<input type="checkbox"/>	<input type="checkbox"/>	NUM	NUMC	2		Sequence number
APPROVED	<input type="checkbox"/>	<input type="checkbox"/>	CHAR1	CHAR	1		Single-Character Indicator
	<input type="checkbox"/>	<input type="checkbox"/>					
	<input type="checkbox"/>	<input type="checkbox"/>					
	<input type="checkbox"/>	<input type="checkbox"/>					
	<input type="checkbox"/>	<input type="checkbox"/>					
	<input type="checkbox"/>	<input type="checkbox"/>					
	<input type="checkbox"/>	<input type="checkbox"/>					
	<input type="checkbox"/>	<input type="checkbox"/>					
	<input type="checkbox"/>	<input type="checkbox"/>					
	<input type="checkbox"/>	<input type="checkbox"/>					
	<input type="checkbox"/>	<input type="checkbox"/>					
	<input type="checkbox"/>	<input type="checkbox"/>					
	<input type="checkbox"/>	<input type="checkbox"/>					
	<input type="checkbox"/>	<input type="checkbox"/>					
	<input type="checkbox"/>	<input type="checkbox"/>					
	<input type="checkbox"/>	<input type="checkbox"/>					
	<input type="checkbox"/>	<input type="checkbox"/>					
	<input type="checkbox"/>	<input type="checkbox"/>					
	<input type="checkbox"/>	<input type="checkbox"/>					
	<input type="checkbox"/>	<input type="checkbox"/>					
	<input type="checkbox"/>	<input type="checkbox"/>					
	<input type="checkbox"/>	<input type="checkbox"/>					
	<input type="checkbox"/>	<input type="checkbox"/>					
	<input type="checkbox"/>	<input type="checkbox"/>					

Start | (10186 unread) Yahoo! ... | SAP Logon 710 | New Microsoft Office Wo... | Function Builder: Change... | Dictionary: Maintai

Figure 14. Defining Database Table

Here we have added the columns for the table and defined the data types of those columns respectively. Following columns and their associated data types have been added

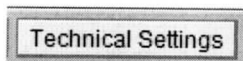
Table Name - YTABPROJ

Column name	Data Type
CLNT	MANDT
WEEK END DATE	DATUM
EMP ID	CHAR8
LAST NAME	CHAR20
FIRST NAME	CHAR20
HOURS BILLED	NUM
APPROVED	CHAR1
PROJ ID	CHAR10

The composite primary key of the table consists of columns – CLNT, WEEK END DATE, EMP ID, PROJ ID

Every table in SAP needs to be client specific, so it is mandatory to include column name CLNT for ever table we create.

Then we go to Technical Settings tab



Which takes us to the following screen

Settings Edit Goto System Help

Dictionary: Maintain Technical Settings

Revised<->Active

Name YTABPROJ Transparent Table

Short text Y CUSTOM TABLE FOR PROJECT ASSET NAMES & BILLING

Last Change AJAIN 12/09/2010

Status New Not saved

Logical storage parameters

Data class APPL0

Size category 0

Buffering

Buffering not allowed

Buffering allowed but switched off

Buffering switched on

Buffering type

Single records buff.

Generic Area Buffered No. of key fields

Fully Buffered

Log data changes

Write access only with JAVA

Figure 15. Database Table-Technical Settings

Here we can specify the technical settings of the table.

Then to let the rows need to be added to the table we need to go to Table Maintenance Generator

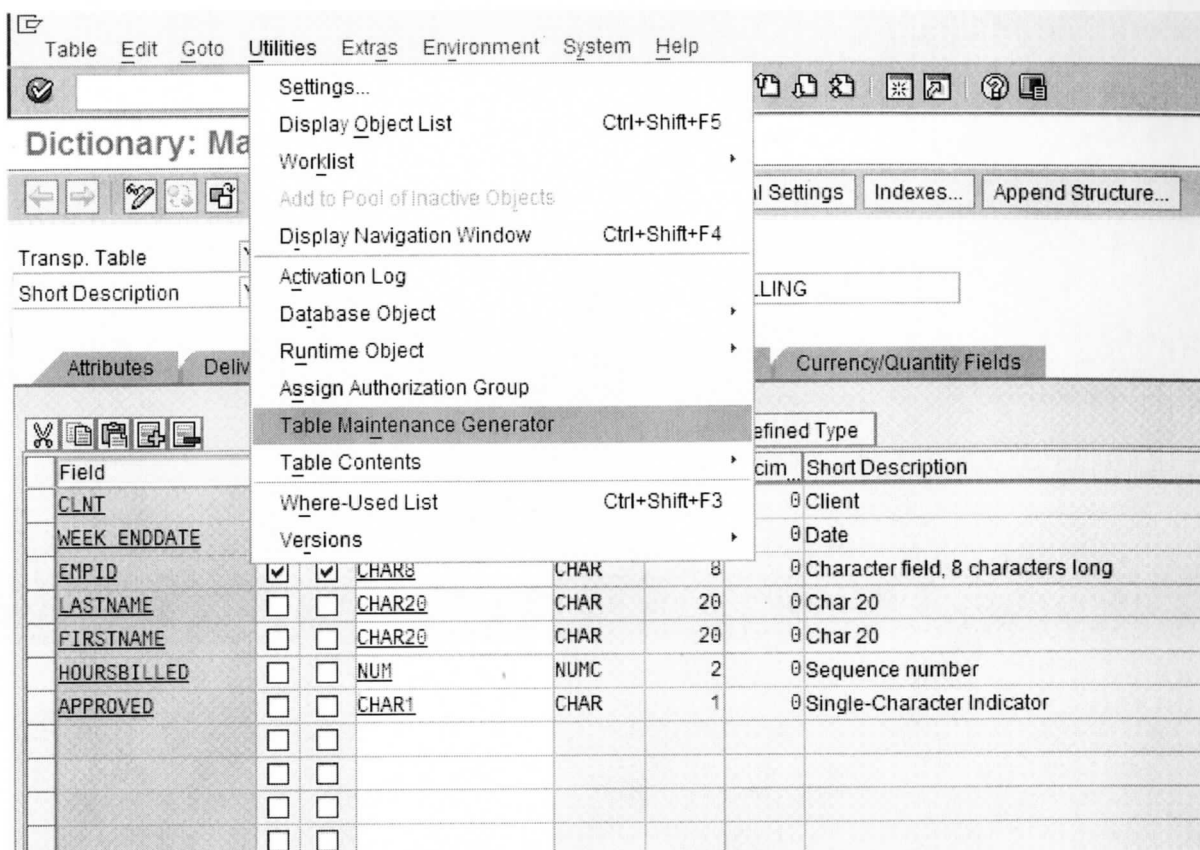


Figure 16. Selecting Table Maintenance Generator

This takes to the following screen

Generated Objects Edit Goto Environment Utilities System Help

Generate Table Maintenance Dialog: Generation Environment

Find Scr. Number(s)

Table/View YTABPROJ

Technical Dialog Details

Authorization Group &NC& w/o auth. group

Authorization object S_TABU_DIS

Function group Y_CORP_PROJ_MEMB Fn.Gr.Text

Package \$TMP Temporary Objects (never transported!)

Maintenance Screens

Maintenance type one step
 two step

Maint. Screen No. Overview screen 100
Single screen 200

Dialog Data Transport Details

Recording routine Standard recording routine
 no, or user, recording routine

Compare Flag Automatically Adjustable Note

Figure 17. Table Maintenance Dialog

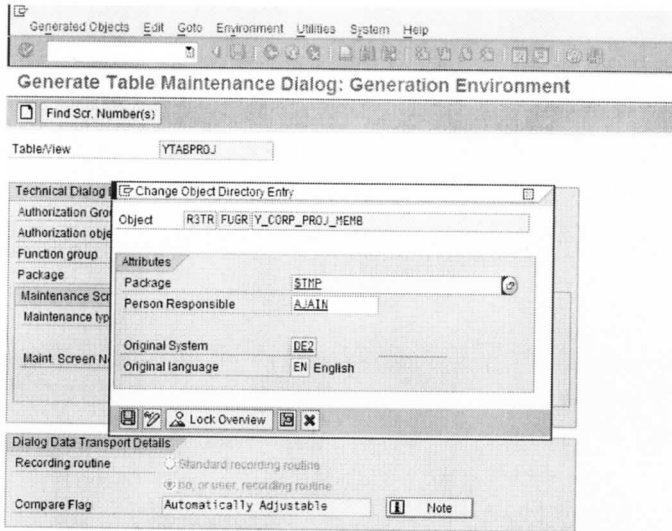


Figure 18. Saving Settings in Table Maintenance

The table is also specified in the package \$TMP and local to the system.

Then to add data to the table we need to go to the SM30 transaction which takes us to the Maintain Tables Screen

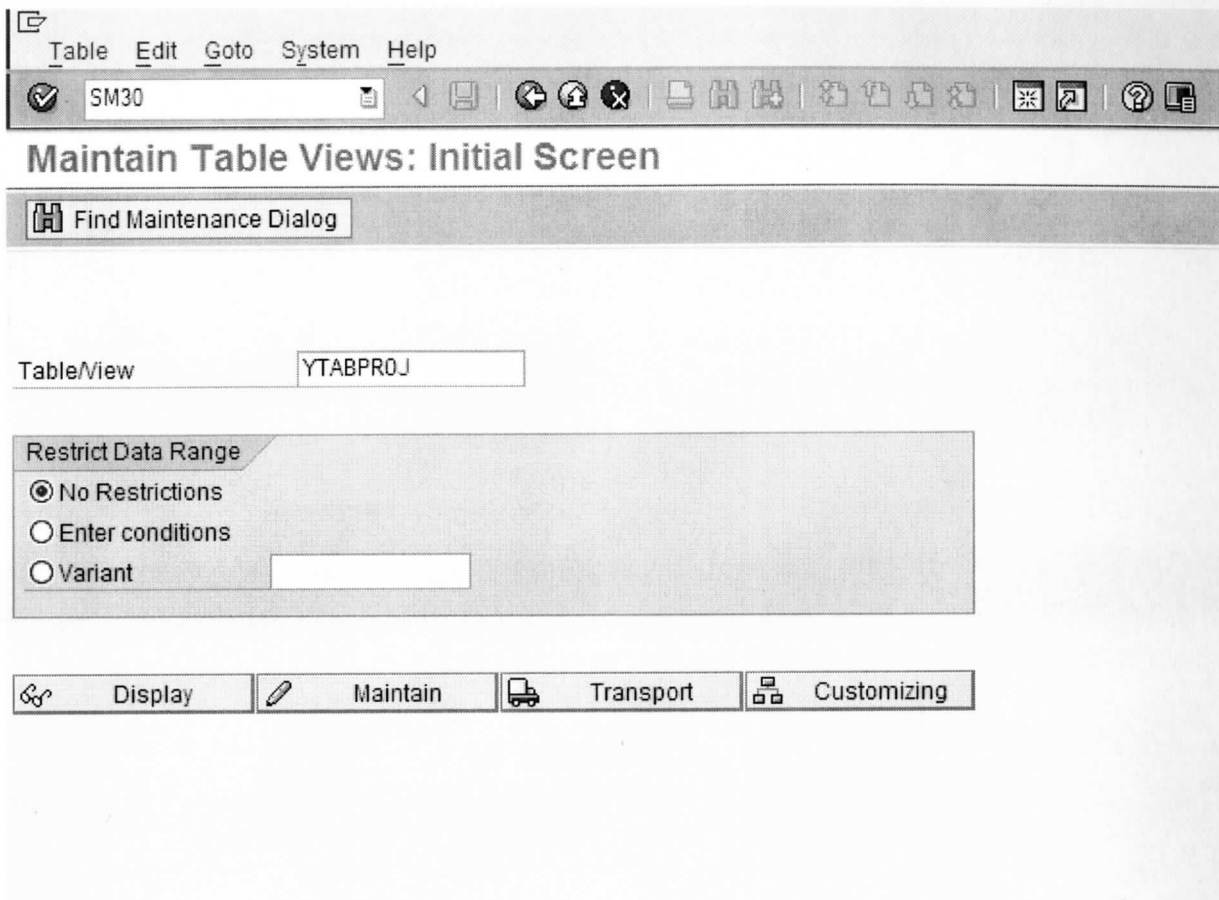


Figure 19. Maintain Table Views:Initial Screen

To add rows to the table we need to select Maintain option .
Then we can add the rows in the table as shown below

Data Browser: Table YTABPROJ Select Entries 2

Menu | Back | Exit | Cancel | System | Display | Choose | Sort Ascending | Sort Descending

Table: YTABPROJ
Displayed Fields: 8 of 8 Fixed Columns: 4 List Width 0250

	CLNT	PROJID	WEEK_ENDDATE	EMPID	LASTNAME	FIRSTNAME	HOURSBILLED	APPROVED
<input type="checkbox"/>	600		11/06/2010	01875943	JAIN	SHIKHA	40	Y
<input type="checkbox"/>	600		11/13/2010	10012760	ROB	MARIA	36	Y

Figure 20. Data Browser-Table YTABPROJ

Table type

Since we will be using the tables YTABPROJ in our function module as a data type we need to define a data type of the table YTABPROJ before we can add it in our function module. To do this we go to SE11 transaction, choose the function group Y_PROJ_MEMBERS_HOURS

It will display all types of objects which can be created under a particular function group. Since we need to create the table type , we choose create table type option

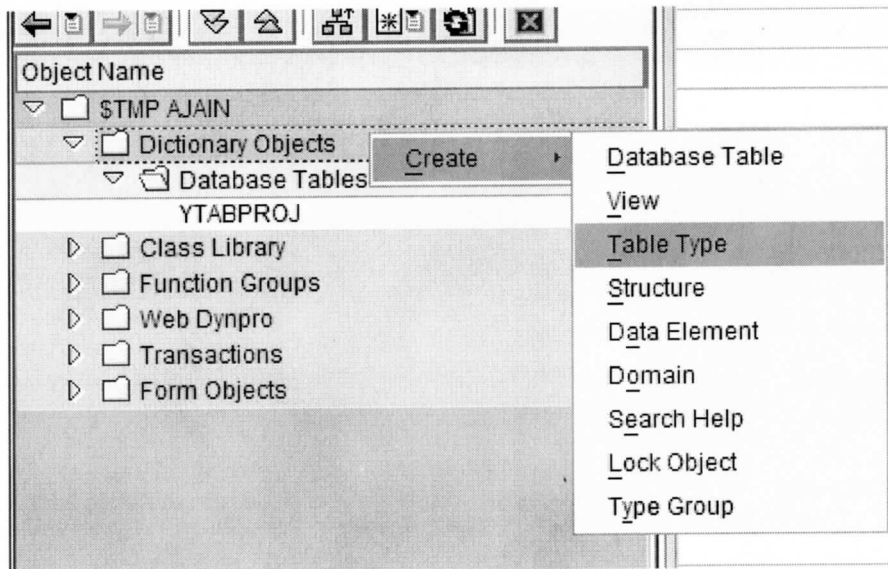


Figure 21. Selection to create Table Type

The screen asks about the options we need to give to the table type we want to create

Hierarchy Display	
Table Type	YTABTYP_PROJ New(Revised)
Short text	TABLE TYPE FOR YTABPROJ
<div style="display: flex; justify-content: space-around; border-bottom: 1px solid black;"> Attributes Line Type Initialization and Access Key </div>	
<input checked="" type="radio"/> Line Type	YTABPROJ
<input type="radio"/> Predefined Type	Data Type <input type="text"/> No. of Characters <input type="text" value="0"/> Decimal Places <input type="text" value="0"/>
<input type="radio"/> Reference type	<input type="radio"/> Name of Ref. Type <input type="text"/>
<input type="radio"/> Reference to Predefined Type	Data Type <input type="text"/> Length <input type="text" value="0"/> Decimal Places <input type="text" value="0"/>

Figure 22. Creating Table Type

Remote Function Module

After the table type is created we can create the **Remote Function Module** which takes the data from the SAP database and displays it in Excel. We call it remote because it is connected to system other than SAP (to which it belongs) through web services. It will display the data in Excel using Excel web services.

To create the function module go to the SE11 transaction of SAP and create the function named Y_RFC_W_CORP_PROJ_MEM_HOURS as shown below

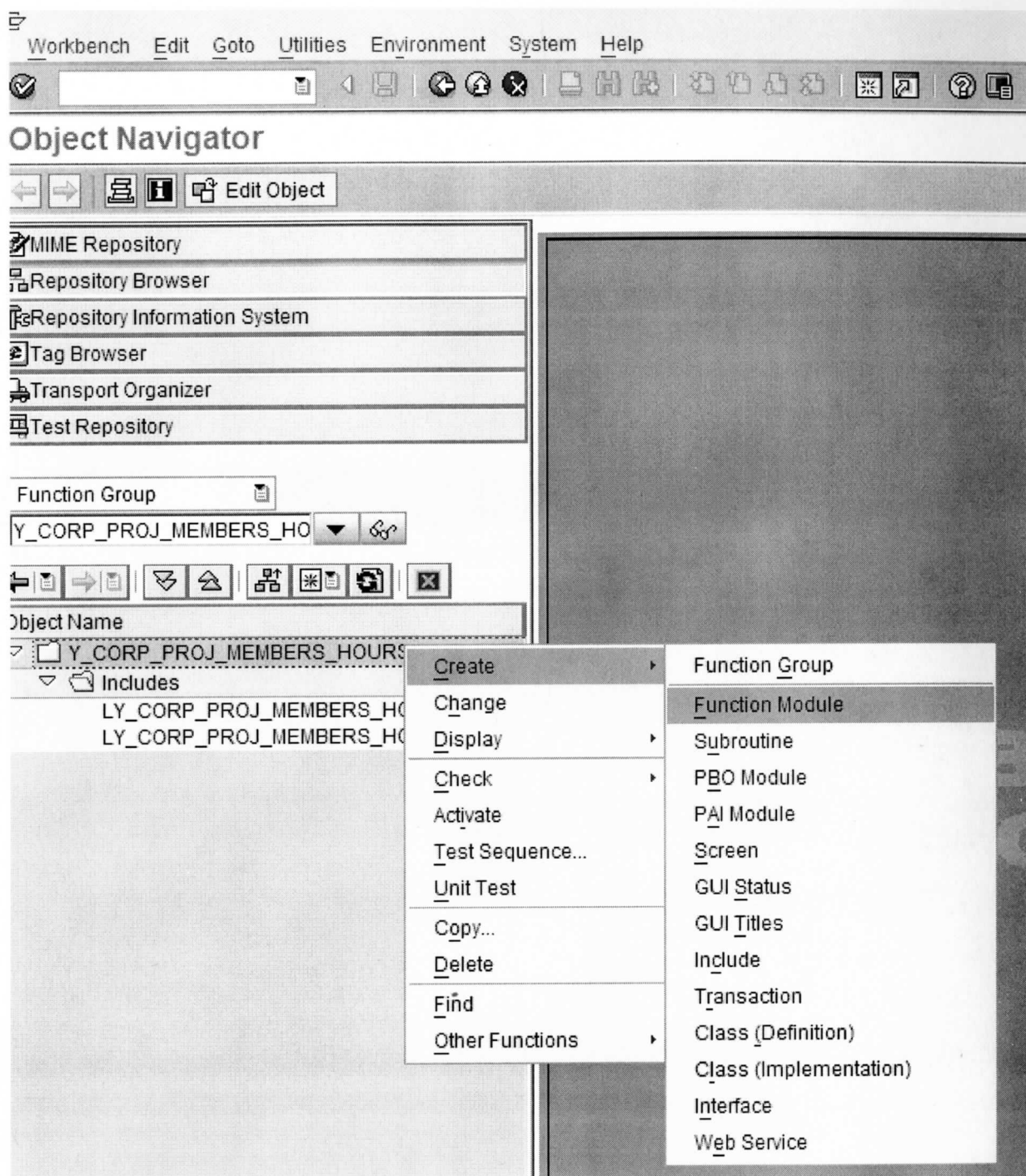


Figure 23. Selection to create Function module

Here we need to specify the function name and the function group name

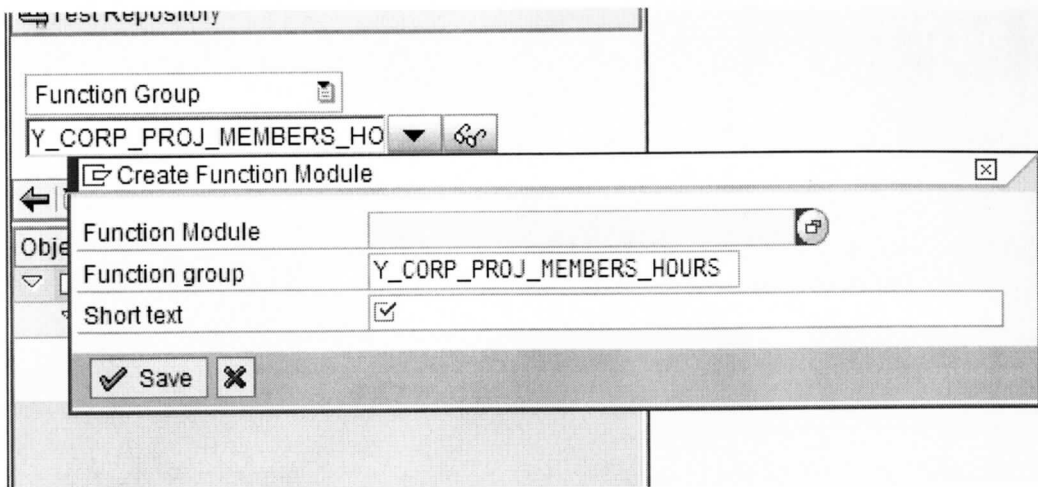


Figure 24. Create Function Module Screen

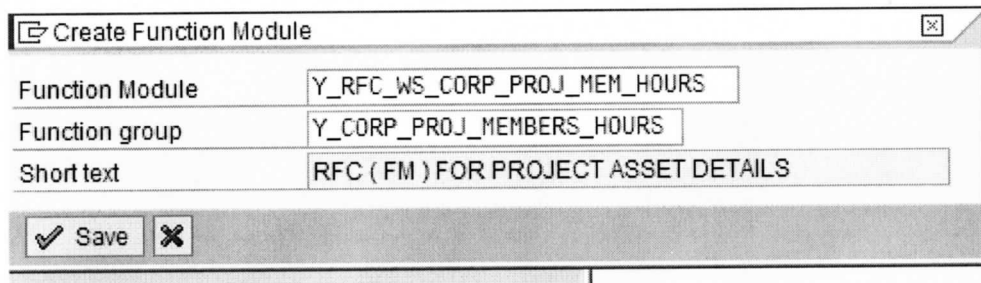


Figure 25. Create Function Module-Filling up information

When we Save the function module name it gives us the following screen

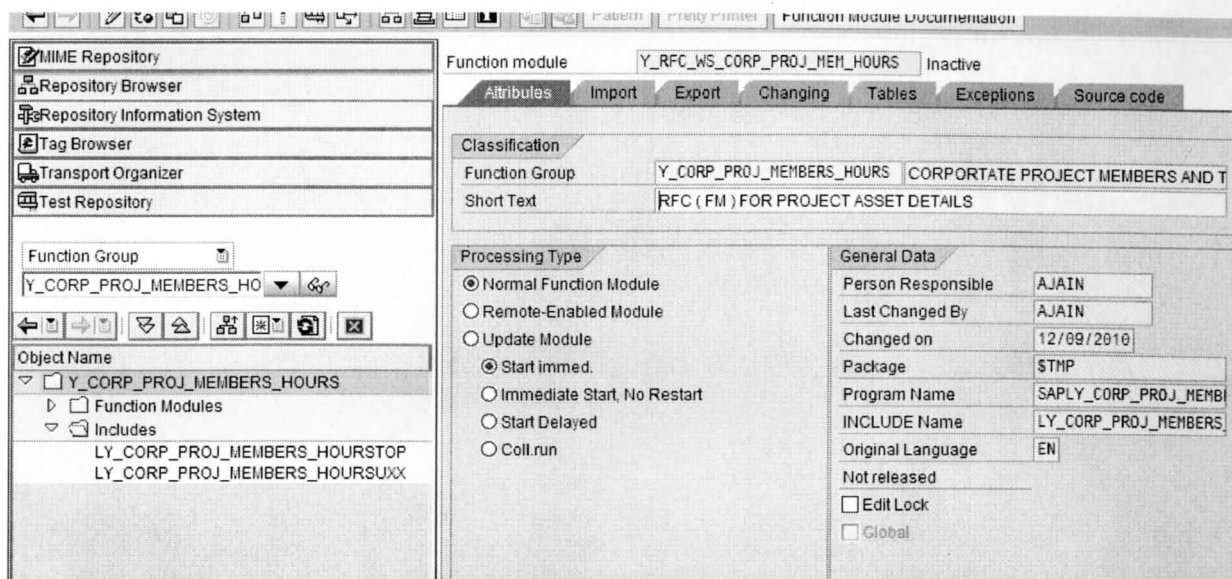


Figure 26. Function Module properties

Since the function module we need should be remote function module we check the box 'Remote –Enabled Module'

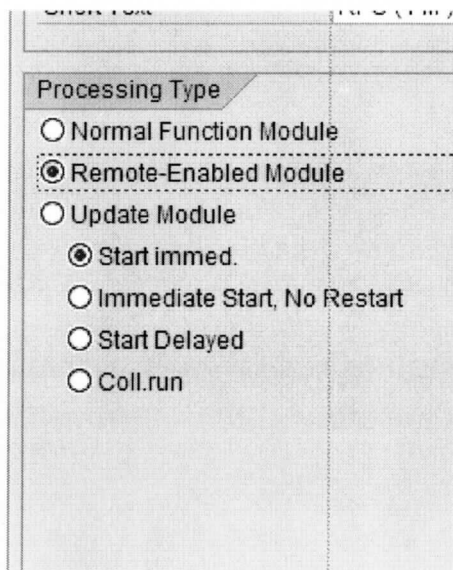


Figure 27. Selecting Function Module as Remote

The default source code of any function module created within SAP looks like below if we check the tab 'Source Code'

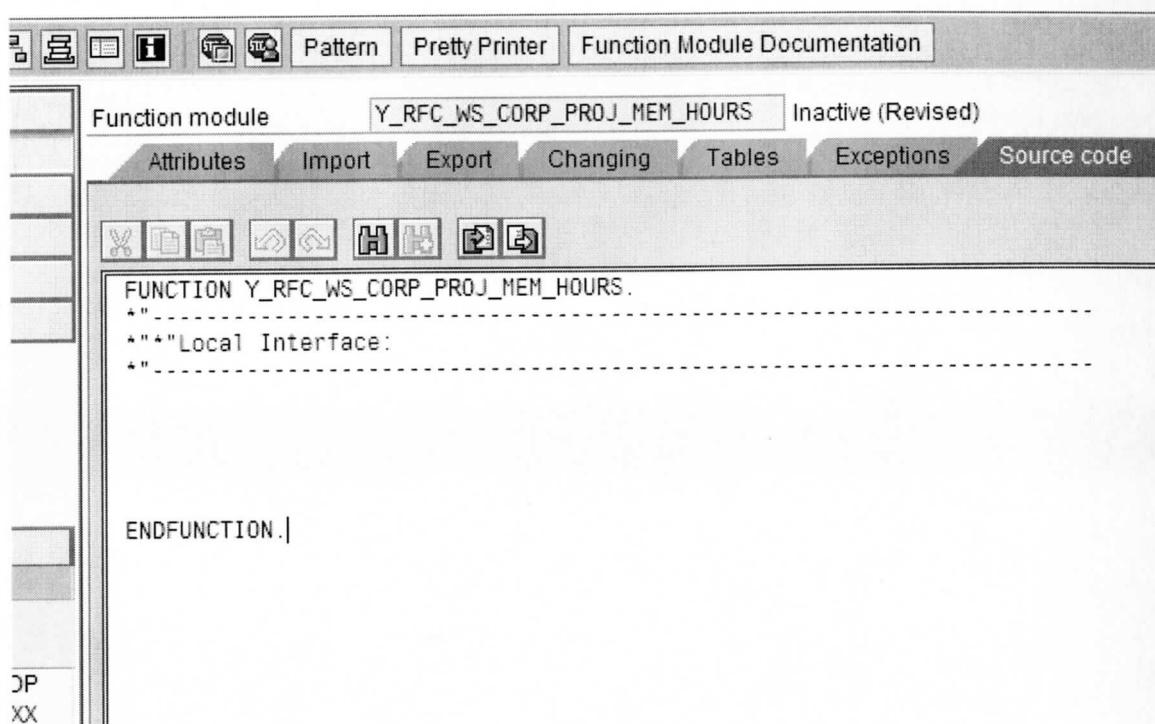


Figure 28.Default source code of a function module

Since we need to export the data into the Excel sheet and the data is of the form of the table as in the database, the export parameter PROJTAB is of the type YTABTYP_PROJ.

*"-----

DATA: IT_PROJTAB TYPE STANDARD TABLE OF YTABPROJ.

DATA: IT_PROJTAB_WEEK TYPE STANDARD TABLE OF YTABPROJ.

DATA: WA_PROJTAB TYPE YTABPROJ.

CALL FUNCTION 'Y_FM_PROJ_MEMBERS'

EXPORTING

PROJECT_ID = PROJECT_ID

IMPORTING

PROJECT_TAB = IT_PROJTAB.

**SELECT * FROM YTABPROJ APPENDING TABLE PROJTAB WHERE PROJID = PROJECT_ID AND
WEEK_ENDDATE = WEEK_ENDDATE.

IF WEEK_ENDDATE IS NOT INITIAL.

LOOP AT IT_PROJTAB INTO WA_PROJTAB WHERE WEEK_ENDDATE = WEEK_ENDDATE.

**READ * FROM IT_PROJTAB INTO WA_PROJTAB WITH KEY WEEK_ENDDATE = WEEK_ENDDATE.

APPEND WA_PROJTAB TO IT_PROJTAB_WEEK.

ENDLOOP.

ELSE.

IT_PROJTAB_WEEK = IT_PROJTAB.

ENDIF.

PROJTAB = IT_PROJTAB_WEEK.

ENDFUNCTION.

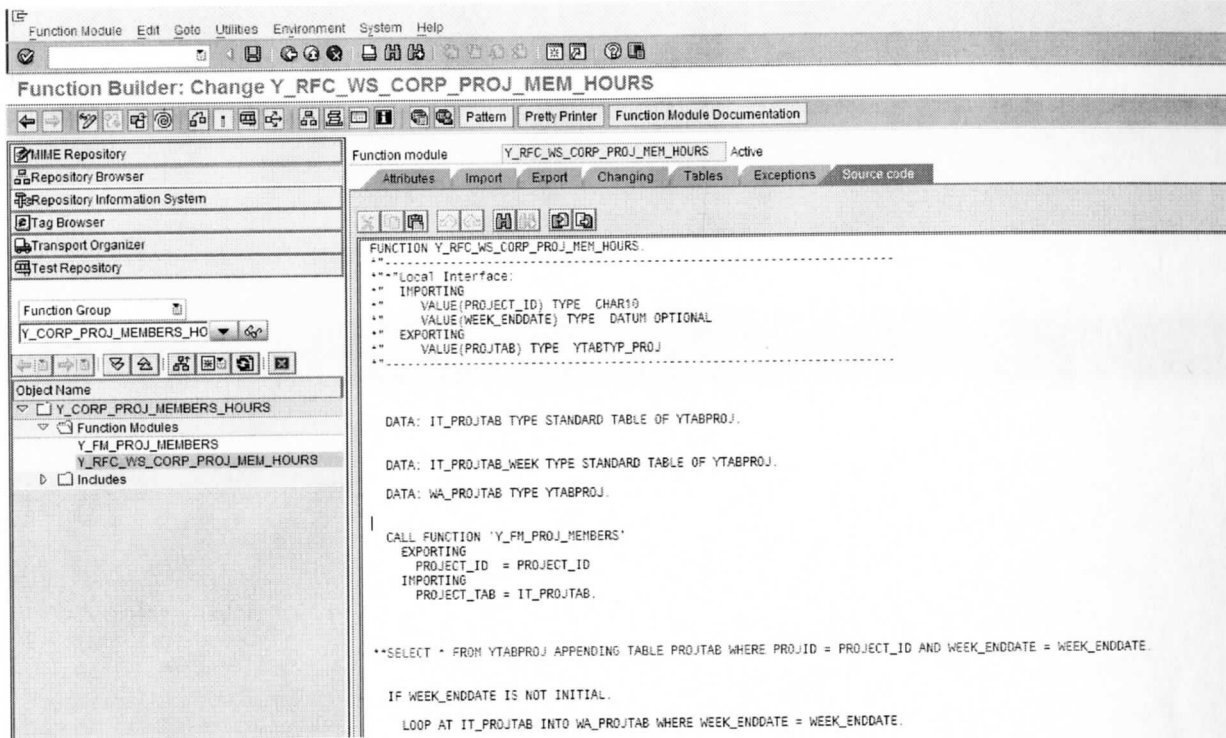


Figure 30. Source code of the function module

The remote function module Y RFC WS CORP PROJ MEM HOURS calls another function module Y FM PROJ MEMBERS. The source code Y FM PROJ MEMBERS is shown below

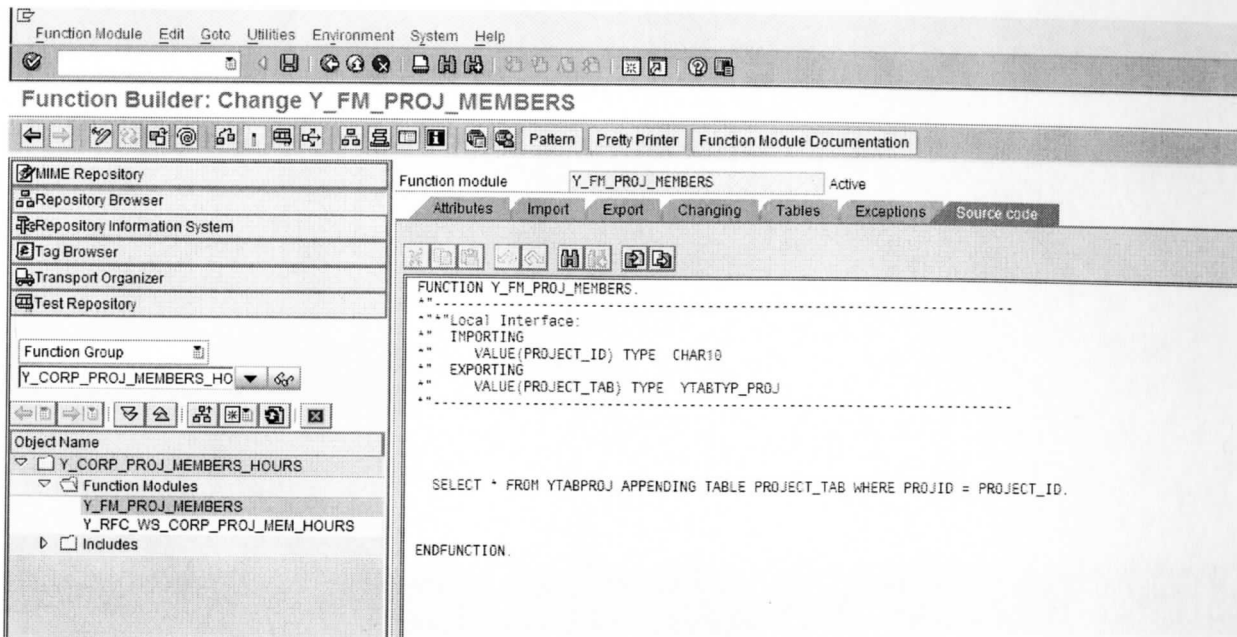


Figure 31. Source code of function module Y_FM_PROJ_MEMBERS

Function module Y_FM_PROJ_MEMBERS has an importing parameter PROJ_ID and an exporting parameter PROJECT_TAB.

It takes project id as an input and returns all the rows from the database table associated with that project id.

Executing function module Y_CORP_PROJ_MEMBERS_HOURS

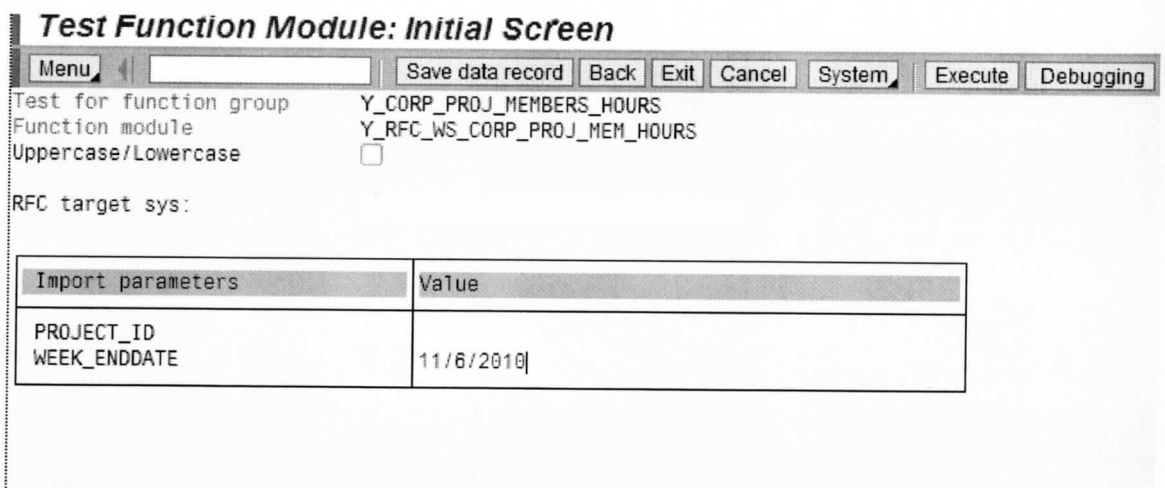


Figure 32. Executing function Y_CORP_PROJ_MEMBERS_HOURS

Result is displayed for the week end date of 11/6/2010

Test Function Module: Result Screen

Menu | Save data record | Back | Exit | Cancel | System | Choose

Test for function group Y_CORP_PROJ_MEMBERS_HOURS
 Function module Y RFC_WS_CORP_PROJ_MEM_HOURS
 Uppercase/Lowercase

Runtime: 1,853 Microseconds

RFC target sys:

Import parameters	Value
PROJECT_ID	
WEEK_ENDDATE	11/06/2010

Export parameters	Value
PROJTAB	1 Entry

Figure 33.Result screen-Y_CORP_PROJ_MEMBERS_HOURS

Clicking on PROJTAB 1 Entry shows the list of rows returned from the database table YTABPROJ

Structure Editor: Display PROJTAB from Entry 1

Menu | Back | Exit | Cancel | System | Single Entry | Left Margin | Scroll Left | Scroll Right | Right Margi

1 Entry

CLN	PROJID	WEEK_ENDDA	EMPID	LASTNAME	FIRSTNAME	HO	A
600		11/06/2010	01875943	JAIN	SHIKHA	40	Y

Figure 34.Display data from PROJTAB from results screen

ABAP Web Dynpro Object

Now the next step is to create an ABAP Web Dynpro object which will be a web service to export the data from SAP database to the excel sheet or to update SAP database from a excel sheet.

We can create ABAP Web Dynpro object in se80 transaction on ABAP workbench
The ABAP Web Dynpro object we are creating is Y_WDA_PROJ_MEMB_HOURS.

Name	Y_WDA_PROJ_MEMB_HOURS
Description	Y_WDA_PROJ_MEMB_HOURS
Type	<input checked="" type="radio"/> Web Dynpro Component <input type="radio"/> Web Dynpro Component Interface
Window Name	Y_WDA_PROJ_MEMB_HOUR
View Name	MAIN

Figure 35. Creating ABAP Web Dynpro Object

To call the remote function module we created earlier from ABAP Web Dynpro object we need to create a service call from Web Dynpro Object as shown below

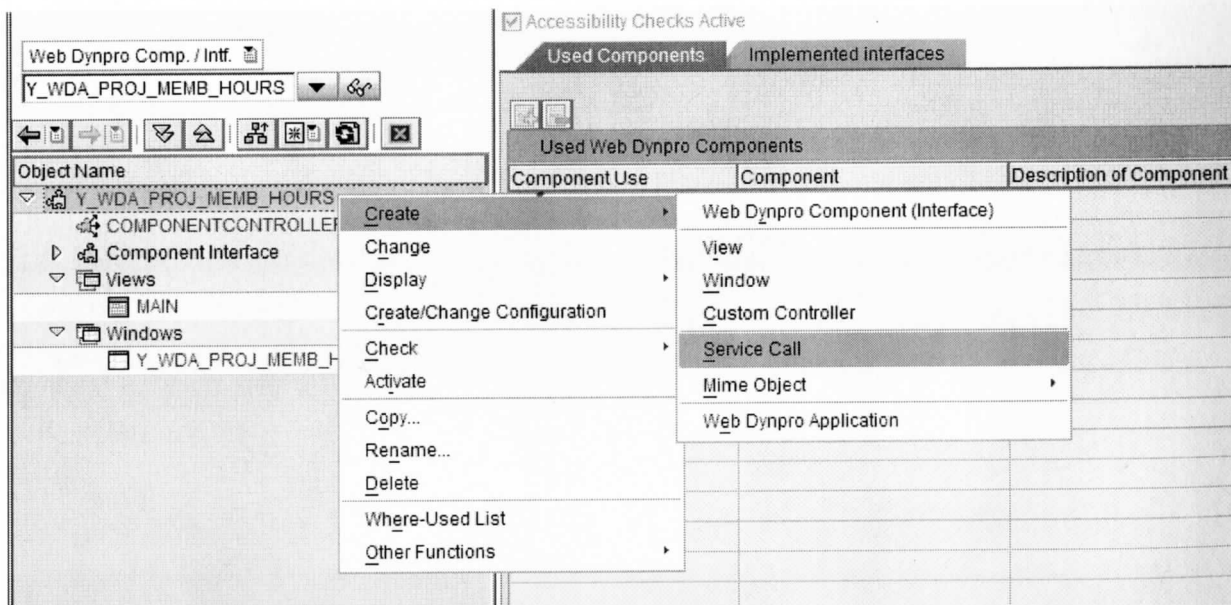


Figure 36. Create Service call from Web Dynpro Object

This opens a wizard which will automatically create a code to call RFC from Web Dynpro Object.

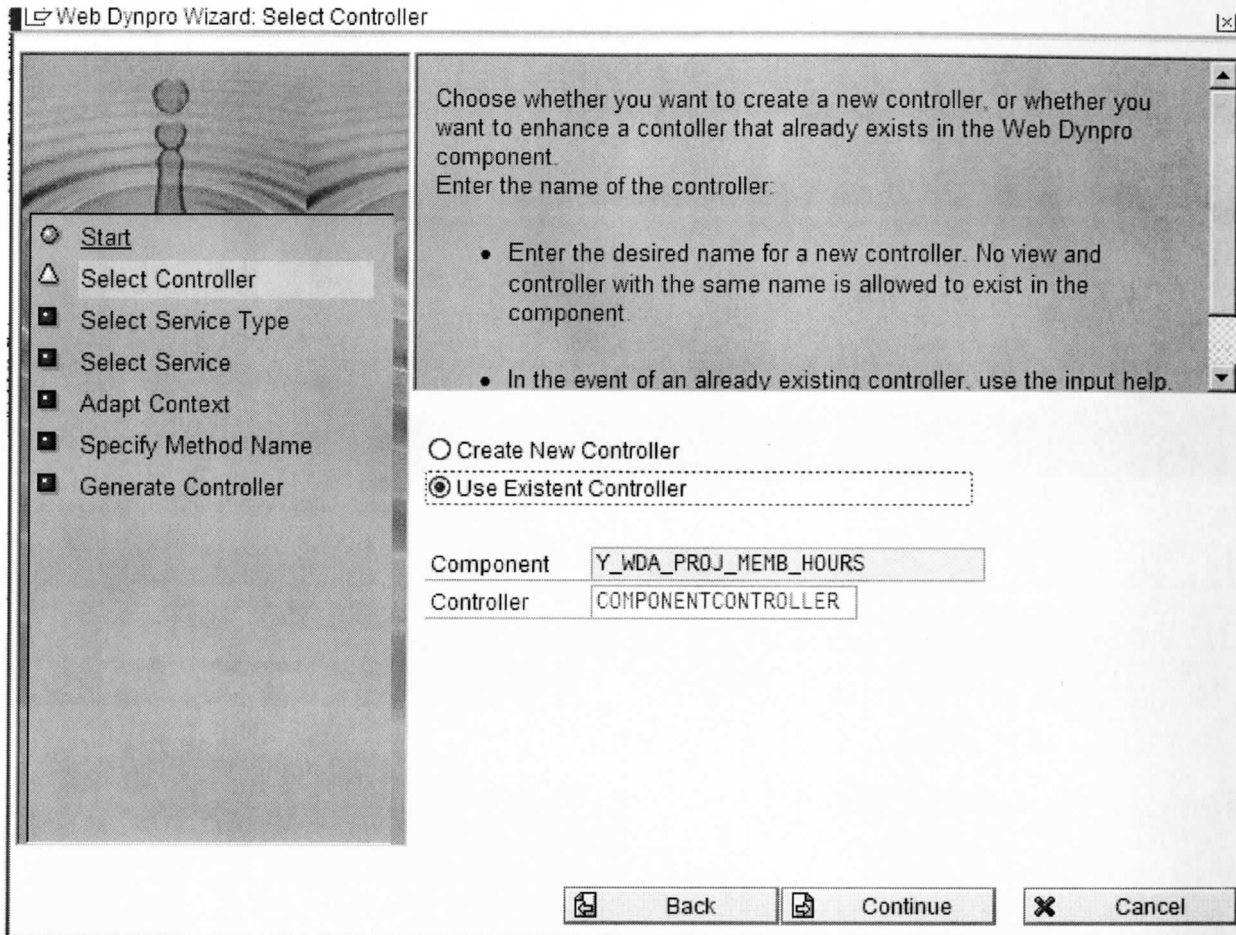


Figure 37. Web Dynpro wizard

In the wizard we select the RFC name to be used in the Web Dynpro Service Call

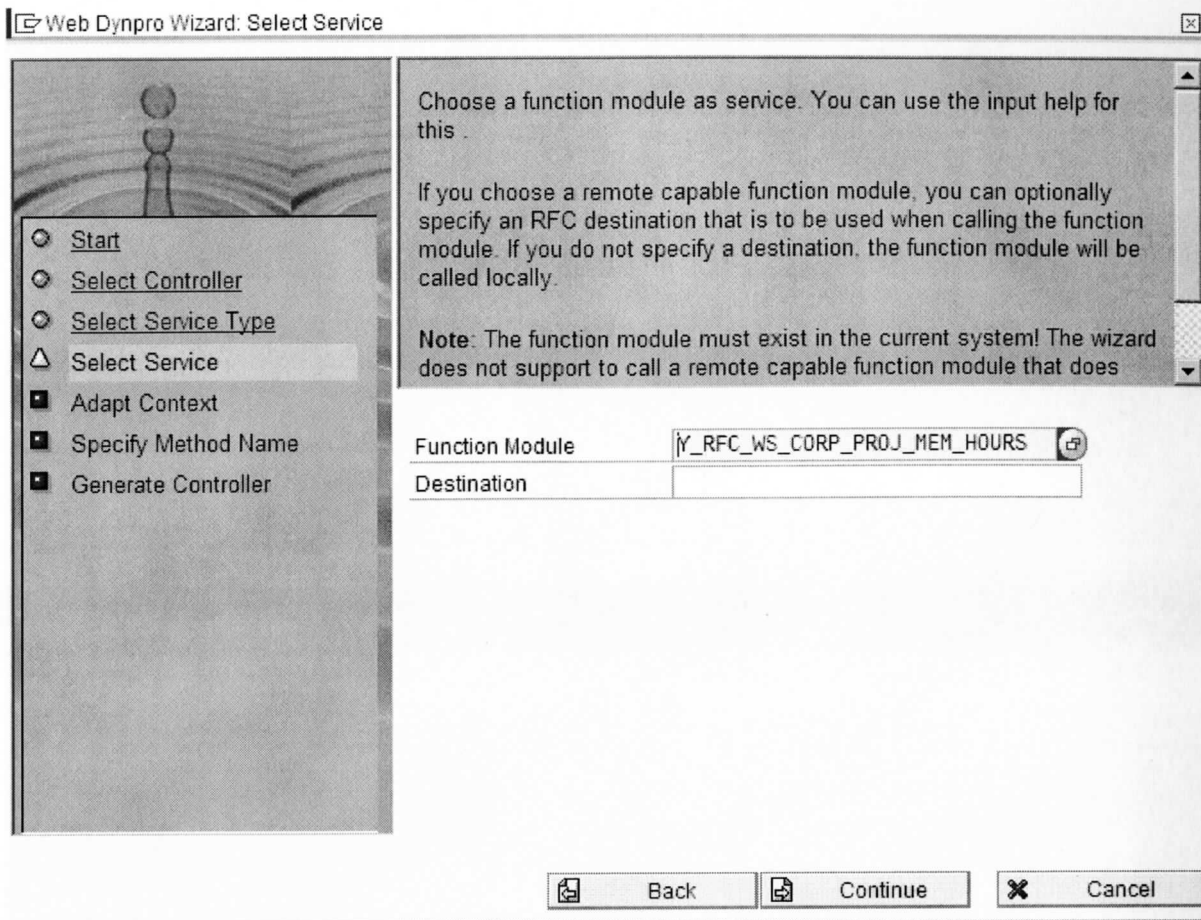


Figure 38. Web Dynpro Wizard: Select Service

Then we map controller's context with view's context as shown below.

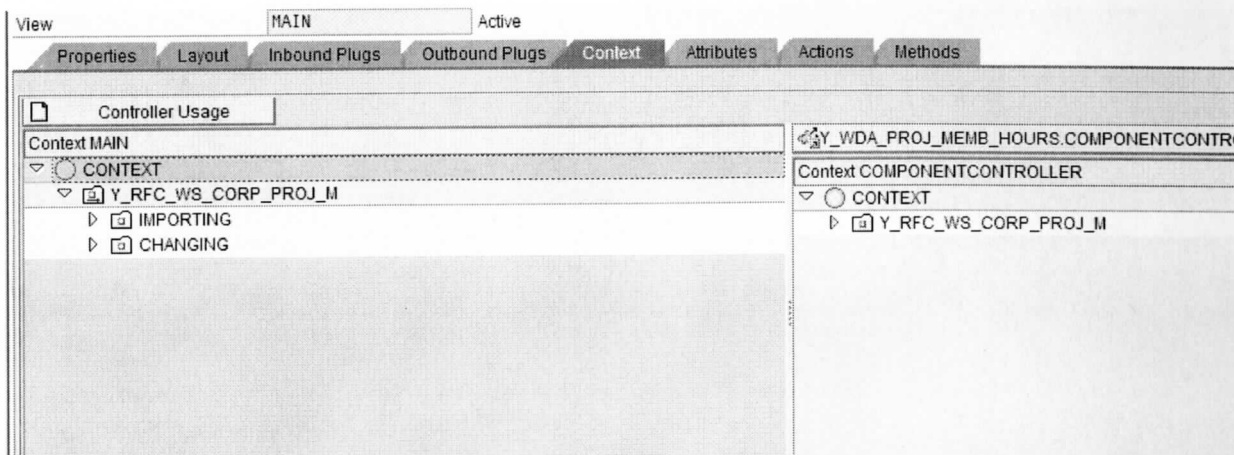


Figure 39. Mapping controller context with view context

Now we are designing the end user view where user can input the parameters PROJECT_ID and WEEK_ENDDATE and get the information from the SAP database.

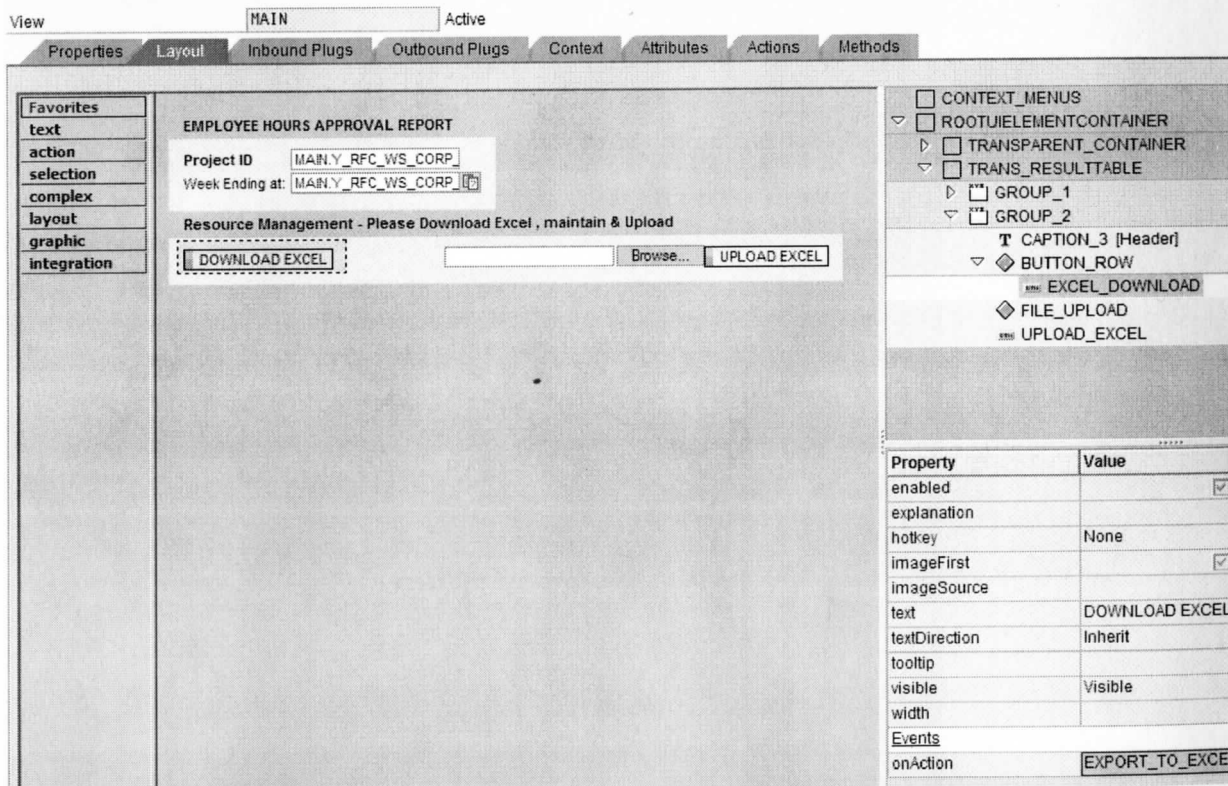


Figure 40. Designing end user view

Download to Excel Sheet

To download the results to excel sheet we need to create context variables in the web dynpro object which can map to the user defined input and the variables which can map to the excel sheet columns which needs to be generated. Figure below shows the IMPORTING variables as PROJ_ID and WEEK_ENDDATE which map to the user input. The CHANGING variables which are defined in the figure below map to the columns in the excel sheet to be generated

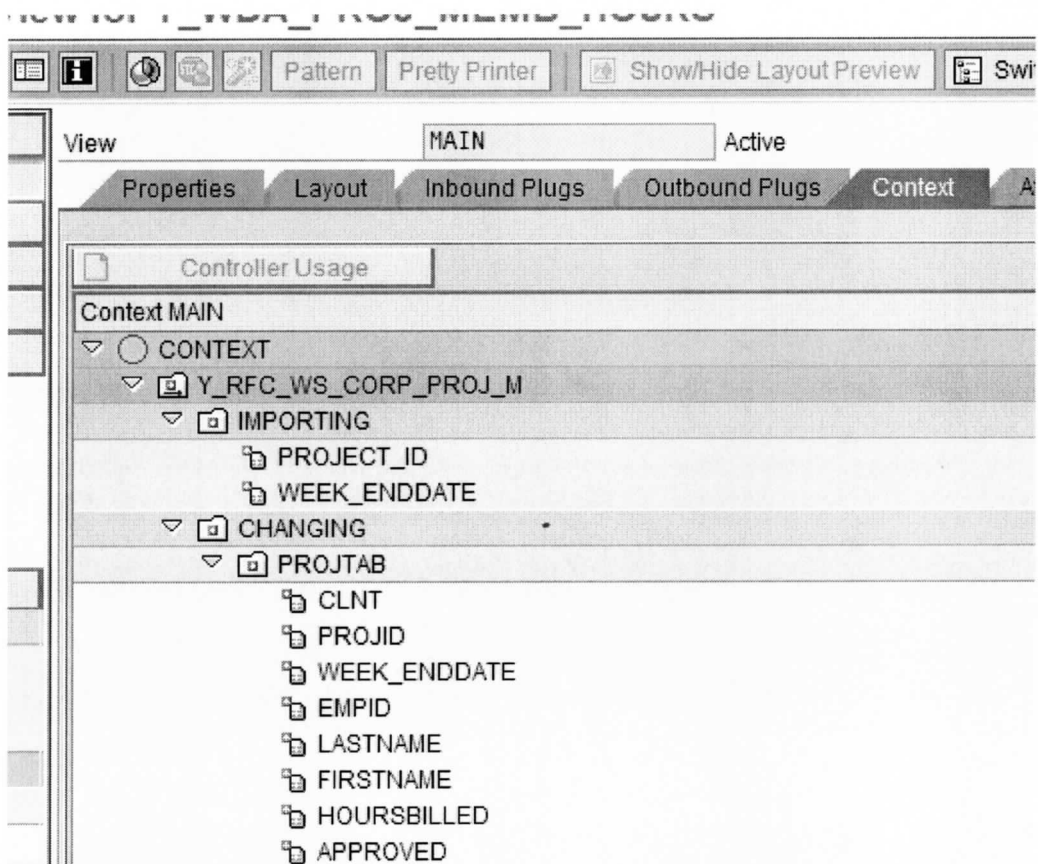


Figure 41. Context mapping for Download Excel

To download the results into excel sheet after the end user enters the input parameters PROJ_ID and WEEK_END_DATE we have created the button **Download Excel** at the end user view layout. This button is linked to action method Export_To_Excel of the Web Dynpro object Y_WDA_PROJ_MEMB_HOURS as shown in Figure 39

The method Export_To_Excel is shown below:-

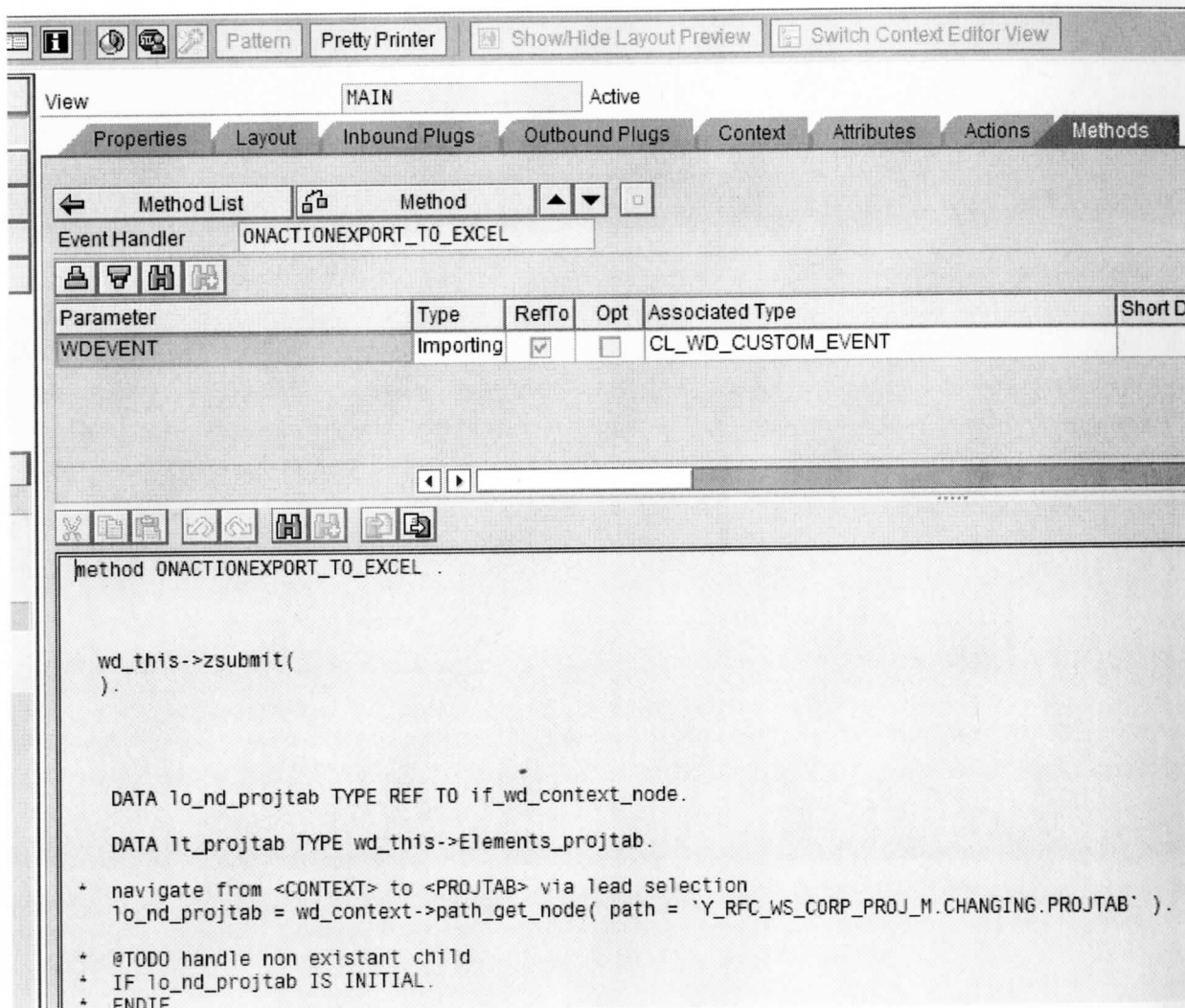


Figure 42. Method Export_To_Excel

Below is the code of method Export_To_Excel. This method basically first calls the method Z_SUBMIT to retrieve the values from the database depending upon user input by

calling RFC Y_RFC_PROJ_MEM_HOURS. Then it converts the data into the excel format to show it to the end user.

```
method ONACTIONEXPORT_TO_EXCEL .
```

```
* Calling the execute method of the rfc ( Code of this custom execute method showed in
screen below ) -
```

```
wd_this->zsubmit(
).
```

```
* Below code to convert the context ( output of the rfc ) to excel -
```

```
DATA lo_nd_projtab TYPE REF TO if_wd_context_node.
```

```
DATA lt_projtab TYPE wd_this->Elements_projtab.
```

```
* navigate from <CONTEXT> to <PROJTAB> via lead selection
lo_nd_projtab = wd_context->path_get_node( path =
`Y_RFC_WS_CORP_PROJ_M.CHANGING.PROJTAB` ).
```

```
* @TODO handle non existant child
* IF lo_nd_projtab IS INITIAL.
* ENDIF.
```

```
lo_nd_projtab->get_static_attributes_table( importing table = lt_projtab ).
```

```
DATA LW_PROJTAB TYPE ytabproj.
data str type string.
data xstr type xstring.
```

```
concatenate str
    'PROJECT ID'
    'WEEK ENDING DATE'
    'EMPLOYEE ID'
    'LAST NAME'
    'FIRST NAME'
    'HOURS'
    'APPROVED'
    cl_abap_char_utilities=>newline into str
    separated by cl_abap_char_utilities=>horizontal_tab.
```

```
Loop at lt_projtab into lw_projtab.
```

```
concatenate str
    LW_PROJTAB-PROJID
    LW_PROJTAB-WEEK_ENDDATE
    LW_PROJTAB-EMPID
    LW_PROJTAB-LASTNAME
```

```
LW_PROJTAB-FIRSTNAME
LW_PROJTAB-HOURSBILLED
LW_PROJTAB-APPROVED
cl_abap_char_utilities=>newline into str
    separated by cl_abap_char_utilities=>horizontal_tab.
endloop.
```

```
CALL FUNCTION 'SCMS_STRING_TO_XSTRING'
  EXPORTING
    text          = str
  * MIMETYPE      = ''
  * ENCODING      =
  IMPORTING
    BUFFER        = xstr
  EXCEPTIONS
    FAILED        = 1.

CALL METHOD cl_wd_runtime_services=>attach_file_to_response
  EXPORTING
    i_filename     = 'Hours.xls'
    i_content      = xstr
    i_mime_type    = 'EXCEL'
    i_in_new_window = ABAP_FALSE
    i_inplace      = ABAP_FALSE.

endmethod.
```

The method Z_SUBMIT is called by the method above to execute the RFC Y_RFC_PROJ_MEM_HOURS to get the values depending upon the input values from the user

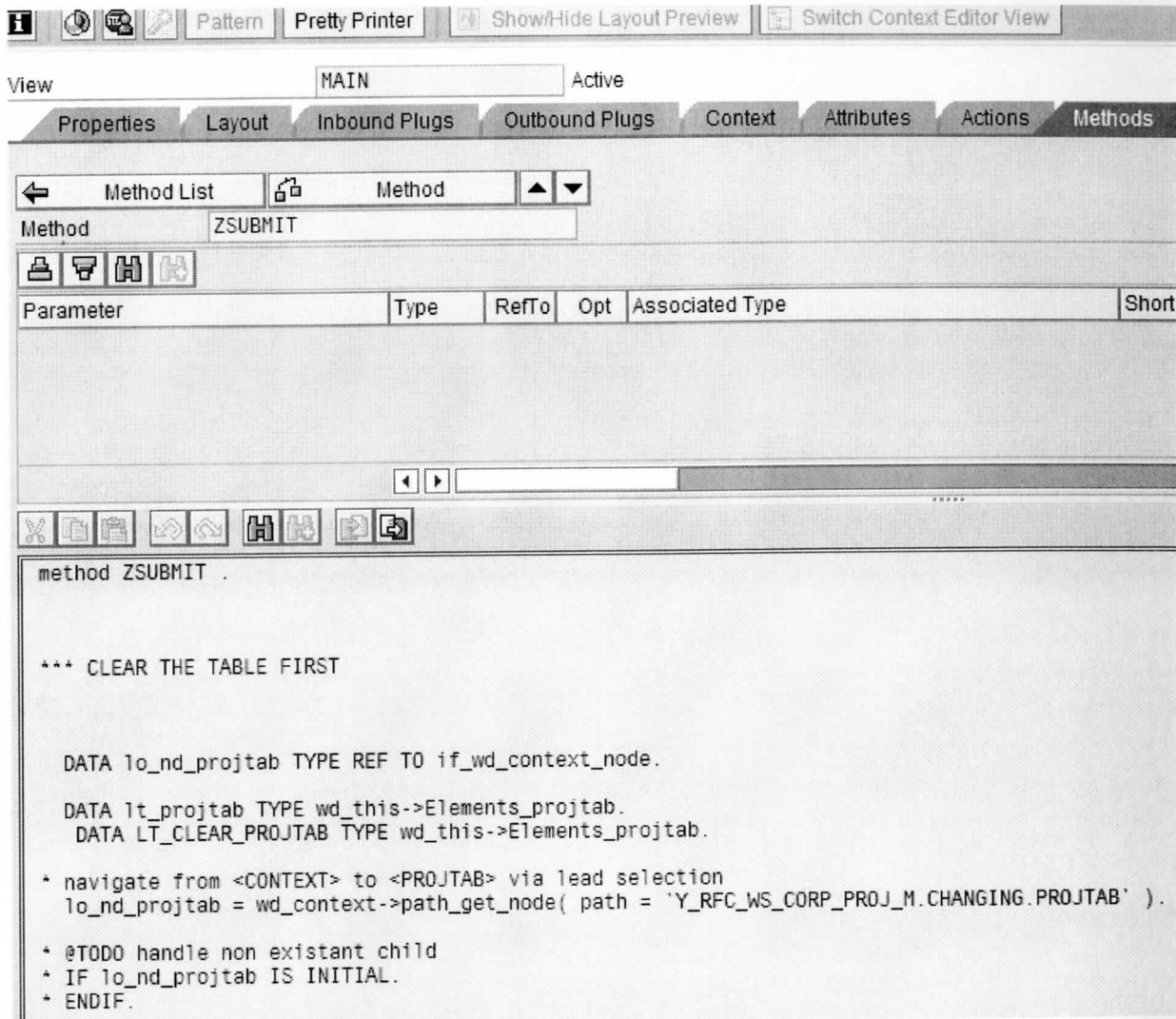


Figure 43. Method Z_SUBMIT

Below is the code shown for Z_SUBMIT:-

```
method ZSUBMIT .

*** CLEAR THE TABLE FIRST

DATA lo_nd_projtab TYPE REF TO if_wd_context_node.

DATA lt_projtab TYPE wd_this->Elements_projtab.
DATA LT_CLEAR_PROJTAB TYPE wd_this->Elements_projtab.

* navigate from <CONTEXT> to <PROJTAB> via lead selection
```

```

        lo_nd_projtab = wd_context->path_get_node( path =
`Y RFC_WS_CORP_PROJ_M.CHANGING.PROJTAB` ).

    * @TODO handle non existant child
    * IF lo_nd_projtab IS INITIAL.
    * ENDIF.

    ** @TODO compute values
    ** e.g. call a model function
    *
    ** lo_nd_projtab->bind_table( new_items = lt_projtab set_initial_elements =
abap_true ).

    lo_nd_projtab->bind_table( LT_CLEAR_PROJTAB ).

*** EXECUTE THE RFC

DATA lo_COMPONENTCONTROLLER TYPE REF TO IG_COMPONENTCONTROLLER .
lo_COMPONENTCONTROLLER = wd_this->get_componentcontroller_ctr( ).

    lo_componentcontroller->execute_y_rfc_ws_corp_proj_mem(
    ).

endmethod.

```

Upload Excel Sheet into SAP database

Figure below shows the UI Element we have created to upload the excel file and update the data into SAP

database.

The screenshot displays a web application interface for an 'EMPLOYEE HOURS APPROVAL REPORT'. The form includes fields for 'Project ID' and 'Week Ending at', both containing the value 'MAIN.Y_RFC_WS_CORP'. Below these fields is a section titled 'Resource Management - Please Download Excel , maintain & Upload' containing two buttons: 'DOWNLOAD EXCEL' and 'UPLOAD EXCEL'. A dashed box highlights the 'UPLOAD EXCEL' button. To the right, a tree view shows the UI component hierarchy, with 'FILE_UPLOAD' and 'UPLOAD_EXCEL' selected. Below the tree is a table of properties for the 'FileUpload' component.

Property	Value
Properties (FileUpload)	
ID	FILE_UPLOAD
activateAccessKey	<input type="checkbox"/>
contextMenuBehaviour	Inherit

Figure 44. UPLOAD EXCEL UI Element

To integrate this UI element with the web dynpro object we need to define a context (variable) named DATASOURCE which can hold the excel sheet as a buffer string.

View MAIN Active

Properties Layout Inbound Plugs Outbound Plugs **Context** Attributes Actions

Controller Usage

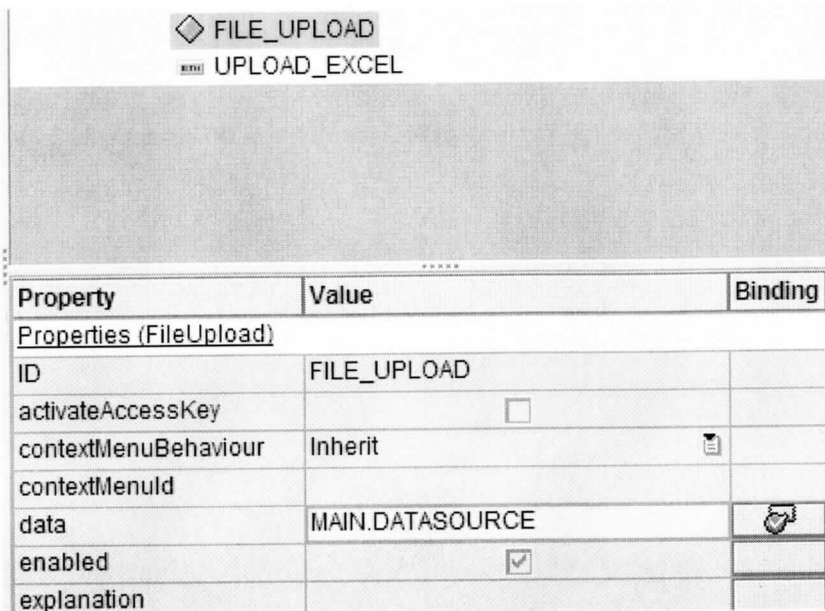
Context MAIN

- CONTEXT
 - Y_RFC_WS_CORP_PROJ_M
 - DATASOURCE**

Property	Value
<u>Attribute</u>	
Attribute Name	DATASOURCE
Type assignment	Type
Type	STRING
Read-only	
Default Value	
Null Value	
Input Help Mode	Automatic
Determined Input Help	
Type of Input Help	
<u>Formatting</u>	
Compression	Default Value

Figure 45. Buffer string to hold uploaded excel file

To browse the excel file on the users system we create a File Upload element as shown below



Property	Value	Binding
Properties (FileUpload)		
ID	FILE_UPLOAD	
activateAccessKey	<input type="checkbox"/>	
contextMenuBehaviour	Inherit	
contextMenuId		
data	MAIN.DATASOURCE	
enabled	<input checked="" type="checkbox"/>	
explanation		

Figure 46. File upload element to browse the file on the users system

The UI Element UPLOAD EXCEL is linked to the method UPLOAD_EXCEL on web dynpro object Y_WDA_PROJ_MEMB_HOURS

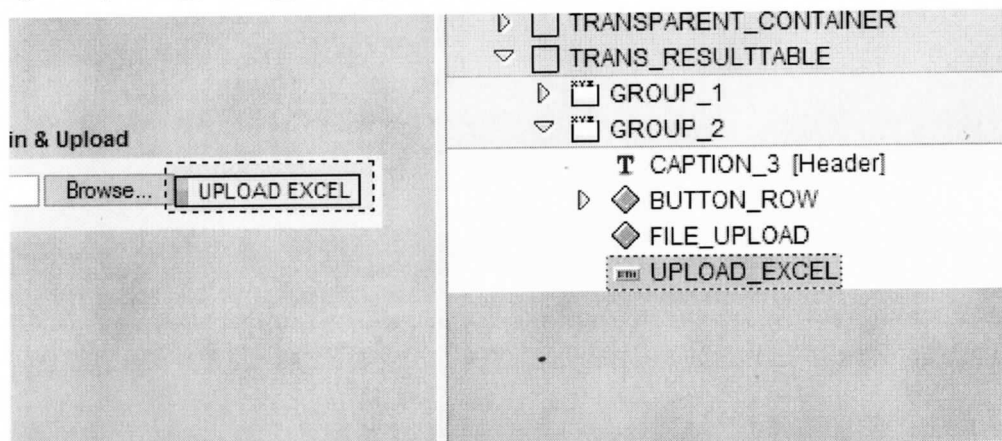


Figure 47. UI Element UPLOAD EXCEL linked to method UPLOAD_EXCEL

The method UPLOAD_EXCEL is shown below

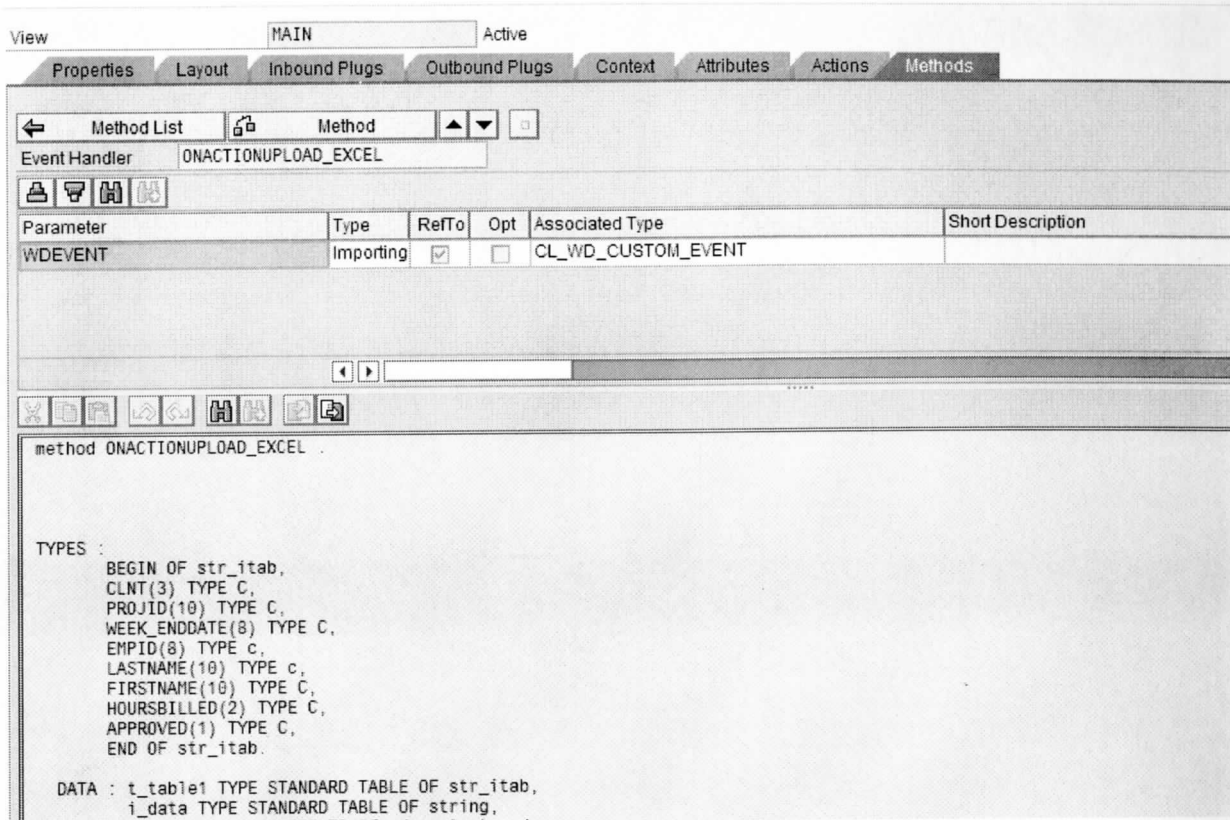


Figure 48. Method `UPLOAD_EXCEL`

The code of `UPLOAD_EXCEL` is below

```
method ONACTIONUPLOAD_EXCEL .
```

```

TYPES :
    BEGIN OF str_itab,
        CLNT(3) TYPE C,
        PROJID(10) TYPE C,
        WEEK_ENDDATE(8) TYPE C,
        EMPID(8) TYPE c,
        LASTNAME(10) TYPE c,
        FIRSTNAME(10) TYPE C,
        HOURS_BILLED(2) TYPE C,
        APPROVED(1) TYPE C,
    END OF str_itab.

DATA : t_table1 TYPE STANDARD TABLE OF str_itab,
        i_data TYPE STANDARD TABLE OF string,
        lo_nd_data TYPE REF TO if_wd_context_node,
        lo_el_data TYPE REF TO if_wd_context_element,
        l_string TYPE string,
        fs_table TYPE str_itab,
        l_xstring TYPE xstring,

```

```

        fields TYPE string_table,
        lv_field TYPE string.

** DATA : t_table TYPE if_main=>elements_PROJTAB,
**        data_table TYPE if_main=>elements_PROJTAB.

DATA lo_nd_projtab TYPE REF TO if_wd_context_node.

DATA lo_el_projtab TYPE REF TO if_wd_context_node.

DATA T_TABLE TYPE wd_this->Elements_projtab.
DATA DATA_TABLE TYPE wd_this->Elements_projtab.
DATA lt_projtabprev TYPE wd_this->Elements_projtab.

**** navigate from <CONTEXT> to <PROJTAB> via lead selection
    lo_nd_projtab = wd_context->path_get_node( path =
`Y_RFC_WS_CORP_PROJ_M.CHANGING.PROJTAB` ).

****
**** @TODO handle non existant child
**** IF lo_nd_projtab IS INITIAL.
**** ENDF.
****
**** cLEAR THE CURRENT TABLE.
****
**** lo_nd_projtab->get_static_attributes_table( importing table = lt_projtabprev ).
****
****DATA: WA_TEMP TYPE YTABPROJ.
****
**** LOOP AT lt_projtabprev INTO wa_temp.
***   lo_nd_projtab->remove_element( EXPORTING element = lo_nd_projtab ).
****   ENDLLOOP.

* get single attribute
  wd_context->get_attribute(
    EXPORTING
      name = `DATASOURCE`
    IMPORTING
      value = l_xstring ).

CALL FUNCTION 'HR_KR_XSTRING_TO_STRING'
  EXPORTING
    in_xstring = l_xstring
  IMPORTING
    out_string = l_string.

SPLIT l_string AT cl_abap_char_utilities=>newline INTO TABLE i_data.

* Bind With table Element.
LOOP AT i_data INTO l_string.
  SPLIT l_string AT cl_abap_char_utilities=>horizontal_tab INTO TABLE fields.

```



```

*   READ TABLE fields INTO lv_field INDEX 1.
*   fs_table-CLNT = lv_field.
   READ TABLE fields INTO lv_field INDEX 1.

IF LV_FIELD NE 'PROJECT ID'.

   fs_table-PROJID = lv_field.
   READ TABLE fields INTO lv_field INDEX 2.
   fs_table-WEEK_ENDDATE = lv_field.
   READ TABLE fields INTO lv_field INDEX 3.
   fs_table-EMPID = lv_field.
   READ TABLE fields INTO lv_field INDEX 4.
   fs_table-LASTNAME = lv_field.
   READ TABLE fields INTO lv_field INDEX 5.
   fs_table-FIRSTNAME = lv_field.
   READ TABLE fields INTO lv_field INDEX 6.
   fs_table-HOURSBILLED = lv_field.
   READ TABLE fields INTO lv_field INDEX 7.
   fs_table-APPROVED = lv_field.

   APPEND fs_table TO t_table1.

ENDIF.

ENDLOOP.

**lo_nd_data = wd_context->get_child_node( 'PROJTAB' ).
**lo_nd_data->bind_table( T_TABLE1 ).

CLEAR DATA_TABLE.
*
lo_nd_projtab->bind_table( DATA_TABLE ).

lo_nd_projtab->bind_table( new_items = T_TABLE1 set_initial_elements = abap_false
).

**=====

*** DATA lo_nd_projtab TYPE REF TO if_wd_context_node.
***
*** DATA lt_projtab TYPE wd_this->Elements_projtab.
***
**** navigate from <CONTEXT> to <PROJTAB> via lead selection
*** lo_nd_projtab = wd_context->path_get_node( path =
`Y_RFC_WS_CORP_PROJ_M.CHANGING.PROJTAB` ).
***
**** @TODO handle non existant child
**** IF lo_nd_projtab IS INITIAL.
**** ENENDIF.
***
*** lo_nd_projtab->get_static_attributes_table( importing table = lt_projtab ).
***
***
***

```



```

**=====
*** DATA lo_nd_projtab TYPE REF TO if_wd_context_node.
***
*** DATA lt_projtab TYPE wd_this->Elements_projtab.
***
**** navigate from <CONTEXT> to <PROJTAB> via lead selection
*** lo_nd_projtab = wd_context->path_get_node( path =
`Y_RFC_WS_CORP_PROJ_M.CHANGING.PROJTAB` ).
***
**** @TODO handle non existant child
**** IF lo_nd_projtab IS INITIAL.
**** ENDIF.
***
***** @TODO compute values
***** e.g. call a model function
****
*** lo_nd_projtab->bind_table( new_items = lt_projtab set_initial_elements =
abap_true ).
***

wd_this->save_db( ).

endmethod.

```

The web dynpro context PROJTAB where the excel sheet is uploaded is shown below

Property	Value
Nodes	
Node Name	PROJTAB
Dictionary structure	YTABPROJ
Cardinality	0..n
Selection	0..1
Initialization Lead Selection	<input type="checkbox"/>
Singleton	<input type="checkbox"/>
Supply Function	<input type="checkbox"/>

Figure 49. Upload Excel context in web dynpro object

The context table PROJTAB is saved to the database table by the method SAVE_DB.

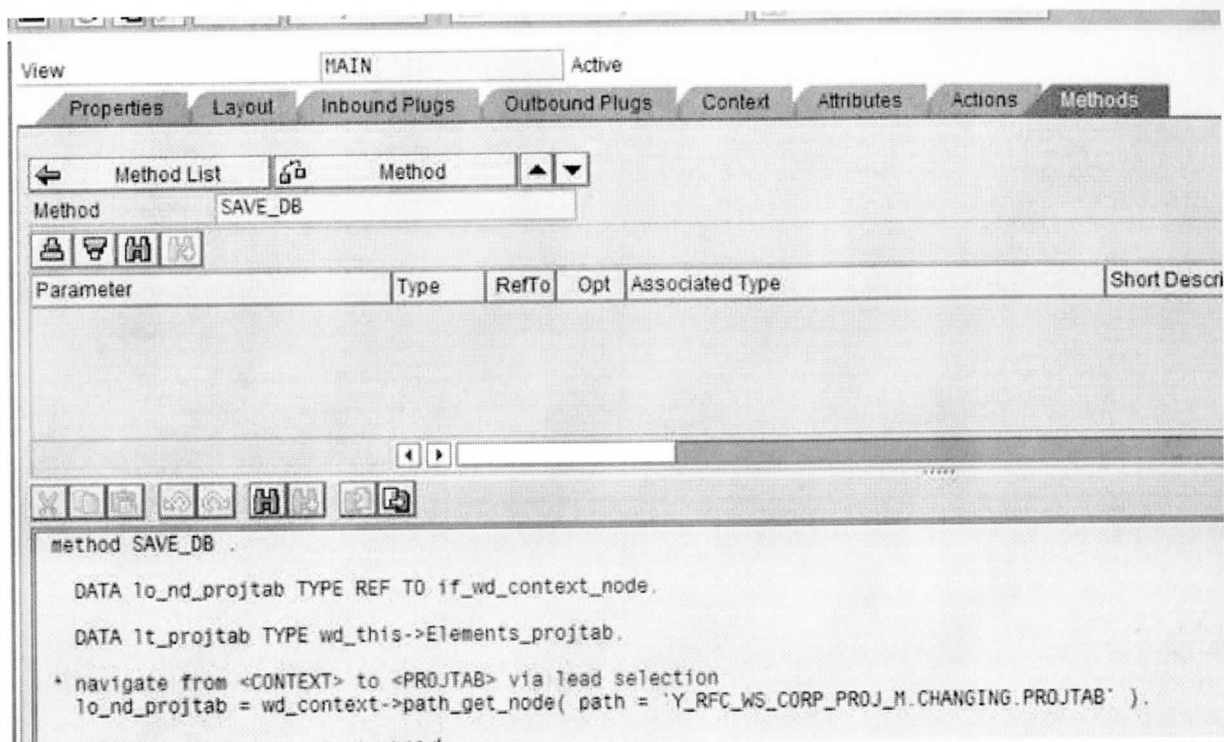


Figure 50. Method SAVE_DB

The code for the method SAVE_DB is as below:-

```
method SAVE_DB .

    DATA lo_nd_projtab TYPE REF TO if_wd_context_node.

    DATA lt_projtab TYPE wd_this->Elements_projtab.

    * navigate from <CONTEXT> to <PROJTAB> via lead selection
    lo_nd_projtab = wd_context->path_get_node( path = `Y_RFC_WS_CORP_PROJ_M.CHANGING.PROJTAB`
    ).

    * @TODO handle non existant child
    * IF lo_nd_projtab IS INITIAL.
    * ENDIF.

    lo_nd_projtab->get_static_attributes_table( importing table = lt_projtab ).

DATA: WT_TABLE1 TYPE YTABPROJ.

LOOP AT lt_projtab INTO WT_TABLE1.

MODIFY YTABPROJ FROM WT_TABLE1.
```

ENDLOOP.

Endmethod

Web Service

After we have created the methods to import data into excel and update data from excel into SAP database we need to create a web service from the ABAP web dynpro object which acts an interface to the end user and the SAP application. For that first we need to create web dynpro application from abap web dynpro object/component.

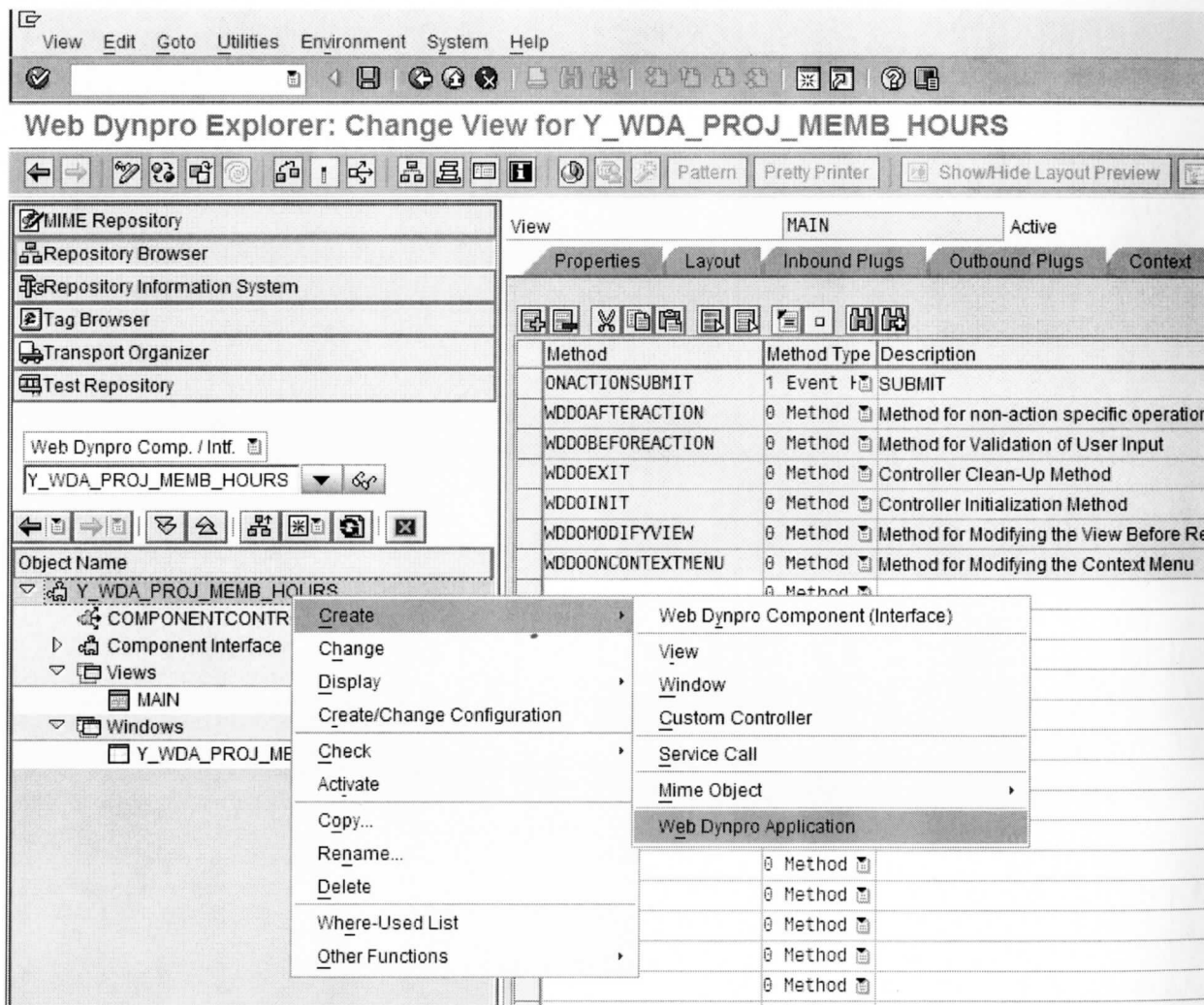


Figure 51. Creating Web dynpro application

Following figure shows that web dynpro application y_wda_proj_mem_hours is created. Then we need to activate the application to execute it.

The screenshot shows the SAP 'Maintain service' interface. The table below lists various services and their details.

Virtuelle Hosts / Services	Documentation	Referenz Service
yttest1234	test appl.	
yttest_01	Test Application	
yttest_3303	testing	
yttest_adobe	Test	
yttest_dyn_atr	Web Dynpro Application yttest_dyn_atr / Co...	
yttest_excel_upd	Test Application	
yttest_sm30	Test Application	
yttest_sm30_kb	dasdasda	
yttest_sm30_utility	SM30 Utility Application	
yttest_std_screen	Test Application	
yttest_tablock	Web Dynpro Application yttest_tablock / Co...	
y_bapi_get_detail	Get details of flight	
y_bapi_get_sales	Web Dynpro Application y_bapi_get_sales /...	
y_bapi_xml_download	Using XML Download to XLS	
y_wda_proj_mem_hours	y_wda_proj_mem_hours	
y_wdt_flightlist_config	view container example	
zmwdgt3355_pay_inq2	PAYMENT INQUIRY	
zmwdgt3356_vendor_inquiry	Vendor Tree with Bank and Address Details	
zmwdgt3356_vend_bank_addr	View of Vendor bank and address details	
zmwdgt3357_email_notify	Email notification	
zwdgt3355_pay_inq	Payment_inquiry	
ZWWDGT3355_Pay_Inq_app	Payment Inquiry App	
zwdgt3356_email	Send emails	

Figure 52. Activating web dynpro application

After activating the application we can execute the application as shown below

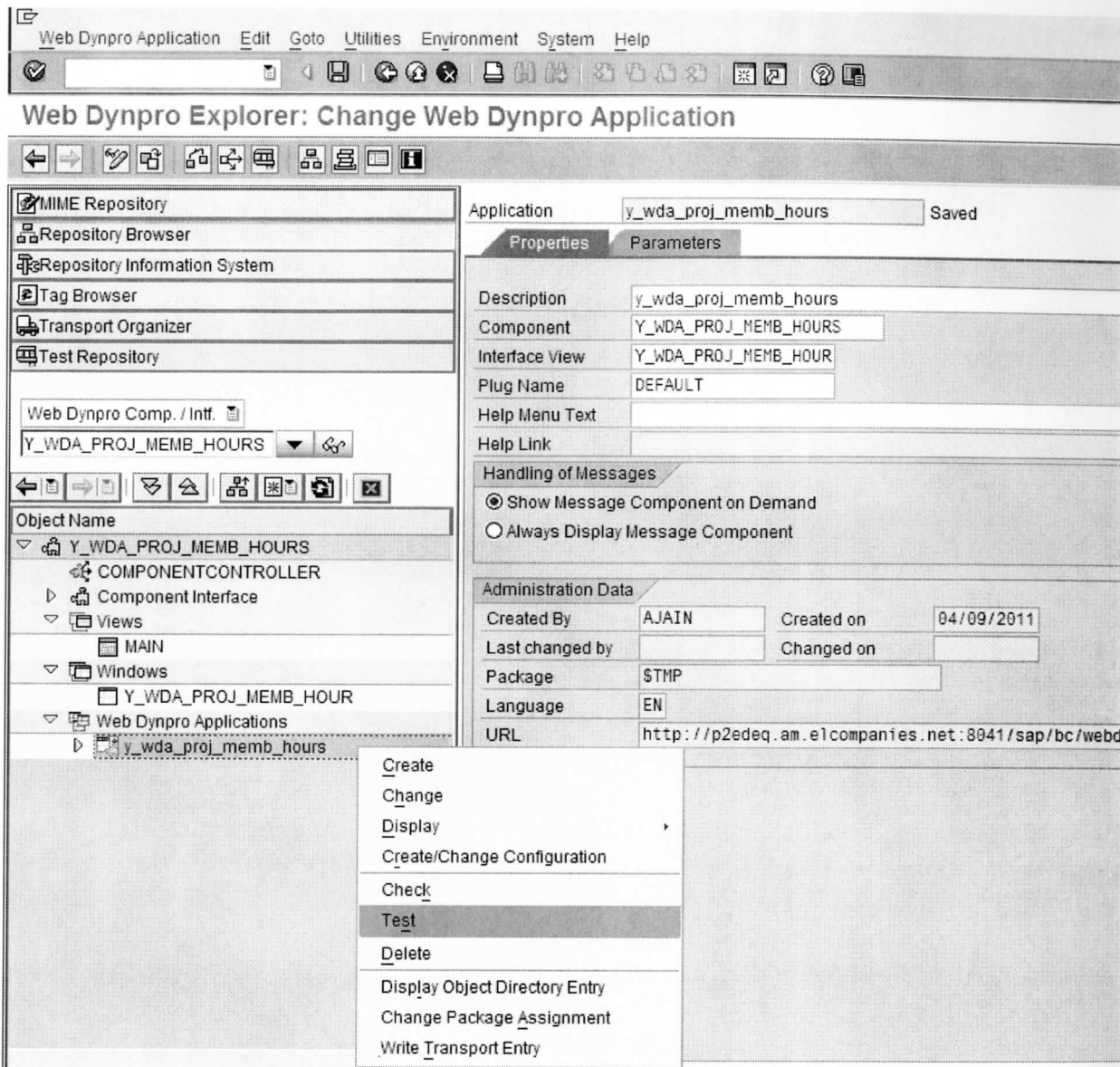


Figure 53. Executing web dynpro application

Executing web dynpro application takes us to the following screen

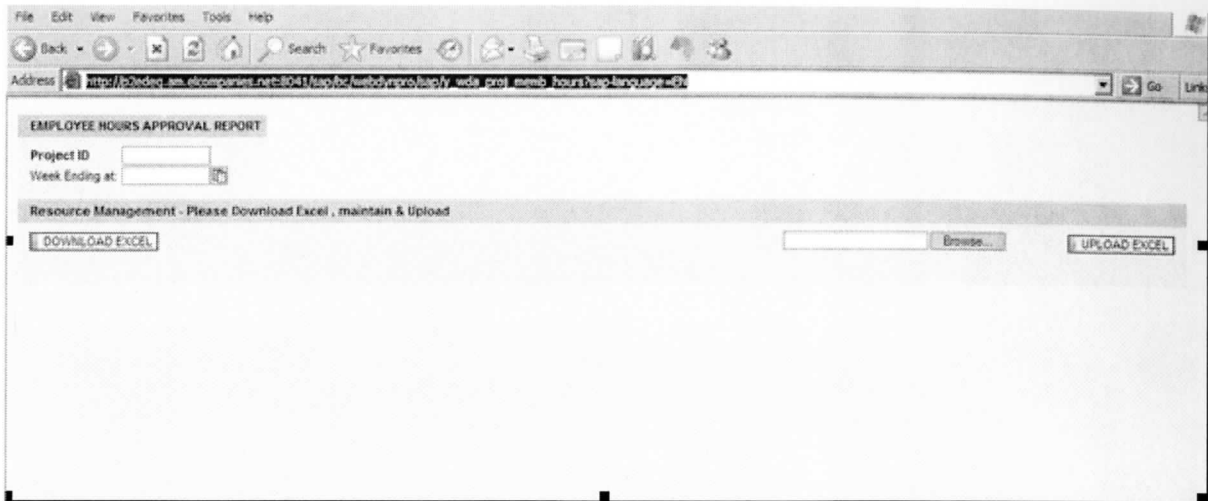


Figure 54. User View

To the users we can give the URL which is provided by the Web Dynpro application.

The users can use the above URL as long as they are in the companies secure vpn connection. But they don't need to have SAP installed on their systems. This URL acts as a web service which is independent of the SAP application but can be used from anywhere.

CHAPTER 4

RESULTS

This section contains the result table generated in the MS Excel after user specified input and also the SAP database which is updated upon upload of MS Excel sheet.

Download Excel

First we will show how the excel sheet is generated on user specified input criteria. The users enters the PROJECT ID as PROJECTDSU.

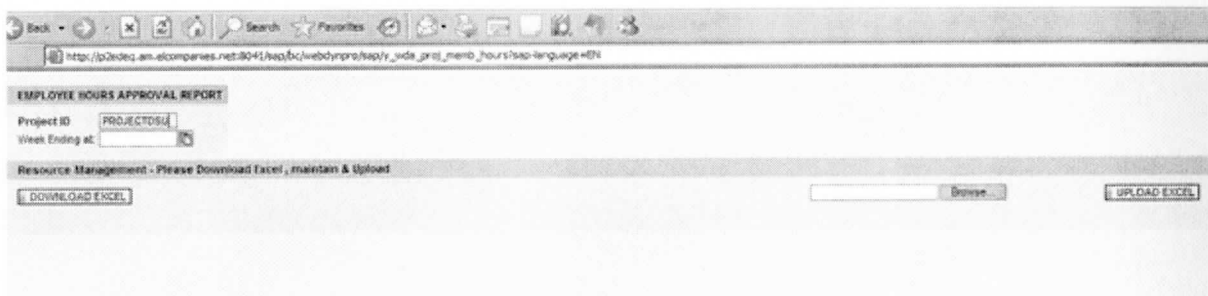


Figure 55. User entered PROJECT ID

The users then clicks DOWNLOAD EXCEL button which pops up the following screen

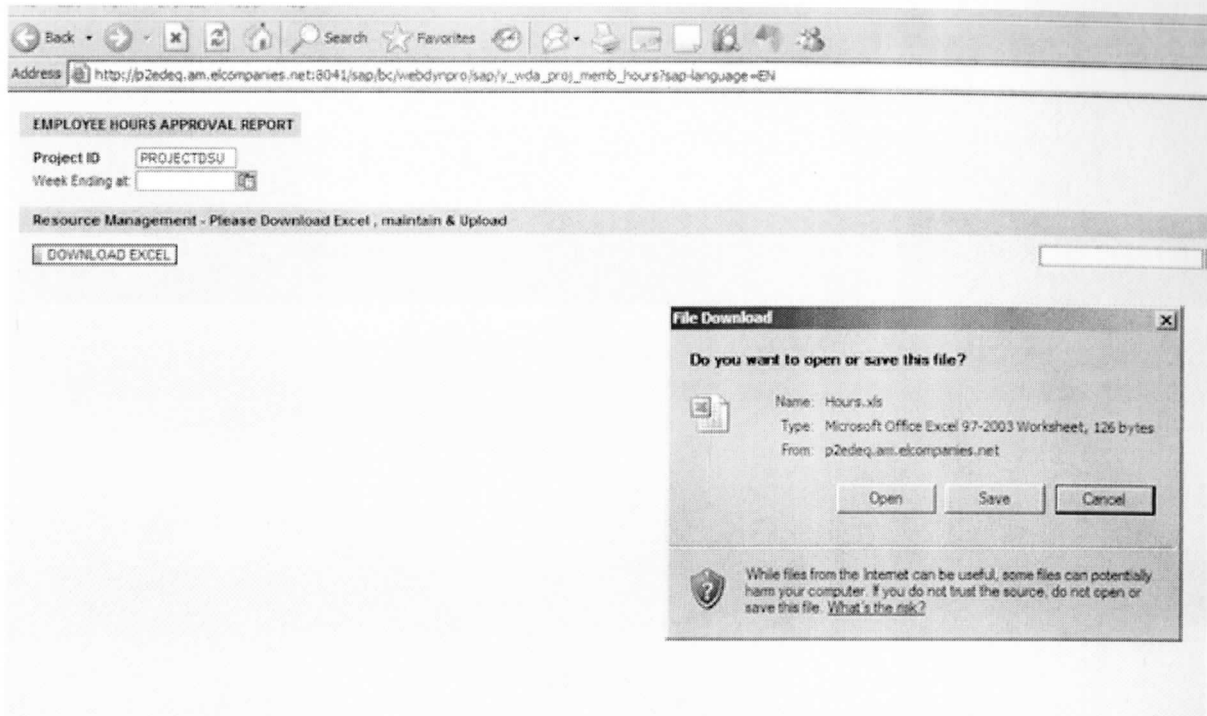


Figure 56. User hits DOWNLOAD EXCEL BUTTON

The user has the option of saving or opening up the excel sheet. The excel sheet generated is shown below.

The screenshot shows the Microsoft Excel interface with the following data in the worksheet:

1	PROJECT ID	WEEK END	EMPLOYEE	LAST NAM	FIRST NAM	HOURS	APPROVED
2	PROJECTDSU	20110424	1875943	SHIKHA	JAIN	75	Y
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							

Figure 57. Generated Excel Sheet

Upload Excel

Now we will show how the user can update the SAP database table using excel sheet.

For that we first check what all values we have in SAP database table.

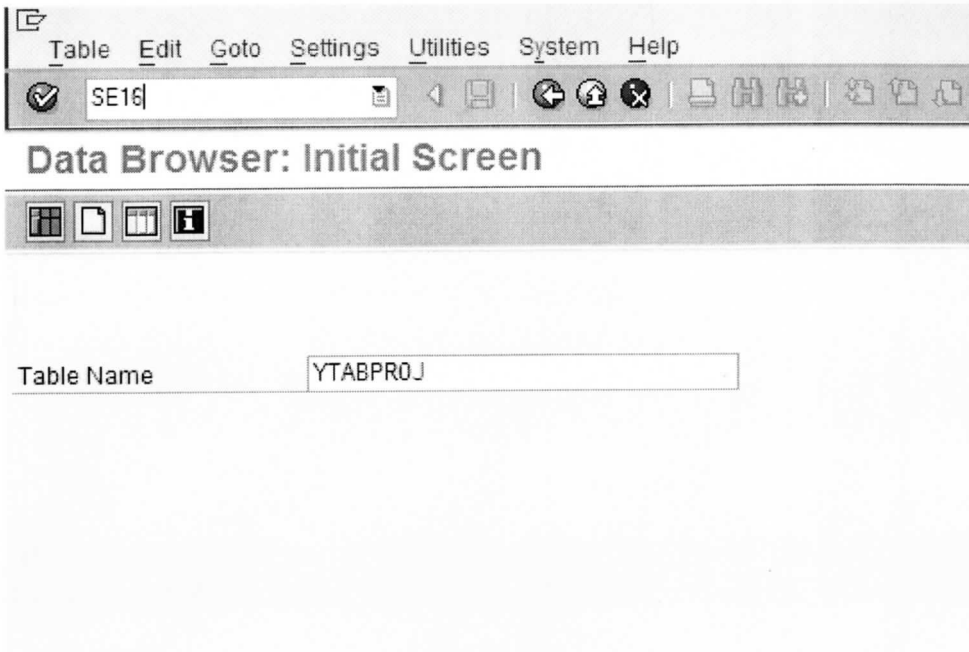


Figure 58. Browsing SAP database table

Figure below shows the values we have in SAP database table

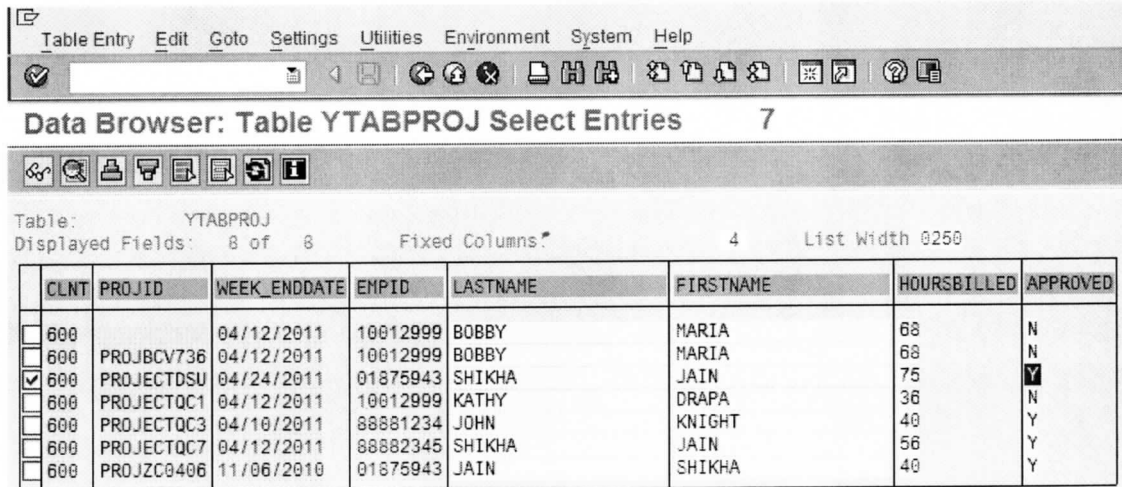


Figure 59. Table YTABPROJ entries

Now if the end user wants to update using excel sheet the APPROVED column of the YTABPROJ to N for PROJECT ID PROJECTDDSU

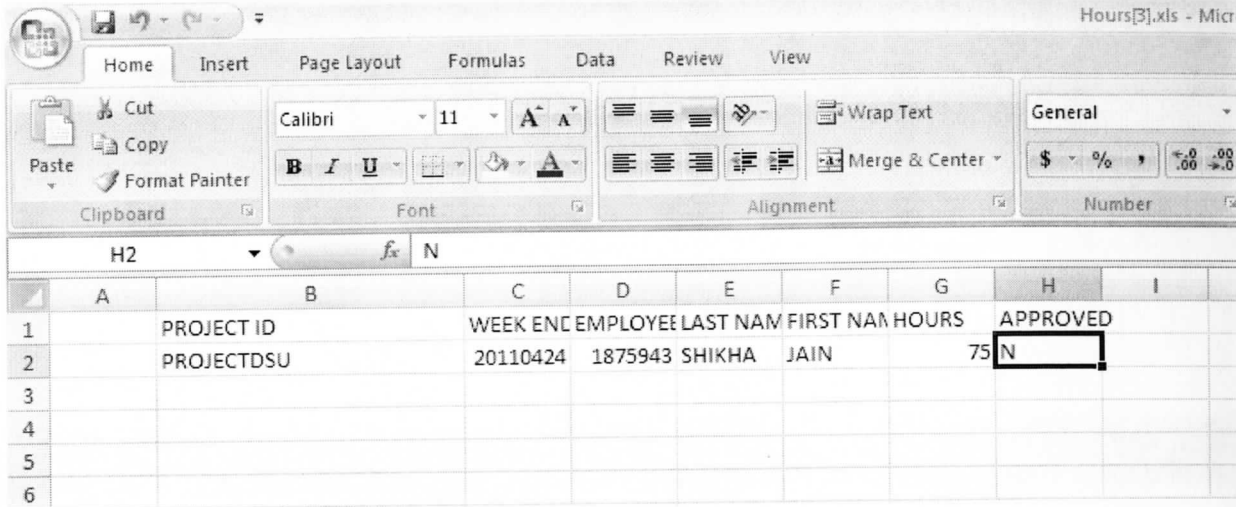


Figure 60. Changing APPROVED column in the excel sheet

Then the user can locate the excel sheet from his system using Browse button of the web service

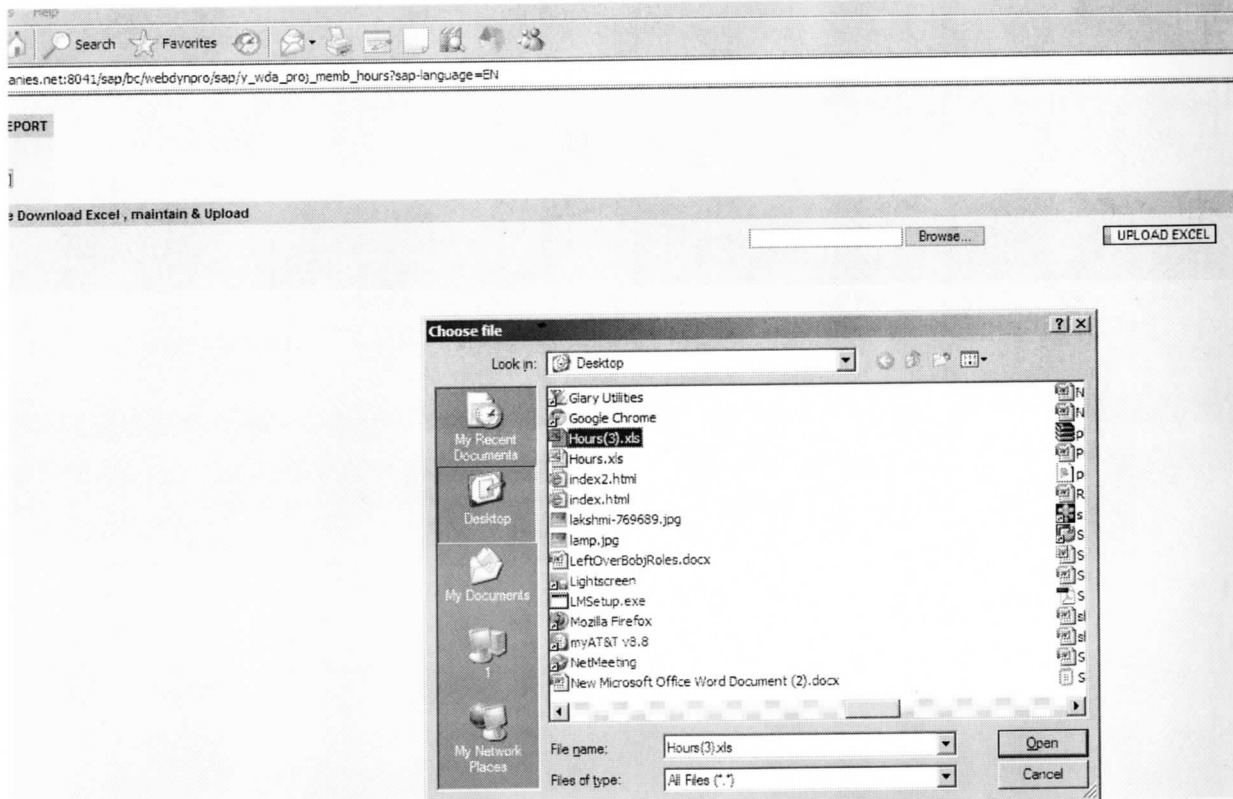


Figure 61. Browsing the excel file to be uploaded.

The user then clicks on UPLOAD EXCEL button to upload the excel file on the SAP database to update table YTABPROJ.

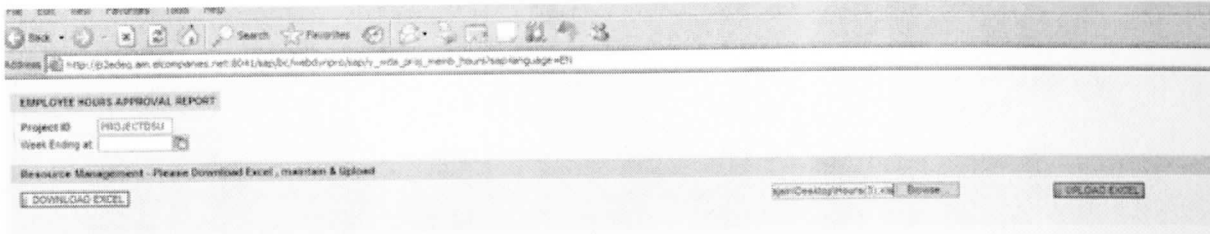


Figure 62. Users clicks UPLOAD EXCEL button

Figure 62. Users clicks UPLOAD EXCEL button

Now if we check the database table YTABPRO, it shows the updated entry for PROJID PROJECTDSU as APPROVED column to be N (was Y previously)

The screenshot shows the SAP Data Browser interface for table YTABPROJ. The table has 8 columns: CLNT, PROJ.ID, WEEK_ENDDATE, EMPID, LASTNAME, FIRSTNAME, HOURS.BILLED, and APPROVED. The entry for PROJID PROJECTDSU now has an APPROVED value of 'N'.

CLNT	PROJ.ID	WEEK_ENDDATE	EMPID	LASTNAME	FIRSTNAME	HOURS.BILLED	APPROVED	
<input type="checkbox"/>	600						N	
<input type="checkbox"/>	600	PROJBCV736	04/12/2011	10012999	BOBBY	MARIA	68	N
<input type="checkbox"/>	600	PROJECTDSU	04/24/2011	1875943	SHIKHA	JAIN	75	N
<input checked="" type="checkbox"/>	600	PROJECTQC1	04/12/2011	10012999	KATHY	DRAPA	36	N
<input type="checkbox"/>	600	PROJECTQC3	04/10/2011	88881234	JOHN	KNIGHT	40	Y
<input type="checkbox"/>	600	PROJECTQC7	04/12/2011	88882345	SHIKHA	JAIN	56	Y
<input type="checkbox"/>	600	PROJZC0406	11/06/2010	01875943	JAIN	SHIKHA	40	Y

Figure 63. Updated database table YTABPROJ

CHAPTER 5

CONCLUSIONS

1. SAP ERP helps to automate a company's business and also helps to enhance the information systems.
2. The HR department can save a lot of time and resources by integrating the SAP database with the Excel Sheet.
3. Excel Sheet provides a simple and easy way to make calculation on the data, make comparisons among different set of values etc. instead of doing it manually.
4. It is easy to send the information in excel sheet to the user who does not have access to the company's database.

There can be many future enhancements to this application like:-

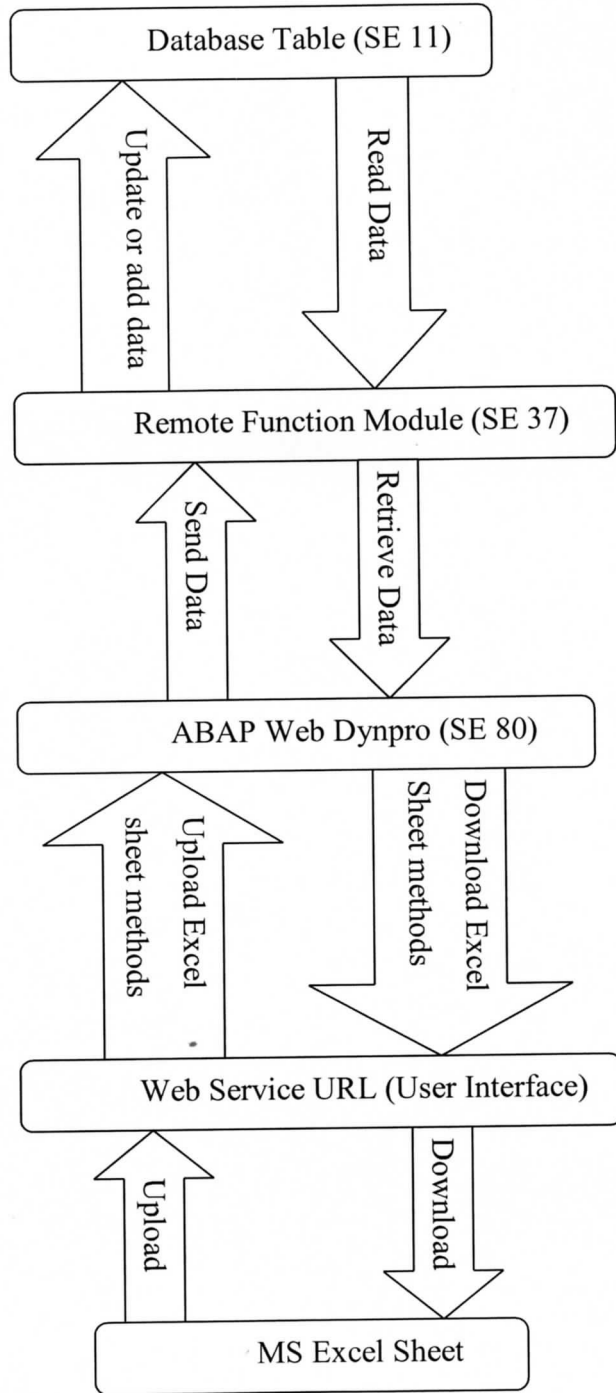
- The user can have the option of directly emailing the excel file generated just by specifying the email addresses instead of manually emailing it to each address
- Same architecture could be used in other set of business requirements like viewing the leave history of an employee, viewing and updating the inventory in excel format etc.

REFERENCES

1. SAP Library. - <http://help.sap.com/>
2. SAP Developers Network - <https://www.sdn.sap.com>
3. <http://www.ibm.com/developerworks/webservices/library/ws-wslover/>
4. http://en.wikipedia.org/wiki/SAP_ERP
5. http://en.wikipedia.org/wiki/Remote_Function_Call
6. http://en.wikipedia.org/wiki/Visual_Basic
7. http://help.sap.com/saphelp_nw04s/helpdata/en/77/3545415ea6f523e10000000a155106/content.htm

APPENDICES

Appendix A: Technical architecture diagram



Appendix B: SAP Code

Function Y RFC_WS_CORP_PROJ_MEM_HOURS

```

FUNCTION Y RFC_WS_CORP_PROJ_MEM_HOURS.
*-----
***Local Interface:
* IMPORTING
*   VALUE(PROJECT_ID) TYPE CHAR10
*   VALUE(WEEK_ENDDATE) TYPE DATUM OPTIONAL
* EXPORTING
*   VALUE(PROJTAB) TYPE YTABTYP_PROJ
*-----

DATA: IT_PROJTAB TYPE STANDARD TABLE OF YTABPROJ.

DATA: IT_PROJTAB_WEEK TYPE STANDARD TABLE OF YTABPROJ.

DATA: WA_PROJTAB TYPE YTABPROJ.

CALL FUNCTION 'Y_FM_PROJ_MEMBERS'
  EXPORTING
    PROJECT_ID = PROJECT_ID
  IMPORTING
    PROJECT_TAB = IT_PROJTAB.

**SELECT * FROM YTABPROJ APPENDING TABLE PROJTAB WHERE PROJID = PROJECT_ID AND
WEEK_ENDDATE = WEEK_ENDDATE.

IF WEEK_ENDDATE IS NOT INITIAL.

  LOOP AT IT_PROJTAB INTO WA_PROJTAB WHERE WEEK_ENDDATE = WEEK_ENDDATE.

```

```

**READ * FROM IT_PROJTAB INTO WA_PROJTAB WITH KEY WEEK_ENDDATE = WEEK_ENDDATE.

        APPEND WA_PROJTAB TO IT_PROJTAB_WEEK.

    ENDLLOOP.

ELSE.
    IT_PROJTAB_WEEK = IT_PROJTAB.
ENDIF.

PROJTAB = IT_PROJTAB_WEEK.

ENDFUNCTION.

```

Download Excel Methods

ONACTIONEXPORT_TO_EXCEL

```
method ONACTIONEXPORT_TO_EXCEL .
```

* Calling the execute method of the rfc (Code of this custom execute method showed in screen below) -

```
wd_this->zsubmit(
).
```

* Below code to convert the context (output of the rfc) to excel -

```

DATA lo_nd_projtab TYPE REF TO if_wd_context_node.

DATA lt_projtab TYPE wd_this->Elements_projtab.

* navigate from <CONTEXT> to <PROJTAB> via lead selection
lo_nd_projtab = wd_context->path_get_node( path =
`Y_RFC_WS_CORP_PROJ_M.CHANGING.PROJTAB` ).

* @TODO handle non existant child
* IF lo_nd_projtab IS INITIAL.
* ENDF.

lo_nd_projtab->get_static_attributes_table( importing table = lt_projtab ).

```

```
DATA LW_PROJTAB TYPE ytabproj.
data str type string.
data xstr type xstring.
```

```
concatenate str
      'PROJECT ID'
      'WEEK ENDING DATE'
      'EMPLOYEE ID'
      'LAST NAME'
      'FIRST NAME'
      'HOURS'
      'APPROVED'
      cl_abap_char_utilities=>newline into str
      separated by cl_abap_char_utilities=>horizontal_tab.
```

```
Loop at lt_projtab into lw_projtab.
```

```
concatenate str
      LW_PROJTAB-PROJID
      LW_PROJTAB-WEEK_ENDDATE
      LW_PROJTAB-EMPID
      LW_PROJTAB-LASTNAME
      LW_PROJTAB-FIRSTNAME
      LW_PROJTAB-HOURSBILLED
      LW_PROJTAB-APPROVED
      cl_abap_char_utilities=>newline into str
      separated by cl_abap_char_utilities=>horizontal_tab.

endloop.
```

```
CALL FUNCTION 'SCMS_STRING_TO_XSTRING'
  EXPORTING
    text          = str
  * MIMETYPE     = ''
  * ENCODING     =
  IMPORTING
    BUFFER       = xstr
  EXCEPTIONS
    FAILED       = 1.
```

```
CALL METHOD cl_wd_runtime_services=>attach_file_to_response
  EXPORTING
    i_filename    = 'Hours.xls'
    i_content     = xstr
    i_mime_type   = 'EXCEL'
    i_in_new_window = ABAP_FALSE
    i_inplace     = ABAP_FALSE.
```

```
endmethod.
```

ZSUBMIT

```
method ZSUBMIT .
```

```
*** CLEAR THE TABLE FIRST
```

```
DATA lo_nd_projtab TYPE REF TO if_wd_context_node.
```

```
DATA lt_projtab TYPE wd_this->Elements_projtab.
```

```
DATA LT_CLEAR_PROJTAB TYPE wd_this->Elements_projtab.
```

```
* navigate from <CONTEXT> to <PROJTAB> via lead selection
  lo_nd_projtab = wd_context->path_get_node( path =
`Y_RFC_WS_CORP_PROJ_M.CHANGING.PROJTAB` ).
```

```
* @TODO handle non existant child
```

```
* IF lo_nd_projtab IS INITIAL.
```

```
* ENDIF.
```

```
** @TODO compute values
```

```
** e.g. call a model function
```

```
*
```

```
** lo_nd_projtab->bind_table( new_items = lt_projtab set_initial_elements =
abap_true ).
```

```
lo_nd_projtab->bind_table( LT_CLEAR_PROJTAB ).
```

```
*** EXECUTE THE RFC
```

```
DATA lo_COMPONENTCONTROLLER TYPE REF TO IG_COMPONENTCONTROLLER .
```

```
lo_COMPONENTCONTROLLER = wd_this->get_componentcontroller_ctr( ).
```

```
lo_componentcontroller->execute_y_rfc_ws_corp_proj_mem(
 ).
```

```
endmethod.
```

Upload excel sheet methods

ONACTIONUPLOAD_EXCEL

```
method ONACTIONUPLOAD_EXCEL .
```

```
TYPES :
```

```
  BEGIN OF str_itab,
    CLNT(3) TYPE C,
    PROJID(10) TYPE C,
    WEEK_ENDDATE(8) TYPE C,
    EMPID(8) TYPE c,
    LASTNAME(10) TYPE c,
    FIRSTNAME(10) TYPE C,
    HOURS_BILLED(2) TYPE C,
    APPROVED(1) TYPE C,
  END OF str_itab.
```

```
DATA : t_table1 TYPE STANDARD TABLE OF str_itab,
      i_data TYPE STANDARD TABLE OF string,
      lo_nd_data TYPE REF TO if_wd_context_node,
      lo_el_data TYPE REF TO if_wd_context_element,
      l_string TYPE string,
      fs_table TYPE str_itab,
      l_xstring TYPE xstring,
      fields TYPE string_table,
      lv_field TYPE string.
```

```
** DATA : t_table TYPE if_main=>elements_PROJTAB,
**       data_table TYPE if_main=>elements_PROJTAB.
```

```
DATA lo_nd_projtab TYPE REF TO if_wd_context_node.
```

```
DATA lo_el_projtab TYPE REF TO if_wd_context_node.
```

```
DATA T_TABLE TYPE wd_this->Elements_projtab.
DATA DATA_TABLE TYPE wd_this->Elements_projtab.
DATA lt_projtabprev TYPE wd_this->Elements_projtab.
```

```
***** navigate from <CONTEXT> to <PROJTAB> via lead selection
      lo_nd_projtab = wd_context->path_get_node( path =
`Y_RFC_WS_CORP_PROJ_M.CHANGING.PROJTAB` ).
```

```
****
```

```
***** @TODO handle non existant child
```

```
***** IF lo_nd_projtab IS INITIAL.
```

```
***** ENDIF.
```

```
****
```

```
***** cLEAR THE CURRENT TABLE.
```

```
****
```

```
**** lo_nd_projtab->get_static_attributes_table( importing table = lt_projtabprev ).
```

```
****
```

```
****DATA: WA_TEMP TYPE YTABPROJ.
```

```
****
```

```

**** LOOP AT lt_projtabprev INTO wa_temp.
***   lo_nd_projtab->remove_element( EXPORTING element = lo_nd_projtab ).
**** ENDLOOP.

* get single attribute
wd_context->get_attribute(
  EXPORTING
    name = `DATASOURCE`
  IMPORTING
    value = l_xstring ).

CALL FUNCTION 'HR_KR_XSTRING_TO_STRING'
  EXPORTING
    in_xstring = l_xstring
  IMPORTING
    out_string = l_string.

SPLIT l_string AT cl_abap_char_utilities=>newline INTO TABLE i_data.

* Bind With table Element.
LOOP AT i_data INTO l_string.
  SPLIT l_string AT cl_abap_char_utilities=>horizontal_tab INTO TABLE fields.

*   READ TABLE fields INTO lv_field INDEX 1.
*   fs_table-CLNT = lv_field.
  READ TABLE fields INTO lv_field INDEX 1.

  IF LV_FIELD NE 'PROJECT ID'.

    fs_table-PROJID = lv_field.
    READ TABLE fields INTO lv_field INDEX 2.
    fs_table-WEEK_ENDDATE = lv_field.
    READ TABLE fields INTO lv_field INDEX 3.
    fs_table-EMPID = lv_field.
    READ TABLE fields INTO lv_field INDEX 4.
    fs_table-LASTNAME = lv_field.
    READ TABLE fields INTO lv_field INDEX 5.
    fs_table-FIRSTNAME = lv_field.
    READ TABLE fields INTO lv_field INDEX 6.
    fs_table-HOURSBILLED = lv_field.
    READ TABLE fields INTO lv_field INDEX 7.
    fs_table-APPROVED = lv_field.

    APPEND fs_table TO t_table1.

  ENDIF.

ENDLOOP.

**lo_nd_data = wd_context->get_child_node( 'PROJTAB' ).
**lo_nd_data->bind_table( T_TABLE1 ).

```

```

CLEAR DATA_TABLE.
*
lo_nd_projtab->bind_table( DATA_TABLE ).

lo_nd_projtab->bind_table( new_items = T_TABLE1 set_initial_elements = abap_false
).

**=====

*** DATA lo_nd_projtab TYPE REF TO if_wd_context_node.
***
*** DATA lt_projtab TYPE wd_this->Elements_projtab.
***
**** navigate from <CONTEXT> to <PROJTAB> via lead selection
*** lo_nd_projtab = wd_context->path_get_node( path =
`Y RFC_WS_CORP_PROJ_M.CHANGING.PROJTAB` ).
***
**** @TODO handle non existant child
**** IF lo_nd_projtab IS INITIAL.
**** ENDF.
***
*** lo_nd_projtab->get_static_attributes_table( importing table = lt_projtab ).
***
***

**=====

*** DATA lo_nd_projtab TYPE REF TO if_wd_context_node.
***
*** DATA lt_projtab TYPE wd_this->Elements_projtab.
***
**** navigate from <CONTEXT> to <PROJTAB> via lead selection
*** lo_nd_projtab = wd_context->path_get_node( path =
`Y RFC_WS_CORP_PROJ_M.CHANGING.PROJTAB` ).
***
**** @TODO handle non existant child
**** IF lo_nd_projtab IS INITIAL.
**** ENDF.
***
**** @TODO compute values
**** e.g. call a model function
****
*** lo_nd_projtab->bind_table( new_items = lt_projtab set_initial_elements =
abap_true ).
***

wd_this->save_db( ).

endmethod.

```

SAVE_DB

method SAVE_DB .

DATA lo_nd_projtab TYPE REF TO if_wd_context_node.

DATA lt_projtab TYPE wd_this->Elements_projtab.

* navigate from <CONTEXT> to <PROJTAB> via lead selection
lo_nd_projtab = wd_context->path_get_node(path = `Y_RFC_WS_CORP_PROJ_M.CHANGING.PROJTAB`
).

* @TODO handle non existant child
* IF lo_nd_projtab IS INITIAL.
* ENDIF.

lo_nd_projtab->get_static_attributes_table(importing table = lt_projtab).

DATA: WT_TABLE1 TYPE YTABPROJ.

LOOP AT lt_projtab INTO WT_TABLE1.

MODIFY YTABPROJ FROM WT_TABLE1.

ENDLOOP.

Endmethod

