Developing JSF-Portlets with regard to SAP integration using the IBM Rational Application Developer V7

Version 1.0

13th July 2007

Table of contents

1 Preface

1.1 Preface and Scope

This document is intended to be a technical description and design document for use by technical people.

1.2 Disclaimer

This document is subject to change without notification and will not comprehensively cover the issues encountered in any customer situation. It should only be used in conjunction with the product literature accompanying the J2EE products from IBM and SAP.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS.

1.3 Scope

This document is a guide containing a sample that shows how to develop a JSF portlet with IBM Rational Application Developer Version 7. The portlet will be able to access an SAP system and get its containing data using service data objects (SDO). Furthermore the received data will be visualized within the Portlet using the Java Server Faces (JSF) technology.
The guide is illustrated by screen captures so that you can reproduce it step by step.

2 System prerequisites

2.1 IBM Rational Application Developer V7

The IBM Rational Application Developer V7 must be properly installed in order to create SAP content-based JSF-Portlets.
You also need to install the SAP JAVA Connector (SAP - JCo) to connect to a SAP back-end system from Rational Application Developer. You should use the latest version of the SAP-JCo.

2.2 WebSphere Portal and Application Server

To use the SAP-Mediator functions of the SDO, you must have properly installed WebSphere Portal Server V6, which also assumes a proper installation of the WebSphere Application Server. Moreover, note that the WebSphere Application Server is also SAP-JCo enabled. You should use at least version 2.1.1 of the SAP-JCo.
This sample uses the integrated WebSphere test-environment of the IBM Rational Application Developer.

2.3 SAP/R3

You must be able to access a SAP/R3 system and have the appropriated rights to access the BAPI you want to call.

2.4 SAP JCo

The SAP Java Connector is a free Java library to connect to the different SAP back end systems using Java code to run BAPI calls.
It can be downloaded at http://service.sap.com in the current version 2.1.8 .

3 Introduction to JSF and SDO

Java Server Faces (JSF) is a framework that simplifies building user interfaces for Web applications. The combination of the JSF technology and the tools provided by Rational Application Developer affords developers of differing skill levels the ability to achieve the promises of rapid Web development.
This section provides an overview examining three aspects of JSF:

- JSF features and benefits
- JSF application architecture
- IBM Rational Application Developer support for JSF

3.1 JSF features and benefits

3.1.1 JSF standards-based Web application framework

Java Server Faces technology is the result of the Java Community process JSR-127 and evolved from Struts. The original author of Struts (Craig McClanahan) is a participating author of the JSF specification. JSF addresses more of the Model-View-Controller pattern than Struts, in that it more strongly addresses the View or presentation layer though UI components, and addresses the Model through Managed Beans. Although JSF is an emerging technology and will likely become a dominant standard, like Struts is today. JSF is targeted at Web developers with little knowledge of Java by eliminating much of the hand coding involved in integrating Web applications with back-end systems.

3.1.2 Event driven architecture

JSF provides server-side rich UI components that respond to client events.

3.1.3 User interface development

UI components are decoupled from its rendering. This allows for other technologies such as WML to be used (for example, mobile devices).
JSF allows direct binding of user interface (UI) components to model data. Developers can use extensive libraries of ready-made UI components that provide both basic and advanced Web functionality.

3.1.4 Session and object management

JSF manages designated model data objects by handling their initialization, persistence over the request cycle and cleanup.

3.1.5 Validation and error feedback

JSF allows direct binding of reusable validators to UI components. The framework also provides a queue mechanism to simplify error and message feedback to the application user. These messages can be associated with specific UI components.

3.1.6 Internationalization

JSF provides tools for internationalizing Web applications, supporting number, currency, time, and date formatting, and externalizing of UI strings.

3.2 JSF application architecture

The JSF application architecture can be easily extended in a variety of ways to suit the requirements of your particular application. You can develop custom components, renderers, validators, and other JSF objects and register them with the JSF runtime.

- JSF page JSPs are built from JSF components, where each component is represented by a server-side class.
- Faces Servlet - One Servlet (FacesServlet) controls the execution flow.
- Configuration file - An XML file (faces-config.xml) that contains the navigation rules between the JSPs, validators, and managed beans.
- Tag libraries - The JSF components are implemented in tag libraries.
- Validators - Java classes to validate the content of JSF components, for example, to validate user input.
- Managed beans - JavaBeans defined in the configuration file to hold the data from JSF components. Managed beans represent the data model and are passed between business logic and user interface. JSF moves the data between managed beans and user interface components.
- Events - Java code that is run in the server for events (for example, a push button). Event handling is used to pass managed beans to business logic.

3.3 IBM Rational Application Developer support for JSF

IBM Rational Application Developer V7 includes several features for building highly functional Web applications. It includes full support for less skilled developers create drag-and-drop Web applications.
Rational Application Developer includes the following support and tools for JSF Web application development:

- Visual page layout of JSF components using a page designer
- Built-in component property editor
- Built-in tools to simplify and automate event handling
- Built-in tools simplify page navigation
- Page navigation is defined declaratively
- Automatic code generation for data validation, formatting and create-read-update-delete functions for data access.
- Relational database support
- EJB support
- Web services support
- Data abstraction objects for easy data connectivity (SDO)
- Data objects can be bound easily to user interface components

3.4 Service data objects (SDO)

Service data objects are designed to simplify and unify the way in which applications handle data. Using SDO, application programmers can uniformly access and manipulate data from heterogeneous data sources, including relational databases, XML data sources, Web services, and enterprise information systems.

SDO is based on the concept of disconnected data graphs. A data graph is a collection of tree-structured or graph-structured data objects. Under the disconnected data graphs architecture, a client retrieves a data graph from a data source, mutates the data graph, and can then apply the data graph changes back to the data source.

The task of connecting applications to data sources is performed by data mediator services. Client applications query a data mediator service and get a data graph in response. Client applications send an updated data graph to a data mediator service to have the updates applied to the original data source. This architecture allows applications to deal principally with data graphs and data objects.

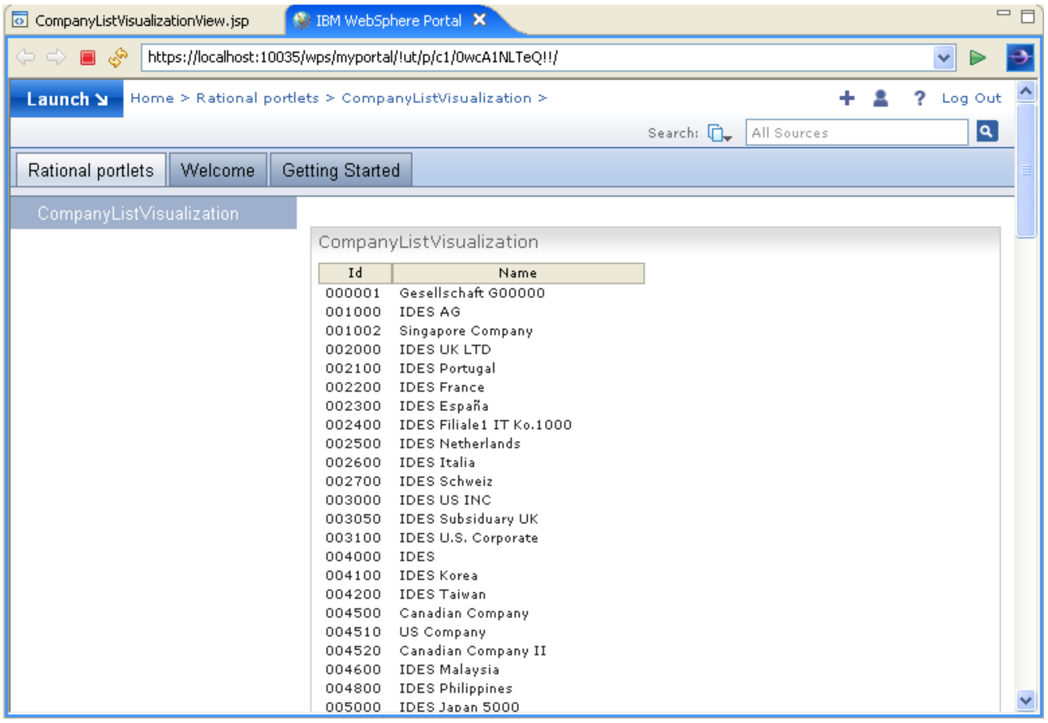
SDO enables both a static (or strongly typed) programming model and a dynamic (or loosely typed) programming model. This enables a simple programming model without sacrificing the dynamic model needed by tools and frameworks.

SDO also provides a metadata API, which allows applications, tools, and frameworks to introspect the data model for a data graph. The SDO metadata API unifies data-source-specific metadata APIs to enable applications to handle data from heterogeneous data sources in a uniform way.

4 Creating the sample with Rational Application Developer

4.1 *Description*

The presented sample is able to show a page with the companies of an SAP system in a simple table.


CompanyListVisualizationView.jsp
IBM WebSphere Portal
https://localhost:10035/wps/myportal/!ut/p/c1/0wcA1NLTeQ!!/
Launch
Home > Rational portlets > CompanyListVisualization >
Log Out
Search:
All Sources
Rational portlets
Welcome
Getting Started
CompanyListVisualization
CompanyListVisualization

Id	Name
000001	Gesellschaft G00000
001000	IDES AG
001002	Singapore Company
002000	IDES UK LTD
002100	IDES Portugal
002200	IDES France
002300	IDES España
002400	IDES Filiale1 IT Ko.1000
002500	IDES Netherlands
002600	IDES Italia
002700	IDES Schweiz
003000	IDES US INC
003050	IDES Subsiduary UK
003100	IDES U.S. Corporate
004000	IDES
004100	IDES Korea
004200	IDES Taiwan
004500	Canadian Company
004510	US Company
004520	Canadian Company II
004600	IDES Malaysia
004800	IDES Philippines
005000	IDES Japan 5000

Figure 1: Overview of the final sample page

In the following pages, you will see how to develop the project and its needed components. First you will see how to create a new portlet project with Rational Application Developer, then how to initialize the Portal Server where the final Project will be deployed. Finally you will see how to create the SDO to integrate the data of the SAP system with JSF.

4.2 *Creating the Portlet Project*

After Rational Application Developer has been started, use *File → New → Project* and select the Portlet Project Wizard as shown in the following figure and click Next.

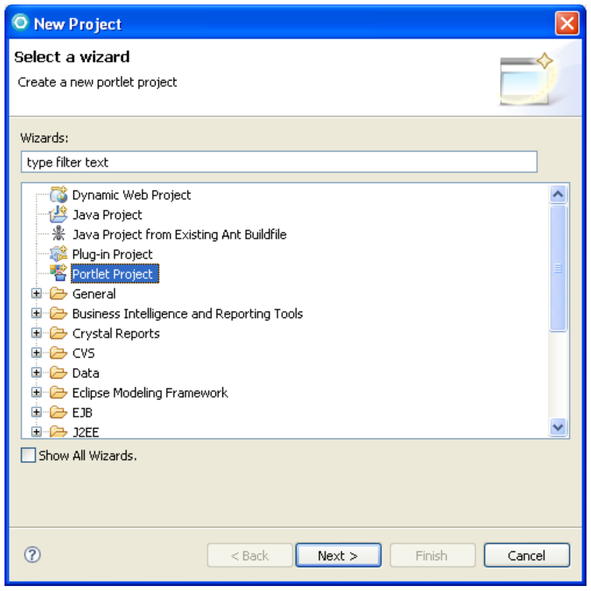
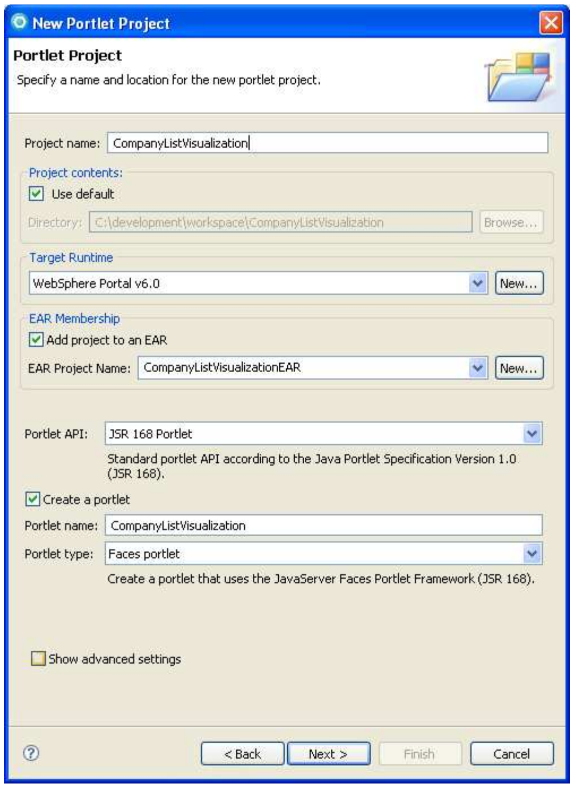


Figure 2: Select the Portlet Project Wizard

Now you have to specify the name and basic settings of the portlet project. Make sure that the selected target runtime is the WebSphere Portal V6.0 server and that the Portlet API is the JSF 168 Portlet. Furthermore the type of the automatically created Portlet must be a Faces Portlet. Proceed by clicking on Next.



Portlet Project

Specify a name and location for the new portlet project.

Project name: CompanyListVisualization

Project contents:

☑ Use default

Directory: C:\development\workspace\CompanyListVisualization

Target Runtime

WebSphere Portal v6.0

EAR Membership

☑ Add project to an EAR

EAR Project Name: CompanyListVisualizationEAR

Portlet API: JSR 168 Portlet

Standard portlet API according to the Java Portlet Specification Version 1.0 (JSR 168).

☑ Create a portlet

Portlet name: CompanyListVisualization

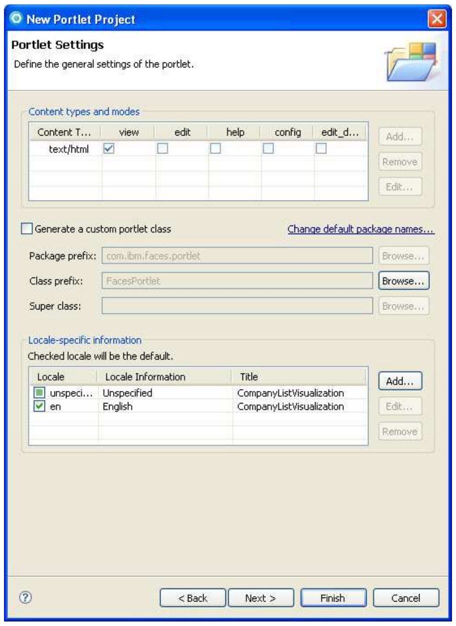
Portlet type: Faces portlet

Create a portlet that uses the JavaServer Faces Portlet Framework (JSR 168).

☐ Show advanced settings

Figure 3: Specification of the name and the settings of the Portlet Project

The next window provides general settings like content types and locale-specific information. There is no need to make any changes for this sample, so continue with Next.



New Portlet Project

Portlet Settings

Define the general settings of the portlet.

Content types and modes

Content T...	view	edit	help	config	edit_d...
text/html	✓	☐	☐	☐	☐

☐ Generate a custom portlet class

Change default package names...

Package prefix: com.ibm.faces.portlet

Class prefix: FacesPortlet

Super class:

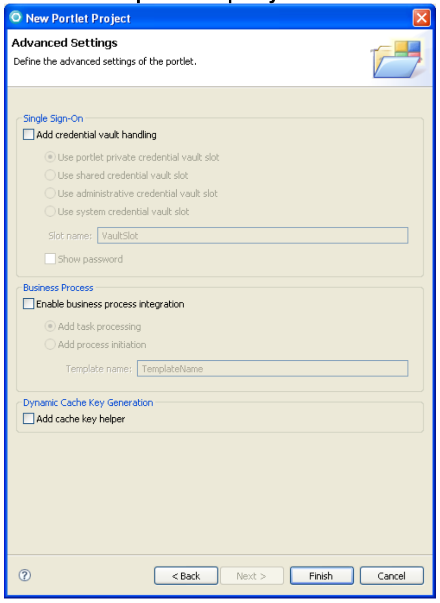
Locale-specific information

Checked locale will be the default.

Locale	Locale Information	Title
☐ unspeci...	Unspecified	CompanyListVisualization
✓ en	English	CompanyListVisualization

Figure 4: General settings of the Portlet Project

The next screen lets you configure the advanced settings, but that is not needed in this case. Finalize the portlet project creation by clicking Finish.



New Portlet Project

Advanced Settings

Define the advanced settings of the portlet.

Single Sign-On

☐ Add credential vault handling

● Use portlet private credential vault slot

○ Use shared credential vault slot

○ Use administrative credential vault slot

○ Use system credential vault slot

Slot name: VaultSlot

☐ Show password

Business Process

☐ Enable business process integration

● Add task processing

○ Add process initiation

Template name: TemplateName

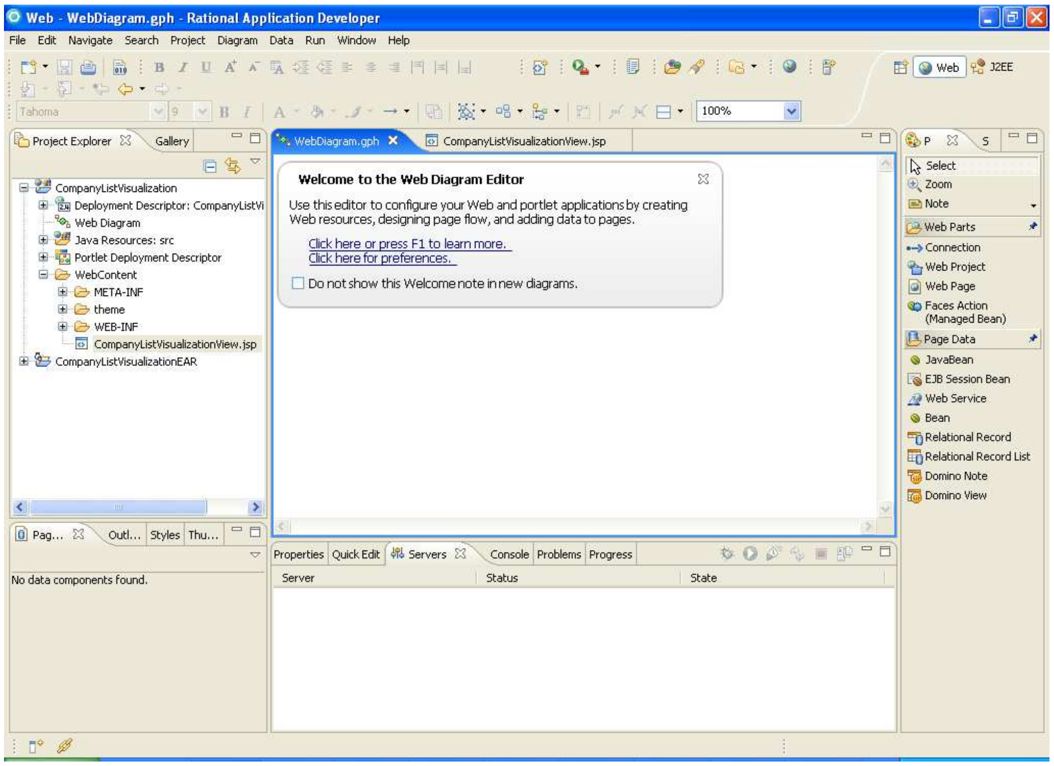
Dynamic Cache Key Generation

☐ Add cache key helper

Figure 5: Advanced settings of the Portlet Project

The Portlet Project is now located in the *Project Explorer* on the left side of the IBM Rational Application Developer desktop. The central window shows the Web diagram editor which is not needed any more and can be closed.

Make sure that the Web perspective is selected (menu: *Window* → *Open Perspective*) to be able to follow the further instructions.


Web - WebDiagram.gph - Rational Application Developer
File Edit Navigate Search Project Diagram Data Run Window Help
Web J2EE
Tahoma 9 100%
Project Explorer Gallery
CompanyListVisualization
Deployment Descriptor: CompanyListVi
Web Diagram
Java Resources: src
Portlet Deployment Descriptor
WebContent
META-INF
theme
WEB-INF
CompanyListVisualizationView.jsp
CompanyListVisualizationEAR
WebDiagram.gph
CompanyListVisualizationView.jsp
Pag… Outl… Styles Thu…
Properties Quick Edit Servers Console Problems Progress
Server Status State
P S
Select
Zoom
Note
Web Parts
Connection
Web Project
Web Page
Faces Action (Managed Bean)
Page Data
JavaBean
EJB Session Bean
Web Service
Bean
Relational Record
Relational Record List
Domino Note
Domino View

Welcome to the Web Diagram Editor

Use this editor to configure your Web and portlet applications by creating Web resources, designing page flow, and adding data to pages.

Click here or press F1 to learn more.
Click here for preferences.

☐ Do not show this Welcome note in new diagrams.

No data components found.

Figure 6: IBM Rational Application Developer desktop with Web perspective

4.3 Creating the Portal Server

To be able to run the created portlet project, you need to create a Portal Server where the Portlet can be deployed. Change to the server tab in the middle bottom panel. Right-click, and select New, then Server.

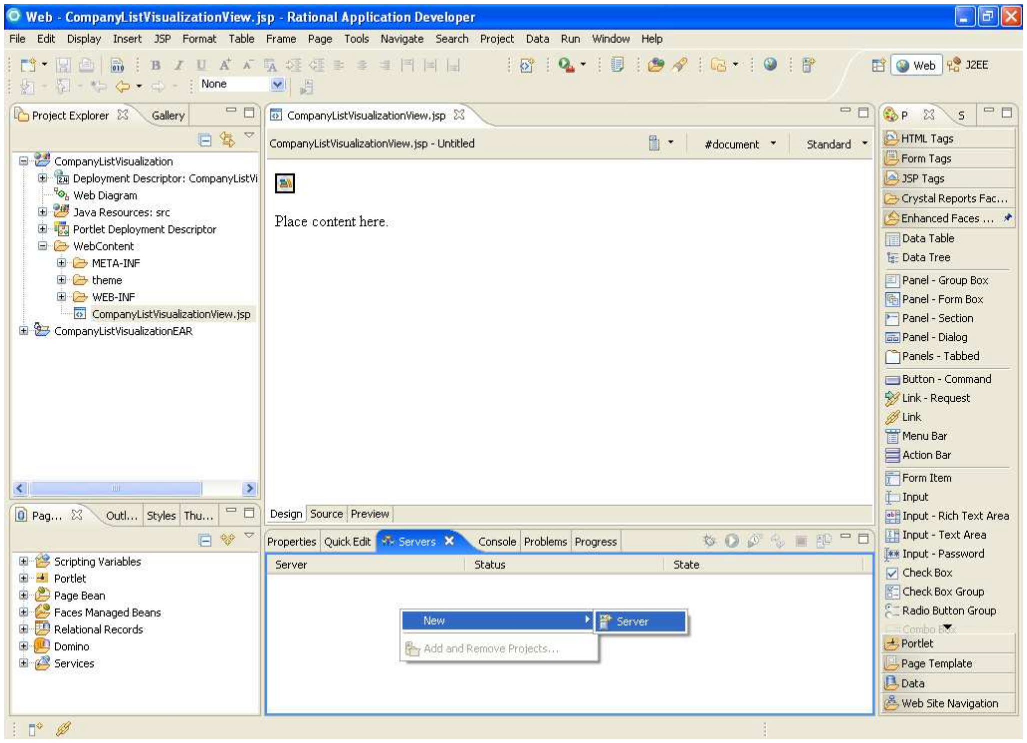
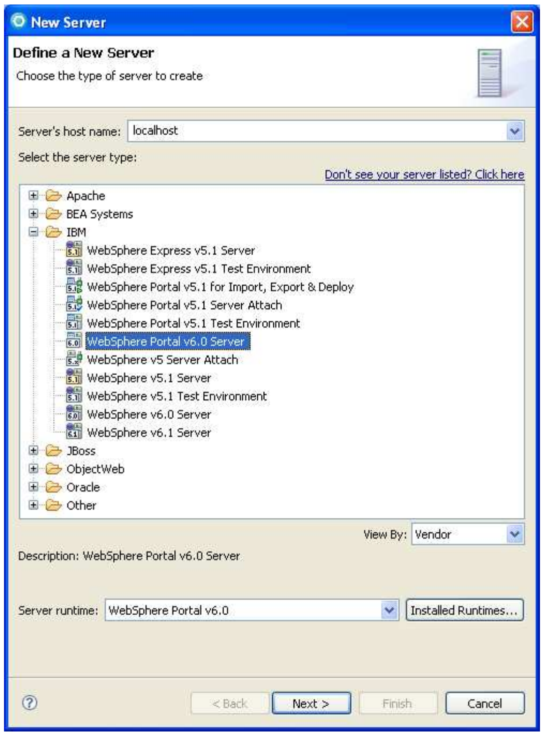

Web - CompanyListVisualizationView.jsp - Rational Application Developer
Project Explorer
CompanyListVisualization
Servers
Server
Status
State
New
Server
Add and Remove Projects...

Figure 7: Create a new Portal Server

For this sample, use the WebSphere Portal Server V6.0. Specify the host name and type of the portal server and click on Next.


New Server

Define a New Server

Choose the type of server to create

Server's host name: localhost

Select the server type:

Don't see your server listed? Click here

- Apache
- BEA Systems
- IBM
  - WebSphere Express v5.1 Server
  - WebSphere Express v5.1 Test Environment
  - WebSphere Portal v5.1 for Import, Export & Deploy
  - WebSphere Portal v5.1 Server Attach
  - WebSphere Portal v5.1 Test Environment
  - WebSphere Portal v6.0 Server
  - WebSphere v5 Server Attach
  - WebSphere v5.1 Server
  - WebSphere v5.1 Test Environment
  - WebSphere v6.0 Server
  - WebSphere v6.1 Server
- JBoss
- ObjectWeb
- Oracle
- Other

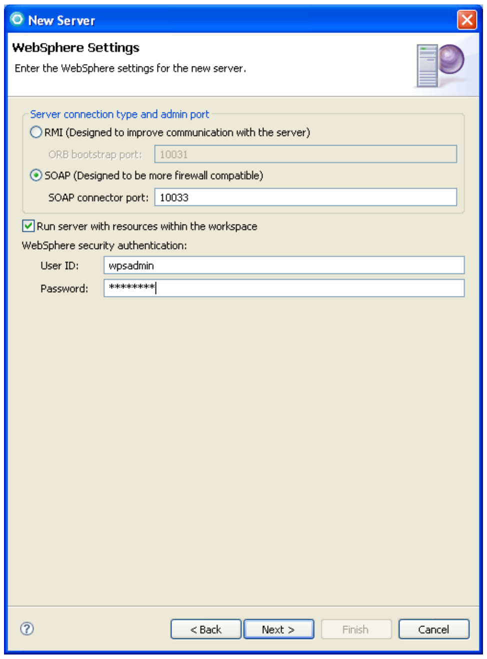
View By: Vendor

Description: WebSphere Portal v6.0 Server

Server runtime: WebSphere Portal v6.0

Figure 8: Specify the Portal Server

Afterwards you can change the server connection type and port but you can also keep the default values in this case. Specify user ID and password for the server and click Next.



New Server

WebSphere Settings

Enter the WebSphere settings for the new server.

Server connection type and admin port

RMI (Designed to improve communication with the server)

ORB bootstrap port: 10031

[✓ SOAP (Designed to be more firewall compatible)]

SOAP connector port: 10033

[✓] Run server with resources within the workspace

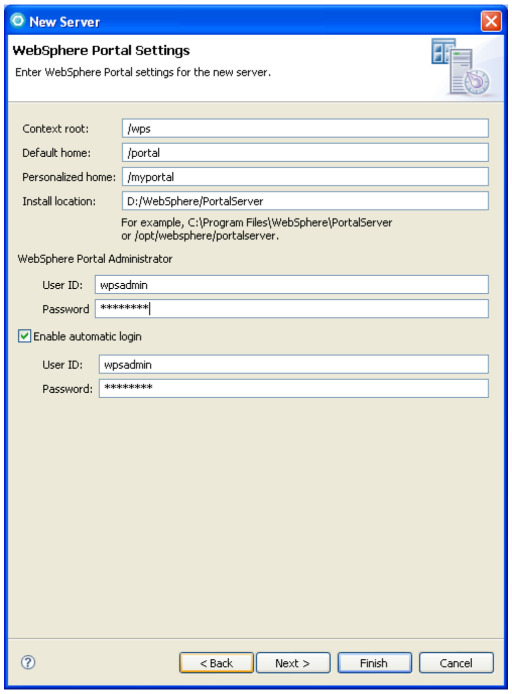
WebSphere security authentication:

User ID: wpsadmin

Password: ********

Figure 9: Websphere settings

Specify the settings of the portal and set again the user ID and password so that you are able to start and stop the Portal Server directly from IBM Rational Application Developer. Click Next.



New Server

WebSphere Portal Settings

Enter WebSphere Portal settings for the new server.

Context root: /wps

Default home: /portal

Personalized home: /myportal

Install location: D:/WebSphere/PortalServer

For example, C:\Program Files\WebSphere\PortalServer or /opt/websphere/portalserver.

WebSphere Portal Administrator

User ID: wpsadmin

Password ********

[✓] Enable automatic login

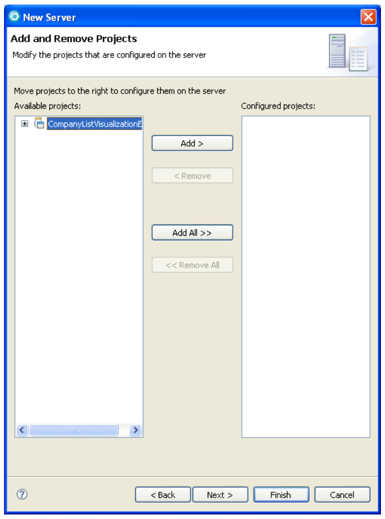
User ID: wpsadmin

Password: ********

Figure 10: Websphere Portal settings

The settings in the next panel do not need to be changed. Click Next.

Now you can select the previously created portlet project and add it to the server where it will be deployed. Click Finish to complete the creation of the portal server.



New Server

Add and Remove Projects

Modify the projects that are configured on the server

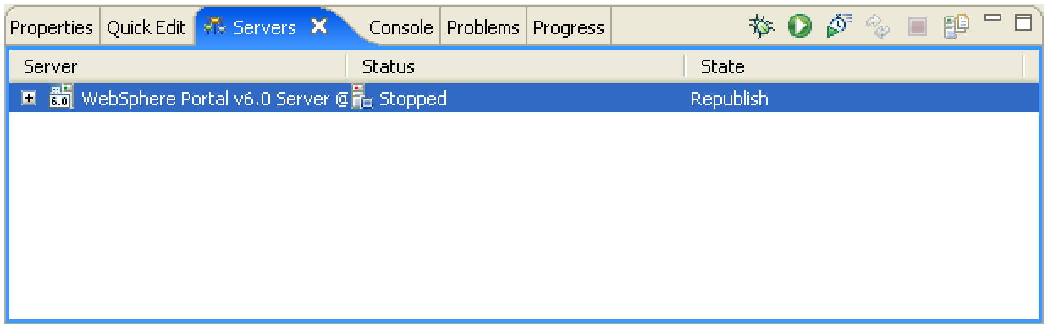
Move projects to the right to configure them on the server

Available projects: CompanyListVisualizationE

Configured projects:

Figure 11: Add Portlet Project to the server

The new server is then added to the server tab.



Server	Status	State
WebSphere Portal v6.0 Server @	Stopped	Republish

Figure 12: Server tab with the created server

4.4 Creating the JSF Portlet

The file *CompanyListVisualizationView.jsp* was automatically created from the portlet project wizard. This is the JSF page that should finally show the table of companies of the SAP system.

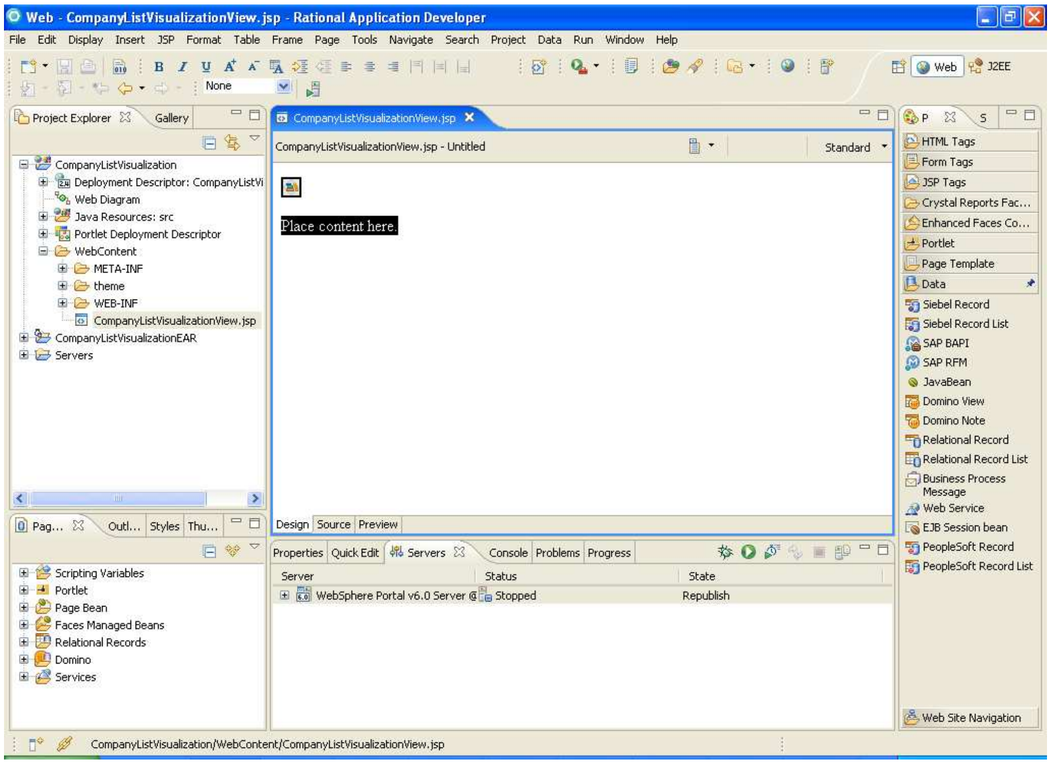
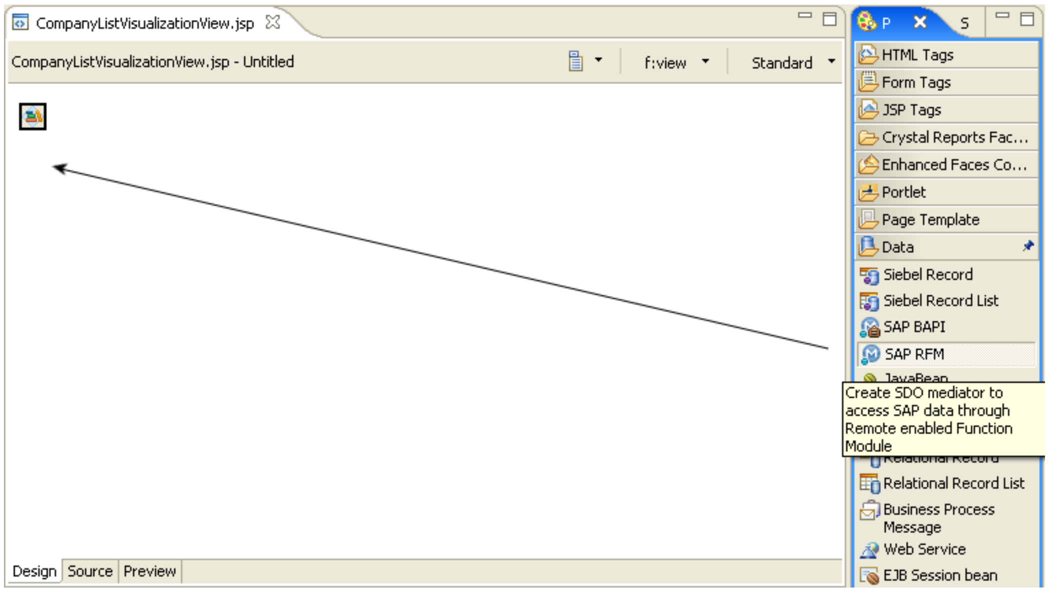


Figure 13: View of the IBM Rational Application Developer desktop after creating the portlet project and the portal server

4.4.1 Adding the SAP SDO

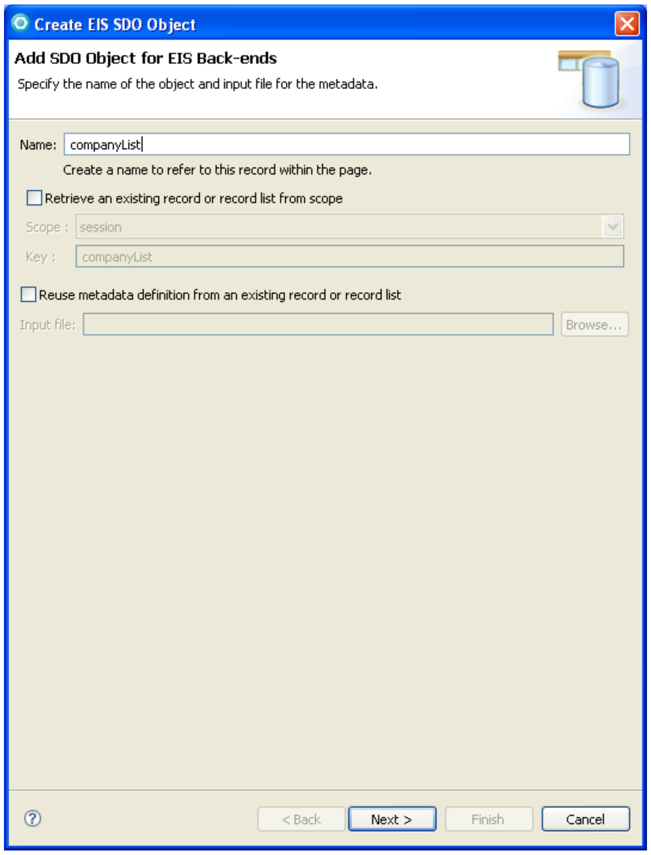
First of all delete the default page view *Place content here*. As you are going to display data of the SAP System, you need an SDO that is able to access the SAP server and provide its content. Select the *Data* menu from the palette on the right side of the desktop. Now drag the entry *SAP RFM* in the design view of the CompanyListDetail.jsp file and drop it there.



Create SDO mediator to access SAP data through Remote enabled Function Module

Figure 14: Adding the SAP SDO to the Portlet

After releasing the mouse button the SDO object wizard appears where you have to enter the name of the SDO. Click Next.


Create EIS SDO Object

Add SDO Object for EIS Back-ends

Specify the name of the object and input file for the metadata.

Name: companyList

Create a name to refer to this record within the page.

☐ Retrieve an existing record or record list from scope

Scope : session

Key : companyList

☐ Reuse metadata definition from an existing record or record list

Input file:

Figure 15: SDO Object Wizard

In the next panel, define the parameters for the SAP server connection. Make sure that all fields have values filled in; otherwise it could lead to a runtime exception.

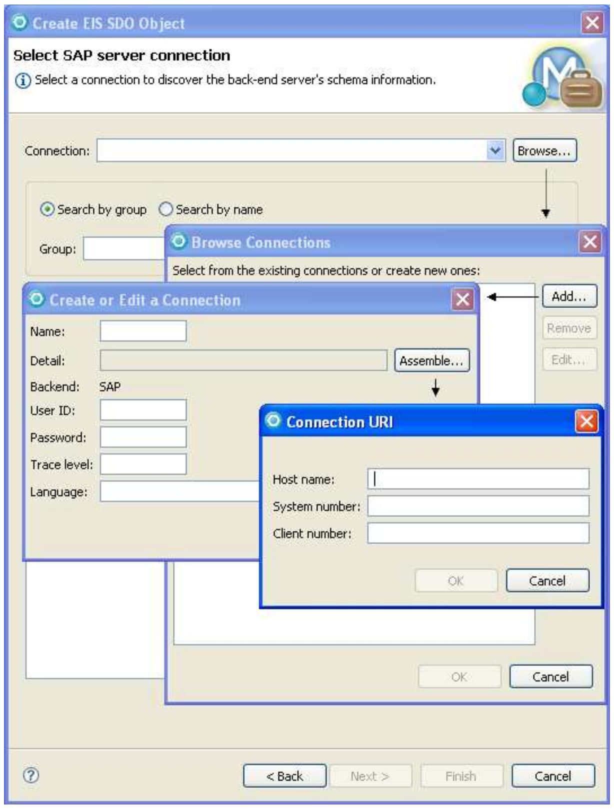


Figure 16: Set up the SAP server connection

After the settings have been made the connection shows up in the *Browse Connections* window. Click OK to proceed with the creation of the SDO.

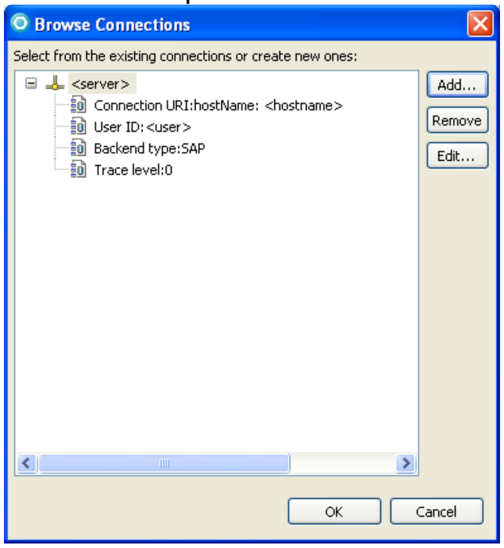


Figure 17: Completed connection

Now it is possible to search for business objects in the SAP system. For this sample the BAPI *BAPI_COMPANY_GETLIST* is needed which can be found using the search by name.

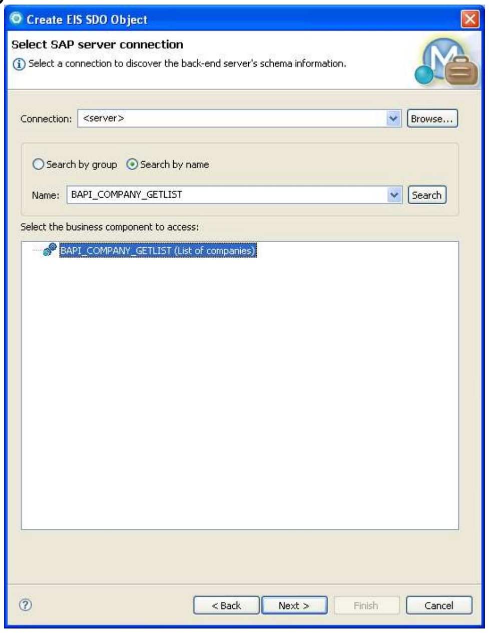
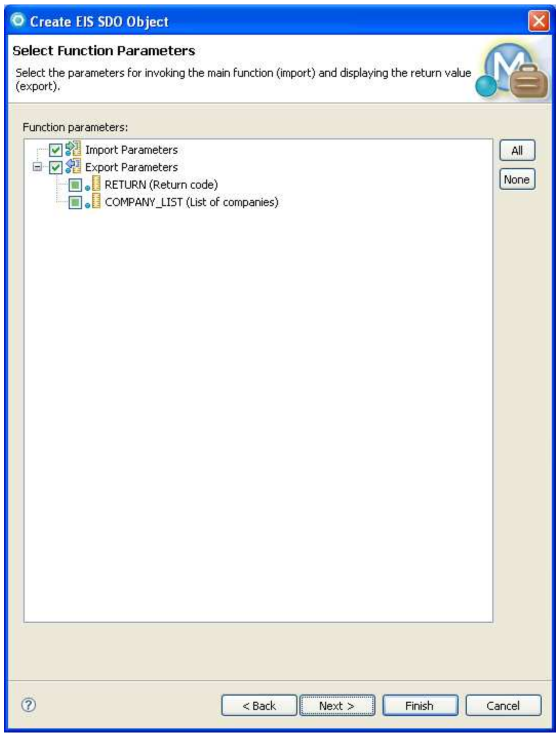

Create EIS SDO Object
Select SAP server connection
Select a connection to discover the back-end server's schema information.
Connection: <server>
Browse...
Search by group
Search by name
Name: BAPI_COMPANY_GETLIST
Search
Select the business component to access:
BAPI_COMPANY_GETLIST (List of companies)
< Back
Next >
Finish
Cancel

Figure 18: Select the BAPI

After clicking on Next the import and export parameters are shown; all should be selected.



Create EIS SDO Object

Select Function Parameters

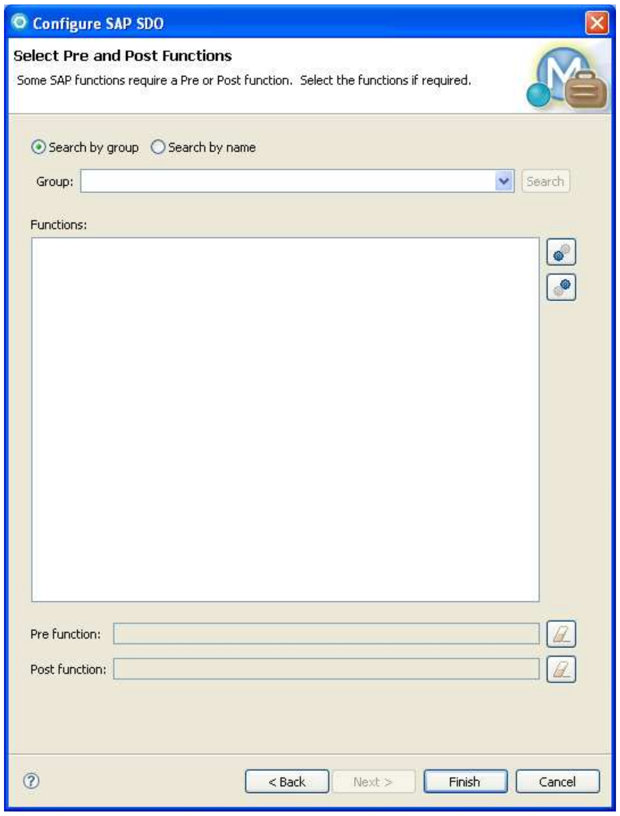
Select the parameters for invoking the main function (import) and displaying the return value (export).

Function parameters:

- Import Parameters
- Export Parameters
  - RETURN (Return code)
  - COMPANY_LIST (List of companies)

Figure 19: Select the BAPI parameters

The next panel lets you define pre- and post- functions that are needed for some SAP Business Objects. In this case none of the functions is required so that the SDO creation can be finished.


Configure SAP SDO

Select Pre and Post Functions

Some SAP functions require a Pre or Post function. Select the functions if required.

(●) Search by group ( ) Search by name

Group:

Search

Functions:

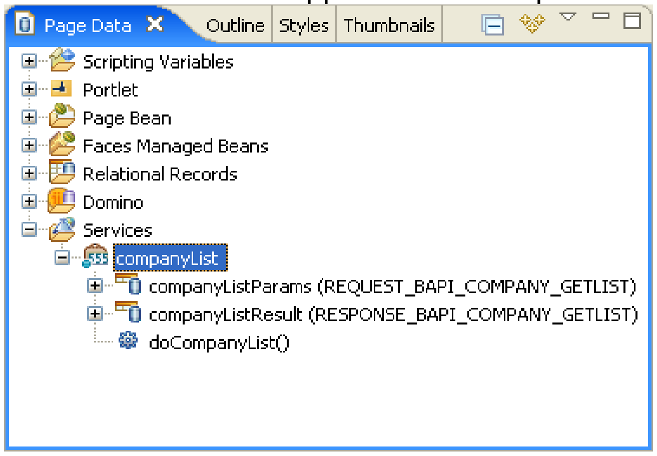
Pre function:

Post function:

< Back Next > Finish Cancel

Figure 20: Pre and post function configuration

The created SDO appears in the *Services* menu of the *Page Data* panel in the bottom left corner of the Rational Application Developer desktop.


Page Data | Outline | Styles | Thumbnails

- Scripting Variables
- Portlet
- Page Bean
- Faces Managed Beans
- Relational Records
- Domino
- Services
  - companyList
    - companyListParams (REQUEST_BAPI_COMPANY_GETLIST)
    - companyListResult (RESPONSE_BAPI_COMPANY_GETLIST)
    - doCompanyList()

Figure 21: SAP SDO in the Page Data view

4.4.2 Adding the JSF components

To create the visualization for the company list data, expand the entry *companyListResult* of the SDO. Select the two entries *COMPANY* and *NAME1* of the COMPANY_LIST_RESPONSE_BAPI and drag them to the design view of the JSF Portlet.

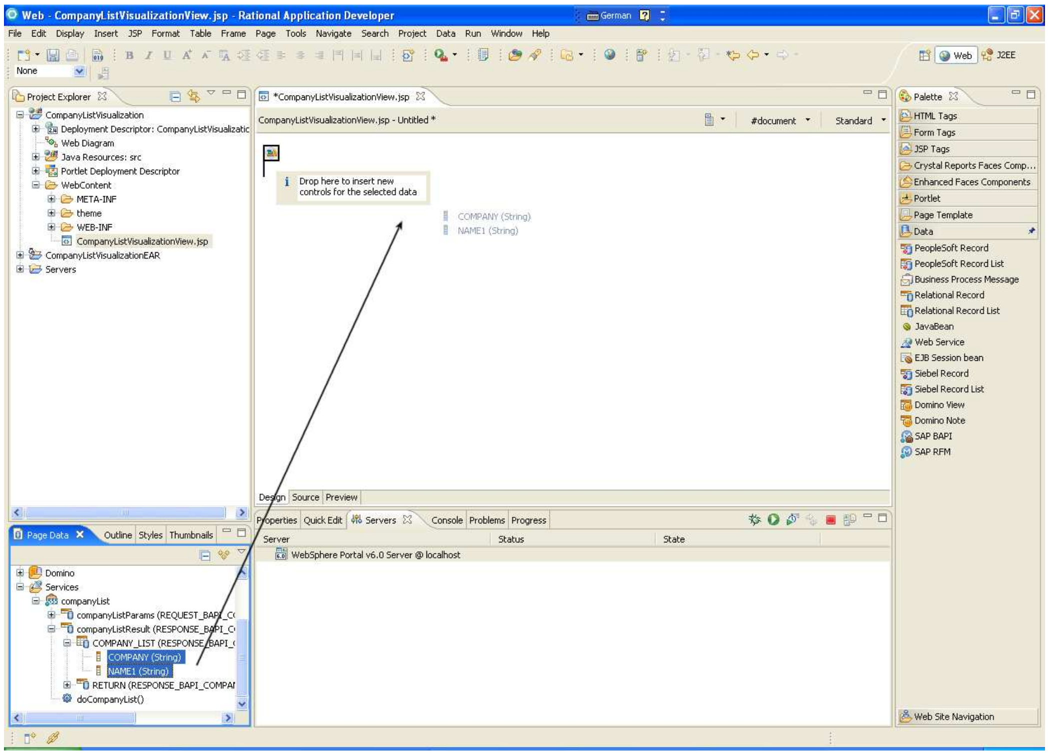
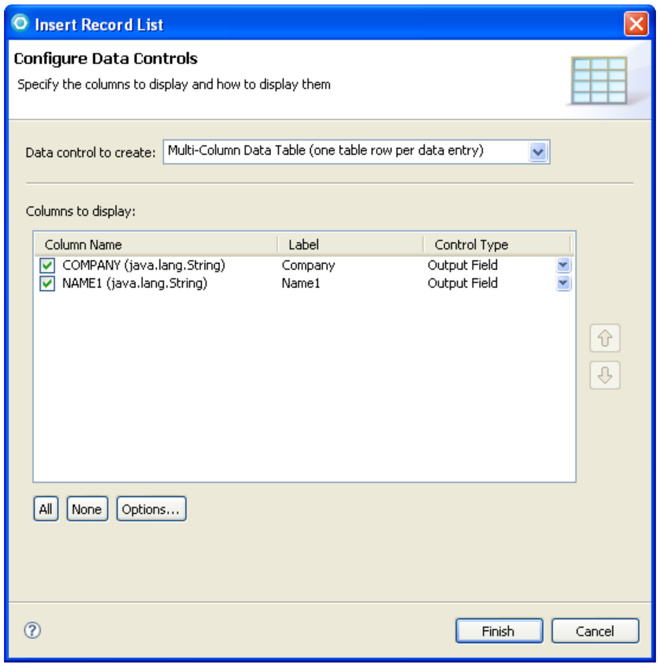

Web - CompanyListVisualizationView.jsp - Rational Application Developer
Project Explorer
Drop here to insert new controls for the selected data
COMPANY (String)
NAME1 (String)
Palette
Page Data
companyListResult (RESPONSE_BAPI_C
COMPANY_LIST (RESPONSE_BAPI_
COMPANY (String)
NAME1 (String)
WebSphere Portal v6.0 Server @ localhost

Figure 22: Add the company list attributes to the JSF Portlet

Drop at the wanted place to insert new controls for the selected data as shown by the information box. The following panel appears:



Configure Data Controls

Specify the columns to display and how to display them

Data control to create: Multi-Column Data Table (one table row per data entry)

Columns to display:

Column Name	Label	Control Type
✓ COMPANY (java.lang.String)	Company	Output Field
✓ NAME1 (java.lang.String)	Name1	Output Field

Figure 23: Configure the appearance of the attributes

Keep the selection “Multi-Column Data Table” because it will create a table that dynamically adds a row for each entry in the result set of the SDO. Click Finish.

The design view should now show a table with two columns belonging to the selected attributes.

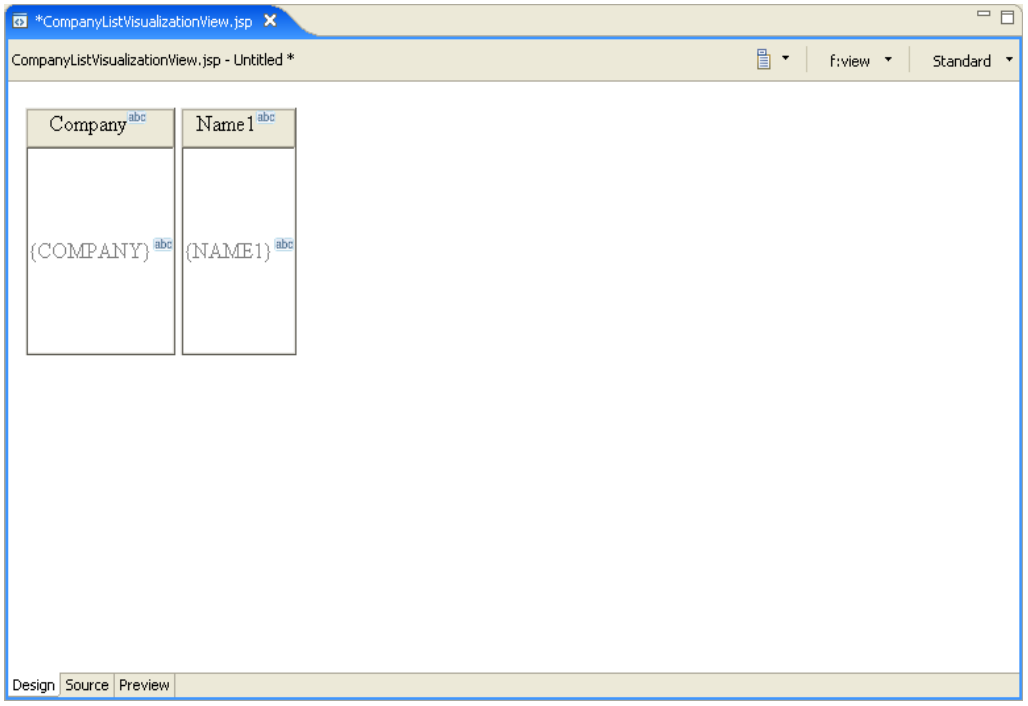

*CompanyListVisualizationView.jsp
CompanyListVisualizationView.jsp - Untitled *
f:view
Standard
Company
Name1
{COMPANY}
{NAME1}
Design
Source
Preview

Figure 24: View of the table that contains the attributes of the company list

The attribute names in the curly brackets indicate that the column body will not show a fixed value but the values from the SAP system. Contrary to that the column title is a fixed value that should be adapted to a more common name.
In order to change the title of a column you have to click on the output text field located in the column header. Then select the properties tab below and enter the name into the value field. For this sample change the first column title to “Id” and the second to “Name”.

Since the data from the SAP system is needed at the time of page load, it is necessary to invoke the appropriate method. For this purpose you have to select the *hx:scriptCollector* as it is shown in Figure 26.

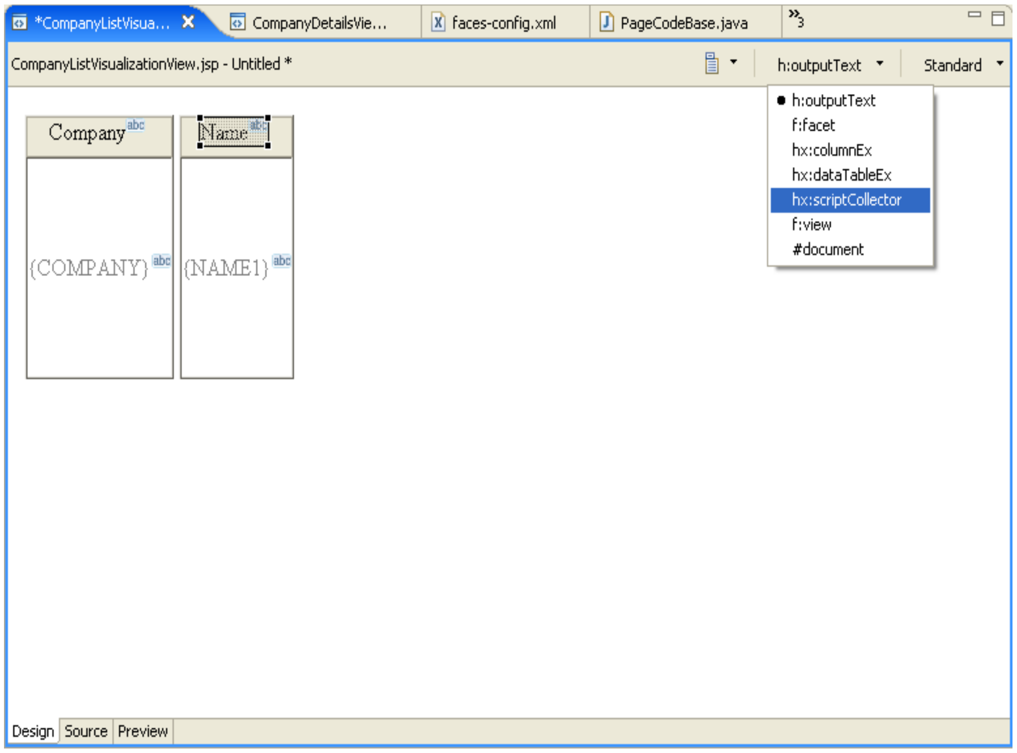
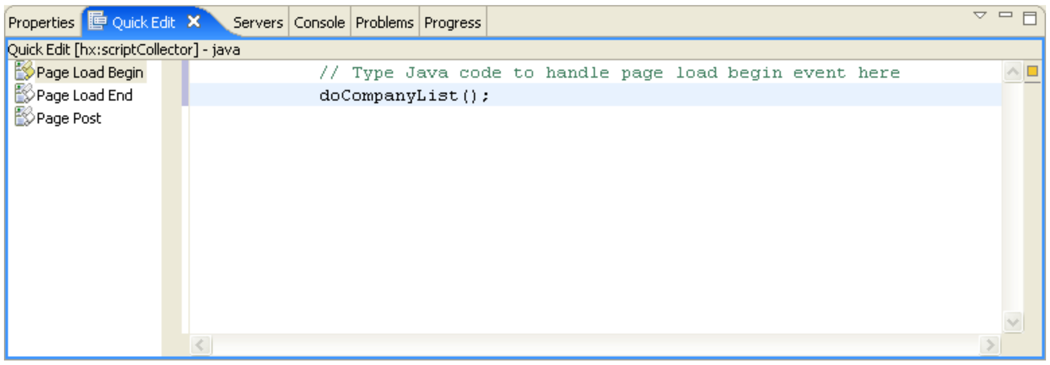


Figure 25: Select the script collector object

Then change to the Quick Edit tab and select the entry *Page Load Begin*. There you can add code that is invoked before the JSP Faces page has been rendered.
Add the line:

doCompanyList();

which is the method call that retrieves the data from the SAP system so that it is accessible through the SAP SDO.



```
// Type Java code to handle page load begin event here
doCompanyList();
```

Figure 26: Add the java code to initialize the SAP SDO

After saving and compiling the project it is ready for the final deployment to the server. Open the context menu by right-clicking on the project in the Project Explorer.

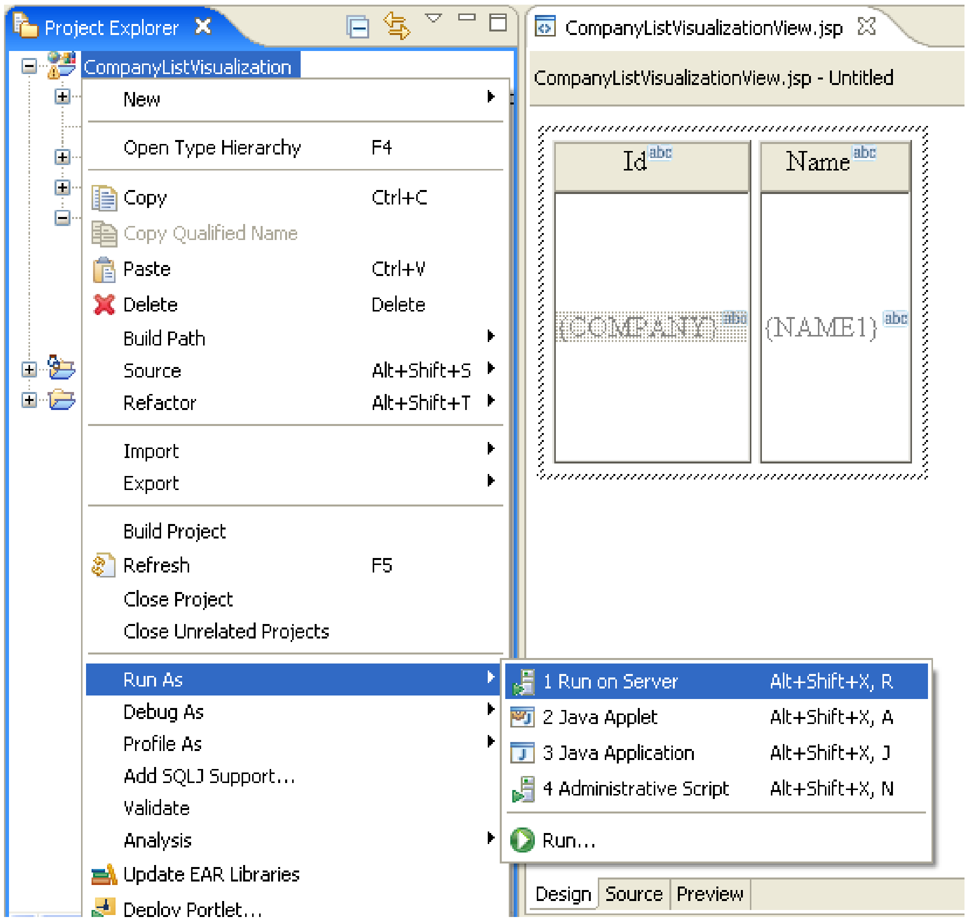
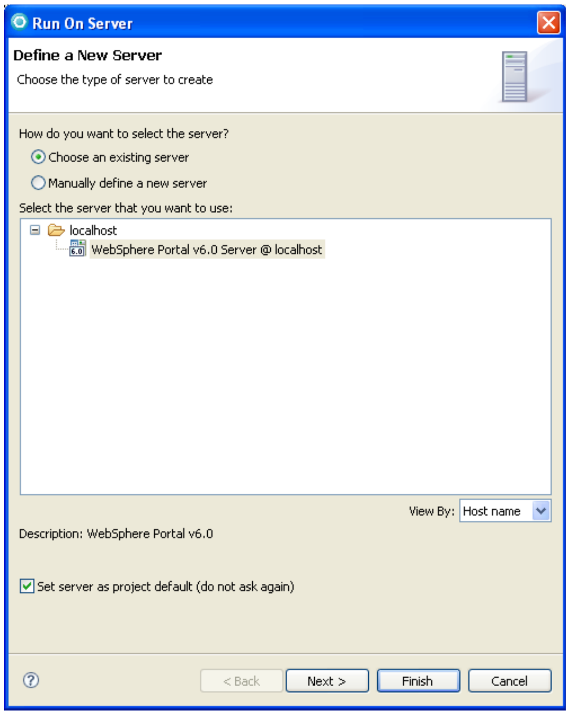


Figure 27: Deploy the project to the server

Select *Run As* and then select *Run on Server*.
The next panel prompts you to choose the server where you have to select the prior created WebSphere Portal Server V6.0. Click Finish and the project will be deployed to the server.


Run On Server

Define a New Server

Choose the type of server to create

How do you want to select the server?

(●) Choose an existing server

( ) Manually define a new server

Select the server that you want to use:

localhost
  WebSphere Portal v6.0 Server @ localhost

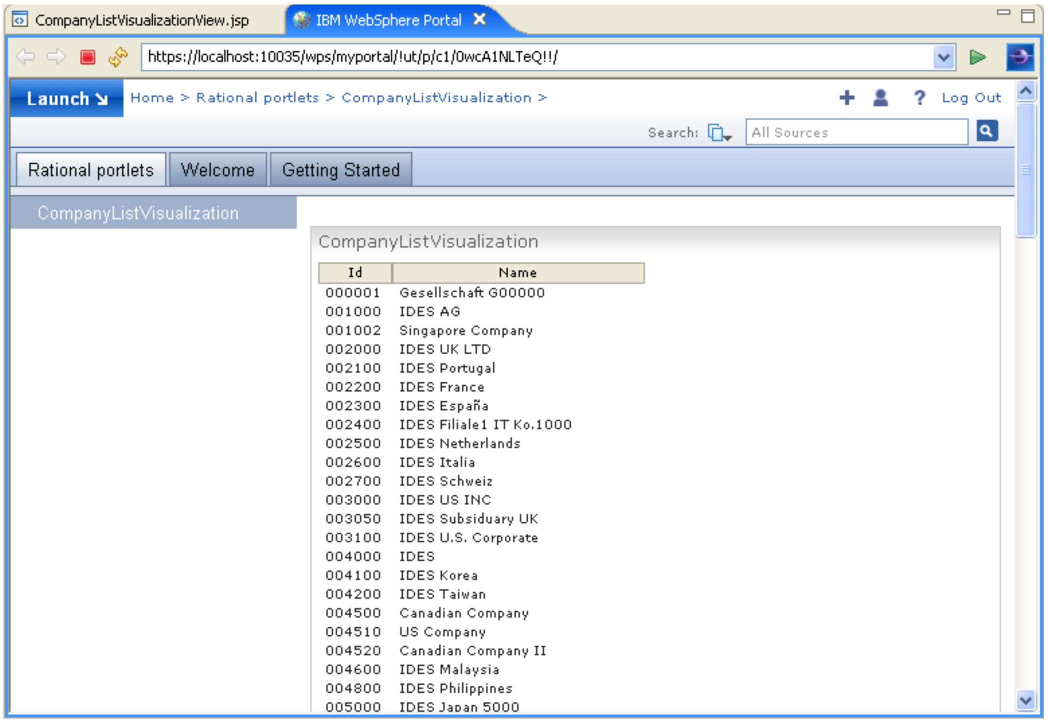
View By: Host name

Description: WebSphere Portal v6.0

[✓] Set server as project default (do not ask again)

Figure 28: Define the target server

If the project has been successfully deployed, IBM Rational Application Developer automatically opens a browser window that shows the generated page containing the visualization of the company list. This should look as follows:



CompanyListVisualizationView.jsp
IBM WebSphere Portal
https://localhost:10035/wps/myportal/!ut/p/c1/0wcA1NLTeQ!!/
Launch
Home > Rational portlets > CompanyListVisualization >
Log Out
Search:
All Sources
Rational portlets
Welcome
Getting Started
CompanyListVisualization
CompanyListVisualization

Id	Name
000001	Gesellschaft G00000
001000	IDES AG
001002	Singapore Company
002000	IDES UK LTD
002100	IDES Portugal
002200	IDES France
002300	IDES España
002400	IDES Filiale1 IT Ko.1000
002500	IDES Netherlands
002600	IDES Italia
002700	IDES Schweiz
003000	IDES US INC
003050	IDES Subsiduary UK
003100	IDES U.S. Corporate
004000	IDES
004100	IDES Korea
004200	IDES Taiwan
004500	Canadian Company
004510	US Company
004520	Canadian Company II
004600	IDES Malaysia
004800	IDES Philippines
005000	IDES Japan 5000

Figure 29: Browser window with the final sample page view

4.5 Summary

In this document, you have seen how to create a JSF Portlet with SAP integration supported by IBM Rational Application Developer tools. As shown by the screen captures, this process is facilitated by various wizards that make it easy for developers to access the data of an SAP system and create a pleasing visualization from it.
Because of the automatic support, there is almost no need to write code yourself and for this reason, less skilled developers are able to develop very fast.

5 References

There is information available about a prior published document dealing with SAP integration using IBM Rational Application Developer 6.0 and Portal V5.1 Server:
http://www.ibm.com/developerworks/rational/library/05/607_sasch/

Detailed information on the SDO Java specification request is JSR-235 and can be found at:
http://www.jcp.org/en/jsr/detail?id=235

For more information about the goals and architecture of SDO, see the Whitepaper: Next-Generation Data Programming: Service Data Objects at:
http://ftpna2.bea.com/pub/downloads/commonj/Next-Gen-Data-Programming-Whitepaper.pdf

Detailed information on the JSF specification can be found at:
http://java.sun.com/j2ee/javaserverfaces/download.html

For more detailed information on JavaServer Faces and Service Data Objects and IBM Rational Application Developer 6.0 developing check the Redbook:
Rational Application Developer V6 Programming Guide
http://www.redbooks.ibm.com