

Getting Started Guide

Version: 3.3.0.M5

1. Installation Instructions	1
1.1. Installing JBoss Tools Plugins	1
1.2. Usage Reporting	10
1.2.1. Collected usage information guide	11
2. Manage JBoss AS with JBoss Tools	13
2.1. How to Manage JBoss AS with JBoss Tools	13
2.1.1. Starting JBoss Server	15
2.1.2. Stopping JBoss Server	16
2.1.3. Server Container Preferences	17
2.2. How to Use Your Own JBoss AS Instance with JBoss Developer Studio	19
2.2.1. JBoss AS Installation	19
2.2.2. Adding and Configuring JBoss Server	19
3. Write Your First Project with JBoss Developer Studio	29
3.1. Create a Seam Application	29
3.1.1. Start Development Database	29
3.1.2. Create and deploy Seam Web Project	31
3.1.3. Start JBoss Application Server	47
3.1.4. Workshop Project Code Overview	53
3.2. Seam Action Development	54
3.2.1. Create a New Seam Action	55
3.2.2. Test Seam Action	56
3.2.3. Modify Seam Action User Interface	58
3.3. Declarative Security	61
3.3.1. Edit Login Authentication Logic	61
3.3.2. Secure Seam Page Component	62
3.4. Browsing Workshop Database	64
3.4.1. Database Connectivity Setup	65
3.4.2. Browse Workshop Database	65
3.5. Database Programming	67
3.5.1. Reverse Engineer CRUD from a Running Database	67
3.5.2. Use Hibernate Tools to Query Data via JPA	71
3.5.3. Use Hibernate Tools to visualize the Data Model	78
3.6. Rich Components	81
3.6.1. Add a Richfaces component to the CRUD Application	81
4. Developing a simple JSP web application	85
4.1. Setting Up the Project	85
4.2. Creating JSP Page	87
4.2.1. Editing a JSP Page	89
4.2.2. web.xml file	90
4.2.3. Deploying the project	92
4.2.4. JSP Page Preview	95
4.2.5. Launch JSP Project	95
5. RAD development of a simple JSF application	97
5.1. Setting up the project	97

5.2. Creating JSP Pages	100
5.3. Creating Transition between two views	103
5.4. Creating Resource File	105
5.5. Creating Java Bean	109
5.6. Editing faces-config.xml File	114
5.7. Editing the JSP View Files	115
5.7.1. Editing inputnumber.jsp page	115
5.7.2. Editing success.jsp page	126
5.8. Creating index.jsp page	128
5.9. Running the Application	129
6. Project Examples	135
6.1. User Sites	137
6.2. Downloading a Project Example	137
6.3. Quick Fixes	144
7. FAQ	151
7.1. What should I do if the Visual Page Editor does not start under Linux	151
7.2. Visual Editor starts OK, but the Missing Natures dialog appears	152
7.3. Do I need to have JBoss Server installed to run JBoss Developer Studio?	153
7.4. I have an existing Seam 1.2.1 project. Can I migrate or import the project into a JBoss Developer Studio Seam project?	153
7.5. I have an existing Struts or JSF project. Can I open the project in JBoss Developer Studio?	153
7.6. Can I import a WAR file?	153
7.7. Is it possible to increase the performance of Eclipse after installing your product?..	154
7.8. How can I add my own tag library to the JBoss Tools Palette?	154
7.9. How to get Code Assist for Seam specific resources in an externally generated project?	154
7.10. How to import an example Seam project from jboss-eap directory?	154
7.11. Is a cross-platform project import possible for JBoss Developer Studio?	154
8. Further Reading	157

Installation Instructions

1.1. Installing JBoss Tools Plugins

The JBoss Tools plugins can be installed in Eclipse from the JBoss.org update site. JBoss Tools 3.2 requires Eclipse 3.6, which can be downloaded from the [Eclipse web site](http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/helios/) [http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/helios/].

To install the JBoss Tools plugins start Eclipse and select **Help** → **Install New Software...**

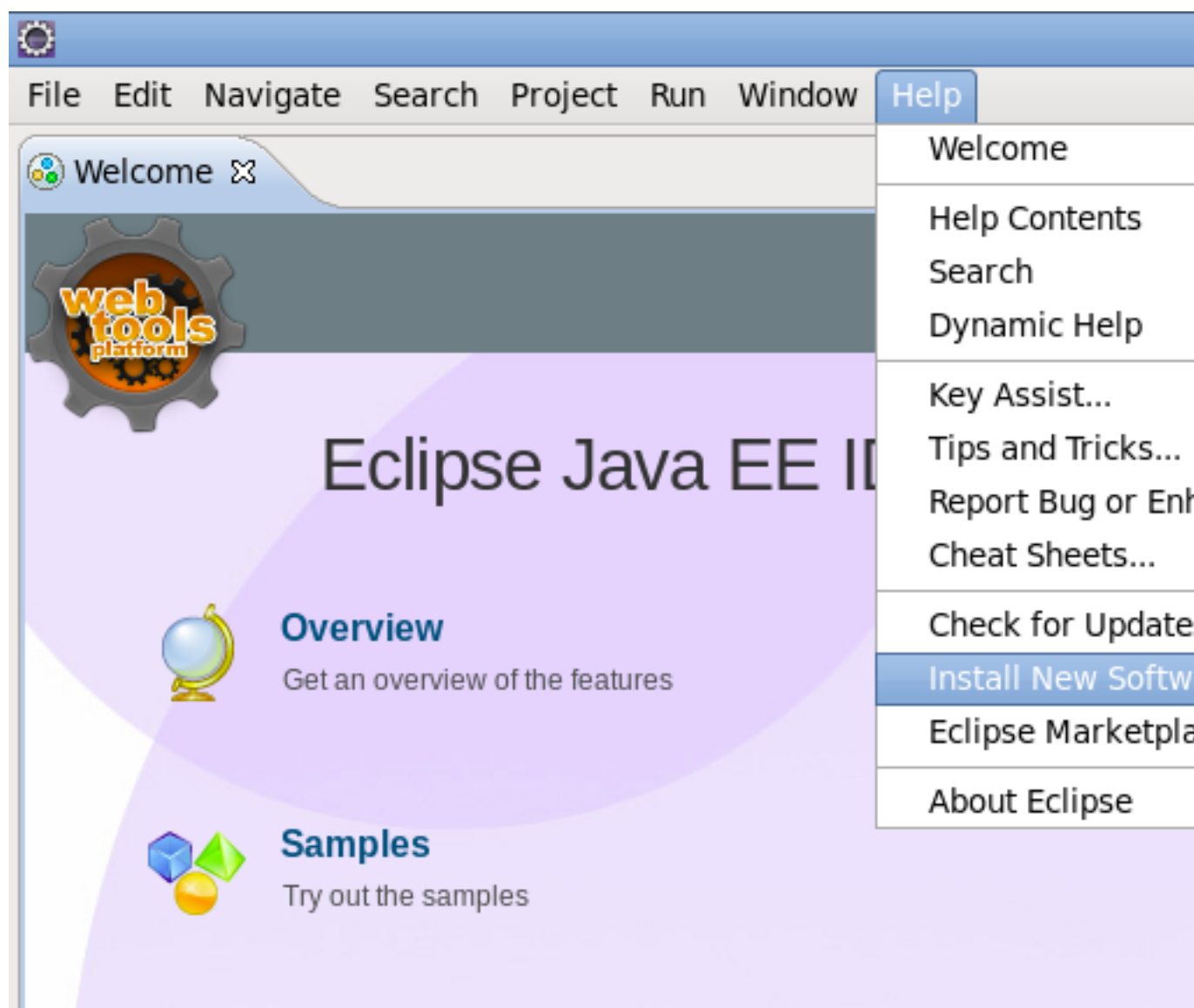


Figure 1.1. Install New Software

Click the **Add...** button.

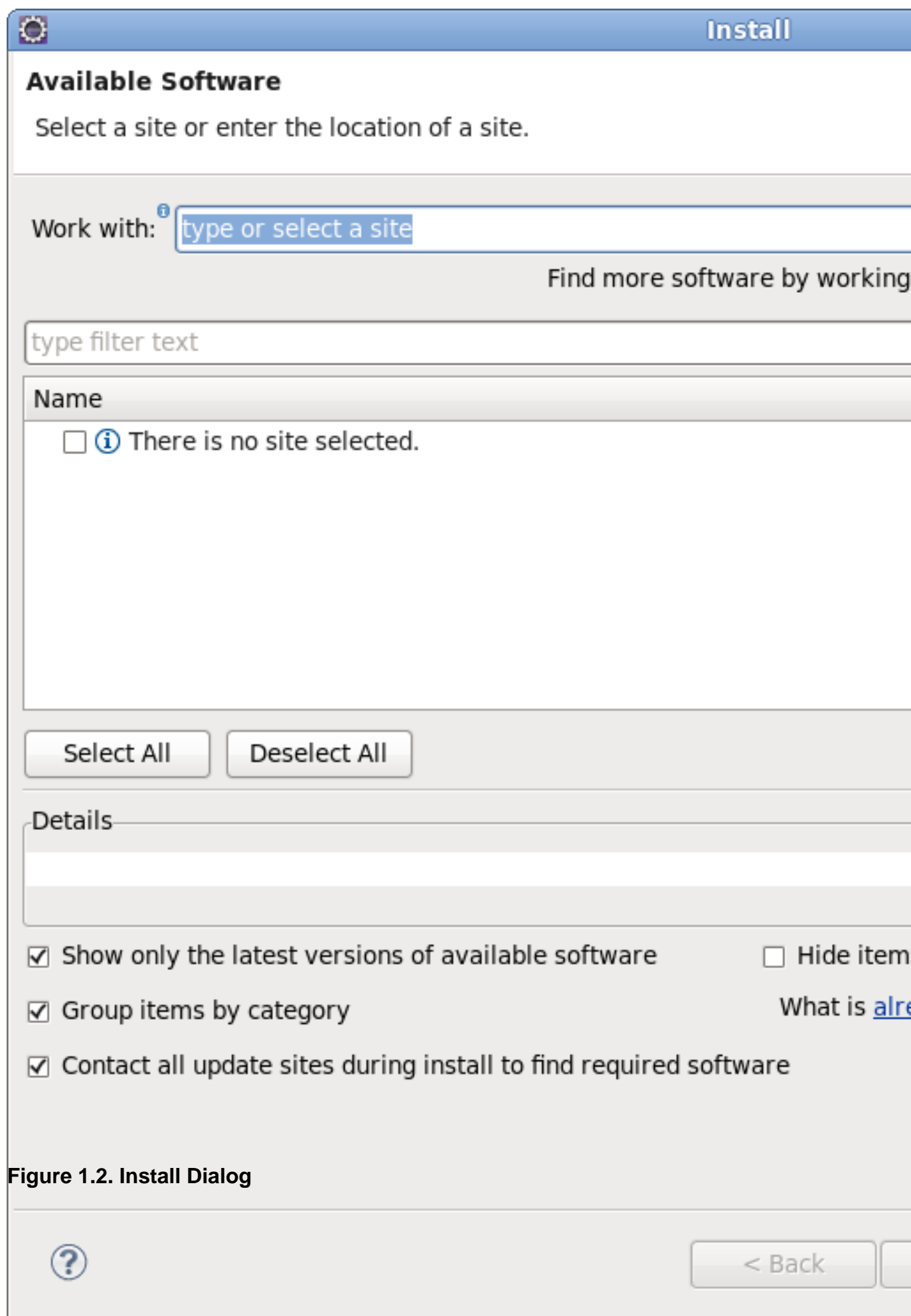


Figure 1.2. Install Dialog

This will display the **Add Repository** dialog. Enter **JBoss.org Tools** in the **Name** field, and <http://download.jboss.org/jbosstools/updates/JBossTools-3.2.0.GA/> in the **Location** field. Click the **OK** button to save the changes and close the dialog.

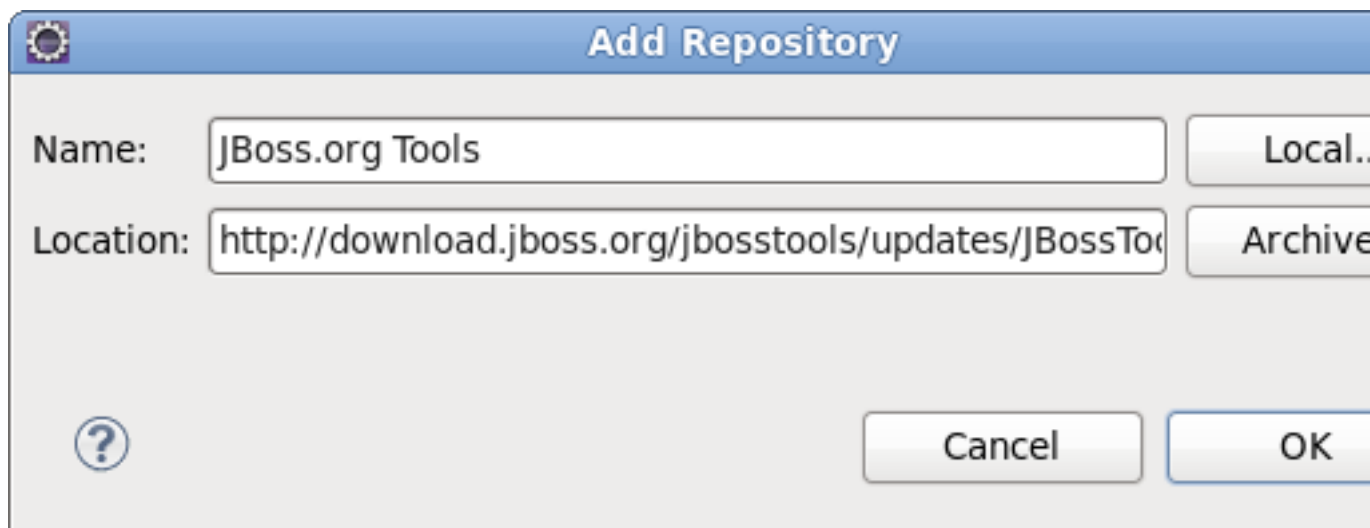


Figure 1.3. Add Repository

The **JBoss.org Tools** site will be selected in the **Work with** drop down list, and after a moment the list of plugins that are included in the JBoss Tools package will be listed. From this list you can individually select the desired plugin, or select the **All JBoss Tools 3.2.0** option to install all the plugins.

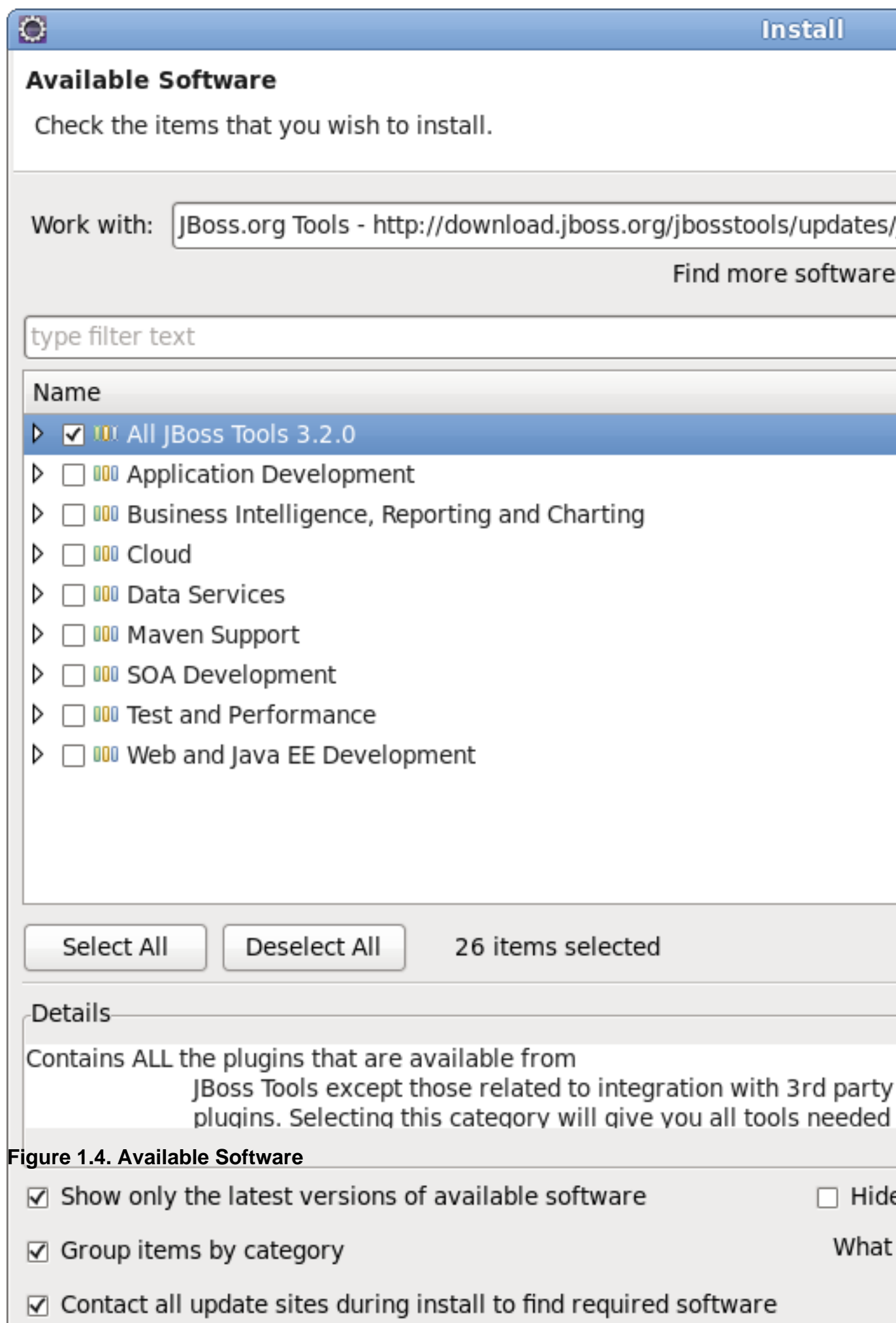


Figure 1.4. Available Software

Click the **Next** button to calculate the system requirements and dependencies (this may take a little while). You will then be given an opportunity to review the plugins that will be installed.

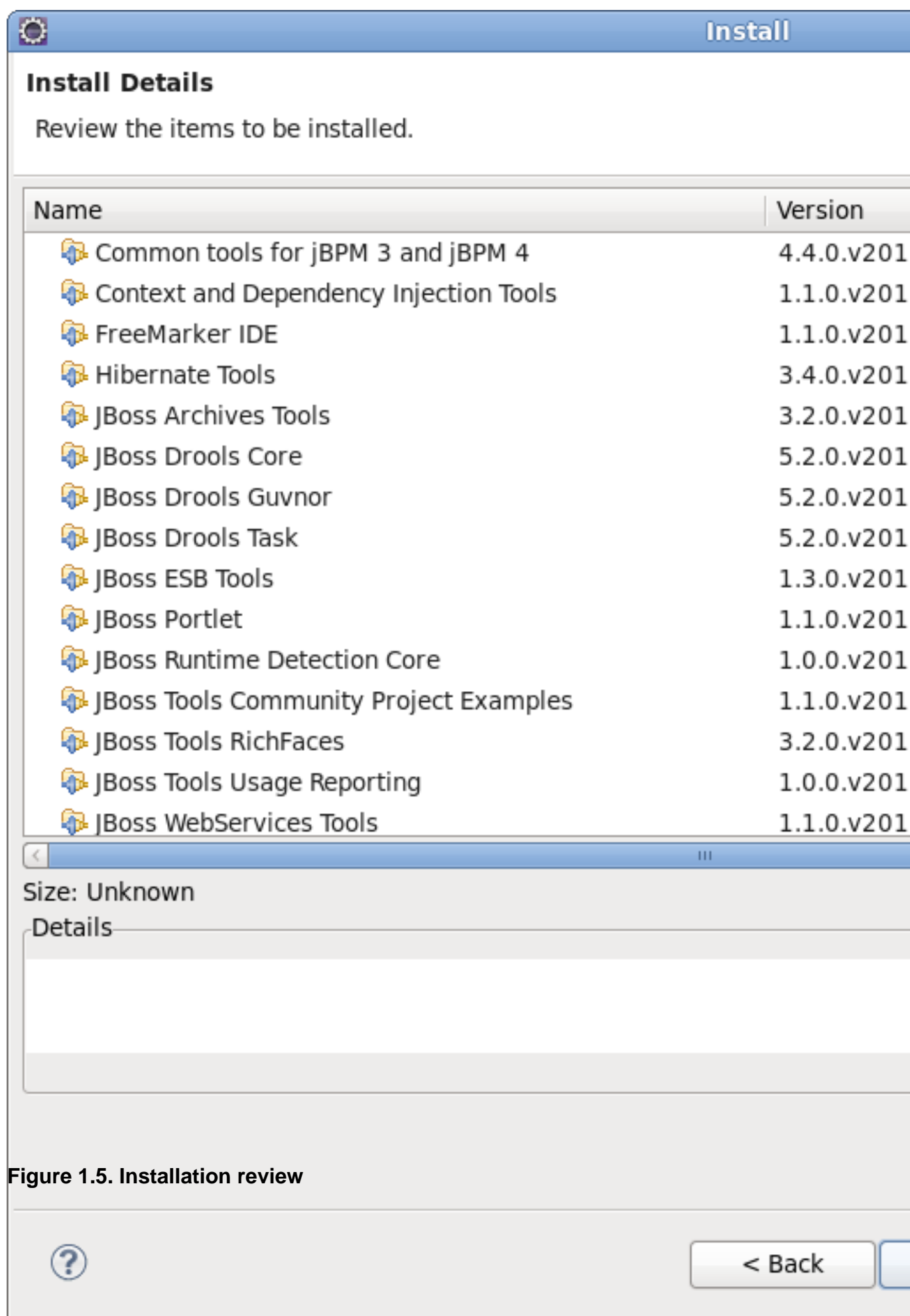


Figure 1.5. Installation review

Click the **Next** button to install the selected plugins. You will be prompted to accept the various license agreements that cover the plugins that are to be installed. Review the licenses, select the **I accept the terms of the license agreements** option, and click the **Finish** button to install the plugins.

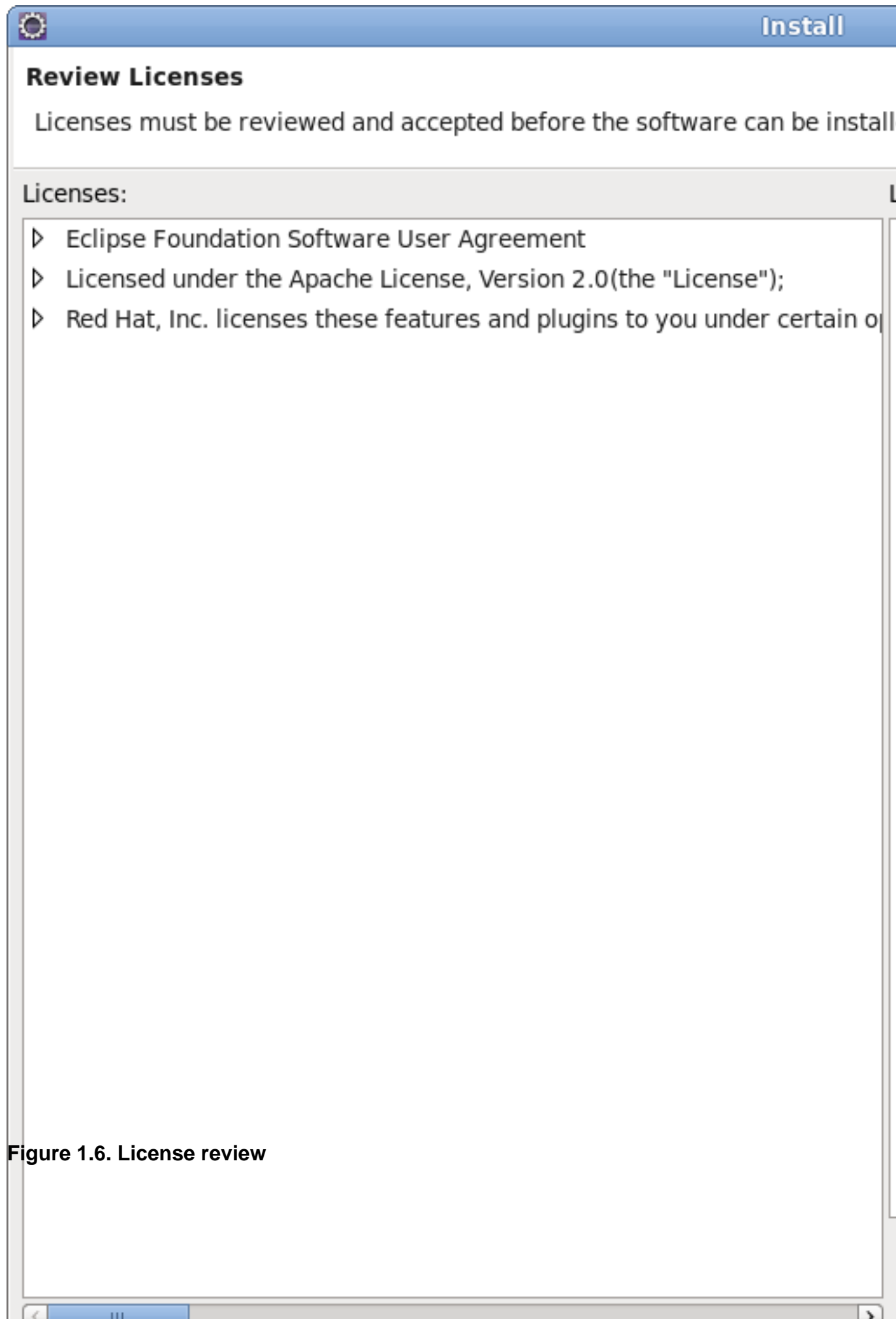


Figure 1.6. License review

Wait while the plugins are downloaded and installed.

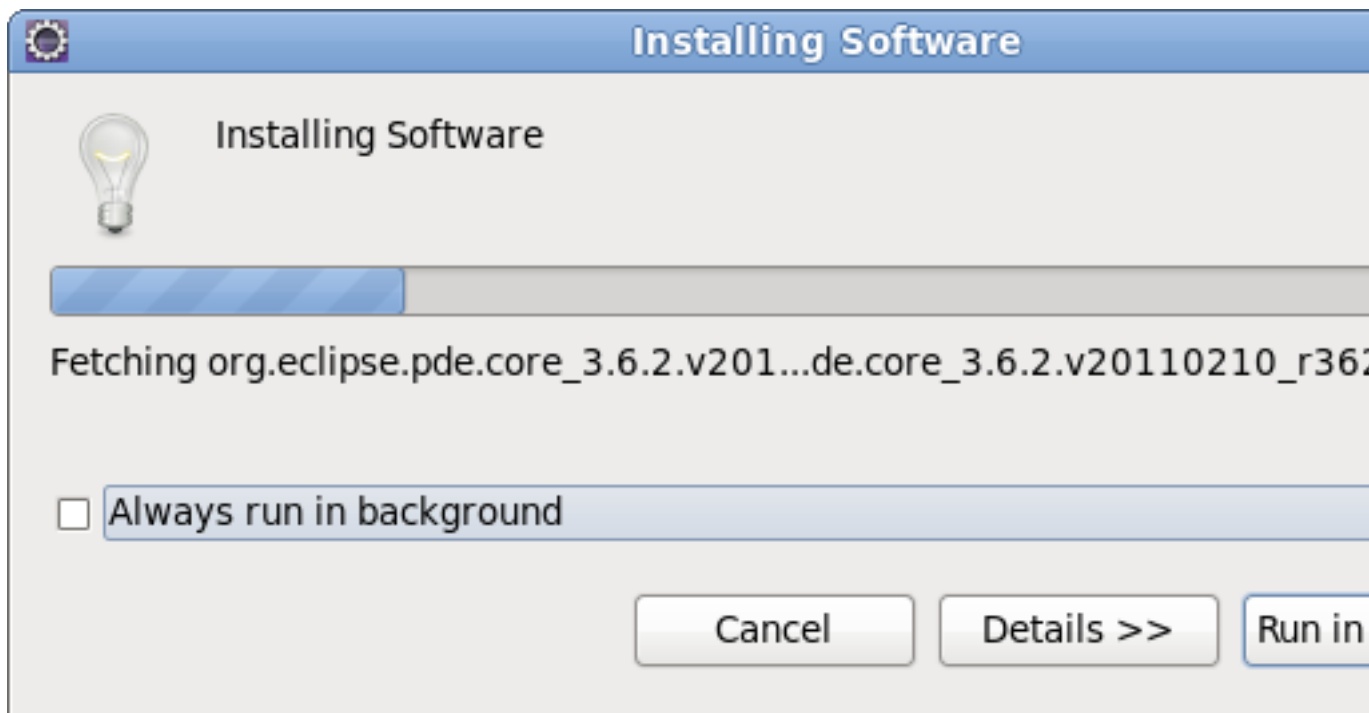


Figure 1.7. Installing Software

You may be prompted with a warning informing you that you are attempting to install unsigned content. Click the **OK** button to continue.



Figure 1.8. Unsigned Software Warning

You will then have to restart Eclipse to apply the new plugins. Click the **Restart Now** button to restart Eclipse.

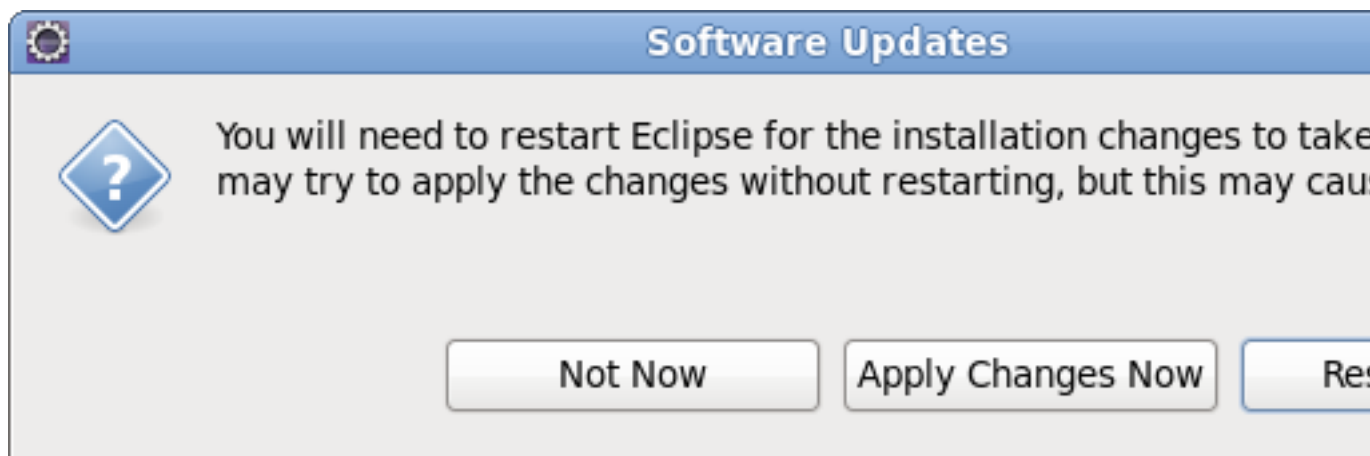


Figure 1.9. Restart Eclipse

The plugin is now installed and ready to use.

1.2. Usage Reporting

JBoss Tools now includes a usage plug-in that anonymously reports information back to JBoss. The plug-in is not enabled by default. To enable, click the **Yes** button.

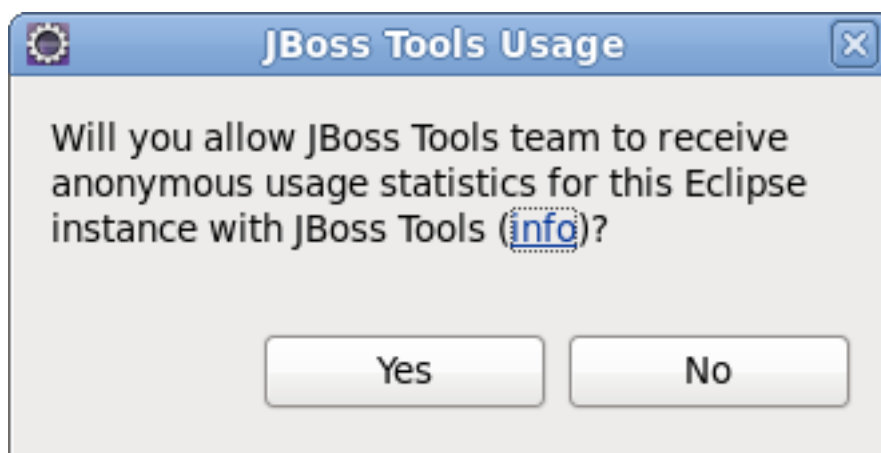


Figure 1.10. Usage plug-in pop-up

Once enabled, the plug-in will remain active until turned off. To turn the active plug-in off, navigate to **Window** → **Preferences** → **JBoss Tools** → **Usage Reporting**.

The gathered data allows JBoss to see how the tools are being used and where they are being used geographically. Currently we are looking into the operating systems being used, screen resolution and how often the tooling environment is started. In the future geographic information will assist in focusing translation resources to areas where the developer environment is most used.

The plug-in uses Google Analytics to track and report data by acting as if you were visiting the site <http://jboss.org/tools/usage/>. To view the type of information being collected, refer to [Section 1.2.1, “Collected usage information guide”](#).

To view the source code of the usage plug-in visit <http://anonsvn.jboss.org/repos/jbosstools/trunk/usage/>.

1.2.1. Collected usage information guide

Below you will find an outline of the information that is reported and the Google Analytics fields that are used to gather this information.

Version

The **Content** field has been modified to report the installed JBoss Developer Studio version. Sample returned values include: `jbdevstudio-linux-gtk-x86_64-4.0.0.v201009301221R-H20-Beta1.jar` and `jbdevstudio-linux-gtk-3.0.2.v201009161622R-H138-GA.jar`.

Installed components

The **Keyword** field has been modified to report the installed JBoss Developer Studio components. Sample returned values include: JBoss AS, Drools, Teiid and ModeShape.

Visitor type

The **Visitor type** field reports if the current user is new or returning.

Language

The **Language** field reports the localized language the product is being used in. Sample returned values include: `en-US`, `de-DE` and `fr-FR`.

Location fields

The location fields report the geographical location where the product is being used based on the continent, country and city. Sample returned values include: `Europe` (continent), `Germany` (country) and `Munich` (city).

Eclipse interface and version

The **Browser** field has been modified to report the Eclipse interface and version being used. Sample returned values include: `JBoss Developer Studio: 3.0.0` and `JBoss Developer Studio: 3.0.1`.

Operating System

The **Operating System** field reports the Operating System and its version that the product is running on (with Linux distribution version reporting conducted through the **User Defined Value** field). Sample returned values include: `Linux`, `Macintosh 10.4`, `Macintosh 10.6`, `Windows XP` and `Windows 7`.

Linux distribution version

The **User Defined Value** field reports the distribution and version of Linux, if one is being used as the Operating System. Sample returned values include: `Red Hat Enterprise Linux 5.4` and `Fedora 13`.

Screen colors

The **Screen colors** field reports the color depth being used. Sample returned values include: 32-bit and 24-bit.

Screen resolution

The **Screen resolution** field reports the resolution being used. Sample returned values include: 2048x1536 and 1920x1080.

Java version

The **Flash version** field has been modified to report the Java version used. Sample returned values include: 1.6.0_20 and 1.5.0_9.

Connection speed

The **Connection speed** field reports the type of internet connection being used. Sample returned values include: T1, Cable and DSL.

Manage JBoss AS with JBoss Tools

In this chapter we'll focus more on how to operate the JBoss AS from JBoss Tools.

JBoss Tools can be used to operate a wide range of JBoss Application Server versions, which can be downloaded from the [JBoss Application Server](http://www.jboss.org/jbossas/downloads.html) [http://www.jboss.org/jbossas/downloads.html] website.

2.1. How to Manage JBoss AS with JBoss Tools

This section covers the basics of working with the JBoss Server supported directly by JBoss Developer Studio via bundled AS plug-in. The server points to the JBoss Enterprise Application Platform Runtime shipped with JBoss Developer Studio.

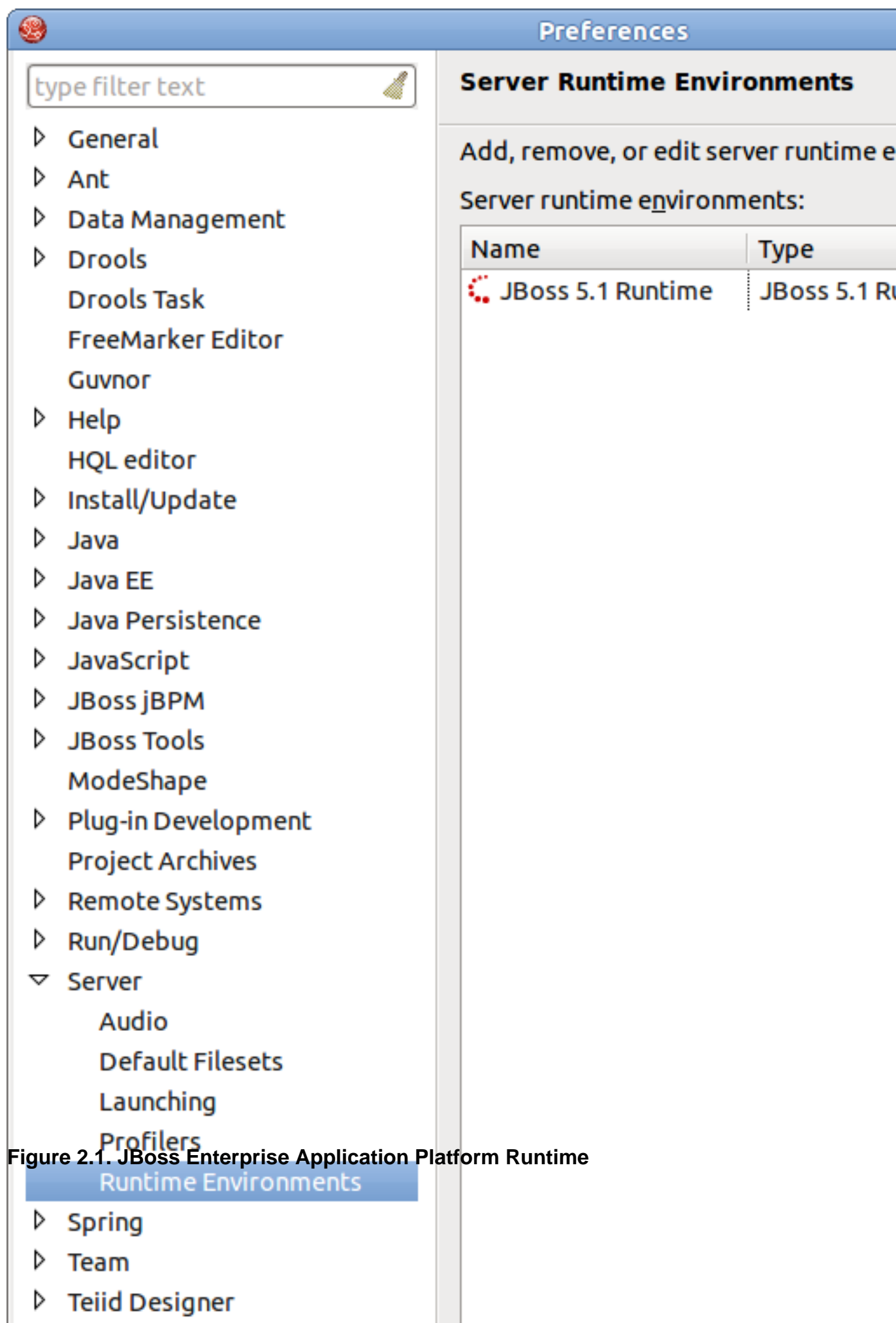


Figure 2.1. JBoss Enterprise Application Platform Runtime

To read more about AS plug-in, refer to the Server Manager guide.

2.1.1. Starting JBoss Server

Starting JBoss Server is quite simple. JBoss Developer Studio allows you to control its behavior with the help of a special toolbar, where you could start it in a regular or debug mode, stop it or restart it.

- To launch the server click the green-with-white-arrow icon in the Servers view or right click server name in this view and click the **Start** button. If this view is not open, select **Window** → **Show View** → **Other** → **Server** → **Servers**

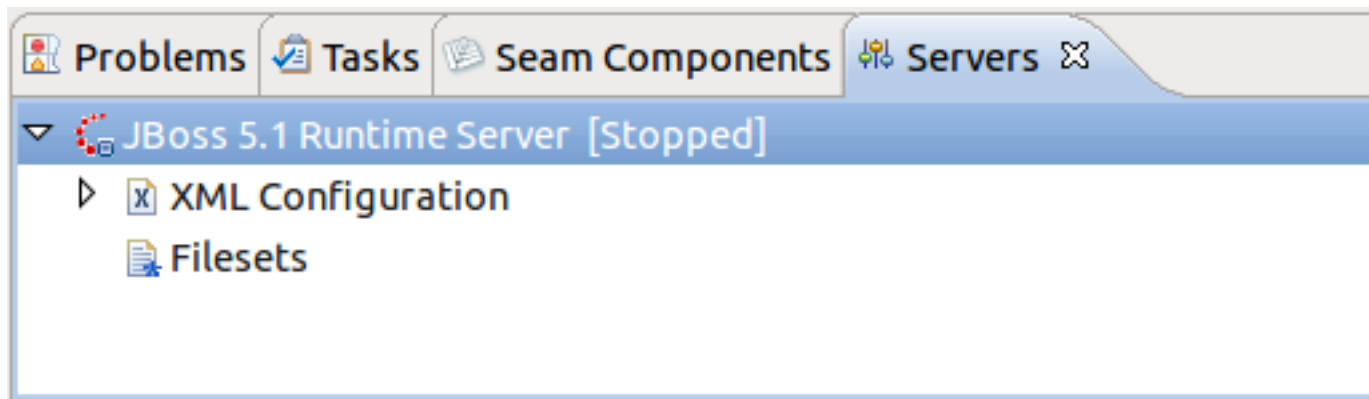


Figure 2.2. Starting from Icon

While launching, server output is written to the Console view:

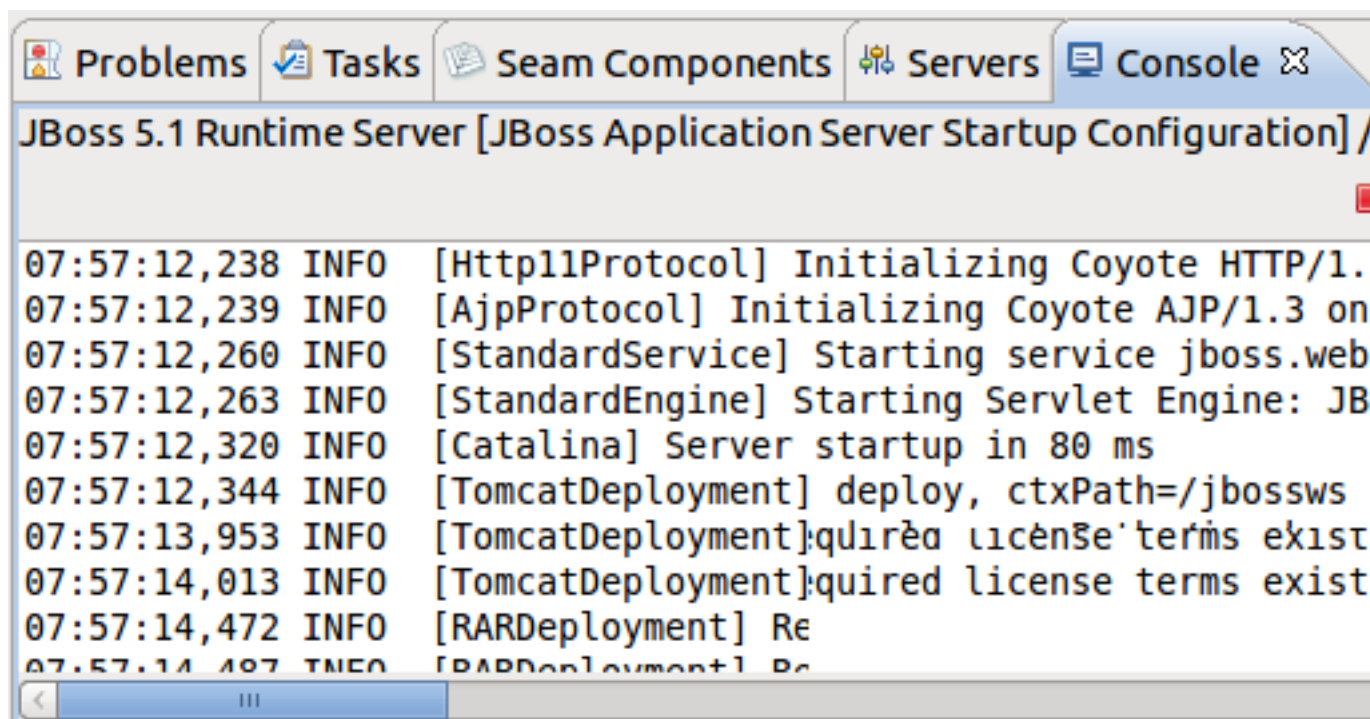


Figure 2.3. Console Output

When the server is started you should see *Started* in the square brackets right next its name in the Servers view.

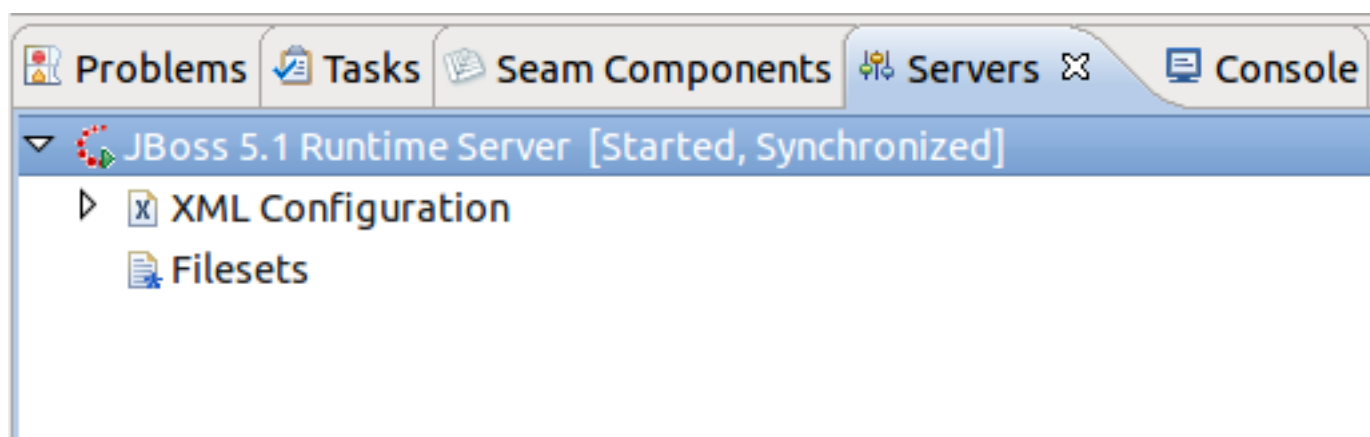


Figure 2.4. Server is Started

2.1.2. Stopping JBoss Server

To stop the server, click the **Stop** button icon in Servers or right click the server name and press **Stop**.

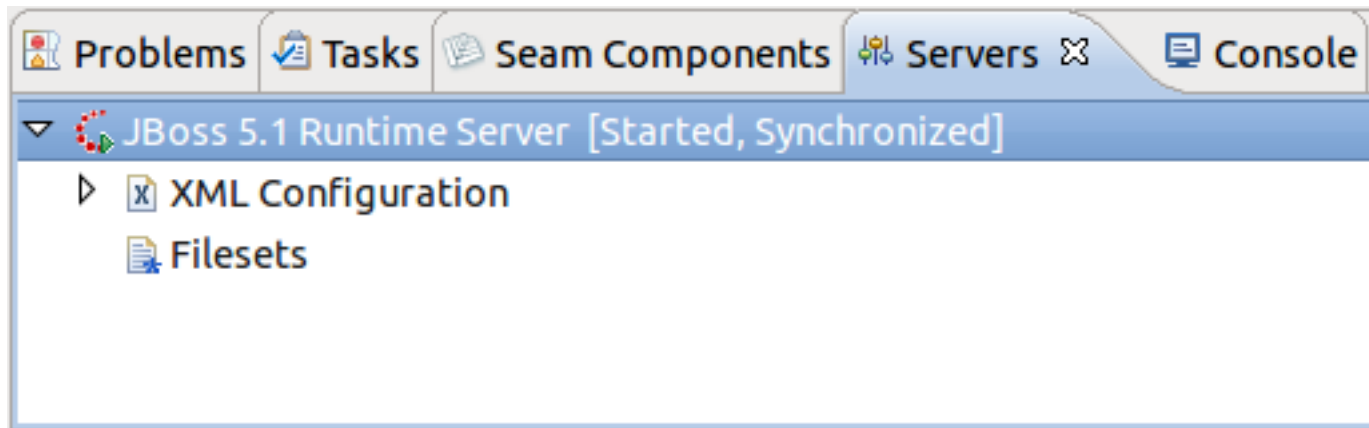


Figure 2.5. Stopping Server

When the server is stopped you will see *Stopped* in the square brackets next to its name.

2.1.3. Server Container Preferences

You can control how JBoss Developer Studio interacts with server containers in the Server editor. Double-click the server to open it in the editor.

JBoss 5.1 Runtime Server

Overview

General Information
Specify the host name and other common settings.

Server name:

JBoss 5.1 Runtime Server

Host name:

localhost

Runtime Environment:

JBoss 5.1 Runtime

[Open launch configuration](#)

Server Behaviour

Local

JMX Login Credentials

Set the JMX login and password for your server.
This is used by all JMX commands, and during server shutdown.

User Name

admin

Password

admin

Overview

Deployment

Figure 2.6. Server Overview

Here you can specify some common settings: host name, server name, runtime as well as settings related to publishing, timeouts and server ports.

2.2. How to Use Your Own JBoss AS Instance with JBoss Developer Studio

Although JBoss Developer Studio works closely with JBoss EAP 5 we do not ultimately tie you to any particular server for deployment. There are some servers that Studio supports directly (via the bundled Eclipse WTP plug-ins). In this section we discuss how to manage self-installed JBoss AS. Suppose you want to deploy the application to JBoss 4.2.3 server. First of all you need to install it.

2.2.1. JBoss AS Installation

- Download the binary package of JBoss AS, e.g. JBoss 4.2.3 and save it on your computer:
<http://labs.jboss.com/jbossas/downloads>

It does not matter where on your system you install JBoss server.



Note:

The installation of JBoss server into a directory that has a name containing spaces provokes problems in some situations with Sun-based VMs. Try to avoid using installation folders that contain spaces in their names.

There is no requirement for root access to run JBoss Server on UNIX/Linux systems because none of the default ports are within the 0-1023 privileged port range.

- After you have the binary archive you want to install, use the JDK jar tool (or any other ZIP extraction tool) to extract the `jboss-4.2.3.GA.zip` archive contents into a location of your choice. The `jboss-4.2.3.GA.tgz` archive is a gzipped tar file that requires a gnutar compatible tar which can handle the long pathnames in the archive. The extraction process will create a `jboss-4.2.3.GA` directory.

2.2.2. Adding and Configuring JBoss Server

Now we should add the just installed server into server manager in JBoss Developer Studio.

- Select the Servers view by selecting **Window** → **Show View** → **Other** → **Server** → **Servers**.
- Right click anywhere in this view and select **New** → **Server**.
- Select **JBoss Community** → **JBoss 4.2 Server**

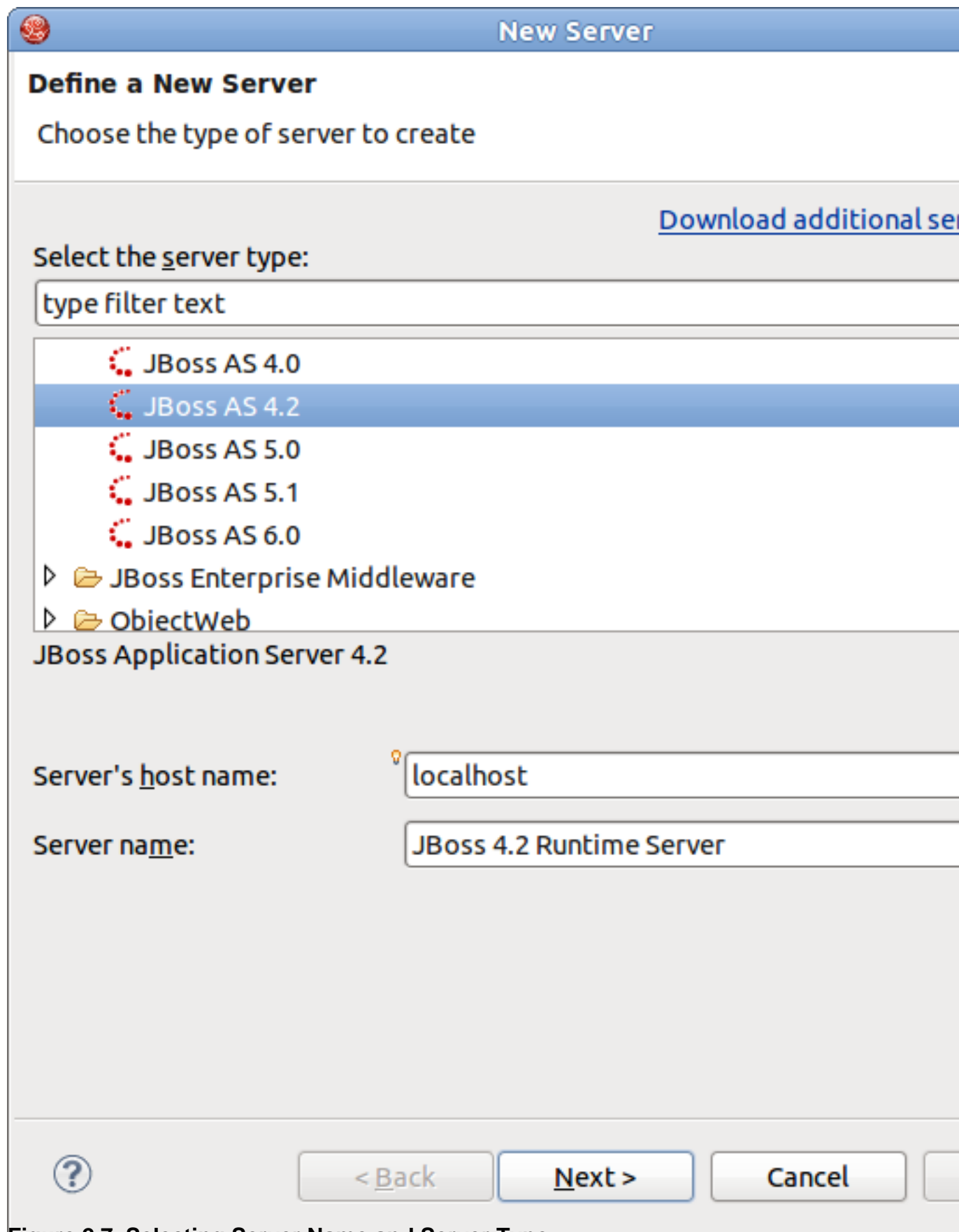


Figure 2.7. Selecting Server Name and Server Type



Note:

Now in the New Server wizard there is a separation between the .org servers (the *JBoss Community* category) and product server that comes bundled with JBoss EAP (the *JBoss Enterprise Middleware* category).

- To create a new runtime, which Jboss AS 4.2 matches to, click the **Next** button
- In the next step you need to specify the location of the Server and define JRE to be used.

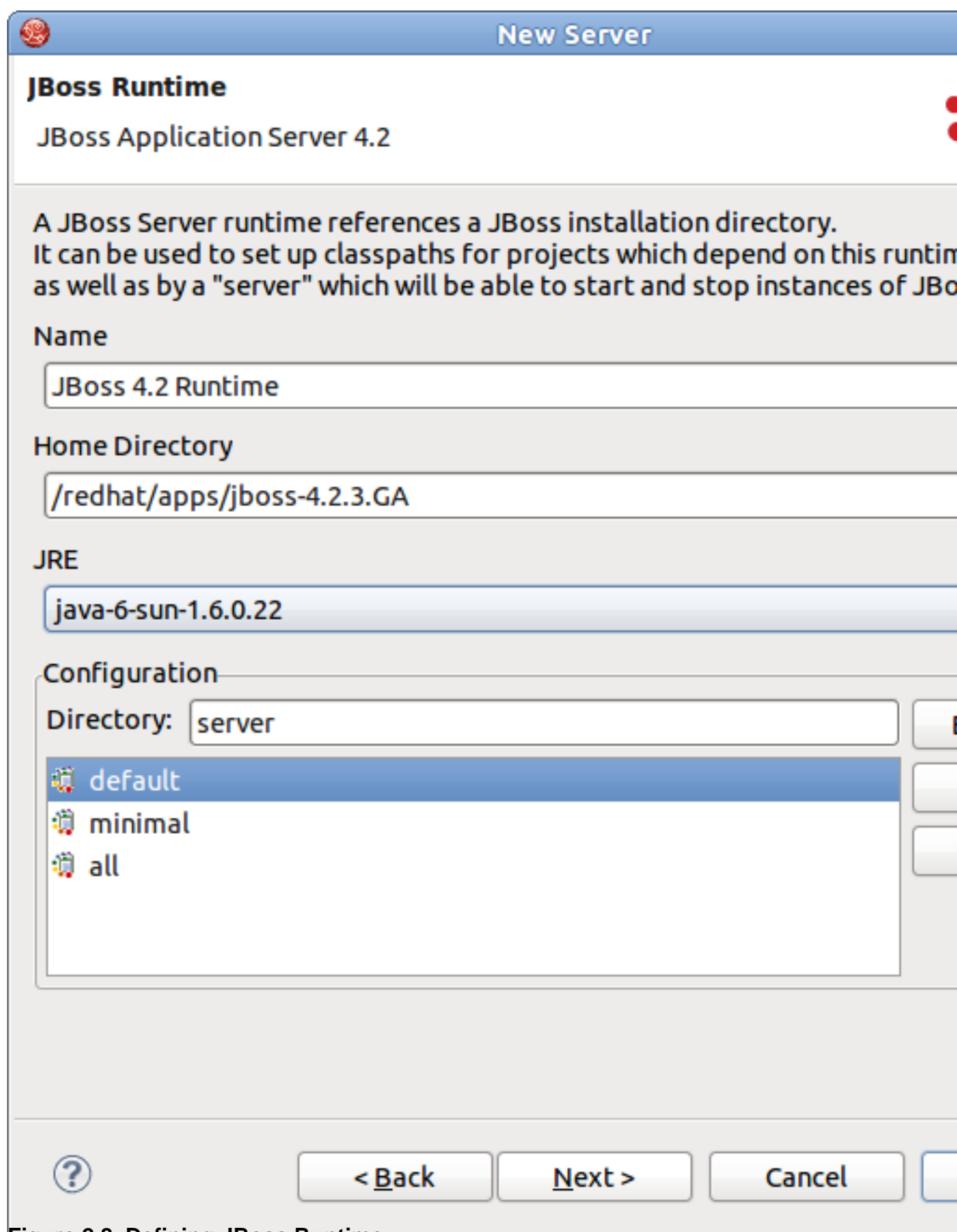


Figure 2.8. Defining JBoss Runtime



Note:

When adding a new server you will need to specify what JRE to use. It is important to set this value to a full JDK, not JRE. Again, you need a full JDK to run Web applications, JRE will not be enough.

- In the next dialog verify the specified information and if something is unfair go back and correct it

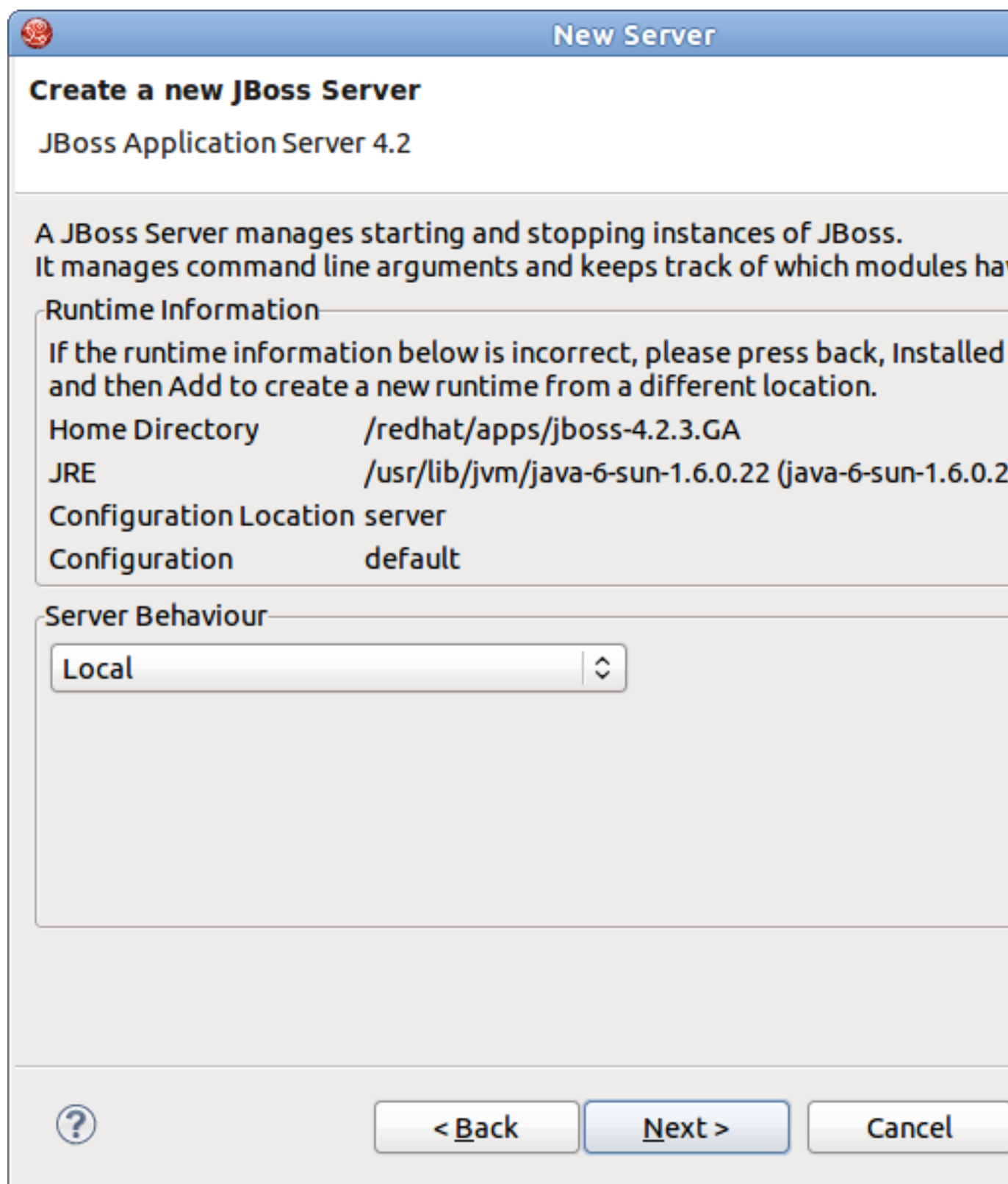


Figure 2.9. JBoss Runtime Summary

- In the last wizard's dialog modify the projects that are configured on the server and click the **Finish** button.

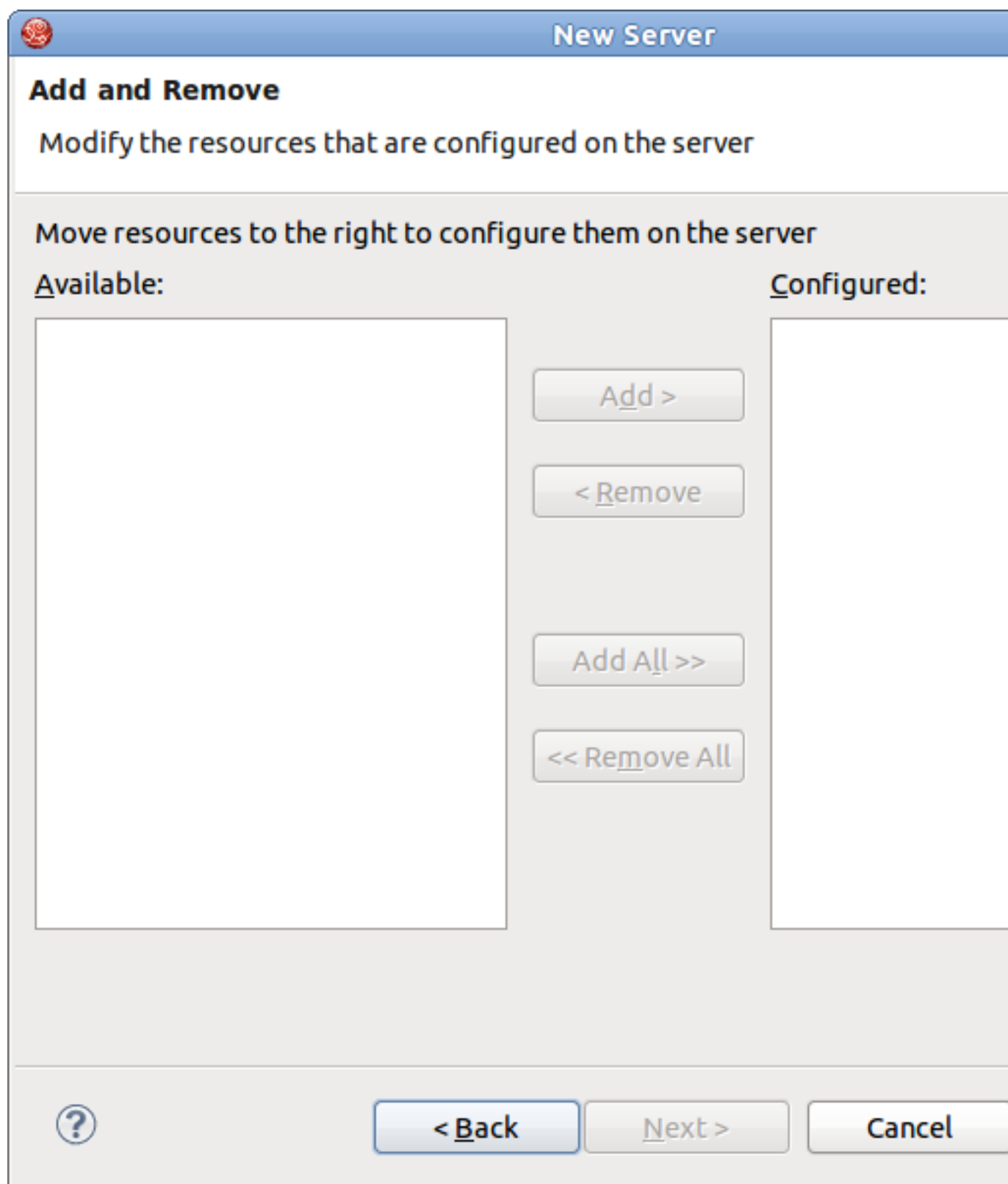


Figure 2.10. Configuring Projects

A new JBoss Server should now appear in the Servers view.

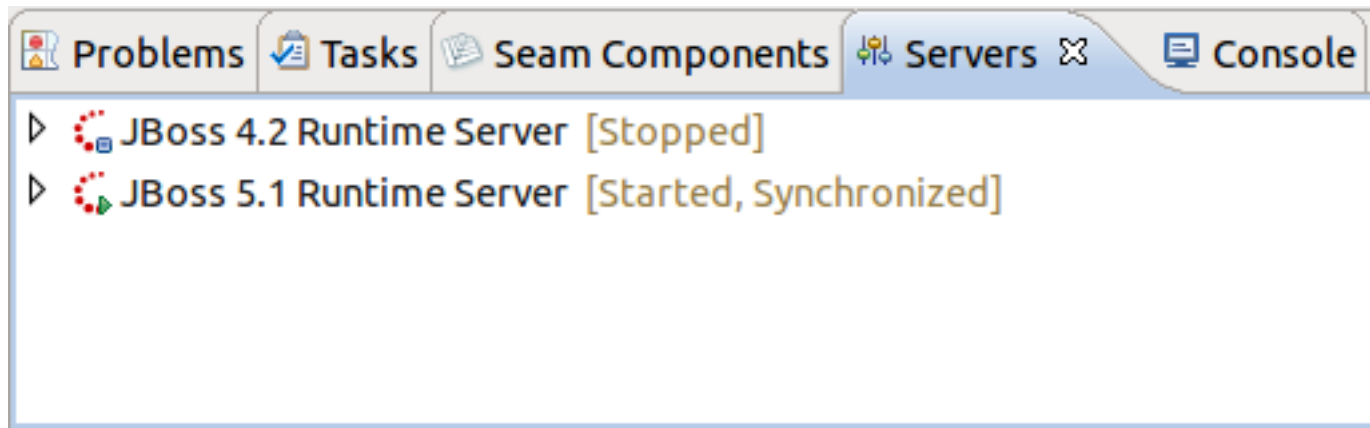


Figure 2.11. New JBoss Server

Now, we are ready to create the first web application.

Write Your First Project with JBoss Developer Studio

This chapter is a set of hands-on labs. You get step-by-step information about how JBoss Developer Studio can be used during the development process.

3.1. Create a Seam Application

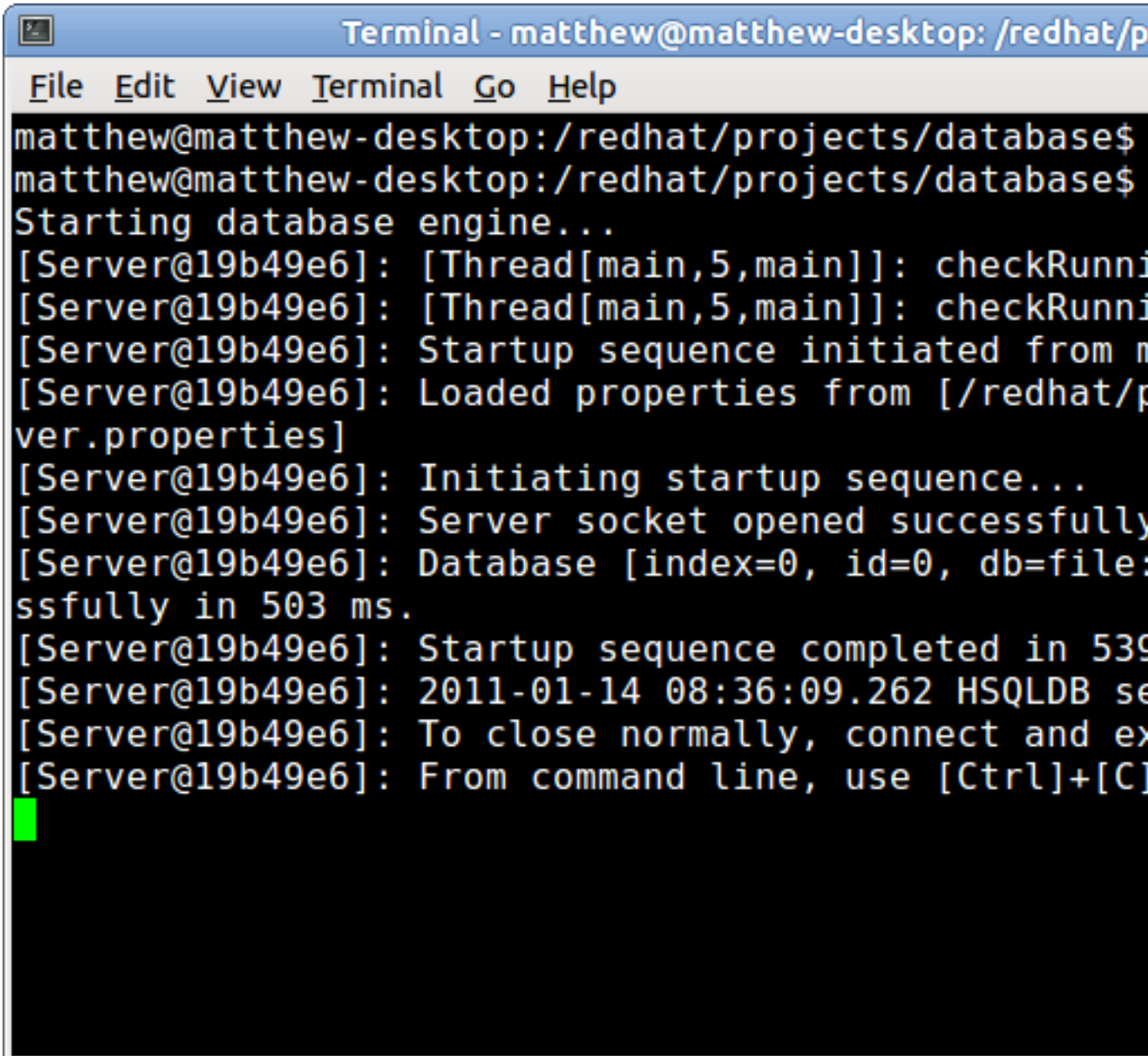
In this section you will learn how to create a Seam project in JBoss Developer Studio, how to start the server and what structure your project has after it is created.

3.1.1. Start Development Database

Before opening the JBoss Developer studio you need to download and start the [Workshop Database](http://docs.jboss.org/tools/resources/GSG_database.zip) [http://docs.jboss.org/tools/resources/GSG_database.zip] .

To start the database just run `./runDBServer.sh` or `runDBServer.bat` from the database directory.

The end result should be a console window that looks like:



The image shows a terminal window titled "Terminal - matthew@matthew-desktop: /redhat/p". The window has a menu bar with "File", "Edit", "View", "Terminal", "Go", and "Help". The terminal output shows the user at the prompt `matthew@matthew-desktop:/redhat/projects/databases$` typing `Starting database engine...`. The output then shows several status messages from the server, including thread information, startup sequence initiation, property loading, socket opening, and completion time. The terminal ends with a green cursor on a new line.

```
matthew@matthew-desktop:/redhat/projects/databases$
matthew@matthew-desktop:/redhat/projects/databases$
Starting database engine...
[Server@19b49e6]: [Thread[main,5,main]]: checkRunni
[Server@19b49e6]: [Thread[main,5,main]]: checkRunni
[Server@19b49e6]: Startup sequence initiated from m
[Server@19b49e6]: Loaded properties from [/redhat/p
ver.properties]
[Server@19b49e6]: Initiating startup sequence...
[Server@19b49e6]: Server socket opened successfully
[Server@19b49e6]: Database [index=0, id=0, db=file:
ssfully in 503 ms.
[Server@19b49e6]: Startup sequence completed in 539
[Server@19b49e6]: 2011-01-14 08:36:09.262 HSQldb se
[Server@19b49e6]: To close normally, connect and ex
[Server@19b49e6]: From command line, use [Ctrl]+[C]
```

Figure 3.1. Starting the Database

**Tip**

You may need to set the `runDBServer.sh` executable flag with the following command:

```
chmod +x runDBServer.sh
```

3.1.2. Create and deploy Seam Web Project

Minimize the terminal window and run JBoss Developer Studio from Applications Menu or from the desktop icon.

First you will see the Workspace Launcher. Change the default workspace location if it's needed. Click the **OK** button.

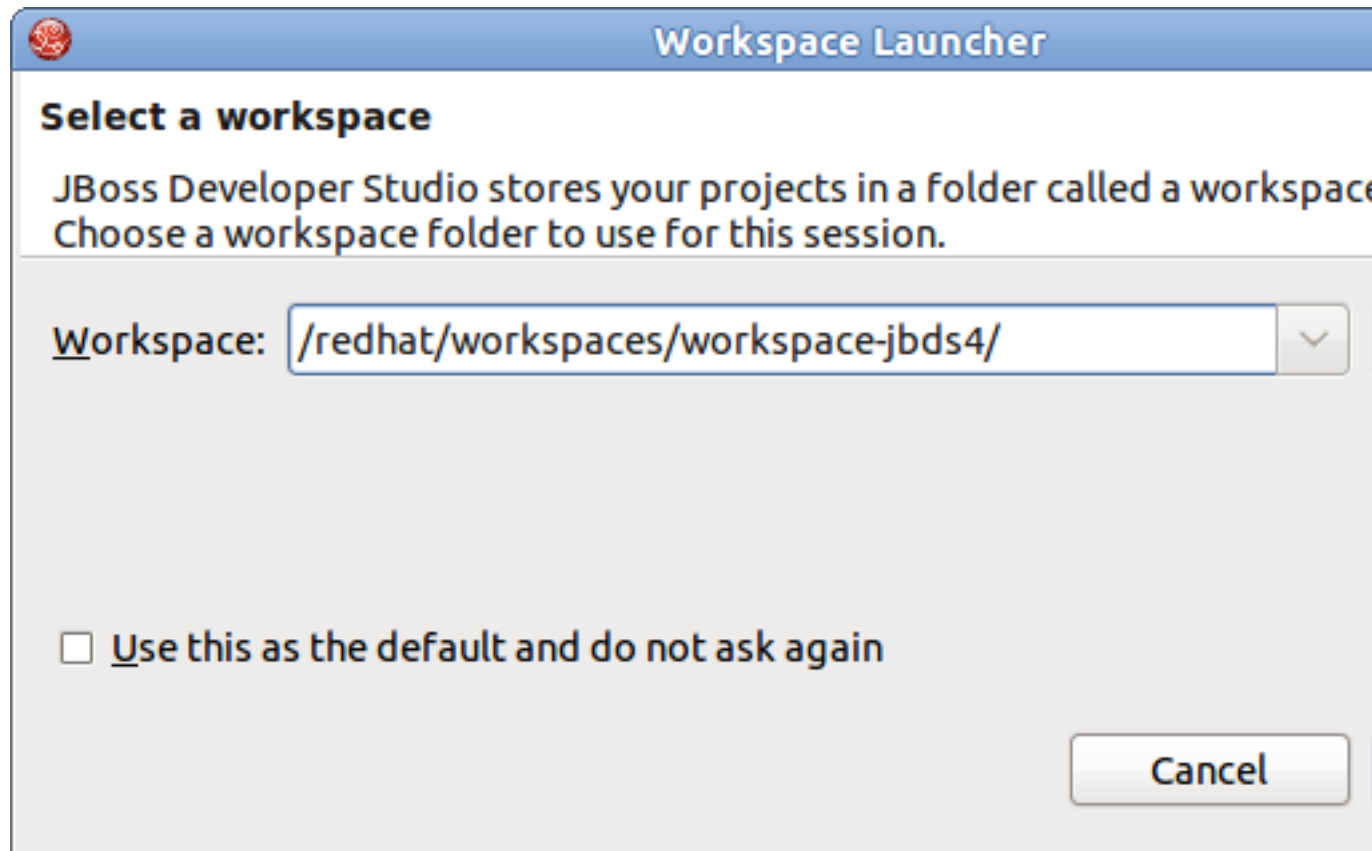


Figure 3.2. Workspace Launcher Dialog

After startup, you see the welcome page. You could read how to work with welcome pages in [previous](#) chapter. Now select **Create New...** icon and then press on **Create Seam Project** link.

The New Seam Project wizard is started. You need to enter a name (e.g., "workshop") and a location for your new project. The wizard has an option for selecting the actual Server (and not just WTP runtime) that will be used for the project. This allows the wizard to correctly identify where the destination folder for the required datasource and driver libraries.

New Seam Project

Seam Web Project

Create standalone Seam Web Project

Project name:

Project location

☒ Use default location

Location:

Target runtime

Dynamic web module version

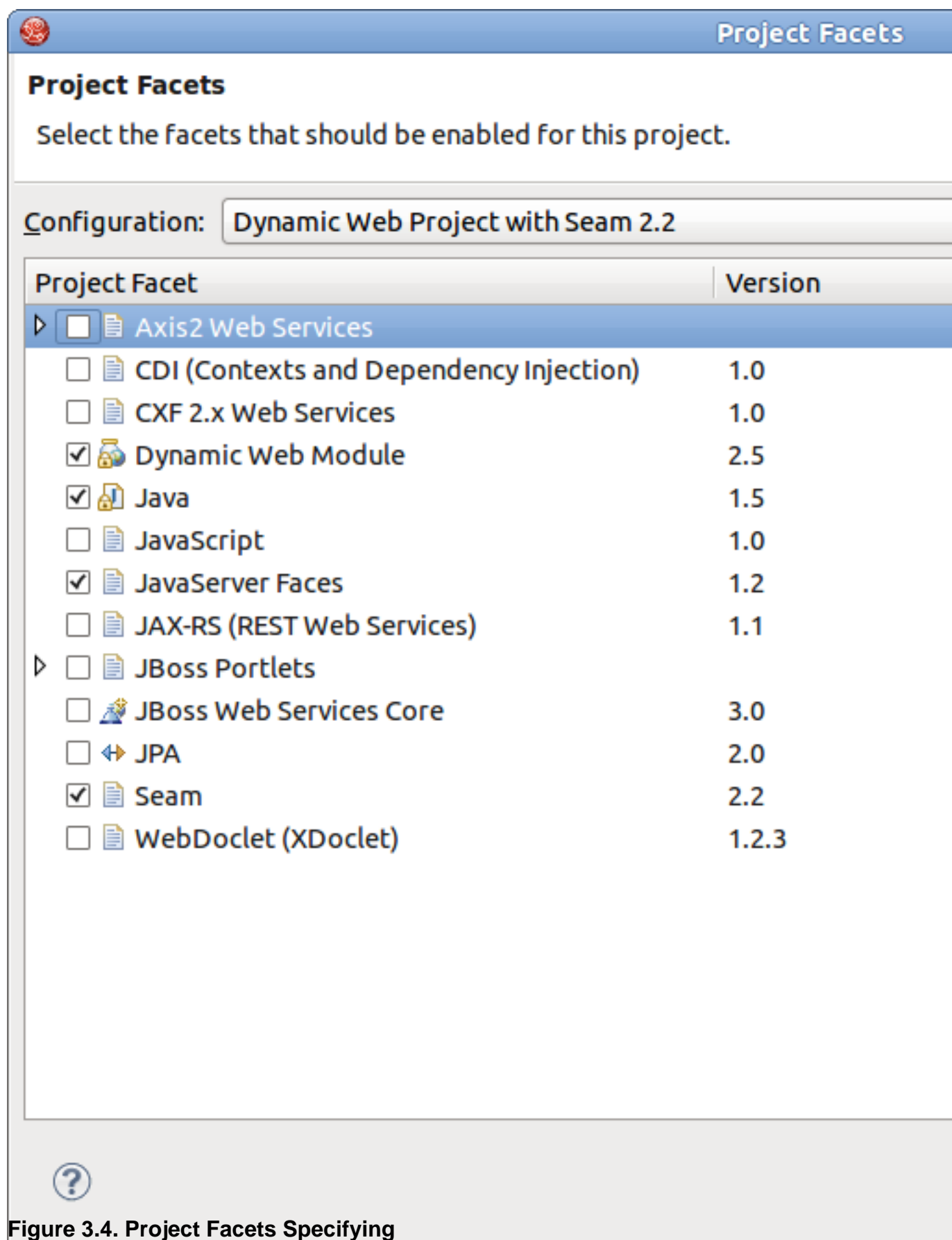
Target Server

Configuration

Configures a Dynamic Web application to use Seam v2.2

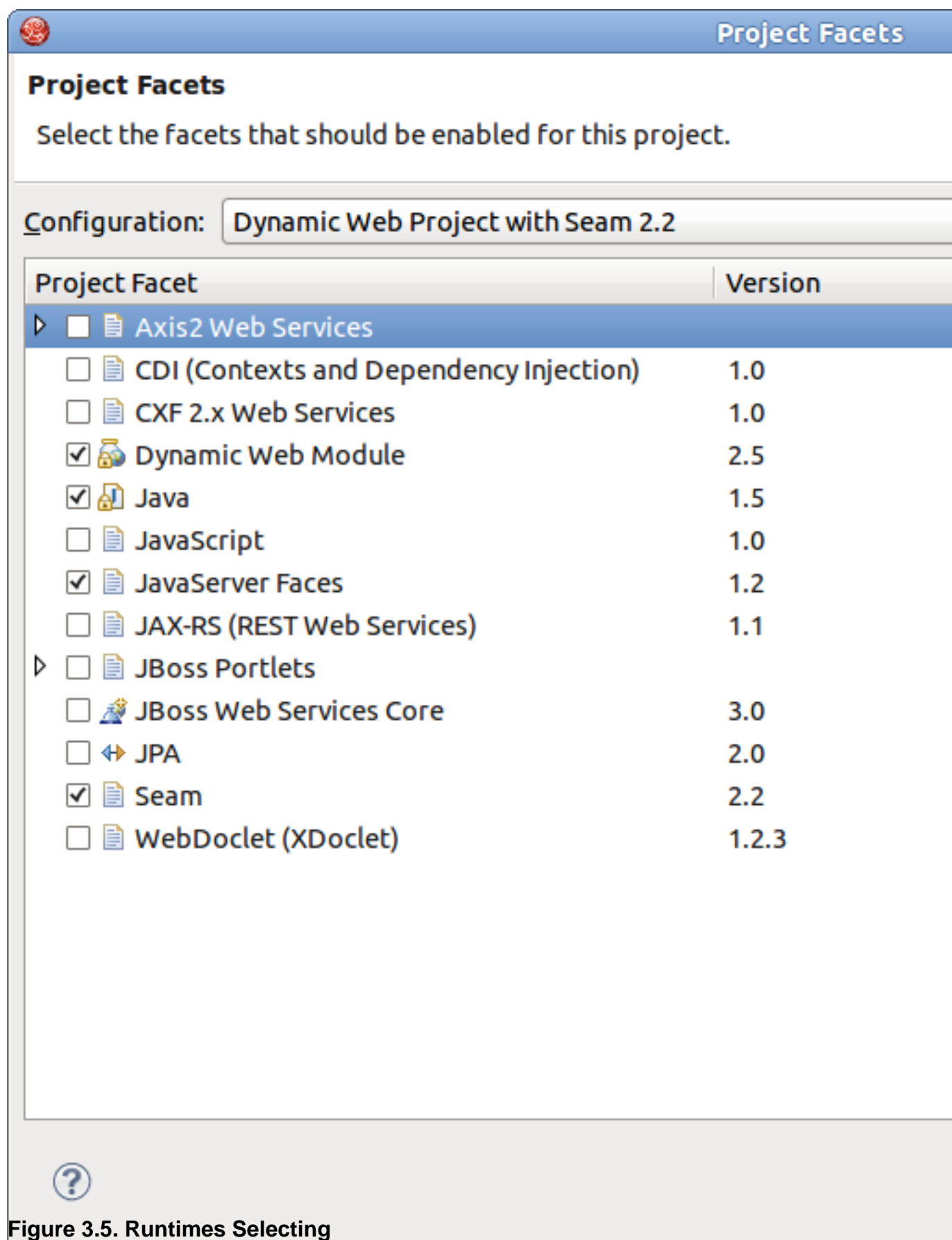
Figure 3.3. New Seam Project Wizard

All settings are already specified here, you can just modify the Configuration. Click on the **Modify...** button to configure your custom facet options:



On the whole the dialog allows to select the "features" you want to use in your project. JBoss Developer Studio will then setup the appropriate tooling for your project. Since JBoss Seam integrates all popular Java EE frameworks, you can select any combination of technologies from the list. Here, for the default configuration, Dynamic Web Module, Java, JavaServer Faces (JSF), and Seam Facet are already selected for a typical database-driven web application. The default project facets should suffice.

In the Project Facets form you can also bring up server runtimes panel by clicking Runtimes tab on the right corner. This panel shows available server runtimes.



Click the **OK** and then the **Next** button to proceed to the next step.

A dynamic web application contains both web pages and Java code. The next wizard will ask you where you want to store Java files.

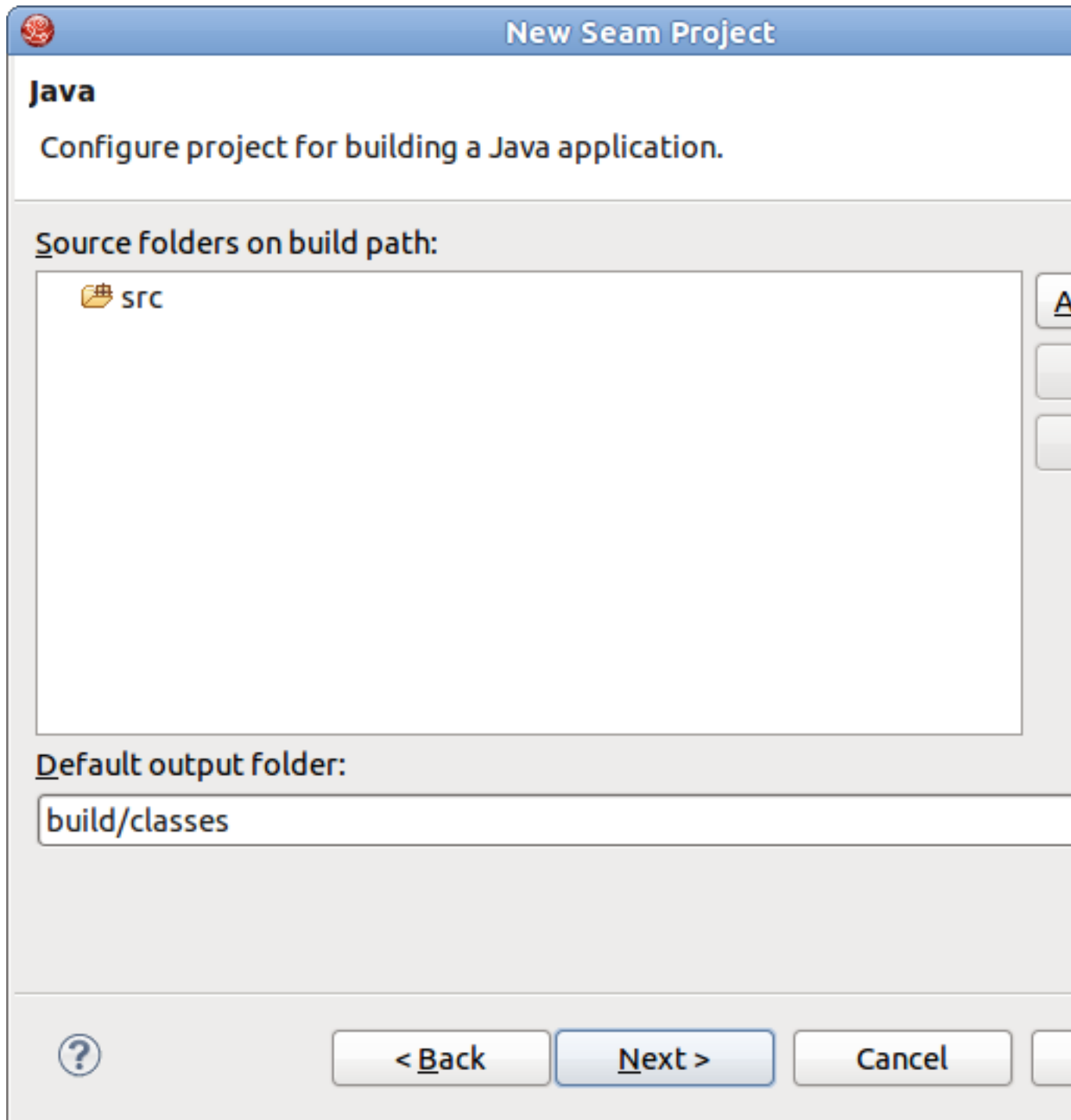
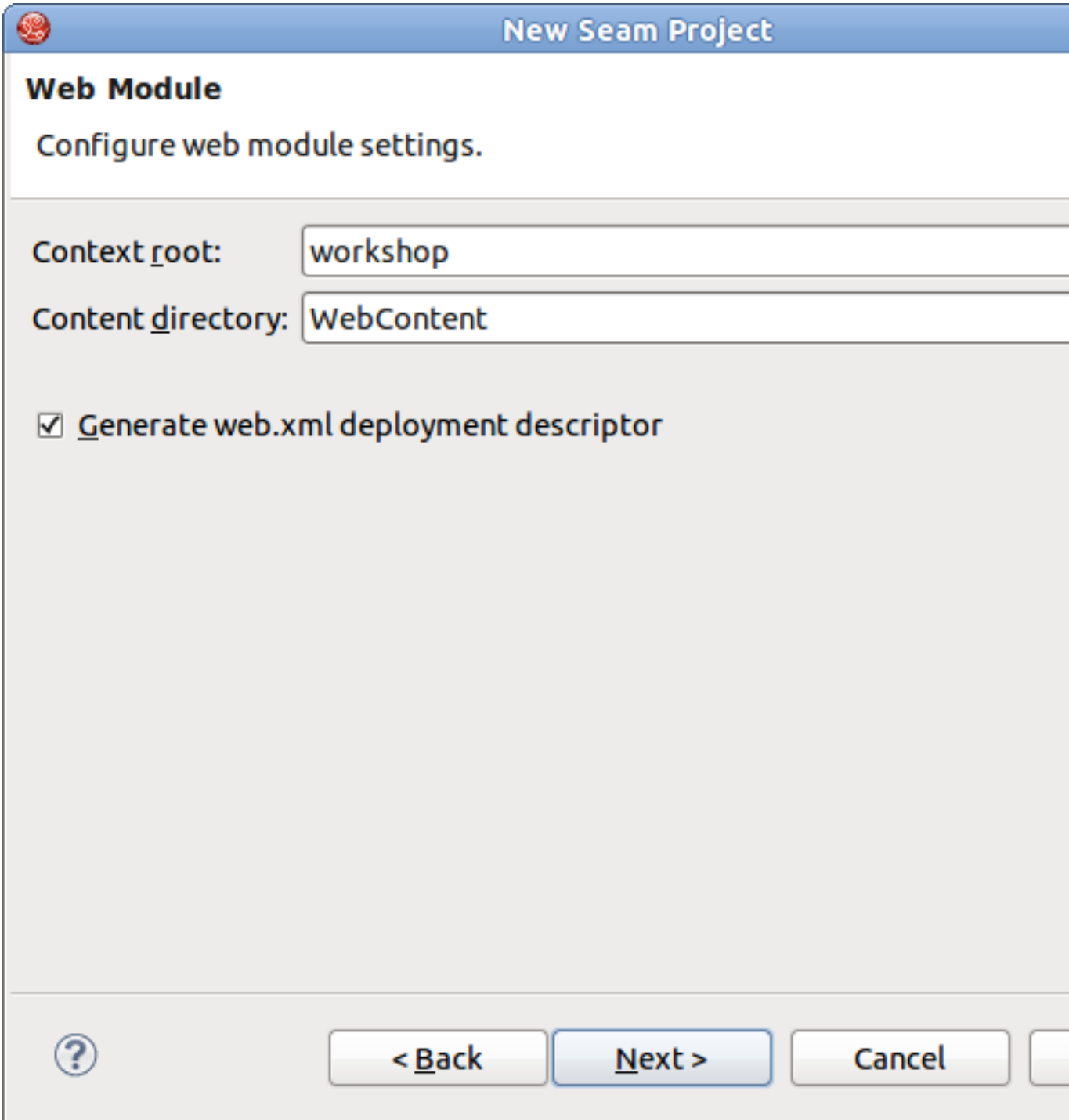


Figure 3.6. Java Build Path

Following page provides you Web Module Settings .You can just leave the default values or choose another folder.



New Seam Project

Web Module

Configure web module settings.

Context root: workshop

Content directory: WebContent

☒ Generate web.xml deployment descriptor

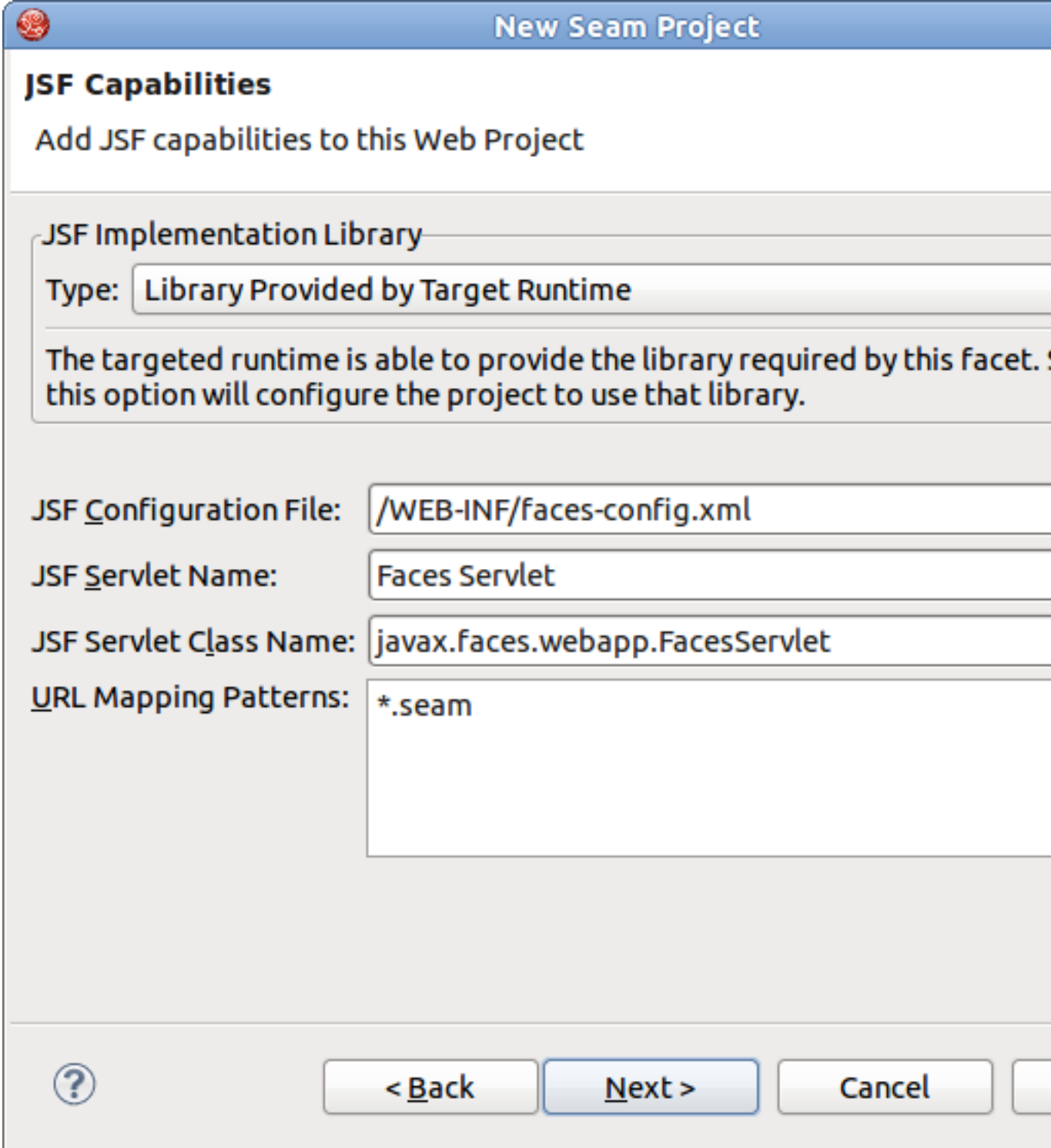
? < Back Next > Cancel

Figure 3.7. Web Module Settings

On the next form, you will be able to select where those library JARs come from. The easiest is just to select the JARs provided by the JBoss AS runtime associated with this project. That is why it is important to choose the right JBoss AS 4.2 runtime in the project setup window.

- Select *Library Provided by Target Runtime* as Type of JSF Implementation Library. We will use the JSF implementation that comes with JBoss server.

- Click the **Next** button



The screenshot shows the 'New Seam Project' wizard window. The title bar says 'New Seam Project'. The main section is titled 'JSF Capabilities' with the subtitle 'Add JSF capabilities to this Web Project'. Below this is a section for 'JSF Implementation Library' where the 'Type' is set to 'Library Provided by Target Runtime'. A message states: 'The targeted runtime is able to provide the library required by this facet. Selecting this option will configure the project to use that library.' Below the message are four input fields: 'JSF Configuration File' with the value '/WEB-INF/faces-config.xml', 'JSF Servlet Name' with the value 'Faces Servlet', 'JSF Servlet Class Name' with the value 'javax.faces.webapp.FacesServlet', and 'URL Mapping Patterns' with the value '*.seam'. At the bottom, there is a help icon (question mark in a circle) and three buttons: '< Back', 'Next >', and 'Cancel'.

JSF Capabilities
Add JSF capabilities to this Web Project

JSF Implementation Library
Type:

The targeted runtime is able to provide the library required by this facet. Selecting this option will configure the project to use that library.

JSF Configuration File:

JSF Servlet Name:

JSF Servlet Class Name:

URL Mapping Patterns:




Figure 3.8. JSF Capabilities Adding

Next wizard step needs more settings than previous. Let's start with General section.

Leave the default Seam runtime and check a WAR deployment.

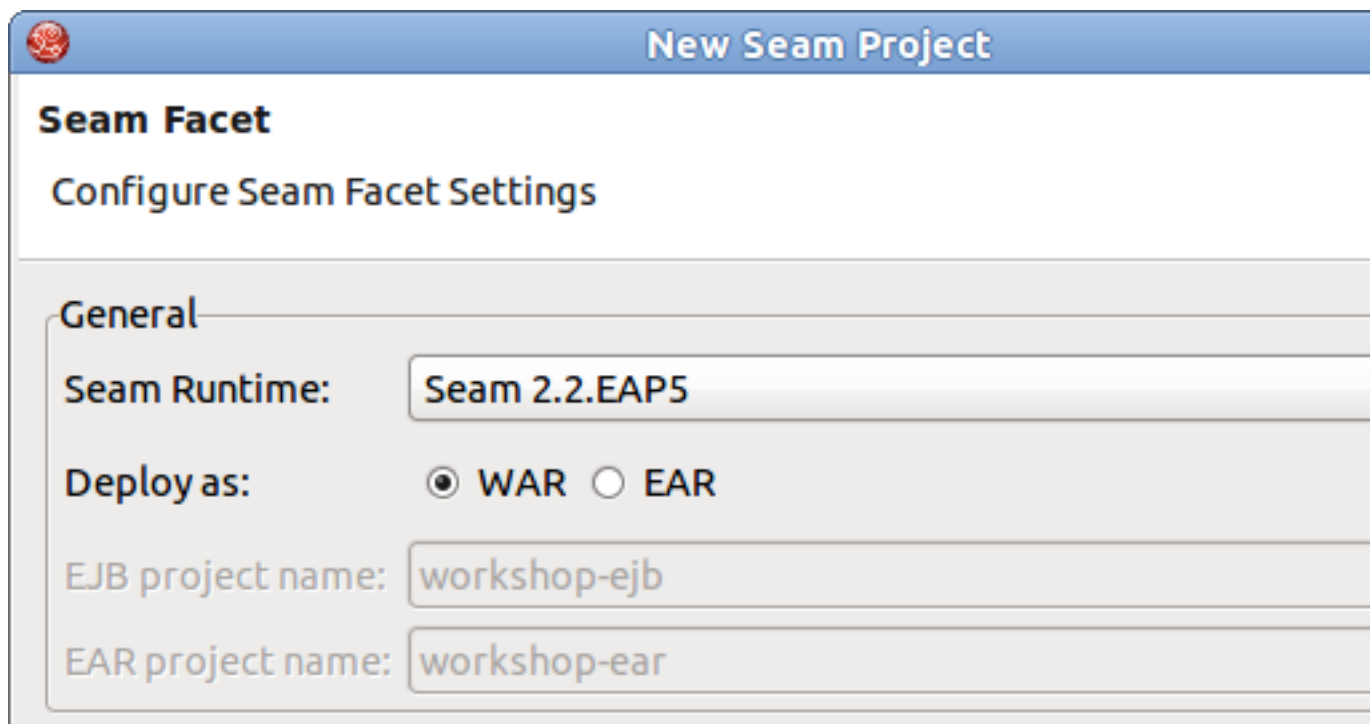


Figure 3.9. Seam Facet Setting

The Database section is a little tricky. The Connection Profile needs to be edited so that the new project works properly with the external HSQLDB server. By default the project wizard tries to use the JBoss embedded HSQLDB, but the tutorial uses an external database to replicate a more real world development scenario. Click on the **Edit** button to modify the Connection Profile.

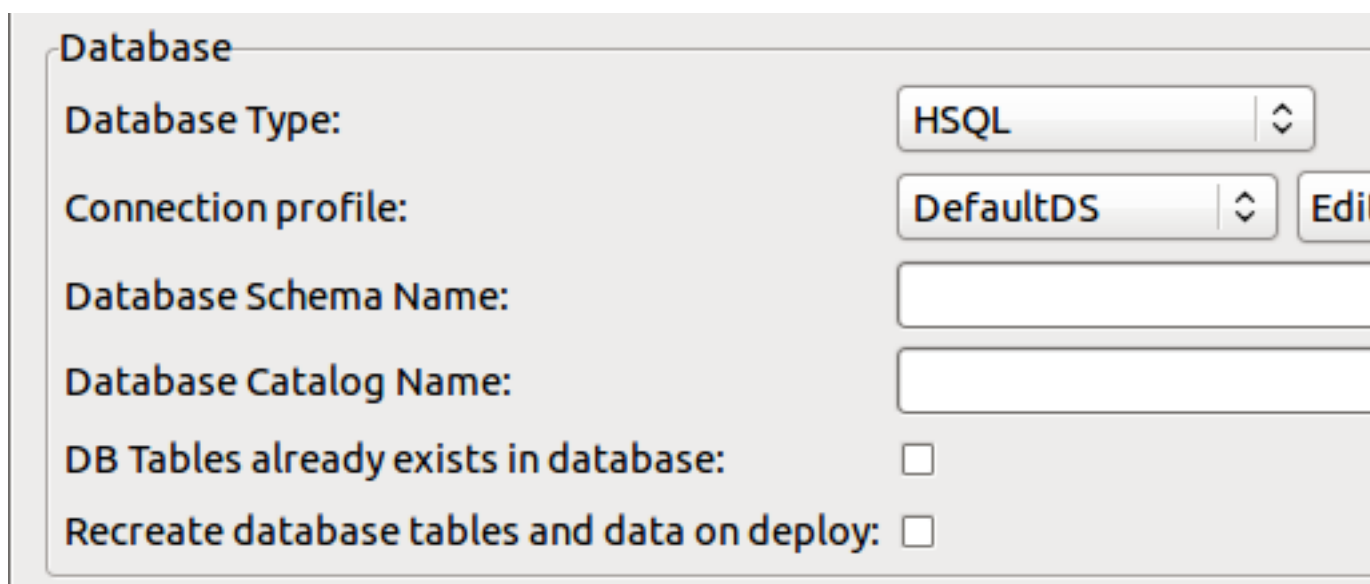


Figure 3.10. DataBase Setting

Select HSQLDB Profile Properties. Make sure the Database location is set to `hsql://localhost:1701`

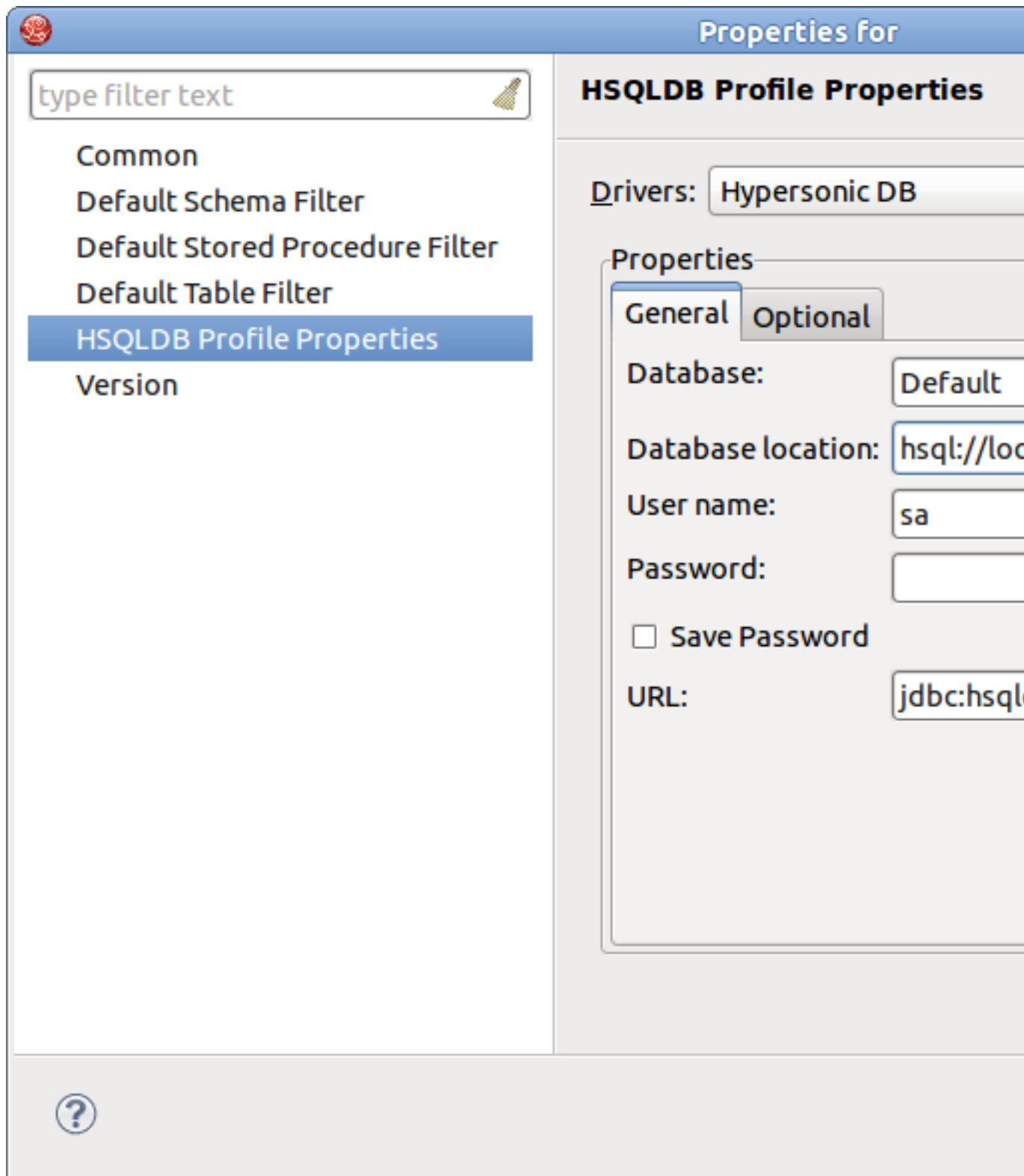


Figure 3.11. JDBC Connection Properties

Click the **Test Connection** button. At this point it probably won't work. This happens if the HSQL JDBC driver is not exactly the same. This can be solved by modifying the HSQLDB database driver settings. To modify the settings, click the **Edit Driver Definition Driver** button.

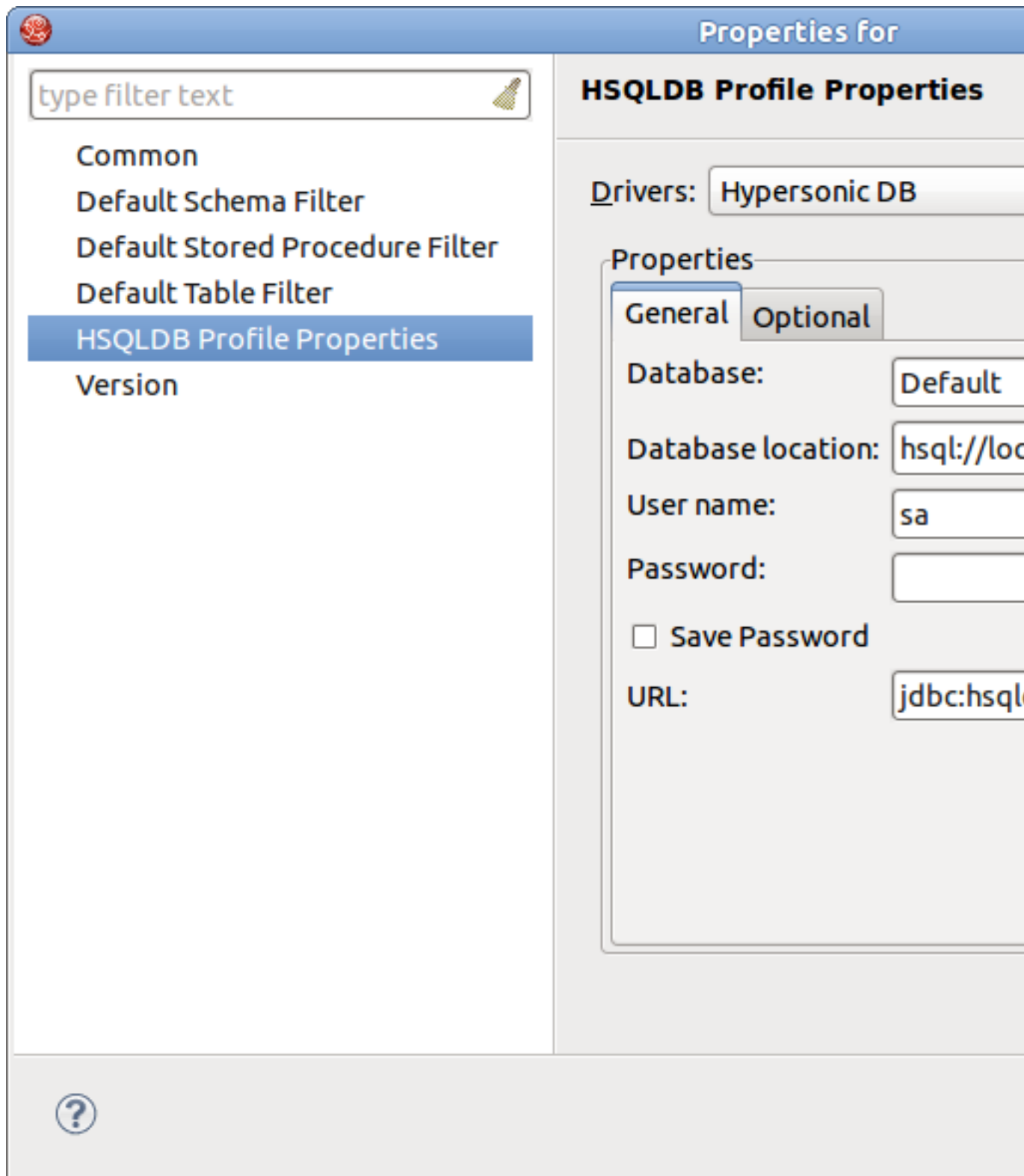


Figure 3.12. Driver Details

The proper Driver JAR File should be listed under Jar List. Select the `hsqldb.jar` file found in the `jbdevstudio/jboss-eap/jboss-as/common/lib/` directory and click the **OK** button.

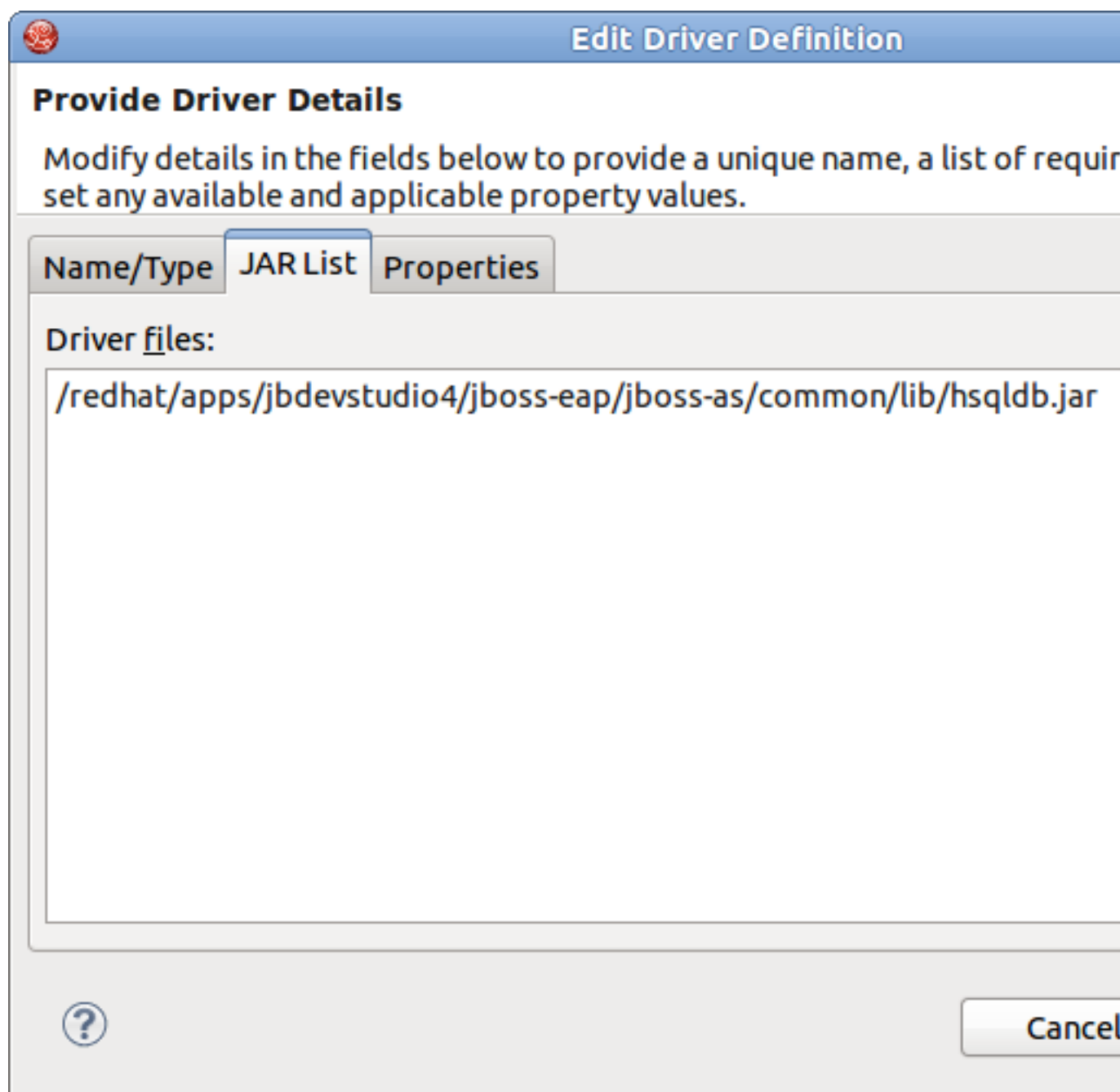


Figure 3.13. Driver Details

Now, the **Test Connection** should succeed. After testing the connection, click the **OK** button.

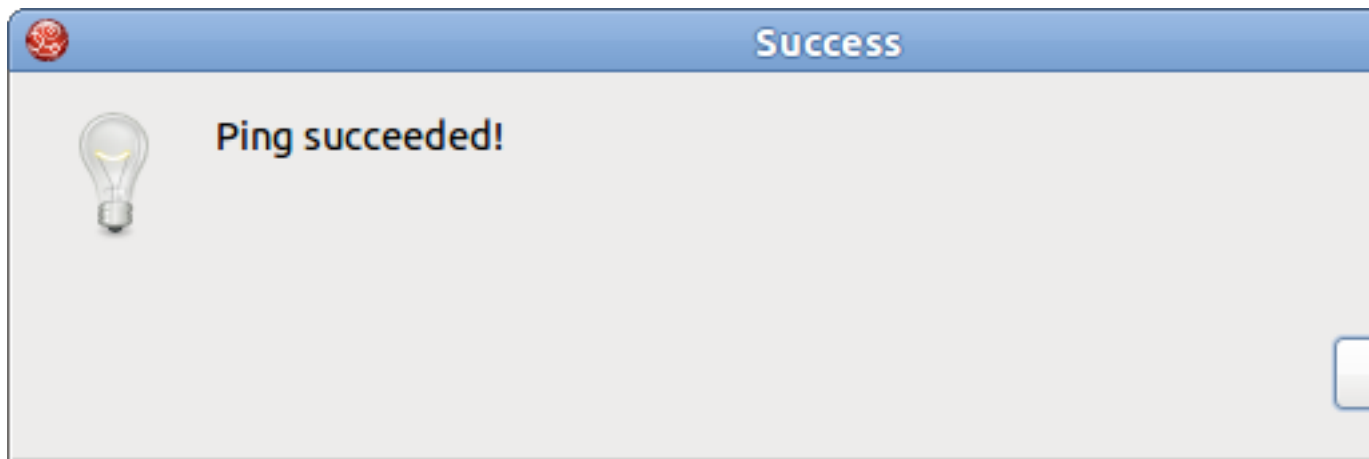


Figure 3.14. Connection Testing

You can leave the Code Generation section as is. It refers to Java packages in which the generated code will be placed.

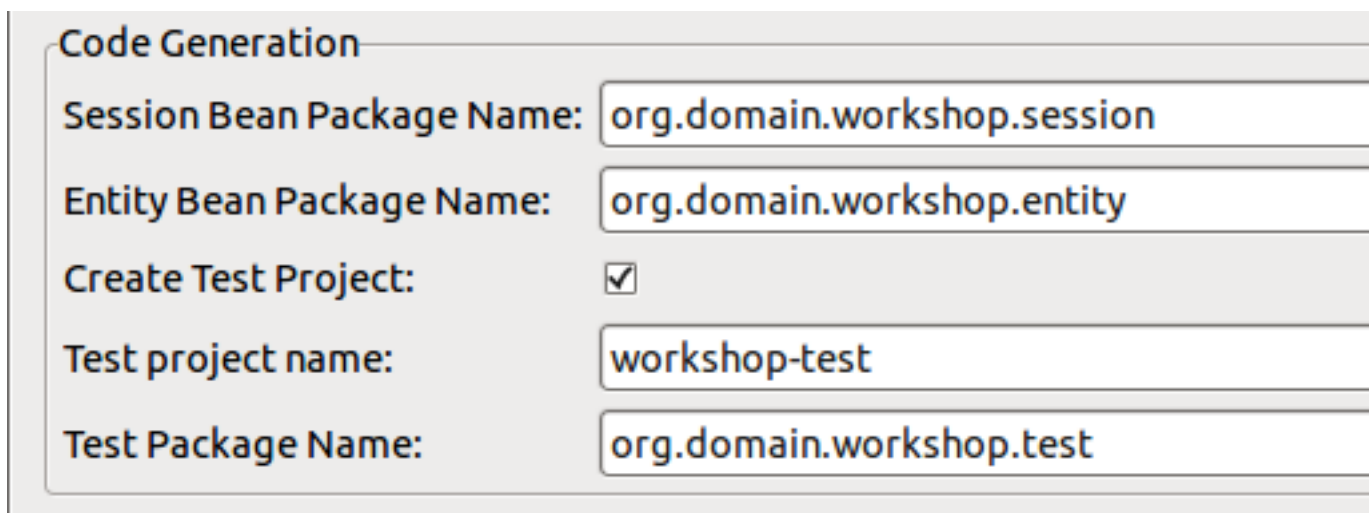
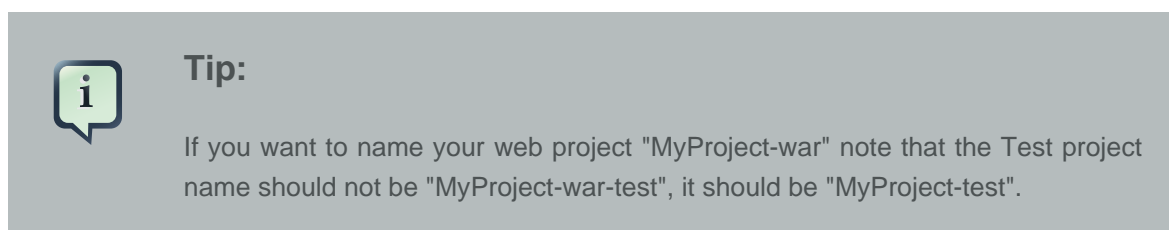


Figure 3.15. Code Generation Setting



Click on **Finish** button. Now, there should be a new Seam project called “workshop” listed in the Package Explorer view.

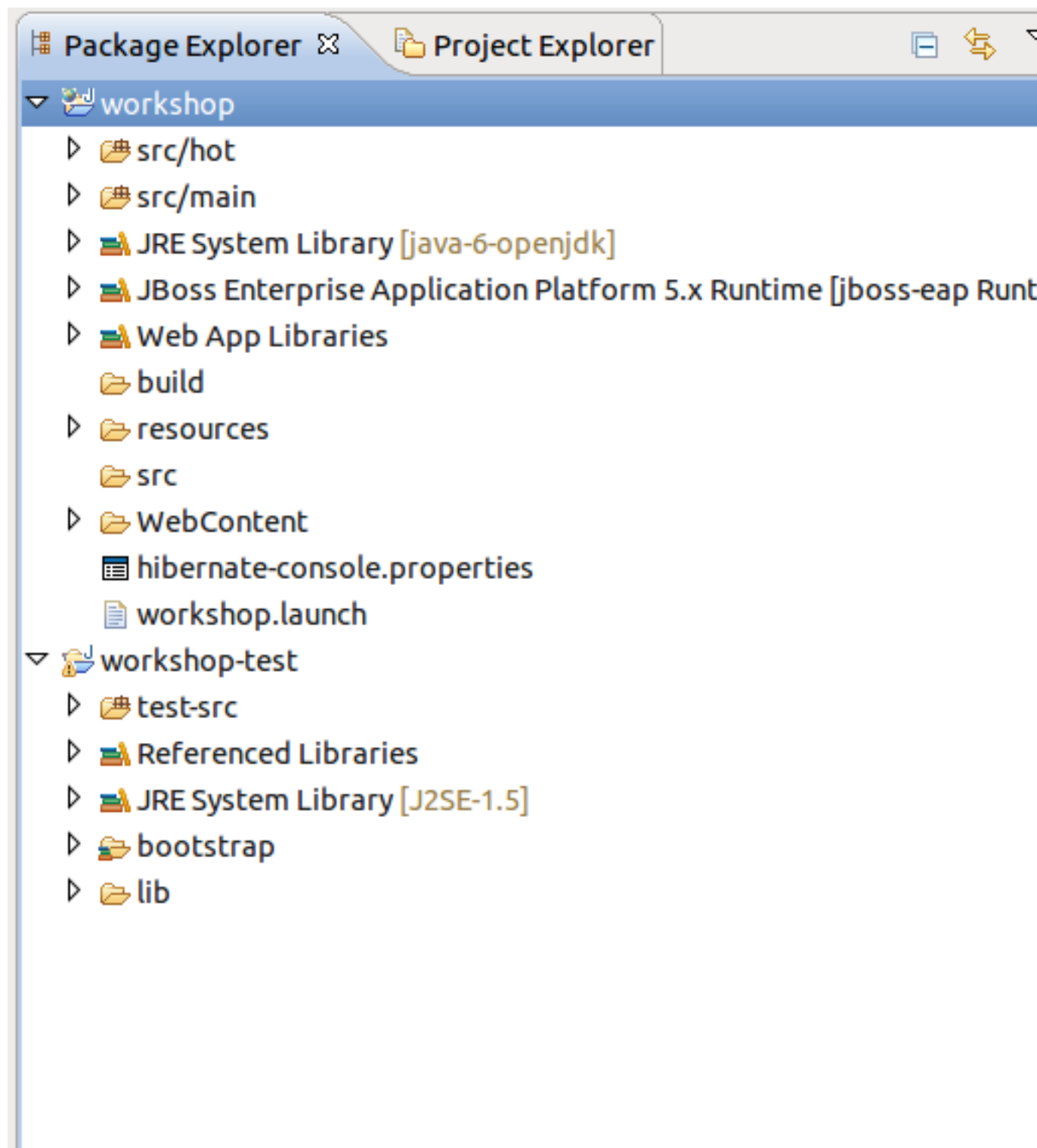



Figure 3.16. "workshop" Project in the Package Explorer

3.1.3. Start JBoss Application Server

The complete information on how to manage JBoss AS from JBoss Developer Studio can be found in the [corresponding chapter](#).

Now you just need to start the server by clicking on the Start the server icon () in the Servers view.

Then run the project by selecting the project then selecting **Run As... → Run on Server**.

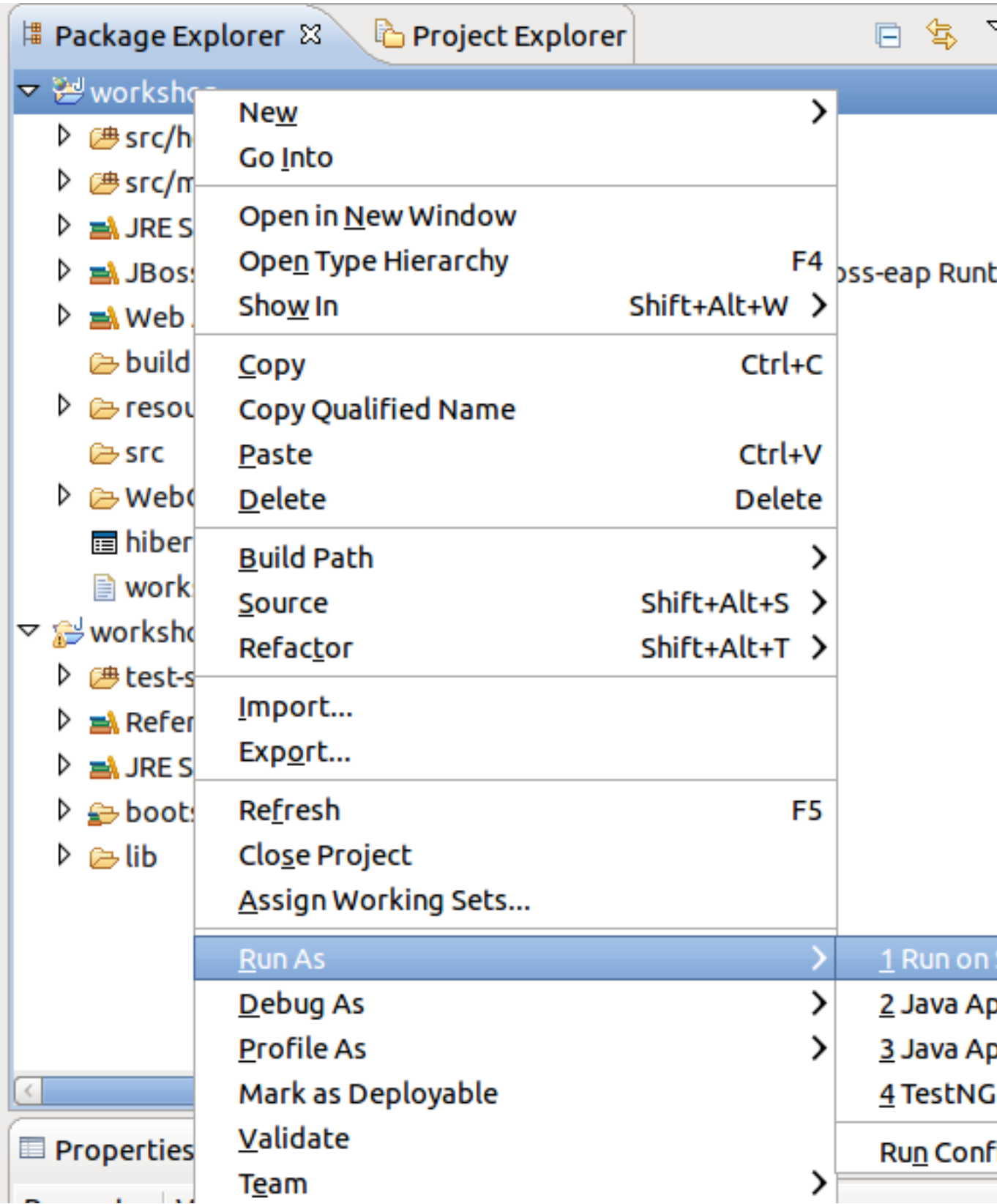


Figure 3.17. "workshop" Run As

Select the server you want to run the project on, and click the **Finish** button.

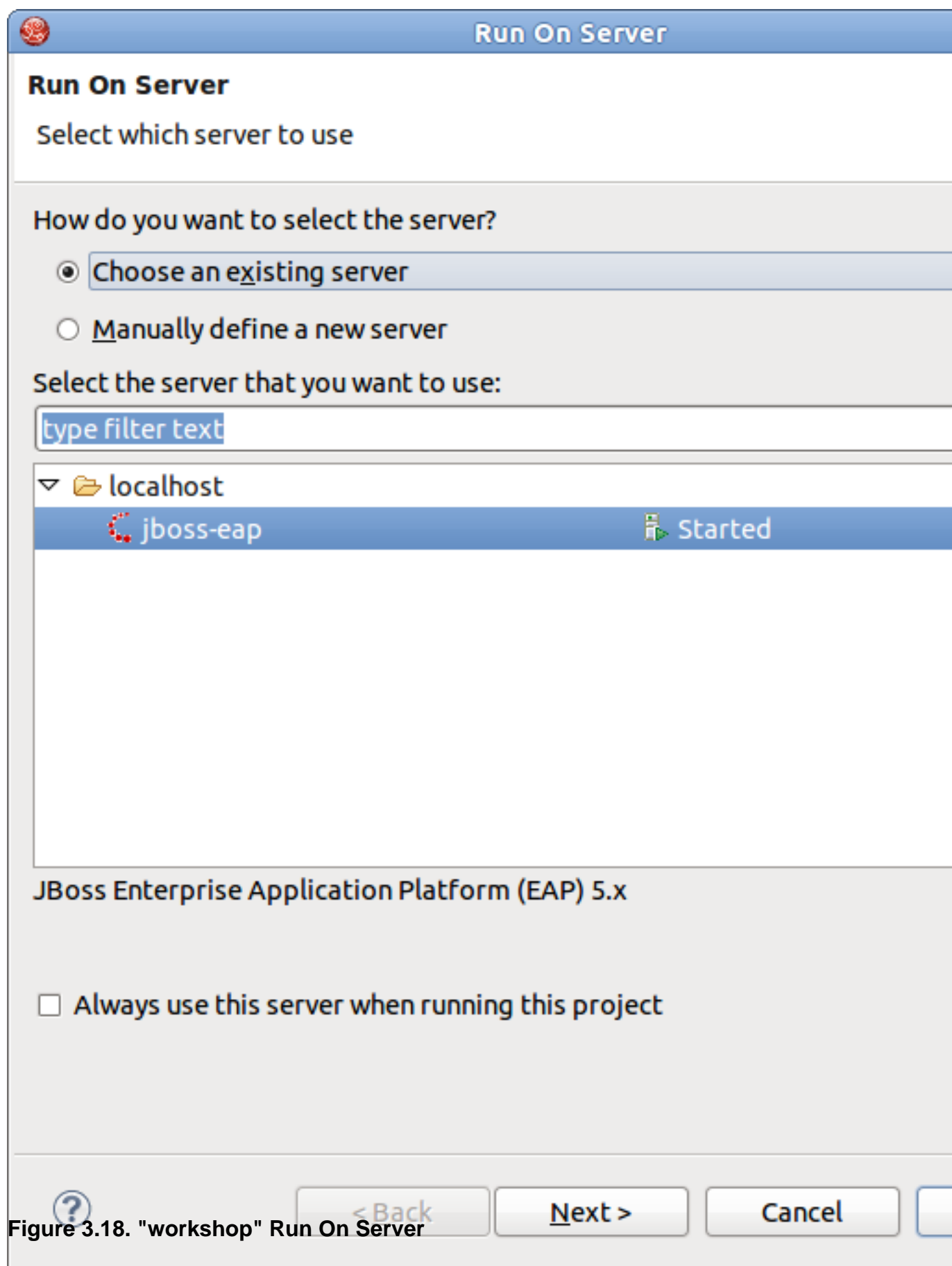


Figure 3.18. "workshop" Run On Server



Note:

If the project does not show up, then you can use a normal browser and use *<http://localhost:8080/workshop/home.seam>* as the URL.

Your project looks like this:

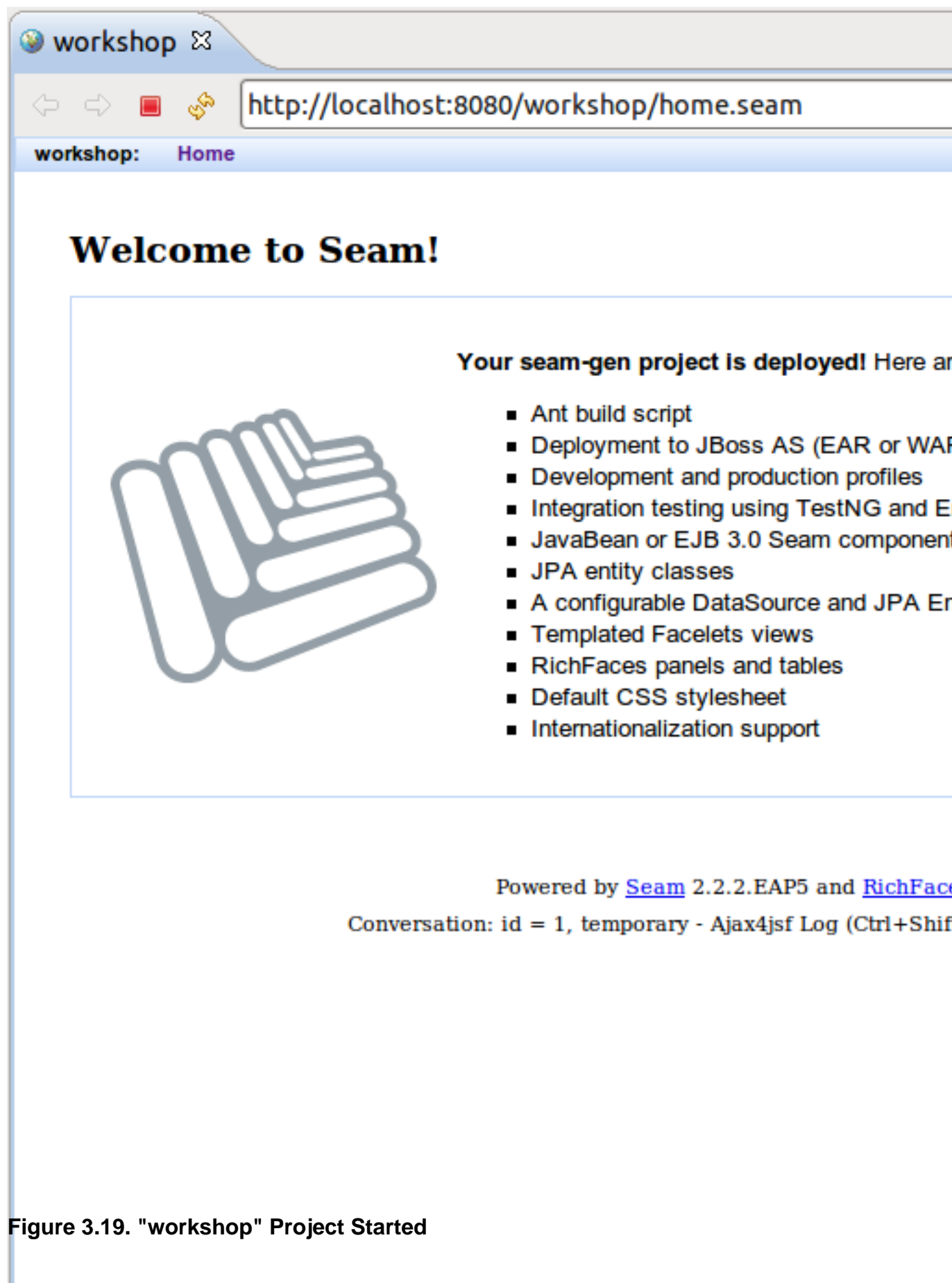


Figure 3.19. "workshop" Project Started

3.1.4. Workshop Project Code Overview

Now let's examine the project and its structure. Go back to the Package Explorer view in JBoss Developer Studio.

It seems like it's not much for a project but this shell application contains a login screen with default login logic, a menu template that can be further modified, and other layout templates.

It's important to note that the business logic will reside in the `src/hot` folder, by default. And, the package naming conventions that were used in New Seam project wizard could have been changed to something different from `org.domain.workshop.session`. Also, notice that there is a default `Authenticator.java` file. This is where custom security logic can be added. Seam has a nice declarative security model that we will explore in more detail later on. The `src/main` folder is a model directory. It stores the project's JPA entity beans.

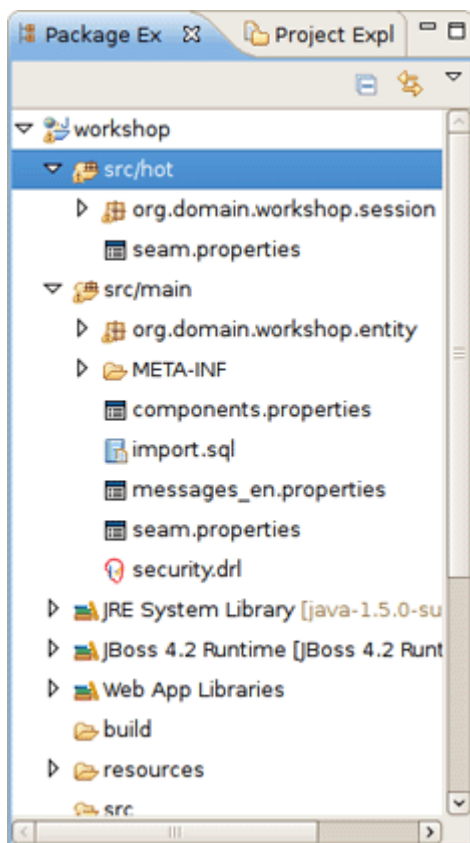


Figure 3.20. Project Structure

The view tier of the application is also important. Seam uses facelets and there is a built-in facelets GUI editor that includes nice WYSIWYG and component drag/drop functionality. Try this out by opening `home.xhtml` from `WebContent` folder.

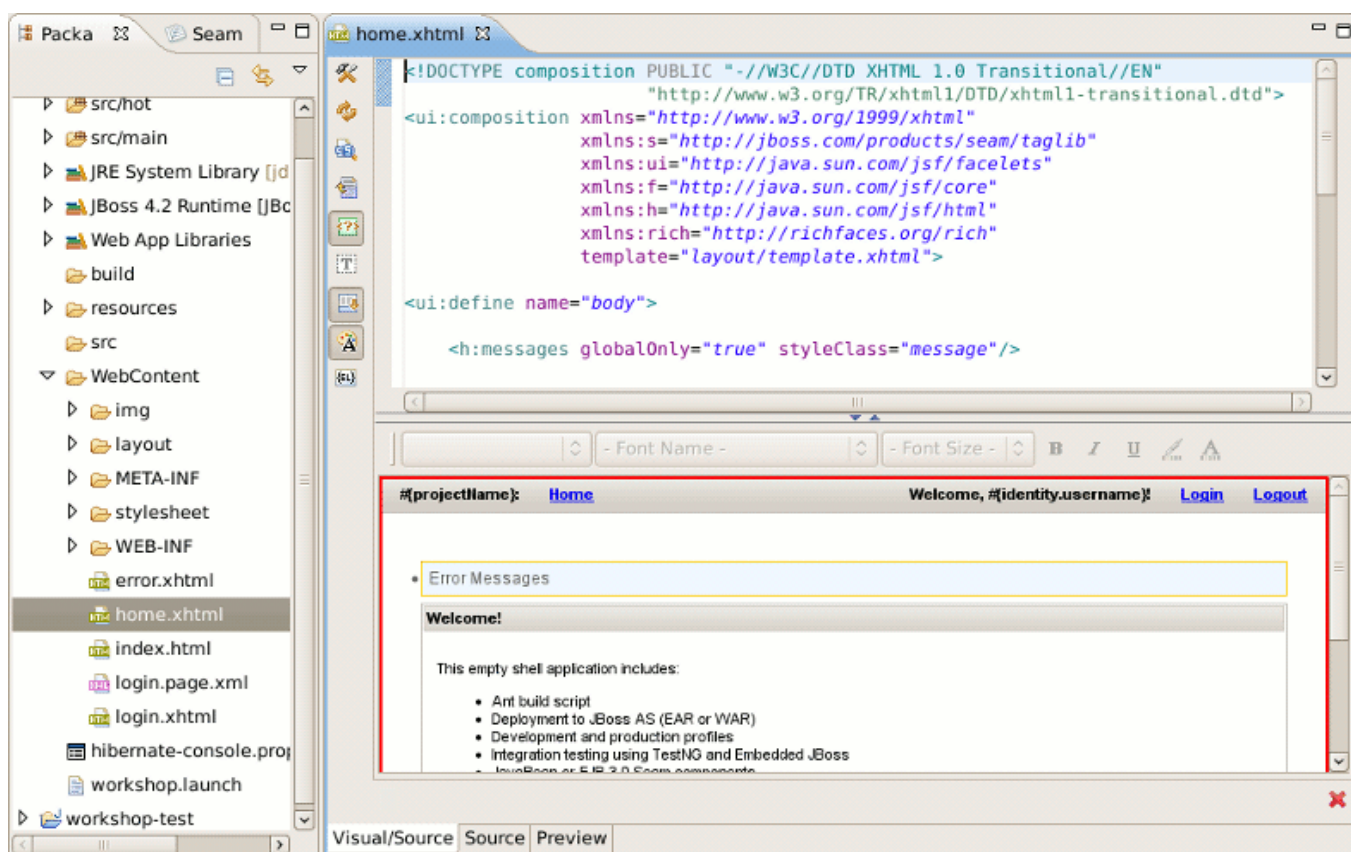


Figure 3.21. Facelets GUI Editor

Notice that the templates reside in the `WebContent/layout` folder. There is a stylesheet in the `WebContent/stylesheet` folder. There is also a login and default error page. The Facelet editor will be explored in more detail later in the lab.

The project already has a datasource that was created via the Seam project wizard database settings. All of the Seam specific configuration files and JAR dependencies are included and located in their proper locations. On last noteworthy line item is related to the build script. There isn't a build script because the Eclipse WTP (Web Tools Project) plugin is used to publish web application changes. As you can see, JBoss Developer Studio is removing a great deal of complexity from the enterprise Java project setup and deployment process. The end result is the developer is writing code, not spending time trying to figure out how to get a decent development environment and project build process.

3.2. Seam Action Development

Now it's time to write some code. The good news is that JBoss Developer Studio can also help out in this respect. In this section, we will create a new Seam Action POJO and facelet with some custom business logic and some GUI changes.

3.2.1. Create a New Seam Action

Go to main menu bar and click on **File** → **New** → **New Seam Action** to start the New Seam Action wizard.

Specify a Seam component name (e.g., "myAction"). The other properties will be auto-completed for you so there is no need to change them. Click on the **Finish** button.

New Seam Action

Seam Action

Create a new Seam action

Seam Project: workshop Browse...

Seam component name: myAction

POJO class name: MyAction

Bean name: MyActionBean

Method name: myAction

Page name: myAction

Package name: org.domain.workshop.session

?

Cancel

Figure 3.22. New Seam Action Wizard

Now, open the `MyAction.java` file and replace the "myAction" method with this logic:

```
public void myAction() {  
    Calendar cal = Calendar.getInstance();  
    log.info("myAction.myAction() action called");  
    statusMessages.add("MyAction Executed on:" + cal.getTime());  
}
```

You also need to import the `java.util.Calendar` class by clicking **CTRL+Shift+O**.

3.2.2. Test Seam Action

The new action can be tested by browsing the workshop-test project. JBoss Developer Studio has already created a TestNG test case for you.

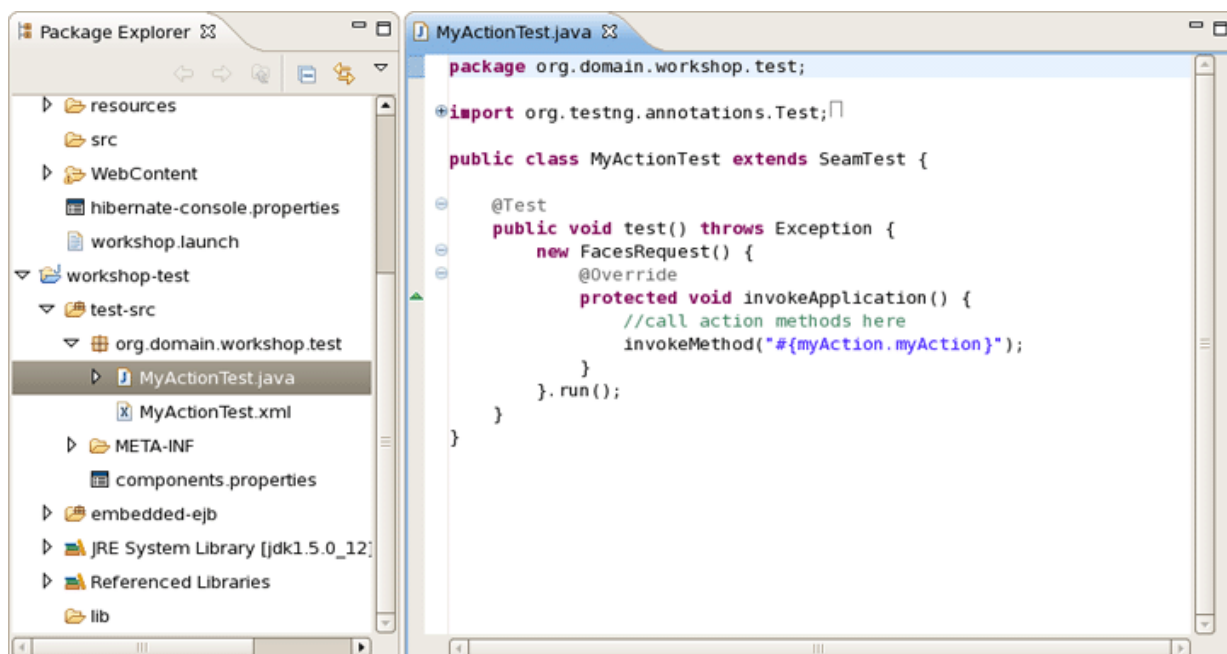


Figure 3.23. "workshop-test" Project



Tip

You may have to refresh the project to see the new files.

The test case simulates a Seam method execution for the `MyAction.myAction()` logic.

To run the test case, right click on `MyActionTest.xml` and click **Run As** → **TestNG Suite** or use the **Run As...** toolbar shortcut as shown below.

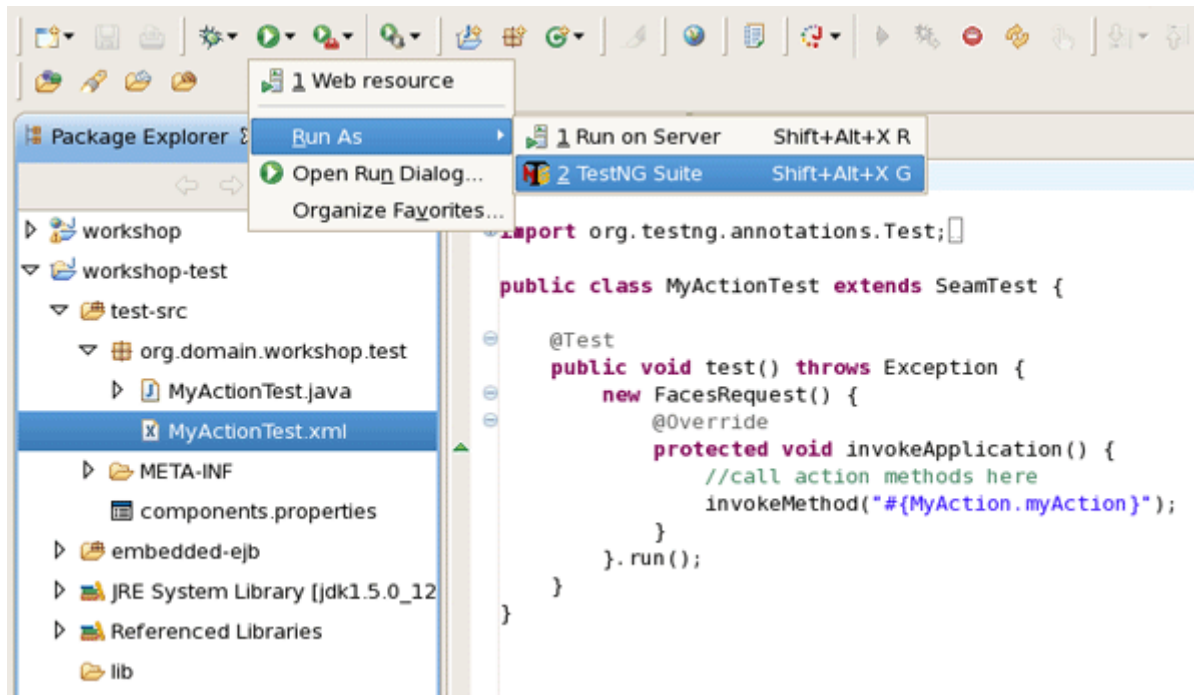


Figure 3.24. TestNG Running

With any luck, the test case will pass. Look at the TestNG view.

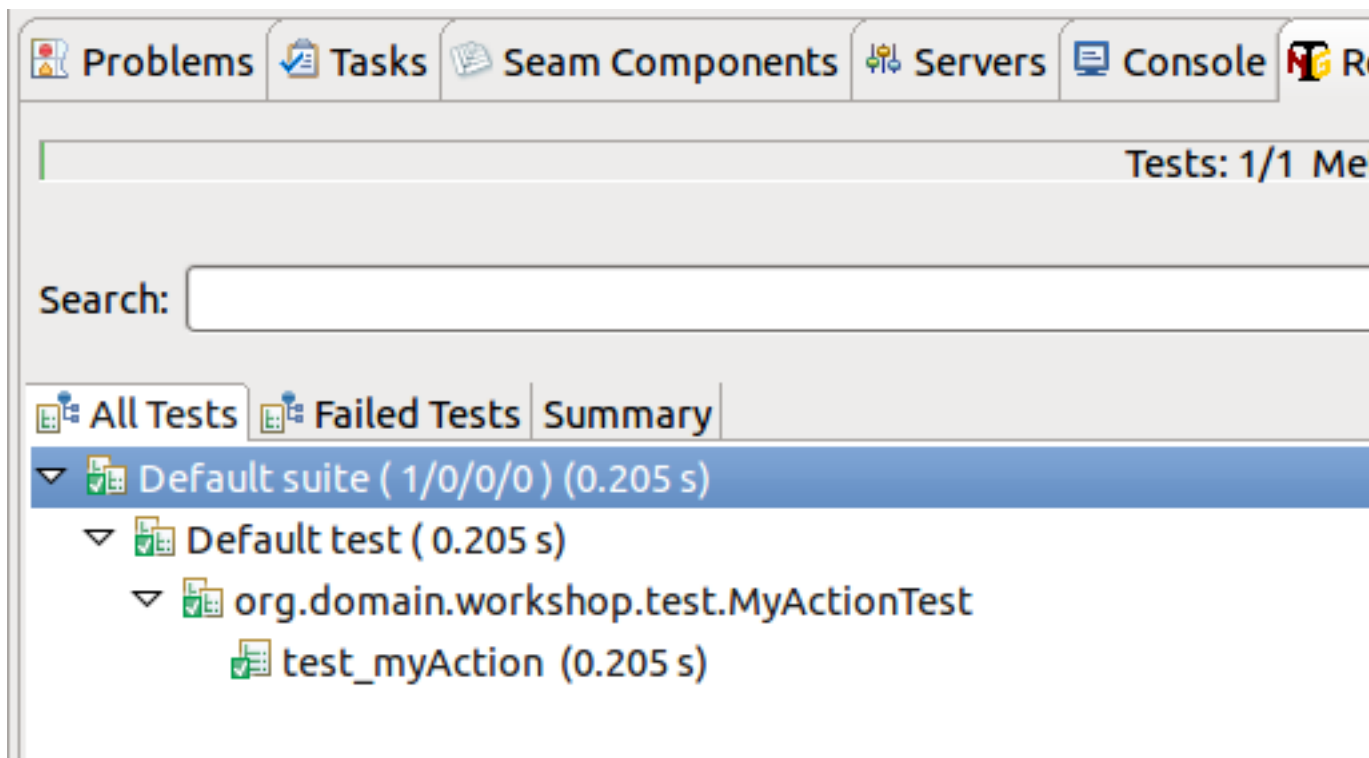


Figure 3.25. TestNG Results

Now, it's safe to test the new Seam Action in a web browser. The fastest way to do that is to right click on `myAction.xhtml` and use **Run As... → Run On Server** which will show the appropriate URL in the browser. Alternatively you can manually enter `http://localhost:8080/workshop/myAction.seam` into a browser.

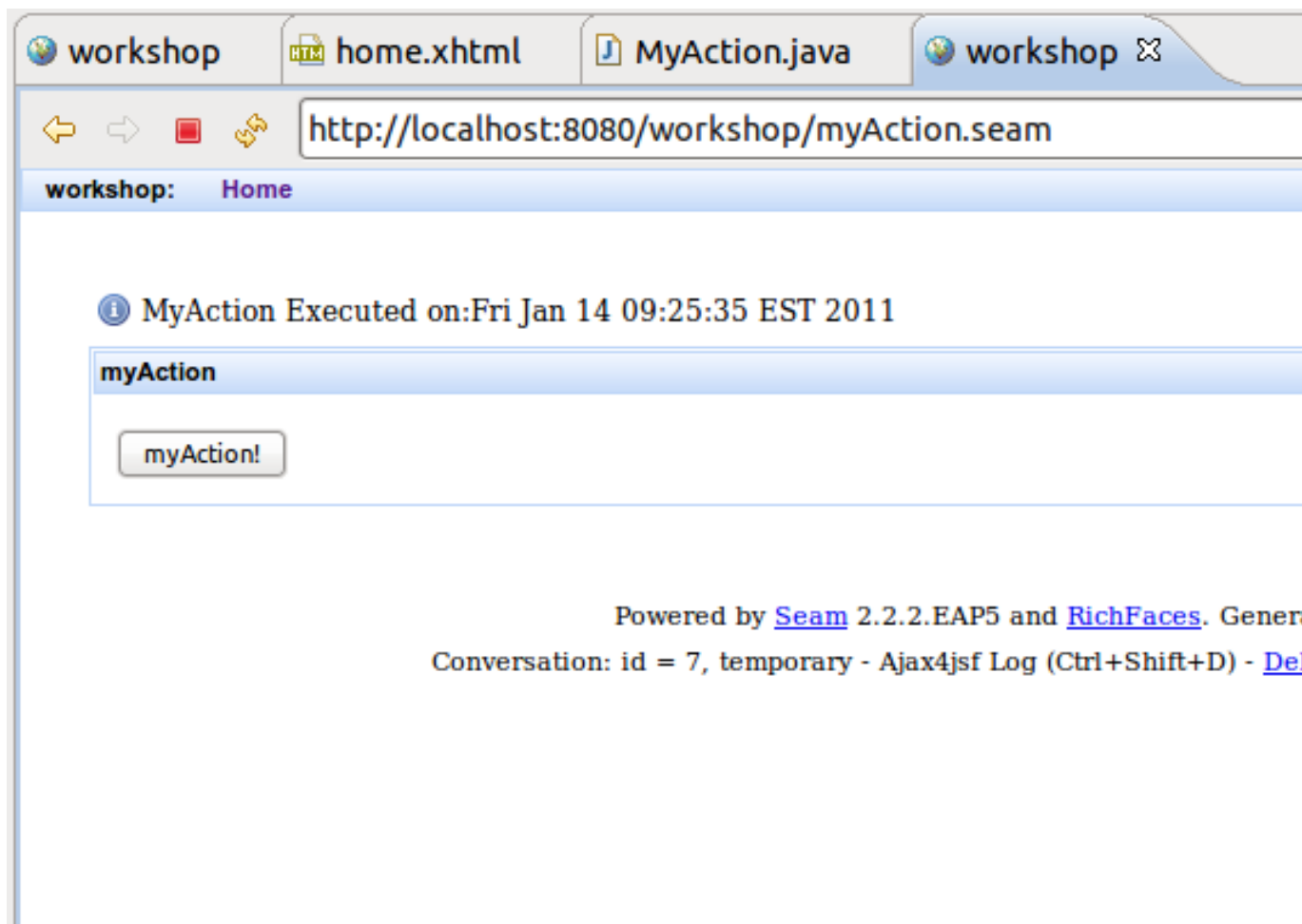


Figure 3.26. Seam Action in a Web Browser

3.2.3. Modify Seam Action User Interface

Browse to `http://localhost:8080/workshop/myAction.seam` and click on the **myAction** button. This executes the “myAction” method. This looks pretty good, but we could make this page look a little better.

Open `WebContent/myAction.xhtml` in JBoss Developer Studio to use the nice facelets editor.

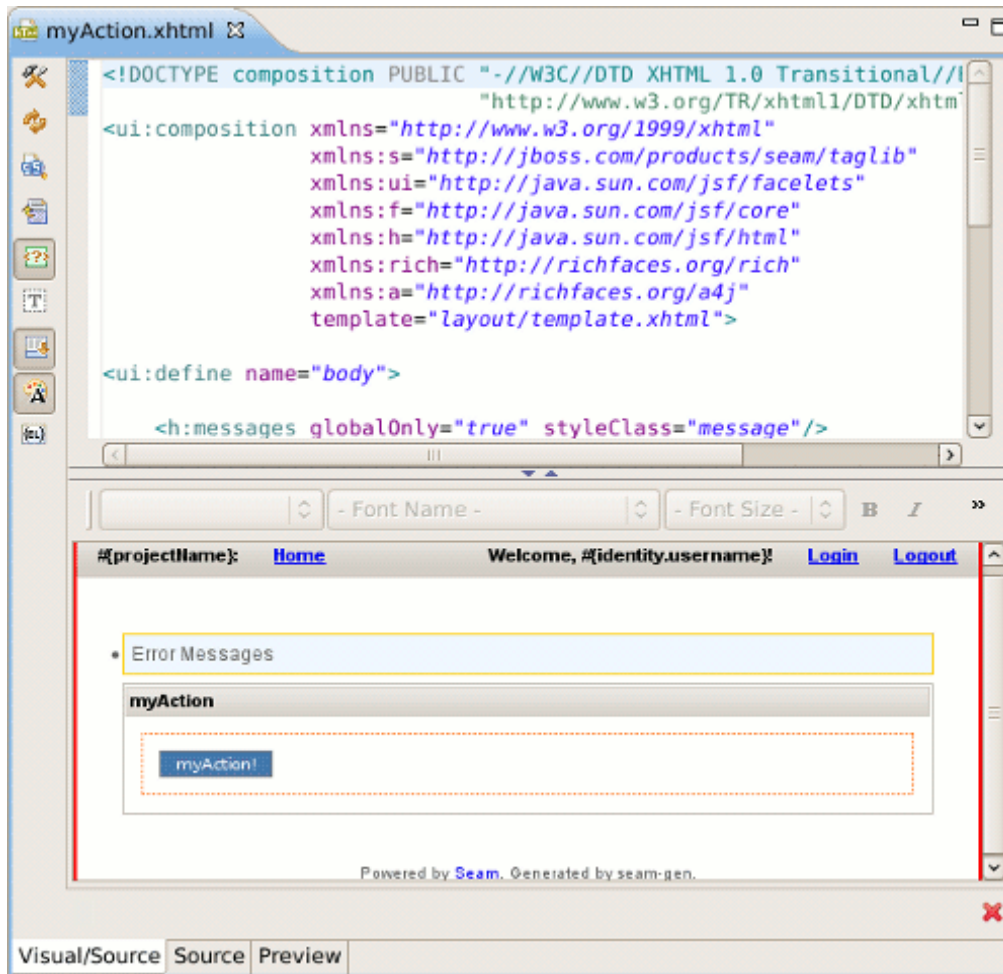


Figure 3.27. Open Seam Action with Editor

Right click on the "myAction!" button in the visual part of editor and select `<h:commandButton>` Attributes.

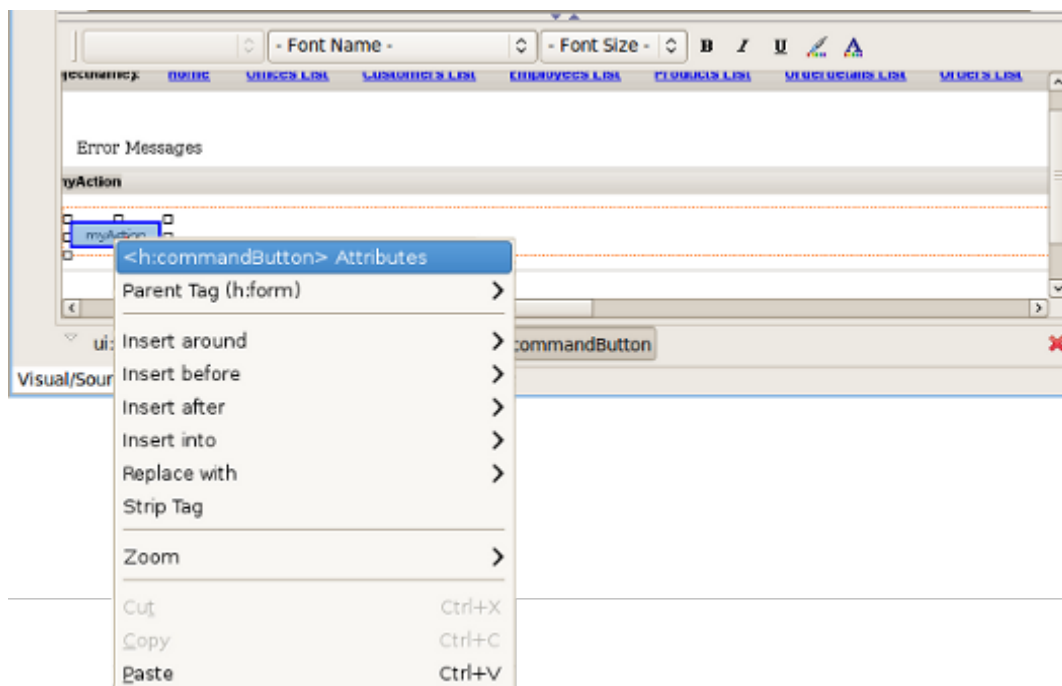


Figure 3.28. Seam Action Editing

Change the value of the button to something different. If desired, you can change any other text on the page. Then, type **CTRL+S** to save the facelet.

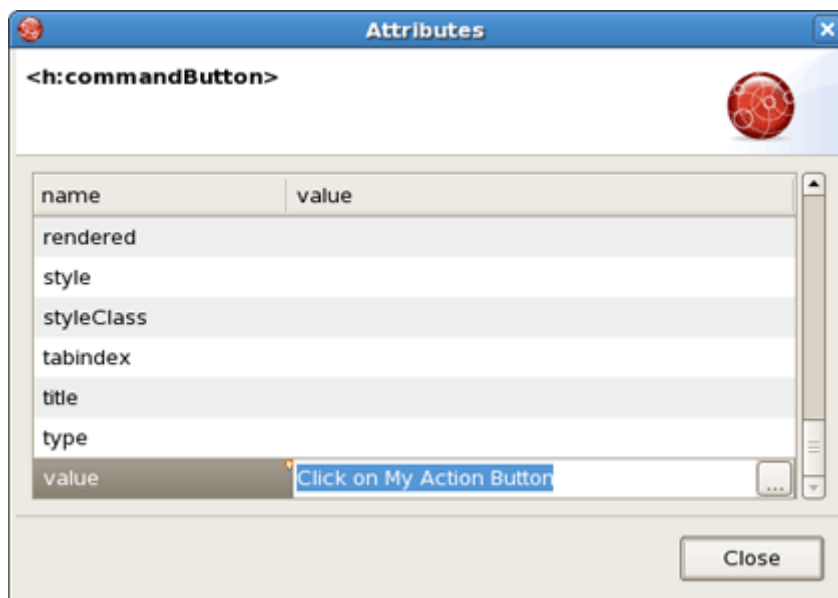


Figure 3.29. Attributes Dialog

Refresh <http://localhost:8080/workshop/myAction.seam> and now you should see your changes.

Notice that you did not have to publish the application. JBoss Developer Studio auto-published it for you.

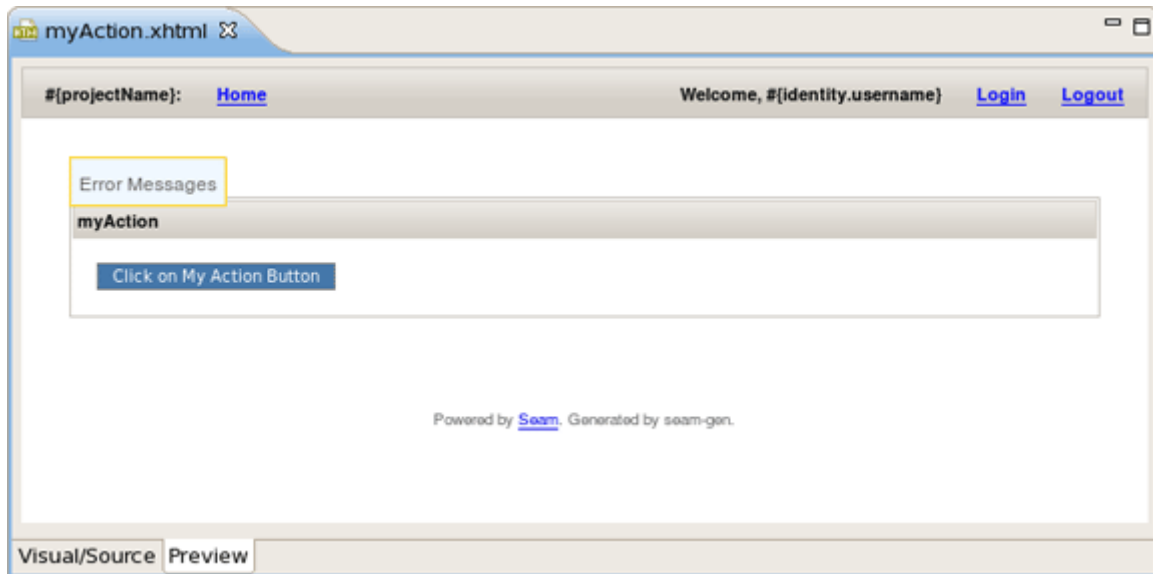


Figure 3.30. Seam Action Is Modified

3.3. Declarative Security

In this section you will see how easy it is to secure the facelets and facelet components in Seam. Let's go ahead and secure the action button, then we will secure the entire page.

3.3.1. Edit Login Authentication Logic

There is a class called `Authenticator.java`. The login page will execute the `Authenticator.authenticate()` method by default, so we'll start by viewing the authentication logic.

Open `Authenticator.java` in JBoss Developer Studio and you will see that it contains the `authenticate()` method with this code:

```
public boolean authenticate()
{
    log.info("authenticating {0}", credentials.getUsername());
    //write your authentication logic here,
    //return true if the authentication was
    //successful, false otherwise
    if ("admin".equals(credentials.getUsername()))
    {
        identity.addRole("admin");
        return true;
    }
    return false;
}
```

3.3.2. Secure Seam Page Component

Open `myAction.xhtml` and add a new secured command button:

```
<h:commandButton id="myActionSecured"
value="Secured Action Button"
action="#{myAction.myAction}"
rendered="#{s:hasRole('admin')}" />
```

Refresh `http://localhost:8080/workshop/myAction.seam` If you are not logged in you will only see one button. If you are logged in, there will be two buttons.

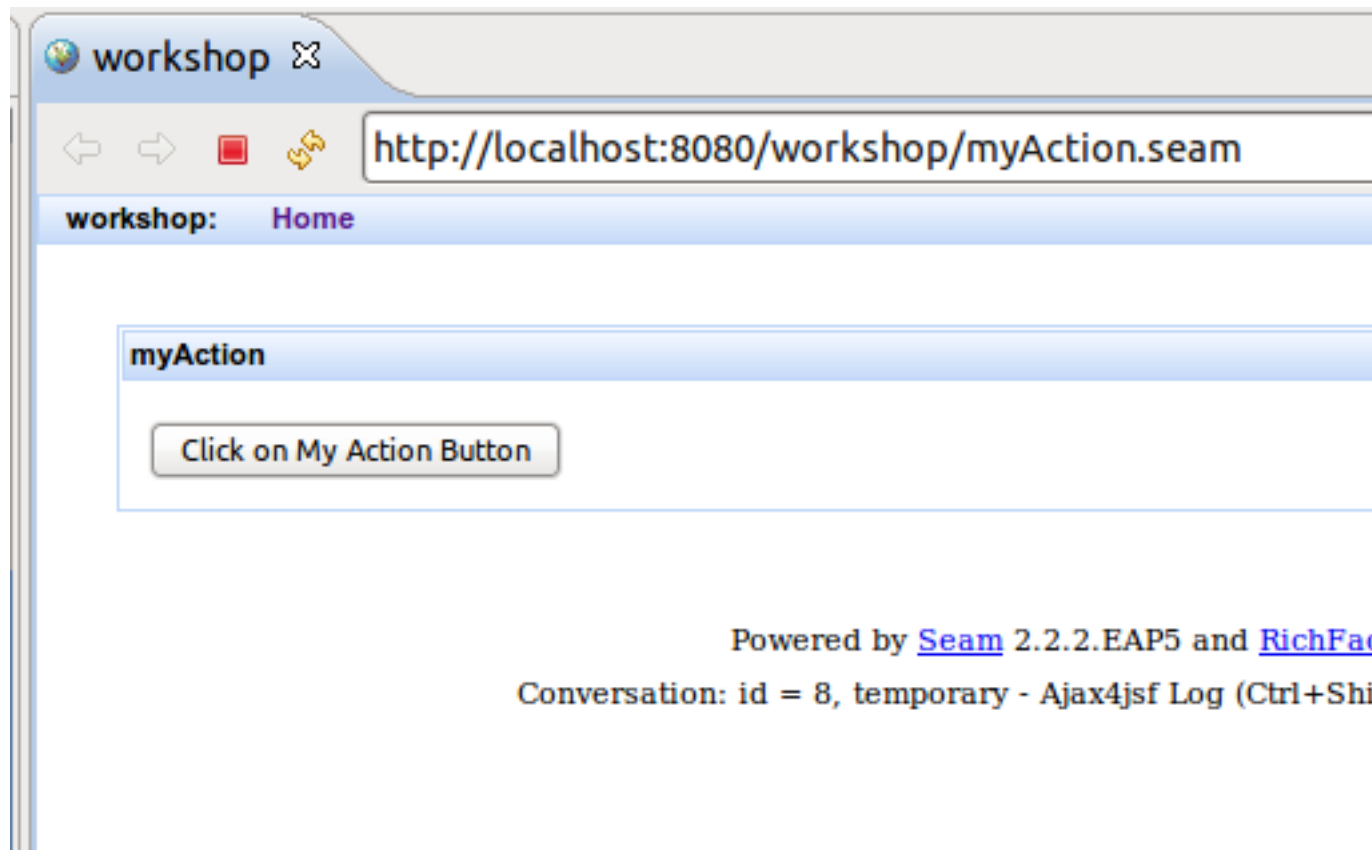


Figure 3.31. One Button on a Page

The secured button is not visible because the user isn't logged in as "admin".

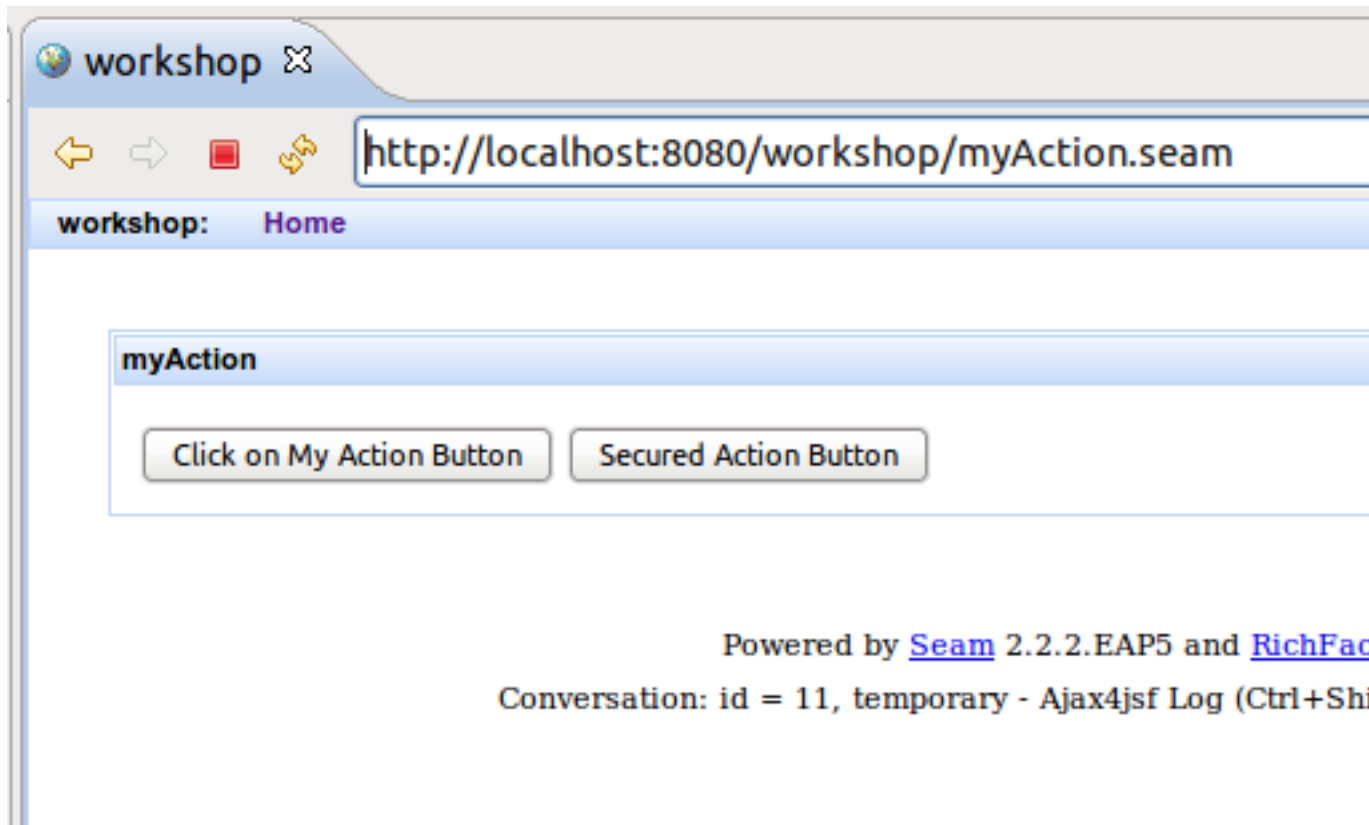


Figure 3.32. Secured Button is Visible

The user is logged in as "admin". Securing components is easy but securing pages is pretty simple as well.

Open `WebContent/WEB-INF/pages.xml`. Then add this markup directly underneath the `<pages>` element:

```
<page view-id="/myAction.xhtml" login-required="true"/>
```

Refresh `http://localhost:8080/workshop/myAction.seam`. If you are not logged in you will get bounced back to the login page.

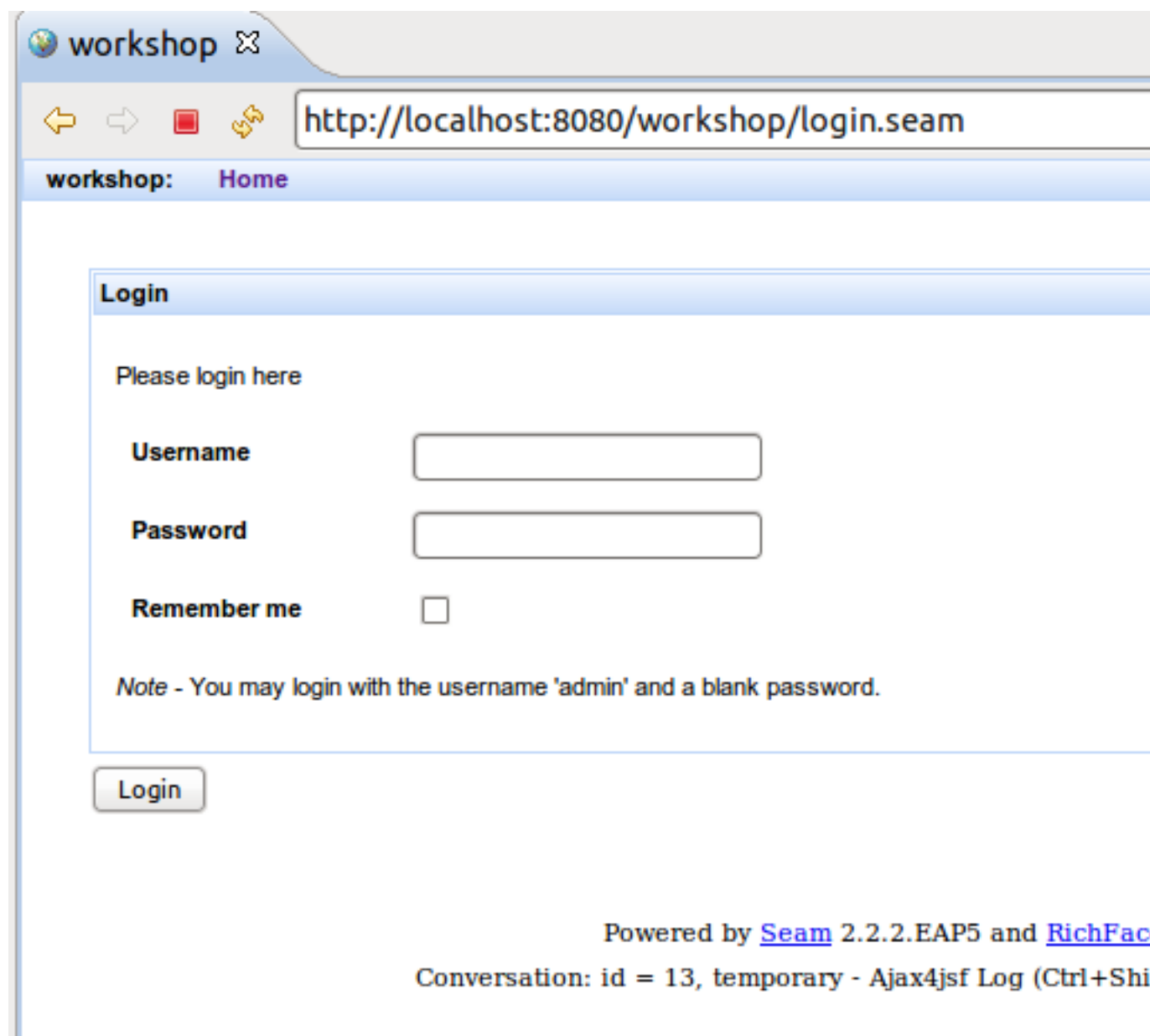


Figure 3.33. Login Page

Thus, if you enter login credentials for the "admin" user, you will be re-directed to the secured page and secured component. If you enter different login credentials, page access will be granted, but the secured component will not be displayed.

Congratulations! You have secured your new action both at the facelet component and page level. You also added custom authentication logic to the login action.

3.4. Browsing Workshop Database

In this section you get to know how to use the workshop database that was started at the beginning of the lab.

3.4.1. Database Connectivity Setup

The workshop data can be browsed inside of JBoss Developer Studio.

To open the Data Source Explorer, click on **Window** → **Open Perspective** → **Other** → **Database Development**.

In the Data Source Explorer, expand the Databases node and select the Default database. Right click on it, select **Connect** from the context menu.

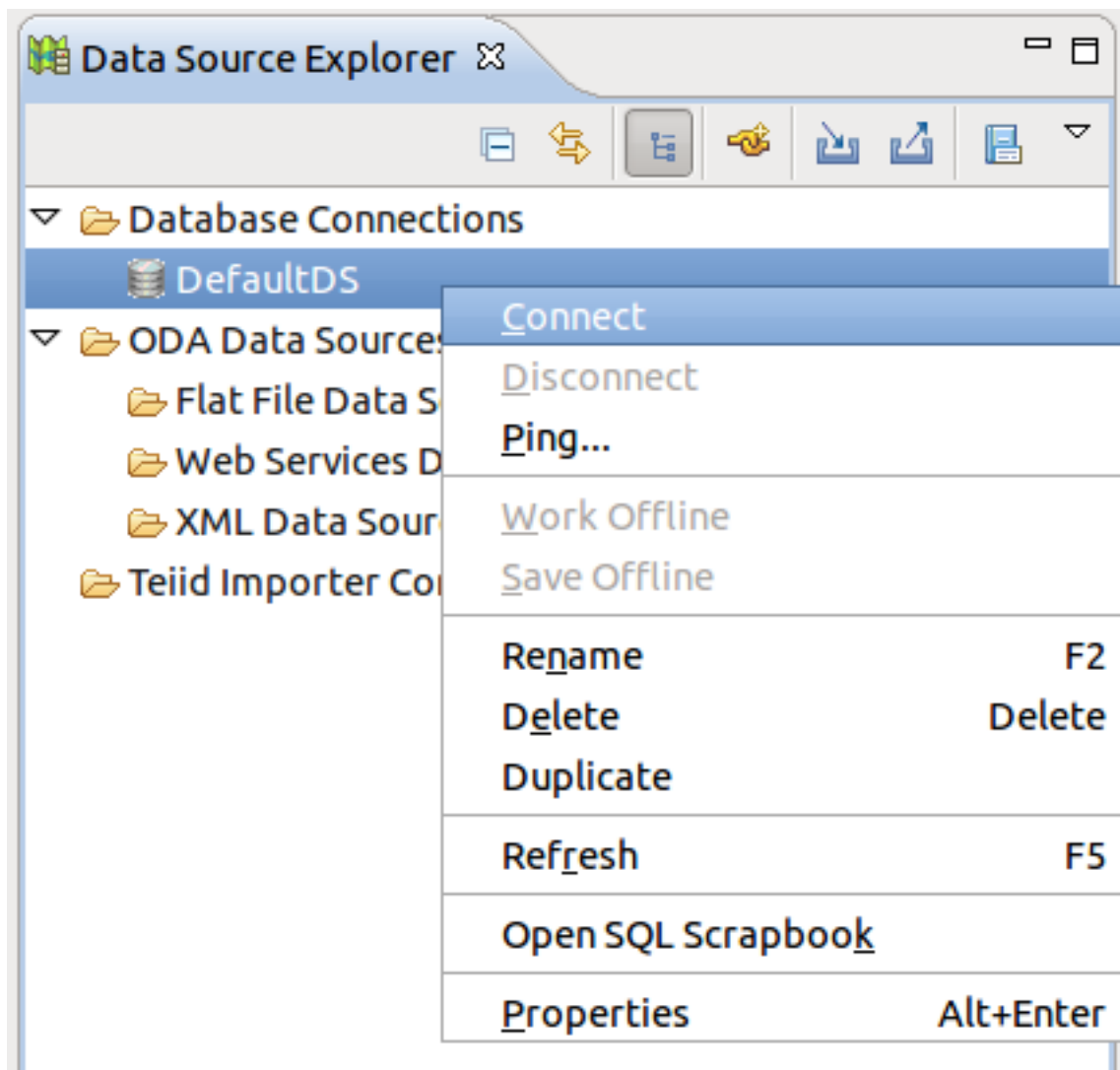


Figure 3.34. Data Source Explorer

3.4.2. Browse Workshop Database

Then in the current view, drill down to the CUSTOMERS table.

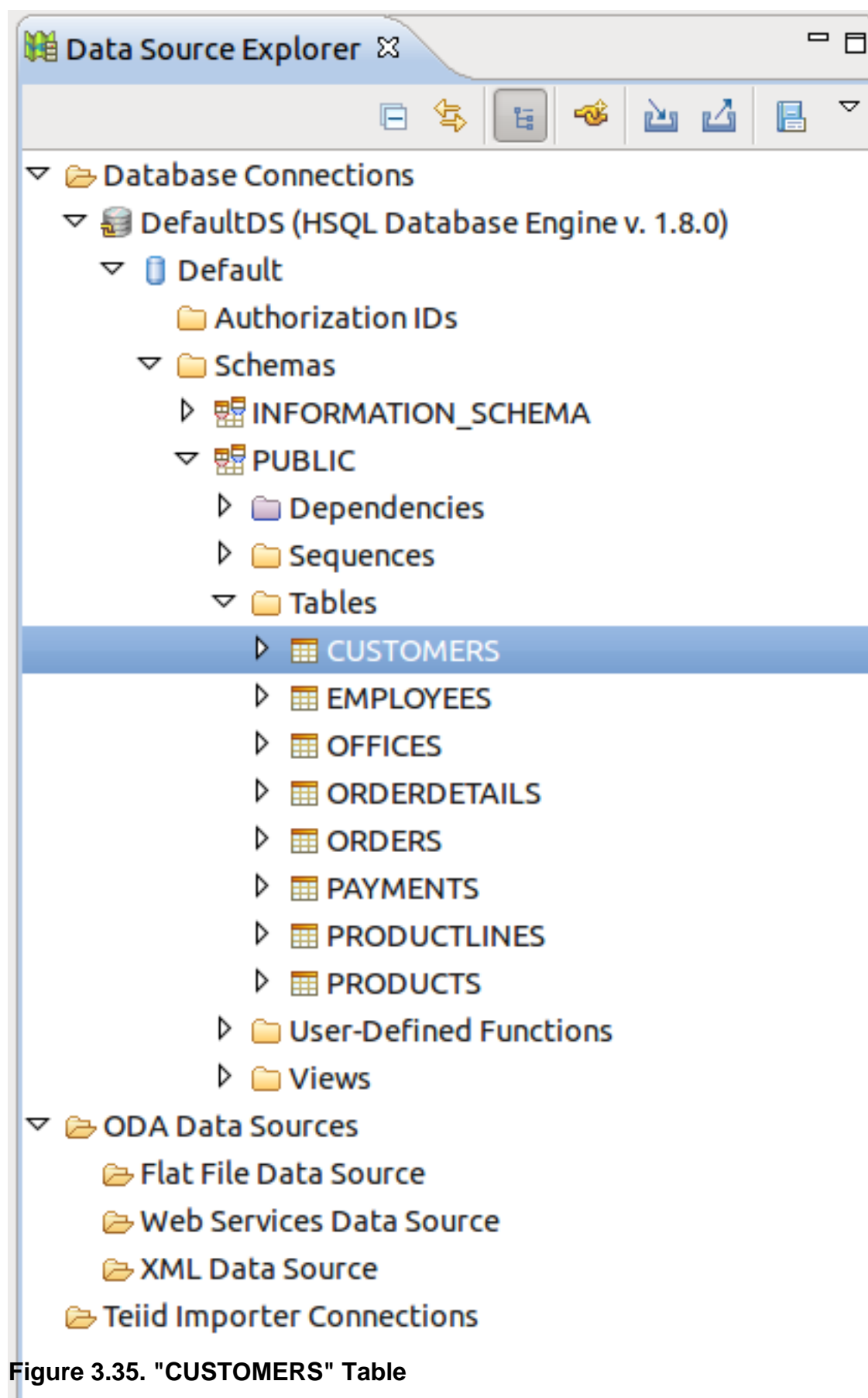
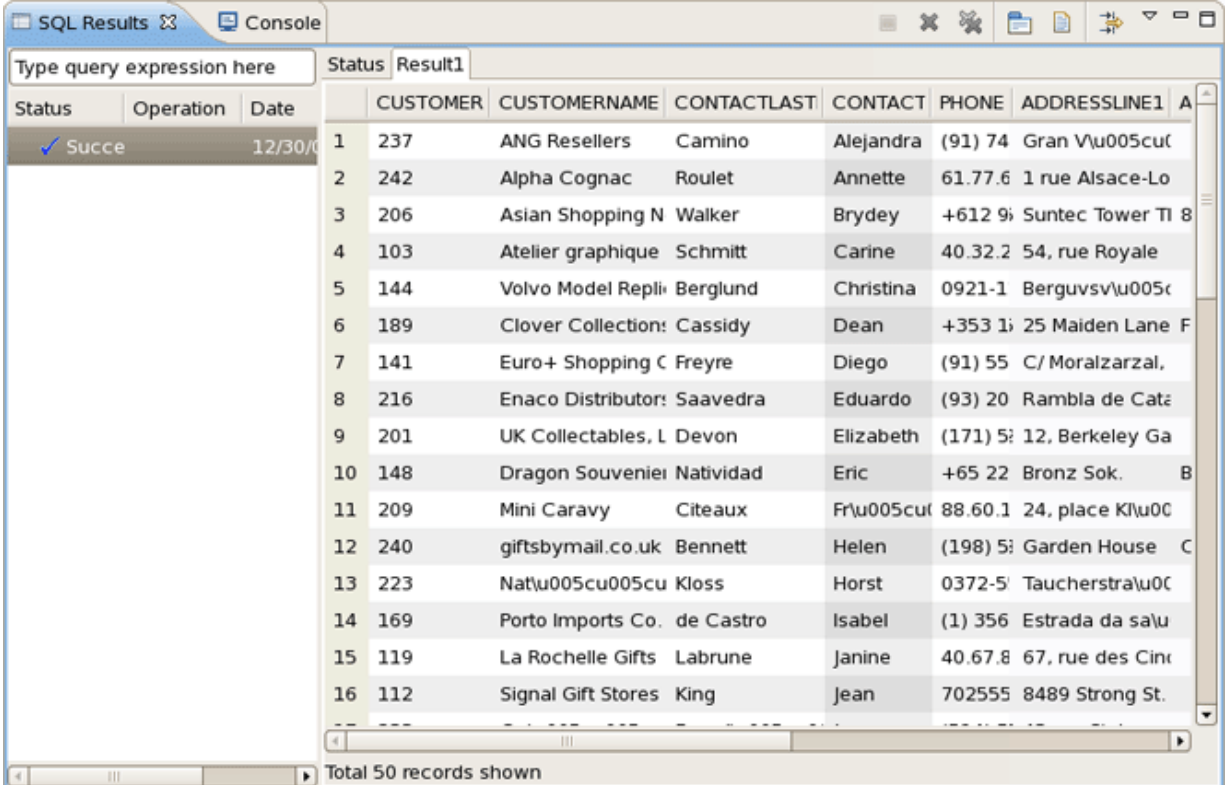


Figure 3.35. "CUSTOMERS" Table

Right click on CUSTOMERS, select **Data** → **Sample Contents** to view the data in the table.

There should be a SQL Results view on the workbench, but it could be hidden. Click on the "Result1" tab in the right side and you should see the data in the CUSTOMERS table.



Status	Operation	Date	CUSTOMER	CUSTOMERNAME	CONTACTLAST	CONTACT	PHONE	ADDRESSLINE1
✓ Success		12/30/0	237	ANG Resellers	Camino	Alejandra	(91) 74	Gran V\u0005cu
			242	Alpha Cognac	Roulet	Annette	61.77.6	1 rue Alsace-Lo
			206	Asian Shopping N	Walker	Brydey	+612 9	Suntec Tower TI 8
			103	Atelier graphique	Schmitt	Carine	40.32.2	54, rue Royale
			144	Volvo Model Repli	Berglund	Christina	0921-1	Berguvs\u0005c
			189	Clover Collection	Cassidy	Dean	+353 1	25 Maiden Lane F
			141	Euro+ Shopping C	Freyre	Diego	(91) 55	C/ Morazarzal,
			216	Enaco Distributor	Saavedra	Eduardo	(93) 20	Rambla de Cat
			201	UK Collectables, L	Devon	Elizabeth	(171) 5	12, Berkeley Ga
			148	Dragon Souvenir	Natividad	Eric	+65 22	Bronz Sok. B
			209	Mini Caravy	Citeaux	Fr\u0005cu	88.60.1	24, place Kl\u0000
			240	giftsbymail.co.uk	Bennett	Helen	(198) 5	Garden House C
			223	Nat\u0005cu005cu	Kloss	Horst	0372-5	Taucherstra\u0000
			169	Porto Imports Co.	de Castro	Isabel	(1) 356	Estrada da sa\u0000
			119	La Rochelle Gifts	Labrune	Janine	40.67.8	67, rue des Cinc
			112	Signal Gift Stores	King	Jean	702555	8489 Strong St.

Total 50 records shown

Figure 3.36. SQL Results View



Note:

If you can't find the SQL Results view tab, click on **Window** → **Show View** → **Other** → **SQL Development** → **SQL Results**.

Congratulations! You just connected to the workshop database and queried the content using Database Explorer tools.

3.5. Database Programming

Now, it's time to reverse engineer the workshop database into a fully functioning Seam CRUD (Create Read Update Delete) application.

3.5.1. Reverse Engineer CRUD from a Running Database

In JBoss Developer Studio, switch to the Seam perspective, and then right-click the project and select **New** → **Seam Generate Entities**.

The "workshop" project in the Seam Generate Entities wizard will be selected automatically. There is no need to change something more, click the **Next** button to proceed to the next step.

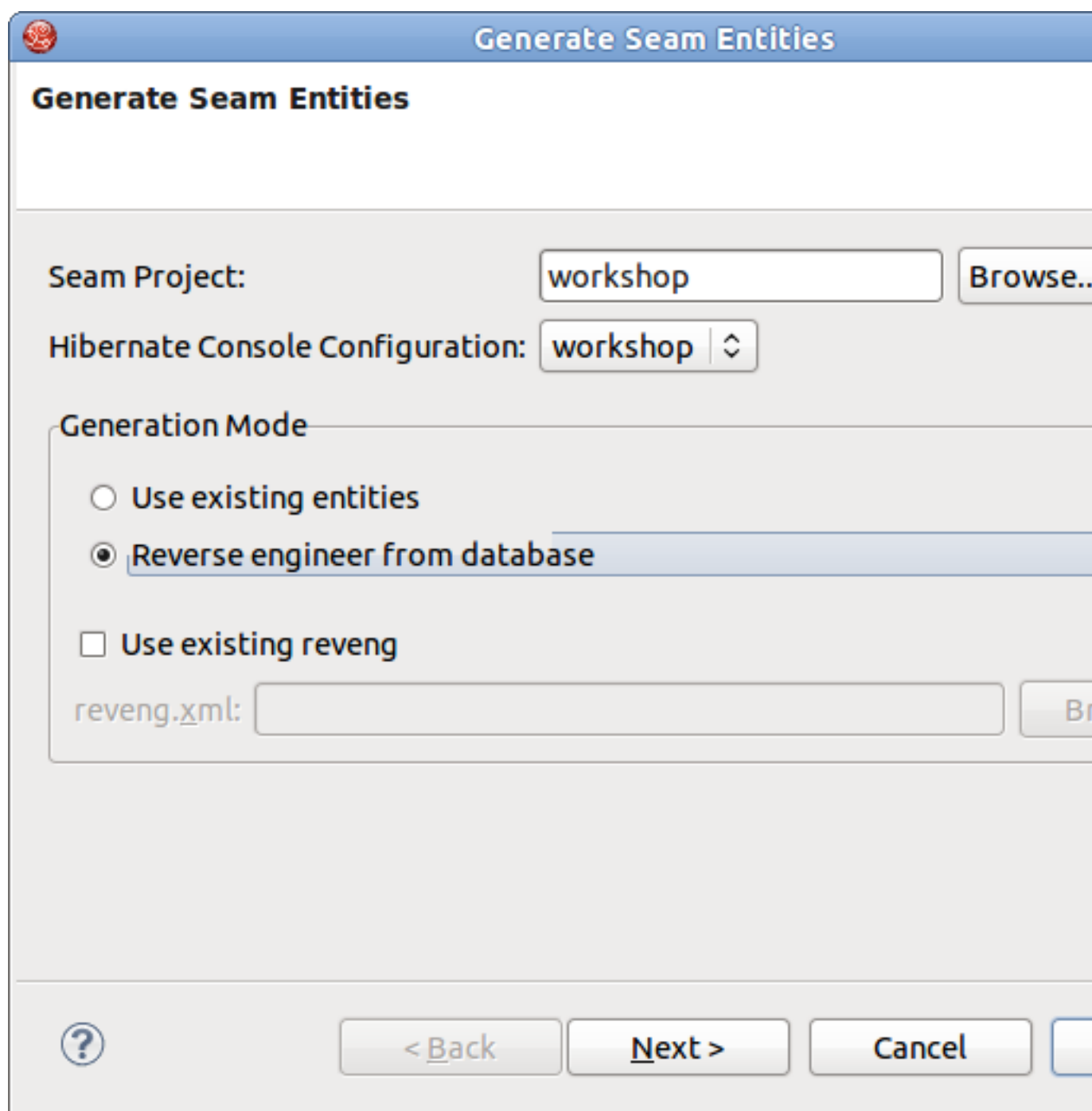


Figure 3.37. Generate Seam Entities

On the next page use the **Refresh** button to display the database, then click the **Include** button to include all the tables from the database, and finally click the **Finish** button.

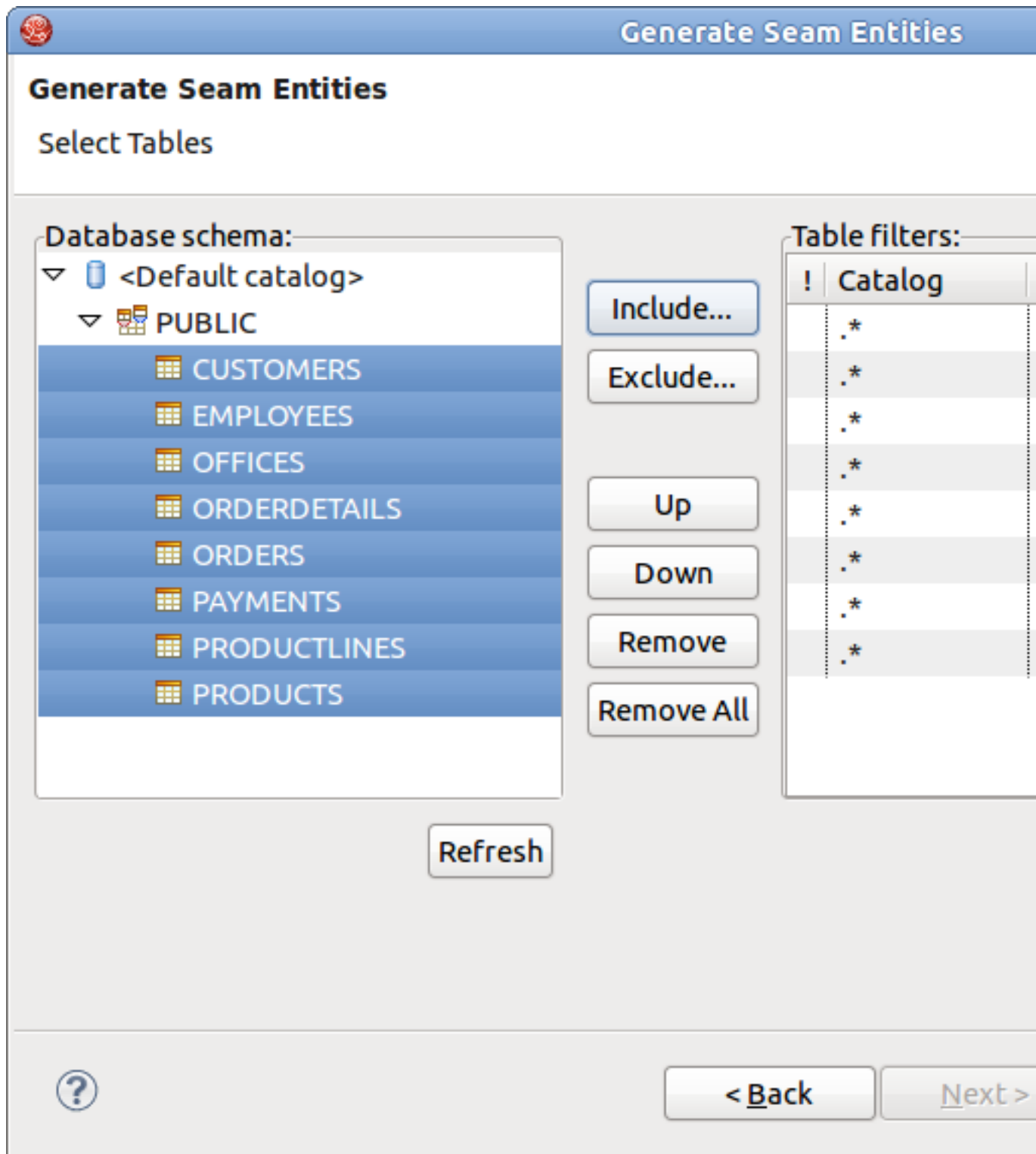


Figure 3.38. Selecting Tables

After running the Generate Entities action, you will see new *org.domain.workshop.entity* classes. These classes represent insert/update/delete/query logic.

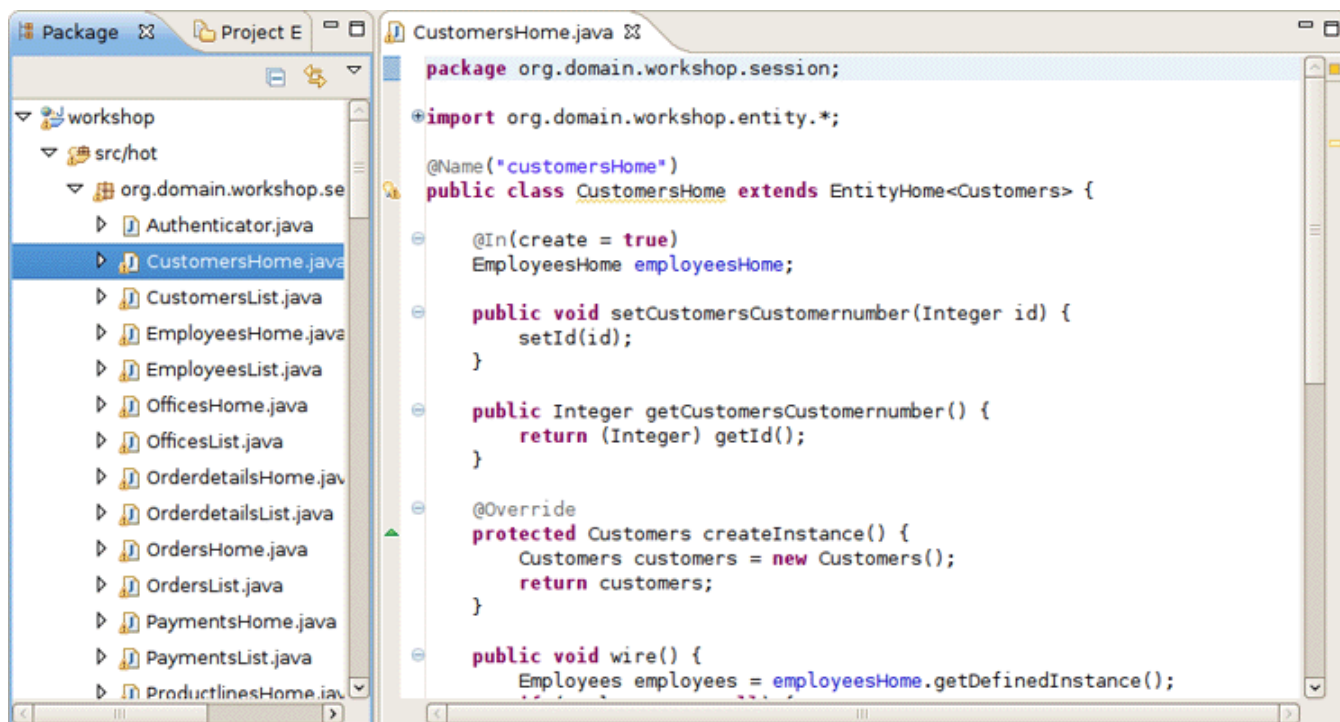


Figure 3.39. org.domain.workshop.entity Classes

There is also the `org.domain.workshop.entity` package that contains the JPA classes. These are the entity beans that are mapped to database tables. Note that you can use Seam refactoring tools with Seam components. Read more about it in [Seam refactoring tools chapter](http://download.jboss.org/jbosstools/nightly-docs/en/seam/html_single/index.html#seam_refactoring) [http://download.jboss.org/jbosstools/nightly-docs/en/seam/html_single/index.html#seam_refactoring] of Seam Dev Tools Reference Guide.

Last, but not least, there are facelets for all of the CRUD screens. The best way to get a feel for the generated code is to open a browser and play around with the application. Go to `http://localhost:8080/workshop` and insert/update/delete/query a few records. There is quite a bit of AJAX in this application, but we will explore that later on in the lab. For now, take note of the page tabs, required field logic and data table sorting in the list pages.



Tip

If you see the error `java.lang.ClassNotFoundException: org.jboss.seam.servlet.SeamListener` in the console output from the Application Server, you may need to copy the `jboss-seam.jar` file from the `lib` subdirectory in the Seam library (which can be downloaded from [here](http://seamframework.org/Seam2/Seam2DistributionDownloads) [http://seamframework.org/Seam2/Seam2DistributionDownloads]) into the `/server/default/deploy/workshop.war/WEB-INF/lib/` subdirectory in your Application Server (where "default" refers to the server profile that you are using).



Tip

If you see the error `Could not instantiate Seam component: org.jboss.seam.security.ruleBasedPermissionResolver`, copy the `mvel2.jar` file from the Seam library to the same destination directory mentioned in the tip above.

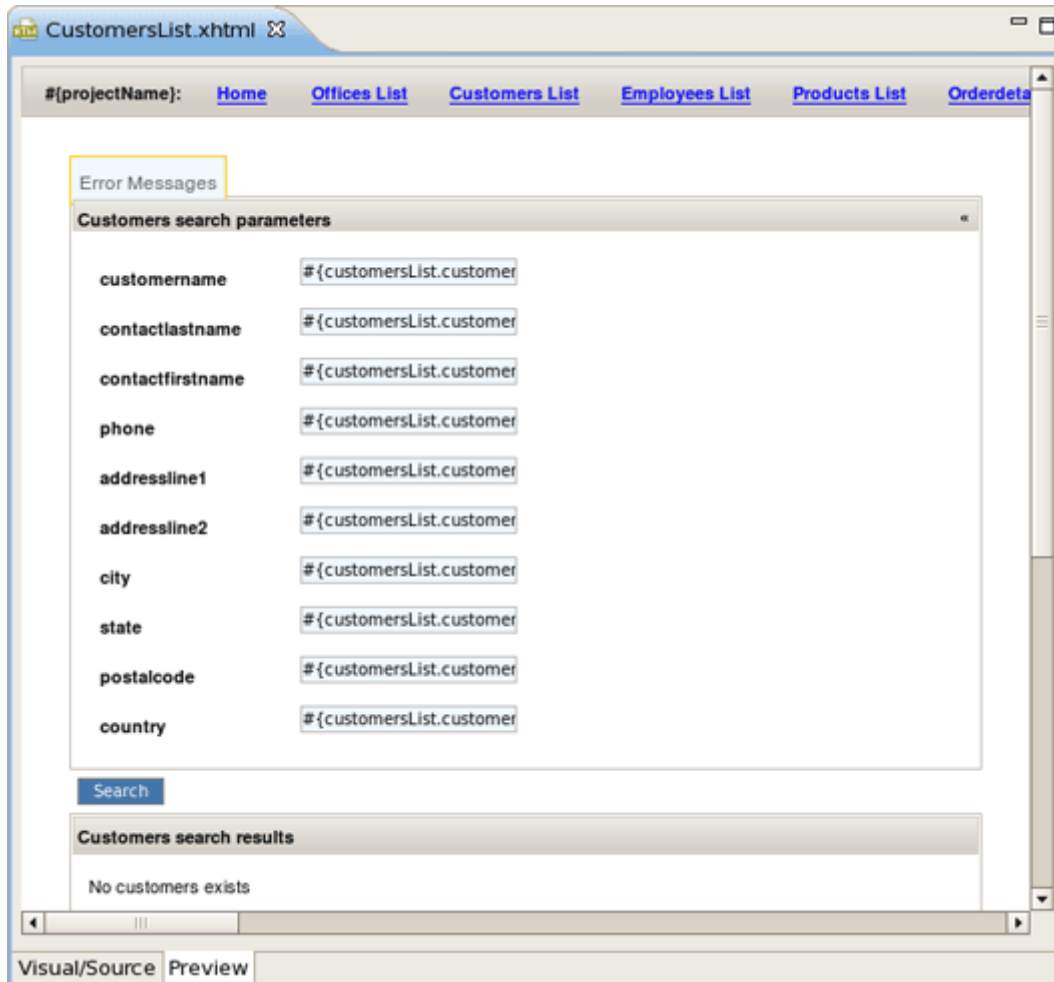


Figure 3.40. CustomersList.xhtml in the Editor

Congratulations! You now have a fully functioning CRUD application that is AJAX enabled.

3.5.2. Use Hibernate Tools to Query Data via JPA

Now, it's time to write some JPA queries using the Hibernate perspective in JBoss Developer Studio.

In the upper right corner of the workbench there is a small icon (see the figure below), click on it and select **Hibernate**.

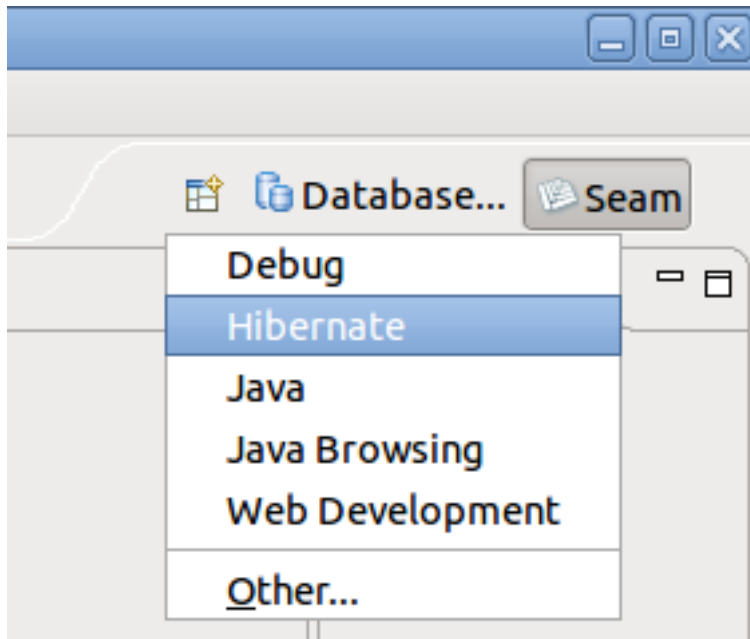


Figure 3.41. Hibernate Perspective

Look at the Hibernate Configurations view. In the "workshop" project, drill down on the Session Factory and notice that the JPA entities/attributes are listed in a nice tree view.

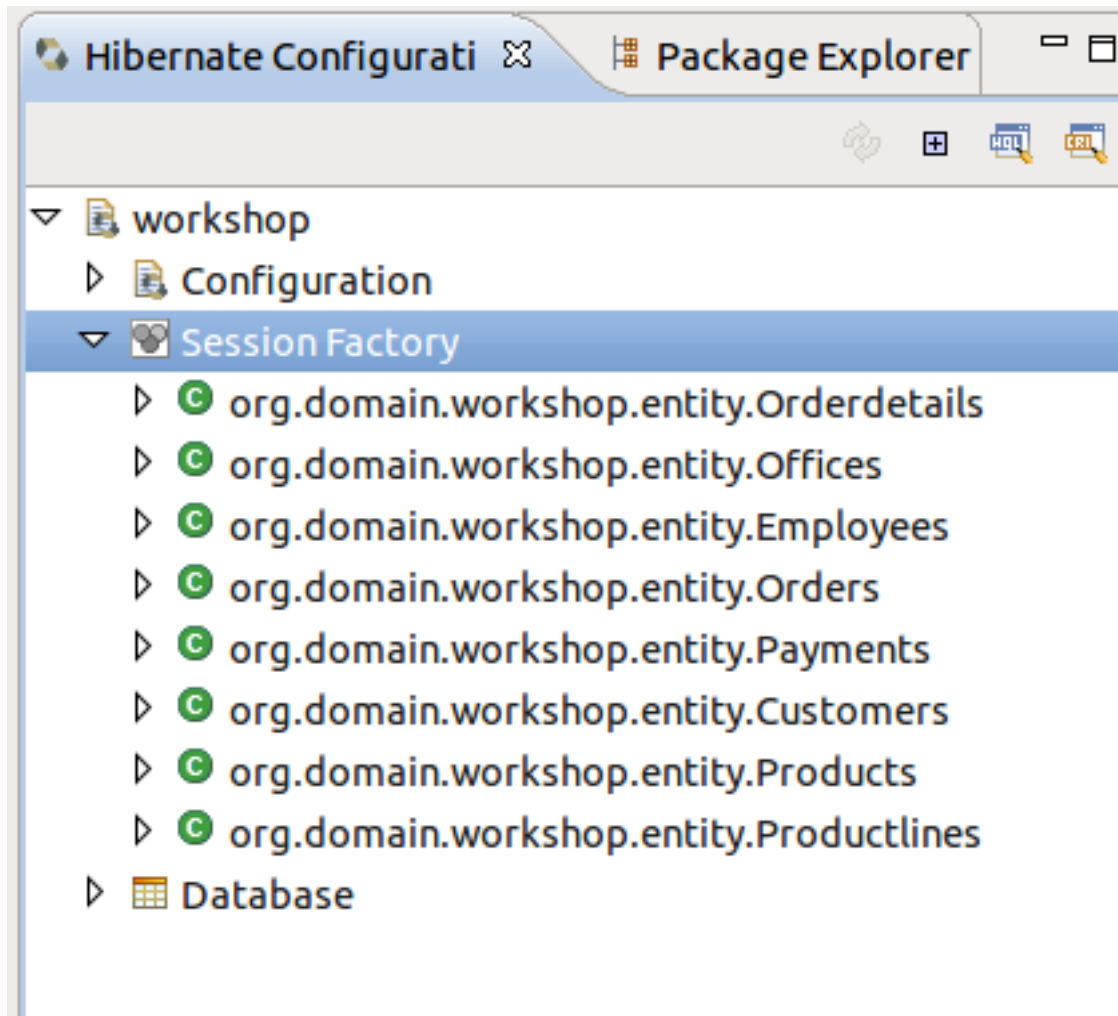


Figure 3.42. Hibernate Configurations View

Right click on the Session Factory and select **HQL Editor**. This will open a JPA query scratch pad window.

Write your query and click on the "Hibernate Dynamic SQL Preview" tab. You should see the SQL that will be executed if this JPA query is run.

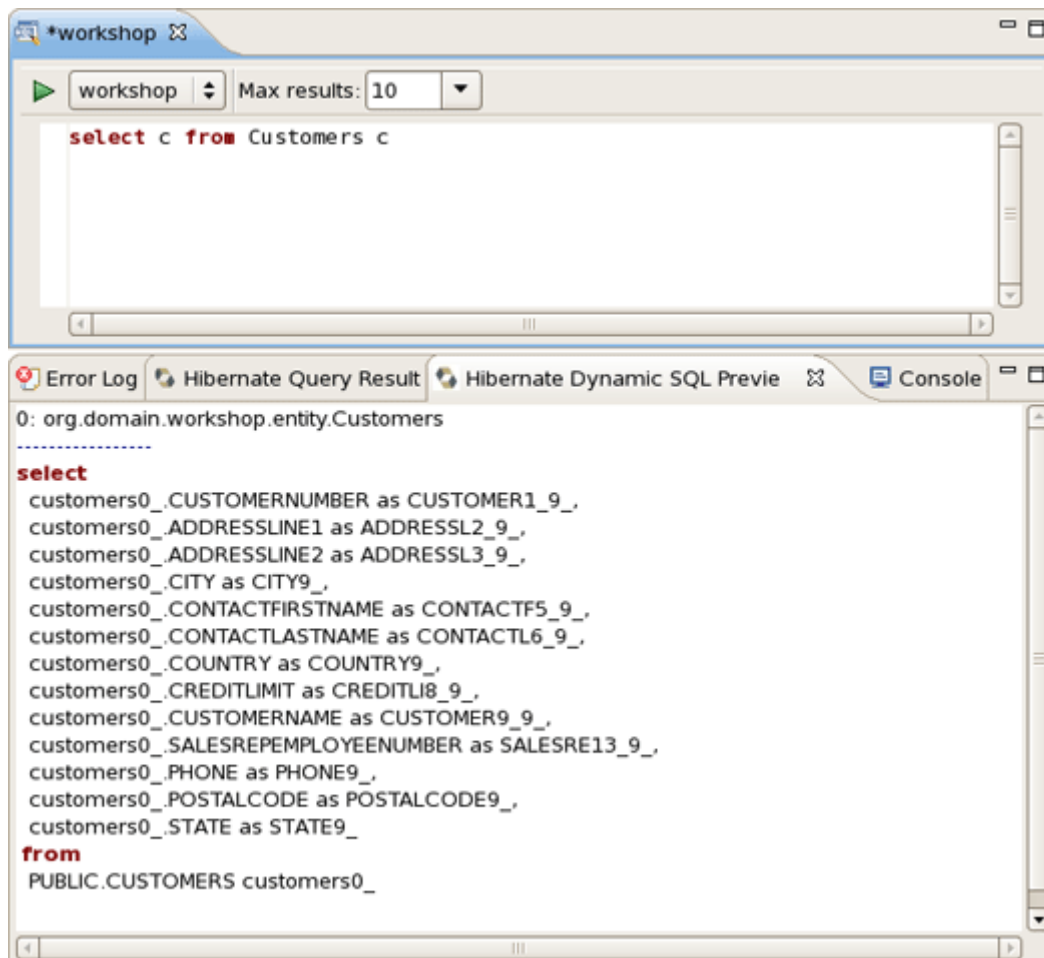


Figure 3.43. JPA Query Editor

Run the query by clicking on the green run icon.

The results are listed in the "Hibernate Query Result" view. There is a "Properties" tab in the workbench that can be used to see a specific JPA result. These results represent the JPA objects because our query did not specify column names.

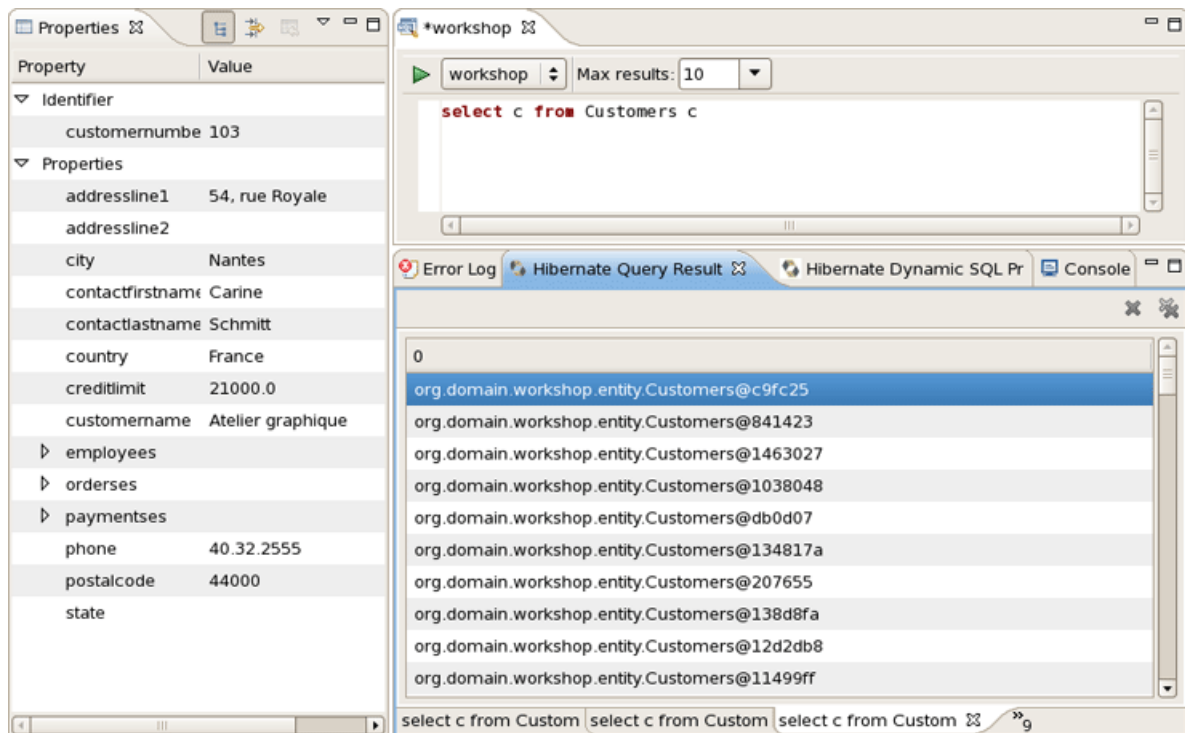


Figure 3.44. Hibernate Query Result View

The query can be refined, and take note that there is nice code completion in the JPA query editor.

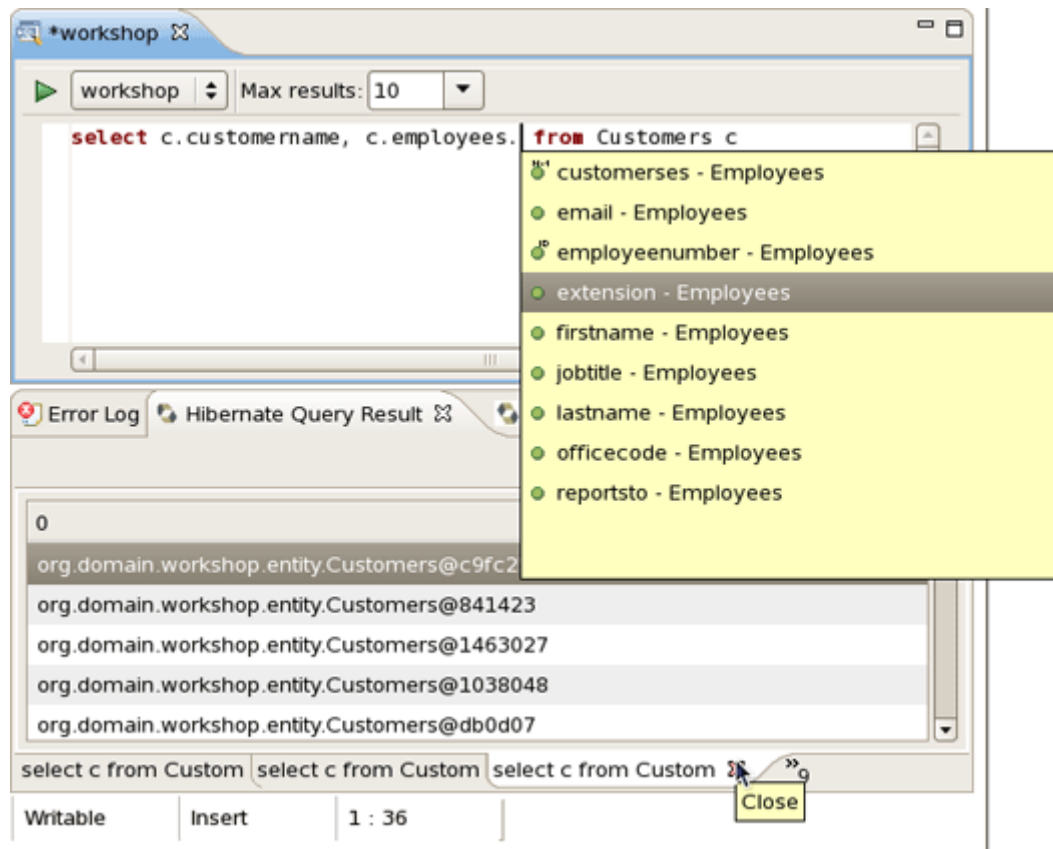


Figure 3.45. Code Completion

A refined query will return results that are more ResultSet oriented. Notice the join logic that JPA supports.

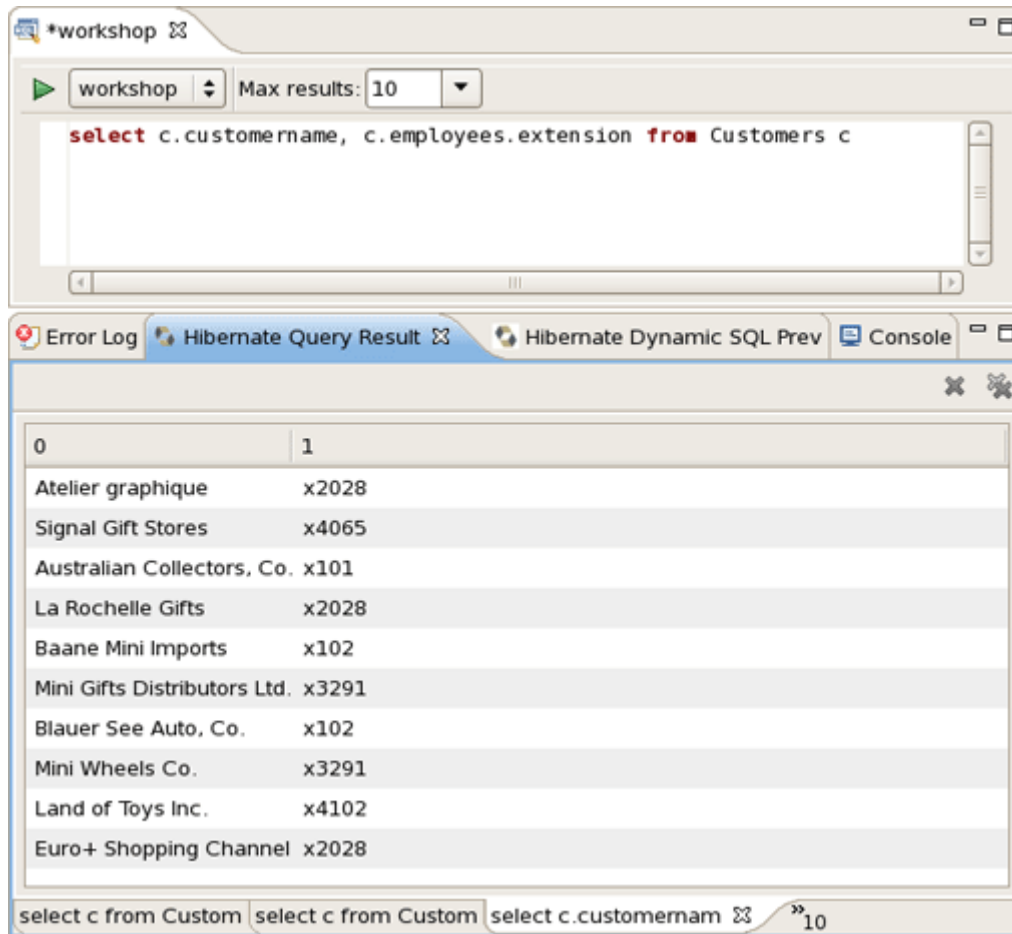


Figure 3.46. The Hibernate Query Result

There was no need to specify an Employees table in the from part of the JPA query because JPA supports reference traversal via Java class attribute references. Not only are JPA and HQL queries fully supported, but Criteria based queries can also be written in the Criteria Editor. You should spend some time tinkering with different queries and possibly Criteria based queries, even though the instructions are not provided in this lab.

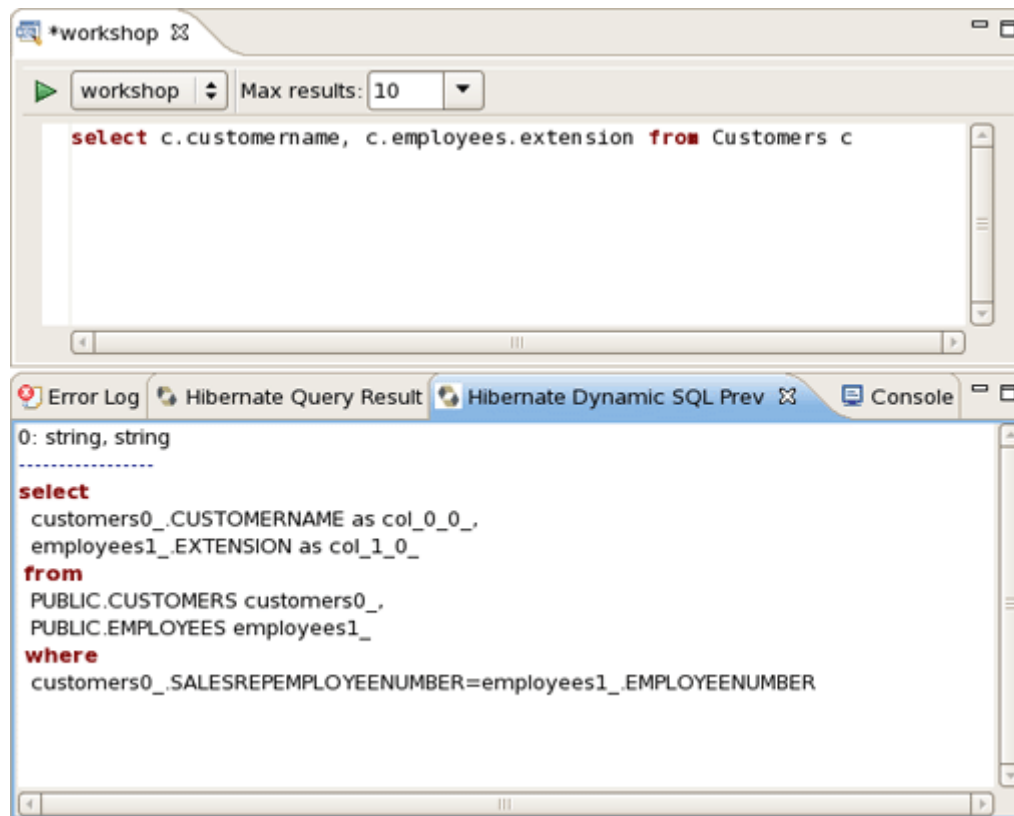


Figure 3.47. Criteria Editor

3.5.3. Use Hibernate Tools to visualize the Data Model

Now, it's time to view the data model for the workshop database.

In the Hibernate Configurations view, select "workshop" project and expand the Configuration node. Select the Customers entity, right click on it, choose **Mapping Diagram**.

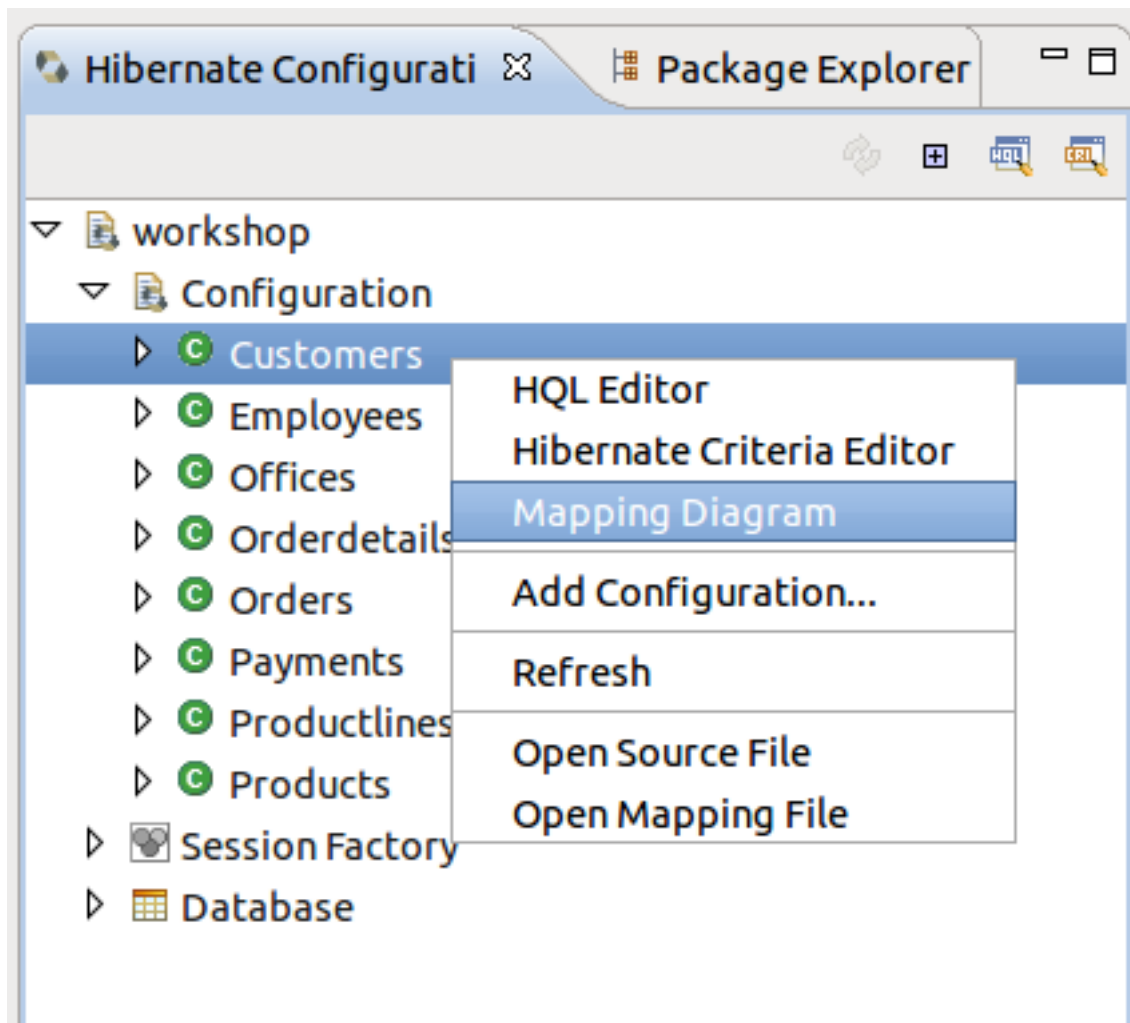


Figure 3.48. Mapping Diagram Opening

You see a Diagram tab for the CUSTOMERS table and any tables that have FK references. This is a handy way to view the data model and JPA mappings. Now, you've got access to something that the Erwin Data Modeler can't do.

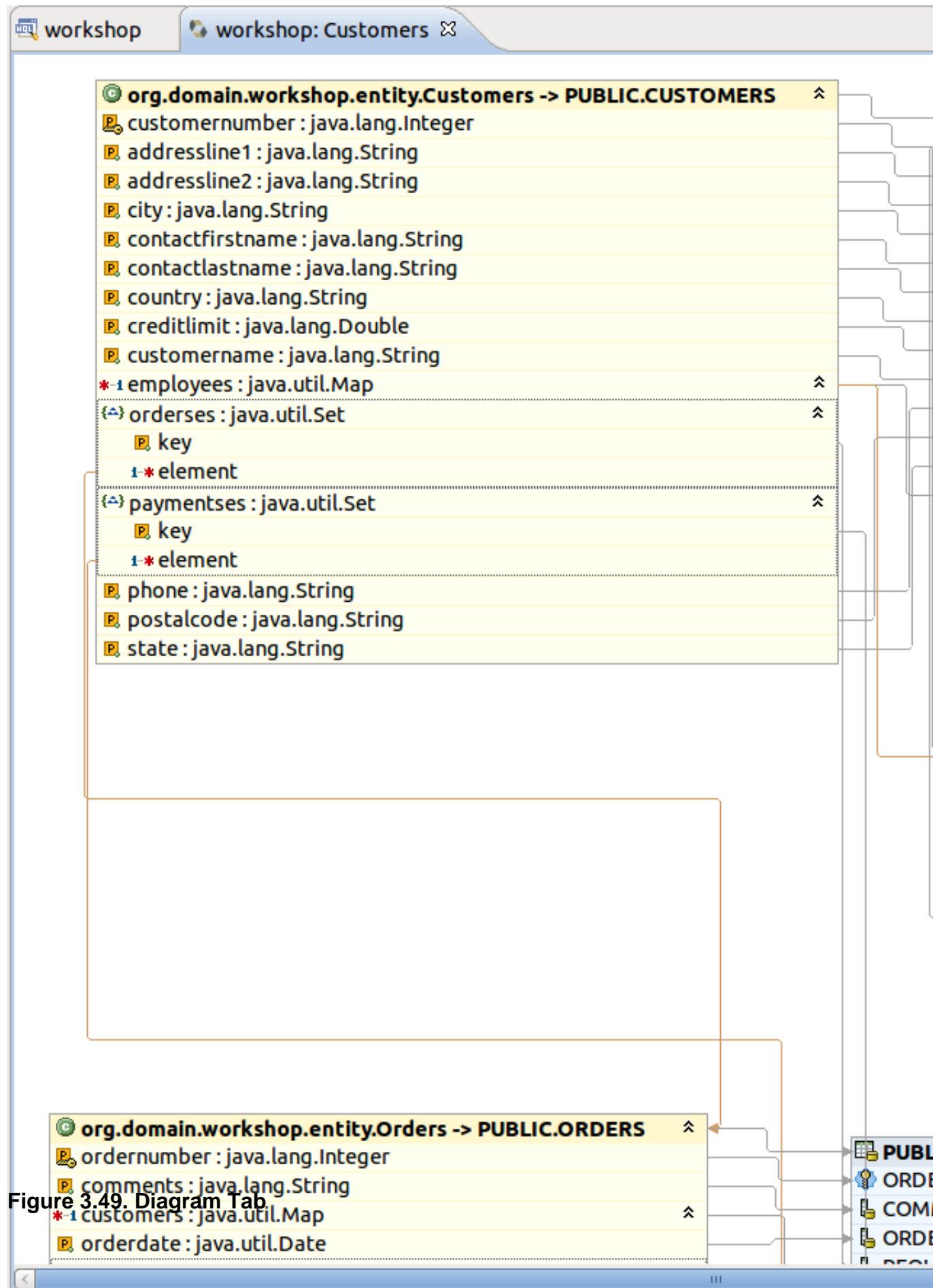


Figure 3.49. Diagram Tab

3.6. Rich Components

This lab will conclude with one last AJAX twist. In this section we add a RichFaces `inputNumberSlider` to the Order Details edit screen.

3.6.1. Add a Richfaces component to the CRUD Application

Switch to Seam perspective, and open `WebContent/OrderdetailsEdit.xhtml` in JBoss Developer Studio.

Change the form field values using the visual editor. Seam has generated the form field names that match the database column names. This is not ideal for business users.

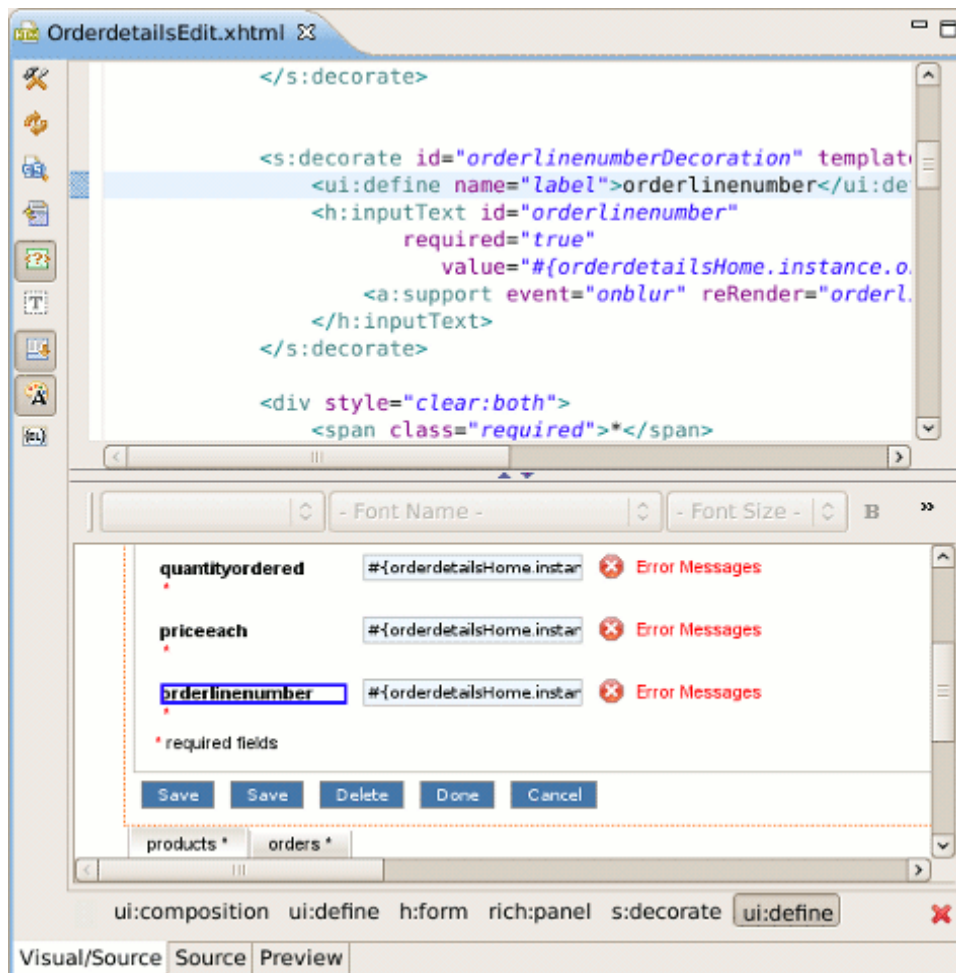


Figure 3.50. Form Fields Editing

Also, replace the QTY Ordered input field with a `inputNumberSlider`. You can use the JBoss Developer Studio palette or right click on the form and insert the RichFaces component.

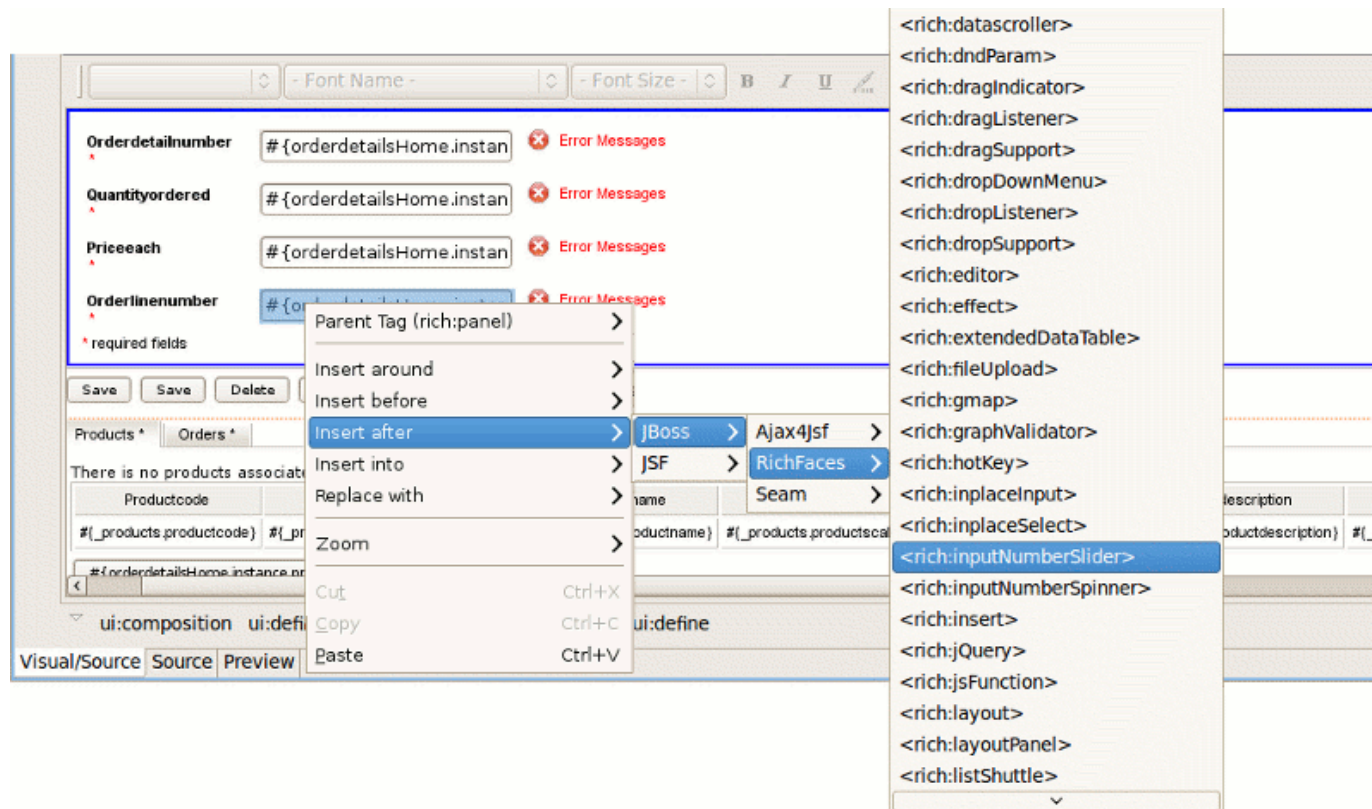


Figure 3.51. Insert RichFaces Component from Context Menu

One the last option is to use the source view and manually copy the `inputNumberSlider` markup listed below:

```
<rich:inputNumberSlider id="quantityOrdered" required="true"
    value="#{orderdetailsHome.instance.quantityordered}" />
```

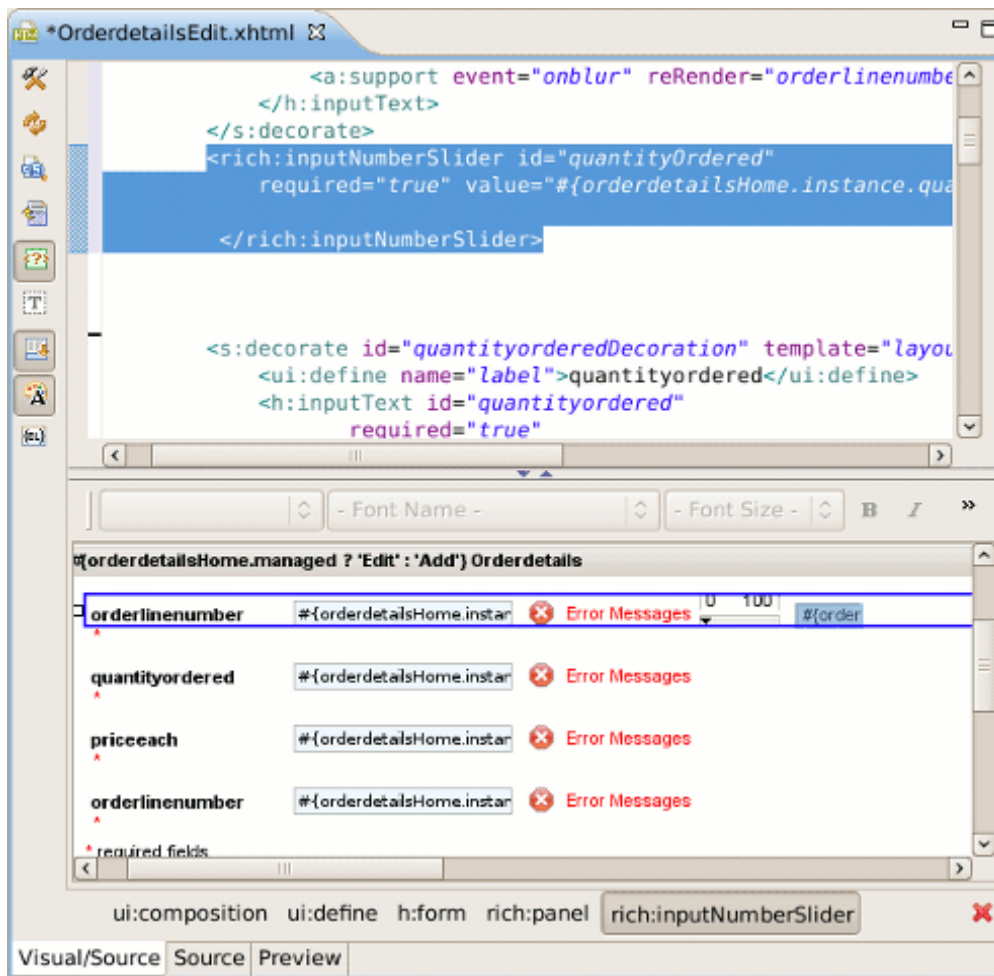


Figure 3.52. Manually copying Source Code

The end result is an edit page that has better form labels and a new RichFaces control.

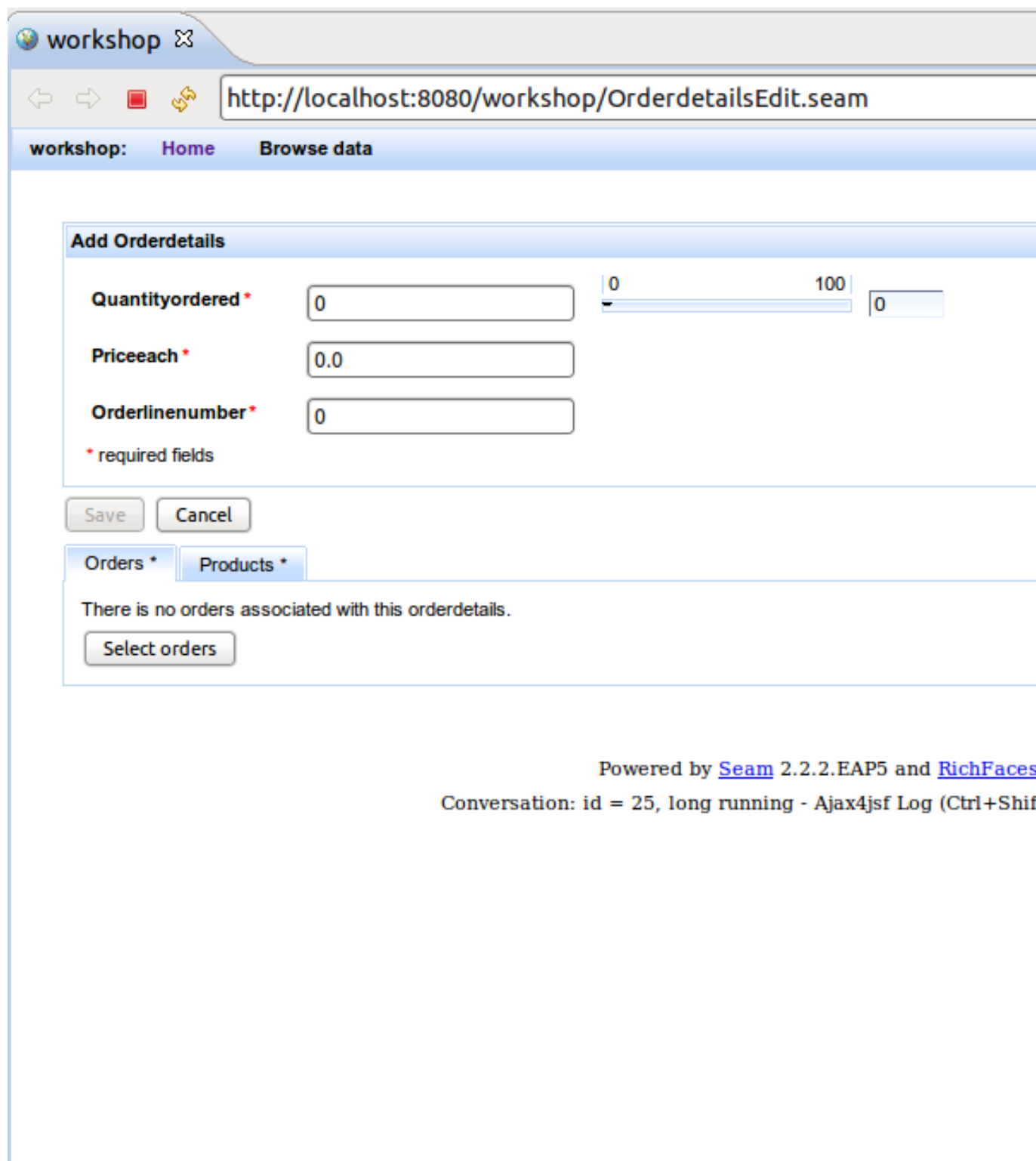


Figure 3.53. The Result Page

Congratulations! You have completed the JBoss Developer Studio lab.

Developing a simple JSP web application



Note:

We highly recommend developing in Seam. This chapter is for users who for some reason cannot use Seam.

In this chapter you'll find out how to create a simple [JSP](http://java.sun.com/products/jsp/) [http://java.sun.com/products/jsp/] application using JBoss Developer Studio. The application will show a classic "Hello World!" on the page.

We'll assume that you have already launched JBoss Developer Studio and also that the Web Development perspective is the current perspective. If not, make it active by selecting **Window** → **Open Perspective** → **Web Development** from the menu bar or by selecting **Window** → **Open Perspective** → **Other...** from the menu bar and then selecting Web Development from the Select Perspective dialog box.

4.1. Setting Up the Project

We are going to start by creating a Dynamic Web Project with a minimal structure, i.e. with just required facets. Thus this section will perform you all necessary steps on how to do this.

- Go to the menu bar and select **File** → **New** → **Other...**
- Select **Web** → **Dynamic Web Project** in the New Project dialog box
- Click the **Next** button
- Enter "jspHello" as a project name
- Then select *Minimal Configuration* from the list of possible configurations and click the **Finish** button.

New Dynamic Web Project

Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name:

Project location

☒ Use default location

Location:

Target runtime

Dynamic web module version

Configuration

The most conservative starting point. Only the required facets are installed. Additional facets can later be installed to add new functionality to the project.

EAR membership

☐ Add project to an EAR

EAR project name:

Working sets

☐ Add project to working sets

Working sets:

Figure 4.1. Create New Web Project

The *jspHello* node should appear in the upper-left Package Explorer view.

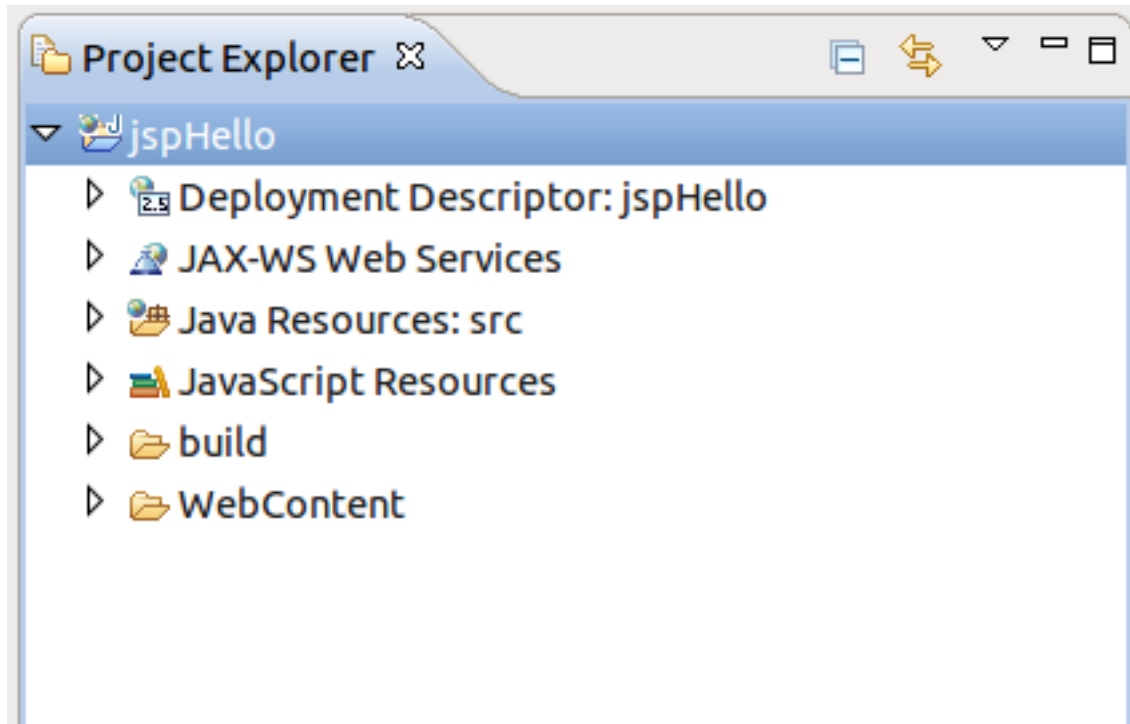


Figure 4.2. New Web Project

4.2. Creating JSP Page

This section covers all the points how to create, edit and then preview JSP page.

In our simple application we need to create only one JSP page which displays a *"Hello World!"* message.

- Right click the `WebContent` folder and select **New** → **JSP**.
- Type `hello.jsp` for a file name and click the **Next** button.

In the next window you can choose a template for your JSP page and see its preview.

- Select *New JSP File (xhtml)* template and click the **Finish** button.

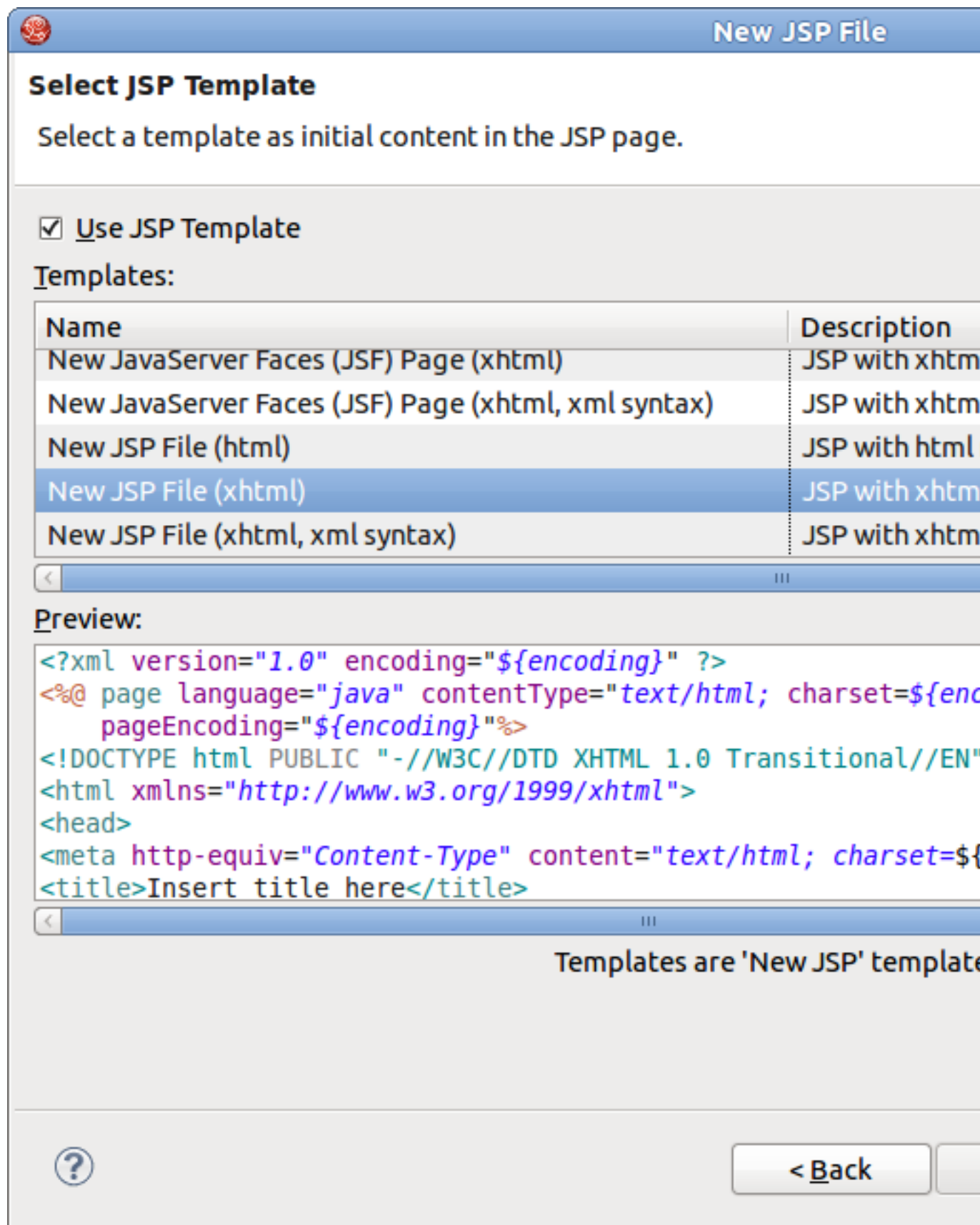


Figure 4.3. Create JSP Page

Our `hello.jsp` page will now appear in the Project Explorer view.

4.2.1. Editing a JSP Page

Let's now make a little change so that a JSP page displays *"Hello World!"* message.

- Insert this line inside the `<body>` `</body>` tag:

```
<% System.out.println("Hello World!"); %>
```

Notice that content assist functionality is always available when you are typing:

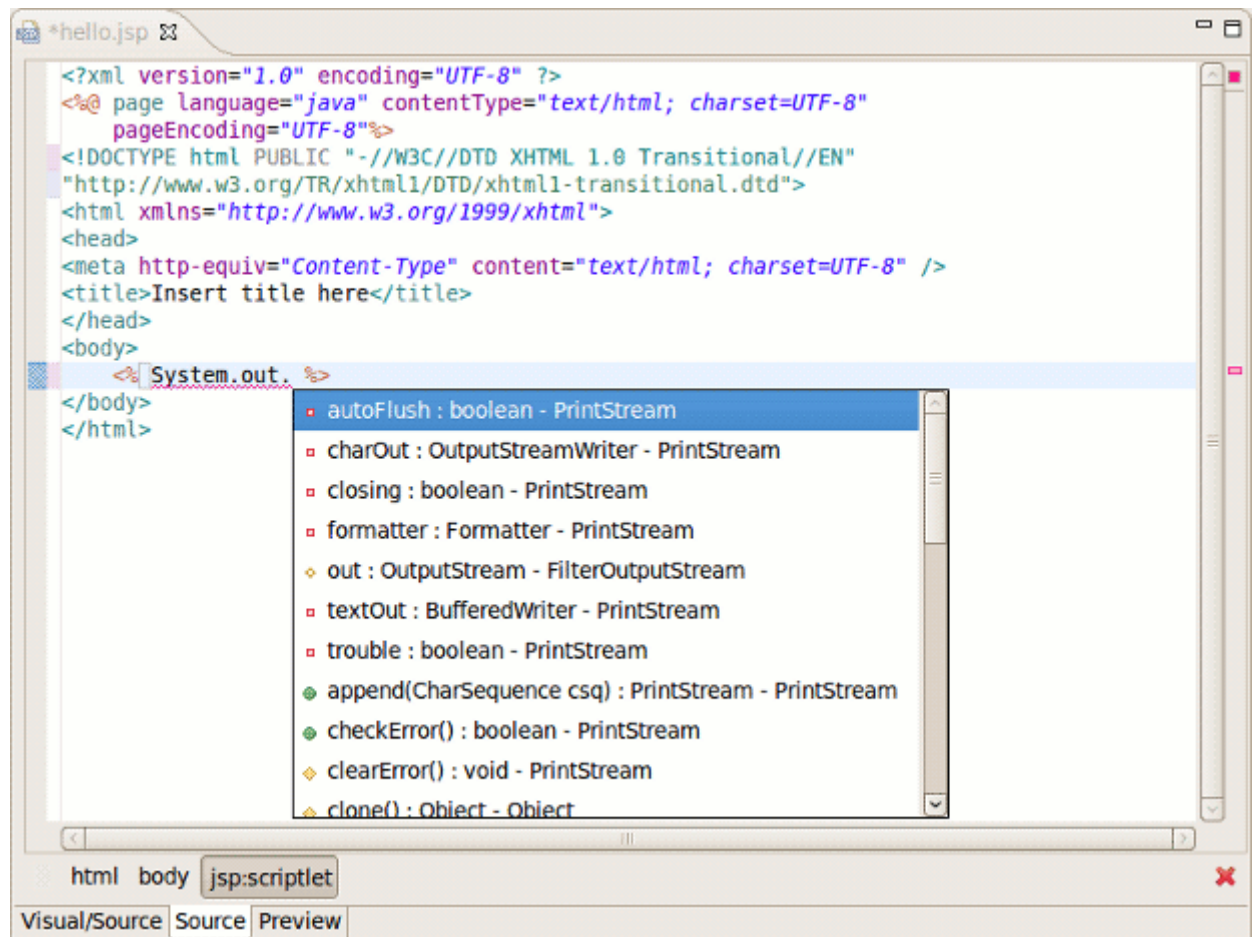


Figure 4.4. Content Assist in JSP Page

After changes made your `hello.jsp` page should look like this:

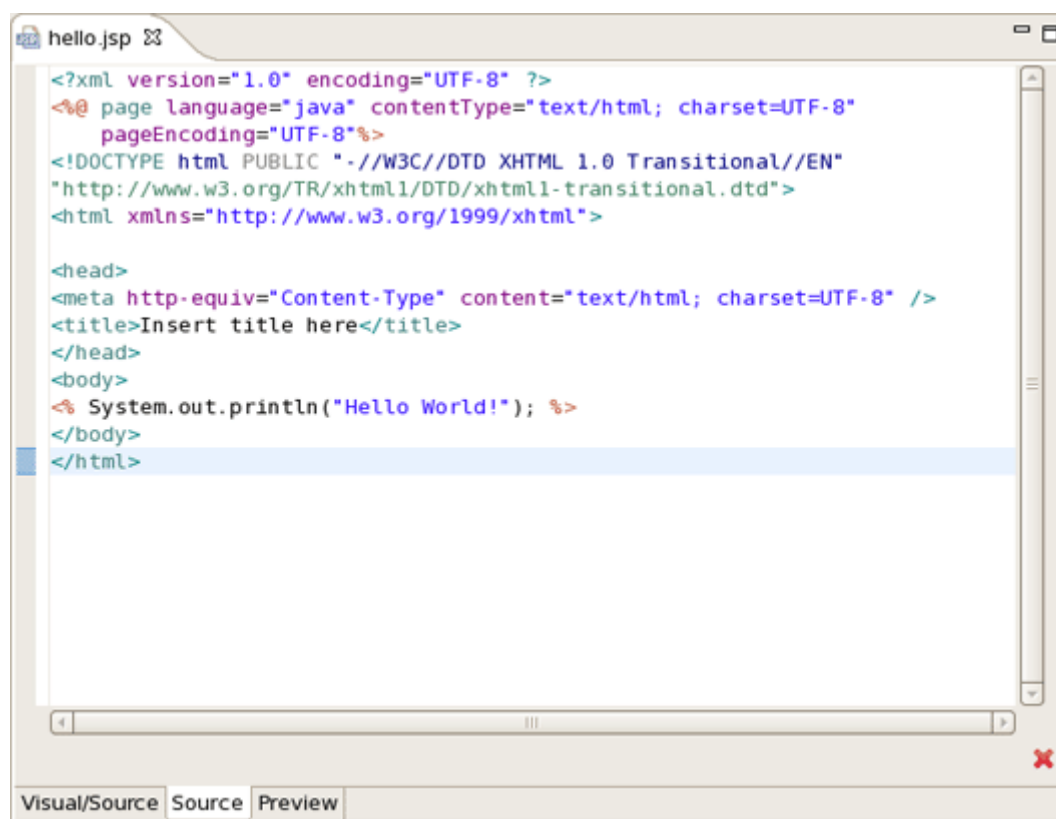


Figure 4.5. Hello.jsp Page Source

This line will actually output *"Hello World!"* message in the Console. To make the message displayed in the Browser, just replace this line with the simple *Hello World!*.

4.2.2. web.xml file

When you are creating web project the wizard creates the `web.xml` file for you automatically. The `web.xml` file editor provided by JBoss Developer Studio is available in two modes: Tree and Source.

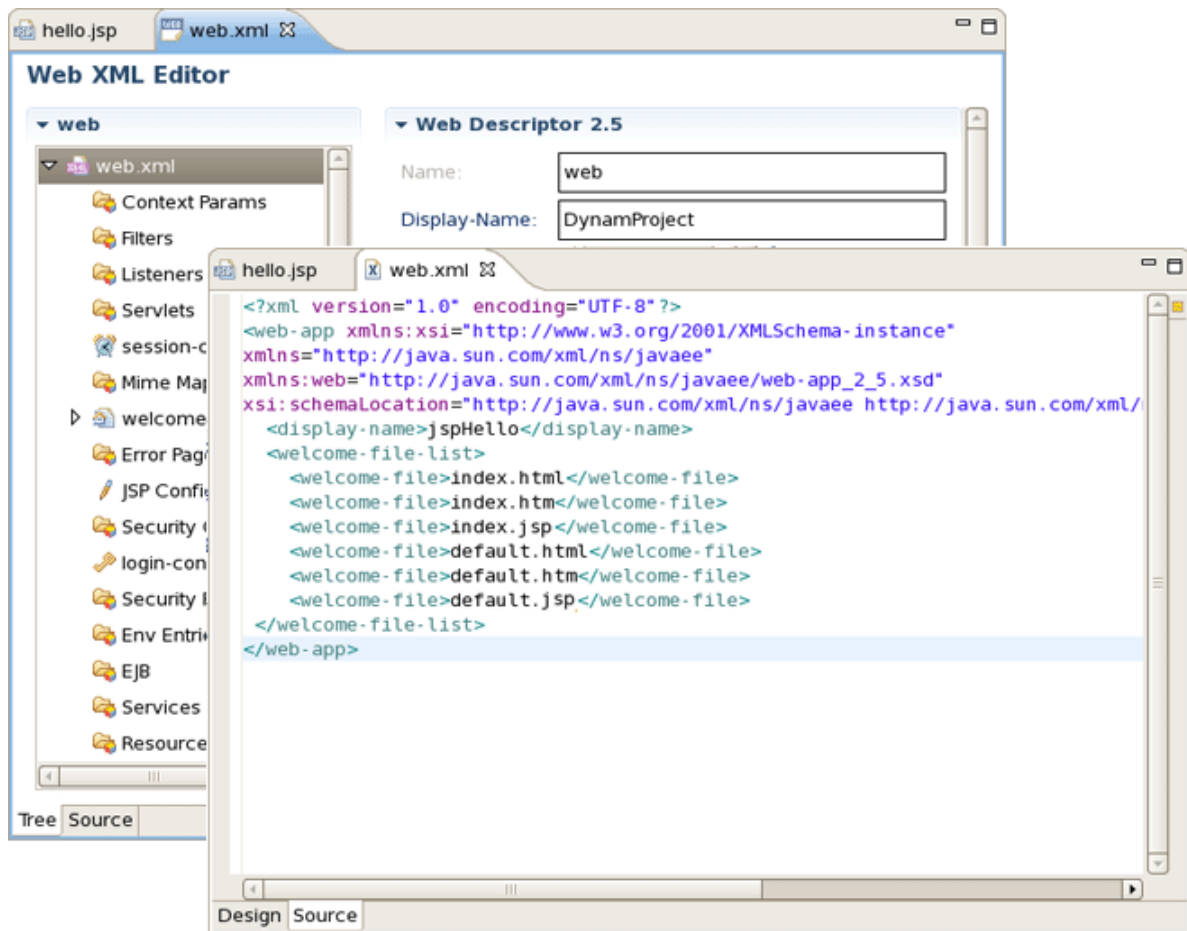


Figure 4.6. Web.xml in Design and Source Mode

Both modes are fully synchronized. Let's add a mapping to our `hello.jsp` page in the `web.xml` file.

- Switch to the Source tab.
- Add the next code into `<welcome-file-list>` :

```
<welcome-file>hello.jsp</welcome-file>
```

If you go back to Tree tab you will see that the changes made in the Source tab are automatically reflected.

Actually you do not really need to do any configurations right now.

4.2.3. Deploying the project

Writing ant scripts and managing the packaging process can be quite a complicated and time consuming task for even the most trivial web applications. However, JBoss Developer Studio relieves you of this burden. All you need is to start JBoss Server and launch your application in your favorite browser.

You can also create a JAR archive with JBoss Developer Studio's Archive Tools and export it to any web server.

4.2.3.1. JAR Config

Project archives managing is available through Project Archives view.

- Select **Window** → **Show view** → **Other** → **JBoss Tools** → **Project archives** from the menu bar
- Select a project in Package Explorer you want to be archived

In the Project Archives view you will see the that the project is now listed:

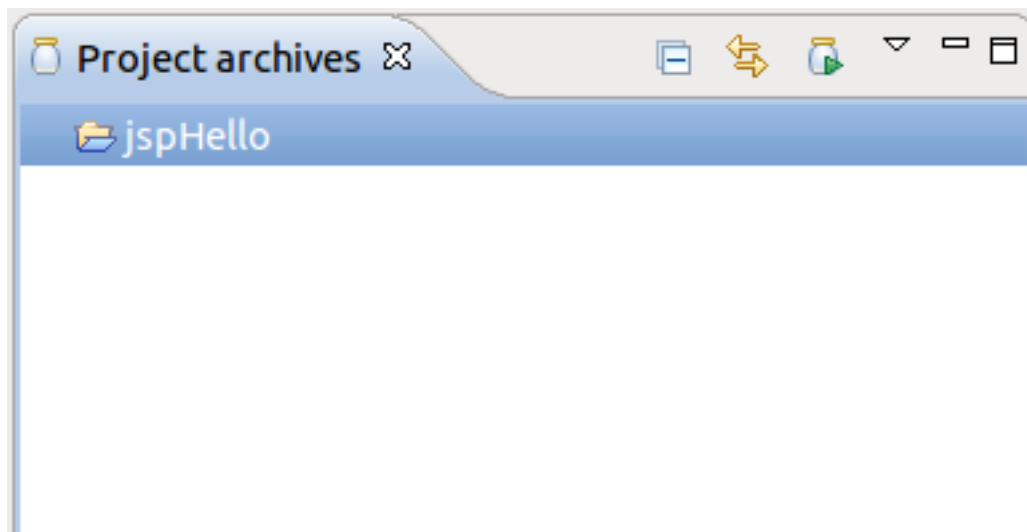


Figure 4.7. Project Archives

Right click on the project and select the JAR type of archive.

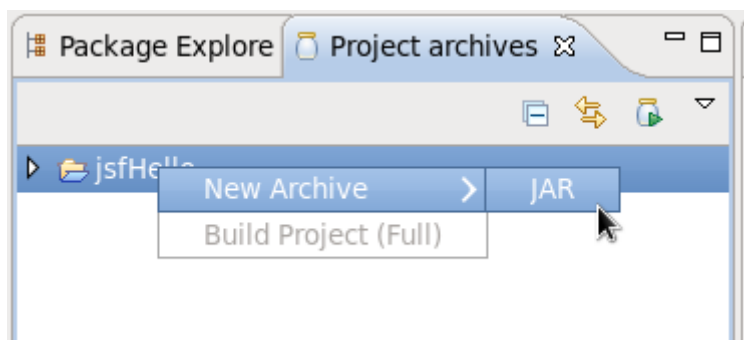


Figure 4.8. Project Archives

In the New JAR dialog you can see automatically selected default values.

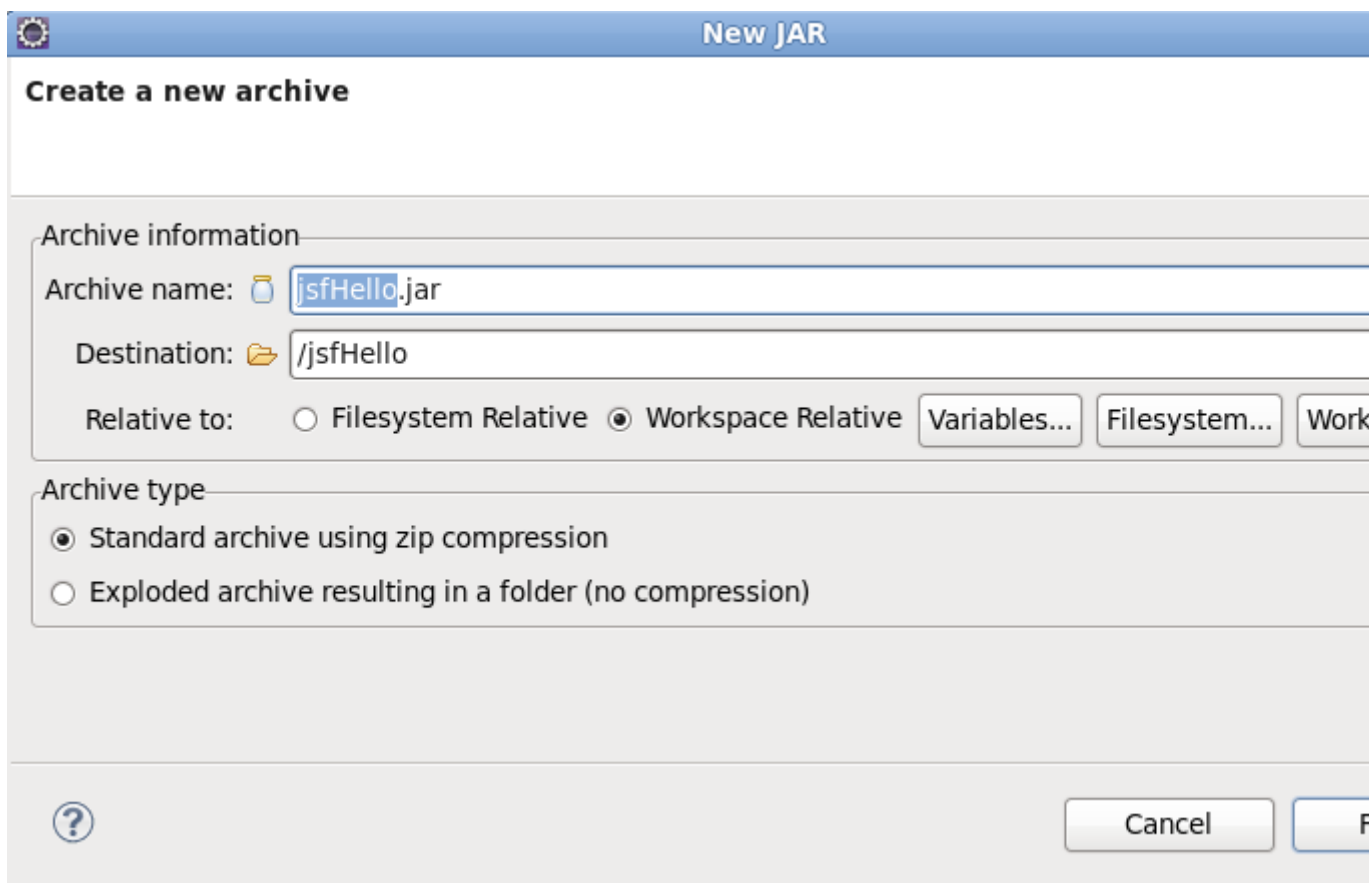


Figure 4.9. New JAR Archive

- Click the **Finish** button. The **.JAR** file will appear in Package Explorer and also in Project Archives view as structure tree:

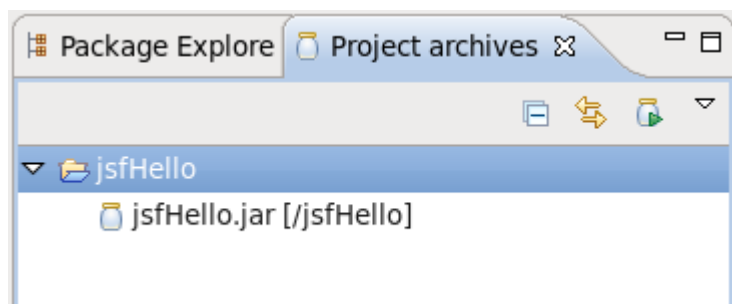


Figure 4.10. Archive is Created

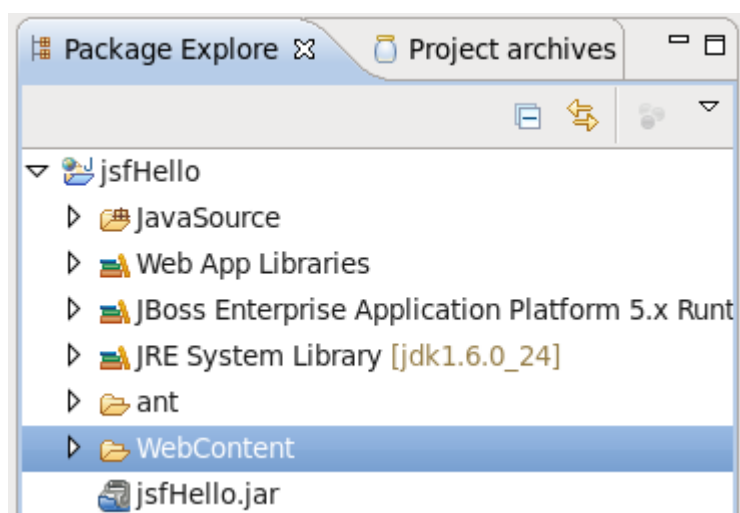


Figure 4.11. Archive in Package Explorer

Using the Project Archives view you can rebuild the archive:

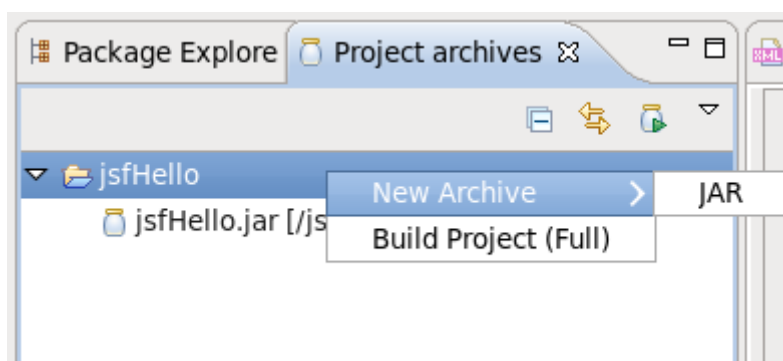


Figure 4.12. Configure Archive

4.2.3.2. Auto redeploy

When you are creating a web application and register it on JBoss Server it is automatically deployed into the `/deploy` directory of the server. JBoss Developer Studio comes with the feature

of auto-redeploy. It means that you don't need to restart JBoss Server. Any changes made in the application in exploded format will trigger a redeployment on the server.

You can also use the "Finger touch" button for a quick restart of the project without restarting the server:



Figure 4.13. Finger Touch button

The "Finger" touches descriptors dependent on project (i.e. web.xml for WAR, application.xml for EAR, jboss-esb.xml in ESB projects).

4.2.4. JSP Page Preview

JBoss Developer Studio comes with JSP design-time preview features. When designing JSP pages you can easily preview how they will look during runtime. You can even attach your stylesheet to the Preview.


- Make a little change to `hello.jsp` page, e.g. put this code snippet:

```
<%= new java.util.Date() %>
```

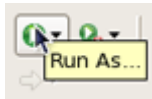
- Click the **Save** button.
- Switch to Preview page by clicking the Preview tab at the bottom of the page. You will see how the page will look at runtime.

4.2.5. Launch JSP Project

Let's now launch our project on server. We'll use JBoss Server that is shipped with JBoss Developer Studio. You can do it by performing one of the following actions:

- Start JBoss Server from Servers view by clicking the Start the server icon ().
- Click the **Run** icon or right click your project folder and select **Run As** → **Run on Server**. If you haven't made any changes in the `web.xml` file or cleared it out you can launch

the application by right clicking the `hello.jsp` page and selecting **Run on the Server**(



).

You should see the next page in a Browser :



Figure 4.14. Running Project

Thus with the help of this chapter you've learnt how to organize a Dynamic Web Project with a minimal configuration, add new elements to it (in our case it's just one JSP page) and deploy and run it on the JBoss Server shipped with JBoss Developer Studio.

RAD development of a simple JSF application



Note:

We highly recommend developing in Seam. This chapter is for users who for some reason cannot use Seam.

In this chapter you will learn how to create a simple JSF application being based on the "RAD" philosophy. We will create the familiar Guess Number application. The game is played according to the following rules. You are asked to guess a number between 0 and 100. If the guess is correct, a success page is displayed with a link to play again. If the guess is incorrect, a message is printed notifying that a smaller or a larger number should be entered and the game continues.

We'll show you how to create such an application from scratch, along the way demonstrating the powerful features included in JBoss Developer Studio such as project templating, Visual Page Editor, code completion and others. You will design the JSF application and then run the application from inside JBoss Developer Studio using the bundled JBoss server.

5.1. Setting up the project

First, you should create a JSF 1.2 project using an integrated JBoss Developer Studio's new project wizard and predefined templates. Follow the next steps:

- In the Web Projects view (if it is not open select **Window** → **Show View** → **Others** → **JBoss Tools Web** → **Web Projects**) click **Create New JSF Project** button.

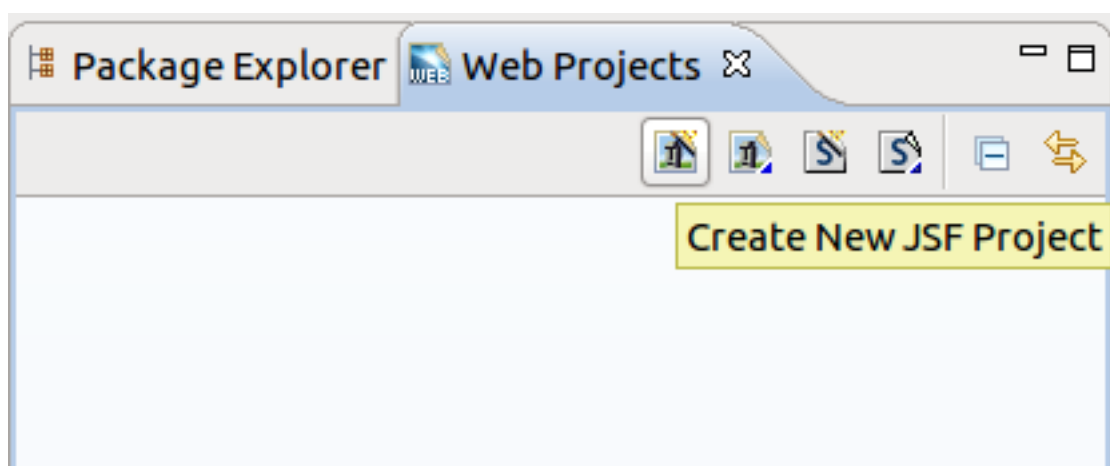


Figure 5.1. Create New JSF Project

- Enter GuessNumber as a project name, in JSF Environment drop down list choose JSF 1.2
- Leave everything else as it is and click the **Finish** button

Our project will appear in the Project Explorer and Web Projects views. As you can see JBoss Developer Studio has created the entire skeleton for the project with all required libraries, `faces-config.xml` file and `web.xml` file.

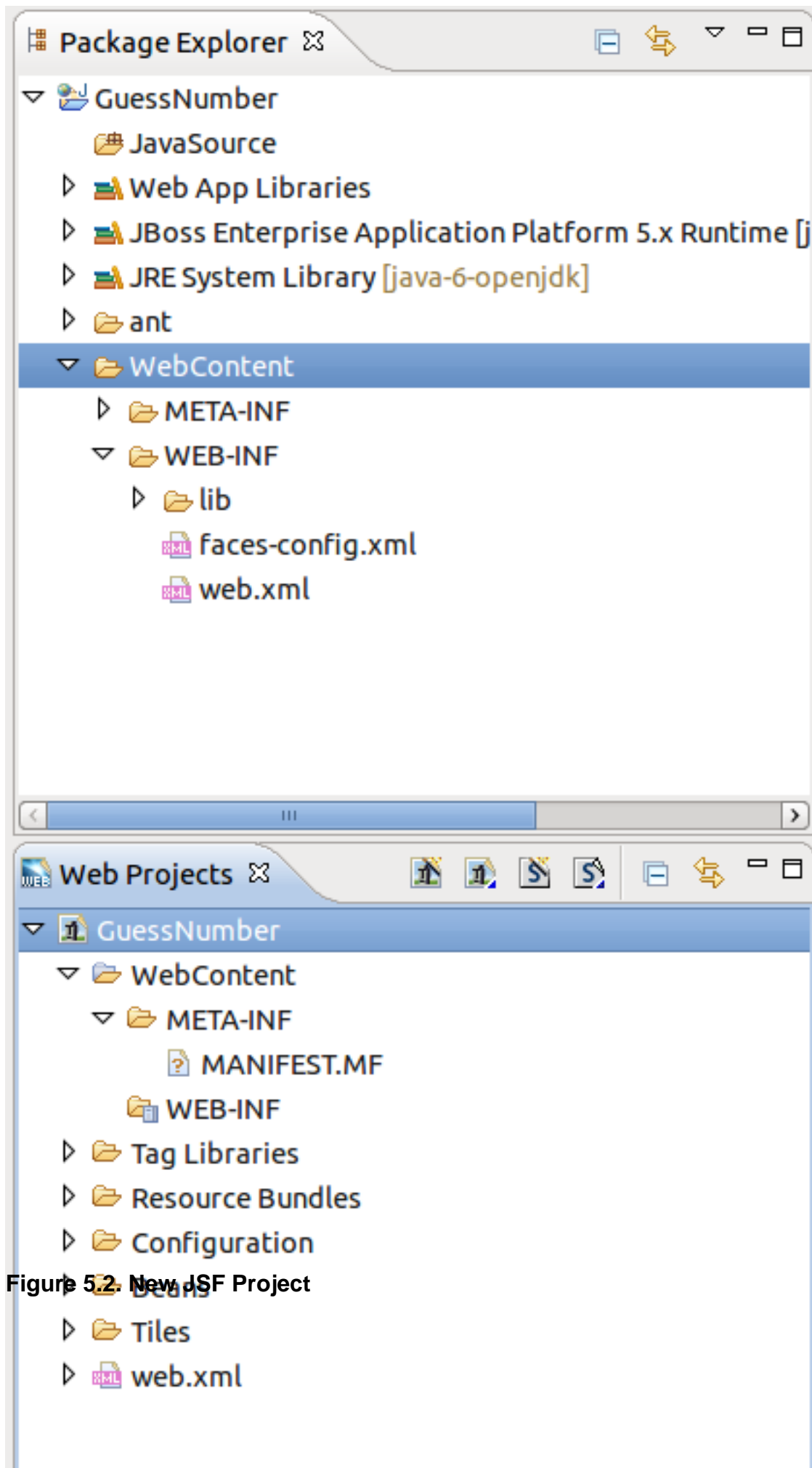


Figure 5.2. New JSF Project

As the project has been set up, new JSP pages should now be created.

5.2. Creating JSP Pages

Here, we are going to add two pages to our application. The first page is called `inputnumber.jsp`. It prompts you to enter a number. If the guess is incorrect, the same page will be redisplayed with a message indicating whether a smaller or a larger number should be tried. The second page is called `success.jsp`. This page will be shown after you guess the number correctly. From this page you also have the option to play the game again.

Now, we will guide you through the steps on how to do this.

- First a folder called `pages` needs to be created under the `WebContent` folder. To do this right click on the `WebContent` folder in the Package Explorer view and select **New** → **Folder**. Set the **Folder Name** to `pages` and click the **Finish** button.

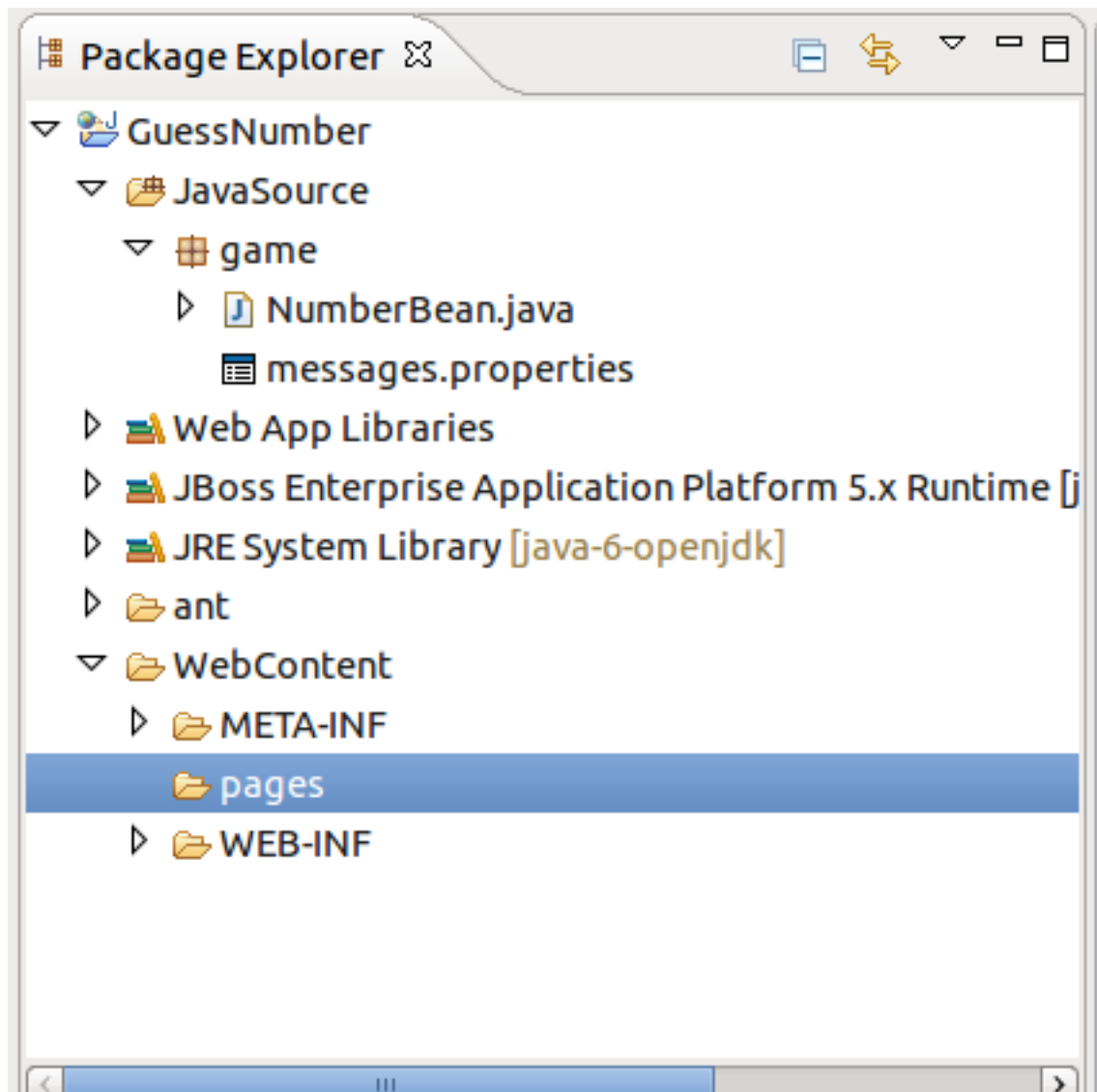


Figure 5.3. Create pages folder

- Open the `faces-config.xml` file.
- Right click anywhere on the diagram mode
- From the context menu select **New View**

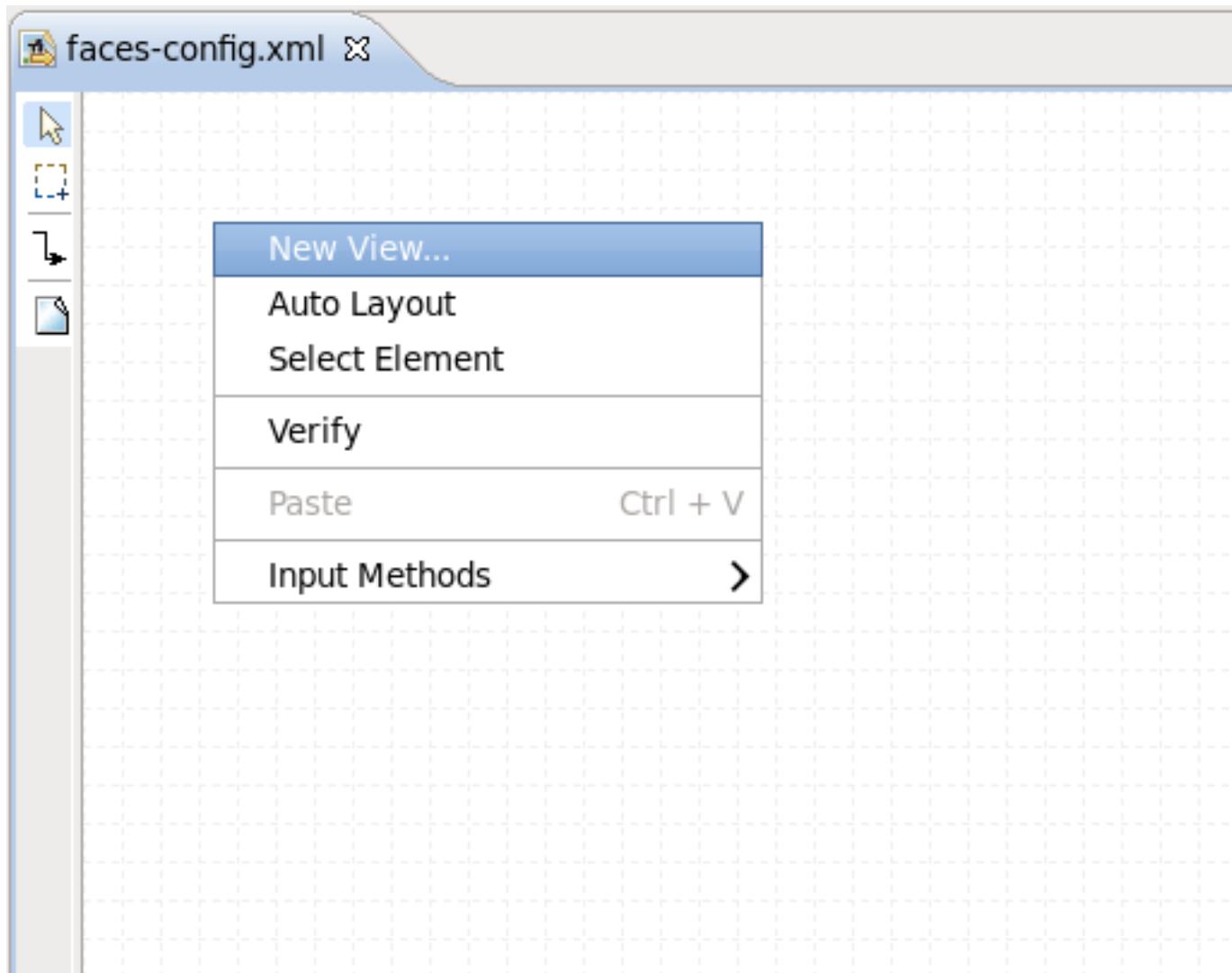


Figure 5.4. Create New View

- Type *pages/inputnumber* as the value for *the From View ID field*
- Leave everything else as is and click the **Finish** button
- In the same way create another JSF view. Type *pages/success* as the value for *From View ID*
- Select **File** → **Save**

On the diagram you will see two created views.

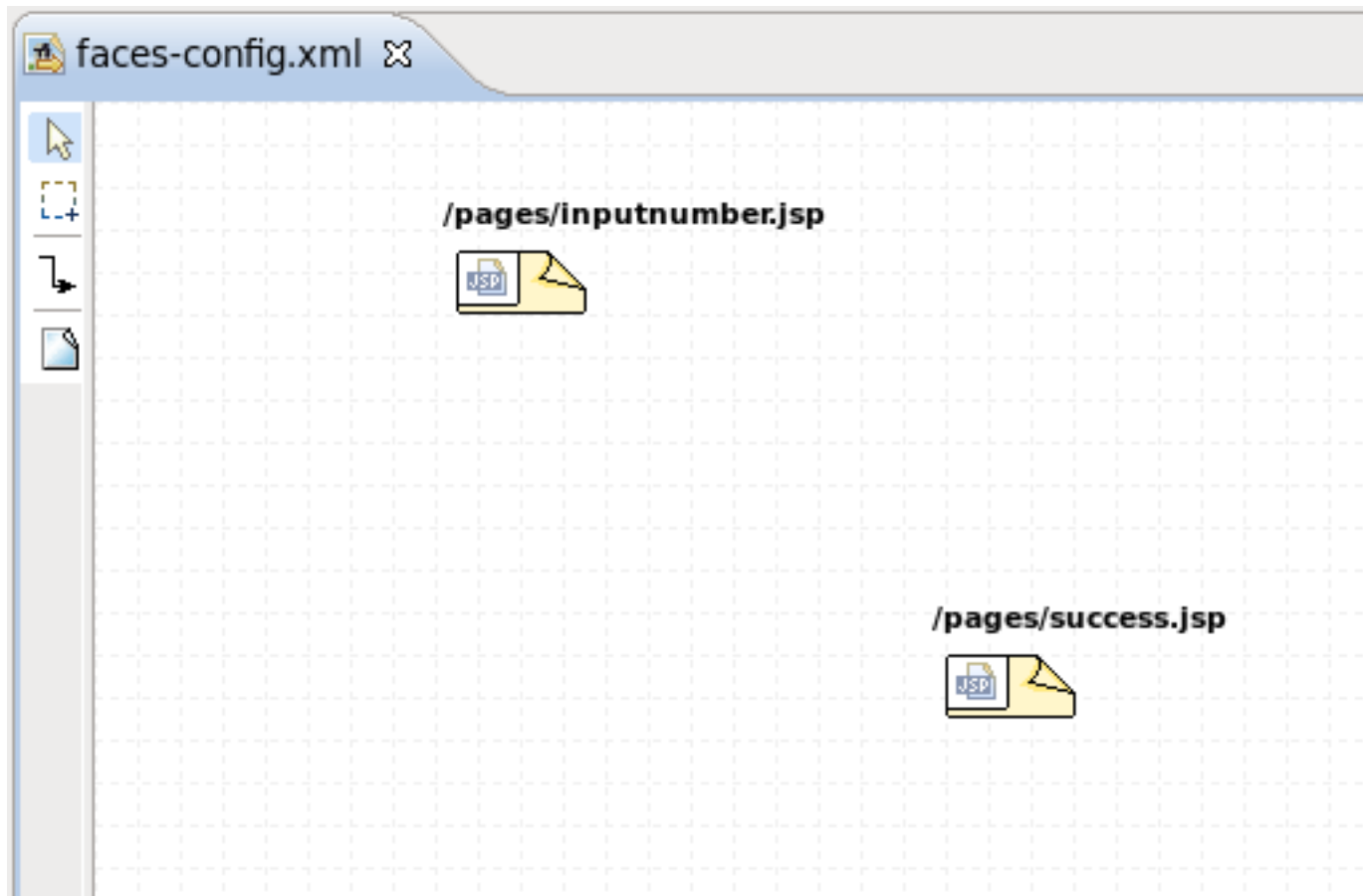


Figure 5.5. New Views

5.3. Creating Transition between two views

Then, we should create connection between JSP pages.

- In the diagram, select the **Create New Connection** icon third from the top along the upper left side of the diagram to get an arrow cursor with a two-pronged plug at the arrow's bottom

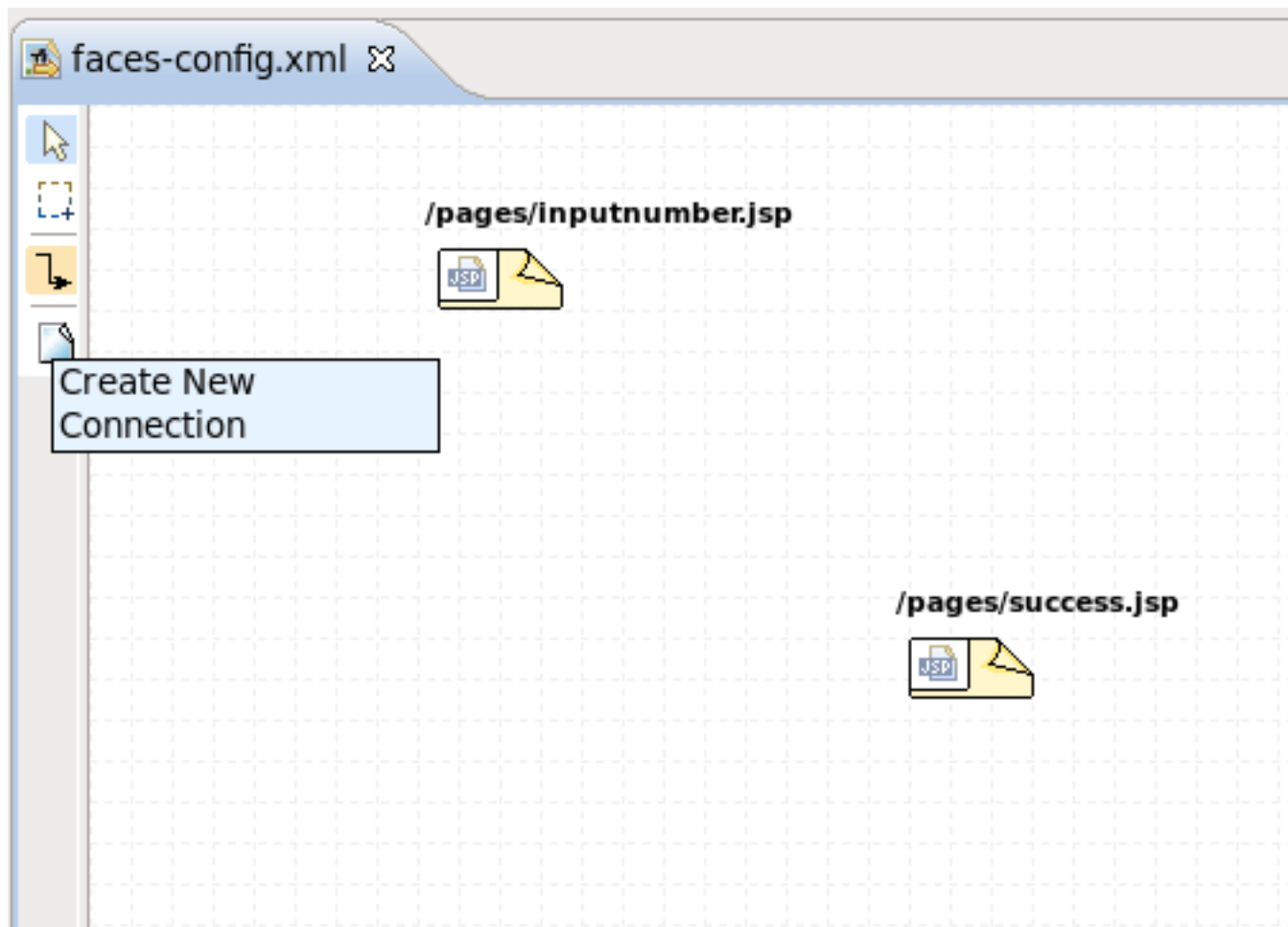


Figure 5.6. Create Connection

- Click on the *pages/inputnumber* page icon and then click on the *pages/success* page icon

A transition should appear between the two icons of views.

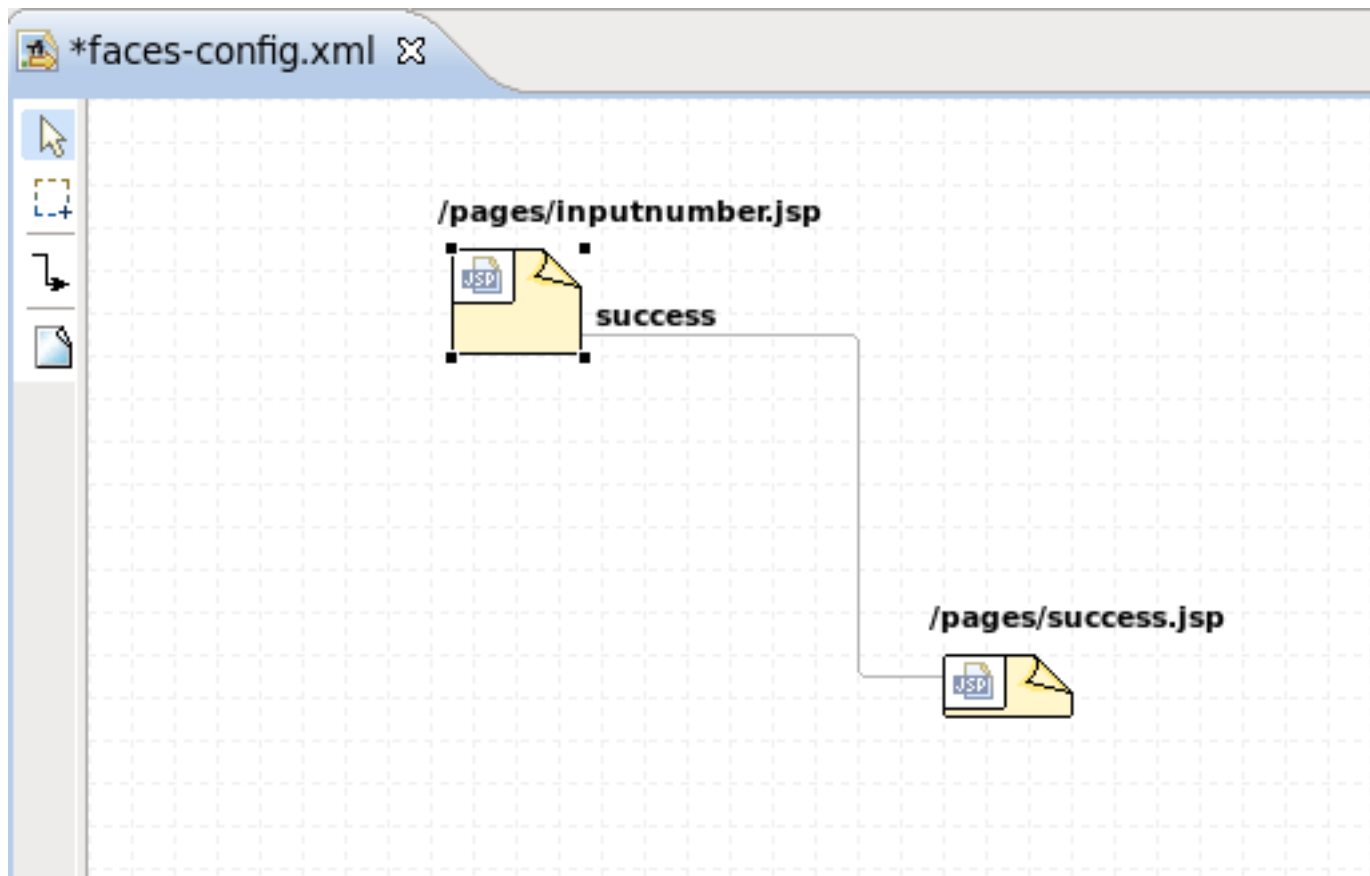


Figure 5.7. Created Connection

- Select **File** → **Save** from the menu bar

5.4. Creating Resource File

A resource file is just a file with a *.properties* extension for collecting text messages in one central place. JBoss Developer Studio allows you to create quickly a resource file. The messages stored in resource file can be displayed to you on a Web page during application execution.

With resource file you don't hard code anything into the JSP pages. It also makes it easier to translate your application to other languages. All you have to do is to translate all your messages to the other language and save them in a new properties file with a name that ends with the appropriate ISO-639 language code.

It is a good idea to keep your resources inside the `JavaSource` folder, where you keep your *.java* files. Every time you build the project, all *.properties* files will then be copied to the `classes` folder by default.

- Right click the `JavaSource` folder and select **New** → **Folder**

- Enter `game` as the Folder name and click the **Finish** button

Your resource file and java bean will be stored in this folder.

- Right click on the `game` folder and select **New** → **Properties File**
- Type `messages` as the value for "name" attribute and click the **Finish** button

JBoss Developer Studio will automatically open `messages.properties` file for editing.

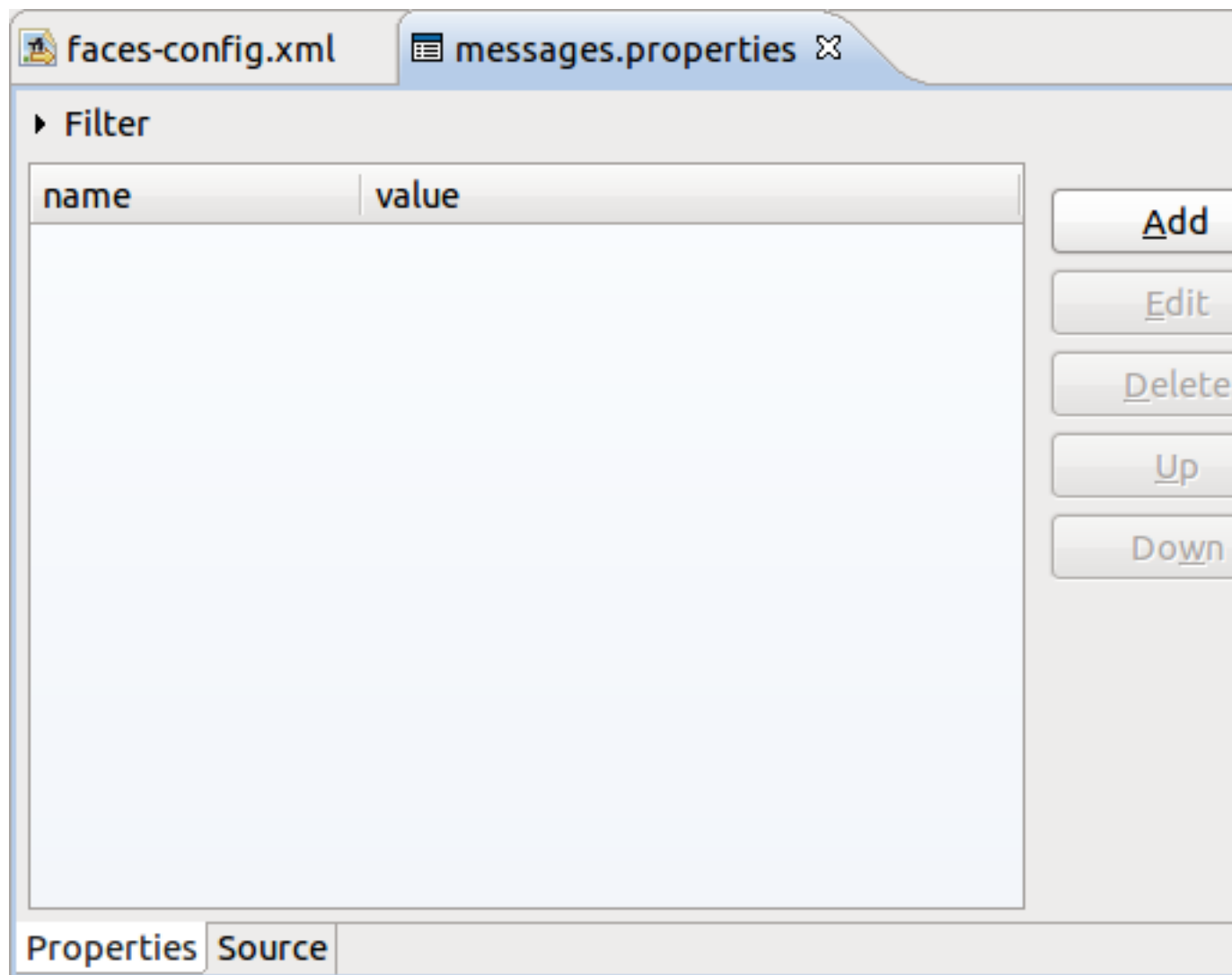


Figure 5.8. Messages.properties File

- Click the **Add** button for adding new attribute to your resource file
- Enter `how_to_play` for the "name" and `Please pick a number between 0 and 100.` for the value

- Click the **Finish** button
- Add the following properties using the same process:

```
makeguess_button=Make Guess
trayagain_button=Play Again?
success_text=How cool.. You have guessed the number, {0} is correct!
tryagain_smaller=Oops..incorrect guess. Please try a smaller number.
tryagain_bigger=Oops..incorrect guess. Please try a bigger number.
```

- Select **File** → **Save** from the menu bar

Your .properties file should now look like follows:

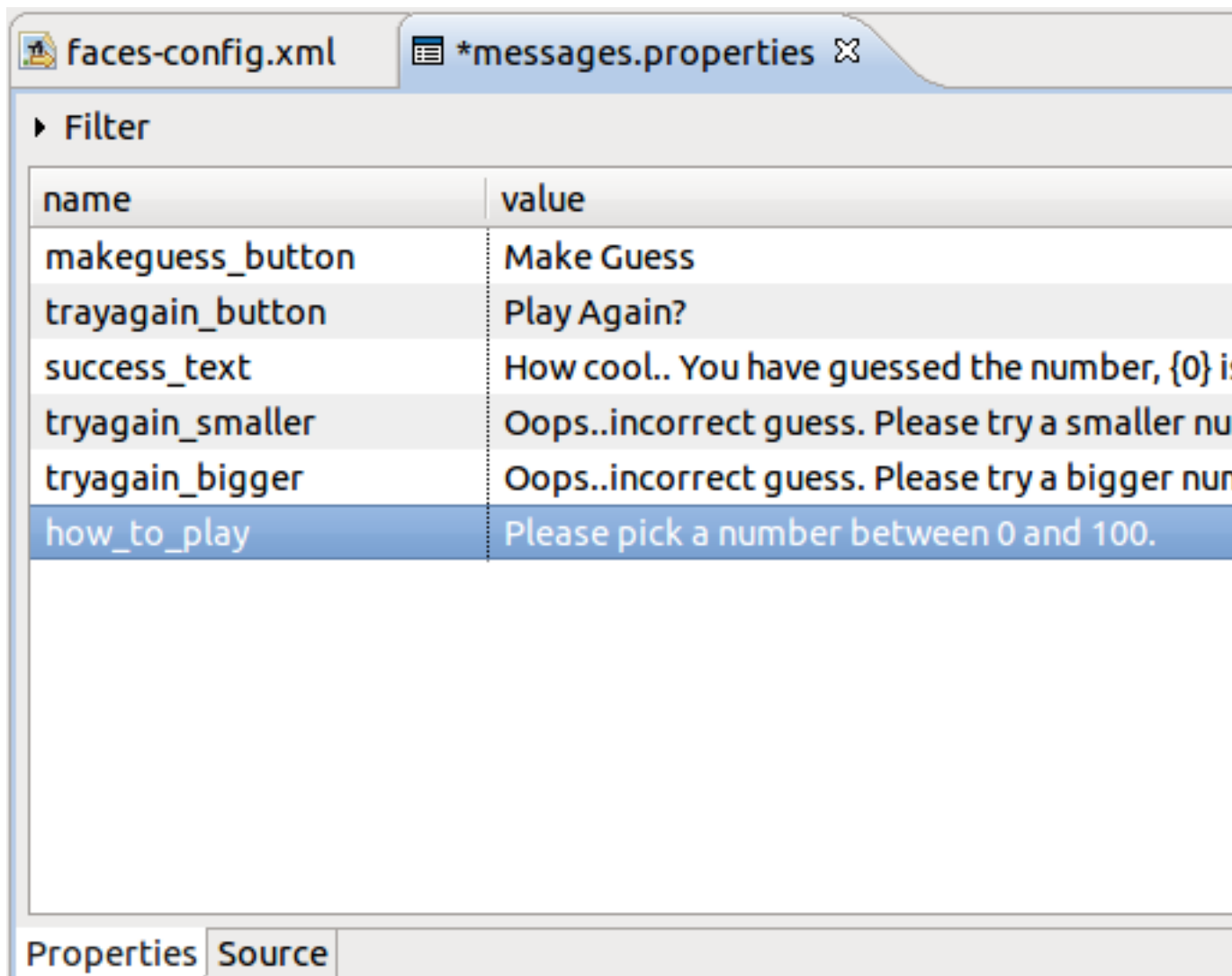


Figure 5.9. Properties are Added

The **Up** and **Down** buttons allow you to move the attributes in the list. To delete the attribute, select it and press the **Delete** button.

If you want to change a value or a name of your attribute, select it and then click the **Edit** button.

If the .properties file is rather big and there are a lot of entries in it, you can use filtering and regular expressions narrow down the list. The Filter and Regular Expressions Search is implemented by an expandable panel, closed by default:

When "Expression" is not selected (as by default), filter is case insensitive. When "Expression" is selected, filter uses regular expressions which are case sensitive

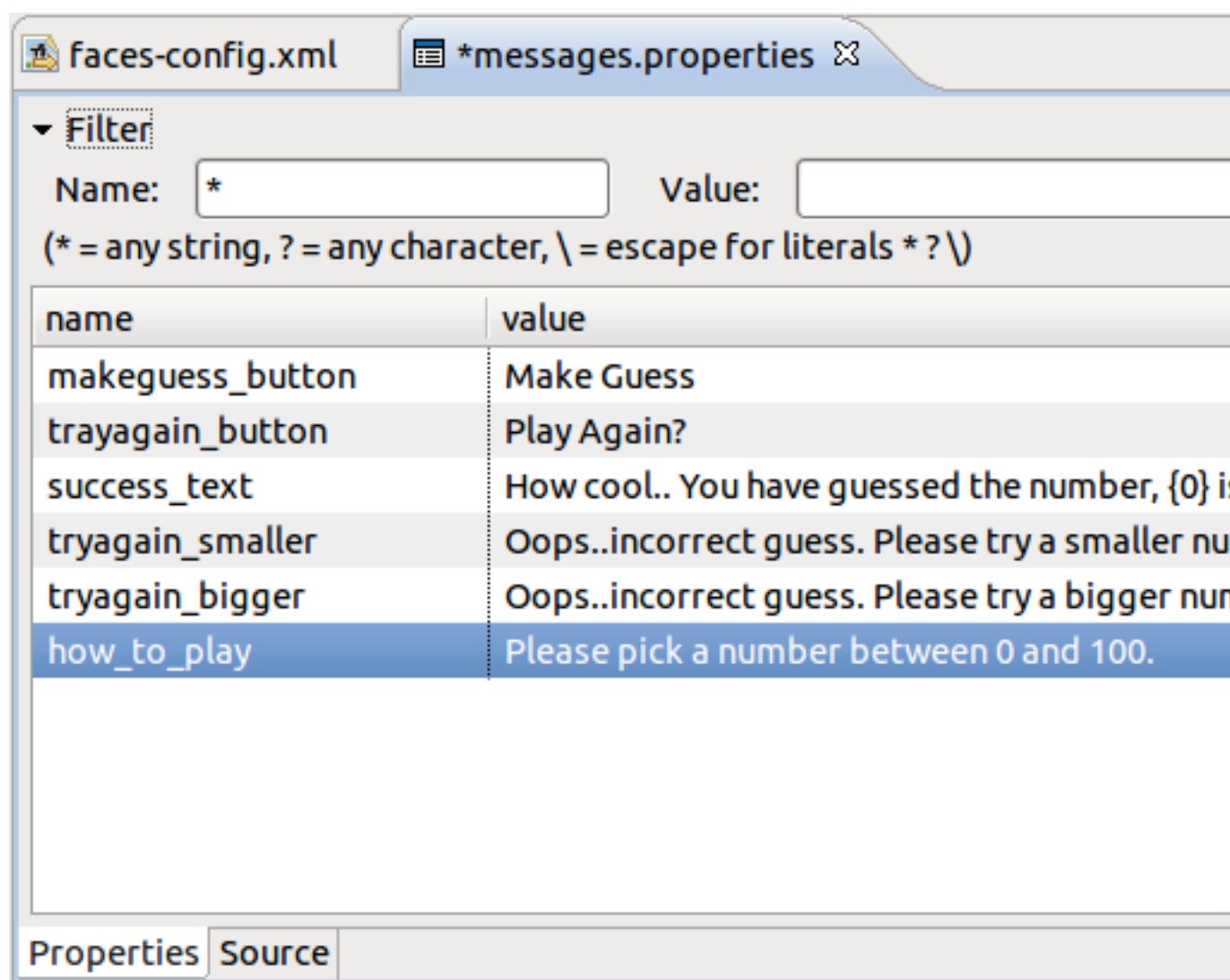


Figure 5.10. Filter and Regular Expressions Search Panel

Enter the characters that should be searched for in the entries to the 'name' or 'value' input fields accordingly. The filtered results will be displayed in the table below:

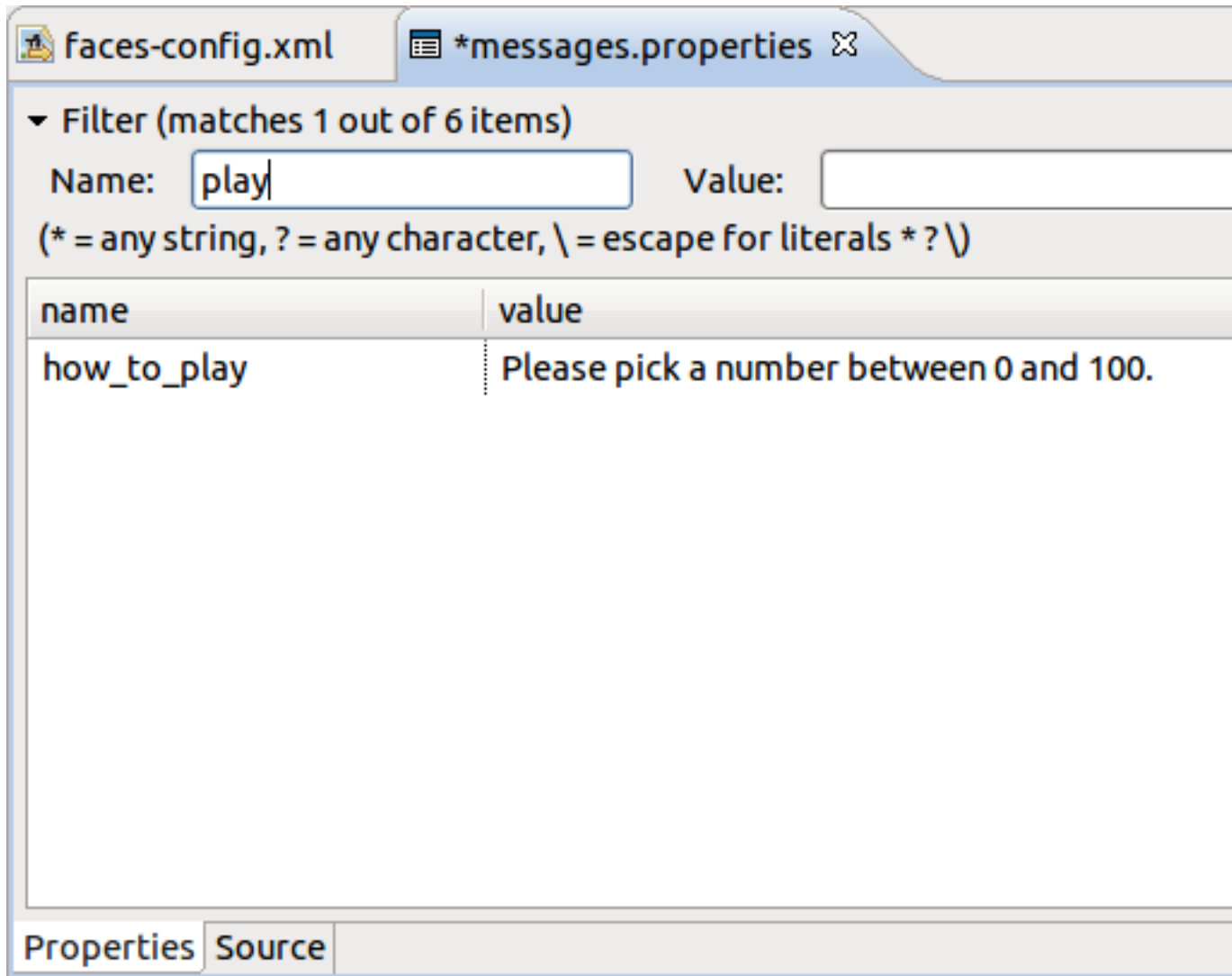


Figure 5.11. Filter results

When using regular expressions please note, that regular expression syntax does not use `"` for any characters and `?` for any one character. It's necessary to use `.` for any one character and `*` for any characters. Symbols `"` and `?` are used to show that the preceding token is not required, for example, `"a.a"` matches `"aba"` but not `"aa"`, while `"a.?a"` or `a.*a` matches both; besides `"a.*a"` matches `"abcda"`.

To find the exact match, use sequences `\A` and `\z` in expression. For example, expression `"\Adate\z"` matches only string `"date"`; expression `"\Adate"` matches `"date"` and `"dateline"`, expression `"date\z"` matches `"date"` and `"Begin date"`, and expression `"date"` matches all of them.

5.5. Creating Java Bean

In this section you'll learn how to create a Java bean that will hold business logic of our application.

- Right click the `game` folder

- Select **New** → **Class**
- Type *NumberBean* for bean name

A java bean is created.

- Declare the variable of your entered number:

```
Integer userNumber;
```

JBoss Developer Studio allows to quickly generate getters and setters for java bean.

- Right click the *NumberBean.java* file in the Package Explorer view
- Select **Source** → **Generate Getters and Setters...**
- Check *userNumber* box and click the **OK** button

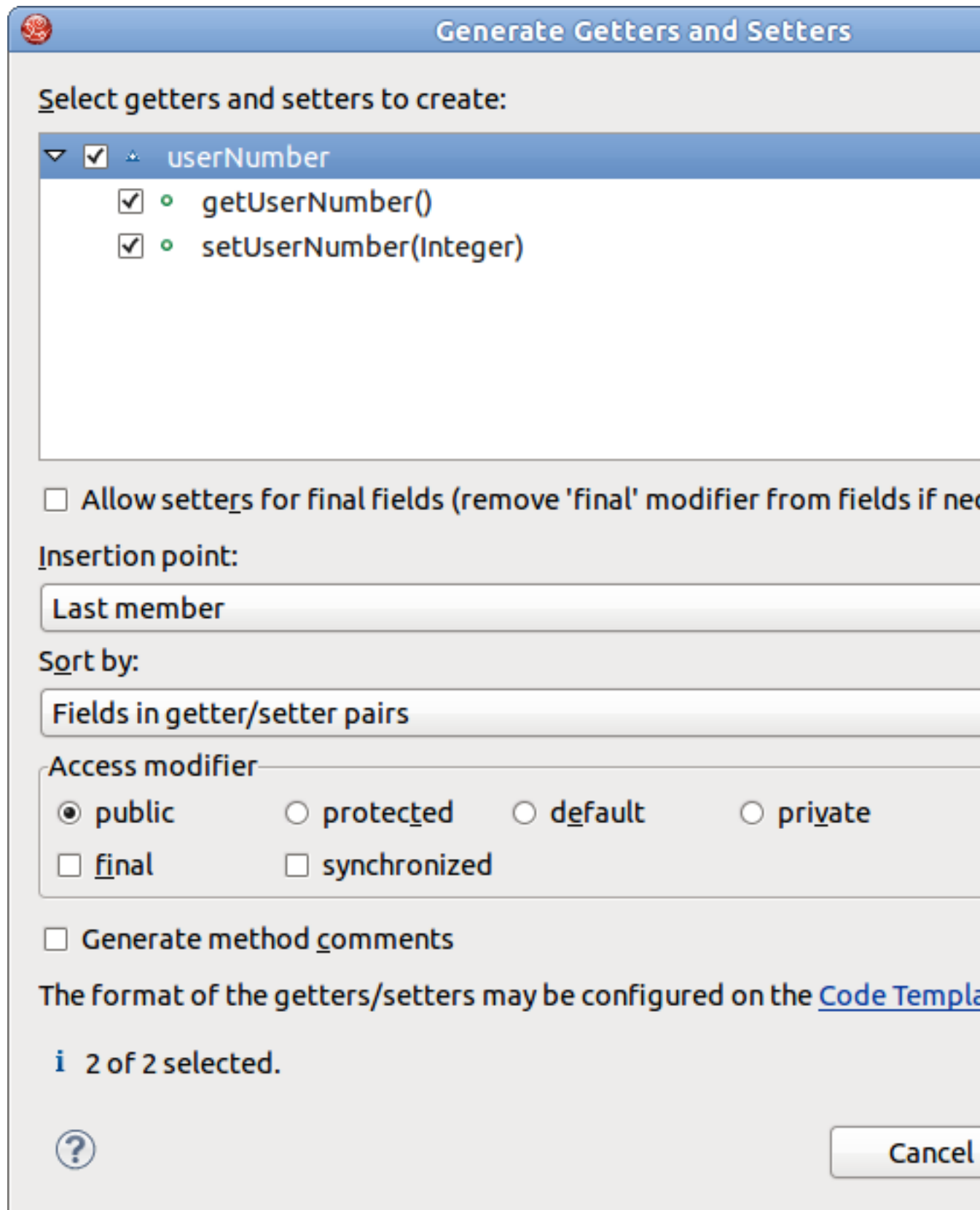


Figure 5.12. Generate Getters and Setters

- Add the declaration of the second variable

```
int randomNumber;
```

- .. other bean methods:

```
public NumberBean ()
{
    randomNumber = (int)(Math.random()*100);
    System.out.println ( "Random number: "+randomNumber);
}
public String playagain ()
{
    FacesContext context = FacesContext.getCurrentInstance();
    HttpSession session =
        (HttpSession) context.getExternalContext().getSession(false);
    session.invalidate();
    return "playagain";
}
public String checkGuess ()
{
    // if guessed, return 'success' for navigation
    if ( userNumber.intValue() == randomNumber )
    {
        return "success";
    }
    else
    {
        FacesContext context = FacesContext.getCurrentInstance();
        ResourceBundle bundle = ResourceBundle.getBundle("game.messages",
            context.getViewRoot().getLocale());
        String msg = "";
        // if number bigger, get appropriate message
        if ( userNumber.intValue() > randomNumber )
            msg = bundle.getString("tryagain_smaller");
        else // if number smaller, get appropriate message
            msg = bundle.getString("tryagain_bigger");
        // add message to be displayed on the page via <h:messages> tag
        context.addMessage (null, new FacesMessage(msg));
        // return 'tryagain' for navigation
        return "tryagain";
    }
}
```

- And the import declarations:

```
import javax.faces.context.FacesContext;
import javax.servlet.http.HttpSession;
import javax.faces.application.FacesMessage;
import java.util.ResourceBundle;
```

The whole java bean contain the following code:

```
package game;

import javax.faces.context.FacesContext;
import javax.servlet.http.HttpSession;
import javax.faces.application.FacesMessage;
import java.util.ResourceBundle;

public class NumberBean
{
    Integer userNumber;
    int randomNumber; // random number generated by application

    public Integer getUserNumber ()
    {
        return userNumber;
    }
    public void setUserNumber (Integer value)
    {
        this.userNumber = value;
    }

    // constructor, generates random number
    public NumberBean ()
    {
        randomNumber = (int)(Math.random()*100);
        System.out.println (
            "Random number: " + randomNumber);
    }

    public String playagain ()
    {
        FacesContext context = FacesContext.getCurrentInstance();
        HttpSession session =
            (HttpSession) context.getExternalContext().getSession(false);
        session.invalidate();
        return "playagain";
    }

    // check if user guessed the number
    public String checkGuess ()
```

```
{
    // if guessed, return 'success' for navigation
    if ( userNumber.intValue() == randomNumber )
    {
        return "success";
    }
    // incorrect guess
    else
    {
        // get a reference to properties file to retrieve messages
        FacesContext context = FacesContext.getCurrentInstance();
        ResourceBundle bundle =
            ResourceBundle.getBundle("game.messages",
                context.getViewRoot().getLocale());
        String msg = "";
        // if number is bigger, get appropriate message
        if ( userNumber.intValue() > randomNumber )
            msg = bundle.getString("tryagain_smaller");
        else // if number smaller, get appropriate message
            msg = bundle.getString("tryagain_bigger");

        // add message to be displayed on the page via <h:messages> tag
        context.addMessage (null, new FacesMessage(msg));
        // return 'tryagain' for navigation
        return "tryagain";
    }
}
}
```

5.6. Editing faces-config.xml File

In this section you will learn about the `faces-config.xml` file.

This file holds two navigation rules and defines the backing bean used.

- Open the `faces-config.xml` file in a source mode
- Here we will add one more navigation rule and a managed bean declaration, so that the content of the file looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config
    version="1.2"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xi="http://www.w3.org/2001/XInclude"
```

```

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-facesconfig_1_2_.xsd">

<navigation-rule>
  <from-view-id>*</from-view-id>
  <navigation-case>
    <from-outcome>playagain</from-outcome>
    <to-view-id>/pages/inputnumber.jsp</to-view-id>
  </navigation-case>
</navigation-rule>

<navigation-rule>
  <from-view-id>/pages/inputnumber.jsp</from-view-id>
  <navigation-case>
    <from-outcome>success</from-outcome>
    <to-view-id>/pages/success.jsp</to-view-id>
  </navigation-case>
</navigation-rule>

<managed-bean>
  <managed-bean-name>NumberBean</managed-bean-name>
  <managed-bean-class>game.NumberBean</managed-bean-class>
  <managed-bean-scope>session</managed-bean-scope>
</managed-bean>

</faces-config>

```

The first navigation rule states that from any page (* stands for any page) an outcome of playagain will take you to the `/pages/inputnumber.jsp` file. Outcome values are returned from backing bean methods in this example. The second navigation rule states that if you are at the page `/pages/inputnumber.jsp`, and the outcome is success, then navigate to the `/pages/success.jsp` page.

5.7. Editing the JSP View Files

Now, we will continue editing the JSP files for our two "views" using the Visual Page Editor.

5.7.1. Editing inputnumber.jsp page

First, edit the `inputnumber.jsp` file.

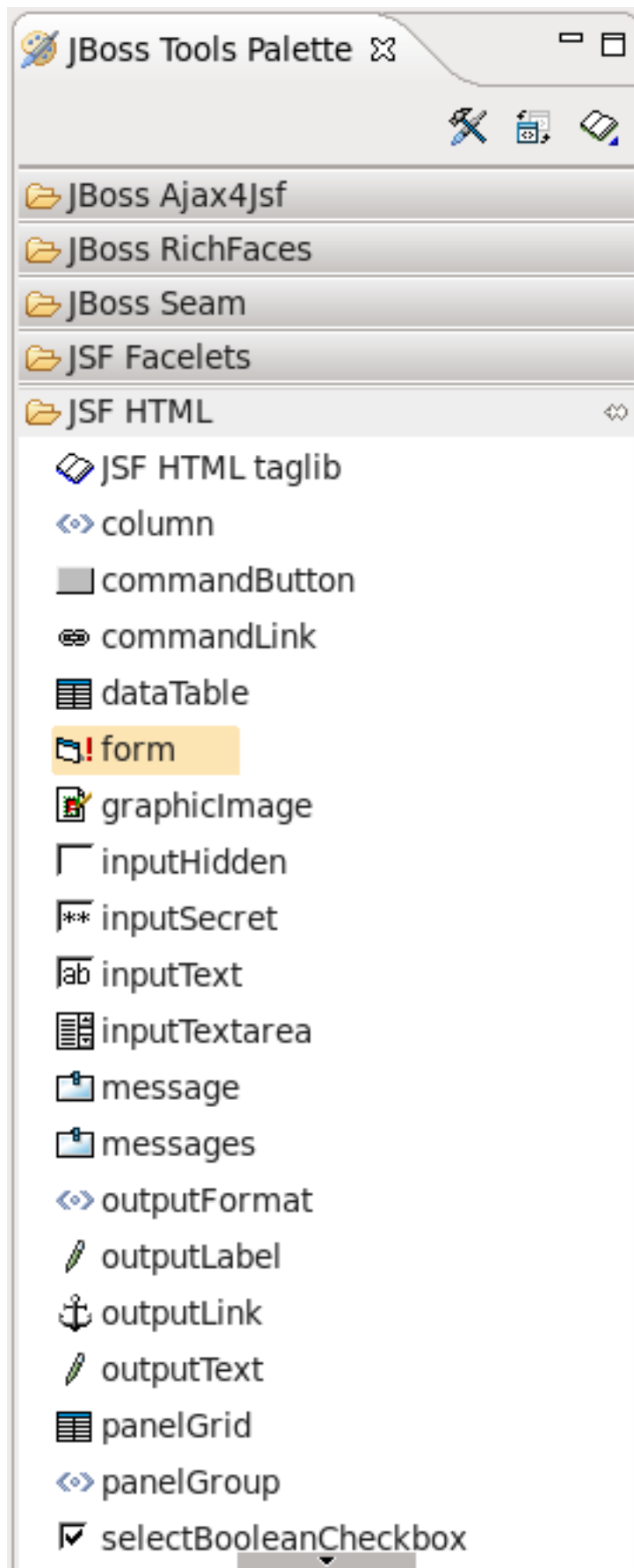
On this page we will have an output text component displaying a message, a text field for user's number entering and a button for input submission.

- Open the `inputnumber.jsp` file by double-clicking on the `/pages/inputnumber.jsp` icon

The Visual Page Editor will open in a screen split between source code along the top and a WYSIWIG view along the bottom. You can see that some JSF code will have already been generated since we chose a template when creating the page.

At the beginning it's necessary to create a `<h:form>` component that will hold the other components.

- Place the mouse cursor inside the `<f:view></f:view>` tag
- Go to JBoss Tools Palette and expand JSF HTML folder by selecting it
- Click on the `<h:form>` tag



- In the Insert Tag dialog select the *id* field and click on the second column. A blinking cursor will appear in a input text field inviting to enter a value of id

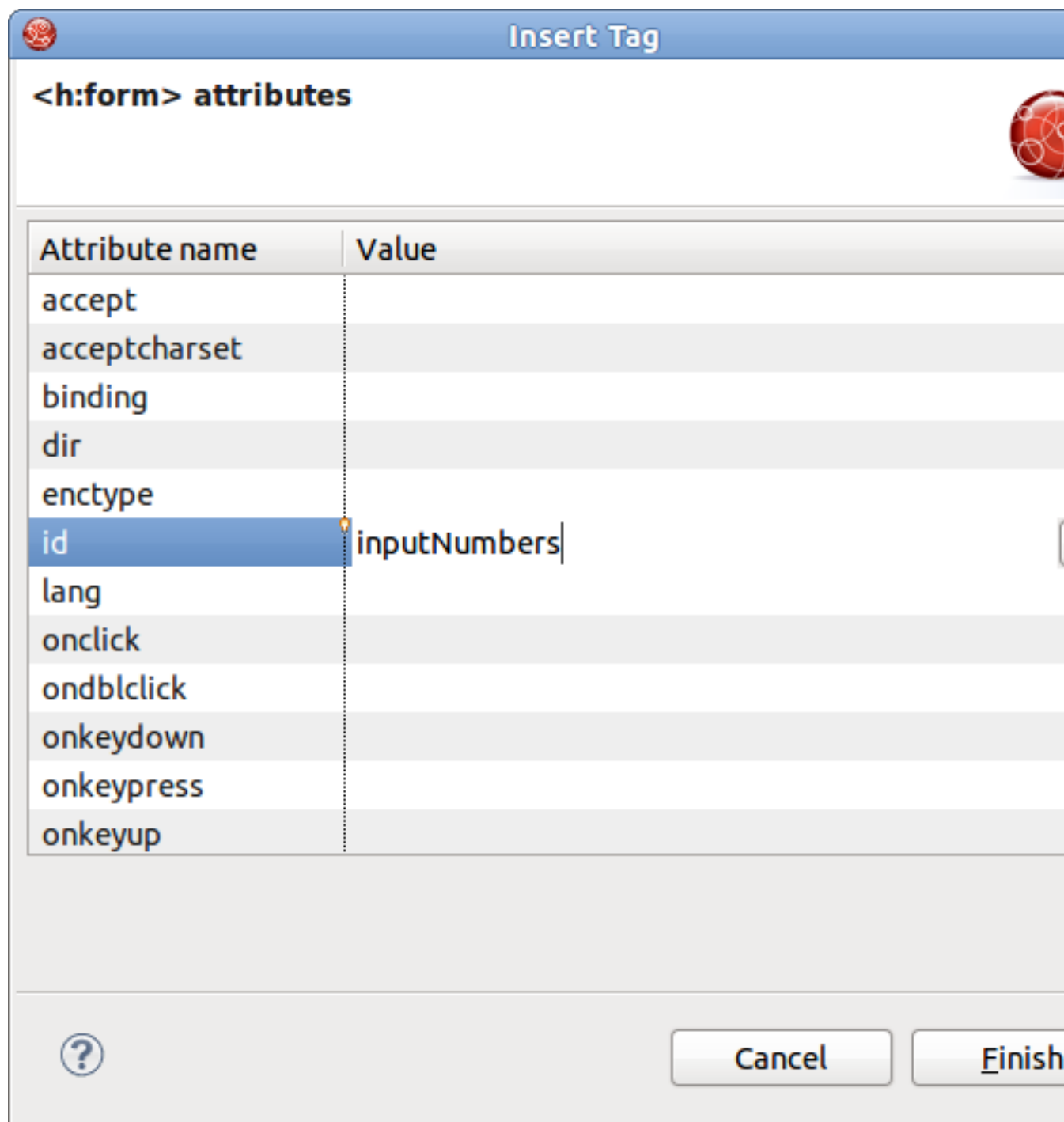


Figure 5.14. Define Id of Form

- Enter *inputNumbers* and click the **Finish** button

In source view you can see the declaration of a form.

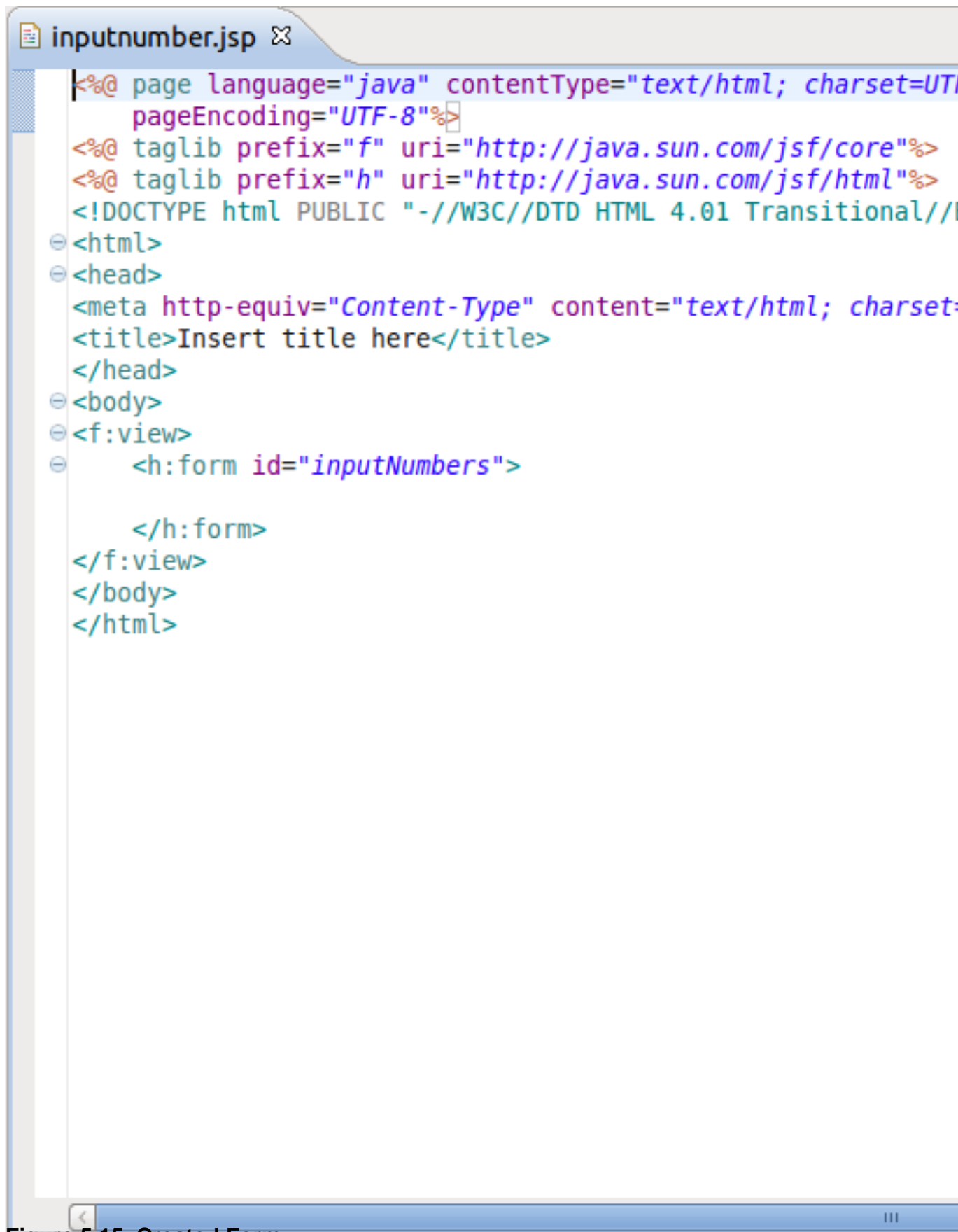


Figure 5.15. Created Form

First let's declare the properties file in the `inputnumber.jsp` page using the `loadBundle` JSF tag.

- Add this declaration on the top of a page, right after the first two lines:

```
<f:loadBundle basename="game.messages" var="msg" />
```

As always JBoss Developer Studio provides code assist:

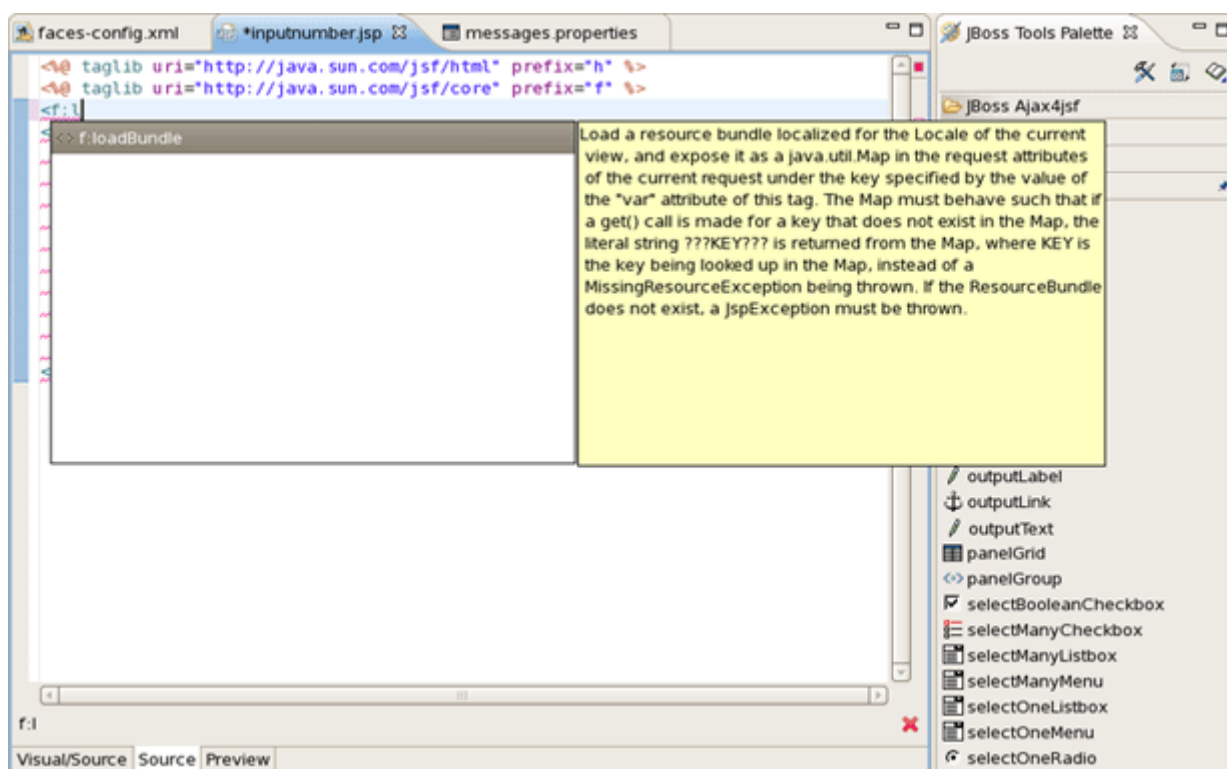


Figure 5.16. Code Assist

- Switch to Visual tab, where it is possible to work with the editor through a WYSIWYG interface
- Click the `outputText` item from the **JSF HTML** group in the **JBoss Tools Palette** view, drag the cursor over to the editor, and drop it inside the blue box in the editor
- Select the second column in the value row.
- Click the ... button next to the value field

JBoss Developer Studio will display a list of possible values:

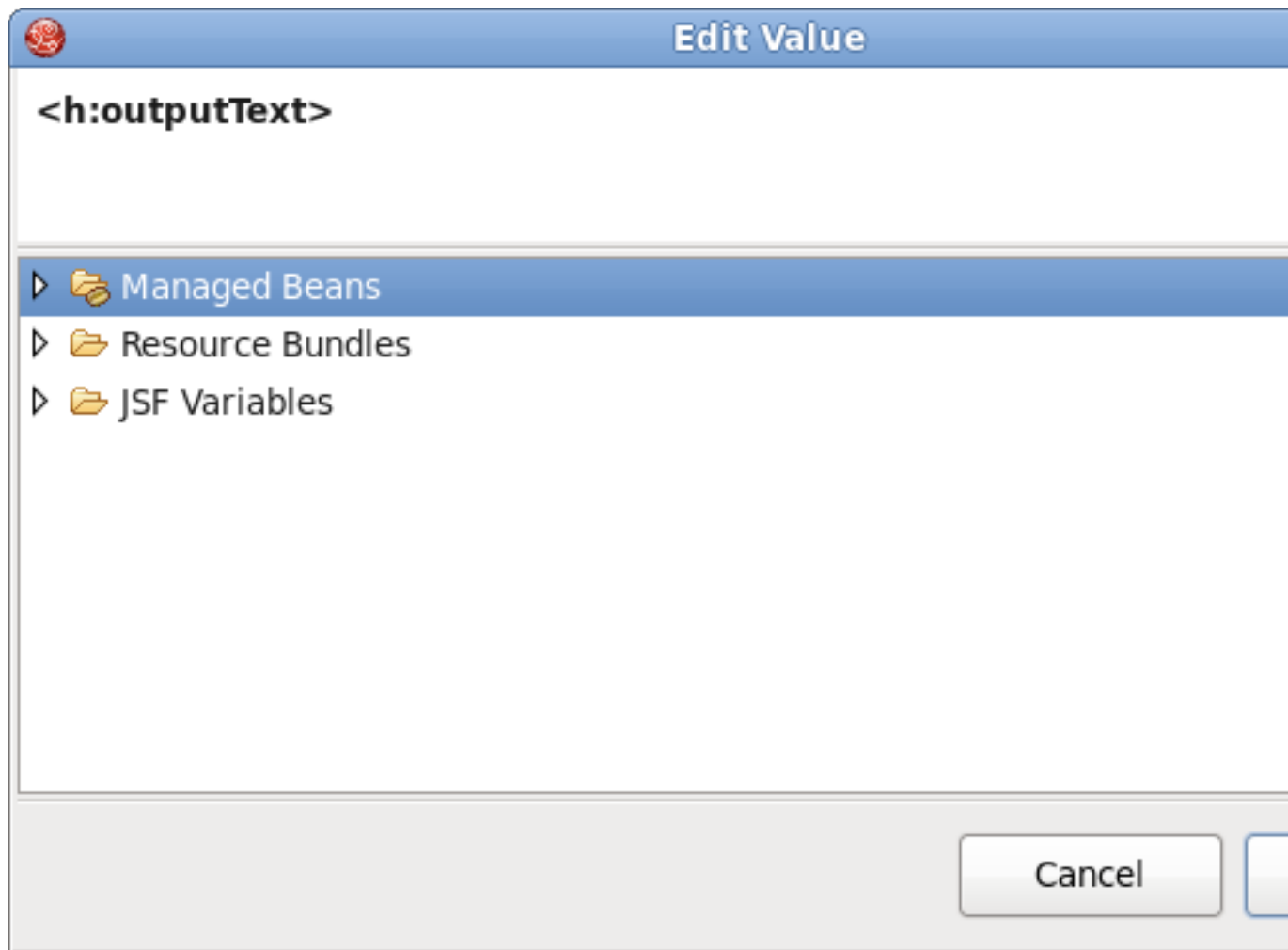


Figure 5.17. Choose Value

- Expand **Resource Bundles** → **msg**
- Select the *how_to_play* value and click the **OK** button. Then click the **Finish** button.

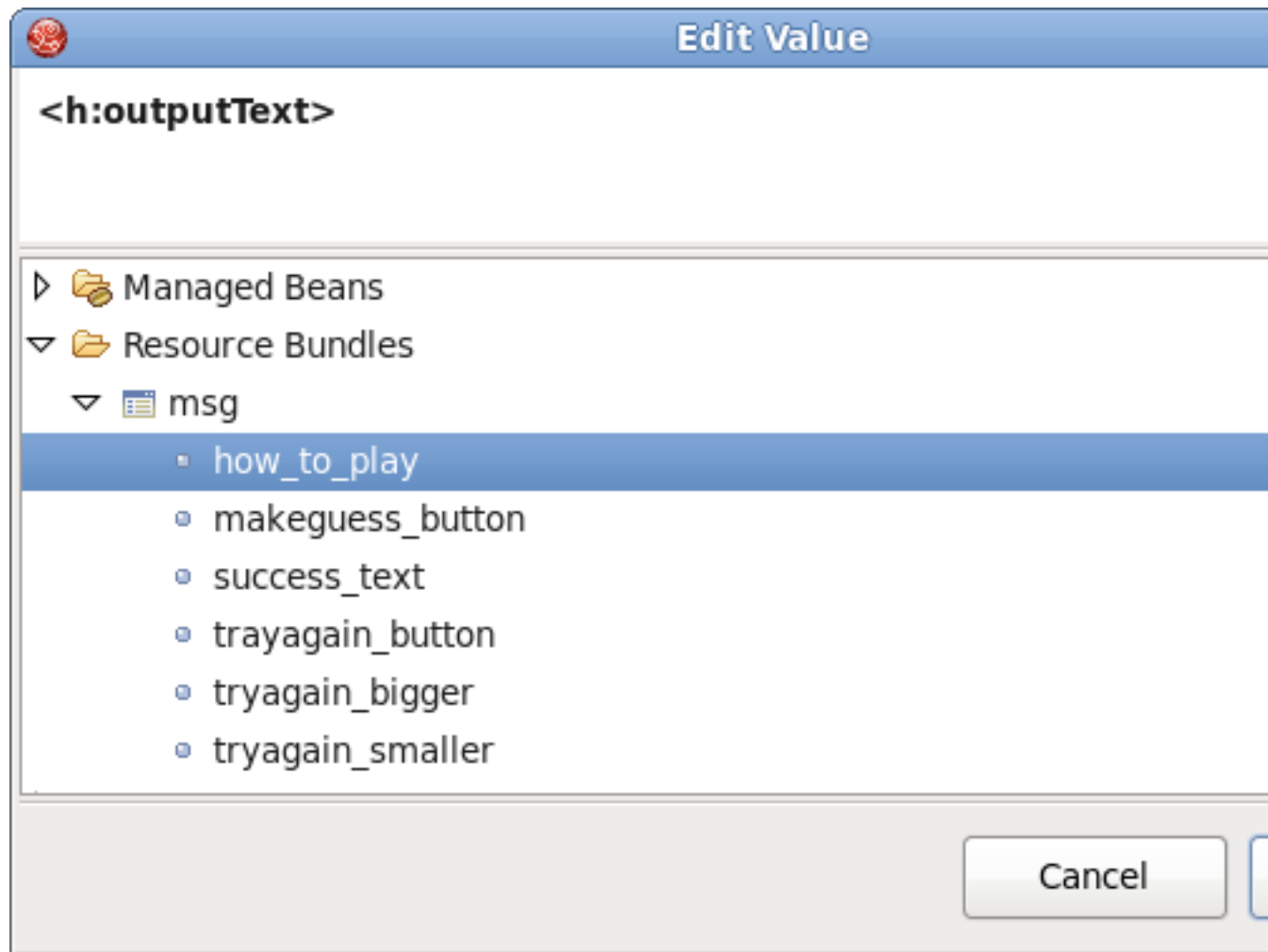


Figure 5.18. Selecting Value

The text will appear on the page:

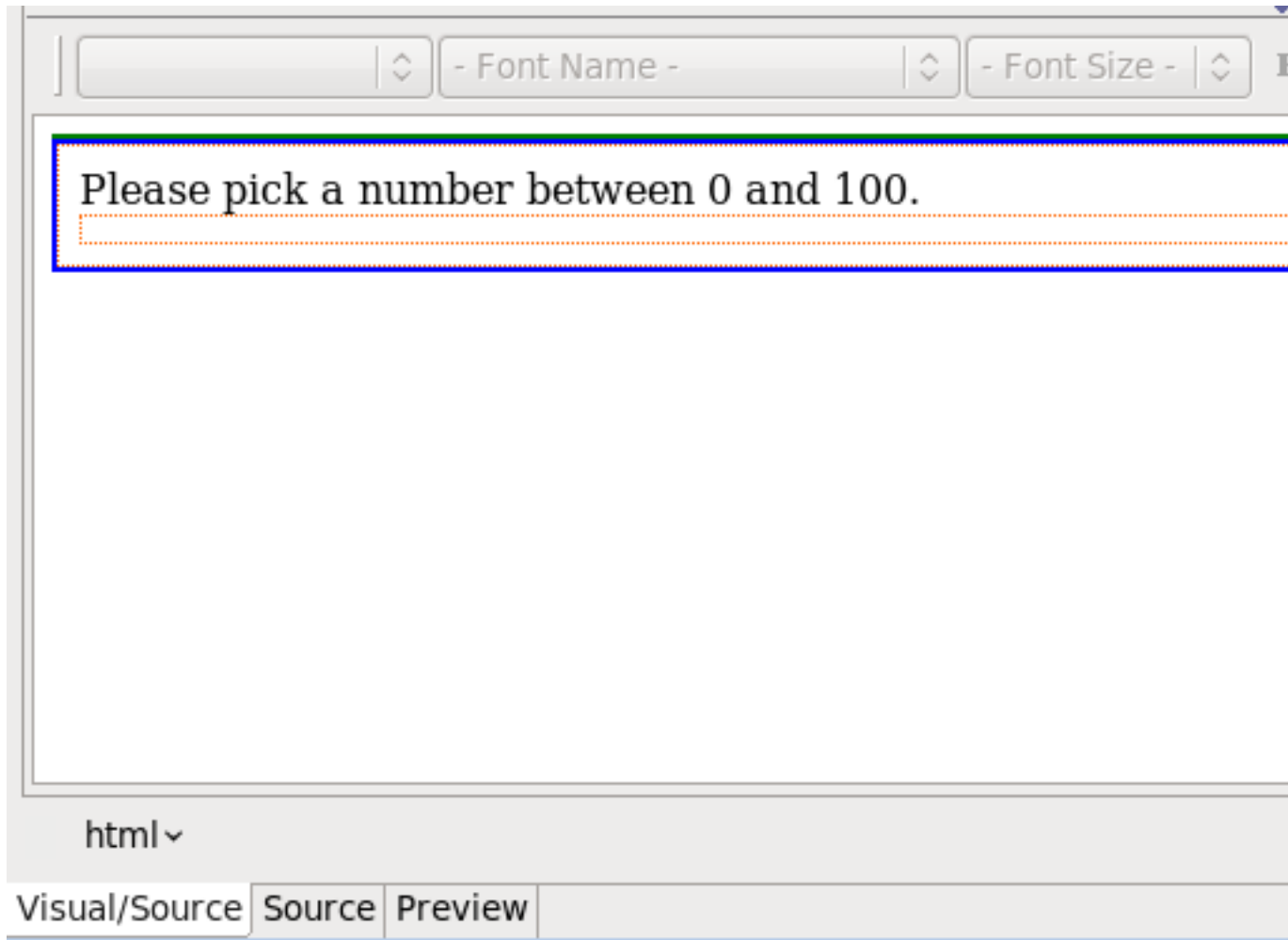


Figure 5.19. Created OutputText Component

- Switch to Source mode and insert a `
` tag after the `<h:outputText>` component to make a new line
- Click the **Save** button
- On the Palette click on *inputText*, drag the cursor over to the editor, and drop it inside the editor after the text
- Select the *value* row and click in the second column
- Click the ... button next to the value field
- Expand **Managed Beans** → **NumberBean**
- Select *userNumber* value and click the **OK** button
- Select the *Advanced* tab

- Select the *id* row and click in the second column
- Type *userNumber* in the text field
- Select the *required* row and click in the second column
- Click ... button next to the value field
- Expand *Enumeration* and select *true* as a value

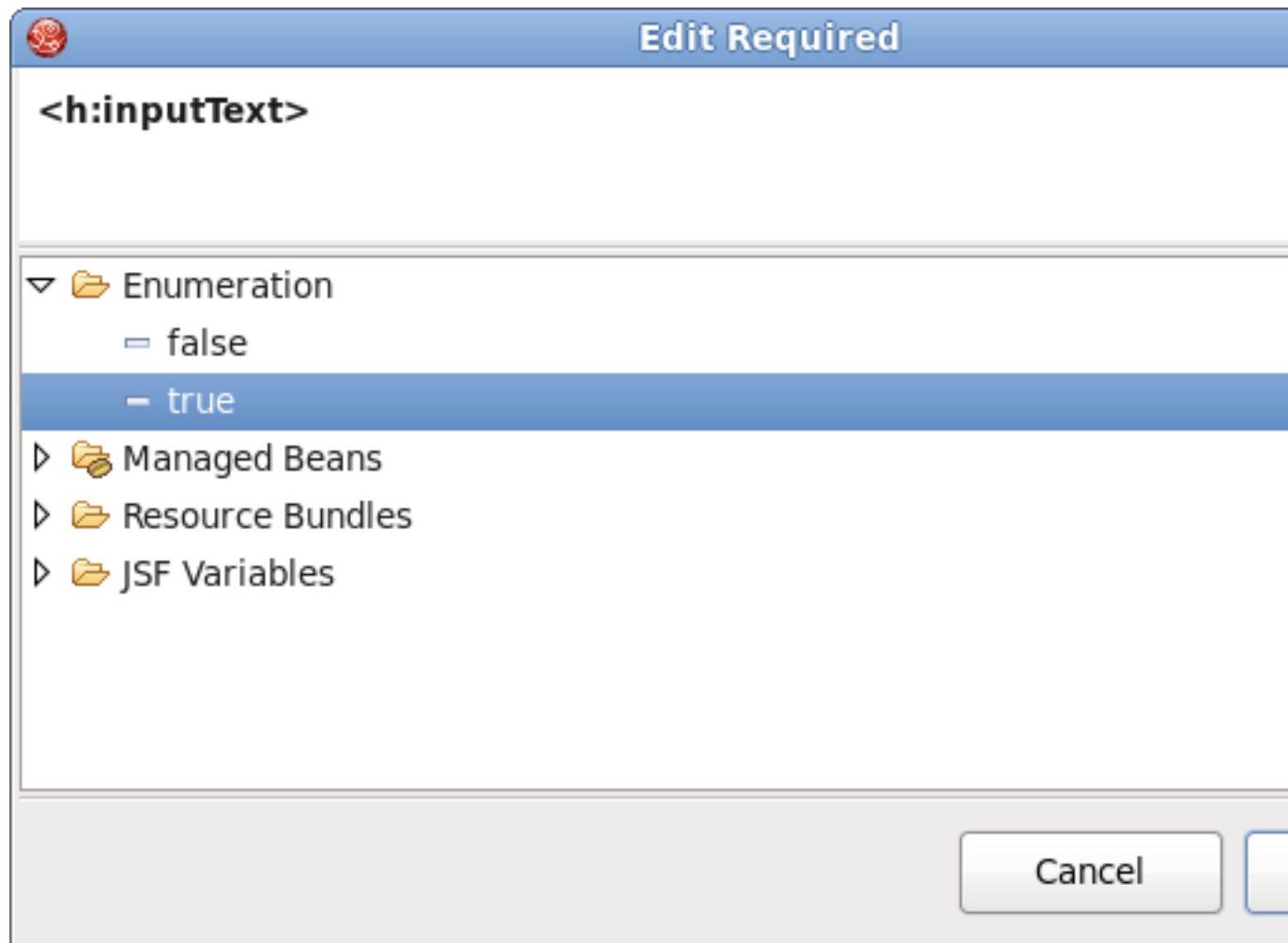


Figure 5.20. Add "required" Attribute

- Click the **OK** button, then click the **Finish** button
- Go to Source mode
- Add the validation attribute to <f:validateLongRange> for user input validation

```
<h:inputText id="userNumber" value="#{NumberBean.userNumber}" required="true">
```



```
<f:validateLongRange minimum="0" maximum="100"/>
</h:inputText>
```

- Click the **Save** button
- Again select *Visual* mode
- On the Palette, click on *commandButton*, drag the cursor over to the editor, and drop it inside the editor after the *inputText* component.
- In the editing dialog select the *value* row and click on the second column
- Click the ... button next to the value field
- Expand **Resource Bundles** → **msg** and select *makeguess_button* as a value
- Click the **OK** button
- Select the *action* row and click in the second column
- Type `#{NumberBean.checkGuess}` in the text field
- Click the **Finish** button
- In **Source** mode add `
` tags between the `<outputText>`, `<inputText>` and `<commandButton>` components to place them on different lines

inputnumber.jsp page should look like this:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="f" uri="http://java.sun.com/jsf/core"%>
<%@ taglib prefix="h" uri="http://java.sun.com/jsf/html"%>
<f:loadBundle basename="game.messages" var="msg"/>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<f:view>
<h:form id="inputNumbers">
<h:outputText value="#{msg.how_to_play}"/>
<br/>
<h:messages style="color: blue" />
```

```
<br/>
                <h:inputText          id="userNumber"          required="true"
value="#{NumberBean.userNumber}">
    <f:validateLongRange minimum="0" maximum="100" />
</h:inputText>
<br/>
<br/>
                <h:commandButton      action="#{NumberBean.checkGuess}"
value="#{msg.makeguess_button}" />
    </h:form>
</f:view>
</body>
</html>
```

5.7.2. Editing success.jsp page

We now edit the `success.jsp` page in the same way as we just edited the `inputnumber.jsp` file. The code for the `success.jsp` page should look like the following:

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<f:loadBundle basename="game.messages" var="msg" />

<html>
<head>
    <title></title>
</head>
<body>
    <f:view>
        <h:form id="result">
            <h:outputFormat value="#{msg.success_text}">
                <f:param value="#{NumberBean.userNumber}" />
            </h:outputFormat>
            <br />
            <br />
            <h:commandButton value="#{msg.trayagain_button}"
                action="#{NumberBean.playagain}" />
        </h:form>
    </f:view>
</body>
</html>
```

Again you can use code assist provided by JBoss Developer Studio when editing jsp page:

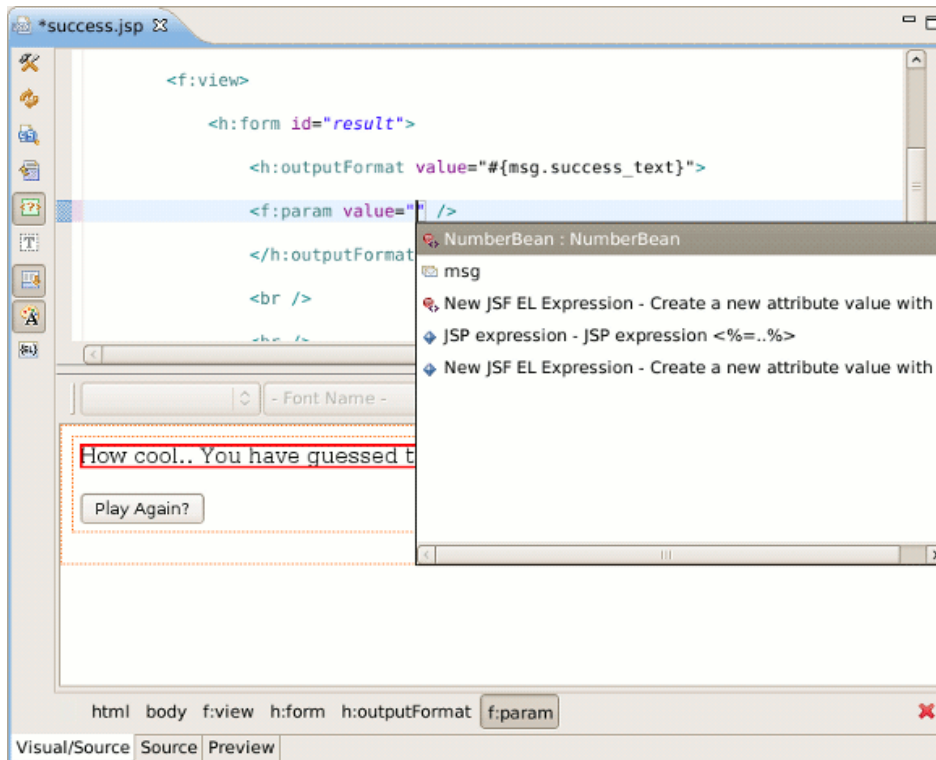


Figure 5.21. Code Assist for `<f:param>`

The `success.jsp` page is shown if you correctly guessed the number. The `<h:outputFormat>` tag will get the value of `success_text` from the properties file. The `{0}` in `success_text` will be substituted for by the value of the `value` attribute within the `<f:param>` tag during runtime.

In the final result you have a button which allows you to replay the game. The `action` value references a backing bean method. In this case, the method only terminates the current session so that when you are shown the first page, the input text box is clear and a new random number is generated.

- Switch to Preview mode to see how this page will look in a browser:

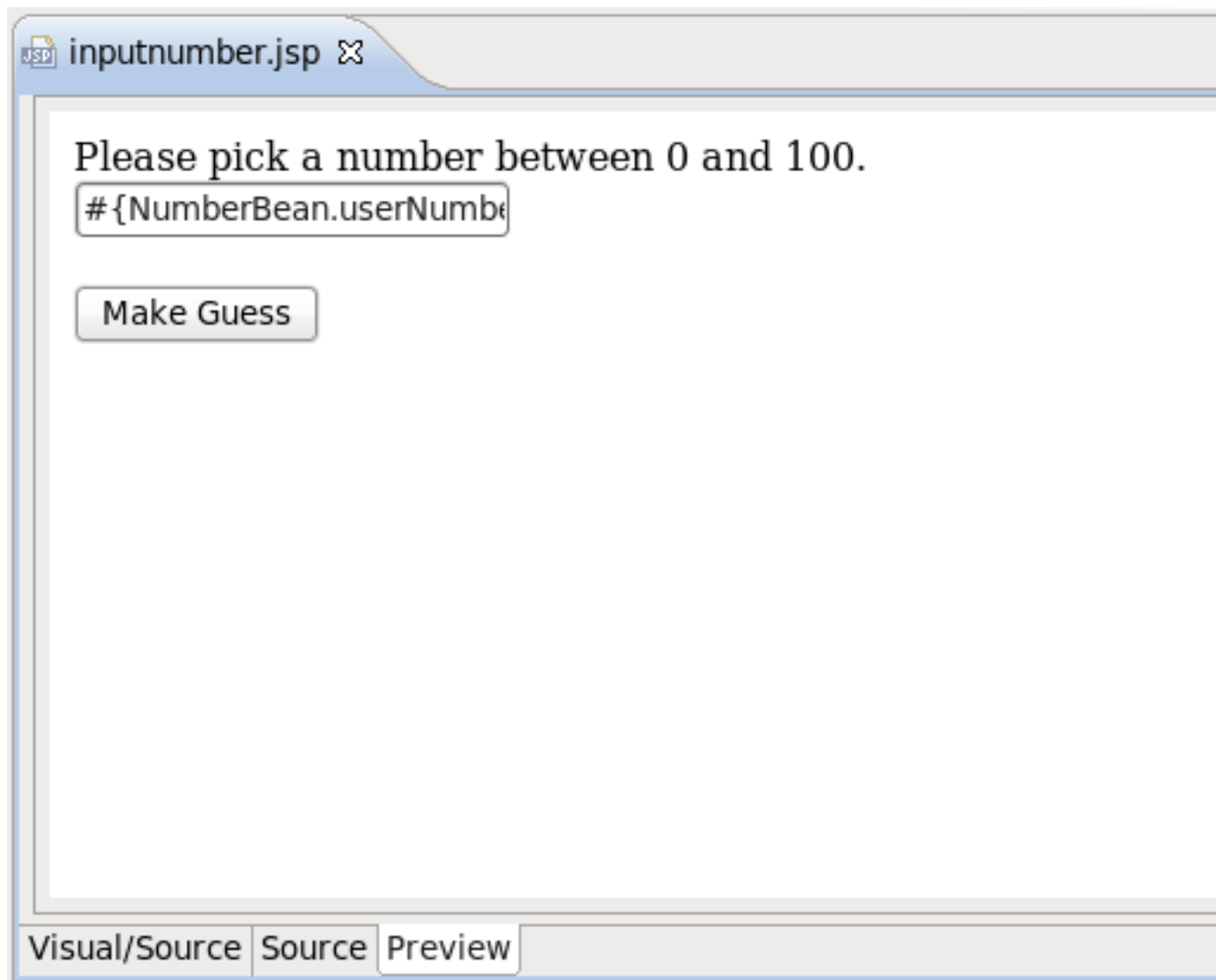


Figure 5.22. Success.jsp in Preview Mode

5.8. Creating index.jsp page

Now we need to create the `index.jsp` page.

The `index.jsp` page is the entry point of our application. It's just forwarding to the `inputnumber.jsp` page.

- Right click the `WebContent` folder and select **New** → **JSP File**
- Enter `index` for name field and click the **Next** button.
- Untick the **Use JSP Template** check box and click the **Finish** button.
- Edit the source of the file so it looks like the following:

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
  <body>
    <jsp:forward page="/pages/inputnumber.jsf" />
  </body>
</html>
```

Note the *.jsf* extension of a page. It means that we trigger the JSF controller servlet to handle the page according the servlet mapping in the `faces-config.xml` file.

5.9. Running the Application

Finally, we have all the pieces needed to run the application.

- Start up JBoss server by clicking on the **Start** icon in the Servers view. (If the JBoss Server is already running, stop it by clicking on the red icon and then start it again. After the messages in the Console tabbed view stop scrolling, JBoss is available)
- Right-click on the project and select **Run As** → **Run on Server**
- Play with the application by entering correct as well as incorrect values

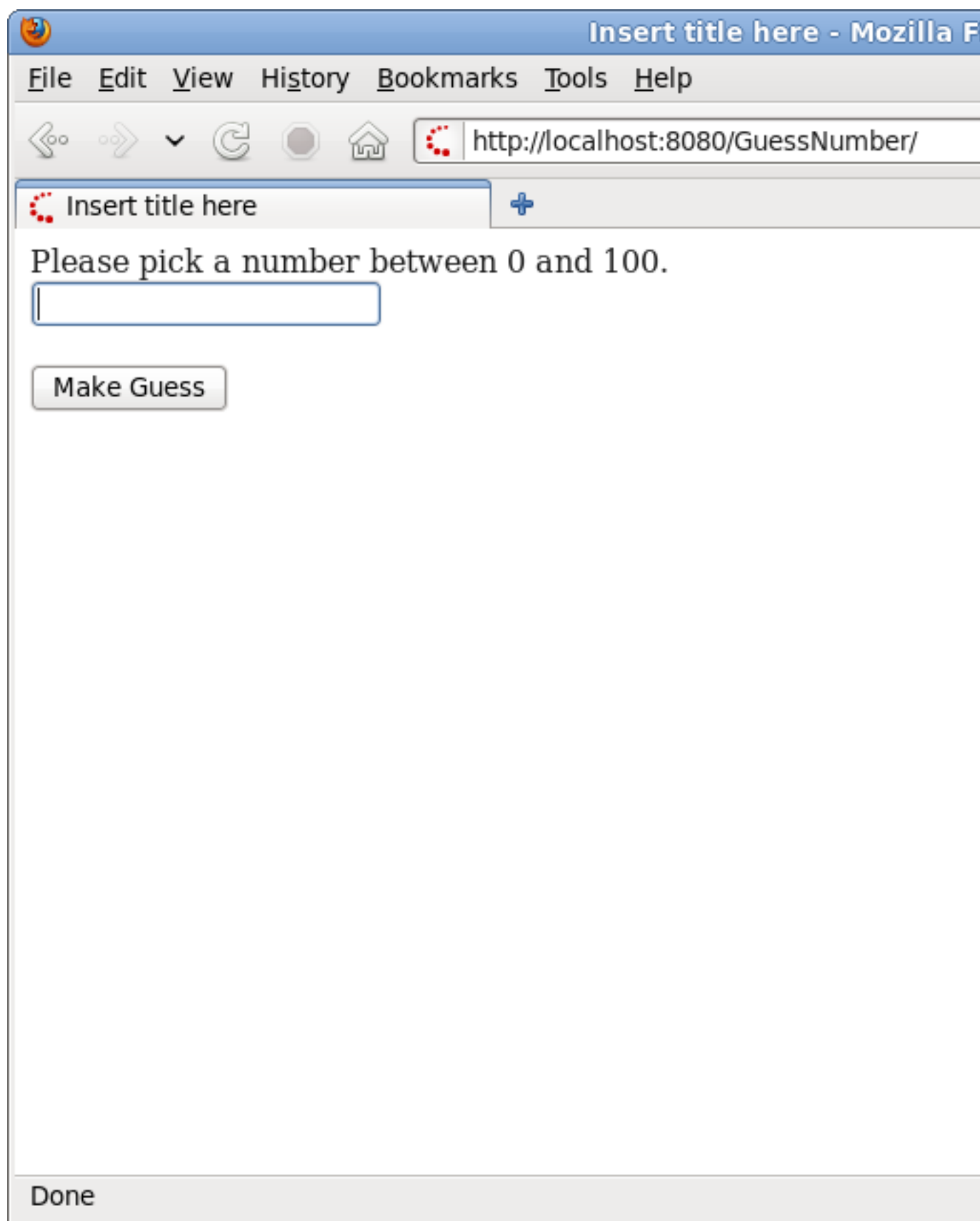


Figure 5.23. You are Asked to Enter a Number Between 0 and 100

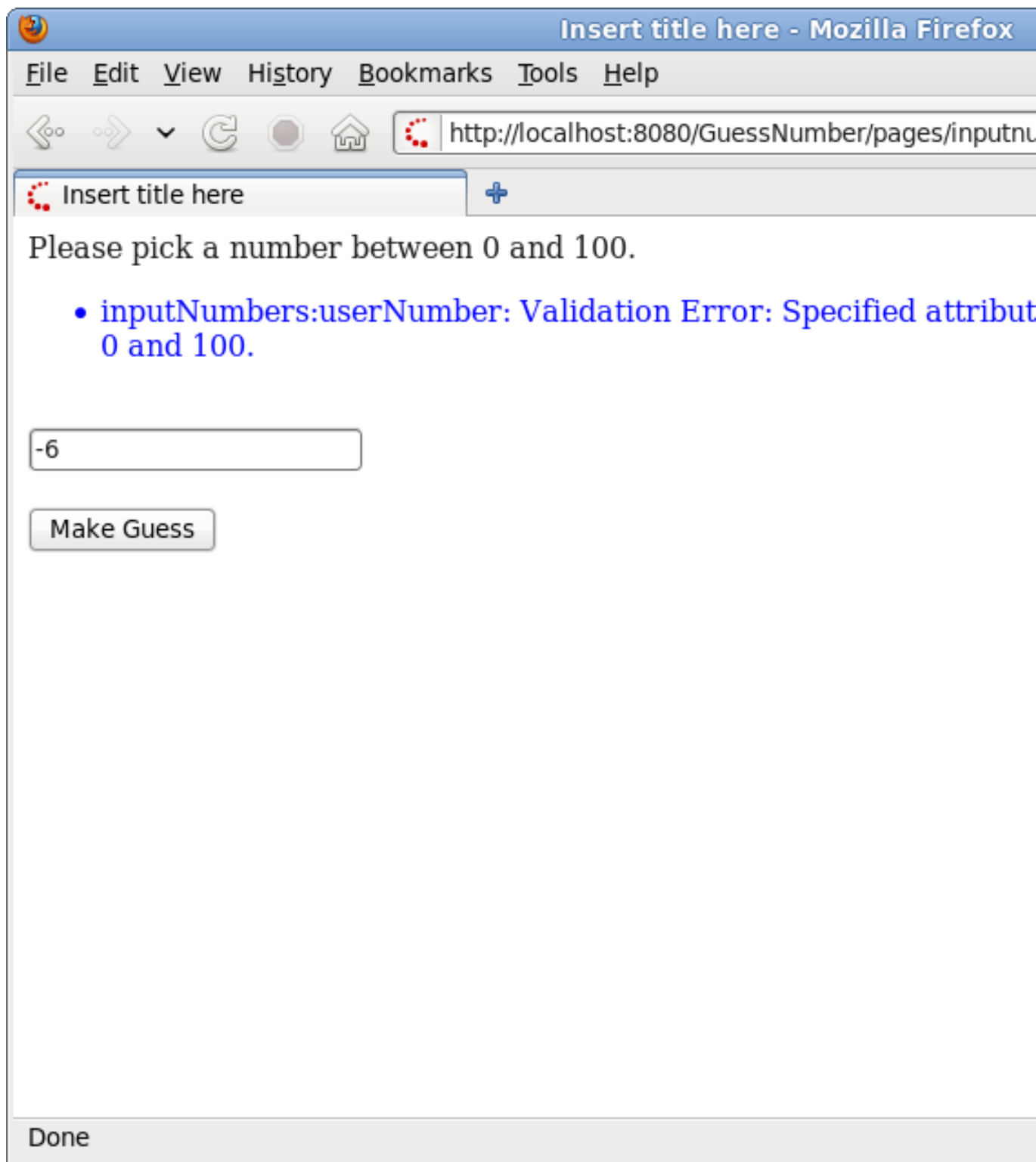


Figure 5.24. Your Input is Validated and an Error Message is Displayed if Invalid Input was Entered

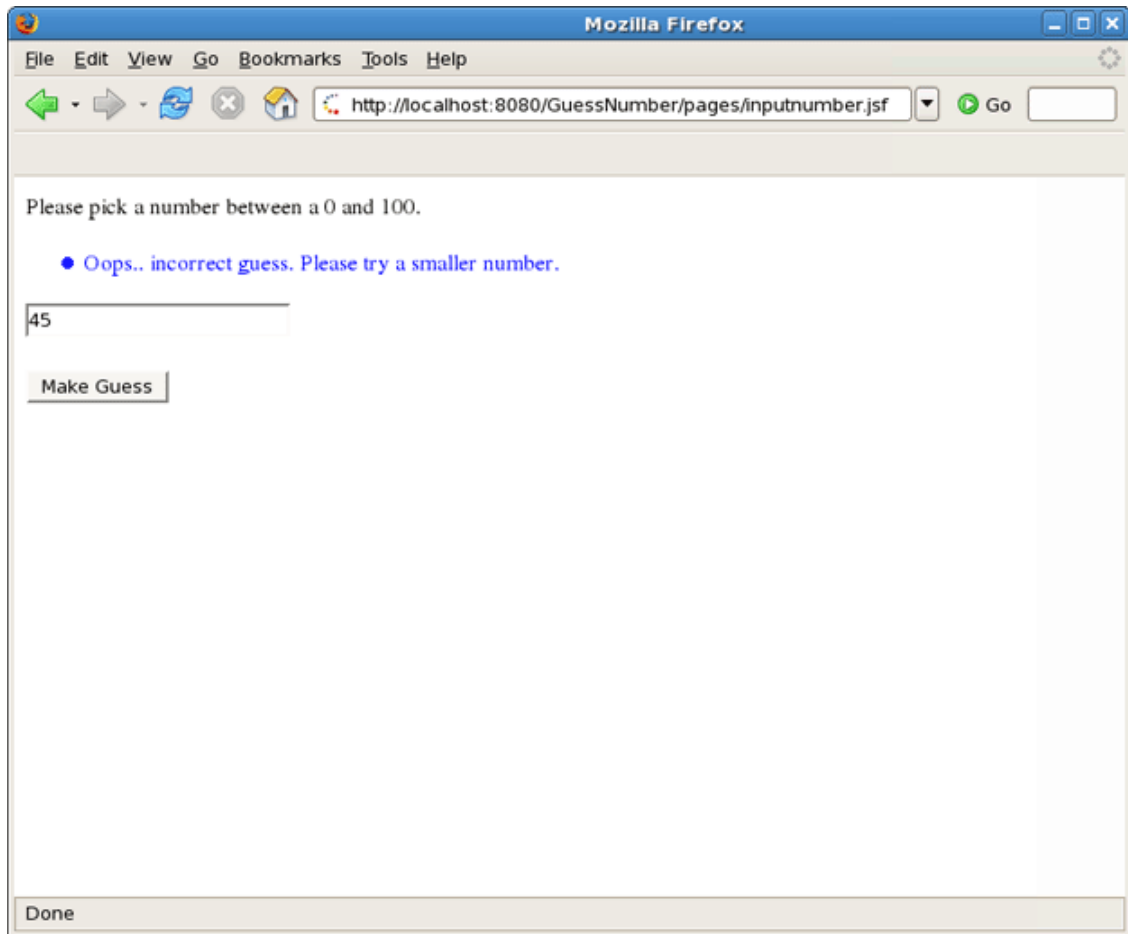


Figure 5.25. After You Enter a Guess, the Application Tells You Whether a Smaller or a Larger Number Should be Tried

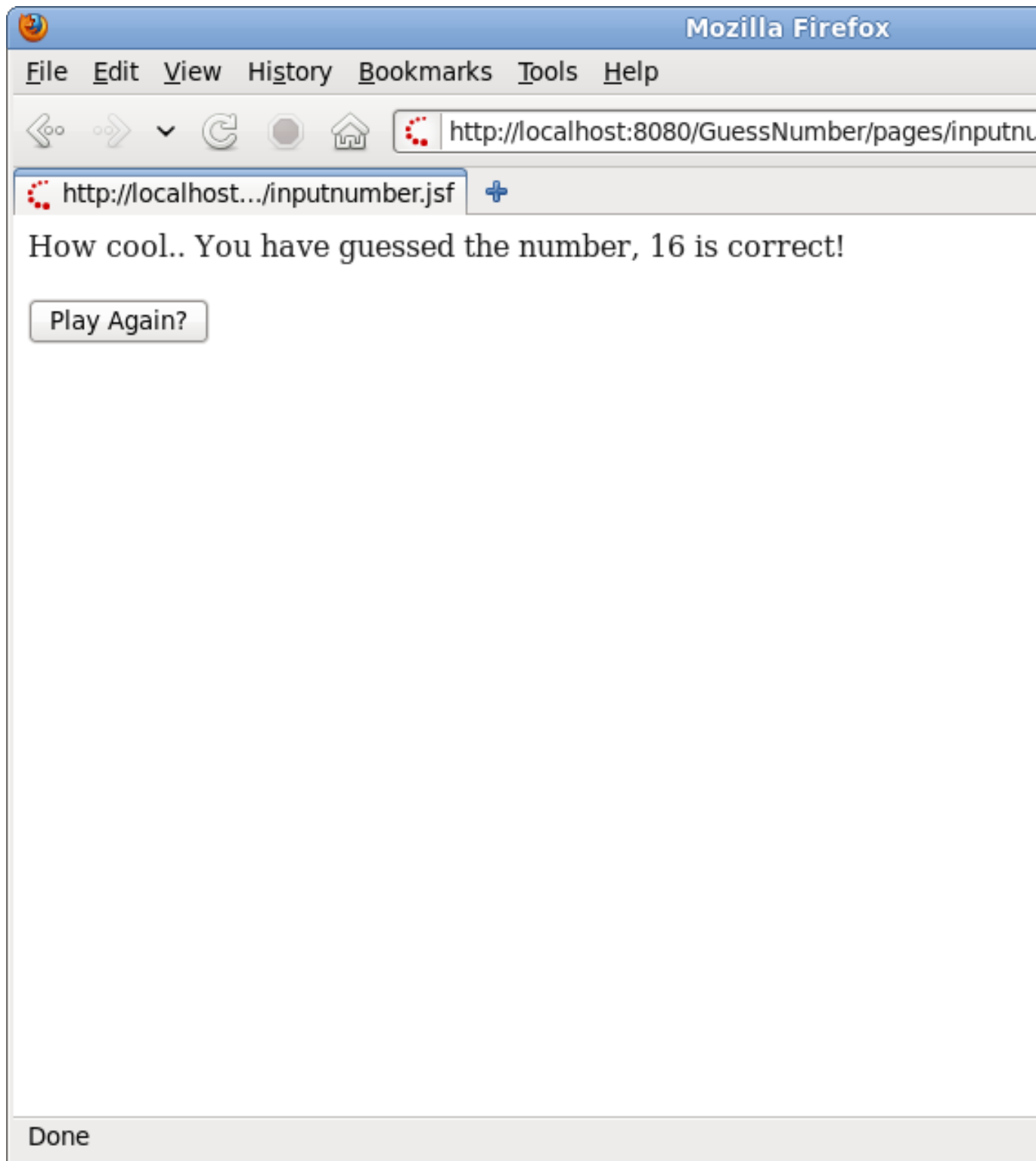


Figure 5.26. Your Guess is Correct

Project Examples

JBoss Developer Studio provides an option to download and import a ready-made project that you can explore learn from.

To adjust the settings of the Project Examples feature you need to navigate to **Windows** → **Preferences** → **JBoss Tools** → **Project Examples**.

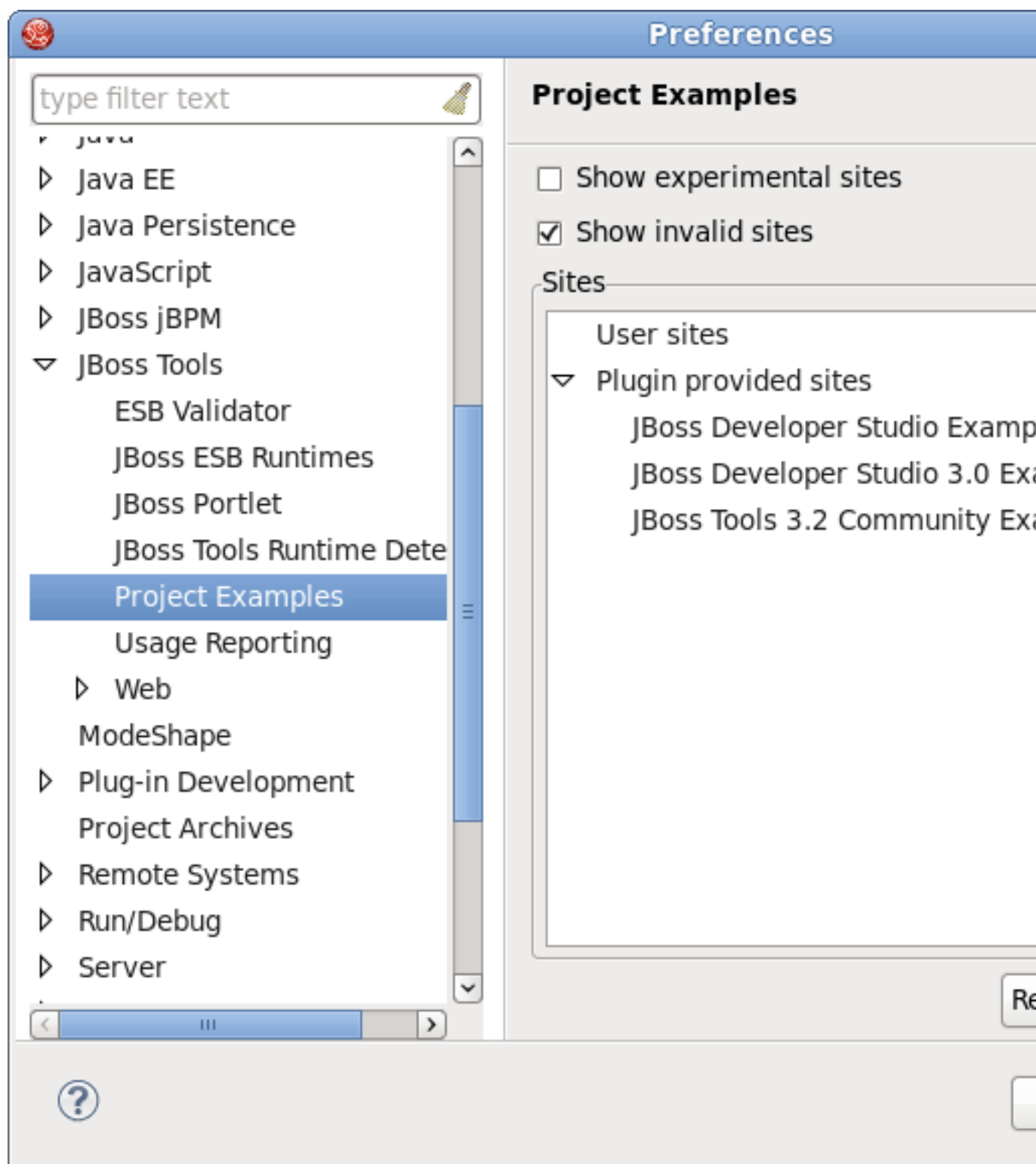


Figure 6.1. Project Examples Preferences

The Show experimental sites checkbox serves to enable or disable user sites in the Project Example dialog (**Help** → **Project Examples**).

6.1. User Sites

As you can see from the **Project Examples Preferences** image you can add a custom project example that can be provided by anyone. This feature can, for example, facilitate project testing.

In order to add a new project example you need to select the User sites option and press the **Add** button to the right.

When the **Add** button is pressed the Add Project Example Site dialog is displayed. The dialog contains two input fields: Name, where you need to specify the name of the new entry and URL, which has to point to the XML file that contains example project(s) properties. The structure of the XML file is discussed in more detail in a later chapter of this guide. Alternatively, if the XML is stored on your local machine, you can hit the **Browse** button to select the file in the file system.

Here is an example of the XML file that holds project example settings:

```
<projects>
  <project>
    <category>User Examples</category>
    <name>User Project Example</name>
    <shortDescription>
      Short project description.
    </shortDescription>
    <description>
      Full project description.
    </description>
    <size>10900</size>
    <url>
      http://projectexample.org/projectexample.zip
    </url>
  </project>
</projects>
```

Once you define the location of the XML file with projects settings you will see a new user site entry added. Please note now if you select the entry you can edit and remove it with the corresponding buttons to the right. You can not perform such operations with the Plugin provides sites.

When the user sites location is set up you can download and install the project(s). Please see the next chapter of the guide for more details.

6.2. Downloading a Project Example

To download a project example and start working with it you need to perform a few steps:

- Go to the menu bar and select **File** → **New** → **Other...**

- Select **Jboss Tools** → **Project Examples** (You can also select **Project Examples** from menu bar: **Help** → **Project Examples...** or directly by selecting **File** → **New** → **Example...** menu)

New Project Example

Project Example

Import Project Example

☐ Show experimental sites

Site:

Projects:

▶ Portlet

▶ ESB for SOA-P 4.3

▶ ESB for SOA-P 5.0

▶ Teiid Designer

Description:

Project name:

Project size:

URL:


☒ Show the Quick Fix dialog

Figure 6.2. Project Examples

Alternatively, you should navigate to **New** → **Other...**, scroll down to find the **JBoss Tools** option (or just type in the first letters of the word "JBoss" for quick search), expand the option and select **Project Examples**, and click the **Next** button.

- Now in the **New Project Example** dialog you can select a project you would like to explore and a site to download it from

The **Project Examples Wizard** provides a filter field to more easily locate the project examples you want, so you can type in the project you would like to explore in the field.



New Project Example

Project Example

Import Project Example

☐ Show experimental sites

Site:

Projects:

Seam Booking Example - EAR

Seam Booking Example - EAR (including a test project)

Seam Booking Example - WAR Standalone

Seam Booking Example - WAR Standalone (including a test project)


Description:

This example demonstrates the use of Seam in a Java EE 5 environment. Transaction and persistence context management is handled by the EJB container. This example requires JBoss EAP 4.3/JBoss AS 4.2.x and Seam 2.0.

Project name:

Project size:

URL:



This example has some requirements that could not be automatically configured. When importing the example you might see some errors which would need fixing manually or via Quick

☒ Show the Quick Fix dialog

Figure 6.3. Selecting a Project Example

If you have previously specified user sites (see the [User Sites](#) chapter) they also will be displayed in the list of project examples in the category that was defined in the XML file with user sites settings.

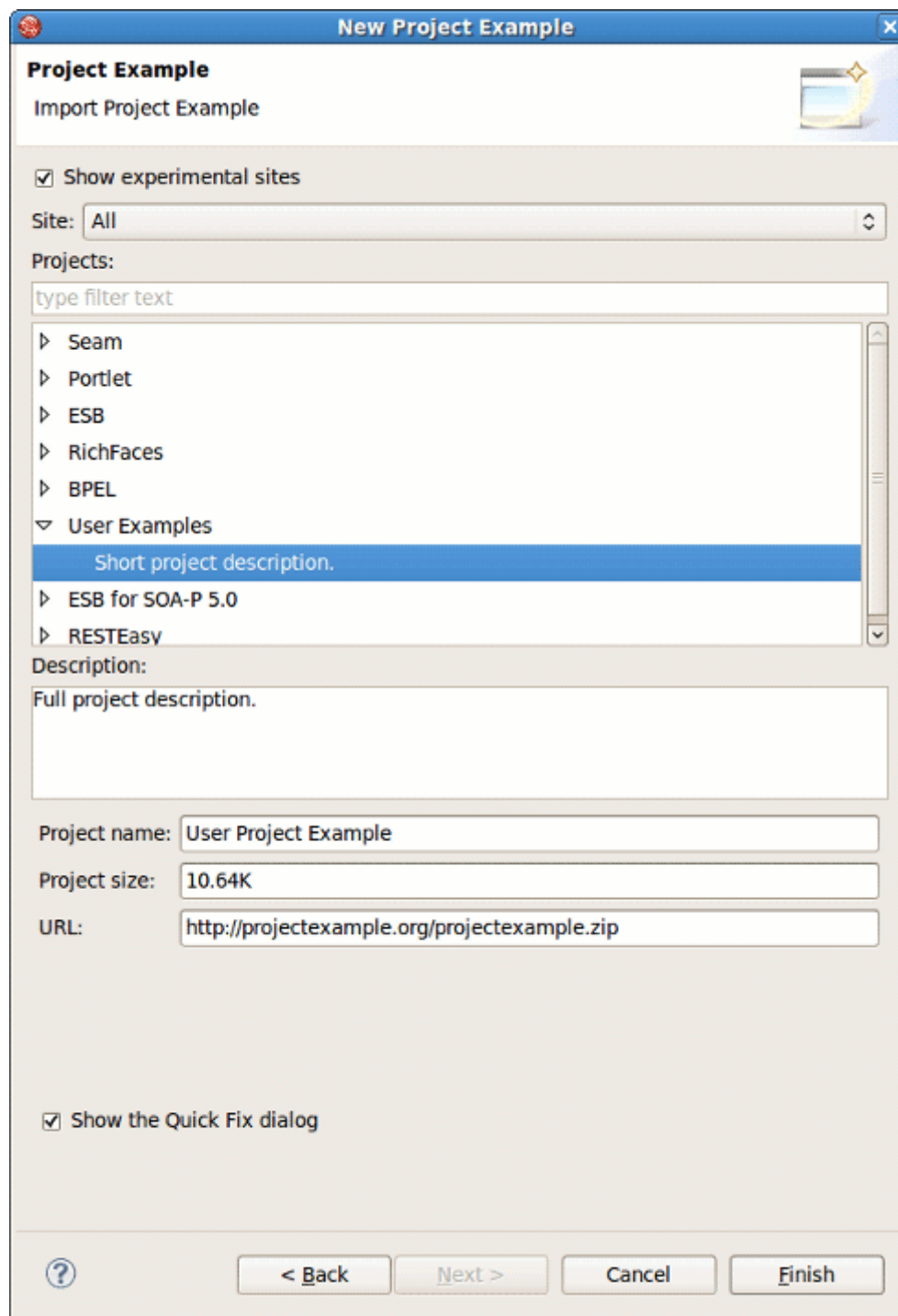


Figure 6.4. User Site

Please note that to view the user sites you need to have Show experimental sites checked.

**Note:**

The **Show the Quick Fix Dialog** option is described in the [Quick Fixes](#) section.

- Press the **Finish** button to start downloading the project from the repository

When downloading is finished the project will be imported automatically and you will be able to see it in the **Package Explorer** view.

Now you can run the application on the server.

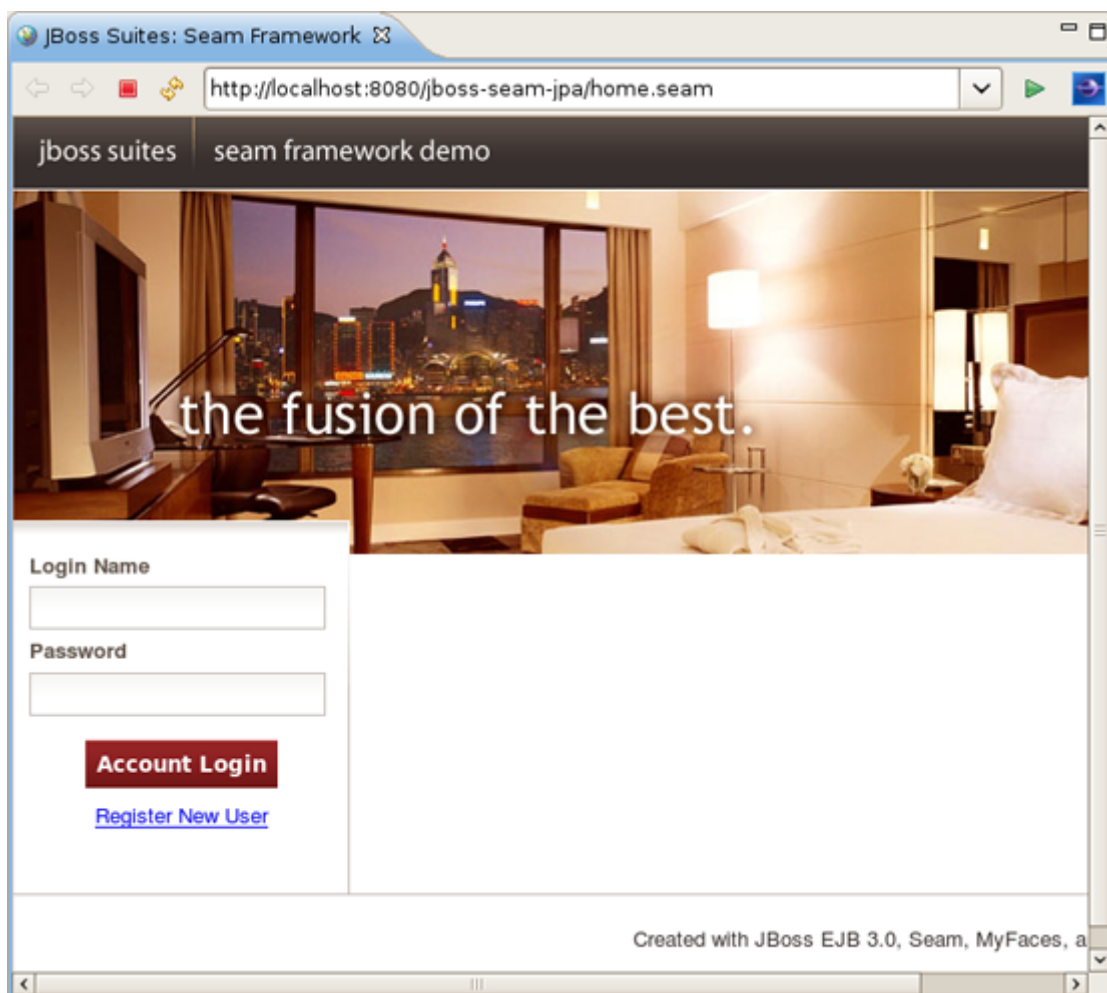


Figure 6.5. Seam Demo Application run on the Server

For further operation add the following code to .project files of your Web project example.

```
<buildCommand>
```

```
<name>org.jboss.tools.jst.web.kb.kbbuilder</name>
<arguments>
</arguments>
</buildCommand>
...
<nature>org.jboss.tools.jst.web.kb.kbnature</nature>
```

It is needed for Code Assist and JSF EL Validation to function correctly.

6.3. Quick Fixes

The **Project Examples Wizard** has an option for making quick fixes for the imported project to easily fix possible issues like missing servers, Seam runtimes etc.

To enable the quick fix option you need to check the **Show the Quick Fix dialog** while choosing the [Project Example](#).

When the project you selected is downloaded it will be checked for missing dependences and if any are detected you will see a dialog listing the problems.

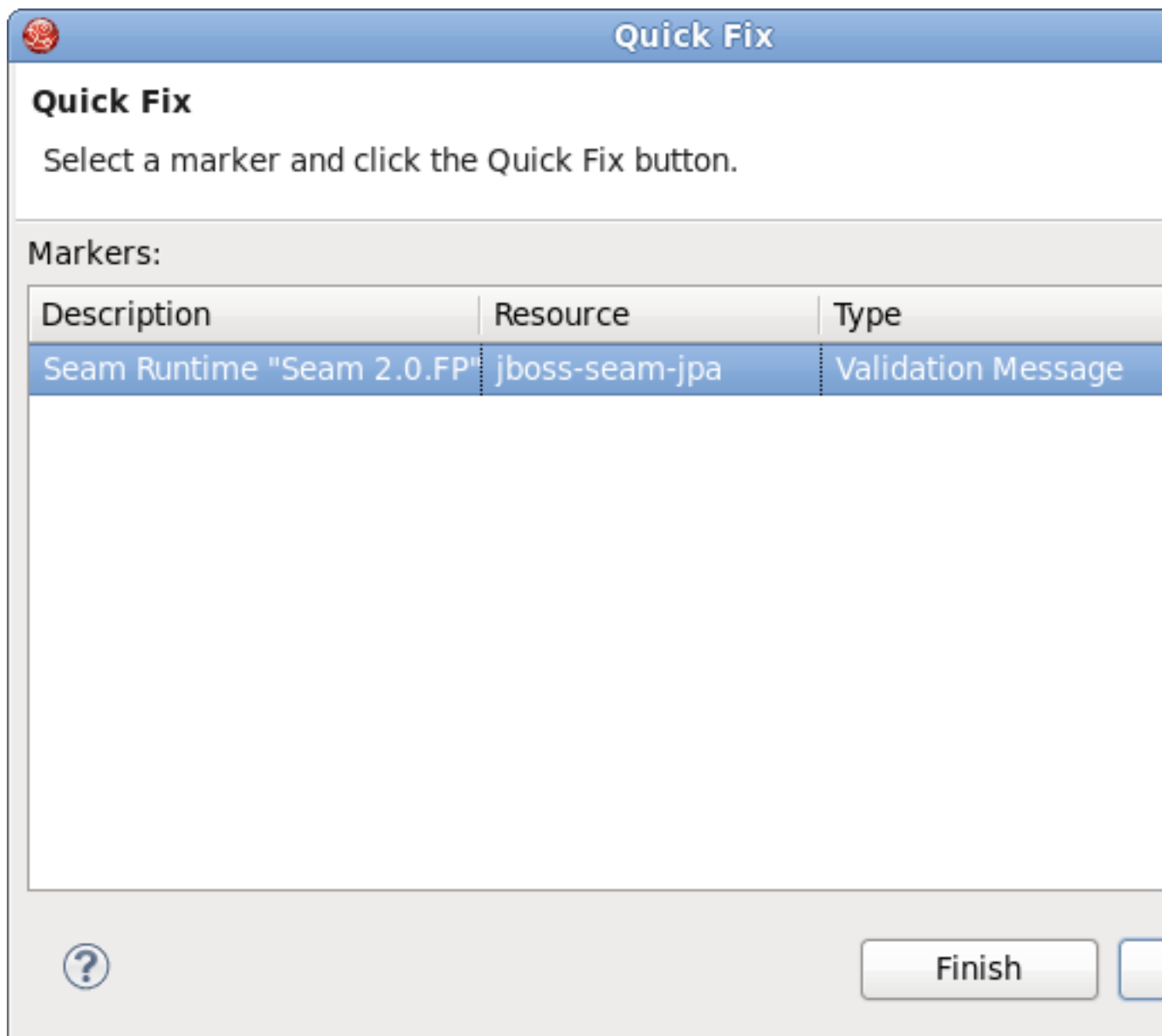


Figure 6.6. Quick Fix Dialog box

To fix the problem you need to:

- Select the problem from the list
- Click the **Quick Fix** button

You will be offered a solution or a number of solutions to the problem.

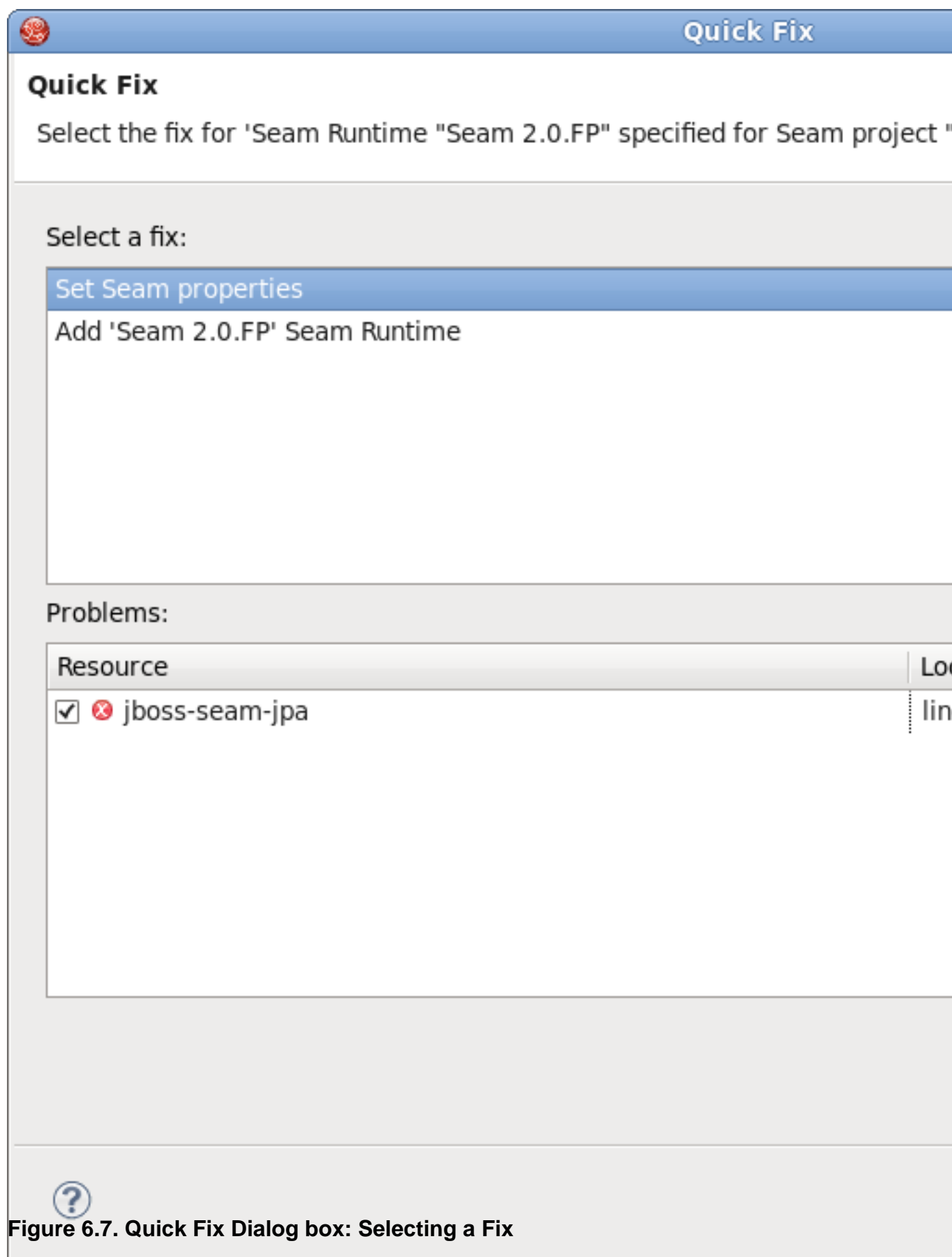


Figure 6.7. Quick Fix Dialog box: Selecting a Fix

In this case (see the image above), when the **Finish** button is pressed, the Seam Settings dialog box will be displayed where you need to provide a path to the Seam environment in order to fix the issue.

When the problem is fixed you will be returned to the Quick Fix dialog box with the remaining problems to be fixed.

You can also fix problems before downloading. When the project example is selected you will see warning message on the **New Project Example** dialog.

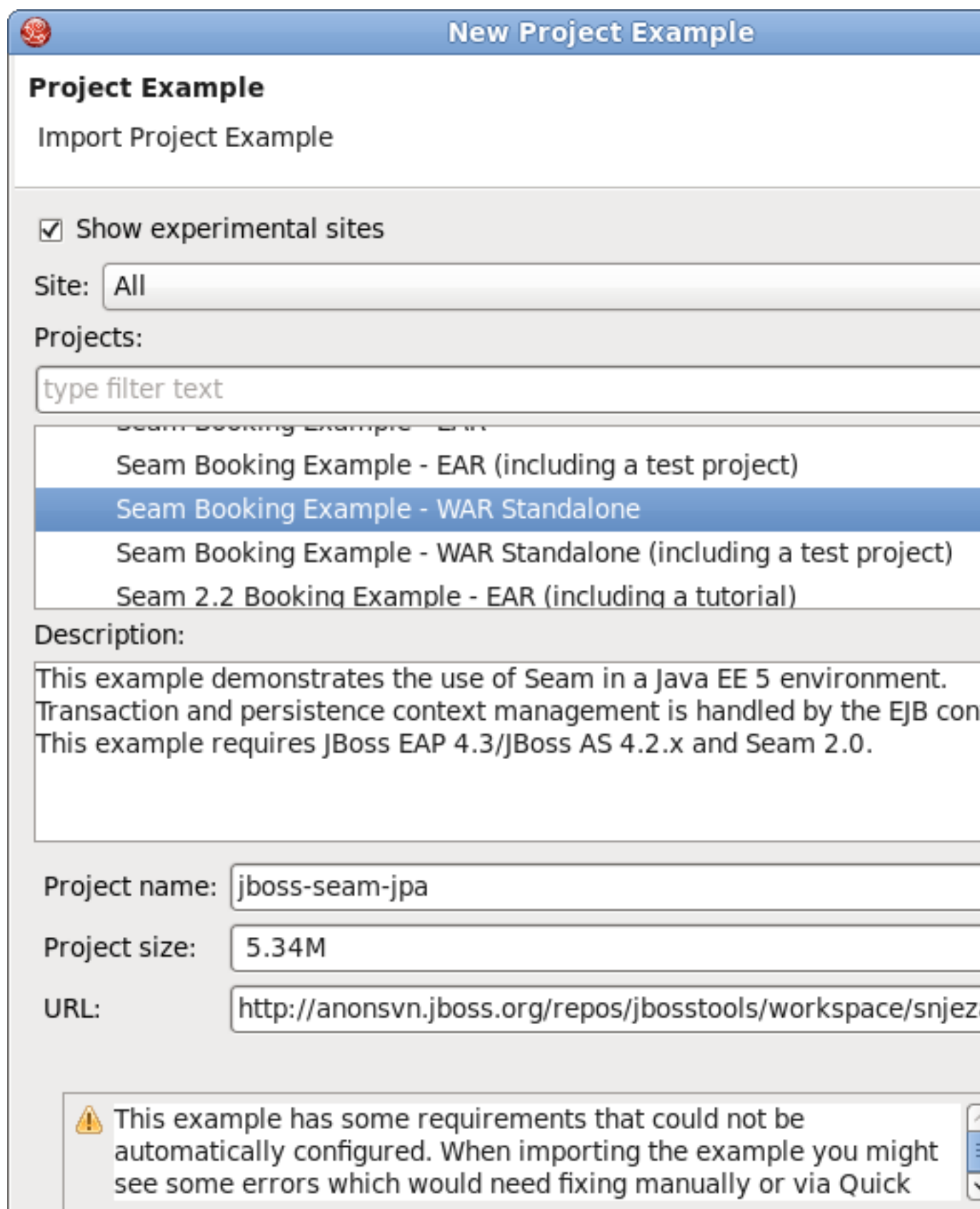


Figure 6.8. Requirements warning in the Project Example wizard

To fix the problem immediately you need to:

- Click the **Details...** button in the **New Project Example** dialog
- Select the problem from the list in the **Requirement details** dialog box
- Click the **Fix** button

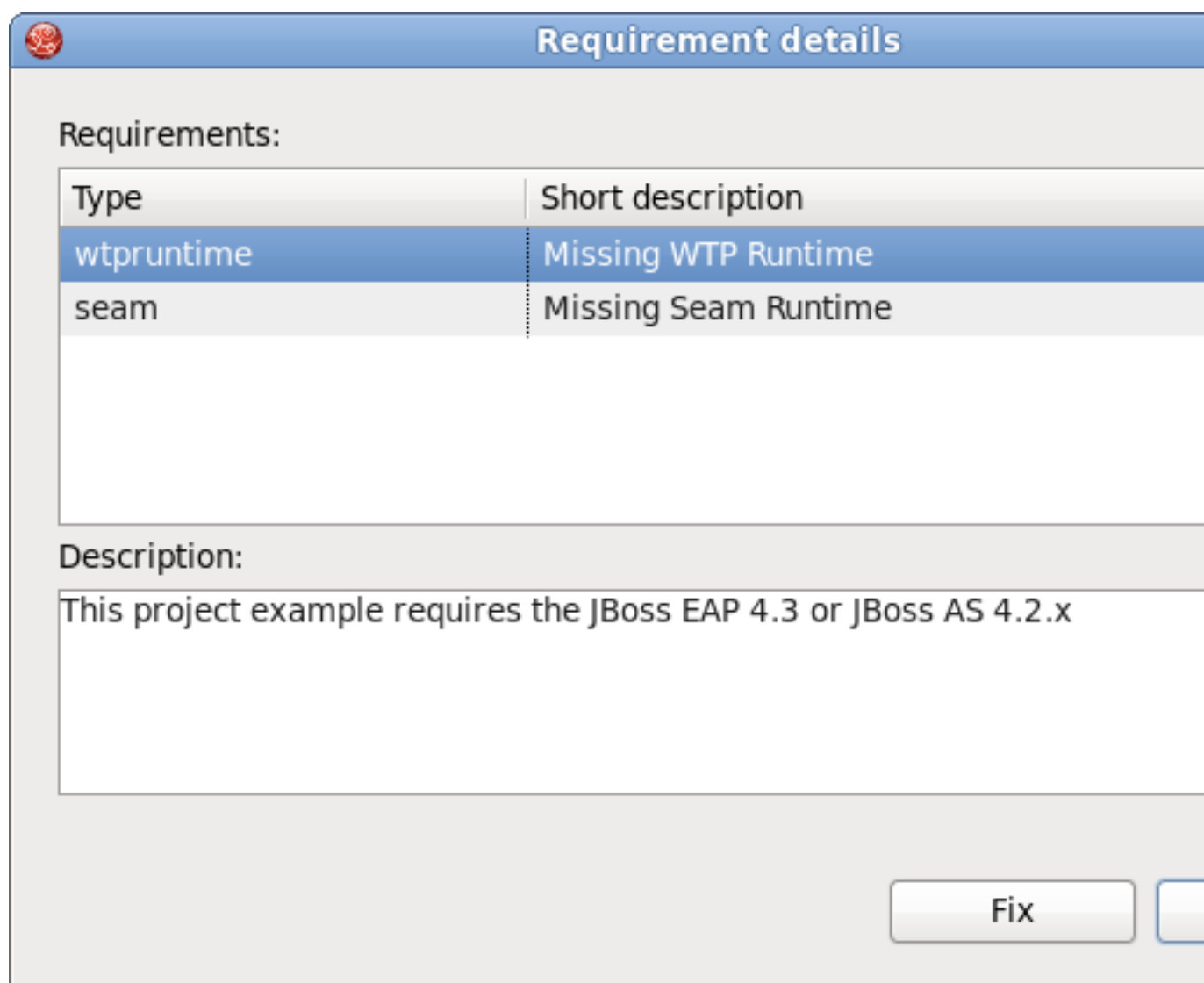


Figure 6.9. Requirement Details Dialog box: Selecting a Fix

You will be offered a solution to the problem.

FAQ

Refer to the following FAQ to get the answers on the most "popular" questions concerning JBoss Developer Studio.

7.1. What should I do if the Visual Page Editor does not start under Linux

Linux users may need to do the following to get the Visual Page Editor to work correctly on their machines.

1. On Red Hat based Linux distributions install the libXp.i386 package

2. Type

```
ln -s libstdc++.so.5.0.7 libstdc++.so.5
```

3. and/or use

```
yum install libXp
```

4. Open the JBoss Developer Studio perspective. If you see the Help view open, close it and restart JBoss Developer Studio

5. If it doesn't help and you use Fedora with Eclipse Version: 3.4.1, the issue can be produced because the libswt-xulrunner-gtk-3449.so file doesn't present in eclipse-swt-3.4.1-5.fc10.x86_64.rpm/eclipse/plugins/org.eclipse.swt.gtk.linux.x86_64_3.4.1.v3449c.jar. To add this file to eclipse you should:

- Decompress eclipse/plugins/org.eclipse.swt.gtk.linux.x86_3.4.1.v3449c.jar from eclipse-SDK-3.4.1-linux-gtk-x86_64.tar.gz
- Copy libswt-xulrunner-gtk-3449.so file to your Fedora Eclipse location.
- Open the file jbdevstudio.ini, which can be found in your Fedora Eclipse location and add the following line:

```
-Dswt.library.path=/usr/lib/eclipse
```

,where /usr/lib/eclipse is the path to your eclipse folder.

6. If none of these work, do the following:

- Clear the JBoss Developer Studio log file, <workspace>\.metadata\.log
- Start JBoss Developer Studio with the -debug option:

```
jbdevstudio -debug
```

- Post the JBoss Developer Studio log file(<workspace>\.metadata\.log) on the forums.

7.2. Visual Editor starts OK, but the Missing Natures dialog appears

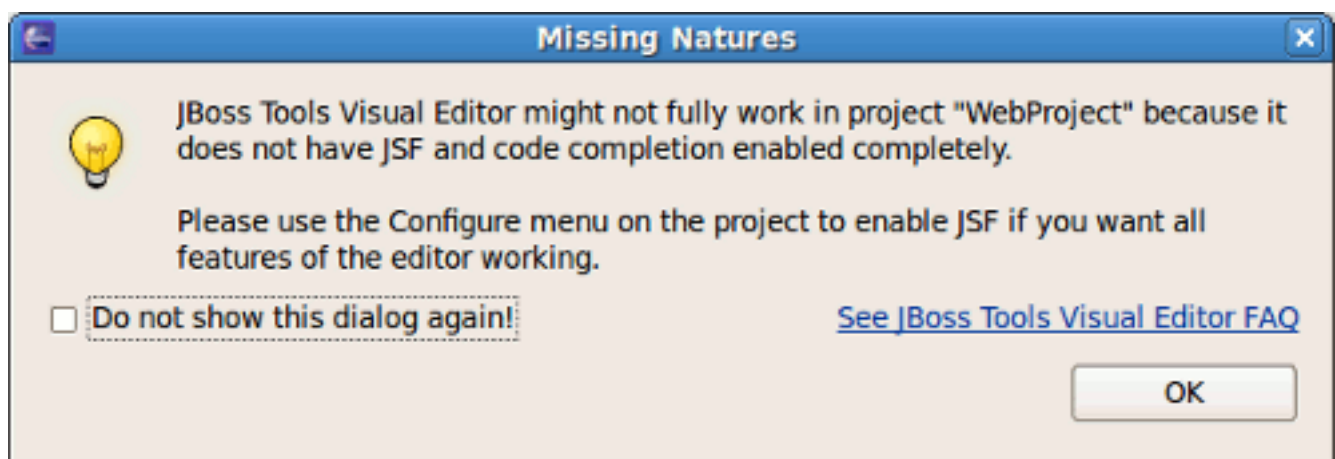


Figure 7.1. Missing Nature

Some functionality of Visual Editor may not work if a project doesn't have `org.jboss.tools.jsf.jsfnature` or `org.jboss.tools.jst.web.kb.kbnature` in .project configuration. To fix this problem and turn off the message box execute next steps:

1. Right mouse button click on a project in Package Explorer.
2. Select **Configure** → **Add JSF Capabilities** from the context menu.
3. Configure your project using Add JSF Capabilities wizard and press Finish.

If you are sure that your project does not need JSF capabilities, just disable this message box by checking `Do not show this dialog again!` checkbox.

7.3. Do I need to have JBoss Server installed to run JBoss Developer Studio?

No. JBoss Developer Studio already comes bundled with JBoss Server. We bundle it together so that you don't need to download any additional software and can test your application in a Web browser right away.

If you want to use a different JBoss server installation, after JBoss Developer Studio is installed open Servers View (select **Window** → **Show View** → **Others** → **Server** → **Servers**), then right click and select **View** → **New** → **Server** and follow the wizards steps to point to another Jboss Server installation.

JBoss Developer Studio works with any servlet container, not just JBoss. For more information on deployment, please see the Deploying Your Application section.

7.4. I have an existing Seam 1.2.1 project. Can I migrate or import the project into a JBoss Developer Studio Seam project?

Use the following steps to manually transfer an existing Seam 1.2.1 project into a new JBoss Developer Studio Seam project:

- Create a Seam Web project to get the JBoss tools structure

Then from your Seam 1.2.1 seam-gen project start doing the following:

- Copy `src` to `src`
- Copy `view` to `Web content`
- Copy resources individual files to where they are in the seam web project etc.

7.5. I have an existing Struts or JSF project. Can I open the project in JBoss Developer Studio?

Yes. From main menu select **File** → **File** → **Import** → **Other** → **JSF Project (or Struts Project)** and follow wizards steps.

7.6. Can I import a WAR file?

Yes. Select **File** → **Import** → **Web** → **WAR file** then follow importing steps.

7.7. Is it possible to increase the performance of Eclipse after installing your product?

JBoss Developer Studio configures eclipse via the `jbdevstudio.ini` file to allocate extra memory, but if you for some reason need more memory then by default, you can manually make adjustments in this file. For example:

```
-vmargs -Xms128m -Xmx512m -XX:MaxPermSize=128m
```

7.8. How can I add my own tag library to the JBoss Tools Palette?

See the section on Adding Tag Libraries in the Visual Web Tools Guide.

7.9. How to get Code Assist for Seam specific resources in an externally generated project?

To get Code Assist for Seam specific resources in an externally generated project, you should enable Seam features in Project Preferences. Right click an imported project and navigate **Properties** → **Seam Settings**. Check *Seam support* box to enable all available Seam Settings.

7.10. How to import an example Seam project from jboss-eap directory?

To import an example Seam project from *jboss-eap* into your working directory, you should perform the following steps:

- Select **New** → **Other** → **Java Project from Existing Buildfile**
- Point to the `build.xml` file of any chosen project by clicking the **Browse** button
- Click the **Finish** button to open the project

As these seam examples are non WTP projects, next you should enable Seam support for them. To do that, right click the project and select **Properties** → **Seam Settings**.

7.11. Is a cross-platform project import possible for JBoss Developer Studio?

Yes. You can easily import created in Linux JSF, Struts or Seam project to Windows and vice versa.

To do the transferring JSF, Struts or Seam project, select **Menu → Import → General → Existing Projects into Workspace**.

Further Reading

- **Seam Dev Tools Reference Guide**

This guide helps you to understand what Seam is and how to install Seam plug-in into Eclipse. It tells you the necessary steps to start working with Seam Framework and assists in a simple Seam Project creation. Also you will learn how to create and run the CRUD Database Application with Seam as well as find out what Seam Editors Features and Seam Components are.

- **Visual Web Tools Reference Guide**

provides general orientation and an overview of visual web tools functionality. This guide discusses the following topics: editors, palette, web properties view, openOn, content assist, RichFaces support.

- **JBoss Server Manager Reference Guide**

This guide covers the basics of working with the JBoss server manager. You will read how to install runtimes and servers and quickly learn how to configure, start, stop the server and know how deployment and archiving process. You will find out how to manage installed JBoss Servers via JBoss AS Perspective. You will also read how to deploy modules onto the server.

- **jBPM Tools Reference Guide**

With jBPM Tools Reference Guide we'll help you to facilitate a cross-product learning and know how you can speed your development using special editors and visual designers. We'll also guide you through the steps on how to create a simple process and test it within jBPM jPDL perspective.

- **Hibernate Tools Reference Guide**

Throughout this guide you will learn how to install and use Hibernate Tools bath via Ant and through Eclipse. We'll supply you with the information on how to create mapping files, configuration file as well as a file for controlling reverse engineering by using specific wizards that Hibernate tooling provides. Also you will know about Code Generation and peculiarities of work within Hibernate Console Perspective.

- **ESB Editor Reference Guide**

This guide provides you with the information on ESB Editor and all necessary wizards for ESB files development.

- **JBoss Portal Tools Reference Guide**

The guide gives a detail look at how you can easily build a Portlet Web Application with JBoss Tools and deploy it onto JBoss Portal.

- **JBoss WS User Guide**

This guide gives you practical help on JBossWS usage. You will learn how to create a web service using JBossWS runtime, find out how to create a web service client from a WSDL document using JBoss WS and also see how to set your development environment.

- **Smooks Tools Reference Guide**

This guide is packed with useful and easy-to-understand information about graphical, configuration and source editor pages.

- **Drools Tools Reference Guide**

The guide help you to discover how to create a new Drools project, use debugging rules and work with different editors.

- **JMX Tools Reference Guide**

With the help of this guide you'll explore the best practices to follow when working with MBean Explorer, MBean Editor, Connections and etc.

- **Eclipse Guvnor Tools Reference Guide**

The purpose of this guide is to describe briefly the functionality present in the Eclipse Guvnor Tools (EGT) for Drools 5.

- **JSF Tools Tutorial**

This tutorial will describe how to deal with classic/old style of JSF development and how to create a simple JSF application.

- **JSF Tools Reference Guide**

From this guide you'll discover all peculiarities of work at a JSF project. You'll learn all shades that cover the process of project creation and take a closer look at the JSF configuration file. Also you'll get to know managed beans and how to work with them and find out, how to create and register a custom converter, custom validator and referenced beans in a JSF project.

- **Struts Tools Reference Guide**

In Struts Tools Reference Guide you will learn how to create and work with a new struts project. This guide also provides information about graphical editor for struts configuration files, tiles files, and struts validation files.

- **Struts Tools Tutorial**

This tutorial will describe the classical style of Struts development, and will step-by-step show you how to create a simple Struts application.