

Getting Started Guide

Version: 3.2.0.Beta

1. Getting Started with JBoss Developer Studio	1
1.1. What is JBDS?	1
1.2. Key Features of JBoss Developer Studio and Requirements	1
1.3. Configuring Your Java Environment	3
1.3.1. Installing and Configuring 32-bit Sun JDK 5.0 on Linux	3
1.3.2. Installing and Configuring 32-bit Sun JDK 5.0 on Microsoft Windows	5
1.4. JBoss Developer Studio Installation	5
1.5. JBoss Developer Studio and JBoss Tools	11
1.5.1. What is the difference?	11
1.5.2. JBoss Tools Installation and Update	12
1.6. Welcome to JBoss Developer Studio	17
1.7. Upgrading	22
1.8. Uninstalling	22
1.9. Usage Reporting	22
1.9.1. Collected usage information guide	23
1.10. Support	24
1.11. Other relevant resources on the topic	24
2. Manage JBoss AS from JBoss Developer Studio	25
2.1. How to Manage the JBoss AS Bundled in JBDS	25
2.1.1. Starting JBoss Server	26
2.1.2. Stopping JBoss Server	27
2.1.3. Server Container Preferences	28
2.2. How to Use Your Own JBoss AS Instance with JBDS	29
2.2.1. JBoss AS Installation	29
2.2.2. Adding and Configuring JBoss Server	29
3. Write Your First Project with JBoss Developer Studio	34
3.1. Create a Seam Application	34
3.1.1. Start Development Database	34
3.1.2. Create and deploy Seam Web Project	34
3.1.3. Start JBoss Application Server	44
3.1.4. Workshop Project Code Overview	45
3.2. Seam Action Development	47
3.2.1. Create a New Seam Action	47
3.2.2. Test Seam Action	48
3.2.3. Modify Seam Action User Interface	51
3.3. Declarative Security	53
3.3.1. Edit Login Authentication Logic	53
3.3.2. Secure Seam Page Component	53
3.4. Browsing Workshop Database	55
3.4.1. Database Connectivity Setup	55
3.4.2. Browse Workshop Database	56
3.5. Database Programming	58
3.5.1. Reverse Engineer CRUD from a Running Database	58
3.5.2. Use Hibernate Tools to Query Data via JPA	61

3.5.3. Use Hibernate Tools to visualize the Data Model	67
3.6. Rich Components	68
3.6.1. Add a Richfaces component to the CRUD Application	69
4. Developing a simple JSP web application	73
4.1. Setting Up the Project	73
4.2. Creating JSP Page	75
4.2.1. Editing a JSP Page	76
4.2.2. web.xml file	78
4.2.3. Deploying the project	79
4.2.4. JSP Page Preview	83
4.2.5. Launch JSP Project	83
5. RAD development of a simple JSF application	85
5.1. Setting up the project	85
5.2. Creating JSP Pages	86
5.3. Creating Transition between two views	88
5.4. Creating Resource File	89
5.5. Creating Java Bean	93
5.6. Editing faces-config.xml File	98
5.7. Editing the JSP View Files	99
5.7.1. Editing inputnumber.jsp page	99
5.7.2. Editing success.jsp page	106
5.8. Creating index.jsp page	108
5.9. Running the Application	109
6. Project Examples	113
6.1. User Sites	113
6.2. Downloading a Project Example	114
6.3. Quick Fixes	121
7. FAQ	125
7.1. What should I do if Visual Page Editor does not start under Linux	125
7.2. Visual Editor starts OK, but the Missing Natures dialog appears	126
7.3. Do I need to have JBoss Server installed to run JBoss Developer Studio?	126
7.4. I have an existing Seam 1.2.1 project. Can I migrate/import the project to a JBDS Seam project?	127
7.5. I have an existing Struts or JSF project. Can I open the project in JBDS?	127
7.6. Can I import a .war file?	127
7.7. Is it possible to increase the performance of Eclipse after installing your product?..	127
7.8. How can I add my own tag library to the JBoss Tools Palette?	128
7.9. How to get Code Assist for Seam specific resources in an externally generated project?	128
7.10. How to import an example Seam project from jboss-eap directory?	128
7.11. Is a cross-platform project import possible for JBDS?	128
8. Further Reading	129

Getting Started with JBoss Developer Studio

1.1. What is JBDS?

[JBoss Developer Studio](#) is a set of eclipse-based development tools that are pre-configured for JBoss Enterprise Middleware Platforms and Red Hat Enterprise Linux. Developers are not required to use [JBoss Developer Studio](#) to develop on JBoss Enterprise Middleware and/or Red Hat Linux. But, many find these pre-configured tools offer significant time-savings and value, making them more productive and speeding time to deployment.

This guide covers the first steps to get started working with [JBoss Developer Studio](#). You will learn how to install and configure necessary software for your OS (currently Linux, Microsoft Windows or Mac OSX).

Thus this guide will provide you with detailed info on how to start JDK, JBDS and JBoss Tools.

1.2. Key Features of JBoss Developer Studio and Requirements

For a start, we propose you to look through the table of main features of JBoss Developer Studio:

Table 1.1. Key Functionality of JBoss Developer Studio

Feature	Benefit
JBoss Enterprise Application Platform	JBoss Developer Studio includes and integrates JBoss Enterprise Application Platform, the same version of the middleware platform that Red Hat supports in production for 5 years.
New Seam Tools	JBoss Developer Studio includes brand new tools for Seam. They provide wizards for creating new Seam projects, validation tools, expression language code completion, testing tools, and wizards for generating entity objects.
Visual Page Editor	JBoss Developer Studio includes a Visual Page Editor for combined visual and source editing of Web pages. The Visual Page Editor even renders AJAX-enabled RichFaces components.
AJAX Capabilities	JBoss Developer Studio includes a Technology Preview of JBoss RichFaces. RichFaces provides nearly 70 skinnable components, including calendar, tree, dropdown menu, drag-and-drop components, and more. JBoss Developer Studio's Visual Page Editor can render RichFaces components.

Feature	Benefit
JBoss Tools Palette	The JBoss Tools Palette contains a developer's project tag libraries and enables inserting tags into a JSP with one click. The Tools Palette supports custom and 3rd party tags.
JBoss jBPM Tools	JBoss Developer Studio includes jBPM tooling for defining Seam page flows.
Spring Tools	JBoss Developer Studio includes Spring IDE for developing Spring applications.
Struts Tools	JBoss Developer Studio includes Struts tools for Struts 1.x applications.
Optimized JBoss Application Server adapter	JBoss Developer Studio's advanced JBoss AS adapter includes features like incremental deployment for fast development and is pre-configured for the included JBoss Application server.
Red Hat Enterprise Linux and RHN Access	JBoss Developer Studio includes access to Red Hat Enterprise Linux, Linux development tools, and Red Hat Network for development purposes.
Project Examples	Tooling for easy installation of Examples.
JBossWeb Service	JBoss WS is a web service framework developed as a part of the JBoss Application Server. It implements the JAX-WS specification that defines a programming model and run-time architecture for implementing web services in Java, targeted at the Java Platform, Enterprise Edition 5 (Java EE 5).
Usage reporting	The usage plug-in anonymously gathers information on how you use the provided tools. This information is used to improve and shape the JBoss Developer Studio product to better cater to your needs. For more information refer to Section 1.9, "Usage Reporting" .

[JBDS](#) requires JDK 5+ (JDK 6 in case of EAP 5.0). Also it's important that the JVM's architecture matches the particular [JBDS](#) build, i.e. if you're on a 64-bit system and use the 64-bit JVM, you need the 64-bit JBDS.

For the 32-bit version of JBDS use the 32-bit JVM or just use `-vm` flag to tell JBDS to use 32-bit java if the 64-bit one is found as the default.



Note:

On Linux system with 64-bit JVM the visual editor may fail to start. In this case you need the `libstdc++.so.5` library. See also the [FAQ](#).

1.3. Configuring Your Java Environment

You must have a working installation of JDK before you install [JBoss Developer Studio](#). Thus in this guide we will show you how to install a 32-bit Sun JDK 5.0 on a Linux Platform and Microsoft Windows Platform.

1.3.1. Installing and Configuring 32-bit Sun JDK 5.0 on Linux

To install 32-bit Sun JDK 5.0 on Linux and configure it, you should follow the next steps:

- Download the [Sun JDK 5.0 \(Java 2 Development Kit\)](http://java.sun.com/javase/downloads/index_jdk5.jsp) [http://java.sun.com/javase/downloads/index_jdk5.jsp] from Sun's website. Choose "JDK 5.0 Update <x>" (where "x" is the latest update number) for download and then select "Red Hat Package Manager in self-extracting" file for Linux. Read the instructions on Sun's website for installing the JDK.
- If you don't want to use SysV service scripts you can install the "self-extracting file" for Linux instead of choosing the "RPM in self-extracting" file. In that case you can skip the next step mentioned here. But it is recommended to use the SysV service scripts for production servers.
- Download and install the appropriate -compat RPM from JPackage [here](ftp://jpackage.hmdc.harvard.edu/JPackage/1.7/generic/RPMS.non-free/) [ftp://jpackage.hmdc.harvard.edu/JPackage/1.7/generic/RPMS.non-free/]. Please ensure you choose a matching version of the -compat package to the JDK you've installed.
- Create an environment variable that points to the JDK installation directory and call it JAVA_HOME. Add [\\$JAVA_HOME/bin](#) to the system path to be able to run java from the command line. You can do this by adding the following lines to the .bashrc file in your home directory.

```
#In this example /usr/java/jdk1.5.0_11 is the JDK installation directory.  
export JAVA_HOME=/usr/java/jdk1.5.0_11  
export PATH=$PATH:$JAVA_HOME/bin
```



Note:

If you have JDK already installed and added in your system path, you should add `$JAVA_HOME/bin` before the old `$PATH` (not after it) so that the new version of JDK can be found first, i. e. `export PATH=$JAVA_HOME/bin:$PATH` This way, the machine will pick up the new JVM first. You only need to run "alternative" as a safe guard for the right JVM.

Set this variable for your account doing the installation and also for the user account that will run the server.

- If you have more than one version of JVM installed on your machine, make sure you are using the JDK 1.5 installation as the default java and javac. You can do this using the alternatives system. The alternatives system allows different versions of Java from different sources to co-exist on your system.

1.3.1.1. Select alternatives for java, javac and java_sdk_1.5.0

- As a root user, type the following command at the shell prompt and you should see something like this:

```
[root@vsr ~]$ /usr/sbin/alternatives --config java
There are 2 programs that provide 'java'.
Selection    Command
-----
1            /usr/lib/jvm/jre-1.4.2-gcj/bin/java
*+ 2        /usr/lib/jvm/jre-1.5.0-sun/bin/java
Enter to keep the current selection[+], or type selection number:
```

Make sure the Sun version [jre-1.5.0-sun in this case] is selected (marked with a '+' in the output), or select it by entering its number as prompted.

- Repeat the same for javac and java_sdk_1.5.0.

```
[root@vsr ~]$ /usr/sbin/alternatives --config javac
There is 1 program that provides 'javac'.
Selection    Command
-----
*+ 1          /usr/lib/jvm/java-1.5.0-sun/bin/javac
Enter to keep the current selection[+], or type selection number:
```

```
[root@vsr ~]$ /usr/sbin/alternatives --config java_sdk_1.5.0
There is 1 program that provide 'java_sdk_1.5.0'.
Selection    Command
-----
*+ 1          /usr/lib/jvm/java-1.5.0-sun
Enter to keep the current selection[+], or type selection number:
```

You should verify that java, javac and java_sdk_1.5.0 all point to the same manufacturer and version.



Note:

You can always override this step by setting the `JAVA_HOME` environment variable as explained in the previous step.

- Make sure that the java executable is in your path and that you are using an appropriate version. To verify your Java environment, type "java -version" at the shell prompt and you should see something like this:

```
[root@vsr ~]$ java -version
java version "1.5.0_11"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_11-b03)
Java HotSpot(TM) Client VM (build 1.5.0_11-b03, mixed mode, sharing)
```

1.3.2. Installing and Configuring 32-bit Sun JDK 5.0 on Microsoft Windows

To install and configure 32-bit Sun JDK 5.0 on Microsoft Windows, follow these steps:

- Download the [Sun JDK 5.0 \(Java 2 Development Kit\)](http://java.sun.com/javase/downloads/index_jdk5.jsp) [http://java.sun.com/javase/downloads/index_jdk5.jsp] from Sun's website. Choose "JDK 5.0 Update <x>" (where "x" is the latest update number) for download and then select your Windows Platform options to perform the installation.
- Create an environment variable called `JAVA_HOME` that points to the JDK installation directory, for example: `C:\Program Files\Java\jdk1.5.0_11\`

In order to run java from the command line, add the `java\bin` directory to your path, for example:

```
C:\Program Files\Java\jdk1.5.0_11\java\bin
```

To do this, open the [Control Panel](#) from the [Start](#) menu, switch to Classic View if necessary, open the System Control Panel applet ([System](#)), select the [Advanced](#) Tab, and click on the [Environment Variables](#) button.

Now, when 32-bit Sun JDK 5.0 has been successfully installed, we can pass on to the next step.

1.4. JBoss Developer Studio Installation

This chapter will provide you with detailed information on how to install [JBoss Developer Studio](#).

JBDS comes with a simple installer, bundled with tested/pre-configured versions of Eclipse, WTP, JBossEAP, Seam, and SpringIDE. Thus, to start perform the next steps:

Chapter 1. Getting Started with JBoss Developer Studio

- First of all you need the appropriate installation file for your platform from [Red Hat website](https://www.redhat.com/apps/store/developers/jboss_developer_studio.html) [https://www.redhat.com/apps/store/developers/jboss_developer_studio.html].
- Then run in console:

```
java -jar jbdevstudio-eap-linux-gtk-3.0.0.GA.jar
```

- Follow the instructions presented by the installation wizard:

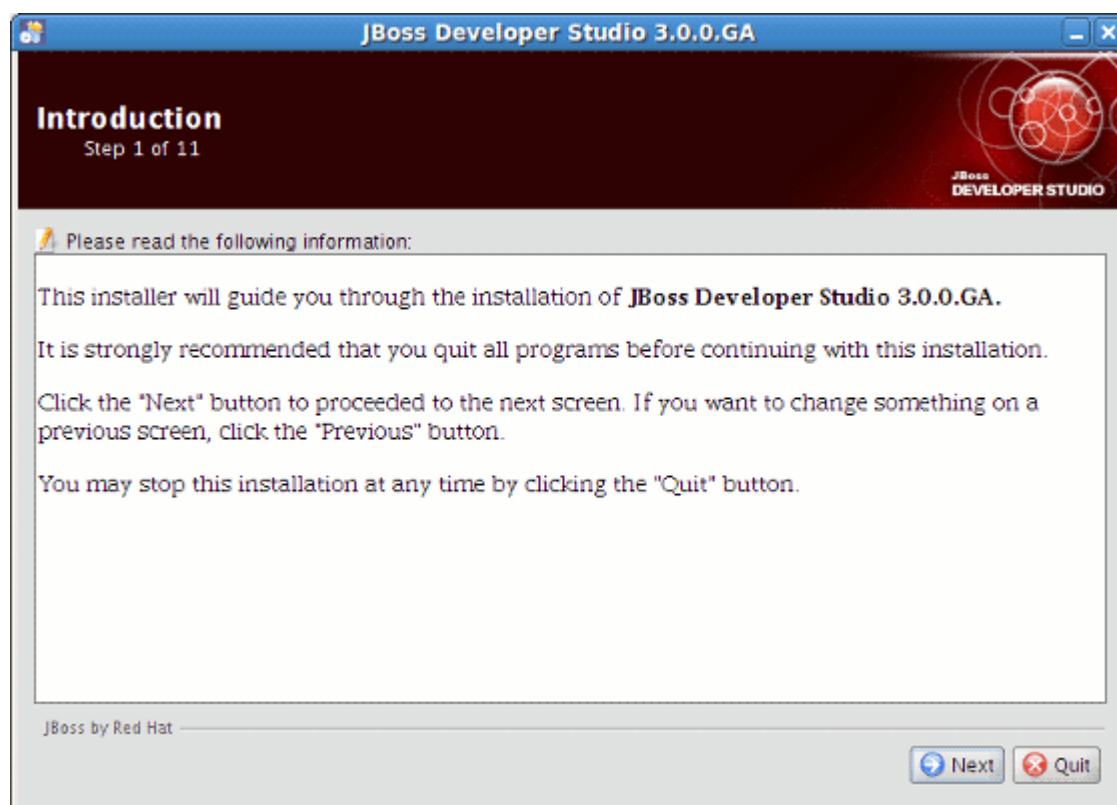


Figure 1.1. JBoss Developer Studio Installation Wizard

- Provide the installation path
- Select Java VM



Figure 1.2. Select Java VM



Tip:

By selecting [Default Java VM](#) you set default Java VM of your system (to verify your Java environment, type "java -version" in console).

Selecting [Specific Java VM](#) you can provide the path to non-default Java VM.



Note:

JBoss Developer Studio 3.0 comes integrated with JBoss EAP 5.0 that requires JDK 6.0 to run and "gij" isn't available on every platform.

- Installation process includes [JBoss Enterprise Application Platform](http://www.jboss.com/products/platforms/application) [http://www.jboss.com/products/platforms/application]. Select [Yes](#) to use it in JBoss Developer Studio.



Note:

Like in the previous version of JBDS, JBDS 3.0.0.GA comes integrated with JBoss EAP 5.0 that supports EAP 5 adapter and Seam 2.2.

This step lets you configure locally available JBoss Application Servers:



Figure 1.3. JBoss Enterprise Application Platform Installing

- You can fill the list automatically using the [Find](#) button: click [Find](#), select a folder where search of available JBoss Application Servers should be started and click on [Ok](#).

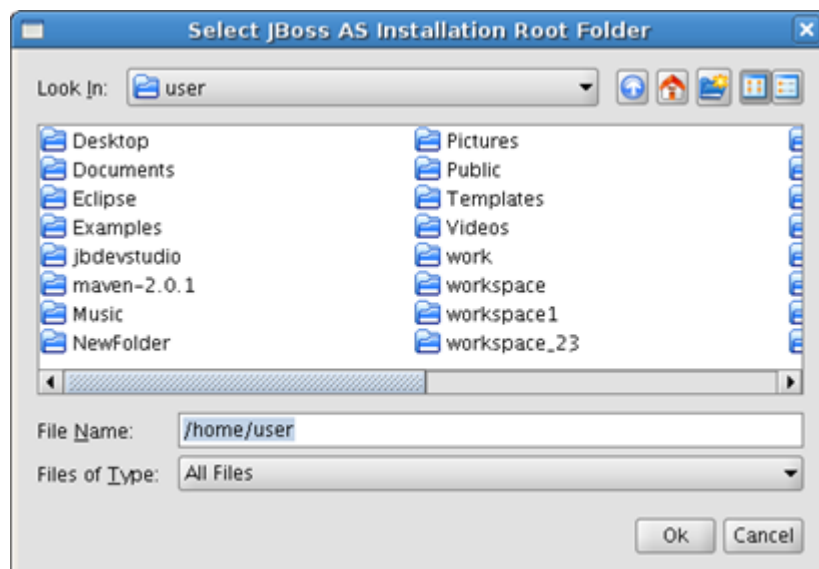


Figure 1.4. Finding Servers in the Selected Directory

- All available servers in the selected directory will be added to the list with the following information specified: Name, Type, Version and Location.
- You can also add servers one by one using the [Add](#) button:

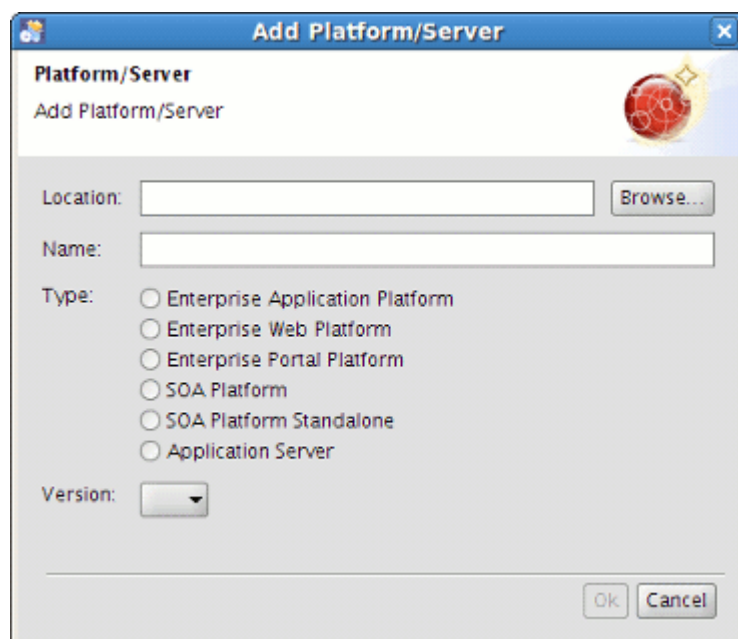


Figure 1.5. Add Server to be Configured

Click on [Browse](#) and select the server location. Fields Name, Type and Version will be filled in automatically:

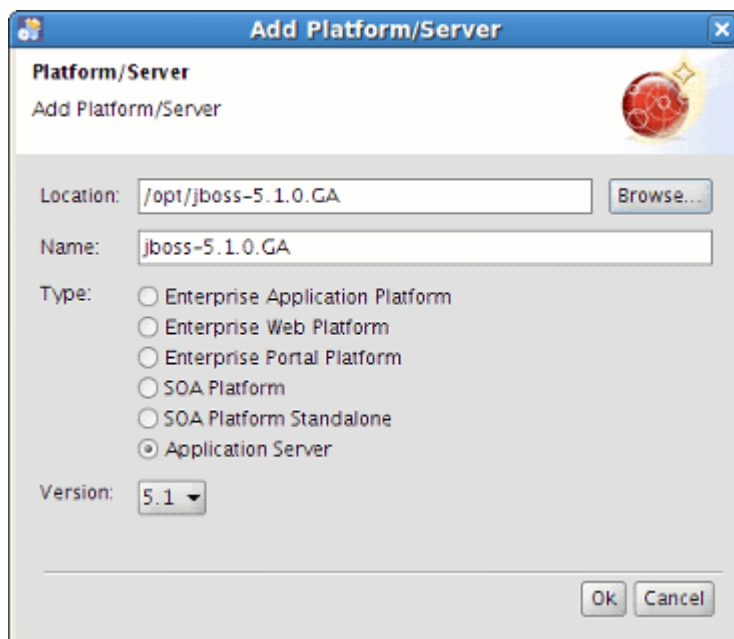


Figure 1.6. Specify Server Location

Click on [Ok](#). The server is added to the list with the details on its type, version and location.

The [Remove](#) button will remove the selected server from the list. If necessary, you can edit the server details clicking on the [Edit](#) button:

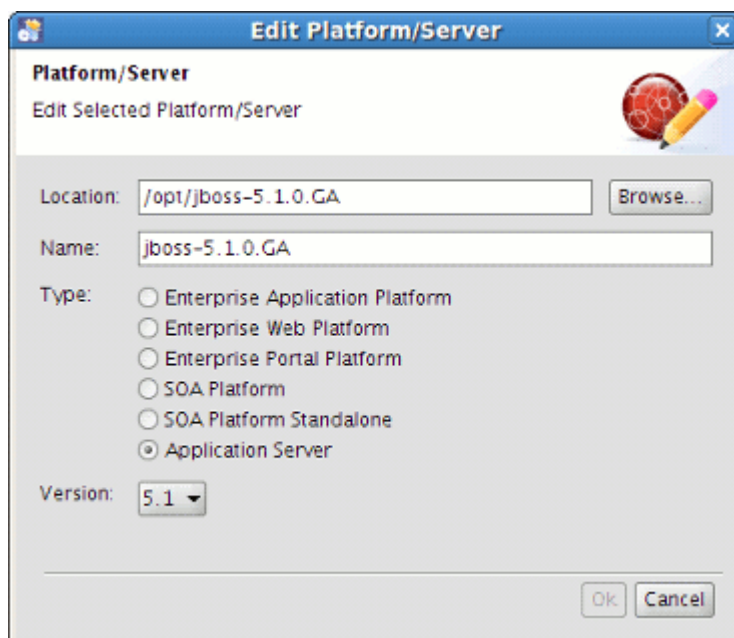


Figure 1.7. Edit Server

- Click [Next](#). Check your installation paths and see the components to install. If you'd like to change something, press the [Previous](#) button. Click [Next](#) to start installation.

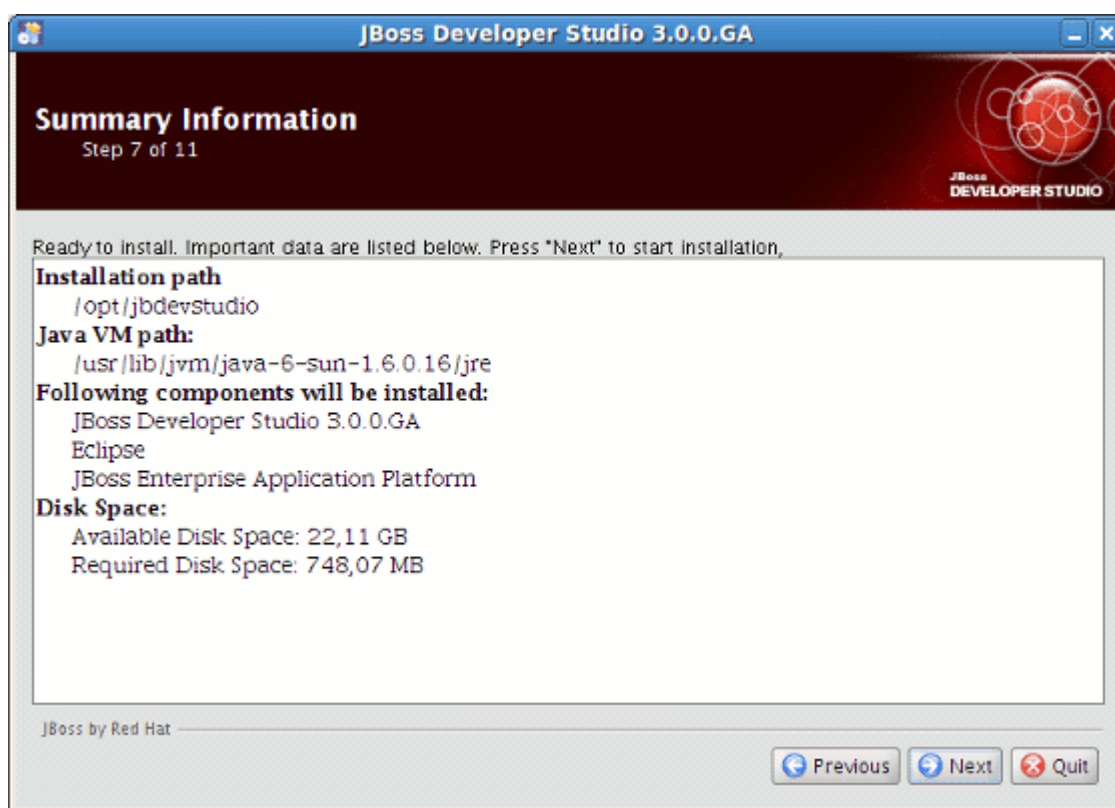


Figure 1.8. Summary Information

1.5. JBoss Developer Studio and JBoss Tools

This section uncovers the points on the differences between [JBoss Developer Studio](#) and [JBoss Tools](#) and provides the steps on [JBoss Tools](#) installation as well.

1.5.1. What is the difference?

[JBoss Tools](#) is what went into our [JBoss Developer Studio](#) which comes as an easy-to-install Eclipse based IDE fully configured and ready to run with the bundled [JBoss Enterprise Application Platform](#).

In short [JBoss Tools](#) are just a set of Eclipse plugins and [JBoss Developer Studio](#) adds:

- An installer
- Eclipse and Web Tools preconfigured
- JBoss EAP with JBoss AS and Seam preconfigured
- 3rd party plugins bundled and configured
- Access to RHEL and Red Hat Network

- Access to the JBoss/Red Hat supported software

For additional information see [JBoss.com](http://www.jboss.com/products/devstudio) [http://www.jboss.com/products/devstudio]

1.5.2. JBoss Tools Installation and Update

Here, let's consider the installation of the [JBoss Tools](#) modules.

[JBoss Tools](#) is an umbrella project for the JBoss developed plugins that will make it into [JBoss Developer Studio](#). The JBoss Tools modules are:

- JBoss AS Tools
- Seam Tools
- Hibernate Tools
- Visual Page Editor
- JST Tools
- JBPM Tools
- Smooks Tools

To install the JBoss Tools plugins for Eclipse, you need the following:

- Get Eclipse and Web Tools (make sure you have the [appropriate version of Eclipse](http://www.jboss.org/community/wiki/MatrixofsupportedplatformsruntimesandtechnologiesinJBossToolsJBDS) [http://www.jboss.org/community/wiki/MatrixofsupportedplatformsruntimesandtechnologiesinJBossToolsJBDS]).

The current version of JBoss Tools (3.1.0.GA) targets at Eclipse 3.5.2 Galileo JEE Bundle which is available from [eclipse.org](http://www.eclipse.org/downloads/) [http://www.eclipse.org/downloads/].



Tip:

We recommend you do not use Eclipse.org update site to go from old version of Eclipse to new one. Instead we suggest that you download the full binary from [here](http://www.eclipse.org/downloads/) [http://www.eclipse.org/downloads/].

If you can only use Eclipse 3.3 use [JBoss Tools 2.1.2](http://www.jboss.org/tools/download.html) [http://www.jboss.org/tools/download.html], but JBoss Tools 2.x will not have any of the new features.



Note:

Remember to choose the download that matches your OS and use Java 6 when you run it.

If you need to install any standalone plug-in from JBoss Tools visit a [JBoss Tools Wiki](http://community.jboss.org/wiki/InstallingJBossTools) [http://community.jboss.org/wiki/InstallingJBossTools] page to read about dependencies between standalone plug-ins.

For more information about how to install JBoss Tools 3.0 and JBoss Tools 3.1 please read the following docs:

- [JBoss Tools 3.0 Installation](https://www.jboss.org/tools/download/installation/update_3_0.html) [https://www.jboss.org/tools/download/installation/update_3_0.html]
- [JBoss Tools 3.1 Installation](https://www.jboss.org/tools/download/installation/update_3_1.html) [https://www.jboss.org/tools/download/installation/update_3_1.html]

1.5.2.1. Installation From Update Site

Instead of downloading installation/updating manually, it's also possible to get the latest release of [JBoss Tools](#) from one of our update sites:

- Stable Updates: <http://download.jboss.org/jbosstools/updates/stable>
- Development Updates: <http://download.jboss.org/jbosstools/updates/development>
- Nightly Builds: <http://download.jboss.org/jbosstools/updates/nightly/trunk/>



Note:

During the JBT installation/updating process using some of Jboss update sites *No repository found containing...* error can appear. It is a Eclipse P2 bug. To work around it try: 1. Restart your Eclipse. 2. Add the update site again. 3. Do the update again.

To install/update Jboss Tools from one of the update sites you should follow the steps:

- First, download the correct version of Eclipse for your platform which is compatible with the version of JBoss Tools you plan to install. For example if you want to install 3.1.0.GA Eclipse 3.5.2 Galileo JEE Bundle is required.
- If you do not already have Java 5 or 6 installed and in your path, download and install it.
- When Eclipse is started, go to
 - Help->Software Updates... ->Available Software->Manage Sites...->Add... (Eclipse 3.4)
 - Help->Install New Software->Add.. (Eclipse 3.5)
- After clicking *Add* button *Add Site* View should appear. In the *Location* field you should paste one of the update sites described [above](#). On some platforms, you can also drag and drop the link from your browser directly into Eclipse. In *Name* you can write any name you like. In the end press the *Ok* button.

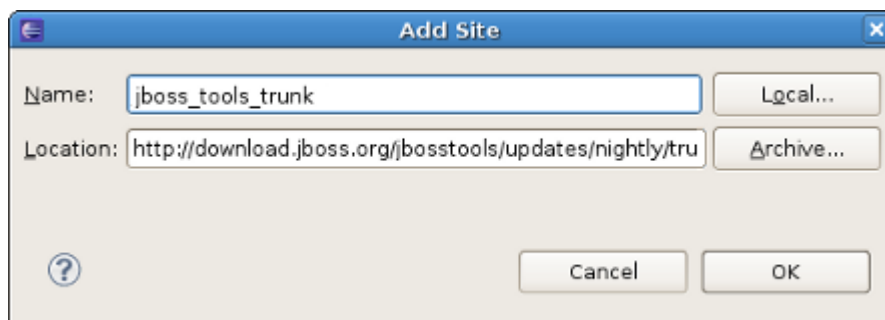


Figure 1.9. Add Update Site

- In the [Install](#) View select features to install, then click *Next*.

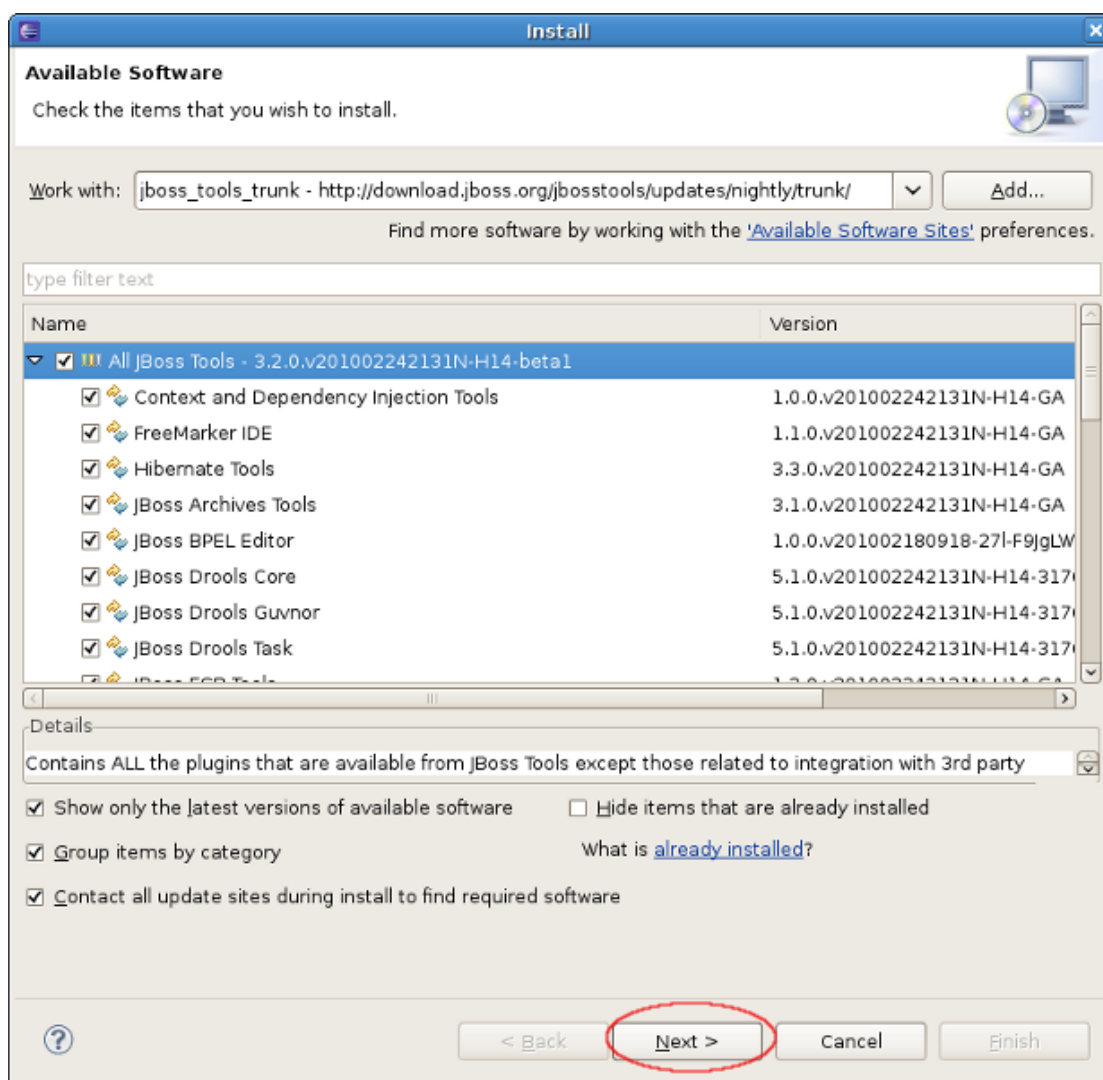


Figure 1.10. Jboss Tools installation

- And the system checks whether all necessary requirements and dependencies are installed on your Eclipse.

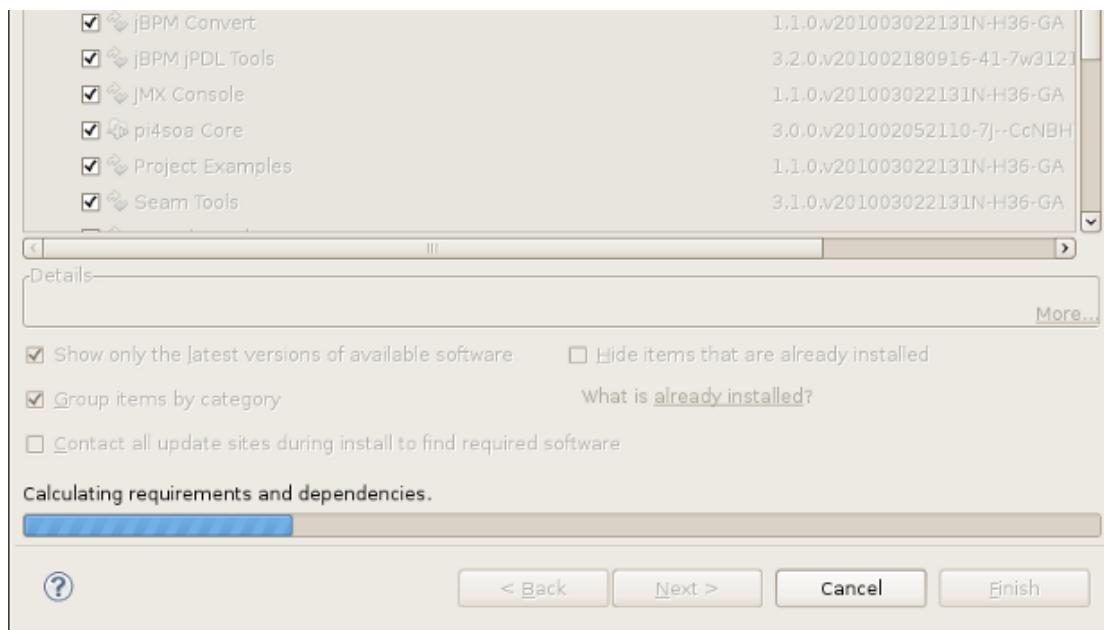


Figure 1.11. Requirements and dependencies check

If some of them are missing an error message will appear and the installation process will be terminated.



Note:

Sometimes when a user changes the list of plugins for download and clicks *Next*, after required dependencies check, the list of missing dependencies of the previous attempt is showed. In this case the completely close of the *Install New Software* dialog can help.

If Jboss Tools are already installed on your Eclipse and you are doing updating but not installation the following messages will appear in your *Details* section. Ignore them and press *Next*.

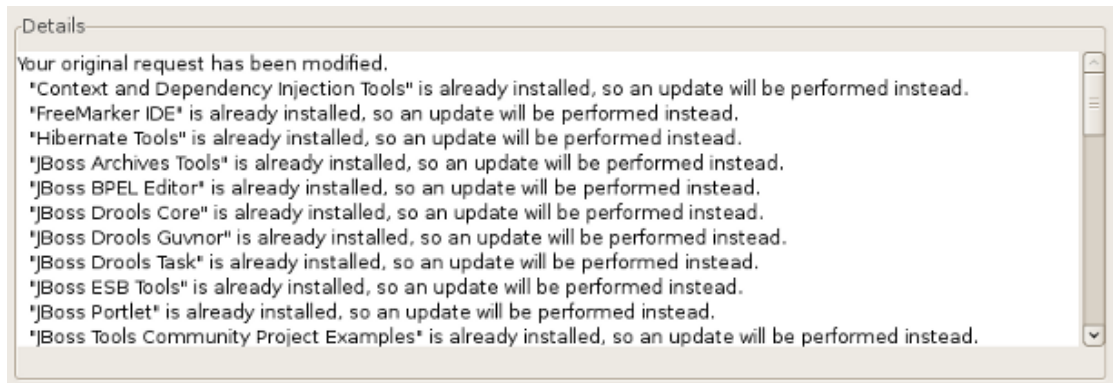


Figure 1.12. Possible messages

- When you click Finish the processes of Installation/Updating starts up.
- Install. Restart when prompted.

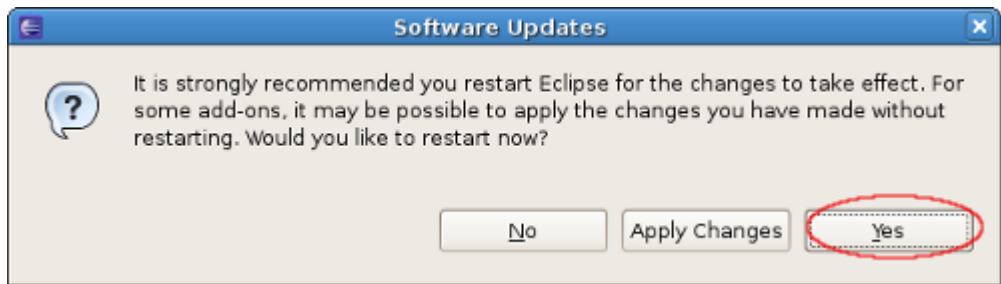


Figure 1.13. Message dialog

- After fetching all the features and plugins for installation, you may be prompted that your selection contains unsigned jars. This is normal, simply accept the warning and continue.

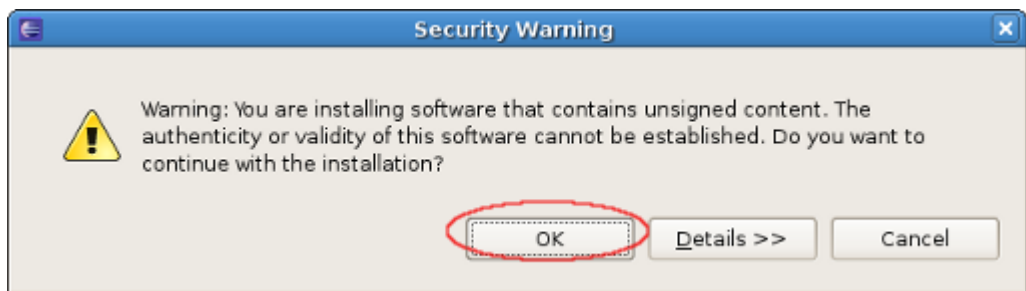


Figure 1.14. Warning Message



Note:

While installing JBoss Tools on GNU/Linux (64-bit) through the eclipse installation manager you may run into some problems which can interrupt

your installation process. Please, read the [Manual installation of JBoss Tools 3.1.x on Eclipse 3.5 on GNU/Linux \(64-bit\)](http://www.jboss.org/community/wiki/ManualinstallationofJBossTools31xonEclipse35onGNULinux64-bit) [http://www.jboss.org/community/wiki/ManualinstallationofJBossTools31xonEclipse35onGNULinux64-bit] wiki article which provides you with the necessary steps to do it correct.

1.5.2.2. Installation From Zip file

Now you can install JBoss Tools using Eclipse P2 update manager and a common zip file which is one installer for all platforms.

You can download Jboss Tools in a zip file from one of the pages below:

- Stable Releases: <http://www.jboss.org/tools/download/stable.html>
- Development Updates: <http://www.jboss.org/tools/download/dev.html>
- Nightly Builds: <http://download.jboss.org/jbosstools/updates/nightly/>

When you downloaded Jboss Tools Zip please *DO NOT UNZIP IT* and follow the steps described below:

- When Eclipse is started, go to [Help -> Install New Software...](#) -> [Add...](#) -> [Archive...](#)
- Browse for the update site zip. In the *Name* field you can write any name you like. In the end press [Ok](#) button.

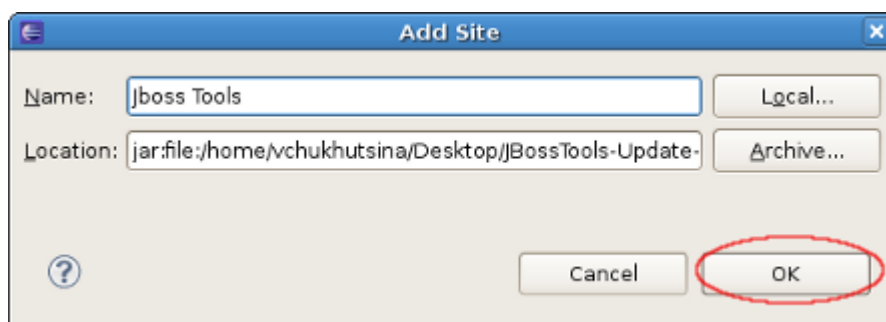


Figure 1.15. Installation from a zip file

- All the other steps are the same as it was described in the [Installation From Update Site section](#).

1.6. Welcome to JBoss Developer Studio

In this section we'll show you how to work with the welcome page of the [JBoss Developer Studio](#).

The welcome page is the first page you see when you first launch [JBoss Developer Studio](#).

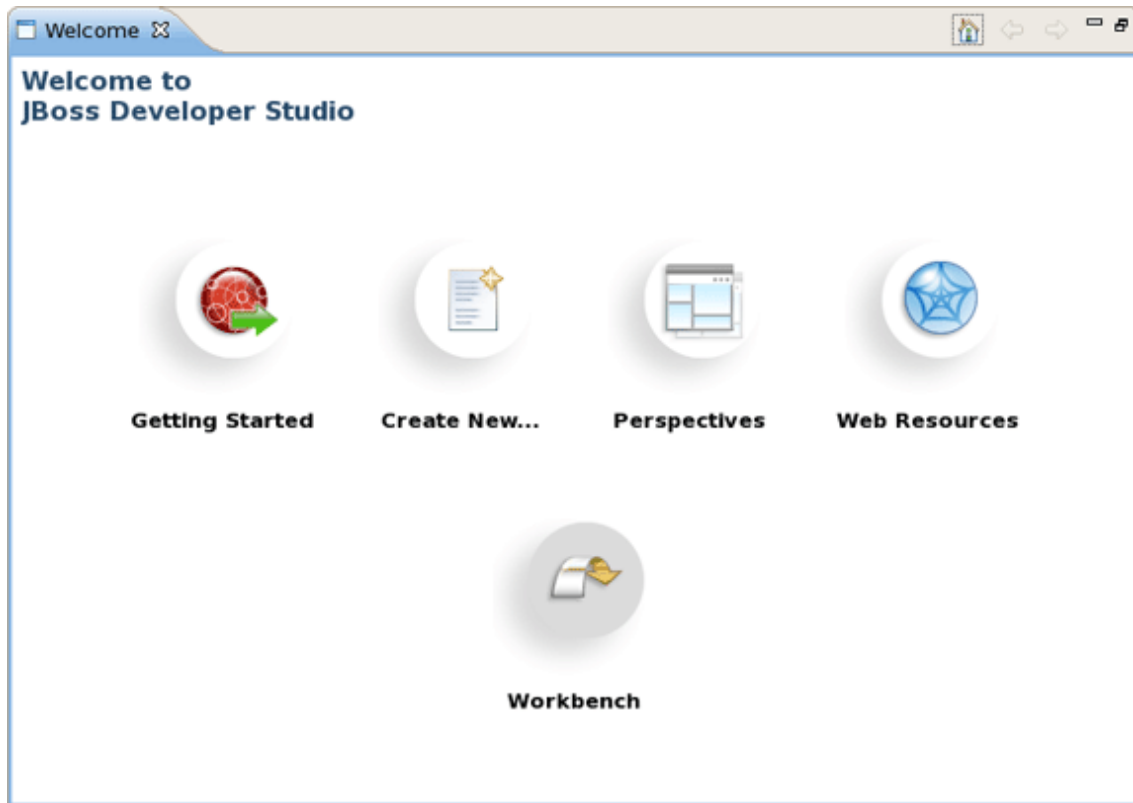


Figure 1.16. Welcome to JBoss Developer Studio

With the help of its page you will be able:

- to get quick access to Getting Started Documentation (guides, tutorials and viewlets)

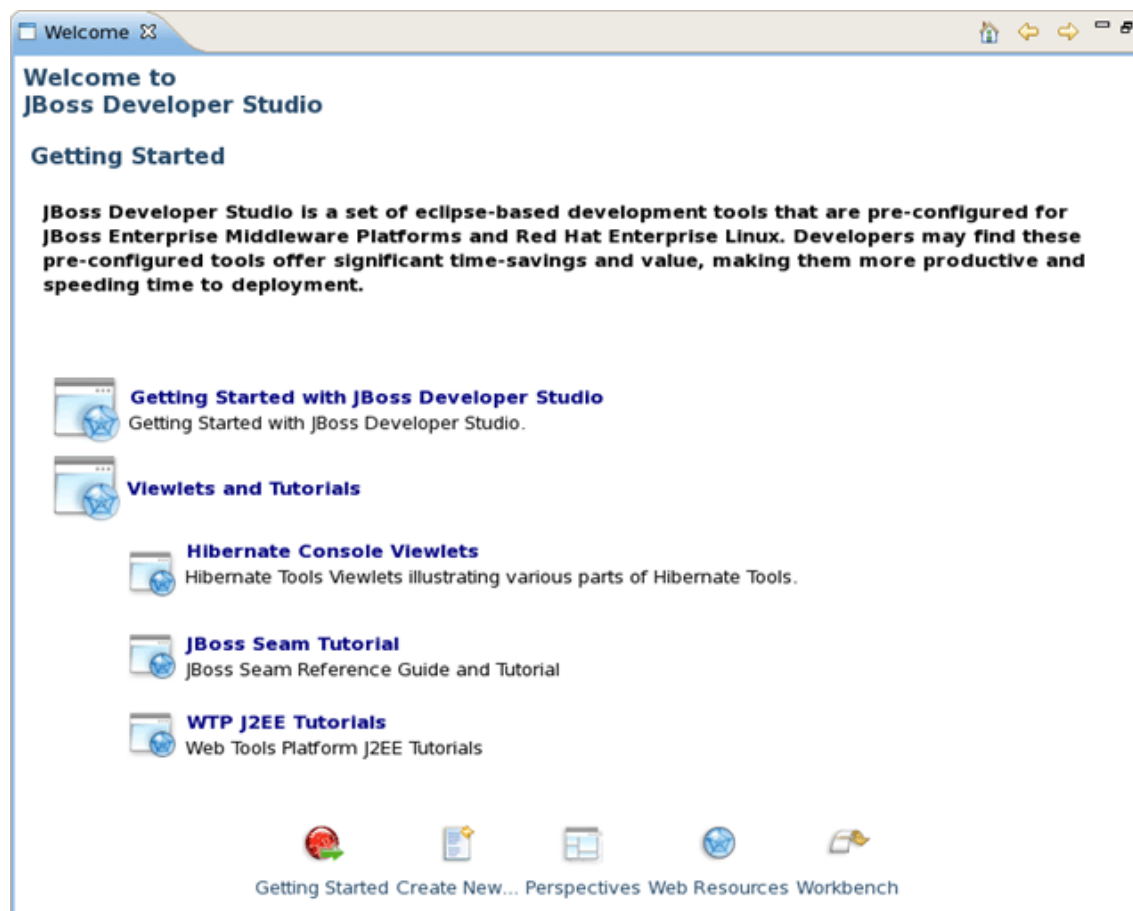


Figure 1.17. Getting Started Documentation

- to create new Seam projects, jBPM Process, JSF or Struts projects using JBDS wizards

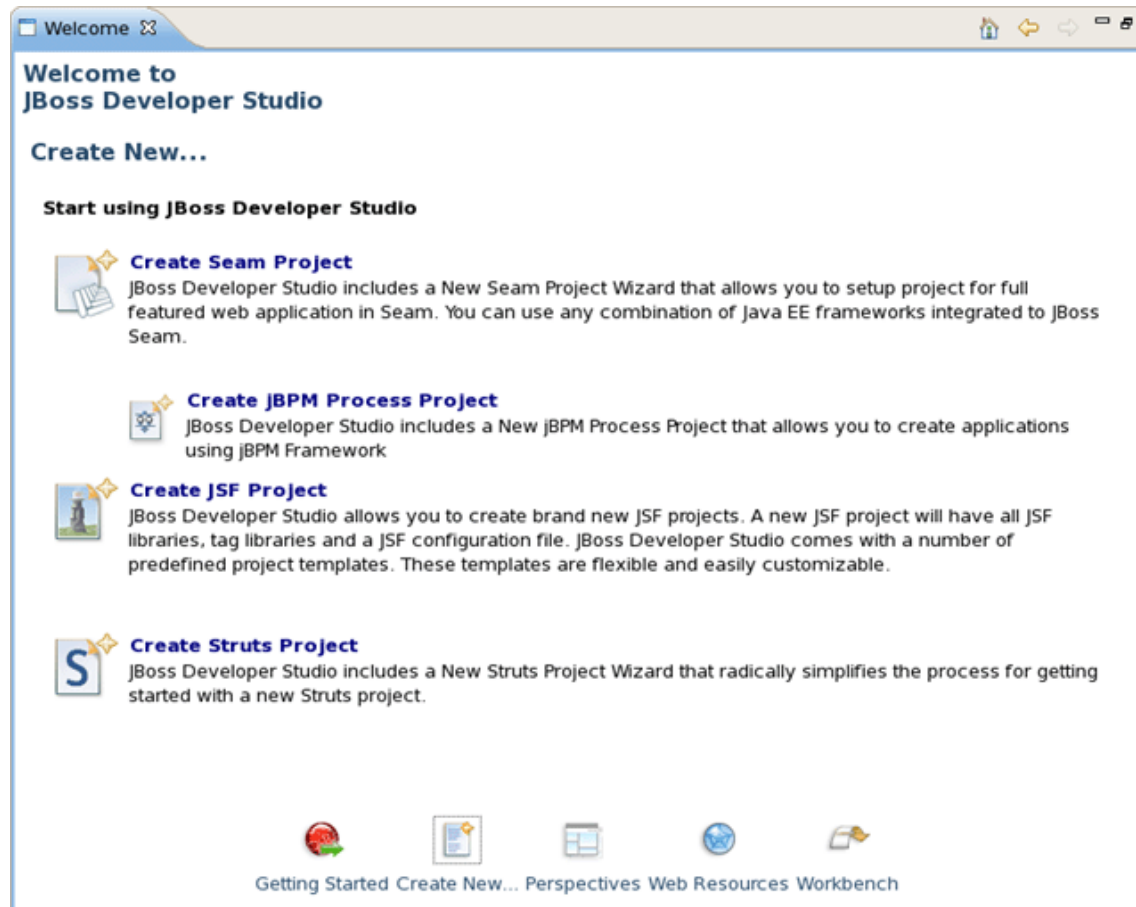


Figure 1.18. Create New...

- to get short description of perspectives that JBDS offers for more productive development

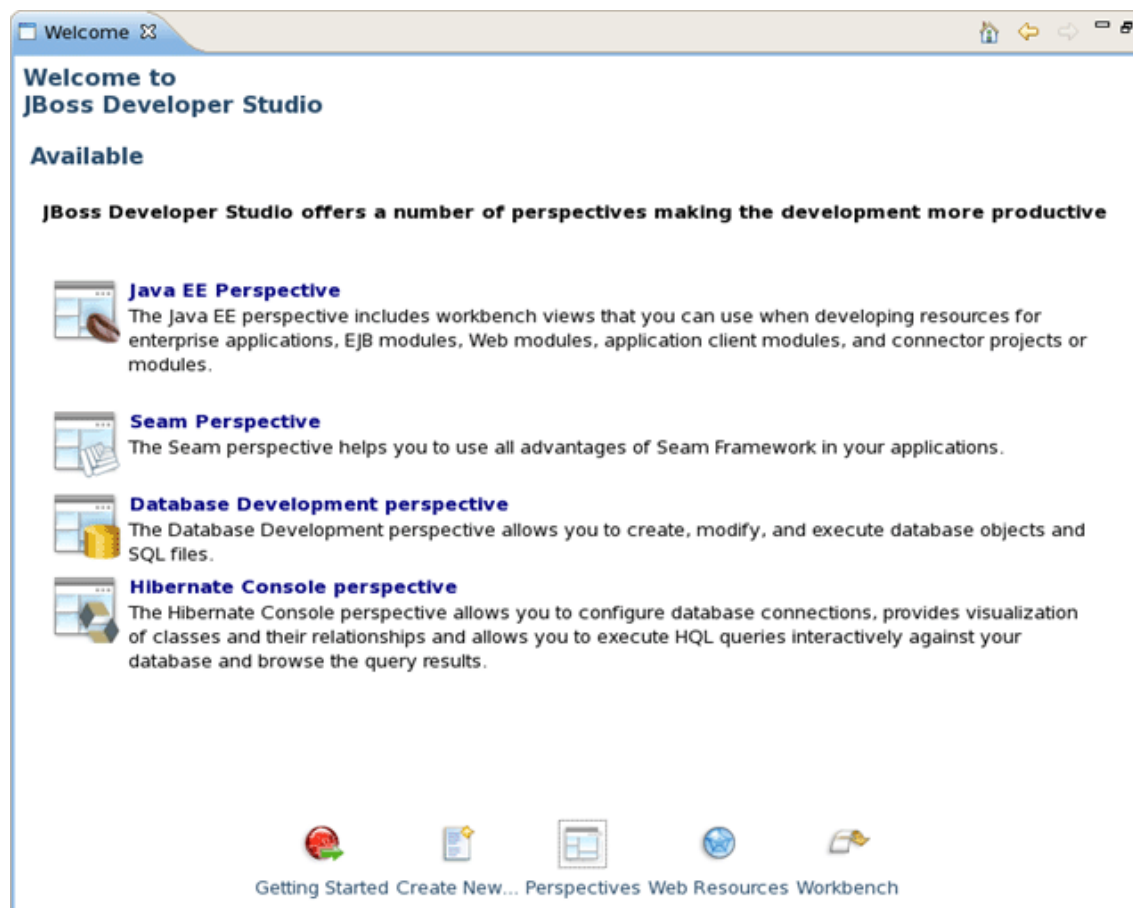


Figure 1.19. Perspectives

- to visit JBoss Developer Studio web resources.

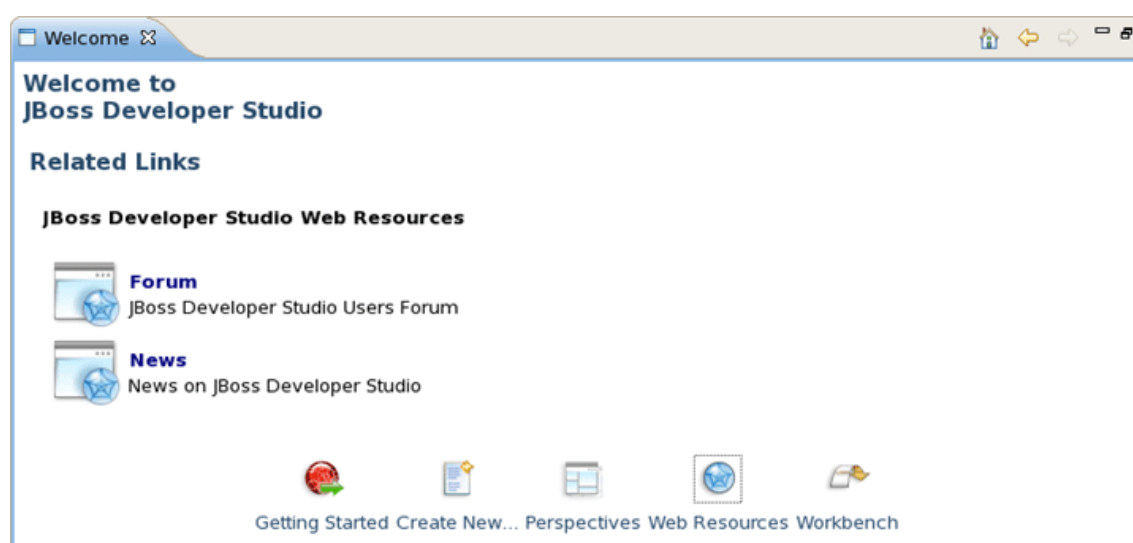


Figure 1.20. Web Resources

Start work with [JBoss Developer Studio](#) by clicking on [Workbench](#) button or simply close the Welcome page.

1.7. Upgrading

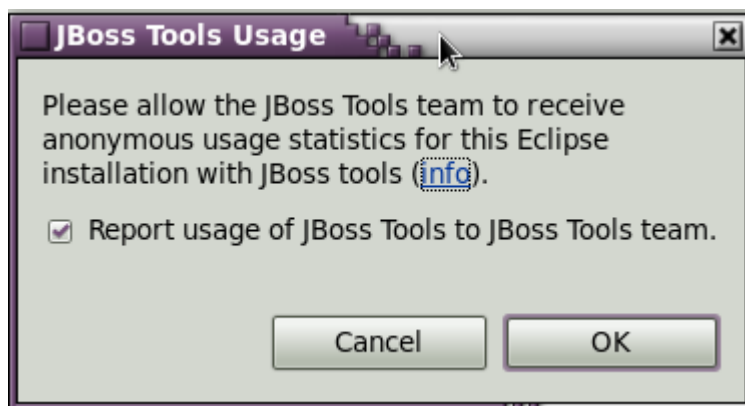
As opposed to upgrading your JBDS from 2.0 to 2.1 using the update site, to upgrade your JBDS from 2.1 to 3.0 you have to uninstall your 2.1 version and install the new one because of some platform change.

1.8. Uninstalling

- Make sure [JBoss Developer Studio](#) is not running
- Uninstall your current version of [JBoss Developer Studio](#) by running Uninstaller

1.9. Usage Reporting

The JBoss Developer Studio now includes a usage plug-in that anonymously reports information back to JBoss. The plug-in is not enabled by default. To enable, make sure the **Report usage of JBoss Tools to JBoss Tools team.** box is ticked.



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 Preferences**.

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 focussing 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 1.9.1, "Collected usage information guide"](#).

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

1.9.1. Collected usage information guide

Outlined is the information type being reported and the Goolge Analytics fieds implemented to gather this information.

Installed components

The **Keyword** field has been modified to report the installed JBoss Tools 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.

Locations

The **Locations** field reports the geographical location where the product is being used. Sample returned values include: Munich, Wroclaw and Moscow.

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 the product is running on. Sample returned values include: Linux, Macintosh and Windows.

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.

1.10. Support

If you have comments or questions, you can discuss them at our [JBoss Developer Studio Forum](http://www.jboss.com/index.html?module=bb&op=viewforum&f=258) [http://www.jboss.com/index.html?module=bb&op=viewforum&f=258].

When writing to the forum for questions, please include the following information:

1. JBoss Developer Studio version
2. Exact error message
3. Steps to reproduce the issue

JBDS subscribers can get necessary support on our [Support Portal](https://network.jboss.com/jbossnetwork/login.html) [https://network.jboss.com/jbossnetwork/login.html].

1.11. Other relevant resources on the topic

JBDS on RedHat: [JBoss Developer Studio](http://www.redhat.com/developer_studio) [http://www.redhat.com/developer_studio]

Forum: [JBoss Forum](http://www.jboss.com/index.html?module=bb&op=viewforum&f=258) [http://www.jboss.com/index.html?module=bb&op=viewforum&f=258]

Wiki: [JBossToolsWiki](http://www.jboss.org/community/wiki/JBossTools) [http://www.jboss.org/community/wiki/JBossTools]

Subscription: [JBDS Subscription](https://www.redhat.com/apps/store/developers/jboss_developer_studio.html) [https://www.redhat.com/apps/store/developers/jboss_developer_studio.html]

The latest documentation builds are available [here](http://download.jboss.org/jbosstools/nightly-docs/) [http://download.jboss.org/jbosstools/nightly-docs/].

[Matrix of supported platforms, runtimes and technologies in JBossTools/JBDS](http://www.jboss.org/community/wiki/MatrixofsupportedplatformsruntimesandtechnologiesinJBossToolsJBDS) [http://www.jboss.org/community/wiki/MatrixofsupportedplatformsruntimesandtechnologiesinJBossToolsJBDS]

Manage JBoss AS from JBoss Developer Studio

In this chapter we'll focus more on how to operate the [JBoss AS](#) from [JBoss Developer Studio](#).

[JBoss Developer Studio 3.0.0.GA](#) is shipped with [JBoss EAP 5](#). When you followed the default installation of [JBoss Developer Studio](#), you should already have a JBoss EAP 5 Server installed and defined. To run JBoss AS you need preinstalled JDK 6.



Note:

You can leave [JDK 1.5](#) configuration for [JBDS](#) and set [JDK 6](#) as [JRE](#) only for [JBoss EAP Server](#). For this you need on [Windows > Preferences](#) menu click [Server > Runtime Enviroments](#). On the opened preference page you should select [JBoss EAP Runtime](#) and click [Edit](#). In [Edit Server Runtime Enviroment](#) dialog you can configure [Jave Runtime Enviroment](#) by clicking [JRE](#) button.

2.1. How to Manage the JBoss AS Bundled in JBDS

This section covers the basics of working with the [JBoss Server](#) supported directly by [JBDS](#) via bundled AS plug-in. The server points to the JBoss Enterprise Application Platform 5.0 Runtime shipped with [JBDS](#).

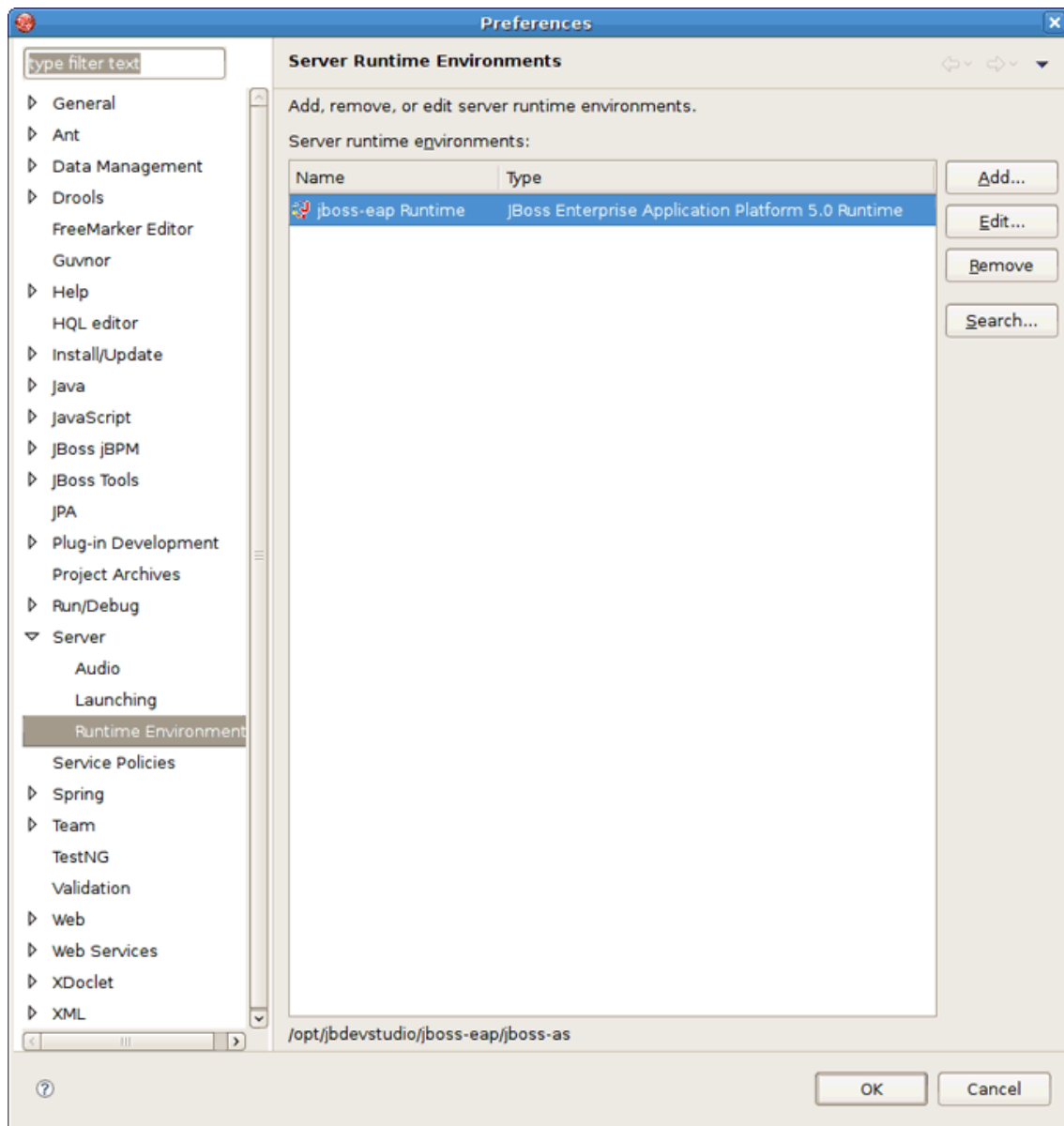


Figure 2.1. JBoss EAP 5.0 Runtime

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 behaviour 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 on the [Servers](#) or right click server name in this view and select [Start](#) . 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](#):

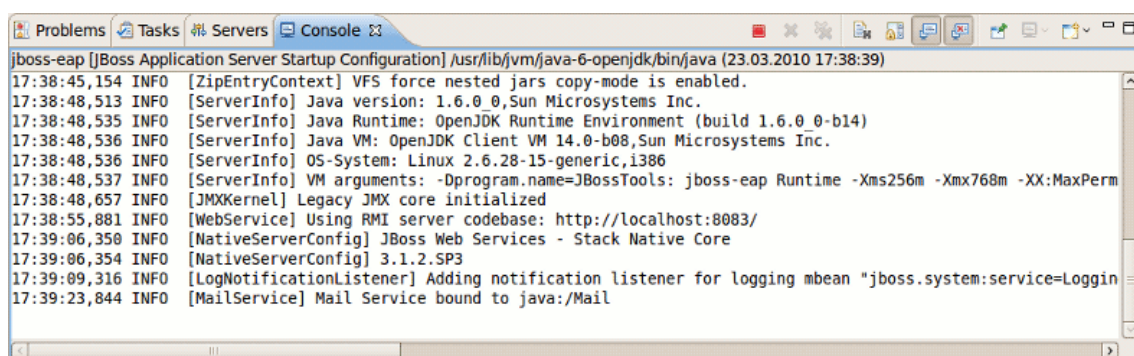


Figure 2.3. Console Output

When the server is started you should see [Started](#) in the square brackets right next its name in [Servers](#).

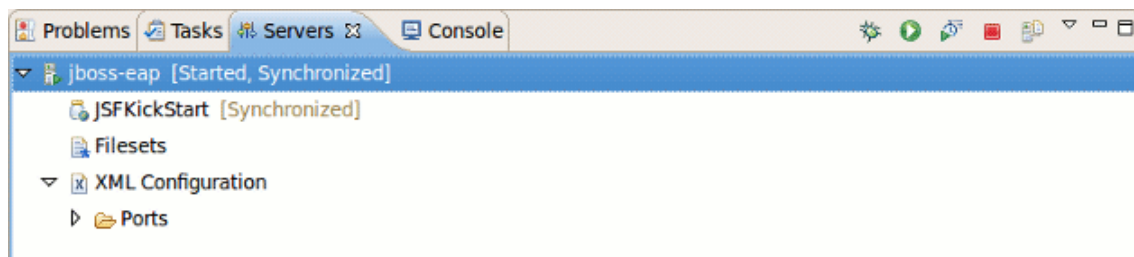


Figure 2.4. Server is Started

2.1.2. Stopping JBoss Server

To stop the server, click the [Stop](#) icon in [Servers](#) or right click the server name and press [Stop](#).

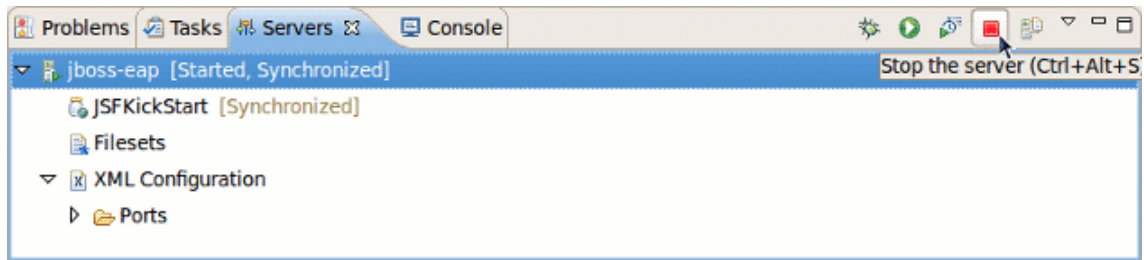


Figure 2.5. Stopping Server

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.

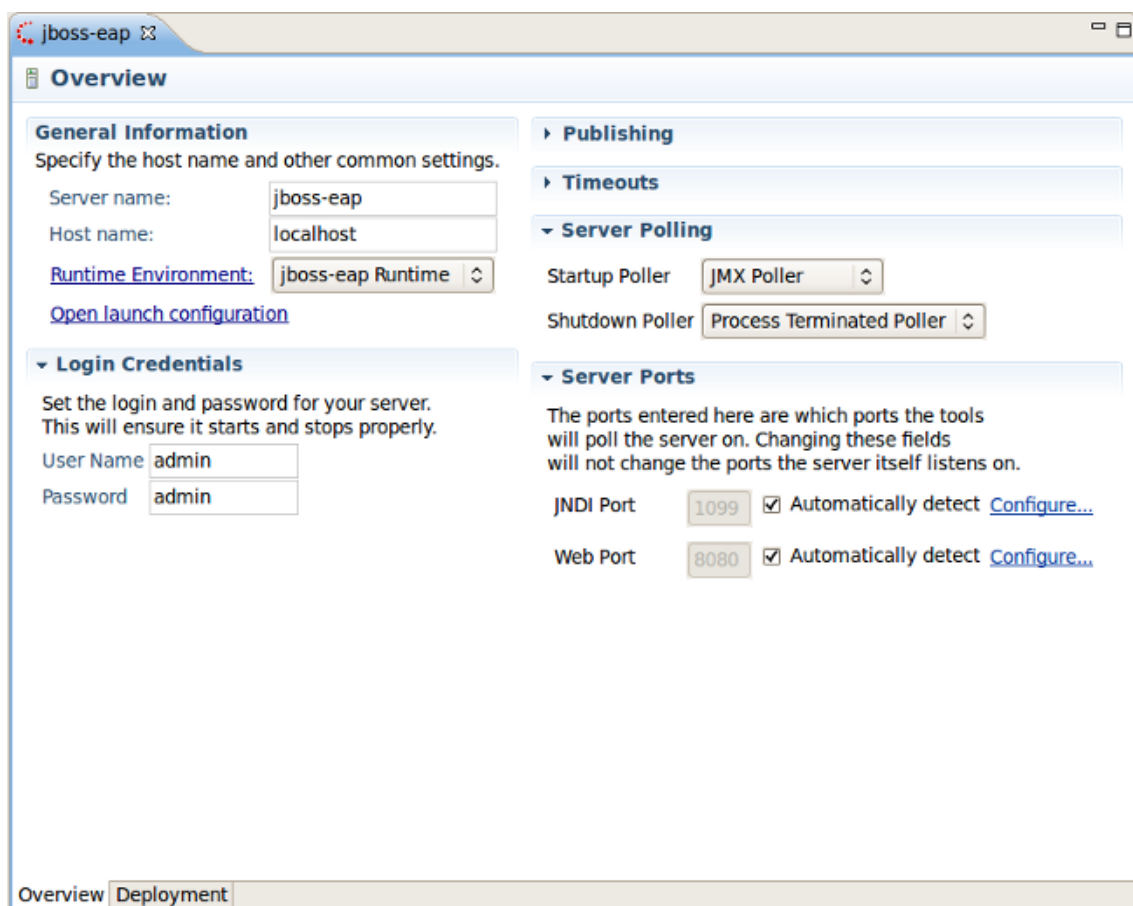


Figure 2.6. Server Overview

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

2.2. How to Use Your Own JBoss AS Instance with JBDS

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 have 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 just installed server into server manager in [JBoss Developer Studio](#).

- Open the [Servers](#) 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](#)

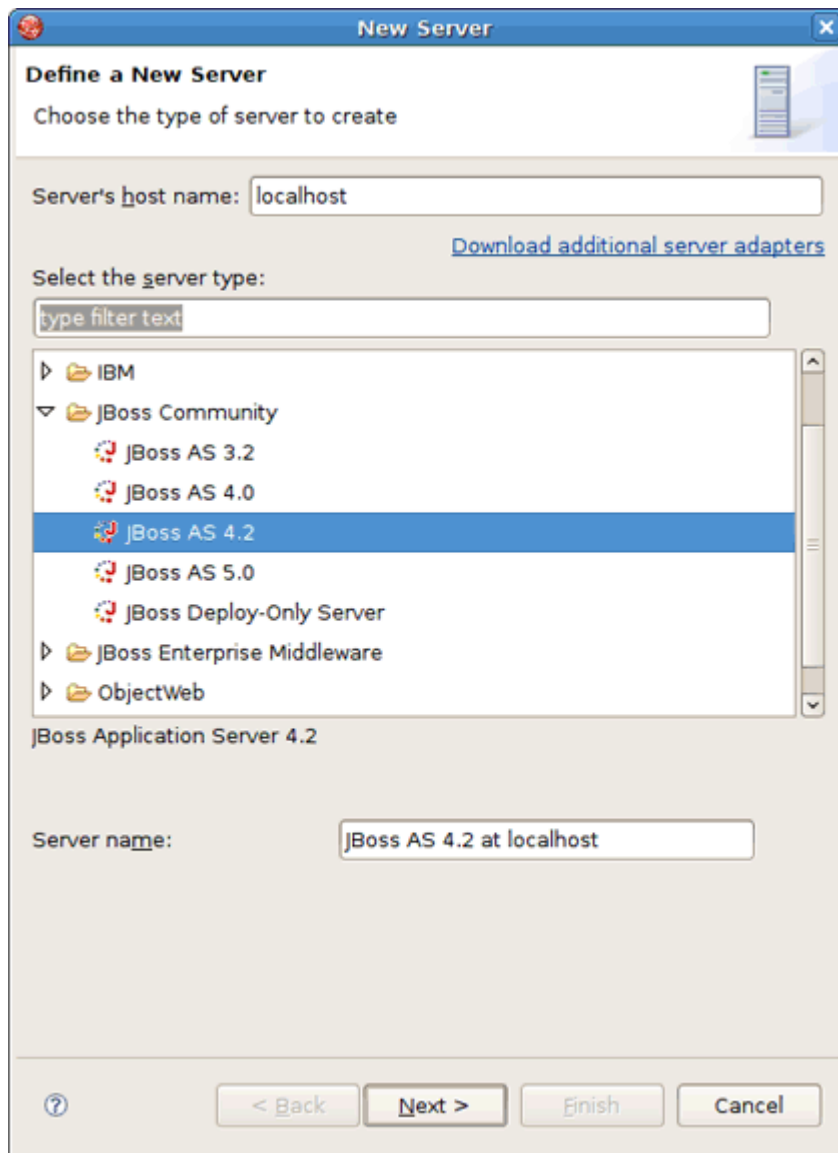


Figure 2.7. Selecting Server Name and Server Type



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 [Next](#)
- In the next step make [JBoss Developer Studio](#) to know where you have installed the Server and define JRE.

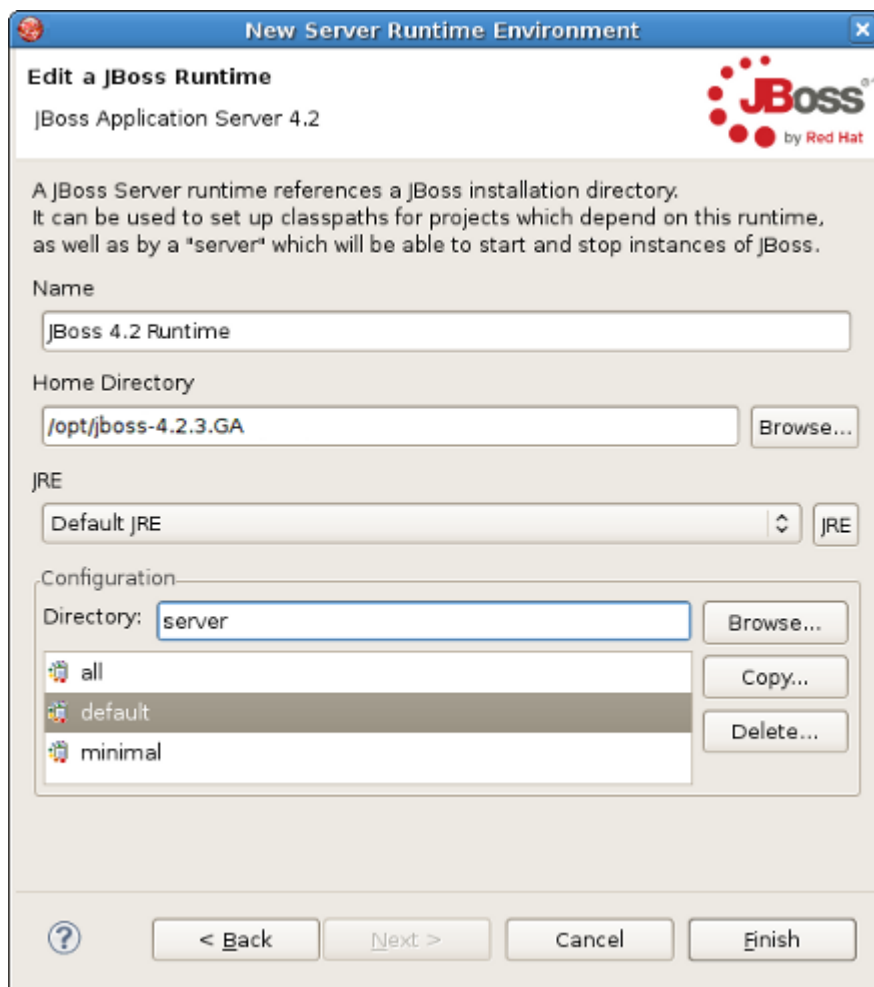


Figure 2.8. Defining JBoss Runtime



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

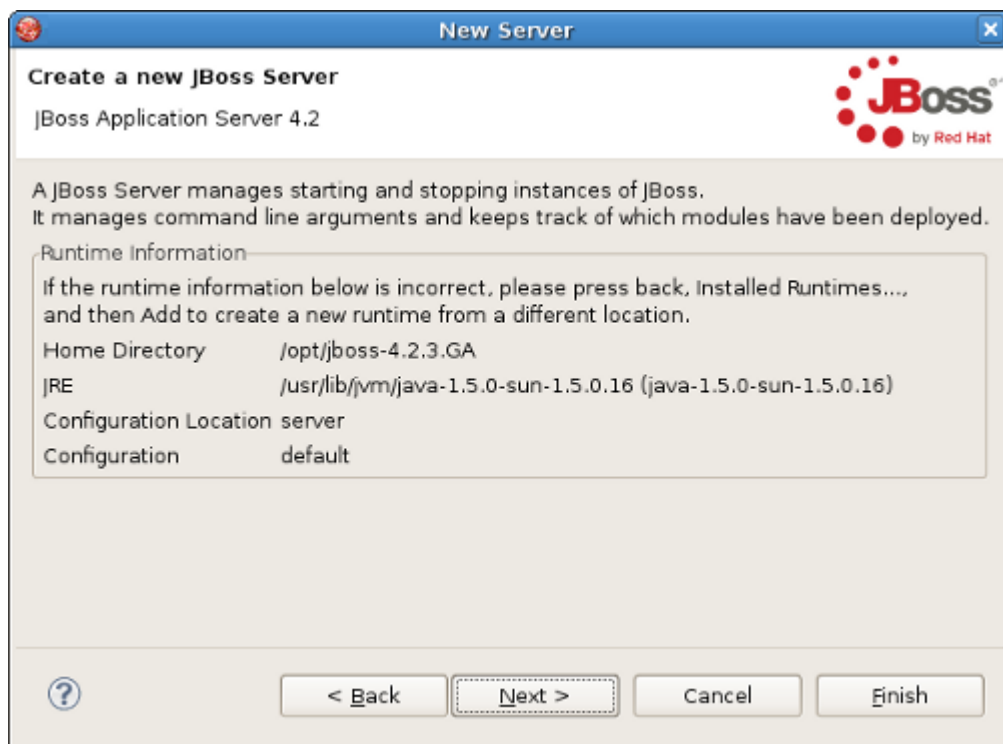


Figure 2.9. JBoss Runtime Summary

- In the last wizard's dialog modify the projects that are configured on the server and click [Finish](#).

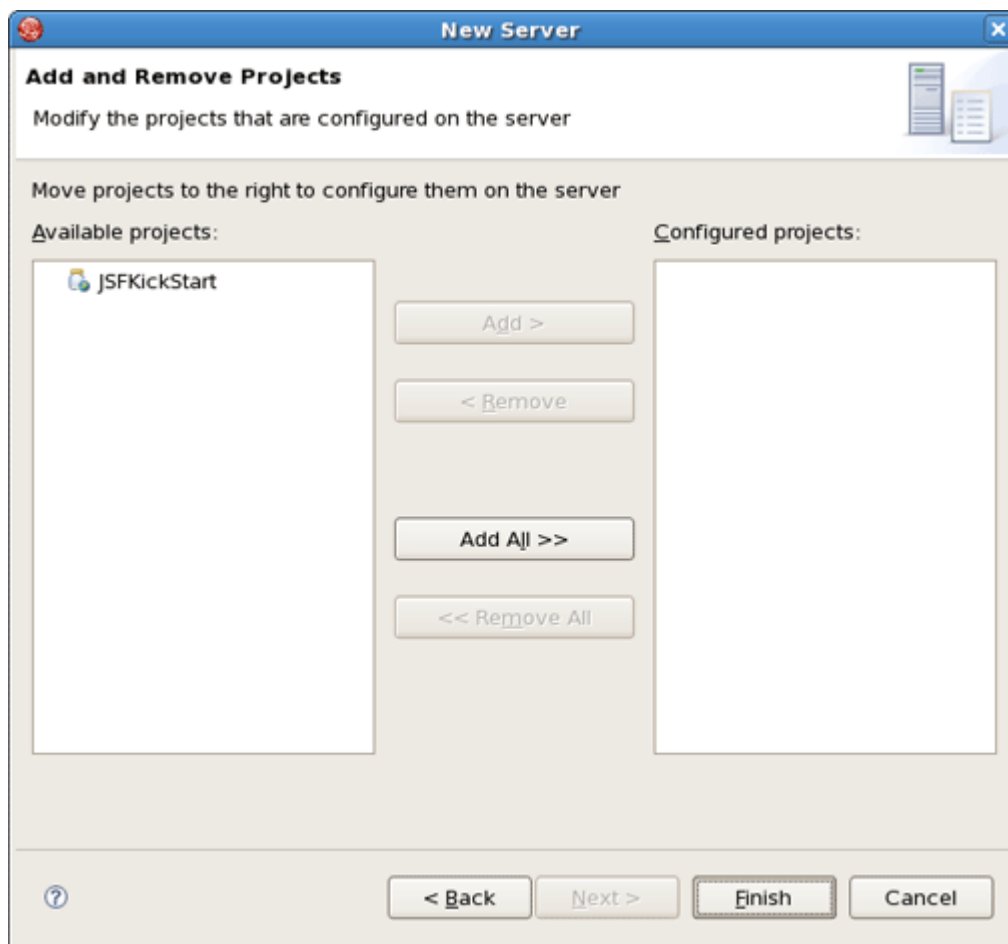


Figure 2.10. Configuring Projects

A new [JBoss Server](#) should now appear in the [Servers](#) view.

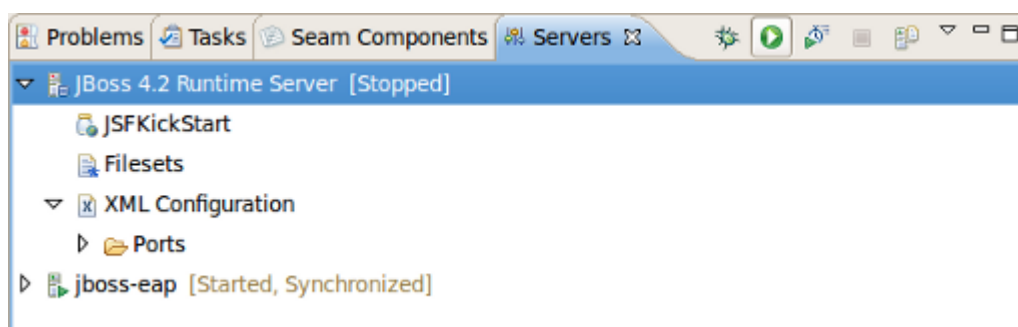


Figure 2.11. New JBoss Server

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 the JBoss Developer Studio can be used during the development process.

3.1. Create a Seam Application

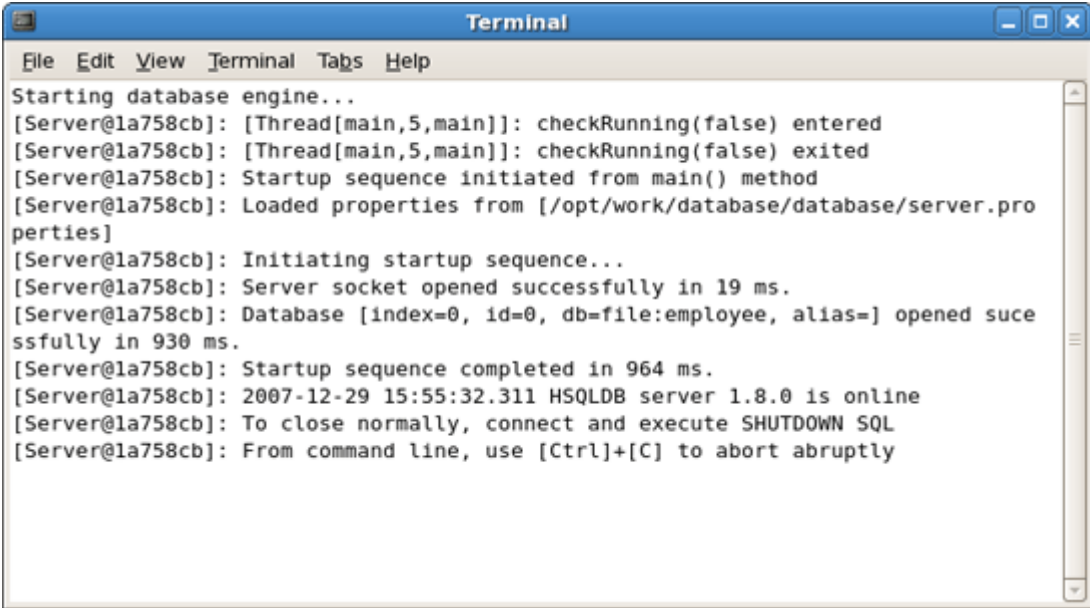
In this section you get to know how to create a Seam project in JBDS, how to start the server and what a structure your project has after creating.

3.1.1. Start Development Database

Before opening the JBoss Developer studio you need to download and start [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:



```
Terminal
File Edit View Terminal Tabs Help
Starting database engine...
[Server@1a758cb]: [Thread[main,5,main]]: checkRunning(false) entered
[Server@1a758cb]: [Thread[main,5,main]]: checkRunning(false) exited
[Server@1a758cb]: Startup sequence initiated from main() method
[Server@1a758cb]: Loaded properties from [/opt/work/database/database/server.properties]
[Server@1a758cb]: Initiating startup sequence...
[Server@1a758cb]: Server socket opened successfully in 19 ms.
[Server@1a758cb]: Database [index=0, id=0, db=file:employee, alias=] opened successfully in 930 ms.
[Server@1a758cb]: Startup sequence completed in 964 ms.
[Server@1a758cb]: 2007-12-29 15:55:32.311 HSQLDB server 1.8.0 is online
[Server@1a758cb]: To close normally, connect and execute SHUTDOWN SQL
[Server@1a758cb]: From command line, use [Ctrl]+[C] to abort abruptly
```

Figure 3.1. Starting DataBase

3.1.2. Create and deploy Seam Web Project

Minimize the terminal window and run the [JBoss Developer Studio](#) from Applications Menu or from the desktop icon.

First of all you get the Workspace Launcher. Change the default workspace location if it's needed. Click on [Ok](#).

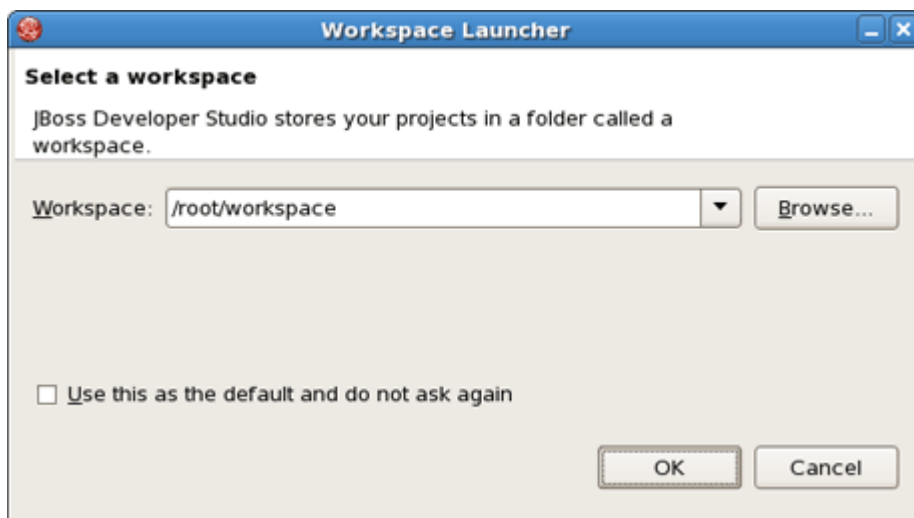


Figure 3.2. Workspace Launcher Dialog

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 directory 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 needed datasource and driver libraries need to go.



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 facets pattern:

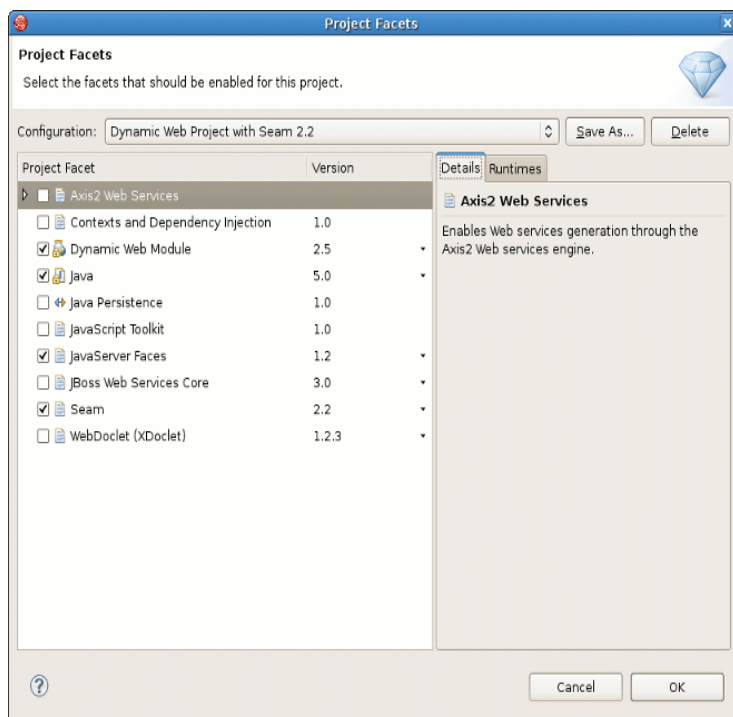


Figure 3.4. Project Facets Specifying

On the whole the dialog allows to select the "features" you want to use in your project. Doing this [JBoss Developer Studio](#) setups 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.

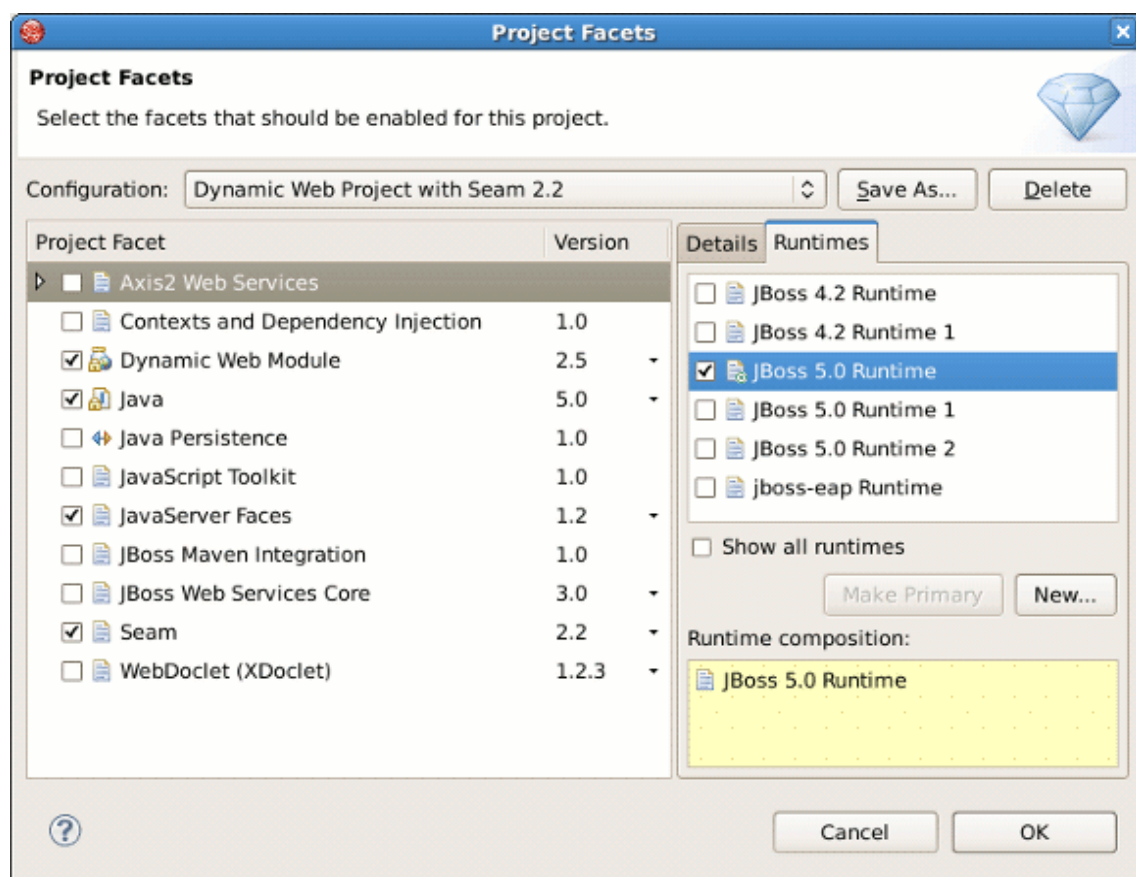


Figure 3.5. Runtimes Selecting

Click on [Ok](#) and then [Next](#) to proceed further.

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

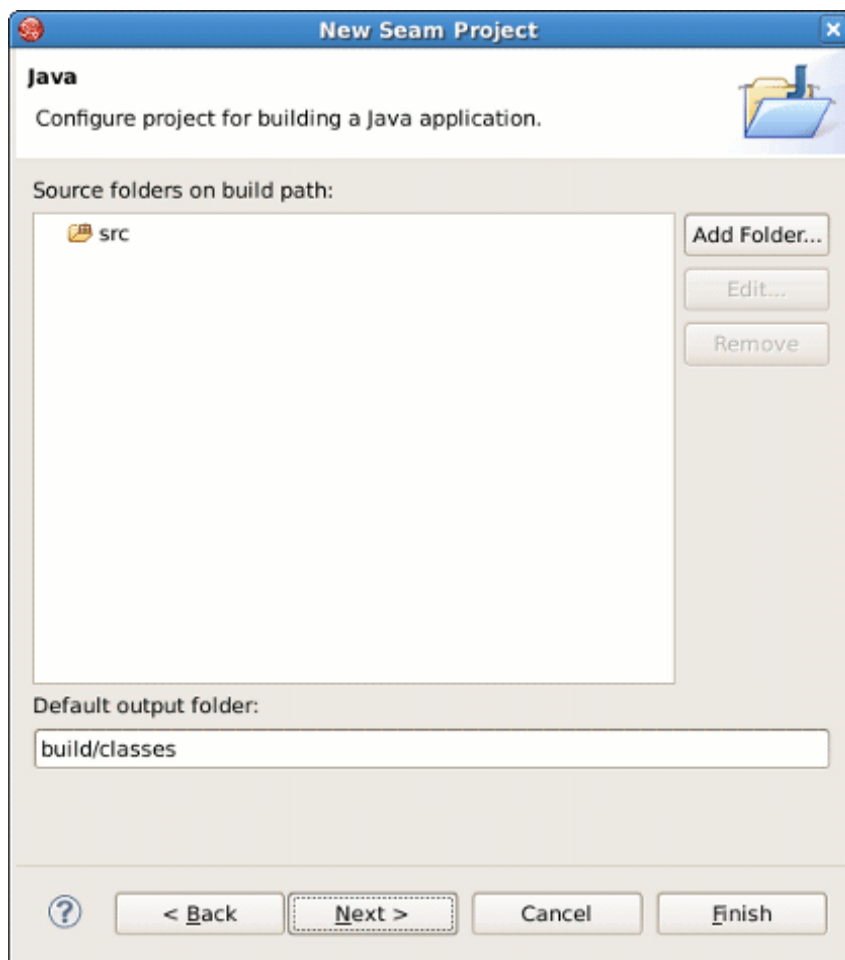


Figure 3.6. Java Build Path

Following page provides you [Web Module Settings](#) .You can just leave the default values or choose another folder.

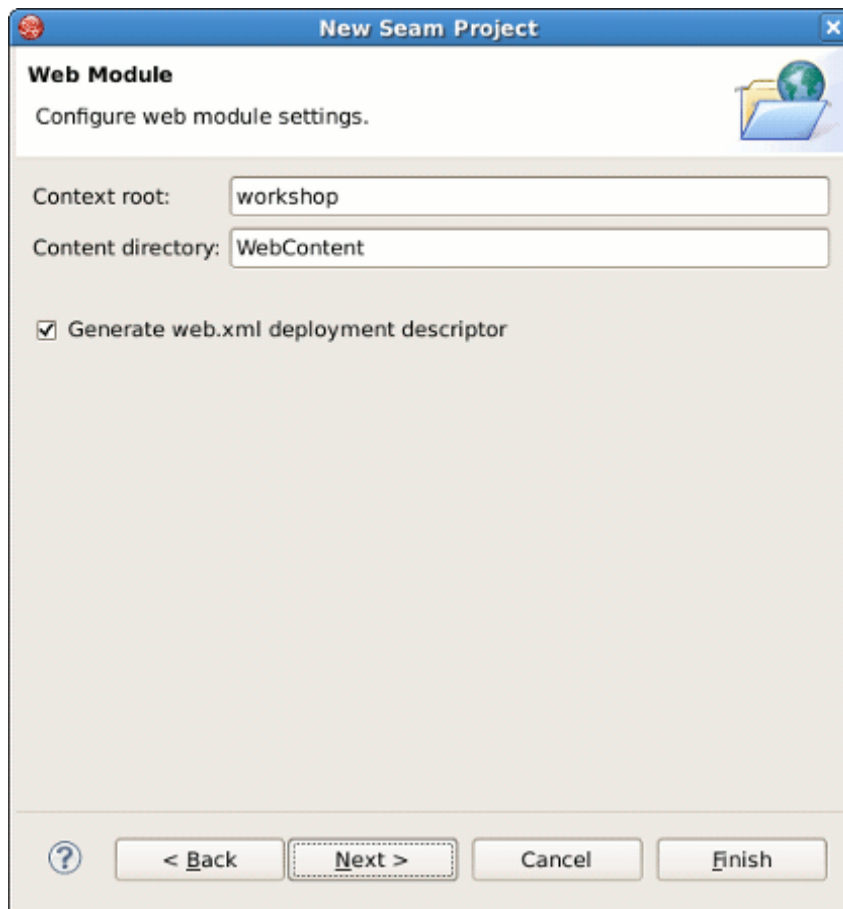


Figure 3.7. Web Module Settings

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 [Next](#)

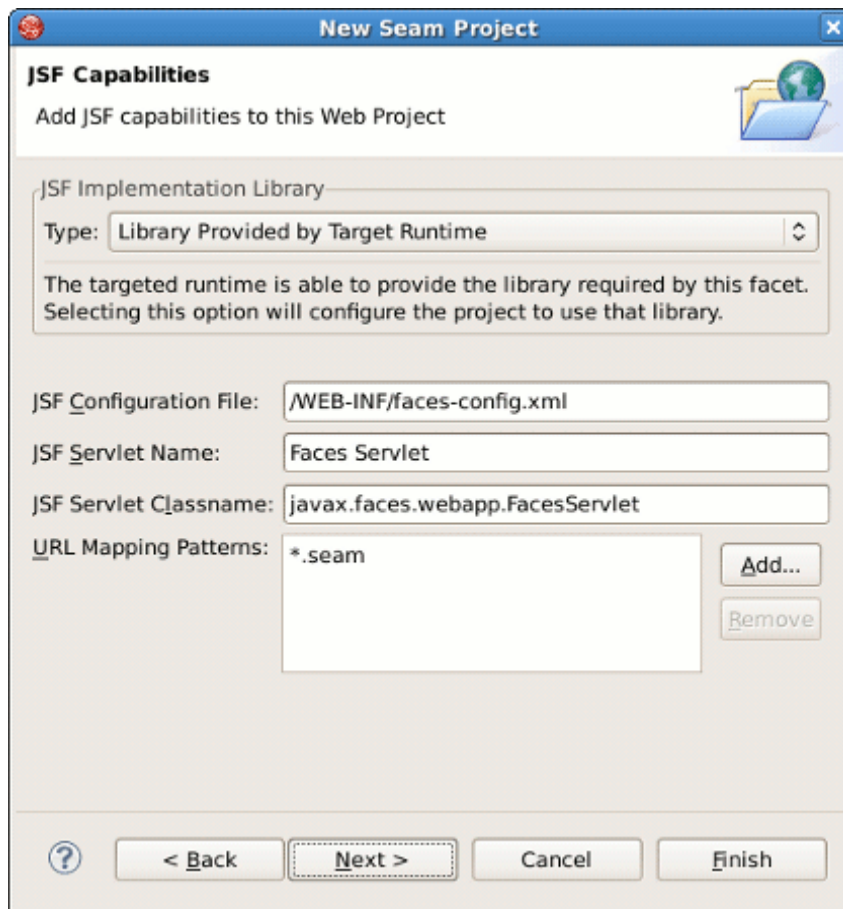


Figure 3.8. JSF Capabilities Adding

Next wizard step needs more settings than previous. Let's start with [General](#) section.

Leave the default Seam runtime and check a WAR deployment.

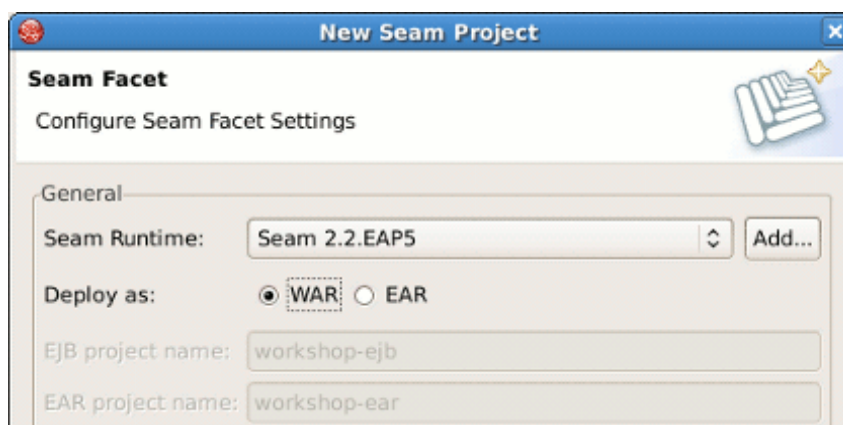


Figure 3.9. Seam Facet Setting

Next [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 [Edit](#) to modify the Connection Profile.

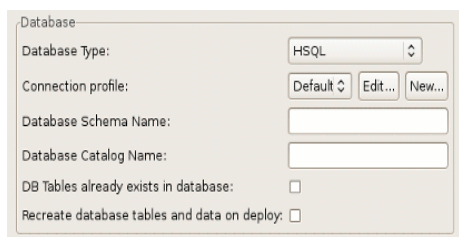


Figure 3.10. DataBase Setting

Select [HSQLDB Profile Properties](#). Make sure the Database location is set to `hsq://localhost:1701`

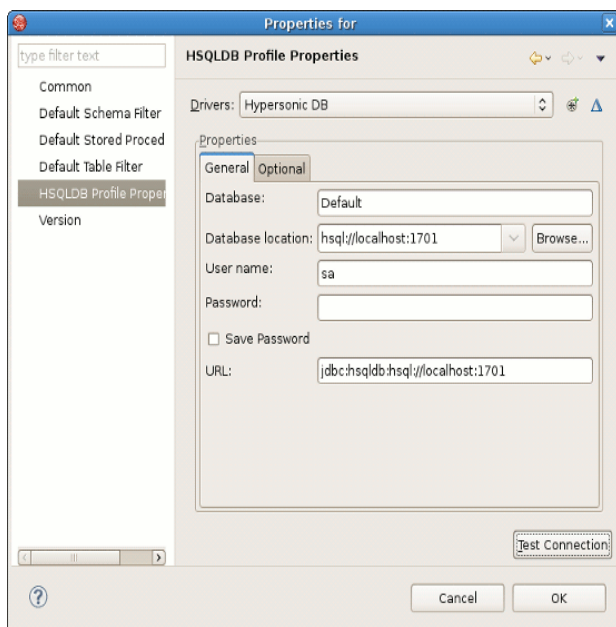


Figure 3.11. JDBC Connection Properties

Try click on [Test Connection](#) button. It probably won't work. This happens if the hsql jdbc driver is not exactly the same. This can be worked around by modifying the HSQLDB database driver settings. To modify the settings, click the Edit Driver Definition Driver() .

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 [Ok](#).

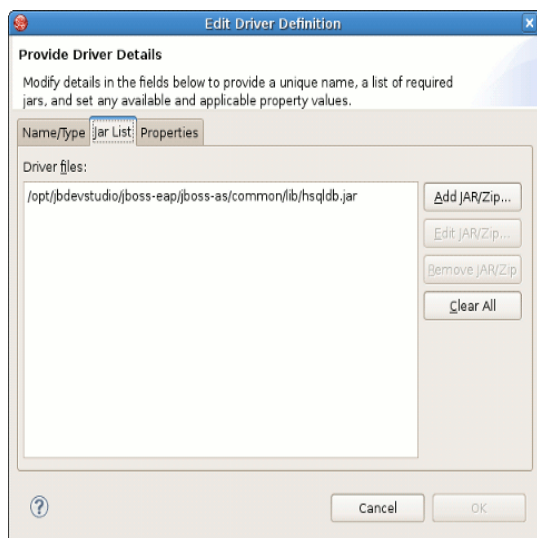


Figure 3.12. Driver Details

Now, the [Test Connection](#) should succeed. After testing the connection, click [Ok](#).

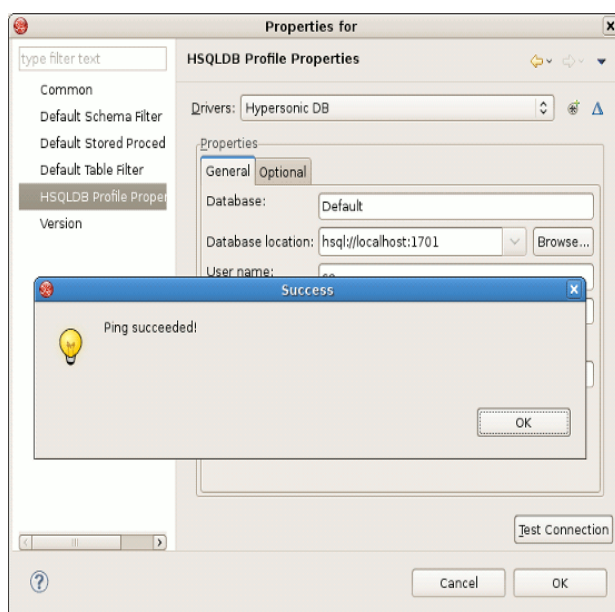
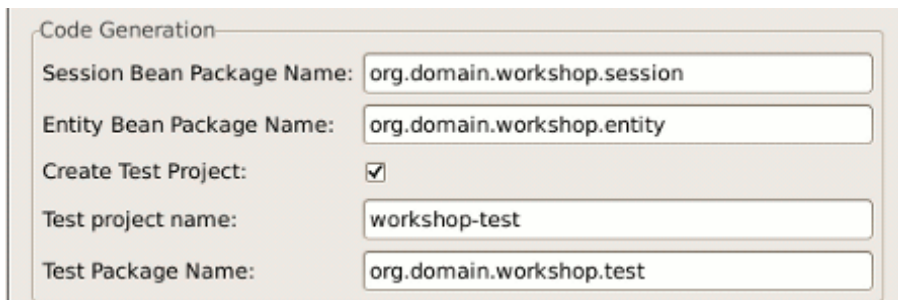


Figure 3.13. Connection Testing

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



The image shows a 'Code Generation' dialog box with the following fields and values:

Field	Value
Session Bean Package Name:	org.domain.workshop.session
Entity Bean Package Name:	org.domain.workshop.entity
Create Test Project:	<input checked="" type="checkbox"/>
Test project name:	workshop-test
Test Package Name:	org.domain.workshop.test

Figure 3.14. Code Generation Setting



Tip:

If you want to name your web project "MyProject-war" note that the Test project name should not be "MyProject-war-test", it should be "MyProject-test".

Click on [Finish](#) button. Now, there should be a new Seam project called "workshop" listed in Package Explorer view.

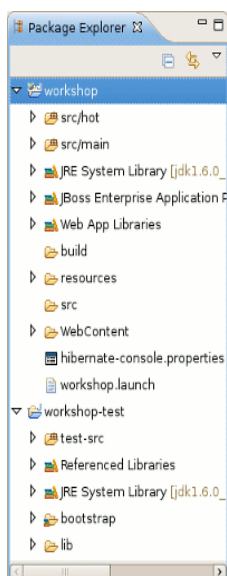



Figure 3.15. "workshop" Project in the Package Explorer

3.1.3. Start JBoss Application Server

The complete information on how to manage JBoss AS from JBoss Developer Studio you can read in a [corresponding chapter](#).

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

Then run the project by selecting the project and use [Run As... > 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:

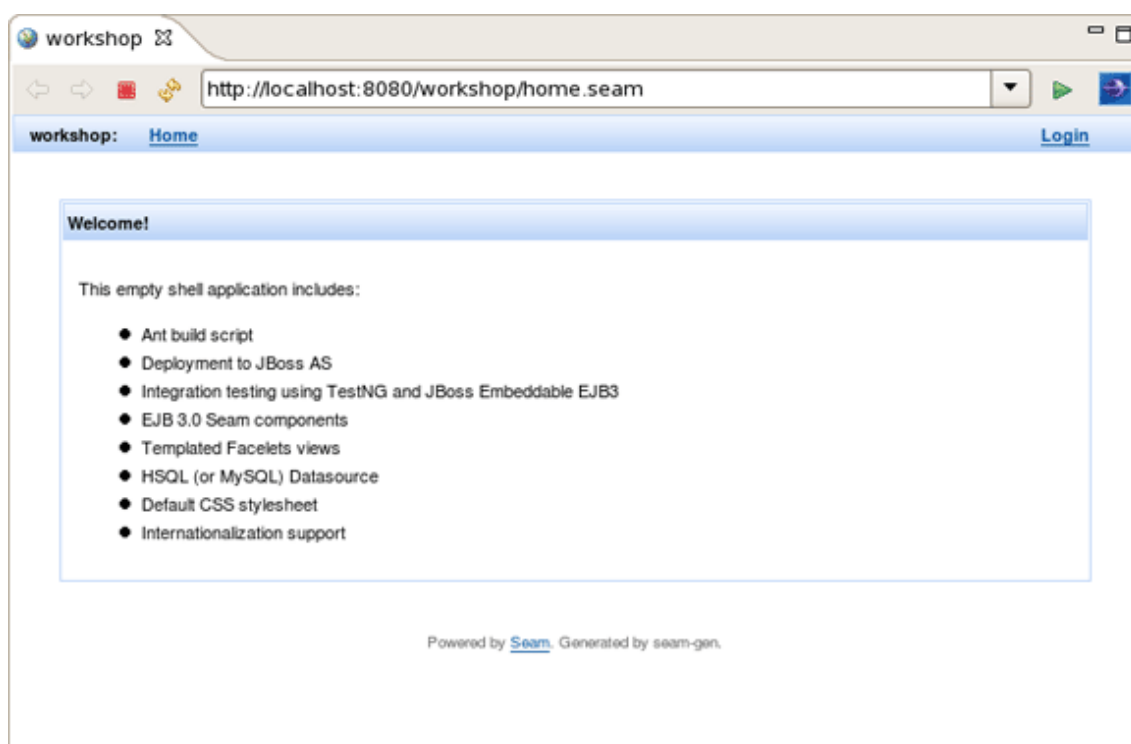


Figure 3.16. "workshop" Project Started

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 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 a bit more detail later on. The [src/main](#) folder is a model directory. It stores the project's JPA entity beans.

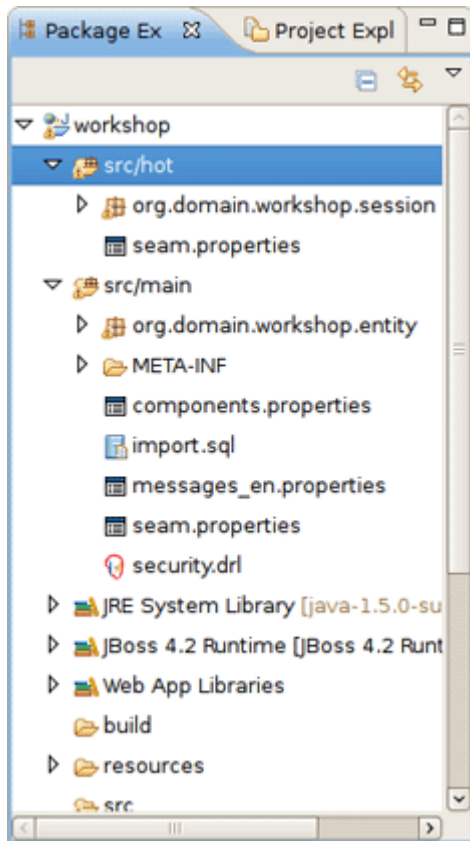


Figure 3.17. Project Structure

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

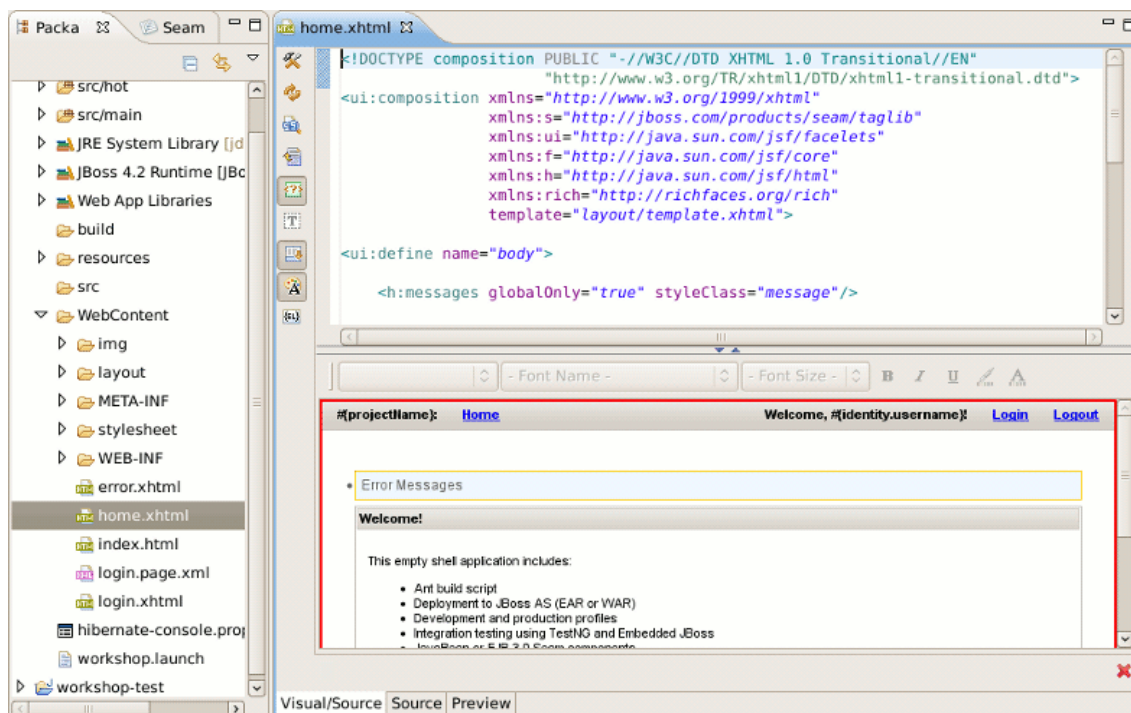


Figure 3.18. Facelets GUI Editor

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 per the Seam project wizard database settings. And, obviously all of the Seam specific configuration files and JAR dependencies are included and placed in the 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/deployment process. The end result is a developer that is writing code, not spending days/weeks 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 [Finish](#).

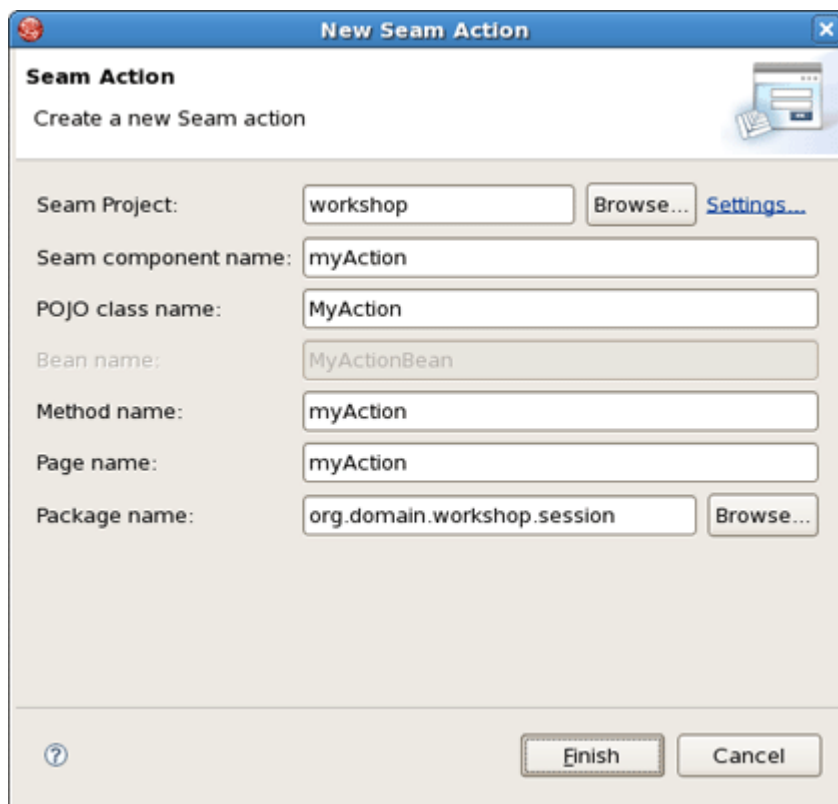


Figure 3.19. New Seam Action Wizard

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");  
    facesMessages.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.

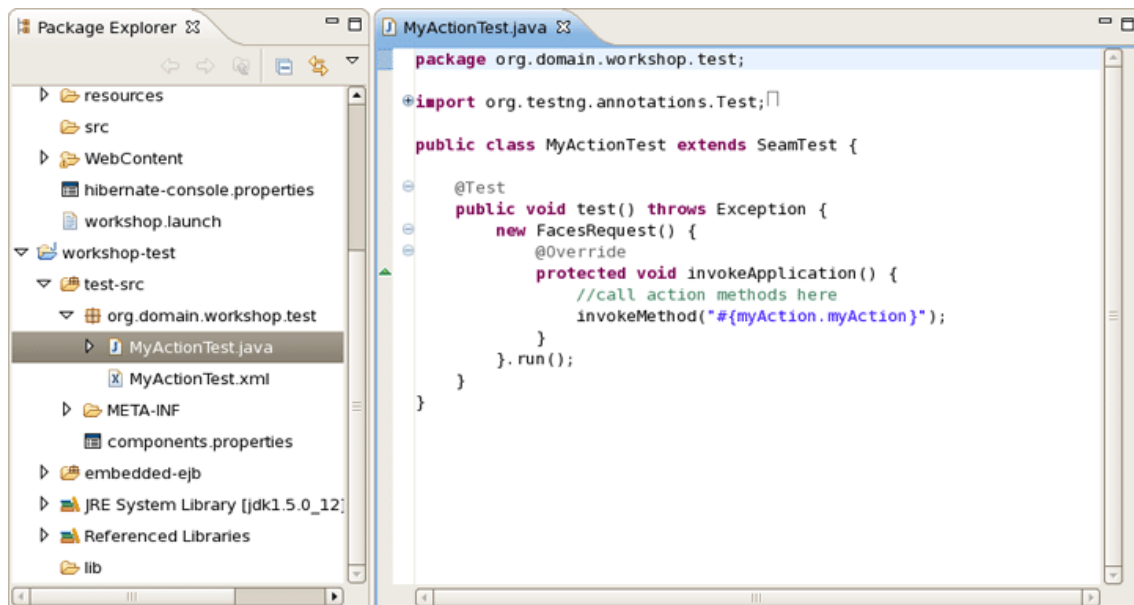


Figure 3.20. "workshop-test" Project

The test case simulates a Seam component/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.

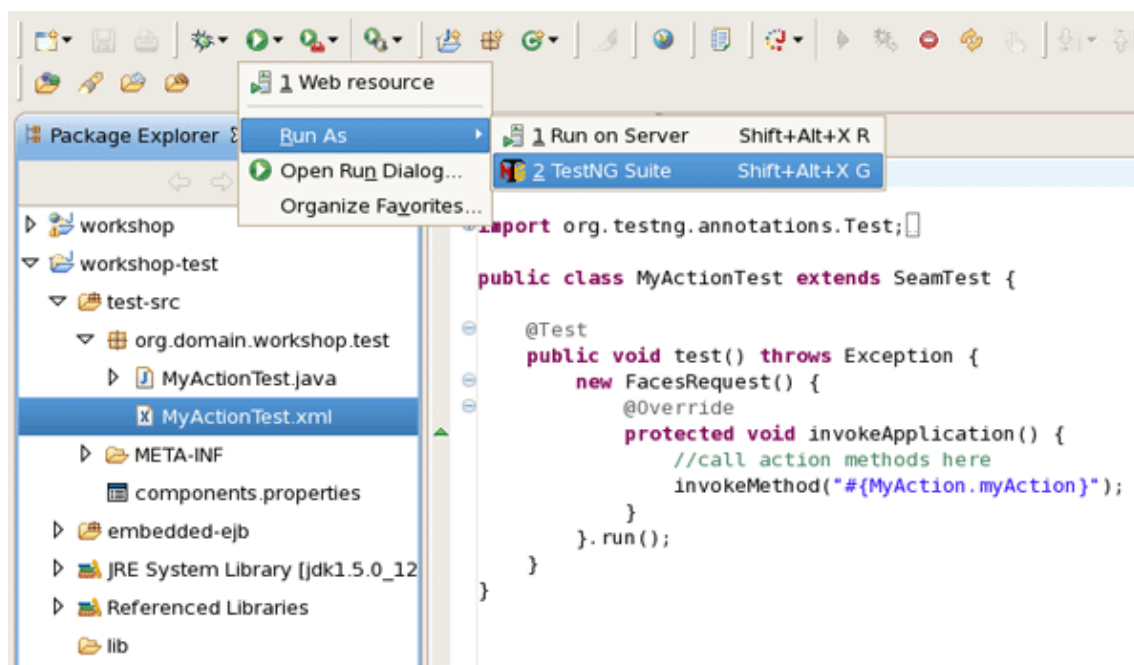


Figure 3.21. TestNG Running

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

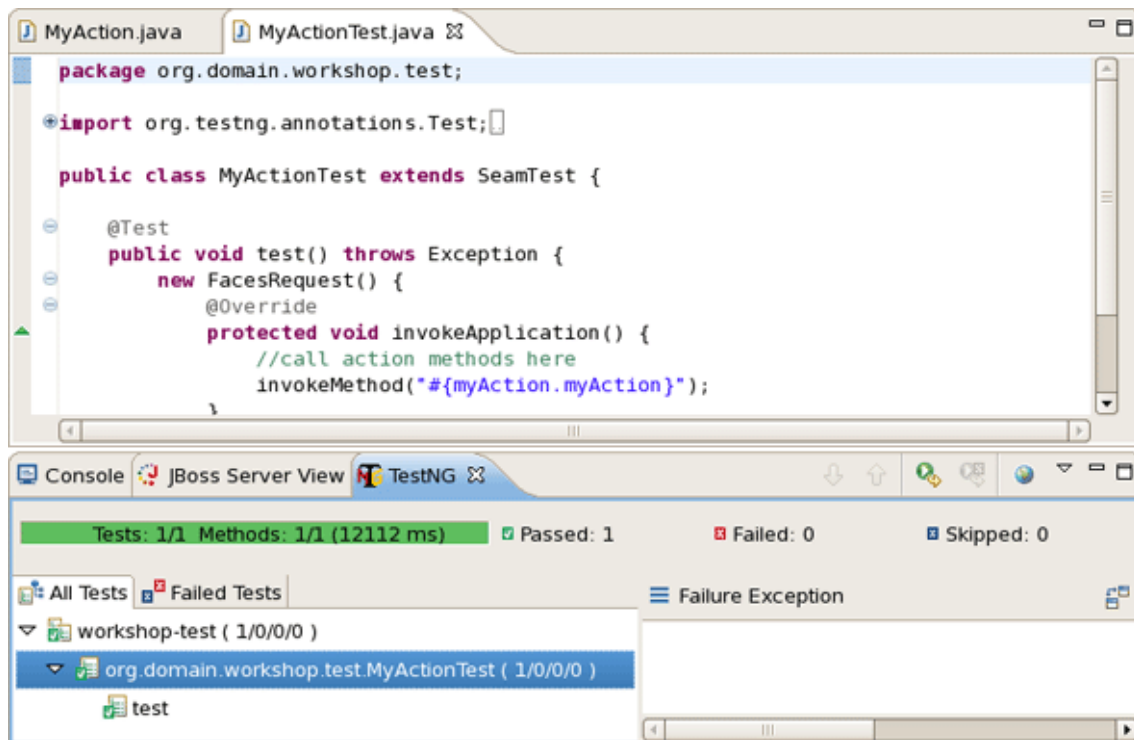


Figure 3.22. TestNG Results

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.

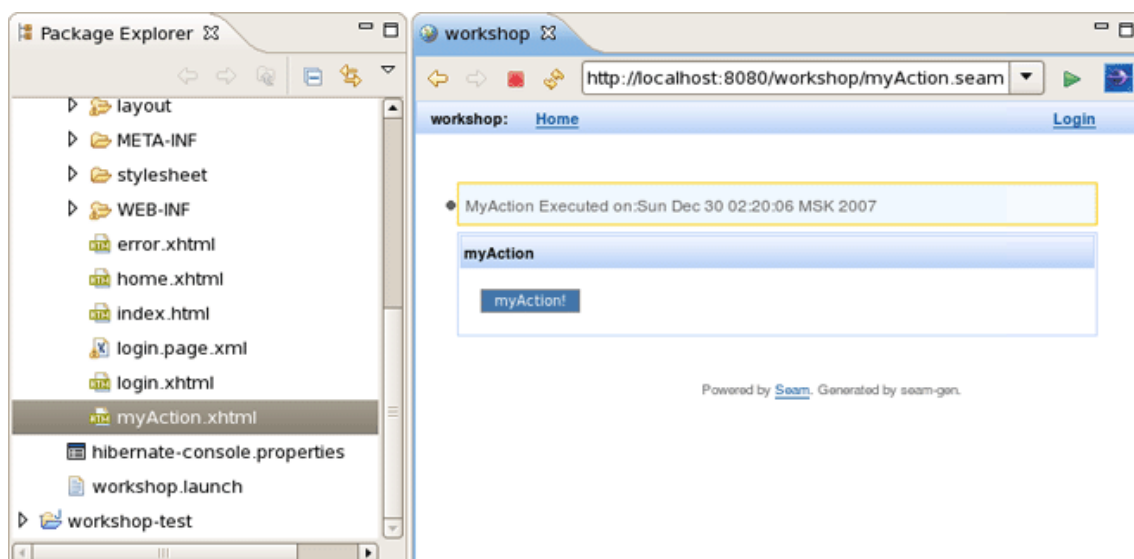


Figure 3.23. Seam Action in a Web Browser

3.2.3. Modify Seam Action User Interface

Browse to <http://localhost:8080/workshop/myAction.seam> and click on [myAction](#). 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.

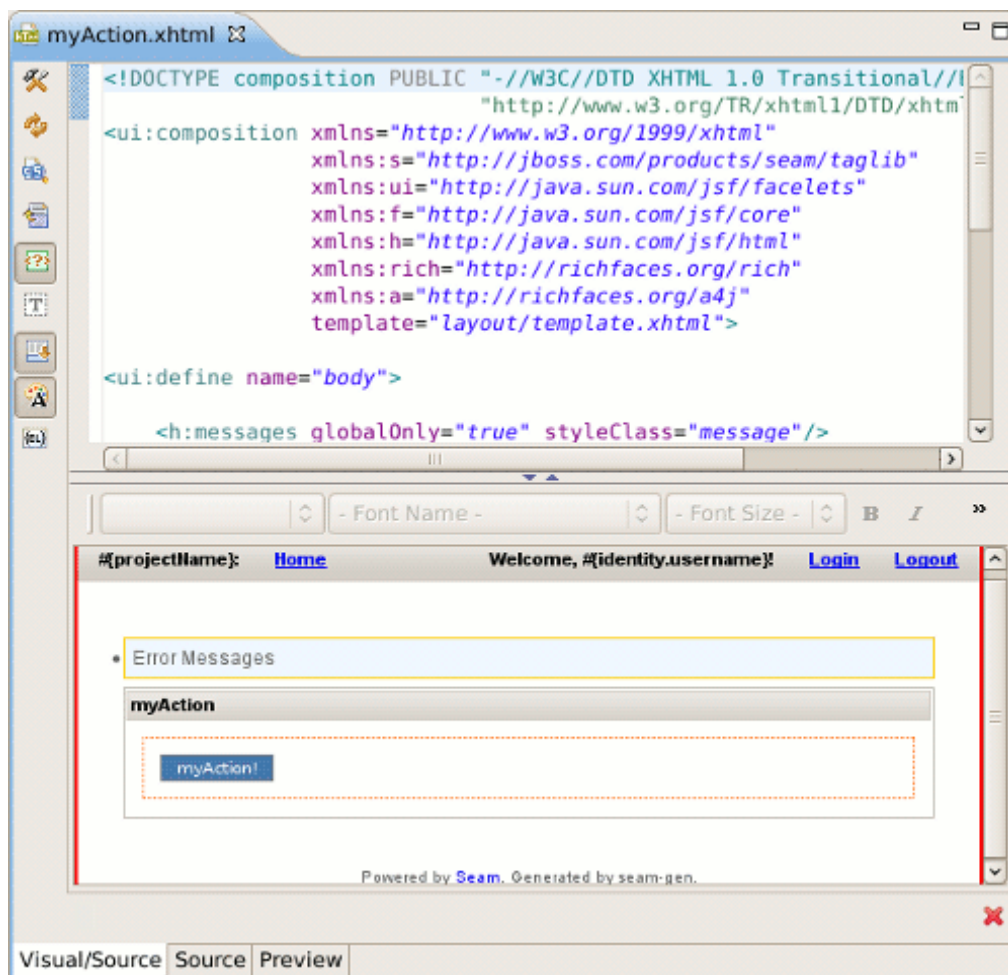


Figure 3.24. Open Seam Action with Editor

Right click on the "myAction!" button in the visual part of editor and select [<h:commandButton> Attributes](#).

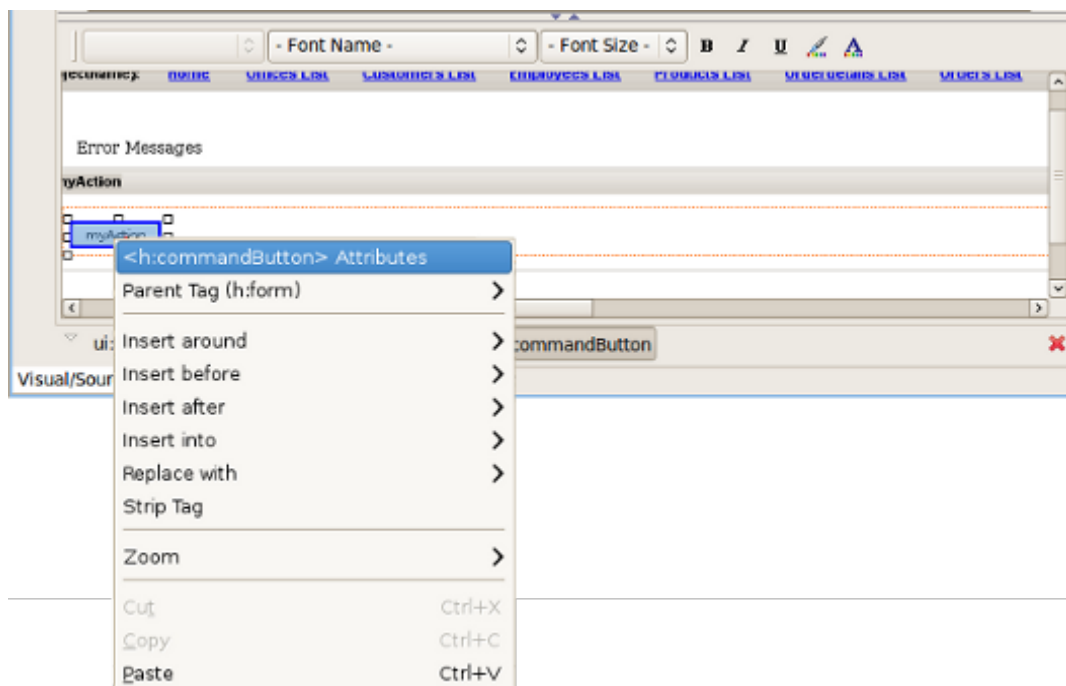


Figure 3.25. Seam Action Editing

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.

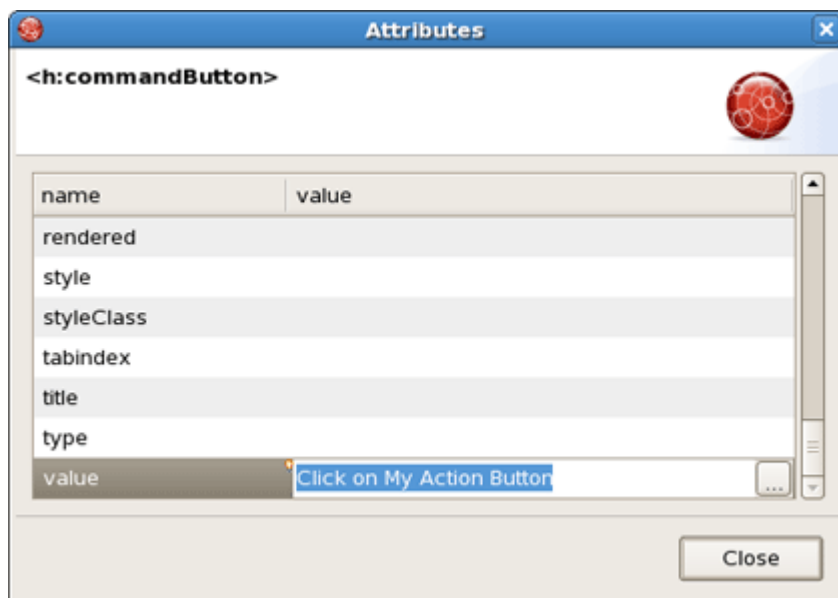


Figure 3.26. Attributes Dialog

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.

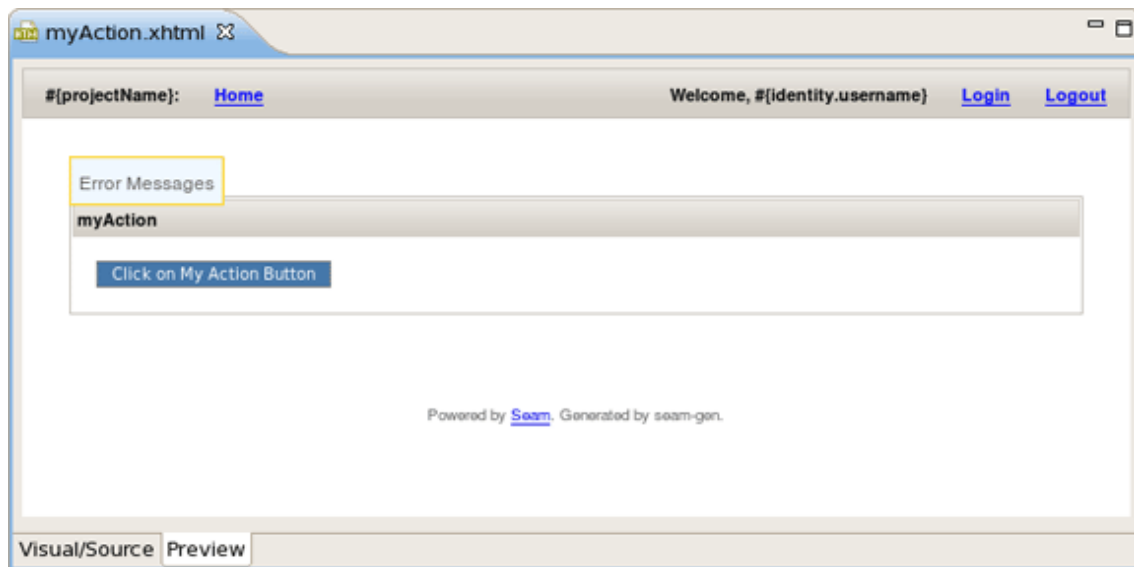


Figure 3.27. Seam Action Is Modified

3.3. Declarative Security

In this section you will see how it's easy 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 adding some custom login logic.

Open `Authenticator.java` in JBoss Developer Studio and replace the `authenticate()` method with this code:

```
public boolean authenticate()
{
    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.

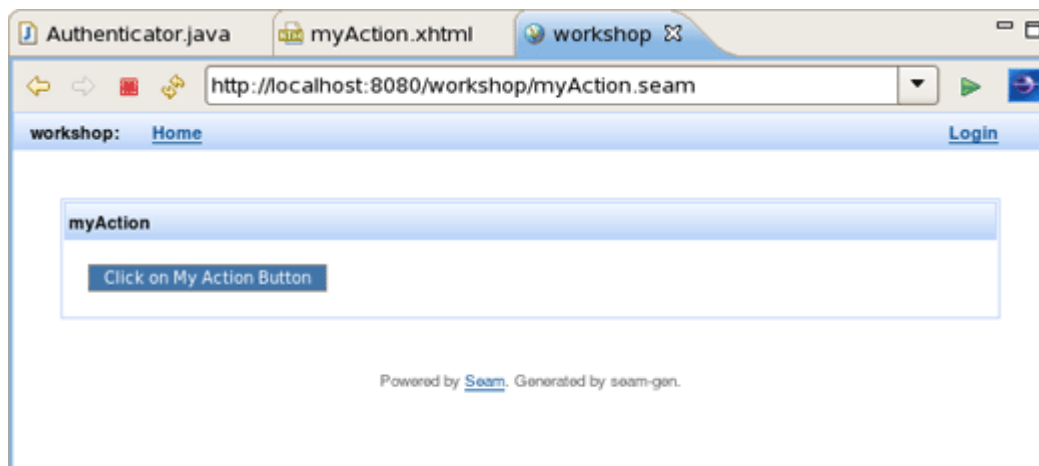


Figure 3.28. One Button on a Page

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

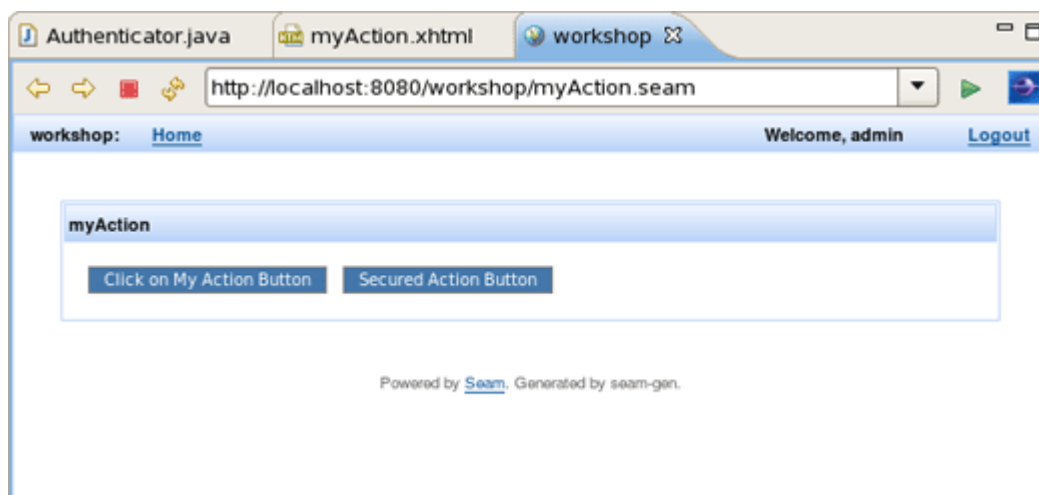


Figure 3.29. Secured Button is Visible

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.

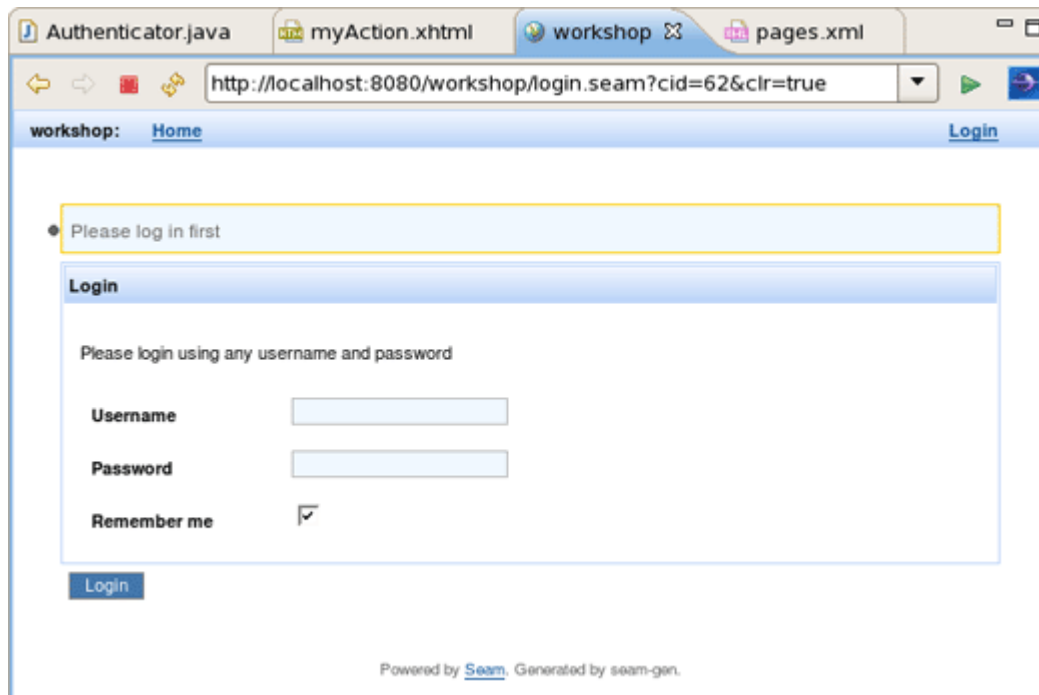


Figure 3.30. Login Page

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 a [Databases](#) node and select a [Default](#) database. Right click on it, select [Connect](#) from the context menu.

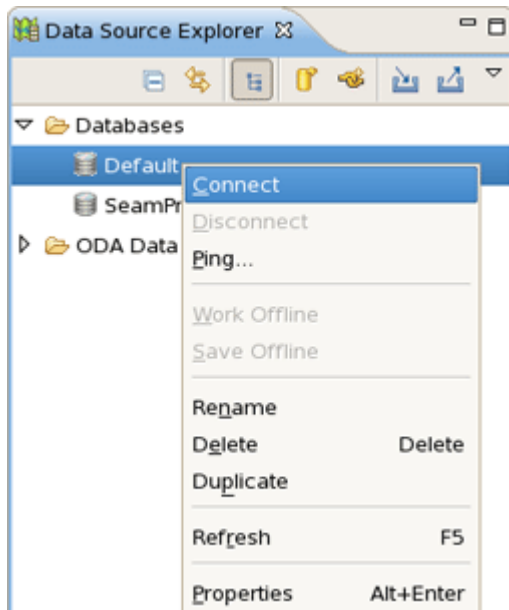


Figure 3.31. Data Source Explorer

3.4.2. Browse Workshop Database

Then in the current view, drill down to the [CUSTOMERS](#) table.

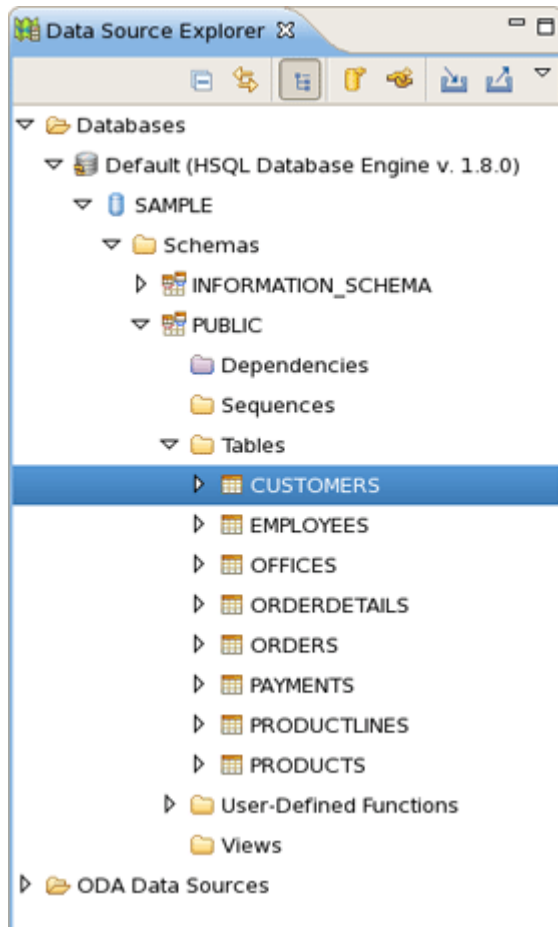


Figure 3.32. "CUSTOMERS" Table

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.

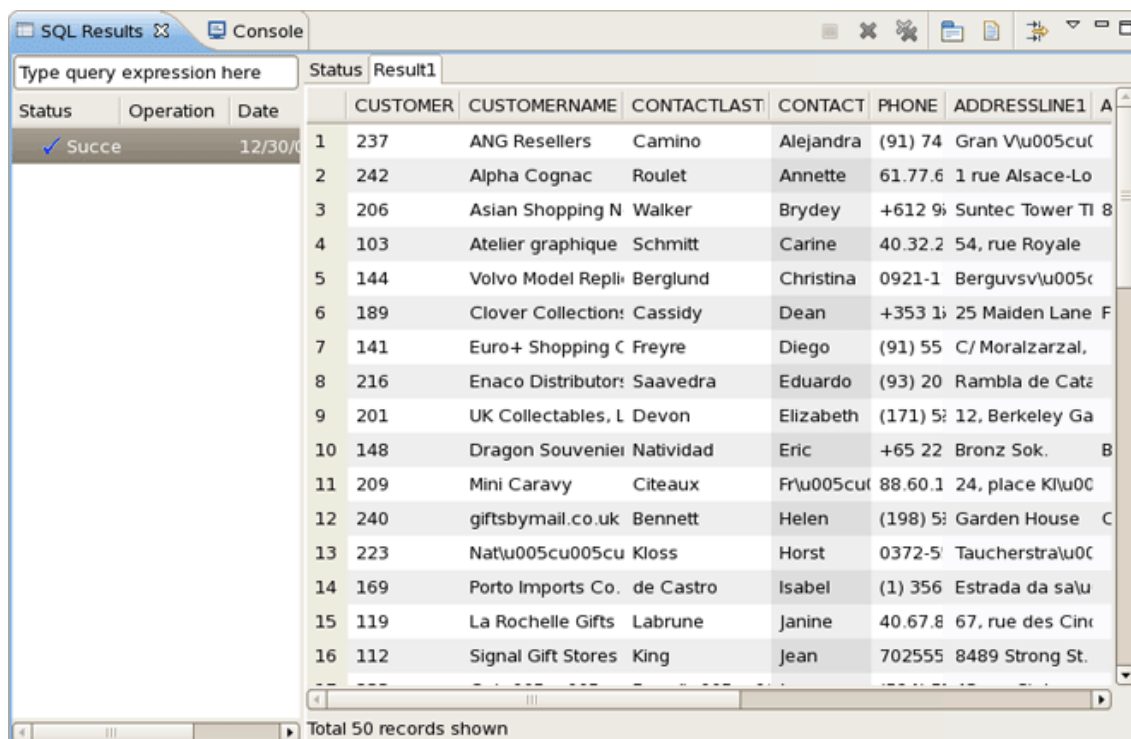


Figure 3.33. SQL Results View



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 **Seam perspective**, and then right-click the project and select **New > Seam Generate Entities**.

"workshop" project in the [Seam Generate Entities wizard](#) will be selected automatically. There is no need to change something more, click [Next](#) to proceed further.

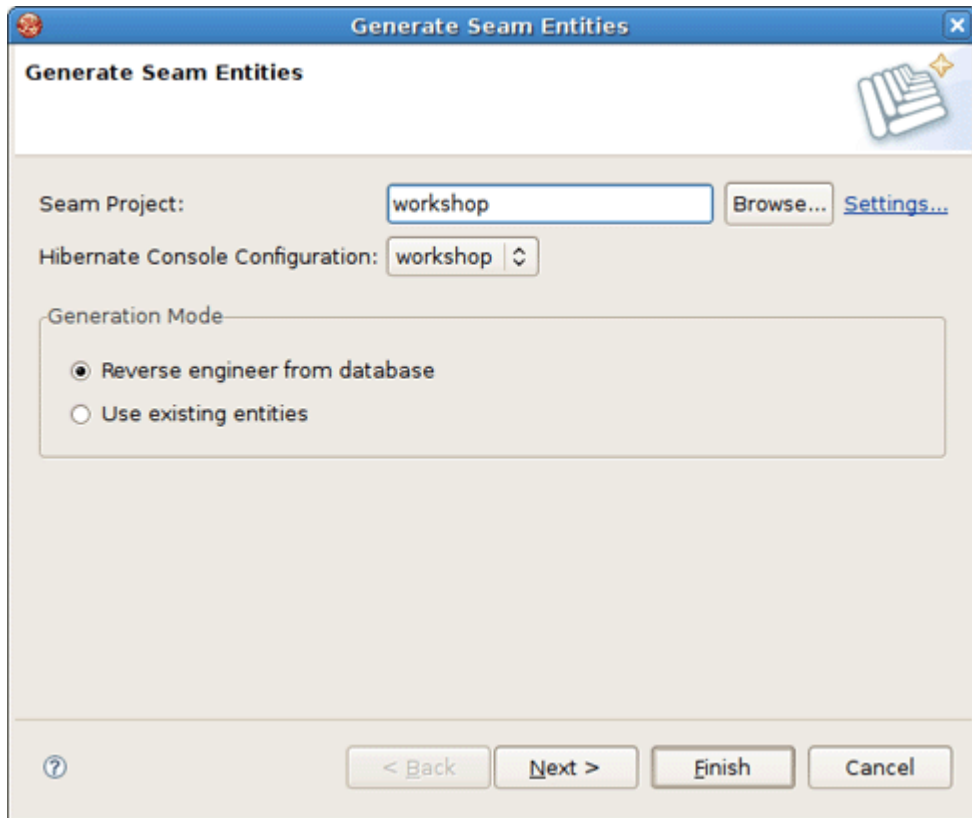


Figure 3.34. Generate Seam Entities

On the next page use the [Include](#) button to include all the tables from the database and click [Finish](#).

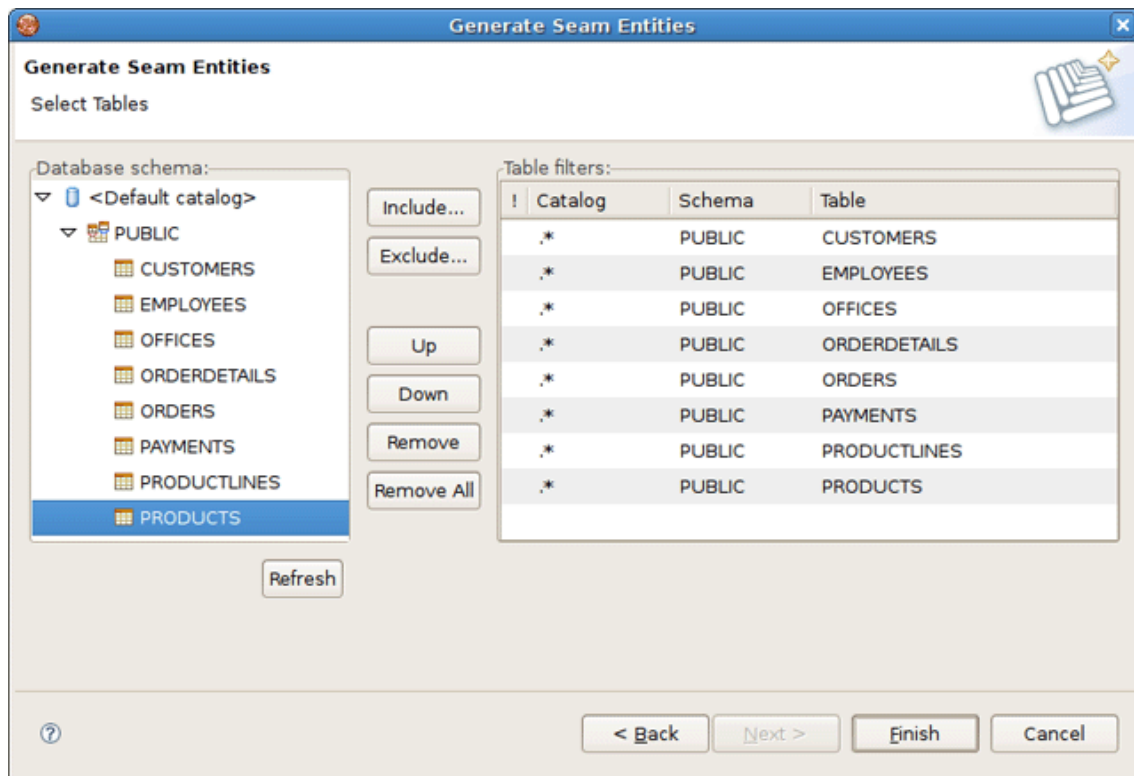


Figure 3.35. Selecting Tables

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

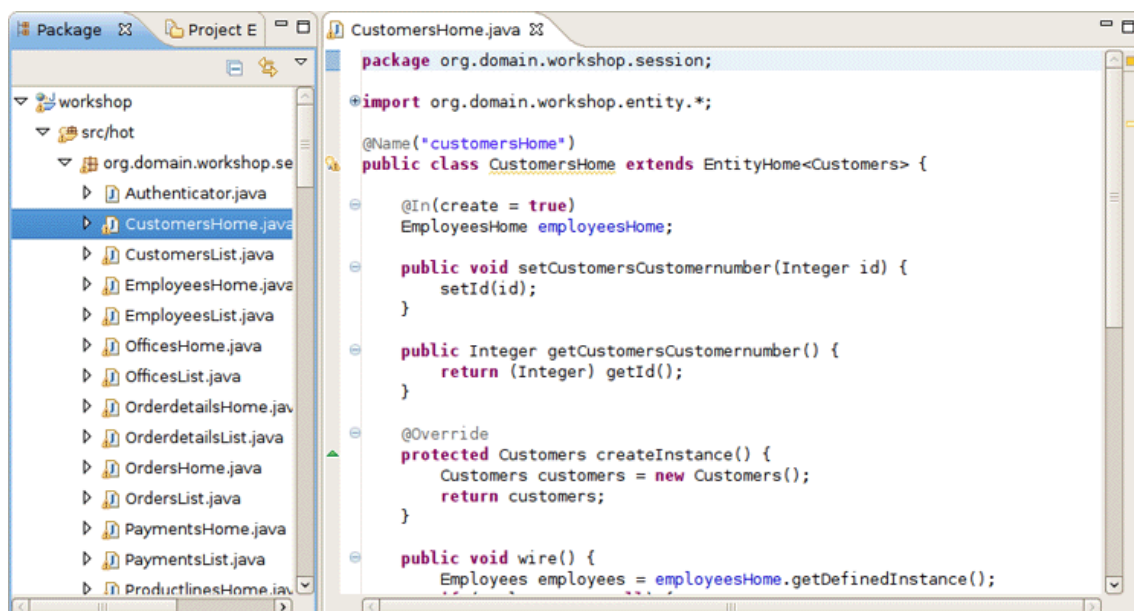


Figure 3.36. org.domain.workshop.entity Classes

There are also [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 which we will explore further later on in the lab. For now, take note of the page tabs, required field logic and data table sorting in the list pages.

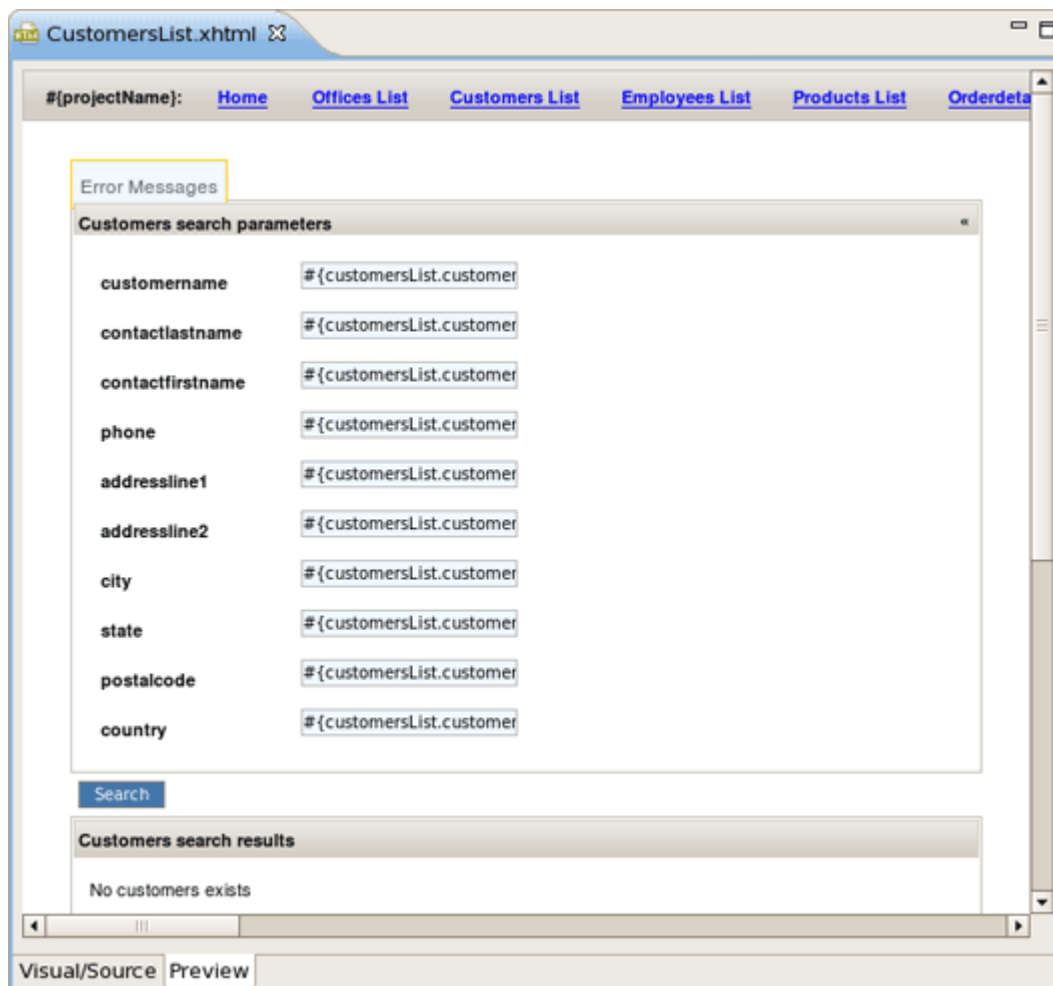


Figure 3.37. CustomersList.xhtml in the Editor

Congratulations! You now have a fully functioning CRUD application that is already 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 choose [Hibernate](#).

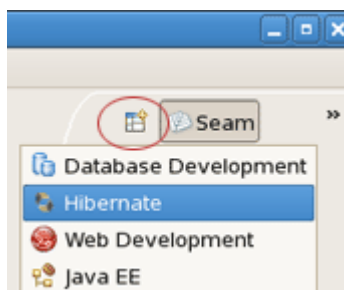


Figure 3.38. Hibernate Perspective

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.

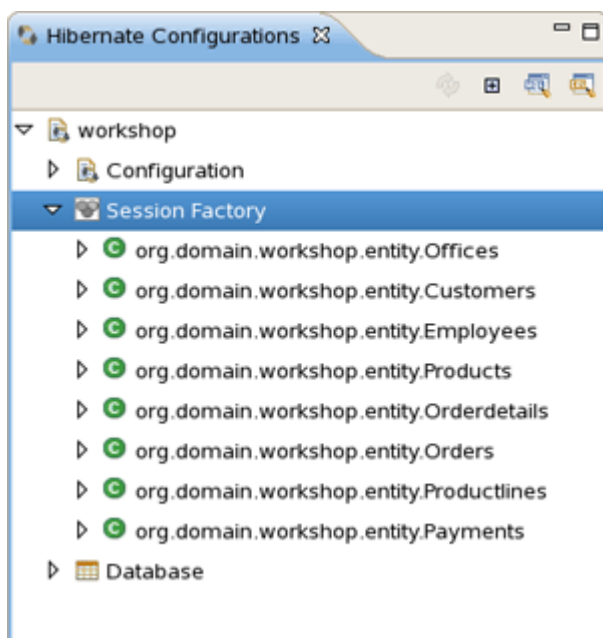


Figure 3.39. Hibernate Configurations View

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.

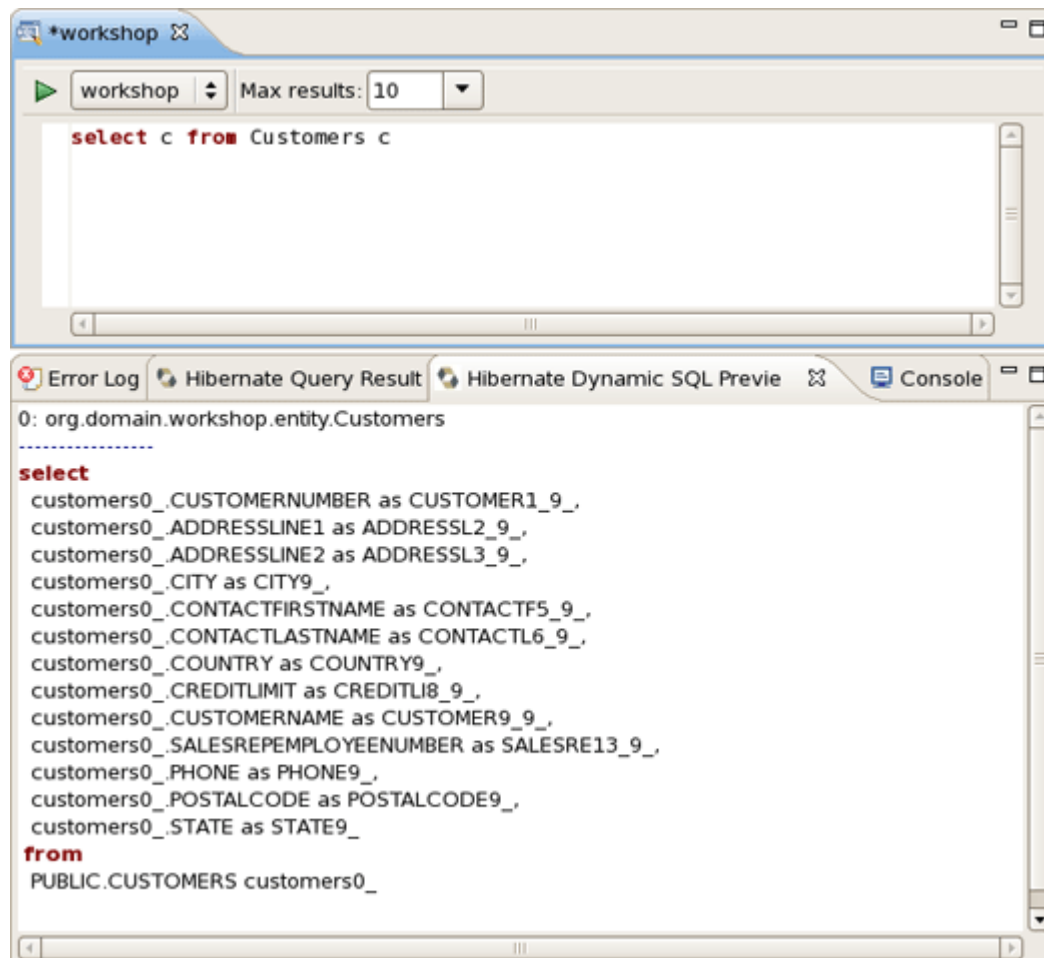


Figure 3.40. JPA Query Editor

Run the query by clicking on the green run icon.

The results are listed in the "Hibernate Query Result" tab. 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.

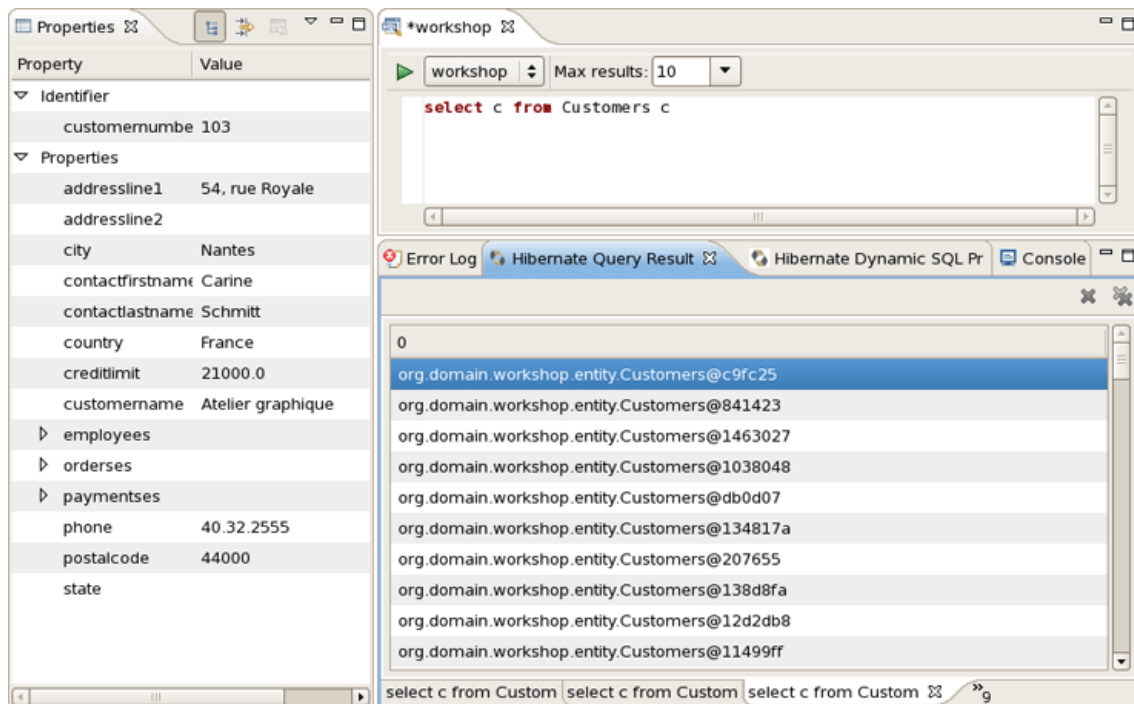


Figure 3.41. Hibernate Query Result View

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

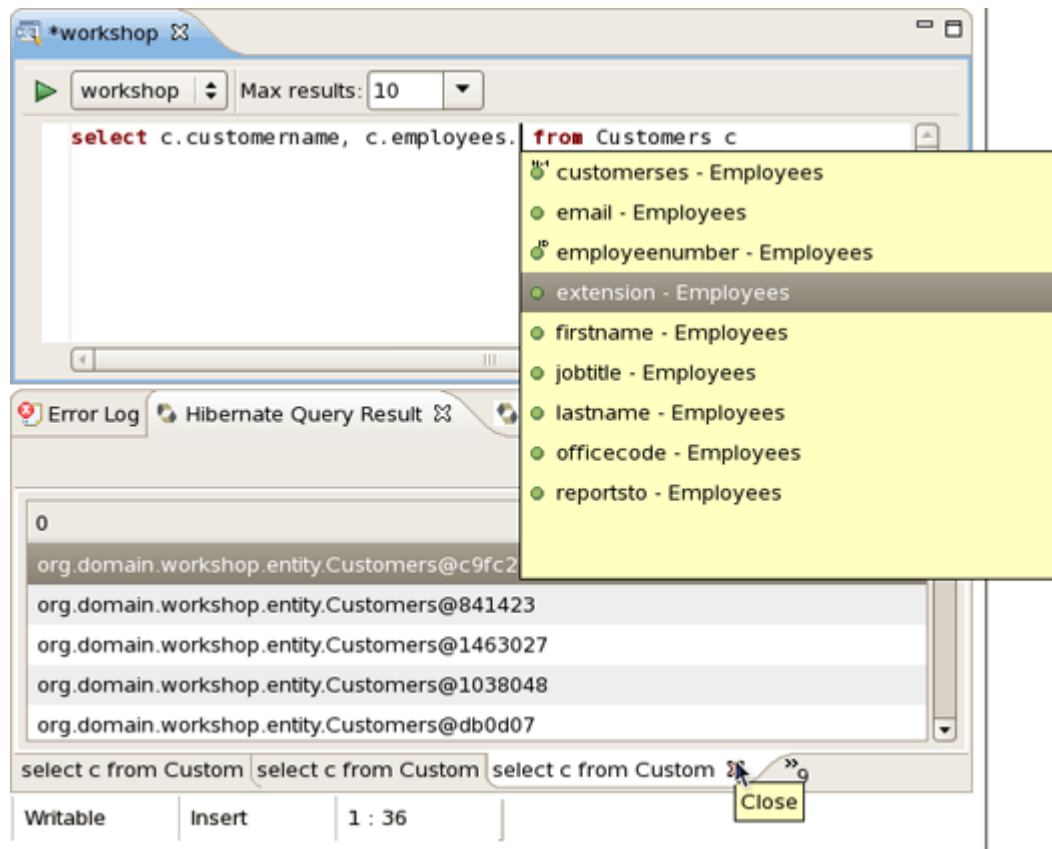


Figure 3.42. Code Completion

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

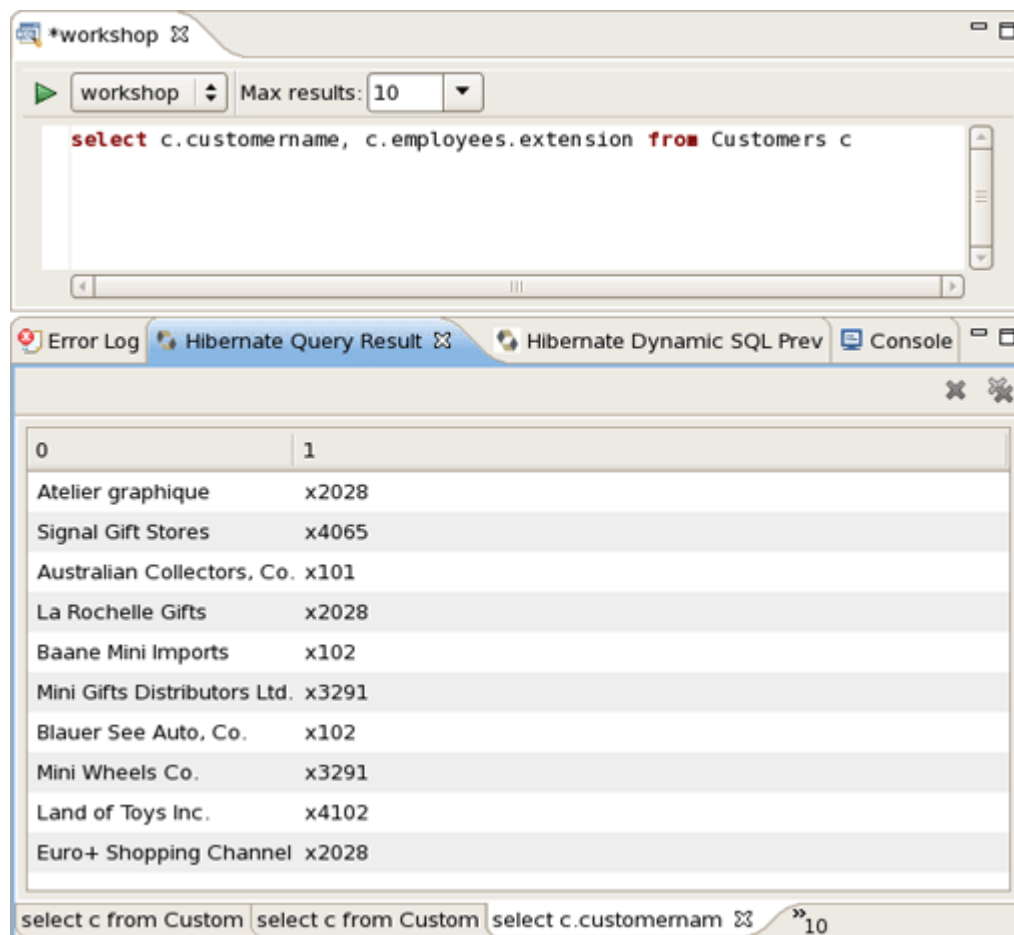


Figure 3.43. The Hibernate Query Result

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.

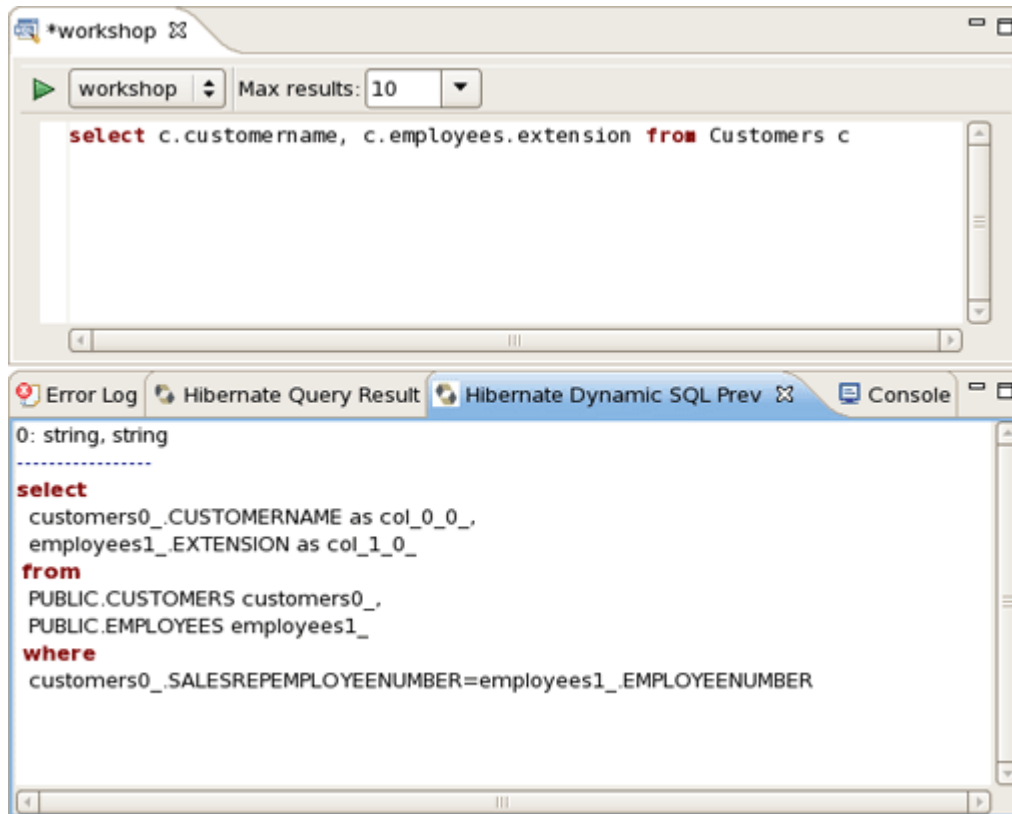


Figure 3.44. Criteria Editor

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 [Open Mapping Diagram](#).

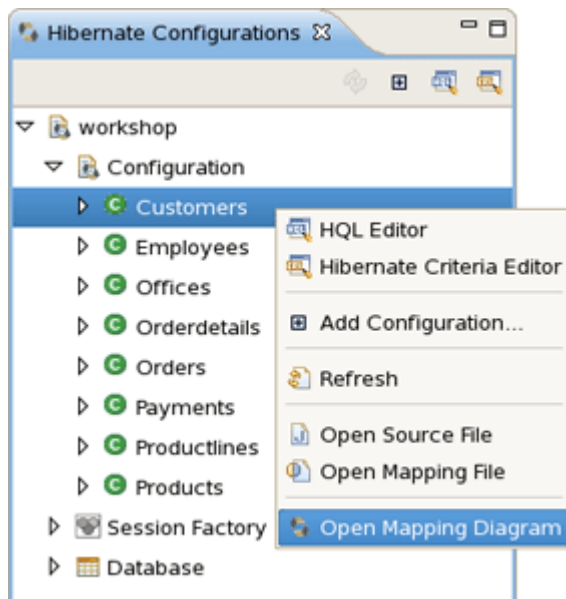


Figure 3.45. Mapping Diagram Opening

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.

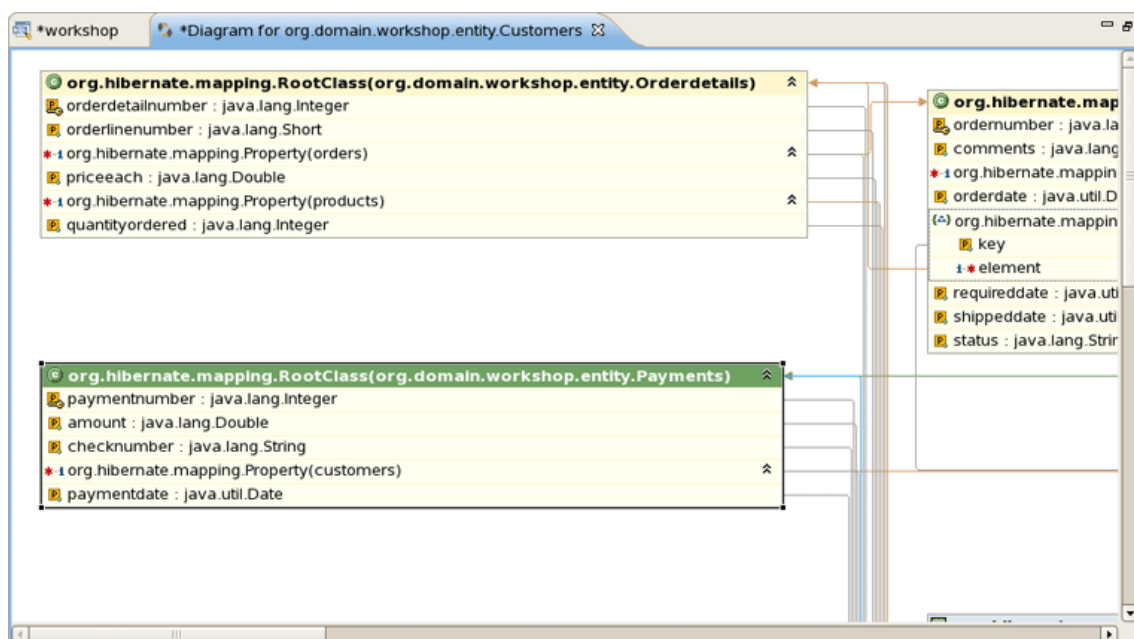


Figure 3.46. Diagram Tab

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, 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.

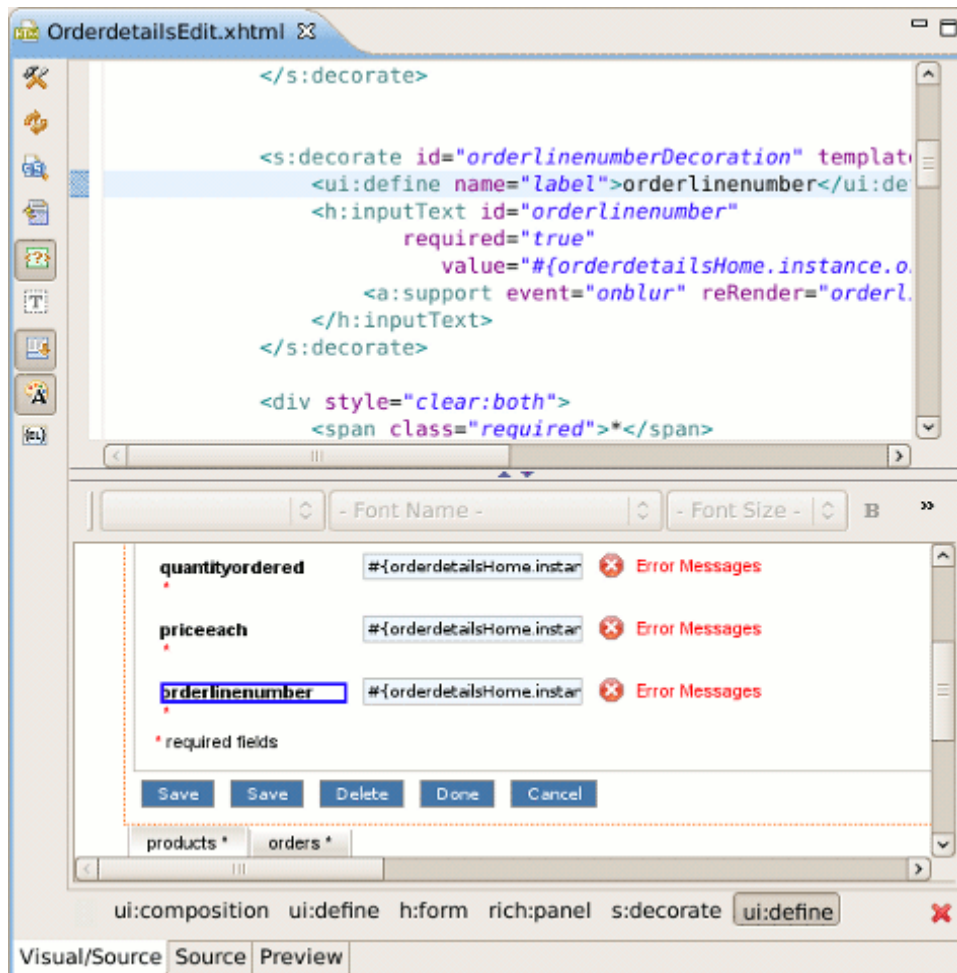


Figure 3.47. Form Fields Editing

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.

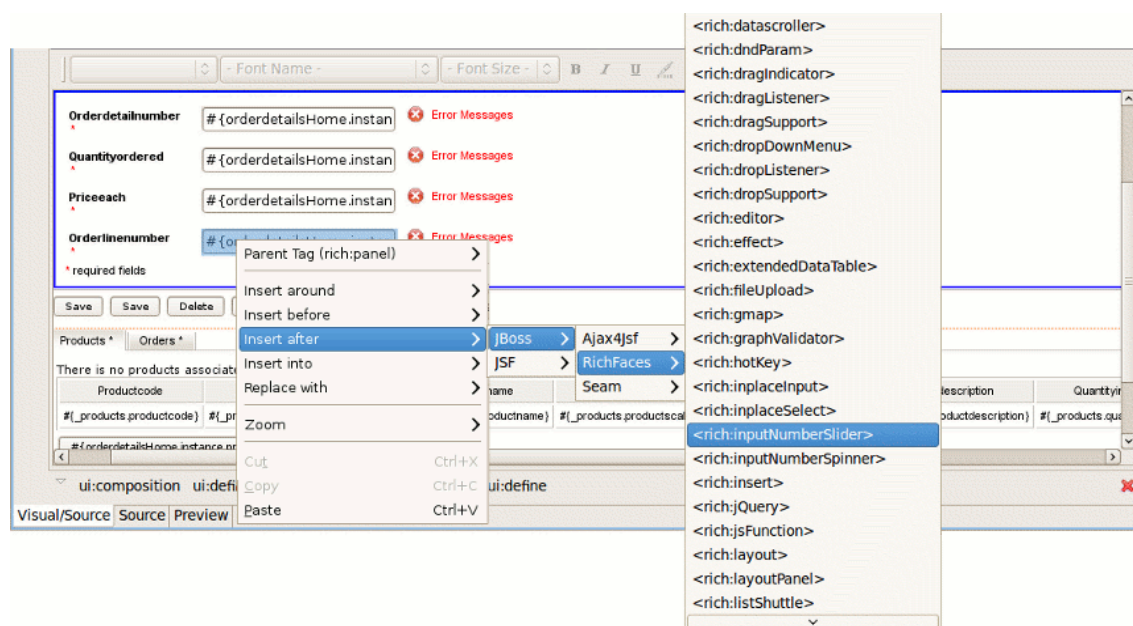


Figure 3.48. Insert RichFaces Component from Context Menu

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}"/>
```

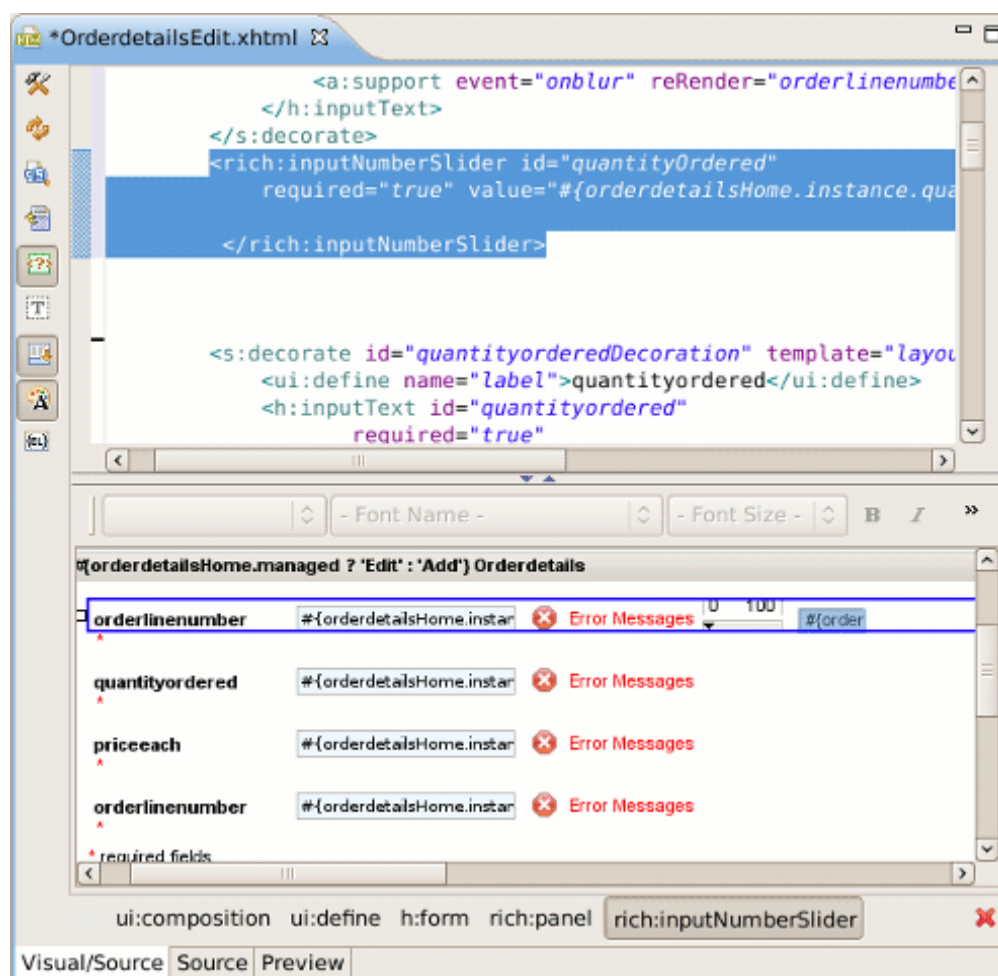


Figure 3.49. Manually coping Source Code

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

Chapter 3. Write Your First Project with JBoss Developer Studio

OrderdetailsEdit.xhtml workshop

http://localhost:8080/workshop/OrderdetailsEdit.seam

workshop: Home Offices List Customers List Employees List Products List Orderdetails List Orders List Productlines List Payments List

Edit Orderdetails

orderdetailnumber * 0 0 100 0

quantityordered * 0

priceeach * 0.0

orderlinenumber * 0

* required fields

Save Cancel

products * orders *

productcode	productlines productline	productname	productscale	productvendor	productdescription	quantityinstock	buyprice	msrp	action
S18_1749	Vintage Cars	1917 Grand Touring Sedan	1:18	Welly Diecast Productions	This 1:18 scale replica of the 1917 Grand Touring car has all the features you would expect from museum quality reproductions: all four doors and bi-fold hood opening, detailed engine and instrument panel, chrome-look trim, and tufted upholstery, all topped off with a factory baked-enamel finish.	2724	86.7	170.0	View

Powered by Seam. Generated by seam-gen.

Figure 3.50. 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 the [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 with the 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 [Next](#)
- Enter "jspHello" as a project name
- Then select [Minimal Configuration](#) from the list of possible configurations and click [Finish](#)

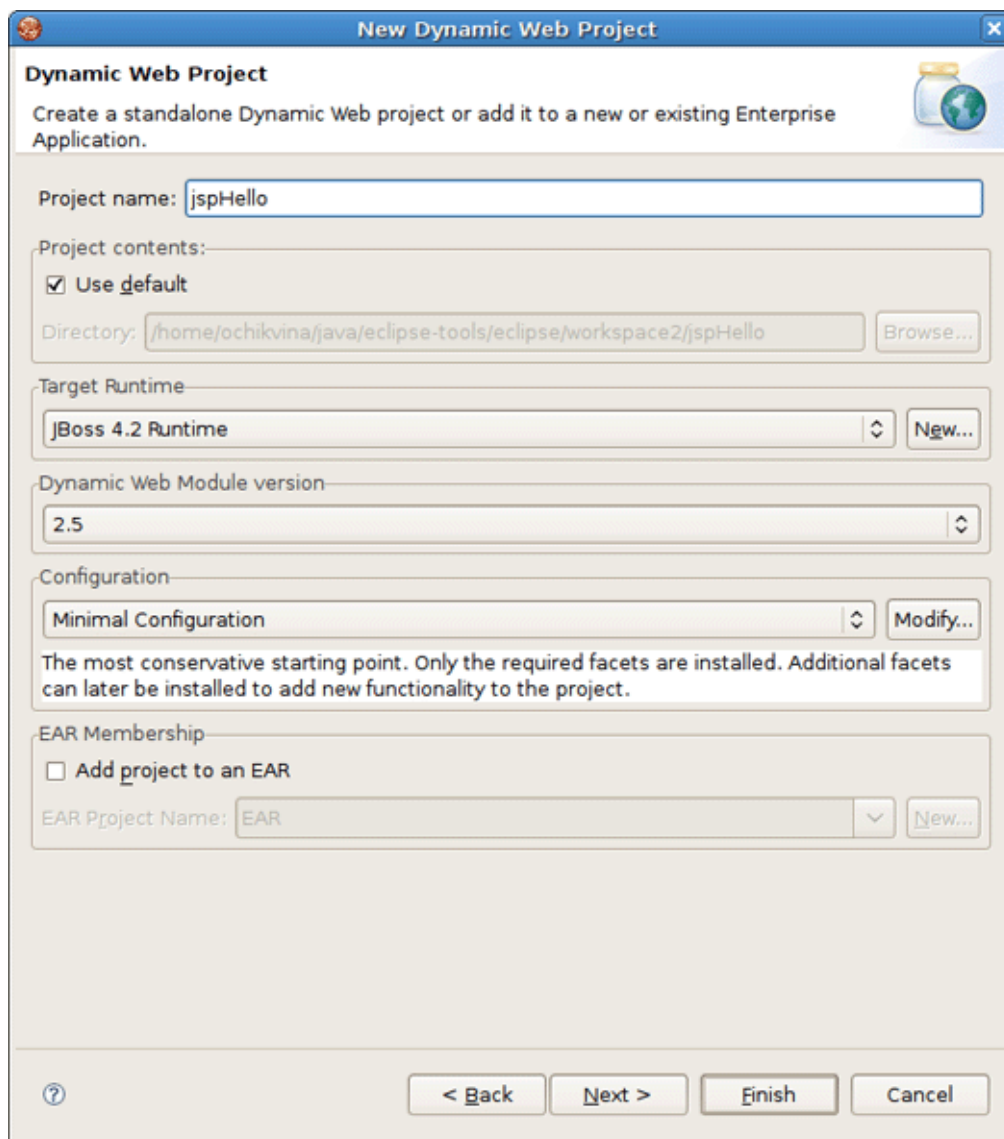


Figure 4.1. Create New Web Project

The *jspHello* node should appear in the upper-left [Package Explorer](#) view.

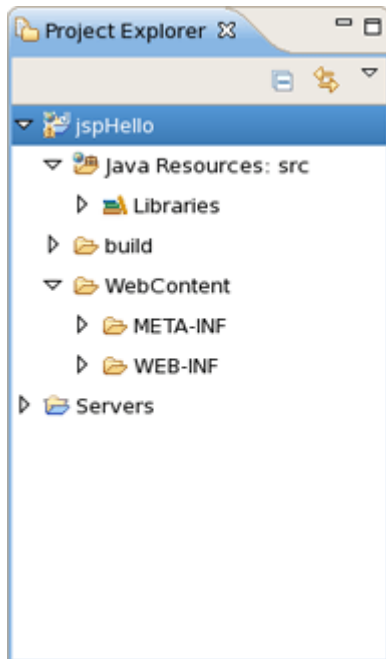


Figure 4.2. New Web Project

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 *WebContent* > *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 *Finish* button.

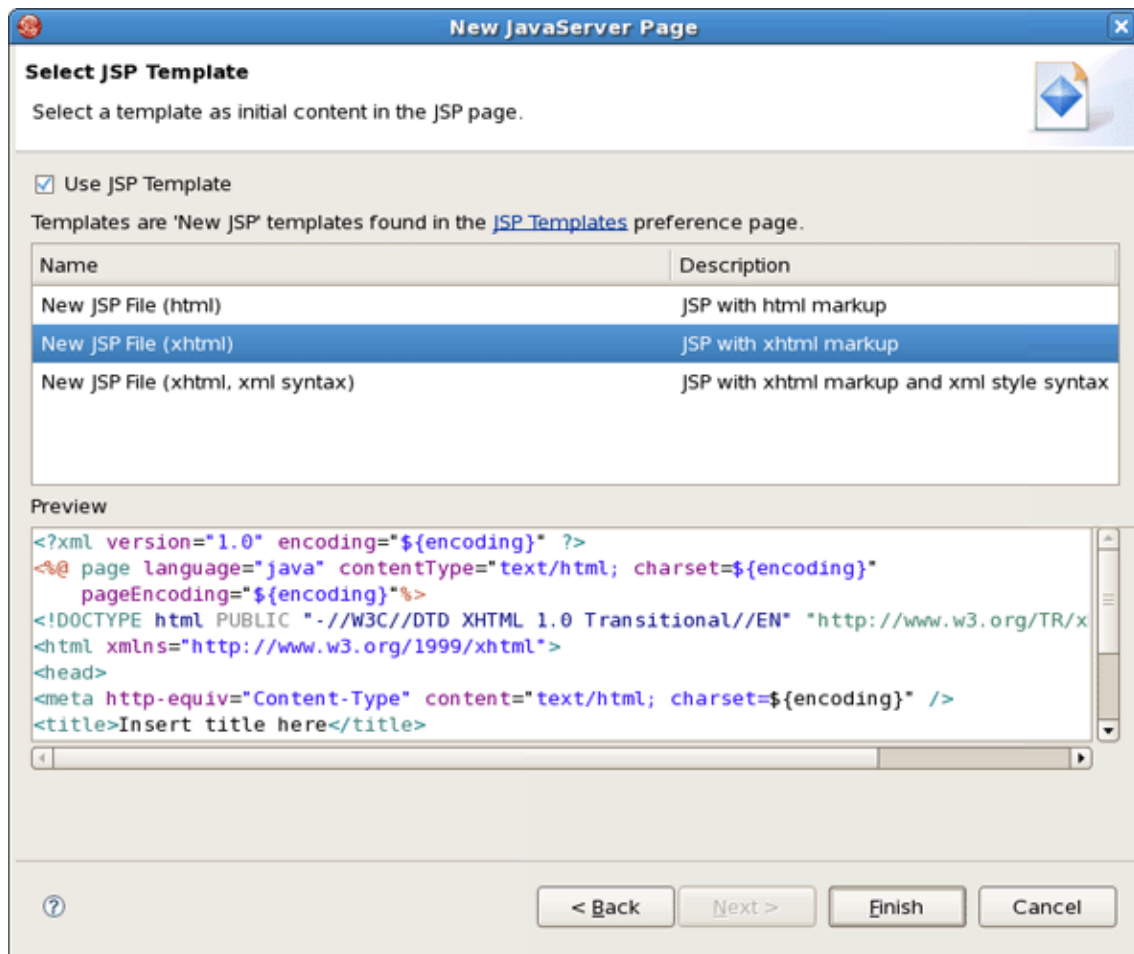


Figure 4.3. Create JSP Page

Our [hello.jsp](#) page will now appear in [Project Explorer](#).

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:

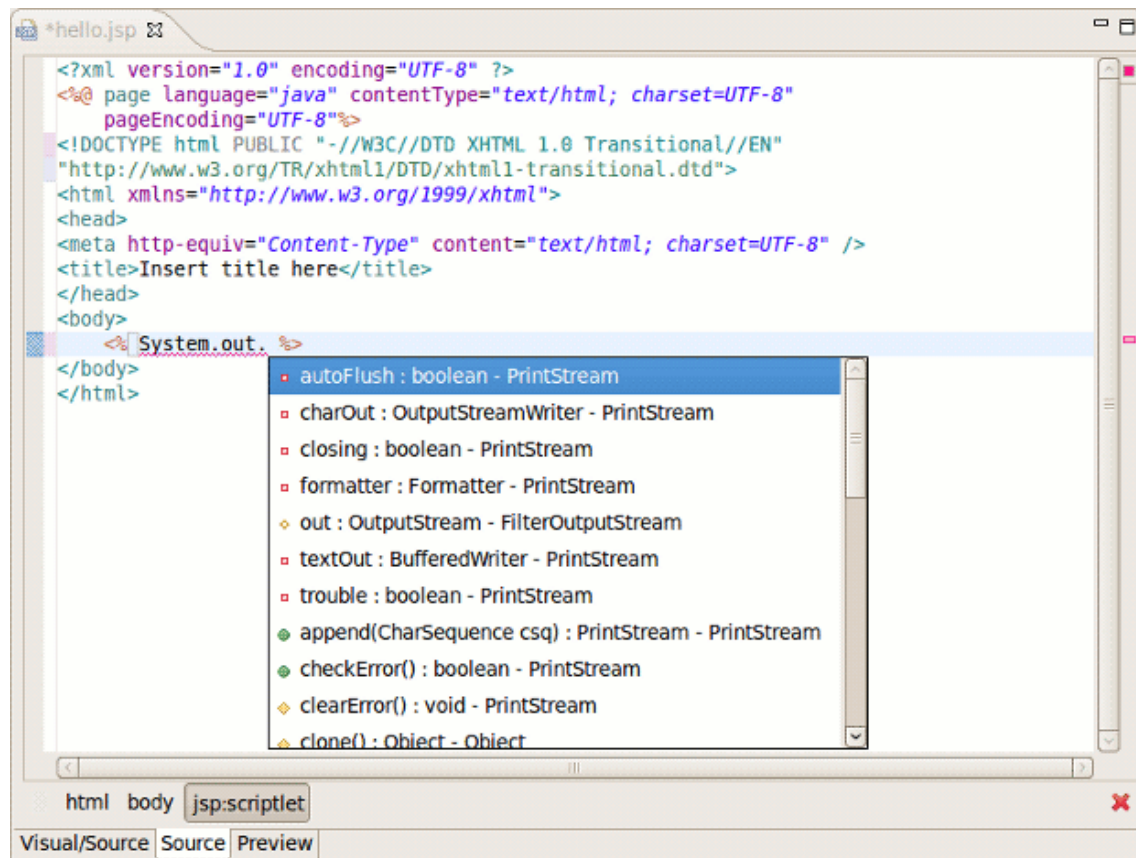


Figure 4.4. Content Assist in JSP Page

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

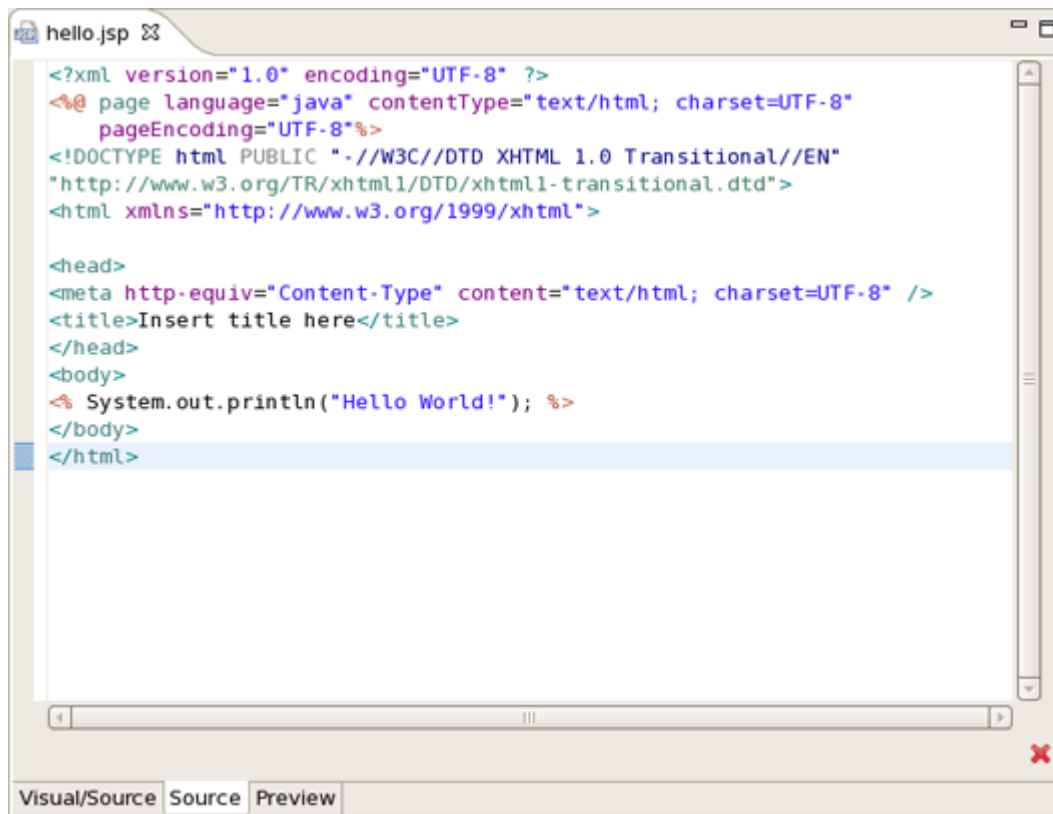


Figure 4.5. Hello.jsp Page Source

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](#) for you automatically. The [web.xml file editor](#) provided by [JBoss Developer Studio](#) is available in two modes: [Tree](#) and [Source](#).

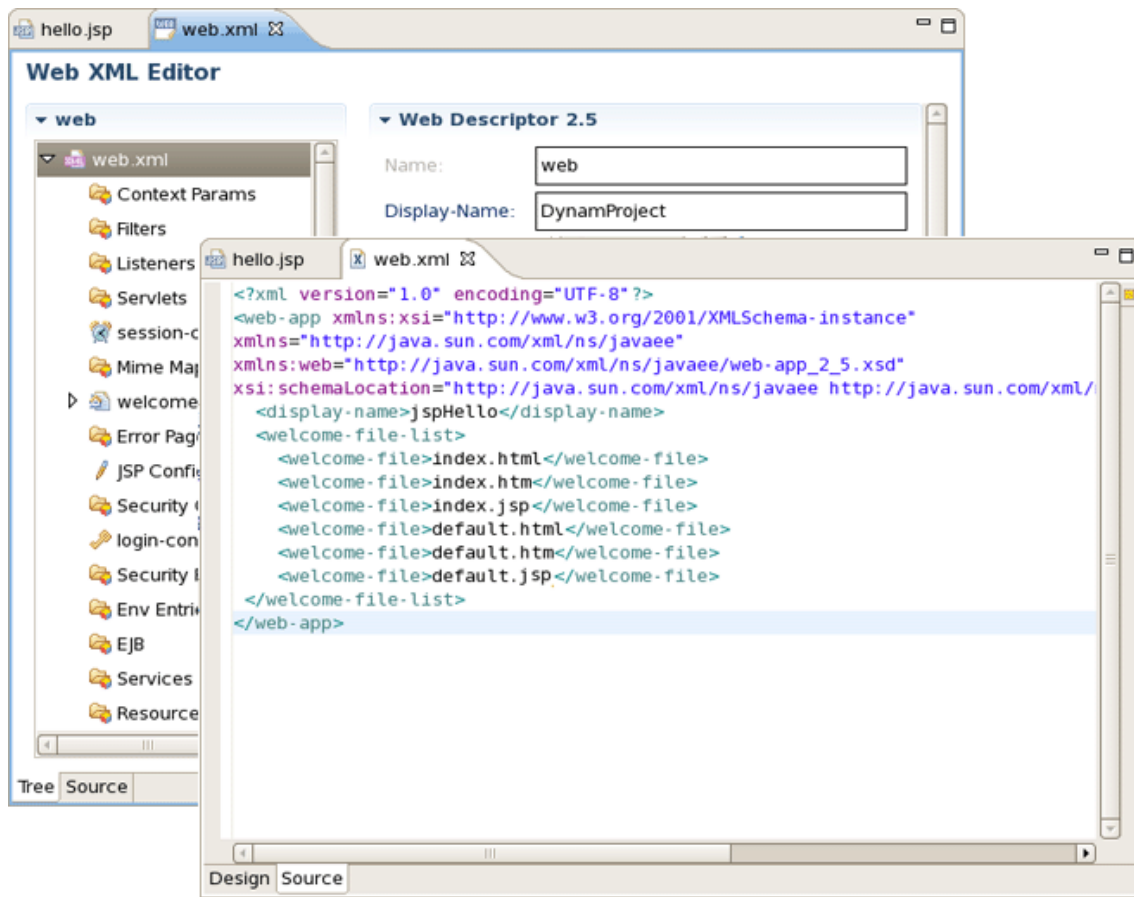


Figure 4.6. Web.xml in Design and Source Mode

Both modes are fully synchronized. Let's add mapping to our [hello.jsp](#) page in [web.xml](#) file.

- Switch to [Source](#) tab.
- Add the next code into `<welcome-file-list>` :

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

If you come back to [Tree](#) mode you will see that the changes made are automatically reflected in that mode.

Actually you don't really need to do any configurations right now.

4.2.3. Deploying the project

While creating any web project you could experience a pain writing ant scripts and managing the packaging even when writing the most trivial web applications. With [JBoss Developer Studio](#) you are saved from such a pain. All you need is to start [JBoss Server](#) and launch your application in your favorite browser.

You can also create a war archive with JBDS's Archive Tools and export it to any web server.

4.2.3.1. WAR Config

Project archives managing is available through Project Archives view.

- Select [Window > Show view > Other > JBoss Tools > Project archives](#) from menu bar
- Select a project in Package Explorer you want to be archived

In [Project Archives](#) you will see available archive types for the project:

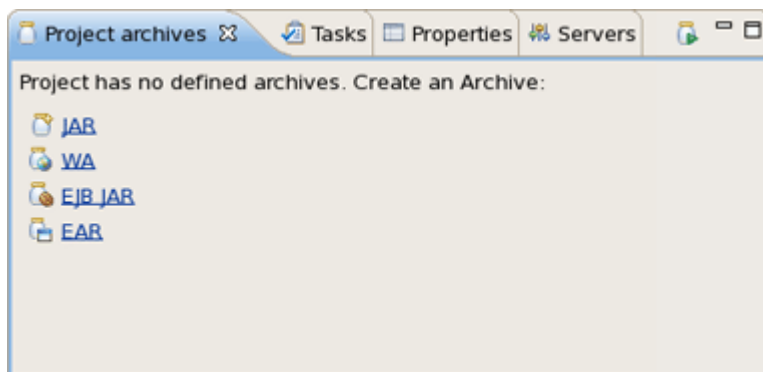


Figure 4.7. Project Archives

- Click, for example, [WAR](#) option to create war archive

In the [New WAR](#) dialog you can see automatically selected default values.

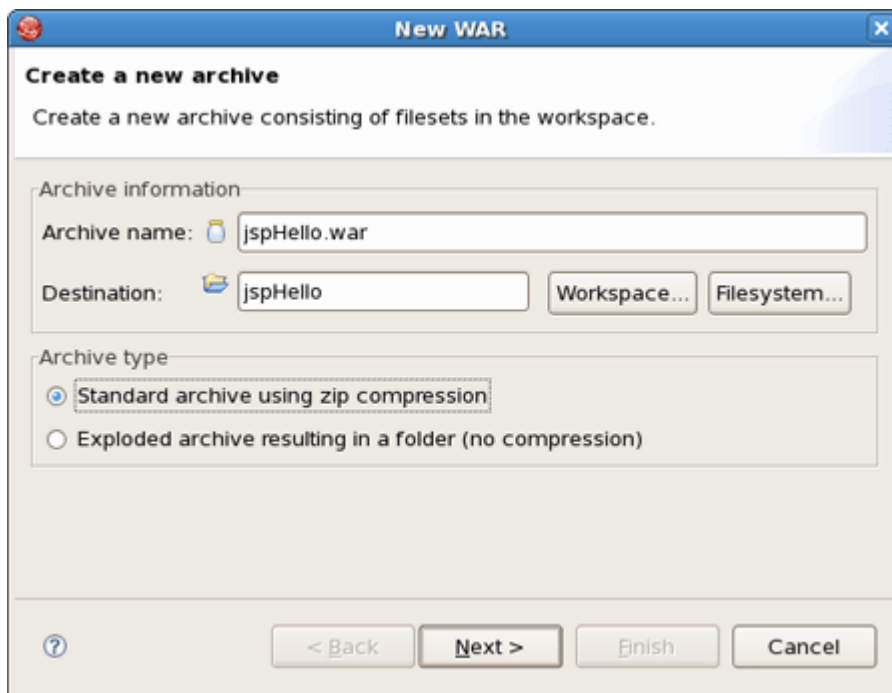


Figure 4.8. New WAR Archive

- Click [Next](#) to see a stub archive configuration for your project:

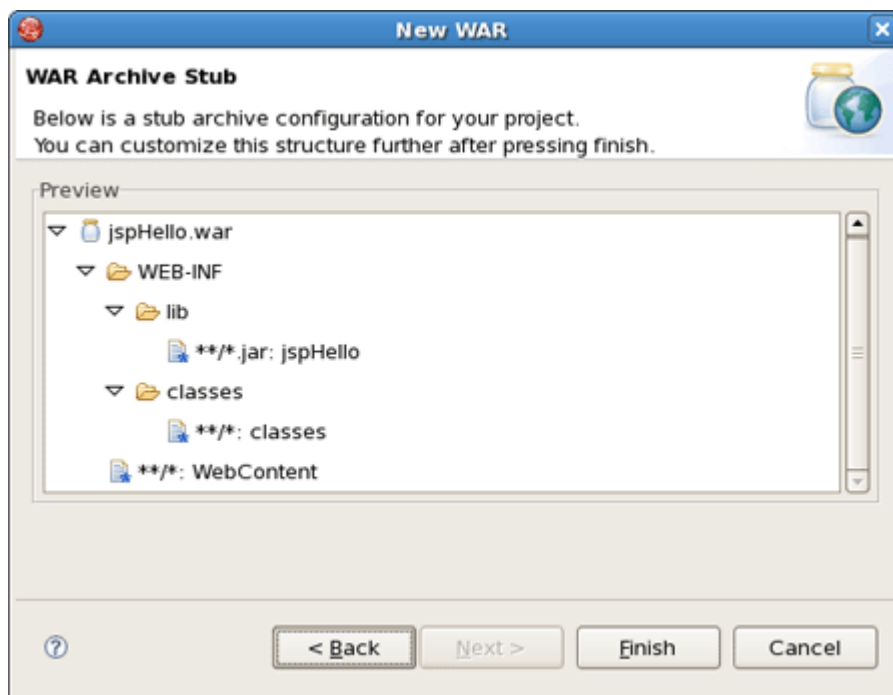


Figure 4.9. Stub Archive Configuration

- Click [Finish](#). The .war file will appear in [Package Explorer](#) and also in [Project Archives](#) view as structure tree:

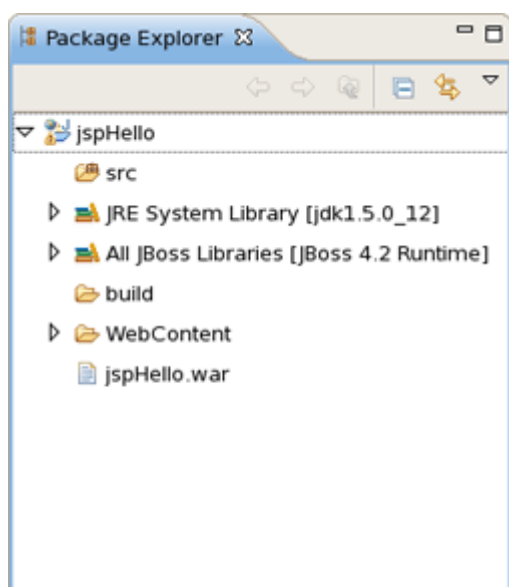


Figure 4.10. Archive is Created

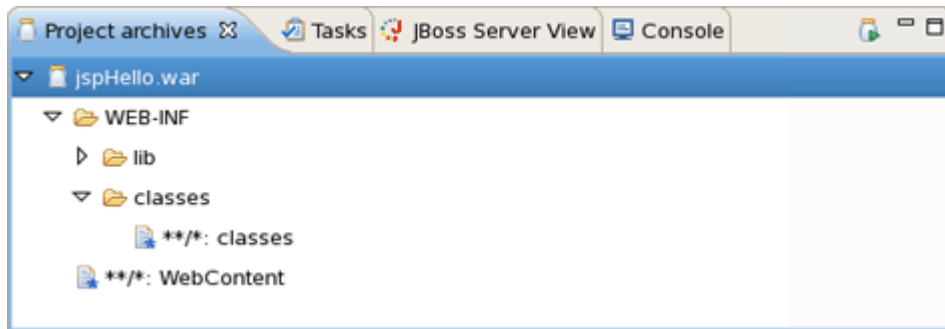


Figure 4.11. Archive in Project Archives View

Via [Project Archives](#) view you could now edit your archive, add new folders, publish to server, and so on:

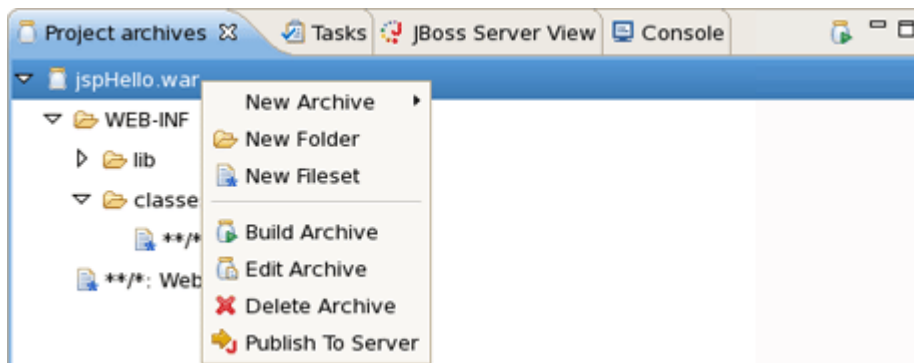


Figure 4.12. Configure Archive

4.2.3.2. Auto redeploy

When you are creating a web application and register it on [JBoss Server](#) it is automatically deployed into `/deploy` directory of the server. JBDS 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

JBDS 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 [Save](#) button.
- Switch to Preview page by clicking [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 [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 any stuff to it (in our case it's just one jsp page) and deploy and run it on the [JBoss Server](#) shipped with [JBDS](#).

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 see how to create a simple JSF application being based on "RAD" philosophy. We will create the familiar Guess Number application. The scenario is the following. 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 powerful features of 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 JBDS's new project wizard and predefined templates. Follow the next steps:

- In Web Projects (if it is not open select [Window > Show View > Others > JBoss Tools Web > Web Projects](#)) click [Create New JSF Project](#) button.

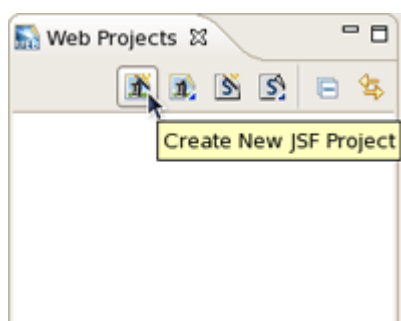


Figure 5.1. Create New JSF Project

- Put [GuessNumber](#) as a project name, in JSF Environment drop down list choose [JSF 1.2](#)
- Leave everything else as it is and click [Finish](#)

Our project will appear in Project Explorer and Web Projects. As you can see JBoss Developer Studio has created for us the whole skeleton for the project with all needed libraries, faces-config.xml and web.xml files.

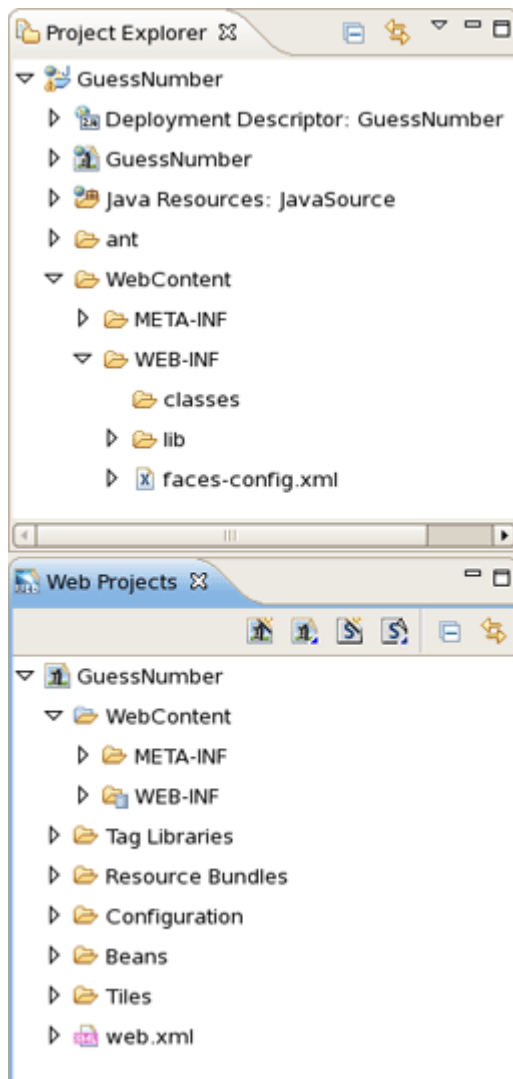


Figure 5.2. New JSF Project

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

5.2. Creating JSP Pages

Here, we are going to add two pages to our application. The first page is `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 `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.

- Open [faces-config.xml](#) file
- Right click anywhere on the diagram mode
- From the context menu select [New View](#)

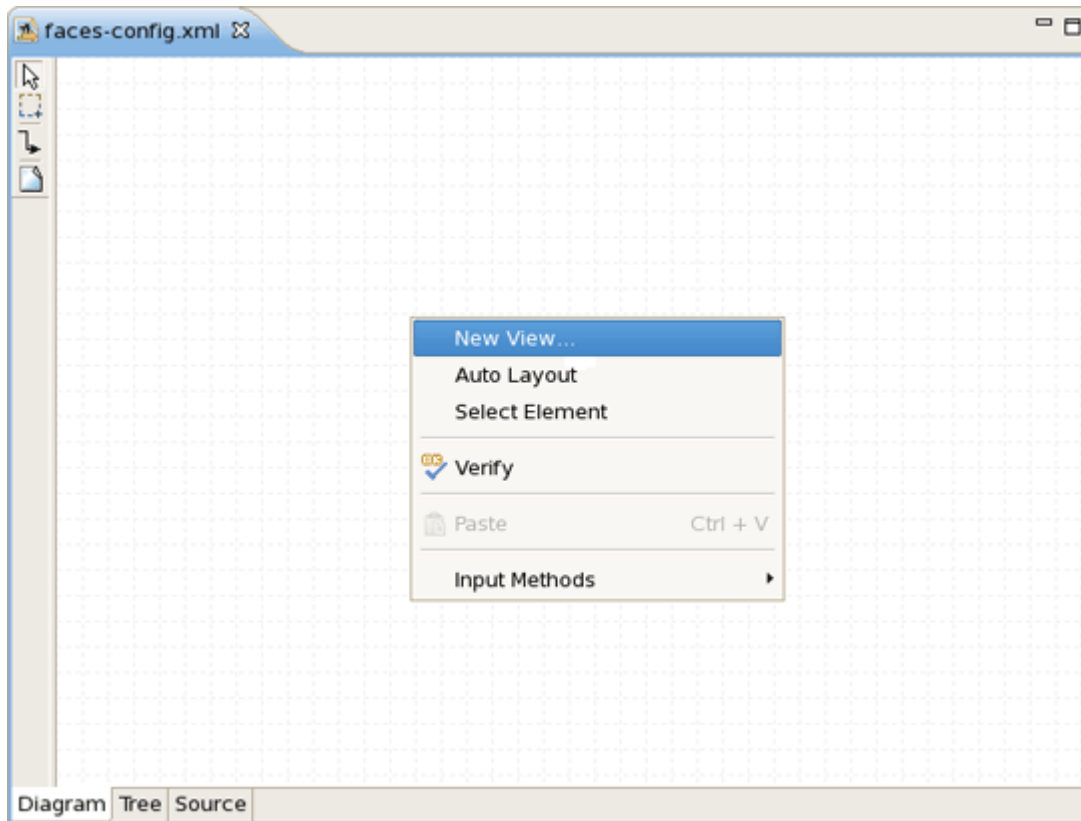


Figure 5.3. Create New View

- Type [pages/inputnumber](#) as the value for [From-view-id](#)
- Leave everything else as is and click [Finish](#)
- 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.

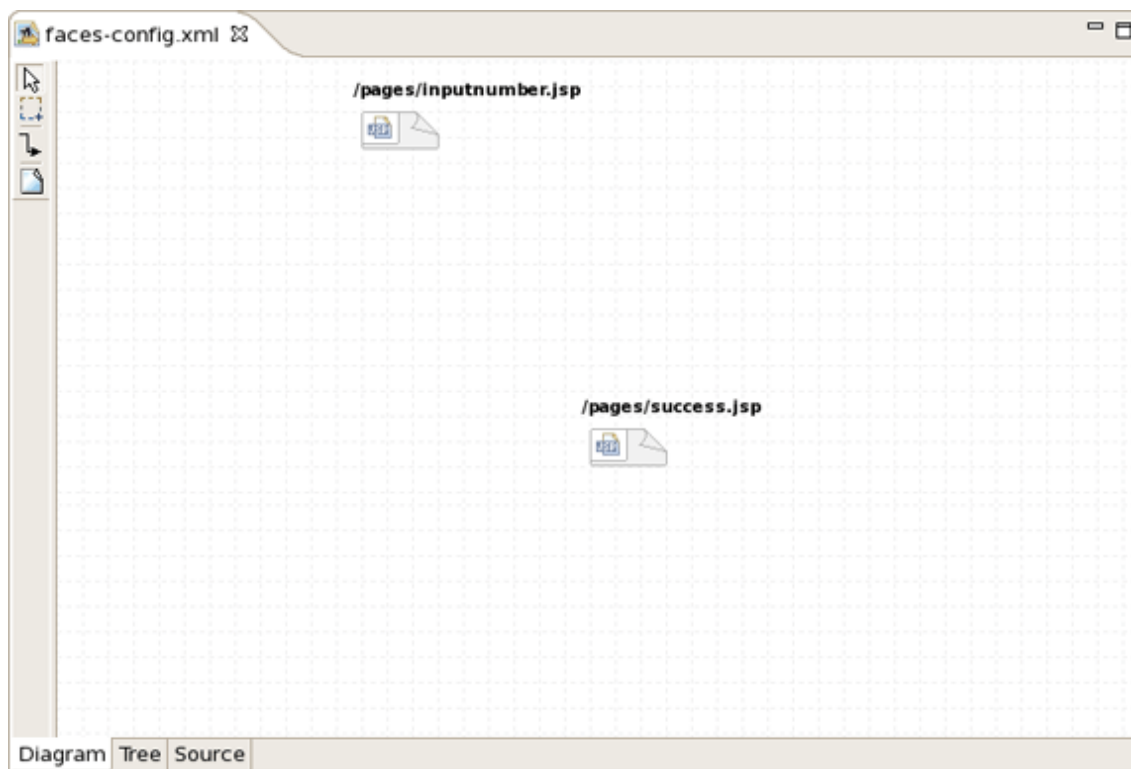


Figure 5.4. New Views

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

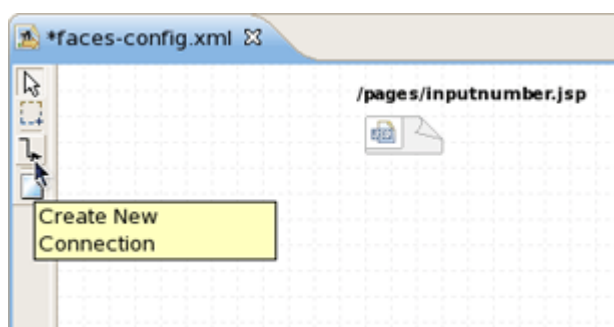


Figure 5.5. Create Connection

- 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.

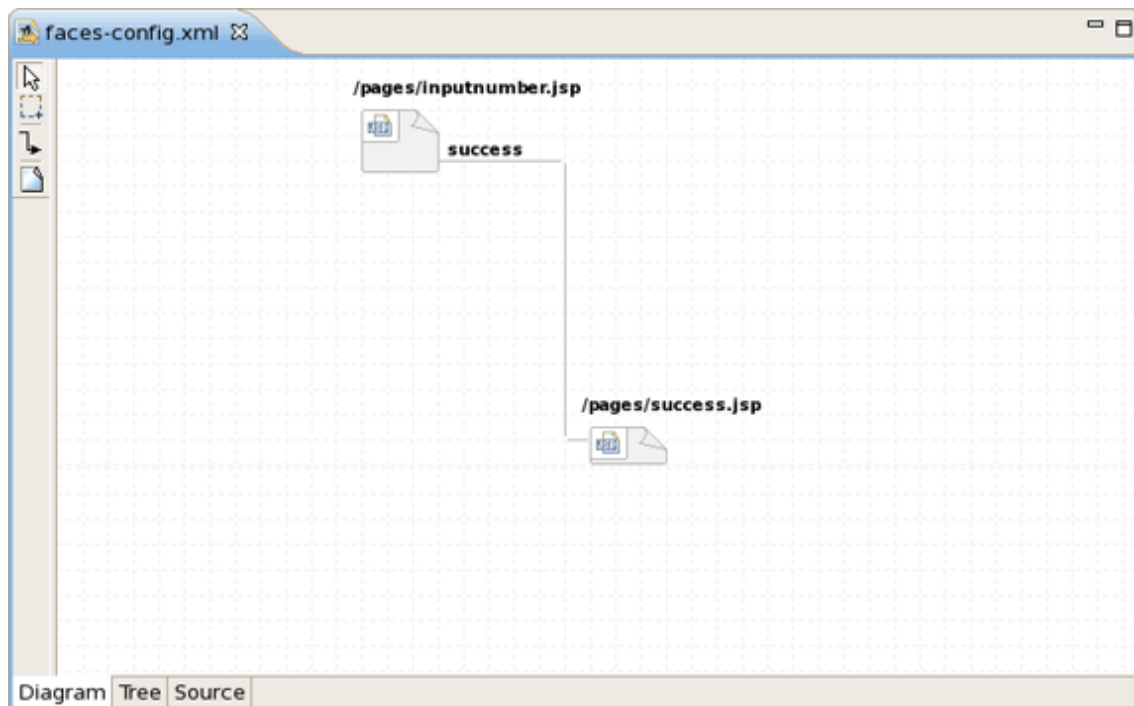


Figure 5.6. Created Connection

- 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 first, you don't hard code anything into the JSP pages. And second, it 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 [JavaSource](#) folder and select [New > Folder](#)
- Type [game](#) for Folder name and click [Finish](#)

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

- Right click on [game folder](#) and select [New > Properties File](#)

- Type `messages` as the value for "name" attribute and click `Finish`

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

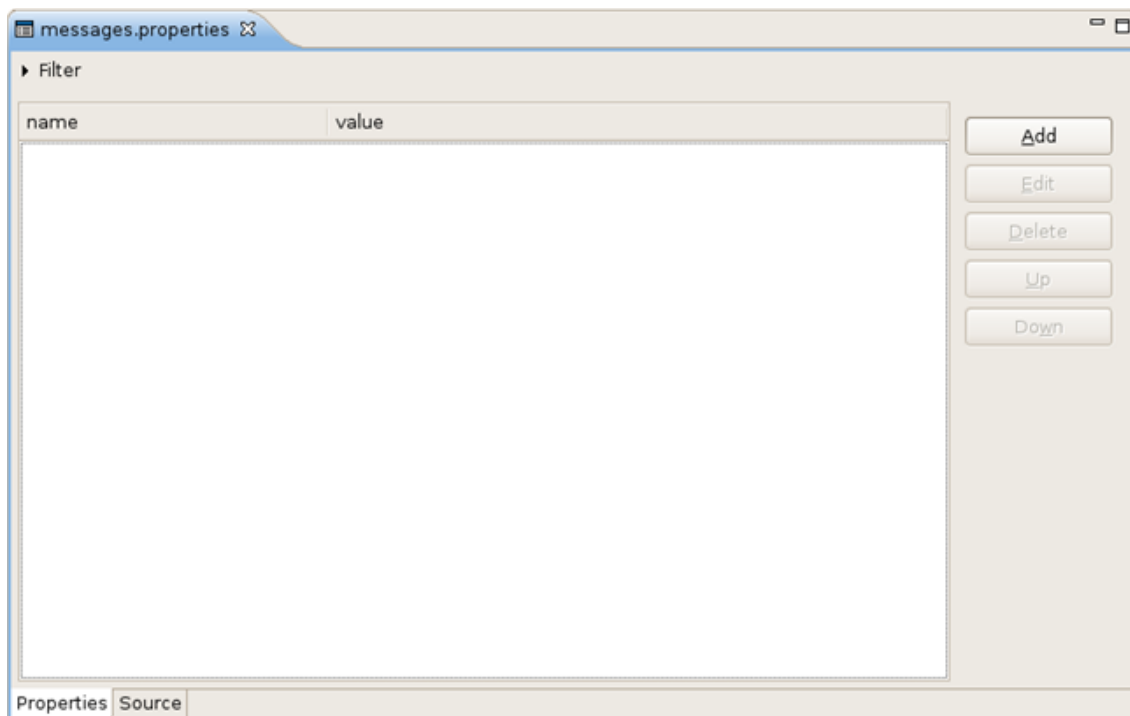


Figure 5.7. Messages.properties File

- Click `Add` button for adding new attribute to your resource file
- Type `how_to_play` for "name" and `Please pick a number between 0 and 100.` for value
- Click `Finish`
- In such a way add the next properties:

```
makeguess_button=Make Guess
tryagain_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.
```

- Click `File > Save` from the menu bar

Your .properties file should now look like follows:

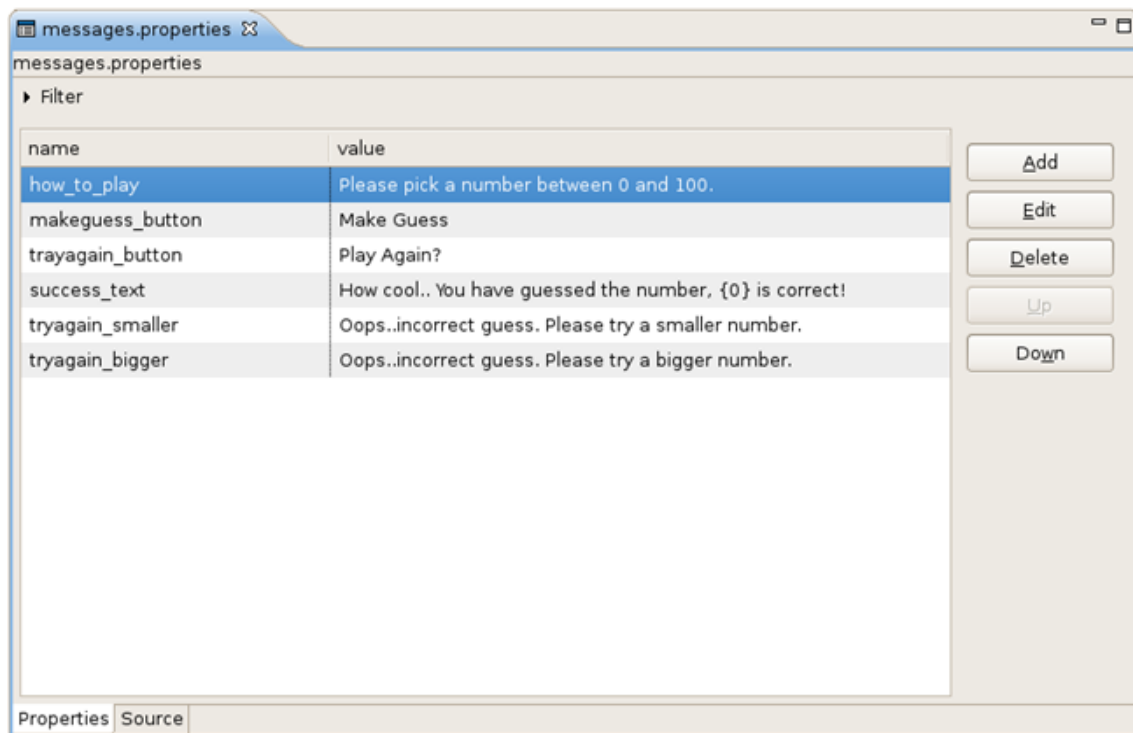


Figure 5.8. Properties are Added

[Up](#) and [Down](#) buttons allow you to move you attribute on the list. For delete the attribute, choose it and press [Delete](#) button.

If you want to change a value or a name of your attribute, click on it and than on [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 to find the necessary ones. 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

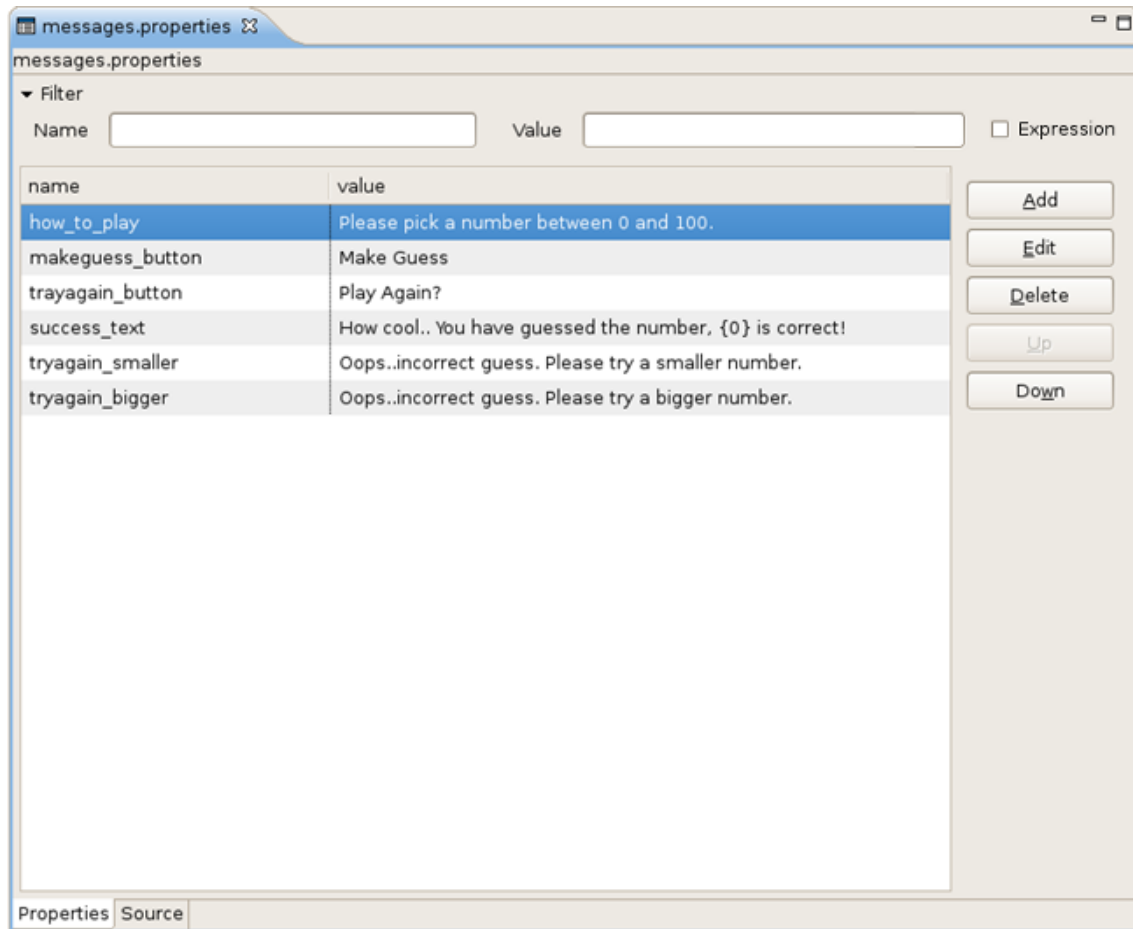


Figure 5.9. Filter and Regular Expressions Search Panel

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:

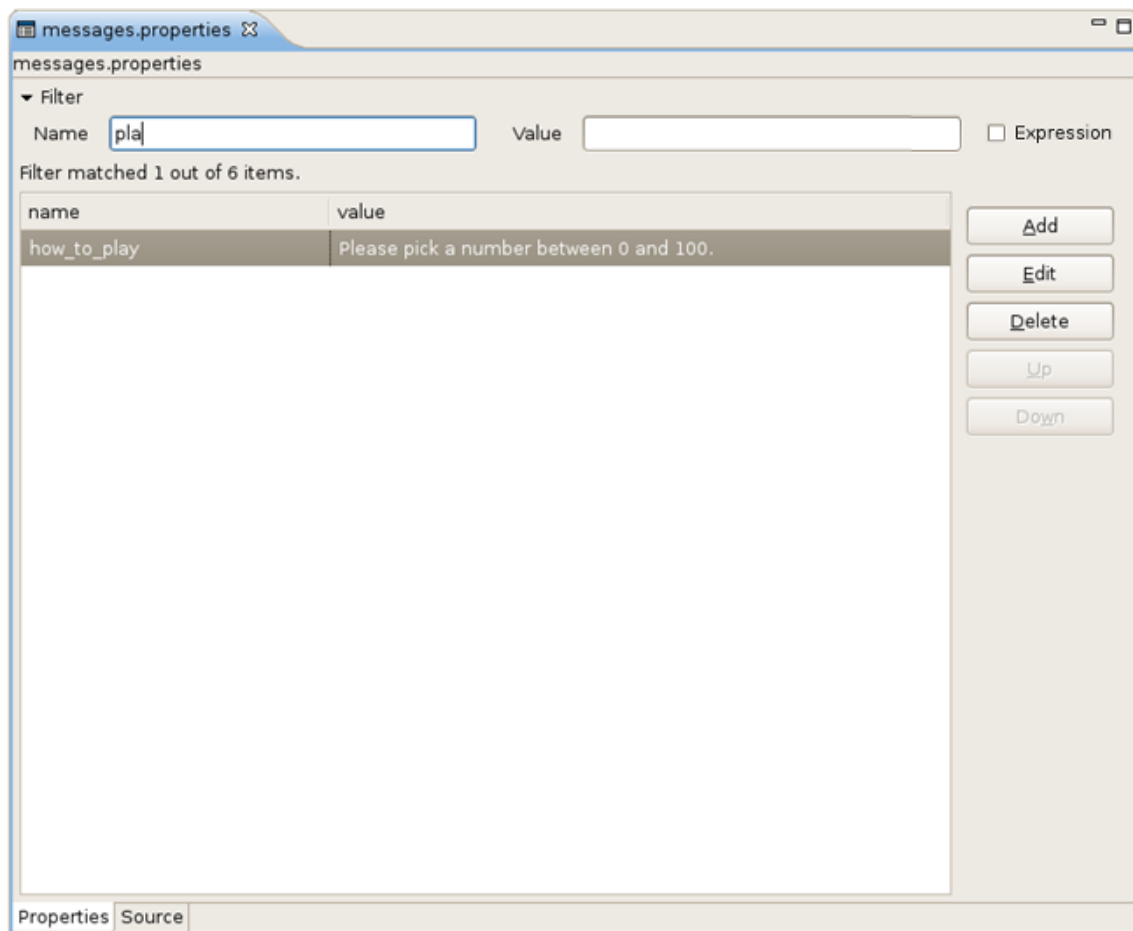


Figure 5.10. Filter results

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 see how to create a Java bean that will hold business logic of our application.

- Right click [game folder](#)
- Select [New > Class](#)
- Type [NumberBean](#) for bean name

A java bean is created.

- Declare the variable of your entered number:

```
Integer userNumber;
```

JBDS allows to quickly generate getters and setters for java bean.

- Right click *NumberBean.java* in Package Explorer
- Select *Source > Generate Getters and Setters...*
- Check *userNumber* box and click *OK*

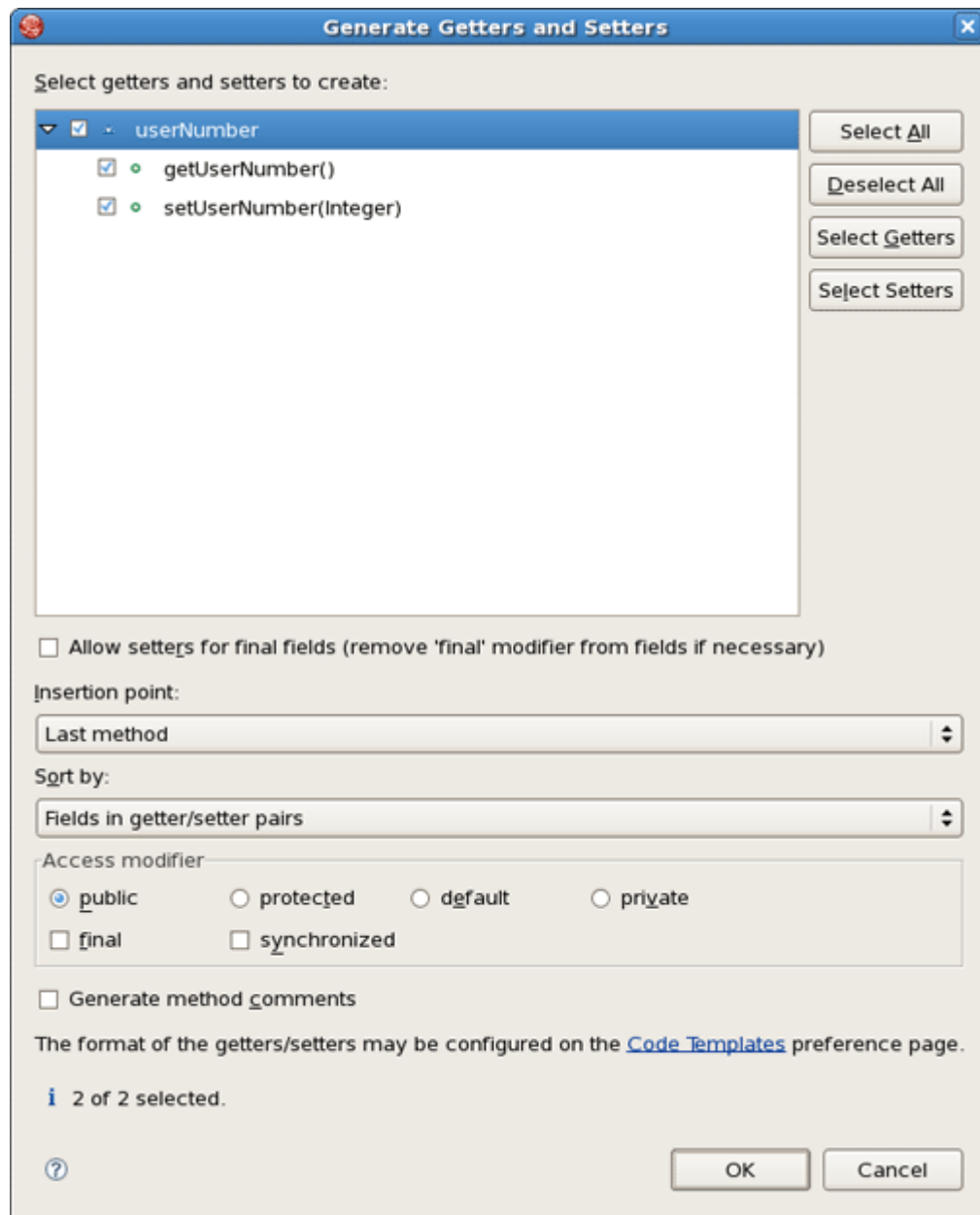


Figure 5.11. Generate Getters and Setters

- 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 should look as follows:

```
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 know about faces-config.xml file.

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

- Open faces-config.xml file in a source mode
- Add here one more navigation rule and a managed bean declarations 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 `/pages/inputnumber.jsp`. 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 Visual Page Editor.

5.7.1. Editing inputnumber.jsp page

First, let's dwell on how to edit `inputnumber.jsp`.

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 `inputnumber.jsp` 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 be already generated as we choose a template when creating the page.

At the beginning it's necessary to create a `<h:form>` component where all others components are put.

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

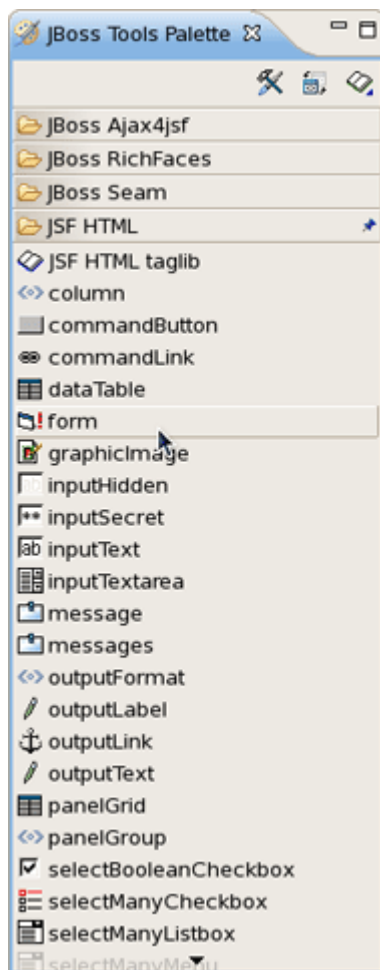


Figure 5.12. Insert h:form

- In the dialog Insert Tag select `id` and click on this line below the value header. A blinking cursor will appear in a input text field inviting to enter a value of id

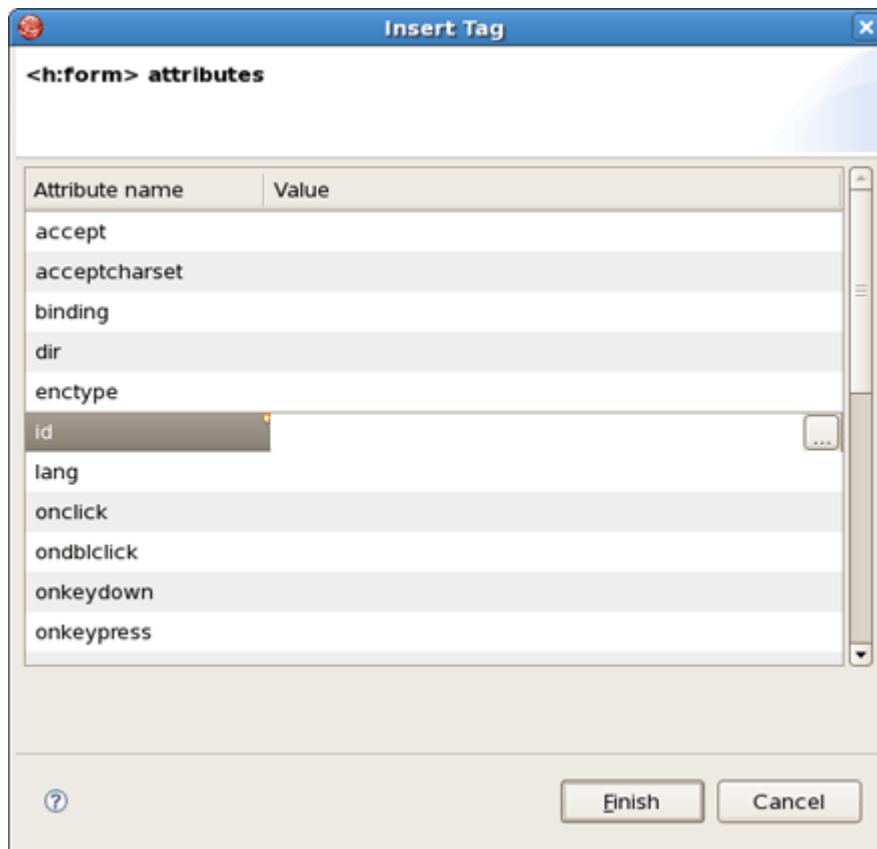


Figure 5.13. Define Id of Form

- Type `inputNumbers` and click `Finish`

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

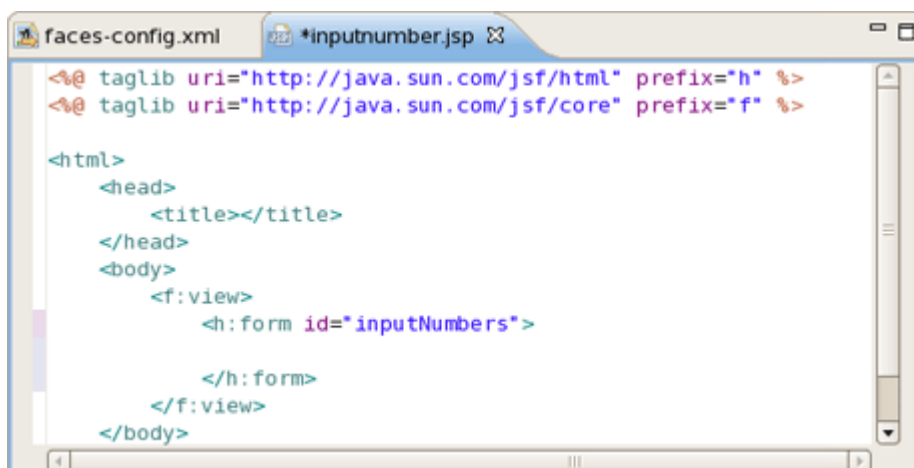


Figure 5.14. Created Form

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

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

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

As always JBDS provides code assist:

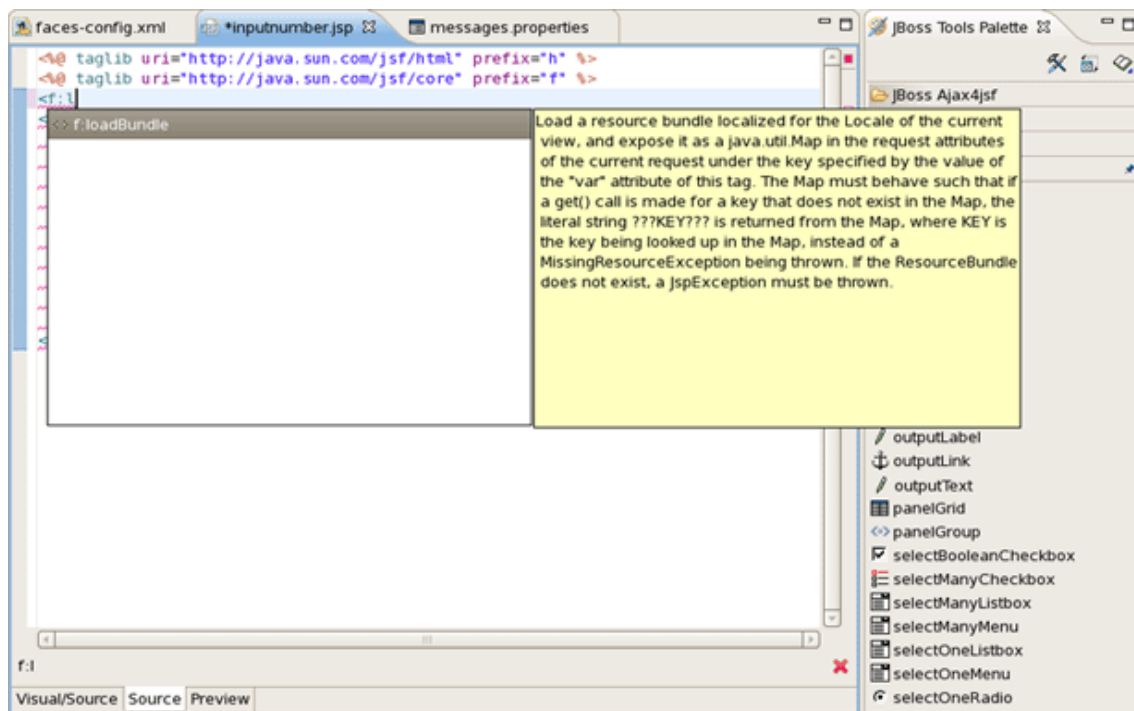


Figure 5.15. Code Assist

- Switch to Visual tab, so it could be possible to work with the editor completely in its WYSIWYG mode
- Click on *outputText*, drag the cursor over to the editor, and drop it inside the blue box in the editor
- Select *value* and click on this line below "value" header
- Click ... button next to the value field

JBDS will nicely propose you to choose within available values:

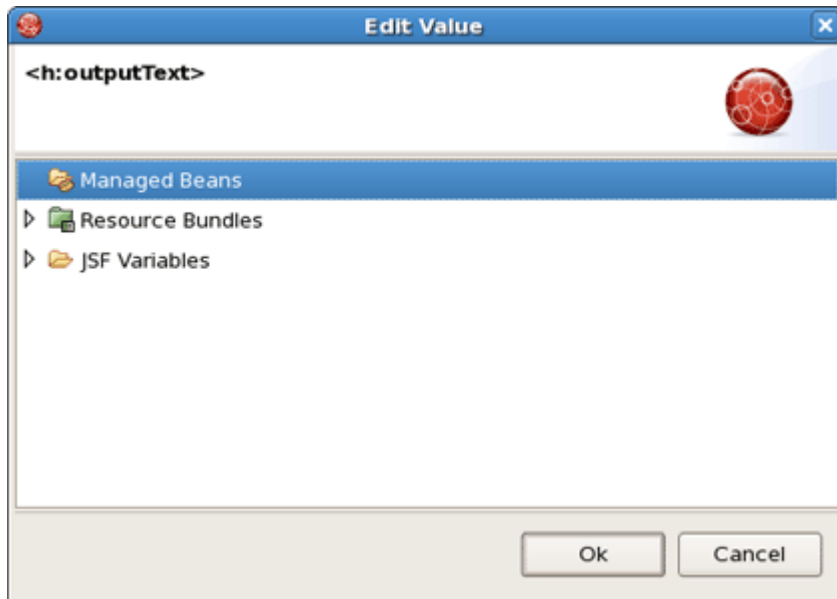


Figure 5.16. Choose Value

- Expand [Resource Bundles > msg](#)
- Select [how_to_play](#) value and click [Ok](#). Then click [Finish](#)

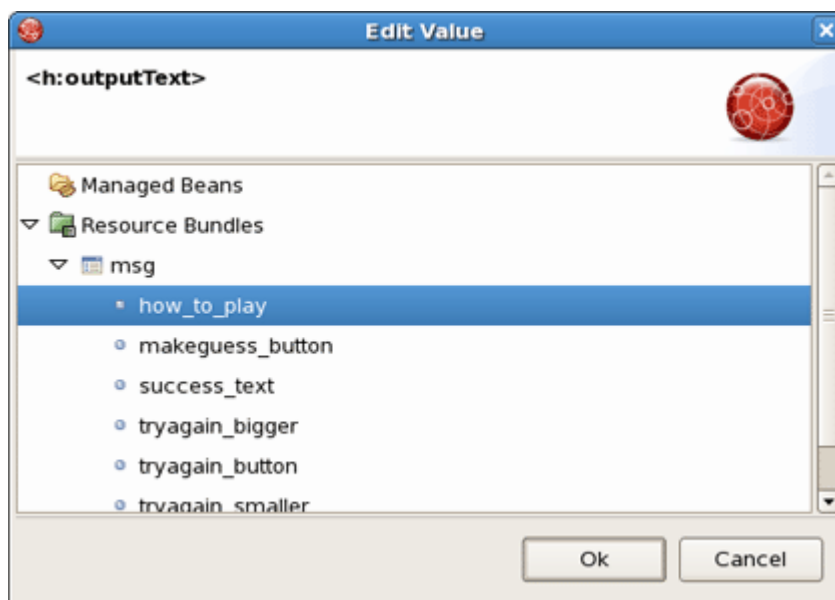


Figure 5.17. Selecting Value

The text will appear on the page:

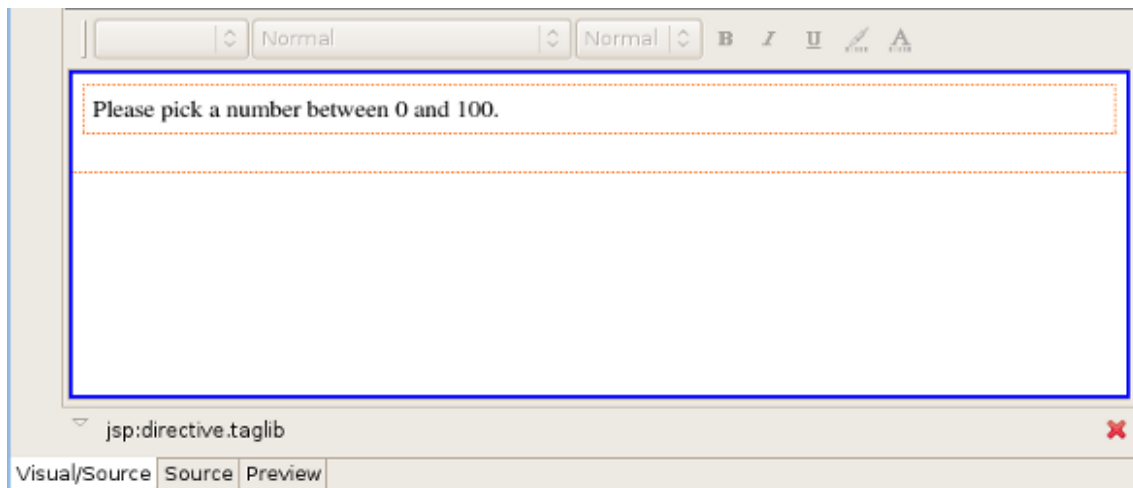


Figure 5.18. Created OutputText Component

- Switch to Source mode and insert `
` tag after `<h:outputText>` component to make a new line.
- Click [Save](#) button.
- On the Palette click on [inputText](#), drag the cursor over to the editor, and drop it inside the editor after the text.
- Switch to a Source mode and insert `
` tag after `<h:outputText>` component to make a new line
- Click [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 [value](#) and click on this line below "value" header
- Click [...](#) button next to the value field
- Expand [Managed Beans > NumberBean](#)
- Select [userNumber](#) value and click [Ok](#)
- Switch [Advanced](#) tab
- Select [id](#) and click on this line below "value" header
- Type [userNumber](#) in text field
- Select [required](#) and click on this line below "value" header
- Click [...](#) button next to the value field
- Expand [Enumeration](#) and select [true](#) as a value

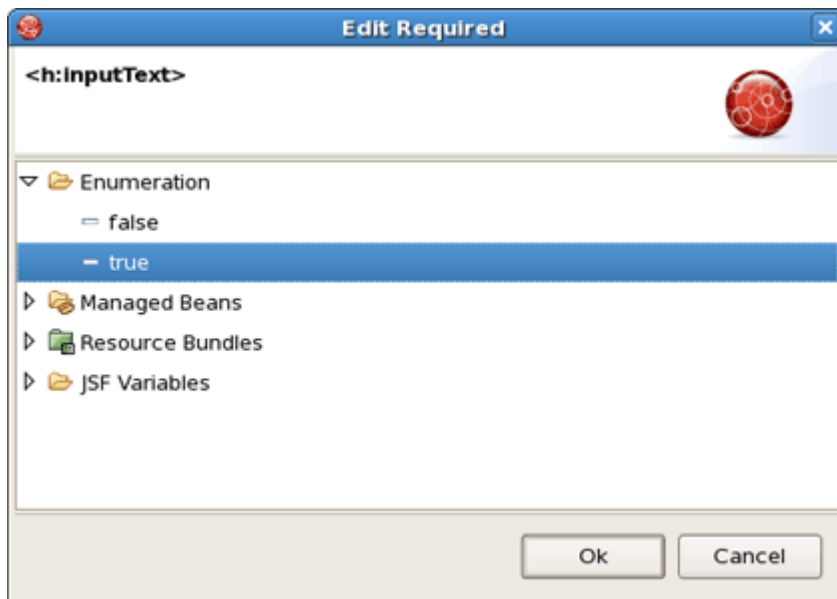


Figure 5.19. Add "required" Attribute

- Click [Ok](#), then click [Finish](#)
- 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 [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 [value](#) and click on this line below "value" header
- Click [...](#) button next to the value field
- Expand [Resource Bundles > msg](#) and select [makeguess_button](#) as a value
- Click [Ok](#)
- Select [action](#) and click on this line below "value" header
- Type [NumberBean.checkGuess](#) in text field
- Click [Finish](#)

- In Source mode add `
` tags between `<outputText>` , `<inputText>` and `<commandButton>` components to place them on different lines

inputnumber.jsp page should look like this:

```
<%@ 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="inputNumbers">
      <h:outputText value="#{msg.how_to_play}" />
      <br />
      <h:messages style="color: blue" />
      <br />
      <h:inputText value="#{NumberBean.userNumber}" id="userNumber"
        required="true">
        <f:validateLongRange minimum="0" maximum="100" />
      </h:inputText>
      <br />
      <br />
      <h:commandButton value="#{msg.makeguess_button}"
        action="#{NumberBean.checkGuess}" />
    </h:form>
  </f:view>
</body>
</html>
```

5.7.2. Editing success.jsp page

In the same way like inputnumber.jsp, edit success.jsp page. Its whole source should be the next:

```
<%@ 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 JBDS when editing jsp page:

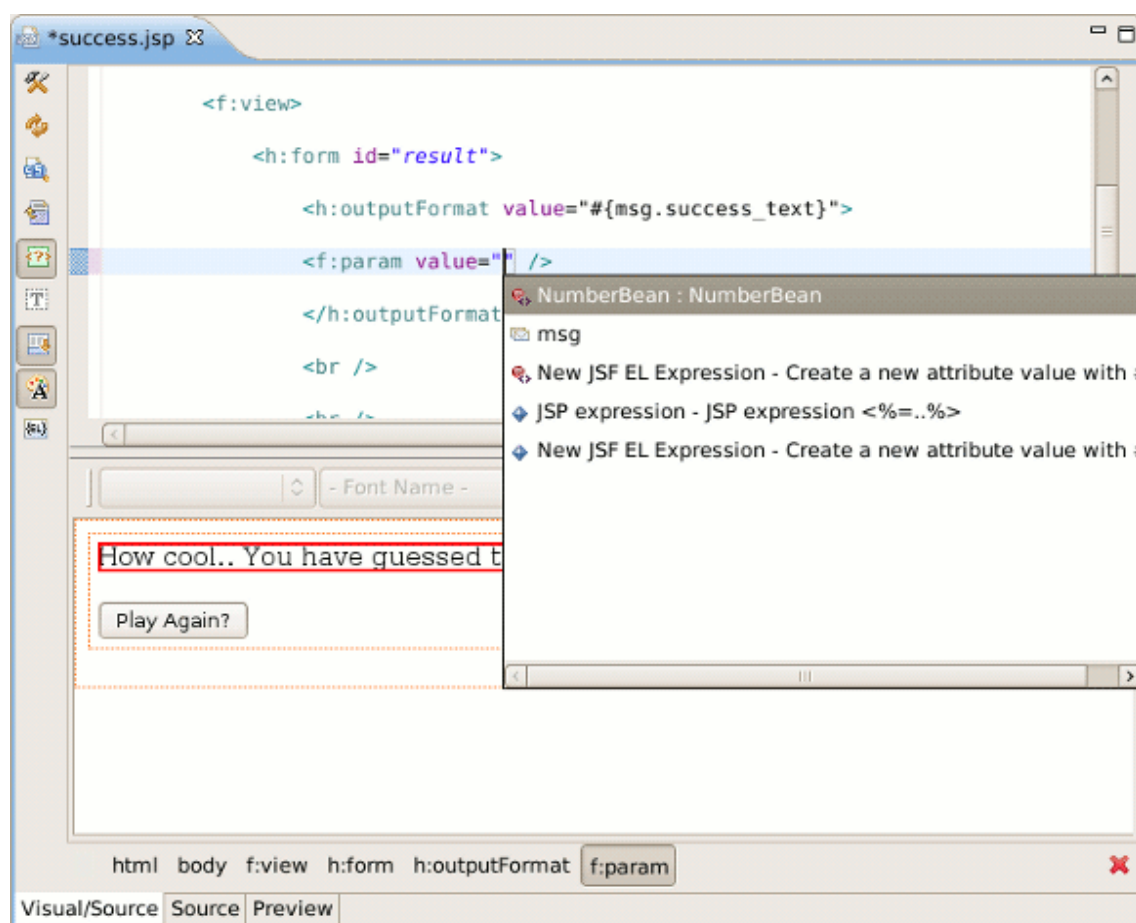


Figure 5.20. Code Assist for <f:param>

This page, `success.jsp`, 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.

At the end, 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:

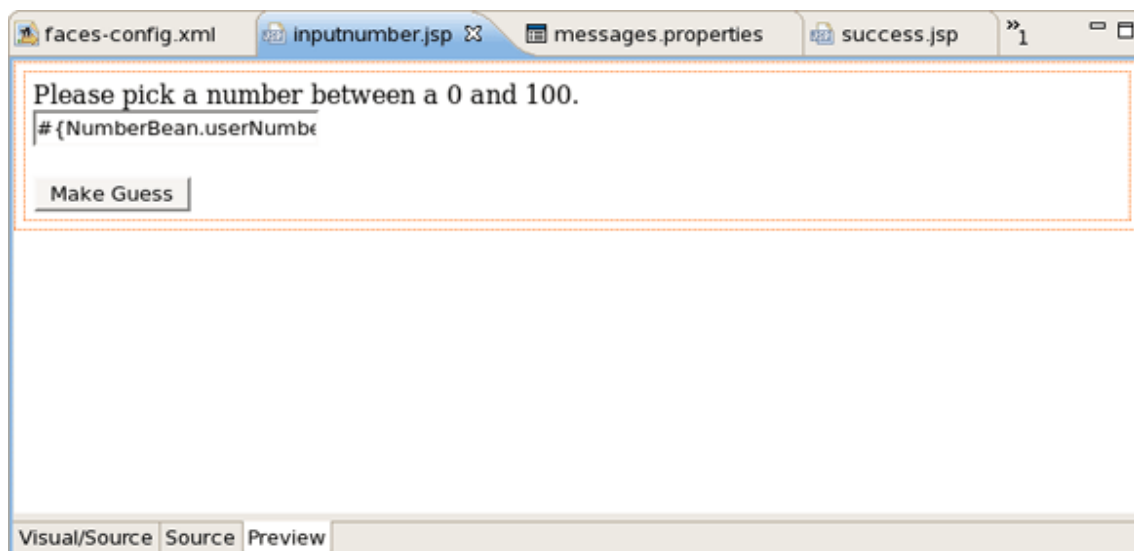


Figure 5.21. Success.jsp in Preview Mode

5.8. Creating index.jsp page

Now you know how to create `index.jsp` page.

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

- Right click [WebContent](#) > [New](#) > [JSP File](#)
- Type [index](#) for name field and choose [JSPRedirect](#) as a template
- Click [Finish](#)
- The source for this page should be 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 [Servers](#). (If JBoss 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 project [Run AS > Run on Server](#)
- Play with the application by entering correct as well as incorrect values

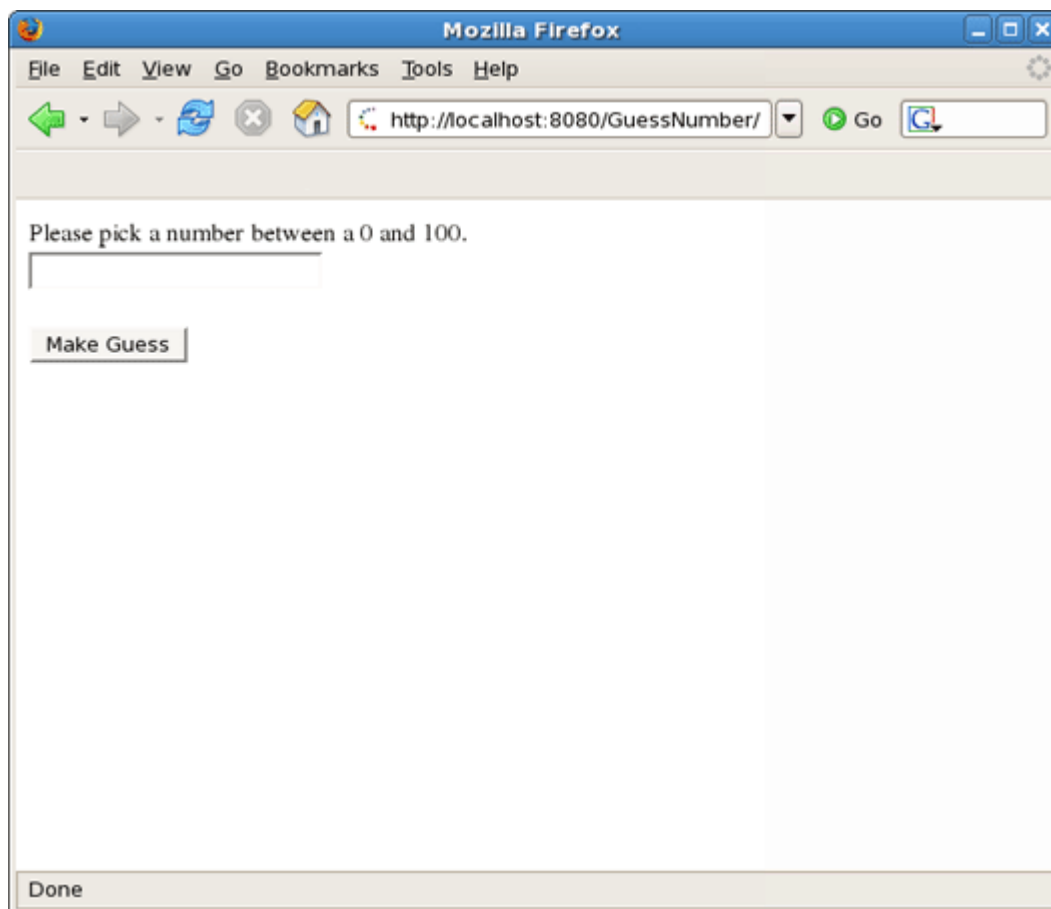


Figure 5.22. You are Asked to Enter a Number Between 0 and 100

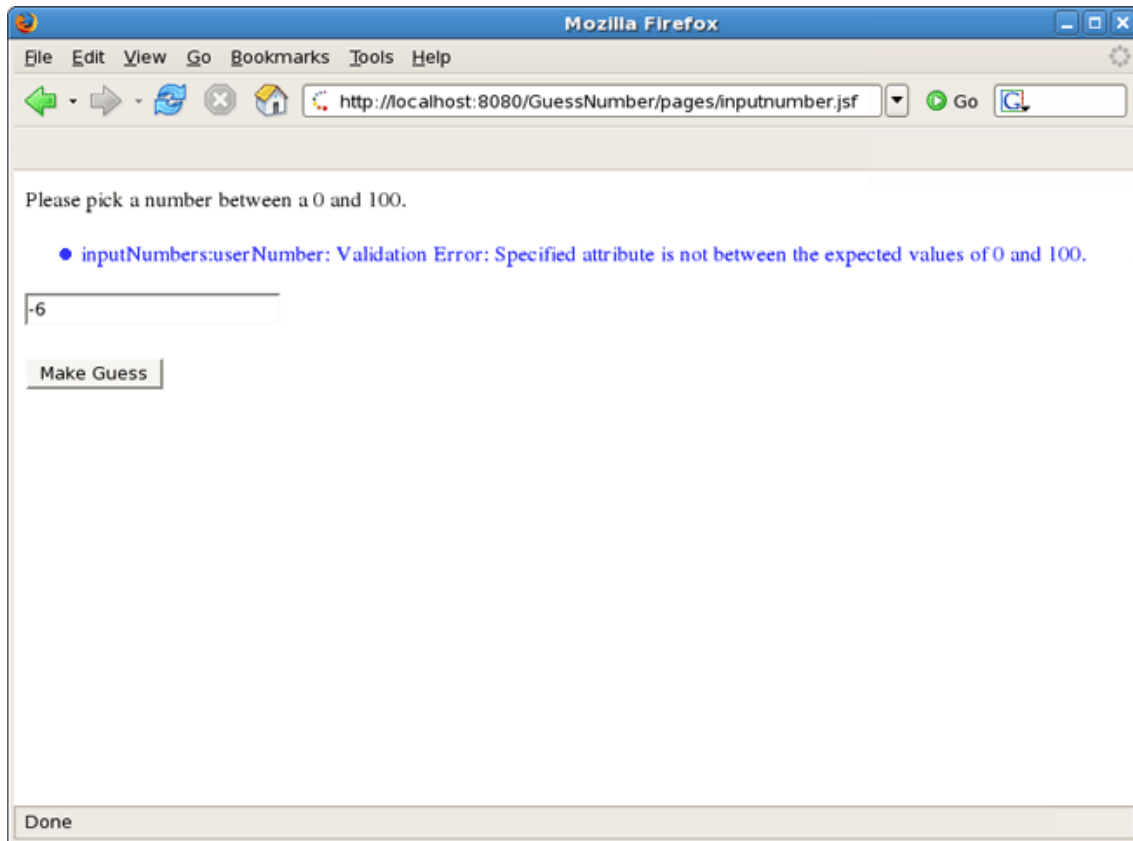


Figure 5.23. Your Input is Validated and an Error Message is Displayed if Invalid Input was Entered

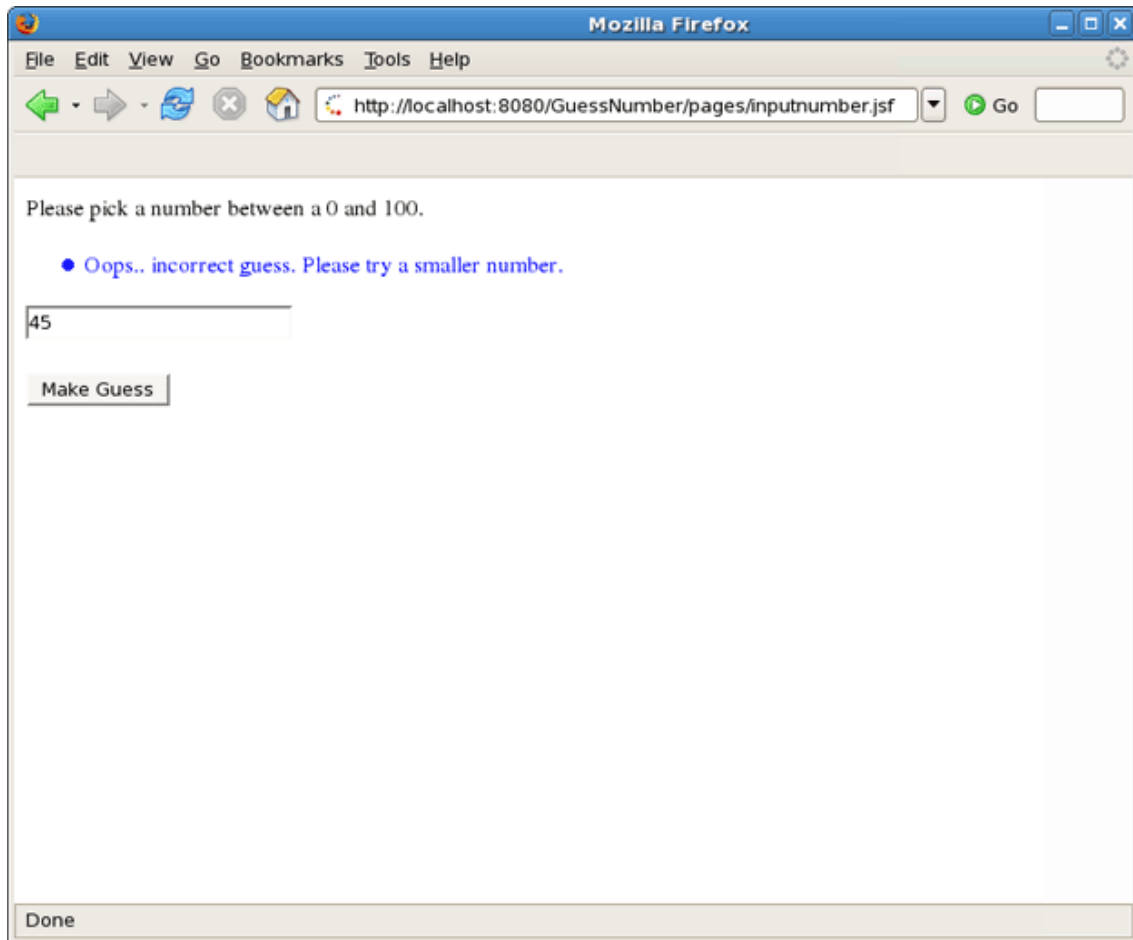


Figure 5.24. After You Enter a Guess, the Application Tells You Whether a Smaller or a Larger Number Should be Tried

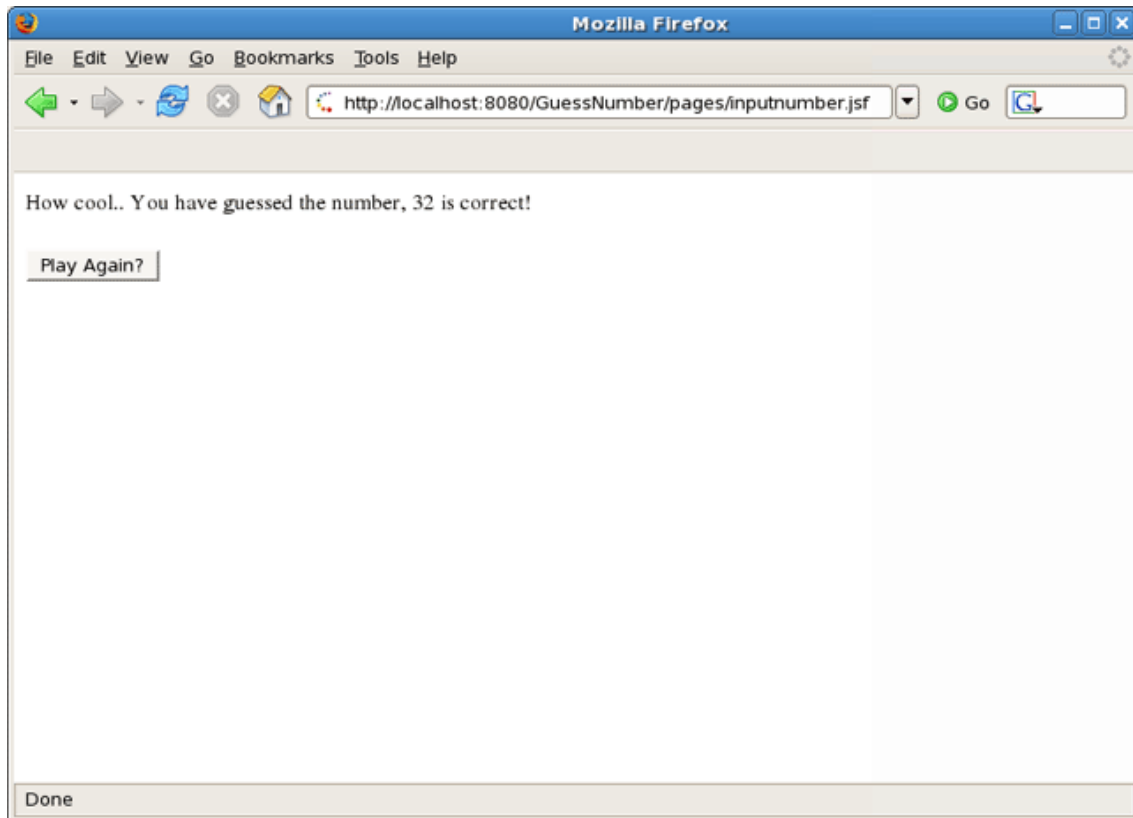


Figure 5.25. Your Guess is Correct

Project Examples

JBoss Developer Studio provides an option to download and import a ready-made project that you can explore and derive some useful technology implementation lessons from.

To adjust the settings of the Project Examples feature you need to navigate to [Windows > Preferences > JBoss Tools > Project Examples](#).

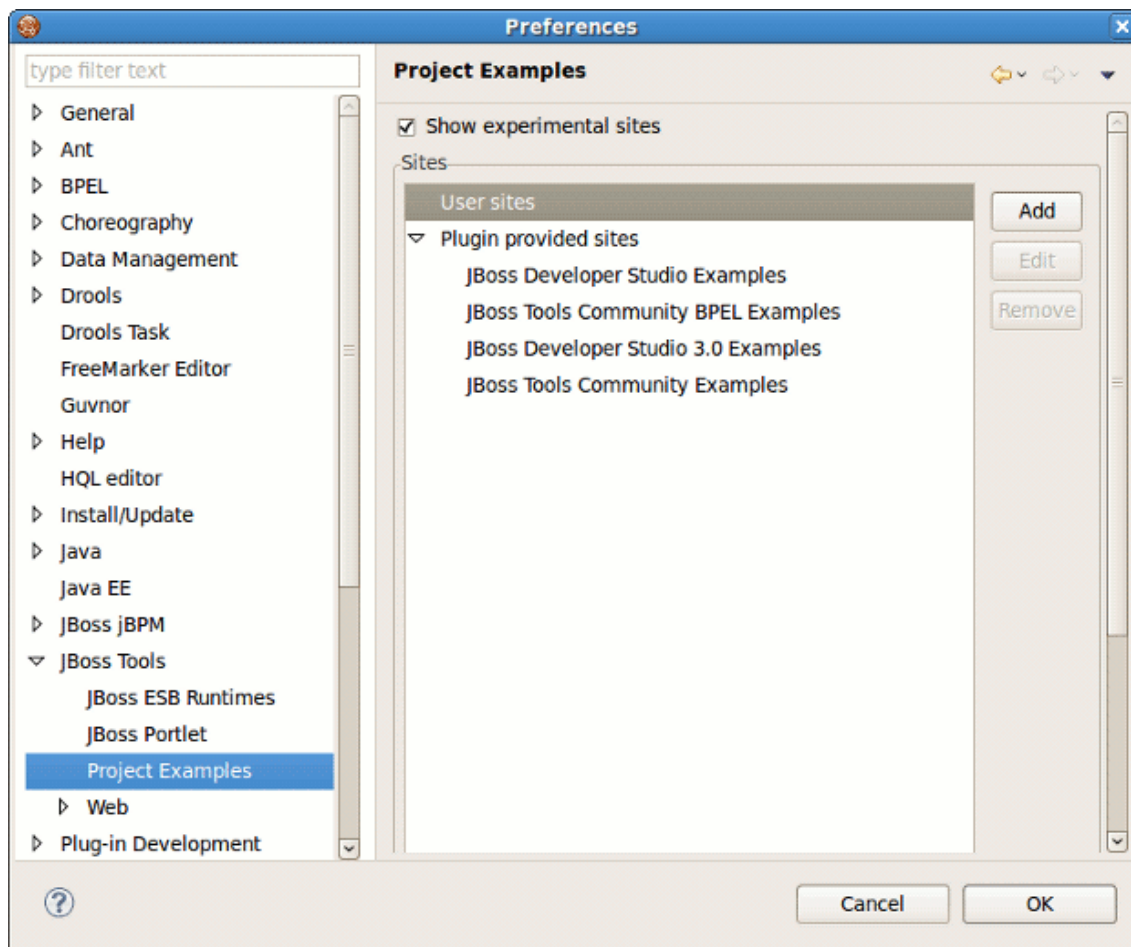


Figure 6.1. Project Examples Preferences

The [Show experimental sites](#) checkbox serves to enable/disable representing the 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 2 input fields: [Name](#) where you need to specify the name of the new entry and [URL](#) that has to point to the xml file that contains example project(s) properties. In more detail the structure of the xml file is discussed further in the chapter of the 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>
```

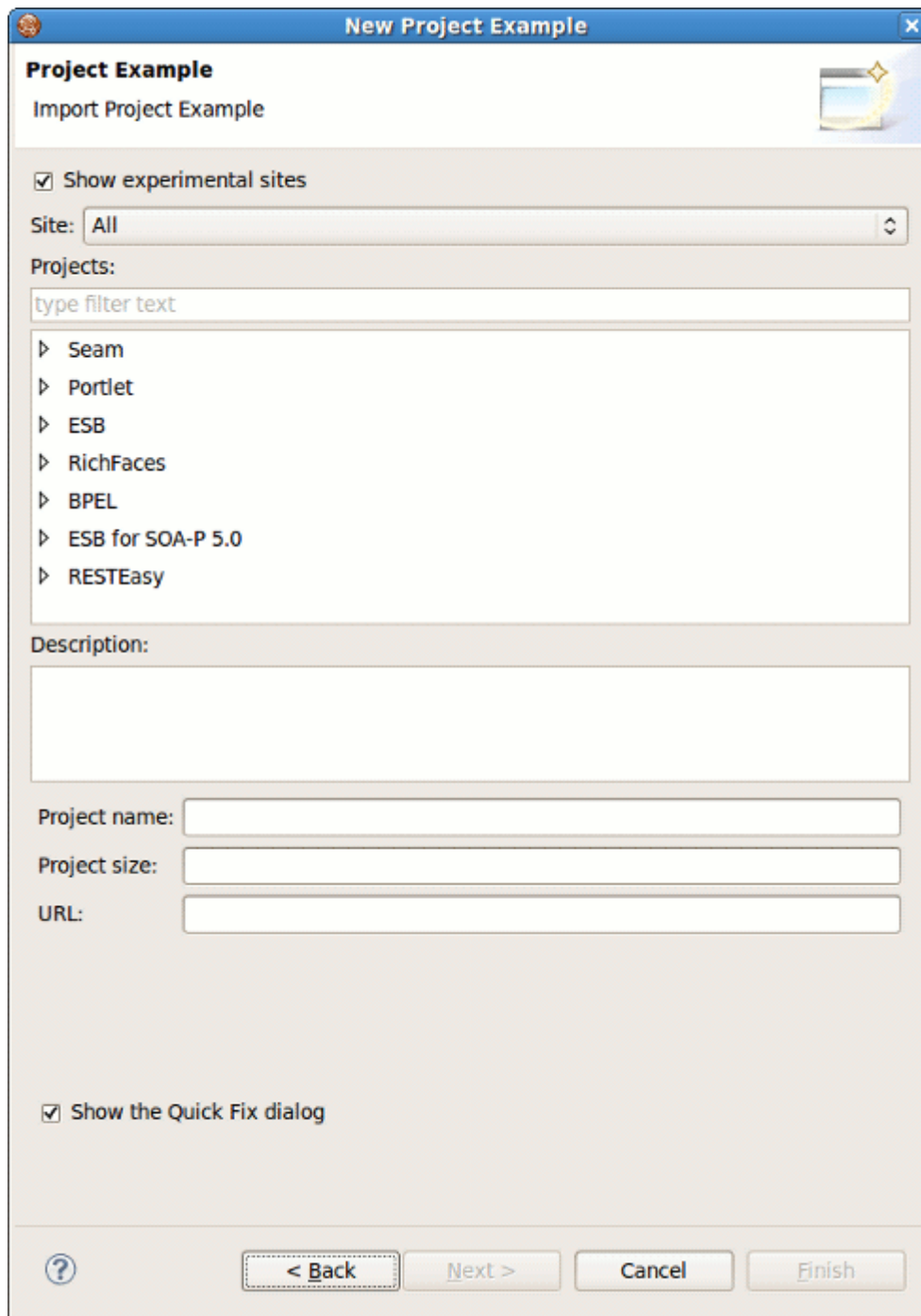
Once you define the location of the xml file with projects settings you will see a new user site entry added. Please note now if you select the entry you can edit and remove it with the corresponding buttons to the right. You can not do such operations with the [Plugin provides sites](#).

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

6.2. Downloading a Project Example

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

- Go to the menu bar and select [File > New > Other...](#)
- Select [Jboss Tools > Project Examples](#) (You can also call the [Project Examples](#) from menu bar: [Help > Project Examples...](#) or directly from [File > New > Example...](#) menu)

**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](#), click [Next](#).

- Now in the [New Project Example](#) dialog you can select a project you would like to explore and a site to download it from

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.

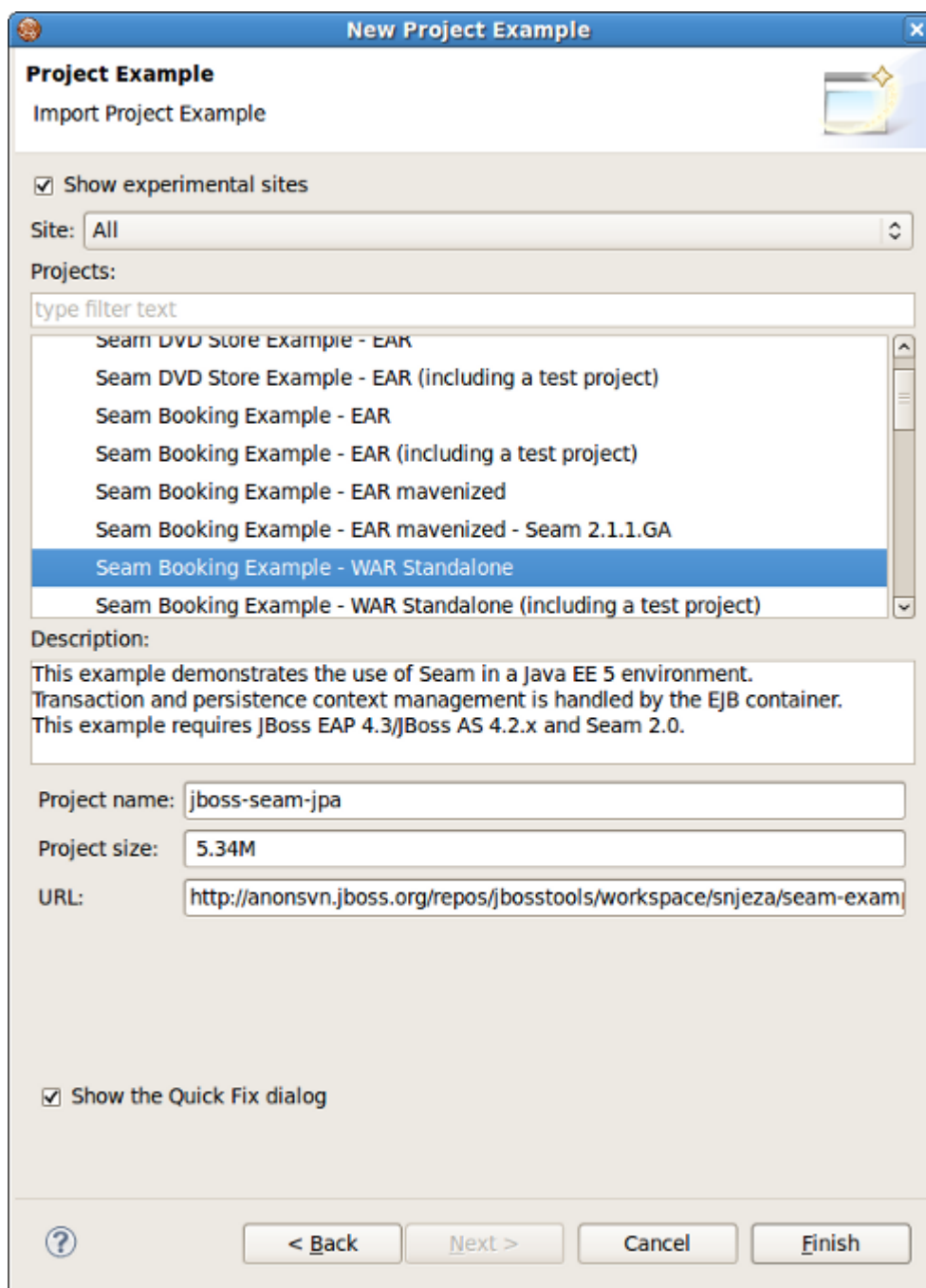


Figure 6.3. Selecting a Project Example

Some project examples have dependencies which could not be automatically configured. In such cases you will receive the message prompted about detected requirements (see the figure below).

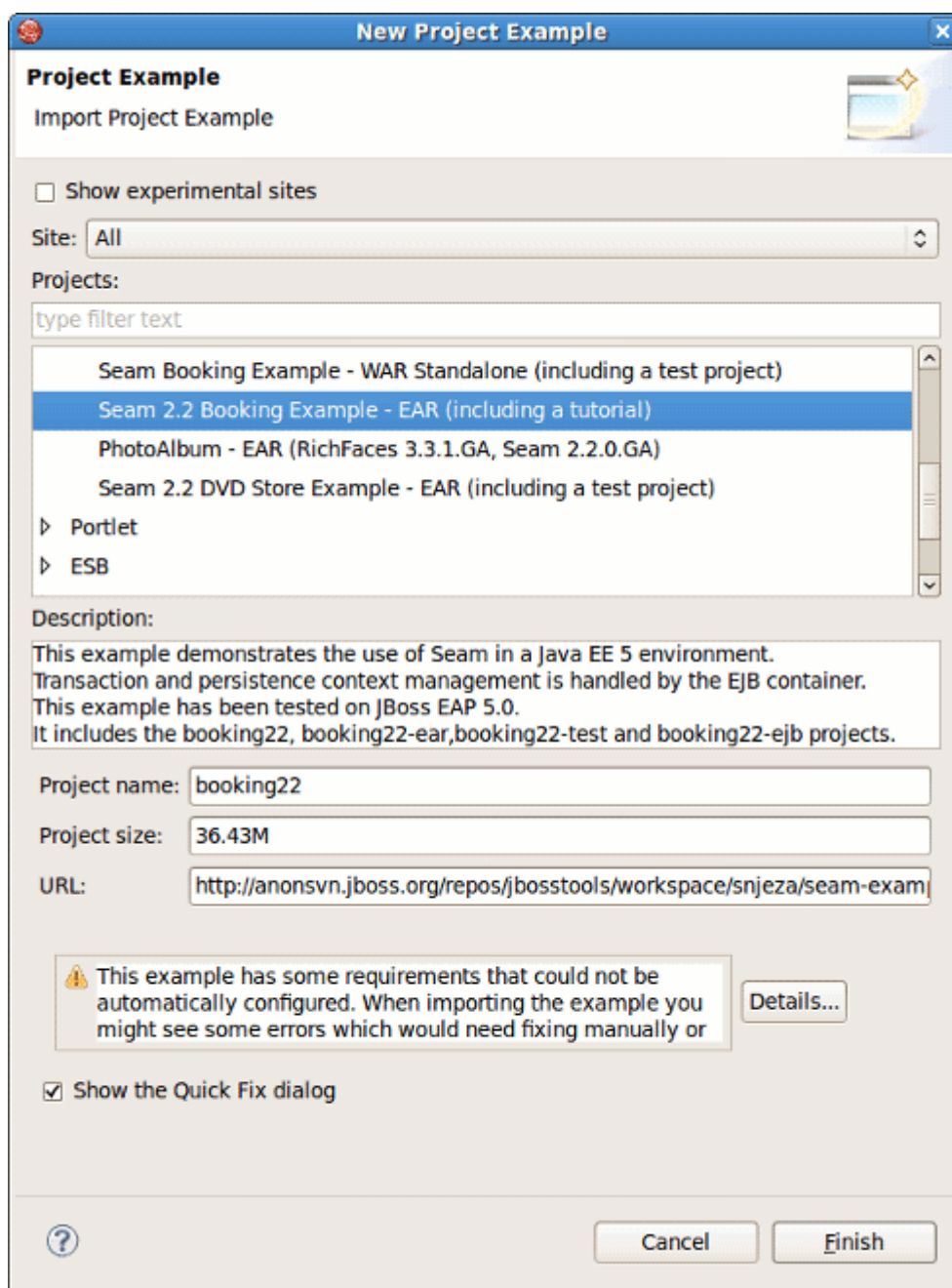


Figure 6.4. Prompt about Project Example Requirements

You should click [Details](#) to see the list of requirements and make fixes.

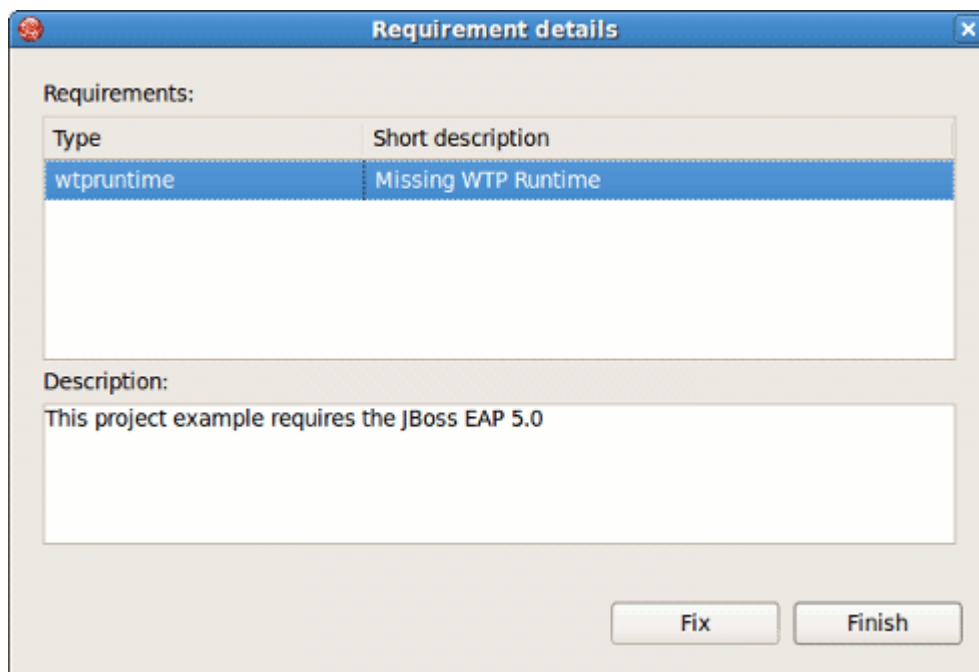
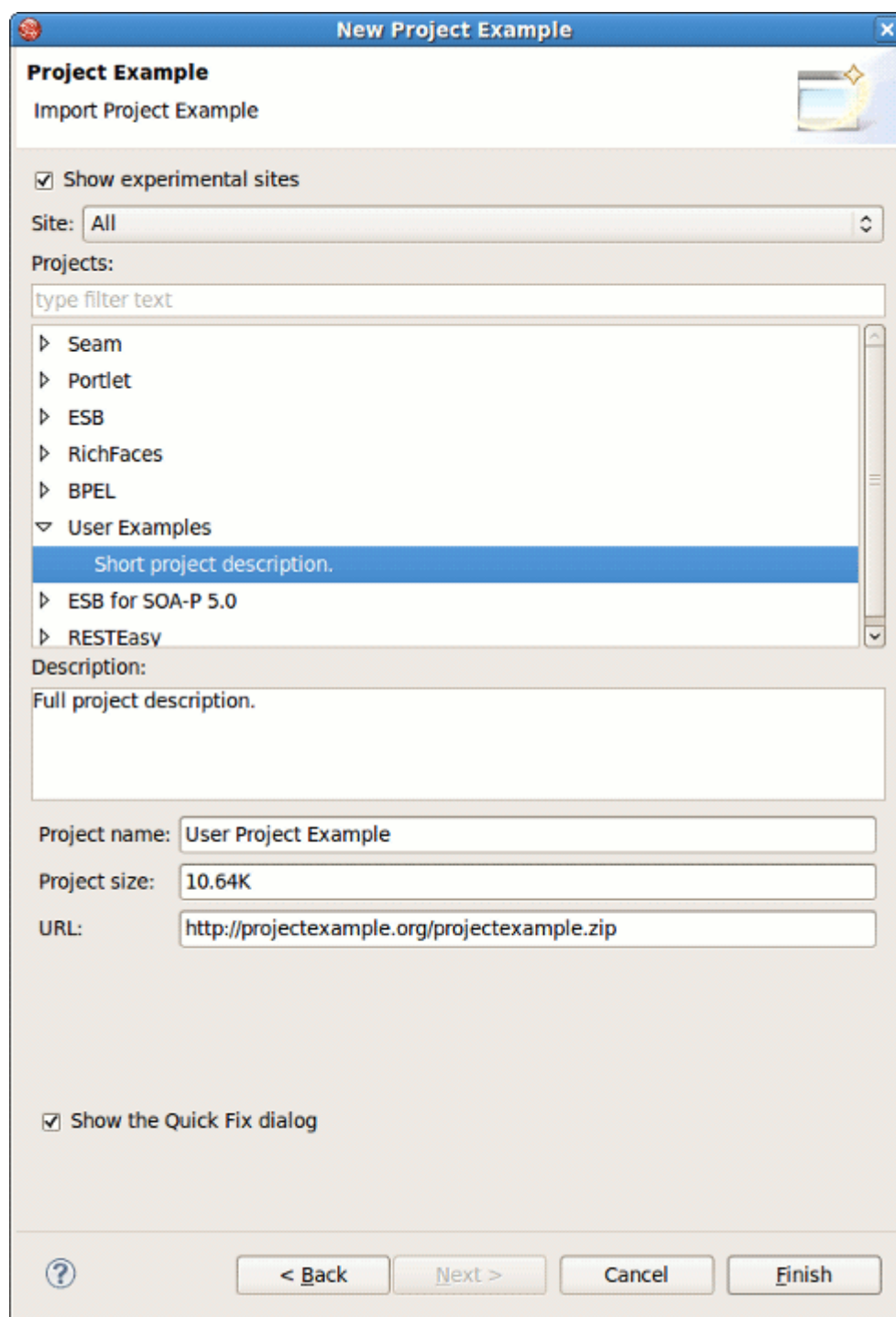


Figure 6.5. Requirements Details Dialog

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 6.6. 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 [Finish](#) 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.

Now you can run the application on the server.

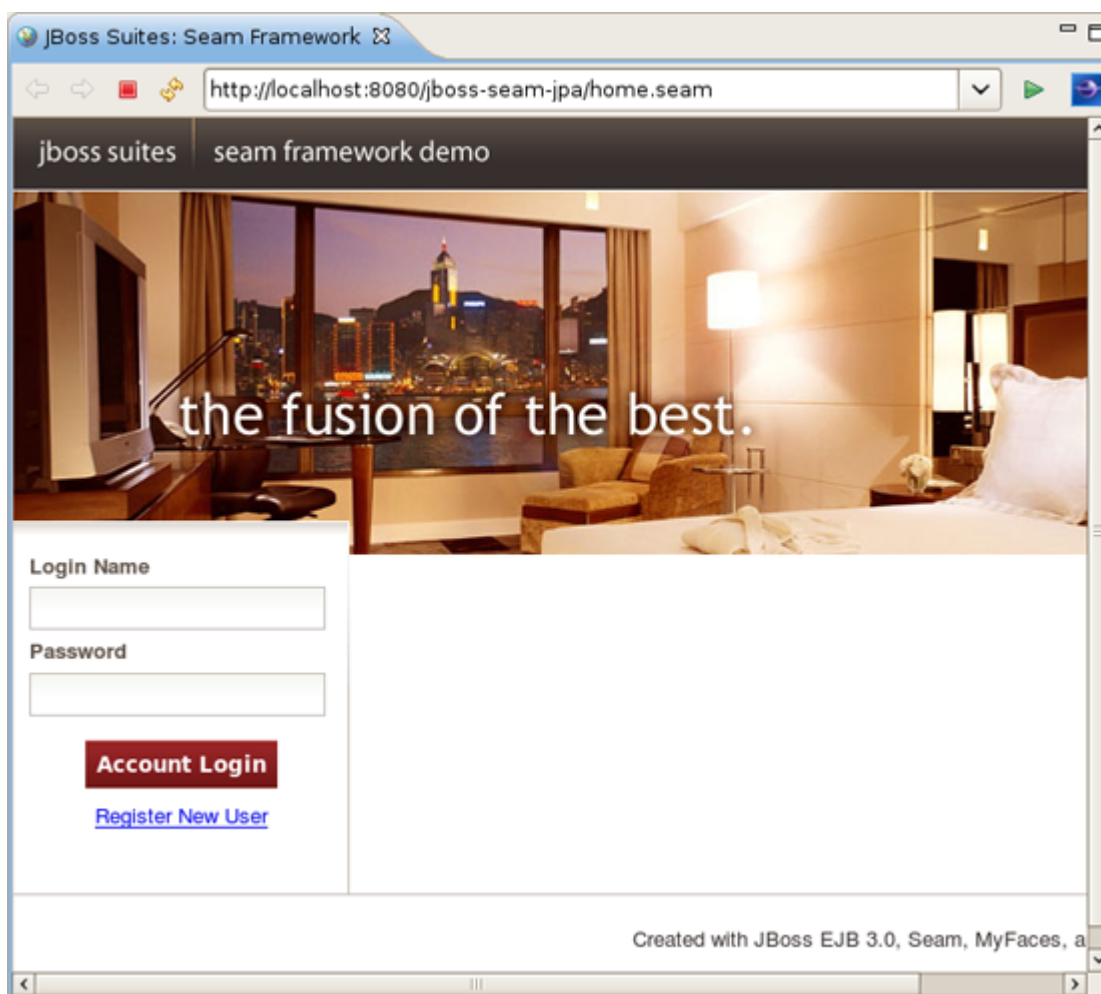


Figure 6.7. Seam Demo Application run on the Server

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>  
</buildCommand>
```

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

It is needed for correct work of [Code Assist](#) and [JSF EL Validation](#).

6.3. Quick Fixes

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 quick fixing 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 there are some you will see a dialog listing the problems.

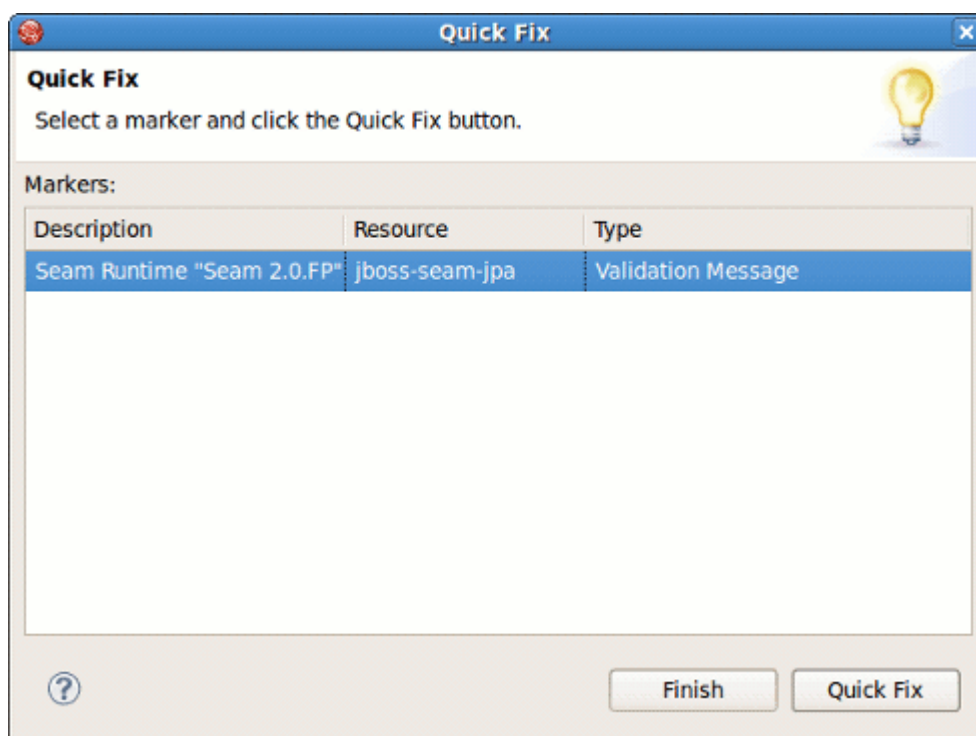


Figure 6.8. Quick Fix Dialog box

To fix the problem you need to:

- Select the problem from the list

- Click [Quick Fix](#) button

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

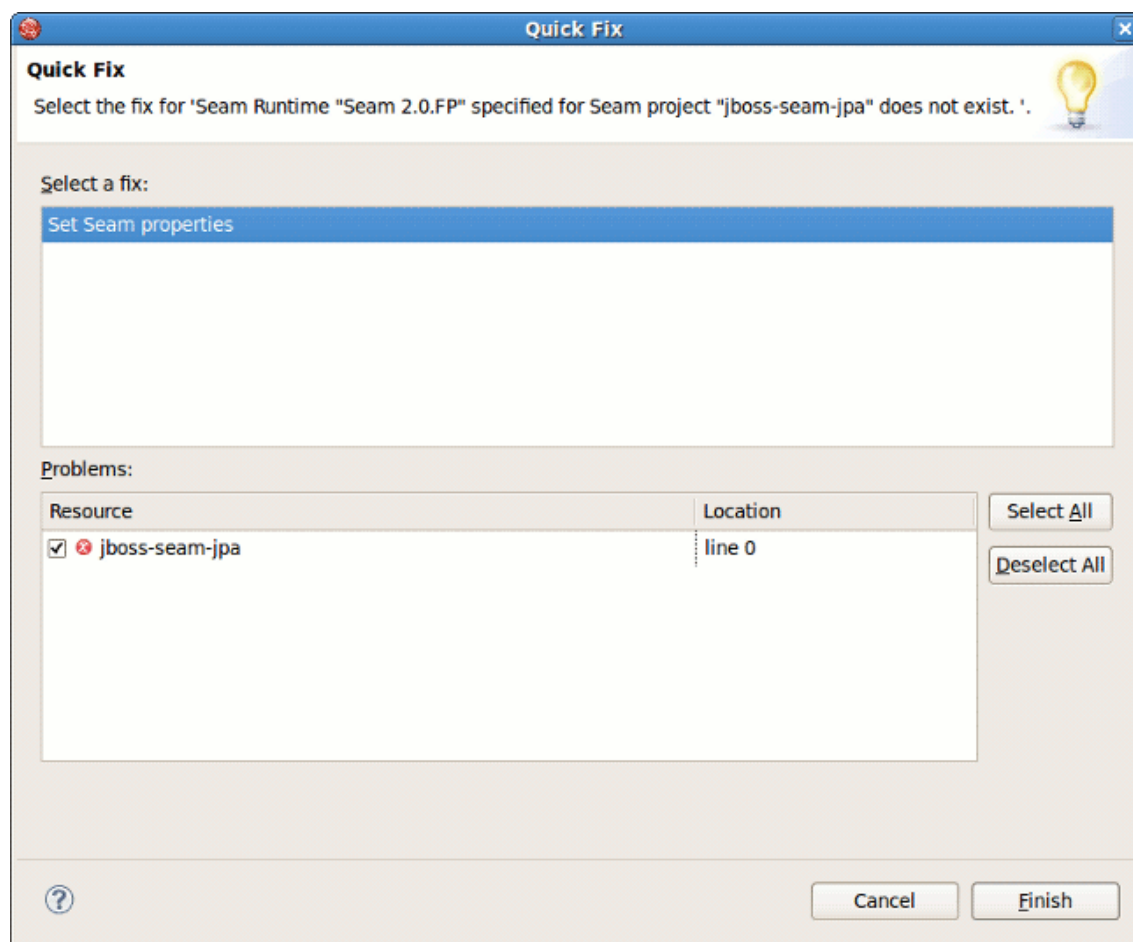


Figure 6.9. Quick Fix Dialog box: Selecting a Fix

In this case (see the image above), when the [Finish](#) button is pressed, Seam Settings dialog box will be displayed where you need to provide a path to the Seam environment 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.

There is a possibility to fix problems before downloading. When the project example is selected you will see warning message on the [New Project Example](#) dialog.

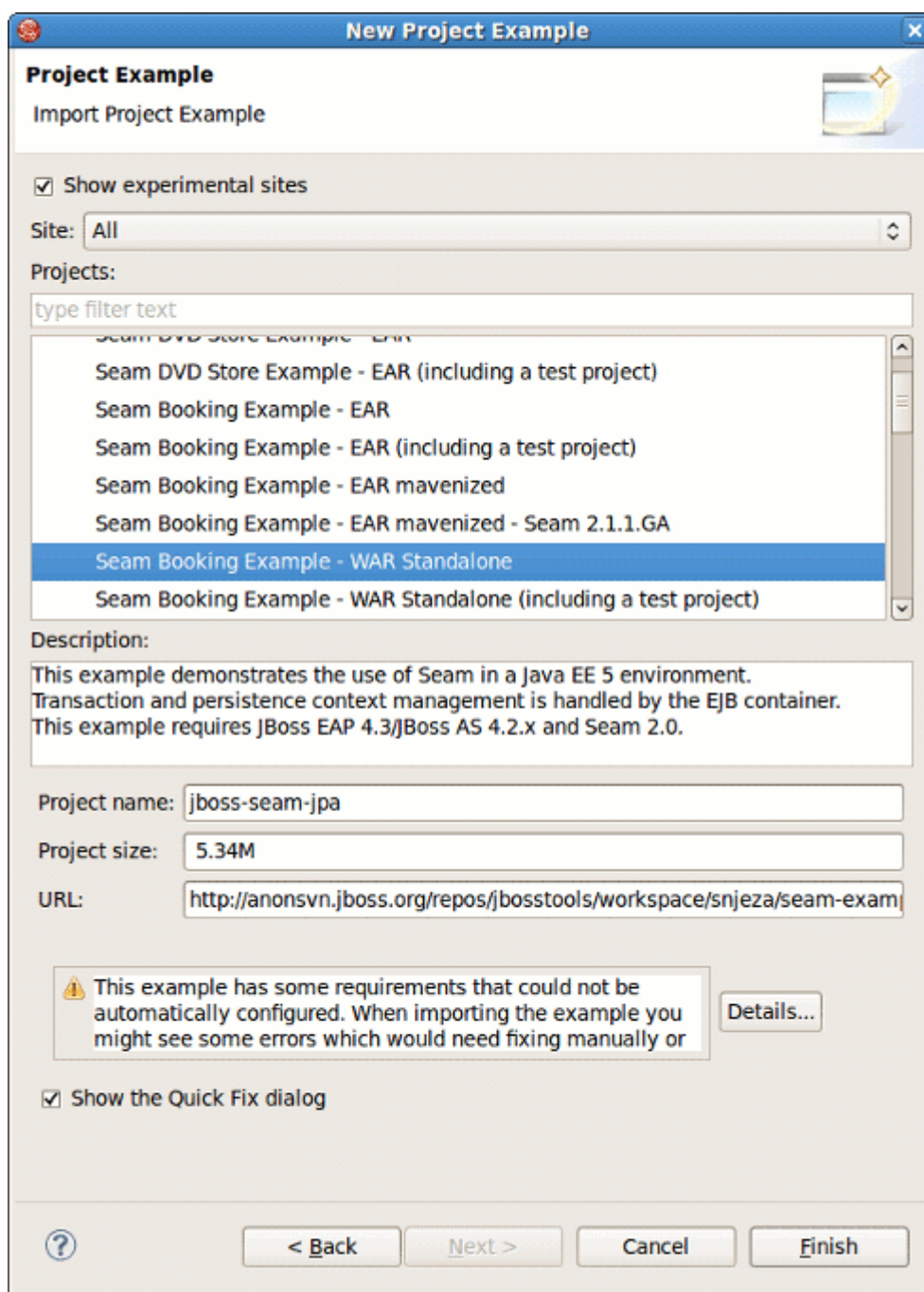


Figure 6.10. Requirements warning in the Project Example wizard

To fix the problem immediately you need to:

- Click [Details...](#) button in the [New Project Example](#) dialog
- Select the problem from the list in the [Requirement details](#) dialog box
- Click [Fix](#) button

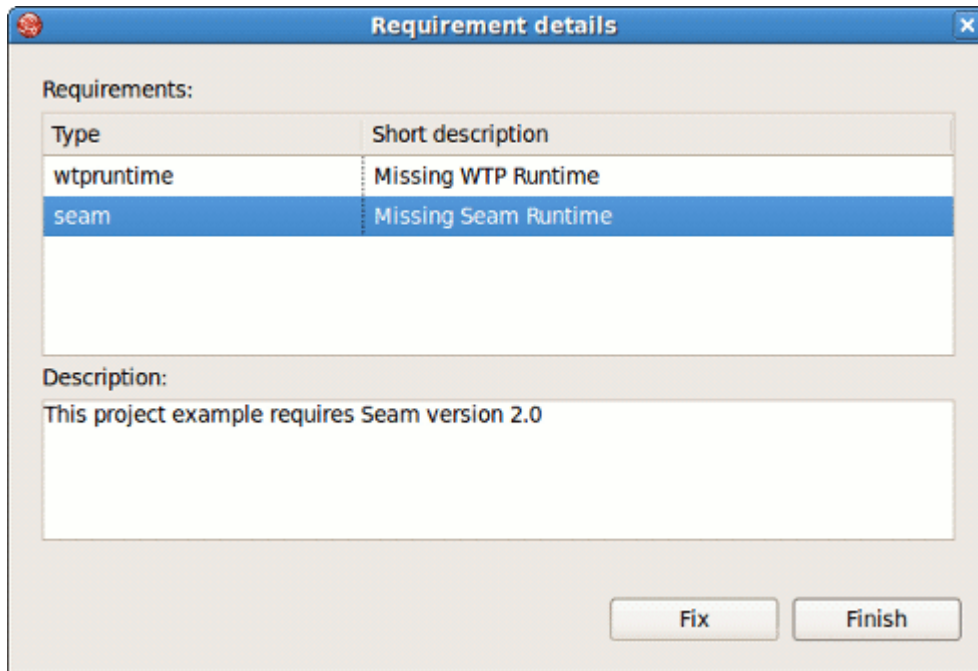


Figure 6.11. Requirement Details Dialog box: Selecting a Fix

You will be offered a solution to the problem.

FAQ

For more information on [JBoss Developer Studio](#) features, refer to the following FAQ to get the answers on the most "popular" questions.

7.1. What should I do if 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 `xpLib.i386` package

2. Type

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

3. and/or use

```
yum install libXp
```

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

5. If it doesn't help and you use Fedora Core Linux and Eclipse Version: 3.4.1, the issue can be produced because `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_64_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 `eclipse.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 Eclipse log file, `<workspace>\.metadata\log`
- Start Eclipse with the `-debug` option:

```
eclipse -debug
```

- Post the Eclipse log file (`<workspace>\.metadata\log`) on the forums.

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

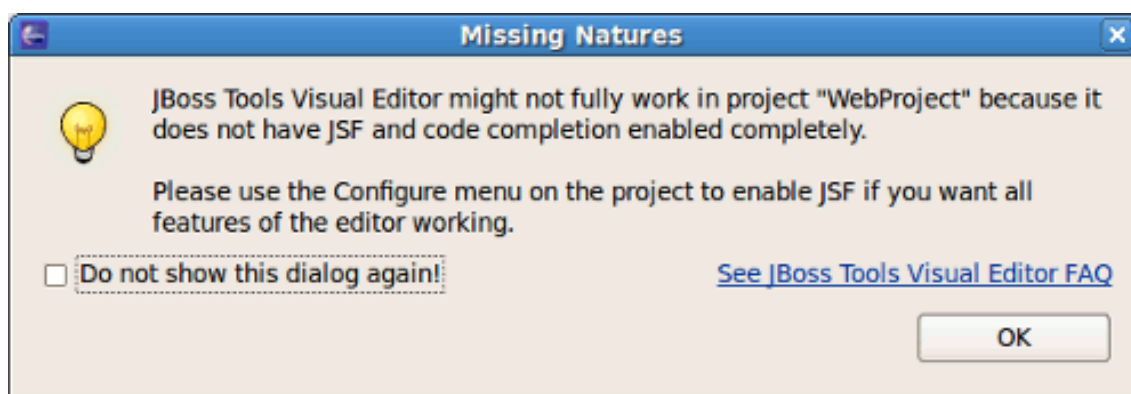


Figure 7.1. Missing Nature

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 shure 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 on this [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/import the project to a JBDS Seam project?

We highly recommend you to create Seam 1.2.1 project using the [JBDS](#). In other case try to do manually:

- 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 JBDS?

Yes. From main menu select [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](#) preconfigures eclipse via the eclipse.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:

- Go to [New > Other > Java Project from Existing Buildfile](#)
- Point to the [build.xml](#) of any chosen project by pressing [Browse](#) button
- Hit [Finish](#) 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 go to [Properties > Seam Settings](#).

7.11. Is a cross-platform project import possible for JBDS?

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, go to [Menu > Import > General > Existing Projects into Workspace](#), select the folder where your project stored and press [Finish](#).

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 JBDS 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 both 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 using the JBoss Developer Studio.

- **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 in JBoss Developer Studio.

- **Exadel Studio Migration Guide**

This document is intended to help you to migrate an existing Exadel JSF or Struts projects from Exadel Studio into JBoss Developer Studio.

If there's anything we didn't cover in this guide, please feel free to visit our [JBoss Developer Studio Users Forum](http://www.jboss.com/index.html?module=bb&op=viewforum&f=258) [http://www.jboss.com/index.html?module=bb&op=viewforum&f=258] or [JBoss Tools Users Forum](http://www.jboss.com/index.html?module=bb&op=viewforum&f=201) [http://www.jboss.com/index.html?module=bb&op=viewforum&f=201] to ask questions. There we are also looking for your suggestions and comments.