

# **Getting Started with JBoss Developer Studio**

**Version: 1.0.0.GA**

---

---

---

<b>1. Getting Started with JBoss Developer Studio .....</b>	<b>1</b>
1.1. What is JBDS? .....	1
1.2. Configuring Your Java Environment .....	1
1.2.1. Installing and Configuring 32-bit Sun JDK 5.0 on Linux .....	1
1.2.2. Installing and Configuring 32-bit Sun JDK 5.0 on Microsoft Windows .....	3
1.3. JBoss Developer Studio Installation .....	4
1.3.1. Installing from the downloaded version .....	4
1.3.2. What is the difference between JBoss Developer Studio and JBoss Tools... ..	7
1.4. Welcome to JBoss Developer Studio .....	9
1.5. Upgrading .....	13
1.6. Uninstalling .....	13
1.7. Support .....	13
1.8. FAQ .....	13
1.8.1. Installation Issues .....	13
1.8.2. Importing Projects .....	14
1.8.3. Troubleshooting, Problems, Configuration, Error Messages .....	15
1.9. Other relevant resources on the topic .....	15
<b>2. Manage JBoss AS from JBoss Developer Studio .....</b>	<b>17</b>
2.1. How to Manage the JBoss AS Bundled in JBDS .....	17
2.1.1. Starting JBoss server .....	17
2.1.2. Stopping JBoss Server .....	19
2.1.3. Server Container Preferences .....	19
2.2. How to Use Your Own JBoss AS Instance with JBDS .....	21
2.2.1. JBoss AS Installation .....	21
2.2.2. Adding and configuring JBoss server .....	22
<b>3. Write Your First Seam Web Application .....</b>	<b>31</b>
3.1. Create a Seam Project .....	31
3.2. Build and Deploy the Seam Application .....	40
3.3. Add a Web Page and an Action .....	43
3.4. Input Validation .....	47
3.5. Add a new UI Component .....	48
3.6. Add Security to the Application .....	50
3.7. Other relevant resources on the topic .....	51
<b>4. Developing a simple JSP web application .....</b>	<b>53</b>
4.1. Setting Up the Project .....	53
4.2. Creating JSP Page .....	55
4.2.1. Editing a JSP Page .....	56
4.2.2. web.xml file .....	58
4.2.3. Deploying the project .....	60
4.2.4. JSP Page Preview .....	63
4.2.5. Launch JSP Project .....	63
<b>5. RAD development of a simple JSF application .....</b>	<b>65</b>
5.1. Setting up the project .....	65
5.2. Creating JSP Pages .....	66

5.3. Creating Transition between two views .....	68
5.4. Creating Resource File .....	69
5.5. Creating Java Bean .....	71
5.6. Editing faces-config.xml File .....	75
5.7. Editing the JSP View Files .....	76
5.7.1. Editing inputnumber.jsp page .....	76
5.7.2. Editing success.jsp page .....	83
5.8. Creating index.jsp page .....	85
5.9. Running the Application .....	85
<b>6. Further Reading .....</b>	<b>89</b>

# 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.

## 1.2. Configuring Your Java Environment

You must have a working installation of JDK 5 before you install JBoss Developer Studio. Currently it will only fully work with a 32-bit JVM, not a 64-bit JVM. On a 64-bit JVM the visual editor fails to launch. 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.

### 1.2.1. 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 "RPM 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:

When you add `$JAVA_HOME/bin` to `$PATH`, you should add it **before** the old `$PATH` not after it. 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.

### 1.2.1.1. 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)
```

## 1.2.2. 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 *jre\bin* directory to your path, for example:

```
C:\Program Files\Java\jdk1.5.0_11\jre\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.

### 1.3. JBoss Developer Studio Installation

This chapter will provide you with detailed information on how to install JBoss Developer Studio and all the JBoss Tools modules.

#### 1.3.1. Installing from the downloaded version

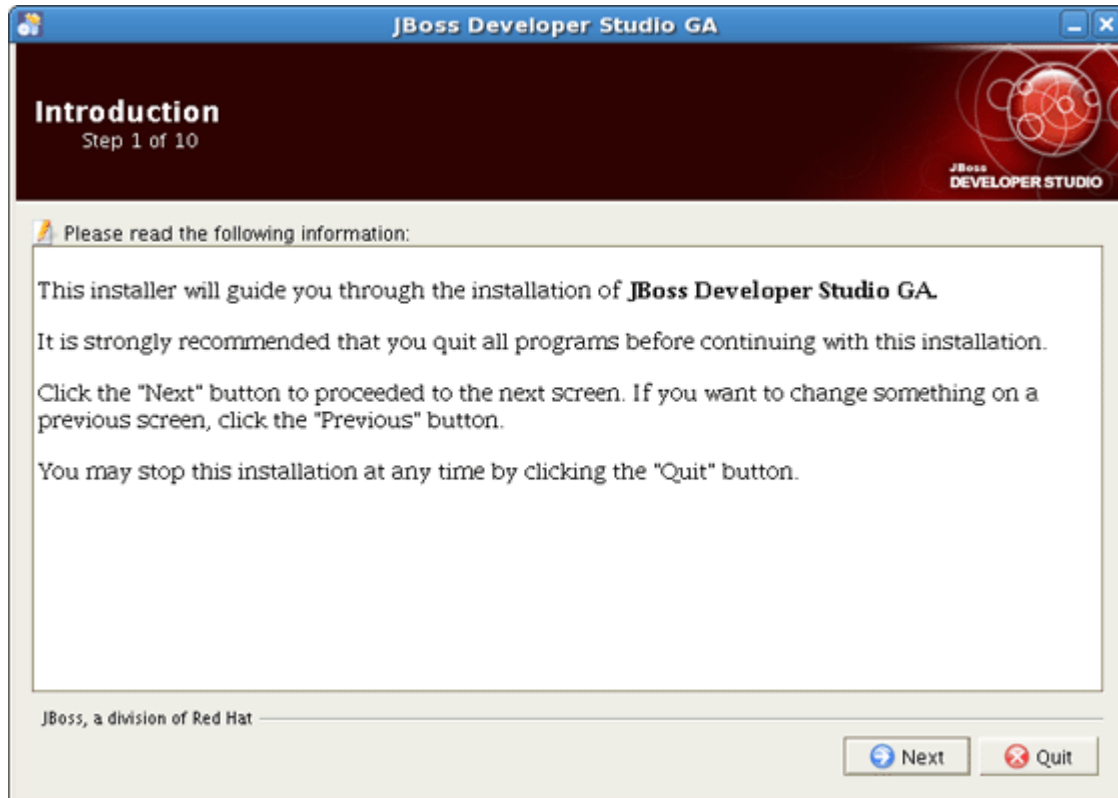
Let's start with the JBDS installation.

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:

- Download the appropriate installation file for your platform from [Red Hat website](http://www.jboss.com/products/devstudio) [http://www.jboss.com/products/devstudio].
- Run in console:

```
java -jar jbdevstudio-linux-gtk-1.0.0.GA.jar
```

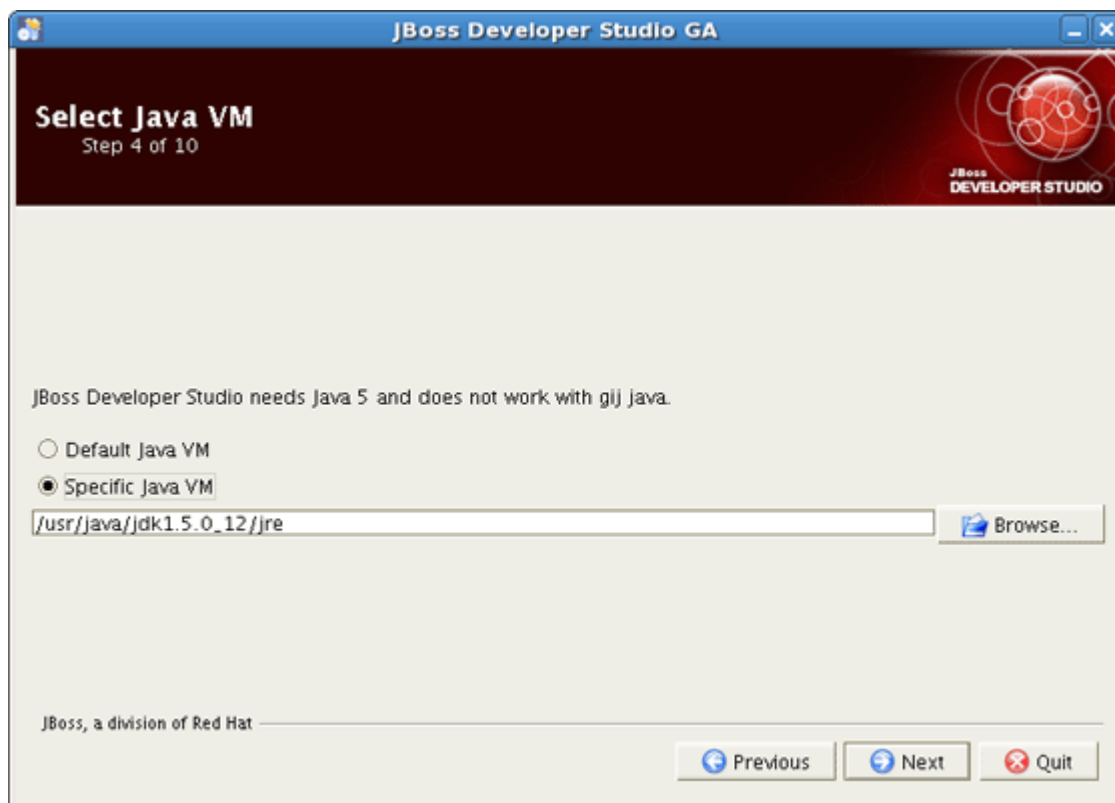
- Follow the instructions presented by the installation wizard



**Figure 1.1. JBoss Developer Studio Installation Wizard**



- Provide the installation path
- Select Java VM



**Figure 1.2. Select Java VM**

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 needs Java 5 and doesn't work with gjc Java.

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



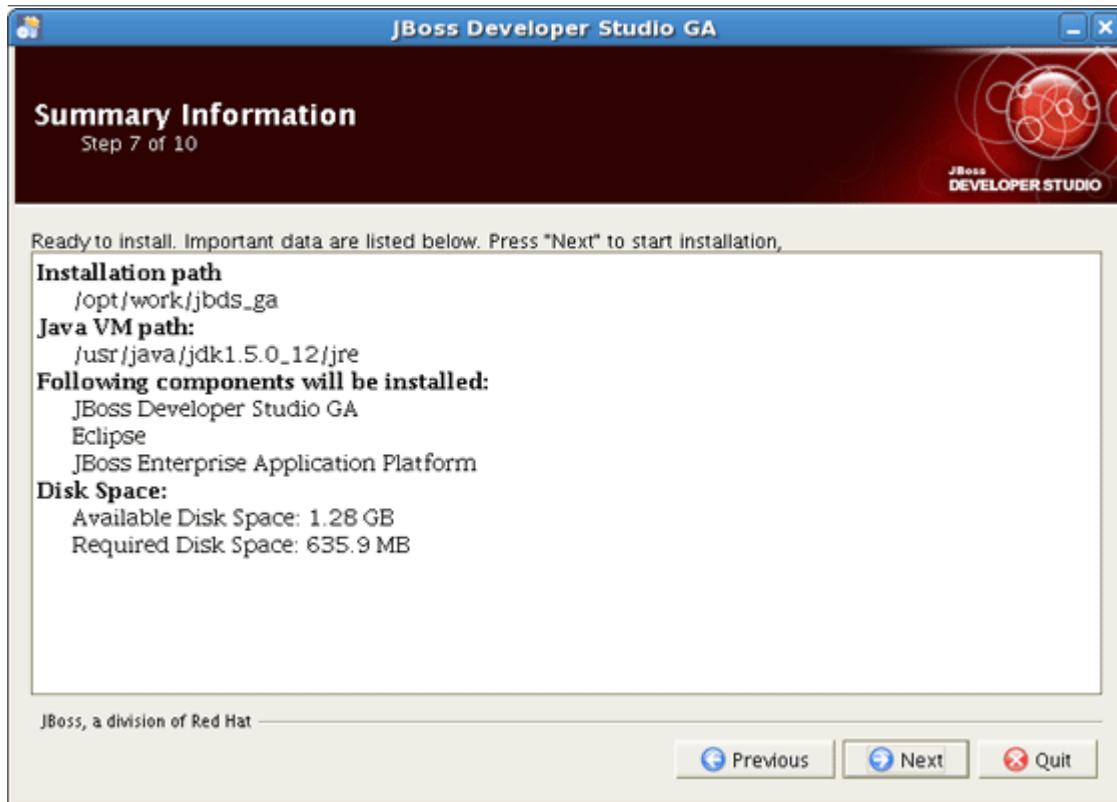
**Figure 1.3. JBoss Enterprise Application Platform Installing**



**Note:**

The installer installs JBoss Enterprise Application Platform for running your applications if you select this option during the installation process. If you want to use a different server than ours, you can change the setting in JBoss Developer Studio.

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



**Figure 1.4. Summary Information**

### 1.3.2. What is the difference between JBoss Developer Studio and JBoss Tools

This release of JBoss Tools is what went into our JBoss Developer Studio which provides 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]

### 1.3.2.1. JBoss Tools Installation

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

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

- Get Eclipse 3.3.1 and Web Tools 2.0.1

The quickest way to get a WTP version is to download "Eclipse IDE for Java EE Developers" via [www.eclipse.org](http://www.eclipse.org) [http://www.eclipse.org].



#### Note:

Remember to choose the download that matches your OS and use Java 5 when you run it.

- Get the [JBoss Tools build](http://labs.jboss.com/tools/download/index.html) [http://labs.jboss.com/tools/download/index.html]

You can also find the latest development release of JBossTools from [JBossTools Stable Update Site](http://download.jboss.org/jbosstools/updates/stable/) [http://download.jboss.org/jbosstools/updates/stable/]

- Finally, install the build

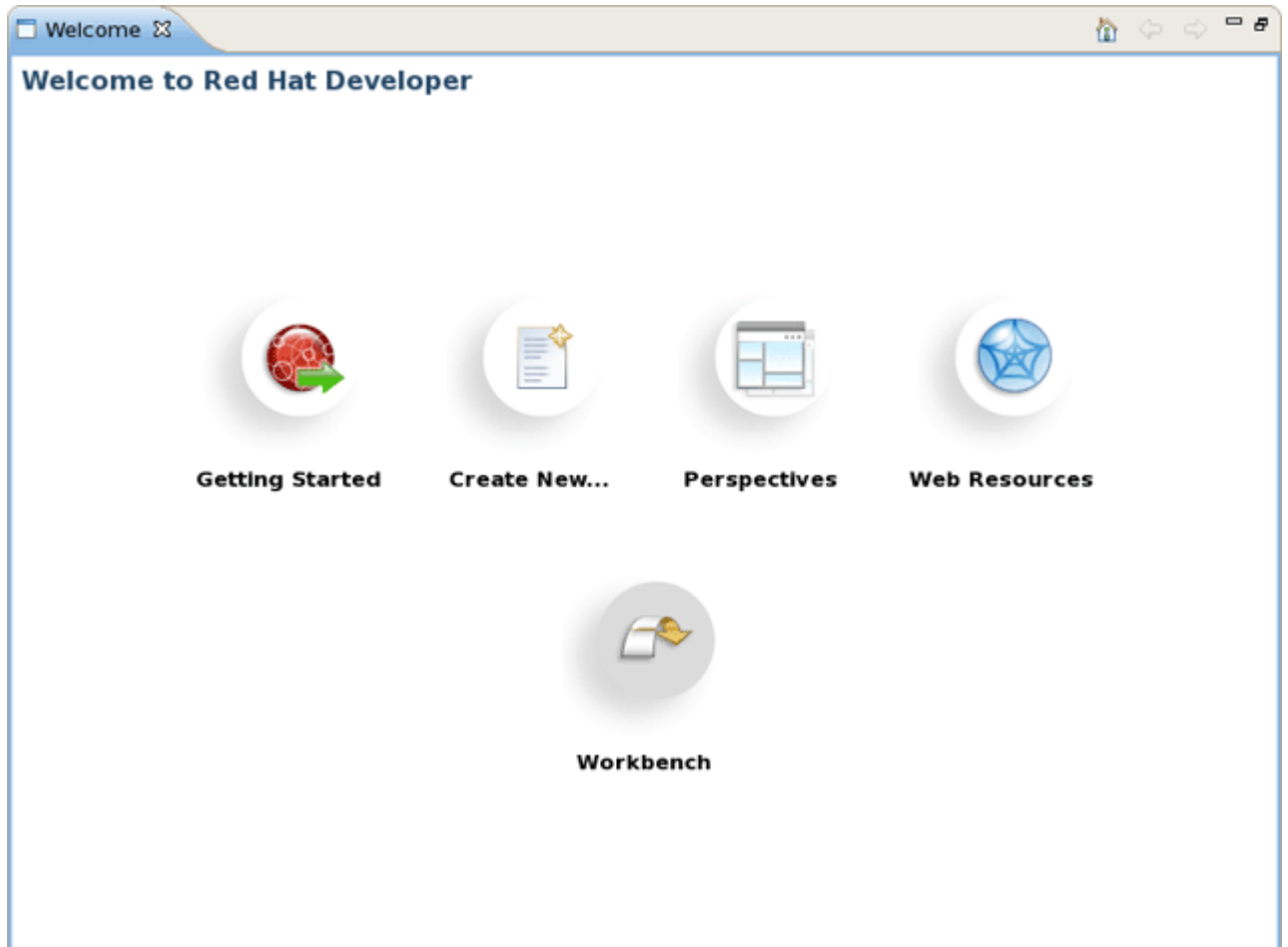
Unzip the file(s) directly into your Eclipse *plugins/features* directory and it will be readily available. It might be necessary to start Eclipse with `eclipse -clean` to make sure it starts clean and rereads the new list of plugins.

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

## 1.4. Welcome to JBoss Developer Studio

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

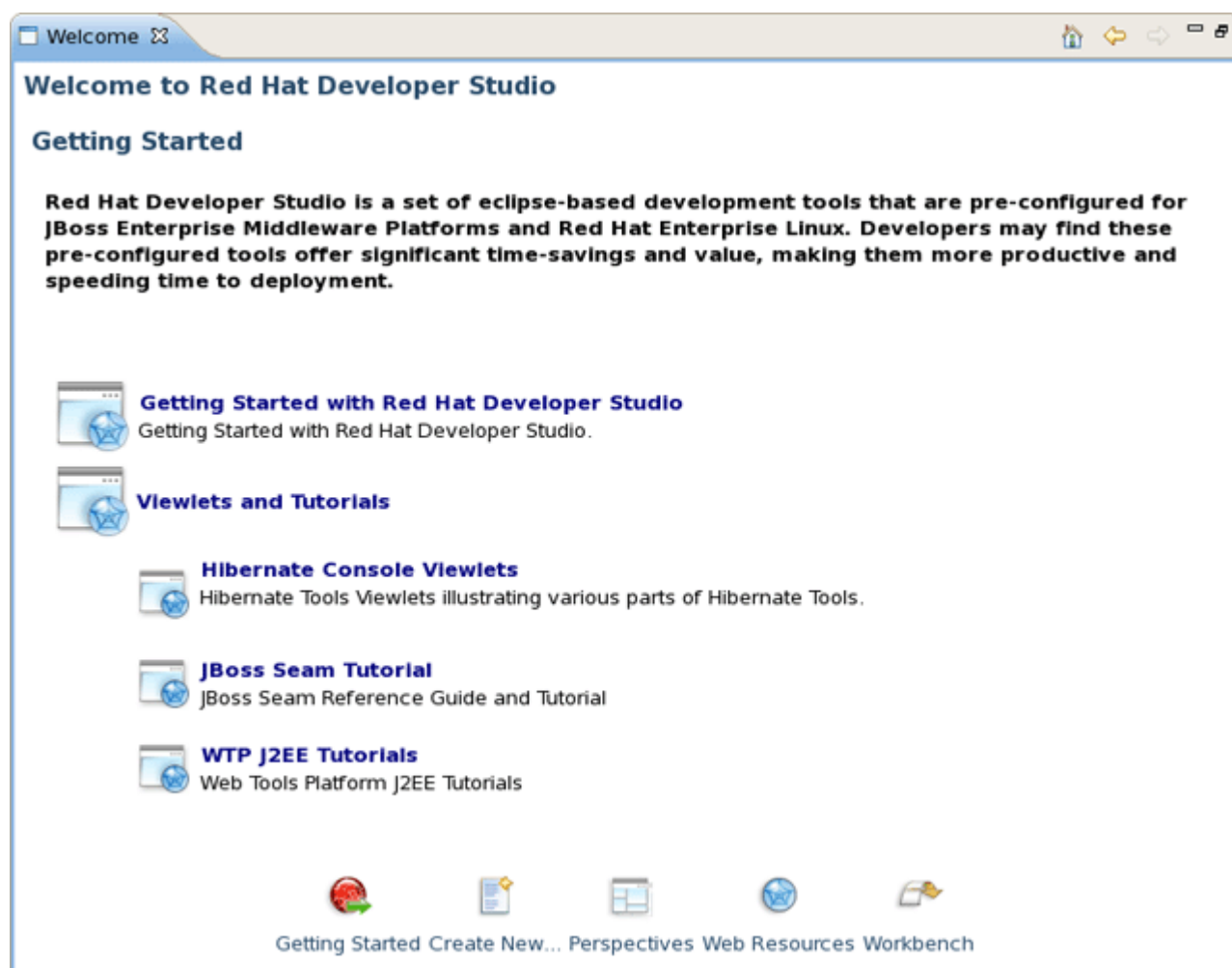
The welcome page is the first page you see when you first launch JBoss Developer Studio.



**Figure 1.5. 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 1.6. Getting Started Documentation**

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

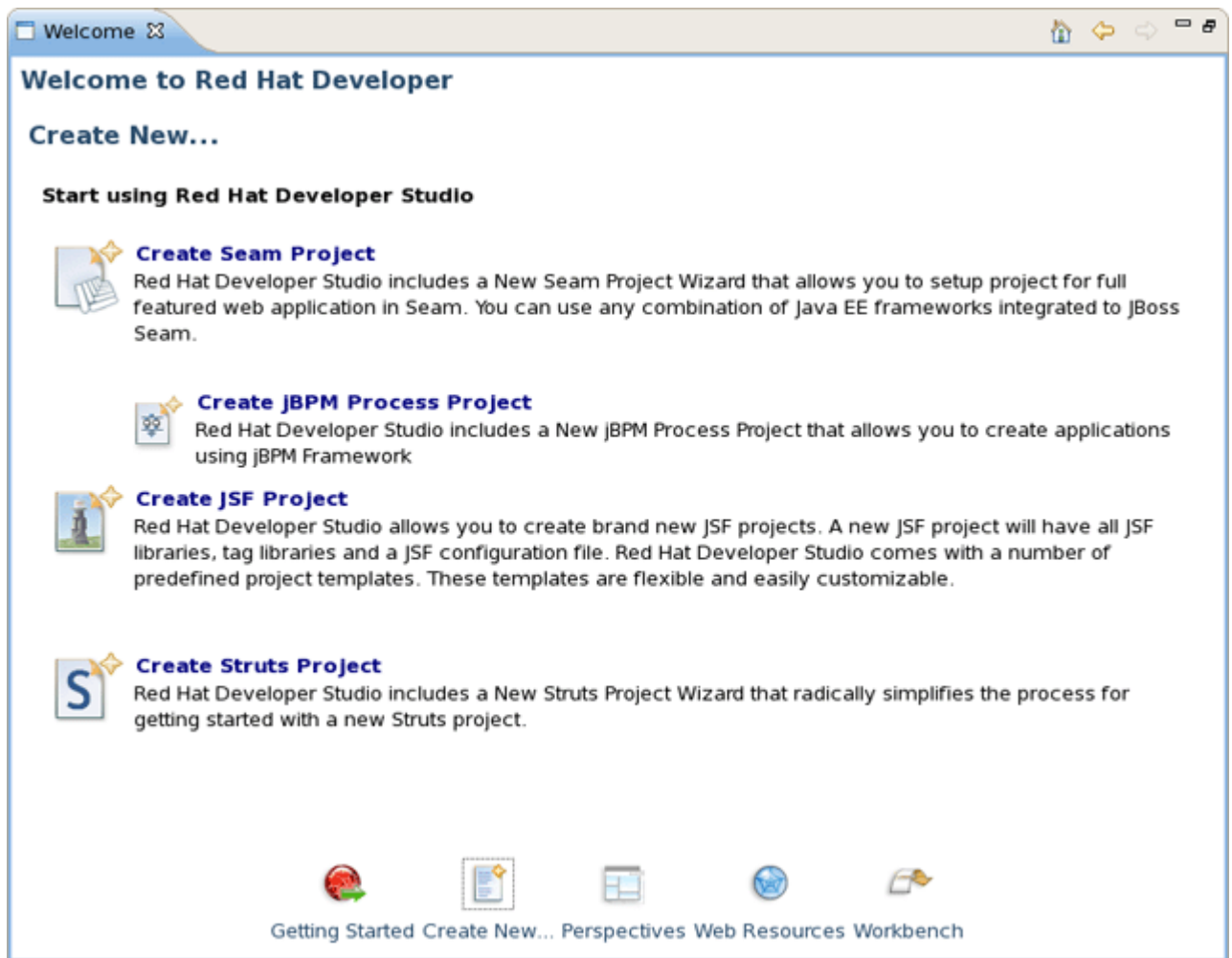


Figure 1.7. Create New...

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

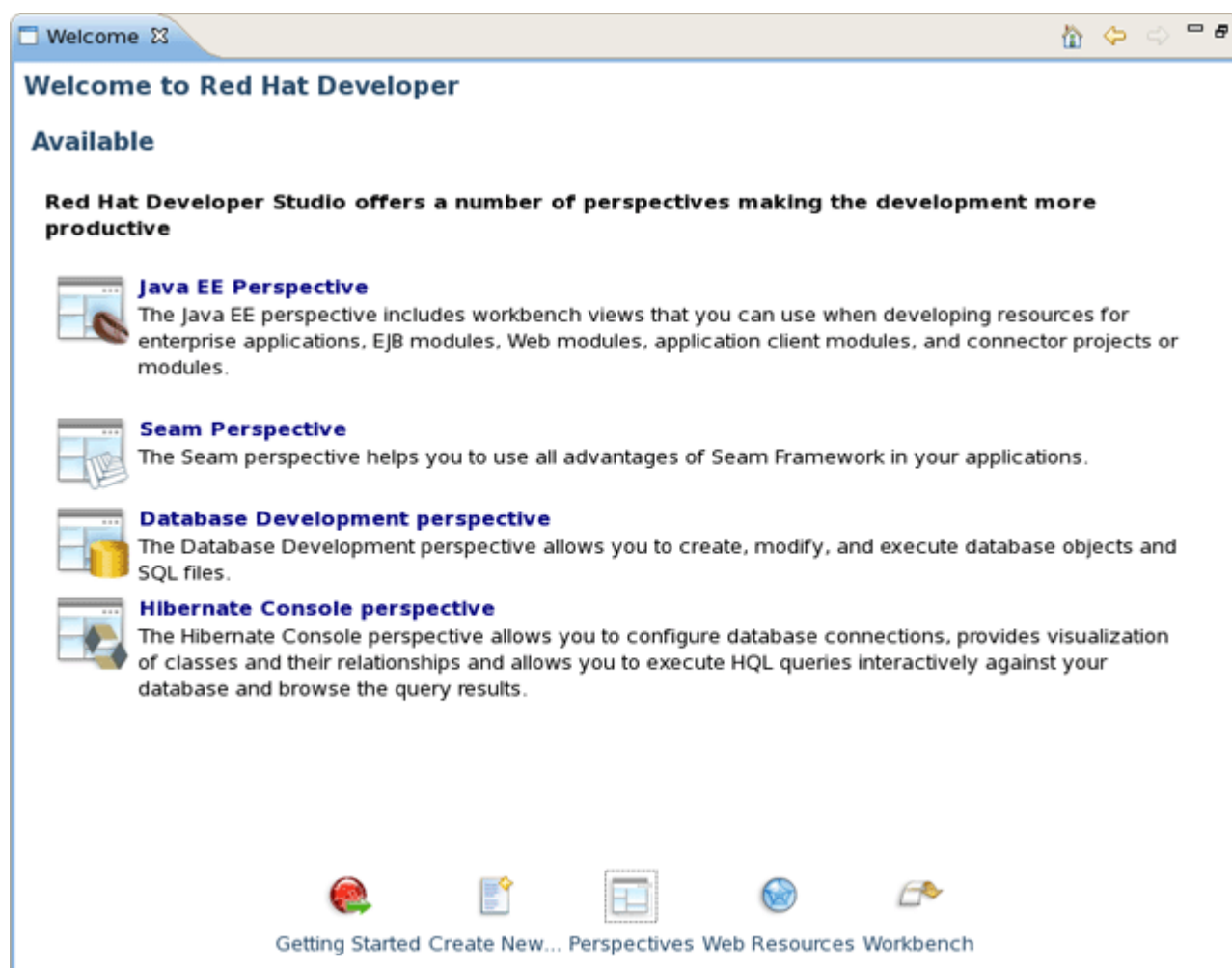


Figure 1.8. Perspectives

- to visit JBoss Developer Studio web resources.



Figure 1.9. Web Resources



Start work with JBoss Developer Studio by clicking on *Workbench* button or simply close the welcome page.

## 1.5. Upgrading

To upgrade, just uninstall your current version and install the new one.

## 1.6. Uninstalling

- Make sure JBoss Developer Studio is not running
- Uninstall your current version of JBoss Developer Studio by running uninstaller

## 1.7. 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

## 1.8. FAQ

For easy reference to JBoss Developer Studio related questions, our FAQ provides answers to the most "popular" questions. The sections of questions are organized by type.

### 1.8.1. Installation Issues

#### **Visual Editor does not start under Linux**

Linux users may need to do the following to get the visual editor to work correctly on their machines.

1. On Red Hat based Linux distributions install the `xpLib.i386` package
2. Type

```
In -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 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.

### 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.

## 1.8.2. Importing Projects

### 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.

---

### **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.

### **Can I import a .war file?**

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

## **1.8.3. Troubleshooting, Problems, Configuration, Error Messages**

### **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
```

### **How can I add my own tag library to the JBoss Tools Palette?**

See [Adding Tag Libraries](http://www.redhat.com/developer_studio/guides/jsf/html_single/#AddingJSFCapabilityToAnyExistingEclipseProject) [http://www.redhat.com/developer\_studio/guides/jsf/html\_single/#AddingJSFCapabilityToAnyExistingEclipseProject] in Visual Web Tools Guide.

### **I see the Oracle ADF Faces component library tags in the JBoss Tools Palette, but I can't seem to find the libraries for ADF. How do I use this component library with JBDS?**

See [Adding Support for the Oracle ADF Components Library](http://www.redhat.com/developers/jbds/JSFTools/JavaServerFacesSupport.html#AddingSupportForTheOracleADFComponentsLibraryOrAnyOtherSupport64) [http://www.redhat.com/developers/jbds/JSFTools/JavaServerFacesSupport.html#AddingSupportForTheOracleADFComponentsLibraryOrAnyOtherSupport64] in the JBDS User Guide.

## **1.9. Other relevant resources on the topic**

JBDS on JBoss: [JBoss Developer Studio](http://labs.jboss.com/rhdevstudio/) [http://labs.jboss.com/rhdevstudio/]

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]

Download: [JBDS Download](http://www.jboss.com/products/devstudio) [http://www.jboss.com/products/devstudio]



# 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 ships with JBoss EAP v.4.2 bundled. When you followed the default installation of JBoss Developer Studio, you should already have a JBoss 4.2 server installed and defined. To run JBoss AS 4.2 you need JDK 1.5, JDK 6 is not formally supported yet, although you may be able to start the server with it.

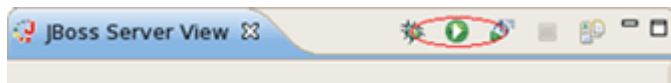
## 2.1. How to Manage the JBoss AS Bundled in JBDS

This section covers the basics of working with the JBoss server supported directly by JBDS via bundled AS plug-in. To read more about AS plug-in, read [Server Manager guide](#) [../as/en/html\_single/index.html].

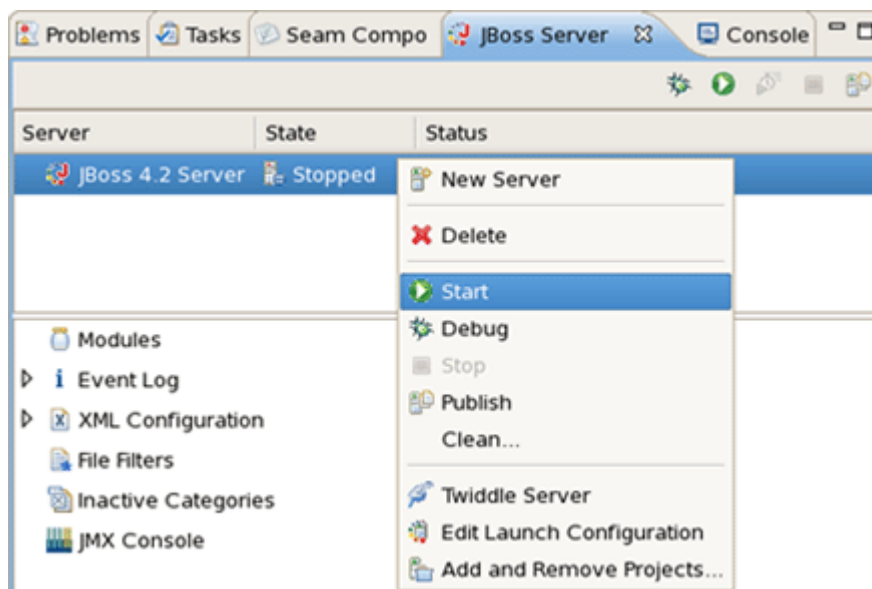
### 2.1.1. Starting JBoss server

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 JBoss Server View or right click server name in this view and select *Start*. If this view is not open, select *Window > Show View > Other > Server > JBoss Server View*

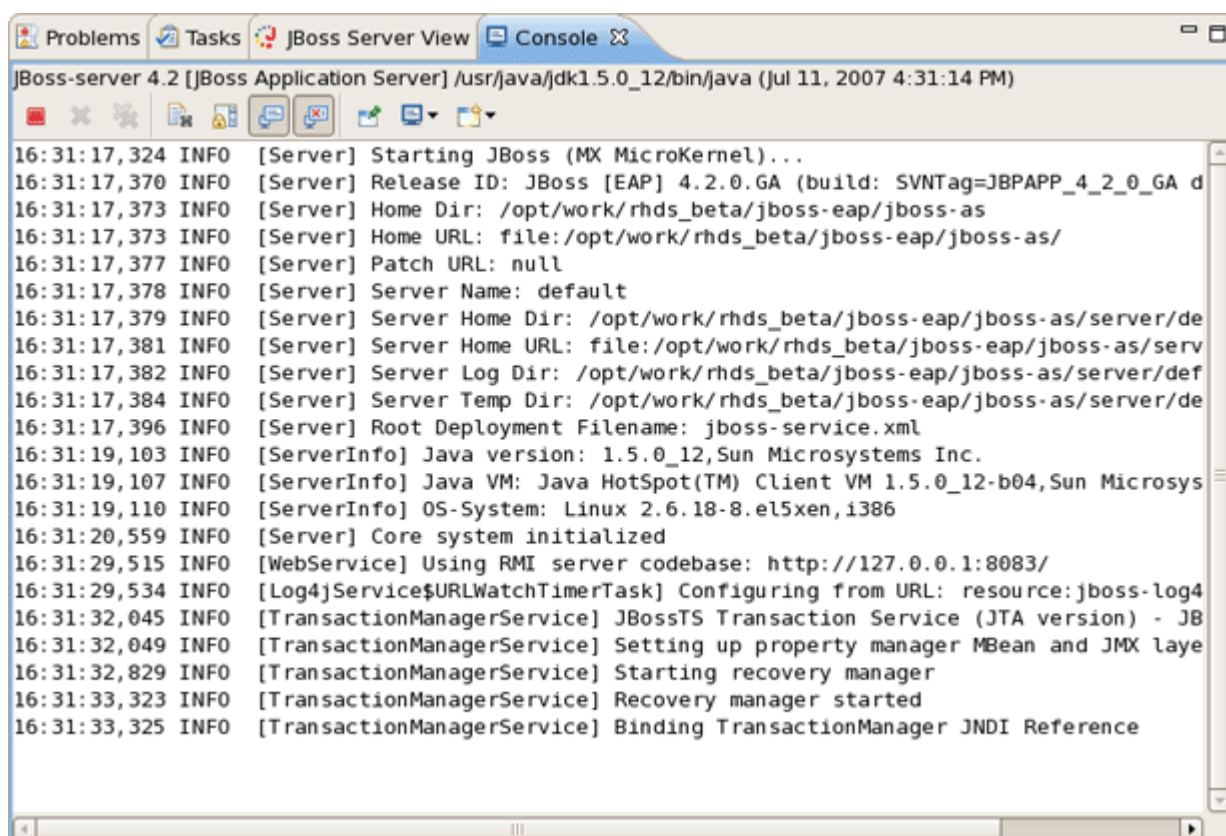


**Figure 2.1. Starting from Icon**



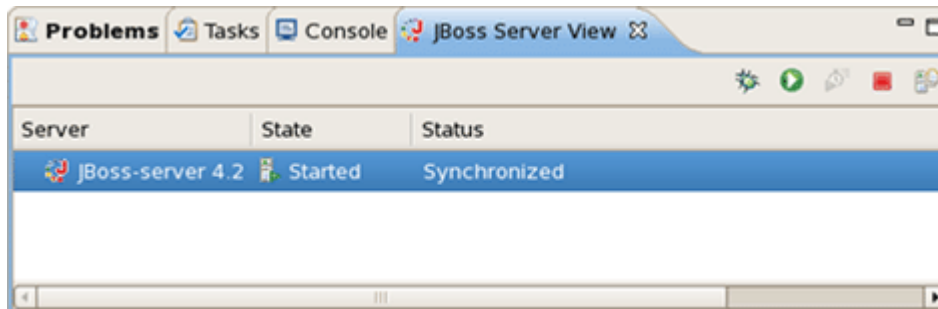
**Figure 2.2. Starting from JBoss Server View**

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



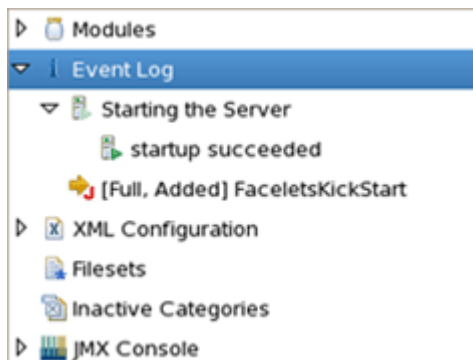
**Figure 2.3. Console Output**

When the server is started you should see *Started* right to its name in JBoss Server View (column "Status").



**Figure 2.4. Server is Started**

To see event log after the server is started, expand Event Log branch beneath JBoss Server View:



**Figure 2.5. Event Log**

## 2.1.2. Stopping JBoss Server

To stop the server, click the *Stop* icon in JBoss Server View or right click the server name and press *Stop*.

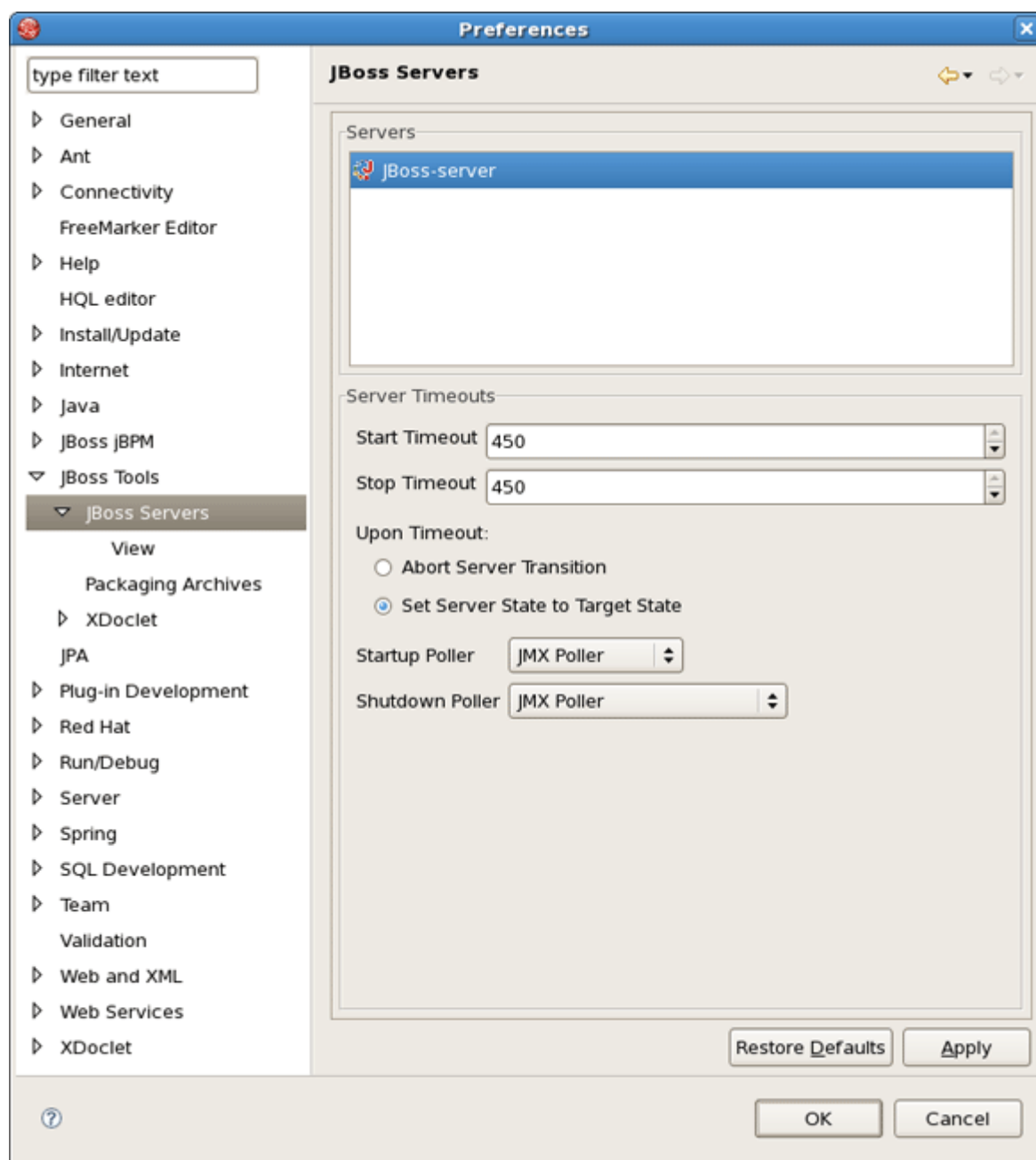


**Figure 2.6. Stopping Server**

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

## 2.1.3. Server Container Preferences

You can control how JBoss Developer Studio interacts with servlet containers in Preferences. Select *Window > Preferences > JBoss Tools > JBoss Servers* and switch to the desired server:

**Figure 2.7. Server Preferences**

Also you can double click the server name in JBoss Server View and open an overview of the server. Here you can specify some common settings: host name, server name, runtime and so on.



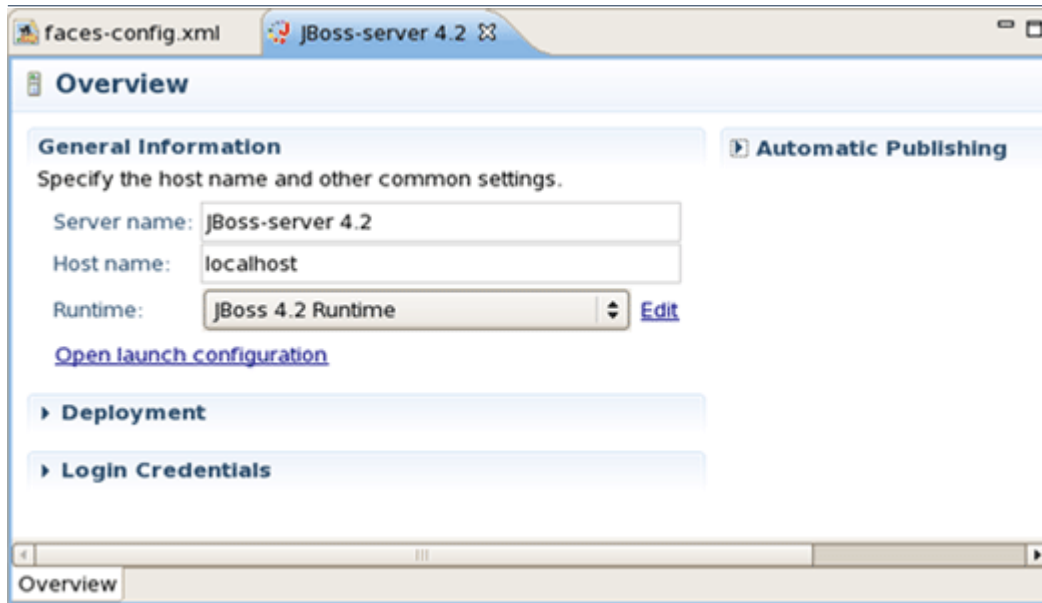


Figure 2.8. Server Overview

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

Although JBoss Developer Studio works closely with JBoss EAP 4.2 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 4.2.1 server. First of all you need to install it.

### 2.2.1. JBoss AS Installation

- Download the binary package of JBoss 4.2.1 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.

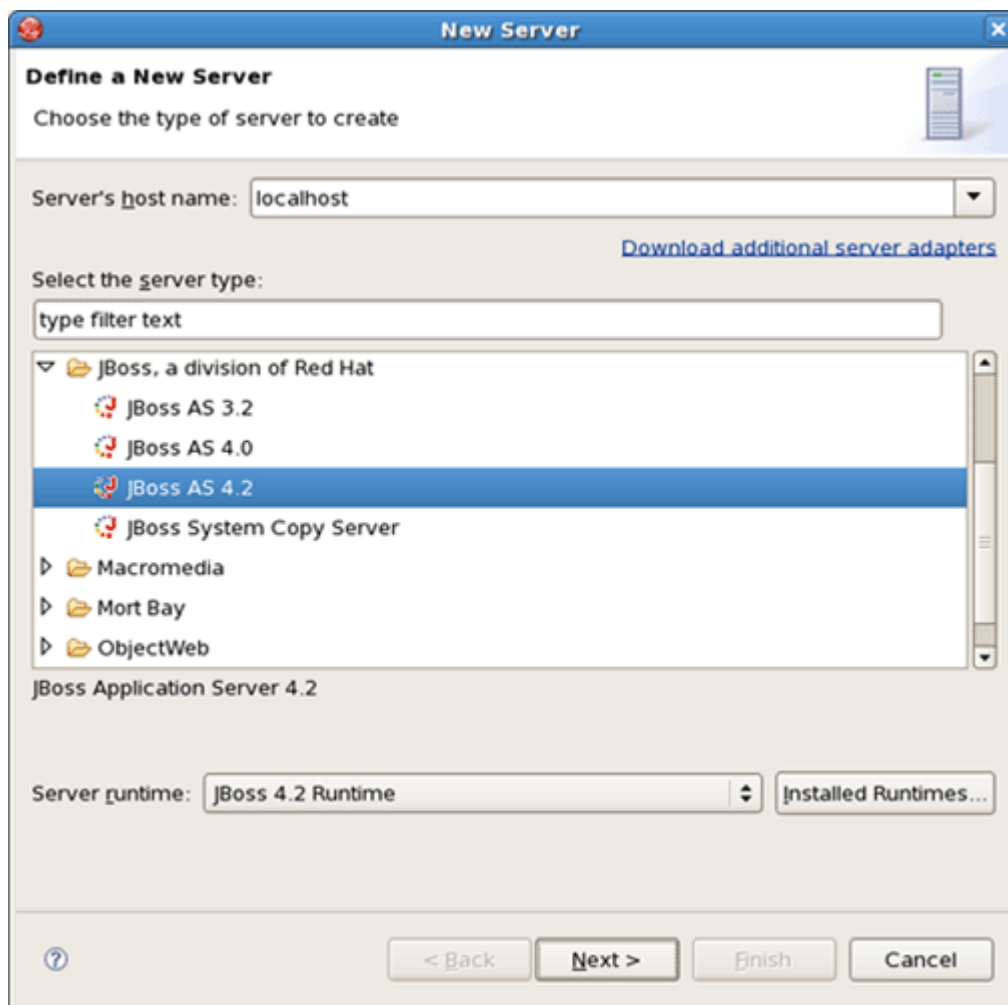
- After you have the binary archive you want to install, use the JDK jar tool (or any other ZIP extraction tool) to extract the jboss-4.2.1.zip archive contents into a location of your choice.

The jboss-4.2.1.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-4.2.1 directory.

### 2.2.2. Adding and configuring JBoss server

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

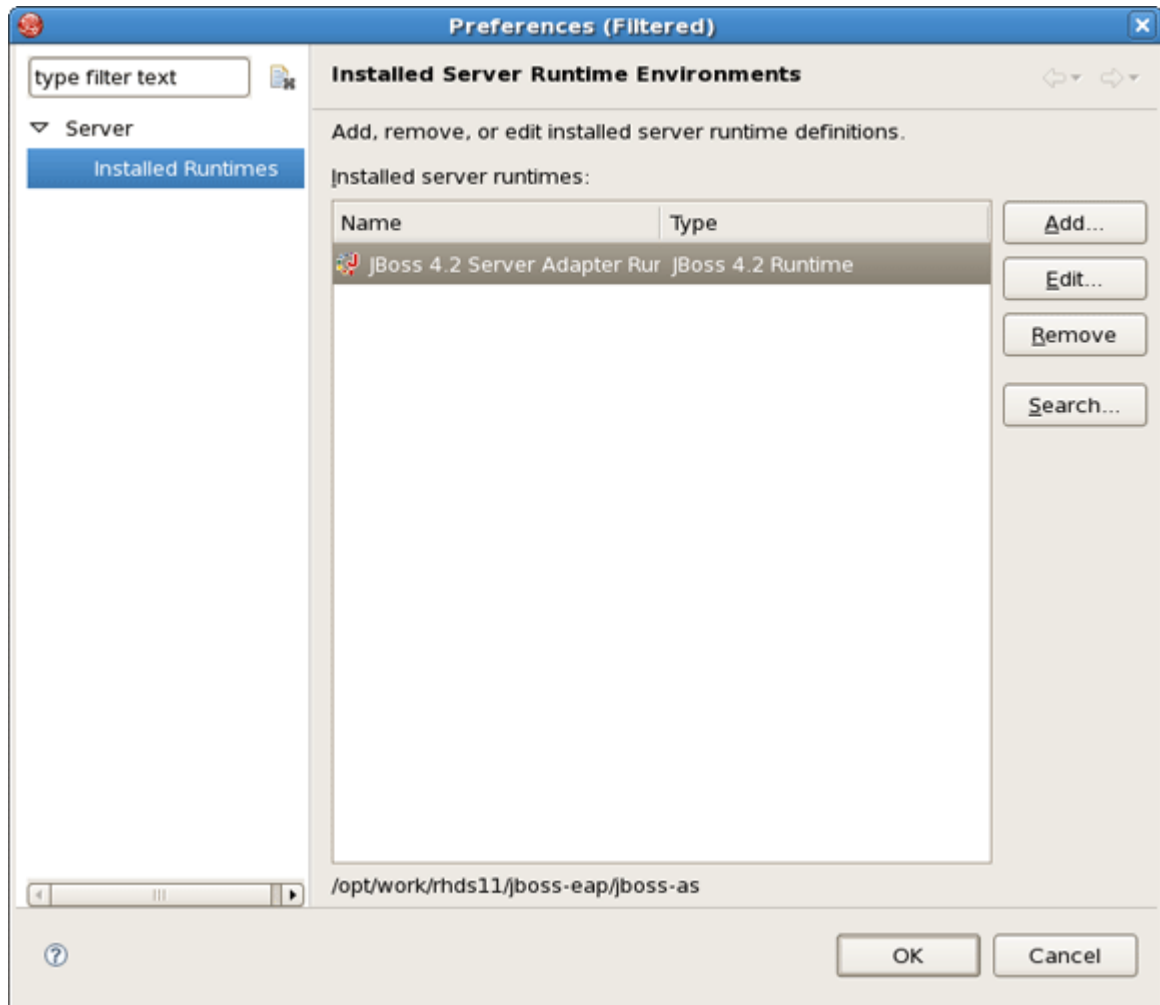
- Open the JBoss Server View by selecting *Window > Show View > Other > Server > JBoss Server View*. You will see JBoss Server view.
- Right click anywhere in this view and select *New Server*.
- Select *JBoss, a division of Red Hat > JBoss v4.2* and click the *Installed Runtimes* button to select a new installed runtime.



**Figure 2.9. Selecting Server Type**

- Click *Add* button to add a new jboss runtime.

- Select *JBoss, a division of Red Hat > JBoss v4.2* and press *Next*.



**Figure 2.10. Installed Runtimes**

- In the next step make JBoss Developer Studio to know where you have installed the server and define JRE.



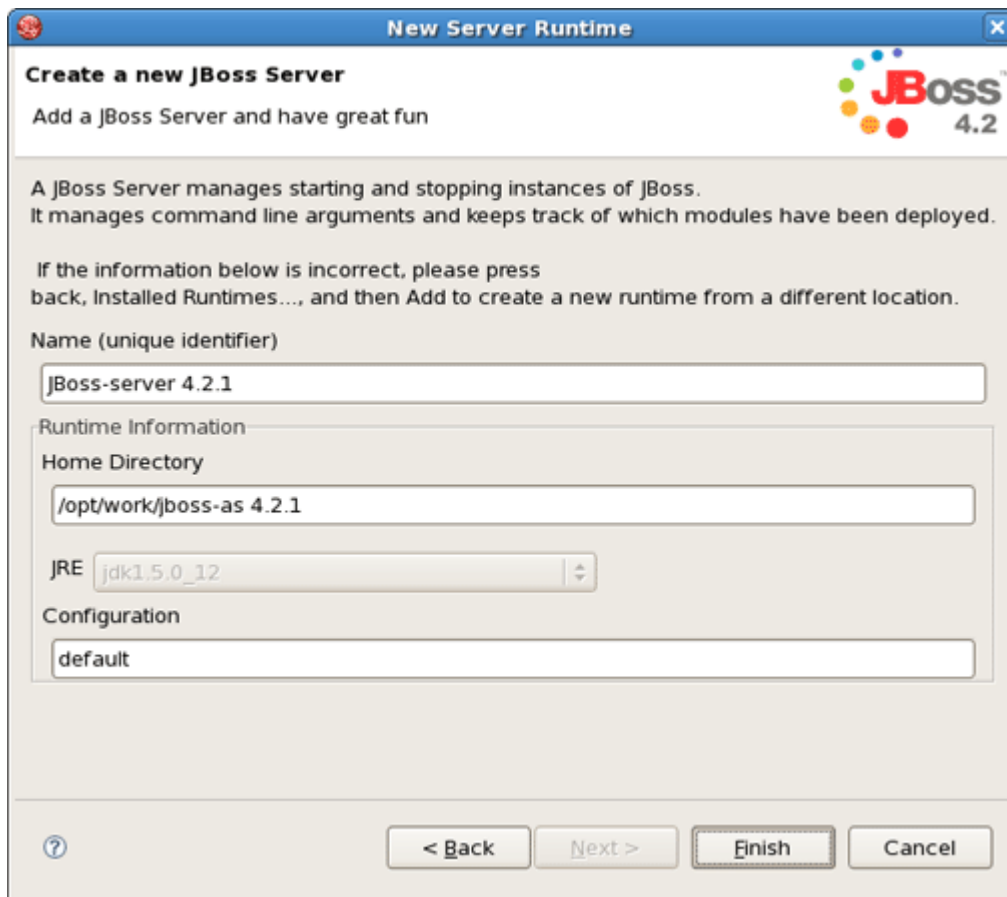
Figure 2.11. Defining JBoss Runtime



**Note:**

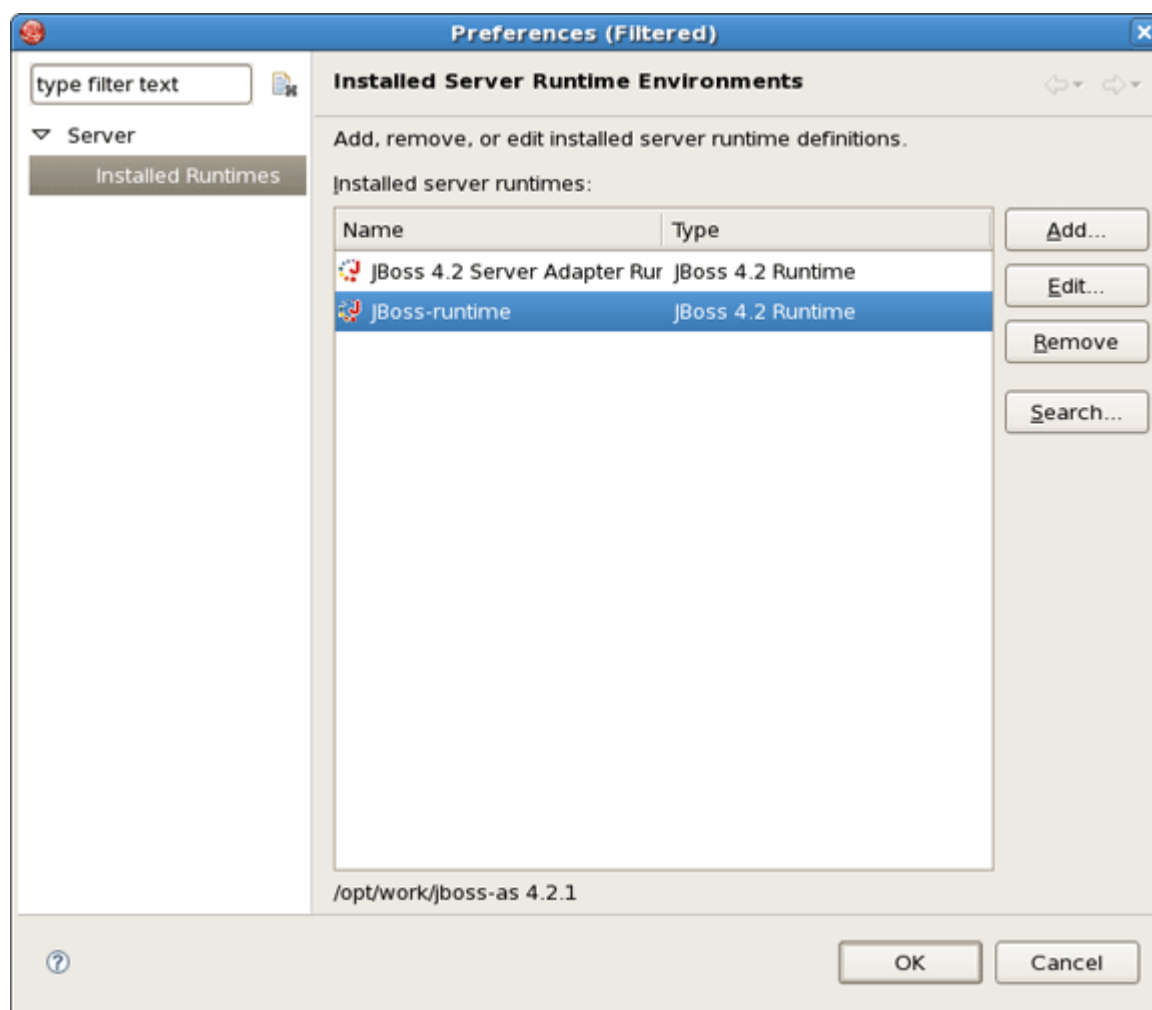
When adding a new server you will need to specify what JRE to use. It is important to set this value to a full JDK, not JRE. Again, you need a full JDK to run Web applications, JRE will not be enough.

- In the following window leave all settings default or give your name to a new jboss server and press *Finish*.



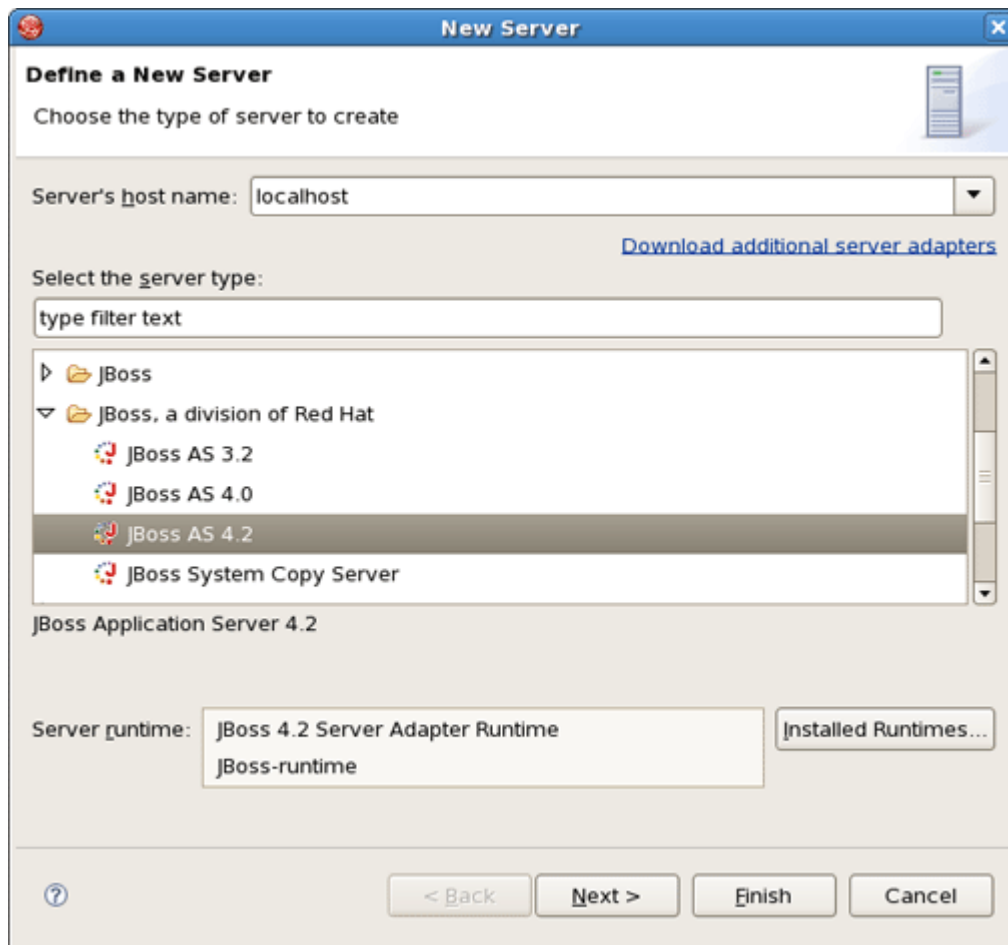
**Figure 2.12. Adding New Runtime**

A new runtime will now appear in the *Preferences > Server > Installed Runtimes* dialog.



**Figure 2.13. Runtime is Added**

- Click *OK*. Then select a new added runtime in Server runtime drop down list and click *Next* button twice.



**Figure 2.14. Choosing Runtime**

- In the next dialog verify a JBoss runtime information and if something is unfair go back and correct it.



**Create a new JBoss Server**

A JBoss Server manages starting and stopping instances of JBoss.  
It manages command line arguments and keeps track of which modules have been deployed.

Name

Runtime Information  
If the runtime information below is incorrect, please press back, Installed Runtimes...,  
and then Add to create a new runtime from a different location.

Home Directory /opt/work/jboss-4.2.0.GA  
JRE /usr/java/jdk1.5.0\_12 (jdk1.5.0\_12)  
Configuration default

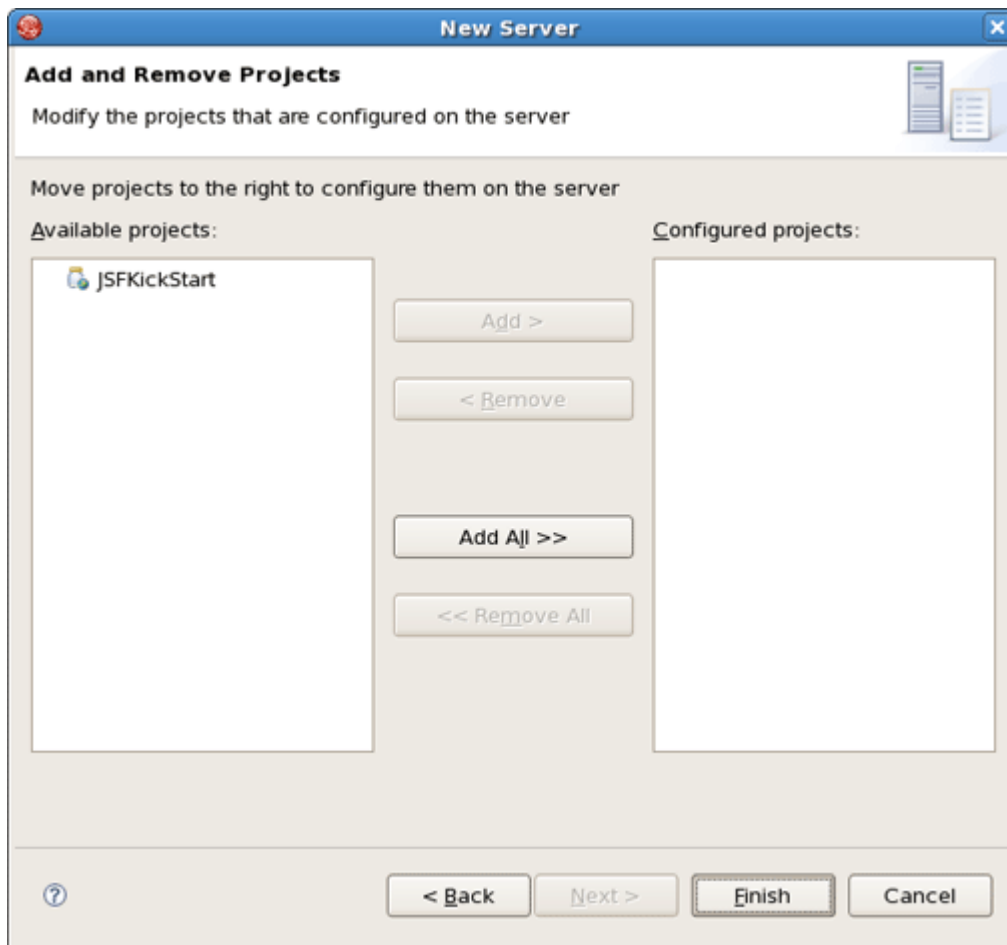
Login Credentials  
JMX Console Access  
User Name   
Password

Deployment  
Deploy Directory

**Figure 2.15. Configuring Projects**

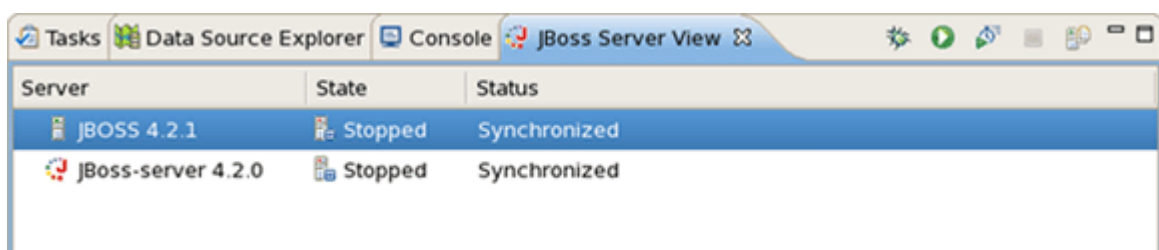
- In the last wizard's dialog modify the projects that are configured on the server and click *Finish*.





**Figure 2.16. Configuring Projects**

A new JBoss server should now appear in JBoss Server View.



**Figure 2.17. New JBoss Server**

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

---

# Write Your First Seam Web Application

The JBoss Developer Studio provides sophisticated tools for enterprise applications. With the JBoss Developer Studio, you can get started very quickly with a web prototype, and then scale up your application to include enterprise features (e.g., business processes, web services, etc.) using the same developer tools. It is a "scalable" RAD (Rapid Application Development) tool.

A core element that makes the JBoss Developer Studio "scalable" is the JBoss Seam framework.

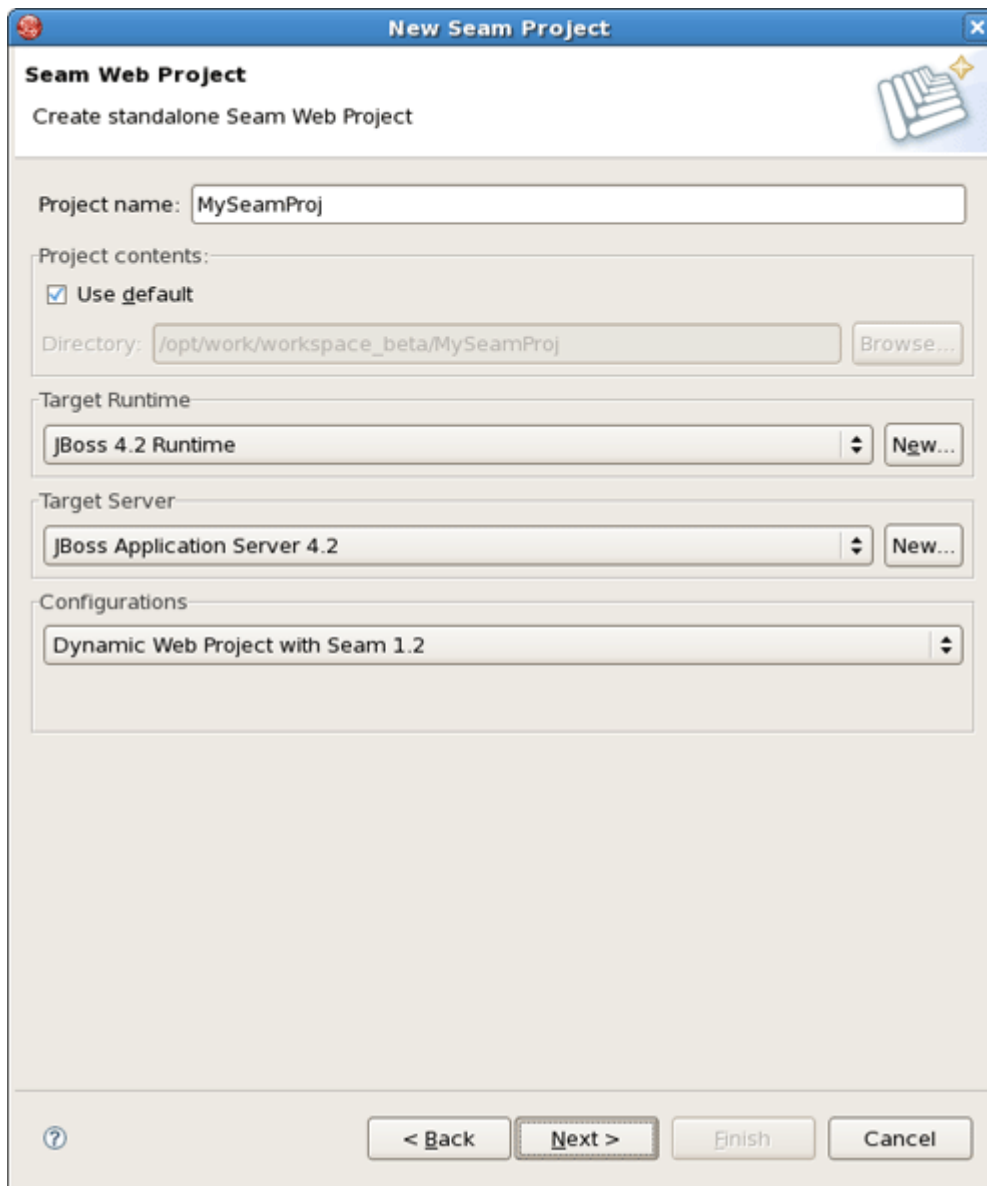
The main purpose of this chapter is to tell you about build a simple Seam web application in minutes with the JBoss Developer Studio.

## 3.1. Create a Seam Project

This section helps you to create a simple Seam project.

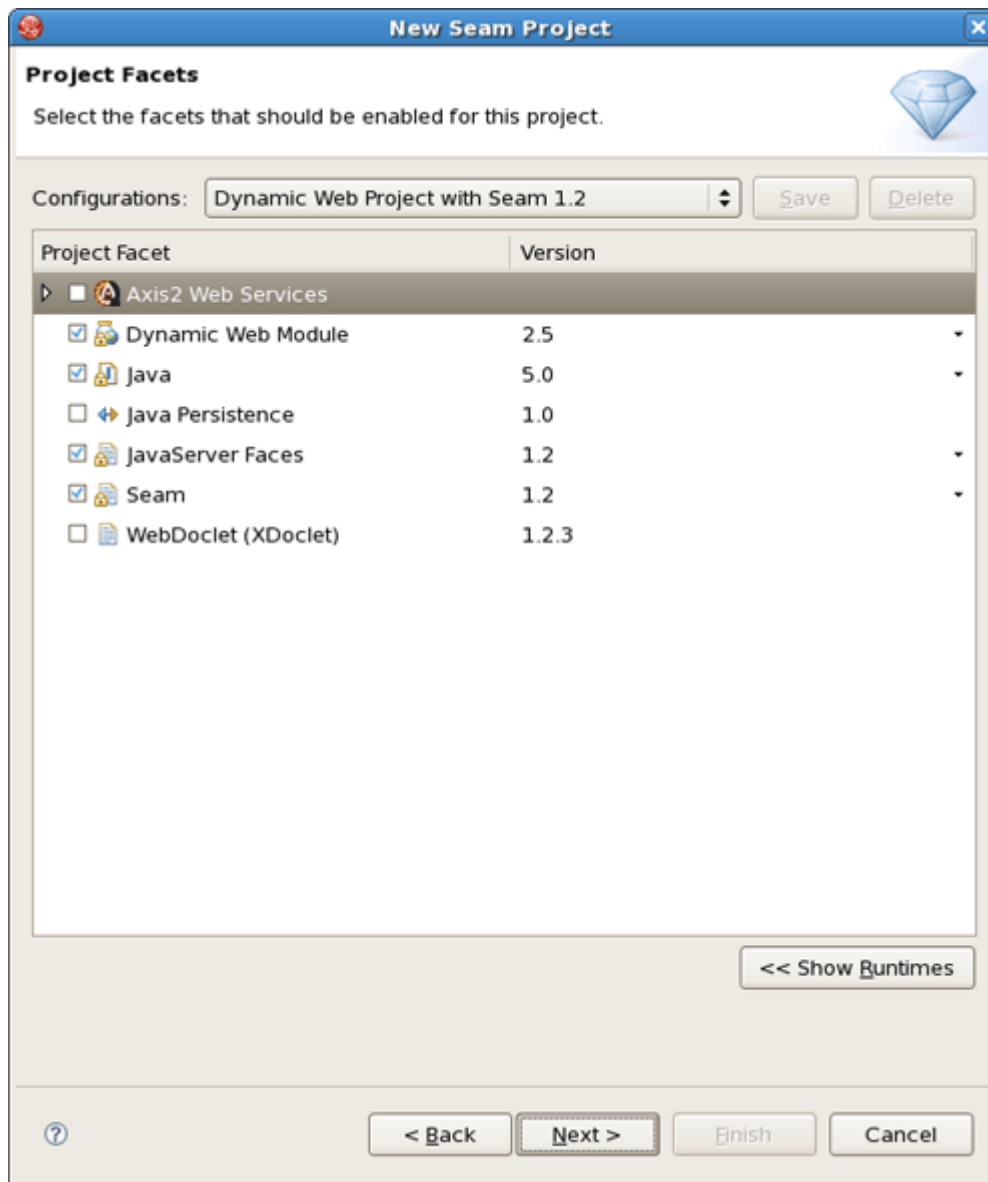
To create a new web application in Seam, you should create a Seam web project. This section provides all the necessary steps to organize a new project with appropriate tooling and adjust the settings that match your needs. In order to find out more information, see [Seam Dev Tools Reference guide](#) [[../en/html\\_single/index.html](#)]

First, select *New > Project ... > Seam > Seam Web Project* . You will be prompted to enter a name 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 allow the wizard to correctly identify where the needed datasource and driver libraries need to go.



**Figure 3.1. Create a Seam Project**

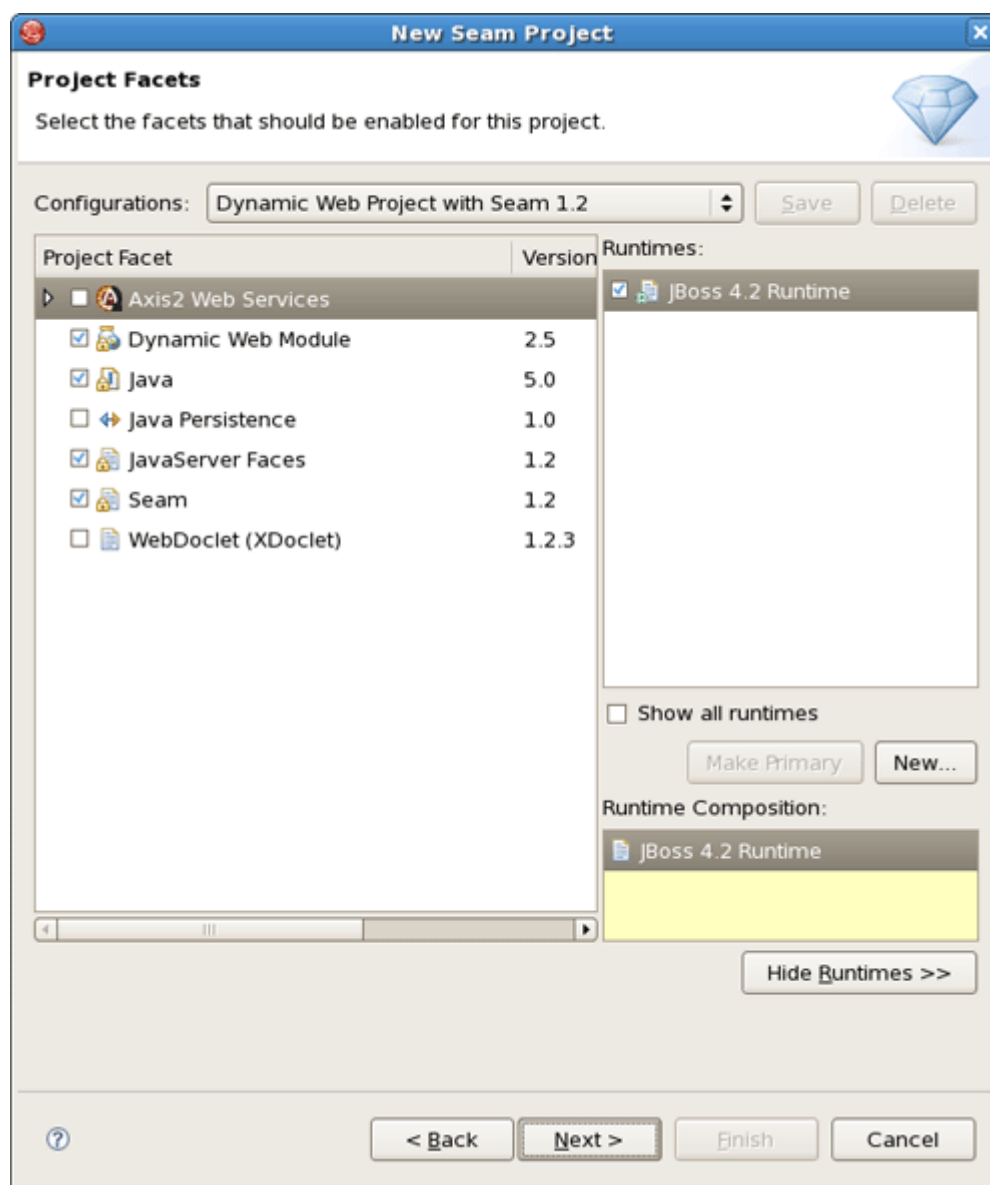
Next, you will be asked to select the "features" you want to use in your project. This allows JBoss Developer Studio to setup the appropriate tooling for your project. Since JBoss Seam integrates all popular Java EE frameworks, you can select any combination of technologies from the list. Here, for this project, we will select Dynamic Web Module, Java, JavaServer Faces (JSF), and Seam Facet for a typical database-driven web application.



**Figure 3.2. Select Toolings for the Project**

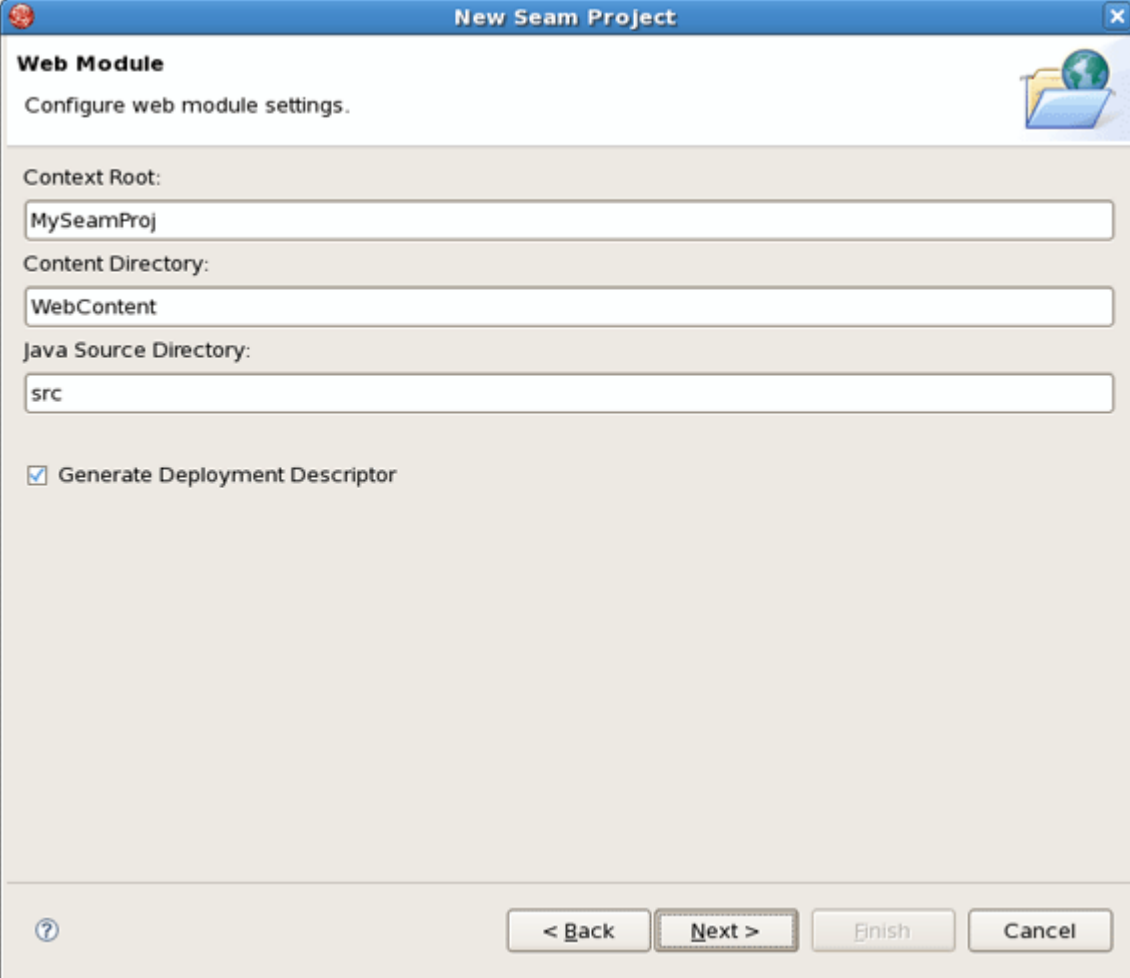
In this screen you can also bring up server runtimes panel by clicking *Show Runtimes* in the bottom right corner. This panel shows available server runtimes. Then this button will be changed into *Hide Runtimes* one.

Click *Next* to proceed further.



**Figure 3.3. Available Server Runtimes**

A dynamic web application contains both web pages and Java code. The wizard will ask you where you want to put those files. You can just leave the default values or choose another folder.



**New Seam Project**

**Web Module**  
Configure web module settings.

Context Root:  
MySeamProj

Content Directory:  
WebContent

Java Source Directory:  
src

☒ Generate Deployment Descriptor

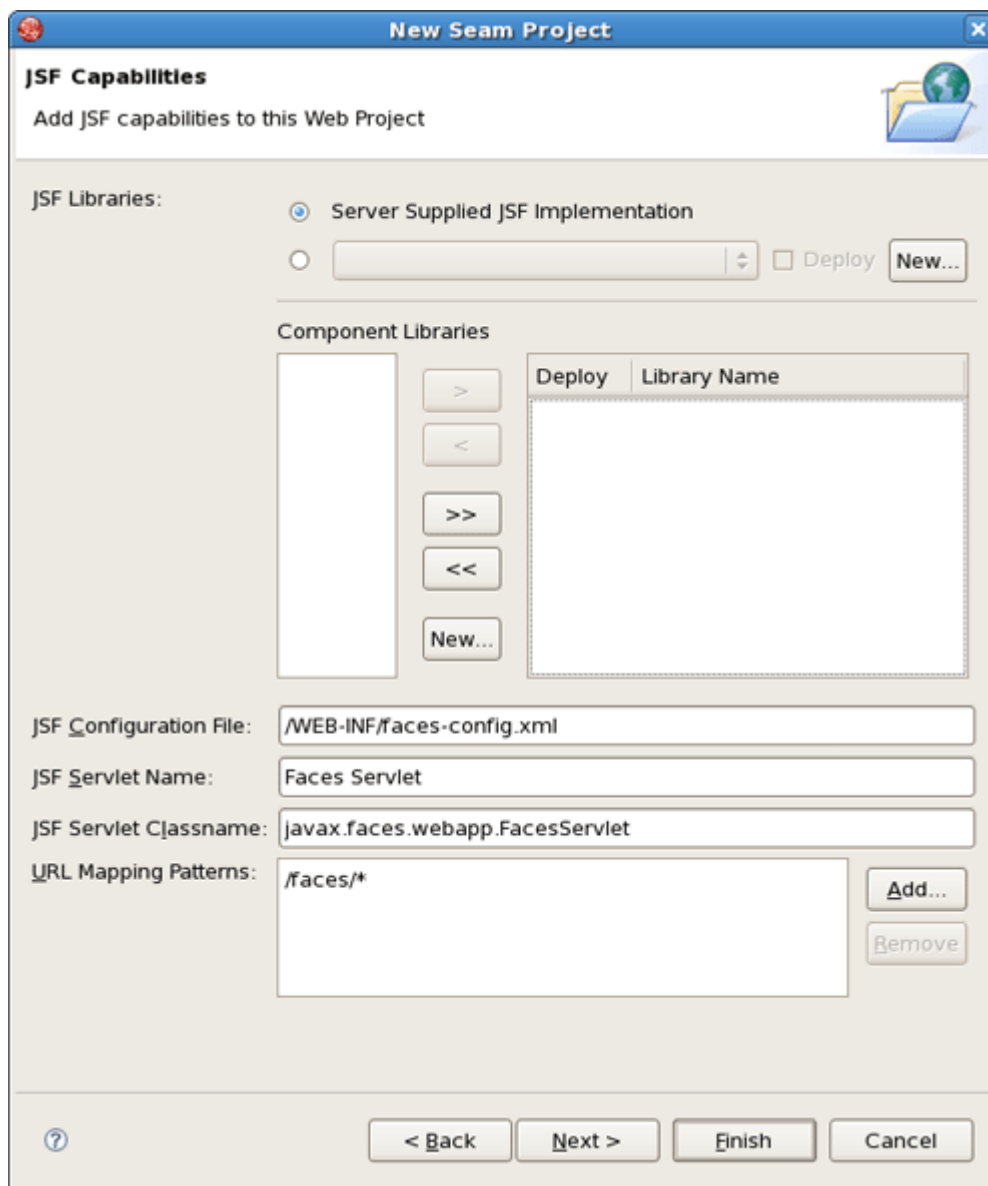
? < Back Next > Finish Cancel

**Figure 3.4. Select Directory Names for Web Pages and Java Files**

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.

6. Check *Server Supplied JSF Implementation*. We will use [JSF implementation](#) [../../seam/en/html\_single/index.html#addJSFCapab] that comes with JBoss server

7. Click *Next*



**Figure 3.5. Define JSF Implementation**

We will also use a default Hibernate Dialect - *org.hibernate.dialect.HSQLDialect* and deploy as a *war* archive.

The project setup wizard also asks you to configure how Seam generates code for the project. The Seam Home Folder should point to a valid Seam distribution. By default, it is set to the Seam distribution bundled in your JBoss Developer Studio tool. If you need another one choose setting up the appropriate check box:



**New Seam Project**

**Seam Facet**  
Configure Seam Facet Settings

**General**

Seam Runtime: Seam 1.2.AP [Add]

Deploy as: ☒ WAR ☐ EAR

**Database**

Database Type: HSQL

Connection profile: DefaultDS [Edit...] [New...]

Database Schema Name: [ ]

Database Catalog Name: [ ]

DB Tables already exists in database: ☐

Recreate database tables and data on deploy: ☐

**Code Generation**

Session Bean Package Name: org.domain.MySeamProj.session

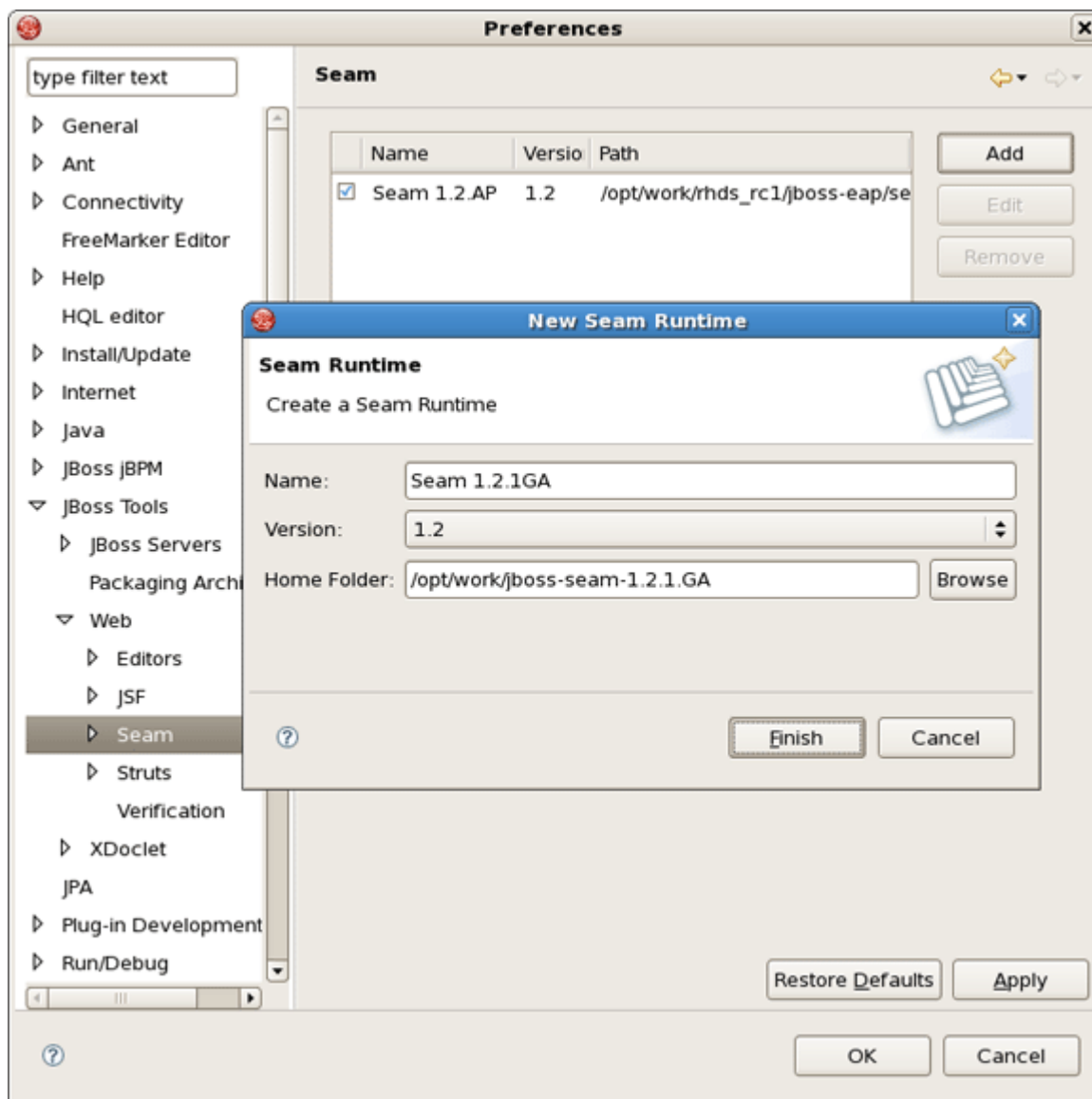
Entity Bean Package Name: org.domain.MySeamProj.entity

Test Package Name: org.domain.MySeamProj.test

[?] < Back Next > Finish Cancel

**Figure 3.6. Enter Java Packages for Generated Code**

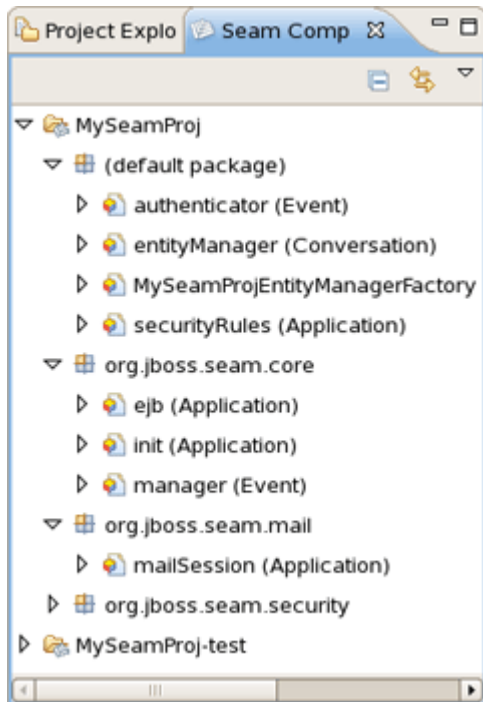
If in this list there is no Seam runtime you want to use add it through *Window > Preferences > JBoss Tools > Web > Seam* or just click *Add* button near the Seam Runtime list:



**Figure 3.7. Add New Seam Runtime**

For the deployment format, choose WAR deployment if you want to use POJOs for UI event handling and business logic; choose EAR deployment if you want to EJB3 beans for added features. In most web applications, the WAR deployment option would suffice. You should also enter Java packages for the entity beans (for database mapping) and session beans (for action handlers). All generated code will be placed in those packages.

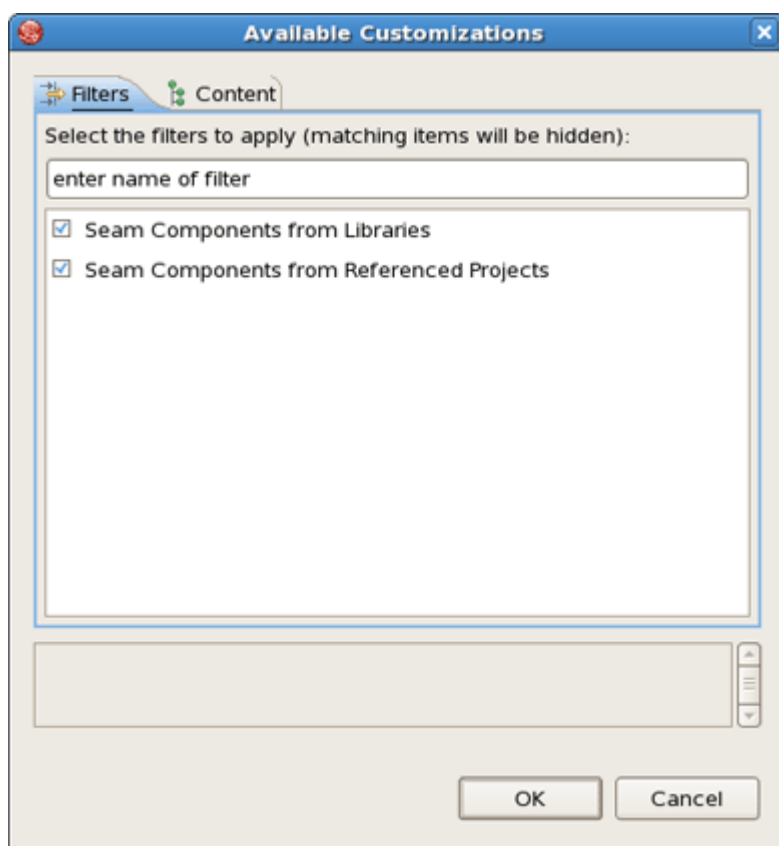
Click on *Finish* to generate a project. The generated project contains all the necessary library JARs, XML configuration files, the ANT build script, as well as simple XHTML web pages and Java classes for the skeleton web application. The project will be shown in Project Explorer as well as in *Seam Components* view. If Seam Components view is not open select *Window > Show View > Seam Components*.



**Figure 3.8. Seam Components View**

You can hide unused Seam components from this view.

- Click the button *Menu* on the top of the view (down-pointing arrow)
- Choose *Customize View..*
- In the dialog *Available Customization* check the filter you want to apply under the *Filters* tab

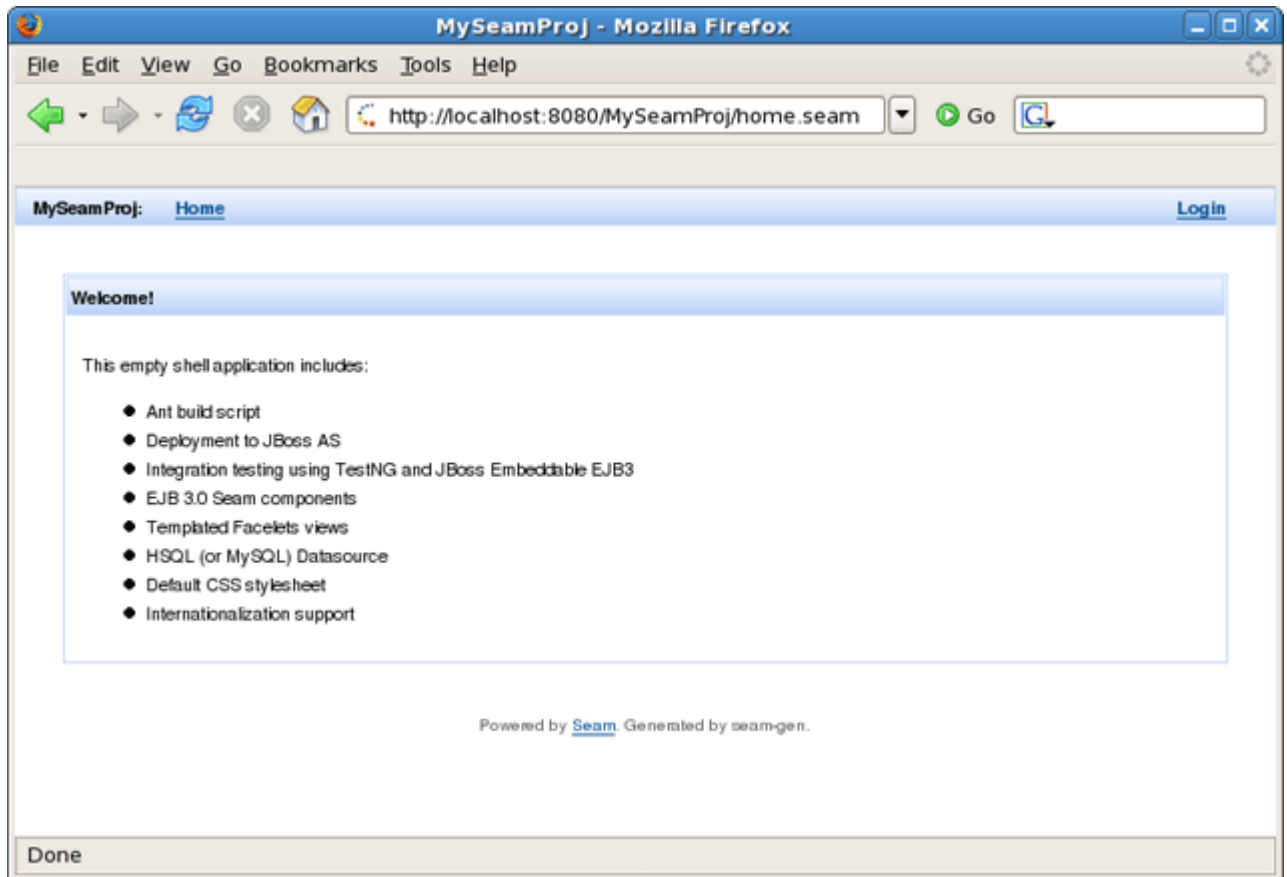


**Figure 3.9. Seam Components Filtering**

## 3.2. Build and Deploy the Seam Application

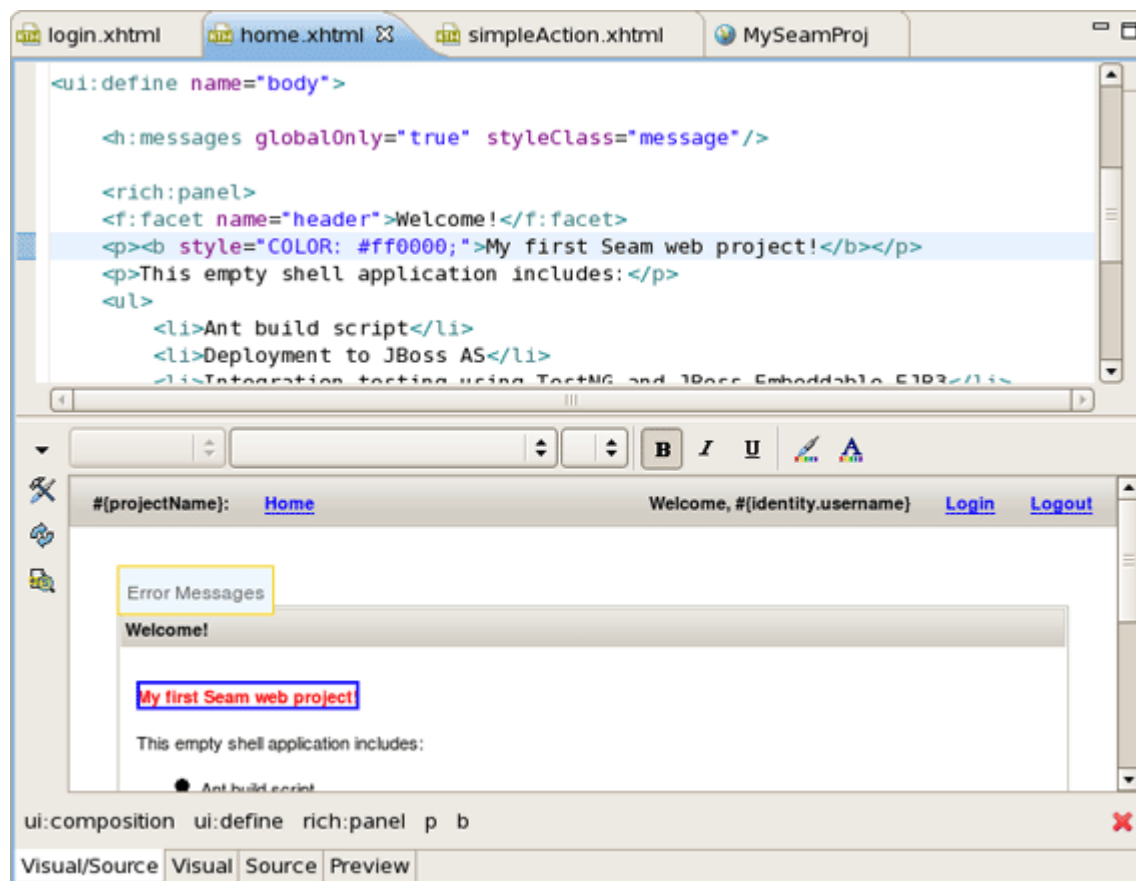
Here, we will show you how to deploy our web project to the server and then view the application as a web site from a URL.

Once the application is generated, you can use the "Run on server" menu to build and deploy it into the JBoss AS runtime associated with the project. All you need is to start JBoss AS in the server manager, and load the browser at URL <http://localhost:8080/MySeamProj/>. You should see the following web page.



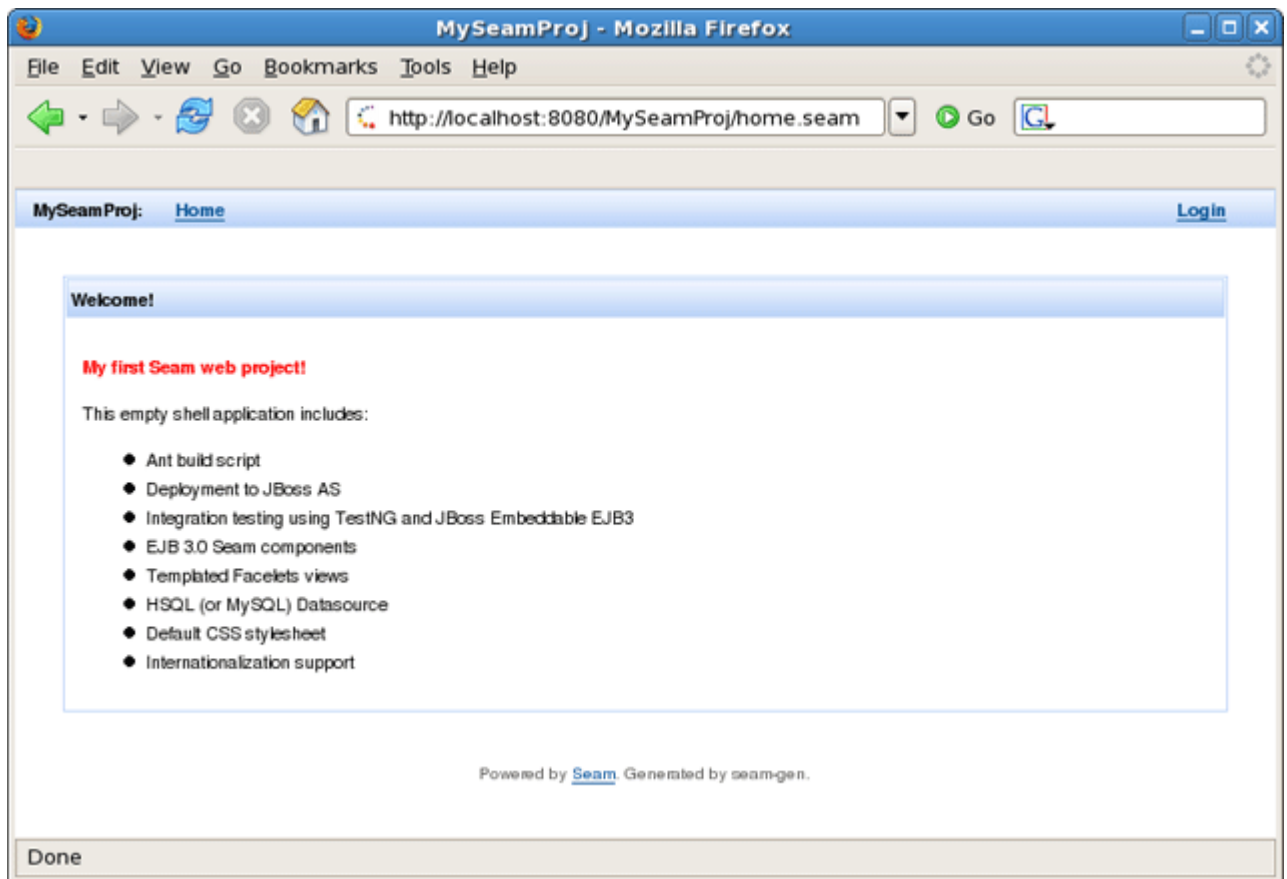
**Figure 3.10. The Generated Application in Action**

To make simple changes to the page, you just need to double click on the *WebContent/home.xhtml* file and edit it in the visual editor. Notice that the visual editor lets you both the XHTML code and the rendered page. The rendered view is designed to make it easy to find stuff in a complex XHTML page. If you'd like to learn more about the VPE, read the Editors section in the [Visual Web Tools Reference guide](#) [[../en/html\\_single/index.html](#)].



**Figure 3.11. Making Changes in the Visual Editor**

Once you finished editing, save the file ( *File > Save* ), and reload the browser to see the changes.



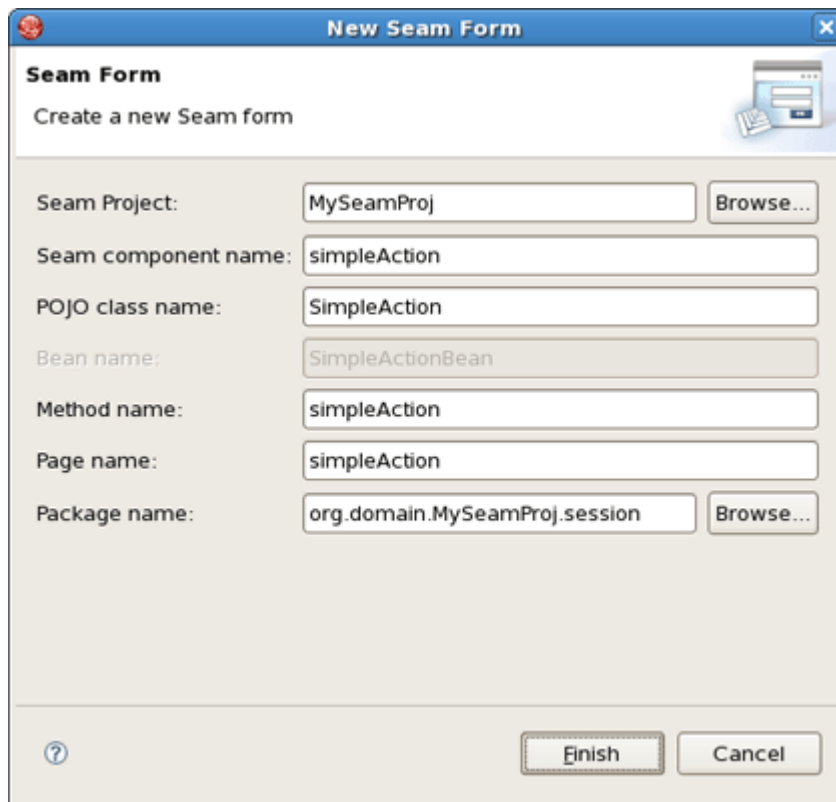
**Figure 3.12. The Front Page is Changed**

Notice that we do not need to re-build and re-deploy the application. Just save the edited page and reload the browser.

### 3.3. Add a Web Page and an Action

Here, we are going to add a new page and related UI action to the project.

To do this use the *New > Other ... > Seam > Seam Form* wizard. You are prompted to enter the name of the project and seam component name, all the others fields will be filled by the wizard.



**Figure 3.13. New Form for the Application**

The wizard generate a web page with a single text input field and an action button. Notice that the generated page uses *layout/template.xhtml* as a template. The template page provides the page header, footer, side menu, and CSS styles (see the *template.xhtml* for more details). The *simpleAction.xhtml* is assembled into the template when the *simpleAction.seam* URL is loaded.

```
<!DOCTYPE composition PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<ui:composition> xmlns:s="http://jboss.com/products/seam/taglib"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:h="http://java.sun.com/jsf/html"
    template="layout/template.xhtml">

<ui:define name="body">

    <h:messages globalOnly="true" styleClass="message"/>

    <h:form id="simpleActionForm">
        <rich:panel>
```



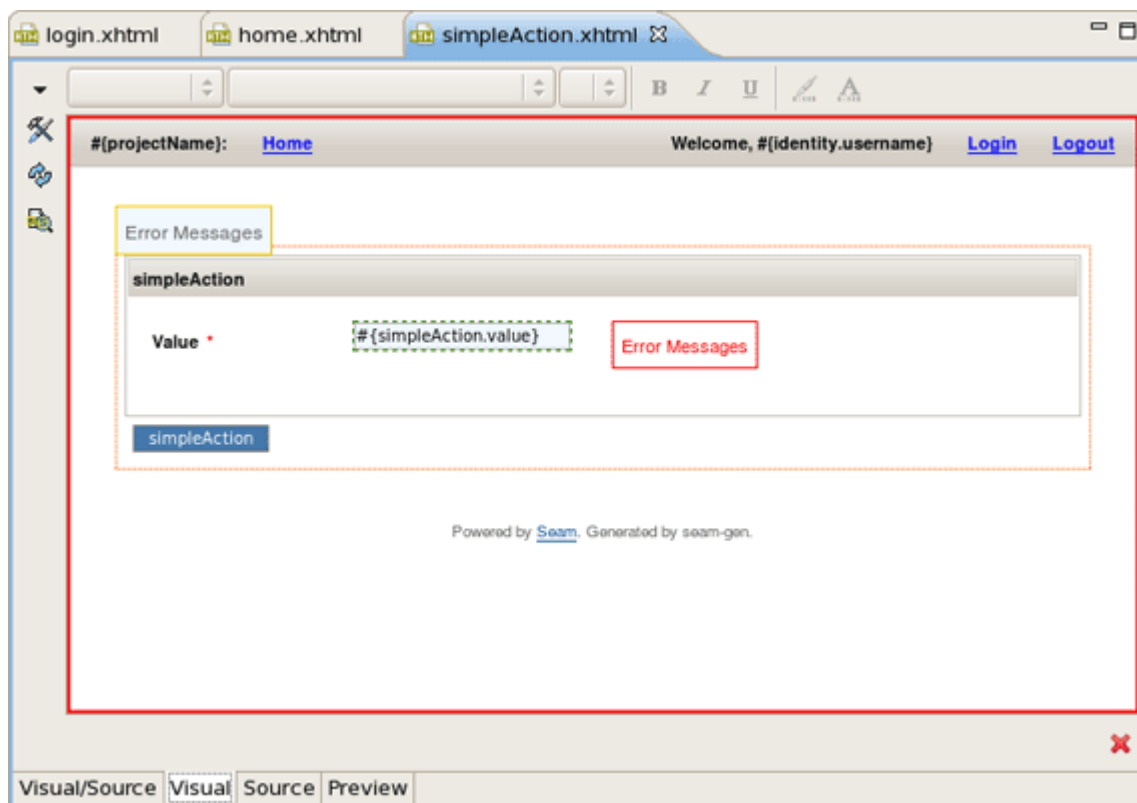
```

<f:facet name="header">simpleAction</f:facet>
  <s:decorate id="valueDecoration" template="layout/edit.xhtml">
    <ui:define name="label">Value</ui:define>
    <h:inputText id="value" required="true"
      value="#{simpleAction.value}"/>
  </s:decorate>
  <div style="clear:both"/>
</rich:panel>

<div class="actionButtons">
  <h:commandButton id="simpleAction" value="simpleAction"
    action="#{simpleAction.simpleAction}"/>
</div>
</h:form>
</ui:define>

</ui:composition>

```



**Figure 3.14. Generated Form**

The `#{simpleAction.value}` notation on the web page maps to the "value" property in the backend component named "simpleAction", and the `#{simpleAction.simpleAction}` notation indicates that

the `simpleAction()` method is called when the button is clicked on. Here is the "simpleAction" named backend Seam component generated by the wizard.

```
package org.domain.MySeamProj.session;

import org.jboss.seam.annotations.Name;
import org.jboss.seam.annotations.In;
import org.jboss.seam.annotations.Logger;
import org.jboss.seam.log.Log;
import org.jboss.seam.core.FacesMessages;
import org.hibernate.validator.Length;

@Name("simpleAction")
public class SimpleAction {

    @Logger private Log log;

    @In
    FacesMessages facesMessages;

    private String value;

    //seam-gen method
    public void simpleAction()
    {
        //implement your business logic here
        log.info("simpleAction.simpleAction()
            action called with: #{simpleAction.value}");
        facesMessages.add("simpleAction #{simpleAction.value}");
    }

    //add additional action methods

    @Length(max=10)
    public String getValue()
    {
        return value;
    }

    public void setValue(String value)
    {
        this.value = value;
    }
}
```

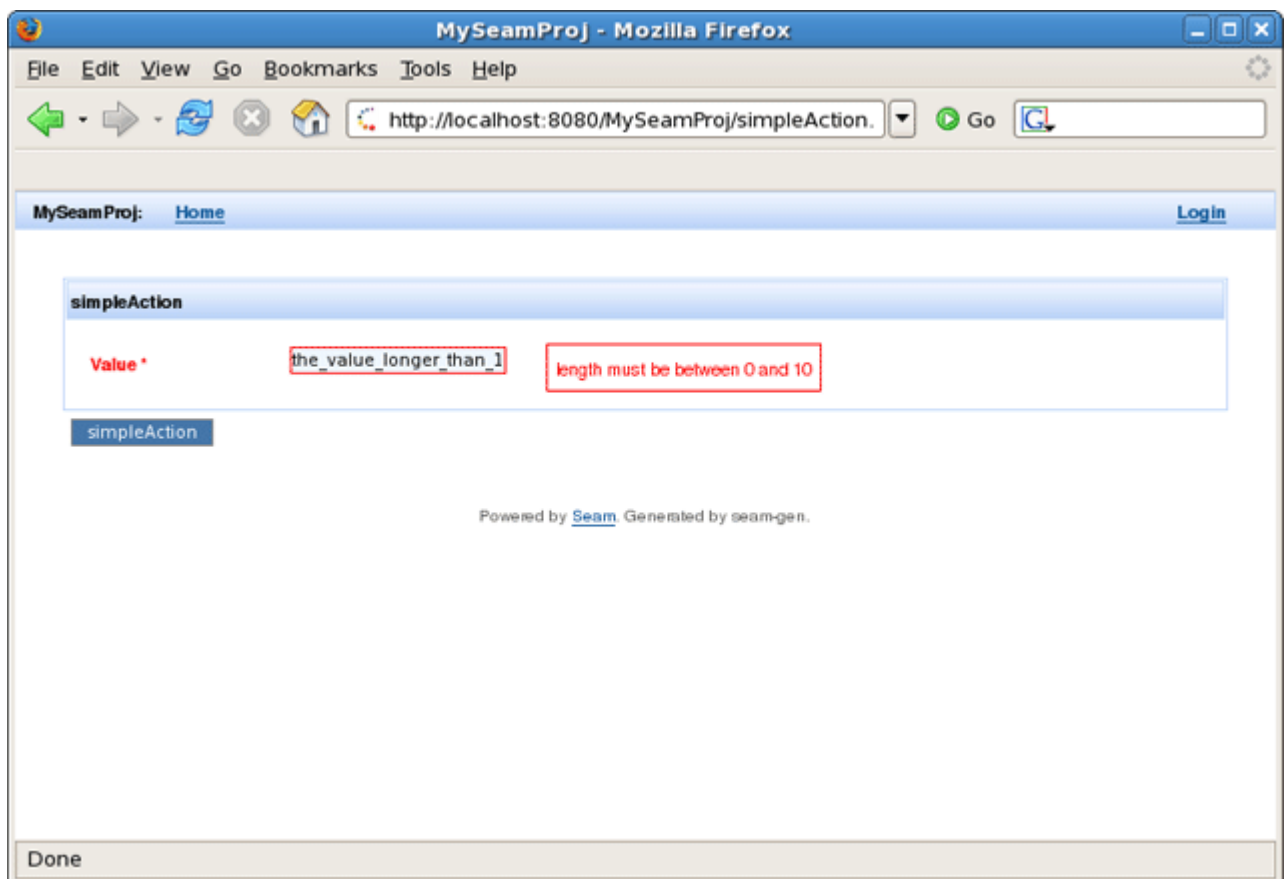
```
}
```

Load the Simplepage.seam in the web browser. Type something in the text field and click on the "simpleAction" button. A JSF message containing the input string is created by the `simpleAction.simpleAction()` method. The message is displayed on the page via the `<h:message>` tag.

### 3.4. Input Validation

In this section we'll focus on the support of input validations.

Notice that in the generated SimpleAction class, there is a `@Length` annotation to validate the input when the input string is bound to `#{simpleAction.value}`. To see how this works, enter a text string longer than 10 chars and click on the button. This is what you should see.

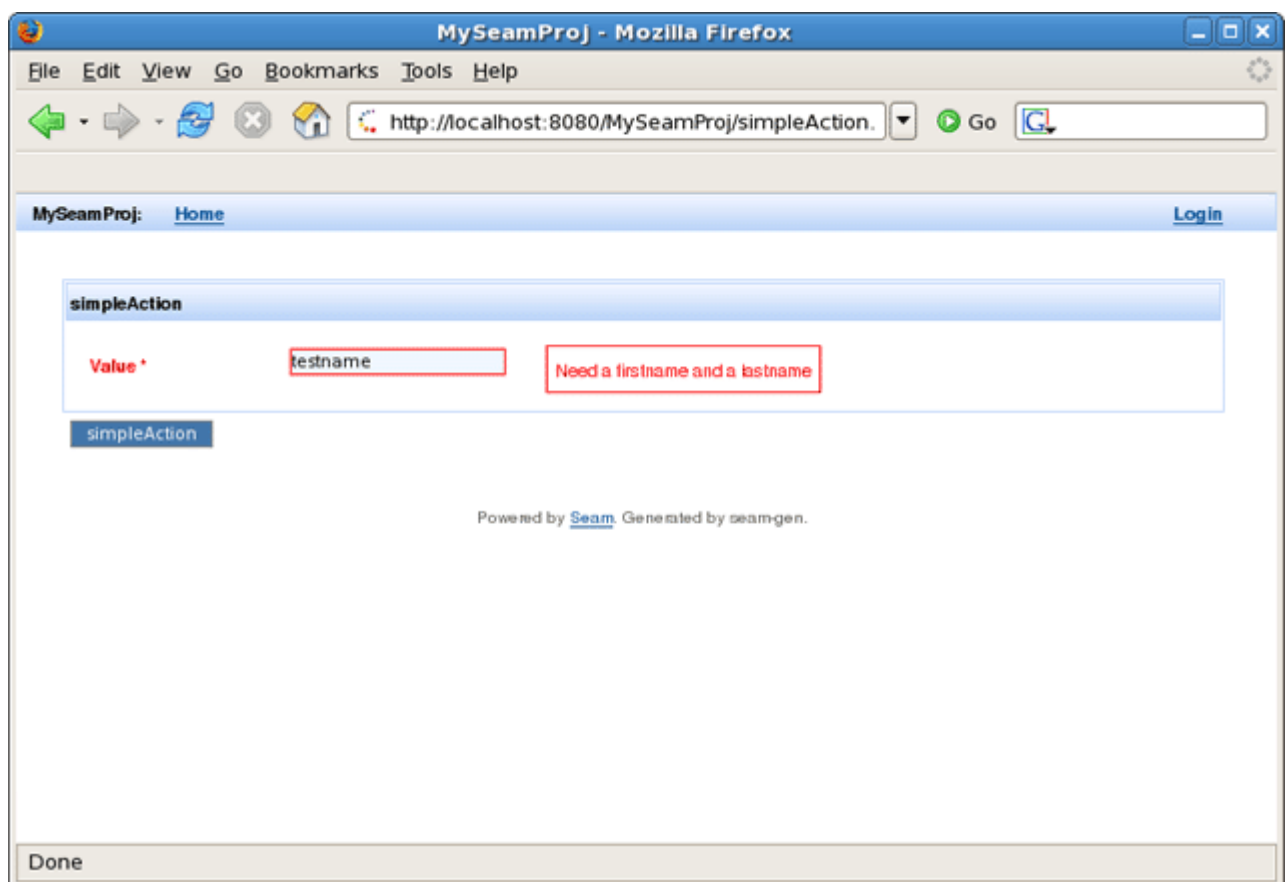


**Figure 3.15. The Input Validation in Action**

Seam supports many different input validation annotations. To see an example, you can replace the `@Length(max=10)` annotation with the following. It would require the input string to have a first name and last name separated by a space. If the validation fails, the web page would print the customized error message.

```
@NotNull
@Pattern(regex="^[a-zA-Z.-]+ [a-zA-Z.-]+$",
        message="Need a firstname and a lastname")
public String getValue()
{
    return value;
}
```

Save the Java file, deploy the application, and reload the browser to see the new validation scheme in action.



**Figure 3.16. More Input Validation**

### 3.5. Add a new UI Component

This section tells you about how you can add a new UI Component.

Now, let's add a little more logic to the application. We will add a new boolean property to the action component. If it is set to true, the action would capitalize the input string and display it on the web page. The following code in the SimpleAction class implements the logic.

```

@Name("simpleAction")
public class SimpleAction {

    private boolean convertToCap;

    public boolean getConvertToCap () { return convertToCap; }
    public void setConvertToCap (boolean b) { convertToCap = b; }

    public String hello()
    {
        if (convertToCap) {
            value = value.toUpperCase ();
        }
        return null;
    }
    ... ..
}

```

Next, on the web page, add the following line to display the value property on the simpleAction component. Notice that code completion is supported for the JSF EL expression.

```
<p><b>Hello, #{simpleAction.value}</b></p>
```

Finally, on the web page, we add a boolean selection box component. It is bound to the *convertToCap* property on the backend component.

```

<h:selectBooleanCheckbox title="convertToCap"
    value="#{simpleAction.convertToCap}" />
Capitalize the input?

```

Deploy the application and see it in action now.

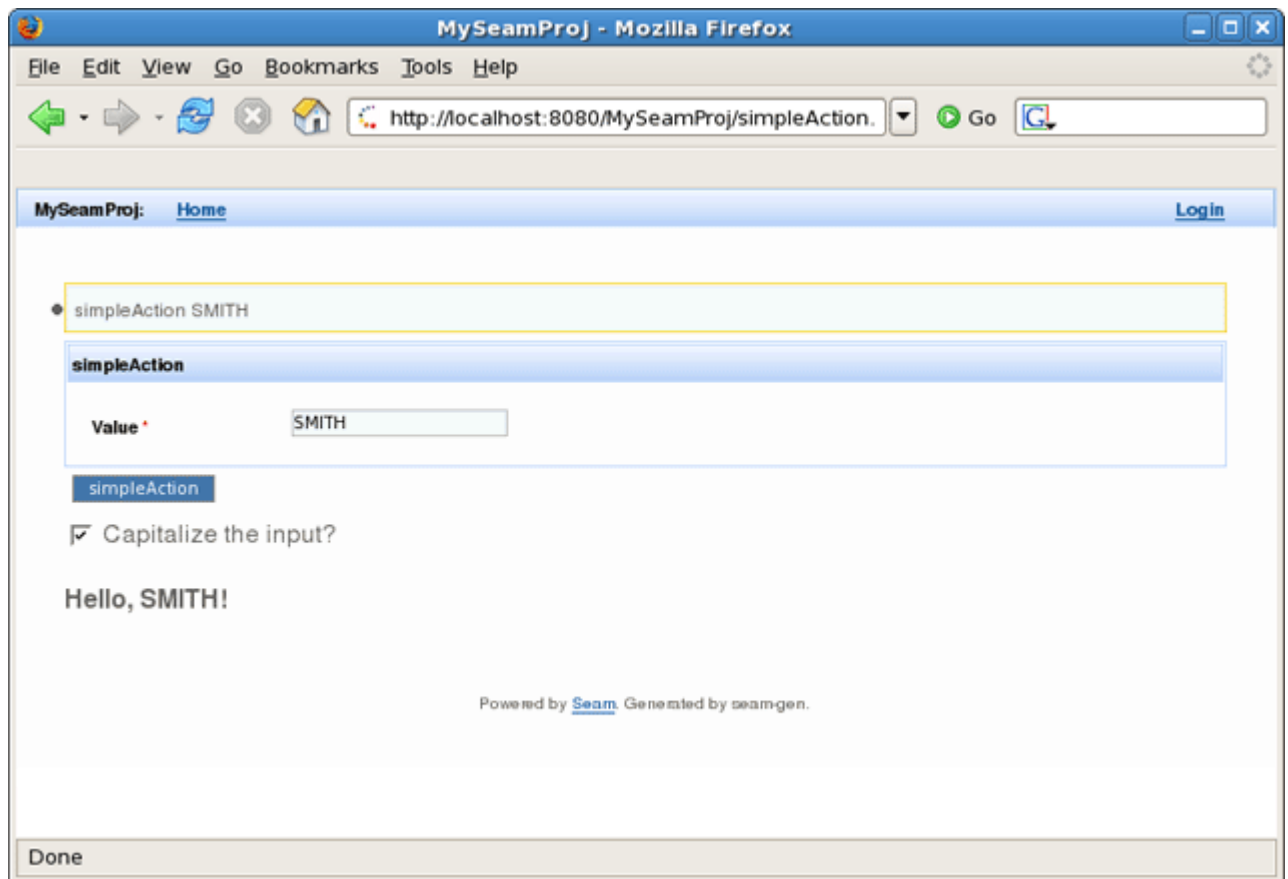


Figure 3.17. Add UI Components and Business Logic

### 3.6. Add Security to the Application

As the last point, we can add an access control to the application.

You have probably noticed that the web page template has a login link at the top of the page. You can use the Seam security framework to secure access to any web page or web action. We just use hardcoded username and password but you can easily change it to use database, LDAP or any other means. The simplest use case for Seam security is to add a declarative security in pages.xml ( *WebContent* > *WEB-INF* > *pages.xml* ) like this:

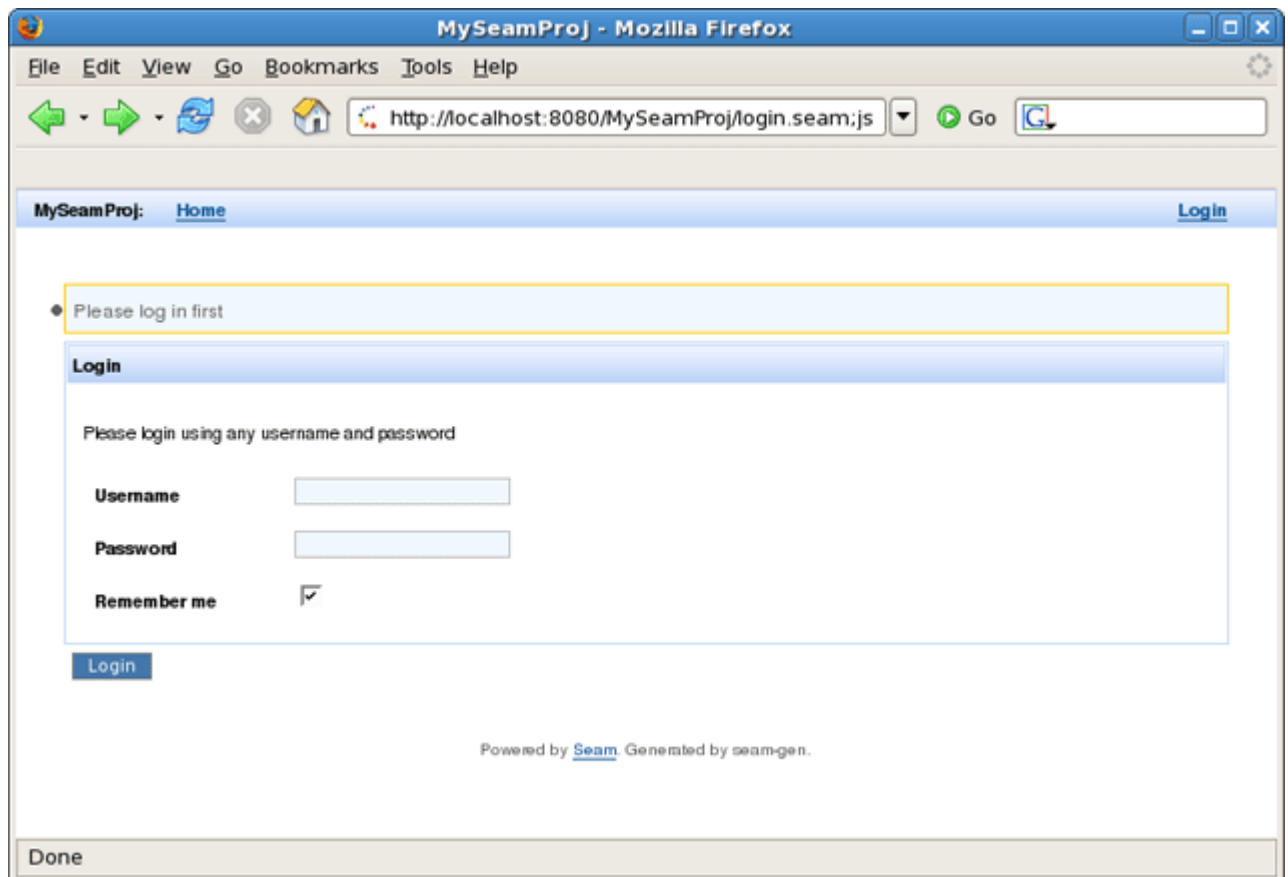
```
<!DOCTYPE pages PUBLIC
    "-//JBoss/Seam Pages Configuration DTD 1.2//EN"
    "http://jboss.com/products/seam/pages-1.2.dtd">

<pages no-conversation-view-id="/home.xhtml"
    login-view-id="/login.xhtml">
...

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

```
</pages>
```

Re-deploy the application and try the action button. The application redirects to the *login* page asking for login credentials. The method is invoked after you successfully logged in.



**Figure 3.18. Access Control for Action Methods**

## 3.7. Other relevant resources on the topic

Seam on JBoss: [Seam Framework](http://www.jboss.com/products/seam) [http://www.jboss.com/products/seam]

Ten Good Reasons to use Seam: [Why Seam](http://www.jboss.com/products/seam/whyseam) [http://www.jboss.com/products/seam/whyseam]

Getting Started: [Getting Started with JBoss Seam](http://labs.jboss.com/jbossseam/gettingstarted) [http://labs.jboss.com/jbossseam/gettingstarted]

Wiki: [JBoss Wiki](http://www.jboss.com/wiki/Wiki.jsp?page=JBossSeam) [http://www.jboss.com/wiki/Wiki.jsp?page=JBossSeam]

FAQ: [JBoss Seam FAQ](http://labs.jboss.com/jbossseam/faq/index.html) [http://labs.jboss.com/jbossseam/faq/index.html]

Downloads: [JBoss Seam Downloads](http://labs.jboss.com/jbossseam/download) [http://labs.jboss.com/jbossseam/download]

Jira: [Jira issue tracker](http://jira.jboss.org/jira/browse/JBSEAM) [http://jira.jboss.org/jira/browse/JBSEAM]

Rules Framework: [JBoss Rules](http://www.jboss.com/products/rules) [http://www.jboss.com/products/rules]

Seam Tools - New and Noteworthy: [What's new and noteworthy](http://fisheye.jboss.org/browse/~raw,r=3993/JBossTools/trunk/documentation/whatsnew/seam/seam-news-1.0.0.beta2.html) [http://fisheye.jboss.org/browse/~raw,r=3993/JBossTools/trunk/documentation/whatsnew/seam/seam-news-1.0.0.beta2.html]

Max Andersen's blogs: [Max's blog](http://blog.xam.dk/) [http://blog.xam.dk/], [In Relation To...](http://in.relation.to/Bloggers/Max) [http://in.relation.to/Bloggers/Max]



# 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.

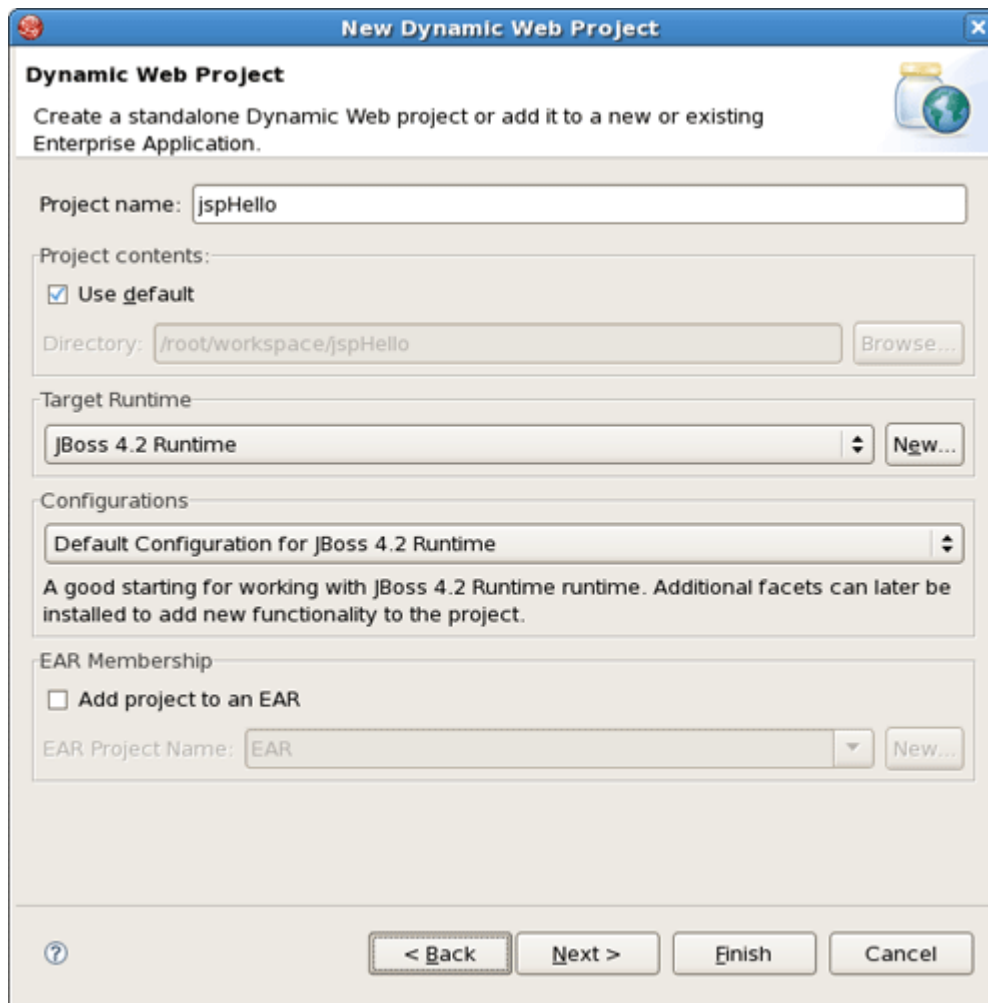
In this chapter you'll find out how to create a simple [JSP](http://java.sun.com/products/jsp/) application using the JBoss Developer Studio. The application will show a classic "Hello World!" on the page.

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

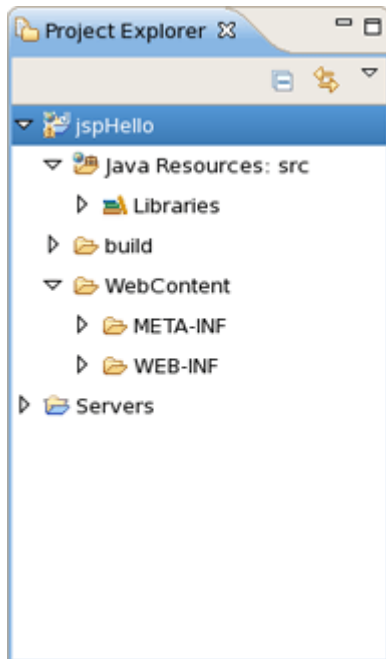
The main purpose of this section is to tell you about creation a Dynamic Web Project.

- Go to the menu bar and select *File > New > Project...*
- Select *Web > Dynamic Web Project* in the New Project dialog box
- Click *Next*
- Enter "jspHello" as a project name
- Leave everything else as is, and click *Finish*



**Figure 4.1. Create New Web Project**

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



**Figure 4.2. New Web Project**

## 4.2. Creating JSP Page

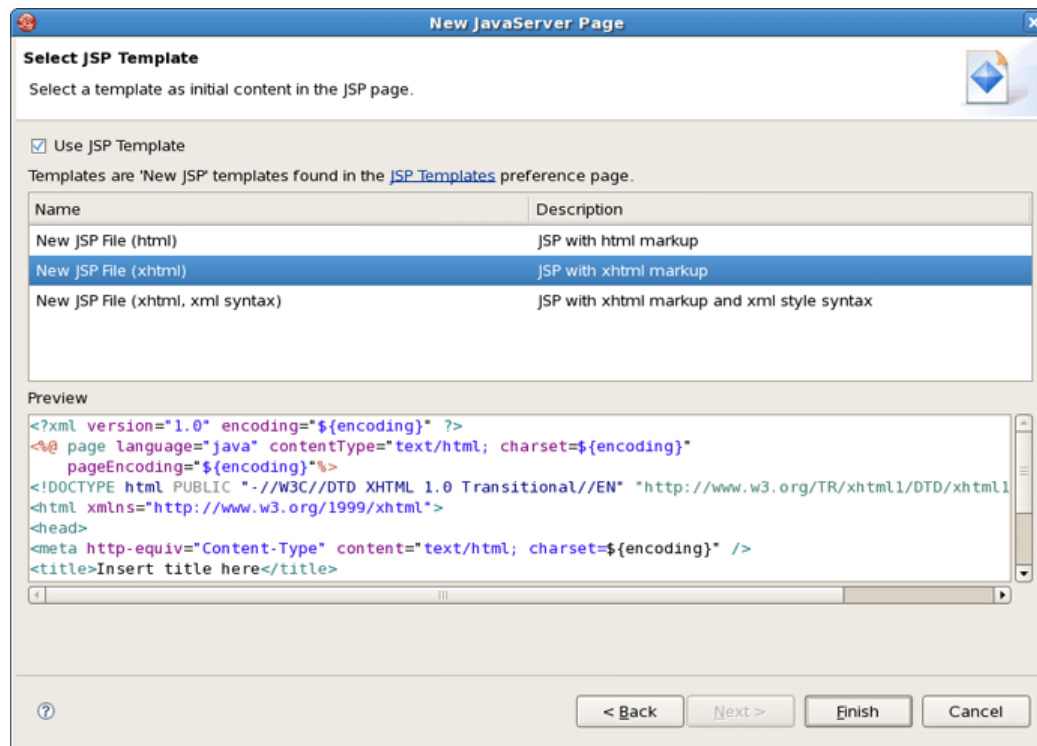
This section covers all the points how to create, edit and then preview JSP page.

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.

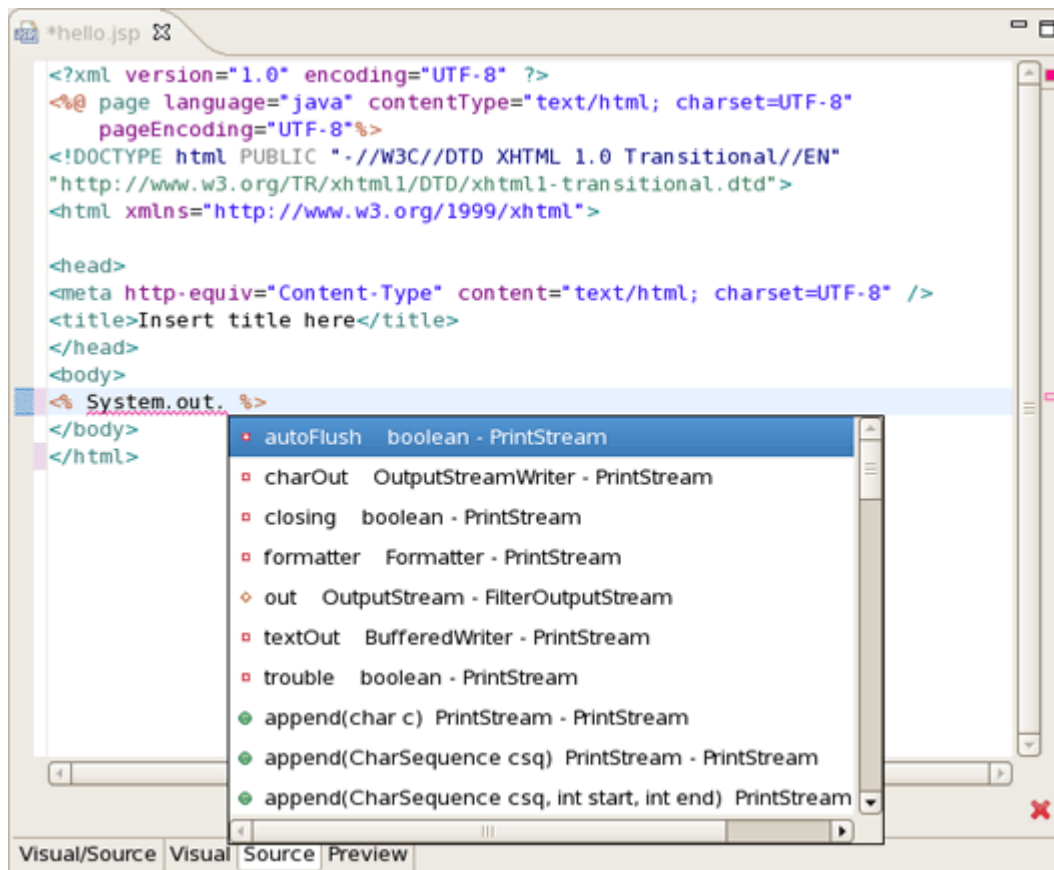
### 4.2.1. Editing a JSP Page

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

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

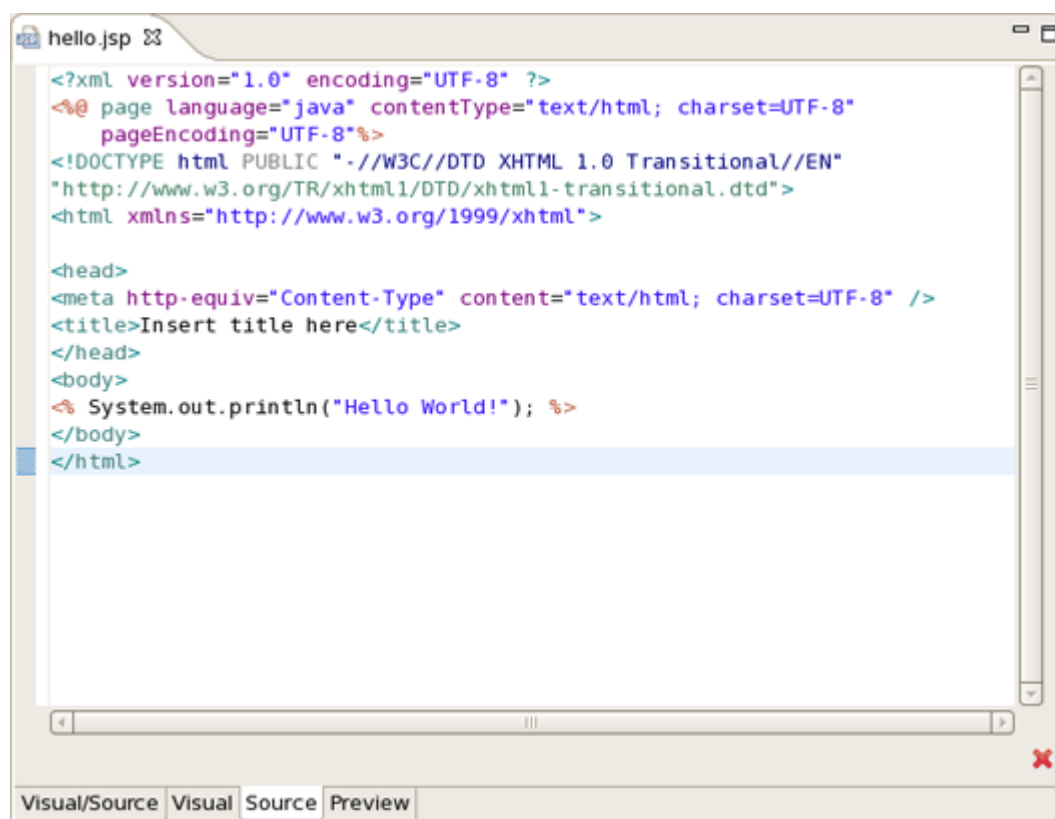
```
<% System.out.println("Hello World!"); %>
```

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



**Figure 4.4. Content Assist in JSP Page**

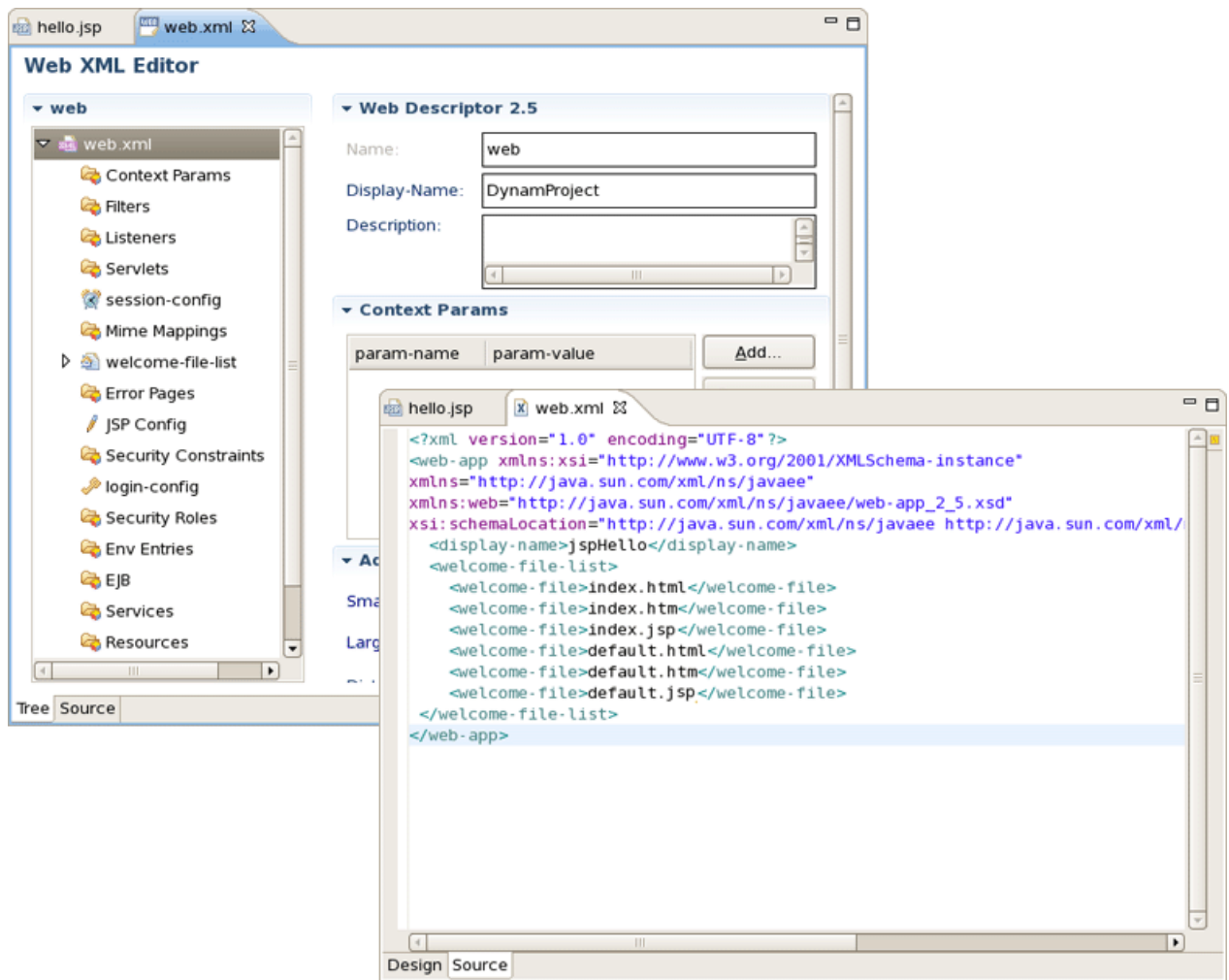
After changes made your hello.jsp page should look like this:



**Figure 4.5. Hello.jsp Page**

### 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 mode.
- Add the next code into **<welcome-file-list>** :

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

If you come back to design 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. You can clear the web.xml file, save it and you'll still be able to launch your application.

### 4.2.3. Deploying the project

While creating any web project you could experience a pain writing ant scripts and managing the packaging even if a developer is 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.

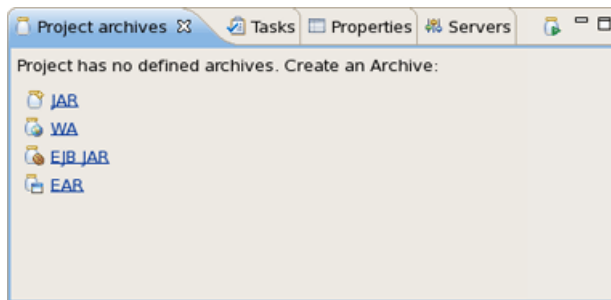
You can also create a war archive with JBDS's Archive Tools and export it to any web server.

#### 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:

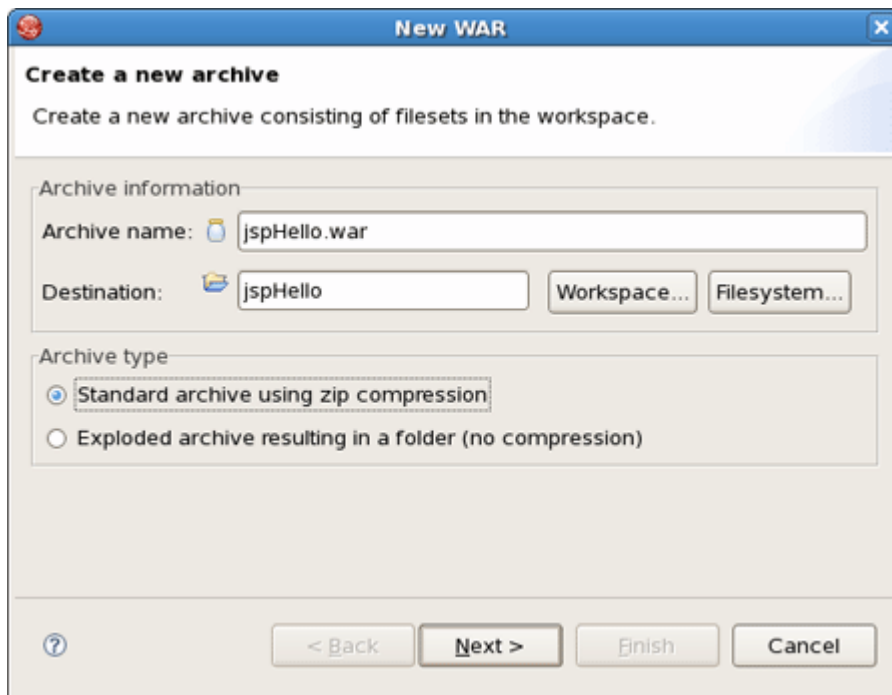


**Figure 4.7. Project Archives**

- Click, for example, *WAR* option to create war archive

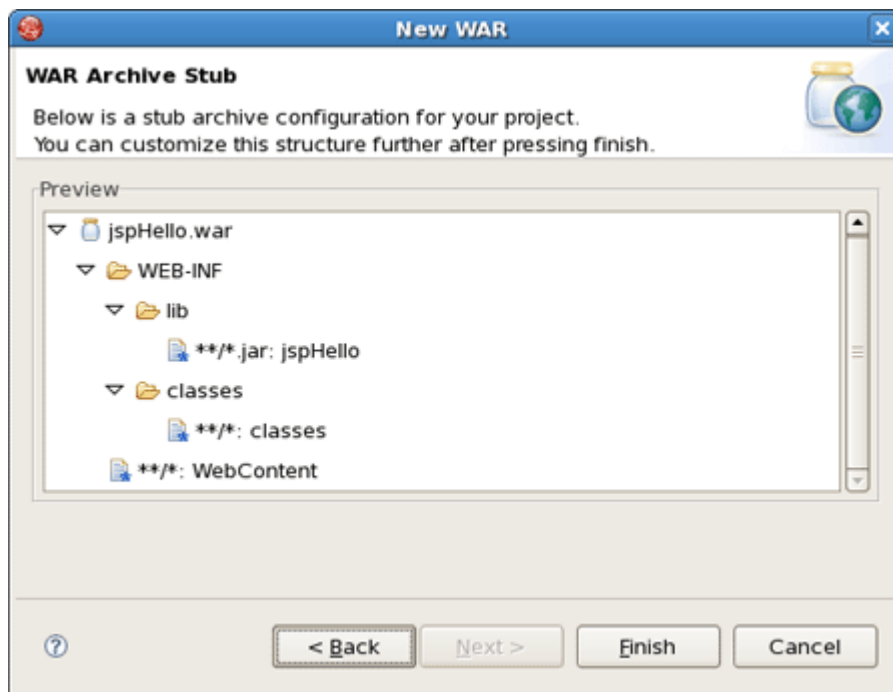
In the dialog New WAR you can see automatically selected default values





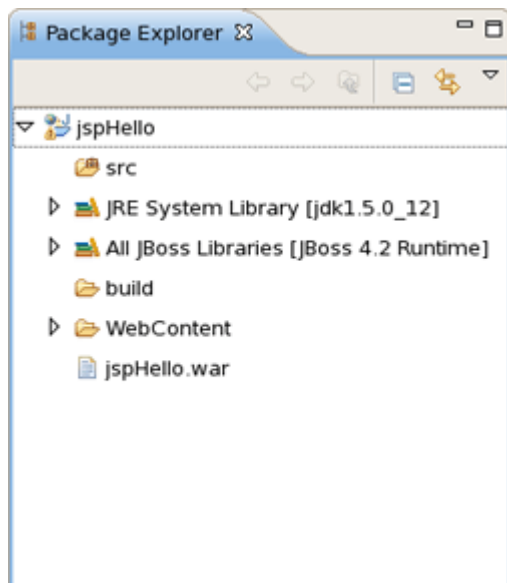
**Figure 4.8. New WAR Archive**

- Click *Next* to see a stub archive configuration for your project:

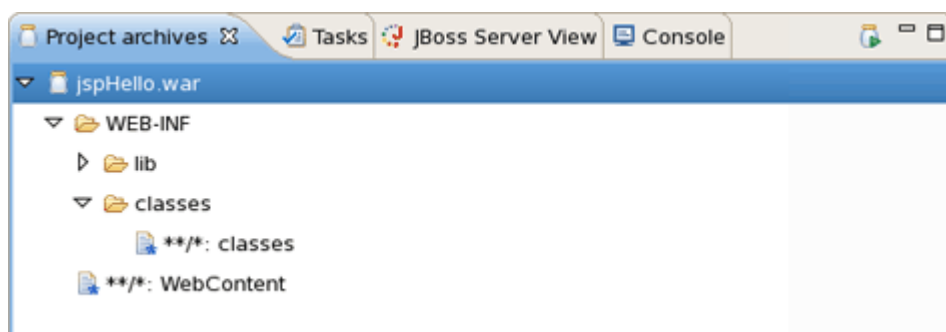


**Figure 4.9. Stub Archive Configuration**

- Click *Finish*. The *.war* file will appear in Package Explorer and in Project archives view as structure tree:

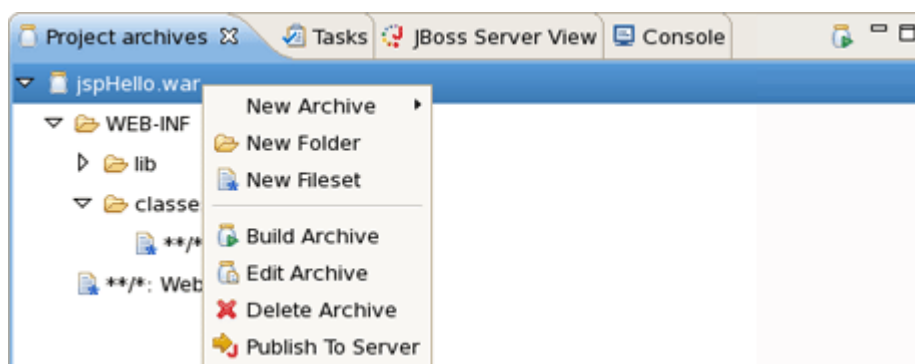


**Figure 4.10. Archive is Created**



**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. Any changes made in the application in exploded format will trigger a redeployment on the server.

#### 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:

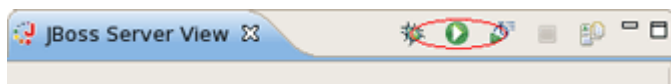
```
<%= new java.util.Date() %>
```

- 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

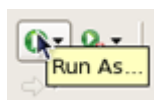
Let's now launch our project on server. We'll use JBoss server that is shipped with JBoss Developer Studio.

- Start JBoss server from JBoss Server view by clicking the **Start** icon.



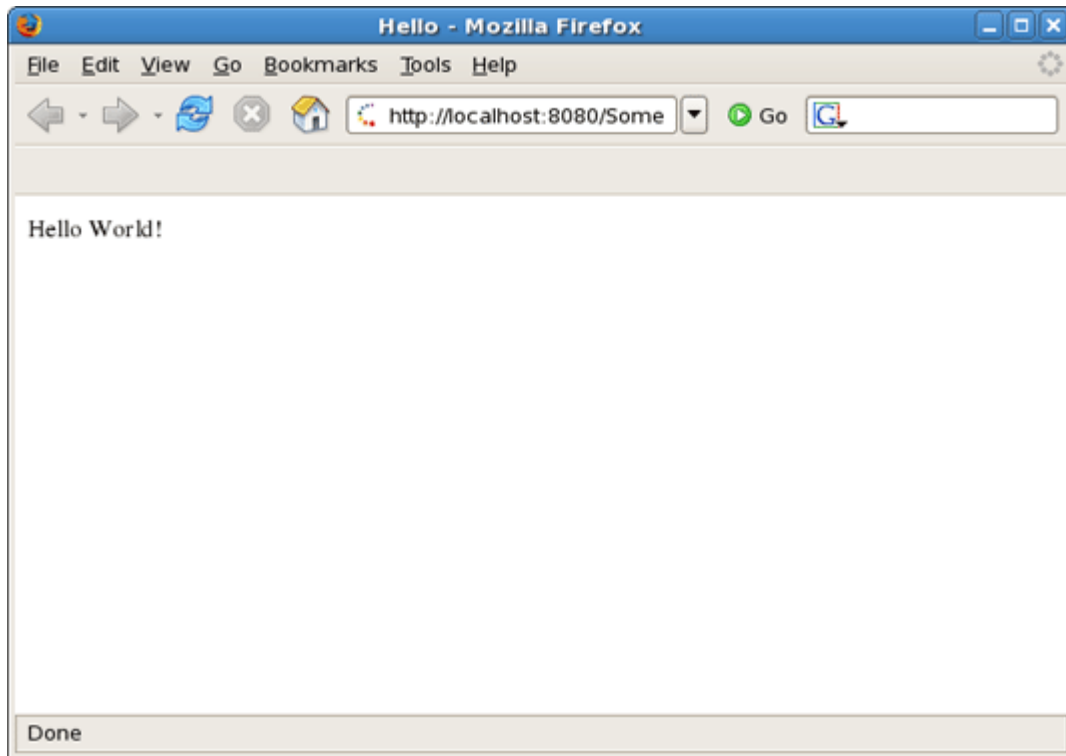
**Figure 4.13. Starting Server**

- 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*.



**Figure 4.14. Run Project**

You should see the next page in a browser :



**Figure 4.15. Running Project**

# 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.

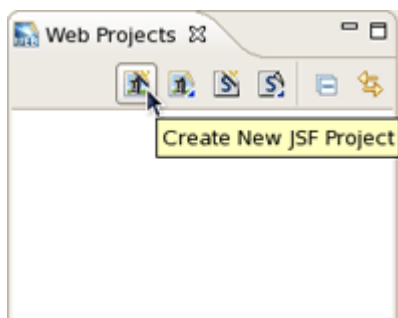
In this chapter you will see how to create a simple JSF application being based on "RAD" philosophy. We will create the familiar Guess Number application. The scenario is the following. You are asked to guess a number between 0 and 100. If the guess is correct, a success page is displayed with a link 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:

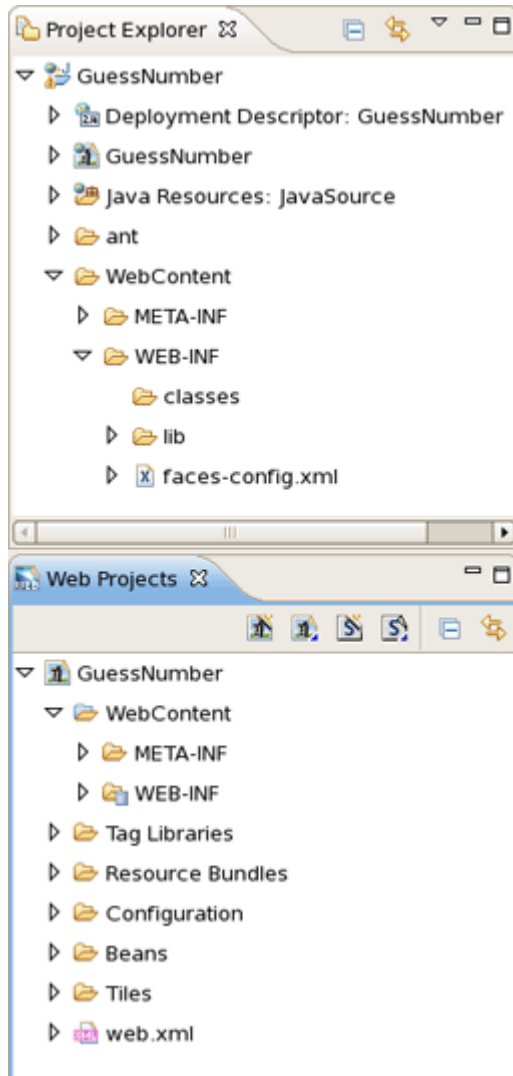
- In Web Projects View (if it is not open select *Window > Show View > Others > JBoss Tools Web > Web Projects View*) click *Create New JSF Project* button.



**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 Views. 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**

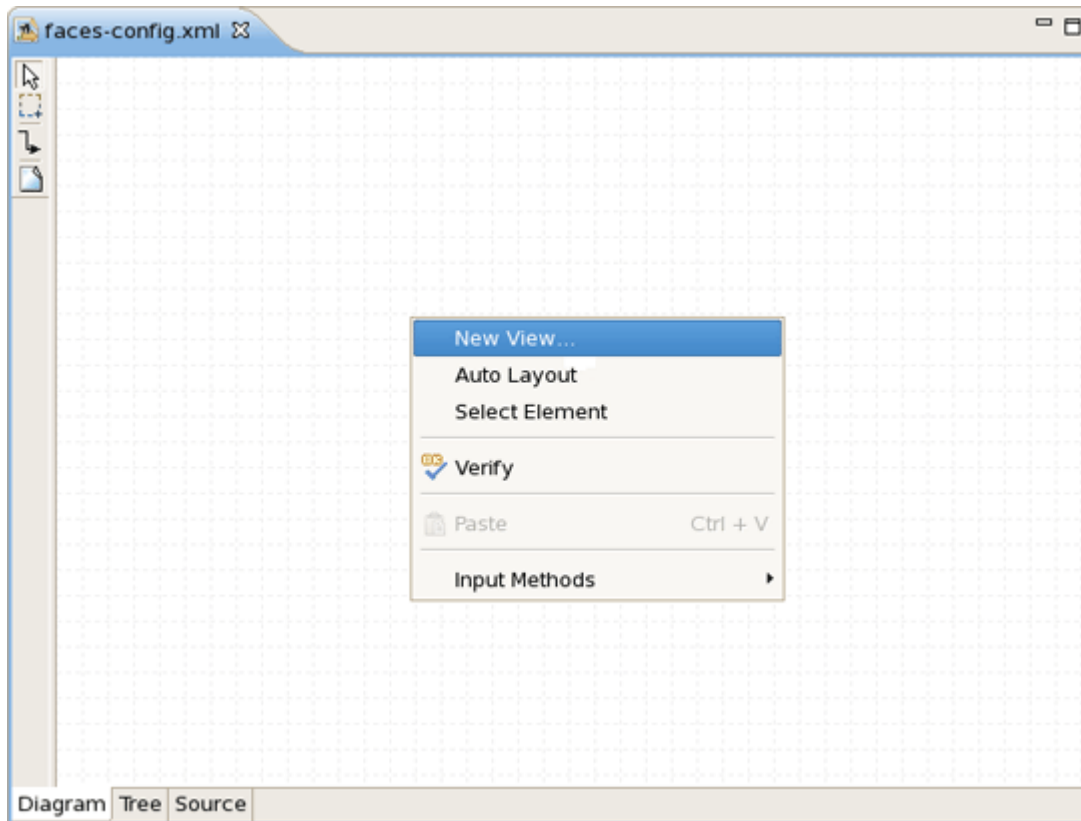
As the project has been set up, new JSP pages should be created now.

## 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.

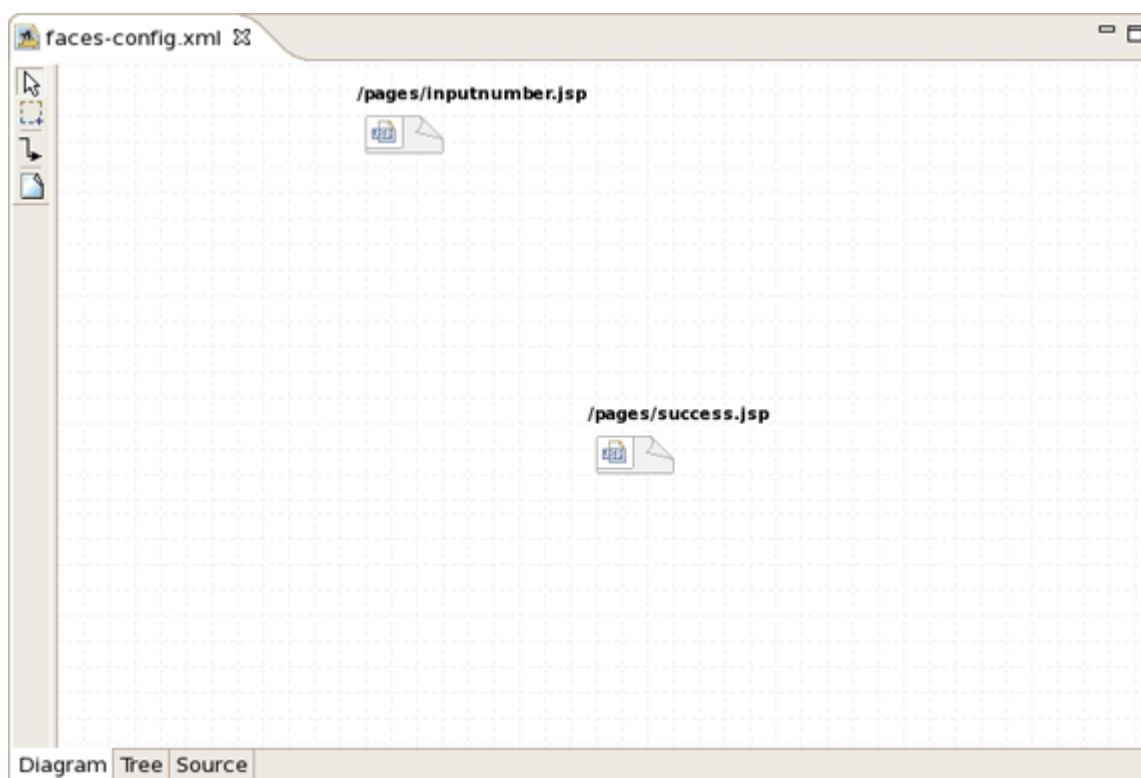
- Open *faces-config.xml* file
- Right click anywhere on the diagram mode
- From the context menu select *New View*



**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.

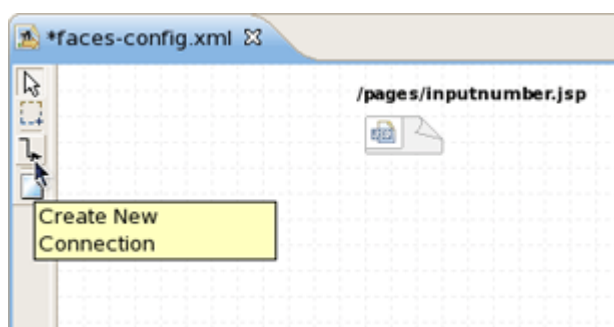


**Figure 5.4. New Views**

## 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

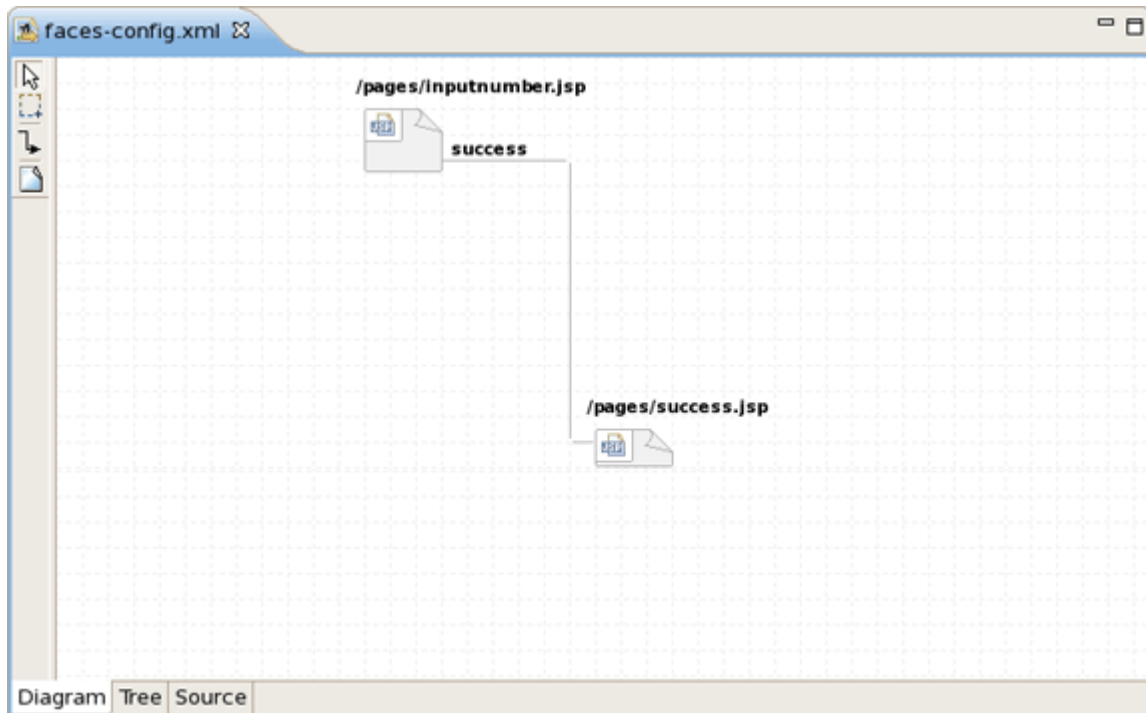


**Figure 5.5. Create Connection**

- 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.

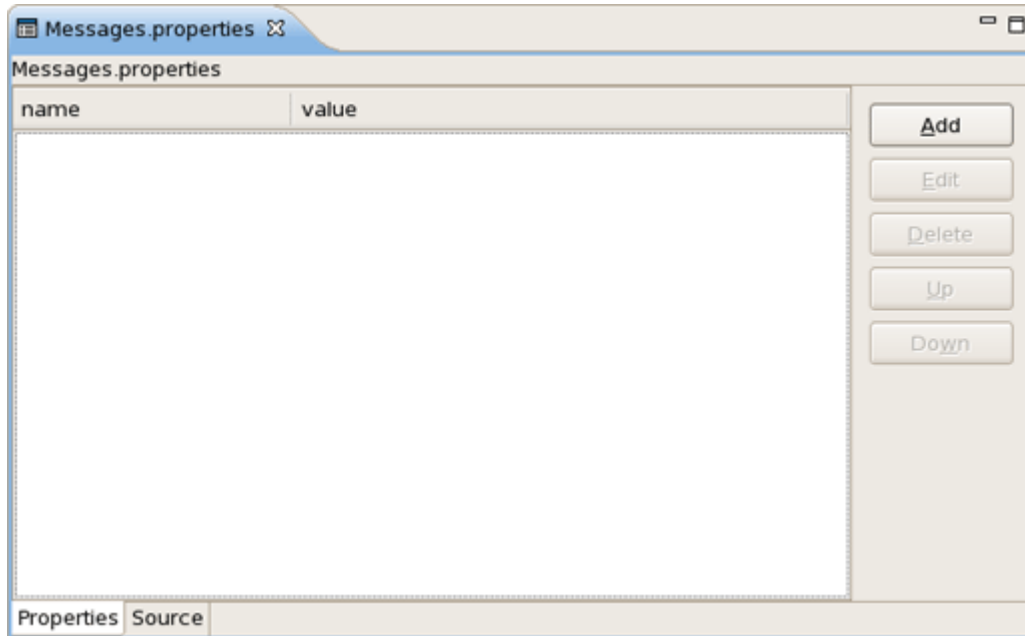
It is a good idea to keep your resources inside the *JavaSource* folder, where you keep your *.java* files. Every time you build the project, all *.properties* files will then be copied to the *classes* folder by default.

- 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.



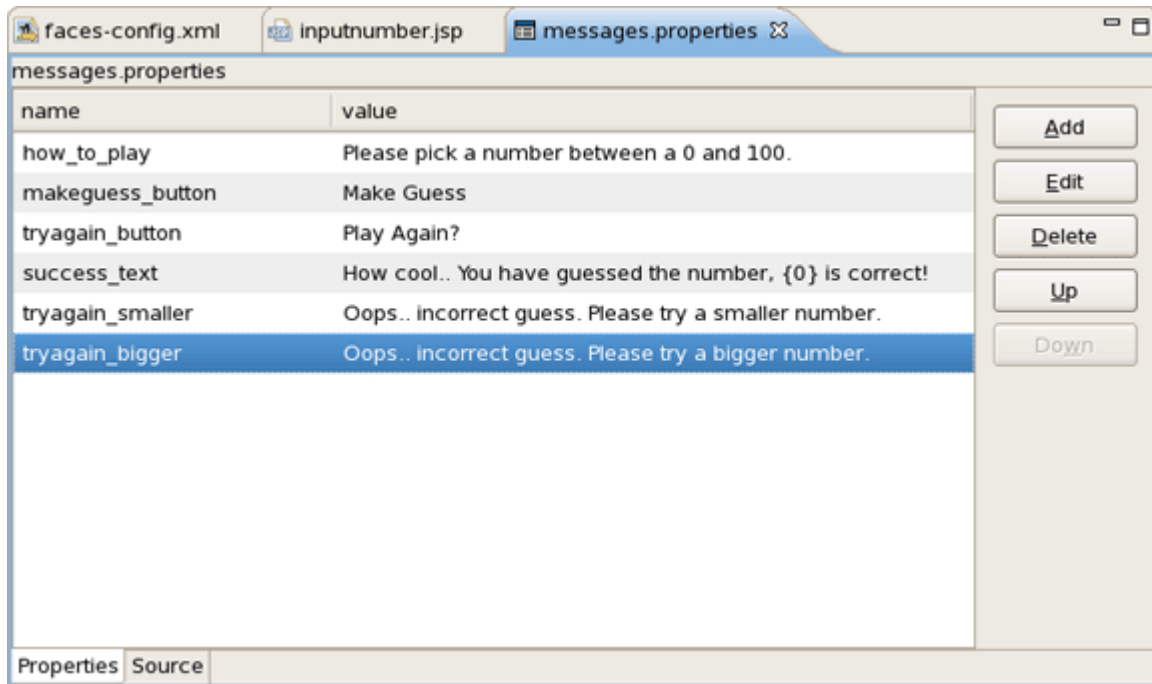
**Figure 5.7. Messages.properties File**

- Click *Add* button
- 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
trayagain_button=Play Again?
success_text=How cool.. You have guessed the number, {0} is correct!
trayagain_smaller=Oops..incorrect guess. Please try a smaller number.
trayagain_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**

## 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

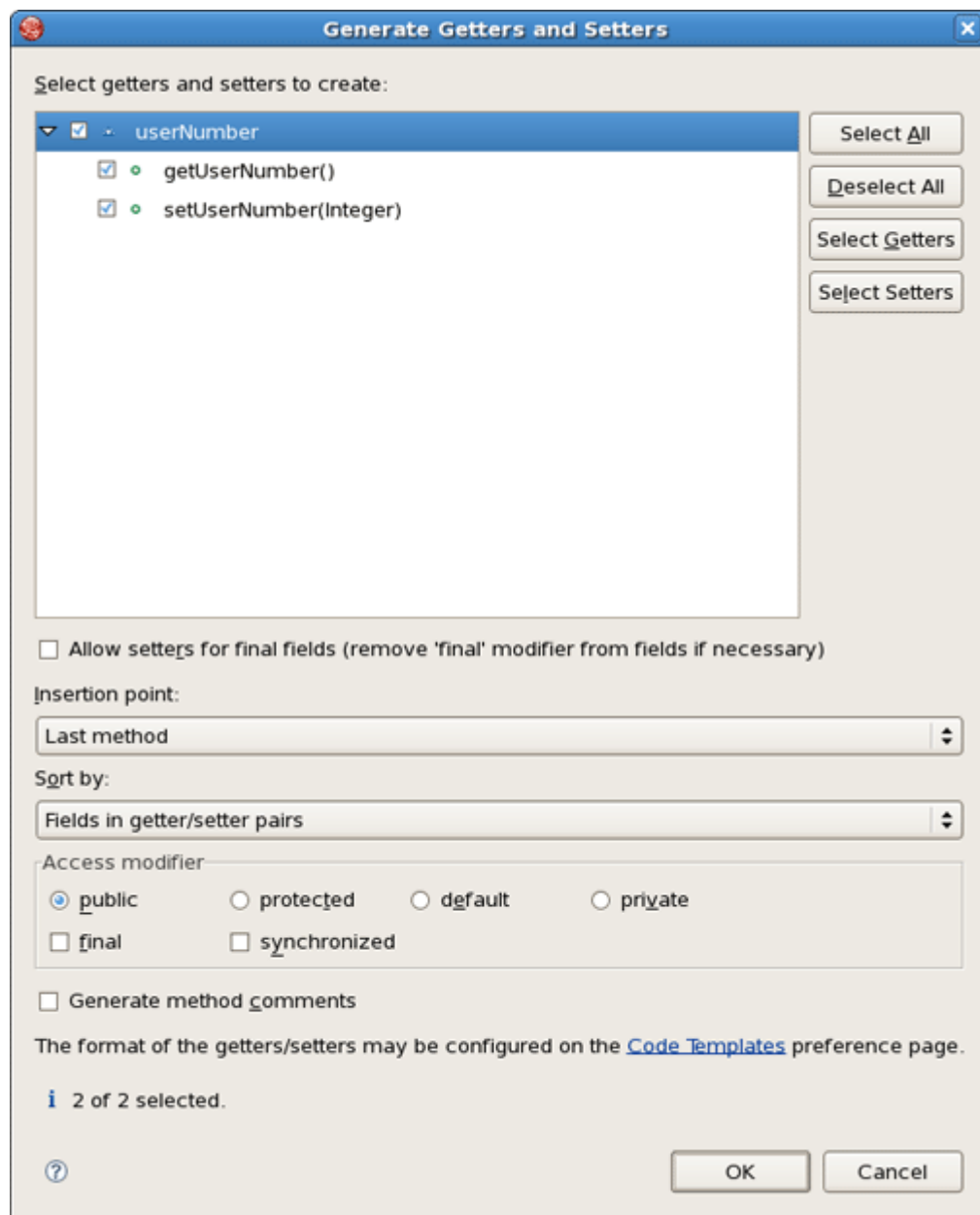
A java bean is created.

- 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 5.9. 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.Locale;
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.Locale;
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 finish 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.

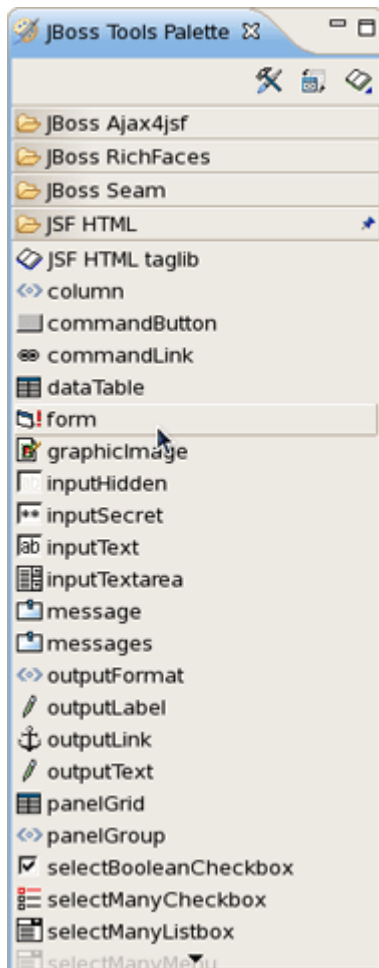
- Open `inputnumber.jsp` by double-clicking on the `/pages/inputnumber.jsp` icon



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.

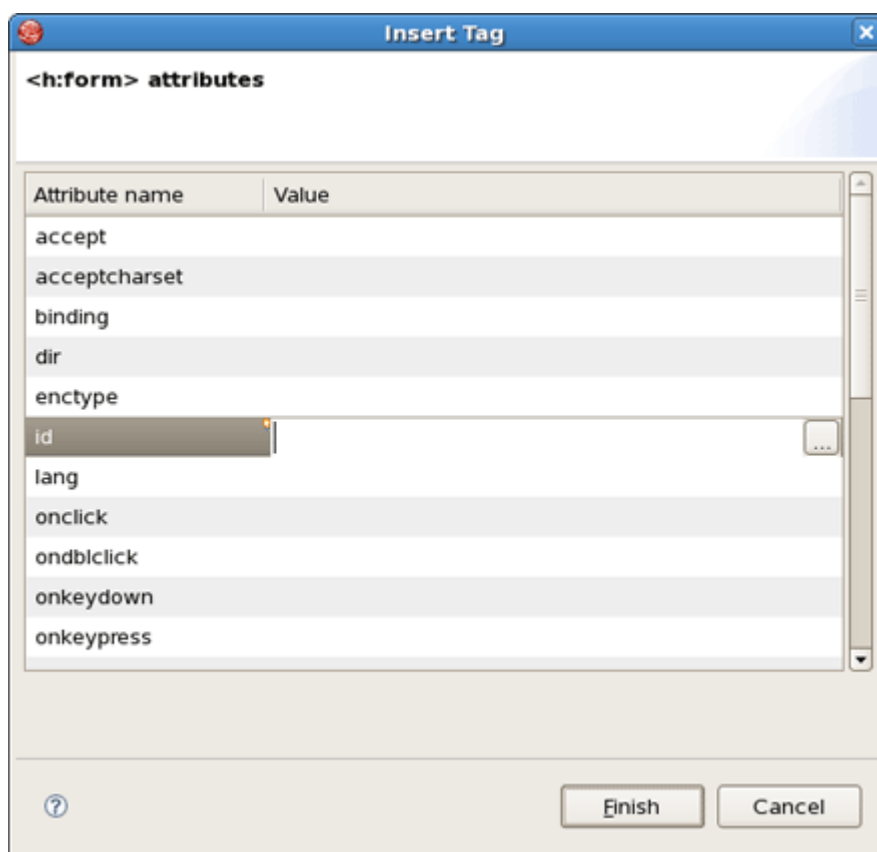
At the beginning it's necessary to create a **<h:form>** component where all others components are put.

- 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 5.10. Insert h:form**

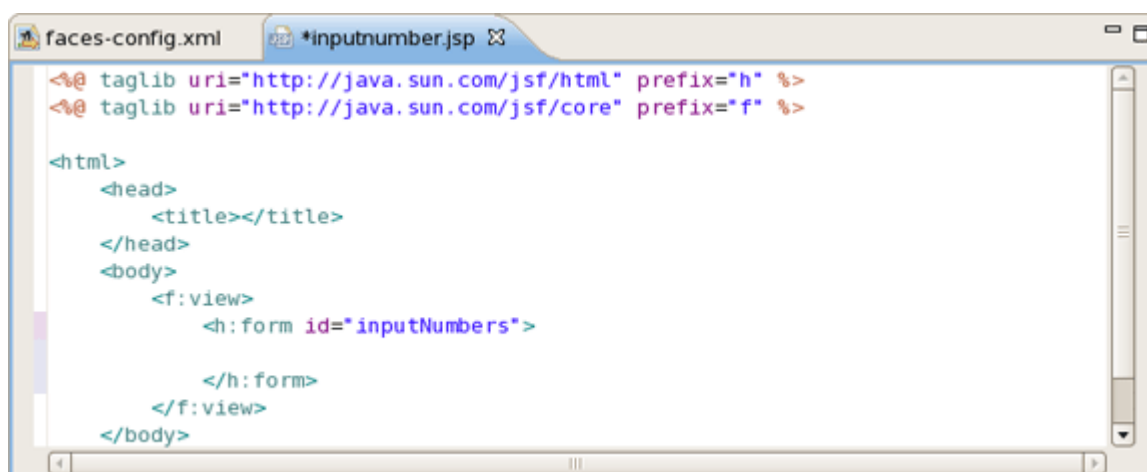
- In the dialog Insert Tag select *id* and click on this line below the value header. A blinking cursor will appear in a input text field inviting to enter a value of *id*



**Figure 5.11. Define Id of Form**

- Type `inputNumbers` and click `Finish`

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



**Figure 5.12. 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:

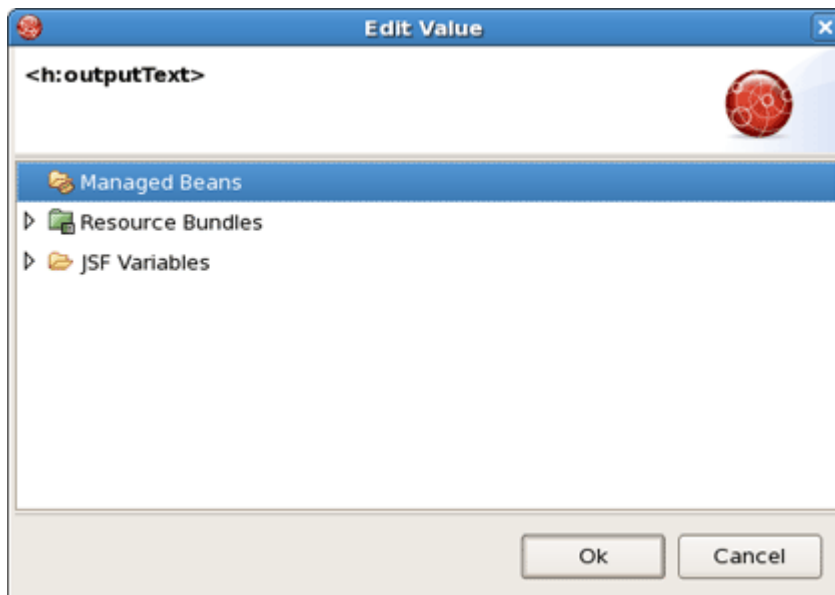
```
<f:loadBundle basename="game.messages" var="msg"/>
```

As always JBDS provides code assist:

### Figure 5.13. Code Assist

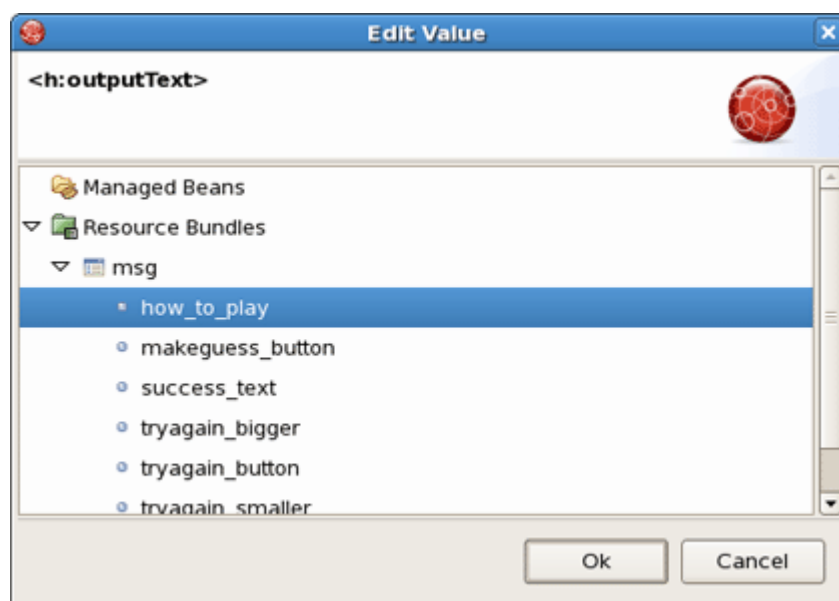
- Switch to Visual tab, so it could be possible to work with the editor completely in its WYSIWYG mode
- Click on *outputText*, drag the cursor over to the editor, and drop it inside the blue box in the editor
- Select *value* and click on this line below "value" header
- Click ... button next to the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

JBDS will nicely propose you to choose within available values:



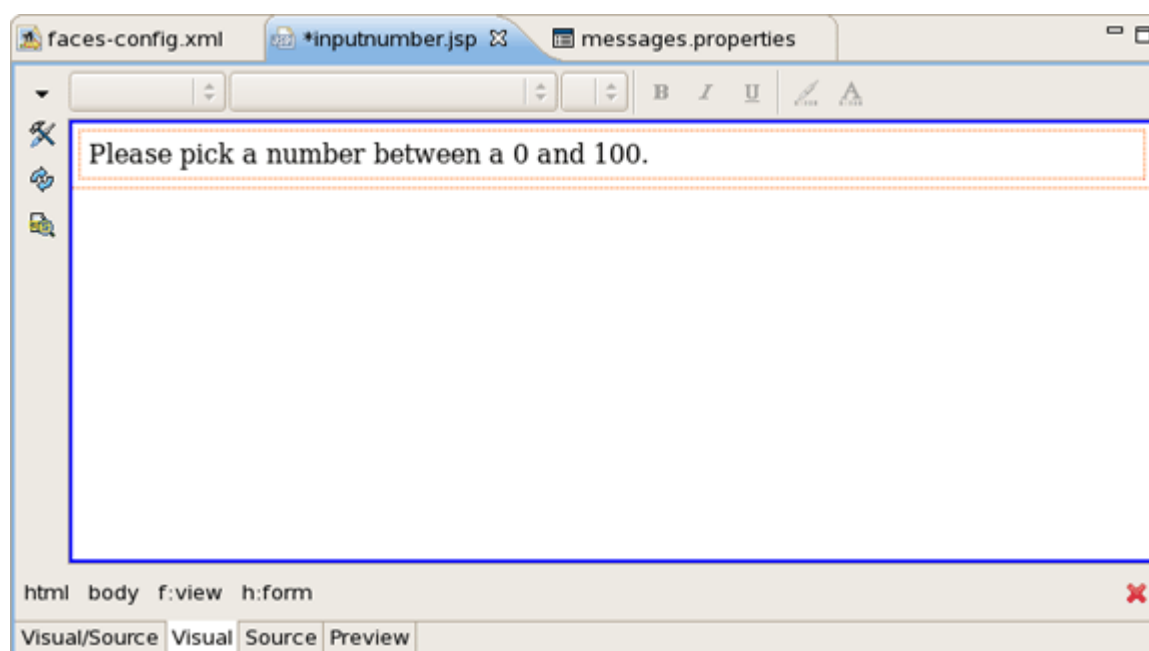
### Figure 5.14. Choose Value

- Expand *Resource Bundles > msg*
- Select *how\_to\_play* value and click *Ok*. Then click *Finish*



**Figure 5.15. Selecting Value**

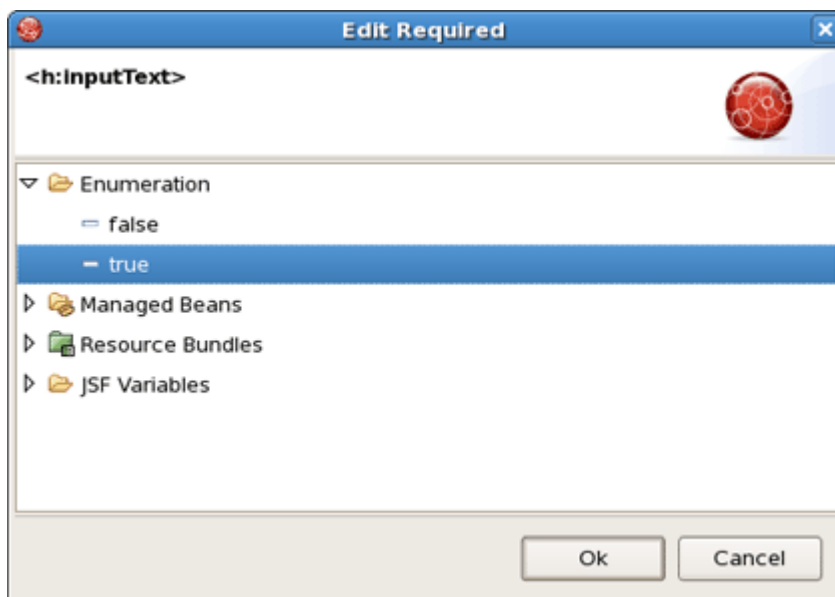
The text will appear on the page:



**Figure 5.16. Created OutputText Component**

- Switch to Source mode and insert `<br/>` 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 `<br/>` 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 5.17. 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 **<br/>** 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>
<f:view>
    <h:form id="inputNumbers">
        <h:outputText value="#{msg.how_to_play}"/>
        <br/>
        <h:inputText id="userNumber" value="#{NumberBean.userNumber}" required="true">
            <f:validateLongRange minimum="0" maximum="100"/>
        </h:inputText>
        <h:commandButton value="#{msg.makeguess_button}" action="#{NumberBean.checkGuess}"/>
    </h:form>
</f:view>
</html>
```

```

<h:messages style="color: blue"/>
<br/>
<h:inputText id="userNumber" value="#{NumberBean.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>
</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:

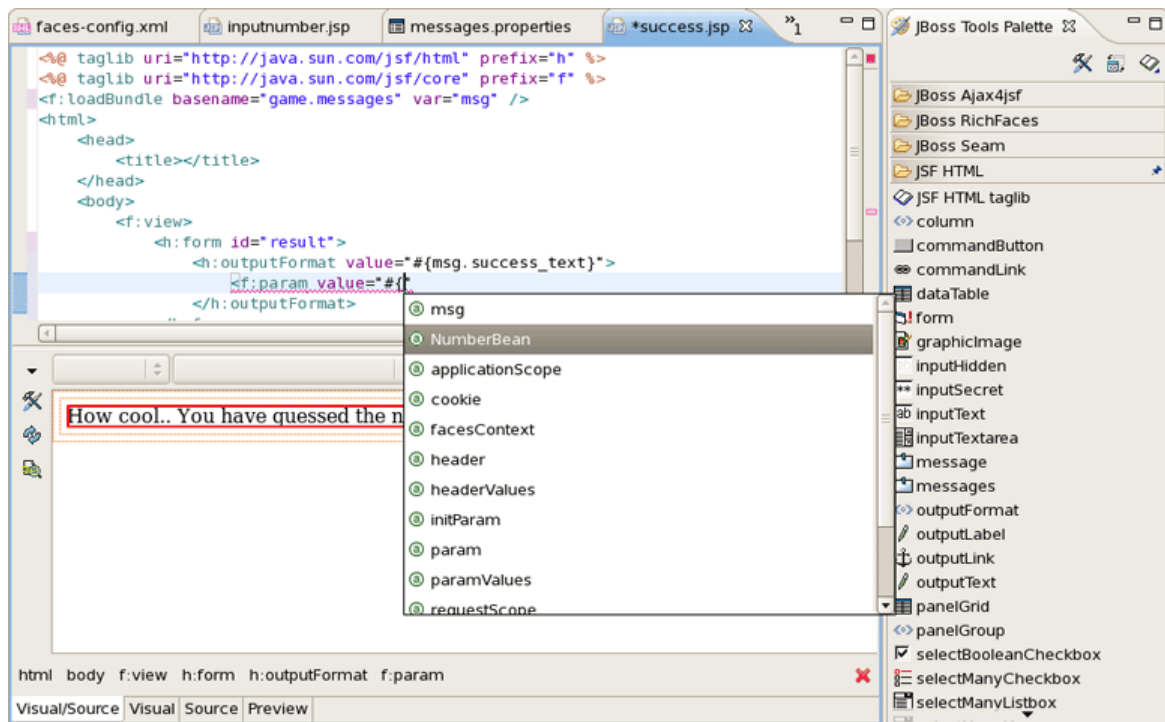
```

<%@ 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>
<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>
</html>

```

Again you can use code assist provided by JBDS when editing jsp page:



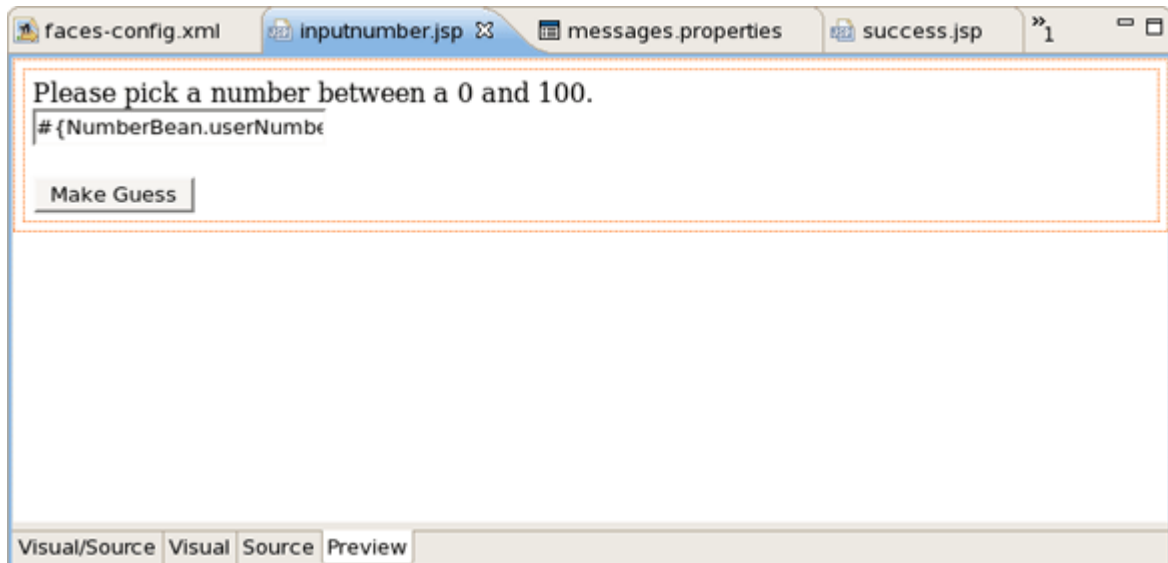
**Figure 5.18. Code Assist for <f:param>**

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 5.19. 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:

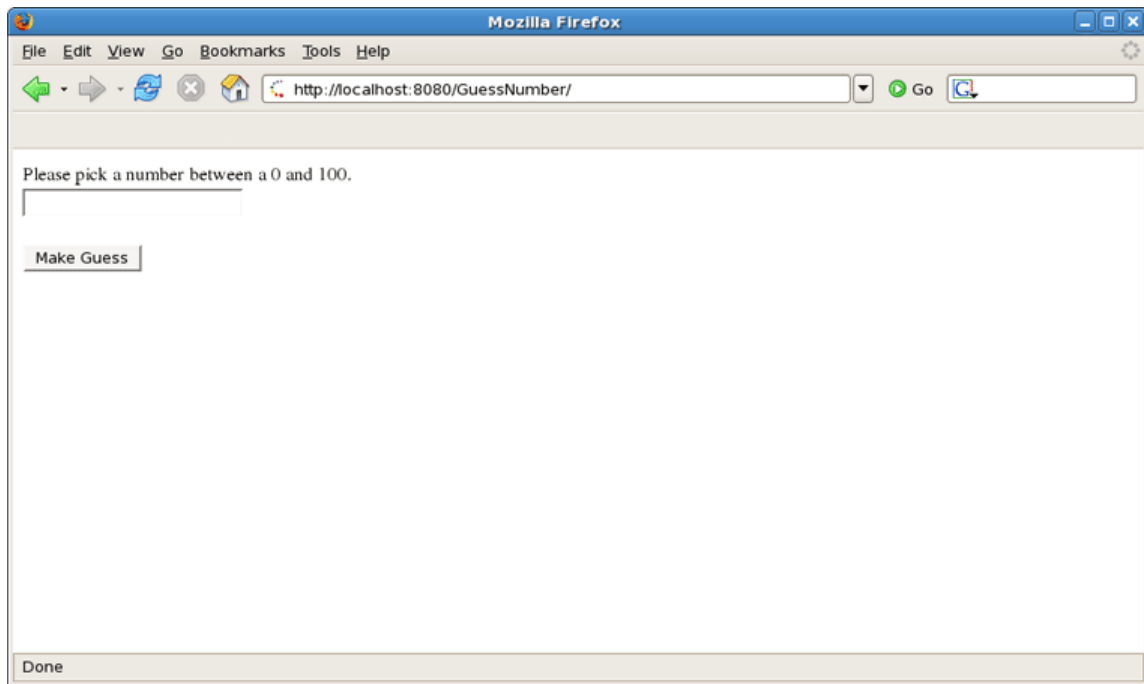
```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<body>
  <jsp:forward page="/pages/inputnumber.jsf" />
</body>
</html>
```

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.

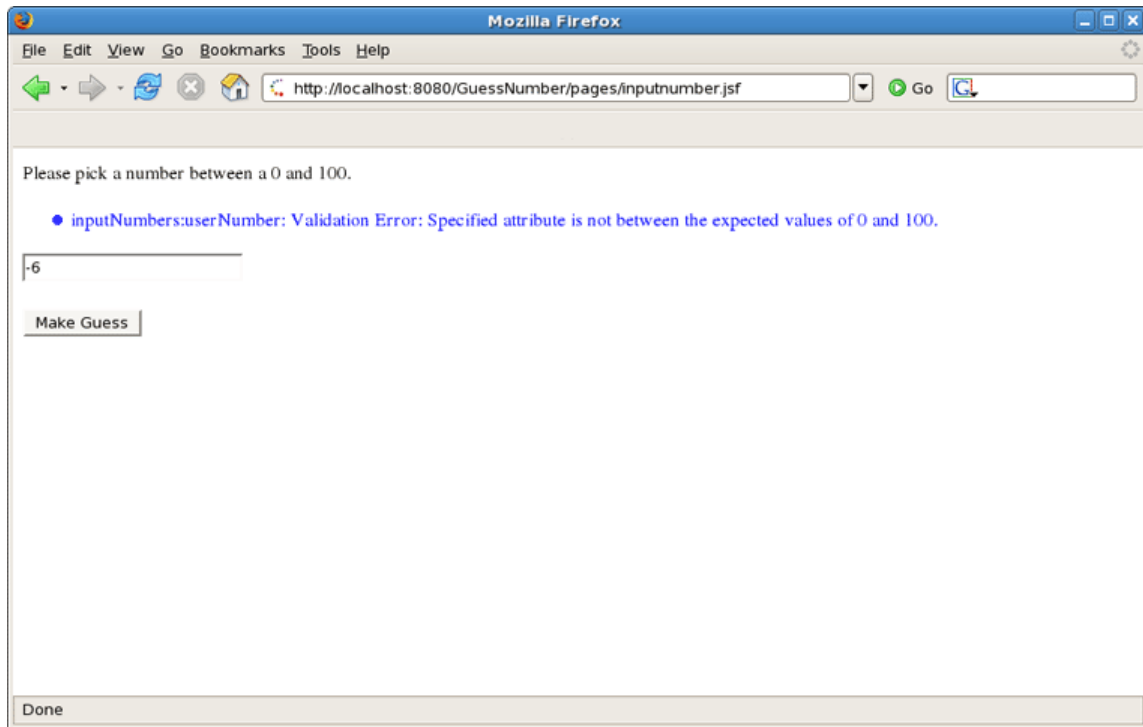
## 5.9. Running the Application

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

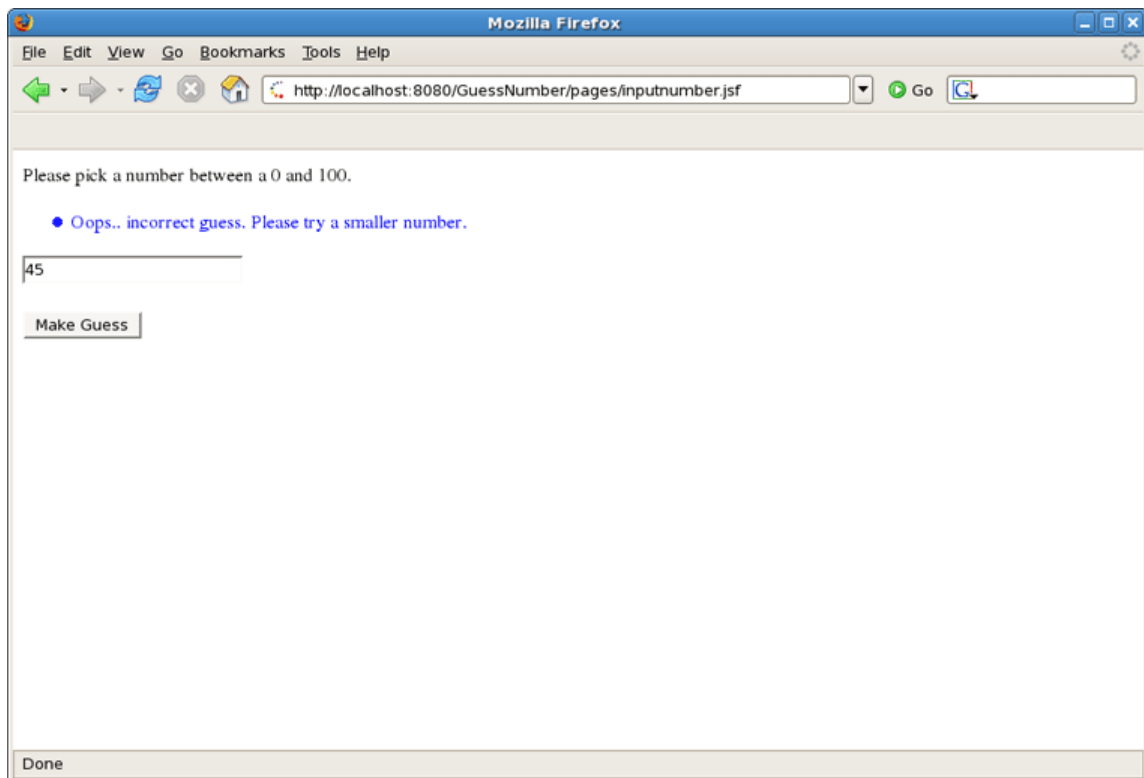
- Start up JBoss server by clicking on the *Start* icon in JBoss Server view. (If JBoss is already running, stop it by clicking on the red icon and then start it again. After the messages in the Console tabbed view stop scrolling, JBoss is available)
- Right-click on project *Run AS > Run on Server*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



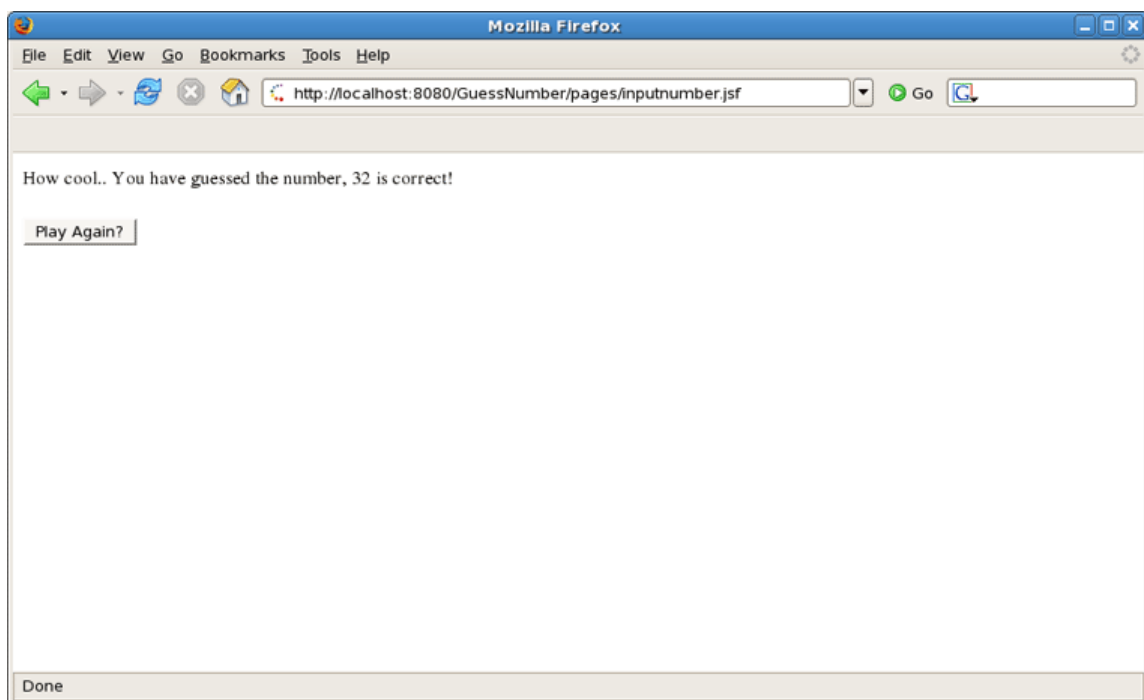
**Figure 5.20. You are Asked to Enter a Number Between 0 and 100**



**Figure 5.21. Your Input is Validated and an Error Message is Displayed if Invalid Input was Entered**



**Figure 5.22. After You Enter a Guess, the Application Tells You Whether a Smaller or a Larger Number Should be Tried**



**Figure 5.23. Your Guess is Correct**

## Further Reading

**JSF Tools Reference Guide** ([html](#)) [[../../jsf\\_tools\\_ref\\_guide/en/html/index.html](#)] ([html single](#)) [[../../jsf\\_tools\\_ref\\_guide/en/html\\_single/index.html](#)] ([PDF](#)) [[../../jsf\\_tools\\_ref\\_guide/en/pdf/Seam\\_Dev\\_Tools\\_Reference\\_Guide.pdf](#)]

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.

**JSF Tools Tutorial** ([html](#)) [[../../jsf\\_tools\\_tutorial/en/html/index.html](#)] ([html single](#)) [[../../jsf\\_tools\\_tutorial/en/html\\_single/index.html](#)] ([PDF](#)) [[../../jsf\\_tools\\_tutorial/en/pdf/Seam\\_Dev\\_Tools\\_Reference\\_Guide.pdf](#)]

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.

**Struts Tools Reference Guide** ([html](#)) [[../../struts\\_tools\\_ref\\_guide/en/html/index.html](#)] ([html single](#)) [[../../struts\\_tools\\_ref\\_guide/en/html\\_single/index.html](#)] ([PDF](#)) [[../../struts\\_tools\\_ref\\_guide/en/pdf/Seam\\_Dev\\_Tools\\_Reference\\_Guide.pdf](#)]

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** ([html](#)) [[../../struts\\_tools\\_tutorial/en/html/index.html](#)] ([html single](#)) [[../../struts\\_tools\\_tutorial/en/html\\_single/index.html](#)] ([PDF](#)) [[../../struts\\_tools\\_tutorial/en/pdf/Seam\\_Dev\\_Tools\\_Reference\\_Guide.pdf](#)]

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.

**Seam Dev Tools Reference Guide** ([html](#)) [[../../seam/en/html/index.html](#)] ([html single](#)) [[../../seam/en/html\\_single/index.html](#)] ([PDF](#)) [[../../seam/en/pdf/Seam\\_Dev\\_Tools\\_Reference\\_Guide.pdf](#)]

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** ([html](#)) [[../../jsf/en/html/index.html](#)] ([html single](#)) [[../../jsf/en/html\\_single/index.html](#)] ([PDF](#)) [[../../jsf/en/pdf/Visual\\_Web\\_Tools\\_Reference\\_Guide.pdf](#)]

**JBoss Server Manager Reference Guide** ([html](#)) [[../../as/en/html/index.html](#)] ([html single](#)) [[../../as/en/html\\_single/index.html](#)] ([PDF](#)) [[../../as/en/pdf/server\\_manager\\_guide.pdf](#)]

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** ([html](#)) [[../../jbpm/en/html/index.html](#)] ([html single](#)) [[../../jbpm/en/html\\_single/index.html](#)] ([PDF](#)) [[../../jbpm/en/pdf/jBPM\\_Tools\\_Ref.pdf](#)]

**Hibernate Tools Reference Guide** ([html](#)) [[../../hibernatetools/en/html/index.html](#)] ([html single](#)) [[../../hibernatetools/en/html\\_single/index.html](#)] ([PDF](#)) [[../../hibernatetools/en/pdf/hibernate\\_tools.pdf](#)]

**Exadel Studio Migration Guide** ([html](#)) [[../../Exadel-migration/en/html/index.html](#)] ([html single](#)) [[../../Exadel-migration/en/html\\_single/index.html](#)] ([PDF](#)) [[../../Exadel-migration/en/pdf/exadel-migration.pdf](#)]

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