JSF Tools Reference Guide

Version: 3.3.0.M5

1. Introduction	. 1
1.1. Key Features of JSF Tools	1
1.2. Other relevant resources on the topic	. 2
2. JavaServer Faces Support	. 3
2.1. Facelets Support	. 4
2.1.1. Creating a JSF project with Facelets	. 4
2.1.2. Facelets components	. 6
2.1.3. Code assist for Facelets	. 6
2.1.4. Open On feature	. 8
3. Projects	11
3.1. Creating a New JSF Project	11
3.2. Importing Existing JSF Projects with Any Structure	17
3.3. Adding JSF Capability to Any Existing Project	17
3.4. Adding Your Own Project Templates	19
3.5. Relevant Resources Links	21
4. Web.xml Editor	23
5. JSF Configuration File Editor	25
5.1. Diagram view	25
5.2. Tree View	30
5.3. Source View	48
5.4. Editor Features	50
Edd Open Op	50
5.4.1. Open OIT	00
5.4.2. Code Assist	50
5.4.2. Code Assist	50 52
5.4.1. Open On 5.4.2. Code Assist	50 52 55
5.4.1. Open Off 5.4.2. Code Assist 5.4.3. Error Reporting 6. Managed Beans	50 52 55 55
 5.4.1. Open Off	50 52 55 55 64
 5.4.1. Open Off 5.4.2. Code Assist 5.4.3. Error Reporting 6. Managed Beans 6.1. Code Generation for Managed Beans 6.2. Add Existing Java Beans to a JSF Configuration File 7. Creation and Registration 	50 52 55 55 64 67
 5.4.1. Open Off 5.4.2. Code Assist 5.4.3. Error Reporting 6.4.3. Error Reporting 6.4.3. Error Reporting 6.4.4.4.4.4.4.4.4.4.4.4.4.4.4.4.4.4.4.4	50 52 55 55 64 67 67
 5.4.1. Open Off 5.4.2. Code Assist 5.4.3. Error Reporting 6. Managed Beans 6.1. Code Generation for Managed Beans 6.2. Add Existing Java Beans to a JSF Configuration File 7. Creation and Registration 7.1. Create and Register a Custom Converter 7.2. Create and Register a Custom Validator 	50 52 55 55 64 67 67 74
 5.4.1. Open Off 5.4.2. Code Assist 5.4.3. Error Reporting 6.4.3. Error Reporting 6.1. Code Generation for Managed Beans 6.2. Add Existing Java Beans to a JSF Configuration File 7. Creation and Registration 7.1. Create and Register a Custom Converter 7.2. Create and Register a Custom Validator 7.3. Create and Register Referenced Beans 	50 52 55 55 64 67 67 74 82

Introduction

JSF Tools are especially designed to support JSF and JSF-related technologies. JSF Tools provide extensible tools for building JSF-based applications as well as adding JSF capabilities to existing web projects, importing JSF projects and choosing any JSF implementation while developing JSF application.

This guide provides the information on JSF tooling you need to allow you to quickly develop JSF applications with far fewer errors.

1.1. Key Features of JSF Tools

The table below lists the functionality provided by the JSF Tools.

Feature	Benefit	Chapter
JSF and Facelets support	Step-by-step wizards for creating new JSF and Facelets projects with a number of predefined templates, importing existing ones and adding JSF capabilities to non-JSF web projects.	Chapter 2, JavaServer Faces Support
Flexible and customizable project template management	Jump-start development with the supplied templates or easily create customized templates for re-use.	Chapter 3, Projects
Support for JSF Configuration File	Work on a file using three modes: diagram, tree and source. Automatic synchronization between the modes and full control over the code. Easily move around the diagram using the Diagram Navigator.	Chapter 5, JSF Configuration File Editor
Support for Managed Beans	Adding new managed beans, generating code for attributes, properties and getter/setter methods.	Chapter 6, Managed Beans
Support for Custom Converters and Validators	Fast creation of custom converters and validators with a tree view of the faces-config.xml file.	Chapter 7, Creation and Registration
Verification and Validation	All errors will be immediately reported by verification feature, no matter in what view you are working. Constant validation and error checking allows you to catch many of the errors during development process that significantly reduces development time.	Chapter 8, JSF Project Verification

Table 1.1. Key Functionality for JSF Tools

1.2. Other relevant resources on the topic

All JBoss Developer Studio and JBoss Tools release documentation can be found on the *RedHat Documentation* [http://docs.redhat.com/docs/en-US/JBoss_Developer_Studio/ index.html] website.

Nightly documentation builds are available at *http://download.jboss.org/jbosstools/nightly-docs* [http://download.jboss.org/jbosstools/nightly-docs/].

JavaServer Faces Support

JSF Tools does not lock you into any one JavaServer Faces implementation. You can always specify the desired JavaServer Faces implementation while creating a new JSF project (see *Section 3.1, "Creating a New JSF Project"*), adding JSF capability to any existing Eclipse project (see *Section 3.3, "Adding JSF Capability to Any Existing Project"*), or importing existing JSF projects (see *Section 3.2, "Importing Existing JSF Projects with Any Structure"*).

At this point the special wizard will prompt you to specify an appropriate JSF environment. It may be JSF 1.1.02 RI, JSF 1.2 or JSF 2.0. The wizard also lets you select JSF implementation with a component orientation such as JSF 1.2 with Facelets or MyFaces 1.1.4.

0	New JSF Project 🛛 🕅
Create JSF Proje	ct 🌍
Project Name*	JSFProject
	✓ Use default path*
Location*	/home/irooskov/Work/JBDS/workspace/JSFProject
JSF Environment*	JSF 2.0 \$
Template*	JSFKickStartWithoutLibs
4 4	
?	< Back Next > Cancel Finish

Figure 2.1. Choosing JSF Environment

After specifying an appropriate JSF environment, all the required libraries associated with the selected version will be added to your project.

2.1. Facelets Support

In this section we will focus all the concepts that relate to working with Facelets.

Facelets extend JavaServer Faces by providing a lightweight framework that radically simplifies the design of JSF presentation pages. Facelets can be used in a variety of ways that we will consider further in this section.

2.1.1. Creating a JSF project with Facelets

If you want to build an application using Facelets, create a project with Facelets based on version 1.2 of the JSF Reference Implementation, i. e. select the **JSF 1.2 with Facelets** option in the **JSF Environment** section of the **New JSF Project** wizard.

9	New JSF Project 🛛 🗙
Create JSF Projec	t 🚳
Project Name*	JSFProjectwithFacelets ✓ Use default path*
Location*	t/workspaces/workspace-jbds4/JSFProjectwithFacelets Browse
JSF Environment*	JSF 1.2 with Facelets
Template*	FaceletsBlankWithoutLibs \$
?	< Back Next > Cancel Finish

Figure 2.2. Choosing Facelets Environment

Once you have selected the environment, it is possible to specify one of three available templates:

()	New JSF Project 🛛 🕅
Create JSF Proje	ct 🍥
Project Name*	JSFProjectwithFacelets
	✓ Use default path*
Location*	t/workspaces/workspace-jbds4/JSFProjectwithFacelets Browse
JSF Environment*	JSF 1.2 with Facelets
Template*	FaceletsBlankWithoutLibs
	FaceletsKickStartWithRILibs
	FaceletsKickStartWithoutLibs
?	< Back Next > Cancel Finish

Figure 2.3. Choosing Facelets Template

The following table lists the templates that can be used with Facelets for any JSF project, and gives a detailed description for each one.

Table 2.1. Facelets Templates

Template	Description
FaceletsBlankWithoutLibs	Some servers already provide JSF libs and you risk library conflicts while deploying your project. To avoid such conflicts, use a template without libs if you have a server with its own JSF libraries.
FaceletsKickStartWithRILibs	A sample application with Facelets that is ready to run.
FaceletsKickStartWithoutLibs	A sample application without libraries.

2.1.2. Facelets components

The JBoss Tools Palette comes with the Facelets components ready to use. A useful tip appears when you hover the mouse cursor over the tag; this tip includes a detailed description of the tag component, the syntax and available attributes.

		😳 Palette 🞾 JBoss Tools Pa 🕱 🛛 🗖 🗖
<u> </u>		🛠 🗟 📀
		궏 JBoss Ajax4Jsf
		😕 JBoss RichFaces
		😕 JBoss Seam
		🗁 JSF Facelets ∞
=		component
		composition
The component tag and the composition tag behave exactly the same, except the component tag will insert a new UIComponent instance into the tree as the root of all the child components/fragments it has. Syntax: <ui:component> </ui:component> Attributes: id, binding		

Figure 2.4. Facelets Components

2.1.3. Code assist for Facelets

JSF Tools provides Facelets code assistance, which can be accessed by pressing **Ctrl+Space**. It is available for Facelets tags while editing .xhtml files.

e	👌 *inde	ex.jsp	💼 *inp	utname.xhtml 😫	
	1⊜< 2⊝ 3 4 5⊝	html> <head </head <body< th=""><th>d> <title> ad> /></title></th><th>Test</th><th></th></body<>	d> <title> ad> /></title>	Test	
	6		input		
	7	<th>iy></th> <th><> Close with " />"</th> <th>Attribute : accept</th>	iy>	<> Close with " />"	Attribute : accept
	8 <	/html>		(a) accept	Data Type : CDATA
				accesskey	
				(a) align="top"	
				(a) alt	
				③ checked="checked"	
				(a) class	
		((a) dir="ltr"	
				(a) disabled="disabled"	
			 ♀	(8) id	
				(a) lang	
				Press 'Ctrl+Space' to show HTML Template Proposals	
					<i>a</i>

Figure 2.5. XHTML File Code Assist

Code assist is also available for jsfc attributes in any HTML tag.

C	1	*inputname.xhtml 🛙		-	٥
ſ	I	.ns:f="http://java. .ns:c="http://java.	sun.com/jst/core* sun.com/jstl/core*	-	-
		f:loadBundle basen	ame="resources" var="msg" />		
l		ui:composition tem	<pre>plate="/templates/common.xhtml"></pre>		
l		<ui:define name<="" th=""><th>="pageTitle">JSF 1.2 and Facelets under Tomcat. KickStart Application</th><th><</th><th></th></ui:define>	="pageTitle">JSF 1.2 and Facelets under Tomcat. KickStart Application	<	
		<ui:define name<="" th=""><th>="pageHeader">JSF 1.2 Hello Application</th></ui:define>	="pageHeader">JSF 1.2 Hello Application		
4	B	<ui:define name<br=""><h:message <form jsfc="<br">\${msg.p</form></h:message </ui:define>	<pre>="body"> showSummary="true" showDetail="false" style="color: red; font-weight: "h:form" id="helloForm"> rompt}</pre>	11	
Monda		<input< th=""><th>/></th><th></th><th></th></input<>	/>		
e	1) 1) 1)	<input act </input 	dir disabled disabled		
l			© jsfc		
l		:/ui:composition>	(8) lang		
l			(a) maxlength	-	
l		•	() name	ີ	1
l			(a) onblur	3	ĸ
,	Vis	sual/Source Source Pr	(a) onchange		
			(8) onclick		
			ondblclick		

Figure 2.6. Code Assist for JSFC Attribute

After selecting an jsfc attribute, the code assist feature will list the JSF components available on a page.



Figure 2.7. Code Assist for JSF Components

When a component is selected you will see all available attributes for it.



Figure 2.8. Available Attributes for the Component

2.1.4. Open On feature

Finally, JSF Tools supports Eclipse's OpenOnTM feature while editing Facelets files. Using this feature, you can easily navigate between the Facelets templates and other parts of your projects.

By holding down the **Ctrl** key while hovering the mouse cursor over a reference to a template, the reference becomes a hyperlink to navigate to that template.



Figure 2.9. Template Hyperlink

Projects

To take an advantage of JSF you will need to perform one of the next steps:

- Create new JSF projects
- Import (open) existing JSF projects
- Add JSF capability to any existing Eclipse project
- Import and add JSF capability to any existing project created outside Eclipse.

This section will go into more detail for each step.

3.1. Creating a New JSF Project

It is easy to create a new project that contains all the JSF libraries, tag libraries and JSF configuration file with the aid of a special wizard. To get it, select File \rightarrow New \rightarrow Other \rightarrow JBoos Tools Web \rightarrow JSF \rightarrow JSF Project and click the Next button.

😔 New	\mathbf{X}
Select a wizard	
Create a JSF Project	
Wizards:	
type filter text	
web bescriptor	
🚵 XHTML Page	
▽ 🗁 JSF	
🙍 Faces Config	
🛣 JSF Project	
👂 🗁 Portlet	=
👂 🗁 Struts	
👂 🗁 JPA	
👂 🗁 Plug-in Development	
? < Back N	ext > Cancel Finish

Figure 3.1. Choosing a JSF Project

On the next page you will be prompted to enter the **Project Name** and select a location for the project (or just leave a default path).

The **JSF Version** option also allows you to specify the JSF implementation to use.

@	New JSF Project 🛛 🕅
Create JSF Proje	ct 🏼 💮 🄶
Project Name*	JSFProject
	✓ Use default path*
Location*	/home/irooskov/Work/JBDS/workspace/JSFProject Browse
JSF Environment*	JSF 2.0 ♀
Template*	JSFKickStartWithoutLibs
?	< Back Next > Cancel Finish

Figure 3.2. Creating a New JSF Project

There are a number of predefined project templates that are both flexible and easily customizable. You can pick a different template on which the projects Importing Existing should be based on. Almost all templates come in two variations: with and without JSF libraries.

e	New JSF Project 🛛 🗙
Create JSF Proje	ct 🌍
Project Name*	JSFProject
	✓ Use default path*
Location*	me/irooskov/Work/JBDS_5.0/workspace/JSFProject Browse
JSF Environment*	JSF 2.0
Template*	JSFBlankWithLibs
	JSFBlankWithoutLibs
	JSFKickStartWithoutLibs
?	< Back Next > Cancel Finish

Figure 3.3. Choosing JSF Templates

The table below provides description for each possible JSF template.

Table 3.1. JSF Project Templates

Template	Description
JSFBlankWithoutLibs	This template will create a standard Web project structure with all the JSF capabilities.
	Use a template without libs to avoid library conflicts when your server already has JSF libraries installed.
JSFKickStartWithoutLi	bathis template will create a standard Web project structure, and also include a sample application that is ready to run.
	Use a template without libs to avoid library conflicts when your server already has JSF libraries installed.

On the next page you need to select which **Servlet version** to use, and specify whether or not to register this application with JBoss AS (or other server) in order to run and test your application.

The Context Path option defines the name under which the application will be deployed.

The **Runtime** value tells Eclipse where to find the Web libraries necessary to build (compile) the project. It is not possible to finish the project creation without selecting a Runtime. If you do not have any values, click the **New...** button to add new Runtime.

The **Target Server** option allows you specifying whether or not to deploy the application. The Target Server corresponds to the Runtime value selected above. If you do not want to deploy the application, uncheck this option.

e	New JSF Project	X
Web		
Servlet Version:	2.5	~
Context Path:*	JSFProject	
Runtime:*	jboss-eap Runtime	New
Target Server:	✓ jboss-eap	New Select All Deselect All
?	< Back Next > Cancel	Finish

Figure 3.4. Registering the Project on Server

When you are all done, you should see that the project has appeared in the Package Explorer view:



Figure 3.5. A New Project in the Package Explorer

At this point you can open the faces-config.xml file and start working on your application. There are a lot of features available when developing JSF applications. These features will be discussed in more detail later in this document.

3.2. Importing Existing JSF Projects with Any Structure

For detailed information on migration of JSF projects into a workspace see the Migration Guide.

3.3. Adding JSF Capability to Any Existing Project

It is also possible to add JSFTM capabilities (JSF libraries, tag libraries) to any existing project in your workspace. After that you will be able to make use of features such as the JSF configuration editor, JBoss Tools JSP editor and any others. No pre-existing web.xml file is necessary.

Right-click on the project in the **Project Explorer**, bringing up the context menu. From this menu navigate to **Configure** \rightarrow **Add JSF Capabilities**.

TestNG >	Add JBoss Tools Knowledge Base Support	
Source >	Add JSF Capabilities	
Configure >	Add CDI (Context and Dependency Injection) sup	

Figure 3.6. Add JSF Capabilities menu item

This will open the **Project Facets** dialog for the project. Click the checkbox next to **JavaServer Faces**. You undertake further configuration by clicking the **Further configuration available** button at the bottom of the dialog; this will allow you to define specific configuration options. Click **Apply** and then the **OK** on the **Project Facets** dialog when you are finished.

type filter text 🔏	Project Facets			
Project Facets	Configuration: <pre><custom></custom></pre>			
	Project Facet	Version		
	 Axis2 Web Services CDI (Contexts and Dependency Injection) 	1.0		
	CXF 2.x Web Services 1.0			
	Dynamic Web Module 3.0			
	I Java	1.7		
	✓ Image: A state of the s			
	 □ □ JAX-RS (REST Web Services) ▷ □ □ JBoss Portlets □ ↓ JBoss Web Services Core 	1.1 3.0		
	□ ↔ JPA	2.0		
	🗌 📄 Seam 2	2.2		
	WebDoclet (XDoclet)	1.2.3		
	i <u>Further configuration available</u>			
< III >				
?				

Figure 3.7. Project Facets dialog

The project will now contain a new node (visible through the **Project Explorer**) named **Web Resources**.You will also notice new files within the **WebContent** folder.



Note:

Some application servers provide their own JSF implementation libraries. To avoid conflicts you should not add JSF libraries while adding JSF capabilities.

You can now open the new faces-config.xml file, that is found under your projects **WebContent** \rightarrow **WEB-INF** folder. This file can be opened in a unique editor (see *Chapter 5, JSF Configuration File Editor*).

3.4. Adding Your Own Project Templates

A template is a set of files that is provided as a basis when creating a new project. Project templates provide content and structure for a project.

JSF Tools provides powerful template capabilities which allow you to create new templates and import existing Struts and JSF projects. This templating facility has a variety of aspects to consider. Let's start with the most straightforward case and consider the process of creating a template from your existing JSF project.

Let's say you have a project that you want to use as the basis for a new template. The following steps will show you how to achieve this:

• In the Web Projects view, right-click the project and select JBoss Tools JS \rightarrow Save As Template

	ቹ Package Expl 🔝 Web Proje	ects 🛿 🗖 🗖	
	<u>n</u>	遂 🛐 🖻 🔄	
	JSFProject	JBoss Tools JSF 💙	Save As Template
▷ 🟦 SomeWebProject		Delete	Remove JSF Capa
		Properties	Verify
			Add Custom Capa
			Add ORM Capabil
ļ			

Figure 3.8. Saving Your Project as Template

• In the first dialog box, you can specify a name for the template (it will default to the project name) and confirm what run-time implementation of the project technology will be used.

9	Add JSF Project Template 🛛 🗙		
Define Common Template Properties			
Name:*	MyJSFProject		
Implementation:*	JSF 1.1.02 - Reference Implementation		
	Next >> Finish Cancel		

Figure 3.9. Define Template Properties

• When you click the **Next** button a dialog box will be presented with your project structure displayed, along with a number of check boxes. Here you can select only those parts and files in your project directory that should be part of the template.

😔 🛛 🕹 Add JSF Proje	ct Template	×
Select Folders and Files		
 ✓ ■ SFProject ∴ settings ✓ JavaSource ✓ WebContent ✓ META-INF ✓ WEB-INF Classes ⇒ lib Method >> lib 		
<< Back Next >>	Finish	Cancel

Figure 3.10. Define Template Properties

• At this point, unless you want to designate some extra files as having Velocity template coding inside them, you should click the **Finish** button.

That's it. This template can be used with any new or imported project that uses the same run-time implementation as the project you turned into a template.

At this point you have a fully configured project. Now you can add some additional logic to it starting with the JSF configuration file.

3.5. Relevant Resources Links

You can find a more in-depth explanation on how to work with the special wizards, editors and views that can be used while developing JSF applications in our Visual Web Tools Guide.

Web.xml Editor

The web.xml file inside the WEB-INF folder is a deployment descriptor file for a Web Application. It describes the servlets and other components and deployment properties that make up your application.

JBoss Tools add the web.xml file to created JSF project automatically and provides a special editor for its editing. See the Visual Web Tools guide for more information on the web.xml editor.

JSF Configuration File Editor

First, we should mention that JSF configuration file (faces-config.xml) is intended for registering JSF application resources such as Converters, Validators, Managed Beans and page-to-page navigation rules.

Now, let's look at how you can easily configure this file by means of a special graphical editor for the JSF configuration file. The editor has three main views:

- Diagram
- Tree
- Source

They can be selected via the tabs at the bottom of the editor.

5.1. Diagram view

Here, we will show you how to work with JSF configuration file through the Diagram view of the editor.

As you can see on the figure below, the Diagram view displays the navigation rules container in the faces-config.xml file:



Figure 5.1. Diagram View

If you have a large diagram, make use of the Outline view. Within it you can switch to a **Diagram Navigator** mode by selecting the middle icon at the top of the view window. This allows you to

easily move around the diagram. Just move the blue area in any direction, and the diagram on the left will also move:

ा *faces-config.xml छ	- 8	🗄 Outline 🛿 🔅 Palette 🚿 JBoss Tools Palette 🗖 🗖
		H 🛃 🗢
/testpage1.jsp testpage2		/henty ag eLjop
/testpage2.jsp	nput	restoranting of Jap Control and Angel Final Angel Fina
Diagram Tree Source		

Figure 5.2. Outline View for Diagram

To create a new page here, you should click the page icon (View Template) on the toolbar from the left and then click anywhere on the diagram. A New Page Wizard will appear.

To create a transition for connecting pages:

- Select the transition icon from the toolbar (New Connection).
- Click the source page.
- Click the target page.

A transition will appear between the two pages:



Figure 5.3. Transition between JSP Pages

It is also possible to create a new page with context menu by right-clicking anywhere on the diagram and selecting the **New View...** option.



Figure 5.4. Creating a New View

To edit an existing transition, first select the transition line. Then, place the mouse cursor over the last black dot (on the target page). The mouse cursor will change to a big +. At this point, drag the line to a new target page:



Figure 5.5. Editing Transition between Views

5.2. Tree View

You can find it more convenient to edit your JSF Configuration file in the Tree view of the VPE.

The view displays all JSF application artifacts referenced in the configuration file in a tree format. By selecting any node on the left, you can view and edit its properties which will appear in the right-hand area. Let's look at the structure of this tree more closely.

• Under the **Application** node you can adjust JSF application specific settings such as internationalization, extensions, adding property and variable resolvers, etc.

া≱ *faces-config.xml প্ল		
Faces Config Editor		
🝷 faces-config	Application	
マ 🗟 faces-config.xml∗	EL Resolvers	
Application	Property Resolvers	
Components	Variable Resolvers	
	Message Bundles	
Managed Beans	Resource Bundles	
Referenced Beans	- Locale Config	
Render Kits	Default Locale:	
✓ Validators	Supported Locale	
Section Extensions	Locale	
	en_US	
	Extensions	
	Advanced	
Diagram Tree Source		
Diagram nee source		

Figure 5.6. JSF Application Specific Settings

The Components node is for registering custom JSF components. Right-click and select New
 → Component or just click the Add button in the right-hand area to add a new component to
 the JSF Configuration file.
races-config		Components	
✓ 10 faces-config.xml* In the second sec		type class	
Components	New	>	Component
Converters	Cut	Ctrl + X	
Managed Beans	Сору	Ctrl + C	
Referenced Beans	Paste	Ctrl + V	
Render Kits	Delete	Delete	
✓ Validators	Droporti	105	
Sections -	Properti	es	
	Verify		

Figure 5.7. Registering a New JSF Component

In the **Add Component** wizard you should set a component type and point to a component class by using the **Browse** button or create a new class for this component by using the **Component-Class** link.

9	Add Componen	t	×
Component Attribute Componen	t Type must be set.		
Component Type:*			
Component Class:*			Browse
?		Cancel	Finish

Figure 5.8. Adding a New JSF Component to the JSF Configuration File

• Use the **Render Kit** node to create and register a set of related renderers for custom JSF components.

 ✓ faces-config ∞ faces-config.xml* Application Components Converters Managed Beans Navigation Rules Referenced Beans 		• Render Kits id	class
Render Kits	New	>	Render Kit
 Wandators Extensions 	Cut Copy Paste	Ctrl + X Ctrl + C Ctrl + V	
_	Delete	Delete	
	rioperu		

Figure 5.9. Adding a New JSF Renderer Kit to the JSF Configuration File

• Under the **Converters** node you can create a converter class for your JSF application either with an id or for a proper class. For more information on this procedure see Section 7.1, "Create and Register a Custom Converter".

1 *faces-config.xml ន			
Faces Config Editor			
✓ faces-config	- Conv	erters	
▽ 🏂 faces-config.xml*	id		class
Application			
Components			
Managed Beans	New	\rightarrow	with id
Managed Deans Navigation Rules	Cut	Ctrl + X	for class
log Referenced Beans	Сору	Ctrl + C	
🗟 Render Kits	Paste	Ctrl + V	
😿 Validators	Delete	Delete	
Section Extensions	Properties		
	Verify		
Diagram Tree Course			
Diagram Tree Source			

Figure 5.10. Creating a New Custom Converter

• The **Managed Bean** node allows you to create and register Bean classes in your JSF application. Read more on the topic in *Chapter 6, Managed Beans*.

+ faces-config	✓ Mai	naged Beans	5
✓ ▲ faces-config.xml* ♦ Application ♠ Components ♠ Converters	nam	e	
🗟 Managed Beans	New	\rightarrow	Managed Be
Navigation Rules	Cut	Ctrl + X	
Service Referenced Beans	Сору	Ctrl + C	
Validators	Paste	Ctrl + V	
 Extensions 	Delete	Delete	
	Properties		
	Verify		

Figure 5.11. Managed Beans

• Use the **Navigation Rules** node to configure a navigation between the pages in your application. Here you can create a new navigation rule and adjust necessary properties for it in the right-hand area.



Tip:

The same you can do in the Diagram view of the JSF Configuration file editor (see *Section 5.1, "Diagram view"*).



Figure 5.12. Configuring Navigation Rules

• Under the **Referenced Beans** node you can add a new Referenced Bean and configure various properties for it. To learn more on this refer to *Section 7.3, "Create and Register Referenced Beans"*.

New	Referenced Be	ans class
New	Referenced Be	class
New	Referenced Be	class
New	name	class
New		
	<u>></u>	Referenced
Cut	Ctrl + X	
Сору	Ctrl + C	
Paste	Ctrl + V	
Delete	Delete	
Proper	ties	
Verify		
	Cut Copy Paste Delete Proper Verify	Cut Ctrl + X Copy Ctrl + C Paste Ctrl + V Delete Delete Properties Verify

Figure 5.13. Referenced Beans

• The **Validators** node is needed to create validator classes for organizing the validation of your application data. You can read more on the topic in *Section 7.2, "Create and Register a Custom Validator"*.

🔬 faces-config.xml 🛛			
Faces Config Editor			
🝷 faces-config	👻 Val	idators	
 ✓ Application ♦ Application ♦ Components ♦ Converters ♦ Managed Beans ♦ Navigation Rules ♦ Referenced Beans 	id		class
Render Kits			
	New	>	Validator
Se Extensions	Cut Copy Paste	Ctrl + X Ctrl + C Ctrl + V	
	Delete	Delete	
	Properties		
Diagram Tree Source			

Figure 5.14. Validators

• The **Extensions** node is for setting extensions in your <code>faces-config.xml</code> file.

🔊 faces-config.xml 🛙			
Faces Config Editor			
🝷 faces-config	- E	xtensions	
 ✓ Marces-config.xml ♦ Application ♦ Components ♦ Converters ♦ Managed Beans ♦ Navigation Rules ♦ Referenced Beans ♦ Referenced Beans ♦ Render Kits ♦ Validators 	ele	ement	
Section Extensions	New	\rightarrow	Extension
-	Cut	Ctrl + X	
	Сору	Ctrl + C	
	Paste	Ctrl + V	
-	Delete	Delete	
	Properties		

Figure 5.15. Adding Extensions

In the **Tree view** you can also edit the properties of the selected element with the help of the **Properties view** as shown below:

🔊 *faces-config.xml 없				- 8	□ Properties 🛛	یہ 😫 🖪	~ - 0
Faces Config Editor					Property	Value	
	 Managed Bean 			1	comment		
V Staces-config ymlt					description		
	Managed-Bean-Name:	person			display-name		
Application	Managed-Bean-Class:	demo.Person	Browse		id		
Components	Managed Reap Scope	request		=	large-icon		
Converters	Manageu-bean-scope.	Tequest			managed-bean-class	demo.Person	
V 🖓 Managed Bean:	Description:		4		managed-bean-name	person	
🗢 🥏 person 🗧		CI.	×		managed-bean-scope	e request	
 name 		4		-	small.icon	e request	
👂 🍓 Navigation Rule	 Properties 				sman-icon		
🍓 Referenced Be	name	class value	<u>A</u> dd				
🔄 Render Kits	name		Bamaua				
🐼 Validators 📄			<u>Remove</u>				
			<u>E</u> dit	•			
Diagram Tree Source					III		

Figure 5.16. Properties View

5.3. Source View

Here, we'll discuss how you can configure your ${\tt faces-config.xml}$ file with the help of the Source View.

The **Source** View for the editor displays the text content of the JSF configuration file. It is always synchronized with other two views, so any changes made in one of the views will immediately appear in the other:



Figure 5.17. Source View

You can also work in the **Source** View with the help of the **Outline** View. The **Outline view** shows a tree structure of the JSF configuration file. Simply select any element in the **Outline** View, and it will jump to the same place in the Source editor, so you can navigate through the source code with **Outline** View.



Figure 5.18. Outline View

5.4. Editor Features

Here we'll discuss a very important features that JSF configuration file editor provides when working with JSF resources.

5.4.1. Open On

The JSF configuration file editor comes with the very useful OpenOn navigation feature. You can find more information on this feature in the Visual Web Tools Reference Guide.

5.4.2. Code Assist

Code Assist provides a pop-up tip to help you complete your code statements. It allows you to write your code faster and with more accuracy.

Code assist is always available in the Source mode:

🛃 *fa	ces	-config.xml 🕱	
$ \begin{array}{c} 1 \\ 2^{\odot} \\ 3 \\ 4 \\ 5^{\odot} \\ 6 \\ 7 \\ 8 \\ 9^{\odot} \\ 10 \\ 11 \\ 12 \\ 13 \\ 14^{\odot} \\ 15 \\ 16 \\ 17^{\odot} \\ 18 \\ \end{array} $	xi<br <fa xm <m <i <i <i <i <i </i <n <i <i <i <i </i </i </i </i </n </i </i </i </i </m </fa 	<pre>ml version="1.0" encoding="UTF-8"?> ccs-config version="1.2" xmlns="http://java.sun.com/xml/ns/javaee" lns:xi="http://www.w3.org/2001/XInclude" lns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocat anaged-bean> nanaged-bean-name>person nanaged-bean-class>demo.Person nanaged-bean-scope>request managed-bean-scope>request managed-property> <property-name>name</property-name> <value></value> //managed-property> managed-bean> avigation-rule> description>This is a test from-view-id>/testpagel.jsp havigation-case> </pre>	ion="http://java.sun.co
18		<pre><trom-outcome>destpage2</trom-outcome></pre>	Element : from-outcome
19 20 21 22⊕ 23 24⊕ 25 26 27 28 29⊕ 30 31⊕ 32 33 31⊕	<	<> description <> display-name <> from-action <> icon <> from-outcome <> redirect <> to-view-id # XSL processing instruction - XSL processing instruction # comment - xml comment , person : Person Press 'Ctrl+Space' to show JBoss CDI (Context and Dependency Injection) Class Proposal	The "from-outcome" eleme execution of an application (or a literal value specified component. If specified, thi matches this element's val matter what the outcome v Data Type : token
35 36⊕ 37⊕ 38 39 40 41 42		<pre>press carrier back to show jobs context and Dependency mjection/ class repose oplication> locale-config> <supported-locale>en_US</supported-locale> /locale-config> application> aces-config></pre>	
Disa	<	The Course	III
Diagr	am	Tree Source	

Figure 5.19. Code Assist in Source View

5.4.3. Error Reporting

Constant error checking is provided while you are developing your project. This greatly reduces your development time as it allows you to catch many errors during the development process.

Errors will be reported by Chapter 8, JSF Project Verification facility:



Figure 5.20. Error Reporting in Source View

Other errors are also reported.

🔊 faces-config.xml 😫					
<pre></pre>					
15 <description>This is a test</description>					
<pre>16 <from-view-id>/testpage1.jsp</from-view-id></pre>					
170 <navigation-case></navigation-case>					
Diagram Iree Source					
🖹 Problems 🕱 🏼 🖉 Tasks 🤻 Servers 📮 Console	~				
2 errors, 2 warnings, 0 others					
Description Resource P	Path				
▽ 😣 Errors (2 items)					
😣 error: Attribute managed-bean-class references to non-existent class faces-config.xm /JSFP					
Output to the second					
Warnings (2 items)					

Figure 5.21. Other Errors Reporting

Managed Beans

JSF Tools provides a number of useful features when working with managed beans, such as:

- Adding and generating code for new managed beans
 - Generating code for attributes and getter/setter methods
- Adding existing managed beans to a JSF configuration file

This guide will look at each of these features in more detail.

6.1. Code Generation for Managed Beans

To begin, create a new managed bean in JSF configuration file editor using the Tree view.

▲ faces-config Editor ▼ faces-config ▼ Managed Beans ▲ Application ▲ Application ▲ Components ▲ Converters ▼ Managed Beans ▶ < person ▶ < managed Beans ▶ < managed Be				
Faces Config Editor faces-config.xml Application Components Converters Managed Beans Managed Beans Managed Beans Managed Beans Managed Beans Managed Beans Managed Beans Cut Ctrl + X Copy Ctrl + C Paste Ctrl + V Delete Delete Properties Verify 	🔊 faces-config.xml 🕴			
 ✓ faces-config.xml Application Components Converters ✓ Managed Beans ▷ <pre>Person</pre> ▷ <pre>Managed Beans</pre> 	Faces Config Editor			
 Application Application Components Converters Managed Beans Managed Managed Managed Cut Ctrl + X Copy Ctrl + C Render Kits New Validators Validators Verify 			- Managed Bea	ns
Image: Managed Beans	✓ Application Applic		name person	
> Inclusion > Inclusion	V G Managed Beans	New	>	Managed Be
	 Referenced Beans Render Kits Validators Extensions 	Cut Copy Paste Delete Proper Verify	Ctrl + X Ctrl + C Ctrl + V Delete	
Diagram Tree Source	Diagram Tree Source			

Figure 6.1. Creation of New Managed Bean



Note:

When you define a new managed bean, make sure that **Generate Source Code** option is checked as shown in the figure below.

	New Managed Bean	×
Manage	d Bean	
Scope:	request	~
Class:*	example.carBean	Browse
Name:*	carBean	
	☑ Generate Source Code	
	Next >> Finish	Cancel

Figure 6.2. New Managed Bean

After the Java class has been generated you can open it for additional editing. There are two ways to open a Java class:

- Click on the Managed-Bean-Class link in the editor.
- Right click the *managed bean* and select the **Open Declaration** option.

📧 *faces-config.xml 🛛 🚺 carBean	.java	
Faces Config Editor		
	- Managed Bean	
 ✓ Managed Beans 	Managed Bean Name: <u>Managed Bean Class:</u> Mana Open n Scope:	carBear exampl request
person	Description:	
 Navigation Rules Referenced Beans Render Kits Validators Extensions 	 → Properties name ✓ ✓ ✓ Advanced 	
	ID:	
	Display Name:	
	Small Icon:	
Diagram Tree Source		

Figure 6.3. Opening of Created Managed Bean

The generated Java source should look as follows:



Figure 6.4. Java Source Code

You can also generate source code for properties, also includes getter and setter methods. Right click on the bean and select $New \rightarrow Property$. You will then see the Add Property dialog.

📧 *faces-config.xml 🛛 🚺	carBean.	java		
Faces Config Editor				
🝷 faces-config		- Managed B	ean	
✓ Maces-config.xml*		Managed Bean <u>Managed Bean</u> Managed Bean	Managed Bean Name: <u>Managed Bean Class:</u> Managed Bean Scope:	
		Description:		
🥔 carBean	New		> Pr	operty
Referenced Bear	Open Declaration		Li	st-Entries
Render Kits	Rename Class		М	ap-Entrie
🐱 Validators	Cut	Ctrl +	х	
Section Extensions	Сору	Ctrl +	с	
	Paste	Ctrl +	V	
	Delete	Dele	te	
	Properti	es		
	Verify			
		ID:		
		Display Name:		
		Small Icony		
Diagram Tree Source				

Figure 6.5. Generation of Source Code for Properties

When the form is open make sure that all the check boxes are selected:

- Add Java property
- Generate Getter
- Generate Setter

8	Add Property 🛛 🗙
Property	
Property Name:*	carName 🗸
Property Class:	java.lang.String Browse
Value Kind:	value
Value:	
	K III >
	Add Java property
	Generate Getter
	✓ Generate Setter
	Finish Cancel

Figure 6.6. "Add Property" Form

Once the generation is complete, you can open the file and see the newly added property with accompanying "get" and "set" methods:

```
🚺 carBean.java 🖾
1 *faces-config.xml
  10/**
  2
      */
  3
  4
    package example;
  5
  69/**
  7
      * @author matthew
  8
  9
      */
 10 public class carBean {
         private java.lang.String carName;
 11
 12
 13<del>0</del>
         public carBean() {
 14
         }
 15
 160
         public java.lang.String getCarName() {
 17
              return carName;
 18
         }
 19
 200
         public void setCarName(java.lang.String carName) {
 21
             this.carName = carName;
         }
 22
 23
    }
 24
                                                            ш
     1
```

Figure 6.7. Generated Java Source Code for Property

This covers the options available when creating a new Managed Bean. The next section will show you how to add an existing Bean into a JSF configuration file.

6.2. Add Existing Java Beans to a JSF Configuration File

If you already have a Java bean you can easily add it to a JSF configuration file.

You should start the same way you create a new managed bean. Use the **Browse...** button to add your existing Java class.

@	New Managed Bean	×
Manage	d Bean	
Scope:	request	~
Class:*	example.carBean	Browse
Name:*	car	
	☑ Generate Source Code	
	Next >> Finish	Cancel

Figure 6.8. New Managed Bean Form

Once the class is set, its **Name** will be set as well. But you can easily substitute it for the other one. Notice that **Generate Source Code** option is not available as the Java class already exists.

After adding your class the **Next** button will be activated. When you click it you will be presented with the **Managed Properties** dialog where all corresponding properties are displayed. Checking the appropriate ones will add them into your JSF Configuration File.

9	New Managed Bean
Managed Propertie Select properties you	want to add to the managed-bean
name	value
carName	
	<< Back Finish Cancel

Figure 6.9. Selection of Bean's Properties.

If you don't want to add any, just click the $\ensuremath{\textit{Finish}}$ button.

The steps above have demonstrated how you can add an existing Bean to the JSF configuration file, i.e. <code>faces-config.xml</code>. The next chapter will demonstrate how to organize and register other kinds of artifacts.

Creation and Registration

7.1. Create and Register a Custom Converter

It's also possible to create a custom Converter in order to specify your own converting rules. Let's look at how you can do this.

To create and register a custom converter it is necessary perform the following steps:

• In the Project Explorer view open the faces-config.xml file and select Tree tab.

🙍 faces-config.xml 🛛 🕖 carBean.java	
Faces Config Editor	
✓ faces-config ✓ Converters	
 faces-config.xml Application Components Converters Managed Beans Mavigation Rules Referenced Beans Render Kits Validators Extensions 	

Figure 7.1. Converters

- Select **Converters** and click the **Add** button.
- On the form type the name of your converter in the *Converter-id* field and name of the class for converters. After clicking **Finish** button your custom converter is registered under the entered name.


Figure 7.2. Add Converter Form

• Now you can create a *"converter"* class. In the Converter section you should see your **Converter-id** and **Converter-class**. Click on the **Converter-Class** link to generate the source code.



Figure 7.3. Generation of Source Code for Converter Class

• A usual wizard for creating a Java class will appear. All needed fields here will be adjusted automatically. Just leave everything without changes and click the **Finish** button.

e	New Java Class
Java Class	
Create a new Java	class.
Source folder:	JSFProject/JavaSource
Package:	test
Enclosing type:	
Name:	Customconverter
Modifiers:	public Odefault O private O protected
Modifiers.	□ abstract □ final □ static
Currentees	
Superclass:	Java.lang.Object
Interfaces:	javax.faces.convert.Converter
Which method stub	s would you like to create?
	public static void main(String[] args)
	Constructors from superclass
	Inherited abstract methods
Do you want to add	l comments? (Configure templates and default value <u>here</u>
	Generate comments
Figure 7.4. New Java Cla	ss Form
(?)	Cancel

• To open a converter class click again on the **Converter-Class** link in the Converter section.

```
📧 *faces-config.xml
                       carBean.java
                                           🚺 Customconverter.java 🔀
     package test;
   1
   2
   3@ import javax.faces.component.UIComponent;
   6
      public class Customconverter implements Converter {
   7
   8
          public Customconverter() {
   9<del>0</del>
               // TODO Auto-generated constructor stub
10
  11
          }
  12
  13<del>0</del>
          @Override
          public Object getAsObject(FacesContext arg0, UICompone
<u>∽</u>14
               // TODO Auto-generated method stub
15
               return null;
  16
  17
          }
  18
  190
          @Override
          public String getAsString(FacesContext arg0, UICompone
<u>∽</u>20
               // TODO Auto-generated method stub
21
               return null;
  22
  23
          }
  24
  25 }
  26
Figure 7.5. Converter Class
```

1

Now you are able to add a business logic of converter in the Java editor.

7.2. Create and Register a Custom Validator

It is also quite easy to develop your own custom Validators. The required steps are similar to those shown previously:

• In the Project Explorer view open the faces-config.xml and select the Tree tab.

🔊 faces-config.xml 😫		
Faces Config Editor		
✓ faces-config	✓ Validators	
 faces-config Application Components Converters Some Managed Beans Some Managed Beans Some Referenced Beans Render Kits Validators Extensions 	id class	
Diagram Tree Source		

- Select the Validators option and click the Addbutton.
- Type the name of your validator in the **Validator-id** field and name of the class for validators. After clicking the **Finish** button your custom validator is registered under the entered name.

۲	Add Validator
Validator	
Validator ID:*	MyValidator
Validator Class:*	test.CustomValidator
0	Cancel
	Cancer

Figure 7.7. Adding Validator

Now you can create the "validator" class.

• In the Validator section you can see your Validator-id and Validator-class. To generate the source code click on Validator-class.

🔊 *faces-config.xml 🕱		
Faces Config Editor		
👻 faces-config	- Validator	
∽ 🔊 faces-config.xml*	Validator ID:	MyValidator
 Application Components 	Validator Class:	test.CustomV
Converters	Description:	
Managed Beans Solution Rules		<
Referenced Beans	 Attributes 	
Render Kits	name	class
MyValidator		
Sections Sections		
	 Properties 	
	name	class
Figure 7.8. Creating Validator Class	- Extensions	
	element	
Diagram Tree Source		

• Java class will be created automatically. Leave everything without changes and click the **Finish**.

e	New Java Class
Java Class	
Create a new Java	class.
Source folder:	JSFProject/JavaSource
Package:	test
Enclosing type:	
Name:	CustomValidator
Modifiers	public O default O private O protected
induiter 5.	\square abstract \square final \square static
Superclass	iava lang Object
superclass:	Java.iang.Object
Interfaces:	javax.faces.validator.Validator
Which method stub	s would you like to create?
	public static void main(String[] args)
	Constructors from superclass
	Inherited abstract methods
Do you want to add	l comments? (Configure templates and default value <u>her</u>
	Generate comments
Figure 7.9. New Java Cla	ss Form
?	Cancel

• To open the validator class click on the **Validator-Class** link in the Validator section. Now you are able to write a business logic of validator in the Java editor.



7.3. Create and Register Referenced Beans

The creation of Referenced Beans is similar to the creation of Custom Validators. The steps below show you the steps required to create Referenced Beans.

• In the Project Explorer view open the faces-config.xml and select the Tree tab.

staces-config.xml छ		
Faces Config Editor		
🝷 faces-config	- Referenced Beans	
 ✓ Application ♦ Application ♦ Components ♦ ♦ Converters ♦ ♦ Managed Beans ♦ ♦ Navigation Rules ♦ Referenced Beans ♦ Referenced Beans 	name	class
 Validators ✓ Extensions 		
Diagram Tree Source		

- Select the **Referenced Beans** option and click on the **Add** button.
- Type in the name of your Referenced Bean and type in or select the **Referenced-Bean-Class** value by clicking the **Browse** button.

(Add Referenced Bean
Referenced Bean	
Referenced Rean Names*	Murafaran cad Daan
Referenced Bean Name:*	MyreferencedBean
Referenced Bean Class:*	test.ReferencedBean
?	Cancel

Figure 7.12. Add Referenced Bean

• In the Referenced Bean section you should see your **Referenced-Bean-Name** and **Referenced-Bean-Class**. Click on the link to open the Java creation wizard.

sfaces-config.xml 😫		
Faces Config Editor		
👻 faces-config	- Referenced Bean	
▽ 🙇 faces-config.xml*	Referenced Bean Name:	Myre
Application	Referenced Bean Class	test f
Components	Description:	costi
Converters	Description:	
Managed Beans A law inaction Bules		$\overline{\langle}$
Interrigence d Beans	- Advanced	
Ø MyreferencedBean	ID:	
Render Kits	Display Namo	
Validators		
Extensions	Small Icon:	
	Large Icon:	
Figure 7.12 Create Deferenced Peen Class		
rigure 7.13. Greate Referenced Dean Class		
Diagram Tree Source		

• The Java class will be created automatically. Leave everything with their default values and click the **Finish** button.

8	New Java Class
Java Class	
Create a new Java	class.
Source folder:	JSFProject/JavaSource
Package:	test
Enclosing type:	
Name:	ReferencedBean
Modifiers:	public Odefault Oprivate Oprotected
Mounters.	□ abstract □ final □ static
Currentere	
Superclass:	Java.lang.Object
Interfaces:	
Which method stub	s would you like to create?
	public static void main(String[] args)
	 Constructors from superclass
	Inherited abstract methods
Do you want to add	l comments? (Configure templates and default value <u>her</u>
	Generate comments
Figure 7.14. New Java C	ass Form
?	Cancel

• To open a Referenced Bean class click the **Referenced-Bean-Class** in the Referenced Bean section. Now you are able to write business logic of Referenced Bean in the Java editor.



JSF Project Verification

In this chapter we'll discuss a possible verification that you can take advantage of.

Many different rules are checked for a JSF project that can be configured by selecting $\ensuremath{\textbf{Window}}$

 \rightarrow **Preferences** from the menu bar, selecting **JBoss Tools** \rightarrow **Web** \rightarrow **Verification** from the Preferences dialog box and then expanding the JSF Rules node.

type filter text JBoss Tools ESB Validator JBoss ESB Runtimes JBoss Tools Runtime Detection Project Examples Usage Reporting Verification D Expression Language JSF Label Decorations Struts Verification ModeShape Plug-in Development Project Archives Remote Systems	Preferences
 ✓ JBoss Tools ESB Validator JBoss ESB Runtimes JBoss Portlet JBoss Tools Runtime Detection Project Examples Usage Reporting ✓ Web ▷ CDI (Context and Dependency Inje ▷ Editors ▷ Expression Language ▷ JSF Label Decorations ▷ Seam ▷ Struts Verification ModeShape ▷ Plug-in Development Project Archives ▷ Remote Systems ▷ Papart Design 	ification
ESB Validator JBoss ESB Runtimes Verifities JBoss Portlet JBoss Tools Runtime Detection ▼ Project Examples Usage Reporting > Verification > > Verification > > Verification > > Verification > > Plug-in Development > > Project Archives > > Project Archives > >	les Configuration Opt
 ▷ Struts ▷ Struts ▷ Verification ▷ ModeShape ▷ Plug-in Development ▷ Project Archives ▷ Remote Systems ▷ Papert Design 	erification Level: and ✓ JSF Rules ✓ ✓ JSF Rules ✓ ✓ Check Faces C ✓ ✓ Check Manage ✓ ✓ Check Manage ✓ ✓ Check Manage ✓ ✓ Check Manage
Verification ▷ ModeShape ▷ ▷ Plug-in Development ▷ Project Archives ▷ Remote Systems ▷ Report Design	Check Phase L
 Run/Debug SCA Tools Screenshot Utility 	 Check Referent Check Render Check Render Check Render Check Validate Check Web De Check Web De Check Web De Check Struts N Check Struts C



Suppose you are working in the Source viewer for a JSF configuration file as shown below:

aces-config.xml צ		
1	xml version="1.0" encoding="UTF-8"?	
20	<faces-config 2001="" http:="" pre="" version="1.2" www.w3.org="" xinclude"<="" xmlns="http://java.sun.com/xml</th><th>ml,</th></tr><tr><th>3</th><th><pre>xmlns:xi="></faces-config>	
4	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" x	si
5⊝	<converter></converter>	
6	<converter-id>MyConverter</converter-id>	
7	<converter-class>test.Customconverter</converter-class>	>
8		
90	<managed-bean></managed-bean>	
10	<managed-bean-name>person</managed-bean-name>	
11	<managed-bean-class>demo.Person</managed-bean-class>	
12	<managed-bean-scope>request</managed-bean-scope>	
130	<managed-property></managed-property>	
14	<property-name>name</property-name>	
15	<value></value>	
16		
17		
180	<managed-bean></managed-bean>	
19	<managed-bean-name>carBean</managed-bean-name>	
20	<managed-bean-class>example.carBean<th>S></th></managed-bean-class>	S>
21	<managed-bean-scope>request</managed-bean-scope>	
220	<managed-property></managed-property>	
23	<property-name>carName</property-name>	
24	<property-class>java.lang.String</property-class>	
25	<value></value>	
26		
	< III	
Diagr	ram Tree Source	

Figure 8.2. Faces-config.xml File

While typing a class name, you might make a minor typo (like *"demo.Person9"* instead of *"demo.Person"*). After saving the file, verification checks to make sure everything is correct and finds the error below:



Figure 8.3. Error in Source View

Notice that the Package Explorer View shows a marked folder and a marked file where the error is.

You can place the cursor over the line with an error message and get a detailed error message:

ĺ	is faces-config.xml 🛱		
I	1	xml version="1.0" encoding="UTF-8"?	
I	20	<faces-config 2001="" http:="" pre="" version="1.2" www.w3.org="" xinclude"<="" xmlns="http://java.sun.com/xml,</th></tr><tr><th>I</th><th>3</th><th><pre>xmlns:xi="></faces-config>	
I	4	<pre>xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi</pre>	
I	50	<converter></converter>	
I	6	<converter-id>MyConverter</converter-id>	
I	7	<converter-class>test.Customconverter</converter-class>	
I	8		
I	9⊝	<managed-bean></managed-bean>	
I	10	<managed-bean-name>person</managed-bean-name>	
I	<mark>⊗</mark> 11	Multiple annotations found at this line:	
I	12	 Cannot find fully qualified class: demo.Person9 	
I	130	 error: Attribute managed-bean-class references to non-existent class 	
I	₫14	<property-name>name</property-name>	
I	15	<value></value>	
I	16		
I	17		
I	180	<managed-bean></managed-bean>	
I	19	<managed-bean-name>carBean</managed-bean-name>	
I	20	<managed-bean-class>example.carBean</managed-bean-class>	
I	21	<managed-bean-scope>request</managed-bean-scope>	
	220	<managed-property></managed-property>	
I	23	<property-name>carName</property-name>	
I	24	<property-class>java.lang.String</property-class>	
I	25	<value></value>	
I	26		
	- 1		
	Diagr	am Tree Source	

Figure 8.4. Error Message

Verification also checks navigation rules:



Figure 8.5. Checking Navigation Rules

If you provide a page name that does not exist, verification will let you know about that:



Figure 8.6. Page Name Verification

You can always call up verification explicitly by right-clicking any element in the tree and selecting Verify from the context menu. This works from both the Tree and Diagram viewers for the JSF configuration file editor. You can also invoke verification from the Web Projects view. Below we are checking all of the elements in the configuration file.



Figure 8.7. Verify Command

In summary, this document highlights all the JSF-specific features of JBoss Tools meant for enhancing the development of rich Web applications based on JSF technology. The reference introduces you to wizards for creating and importing JSF projects, JSF Configuration File editor features, functionality for enabling JSF capabilities and etc.

If you have questions or good suggestions, please refer to *JBoss Tools Forum* [http:// www.jboss.com/index.html?module=bb&op=viewforum&f=201].