**JSF Tools Reference Guide** 



Version: 3.1.0.CR1

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	. 3
2.1.1. Facelets templates	4
2.1.2. Facelets components	. 5
2.1.3. Code assist for Facelets	6
2.1.4. Open On feature	8
3. Projects	10
3.1. Creating a New JSF Project	10
3.2. Importing Existing JSF Projects with Any Structure	14
3.3. Adding JSF Capability to Any Existing Eclipse Project	14
3.4. Adding Your Own Project Templates	18
3.5. Relevant Resources Links	20
4. Web.xml Editor	21
5. JSF Configuration File Editor	22
5.1. Diagram view	22
5.2. Tree View	24
5.3. Source View	34
5.4. Editor Features	36
5.4.1. Open On	36
5.4.2. Code Assist	36
5.4.3. Error Reporting	37
6. Managed Beans	39
6.1. Code Generation for Managed Beans	39
6.2. Add Existing Java Beans to a JSF Configuration File	44
7. Creation and Registration	46
7.1. Create and Register a Custom Converter	46
7.2. Create and Register a Custom Validator	49
7.3. Create and Register Referenced Beans	54
8. JSF Project Verification	60

# Introduction

JSF Tools are especially designed for supporting JSF and JSF-related technologies. JSF Tools provide extensible and exemplary 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.

In this guide we provide you with the information on JSF tooling which allows you to develop JSF applications much faster and with far fewer errors so sparing your time.

# **1.1. Key Features of JSF Tools**

Here, we provide you with a key functionality which is integrated in JSF tooling.

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.	j <u>sf support</u>
Flexible and customizable project template management	Jump-start development with out-of-the-box templates or easily customized templates for re-use.	<u>projects</u>
Support for JSF Configuration File	Working on file using three modes: diagram, tree and source. Synchronization between the modes and full control over the code. Easy moving around the diagram using the Diagram Navigator.	graphical editor for jsf
Support for Managed Beans	Adding new managed beans, generating code for attributes, properties and getter/setter methods.	managed beans
Support for Custom	Fast creating of custom converters and	converters and
Converters and Validators	validators with tree view of faces-config.xml file.	<u>validators</u>
Verification and	All occuring errors will be immediately reported	verification and
Validation	by verification feature, no matter in what view you are working. Constant validation and errors checking allows to catch many of the errors during development process that significantly reduces development time.	<u>validation</u>

#### Table 1.1. Key Functionality for JSF Tools

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

All JBoss Developer Studio/JBoss Tools release documentation you can find at<u>http://</u> <u>docs.jboss.org/tools</u> [http://docs.jboss.org/tools/] in the corresponding release directory.

The latest documentation builds are available at <u>http://download.jboss.org/jbosstools/nightly-docs</u> [http://download.jboss.org/jbosstools/nightly-docs/].

# **JavaServer Faces Support**

We don't lock you into any one JavaServer Faces implementation. You can always select the one which is necessary for you while <u>creating a new JSF project</u>, <u>adding JSF capability</u> to any existing Eclipse project or <u>importing existing JSF projects</u> as well.

At this point the special wizard will prompt you to specify a proper JSF environment. It may be JSF 1.1.02 RI or JSF 1.2 which integrates a number of new features and changes. The wizard also lets you select JSF implementation with a component orientation such as JSF 1.2 with Facelets or MyFaces 1.1.4.

٥	New JSF Project
Create JSF Project	
brand new project. If	ect Wizard is used for creating a you already have a pre-existing Import Project Wizard to start working
Project Name*	JSFProject
	✓ Use default path*
Location*	he/user/workspace/JSFProject Browse
JSF Environment*	JSF 1.1.02 - Reference Implementation
	JSF 1.2
Template*	JSF 1.2 with Facelets
?	< <u>B</u> ack <u>N</u> ext > Cancel <u>F</u> inish

#### Figure 2.1. Choosing JSF Environment

After specifying a proper JSF environment all the required libraries for the selected version will be added to your project.

# 2.1. Facelets Support

In this section we will focus more on all concepts that are integrated for working with Facelets.

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

## 2.1.1. Facelets templates

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

•	New JSF Project	
Create JSF Projec	t 🌍	
The Create New Project Wizard is used for creating a brand new project. If you already have a pre-existing project, just use the Import Project Wizard to start working with it in JBoss Tools.		
Project Name*	JSFProjectwithFacelets	
	☑ Use default path*	
Location*	he/user/workspace/JSFProjectwithFacelets Browse	
JSF Environment*	JSF 1.1.02 - Reference Implementation	
	JSF 1.2	
Template*	JSF 1.2 with Facelets	
?	< <u>B</u> ack <u>N</u> ext > Cancel <u>F</u> inish	

### Figure 2.2. Choosing Facelets Environment

Once you've selected the environment, it's possible to specify the one of three available templates:

۲	New JSF Project 🗙	
Create JSF Proje	ct 🚳	
a brand new projec	oject Wizard is used for creating t. If you already have a pre-existing e Import Project Wizard to start working 5.	
Project Name*	JSFProjectwithFacelets	
	Use default path*	
Location*	e/user/workspace/JSFProjectwithFacelets	
JSF Environment*	JSF 1.2 with Facelets	
Template*	FaceletsBlankWithoutLibs	
	FaceletsKickStartWithRILibs	
	FaceletsKickStartWithoutLibs	
0	< Back Next > Finish Cancel	

#### Figure 2.3. Choosing Facelets Template

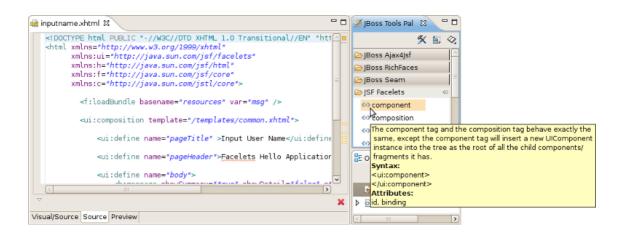
The following table lists possible templates with Facelets for any JSF project and gives a proper description for each one.

#### **Table 2.1. Facelets Templates**

Template	Description
FaceletsBlankWithoutLibs	Some servers already provide jsf libs and you take risk of getting conflicting libraries 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 <u>JBoss Tools Palette</u> [../../jsf/html\_single/index.html#palette] comes with the Facelets components ready to use. A useful tip appears when you hover the mouse cursor over the tag, the tip includes a detailed description of the tag component, the syntax and available attributes.



#### Figure 2.4. Facelets Components

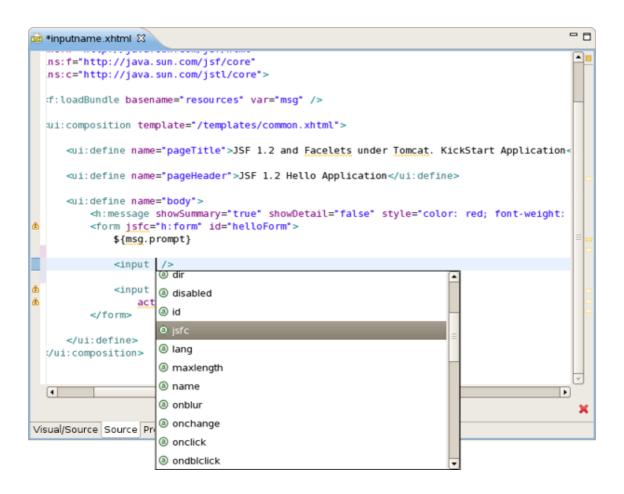
## 2.1.3. Code assist for Facelets

One more feature which comes with Facelets support is code assist (Ctrl + Space). It is available for Facelets tags while editing *.xhtml* files.

ໝ *inputname.xhtml ⊠				
<pre>\$ \$ (msg.prompt)</pre>				
Attribute : accept	accept			
Data Type : CDATA	(a) accesskey			
	(a) align="top"			
	(a) alt			
	checked="checked"			
	(a) class			
	(a) dir="ltr"			
	(a) disabled="disabled"			
	® lang			
(i) maxlength				
	la name			
Your Name: #{person.name}	Say Hello			
☆ html ui:composition ui:define form input	×			
Visual/Source Source Preview	Visual/Source Source Preview			

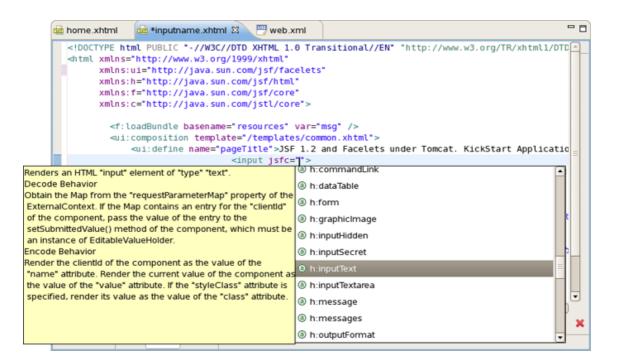
#### Figure 2.5. XHTML File Code Assist

What's more, code assist is also available for "jsfc" attribute in any HTML tag.



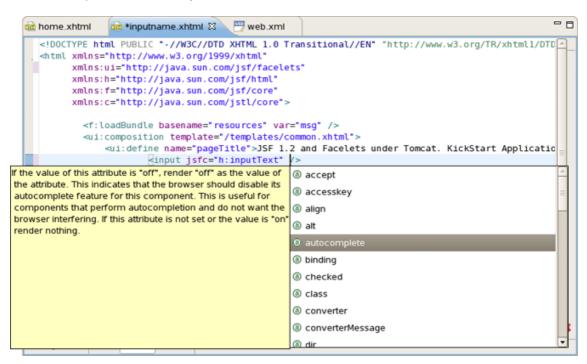
#### Figure 2.6. Code Assist for JSFC Attribute

After selecting "jsfc" you get the code assist for JSF components available on a page.



#### Figure 2.7. Code Assist for JSF Components

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



#### Figure 2.8. Available Attributes for the Component

#### 2.1.4. Open On feature

Finally, Eclipse's *OpenOn* feature for editing Facelets files is supported. Using this feature, you can easily navigate between the Facelets templates and other parts of your projects. Just by

holding down the Control key while hovering the mouse cursor over a reference to a template, the reference becomes a hyperlink to open that template.

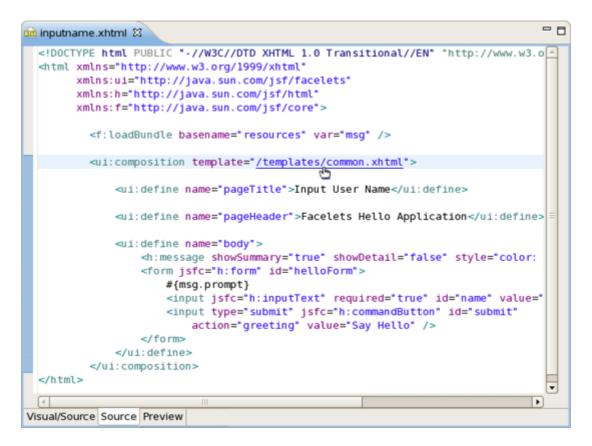


Figure 2.9. Template Hyperlink

# **Projects**

To take an advantage of JSF firstly you should 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.

In this section we're going to stop on each of them in detail.

## 3.1. Creating a New JSF Project

If you want your project to already contain all JSF libraries, tag libraries and JSF configuration file, just organize a new brand JSF project. It is possible to do this easily with the help of the special wizard. To get it, select *File > New > Project > JBoos Tools Web > JSF > JSF Project* and click *Next*.

۲	New Project	×
Select a wizard		
Create a JSF Project		
<u>W</u> izards:		
type filter text		
v 😅 mbernate		
J2EE		
👂 🗁 Java		
JBoss jBPM		
🗢 🗁 JBoss Tools Web		=
🗢 🗁 JSF		
🛣 JSF Project		
Struts		
👂 🗁 JPA		•
(2) < <u>Back</u>	Next >	nish Cancel

Figure 3.1. Choosing a JSF Project

On the next form you'll be prompted to enter Project Name and select a location for the project or just leave a default path.

Here, JSF Version also allows you to select which JSF implementation to use.

۲	New JSF Project 🗙
Create JSF Project	
brand new project. I	ject Wizard is used for creating a f you already have a pre-existing Import Project Wizard to start working
Project Name*	JSFProject
	☑ Use default path*
Location*	he/user/workspace/JSFProject Browse,
JSF Environment*	JSF 1.1.02 - Reference Implementation
	JSF 1.2
Template*	JSF 1.2 with Facelets
?	< <u>B</u> ack <u>N</u> ext > Cancel <u>F</u> inish

#### Figure 3.2. Creating a New JSF Project

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

۹	New JSF Project 🗙	
Create JSF Projec	ct 🚳 🗳	
a brand new projec	oject Wizard is used for creating t. If you already have a pre-existing Import Project Wizard to start working t.	
Project Name*	JSFProject	
	☑ Use default path*	
Location*	/home/user/workspace/JSFProject	
JSF Environment*	JSE 1 1.02 Reference Implementation  SFBlank	
Template*	JSFBlankWithLibs	
	JSFKickStart	
	JSFKickStartWithLibs	
	JSFKickStartWithoutLibs	
0	< <u>B</u> ack <u>N</u> ext > <u>F</u> inish Cancel	

#### Figure 3.3. Choosing JSF Templates

The table below provides description for each possible JSF template.

#### Table 3.1. JSF Project Templates

Template	Description
JSFBlankWithLibs	This template will create a standard Web project structure with all JSF capabilities
JSFKickStartWithLibs	This template will create a standard Web project structure but will also include a sample application that is ready to run
JSFKickStartWithoutLi	Some servers already provide jsf libs and you take risk of getting conflicting libraries while deploying your project. To avoid such conflicts, use a template without libs if you have a server with its own jsf libraries

On the next screen select what *Servlet version* to use and whether to register this application with JBoss AS (or other server) for running and testing your application.

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

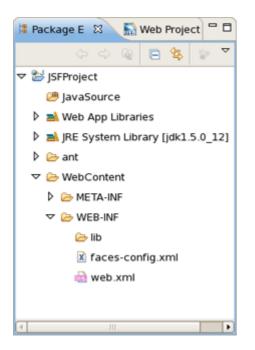
The *Runtime* value tells Eclipse where to find Web libraries in order to build (compile) the project. It is not possible to finish project creation without selecting Runtime. If you don't have any values, select *New...* to add new Runtime.

The *Target Server* allows you specifying whether to deploy the application. The Target Server corresponds to the Runtime value selected above. If you don't want to deploy the application, uncheck this value.

۲	New JSF Project	×
Web		
Servlet Version	2.4	•
Context Path*	JSFProject	
Runtime:*	JBoss 4.2 Runtime	New
Target Server:	JBoss Application Server 4.2	New
		Select All
		Deselect All
0	< <u>B</u> ack <u>N</u> ext > <u>Finish</u>	Cancel

#### Figure 3.4. Registering the Project on Server

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



#### Figure 3.5. A New Project in the Package Explorer

At this point you can open *faces-config.xml* and start working on your application. There are a lot of features to develop JSF applications. We will describe the features further.

## 3.2. Importing Existing JSF Projects with Any Structure

For detailed information on migration of JSF projects into a workspace see <u>Migration Guide</u> [../../ Exadel-migration/html\_single/index.html#jsf\_struts].

# 3.3. Adding JSF Capability to Any Existing Eclipse Project

It's also possible to add JSF capability (JSF libraries, tag libraries) to any existing Eclipse project in your workspace. After that you'll be able to make use of such editors as JSF configuration editor, JBoss Tools JSP editor and any others.

Right click the project and select *JBoss Tools > Add JSF Capabilities*. This will start the process of adding all necessary libraries, files to make this a Web JSF project.

🚦 Package Explo 😫 🔚	Web Projects 🗖 🗖		
(→			
Þ 🛃 JSF			
SomeWebProject	Ne <u>w</u>	•	
	Go Into		
	Open in <u>N</u> ew Window		
	Ope <u>n</u> Type Hierarchy	F4	
	Sho <u>w</u> In	Shift+Alt+W	
Trans	È <u>C</u> opy	Ctrl+C	
1	Copy Qualified Name		
1	Paste	Ctrl+V	
3	Celete	Delete	
Lass	Remove from Context	Shift+Ctrl+Alt+Down	
	Build Path	,	
	Source	Shift+Alt+S	
	Refactor	Shift+Alt+T 🕨	
2	<u>Import</u>		
E	🛓 Exp <u>o</u> rt		
0	Run XDoclet	Shift+Ctrl+F1	
4	P Refresh	F5	
	Clo <u>s</u> e Project		
	Close Unrelated Projects		
	Assign Working Sets		
	<u>R</u> un As	•	
	<u>D</u> ebug As	,	
	Profile As	•	
	Validate		S Add Struts Capabilities
	T <u>e</u> am	•	Add JSF Capabilities
	Comp <u>a</u> re With	•	Add Jor Capabilities
	Restore from Local History		Remove Struts Capabilities
	JBoss Tools	+	Add Custom Capabilities
	PDE To <u>o</u> ls	,	Remove JSF Capabilities
_	Java EE	•	Verify
	Properties	Alt+Enter	Modules Configuration

#### Figure 3.6. Adding JSF Capabilities

The wizard will first ask you to show the *web.xml* file location and the project name.

6	Import JSF Project X
Project Location Please select web.xr	ml location
Project Name* web.xml Location*	SomeWebProject /home/user/workspace/SomeWebProject/WebContent/WEB-INF/web.xml
0	< <u>Back</u> <u>N</u> ext > <u>Finish</u> Cancel

#### Figure 3.7. Project Location

On the last form you can set the different folders for your project as well as register this application with a servlet container.

Make sure to select Add Libraries to add all required JSF related libraries to this project.

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

The *Runtime* value tells Eclipse where to find Web libraries in order to build (compile) the project. It is not possible to finish project import without selecting Runtime. If you don't have any values, select *New...* to add new Runtime.

The *Target Server* allows you to specify whether to deploy the application. The Target Server corresponds to the Runtime value selected above. If you don't want to deploy the application, uncheck this value.

٥	Import JSF Project 🗙
Project Folders Select Project Fol	
Web Root:*	/home/user/workspace/SomeWebProject/WebContent
Source Folder	/home/user/workspace/SomeWebProject/src
Classes Folder	/home/user/workspace/SomeWebProject/build/classes
Lib Folder	/home/user/workspace/SomeWebProject/WebContent/WEB-INF/lib
	☑ Add Libraries
Environment	JSF 1.1.02 - Reference Implementation
Servlet Version:	2.5
Context Path*	SomeWebProject
Runtime:*	JBoss 4.2 Runtime New
Target Server:	JBoss Application Server 4.2 New
	Select All
	Deselect All
0	< <u>B</u> ack <u>Next</u> <u>Finish</u> Cancel

#### Figure 3.8. Project Folders

Once your project is imported you can see that JSF related libraries have been added to your project: *jsf-api.jar* and *jsf-impl.jar*.



You are now ready to work with JSF by creating a new JSF configuration file:

🕯 Package E 🛛 🔝	Web Project	
	🖻 🔩 🐌 🍸	
👂 ڬ JSFProject 💦 🗧		
🗢 涉 SomeWebProjec	Ne <u>w</u>	🟦 JSF Project
👂 🤔 JavaSource	Go Into	Struts Project
👌 📸 Web App Libr	Open in <u>N</u> ew Window	🎦 Project
👂 勳 JRE System L	Sho <u>w</u> In Shift+Alt+W	豑 Package
🗢 🧀 ant	È⊆opy Ctrl+C	G Class
In collection on the second	Copy Qualified Name	🕃 Interface
The second se	Paste Ctrl+V	Source Folder
	K Delete Delete	Folder
D 🗁 META-INF	Remove from Context Shift+Ctrl+Alt+Down	Struts Config
👂 🗁 pages	Build Path	A Faces Config
🗢 🗁 WEB-INF	Refactor Shift+Alt+T	Tiles File
Þ 🗁 lib 🚽	-	🕙 Validation File
V Su urs-D	import	📸 JSP File
🖹 struts-c	A Export	🚵 XHTML File
< struts-h 🤞	Refresh F5	🃸 HTML File
📀 struts-lo	Assign Working Sets	🚳 CSS File
🗴 validati	Validate	🖄 JS File
🗴 validato	Run As	Properties File
🗟 web.xn	Debug As	📑 E <u>x</u> ample
📾 index.jsp	Profile As	📑 <u>O</u> ther
	T <u>e</u> am •	Ca Quier
	Comp <u>a</u> re With	
< III	Restore from Local History	
	Source	
	Properties Alt+Enter	

#### Figure 3.9. Creating a New JSF Configuration File

Once the file has been created, it should be opened in a special *Faces Config Editor*.

## 3.4. Adding Your Own Project Templates

Template is a set of files that is served as a basis to facilitate the creation of a new project. Project templates provide content and structure for a project.

There is a powerful templating capability for creating new and importing existing Struts and JSF projects. This templating facility has a variety of aspects to consider. But, 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. Follow these steps to make a template out of it:

 In the Web Projects view, right-click the project and select JBoss Tools JSF > Save As Template

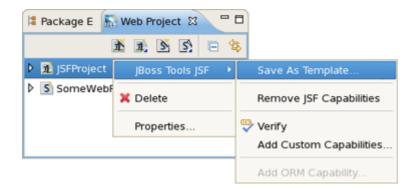


Figure 3.10. Saving Your Project as Template

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

۹	Add JSF Project Template	×
Define Common	Template Properties	0
Name:*	MyJSFProject	
Implementation:*	JSF 1.1.02 - Reference Implementation	•
	Next >> Finish C	ancel

#### Figure 3.11. Define Template Properties

• Select *Next* and you will be sent to a dialog box with your project structure displayed with check boxes. Here you can check 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	
▽ 🖃 🐸 JSFProject	
🗆 🗁 .settings	
🗹 🗁 JavaSource	
🗢 🗹 🗁 WebContent	
🗹 🗁 META-INF	
arr 🗹 🗁 WEB-INF	
🗆 🗁 classes	
🗆 🗁 lib	
🗹 🗁 ant	
<< Back Next >	> Finish Cancel

#### Figure 3.12. Define Template Properties

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

That's it. Now, you can use this template 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 and now you can bring some new logic to it starting from JSF configuration file.

## 3.5. Relevant Resources Links

You can find more in-depth explanation on how to work with special wizards, editors and views that can be used in various scenarios while developing JSF applications in our <u>Visual Web Tools</u> <u>guide</u> [../../jsf/html\_single/index.html].

# 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 that gives a descriptive information on the *web.xml editor* [../../jsf/html\_single/index.html#GraphicalWebApplicationFileEditor].

# **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 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 in the facesconfig.xml:

🔊 faces-config.xml ដ	- 0
/chooseLocale.jsp /storeFront.jsp storeFront carDetail	
/carDetail.jsp /confirmChoices.jsp confirmChoices CustomerInfo carDetail /CustomerInfo.jsp	
/finish.jsp finish	
Diagram Tree Source	

Figure 5.1. Diagram View

If your diagram is large, 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. It 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:

אfaces-config.xml מ	' D	🗄 Outline 🕴 🚿 JBoss Tools Palette	- 8
/chooseLocale.jsp	Ĥ	E	a 🗗 🗸
<pre>/chooseLocale.jsp</pre>		Para and a second secon	-
		B- B-	
			10201001101
/storeFront.jsp			- D-
carDetail.jsp			
confirmChoice	e		
	Ū		
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:

🔊 fac	es-config.xml 🛿	
	/chooseLocale.jsp	/storeFront.jsp

#### 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 *New View*.

	oseLocale.j:	/storeFront.jsp Front carDetail		
the standard state		/confirmChoices.jsp nfirmChoices CustomerInf carDetail New View	rinfo.jsp	
		Auto Layout Select Element	finish	/finish.jsp
		Verify           Paste         Ctrl + V		
		Input Methods		
Diagram Tre	e Source			

#### 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:

		/carDetail.jsp
		X
1.1	/storeFront.jsp	
	arDetail	
	carbeam	/chooseLocale.jsp
117		

#### 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 see and edit its properties which will appear in the righthand 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, possibility to set extensions, add property and variable resolvers, etc.

🔊 *faces-config.xml 🛙		- 0	
Faces Config Editor			
▼ faces-config	► Application		
▽ 🙇 faces-config.xml*	EL Resolvers		
Application	Property Resolvers		
🄄 Components	Variable Resolvers		
🏟 Converters	Message Bundles		
👂 🍋 Managed Beans	Resource Bundles		
👂 🍓 Navigation Rules	- Locale Config		
leans 😂 Referenced Beans	Default-Locale: en		
🔄 Render Kits	Supported Locale		
✓ Validators	Locale		
Section Sectio	en_US	<u>A</u> dd	
	de	Bemove	
		<u>E</u> dit	
		Шр	
		Down	
	Extensions		
	- Advanced		
	ID:		
	L		
Diagram Tree Source			

#### Figure 5.6. JSF Application Specific Settings

The *Components* node is for registering custom JSF components. Right-click and choose *New Component* or just press the *Add* button in the right-hand area to add a new component to the JSF Configuration file.

aces-config		Components		
saces-config.xml		type	class	<u>A</u> dd
Application				<u>R</u> emove
Components	New		ڬ 💽 Component	<u>E</u> dit
A Converters Managed Beans	of Cut	Ctrl + X		·
Navigation Rules	📄 Сору	Ctrl + C		Цр
Referenced Beans	👔 Paste	Ctrl + V	1	Down
Render Kits	🕱 Delete	Delete		
🗭 Validators	Propert	ies		
🗭 Extensions		les		
	💛 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.

G	Add Component	×
Component Attribute Component	-Type must be set.	
Component-Type* <u>Component-Class*</u>		Browse
Ø		Einish Cancel

## 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.

aces-config	▼ Ren	der Kits		
s faces-config.xml	id		class	<u>A</u> dd
Application Components				<u>R</u> emove
Converters				Edit
👂 🍓 Managed Beans				Up
👂 🍓 Navigation Rules				Down
🗟 Referenced Bean	s			Dowu
🔄 Render Kits	New		> 🌆 Render Kit	
✓ Validators	∘≴ Cut	Ctrl + X		
🐼 Extensions	Сору	Ctrl + C		
	💼 Paste	Ctrl + ∨	1	
	🗱 Delete	Delete		
	Properties			

## Figure 5.9. Adding a New JSF Component to the JSF Configuration File

• Under the *Converters* node you can create a converter class for your JSF application either with id or for a proper class. How to do that see the *Create and Register a Custom Converter* section.

faces-config	- Conve	rters		
🗸 🔊 faces-config.xm	id	class		<u>A</u> dd
Application Components				<u>R</u> emove
🗟 Converters	New		🕸 Converter with id	<u>E</u> dit
👂 🍓 Managed Bea			Converter for class	Цр
Navigation Ru		Ctrl + X Ctrl + C		Down
leferenced B				Benn
🔄 Render Kits	💼 Paste	Ctrl + V		
🧭 Validators	🗱 Delete	Delete		
	Properties			
	😌 Verify			
	Venry			

#### Figure 5.10. Creating a New Custom Converter

• The *Managed Bean* node is meant for creating and registering Bean classes in your JSF application. Read more on the topic in the *Managed Beans* chapter.

faces-cor	nfig	<ul> <li>Managed I</li> </ul>	Beans		
🔊 faces-	config.xml	name	class	scope	<u>A</u> dd
	lication nponents	NA	components.	moc applicatio	<u>R</u> emove
	iverters				Edit
-	aged Beans New	•	A Managed Bean		<u>Ц</u> р
Þ 001			🥔 Managed Bean		Down
👂 🍓 Nav	of Cut	Ctrl + X			Down
🗟 Ref	[ Сору	Ctrl + C			
🔄 Rer	💼 Paste	Ctrl + V			
🧭 Vali	💢 Delete	Delete			
	Properties				
	哭 Verify				

#### Figure 5.11. Managed Beans

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



🛕 faces-config.xml 🕴					- 0
Faces Config Editor					
▼ faces-config	• Navigation R	ule			
▼ M faces-config.xml	From-View-ID: Description:	/pages/inputUserNa	ime.jsp		Browse
Converters		<	III		>
Managed Beans	<ul> <li>Navigation C</li> </ul>	ases			
	From-Outcome	From-Action	To-View-ID	Redir	Add
✓ I /pages/inputUserNan → hello	hello		/pages/hello.jsp	no	Bemove
😂 Referenced Beans					Edit
🖓 Render Kits					Up
✓ Validators					
✓ Extensions					Down
	<ul> <li>Advanced</li> </ul>				
	ID:				
	Display-Name:				
	Small-Icon:				
	Large-lcon:				
Diagram Tree Source					

## 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 the <u>Create and Register Referenced Beans</u> section.

faces-config	▼ Rei	erenced Bean	5	
faces-config.xml	nam	e	class	<u>A</u> dd
🔊 Application 😂 🏹				<u>B</u> emove
Converters				Edit
👂 🍋 Managed Beans				
👂 🍓 Navigation Rules				Down
lean:	s New	>	🥏 Referenced Bean	
Render Kits	-k Cut			
✓ Validators	of Cut	Ctrl + X Ctrl + C		
😸 Extensions	Copy	Ctrl + V		
	🗱 Delete	Delete		
	Properties			
	💛 Verify			
			1	
	Verily		]	

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 the <u>Create and Register a Custom Validator</u> section.

faces-config	▼ Va	lidators	
iaces-config.xm	id	class	<u>A</u> dd
Application Components			<u>R</u> emove
Converters			<u>E</u> dit
Managed Bea			<u>U</u> p
Referenced B			Down
Render Kits			
🖋 Validators	New	> 🗹 Validator	
🧭 Extensions	o∉ Cut	Ctrl + X Ctrl + C	
	Paste	Ctrl + V	
	🗶 Delete	Delete	

Figure 5.14. Validators

• The *Extensions* node is for setting extensions for your *faces-config.xml*.

🔹 faces-config.xml 🕱				- 0
Faces Config Editor				
✓ faces-config	▼ Extensio	ons		
▽ 🙇 faces-config.xml	element			
Application				<u>A</u> dd
Components				Bemove
<ul><li>Converters</li><li>Managed Beans</li></ul>				Edit
Navigation Rules				
Referenced Beans				
🖳 Render Kits				
✓ Validators				
🖌 Extensions	New		Extension	
06	Cut	Ctrl + X		
	Сору	Ctrl + C		
Ci i	Paste	Ctrl + V		
34	Celete Celete	Delete		
< 111	Properties			
Diagram Tree Source				

#### 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 않				- 0	Properties	🗉 🔅 🛤	~ - 8
Faces Config Editor					Property	Value	
▼ faces-config	✓ Managed Bean			1	comment		
<ul> <li>✓ Mathematical faces-config.xml<sup>4</sup></li> <li>▲ Application</li> <li>⇐ Components</li> <li>⇐ Converters</li> <li>⇐ Managed Bean:</li> <li>♥ Ø person</li> </ul>	Managed-Bean-Name: <u>Managed-Bean-Class:</u> Managed-Bean-Scope: Description:	person demo.Person request	Browse	=	description display-name id large-lcon managed-bean-class managed-bean-name		
<ul> <li>name</li> <li>込 (確認) Navigation Rule</li> </ul>	- Properties	4			managed-bean-scope small-icon	request	
Referenced Ben Render Kits Validators C	name d	class value	<u>A</u> dd <u>R</u> emove <u>E</u> dit	•			
Diagram Tree Source					( III		

Figure 5.16. Properties View

## 5.3. Source View

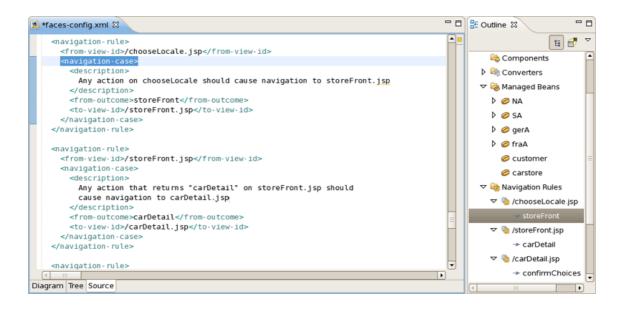
Here, we'll discuss how you can configure your faces-config.xml with the help of Source View.

The Source view for the editor displays a 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:

```
- 8
🏄 *faces-config.xml 🖾
                                                                               .
   <managed-bean>
     <description>
       Causes the default VariableResolver implementation to instantiate
       the managed bean, CustomerBean of the class, carstore.CustomerBean
       in session scope if the bean does not already exist in any scope.
     </description>
     <managed-bean-name> customer </managed-bean-name>
     <managed-bean-class> carstore.CustomerBean </managed-bean-class>
     <managed-bean-scope> session </managed-bean-scope>
   </managed-bean>
   <managed-bean>
     <description>
       The main backing file mean
     </description>
     <managed-bean-name> carstore </managed-bean-name>
     <managed-bean-class> carstore.CarStore </managed-bean-class>
     <managed-bean-scope> session </managed-bean-scope>
   </managed-bean>
   <navigation-rule>
     <from-view-id>/chooseLocale.jsp</from-view-id>
     <navigation-case>
       <description>
         Any action on chooseLocale should cause navigation to storeFront.jsp
       </description>
       <from-outcome>storeFront</from-outcome>
        <to-view-id>/storeFront.jsp</to-view-id>
                                                                               Ŧ
      </navigation-case>
                                                                             .
Diagram Tree Source
```

#### 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 for work with JSF resources.

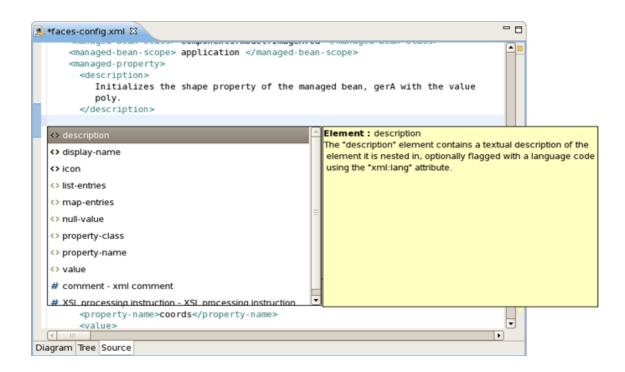
#### 5.4.1. Open On

The JSF configuration file editor comes with a very useful OpenOn navigating feature. More fully you can read about it in our <u>Visual Web Tools Guide</u> [../../jsf/html\_single/ index.html#OpenOnSelection4Hyperlinknavigation].

#### 5.4.2. Code Assist

Code Assist provides 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:



#### Figure 5.19. Code Assist in Source View

### 5.4.3. Error Reporting

When you are developing your project, error checking is constantly provided. This greatly reduces your development time as it allows you to catch many of the errors during development.

Errors will be reported by <u>verification</u> facility:



#### Figure 5.20. Error Reporting in Source View

Other errors are also reported.

<pre><man -="" aged="" bean="" name=""> </man></pre> <sometag <="" pre=""> <navigation -="" rule=""> <pre><from -="" id="" view="">/chooseLocale.j </from></pre> <pre></pre></navigation></sometag>	<b>sp</b> <th>?w-id&gt;</th> <th>•</th>	?w-id>	•
Diagram Tree Source			Þ
Problems 🛱 Tasks Servers			
2 errors, 9 warnings, 0 infos			
Description	Resource	Path	Location
🗢 🔚 Errors (2 items)			
8 181:12 Element type "sometag"	ı faces-config.	cardemo/WebContent/WE	line 181
😣 182:4 The element type "someta	a faces-config.	cardemo/WebContent/WE	line 182

Figure 5.21. Other Errors Reporting

# **Managed Beans**

There is lots of power to work with managed beans.

- · Add and generate code for new managed beans
  - · Generate code for attributes and getter/setter methods
- Add existing managed beans to JSF configuration file

Thus, in this section we will guides you through all this possibilities.

### 6.1. Code Generation for Managed Beans

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

🔊 faces-config.xml 🛙				- 8
Faces Config Editor				
▼ faces-config	• Managed Bea	ins		
🗢 🔊 faces-config.xml	name	class	scope	<u>A</u> dd
Application	NA	components.r	n request	Remove
Components	SA	components.r	n request	
Converters				Edit
Vew	•	🥔 Managed Bean		Цр
⊘ N. ⊘ S, & Cut	Ctrl + X			Down
Navi Copy	Ctrl + C			
Refe	Ctrl + V			
Renc X Delete	Delete			
Valid Properties				
Verify				
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* is checked as shown in the figure below.

0	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 the "Java" class:

• click on Managed-Bean-Class link in the editor

or

• right click the managed bean and select Open Source

🏄 faces-config.xml 🛿				- 0
Faces Config Editor				
▼ faces-config	<ul> <li>Managed Bean</li> </ul>			<b>^</b>
🗢 🔊 faces-config.xml 🦳	Managed-Bean-Name:	carBean		1
<ul> <li>Application</li> <li>Components</li> </ul>	Managed-Bean-Class:	example.car8ean	Browse	
Converters	Managed Open Cope:	request	•	=
🗢 🍓 Managed Bean	Description:			i
🥟 NA 🔤		<u></u>		
Ø SA			Þ	
🥏 carBean	<ul> <li>Properties</li> </ul>			
🍓 Navigation Rule	name	class value	<u>A</u> dd	
🗟 Referenced Be			Remove	
Render Kits			Memove	
Solution Validators			<u>E</u> dit	
			11=	•
Diagram Tree Source				

#### Figure 6.3. Opening of Created Managed Bean

The generated Java source should look as follows:

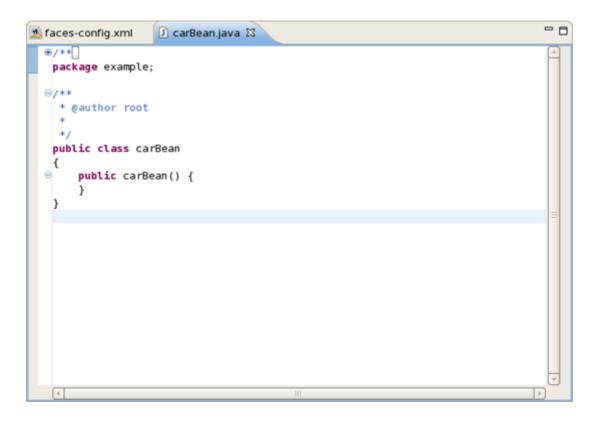


Figure 6.4. Java Source Code

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

🔊 fac	es-c	:onfig.xml 🛛 🛛	🕽 carBean.jav	а					- 0
Fac	es	Config Editor							
▼ fa	ices	-config	▼ Mana	geo	i Bean				A
▼.	ē	Application Components	-		ean-Name: ean-Class:	carBean example.c	arBean	<u>B</u> rowse	
	G	Converters	Manage	d-Be	ean-Scope:	request		•	=
	⊽ 緯	Managed Bean	Descrip	tion	:	4	Ш	×	
		🥏 carBean	+ Prop	erti	es				H
		New	•	٠	Property	class	value	<u>A</u> dd	
		Open Declaration	1		List-Entries			Remove	
		Rename Class		_	Map-Entries	]			
ار Diagra		Cut Copy Paste	Ctrl + X Ctrl + C Ctrl + V					<u>E</u> dit	•
	×	Delete	Delete						
		Properties							
	•	Verify							

#### 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

۲	Add Property	×
Property		0
Property-Name:*	carName	•
Property-Class:	java.lang.String	<u>B</u> rowse
Value Kind	value	-
Value:		<u>C</u> hange
	🗹 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 added property with "*get*" and "*set*" methods:

🔊 *faces-config.xml 🕖 carBean.java 🕴	- 8	
*/ package example;	<b>^</b>	
⊖/**		
* @author user * */		
<pre>public class carBean {     private java.Lang.String carName;</pre>		
<pre> public carBean() {     } }</pre>	=	
<pre> public java.Lang.String getCarName() {     return carName; } </pre>		
<pre> public void setCarName(java.Lang.String carName) {     this.carName = carName;   } }</pre>	*	
	Ð	

#### Figure 6.7. Generated Java Source Code for Property

Thus, we've discussed everything which comes to 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*	CarBean
	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 *Next* button will be activated. Pressing it you'll get *Managed Properties* dialog where all corresponding properties are displayed. Check the necessary ones to add them into your JSF Configuration File.

8	New Managed Bean
Managed Pro Select propertie	perties es 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 Finish.

Above-listed steps have demonstrated how you can specify an existing Bean in the JSF configuration file, i.e. *faces-config.xml*. In the next chapter you'll know how to organize and register another kind 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's necessary to go through the following steps:

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

Id     class       Id     class       Id     class       Id     class       Id     class	Add <u>Remove</u> <u>E</u> dit
Components	
😹 Managed Beans	
	Up
kavigation Rules	Down
Seferenced Beans	
Image: A start of the start	

#### Figure 7.1. Converters

- Select Converters and click on 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.

۹	Add Converter	×
Converter		0
Converter-id:*	MyConverter	
Converter-Class:*	test.Customconverter	Browse
0	Enish	Cancel

Figure 7.2. Add Converter Form

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

faces config	- Converter		
faces-config	* Converter		
✓ Maces-config.xml*	Converter-id:	MyConverter	
Application Components	Converter-for-Class:		<u>B</u> rowse
	Converter-Class:	test.Customonverter	<u>B</u> rowse
均 MyConverter	Description:		
leans 🖉		III	V
🍓 Navigation Rules 阔 Referenced Beans	★ Attributes		
Render Kits	name	class	<u>A</u> dd
🧭 Validators			<u>R</u> emove
			<u>E</u> dit
			Цр
			Down

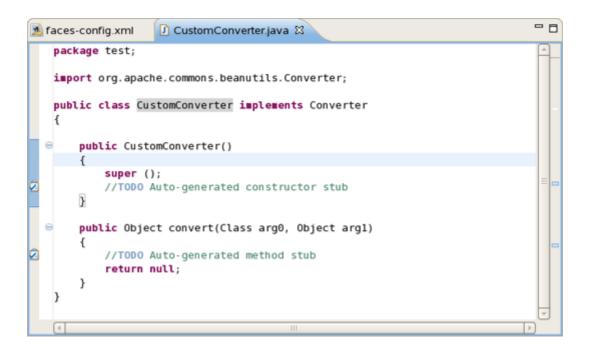
#### Figure 7.3. Generation of Source Code for Converter Class

• 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 *Finish*.

0	New Java Class	×
<b>Java Class</b> Create a new Java	class.	C
Source folder:	JSFProject/JavaSource	Br <u>o</u> wse
Pac <u>k</u> age:	test	Bro <u>w</u> se
Enclosing type:		Bro <u>w</u> se
Na <u>m</u> e:	Customconverter	
Modifiers:	public O default O private O protected	
<u>S</u> uperclass:	java.lang.Object	Brows <u>e</u>
Interfaces:		Add
		<u>R</u> emove
Which method stub	s would you like to	
	public static void main(String[] args)	
	☑ <u>C</u> onstructors from superclass	
	☑ Inherited abstract methods	
Do you want to add	comments as configured in the properties of the curren	t project?
	Generate comments	
0	Enish	Cancel

#### Figure 7.4. New Java Class Form

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



#### Figure 7.5. Converter Class

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

# 7.2. Create and Register a Custom Validator

It's also quite easy to develop your own custom Validators. You should perform the actions similar to the previous one. Go through the following steps:

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

aces Config Editor			
faces-config	- Validat	tors	
🗸 🔊 faces-config.xml	id	class	<u>A</u> dd
Application			Remove
🍣 Components 🏟 Converters			Edit
Managed Beans			
avigation Rules			<u>Up</u>
leferenced Beans			Down
Render Kits			
💕 Validators			

Figure 7.6. Validator in Faces Config Editor

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

	Add Validator	×
Validator		
Validator-id:*	MyValidator	
Validator-Class:*	test.CustomValidator	Browse
0		<u>F</u> inish Cancel

#### 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*.

aces Config Editor			
faces-config	- Validator		
7 🛃 faces-config.xml*	Validator-id:	MyValidator	
Application     Application     Components     Application     Applica	Validator-Class:	test.CustomValidator	Browse
🍳 Converters	Description:	4	
🍓 Navigation Rules 🍓 Referenced Bean:	- Attributes		
Render Kits	name cl	ass	Add
🗢 🧭 Validators			<u>B</u> emove
🗹 MyValidator			<u>E</u> dit
			<u>⊔</u> р
			<u>D</u> own
	• Properties		

Figure 7.8. Creating Validator Class

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

•	New Java Class	×
<b>Java Class</b> Create a new Java	class.	C
Source fol <u>d</u> er:	JSFProject/JavaSource	Browse
Pac <u>k</u> age:	test	Bro <u>w</u> se
Enclosing type:		Bro <u>w</u> se
Na <u>m</u> e:	CustomValidator	
Modifiers:	<u>public</u> O default O private O protected	
	abstract final static	
<u>S</u> uperclass:	java.lang.Object	Brows <u>e</u>
Interfaces:		<u>A</u> dd
		<u>R</u> emove
Which method stub	s would you like to	
	public static void main(String[] args)	
	☑ Constructors from superclass	
	☑ Inherited abstract methods	
Do you want to add	comments as configured in the properties of the curren	t project?
	<u>Generate comments</u>	
0	Einish	Cancel

#### Figure 7.9. New Java Class Form

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

```
- 8
                     🕖 CustomValidator.java 🛛
🏂 *faces-config.xml
   package test;
  import javax.faces.component.UIComponent;
   import javax.faces.context.FacesContext;
   import javax.faces.validator.Validator;
   import javax.faces.validator.ValidatorException;
   public class CustomValidator implements Validator
   {
       public CustomValidator()
           super ();
           // TODO Auto-generated constructor stub
       3
       public void validate(FacesContext arg0, UIComponent arg1, Object arg2) throws
  e
ValidatorException
       {
           //TODO Auto-generated method stub
1
       }
   }
```

#### Figure 7.10. Converter Class Editing

# 7.3. Create and Register Referenced Beans

Creation of Referenced Beans is similar to creation of Custom Validator as well. To perform this, let's walk through the necessary steps.

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

faces-config	▼ Reference	ed Beans	
🔊 faces-config.xml	name	class	<u>A</u> dd
Application			Remove
a Components			
😂 Converters			<u>E</u> dit
👂 😂 Managed Beans			Up
🍓 Navigation Rules			
leans 😽 Referenced Beans			Down
🔄 Render Kits			
👂 🥩 Validators			

#### Figure 7.11. Referenced Beans in Faces Config Editor

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

Ad Ad	d Referenced Bean
Referenced Bean	
Referenced-Bean-Name:*	MyReferencedBean
Referenced-Bean-Class:*	test.ReferencedBean <u>B</u> rowse
0	Einish 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.

Referenced Bear-N Referenced-Bean-N		
	Name: MyReferencedBean	
Referenced-Bean-C         Description:         ✓ Advanced         Id:         Display-Name:         Small-Icon:         Large-Icon:	Class: test.ReferencedBean Brow	/se
	Advanced  Id: Display-Name: Small-Icon:	✓ Advanced  Id: Display-Name: Small-Icon:

Figure 7.13. Create Referenced Bean Class

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

۹	New Java Class	×
<b>Java Class</b> Create a new Java	class.	C
Source fol <u>d</u> er:	JSFProject/JavaSource	Br <u>o</u> wse
Package:	test	Bro <u>w</u> se
Enclosing type:		Bro <u>w</u> se
Na <u>m</u> e:	ReferencedBean	
Modifiers:	public O default O private O protected     abstract I final I static	
<u>S</u> uperclass:	java.lang.Object	Brows <u>e</u>
Interfaces:		<u>A</u> dd
		<u>R</u> emove
Which method stub	s would you like to	
	public static void main(String[] args)	
	☑ Constructors from superclass	
	Inherited abstract methods	
Do you want to add	comments as configured in the properties of the curren	t project?
	Generate comments	
0	Einish	Cancel

#### Figure 7.14. New Java Class Form

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

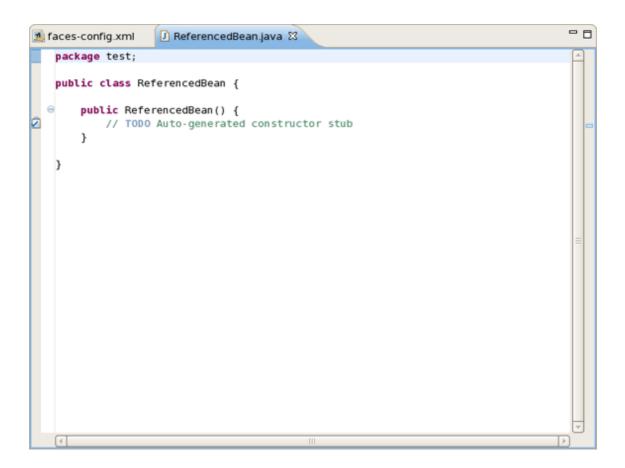
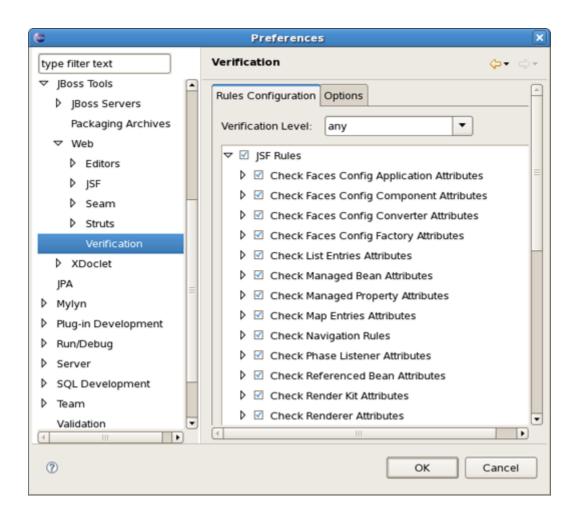


Figure 7.15. Referenced Bean Class Editing

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



#### Figure 8.1. JSF Rules

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

faces-config.xml 🛙		-
xml version="1.0"?		E A
faces-config PUBLIC</td <td>C "-//Sun Microsystems, Inc.//DTD JavaServer Fac</td> <td></td>	C "-//Sun Microsystems, Inc.//DTD JavaServer Fac	
	"http://java.sun.com/dtd/web-facesconfig_1_1.d	td">
<faces-config></faces-config>		
<managed-bean></managed-bean>		
<man -="" aged="" ame="" bean="" n="">personBe</man>		
	lo.PersonBean	
<managed-bean-scope>reques</managed-bean-scope>	t	
<managed-property></managed-property>		
<pre><pre>roperty-name&gt;name</pre></pre>	erty-name>	
<value></value>		
<navigation-rule></navigation-rule>		
<from-view-id>/pages/input</from-view-id>	name.jsp	
<navigation-case></navigation-case>		
<from-outcome>greeting<td>rom-outcome&gt;</td><td></td></from-outcome>	rom-outcome>	
<to-view-id>/pages/greeting</to-view-id>	ng.jsp	
		L
G.		

#### Figure 8.2. Faces-config.xml File

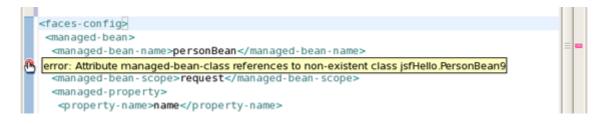
While typing a class name, you might make a minor typo (like *"jsfHello.PersonBean9"* instead of *"jsfHello.PersonBean"*). After saving the file, verification checks to make sure everything is correct and finds the error below:



#### 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:



#### 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 <u>JBoss Tools Forum</u> [http:// www.jboss.com/index.html?module=bb&op=viewforum&f=201].