Visual Web Tools Reference Guide

Version: 3.3.0.M5

1. Visual Web Tools 1
1.1. Key Features of Visual Web Tools 1
1.2. Other relevant resources on the topic 3
2. Spring Tools 5
2.1. Spring IDE guide 5
2.1.1. Add Spring Project Nature 5
2.1.2. Create New Spring Project 5
2.1.3. Add References To Other Spring Projects 5
2.1.4. Add Spring Beans Config Files 5
2.1.5. Create Spring Beans Config Sets 5
2.1.6. Open Spring Explorer 5
2.1.7. Validate Spring Beans Config 5
2.1.8. Open Spring Beans Graph 6
2.1.9. Search Spring Beans 6
3. Editors 7
3.1. Editors Features 7
3.1.1. OpenOn 7
3.1.2. Content Assist 18
3.1.3. Synchronized Source and Visual Editing 44
3.2. Visual Page Editor 46
3.2.1. Visual/Source View 47
3.2.2. Pages Styling 58
3.2.3. Visual Templates for Unknown Tags 69
3.2.4. Export/Import of the Templates for Unknown Tags 72
3.2.5. VPE Toolbar 74
3.2.6. Page Preview 91
3.2.7. Error Messages 92
3.2.8. Support for Custom Facelets Components 93
3.3. More Editors 98
3.3.1. Graphical Properties Editor 98
3.3.2. Graphical Tag Library Editor 101
3.3.3. Graphical Web Application File (web.xml) Editor 106
3.3.4. CSS Editor 112
3.3.5. JavaScript Editor 114
3.3.6. XSD Editor 116
3.3.7. Support for XML Schema 122
4. Browsers 125
4.1. Generic web browser 125
4.2. Mobile web browser 126
4.2.1. System requirements 126
4.2.2. Using BrowserSim 127
5. JBoss Tools Palette 135
5.1. Palette Options 136
5.1.1. Palette Editor 136

5.1.2. Show/Hide 144
5.1.3. Import 145
5.2. Using the Palette 146
5.2.1. Inserting Tags into a JSP File 146
5.2.2. Adding Custom JSF Tags to the JBoss Tools Palette 148
6. CSS Editing Perspective 153
6.1. Outline view 154
6.2. Properties view 156
6.3. CSS Properties view 157
6.4. CSS Preview 161
7. RichFaces Support 163
7.1. Code Assist for RichFaces 163
7.2. OpenOn for RichFaces 164
7.3. RichFaces in the JBoss Tools Palette 165
7.4. Relevant Resources Links 168
8. Web Projects View 169
8.1. Project Organization 169
8.2. Drag and Drop 170
8.2.1. For a Property 170
8.2.2. For Managed Bean Attributes 172
8.2.3. Navigation Rules 172
8.2.4. For a Tag Library File Declaration 174
8.2.5. For JSP Pages 175
8.3. Developing the Application 176
8.4. Expanding Tag Library Files 177
8.5. Drag and Drop Tag Libraries on to JBoss Tools Palette 179
8.6. Create and Import JSF Projects 179
9. JBoss Tools Preferences 181
9.1. Project Archives 183
9.2. Editors 185
9.3. Visual Page Editor 187
9.4. Visual Page Editor Code Templates 192
9.5. EL Variables 194
9.6. JSF 196
9.7. JSF Project 198
9.8. JSF Flow Diagram 202
9.9. Label Decorations 206
9.10. Seam 209
9.11. Seam Validator 211
9.12. Seam Pages Diagram 213
9.13. Tiles Diagram 215
9.14. Server Preferences 217
9.15. XDoclet 225
10. Context Menu Preferences and Options 231

10.1. Add/Remove JSF Capabilities 231
10.2. Add Custom Capabilities 231
11. FAQ 235
11.1. What should I do if Visual Page Editor does not start under Linux? 235
11.2. How do I change the auto-formating preferences for the Visual Page Editor? 236
11.3. Visual Editor starts OK, but the Missing Natures dialog appears 236
12. Conclusion 239

Visual Web Tools

This guide covers the usage of Visual Web Tools in JBoss Developer Studio and JBoss Tools. The difference between these products is that JBoss Tools are just a set of Eclipse plugins, where JBoss Developer Studio adds the following functionality:

- An installer
- Eclipse and Web Tools preconfigured
- JBoss EAP with JBoss AS and Seam preconfigured
- Third party plugins bundled and configured
- Access to Red Hat Enterprise Linux and Red Hat Network
- Access to the JBoss and Red Hat supported software

For additional information, please visit the *JBoss Developer Studio home page* [http://www.jboss.com/products/devstudio].

In JBoss Tools there is an extensive collection of specialized wizards, editors and views that can be used in various scenarios while developing Web applications. The following chapters walk through these features.

1.1. Key Features of Visual Web Tools

Here is the table of the main features of Visual Web Tools:

Table 1.1. Key Functionality for Visual Web Tools

Feature	Benefit	Chapter
Visual Page Editor	Powerful and customizable visual page editor that provides the ability to develop an application using any web technology including JSF, Seam, JSP, HTML and others. Development is done using three tabs: **Visual/ Source**, **Source** and **Preview**. Fast and easy switching between these tabs. Split screen design of visual and source views. Full and instant synchronization between source and visual views. Integration with properties and outline views. Graphical toolbar to add inline styling to any tag.	*Section 3.2, "Visual Page Editor"*
Multiple Editors	An extensive collection of specialized editors for different file types including properties, TLD, `web.xml`, tiles, and so on. These	*Section 3.3, "More Editors"*

Feature	Benefit	Chapter
	include Graphical Properties Editor, Graphical TLD Editor, Graphical Web Application File (`web.xml`) Editor, CSS Editor, JavaScript Editor, XSD Editor, and support for XML Schema.	
JBoss Tools Palette	Organizing various tags by groups, inserting tags into a `JSP` or `XHTML` page with one click, adding custom or 3rd party tag libraries into the palette, easy controlling the number of tag groups shown on the palette.	*Chapter 5, JBoss Tools Palette*
Web Projects View	Visualizing and displaying projects by function. Easy selecting of different kinds of items and dropping them into `JSP` pages. Using context menus to develop the application. Using icon shortcuts to create and import JSF projects. Expanding and inspecting tag library files. Selecting custom and third-party tag libraries to drag and drop onto the JBoss Tools Palette.	*Chapter 8, Web Projects View*
OpenOn	Easy navigation between views and other parts of your projects.	*Section 3.1.1, "OpenOn"*
Content Assist	Code completion proposals while working with HTML, Java, JavaScript , XML, JSP, XHTML, seam project and JSF configuration files. Content assist based on project data (dynamic code assist). Code completion for values from property files, beans attributes and methods, navigation rule outcomes and JSF variables.	*Section 3.1.2, "Content Assist"*
Drag-and-Drop	Possibility of inserting any tag onto the page you are editing by just drag-and-droping it from the palette to this page. Adding any properties, managed bean attributes, navigation rules, tag library file declarations, `JSP` files from web projects view by clicking them and dragging to source code.	*Section 3.2, "Visual Page Editor"* *Section 8.2, "Drag and Drop"*
RichFaces Support	Tight integration between JBoss Developer Studio and *RichFaces* [http://www.jboss.org/jbossrichfaces] frameworks. Easy managing RichFaces components in any web application. Support for RichFaces and Ajax4jsf libraries in JBoss Tools Palette. Rendering RichFaces components in Visual Page Editor.	*Chapter 7, RichFaces Support*

Feature	Benefit	Chapter
Flexible Configuration	Various features of JBoss Developer Studio can be easily configured via the Preferences screen.	*Chapter 9, JBoss Tools Preferences*

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/index.html] website in the corresponding release directory.

The latest documentation is available as *nightly builds* [http://download.jboss.org/jbosstools/nightly-docs/].

Spring Tools

JBoss Developer Studio is bundled with the *Spring IDE* [http://springide.org/project] for Eclipse. Visit Spring IDE site for the latest versions and documentation.

2.1. *Spring IDE guide* [http://springide.org/project/wiki/SpringideGuide]

The *Spring IDE* [http://springide.org/project] is a graphical user interface for the configuration files used by the *Spring Framework* [http://www.springframework.org/]. It is built as a set of plugins for the Eclipse platform.

2.1.1. *Add Spring Project Nature* [http://springide.org/project/wiki/SpringideGuide#AddProjectNature]

2.1.2. *Create New Spring Project* [http://springide.org/project/wiki/SpringideGuide#CreateNewProject]

2.1.3. *Add References To Other Spring Projects* [http://springide.org/project/wiki/SpringideGuide#AddProjectReferences]

2.1.4. *Add Spring Beans Config Files* [http://springide.org/project/wiki/SpringideGuide#AddBeansConfigs]

2.1.5. *Create Spring Beans Config Sets* [http://springide.org/project/wiki/SpringideGuide#CreateBeansConfigSets]

2.1.6. *Open Spring Explorer* [http://springide.org/project/wiki/SpringideGuide#OpenSpringExplorer]

2.1.7. *Validate Spring Beans Config* [http://springide.org/project/wiki/SpringideGuide#ValidateBeansConfig]

2.1.8. *Open Spring Beans Graph* [http://springide.org/project/wiki/SpringideGuide#OpenBeansGraph]

2.1.9. *Search Spring Beans* [http://springide.org/project/wiki/SpringideGuide#SearchBeans]

Editors

In the JSF Tools Reference Guide you may have read about the Graphical Editors for JSF configuration files and Tiles Files. All these editors have OpenOn (see *Section 3.1.1, "OpenOn"*) and Code Assist (see *Section 3.1.2, "Content Assist"*) features, which are described in more detail in this document. In addition, this document will cover the Visual Page Editor (see *Section 3.2, "Visual Page Editor"*), which provides combined visual and source editing of Web pages, as well as a number of additional editors (see *Section 3.3, "More Editors"*) for different types of files.

3.1. Editors Features

JBoss Developer Studio has powerful editing features that help you easily navigate within your application and make use of content and code assist no matter what type of project file (`JSP`, `XHTML`, `XML`, `CSS` etc.) you are working on.

The mentioned features are the following:

- *Section 3.1.1, "OpenOn"*
- *Section 3.1.2, "Content Assist"*
- *Section 3.1.3, "Synchronized Source and Visual Editing"*

3.1.1. OpenOn

OpenOn lets you easily link directly from one resource to another in your project without using the **Package Explorer** view (i.e. the project tree). With OpenOn, you can simply use **F3** or **Ctrl**+**Click** on a reference to another file and the file will be opened.

OpenOn is available for:

- *Section 3.1.1.1, "XML Files"*
- *Section 3.1.1.2, "JSP/XHTML Pages"*
- Java files
- *Section 3.1.1.3, "CSS Classes"*
- *Section 3.1.1.4, "OpenOn for EL variables"*

3.1.1.1. XML Files

When editing an XML file press and hold down the **Ctrl** key. As you move the mouse cursor over different file references in the file, they are displayed with an underline. In this state these file

references effectively become links, and when they are clicked the appropriate file will be opened in its own editor.

Use the OpenOn functionality for the next entries defined in `XML` file:

1. Managed beans

   In this example source code of the managed bean `User` will be open.

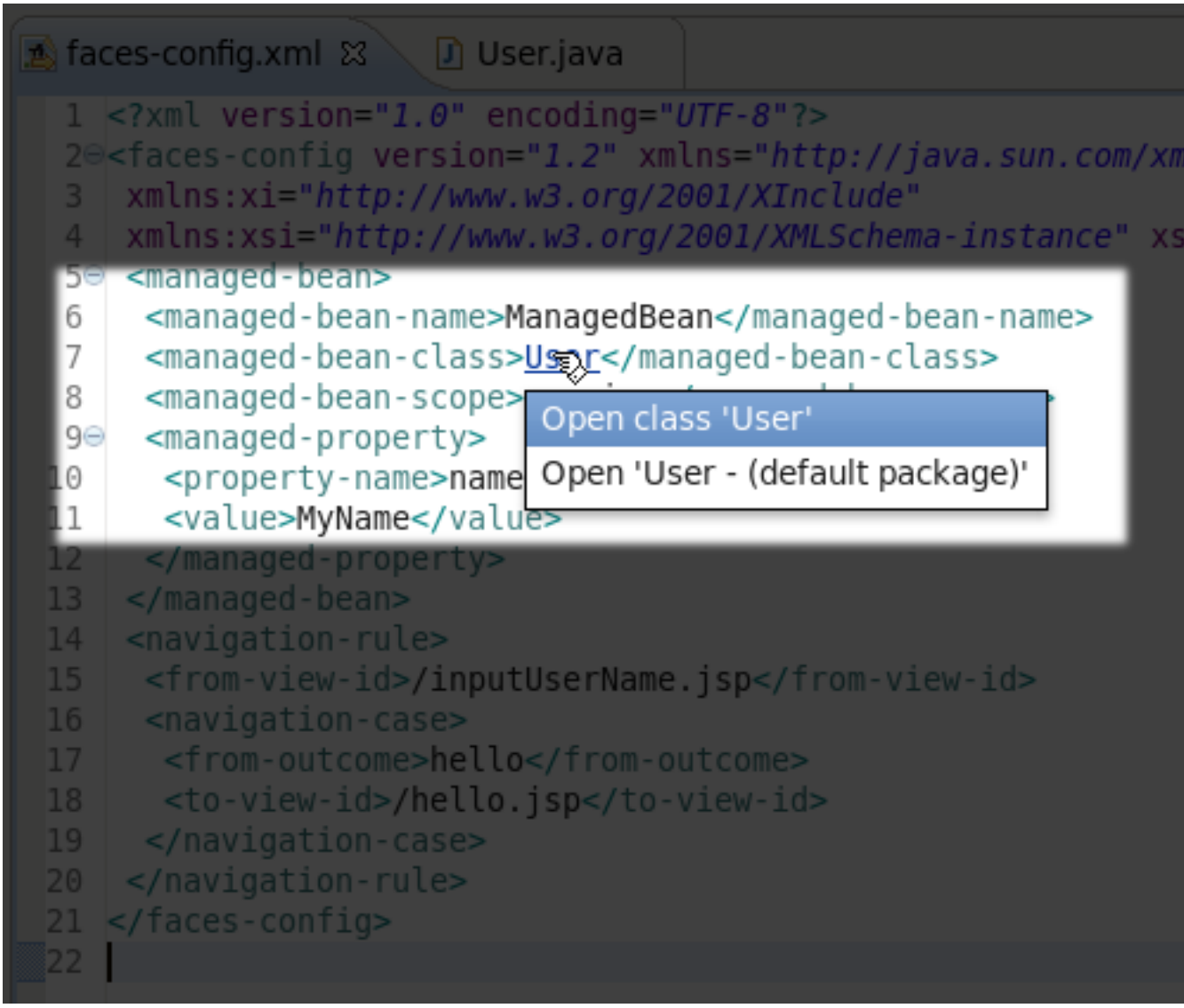

Figure 3.1. Opening a Managed Bean

The image below shows the result of using OpenOn.

faces-config.xml | User.java

```
1
2  public class User
3  {
4      private String name;
5
6      public String getName()
7      {
8          return name;
9      }
10
11     public void setName(String value)
12     {
13         name = value;
14     }
15
16     public User()
17     {
18
19     }
20
21 }
22
```

Figure 3.2. Opened Managed Bean

2. Beans properties

Figure 3.3. OpenOn for the Bean Property

3. JSP file references

Figure 3.4. OpenOn for JSP Page

3.1.1.2. JSP/XHTML Pages

OpenOn is also available in JSP and XHTML pages edited in the Visual Page Editor. It will allow you to quickly jump to the reference instead of having to hunt around in the project structure.

You can use OpenOn for the following JSP or XHTML file entries:

1. Imported property files.

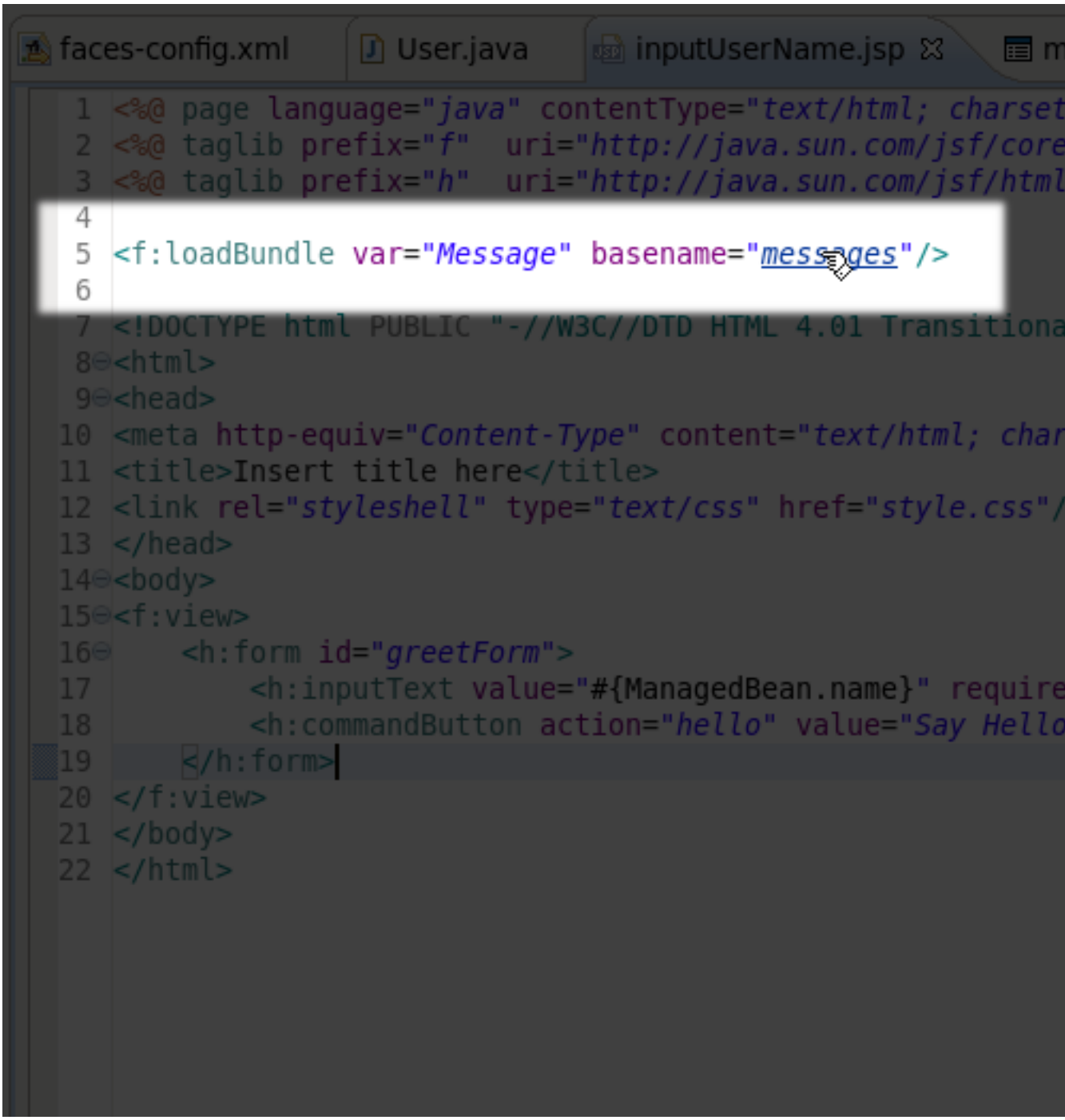

Figure 3.5. OpenOn for Property File Imported to the JSP Page

2. CSS files used in a JSP or XHTML page.

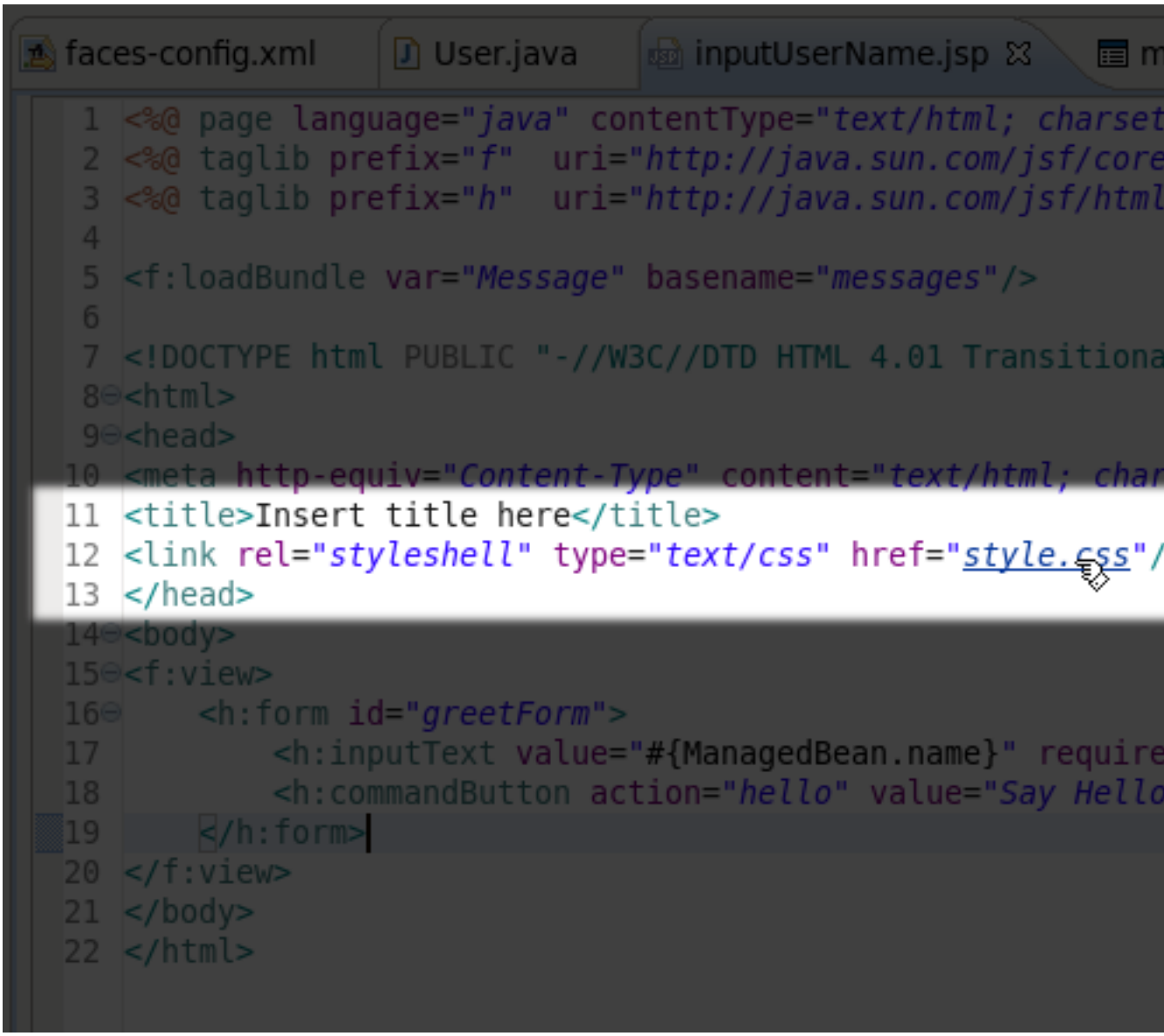

Figure 3.6. OpenOn With CSS File

3. Managed beans and their properties.

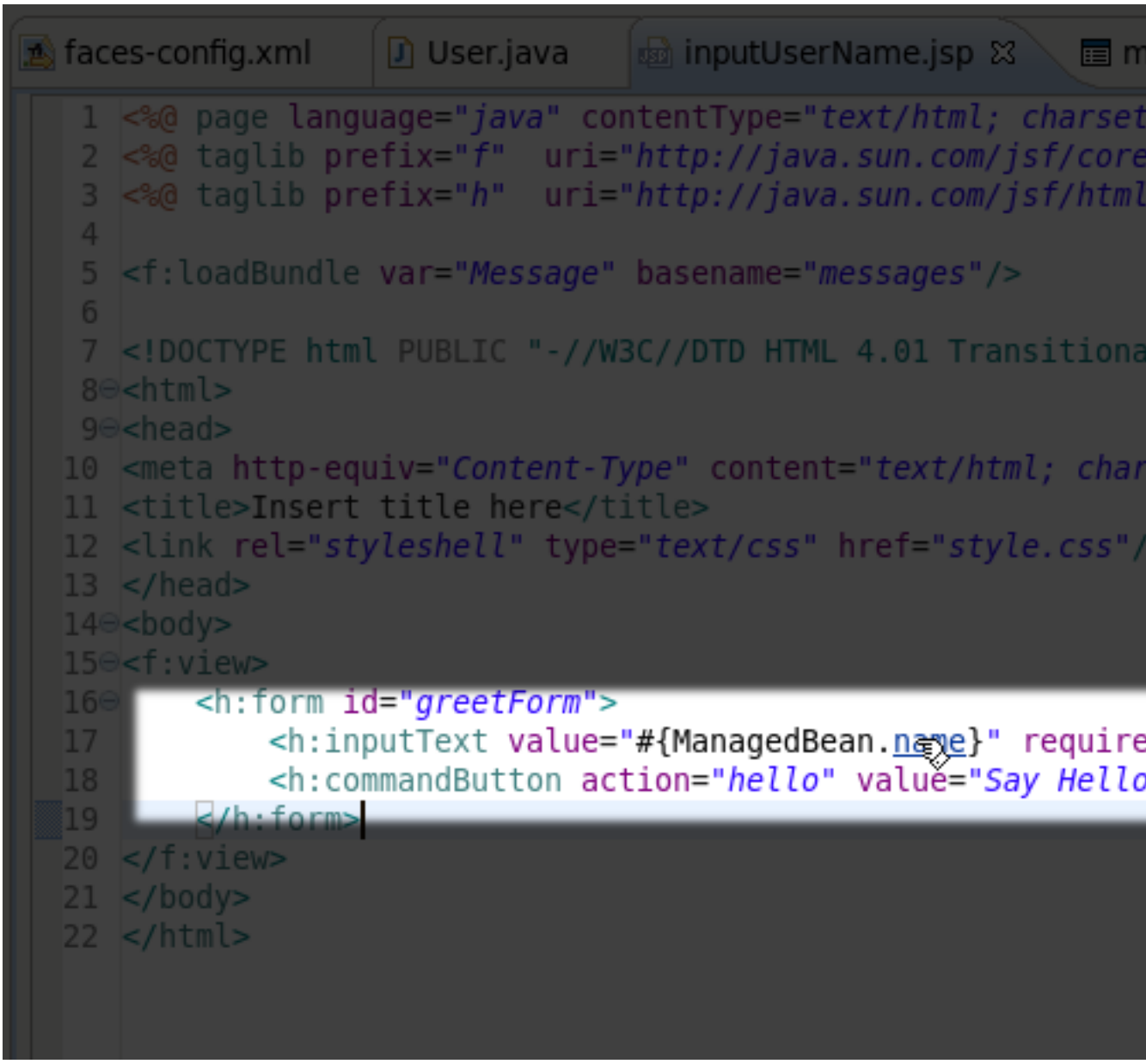

Figure 3.7. OpenOn With Managed Beans

4. Navigation rules in JSP files.

   For JSP files in a JSF project, you can easily open the navigation rules by applying OpenOn to the JSF tag for the navigation outcome:

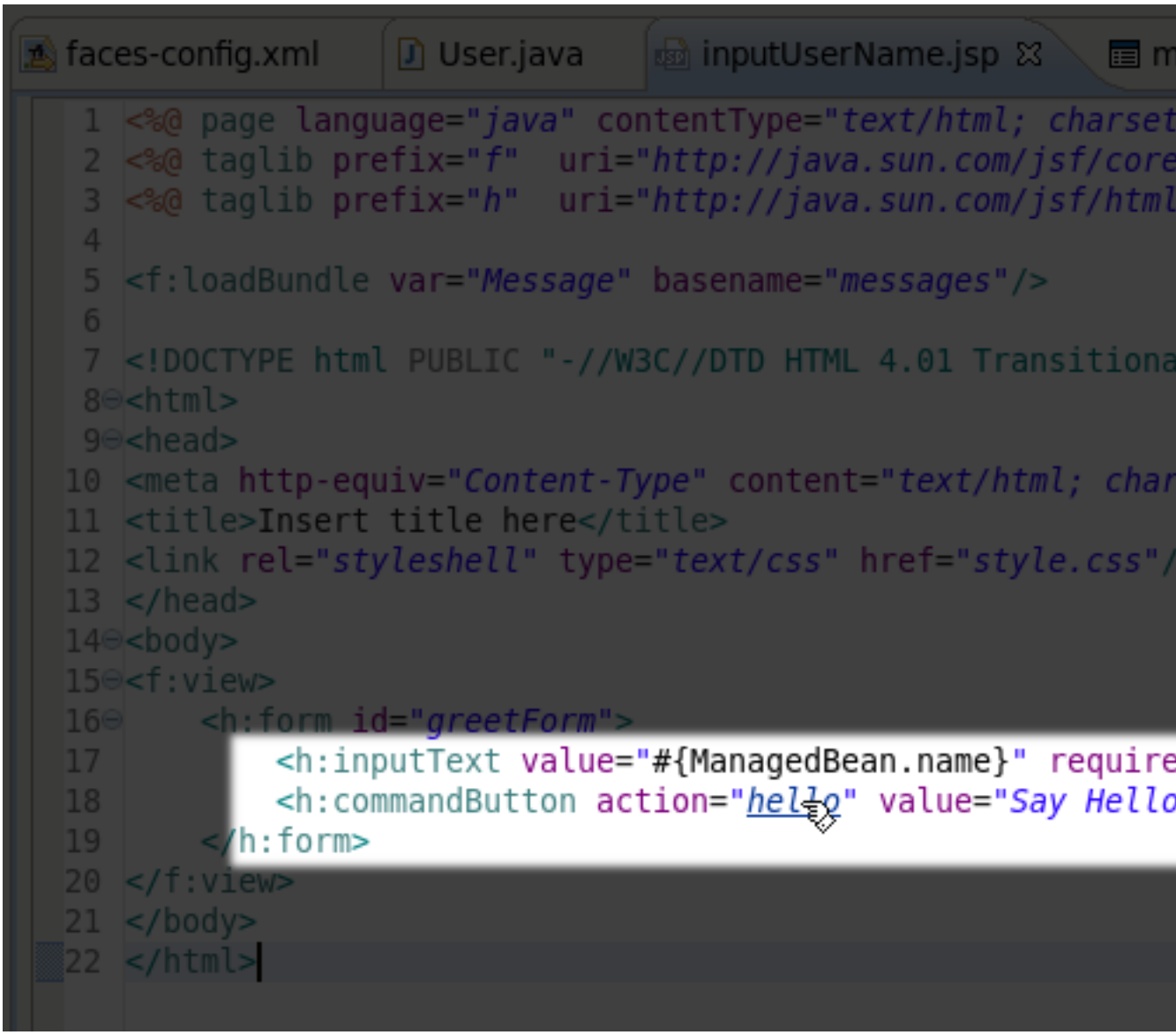

Figure 3.8. OpenOn with JSF Tag

5. Custom Facelets tag libraries in `XHTML` pages.

   For details, see *Section 3.2.8.2, "OpenOn for Custom Facelets Components"* later in this guide.

6. Custom JSF 2.0 components.

Figure 3.9. OpenOn with JSF 2.0 Component

3.1.1.3. CSS Classes

You can quickly navigate through CSS classes using OpenOn.

Figure 3.10. OpenOn With CSS Class

OpenOn is also implemented for CSS classes added by a complex link.

Figure 3.11. OpenOn With CSS Class added by a complex link

3.1.1.4. OpenOn for EL variables

OpenOn can be used for paths to files set with EL variable.

Figure 3.12. OpenOn for paths to files set with EL variable

3.1.2. Content Assist

Content assist is available when working with:

- *Seam project files* [../../seam/html_single/index.html#ContentAssist]
- *Section 3.1.2.1, “JSF Project Files”*
- *Section 3.1.2.2, “JSP Pages”*
- *Section 3.1.2.3, “Content Assist for XHTML Pages”*
- *Section 3.1.2.4, “Content Assist for Java Files”*
- *Section 7.1, “Code Assist for RichFaces”*
- *ESB XML files* [../../esb_ref_guide/html_single/index.html#ESBContentAssist]
- *Section 3.1.2.5, “Content Assist for Insert Tag Wizard”*

Notice that code completion for EL variables have icons illustrating what they are from. These icons are described in the table below.

Table 3.1. Content assist icons

Icon	Type	Context
	Enumeration	Used to show items which exist in the predefined set of equivalent proposals.
	Seam Proposal	Used to show Seam Context variables, its properties and methods.
	JSF EL	Used to show Managed Beans, Managed Bean Properties, Managed Bean Methods, Constants, Resource Bundles, Resource Bundle Properties.
	JSF Action	Used to show navigation rules defined in the `faces-config.xml`.
	Message Bundle	Used to show Messages Resources items.

Icon	Type	Context
	Resource path	Used to show paths which are accessible from the cursor place.

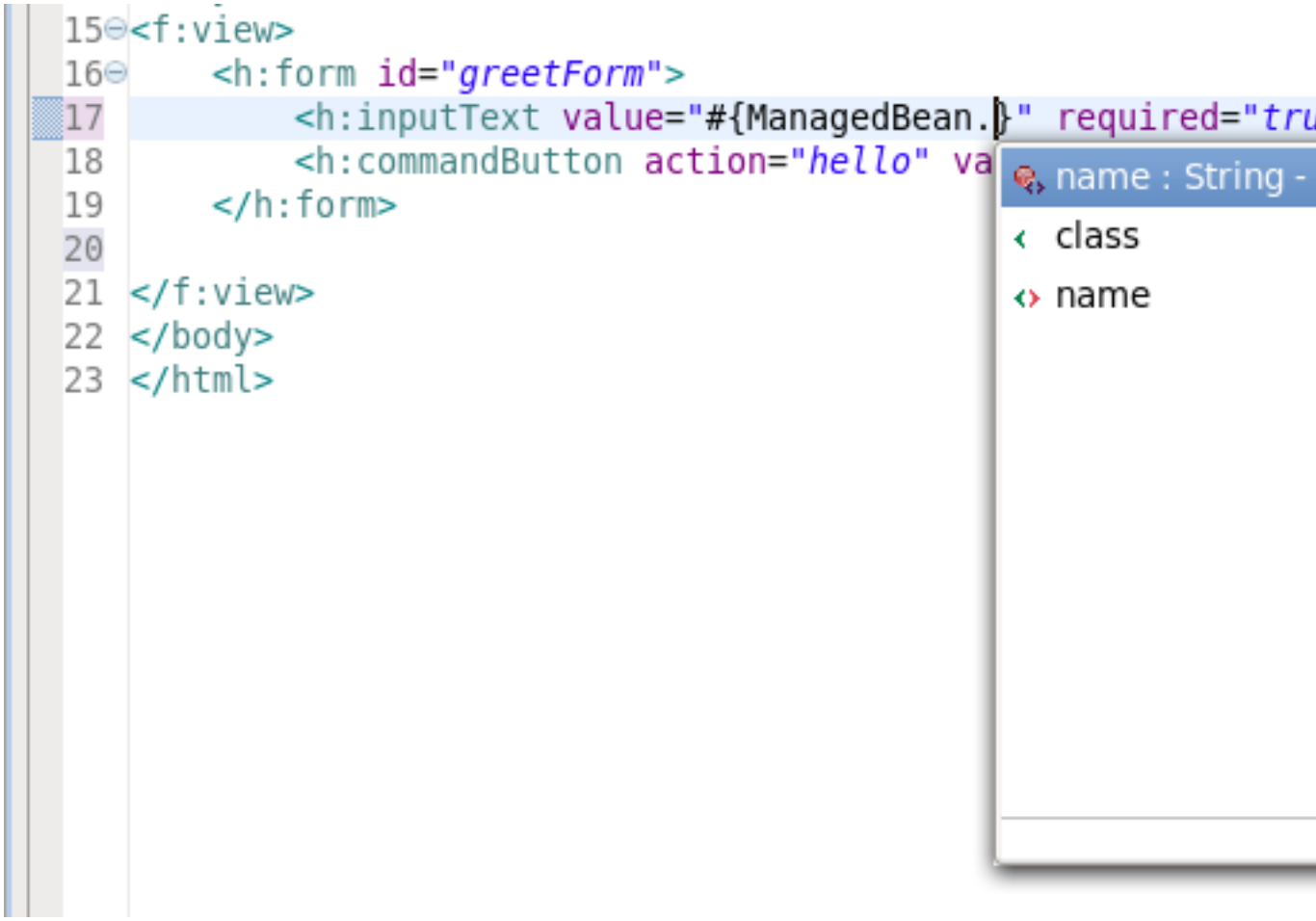

Figure 3.13. JSF Content Assist

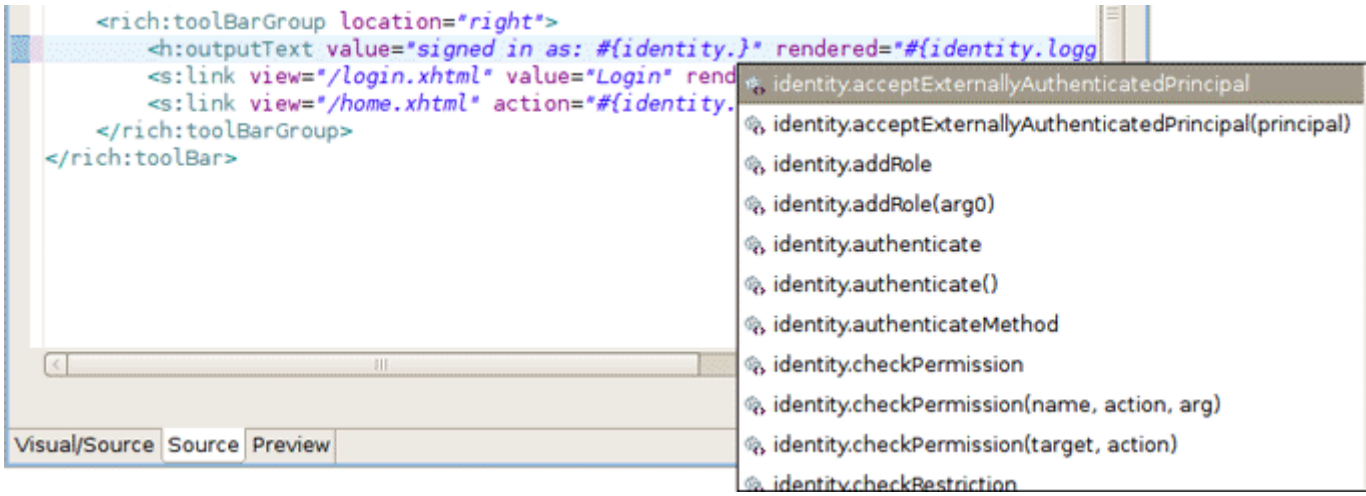

Figure 3.14. Seam Content Assist

The ranking and sorting are available in EL code completions.

As you can see, in addition to proposals, content assist also provides descriptions of selected tags or attributes.

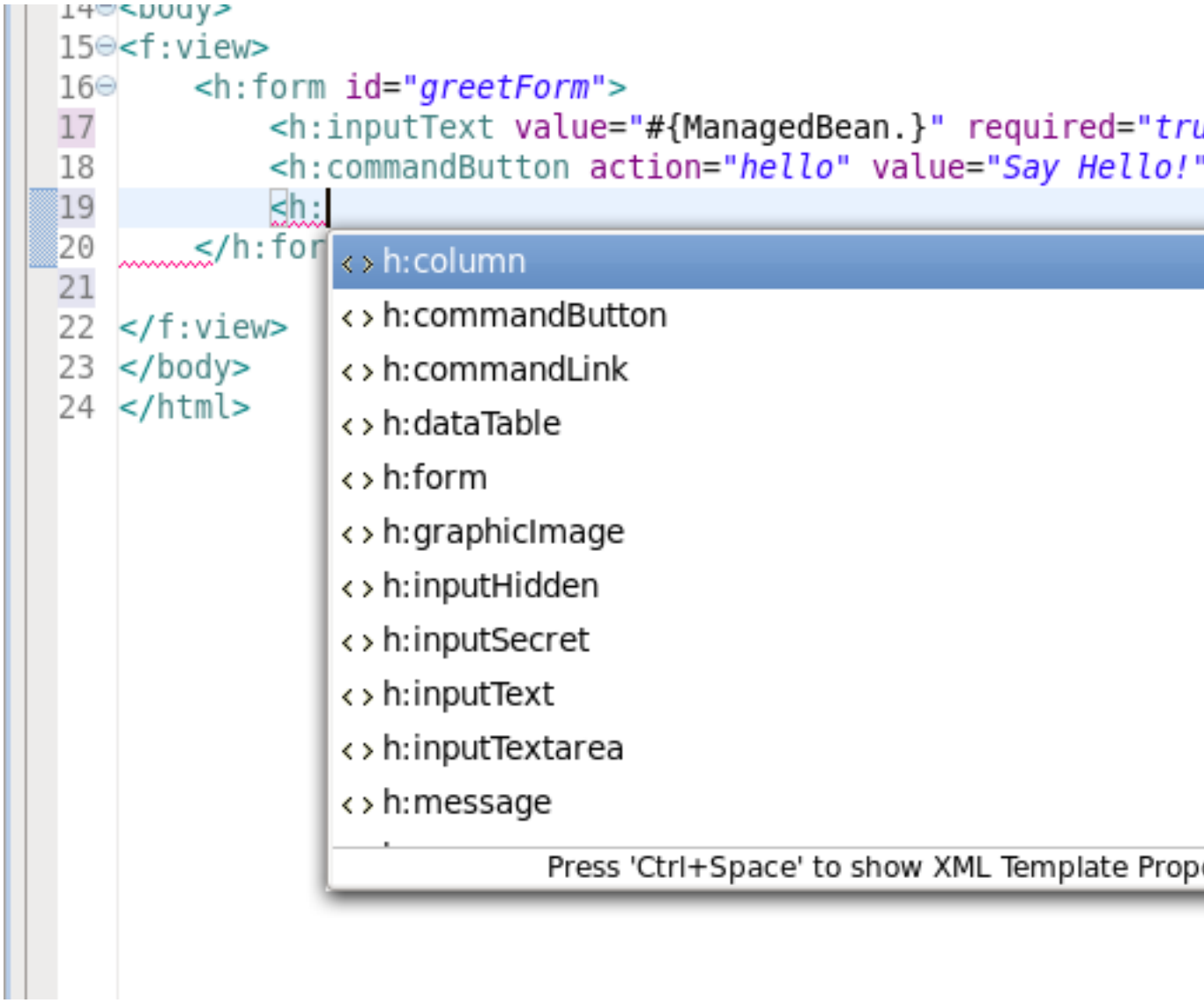

Figure 3.15. Tag description

3.1.2.1. JSF Project Files

When working with a JSF project in JBoss Developer Studio, you can use various Content Assist features while developing:

- Content Assist for `XML`, `XHTML`, `JSP` and `JSF` configuration files
- Content Assist for Composite Components
- Content Assist based on project data
- Content Assist with graphical `JSF` editor

3.1.2.1.1. Content Assist for XML, JSP and JSF configuration files

Content Assist is available to help you at any point when working with any XML, JSP and JSF configuration files. Simply press **Ctrl**+**Space** to see what options are available.

Content Assist for JSF configuration file:

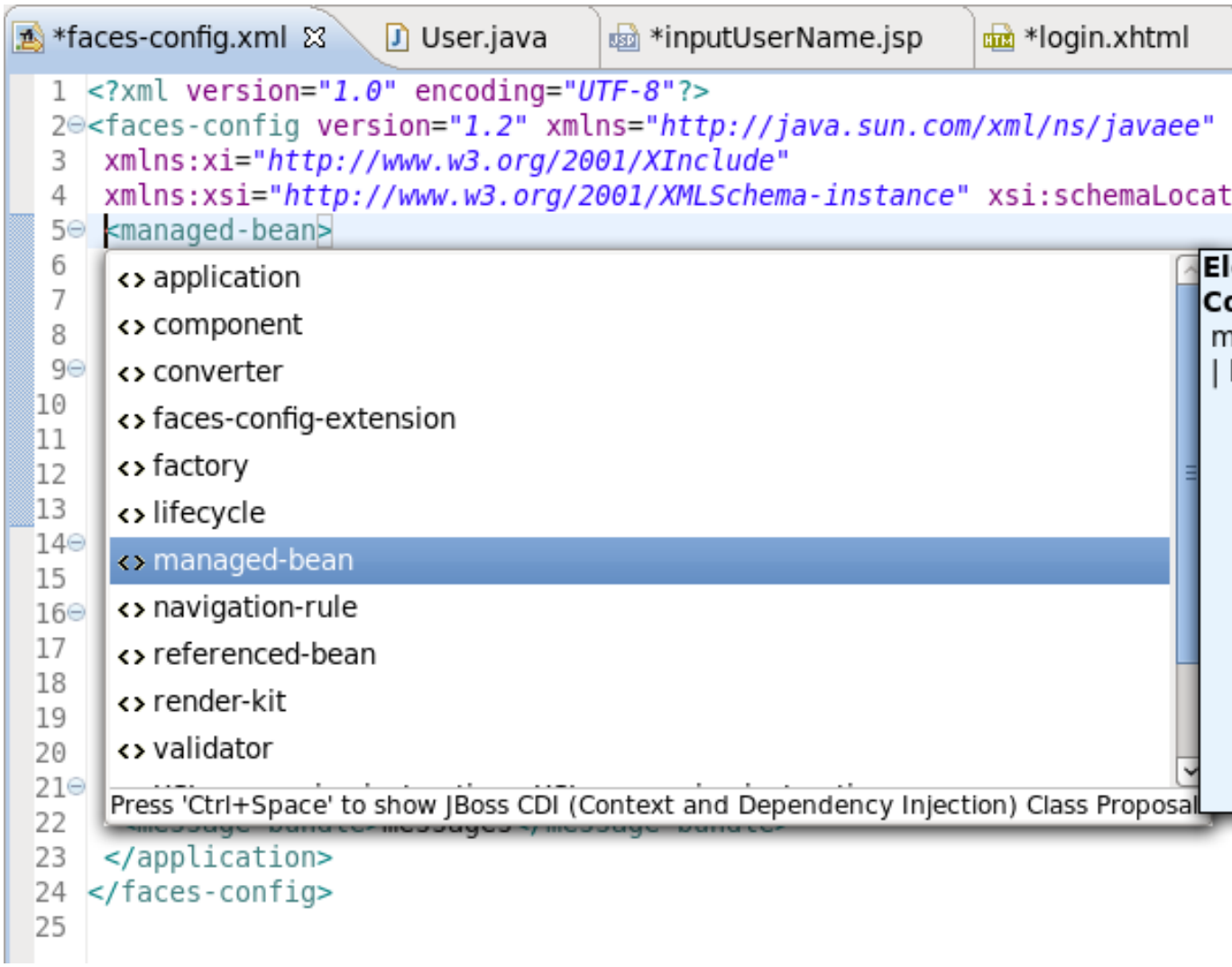

Figure 3.16. Content Assist in JSF Configuration File

Content Assist for JSF JSP file:

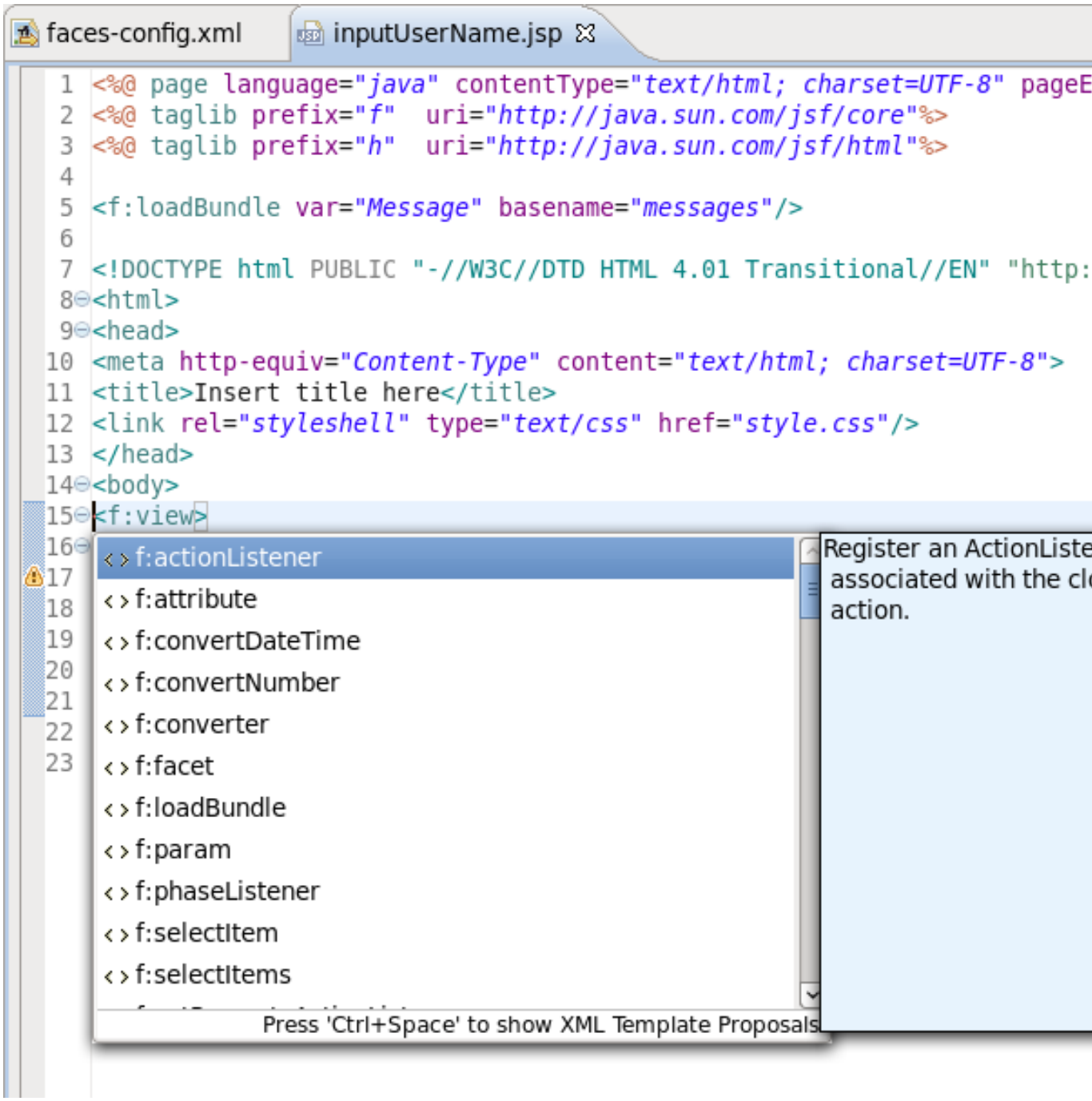

Figure 3.17. Content Assist in JSP File

Content Assist for other JSF XML project files (`web.xml` shown):

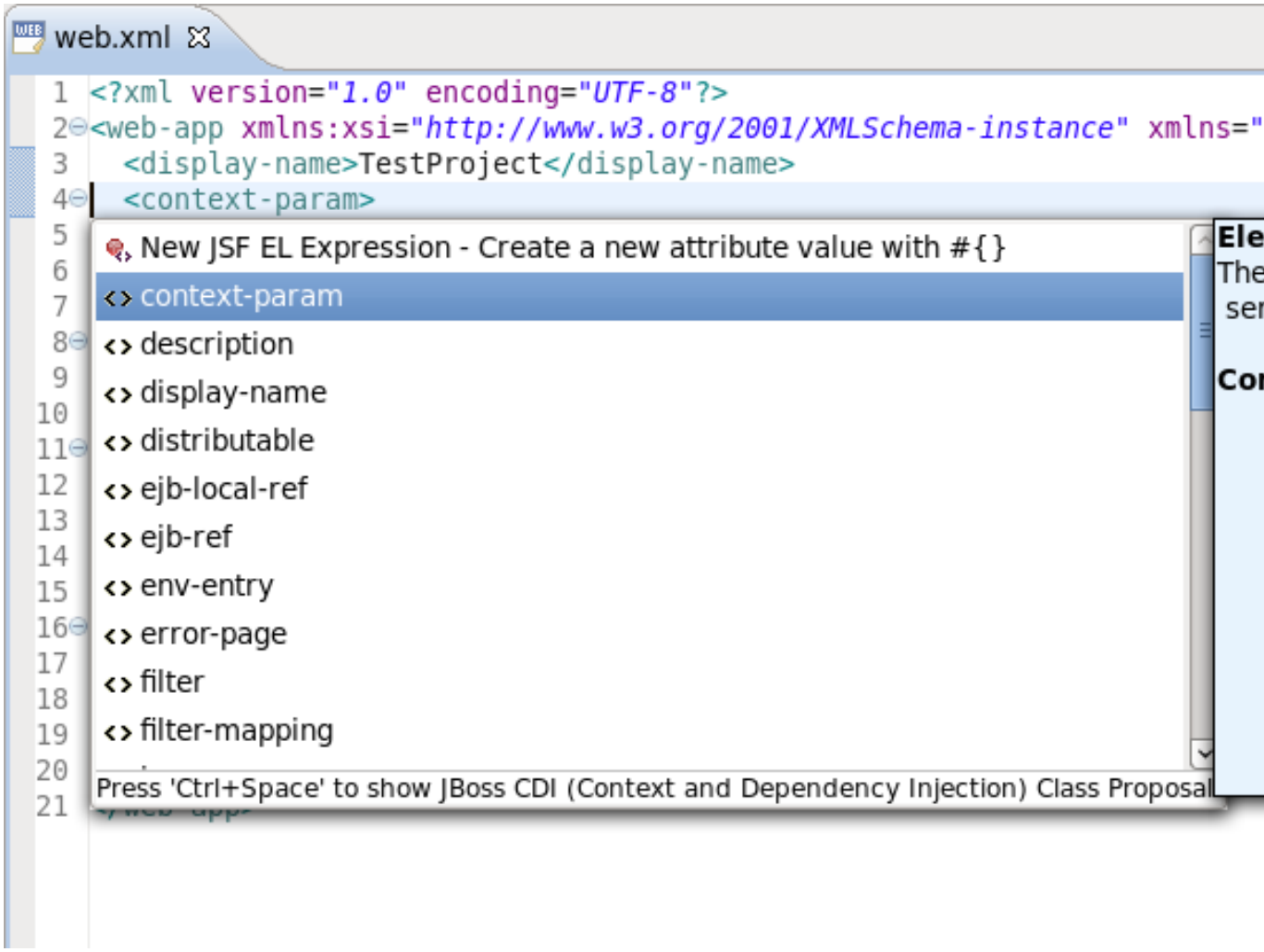

Figure 3.18. Content Assist in web.xml File

3.1.2.1.2. Content Assist for Composite Components

Content assist functionality is also available for composite components. The image below shows content assist used with a composite component file named `loginPanel.xhtml` within a JSF 1.2 project with facelets.

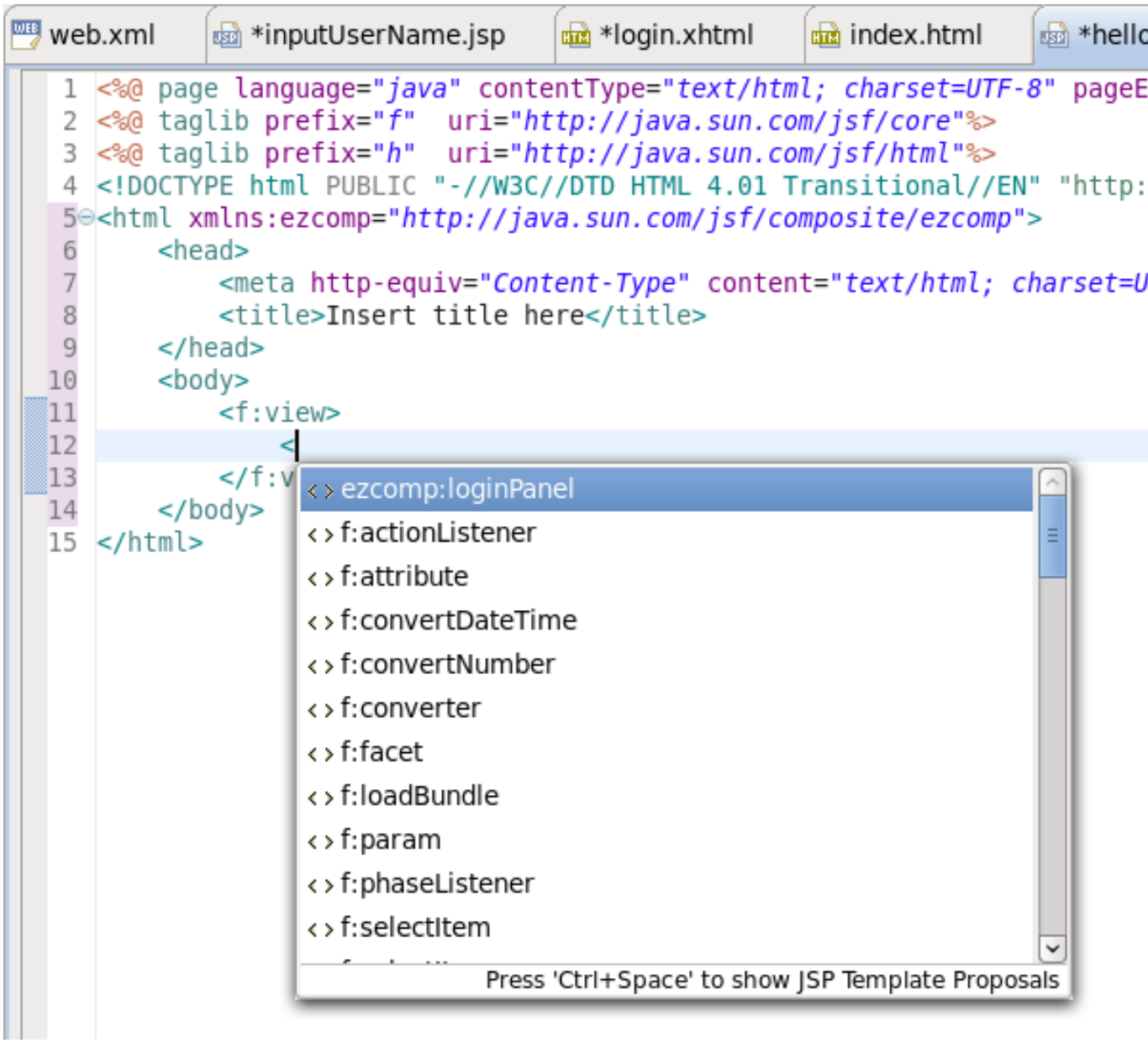

Figure 3.19. Content Assist for Composite Components

3.1.2.1.3. Content Assist Based on Project Data

JBoss Developer Studio takes Content Assist to the next level. JBoss Developer Studio will constantly scan your project, and you will be able to insert code into the JSP page from your project including:

- Values from Property files
- Managed beans attributes and methods
- Navigation Rule Outcomes

- JSF variables (context, request etc...)
- Resource Bundles from template page

The figure below demonstrates how to insert a message from a Properties file. You simply put the cursor inside the `value` attribute and press **Ctrl**+**Space**. JBoss Developer Studio will scan your project and display a list of possible values that can be inserted.

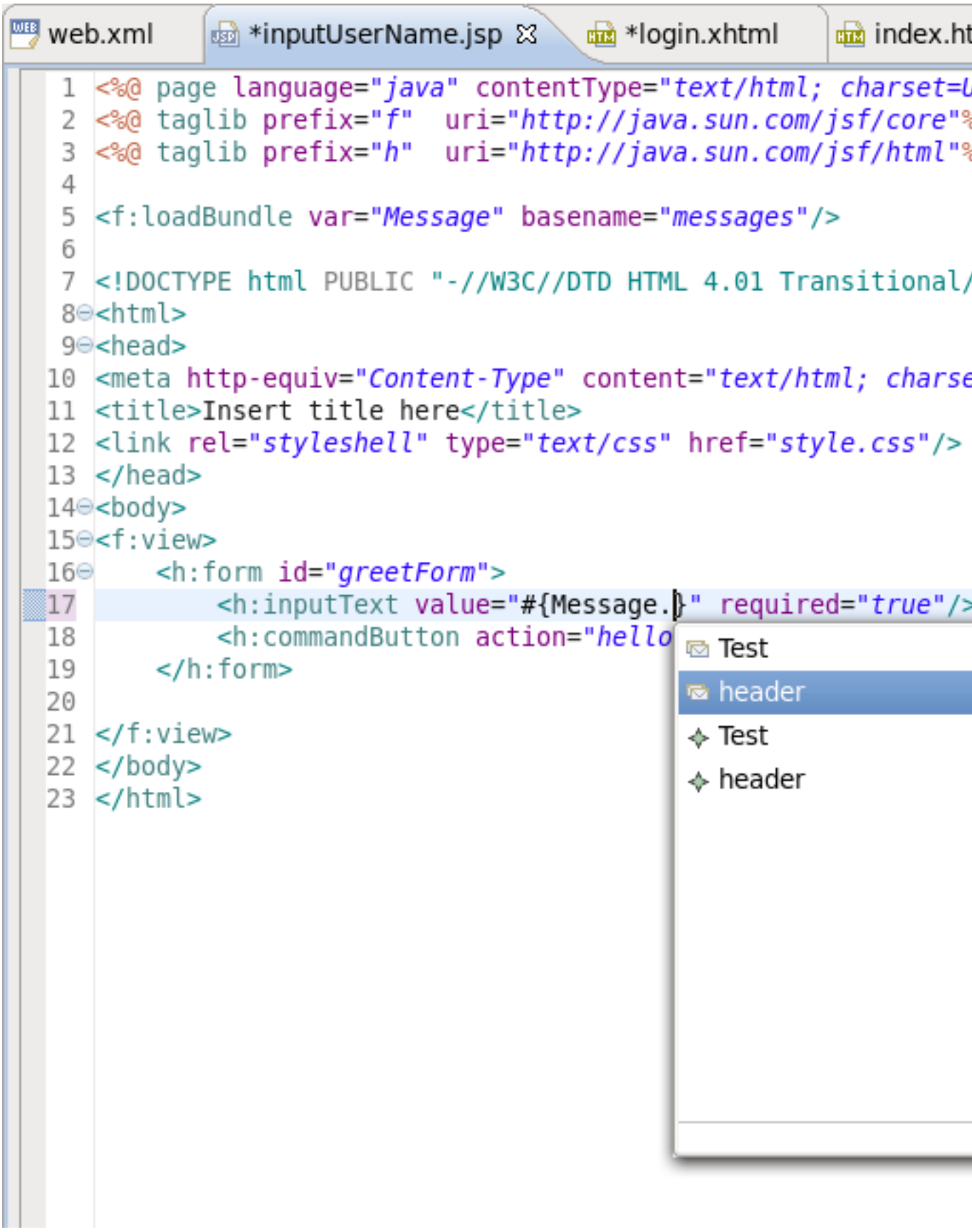

Figure 3.20. Inserting Message

In the following screenshot we are inserting a Managed bean attribute value. Again, by simply pressing **Ctrl**+**Space**, JBoss Developer Studio will show a list of all possible values that can be inserted.

Once you select a Managed bean, it will show you a list of all available attributes for the selected Managed bean.

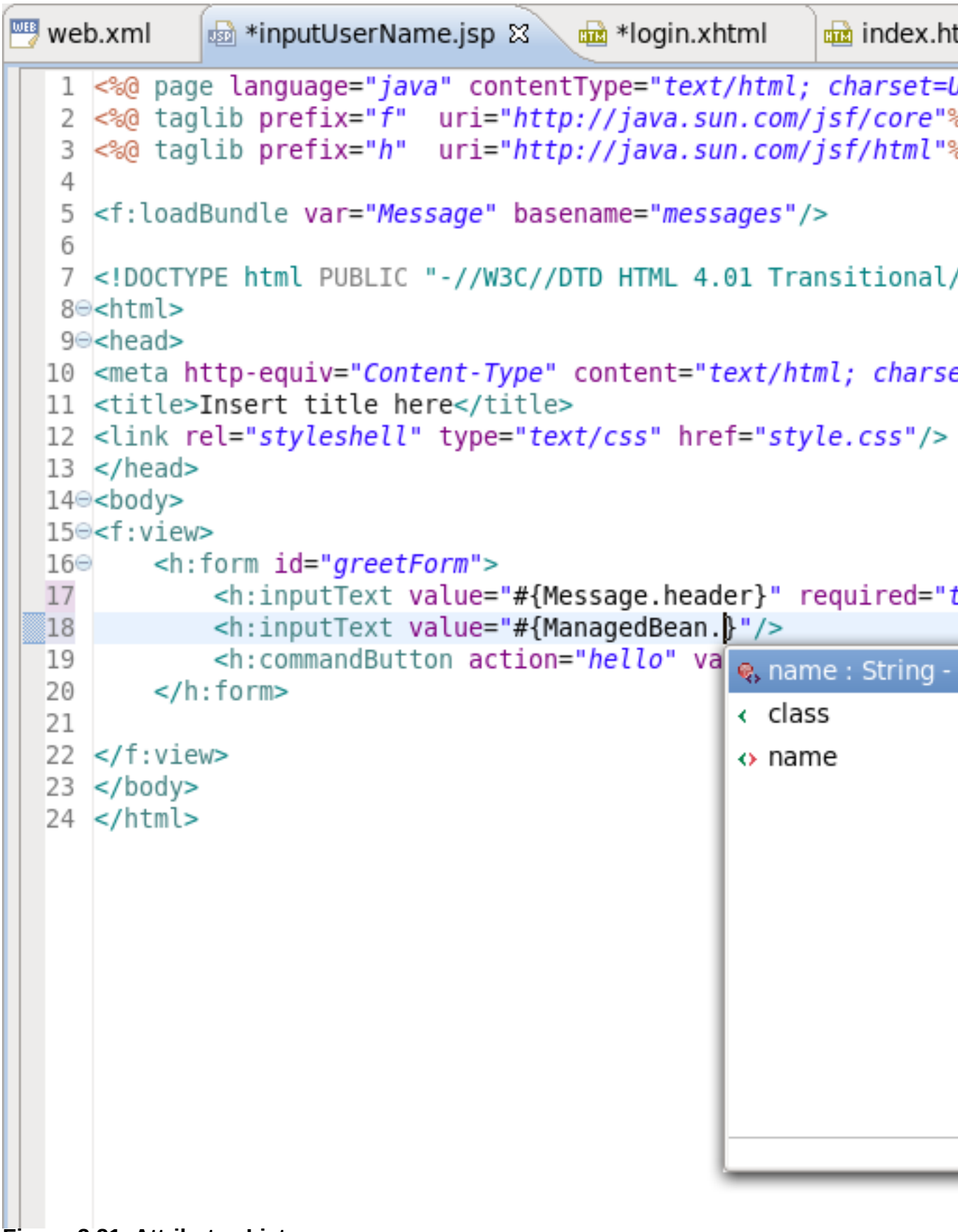

Figure 3.21. Attributes List

Code Assist based on project data will also prompt you for navigation rules that exist in your JSF configuration file.

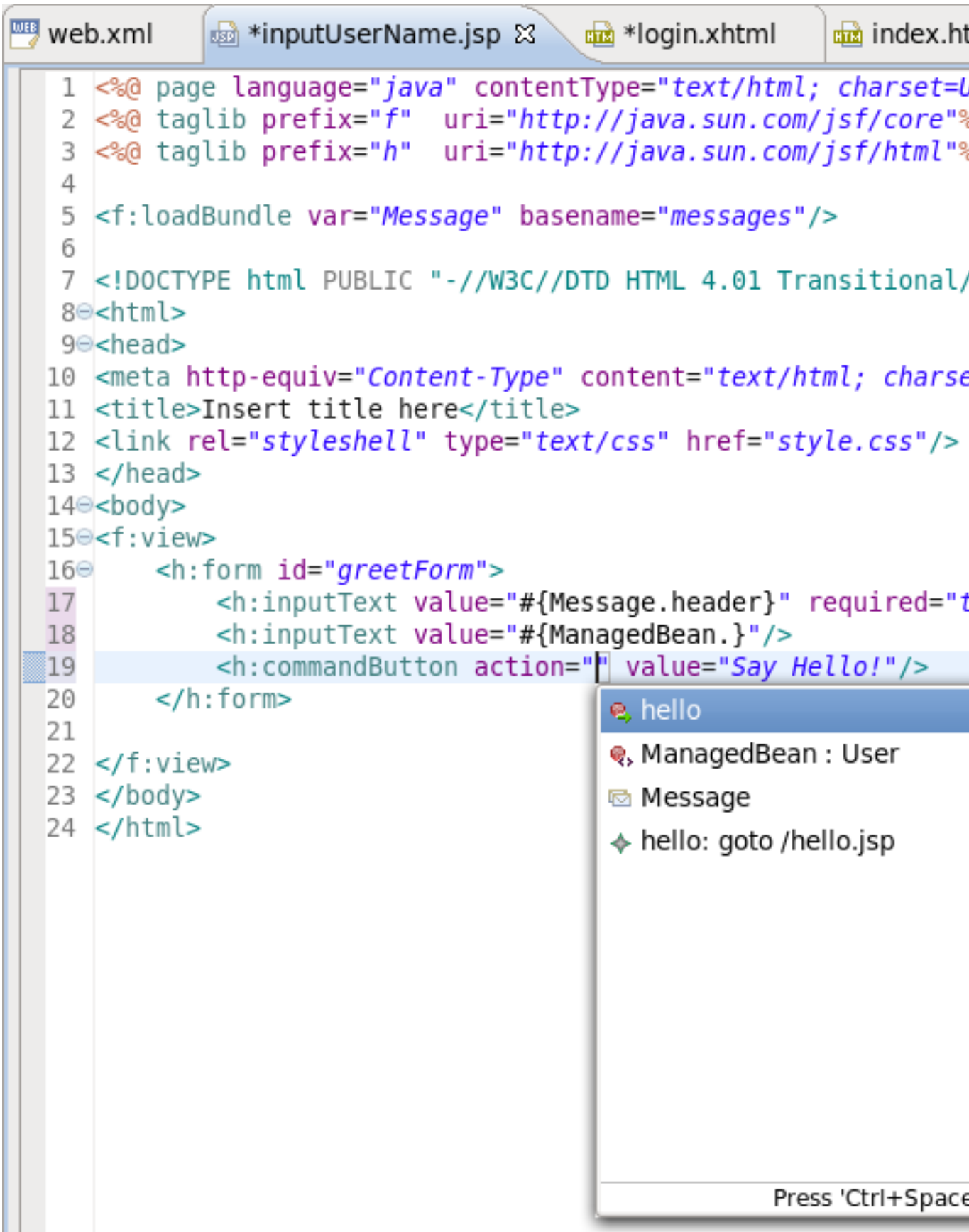

Figure 3.22. Code Assist

Code Assist can also provide you with access to the beans located in JAR archives.

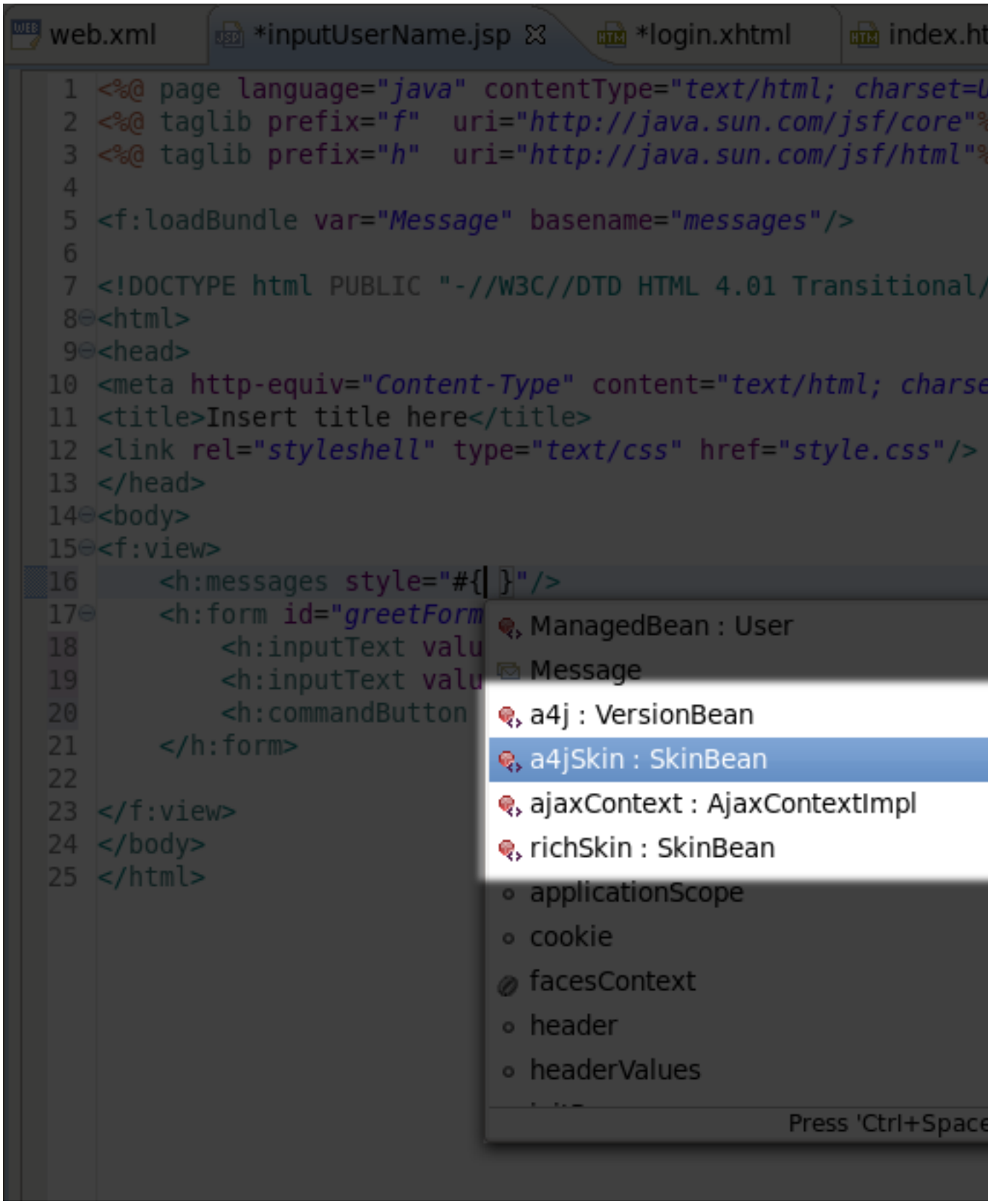

Figure 3.23. Code Assist: accessing beans in jar archives

Code Assist is able to define Resource Bundles on template pages and provides the proposals on the client page.

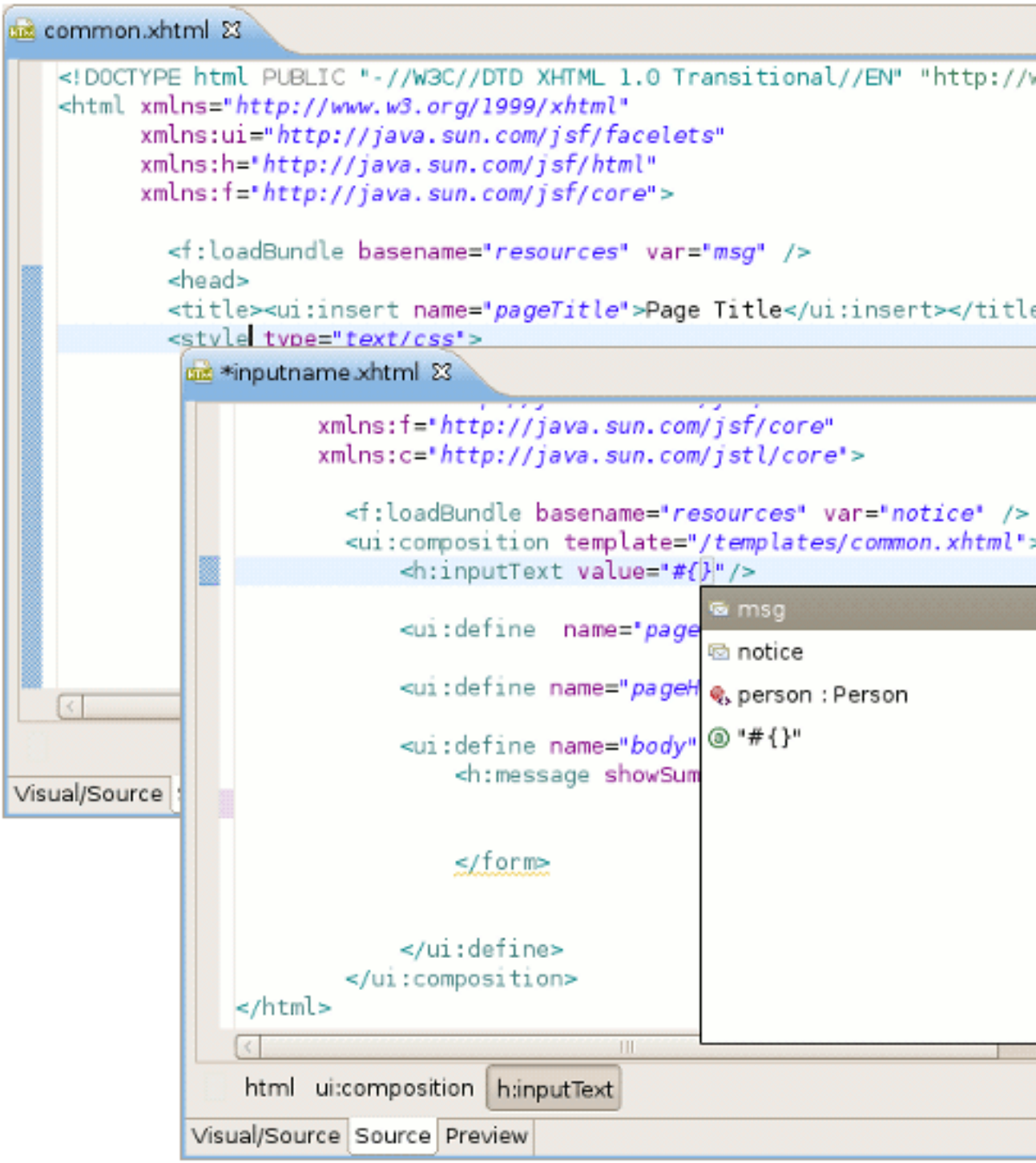

Figure 3.24. Code Assist: Message Bundles proposals from template page

3.1.2.1.4. Content Assist within Tree JSF Editor

JBoss Developer Studio also provides Content Assist when working within the **Tree** JSF configuration editor. Just press **Ctrl**+**Space**.

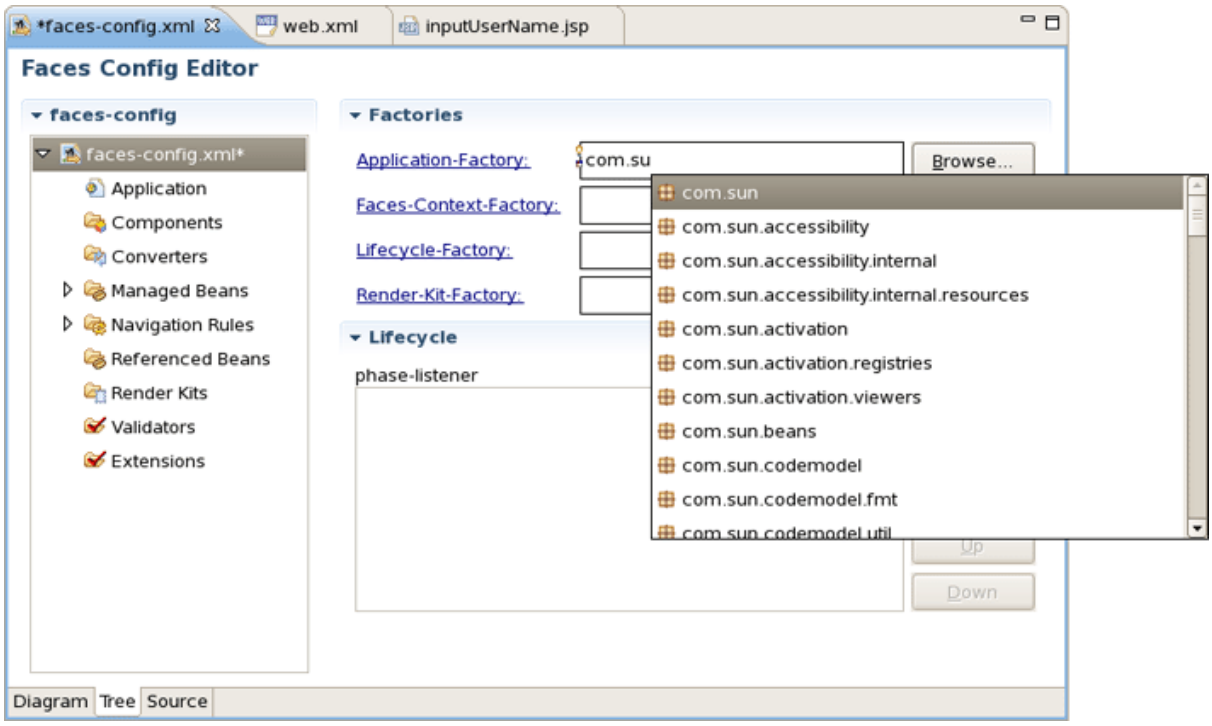

Figure 3.25. Content Assist in Tree JSF Configuration Editor

3.1.2.2. JSP Pages

3.1.2.2.1. Content Assist for JSF Tags

JBoss Developer Studio provides full code completion for JSF tags:

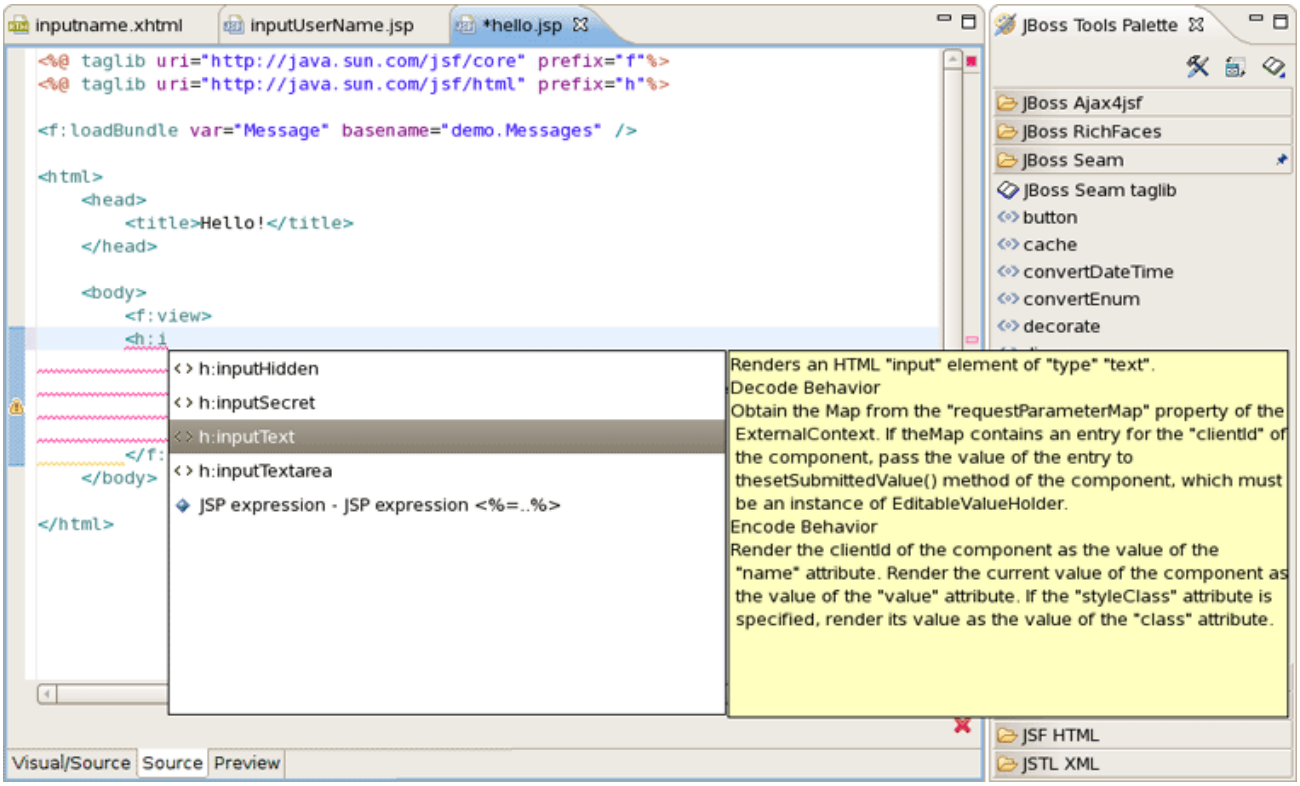

Figure 3.26. JSF Tags Content Assist

When the tag is selected, the required attributes, if there are any, are already inserted and the cursor is moved to the first attribute. At this point you can ask for attribute proposals.

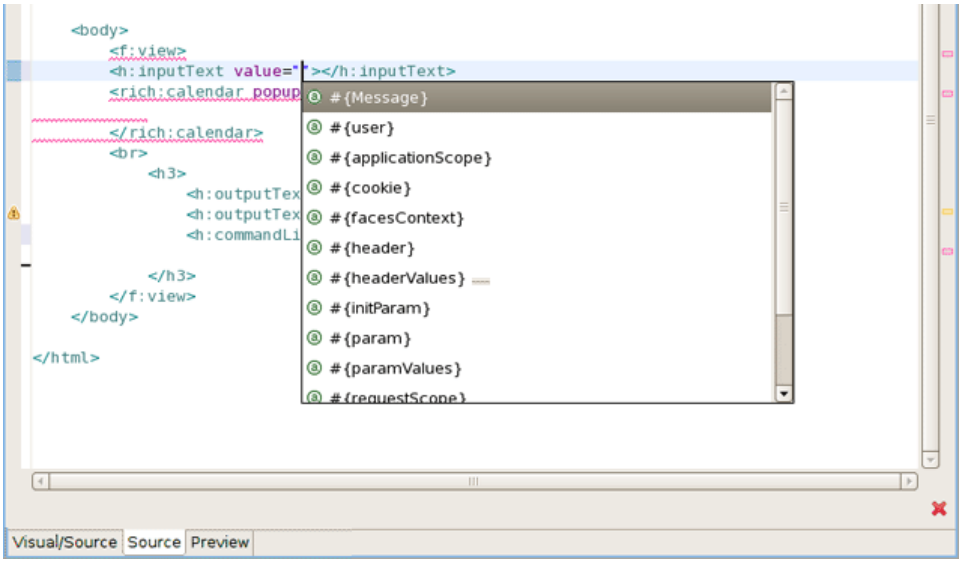

Figure 3.27. Attributes Content Assist

3.1.2.2.2. Content Assist for JSTL Tags

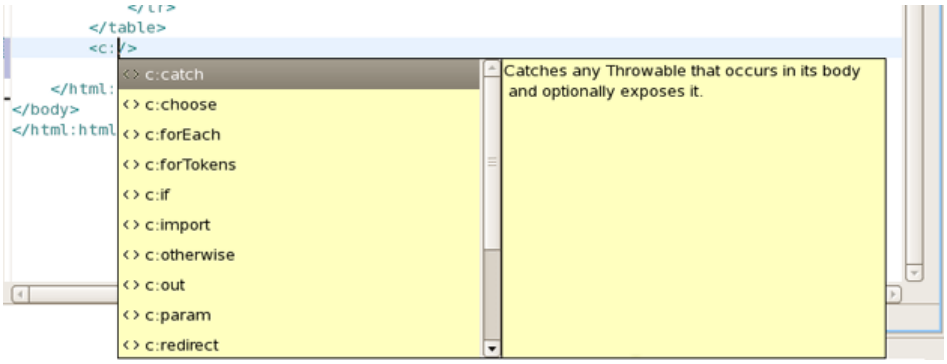

Figure 3.28. JSTL Tags Content Assist

3.1.2.2.3. Content Assist for HTML Tags

Content assist for HTML tags works in the same manner as the JSF tags:

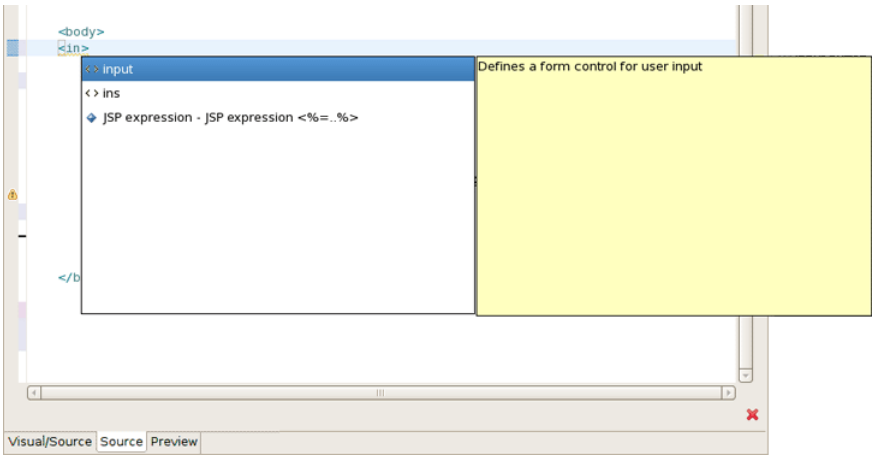

Figure 3.29. HTML Tags Content Assist

Content Assist can also be used for HTML tag attributes:

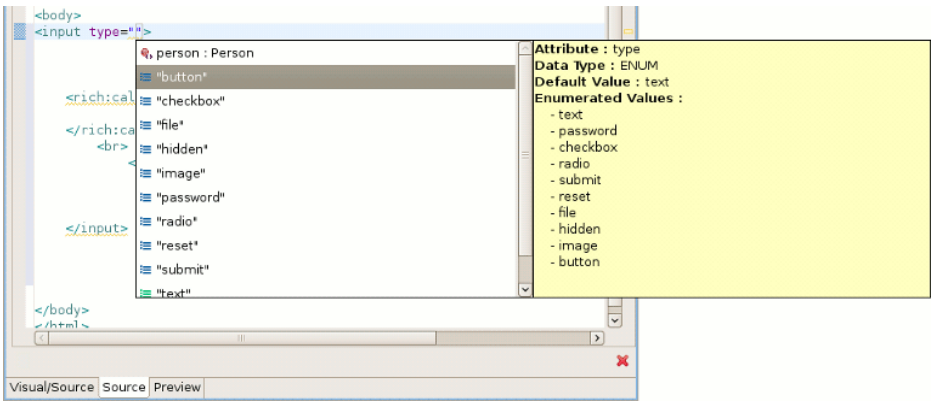

Figure 3.30. HTML Tags Content Assist

3.1.2.2.4. Content Assist for JavaScript Tags

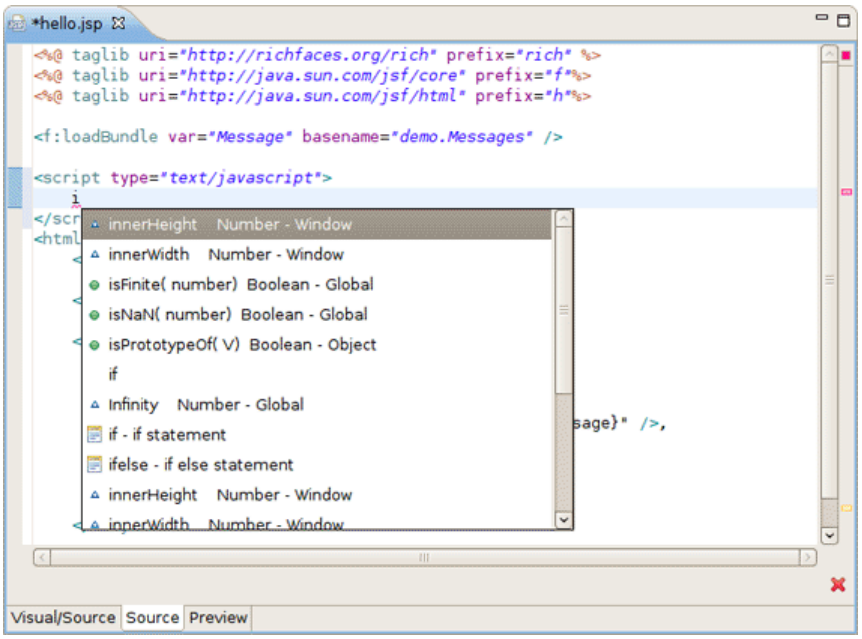

Figure 3.31. JavaScript Tags Content Assist

3.1.2.2.5. Content Assist for EL expressions

Content Assist also provides expression language (JSF EL) support. It is used in web application pages to access the JavaBeans components in the page bean and in other beans associated with the web application, such as the session bean and the application bean.

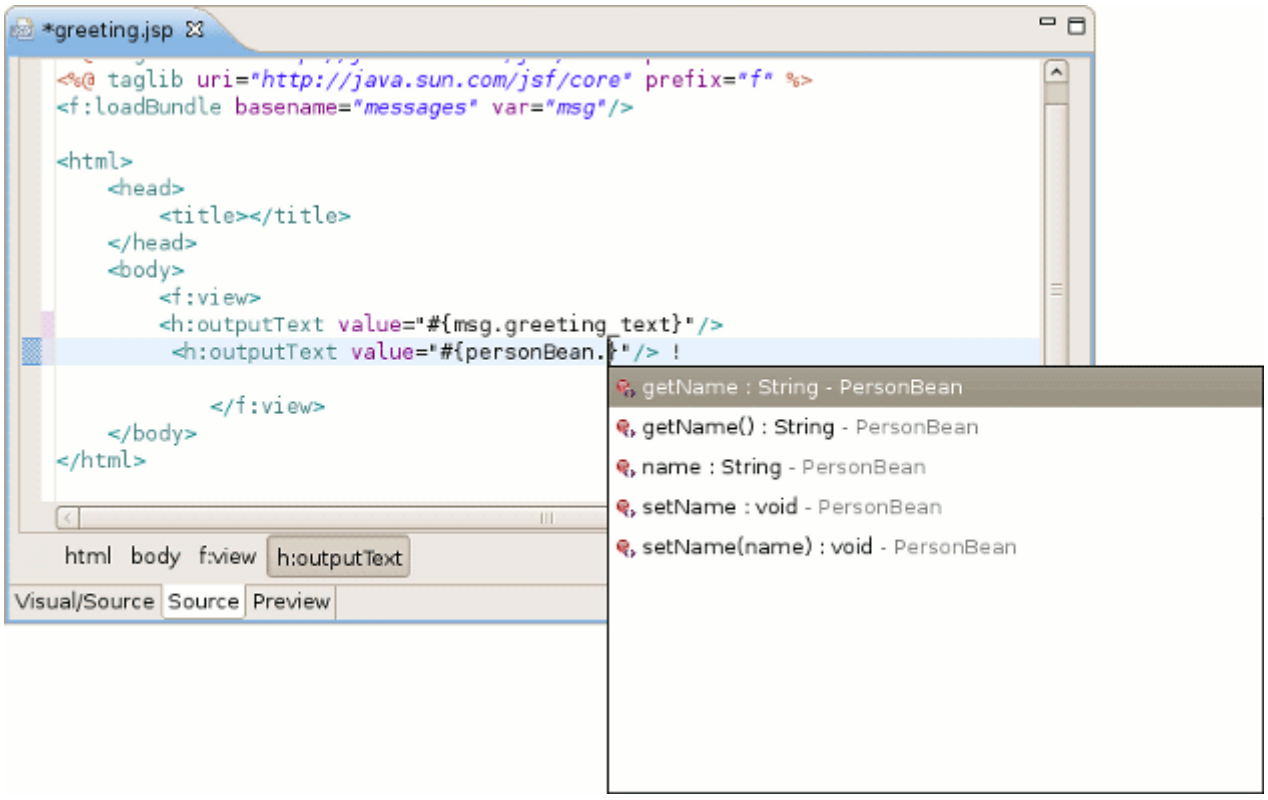

Figure 3.32. EL Content Assist

3.1.2.3. Content Assist for XHTML Pages

The code completion menu items for the Seam components in a Seam project shows the proposals marked with Seam icon.

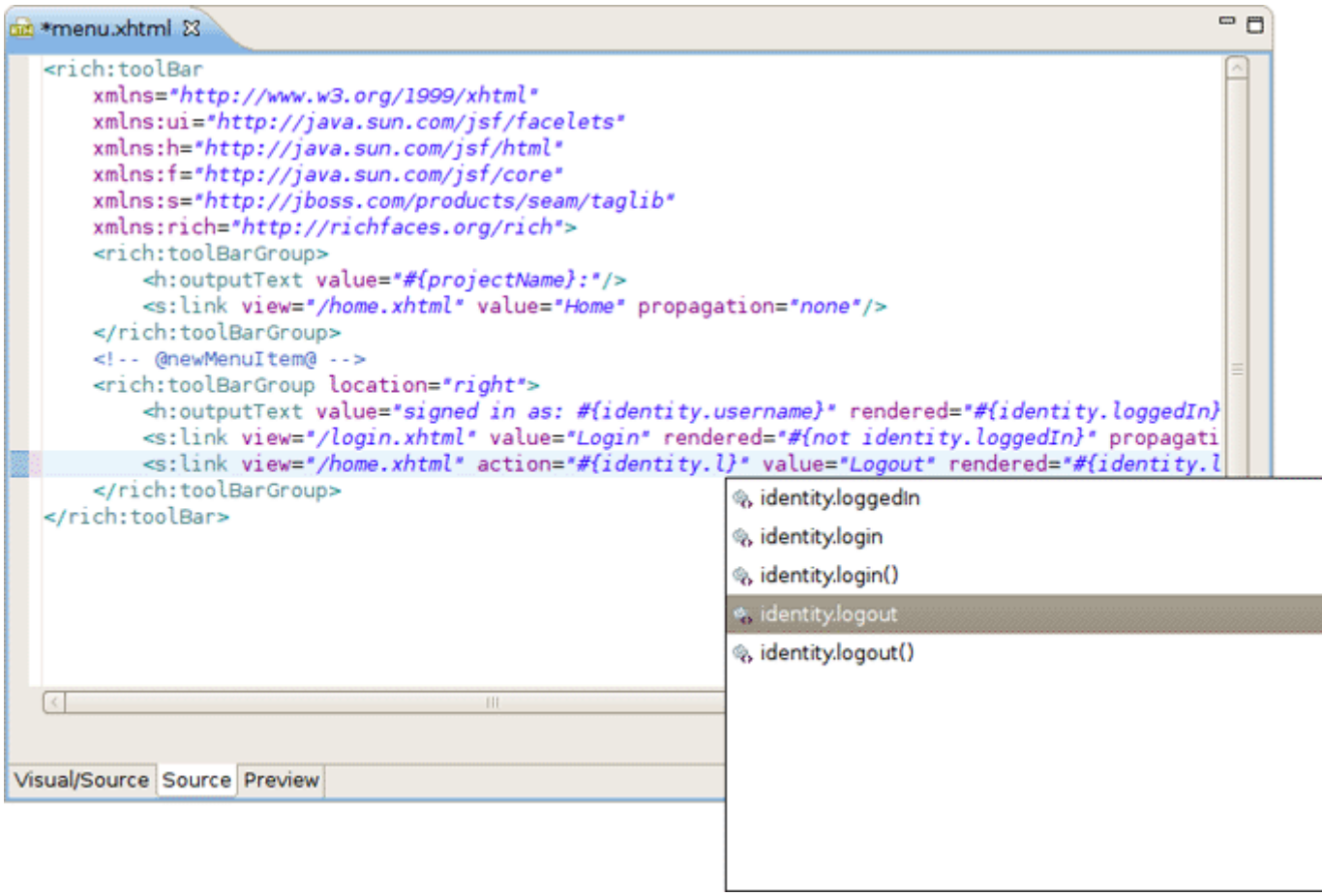

Figure 3.33. Content Assist for Seam Components in the XHTML Page

If an XHTML file uses custom Facelets components, the Content Assist should also be available for them. For details, see *Section 3.2.8.1, "Content Assist for Custom Facelets Components"* later in this guide.

3.1.2.4. Content Assist for Java Files

Various tools tips provide you additional information about Java elements (JavaDocs, source classes, return types, method names, parameters and etc.) when working with Java files.

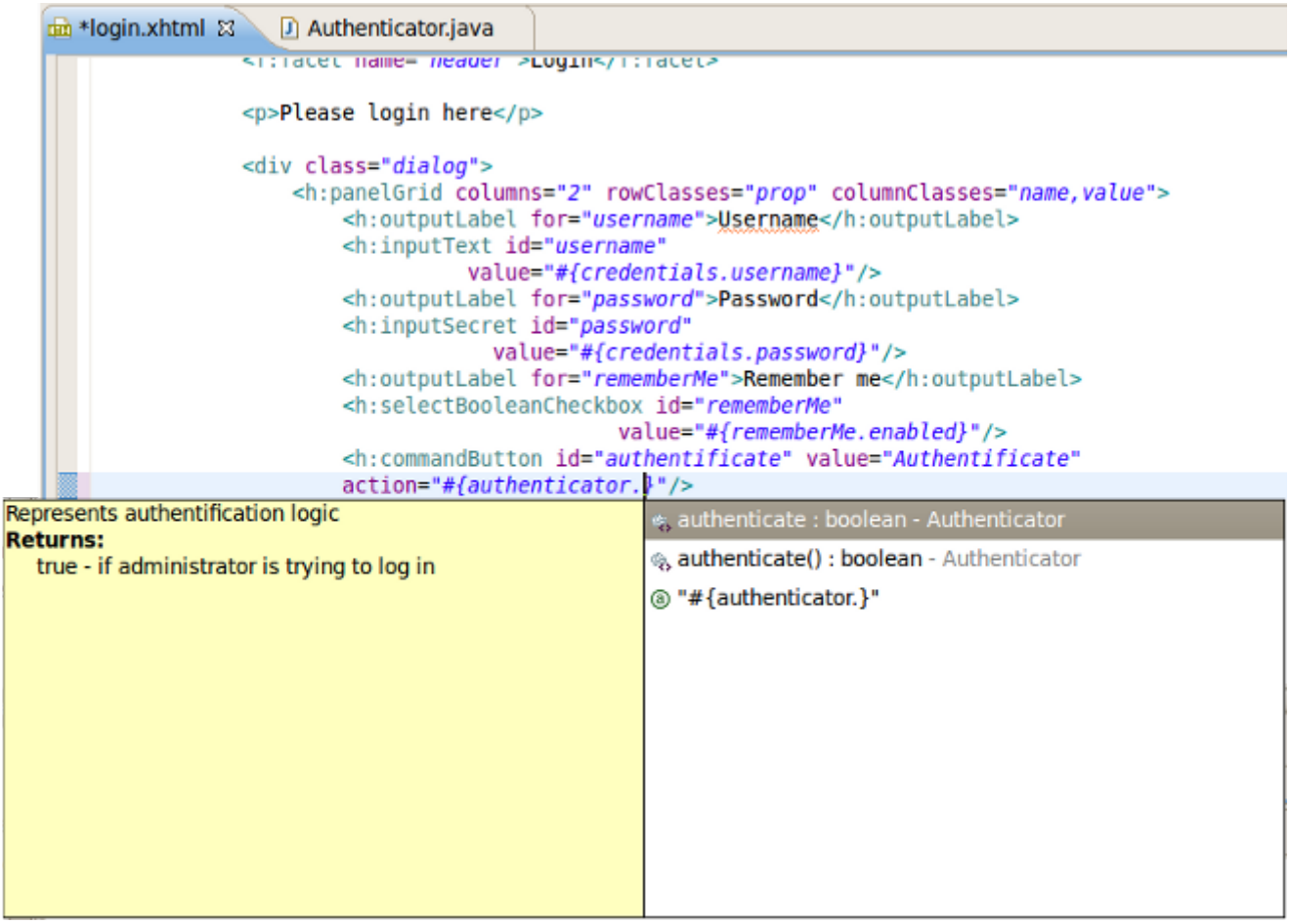

Figure 3.34. Content assist for JavaDoc

3.1.2.5. Content Assist for Insert Tag Wizard

Content Assist is also available for any attribute value in the **Insert Tag** wizard.

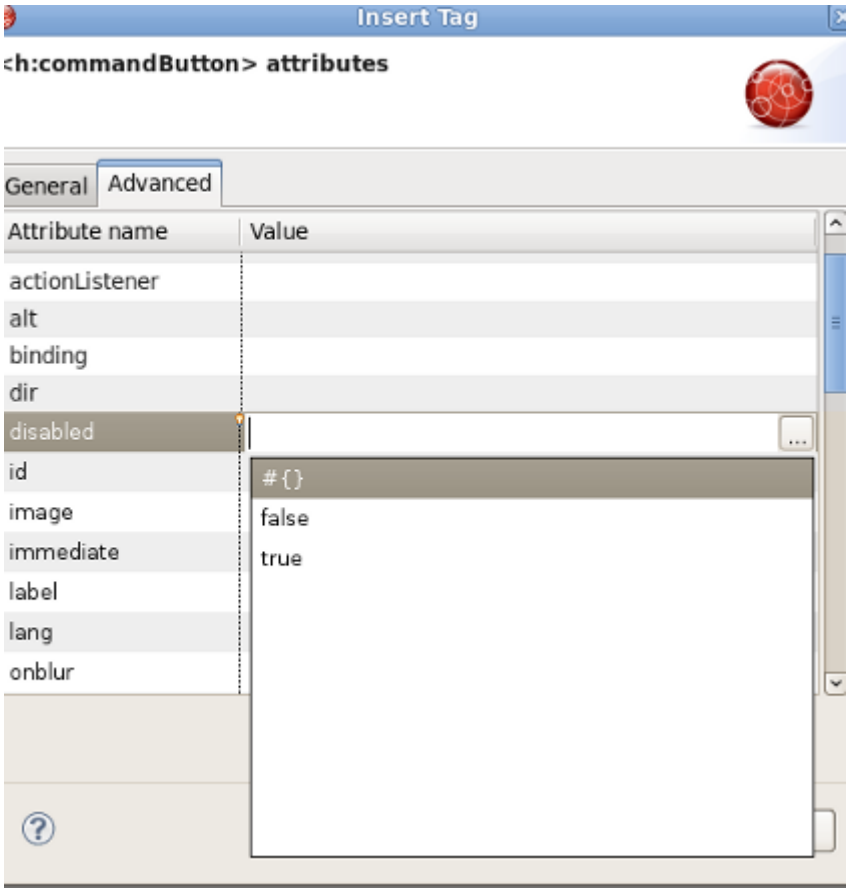

Figure 3.35. Content Assist for Insert Tag Wizard

3.1.2.6. Adding dynamic code assist to custom components that were added to JBoss Tools Palette

If you open projects that were created in older studio versions you may see the following message:

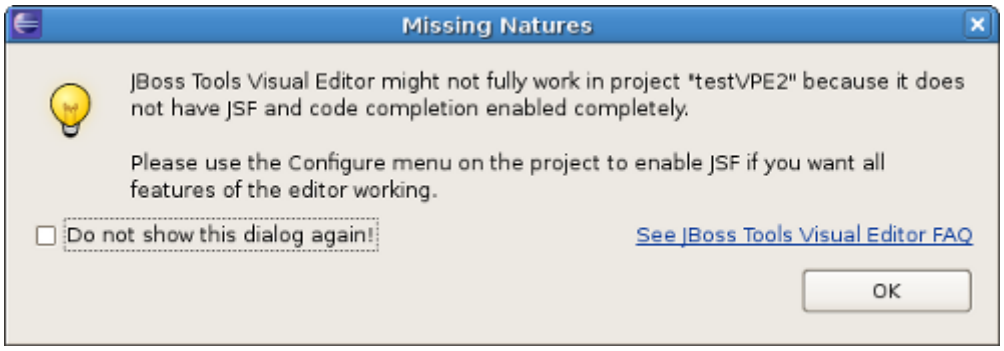

Figure 3.36. Missing Natures Message

It warns that some features of content assist may not work. Use the following steps to fix the problem and turn off the message box:

- Right click the project in the **Package Explorer** view
- Select **Configure** → **Add JSF Capabilities** from the context menu
- Configure your project using the **Add JSF Capabilities** wizard and click the **Finish** button

3.1.3. Synchronized Source and Visual Editing

JBoss Developer Studio offers the ability to edit the source code of a file, as well as providing visual editors for many file types. The source code and visual editors can be viewed and edited at the same time in a split screen view, and any changes you make in the source code editor will immediately appear in the visual editor.

The JSF configuration file editor has three views: **Diagram**, **Tree** and **Source**. All views are synchronized and you can edit the file in any view.

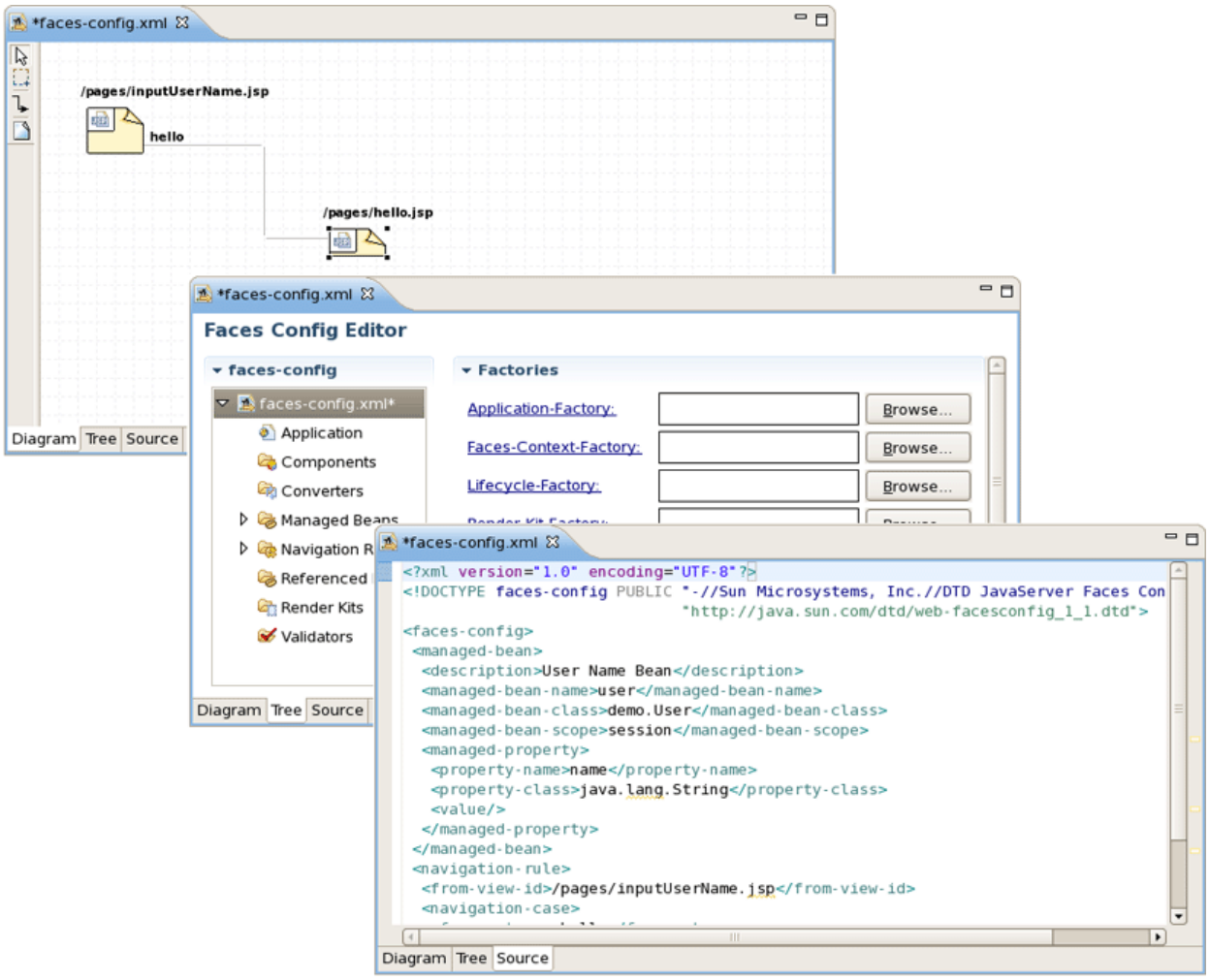

Figure 3.37. Three Views are Synchronized

The same is true of all other JBoss Developer Studio editors.

The Web XML editor is shown. The Web XML editor has a graphical view, accessed by the **Tree** tab, and a source view, accessed by the **Source** tab.

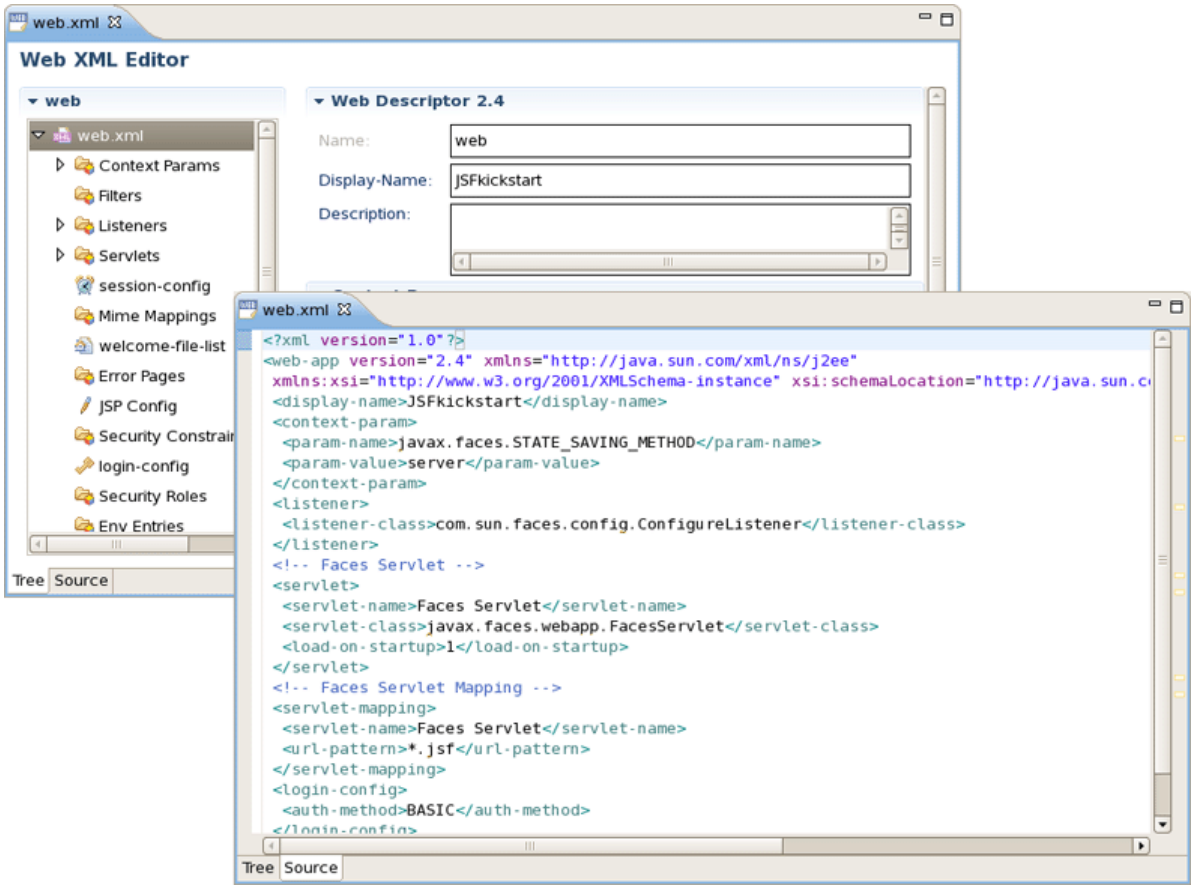

Figure 3.38. Two Views are Synchronized

The JBoss Developer Studio TLD file editor is shown below in **Tree** view. At any point you can edit the source by switching to the **Source** view.

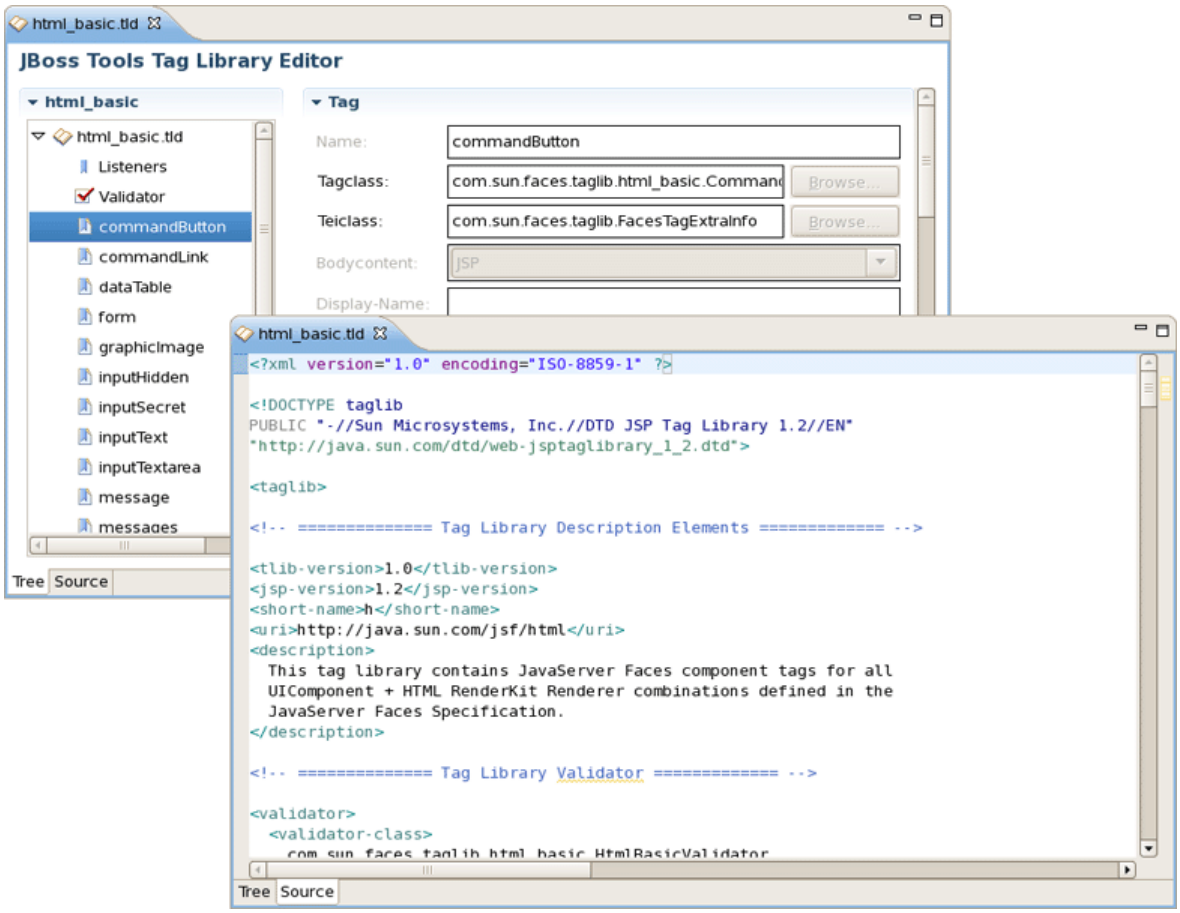

Figure 3.39. Two Views are Synchronized

3.2. Visual Page Editor

JBoss Developer Studio comes with a powerful and customizable Visual Page Editor (VPE). You can use the Visual Page Editor to develop an application using any technology such as JSF, JSP, HTML and more. Double-click on a file in the **Package Explorer** view to open it in the Visual Editor, or just drag-and-drop it into perspective (the drag-and-drop feature can be also applied to `JSP`, `XHTML` or `HTML` files created locally).

As a new JSF 2.0 specification has been released, support of new features is now implemented in the Visual Page Editor. The JSF 2.0 tags like `<h:body>`, `<h:head>`, `<h:outputscript>`, `<h:outputstyle>` are supported in the editor, as well as the composite components and expression language resource handling. (See the *JSF 2 fu, Part 2: Templating and composite components* [http://www.ibm.com/developerworks/java/library/j-jsf2fu2/index.html] for information on how to use composite components and *JSF 2.0 New Feature Preview Series (Part 2.3): Resources* [http://blogs.sun.com/rlubke/entry/jsf_2_0_new_feature3] on how to handle EL resources).

The current VPE version has three tabs: **Visual/Source**, **Source** and **Preview**. To switch between the views you can use tabs at the bottom of the VPE or the shortcut keys **Ctrl**+**PageUp** and **Ctrl**+**PageDown**.

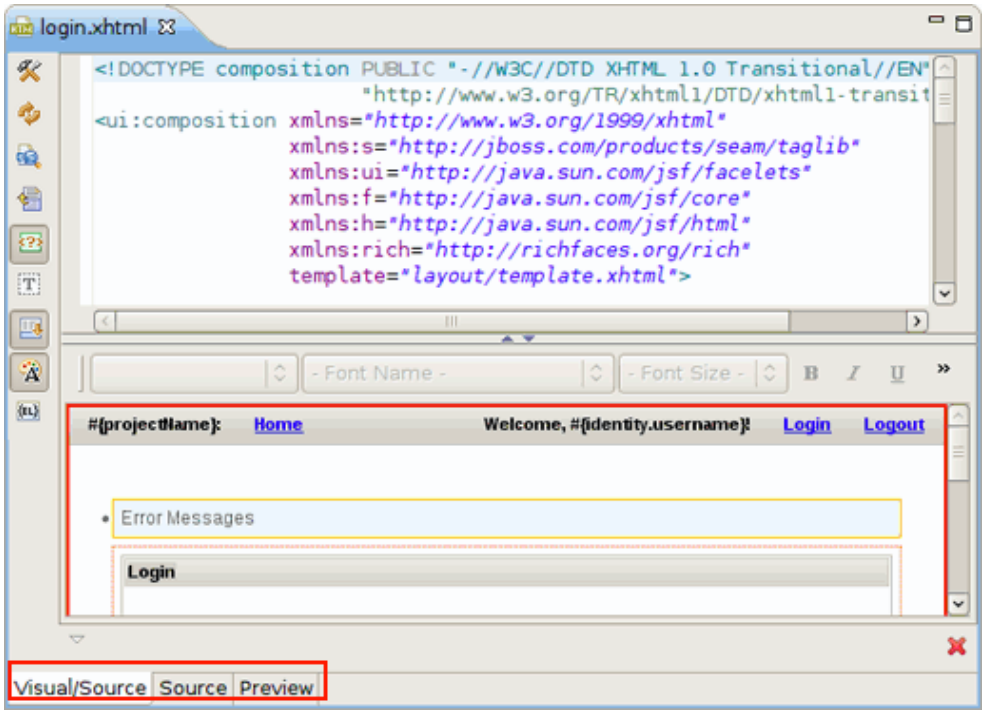

Figure 3.40. Visual Page Editor

3.2.1. Visual/Source View

Using the **Visual/Source** view you can edit your pages in the Source and Visual modes simultaneously, with instant synchronization between them:

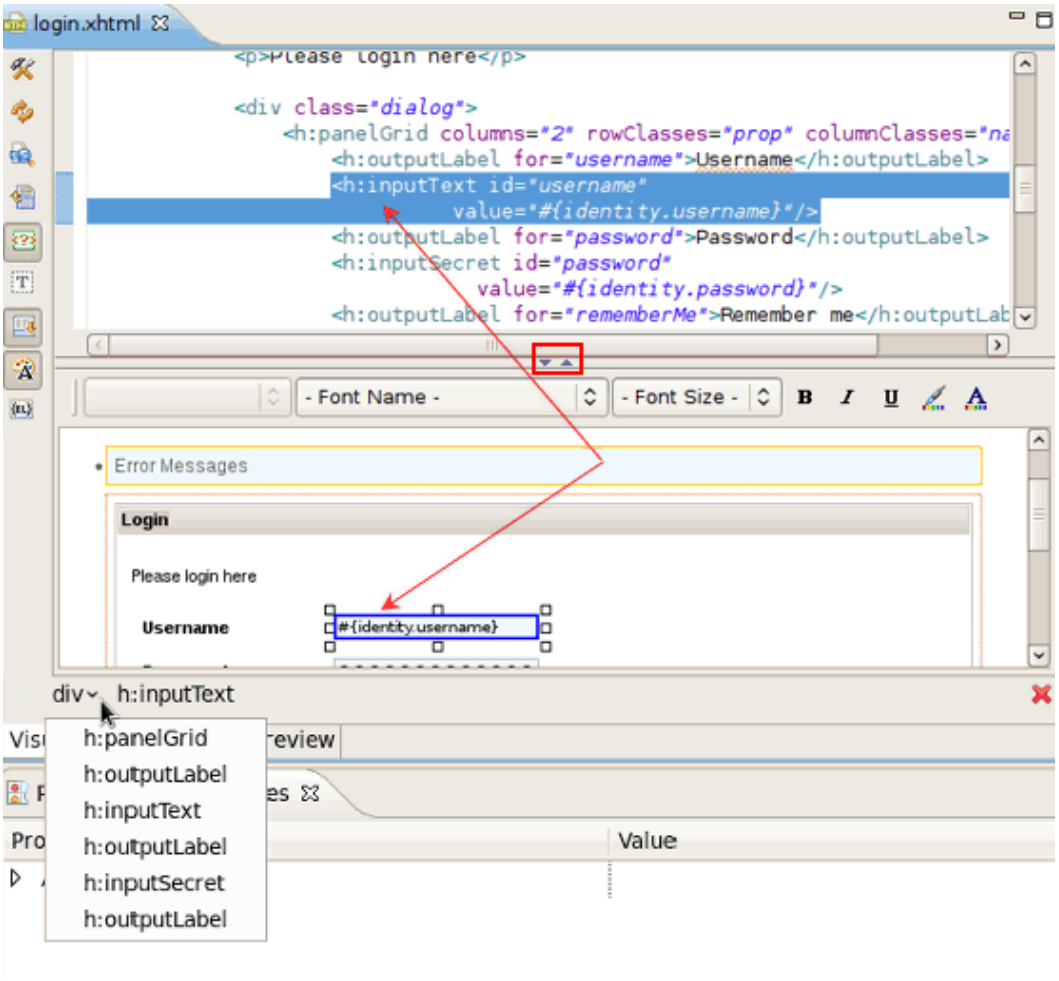

Figure 3.41. Visual/Source View

The view is designed in the form of a split pane with toggle buttons for quickly moving between **Source**, **Visual** or **Source/Visual** modes, as shown on the figure above.

One more way to toggle between the various states of the split pane is using the **Shift**+**F6** keyboard shortcut for maximizing or restoring the **Source** part and **Shift**+**Alt**+**F6** for maximizing or restoring the **Visual** part.

You can synchronize the scrolling between the source and visual panes by clicking the **Synchronize scrolling between source and visual panes** ( ) button in the toolbar of the Web Development perspective. This will ensure that scrolling in one window pane will automatically scroll the other pane to the same location.

Tip:

When editing large documents hiding the Visual part will speed up the editing.

It should be pointed out that, no matter what mode you are working in, you get a full integration with the **Properties** and **Outline** views:

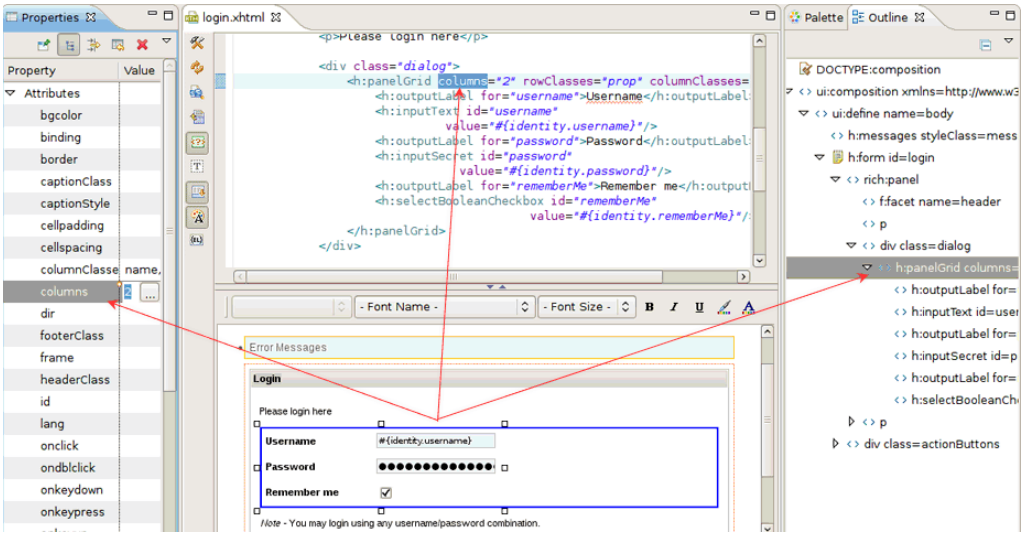

Figure 3.42. Integration with Properties and Outline Views

The **Outline** view displays a specific outline of a structured file that is currently open in the editor area, and lists its structural elements. Right-clicking on these elements will open additional options that allow other specific elements to be added in their appropriate positions.

The **Properties** view shows the property names and their values for a selected item. The values are editable: just select any value and click on the button that will appear to choose a new value. The key combination **Ctrl**+**Z** will return the previous value, while **Ctrl**+**Y** will return the new value again. The **Properties** view has additional options and can be set up to display categories and advanced properties.

It is also possible to use the **JBoss Tools Palette** (see *Chapter 5, JBoss Tools Palette*) to insert any tag from the list of tag libraries into the page you are editing with just a click or by dragging-and-dropping.

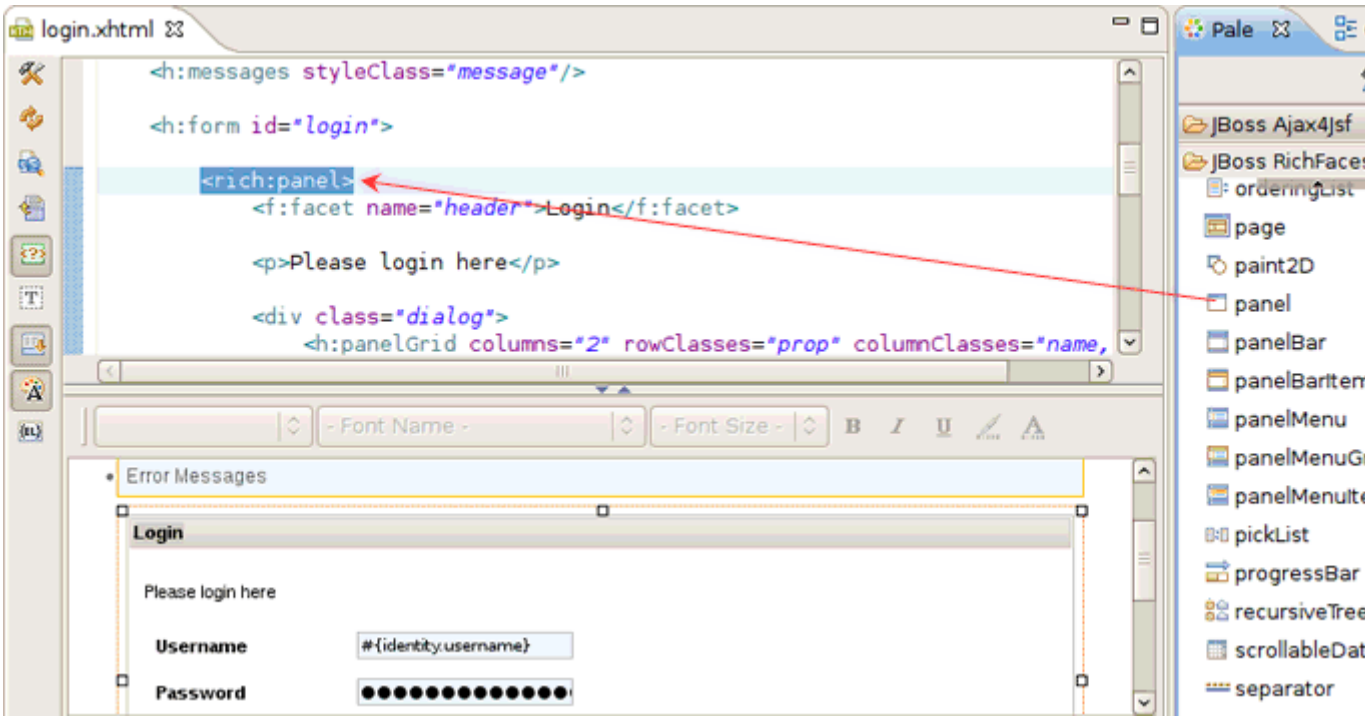

Figure 3.43. Inserting Tag From the Palette

You can insert a tag or component from the palette into either the **Source** or the **Visual** part by displaying the context menu and selecting **Insert around**, **Insert before**, **Insert after** or **Replace With**, picking the type of the tag and finally choosing the tag you want to insert.

The image below illustrates how you can insert a tag into the **Source** part.

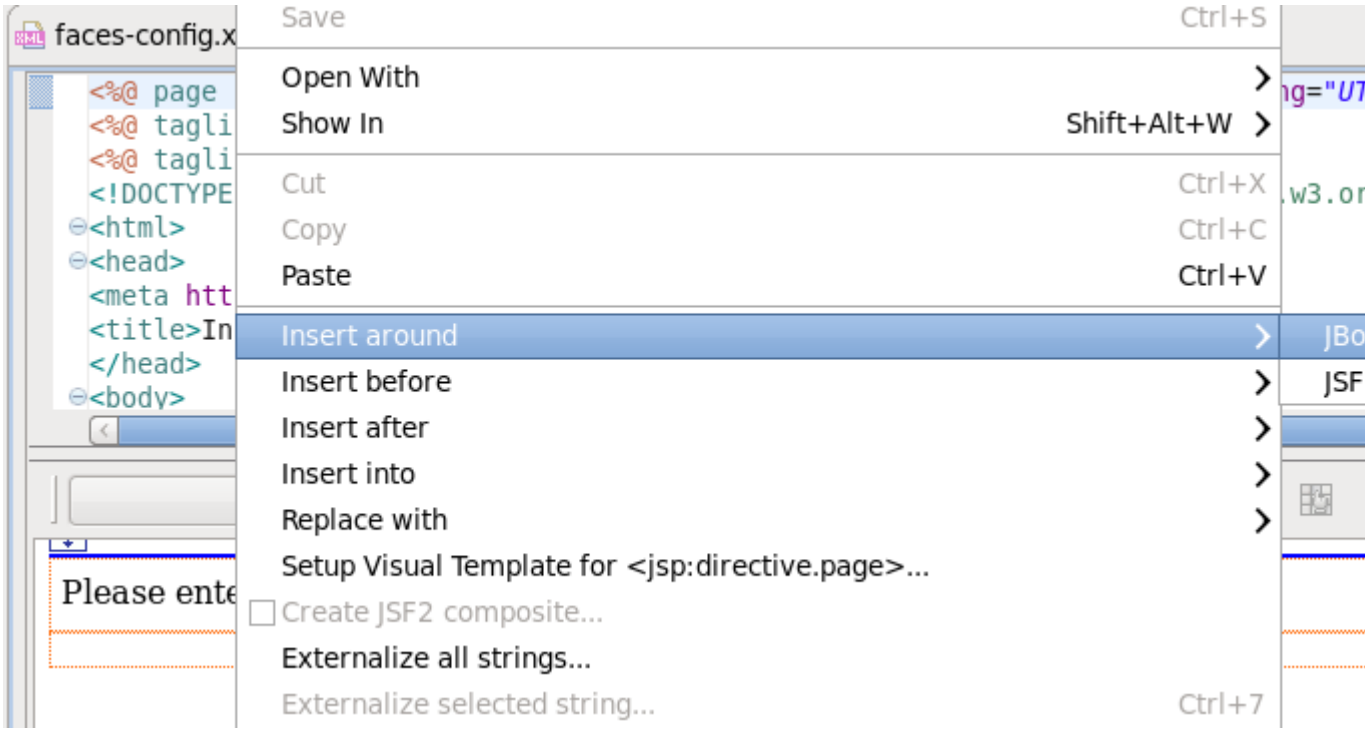

Figure 3.44. Inserting a tag into the Source part

And this is how a tag is inserted using a context menu in the **Visual** part.

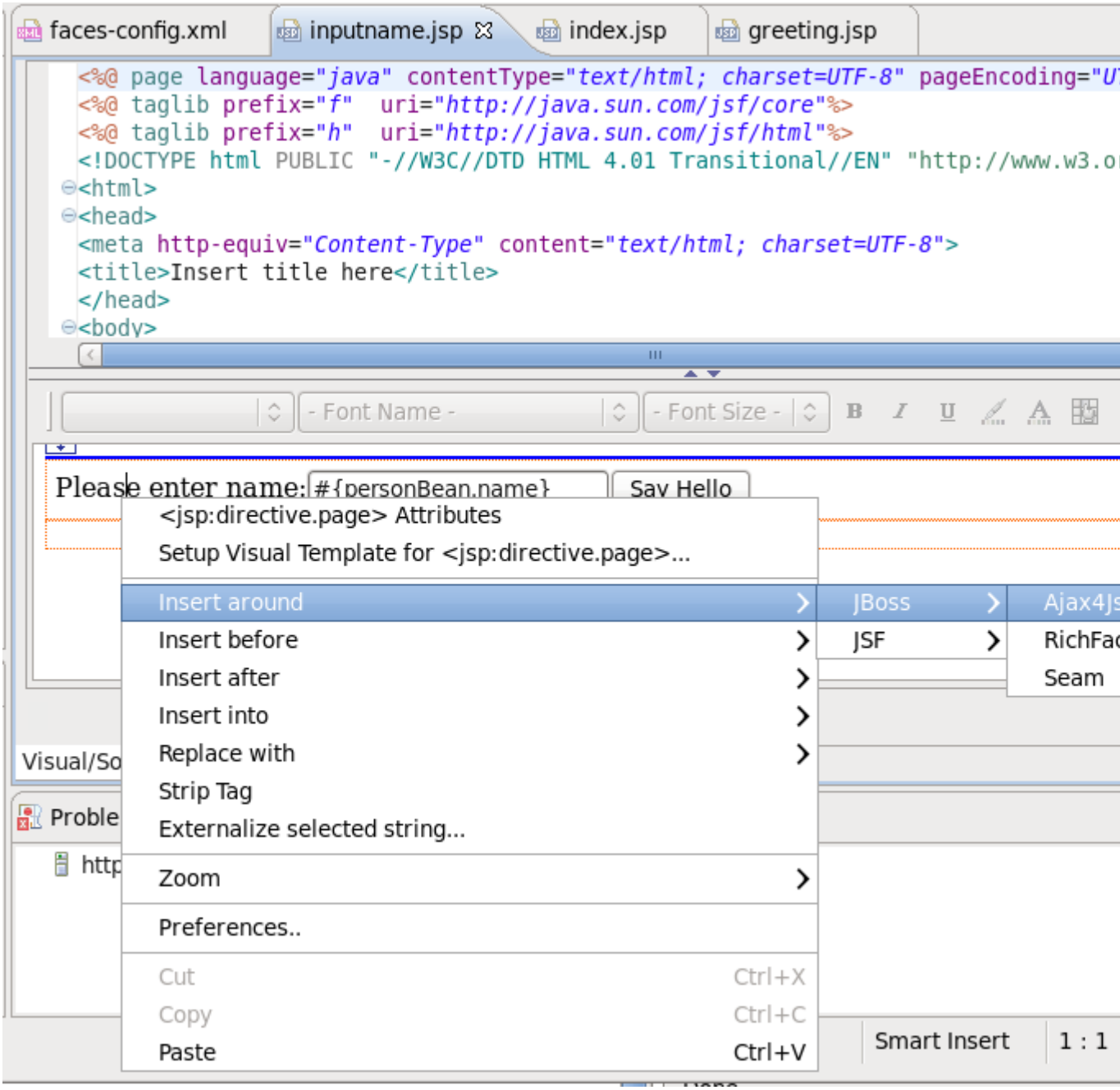

Figure 3.45. Inserting a tag into the Visual part

The Visual Page Editor also displays custom tags correctly if they are configured properly. The picture below shows an example how the custom tags `pagination` and `echo` will be displayed in VPE.

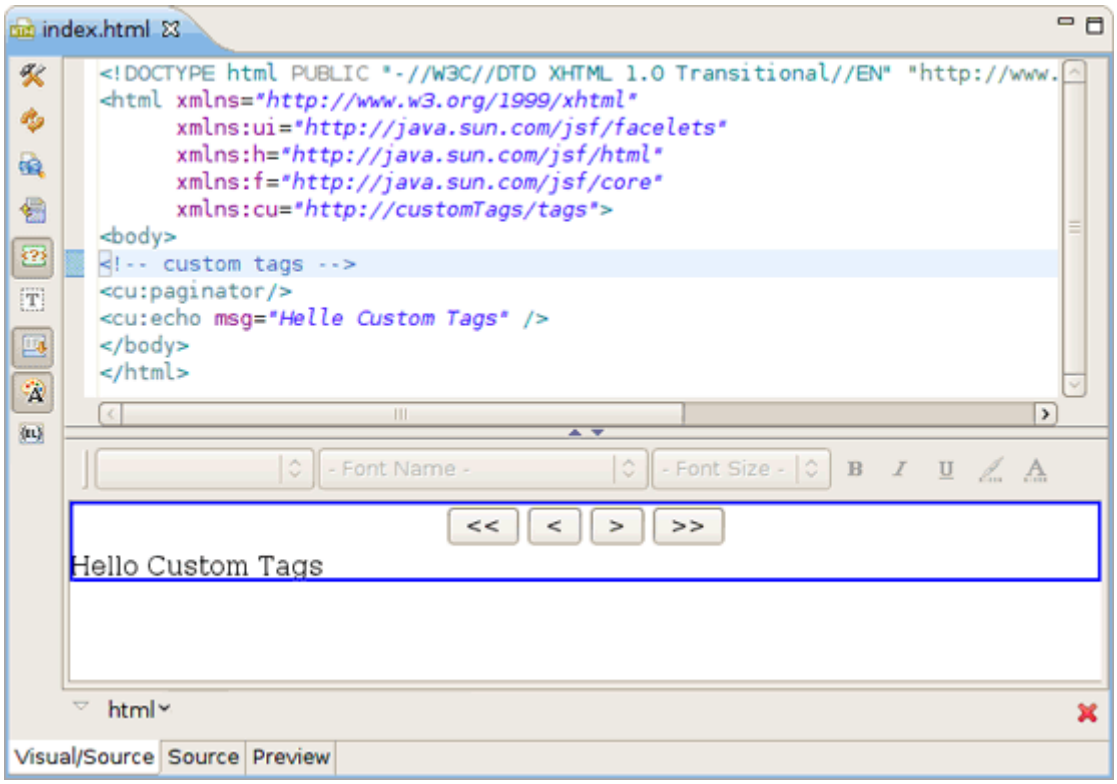

Figure 3.46. Custom Tags in the VPE

The listings of the custom tag implementations will help to demonstrate how VPE works.

- `echo.xhtml`:

```
<ui:composition xmlns:ui="http://java.sun.com/jsf/facelets">
 <span class="message">#{msg}</span>
</ui:composition>
```

- `paginator.xhtml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
 xmlns:ui="http://java.sun.com/jsf/facelets"
 xmlns:h="http://java.sun.com/jsf/html"
 xmlns:f="http://java.sun.com/jsf/core">
<ui:component>
<!-- h:inputHidden id="currentPage" replace, because if on page two fields,
```

```
two elements with equal id has been used, but should be used only one -->
 <h:panelGrid style="margin-right:auto;margin-left:auto;" columns="4">
  <h:commandButton value="&lt;&lt;" type="submit"
  onclick="document.getElementById('currentPage').value=0" >
  </h:commandButton>
  <h:commandButton value="&lt;" type="submit"
    onclick="document.getElementById('currentPage').value=#{user.currentPage-
user.rowsPerPage}">
  </h:commandButton>
  <h:commandButton value="&gt;" type="submit"
     onclick="document.getElementById('currentPage').value=#{user.currentPage
+user.rowsPerPage}">
  </h:commandButton>
  <h:commandButton value="&gt;&gt;" type="submit"
   onclick="document.getElementById('currentPage').value=#{user.numberOfItems
 - user.rowsPerPage}">
  </h:commandButton>
 </h:panelGrid>
 <h:inputHidden id="currentPage" value=""/>
</ui:component>
</html>
```

If your custom tags are not configured correctly your Visual mode will look like this:

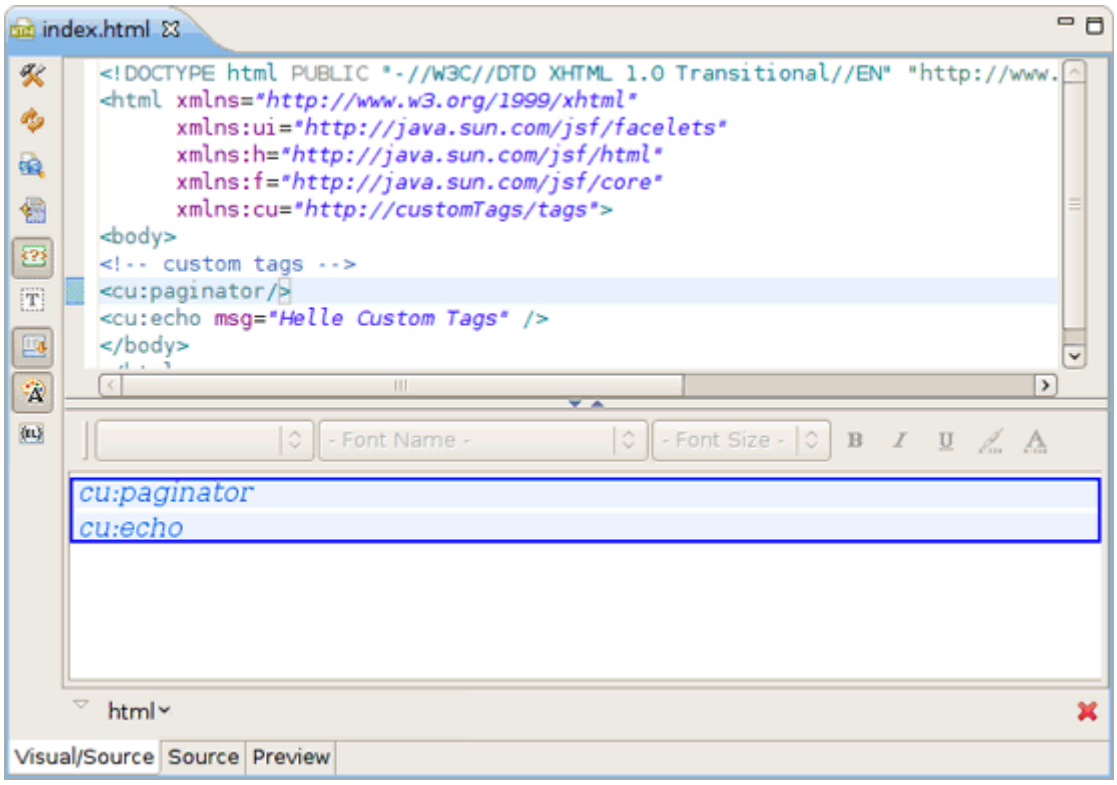

Figure 3.47. Wrong configured Custom Tags in the VPE

When you make a selection of tags in the source part of the editor, they will all be selected in the visual part of the editor as well. This makes it easy to link code that you have written, with the visual output.

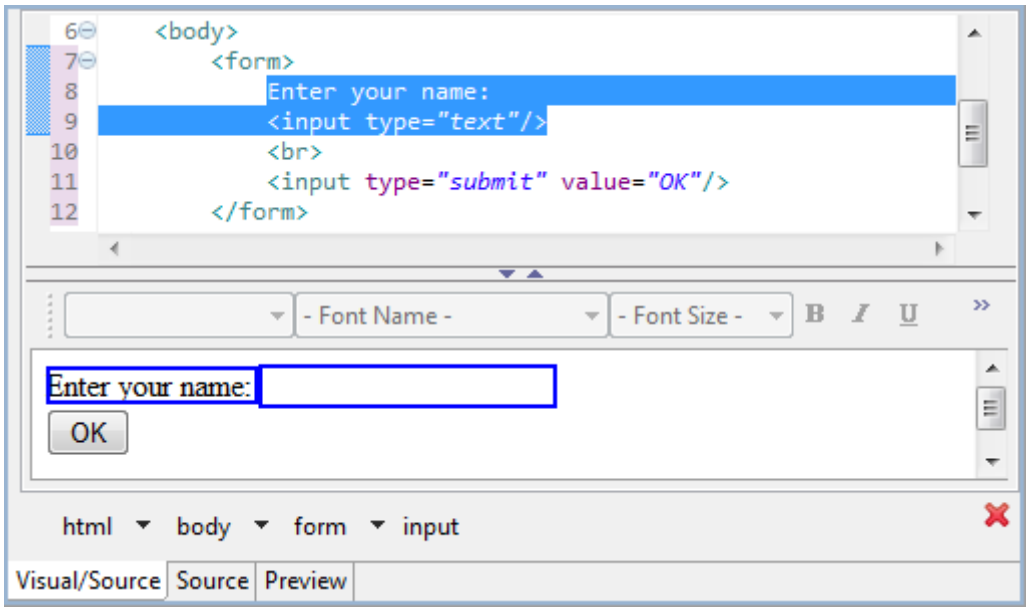

Figure 3.48. All tags from the source selection are selected in the visual part

3.2.1.1. Commenting out Code

The Visual Page Editor supports the ability to add comments in files you are working with (`JSP`, `XHTML`, etc.):

- HTML comments (`<!-- -->`) which are output to the client
- JSP comments (`<%-- --%>`) which are not output to the client as part of the JSP page output

3.2.1.2. Using Code Folding

The Visual Page Editor lets you collapse and expand (or hide and show) sections of your code to make it easier to navigate and read.

Code folding can be enabled by right-clicking on the left margin on the **Source** part of Visual Page Editor, selecting **Folding**, and checking the **Enable Folding** checkbox or using the **Ctrl**+**Numpad Divide** shortcut.

When the code folding is enabled a minus sing (⊖) will appear on the left margin of the editor next to each opening block tag.

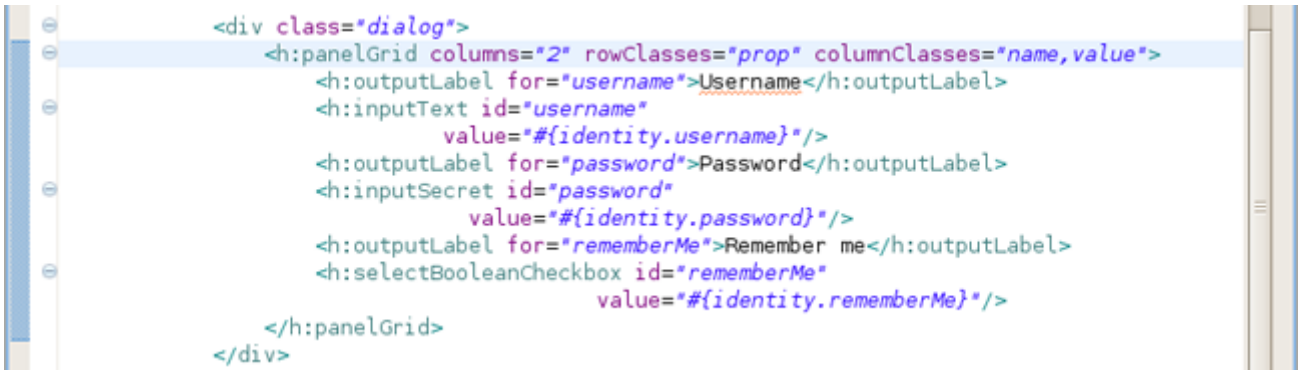

Figure 3.49. Enabled Code Folding

Click the minus sign to collapse a block tag.

When the minus sign is clicked on the appropriate tag collapses and a plus sign ( ) is displayed on the left margin as well as a gray rectangle with two dots ( ), which appears after opening and closing tags.

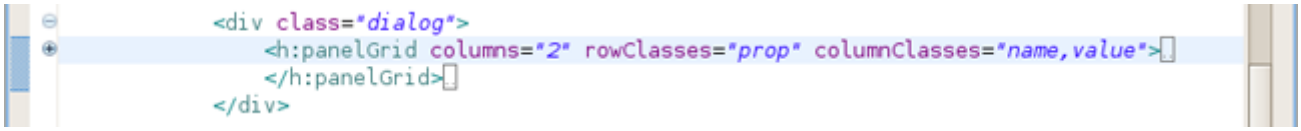

Figure 3.50. Collapsed Code

3.2.1.3. Creating a JSF2 component from the Source view

You can now create a JSF2 component while in the **Source** view. To do this, highlight the code you wish to create into a JSF2 component and then right-click the highlighted code segment to bring up the context menu.

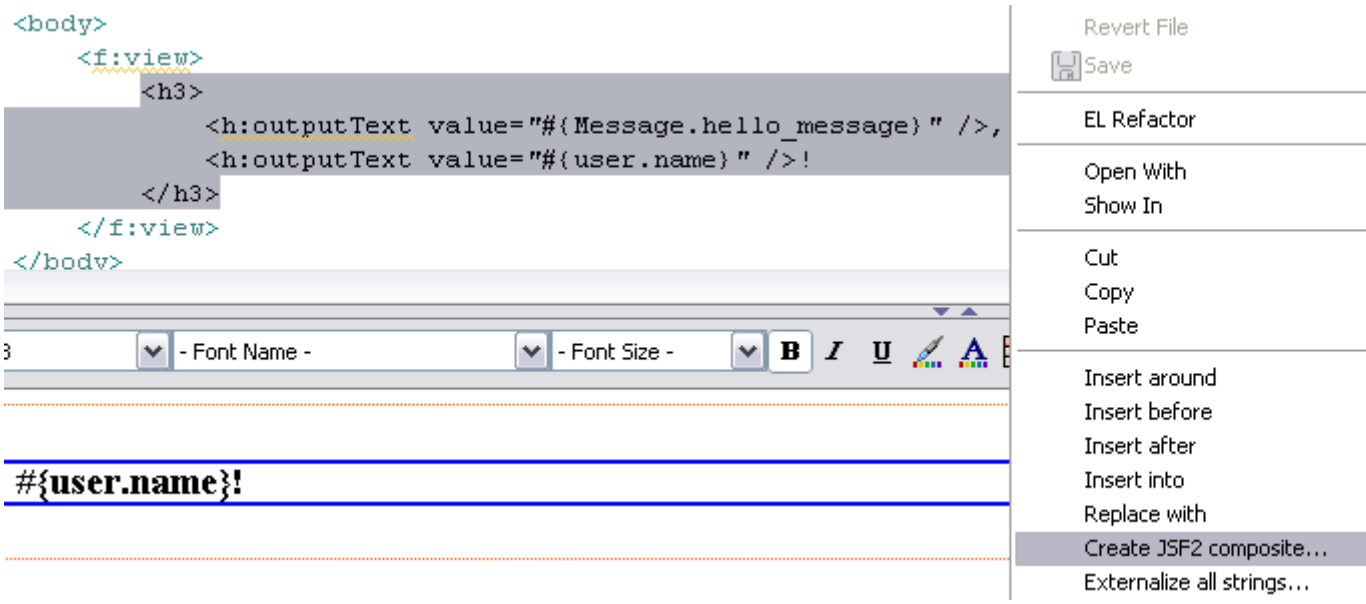

Figure 3.51. Collapsed Code

Within the context menu, select **Create JSF2 composite**. A new component will now be created containing the highlighted code.

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

<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:composite="http://java.sun.com/jsf/composite">

    <!-- INTERFACE -->
    <composite:interface>
    </composite:interface>

    <!-- IMPLEMENTATION -->
    <composite:implementation>
<h3>
                <h:outputText value="#{Message.hello_message}" />,
                <h:outputText value="#{user.name}" />!
            </h3>
    </composite:implementation>

</html>
```

Figure 3.52. Collapsed Code

3.2.1.4. JSP Syntax Validation

When working in the JBoss Tools JSP editor you are constantly provided with feedback and contextual error checking as you type.

3.2.1.5. Support for custom TagLibs and Taglib versions

VPE templates support custom tag libs, e.g. Seam Mail facelet taglib, RichFaces taglibs or any other created by you.

VPE templates also provides support for various versions of tag libraries, meaning that the Visual Page Editor takes control over those components which have different parameters or preview according to the framework version (like seam 1.2 and seam 2.0, or JSF 1.1 and JSF 1.2).

For example, the `<s:decorate>` element in Seam has different parameters in versions 1.2 and 2.0, and the `<h:outputLink>` JSF element has different preview in versions 1.1 and 1.2.

3.2.2. Pages Styling

Most web pages use the cascading style sheets (CSS) to control the way they look. With Visual Page Editor you can easily stylize your pages. In this section we are going to introduce you to a powerful mechanism that Visual Page Editor provides for complete control over a pages' styling. Additional information on working with CSS files can be found in *Chapter 6, CSS Editing Perspective*

3.2.2.1. Text Formatting

In the **Visual** part of the Visual Page Editor there is a graphical toolbar, which is used to add inline styling to JSF tags on your page. The toolbar can be hidden with the help of the special button ( ) on the Visual Page Editor toolbar.

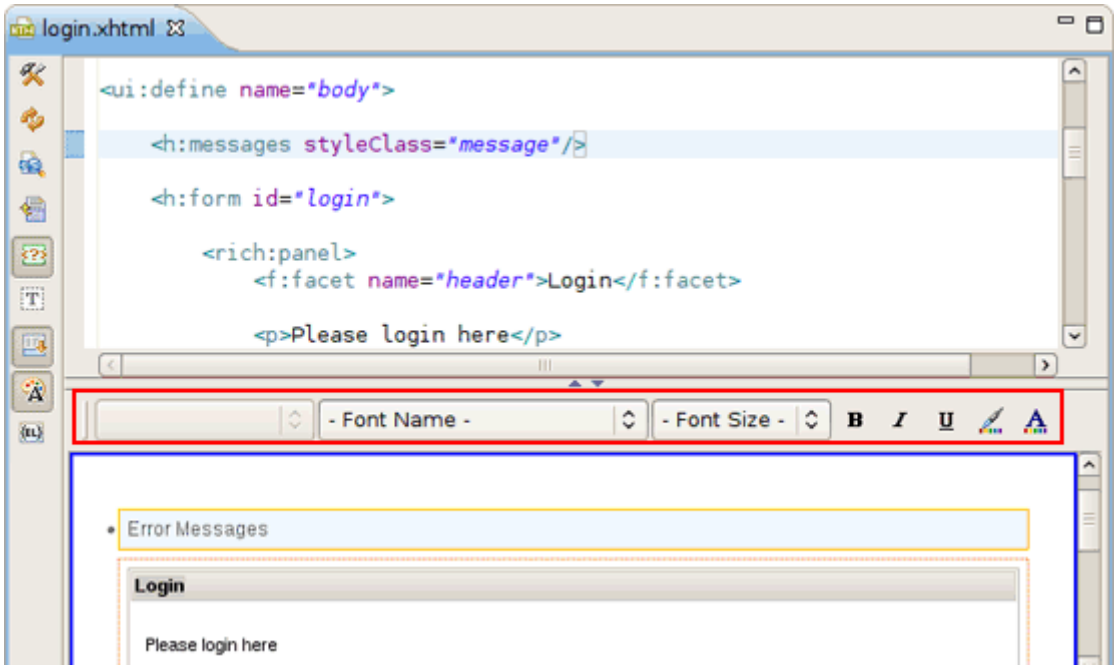

Figure 3.53. Text Formatting

For editing inline styles for DOM elements, the Visual Page Editor provides the **CSS Dialog**. It can be called from the **style** line in the **Properties** view of a currently selected element.

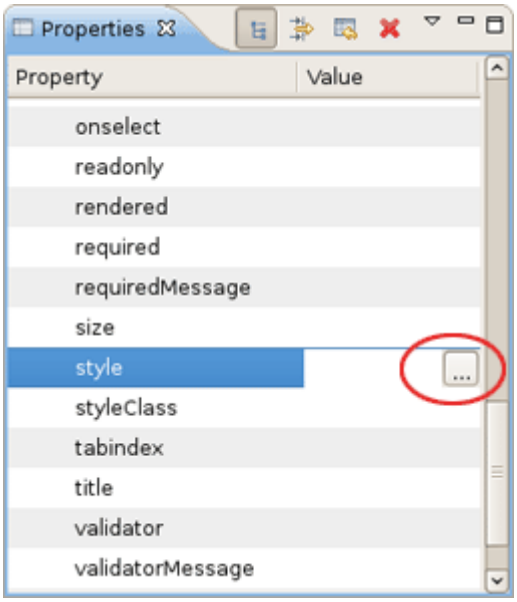

Figure 3.54. Call the CSS Dialog

The **CSS Style** dialog has several tabs where CSS properties for text, background, borders and others can be specified. A simple preview which is generated at the top of the **CSS Style** dialog allows you to see the changes before you apply them.

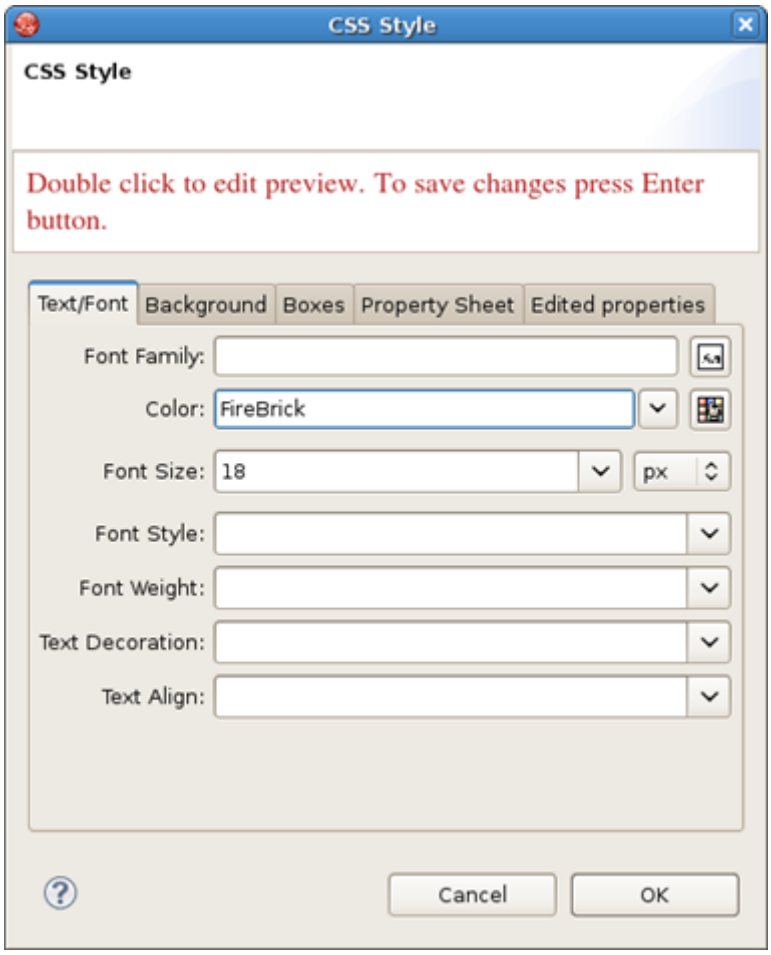

Figure 3.55. CSS Style Dialog

3.2.2.2. External Stylesheets (CSS)

The pages you are working with in the Visual Page Editor can use external stylesheets.

3.2.2.2.1. Importing a stylesheet

You can import an existing stylesheet by using the @import annotation within your webpage file. The annotation for importing a file is structured as @import "path/to/file.css";.

An example of how it is used can be seen below:

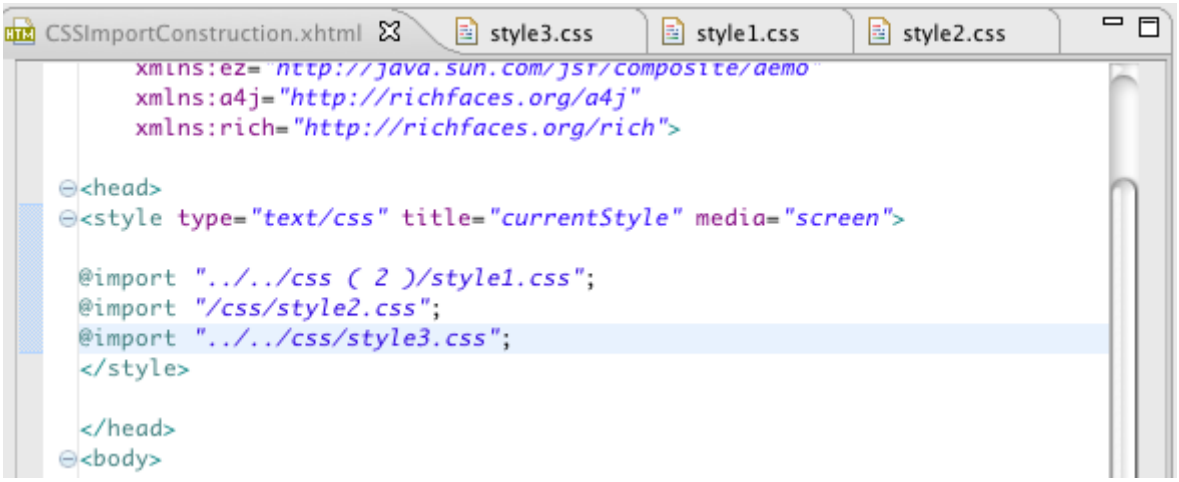

Figure 3.56. Importing a stylesheet

3.2.2.2.2. Modifying an existing stylesheet

The Visual Page Editor allows you to create new style classes in existing stylesheets, as well as edit them. The **Edit Style Class** dialog is provided for this purposes.

Select the element for which you need to create or edit style class, and press button next to the **styleClass** field in the **Properties** view.

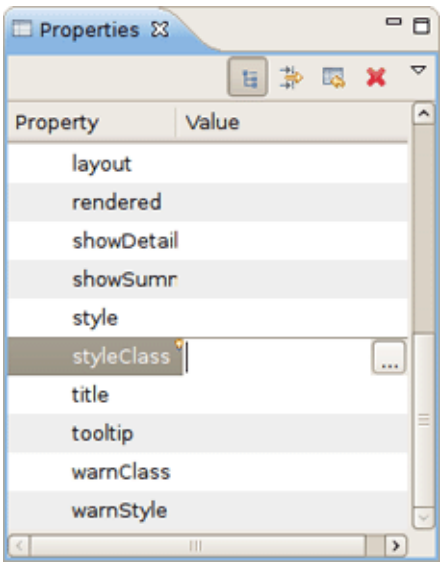

Figure 3.57. Calling the Edit Style Class Dialog

This will display the **Edit Style Class** dialog, which is shown in the image below:

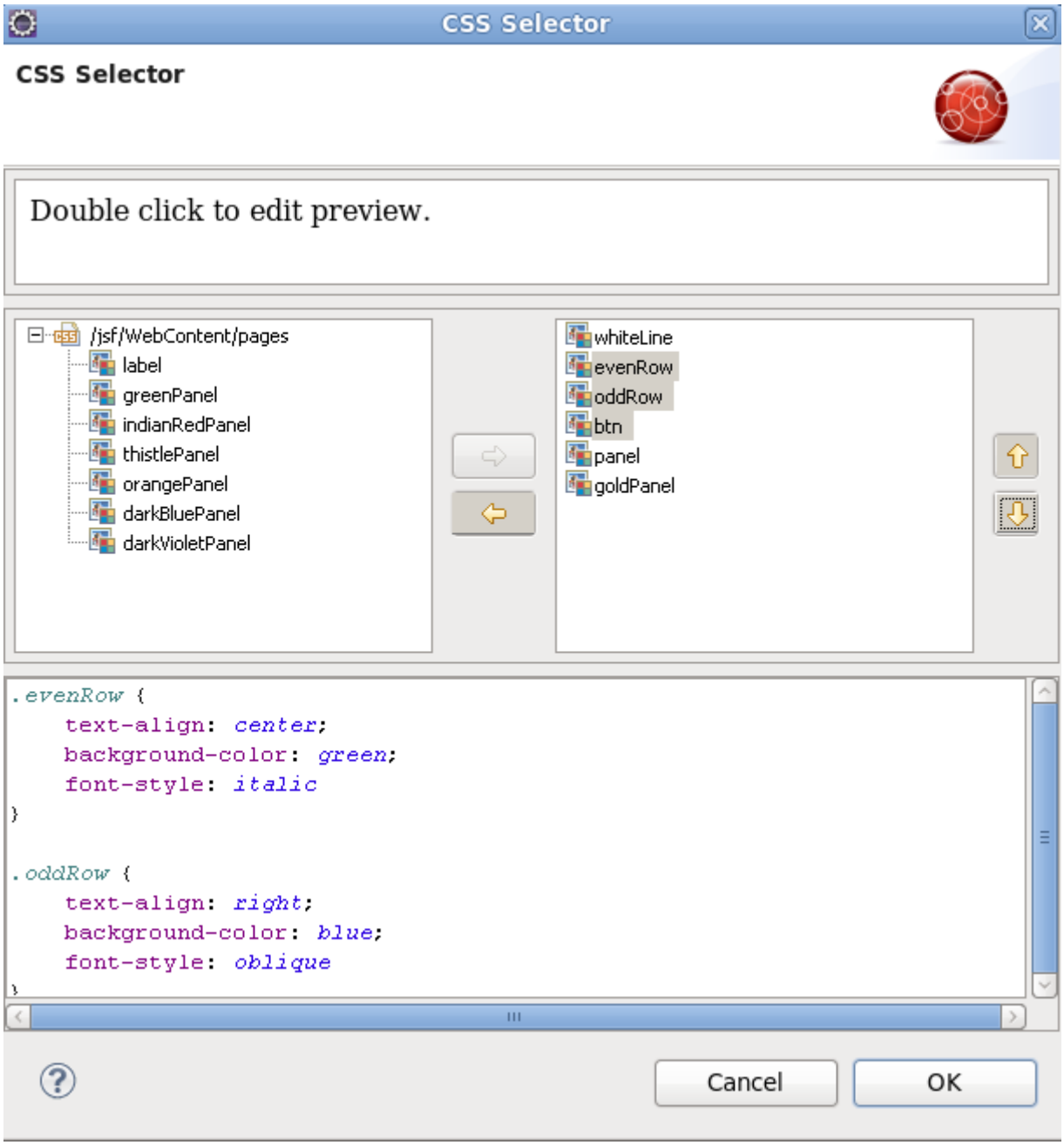

Figure 3.58. Edit Style Class Dialog

Choose a style class from the variants provided, and click on the **OK** button to apply the changes.

To open a CSS dialog based on the active CSS file click on in the top panel or use hot-keys (**Shift**+**Ctrl**+**C**).

To create a new CSS class for the file click on the **Add CSS Class** button, enter its name in the textbox, and click on the **OK** button:

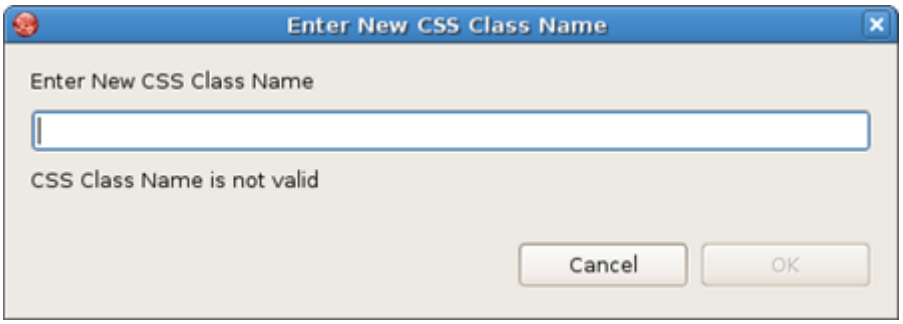

Figure 3.59. Add CSS Class

Then you can configure style settings by switching between the tabs: **Text/Font**, **Background**, **Boxes**, **Property Sheet**. The list of existing classes with names beginning with the symbols printed will be displayed by using the standard **Ctrl**+**Space** key combination. To add an existing style to the chosen element just point to the necessary one. Each time you select any class it is displayed in the **Preview** tab. Click on the **Apply** button to apply the changes without closing the window.

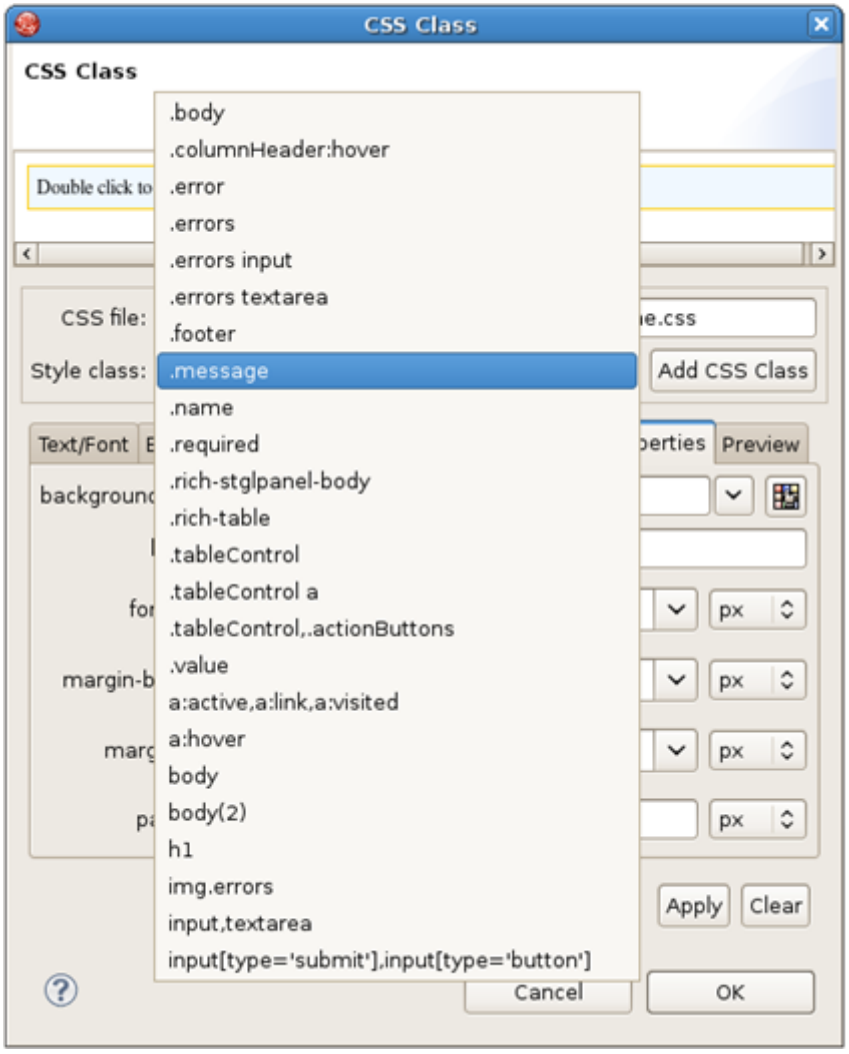

Figure 3.60. Style Class Selection

The **Edited properties** tab provides a preview of the properties which are set for the existing style class. You can easily modify them with the help of this wizard.

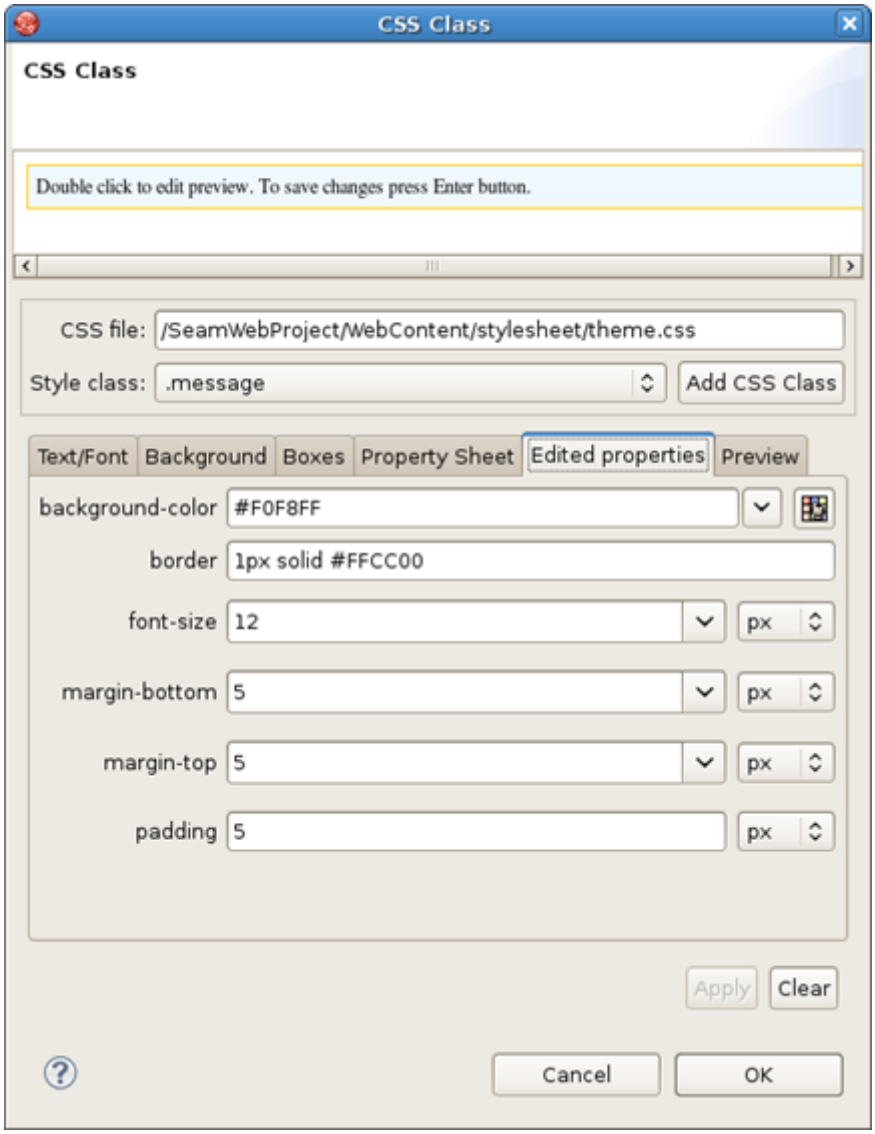

Figure 3.61. Edited Properties

If the style class is not chosen, the tab does not show any properties.

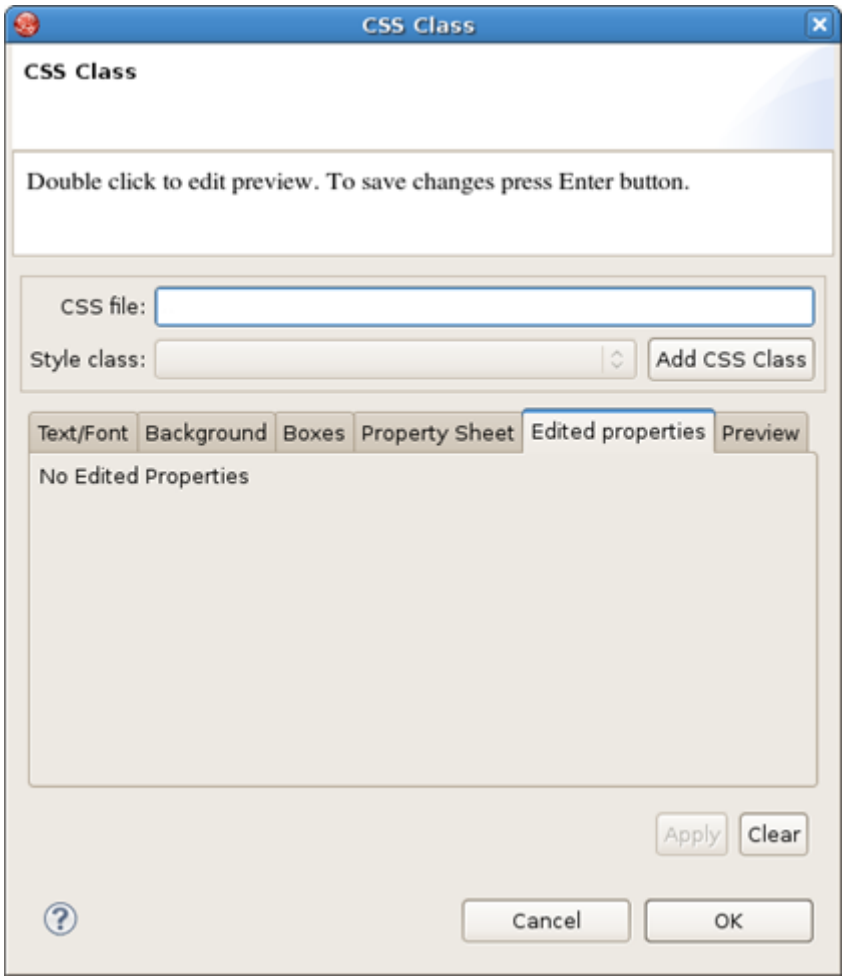

Figure 3.62. Edited Properties when the style class isn't chosen

The **Preview** tab provides a way to view the content of the selected CSS file. This tab is hidden if no CSS file is selected.

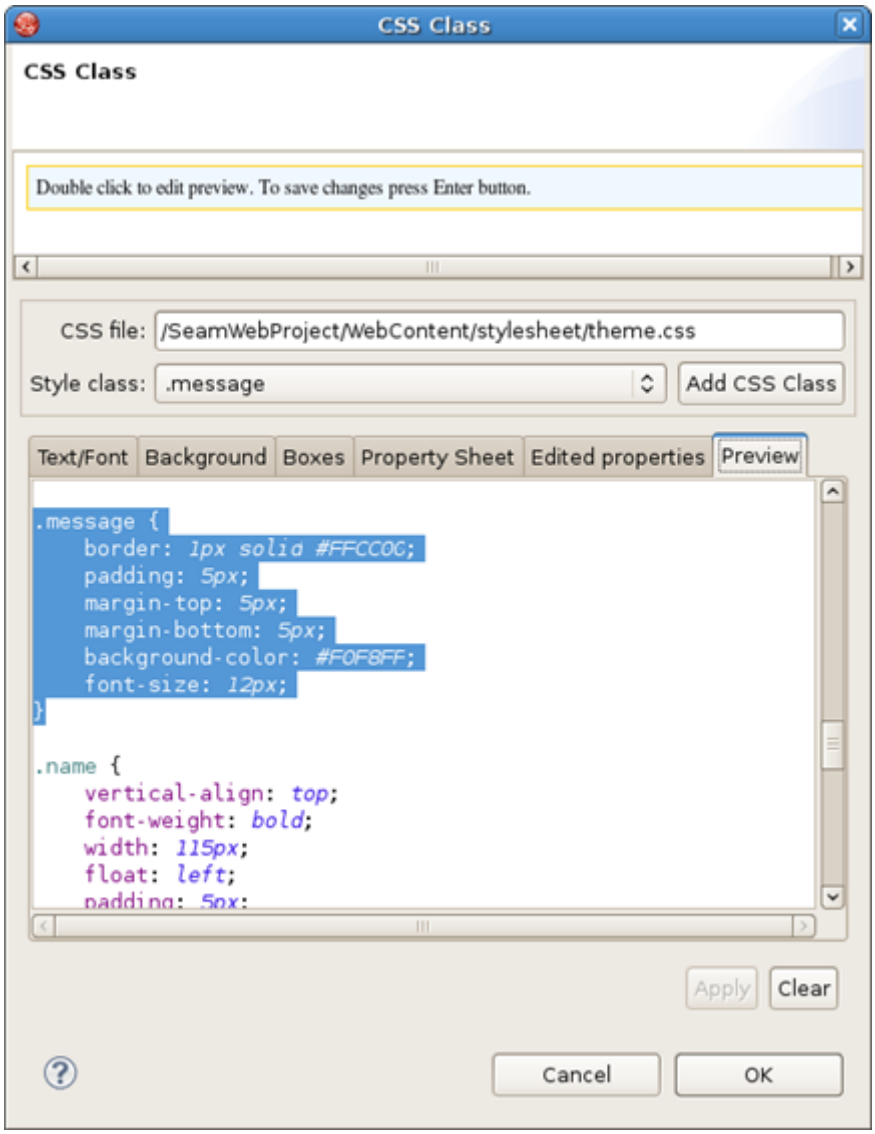

Figure 3.63. Preview Tab

At the top of the **CSS Class** dialog you can see a preview box which visualizes the result. To edit the preview you should double click in the box. To leave the focus, use **Ctrl**+**Tab**.

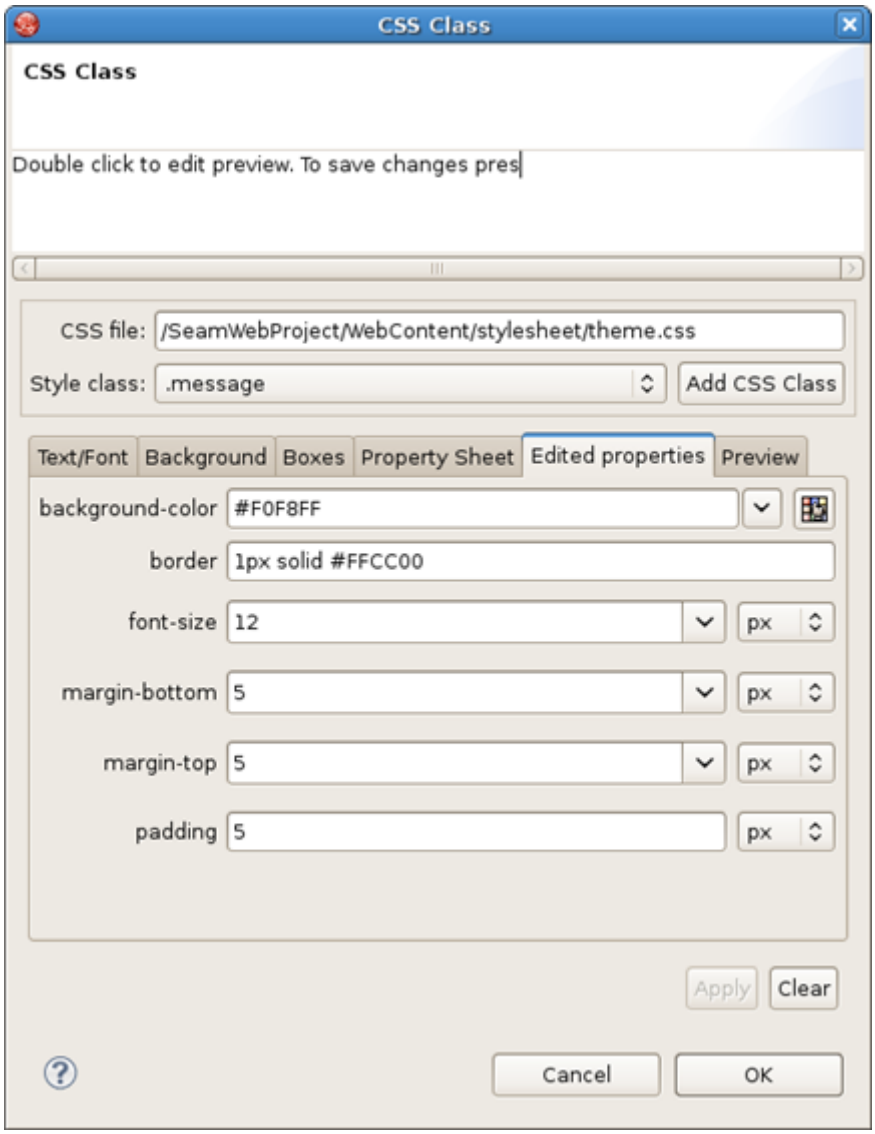

Figure 3.64. Editing the Preview

The dialog for creating a new CSS class, which is called from **New** → **Other...** → **JBoss Tools Web** → **CSS Class**, is shown in the image below:

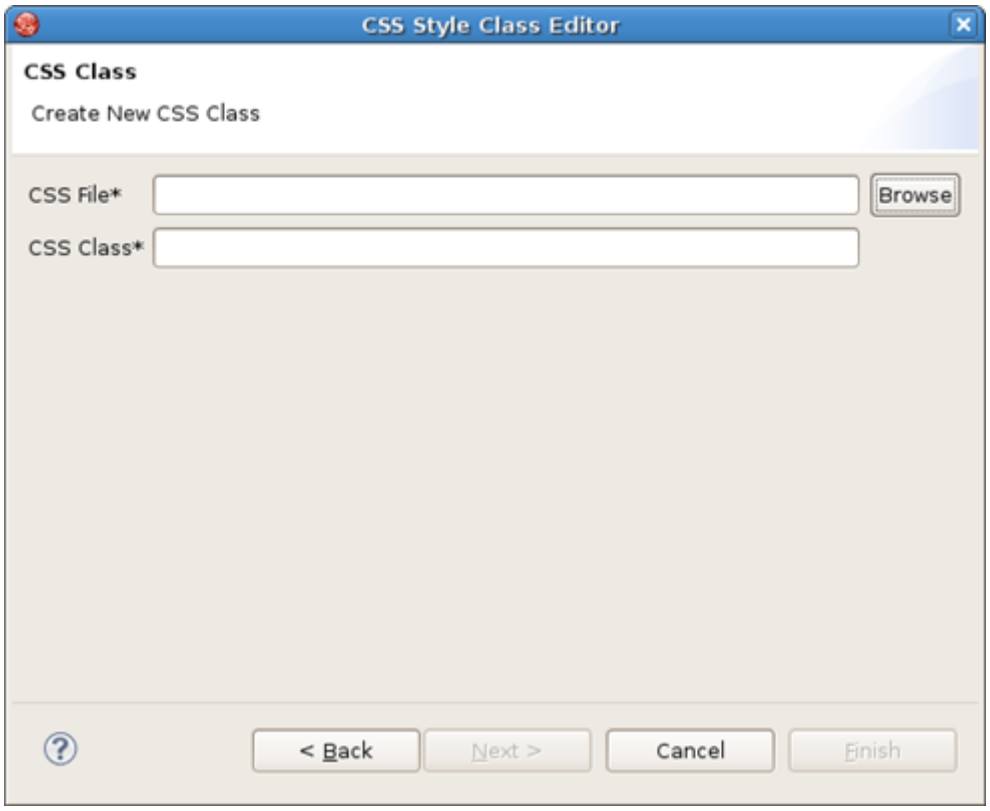

Figure 3.65. New CSS Class Dialog

Click on the **Browse** button to open a dialog where you can select the CSS file to create a CSS class for:

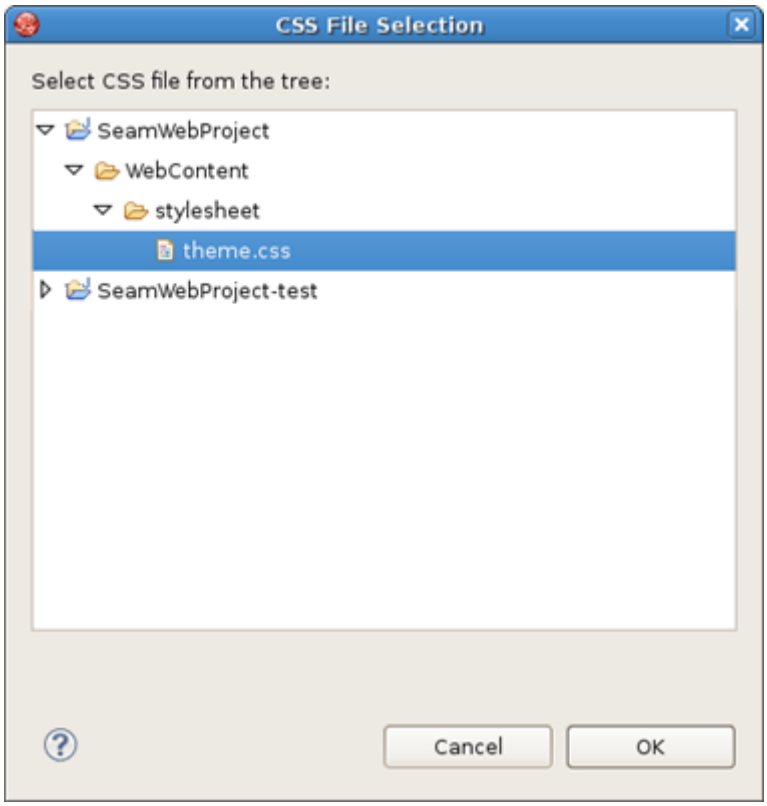

Figure 3.66. CSS File Selection

Choose the appropriate CSS file and click on the **OK** button.

3.2.3. Visual Templates for Unknown Tags

The Visual Page Editor also makes it possible to create visual templates for unknown tags.

To display the **Template** dialog for a tag, right-click on it in Visual mode and select **Setup Visual Template for <tag name>** option.

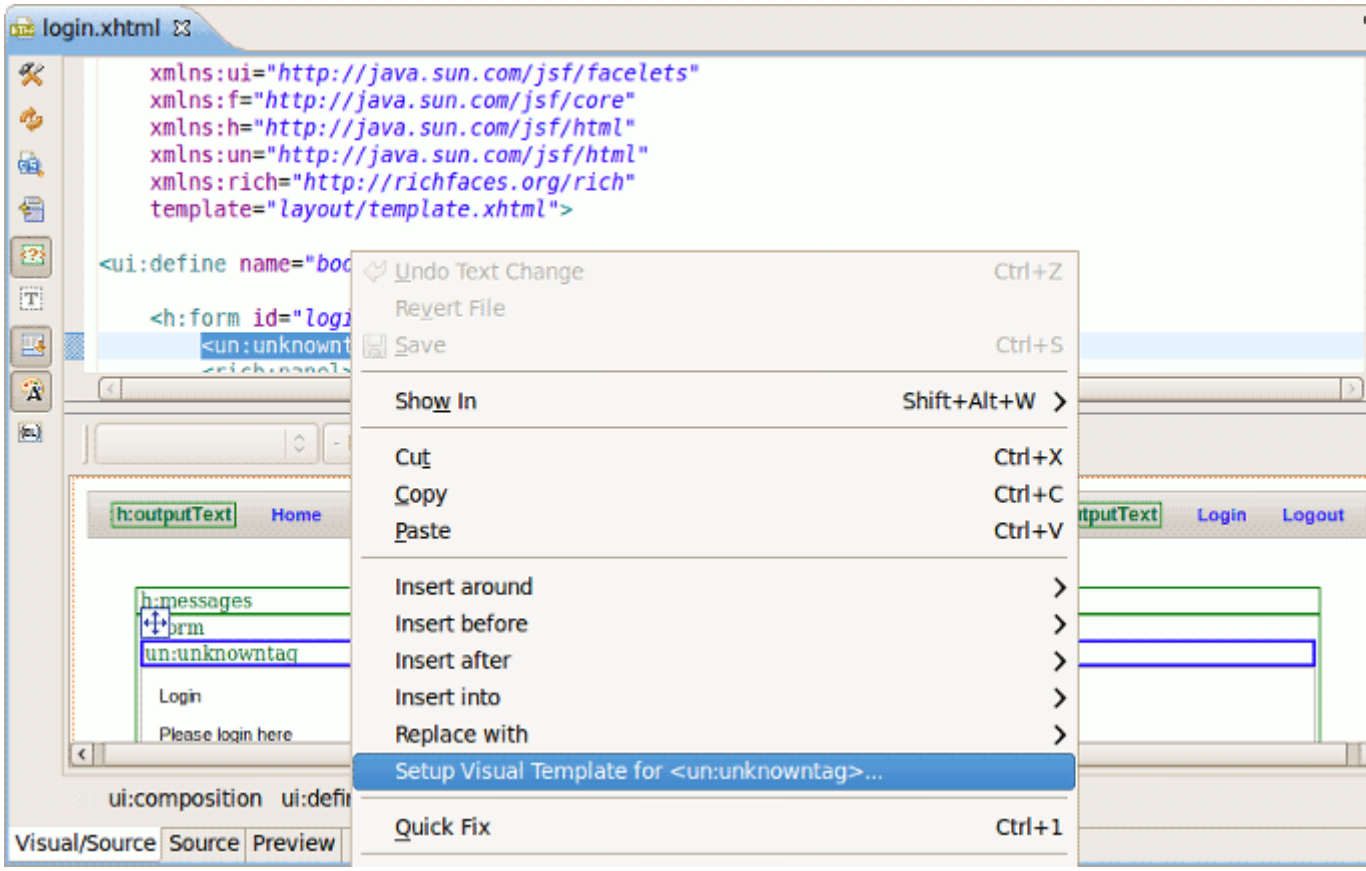

Figure 3.67. Calling Template Dialog

The **Template** dialog is shown in the image below:

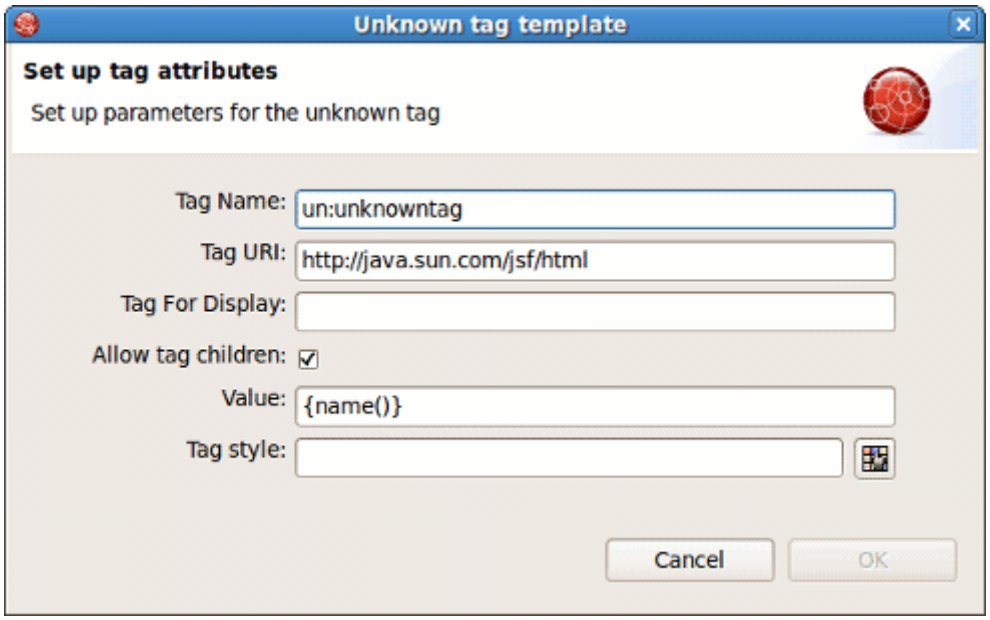

Figure 3.68. Template Dialog

The **Tag Name** field is used to define the name of the unknown tag.

Note:

The given field should be filled in according to the pattern: `taglib:tag`. Also make sure you do not surround the name with angle brackets which will cause the validation error (see the figure below).

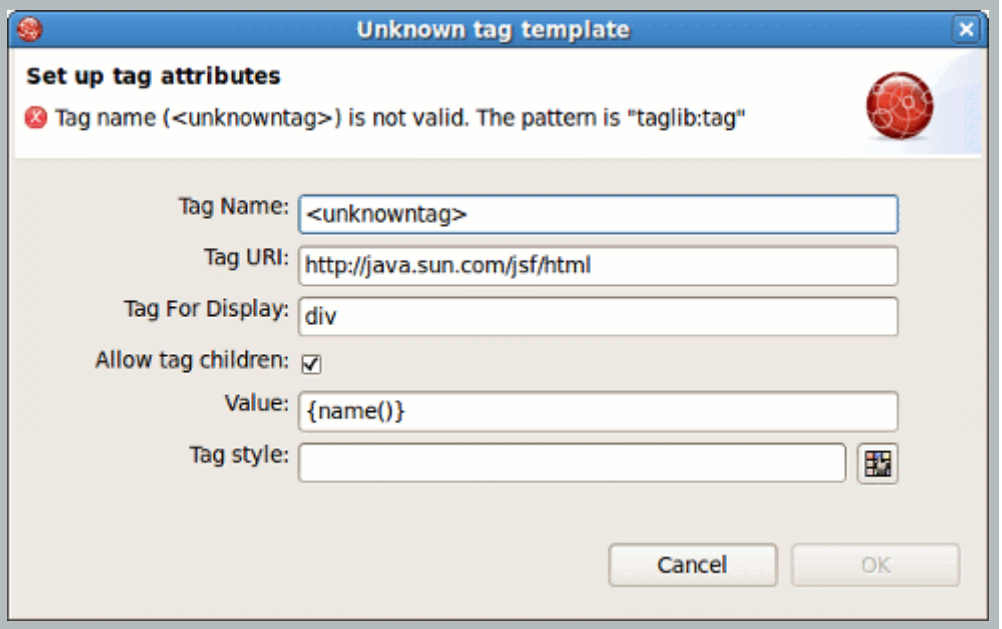

Figure 3.69. Validation Error in the Template Dialog

The **Tag for Display** field in the **Template** dialog requires specifying a type of tag. It can be SPAN, DIV, TABLE or any other HTML element. Check the **Children** field if you want to mark a tag as a child element.

The **Value** defines a tags' value.

As for the **Style** field, you can fill it out manually or make use of the button next to the field to bring the **CSS Style** dialog (See *Section 3.2.2.1, "Text Formatting" [59]*) for editing styles.

You can view all defined templates in the *Section 9.3, "Visual Page Editor"* on the **Visual Templates** tab which, you can quickly access by pressing the
toolbar button (see *Section 3.2.5, "VPE Toolbar"*).

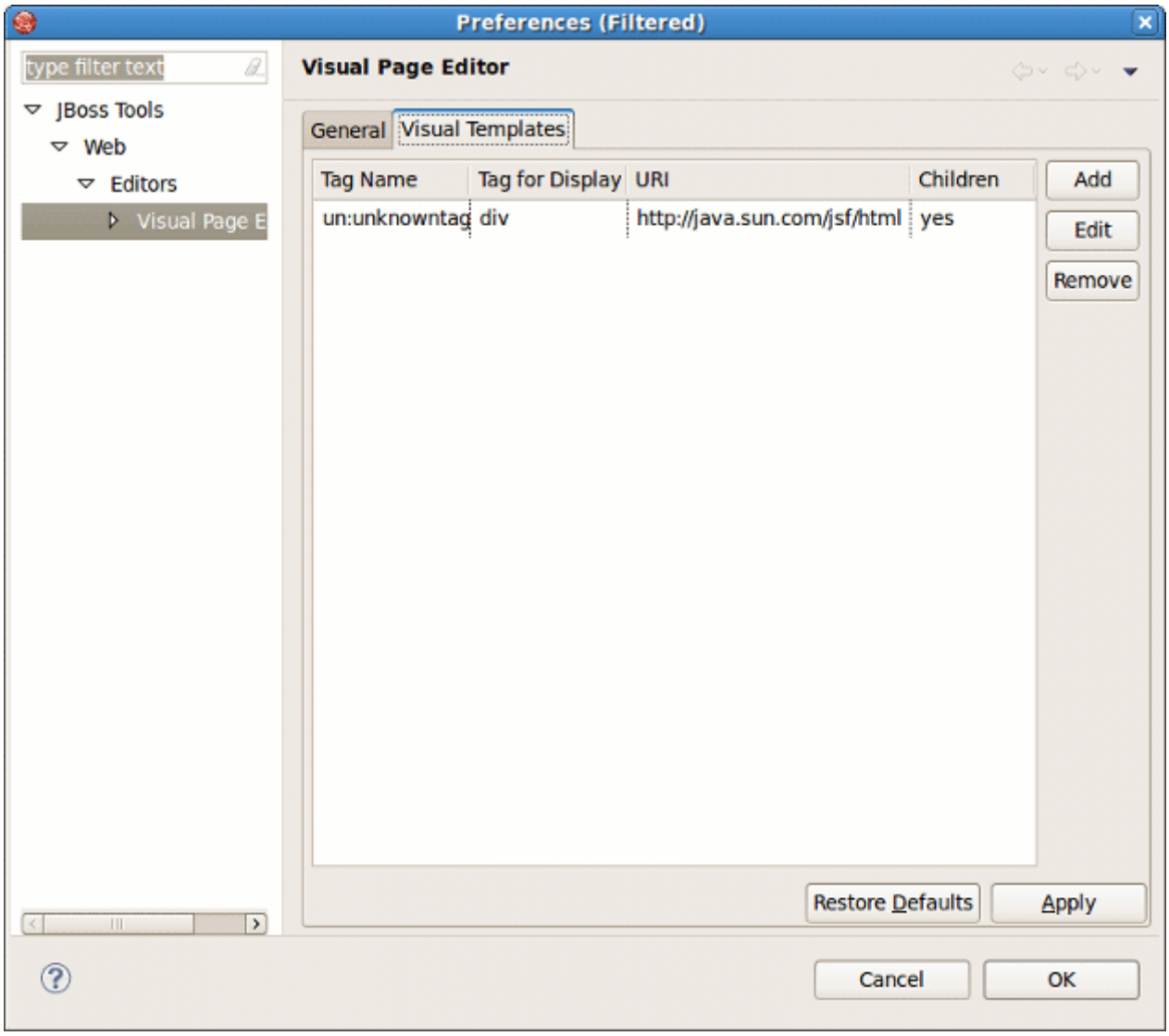

Figure 3.70. Templates Tab of the VPE Preferences Page

Here it's possible to add, edit or remove any listed in the table template.

3.2.4. Export/Import of the Templates for Unknown Tags

If you have a number of custom tags for which you have defined visual templates, you may need to share the templates definitions with other team members. In this case you can use export and import functionality for unknown tag templates.

To export all visual templates you defined for unknown tags, select **File** → **Export** → **Other** → **Unknown tags templates**. Here is what the wizard looks like.

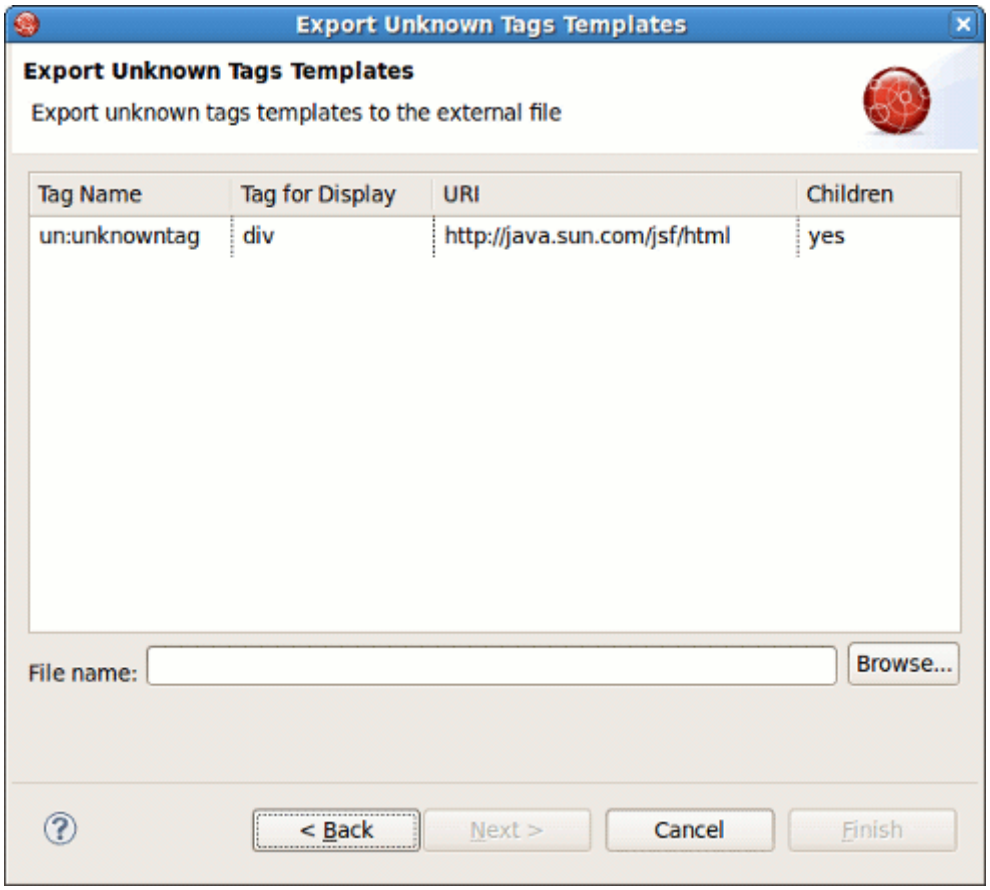

Figure 3.71. Export of Unknown Tags Templates

At this point click the **Browse** button to set the path where to save the external XML file with templates, and then click the **Finish** button to complete the export.

Importing follows a similar procedure. Select **File** → **Import** → **Other** → **Unknown tags templates** to open the import wizard. Click the **Browse** to point to the `XML` file which stores the custom tags templates, and then click the **Finish** button to complete the import.

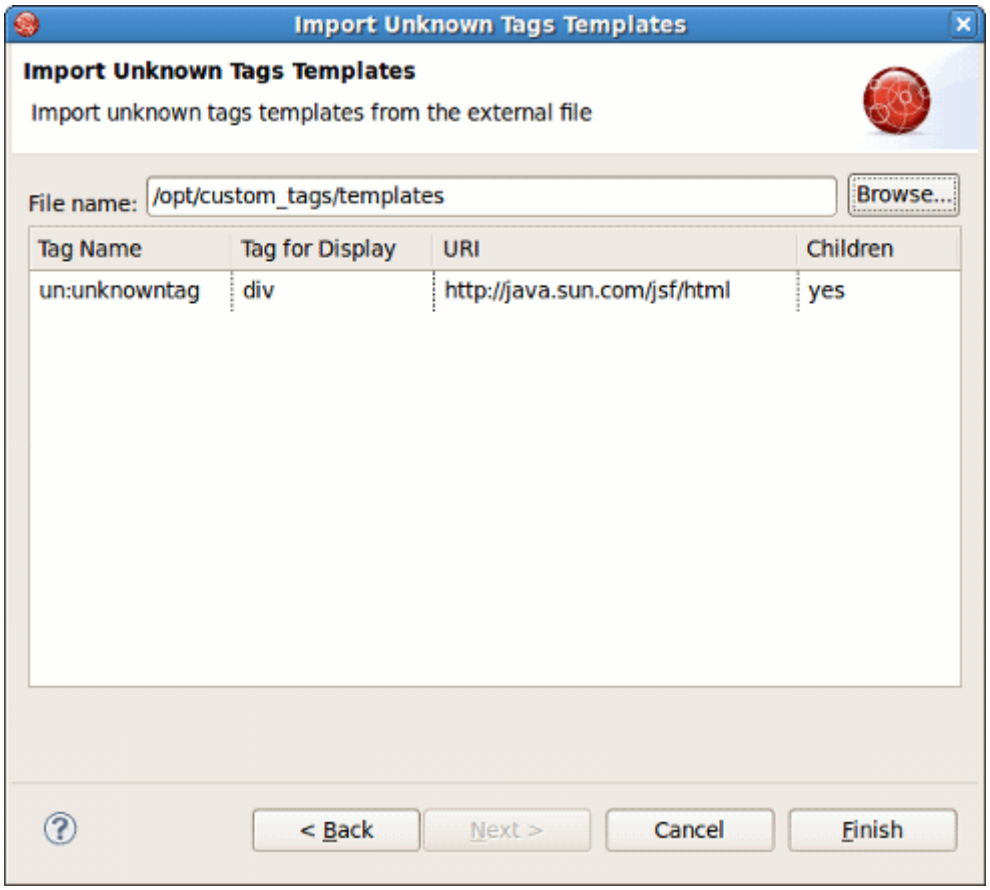

Figure 3.72. Import of Custom Tags Templates

3.2.5. VPE Toolbar

The Visual Page Editor toolbar includes the following buttons:

- *Section 3.2.5.1, "Externalize string"* (

)
- *Section 3.2.5.2, "Preferences"* (

)
- *Section 3.2.5.3, "Refresh"* (

)
- *Section 3.2.5.4, "Page Design Options"* (

)

- *Section 3.2.5.5, “Visual/Source Editors splitting buttons”* ( )
- *Section 3.2.5.6, “Show Border for Unknown Tags”* ( )
- *Section 3.2.5.7, “Show Non-visual Tags”* ( )
- *Section 3.2.5.8, “Show Selection Bar”* ( )
- *Section 3.2.2.1, “Text Formatting”* ( )
- Show bundle's messages as EL expressions ( )

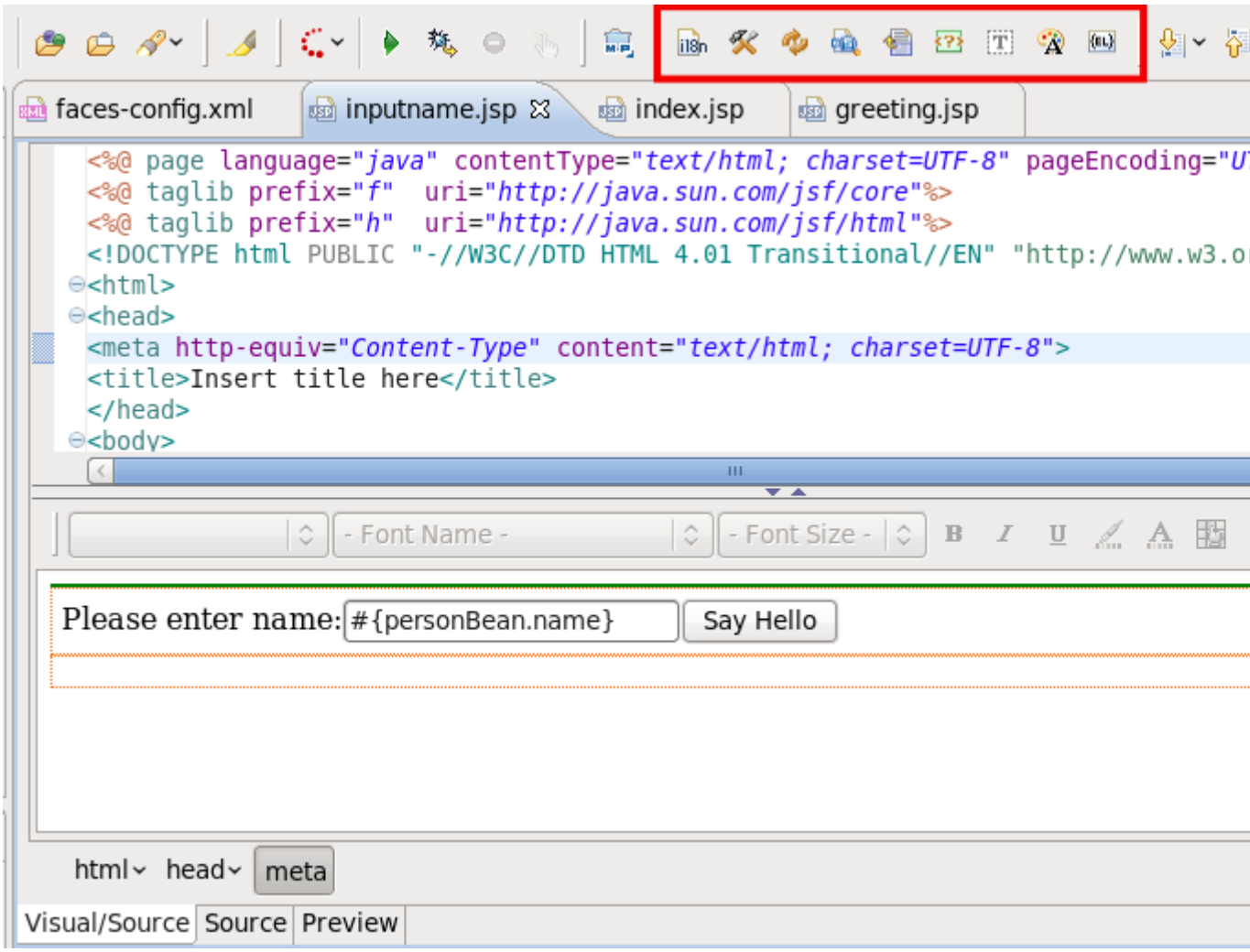

Figure 3.73. Buttons on the VPE Toolbar

3.2.5.1. Externalize string

The **Externalize string** button (

) provides the ability to export a selected string.

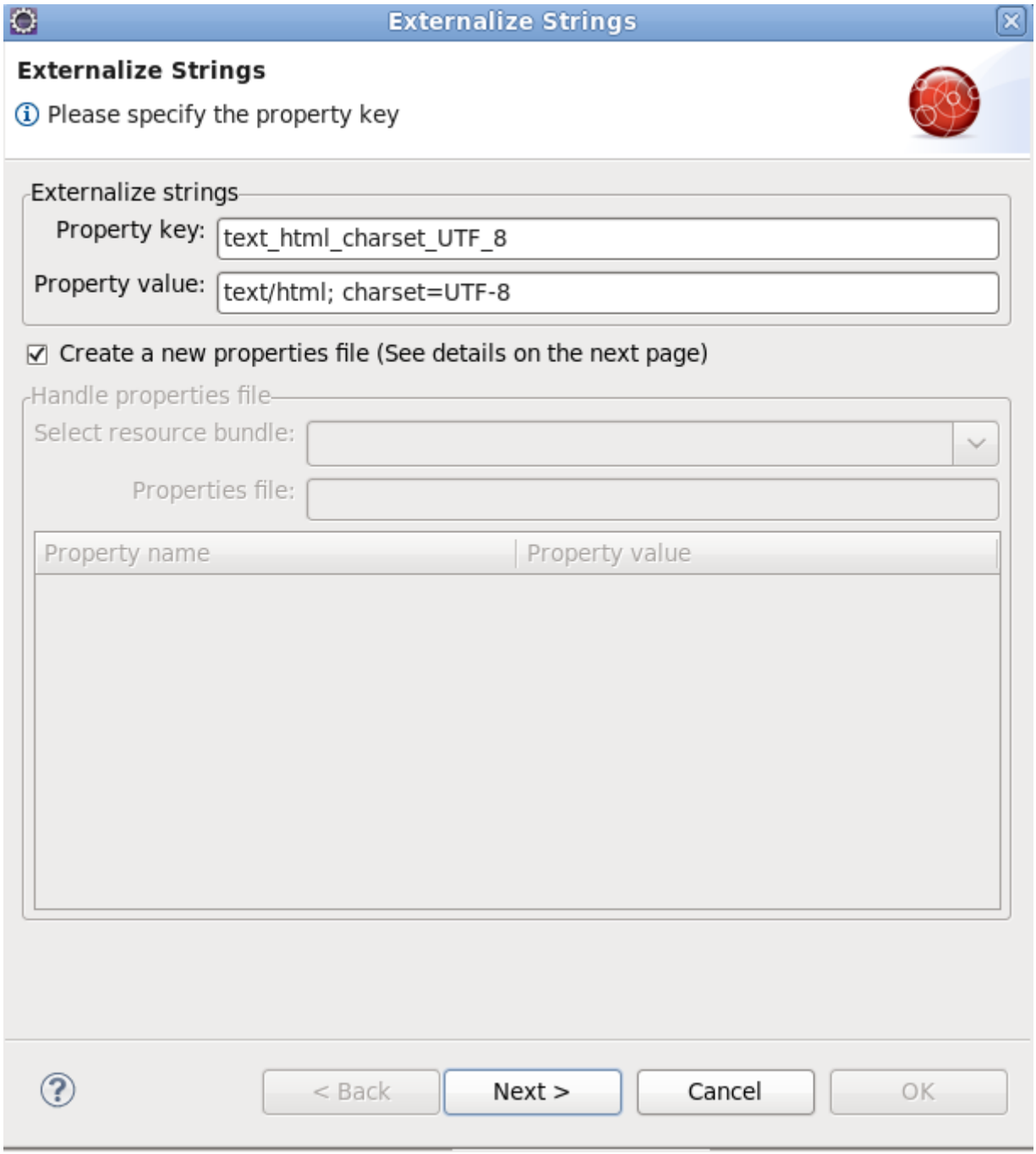

Figure 3.74. Visual Page Editor Externalize string wizard page 1

The first page of the Visual Page Externalize string wizard asks you for the name of the **Property key** you wish to create and the then **Property value** for that key. You then have the choice to either have the wizard generate a new properties file for the string or to select a property file that already exists (if one is available).

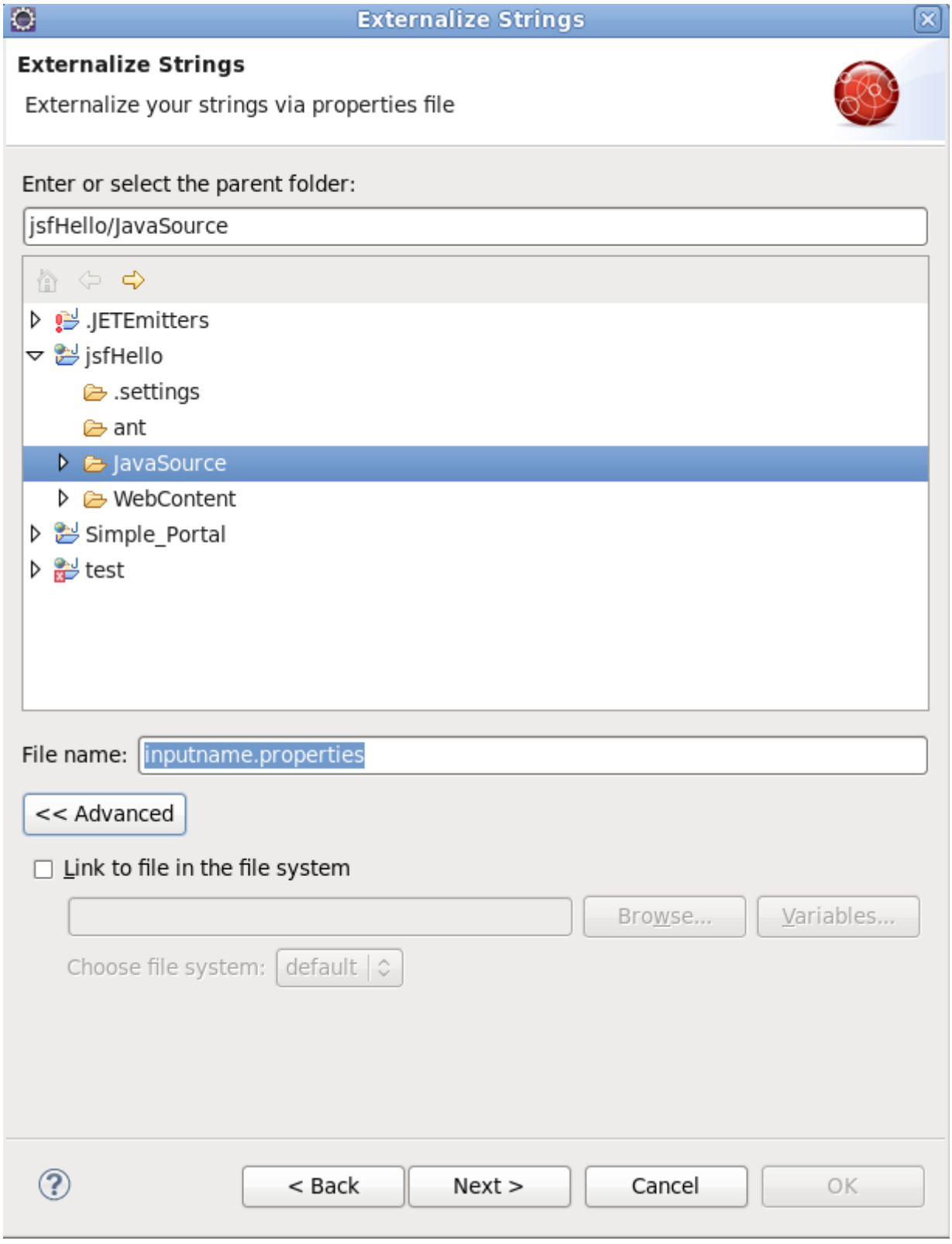

Figure 3.75. Visual Page Editor Externalize string wizard page 2

On the second page of the wizard enter a new or select an existing folder where the property file of the string will be stored. Be sure to also name the property file.

By clicking on the **Advanced** button you will be shown an option to link the property file that will be created, to a file already on your computer. This step is not necessary for externalizing a string.

Figure 3.76. Visual Page Editor Externalize string wizard page 3

The final page asks you to choose a place for the string bundle to be registered. By default the option of **manually by user** will be selected.

3.2.5.2. Preferences

The **Preferences** button ( ) provides quick access to the Visual Page Editor preferences.

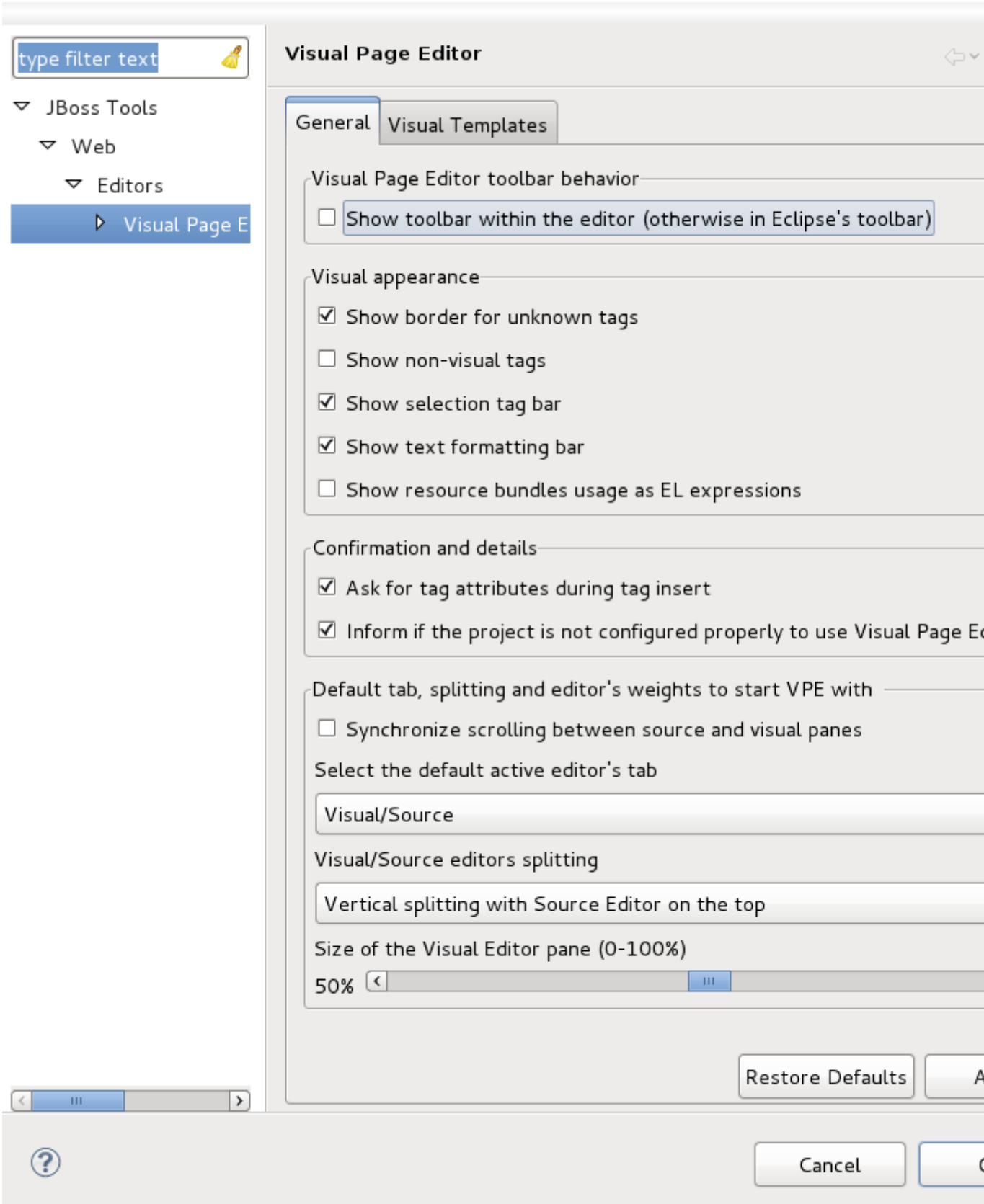

Figure 3.77. Visual Page Editor Preferences Window

This page provides a number of options associated with the editor representation. For more detailed description on each option please read the "JBoss Tools Preferences" chapter under *Section 9.3, "Visual Page Editor"*.

3.2.5.3. Refresh

Clicking on the **Refresh** button ( ) refreshes the displayed information.

3.2.5.4. Page Design Options

The **Page Design Options** button ( ) displays a window which helps you specify necessary references of the resources. It is represented by a window with four tabs. The first one, **Actual Run-Time folders**, is used to replace absolute and relative path values when generating a preview:

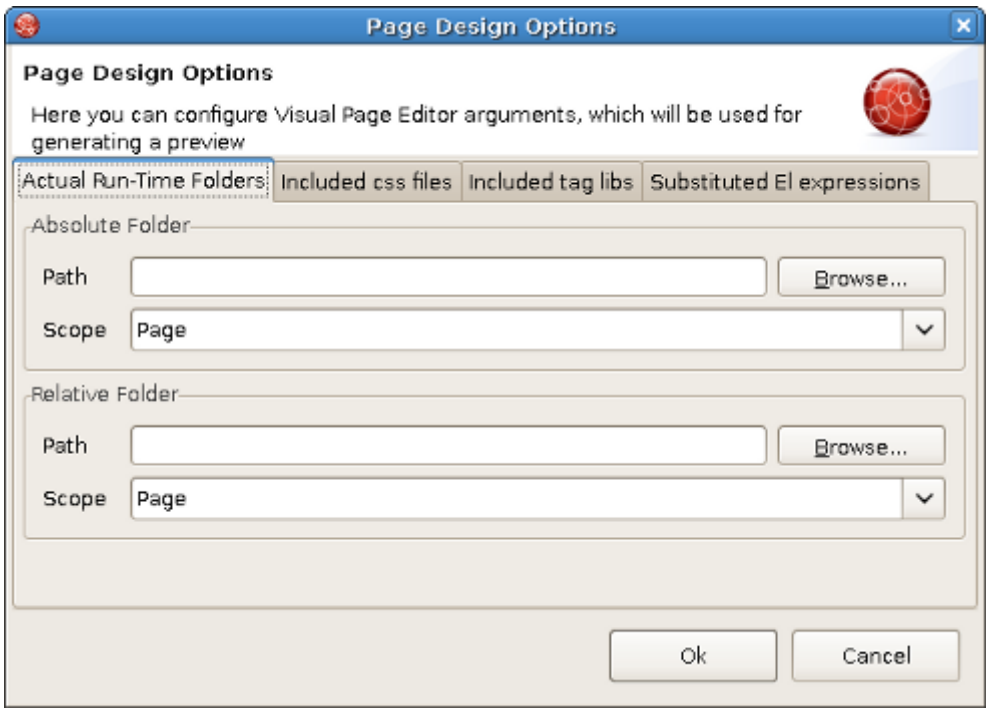

Figure 3.78. Page Design Options: Actual Run-Time folders

The second tab, **Included CSS files**, is used to add CSS files to be linked by Visual Page Editor when generating a preview:

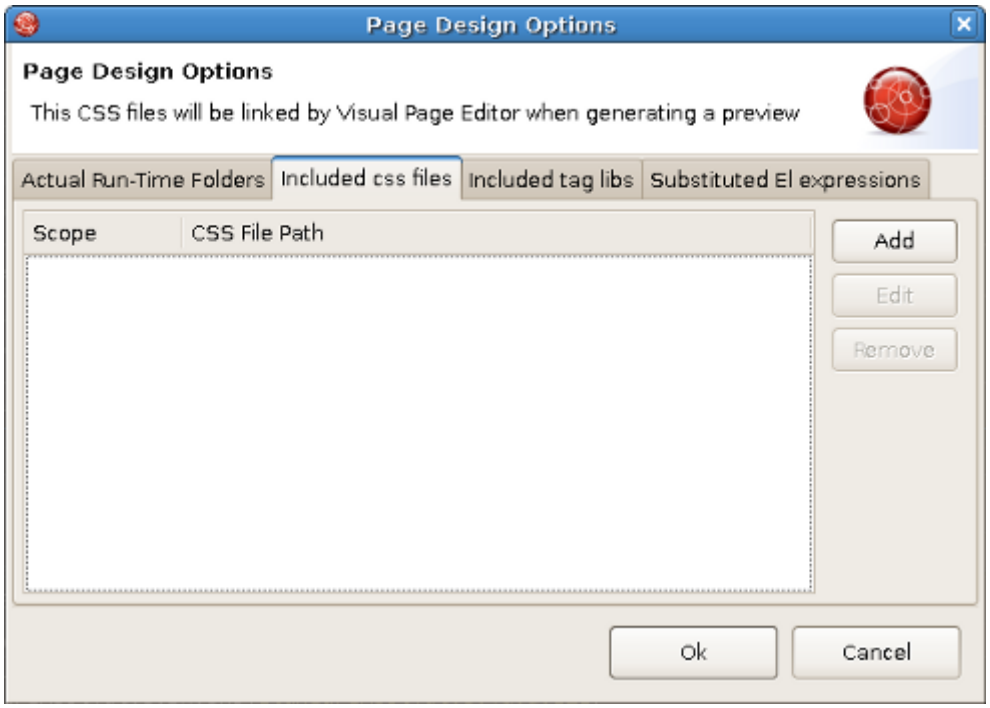

Figure 3.79. Page Design Options: Included CSS files

The third tab, **Included tag libs**, can be used to add Taglibs that can be used by the editor for getting appropriate templates to generate a preview:

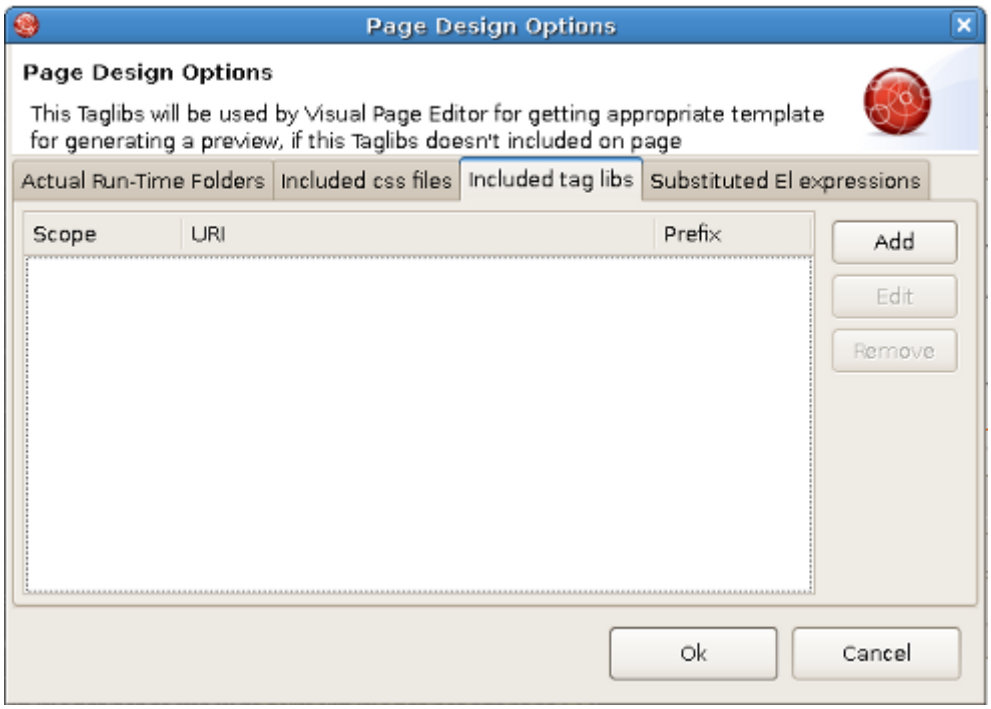

Figure 3.80. Page Design Options: Included tag libs

And finally, the **Substituted El expressions** tab is used to add El expressions that will be substituted by the editor when generating a preview:

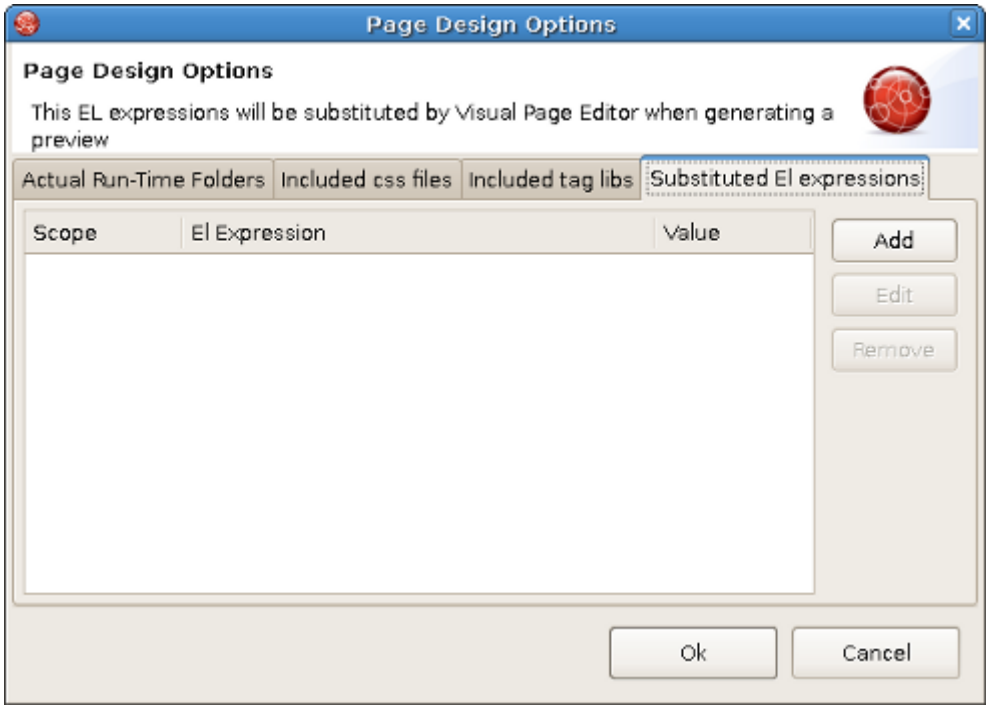

Figure 3.81. Page Design Options: Substituted El expressions

The first two tabs of the window let you define actual runtime folders. The example below will help you understand how this can be done.

Suppose you have the following project structure:

```
WebContent/
  pages/
    img/
       a.gif
    header.jsp
  main.jsp
```

The content of the `header.jsp` is:

```
My Header
<img src="img/a.gif"/>
```

```
```

and `main.jsp` content is:

```
<jsp:include page="pages/header.jsp" />
```

When you open the `main.jsp` file in the Visual Page Editor, it will not be able to resolve the image from the header, however it will work fine in runtime. To fix this in design time, click the **Page Design Options** button and set **Actual Run-Time Relative Folder** to **[Project Name]** → **WebContent** → **pages**, and you will see the image appeared.

Let's consider an example for other tabs. For instance, the definition of your CSS on the page is the next:

```
<link rel="stylesheet" type="text/css"
    href="#{facesContext.externalContext.requestContextPath}/style.css"/>
```

This will work fine in runtime, but the Visual Page Editor does not know the value of `requestContextPath` in design time. In order to see the necessary styles applied in design time your should add a path to your stylesheet in the **CSS File Path** section.

The next URI section lets you add URI taglibs so that the editor knows where to find the tag libraries.

And the last Substituted EL expressions section is provided to specify the values for specific EL variables. It can be useful for a preview generation.

As an example look at the figure below:

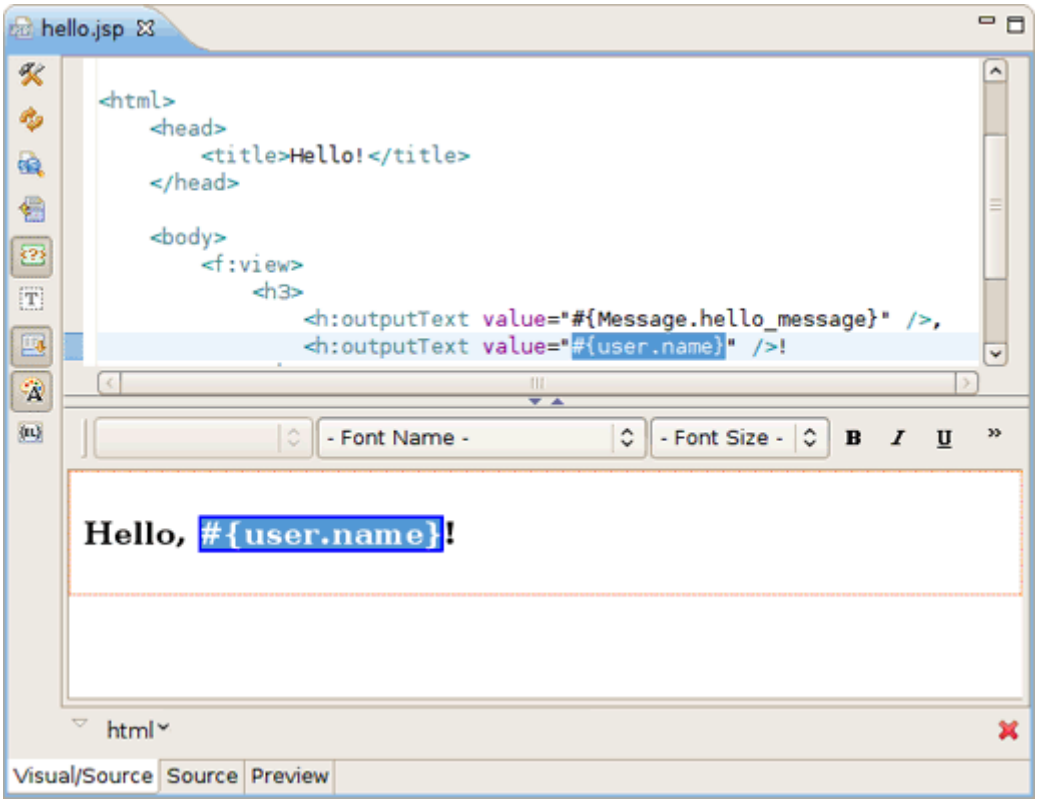

Figure 3.82. EL Expression

Here both in **Source** and **Visual** modes you see the EL expression `#{user.name}`. When you switch to **Preview** view, you will also see this expression. Now click the **Page Design Options** button and set the value for the **user.name** as **World**.

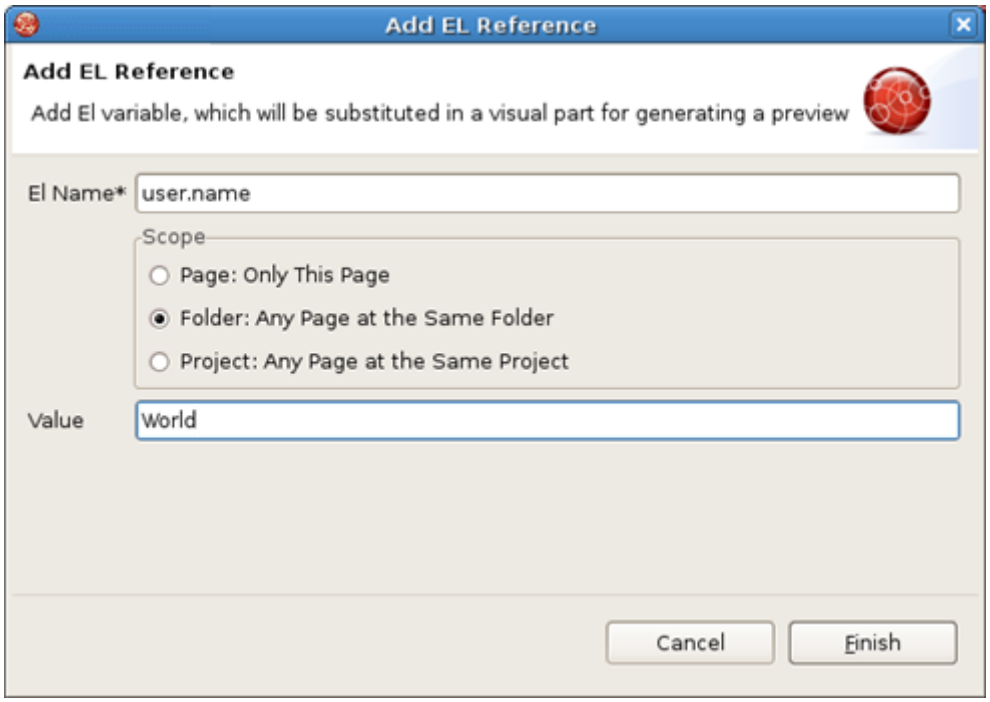

Figure 3.83. Setting the Value for the EL Expression

As a result in **Visual** mode and **Preview** view the word **World** is displayed.

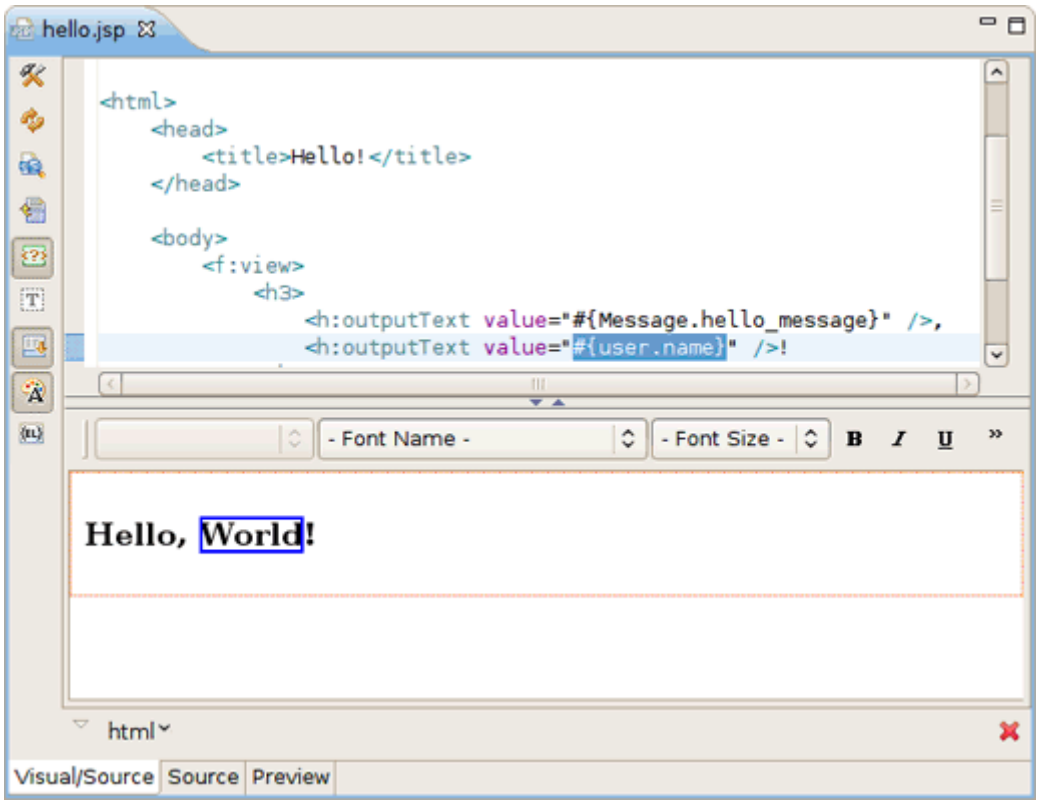

Figure 3.84. The EL Expression Value

3.2.5.5. Visual/Source Editors splitting buttons

The **Visual/Source** Editors splitting buttons provide a way to choose one of the four possible layouts for the **Visual/Source** Editor.

The available layouts and corresponding buttons are as follows:

- Vertical Source on top( )
- Vertical Visual on top ( )
- Horizontal Source to the left ( )
- Horizontal Visual to the left ( )

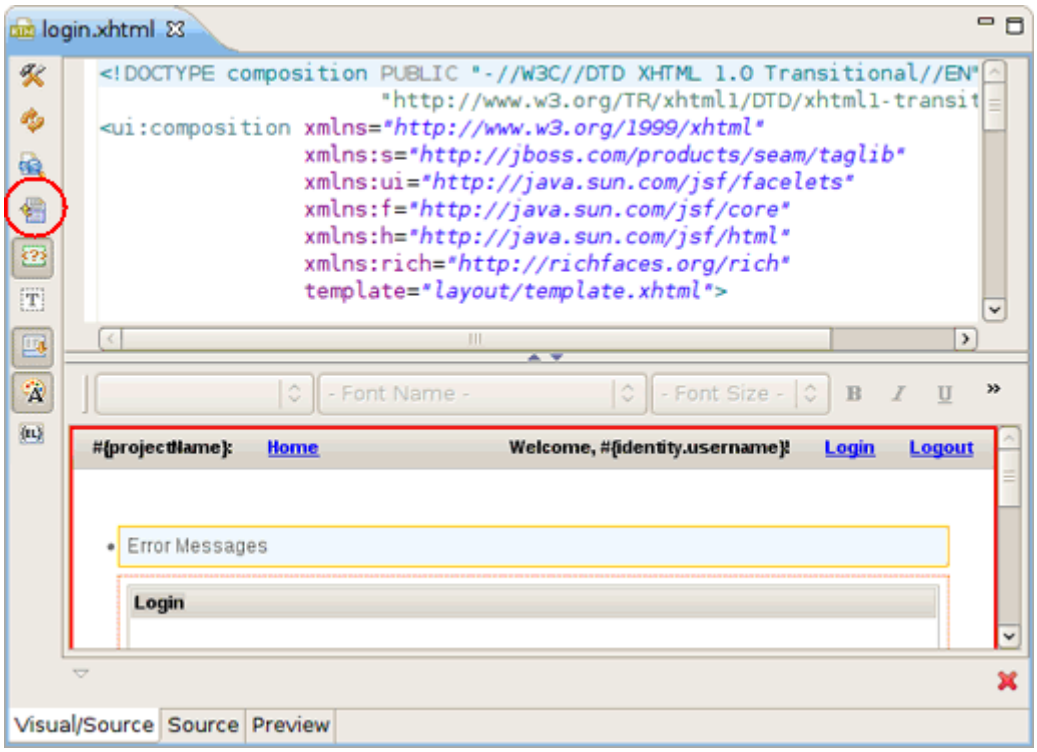

Figure 3.85. Visual Page Editor Before Layout Changing

Note, with the current view there is only *one* button, which provides the ability to move the **Source** and the **View** *in a clockwise direction*.

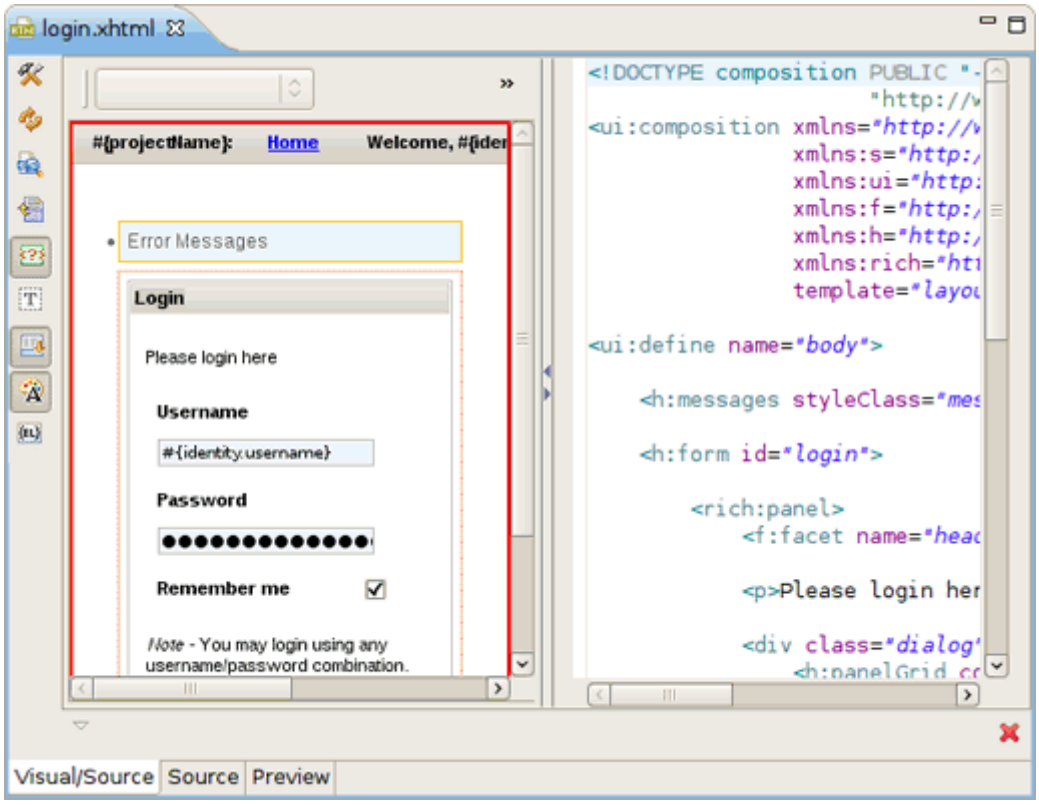

Figure 3.86. Visual Page Editor After Layout Changing

3.2.5.6. Show Border for Unknown Tags

The **Show border for unknown tags** button ( ) will display unknown tags in a border in the **Visual** section of the Visual Page Editor.

3.2.5.7. Show Non-visual Tags

The **Show non-visual tags** button ( ) will display non-visual tags in the **Visual** section of the Visual Page Editor.

In the figure you can see that the non-visual elements are displayed with gray dashed borders.

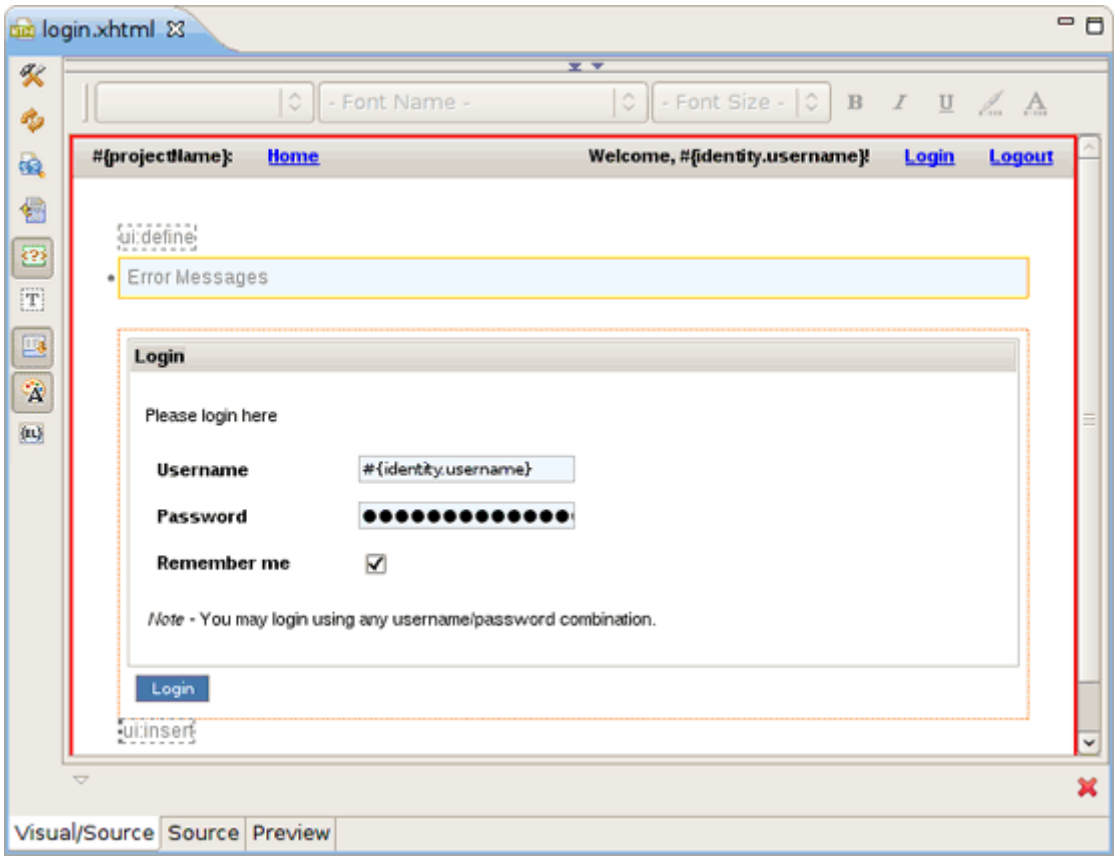

Figure 3.87. Non-visual Tag in the VPE

You can also switch on this option in the Visual Page Editor preferences, having clicked on the **Preferences** button ( ).

3.2.5.8. Show Selection Bar

At the bottom of the **Visual/Source** view there is a **Selection Tag Bar**. It's updated automatically, allowing you to see tags tree for a current component selected in Visual or Source mode. It also allows you to select tags parent and child tags.

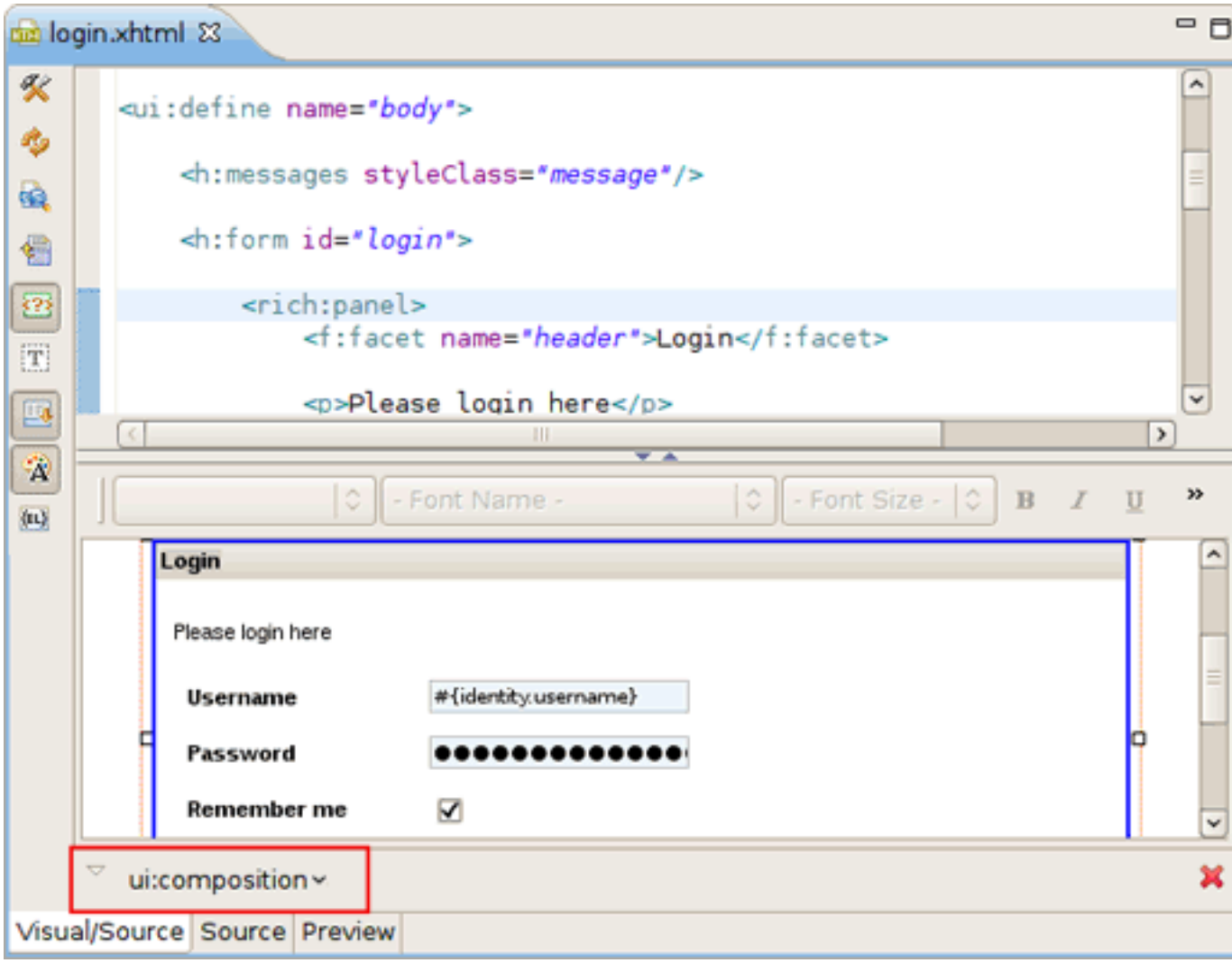

Figure 3.88. Selection Tag Bar

If you want to hide the **Selection Tag Bar**, use the **Show Selection Bar** button ( ) on the Visual Page Editor toolbar.

3.2.6. Page Preview

The Visual Page Editor comes with a design-time preview feature, which is available for:

- JSF Pages
- Seam Pages

Preview view is read-only, and it shows how the page will look like in a browser.

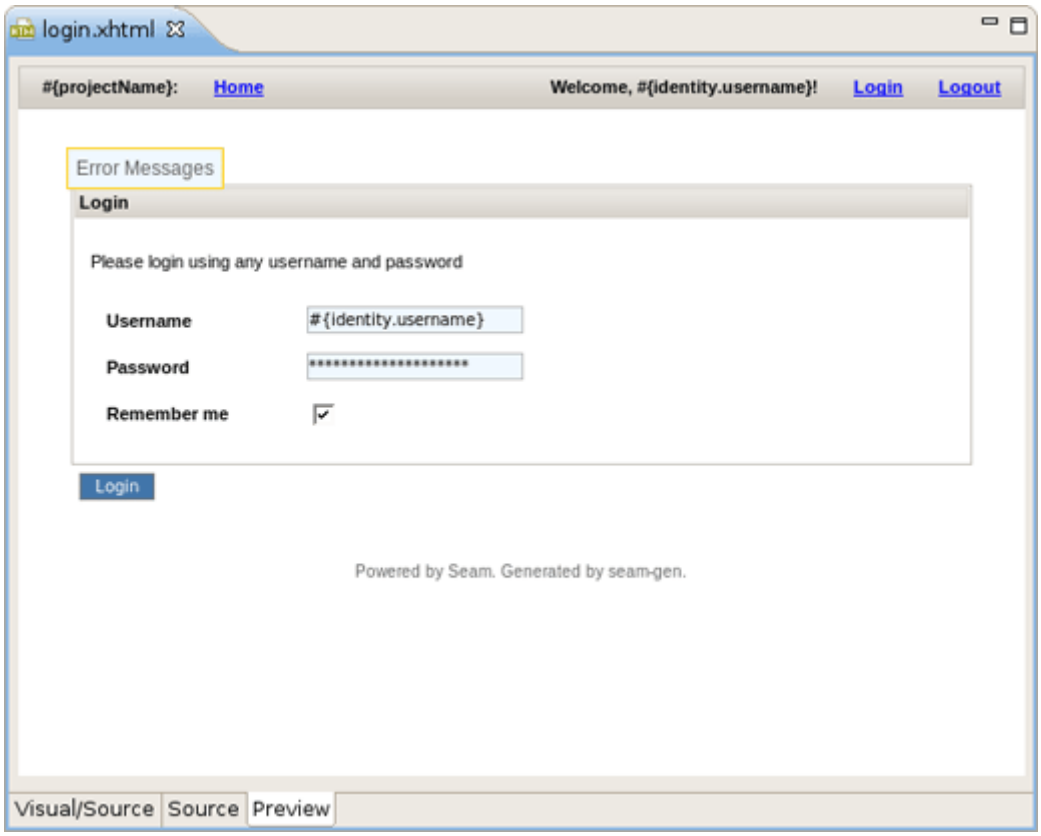

Figure 3.89. Preview View

3.2.7. Error Messages

The Visual Page Editor provides user friendly and effective error messages, which should make solving problems easier. The error messages contains a reference to the problem and its description. Also in the **Error** area you can find a link to Visual Page Editor forum and a **Details** button which is used to see a error trace.

If the error occurs while the editor is loading, the error message will contain information about of what might have caused the error (e.g. a missing library or errors in source code).

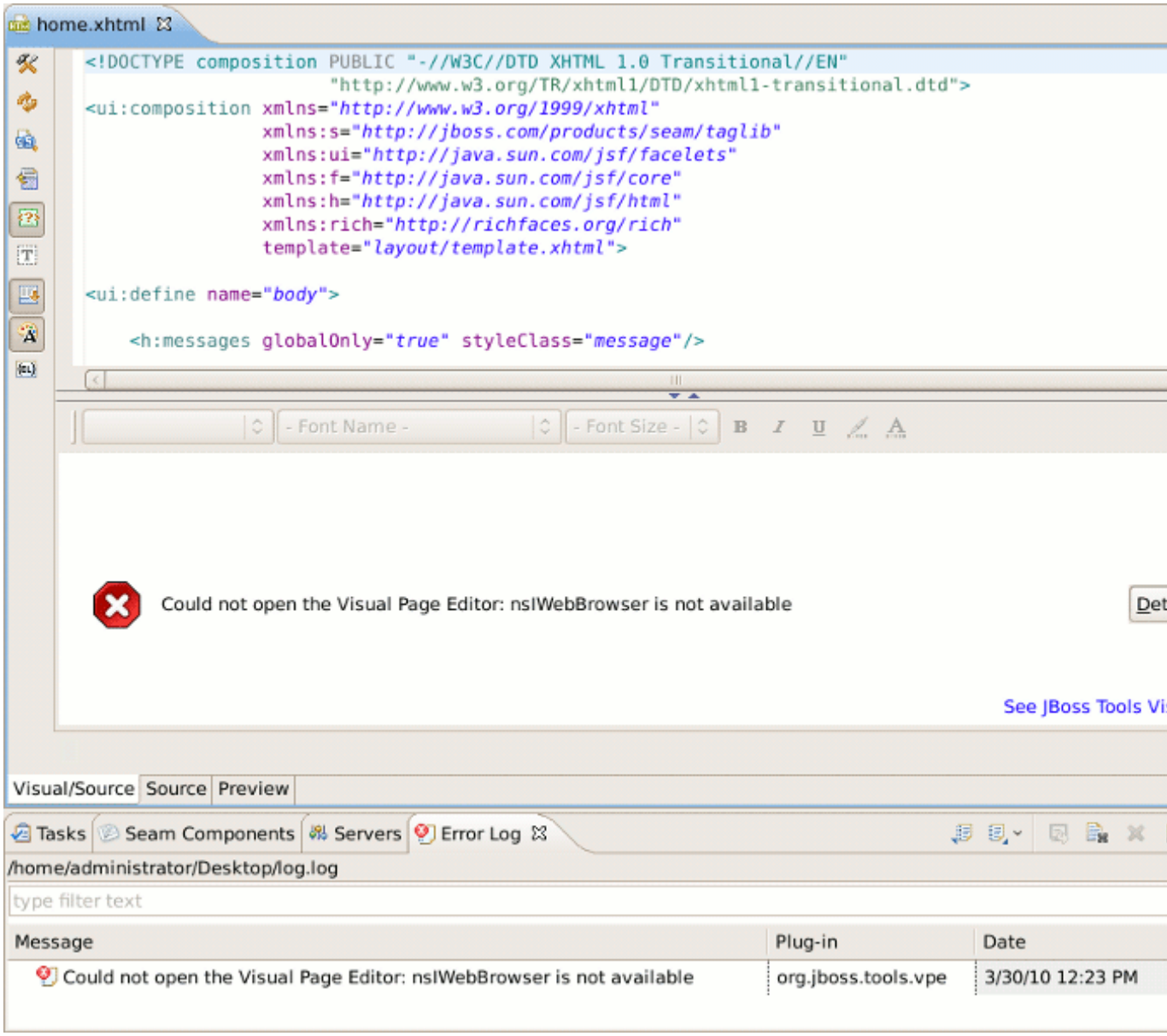

Figure 3.90. Visual Page Error Message

3.2.8. Support for Custom Facelets Components

Visual Page Editor supports custom Facelets tag libraries both declared in the `web.xml` file (for details, see *Creating a component* [http://www.ibm.com/developerworks/java/library/j-facelets/#N10294]) and packed into a JAR file.

Tip:

In case of Facelets tag library packed in `.jar`, remember to put `*.taglib.xml` in right place: `[filename].jar/META-INF/*.taglib.xml`

Visual Page Editor recognizes the tags from the custom Facelets tag library and correctly renders them both in source and visual view of the editor.

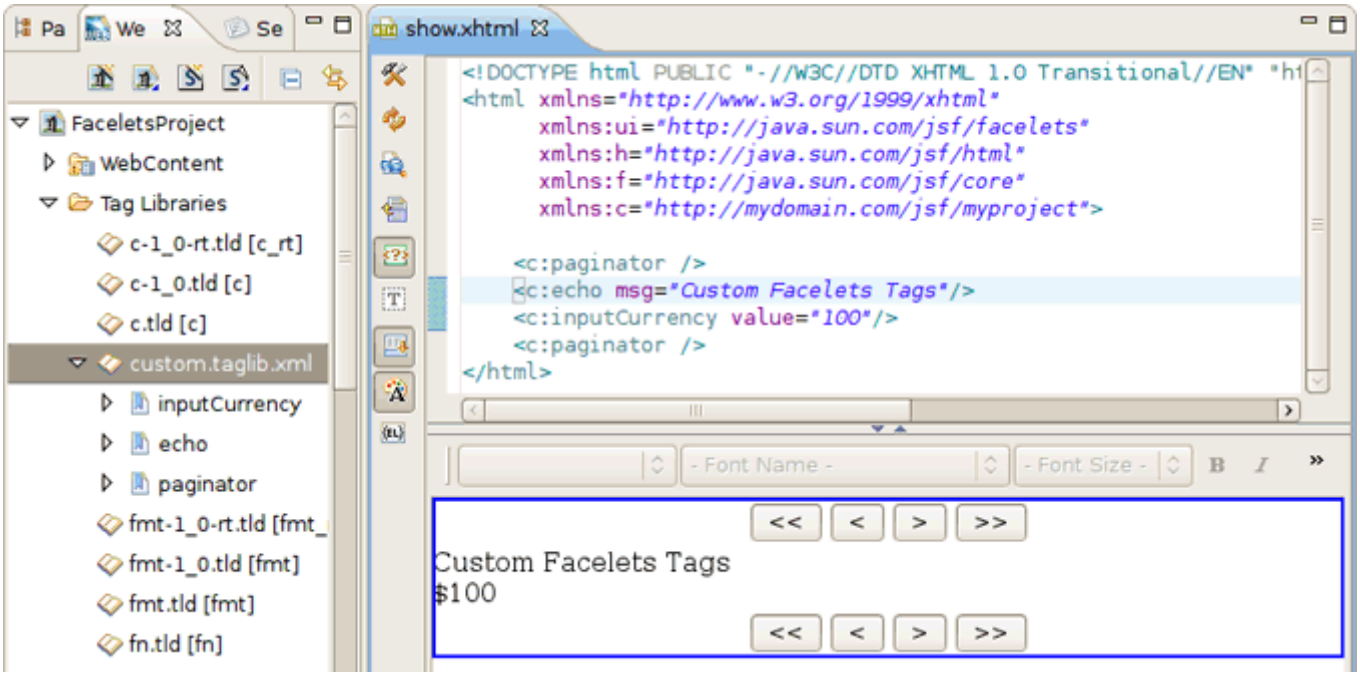

Figure 3.91. Custom Facelets Tags in the VPE

While editing an `XHTML` file that uses a custom Facelets components you can always make use of the following editor's features:

- *Section 3.2.8.1, "Content Assist for Custom Facelets Components"*
- *Section 3.2.8.2, "OpenOn for Custom Facelets Components"*

3.2.8.1. Content Assist for Custom Facelets Components

Call the content assist as usual by using **Ctrl**+**Space** when typing a tag. You should see the custom Facelets tags defined in your Facelets tag library listed as proposals.

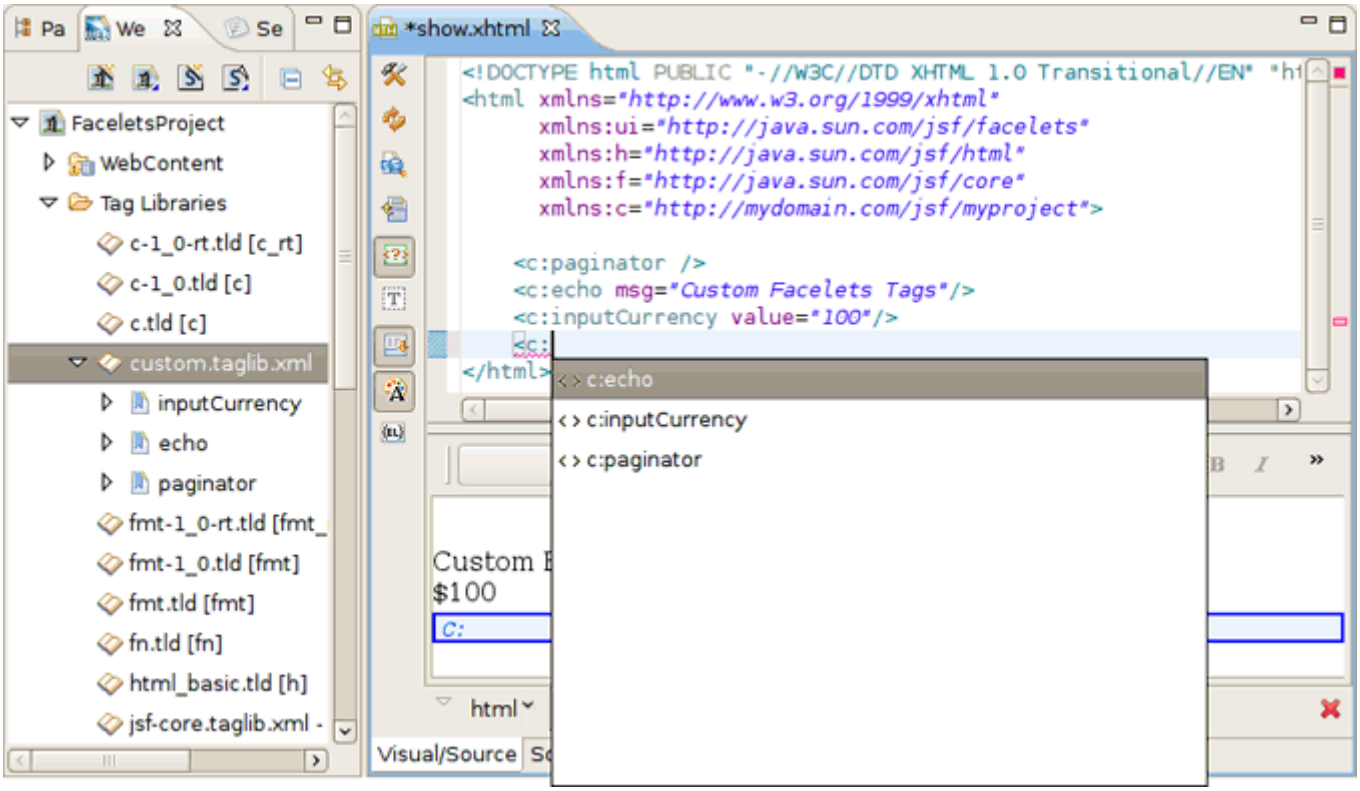

Figure 3.92. Content Assist for Custom Facelets Tags

3.2.8.2. OpenOn for Custom Facelets Components

While developing using Facelets you can make use of:

- *Section 3.2.8.2.1, “OpenOn in XHTML Files That Use Custom Facelets Components”*
- *Section 3.2.8.2.2, “OpenOn in Custom Facelets Tag File (*.taglib.xml)”*

3.2.8.2.1. OpenOn in XHTML Files That Use Custom Facelets Components

OpenOn functionality in **XHTML** files is available in two views of the Visual Page Editor:

1. Source view

   Using the **Ctrl**+**Click** keyboard shortcut on the namespace will open the Facelets tag file in a separate window.

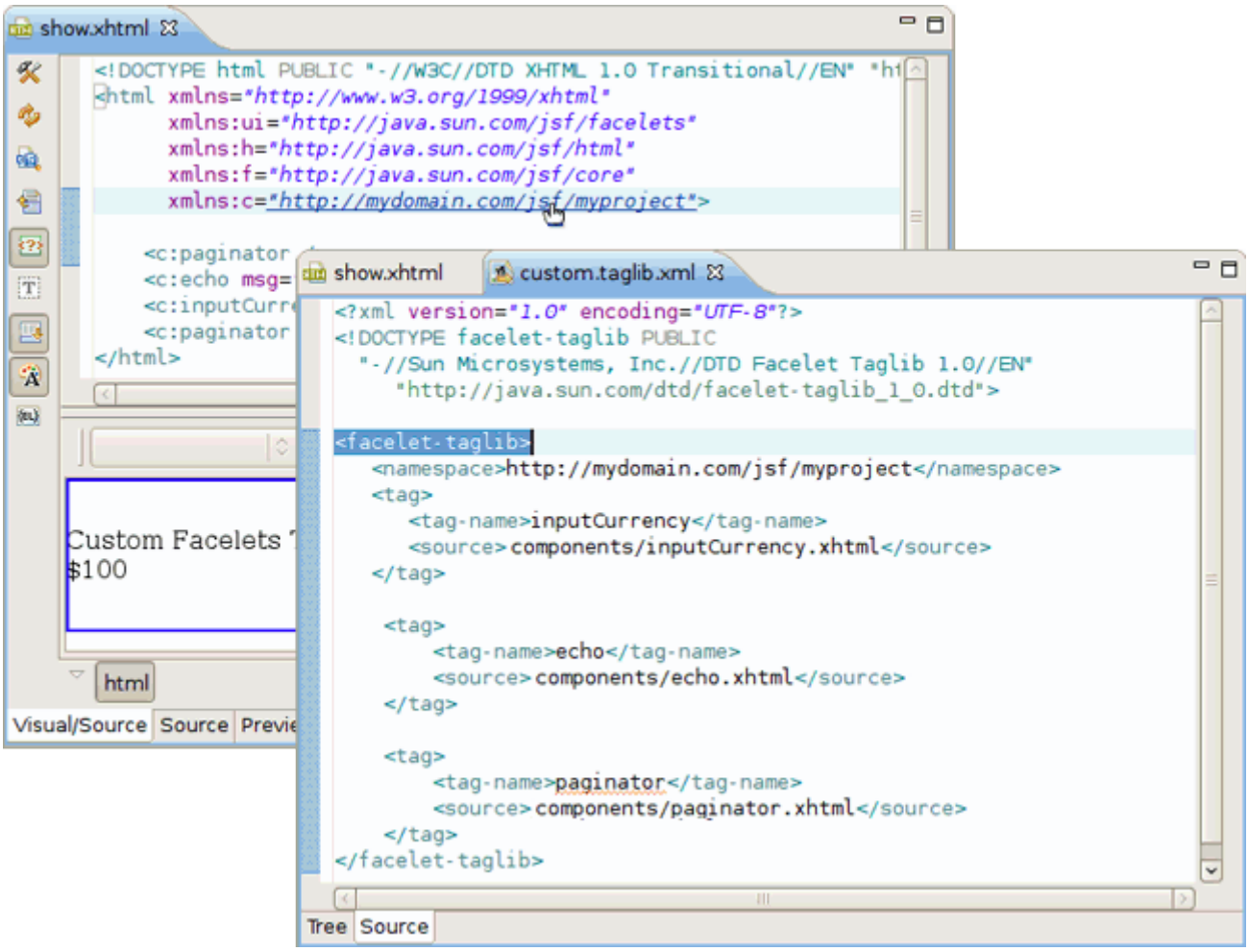

Figure 3.93. Opening a Custom Facelets Tag File

Using the **Ctrl**+**Click** keyboard shortcut on any custom Facelets tag declared on the page will do the same. The selected tag will be highlighted in the opened file.

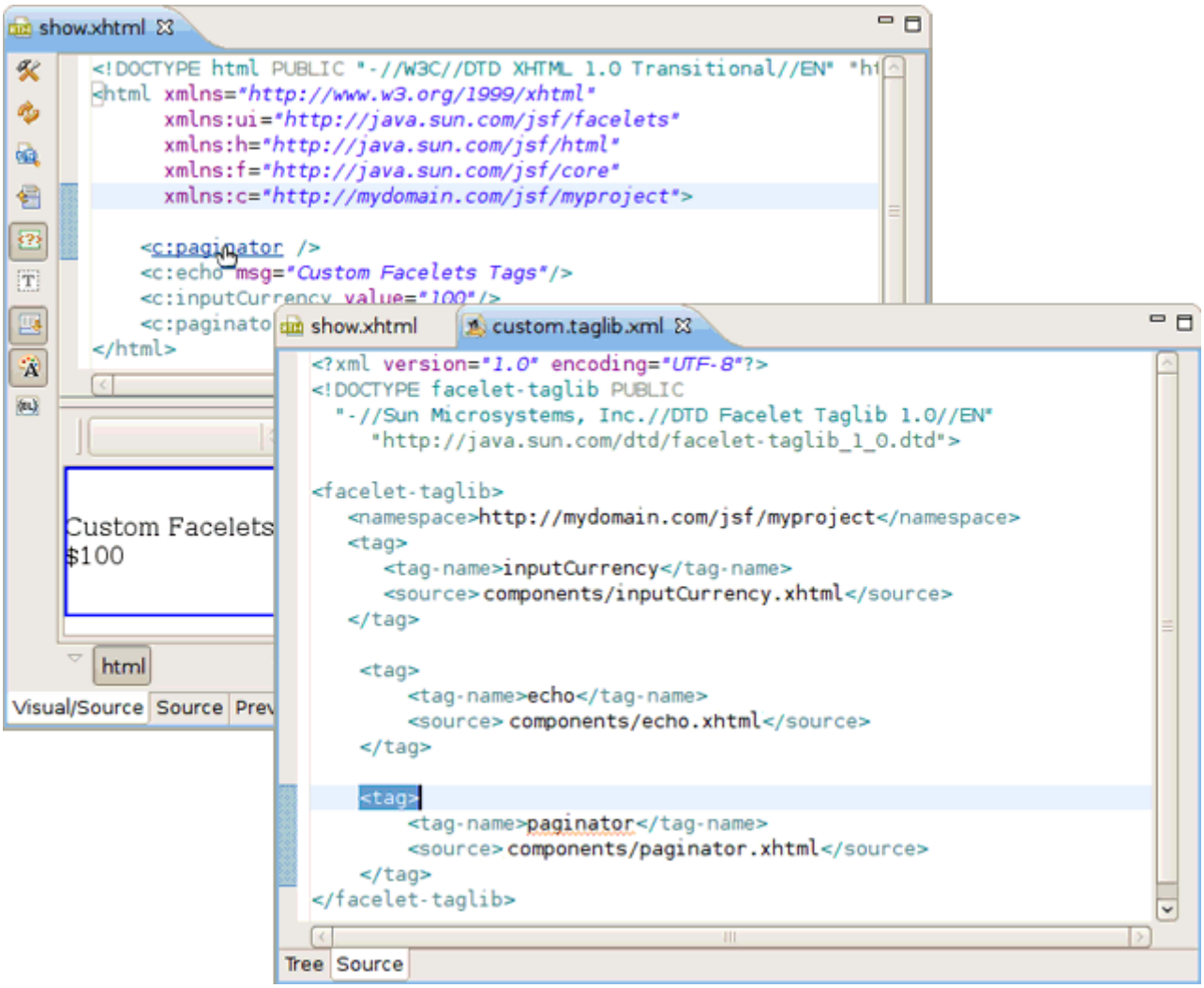

Figure 3.94. Opening a Custom Facelets Tag File

2. Visual view

   In the visual view of the Visual Page Editor, double-click a custom component and the Facelets tag file (`*.taglib.xml`) where it is declared will be opened.

3.2.8.2.2. OpenOn in Custom Facelets Tag File (`*.taglib.xml`)

Using the **Ctrl**+**Click** keyboard shortcut on the path to source of the Facelets tag will open the component in its own editor.

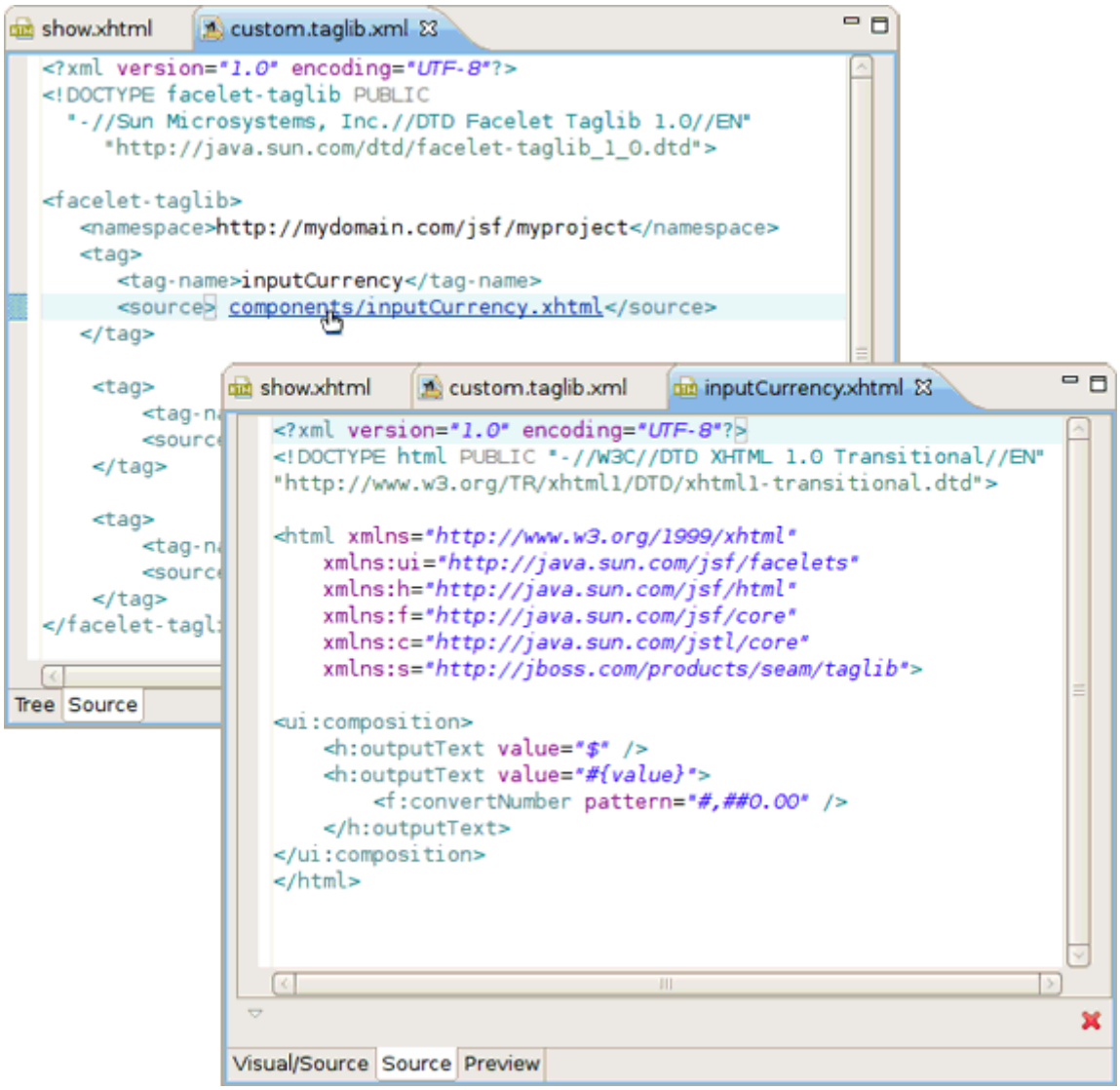

Figure 3.95. Opening a Custom Facelets Component

3.3. More Editors

Besides Visual Page Editor JBDS is supplied with a huge range of various editors for different file types: properties, TLD, `web.xml`, tiles and so on.

3.3.1. Graphical Properties Editor

The **Properties** editor allows you to work in two different modes and also supports unicode characters.

To create a new properties file in the **Package Explorer** view, select **New** → **Properties File** from the right-click context menu on the folder where you want to create the file.

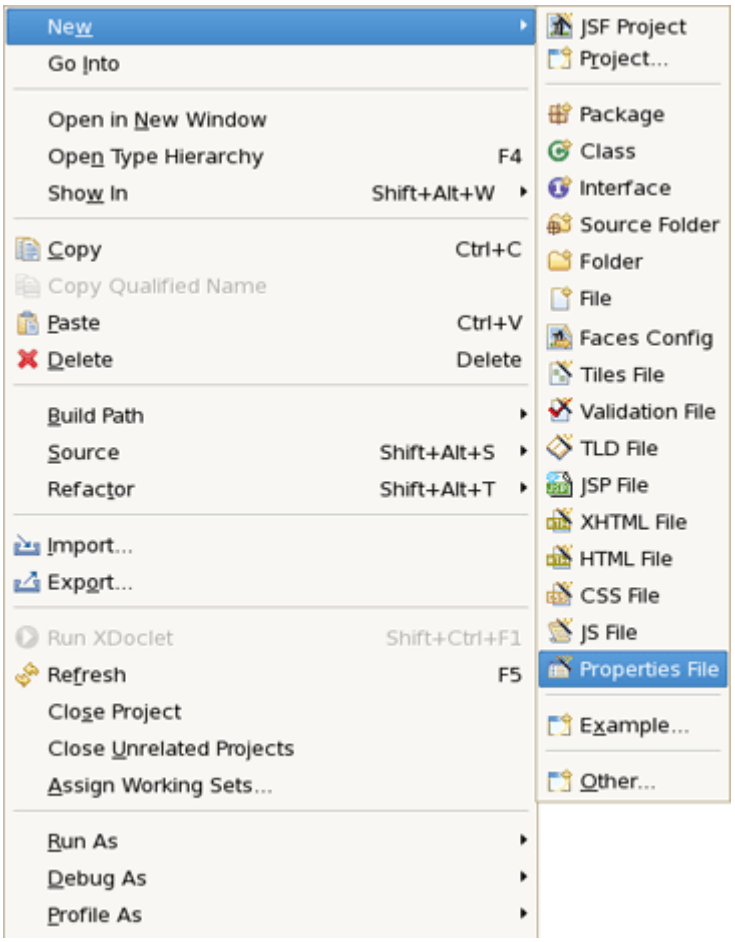

Figure 3.96. Selecting Properties File

You can edit the file using a table-oriented "Properties" viewer:

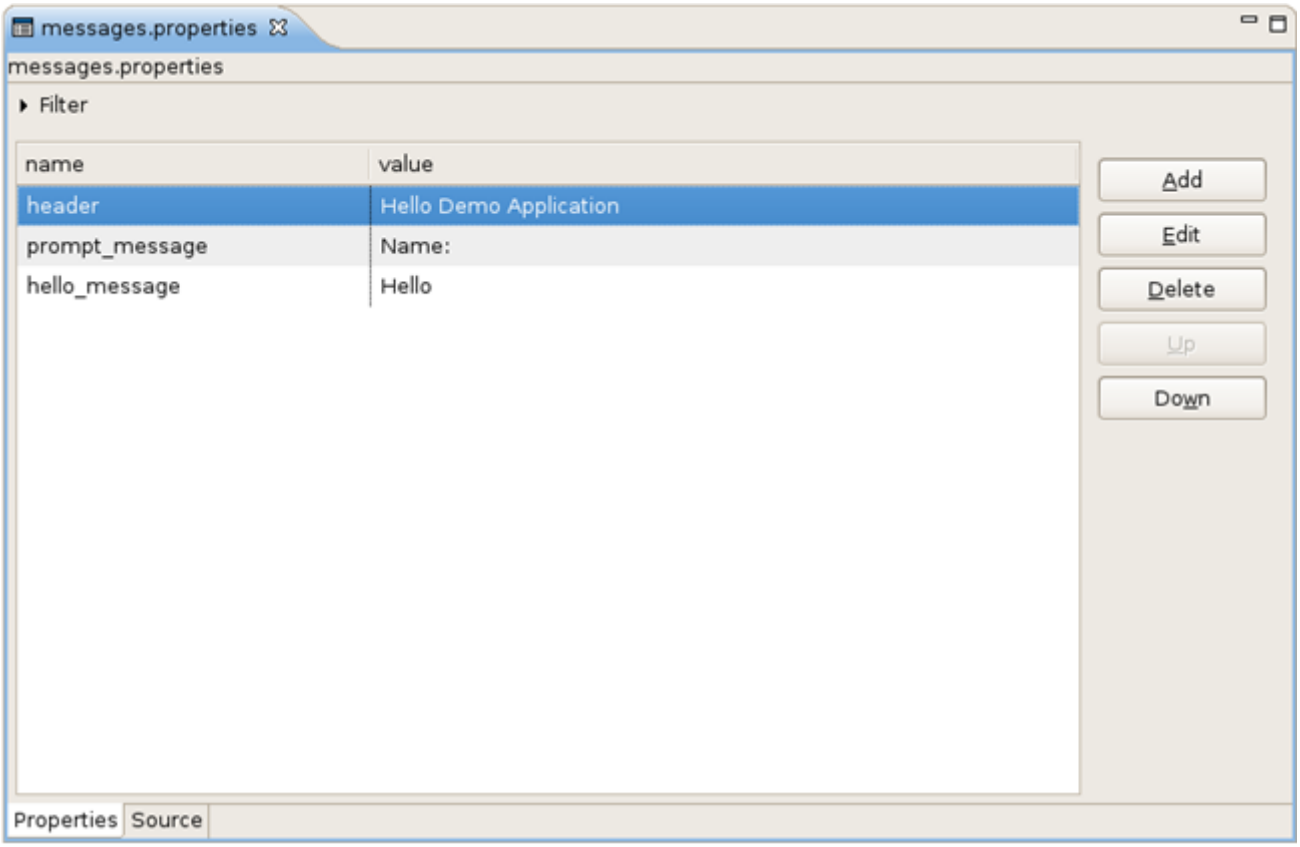

Figure 3.97. "Properties" Viewer

You can also use a **Source** viewer for editing the file:

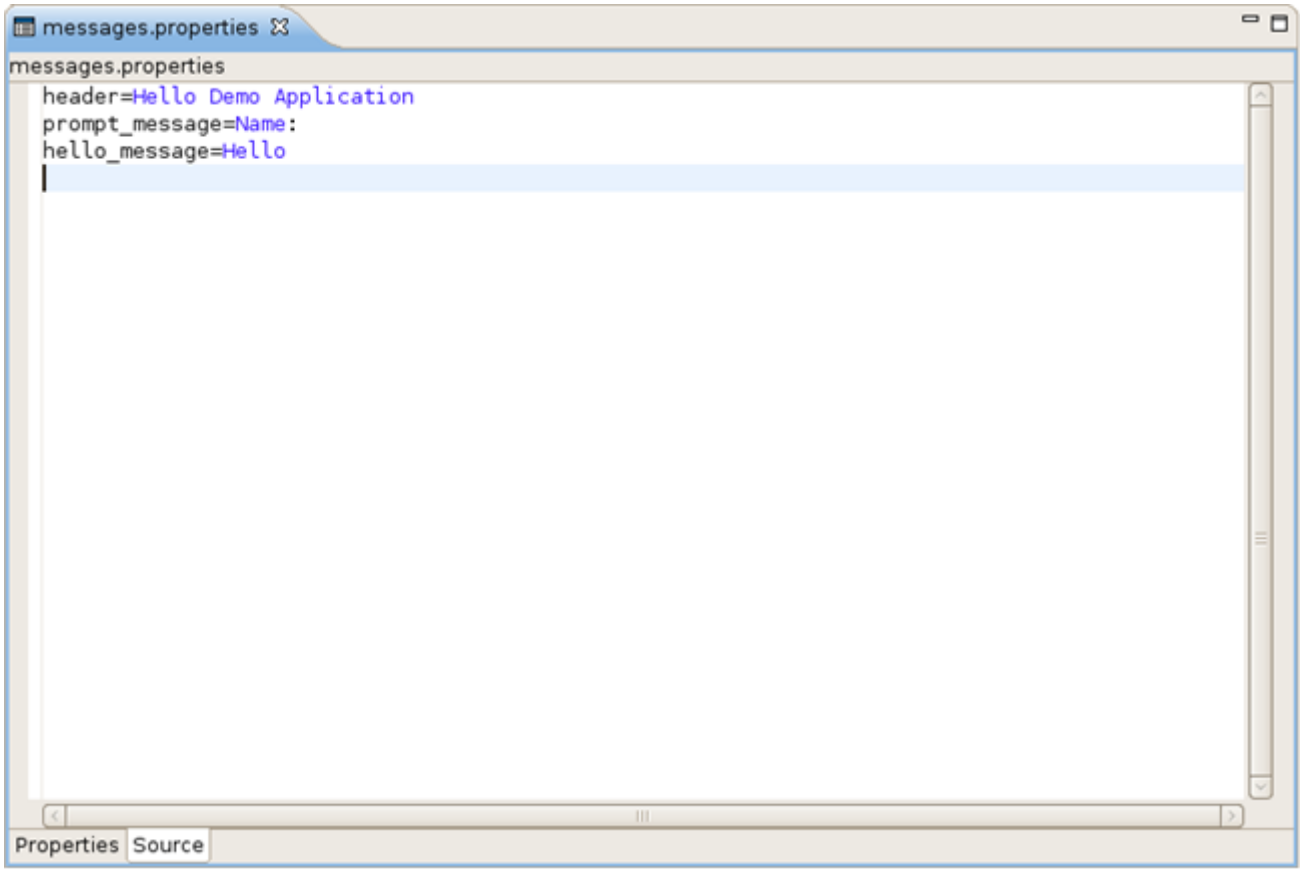

Figure 3.98. Source Viewer

3.3.2. Graphical Tag Library Editor

The **Tag Library Editor** comes with same features you will find in all other JBoss Developer Studio editors:

- Graphical and source edit modes
- Validation and error checking

3.3.2.1. Tree view

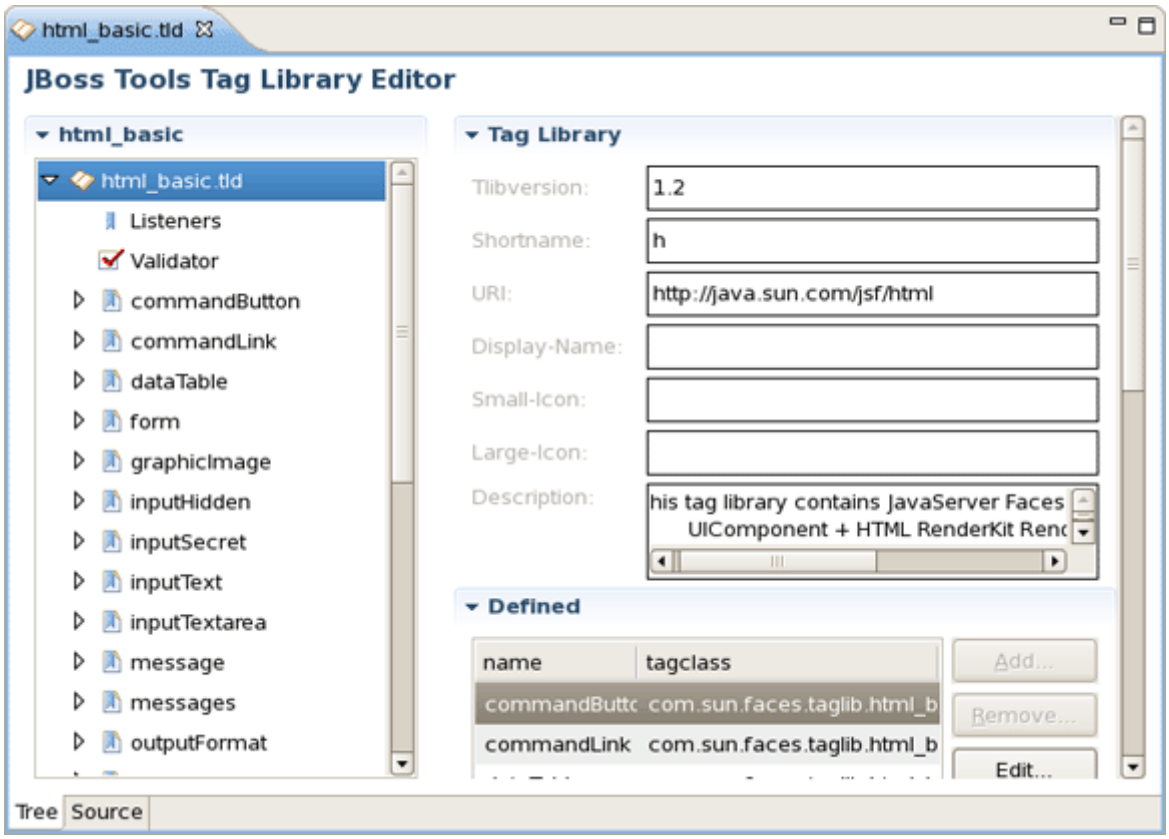

Figure 3.99. Tree View

3.3.2.2. Source view

You can easily switch from **Tree** to **Source** by selecting the **Source** tab at the bottom of the editor.

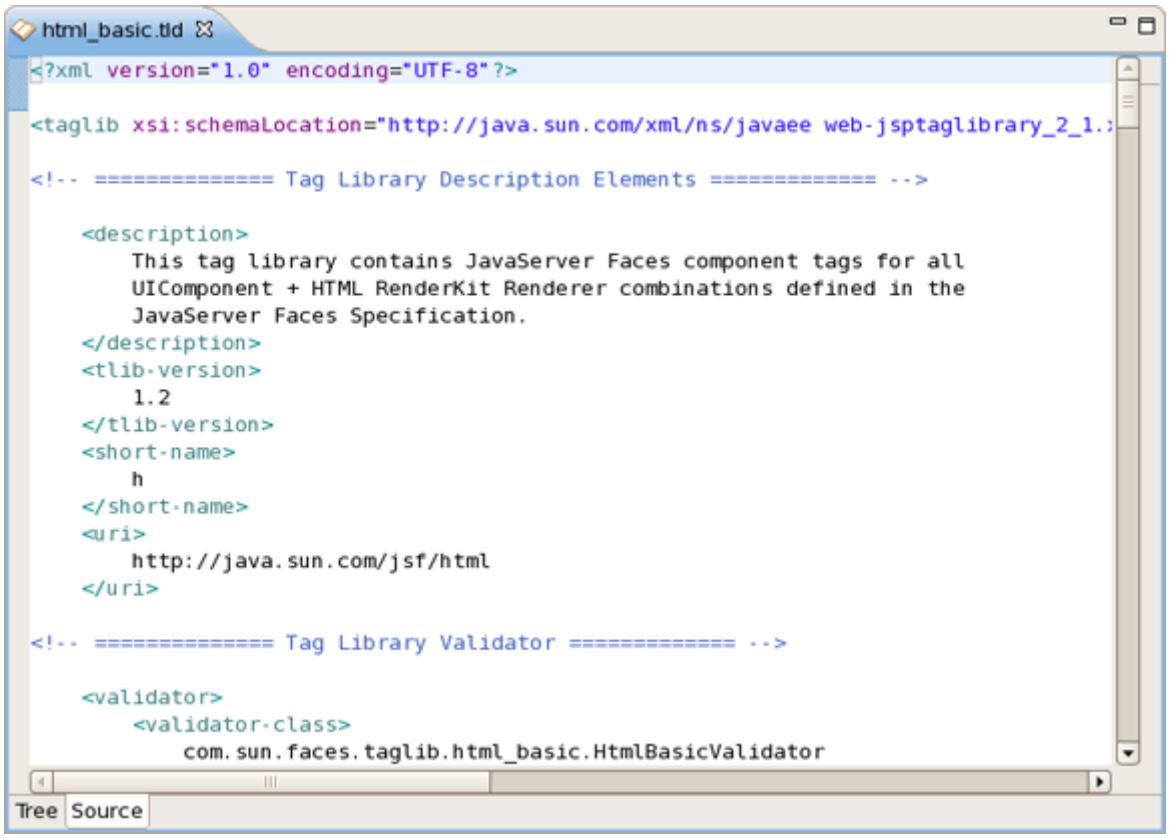

Figure 3.100. Source View

You can easily add a new tag:

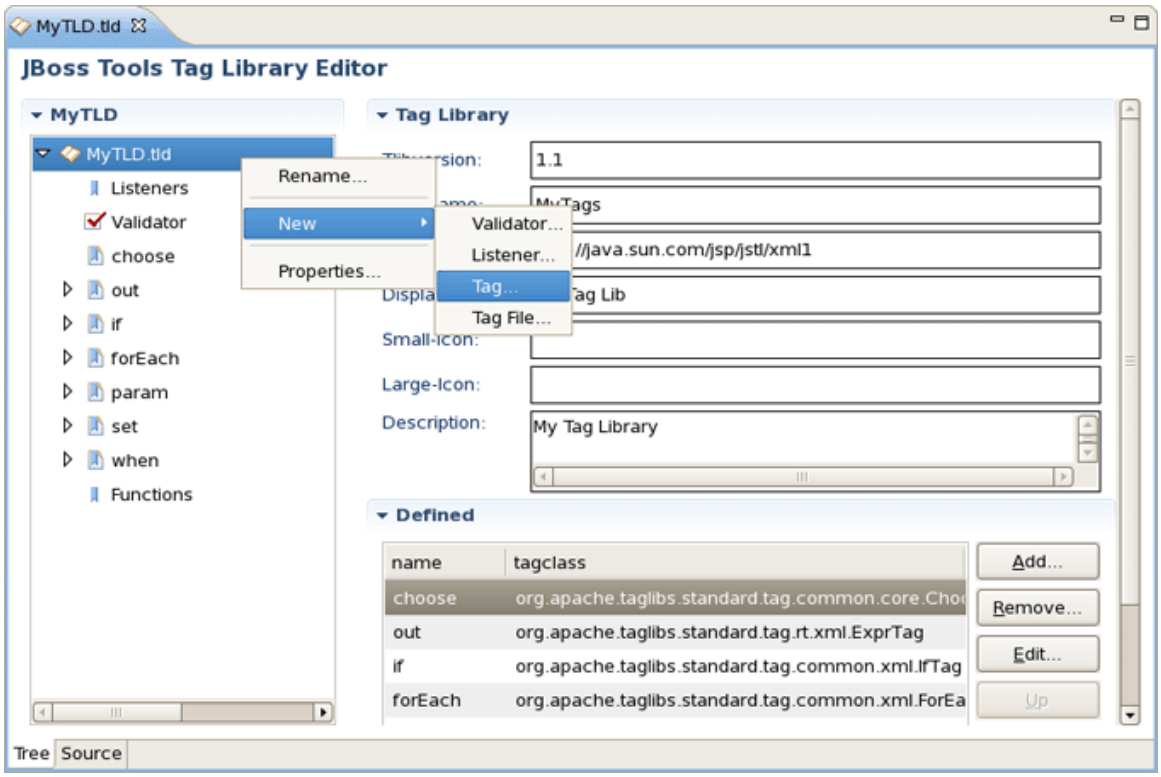

Figure 3.101. Adding a New TLD Tag

You can also easily add a new attribute to an existing tag:

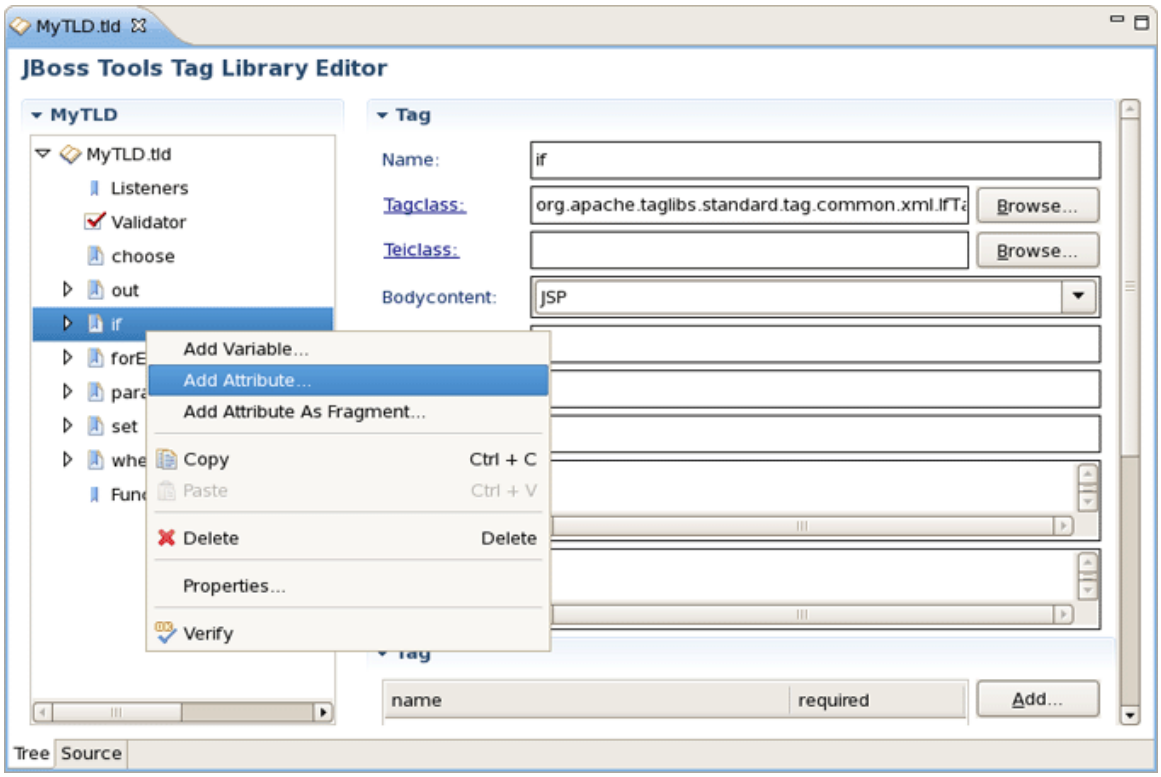

Figure 3.102. Adding a New Attribute to TLD tag

Content assist is available when editing the file using the **Source** viewer:

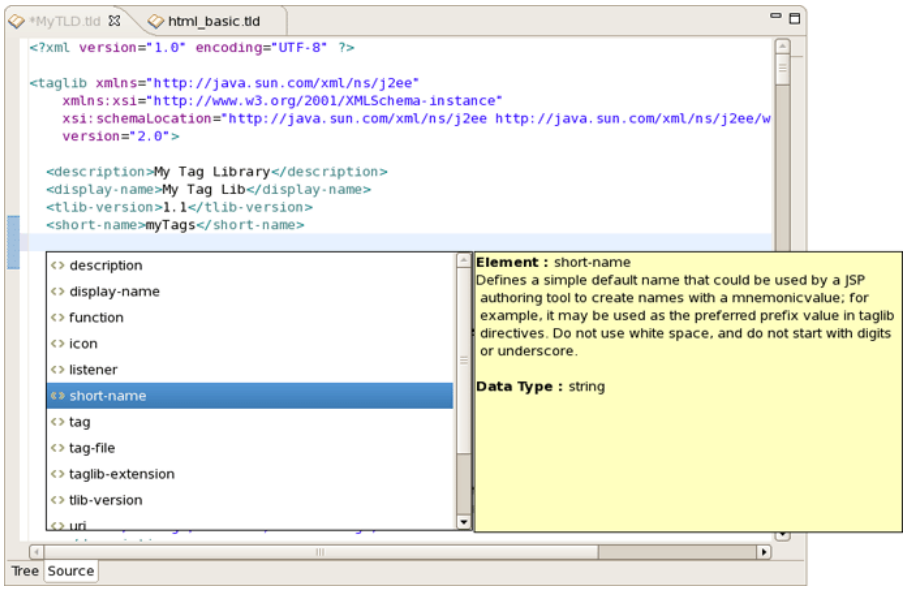

Figure 3.103. TLD Content Assist

In the **Source** viewer, if at any point a tag is incorrect or incomplete, an error will be indicated next to the line and also in the **Problems** view below.

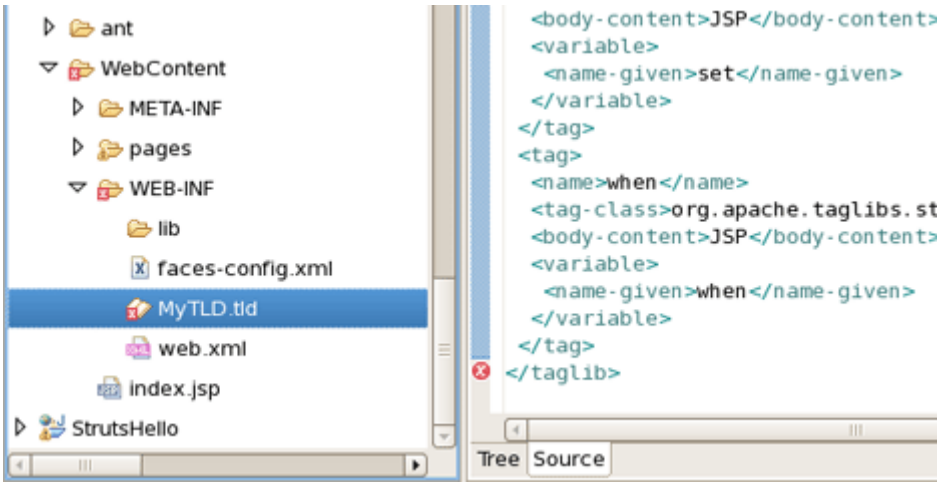

Figure 3.104. Error Reporting

3.3.3. Graphical Web Application File (web.xml) Editor

The deployment descriptor `web.xml` file is intended for describing the servlets, container-managed security constraints and various deployment properties specific for your web application.

To edit the deployment descriptor JBoss Developer Studio provides its own `web.xml` editor that comes with the same features you will find in all other JBoss Developer Studio editors:

- Graphical and source edit modes
- Validation and error checking

3.3.3.1. Tree View

Switch to the **Tree** view if you want to edit the `web.xml` file in a graphical mode. All elements that `web.xml` could include are located in the left area of the editor in a tree format. Click a node on the left to display and edit its properties, which will appear in the right-hand area.

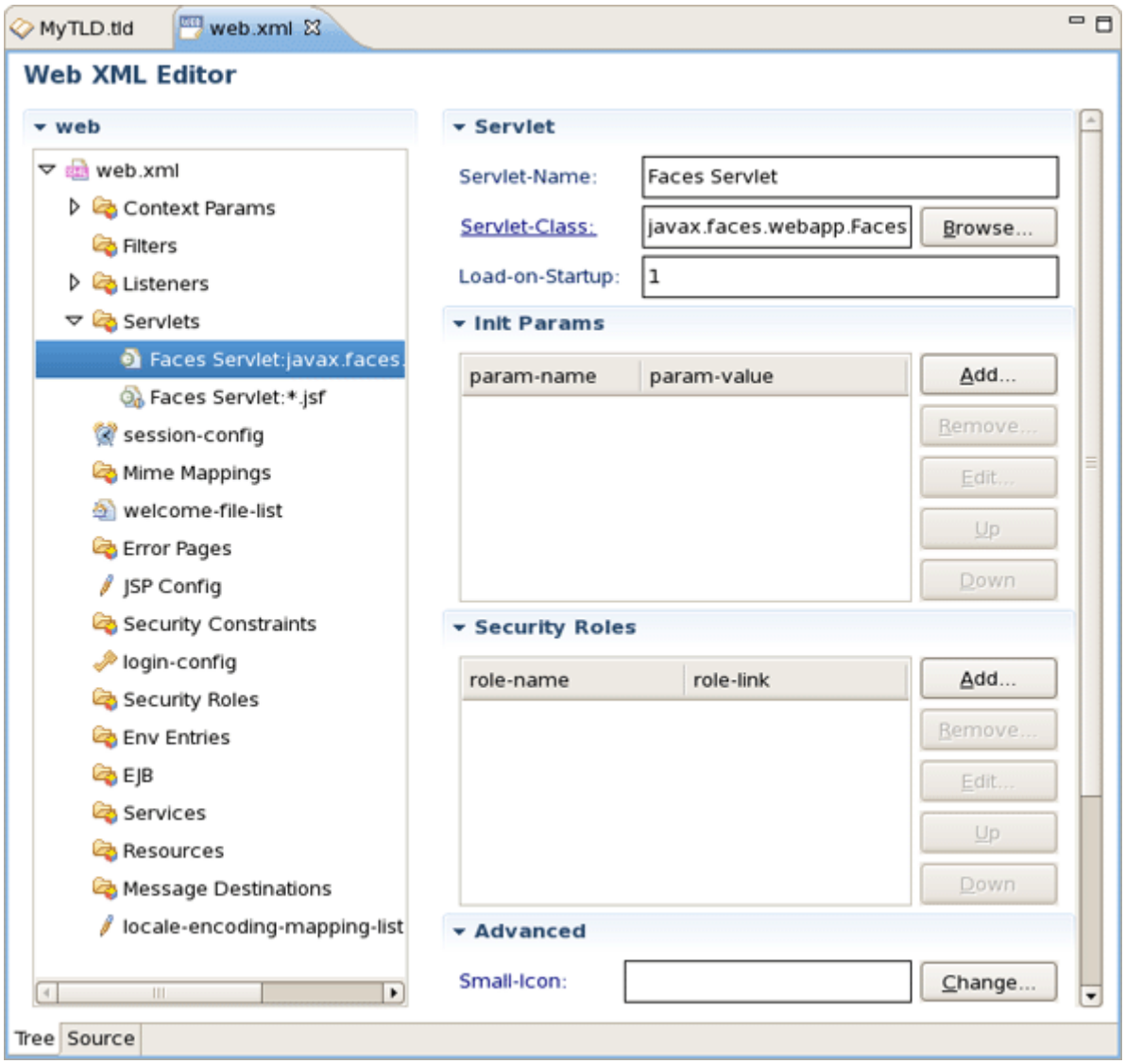

Figure 3.105. Tree View for editing web.xml in a graphical mode

You can add any new elements right in the **Tree** viewer:

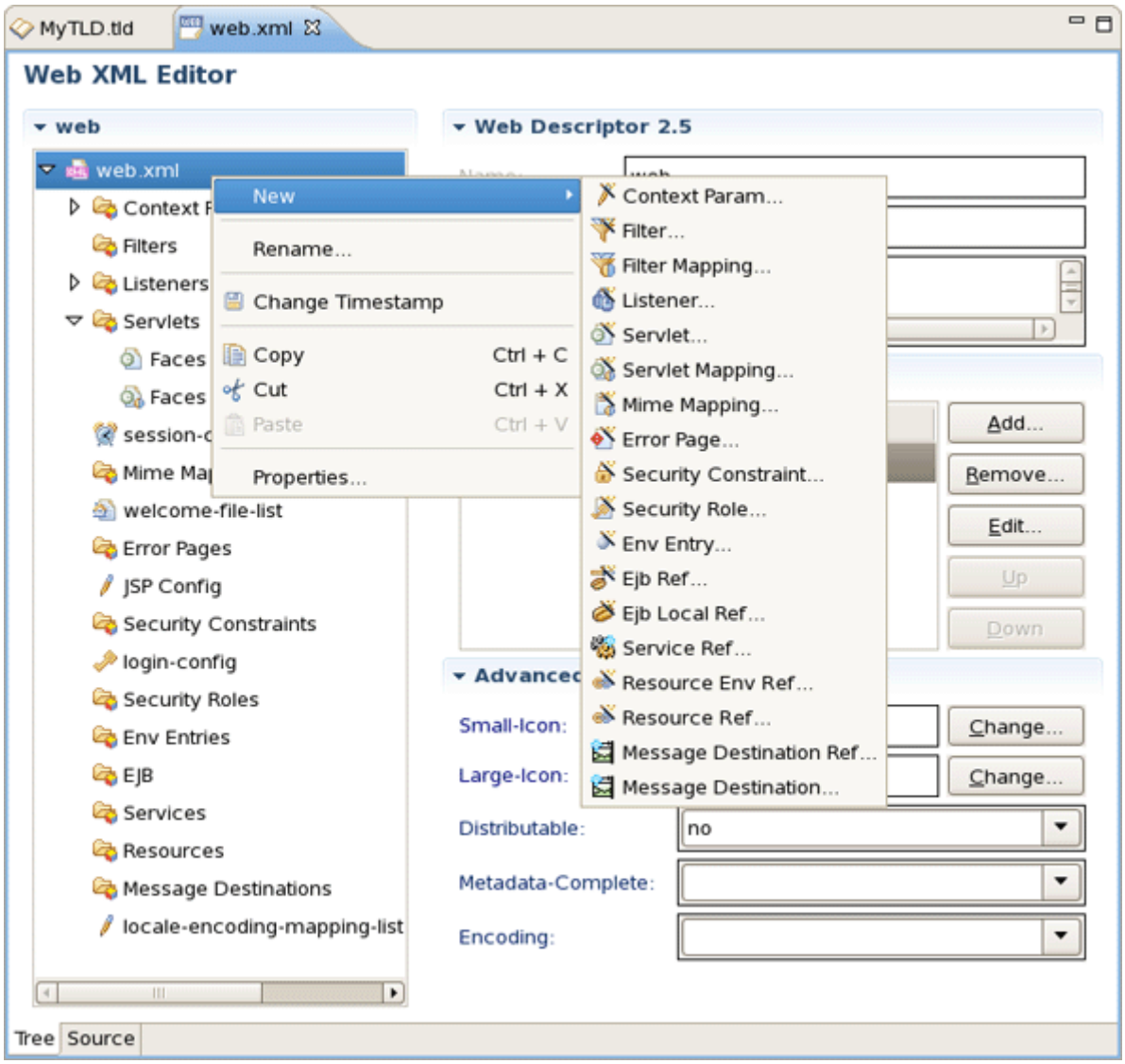

Figure 3.106. Adding New Elements in Web XML Editor

You can use the **Servlet-Name** drop-down field to select an XML element, and navigate to the location of the element within the **Source** of the XML file.

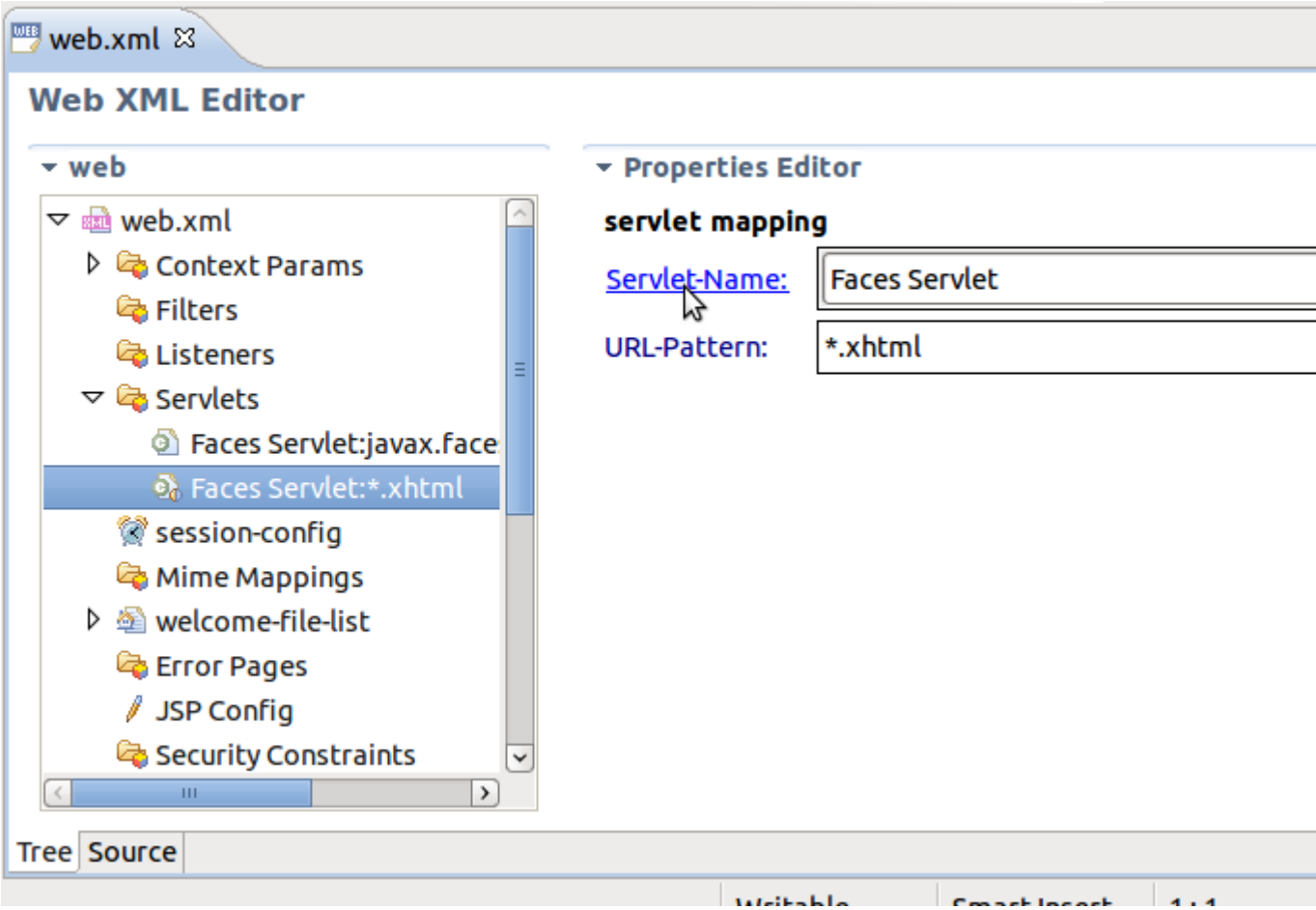

Figure 3.107. Navigating to XML elements from the Properties Editor

3.3.3.2. Source View

Switch to the **Source** viewer to edit the `web.xml` file by hand at any time:

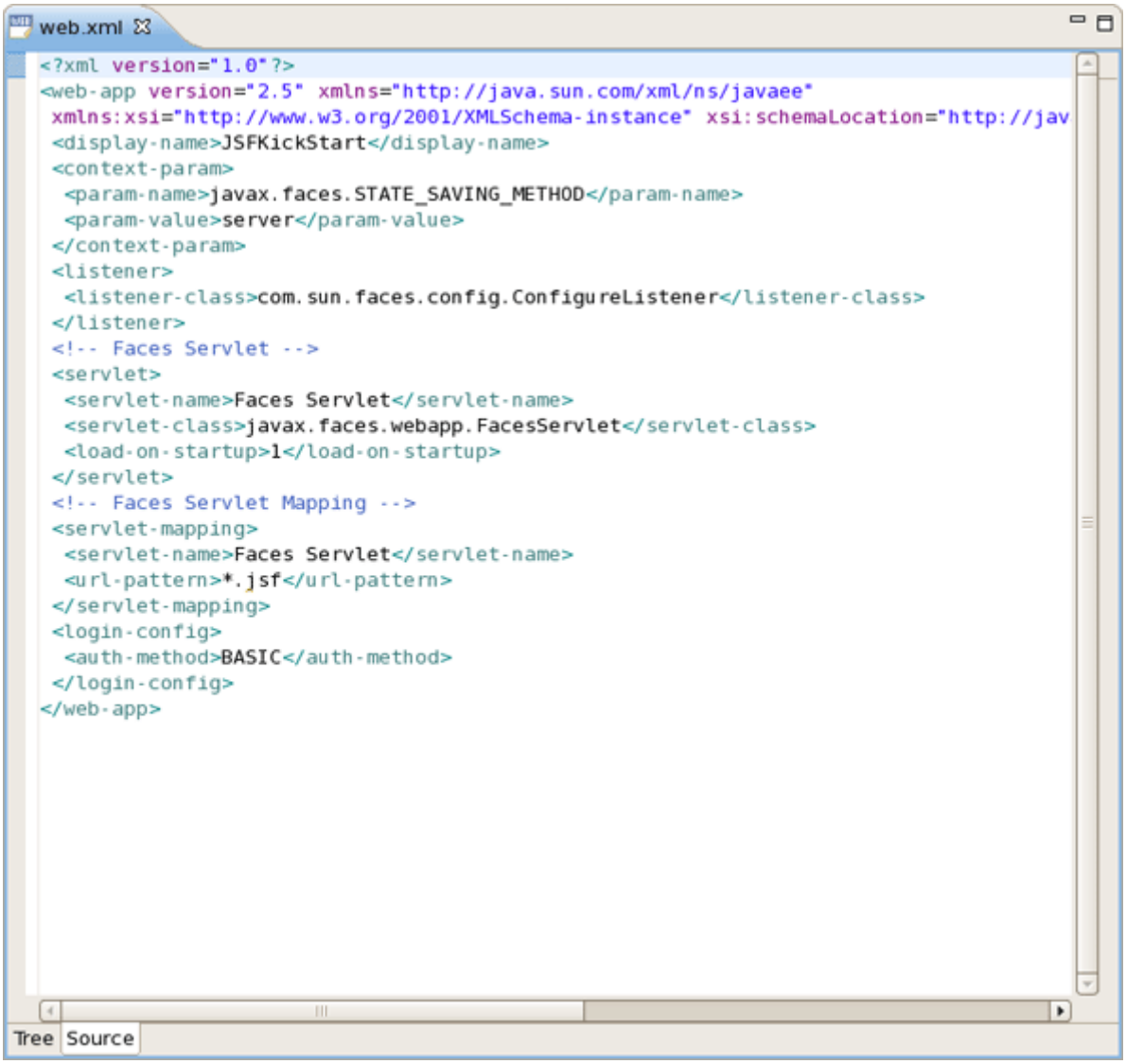

Figure 3.108. Web XML Source View

3.3.3.3. Content Assist

Content assist is available in the **Source** viewer. Simply click **CTRL**+**Space** anywhere in the file.

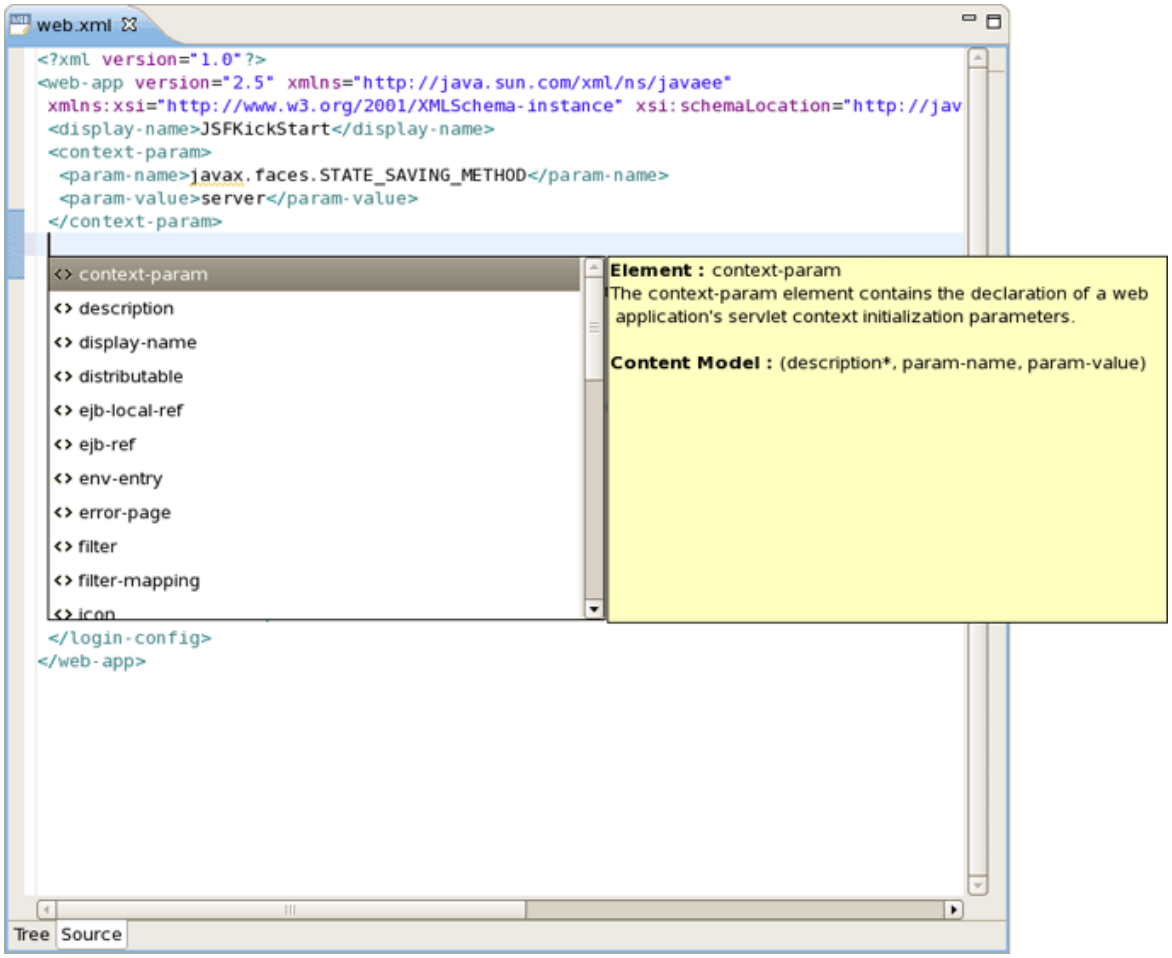

Figure 3.109. Web XML Content Assist

3.3.3.4. Errors Checking and Validation

If errors occur anywhere in the file, small red dots will appear next to the lines where the errors occurred. Also note that the file is marked by a small x in the **Package Explorer** view.

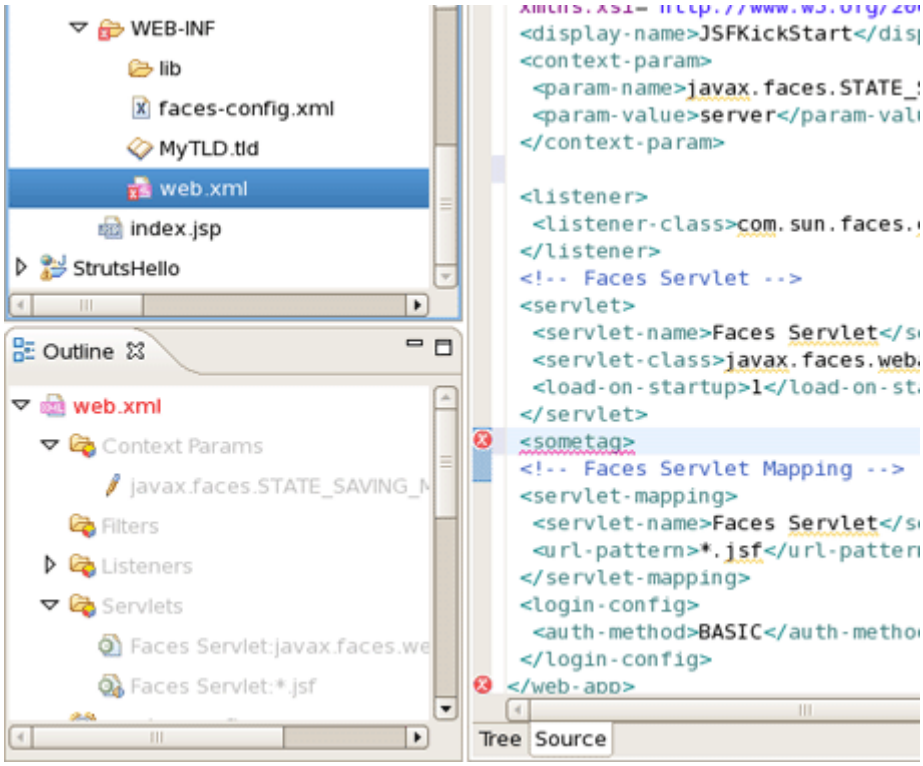

Figure 3.110. Errors Reporting

3.3.4. CSS Editor

The CSS editor comes with the same features you will find in all other JBoss Developer Studio editors.

- Content assist
- Validation and error checking

With the CSS (Cascading Style Sheet) editor, you can take advantage of code prompting:

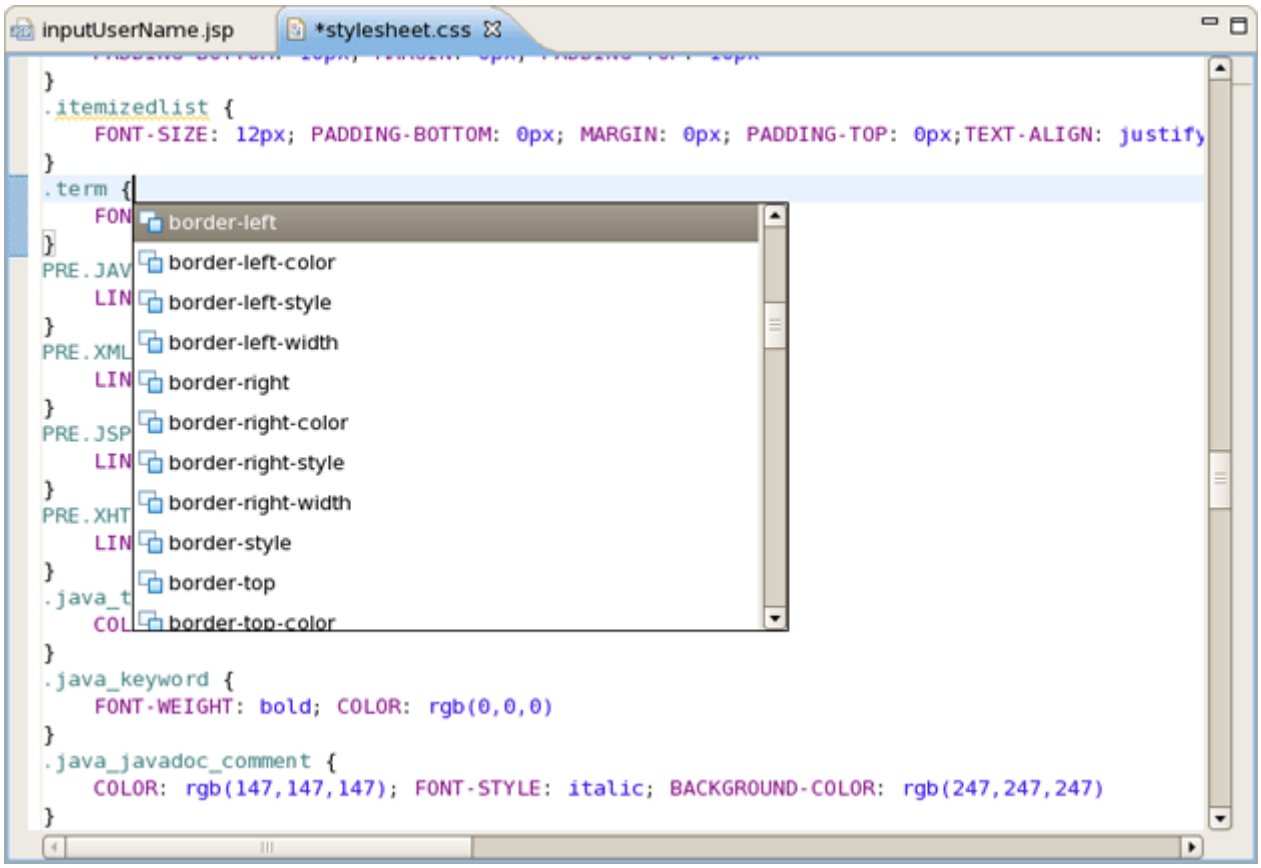

Figure 3.111. CSS Editor

And you can also use the **Properties** view next to the editor to edit existing stylesheet declaration properties:

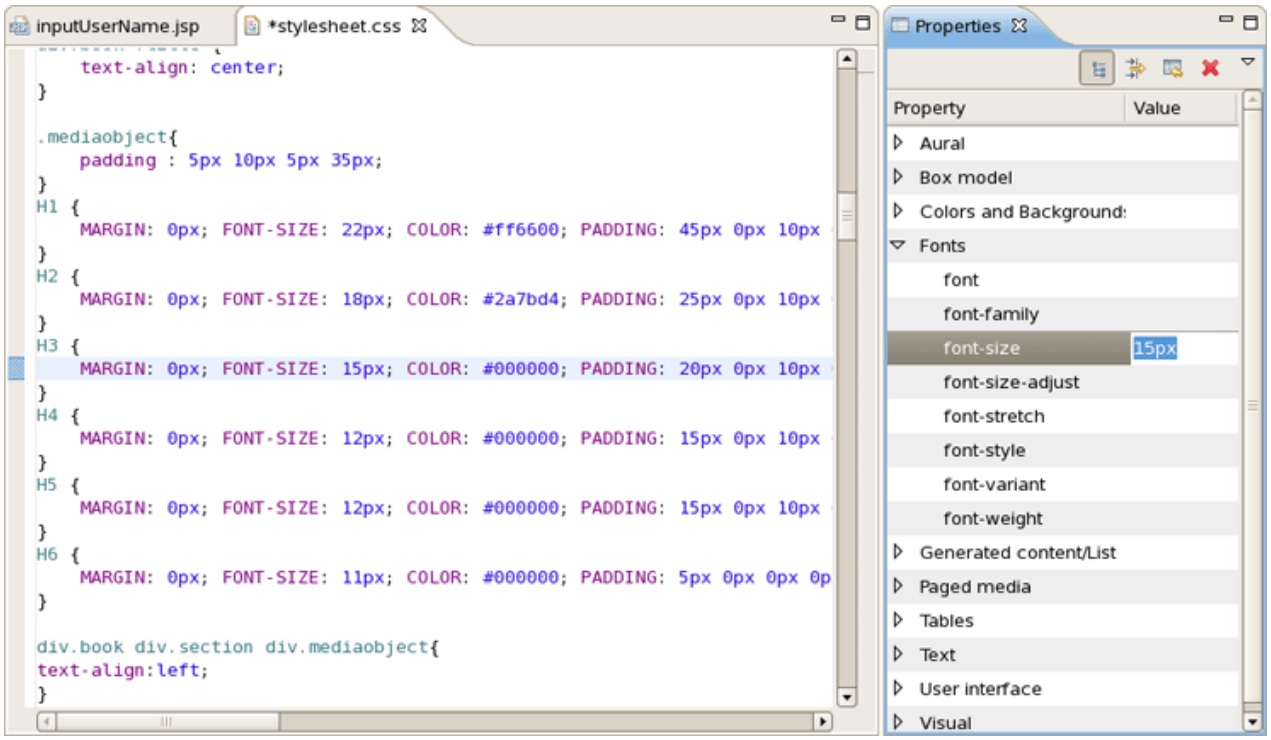

Figure 3.112. Properties View in CSS Editor

The **CSS** perspective is available to facilitate working on Cascading Style Sheets. For more information please read *Chapter 6, CSS Editing Perspective*

3.3.5. JavaScript Editor

The JavaScript editor is a Source viewer in which you can use code assist:

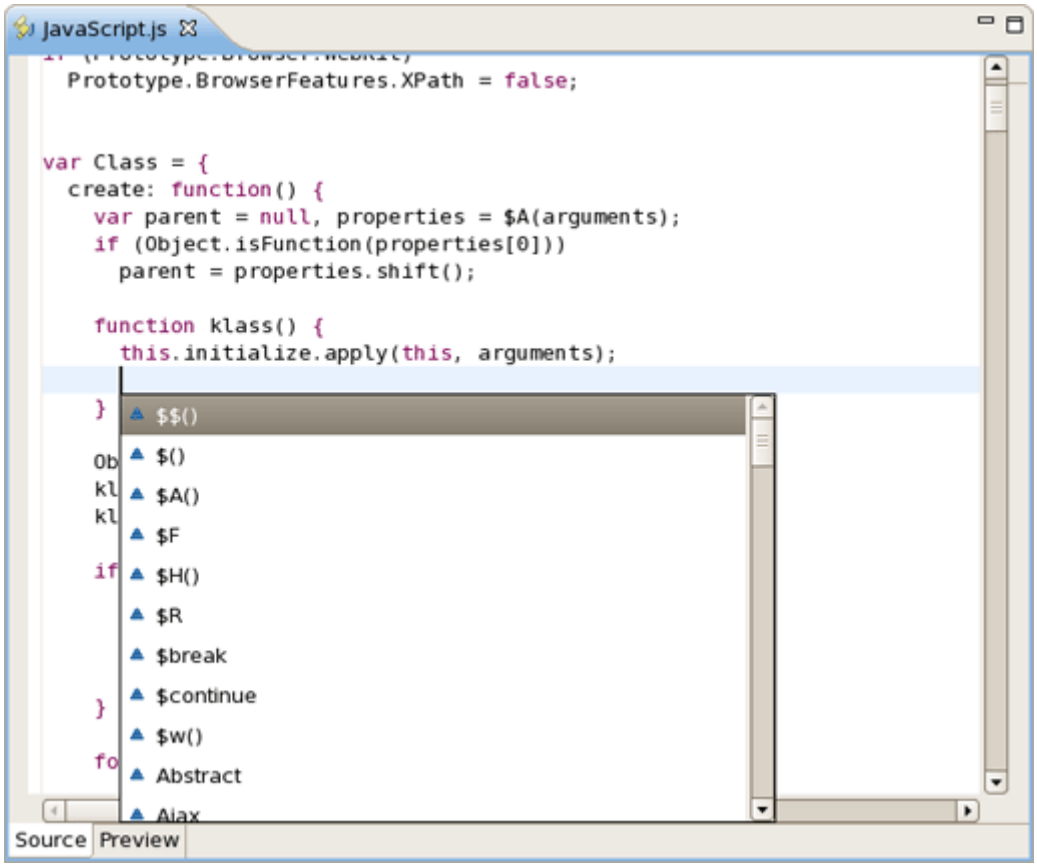

Figure 3.113. JavaScript Editor

You can also use the JavaScript editor with the **Outline** view to navigate around the file:

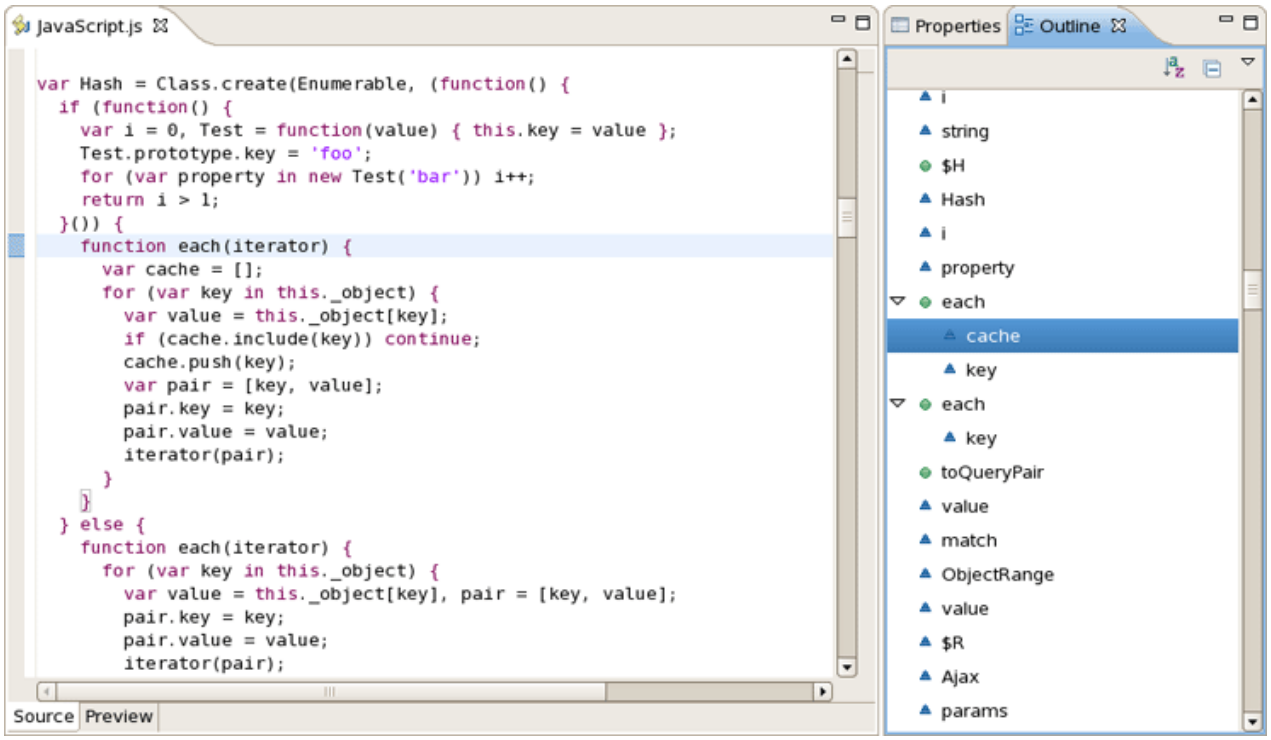

Figure 3.114. JavaScript Editor with the Outline view

3.3.6. XSD Editor

JBoss Developer Studio comes with an XSD Editor for XML Schema files. This editor comes from the Web Tools Project (WTP) (see *WTP Getting Started* [http://www.eclipse.org/webtools/testtutorials/gettingstarted/GettingStarted.html]).

To create a new XSD file, right-click a folder in the Package Explorer view, select **New** → **Other...** from the context menu and then select **XML** → **XML Schema File** in the dialog box.

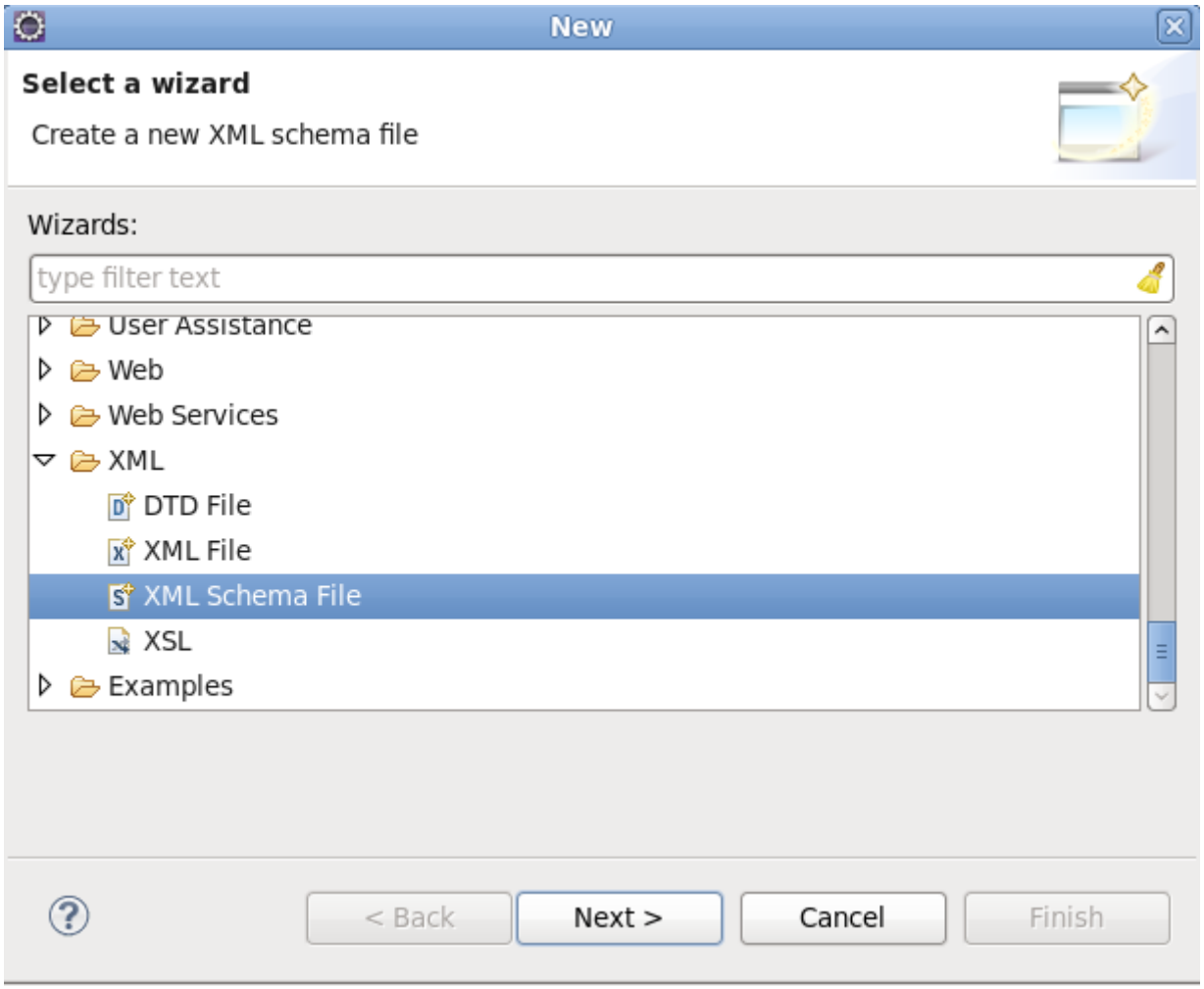

Figure 3.115. Creating New XSD file

The XSD Editor includes two viewers for working on the file, a **Design** viewer and a **Source** viewer:

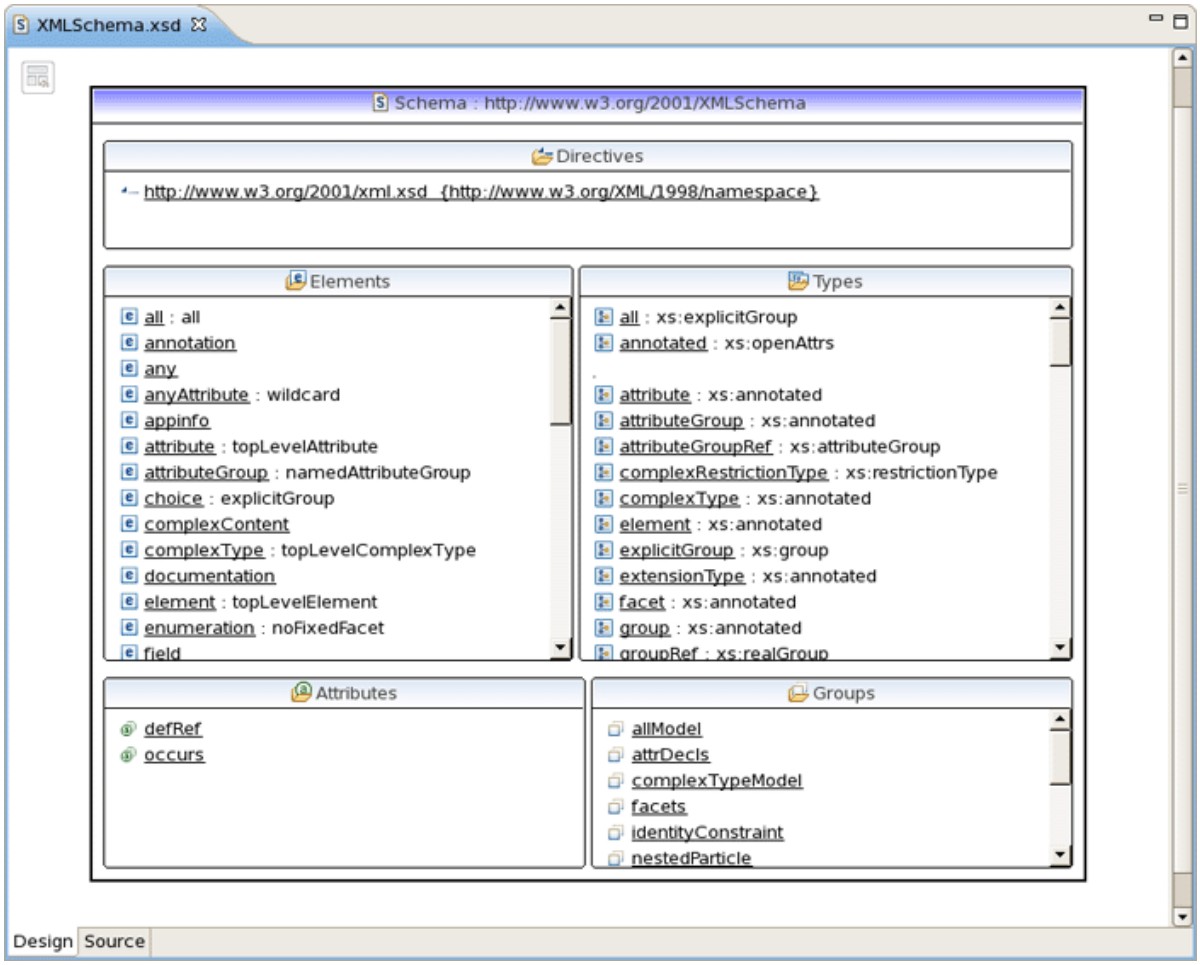

Figure 3.116. Source Viewer in XSD Editor

In the **Design** viewer you can drill down on an element by double-clicking on it:

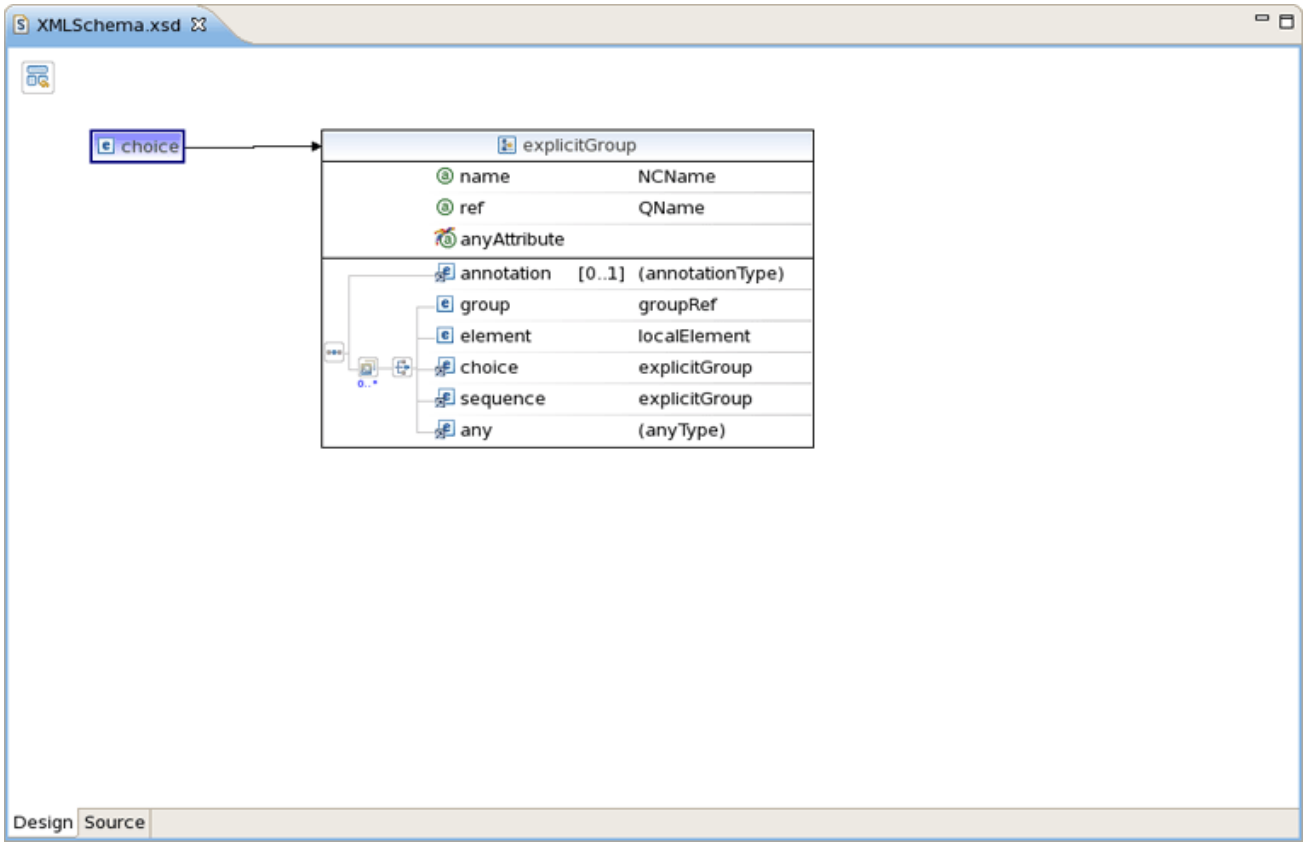

Figure 3.117. Design Viewer in XSD Editor

Various edit options are available when you right-click an element in the diagram:

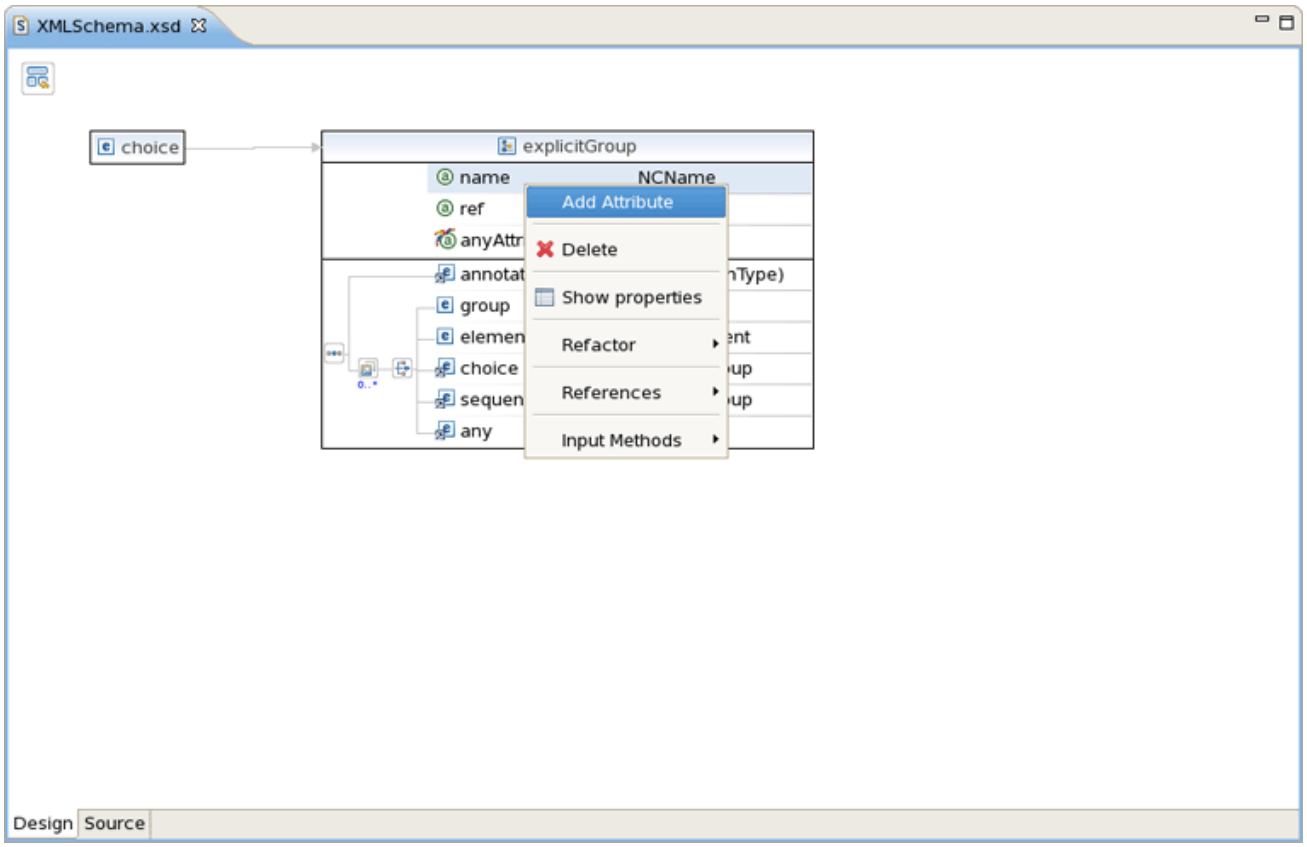

Figure 3.118. Edit Options in XSD Editor Context Menu.

You can also use the **Properties** view to edit a selected element:

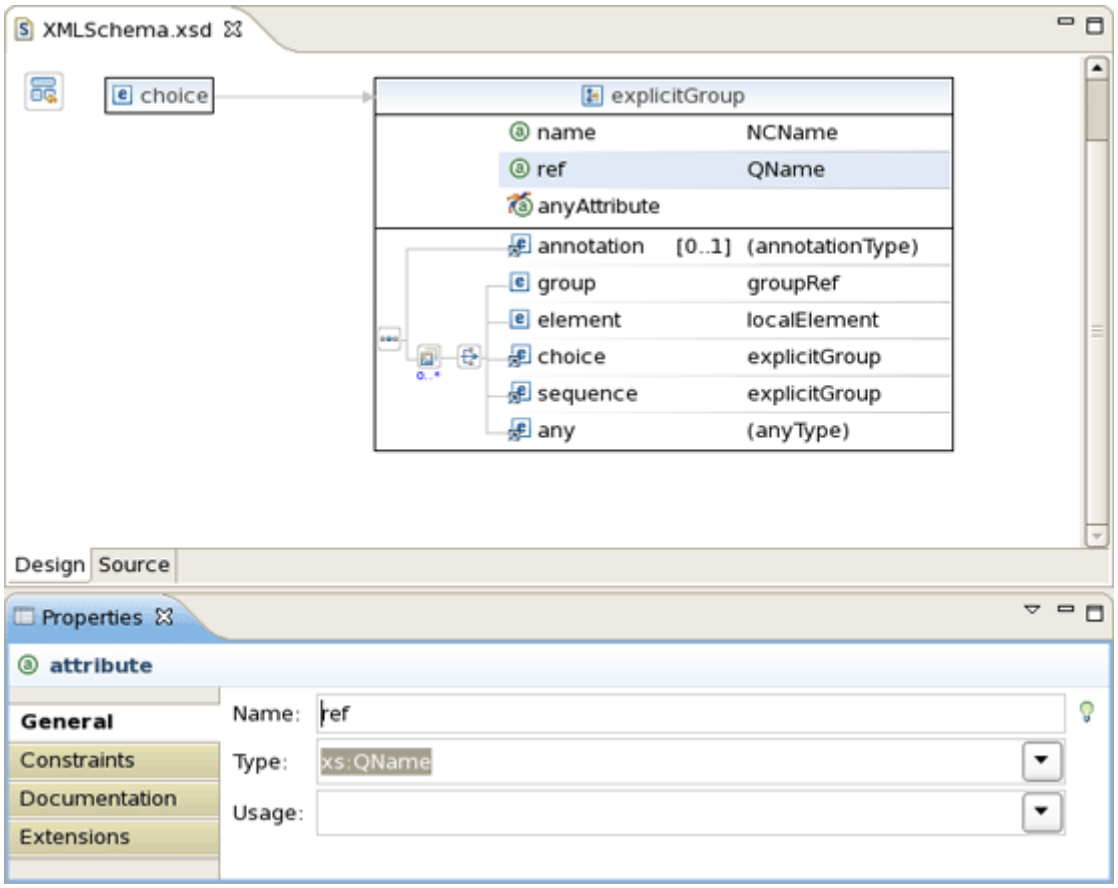

Figure 3.119. Properties View in XSD Editor

You can also use a **Source** viewer for the file. In this viewer, along with direct editing of the source code, you can also edit the file by using the **Properties** view on the right:

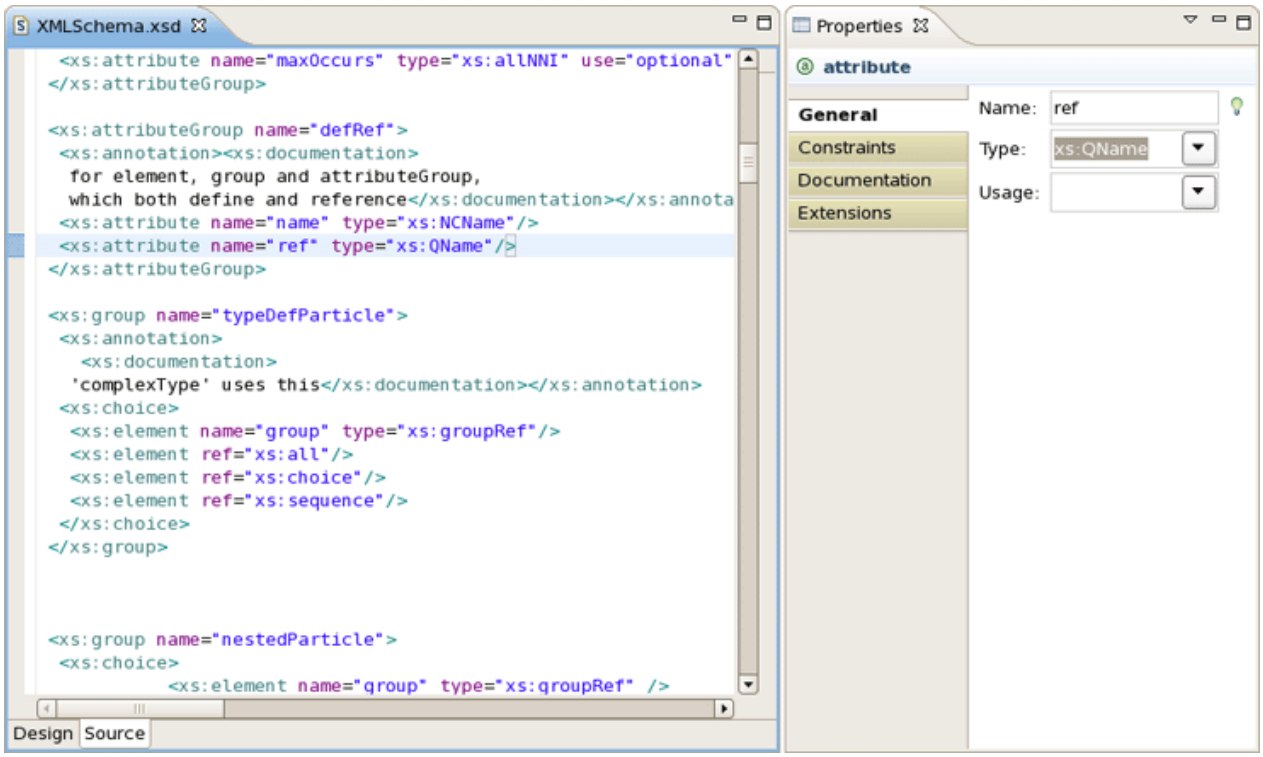

Figure 3.120. Using Source Viewer and Properties View together for source code editing

3.3.7. Support for XML Schema

JBoss Developer Studio fully supports XML files based on schemas as well as DTDs:

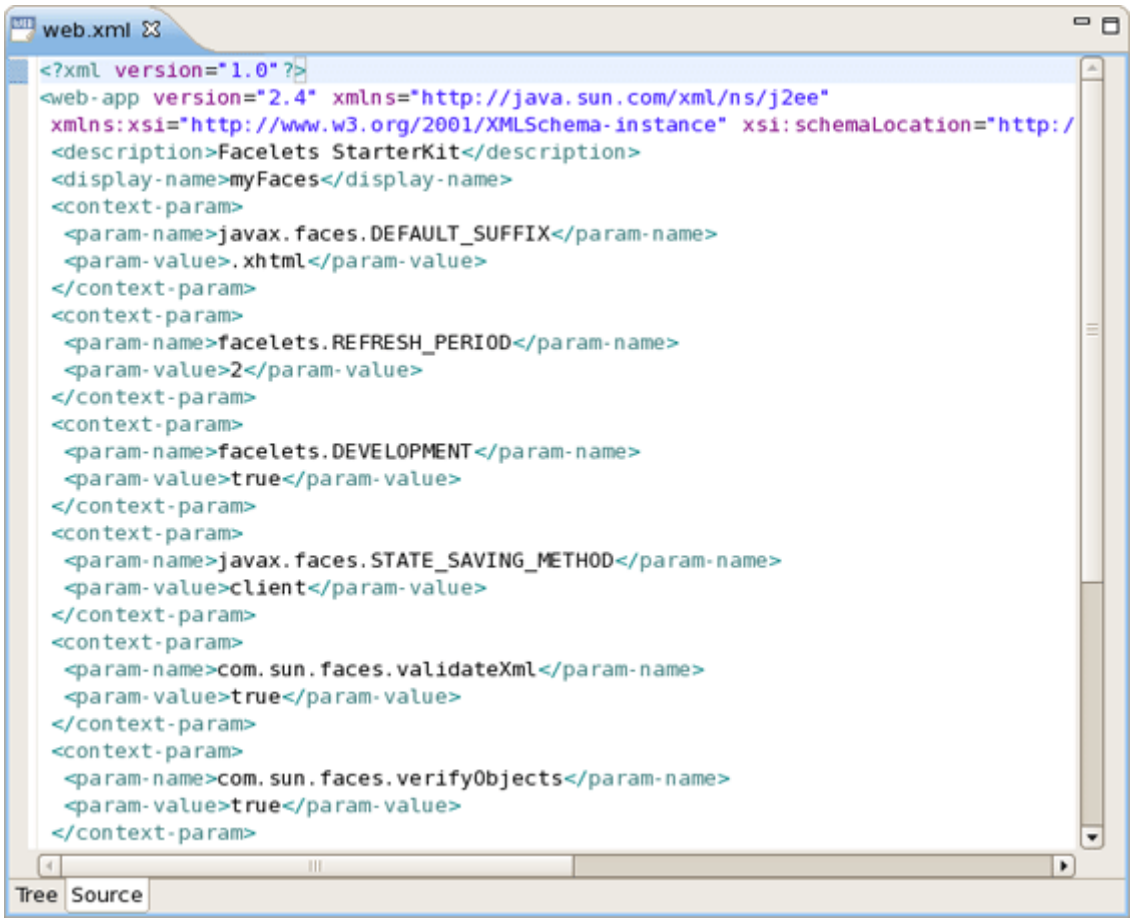

Figure 3.121. XML File

Note:

In case you want to use your own DTD or XML Schema make sure that this DTD or XML Schema is not listed in XML Catalog. If it is, you can't work with your DTD and XML Schema and JBoss Tool which uses this DTD or XML Schema. More information about XML Catalog you can find in *Eclipse Documentation* [http://help.eclipse.org/galileo/index.jsp?topic=/org.eclipse.wst.xmleditor.doc.user/topics/cxmlcat.html] and on *XML Catalog Tutorial page* [http://www.eclipse.org/webtools/community/tutorials/XMLCatalog/XMLCatalogTutorial.html].

Browsers

Different browsers are available for testing the look of a web page or site before going to production. This chapter outlines each browser type available and how to utilize them.

Note

These web browsers are only for testing, contents may appear diferently depending on the browser used to view your page or site outside of the workbench environment.

4.1. Generic web browser

A generic web browser is available for testing within your workbench environment. To access the browser, right-click on your `HTML`, `XHTML` or other web page extention file and navigate through the context menu to **Open With** → **Web Browser**.

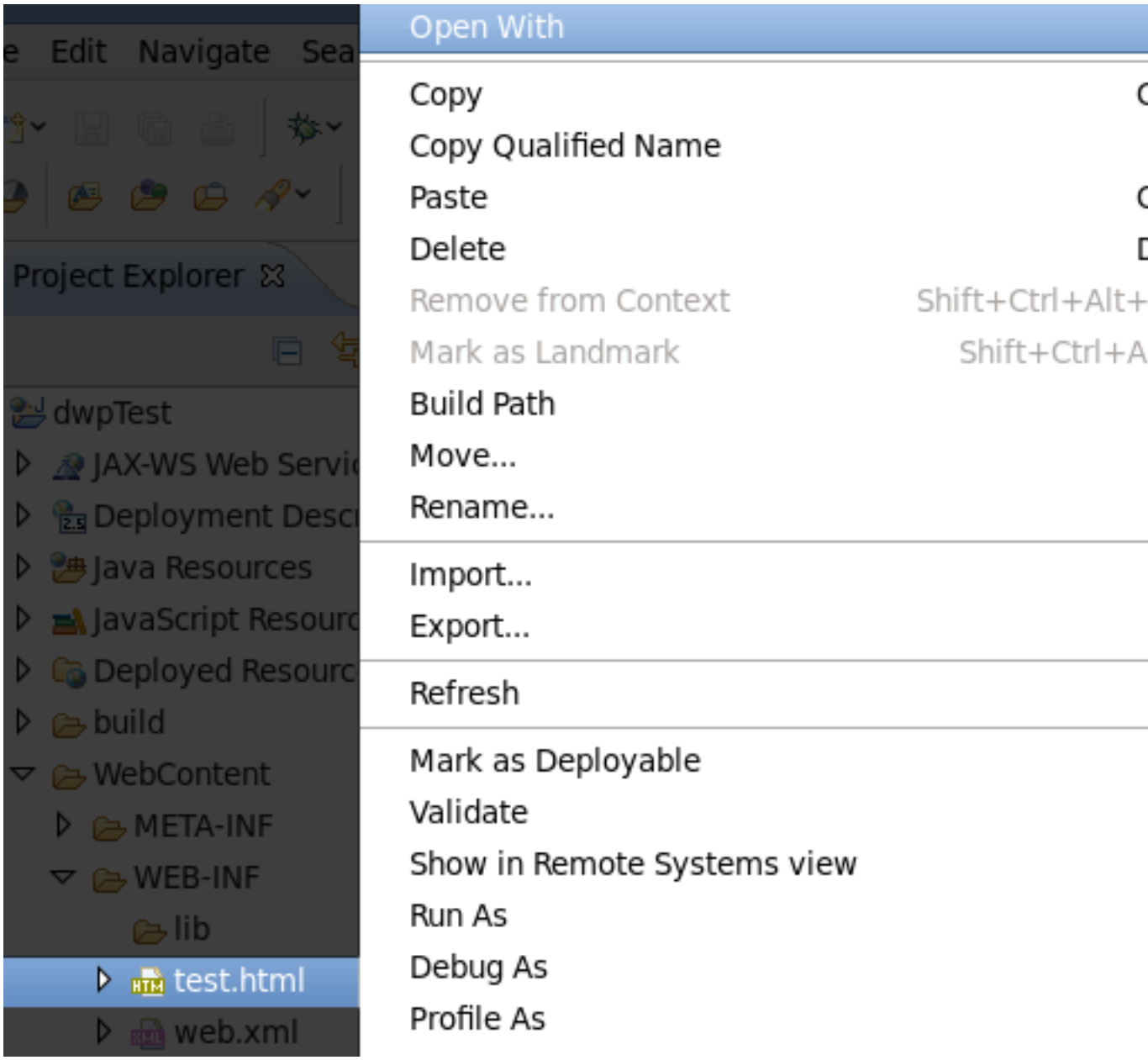

Figure 4.1. Mobile web browser simulator

A new tab will launch in your workbench, displaying the contents of the page you selected.

4.2. Mobile web browser

BrowserSim simulates a mobile web browser so you can see how your web page or site will be viewable on mobile devices.

4.2.1. System requirements

To run the mobile browser simulator you will require certain configurations depending on your operating system.

Linux distributions:

- WebKitGTK 1.2.0 or newer must be installed and available within the library load path. This is setup by default on Red Hat Enterprise Linux and Ubuntu. For other distributions you may need to install the `libwebkitgtk` package.

Windows operating systems:

- 32-bit version of JBoss tooling.
- Quicktime or iTunes. Each of these installs Apple Application Support in the folder BrowserSim assumes. Apple Application Support is necessary for BrowserSim to work correctly on Windows.

There are no specific requirements for Red Hat Enterprise Linux and Macintosh operating systems.

4.2.2. Using BrowserSim

To test your page or site using **BrowserSim**, right-click on your `HTML`, `XHTML` or other web page extention file and navigate through the context menu to **Open With** → **BrowserSim**.

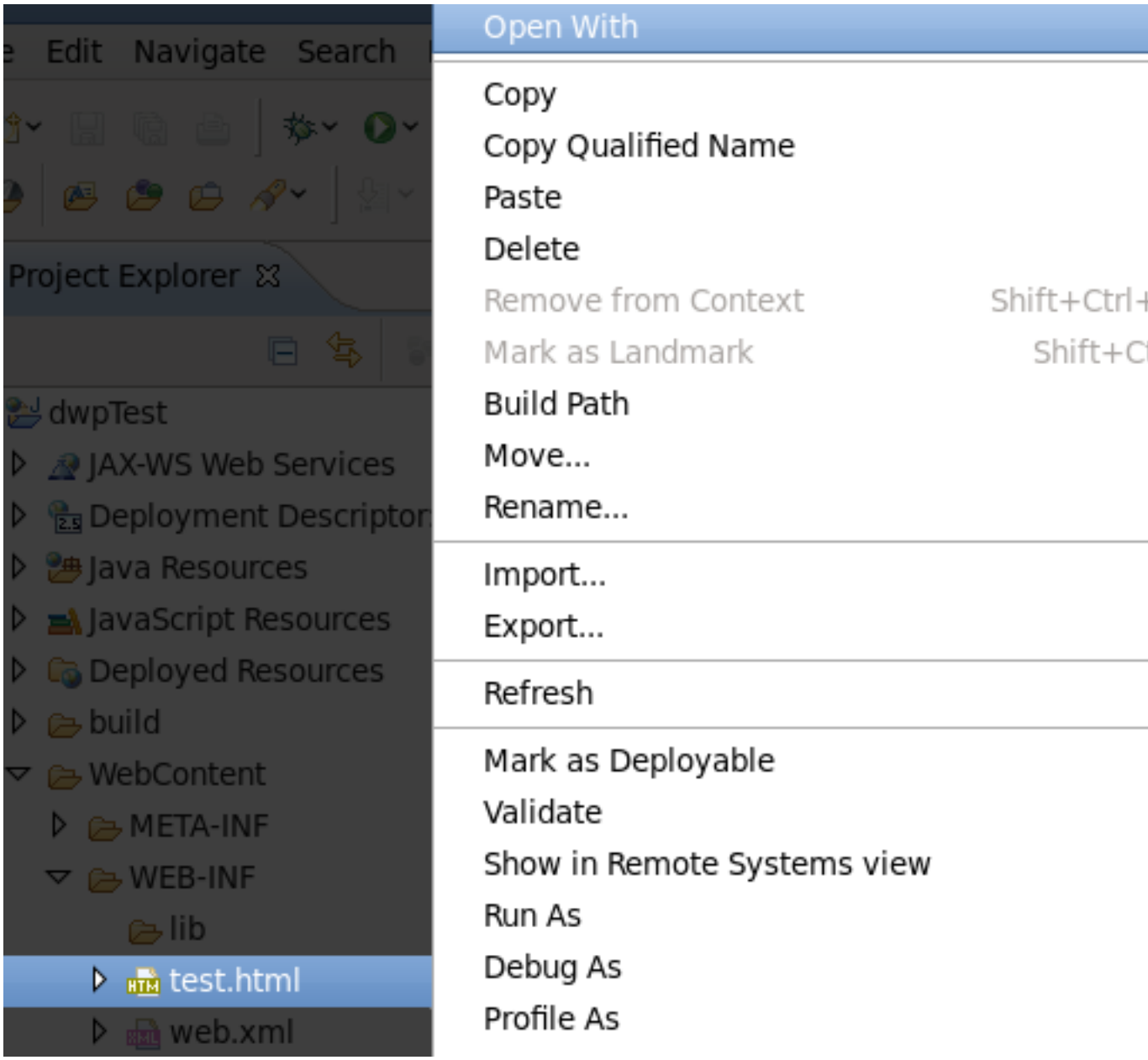

Figure 4.2. Mobile web browser simulator

A new window will launch, displaying the contents of the page you selected as it will appear on mobile devices.

You are able to select the type of mobile device to simulate the browser on by right-clicking on part of the theme and selecting from the context menu.

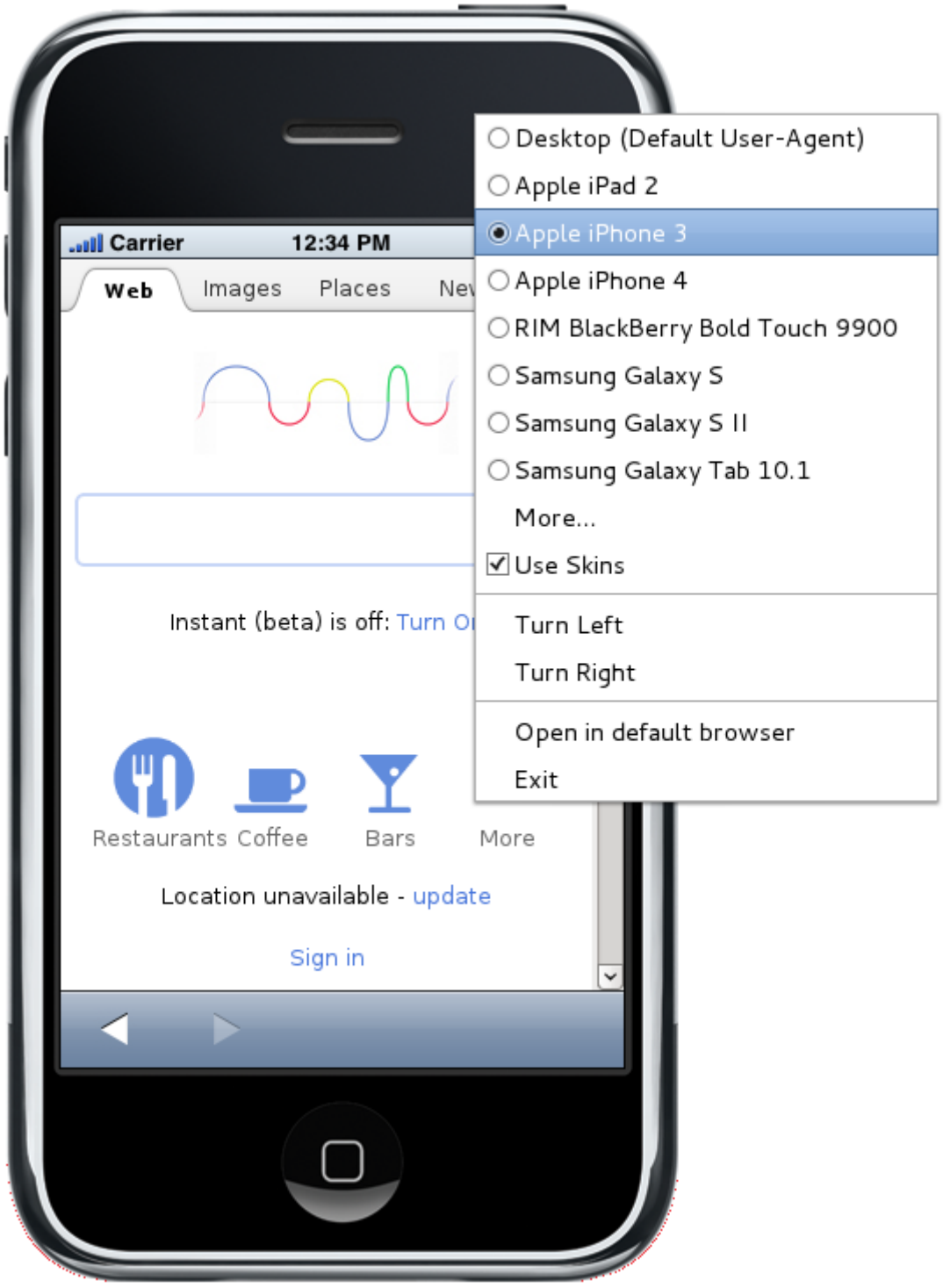

Figure 4.3. Mobile web browser device selection

A **View page source** is also available from the contexzt menu. This can be helpful when wishing to view the HTML code of an Internet site.

To open the **BrowserSim** without using a context menu, click on the **BrowserSim** button
in your toolbar. If the button is not in your toolbar you can add it by undertaking the following:

- Right-click on the toolbar at the top of your workspace and select **Customize Perspective** from the context menu.

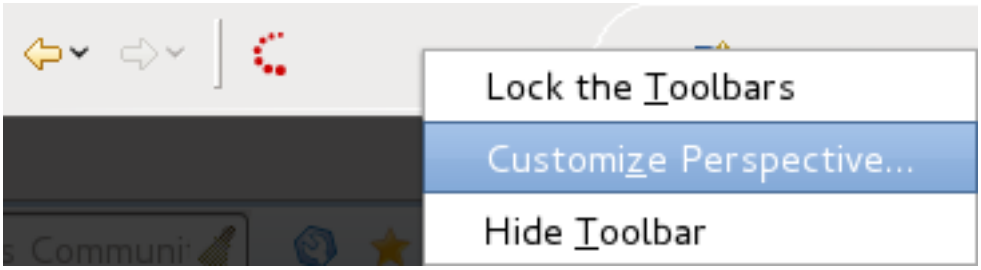

Figure 4.4. Toolbar context menu

- Click on the **Command Groups Availability** tab and ensure the **BrowserSim** command group is checked.

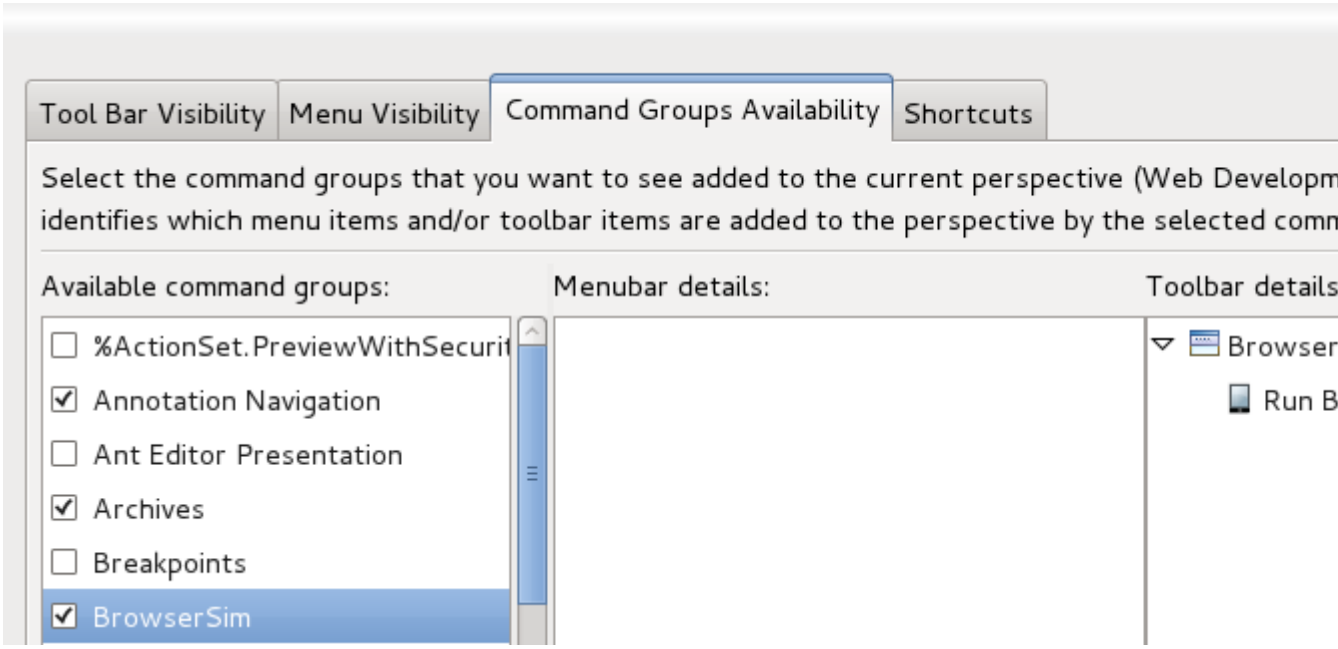

Figure 4.5. Command Groups Availability tab

- Click on the **Tool Bar Visibility** tab and select the **BrowserSim** toolbar items menu and click the **OK** button.

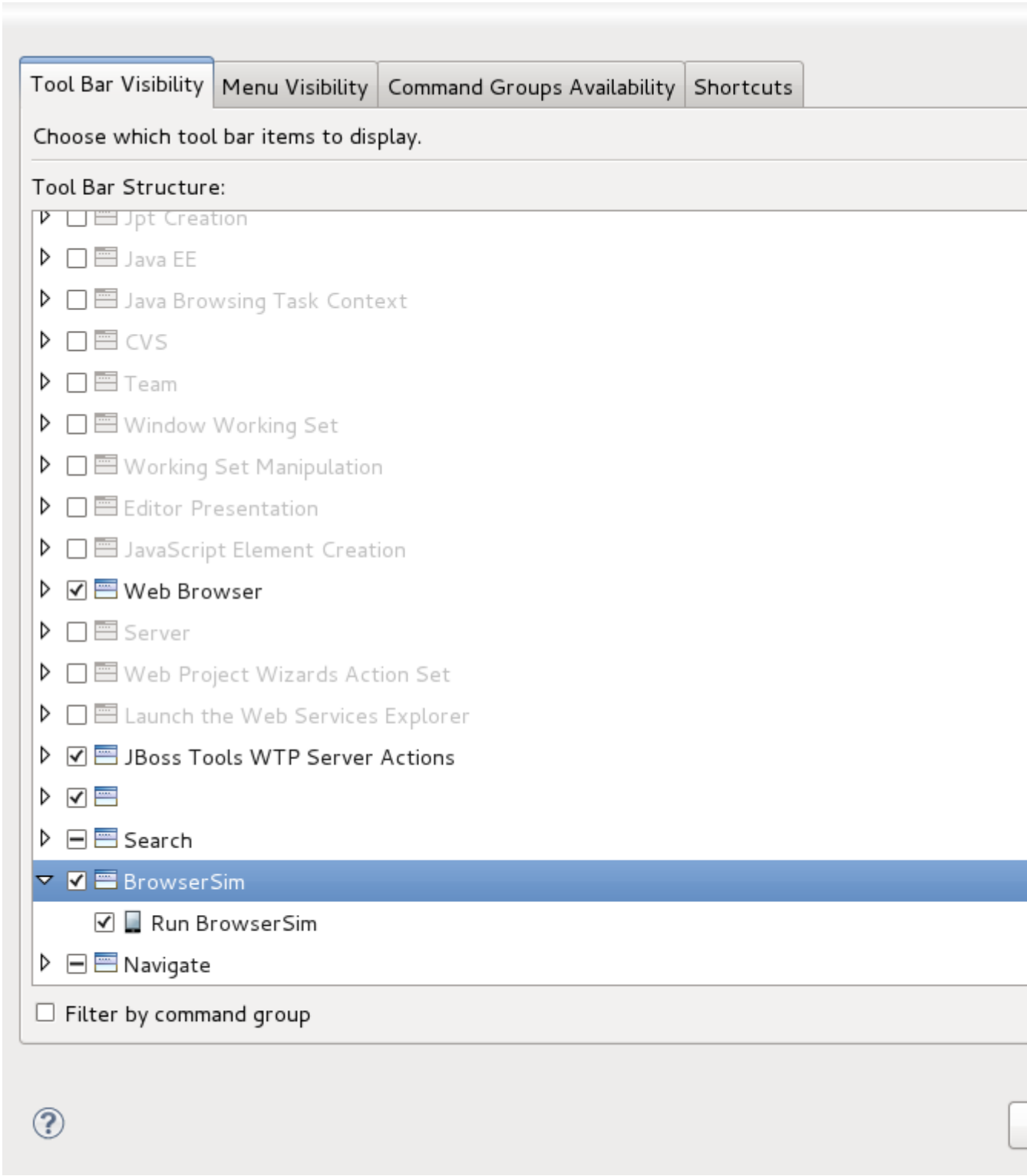

Figure 4.6. Tool Bar Visibility tab

- The **BrowserSim** toolbar button will now be present in the toolbar for your current perspective.

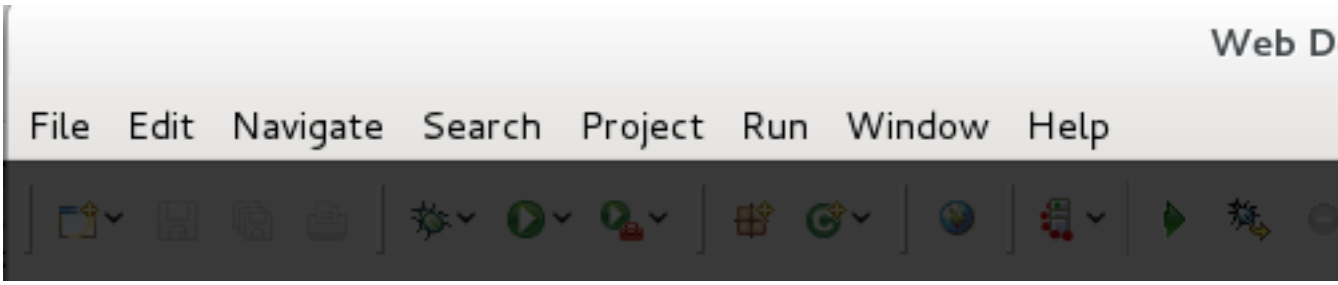

Figure 4.7. Tool Bar Visibility tab

Launching **BrowserSim** from the toolbar will either present you with the simulator and a blank webpage or the webpage currently open in the **Internal Web Browser**.

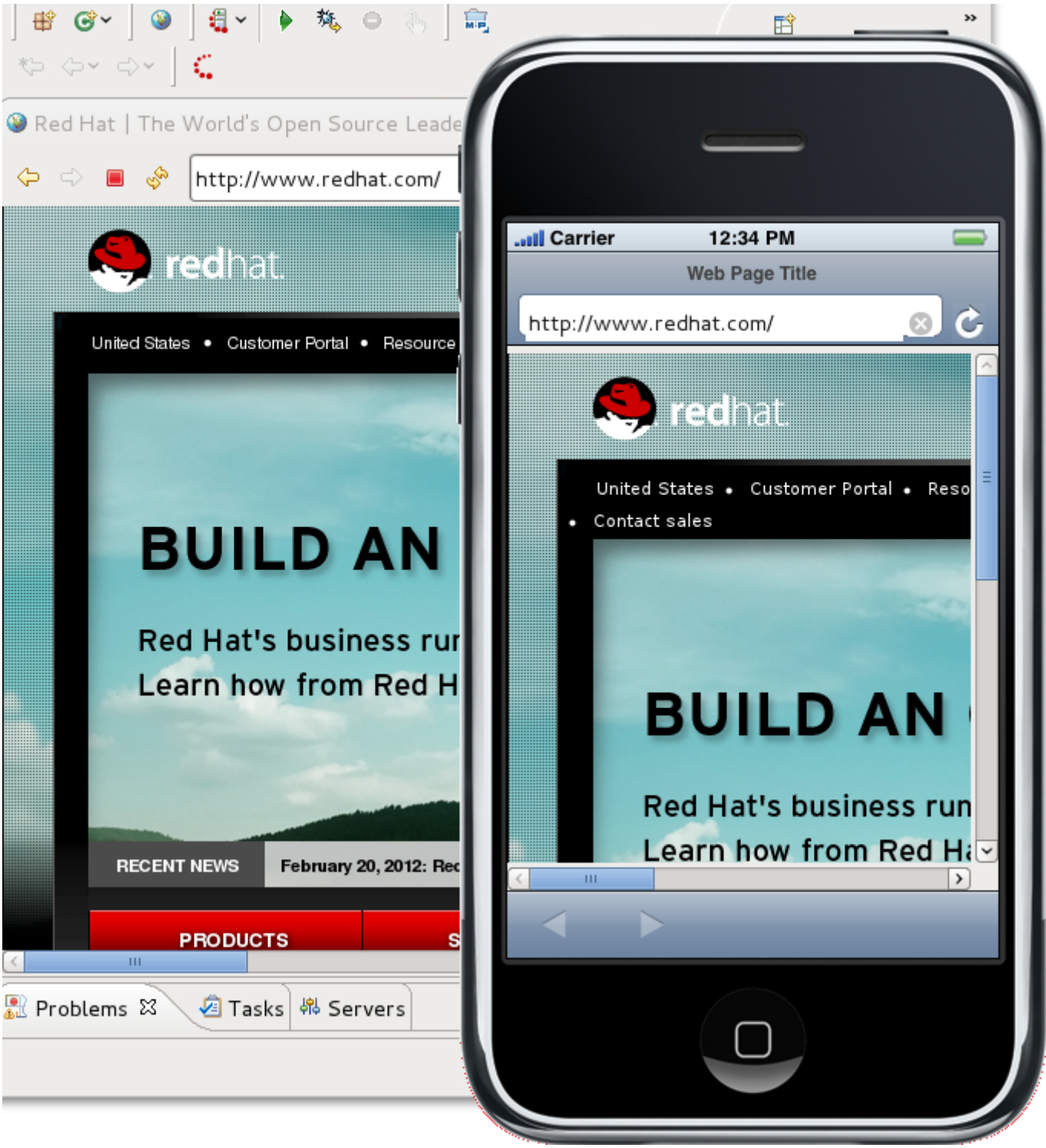

Figure 4.8. Tool Bar Visibility tab

JBoss Tools Palette

This chapter will introduce you to the functionality provided by **JBoss Tools Palette**. The Palette allows you to quickly and easily create your JSP or JSF pages.

The **JBoss Tools Palette** allows you to:

- Insert tags into a JSP or JSF page with one click
- Add custom and 3rd party tags

The **JBoss Tools Palette** contains a developer's project tag libraries and provides possibility to add any tag libraries to it. Also you can choose a necessary one from the list of already existing tag libraries:

- HTML
- JBoss
- JSF
- JSTL
- MyFaces
- Oracle ADF Faces
- XHTML

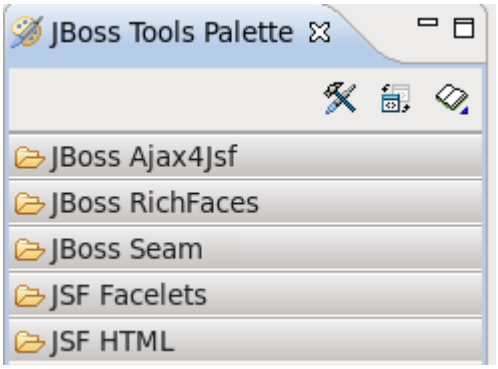

Figure 5.1. Default View of The JBoss Tools Palette

By default the **JBoss Tools Palette** is not displayed. If you want to use it select **Window** → **Show View Other...** → **JBoss Tools Web** → **JBoss Tools Palette** from the menu bar.

The standard **Eclipse Palette** is displayed by default in both **Web Development** and **Seam** perspectives. Now the standard **Eclipse Palette** is featured with all **JBoss Tools Palette** options and capabilities.

To open the standard **Eclipse Palette** navigate to **Window** → **Show View** → **Others** → **General** → **Palette**.

The differences between the two palettes are as follows:

- The standard **Eclipse Palette** is blank by default. Content of the palette is available only if Visual Page Editor is open and active, while **JBoss Tools Palette** always contains a predefined set of components.
- The Expanded/Collapsed state of components in the standard **Eclipse Palette** is not global as in **JBoss Tools Palette**. State is associated with an instance of Visual Page Editor. It means that the state can be different for various files and each new file opened in Visual Page Editor will have the default state of Palette with all components collapsed.

5.1. Palette Options

The Palette can be customized by using the following buttons on the Palette toolbar, which provide the following functionality:

- editing the palette content by adding, removing or changing the palette elements
- showing or hiding groups and subgroups
- importing groups and subgroups

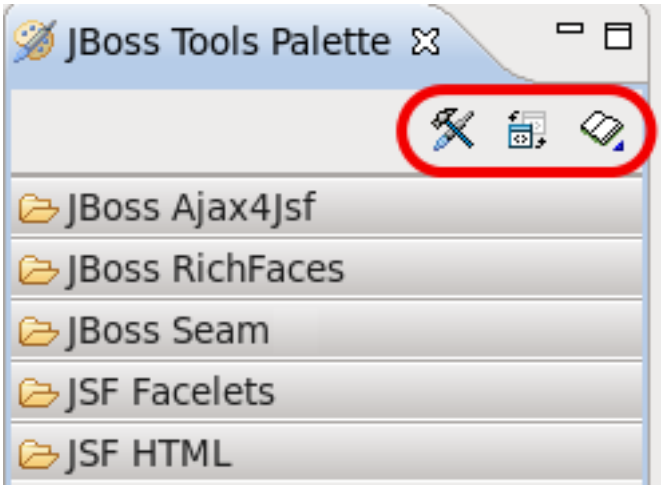

Figure 5.2. Palette Buttons

5.1.1. Palette Editor

The **JBoss Tools Palette** contains existing libraries of tags, and the **Palette editor** provides a way to add a new library, or edit existing libraries.

To open the editor, click on the **Palette Editor** icon ( ).

The window has two parts. There is a reflected grouped list of components on the left side of the palette editor. Each group is divided into multiple groups, each of which is a tag library. To the right side of the palette editor is an editing window where it is possible to change values of group or tag library attributes that you have chosen on the left part of the window.

It can also be done by right click and using **Edit…** option.

For example, **JSF** group consists of **Core**, **Facelets**, **HTML** tag libraries and the attributes as **name**, **description** and **hidden** which are available for editing:

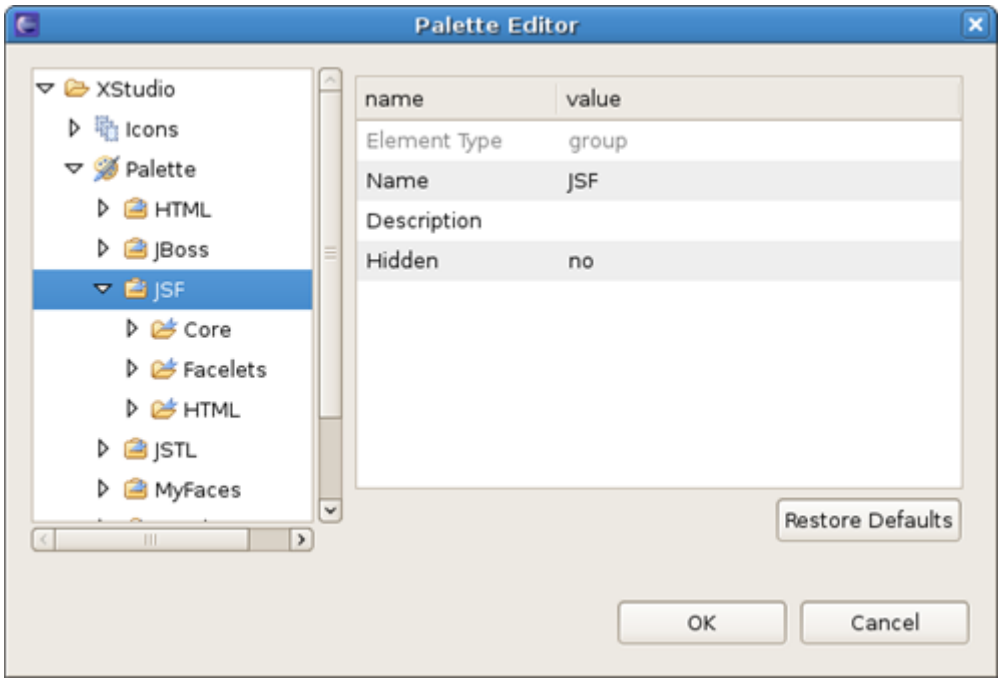

Figure 5.3. Tag Libraries of the JSF Group

The **Palette Editor** provides the following functionality when working with existing tags or icons:

- **To work with a set of icons.** **Icons** is the root folder for the icon sets. The first step is creating the icon set. Right click on the **Icons** folder and select **Create** → **Create Set...**.

  Set the value of the name in the **Add Icons** window and click the **Finish** button. A new element will appear in the list.

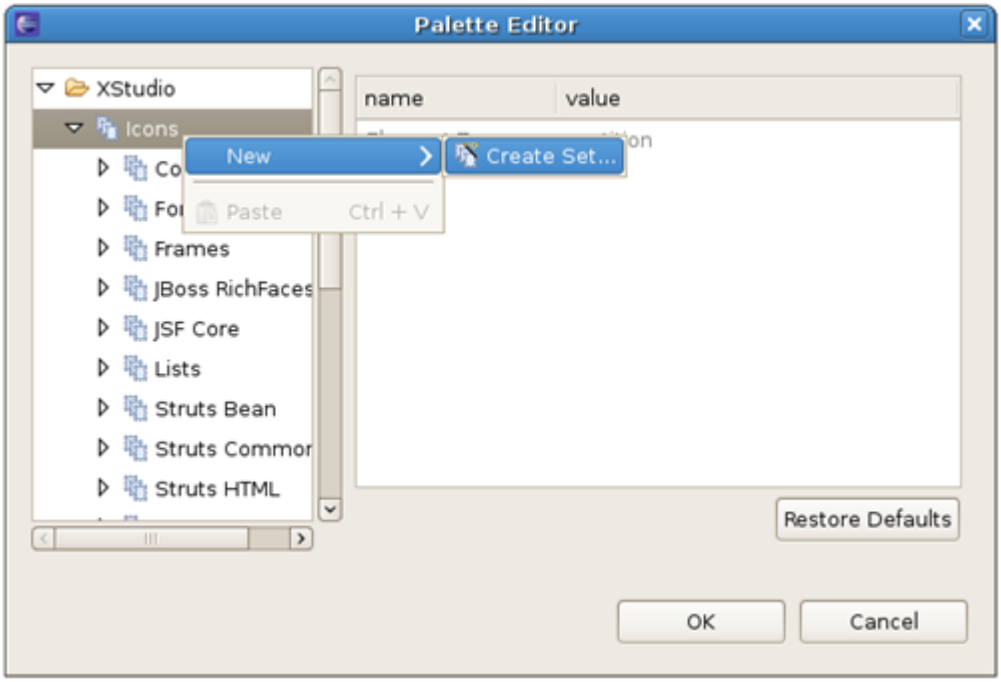

Figure 5.4. Creating a Set of Icons

Also you can delete the set. Right click on the set of icons that you wish to remove and select the **Delete** option from the pop-up menu, or click the **Delete** keyboard button.

- **To edit icons in the chosen set.** When the set of icons is created, new icons can be imported to it. Choose the required set and select the option **Create** → **Import Icon...** from the pop-up menu that appears after you right-click on a folder.

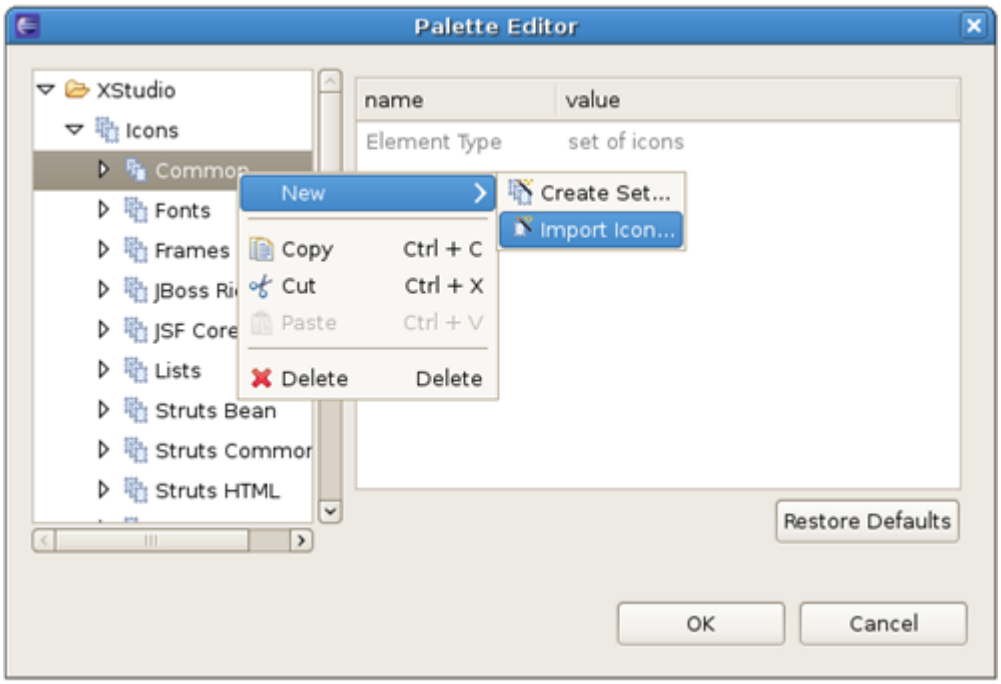

Figure 5.5. Creating Icons

Set the name of the icon and the path and click the **Finish** button.

- **To work with a group of tag libraries.** The first step in working with the editor is creating a group of libraries. It's very easy to do, right click on the **Palette** folder and select **Create** → **Create Group…**

  Set a name of a group in the **Create Group** window and click the **OK** button. A new element will appear at the end of the list.

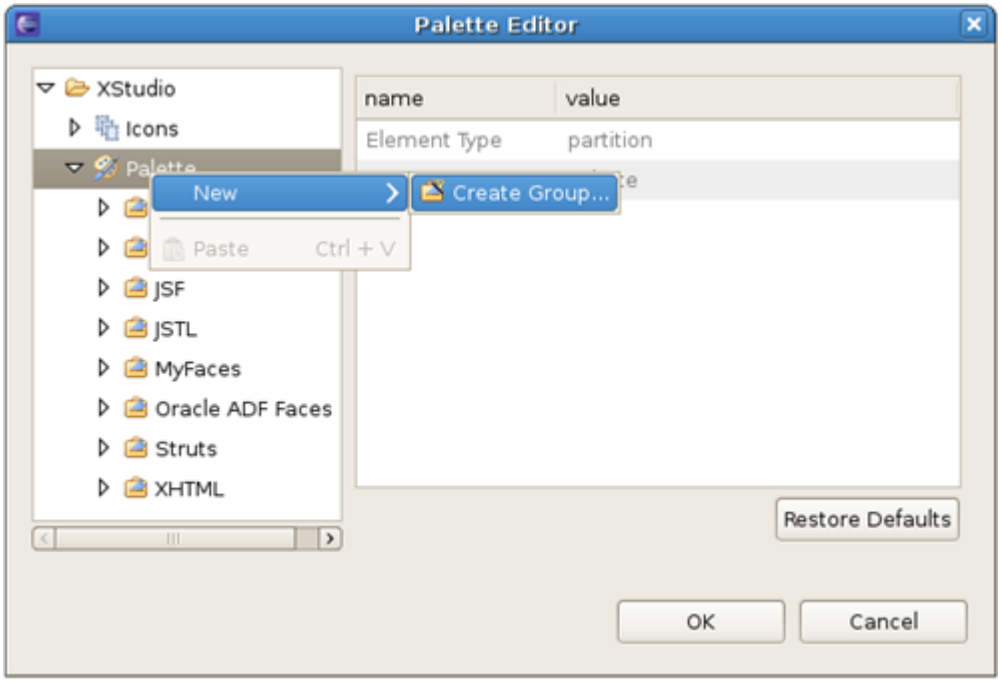

Figure 5.6. Creating a Group of Tag Libraries

You are allowed to edit or delete a group as well. If you'd like to change attributes of a group, use the right editing window of the palette editor or the **Edit...** option, like it was mentioned before. In order to remove the group, right click on the group that you wish to remove and choose the **Delete** option or click the **Delete** keyboard button.

Important:

The removal option is enabled only for custom folders.

- **To work with a tag library.** The group maintains a list of tag libraries. If you would like to create your own library, right click on the group and select the **Create Group…** option.

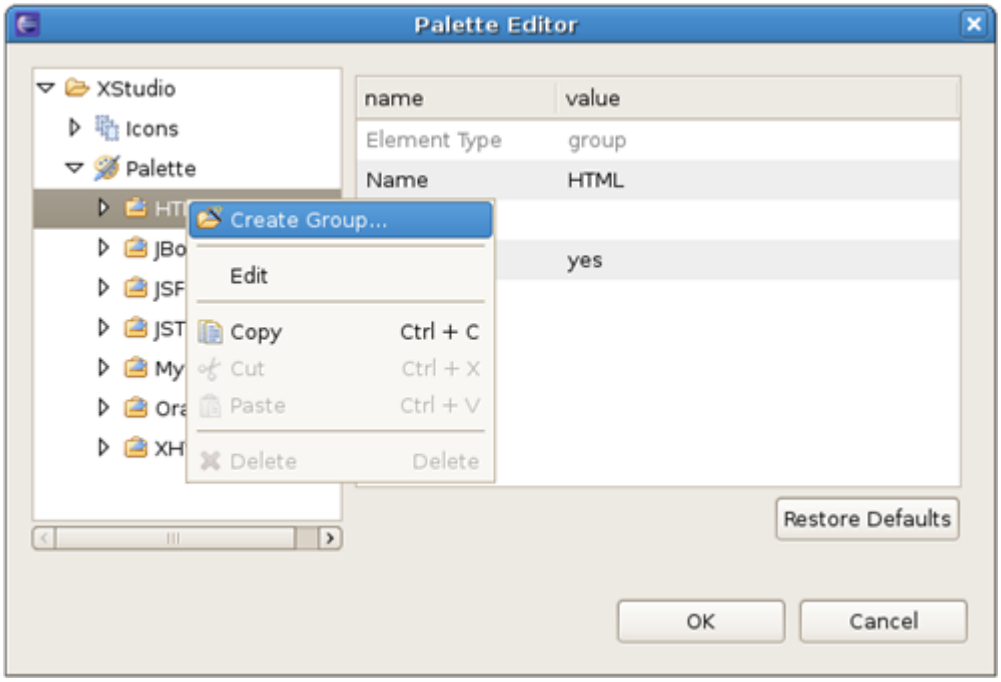

Figure 5.7. Creating a tag library

After setting the attribute name and the path of the icon, click the **OK** button.

Note:

If you do not choose an icon the default one will be assigned.

You are allowed to edit or delete the tag library, as well. If you'd like to change attributes of the library or choose another icon, use the right editing window of the palette editor or the **Edit...** option. In order to remove the tag library, right click on the library that you wish to remove and chose the **Delete** option or click the **Delete** keyboard button.

Important:

The removal option is enabled only for custom tag libraries.

- **To work with a tag element.** When the library folder is created, new tags can be added to it. Choose the required library and select the **Create** → **Create Macro…** option from the pop-up menu that appears after you right-click on a folder.

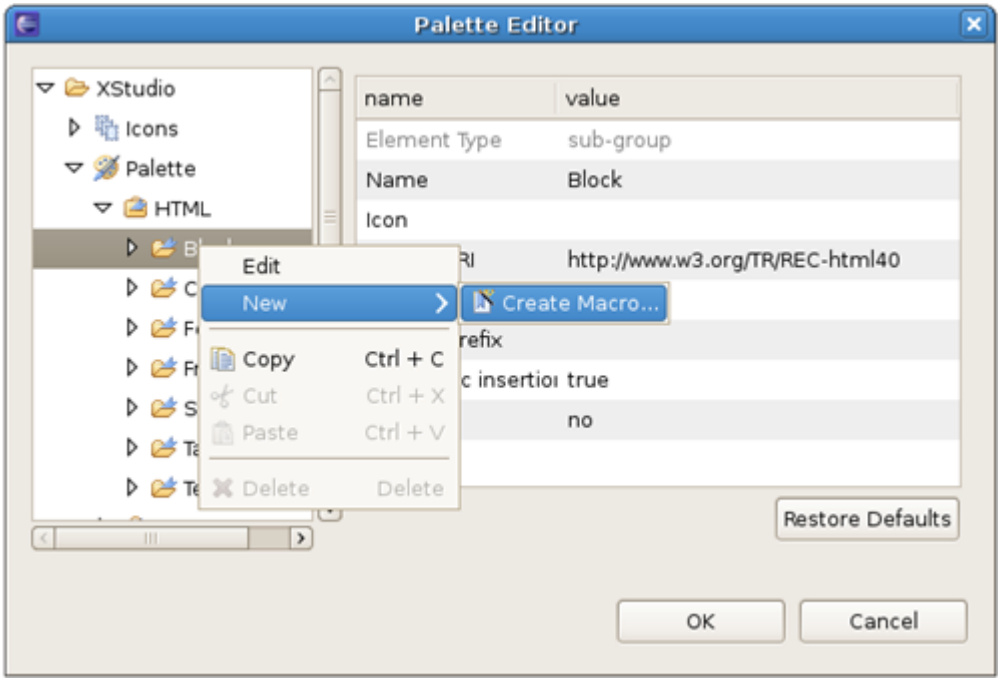

Figure 5.8. Creating a tag element

In the **Add Palette Macro** window you can configure the tag element. The **Name** attribute is mandatory, as it defines the name of the tag element. Other settings are optional. You can choose the icon and set the **Start Text** and the **End Text** for your tag element. If your tag text is too long, use the **Change...** button to see it all. The pipe ("|") symbol can be used to control the cursors position for the **start text** and **end text** values.

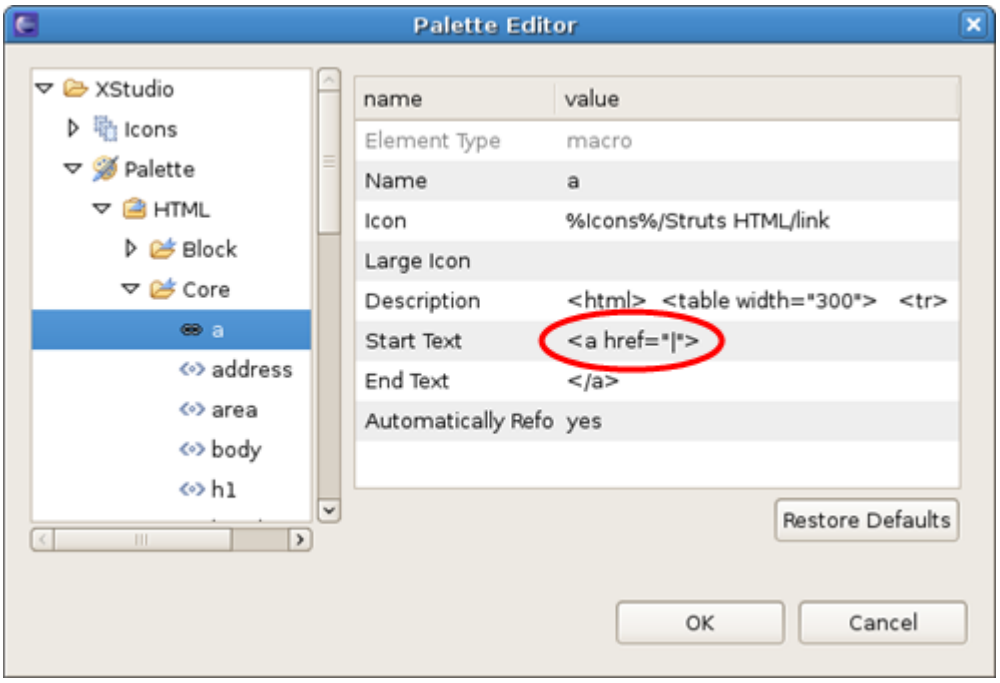

Figure 5.9. Parameters of the Palette element

After all the attributes are set, click the **Finish** button.

Note:

If you do not choose an icon the default one will be assigned.

You are also allowed to edit or delete the tag. If you would like to change the attributes of the tag or choose another icon for it, use the right editing window of the palette editor or the **Edit...** option from the pop-up menu. In order to remove the tag, right click on the tag that you wish to remove and chose the **Delete** option, or click the **Delete** keyboard button.

Important:

The removal option is enabled only for custom tags. JBoss Palette tags can not be removed but can be modified.

If you have changed any object in the tree view and you don't like the final result you can always use the **Restore Defaults** button. Clicking this button will restore defaults for the object selected and for its children elements. Please remember that the button will only restore data for objects defined in the default palette. If selected object is created by you, the button will be disabled. Child objects added by you will not be removed.

When updating JBoss Tools the palette content is not updated.

5.1.2. Show/Hide

Show/Hide is a very useful feature that allows you to control the number of tag groups that are shown on the palette.

- Click the **Show/Hide** button( ), at the top right side of the JBoss Tools Palette.

- In the dialog Show/Hide Drawers check the groups the libraries of which you want to be shown on the palette:

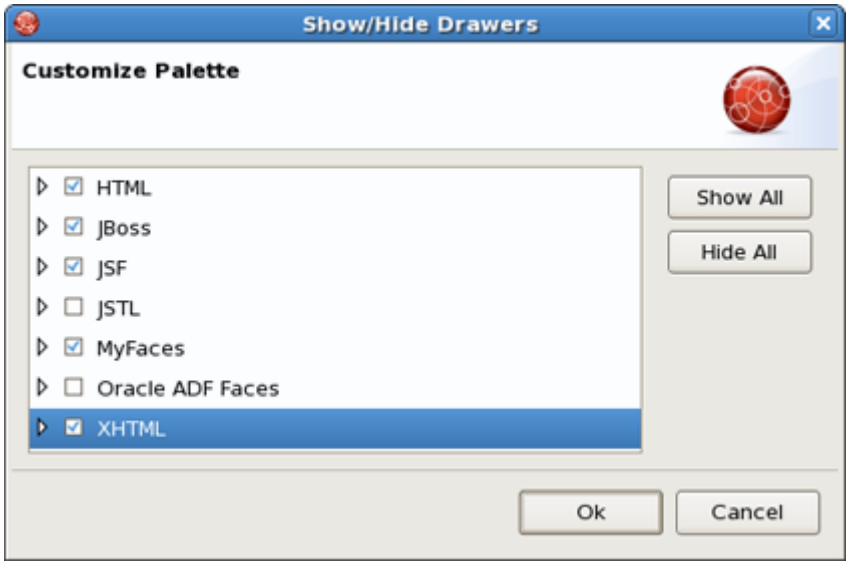

Figure 5.10. Show/Hide Drawers

If libraries are not displayed in the palette, check whether they are selected. Click the plus sign to expand the libraries of the group and make sure that a tick is put next to the wanted libraries.

- Click the **OK** button. The new groups will now be shown on the palette:

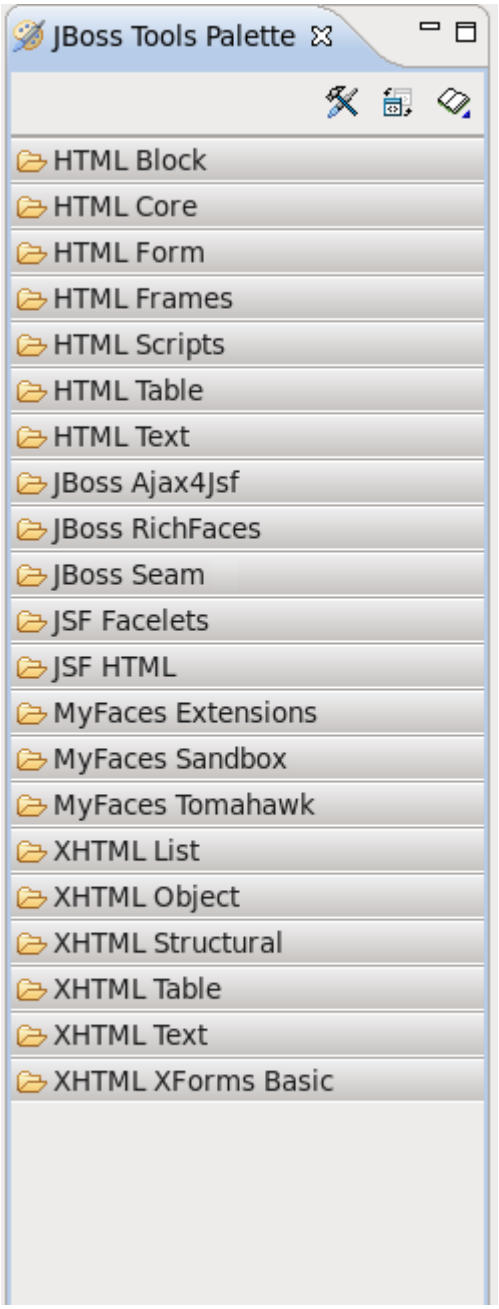

Figure 5.11. New Added Groups

The names of the elements are compound. The first part is the group name and the second is the library name.

5.1.3. Import

The **Import** button lets you add a custom or 3rd party tag library to JBoss Tools Palette. Find out more information on how to add particular tags see the *Section 5.2.2.2, "Import Button"* section.

5.2. Using the Palette

5.2.1. Inserting Tags into a JSP File

A new tag can be added into any text file including `JSP`, `HTM`, `HTML` and `XHTML`.

Open your `JSP` file, place the cursor in a place where you would like to add a tag, and then click that tag in the palette. In the **Insert Tag** window that appears, you can set the value of **general** and **advanced** attributes of the tag that you choose.

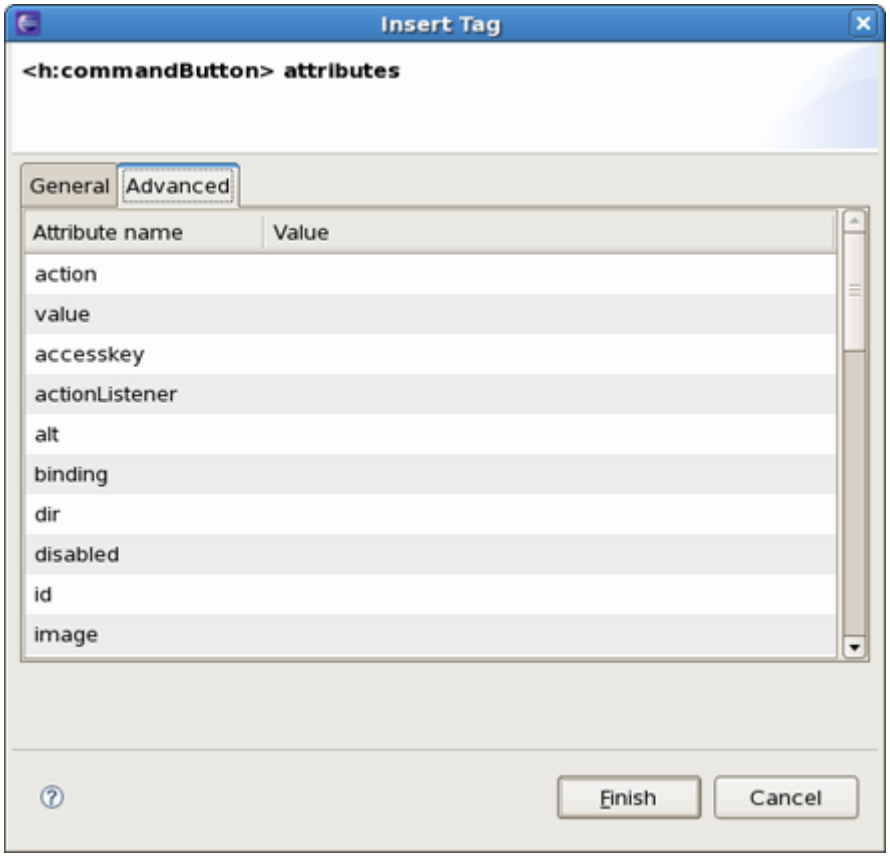

Figure 5.12. Inserting Tag

In the example below the **commandButton** tag has been inserted.

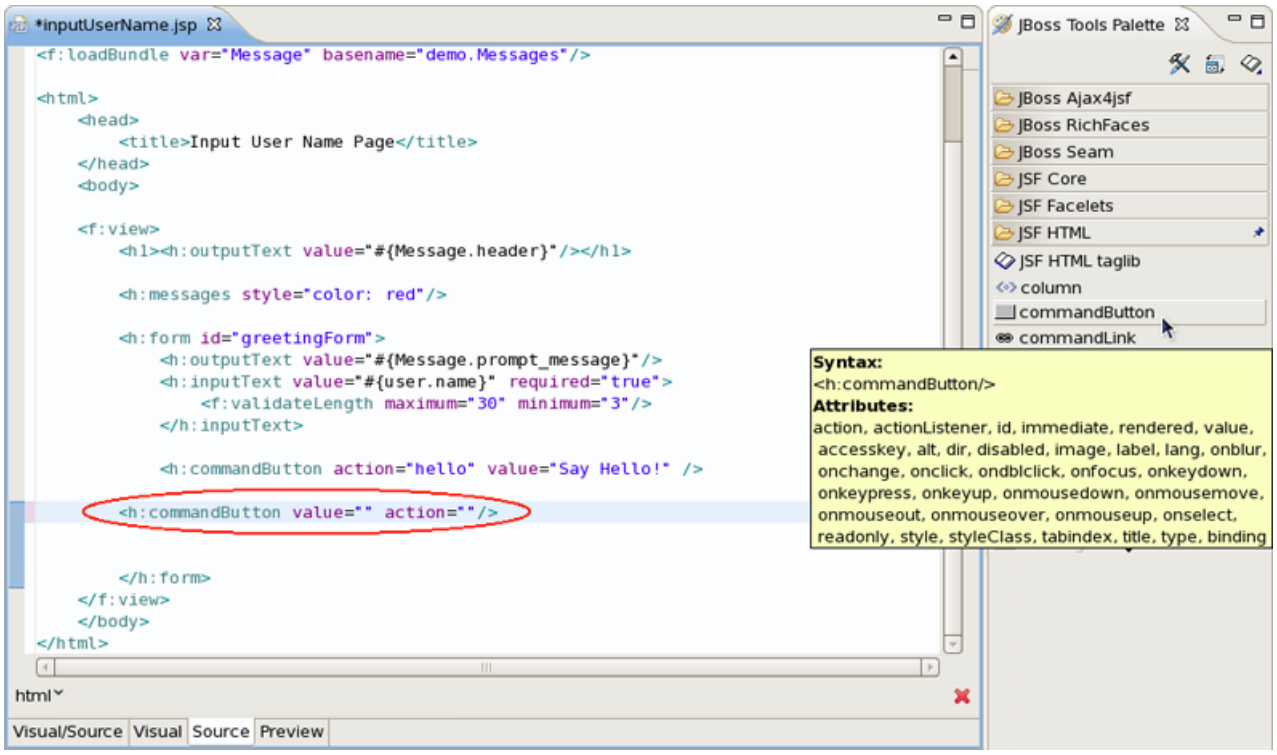

Figure 5.13. Inserting Tag

Tip:

If you place the cursor over any tag, a balloon hint is shown with all the tag attributes.

The cursor position after adding a tag into a file is specified by "|" symbol in the tag template on the right in the Palette Editor window.

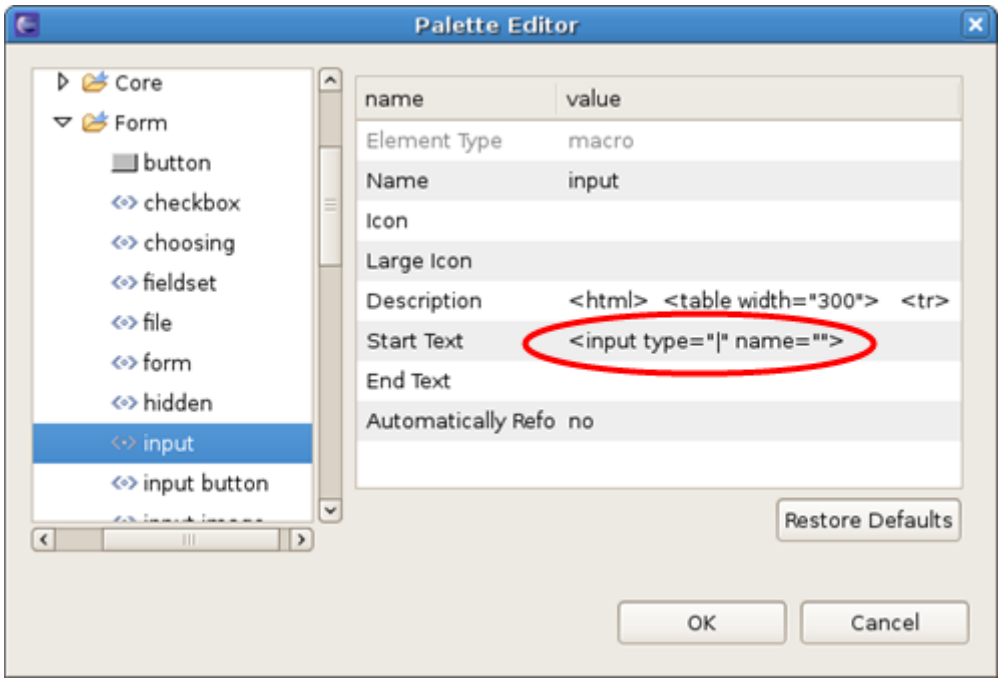

Figure 5.14. Palette Editor

Above you can see where the cursor position for **HTML** → **Form** → **input** is set. After adding this tag into your file the cursor will be in the attribute `type`. At this point you can straight use the **Ctrl**+**Space** keyboard shortcut to display a list of possible values.

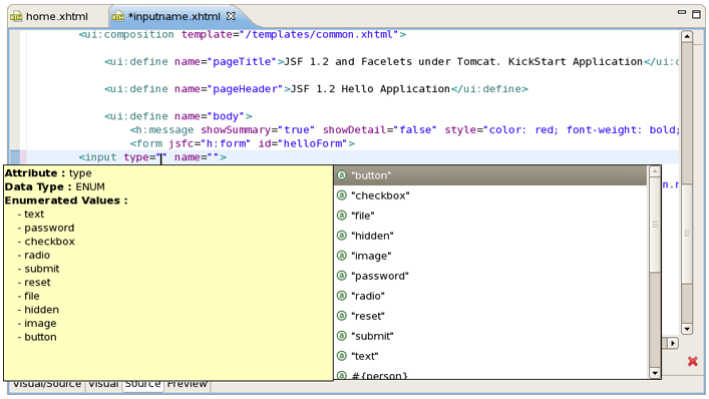

Figure 5.15. Cursor position

5.2.2. Adding Custom JSF Tags to the JBoss Tools Palette

There are two ways to add any custom (including custom Facelets libraries) or 3rd party tag library to the JBoss Tools Palette:

- Drag-and-drop from the Web Projects view

- The Import button on the JBoss Tools Palette

Before you add your custom component library, you need to make sure it is included in your project. You need to either place the `TLD` file or the `JAR` that includes your tag library under the lib folder in your project. Or you can just add `TLD` or `JAR` file to the classpath and the library will be added to the Tag Library List in Web Projects View.

5.2.2.1. Drag-and-Drop

Switch to the **Web Projects** view and expand the **Tag Libraries** folder. If the view is not active, select **Window** → **Show View** → **Web Projects** from the menu bar.

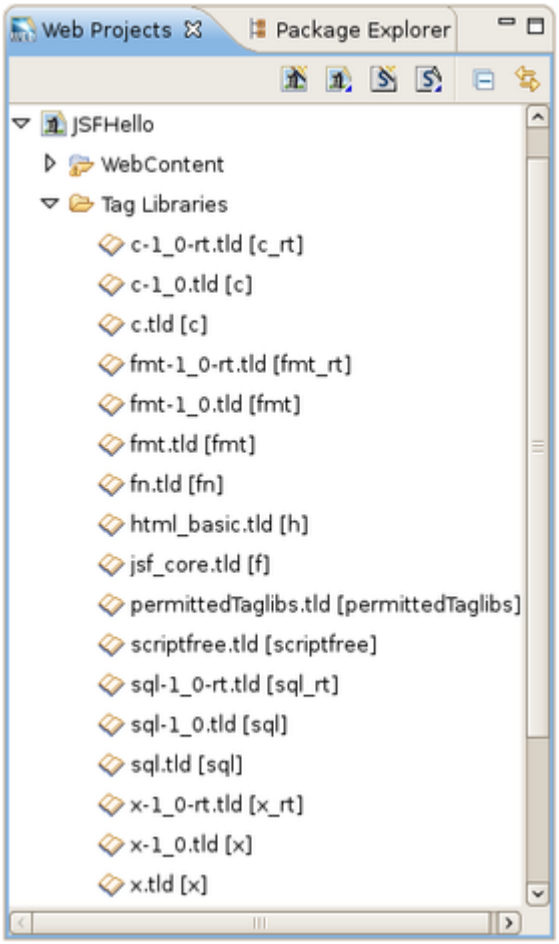

Figure 5.16. Web Projects View

Also make sure that the **JBoss Tools Palette** is open. Select the tag library that you want to add and simply drag-and-drop it on to the the **JBoss Tools Palette**.

You will see the following dialog window. As you can see JBoss Developer Studio takes care of all the details. Chosen **TLD file**, **name** and **prefix** of the library and **Library URL** are detected thus just need to set the **Group name** to which you wish to place this tag library. You can either add this tag library to an existing Group or just create a new one.

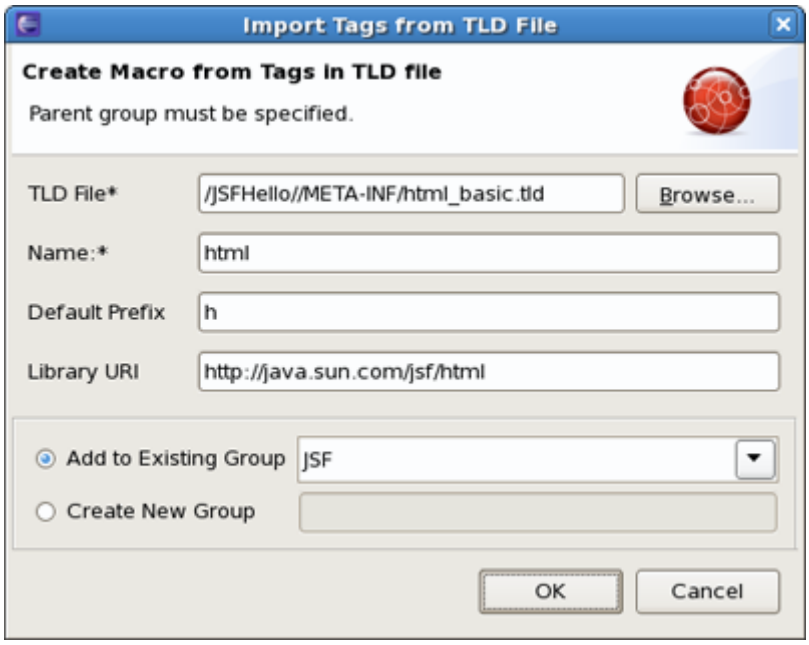

Figure 5.17. Import Tags From TLD File Form

Once you are finished, you will see the new tag library added to the **JBoss Tools Palette**.

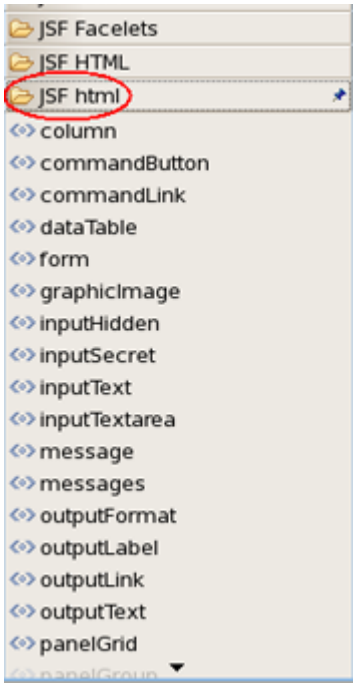

Figure 5.18. JBoss Tools Palette with New Tag Library

5.2.2.2. Import Button

Tag libraries can also be imported with the **Import** button ( ). This button is found at the top right side of the **JBoss Tools Palette**.

By clicking on the **Import** button you will see the **Import Tag** window a similar like in the Drag-and-Drop method (see *Section 5.2.2.1, "Drag-and-Drop"*). Set the name and prefix of the library and Library URL. You also need to set the Group name to which you'd like to add your tag library.

Like in the previous method you can add it to an existing Group or create a new one. On this **Import Tag** form you can use **Browse...** button to locate the tag library that you want to add:

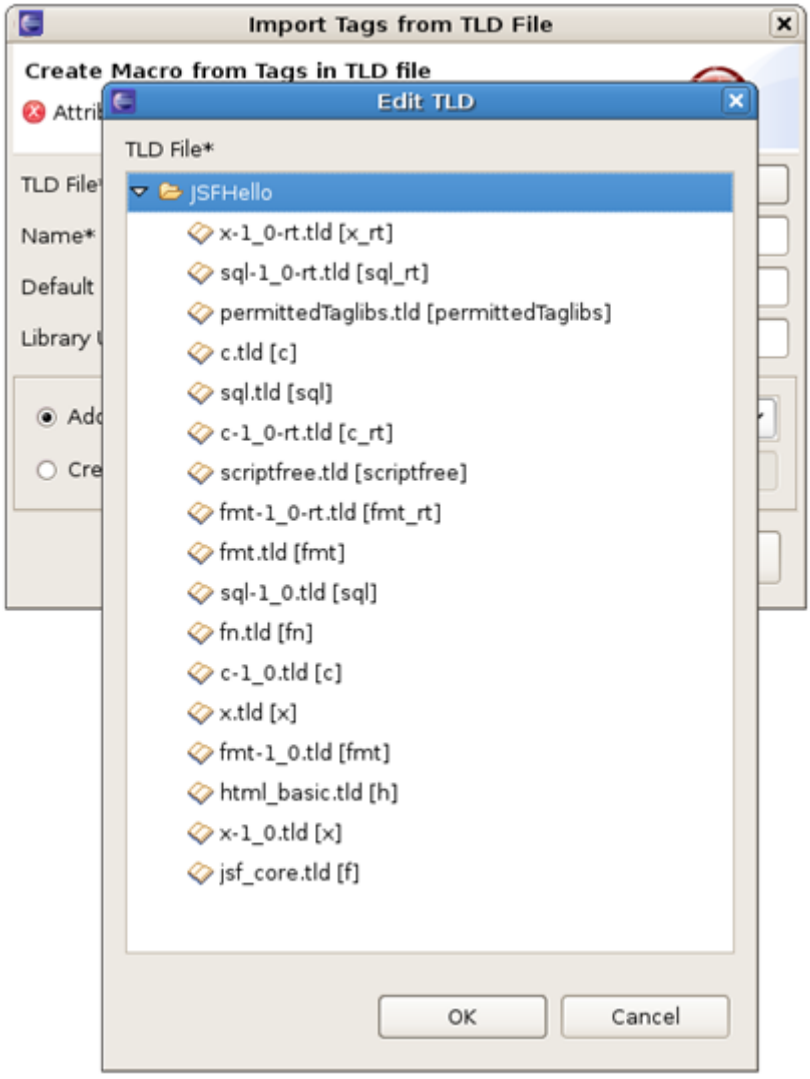

Figure 5.19. Select TLD File

CSS Editing Perspective

In this chapter we will discuss CSS Editing Perspective views. More information about style sheets can be found in *Section 3.2.2, "Pages Styling"* of Editor chapter.

The CSS Editing Perspective combines a set of views which allow you to see the structure of your css files, edit them and see the results. To use this perspective you need to select **Window** → **Open Perspective** → **CSS Editing**. All of the views are fully synchronized with each other: the changes being made in one view are reflected in the others.

As you know there are three ways of inserting a style sheet:

- External style sheet (.css file)
- Internal style sheet (using the `<style>` tag in the head section of an `HTML`/`XHTML`/`JSP` page)
- Inline style (using style attribute)

Using the **CSS Editing** Perspective you can change your style sheet, inserted in any of the possible places described before in three ways:

- directly in your Editor
- using *Section 6.3, "CSS Properties view "*
- using *Section 6.2, "Properties view"*

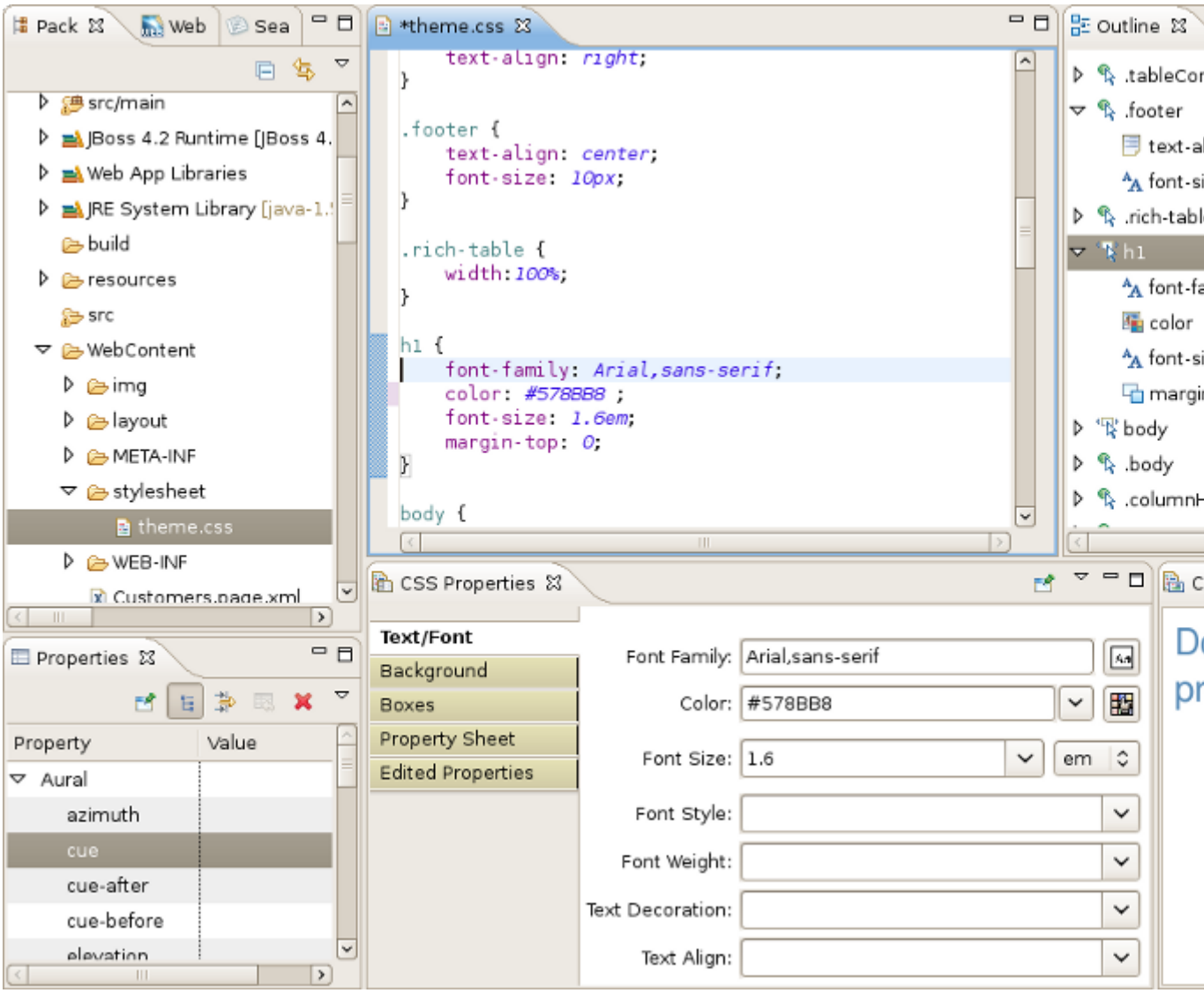

Figure 6.1. CSS Editing Perspective

6.1. Outline view

Using this view you can easily skip between the selectors described in the source files. See the list of properties in any selector just by clicking the triangle near it.

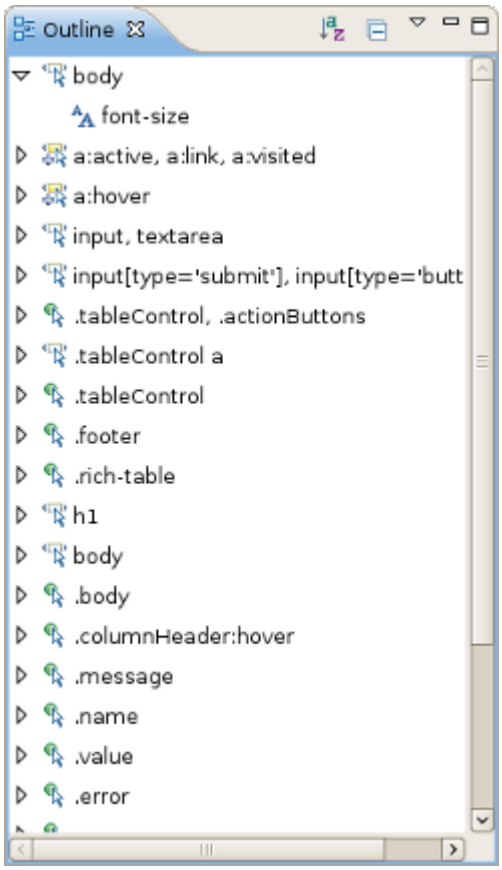

Figure 6.2. Outline view

You can use the **Source** viewer with the **Outline** view to navigate around the file. To do this you should left click the selector or property you want and it will be automatically highlighted in the source code:

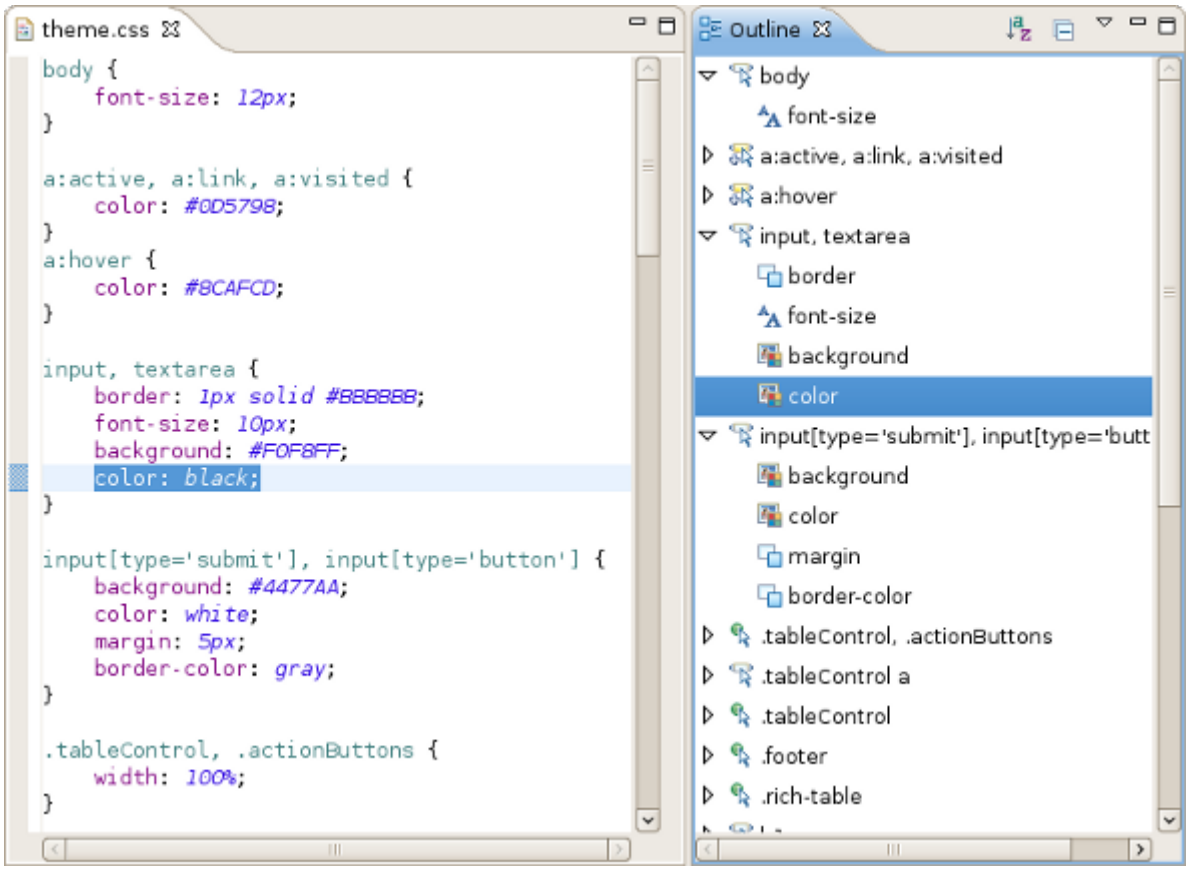

Figure 6.3. Navigating around the file

6.2. Properties view

Properties view provides a full list of properties of a chosen selector. The properties are divided into logic groups for better navigation.

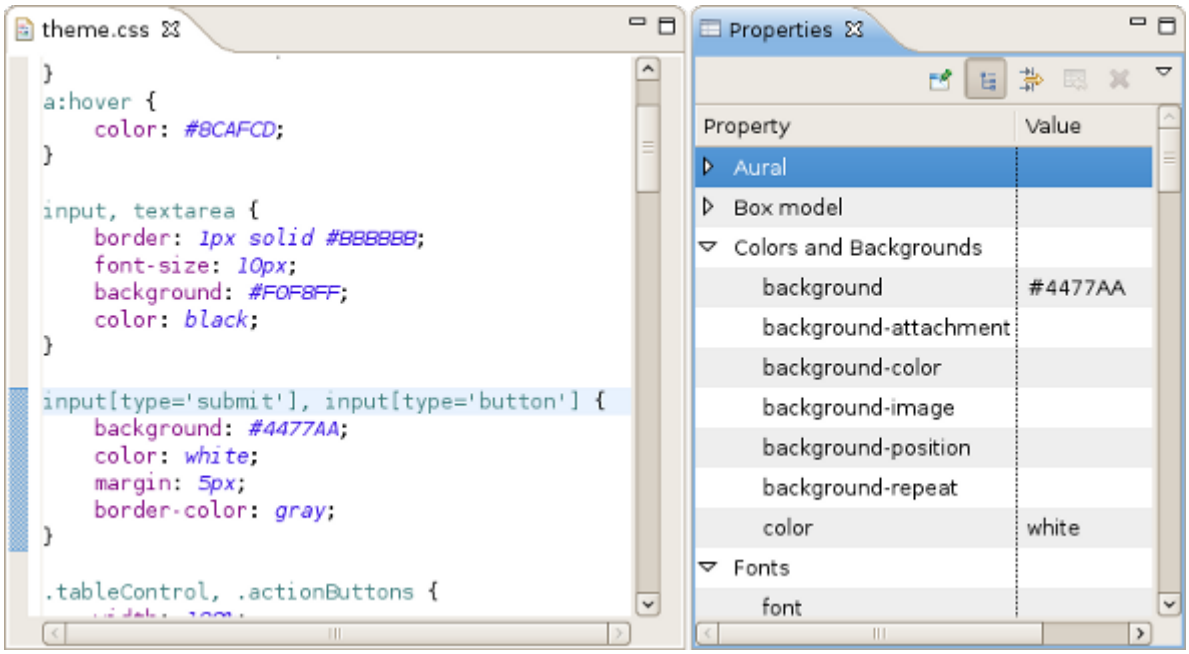

Figure 6.4. Properties view

With the help of **Properties** view you have also the ability to edit the CSS file by adding, editing or removing properties in the selector. Left click the **Value** field near the property you want to edit and write the changes in the text field.

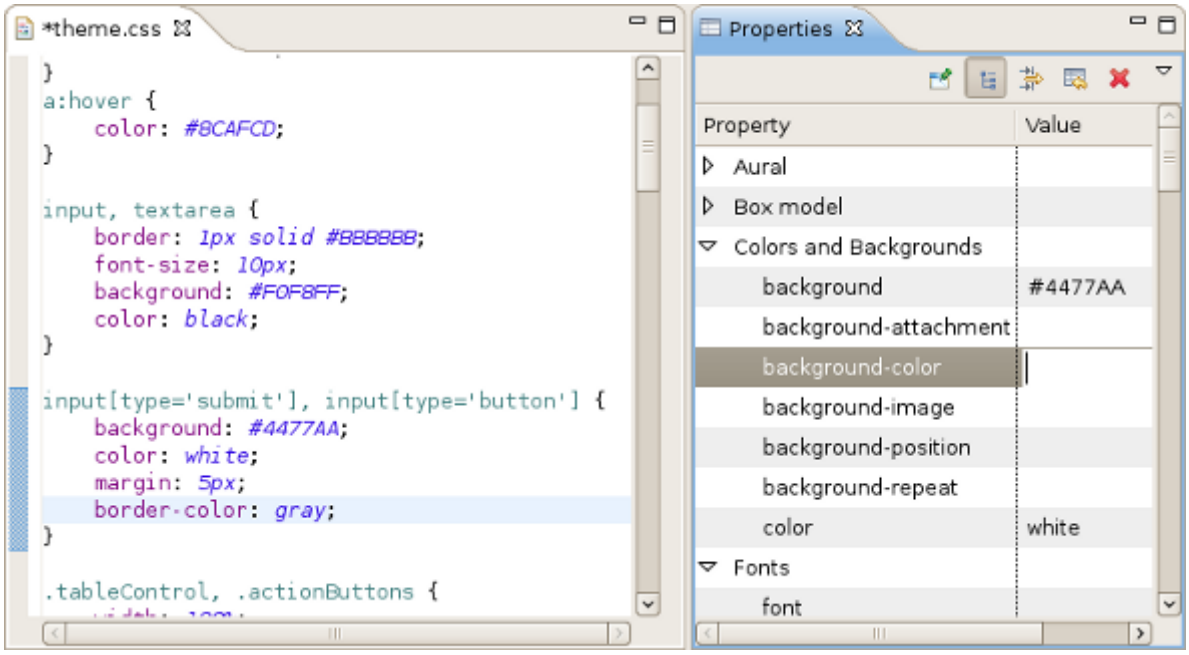

Figure 6.5. Updating css using Properties view

6.3. CSS Properties view

CSS Properties view has five tabs:

- *Text and Font properties*
- *Background properties*
- *Boxes and border properties*
- *Property Sheet*
- *Edited Properties*

- **Text and Font properties.** CSS **Text/Font** properties define the appearance of text, its font family, boldness, size and the style.

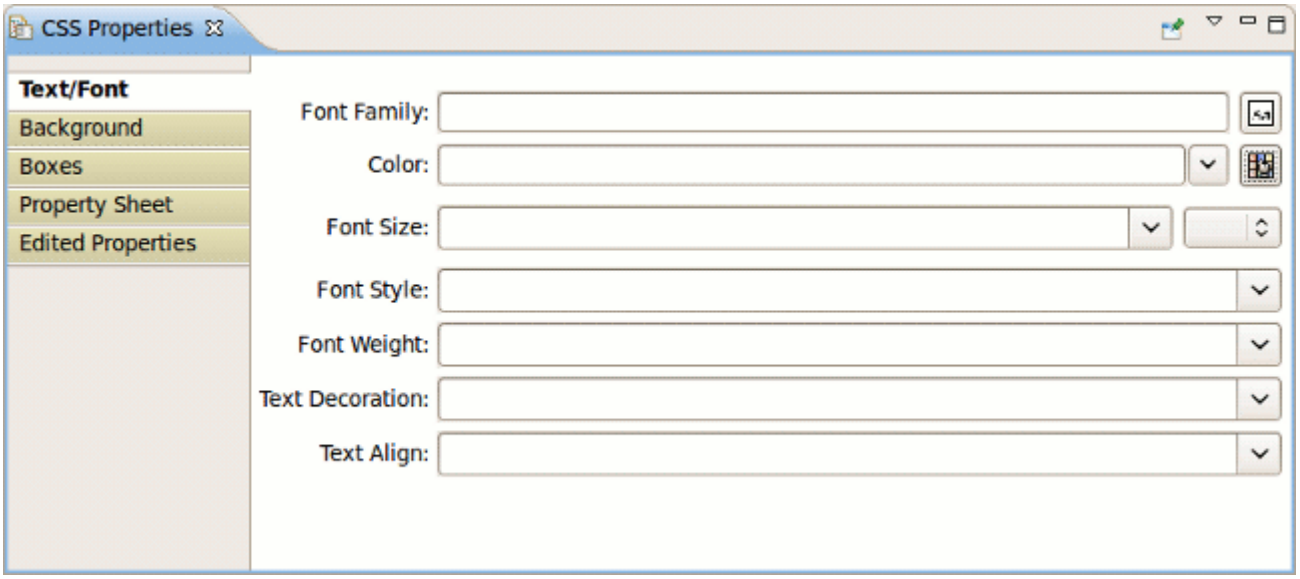

Figure 6.6. Text/Font tab

For example, to define the `font-family` property you should click **Choose font family** button( ) near *Font Family* text field and select the fonts you want to use from the list.

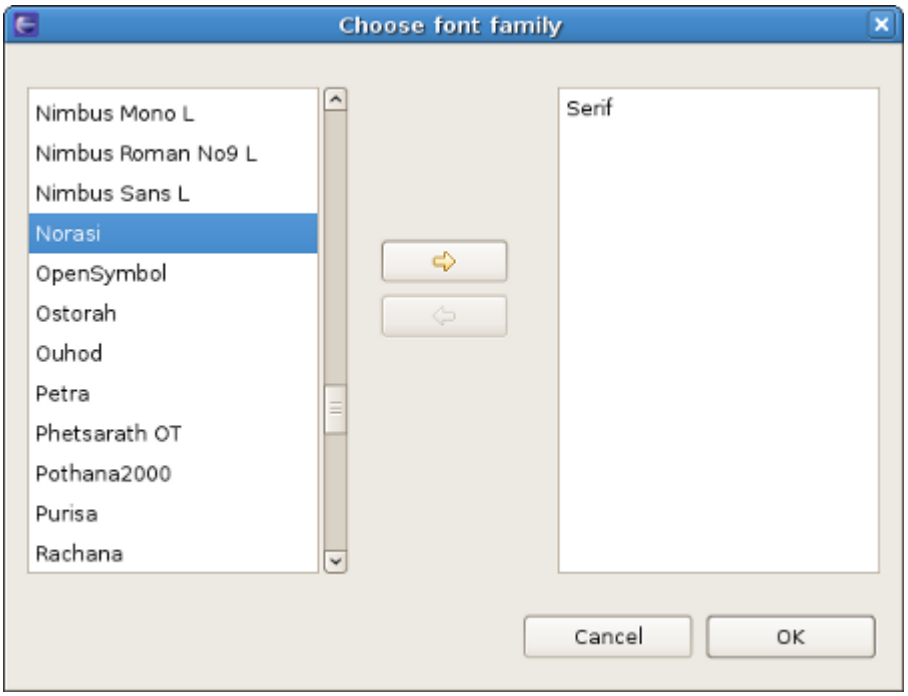

Figure 6.7. Choose font family

When you click the **OK** button the chosen fonts should appear in **Font Family** text field and in the source css file. To define other properties in CSS **Text/Font** tab you should just click button near the corresponding field you want and select the appropriate option in the list. Or if you are absolutely sure of the property's value to use you can just write it in the text field.

- **Background properties.** You should use CSS background properties and **Background** tab to define the background effects of an element.

  Boxes and border properties. The **Boxes** tab is used to define CSS border properties, the box model and dimensions. The CSS border properties allow you to specify the style and color of an element's border.

  As well as in Text/Font tab, it's also possible to define the property in two ways:

  - clicking

    and choosing it from the list of options:

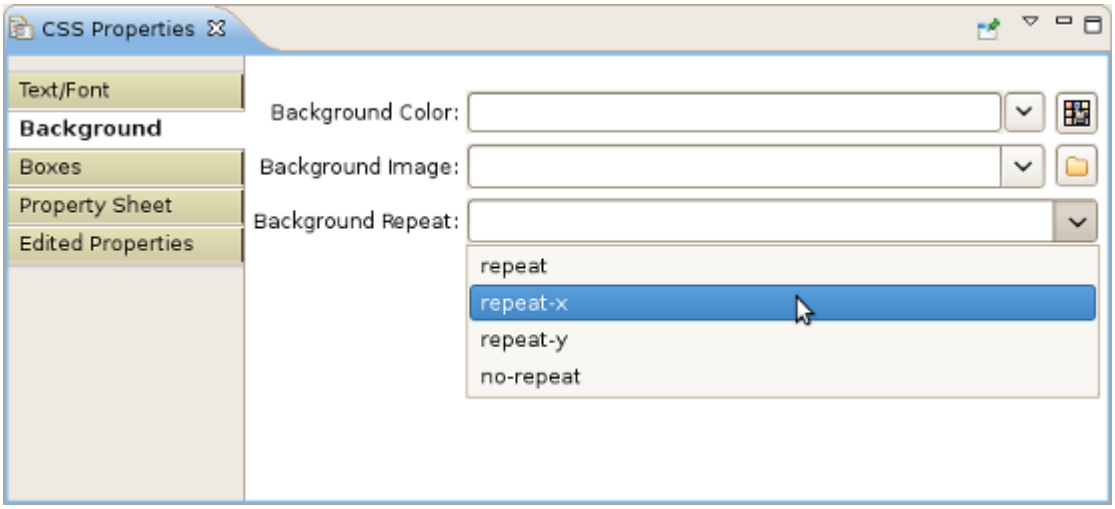

Figure 6.8. Defining the property

- writing the property in the appropriate text field

- **Property Sheet.** The **Property Sheet** tab contains the categorized list of properties. Similarly to *Section 6.2, "Properties view"*, it's possible to edit the properties values.

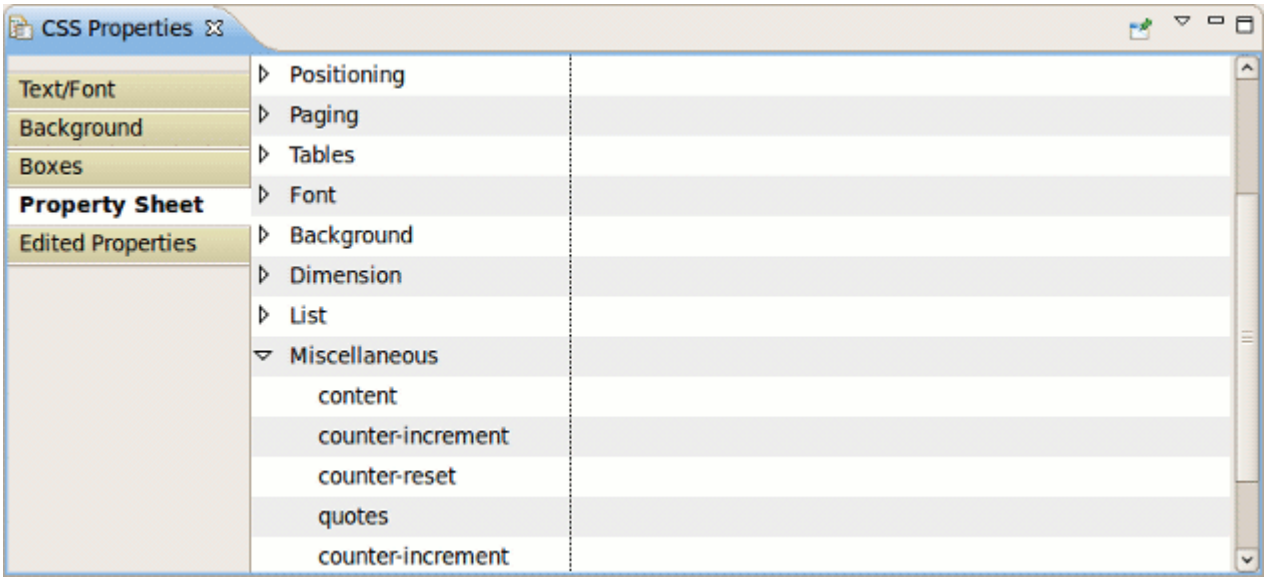

Figure 6.9. Property Sheet tab

- **Edited Properties.** **Edited Properties** tab contains only **overflow-y** property which determines clipping of the element's content at the top and bottom edges.

Figure 6.10. Edited Properties tab

It's also possible to edit the properties in the tab.

6.4. CSS Preview

Using **CSS Preview** you can see how a selector affects any text.

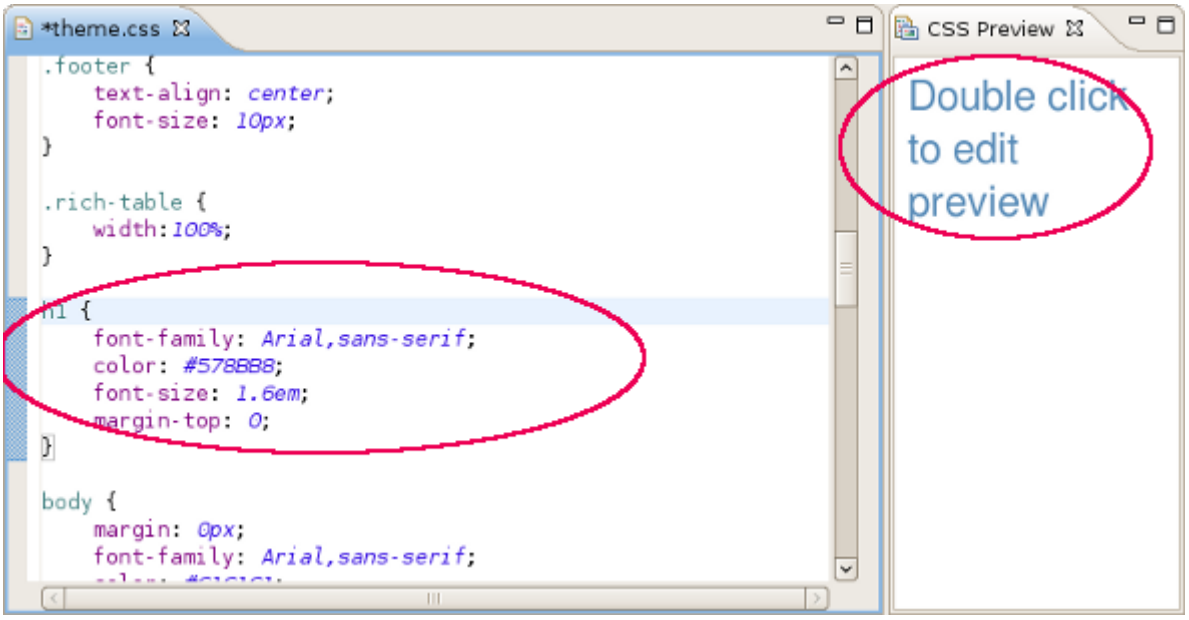

Figure 6.11. CSS Preview

The preview can edited by double clicking on it. You can add any text you want, including HTML tags.

RichFaces Support

JBoss Developer Studio comes with a tight integration with *RichFaces component framework* [http://labs.jboss.com/jbossrichfaces/].

Note:

RichFaces 3.3.X [http://www.jboss.org/jbossrichfaces/downloads/] is fully supported in the current version of JBoss Developer Studio and JBoss Tools 3.2.0.GA.

The following features are implemented and fully supported for the current version of the RichFaces components:

- *Section 7.1, "Code Assist for RichFaces"*
- *Section 7.2, "OpenOn for RichFaces"*
- *Section 7.3, "RichFaces in the JBoss Tools Palette"*

All you have to do is to *download* [http://www.jboss.org/jbossrichfaces/downloads/] and install RichFaces libraries into your project, i. e. just put `richfaces-*.jar` files into the `/lib` project folder. For more information on how to get started with RichFaces, please read the *RichFaces documentation* [http://jboss.org/file-access/default/members/jbossrichfaces/freezone/docs/devguide/en/html_single/index.html#GettingStarted].

7.1. Code Assist for RichFaces

JBoss Developer Studio provides code completion for *RichFaces* [http://www.jboss.org/jbossrichfaces] framework components.

Tip:

RichFaces 3.3.X is now fully supported in code completion.

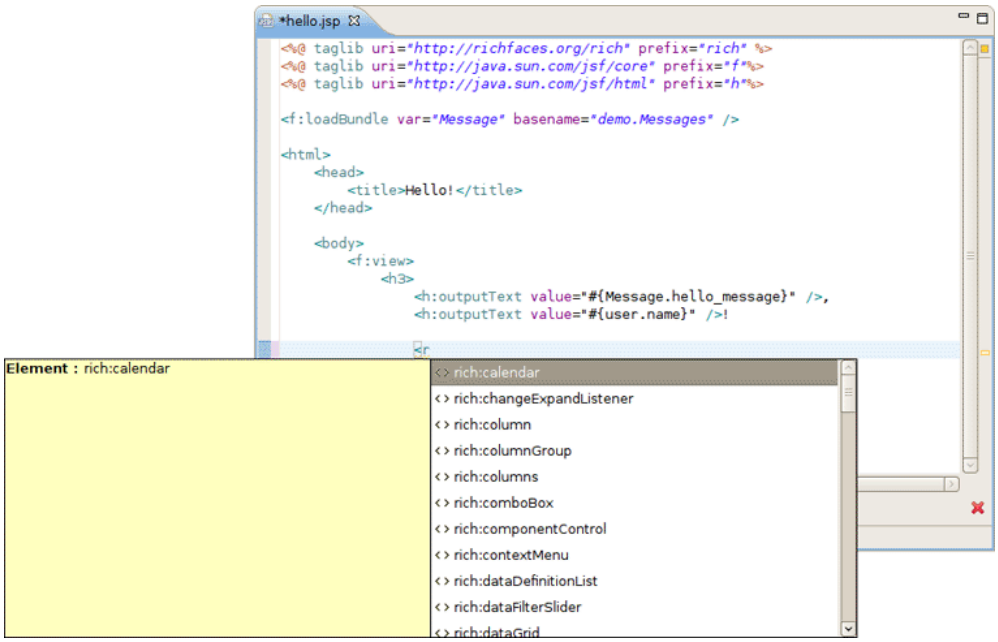

Figure 7.1. Content Assist for RichFaces Components

7.2. OpenOn for RichFaces

While working with JSP and XHTML pages in the Visual Page Editor you can also take the advantage of the *OpenOn* feature with RichFaces components.

For example, the Richfaces tags `<rich:insert>` and `<a4j:include>` have OpenOn support.

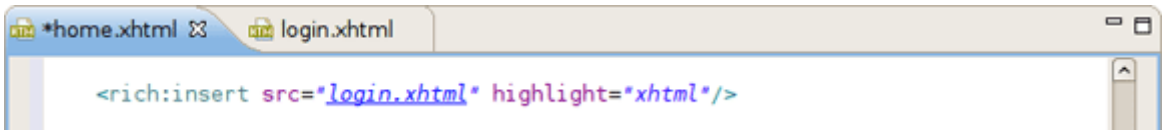

Figure 7.2. OpenOn With Richfaces Tag

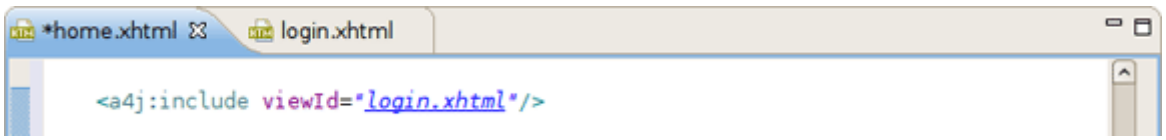

Figure 7.3. OpenOn With A4j Tag

OpenOn is also supported in "ForID"-like attributes (the attributes, where the value should be ID or the list of IDs) in RichFaces.

Figure 7.4. OpenOn With "ForID"-like attributes

7.3. RichFaces in the JBoss Tools Palette

RichFaces and *Chapter 5, JBoss Tools Palette*.

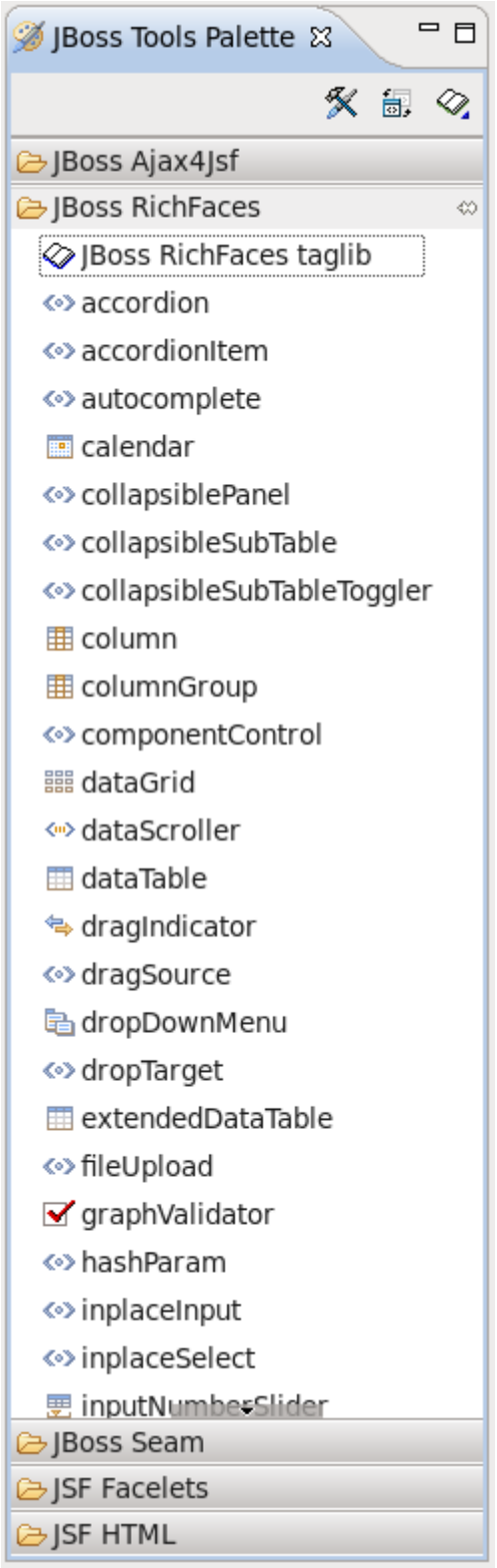

Figure 7.5. RichFaces Components

The **JBoss RichFaces** menu in the **JBoss Tools Palette** contains all items relevant for RichFaces 4, by default. If you are using RichFaces 3 or earlier, you will need to add the **RichFaces 3**

menu item to the **JBoss Tools Palette** through the **Show/Hide** menu. To access this menu see *Section 5.1.2, "Show/Hide"*.

Figure 7.6. RichFaces 3 palette menu

To insert a RichFaces component on a page:

- expand **JBoss RichFaces** group on the palette
- click on some component
- put the needed attributes in the **Insert Tag** dialog and click **Finish** button

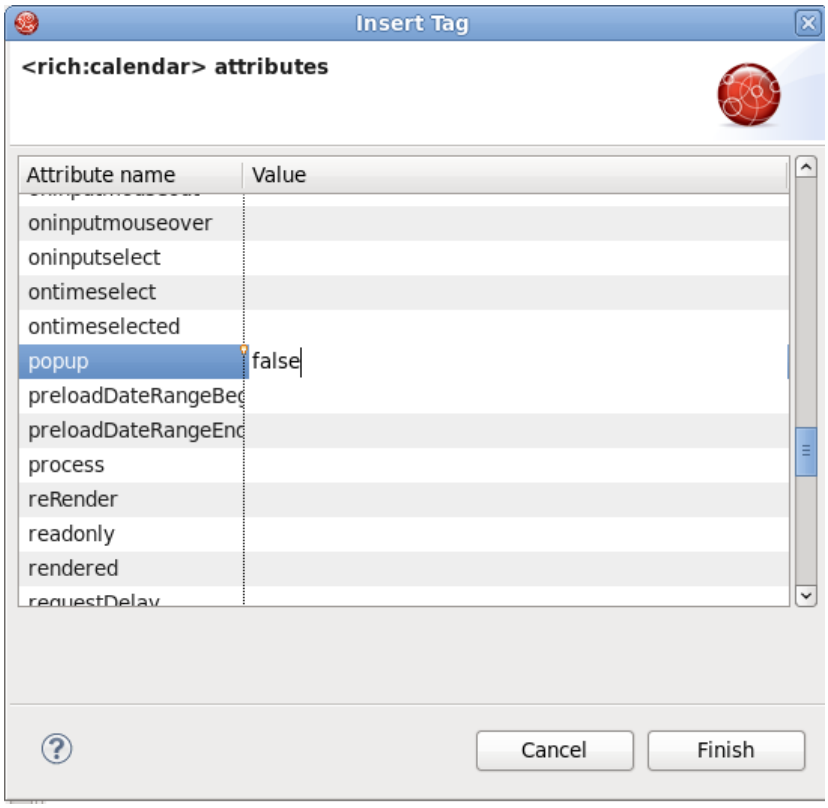

Figure 7.7. Inserting Tag

The RichFaces component will be inserted on your page and displayed in the **Source** and **Visual** modes:

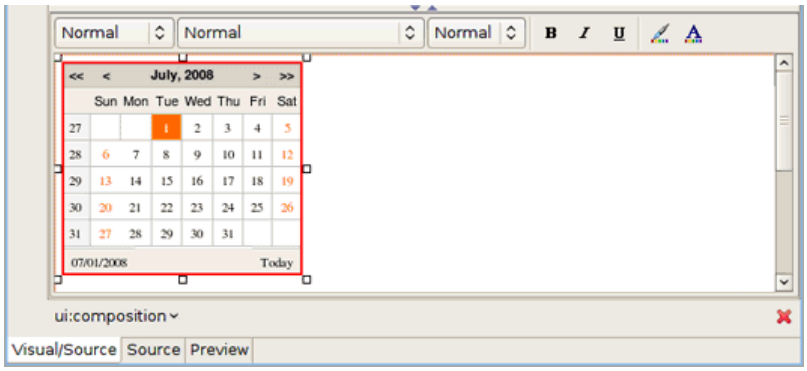

Figure 7.8. RichFaces Component

7.4. Relevant Resources Links

To get more in-depth information on RichFaces framework refer to the *RichFaces Developer Guide* [http://jboss.org/file-access/default/members/jbossrichfaces/freezone/docs/devguide/en/html_single/index.html].

It may be also helpful for you to view the *movies* [http://docs.jboss.org/tools/movies/] that demonstrate the usage of RichFaces components.

Web Projects View

Web Projects is a special view that comes with JBoss Developer Studio.

If the **Web Projects** view's tab is not visible next to the Package Explorer tab, select **Window** → **Show View** → **Other** → **JBoss Tools Web** → **Web Projects** from the menu bar.

With the Web Projects view you can:

- Visualize the project better because the project artifacts for JSF and Seam projects are organized and displayed by function.
- Select these kinds of items to drag and drop into JSP pages:
  - JSF managed bean attributes
  - JSF navigation rules outcomes
  - Property file values
  - Tag library files
  - Tags from tag libraries
  - JSP page links
- Use context menus to develop the application (all create and edit functions are available)
- Use icon shortcuts to create and import JSF projects
- Expand and inspect tag library files
- *Section 5.2.2, “Adding Custom JSF Tags to the JBoss Tools Palette”*

8.1. Project Organization

The Web Projects view organizes your project in a different way. The physical structure of course stays the same. The new organization combines common project artifacts together which makes it simpler to locate what you are looking for and develop.

The screen shot below shows a JSF project in **Web Projects** view.

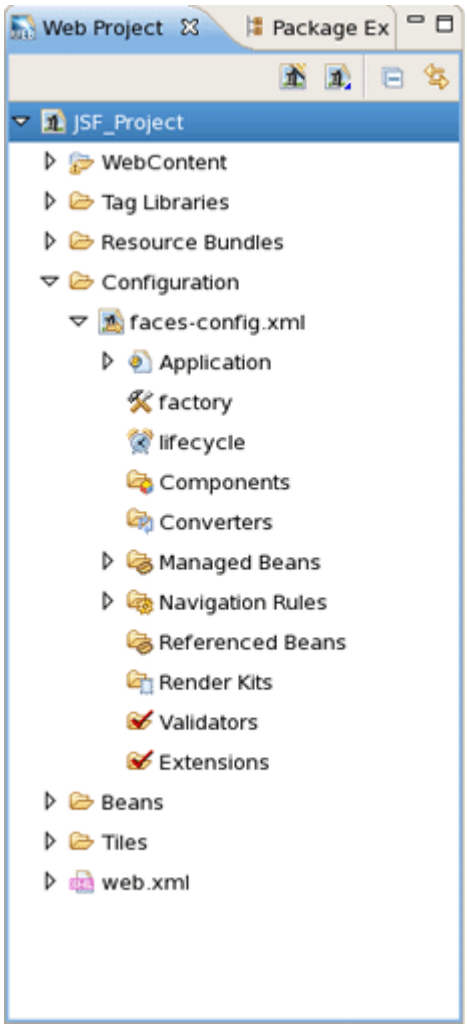

Figure 8.1. Web Projects View

8.2. Drag and Drop

The **Web Projects** view has a drag and drop option that can be used for property, managed bean attributes, navigation rules, tag library file declaration and JSP Pages.

8.2.1. For a Property

Expand the Resources Bundles folder that holds all the Property files in your project. Select the file from which you want to add the property and then select the property.

We will be dragging and dropping a property file value inside the outputText tag for the `value` attribute.

```
<html>
    <head>
            <title>Input User Name Page</title>
    </head>
    <body>

    <f:view>
            <h1><h:outputText value=""/></h1>
```

Figure 8.2. OutputText Tag

Select the property:

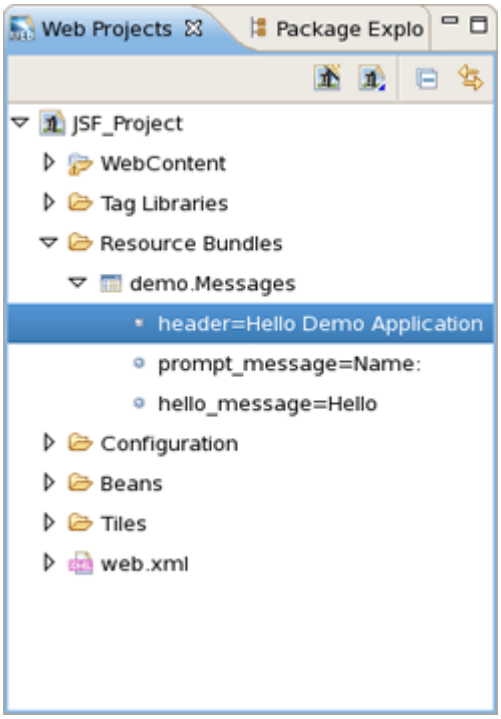

Figure 8.3. Selecting Property

Drag the property and drop it between the quotes for the value attribute in the JSP file. Notice that JBoss Developer Studio added the correctly formatted expression for referring to the property value `#{Message.header}` automatically.

```
<html>
    <head>
        <title>Input User Name Page</title>
    </head>
    <body>

    <f:view>
        <h1><h:outputText value="#{Message.header}"/></h1>

        <h:messages style="color: red"/>
```

Figure 8.4. Inserted Property

You can actually place the tag anywhere in the page, not just inside an existing tag. In this case JBoss Developer Studio will place the complete tag `<h:outputText value="#{Message.header}"/>` in the page.

8.2.2. For Managed Bean Attributes

Select a "managed bean" attribute and then drag and drop it onto the JSP page. We are going to place it inside the `value` attribute of the `inputText` tag.

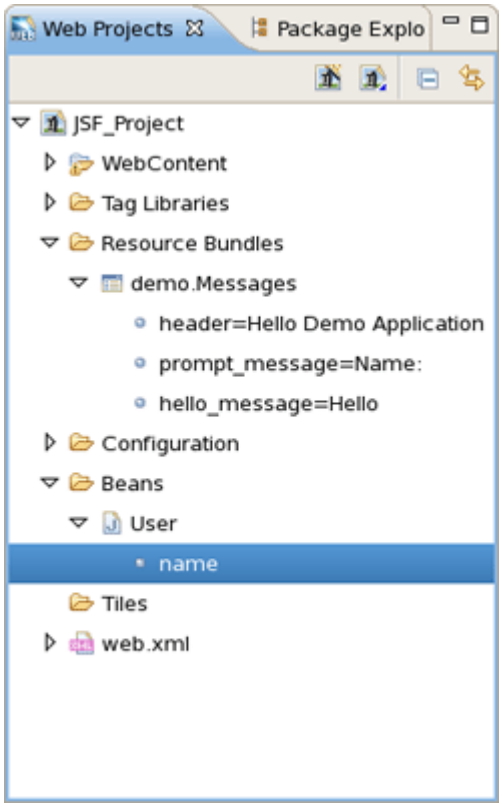

Figure 8.5. Selecting Managed Bean Attribute

Once again, JBoss Developer Studio adds the correct expression, `#{user.name}`.

```
<h:form id="greetingForm">
    <h:outputText value="#{Message.prompt_message}"/>
    <h:inputText value="#{user.name}" required="true">
        <f:validateLength maximum="30" minimum="3"/>
    </h:inputText>
```

Figure 8.6. Added Expression

8.2.3. Navigation Rules

Select the navigation rule under **Configuration** → **faces-config.xml** → **Navigation Rules**:

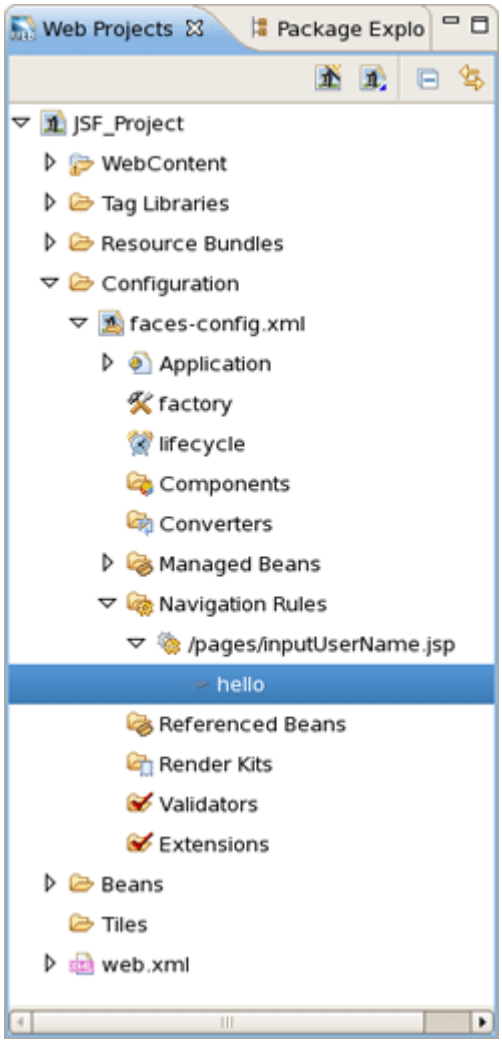

Figure 8.7. Selecting Navigation Rule

Drag and drop it inside the `commandButton` tag:

```
            <f:validateLength maximum="30" minimum="3"/>
        </h:inputText>

        <h:commandButton action="hello" value="Say Hello!" />

    </h:form>
</f:view>
```

Figure 8.8. Navigation Rule in CommandButton Tag

You could do the same if the navigation rule was defined inside an action method:

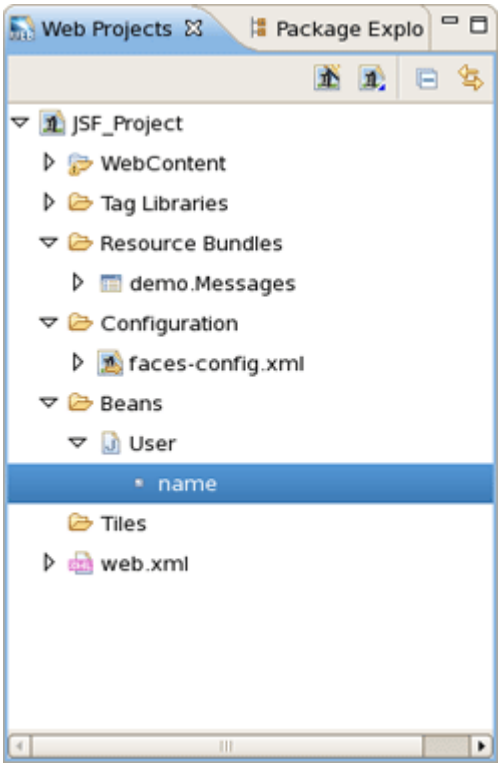

Figure 8.9. Navigation Rule in Action Method

Here is how it would look after drag and drop:

```
            <f:validateLength maximum="30" minimum="3"/>
        </h:inputText>

        <h:commandButton action="#{user.name}" value="Say Hello!" />

    </h:form>
```

Figure 8.10. Inserted Navigation Rule

8.2.4. For a Tag Library File Declaration

Select a TLD file:

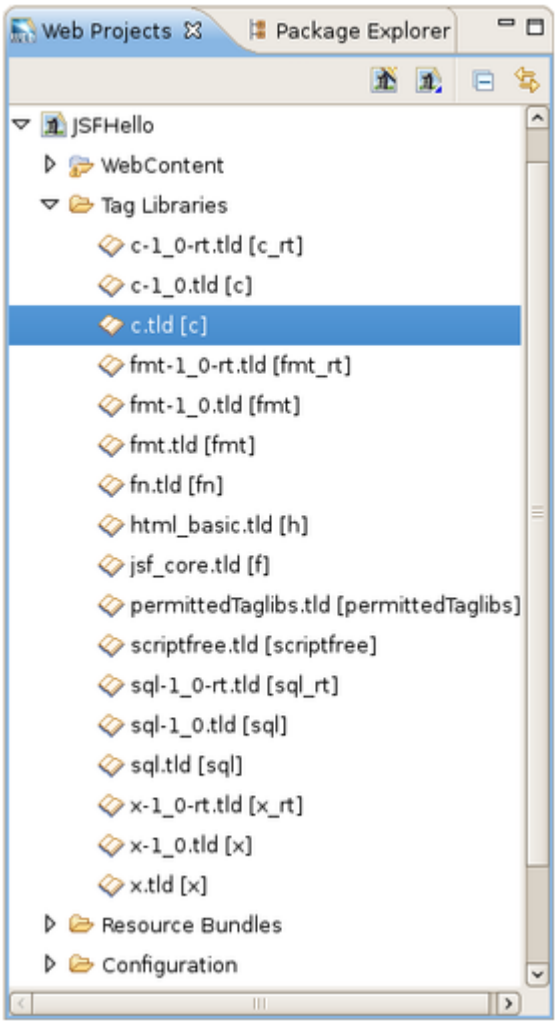

Figure 8.11. Selecting TLD File

Then drag and drop it onto the JSP page to add a declaration at the top of the page:

Figure 8.12. Inserted TLD File

8.2.5. For JSP Pages

You can also drag and drop a JSP page path to a JSP page to create a forward as shown:

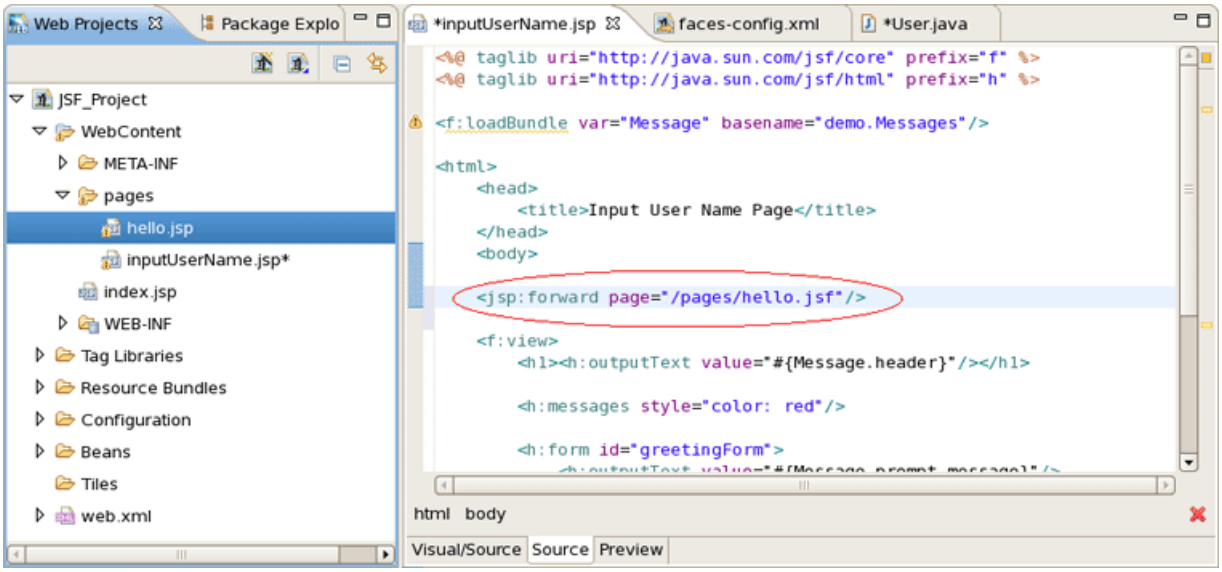

Figure 8.13. Creating JSP Forward

8.3. Developing the Application

It is also possible to develop your application right from the Web Projects view. Simply right-click any node in the tree and select an appropriate action from the context menu. For instance, this screen capture shows creating a new navigation rule.

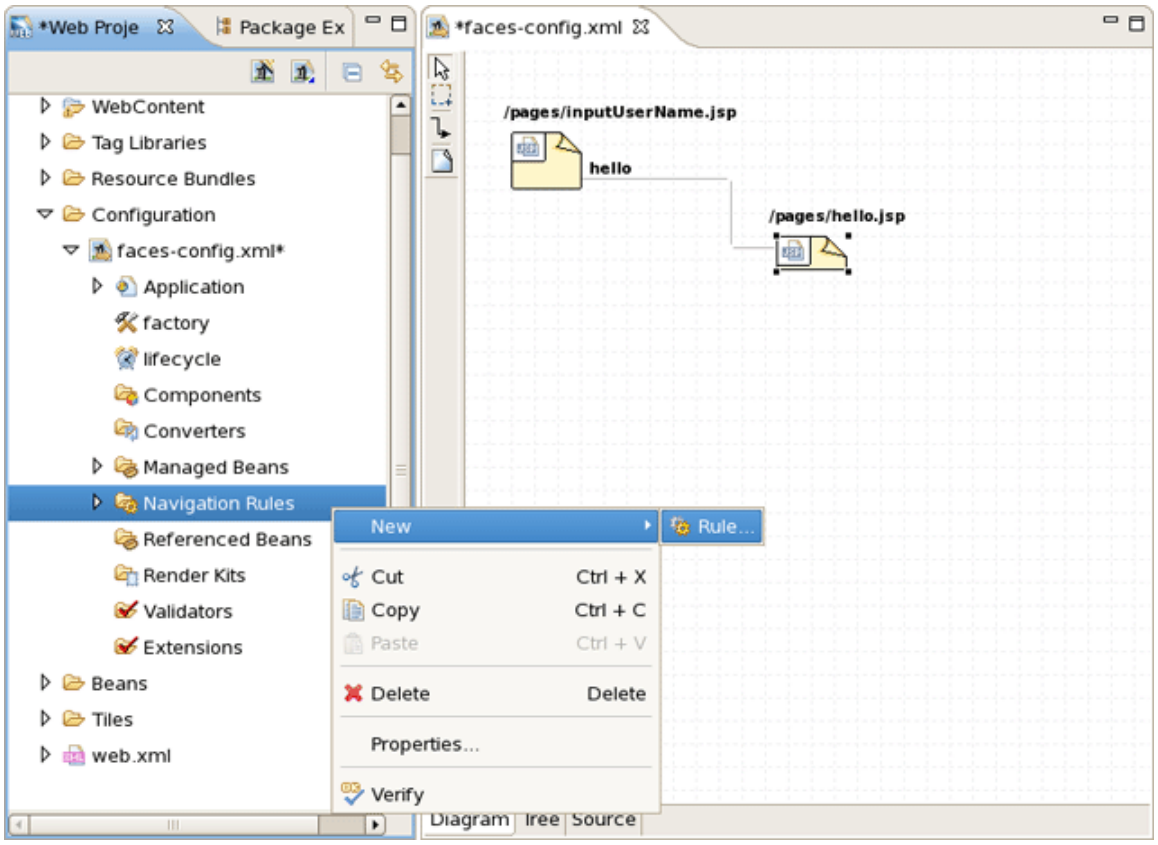

Figure 8.14. Creating New Navigation Rule

8.4. Expanding Tag Library Files

You can easily expand any `TLD` file in the project. Browse to the Tag Libraries folder. Right-click a `TLD` file and select **Expand**:

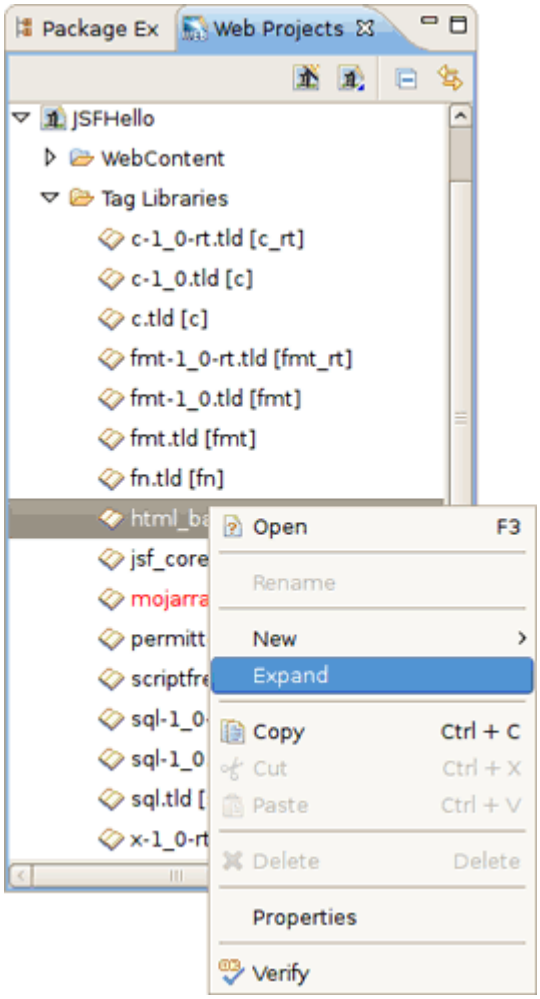

Figure 8.15. Expanding Tag Library File

The TLD file will now be expanded:

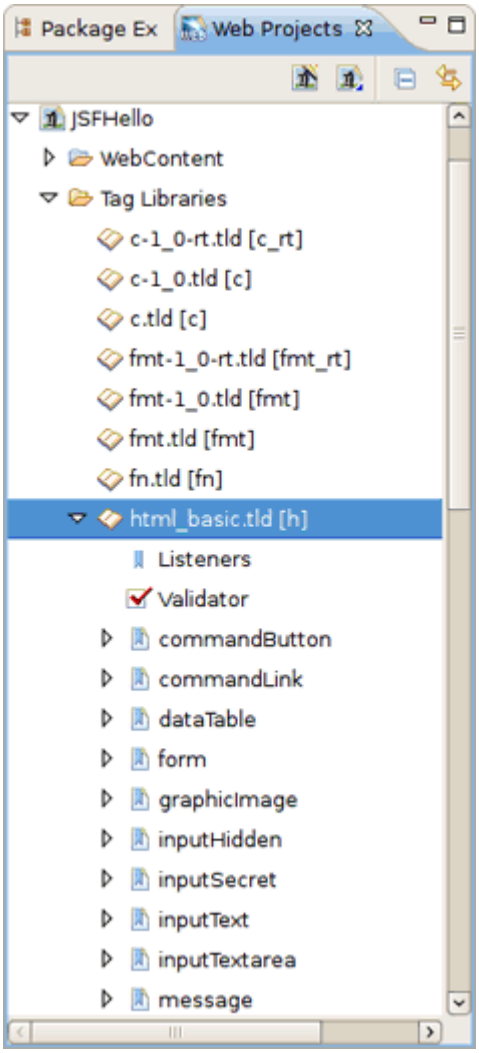

Figure 8.16. Expanded File

You can then select any tag and drag it onto a JSP page.

8.5. Drag and Drop Tag Libraries on to JBoss Tools Palette

Read *Section 5.2.2, "Adding Custom JSF Tags to the JBoss Tools Palette"* to learn about this.

8.6. Create and Import JSF Projects

You can also create and import JSF project from Web Projects view by selecting the buttons below.

From left to right:

1. Create New JSF Project
2. Import JSF Project

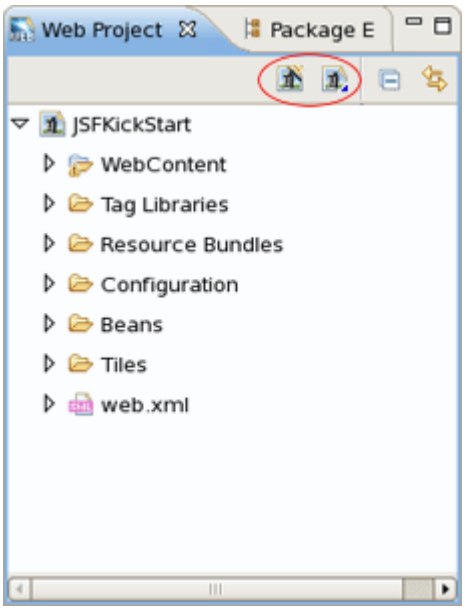

Figure 8.17. Web Projects View Buttons

JBoss Tools Preferences

Configuring the various JBoss Developer Studio features is done via the **Preferences** screen by selecting **Window** → **Preferences** → **JBoss Tools** from the menu bar.

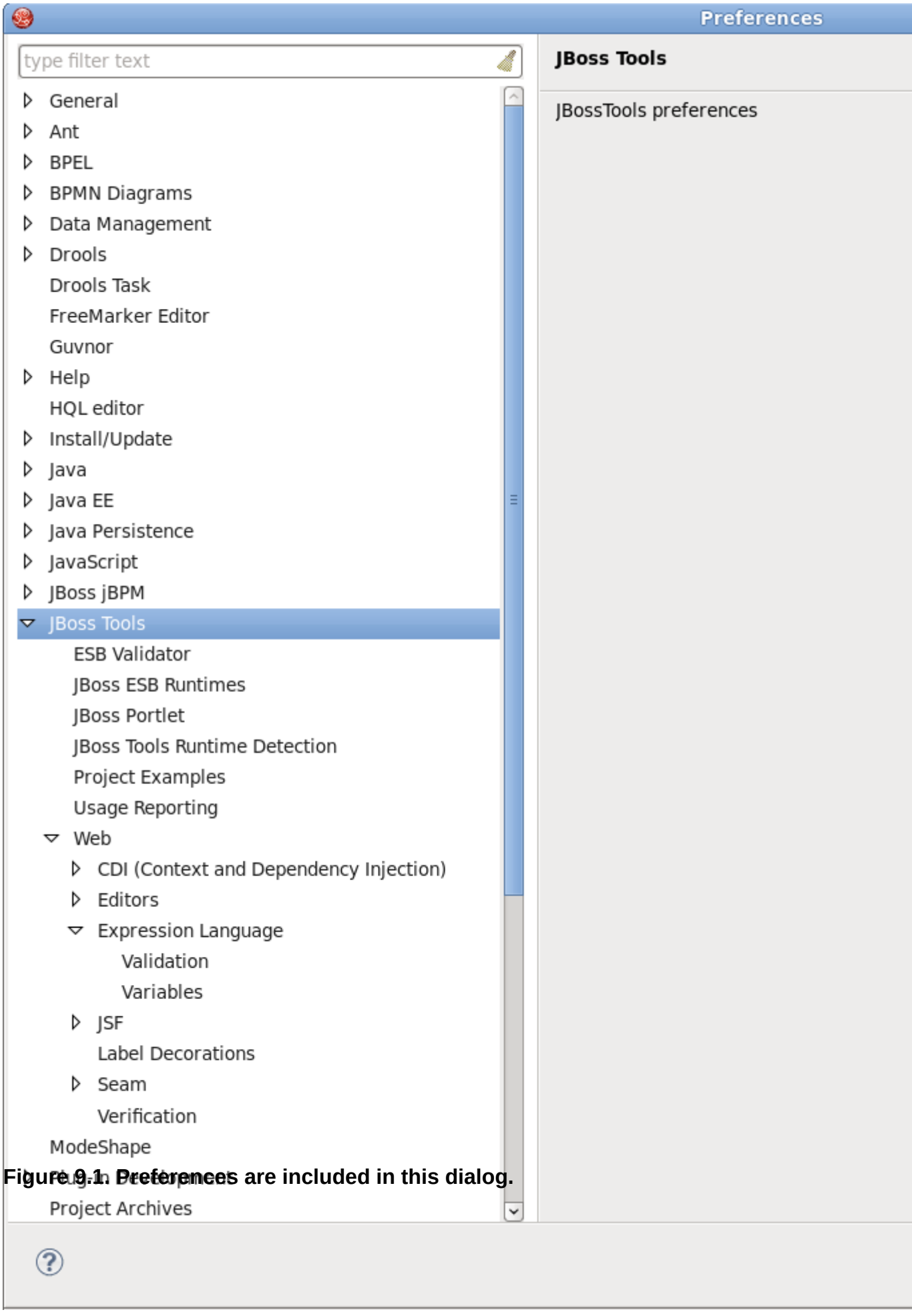

Figure 9.1. Preferences are included in this dialog.

From this screen, you can select these more specific sets of JBoss Tools preferences:

- *Section 9.1, “Project Archives”*
- *Section 9.2, “Editors”*
- *Section 9.3, “Visual Page Editor”*
- *Section 9.4, “Visual Page Editor Code Templates”*
- *Section 9.5, “EL Variables”*
- *Section 9.6, “JSF”*
- *Section 9.7, “JSF Project”*
- *Section 9.8, “JSF Flow Diagram”*
- *Section 9.9, “Label Decorations”*
- *Section 9.10, “Seam”*
- *Section 9.11, “Seam Validator”*
- *Section 9.12, “Seam Pages Diagram”*
- *Section 9.13, “Tiles Diagram”*
- *Section 9.14, “Server Preferences”*
- *Section 9.15, “XDoclet”*

The **Preferences** dialog (**Window** → **Preferences**) also allows to adjust settings for *Section 9.14, “Server Preferences”* and *Section 9.15, “XDoclet”* module.

9.1. Project Archives

Click on the **Project Archives** to open the page for changing Project Archives preferences.

Here you can determine settings for Core Preferences, Project Archives View, Project Explorer Preferences and Fileset Preferences.

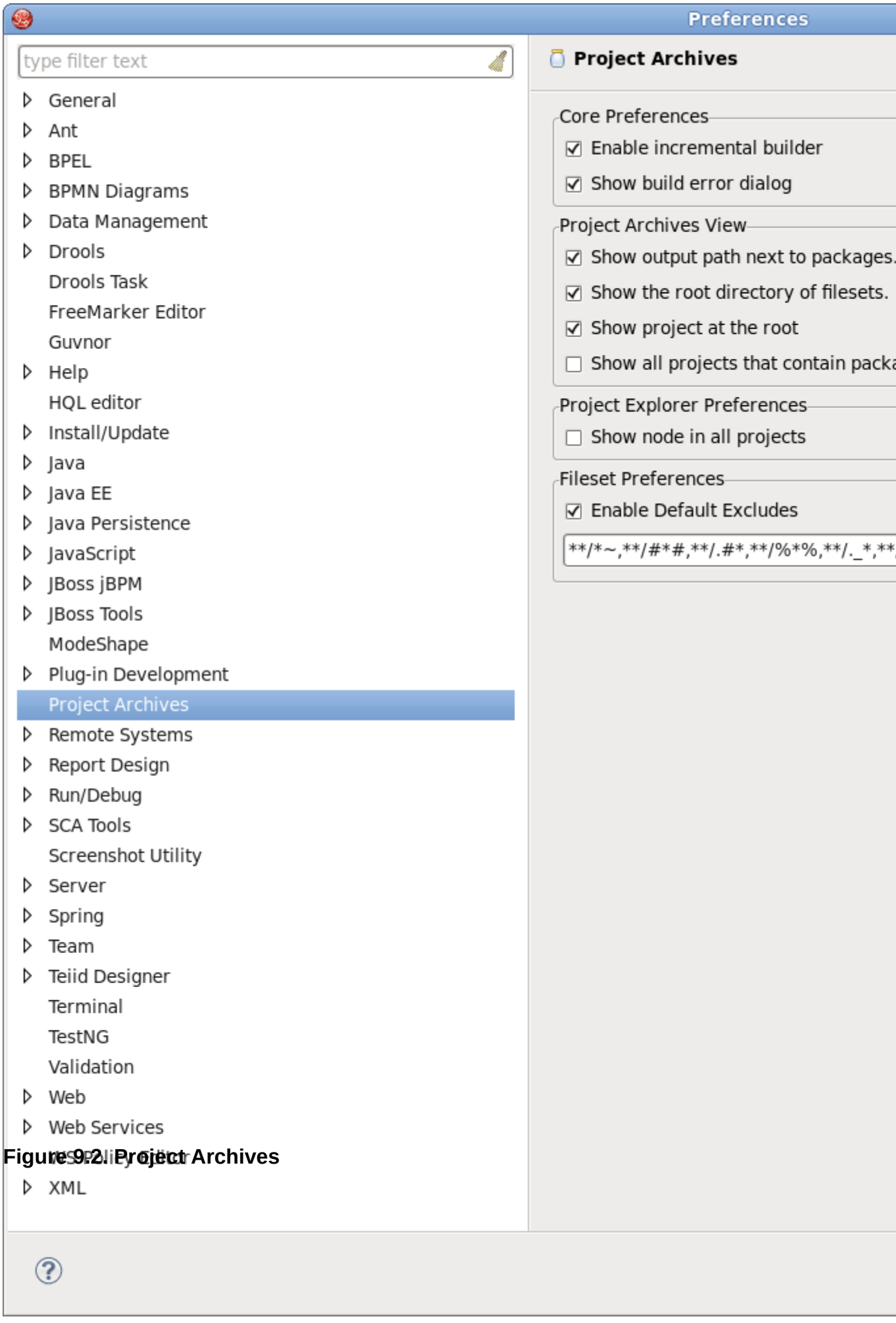

Figure 9.2. Project Archives

The next table lists all available preferences for Project Archives and their description.

Table 9.1. Project Archives Preferences

Option	Description	Default
Enable incremental builder	Uncheck this option if you don't want to enable incremental builder for your resources	On
Show build error dialog	If on, the Project Archives will show an error dialog in case of a build or incremental update fails.	On
Show output path next to packages	This option allows you to show or hide an output path next to packages .	On
Show the root directory of filesets	If on, the root directory is displayed next to filesets. Otherwise, it's hidden .	On
Show project at the root	This option allows you to choose whether to display a project name at the root of the packages or not. When checked, 'Show all projects that contain packages' is enabled .	On
Show all projects that contain packages	Selecting this setting enables the Projects Archiving view to show or hide all projects that contain packages. The option is available when the previous one is checked.	Off
Show node in all projects	Selecting this setting enables the Projects Archiving view to show node in all projects.	Off
Enable Default Excludes	You can set the list of files which will be excluded by default. Other files will be omitted.	On

9.2. Editors

To adjust settings common for all editors supplied with JBoss Developer Studio you should select **JBoss Tools** → **Web** → **Editors**.

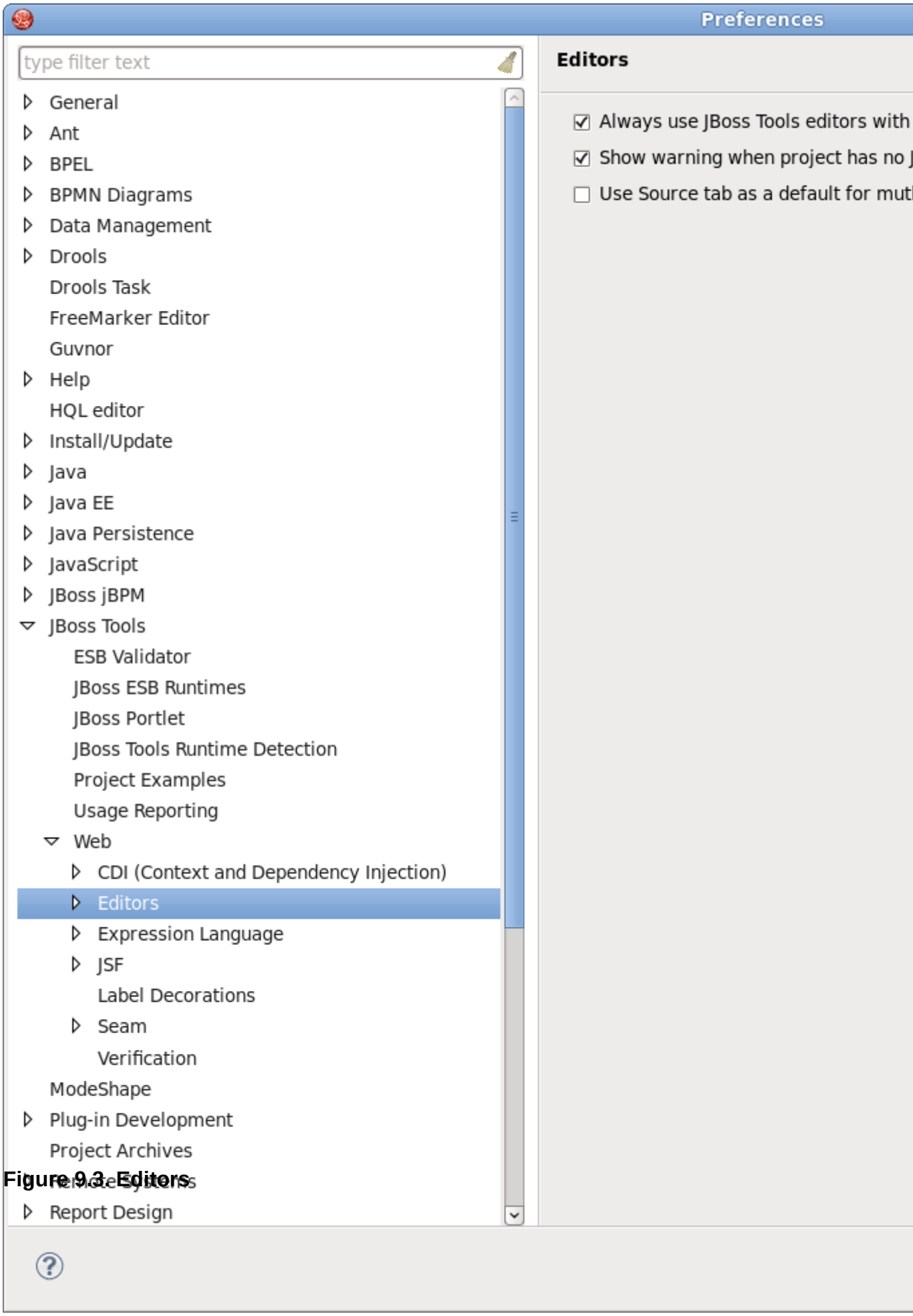

Figure 9.3. Editors

On the Editors page the following preferences are available:

Table 9.2. Editors Preferences

Option	Description	Default
Always use JBoss Tools editors with Open option		On
Show warning when project has no JBoss Tools capabilities	Check this option to be sure that any JBoss Tools editor is fully available for a particular type of file. If no, you'll be warned about this.	On
Use Source tab as a default for multi-tab editors	If on, an editor will open the files in the Source view by default	Off

9.3. Visual Page Editor

JBoss Tools → **Web** → **Editors** → **Visual Page Editor** screen allows you to control some aspects of the behavior of the Visual Page Editor (VPE) for JSF/HTML files.

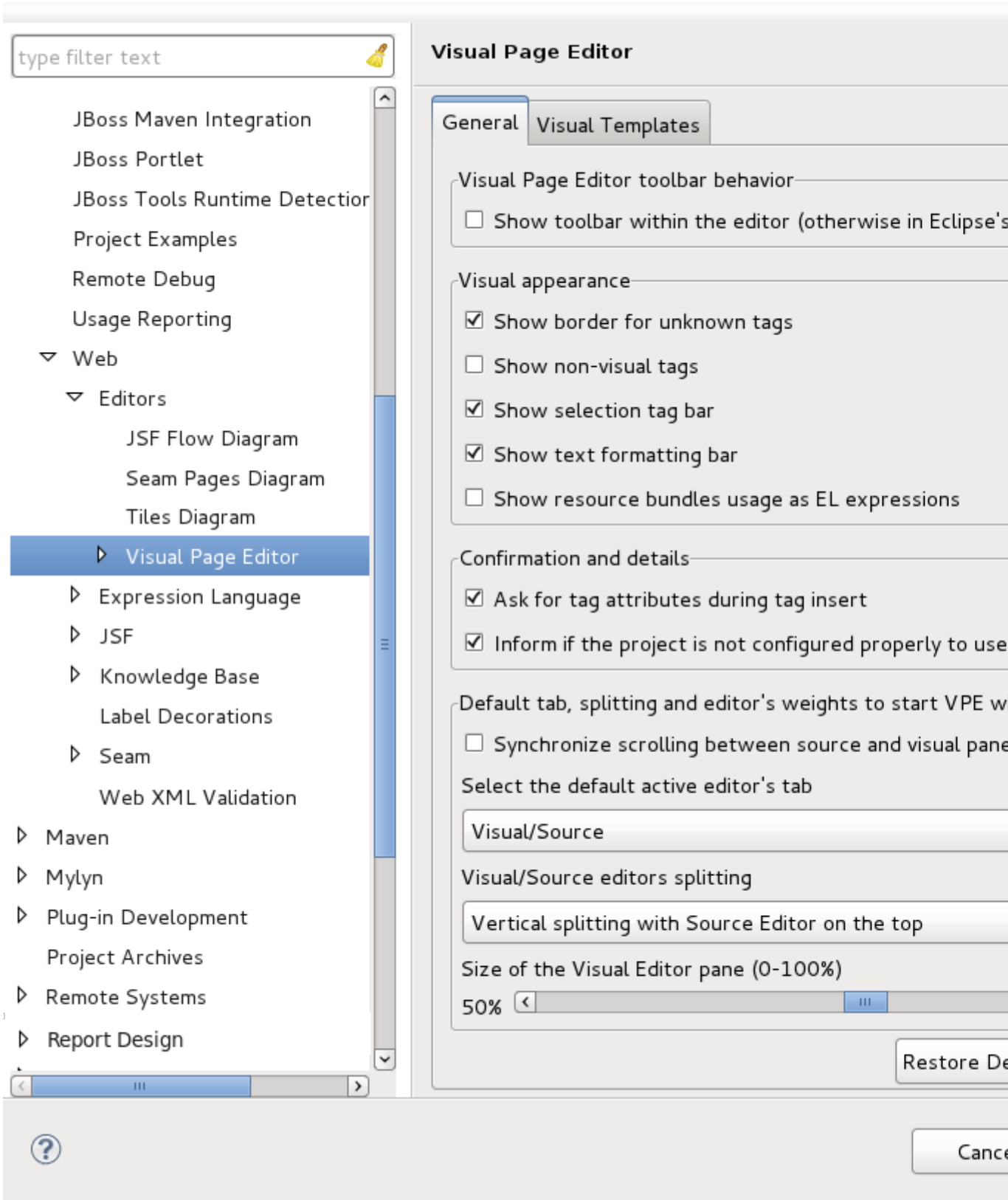

Figure 9.4. Visual Page Editor

The next table lists the possible settings that you can adjust on the General tab of the VPE Preferences page.

Table 9.3. VPE General Preferences

Option	Description	Default
Show toolbar within the editor (otherwise in Eclipse's toolbar)	The option allows to select where the Visual Page Editor toolbar appears. By default it appears as part of the Eclipse toolbar.	Off
Show border for unknown tags	The option allows to place the border around unknown tags or undo this	On
Show non-visual tags	Check this box, if you want the editor shows non-visual elements on the page you're editing	Off
Show selection tag bar	The option allows to show/hide the Selection Bar	On
Show text formatting bar	Check this box in order to show/hide the Text Formatting bar	On
Show resource bundles usage as EL expressions	If the option is checked, the editor will show EL expressions instead of the resource values	Off
Ask for tag attributes during tag insert	Having this option off, the dialog with possible attributes for inserting tag won't appear if all its attributes are optional	On
Inform if the project is not configured properly to use Visual Page Editor	If this option is deselected you will not be notified that a project is not configured for use with the Visual Page Editor. This may cause unexpected results.	On
Synchronize scrolling between source and visual panes	This option allows you to set the scrollbars of the source and visual panes of the editor to be synchronized by default. An option to activate and deactivate this option exists as part of the toolbar.	Off
Select the default active editor's tab	The option provides possibility to choose one of the following views - Visual/Source, Source or Preview, as default when opening the editor	Visual/Source
Visual/Source editors splitting	The option allows to choose one of the following Visual,Source layouts - Vertical Source on top, Vertical Visual on top,Horizontal Source to the left or Horizontal Visual to the left, as a default one when opening the Visual/ Source view	Vertical splitting with Source Editor on the top

Option	Description	Default
Size of the Visual Editor pane (0 – 100%)	With the help of this scroll bar you can adjust the percentage rating between the Source and Visual modes of the Visual/Source view	50%

On the Visual Templates tab you can add, edit or remove *Section 3.2.3, "Visual Templates for Unknown Tags"*.

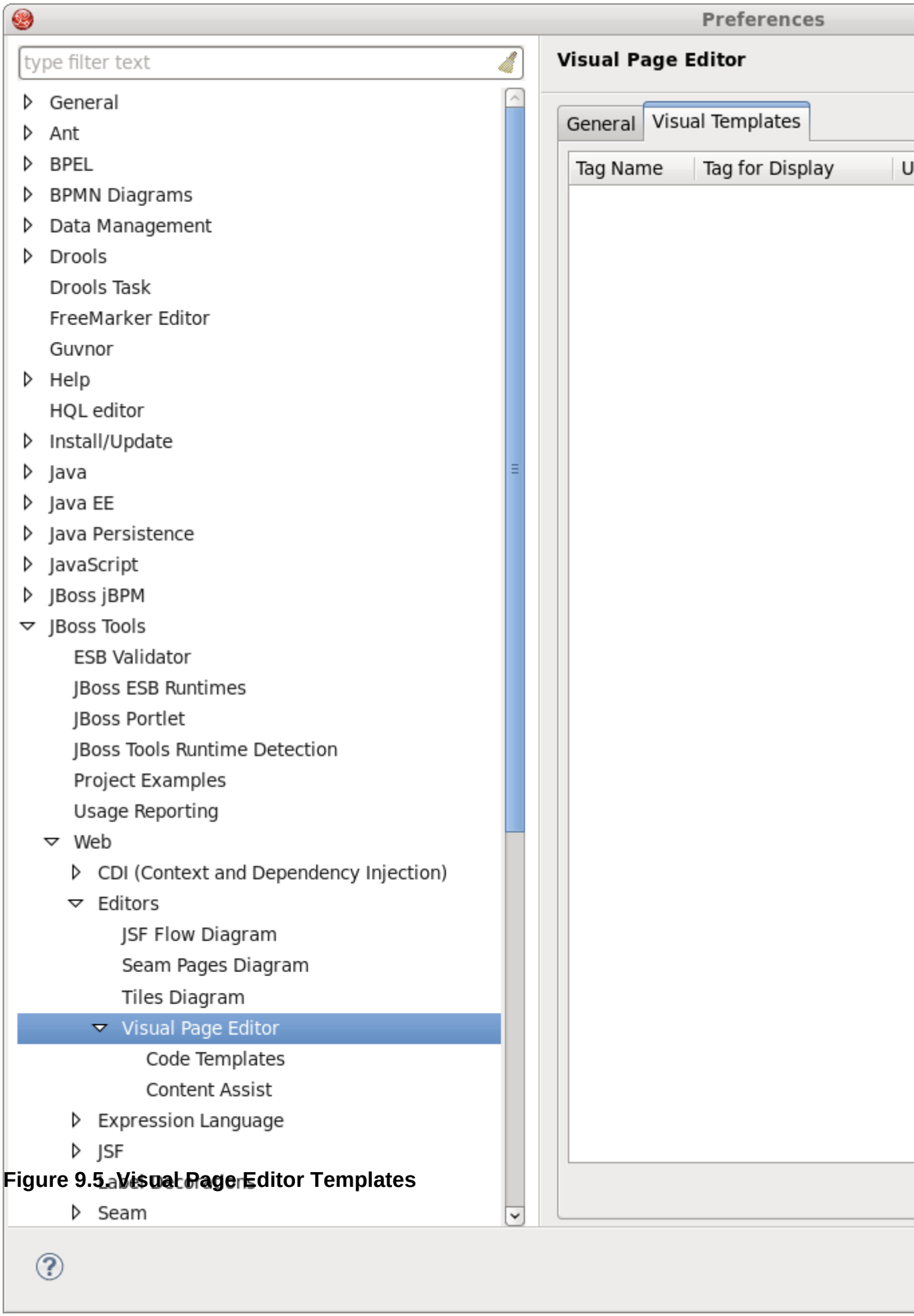

Figure 9.5. Visual Page Editor Templates

Select a template for editing from the available list and press *Edit* button. It will pick up the *Section 3.2.3, "Visual Templates for Unknown Tags" [70]* where you can adjust new settings.

9.4. Visual Page Editor Code Templates

On the **JBoss Tools** → **Web** → **Editors** → **Visual Page Editor** → **Code Templates** preferences page you can create new and edit existing XHTML templates. Such a template allows you to quickly insert an often used snippet of XHTML code.

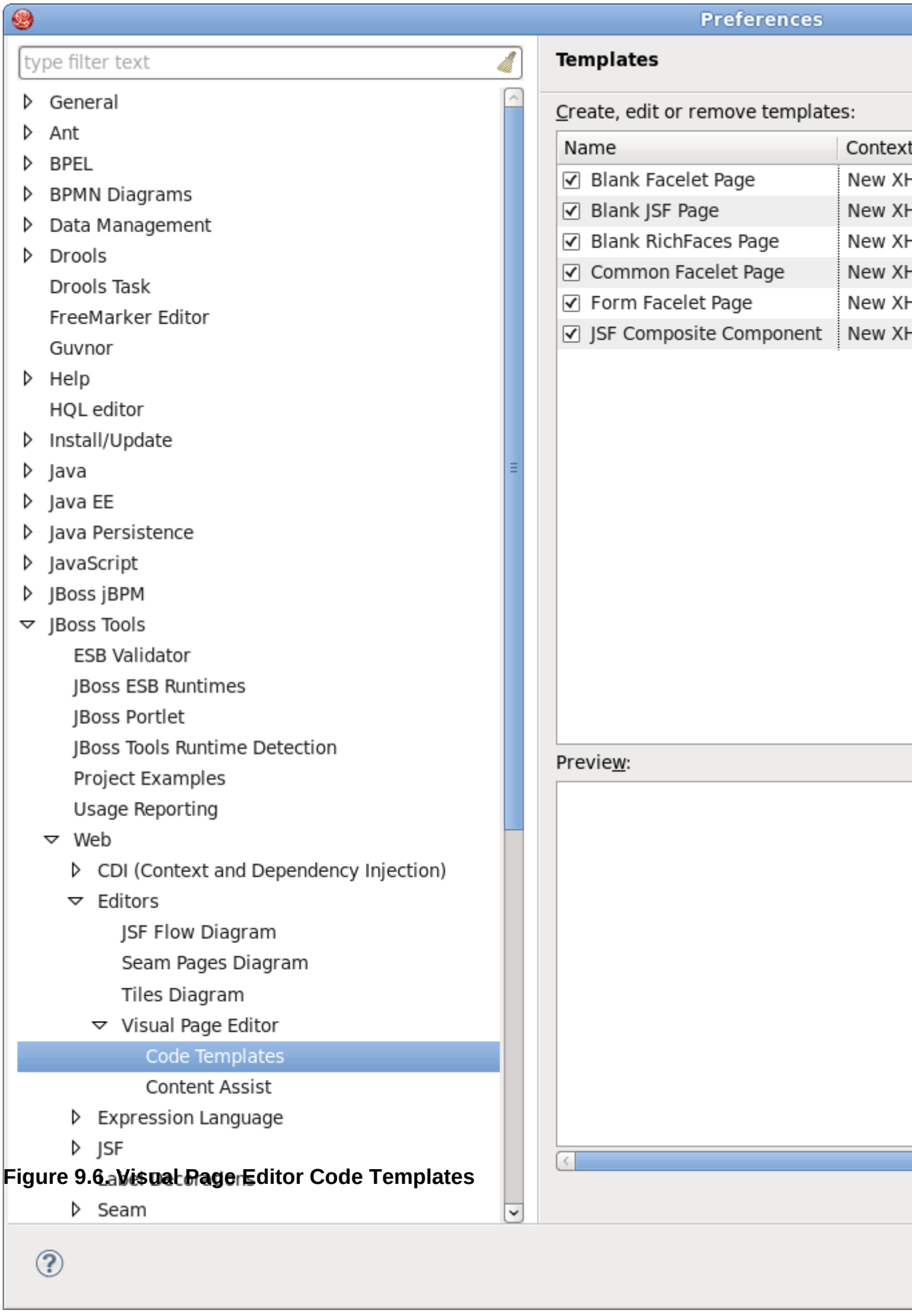

Figure 9.6. Visual Page Editor Code Templates

VPE provides four predefined templates:

- XHTML blank facelet page template
- Common facelet page template
- Form facelet page template
- New JSF composite component template

The following table lists the configuration options for the code templates.

Table 9.4. VPE Code Templates Options

Option	Description
New	Opens the dialog to create a new template.
Edit	Opens the dialog to edit the currently selected template.
Remove	Removes all selected templates.
Restore Removed	Restores the removed templates.
Revert to Default	Reverts the code templates list to default.
Import	Allows you to import templates from the file system.
Export	Allows you to export all selected templates to the file system.

9.5. EL Variables

To specify necessary EL variables globally, i. e. for all projects and resources in your workspace, you should go to **JBoss Tools** → **Web** → **Expression Language** → **Variables**.

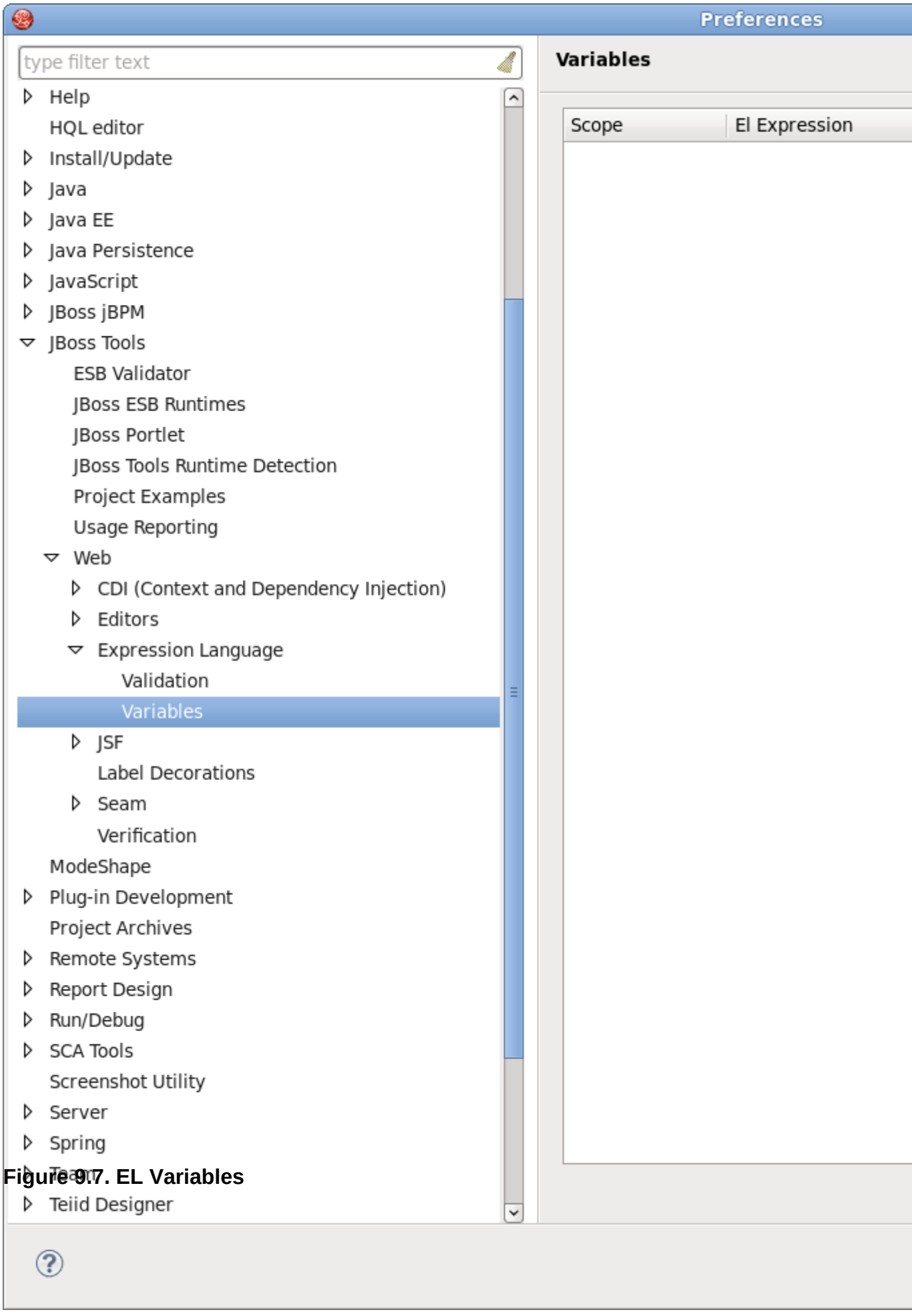

Figure 9.7. EL Variables

Click **Add...** to set value for a new EL variable. In the appeared wizard you should specify the global values and press **Finish**.

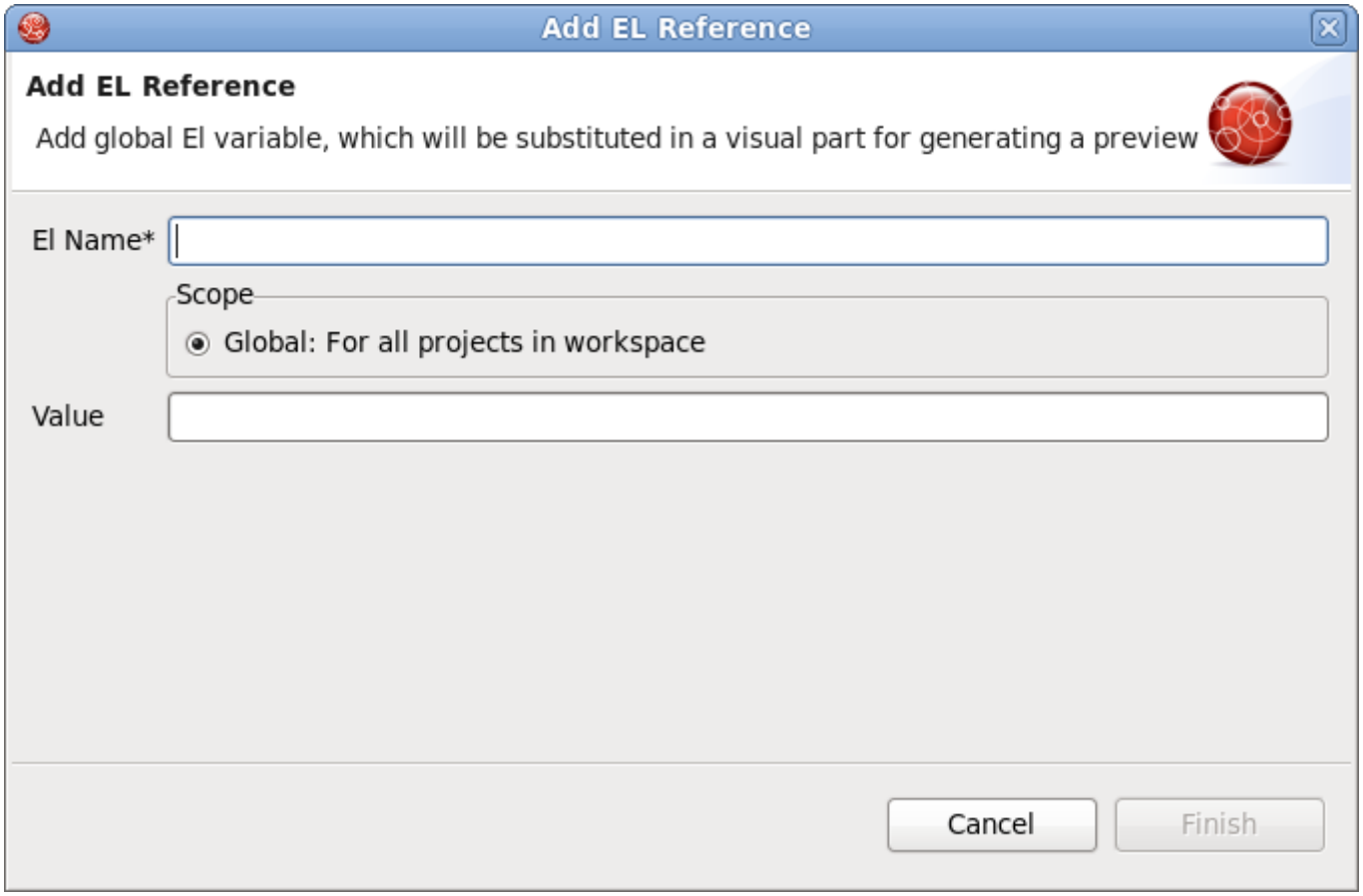

Figure 9.8. Adding a Global EL Variable

Tip:

If you specify an equal variable in *Section 3.2.5.4, "Page Design Options" [84]* and in Preference EL dialog, variable from preference dialog will have priority.

9.6. JSF

Select **JBoss Tools** → **Web** → **JSF** to get to the JSF Project specific preferences.

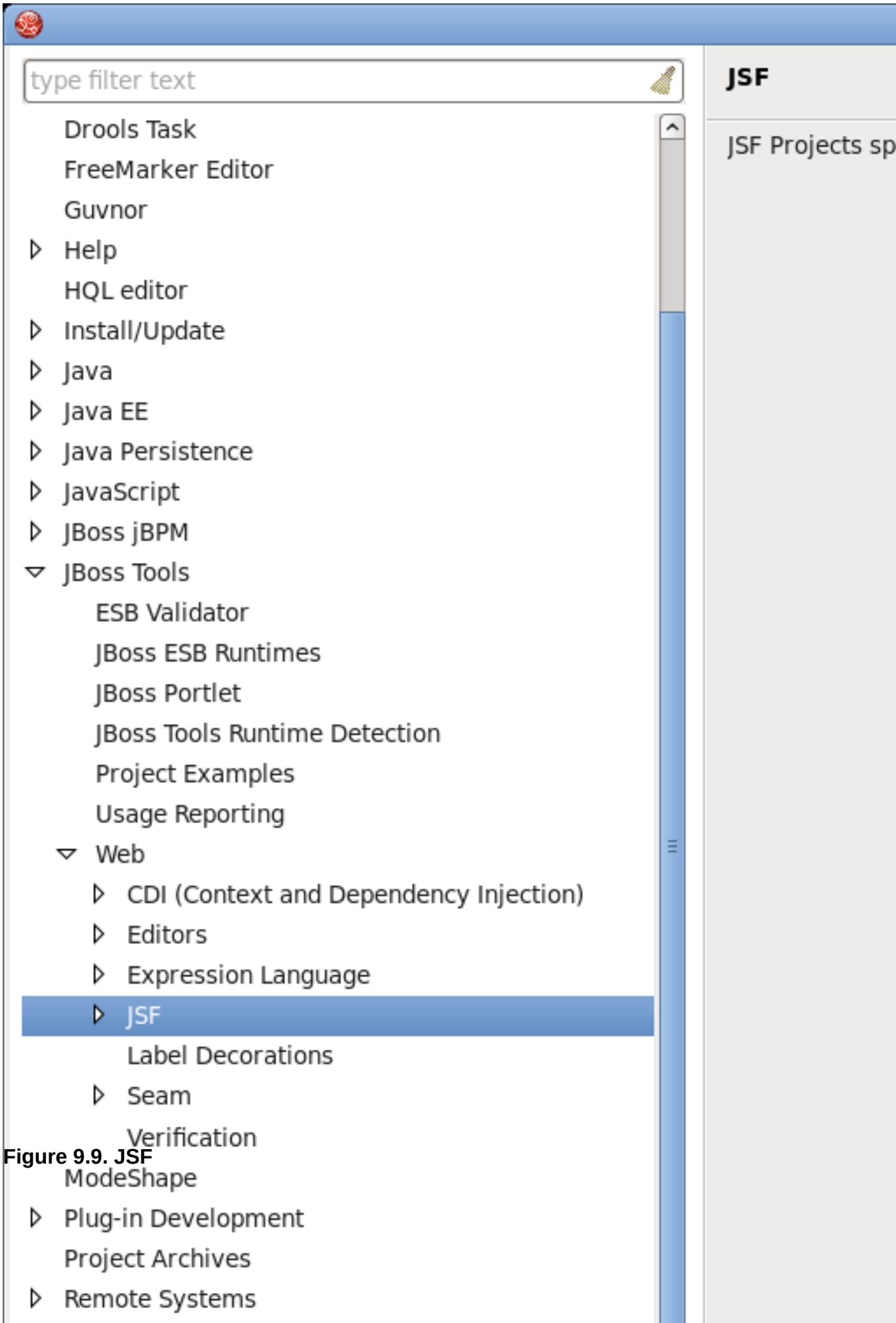

Figure 9.9. JSF

9.7. JSF Project

Select **JBoss Tools** → **Web** → **JSF** → **Project** to see JSF Project preferences page.

On the **New Project** tab you can set default values for New JSF Project wizard:

- **Version** for setting the default JSF Environment
- **Project Template** so as New JSF Project wizard shows this template as default for the chosen JSF Environment
- **Project Root** for specifying default location for a new JSF project

  If you check **Use Default Path** here, this box will be also checked in the **New JSF Project** wizard.
- **Servlet Version** for setting the default Servlet version of a new JSF project

  Here it's also possible to define whether to register Web Context in `server.xml` while organizing a new project or not. Check the proper box in order to do that.

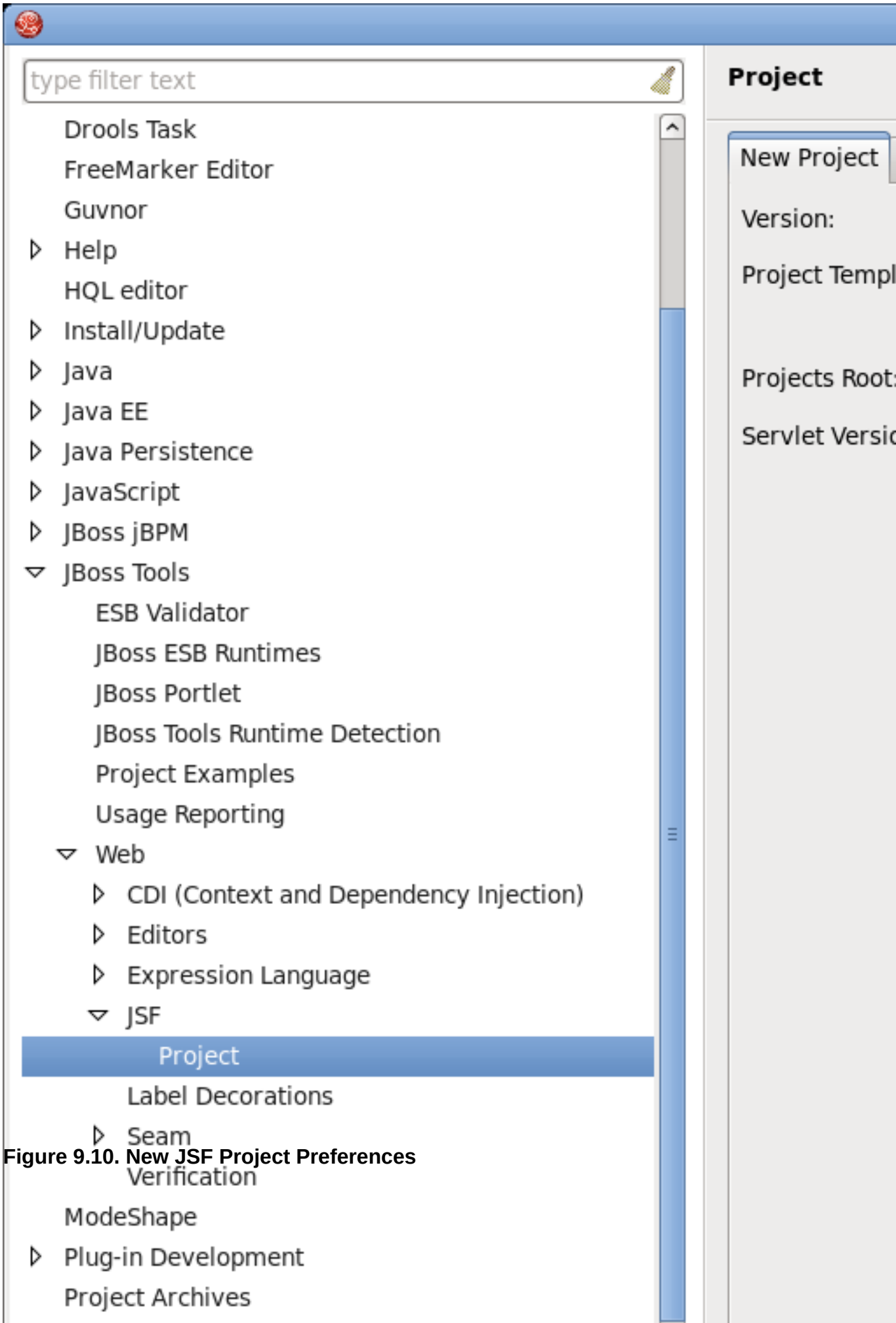

Figure 9.10. New JSF Project Preferences

On the **Import Project** tab in the JSF Project screen you can determine the default Servlet version for the Import JSF Project wizard and also whether to register Web Context in `server.xml` or not.

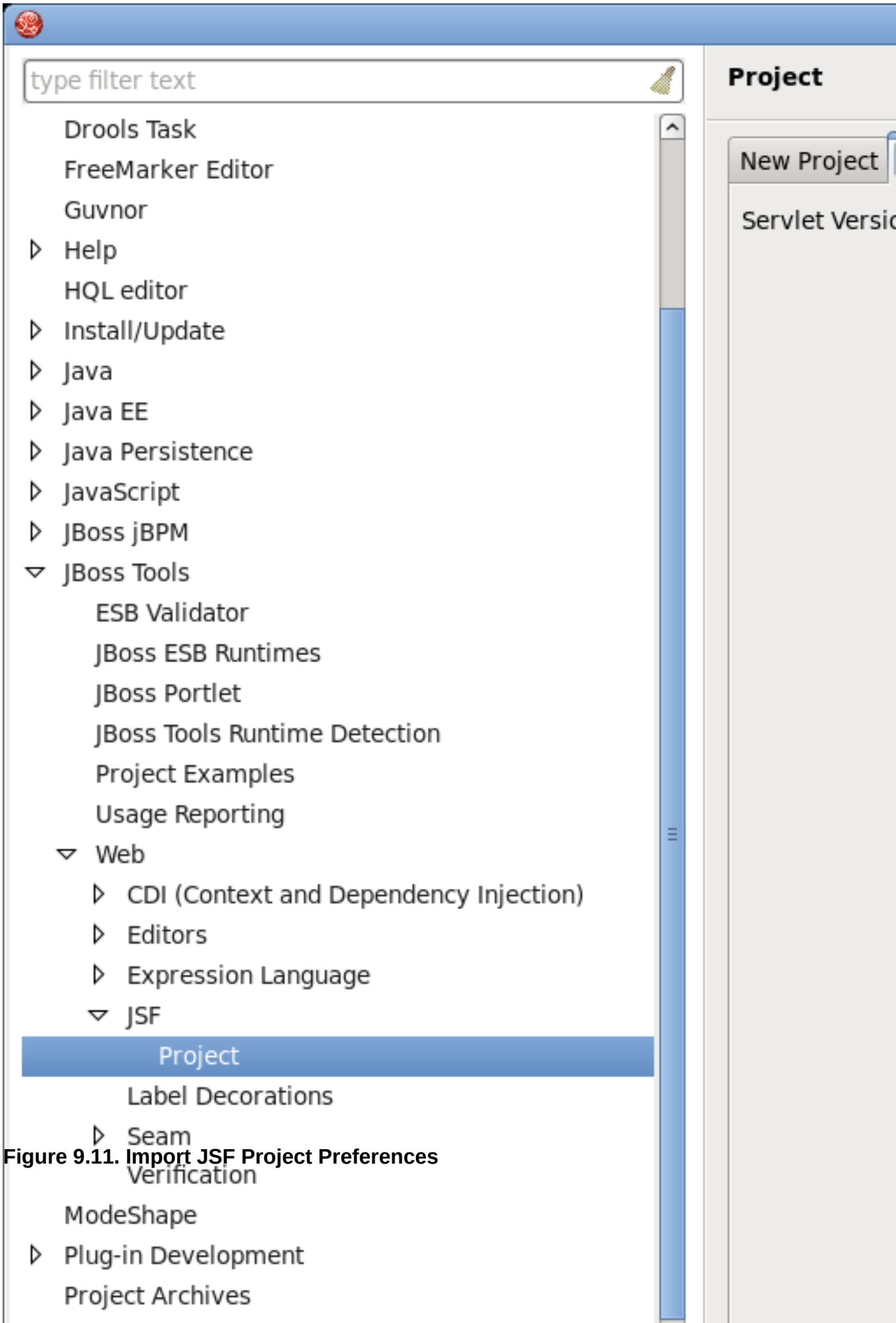

Figure 9.11. Import JSF Project Preferences

9.8. JSF Flow Diagram

Selecting **JBoss Tools** → **Web** → **Editors** → **Editors** → **JSF Flow Diagram** allows you to specify some aspects of the Diagram mode of the JSF configuration file editor.

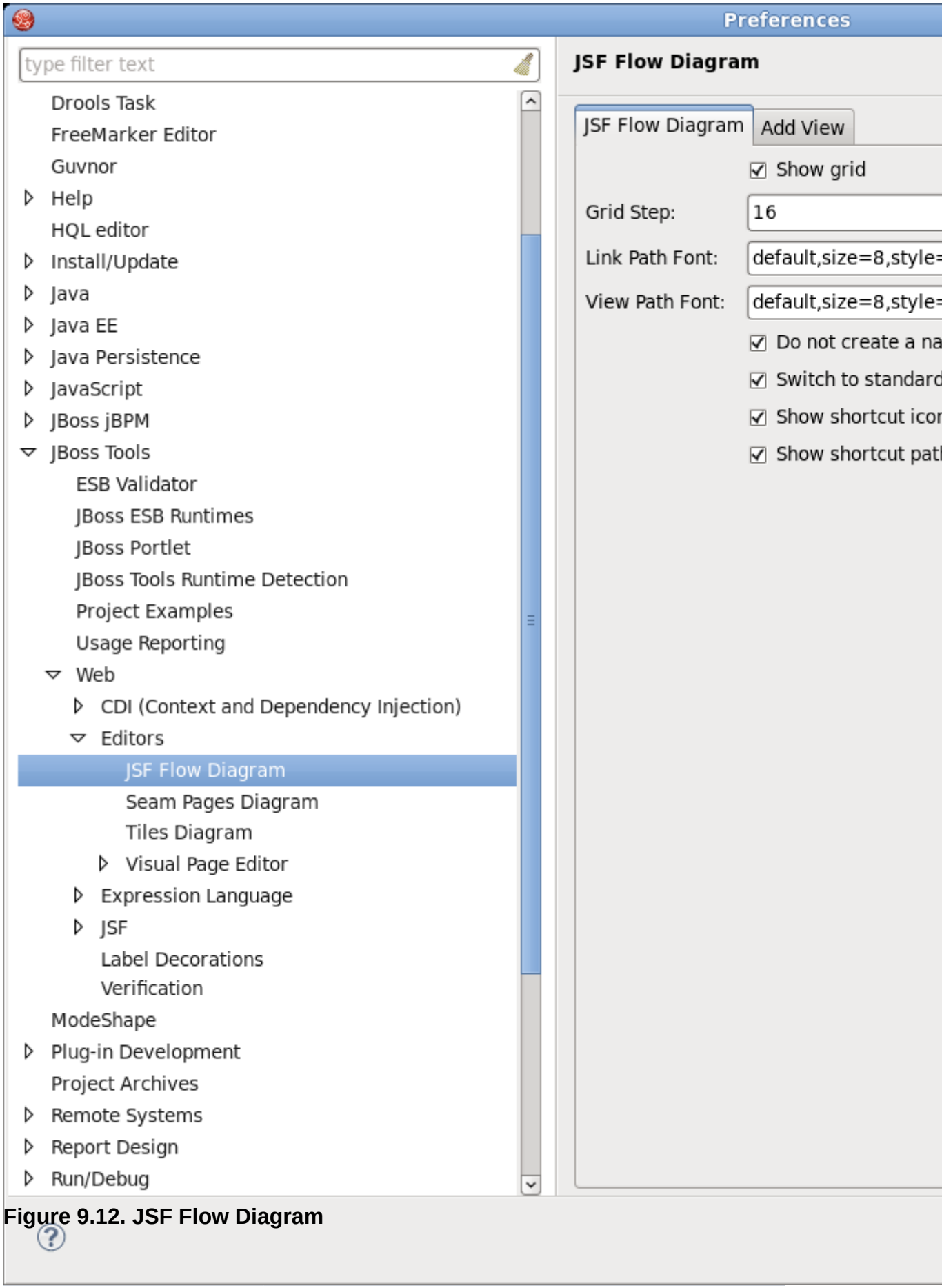

Figure 9.12. JSF Flow Diagram

The first two items control the background grid for the diagram. The next two items allow you to control the appearance of the labels for views (pages) and the transitions between views. For these two items clicking the **Change...**button allows you to assign a font with a dialog box.

The first check box determines whether a view in the diagram that doesn't have a transition connecting it to another view yet should be written to the source code as a partial navigation rule. The next check box determines whether the diagram cursor reverts immediately to the standard selection mode after it's used in the transition-drawing mode to draw a transition. Finally the last two check boxes concern shortcuts. A shortcut is a transition that is there but isn't actually displayed in the diagram as going all the way to the target view it's connected to, in order to make the diagram clearer. With the check boxes you can decide whether to display a small shortcut icon as part of the shortcut and also whether to display the target view as a label or not.

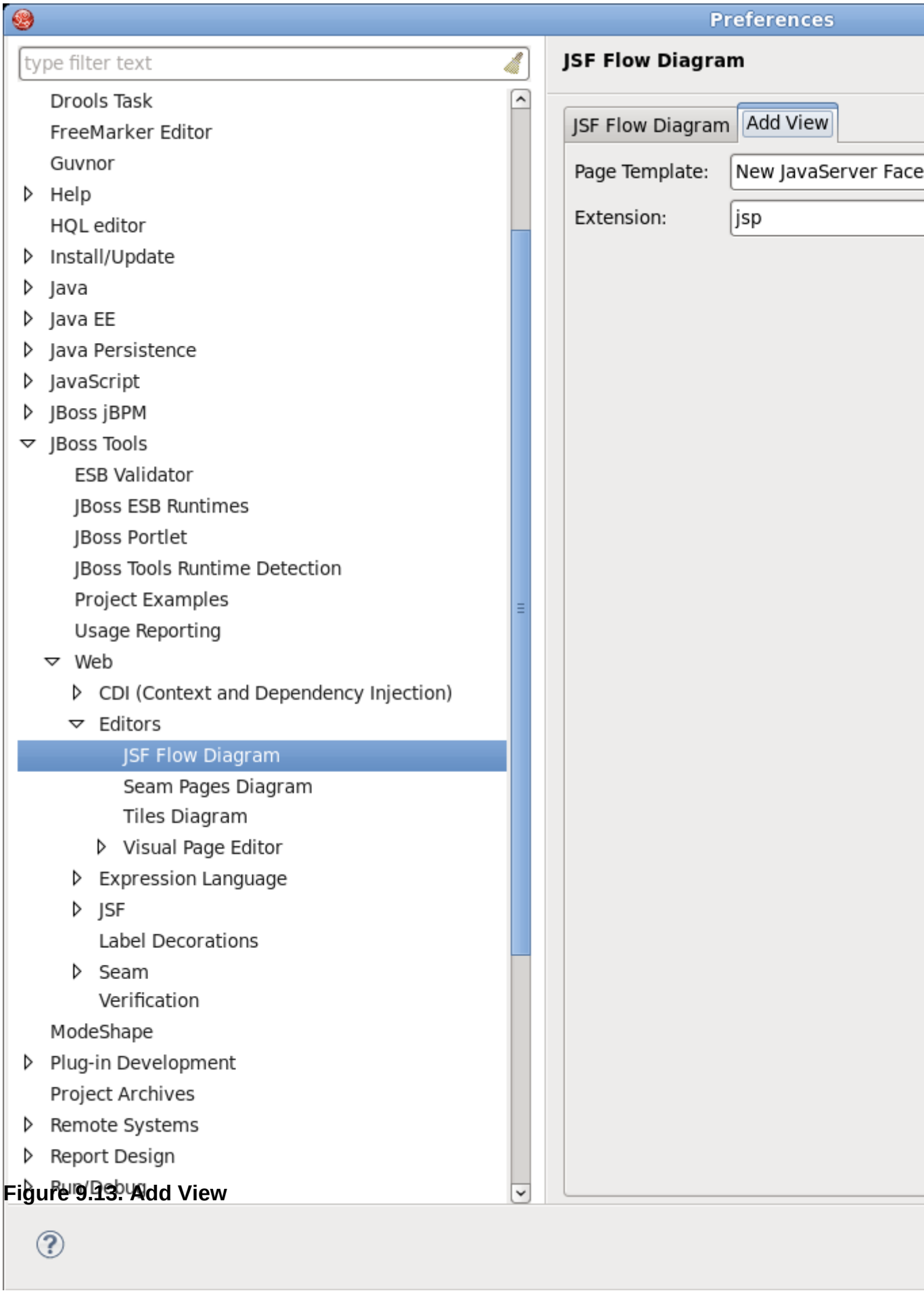

Figure 9.13. Add View

Selecting the Add View tab in the JSF Flow Diagram screen allows you to determine the default template and file extension for views (pages) you add directly into the diagram using a context menu or the view-adding mode of the diagram cursor.

9.9. Label Decorations

The Label Decorations page is opened from **JBoss Tools** → **Web** → **Label Decorations**.

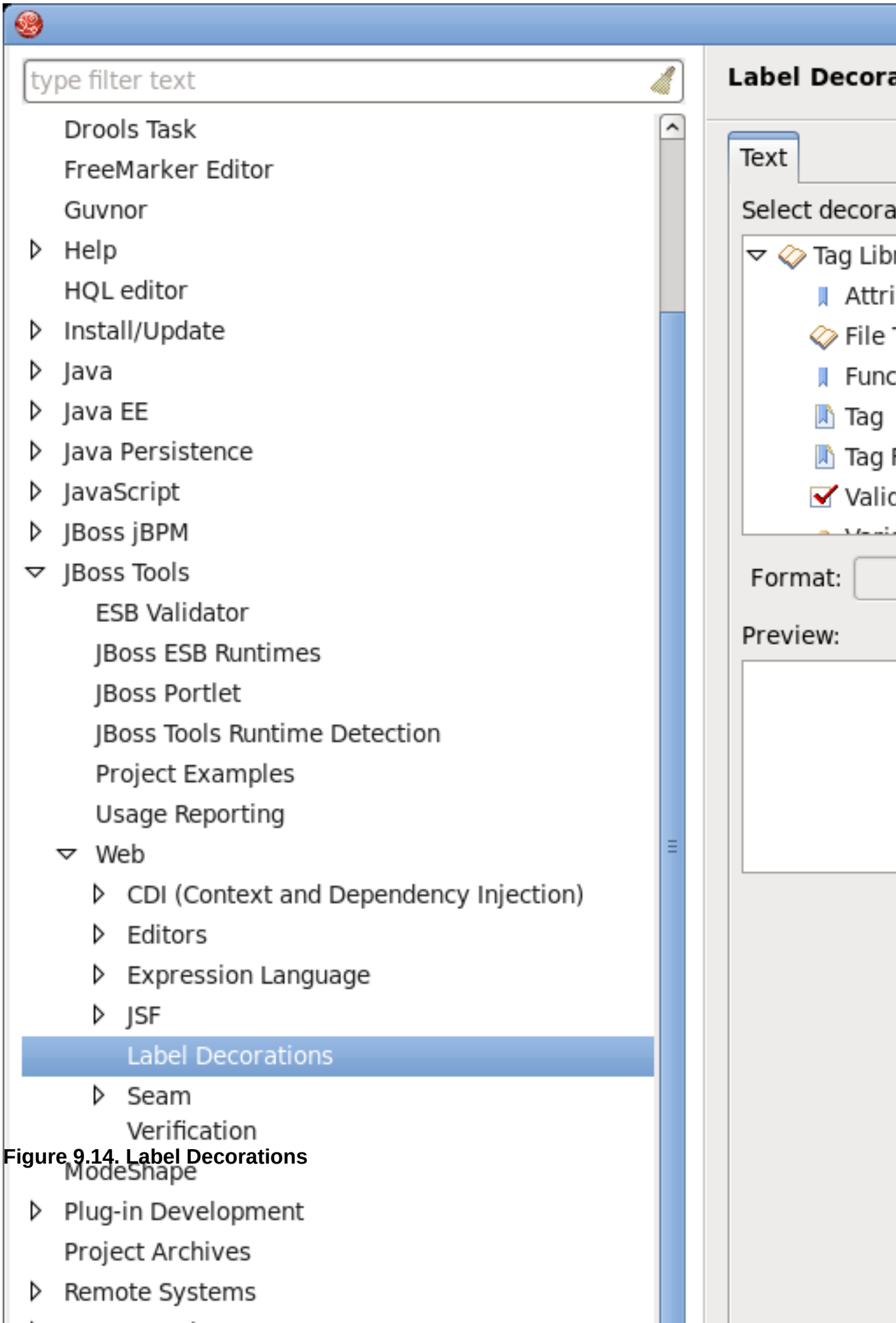

Figure 9.14. Label Decorations

On this page you can determine the format for a text output near to the decoration label for different Web resources. To change the value for selected element, click **Add Variable...** button next to **Format** field. Appeared wizard will prompt you to select one from the available list.

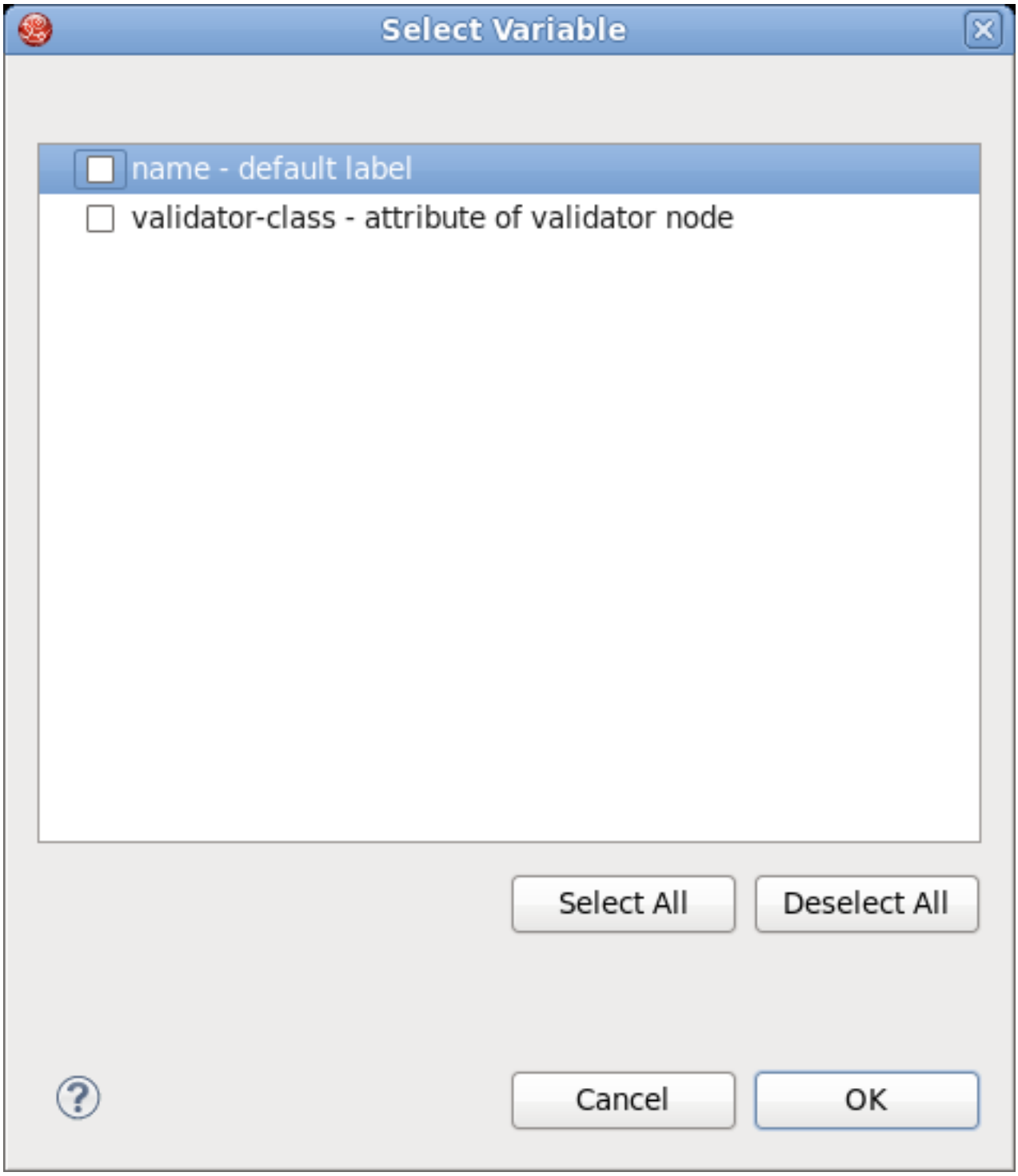

Figure 9.15. Label Decoration for Validator

9.10. Seam

The following preferences can be changed on the **JBoss Tools** → **Web** → **Seam** page.

On **Seam** screen you can add and remove Seam runtimes.

Here is what Seam preference page looks like:

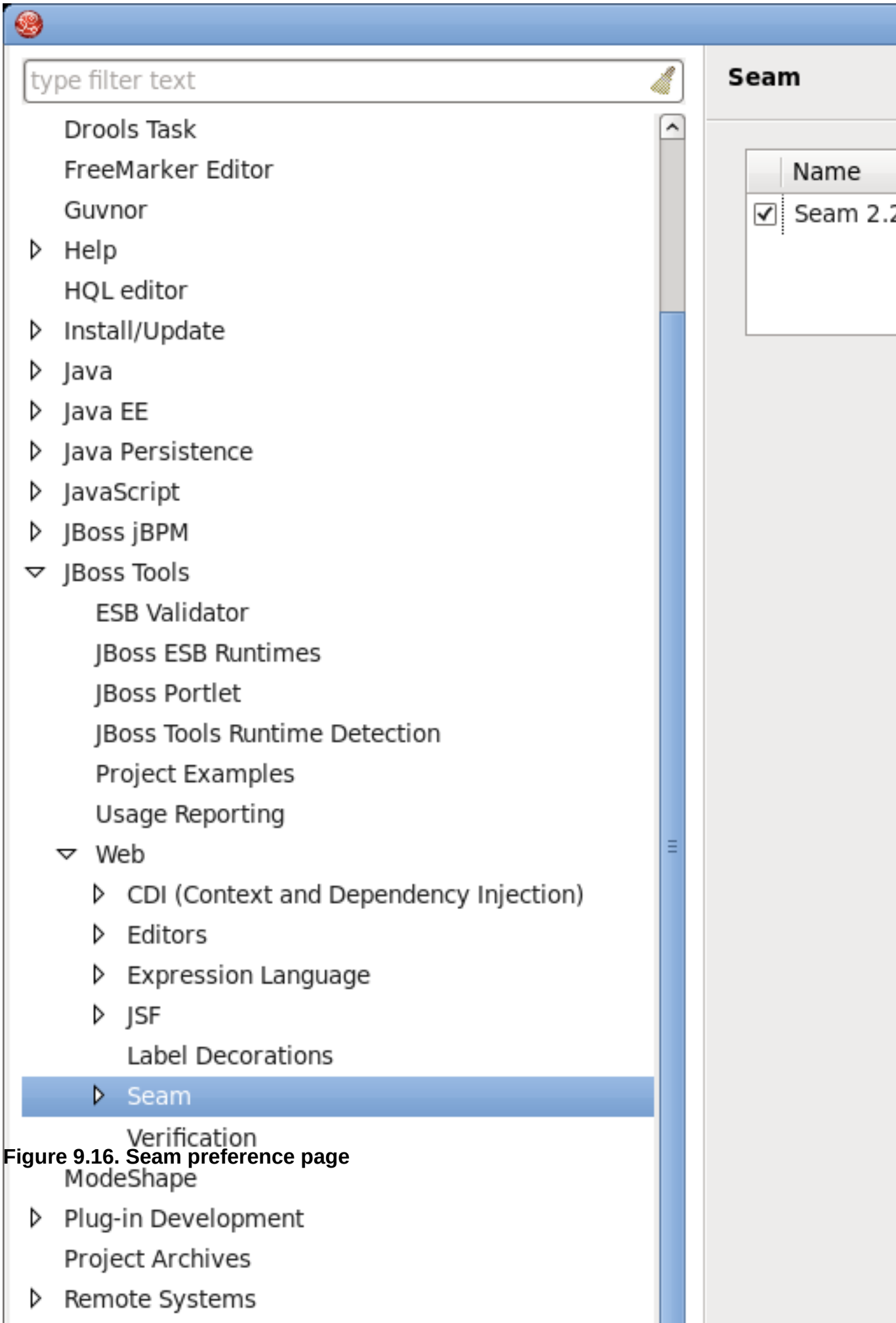

Figure 9.16. Seam preference page

9.11. Seam Validator

The following preferences can be changed on the **JBoss Tools** → **Seam** → **Validator** page.

In **Validator** panel you configure seam problems that will be processed by validator.

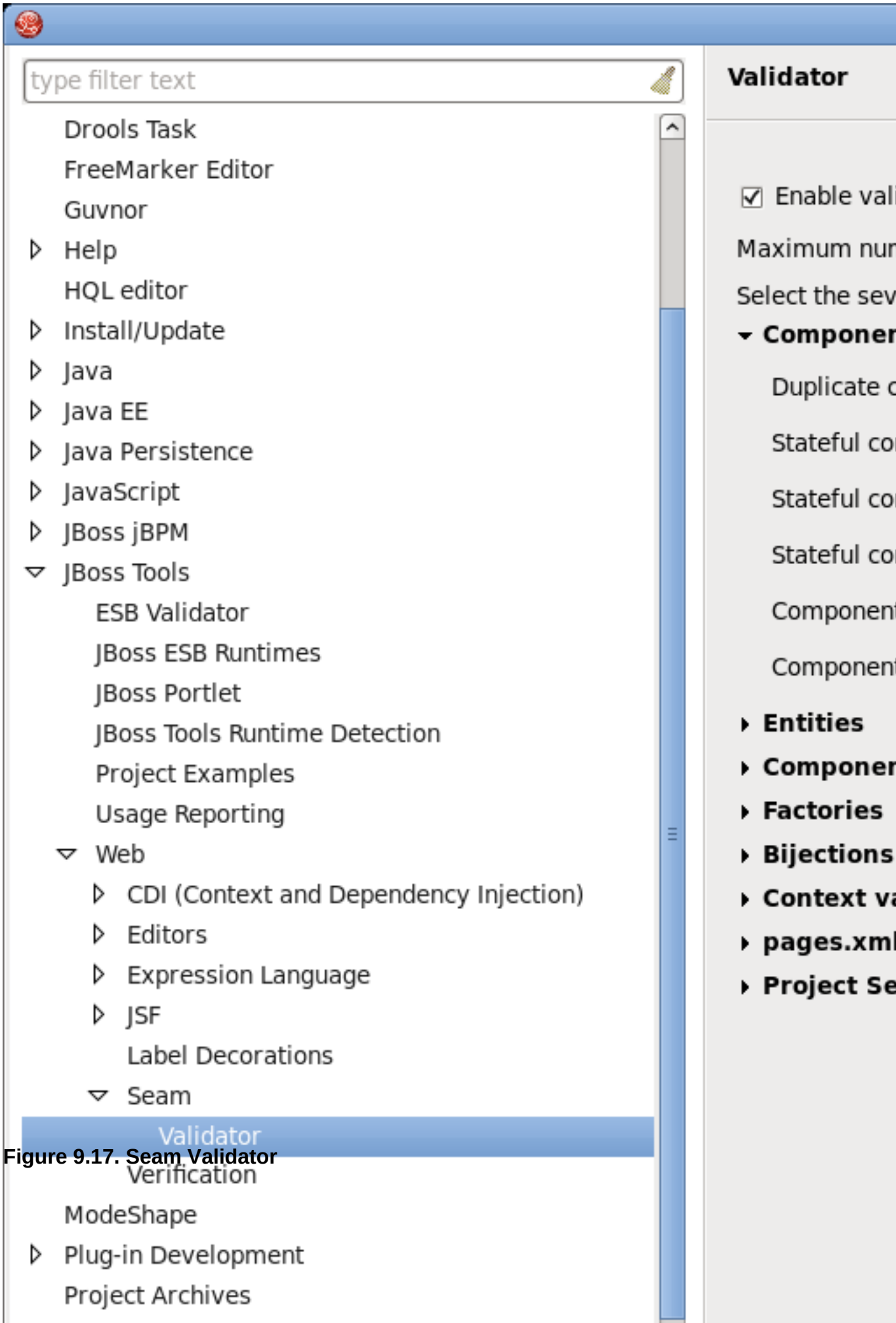

Figure 9.17. Seam Validator

9.12. Seam Pages Diagram

In order to customize the layout of the Diagram used for editing and composing `page.xml` file in Graphical mode of Seam Pages Editor you can go to **Window** → **Preferences** → **JBoss Tools** → **Web** → **Editors** → **Seam Pages Diagram**.

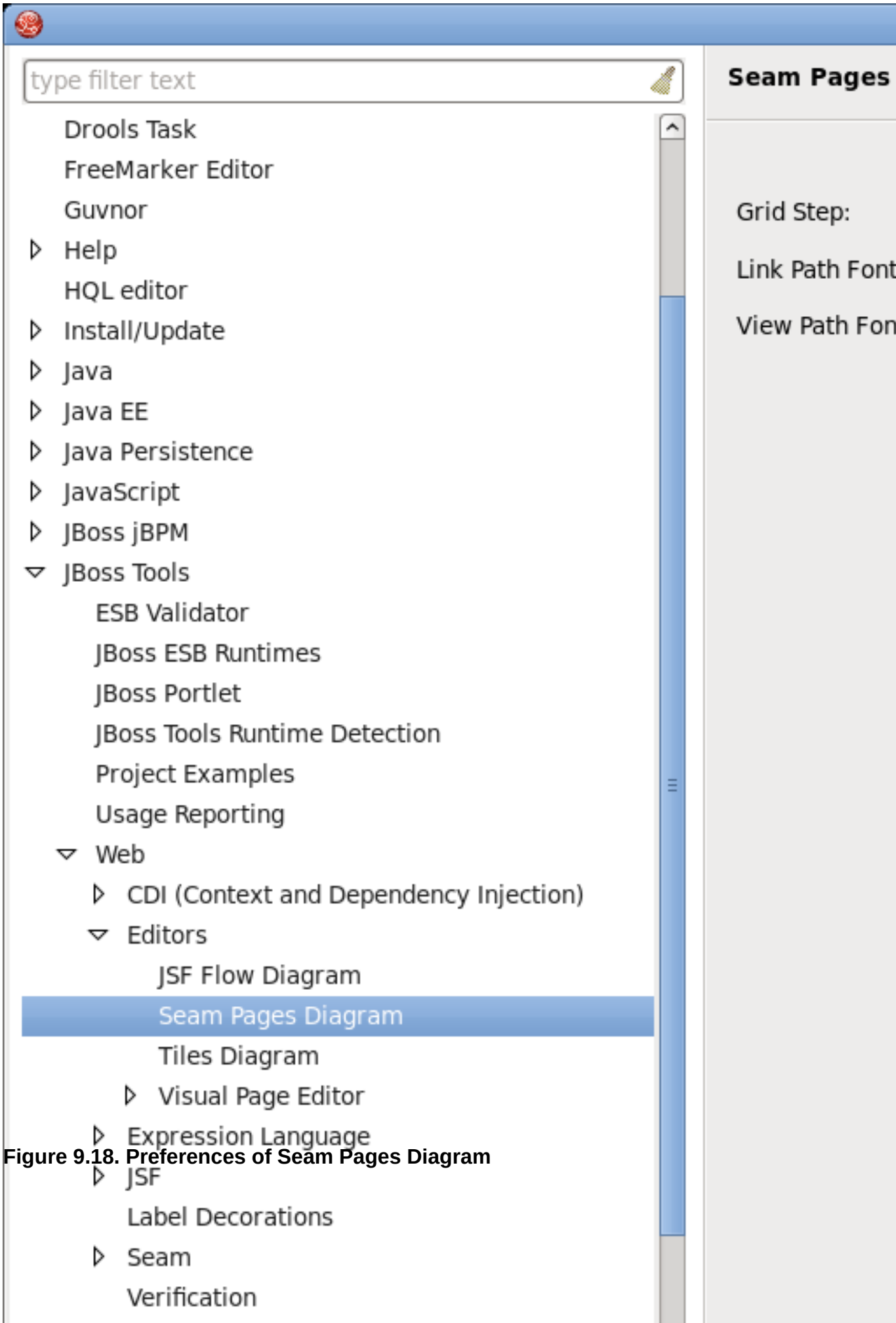

Figure 9.18. Preferences of Seam Pages Diagram

9.13. Tiles Diagram

JBoss Tools → **Web** → **Editors** → **Title Diagram** screen allows you control some settings for the placement of Tiles definitions in the Diagram mode of the JBoss Tools Tiles editor.

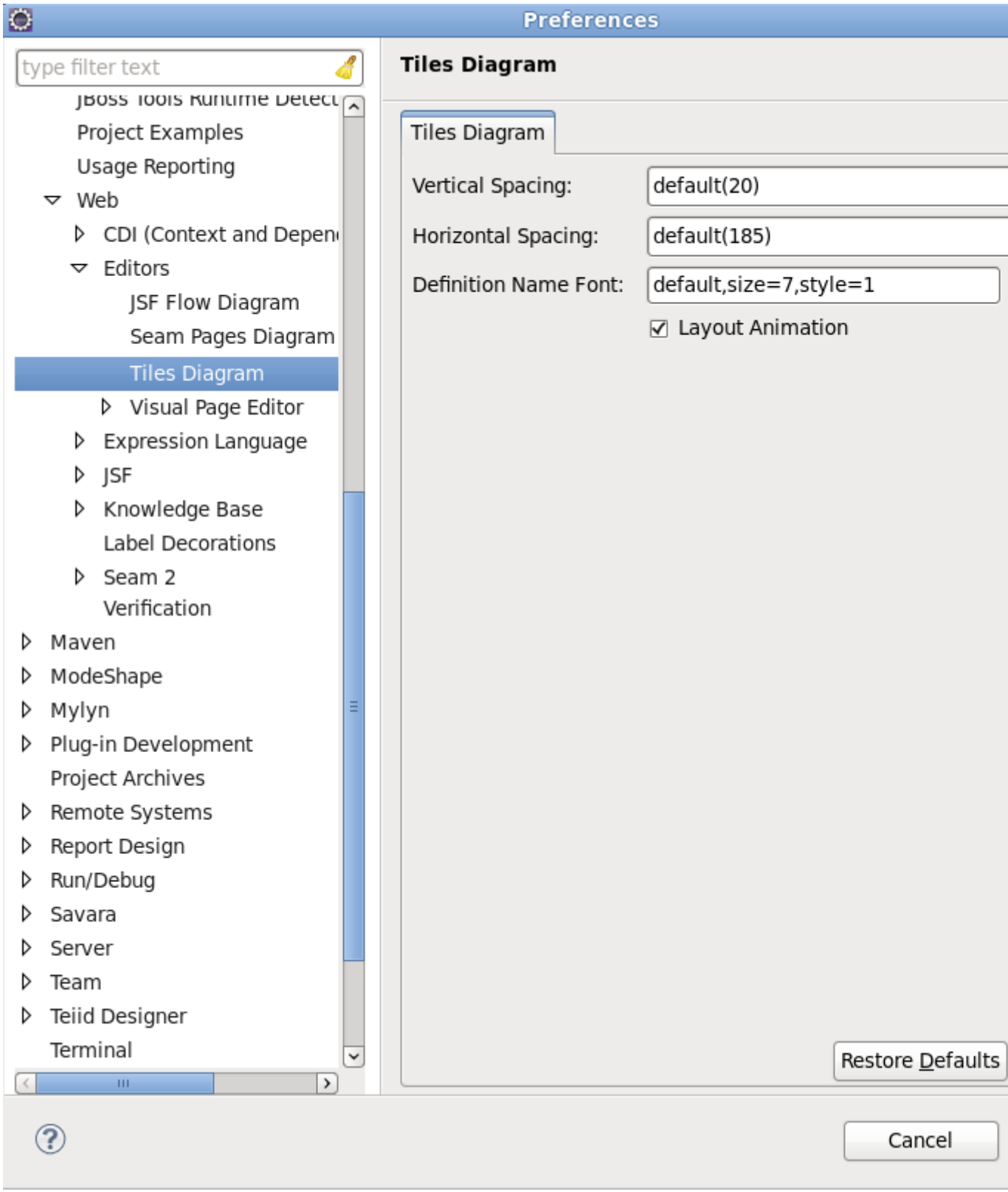

Figure 9.19. Tiles Diagram

9.14. Server Preferences

Preferences for JBoss Server and other servers can be changed on the **Server** page.

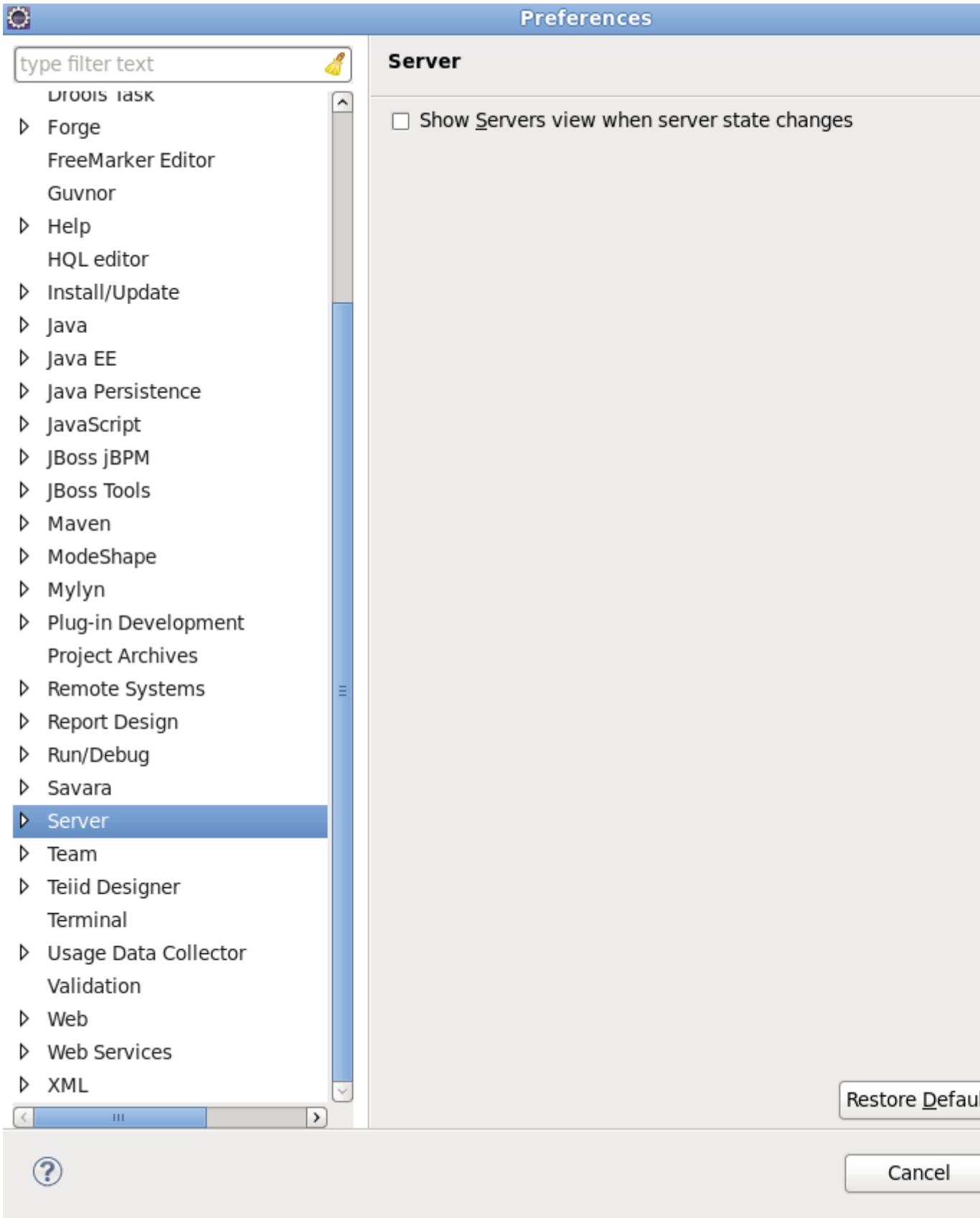

Figure 9.20. Server Preferences

On the **Server** → **Runtime Environments** page you can add new or modify already defined Server Runtime.

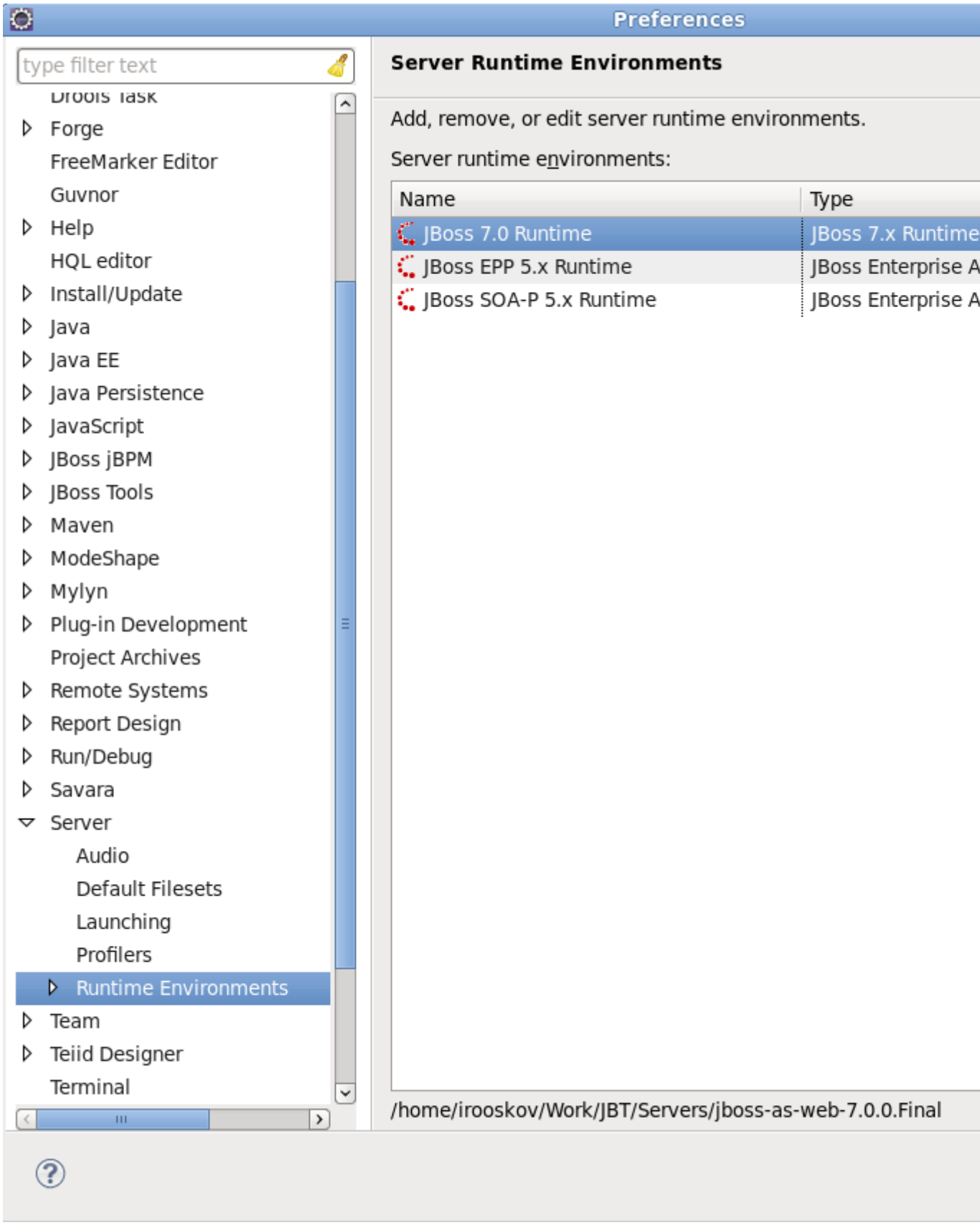

Figure 9.21. Runtime Environments

Server Launching preferences can be configured on the **Server** → **Launching** page.

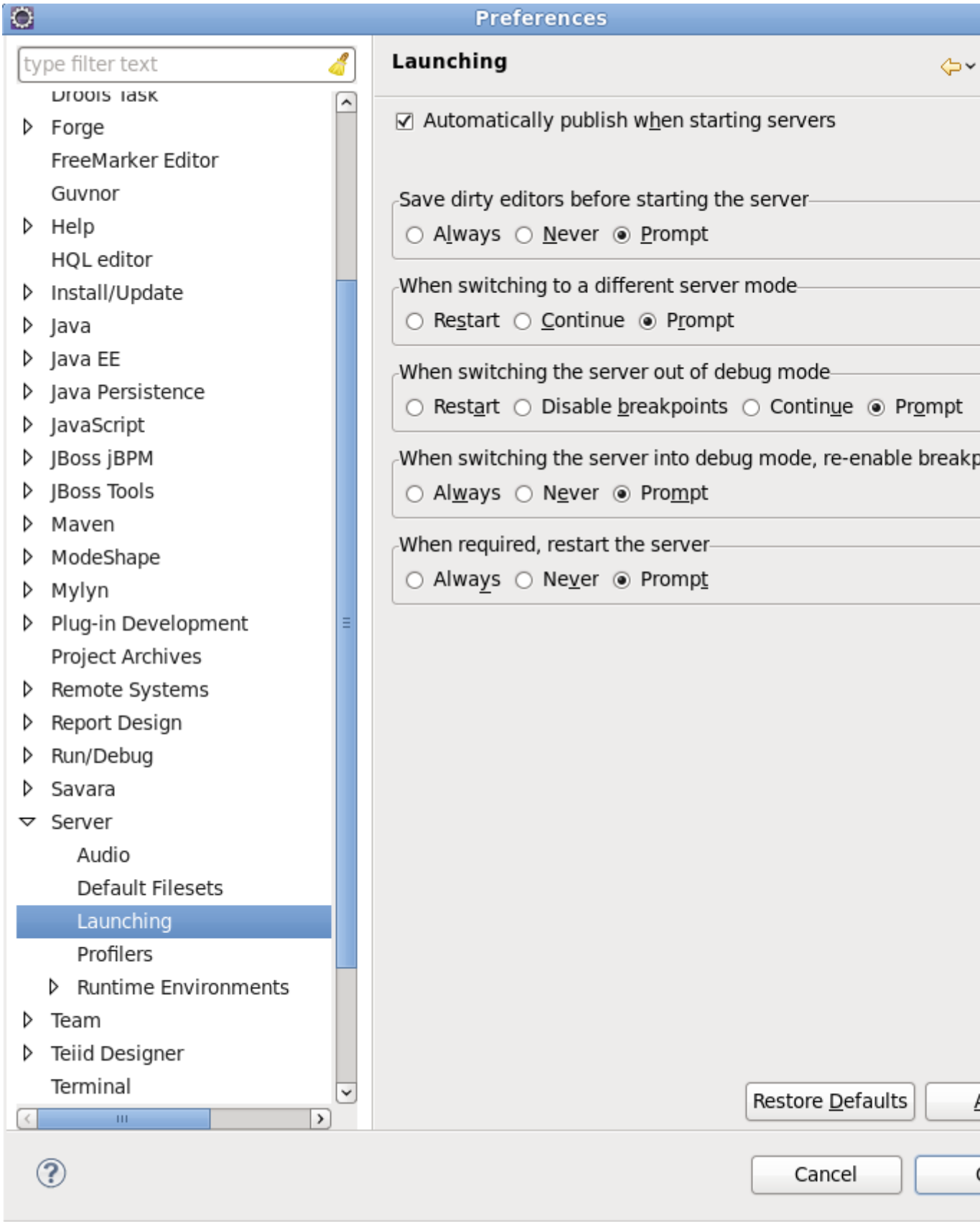

Figure 9.22. Server Launching Preferences

Going to **Server** → **Audio** you can enable or disable the sound notification for different Server states and actions and set the sound volume as well.

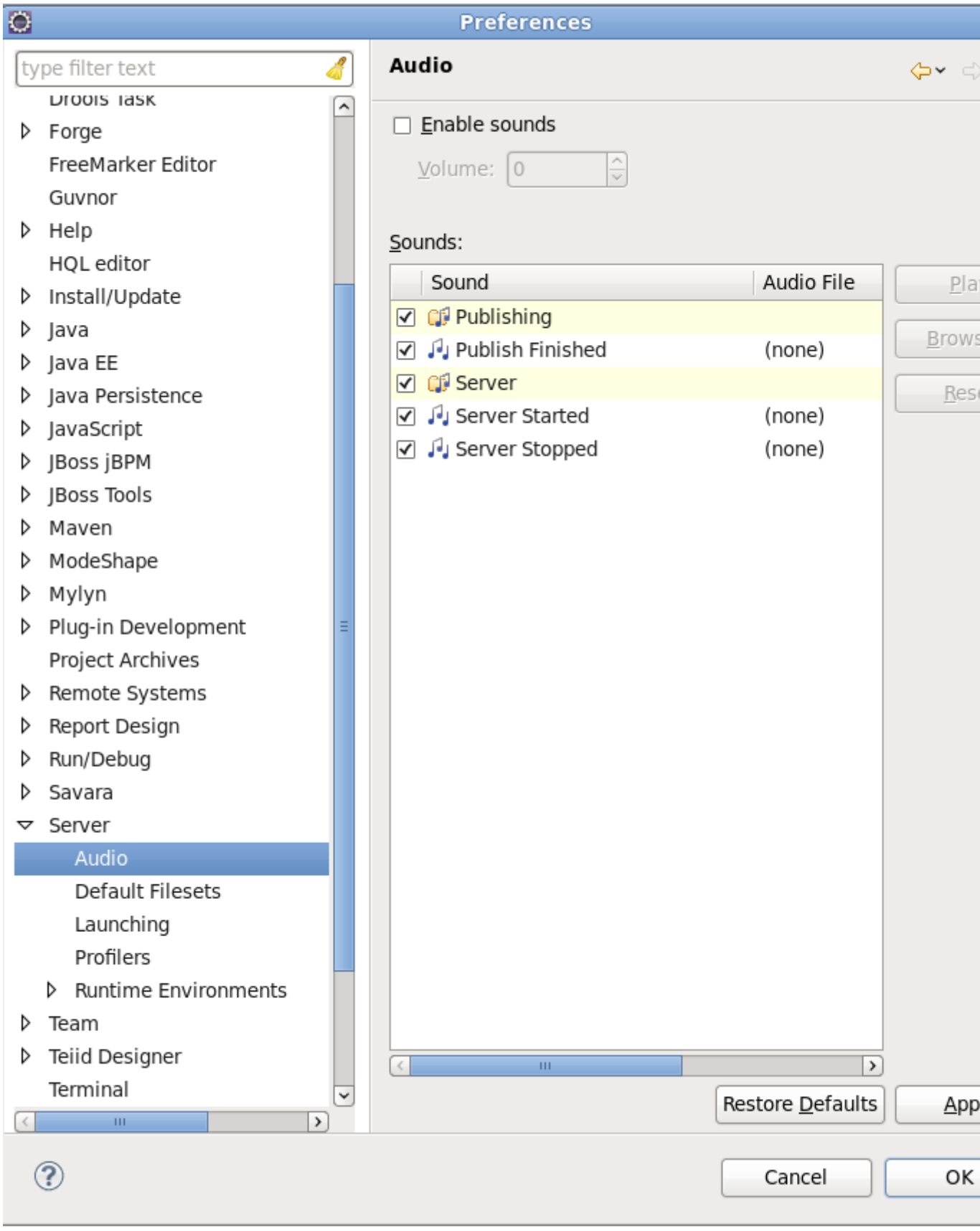

Figure 9.23. Sound Notification Adjustment

9.15. XDoclet

The preferences for XDoclet can be changed if you click **Java EE** → **XDoclet** on the left navigation bar.

On the **XDoclet** screen it is possible to enable or disable XDoclet builder by checking proper box, specify XDoclet home and determine XDoclet module version as well.

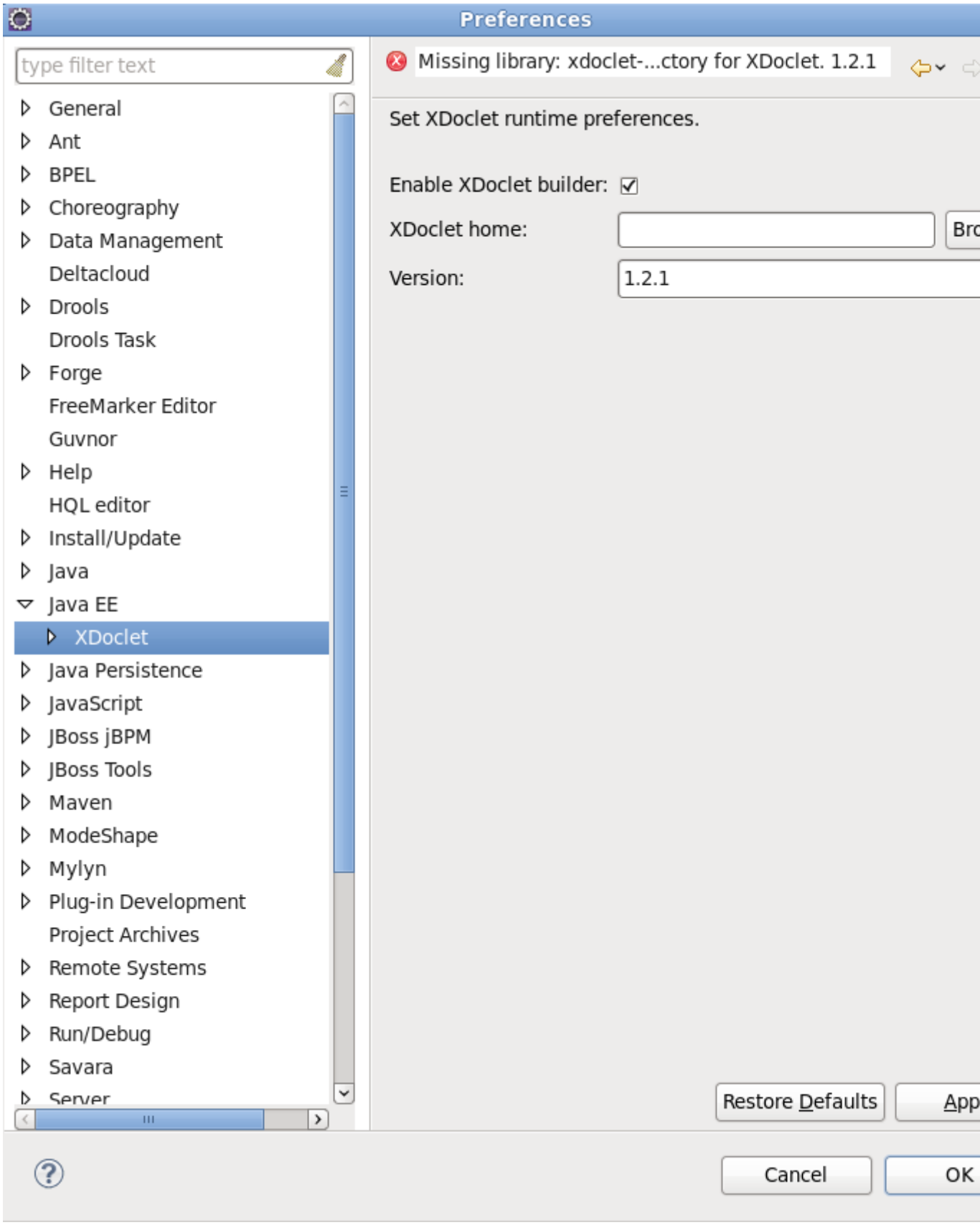

Figure 9.24. XDoclet Runtime Preferences Page

Switch to **Java EE** → **XDoclet** → **ejbdoclet** page in order to adjust settings for EJB-specific sub-tasks.

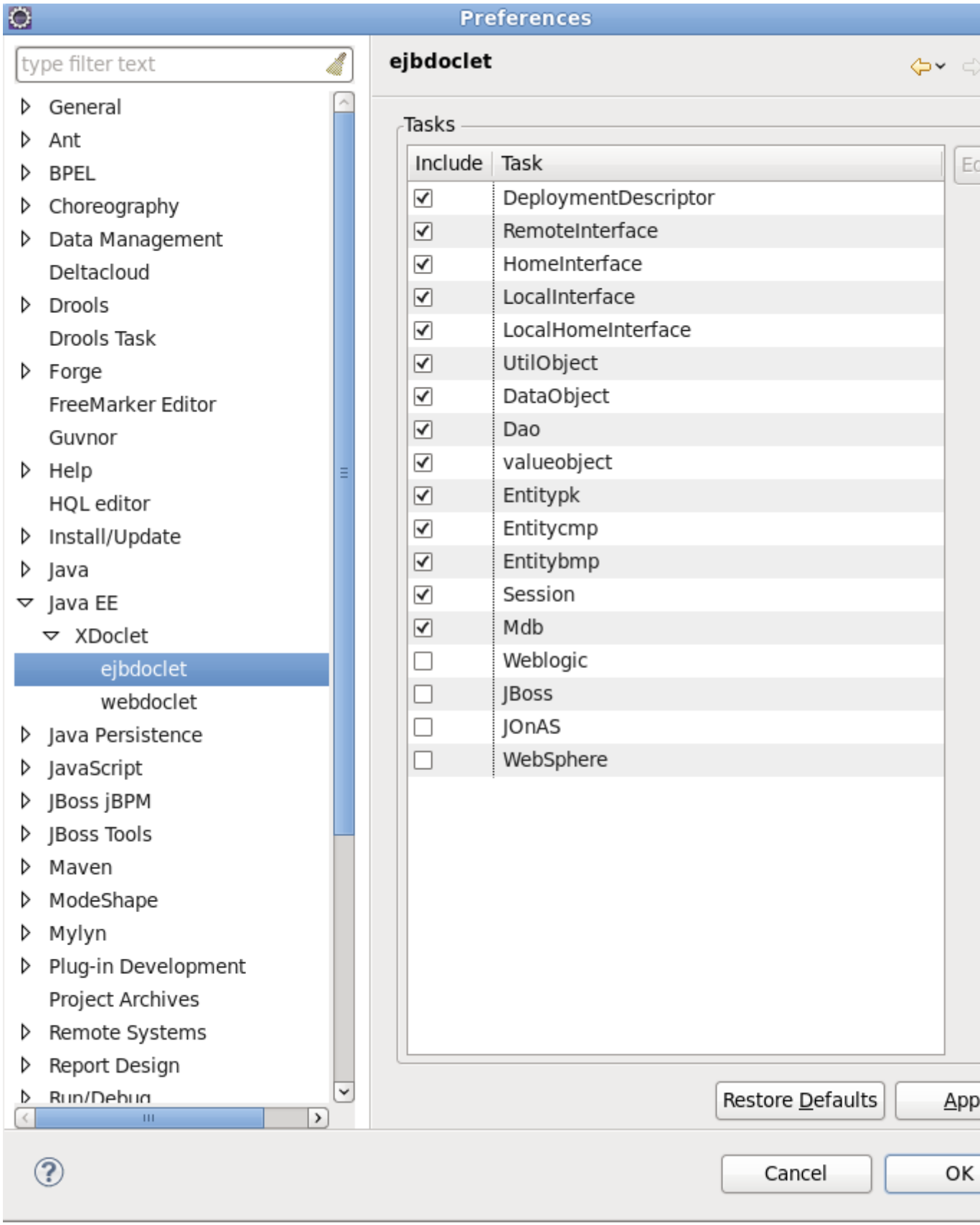

Figure 9.25. ejbdoclet

To configure settings for various web-specific XDoclet sub-tasks, follow to **Java EE** → **XDoclet** → **webdoclet** page.

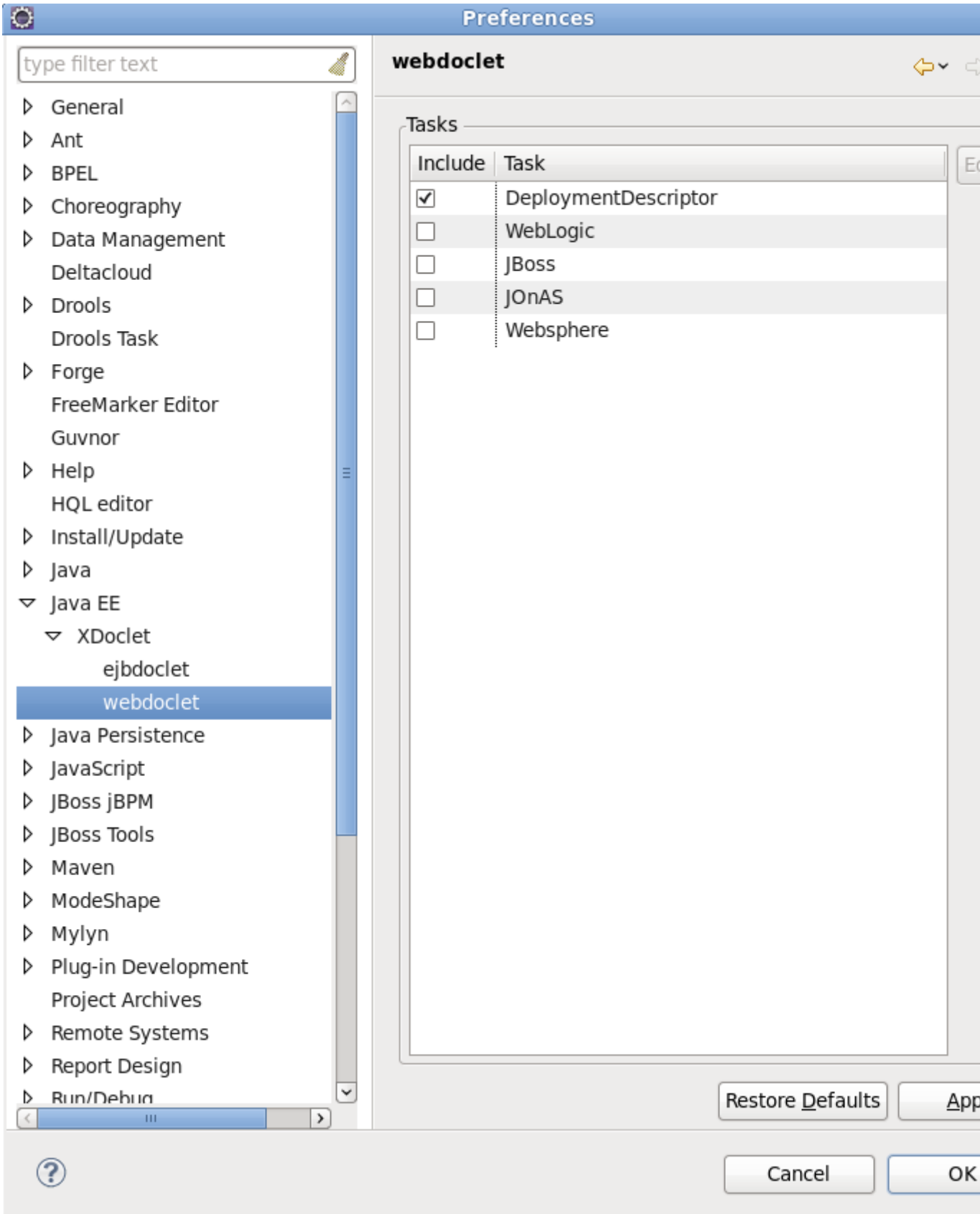

Figure 9.26. webdoclet

Context Menu Preferences and Options

To adjust the project specific preferences, you should bring the context menu for your project and select the **Properties** option. More details on what adjustments you can perform in the Preferences screen, see in the *Chapter 9, JBoss Tools Preferences* chapter.

Under the **Configure** option in the context menu there are also several actions provided by JBDS:

- Add/Remove JSF Capabilities
- Add Custom Capabilities

10.1. Add/Remove JSF Capabilities

Please, for details refer to the JSF Tools Reference Guide.

10.2. Add Custom Capabilities

You can add custom capabilities to any JSF or Seam project made within JBDS, i.e. add a support of additional frameworks built on top of JSF, such as

- ADF
- Facelets
- JBoss Rich Faces (versions 3.1, 3.2, 3.3)

When the option is selected, the Add Custom Capabilities dialog appears. You should check the needed modules and press the **Finish** button.

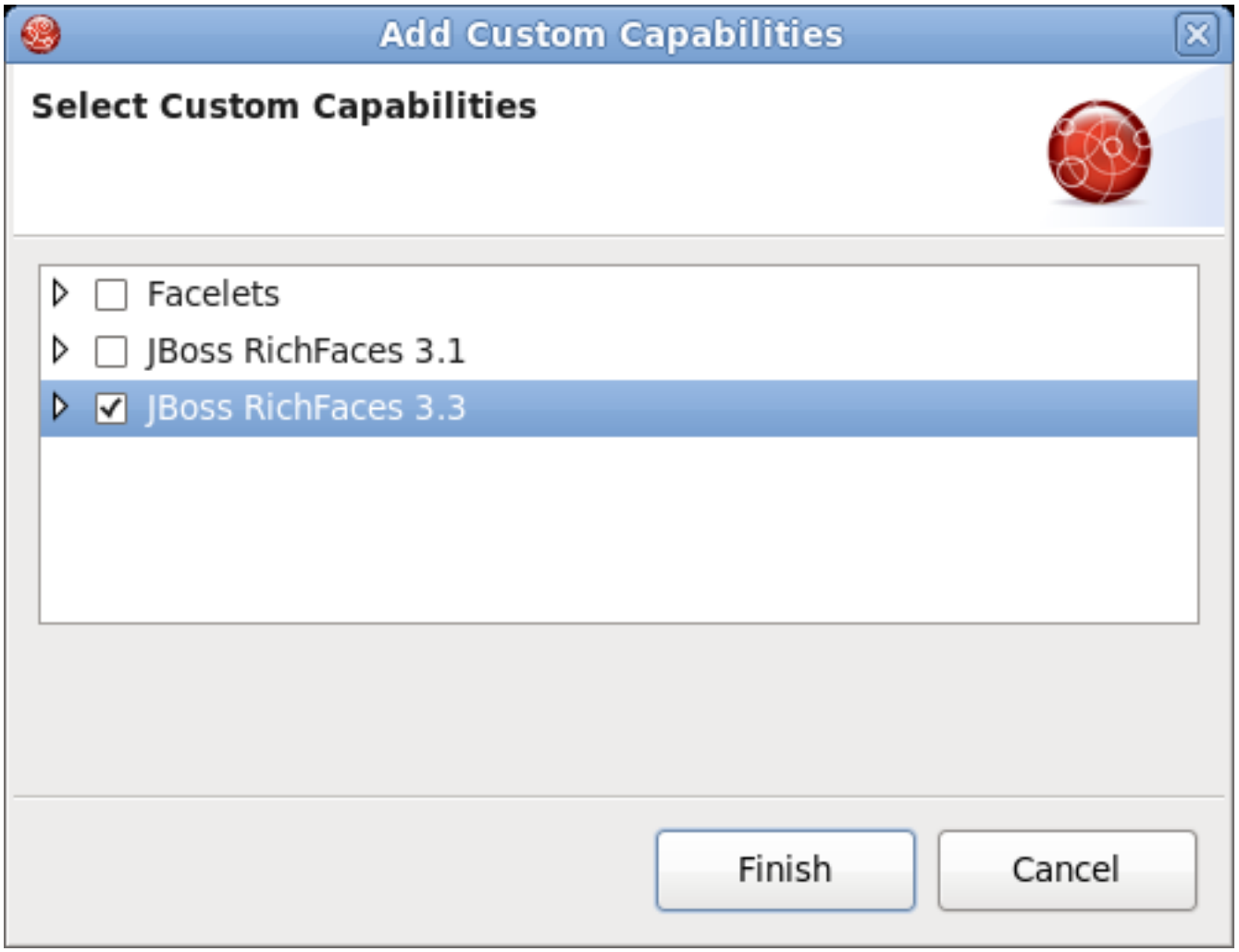

Figure 10.1. Adding Custom Capabilities

The next page displays all the updates that have been made to the project.

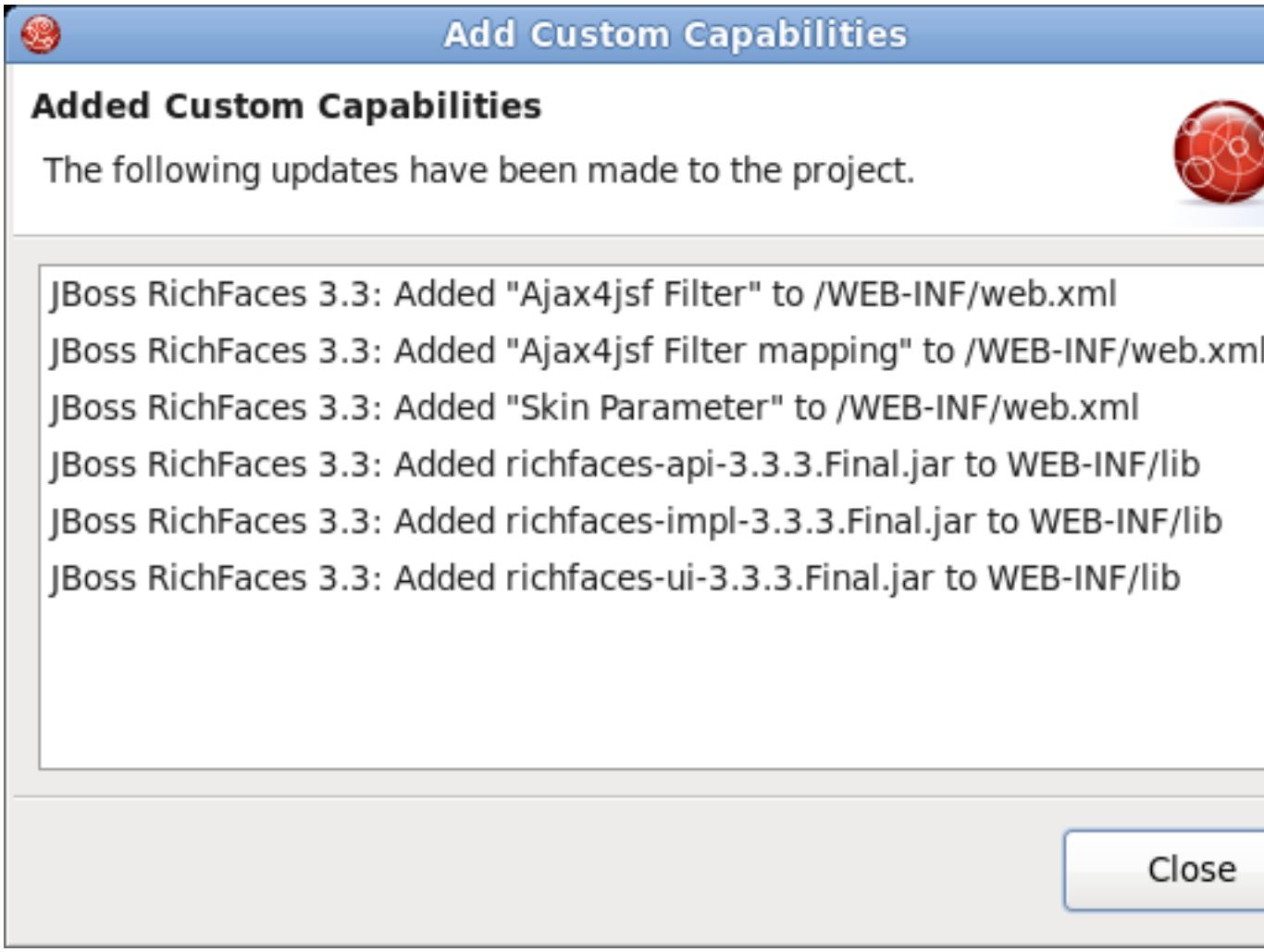

Figure 10.2. Updates Displayed

FAQ

11.1. What should I do if Visual Page Editor does not start under Linux?

The Visual Page Editor requires the library `libstdc++.so.5`. This library is contained in the `compat-libstdc++-33.i386` package.

- To install this package on Fedora Core or Red Hat Enterprise Linux run the following command:

```
yum install compat-libstdc++-33.i386
```

- On any other rpm based distributions download `libstdc++.so.5` and run the following command:

```
rpm -Uvh compat-libstdc++-33.i386
```

- On Debian based distributions run the following command:

```
apt-get install compat-libstdc++-33.i386
```

In case you have the library installed and you still have issue with starting the Visual Page Editor then close all browser views/editors and leave one Visual Page Editor open and restart eclipse. This should force a load of the right XULRunner viewer.

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

- Decompress `eclipse/plugins/org.eclipse.swt.gtk.linux.x86_3.4.1.v3449c.jar` from `eclipse-SDK-3.4.1-linux-gtk-x86_64.tar.gz`

- Copy `libswt-xulrunner-gtk-3449.so` file to your Fedora Eclipse location.
- Open the file eclipse.ini,which can be found in your Fedora Eclipse location and add the following line:

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

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

11.2. How do I change the auto-formating preferences for the Visual Page Editor?

JBoss HTML/JSP editor uses basic eclipse HTML formatter to format files. So if you want to change preferences of formatter for the Visual Page Editor, you should change it for eclipse html editor (open **Window** → **Preferences** then choose **Web** → **HTML Files** → **Editor**).

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

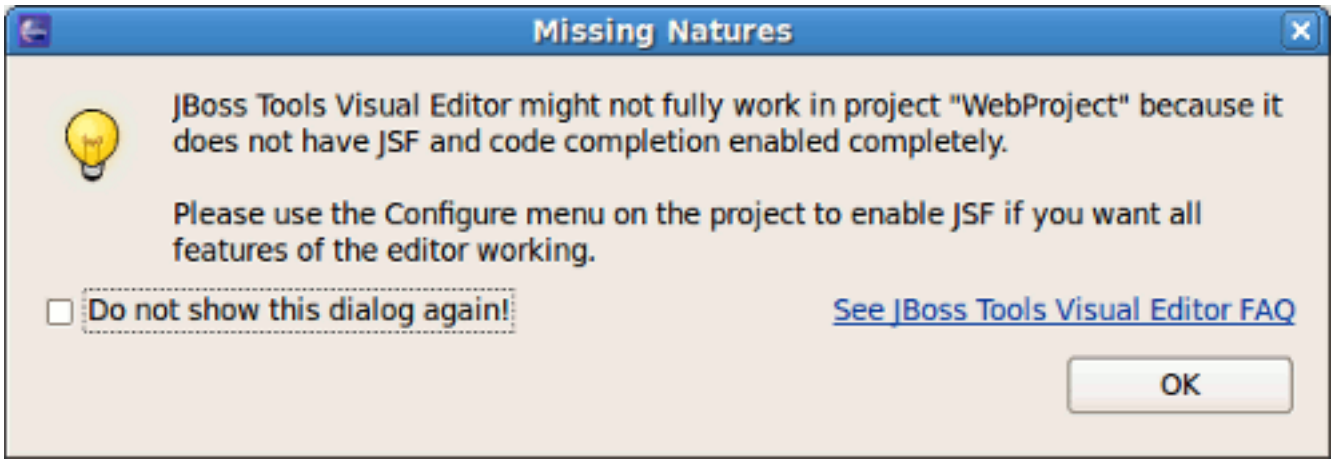

Figure 11.1. Missing Nature

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

1. Right mouse button click on a project in Package Explorer.
2. Select **Configure** → **Add JSF Capabilities** from the context menu.

3. Configure your project using Add JSF Capabilities wizard and press Finish.

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

Conclusion

On the whole, this document should guide you to those parts of JBoss Tools which you specifically need to develop Web Applications. It covers different aspects of visual components such as editors, views, etc. for browsing, representing and editing web resources you are working with.

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