JSF 2.0 Support in NetBeans IDE 6.8

To coincide with the release of Java EE 6, NetBeans IDE 6.8 provides numerous features that enable built-in support for JavaServer Faces 2.0. The IDE's new JSF 2.0 support builds upon its previous support for JavaServer Faces, and includes versatile editor enhancements for Facelets pages, various facilities for working with entity classes, and a suite of JSF wizards for common development tasks, such as creating JSF managed beans, Facelets templates and composite components.

The following topics demonstrate the JSF 2.0 features that are at your disposal when working in NetBeans IDE 6.8. To try out the new JSF features, download the Java Bundle of the NetBeans IDE, which includes Java Web and EE technologies.

Contents

JSF 2.0 Support for Projects

JSF support for projects can be categorized as follows.

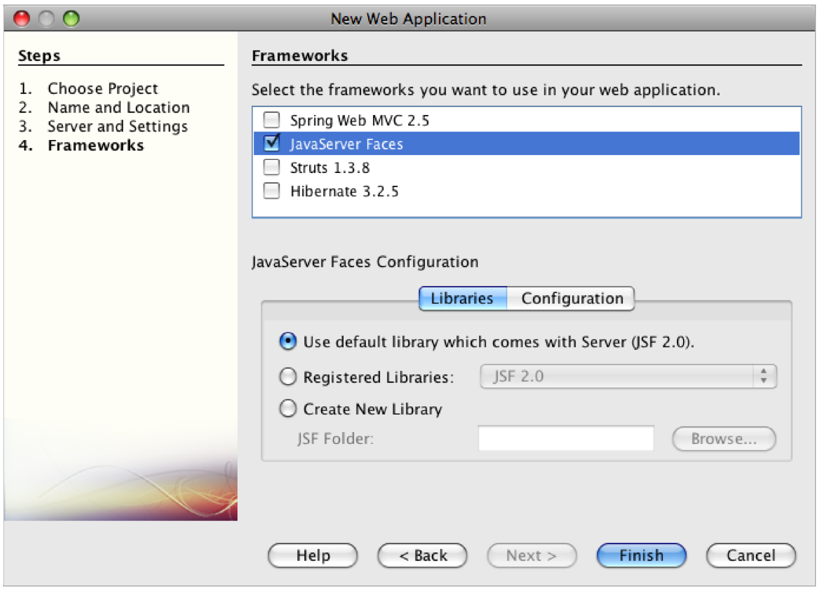
- Facelets template files are included in the project
- JSF 2.0 libraries are added to the project's classpath
- The Faces servlet and servlet mapping are added to the project's deployment descriptor

Using GlassFish, or any other Java EE 6-compliant server, you can create projects with JSF 2.0 support, or add JSF 2.0 support to an existing project.

- Creating a New Project with JSF 2.0 Support
- Adding JSF 2.0 Support to an Existing Project

Creating a New Project with JSF 2.0 Support

Use the IDE's Project wizard to create a new Java web application. To do so, click the New Project ( ) button in the IDE's main toolbar, or press Ctrl-Shift-N (⌘-Shift-N on Mac). When you arrive at Step 4: Frameworks, select JavaServer Faces.



New Web Application

Steps

1. Choose Project
2. Name and Location
3. Server and Settings
4. **Frameworks**

Frameworks

Select the frameworks you want to use in your web application.

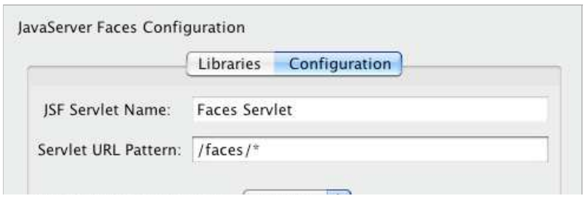
☐ Spring Web MVC 2.5
☑ JavaServer Faces
☐ Struts 1.3.8
☐ Hibernate 3.2.5

JavaServer Faces Configuration

Libraries | Configuration

● Use default library which comes with Server (JSF 2.0).
○ Registered Libraries: JSF 2.0
○ Create New Library
JSF Folder:

After selecting JavaServer Faces, various configuration options become available to you, as shown in the above image. You can determine how your project has access to JSF 2.0 libraries. Click the Configuration tab to specify how the Faces servlet will be registered in the project's deployment descriptor.



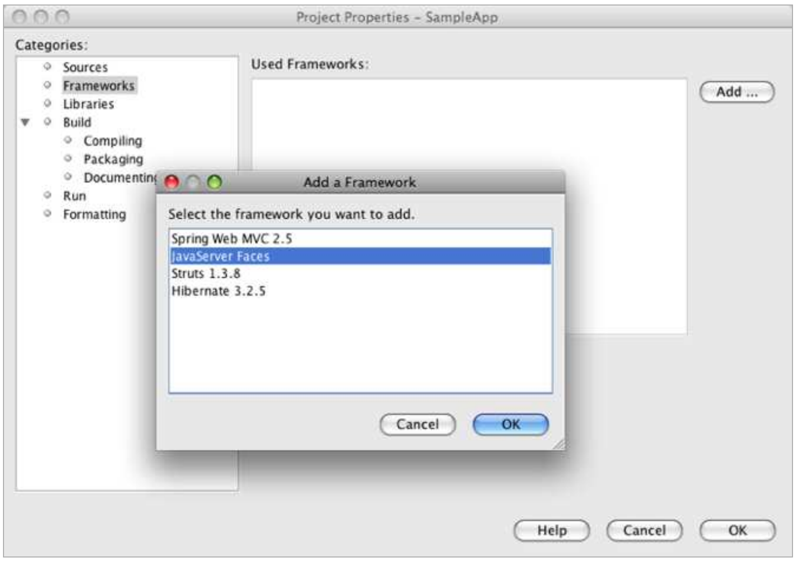
JavaServer Faces Configuration

Libraries | Configuration

JSF Servlet Name: Faces Servlet

Servlet URL Pattern: /faces/*

2. Select the Frameworks category, then click the Add button. In the dialog that displays, select JavaServer Faces.



After selecting JavaServer Faces, various configuration options become available, such as specifying the path to JSF 2.0 libraries, and registration of the Faces servlet in the project's deployment descriptor.

Utilizing the Editor

The IDE's editor is language-specific, and provides support depending on the file type you are working in. Generally speaking, you can press Ctrl-Space on an element in your file to invoke code completion and API documentation. You can also take advantage of keyboard shortcuts and code templates.

> Choose Help > Keyboard Shortcuts Card from the IDE's main menu to view common keyboard shortcuts and code templates. For the full list, see the NetBeans IDE 6.x Keyboard Shortcuts Specification.

The IDE provides built-in Javadoc support for the JSF 2.0 API, as well as JSF's Tag Library Documentation. To take advantage of these resources in your work, simply press Ctrl-Space on a given element in the editor.

> If you prefer continuous access to Javadoc documentation, you can open the IDE's Javadoc window (Window > Other > Javadoc). The Javadoc window automatically refreshes depending on the location of your cursor in the editor.

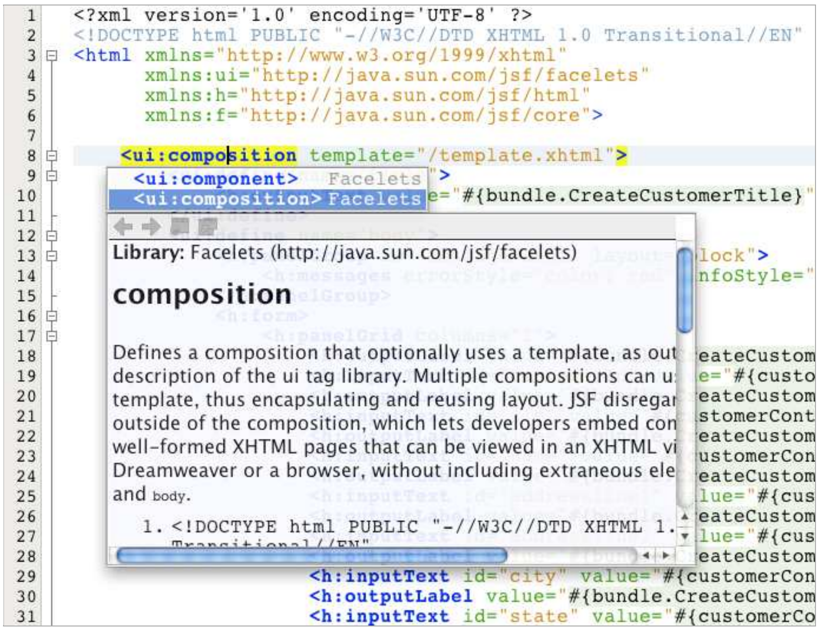
When working on a JSF project, your editing efforts will primarily be spent in Facelets files, JSF managed beans, and the Faces configuration file (`faces-config.xml`). The following briefly demonstrates the editor support that is at your disposal.

- Facelets editor
- Faces XML configuration editor

Facelets Editor

The IDE's Facelets editor provides numerous features that facilitate JSF development, including syntax highlighting and error checking for JSF tags, documentation support, and code completion for EL expressions, core Facelets libraries and namespaces.

You can press Ctrl-Space to invoke code completion and documentation support, where applicable.



```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
<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:composition template="/template.xhtml">
```

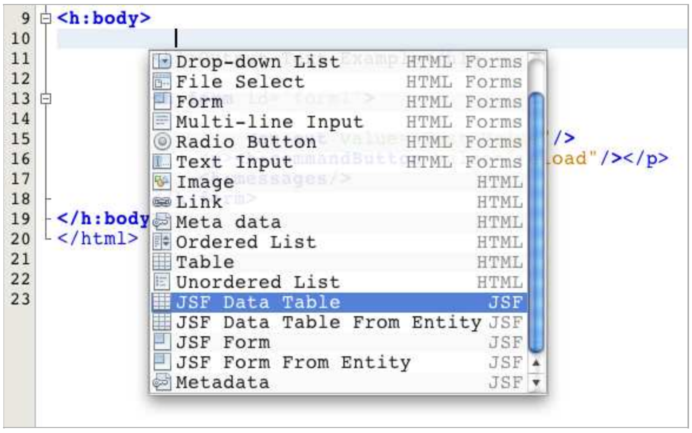
Library: Facelets (http://java.sun.com/jsf/facelets)

composition

Defines a composition that optionally uses a template, as out
description of the ui tag library. Multiple compositions can u
template, thus encapsulating and reusing layout. JSF disregar
outside of the composition, which lets developers embed con
well-formed XHTML pages that can be viewed in an XHTML vi
Dreamweaver or a browser, without including extraneous ele
and body.

```
1. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.
   Transitional//EN"
```

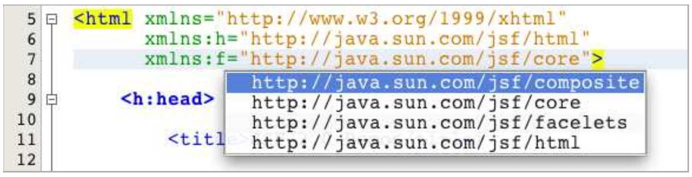
```
                <h:inputText id="city" value="#{customerCon
                <h:outputLabel value="#{bundle.CreateCustom
                <h:inputText id="state" value="#{customerCo
```



```
<h:body>

</h:body>
</html>
```

```
Drop-down List               HTML Forms
File Select                  HTML Forms
Form                         HTML Forms
Multi-line Input             HTML Forms
Radio Button                 HTML Forms
Text Input                   HTML Forms
Image                              HTML
Link                               HTML
Meta data                          HTML
Ordered List                       HTML
Table                              HTML
Unordered List                     HTML
JSF Data Table                      JSF
JSF Data Table From Entity          JSF
JSF Form                            JSF
JSF Form From Entity                JSF
Metadata                            JSF
```

You can press Ctrl-Space to invoke code completion for Facelets namespaces.

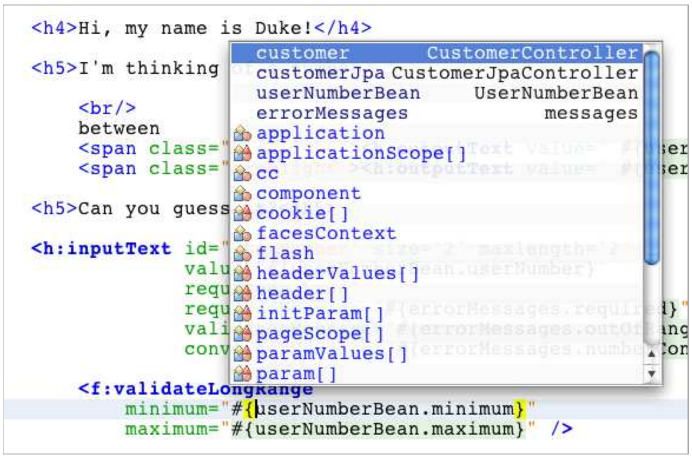


```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">

    <h:head>
        <titl
```

```
http://java.sun.com/jsf/composite
http://java.sun.com/jsf/core
http://java.sun.com/jsf/facelets
http://java.sun.com/jsf/html
```

Similarly, if you type in a JSF tag whose namespace has not been declared in the page, the IDE automatically adds it to the page's `<html>` tag.

The editor provides completion support for Expression Language (EL) syntax. Press Ctrl-Space on EL code to invoke suggestions for implicit objects, JSF managed beans, and their properties.



```
<h4>Hi, my name is Duke!</h4>

<h5>I'm thinking

    <br/>
    between
    <span class="
    <span class="

<h5>Can you guess

<h:inputText id="
             valu
             requ
             requ
             vali
             conv

    <f:validateLongRange
        minimum="#{userNumberBean.minimum}"
        maximum="#{userNumberBean.maximum}" />
```

```
customer               CustomerController
customerJpa         CustomerJpaController
userNumberBean             UserNumberBean
errorMessages                    messages
application
applicationScope[]
cc
component
cookie[]
facesContext
flash
headerValues[]
header[]
initParam[]
pageScope[]
paramValues[]
param[]
```

You can also highlight code snippets in the editor, and choose Convert to Composite Component in order to create JSF composite components. See the Composite Component wizard for more details.

The editor provides basic error checking capabilities. An error displays with a red underline and corresponding badge in the left margin. Warnings are underlined in yellow and are denoted by a yellow badge in the left margin. You can hover your mouse over the badge or underlined text to view a description of the error.

When you enter JSF tags, various checks are performed. These include whether:

- the declared library exists
- the library matched by the tag prefix contains such a component or tag
- the tag contains all required attributes
- all entered attributes are defined in the component's interface

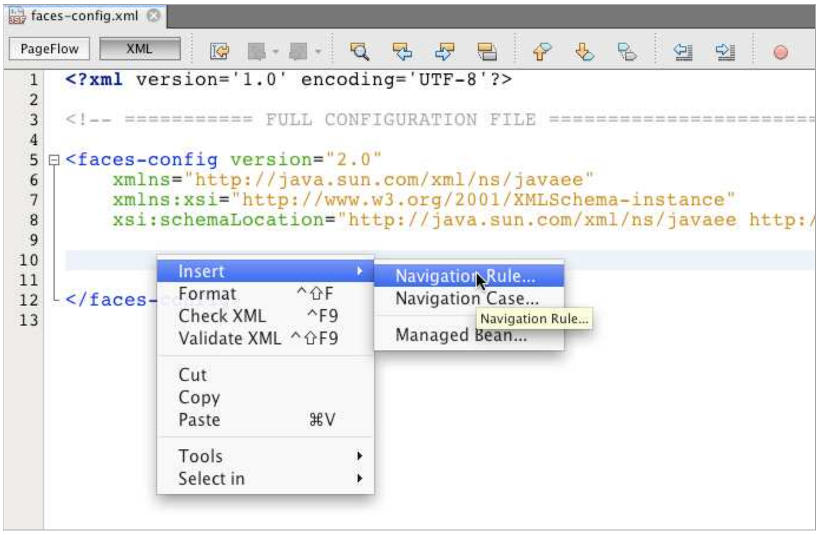
The editor also checks for:

- the existence of undeclared components
- the presence of taglib declarations without usages

Faces XML Configuration Editor

If you include a `faces-config.xml` file in your JSF project, you can press Ctrl-Space when defining navigation rules or declaring managed beans in order to bring up code completion and documentation support.

If you prefer to enter navigation rules and managed beans using dialogs rather than hand-coding them, the IDE provides several JSF-specific dialogs for this purpose. These are accessible from the editor's right-click menu.



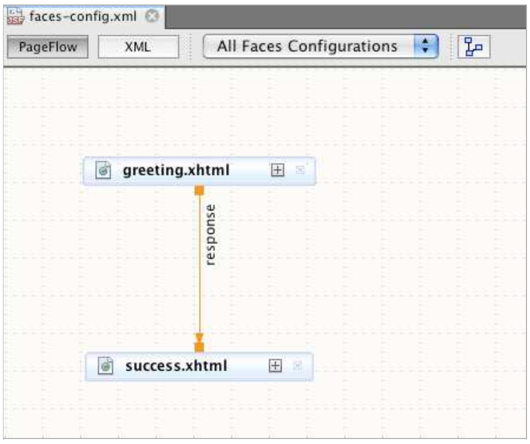
```
<?xml version='1.0' encoding='UTF-8'?>

<!-- =========== FULL CONFIGURATION FILE ======================

<faces-config version="2.0"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http:/


</faces-
```

The IDE provides two distinct *views* for the `faces-config.xml` file: the XML View, which displays the XML source code, and the PageFlow view, which is a graphical interface that depicts JSF navigation rules defined in the `faces-config.xml` file.

For example, if your file contains the following navigation rule:

```
<navigation-rule>
    <from-view-id>/greeting.xhtml</from-view-id>
    <navigation-case>
        <from-outcome>response</from-outcome>
        <to-view-id>/success.xhtml</to-view-id>
    </navigation-case>
</navigation-rule>
```

The PageFlow view displays the following relationship, indicating that a navigation from `greeting.xhtml` to `success.xhtml` occurs when "`response`" is passed to JSF's `NavigationHandler`.

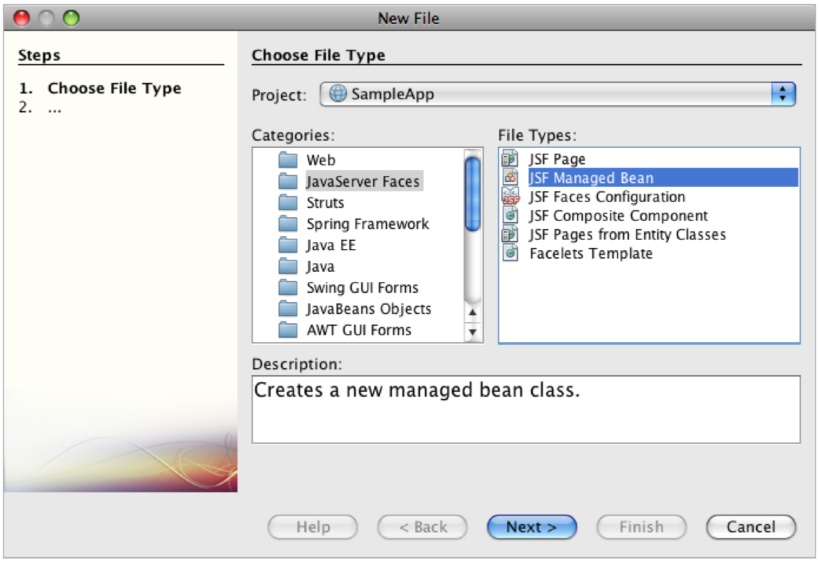


Double-clicking components in the PageFlow view enables you to navigate directly to the source file. For example, when you double-click the `greeting.xhtml` component, the `greeting.xhtml` file opens in the editor. Likewise, if you double-click the arrow between the two components, the editor will focus on the navigation rule defined in the `faces-config.xml` XML view.

JSF Wizards

The NetBeans IDE provides numerous wizards that facilitate development with JSF 2.0. You can create new Facelets pages, Facelets templates, JSF managed beans, composite components, Faces configuration files, and more.

All wizards are accessible via the IDE's generic File wizard. To access the File wizard, press the New File (⊕) button, or choose File > New File from the main menu (or press Ctrl-N; ⌘-N on Mac). JSF-specific wizards are listed within the JavaServer Faces category.



Project: SampleApp

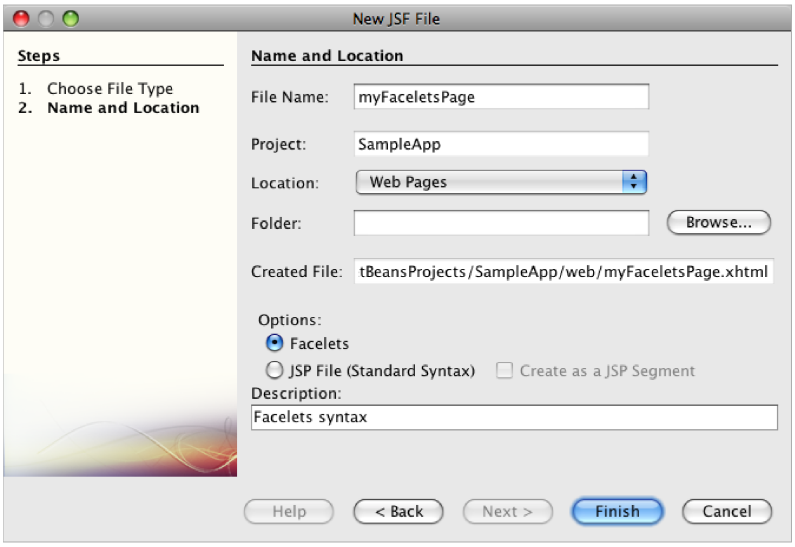
Description:
Creates a new managed bean class.

The following wizards are available to you when working in a Java web project with JSF support.

- JSF Page Wizard
- JSF Managed Bean Wizard
- Faces Configuration Wizard
- Composite Component Wizard
- JSF Pages from Entity Classes Wizard
- Facelets Template Wizard

JSF Page Wizard

Use the JSF Page wizard to create Facelets and JSP pages for your project. In the IDE's File wizard, select the JavaServer Faces category, then select JSF Page. In JSF 2.0, Facelets is the preferred way to declare JSF pages. The Facelets option in the wizard is selected by default. Select the JSP File option if you want to create new JSP pages, or JSP fragments (`.jspf` files).



File Name: myFaceletsPage

Project: SampleApp

Location: Web Pages

Folder:

Created File: tBeansProjects/SampleApp/web/myFaceletsPage.xhtml

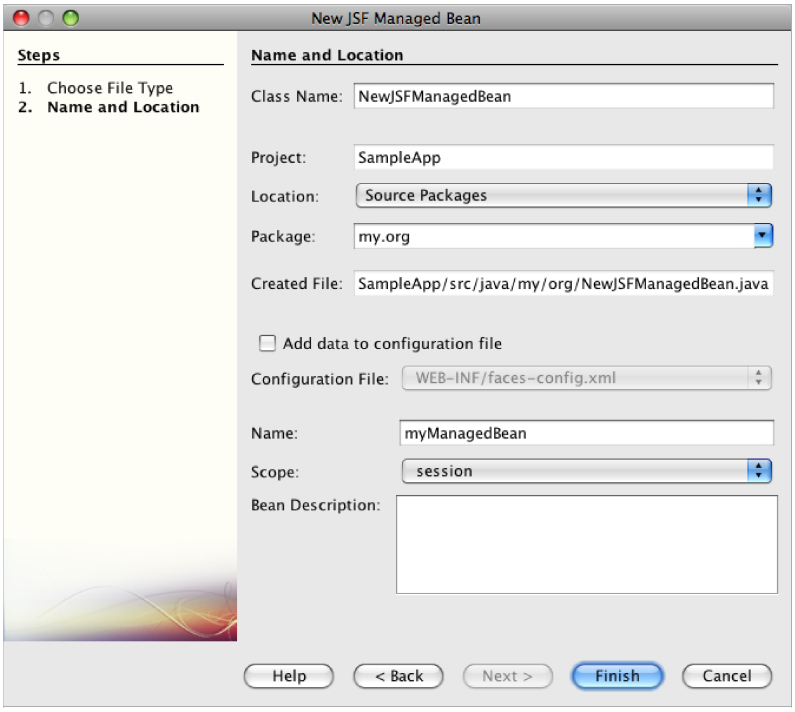
Options:
● Facelets
○ JSP File (Standard Syntax) □ Create as a JSP Segment

Description:
Facelets syntax

Managed Bean Wizard

You can create JSF managed beans for your application using the IDE's Managed Bean wizard. From the JavaServer Faces category in the IDE's File wizard, select JSF Managed Bean.

By default, any metadata that you specify in the wizard is translated into annotations that are applied to the managed bean once it is generated. For example, in the image below, you can create a new, session-scoped class named `NewJSFManagedBean` and name it `myManagedBean`.



New JSF Managed Bean

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name: NewJSFManagedBean

Project: SampleApp

Location: Source Packages

Package: my.org

Created File: SampleApp/src/java/my/org/NewJSFManagedBean.java

☐ Add data to configuration file

Configuration File: WEB-INF/faces-config.xml

Name: myManagedBean

Scope: session

Bean Description:

When the managed bean is generated, it appears as follows with appropriate annotations.

```
package my.org;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;

@ManagedBean(name="myManagedBean")
@SessionScoped
public class NewJSFManagedBean {

    /** Creates a new instance of NewJSFManagedBean */
    public NewJSFManagedBean() {
    }

}
```

Also, if your project already contains a `faces-config.xml` file, the wizard's 'Add data to configuration file' option becomes active, enabling you to either declare the managed bean in the Faces configuration file, or have any metadata specified via annotations in the managed bean.

Faces Configuration Wizard

JSF 2.0 introduces annotations as an alternative to the standard Faces configuration file (`faces-config.xml`) for configuring your application. Therefore, when adding JSF 2.0 support to a project, the IDE *does not* generate a default `faces-config.xml` file (as was the case for JSF 1.2). Naturally, you may want to add a `faces-config.xml` file to your project in order to define certain configuration settings. To do so, use the IDE's Faces Configuration wizard.

From the JavaServer Faces category in the IDE's File wizard, select JSF Faces Configuration. This enables you to create a new `faces-config.xml` file, which is placed in your project's `WEB-INF` folder by default.

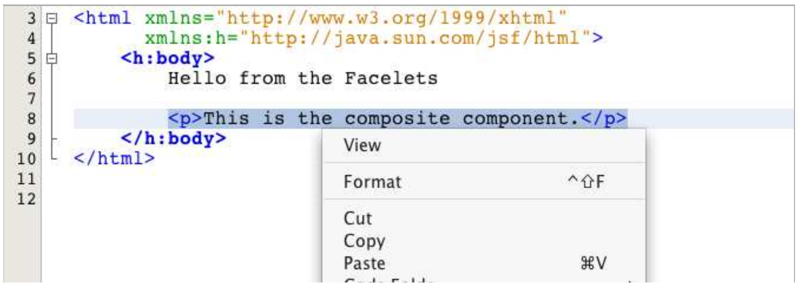
> See Faces XML configuration editor for a description of the IDE's editor support for `faces-config.xml`.

Composite Component Wizard

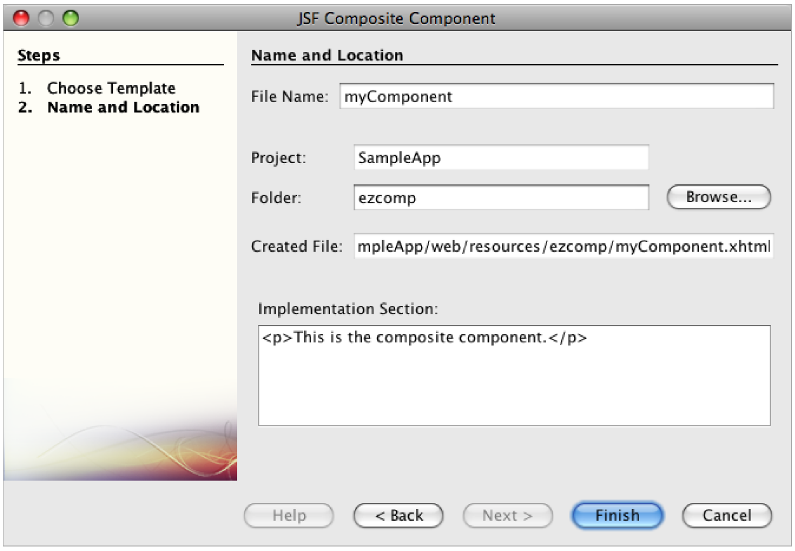
JSF 2.0 has simplified the process of creating composite user interface (UI) components, which can be reused in web pages. You can use the IDE's Composite Component wizard to generate a Facelets template for a JSF composite component.

Like all JSF-related wizards, you can access the Composite Component wizard from the JavaServer Faces category in the IDE's File wizard. However, a more intuitive way to prompt the wizard is by highlighting the code snippet from a Facelets page in the editor, then choosing Convert to Composite Component from the right-click menu.

The following example describes the actions that occur, and facilities at your disposal, when invoking the Composite Component wizard on the snippet, '`<p>This is the composite component.</p>`'.



```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
    <h:body>
        Hello from the Facelets

        <p>This is the composite component.</p>
    </h:body>
</html>
```



JSF Composite Component

Steps

1. Choose Template
2. **Name and Location**

Name and Location

File Name: myComponent

Project: SampleApp

Folder: ezcomp

Created File: mpleApp/web/resources/ezcomp/myComponent.xhtml

Implementation Section:

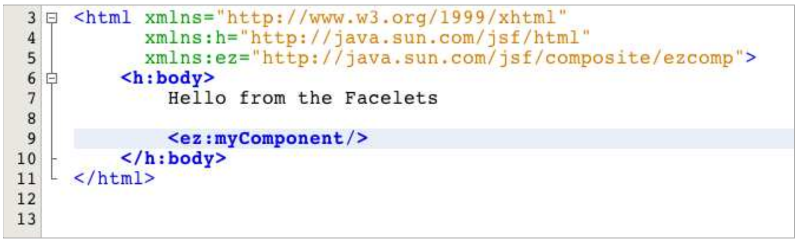
```
<p>This is the composite component.</p>
```

By default, the wizard creates an `ezcomp` folder to contain composite components. For example, if you are creating a new component named `myComponent`, the wizard generates a `myComponent.xhtml` Facelets page, residing in the `resources/ezcomp` folder of your application's web root.

When you complete the wizard, the composite component source file is generated for the given code snippet. The template includes a reference to JSF 2.0's `composite` tag library.

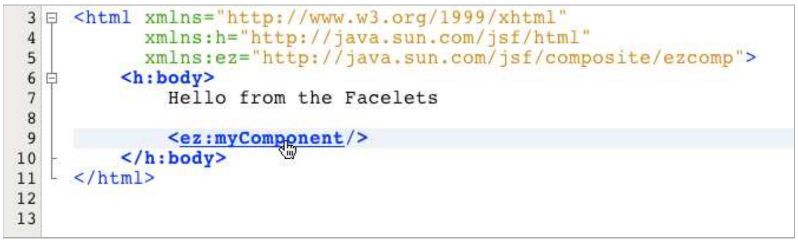
```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xht
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:composite="http://java.sun.com/jsf/composite">

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

  <!-- IMPLEMENTATION -->
  <composite:implementation>
    <p>This is the composite component.</p>
  </composite:implementation>
</html>
```

Also, a new component tag is inserted into the location in the editor where you highlighted the snippet. In this case, the generated tag is: `<ez:myComponent/>`. Note that the IDE automatically adds the namespace where the composite component resides to the page's `<html>` tag.



```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:ez="http://java.sun.com/jsf/composite/ezcomp">
    <h:body>
        Hello from the Facelets

        <ez:myComponent/>
    </h:body>
</html>
```

The IDE also supports hyperlinking to composite component source files. You can navigate to a composite component from a Facelets page by pressing Ctrl (⌘ on Mac) while hovering your mouse over the component tag. Clicking the hyperlink causes the composite component source file to open in the editor.



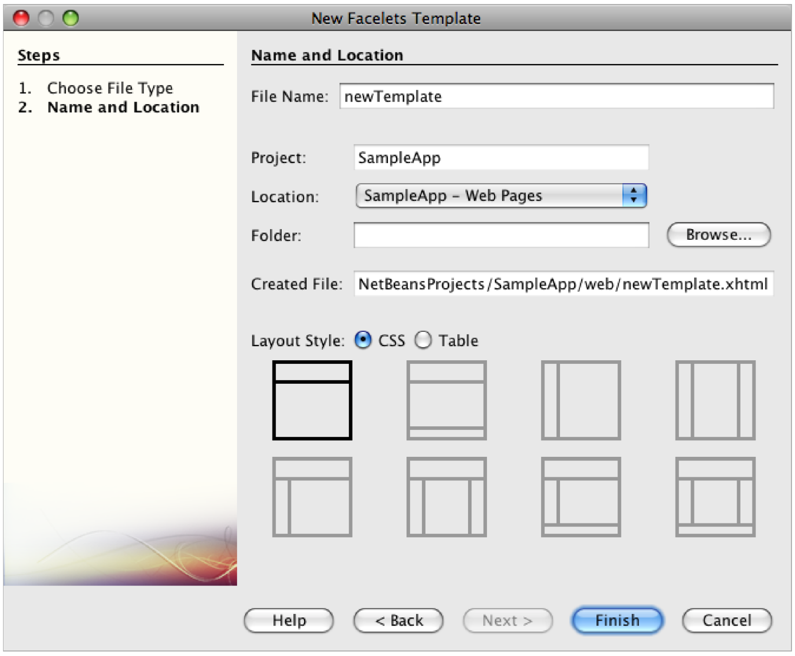
```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:ez="http://java.sun.com/jsf/composite/ezcomp">
    <h:body>
        Hello from the Facelets

        <ez:myComponent/>
    </h:body>
</html>
```

For more information on composite components in JSF 2.0, see True Abstraction: Composite UI Components in JSF 2.0.

JSF Pages from Entity Classes Wizard

See the topic Creating JSF Pages from Entity Classes under Support for Entity Classes.

Facelets Template Wizard


New Facelets Template

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

File Name: newTemplate

Project: SampleApp

Location: SampleApp - Web Pages

Folder:

Created File: NetBeansProjects/SampleApp/web/newTemplate.xhtml

Layout Style: ● CSS ○ Table

The wizard creates an XHTML template file using `<h:head>` and `<h:body>` tags, and places associated stylesheets in the `resources/css` folder of your application's web root. The wizard generates a `default.css` file, and a `cssLayout.css` or `tableLayout.css` file, depending on your layout selection.

To view your template in a browser, right-click in the editor and choose View. A browser window opens to display the template.

Support for Entity Classes

If you are using Java persistence in your application and have entity classes based on your database schema, the IDE provides functionality that lets you work efficiently with entity class data.

Note: To create entity classes from a database table, use the IDE's Entity Classes from Database wizard, accessible from the Persistence category in the IDE's File wizard.

- Creating JSF Pages from Entity Classes
- Creating a JSF Form for Entity Data
- Creating a JSF Data Table for Entity Data

Creating JSF Pages from Entity Classes

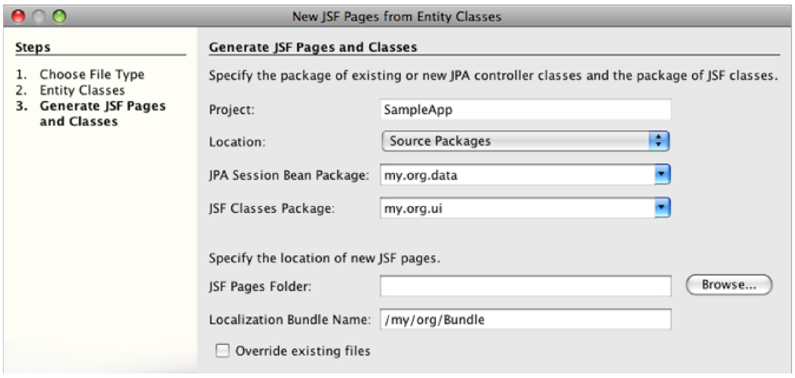
Once you have entity classes in your application, you can use the IDE's JSF Pages from Entity Classes wizard to create a web interface for displaying and modifying entity class data. The code generated by the wizard is based on persistence annotations contained in the entity classes.

For each entity class, the wizard generates the following:

- a stateless session bean for creation, retrieval, modification and removal of entity instances
- a JSF session-scoped, managed bean
- a directory containing four Facelets files for CRUD capabilities (`Create.xhtml`, `Edit.xhtml`, `List.xhtml`, and `View.xhtml`)
- utility classes used by the JSF managed beans (`JsfUtil`, `PaginationHelper`)
- a properties bundle for localized messages, and a corresponding entry in the project's Faces configuration file (A `faces-config.xml` file is created if one does not already exist.)
- auxilary web files, including a default stylesheet for rendered components, and a Facelets template file

To use the JSF Pages from Entity Classes wizard, access the IDE's File wizard. Select the JavaServer Faces category, then select JSF Pages from Entity Classes.

When you reach Step 3: Generate JSF Pages and Classes, you can specify the locations of the files that will be generated.


New JSF Pages from Entity Classes

Steps

1. Choose File Type
2. Entity Classes
3. **Generate JSF Pages and Classes**

Generate JSF Pages and Classes

Specify the package of existing or new JPA controller classes and the package of JSF classes.

Project: SampleApp

Location: Source Packages

JPA Session Bean Package: my.org.data

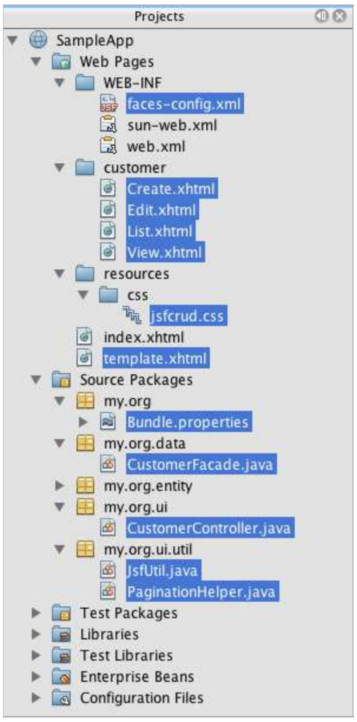
JSF Classes Package: my.org.ui

Specify the location of new JSF pages.

JSF Pages Folder:

Localization Bundle Name: /my/org/Bundle

□ Override existing files



- A `faces-config.xml` file to register the location of the properties bundle that contains localized messages for the JSF views. For example, specifying `/my/org/Bundle` for Localization Bundle Name in the wizard generates the following entry:

```
<application>
    <resource-bundle>
        <base-name>/my/org/Bundle</base-name>
        <var>bundle</var>
    </resource-bundle>
</application>
```

- A `customer` folder in your web root, that contains four Facelets files for CRUD capabilities:
  - `Create.xhtml`: A JSF form for creating a new customer.
  - `Edit.xhtml`: A JSF form for editing a customer.
  - `List.xhtml`: A JSF data table for scrolling through customers.
  - `View.xhtml`: A JSF form for viewing customer details.
- `jsfcrud.css`: A stylesheet used to render the JSF forms and data table.
- `template.xhtml`: An optional Facelets template page, which includes a reference to the generated `jsfcrud.css` stylesheet.
- A stateless session (enterprise) bean named `CustomerFacade`, that resides in the `my.org.data` package. This class can equally be accessed from the project's Enterprise Beans node.
- `Bundle.properties`: A properties bundle that contains default localized messages for the JSF views.
- A JSF session-scoped, managed bean named `CustomerController`, that resides in the `my.org.ui` package.
- Two utility classes (`JsfUtil` and `PaginationHelper`) residing in the `my.org.ui.util` package. These are used by the `CustomerController` managed bean.

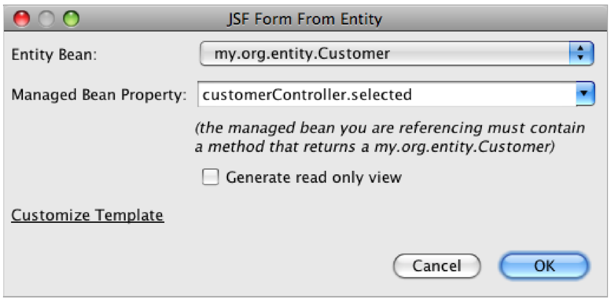
Creating a JSF Form for Entity Data

You can use the Form from Entity dialog to generate a JSF form that contains fields for all properties contained in an entity class. You must already have a JSF managed bean created to handle any user data associated with the form.

Note: If you use this dialog without having an associated managed bean, you can enter a name for the managed bean in the dialog, and that name will be used in the page regardless of whether it is valid or not. You can then create a managed bean using the IDE's Managed Bean wizard, or if you use the JSF Pages from Entity Classes wizard, managed beans are generated for all selected entity classes.

You can access the Form from Entity dialog either by pressing Ctrl-Space in the editor of a Facelets page then choosing JSF Form From Entity, or by double-clicking the Form From Entity item listed in the IDE's Palette (Ctrl-Shift-8; ⌘-Shift-8 on Mac).

For example, in the following image, a `Customer` entity class already exists in the `my.org` package of the given project. A `customerController` managed bean also already exists in the given project, and the managed bean contains a property named `selected` which returns a `Customer` object.


JSF Form From Entity

Entity Bean: my.org.entity.Customer

Managed Bean Property: customerController.selected

(the managed bean you are referencing must contain a method that returns a my.org.entity.Customer)

☐ Generate read only view

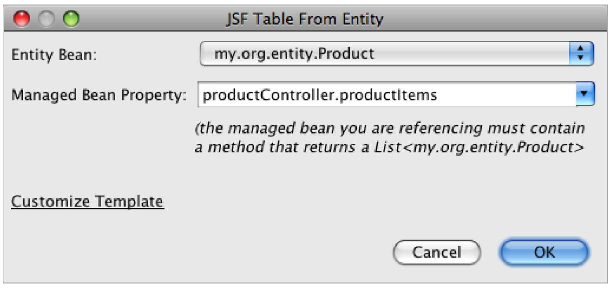
Note: Select the 'Generate read only view' option to create a form that contains read-only fields. When this option is selected, the IDE applies `<h:outputText>` tags for form fields, whereas `<h:inputText>` tags are applied when the option is not selected.

When you complete the dialog, the IDE generates code for your Facelets page. For example, a `Customer` entity class containing a `customerId` property is displayed in the following format:

```
<f:view>
    <h:form>
        <h1><h:outputText value="Create/Edit"/></h1>
        <h:panelGrid columns="2">
            <h:outputLabel value="CustomerId:" for="customerId" />
            <h:inputText id="customerId" value="#{customerController.selected.customerId}" title="C
            ...
            [ Other fields added here. ]
            ...
        </h:panelGrid>
    </h:form>
</f:view>
```

To modify the template used for the generated code, click the Customize Template link within the Form from Entity dialog.

Creating a JSF Data Table for Entity Data



Entity Bean: my.org.entity.Product

Managed Bean Property: productController.productItems

(the managed bean you are referencing must contain a method that returns a List<my.org.entity.Product>)

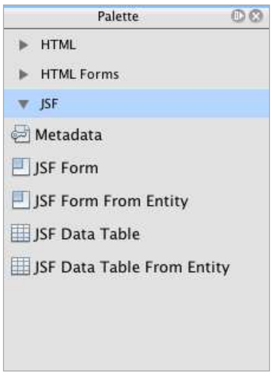
When you complete the dialog, the IDE generates code for your Facelets page. For example, a `Product` entity class containing a `productId` property is displayed in the following format:

```
<f:view>
    <h:form>
        <h1><h:outputText value="List"/></h1>
        <h:dataTable value="#{productController.productItems}" var="item">
            <h:column>
                <f:facet name="header">
                    <h:outputText value="ProductId"/>
                </f:facet>
                <h:outputText value="#{item.productId}"/>
            </h:column>
            ...
            [ Other columns added here. ]
            ...
        </h:dataTable>
    </h:form>
</f:view>
```

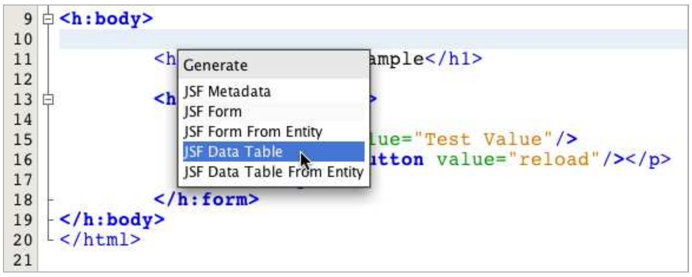
To modify the template used for the generated code, click the Customize Template link within the Form from Data Table dialog.

JSF Palette Components

When working in Facelets pages, you can take advantage of the IDE's Palette to drag and drop JSF tags onto the page. You can access the Palette either by choosing Window > Palette from the main menu, or press Ctrl-Shift-8 (⌘-Shift-8 on Mac).



You can also choose Source > Insert Code (Alt-Insert; Ctrl-I on Mac) from the IDE's main menu to invoke a pop-up list that contains JSF-specific components contained in the Palette.



```
<h:body>

        <h                  ample</h1>

        <h

                            lue="Test Value"/>
                            utton value="reload"/></p>

        </h:form>
</h:body>
</html>
```

The Palette provides you with five JSF-related components:

- **Metadata:** Invokes a dialog to add name-value pairs within JSF metadata tags. For example, if you specify '`myId`' and '`myValue`' as a name-value pair, the following code snippet is produced:

```
<f:metadata>
    <f:viewParam id='myId' value='myValue'/>
</f:metadata>
```

- **JSF Form:** Adds the following code snippet to the page.

```
<f:view>
    <h:form>
        <h:dataTable value="#{}" var="item">
        </h:dataTable>
    </h:form>
</f:view>
```

- **JSF Data Table from Entity:** Invokes a dialog enabling you to associate data from an entity class to fields contained in a JSF data table. See Creating a JSF Data Table for Entity Data.

Send Us Your Feedback

See Also

For more information about JSF 2.0, see the following resources.

- **± NetBeans Articles and Tutorials**
- **± External Resources**
- **± Blogs**