Introduction to Developing Web Applications

This document takes you through the basics of using NetBeans IDE to develop web applications. It demonstrates how to create a simple web application, deploy it to a server, and view its presentation in a browser. The application employs a JavaServer Pages™ (JSP) page to ask you to input your name. It then uses a JavaBeans component to persist the name during the HTTP session, and retrieves the name for output on a second JSP page.

Contents

REQUIRES NetBeans 8.0/7.4/7.3

To follow this tutorial, you need the following software and resources.

Software or Resource	Version Required
NetBeans IDE	7.2, 7.3, 7.4, 8.0, Java EE version
Java Development Kit (JDK)	version 7 or 8
GlassFish Server Open Source Edition *or*	4.x
Tomcat servlet container *or*	7.x or 8.x
Oracle Web Logic server	11gR1 (10.3.3) or later

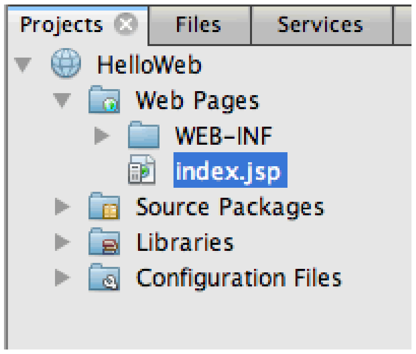
Notes:

- The Java EE installation (not Java SE!) enables you to optionally install the GlassFish server and the Apache Tomcat servlet container.
- If you would like to compare your project with a working solution, you can download the sample application.

Setting Up a Web Application Project

1. Choose File > New Project (Ctrl-Shift-N) from the main menu. Under Categories, select Java Web. Under Projects, select Web Application then click Next.
2. In Step 2, enter `HelloWeb` in the Project Name text box.
3. Specify the Project Location to any directory on your computer. For purposes of this tutorial, this directory is referred to as *$PROJECTHOME*.

4. (Optional) Select the Use Dedicated Folder for Storing Libraries checkbox and specify the location for the libraries folder. See Creating Java Projects in *Developing Applications with NetBeans IDE* for more information on this option.
5. Click Next. The Server and Settings panel opens. Select the version of Java EE you want to use with your application.
6. Select the server to which you want to deploy your application. Only servers that are registered with the IDE are listed. Note that the Context Path (i.e., on the server) becomes `/HelloWeb`, which is based on the name you gave the project in a previous step.
7. Click Finish.
   The IDE creates the *$PROJECTHOME*`/HelloWeb` project folder. You can view the project's file structure in the Files window (Ctrl-2), and its logical structure in the Projects window (Ctrl-1).



The project folder contains all of your sources and project metadata, such as the project's Ant build script. The HelloWeb project opens in the IDE. The welcome page, `index.jsp`, opens in the Source Editor in the main window.

> **Note.** Depending on the server and Java EE version that you specified when you created the project, the IDE might generate `index.html` as the default welcome page for the web project. You can perform the steps in this tutorial and use the `index.html` file or you can use the New File wizard to generate an `index.jsp` file to use as the welcome page, in which case you should delete the `index.html` file.

Creating and Editing Web Application Source Files

Creating and editing source files is the most important function that the IDE serves. After all, that is probably what you spend most of your day doing. The IDE provides a wide range of tools that can compliment any developer's personal style, whether you prefer to code everything by hand or want the IDE to generate large chunks of code for you.

Creating a Java Package and a Java Source File

1. In the Projects window, expand the Source Packages node. Note the Source Packages node only contains an empty default package node.
2. Right-click the Source Packages node and choose New > Java Class. Enter `NameHandler` in the Class Name text box and type `org.mypackage.hello` in the Package combo box. Click Finish. Notice that the new `NameHandler.java` file opens in the Source Editor.
3. In the Source Editor, declare a `String` variable by typing the following line directly below the class declaration.

```
String name;
```

4. Add the following constructor to the class:

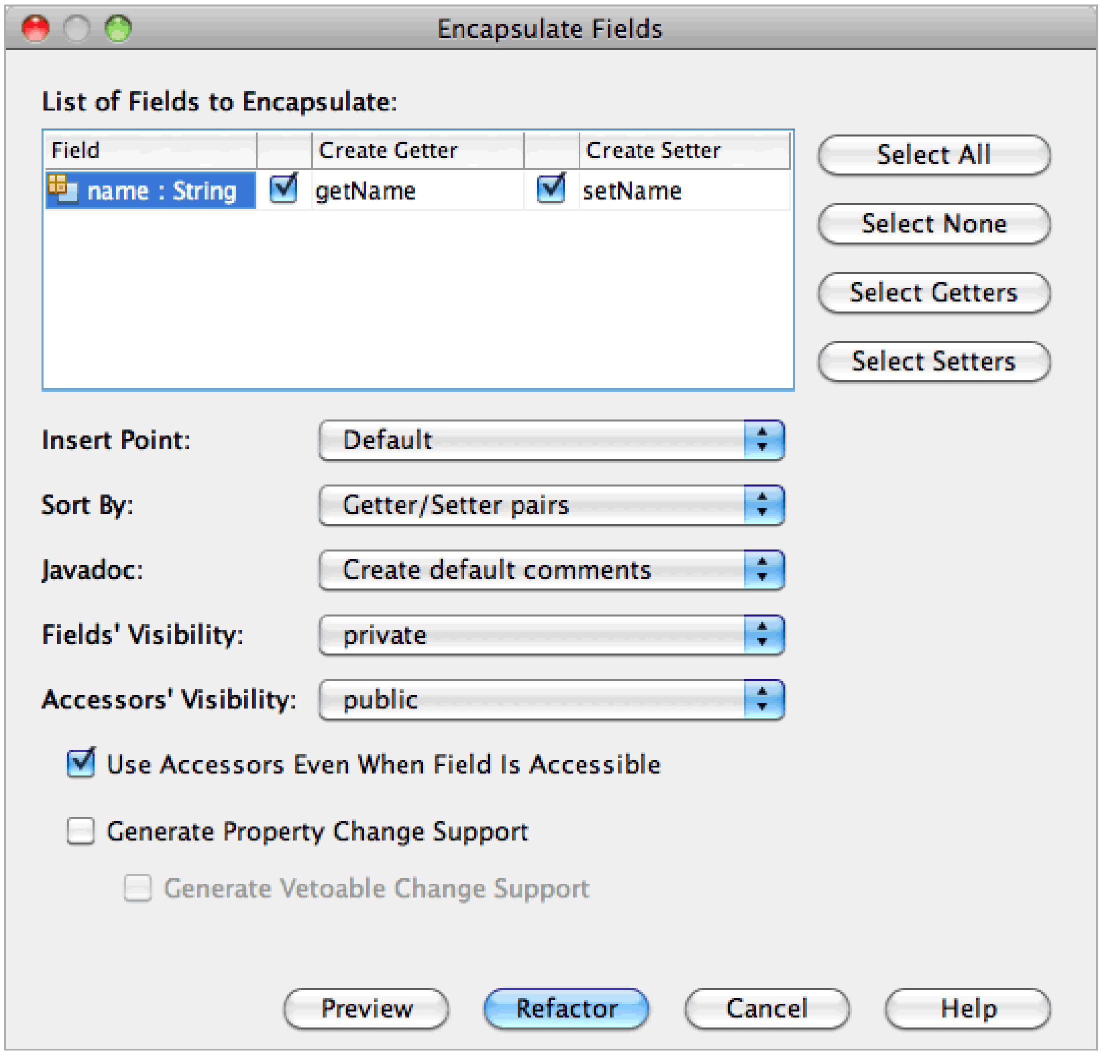
```
public NameHandler() { }
```

5. Add the following line in the `NameHandler()` constructor:

```
name = null;
```

Generating Getter and Setter Methods

1. Right-click the `name` field in the Source Editor and choose Refactor > Encapsulate Fields.
   The Encapsulate Fields dialog opens, listing the `name` field. Notice that Fields' Visibility is by default set to private, and Accessors' Visibility to public, indicating that the access modifier for class variable declaration will be specified as private, whereas getter and setter methods will be generated with `public` and `private` modifiers, respectively.



Encapsulate Fields

List of Fields to Encapsulate:

Field		Create Getter		Create Setter
name : String	✓	getName	✓	setName

Insert Point: Default

Sort By: Getter/Setter pairs

Javadoc: Create default comments

Fields' Visibility: private

Accessors' Visibility: public

[✓] Use Accessors Even When Field Is Accessible

[ ] Generate Property Change Support

[ ] Generate Vetoable Change Support

2. Click Refactor.
   Getter and setter methods are generated for the `name` field. The modifier for the class variable is set to `private`

while getter and setter methods are generated with public modifiers. The Java class should now look similar to the following.

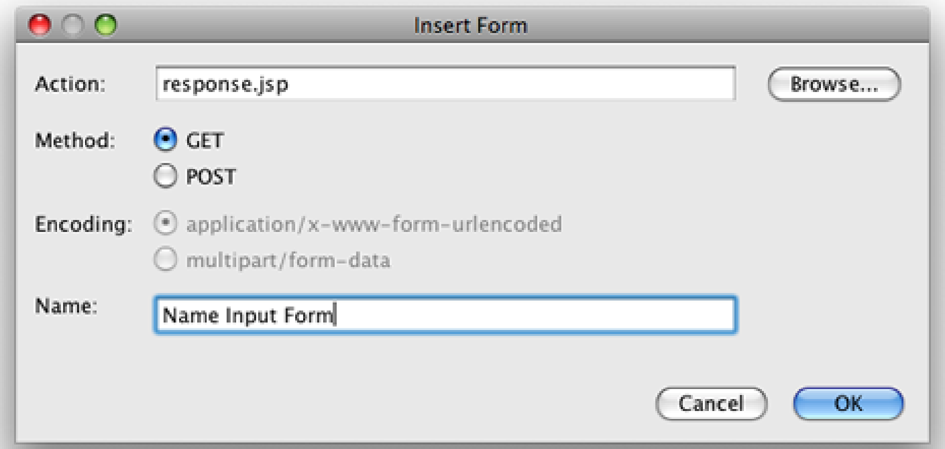
```
package org.mypackage.hello;

/**
 *
 * @author nbuser
 */

public class NameHandler {

    private String name;

    /** Creates a new instance of NameHandler */
    public NameHandler() {
       name = null;
    }

    public String getName() {
       return name;
    }

    public void setName(String name) {
       this.name = name;
    }

}
```

Editing the Default JavaServer Pages File

1. Refocus the `index.jsp` file by clicking its tab displayed at the top of the Source Editor.
2. In the Palette (Ctrl-Shift-8) located to the right of the Source Editor, expand HTML Forms and drag a Form item to a point after the `<h1>` tags in the Source Editor.

   The Insert Form dialog box displays.
3. Specify the following values:
   - **Action:** response.jsp
   - **Method:** GET
   - **Name:** Name Input Form

   Click OK. An HTML form is added to the `index.jsp` file.


Insert Form

Action: response.jsp

Method: [✓ GET] POST

Encoding: [✓ application/x-www-form-urlencoded] multipart/form-data

Name: Name Input Form

4. Drag a Text Input item to a point just before the `</form>` tag, then specify the following values:
   - **Name:** name
   - **Type:** text

   Click OK. An HTML `<input>` tag is added between the `<form>` tags. Delete the `value` attribute from this tag.
5. Drag a Button item to a point just before the `</form>` tag. Specify the following values:
   - **Label:** OK
   - **Type:** submit

   Click OK. An HTML button is added between the `<form>` tags.
6. Type `Enter your name:` just before the first `<input>` tag, then change the default `Hello World!` text between the `<h1>` tags to `Entry Form`.
7. Right-click within the Source Editor and choose Format (Alt-Shift-F) to tidy the format of your code. Your `index.jsp` file should now appear similar to the following:

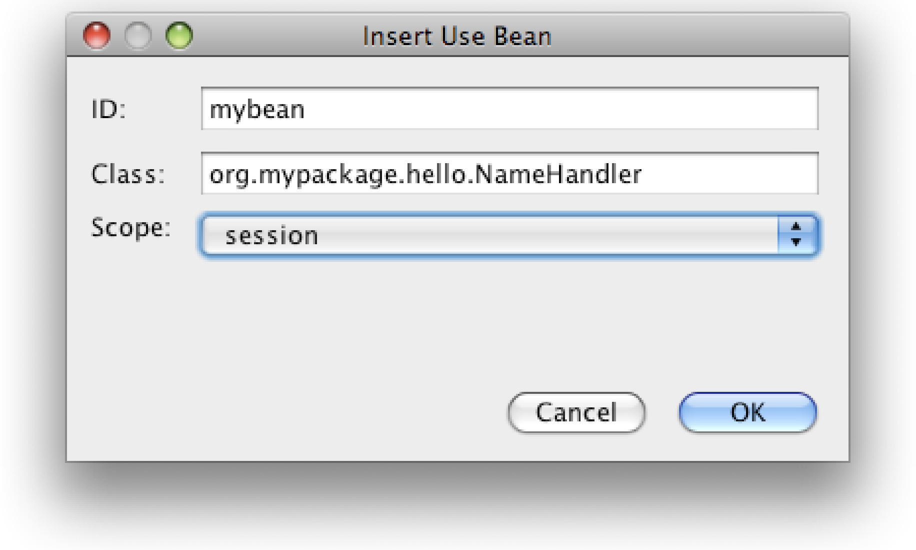
```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Entry Form</h1>

        <form name="Name Input Form" action="response.jsp">
            Enter your name:
            <input type="text" name="name" />
            <input type="submit" value="OK" />
        </form>
    </body>
```

```
</html>
```

Creating a JavaServer Pages File

1. In the Projects window, right-click the HelloWeb project node and choose New > JSP. The New JSP File wizard opens. Name the file `response`, and click Finish. Notice that a `response.jsp` file node displays in the Projects window beneath `index.jsp`, and the new file opens in the Source Editor.

2. In the Palette to the right of the Source Editor, expand JSP and drag a Use Bean item to a point just below the `<body>` tag in the Source Editor. The Insert Use Bean dialog opens. Specify the values shown in the following figure.



ID: mybean
Class: org.mypackage.hello.NameHandler
Scope: session

   - **ID:** mybean
   - **Class:** org.mypackage.hello.NameHandler
   - **Scope:** session

   Click OK. Notice that the `<jsp:useBean>` tag is added beneath the `<body>` tag.

3. Drag a Set Bean Property item from the Palette to a point just before the `<h1>` tag and click OK. In the `<jsp:setProperty>` tag that appears, delete the empty `value` attribute and edit as follows. Delete the `value = ""` attribute if the IDE created it! Otherwise, it overwrites the value for `name` that you pass in `index.jsp`.

```
<jsp:setProperty name="mybean" property="name" />
```

   As indicated in the

   `<jsp:setProperty>` documentation, you can set a property value in various ways. In this case, the user input coming from `index.jsp` becomes a name/value pair that is passed to the `request` object. When you set a property using the `<jsp:setProperty>` tag, you can specify the value according to the name of a property contained in the `request` object. Therefore, by setting `property` to `name`, you can retrieve the value specified by user input.

4. Change the text between the <h1> tags so that it looks like this:

   ```
   <h1>Hello, !</h1>
   ```

5. Drag a Get Bean Property item from the Palette and drop it after the comma between the `<h1>` tags. Specify the following values in the Insert Get Bean Property dialog:
   - **Bean Name:** mybean
   - **Property Name:** name

   Click OK. Notice that `<jsp:getProperty>` tag is now added between the `<h1>` tags.

   Caution: Property names are case-sensitive. The "name" property must be in the same case in `response.jsp` and in the input form in `index.jsp`.

6. Right-click within the Source Editor and choose Format (Alt-Shift-F) to tidy the format of your code. The `<body>` tags of your `response.jsp` file should now appear similar to the following:

   ```
   <body>
       <jsp:useBean id="mybean" scope="session" class="org.mypackage.hello.NameHandler"
   />
       <jsp:setProperty name="mybean" property="name" />
       <h1>Hello, <jsp:getProperty name="mybean" property="name" />!</h1>
   </body>
   ```

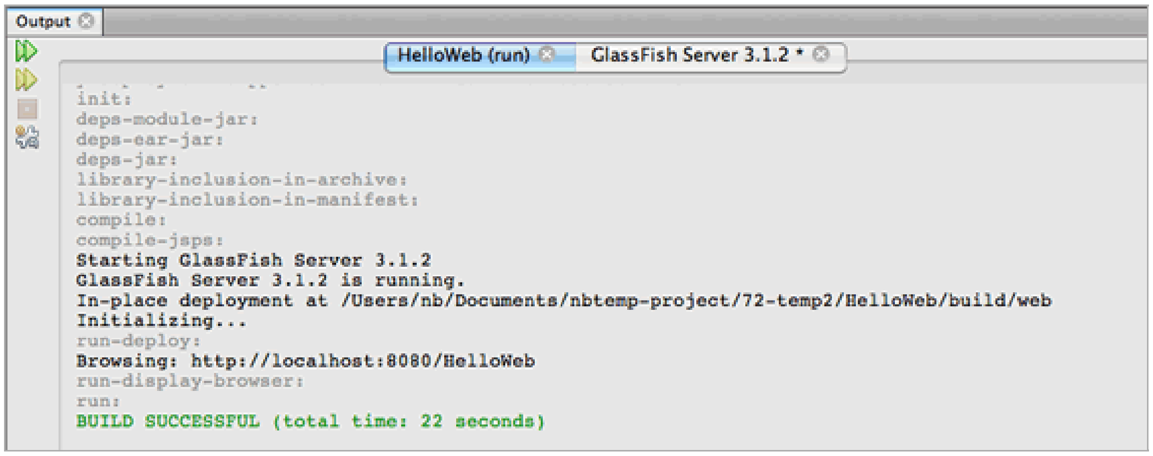
Running a Web Application Project

The IDE uses an Ant build script to build and run your web applications. The IDE generates the build script based on the options you specify in the New Project wizard, as well as those from the project's Project Properties dialog box (In the Projects window, choose Properties from the project node's right click menu).

1. In the Projects window, right-click the HelloWeb project node and choose Run (F6). When you run a web application, the IDE performs the following steps:
   - Building and compiling the application code (see note below). You can perform this step in isolation by selecting Build or Clean and Build from the project node context menu.
   - Launching the server.
   - Deploying the application to the server. You can perform this step in isolation by selecting Deploy from the project node context menu.
   - Displaying the application in a browser window.

   Note: By default, the project has been created with the Compile on Save feature enabled, so you do not need to compile your code first in order to run the application in the IDE.

2. The IDE opens an output window that shows the progress of running the application. Look at the HelloWeb tab in the Output window. In this tab, you can follow all the steps that the IDE performs. If there is a problem, the IDE displays error information in this window.

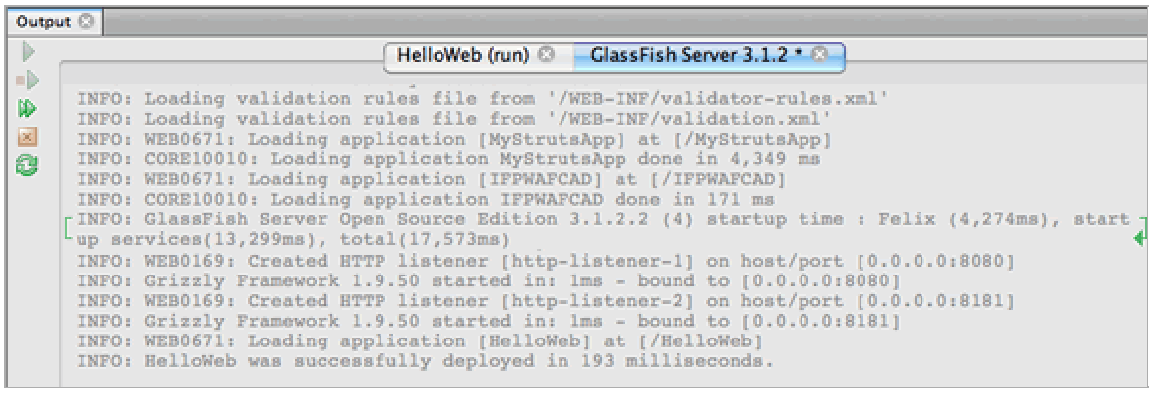


```
init:
deps-module-jar:
deps-ear-jar:
deps-jar:
library-inclusion-in-archive:
library-inclusion-in-manifest:
compile:
compile-jsps:
Starting GlassFish Server 3.1.2
GlassFish Server 3.1.2 is running.
In-place deployment at /Users/nb/Documents/nbtemp-project/72-temp2/HelloWeb/build/web
Initializing...
run-deploy:
Browsing: http://localhost:8080/HelloWeb
run-display-browser:
run:
BUILD SUCCESSFUL (total time: 22 seconds)
```

3. The IDE opens an output window showing the server status. Look at the tab in the Output window with the name of your server.

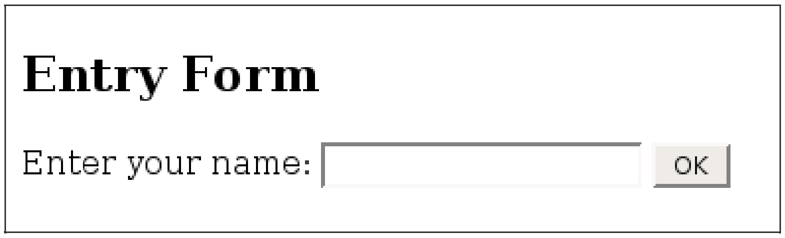
   Important: If the GlassFish server fails to start, start it manually and run the project again. You can start the server manually from the Services window, by right-clicking the server node and selecting Start.

   The server output window is very informative about problems running Web applications. The server's logs can also be helpful. They are located in the server's relevant domain directory. You can also view the IDE log, visible by selecting View > IDE log.



```
INFO: Loading validation rules file from '/WEB-INF/validator-rules.xml'
INFO: Loading validation rules file from '/WEB-INF/validation.xml'
INFO: WEB0671: Loading application [MyStrutsApp] at [/MyStrutsApp]
INFO: CORE10010: Loading application MyStrutsApp done in 4,349 ms
INFO: WEB0671: Loading application [IFPWAFCAD] at [/IFPWAFCAD]
INFO: CORE10010: Loading application IFPWAFCAD done in 171 ms
INFO: GlassFish Server Open Source Edition 3.1.2.2 (4) startup time : Felix (4,274ms), start
up services(13,299ms), total(17,573ms)
INFO: WEB0169: Created HTTP listener [http-listener-1] on host/port [0.0.0.0:8080]
INFO: Grizzly Framework 1.9.50 started in: 1ms - bound to [0.0.0.0:8080]
INFO: WEB0169: Created HTTP listener [http-listener-2] on host/port [0.0.0.0:8181]
INFO: Grizzly Framework 1.9.50 started in: 1ms - bound to [0.0.0.0:8181]
INFO: WEB0671: Loading application [HelloWeb] at [/HelloWeb]
INFO: HelloWeb was successfully deployed in 193 milliseconds.
```

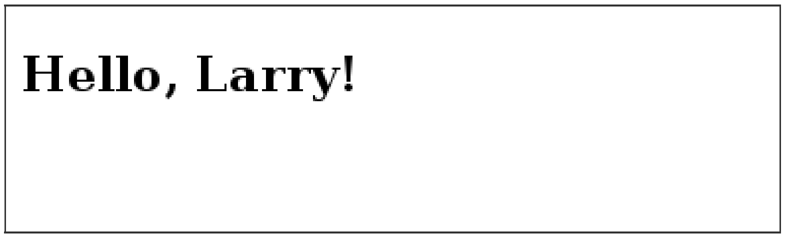
4. The `index.jsp` page opens in your default browser. Note that the browser window may open before the IDE displays the server output.



Entry Form

Enter your name:

5. Enter your name in the text box, then click OK. The `response.jsp` page displays, providing you with a simple greeting.



Hello, Larry!

Troubleshooting

I've built and run the project. When I click the OK button for `index.jsp`, an error page displays indicating that `response.jsp` is not available.

Have you looked in the IDE's Output window (Ctrl-4) in the project tab or in the server tab? What error messages are there? What JDK does your project use? What server? JDK 7 requires GlassFish 3.x or Tomcat 7.x. Right-click the project's node in the Projects window and select Properties. The JDK is in the Libraries category, in the Java Platform field. The server version is in the Run category. Lastly, download the sample project and compare it with your own.

I've built and run the project but no name appears, only "Hello, !"

Does your <jsp:setProperty> tag contain a `value = ""` attribute? This overwrites the value you passed in the `index.jsp` form and replaces it with an empty string. Delete the `value` attribute.

I've built and run the project but get "Hello, null!"

First, check the IDE's Output windows for both application and server, and the server log. Is the server running? Was the application deployed? If the server is running and the application was deployed, are you getting an `org.apache.jasper.JasperException: java.lang.NullPointerException`? This usually means that a value in your code is not initialized correctly. In this tutorial, it means that you probably have a typo somewhere in a property name in your JSP files. Remember that property names are case-sensitive!

Send Feedback on This Tutorial

See Also

This concludes the Introduction to Developing Web Applications tutorial. This document demonstrated how to create a simple web application using NetBeans IDE, deploy it to a server, and view its presentation in a browser. It also showed how to use JavaServer Pages and JavaBeans in your application to collect, persist, and output user data.

For related and more advanced information about developing web applications in NetBeans IDE, see the following resources:

- Introduction to the Struts Web Framework. Describes the basics of using NetBeans IDE to develop web applications using the Struts framework.
- Java EE & Java Web Learning Trail