

## Contents:

1. Machine specification used in this task
2. Pre requirement
3. Setting Up a Web Application Project
4. Creating and Editing Web Application Source Files
5. Creating a Java Package and a Java Source File
6. Generating Getter and Setter Methods
7. Editing the Default JavaServer Pages File
8. Creating a JavaServer Pages (JSP) File
9. Building and Running a Web Application Project

The original tutorial can be found at [netbeans.org](http://netbeans.org). All credits must go to the original authors. In this tutorial we try to refine all the steps to more details using screen shots.

## Machine specification used in this task are:

- Intel Pentium Core 2 Duo, 2.2 GHz,
- Windows XP Pro SP2 + periodical patches + periodical updates...
- 2 GB DDR2 RAM
- 160 GB SATA HDD
- 17" SyncMaster 713N monitor.

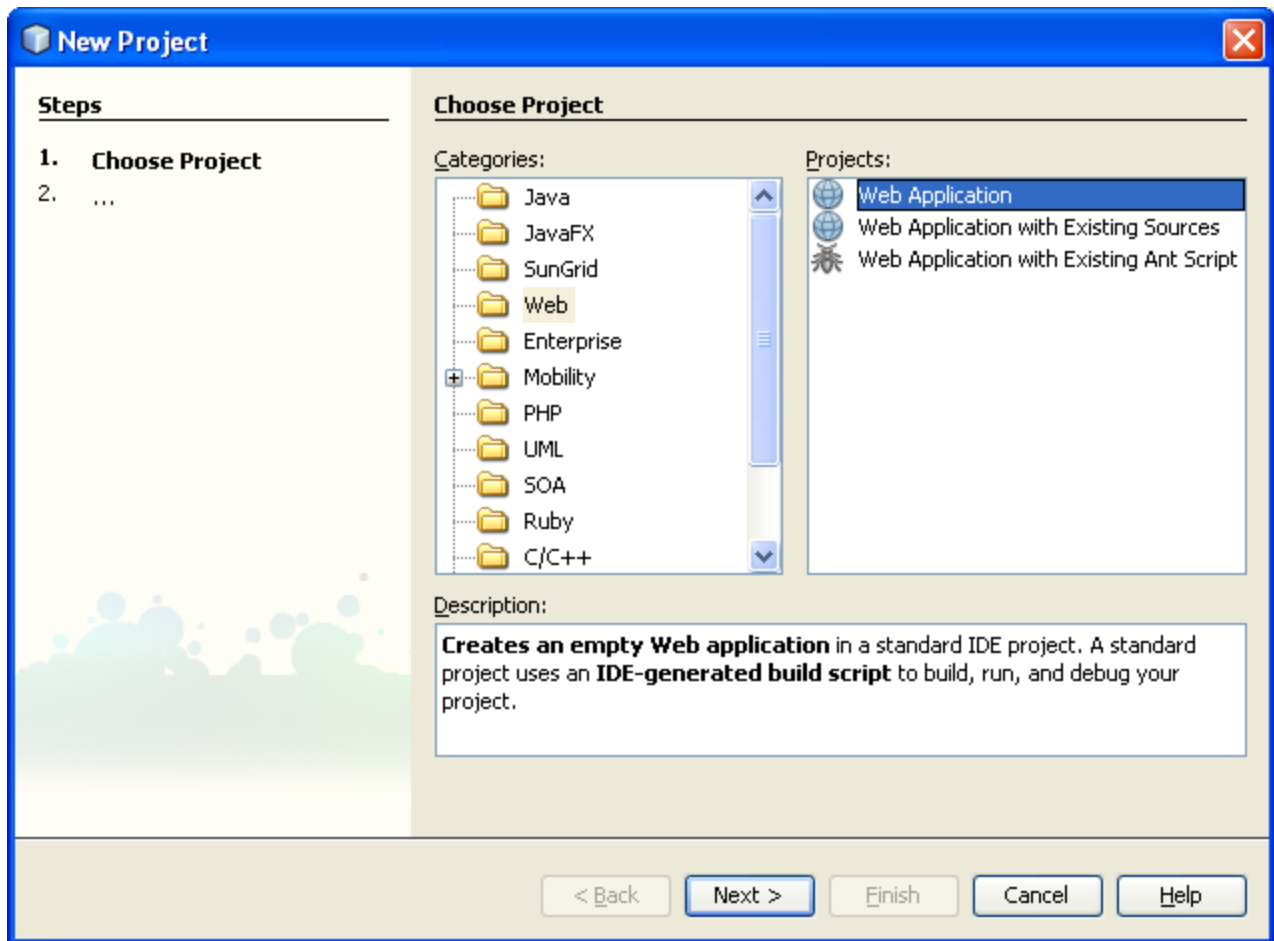
## Pre requirement

1. [NetBeans 6.x.x](#)
2. To work through this tutorial, you must have a server registered in the IDE. The Web and Java EE installation enables you to optionally install and register the Apache Tomcat servlet container 6.0.14, and the GlassFish V2 application server.
3. If you are installing NetBeans IDE for the first time, you need to have the [Java SE Development Kit](#) (JDK) installed. The JDK includes the Java Runtime Environment (JRE), as well as various tools and API's necessary for development in Java.

## Setting Up a Web Application Project

1. Choose File > New Project (Ctrl-Shift-N) from the main menu. Under Categories, select Web. Under Projects, select Web Application then click Next.





2. In Step 2, enter `HelloWeb` in the Project Name text box. Notice that the Context Path (i.e., on the server) becomes `/HelloWeb`.
3. Specify the Project Location to any directory on your computer. For purposes of this tutorial, this directory is referred to as `$PROJECTHOME`.
4. Select the server to which you want to deploy your application. Only servers that are registered with the IDE are listed. In this case we select GlassFish V2. Click Next.

Server:	GlassFish V2	Add...
Java EE Version:	Apache Tomcat 6.0.14 GlassFish V2	
Context Path:	/HelloWeb	

**New Web Application**

**Steps**

1. Choose Project
- 2. Name and Location**
3. Frameworks

**Name and Location**

Project Name: HelloWeb

Project Location: C:\myjavaproject Browse...

Project Folder: C:\myjavaproject\HelloWeb

Add to Enterprise Application: <None>

Server: GlassFish V2 Add...

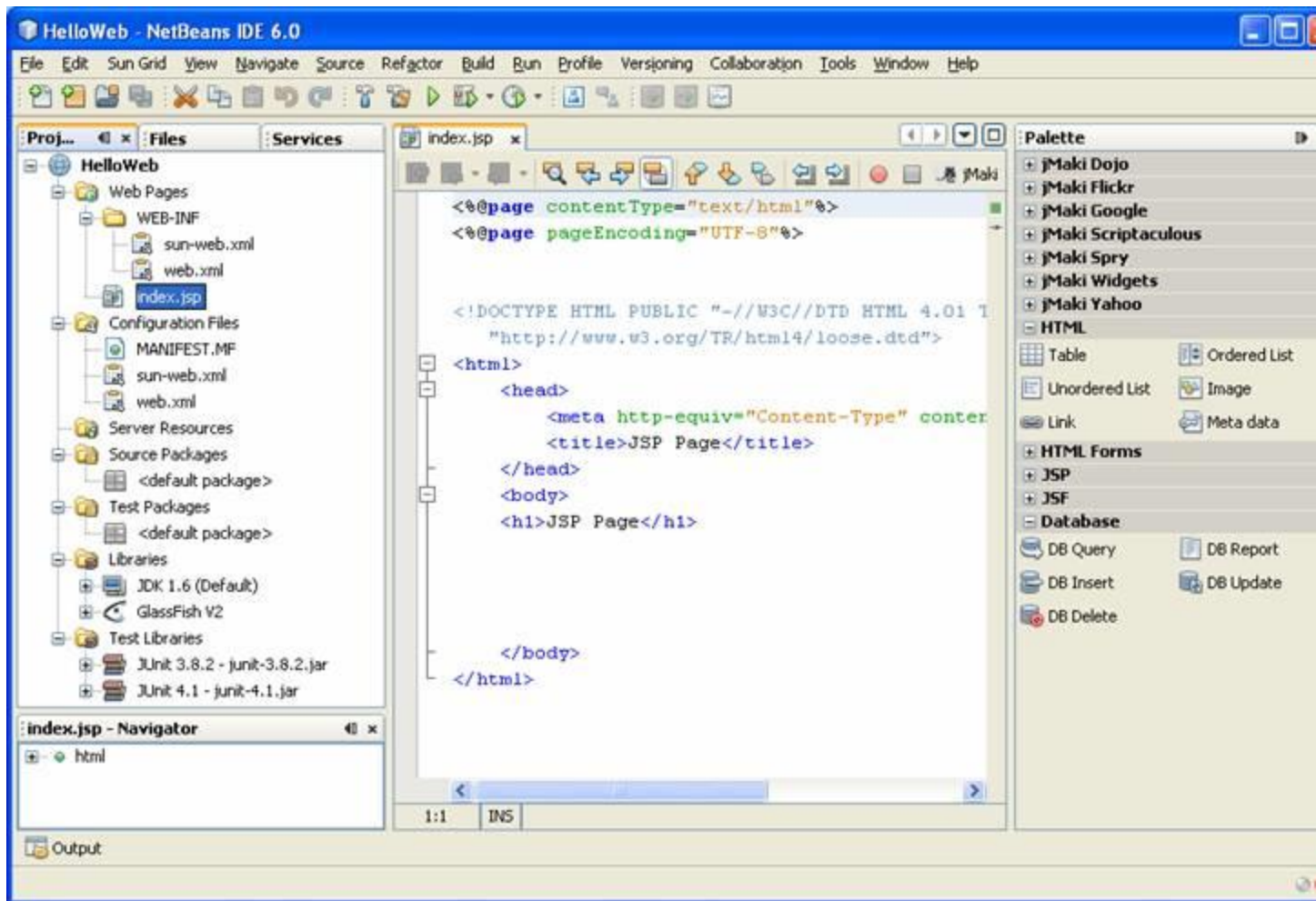
Java EE Version: Java EE 5

Context Path: /HelloWeb

☒ Set as Main Project

< Back   Next >   Finish   Cancel   Help

5. Leave the Set as Main Project option selected and click Finish. The IDE creates the `$PROJECTHOME/HelloWeb` project folder. 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. You can view the project's file structure in the Files window (Ctrl-2), and its logical structure in the Projects window (Ctrl-1):

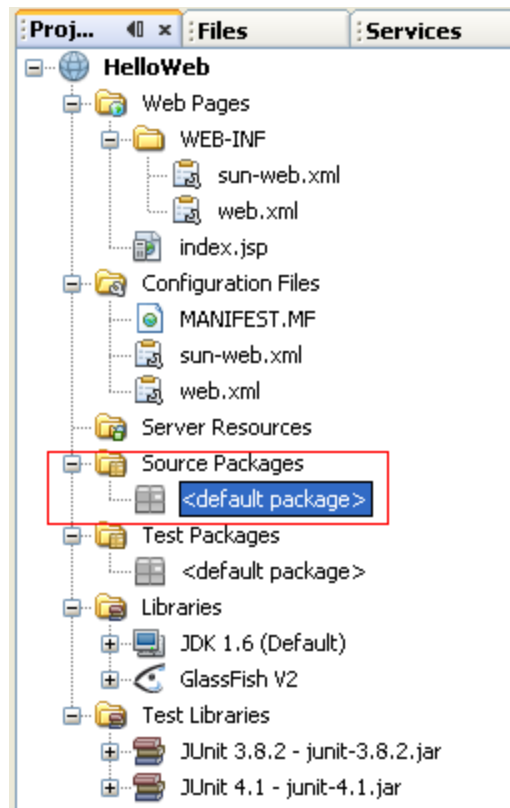


## Creating and Editing Web Application Source Files

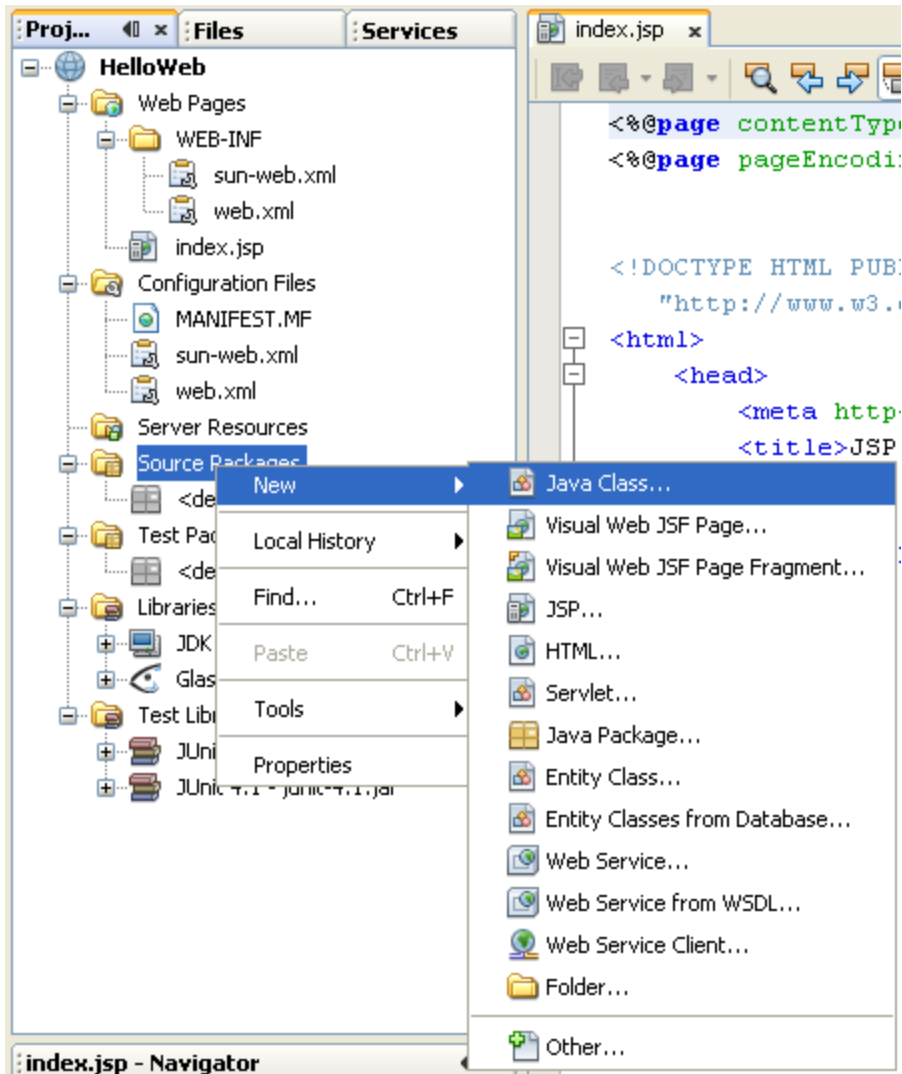
Creating and editing source files is the most important function that the IDE serves. After all, that is probably what you spend most of your day doing. The IDE provides a wide range of tools that can compliment any developer's personal style, whether you prefer to code everything by hand or want the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

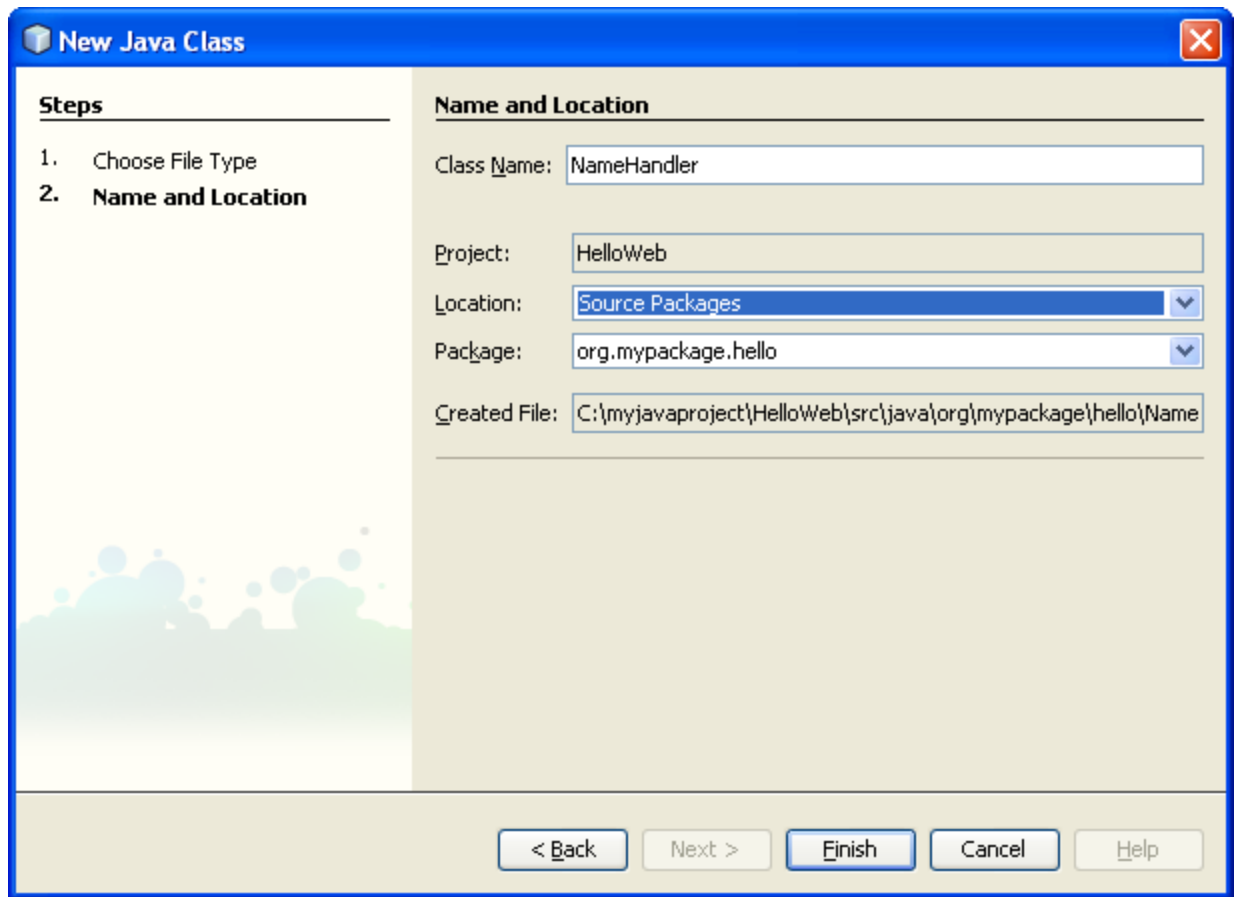
### Creating a Java Package and a Java Source File

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:

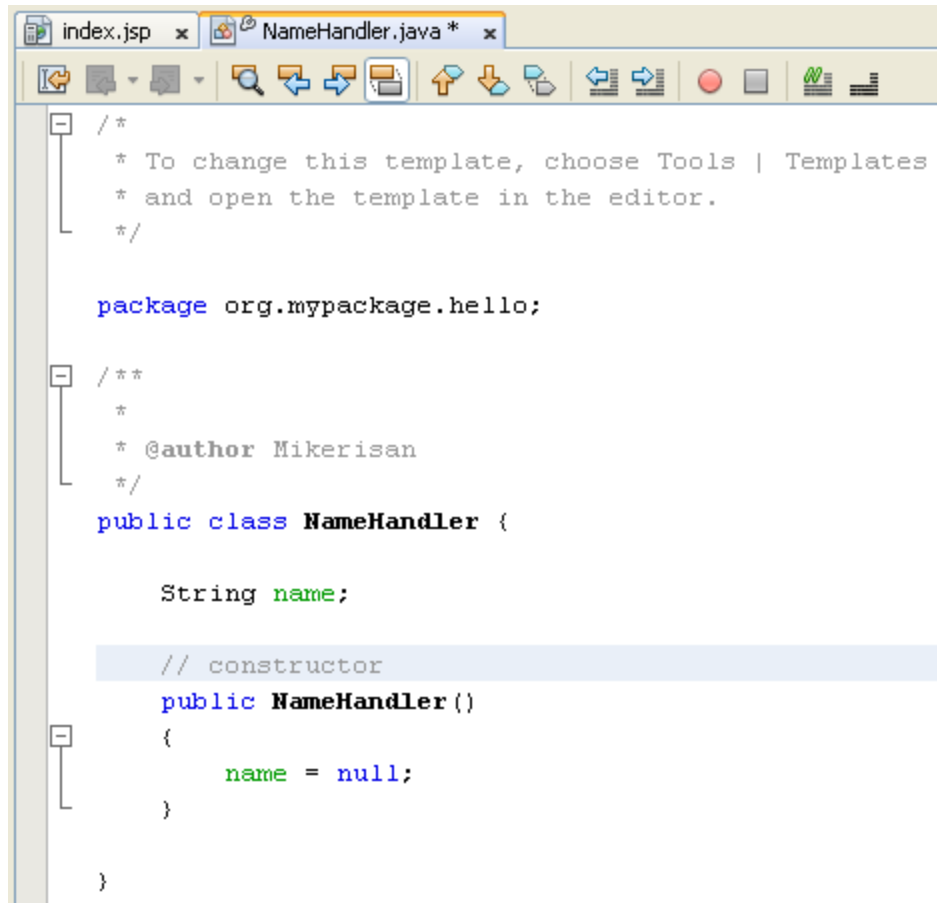
```
String name;
```

4. Add the following constructor to the class:

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

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

```
name = null;
```



```
index.jsp x NameHandler.java * x
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package org.mypackage.hello;

/**
 *
 * @author Mikerisan
 */
public class NameHandler {

    String name;

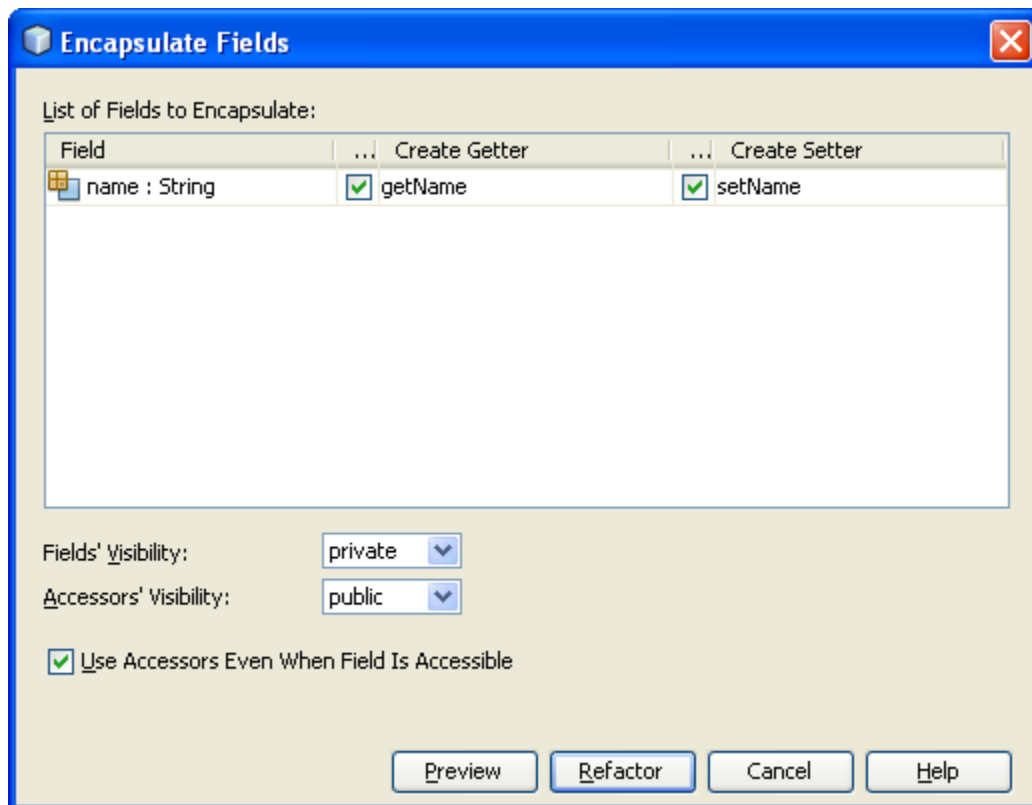
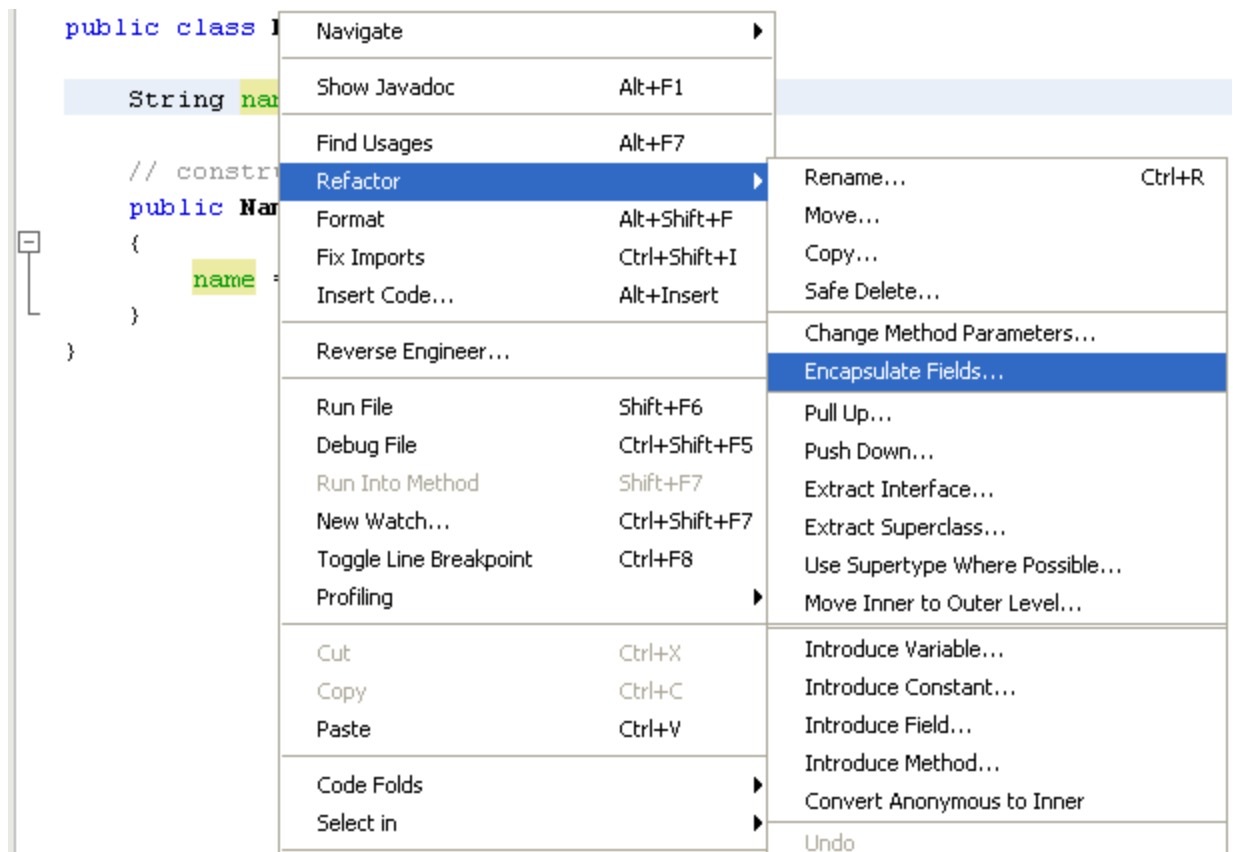
    // constructor
    public NameHandler()
    {
        name = null;
    }

}
```

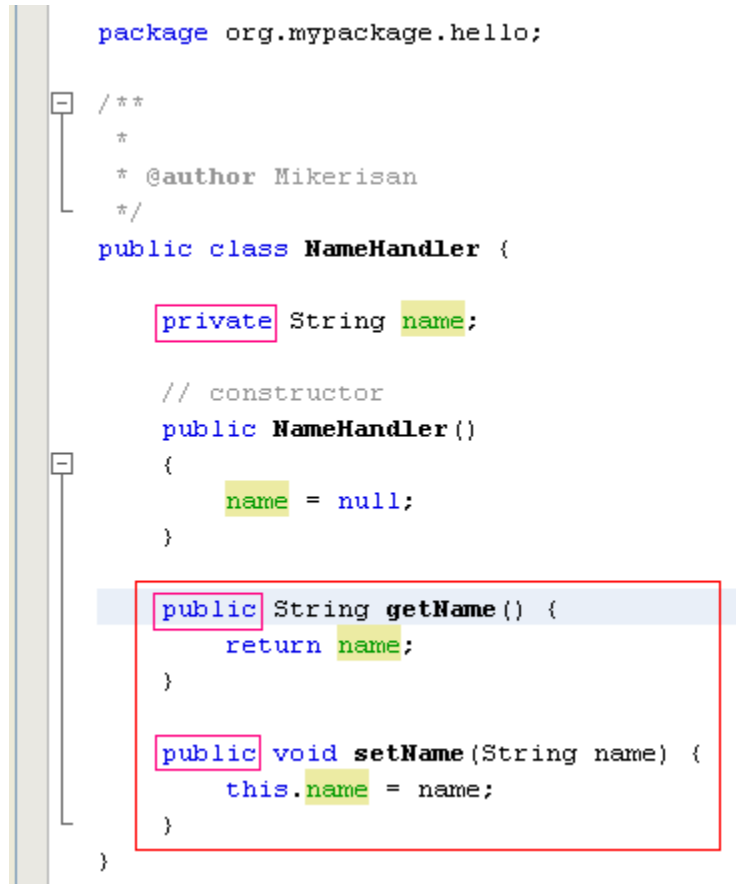
## Generating Getter and Setter Methods

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.





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 Mikerisan
 */
public class NameHandler {

    private String name;

    // constructor
    public NameHandler()
    {
        name = null;
    }

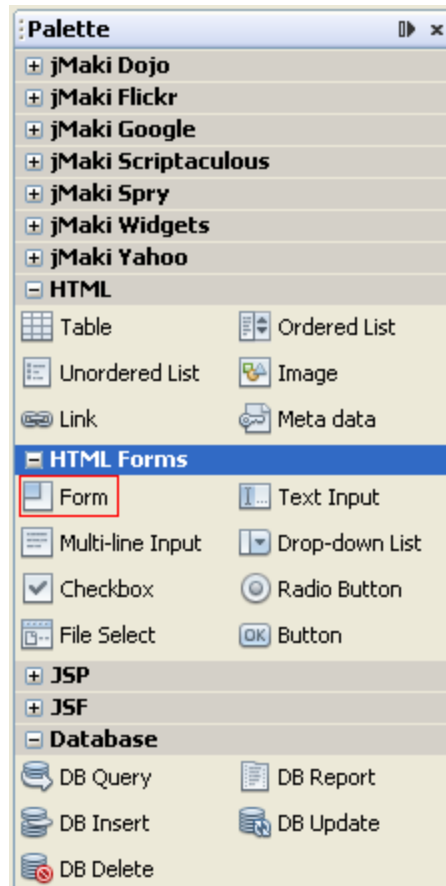
    public String getName() {
        return name;
    }

    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 into the Source Editor. The Insert Form dialog box displays:

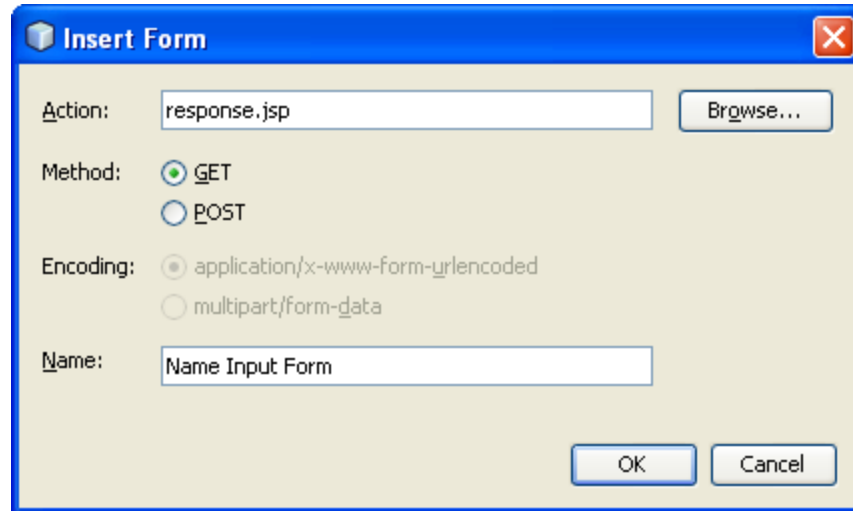


The image shows an 'Insert Form' dialog box with the following fields and options:

- Action:
- Method: ☒ GET, ☐ POST
- Encoding: ☒ application/x-www-form-urlencoded, ☐ multipart/form-data
- Name:
- Buttons: Browse..., OK, Cancel

3. Specify the following values:

- Action: `response.jsp`
- Method: `GET`
- Name: `Name Input Form`

A dialog box titled "Insert Form" with a blue border and a close button in the top right corner. It contains several fields and options: "Action:" with a text box containing "response.jsp" and a "Browse..." button; "Method:" with two radio buttons, "GET" (selected) and "POST"; "Encoding:" with two radio buttons, "application/x-www-form-urlencoded" (selected) and "multipart/form-data"; and "Name:" with a text box containing "Name Input Form". At the bottom right are "OK" and "Cancel" buttons.

Insert Form

Action: response.jsp Browse...

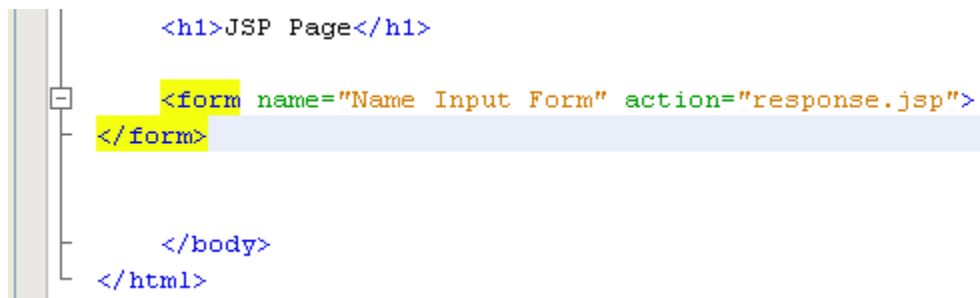
Method: ☒ GET ☐ POST

Encoding: ☒ application/x-www-form-urlencoded ☐ multipart/form-data

Name: Name Input Form

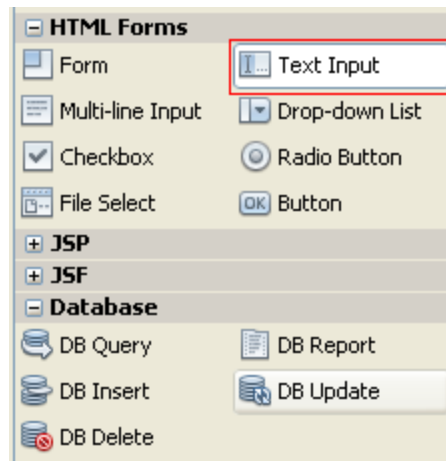
OK Cancel

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

A snippet of HTML code from a file named index.jsp. The code is displayed in a light blue background with a vertical scrollbar on the left. The code includes an opening <h1> tag, a <form> tag with attributes name="Name Input Form" and action="response.jsp", a closing </form> tag, a </body> tag, and a </html> tag. The <form> and </form> tags are highlighted in yellow.

```
<h1>JSP Page</h1>
<form name="Name Input Form" action="response.jsp">
</form>
</body>
</html>
```

- 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.



**Insert Text Input**

Name:

Initial Value:

Type: ☒ text  
☐ password  
☐ hidden

Initial State: ☐ disabled  
☐ readonly

Width:

OK Cancel

**Insert Text Input**

Name:

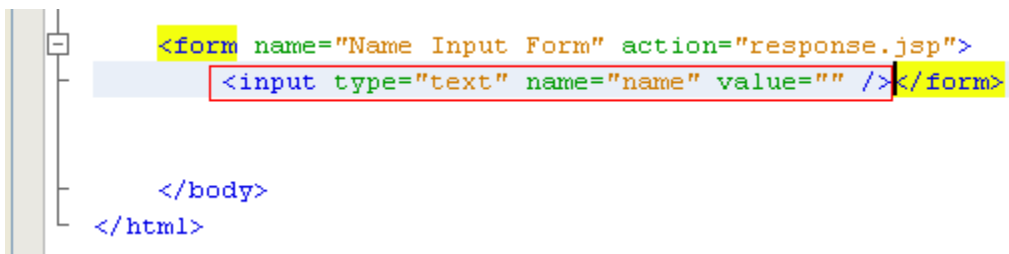
Initial Value:

Type: ☒ text  
☐ password  
☐ hidden

Initial State: ☐ disabled  
☐ readonly

Width:

OK Cancel



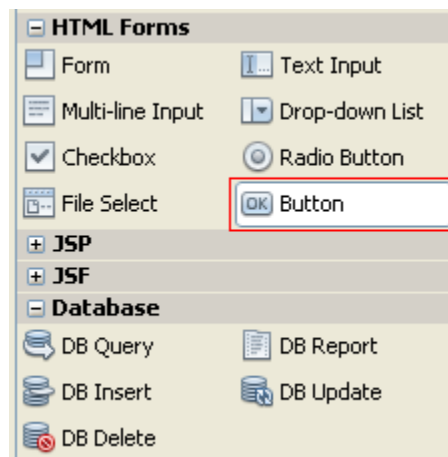
7. Delete the empty value= " " attribute. The code should be like this:

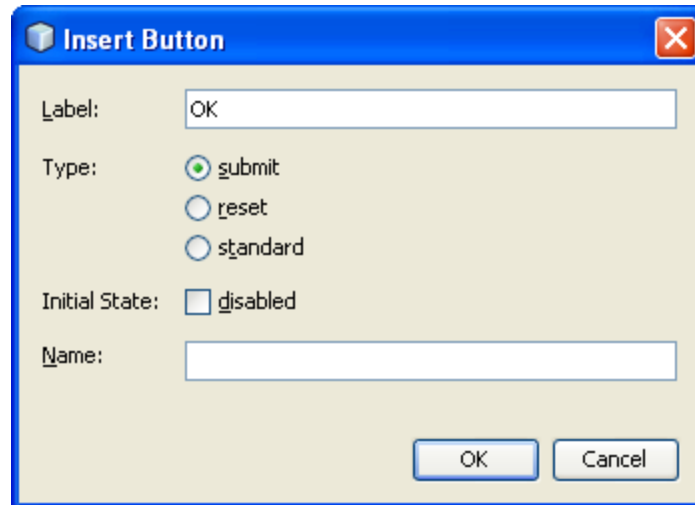
```
<input type="text" name="name" />
```

8. Drag a Button item to a point just before the `</form>` tag. Specify the following values:

- Label: `OK`
- Type: `submit`

9. Click OK. An HTML button is added between the `<form>` tags.





```

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

</body>
html>

```

10. Delete the empty value= " " attribute. Type Enter your name: just before the `<input>` tag, then change the default JSP Page text between the `<h1>` tags to Entry Form.

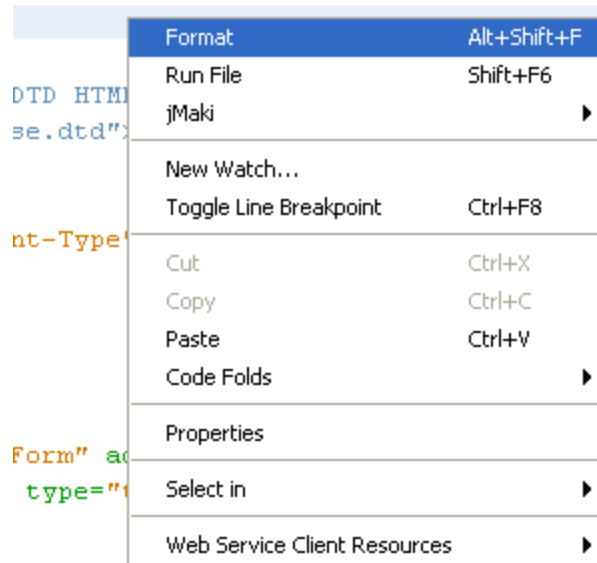
```

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; cha
    <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" />

```

11. 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:



```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<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>
```



```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

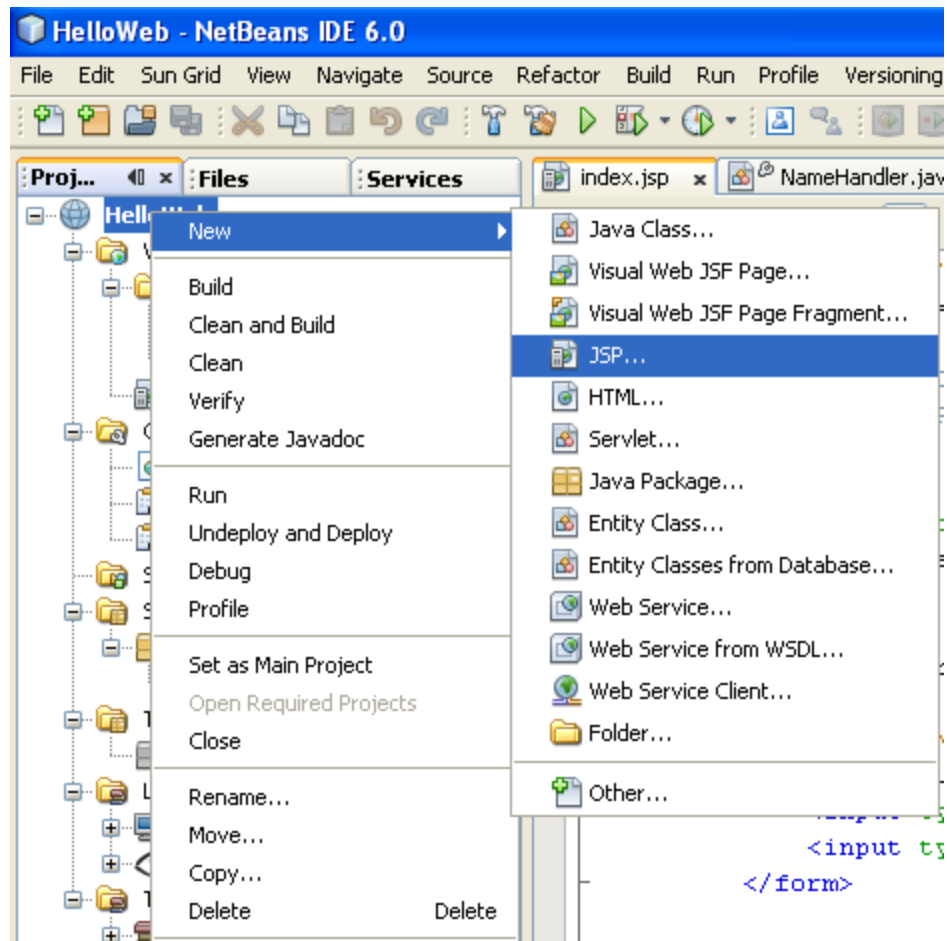
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <h1>Entry Form</h1>

    <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 (JSP) 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.



New JSP File

Steps

1. Choose File Type

2. **Name and Location**

JSP File Name:

response

Project:

HelloWeb

Location:

Web Pages

Folder:

Browse...

Created File:

C:\myjavaproject\HelloWeb\web\response.jsp

Opti...

☒ JSP File (Standard Syntax)

☐ JSP Document (XML Syntax)

☐ Create as a JSP Segment

Description:

A JSP file using JSP standard syntax.

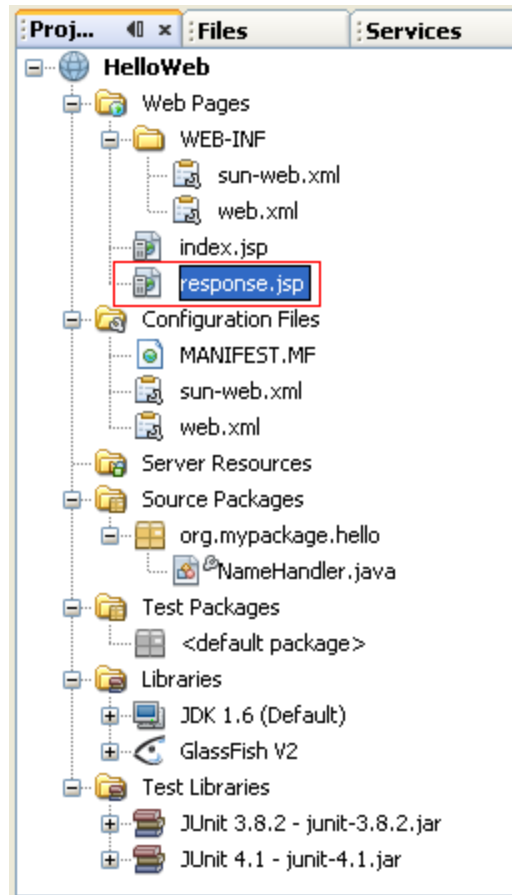
< Back

Next >

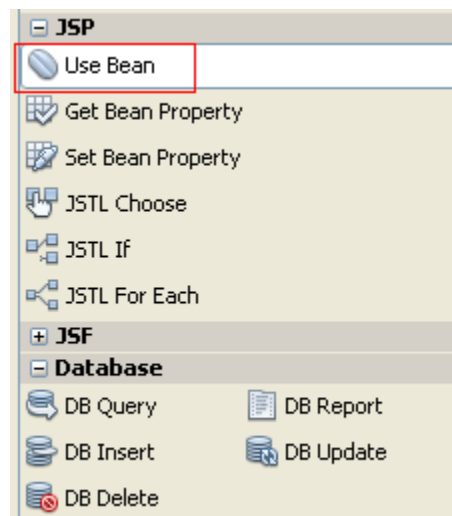
Finish

Cancel

Help

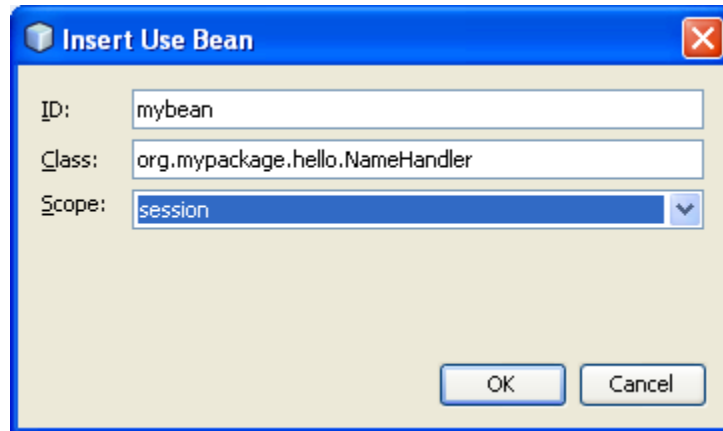


2. In the Palette to the right of the Source Editor, expand JSP and drag a Use Bean item to a point just below the <body> tag in the Source Editor.



3. The Insert Use Bean dialog opens. Specify the following values:
  - ID: `mybean`
  - Class: `org.mypackage.hello.NameHandler`

- Scope: `session`



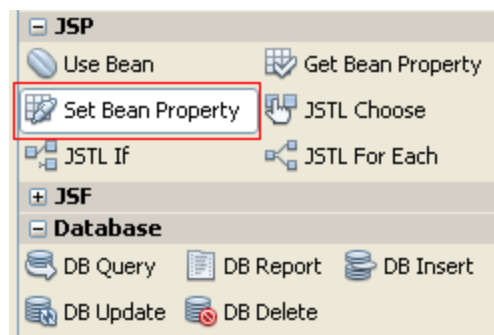
4. Click OK. Notice that `<jsp:useBean>` tag is added beneath the `<body>` tag.

```

</head>
<body>
  <jsp:useBean id="mybean" scope="session" class="org.mypackage.hello.NameHandler" />
  <h1>JSP Page</h1>

```

5. Drag a Set Bean Property item from the Palette to a point just before the `<h1>` tag and click OK.



```

<body>
  <jsp:useBean id="mybean" scope="session" class="org.mypackage.hello.NameHandler" />
  <jsp:setProperty name="request" property="" value="" />
  <h1>JSP Page</h1>

```

6. In the `<jsp:setProperty>` tag that appears, delete the empty value (`value=""`) attribute and edit as follows:

```

<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.

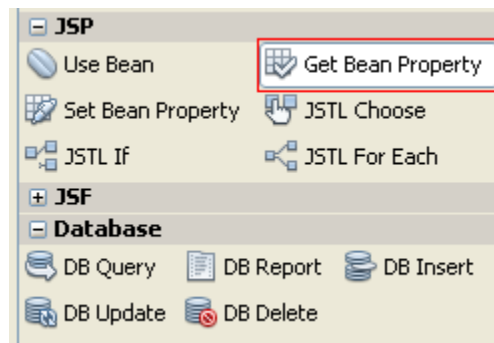
7. Change the text between the `<h2>` tags so that it looks like this:

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

and delete the `value=""` attribute.

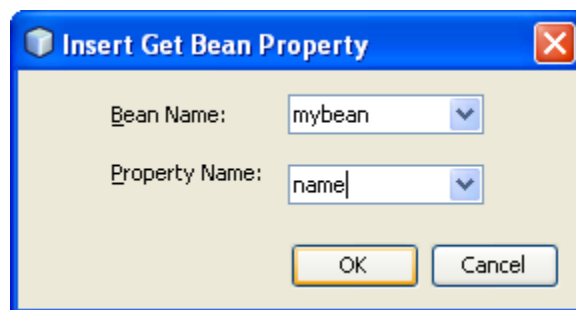
```
<body>
  <jsp:useBean id="mybean" scope="session" class="org.mypackage.hello.N
  <jsp:setProperty name="mybean" property="name" />
  <h1>Hello, !</h1>
```

8. Drag a Get Bean Property item from the Palette and drop it after the comma between the `<h1>` tags.

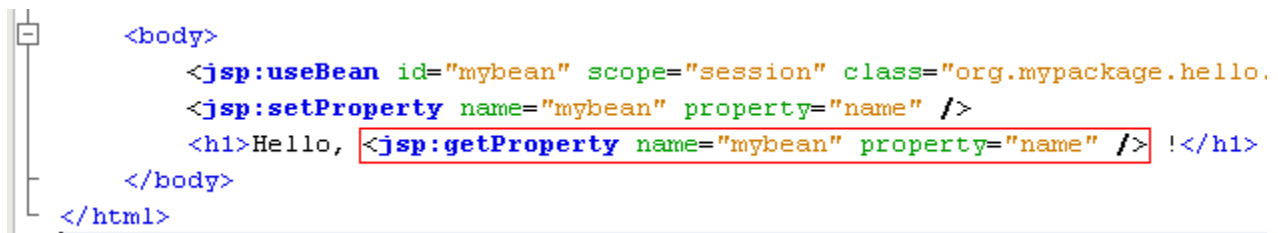


9. Specify the following values in the Insert Get Bean Property dialog:

- Bean Name: `mybean`
- Property Name: `name`



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



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

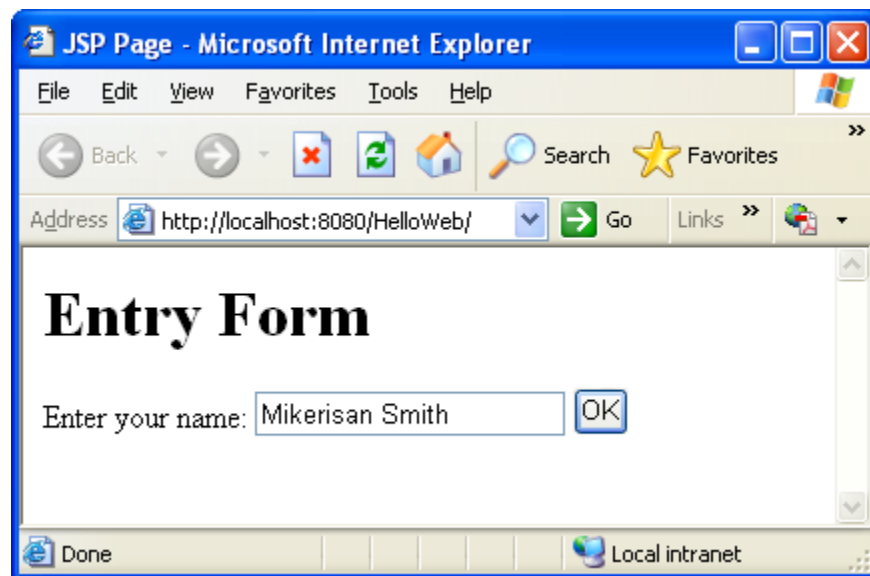
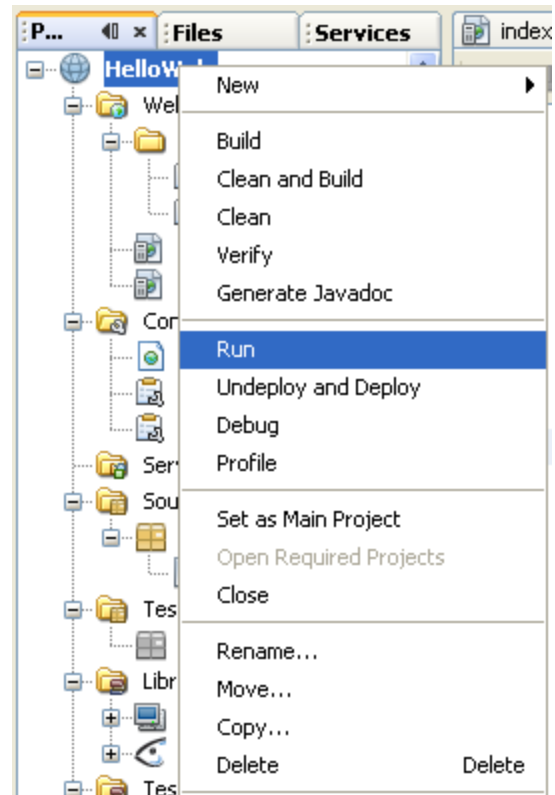
11. Right-click within the Source Editor and choose Format (Alt-Shift-F) to tidy the format of your code. The <body> tags of your response.jsp file should now appear similar to the following:

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <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>
</html>
```

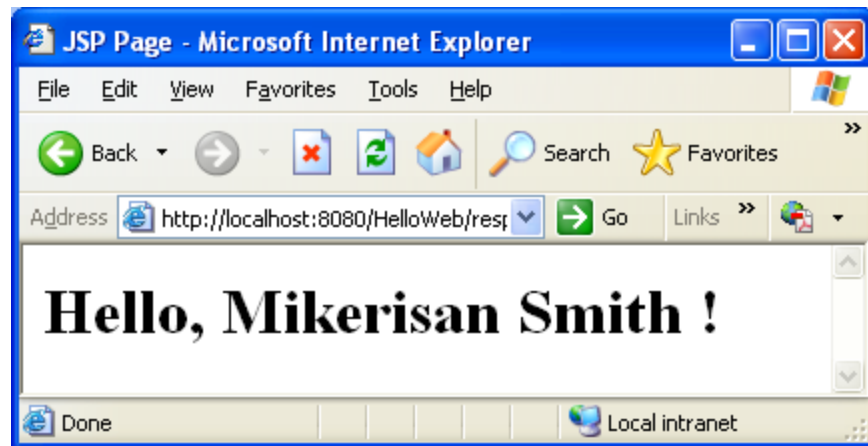
## Building and Running a Web Application Project

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). The IDE builds the web application and deploys it to the server you specified when creating the project. The index.jsp page opens in your default browser:







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.