# CREATE A JAVABEAN

A JavaBean is a class file that stores Java code for a JSP page. Although you can use a scriptlet to place Java code directly into a JSP page, it is considered better programming practice to store the code in a JavaBean.

JavaBeans offer several advantages. Using a JavaBean allows you to keep the Java code for a JSP page separate from the HTML code for the page, which can help keep the code from becoming long and difficult to work with. In addition, one JavaBean can be used with multiple JSP pages, which saves you from having to retype the same Java code in several pages.

JavaBeans also enable specialization when developing a Web site. For example, experts in Web page design can work with the HTML content of a JSP page, while programmers develop the Java code that will make the page dynamic. This allows both types of professionals to concentrate on their own areas of expertise.

You create a JavaBean as you would create any class file. JavaBeans can contain one or more methods. After creating the Java source code for a JavaBean, you must compile the code.

Once the JavaBean code has been compiled, you must copy the JavaBean to the appropriate directory on your Web server. On the Tomcat Web server, JavaBeans are usually stored in a directory named classes. Consult the documentation for your Web server to determine which directory you should use to store your JavaBeans.

You can organize your JavaBeans by storing them in packages. A package is a set of related class files stored together in a directory. To create a package for your JavaBeans, create a subdirectory within the directory that stores JavaBeans on your Web server and then store your JavaBeans in that subdirectory. Packages you create for JavaBeans are similar to packages that store class files in the Java SDK. For information about creating packages in the Java SDK, see page 50.

## Extra

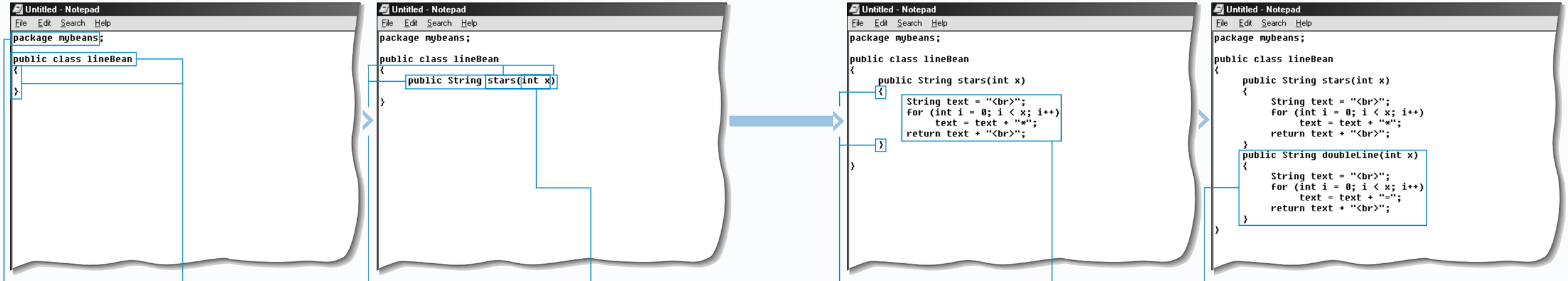
The JavaBeans you create for your JSP pages are platform independent. This means that a JavaBean created on a computer with a UNIX-based operating system, such as Linux, can be used on a computer running a different operating system, such as Windows.

There are many JavaBeans available on the Internet that you can use with your JSP pages. If you need a JavaBean to perform a specific task but do not want to create the JavaBean yourself, you may be able to purchase a ready-made JavaBean or have a programmer create a custom JavaBean according to your specifications.

JavaBeans were originally developed as a way for Java programmers to easily create and share re-usable portions of code for Java programs. If you want to create JavaBeans for use with your own stand-alone Java applications, you should obtain the Beans Development Kit (BDK). The BDK is available for download free of charge at the java.sun.com/products/javabeans Web site.

Integrated Development Environments (IDEs) are specialized programs used to create Java applications. Most Java IDEs can also be used to create JSP pages and JavaBeans. If you intend to create many JavaBeans for use with your JSP pages, you should consider purchasing an IDE.

## CREATE A JAVABEAN




**1** If you want to store the JavaBean in a package, type **package** followed by the name of the package directory you created on the Web server.

**2** To declare a class in the JavaBean, type **public class** followed by a name for the class.

**3** Type the opening and closing braces that will contain the body of the class.

**4** To declare a method, type **public** followed by the return type of the method.

**5** Type the name of the method followed by ( ).

**6** Between the parentheses, type any arguments the method requires.

**7** Type the opening and closing braces that will contain the body of the method.

**8** Type the code for the task you want the method to perform.

**9** Repeat steps 4 to 8 for each method you want the JavaBean to contain.

**10** Save the file with the .java extension and then compile the source code for the file. For information about compiling Java code, see page 20.

**11** Copy the compiled class file to the appropriate directory on your Web server.

■ You can now set up a JSP page to use the JavaBean.

# SET UP A JSP PAGE TO USE A JAVABEAN

Once you have created a JavaBean, you can use the `<jsp:useBean>` tag to set up a JSP page to access the JavaBean.

The `<jsp:useBean>` tag associates the JSP page with a specific JavaBean. This tag has several attributes that you must use in order to ensure that the correct JavaBean is used and that the JSP page can access the JavaBean. The `class` attribute allows you to specify the full class name of the JavaBean you want the JSP page to use. If the JavaBean is stored in a package, the full name will consist of the name of the package and the name of the class, separated by a dot.

The `id` attribute allows you to specify a case-sensitive name that identifies the JavaBean instance. If an instance of the JavaBean already exists, the `id` attribute identifies the instance. If the JavaBean does not already exist, a new instance is created.

The `scope` attribute allows you to specify when the JavaBean instance will be available. If you want the JavaBean instance to be available to a client only on the page in which the instance is created, you can specify the `page` value. This is the default value of the `scope` attribute. The `request` value allows you to make the JavaBean instance accessible to a client during a single request.

The `session` and `application` values of the `scope` attribute allow you to make a JavaBean available within any JSP page in the application. The `session` attribute makes the JavaBean available to a single client for the duration of the session, while the `application` attribute makes the JavaBean available to multiple clients for the duration of the application.

## Extra

When the JSP engine encounters the `<jsp:useBean>` tag, it uses the information specified in the tag to locate or create an instance of the JavaBean. If an instance already exists, the `<jsp:useBean>` tag is not processed. To ensure that code is processed only if a new instance of a JavaBean is created, you can place the code between an opening `<jsp:useBean>` tag and a closing `</jsp:useBean>` tag. For example, using the `<jsp:setProperty>` tag between the `<jsp:useBean>` tags allows you to initialize properties only when a new instance of the JavaBean is created.

**Example:**

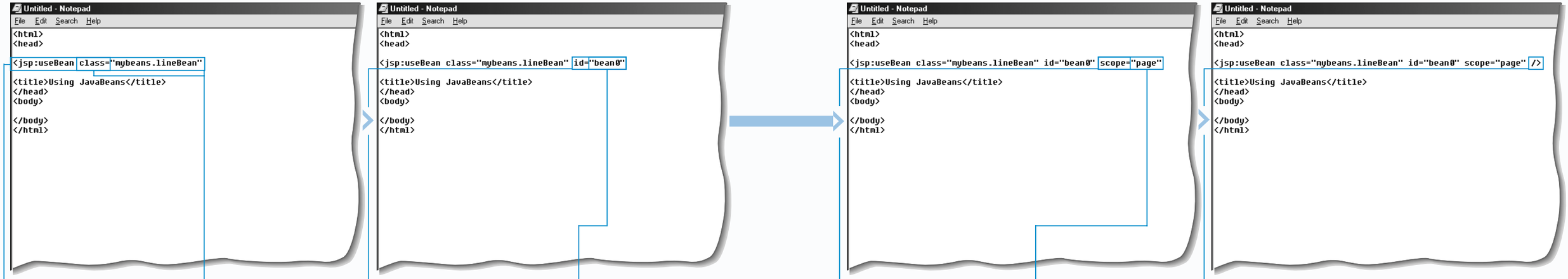
```
<jsp:useBean class="mybeans.lineBean" id="lineBeanId" scope="session">
     <jsp:setProperty name="lineBeanId" property="counter" value="0" />
</jsp:useBean>
```

When creating an instance of a JavaBean, the `<jsp:useBean>` tag may insert an additional method into the JavaBean instance that does not exist in the original JavaBean. This method, called a constructor, is a special method that has the same name as the class and is processed when the JavaBean instance is first created. Constructors are often used to initialize values. It is good programming practice to include a constructor in your JavaBean code, even if you do not plan to use the constructor. If a JavaBean does not include a constructor, the `<jsp:useBean>` tag will automatically create an empty constructor for you.

**Example:**

```
public void lineBean
{
}
```

## SET UP A JSP PAGE TO USE A JAVABEAN




1. To locate or create an instance of the JavaBean, type **<jsp:useBean**.
2. Type **class=**.
3. Type the full class name of the JavaBean, enclosed in quotation marks.
4. Type **id=**.
5. Type a name that will identify the JavaBean instance, enclosed in quotation marks.
6. Type **scope=**.
7. Type the scope for the JavaBean instance, enclosed in quotation marks.
8. Type **/>** to close the tag.
9. Save the page with the .jsp extension.

■ You can now use the JSP page to access a JavaBean.

# ACCESS A JAVABEAN METHOD

Once you have created a JavaBean, you can add the code that will access the methods in the JavaBean to your JSP page.

Before accessing a JavaBean method, you must add the `<jsp:useBean>` tag to the JSP page. This tag and its attributes ensure that the correct JavaBean will be used and that the JSP page has access to the JavaBean. For more information about setting up a JSP page to use a JavaBean, see page 124.

To access a method in a JavaBean, you create a scriptlet that includes the name of the JavaBean, followed by the name of the method you want to call. The name of the JavaBean is specified in the `id` attribute of the `<jsp:useBean>` tag. The name is case-sensitive and must be typed in the code that calls the method exactly as it was typed in the tag.

You can also include any arguments that the method may require in the scriptlet. If a method returns a result that can be displayed on a JSP page, you can use an expression to display the returned value.

Prior to accessing a method in a JavaBean, you should make sure that the JavaBean source code you created has been compiled and that the Web server has access to the JavaBean class files. JavaBeans are usually stored in a directory called classes on the server. For example, when using a Tomcat Web server, you must save the class files in the classes subdirectory of the main Tomcat directory. If the classes subdirectory does not exist, you should create the subdirectory. To determine which directory you should use to store your JavaBeans, consult the documentation for your Web server.

## Apply It

One of the features of JavaBeans is its ability to access methods that share the same name but accept different data types as arguments, referred to as method overloading. For example, you can create two methods with the same name, but one method may use an `int` data type as an argument and the other method may use a `String` data type.

**To create a JavaBean, type:**

```
public String stars(int x)
{
      String text = "<br>";
      for (int i = 0; i < x; i++)
            text = text + "*";
      return text + "<br>";
}
public String stars(String x)
{
      return "<br>* * " + x +  " * *<br>";
}
```

**To use the methods declared in the JavaBean, type:**

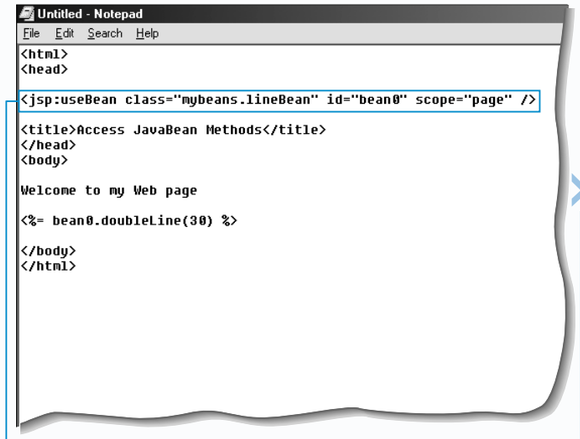
```
<%= lineBeanId.stars(30) %>
<%= lineBeanId.stars("Welcome") %>
```

**Result:**

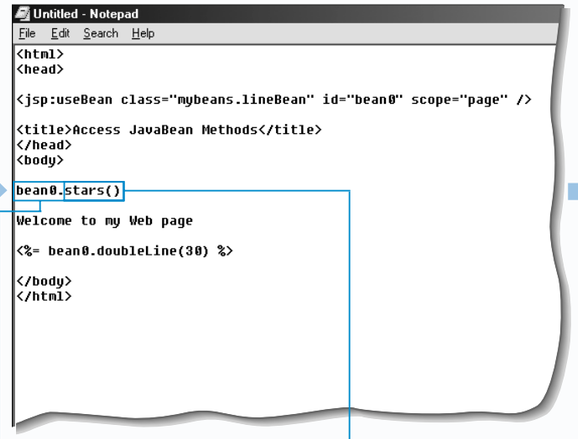
```
******************************

* * Welcome * *
```

## ACCESS A JAVABEAN METHOD



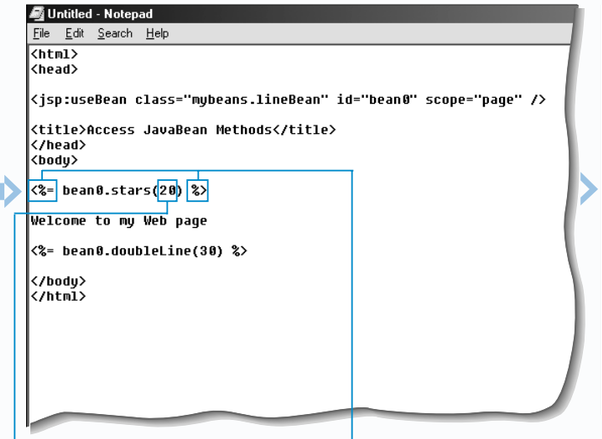

1 Type the code that sets up the JSP page to use a JavaBean.




2 To access a method in the JavaBean, type the name of the JavaBean followed by a dot.

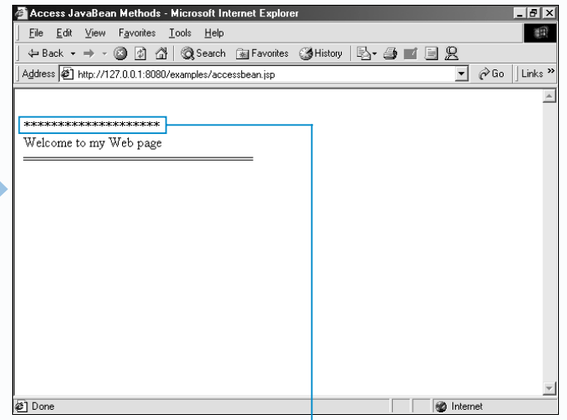
- The name of the JavaBean must be the same as the value assigned to the `id` attribute of the `<jsp:useBean>` tag in step 1.

3 Type the name of the method you want to access, followed by a set of parentheses.




4 If necessary, specify any arguments required by the method within the parentheses.

5 Type the code that processes the method call in the JSP page.




6 Save the page with the .jsp extension and then display the JSP page in a Web browser.

- The Web browser displays the results of accessing a JavaBean method.

# CREATE A JAVABEAN PROPERTY

You can create a property to store information about a JavaBean. Properties are fields that define the behavior of a JavaBean.

After creating the code for a JavaBean, you can specify a property you want the JavaBean to contain. To do so, you specify the data type of the value the property will store and a name for the property. You can then create a method that assigns the property an initial value.

JavaBean properties are private or protected fields, which means they cannot be directly accessed by a JSP page. In order to access the value of a property, you must create a special method, called a getter method. A getter method, also referred to as an accessor method, returns the value of a property.

There are specific rules that must be followed when declaring a getter method. The access modifier of the method must be set to `public` and the data type of the value to be returned must be specified. The name of the method is the same as the name of the property, but begins with a capital letter and is prefixed by the word `get`. For example, if you want to return the value of the loginTime property, you would create a getter method called getLoginTime. The name of a getter method is followed by parentheses.

The body of a getter method includes a `return` statement that specifies the name of the property whose value you want to return.

When you finish creating the JavaBean source code, you must compile the code and store the resulting class file in the appropriate directory on your Web server. If you are using the Tomcat Web server, the class file should be saved in the classes directory located in the main Tomcat directory. If the JavaBean is part of a package, the class file would be stored in the appropriate package directory within the classes directory.

## Extra

A getter method can be used to return various types of data, including a boolean value of true or false. When declaring a getter method that returns a boolean value, the method name can be prefixed by `is` instead of `get`. This can help make your code easier to read and understand. If you choose to use `is` when returning a boolean value, you should do so consistently throughout your code.

**For example:**

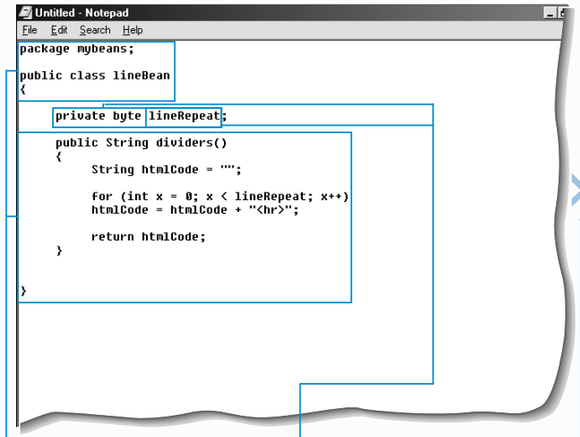
```
public boolean getJobStatus()
     {
          return jobStatus;
     }
```

**Can be typed as:**

```
public boolean isJobStatus()
     {
          return jobStatus;
     }
```

You should declare the method that assigns a property an initial value in the body of the class. If you declare the method elsewhere, such as in another method, the getter method may not be able to access the value.

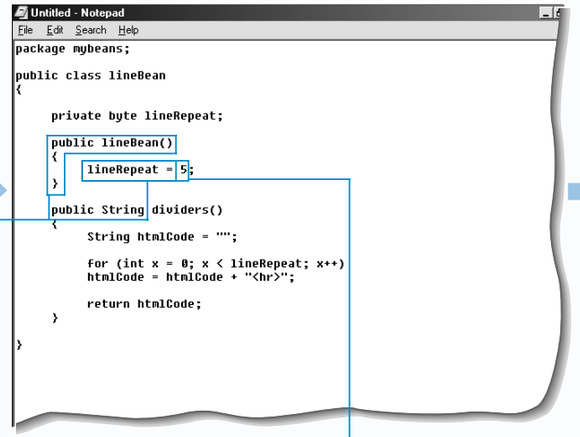
## CREATE A JAVABEAN PROPERTY




1 Type the code that creates a JavaBean. For information about creating a JavaBean, see page 122.

2 To create a property for the JavaBean, type **private** followed by the data type of the value the property will store.

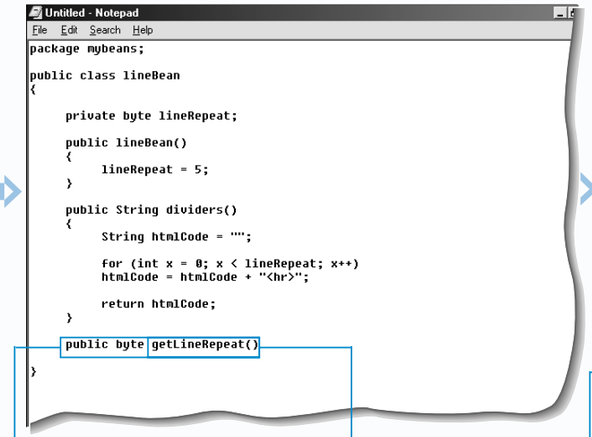
3 Type a name for the property.




4 Declare a method to assign the property an initial value.

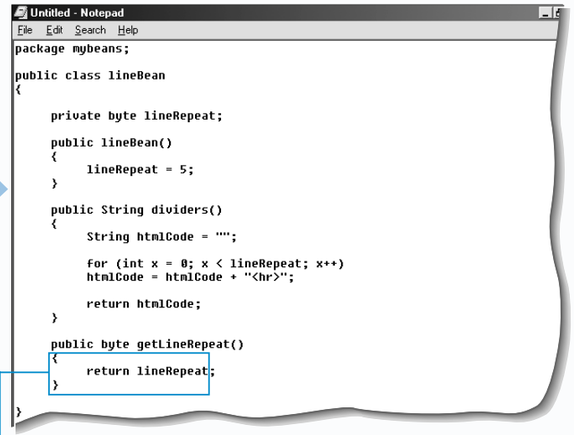
5 In the body of the method, type the name of the property followed by **=**.

6 Type an initial value for the property.




7 To declare a getter method that will return the value of the property, type **public** followed by the data type of the value.

8 To name the method, type **get** immediately followed by the name of the property, beginning with a capital letter. Then type **()**.




9 Type **return** followed by the name of the property. Enclose the code in braces.

10 Save the file with the .java extension and then compile the source code for the file.

11 Copy the compiled class file to the appropriate directory on your Web server.

■ You can now access the JavaBean property in a JSP page.

# ACCESS A JAVABEAN PROPERTY

The <jsp:getProperty> tag can be used in a JSP page to access a property of a JavaBean. When the <jsp:getProperty> tag retrieves the value of a property, the tag converts the value to a string and then inserts the value into the JSP page output.

Before accessing a JavaBean property, you must add the <jsp:useBean> tag to the JSP page. This tag and its attributes ensure that the correct JavaBean is used and that the JSP page has access to the JavaBean.

The <jsp:getProperty> tag must be embedded in the HTML code of a JSP page. You cannot place the <jsp:getProperty> tag in Java code that generates HTML code. The tag must also be placed in an area of a page that can be displayed in a Web browser, such as within the <body> tag.

The <jsp:getProperty> tag has two required attributes. The name attribute allows you to specify the name of the JavaBean that contains the property you want to access. The value you assign to the name attribute must be the same as the value you assigned to the id attribute of the <jsp:useBean> tag.

You use the property attribute to specify the name of the property you want to access. You should be careful to use the correct uppercase and lowercase letters when typing the name of the property you want to access. If you do not enter the name of the property correctly, the <jsp:getProperty> tag will not be able to locate the property in the JavaBean and will return an error message.

## Extra

JavaServer Pages code is processed by the Web server and then sent to the Web browser as HTML code. People can view the HTML source code for a JSP page, but they will not be able to view the JavaServer Pages code. This means that users viewing the source code for a JSP page that contains the <jsp:getProperty> tag will not see the actual tag. Instead, they will see the value returned by the tag.

**The source code viewed in a Web browser:**

javabeans - Notepad

```
<html>
<head>
<title>Using JavaBeans</title>
</head>
<body>

Welcome to my Web site.

<hr><hr><hr><hr><hr>
<br>
The default number of horizontal lines that will be displayed using this
JavaBean is:
5

</body>
</html>
```

**The actual code of the JSP page:**

javabeans - Notepad

```
<html>
<head>
<title>Using JavaBeans</title>
</head>
<body>

<jsp:useBean id="lineBeanId" scope="session" class="mybeans.lineBean" />

Welcome to my Web site.

<%= lineBeanId.dividers() %>
<br>
The default number of horizontal lines that will be displayed using this
JavaBean is:
<jsp:getProperty name="lineBeanId" property="lineRepeat" />

</body>
</html>
```

## ACCESS A JAVABEAN PROPERTY

Untitled - Notepad

```
<html>
<head>
<title>Using JavaBeans</title>
</head>
<body>

<jsp:useBean id="lineBeanId" scope="session" class="mybeans.lineBean" />

Welcome to my Web site. <br>

The default number of horizontal lines that will be displayed using this
JavaBean is:
<jsp:getProperty

</body>
</html>
```

**1** Type the code that sets up the JSP page to use a JavaBean. For information about setting up a JSP page, see page 124.

**2** To access a property in the JavaBean, type **<jsp:getProperty**.

Untitled - Notepad

```
<html>
<head>
<title>Using JavaBeans</title>
</head>
<body>

<jsp:useBean id="lineBeanId" scope="session" class="mybeans.lineBean" />

Welcome to my Web site. <br>

The default number of horizontal lines that will be displayed using this
JavaBean is:
<jsp:getProperty name="lineBeanId"

</body>
</html>
```

**3** Type **name=** followed by the name of the JavaBean that contains the property you want to access, enclosed in quotation marks.

■ The value you assign to the name attribute must be the same as the value you assigned to the id attribute of the <jsp:useBean> tag in step 1.

Untitled - Notepad

```
<html>
<head>
<title>Using JavaBeans</title>
</head>
<body>

<jsp:useBean id="lineBeanId" scope="session" class="mybeans.lineBean" />

Welcome to my Web site. <br>

The default number of horizontal lines that will be displayed using this
JavaBean is:
<jsp:getProperty name="lineBeanId" property="lineRepeat" />

</body>
</html>
```

**4** Type **property=** followed by the name of the property you want to access, enclosed in quotation marks.

**5** Type **/>** to close the tag.

Using JavaBeans - Microsoft Internet Explorer

Welcome to my Web site.
The default number of horizontal lines that will be displayed using this JavaBean is: 5

**6** Save the page with the .jsp extension and then display the JSP page in a Web browser.

■ The Web browser displays the result of accessing a JavaBean property.

# SET A JAVABEAN PROPERTY

A JSP page can be used to change the initial value of a JavaBean property. Changing the value of a property will affect how the JavaBean works.

To change the value of a JavaBean property, you must declare a setter method in the code for the JavaBean. Setter methods are often referred to as accessor methods. A setter method can work with a getter method. When used together, they allow a JSP page to write and read the value of a JavaBean property. For more information about getter methods, see page 128.

The access modifier of a setter method must be set to `public` and the return type set to `void`, since the method does not return a value. The name of the method is the same as the name of the property to be changed, but begins with a capital letter and is prefixed by the word `set`. The parentheses at the end of the setter method name enclose the data type of the property and a variable to store the value passed by a JSP page.

The `<jsp:useBean>` tag must be included in the JSP page you want to use to access a JavaBean. For information about this tag, see page 124. The `<jsp:setProperty>` tag and its attributes can then be used to change the value of a JavaBean property.

The `name` attribute allows you to specify the name of the JavaBean that contains the property you want to set. The value you assign to the `name` attribute must be the same as the value you assigned to the `id` attribute of the `<jsp:useBean>` tag. The `property` attribute specifies the name of the property whose value you want to set. The `value` attribute specifies the value for the property.

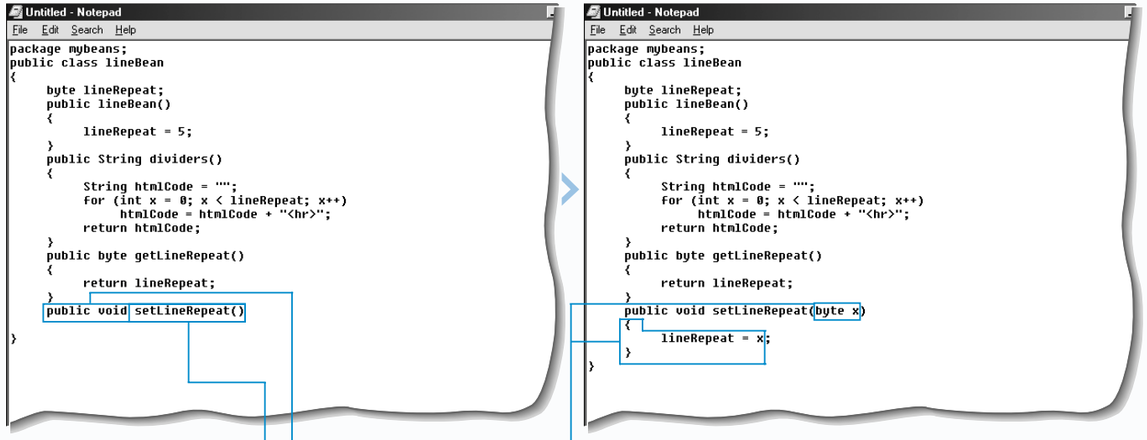
## Extra

Before you can change or retrieve a JavaBean property, the JSP engine must examine the JavaBean to determine if the property exists. A JSP engine is Web server software that processes JSP pages. When a JSP page attempts to access a JavaBean property, the JSP engine uses a process called *introspection* to examine the JavaBean. If the JSP engine finds a getter method, setter method or both for the property, then the property exits and can be accessed. A property without an accessor method does not exist. This is why the naming conventions for accessor methods are so rigid.

When the `<jsp:getProperty>` tag retrieves the value of a property, the value is converted to a string and then inserted into the JSP page output. If the value retrieved is a string, then no data conversion is required. Otherwise, the JSP engine will automatically convert the data using the appropriate class of the `java.lang` package.

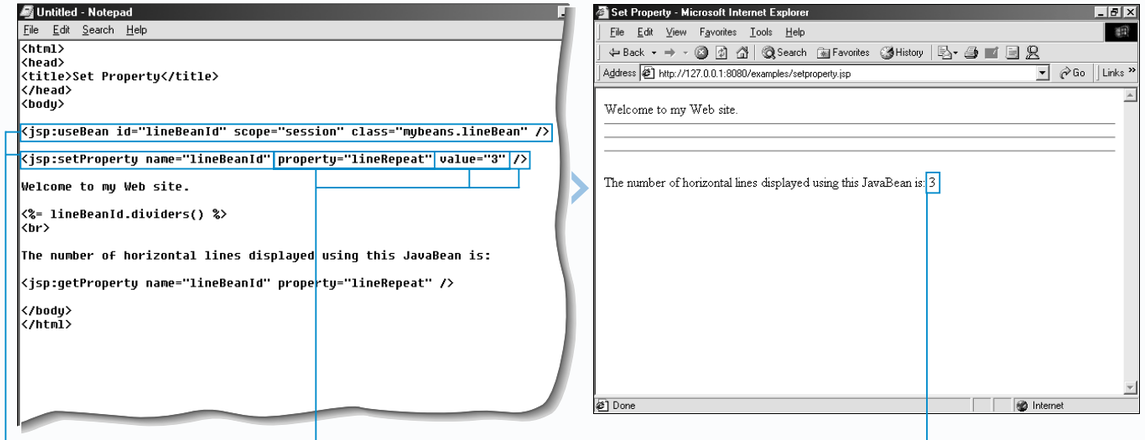
| VALUE DATA TYPE: | DATA CONVERSION CLASS: |
|---|---|
| boolean | `java.lang.Boolean.valueOf(string)` |
| byte | `java.lang.Byte.valueOf(string)` |
| char | `java.lang.Character.valueOf(string)` |
| double | `java.lang.Double.valueOf(string)` |
| int | `java.lang.Integer.valueOf(string)` |
| float | `java.lang.Float.valueOf(string)` |
| long | `java.lang.Long.valueOf(string)` |

## SET A JAVABEAN PROPERTY




### DECLARE A SETTER METHOD

1. Type the code that creates a JavaBean and include any properties you want the JavaBean to contain.
2. To declare a setter method that will change the value of a property, type **public void**.
3. To name the method, type **set** immediately followed by the name of the property you want to change, beginning with a capital letter. Then type **()**.
4. Between the parentheses, type the data type of the property followed by a variable.
5. Type the name of the property followed by **=** and the variable. Enclose the code in braces.
6. Save the file with the .java extension and then compile the source code for the file.
7. Copy the compiled class file to the appropriate directory on your Web server.




### USE A JSP PAGE TO SET A JAVABEAN PROPERTY

1. Type the code that sets up the JSP page to use a JavaBean.
2. Type **<jsp:setProperty name=** followed by the name of the JavaBean that contains the property you want to set, enclosed in quotation marks.
3. Type **property=** followed by the name of the property you want to set, enclosed in quotation marks.
4. Type **value=** followed by a value for the property, enclosed in quotation marks.
5. Type **/>** to close the tag.
6. Save the page with the .jsp extension and then display the JSP page in a Web browser.
7. The Web browser displays the result of setting a JavaBean property.

# CREATE AN INDEXED JAVABEAN PROPERTY

You can create an indexed property to store a collection of related information about a JavaBean. To create an indexed property, you first specify an array of values the indexed property will store. For information about creating arrays, see page 48.

In order to allow a JSP page to access the values of an indexed property, you must declare a getter method in the JavaBean. The access modifier of the getter method must be set to `public` and the data type of the return value must be specified. The name of the method is the same as the name of the indexed property, but begins with a capital letter and is prefixed by the word `get`. The body of a getter method includes a `return` statement that returns the values stored in the indexed property.

To allow a JSP page to change the values stored in an indexed property, you must declare a setter method in the JavaBean.

The access modifier of a setter method must be set to `public` and the return value set to `void`. The name of the method is the same as the name of the indexed property but begins with a capital letter and is prefixed by the word `set`. The parentheses at the end of the setter method name must contain two arguments. The first argument represents the index of the element that is to be changed in an indexed property, while the second argument represents the new value to be placed in the element.

When you finish creating the JavaBean source code, you must compile the code and store the resulting class file in the appropriate directory on your Web server. If the JavaBean is part of a package, the class file must be stored in the appropriate package directory within the class file directory.

## Extra

In a JavaBean, you may want to declare a getter method that allows the JSP page to retrieve the value of a single element of an indexed property. You can use this getter method to display the value of a single element from an indexed property instead of having to display all the values of an indexed property at one time. You can have more than one getter method with the same name in a JavaBean.

**Example:**

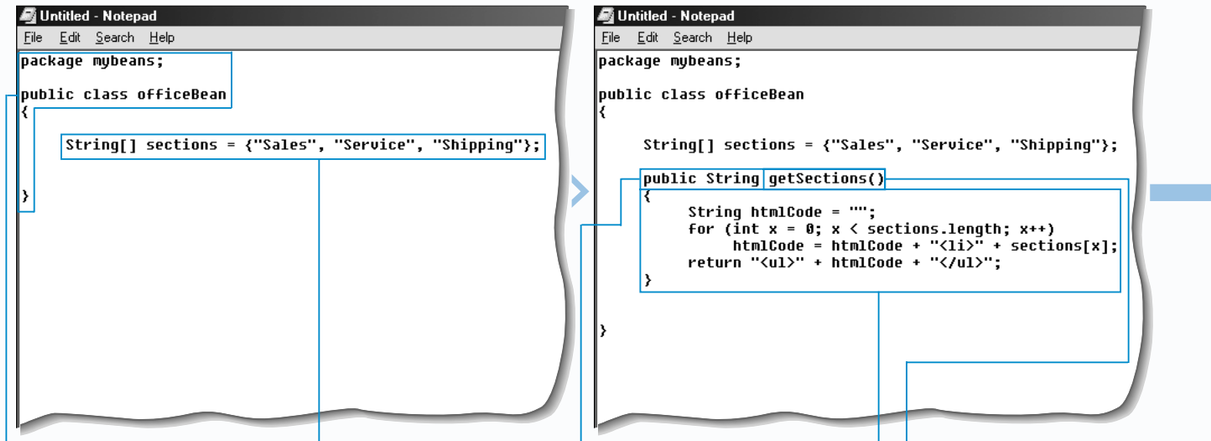
```
public String getSections(int x)
{
      return sections[x];
}
```

You can declare a setter method that allows the JSP page to change all the values stored in an indexed property in a JavaBean. Changing all the values stored in an indexed property at once is more efficient that using a setter method to change the value of each element of an indexed property individually.

**Example:**

```
public void setSections(String[] i)
{
      for (int x = 0; x < i.length; x++)
            sections[x] = i[x];
}
```

## WORK WITH AN INDEXED JAVABEAN PROPERTY




### CREATE AN INDEXED JAVABEAN PROPERTY

**1** Type the code that creates a JavaBean. For information about creating a JavaBean, see page 122.

**2** Type the code that creates an array to be used as an index property and specifies the values you want to store in the array. For information about creating an array, see page 48.

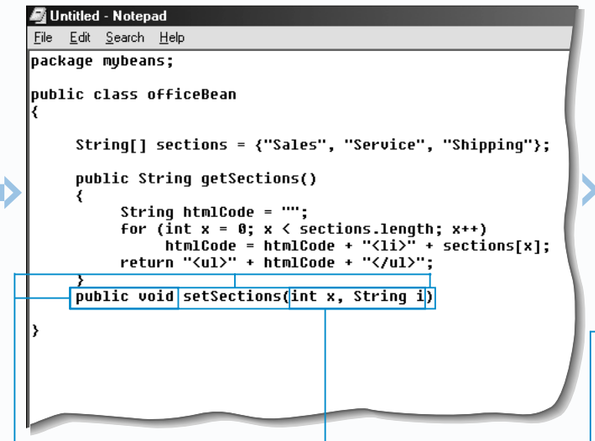



### DECLARE A GETTER METHOD

**3** Type **public** followed by the data type of the return value of the method.

**4** Type **get** immediately followed by the name of the method, beginning with a capital letter. Then type ().

**5** Type the code that will access and return the values from the indexed property.

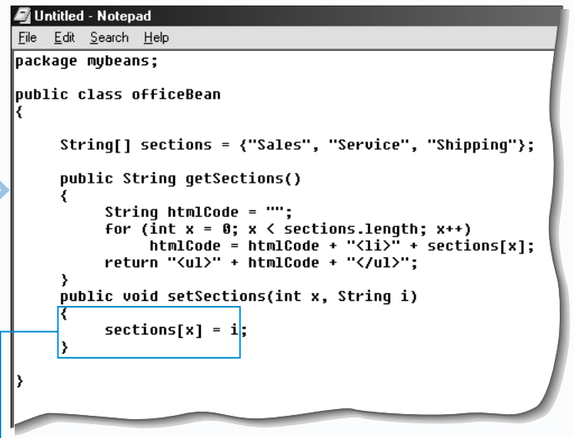



### DECLARE A SETTER METHOD

**6** Type **public void**.

**7** Type **set** immediately followed by the name of the indexed property, beginning with a capital letter. Then type ().

**8** Between the parentheses, type **int x**, followed by the data type of the indexed property and a variable name.




**9** Type the name of the indexed property followed by **[x] =**. Then type the variable and enclose the code in braces.

**10** Save the file with the .java extension and compile the source code. Then copy the compiled class file to the appropriate directory on your Web server.

■ You can now access the indexed JavaBean property in a JSP page.

# ACCESS AN INDEXED JAVABEAN PROPERTY

After creating a JavaBean that has an indexed property, you can access the values stored in the indexed property from your JSP page.

First, you must set up the JSP page to use a JavaBean. To set up a JSP page, you must add the `<jsp:useBean>` tag to the page. This tag and its attributes ensure that the correct JavaBean is used and that the JSP page has access to the JavaBean. See page 124 for more information about setting up a JSP page to use a JavaBean.

You should make sure that the name of the JavaBean you specify in the `<jsp:useBean>` tag is the JavaBean that contains the indexed property you want to access. A JavaBean must also contain the accessor methods that allow the JSP page to retrieve and alter the values of the indexed property.

To display the values of an indexed property, you add code that calls a getter method to your JSP page. Since a getter method usually returns a value, you can use an expression to display the returned value.

You can change the value of an element of an indexed property by calling a setter method declared in the JavaBean. A setter method is called from within a scriplet in your JSP page and usually does not return any value. To change the value of an element using the setter method, you must pass the index number of the element and the new value to the method as arguments.

After changing the value of an element of an indexed property, you may want to once again call the getter method to display the values. This is an easy way to confirm that the setter method is working properly.

## Extra

If you declared a getter method in your JavaBean that allows you to retrieve the value of a single element of an indexed property, you can call the method in your JSP page. When retrieving all the values of an indexed property, you usually do not need to pass any arguments to the getter method, however, if you want to retrieve the value of a single element, you must pass the index number of the element. Keep in mind that the first element of an indexed property has an index number of 0. This means that if you want to retrieve the third element of an indexed property, you must pass an index number of 2.

**Example:**

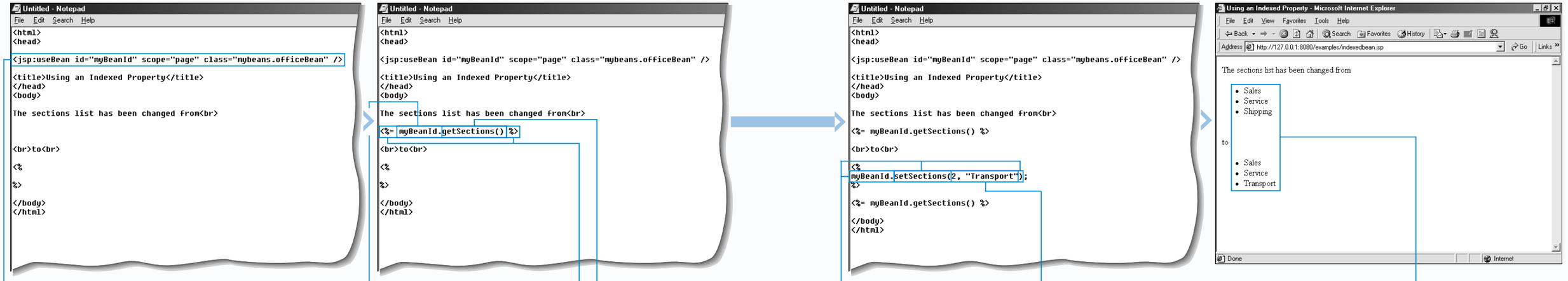
```
<%= myBeanId.getSections(2) %>
```

If you declared a setter method in your JavaBean that allows you to change all the values stored in an indexed property, you can call the method in your JSP page. When changing all the values of an indexed property, you pass an array containing all the new values to be stored in the indexed property as an argument.

**Example:**

```
<%
String[] t = {"Accounting", "Operations", "Transport"};
myBeanId.setSections(t);
%>
```

## ACCESS AN INDEXED JAVABEAN PROPERTY




**1** Type the code that sets up the JSP page to use a JavaBean. For information about setting up a JSP page to use a JavaBean, see page 124.

### CALL A GETTER METHOD

**2** Type the name of the JavaBean that contains the indexed property you want to access followed by a dot.

- The name of the JavaBean must be the same as the value assigned to the `id` attribute of the `<jsp:useBean>` tag in step 1.

**3** Type the name of the getter method followed by a set of parentheses.

**4** Type the code that uses the method call.

### CALL A SETTER METHOD

**5** Type the name of the JavaBean that contains the indexed property you want to access followed by a dot.

**6** Type the name of the setter method followed by a set of parentheses.

**7** Between the parentheses, type the index number of the element you want to change followed by a comma. Then type the new value you want to assign to the element.

**8** Save the page with the .jsp extension and then display the JSP page in a Web browser.

- The Web browser displays the results of using the getter and setter methods to access an indexed property in a JavaBean.

# PROCESS FORM DATA USING A JAVABEAN

JavaBeans can be used to help a JSP page process data submitted by a form. For example, a JavaBean can store the values submitted by a form and allow the JSP page to retrieve and manipulate the values. Using a JavaBean to manage the data submitted by a form can make the data easier to work with.

While simple forms can be processed entirely by scriptlets within a JSP page, complex forms that contain many elements are best handled using JavaBeans. Using JavaBeans helps make the code that processes a form easier to manage, maintain and modify.

A JavaBean that will manage data submitted by a form should contain a property for each element in the form. The properties will store the values from the form elements. The names of the JavaBean properties should match the names of the form elements. For example, a property that will store the value of a text box named username should be called username.

You must declare a setter method for each property you create. If you want the JSP page to be able to retrieve the values stored in the properties, you must also declare a getter method for each property. These methods are often referred to as accessor methods. A setter method allows you to assign a value from a form to a JavaBean property. A getter method returns the value of a property. The conventions you must use when declaring setter and getter methods are very strict. For information about declaring a setter method, see page 135. For information about declaring a getter method, see page 134.

Once you have created the JavaBean source code, you must compile the code and store the resulting class file in the appropriate directory on your Web server.

**Extra**

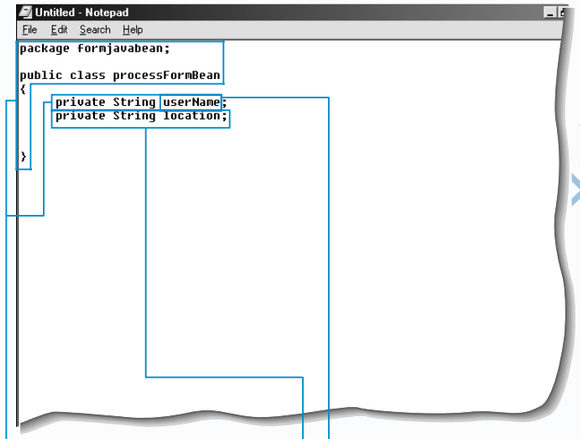
Data can also be passed to a JSP page by a query string. A query string is one or more name and value pairs appended to the URL of a page. To create a query string, you enter the URL of the JSP page in a Web browser, followed by a question mark. You then enter a name followed by an equal sign and a value for the name. To enter multiple name and value pairs, separate each pair with an ampersand (&). A query string should not exceed 2000 characters and should not contain spaces.

**Example:**

```
http://www.abccorp.com/processform.jsp?userName=Ernest&location=USA
```

You can create a JavaBean that will store data submitted by a query string just as you would create a JavaBean to store data from a form. The JavaBean should contain a property for each name and value pair that will be submitted. The property names should match the names submitted by the query string. For example, if the name and value pair userName=Ernest will be submitted by a query string, the JavaBean should contain a property called userName.

## CREATE A JAVABEAN TO STORE FORM DATA

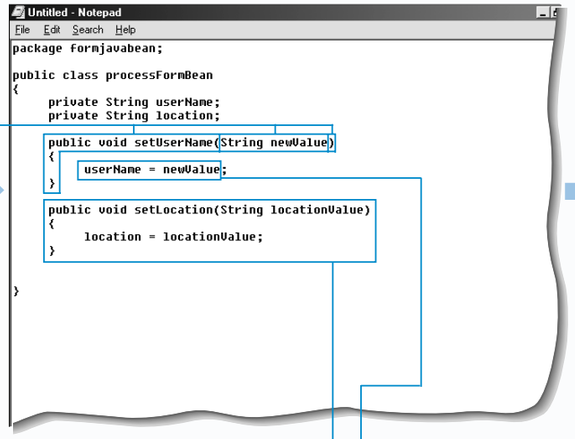



1 Type the code that creates a JavaBean.

2 To create a property that will store a value from a form element, type **private** followed by the data type of the value the property will store.

3 Type a name for the property. The name should match the name of the corresponding form element.

4 Repeat steps 2 and 3 for each property you want to create.

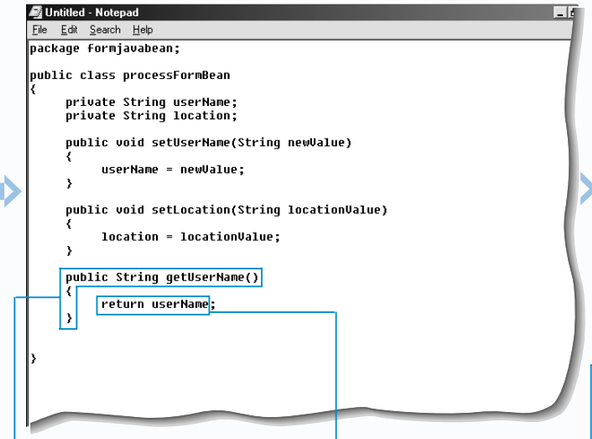



5 Type the code that declares a setter method. The setter method will assign a form value to the specified property.

6 Between the parentheses at the end of the setter method name, type the data type of the property followed by a variable.

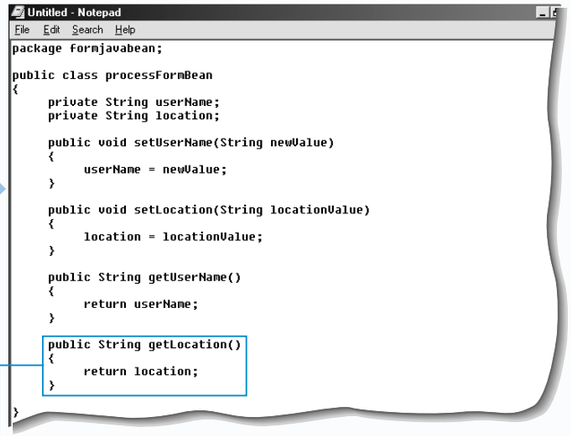
7 In the body of the setter method, type the name of the property followed by = and the variable.

8 Repeat steps 5 to 7 for each setter method you want to declare.




9 Type the code that declares a getter method. The getter method will return the value of the specified property.

10 In the body of the getter method, type **return** followed by the name of the property.




11 Repeat steps 9 and 10 for each getter method you want to declare.

12 Save the file with the .java extension and then compile the source code for the file.

13 Copy the compiled class file to the appropriate directory on your Web server.

■ You can now set up a JSP page to use the JavaBean when processing form data.

CONTINUED

# PROCESS FORM DATA USING A JAVABEAN

The JSP page you want to process form data using a JavaBean must be set up to use the JavaBean. The JSP page can simply pass form data to the JavaBean and then retrieve and display the data. A JSP page and JavaBean can also be used to perform a more complicated task, such as storing the data in a database.

The `<jsp:useBean>` tag allows you to associate the JSP page with the JavaBean you created to manage form data. This tag must appear in the JSP page before the `<jsp:setProperty>` tag, which is used to set JavaBean properties. The `property` attribute of the `<jsp:setProperty>` tag allows you to specify the properties you want to set. If the names of the properties match the names of the elements in the form, you can quickly set all the properties in the JavaBean using the `*` wildcard character.

When a form is submitted to the JSP page, the `<jsp:setProperty>` tag will pass the form values from the JSP page to the JavaBean, assigning the values to the appropriate properties. The values are assigned during the process of *introspection*, in which the JavaBean is examined and its properties detected.

The `<jsp:getProperty>` tag allows the JSP page to access a property of the JavaBean. This tag retrieves the value of a property and automatically inserts the value into the output of the JSP page. You must use a `<jsp:getProperty>` tag for each property you want to access.

In the code for the form the JSP page will process, the `action` attribute of the `<form>` tag must specify the correct filename and location of the JSP page. The `method` attribute can specify either `get` or `post`. The JSP page will be able to process the form regardless of the method used to pass information to the page.

## Extra

If the names of the properties in the JavaBean do not match the names of the elements in the form, you must set the value of each property individually. To set the value of a property, use the `property` attribute to specify the name of the property and the `param` attribute to specify the name of its corresponding form element. For example, the following line of code assigns the value of a form element named clientName to a property called userName.

**Example:**

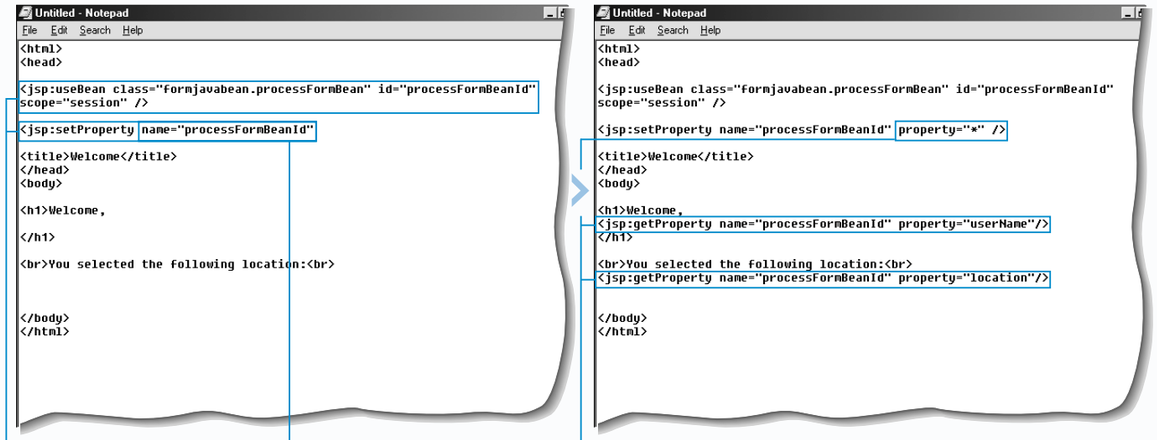
```
<jsp:setProperty name="processFormBean"
property="userName" param="clientName"/>
```

A form does not have to be submitted directly to the JSP page that will process the form data. The form can be submitted to a JSP page and then forwarded to another JSP page. For example, form data can be submitted to a JSP page that verifies data before being forwarded to the JSP page that will process the data. The `<jsp:forward>` tag can be used to pass form data from one JSP page to another.

**Example:**

```
<jsp:forward page="process.jsp"/>
```

## SET UP A JSP PAGE TO PROCESS FORM DATA




**1** Type the code that sets up the JSP page to use a JavaBean.

**2** To set the JavaBean properties with values from a form, type **<jsp:setProperty**.

**3** Type **name=** followed by the name of the JavaBean that contains the properties you want to set, enclosed in quotation marks.

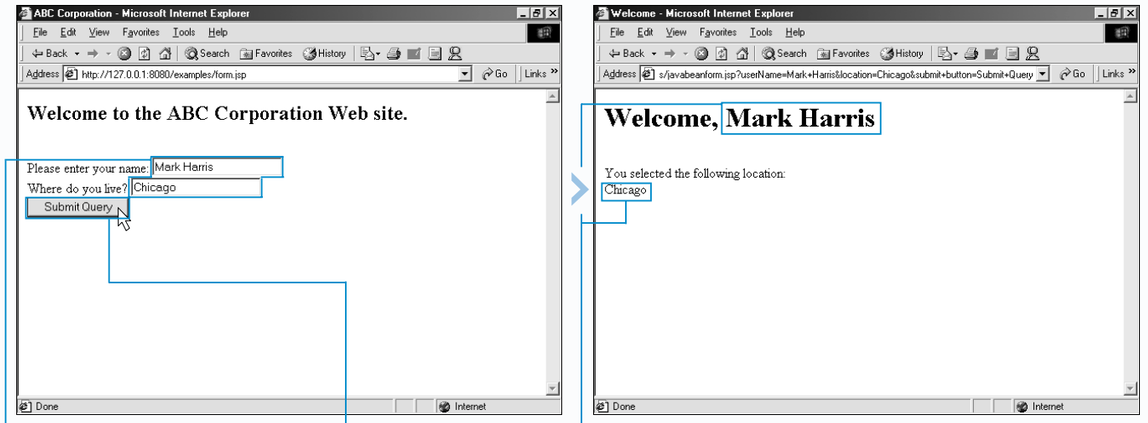
- The value you assign to the `name` attribute must be the same as the value you assigned to the `id` attribute of the `<jsp:useBean>` tag in step 1.

**4** Type **property="*" />**.

**5** Type the code that will access the JavaBean properties.

**6** Save the page with the .jsp extension.

## PROCESS FORM INFORMATION




**1** In a Web browser, display the Web page containing the form you want to process.

**2** Enter data into the form.

**3** Click the submit button to pass the data in the form to the JSP page.

- The Web browser displays the result of using a JavaBean to process data from the form.