14.3 Using Beans: Basic Tasks

You use three main constructs to build and manipulate JavaBeans components in JSP pages:

- **`jsp:useBean`.** In the simplest case, this element builds a new bean. It is normally used as follows:

  ```
  <jsp:useBean id="beanName"
               class="package.Class" />
  ```

  If you supply a `scope` attribute (see Section 14.6, "Sharing Beans"), the `jsp:useBean` element can either build a new bean or access a preexisting one.

- **`jsp:getProperty`.** This element reads and outputs the value of a bean property. Reading a property is a shorthand notation for calling a method of the form `get`*`Xxx`*. This element is used as follows:

  ```
  <jsp:getProperty name="beanName"
                   property="propertyName" />
  ```

- **`jsp:setProperty`.** This element modifies a bean property (i.e., calls a method of the form `set`*`Xxx`*). It is normally used as follows:

  ```
  <jsp:setProperty name="beanName"
                   property="propertyName"
                   value="propertyValue" />
  ```

The following subsections give details on these elements.

Building Beans: jsp:useBean

The `jsp:useBean` action lets you load a bean to be used in the JSP page. Beans provide a very useful capability because they let you exploit the reusability of Java classes without sacrificing the convenience that JSP adds over servlets alone.

The simplest syntax for specifying that a bean should be used is the following.

```
<jsp:useBean id="name" class="package.Class" />
```

This statement usually means "instantiate an object of the class specified by `Class`, and bind it to a variable in `_jspService` with the name specified by `id`." Note, however, that you use the fully qualified class name—the class name with packages included. This requirement holds true regardless of whether you use `<%@ page import... %>` to import packages.

Core Warning

You must use the fully qualified class name for the `class` attribute of `jsp:useBean`.

So, for example, the JSP action

```
<jsp:useBean id="book1" class="coreservlets.Book" />
```

can normally be thought of as equivalent to the scriptlet

```
<% coreservlets.Book book1 = new coreservlets.Book(); %>
```

Installing Bean Classes

The bean class definition should be placed in the same directories where servlets can be installed, *not* in the directory that contains the JSP file. Just remember to use packages (see Section 11.3 for details). Thus, the proper location for individual bean classes is `WEB-INF/classes/`*`subdirectoryMatchingPackageName`*, as discussed in Sections 2.10 (Deployment Directories for Default Web Application: Summary) and 2.11 (Web Applications: A Preview). JAR files containing bean classes should be placed in the `WEB-INF/lib` directory.

Core Approach

Place all your beans in packages. Install them in the normal Java code directories: `WEB-INF/classes/subdirectoryMatchingPackageName` for individual classes and `WEB-INF/lib` for JAR files.

Using jsp:useBean Options: scope, beanName, and type

Although it is convenient to think of `jsp:useBean` as being equivalent to building an object and binding it to a local variable, `jsp:useBean` has additional options that make it more powerful. As we'll see in Section 14.6, you can specify a `scope` attribute that associates the bean with more than just the current page. If beans can be shared, it is useful to obtain references to existing beans, rather than always building a new object. So, the `jsp:useBean` action specifies that a new object is instantiated only if there is no existing one with the same `id` and `scope`.

Rather than using the `class` attribute, you are permitted to use `beanName` instead. The difference is that `beanName` can refer either to a class or to a file containing a serialized bean object. The value of the `beanName` attribute is passed to the `instantiate` method of `java.beans.Bean`.

In most cases, you want the local variable to have the same type as the object being created. In a few cases, however, you might want the variable to be declared to have a type that is a superclass of the actual bean type or is an interface that the bean implements. Use the `type` attribute to control this declaration, as in the following example.

```
<jsp:useBean id="thread1" class="mypackage.MyClass"
                          type="java.lang.Runnable" />
```

This use results in code similar to the following being inserted into the `_jspService` method.

```
java.lang.Runnable thread1 = new myPackage.MyClass();
```

A `ClassCastException` results if the actual class is not compatible with `type`. Also, you can use `type` without `class` if the bean already exists and you merely want to access an existing object, not create a new object. This is useful when you share beans by using the `scope` attribute as discussed in Section 14.6.

Note that since `jsp:useBean` uses XML syntax, the format differs in three ways from HTML syntax: the attribute names are case sensitive, either single or double quotes can be used (but one or the other *must* be used), and the end of the tag is marked with `/>`, not just `>`. The first two syntactic differences apply to all JSP elements that look like `jsp:`*`xxx`*. The third difference applies unless the element is a container with a separate start and end tag.

A few character sequences also require special handling in order to appear inside attribute values. To get `'` within an attribute value, use `\'`. Similarly, to get `"`, use `\"`; to get `\`, use `\\`; to get `%>`, use `%\>`; and to get `<%`, use `<\%`.

Accessing Bean Properties: jsp:getProperty

Once you have a bean, you can output its properties with `jsp:getProperty`, which takes a `name` attribute that should match the `id` given in `jsp:useBean` and a `property` attribute that names the property of interest.

Core Note

With `jsp:useBean`, the bean name is given by the `id` attribute. With `jsp:getProperty` and `jsp:setProperty`, it is given by the `name` attribute.

Instead of using `jsp:getProperty`, you could use a JSP expression and explicitly call a method on the object with the variable name specified by the `id` attribute. For example, assuming that the `Book` class has a `String` property called `title` and that you've created an instance called `book1` by using the `jsp:useBean` example given earlier in this section, you could insert the value of the `title` property into the JSP page in either of the following two ways.

```
<jsp:getProperty name="book1" property="title" />
<%= book1.getTitle() %>
```

The first approach is preferable in this case, since the syntax is more accessible to Web page designers who are not familiar with the Java programming language. If you create objects with `jsp:useBean` instead of an equivalent JSP scriptlet, be syntactically consistent and output bean properties with `jsp:getProperty` instead of the equivalent JSP expression. However, direct access to the variable is useful when you are using loops, conditional statements, and methods not represented as properties.

For you who are not familiar with the concept of bean properties, the standard interpretation of the statement "this bean has a property of type `T` called `foo`" is "this class has a method called `getFoo` that returns something of type `T`, and it has another method called `setFoo` that takes a `T` as an argument and stores it for later access by `getFoo`."

Setting Simple Bean Properties: jsp:setProperty

To modify bean properties, you normally use `jsp:setProperty`. This action has several different forms, but with the simplest form you supply three attributes: `name` (which should match the `id` given by `jsp:useBean`), `property` (the name of the property to change), and `value` (the new value). In Section 14.5 we present some alternative forms of `jsp:setProperty` that let you automatically associate a property with a request parameter. That section also explains how to supply values that are computed at request time (rather than fixed strings) and discusses the type conversion conventions that let you supply string values for parameters that expect numbers, characters, or boolean values.

An alternative to using the jsp:setProperty action is to use a scriptlet that explicitly calls methods on the bean object. For example, given the book1 object shown earlier in this section, you could use either of the following two forms to modify the title property.

```
<jsp:setProperty name="book1"
                 property="title"
                 value="Core Servlets and JavaServer Pages" />
<% book1.setTitle("Core Servlets and JavaServer Pages"); %>
```

Using jsp:setProperty has the advantage that it is more accessible to the nonprogrammer, but direct access to the object lets you perform more complex operations such as setting the value conditionally or calling methods other than get*Xxx* or set*Xxx* on the object.

[Team LiB] PREVIOUS NEXT