14.6 Sharing Beans

Up to this point, we have treated the objects that were created with `jsp:useBean` as though they were simply bound to local variables in the `_jspService` method (which is called by the `service` method of the servlet that is generated from the page). Although the beans are indeed bound to local variables, that is not the only behavior. They are also stored in one of four different locations, depending on the value of the optional `scope` attribute of `jsp:useBean`.

When you use `scope`, the system first looks for an existing bean of the specified name in the designated location. Only when the system fails to find a preexisting bean does it create a new one. This behavior lets a servlet handle complex user requests by setting up beans, storing them in one of the three standard shared locations (the request, the session, or the servlet context), then forwarding the request to one of several possible JSP pages to present results appropriate to the request data. For details on this approach, see Chapter 15 (Integrating Servlets and JSP: The Model View Controller (MVC) Architecture).

As described below, the `scope` attribute has four possible values: `page` (the default), `request`, `session`, and `application`.

- **`<jsp:useBean ... scope="page" />`**

  This is the default value; you get the same behavior if you omit the `scope` attribute entirely. The `page` scope indicates that, in addition to being bound to a local variable, the bean object should be placed in the `PageContext` object for the duration of the current request. Storing the object there means that servlet code can access it by calling `getAttribute` on the predefined `pageContext` variable.

  Since every page and every request has a different `PageContext` object, using `scope="page"` (or omitting `scope`) indicates that the bean is not shared and thus a new bean will be created for each request.

- **`<jsp:useBean ... scope="request" />`**

  This value signifies that, in addition to being bound to a local variable, the bean object should be placed in the `HttpServletRequest` object for the duration of the current request, where it is available by means of the `getAttribute` method.

  Although at first glance it appears that this scope also results in unshared beans, two JSP pages or a JSP page and a servlet will share request objects when you use `jsp:include` (Section 13.1), `jsp:forward` (Section 13.3), or the `include` or `forward` methods of `RequestDispatcher` (Chapter 15).

  Storing values in the request object is common when the MVC (Model 2) architecture is used. For details, see Chapter 15.

- **`<jsp:useBean ... scope="session" />`**

  This value means that, in addition to being bound to a local variable, the bean will be stored in the `HttpSession` object associated with the current request, where it can be retrieved with `getAttribute`.

  Thus, this scope lets JSP pages easily perform the type of session tracking described in Chapter 9.

- **`<jsp:useBean ... scope="application" />`**

  This value means that, in addition to being bound to a local variable, the bean will be stored in the `ServletContext` available through the predefined `application` variable or by a call to `getServletContext`. The `ServletContext` is shared by all servlets and JSP pages in the Web application. Values in the `ServletContext` can be retrieved with the `getAttribute` method.

Creating Beans Conditionally

To make bean sharing more convenient, you can conditionally evaluate bean-related elements in two situations.

First, a `jsp:useBean` element results in a new bean being instantiated only if no bean with the same `id` and `scope` can be found. If a bean with the same `id` and `scope` *is* found, the preexisting bean is simply bound to the variable referenced by `id`.

Second, instead of

```
<jsp:useBean ... />
```

you can use

```
<jsp:useBean ...>statements</jsp:useBean>
```

The point of using the second form is that the statements between the `jsp:useBean` start and end tags are executed *only* if a new bean is created, *not* if an existing bean is used. Because `jsp:useBean` invokes the default (zero-argument) constructor, you frequently need to modify the properties after the bean is created. To mimic a constructor, however, you should make these modifications only when the bean is first created, not when an existing (and presumably updated) bean is accessed. No problem: multiple pages can contain `jsp:setProperty` statements between the start and end tags of `jsp:useBean`; only the page first accessed executes the statements.

For example, Listing 14.8 shows a simple bean that defines two properties: `accessCount` and `firstPage`. The `accessCount` property records cumulative access counts to any of a set of related pages and thus should be executed for all requests. The `firstPage` property stores the name of the first page that was accessed and thus should be executed only by the page that is first accessed. To enforce the distinction, we place the `jsp:setProperty` statement that updates the `accessCount` property in unconditional code and place the `jsp:setProperty` statement for `firstPage` between the start and end tags of `jsp:useBean`.

Listing 14.9 shows the first of three pages that use this approach. The source code archive at http://www.coreservlets.com/ contains the other two nearly identical pages. Figure 14-5 shows a typical result.

Listing 14.8 AccessCountBean.java

```
package coreservlets;

/** Simple bean to illustrate sharing beans through
 *  use of the scope attribute of jsp:useBean.
 */

public class AccessCountBean {
  private String firstPage;
  private int accessCount = 1;
```

```
  public String getFirstPage() {
    return(firstPage);
  }

  public void setFirstPage(String firstPage) {
    this.firstPage = firstPage;
  }

  public int getAccessCount() {
    return(accessCount);
  }

  public void setAccessCountIncrement(int increment) {
    accessCount = accessCount + increment;
  }
}
```

Listing 14.9 SharedCounts1.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Shared Access Counts: Page 1</TITLE>
<LINK REL=STYLESHEET
      HREF="JSP-Styles.css"
      TYPE="text/css">
</HEAD>
<BODY>
<TABLE BORDER=5 ALIGN="CENTER">
  <TR><TH CLASS="TITLE">
      Shared Access Counts: Page 1</TABLE>
<P>
<jsp:useBean id="counter"
             class="coreservlets.AccessCountBean"
             scope="application">
  <jsp:setProperty name="counter"
                   property="firstPage"
                   value="SharedCounts1.jsp" />
</jsp:useBean>
Of SharedCounts1.jsp (this page),
<A HREF="SharedCounts2.jsp">SharedCounts2.jsp</A>, and
<A HREF="SharedCounts3.jsp">SharedCounts3.jsp</A>,
<jsp:getProperty name="counter" property="firstPage" />
was the first page accessed.
<P>
Collectively, the three pages have been accessed
<jsp:getProperty name="counter" property="accessCount" />
times.
<jsp:setProperty name="counter" property="accessCountIncrement"
                 value="1" />
</BODY></HTML>
```

Figure 14-5. Result of a user visiting `SharedCounts3.jsp`. The first page visited by any user was `SharedCounts2.jsp`. `SharedCounts1.jsp`, `SharedCounts2.jsp`, and `SharedCounts3.jsp` were collectively visited a total of twelve times after the server was last started but before the visit shown in this figure.

[Team LiB]

PREVIOUS NEXT