14.7 Sharing Beans in Four Different Ways: An Example

In this section, we give an extended example that illustrates the various aspects of bean use:

- Using beans as utility classes that can be tested separately from JSP pages.
- Using unshared (page-scoped) beans.
- Sharing request-scoped beans.
- Sharing session-scoped beans.
- Sharing application-scoped (i.e., `ServletContext`-scoped) beans.

Before moving on to the examples, one caution is warranted. When you store beans in different scopes, be sure to use different names for each bean. Otherwise, servers can get confused and retrieve the incorrect bean.

Core Warning

Do not use the same bean name for beans stored in different locations. For every bean, use a unique value of `id` in `jsp:useBean`.

Building the Bean and the Bean Tester

The fundamental use of beans is as basic utility (helper) classes. We want to reiterate as strongly as possible: except for very short snippets, Java code that is directly inserted into JSP pages is harder to write, compile, test, debug, and reuse than regular Java classes.

For example, Listing 14.10 presents a small Java object that represents a food item with two properties: `level` and `goesWith` (i.e., four methods: `getLevel`, `setLevel`, `getGoesWith`, and `setGoesWith`). Perhaps this object is so simple that just looking at the source code suffices to show that it is implemented correctly. Perhaps. But how many times have you thought that before, only to uncover a bug later? In any case, a more complex class would surely require testing, so Listing 14.11 presents a test routine. Notice that the utility class represents a value in the application domain and is not dependent on any servlet- or JSP-specific classes. So, it can be tested entirely independently of the server. Listing 14.12 shows some representative output.

Listing 14.10 BakedBean.java

```
package coreservlets;

/** Small bean to illustrate various bean-sharing mechanisms. */

public class BakedBean {
  private String level = "half-baked";
  private String goesWith = "hot dogs";

  public String getLevel() {
```

```
    return(level);
  }

  public void setLevel(String newLevel) {
    level = newLevel;
  }

  public String getGoesWith() {
    return(goesWith);
  }

  public void setGoesWith(String dish) {
    goesWith = dish;
  }
}
```

Listing 14.11 BakedBeanTest.java

```
package coreservlets;

/** A small command-line program to test the BakedBean. */

public class BakedBeanTest {
  public static void main(String[] args) {
    BakedBean bean = new BakedBean();
    System.out.println("Original bean: " +
                       "level=" + bean.getLevel() +
                       ", goesWith=" + bean.getGoesWith());
    if (args.length>1) {
      bean.setLevel(args[0]);
      bean.setGoesWith(args[1]);
      System.out.println("Updated bean: " +
                         "level=" + bean.getLevel() +
                         ", goesWith=" + bean.getGoesWith());
    }
  }
}
```

Listing 14.12 Output of BakedBeanTest.java

```
Prompt> java coreservlets.BakedBeanTest gourmet caviar
Original bean: level=half-baked, goesWith=hot dogs
Updated bean: level=gourmet, goesWith=caviar
```

Using scope="page"—No Sharing

OK, after (and *only* after) we are satisfied that the bean works properly, we are ready to use it in a JSP page. The first application is to create, modify, and access the bean entirely within a single page request. For that, we use the following:

- **Create the bean:** use `jsp:useBean` with `scope="page"` (or no `scope` at all, since `page` is the default).
- **Modify the bean:** use `jsp:setProperty` with `property="*"`. Then, supply request parameters that match the bean property names.
- **Access the bean:** use `jsp:getProperty`.

Listing 14.13 presents a JSP page that applies these three techniques. Figures 14-6 and 14-7 illustrate that the bean is available only for the life of the page.

Listing 14.13 BakedBeanDisplay-page.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Baked Bean Values: page-based Sharing</TITLE>
<LINK REL=STYLESHEET
      HREF="JSP-Styles.css"
      TYPE="text/css">
</HEAD>
<BODY>
<H1>Baked Bean Values: page-based Sharing</H1>
<jsp:useBean id="pageBean" class="coreservlets.BakedBean" />
<jsp:setProperty name="pageBean" property="*" />
<H2>Bean level:
<jsp:getProperty name="pageBean" property="level" /></H2>
<H2>Dish bean goes with:
<jsp:getProperty name="pageBean" property="goesWith" /></H2>
</BODY></HTML>
```

Figure 14-6. Initial request to `BakedBeanDisplay-page.jsp`—`BakedBean` properties persist within the page.

Figure 14-7. Subsequent request to `BakedBeanDisplay-page.jsp`—`BakedBean` properties do not persist between requests.

Using Request-Based Sharing

The second application is to create, modify, and access the bean within two different pages that share the same request object. Recall that a second page shares the request object of the first page if the second page is invoked with `jsp:include`, `jsp:forward`, or the `include` or `forward` methods of `RequestDispatcher`. To get the desired behavior, we use the following:

- **Create the bean:** use `jsp:useBean` with `scope="request"`.
- **Modify the bean:** use `jsp:setProperty` with `property="*"`. Then, supply request parameters that match the bean property names.
- **Access the bean in the first page:** use `jsp:getProperty`. Then, use `jsp:include` to invoke the second page.
- **Access the bean in the second page:** use `jsp:useBean` with the same `id` as on the first page, again with `scope="request"`. Then, use `jsp:getProperty`.

Listings 14.14 and 14.15 present a pair of JSP pages that applies these four techniques. Figures 14-8 and 14-9 illustrate that the bean is available in the second page but is not stored between requests.

Listing 14.14 BakedBeanDisplay-request.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Baked Bean Values: request-based Sharing</TITLE>
<LINK REL=STYLESHEET
      HREF="JSP-Styles.css"
      TYPE="text/css">
</HEAD>
<BODY>
<H1>Baked Bean Values: request-based Sharing</H1>
<jsp:useBean id="requestBean" class="coreservlets.BakedBean"
             scope="request" />
```

```
<jsp:setProperty name="requestBean" property="*" />
<H2>Bean level:
<jsp:getProperty name="requestBean" property="level" /></H2>
<H2>Dish bean goes with:
<jsp:getProperty name="requestBean" property="goesWith" /></H2>
<jsp:include page="BakedBeanDisplay-snippet.jsp" />
</BODY></HTML>
```

Listing 14.15 BakedBeanDisplay-snippet.jsp

```
<H1>Repeated Baked Bean Values: request-based Sharing</H1>
<jsp:useBean id="requestBean" class="coreservlets.BakedBean"
             scope="request" />
<H2>Bean level:
<jsp:getProperty name="requestBean" property="level" /></H2>
<H2>Dish bean goes with:
<jsp:getProperty name="requestBean" property="goesWith" /></H2>
```

Figure 14-8. Initial request to `BakedBeanDisplay-request.jsp`—`BakedBean` properties persist to included pages.

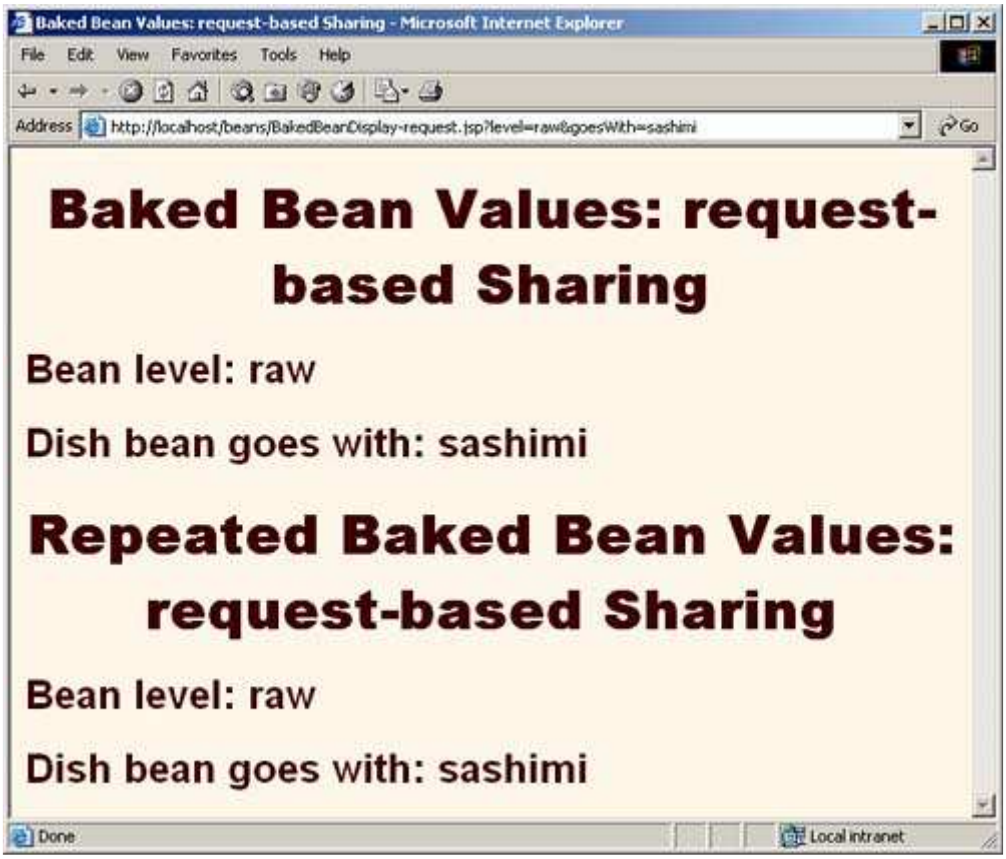
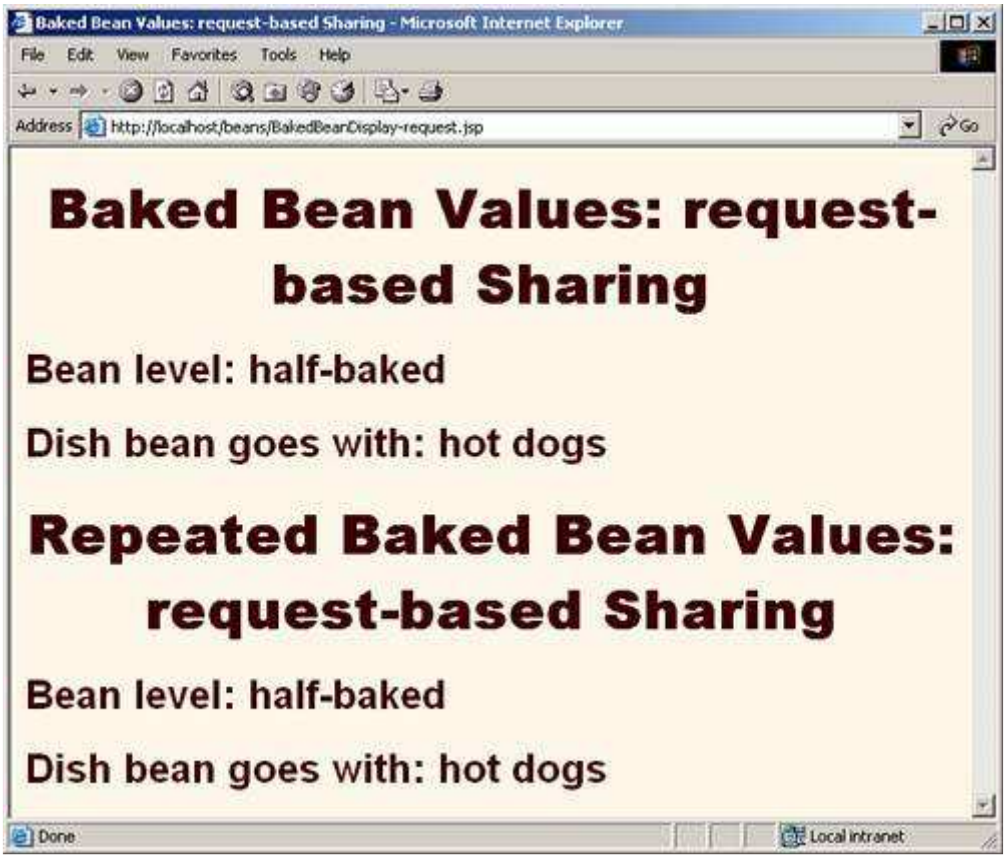


Figure 14-9. Subsequent request to `BakedBeanDisplay-request.jsp`—`BakedBean` properties do not persist between requests.



Using Session-Based Sharing

The third application involves two parts. First, we want to create, modify, and access the bean within a page. Second, if the *same* client returns to the page, he or she should see the previously modified bean. A classic case of session tracking. So, to get the desired behavior, we use the following:

- **Create the bean:** use `jsp:useBean` with `scope="session"`.
- **Modify the bean:** use `jsp:setProperty` with `property="*"`. Then, supply request parameters that match the bean property names.
- **Access the bean in the initial request:** use `jsp:getProperty` in the request in which `jsp:setProperty` is invoked.
- **Access the bean later:** use `jsp:getProperty` in a request that does not include request parameters and thus does not invoke `jsp:setProperty`. If this request is from the same client (within the session timeout), the previously modified value is seen. If this request is from a different client (or after the session timeout), a newly created bean is seen.

Listing 14.16 presents a JSP page that applies these techniques. Figure 14-10 shows the initial request. Figures 14-11 and 14-12 illustrate that the bean is available in the same session, but not in other sessions. Note that we would have gotten similar behavior if the `jsp:useBean` and `jsp:getProperty` code were repeated in multiple JSP pages: as long as the pages are accessed by the same client, the previous values will be preserved.

Listing 14.16 BakedBeanDisplay-session.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
```

```
<HEAD>
<TITLE>Baked Bean Values: session-based Sharing</TITLE>
<LINK REL=STYLESHEET
      HREF="JSP-Styles.css"
      TYPE="text/css">
</HEAD>
<BODY>
<H1>Baked Bean Values: session-based Sharing</H1>
<jsp:useBean id="sessionBean" class="coreservlets.BakedBean"
             scope="session" />

<jsp:setProperty name="sessionBean" property="*" />
<H2>Bean level:
<jsp:getProperty name="sessionBean" property="level" /></H2>
<H2>Dish bean goes with:
<jsp:getProperty name="sessionBean" property="goesWith" /></H2>
</BODY></HTML>
```

Figure 14-10. Initial request to `BakedBeanDisplay-session.jsp`.

Figure 14-11. Subsequent request to `BakedBeanDisplay-session.jsp`—`BakedBean` properties persist between requests if the request is from the same client in the same session.

Figure 14-12. Subsequent request to `BakedBeanDisplay-session.jsp`—`BakedBean` properties do not persist between requests if the request is from a different client (as here) or is in a different session.

Using ServletContext-Based Sharing

The fourth and final application also involves two parts. First, we want to create, modify, and access the bean within a page. Second, if *any* client comes to the page later, he or she should see the previously modified bean. What else besides the `ServletContext` provides such global access? So, to get the desired behavior, we use the following:

- **Create the bean:** use `jsp:useBean` with `scope="application"`.
- **Modify the bean:** use `jsp:setProperty` with `property="*"`. Then, supply request parameters that match the bean property names.
- **Access the bean in the initial request:** use `jsp:getProperty` in the request in which

`jsp:setProperty` is invoked.

- **Access the bean later:** use `jsp:getProperty` in a request that does not include request parameters and thus does not invoke `jsp:setProperty`. Whether this request is from the same client or a different client (regardless of the session timeout), the previously modified value is seen.

Listing 14.17 presents a JSP page that applies these techniques. Figure 14-13 shows the initial request. Figures 14-14 and 14-15 illustrate that the bean is available to multiple clients later. Note that we would have gotten similar behavior if the `jsp:useBean` and `jsp:getProperty` code were repeated in multiple JSP pages.

Listing 14.17 BakedBeanDisplay-application.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Baked Bean Values: application-based Sharing</TITLE>
<LINK REL=STYLESHEET
      HREF="JSP-Styles.css"
      TYPE="text/css">
</HEAD>
<BODY>
<H1>Baked Bean Values: application-based Sharing</H1>
<jsp:useBean id="applicationBean" class="coreservlets.BakedBean"
             scope="application" />
<jsp:setProperty name="applicationBean" property="*" />
<H2>Bean level:
<jsp:getProperty name="applicationBean" property="level" /></H2>
<H2>Dish bean goes with:
<jsp:getProperty name="applicationBean" property="goesWith"/></H2>
</BODY></HTML>
```

Figure 14-13. Initial request to `BakedBeanDisplay-application.jsp`.

Figure 14-14. Subsequent request to `BakedBeanDisplay-application.jsp`—`BakedBean` properties persist between requests.

Figure 14-15. Subsequent request to `BakedBeanDisplay-application`—`BakedBean` properties persist between requests even if the request is from a different client (as here) or is in a different session.

[Team LiB]
PREVIOUS NEXT