14.5 Setting Bean Properties: Advanced Techniques

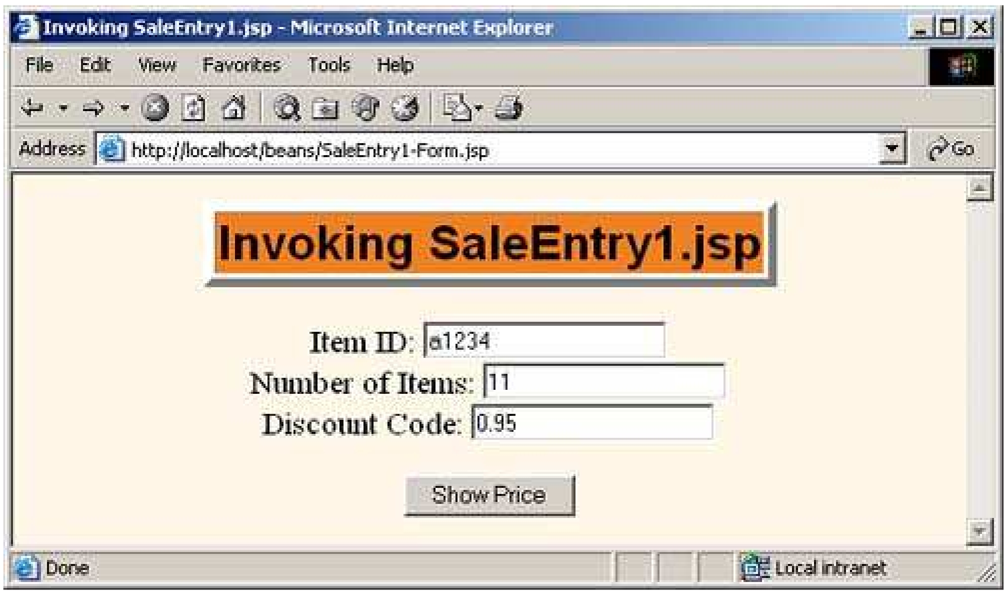
You normally use `jsp:setProperty` to set bean properties. The simplest form of this action takes three attributes: `name` (which should match the `id` given by `jsp:useBean`), `property` (the name of the property to change), and `value` (the new value).

For example, the `SaleEntry` class shown in Listing 14.3 has an `itemID` property (a `String`), a `numItems` property (an `int`), a `discountCode` property (a `double`), and two read-only properties, `itemCost` and `totalCost` (each of type `double`). Listing 14.4 shows a JSP file that builds an instance of the `SaleEntry` class by means of:

```
<jsp:useBean id="entry" class="coreservlets.SaleEntry" />
```

Listing 14.5 (Figure 14-3) gives the HTML form that collects the request parameters. The results are shown in Figure 14-4.

Figure 14-3. Front end to `SaleEntry1.jsp`.


Invoking SaleEntry1.jsp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address
Go
Done
Local intranet

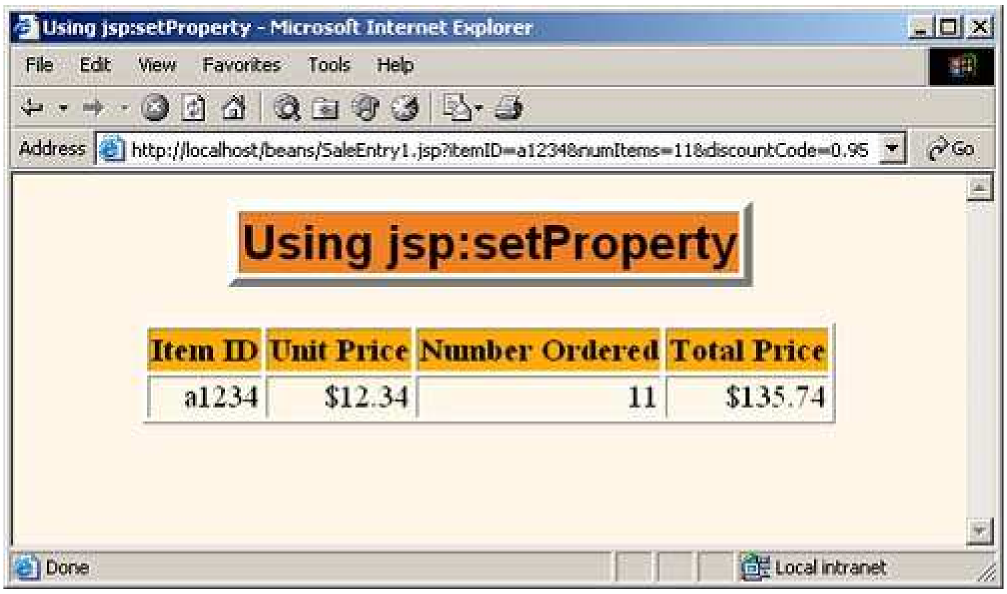
http://localhost/beans/SaleEntry1-Form.jsp

Invoking SaleEntry1.jsp

Item ID: a1234
Number of Items: 11
Discount Code: 0.95

Show Price

Figure 14-4. Result of `SaleEntry1.jsp`.


Using jsp:setProperty - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/beans/SaleEntry1.jsp?itemID=a1234&numItems=11&discountCode=0.95
Go
Done
Local intranet

Using jsp:setProperty

Item ID	Unit Price	Number Ordered	Total Price
a1234	$12.34	11	$135.74

Once the bean is instantiated, using a request parameter to set the `itemID` is straightforward, as shown below.

```
<jsp:setProperty
    name="entry"
    property="itemID"
    value='<%= request.getParameter("itemID") %>' />
```

Notice that we used a JSP expression for the `value` attribute. Most JSP attribute values have to be fixed strings, but the `value` attribute of `jsp:setProperty` is permitted to be a request time expression. If the expression uses double quotes internally, recall that single quotes can be used instead of double quotes around attribute values and that `\'` and `\"` can be used to represent single or double quotes within an attribute value. In any case, the point is that it is *possible* to use JSP expressions here, but doing so requires the use of explicit Java code. In some applications, avoiding such explicit code is the main reason for using beans in the first place. Besides, as the next examples will show, the situation becomes much more complicated when the bean property is not of type `String`. The next two subsections will discuss how to solve these problems.

Listing 14.3 SaleEntry.java

```
package coreservlets;

/** Simple bean to illustrate the various forms
 *  of jsp:setProperty.
 */

public class SaleEntry {
  private String itemID = "unknown";
  private double discountCode = 1.0;
  private int numItems = 0;

  public String getItemID() {
    return(itemID);
  }

  public void setItemID(String itemID) {
    if (itemID != null) {
      this.itemID = itemID;
    } else {
      this.itemID = "unknown";
    }
  }

  public double getDiscountCode() {
    return(discountCode);
  }

  public void setDiscountCode(double discountCode) {
    this.discountCode = discountCode;
  }

  public int getNumItems() {
    return(numItems);
  }

  public void setNumItems(int numItems) {
    this.numItems = numItems;
  }

  // In real life, replace this with database lookup.
  // See Chapters 17 and 18 for info on accessing databases
  // from servlets and JSP pages.

  public double getItemCost() {
    double cost;
    if (itemID.equals("a1234")) {
      cost = 12.99*getDiscountCode();
    } else {
      cost = -9999;
    }
    return(roundToPennies(cost));
  }

  private double roundToPennies(double cost) {
    return(Math.floor(cost*100)/100.0);
  }

  public double getTotalCost() {
    return(getItemCost() * getNumItems());
  }
}
```

Listing 14.4 SaleEntry1.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Using jsp:setProperty</TITLE>
<LINK REL=STYLESHEET
      HREF="JSP-Styles.css"
      TYPE="text/css">
</HEAD>
<BODY>
<CENTER>
<TABLE BORDER=5>
  <TR><TH CLASS="TITLE">
      Using jsp:setProperty</TABLE>
<jsp:useBean id="entry" class="coreservlets.SaleEntry" />
<jsp:setProperty
    name="entry"
    property="itemID"
    value='<%= request.getParameter("itemID") %>' />
<%
int numItemsOrdered = 1;
```

```
try {
  numItemsOrdered =
    Integer.parseInt(request.getParameter("numItems"));
} catch(NumberFormatException nfe) {}
%>
<jsp:setProperty
    name="entry"
    property="numItems"
    value="<%= numItemsOrdered %>" />
<%
double discountCode = 1.0;
try {
  String discountString =
    request.getParameter("discountCode");
  discountCode =
    Double.parseDouble(discountString);
} catch(NumberFormatException nfe) {}
%>
<jsp:setProperty
    name="entry"
    property="discountCode"
    value="<%= discountCode %>" />
<BR>
<TABLE BORDER=1>
<TR CLASS="COLORED">
  <TH>Item ID<TH>Unit Price<TH>Number Ordered<TH>Total Price
<TR ALIGN="RIGHT">
  <TD><jsp:getProperty name="entry" property="itemID" />
  <TD>$<jsp:getProperty name="entry" property="itemCost" />
  <TD><jsp:getProperty name="entry" property="numItems" />
  <TD>$<jsp:getProperty name="entry" property="totalCost" />
</TABLE>
</CENTER></BODY></HTML>
```

Listing 14.5 SaleEntry1-Form.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Invoking SaleEntry1.jsp</TITLE>
<LINK REL=STYLESHEET
      HREF="JSP-Styles.css"
      TYPE="text/css">
</HEAD>
<BODY>
<CENTER>
<TABLE BORDER=5>
  <TR><TH CLASS="TITLE">
      Invoking SaleEntry1.jsp</TABLE>
<FORM ACTION="SaleEntry1.jsp">
  Item ID: <INPUT TYPE="TEXT" NAME="itemID"><BR>
  Number of Items: <INPUT TYPE="TEXT" NAME="numItems"><BR>
  Discount Code: <INPUT TYPE="TEXT" NAME="discountCode"><P>
  <INPUT TYPE="SUBMIT" VALUE="Show Price">
</FORM>
</CENTER></BODY></HTML>
```

Associating Individual Properties with Input Parameters

Setting the `itemID` property is easy since its value is a `String`. Setting the `numItems` and `discountCode` properties is a bit more problematic since their values must be numbers whereas `getParameter` returns a `String`. Here is the somewhat cumbersome code required to set numItems.

```
<%
int numItemsOrdered = 1;
try {
  numItemsOrdered =
    Integer.parseInt(request.getParameter("numItems"));
} catch(NumberFormatException nfe) {}
%>
<jsp:setProperty
    name="entry"
    property="numItems"
    value="<%= numItemsOrdered %>" />
```

Fortunately, JSP has a nice solution to this problem. It lets you associate a property with a request parameter and automatically perform type conversion from strings to numbers, characters, and boolean values. Instead of using the value attribute, you use param to name an input parameter. The value of the named request parameter is automatically used as the value of the bean property, and type conversions from String to primitive types (byte, int, double, etc.) and wrapper classes (Byte, Integer, Double, etc.) are automatically performed. If the specified parameter is missing from the request, no action is taken (the system does not pass null to the associated property). So, for example, setting the numItems property can be simplified to:

```
<jsp:setProperty
    name="entry"
    property="numItems"
    param="numItems" />
```

You can simplify the code slightly if the request parameter name and the bean property name are the same. In that case, you can omit param as in the following example.

```
<jsp:setProperty
    name="entry"
    property="numItems" /> <%-- param="numItems" is assumed. --%>
```

We prefer the slightly longer form that lists the parameter explicitly. Listing 14.6 shows the relevant part of the JSP page reworked in this manner.

Listing 14.6 SaleEntry2.jsp

```
...
<jsp:useBean id="entry" class="coreservlets.SaleEntry" />
<jsp:setProperty
    name="entry"
    property="itemID"
    param="itemID" />
<jsp:setProperty
    name="entry"
    property="numItems"
    param="numItems" />
<jsp:setProperty
    name="entry"
    property="discountCode"
    param="discountCode" />
...
```

Associating All Properties with Request Parameters

Associating a property with a request parameter saves you the bother of performing conversions for many of the simple built-in types. JSP lets you take the process one step further by associating *all* properties with identically named request parameters. All you have to do is to supply `"*"` for the `property` parameter. So, for example, all three of the `jsp:setProperty` statements of Listing 14.6 can be replaced by the following simple line. Listing 14.7 shows the relevant part of the page.

```
<jsp:setProperty name="entry" property="*" />
```

Listing 14.7 SaleEntry3.jsp

```
...
<jsp:useBean id="entry" class="coreservlets.SaleEntry" />
<jsp:setProperty name="entry" property="*" />
...
```

This approach lets you define simple "form beans" whose properties correspond to the request parameters and get populated automatically. The system starts with the request parameters and looks for matching bean properties, not the other way around. Thus, no action is taken for bean properties that have no matching request parameter. This behavior means that the form beans need not be populated all at once; instead, one submission can fill in part of the bean, another form can fill in more, and so on. To make use of this capability, however, you need to share the bean among multiple pages. See Section 14.6 (Sharing Beans) for details. Finally, note that servlets can also use form beans, although only by making use of some custom utilities. For details, see Section 4.7 (Automatically Populating Java Objects from Request Parameters: Form Beans).

Although this approach is simple, three small warnings are in order.

- **No action is taken when an input parameter is missing.** In particular, the system does not supply `null` as the property value. So, you usually design your beans to have identifiable default values that let you determine if a property has been modified.
- **Automatic type conversion does not guard against illegal values as effectively as does manual type conversion.** In fact, despite the convenience of automatic type conversion, some developers eschew the automatic conversion, define all of their settable bean properties to be of type `String`, and use explicit `try`/`catch` blocks to handle malformed data. At the very least, you should consider the use of error pages when using automatic type conversion.
- **Bean property names and request parameters are case sensitive.** So, the property name and request parameter name must match exactly.

[Team LiB] PREVIOUS NEXT