Chapter 6: A Sample Web Application: An Online Boat Shop

Topics in This Chapter

- Defining and using a larger Web application
- The interaction among components in a Web application
- Using sessions for per-user data
- Using the servlet context for multiuser data
- Managing information that is accessed by multiple servlets and JSP pages
- Eliminating dependencies on the Web application name

Chapter 6

Many people find it helpful to see a variety of individual capabilities brought together in a single example. This chapter walks you through a small application (an online boat shop) that contains many of the different components discussed up to this point in the book. In particular, it illustrates:

- Web application definition, structure, and use
- The *web.xml* file
- The use of relative URLs by servlets and JSP pages at various locations within the Web application
- Custom tags that are used by JSP pages in various directories
- Shared beans
- Session tracking
- The use of the `ServletContext` to share data among multiple pages
- The MVC architecture applied within a Web application
- The elimination of any dependency on the Web application name

6.1 General Configuration Files

Registering a Web application (designating the location of the top-level directory and specifying a URL prefix) is a completely server-specific process. For information on the registration process, see Section 4.1 (Registering Web Applications). Here, I'll just show the settings necessary for Tomcat.

Listing 6.1 presents the relevant part of the Tomcat *server.xml* file. Recall that, if you use the default Web application settings, it is only necessary to edit *server.xml* in Tomcat 3. In Tomcat 4, just drop the *boats* directory into *install_dir/webapps*.

Listing 6.1 *install_dir/conf/server.xml* for Tomcat (partial)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Server>
  ...
  <Context path="/boats" docBase="boats" />
</Server>
```

Listing 6.2 shows the complete *web.xml* file. Recall from Section 5.2 (The Order of Elements within the Deployment Descriptor) that it is important to arrange the *web.xml* elements within `web-app` in the proper order. Also, because the `boats` application does not use filters, life-cycle event handlers, or other new features, it uses the servlet 2.2 DTD to maximize portability.

This version of *web.xml* specifies the following behaviors:

- The `ShowItem` servlet can be accessed with the URL *http://host/boats/DisplayItem* instead of the default URL of *http://host/boats/servlet/moreservlets.ShowItem*.
- Similarly, the `ShowPurchases` servlet can be accessed with the URL *http://host/boats/DisplayPurchases* instead of *http://host/boats/servlet/moreservlets.ShowPurchases*.
- The file *index.jsp* (if it exists) should be used as the default filename for URLs that name a directory but not a file. If *index.jsp* is not found, *index.html* should be used. If neither is found, the result is server specific.

For more information on using the *web.xml* file, see Chapter 5 (Controlling Web Application Behavior with web.xml).

Listing 6.2 *boats/WEB-INF/web.xml*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC
    "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
    "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">

<web-app>
  <!-- Register names for the ShowItem and ShowPurchases
       servlets. These names will be used with servlet-mapping
       to set custom URLs.
  -->
  <servlet>
    <servlet-name>ShowItem</servlet-name>
    <servlet-class>moreservlets.ShowItem</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>ShowPurchases</servlet-name>
    <servlet-class>moreservlets.ShowPurchases</servlet-class>
  </servlet>

  <!-- Set the URL http://host/webAppName/DisplayPurchases
       to invoke the servlet that would otherwise be
       available with the URL
       http://host/webAppName/servlet/moreservlets.ShowPurchases
  -->
  <servlet-mapping>
    <servlet-name>ShowPurchases</servlet-name>
    <url-pattern>/DisplayPurchases</url-pattern>
  </servlet-mapping>

  <!-- Set the URL http://host/webAppName/DisplayItem
       to invoke the servlet that would otherwise be
       available with the URL
       http://host/webAppName/servlet/moreservlets.ShowItem
  -->
  <servlet-mapping>
    <servlet-name>ShowItem</servlet-name>
    <url-pattern>/DisplayItem</url-pattern>
  </servlet-mapping>
```

Listing 6.2 *boats/WEB-INF/web.xml (continued)*

```
  <!-- If URL gives a directory but no filename, try index.jsp
       first and index.html second. If neither is found,
       the result is server specific (e.g., a directory
       listing).
  -->

  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>
</web-app>
```

6.2 The Top-Level Page

Listing 6.3 shows *index.jsp*, the top-level page for the boat store. The result is shown in Figure 6–1. There are a couple of things to note about this page.

First, since the style sheet *app-styles.css* is also in the top-level *boats* directory, the `LINK` element in the `HEAD` of the page can simply use the style sheet filename for the `HREF` attribute. See the source code archive at *http://www.moreservlets.com* if you want the source for *app-styles.css*. Similarly, the hypertext links to the yachts, tankers, and carriers JSP pages require only a simple filename within the `HREF` attribute. One of the goals of a Web application is that it require no modifications when it is moved from server to server or when the URL prefix is changed. Section 4.5 (Handling Relative URLs in Web Applications) gives details on handling relative URLs so that they don't need modifications when the URL prefix changes, but the approach is trivial for URLs that refer to files in the same directory—just use the filename.

Second, since the yacht image is in the *boats/images/* directory, the `IMG` element uses a URL of *images/yacht.jpg*.

Third, the `taglib` element uses a `uri` of */WEB-INF/tlds/count-taglib.tld*. Although this URL begins with a slash, it refers to the */tlds* subdirectory of the *WEB-INF* directory within the *boats* directory, not to the *tlds* directory of the server's top-level *WEB-INF* directory. The reason for this behavior is that the URL is resolved by the server, not sent to the client, and the server resolves this type of URL within the Web application. See Listings 6.4 through 6.6 for the tag library definition and the tag library descriptor file.

Finally, as Listings 6.4 and 6.5 show, the `count` tag uses the servlet context to store the access count. As illustrated in Section 4.6 (Sharing Data Among Web Applications), each Web application has its own servlet context. Thus, servlets and JSP pages in other Web apps do not interfere with the access count, even if they use the same tag library.

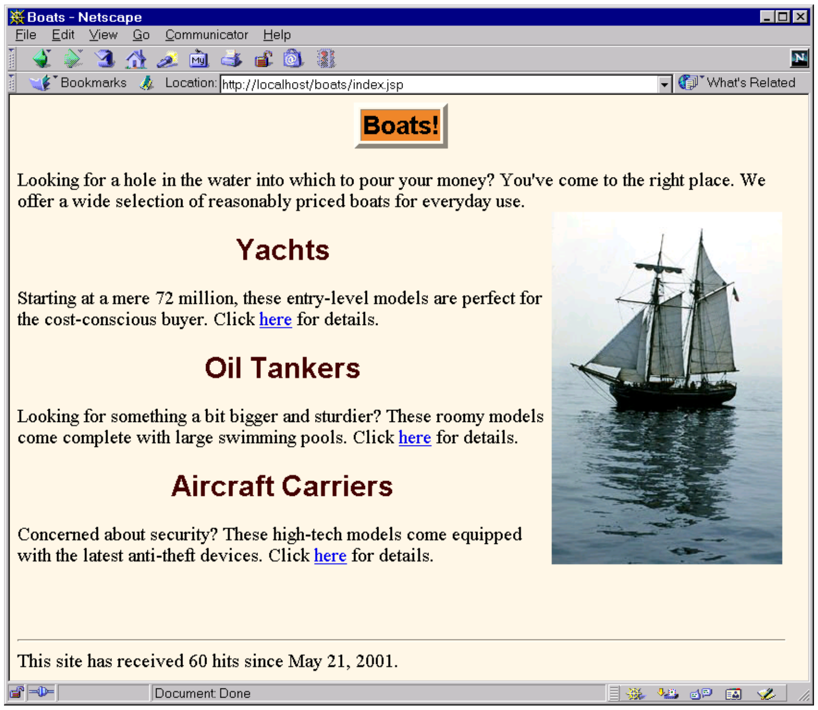
Listings 6.4 and 6.5 give the source code for the custom tag used to keep the access count for this page and the other pages on the boats site. Listing 6.6 shows the tag library descriptor file that associates the `CounterTag` code with the base tag name of `count`.

Listing 6.3 *boats/index.jsp*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Boats</TITLE>
<LINK REL=STYLESHEET
      HREF="app-styles.css"
      TYPE="text/css">
</HEAD>

<BODY>
<TABLE BORDER=5 ALIGN="CENTER">
  <TR><TH CLASS="TITLE">Boats!</TABLE>
<P>
Looking for a hole in the water into which to pour your money?
You've come to the right place. We offer a wide selection of
reasonably priced boats for everyday use.

<IMG SRC="images/yacht.jpg" WIDTH=240 HEIGHT=367
     ALIGN="RIGHT" ALT="Base-model yacht">
<H2>Yachts</H2>
Starting at a mere 72 million, these entry-level models are
perfect for the cost-conscious buyer.
Click <A HREF="yachts.jsp">here</A> for details.

<H2>Oil Tankers</H2>
Looking for something a bit bigger and sturdier? These
roomy models come complete with large swimming pools.
Click <A HREF="tankers.jsp">here</A> for details.
```

Listing 6.3 *boats/index.jsp (continued)*

```
<H2>Aircraft Carriers</H2>
Concerned about security? These high-tech models come
equipped with the latest anti-theft devices.
Click <A HREF="carriers.jsp">here</A> for details.
<P>

<%@ taglib uri="/WEB-INF/tlds/count-taglib.tld" prefix="boats" %>
<boats:count />

</BODY>
</HTML>
```


Boats - Netscape
File Edit View Go Communicator Help
Bookmarks
Location: http://localhost/boats/index.jsp
What's Related
Document: Done

Boats!

Looking for a hole in the water into which to pour your money? You've come to the right place. We offer a wide selection of reasonably priced boats for everyday use.

Yachts

Starting at a mere 72 million, these entry-level models are perfect for the cost-conscious buyer. Click here for details.

Oil Tankers

Looking for something a bit bigger and sturdier? These roomy models come complete with large swimming pools. Click here for details.

Aircraft Carriers

Concerned about security? These high-tech models come equipped with the latest anti-theft devices. Click here for details.

This site has received 60 hits since May 21, 2001.

Figure 6–1 Result of *index.jsp*.

Listing 6.4 *boats/WEB-INF/classes/moreservlets/CounterTag.java*[*]

```
package moreservlets;

import javax.servlet.*;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import java.io.*;
import java.text.*;

/** Tag that outputs a Web-app-specific hit count.
 *  For boats example.
 *  <P>
 *  The actual name of the tag is not defined here;
 *  that is given by the Tag Library Descriptor (TLD)
 *  file that is referenced by the taglib directive
 *  in the JSP file.
 */

public class CounterTag extends TagSupport {
  public int doStartTag() {
    try {
      ServletContext application =
        pageContext.getServletContext();
      Count count = (Count)application.getAttribute("count");
      if (count == null) {
        count = new Count();
        application.setAttribute("count", count);
      }
      DateFormat formatter =
        DateFormat.getDateInstance(DateFormat.MEDIUM);
      JspWriter out = pageContext.getOut();
      out.println("<BR CLEAR=\"ALL\"><BR><HR>");
      out.println("This site has received " +
                  count.getCount() + " hits since " +
                  formatter.format(count.getStartDate()) +
                  ".");
      count.incrementCount();
    } catch(IOException ioe) {
      System.out.println("Error in CounterTag: " + ioe);
    }
    return(SKIP_BODY);
  }
}
```

[*] Technically, only the *.class* file needs to go in this directory. The only requirement for the source code is that it go in a directory matching the package name (`moreservlets`).

Listing 6.5 *boats/WEB-INF/classes/moreservlets/Count.java*[*]

```
package moreservlets;

import java.util.Date;

/** Simple bean used by CounterTag. For boats example. */

public class Count {
  private int count = 1;
  private Date startDate = new Date();

  public int getCount() {
    return(count);
  }

  public void incrementCount() {
    count++;
  }

  public Date getStartDate() {
    return(startDate);
  }
}
```

[*] Technically, only the *.class* file needs to go in this directory.

Listing 6.6 *boats/WEB-INF/tlds/count-taglib.tld*

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib
 PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.1//EN"
 "http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd">

<taglib>
  <tlibversion>1.0</tlibversion>
  <jspversion>1.1</jspversion>
  <shortname>Counts</shortname>
  <info>
    A tag library for counters. From More Servlets and
    JavaServer Pages, http://www.moreservlets.com.
  </info>
  <tag>
    <name>count</name>
    <tagclass>moreservlets.CounterTag</tagclass>
    <bodycontent>empty</bodycontent>
    <info>Hit count</info>
  </tag>
</taglib>
```

6.3 The Second-Level Pages

The top-level page introduces the site and gives links to each of the three second-level pages that describe specific varieties of boats: yachts (Listing 6.7, Figure 6–2), oil tankers (Listing 6.8, Figure 6–3), and aircraft carriers (Listing 6.9, Figure 6–4). Note that simple relative URLs are used for the style sheet and for the servlet with which the form communicates when the user asks for details. Note also that the custom tag maintains the count from the top-level page. That's because the servlet context is shared by all pages in the Web app.

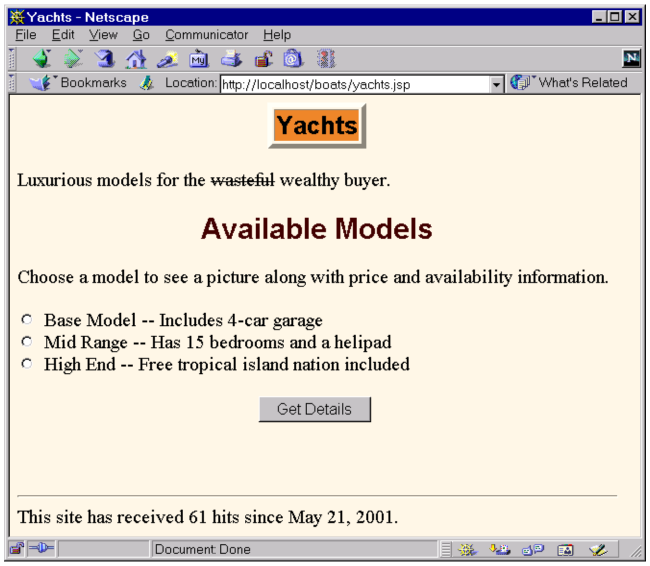
Once the user chooses a particular model and asks for more information on it, the item display servlet is invoked. Since the *web.xml* file (Listing 6.2) registers this servlet with the URL suffix *DisplayItem*, a simple relative URL can be used for the `ACTION` attribute of the `FORM` element. As described in Section 4.5 (Handling Relative URLs in Web Applications), this capability is important because it permits the servlet to use simple relative URLs either directly or by means of JSP pages to which it forwards the request. That means that the Web application name can be changed without necessitating changes to these JSP pages. See the following subsection for details on the item display servlet.

Listing 6.7 *boats/yachts.jsp*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Yachts</TITLE>
<LINK REL=STYLESHEET
      HREF="app-styles.css"
      TYPE="text/css">
</HEAD>

<BODY>
<TABLE BORDER=5 ALIGN="CENTER">
  <TR><TH CLASS="TITLE">Yachts</TABLE>
<P>
Luxurious models for the <S>wasteful</S>
wealthy buyer.

<H2>Available Models</H2>
Choose a model to see a picture along with price and
availability information.
```

Listing 6.7 *boats/yachts.jsp (continued)*

```
<FORM ACTION="DisplayItem">
<INPUT TYPE="RADIO" NAME="itemNum" VALUE="BM1">
Base Model -- Includes 4-car garage<BR>
<INPUT TYPE="RADIO" NAME="itemNum" VALUE="MR1">
Mid Range -- Has 15 bedrooms and a helipad<BR>
<INPUT TYPE="RADIO" NAME="itemNum" VALUE="HE1">
High End -- Free tropical island nation included
<P>
<CENTER>
<INPUT TYPE="SUBMIT" VALUE="Get Details">
</CENTER>
</FORM>

<%-- Note the lack of "boats" at the front of URI below --%>
<%@ taglib uri="/WEB-INF/tlds/count-taglib.tld" prefix="boats" %>
<boats:count />
</BODY>
</HTML>
```


Yachts - Netscape
File Edit View Go Communicator Help
Bookmarks Location: http://localhost/boats/yachts.jsp What's Related
Document: Done

Yachts

Luxurious models for the ~~wasteful~~ wealthy buyer.

Available Models

Choose a model to see a picture along with price and availability information.

- Base Model -- Includes 4-car garage
- Mid Range -- Has 15 bedrooms and a helipad
- High End -- Free tropical island nation included

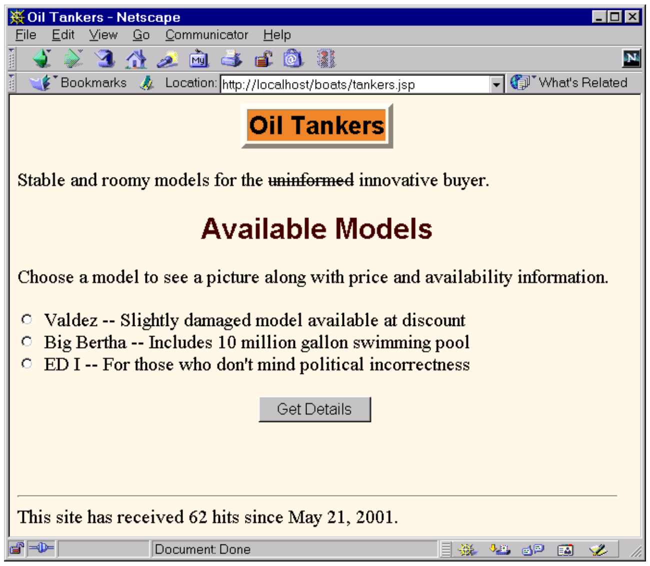
Get Details

This site has received 61 hits since May 21, 2001.

Figure 6–2 Result of *yachts.jsp*.

Listing 6.8 *boats/tankers.jsp*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Oil Tankers</TITLE>
<LINK REL=STYLESHEET
      HREF="app-styles.css"
      TYPE="text/css">
</HEAD>

<BODY>
<TABLE BORDER=5 ALIGN="CENTER">
  <TR><TH CLASS="TITLE">Oil Tankers</TABLE>
<P>
Stable and roomy models for the <S>uninformed</S>
innovative buyer.

<H2>Available Models</H2>
Choose a model to see a picture along with price and
availability information.

<FORM ACTION="DisplayItem">
<INPUT TYPE="RADIO" NAME="itemNum" VALUE="Valdez">
Valdez -- Slightly damaged model available at discount<BR>
<INPUT TYPE="RADIO" NAME="itemNum" VALUE="BigBertha">
Big Bertha -- Includes 10 million gallon swimming pool<BR>
<INPUT TYPE="RADIO" NAME="itemNum" VALUE="EcoDisaster">
ED I -- For those who don't mind political incorrectness
<P>
<CENTER>
<INPUT TYPE="SUBMIT" VALUE="Get Details">
</CENTER>
</FORM>

<%-- Note the lack of "boats" at the front of URI below --%>
<%@ taglib uri="/WEB-INF/tlds/count-taglib.tld" prefix="boats" %>
<boats:count />
</BODY>
</HTML>
```


Oil Tankers - Netscape
File Edit View Go Communicator Help
Bookmarks Location: http://localhost/boats/tankers.jsp
What's Related
Get Details
Document: Done

Oil Tankers

Stable and roomy models for the ~~uninformed~~ innovative buyer.

Available Models

Choose a model to see a picture along with price and availability information.

- Valdez -- Slightly damaged model available at discount
- Big Bertha -- Includes 10 million gallon swimming pool
- ED I -- For those who don't mind political incorrectness

This site has received 62 hits since May 21, 2001.

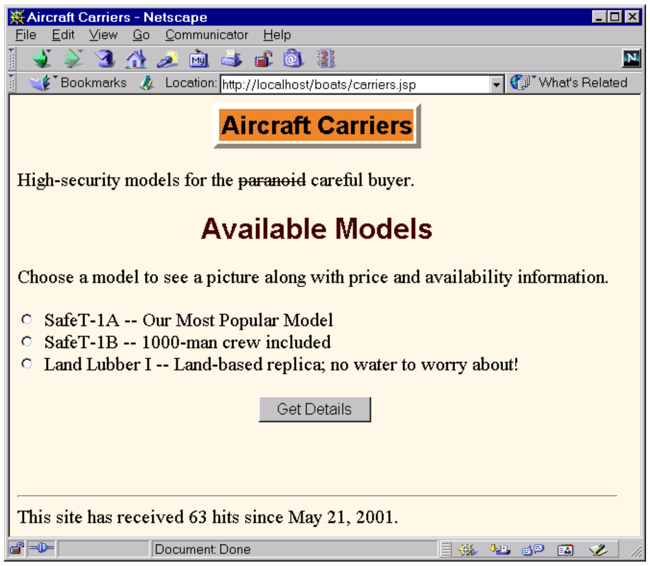
Figure 6–3 Result of *tankers.jsp*.

Listing 6.9 *boats/carriers.jsp*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Aircraft Carriers</TITLE>
<LINK REL=STYLESHEET
      HREF="app-styles.css"
      TYPE="text/css">
</HEAD>

<BODY>
<TABLE BORDER=5 ALIGN="CENTER">
  <TR><TH CLASS="TITLE">Aircraft Carriers</TABLE>
<P>
High-security models for the <S>paranoid</S> careful buyer.

<H2>Available Models</H2>
Choose a model to see a picture along with price and
availability information.
```

Listing 6.9 *boats/carriers.jsp (continued)*

```
<FORM ACTION="DisplayItem">
<INPUT TYPE="RADIO" NAME="itemNum" VALUE="SafeT-1A">
SafeT-1A -- Our Most Popular Model<BR>
<INPUT TYPE="RADIO" NAME="itemNum" VALUE="SafeT-1B">
SafeT-1B -- 1000-man crew included<BR>
<INPUT TYPE="RADIO" NAME="itemNum" VALUE="Lubber-1">
Land Lubber I -- Land-based replica; no water to worry about!
<P>
<CENTER>
<INPUT TYPE="SUBMIT" VALUE="Get Details">
</CENTER>
</FORM>

<%-- Note the lack of "boats" at the front of URI below --%>
<%@ taglib uri="/WEB-INF/tlds/count-taglib.tld" prefix="boats" %>
<boats:count />
</BODY>
</HTML>
```


Aircraft Carriers - Netscape
File Edit View Go Communicator Help
Bookmarks Location: http://localhost/boats/carriers.jsp
What's Related
Get Details
Document: Done

Aircraft Carriers

High-security models for the ~~paranoid~~ careful buyer.

Available Models

Choose a model to see a picture along with price and availability information.

○ SafeT-1A -- Our Most Popular Model
○ SafeT-1B -- 1000-man crew included
○ Land Lubber I -- Land-based replica; no water to worry about!

This site has received 63 hits since May 21, 2001.

Figure 6–4 Result of *carriers.jsp*.

6.4 The Item Display Servlet

Since prices, descriptions, and pictures of sale items can change, you don't want to create item description pages by hand. Instead, pages of this sort should be automatically generated. In real life, the data source would probably be a database that is accessed with JDBC (see Chapter 18 of *Core Servlets and JavaServer Pages*, available in PDF at *http://www.moreservlets.com*). In this case, a simple data file is used so that the example is short and so that you can run the it without a database.

The item display servlet uses the MVC architecture (see Section 3.8) to display information. First, the servlet (Listing 6.10) reads the `itemNum` request parameter. If the item number is found, the servlet uses the number as a key into the table of ships (Listing 6.13, a subclass of the more general table shown in Listing 6.14) and creates a `SimpleItem` object (Listing 6.15) from the result. The servlet then places the `SimpleItem` in the request object and forwards the request to the *ShowItem.jsp* page (Listing 6.11, Figure 6–5). The *ShowItem.jsp* page uses `jsp:useBean` (Section 3.6) to access the `SimpleItem` object, then uses that object to look up the item's description, its cost, and the location of an image file that illustrates it. If, however, the item number is missing, the `ShowItem` servlet sends the request to *MissingItem.jsp* (Listing 6.12, Figure 6–6).

Note the importance of using the *web.xml* file to give the servlet a URL at the top level of the Web app (e.g., *http://host/boats/DisplayItem*), rather than using the default URL (e.g., *http://host/boats/servlet/moreservlets.ShowItem*). Because of this simplification, the JSP page need not call `getContextPath` before using relative URLs. See Section 4.5 (Handling Relative URLs in Web Applications) for details.

Listing 6.10 *boats/WEB-INF/classes/moreservlets/ShowItem.java*[*]

```
package moreservlets;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

/** Servlet that looks up information on an item that is for
 *  sale. Uses the MVC architecture, with either
 *  MissingItem.jsp or ShowItem.jsp doing the presentation.
 *  Used in the boats Web app.
 */

public class ShowItem extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
```

Listing 6.10 *boats/WEB-INF/classes/moreservlets/ShowItem.java*[*] *(continued)*

```
    String itemNum = request.getParameter("itemNum");
    String destination;
    if (itemNum == null) {
      destination = "/MissingItem.jsp";
    } else {
      destination = "/ShowItem.jsp";
      ItemTable shipTable = ShipTable.getShipTable();
      SimpleItem item = shipTable.getItem(itemNum);
      request.setAttribute("item", item);
    }
    RequestDispatcher dispatcher =
      getServletContext().getRequestDispatcher(destination);
    dispatcher.forward(request, response);
  }
}
```

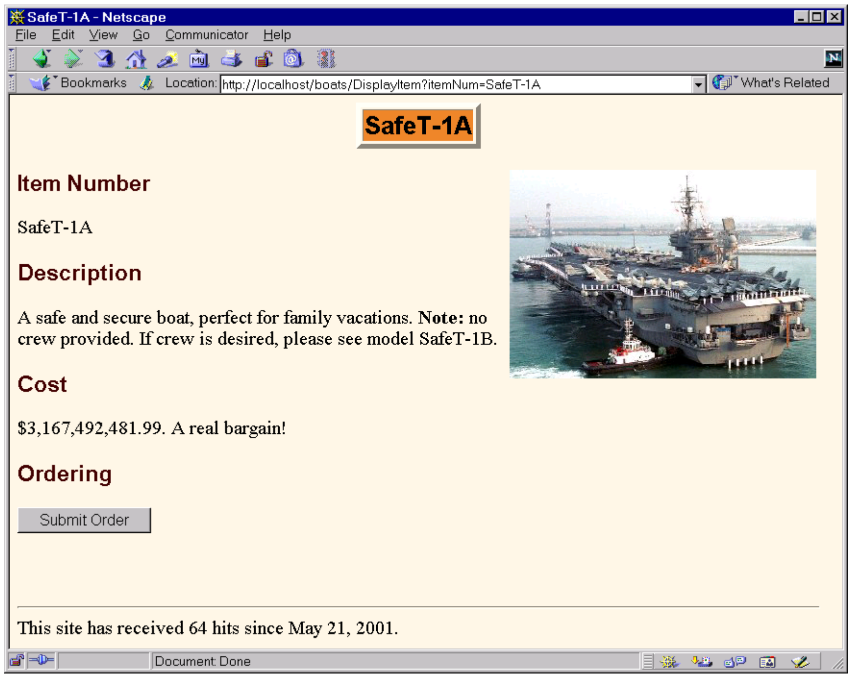
* Technically, only the *.class* file needs to go in this directory.

Listing 6.11 *boats/ShowItem.jsp*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<jsp:useBean id="item"
             class="moreservlets.SimpleItem"
             scope="request" />
<TITLE><jsp:getProperty name="item" property="itemNum" /></TITLE>
<LINK REL=STYLESHEET
      HREF="app-styles.css"
      TYPE="text/css">
</HEAD>

<BODY>
<TABLE BORDER=5 ALIGN="CENTER">
  <TR><TH CLASS="TITLE">
  <jsp:getProperty name="item" property="itemNum" /></TABLE>
<P>
<IMG SRC="<jsp:getProperty name='item' property='imageURL' />"
     ALIGN="RIGHT">

<H3>Item Number</H2>
<jsp:getProperty name="item" property="itemNum" />

<H3>Description</H2>
<jsp:getProperty name="item" property="description" />
```

Listing 6.11 *boats/ShowItem.jsp (continued)*

```
<H3>Cost</H2>
<jsp:getProperty name="item" property="costString" />.
A real bargain!

<H3>Ordering</H2>
<FORM ACTION="DisplayPurchases">
  <INPUT TYPE="HIDDEN" NAME="itemNum"
         VALUE="<jsp:getProperty name='item'
                                 property='itemNum' />">
  <INPUT TYPE="SUBMIT" VALUE="Submit Order">
</FORM>

<%-- Note the lack of "boats" at the front of URI below --%>
<%@ taglib uri="/WEB-INF/tlds/count-taglib.tld" prefix="boats" %>
<boats:count />
</BODY>
</HTML>
```


SafeT-1A - Netscape
File Edit View Go Communicator Help
Bookmarks Location: http://localhost/boats/DisplayItem?itemNum=SafeT-1A
What's Related
Submit Order
Document: Done

SafeT-1A

Item Number

SafeT-1A

Description

A safe and secure boat, perfect for family vacations. **Note:** no crew provided. If crew is desired, please see model SafeT-1B.

Cost

$3,167,492,481.99. A real bargain!

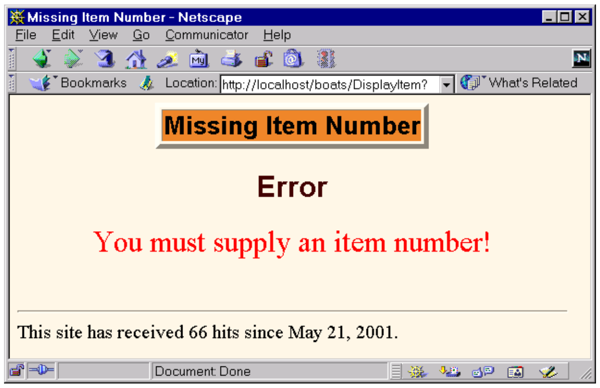
Ordering

This site has received 64 hits since May 21, 2001.

Figure 6–5 Result of the `ShowItem` servlet when model `SafeT-1A` is selected. The servlet is invoked with the registered name (`DisplayItem`) and forwards the request to *ShowItem.jsp*.

Listing 6.12 *boats/MissingItem.jsp*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Missing Item Number</TITLE>
<LINK REL=STYLESHEET
      HREF="app-styles.css"
      TYPE="text/css">
</HEAD>

<BODY>
<TABLE BORDER=5 ALIGN="CENTER">
  <TR><TH CLASS="TITLE">Missing Item Number</TABLE>
<P>

<H2>Error</H2>
<SPAN CLASS="ERROR">You must supply an item number!</SPAN>

<%-- Note the lack of "boats" at the front of URI below --%>
<%@ taglib uri="/WEB-INF/tlds/count-taglib.tld" prefix="boats" %>
<boats:count />
</BODY>
</HTML>
```


Missing Item Number - Netscape
File Edit View Go Communicator Help
Bookmarks
Location: http://localhost/boats/DisplayItem?
What's Related
Document: Done

Missing Item Number

Error

You must supply an item number!

This site has received 66 hits since May 21, 2001.

Figure 6–6 Result of the `ShowItem` servlet when no model number is selected. The servlet is invoked with the registered name (`DisplayItem`) and forwards the request to *MissingItem.jsp*.

Listing 6.13 *boats/WEB-INF/classes/moreservlets/ShipTable.java*[*]

```
package moreservlets;

/** A small collection of ships. Used in the boats Web app. */

public class ShipTable {
  private static SimpleItem[] ships =
  { // Yachts
    new SimpleItem
      ("BM1",
       "Base model yacht. Features sauna, two kitchens, " +
         "and four-car garage. <B>Perfect entry-level " +
         "yacht for the first-time buyer</B>.",
       "images/yacht.jpg",
       72678922.99),
    new SimpleItem
      ("MR1",
       "Mid-range yacht. Features helipad, bowling alley, " +
         "and 15 bedrooms. <B>Trade up from a BM1 " +
         "today!</B>.",
       "images/yacht.jpg",
       145357845.98),
    new SimpleItem
      ("HE1",
       "High-end yacht. Features onboard 18-hole golf " +
         "course, onboard polo grounds, and ski jump. " +
         "<B>Bonus: for a limited time only, a mid-sized " +
         "tropical island country will be included at " +
         "no extra cost.</B>",
       "images/yacht.jpg",
       7267892299.00),
    // Oil Tankers
    new SimpleItem
      ("Valdez",
       "Slightly damaged former Alaskan touring boat. " +
         "<B>Special price won't last long!</B>",
       "images/tanker.jpg",
       9.95),
    new SimpleItem
      ("BigBertha",
       "Tired of cramped quarters on your boat? " +
         "This roomy model has plenty of space to stretch " +
         "your legs. <B>10 million gallon onboard " +
         "swimming pool included!</B>",
       "images/tanker.jpg",
       20000000.00),
```

Listing 6.13 *boats/WEB-INF/classes/moreservlets/ShipTable.java** *(continued)*

```
     new SimpleItem
       ("EcoDisaster",
        "OK, ok, so this model is not exactly politically " +
          "correct. <B>But you're not one to pass up " +
          "a bargain just because of a few " +
          "<S>Greenpeace</S> pesky demonstrators, " +
          "are you?</B>.",
        "images/tanker.jpg",
        100000000),
     // Aircraft Carriers
     new SimpleItem
       ("SafeT-1A",
        "A safe and secure boat, perfect for family " +
          "vacations. <B>Note:</B> no crew provided. If crew " +
          "is desired, please see model SafeT-1B.",
        "images/carrier.jpg",
        3167492481.99),
      new SimpleItem
       ("SafeT-1B",
        "Just like the 1A model, but we provide the crew. " +
          "<B>Note:</B> You must pay the one million dollar " +
          "annual salary for the crew.",
        "images/carrier.jpg",
        3267492481.99),
     new SimpleItem
       ("Lubber-1",
        "All the comfort of the other models, but without " +
          "the danger. Realistic simulation provides " +
          "continuous water sounds. <B>Note:</B> " +
          "currently located in Siberia. Shipping and " +
          "handling not included.",
        "images/carrier.jpg",
        152.99)
   };

   private static ItemTable shipTable =
     new ItemTable(ships);

   public static ItemTable getShipTable() {
     return(shipTable);
   }
 }
```

* Technically, only the *.class* file needs to go in this directory.

Listing 6.14 *boats/WEB-INF/classes/moreservlets/ItemTable.java*[*]

```
package moreservlets;

import java.util.HashMap;

/** Small class that puts an array of items into a
 *  hash table, making the item number the key.
 *  Used in the boats Web app example.
 */

public class ItemTable {
  private HashMap itemMap = new HashMap();

  public ItemTable(SimpleItem[] items) {
    if (items != null) {
      SimpleItem item;
      for(int i=0; i<items.length; i++) {
        item = items[i];
        itemMap.put(item.getItemNum(), item);
      }
    }
  }

  public SimpleItem getItem(String itemNum) {
    return((SimpleItem)itemMap.get(itemNum));
  }
}
```

* Technically, only the *.class* file needs to go in this directory.

Listing 6.15 *boats/WEB-INF/classes/moreservlets/SimpleItem.java*[*]

```
package moreservlets;

import java.text.*;

/** An item that is for sale. Used in the boats Web app. */

public class SimpleItem {
  private String itemNum = "Missing item number";
  private String description = "Missing description";
  private String imageURL = "Missing image URL";
  private double cost;
  private NumberFormat formatter =
    NumberFormat.getCurrencyInstance();
```

Listing 6.15 *boats/WEB-INF/classes/moreservlets/SimpleItem.java*[*] *(continued)*

```
  public SimpleItem(String itemNum,
                    String description,
                    String imageURL,
                    double cost) {
    setItemNum(itemNum);
    setDescription(description);
    setImageURL(imageURL);
    setCost(cost);
  }

  public SimpleItem() {}

  public String getItemNum() {
    return(itemNum);
  }

  private void setItemNum(String  itemNum) {
    this.itemNum = itemNum;
  }

  public String getDescription() {
    return(description);
  }

  private void setDescription(String  description) {
    this.description = description;
  }

  public String getImageURL() {
    return(imageURL);
  }

  private void setImageURL(String  imageURL) {
    this.imageURL = imageURL;
  }

  public double getCost() {
    return(cost);
  }

  private void setCost(double cost) {
    this.cost = cost;
  }

  public String getCostString() {
    return(formatter.format(getCost()));
  }
}
```

6.5 The Purchase Display Page

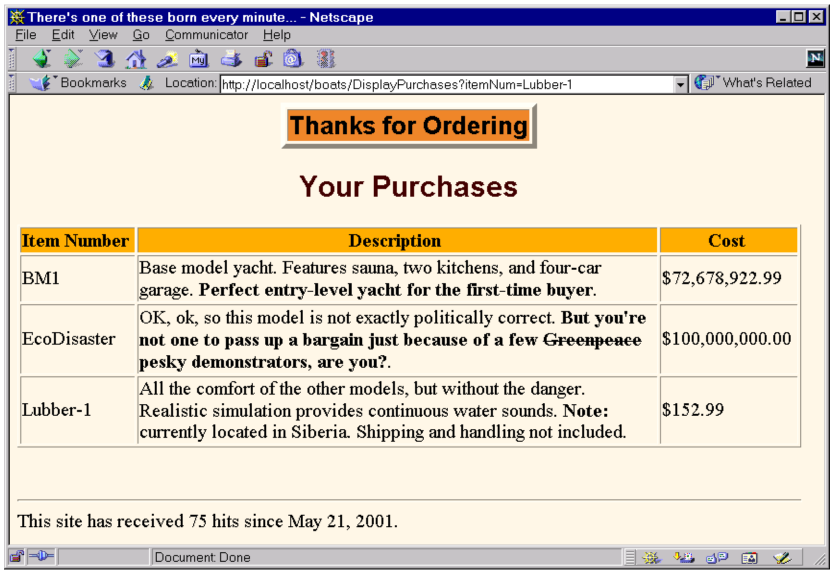
Section 9.4 of *Core Servlets and JavaServer Pages* gives detailed code for creating and using a shopping cart. (Remember that *Core Servlets and JavaServer Pages* is available in its entirety in PDF at *http://www.moreservlets.com*.) Here, I use a much more simplified “cart” to illustrate how session tracking fits in with Web applications. When a user presses the Submit Order button from one of the item display pages of the previous subsection, the item number is sent to the `ShowPurchases` servlet of Listing 6.16 (by means of the registered name of `DisplayPurchases`). This servlet looks up the item associated with the item number, puts the item into an `ItemList` (Listing 6.17), and forwards the request to the *sucker.jsp* page (Listing 6.18) to display all the items being purchased by that client in this session. See Figure 6–7 for a typical result.

Listing 6.16 *boats/WEB-INF/classes/moreservlets/ShowPurchases.java*[*]

```
package moreservlets;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

/** A simple servlet that shows a table of purchases. */

public class ShowPurchases extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    String itemNum = request.getParameter("itemNum");
    ItemTable shipTable = ShipTable.getShipTable();
    SimpleItem item = shipTable.getItem(itemNum);
    HttpSession session = request.getSession(true);
    ItemList previousItems =
      (ItemList)session.getAttribute("items");
    if (previousItems == null) {
      previousItems = new ItemList();
      session.setAttribute("items", previousItems);
    }
    previousItems.setNewItem(item);
    RequestDispatcher dispatcher =
      getServletContext().getRequestDispatcher("/sucker.jsp");
    dispatcher.forward(request, response);
  }
}
```

[*] Technically, only the *.class* file needs to go in this directory.

Note that the ItemList class (Listing 6.17) uses an ArrayList, not a Vector. Version 2.3 of the servlet API mandates the use of the Java 2 platform, so this usage does not limit portability.


There's one of these born every minute... - Netscape
File Edit View Go Communicator Help
Bookmarks
Location: http://localhost/boats/DisplayPurchases?itemNum=Lubber-1
What's Related

Thanks for Ordering

Your Purchases

Item Number	Description	Cost
BM1	Base model yacht. Features sauna, two kitchens, and four-car garage. **Perfect entry-level yacht for the first-time buyer**.	$72,678,922.99
EcoDisaster	OK, ok, so this model is not exactly politically correct. **But you're not one to pass up a bargain just because of a few ~~Greenpeace~~ pesky demonstrators, are you?**.	$100,000,000.00
Lubber-1	All the comfort of the other models, but without the danger. Realistic simulation provides continuous water sounds. **Note:** currently located in Siberia. Shipping and handling not included.	$152.99

This site has received 75 hits since May 21, 2001.

Figure 6–7 The ShowPurchases servlet after the client makes three small acquisitions. The servlet is invoked with the registered name (DisplayPurchases) and uses the *sucker.jsp* page to present the results.

Listing 6.17 *boats/WEB-INF/classes/moreservlets/ItemList.java*[*]

```
package moreservlets;

import java.util.*;

/** Very simple pseudo shopping cart. Maintains a list
 *  of items and can format them in an HTML table.
 *  Used in the boats Web app example to show that
 *  each Web app maintains its own set of sessions.
 */

public class ItemList {
  private ArrayList items = new ArrayList();
```

Listing 6.17 *boats/WEB-INF/classes/moreservlets/ItemList.java*[*] *(continued)*

```
  public synchronized void setNewItem(SimpleItem newItem) {
    if (newItem != null) {
      items.add(newItem);
    }
  }

  public synchronized String getItemTable() {
    if (items.size() == 0) {
      return("<H3>No items...</H3>");
    }
    String tableString =
      "<TABLE BORDER=1>\n" +
      "  <TR CLASS=\"COLORED\">\n" +
      "      <TH>Item Number\n" +
      "      <TH>Description\n" +
      "      <TH>Cost\n";
    for(int i=0; i<items.size(); i++) {
      SimpleItem item = (SimpleItem)items.get(i);
      tableString +=
        "  <TR><TD>" + item.getItemNum() + "\n" +
        "      <TD>" + item.getDescription() + "\n" +
        "      <TD>" + item.getCostString() + "\n";
    }
    tableString += "</TABLE>";
    return(tableString);
  }

  public synchronized String toString() {
    return("[Item List: " + items.size() + " entries.]");
  }
}
```

* Technically, only the *.class* file needs to go in this directory.

Listing 6.18 *boats/sucker.jsp*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>There's one of these born every minute...</TITLE>
<LINK REL=STYLESHEET
      HREF="app-styles.css"
      TYPE="text/css">
</HEAD>

<BODY>
<TABLE BORDER=5 ALIGN="CENTER">
  <TR><TH CLASS="TITLE">Thanks for Ordering</TABLE>

<H2>Your Purchases</H2>
<jsp:useBean id="items"
             class="moreservlets.ItemList"
             scope="session" />
<jsp:getProperty name="items" property="itemTable" />

<%-- Note the lack of "boats" at the front of URI below --%>
<%@ taglib uri="/WEB-INF/tlds/count-taglib.tld" prefix="boats" %>
<boats:count />
</BODY>
</HTML>
```