Custom-tag libraries

Since JSP1.1. Allow to create your own JSP tags. They are used for creating dynamic content

– Package **`javax.servlet.jsp.tagext`**

Developers define tag, its attributes and its how its body are interpreted, then group your tags into collections called tag libraries that can be used in any number of JSP files.

Custom tags accomplish some of the same goals as `jsp:useBean` , however,

1. Custom tags can manipulate JSP contents; beans cannot
2. Complex operations can be reduced to a significantly simpler form with custom tags that with beans
3. Custom tags require more work to set up than do beans
4. Custom tags usually define self-contained behavior, whereas beans are often define in one servlet and than used in a different servlet /JSP
5. Beans can be used in < JSP1.1

Three components:

1. Define tag handler **class** encapsulating tag behavior and must be defined in java packages

   Usually achieved by extending
   a. `javax.servlet.jsp.tagext.TagSupport` class (if the body of the tag is ignored or simply output during custom tag processing)
   b. or `javax.servlet.jsp.tagext.BodyTagSupport` (if the handler interacts with the tags's body content)

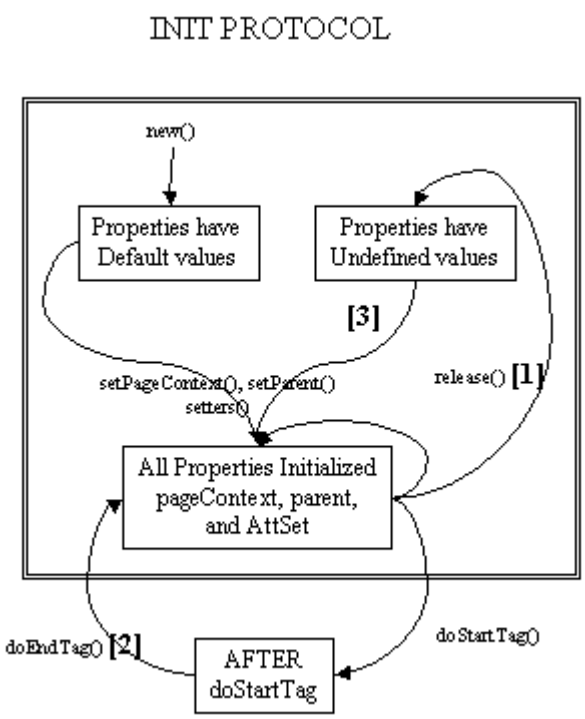
2. Define tag library descriptor file that maps the XML element names to the tag implementation
3. Define JSP that uses the tag

Attribute	Description
`uri`	Specifies the relative or absolute URI of the tag library descriptor.
`tagPrefix`	Specifies the required prefix that distinguishes custom tags from built-in tags. The prefix names **`jsp`**, **`jspx`**, **`java`**, **`javax`**, **`servlet`**, **`sun`** and **`sunw`** are reserved.

JSP container invokes `doStartTag` and `doEndTag` from interface `Tag` when it encounters starting and ending custom tags.
Exception: `JspException`

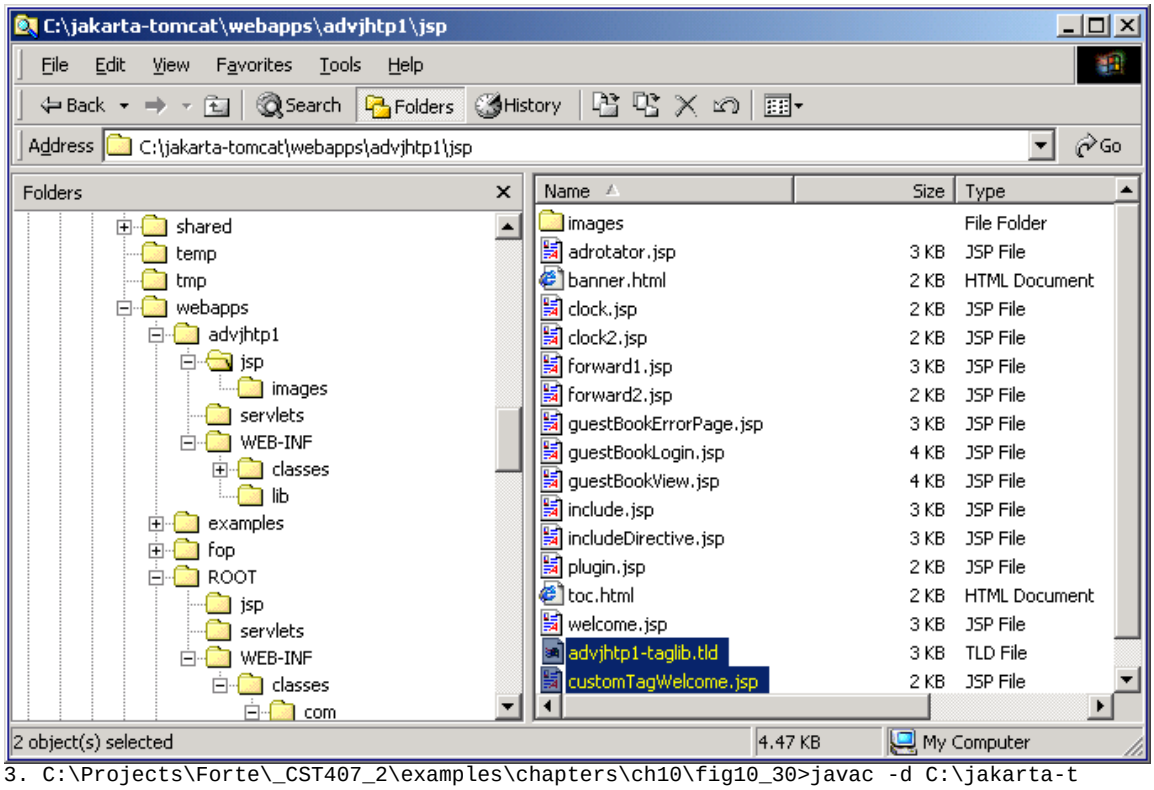
Lifecycle details are described by the transition diagram below, with the following comments:

- [1] This transition is intended to be for releasing long-term data. No guarantees are assumed on whether any properties have been retained or not.
- [2] This transition happens if and only if the tag ends normally without raising an exception
- [3] Note that since there are no guarantees on the state of the properties, a tag handler that had some optional properties set can only be reused if those properties are set to a new (known) value. This means that tag handlers can only be reused within the same "AttSet" (set of attributes that have been set).
- Check the TryCatchFinally interface for additional details related to exception handling and resource management.



Once all invocations on the tag handler are completed, the release method is invoked on it. Once a release method is invoked *all* properties, including parent and pageContext, are assumed to have been reset to an unspecified value. The page compiler guarantees that release() will be invoked on the Tag handler before the handler is released to the GC.

```
1. C:\Projects\Forte\_CST407_2\examples\chapters\ch10\fig10_30>copy customTagWelcome
.jsp C:\jakarta-tomcat\webapps\advjhtp1\jsp
        1 file(s) copied.

2. C:\Projects\Forte\_CST407_2\examples\chapters\ch10\fig10_30>copy advjhtp1-taglib.
tld C:\jakarta-tomcat\webapps\advjhtp1\jsp
        1 file(s) copied.
```




```
3. C:\Projects\Forte\_CST407_2\examples\chapters\ch10\fig10_30>javac -d C:\jakarta-t
omcat\webapps\advjhtp1\WEB-INF\classes WelcomeTagHandler.java
```

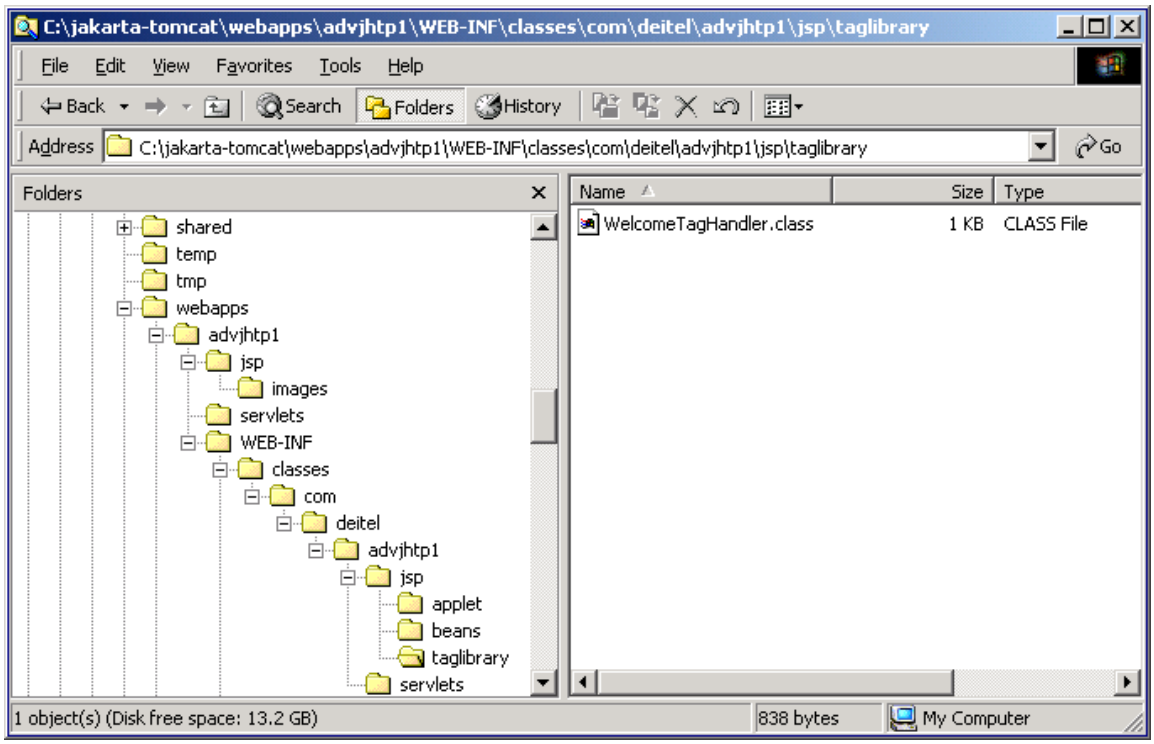


Fig. 10.30 JSP **customTag-Welcome.jsp** uses a simple custom tag.

```
1    <?xml version = "1.0"?>
2    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

```
4
5    <!-- Fig. 10.30: customTagWelcome.jsp               -->
6    <!-- JSP that uses a custom tag to output content. -->
7
8    <%-- taglib directive --%>
9    <%@ taglib uri = "advjhtp1-taglib.tld" prefix = "advjhtp1" %>
10
11   <html xmlns = "http://www.w3.org/1999/xhtml">
12
13      <head>
14         <title>Simple Custom Tag Example</title>
15      </head>
16
17      <body>
18         <p>The following text demonstrates a custom tag:</p>
19         <h1>
20            <advjhtp1:welcome />
21         </h1>
22      </body>
23
24   </html>
```

Note: If you see xml document instead of above picture replace in jsp page

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

with

```
 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.1 Transitional//EN">
```

Fig. 10.31 WelcomeTag-Handler **custom tag handler.**

```
1    // Fig. 10.31: WelcomeTagHandler.java
2    // Custom tag handler that handles a simple tag.
3    package com.deitel.advjhtp1.jsp.taglibrary;
4
5    // Java core packages
6    import java.io.*;
7
8    // Java extension packages
9    import javax.servlet.jsp.*;
10   import javax.servlet.jsp.tagext.*;
11
12   public class WelcomeTagHandler extends TagSupport {
13
14      // Method called to begin tag processing
```

```
15      public int doStartTag() throws JspException
16      {
17         // attempt tag processing
18         try {
19            // obtain JspWriter to output content
20            JspWriter out = pageContext.getOut();
21
22            // output content
23            out.print( "Welcome to JSP Tag Libraries!" );
24         }
25
26         // rethrow IOException to JSP container as JspException
27         catch( IOException ioException ) {
28            throw new JspException( ioException.getMessage() );
29         }
30
31         return SKIP_BODY;  // ignore the tag's body
32      }
33   }
```

Fig. 10.32 Custom tag library **descriptor** file **advjhtp1-taglib.tld**.

```
1    <?xml version = "1.0" encoding = "ISO-8859-1" ?>
2    <!DOCTYPE taglib PUBLIC
3       "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.1//EN"
4       "http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd">
5
6    <!-- a tag library descriptor -->
7
8    <taglib>
9       <tlibversion>1.0</tlibversion>
10      <jspversion>1.1</jspversion>
11      <shortname>advjhtp1</shortname>
12
13      <info>
14         A simple tab library for the examples
15      </info>
16
17      <!-- A simple tag that outputs content -->
18      <tag>
19         <name>welcome</name>
20
21         <tagclass>
22            com.deitel.advjhtp1.jsp.taglibrary.WelcomeTagHandler
23         </tagclass>
24
25         <bodycontent>empty</bodycontent>
26
27         <info>
28            Inserts content welcoming user to tag libraries
29         </info>
30      </tag>
31   </taglib>
```

1. **`name`** whose body defines the base tag name to which the prefix of taglib directive will be attached
   `<name>progName</name>`
2. **`tagclass`** fully qualified class name of the tag handler (in JSP 1.2 it is renamed to tag-class)
3. **`bodycontent`** –can be omitted but if present have to have `empty` value.
4. **`info`** –short description

Custom Tag with Attributes

XHTML and JSP elements

Use attributes to customize functionality

e.g. XHTML element can specify a **style** attribute

Specify attributes for custom tags

Use set and get methods

Developers provide methods that enable the JSP container to set/get the attributes

Fig. 10.33 Specifying attributes for a custom tag

```
1    <?xml version = "1.0"?>
2    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5    <!-- Fig. 10.33: customTagAttribute.jsp             -->
6    <!-- JSP that uses a custom tag to output content. -->
7
8    <%-- taglib directive --%>
9    <%@ taglib uri = "advjhtp1-taglib.tld" prefix = "advjhtp1" %>
10
11   <html xmlns = "http://www.w3.org/1999/xhtml">
12
13      <head>
14         <title>Specifying Custom Tag Attributes</title>
15      </head>
16
17      <body>
18         <p>Demonstrating an attribute with a string value</p>
19         <h1>
20            <advjhtp1:welcome2 firstName = "Paul" />
21         </h1>
22
23         <p>Demonstrating an attribute with an expression value</p>
24         <h1>
25            <%-- scriptlet to obtain "name" request parameter --%>
26            <%
27               String name = request.getParameter( "name" );
28            %>
29
30            <advjhtp1:welcome2 firstName = "<%= name %>" />
31         </h1>
32      </body>
33
34   </html>
```

Fig. 10.34 **Welcome2Tag-Handler** custom tag handler for a tag with an attribute

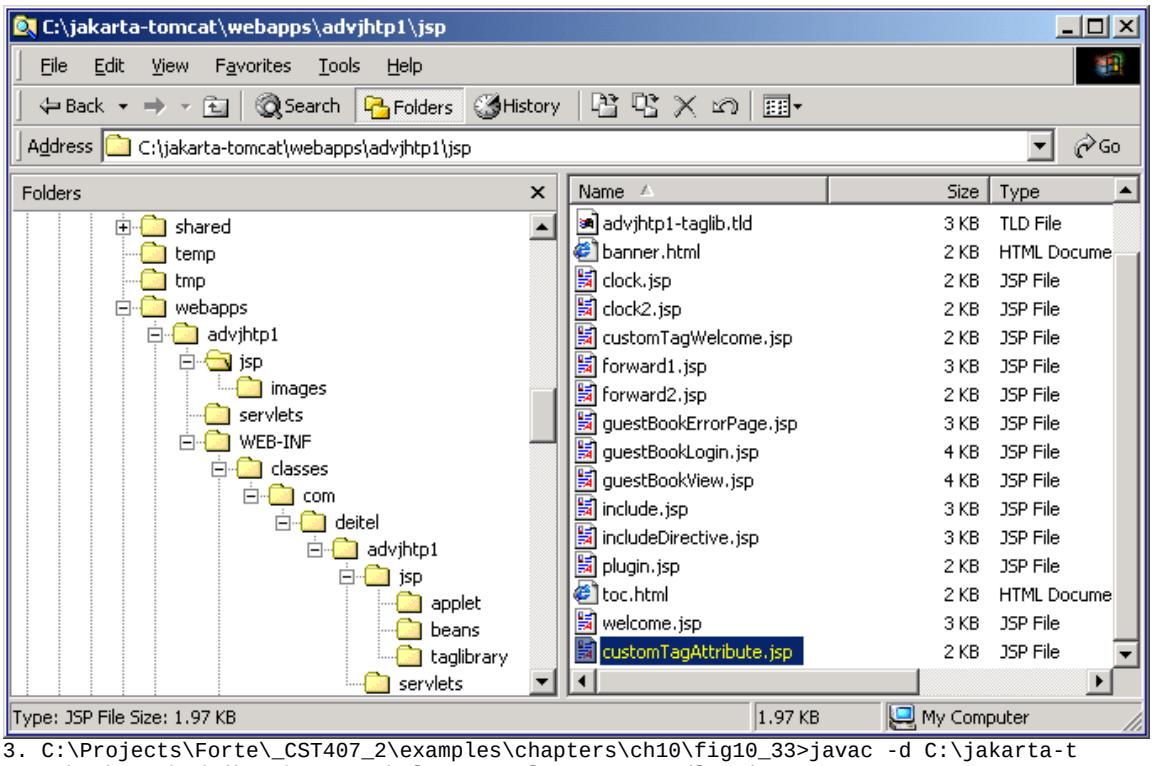
```
1    // Fig. 10.34: Welcome2TagHandler.java
2    // Custom tag handler that handles a simple tag.
3    package com.deitel.advjhtp1.jsp.taglibrary;
4
5    // Java core packages
6    import java.io.*;
7
8    // Java extension packages
9    import javax.servlet.jsp.*;
10   import javax.servlet.jsp.tagext.*;
11
12   public class Welcome2TagHandler extends TagSupport {
13      private String firstName = "";
14
15      // Method called to begin tag processing
```

```
16      public int doStartTag() throws JspException
17      {
18         // attempt tag processing
19         try {
20            // obtain JspWriter to output content
21            JspWriter out = pageContext.getOut();
22
23            // output content
24            out.print( "Hello " + firstName +
25               ", <br />Welcome to JSP Tag Libraries!" );
26         }
27
28         // rethrow IOException to JSP container as JspException
29         catch( IOException ioException ) {
30            throw new JspException( ioException.getMessage() );
31         }
32
33         return SKIP_BODY;  // ignore the tag's body
34      }
35
36      // set firstName attribute to the users first name
37      public void setFirstName( String username )
38      {
39         firstName = username;
40      }
41   }
```

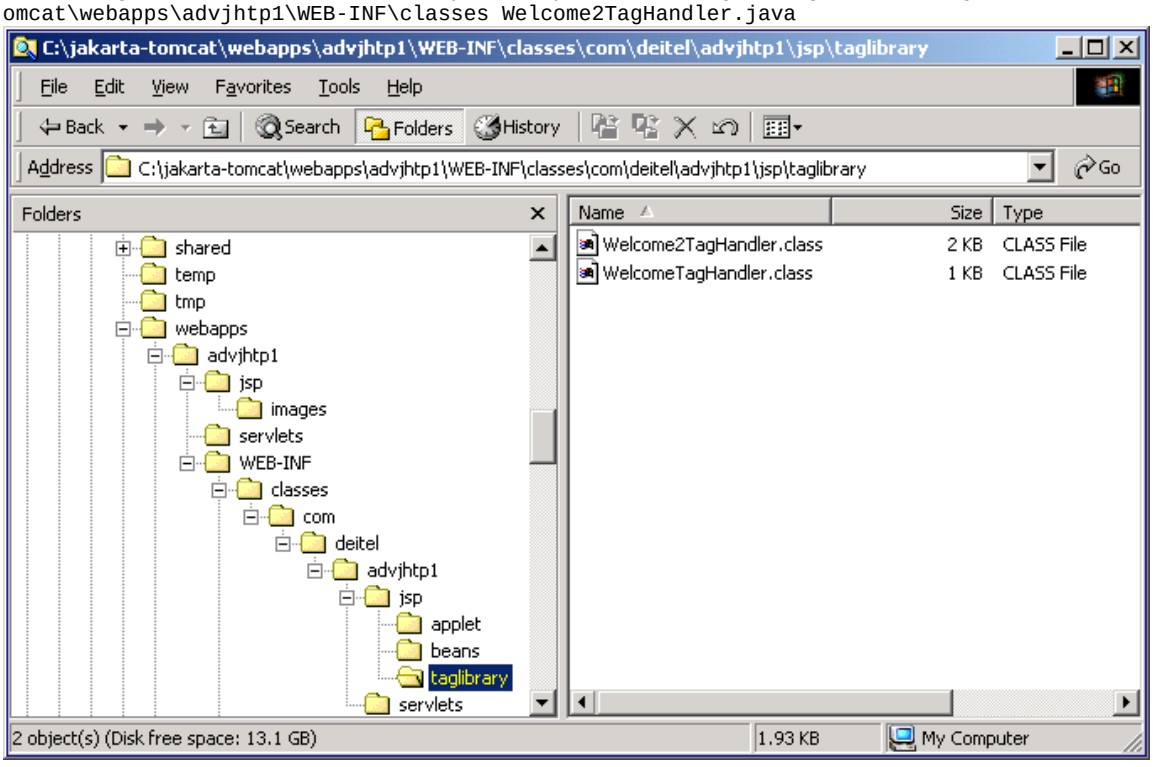
Fig. 10.35 Element **tag** for the **welcome2** custom tag.
Makes the JSP container aware of the tag, by adding it to a tag library

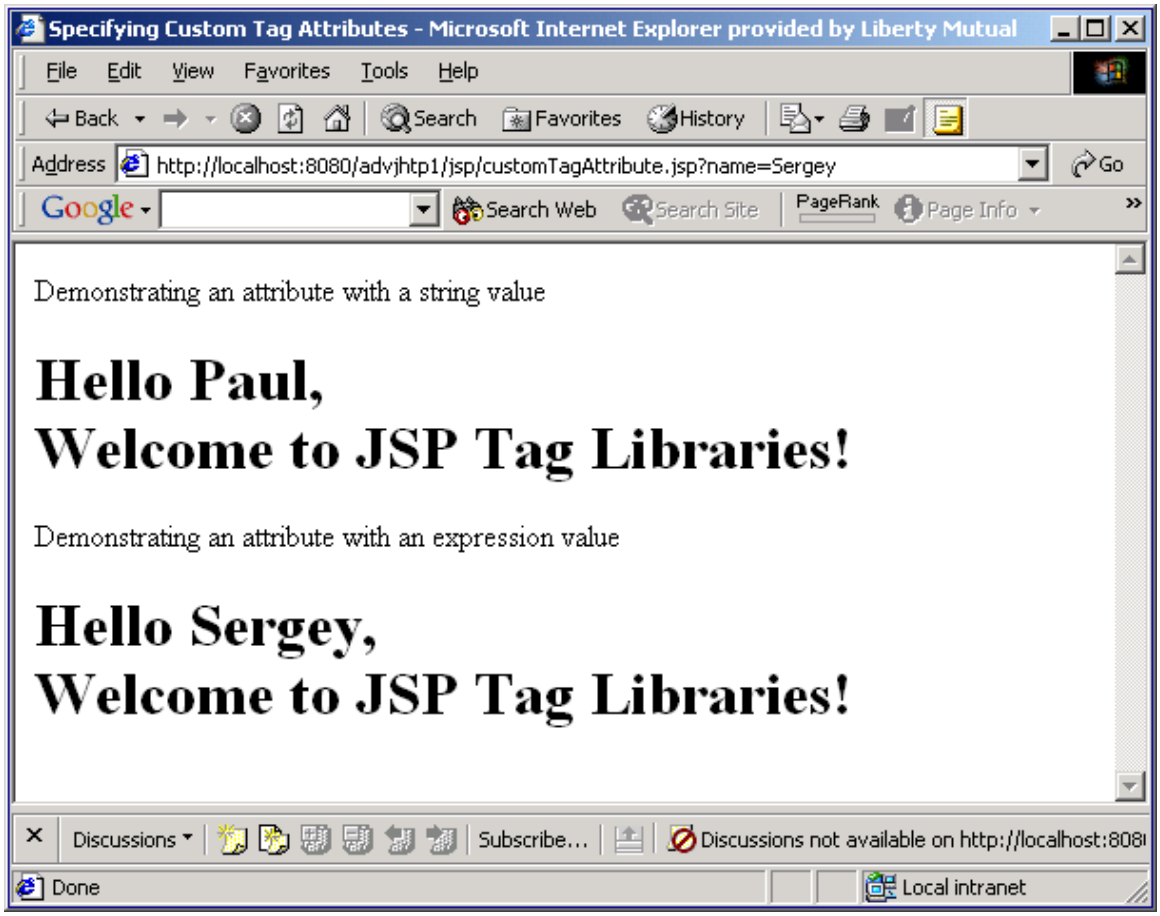
```
1    <!-- A tag with an attribute -->
2    <tag>
3       <name>welcome2</name>
4
5       <tagclass>
6          com.deitel.advjhtp1.jsp.taglibrary.Welcome2TagHandler
7       </tagclass>
8
9       <bodycontent>empty</bodycontent>
10
11      <info>
12         Inserts content welcoming user to tag libraries. Uses
13         attribute "name" to insert the user's name.
14      </info>
15
16      <attribute>
17         <name>firstName</name>
18         <required>true</required>
19         <rtexprvalue>true</rtexprvalue>
20      </attribute>
21   </tag>
```

```
1. C:\Projects\Forte\_CST407_2\examples\chapters\ch10\fig10_33>copy advjhtp1-taglib.
tld C:\jakarta-tomcat\webapps\advjhtp1\jsp
Overwrite C:\jakarta-tomcat\webapps\advjhtp1\jsp\advjhtp1-taglib.tld? (Yes/No/All): y
        1 file(s) copied.

2. C:\Projects\Forte\_CST407_2\examples\chapters\ch10\fig10_33>copy customTagAttribu
te.jsp C:\jakarta-tomcat\webapps\advjhtp1\jsp
        1 file(s) copied.
```



3. C:\Projects\Forte_CST407_2\examples\chapters\ch10\fig10_33>javac -d C:\jakarta-tomcat\webapps\advjhtp1\WEB-INF\classes Welcome2TagHandler.java





Note: If you see xml document instead of above picture replace

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

with

```
      <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.1 Transitional//EN">
```

Evaluating the Body of a Custom Tag

Particularly useful for processing element body. When custom tag interacts with element body, additional methods are required to perform those interactions.

Methods defined in `BodyTagSupport (has additional methods: doInitBody (called once), doAfterBody.`

Example replacing JavaBean for guestlist.

- The custom tag handler places the variables in the JSP's PageContext, so the variables can be used throughout the page.
- Without repletion, the custom tag handler is defined to iterate over all the guest in the guestbook db

Fig. 10.36 Using a custom tag that interacts with its body

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- customTagBody.jsp -->

<%-- taglib directive --%>
<%@ taglib uri = "advjhtp1-taglib.tld" prefix = "advjhtp1" %>

<html xmlns = "http://www.w3.org/1999/xhtml">

   <head>
      <title>Guest List</title>

      <style type = "text/css">
         body {
            font-family: tahoma, helvetica, arial, sans-serif
         }

         table, tr, td, th {
            text-align: center;
            font-size: .9em;
            border: 3px groove;
            padding: 5px;
            background-color: #dddddd
         }
      </style>
   </head>

   <body>
      <p style = "font-size: 2em">Guest List</p>

      <table>
         <thead>
            <th style = "width: 100px">Last name</th>
            <th style = "width: 100px">First name</th>
            <th style = "width: 200px">Email</th>
         </thead>

         <%-- guestlist custom tag --%>
         <advjhtp1:guestlist>
            <tr>
               <td><%= lastName %></td>

               <td><%= firstName %></td>

               <td>
                  <a href = "mailto:<%= email %>">
                     <%= email %></a>
               </td>
            </tr>
         </advjhtp1:guestlist>
      </table>
   </body>

</html>
```

```
// Fig. 10.37: GuestBookTag.java
// Custom tag handler that reads information from the guestbook
// database and makes that data available in a JSP.
package com.deitel.advjhtp1.jsp.taglibrary;

// Java core packages
import java.io.*;
```

```
8    import java.util.*;
9
10   // Java extension packages
11   import javax.servlet.jsp.*;
12   import javax.servlet.jsp.tagext.*;
13
14   // Deitel packages
15   import com.deitel.advjhtp1.jsp.beans.*;
16
17   public class GuestBookTag extends BodyTagSupport {
18      private String firstName;
19      private String lastName;
20      private String email;
21
22      private GuestDataBean guestData;
23      private GuestBean guest;
24      private Iterator iterator;
25
26      // Method called to begin tag processing
27      public int doStartTag() throws JspException
28      {
29         // attempt tag processing
30         try {
31            guestData = new GuestDataBean();
32
33            List list = guestData.getGuestList();
34            iterator = list.iterator();
35
36            if ( iterator.hasNext() ) {
37               processNextGuest();
38
39               return EVAL_BODY_TAG;    // continue body processing
40            }
41            else
42               return SKIP_BODY;        // terminate body processing
43         }
44
45         // if any exceptions occur, do not continue processing
46         // tag's body
47         catch( Exception exception ) {
48            exception.printStackTrace();
49            return SKIP_BODY;   // ignore the tag's body
50         }
51      }
52
53      // process body and determine if body processing
54      // should continue
55      public int doAfterBody()
56      {
57         // attempt to output body data
58         try {
59            bodyContent.writeOut( getPreviousOut() );
60         }
61
62         // if exception occurs, terminate body processing
63         catch ( IOException ioException ) {
64            ioException.printStackTrace();
65            return SKIP_BODY;        // terminate body processing
66         }
67
68         bodyContent.clearBody();
69
70         if ( iterator.hasNext() ) {
71            processNextGuest();
72
73            return EVAL_BODY_TAG;    // continue body processing
74         }
75         else
76            return SKIP_BODY;        // terminate body processing
77      }
78
```

```
79      // obtains the next GuestBean and extracts its data
80      private void processNextGuest()
81      {
82         // get next guest
83         guest = ( GuestBean ) iterator.next();
84
85         pageContext.setAttribute(
86            "firstName", guest.getFirstName() );
87
88         pageContext.setAttribute(
89            "lastName", guest.getLastName() );
90
91         pageContext.setAttribute(
92            "email", guest.getEmail() );
93      }
94   }
```

Fig. 10.38 GuestBookTag-ExtraInfo used by the container **to define scripting variables in a JSP that uses the guestlist custom tag.**

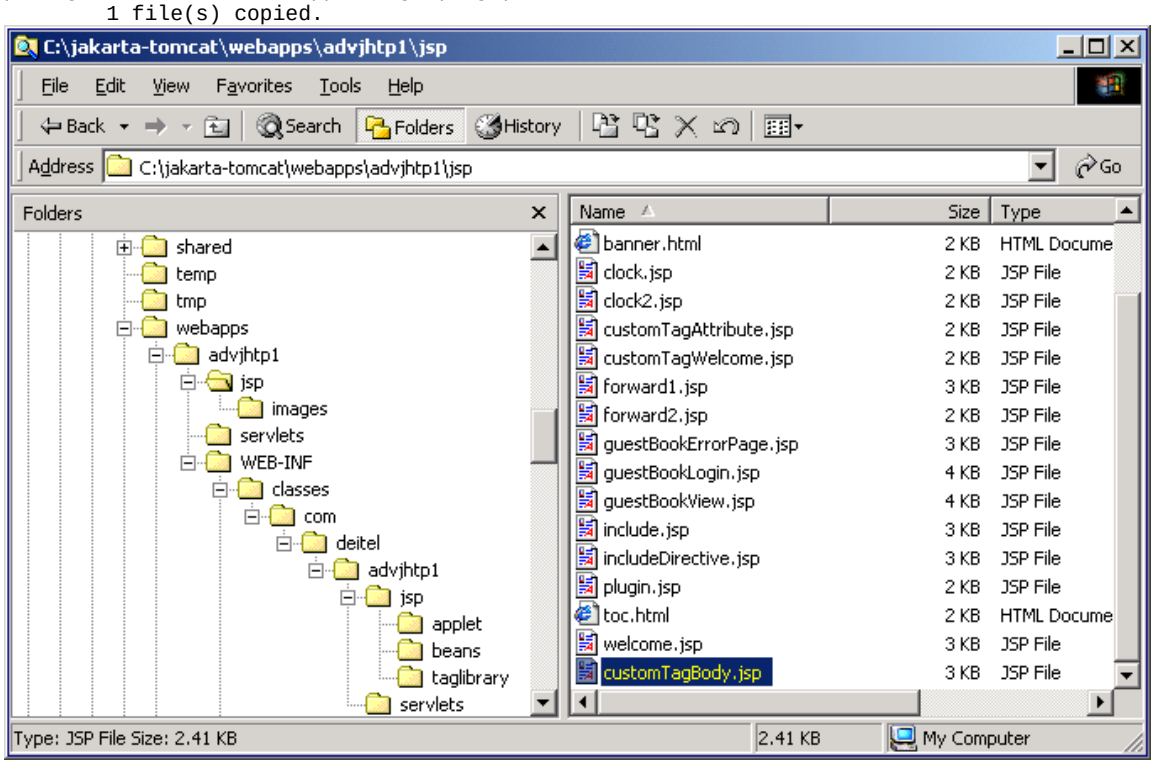
```
1    // Fig. 10.38: GuestBookTagExtraInfo.java
2    // Class that defines the variable names and types created by
3    // custom tag handler GuestBookTag.
4    package com.deitel.advjhtp1.jsp.taglibrary;
5
6    // Java core packages
7    import javax.servlet.jsp.tagext.*;
8
9    public class GuestBookTagExtraInfo extends TagExtraInfo {
10
11      // method that returns information about the variables
12      // GuestBookTag creates for use in a JSP
13      public VariableInfo [] getVariableInfo( TagData tagData )
14      {
15         VariableInfo firstName = new VariableInfo( "firstName",
16            "String", true, VariableInfo.NESTED );
17
18         VariableInfo lastName = new VariableInfo( "lastName",
19            "String", true, VariableInfo.NESTED );
20
21         VariableInfo email = new VariableInfo( "email",
22            "String", true, VariableInfo.NESTED );
23
24         VariableInfo varableInfo [] =
25            { firstName, lastName, email };
26
27         return varableInfo;
28      }
29   }
```

Fig. 10.39 Element tag for the guest-list custom tag.

```
1    <!-- A tag that iterates over an ArrayList of GuestBean -->
2    <!-- objects, so they can be output in a JSP            -->
3    <tag>
4       <name>guestlist</name>
5
6       <tagclass>
7          com.deitel.advjhtp1.jsp.taglibrary.GuestBookTag
8       </tagclass>
9
10      <teiclass>
11         com.deitel.advjhtp1.jsp.taglibrary.GuestBookTagExtraInfo
12      </teiclass>
13
14      <bodycontent>JSP</bodycontent>
15
16      <info>
17         Iterates over a list of GuestBean objects
```

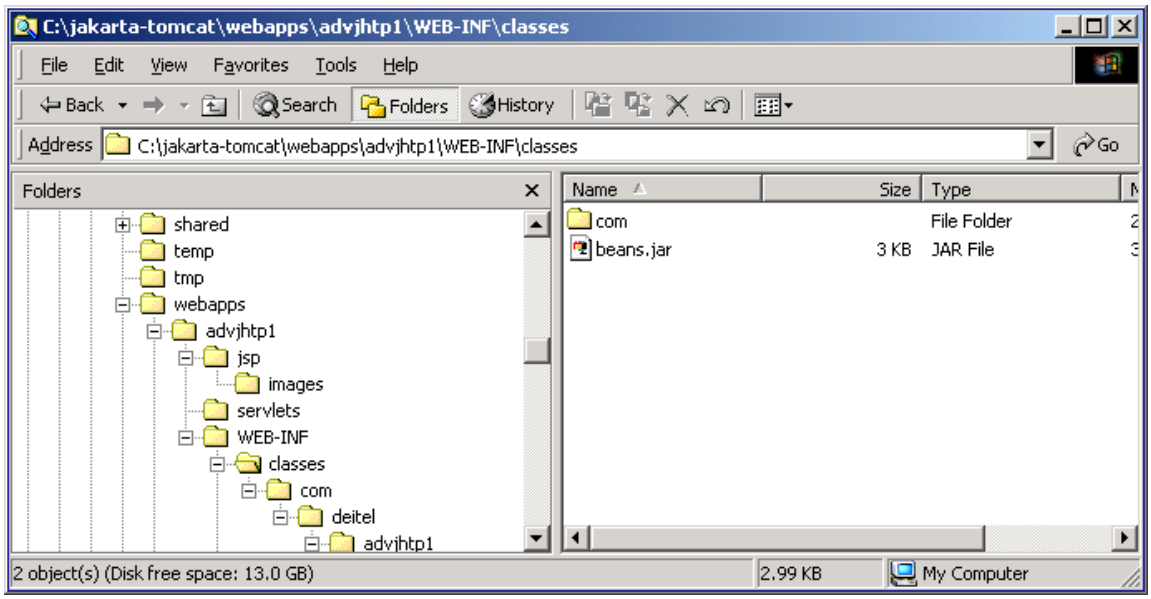
```
18      </info>
19   </tag>
```

```
1. C:\Projects\Forte\_CST407_2\examples\chapters\ch10\fig10_36>copy advjhtp1-taglib.
tld C:\jakarta-tomcat\webapps\advjhtp1\jsp
Overwrite C:\jakarta-tomcat\webapps\advjhtp1
\jsp\advjhtp1-taglib.tld? (Yes/No/All): y
        1 file(s) copied.

2. C:\Projects\Forte\_CST407_2\examples\chapters\ch10\fig10_36>copy customTagBody.js
p C:\jakarta-tomcat\webapps\advjhtp1\jsp
        1 file(s) copied.
```

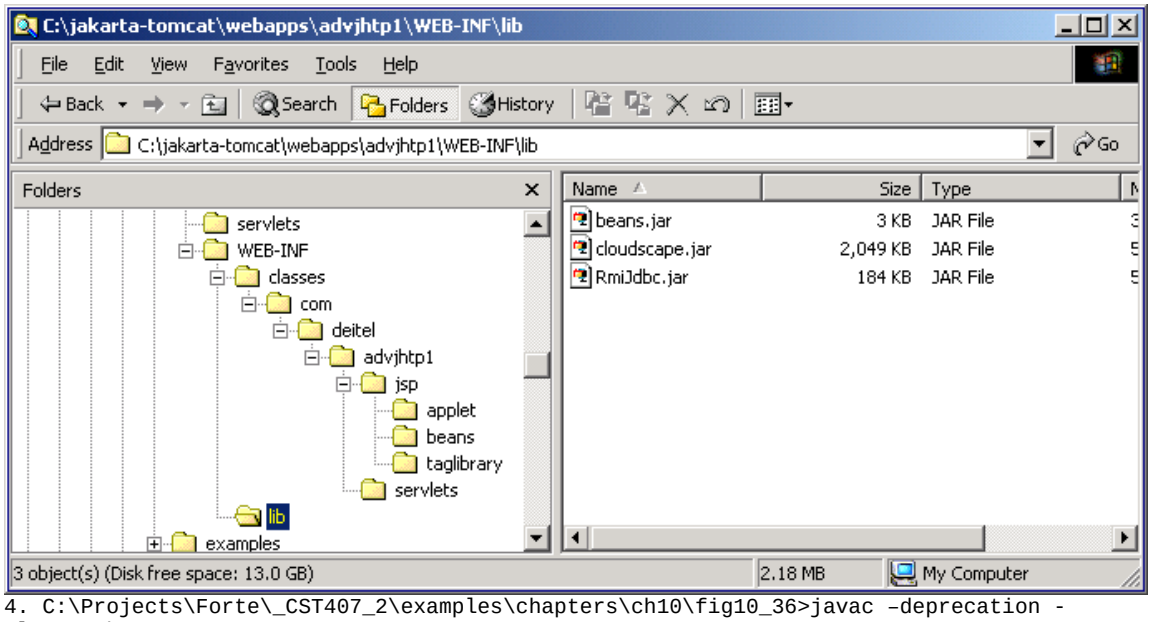


3. Create jar (missing step).

```
C:\jakarta-tomcat\webapps\advjhtp1\WEB-INF\classes>jar cvf beans.jar
com\deitel\advjhtp1\jsp\beans\*.*
added manifest
adding: com/deitel/advjhtp1/jsp/beans/GuestBean.class(in = 728) (out= 364)(defla
ted 50%)
adding: com/deitel/advjhtp1/jsp/beans/GuestDataBean.class(in = 2131) (out= 1150)
(deflated 46%)
adding: com/deitel/advjhtp1/jsp/beans/Rotator.class(in = 1127) (out= 576)(deflat
ed 48%)
```



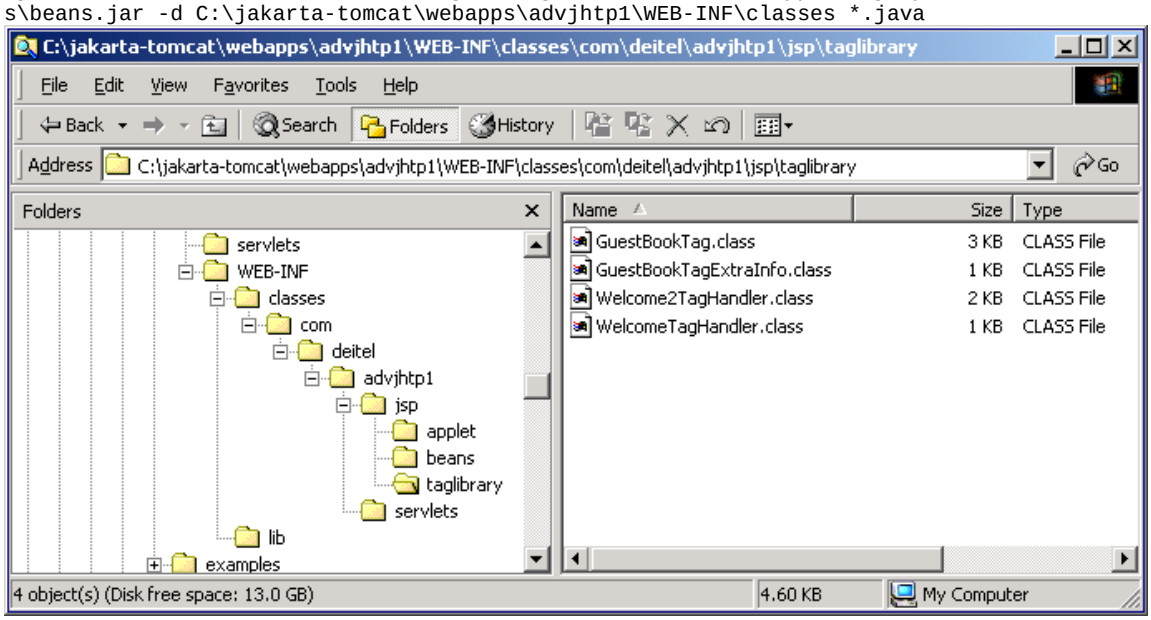
or better put beans.jar to



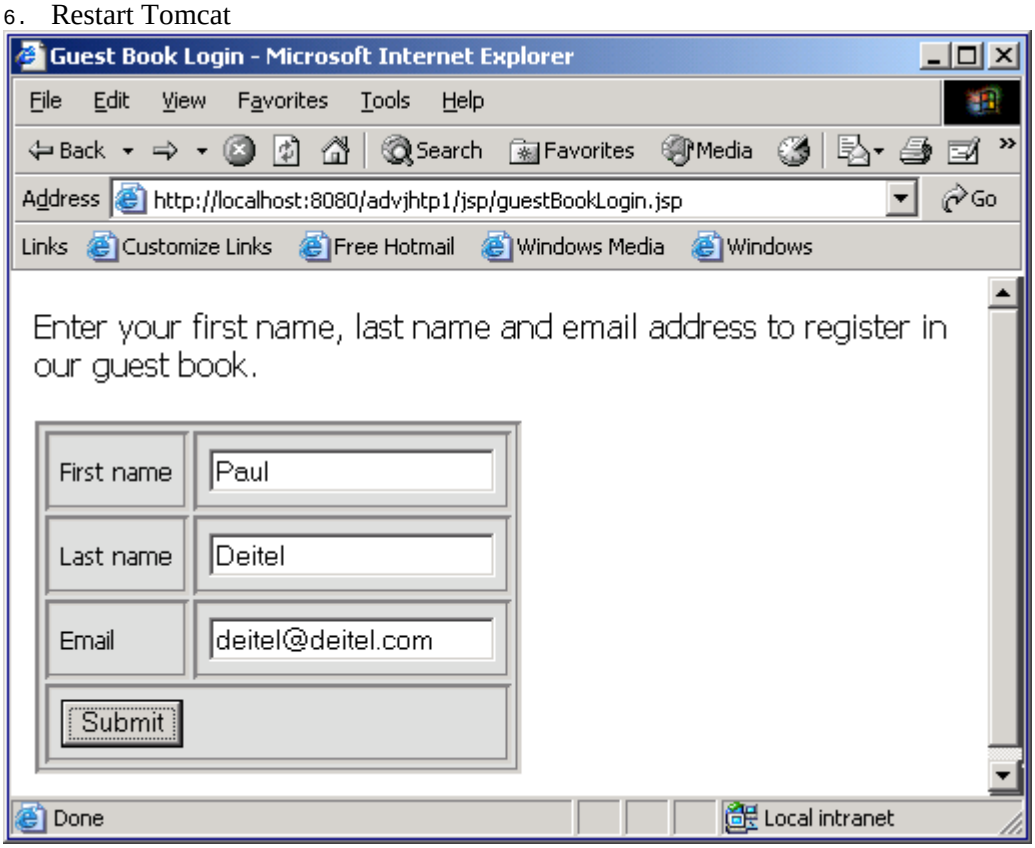

```
4. C:\Projects\Forte\_CST407_2\examples\chapters\ch10\fig10_36>javac –deprecation -
classpath .;C:
\jakarta-tomcat\common\lib\servlet.jar;C:\jakarta-tomcat\webapps\advjhtp1\WEB-INF\classe
s\beans.jar -d C:\jakarta-tomcat\webapps\advjhtp1\WEB-INF\classes *.java
GuestBookTag.java:41: warning: EVAL_BODY_TAG in javax.servlet.jsp.tagext.BodyTag
 has been deprecated
           return EVAL_BODY_TAG;    // continue body processing
                  ^
GuestBookTag.java:75: warning: EVAL_BODY_TAG in javax.servlet.jsp.tagext.BodyTag
 has been deprecated
         return EVAL_BODY_TAG;    // continue body processing
                ^
2 warnings
```

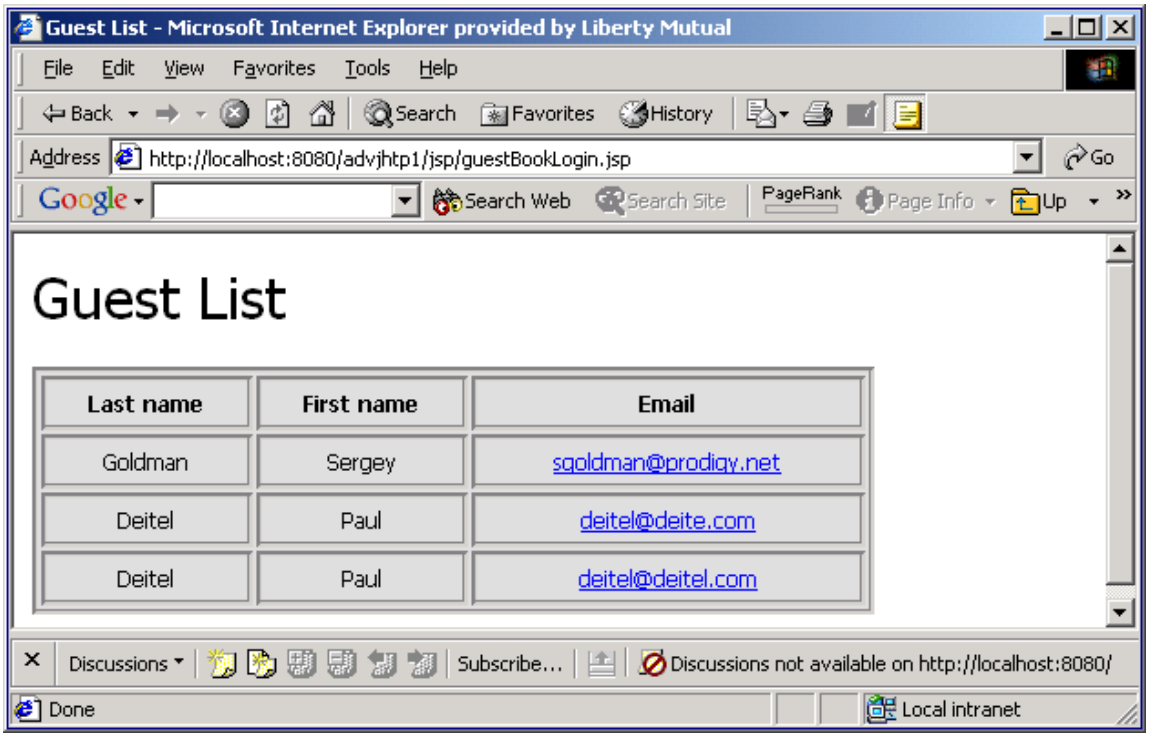
To get rid of deprecated warnings replace EVAL_BODY_TAG with BodyTag.EVAL_BODY_BUFFERED in GuestBookTag.java (lines 41 and 75)

```
C:\Projects\Forte\_CST407_2\examples\chapters\ch10\fig10_36>
C:\Projects\Forte\_CST407_2\examples\chapters\ch10\fig10_36>javac -classpath .;C:
\jakarta-tomcat\common\lib\servlet.jar;C:\jakarta-tomcat\webapps\advjhtp1\WEB-INF\classe
s\beans.jar -d C:\jakarta-tomcat\webapps\advjhtp1\WEB-INF\classes *.java
```



5. Start Cloudscape
6. Restart Tomcat





MVC
Servlet/JSP

Struts http://jakarta.apache.org/struts

The key to letting servlets forward requests or include content is to use a `RequestDispatcher`.

http://yourhost/presentations/presentation1.jsp

```
String url= "/presentation/presentation1.jsp
RequestDispatcher  dispatcher= getServeletContext().setRequestDispatcher(url);
```

Than use `forward(HttpServletRequest request, HttpServletResponse response)`

```
package com.cst407.jsps;

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

/** Top-level travel-processing servlet. This servlet sets up
 *  the customer data as a bean, then forwards the request
 *  to the airline booking page, the rental car reservation
 *  page, the hotel page, the existing account modification
 *  page, or the new account page.
 *  <P>
 *  Taken from More Servlets and JavaServer Pages
```

```
 *  from Prentice Hall and Sun Microsystems Press,
 *  http://www.moreservlets.com/.
 *  &copy; 2002 Marty Hall; may be freely used or adapted.
 */

public class Travel extends HttpServlet {

    private TravelCustomer[] travelData;

    public void init() {
        travelData= TravelData.getTravelData();
    }

    /** Since password is being sent, use POST only. However,
    *  the use of POST means that you cannot forward
    *  the request to a static HTML page, since the forwarded
    *  request uses the same request method as the original
    *  one, and static pages cannot handle POST. Solution:
    *  have the "static" page be a JSP file that contains
    *  HTML only. That's what accounts.jsp is. The other
    *  JSP files really need to be dynamically generated,
    *  since they make use of the customer data.
    *
    */

    public void doPost(HttpServletRequest request, HttpServletResponse response)
                                       throws ServletException, IOException {

        String emailAddress= request.getParameter("emailAddress");
        String password= request.getParameter("password");
        TravelCustomer customer= TravelCustomer.findCustomer(emailAddress,
travelData);

        if ((customer == null) || (password == null) ||
            (!password.equals(customer.getPassword()))) {
             gotoPage("/jsp/travel/accounts.jsp", request, response);
        }

        /* The methods that use the following parameters will
           check for missing or malformed values.
        */
        customer.setStartDate(request.getParameter("startDate"));
        customer.setEndDate(request.getParameter("endDate"));
        customer.setOrigin(request.getParameter("origin"));
        customer.setDestination(request.getParameter("destination"));

        HttpSession session= request.getSession(true);

        session.setAttribute("customer", customer);

        if (request.getParameter("flights") != null) {
            gotoPage("/jsp/travel/BookFlights.jsp", request, response);
        }
        else if (request.getParameter("cars") != null) {
            gotoPage("/jsp/travel/RentCars.jsp", request, response);
        }
        else if (request.getParameter("hotels") != null) {
            gotoPage("/jsp/travel/FindHotels.jsp", request, response);
        }
        else if (request.getParameter("account") != null) {
            gotoPage("/jsp/travel/EditAccounts.jsp", request, response);
        }
        else {
            gotoPage("/jsp/travel/IllegalRequest.jsp", request, response);
        }
    }

    /*-------------------------------------------*/
    private void gotoPage(String address, HttpServletRequest request,
                                          HttpServletResponse response)
                                  throws ServletException, IOException {
```

```
        RequestDispatcher dispatcher=
                    getServletContext().getRequestDispatcher(address);
        dispatcher.forward(request, response);
    }
}
/*-------------------------------------------*/
/*-------------------------------------------*/
/*-------------------------------------------*/
```

Usually forward to a JSP page or another servlet.

In some cases forward to static HTML.

GET – forwarding to HTML is legal, no need for a special syntax.

POST – Since original and forwarded request use the same request method, POST cannot be forwarded to HTML (it cannot handle POST).

SOLUTION: rename somefile.html to somefile.jsp. GET is still OK, but POST can be handled as well.

Supplying Information to Destination Page

Servlet can store data for JSP in 3 main places:

1**. HttpServletRequest**

non-default value of the scope attribute of jsp:useBean - request

Storing data that servlet looked up and that JSP will use only in this request

- Servlet:
  ```
  AClass val= new AClass(…);
  request.setAttribute(“key”, val);
  ```
- than forward to JSP
- JSP:
  ```
  <jsp:useBean id= “key”, class=”AClass”, scope= “request” />
  ```

2. **HttpSession**

non-default values of the scope attribute jsp:useBean – session

Storing data that servlet looked up and that JSP will in this request and in later requests from the SAME client.

- Servlet:
  ```
  AClass val= new AClass(…);
  HttpSession session= request.getSession(true);
  Session.setAttribute(“key”, val);
  ```
- than forward to JSP
- JSP:
  ```
  <jsp:useBean id= “key”, class=”AClass”, scope= “session” />
  ```

3. **ServletContext**

non-default value of the scope attribute jsp:useBean - application

Storing data that servlet looked up and that JSP will in this request and in later requests from ANY client.

- Servlet:
```
AClass val= new AClass(…);
getServletContext.setAttribute(“key”, val);
```
- than forward to JSP
- JSP:
```
<jsp:useBean id= “key”, class=”AClass”, scope= “application” />
```

HttpServletResponse sendRedirect vs RequestDispatcher forward

1 `sendRedirect` - requires the client to reconnect to the new resource
`forward` - handled completely on the server
2 `sendRedirect` - **does not** automatically preserve all request data
`forward` - **does** automatically preserve all request data
3 `sendRedirect` - results in a different final URL
`forward` – URL of original request is maintained

If destination page uses relative URLs for images or stylesheets, it needs to make them relative to the server root, not to destination page’s actual location.
<LINK REL=STYLESHEET HREF=”my-styles.css TYPE=”text/css”>
If the JSP page containing this entry is accessed by means of a forwarded request. my-styles.css will be interpreted relative to the URL of the originating servlet, not relative to the JSP page itself – ERROR.

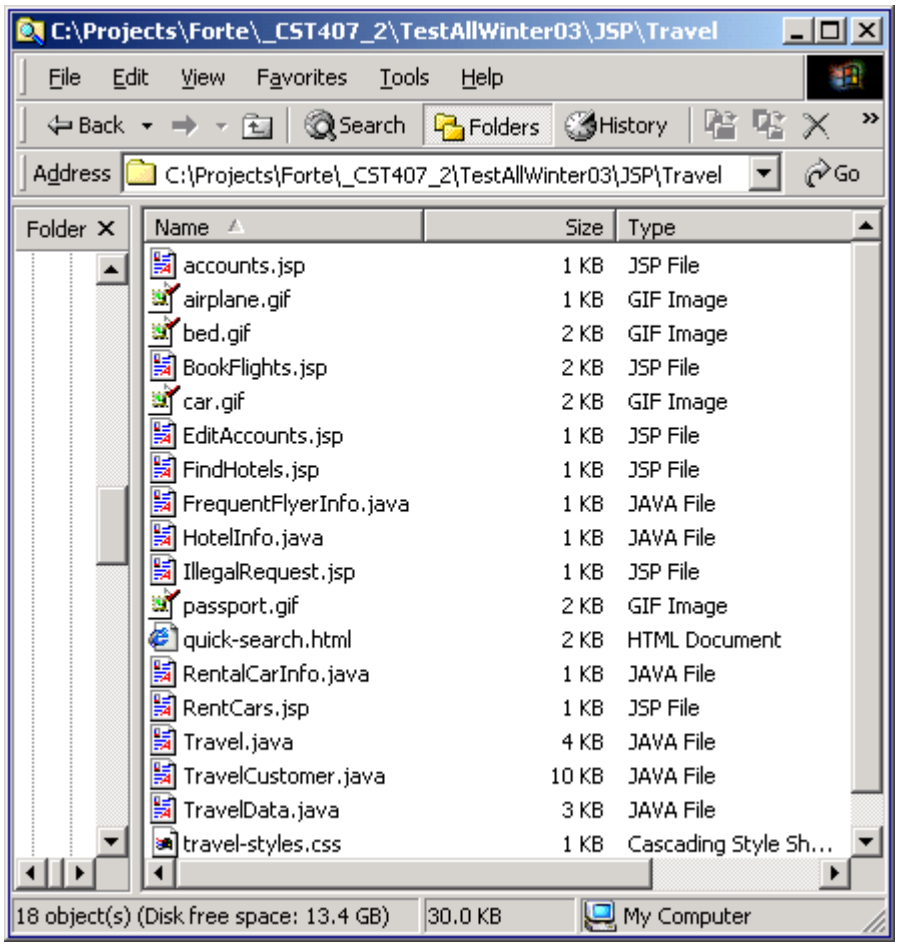
Solution:
<LINK REL=STYLESHEET HREF=”/path/my-styles.css TYPE=”text/css”>

Servlet 2.3, 2.3 can use
getNameDispatcher

Travel Agent Example

- quick-search.html. Front end to the online travel agency. Requires four images (airplane.gif, bed.gif, car.gif, and passport.gif) and the travel-styles.css style sheet. Links to the accounts.jsp placeholder.
- Travel.java. Top-level travel-processing servlet. This servlet sets up the customer data as a bean, then forwards the request to the airline booking page, the rental car reservation page, the hotel page, the existing account modification page, or the new account page.
- BookFlights.jsp. Display page for the plane flights part of the travel-processing servlet. Uses the travel-styles.css style sheet. There are also several placeholder pages: RentCars.jsp, FindHotels.jsp, EditAccounts.jsp, and IllegalRequest.jsp.
- TravelCustomer.java. Describes a travel services customer. Implemented as a bean with methods that return data in HTML format, suitable for access from JSP. Uses the FrequentFlyerInfo.java, HotelInfo.java, RentalCarInfo.java, and TravelData.java classes.

•



Travel agent has a search page. Agents enter email, password to associate the request with their previously established account. Also they enter trip origin, destination, start data, end date.

Buttons:
"Book Flights" shows list of available flights on the date specified ordered by price
This page shows real users' name, frequent flyer information, credit card number.

"Edit Account" show previously entered customer info

"Rent Cars" or "Find Hotels" will share much of the same customer data but different presentation.

Quick-search.html – top level

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!--
Front end to travel servlet.

Taken from More Servlets and JavaServer Pages
from Prentice Hall and Sun Microsystems Press,
http://www.moreservlets.com/.
(C) 2002 Marty Hall; may be freely used or adapted.
```

```
-->
<HTML>
<HEAD>
  <TITLE>Online Travel Quick Search</TITLE>
  <LINK REL=STYLESHEET
        HREF="travel-styles.css"
        TYPE="text/css">
</HEAD>
<BODY>
<BR>
<H1>Online Travel Quick Search</H1>
<FORM ACTION="/servlet/com.cst407.jsps.travel.Travel" METHOD="POST">
<CENTER>
Email address: <INPUT TYPE="TEXT" NAME="emailAddress"><BR>
Password: <INPUT TYPE="PASSWORD" NAME="password" SIZE=10><BR>
Origin: <INPUT TYPE="TEXT" NAME="origin"><BR>
Destination: <INPUT TYPE="TEXT" NAME="destination"><BR>
Start date (MM/DD/YY):
  <INPUT TYPE="TEXT" NAME="startDate" SIZE=8><BR>
End date (MM/DD/YY):
  <INPUT TYPE="TEXT" NAME="endDate" SIZE=8><BR>
<P>
<TABLE CELLSPACING=1>
<TR>
  <TH> <IMG SRC="airplane.gif" WIDTH=100 HEIGHT=29
                 ALIGN="TOP" ALT="Book Flight"> 
  <TH> <IMG SRC="car.gif" WIDTH=100 HEIGHT=31
                 ALIGN="MIDDLE" ALT="Rent Car"> 
  <TH> <IMG SRC="bed.gif" WIDTH=100 HEIGHT=85
                 ALIGN="MIDDLE" ALT="Find Hotel"> 
  <TH> <IMG SRC="passport.gif" WIDTH=71 HEIGHT=100
                 ALIGN="MIDDLE" ALT="Edit Account"> 
<TR>
  <TH><SMALL>
      <INPUT TYPE="SUBMIT" NAME="flights" VALUE="Book Flight">
      </SMALL>
  <TH><SMALL>
      <INPUT TYPE="SUBMIT" NAME="cars" VALUE="Rent Car">
      </SMALL>
  <TH><SMALL>
      <INPUT TYPE="SUBMIT" NAME="hotels" VALUE="Find Hotel">
      </SMALL>
  <TH><SMALL>
      <INPUT TYPE="SUBMIT" NAME="account" VALUE="Edit Account">
      </SMALL>
</TABLE>
</CENTER>
</FORM>
<BR>
<P ALIGN="CENTER">
<B>Not yet a member? Get a free account
<A HREF="accounts.jsp">here</A>.</B></P>
</BODY>
</HTML>
```

com.CST407.jsps.travel.Travel -

1. Look for customer info (hard-coded here, but in reality it will us DB),
2. puts info in HttpSession associating the value (type com.cst407.jsps.travel.TravelCustomer) with name customer
3.forwards the request to a Different JSP page corresponding to each of the possible actions

Destination page **BookFlights.jsp**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!--
Flight-finding page for travel example.

Taken from More Servlets and JavaServer Pages
from Prentice Hall and Sun Microsystems Press,
```

```
http://www.moreservlets.com/.
(C) 2002 Marty Hall; may be freely used or adapted.
-->
<HTML>
<HEAD>
  <TITLE>Best Available Flights</TITLE>
  <LINK REL=STYLESHEET
        HREF="/jsp/travel/travel-styles.css"
        TYPE="text/css">
</HEAD>
<BODY>
<H1>Best Available Flights</H1>
<CENTER>
<jsp:useBean id="customer"
             class="com.cst407.jsps.travel.TravelCustomer"
             scope="session" />
Finding flights for
<jsp:getProperty name="customer" property="fullName" />
<P>
<jsp:getProperty name="customer" property="flights" />
<P><BR><HR><BR>
<FORM ACTION="/servlet/BookFlight">
<jsp:getProperty name="customer"
                 property="frequentFlyerTable" />
<P>
<B>Credit Card:</B>
<jsp:getProperty name="customer" property="creditCard" />
<P>
<INPUT TYPE="SUBMIT" NAME="holdButton" VALUE="Hold for 24 Hrs">
<P>
<INPUT TYPE="SUBMIT" NAME="bookItButton" VALUE="Book It!">
</FORM>
</CENTER>
</BODY>
</HTML>
```

TravelCustomer.java

```
package com.cst407.jsps.travel;

import java.util.*;
import java.text.*;

/** Describes a travel services customer. Implemented
 *  as a bean with some methods that return data in HTML
 *  format, suitable for access from JSP.
 *  <P>
 *  Taken from More Servlets and JavaServer Pages
 *  from Prentice Hall and Sun Microsystems Press,
 *  http://www.moreservlets.com/.
 *  &copy; 2002 Marty Hall; may be freely used or adapted.
 */

public class TravelCustomer {
    private String emailAddress, password, firstName, lastName;
    private String creditCardName, creditCardNumber;
    private String phoneNumber, homeAddress;
    private String startDate, endDate;
    private String origin, destination;
    private FrequentFlyerInfo[] frequentFlyerData;
    private RentalCarInfo[] rentalCarData;
    private HotelInfo[] hotelData;

    public TravelCustomer(String emailAddress,
                          String password,
                          String firstName,
                          String lastName,
                          String creditCardName,
                          String creditCardNumber,
                          String phoneNumber,
```

```
                    String homeAddress,
                    FrequentFlyerInfo[] frequentFlyerData,
                    RentalCarInfo[] rentalCarData,
                    HotelInfo[] hotelData) {
    setEmailAddress(emailAddress);
    setPassword(password);
    setFirstName(firstName);
    setLastName(lastName);
    setCreditCardName(creditCardName);
    setCreditCardNumber(creditCardNumber);
    setPhoneNumber(phoneNumber);
    setHomeAddress(homeAddress);
    setStartDate(startDate);
    setEndDate(endDate);
    setFrequentFlyerData(frequentFlyerData);
    setRentalCarData(rentalCarData);
    setHotelData(hotelData);
}

public String getEmailAddress() {
    return(emailAddress);
}

public void setEmailAddress(String emailAddress) {
    this.emailAddress= emailAddress;
}

public String getPassword() {
    return(password);
}

public void setPassword(String password) {
    this.password= password;
}

public String getFirstName() {
    return(firstName);
}

public void setFirstName(String firstName) {
    this.firstName= firstName;
}

public String getLastName() {
    return(lastName);
}

public void setLastName(String lastName) {
    this.lastName= lastName;
}

public String getFullName() {
    return(getFirstName() + " " + getLastName());
}

public String getCreditCardName() {
    return(creditCardName);
}

public void setCreditCardName(String creditCardName) {
    this.creditCardName= creditCardName;
}

public String getCreditCardNumber() {
    return(creditCardNumber);
}

public void setCreditCardNumber(String creditCardNumber) {
    this.creditCardNumber= creditCardNumber;
}
```

```
/*----------------------------------------*/
public String getCreditCard() {
    String cardName= getCreditCardName();
    String cardNum= getCreditCardNumber();
    cardNum= cardNum.substring(cardNum.length() - 4);

    return(cardName + " (XXXX-XXXX-XXXX-" + cardNum + ")");
}

public String getPhoneNumber() {
    return(phoneNumber);
}

public void setPhoneNumber(String phoneNumber) {
    this.phoneNumber= phoneNumber;
}

public String getHomeAddress() {
    return(homeAddress);
}

public void setHomeAddress(String homeAddress) {
    this.homeAddress= homeAddress;
}

public String getStartDate() {
    return(startDate);
}

public void setStartDate(String startDate) {
    this.startDate= startDate;
}

public String getEndDate() {
    return(endDate);
}

public void setEndDate(String endDate) {
    this.endDate= endDate;
}

public String getOrigin() {
    return(origin);
}

public void setOrigin(String origin) {
    this.origin= origin;
}

public String getDestination() {
    return(destination);
}

public void setDestination(String destination) {
    this.destination= destination;
}

public FrequentFlyerInfo[] getFrequentFlyerData() {
    return(frequentFlyerData);
}

public void setFrequentFlyerData(FrequentFlyerInfo[] frequentFlyerData) {
    this.frequentFlyerData= frequentFlyerData;
}

/*----------------------------------------*/
public String getFrequentFlyerTable() {
    FrequentFlyerInfo[] frequentFlyerData= getFrequentFlyerData();

    if (frequentFlyerData.length == 0) {
        return("<I>No frequent flyer data recorded.</I>");
```

```
    }
    else {
        String table =
            "<TABLE>\n" +
            "  <TR><TH>Airline<TH>Frequent Flyer Number\n";
        for(int i=0; i<frequentFlyerData.length; i++) {
            FrequentFlyerInfo info= frequentFlyerData[i];
            table= table +
                "<TR ALIGN=\"CENTER\">" +
                "<TD>" + info.getAirlineName() +
                "<TD>" + info.getFrequentFlyerNumber() + "\n";
        }
        table= table + "</TABLE>\n";

        return(table);
    }
}

/*----------------------------------------*/
public RentalCarInfo[] getRentalCarData() {
    return(rentalCarData);
}

public void setRentalCarData(RentalCarInfo[] rentalCarData) {
    this.rentalCarData= rentalCarData;
}

public HotelInfo[] getHotelData() {
    return(hotelData);
}

public void setHotelData(HotelInfo[] hotelData) {
    this.hotelData= hotelData;
}

/*----------------------------------------*/
/* This would be replaced by a database lookup
  in a real application. */

public String getFlights() {
    String flightOrigin= replaceIfMissing(getOrigin(), "Nowhere");

    String flightDestination= replaceIfMissing(getDestination(), "Nowhere");
    Date today= new Date();
    DateFormat formatter=  DateFormat.getDateInstance(DateFormat.MEDIUM);
    String dateString= formatter.format(today);
    String flightStartDate= replaceIfMissing(getStartDate(), dateString);
    String flightEndDate= replaceIfMissing(getEndDate(), dateString);
    String [][] flights=  {
          { "Java Airways", "1522", "455.95", "Java, Indonesia",
                            "Sun Microsystems", "9:00", "3:15" },
          { "Servlet Express", "2622", "505.95", "New Atlanta",
                               "New Atlanta", "9:30", "4:15" },
          { "Geek Airlines", "3.14159", "675.00", "JHU",
                             "MIT", "10:02:37", "2:22:19" }
    };

    String flightString= "";
    for(int i=0; i<flights.length; i++) {
        String[] flightInfo= flights[i];
        flightString= flightString + getFlightDescription(flightInfo[0],
                                          flightInfo[1],
                                          flightInfo[2],
                                          flightInfo[3],
                                          flightInfo[4],
                                          flightInfo[5],
                                          flightInfo[6],
                                          flightOrigin,
                                          flightDestination,
                                          flightStartDate,
                                          flightEndDate);
```

```
        }

        return(flightString);
    }

/*---------------------------------------*/
    private String getFlightDescription(String airline,
                                        String flightNum,
                                        String price,
                                        String stop1,
                                        String stop2,
                                        String time1,
                                        String time2,
                                        String flightOrigin,
                                        String flightDestination,
                                        String flightStartDate,
                                        String flightEndDate) {
        String flight =
          "<P><BR>\n" +
          "<TABLE WIDTH=\"100%\"><TR><TH CLASS=\"COLORED\">\n" +
          "<B>" + airline + " Flight " + flightNum +
          " ($" + price + ")</B></TABLE><BR>\n" +
          "<B>Outgoing:</B> Leaves " + flightOrigin +
          " at " + time1 + " AM on " + flightStartDate +
          ", arriving in " + flightDestination +
          " at " + time2 + " PM (1 stop -- " + stop1 + ").\n" +
          "<BR>\n" +
          "<B>Return:</B> Leaves " + flightDestination +
          " at " + time1 + " AM on " + flightEndDate +
          ", arriving in " + flightOrigin +
          " at " + time2 + " PM (1 stop -- " + stop2 + ").\n";

        return(flight);
    }

/*---------------------------------------*/
    private String replaceIfMissing(String value,
                                    String defaultValue) {
        if ((value != null) && (value.length() > 0)) {
            return(value);
        }
        else {
            return(defaultValue);
        }
    }

/*---------------------------------------*/
    public static TravelCustomer findCustomer
                                (String emailAddress,
                                 TravelCustomer[] customers) {
        if (emailAddress == null) {
          return(null);
        }
        for(int i=0; i<customers.length; i++) {
            String custEmail= customers[i].getEmailAddress();
            if (emailAddress.equalsIgnoreCase(custEmail)) {
            return(customers[i]);
            }
        }
        return(null);
    }
}
/*---------------------------------------*/
/*---------------------------------------*/
/*---------------------------------------*/
```

RentCars.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!--
Car rental page not implemented -- see airline booking page.
```

```
Taken from More Servlets and JavaServer Pages
from Prentice Hall and Sun Microsystems Press,
http://www.moreservlets.com/.
(C) 2002 Marty Hall; may be freely used or adapted.
-->
<HTML>
<HEAD>
  <TITLE>Not Implemented</TITLE>
  <LINK REL=STYLESHEET
        HREF="/travel/travel-styles.css"
        TYPE="text/css">
</HEAD>
<BODY>
<H1>Car Rental Page Not Implemented</H1>
<CENTER>
Hey, this is only an example. See the airline booking page.
</CENTER>
</BODY>
</HTML>
```

FindHotels.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!--
Hotel page not implemented -- see airline booking page.

Taken from More Servlets and JavaServer Pages
from Prentice Hall and Sun Microsystems Press,
http://www.moreservlets.com/.
(C) 2002 Marty Hall; may be freely used or adapted.
-->
<HTML>
<HEAD>
  <TITLE>Not Implemented</TITLE>
  <LINK REL=STYLESHEET
        HREF="/travel/travel-styles.css"
        TYPE="text/css">
</HEAD>
<BODY>
<H1>Hotel Page Not Implemented</H1>
<CENTER>
Hey, this is only an example. See the airline booking page.
</CENTER>
</BODY>
</HTML>
```

TravelData.java

```
package com.cst407.jsps.travel;

/** This class simply sets up some static data to
 *  describe some supposed preexisting customers.
 *  Use a database call in a real application.
 *  <P>
 *  Taken from More Servlets and JavaServer Pages
 *  from Prentice Hall and Sun Microsystems Press,
 *  http://www.moreservlets.com/.
 *  &copy; 2002 Marty Hall; may be freely used or adapted.
 */

public class TravelData {

    private static FrequentFlyerInfo[] janeFrequentFlyerData= {
            new FrequentFlyerInfo("Java Airways", "123-4567-J"),
            new FrequentFlyerInfo("Delta", "234-6578-D")
        };
    private static RentalCarInfo[] janeRentalCarData= {
        new RentalCarInfo("Alamo", "345-AA"),
        new RentalCarInfo("Hertz", "456-QQ-H"),
```

```
        new RentalCarInfo("Avis", "V84-N8699")
    };
    private static HotelInfo[] janeHotelData= {
        new HotelInfo("Marriot", "MAR-666B"),
         new HotelInfo("Holiday Inn", "HI-228-555")
    };
    private static FrequentFlyerInfo[] joeFrequentFlyerData= {
        new FrequentFlyerInfo("Java Airways", "321-9299-J"),
        new FrequentFlyerInfo("United", "442-2212-U"),
        new FrequentFlyerInfo("Southwest", "1A345")
    };
    private static RentalCarInfo[] joeRentalCarData= {
        new RentalCarInfo("National", "NAT00067822")
    };
    private static HotelInfo[] joeHotelData= {
        new HotelInfo("Red Roof Inn", "RRI-PREF-236B"),
        new HotelInfo("Ritz Carlton", "AA0012")
    };
    private static TravelCustomer[] travelData= {
        new TravelCustomer("jane@somehost.com",
                           "tarzan52",
                           "Jane",
                           "Programmer",
                           "Visa",
                           "1111-2222-3333-6755",
                           "(123) 555-1212",
                           "6 Cherry Tree Lane\n" +
                             "Sometown, CA 22118",
                           janeFrequentFlyerData,
                           janeRentalCarData,
                           janeHotelData),
        new TravelCustomer("joe@somehost.com",
                           "qWeRtY",
                           "Joe",
                           "Hacker",
                           "JavaSmartCard",
                           "000-1111-2222-3120",
                           "(999) 555-1212",
                           "55 25th St., Apt 2J\n" +
                             "New York, NY 12345",
                           joeFrequentFlyerData,
                           joeRentalCarData,
                           joeHotelData)
    };

    public static TravelCustomer[] getTravelData() {
        return(travelData);
    }
}
```

RentalCarInfo.java

```
package com.cst407.jsps.travel;

/** Simple class describing a car company and associated
 *  frequent renter number, used from the TravelData class
 *  (where an array of RentalCarInfo is associated with
 *  each customer).
 *  <P>
 *  Taken from More Servlets and JavaServer Pages
 *  from Prentice Hall and Sun Microsystems Press,
 *  http://www.moreservlets.com/.
 *  &copy; 2002 Marty Hall; may be freely used or adapted.
 */

public class RentalCarInfo {
    private String rentalCarCompany, rentalCarNumber;

    public RentalCarInfo(String rentalCarCompany,
                         String rentalCarNumber) {
        this.rentalCarCompany = rentalCarCompany;
```

```
        this.rentalCarNumber = rentalCarNumber;
    }

    public String getRentalCarCompany() {
        return(rentalCarCompany);
    }

    public String getRentalCarNumber() {
        return(rentalCarNumber);
    }
}
```

HotelInfo.java

```
package com.cst407.jsps.travel;

/** Simple class describing a hotel name and associated
 *  frequent guest number, used from the TravelData class
 *  (where an array of HotelInfo is associated with
 *  each customer).
 *  <P>
 *  Taken from More Servlets and JavaServer Pages
 *  from Prentice Hall and Sun Microsystems Press,
 *  http://www.moreservlets.com/.
 *  &copy; 2002 Marty Hall; may be freely used or adapted.
 */

public class HotelInfo {
    private String hotelName, frequentGuestNumber;

    public HotelInfo(String hotelName,
                     String frequentGuestNumber) {
        this.hotelName= hotelName;
        this.frequentGuestNumber= frequentGuestNumber;
    }

    public String getHotelName() {
        return(hotelName);
    }

    public String getfrequentGuestNumber() {
        return(frequentGuestNumber);
    }
}
```

accounts.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!--
Account application page not implemented.

Taken from More Servlets and JavaServer Pages
from Prentice Hall and Sun Microsystems Press,
http://www.moreservlets.com/.
(C) 2002 Marty Hall; may be freely used or adapted.
-->
<HTML>
<HEAD>
  <TITLE>Not Implemented</TITLE>
  <LINK REL=STYLESHEET
        HREF="/jsp/travel/travel-styles.css"
        TYPE="text/css">
</HEAD>
<BODY>
<H1>Account Application Page Not Implemented</H1>
<CENTER>
Hey, this is only an example. See the airline booking page.
</CENTER>
</BODY>
</HTML>
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!--
Account page not implemented -- see airline booking page.
```

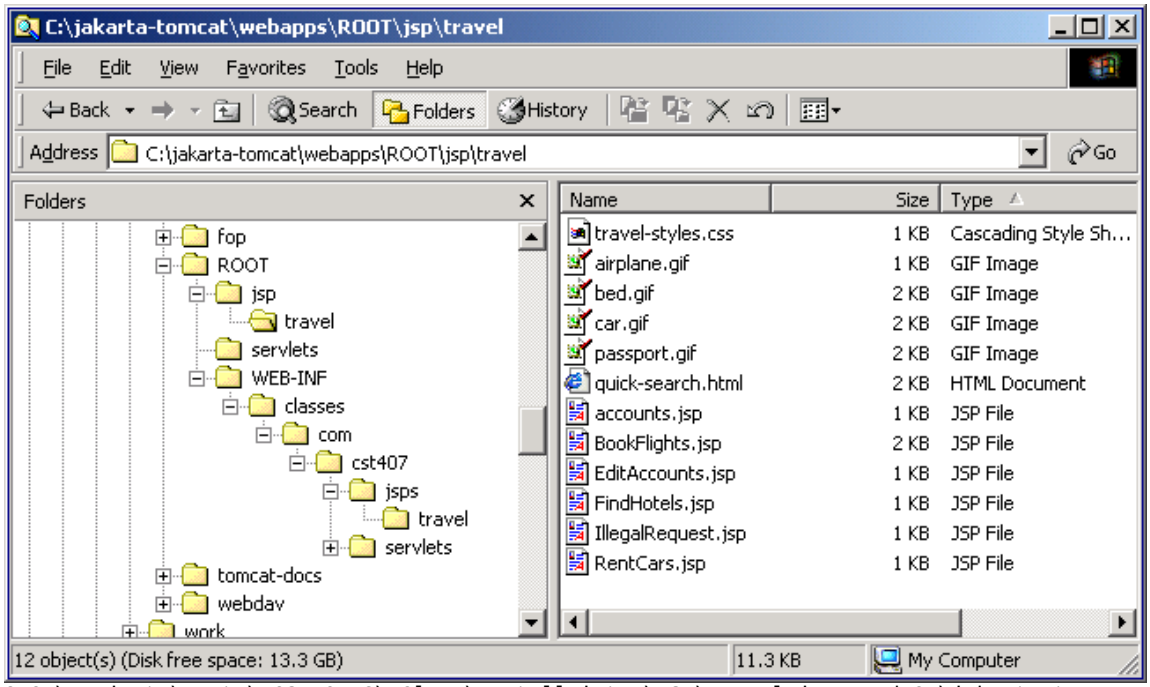
EditAccounts.jsp

```
Taken from More Servlets and JavaServer Pages
from Prentice Hall and Sun Microsystems Press,
http://www.moreservlets.com/.
(C) 2002 Marty Hall; may be freely used or adapted.
-->
<HTML>
<HEAD>
  <TITLE>Not Implemented</TITLE>
  <LINK REL=STYLESHEET
        HREF="/travel/travel-styles.css"
        TYPE="text/css">
</HEAD>
<BODY>
<H1>Accounts Page Not Implemented</H1>
<CENTER>
Hey, this is only an example. See the airline booking page.
</CENTER>
</BODY>
</HTML>
```

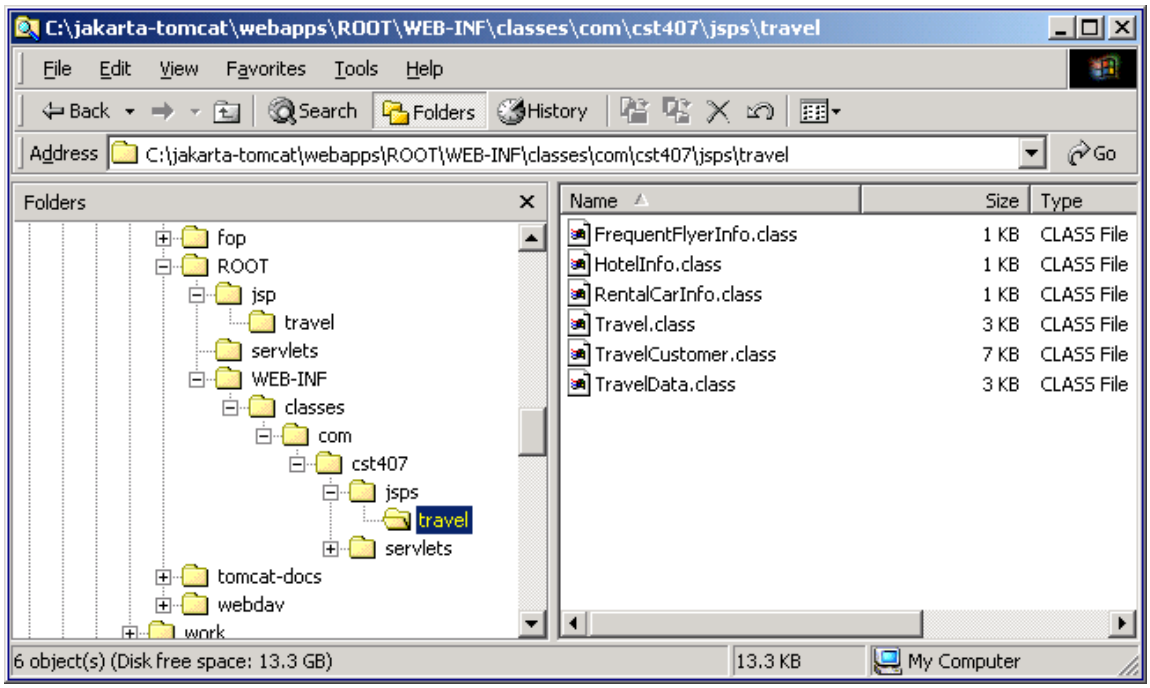
IllegalRequest.jsp

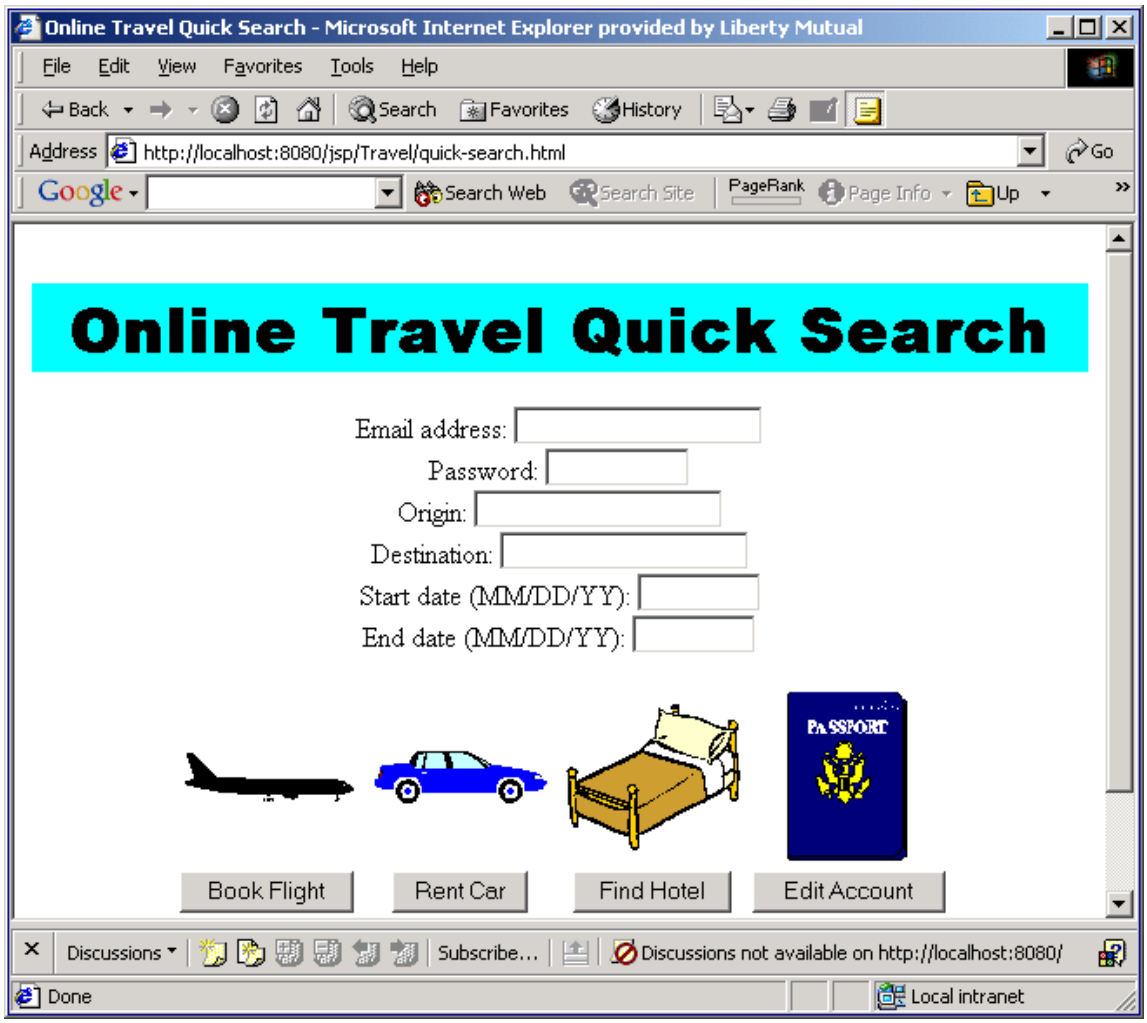
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!--
Illegal request at Travel Quick Search page.

Taken from More Servlets and JavaServer Pages
from Prentice Hall and Sun Microsystems Press,
http://www.moreservlets.com/.
(C) 2002 Marty Hall; may be freely used or adapted.
-->
<HTML>
<HEAD>
  <TITLE>Illegal Request</TITLE>
  <LINK REL=STYLESHEET
        HREF="/travel/travel-styles.css"
        TYPE="text/css">
</HEAD>
<BODY>
<H1>Illegal Request</H1>
<CENTER>Please try again.</CENTER>
</BODY>
</HTML>
```

1.Copy all jsp html, css and gif files to ROOT/jsp/**travel**

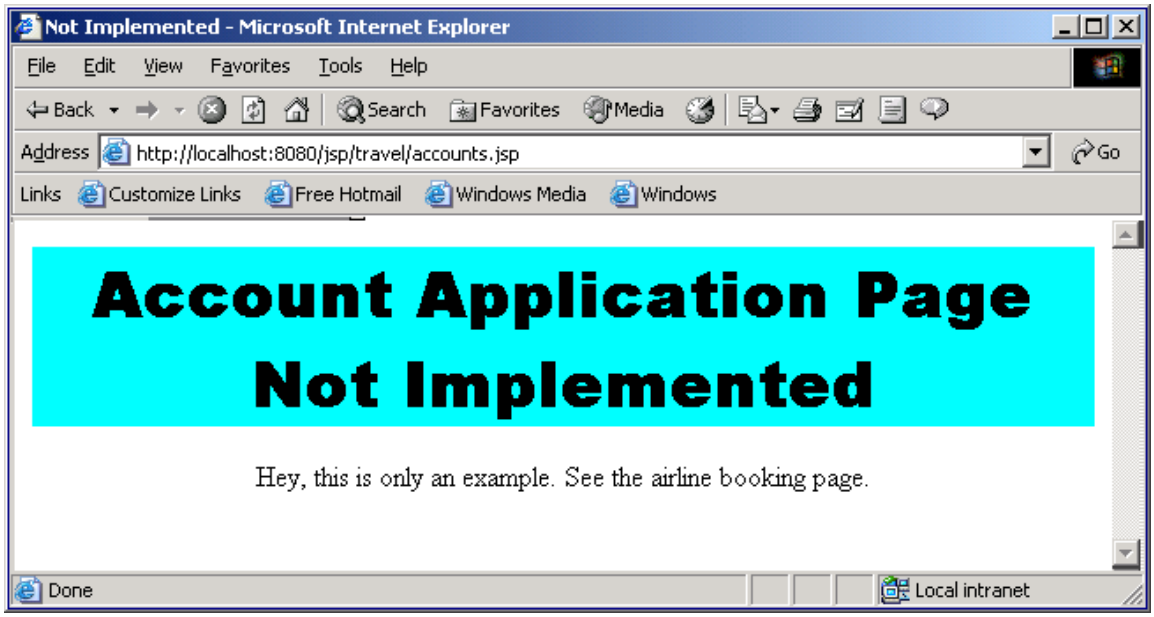


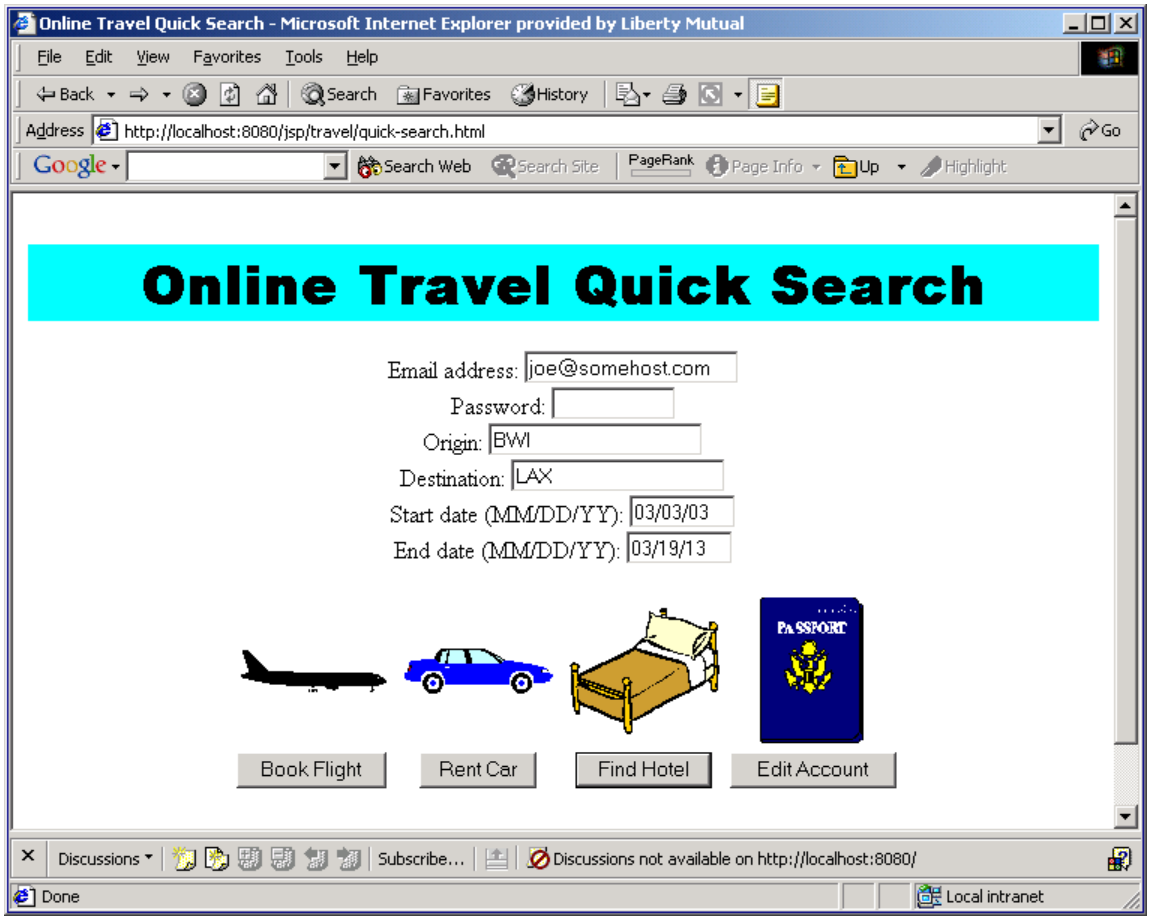

```
2.C:\Projects\Forte\_CST407_2\_Class\TestAllWinter\JSP\Travel>javac -d C:\jakarta-t
omcat\webapps\ROOT\WEB-INF\classes *.java
```



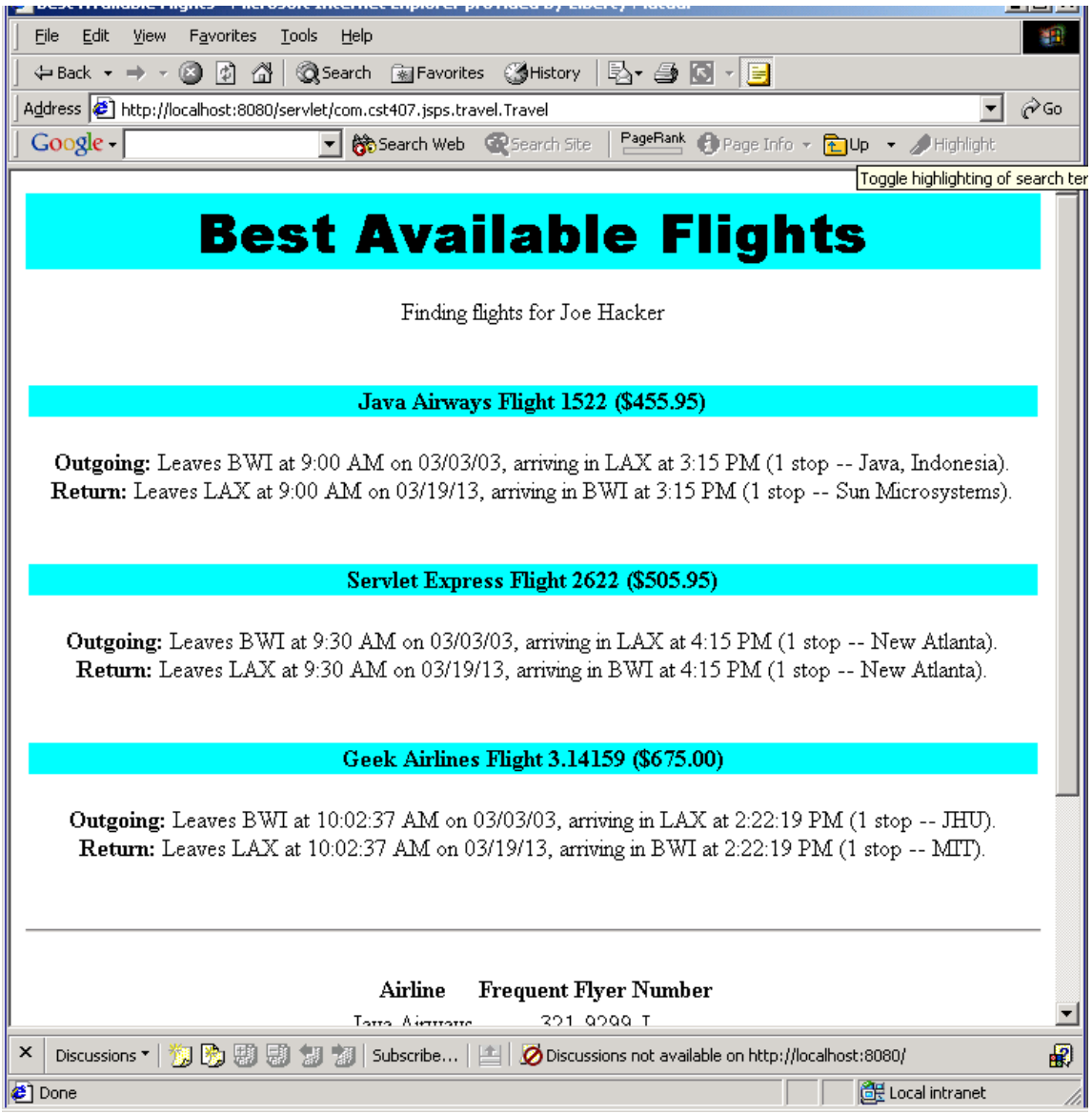


Get a free account:





Password: qWeRtY


File Edit View Favorites Tools Help
Back Search Favorites History
Address http://localhost:8080/servlet/com.cst407.jsps.travel.Travel
Go
Google Search Web Search Site PageRank Page Info Up Highlight
Toggle highlighting of search ter
Best Available Flights
Finding flights for Joe Hacker
Java Airways Flight 1522 ($455.95)
Outgoing: Leaves BWI at 9:00 AM on 03/03/03, arriving in LAX at 3:15 PM (1 stop -- Java, Indonesia).
Return: Leaves LAX at 9:00 AM on 03/19/13, arriving in BWI at 3:15 PM (1 stop -- Sun Microsystems).
Servlet Express Flight 2622 ($505.95)
Outgoing: Leaves BWI at 9:30 AM on 03/03/03, arriving in LAX at 4:15 PM (1 stop -- New Atlanta).
Return: Leaves LAX at 9:30 AM on 03/19/13, arriving in BWI at 4:15 PM (1 stop -- New Atlanta).
Geek Airlines Flight 3.14159 ($675.00)
Outgoing: Leaves BWI at 10:02:37 AM on 03/03/03, arriving in LAX at 2:22:19 PM (1 stop -- JHU).
Return: Leaves LAX at 10:02:37 AM on 03/19/13, arriving in BWI at 2:22:19 PM (1 stop -- MIT).
Airline Frequent Flyer Number
Discussions Subscribe... Discussions not available on http://localhost:8080/
Done
Local intranet