THỰC HÀNH CHUYÊN ĐỀ CƠ SỞ 2
(JAVA NÂNG CAO)

TS. Võ Phương Bình – Email: binhvp@dlu.edu.vn
Khoa CNTT – Trường Đại học Đà Lạt
Website: http://sites.google.com/view/vophuongbinh

Lab 4: MVC & JSP Models

Program Code	**Chuyên đề cơ sở 2 (Java nâng cao)**
Issue/Revision	**8/2018**
Effective date	**3/2011**

Đà Lạt, 8/2018

Contents

1. Lab Guide 01: Struts <HTML> Tags

1. Write a program to display the information entered by a user in Marko career page. In addition, use Html tags to display the information. Create two JavaBeans using the `ActionForm` and `Action` classes to represent data submitted by the user.

Solution:

// ResumeActionForm.java

package MARKO;

```
import org.apache.struts.action.*;

import org.apache.struts.upload.FormFile;

import org.apache.struts.upload.MultipartRequestHandler;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.ByteArrayOutputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;


public class ResumeActionForm extends ActionForm

{

   private String text = "";

   private String textarea = "";

   private String[] selectItems = new String[3];

   private String[] multiBox = new String[5];

   private boolean checkbox = false;

   private String radio = "";

   private FormFile file;
```

```
   private String fileText;


   public String getText()
   {
      return text;
   }
   public void setText(String text)
   {
      this.text = text;
   }
   public String getTextarea()
   {
      return textarea;
   }
   public void setTextarea(String textarea)
   {
      this.textarea = textarea;
   }
   public boolean getCheckbox()
   {
      return checkbox;
   }
   public void setCheckbox(boolean checkbox)
   {
      this.checkbox = checkbox;
   }
   public String getRadio()
   {
      return radio;
```

```
   }
   public void setRadio(String radio)
   {
      this.radio = radio;
   }
   public String[] getSelectItems()
   {
      return selectItems;
   }
   public void setSelectItems(String[] selectItems)
   {
      this.selectItems = selectItems;
   }
   public String[] getMultiBox()
   {
      return multiBox;
   }
   public void setMultiBox(String[] multiBox)
   {
      this.multiBox = multiBox;
   }
   private String[] multipleSelect = {"Multi 3", "Multi 5", "Multi
7"};
   public String[] getMultipleSelect()
   {
      return (this.multipleSelect);
   }
   public void setMultipleSelect(String multipleSelect[])
   {
```

```
      this.multipleSelect = multipleSelect;
   }
   public FormFile getFile()
   {
      return file;
   }
   public void setFile(FormFile file)
   {
      this.file = file;
   }
   public String getFileText()
   {
      try
      {
         ByteArrayOutputStream byteStream = new
         ByteArrayOutputStream();
         InputStream input = file.getInputStream();
         byte[] dataBuffer = new byte[4096];
         int numberBytes = 0;
         while ((numberBytes = input.read(dataBuffer, 0, 4096)) !=
         -1)
         {
            byteStream.write(dataBuffer, 0, numberBytes);
         }
         fileText = new String(byteStream.toByteArray());
         input.close();
      }
      catch (IOException e)
      {
```

```
         return null;

      }

      return fileText;

   }

   public void reset(ActionMapping mapping, HttpServletRequest
request)

   {

   }

}
```

The ResumeActionForm class extends the ActionForm class and stores the data submitted by the user. Struts use ActionForm objects to hold the user data. The **reset()** method is used to reset all properties to their default values.

Save the file as **ResumeActionForm.java** and compile using command prompt. Create a **MARKO** directory in **%TOMCAT_HOME%/webapps/bank/WEB-INF/classes.** Copy the **ResumeActionForm.class** in **MARKO** directory.

// ResumeAction.java
package MARKO;

```
import java.io.*;

import java.util.*;

import MARKO.ResumeActionForm;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.ServletException;

import org.apache.struts.action.*;



public class ResumeAction extends Action

{

   public ActionForward execute(ActionMapping mapping,

   ActionForm form, HttpServletRequest request,

   HttpServletResponse response) throws IOException,
```

```
   ServletException

   {

      ActionErrors actionerrors = new ActionErrors();

      ResumeActionForm dataForm = (ResumeActionForm)form;

      String text = dataForm.getText();

      if(text.trim().equals(""))

      {

         actionerrors.add(ActionErrors.GLOBAL_ERROR, new

         ActionError("error.notext"));

      }

      if(actionerrors.size() != 0)

      {

         saveErrors(request, actionerrors);

         return new ActionForward(mapping.getInput());

      }

      return mapping.findForward("success");

   }

}
```

In the code, the action class interacts with Model component. The action class returns the next View component with the action mapping **findForward()** method.

Save the file as **ResumeActionForm.java** and compile using command prompt. Copy the **ResumeActionForm.class** in **%TOMCAT_HOME%/webapps/bank/WEB-INF/classes /MARKO** directory.

```
//result.jsp
<%@ taglib uri="/tags/struts-bean" prefix="bean" %>
<%@ taglib uri="/tags/struts-logic" prefix="logic" %>

<HTML>
    <HEAD>
    <TITLE>Data Confirmation Screen.....</TITLE>
    </HEAD>

    <BODY>
        <H1>Please confirm the below data</H1>
```

```
        <a href="confirm.do">Confirmed</a>
        <a href="error.do">Error in data</a><br>
        <h2>The Name you provided is:</h2>
        <bean:write name="resumeForm" property="text"/>
        <BR>
        <h2>Your Work experience:</h2>
        <bean:write name="resumeForm" property="textarea"/>
        <BR>
        <h2>Would you like to get relocated</h2>
        <bean:write name="resumeForm" property="checkbox"/>
        <BR>
        <h2>Your selected area of specialization:</h2>
        <bean:write name="resumeForm" property="radio"/>
        <BR>
        <h2>Your location preference:</h2>
        <logic:iterate id="select1" name="resumeForm"
        property="multipleSelect">
        <%= select1 %>
        <BR>
        </logic:iterate>
        <BR>
        <h2>You have heard about Marko Bank from:</h2>
        <logic:iterate id="multibox1" name="resumeForm"
        property="multiBox">
        <%= multibox1 %>
        <BR>
        </logic:iterate>
        <BR>
        <h2>Your Resume in Word format:</h2>
        <bean:write name="resumeForm" property="fileText"/>
    </BODY>
</HTML>
```

The output of the program is as shown in the Figure 20.1.

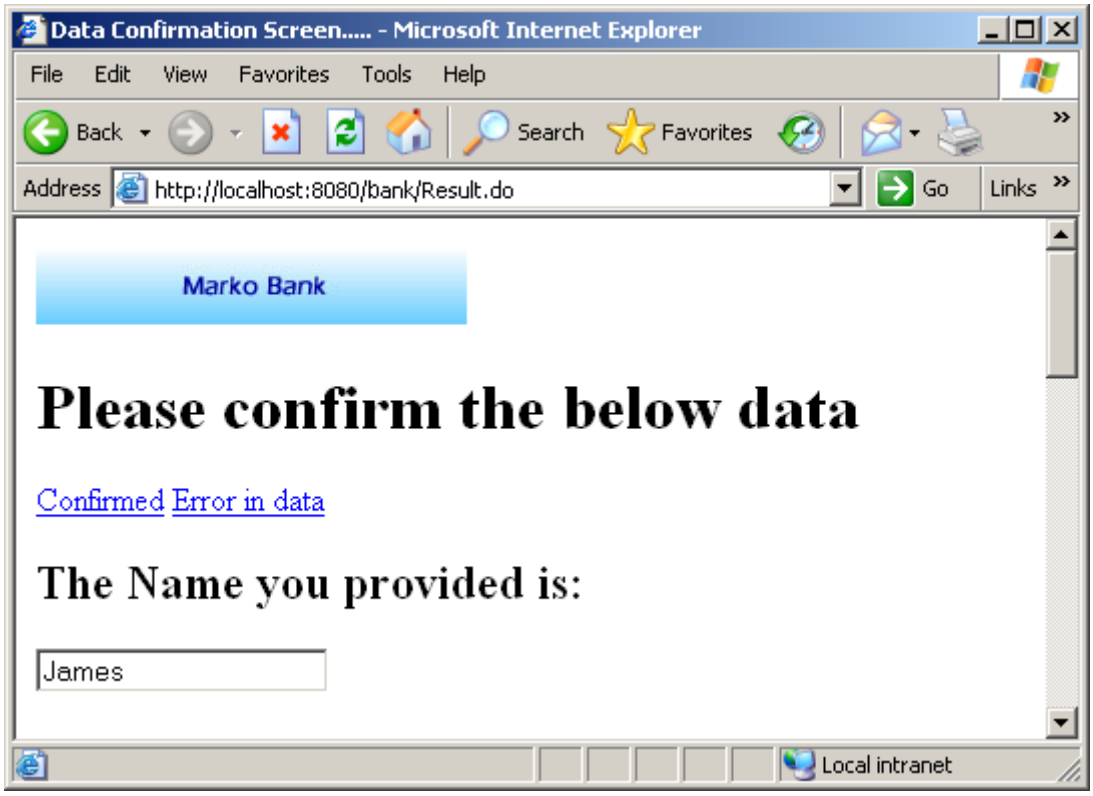

Figure 20.1: Display page

Do It Yourself

1. Consider Example 1 where the user details are displayed. Extend the example by providing hyperlinks for confirmation and error in data.

Solution:

// confirm.jsp
<%@ taglib uri="/tags/struts-html" prefix="html" %>

```
<html:html>


<body>




<h2> Congratulations ! Your Resume has been recorded. You will hear
from us shortly </h2>

</body>

</html:html>
```

Enter the code in Notepad and save the file as **confirm.jsp** in **%TOMCAT_HOME%/webapps/ bank .**

// error.jsp
<%@ taglib uri="/tags/struts-html" prefix="html" %>

```
<html:html>


<BODY>

<H1>Error In Recording information</H1> Please <html:link
action="resume">click here</html:link> to fill the information

</BODY>

</html:html>
```

Enter the code in Notepad and save the file as **error.jsp** in **%TOMCAT_HOME%/webapps/ bank .**

After the user clicks the Confirmed link in Figure 20.1, the output is as shown in Figure 20.2.

Figure 20.2: Confirmation Page

After the user clicks the Error in data, the output is as shown in Figure 20.3.

Figure 20.3: Error Page

When the user clicks the **click here** link, Career Details Form is displayed as shown in Figure 20.2 of Part I.

2. Lab Guide 02: Introduction to <logic> and <bean tags>

1. Extend exercise1 to display the balance in the account entered by a user in Marko Account display page. In addition, use bean and logic tags to display the information.

Solution:

Filenames required for this solution are:

1.display.jsp

2.DisplayActionForm.java

3.Result.jsp

```
//display.jsp


<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<%@ taglib uri="/tags/struts-bean" prefix="bean" %>
<%@ taglib uri="/tags/struts-logic" prefix="logic" %>


<html>
   <head>
      <title>Marko Display Screen</TITLE>
   </head>
   <body>
      <center><h1>Marko Account Display </h1></center>
      <bean:header id="ho" name="user-agent"/>
      <B>You are using :  </B> <%= ho%>
```

```
      <%
         String accNo = "";
         accNo = request.getParameter("accountNo");
         if (accNo!=null)
         {
      %>

      <b><br><br><br><br>
      <center>
         <bean:write  name="displayForm"
         property="accountDetails"/>
      </center>
      <br><br><br></b>
      <%
         }
      %>
      <center>
         <FORM method="post">
            Please enter the account number for which u wish to
            see the balance.<br>
            <input type="text" name="accountNo"><br>
            <input type="submit" value="Show Details">
         </FORM>
      </center>
   </body>
</html>
```

Enter the code in Notepad and save the file as `display.jsp` in `%TOMCAT_HOME%/webapps/ Session8`. A class file needs to be created for retrieving the balance for a particular account number from the database.

Follow the steps to create a class file to retrieve data from database.

```
//DisplayActionForm.java


package MARKO;


import org.apache.struts.action.*;


public class DisplayActionForm extends ActionForm
{
   private String accountDetails = "";
   private String accNo = "";
   public void setAccountNo(String acNo)
   {
      accNo = acNo;
   }
   public String getAccountDetails()
   {
      try
      {
         //accountDetails = "<table><tr><TH>Account
         Number</TH><TH>Balance Amount</TH></tr>";
         Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
         java.sql.Connection connection =
         java.sql.DriverManager.getConnection("jdbc:odbc:userdb");
         java.sql.Statement statement =
```

```
        connection.createStatement();
        String query_car = "select * from userDB where accountNo
        = " + accNo;
        java.sql.ResultSet res =
        statement.executeQuery(query_car);
        if (res.next())
        {
           //accountDetails += "<tr><td>" +
           String.valueOf(res.getInt(1)) + "</td><td>" +
           String.valueOf(res.getInt(2)) + "</td></tr>";
           accountDetails += "Account No : " +
           String.valueOf(res.getInt(1)) + " Balance : " +
           String.valueOf(res.getInt(2));
        }
     //accountDetails += "</table>";
     connection.close();
     }
     catch(Exception e)
     {
        accountDetails = "Error in fetching the records..";
        e.printStackTrace();}
   return accountDetails;
   }
}
```

Save the file as `DisplayActionForm.java` in the `%TOMCAT_HOME%/webapps/Session21` and compile using command prompt. Save the `DisplayActionForm` class file in `MARKO` directory. To display a result to the user, a result page needs to be created.

```
//Result.jsp


<%@ taglib uri="/tags/struts-bean" prefix="bean" %>
<%@ taglib uri="/tags/struts-logic" prefix="logic" %>


<html>
   <head>
      <title>MARKO Account No validator</title>
   </head>


   <body>


      <h1>Marko Account Validator</h1>
      <hr noshade>
      <h2>Using &lt;logic:empty name="LogicActionForm"
      property="userName"&gt;</h2>


      <logic:empty name="LogicActionForm" property="userName">
      User Name cannot be empty…
      </logic:empty>


      <br>
      <hr noshade>
      <h2>Using &lt;logic:notEmpty name="LogicActionForm"
      property="userName"&gt;</h2>


      <logic:notEmpty name="LogicActionForm" property="userName">
      <h2>Welcome :<bean:write name="LogicActionForm"
      property="userName"/></h2>
```

```
<br>
</logic:notEmpty>

<br>
<hr noshade>
<h2>Using &lt;logic:equal name="LogicActionForm"
property="accountNo" value="6"&gt;</h2>

<logic:equal name="LogicActionForm" property="accountNo"
value="5000">
<h4>Your account No is 5000</h4>
</logic:equal>
<br>
<hr noshade>
<h2>Using &lt;logic:notEqual&gt; name="LogicActionForm"
property="accountNo" value="7"</h2>

<logic:notEqual name="LogicActionForm" property="accountNo"
value="5000">
<h4>Your account No is not equal to 5000</h4>
</logic:notEqual>

<br>
<hr noshade>
<h2>Using &lt;logic:greaterEqual name="LogicActionForm"
property="accountNo" value="3"&gt;</h2>

<logic:greaterEqual name="LogicActionForm"
property="accountNo" value="5000">
```

```
<h4>Your account No is Valid</h4>
</logic:greaterEqual>

<br>
<hr noshade>
<h2>Using &lt;logic:greaterThan name="LogicActionForm"
property="accountNo" value="4"&gt;</h2>

<logic:greaterThan name="LogicActionForm"
property="accountNo" value="5000">
<h4>Your account No is Not Valid</h4>
</logic:greaterThan>

<br>
<hr noshade>
<h2>Using &lt;logic:present name="LogicActionForm"
property="accountNo"&gt;</h2>

<logic:present name="LogicActionForm" property="accountNo">
<h4>The Bean has Property account No</h4>
</logic:present>

<br>
<hr noshade>
<h2>Using &lt;logic:notPresent name="LogicActionForm"
property="elephant"&gt;</h2>

<logic:notPresent name="LogicActionForm"
property="elephant">
```

```
      <h4>The Bean doesnt have Property elephant</h4>

      </logic:notPresent>



      <br>

   </body>

</html>
```

Save the file as `result.jsp` in `%TOMCAT_HOME%/webapps/Session21` directory.

Enter the path http://localhost:8080/Session21/index.jsp in the address bar. The welcome page appears, that provides a link to display the account details to the user. When the user clicks on the Display Details link, the output appears as shown in Figure 21.1.

Figure 21.1: Marko Account Display

Enter the account number and click the **Show Details** button. The output appears as shown in Figure 21.2.

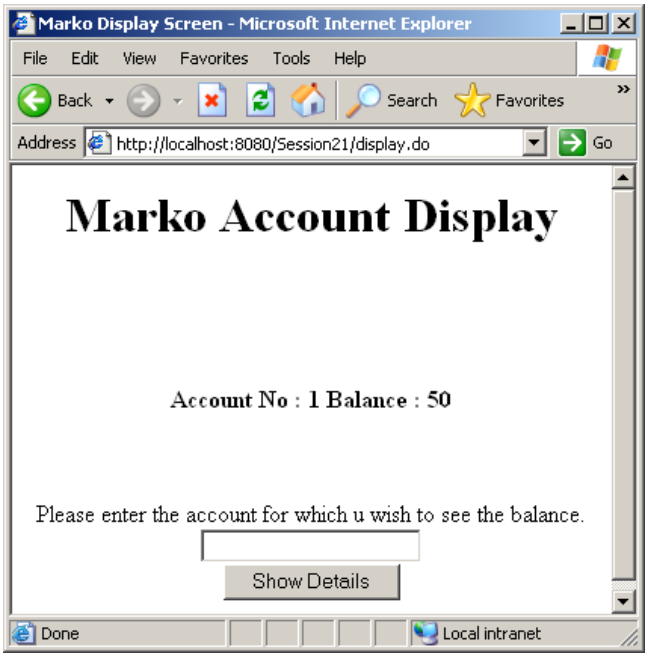

Figure 21.2: Marko Account Details Screen

Do it Yourself

2. Write a program using logic and bean tags to perform the following:

   a. Check whether the cookie is present
   b. Check whether the parameters are present
   c. Check if the test bean is present
   d. Check if the comparison tags are present
   e. Check for the errors and display messages

Solution:

The files used to run this application are:

1. index.jsp
2. PrepareLogicAction.java
3. TestBean.java

```
//index.jsp


<%@ page import="org.apache.struts.action.ActionErrors" %>
```

```
<%@ page import="org.apache.struts.action.ActionMessages" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic" %>

<html>
   <head>
   <title>Logic Tags Example</title>
   </head>
   <body>
      <h1>Logic Tags Example</h1>
      <hr noshade="noshade"/>

      <h2>Present / Not Present</h2>

      <h3>Cookie</h3>
      <logic:present cookie="JSESSIONID">
         <p>Session cookie is present.</p>
      </logic:present>

      <logic:notPresent cookie="UNKNOWN">
         <p>UNKNOWN cookie is not present.</p>
      </logic:notPresent>

      <h3>Parameter</h3>
      <logic:present parameter="param">
         <bean:parameter name="param" id="test"/>
         <p><bean:write name="test"/></p>
      </logic:present>
```

```
<logic:notPresent parameter="param">
   <p>Parameter 'param' not present.
   <html:link action="/prepareLogic?param=The parameter is
   present">
   Redisplay page with parameter present.
   </html:link>
   </p>
</logic:notPresent>

<h3>Bean</h3>
<logic:present name="testBean">
   <p>'testBean' is present.</p>
</logic:present>

<logic:notPresent name="anotherTestBean">
   <p>'anotherTestBean' is not present.</p>
</logic:notPresent>

<logic:present name="testBean" property="fred">
   <p>'fred' property is present on 'testBean'</p>
</logic:present>

<logic:notPresent name="testBean" property="fred">
   <p>'fred' property is not present on 'testBean'</p>
</logic:notPresent>

<logic:present name="testBean" property="stringValue">
   <p>'stringValue' property is present on 'testBean'</p>
```

```
    </logic:present>

    <h2>Empty / Not Empty</h2>
    <logic:present name="items">
       <p>'items' was found.</p>
    </logic:present>

    <logic:empty name="items">
       <p>'items' is empty</p>
    </logic:empty>

    <logic:notEmpty name="items">
       <p>'items' is not empty</p>
       <%--  <bean:size collection="items" id="itemsSize"/>
       <p>Items has <bean:write name="itemsSize" /> items.</p>
       --%>
    </logic:notEmpty>

    <h2>Comparison tags</h2>
    <logic:equal name="intValue" value="7">
       <p>intValue == 7</p>
    </logic:equal>

    <logic:greaterEqual name="intValue" value="7">
       <p>intValue &gt;= 7</p>
    </logic:greaterEqual>

    <logic:greaterEqual name="intValue" value="6">
       <p>intValue &gt;= 6</p>
```

```
    </logic:greaterEqual>

    <logic:greaterThan name="intValue" value="6">
       <p>intValue &gt; 6</p>
    </logic:greaterThan>

    <logic:lessEqual name="intValue" value="7">
       <p>intValue &lt;= 7</p>
    </logic:lessEqual>

    <logic:lessEqual name="intValue" value="8">
       <p>intValue &lt;= 8</p>
    </logic:lessEqual>

    <logic:lessThan name="intValue" value="8">
       <p>intValue &lt; 8</p>
    </logic:lessThan>

    <h2>Checking for and displaying messages</h2>
    <h3>Errors:</h3>

    <logic:messagesPresent>
    <p>Global errors:</p>
    <ul>
       <html:messages id="error"
          property="<%=ActionErrors.GLOBAL_ERROR%>">
          <li><bean:write name="error"/></li>
       </html:messages>
    </ul>
```

```
      <p>Errors for 'test':</p>
      <ul>

      <html:messages id="error" property="test">
         <li><bean:write name="error"/></li>
      </html:messages>
      </ul>
      </logic:messagesPresent>

      <logic:messagesNotPresent>
      <p>There are no errors</p>
      </logic:messagesNotPresent>

      <h3>Messages:</h3>
      <logic:messagesPresent message="true">

      <ul>
      <html:messages id="msg" message="true"
property="<%=ActionMessages.GLOBAL_MESSAGE%>">
      <li><bean:write name="msg"/></li>
      </html:messages>
      </ul>

      <p>Messages for 'test':</p>
      <ul>
         <html:messages id="msg" property="test" message="true">
            <li><bean:write name="msg"/></li>
         </html:messages>
      </ul>
```

```
      </logic:messagesPresent>


      <logic:messagesNotPresent message="true">
         <p>There are no messages</p>
      </logic:messagesNotPresent>


   </body>
</html>
```

Enter the code in Notepad and save the file as `index.jsp` in `%TOMCAT_HOME%/webapps/example`.

```
//PrepareLogicAction.java


package MARKO;


import java.util.*;
import javax.servlet.http.*;
import org.apache.struts.action.*;


public class PrepareLogicAction extends Action
{
   public PrepareLogicAction()
   {
      super();
   }


      public ActionForward execute(ActionMapping mapping,
      ActionForm form, HttpServletRequest request,
```

```
      HttpServletResponse response) throws Exception

      {


         TestBean bean = new TestBean();

         request.setAttribute("testBean", bean);


         ActionErrors errors = new ActionErrors();

         errors.add(ActionErrors.GLOBAL_ERROR,

         new ActionError("errors.detail", "This is a global error
#1"));


         errors.add(ActionErrors.GLOBAL_ERROR,

         new ActionError("errors.detail", "This is a global error
#2"));


         errors.add("test", new ActionError("errors.detail", "This is
a test error"));


         ActionMessages messages = new ActionMessages();

         messages.add(ActionMessages.GLOBAL_MESSAGE,

         new ActionMessage("message.detail", "This is global

         message #1"));

         messages.add(ActionMessages.GLOBAL_MESSAGE,

         new ActionMessage("message.detail", "This is global

         message #2"));

         messages.add("test", new

         ActionMessage("message.example.simple"));

         saveMessages(request, messages);

         saveErrors(request, errors);
```

```
        return mapping.findForward("success");
   }
}
```

Enter the code in Notepad and save the file as `prepareLogicAction.java` in `%TOMCAT_HOME%/webapps/example`. After compilation of Java files, a **MARKO** folder is automatically created with class files in it. Place the **MARKO** folder in "example/WEB-INF/Classes" folder.

```
//TestBean.java


package MARKO;


import java.io.*;
public class TestBean implements Serializable
{
   private boolean booleanValue = false;
   private double doubleValue = 45213.451;
   private float floatValue = -123.582F;
   private int intValue = 256;


   private long longValue = 1321546L;
   private short shortValue = 257;
   private String stringValue = "Hello, world!";
   private java.util.Date dateValue = new java.util.Date();


   public TestBean()
   {
      super();
```

```
    }
    public boolean isBooleanValue()
    {
       return booleanValue;
    }
    public double getDoubleValue()
    {
       return doubleValue;
    }
    public float getFloatValue()
    {
       return floatValue;
    }
    public int getIntValue()
    {
       return intValue;
    }
    public long getLongValue()
    {
       return longValue;
    }
    public short getShortValue()
    {
       return shortValue;
    }
    public String getStringValue()
    {
       return stringValue;
    }
```

```
public void setBooleanValue(boolean booleanValue)
{
   this.booleanValue = booleanValue;
}
public void setDoubleValue(double doubleValue)
{
   this.doubleValue = doubleValue;
}
public void setFloatValue(float floatValue)
{
   this.floatValue = floatValue;
}
public void setIntValue(int intValue)
{
   this.intValue = intValue;
}
public void setLongValue(long longValue)
{
   this.longValue = longValue;
}
public void setShortValue(short shortValue)
{
   this.shortValue = shortValue;
}
public void setStringValue(String stringValue)
{
   this.stringValue = stringValue;
}
public java.util.Date getDateValue()
```

```
    {
       return dateValue;
    }
    public void setDateValue(java.util.Date date)
    {
       this.dateValue = date;
    }
}
```

Enter the code in Notepad and save the file as `TestBean.java` in `%TOMCAT_HOME%/webapps/example`. After compilation, copy the class file in /example/WEB-INF/Classes/MARKO folder.

The output of the program is as shown in the Figure 21.3.

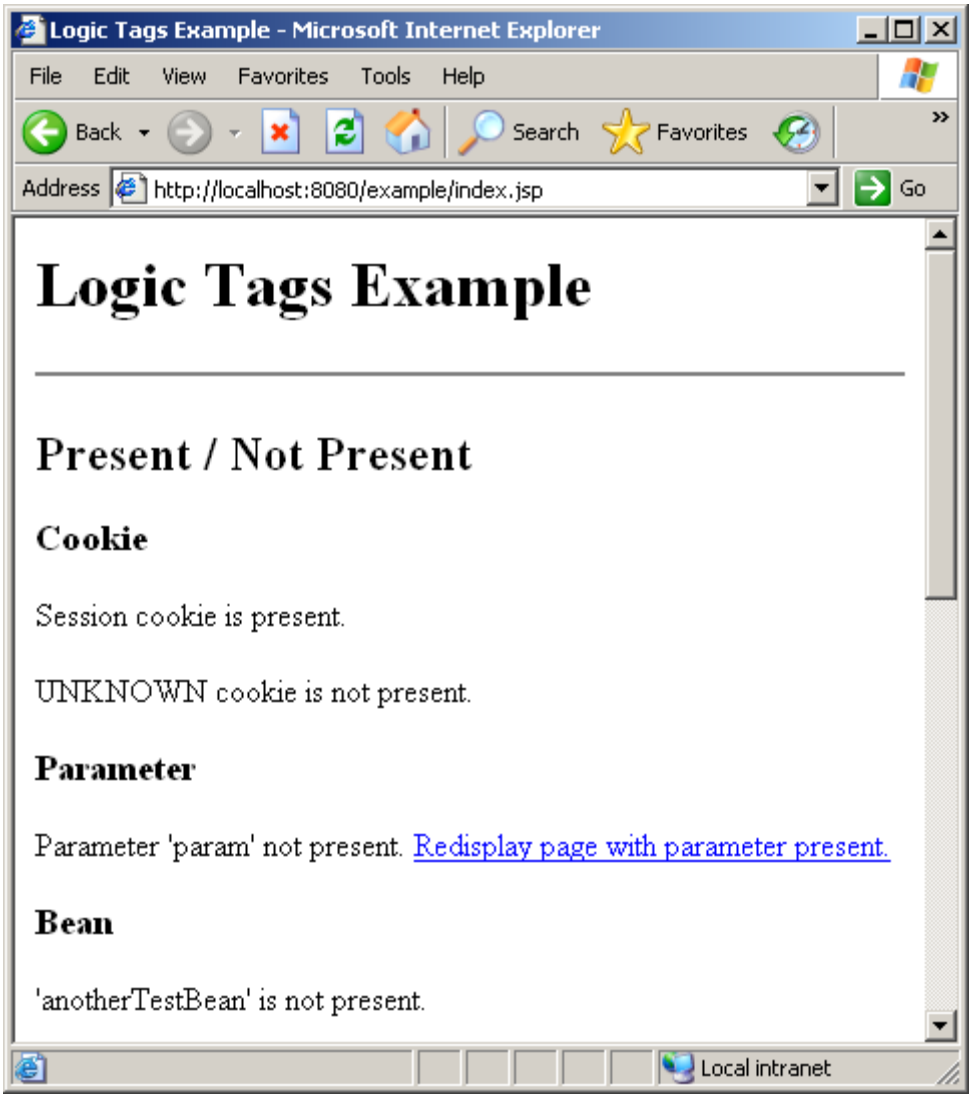

Figure 21.3: Logic Tag Example Page

Click the link `Redisplay page with parameter present` on the Logic Tag Example page. The output of the confirmation page is as shown in the Figure 21.4.

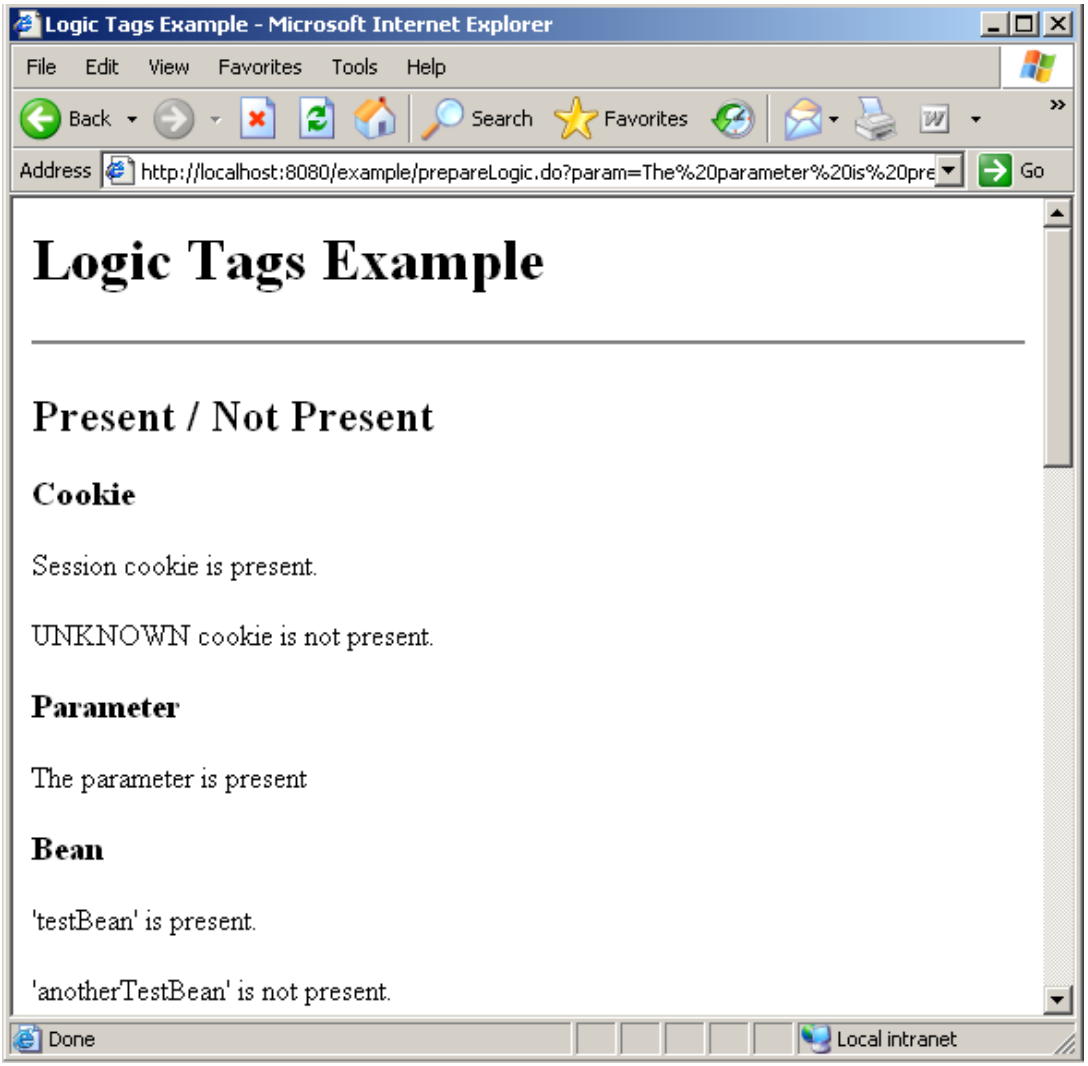

Figure 21.4: Present/Not Present Parameters

3. Lab Guide 03: Creating Custom Tags

1. Consider Example 2 where a customer details link is provided. Extend the example by providing a link to branch details. On clicking the link, a new Web page should display the branch details.

Solution:

The list of files used for this exercise is:

1. index.jsp
2. BranchTag.java
3. Marko.tld
4. display.jsp

// index.jsp

```
<html>
   <head>
      <title>Home Page</title>
   </head>
   <body>
      <center><h1>Marko Bank Home Page</h1></center>
      Please select the options below<br>
      <a href="display.jsp?data=customer">Customer
      Details</A><BR>
      <a href="display.jsp?data=branch">Branch Details</a><br>
   </body>
</html>
```

Update the `index.jsp` page with hyperlink for Branch details. Save the file in `%TOMCAT_HOME%/webapps/tags`.

// BranchTag.java

```
package MARKO;


import javax.servlet.jsp.tagext.*;
import javax.servlet.jsp.*;


public class BranchTag extends TagSupport
{
   public int doEndTag() throws JspException
   {
      try
      {
         String data = "<table border='2' align='center'>
         <TR>
            <TH>Branch Name</TH>
            <TH>Address</TH>
            <TH>Telephone No</TH>
         </TR>";
         Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
         java.sql.Connection connection =
         java.sql.DriverManager.getConnection("jdbc:odbc:userdb");
         java.sql.Statement statement =
         connection.createStatement();
         String query_car = "select * from BranchDetails";
         java.sql.ResultSet res =
```

```
        statement.executeQuery(query_car);
        while(res.next())
        {
           data += "<tr><TD>" + res.getString(1) + "</td>";
           data += "<TD>" + res.getString(2) + "</td></tr>";
           data += "<TD>" + res.getString(3) + "</td></tr>";
        }
        data += "</table>";
        pageContext.getOut().print(data);
        }
        catch (Exception e)
        {
           throw new JspException(e.toString());
        }
      return EVAL_PAGE;
   }
}
```

Enter the Java code in Notepad and save the file as `BranchTag.java`. Compile the file from the command prompt and copy the class file in `%TOMCAT_HOME%/webapps/tags/WEB-INF/classes/MARKO`.

//Marko.tld

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE taglib PUBLIC
"-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.2//EN"
"http://java.sun.com/dtd/web-jsptaglibrary_1_2.dtd">
<taglib>
```

```
   <tlib-version>1.0</tlib-version>
   <jsp-version>1.2</jsp-version>
   <short-name>TagExamples</short-name>
   <description>Example tags.</description>
   <tag>
      <name>customer</name>
      <tag-class>MARKO.CustomerTag</tag-class>
   </tag>

   <tag>
      <name>branch</name>
      <tag-class>MARKO.BranchTag</tag-class>
   </tag>
</taglib>
```

Update the `Marko.tld` created in Example 2. Save the file in `%TOMCAT_HOME%/webapps/tags/WEB-INF`.

//display.jsp

```
<%@ taglib prefix="MARKO" uri="WEB-INF/Marko.tld" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
   <head>
      <title> MARKO </title>
   </head>
   <body>
      <center><h1>Marko Bank Display Page</h1></center>
      <%
```

```
         String action = request.getParameter("data");
         if (action == null || action.equals(""))
         {
      %>
      <jsp:forward page="/index.jsp"/>
      <%
         }
         if (action.equals("customer"))
         {
      %>
      <MARKO:customer />
      <%
      }
      %>
      <%
      if (action.equals("branch")){
      %>
      <MARKO:branch />
      <%
      }
      %>
      <a href="javascript:hostory.back();"> <<back</A>
   </body>
</html>
```

Update the `display.jsp` page with the `If` condition to display the branch details. Save the file in `%TOMCAT_HOME%/webapps/tags`.

The output of the program is as shown in Figure 22.1.

Figure 21.1: Branch Details

Do It Yourself

1. Consider Example 2 where a customer details link is provided. Extend the example by providing a link to contact details. On clicking the link, a new Web page should display the bank's contact details.

Solution:

//index.jsp

```
<html>
   <head>
      <title>Home Page</title>
   </head>
   <body>
      <center><h1>Marko Bank Home Page</h1></center>
      Please select the options below<br>
```

```
      <a href="display.jsp?data=customer">Customer
      Details</A><BR>
      <a href="display.jsp?data=branch">Branch Details</a><br>
      <a href="display.jsp?data=contact">Contact Details</a><br>
   </body>
</html>
```

Update the `index.jsp` page with hyperlink for Branch details. Save the file in `%TOMCAT_HOME%/webapps/tags`.

//BranchTag.java

```
package MARKO;


import javax.servlet.jsp.tagext.*;
import javax.servlet.jsp.*;


public class BranchTag extends TagSupport
{
   public int doEndTag() throws JspException
   {
      try
      {
         String data = "<table border='2' align='center'>
         <TR>
            <TH>Branch Name</TH>
            <TH>Address</TH>
            <TH>Telephone No</TH>
         </TR>";
```

```
         Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
         java.sql.Connection connection =
         java.sql.DriverManager.getConnection("jdbc:odbc:userdb");
         java.sql.Statement statement =
         connection.createStatement();
         String query_car = "select * from BranchDetails";
         java.sql.ResultSet res =
         statement.executeQuery(query_car);
         while(res.next())
         {
            data += "<tr><TD>" + res.getString(1) + "</td>";
            data += "<TD>" + res.getString(2) + "</td></tr>";
            data += "<TD>" + res.getString(3) + "</td></tr>";
         }
         data += "</table>";
         pageContext.getOut().print(data);
      }
      catch (Exception e)
      {
         throw new JspException(e.toString());
      }
      return EVAL_PAGE;
   }
}
```

Enter the Java code in Notepad and save the file as `BranchTag.java`. Compile the file from the command prompt and copy the class file in `%TOMCAT_HOME%/webapps/tags/WEB-INF/classes/MARKO`.

//Marko.tld

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE taglib PUBLIC
"-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.2//EN"
"http://java.sun.com/dtd/web-jsptaglibrary_1_2.dtd">
<taglib>
   <tlib-version>1.0</tlib-version>
   <jsp-version>1.2</jsp-version>
   <short-name>TagExamples</short-name>
   <description>Example tags.</description>
   <tag>
      <name>customer</name>
      <tag-class>MARKO.CustomerTag</tag-class>
   </tag>
   <tag>
      <name>branch</name>
      <tag-class>MARKO.BranchTag</tag-class>
   </tag>
</taglib>
```

Update the `Marko.tld` created in Example 2. Save the file in `%TOMCAT_HOME%/webapps/tags/WEB-INF`.

//display.jsp

```
<%@ taglib prefix="MARKO" uri="WEB-INF/Marko.tld" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
```

```
<html>
   <head>
      <title> MARKO </title>
   </head>
   <body>
      <center><h1>Marko Bank Display Page</h1></center>
      <%
         String action = request.getParameter("data");
         if (action == null || action.equals(""))
         {
      %>
      <jsp:forward page="/index.jsp"/>
      <%
         }
         if (action.equals("customer"))
         {
      %>
      <MARKO:customer />
      <%
      }
      %>
      <%
         if (action.equals("branch")){
      %>
      <MARKO:branch />
      <%
         }
      %>
      <a href="javascript:hostory.back();"> <<back</A>
```

```
   </body>

</html>
```

Update the `display.jsp` page with the `If` condition to display the branch details. Save the file in `%TOMCAT_HOME%/webapps/tags`.

The output of the program is as shown in Figure 22.2.

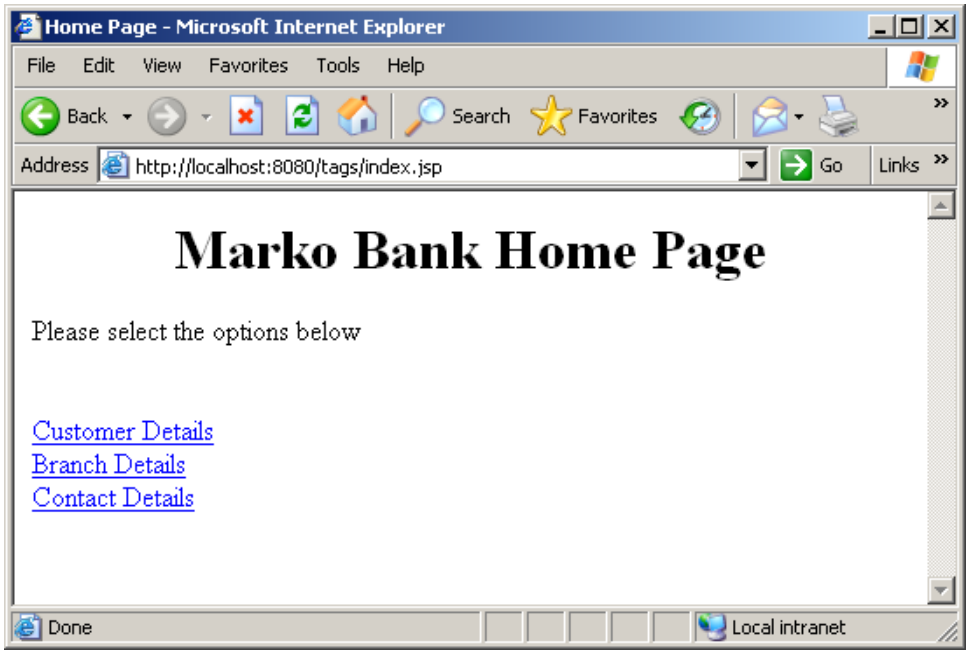

Figure 22.2: Home page

The output of the program to display Branch details is as shown in Figure 22.3.

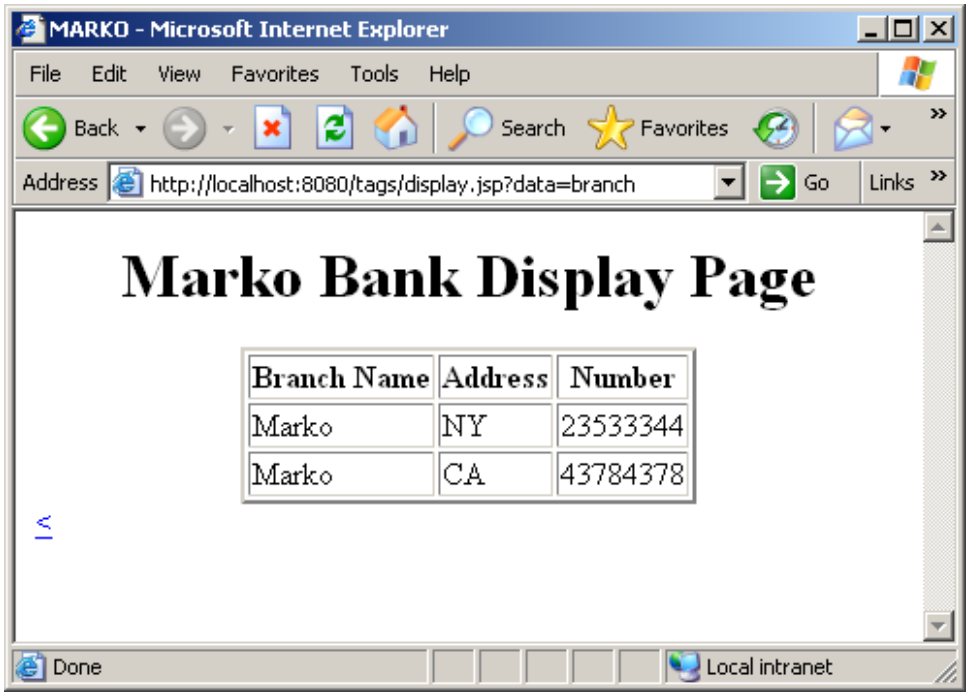

Figure 22.3: Branch Details

4. Lab Guide 04: Struts Validator Framework

1. Create a Web page with two fields: Name and security number. The page should allow the user to search for the security number or the name, on entering either of the fields. Display the security number of the name that is entered in the Name field. Similarly, the page should provide the name after entering the security number.

Solution:

The list of files, required for this application is as follows:

1. index.jsp
2. search.jsp
3. Employee.java
4. EmployeeSearchService.java
5. searchAction.java
6. Searchform.java
7. struts-config.xml
8. validation.xml
9. ApplicationResources.Properties

Update the `index.jsp` file in `%TOMCAT_HOME%/webapps/example/index.jsp`

```
//index.jsp


<%@ taglib uri="/WEB-INF/tlds/struts-html.tld" prefix="html" %>


<html>

   <head>

      <title>Online Banking with Marko</title>

   </head>

   <body>

      <font size="+1">Marko</font><br>

      <hr width="100%" noshade="true">

      &#149; <html:link forward="search">Search for
Account</html:link><br>
```

```
   </body>

</html>
```

The output appears as shown in Figure 23.1.

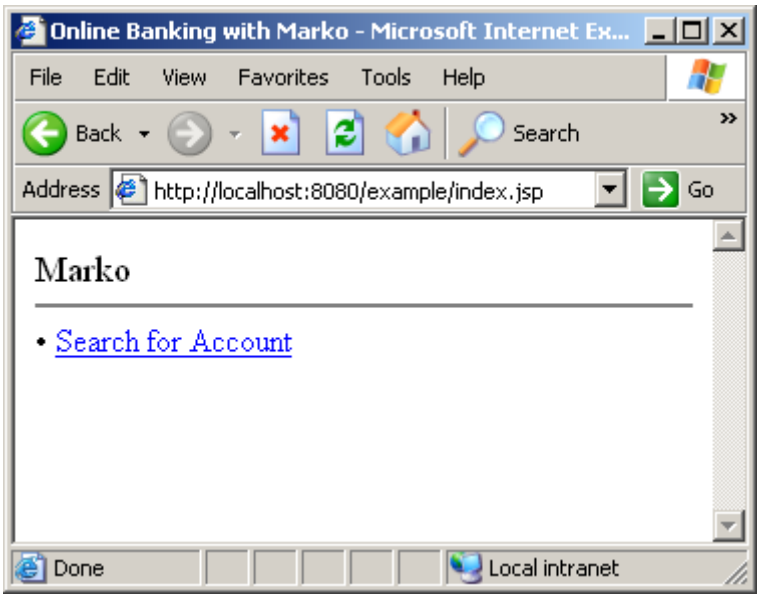

Figure 23.1: Index Page

Enter the code in a Notepad and save the file as `search.jsp` in `%TOMCAT_HOME%/webapps/example/search.jsp`.

```
//search.jsp


<%@ taglib uri="/WEB-INF/tlds/struts-bean.tld" prefix="bean" %>

<%@ taglib uri="/WEB-INF/tlds/struts-html.tld" prefix="html" %>

<%@ taglib uri="/WEB-INF/tlds/struts-logic.tld" prefix="logic" %>


<html>
```

```
   <head>
   <title>Account Search</title>
   </head>
   <body>


      <font size="+1">
      Online Banking with Marko
      </font><br>
      <hr width="100%" noshade="true">
      <html:errors/>
      <html:form action="/search.jsp">
         <table>
            <tr>
               <td align="right"><bean:message
key="label.search.name"/>:</td>
               <td><html:text property="name"/></td>
            </tr>
            <tr>
               <td align="right"><bean:message
key="label.search.ssNum"/>:</td>
               <td><html:text property="ssNum"/> (xxx-xx-xxxx)</td>
               </tr>
               <tr>
               <td></td>
               <td><html:submit/></td>
            </tr>
         </table>
      </html:form>
```

```
<logic:present name="searchForm" property="results">

   <hr width="100%" size="1" noshade="true">

   <bean:size id="size" name="searchForm" property="results"/>
   <logic:equal name="size" value="0">
      <center><font color="red">
         <b>
         No Account Found....Enter valid name and security
         number
         </b>
         </font>
      </center>
   </logic:equal>
   <logic:greaterThan name="size" value="0">
      <table border="1">
         <tr>
            <th>Name</th>
            <th>Social Security Number</th>
         </tr>
         <logic:iterate id="result" name="searchForm"
         property="results">
         <tr>
            <td><bean:write name="result"
            property="name"/></td>
            <td><bean:write name="result"
            property="ssNum"/></td>
         </tr>
         </logic:iterate>
```

```
            </table>

         </logic:greaterThan>


      </logic:present>


   </body>
</html>
```

When the user clicks on the `Search for Account link`, the account search page is displayed as shown in Figure 23.2.

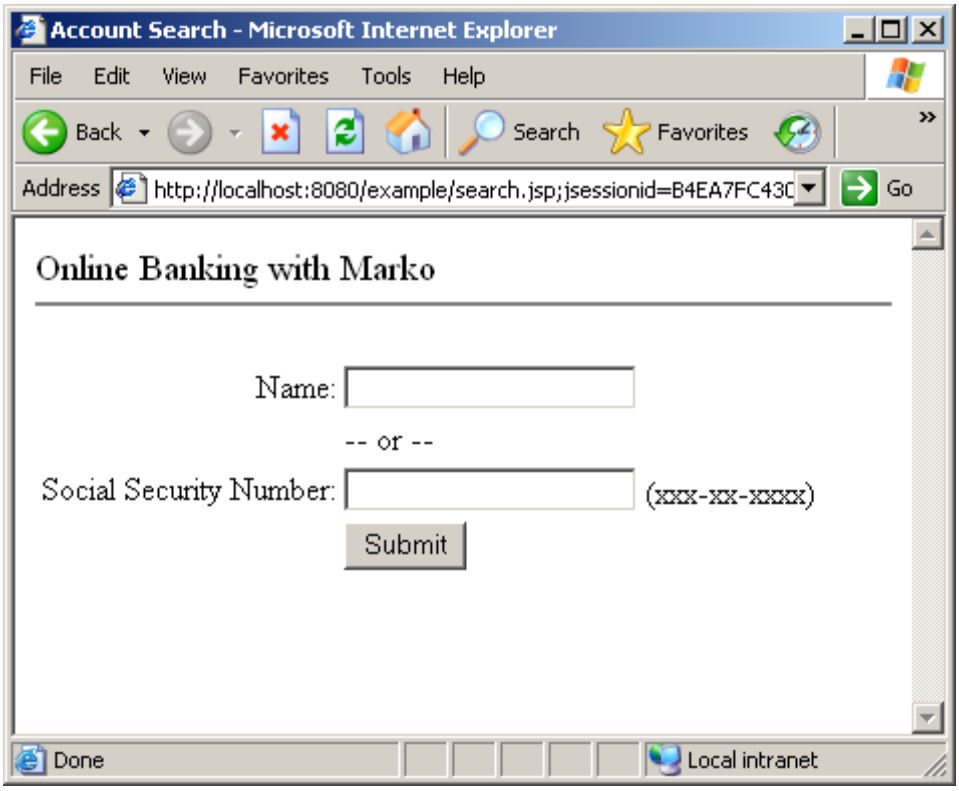

Figure 23.2: Account Search Page

Enter the code in a Notepad and save the file as `Employee.java` in `%TOMCAT_HOME%/webapps/example/Employee.java`.

```
//Employee.java


package com;


public class Employee
{
   private String name;
   private String ssNum;


   public Employee(String name, String ssNum)
   {
      this.name = name;
      this.ssNum = ssNum;
   }


   public void setName(String name)
   {
      this.name = name;
   }


   public String getName()
   {
      return name;
   }


   public void setSsNum(String ssNum)
   {
      this.ssNum = ssNum;
   }
```

```
   public String getSsNum()
   {
      return ssNum;
   }
}
```

Enter the code in a Notepad and save the file as `EmployeeSearchService.java` in `%TOMCAT_HOME%/webapps/example/EmployeeSearchService.java`.

```
//EmployeeSearchService.java

package com;

import java.util.ArrayList;

public class EmployeeSearchService
{
  /* Hard-coded sample data. Normally this would come from a real
data
source such as a database. */
   private static Employee[] employees =
   {
      new Employee("George Smith", "123-45-6789"),
      new Employee("Eldson Stewart", "987-65-4321"),
      new Employee("Roberts Shankle", "111-11-1111"),
      new Employee("Patrick Wilson", "222-22-2222"),
      new Employee("Jim Frank", "333-33-3333"),
   };
```

```
   // Search for employees by name.
   public ArrayList searchByName(String name)
   {
      ArrayList resultList = new ArrayList();

      for (int i = 0; i < employees.length; i++)
         {
         if
(employees[i].getName().toUpperCase().indexOf(name.toUpperCase()) !=
-1)
         {
            resultList.add(employees[i]);
         }
      }
      return resultList;
   }

   // Search for employee by social security number.
   public ArrayList searchBySsNum(String ssNum)
      {
         ArrayList resultList = new ArrayList();

         for (int i = 0; i < employees.length; i++)
         {
            if (employees[i].getSsNum().equals(ssNum))
            {
               resultList.add(employees[i]);
            }
```

```
      }
   return resultList;
   }
}
```

Enter the code in a Notepad and save the file as `searchAction.java` in `%TOMCAT_HOME%/webapps/example/searchAction.java`.

```
//SearchAction.java

package com;

import java.util.ArrayList;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

public final class SearchAction extends Action
{
   public ActionForward execute(ActionMapping mapping,
   ActionForm form,
   HttpServletRequest request,
   HttpServletResponse response)
       throws Exception
```

```
   {
      EmployeeSearchService service = new EmployeeSearchService();
      ArrayList results;
      SearchForm searchForm = (SearchForm) form;
      // Perform employee search based on what criteria was entered.
      String name = searchForm.getName();
      if (name != null && name.trim().length() > 0)
         {
            results = service.searchByName(name);
         }
      else
         {
            results =
            service.searchBySsNum(searchForm.getSsNum().trim());
         }

      // Place search results in SearchForm for access by JSP.
      searchForm.setResults(results);

      // Forward control to this Action's input page.
      return mapping.getInputForward();
   }
}
```

Enter the code in a Notepad and save the file as `Searchform.java` in `%TOMCAT_HOME%/webapps/example/Searchform.java`.

```
//Searchform.java

```

```
package com;


import java.util.List;
import org.apache.struts.validator.ValidatorForm;


public class SearchForm extends ValidatorForm
{
   private String name = null;
   private String ssNum = null;
   private List results = null;


   public void setName(String name)
   {
      this.name = name;
   }
   public String getName()
   {
      return name;
   }


   public void setSsNum(String ssNum)
   {
      this.ssNum = ssNum;
   }


   public String getSsNum()
   {
      return ssNum;
   }
```

```
   public void setResults(List results)
   {
      this.results = results;
   }

   public List getResults()
   {
      return results;
   }
}
```

Update the `struts-config.xml` file in `%TOMCAT_HOME%/webapps/example/WEB-INF/struts-config.xml`.

```
//struts-config.xml

<!DOCTYPE struts-config PUBLIC
  "-//Apache Software Foundation//DTD Struts Configuration 1.1//EN"
  "http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">

<struts-config>

   <!-- Form Beans Configuration -->
   <form-beans>
      <form-bean name="searchForm" type="com.SearchForm"/>
   </form-beans>

   <!-- Global Forwards Configuration -->
```

```
   <global-forwards>
      <forward name="search" path="/search.jsp"/>
   </global-forwards>

   <!-- Action Mappings Configuration -->
   <action-mappings>
      <action path="/search"
      type="com.SearchAction"
      name="searchForm"
      scope="request"
      validate="true"
      input="/search.jsp">
      </action>
   </action-mappings>

   <!-- Message Resources Configuration -->
   <message-resources parameter="com.ApplicationResources"/>

   <!-- Validator Configuration -->
   <plug-in
   className="org.apache.struts.validator.ValidatorPlugIn">
   <set-property property="pathnames"
   value="/WEB-INF/validator-rules.xml,
   /WEB-INF/validation.xml"/>
   </plug-in>

</struts-config>
```

Update the `validation.xml` file in `%TOMCAT_HOME%/webapps/example/WEB-INF/struts-config.xml`.

```
//validation.xml


<!DOCTYPE form-validation PUBLIC

          "-//Apache Software Foundation//DTD Commons

            Validator Rules Configuration 1.0//EN"

"http://jakarta.apache.org/commons/dtds/validator_1_0.dtd">


<form-validation>

<formset>

   <form name="searchForm">

      <field property="ssNum" depends="mask">

      <arg0 key="label.search.ssNum"/>

      <var>

      <var-name>mask</var-name>

      <var-value>^\d{3}-\d{2}-\d{4}$</var-value>

      </var>

      </field>

   </form>

</formset>

</form-validation>
```

Update the `ApplicationResources.Properties` file in `%TOMCAT_HOME%/webapps/example/WEB-INF/classes/com/ApplicationResources.Properties`.

```
//Application.Resources
```

```
# Label Resources

label.search.name=Name

label.search.ssNum=Social Security Number


# Error Resources

error.search.criteria.missing=<li>Search Criteria Missing</li>

error.search.ssNum.invalid=<li>Invalid Social Security Number</li>

errors.header=<font color="red"><b>Validation
Error(s)</b></font><ul>

errors.footer=</ul><hr width="100%" size="1" noshade="true">


errors.invalid=<li>{0} is not valid</li>
```

After entering the path in the browser, the index.jsp page appears. When the user clicks on the `Search for Account link`, the account search page is displayed.

When the user enters an invalid value in either of the fields, and clicks on Submit button, the error message is displayed as shown in Figure 23.3.

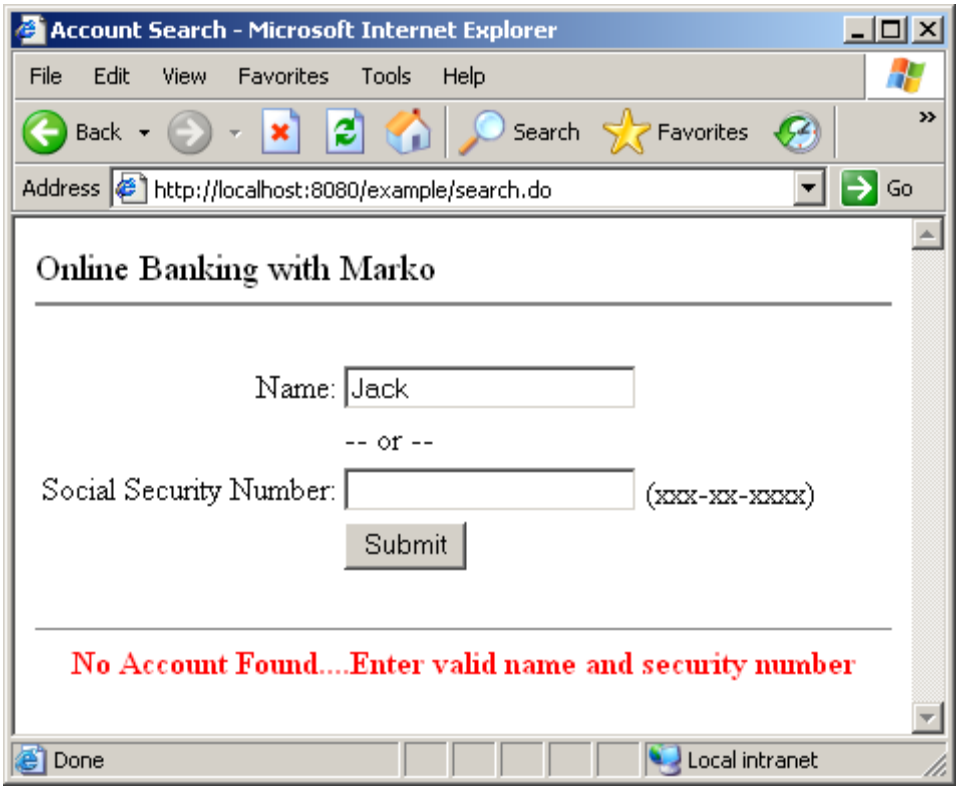

Figure 23.3: Invalid Entry Message

When the user enters a valid name to search for the security number, the output is displayed as shown in Figure 23.4.

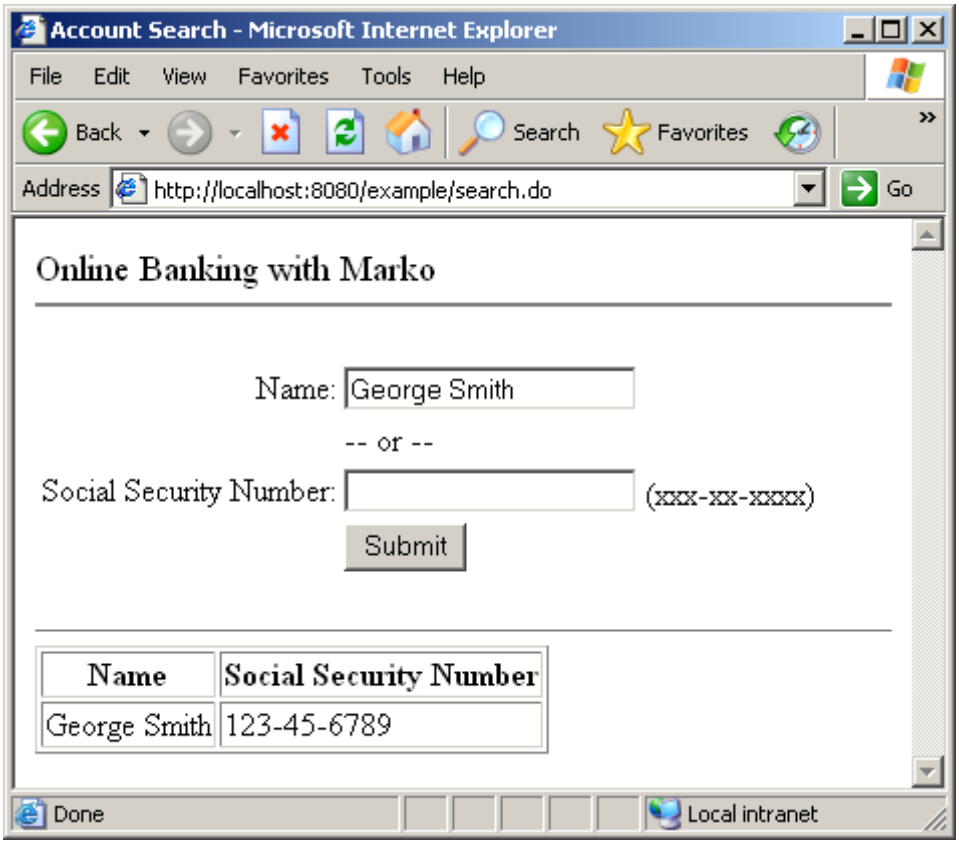

Figure 23.4: Result Page

Do It Yourself

1. Create a Web page to provide various fields to the user to enter details. The page should allow the user to enter the details. The page should contain some mandatory fields. If the user leaves any mandatory field blank, display an error message to the user. If the user enters valid details and provides input to all the mandatory fields, display a confirmation page to the user.

Solution:

The list of files used in this application, is as follows:

1. index.jsp
2. submitcomment.jsp
3. SubmitCommentAction.java
4. thankyou.jsp
5. struts-config.xml
6. validation.xml
7. application.properties

Update the `index.jsp` file in `%TOMCAT_HOME%/webapps/struts1/index.jsp`

```
//index.jsp


<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
prefix="bean" %>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
prefix="html" %>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic"
prefix="logic" %>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-nested"
prefix="nested" %>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-template"
prefix="template" %>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-tiles"
prefix="tiles" %>


<html>

   <head>

       <title><bean:message key="label.index.title" /></title>

   </head>

   <body>


      <h3><bean:message key="label.index.title" /></h3>

      <p><a href="view.submitcomment.do">

      <bean:message key="label.index.commentSubmissionForm"/></a>


   </body>

</html>
```

The output appears as shown in Figure 23.5.

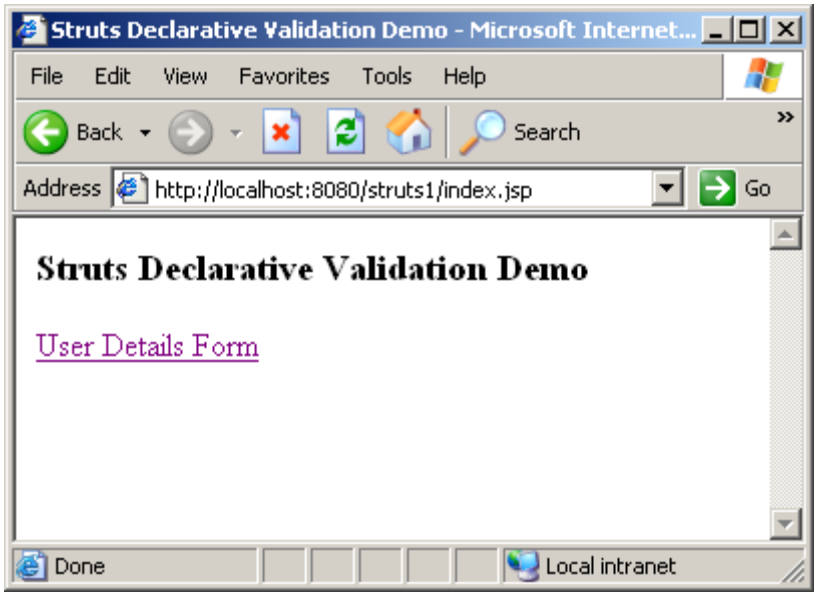

Figure 23.5: User Details Form

Enter the code in a Notepad and save the file as `submitcomment.jsp` in `%TOMCAT_HOME%/webapps/struts1/submitcomment.jsp`.

```
//submitcomment.jsp


<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
prefix="bean" %>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
prefix="html" %>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic"
prefix="logic" %>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-nested"
prefix="nested" %>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-template"
prefix="template" %>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-tiles"
prefix="tiles" %>


<html>
```

```
<header><title>User detail page</title></header>
<body>

<h3>User Details Form</h3>
<ul>
<font color=red>
<html:messages id="message">
<li><bean:write name="message"/></li>
</html:messages>
</font>
</ul>

<p>* Indicates a required field<p>

<table>
   <html:form action="/submitcomment">
      <tr>
         <td>* First name:</td>
         <td><html:text property="userID"></html:text></td>
      </tr>
      <tr>
      <tr>
         <td>* Last name:</td>
         <td><html:text property="last"></html:text></td>
      </tr>
      <tr>
      <tr>
         <td>* Country:</td>
         <td><html:text property="country"></html:text></td>
```

```
      </tr>
      <tr>
         <td>* Email:</td>
         <td><html:text property="email"></html:text></td>
      </tr>
      <tr>
         <td>Zip code:</td>
         <td><html:text property="zipCode"></html:text></td>
      </tr>
         <td>Phone Number:</td>
         <td><html:text property="phone"></html:text></td>
      </tr>
      <tr>
         <td><html:submit value="Submit"></html:submit></td>
      </tr>
    </html:form>
</table>
</body>
</html>
```

The `submitcomment.jsp` page appears as shown in Figure 23.6.

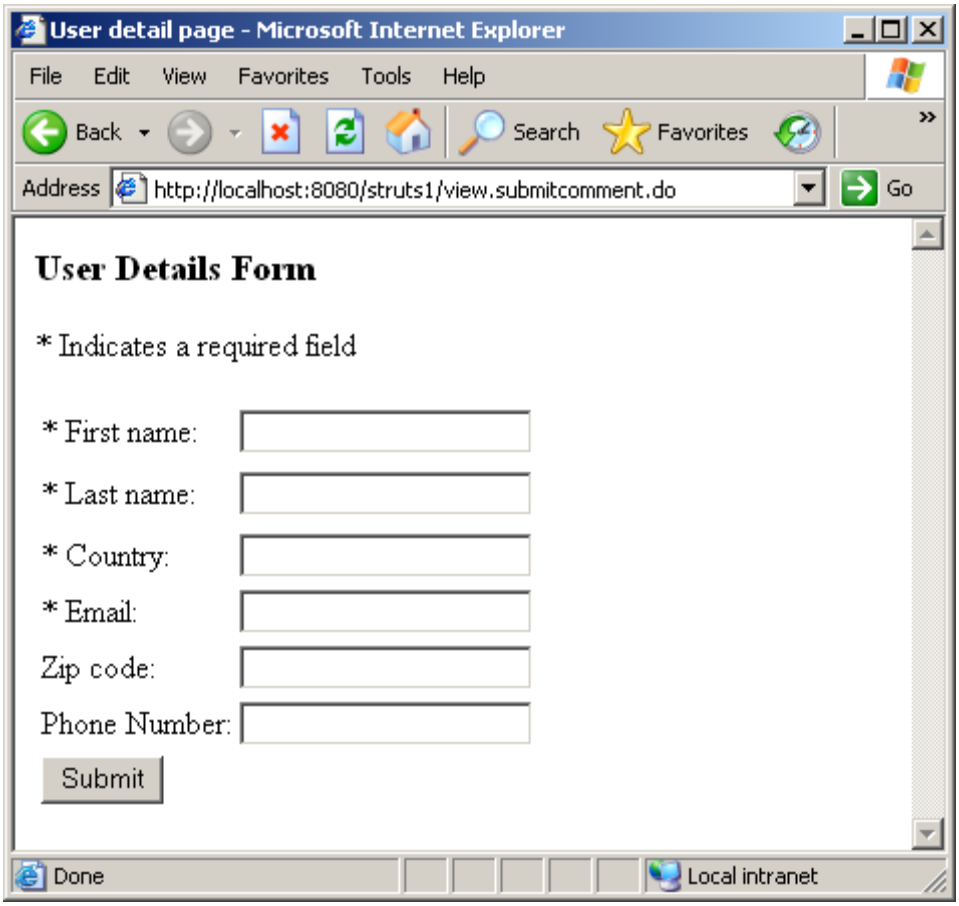

Figure 23.6: User Details Submission Page

Enter the code in a Notepad and save the file as `submitcommentAction.java` in `%TOMCAT_HOME%/webapps/struts1/submitcommentAction.java`.

```
//SubmitCommentAction.java


package app;

import org.apache.struts.action.*;

import javax.servlet.http.*;

import java.io.*;


public class SubmitCommentAction extends Action

{
```

```
   public ActionForward execute(ActionMapping mapping, ActionForm
form, HttpServletRequest req, HttpServletResponse res)

   throws Exception

   {

      // *** Your business logic would go here ***

      // For example, perhaps to save the form values to a database.

      // For simplicity of illustration, we just print a couple
values:

      String userID = req.getParameter("userID");

      String shareEmail = req.getParameter("shareEmail");

      System.out.println("userID = " + userID);

      if (null != shareEmail)

      System.out.println("shareEmail = " + shareEmail);


      return mapping.findForward("success");

      // If your business logic failed, then you would

      // return mapping.findForward("failure");

   }

}
```

Enter the code in a Notepad and save the file as `thankyou.jsp` in `%TOMCAT_HOME%/webapps/struts1/thankyou.jsp`.

```
//thankyou.jsp


<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
prefix="bean" %>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
prefix="html" %>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic"
prefix="logic" %>
```

```
<%@ taglib uri="http://jakarta.apache.org/struts/tags-nested"
prefix="nested" %>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-template"
prefix="template" %>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-tiles"
prefix="tiles" %>



<html>

   <head><title>Thank you submitting your details</title></head>

   <body>

      <h3>Details Accepted</h3><p>



      <bean:parameter id="userID" name="userID"/>



      Thank you for submitting your details, <bean:write
name="userID" />.<p>

      <logic:present parameter="shareEmail">

      We will display your email address with your comments on the
public forum page.

      </logic:present>

   </body>

</html>
```

Update the struts-config.xml file in `%TOMCAT_HOME%/webapps/struts1/WEB-INF/struts-config.xml`.

```
//struts-config.xml



<?xml version="1.0" encoding="ISO-8859-1" ?>



<!DOCTYPE struts-config PUBLIC
```

```
          "-//Apache Software Foundation//DTD Struts Configuration
1.1//EN"

          "http://jakarta.apache.org/struts/dtds/struts-
config_1_1.dtd">

<struts-config>

<form-beans>

   <form-bean name="commentForm
type="org.apache.struts.validator.DynaValidatorForm">

     <form-property name="userID" type="java.lang.String"/>

     <form-property name="last" type="java.lang.String"/>

     <form-property name="country" type="java.lang.String"/>

     <form-property name="email" type="java.lang.String"/>

     <form-property name="shareEmail" type="java.lang.String"/>

     <form-property name="zipCode" type="java.lang.String"/>

     <form-property name="phone" type="java.lang.String"/>

   </form-bean>

</form-beans>

<!-- ================================== Global Exception Definitions
-->



<global-exceptions>

<!-- sample exception handler

<exception key="expired.password"
type="app.ExpiredPasswordException" path="/changePassword.htm"/>

end sample -->

</global-exceptions>



<!-- ================================== Global Forward Definitions
-->



<global-forwards>
```

```
<!-- Default forward to "Welcome" action -->

<!-- Demonstrates using index.jsp to forward -->

<forward name="welcome" path="/Welcome.do"/>

</global-forwards>

<!-- ================================== Action Mapping Definitions
-->


<action-mappings>

   <action path="/Welcome"
type="org.apache.struts.actions.ForwardAction"
parameter="/pages/Welcome.htm"/>

   <action path="/view.submitcomment" parameter="/submitcomment.jsp"

   type="org.apache.struts.actions.ForwardAction"

   name="commentForm" input="/submitcomment.htm" validate="false">

   </action>


   <action path="/submitcomment" type="app.SubmitCommentAction"

   name="commentForm" input="/submitcomment.jsp" validate="true">

   <forward name="success" path="/thankyou.jsp"  />

   <forward name="failure" path="/submitcomment.jsp" />

   </action>

</action-mappings>

<!-- ==================================== Controller Configuration
-->


<controller
processorClass="org.apache.struts.tiles.TilesRequestProcessor"/>


<!-- ================================ Message Resources Definitions
-->

<message-resources parameter="resources.application"/>
```

```
<plug-in className="org.apache.struts.tiles.TilesPlugin" >
   <set-property property="definitions-config"
                           value="/WEB-INF/tiles-defs.xml" />
   <set-property property="moduleAware" value="true" />
   <set-property property="definitions-parser-validate" value="true"
/>
</plug-in>

<plug-in className="org.apache.struts.validator.ValidatorPlugIn">
   <set-property property="pathnames" value="/WEB-INF/validator-
rules.xml,/WEB-INF/validation.xml"/>
</plug-in>

</struts-config>
```

Update the `validation.xml` file in `%TOMCAT_HOME%/webapps/struts1/WEB-INF/struts-config.xml`.

```
//validation.xml

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE form-validation PUBLIC
"-//Apache Software Foundation//DTD Commons Validator Rules
Configuration 1.0//EN"

"http://jakarta.apache.org/commons/dtds/validator_1_0.dtd">
<form-validation>
   <global>
   <constant>
```

```
<constant-name>zipmask</constant-name>
<constant-value>^\d{5}(-\d{4})?$</constant-value>
</constant>
</global>

<formset>
   <form name="commentForm">
   <field property="userID" depends="required,mask">
      <arg0 key="commentForm.userID"/>
      <var>
         <var-name>mask</var-name>
         <var-value>^[0-9a-zA-Z ]*$</var-value>
      </var>
   </field>
   <field property="last" depends="required,mask">
      <arg0 key="commentForm.last"/>
      <var>
         <var-name>mask</var-name>
         <var-value>^[0-9a-zA-Z ]*$</var-value>
      </var>
   </field>

   <field property="country" depends="required,mask">
   <arg0 key="commentForm.country"/>
   <var>
      <var-name>mask</var-name>
      <var-value>^[0-9a-zA-Z ]*$</var-value>
   </var>
   </field>
```

```
    <field property="email" depends="required,mask">
       <arg0 key="commentForm.email"/>
       <var>
          <var-name>mask</var-name>
          <var-value>^[-A-Za-z0-9_.]+@[-A-Za-z0-9_.]+\.[A-Za-
          z]{2,}$</var-value>
       </var>
    </field>
    <field property="zipCode" depends="mask">
       <arg0 key="commentForm.zipCode"/>
       <var>
          <var-name>mask</var-name>
          <var-value>${zipmask}</var-value>
       </var>
    </field>
    <field property="phone" depends="mask">
       <arg0 key="commentForm.phone"/>
       <var>
       <var-name>mask</var-name>
       <var-value>${zipmask}</var-value>
       </var>
    </field>
    </form>
  </formset>
</form-validation>
```

Update the `application.properties` file in `%TOMCAT_HOME%/webapps/struts1/WEB-INF/classes/resources/application.properties`.

```
//Application.properties


# -- standard errors --
errors.header=<UL>
errors.prefix=<LI>
errors.suffix=</LI>
errors.footer=</UL>
# -- validator --
errors.invalid={0} is invalid.
errors.maxlength={0} can not be greater than {1} characters.
errors.minlength={0} can not be less than {1} characters.
errors.range={0} is not in the range {1} through {2}.
errors.required={0} is required.
errors.byte={0} must be an byte.
errors.date={0} is not a date.
errors.double={0} must be an double.
errors.float={0} must be an float.
errors.integer={0} must be an integer.
errors.long={0} must be an long.
errors.short={0} must be an short.
errors.creditcard={0} is not a valid credit card number.
errors.email={0} is an invalid e-mail address.
# -- other --
errors.cancel=Operation cancelled.
errors.detail={0}
```

```
errors.general=The process did not complete. Details should follow.

errors.token=Request could not be completed. Operation is not in
sequence.


#-- application-specific messages --

label.index.title=Struts Declarative Validation Demo

label.index.commentSubmissionForm=User Details Form


commentForm.userID=User name entry

commentForm.email=Email entry

commentForm.zipCode=Zip Code entry

commentForm.phone=Phone Number entry

commentForm.comments=Comments entry
```

After entering the path in the browser, the `index.jsp` page is displayed. When the user clicks on the user clicks on the link, the user details form appears. When the user enters the details and clicks on Submit button, the output appears as shown in Figure 23.7.

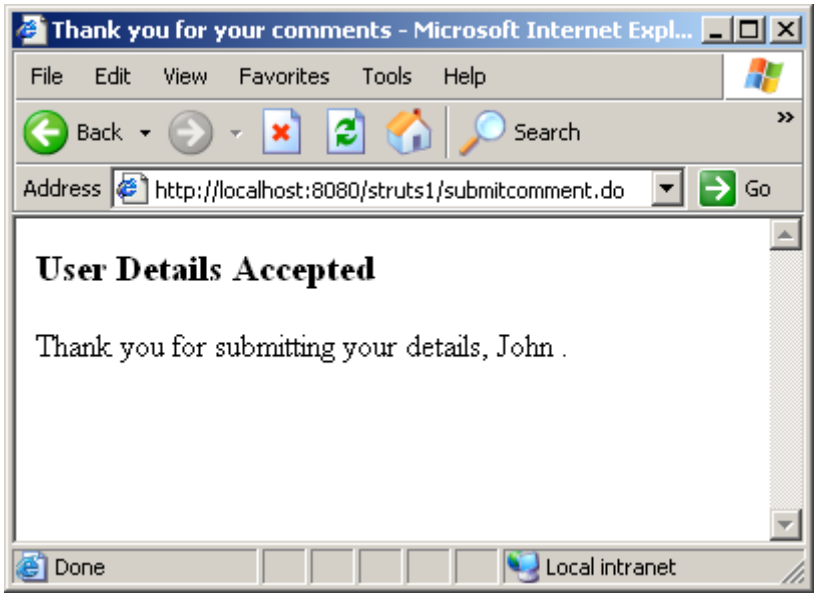

Figure 23.7: Confirmation Page

When the user enters invalid details, an error message is displayed. The output appears as shown in Figure 23.8.

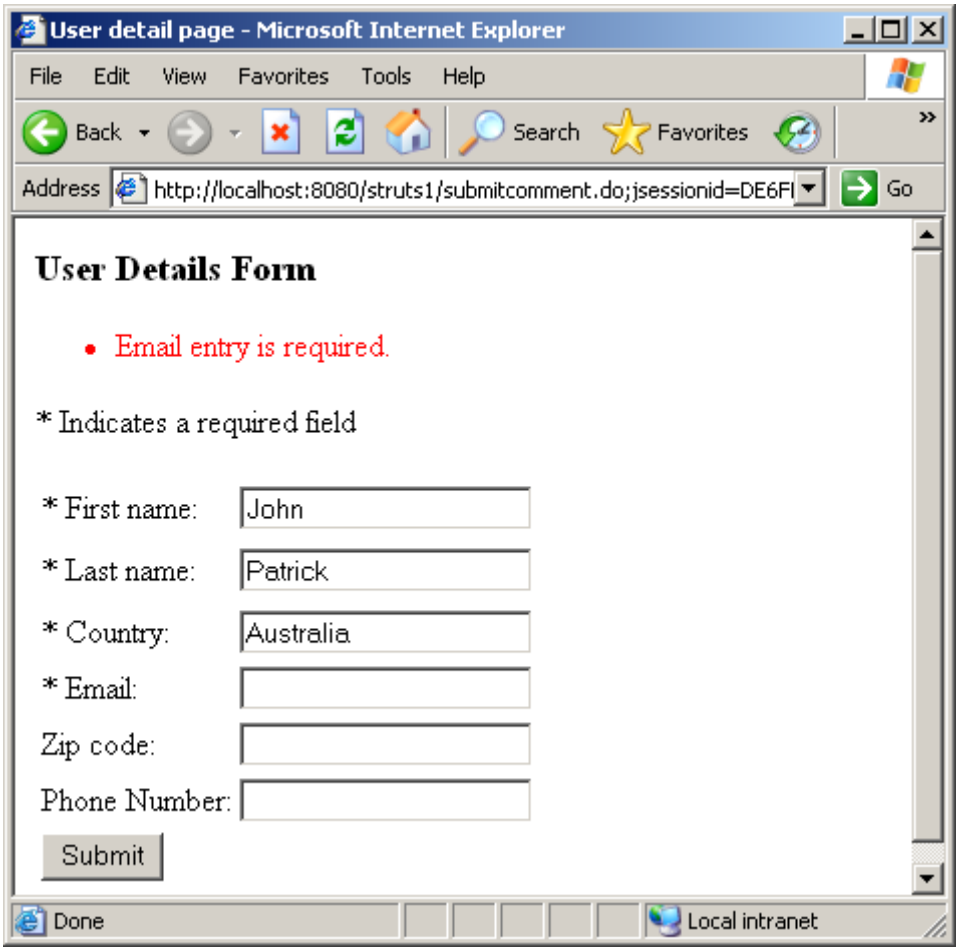

Figure 23.8: Invalid User Details Page

5. Lab Guide 05: Tiles Framework

1. Create a Web page with five regions; header, title, body, footer, and menu. The header should contain welcome text, footer should contain copyrights statement, and the menu should contain direct links to others pages of the Web application.

Hint: Use `tiles:getAsString` and `tiles:insert` to create the Web page.

Solution:

1. layout.jsp
2. header.jsp
3. body.jsp
4. menu.jsp
5. footer.jsp
6. main.jsp

// layout.jsp

```
<%@ page language="java" %>
<%@ taglib uri="/WEB-INF/struts-tiles.tld" prefix="tiles" %>

<!doctype …>
<html>
   <head><title><tiles:getAsString name="title"/></title></head>
   <body>
      <tiles:insert attribute="header"/>
      <p>
      <table border=2 align="center">
         <tr>
            <th>
               <font size="+1">
               <tiles:getAsString name="title"/>
```

```
            </font>
            </th>
         </tr>
      </table>
      <p>
      <table width=25 align="left" cellspacing="5">
         <tr>
            <td>
               <tiles:insert attribute="menu"/>
            </td>
         </tr>
      </table>
      <center>
         <table>
            <tiles:insert attribute="body"/>
            <br clear="all">
         </table>
      </center>
      <br>
      <tiles:insert attribute="footer"/>
   </body>
</html>
```

Enter the code in a Notepad and save the file as `layout.jsp` in `%TOMCAT_HOME%/webapps/ struts-test/WEB-INF/tiles.`

// header.jsp

```
<table border=1 width="100%" bgcolor="#f1f1f1">
```

```
   <tr>
      <th>
         <marquee> Welcome to Online banking.....</marquee>
      </th>
   </tr>
</table>
```

Enter the code in a Notepad and save the file as `header.jsp` in `%TOMCAT_HOME%/webapps/ struts-test/WEB-INF/tiles.`

// body.jsp

```
<table border=0 width="75%">
<tr>
   <th>Marko Bank has tied up with more than 75 organizations to
facilitate online shopping for all its Internet Banking Customers.
Choose your products online and pay conveniently through Marko Bank
Internet Banking Service.
   </th>
   </tr>
</table>
```

Enter the code in a Notepad and save the file as `body.jsp` in `%TOMCAT_HOME%/webapps/ struts-test/WEB-INF/tiles.`

// menu.jsp

```
<a href="<html:rewrite page='/home.html'/>">home</a>
<a href="<html:rewrite page='/contact.html'/>">contact</a>
<a href="<html:rewrite page='/privacy.html'/>">privacy</a>
```

Enter the code in a Notepad and save the file as `menu.jsp` in `%TOMCAT_HOME%/webapps/ struts-test/WEB-INF/tiles.`

// footer.jsp

```
<table border=1 width="100%" bgcolor="#f1f1f1">
   <tr>
      <th>Copyright © Aptech, Inc.</th>
   </tr>
</table>
```

Enter the code in a Notepad and save the file as `footer.jsp` in `%TOMCAT_HOME%/webapps/ struts-test/WEB-INF/tiles.`

// main.jsp

```
<%@ page language="java" %>
<%@ taglib uri="/WEB-INF/struts-tiles.tld" prefix="tiles" %>

<tiles:insert page="/WEB-INF/tiles/layout.jsp">
   <tiles:put name="title" value="SBC Bank"/>
   <tiles:put name="header" value="/WEB-INF/tiles/header.jsp"/>
   <tiles:put name="menu" value="/WEB-INF/tiles/menu.jsp"/>
   <tiles:put name="body" value="/WEB-INF/tiles/body.jsp"/>
   <tiles:put name="footer" value="/WEB-INF/tiles/footer.jsp"/>
</tiles:insert>
```

Enter the code in a Notepad and save the file as `main.jsp` in `%TOMCAT_HOME%/webapps/struts-test`.

The output of the program is as shown in Figure 24.1.

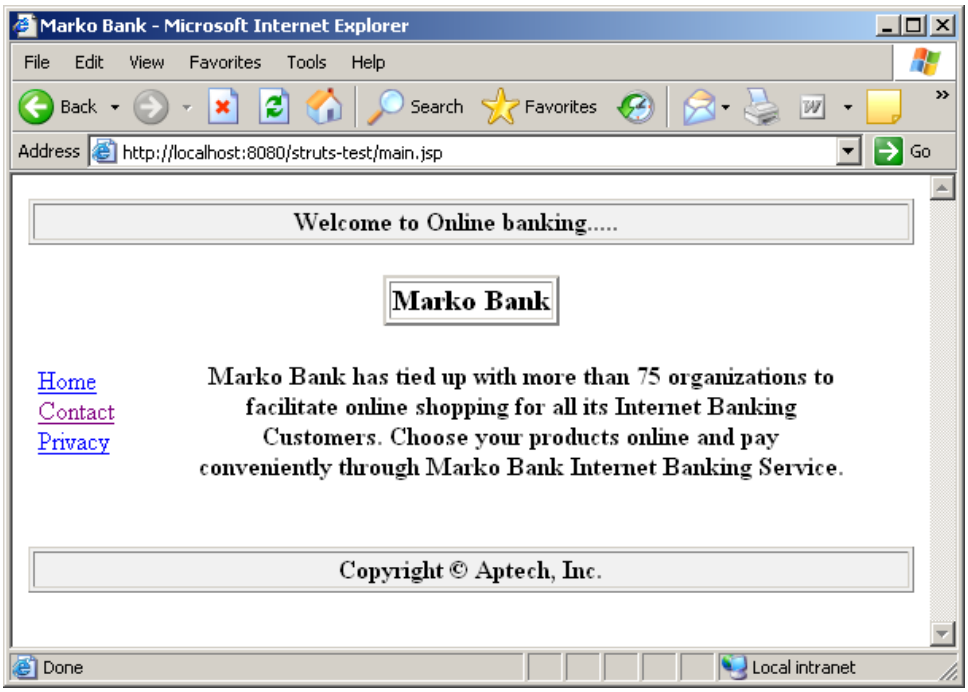

Figure 24.1: About us

Do It Yourself

1. Create a Web page with five regions; header, title, body, footer, and menu. The header should contain the company logo, and the footer should contain direct links to others pages of the Web application.

   Hint: Use `tiles:getAsString` and `tiles:insert` to create the Web page.

Solution:

The list of files, required for this application, is as followed:

1. layout.jsp
2. header.jsp
3. body.jsp
4. footer.jsp

// layout.jsp

```
<%@ page language="java" %>
<%@ taglib uri="/WEB-INF/struts-tiles.tld" prefix="tiles" %>

<!doctype …>
<html>
   <head><title><tiles:getAsString name="title"/></title></head>
   <body>
      <tiles:insert attribute="header"/>
      <p>
         <table border=2 align="center">
         <tr><th><font size="+1">
         <tiles:getAsString name="title"/>
         </font></th></tr></table>
      <p>
      <table width=25 align="left" cellspacing="5">
         <tr>
         <td><tiles:insert attribute="menu"/></td>
         </tr>
      </table>
      <center>
```

```
        <table>
           <tiles:insert attribute="body"/>
           <br clear="all">
        </table>
     </center><br>
     <tiles:insert attribute="footer"/>
   </body>
</html>
```

Enter the code in a Notepad and save the file as `layout.jsp` in `%TOMCAT_HOME%/webapps/ images/WEB-INF/tiles.`

// header.jsp

```
<table border=1 width="100%" bgcolor="#f1f1f1">
<tr>
   <th>
   <img src="<html:rewrite page='/images/sbc.jpg'/>" alt="bank">
   </th>
</tr>
</table>
```

Enter the code in a Notepad and save the file as `header.jsp` in `%TOMCAT_HOME%/webapps/ images/WEB-INF/tiles.`

// body.jsp

```
<table>
```

```
<tr>

   <th>SBC Bank has tied up with more than 75 organizations to
facilitate online shopping for all its Internet Banking customers.
Choose your products online and pay conveniently through SBC Bank
Internet Banking Service.

   </th>

</tr>

</table>
```

Enter the code in a Notepad and save the file as `body.jsp` in `%TOMCAT_HOME%/webapps/ images/WEB-INF/tiles.`

// footer.jsp

```
<table border=1 width="100%" bgcolor="#f1f1f1">

   <tr>

      <th>

         <a href="<html:rewrite page='/home.html'/>">home

         </a>     

         <a href="<html:rewrite page='/contact.html'/>">contact

         </a>     

         <a href="<html:rewrite page='/privacy.html'/>">privacy

         </a>

      </th>

   </tr>

</table>
```

Enter the code in a Notepad and save the file as `footer.jsp` in `%TOMCAT_HOME%/webapps/ images/WEB-INF/tiles.`

The output of the program is as shown in Figure 24.2.

Figure 24.2: Home page