Tutorial 8 - Java Web Applications Using JavaBeans

by Dr. Wenjie He

1. Introduction to JavaBeans

JavaBeans are Java software components used in Java software development.

Properties of JavaBeans

1. The JavaBeans specification provides a list of naming rules, so that the components built following the rules can be recognized by the tools implementing the specification.
2. JavaBeans are Java classes that have *properties* (as `private` instance variables).
3. The methods that change a property's value are called *setter* methods, and the methods that retrieve a property's value are called *getter* methods.
4. JavaBeans must be serializable.
5. JavaBeans with setters are *mutable* objects; and JavaBeans without setters could be *immutable* objects.

Rules for JavaBeans

1. If the property is not a `boolean`, the getter method's prefix must be `get`.
2. The name of the property is *inferred* from getters and setters, not through any variables in your class.
3. If the property is a `boolean`, the getter method's prefix is either `get` or `is`.
4. The setter method's prefix must be `set`.
5. *property name with its first letter to uppercase* = *getter/setter removing prefix* `get, is` *or* `set`
6. The signature of a setter method: `public void set`*`PropertyName`*`(`*`PropertyName`*`)`
7. The signature of a getter method: `public `*`PropertyName`*` get`*`PropertyName`*`()` or `public boolean is`*`PropertyName`*`()`
8. Listener method names used to add a listener must use the prefix `add`, followed by the listener type.
9. Listener method names used to remove a listener must use the prefix `remove`, followed by the listener type.
10. The type of the listener to be added or removed must be passed as the argument to the method.
11. Listener method names must end with the word `Listener`.

2. Get the example

The source code is included in `Tutorial8_JavaBeans.zip`.

1. Get the example in the folder: `Tutorial8_JavaBeans\GuestBook`.
2. Examine the files in this example:
   -

The database script: `Tutorial8_JavaBeans\GuestBook\db\guestbookdb.sql`

- Two JSP files: `Tutorial8_JavaBeans\GuestBook\jsp\guestBookView.jsp, guestBookLogin.jsp`
- Two JavaBeans: `Tutorial8_JavaBeans\GuestBook\src\org\me\jsp\beans\GuestBean.java`, `GuestDataBean.java`

3. Creating the database

1. Create the database `guestbookdb` using the script `Tutorial8_JavaBeans\GuestBook\db\guestbookdb.sql`.

4. Running the example

1. Create a **Dynamic Web Project** called `GuestBook` in the **Eclipse for Java EE**.

   Note: When you create the project, remember to generate the `web.xml`. We will use it to configure our welcome file.

2. Create a package called `org.me.jsp.beans`.

3. Copy the files `GuestBean.java` and `GuestDataBean.java` from the folder `GuestBook\src\org\me\jsp\beans` to the package `org.me.jsp.beans` by drag-and-drop.

4. Copy the files `guestBookLogin.jsp` and `guestBookView.jsp` from the folder `GuestBook\jsp` to the node `WebContent` under the project.

5. Expand the `WEB-INF` node under the node `WebContent`, and double-click `web.xml` to open it.

6. Add the file `guestBookLogin.jsp` to the list of your welcome files by, for example, changing the following line

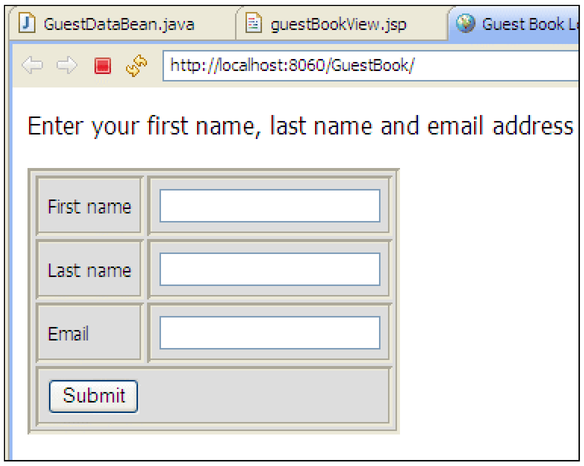
   ```
   <welcome-file>index.jsp</welcome-file>
   ```

   to

   ```
   <welcome-file>guestBookLogin.jsp</welcome-file>
   ```

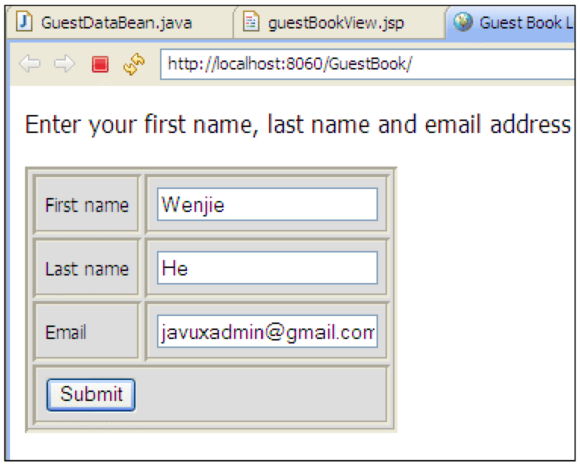
7. Copy the MySQL driver's library file `Tutorial8_JavaBeans\GuestBook\lib\mysql-connector-java-5.1.7-bin.jar` into your `WebContent\WEB-INF\lib` node.

8. Run the project, you would see



Enter your first name, last name and email address

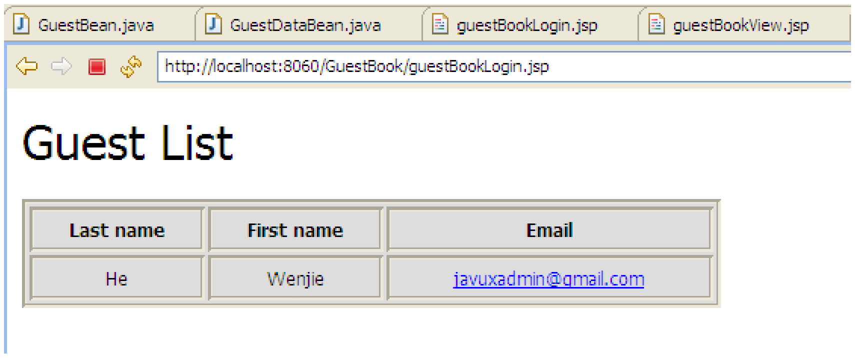
First name	
Last name	
Email	
Submit	



Enter your first name, last name and email address

First name	Wenjie
Last name	He
Email	javuxadmin@gmail.com
Submit	

After you enter the data for a guest and submit, you would see,



Guest List

Last name	First name	Email
He	Wenjie	javuxadmin@gmail.com

5. Creating JavaBeans

In this example, there are two JavaBeans: `GuestBean` for data storage and `GuestDataBean` for data access.

Our data structure: `GuestBean`

Code Listing: `GuestBean.java`

```
package org.me.jsp.beans;

public class GuestBean {
   private String firstName, lastName, email;

   public void setFirstName( String name )
   {
      firstName = name;
   }

   public String getFirstName()
   {
      return firstName;
   }

   public void setLastName( String name )
   {
      lastName = name;
   }

   public String getLastName()
   {
      return lastName;
   }

   public void setEmail( String address )
   {
      email = address;
   }

   public String getEmail()
   {
      return email;
   }
}
```

Our *Data Access Object* (DAO): `GuestDataBean`

Code Listing: `GuestDataBean.java`

```

package org.me.jsp.beans;

// Java core packages
import java.sql.*;
import java.util.*;
import java.net.*;

import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.sql.DataSource;

public class GuestDataBean {
   private Connection connection;
   private PreparedStatement addRecord, getRecords;

   // construct TitlesBean object
   public GuestDataBean() throws Exception {

      Class.forName("com.mysql.jdbc.Driver");
      connection = DriverManager.getConnection(
                     "jdbc:mysql://localhost:3306/guestbookdb",
                     "root", "");

      getRecords = connection.prepareStatement(
                    "SELECT firstName, lastName, email FROM guests");

      addRecord = connection.prepareStatement(
                   "INSERT INTO guests ( "
                  + "firstName, lastName, email ) "
                  + "VALUES ( ?, ?, ? )");

   }

   // return an ArrayList of GuestBeans
   public List<GuestBean> getGuestList() throws SQLException {
      List<GuestBean> guestList = new ArrayList<GuestBean>();

      // obtain list of titles
      ResultSet results = getRecords.executeQuery();

      // get row data
      while (results.next()) {
         GuestBean guest = new GuestBean();

         guest.setFirstName(results.getString(1));
         guest.setLastName(results.getString(2));
         guest.setEmail(results.getString(3));

         guestList.add(guest);
      }

      return guestList;
   }

   // insert a guest in guestbook database
   public void addGuest(GuestBean guest) throws SQLException {
      addRecord.setString(1, guest.getFirstName());
      addRecord.setString(2, guest.getLastName());
      addRecord.setString(3, guest.getEmail());

      addRecord.executeUpdate();
   }

   // close statements and terminate database connection
   protected void finalize() {
      // attempt to close database connection
      try {
         getRecords.close();
         addRecord.close();
         connection.close();
      }

      // process SQLException on close operation
      catch (SQLException sqlException) {
         sqlException.printStackTrace();
```

```
            }
        }

    }
```

1. (Lines 19-30) We put the database connection and query statement definitions inside the constructor, so that when we destroy the object, we can release the recourse occupied by the connection and query objects. Later if we need to access the database again, we can create the database connection by calling the constructor.
2. (Lines 35-36) Here we used the *Java Generic Types* in `List<GuestBean>`. The *Java Generic Types* were designed to reduce the bugs in your code.
3. For the Java collection classes, such as `List, Map, Vector`, etc., they can hold any Java classes. This kind of flexibility may cause confusion sometimes. The Java generics add stability to your code by avoiding this kind of uncertainty.
4. The idea of the Java generics is to convert some Java runtime errors to compile time errors, so that it is much easier to detect those errors.

6 Using JavaBeans in JSPs

In the `guestBookLogin.jsp`, we collect user entered data and store it in a JavaBean. Then we called the data access JavaBean to write the data into the database.

Code Listing: `guestBookLogin.jsp`

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

<%-- beans used in this JSP --%>
<jsp:useBean id = "guest" scope = "page"
   class = "org.me.jsp.beans.GuestBean" />
<jsp:useBean id = "guestData" scope = "request"
   class = "org.me.jsp.beans.GuestDataBean" />

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

<head>
   <title>Guest Book Login</title>

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

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

<body>
   <jsp:setProperty name = "guest" property = "*" />

   <% // start scriptlet

      if ( guest.getFirstName() == null ||
           guest.getLastName() == null ||
           guest.getEmail() == null ) {

   %> <%-- end scriptlet to insert fixed template data --%>

         <form method = "post" action = "guestBookLogin.jsp">
            <p>Enter your first name, last name and email
            address to register in our guest book.</p>

            <table>
               <tr>
                  <td>First name</td>

                  <td>
                     <input type = "text" name = "firstName" />
                  </td>
               </tr>

               <tr>
                  <td>Last name</td>

                  <td>
                     <input type = "text" name = "lastName" />
                  </td>
               </tr>

               <tr>
                  <td>Email</td>

                  <td>
                     <input type = "text" name = "email" />
                  </td>
               </tr>

               <tr>
                  <td colspan = "2">
                     <input type = "submit"
```

```
                           value = "Submit" />
                  </td>
               </tr>
            </table>
         </form>

      <% // continue scriptlet

         }  // end if
         else {
            guestData.addGuest( guest );

      %> <%-- end scriptlet to insert jsp:forward action --%>

         <%-- forward to display guest book contents --%>
         <jsp:forward page = "guestBookView.jsp" />

      <% // continue scriptlet

         }  // end else

      %> <%-- end scriptlet --%>
   </body>
   </html>
```

1. (Line 6)   The `<jsp:useBean>` action is used to create a `GuestBean` instance.
2. (Line 6)   The `scope` attribute tells us that this page has `page` scope. It means that only this page can access the JavaBean of type `GuestBean` with reference `guest`.
3. (Lines 6-7)   The Java implementation for these two lines is:

   ```
   GuestBean guest = new GuestBean();
   ```

   The JSP container does this conversion.
4. (Line 8)   A `GuestDataBean` instance is created with the `request` scope. A `request` may contain several pages within one request. In this example, the two JSP pages are in the same request scope, because we use `forward` in the first JSP to go to the second JSP. Similarly, we have

   ```
   GuestDataBean guestData = new GuestDataBean();
   ```

5. (Line 31)   The `<jsp:setProperty>` action sets the properties for the JavaBean `guest` of type `GuestBean`.

6. (Line 31) The `property="*"` means that all the properties of the `guest` JavaBean have the same names as the corresponding `input` fields in the HTML form.

   In other words, the form fields and the JavaBean properties have the same names.

7. (Line 31) If a property name of a JavaBean is different from the name of its corresponding input field, we need to set the property as follows,

   ```
   <jsp:setProperty name="guest" property="thisProperty"
           param="thatProperty" />
   ```

8. (Line 83) When the bean instance `guest` carries the data (first name, last name, and email), the DAO bean `guestData` is used and its data access method `addGuest()` is called to persist the data in `guest`.

9. (Line 88) The `<jsp:forward>` action is used to forward the request to the next page `guestBeansView.jsp`, and `guestBeansView.jsp` still stays in the same request. Therefore, `guestBeansView.jsp` can access the `request` scope `GuestDataBean` instance `guestData`.

Code Listing: `guestBookView.jsp`

```

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

<%-- page settings --%>
<%@ page import = "java.util.*" %>
<%@ page import = "org.me.jsp.beans.*" %>

<%-- GuestDataBean to obtain guest list --%>
<jsp:useBean id = "guestData" scope = "request"
   class = "org.me.jsp.beans.GuestDataBean" />

<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;
         }
```

```
30          </style>
31       </head>
32
33       <body>
34          <p style = "font-size: 2em;">Guest List</p>
35
36          <table>
37             <thead>
38                <tr>
39                   <th style = "width: 100px;">Last name</th>
40                   <th style = "width: 100px;">First name</th>
41                   <th style = "width: 200px;">Email</th>
42                </tr>
43             </thead>
44             <tbody>
45
46             <% // start scriptlet
47
48                List<GuestBean> guestList = guestData.getGuestList();
49
50                for (GuestBean guest : guestList) {
51             %> <%-- end scriptlet; insert fixed template data --%>
52                   <tr>
53                      <td><%= guest.getLastName() %></td>
54                      <td><%= guest.getFirstName() %></td>
55                      <td>
56                         <a href = "mailto:<%= guest.getEmail() %>">
57                            <%= guest.getEmail() %></a>
58                      </td>
59                   </tr>
60             <% // continue scriptlet
61                } // end for
62             %> <%-- end scriptlet --%>
63             </tbody>
64          </table>
65       </body>
66

   </html>
```

This JSP accesses the database to retrieve all the guest records and display them.

1. (Lines 9-10)   Since the `GuestDataBean` instance `guestData` was created in the previous page, here the JSP does not create a new bean, it just uses the existing bean in the request scope.
2. (Line 47)   We use the DAO bean `GuestDataBean` to access the database to retrieve data.
3. In this example, you can see there is no servlet. We do not use servlet to access the database, we use a DAO JavaBean to access the database. In this way, we can even eliminate the servlets.

4. Actually, many beginners tend to use the method in this example to develop Java web applications due to its simplicity. But there is some problem. In this way, you may not use a clear MVC structure. In deed, you mix the view and controller together, because in your JSP you do both data preparation and request dispatching as you see here. From the long run, separating the MVC layers clearly would make your software maintenance easier in general.
5. One can convert this example by introducing a servlet as a controller.

==========The End==========