Dynamic data handling with Struts 2

Explore powerful, easy-to-use features

Ina Arya

June 15, 2010

In this article, learn how Apache Struts 2 can help you handle dynamic data for your Web-based applications. Walk through a few of the most common use cases with examples and sample code. Explore how the powerful Action, Interceptor, and Results features help you set up UI fields with database data. Also learn about modularizing the application by setting the data value when the application loads from a list of values and about pre-filling fields with the desired data.

Introduction

Almost every Web-based business application requires data storage in various databases. For Struts 2 beginners, the biggest challenge while developing a Web application is displaying the data stored in their database to the presentation page of the browser.

Apache Struts 2 is an open source extensible framework for creating enterprise-ready Java™ Web applications. The framework is designed to streamline the full development cycle, from building, to deploying, to maintaining applications over time. Struts 2 was originally known as WebWork 2.

Developing the best approach for handling data can require a lot of effort. Great applications require a good user interface (UI) and good back-end implementation. The power of Struts 2 lies in its ability to provide clean implementation of the Model-View-Controller (MVC) pattern with the following features:

Actions
Part of the framework's architecture, Actions provides clean implementation by keeping all the business logic in one place. Actions serves as a channel of automatic transfer of data from the request to result pages.

Results
A UI that represents the user's requirements from the application onto the Web browser.

Interceptors
Remove the overhead of Actions and improve performance by implementing common operations, such as validations, logging statements, and error checks, which are common to multiple Actions.
Interceptors are called at both post-processing and pre-processing. For one action request, there can be a stack of Interceptors.

**ValueStack**
An object that holds all the data that gets associated during the processing of a request. Figure 1 shows the components of the Struts 2 framework.

**Figure 1. Struts 2 Action, Interceptor, and Result**

In this article, learn about dynamic data handling in a Web-based application. You will explore how to send user data at run time with a few special use cases, such as pre-filling a form with user data values and setting the UI field to display a default value from a list of values. The article focuses on the implementation details of the flow of dynamic data between Struts 2 Action classes and the result pages.

**Dynamic data handling using Struts 2 ValueStack**

ValueStack, one of the powerful features of Struts 2, holds all the data during processing of a request. Action classes also store data in the form of beans. (In Struts 2 you use getter and setter methods.) Once the data is stored in the action class objects, Struts 2 puts that data onto the ValueStack in the form of the Action object. ValueStack stores a stack of objects, and this data will be exposed to the other parts of the framework.

In Figure 2, the action class sets the values of two variables using getter and setter methods. This data will be saved to the ValueStack during processing of requests. HelloWorld.jsp will get the values of these variables from the ValueStack and display the values using the <s:property> tag from the Struts 2 tag library.
Figure 2. Data on ValueStack

```java
public class HelloWorldAction {
    public void setStr() {
        this.str = "Hello World:";
    }
    public String getStr() {
        return str;
    }
    public void setNewStr() {
        this.newStr = "Hi World";
    }
    public String getNewStr() {
        return newStr;
    }
}
```

ValueStack

HelloWorld object
- String str
- String newStr

HelloWorld.jsp
- `<s:property value="str"/>
- `<s:property value="newStr"/>

Hello World
- Hi World

Static data versus dynamic data

If data that will be displayed on the UI is known in advance, it can be called static data. Static data can be handled easily by coding it directly in JavaServer Pages (JSP). Figure 3 shows a static data situation.

Figure 3. Static data

If a user wants to display the data retrieved from the user database, you need a mechanism to appropriately handle the data in the Action and JSP and to display the data on the UI. With Struts 2, the building blocks for this type of scenario are the Action class and ValueStack. By using getter and setter methods, the Action class will save the user values onto the ValueStack. After that, a JSP will get the values of matching variable names and display them on the UI. See Figure 4.
Setting UI fields and tags with user data

Struts 2 provides a tag library that displays data in rich and reusable HTML to help you create interactive, form-based applications. Struts 2 comes with generic tags (control tags and data tags) and UI tags. UI tags are designed to get data from the ValueStack (action added data onto the ValueStack) or from data tags. The following is an example of the Struts 2 UI tag <select tag>.

1. Write the **Action** class to display the data onto the UI, as shown in Listing 1.

   **Listing 1. Write the Action class to display the data onto the UI**
   
   ```java
   public class DisplayAction extends ActionSupport {
       private List state = new ArrayList();
       private List status = new ArrayList();

       public String execute() {
           setState(this.state);
           setStatus(this.status);
           return "success";
       }

       //getter and setter method for state variable//
       public List getState() {
           return state;
       }

       public void setState(List state) {
           //implement the application specific logic to retrieve the data from the database here//

           //Here for displaying the data on UI, we are using few hardcoded values//
           state.add("Defect");
           state.add("Task");
   ```
In this example, the DisplayAction class sets the ArrayList object's state and status with a few values. The Action class will save this data onto the ValueStack, which JSP will use to display on the UI.

2. To handle the data on the JSP side, you need to write <select> tags that will get the values of state and status, as follows in Listing 2:

Listing 2. Write <select> tags that will get the values of state and status

```html
<%@ page language="java" contentType="text/; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
<html>
<title>Sample Display Page</title>
<body>
<s:select list="%{status}" name="Status"></s:select>
<s:select list="%{state}" name="State"></s:select>
</body>
</html>
```

3. Configure the Action and JSP with struts.xml mapping, as shown in Listing 3.

Listing 3. Configure the Action and JSP with struts.xml mapping

```xml
<!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN" "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
  <package name="sample.example" extends="struts-default">
    <action name="DisplayAction" class="com.sample.example.DisplayAction">
      <result name="success">/DisplayPage.jsp</result>
    </action>
  </package>
</struts>
```

In the code examples above:
• The Action class has the list variables named state and status.
• Data is added in the Action class by making state and status of type arraylist.
• `<s:select>` has the list parameter (mandatory) with the same name as defined in the Action class.
• Place these list variables using `%(<list-variable>)`.
• The name of the list variables in JSP must match the name given in the Action class.

Figure 5 shows an example of the output.

**Figure 5. Output 1: Display of Status list**

![Status list](image1)

In Figure 6, the `<select>` tag status is initialized with the values that are put in the Action class.

**Figure 6. Output 2: Display of State list**

![State list](image2)

The `<select>` tag state is also initialized with the values that are put in the Action class.

**Setting default values in prefilled UI fields**

Assume a user has a list of values. From the elements in the list, the user wants to set one value as the default once the UI gets loaded.

In the following example, the user has a combo box and wants to display the last value after the UI is loaded.

1. Modify the Action class and add code for the default value, as shown in Listing 4.

**Listing 4. Modify the Action class and add code for the default value**

```java
public class DisplayAction extends ActionSupport {
    private List state = new ArrayList();
    private List status = new ArrayList();
    private String defaultState = "";
    private String defaultStatus = "";

    public String execute()
    {
        //set status and state list values
        setState(this.state);
        setStatus(this.status);

        //set default values for state and status which
        //a user wants to display on the User Interface
        setDefaultState("Task");
        setDefaultStatus("Closed");
    }
}
```
return "success";
}

//getter and setter method//
public List getState() {
    return state;
}

public void setState(List state) {
    //implement the application specific logic to retrieve
    //the data from the database here/

    //Here for displaying the data, we are using few
    //hardcoded values
    state.add("Defect");
    state.add("Task");
    state.add("User Stories");
}

public List getStatus() {
    return status;
}

public void setStatus(List status) {
    //implement the application specific logic to retrieve
    //the data from the database here/

    //Here for displaying the data, we are using few
    //hardcoded values/

    status.add("New");
    status.add("Fixed");
    status.add("Resolved");
    status.add("Closed");
}

public String getDefaultState() {
    return defaultState;
}

public void setDefaultState(String defaultState) {
    this.defaultState = defaultState;
}

public String getDefaultStatus() {
    return defaultStatus;
}

public void setDefaultStatus(String defaultStatus) {
    this.defaultStatus = defaultStatus;
}
}

In the Action class above, the example adds two more getter/setter methods to set variables with the default values for the <select> tag.

2. The modified JSP code is shown below in Listing 5.
Listing 5. The modified JSP code

```html
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags" %>

<html>
<title>Sample Display Page</title>
<body>

<s:select list="%{status}" name="Status" value="%{defaultStatus}"></s:select>
<s:select list="%{state}" name="State" value="%{defaultState}"></s:select>

</body>
</html>
```

The struts.xml mapping remains the same in this case.

In the code examples above:

- In the Action class, the default values to be displayed are saved in two string variables.
- In the JSP, these values can be handled using the value parameter of the `<select>` tag.
- The name of the default value variable in the Action class must match the name in the result JSP.

Figure 7 shows an example of the output.

**Figure 7. Output 1: Display of Status default value**

![Status: Closed](status.png)

In Figure 8, Closed has been set as the default value of the Status list when the UI is loaded.

**Figure 8. Output 2: Display of State default value**

![State: Task](state.png)

Task has been set as the default value of the State list when the UI is loaded.

**Summary**

Struts 2 is a second-generation Web application framework that has gained popularity over Java Servlets. It encompasses the commonly used MVC design pattern. Struts 2 automates common tasks, freeing you up to take advantage of the Action-to-result pages mapping using SML-based
configurations. The framework covers the ground-level details of the implementation of Web-based applications and provides a piece of structural software.

In this article you learned about Struts 2 abilities as a complete framework. You explored a few of the most common use cases and the solutions provided with the powerful features of Struts 2.
Related topics

- Become more familiar with Apache Struts 2, the free open source framework for creating Java Web applications.
- Get the Struts 2 documentation: Getting started, Tutorials, FAQs, and Guides.
- Read more about ValueStack, which allows multiple beans to be pushed in and dynamic EL expressions to be evaluated against it.
- Use the Struts 2 UI Tag Reference. Struts UI tags display data in rich and reusable HTML.
- Get Struts 2 documentation and other resources, such as security bulletins, wikis, quick links, and tutorials.
- Evaluate IBM products in the way that suits you best: Download a product trial, try a product online, use a product in a cloud environment, or spend a few hours in the SOA Sandbox learning how to implement Service Oriented Architecture efficiently.

© Copyright IBM Corporation 2010
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)