

Databinding: Passing Data Between Controller Classes and JSP Pages

Table of contents

1 Introduction.....	2
2 Summary of Binding Contexts.....	2
3 Individual Binding Contexts.....	4
3.1 actionForm	4
3.2 applicationScope	5
3.3 bundle	5
3.4 container	5
3.5 pageFlow	6
3.6 pageInput	7
3.7 requestScope	8
3.8 sessionScope	8
3.9 sharedFlow	8
3.10 url	9
4 Related Topics.....	9

1. Introduction

Your web application's presentation layer (JSP pages) and processing layer (Controller classes) are bound together through the process of *databinding*. There are numerous databinding contexts available, including all of the JSP 2.0 implicit objects as well as other contexts peculiar to page flow apps.

Databinding syntax uses the JSP 2.0 Expression Language. For a quick summary of the JSP 2.0 EL see [JSP 2.0: The New Deal, Part 1](http://www.onjava.com/pub/a/onjava/2003/11/05/jsp.html) (<http://www.onjava.com/pub/a/onjava/2003/11/05/jsp.html>) .

It is important to note that the databinding contexts form a two-way street between the presentation and processing layers of a web application. Databinding contexts not only pass data from the processing layer to the presentation layer, some contexts can pass data in the opposite direction from the presentation layer to the processing layer.

The one-way, read-only contexts are: requestScope, url, sessionScope, container, pageContext, bundle, pageInput

The two-way, read-write contexts are: pageFlow, sharedFlow, actionForm

2. Summary of Binding Contexts

Context Name	Object the Context References	Description
actionForm	The current form bean (the form bean named in the current <netui:form> tag)	Individual fields defined in the form bean can be referenced. Values are read-write.
applicationScope	an attribute map on the application object (a JSP 2.0 implicit object)	
bundle	Properties defined in a message resource file	On a JSP page, declare the message resource file using the <netui-data:declareBundle> tag. <netui-data:declareBundle name="myBundle" bundlePath="properties.bundle1"/> Reference individual properties in the file using the bundle databinding context. <netui:span value="{bundle.myBundle.message1}"/>

Databinding: Passing Data Between Controller Classes and JSP Pages

container	The data set referenced in the current <code><netui-data:dataGrid></code> or <code><netui-data:repeater></code> tag	<p>In the example below, the container context refers to the object named in the <code><netui-data:dataGrid></code>'s <code>dataSource</code></p> <pre><netui-data:dataGrid dataSource="pageScope.stocks" name="portfolio"> <netui-data:rows> <netui-data:spanCell value="\${container.item.symbol}"/> <netui-data:spanCell value="\${container.item.price}"/> </netui-data:rows> </netui-data:dataGrid></pre> <p>See below for detailed information on container's sub-contexts.</p>
pageFlow	The current Controller class	<p>Public members of the current Controller class can be referenced and written to. Assume that the Controller class contains the following public member String.</p> <pre>public String someText = "This is some text"</pre> <p>Then that member can be referenced from any JSP page in the page flow.</p> <pre><netui:span value="\${pageFlow.someText}"/></pre>
pageInput	an attribute map on the current Forward object	<p>Page input data can be added to the Forward object using the <code>addActionOutput(String, Object)</code> method..</p> <pre>Forward forward = new Forward("success"); forward.addActionOutput("name", someNameData);</pre> <p>The data can be retrieved on the JSP page using the <code>pageInput</code> databinding context.</p> <pre><netui:span value="\${pageInput.name}"/></pre>
pageScope	an attribute map on the current JSP page (a JSP 2.0 implicit object)	

requestScope	an attribute map on the request object (a JSP 2.0 implicit object)	
sessionScope	an attribute map on the session object (a JSP 2.0 implicit object)	
sharedFlow	Attribute map of all of the shared flows in the web application	Provided that the shared flow is declared in the Controller file <pre>@Jpf.Controller(sharedFlowRefs={ @Jpf.SharedFlowRef(name="mySharedFlow", type=sharedFlows.SharedFlow.class) }) public class Controller extends PageFlowController You can access the members of the shared flow through the sharedFlow context. <netui:span value="\${sharedFlow.mySharedFlow.someMember}" /></pre>
url	Query parameters on the current JSP's URL.	Query parameters that are part of the URL can be referenced. Values are read only.

3. Individual Binding Contexts

3.1. actionForm

To bind to data in a form bean, you use the `actionForm` context. In the following example a JSP's form fields are pre-populated by binding to the fields of a form bean. Assume that you navigate to the JSP page via the following action method. Notice that the `Forward` object is constructed with two parameters: the first parameter is the string "update", the second parameter is a form bean (`DatabaseForm`). By constructing the `Forward` object in this way, the `actionForm` context is automatically filled with the form bean values, values which can be retrieved by the navigated-to JSP page.

```
@Jpf.Action(  
    forwards={  
        @Jpf.Forward( name="update", path="updateItems.jsp" )  
    }  
)  
public Forward updateItems(DatabaseForm form)  
{
```

Databinding: Passing Data Between Controller Classes and JSP Pages

```
...
    // Read from the database
    // and apply the values to the form bean.
    form.applyValuesToForm(getCurrentRow());

    // Construct a Forward object using the form bean.
    return new Forward("update", form);
}
```

The action method above reads a record from a database using a Database control (the control code is omitted in the example) and populates the fields in the form bean with the record data before navigating to the JSP `updateItems.jsp`. The values are retrieved by the JSP page, using the `actionForm` context:

```
<netui:form action="submitUpdate">
    <netui:content value="{actionForm.itemnumber}"/>
    <netui:textBox dataSource="{actionForm.itemname}"/>
    ...
</netui:form>
```

When the `<netui:form>` is submitted (`submitUpdate`), you can then write the updated values back to the database:

```
@Jpf.Action(
    forwards={
        @Jpf.Forward( name="updated", path="getItems.jsp" )
    }
)
public Forward submitUpdate(DatabaseForm aDatabaseForm)
{
    // code that calls the database control and updates the database
    return new Forward("updated");
}
```

3.2. applicationScope

3.3. bundle

3.4. container

The container refers to the data set passed into a `<netui-data>` or `<netui-data:repeater>` tag. The container allows access to the individual rows and cells of the data set.

The container context has the following properties.

Property Name	Description
<code>container.item</code>	Refers to the current row of the data set.
<code>container.item.[field_name]</code>	Refers to an individual field of the current row.

<code>container.index</code>	Refers to the integer index of the current row.
------------------------------	---

The following example shows how to access the name, type breed, and weight properties in a data set passed to a data grid.

The context container refers to the data set `pageInput.petList`.

```
<netui-data:dataGrid dataSource="pageInput.petList" name="petGrid">
  <netui-data:rows>
    <netui-data:spanCell value="{container.item.name}" />
    <netui-data:spanCell value="{container.item.type}" />
    <netui-data:spanCell value="{container.item.breed}" />
    <netui-data:spanCell value="{container.item.weight}" />
  </netui-data:rows>
</netui-data:dataGrid>
```

3.5. pageFlow

When you define a variable in your controller class, you can access this variable from any JSP page that is part of that page flow. For example, if the JPF file contains this code:

```
public class SimpleflowController extends PageFlowController
{
    public String labelName = "This is a label"
    ...
}
```

You can access this variable on a JSP, for instance by using a `<netui:label>`:

```
<netui:label value="{pageFlow.labelName}" />
```

The variable is read-write and can be changed, for instance in the controller's action method as is shown in the next example:

```
@Jpf.Action(
    forwards={
        @Jpf.Forward( name="success", path="nextPage.jsp" )
    }
)
public Forward labelAction(NameActionForm form)
{
    ...
    labelName = "Default Label";
    return new Forward( "success" );
}
```

It can also be changed in a JSP by using a form, as is shown next:

```
<netui:form action="submitAction">
  <netui:textBox dataSource="{pageFlow.labelName}" />
  ...
  <netui:button value="submit" />
</netui:form>
```

In this example, the text box will display the current value of the pageFlow's `labelName` variable when the form is first loaded, and upon submit the variable will hold the new value the user entered.

Accessing Page Flow Properties

In the example used above, the page flow property was defined as a public member variable in the page flow controller class. Alternatively you can define a private variable and provide access with a getter and setter method, as is shown in the following example:

```
private String firstName;

public String getFirstName() {
    return firstName;
}

public void setFirstName(String aStr) {
    firstName = aStr;
}
```

You can access this variable on a JSP exactly like before. For instance, you can bind the value to a `netui:label` as is shown next:

```
<netui:label value="{pageFlow.firstName}" />
```

In the example the method `getFirstName` is used to provide read access to the variable.

You can also define and access variables that are part of an object created in the controller class. You can provide access by using getter and setter methods, or by referencing the object and property name. The latter option is shown next. For instance, if the JPF file contains the code:

```
public class dataFlowController extends PageFlowController
{
    ...
    public Names myNames = new Names();
    ...
    public class Names implements Serializable {
        public String firstName;
        public String lastName;
    }
    ...
}
```

You can access the `firstName` property in a JSP like this:

```
<netui:label value="{pageFlow.myNames.firstName}" />
```

The comments in this section regarding access of page flow properties also apply to properties defined to the [sharedFlow](#) context.

3.6. pageInput

The `pageInput` context is filled with data by an action output in the Controller class.

```
Forward f = new Forward("next_page");
f.addActionOutput("message", sMessage);
```

The data can be retrieved on the JSP page as follows:

```
<netui:span value="{pageInput.message}" />
```

3.7. requestScope

3.8. sessionScope

You can use the `sessionScope` context to read data from the session object (an implicit JSP object). Assuming that the session object contains some data in its attribute map:

```
getSession().setAttribute("myAttributeKey", myObject);
```

then, you can read the data onto a JSP page as follows:

```
<netui:content
value="{sessionScope.myAttributeKey.someProperty}"></netui:content>
```

3.9. sharedFlow

If you need an attribute to be available across all the page flows in a web application's user (browser) session, such as a user's unique session key, you can use shared flow. By convention, shared flows are located in `WEB-INF/src/shared`. (For details on creating a shared flow, see [Shared Flow](#) (`../pageflow/sharedFlow.html`).) In addition to session-wide attributes, you can also use the shared flow to define fallback action methods and exception handlers.

If you define a variable in the shared flow class:

```
public class SharedFlow extends SharedFlowController
{
    public String defaultText = "Please Define";

    ...
}
```

and your controller file references the shared flow:

```
@Jpf.Controller(
    sharedFlowsRefs = {
        @Jpf.SharedFlowRef(name = "specificSharedFlow", type =
"SharedFlow.class")
    }
)
public class SomeController extends Controller
{
    ...
}
```


Databinding: Passing Data Between Controller Classes and JSP Pages

you can use the databinding context `shareFlow` in a JSP file:

```
<netui:textBox dataSource="{actionForm.firstName}"  
defaultValue="{sharedFlow.specificSharedFlow.defaultText}"/>
```

Notice that the `<netui:textBox>` tag in the example binds to a form bean to post data and binds to `sharedFlow`'s `defaultText` to receive its default value.

`SharedFlow` properties can be accessed from a JSP using the scope `sharedFlow`, for example,

```
dataSource="sharedFlow.specificSharedFlow.someProperty"
```

Actions declared in a `SharedFlow` are accessed by specifying a fully qualified action

```
action="specificSharedFlow.someAction"
```

where `specificSharedFlow` was the name given to the `sharedFlow` in the JPF annotation `Jpf.SharedFlowRef`.

3.10. url

4. Related Topics

For more information on the JSP 2.0 Expression Language, see [JSP 2.0: The New Deal, Part 1](http://www.onjava.com/pub/a/onjava/2003/11/05/jsp.html) (<http://www.onjava.com/pub/a/onjava/2003/11/05/jsp.html>)