Selecting Results

In the Coding Actions lesson, we created a Logon class that tests for input. In the *Selecting Results* lesson, we act on the outcome of that test.

Selecting an "Input" Result

After the Action processes a request, a result is selected to provide the response. A result may simply forward to an HTML page, a JavaServer page, a FreeMaker or Velocity template, or the result might construct a PDF or some other complex report (like JasperReports). There may be multiple results available to an action mapping. To indicate which one to select, the Action class returns a name corresponding to the appropriate result.

The Code

struts.xml

```
<action name="Logon" class="tutorial.Logon">
  <result type="redirectAction">Menu</result>
  <result name="input">/Logon.jsp</result>
</action>
```

How The Code Works

- If we enter a username and password into the form, the Logon Action will return "success".
    - "success" is the default result code, so the framework will use the "Menu" action as response. (Which we haven't written yet.)
- If we do not enter both credentials, the Logon Action will return "input", and the framework will use the Logon.jsp as the response.

In the Hello World lesson, our results used the default type, `Dispatcher`. The `Dispatcher` forwards to another web resource. Other kinds of views can be used by specifying a different result type.

The Logon mapping uses a different return type for "success" (the default result code). The `redirectAction` result type takes the name of an Action (as configured in the `struts.xml` file) as a parameter, and then issues a client-side redirect to the new action. As a result, the URI on the browser's location bar will change.

Using a Stub Page

As we develop web applications, we often need to make forward references – we need to refer to an action we haven't written yet. For example, in the first part of the lesson, the next step is to open the "Menu" page. If we Logon successfully, there will be no where to go, since "Menu" doesn't exist yet.

One way to work around this problem is to create a stub "Menu" page.

The Code

Missing.jsp

```
<html>
<head><title>Missing Feature</title></head>

<body>
<p>
    This feature is under construction.
    Please try again in the next iteration.
</p>
</body>
</html>
```

How the Code Works

- When the Login class returns "Menu", the framework will match it to our default wildcard mapping.
- The framework will return the stub "Menu.jsp" for now.

If you are not using wildcards, another way to inject a "missing" page would be to specify a `<default-action-ref>` element

Including a Missing Page

If you are building an application page by page, it can be worthwhile to setup a standard "Missing" page, and then include it from your stubs.

The Code

Menu.jsp

```
<%@ taglib prefix="s" uri="/struts-tags" %>
<s:include value="Missing.jsp" />
```

How the Code Works

- When the Menu.jsp renders, it will include the content of the standard Missing.jsp.

What to Remember

The framework offers a variety of result types. An Action can select the appropriate result by name, without actually knowing what result type will be rendered.

For more, see Result Types in the Core Developers Guide.

Next	Validating Input
Prev	Coding Actions