Struts Flow - Wizard Example

1. Wizard Example

This example shows how Struts Flow can be used to easily create a multi-page form, commonly called a wizard. The example uses a script called `wizard.js` which uses Struts Flow to define a `Wizard` object. The `Wizard` object handles displaying forms, automatic handling of forward-backward navigation buttons, and data model population. The validation and population processes are pluggable so custom code can be inserted at those steps.

To demonstrate the wizard, this example shows a user registration process with three screens: names, hobbies, and a summary display. A `java.util.Map` is used to store the information submitted by the forms. To keep it simple, no Struts JSP tags are used, but could be by wrapping model with an `ActionForm`. There are three parts to the example: the flow code which uses the Wizard object, the Struts config, and the JSP's that display the output.

2. Flow Code

Here is what the flow code looks like:

```
importPackage(Packages.java.util);
context.load("/WEB-INF/wizard.js");

function main() {
  var model = new HashMap();

  var wizard = new Wizard(model);

  // plug in custom population method
  wizard.populate = populate;

  // plug in custom validation method
  wizard.validate = validate;

  wizard.showForm("name-form", {
                  "title" : "User Name Information"
                  });
  wizard.showForm("hobbies-form", {
                  "title" : "User Hobbies"
                  });
  wizard.showForm("summary-form", {
```

```
                  "title" : "User Summary"
                  });
}

function populate() {
  m = context.chainContext.paramValues;
  for (i = m.keySet().iterator(); i.hasNext(); ) {
    key = i.next();
    this.model.put(key, m.get(key)[0]);
  }
  // Bug in commons-chain prevents this
  //this.model.putAll(context.chainContext.getParamValues());
}

function validate() {
  if (this.model.get("name").length() < 2) {
    return "Name must be specified";
  }
}
```

Notice the logic for the wizard itself is really simple. The validation and population methods can either be manually done as show here, or use frameworks like commons-validator and commons-beanutils. Notice also there is no handing of navigation as that is all taken care of by the Wizard object.

3. Struts Configuration

To configure this application with Struts, the following action mapping and plug-in are defined in `struts-config.xml`:

```
  <action-mappings>

    <action    path="/registration"
               type="net.sf.struts.flow.FlowAction"
          className="net.sf.struts.flow.FlowMapping">

      <set-property property="function" value="main" />

      <forward name="name-form"              path="/name-form.jsp"/>
      <forward name="hobbies-form"              path="/hobbies-form.jsp"/>
      <forward name="summary-form"              path="/summary-form.jsp"/>
    </action>
 </action-mappings>


<action-mappings>

<plug-in className="net.sf.struts.flow.FlowPlugIn">
    <set-property property="scripts" value="/WEB-INF/wizard-flow.js" />
</plug-in>
```

The `function` property of the custom action mapping tells `FlowAction` which JavaScript function to call. Each form in the wizard has its own forward.

4. JSP Presentation

This is the first form in the wizard:

```
<html>
<head>
  <title><%=request.getAttribute("title")%></title>
</head>
<body>

  <h1><%=request.getAttribute("title")%></h1>
  <p>
  Enter your name information:
  </p>

  <center style="color:red"><%=(request.getAttribute("errors") != null ? request.getAtt
  <form action="registration.do" method="POST">

  <% java.util.Map form = (java.util.Map)request.getAttribute("form"); %>
  <table>
   <tr>
      <th>First Name</th>
      <td><input type="text" name="name" value="<%=(form.get("name") != null ? form.get
    </tr>

   <tr>
      <th>Last Name</th>
      <td><input type="text" name="lastname" value="<%=(form.get("lastname") != null ?
    </tr>

   <tr>
      <th>Middle Name</th>
      <td><input type="text" name="middlename" value="<%=(form.get("middlename") != nul
    </tr>
  </table>

  <input type="hidden" name="contid" value='<%= request.getAttribute("contid") %>' />
  <input type="submit" name="next" value="Next" />
  </form>

</body>
</html>
```

The hidden input variable `contid` stores the continuation to load from when the form gets submitted. Since no Struts JSP tags are used, scriptlets are necessary to retrieve and display data stored in the request.