WoodyHelloWorld

"Hello World" Simple Woody Sample

- TARGET-AUDIENCE: ***beginner*** advanced expert
- COCOON-RELEASES: 2.1.3, 2.1.4
- DOCUMENT-STATUS: ***draft*** reviewed released

What you will get from this page

An example of the basics required to work with Woody forms.

Your basic skills

- You have basic knowledge about XML
- You have setup Cocoon (and maybe looked at samples)
- You know how to setup a new Cocoon application - see BeginnerSimpleWebappOrganisation
- You have worked through and understood the FlowHelloWorld sample

Technical prerequisites

- You need a cleanly installed version of Cocoon. Please refer to BeginnerInstallation for further information.

Links to other information sources

- Cocoon Forms Documentation

Setting Up

You should create a new subdirectory under the Cocoon installation directory called, for example `HelloFlow`.

Please save the attached `hellowoody.zip` file on your Cocoon machine, then unzip all the files. If you expand the ZIP file properly, this should contain the following subdirectories:

- `documents`
- `resources`
- `forms`

The Sitemap

The following code snippets highlight relevant snippets of the sitemap that are required for forms.

```
<map:transformers default="xslt">

  <!-- woody -->
  <map:transformer
    name="woody"  logger="woody"
    src="org.apache.cocoon.woody.transformation.WoodyTemplateTransformer"/>

  <!-- internationalisation  -->
  <map:transformer
      name="i18n"
      src="org.apache.cocoon.transformation.I18nTransformer">
    <catalogues default="woody">
      <catalogue id="woody" name="WoodyMessages" location="messages"/>
      <catalogue id="forms" name="FormLabels"    location="messages"/>
    </catalogues>
    <cache-at-startup>true</cache-at-startup>
  </map:transformer>

</map:transformers>
```

The above is required to enable the Woody transformer and allow for translations of Woody error messages.

```
<map:selectors default="browser">

  <!-- flow exceptions -->
  <map:selector
     name="exception"
     src="org.apache.cocoon.selection.XPathExceptionSelector">
    <exception name="invalid-continuation"
               class="org.apache.cocoon.components.flow.InvalidContinuationException"/>
    <exception class="java.lang.Throwable" unroll="true"/>
  </map:selector>

</map:selectors>
```

The above is required to handle exceptions from the flow.

```
  <!-- woody -->
  <map:transformer
    name="woody"
    src="org.apache.cocoon.woody.transformation.WoodyTemplateTransformer"
    logger="woody"/>
```

The above is required to use the Woody Template Transformer.

```
  <!-- internationalisation  -->
  <map:transformer
     name="i18n"
     src="org.apache.cocoon.transformation.I18nTransformer">
    <catalogues default="woody">
      <catalogue id="woody" name="WoodyMessages" location="messages"/>
    </catalogues>
    <cache-at-startup>true</cache-at-startup>
  </map:transformer>
```

The above, while not strictly required, is very useful if you need to create an alternatve language representation of form elements. It also allows for the conversion of Woody "error codes" into user-friendly messages; see the `/resources/WoodyMessages.xml`.

```
  <!-- display the form for entry -->
  <map:match pattern="form-display">
    <map:generate src="forms/form_template.xml"/>
    <map:transform type="woody">
    <map:transform type="i18n"><!-- translates form messages and codes -->
      <map:parameter name="locale" value="{request-param:locale}" />
    </map:transform>
    <map:transform src="resources/woody-samples-styling.xsl"/>
    <map:serialize/>
  </map:match>

  <!-- process the info from the form -->
  <map:match pattern="form-success">
    <map:generate type="jxt" src="documents/hellowoody.jxt"/>
    <map:serialize/>
  </map:match>
```

The above matches handle the form display, based on the form definition file; note the `<map:transform type="woody">` step which handles the actual processing - thereafter any internationalisation takes places, followed by use of a tranform to create the actual HTML page.

```
  <!-- resources -->
  <map:match pattern="resources/**">
    <map:read src="{0}"/>
  </map:match>
```

The above ensures the availability of files (used for styling).

The Woody Form Definition

The form definition file (`form_definition.xml`) spells out what form "elements" are required in the form. These are called "widgets". The information associated with widgets is used to both help display (eg. 'label') and validate (eg. 'length') the form elements, but is **not** used for layout.

```
<?xml version="1.0"?>
<wd:form
  xmlns:wd="http://apache.org/cocoon/woody/definition/1.0"
  xmlns:i18n="http://apache.org/cocoon/i18n/2.1">

  <wd:widgets>

    <wd:field id="name" required="true">
      <wd:label>Name</wd:label>
      <wd:datatype base="string">
        <wd:validation>
          <wd:length min="5"/>
        </wd:validation>
      </wd:datatype>
    </wd:field>

  </wd:widgets>

</wd:form>
```

The Woody Form Template

The template file (`form_template.xml`) is used for layout of the form elements (defined in the Form Defintion). It uses the *wt:* prefix for the Woody elements that must appear on the form page.

```
<?xml version="1.0"?>
<html
  xmlns:wt="http://apache.org/cocoon/woody/template/1.0"
  xmlns:wi="http://apache.org/cocoon/woody/instance/1.0">
<head>
  <title>Hello Woody Form</title>
</head>
<body>

  <h1>Enter Your Name</h1>
  <wt:form-template action="#{$continuation/id}.continue" method="POST">

    <wt:widget-label id="name"/>
    <wt:widget id="name"/>
    <br/>

    <input type="submit"/>

  </wt:form-template>

</body>
</html>
```

The Flow Script

The `hellowoody.js` in the `documents` directory looks as follows; explanantions for each step are given in the script comments:

```
cocoon.load("resource://org/apache/cocoon/woody/flow/javascript/woody2.js");

function hellowoody() {
  /*
  create Form object from definition - Form function is defined in the woody2.js
  available from the Cocoon source code repository (note cut in lines...):
  http://cvs.apache.org/viewcvs.cgi/cocoon-2.1/src/blocks/woody/
java/org/apache/cocoon/woody/flow/javascript/woody2.js?rev=1.13&view=auto
  */
  var form = new Form("forms/form_definition.xml");

  /* getModel() function is also defined in woody2.js */
  var model = form.getModel();

  /* set the default value for the "name" widget */
  model.name = "Woody";

  /* call a pipeline in the sitemap */
  form.showForm("form-display");

  /* populate javascript variable with data from the Form object*/
  var bizdata = { "name" : model.name };

  /* display the HelloWorld.jxt page; using the form-success pipeline */
  cocoon.sendPage("form-success", bizdata );

}
```

The Result Page

The `hellowoody.jxt` in the `flow` directory is used to display the `name` value submitted from the form.

```
<?xml version="1.0"?>
<html xmlns:jx="http://apache.org/cocoon/templates/jx/1.0">
<head>
  <title>Cocoon Woody Hello World</title>
</head>
<body>
  <h1>Hello ${name}!</h1>
</body>
</html>
```

Note that this is the same as the `hello.jxt` for the FlowHelloWorld sampl

What did we achieve?

- Introduced the necessary components into the sitemap to allow for Woody forms handling; these have been clearly indicated and can be added to existing or new applications that require forms handling.
- Showed that form **definition** and form **layout** are kept separate
- Showed how basic form **validation** is performed by Cocoon, once correctly specified
- Demonstrated how flow and Woody can work together

---

page metadata

- AUTHOR: DerekH
- REVIEWED-BY: (none yet)
- REVIEWER-CONTACT:BR

Attachment: hellowoody.zip