DataDebuggingTechnique

Overview

Often, in order to identify a problem in the transformation pipeline, it is useful to see the XML data that flows though the system at each stage in the pipeline. At the development stage, it is easy to amend the sitemap to allow one to do just that by embedding conditional serializers between stages in the pipeline.

Example

In this example, an action optionally fires, then an xsp page generates some data, then an XSL stylesheet generates a presentation-independent user interface, which is then transformed by yet another XSL stylesheet to HTML. It could be a good practice pipeline, shortly expressed like that :

```
<map:pipeline>
  <map:match pattern="show-*">
    <!-- Fetch the data -->
    <map:generate type="serverpages" src="{1}.xsp"/>
    <!-- Generate the UI -->
    <map:transform src="{1}-ui.xsl"/>
    <!-- Prepare for presentation in an HTML browser -->
    <map:transform src="ui2html.xsl"/>
    <map:serialize type="html"/>
   </map:match>
</map:pipeline>
```

More steps in the pipeline

It is convenient to be able to intercept the data between all the stages in this process, as well as to intercept the browser's request even before the action fires.

```

<map:pipeline>

  <map:match pattern="show-*">

    <!--If requested to serialize as request, don't even fetch the data -->
    <map:match pattern="as_req" type="request-parameter">
      <map:generate type="request"/>
      <map:serialize mime-type="text/xml" type="xml"/>
    </map:match>

    <!-- Fetch the data -->
    <map:generate type="serverpages" src="{1}.xsp"/>

    <!-- If requested to serialize as data, don't generate the UI -->
    <map:match pattern="as_data" type="request-parameter">
      <map:serialize mime-type="text/xml" type="xml"/>
    </map:match>

    <!-- Generate the UI -->
    <map:transform src="{1}-ui.xsl"/>

    <!-- If requested to serialize as UI XML, don't convert to HTML -->
    <map:match pattern="as_ui" type="request-parameter">
      <map:serialize mime-type="text/xml" type="xml"/>
    </map:match>

    <!-- Prepare for presentation in an HTML browser -->
    <map:transform src="ui2html.xsl"/>

    <map:serialize type="html"/>
   </map:match>
</map:pipeline>
```

Now, if you want to see the XML at any point in the pipeline, you simply add ?as_whatever=1 to the URL and the browser will show you the data at any point in the process:

```
http://mysite/show-asset?as_req=1
http://mysite/show-asset?as_data=1
http://mysite/show-asset?as_ui=1
```

I found that I used this technique very efficient for debugging data transformations.

– IlyaAKriveshko

Using Views

A possibility is the use of View.

– ReinhardPoetz

See DebuggingWithViews for an alternate approach that does not modify pipeline logic

– ILinKuo

XSL

Usage from xsl debugging for html (Cocoon1, off line), add theese lines :

```

<xsl:template name="debug" match="node()" mode="debug">
  <xsl:param name="node" select="."/>
  <textarea rows="5" cols="80" style="width:100%">
    <xsl:copy-of select="$node"/>
  </textarea>
</xsl:template>
...
<xsl:apply-templates select="/" mode="debug"/>
```

For a nicer html view, Cocoon provide a simple-xml2html.xsl in different places (samples/common/style/xsl/html), but it's clearly the Internet Explorer look and feel, with this big problem : lots of "+" and "-", making impossible to fastly copy paste some xml snippets. You can try the one attached, could be useful in different places. You can use it for "ViewSource" pipelines (like upper), but also to show code in your pages.

– FredericGlorieux

Attachment: xml.xsl