PortalEngine

On this page I hope to build up an FAQ for developing web sites in the new Portal Engine.

This is the one included with the stock Cocoon distribution at the url http://localhost:8888/samples/blocks/portal/.

As this page grows I will move some sections to their own pages.

- PortalEngineBookmarks
- WoodyCopletForPortalEngine
- PortalPageLabels

Q: I notice there are two portal blocks, portal (Portal Engine) and portal-fw (Portal Framework), what's the difference?

A: First of all, the portal-fw block is considered obsolete and is replaced by the portal block. If you plan on using the portal-fw block, consider the following:

They both use the authentication framework (authentication-fw block) and therefore provide the same functionality. However, the (newer) portal block can be used with a different authentication mechanism if required while the portal-fw block is tied to the authentication framework.

Q: In `row.xsl` and `column.xsl`, included with the portal framework, attributes `bgcolor`, `width`, `border` etc. are checked for and built into the resulting table if specified. How do I specify them?

A: In your layout xml file, like this example:

```
<composite-layout name="column">

  <!-- LEFT COLUMN -->
  <item>

    <!-- **** Set width to 70% **** -->
    <parameter name="width" value="70%"/>

    <!-- **** Use a red background **** -->
    <parameter name="bgcolor" value="red"/>


    <coplet-layout name="coplet">
      <coplet-instance-data>Coplet-1</coplet-instance-data>
    </coplet-layout>
  </item>
  <!-- RIGHT COLUMN -->
  <item>
    <!-- [...] -->
  </item>
</composite-layout>
```

Q: That's fine, but what I really wanted to do was set a parameter in my layout xml file which can be picked up in the xsl which renders the *box* around my coplet. How do I do it?

A: The way you do it is very similar. In your layout file, pass a `<parameter/>` like this:

```
    <coplet-layout name="coplet" layout-renderer-name="hangingbox">
      <parameter name="class" value="important"/>
      <coplet-instance-data>LoginDetails-1</coplet-instance-data>
    </coplet-layout>
```

You can then pick up this value in your window renderer xsl, like this:

```
<xsl:template match="window">

  <xsl:variable name="class">
    <xsl:choose>
      <xsl:when test="normalize-space(@class)">
        <xsl:text>window </xsl:text>
        <xsl:value-of select="normalize-space(@class)"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:text>window</xsl:text>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:variable>

  <div>
    <xsl:attribute name="class">
      <xsl:value-of select="$class"></xsl:value-of>
    </xsl:attribute>

    <!-- [SNIP] -->

  </div>
</xsl:template>
```

In this case, coplet windows by default get created as `<div class="window"></div>`, but if you set for example `<parameter name="class" value="important"/>` then your window will be rendered as `<div class="window important"></div>`. This can be picked up in your site's CSS file like this:

```
  .window {
    border: 1px solid black;
  }

  .window.important {
    border: 4px solid red;
    background: yellow;
    /* Ugh, good job this is just an example... :-) */
  }
```

Q: One of my coplets does a database lookup, then chunks the resultset through the filter transformer to display it in pages of 5 items. When I'm in full screen mode, I'd like it to display 50 at a time instead of 5. How do I do that?

A: Use the parameter selector to test the value of `aspectDatas/fullscreen` in the sitemap for your coplet. For example:

```
  <map:select type="parameter">
    <map:parameter name="parameter-selector-test"
     value="{coplet:aspectDatas/fullScreen}"/>
    <map:when test="false">
      <map:transform type="filter">
        <map:parameter name="element-name" value="row"/>
        <map:parameter name="count" value="5"/>
        <map:parameter name="blocknr" value="{coplet:attributes/page}"/>
      </map:transform>
    </map:when>
    <map:otherwise>
      <map:transform type="filter">
        <map:parameter name="element-name" value="row"/>
        <map:parameter name="count" value="50"/>
        <map:parameter name="blocknr" value="{coplet:attributes/page}"/>
      </map:transform>
    </map:otherwise>
  </map:select>
```

Note also from the above example that the coplet attribute `page` is being used to indicate the current page. By using a coplet attribute, different coplets can cut and paste the same code, but they all remember their own value of `page`. When your coplet generates the links for each page, it should use the cl:links tags, something like this:

```
<xsl:for-each select="sql:block">
  <cl:link class="activepage"
           path="attributes/page"
           value="{@sql:id}">
    <xsl:value-of select="@sql:id" />
  </cl:link>
</xsl:for-each>
```

- Brief diversion to discuss `cl:links`*

Q: Can you give me some more examples of cl:links?

A: OK. The purpose of the link is to pass some data to a coplet or layout. A single `cl:link` will pass one item of data, or you can combine more than one action by using a `cl:links` tag.

This will set the value of the "page" attribute of the current coplet to 1:

```
<cl:link class="activepage"
         path="attributes/page"
         value="{@sql:id}">

  Go to page 1

</cl:link>
```

As you can see, the content of the `cl:link` tag becomes the clickable part of the anchor tag, the above will get transformed into something like `<a href="..">Go to page 1</a>`.

To set more than one value, use `cl:links`, with an additional `cl:content` tag to set the clickable part of the anchor:

```
<cl:links>
  <!-- set an attribute on this coplet -->
  <cl:link path="attributes/data"
           value="123"/>

  <!-- set an attribute on some other coplet -->
  <cl:link coplet="AnotherCoplet-1"
           path="attributes/data"
           value="456"/>

  <!-- select the first tab on the main layout -->
  <!-- N.B. uses a 0-based index -->
  <cl:link layout="main"
           path="aspectDatas/tab"
           value="0"/>

  <!-- the bit that the user clicks -->
  <cl:content>Click me!</cl:content>
</cl:links>
```

- OK, back to the main discussion.*

Q: That works great, except when I switch to page 3 when it is rendered small, then switch to full screen view, there are no items to display because the page is still set to 3, but when rendering 50 to a page there is only one page.

A: Tricky one. I'm still working on a solution to this one. When you first move to full screen mode, it probably needs to reset `page` back to 1, but I don't know how to do that.

Q: My portal has an outer tab container consisting of a number of tabs. Within each tab there is another tab container, which actually gets rendered as a left nav bar. When the user chooses an item on the left nav bar, then navigates to a different tab, then navigates back again, the item they selected on the left nav bar is still active, but I wanted it to switch back to the top item.

A: In cocoon.xconf, find the entry for the left navbar tab renderer. It has a value for "store" which is set to "session". Simply change it to "temporary". Now, whenever the page containing that tab container is rendered, it will switch back to the first tab. Be warned though, this might not be quite what you want. If you have a coplet on tab 2 of your (now temporary) layout, and you click a link in it to adjust some parameter in it, it will still revert to the first tab. So when you render the link you will have to also explicitly switch to the tab which you are already on (if you see what I mean).

```
<renderer class="org.apache.cocoon.portal.layout.renderer.impl.AspectRenderer"
 logger="portal" name="leftnav">
  <aspects>
    <aspect type="xslt">
      <parameter name="style" value="{global:skin}styles/leftnav.xsl"/>
    </aspect>
    <aspect type="parameter">
      <parameter name="tag-name" value="leftnav-layout"/>
    </aspect>
    <aspect type="history"/>
    <aspect type="tab-content">
       <parameter name="root-tag" value="false"/>

       <!-- Don't remember the selected tab
            between invocations -->
       <parameter name="store" value="temporary"/>


    </aspect>
  </aspects>
</renderer>
```

Q: How do I render a particular coplet without a minimize button?

A: In your copletdata xml file, you need to set the "sizable" attribute on your coplet to false. Like this:

```
 <!-- Line wrapping adjusted to make the Wiki
      slightly easier to read -->
 <coplet-data id="LoginDetails" name="standard">
    <title>box.logindetails.heading</title>
    <coplet-base-data>URICoplet</coplet-base-data>
    <attribute>
            <name>uri</name>
            <value
 xsi:type="java:java.lang.String"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
>cocoon:/coplets/logindetails/coplet</value>
    </attribute>


    <aspect>
      <name>sizable</name>
      <value
 xsi:type="java:java.lang.Boolean"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
>false</value>
    </aspect>


 </coplet-data>
```

You will another little tip in the above example, my window renderer stylesheet renders the title element inside `<i18n:text></i18n:text>` tags, so my coplet title bars are also internationalized.

Q: How does my coplet know if it is being rendered in full screen (maximized) mode?

A: The coplet attribute "aspectDatas/fullScreen" tells you. In the sitemap for the particular coplet, do something like this:

```
  <map:transform src="generatelist.xsl">
    <map:parameter name="fullscreen"
      value="{coplet:aspectDatas/fullScreen}"/>
  </map:transform>
```

Then, at the top of your xsl file, you can declare a parameter to receive the information:

```
<xsl:param name="fullscreen"/>
```

The parameter will be set to `true` or `false` depending on the rendering mode. You can use this parameter in an `xsl:if` or `xsl:choice` block like this:

```
<xsl:if test="$fullscreen='true'">
  <!-- Do some extra thing if we are in fullscreen mode -->
</xsl:if>
```

or

```
<xsl:choice>
  <xsl:when test="$fullscreen='true'">
    <!-- fullscreen stuff -->
  </xsl:when>
  <xsl:otherwise>
    <!-- normal window stuff -->
  </xsl:otherwise>
</xsl:choice>
```

Q: How do I flag a coplet as capable of being rendered in "edit" mode?

A: Don't know yet...

Q: How do I create a link which switches a coplet into or out of edit mode?

A: Don't know yet...

Q: How does my coplet know if it is being rendered in edit mode?

A: Don't know yet...

Q: How can I use woody and flow inside a coplet without jumping out of the coplet when clicking a button in my woody form ?

A: There are several ways. An easy one is described in [this document|http://cocoon.apache.org/2.1/developing/portal/forms.html}

JonEvans

Q: How do I customize my coplet window?

A: You can customize coplet window rendering by creating a new renderer or configuring an existing one in cocoon.xconf

For example, if you want a round window border instead of the standard, add a new renderer:

```
<renderer class="org.apache.cocoon.portal.layout.renderer.impl.AspectRenderer"
 logger="portal" name="round-window">
 <aspects>
   <aspect type="xslt">
     <parameter name="style" value="{global:skin}styles/round-window.xsl"/>
   </aspect>
   <aspect type="parameter">
     <parameter name="tag-name" value="window"/>
   </aspect>
   <aspect type="window">
     <parameter name="root-tag" value="false"/>
   </aspect>
   <aspect type="coplet-full-screen"/>
   <aspect type="coplet-sizing"/>
   <aspect type="history"/>
   <aspect type="coplet-cinclude"/>
 </aspects>
</renderer>
```

then add it in the coplet definition:

```
<layout class="org.apache.cocoon.portal.layout.impl.CopletLayout" name="coplet">
  <renderers default="window">
    <renderer name="window"/>
    <renderer name="nowindow"/>
    <renderer name="round-window"/>
 </renderers>
</layout>
```

and finally, switch the coplet renderer in your layout/portal.xml with the famous:

```
<coplet-layout name="coplet" layout-renderer-name="round-window">
```

AlexRomayev compliments of LaurentTrillaud