

CocoonAppAsJSR168Portlet

Content

This document describes how to run a existing cocoon application as a JSR168 portlet. The following example illustrates how to deploy a subapplication as a JSR168 portlet in Pluto. It should work with any JSR 168 portlet Container.

Tested on

- jdk 1.4.2_04-b05
- cocoon 2.1.6
- pluto-src-1.0.1-rc2
- Tomcat 4.1.31

Deployment

1. copied blocks.properties to local.blocks.properties and disabled the scratchpad and cron (this one caused a problem on my configuration, but you might leave that step out) blocks
2. created a portlet.xml in <COC00N_HOME>/src/webapp/WEB-INF by using <COC00N_HOME>/src/blocks/portal/WEB-INF/portlet.xml as a base and removed the first portlet node so that you only keep the second portlet node (portlet-name = CocoonPortlet). You're might be interested in changing the init-param servlet-path e.g. to the value /yourCocoonSubapp to see the output you normally get when opening <http://localhost:8080/cocoon/yourCocoonSubapp> if you have a subapp in that directory.
3. modify <COC00N_HOME>/src/webapp/WEB-INF/cocoon.xconf to include the following input module

```
<component-instance
  class="org.apache.cocoon.components.modules.input.PortletURLModule"
  logger="core.modules.mapper" name="portlet"/>
```

it will be used by the [LinkRewriterTransformer] described [#linkproblem below].

4. built cocoon war (build war) from source
5. deleted content of <TOMCAT_HOME>/common/endorsed
6. copied <COC00N_HOME>/lib/endorsed/x*.jar to <TOMCAT_HOME>/common/endorsed
7. built pluto-src-1.0.1-rc2 (maven fullDeployment)
8. ran maven deploy -Ddeploy=<COC00N_HOME>/build/cocoon-2.1.6/cocoon.war
9. add <load-on-startup>1</load-on-startup> to the servlets with servlet-name Cocoon and Xindice in <TOMCAT_HOME>/webapps/cocoon/WEB-INF/web.xml
10. deleted <TOMCAT_HOME>/webapps/cocoon/WEB-INF/lib/pluto-1.0.1-rc1.jar (makes trouble)
11. you can savely delete portlet-api-1.0.jar (optional)
12. modified <TOMCAT_HOME>/webapps/pluto/WEB-INF/data/portletcontexts.txt

```
/testsuite
/cocoon
```

13. modified <TOMCAT_HOME>/webapps/pluto/WEB-INF/data/portletentityregistry.xml to include the following inside the <portlet-entity-registry> node:

```
<application id="9">
  <definition-id>cocoon</definition-id>
  <portlet id="1">
    <definition-id>cocoon.CocoonPortlet</definition-id>
  </portlet>
</application>
```

14. modified <TOMCAT_HOME>/webapps/pluto/WEB-INF/data/pageregistry.xml to include the following inside of the <portal> node:

```

<fragment name="cocoon" type="page">
  <navigation>
    <title>Cocoon </title>
    <description>Cocoon distribution...</description>
  </navigation>

  <fragment name="row6" type="row">
    <fragment name="col6" type="column">
      <fragment name="p7" type="portlet">
        <property name="portlet" value="9.1"/>
      </fragment>
    </fragment>
  </fragment>
</fragment>

```

Your cocoon app should now be available as a portlet under <http://localhost:8080/pluto/portal/cocoon>

Anyway, clicking on a link in that portlet won't work. You have to modify the hrefs of you links to use a scheme that is supported by the above defined portlet input module. You then have to transform your links using the [LinkRewriterTransformer](#). The required scheme is described in the [api doc of the module](#). So instead of using `<a href="link.html">link</a>` you have to write `<a href="portlet:render:link.html">link</a>`. You can add the Transformer to your sitemap like this:

```

<map:components>
  <map:transformer name="linkrewriter"
    src="org.apache.cocoon.transformation.LinkRewriterTransformer">
    <link-attrs>href src</link-attrs>
    <schemes>portlet</schemes>
    <input-module name="portlet"/>
  </map:transformer>
</map:transformers>
</map:components>

```

Then modify your pipeline as follows to actually use the linkrewriter:

```

<map:match pattern="">
  <map:read src="cocoon:/index.html"/>
</map:match>

<map:match pattern="*.html">
  <map:generate src="content/{1}.xml"/>
  <map:transform type="linkrewriter"/>
  <map:transform src="style/xsl/page2html.xsl"/>
  <map:serialize type="html"/>
</map:match>

```

I've not tried to use CForms but I think [VadimGritsenko](#) has a solution for this in [a blog entry](#).