TemplateBasedWebSite

The project

⚠️ This approach is hosted as project at cocoondev.org - Othello project (http://cocoondev.org/main/117/160.html)

Content

For every web site we need content. We have content that appear in several or all pages (blocks, menus ...) and the main content for each page.

To keep this example simple we suppose that all our content is in xhtml

Blocks

All our blocks are in blocks.xml file

```
<osm:blocks xmlns:osm="http://osmosis.gr/osml/1.0">
 <osm:block blockID="menu">
  <table width="180">
   <tr>
    <td width="10">-</td>
    <td>
     <a href="Page1">Page1</a>
    </td>
   </tr>
   <tr>
    <td width="10">-</td>
    <td>
     <a href="Page1">Page1</a>
    </td>
   </tr>
  </table>
 </osm:block>
 <osm:block blockID="footer">
  <p>2003 - osmosis.gr</p>
 </osm:block>
</osm:blocks>
```

Main Content

For each request (URI) a content xml is generated

Page1.xml

```
<html xmlns:osm="http://osmosis.gr/osml/1.0">
<osm:containers>
 <osm:container contentID="page-title">
  <p> First Page Title </p>
 </osm:container>

 <osm:container contentID="page-body">
  <p>first page body ......</p>
 </osm:container>
</osm:containers>
</html>
```

Template

XHTML file that can be edit using any visual XHTML editor work as our website template. Inside this template file are some special elements belong to diffirent namespace. Those elements define where to put the content from block or main content.

template.xhtml

```
<?xml version="1.0"?>
<layout>
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:osm="http://osmosis.gr/osml/1.0">
 <body>
  <p><osm:copy-content select="page-title"/></p>
  <table>
   <tr>
    <td><osm:copy-block select="menu"/></td>
    <td><osm:copy-content select="page-body"/></td>
   </tr>
  </table>
  <p><osm:copy-block select="footer"/></p>
  </body>
 </html>
</layout>
```

Put all together

We have to aggregate all these xml content

- blocks.xml
- URI request depenting xml content
- template.xhtml

```
<map:match pattern="*.xml">
  <map:aggregate element="site">
  <map:part src="blocks.xml"/>
  <map:part src="{1}.xml"/>
  <map:part src="template.xhtml"/>
 </map:aggregate>
...
...
</map:match>
```

First transformation

The role of the first transformation is to re-arrange the elements into this .xml aggregation and put the block's and main content where in template <osm:copy-block/> and <osm:copy-content/> elements are.

```
<map:match pattern="*.xml">
 <map:aggregate element="site">
  <map:part src="blocks.xml"/>
  <map:part src="{1}.xml"/>
  <map:part src="template.xhtml"/>
 </map:aggregate>
 <map:transform src="layout.xsl"/>
...
</map:match>
```

layout.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:osm="http://osmosis.gr/osml
/1.0">
 <xsl:output method="xml"/>

 <xsl:template match="/site">
  <osm:site>
   <xsl:apply-templates select="layout"/>
  </osm:site>
 </xsl:template>

 <xsl:template match="layout">
  <xsl:apply-templates/>
 </xsl:template>


 <xsl:template match="osm:copy-content">
  <xsl:call-template name="getContent">
   <xsl:with-param name="select" select="@select"/>
  </xsl:call-template>
 </xsl:template>

 <xsl:template match="osm:copy-block">
  <xsl:call-template name="getBlock">
   <xsl:with-param name="select" select="@select"/>
  </xsl:call-template>
 </xsl:template>

 <xsl:template name="getContent">
  <xsl:param name="select"/>
  <xsl:apply-templates select="//osm:container[@contentID=$select]"/>
 </xsl:template>

 <xsl:template name="getBlock">
  <xsl:param name="select"/>
  <xsl:apply-templates select="//osm:blocks/osm:block[@blockID=$select]"/>
 </xsl:template>

 <xsl:template match="node()|@*" priority="-1">
  <xsl:copy>
   <xsl:apply-templates select="@*"/>
   <xsl:apply-templates/>
  </xsl:copy>
 </xsl:template>
</xsl:stylesheet>
```

Final transformation

Now we have to create the final .html output

```
<map:match pattern="*.xml">
  <map:aggregate element="site">
  <map:part src="blocks.xml"/>
  <map:part src="{1}.xml"/>
  <map:part src="template.xhtml"/>
 </map:aggregate>
 <map:transform src="layout.xsl"/>
 <map:transform src="core.xsl"/>
 <map:serialize type="html"/>
</map:match>
```

core.xsl

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:osm="http://osmosis.gr/osml
/1.0">
<xsl:output method="html" version="1.0" encoding="iso-8859-7" indent="yes" omit-xml-declaration="no"/>

<!-- if you create your custom xml - xsl element add here refernce to your xsl -->
<!--
<xsl:include href="custom.xsl"/>
-->

<xsl:template match="node()|@*" priority="-1">
 <xsl:copy>
  <xsl:apply-templates select="@*"/>
  <xsl:apply-templates/>
 </xsl:copy>
</xsl:template>

</xsl:stylesheet>
```

custom.xsl

here we can create our elements with custom transformation instructions

---

Othello

You can check for example our project (Othello) based on this approach. This idea is how we have create all osmosis's livesites you can find in cocoon 2.0.4 livesites section.

Othello Sample 05.04.2005

While i dont have time (for the moment) to update othello web site or/and to make a patch to bugzilla, (the sample lives in cocoon's scratchpad too) i will put here a *fresh* othello sample.

download

othello05042005.zip

quick setup

- download and unzip othello05042005.zip
- unzip and put [othello] folder in cocoon's sample dir
- edit othello's sitemap.xmap and change

```
...
<base-url></base-url>
...
```

to feet your cocoon installation

- ask for http://cocoon:8080/samples/othello/

themes

- edit sitemap.xmap and choose another theme (this zip is comming with 2 themes) lite is very simple

```
...
                        <theme>tekton</theme>
                        <!--
                        <theme>lite</theme>
                        -->
...
```

help

feel free to ask anything about othello in cocoon's user list

StavrosKounis