ProposalWikiMarkupIntegration

Using WikiMarkup to write content in Lenya

```
Note: This is work in progress! I am experiementing with this and I believe it makes sense, but it is not yet
ready for production and the Lenya developers will have to decide if they want to support this.
```

Note: Michael Wechner has written a Wiki publication, which can be downloaded from wyona.org

Rationale

Setting up a test environment

In order to experiment with Wiki notation for web content we need to set up a test environment. Fortunately, there has been a lot of work done already on this in the Cocoon project. Have a look at the Cocoon Samples (Block Samples, Chaperon blok). This section will explain how to integrate this into a Lenya publication.

In the publication-sitemap.xmap we first need to add the TextGenerator:

```
  <map:components>

    <!-- start inserting here -->

    <map:generators default="file">
      <map:generator logger="sitemap.generator.text" name="text" src="org.apache.cocoon.generation.
TextGenerator"/>
    </map:generators>

   <map:transformers default="xslt">

     <map:transformer name="lexer"
                   src="org.apache.cocoon.transformation.LexicalTransformer"
                   logger="sitemap.transformer.lexer">
       <parameter name="localizable" value="true"/>
      </map:transformer>

      <map:transformer name="parser"
                   src="org.apache.cocoon.transformation.ParserTransformer"
                   logger="sitemap.transformer.parser">
        <parameter name="flatten" value="true"/>
        <parameter name="recovery" value="true"/>
        <parameter name="localizable" value="true"/>
      </map:transformer>

      <map:transformer logger="sitemap.transformer.xsltc" name="xsltc" pool-grow="2" pool-max="32" pool-min="8"
src="org.apache.cocoon.transformation.TraxTransformer">
        <use-request-parameters>false</use-request-parameters>
        <use-session-parameters>false</use-session-parameters>
        <use-cookie-parameters>false</use-cookie-parameters>
        <xslt-processor-role>xsltc</xslt-processor-role>
        <check-includes>true</check-includes>
      </map:transformer>

    </map:transformers>

    <!-- end inserting here -->
```

```
Note: Cocoon uses a different XSLT Transformer than Lenya does. Unfortunately the pipeline serving .xlex files
will only work with XSLTC. This is why we need to put the transformers section in here.
```

Next we need an internal pipeline to serve up some files that the main wikinotation2html pipeline will need later.

```
<map:pipelines>

  <!-- start inserting here -->

  <map:pipeline type="caching" internal-only="true">
   <!-- Hint: During development you could expose this pipeline, to help
     with writing your grammars. Use a text grammar myapp.grm and request
     the myapp.xlex and myapp.xgrm URLs and then view source.
     -->
   <map:match pattern="*.xlex">
    <map:generate  type="text"   src="chaperon/grammars/{1}.grm"/>
    <map:transform type="lexer"  src="chaperon/grammars/grm.xlex"/>
    <map:transform type="parser" src="chaperon/grammars/grm.xgrm"/>
    <map:transform               src="chaperon/stylesheets/text4regex.xsl"/>
    <map:transform type="lexer"  src="chaperon/grammars/regex.xlex"/>
    <map:transform type="parser" src="chaperon/grammars/regex.xgrm"/>
    <map:transform type="xsltc"             src="chaperon/stylesheets/grm2xlex.xsl"/>
    <map:serialize type="xml"/>
   </map:match>

   <map:match pattern="*.xgrm">
    <map:generate  type="text"   src="chaperon/grammars/{1}.grm"/>
    <map:transform type="lexer"  src="chaperon/grammars/grm.xlex"/>
    <map:transform type="parser" src="chaperon/grammars/grm.xgrm"/>
    <map:transform type="xsltc"             src="chaperon/stylesheets/text4regex.xsl"/>
    <map:transform type="lexer"  src="chaperon/grammars/regex.xlex"/>
    <map:transform type="parser" src="chaperon/grammars/regex.xgrm"/>
    <map:transform type="xsltc"             src="chaperon/stylesheets/grm2xgrm.xsl"/>
    <map:serialize type="xml"/>
   </map:match>

   </map:pipeline>
```

Finally we need a matcher which will match .txt files and process them.

```
    <!-- This is the main entry point into the publication. This
    pipeline uses the uriparametrizer to determine the doctype of this
    request. It then aggregates the lenya menu (for the given area) and
    the lenya body, the actual document. -->
    <map:pipeline>

      <!-- start inserting here -->

      <map:match pattern="**/*.txt">

        <map:generate  type="text"    src="content/{1}/{2}.txt"/>
        <map:transform type="lexer"   src="cocoon:/wiki.xlex"/>
        <map:transform type="parser"  src="cocoon:/wiki.xgrm"         label="ast">
          <map:parameter name="failsafe" value="true"/>
        </map:transform>
        <map:transform type="xsltc"   src="chaperon/stylesheets/error4wiki.xsl"/>
        <map:transform type="lexer"   src="chaperon/grammars/error.xlex"/>

        <map:transform type="xsltc"   src="chaperon/stylesheets/wiki2xdoc.xsl" label="xdoc"/>

        <map:transform                src="chaperon/stylesheets/xdoc2html.xsl" />

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

      <!-- end inserting here -->
```

Now unpack this tarball chaperon-in-lenya.tar.gz in your publication directory. This tarball will contain all the files that need to go to the chaperon subdirectory that should be a subdirectory of the publication directory. All the files are taken from Cocoon basically.

Now you should be able to put a .txt file with some Wiki markup in it into a folder under content/authoring, for example

content/authoring/index/wiki.txt

This file might contain something like

```
Let's write some context using Wiki notation.

!!! This will turn into a h1

You can make text __bold__ or ''italics'' and even create [Links|http://www.google.com/] without any real
problems!
```

Now point your browser to http://yourserver/lenya/yourpub/authoring/index/wiki.txt. Enjoy!

Note: You need to put the index into the URL even though you will not need it when calling up XML / HTML files.

This proofs the concept though it has some flaws still.

Where to go from here

- The .txt file is not yet integrated into the Lenya publication. To achive this, wiki markup text files would need to be made a custom document type in Lenya to fit into the rest of the system. This might be hard as .txt files are not XML. (Needs some research!)
- Wiki markup is usually edited in a HTML form editor. It should be quite easy to integrate editing the wiki markup in Lenya.
- If you don't want to edit all of your content in Wiki notation (as this will limit your webdesign choices) you want to have a look at the concept of a XML Include custom doctype. (Coming quite soon). This will allow you to aggregate HTML and and Wiki markup in one page.
- It would be possible to extent the grammar used now to support application specific tags!

```
Note: We may want to use the wiki plugin from forrest and set up a simple forrest xdocs doctype to edit the
wiki.
```

forrest-wiki-plugin

Collecting Information to make this a plugin

Files (new)

$PUBLICATIONS_HOME/$YOUR_PUB/usecase-wikiedit.xmap
$PUBLICATIONS_HOME/$YOUR_PUB/xslt/authoring/edit/wikiform.xsl

... wiki.js

Files (patched)

Files (subclassed)