HowToClosedUserGroup

`Work in progress`

Closed User Groups with Lenya

What is a Closed User Group (CUG)?

A typical example of a closed user groups (aka CUG) is a website that has a "Member's only" section. To access the content in that "Member's only" section the user needs to login using a username and password.

The authoring mode of Lenya is something quite similar: You need to login first. In fact, you can use the very same AAA framework that Lenya uses in authoring mode for Live mode as well. But as you will find out throughout this HOWTO, there are certain things to take into account and certain decisions to make.

For some background on how Lenya handles AAA at all, please see: OverviewAuthenticationAndAuthorization

Design decisions

Member yes/no versus Roles

The registration process

Implementation work in Lenya

Restricting access to live content in authoring

The login screen

The logout button

Though some website designer often overlook this, any website that requires it's users to login should provide a clean way to logout as well. Of course there are several alternatives ways to invalidate the user's session:

- If the user just surfs somewhere else, the session will timeout after some time.
- If the user will close the browser the session cookie will be lost with most browsers. The session will stay on the server (and maybe even occupy resources) until the session timeout has been reached.

None of these methods is a clean way to end a session on a website. As users get more and more concerned with security and privacy while using the web, they will want a distinct logout button with a confirmation that they really invalidated the session they have been using so nobody can steal it.

For some more information on how the underlying Cocoon handles session please see: http://cocoon.apache.org/2.1/userdocs/actions/session-action.html

Implementing a proper logout is quite straigtforward.

First of all, we need a publication specific usecase. Put the file usecase-session.xmap into your PUB_HOME directory:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  Copyright 1999-2004 The Apache Software Foundation

  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
-->

<!-- $Id: usecase-bxeng.xmap,v 1.17 2004/05/23 12:52:44 gregor Exp $ -->

<map:sitemap xmlns:map="http://apache.org/cocoon/sitemap/1.0">
```

```
        <!-- =========================== Components ================================ -->
        <map:components>
                <map:generators default="file"/>
                <map:transformers default="xalan">
      <map:transformer name="cinclude"
        src="org.apache.cocoon.transformation.CIncludeTransformer"/>
      </map:transformers>
                <map:readers default="resource"/>
                <map:serializers default="xhtml"/>
                <map:matchers default="wildcard"/>
    <map:actions>
      <map:action name="upload" logger="sitemap.action.upload" src="org.apache.lenya.cms.cocoon.acting.
UploadAction" />
    </map:actions>

                <map:selectors>
      <map:selector name="request-method" logger="sitemap.selector.request-method"
        src="org.apache.cocoon.selection.RequestMethodSelector"/>
                </map:selectors>

        </map:components>

  <!-- =========================== Resources ================================ -->
  <map:resources>

    <map:resource name="style-cms-page">
      <map:transform type="i18n">
        <map:parameter name="locale" value="{request:locale}"/>
      </map:transform>
      <map:transform src="../../xslt/util/page2xhtml.xsl">
        <map:parameter name="contextprefix" value="{request:contextPath}"/>
      </map:transform>
      <map:transform src="../../xslt/util/strip_namespaces.xsl"/>
      <map:select type="parameter">
        <map:parameter name="statusCode" value="{statusCode}"/>
        <map:when test="">
          <map:serialize type="html" />
        </map:when>
        <map:otherwise>
          <map:serialize type="html" status-code="{statusCode}"/>
        </map:otherwise>
      </map:select>
    </map:resource>

    <map:resource name="cms-screen-xml">
          <map:generate src="../../content/{serverpage}" type="serverpages"/>
          <map:transform src="../../xslt/{stylesheet}">
            <map:parameter name="use-request-parameters" value="true"/>
          </map:transform>
    </map:resource>

    <map:resource name="cms-screen">
      <map:call resource="cms-screen-xml">
        <map:parameter name="serverpage" value="{serverpage}"/>
        <map:parameter name="stylesheet" value="{stylesheet}"/>
      </map:call>
      <map:call resource="style-cms-page"/>
    </map:resource>


  </map:resources>

        <!-- =========================== Pipelines ================================ -->

        <map:pipelines>

    <map:pipeline type="noncaching">

      <map:match type="usecase" pattern="session">
      <map:match type="step" pattern="logout">
        <map:act type="session">
          <map:parameter name="action" value="terminate"/>
        </map:act>
        <map:redirect-to uri="/index.html" />
      </map:match>

      </map:match>

    </map:pipeline>

        </map:pipelines>

</map:sitemap>
```

Note the `<map:redirect-to uri="/index.html" />` part. This is necessary as every pipeline needs to end in a serializer. Alternatively you could generate any type of landing page here. In this example we just invalidate the user's session and redirect to the start page.

Now we need to call this usecase from somewhere.

A typical solution would be to have a page with a form button "Logout" on it somewhere in your application. When the user clicks on that button, the form will be posted to the usecase. This is quite clean and simple. It could look like this:

```
...
<form action="/index.html?lenya.usecase=session&amp;lenya.step=logout" method="POST">
  <input type="submit" value="Logout"/>
</form>
...
```

Lenya Developer's Dicussing:
Lenya itself does not use the SessionAction from Cocoon to invalidate the session when logging out of the authoring mode. Instead it uses an XSP (see: LENYA_HOME/lenya/content/ac/logout.xsp) which invalidates the session upon logout.

Note (AndreasHartmann): This is just legacy. In Lenya 2.0, the login/logout usecases have been refactored using the usecase framework, which means the deprecated XSP+action concept is not necessary anymore.

Optional Features

Displaying information about the current user

TODO: Implement a password reminder function