

Profiling DocBook documents with Forrest

by

Table of contents

1 What's the motivation for this?.....	2
2 Get the plugin!.....	2
3 Where to start?.....	2
4 Adding Cocoon Blocks.....	3
5 Adding Stylesheets.....	4
6 Adding Login Content.....	6
7 Making sitemap.xmap Changes.....	6

1. What's the motivation for this?

I wanted to present content on my website that was appropriate for a specific user level. Let's take a simple example use-case - a product website. This product's website is accessed by both product end users and support staff. Let's introduce an attribute 'userlevel' to represent the two types of users. End user's userlevel value will be 'guest', whilst support staff's value will be 'support'. Anyone visiting the website will automatically be considered a guest user.

The product website will start off with a guide to the product. It will have two sections - user guide and support guide. A guest user should only see the user guide section and not be aware that the support guide also exists. A support user should see both the user guide and the support guide in the same document. In order to identify yourself as a support user, you must login on the website with a username and password to an account that has userlevel set to 'support'.

Once logged in as a support user, viewing the same guide document that you saw when you first visited the website, which only contained the user guide and not the support guide as you were considered a guest, should now produce a document with both user and support guide sections. There should also be a way to send a link to a document that requires a user to login before viewing a document. This will ensure that the user sees the right version of the document straight away.

I use DocBook a lot, and wanted to apply these techniques just to DocBook documents. DocBook stylesheets already have the concept of [profiling](#) using the userlevel element attribute, so I wanted to bring these concepts over from DocBook to forrest.

2. Get the plugin!

For more information please go to the [Trac](#) Wiki page. Along with the latest updates, you will also find links to the SVN repository for the source code of the plugin.

3. Where to start?

The forrest community was great in providing tips on where to start. I was told that I could use an [authenticated pipeline](#) to achieve what I wanted. In order to do that, I had to add authentication-fw and session-fw cocoon blocks to forrest, which resulted in addition of two new jars: cocoon-session-fw-block-2.1.5 and cocoon-authentication-fw-block-2.1.5, as well a modified cocoon.xconf.

Once the cocoon blocks were added, I needed to make changes to sitemap.xmap to include new component configurations and pipelines to support the added frameworks. A new stylesheet had to be added as well, based on the docbook stylesheet, but with new sections added to exclude/include profiled sections. Finally, I had to add a login screen, user list and another two new stylesheets to process the login screen and the user list.

To satisfy the use case defined in the first section, two types of documents have been defined:

- **profiled documents**

These documents do not force a user to authenticate, and are generated with either the current userlevel, if the user has logged in or with the lowest default userlevel,

- **protected documents**

These documents do force the user to authenticate.

As an example then, let's create a document hello.xml. This document will have two sections, one for any level of user, and the second section for support users. We will use DocBook XML standard:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN"
"http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd">
<article>
  <title>Hello</title>

  <section>
    <title>User Section</title>

    <para>Hello user!</para>
  </section>

  <section userlevel="support">
    <title>Support Section</title>

    <para>Hello support!</para>
  </section>
</article>
```

In the site.xml, I want users to be able to view the document in profiled mode e.g. they do not have to authenticate. If they are not logged in, then the lowest userlevel is assumed and only the user section will be generated. If they are logged in and their userlevel is support, then they will see the document with both sections. So, in the site.xml we will refer to hello-profiled.html document. Forrest will then match the *-profiled, strip it away, process the hello.xml document and generate the html document.

If I then need to send a link to the hello file, but want to make sure that the user is logged in to view document according to their userlevel, I can send a link to hello-protected.html. Forrest will then match the *-protected, check if the user is logged in or not, if not redirect to login screen, user will then login and be redirected to the generated hello document. If the user is already logged in, then they will simply be able to view the generated document.

4. Adding Cocoon Blocks

This pipeline is not part of forrest, so it took me a while to figure out how to add the

authentication-fw and session-fw (which is a pre-requisite) blocks to forrest. To help others with the process, I wrote a separate article on [how to add cocoon blocks to forrest](#).

Once you build a new version of cocoon with the added framework, you will find that two new block jars have been built: [cocoon-session-fw-block-2.1.5](#) and [cocoon-authentication-fw-block-2.1.5](#). You need to add them to forrest's library e.g. \$FORREST_ROOT/lib/optional. The [cocoon.xconf](#) file is added to, here is the summary of new lines added:

```
<component-instance
class="org.apache.cocoon.webapps.session.components.ContextInputModule"
  logger="core.modules.input" name="session-context"/>
<builtin-logicsheet>
  <parameter name="prefix" value="xsp-session-fw"/>
  <parameter name="uri"
value="http://apache.org/xsp/session-fw/1.0"/>
  <parameter name="href"
value="resource://org/apache/cocoon/components/language/markup/xsp/java/session-fw.x"
/>
</builtin-logicsheet>
<session-manager logger="core.session-manager"/>
<session-form-manager logger="core.session-manager"/>
<session-transaction-manager logger="core.session-manager"/>
<session-context-manager logger="core.session-manager"/>
<session-media-manager logger="core.media-manager" pool-grow="4"
pool-max="32"
  pool-min="8">
  <mediatypes default="html">
    <media name="wap" useragent="Nokia"/>
    <media name="wap" useragent="UP"/>
    <media name="wap" useragent="Wapalizer"/>
  </mediatypes>
</session-media-manager>
<session-context-providers>
  <component-instance
class="org.apache.cocoon.webapps.session.context.StandardSessionContextProvider"
  name="request"/>
  <component-instance
class="org.apache.cocoon.webapps.session.context.StandardSessionContextProvider"
  name="temporary"/>
  <component-instance
class="org.apache.cocoon.webapps.authentication.context.AuthenticationContextProvide"
  name="authentication"/>
</session-context-providers>
<authentication-manager logger="core.authentication-manager">
</authentication-manager>
```

You need to either replace the forrest's \$FORREST_ROOT/src/core/context/WEB-INF/cocoon.xconf with this new version or the one generated at runtime e.g. build/webapp/WEB-INF/cocoon.xconf.

5. Adding Stylesheets

Now that the cocoon blocks have been added, a set of new stylesheet needs to be added to website/src/documentation/resources/stylesheet:

- [profile-docbook2document](#)
- [simple-page2html](#)
- [authenticate](#)

Both the simple-page2html and authenticate stylesheets are changed copies from the samples in cocoon's authentication framework blocks. The simple-page2html is used to process the login form I will introduce later, and the authenticate stylesheet is used to process the user list. I added the password check to the authentication stylesheet:

```
<xsl:template match="user">
  <!-- Compare the name of the user -->
  <xsl:if test="normalize-space(name) = $name">
    <!-- found, so create the ID if passwords match -->
    <xsl:if test="normalize-space(pass) = $pass">
      <ID>
        <xsl:value-of select="name" />
      </ID>
    </xsl:if>
  </xsl:if>
</xsl:template>
```

And to the simple-page2html, I added the following bit of javascript:

```
<script language="javascript">
function getArgs() {
  var args = new Object();
  var query = location.search.substring(1);
  var pairs = query.split(",");
  for(var i = 0; i < pairs.length; i++) {
    var pos = pairs[i].indexOf('=');
    if (pos == -1) continue;
    var argname = pairs[i].substring(0,pos);
    var value = pairs[i].substring(pos+1);
    args[argname] = unescape(value);
  }
  return args;
}
var args = getArgs();
if (args.resource) {
  document.login.redirect.value = args.resource;
} else {
  document.login.redirect.value = "protected";
};
</script>
```

This processes the passed URL and extracts the parameters. It looks for the resource URL argument and sets the redirect hidden field value in the form to that value, or defaults to protected value. All this is required because the authentication block will redirect the client to the login page and pass the resource parameter to the originating page in the URL. However it is then left up to the client to somehow use that value and redirect back to the page after the authentication has occurred.

Stylesheet profile-docbook2document is a modified version of the standard docbook2document.xsl, with these notable changes:

```

...
<!-- introduce the userlevel attribute that will passed on from the
sitemap.xmap -->
<xsl:param name="userlevel"/>
...
<!-- put an if condition around the section element processing rule to
check if userlevel
matches or is not set at all, to decide if the section is to be process
or not -->

<xsl:if test="$userlevel=@userlevel or not(@userlevel)">
  <xsl:element name="section">
    <xsl:apply-templates/>
  </xsl:element>
</xsl:if>
...

```

6. Adding Login Content

Once the stylesheets are in place, three new xml files need to be added to a location of your choice:

- [login.xml](#)
- [userlist.xml](#)
- [protected.xml](#)

All three of these files are copied from the cocoon's authentication block's samples. The login page was changed to add the new password and redirect fields. Password value is passed to the authenticate stylesheet to be matched against the userlist.xml values (which now also contains a password), and the redirect hidden value is then used to redirect the user after a successful authentication to the document that caused the authentication request.

7. Making sitemap.xmap Changes

Final piece of the puzzle is to make changes to the [sitemap.xmap](#) file:

- Define new actions, transformers, generators and selectors to make the sitemap work with the newly added cocoon blocks.
- Add component configuration section and provide an authentication handler, see authentication block docs for more details.
- Add pipelines that will match *-profiled or *-protected documents, manage the authentication process and use the new profiled stylesheet to process the document.

There are two use-cases here:

- **profiled document**

Instead of requesting a hello.xml document, a site.xml document can refer to hello-profiled.html document. In this case, the sitemap.xml will match the *-profiled pattern and will check if the user has logged in already. If the user has logged in, the userlevel value will be used by the profile stylesheet to generate an html document. If the user is not logged in, the lowest userlevel value will be used to generate the document, but the user will not be prompted to login.

- **protected document**

A user can be sent a link to hello-protected.html page, in this case if the user is logged in, then the document will be generated for the user's userlevel, if the user is not logged in, then the client will be redirected to a login page, and once authorised, redirected back to the protected document. This is useful if you sending a link to someone and want to make sure that they do login prior to viewing the document to guarantee that the content will be applicable to the user.

```
<map:sitemap xmlns:map="http://apache.org/cocoon/sitemap/1.0">
<!-- define components required by the authentication and session
framework blocks -->
    <map:components>
        <map:actions>
            <map:action name="session"
src="org.apache.cocoon.webapps.session.acting.SessionAction"/>
            <map:action name="session-form"
src="org.apache.cocoon.webapps.session.acting.SessionFormAction"/>
            <map:action name="session-form-manager"
src="org.apache.cocoon.webapps.session.acting.FormManagerAction"/>
            <map:action name="auth-protect"
src="org.apache.cocoon.webapps.authentication.acting.AuthAction"/>
            <map:action name="auth-login"
src="org.apache.cocoon.webapps.authentication.acting.LoginAction"/>
            <map:action name="auth-logout"
src="org.apache.cocoon.webapps.authentication.acting.LogoutAction"/>
            <map:action name="auth-loggedIn"
src="org.apache.cocoon.webapps.authentication.acting.LoggedInAction"/>
        </map:actions>
        <map:transformers default="xslt">
            <map:transformer name="session" pool-grow="4"
pool-max="32"
                pool-min="8"
src="org.apache.cocoon.webapps.session.transformation.SessionTransformer"/>
            <map:transformer name="session-pre"
pool-grow="4" pool-max="32"
                pool-min="8"
src="org.apache.cocoon.webapps.session.transformation.SessionPreTransformer"/>
            <map:transformer name="session-post"
pool-grow="4" pool-max="32"
                pool-min="8"
src="org.apache.cocoon.webapps.session.transformation.SessionPostTransformer"/>
            <map:transformer name="encodeURL" pool-grow="4"
pool-max="32"
                />
        </map:transformers>
    </map:components>
</map:sitemap>
```

```

        pool-min="8"
src="org.apache.cocoon.transformation.EncodeURLTransformer"/>
    </map:transformers>
    <map:generators default="file">
        <map:generator label="content" name="auth-conf"
src="org.apache.cocoon.webapps.authentication.generation.ConfigurationGenerator"/>
    </map:generators>
    <map:selectors default="browser">
        <map:selector name="session-media"
src="org.apache.cocoon.webapps.session.selection.MediaSelector"/>
    </map:selectors>
    </map:components>

<!-- define a shared resource that can be used by both protected and
profiled matches -->

    <map:resources>
        <map:resource name="match-profile">
            <map:match pattern="**/*-*.xml">
                <map:generate
src="{project:content.xdocs}{1}/{2}.xml"/>
                <map:transform type="session"/>
                <map:transform
src="{project:resources.stylesheets}/profile-docbook2document.xsl">
                    <map:parameter name="userlevel"
value="{../userlevel}"/>
                    <map:parameter name="docbase"
value="{2}"/>
                </map:transform>
                <map:transform type="encodeURL"/>
                <map:serialize type="xml"/>
            </map:match>
        </map:resource>
    </map:resources>

<!-- define the pipelines required to support profiled and protected
documents -->

    <map:pipelines>

<!-- define the authentication manager handler as required by the
authentication framework -->

        <map:component-configurations>
            <authentication-manager>
                <handlers>
                    <handler name="profilehandler">
                        <redirect-to
uri="cocoon://login"/>
                        <authentication
uri="cocoon:raw:/authenticate"/>
                    </handler>
                </handlers>
            </authentication-manager>
        </map:component-configurations>
    </map:pipeline>

<!-- handles the login process by using the login.xml page -->

```

```

        <map:match pattern="login">
            <!-- if we are already logged in,
redirect to the protected document -->
                <map:act type="auth-loggedIn">
                    <map:parameter name="handler"
value="profilehandler"/>
                    <map:redirect-to
uri="protected"/>
                </map:act>
                <map:generate
src="authentication-fw/login.xml"/>
                <map:transform
src="{project:resources.stylesheets}/simple-page2html.xsl"/>
                <map:transform type="encodeURL"/>
                <map:serialize/>
            </map:match>

<!-- called by login and if authentication is successful, redirects to
the protected document in the forms hidden field -->

        <map:match pattern="do-login">
            <!-- try to login -->
                <map:act type="auth-login">
                    <map:parameter name="handler"
value="profilehandler"/>
                    <map:parameter
name="parameter_name"
value="{request-param:username}"/>
                    <map:parameter
name="parameter_pass"
value="{request-param:password}"/>
                    <map:redirect-to
uri="{request-param:redirect}"/>
                </map:act>
            <!-- something was wrong, try it again
-->
                <map:redirect-to uri="login"/>
            </map:match>

<!-- match profiled document, these do not force an authentication -->

        <map:match pattern="**/*-profiled.xml">
            <map:act type="auth-loggedIn">
                <map:parameter name="handler"
value="profilehandler"/>
                <map:redirect-to
uri="{../1}/{../2}-protected.xml"/>
            </map:act>
            <map:call resource="match-profile">
                <map:parameter name="userlevel"
value="user"/>
            </map:call>
        </map:match>

<!-- match protected documents, these do force an authentication -->

        <map:match pattern="**/*-protected.xml">
            <map:act type="auth-protect">
                <map:parameter name="handler"

```

```

value="profilehandler"/>
                                <map:call
resource="match-profile">
                                <map:parameter
name="userlevel" value="{ID}"/>
                                </map:call>
                                </map:act>
                                <map:redirect-to uri="login"/>
                                </map:match>

<!-- default redirect page after authentication -->
                                <map:match pattern="protected">
                                <map:act type="auth-protect">
                                <map:parameter name="handler"
value="profilehandler"/>
                                <map:generate
src="authentication-fw/protected.xml"/>
                                <map:transform type="session"/>
                                <map:transform
src="{project:resources.stylesheets}/simple-page2html.xsl"/>
                                <map:transform
type="encodeURL"/>
                                <map:serialize/>
                                </map:act>
                                <!-- something was wrong, redirect to
login page -->
                                <map:redirect-to uri="login"/>
                                </map:match>

<!-- logout page -->
                                <map:match pattern="do-logout">
                                <map:act type="auth-protect">
                                <map:parameter name="handler"
value="profilehandler"/>
                                <map:act type="auth-logout"/>
                                </map:act>
                                <map:redirect-to uri="login"/>
                                </map:match>
                                </map:pipeline>

<!-- authentication pipeline, matches user's username and password from
the form against system userlist -->
                                <map:pipeline internal-only="true">
                                <!-- This is the authentication resource -->
                                <map:match pattern="authenticate">
                                <map:generate
src="authentication-fw/userlist.xml"/>
                                <map:transform
src="{project:resources.stylesheets}/authenticate.xsl">
                                <map:parameter
name="use-request-parameters" value="true"/>
                                </map:transform>
                                <map:serialize type="xml"/>
                                </map:match>
                                </map:pipeline>
                                </map:pipelines>

```

```
</map:sitemap>
```