

Adding Cocoon Blocks to Forrest

by

Table of contents

1 What's the motivation for this?.....	2
2 Where to start?.....	2
3 Something is still not right!.....	3
4 Triumphant summary!.....	4

1. What's the motivation for this?

I wanted to add profiling functionality to forrest i.e. DocBook allows you to place attributes like userlevel on various elements and the stylesheets will include and exclude them depending on the passed userlevel value. The forrest community was very helpful, and suggested that I should use cocoon's authentication framework to implement this feature. With that in mind, I searched the web for cocoon's authentication framework to only discover that it is provided in a form of a cocoon block, and it was not part of forrest.

At this point, I asked the user community for help on adding new blocks, but no one was able to provide a definitive step-by-step guide, although I did get some very useful tips. After many hours of stumbling around I finally was able to add the blocks to forrest. To help others, I decided to document my efforts on adding new blocks in this article. I also published another article on how to create [profiled documents in forrest](#). The steps here are by no mean complete, and in fact I still have many open questions. Please [contact me](#) with suggestions.

2. Where to start?

I downloaded forrest-06 binary distribution and seeded my project. As advised, I searched for any keywords in source code that related to authentication, but failed to find any. So I looked on the web, and found cocoon's docs for the [authentication framework](#). A useful document, but at this point I still did not understand how cocoon's blocks worked. I guessed that they were another form of plugins, so again I searched around on the web for notes on adding cocoon blocks to forrest, and found this [wiki](#) link. At this point I was very excited and thought that I would not need to write this article! I was wrong.

I looked at instructions in the wiki article, and

- downloaded and extracted cocoon-2.1.5 source distribution.

Note:

I am still a bit confused about cocoon's versioning in forrest. Looking at the code, forrest includes blocks from cocoon-2.1.5, however when you run a forrest website, error messages are from cocoon-2.2.0-dev. As all of the included blocks in forrest-06 were 2.1.5 according to the name of the jars, I assumed that cocoon 2.1.5 should be fine.

- copied local.build.properties and local.blocks.properties files from \$FORREST_HOME/etc/cocoon_upgrade to \$COCOON_HOME.
- modified the local.blocks.properties file by setting the required blocks to true.

2.1. building cocoon's authentication-fw block

```
#----[dependency]: "authentication-fw" depends on "session-fw".
#----[dependency]: "authentication-fw" is needed by "portal",
"portal-fw".
include.block.authentication-fw=true
```

```
...  
#----[dependency]: "session-fw" depends on "xsp".  
#----[dependency]: "session-fw" is needed by "authentication-fw",  
"portal", "portal-fw".  
include.block.session-fw=true
```

- ran the build.bat in \$COCOON_HOME.
- copied the generated jars from \$COCOON_HOME/build/cocoon-2.1.5/blocks to \$FORREST_HOME/lib/optional.

So, up to this point I have been following the steps in the wiki article. The next suggested steps are:

- run the upgrade_cocoon_jars.sh script in \$FORREST_HOME/etc/cocoon_upgrade.
- rebuild forrest by running build.bat in \$FORREST_HOME

As I was doing all of this on windows, I could not run the .sh script directly. Instead, I tried to run it via cygwin shell, but the script failed. Looking at the script, I only saw that the newly created jars will be renamed according to preset pattern into forrest's lib directory. So I followed the code to copy and rename the generated jars as per the bzcopyp code from upgrade_cocoon_jars.sh:

```
# Copy a block's compiled jar  
function bzcopyp()  
{  
  echo -n "Copying block jar: $1" "  
  push  
  echo "Updating $FLIB/cocoon-$1-block-* = `ls $FLIB/cocoon-$1-block-*`"  
  cd $FLIB  
  rm cocoon-$1-block-*.jar  
  cp $CBLOCKS/$1-block.jar cocoon-$1-block-$JARSUFFIX.jar  
  pop  
  echo "done"  
}
```

I could see no point at this time in rebuilding forrest, but just to be careful I did rebuild it. The derived binary object was identical to the pre-compiled one and so my hunch appeared to be correct.

3. Something is still not right!

With the newly compiled jars now safely in forrest's lib directory, I tried again to create an authenticated pipeline in a forrest website. This failed as the actions I was trying to use were not defined. I searched through the cocoon block and found a number of .xconf and other .x something files, which did indeed define the various actions, generators etc.

At this point it was obvious that I would need to make changes to my sitemap.xmap in the forrest website and at least some other cocoon.xconf file, which I have not used before. Maybe some other files as well. The various .x files in the block had path elements in their headers:

```
<xconf xpath="/cocoon/session-context-providers"
unless="component-instance[@name='authentication']">
  <component-instance
class="org.apache.cocoon.webapps.authentication.context.AuthenticationContextProvider"
name="authentication"/>
</xconf>
```

I assumed therefore that these info was used by the cocoon build script to append information to the generated sitemap.xmap and other files. Problem is that the generated cocoon sitemap.xmap and cocoon.xconf is very different to the one in forrest. I found a build.xml script in \$FORREST_HOME/etc/cocoon_upgrade that used upgrade-cocoon-xconf.xsl transformation. I ran that on the newly generated cocoon.xconf in the cocoon build, but these version did not work with my existing forrest website. So, I resorted to plan B:

- build cocoon without the blocks I wanted
- build cocoon again but this time with blocks
- diff'ed the two build directories
- found that in essence only the sitemap.xmap and cocoon.xconf were different
- diff'ed the two versions of sitemap.xmap and added the extra entries into my website's sitemap.xmap
- diff'ed the two versions of cocoon.xconf and added the extra entries to the forrest's cocoon.xconf file.

I then tested the authentication pipe, and hey - it worked! I am sure that what I have done is very hacky, and would like to find out how to do the following:

- keep the modified cocoon.xconf within my website and use that via forrest.properties setting instead of having to modify forrest's own cocoon.xconf
- keep the generated jars in the website itself as opposed to moving them to forrest's lib and specify the location via forrest.properties, this way I can keep the forrest install as an original unit
- confirm if you really do need to run upgrade_cocoon_jars.sh or is copying the jar's adequate
- confirm whether or not I needed to rebuild forrest if I simply need to add another block
- figure out why the build script to import cocoon.xconf failed to produce a working version
- find or write a build script that will automate this whole process i.e. add-block-to-forrest.bat

4. Triumphant summary!

It's done, it's working! I grow to like cocoon and forrest even more. I can now get on with

creating profiled documents. Here is a brief summary of the steps you need to take to add a new block to forrest:

- download and extract forrest and cocoon distributions
- copy local build and block files from forrest to cocoon
- build cocoon and rename the build dir to build-without
- set the blocks you want to include to true in the local.blocks.properties file in cocoon's dir
- build cocoon again
- copy the generated block jars to forrest's lib dir
- diff the build and build-without dirs to determine which files have changed, pay particular attention to sitemap.xmap and cocoon.xconf
- incorporate the diff's in your forrest's cocoon.xconf and your website's sitemap.xmap

Now, if you are interested, read about [how to profile documents](#) with forrest article.