BuildProperties

Note: the content of this page is moved to 30.html. Updates to the content will be done there. This page will no longer be reviewed for updates and will be removed in due time.

This page is not **yet** a full explanation obviously. Please chip in.

Remember, with ant properties a property is not changed once it is set. The build first loads properties from cocoon.build.properties in the user's home directory, then from local.build.properties in the main project directory, then the original build.properties. If you set a value before Cocoon's default does, your value "wins".

From build.properties (should be copied to local.build.properties):

```
# ---- Webapp ------------------------------------------------------------------

#exclude.webapp.documentation=true
#exclude.webapp.javadocs=true
#exclude.webapp.idldocs=true
#exclude.webapp.scratchpad=true
#exclude.webapp.samples=true
```

The above properties affect what is included **in the webapp** (or war if you build that too).

- **documentation** the cocoon docs as seen on the website. The link for the documention will also be excluded from the main welcome page.
- **javadocs** the javadocs of all cocoon core and blocks classes. The link for the javadocs will also be excluded from the main welcome page.
- **idldocs** some modeling documentation on the flow object model (FOM) described in idl. The link for the FOM docs will also be excluded from the main welcome page.
- **scratchpad** new components or core functions under experimental development. The components will not be configured in sitemap.map, cocoon.xconf, etc.
- **samples** the functioning samples. This excludes core, blocks and scratchpad samples. (blocks and scratchpad may not have been included anyway depending on your choices in other config options).

```
# ---- Build Exclusions --------------------------------------------------------

#exclude.scratchpad=true
#exclude.deprecated=true
#exclude.javadocs=true
#exclude.idldocs=true
```

The above control what is actually assembled, whether they are included in the webapp or not.

- **scratchpad** scratchpad will not be compiled at all.
- **deprecated** classes moved to the deprecated source directory will not be compiled at all.
- **javadocs** the javadocs will not be created at all
- **idldocs** the idl FOM docs will not be created at all

```
# ---- Configuration -----------------------------------------------------------

#include.driver.oracle=true
#include.driver.postgre=true
#include.driver.odbc=true
#config.allow-reloads=true
#config.enable-uploads=true
```

The above are convenience settings to control some init-params in web.xml. All of these can be done or undone manually at any time. These build properties only exist for convenience. An easy way to see what they are doing is to look in the src/confpatch directory and look at the various .xweb files. These are used by the CustomConfigTarget target which you can also use for your own configuration changes either as a part of the Cocoon build, or by defining the xpatch task in your own build file.

- **include.driver.*** causes the driver name to be declared in the load-class param. They **DO NOT** cause the actual jar containing the driver to be placed in WEB-INF/lib nor do they configure your datasource in cocoon.xconf. They were added because declaring the class in web.xml was a frequently forgotten task.
- **allow-reloads** enables a magic "cocoon-reload" request parameter to cause the entire Cocoon controller object to be disposed and recreated. This is convenient when declaring new components in cocoon.xconf during development, but should be disabled for any live site. It is not needed to allow subsitemaps or xsps to reload when their source is modified.
- **enable-uploads** causes file uploads received via multipart html forms (with input type="file" elements) to be automatically parsed by Cocoon and available as objects in the request. There are at least two samples that will not work unless this property is set to true, but it is disabled by default. There are also more settings related to this in web.xml, and you can set **any** of these even after build time.

```
# ---- Validation ----------------------------------------------------------

validate.config=true
validate.xdocs=true
validate.jars=true
```

The validation targets ensure that various config files, documentation xml sources and the core, optional, and endorsed jars are in an expected state. There's really no reason to mess with these. The first two are fairly self-explanatory xml validiations, but the last bears a quick comment. Every jar used in the build is recorded with at least some explanation of its purpose in jars.xml. Only the lib/local directory is excluded from this requirement. The validate.jars target enforces this requirement.

```
# ---- Forrest -------------------------------------------------------------

forrest.home=../xml-forrest/build/dist/shbat/
```

If present, Cocoon uses Forrest (an Apache project based on Cocoon focused on building documentation) to build its own documentation. Forrest does not ship with Cocoon because of the circular dependency it would create. If not present, Cocoon still builds its own documentation.

NOTE: I am surmising this to be the case because I don't have Forrest present on my machine and the docs still build with the old look (GeoffHoward). I have not traced this through the build targets though.

```
# ---- Build ---------------------------------------------------------------

build.root=build
build=${build.root}/${name}-${version}
build.dest=${build}/classes
build.mocks=${build}/mocks
build.test=${build}/test
build.docs=${build}/docs
build.docs.printer=${build}/printer-docs
build.site=${build}/site
build.xdocs=${build}/xdocs
build.idldocs=${build}/idldocs
build.javadocs=${build}/javadocs
build.context=${build}/documentation
build.context.printer=${build}/printer-documentation
build.blocks=${build}/blocks
build.deprecated=${build}/deprecated
build.scratchpad=${build}/scratchpad
build.scratchpad.src=${build.scratchpad}/src
build.scratchpad.dest=${build.scratchpad}/dest
build.samples=${build}/samples
build.temp=${build}/temp

build.docs.loglevel=ERROR
build.docs.printer.loglevel=ERROR
```

These define directories used in the build and should not normally need to be changed by an end user. The two loglevel values define the logging verbosity during the building of the regular docs and the printer friendly docs (if included).

```
# ------ Webapp Build Properties ------------------------------------------

build.webapp=${build.root}/webapp
build.webapp.webinf=${build.webapp}/WEB-INF
build.webapp.classes=${build.webapp.webinf}/classes
build.webapp.lib=${build.webapp.webinf}/lib
build.webapp.samples=${build.webapp}/samples
build.webapp.docs=${build.webapp}/docs
build.webapp.javadocs=${build.webapp}/api/java
build.webapp.idldocs=${build.webapp}/api/fom
build.webapp.loglevel=INFO
build.war=${build}/${name}.war
```

Like the section above, these define directories used during the build. There are two which may be useful for end users:

- **build.webapp** could be changed to an external project folder to "seed" a project based on Cocoon.

- **build.webapp.loglevel** specifies the default loglevel used for logging while Cocoon runs. Like the configuration settings, this can be changed later, but it is convenient to do it here at build time.

```
# ------ Standalone-demo Build Properties -------------------------------------
build.standalone.demo=${build.root}/standalone-demo
```

The standalone demo target creates a directory with all the pieces (including the light version of Jetty) needed to zip up as a demo of Cocoon outside the normal directory. This could also be a starting point for a project based on Cocoon.

NOTE: Could someone who knows more about the Standalone target fill in here and check my facts? (GeoffHoward).

```
# ---- Compilation ------------------------------------------------------------

compiler=modern
compiler.debug=on
compiler.optimize=on
compiler.deprecation=off
compiler.nowarn=on
```

The above are compiler settings that would not normally need to be touched.

```
# ------ System Properties -----------------------------------------------------

# WARNING: you shouldn't need to modify anything below here since there is a
# very high change of breaking the build system. Do it only if you know what
# you're doing.
```

The properties following this warning have been ommited. If you have to ask, you shouldn't be messing with them!

Q&A

Q: Can I add an `#include.driver.mysql=true` to the Configuration section; if so, perhaps this should be a commented line already included in the shipped version, as I would guesstimate that this database has widespread use.

A: Yes you can, but for it to work you'll need to copy one of the existing .xweb driver patch files and modify it with the correct driver, the correct unless/if properties. I have meant to do this, but a patch from you will get the job done a lot faster!

CategoryInDaisy