BlockBuilderDevDoc

How to use the SVN version?

Checkout http(s)://svn.apache.org/repos/asf/cocoon/whiteboard/block-builder. Make sure, that when you call the ant script, the property and blockbuilder.root is available. The easiest way is adding a file with the name "local.block.build.properties" to the root directory of the block.

The content of local.block.build.properties could look like this:

```
# ---- BlockBuilder ------------------------------------------------------------

blockbuilder.root=../../../../dev


```

Then you can call one of the Ant tasks and compile the project (e.g. ant compile).

Directory structure

In order to explain what I have done, let's take a look at the file system:

```
cocoon
 |
 +--blocks
 |    |
 |    +--authentication-fw
 |        |
 |        +--trunk
 |        |   +--descriptor.xml
 |        |   +--build.xml
 |        |   +--legal [DIR]
 |        |   +--src
 |        |   |   +--java
 |        |   |       +--public
 |        |   |       +--private
 |        |   +--samples
 |        |
 |        +--branches
 |
 +--trunk
 |    +--build.xml
 |    +--src
 |    |   +--java
 |    |       +--core
 |    |       +--public
 |    |       +--private
 |    +--samples
 |    +--lib
 |        +--core
 |        +--endorsed
 |        +--blocks
 +--branches
 +--tags
```

- Blocks should move into its own place in SVN - as long as possible we only maintain trunk, if necessary (e.g. for XSP) branches too
- each block has a descriptor (see below)
- the java sources a separated into a public and a private section. We should move as much as possible into private
- if a block depends on another block it can only use public classes/interfaces
- in /trunk/lib there is a repository of all available libraries, rule is that a block ""has to"" use these libs (see block descriptor) if available, otherwise we end in jar versioning hell

Build system and Ant tasks

The build system is based on the descriptor. It uses its information to resolve all dependencies. Technically, the Ant script is generated via XSTL. This way, it can be reused in all blocks via Ant import statements. If necessary, for a block a certain build target can be overriden locally

```
<project default="compile" name="Build autentication-fw block">

    <property file="local.block.build.properties"/>
        <available file="${blockbuilder.root}" property="available.blockbuilder.root"/>
    <fail unless="available.blockbuilder.root"
            message="Property blockbuilder.root has to be set!"/>

        <xslt in="descriptor.xml"
                  out="build/temp/build-by-xslt.xml"
              style="${blockbuilder.root}/targets/block-descriptor2ant-script.xsl">
        </xslt>

        <import file="build/temp/build-by-xslt.xml"/>

</project>
```

To make it run, the property blockbuilder.root has to be set.

Each block has its block.build.properties (e.g. for the authentication-fw block):

```
# ---- Paths ------------------------------------------------------------------

src.public=src/public
lib.dir=../../../trunk/lib

# ---- References -------------------------------------------------------------
root.core=../../../trunk
root.block.session=../../session-fw/trunk
```

What's done so far?

- compiling and packaging that resolves dependencies (snowball effect)
- create the eclipse project for a single block that only loads the necessary libraries

Open issues

- global tasks (compile, webapp, javadoc, ...)
- create gump script
- How much flexibility do we need? (strict directory structure, package names, ...)
- user libraries repository
- splitting Cocoon core and the block sources in private and public parts
- eclipse task isn't error tolerant
- eclipse task not tested on non-win32 environments