AvalonAndGump

outdated

most of the information on this page is no longer relevant.

blah

Avalon is a part of the Gump Effort (see GumpProjectPages). As some of our developers (ie me) find Gump a bit scary and confusing, here's some info 'till we get used to it.

Gump uses an xml file @ {cvsdir}/jakarta-gump/profile/gump.xml which references lots of other files in {cvsdir}/jakarta-gump/repository (definitions of repository locations) and in {cvsdir}/jakarta-gump/project (which contains descriptors for modules (in some repository), and the various projects in those modules, where a project is "the atomic unit of integration".

Avalon is considered part of the cvs repo.

All avalon modules should have an xml file in the project dir to describe it (with a name of {modulename}.xml). Excalibur also has xml files per subproject (probably 'cause the xml file would otherwise be real big; they're named excalibur-{something}.xml).

Inside an xml file, you have a module definition which looks something like

```
<module name="{cvs-module-name}">

{{{    <url href="http://avalon.apache.org/{relevant-page}"/>
    <description>
      {description of module contents}
    </description>

    <cvs repository="jakarta"/>

    <project name="{cvs-module-name}-{project-name}">
        <package>org.apache.{package-name}</package>

        <ant/>

        <depend project="jakarta-ant" inherit="runtime"/>
        <option project="checkstyle" inherit="runtime"/>
        <depend project="{other-projects}"/>

        <work nested="{project-basedir}/build/classes"/>

        <home nested="{project-basedir}"/>
        <jar name="{project-basedir}/build/lib/{project-name}-@<at:var at:name="DATE" />@.jar"/>

        <nag from="Gump Integration Build &lt;avalon-dev@jakarta.apache.org&gt;"
            to="avalon-dev@jakarta.apache.org"/>
    </project>

    <!-- more projects in this cvs module here -->
```

</module>
}}}

the less stuff you need to define in the project definition in order to make things work, the better, as changes in the build files then don't affect gump too much. Other projects in Gump don't (shouldn't) refer to actual avalon jar files, but rather just to a particular project. Gump will then figure out the classpath. Similarly, instead of relying on some jar file in a lib directory, we depend on other projects in gump, and will get the latest cvs build of that project on the classpath. Real cool.

Successful nightly builds are put here. In the event of failure, the use of nag tags will alert the mailing list. It is important to fix these errors quickly as they cascade to other projects that depend on avalon. If things are not fixed, those projects won't have nightlies available!