GumpInternals

GumpInternals

Processing Step #1: Constructing Memory Structures

- The GumpMetadata is loaded (via file/HTTP) and 'completed' into the GumpModel
- The GumpModel is a tree of a workspace containing modules, which contain projects
- Projects in the tree are linked (bi-directionally) via dependencies.

- A GumpRun is build using a pattern match expression against project names, which contains the build sequence [of modules/projects].

Processing Step #2 : Performing the Work

Packages

- Packages are determined (and their state set to 'complete' or 'failed').

Updating from Source Control

- For updates, the build sequence of modules is traversed, any in a good state are cvs|svn updated. Any failures are propogated (see StatePropogation).

Build

- The build sequence of projects is traversed, any in a good state have the following:

Pre-build steps

- Create directories for <mkdir
- Delete directories for <delete Note: Currently disabled for security.

Build steps

- Build using script or ant (or maven). Classpath is determined/set.
- State is set/propogated based of exit code.

Post-build steps

- Check all outputs (jars/licenses) exist, or set state to failed.
- Copy outputs to Jar repository
- 'List' certain files (maven log, junit reports, etc.)

Processing Step #3: Generating Results

- RSS and Atom feeds are generated be walking the tree.
- A results.xml is generated, by walking the tree.
- Statistics are updated
- Documentation (of a run) is generated using Forrest or text.
- Nag e-mails are sent