Future Direction for CIShell Utilities

Future Direction for the Utilities

Current Problems

The current state of the utilities in CIShell has reached a point that it should be reworked. Some of the problems we have identified with the current implementation include:

- Lots of outdated code that later versions of Java have fixed
- Lots of poorly coded and buggy methods
- Lots of code that doesn't have anything to do with CIShell, but rather focuses on adding support for a tool or a library
- An almost universal lack of documentation
- Lots of code that is never used by any of our tools or CIShell itself

Solution

The short term fix will be to deprecate everything in the utilities plugin. **No new code should be added.**

The long term solution will be to have each CIShell utility in its own plugin (e.g. there is a plugin for Logging) that is either in a tool or in CIShell depending on which uses it.

Moving Forward

If you find a class in the utilities that you find useful, you should move it to it's own plugin. Only ones that are CIShell related should remain in CIShell. Others should be moved to the appropriate tool.

Once you've decided the utility is useful, you should remove any methods that do not make sense and add test code and documentation for each method in the utility.

Conservatively add methods to utilities; just because you think something would be useful does not mean that anyone will ever need or use it. If you had a candidate for a utility method, add it as a utility package within your own plugin. If another developer wishes to use the functionality, the onus is on them to extract the utility to the appropriate new plugin with tests and to verify the documentation.

Be very through when adding utility code since it is very difficult to remove or change the behavior once released.