**Offline Documentation Design**

Requirements:

The key requirement of this work is to generate and distribute offline help with the installed version of Mantid where the version of the documentation matches that of the code.

1. The offline docs will contain user documentation. Not things for developers. It may contain links on where to find developer information.
2. The source for the documentation should be auto-generated (from introspection or scraping from the mantidproject mediawiki) as much as possible, and include mechanisms for bringing in hand written documentation where introspection fails.
3. Additional documentation should be written in a markup language other than html. Favorites are wiki(medawiki), markdown (github) and rst (sphynx), as these are already in use within the project. There is extensive documentation in the mediawiki format.
4. It must be easy for the developers to maintain, it is beneficial if it is also alterable by trusted users.
5. There are a few generic areas the docs should cover. In the current menus these are (in priority order):
6. MantidPlot->Algorithms
7. MantidPlot-> Help
8. MantidPlot->User Interfaces
9. Mantid overview
10. MantidPlot->Fitting Functions
11. Mantid Python (the sphynx that is currently generated)
12. Developer links ­ like what is in github's readme
13. For the short term (at least) where there is duplication of the wiki there will be links to the wiki pages.
14. Offline documentation will be created for each announced release of Mantid.
15. Offline docs are generated at build/packaging time so they will always be in sync with the code
16. The resulting documentation files should be stored in a github repository and tagged with the release tag.

Caveat: We need to decide what the wiki describes (version and content).

The wiki will always describe the current state of master + completed new documentation from branches awaiting testing.

The wiki is currently the master of all content apart from algorithms, doxygen and sphinx documentation

Design:

It would appear that the most practical design would be to spider and parse the mediawiki information, removing all of the non-content parts of the page. This would cover all of these areas

1. MantidPlot->Algorithms (this data could also be extracted from the wikimaker)
2. MantidPlot-> Help
3. MantidPlot->User Interfaces
4. Mantid overview
5. MantidPlot->Fitting Functions