History

Some History (and babble)

GShell is the second coming of Twiddle (3rd if you count the first iteration I did for a certain other app server). But what the heck is Twiddle? Well, over the years I have realized that I spend way to much time writing framework around command-line tools, just to get a simple task accomplished. And each time I need to add a tiny other little feature I end up doing it all over again. Twiddle was intended to be a simple framework for implementing command-line tools, where only the functionality needs to be implemented, none of the plumbing. Yet I never really finished it... and then a while ago I got the bug to complete it... and thus was born GShell.

But what plumbing is there for command-line tools? Well, that becomes more apparent when you start to think about how commands (plural) can coexist and cooperate together. At that point you have the commands option parsing and functionality... plus interaction with input/output streams and environment variables, i18n loading, preferences handling, profile loading, blah, blah, blah.

<rant>
But Jason, why not use use the system's shell for this stuff? Well, as many of you know... the Windows cmd.exe sucks ass, and despite the profits of vista hoping that ms-foolios are going to fix it, I can't assume they will, since they have had years to pull their heads out and see the folly of batch.
</rant>

But, more so... I want to be able to SSH into my application server and run those commands. I want to be able to launch a script (language of my choice) in the server from a remote terminal and have it just work as I would expect it to. IMO that is really where the benefit of the common plumbing is.

Yet even before we get there, the plumbing allows for use to build simple and **consistent** command-line tools. Adding a new tool is as simple as adding a new jar to a directory, no need for more scripts to maintain, or configuration files, yada yada.

Where Reality Comes In

So, what GShell really is, is a simple and light-weight framework to facilitate building command-line applications that have a rich user experience.

At the moment, GShell has several significant features:

- Dynamic command discovery
- Rich JLine console
- Simple annotation-based CLI option/argument processing
- Support for fancy ANSI color muck
- Rich dependency injection and component management via Plexus
- A simple command-line syntax parser with minimal support for quoting ("", '') and for basic `${xxx}` expansion
- Branding support to allow application developers to customize the user experience

Many things are missing though... 😒 Some things which are planned to implemented very soon are:

- Customizable command layout (allows you to make a virtual filesystem to organize your commands)
- Remote shell support via custom protocol (ssh stuff may eventually come, but this is coming first)
- Annotation-based preferences access
- Annotation-based i18n access

And other things which are planned, but so far lack resources to implement:

- Full `/bin/bash` compatible shell syntax support