Packaging and Modularity

Possible Solutions

Partitions and Profiles

(We may want make cute names for these, like Note, Beat, Baby, etc. for partition, Tune, Song, or just Profile for Profile)

Partition / Grapes

Ideally every class belongs to one grape, a goal would be to reduce the size of the core or primordial grape and move all truly non-essential items into separate grapes in the core. (a leading name is GRAPE, Groove Advanced Package Management)

Possible Partitions

- Runtime support (groovy.lang, groovy.util, org.codehaus.groovy.runtime, etc)
- Groovy Classloader (includes ASM compiler)
- Swing
- Ant
- SQL
- JMX
- Servlet
- GUnit
- Shell tools (for javax.script support)
- Command Line Tools (Groovyc, GroovyConsole, etc).
- etc.

The exact code to use for grape management is a discussion that is ongoing on the dev list. Options are

- Re-use existing package management software, loading groovy stuff on top
    - OSGi (Apache Felix)
    - Maven
    - Ivy
- Roll our own package management software, mostly modeled off of another pattern
    - .deb / apt
    - Python .egg
    - Ruby Gems

Profile

A Profile is a collection of these partitions. For example, an Applet Profile would include the Swing partition and the WebServices partition. The number of profiles should be kept relatively small. Strictly speaking any of these profiles could be created by installing the kernal and package management portions and installing the grapes in the profile. These exist as an external packaging and instillation convenience.

Possible Profiles

- Embeded (just runtime)
- Enterprise (for J2EE integration)
- Tools (for shell scripts, includeing a command line package management tool)
- Desktop (For Client/applet work, includes SwingBuilder, SwingXBuilder, perhaps a Package Management GUI)
- Developer (net combindation of all, plus command line tools like GroovyC)