Grunt based build scripts planning

On This Page

- Pain Points
- Collaborative Development Project Roadmap
- Potential Technologies
- Requirements
    - Major daily use cases for a build tool
    - Feature Parity with ant scripts
    - Others
- ARCHITECTURAL SKETCH for "multi-resolution head rewriting"

See Also

- pirate pad
- node.js
- grunt.js
- grunt gettings started guidebower
- require.js
- existing build-scripts
- builder

The first pass at implementing Grunt based build scripts has been implement and committed to the project repo. (See: **FLUID-5120** - Getting issue details... STATUS)

This includes

- removal of ant based build scripts
- removal of CSS generation (currently a static set of pre-generated files is committed into the repo)
- dependency management of local modules
- minification
- custom builds

The builder site has been taken down, with no current plans of replacing it. For now, all builds should be done through the grunt build scripts.

Pain Points

1. Head rewriting to switch between using MyInfusion and individual files
2. The tools for debugging JavaScript are more mature in browsers than in Node.js; we'd like to be able to debug kettle apps in the browser (need 'require')
3. Our current builder tool is hard to update and maintain, doesn't support exclusions, and is feeling a little out of date
4. Working with multiple repositories is manual and tedious
    - managing relative include paths is error-prone
    - managing absolute paths is even worse
5. Version handling
    - developers want to be able to say "I want exactly this <version>/<revision>/<repository> of Infusion"
    - currently not possible to use multiple version of infusion from the same pre-release (e.g. 1.5-SNAPSHOT), but a user might like 1.5-REV_HASH
6. Having to build CSS (for UIO) is awkward and easy to forget
    - a CSS pre-processor (less, sass) may be the correct way to implement this.
7. Maven and Ant are hard to setup.
8. No top-level "infusion" directory when you unpack the built zip

Collaborative Development Project Roadmap

1. KILL MAVEN
    - current work
2. Port existing Ant scripts to Grunt
3. Update the PHP Builder to use these Grunt scripts instead of Ant (ASK Cindy for help)

Potential Technologies

1. Node.js
2. Grunt
3. Bower (what does this offer over just npm?)
4. Require.js (will need some form of require, but may not be Require.js)

JIRAs for require-type functionality:

- FLUID-4911 - "Remove requireStub in favour of the default pattern" - contains BIG COMMENT
- FLUID-4675 - "Stub out require() and fluid.require()"

Requirements

Major daily use cases for a build tool

1. Allow the efficient use of "require-style" code in production - the two non-build-time approaches to this are unacceptable in performance -
    - The use of "synchronous AJAX" is conceivable only for use in test cases
    - The use of "asynchrouons AJAX" is still unsupportable for production code, given its great inferiority in performance to code written literally into head in <script> blocks - this is the only good option for fast IPL code
2. Facilitate "multi-repository" work styles together with a checkout of Infusion
    - Right now, the biggest barrier to "unbundling" components from our image, or spawning new repositories based on infusion, is the annoyance of ensuring a "stable relative checkout path" between the different repos. We should have a "one-stop shop" that rebases a checkout relative to a checkout of infusion, but also somehow "by magic" not corrupting tags written into the documents when the user tries to checkin again.
    - **IDEA SCHEME:** perhaps a "magic eraser" mode for the build tool, that is run immediately before any "git commit" in order to wipe out evidence of concrete relative include paths in an Infusion-dependent project - then IMMEDIATELY AFTER git commit, the build tool is run again to unpack the relative paths again
3. Replacement for our pre-flood PHP-based Infusion builder tool on the WEB

Feature Parity with ant scripts

1. Minify js files
2. Concatenate js files
3. Create zip bundles
4. Generate UIO Themes
5. Rewrite URL's in demos/unit tests for relase testing
    a. change URL's to point at the concatenated js file
6. Dependency Management
    a. Include modules
    b. Exclude modules

Others

1. pull in dependencies (e.g. jQuery)
2. minify
3. concatenate
4. build css (e.g. UIO themes)
5. build font icon sets (see: https://github.com/twolfson/grunt-fontsmith)

ARCHITECTURAL SKETCH for "multi-resolution head rewriting"

Should be possible to write a comment, say, that directs "Include Infusion" - and this is expanded to various levels of detail - this can "pack and unpack" based on the development style (when debugging, running a "demo", or "full production").