Loading Groovy files via Resources API

DEV-124 - Getting issue details... STATUS

Rationale

After a first PoC done using custom code and a subsequent architecture meeting it was deliberated to use the Resources API instead. This concept aims at investigating some of the questions raised during that first meeting.

First meeting decisions

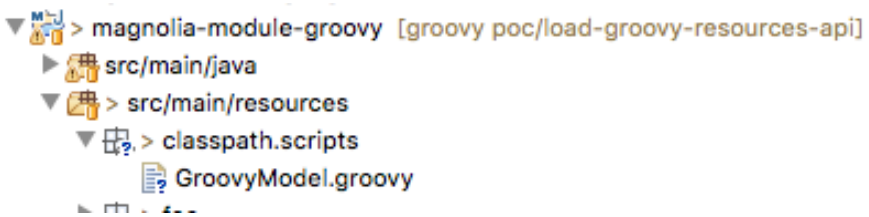
- **Use Resources API**
- **Split scripts vs. classes** (scripts stay in current workspace and functionality for executing scripts stays in the Groovy module. Support for classes will be removed from Groovy module and placed in resources workspace)
- **Maintain support for scripts in JCR > scripts**

Open questions

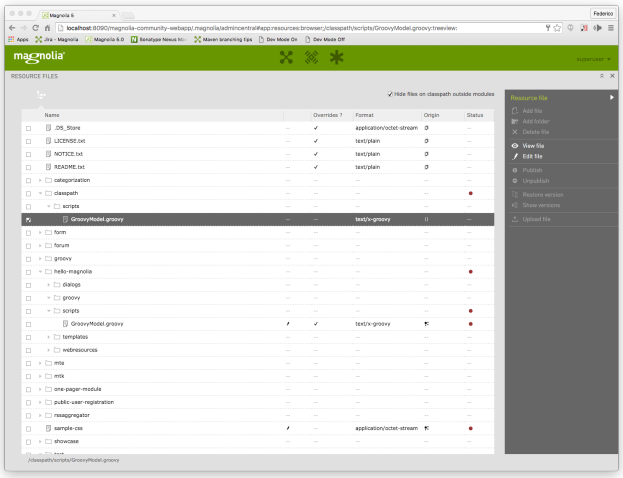
A new PoC using the Resources API has been crafted In order to quickly verify some assumptions and help with the investigation. See https://git.magnolia-cms.com/projects/MODULES/repos/groovy/compare/commits?sourceBranch=refs%2Fheads%2Fpoc%2Fload-groovy-resources-api

Maintain additional classpath fallback after the cascade? Such classes would not show in the Resources app, so how can someone edit them?

Apparently such classes do show up in the Resources App. I created a Groovy script in a package `classpath.scripts` in Eclipse under `src/main/resources`



and the script is displayed in the app and can be *hot fixed* with no problem. The funny thing (but it may be due to the cp resource issues being tackled atm or to my borked IDE setup) is that it only shows up after the parent folder is manually created in the app.



Find a way to load classes via Resources API in deferred manner.

The Groovy module replaces Magnolia's `DefaultClassFactory` with its `GroovyClassFactory` which internally delegates loading the Groovy source files to `MgnlGroovyResourceLoader#loadGroovySource(..)`. The latter is therefore called very early during Magnolia startup (it is indeed called internally by the `GroovyClassLoader`) when `ResourceOrigin` (the entry point for clients of the Resources API) is not ready yet. That would cause a `NoSuchComponentException` when trying to get hold of `ResourceOrigin` via `Components.getComponent(..)`.
Solution in PoC consists in using a `ResourceOriginProvider` and skip Groovy resources loading as long as the provider is null. This seems to be safe as during Magnolia's early startup phase there are no Groovy scripts/classes to be instantiated and the loading of Java classes needed by Magnolia is delegated to the parent class loader.

How/where to locate scripts in file system? (The packaging / structuring issue)

- **module-names may contain dashes but Java does not allow having a package name with dashes. Scripts (which are eventually compiled as Java classes) need to declare a valid, unique package to compile and for this whole feature to work.**
- **if module name is in the package name dash (-) needs to be replaced (by underscore? or use some other convention)**
- **next to the templates (convenient to edit, but hard to specify correct package name) vs. special folder (would make it easier to have package name w/o module name in it**
- **ideally would favor no special tricks**

As far as I could see, having a special folder in the light module structure, e.g. one called `groovy`, would not solve the issue because

- we still would need the module name in order to resolve the path to the file see line #185 at MgnlGroovyResourceLoader
- also for a sub-folder of the special folder users might decide to use dashes

Although a little unfortunate, *I see no better solution than establishing the "convention" of replacing dashes with underscores* in package names in order to have unique paths to Groovy scripts. This convention is also endorsed somehow by Oracle itself https://docs.oracle.com/javase/tutorial/java/package/namingpkgs.html PM too Christopher Zimmermann seemed to be able to live with it (maybe he even suggested it, can't recall) 😑

Next steps and more questions

If the solution outlined here and in the PoC is validated here's what would be left to do

- adapt Groovy module tests to use Resources API
- migrate **enabled** Groovy sources (**non script** ones) from the `scripts` to the `resources` workspace. Groovy source is found in a property called `text` which is also available for resources.
- remove the `script` and `enabled` fields at /modules/groovy/apps/groovy/subApps/detail/editor/form/tabs/script/fields and their values in the remaining scripts in the `scripts` workspace.
- move/adapt Groovy validation to Groovy files in `resources`, e.g. when created in the app or hot fixed
- still re validation, how to signal an invalid Groovy file (syntax, package matching folder structure under light module) when working on the file system?
    - in *dev mode* observe Groovy files via Resources API and validate on resource added/changed, throwing an exception in logs like we do for configuration definitions?
- Do all these novelties and changes require a new Groovy module major version, i.e. `2.5` ?
    - Probably they do.