Concept REST module

⚠ **In progress for 5.2**

A module that helps exposing REST APIs in Magnolia as well as exposes some basic services.

Rationale

We need to expose some of Magnolia's functionality through a REST-style API, among other for integration purposes.

We also need to have a mechanism for modules to plug in their own REST endpoints.

High level requirements

Scope:

- ⊕ In for sure
- ℹ Good to have but can be postponed
- ⊖ Out of scope

Priority: ★★★ to ★

Status:

- ✓ Done
- ✗ Incomplete

The idea with these scopes is to do the minimum minimorum. Anything else can be rescoped later, done on a per-project basis, or even as a separate module. We need to stay focused and get the minimum scope out of the door as soon as possible, to possibly reassign resources to other priorities before the release.

Split ⊕ ★★★ MGNLREST-2 - Getting issue details... STATUS

Currently, everything is in a single module. We should split it in a multi-module project:

- A module which allows the integration of arbitrary service - `magnolia-rest-integration`
- A module (or two or three) which exposes our services - `magnolia-rest-services`
- A tooling module - for self-documentation - `magnolia-rest-tools` (if this ends up being just about the self-doc, name could be `magnolia-rest-docu`

Status: not done. Existing code essentially all moves to `magnolia-rest-integration`. Code under the `info.magnolia.rest.json` and `info.magnolia.rest.json.tree` must be removed. Currently kept as a very basic example.

Promote from forge/ to modules/

Greg will do this upon return. SYS-331 - Getting issue details... STATUS

Allow exposing JAX-RS services

- Use node2bean and observation
- Services are exposed are exposed under `/.rest/` `MGNLREST-9 - Getting issue details...` STATUS
- Status:
  - works
  - remove mentions of endpoints being configured in module descriptors, this is out of scope (and would be inconsistent)
    - undeprecated info.magnolia.rest.RestDispatcherServlet ? It's still the real entry point, right ?
    - register end points under `<my-module>/rest-services` or `rest-endpoints`.
  - there is an issue with observation / restarting of service MGNLREST-7 - Getting issue details... STATUS ?

Authentication

- As a start, we can rely on simple http authentication (which is already implemented)
- We could also consider support token-based authentication - to be researched: I don't think this is a session mechanism per se (i.e no storage other than the token validity)
  - If we want "sticky" sessions, a token needs to be passed for every request.
- Soon enough, we'll need to provide ways to force redirections to HTTPS (currently doable, but very ad-hoc - a generic mechanism to mark an area as "SSL only" would be neat. For websites, we'll want to force https on login forms, and redirect back to http after login)
  - This should be a "core" feature, and/or a generic feature outside of the rest module.

Authorization

- Content-based services are already covered by our security framework
- But we need to be able to restrict access to other services, or restrict access to content services differently anyway
  - See Concept Security and ACLs
- As a first resort, we can always rely on the URI security (url-path-based security) to restrict access to all or certain services
- Ideally, to stay within the style of JAX-RS annotated services, perhaps we could implement support for the `@RolesAllowed` annotation.
  - http://stackoverflow.com/questions/9690574/how-can-a-jax-rs-rest-service-have-authentication-handled-by-annotations
- **Current status:**
  - Role `rest` enables access to the REST services (added by default to `superuser`)
  - `anonymous` role is modified on publicInstance to DENY access to REST services
  - Problem:
    - Currently, there is no possibility to DENY access to REST services by default (for existing/new users)
  - Workaround:
    - DENY access to REST services by default in role `security-base`

Expose basic content-access endpoints

- Simple workspace/path GET method MGNLREST-4 - Getting issue details... STATUS
  - 3 parameters: MGNLREST-10 - Getting issue details... STATUS
    - depth : Determines the depth of children to return
    - filter/include: type (include/exclude which types of subnodes to include/exclude) - defaults to include all
    - include meta properties: whether or not we include `jcr:*` and `mgnl:*` properties in the response
  - What should the return type be ?
    - Default return type could be json or xml, but we should "listen" to what the Accepts headers
    - Avoid something like `info.magnolia.rest.tree.RepositoryNode` which will un-de-re-marschall the returned values twice (once to construct RepositoryNode, once by Jaxb) and kill performance. Instead,
      - See what happens if we feed a Node to Jaxb (bad things, probably)
      - See if we can wrap that in some sort of lazy bean (tbd if we can do something about properties, so that the returned xml/json looks "flat" (see example below)
      - Or use a custom "Marshaller" ?
      - (JSOP and Sling might have good examples or reusable things there)
      - RepositoryNode exposes "name" and "path", which are redundant with the request (I know what I asked for). However, such things could maybe be included in the response as headers ?

        - If we use a custom Marshaller (or re-use someone else's), the **brilliant** side-effect is that a service can be written with methods that simply return `javax.jcr.Node` (or NodeIterator or whatev). Learning curve ? Gone !
- Simple service to write data ? MGNLREST-5 - Getting issue details... STATUS
    - If we can have auth token, split the `save` operation
- Document this/these service(s) explicitly as "not for extensive use" ? Jan Haderka's points against such services is that users will abuse it. Grégory Joseph's point in favor of it was 1) we need a basic service that's useable everywhere as a starting point, 2) having a page-only service will not cover anyone's needs. If we have both, then "basic" is "for simple, once-in-a-while types of use-cases" and "page-oriented" service serves as an example of what a project-oriented service could be like. Jan Haderka what do you think?

Expected response example

This is what I'd expect a returned node to look like:

```
{
  "jcr:primaryType": "nt:unstructured",
  "jcr:uuid": "....",
  "someProperty": "someValue",
  "a": { /* This is subnode */
        "jcr:primaryType":"nt:unstructured"
        "foo":"bar",
        "someInt": 4
       },
  "b": {
        "jcr:primaryType":"nt:unstructured"}
        "foo":"baz",
}
```

Status:

We have `info.magnolia.rest.json.RepositoryJsonEndpoint` but this should be entirely revised or removed for a number of reason:

- The service should not be tied to JSON
- Should use IoC
- Should use node API
- Should use results of the mini-research on tokens (see Authentication)
    - If we can pass a token around, the `Session` should be reused, and a `save` operation would have to be called for write operations.
    - If we can't, than means we need to authenticate the user at every request (basic auth) AND save after every call.
        - Current code uses `ExclusiveWrite`, what's the current good approach ?

Out of scope:

- Use an existing JCR-over-REST API ?
    - prob. not as we have to implement more specific operations
    - See ModeShape
    - See JSOP
    - -> This could be implemented later, and/or as an independent module.
- Look into doing something like a REST Level 3 API ?
    - http://www.slideshare.net/matt_bishop5/l3-rest
    - Seems overkill for now, and could be implemented as an extra module, after more research.
        - But might be necessary if we want good tooling ?
    - Level 5 seems "interesting" (...) but not relevant to a generalist JCR API. In short it seems to be very project-specific. http://www.slideshare.net/cappelaere/rest-level-5-a-trek-to-the-summit
    - -> This could be implemented later, and/or as an independent module.

Expose higher level services

- Execute arbitrary command MGNLREST-6 - Getting issue details... STATUS
    - Pass catalog, command name, map of extra parameters
- Get page, area, component
    - Can also serve as an example of what a project-specific service would be like.
- Custom/advanced services can/should be implemented at project level.
- Get rendered page, area, component
- Get specific data type (contact, etc)

- ? collect business cases for the high level API [Karel, Lars]
  - See subpages. Use this as examples of what people expect.

Self-documentation of available services ⊕ ℹ ★ ★ ✓

- Exposed services should ideally self-document
- ⊕ ★ ★ This is IMO more important than a full, advanced API. It empowers our users and will give THEM the right tools to expose services that fit THEIR needs. Ideally this whole framework will also foster contribution.
- MGNLREST-1 - Getting issue details... STATUS
- Jira and Elastic Path provide such a developer tool to browse/test the API
- Reuse existing tools for this
  - Swagger ? https://developers.helloreverb.com/swagger/
  - Or WADL with Maven plugin - https://developer.atlassian.com/display/DOCS/Documenting+your+APIs+with+the+Atlassian+REST+API+Browser

Versioning of web services ⊕ ★ ★ ★ ✓ MGNLREST-16 - Getting issue details... STATUS

- we have to support multiple versions of web services
- we will put versions in the URI
  - between service path and method name
  - we use v1, v2, v...
- java code (idea)
  - split endpoint and service impl
    - service does the bulk of work
    - endpoint only does receive/respond but delegate to work to the service

Out of scope for this project/version

- Security overview page ? "Who has access to which service"
  - This is something that every single feature, integration point, .. could benefit from. Not at all specific to REST.
- Integrate Sling ?
  - osgi-based. don't want to get down that rabbit hole
  - too complex
  - COULD be done as independent project/module
- Declare REST end points in module descriptor ?
  - no, would be inconsistent

Status

Git repository	`forge/rest.git`	Move to modules/
GroupID	`info.magnolia.rest`	✓
ArtifactID	`magnolia-rest`	Split (see below)

During the first Magnolia 5 proof-of-concept phase, we already implemented something very similar. It was extracted back in May 2013 into its own module, residing at the time of writing at `forge/rest.git`: https://git.magnolia-cms.com/gitweb/?p=forge/rest.git;a=summary

We currently use RestEasy, a JAX-RS implementation. This works for now, but we could we swap for something else later, should we find limitations.

References

Jira: MGNLREST

https://www.evernote.com/shard/s248/sh/87d21417-004e-4753-8920-0e232e529d32/fd456194d25cd9887930637ec7d71f67

Concept REST Services (Outdated)

Open REST issue on JIRA