Concept - Link API overhaul

Purpose

ARCHI-4 - Jira project doesn't exist or you don't have permission to view it.

Current situation

Our current Link API (`info.magnolia.link`) is highly coupled with pages. It provides

- a "storage format" (aka "uuid links") which it can transform to and from. This is used mostly in rich-text editor. We store html like below; links are "processed" and transformed at runtime.
- there are different link transformers - `AbsolutePathTransformer`, `RelativePathTransformer`, `CompletePathTransformer`, which should be able to generate links like `/magnoliaPublic/foo/bar.html`, `../bar.html` or `http://www.example.com/foo/bar.html`respectively.
  - there was also `EditorLinkTransformer`, which was used specifically in rich text editors, but it seems this isn't needed anymore since 5.0
  - tthere is also `STKEditorLinkTransformer` in the multisite module, presumably to handle linking through different sites.

Example of the current "storage format"

```
"<p>Large article pages have a <a href=\"${link:{uuid:{2a98b29f-b514-4949-9cb3-e1162171a2ca},repository:{website},handle:{/features/special-templates},nodeData:{},extension:{html}}}\">Table Of Contents</a> (<a href=\"${link:{uuid:{},repository:{website},handle:{/},nodeData:{},extension:{html}}}\">TOC</a>) navigation.</p>"
```

The Problem(s)

- The “storage format” is tightly coupled with the notion that the content it links to is stored in JCR.
  - We can't use it for DAM, since DAM Assets could be "anywhere" (depending on the provider)
  - We currently abuse it for DAM, such that it really only works with the one default (JCR-based) AssetProvider
- There is no consistent way of generating links to various types of content in the system. Every module or template component "invents" its own way to generate links to its content.
  - Dam's `Asset` links to a servlet
  - For categories it's entirely hidden in the templates IIRC, with using a "selector" or a query param
  - Likewise for the forum, where the "intelligence" is spread between the template scripts (they know which request parameter to add, such as `?threadId=xxx`) and the editors (for every forum-list/forum-view/thread-view page they have to select all the other corresponding pages)
- `Link` is a concrete class, we can't (elegantly) have different types of content return different types of `Link` instances.
- Links handling in RTE is hardcoded.
- ~~MGNLDAM-504~~ - RichTextEditor: DAM image URLs in the rich text editor contain the context path in the created link (and stored text property) CLOSED
- All of this also makes it ~impossible to properly know about content dependencies (page to page, asset to page etc)

Goals

- Be able to link to *any* type of content, including but not limited to content from *any* workspace, as well as external.
- Uniformize how links are handled in the product. Avoid spreading of "knowledge" about how to link to an image, a forum thread, etc etc. Be able to have a component responsible for "links to a forum", "links to asset" etc, and have all those transformed uniformly.

Goals 2

This could be a second step:

- When saving a "link", any component should register references
- These could have one of several formats, but they'd essentially be a pointer to another node
    - this is also valid for external urls, external dams etc, with some limitations
- This can then be used to know dependencies between pages/nodes/assets etc.
    - Useful for activation
    - For caching
    - For generating static sites
    - Simply for user information
- Here's a mind map with some ideas about that: Screen Shot 2014-08-15 at 11.27.51.png
- Storage:
    - 1 multi-value property (with a specific format)
    - or 1 node with a given type, under which each property is one ref (with a specific format)
    - or N nodes with a given type, under which the properties are more typed (workspace, uuid and/or url, external(boolean), type, ...)
- Enables quick query (select * from mgnl:reference or select mgnl:ref from mgnl:content, ...)
- The storage format *in RTE* could then possibly just point to these references rahter than duplicate their content (`<a href="{link:foo1}">hello</a>`)

Use Cases

- From rich text editor or a link field in a dialog, create a "reference" to any content app (see Concept - Linking with any app content)
    - From RTE, it will stored in the html in a "storage format" to be processed later
    - Templating will be able to "render" such references
    - ~~MGNLDAM-504~~ - RichTextEditor: DAM image URLs in the rich text editor contain the context path in the created link (and stored text property) CLOSED
- Forum: pass a forum, thread or message node, the Forum Link Provider knows how to build appropriate links to the appropriate forum page.
    - Editors shouldn't have to select where the forum/thread page is, the link provider could figure it out (by finding it or by configuration)
    - Request parameter names should be hidden and replicated in several places
- Contacts: a reference to a node of the contacts workspace could be transformed into a link to a .vcf file
    - ContantLinkProvider knows where the VCFServlet is (yes, this is an entirely
- Categories: a reference to a node of the categories workspace links to the appropriate "categories" pages.
- With navigation refactorings potentially happening, (sub-) navigation menus could be generated and take advantage of the below to generate menus to categories, forums, ...
    - This could be use (in navigation)
- Search: in an aggregated search scenario, link providers could simply generate the correct links to whatever content is being returned by the search
- DAM links should be trackable in a campaign scenario
    - Example: a page is rendered with a "newsletter" variation, in which a different link transformer is used (it generate links with tracking codes). If this page contains links to DAM, they won't be processed. (DAM links are currently generated outside the link API, they don't go through LinkTransformers)

Proposal

- Modules can contribute "LinkProvider" implementations
- A LinkProvider can parse and generate its own storage-format. The storage format would be something like {<link-type>:<here-is-whatever-that-link-provider-is-able-to-parse>}, so we can identify which link provider to use, or we have a chain-of-responsibility of sorts, where each provider is asked if they can handle this link.
- A LinkProvider returns instances/implementations of Link rather than passing Strings around.
- `Link` is an interface
- We can use either an arbitrary object as an "input" (typically a Node, but why not an Asset, or whatever bean we're integrating with

Ideas & open questions

- image variation links: should we include some sort of “variation” in the new link API, or try to keep those 2 clearly independent ? (href vs src attributes, click vs view ? huuuu)
- likewise,

~~MAGNOLIA-3925~~ - MagnoliaTemplatingUtilities.createLink should have a version where you can supply a custom extension... or none at all CLOSED

kinda goes in the direction of having "variations" possible on all links. Maybe "variations" is at the same time too generic and too restrictive - for an image, we probably just want 1 string ("variation"), but for a page, we might want "selector", "extension", "request params", "anchor", ... from an API perspective I guess those could be exposed on the Link implementation (with setter/builder methods), but i'm not sure how that'd look like in templating.

- Are Link instances mutable or not ?
- Keep the notion of LinkTransformers, because we still need that, but does it need to be a separate component, or simply individual methods on the Link interface ?
    - Separate components: there are use-cases where people have implemented their own transformers; but would that mean implementing support in transformers for each type of link ? Or have them delegate to the Link implementation and THEN postprocess the result ? Maybe Link only returns one canonical String (no domain, no context path) and the transformers do the rest ? Or they work by having a model of the url that the transformer is free to further manipulate ? (e.g String getPath, Map getQueryParameters, getAnchor, ...)
- Package and class names:
    - It's a bit unfortunate that `info.magnolia.link` is taken. Do we find a new package name, or do we use the same ? If we use the same package, we're very limited in terms of class name choices and backwards compatibility.
- If we're courageous, look into `info.magnolia.cms.beans.config.URI2RepositoryMapping` ? The code in this class is scary.

Existing concepts

Some are covered by this, some expand on particular use-cases.

- Concept - Link API review for M5
- Concept - Linking with any app content
- Concept - uuid or permanent links

~~MAGNOLIA-3925~~ - MagnoliaTemplatingUtilities.createLink should have a version where you can supply a custom extension... or none at all
CLOSED

Proposal - merging of URI2RepositoryMapping and VirtualURIMappings