Concept Dynamic Servlet Mapping

Implemented in 5.3

See ~~MAGNOLIA-5708~~ - A way for Servlet to "self map" - so that mapping can be shared with other components CLOSED

Purpose

The Problem

Servlet mappings are currently configured under `/filters/servlets`. Often times, other components in the system need to know about those mappings. A simple example: generating links for the DAM servlet.

Goals

Make it possible for arbitrary components to know how/where a servlet is mapped.

Use Cases

- Generating DAM links
- Generating links to Imaging module
- Links to REST APIs
- essentially anything exposed via a Servlet where we need to generate links to.

Such a mapping could be configured in the module's own class - and therefore used by the servlet mapping AND other arbitrary components (URI2RepoMapping, Link API, templating functions, ...)

Concept

Introduce an optional interface that servlets can implement. The interface exposes a method which servlets implement to return their "self" mapping.

Servlets can still be used with the current mechanism, aka "configured mappings", with the new one ({{SelfMappingServlet}}, or both.

This thus allows other components to use the same path, configured in one single place, for different purposes (typically to generate links)

(Where this method would delegate to MyModule.getFooPath(), and MyModule would be @injected in said filter.

No changes in configuration are necessary. The same filter is still used to wrap the servlet (`info.magnolia.cms.filters.ServletDispatchingFilter`). The new optional interface is `info.magnolia.cms.filters.SelfMappingServlet`.

Implementation example:

```
public class DamDownloadServlet extends HttpServlet implements SelfMappingServlet {
    // ...
    private final DamCoreConfiguration configuration;


    @Inject
    public DamDownloadServlet(DamCoreConfiguration configuration) {
        this.configuration = configuration;
    }


    @Override
    public String getSelfMappingPath() {
        return configuration.getDownloadPath() + "/*";
    }


    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
    // ...
```

In the above example, `DamCoreConfiguration` is a module class which represents the DAM module's configuration (see screenshot below) and is simply injected in the servlet. If the configuration changes, the mapping changes is reflected.