Configuring Swagger

Overview

Documents how to configure the swagger API to work properly.

Details

Swagger is a REST API framework that allows for easy documenation of REST services via annotations and includes an application for allowing users to interact with your services through a simple web interface.

There are several parts to making the Swagger API work properly and they are documented here.

- First, Swagger UI is included with the mapping-rest module so that the "index.html" page under its deployed context path shows the API.
- The web.xml file for the mapping-rest module requires a section to configure Swagger UI:

```
<!-- Swagger configuration -->
<servlet>
  <servlet-name>JerseyJaxrsConfig</servlet-name>
  <servlet-class>com.wordnik.swagger.jersey.config.JerseyJaxrsConfig</servlet-class>
  <init-param>
    <param-name>api.version</param-name>
    <param-value>1.0.0</param-value>
  </init-param>
  <init-param>
    <!-- This URL is the path to the REST web service itself. In the index.html
      page is the path to the api-docs page -->
    <param-name>swagger.api.basepath</param-name>
    <param-value>${base.url}</param-value>
  </init-param>
  <load-on-startup>2</load-on-startup>
</servlet>
```

- The "base.url" should point to the deployed context path of the mapping-rest application (e.g. http://localhost:8080/mapping-rest for a dev deployment).
  - This property is configured in the "config.properties" file used in conjunction with the build.
  - Note: the base.url works well if expressed as a fully qualified URL (not a relative URL). If setting to a relative value, you need to make sure the
- Additionally, the web.xml file for the mapping-rest module configures Jersey and this must contain entries for the Java packages containing code with Swagger API annotations. Note the presence of the "org.ihtsdo.otf" entries in the Jersey configuration of "com.sun.jersey.config.property.packages". This ensures the relevant code is available to the Swagger installation.

```
<!-- Jersey configuration -->
<servlet>
  <servlet-name>Jersey</servlet-name>
  <servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>
  <init-param>
    <param-name>com.sun.jersey.config.property.packages</param-name>
    <param-value>com.fasterxml.jackson.jaxrs.json;org.ihtsdo.otf.mapping.rest;
          org.ihtsdo.otf.mapping.model;
          org.ihtsdo.otf.mapping.rf2;
          org.ihtsdo.otf.mapping.rf2.jpa;
          com.wordnik.swagger.jersey.listing</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
```

- The index.html contains some javascript that informs swagger of the location of the "api-docs" Swagger UI uses to display services info based on the annotations. Here, the value of "base.url" is the same as above (e.g. http://localhost:8080/mapping-rest for a dev deployment).

```
  $(function () {
    window.swaggerUi = new SwaggerUi({
    url: "${base.url}/api-docs",
  ...
```

The code itself is annotated with Swagger annotations that make all of this work. Consider the "findConceptsForQuery" method of the Content REST service:

```
@Path("/content")
@Api(value = "/content", description = "Operations to retrieve RF2 content for a terminology.")
@Produces({
    MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML
})
public class ContentServiceRest extends RootServiceRest {



  @ApiOperation(value = "Find concepts matching a search query.", notes = "Gets a list of search results
that match the lucene query.", response = String.class)
  public SearchResultList findConceptsForQuery(
    @ApiParam(value = "Query, e.g. 'heart attack'", required = true) @PathParam("string") String
searchString,
    @ApiParam(value = "Authorization token", required = true) @HeaderParam("Authorization") String
authToken) {
   ...
```

The class itself has an @Api annotation that describes this overall service. The method itself has an @ApiOperation that describes what this service call itself does. Then, each parameter has an @ApiParam annotation that describes the purpose of that parameter and whether it is required. An attempt was made to provide example values that in a dev deployment of the system produce real values.

Troubleshooting

There are two minor nits with the current configuration to be aware of.

1. When entering the URL for Swagger UI, you have to include "index.html" in the URL (e.g. http://localhost:8080/mapping-rest/index.html for a dev deployment).
2. When configuring "base.url" you have to use a fully qualified base url. Various efforts to configure it with a relative URL (e.g. just the context path of the application) allow Swagger UI to work, but interactively trying to call the services does not work.

References/Links

- http://swagger.io/