PEARSON ALWAYS LEARNING

EQUELLA®

Java REST Tutorial

Version 6.4

Document History

Document No.	Reviewed		Finalised		Published	
1	22/05/2015		22/05/2015			

May 2015 edition.

Contact documentation@equella.com for matters relating to documentation.

Table of Contents

EQUELLA Java REST Tutorial

This document covers an example Java REST client (hereafter "the client") which is a J2EE web application and can be deployed into any J2EE servlet container, e.g. Tomcat (http://tomcat.apache.org/download-70.cgi)

To compile and execute the example code you will need to download a number of library jar files. See the readme.txt file in the **samples/java/public_html/WEB-INF/lib** directory for more information.

If you wish to run the client you will need to build the Java war file before deploying to a web container. To do this you will need to have Apache Ant installed. Once Ant is installed simply run the following on a command prompt:

```
ant product
```

To deploy into Tomcat simply copy the restclient.war file into the Tomcat webapps folder. The war file will automatically be expanded into a restclient folder when Tomcat is running.

You must configure the client via the config.properties file found in the WEB-INF folder of the expanded client. Instructions are included in the config.properties file.

The client is rather basic and does not do any checking of response codes to ensure that everything works as expected (it's intended as demonstration only). Any errors are simply thrown back at the user and may be rather cryptic. Anyone wishing to use the client as a base for further enhancements may wish to make it more robust.

Third Party Libraries

The client relies directly on the following third-party libraries:

- Apache HTTP Client 4 (http://hc.apache.org/httpcomponents-client-ga/) to make HTTP requests and read responses from the EQUELLA server.
- Apache Commons FileUpload (http://commons.apache.org/fileupload/) for reading multipart requests
- Jackson (http://jackson.codehaus.org/) to parse JSON responses and manipulate JSON objects.
- Freemarker (http://freemarker.sourceforge.net/) to render web pages

Login

The client uses OAuth "Authorization Code Grant" login mechanism *(*http://tools.ietf.org/html/draft-ietf-oauth-v2-23#section-4.1*)*. For more information on OAuth see the 6.*4 EQUELLA REST API Guide*.

Each page is protected by the CheckLoginFilter which requires that a token has been obtained for the current user session. If there is no token the user is redirected to the LoginServlet.

The LoginServlet will first redirect to the EQUELLA ***OAuth*** code-request URL, with the URL of the client embedded in the query string. This will force the user to login to EQUELLA (if not already logged in, for example in another browser tab).

Once logged in, EQUELLA will redirect back to the client with a code embedded in the query string. The client takes this code and requests a token from EQUELLA, which it adds to the user's session, and redirects to the client's /search page.

Search

The search page shows all of the fields that can be supplied to the REST API endpoint ***/api/search*** (except the info and collections parameters) and displays the formatted results of the search below the fields. Clicking on a search result will take the user to the client's /view page.

Searching for a resource will perform the following steps in the client

1. SearchServlet retrieves all form data and passes them to the EquellaApiService.search method
2. The EquellaApiService.search method GETs the search API endpoint with the relevant query string parameters e.g.

   ```
   GET [equella.url]/api/search?q=cats&start=0&length=10
   ```

3. The EquellaApiService.search method asks the JSON parser to turn the JSON response into a JSON ObjectNode for easy processing of the results. This is returned to the SearchServlet.
4. SearchServlet populates the display model with data extracted from the JSON object.
5. The search.ftl template is rendered to the response by the template rendering engine using the display model object to display dynamic data.

View

Clicking on a search result will direct the user to the view page. The view page displays basic information about a resource, as well as the thumbnails and links to the attachments.

Viewing a resource will perform the following steps in the client

1. ViewResourceServlet does some simple query string parameter validation and passes the parameters to the EquellaApiService.getResource method.
2. The EquellaApiService.getResource method retrieves a resource from the EQUELLA server using the /api/item endpoint i.e.

   ```
   GET [equella.url]/api/item/[uuid]/[version]?info=all
   ```

3. The EquellaApiService.getResource method asks the JSON parser to turn the JSON response into a JSON ObjectNode for easy processing. This node is returned to ViewResourceServlet.
4. ViewResourceServlet takes the JSON object and populates the display model object.

5. The view.ftl template is rendered to the response by the template rendering engine using the display model object to display dynamic data.

Contribute

To /edit page allows the user to contribute new resources to EQUELLA and attach files to the resource. This relies on the collection.uuid configuration property to determine which collection to contribute to.

When the user clicks "Save" the following steps are performed by the client

1. EditResourceServlet retrieves all form data, including uploaded files, and passes them to the EquellaApiService.saveResource method.
2. The saveResource method sets the name and description of the resource in the appropriate XML nodes in the metadata. This relies on the name.xpath and description.xpath in the configuration properties. The name and description cannot changed by modifying the name and description JSON nodes, these are provided for convenient retrieval only.
3. The saveResource method invokes the uploadFiles method which:
   a. Creates a temporary file area by POSTing to the EQUELLA REST API endpoint /api/file
   b. Retrieve the upload location for this new file area
   c. Upload each file by PUT-ting to the file content URL. i.e.

      ```
      PUT [equella.url]/api/file/[file_area_uuid]/content/[filename]
      ```

      Note that this is a PUT for content creation, not POST, since the location is user defined (via the filename).

      For each file that is uploaded an attachment node is added to the resource JSON within the uploadFiles method.
4. The saveResource method POSTs to the /api/item endpoint (since the resource UUID is blank), including the file area UUID as a parameter if any files were uploaded. i.e.

   ```
   POST [equella.url]/api/item?file=[file_area_uuid]
   ```
5. The saveResource method returns the API location of the new resource
6. The EditResourceServlet obtains the UUID and version of the new resource by invoking EquellaApiService.getResourceByLocation
7. The EditResourceServlet redirects to the /view page of the client to view the new resource

Contact Client Support

We are always happy to help.

If your organisation has a support agreement with EQUELLA then help is available at http://equella.custhelp.com.