JavaScript

Table of Contents:

Overview

The OpenCGA JavaScript client library has been implemented using the ES6 standard features such as classes, template strings or promises. All the methods from the API calling to the web services return *promises* to make easier to implement async callbacks.

Library design

The OpenCGA client library was easily designed to ensure that most of the functions one line long and it is implemented in a single file at https://github.com/opencb/jsorolla/blob/develop/src/core/clients/opencga-client.js. The library can be divided into two main parts, the *configuration* and the actual *clients.* The ***OpenCGAClientConfig*** configuration class is a small class containing some fields to know mainly, where the REST host can be found. Additionally, the developer can choose whether to let the OpenCGA client to use cookies to store the main credentials allowing to easily refresh the page without loosing the authentication credentials and set a cookie prefix.

Create configuration object

```
// Initialise the OpenCGA config
let myConfig = new OpenCGAClientConfig("localhost:8080/opencga");
```

The second part of the file is composed by the actual ***OpenCGAClient.*** A similar approach to Java client is followed to implement the JavaScript clients. We have implemented an *OpenCGAClient* class that acts as a client factory. There exist one class for each available resource *(user, project, study, ...)* that can be invoked through the *OpenCGAClient* instance as shown in the example.

Clients

```
// Initialise the OpenCGAClient
let openCGAClient = new OpenCGAClient(myConfig);


// Get an instance of all the possible resources
let userClient = openCGAClient.users();
let projectClient = openCGAClient.projects();
let studyClient = openCGAClient.studies();
let fileClient = openCGAClient.files();
let jobClient = openCGAClient.jobs();
let sampleClient = openCGAClient.samples();
let individualClient = openCGAClient.individuals();
let familyClient = openCGAClient.families();
let cohortClient = openCGAClient.cohorts();
let clinicalAnalysisClient = openCGAClient.clinical();
let variableSetClient = openCGAClient.variables();
let alignmentClient = openCGAClient.alignments();
let variantClient = openCGAClient.variants();
```

Every method belonging to each resource takes the mandatory parameters individually. Calls to web services using the GET method usually takes an additional *params* field containing any of the additional optional parameters. Calls via POST takes an additional *body* field containing the required information to be sent in the body of the message apart from the *params* field. Both *params* and *body* should be JSON objects containing the previously described information. The list of accepted parameters and fields should be checked in the Swagger documentation.

For example, if we wanted to get the information of two samples excluding the *attributes* and *stats* fields, we could do it as follows:

Get info from samples

```
sampleClient.info("sample1,sample2", {"exclude": "attributes,stats"})
.then(function(response) {
  // Success
  console.log(response);
}).catch(function(e) {
  // Error
  console.log(e);
});
```

How to use the JavaScript client

In order to use the Javascript client, developers will only need to include two files which can be found in https://github.com/opencb/jsorolla/blob/develop/src/core/clients. Additionally, OpenCGA client depends on two external libraries, crypto-js (tested with version 3.1.8) and cookies-js (tested with version 1.1.0).

In the future, it will be possible to add this client as an *npm* dependency (https://www.npmjs.com/~opencb).

```
<!-- OpenCGA dependencies -->
<script type="text/javascript" src="../node_modules/crypto-js/crypto-js.js"
></script>
<script type="text/javascript" src="../node_modules/cookies-js/src/cookies.
js"></script>
<!-- !OpenCGA dependencies -->

<!-- OpenCGA client -->
<script type="text/javascript" src="rest-client.js"></script>
<script type="text/javascript" src="opencga-client.js"></script>
<!-- !OpenCGA client -->
```