OpenTox REST ontology and implementation of OPTIONS
(To be included in API 1.2)

Pantelis Sopasakis

One of the prime attributes of the REST architecture is that is decouples clients from servers since the latter should provide a uniform interface which oughts to be transparent and well documented. To this end, servers have to provide (machine readable) guidelines to the clients regarding all details about the service invocations. Such information include supported media types, expected status codes, possible input parameters (either as POSTed within a form, included in the web address as URL-parameters or contained in the Header of the request), information about the implemented methods and much more; always followed by proper (human readable) annotations.

The establishment of a REST ontology for OpenTox is expected to:

1. Enhance the server-client communication providing information to the client about possible exceptions that might occur.
2. Makes possible the automated creation of clients for OpenTox web services (The user provides an OpenTox URL and a web interface appears, providing all REST-related information about the service plus a web form for it's consumption)
3. Human readable documentation about the service can be retrieved and made available in HTML, PDF or other user friendly formats.

The defining ontology can be found at the OpeTox collaborative Protege and its latest version is attached. If you use Java for your web services, you can consider using ToxOtis (version 0.4.12.2 or later) for the creation of REST specifications for your services. Here is an example:

```
ServiceRestDocumentation doc = new ServiceRestDocumentation(new Task(new VRI("http://myserver.com/task/1")));
RestOperation delete = new RestOperation();
delete.addHttpStatusCodes(
 new HttpStatus(OTRestClasses.STATUS_200()).setMeta(new MetaInfoImpl().addTitle("Success").
  addDescription("The task was successfully deleted from the database").
  addComment("The status code 200 is returned in any case of successful deletion but excluding the "
   + "case where the underlying task was not found in the database")),
 new HttpStatus(OTRestClasses.STATUS_404()).setMeta(new MetaInfoImpl().addTitle("Not Found").
  addDescription("The task was not found in the database")),
 new HttpStatus(OTRestClasses.STATUS_403()),
 new HttpStatus(OTRestClasses.STATUS_401()));
delete.setMethod(MethodsEnum.DELETE);
delete.addOntologicalClasses(
        OTRestClasses.DELETE_Task(), OTRestClasses.OperationTask(), OTRestClasses.OperationNoResult());
delete.getMeta().addDescription("Cancels a running task.");
delete.setProtectedResource(false);

RestOperation get = new RestOperation();
get.addHttpStatusCodes(
 new HttpStatus(OTRestClasses.STATUS_200()).setMeta(new MetaInfoImpl().addTitle("Success").
  addDescription("The task is successfully retrieved from the database and has completed redirecting to the result "
   + "of the calculation")),
 new HttpStatus(OTRestClasses.STATUS_202()).setMeta(new MetaInfoImpl().addTitle("Success").
  addDescription("The task is successfully retrieved from the database and is still running/processing")),
 new HttpStatus(OTRestClasses.STATUS_201()).setMeta(new MetaInfoImpl().addTitle("Success").
  addDescription("The task is successfully retrieved from the database has completed its own part of"
   + "the overall work but redirect to some other task on some other server")),
 new HttpStatus(OTRestClasses.STATUS_404()).setMeta(new MetaInfoImpl().addTitle("Not Found").
  addDescription("The task was successfully deleted from the database")),
```

```
 new HttpStatus(OTRestClasses.STATUS_403()),
 new HttpStatus(OTRestClasses.STATUS_401()));
get.addUrlParameter("media", true, XSDDatatype.XSDstring);
get.addUrlParameter("method", true, XSDDatatype.XSDstring);

doc.addRestOperations(delete, get);
```

The output of the above source code is attached (file task.rdf). Another example is also attached regarding the resource http://<server>:<port>/model (file models.rdf) where one can for instance find that the optional URL-parameters 'max' and 'page' can be used to retrieve only a certain page of the results of specified size.

References

1. http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.2 [OPTIONS method]
2. http://opentox.org/dev/ontology/collaborative_protege [OpenTox Collaborative Protege]
3. https://github.com/alphaville/ToxOtis
   [ToxOtis implementation with support for the abovementioned ontology]
4. https://github.com/alphaville/jaqpot/blob/master/src/org/opentox/jaqpot/resources/TaskResource.java [Example source code of a web service with OPTIONS support]