Genevestigator Data Export API

Revision: 2.2

1 Introduction

1.1 Purpose and Scope

The Genevestigator Data Export API provides programmatic access to the Genevestigator data. This documents describes the structure of the Export API and some examples and tips for developing with the Export API. The details on individual classes and methods are available from the HTML JavaDoc (see below). It is assumed that the reader is familiar with Java programming.

1.2 Features

The Export API and its interfaces and transfer objects provides the following features:

- Login/Logout functionality to the Genevestigator server.
- Requesting the Expression Matrix.
- Requesting the Anatomy Annotations.
- Requesting the Cell type Annotations.
- Requesting the Cell line Annotations.
- Requesting the Neoplasm Annotations.
- Requesting the Development Annotations.
- Requesting the Stimulus Annotations.
- Requesting the Mutation Annotations.
- Requesting the Property Annotations.
- Requesting the Application Area Annotations.
- Requesting the Global Study Type Annotations.
- Requesting the Study Design Annotations.
- Requesting the Experiment Information.
- Requesting the Experiment-Replica-Chip-Relations.
- Requesting the Platforms Information.
- Requesting the Ortholog Information.
- Computing Differential Expression.

2 Development Environment

2.1 Libraries

The file DataExportTool_V2.tar.gz contains the required libraries to use the Export API. It also includes an implementation, sample code and the source of a command line tools using the interface to generate export files for different annotations, experiments and platforms as specified in the Genevestigator Export Format document [1].

The file content is structured as follows:

File	Description
exporttool.sh	Script to export a single file of a platform and experiments.
experimentexport.sh	Script to export experiment data in tabular form (e.g. GEO)
computetool.sh	Script to compute meta-profiles for anatomy, development, cell line, neoplasm, perturbations on stimulus, mutations or perform operations on expression matrices.
dataexport.sh	Script to export all files of a platform.
export_wrapper.R	R-Wrapper on the data export scripts to be called from an R-Script
README	Latest release information and short description of script-usage.
bin\	Directory containing helper scripts for the export and calculation script.
bin\setenv.sh	The environment settings for shell scripts.
bin\export_variables.sh	Subscript of export.sh and calcaverages.sh.
bin\export.sh	Subscript of dataexport.sh, exports all annotations and expression matrices.
bin\calcaverages.sh	Subscript of dataexport.sh, computes all meta-profiles.
bin\utilities.sh	Auxiliary functions.
doc\ExportFormats.pdf	Definition of the export formats.
doc\ExportAPI.pdf	User documentation of the Data Export API.
doc\ExportTools.pdf	Description of the export tool scripts.
logs\	Empty directory containing the log files generated by the export and calculation script.
lib\DataExportTool_V2.jar	The jar file of the export tool.
lib\DataExportAPI_V2.jar	The jar file containing the Data Export API
lib\Algorithms_V1.jar	The jar file containing required classes for DataExportAPI_V1.jar
lib\Client_V4.jar	The jar file containing required classes for DataExportAPI_V1.jar
lib\Client_V4_Server_V4.jar	The jar file containing commonly used services
lib\ClientServerCommunication_V4.jar	The jar file containing required classes for DataExportAPI_V1.jar
lib\Domain_V1.jar	The jar file containing required classes for DataExportAPI_V1.jar
lib\GlobalUtilities_V3.jar	The jar file containing required classes for Client_V4_Server_V4.jar
lib\logback*.jar, slf4j*.jar	3rd party. Logback logging libraries used by the DataExportTool_V1.jar
lib\commons*.jar	3rd party. Math and CLI libraries used by the DataExportTool_V1.jar
lib\opencsv*.jar	3rd party. Library to read/write CSV files.
lib\trove*.jar	3rd party. Library for fast collections.
src\	Directory containing the java source of DataExportTool_V1.jar
javadoc\	JavaDoc for DataExportAPI_V1.jar and DataExportTool_V1.jar

Table 1: Content of DataExportTool_V2.tar.gz

2.2 Classpath

The minimum required JRE-Version to develop applications and directly interact with the Data Export API is 11. The following jar libraries are required on the classpath (All contained in the lib folder of the file DataExportTool_V2.tar.gz):

- DataExportAPI_V2.jar
- DataExportTool_V2.jar
- Algorithms_V1.jar
- Client_V4.jar
- Client_V4_Server_V4.jar
- ClientServerCommunication_V4.jar
- Domain_V1.jar
- GlobalUtilities_V3.jar
- All 3rd party library

2.3 User Account

A user account with export permission (Permission caption "T_API_DA") and data subscriptions is needed to export data using the API. A user named "export" is configured in the local installation by default. This user has the authorization to access all data using the Export API.

3 API

3.1 General Structure

The API functionality consists of three main categories:

- Services that access the Genevestigator server to retrieve data. These results may also be cached locally. These services are available from the class dea.client.ServiceClient.
- Domain objects which are retrieved by the services and represent the basic items of information such as an experiment, an annotation factor or a platform. These objects are implemented by the classes in the packages domain.experiment, domain.factor, domain.organism and domain.mapping.
- Utilities for filtering or transformation of data. These methods are implemented in the classes in the dea.util package.

The class **dea.client.ServiceClient** of the library **DataExportAPI_V2.jar** implements the export functionality. It provides login/logout functionality and methods to get the individual services. The different services provides the methods to query organisms, experiments, ontology trees, the experiment-replica-chip relationship, methods to export expression data of chips/measures combinations for different platforms, methods to get orthologous information or perform a differential expression computation. For details on the method signatures on the provided services consult the JavaDoc.

The class diagram 1 shows the classes and relationships among the classes used by the different services. We can identify three groups of data objects. The first group contains the information on organism, platform, technology and measure. The second group contains information about experiment structure with classes for repository, experiment, replica and chip. The third group contains information on the experiment-level and sample resp. replica-level annotation. The class Comparison represents a comparison between a treatment and control set. It is realized as two sets of replica each with its annotated ontology category (class Factor). The ontology tree is defined by the class Factor which

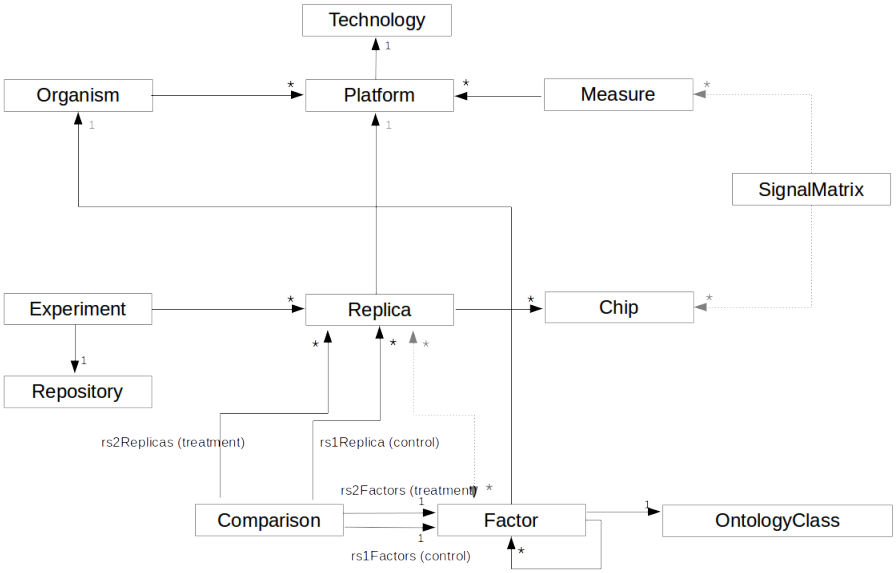

Figure 1: Classes provided by the export API

has a parent-child relationship to itself. Each Factor belongs to an OntologyClass (Anatomy, Cell line, Neoplasm, Development, Stimulus, Mutation, Property) and an organism.

Note that the UML diagram in Figure 1 contains two types of connections: solid black lines designate relationships that are implemented as references in the class while the dotted grey lines designate relationships that need to be retrieved from the services or via utilities. For example: The relationship between the replicas and the annotated factors is retrieved by the method getReplicasByFactors() on the FactorServiceInterface which returns a map from factors to the replicas with this annotation.

3.2 Session Handling

To work with the services a user must be authenticated using the class ServiceClient to get a valid session. It is recommended to always logout if an access is no longer required otherwise an inactive session will be invalidated after a timeout and subsequent requests will be denied and a re-login is required to use the service.

3.3 Example

The following code snippet shows how to authenticate the credentials and checking on authentication success using the class the dea.client.ServiceClient. The specified service URL "https://<hostname>/Server_V4/dispatch/" points to the Genevestigator server. Consult the system administrator for the service URL of your installation. After a successful login all experiments of a mouse platform are requested and the annotations of each replica and experiment are printed to the standard output.

```
ServiceClient serviceClient = null;
try {
    serviceClient = new ServiceClient(UrlUtils.getUrl(SERVICE_URL));
    if (serviceClient.login(USERNAME, PASSWORD)) {
        // serviceClient created and logged in

        // query Affymetrix Murine Genome U74A Version 2 Array
        Platform platform = serviceClient.getBaseDataService().getPlatformByShortCaption("MM_AFFY_U74A_V2");

        // query experiments of mouse platform
        List<Experiment> experiments = serviceClient.getExperimentService().getExperiments(platform);
```

```
        // query annotation on experiment- and replica-level
        FactorServiceInterface factorService = serviceClient.getFactorService();
        Map<Experiment, List<Factor>> factorsByExperiment = factorService.getFactorsByExperiment(experiments);

        // collect replicas of all experiments
        List<Replica> replicas = ExperimentUtils.getReplicasForExperiments(experiments);
        Map<Replica, List<Factor>> factorsByReplica = factorService.getFactorsByReplica(replicas);

        // print annotations of each replica
        for(Experiment experiment: experiments) {
            System.out.println("Experiment: "+experiment.getNbr());
            System.out.print(  "   ");
        for(Factor factor: factorsByExperiment.get(experiment)) {
            System.out.print(factor.getOntology().getClass_().toString());
            System.out.print(": "+factor.getCaption());
            System.out.print(", ");
        }
        System.out.println();
            for(Replica replica: experiment.getReplicas()) {
                System.out.println("  Replica:  "+replica.getCaption());
                System.out.print(  "   ");
                for(Factor factor: factorsByReplica.get(replica)) {
                    System.out.print(factor.getOntology().getClass_().toString());
                    System.out.print(": "+factor.getCaption());
                    System.out.print(", ");
                }
                System.out.println();
            }
        }
    } else {
        System.err.println("Login not successful");
    }
} catch(InitializationException e) {
    System.err.println("InitializationException on ServiceClient construction "+e);
} catch(CommunicationException e) {
    System.err.println("CommunicationException on ServiceClient construction "+e);
} finally {
    // logout
    if (serviceClient != null) {
        try {
            serviceClient.logout();
        } catch(Exception e) {
            System.err.println("Exception on logout "+e);
        }
    }
}
```

4 Utilities

The services of the API mainly returns lists of the classes Organism, Experiment, Measure, Factor. To support filtering and searching on these lists the package dea.util contains utility classes. These classes follow the naming convention <class>Utils. Relations between factors and annotated replicas or annotated comparisons are returned as a map by the services, to filter these maps the utility class RelationUtils is provided. To compute Meta-Profiles for annotations two utility classes AbsoluteProfileStatisticsUtils and RelativeProfileStatisticsUtils are provided in the package det.util.

5 Input/Output

The package det.io of the DataExportTool contains Input/Output-classes to write and read objects of classes Platform, Experiment SignalMatrix and annotations to and from a file. The name of the I/O-classes follow the convention <class>IO, each I/O-class supports at least the methods readFromFile() and writeToFile().

Additionally the package dea.service.task of the API contains implementation of the interface DataTask to handle expression data in an asynchronous design from the DataService. This avoids the allocation for a complete possibly large expression matrix in memory.

References

[1] N. AG. Genevestigator data export formats, 2009. https://www.genevestigator.com.