ToxOtis

A Java Interface to the OpenTox Predictive Toxicology Network

Pantelis Sopasakis[a], Haralambos Sarimveis[b]

[a] IMT Alti Studi Lucca, Piazza San Ponziano, 6, 55100, Lucca, Italy. E-mail: p.sopasakis@imtlucca.it
[b] National Technical University of Athens, School of Chemical Engineering, Heroon Politechniou, 9, 15780, Zografou Campus, Athens, Email: hsarimv@central.ntua.gr

ToxOtis-core

ToxOtis-DB

Powered by: MySQL, semantic web framework Jena, Java, XML, Apache maven, WEKA The University of Waikato, SLF4J, Restlet, GPLv3 Free Software Free as in Freedom

OpenTox

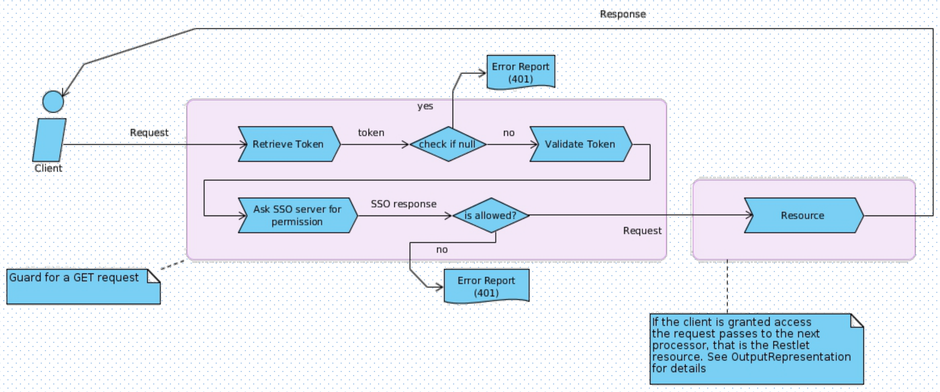
About

OpenTox [1] is a distributed computational network designed based on the principles of the REpresentational State Transfer (REST). ToxOtis offers elaborate search facilities to both online and local databases of chemical compounds and properties. It allows users to :

- Download and upload data to OpenTox web services
- Train predictive models (both regression and classification)
- Analyze data (e.g. select the most descriptive molecular descriptors for a given target endpoint),
- Evaluate the Domain of Applicability (DoA) of a (Q)SAR model,
- Derive toxicological predictions and, last but not least,
- Specify access restrictions on the resources they create on the network.

Asynchronous computational tasks initiated on the OpenTox network are mapped to lightweight Java threads client-side which monitor the progress of the remote execution.

Authentication and Authorisation



ToxOtis offers easy and transparent access to the A&A API of OpenTox [2]. Users providing their credentials receive an authentication token which they then pass to the various services.

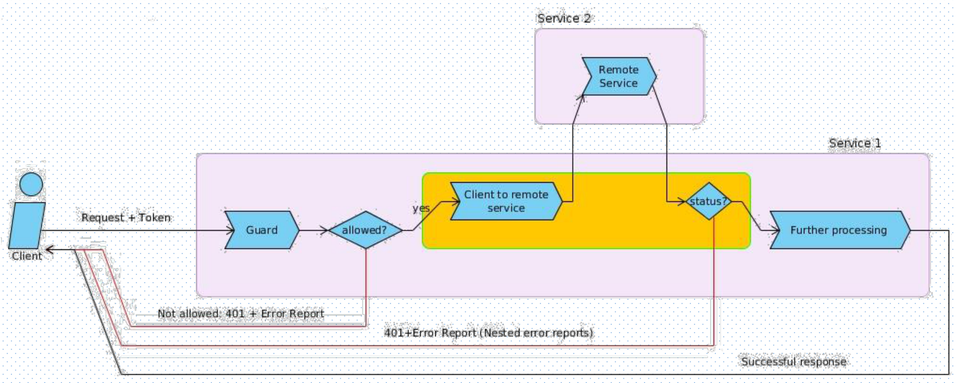
```
AuthenticationToken at = new AuthenticationToken("uname","passwd");
```

It is however safer to use encrypted password files instead of hard-coding passwords:

```
File passwordFile = new File("/path/to/my_sercret.key");
AuthenticationToken at = new AuthenticationToken(passwordFile);
```

Applications with multiple users can benefit from the `TokenPool` of ToxOtis:

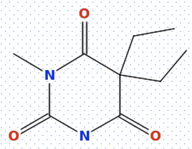
```
TokenPool tokenPool = TokenPool.getInstance();
tokenPool.login("/path/to/jsmith.key");// token for user 'john_smith'
tokenPool.login("/path/to/jsmith.key");// does nothing
tokenPool.login("/path/to/alice.key");// token for user 'alice'
AuthenticationToken at = tokenPool.getToken("john_smith");
tokenPool.logoutAll();// invalidates all tokens
```



Chemical Compounds & Conformers

A chemical conformer (class `Conformer`) is an identifier of a unique chemical substance up to its 3D characteristics. The class `Compound` is used in ToxOtis to describe all sorts of chemical compounds and map them to their properties. Users can easily create a compound or a conformer and assign an identifier to it that is a URI. If this URI corresponds to a valid URL of an OpenTox service, one can download meta-information about the compound or a representation thereof in some desired format as in the following example where the compound with identifier apps.ideaconsult.net:8080/ambit2/compound/10 (CAS-RN: 50-11-3) is stored to a file in MOL format:

```
// Download Compound from a remote OpenTox web service using ToxOtis:
Compound comp =
 new Compound(new VRI(Services.IDEACONSULT).augment("compound","10"));
File destination = new File("/path/to/file.mol");
comp.download(destination, Media.CHEMICAL_MDLMOL, at);
```



Supported Representation of Chemical compounds include:

- SMILES
- SDF
- CML (Chemical Markup Language)
- InChi and InChIKey
- Various Image Formats (PNG, JPEG, PDF)
- Resource Description Framework Representaitons (RDF/XML, RDF/TTL, etc)

Algorithms

An OpenTox Algorithm is characterized by a set of Ontological Classes that classify it according to the OpenTox Algorithms Ontology [3]. As algorithms can be used for a wide variety of purposes (e.g. (Q)SAR/(Q)SPR model building, feature calculation, feature selection, similarity calculation, substructure matching), required and optional input parameters and algorithm results have to be specified in the algorithm representation along with a definition/description of the algorithm. A set of parameters along with their *scope* (optional/mandatory) and *default values* are also available for every algorithm.

Features

In OpenTox, a Feature is an object representing any kind of property assigned to a Compound. The feature types are determined by links to an appropriate ontological class (Feature and Descriptor Ontologies, Endpoint Ontologies). According to the OpenTox API, clients may have access to the metadata of a Feature in RDF or other relevant format and can create their own Features and assign them desired ontological properties. Using ToxOtis, one may lookup on an OpenTox web service for features of some given ontological class; here is an example of use:

```
// Find Features:
AuthenticationToken at = tokenPool.getToken("john_smith");
Set<VRI> features = FeatureFactory.lookupSameAs(OTEchaEndpoints.DissociationConstantPKa(),at);
for (VRI f : features) { /* Do something */ }
```

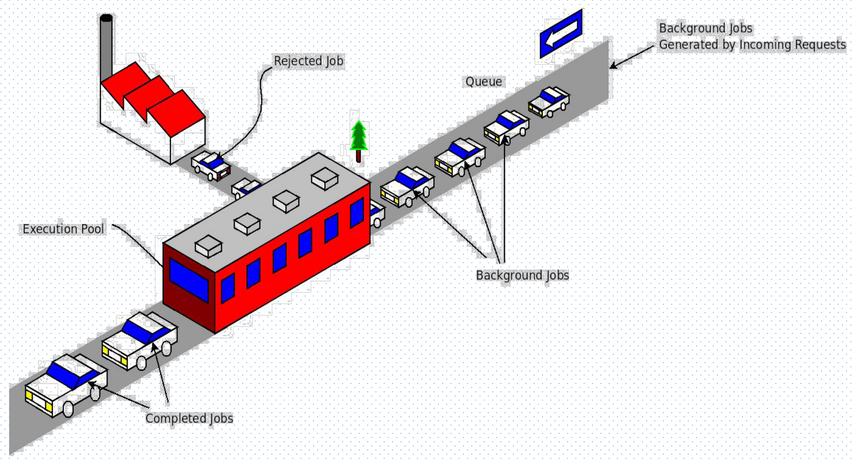
Features are formally defined by ontologies.

Clients can create and publish new features on the OpenTox network:

```
Model m = /* A predictive model */ ;
Feature predFtr=FeatureFactory.createAndPublishFeature("Title", new ResourceValue(m.getUri(),
OTClasses.Model()), featuresServer, at);
```

Asynchronous Execution on the Web

In OpenTox various computational procedures such as model training, data filtering or even data upload operations can be time consuming. While such operations take place, the socket that binds the client to the server has to be released. Such a connection is prone to network errors and additionally it would be quite burdensome for both peers to keep lots of connections open until each operation completes. In order to tackle this problem three structures were adopted by the services: *background jobs*, *queues* and *execution pools*. A client request initializes a *background job* which is put in an *execution queue* (In the meanwhile other jobs may run in the background). A task is then created and returned to the user in a supported MIME such as `application/rdf+xml`.




```
VRI algUri = new VRI("http://server.org/alorithm/mlr");
VRI dsUri = new VRI("http://server.org/dataset/100");
VRI featUri = new VRI("http://server.org/feature/5");
Algorithm alg = new Algorithm(algUri);
Dataset ds = new Dataset(dsUri);
Feature f = new Feature(featUri);
Trainer trainer = new Trainer(alg, ds, f);
Task task = f.train(algUri);
Callable<Task> runner = new TaskRunner(task);
```

Datasets and Weka

Taking into account the popularity of Weka [4] in Machine Learning and its widespread use in predictive toxicology, we implemented converters from and to the Weka Instances objects and ARFF files to `Dataset` objects which are well integrated in the OpenTox framework.

```
// Convert weka.core.Instances to Dataset objects:
Instance myInstances = ...; // The Weka Object
Dataset d = DatasetFactory.createFromArff(myInstances);
```

Datasets are bundles of Conformers along with some of their their Features and values assigned to Conformer-Feature pairs. The elementary blocks of a Dataset are its entries; instances of the class `DataEntry`. Every DataEntry object points to a chemical conformer and associates it with a set of `Feature-Value` pairs. Datasets can be downloaded from remote locations in a straightforward way:

```
// Download a Dataset given its URI:
VRI vri = ...;
// Require that the dataset will contain no more than 10 compounds
vri.addUrlParameter("max", 10);
Dataset ds = new Dataset(vri);
ds.loadFromRemote(at);
```

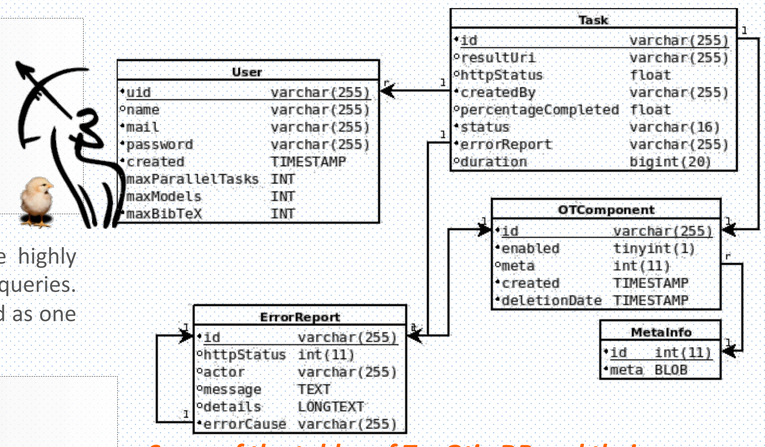
ToxOtis-DB

ToxOtis-DB is a module that can be used for the management of an SQL relational database for OpenTox models, datasets, features, tasks, users, models and BibTeX objects [5]. The following example illustrates how easily one can perform advanced searches in the database:

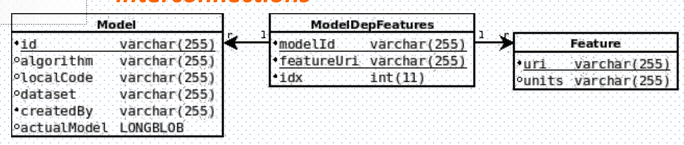
```
// Find a user with ID 'john':
FindUser finder = new FindUser();
finder.setWhere("uid='john'");
User user = null;
IdbIterator<User> iterator = finder.list();
if (iterator.hasNext()) { user = iterator.next(); }
else { /* No such user in the database */ }
```

The database engine and the database scheme of ToxOtis-DB are highly optimised to offer high performance and ability to perform complex queries. The registration of objects into the database is rather straightforward as one can judge by the following snippet:

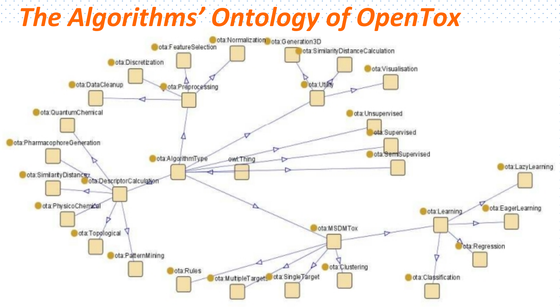
```
Task t = new Task(Services.ntua().augment("task",
   UUID.randomUUID()));
DbWriter writer = new AddTask(t);
writer.write();
writer.close();
```



Some of the tables of ToxOtis-DB and their interconnections



The Algorithms' Ontology of OpenTox



References

1. Hardy, B. *et al.* 2010. Collaborative development of predictive toxicology applications. *Journal of Cheminformatics*, **2**, 1 – 29.
2. The OpenTox API ver. 1.2 – Technical Specifications: Available online at http://opentox.org/dev/apis/api-1.2 (accessed on 12 June, 2013).
3. Tcheremenskaia, O., Benigni, R., Nikolova, I., Jeliazkova, N., Escher, S. E., Batke, M., Baier, T., Poroikov, V., Lagunin, A., Rautenberg, M., Hardy, B. 2012. OpenTox predictive toxicology framework: toxicological ontology and semantic media wiki-based OpenToxipedia, *Journal of Biomedical Semantics*, 3.
4. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann P., Witten, I. H., 2009, The Weka Data Mining Software: An update, *SIGKDD Explorations*, **11**(1).
5. Sopasakis, P., Chomenides H. 2011, http://opentox.ntua.gr/wiki/ToxOtis-db (accessed on 12 June, 2013).

** Toxotis (Τοξότης) in Greek means Sagittarious of archer. ToxOtis is an open-source project hosted on github (see http://github.com/alphaville/ToxOtis).*
*** ToxOtis was developed in Java using the Apache Maven integration platform and is hosted as a Maven artifact on our Nexus repository at http://opentox.ntua.gr:8081/nexus from where one may download it. Detailed documentation is available on our wiki page at http://opentox.ntua.gr/wiki/*