OBA: ontology based answers
Manual

Content

Introduction

The OBA project provides access to hosted ontologies and functions to answer ontological queries based on the structure of the ontologies. The OBA service is the implementation of the RESTful network interface in this project. This interface provides the data in different MIME types and is preferable used by other applications and workflows. In this manual the syntax of the queries and the format of the responses are documented.

Availability

The homepage of the project is http://bioinf.med.uni-goettingen.de/projects/oba. On this page further information is available as well as an example Java connector and news around the service. The service itself is accessible at the address http://oba.sybig.de. This URL should be used by any client or with a web browser for basic testing.

The front page of the server provides an overview page in html format. There is a list of currently loaded ontologies and available functions with their signature. This page is only available in html format, see also below for the MIME types supported by the OBA service.

Usage and installation

The client can be downloaded from the home page of the project. The zip file “oba-client.zip” contains the basic oba client (oba-client-1.2.1.jar) and a client for use with the Cytomer ontology (oba-cytomer-1.2.1.jar) which includes the basic oba client. To use one of the connectors in an application, add the corresponding jar file to the classpath. To create an instance of the connector use:

```
GenericConnector gc = new GenericConnector(ontologyName);
```

or

```
CytomerConnector cc = new CytomerConnector();
```

The description of implemented methods are available in JavaDoc. By default the clients use the public OBA server, no configuration is needed.

Both clients are in two versions in the zip file, with and without all dependencies packed in the jar file. For both clients there is also a jar file containing the sources in the zip file.

If the Cytomer client is started on the command line (`java -jar oba-cytomer-with-dependencies-1.2.1.jar`) a simple test client is started. This client calls several functions on the server and print the results to the console.

Server installation

The installation of an oba sever is only required, if the public server can not be used, i.e. if custom ontologies or plugins should be used. To use a local server the file “oba-server.zip” has to be downloaded from the project page and unzipped. The server is started with the command “`java -jar oba-server-1.2.1.jar oba.properties`. The server will run without a graphical interview and is available at `http://localhost:9998` after startup. Bundled with the server the Cytomer ontology and the Tribolium ontology is shipped together with corresponding plugins in the directory “ontologies” and “plugins” respectively. If required the paths to the plugin and the ontology directory can be adjusted in the property file `oba.properties`.

Required dependencies

- slf4j-api
- owl-api
- jackson-mapper-asl
- jackson-core-asl
- grizzly-servlet-webserver
- jersey-json
- jersey-server
- lucene-core
- jsr311-api
- lucene:lucene-core
- hsqldb

MIME-types

The OBA service provides data in the MIME types “application/json”, “text/plain” and “text/html”. With exception of “text/html” these MIME types are also accepted for uploaded data.

MIME type: text/plain

In general the content of the ontology classes are represented as key value pairs separated by a single tab character. A key may occur more than once in a document for example if an ontology class has multiple “hasPart” relations, each relation is represented by a line starting with the key “hasPart”

Ontology class

Key	Description
name	The name of the ontology class.
namespace	The namespace of the ontology class.
parent	One line for each parent. If the namespace of the parent is equal to the default namespace the namespace is omitted.
child	One line for each child. If the namespace of the child is equal to the default namespace the namespace is omitted.
name of the annotation field	For the annotation fields the name of the annotation field is used. The language of the annotation value is indicated inside of “[]”. If no language is defined, the brackets are empty.
name of the relation	Each relation is represented by a line, starting with the name of the relation as key. The value is the target class of the relation.

Table 1: Keys of an ontology class in the MIME-type "text/plain"

Below the class “digestive_organ” from the Cytomer ontology is given as example:

```
name    digestive_organ
namespace       http://protege.stanford.edu/plugins/owl/protege#
parent   organ
child    stomach
child    mouth
child    intestine
child    liver
child    oesophagus
child    gallbladder
child    pancreas
abstract []     true
ACC []  cy0044711
```

List of ontology classes

If the result of a request is a list of ontology classes, each ontology class of the result set is printed on one line. The class is specified by its name preceded by its namespace.

The first lines of the result of the search for “liver” is:

```
http://protege.stanford.edu/plugins/owl/protege#liver
http://cytomer.bioinf.med.uni-goettingen.de/organ#parenchyma_of_liver
http://cytomer.bioinf.med.uni-goettingen.de/organ#acinus_of_liver
http://cytomer.bioinf.med.uni-goettingen.de/organ#hilus_of_liver
http://cytomer.bioinf.med.uni-goettingen.de/organ#parenchyma_of_right_lobe_of_liver
...
```

Two dimensional list of ontology classes

If the result is a list of list (two dimensional list) in each row a list of ontology classes is printed. The classes of the inner list are printed in a single row and are separated by tab characters. The name of each ontology class is printed together with its namespace.

MIME type: application/json

Please refer to the Java classes in the oba-common module for a definition of the JSON format used in the OBA project.

URL pattern

The OBA service hosts multiple ontologies and is divided into three main parts. The first part of the URI is composed by the ontology to use and the main part. The next three sections describe the parts in more detail.

Each ontology class is identified by a name and a namespace. The name can be used as path segment, the namespace should be given as the parameter 'ns' to this segment. If the class name is the last part of the URI the namespace is the option parameter, for example: `http://.../classname?ns=http://namespace.org`. If the segment of class is some where in the middle of the URI, the namespace has to be specified as matrix parameter. For the use as matrix parameter the '/' character has to be replaced by '$'. An example for a class with the namespace as matrix parameter would be:

`http://.../`**`classname;ns=http:$$namespace.org`**`/another/path/segment`

If no namespace is specified, the default namespace of the ontology is assumed. In the following the namespace is omitted for better readable URIs.

Accessing entities of an ontology

Commands:

Getting the root class of an ontology

```
GET  http://oba.sybig.de/{ontology}/cls/
```

Getting a ontology class

```
GET  http://oba.sybig.de/{ontology}/cls/{clsName}
```

Getting the children of an ontology class

```
GET  http://oba.sybig.de/{ontology}/cls/{clsName}/children
```

Getting the parents of an ontology class

```
GET  http://oba.sybig.de/{ontology}/cls/{clsName}/parents
```

Getting all object properties defined by an ontology

```
GET  http://oba.sybig.de/{ontology}/objectProperty
```

Getting a specific object properties of an ontology

```
GET  http://oba.sybig.de/{ontology}/objectProperty/{propertyName}
```

The placeholder should be replaced like the following:

{ontology}	The name of the ontology as listed on the front page of the service
{clsName}	The name of ontology class. The name space can be specified as parameter.
{propertyName}	The name of the object property, for example “isPartOf”

The storage handler

The OBA server can be used to deposit sets of ontology classes. The sets are stored in partitions, so that each user or work group can create their own partition. The name of the partitions and sets should only contain upper and lower case characters and digits. The length of the name for both should be between six and twelve characters. The amount of uploaded data is limited to 1000kb per set. The user is asked to remove sets which are not needed anymore. The sets are not removed on a regular basis, but the user should backup the data.

The following command will produce a unique ID to use as partition and creates the partition. The names of the partitions are not listed, to provide a basic access restriction.

Commands:

Create a new unique partition. The name of the new partition is always returned as text document containing only the name.

```
GET        http://oba.sybig.de/storage/uniqueID
```

List the content of a partition:

```
GET        http://oba.sybig.de/storage/{partition}/
```

Upload or download a set. If the partition does not exist it is created through the PUT command. Data can be uploaded as “text/plain” or “application/json”. For the format please refer to the sections describing the MIME types or simply reused the output from a previous search for example.

```
PUT|GET    http://oba.sybig.de/storage/{partition}/{set}
```

The substitutions are:

{partition}	The name of the partition, for example previous created with “uniqueID”
{set}	The name of the set with the stored ontology classes

Ontological functions

The section of the OBA service providing the functions is separated into two parts. One part contains generic functions, useful for all ontologies. The other part encompass functions for one specific ontology into one group.

Basic functions

Searching a class

On the server the name of the ontology classes and their annotation fields are indexed using Lucene[1]. These index fields are used by default in each search. With the optional matrix parameter “field” the search can be limited to a list of fields. If the field parameter is specified, the class name is not included by default, but can be explicitly added as “classname”. The annotations indexed for the loaded ontologies are listed in table 2. The search pattern can include the wild card characters “*” and “?”, where the second one has to be encoded with “%3f” in an URL.

The result of a search is a list of ontology classes matching the search pattern as whole word in one of the search fields. The list is ordered by relevance.

[1] http://lucene.apache.org/

Ontology	List of annotation fields.
Cytomer	NameEnglish, nameMedicine, nameGerman, definitionEnglish, ACC, definitionGerman
GO	label

Table 2: List of indexed annotation fields.

```
GET  htttp://oba.sybig.de/
{ontology}/functions/basic/searchCls[;field=]/{searchpattern}
```

Returns:

A list of ontology classes.

Substitutions:

{ontology}	The ontology to search the class in.
{searchpattern}	A (sub) string to search.
field	An optional list of annotation fields to use for the search.

Search the path to an ancestor

The function “XdownstreamOfY” checks if the class X is a direct or indirect successor of class Y. If this is the case the list of shortest paths between these two classes is returned. Otherwise an empty document is returned.

```
GET  http://host:port/{ontology}/functions/basic/XdownstreamOfY/
{x}/{y}
```

Returns:

A list of list of ontology classes.

Substitutions:

{ontology}	The ontology to use.
{x}, {y}	Two classes of the ontology.

HTTP status codes:

404	If either class X or class Y is not found in the ontology.

Map to abstract level

The function “reduceToLevel” maps an ontology class to its ancestors at the given level below of the root node. If the level of the input class is below or equal to the required level the input class is the only member of the list returned. While the function “reduceToLevel” returns the ancestors of all paths between the input class and the root node “reduceToLevelShortestPath” consider only the shortest paths between the input class and the root node.

```
GET  http://oba.sybig.de/{ontology}/functions/basic/reduceToLevel/
{level}/{cls}
```

```
GET  http://oba.sybig.de/
{ontology}/functions/basic/reduceToLevelShortestPath/{level}/{cls}
```

Returns:

A list of ontology classes.

Substitutions:

{ontology}	The ontology to use.
{level}	The level below of the root node of the returned ontology classes
{cls}	The ontology class which should be mapped to the lower level

HTTP status codes:

404	If the input class is not found in the ontology.

Cluster a set of ontology classes

The function “reduceToClusterSize” clusters a set of ontology classes to a set of more abstract classes. In each iteration the algorithm maps the classes with the greatest distance to the root node to its parents. The algorithm stops, if the number of cluster is smaller or equal the size specified as input. Beside of the number of clusters the input classes are specified as set stored before on the server. The result is a two dimensional list. On the first level is the ontology classes representing the cluster, on the second level is a list of ontology classes mapped to the class above.

```
GET http://oba.sybig de/
{ontology}/functions/basic/reduceToClusterSize/{size}/{partition}/
{name}
```

Returns:

A list of list of ontology classes.

Substitutions:

{ontology}	The ontology to use.
{size}	The maximal number of cluster to get
{partition}	The name of the partition
{name}	The name of the list with the ontology classes used as input.

Get domain classes of a restriction

The function “clsFroObjectProperty” returns all domain classes of an object property. With this function all classes using this object property can be found. If class “A” has the restriction partOf class “B” this function would return class “A” when called with the restriction partOf as parameter.

```
GET http://oba.sybig.de/
{ontology}/functions/basic/clsForObjectProperty/{restriction}
```

Returns:

A list of list of ontology classes.

Substitutions:

{ontology}	The ontology to use.
{restriction}	The object properties to get the domain classes for.

Cytomer functions

For the ontology about anatomical structures Cytomer a set of functions are provided.

Organ list

Get a list of all organs from the ontology. Organs are direct or indirect successor of the class "organ" which are not annotated to be an abstract group.

```
GET http://oba.sybig de/cytomer/function/cytomer/organList
```

Returns:

A list of ontology classes.

Organ of an anatomical structure

The function "organsOf" returns to an anatomical entity a list of organs this structure belongs to.

```
GET http://oba.sybig de/cytomer/function/cytomer/organsOf/{cls}
```

Returns:

A list of ontology classes.

Substitutions:

{cls} The ontology class representing the anatomical structure.

Physiological systems of an anatomical structure

The function "systemsOf" works analog to "organsOf" but returns a list of physiological systems for the anatomical structure.

```
GET http://oba.sybig de/cytomer/function/cytomer/systemsOf/{cls}
```

Returns:

A list of ontology classes.

Substitutions:

{cls} The ontology class representing the anatomical structure.

Search corresponding class in a set

The two functions "findUpstreamToList" and "findDownstreamToList" can be used to search a corresponding class in a set of ontology classes starting with an class unknown to this set. From the ontology class used as input as upstream or downstream search is started, until one or more classes of the set is found. The search follows the class hierarchy and along selected relations.

```
GET   http://oba.sybig.de/cytomer/functions/cytomer/findUpstreamInSet/{cls}/
{partition}/{set}
```

```
GET   http://oba.sybig.de/cytomer/functions/cytomer/findDownstreamInSet/{cls}/
{partition}/{set}
```

Returns:

A list of ontology classes.

Substitutions:

{cls} The ontology class from which the search is started

{partition}	The name of the partition
{set}	The name of the set with the reference set of classes