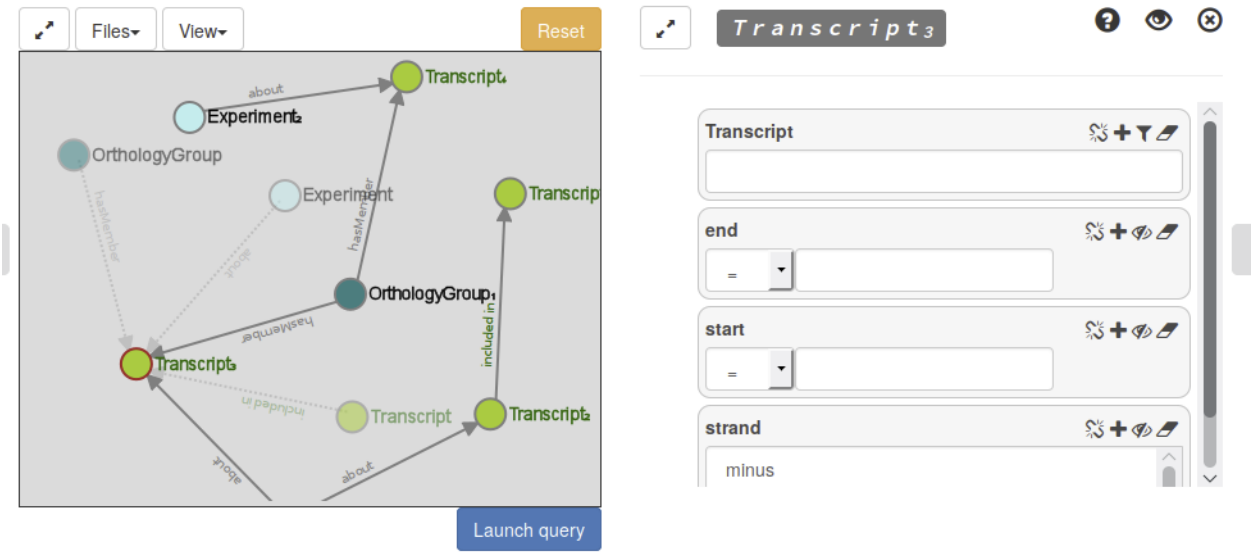
Python

Dec 20, 2018

Contents

AskOmics is a visual SPARQL query interface supporting both intuitive data integration and querying while shielding the user from most of the technical difficulties underlying RDF and SPARQL



Some (possibly outdated) documentation is also on the AskOmics Wiki. It should be imported here some day.

CHAPTER 1

Deployment

1.1 User

1.1.1 Dependencies

AskOmics need the Virtuoso triplestore to work.

Compile virtuoso

or install via docker

```
docker pull askomics/virtuoso

docker run --name my-virtuoso \
    -p 8890:8890 -p 1111:1111 \
    -e SPARQL_UPDATE=true \
    -v /tmp/virtuoso_data:/data \
    -d askomics/virtuoso
```

replace `/tmp/virtuoso` with a directory of your choice.

Your virtuoso is available at localhost:8890.

1.1.2 Manual installation

Dependencies

Installation needs some dependencies,

Ubuntu 18.04

```
sudo apt install -y python3 python3-venv python3-dev zlib1g-dev npm
```

Fedora 28

```
sudo dnf install -y gcc gcc-c++ redhat-rpm-config zlib-devel bzip2 python3-devel npm
```

Installation

Get the latest stable version of AskOmics : 18.10

```
wget https://github.com/askomics/askomics/archive/18.10.zip
unzip 18.10.zip
cd askomics
```

Or clone the repository

```
git clone https://github.com/askomics/askomics.git
cd askomics
git checkout 18.10
```

If you have installed virtuoso via docker, you have to inform AskOmics that the load url is not localhost:6543, but another ip address (dockers can't access host by http://localhost)

Run

```
docker exec my-virtuoso netstat -nr | grep '^0\.0\.0\.0' | awk '{print $2}'
```

and add

```
askomics.load_url=http://xxx.xx.x.x:6543
```

into `configs/production.virtuoso.ini` and `configs/development.virtuoso.ini` (replace `xxx.xx.x.x` with the ip obtained)

Install and run

```
./startAskomics.sh -d prod -t virtuoso
```

AskOmics is available at localhost:6543

1.1.3 Installation with docker

Pull the latest stable version of AskOmics

```
docker pull askomics/askomics:18.10
```

Run

```
docker run askomics/askomics -p 6543:6543
```

AskOmics is available at localhost:6543

1.1.4 Installation with docker-compose

Clone the askomics-docker-compose repository

```
git clone https://github.com/askomics/askomics-docker-compose
```

Choose which services you need and run with the docker-compose command. for example, if you need askomics+virtuoso :

```
cd askomics-docker-compose/virtuoso
docker-compose up
```

1.2 Developer

Fork the askomics repository

then, clone your fork

```
git clone https://github.com/USERNAME/askomics.git
```

Install AskOmics

Run it with dev mod

```
./startAskomics.sh -d dev -t virtuoso
```

AskOmics is available at localhost:6543

CHAPTER 2

Usage

TODO

CHAPTER 3

Contribute to AskOmics

3.1 Issues

If you have an idea for a feature to add or an approach for a bugfix, it is best to communicate with developers early. The most common venues for this are GitHub issues.

3.2 Pull requests

All changes to AskOmics should be made through pull requests to this repository.

For the askomics repository to your account. To keep your copy up to date, you need to frequently sync your fork:

```
git remote add upstream https://github.com/askomics/askomics
git fetch upstream
git checkout master
git merge upstream/master
```

Then, create a new branch for your new feature

```
git checkout -b my_new_feature
```

Commit and push your modification to your fork. If your changes modify code, please ensure that is conform to *AskOmics style*

Write tests for your changes, and make sure that they *passes*.

```
cd askomics
source venv/bin/activate
nosetests
```

Open a pull request against the master branch of askomics. The message of your pull request should describe your modifications (why and how).

The pull request should pass all the continuous integration tests which are automatically run by Github using Travis CI. The coverage must be at least remain the same (but it's better if it increases)

3.3 Tests

AskOmics use nosetests for Python tests. To run the tests, activate the Python virtual environment and run nosetests.

```
source venv/bin/activate
nosetests
```

Tests needs a Virtuoso instance and a Galaxy instance. You can use docker images.

To skip the Galaxy tests, run

```
nosetests -a '!galaxy'
```

To target a single file test

```
nosetests --tests askomics/test/askView_test.py
```

```
# Virtuoso
docker pull askomics/virtuoso
docker run -d -p :8890:8890 -e DBA_PASSWORD=dba -e SPARQL_UPDATE=true -e DEFAULT_
↪GRAPH=http://localhost:8890/DAV --net="host" -t tenforce/virtuoso
# Galaxy
docker pull bgruening/galaxy-stable
docker run -d -p 8080:80 bgruening/galaxy-stable
```

The testing configuration is set in the `askomics/config/test.virtuoso.ini` INI file. You can see that the Galaxy account API key is `admin`. The docker image `bgruening/galaxy-stable` have a default admin account with this API key. If you use another galaxy instance, change the url and API key.

3.4 Coding style guidelines

3.4.1 General

Ensure all user-enterable strings are unicode capable. Use only English language for everything (code, documentation, logs, comments, . . .)

3.4.2 Python

We follow PEP-8, with particular emphasis on the parts about knowing when to be inconsistent, and readability being the ultimate goal.

- Whitespace around operators and inside parentheses
- 4 spaces per indent, spaces, not tabs
- Include docstrings on your modules, class and methods
- Avoid from module import *. It can cause name collisions that are tedious to track down.
- Class should be in `CamelCase`, methods and variables in `lowercase_with_underscore`

3.4.3 Javascript

We follow W3 JavaScript Style Guide and Coding Conventions

CHAPTER 4

Abstraction

4.1 Definition

What we called abstraction is the askomics ontology, this is what describe the data. It is quite small and defines what is a bubble and what is a link the the graphical interface. Its prefix is "askomics:".

- entity : what will be bubble, usually a owl:Class
- startPoint : an entity that could start an askomics query. What will be displayed in the first query page.
- attribute : what will be links between bubbles or bubble and value.
- category : what will be choice list, used in some attribute value.

4.2 Turtle Example

Here i show you the minimal information to provide as an abstraction.

4.2.1 prefixes

```
@prefix xsd:      <http://www.w3.org/2001/XMLSchema#>
@prefix owl:      <http://www.w3.org/2002/07/owl#> .
@prefix rdfs:     <http://www.w3.org/2000/01/rdf-schema#> .
@prefix askomics: <askomics_is_good#> .
@base <scrap#> .
```

4.2.2 entity

```
# entity (startpoint to have a start, can be avoid in standard entity)
<People>
            askomics:entity "true"^^xsd:boolean ;
            rdfs:label "People"^^xsd:string ;
            askomics:startPoint "true"^^xsd:boolean ;
.
```

4.2.3 <entity> –relation–> value

```
# attribute DatatypeProperty
<First_name>
            askomics:attribute "true"^^xsd:boolean ;
            rdf:type    owl:DatatypeProperty ;
            rdfs:label  "First_name"^^xsd:string ;
            rdfs:domain <People> ;
            rdfs:range  xsd:string ;
.
```

4.2.4 <entity> –relation–> category=short list

```
# attribute DatatypeProperty
<Sex>
        askomics:attribute "true"^^xsd:boolean ;
        rdf:type    owl:DatatypeProperty ;
        rdfs:label  "Sex"^^xsd:string ;
        rdfs:domain <People> ;
        rdfs:range  <SexCategory> ;
.
<SexCategory>
        askomics:category <M>, <F> ;
.
<M>
    rdfs:label "M"^^xsd:string ;
.
<F>
    rdfs:label "F"^^xsd:string ;
.
```

4.2.5 <entity> –relation–> <entity>

```
# attribute ObjectProperty
<PlayWith>
        askomics:attribute "true"^^xsd:boolean ;
        rdf:type    owl:ObjectProperty ;
        rdfs:label  "play with"^^xsd:string ;
        rdfs:domain <People> ;
        rdfs:range  <People> ;
.
```

full file in people_mini.abstract.ttl

4.3 Python Management Code

As seen above, we have 2 kind of classes, “entity” and “attribute”/relation. To manage them (~get turtle strings), we use the 2 classes `AbstractedEntity__` and `AbstractedRelation__` in libaskomics/integration.

cf **python doc** to have details.

4.3.1 basics uses

```
ttl  += AbstractedEntity__( uri, label, startpoint=True ).get_turtle()
ttl  += AbstractedRelation__( uri, rdf_type, domain, range_, label ).get_turtle()
```

CHAPTER 5

askomics package

5.1 Subpackages

5.1.1 askomics.libaskomics package

Subpackages

askomics.libaskomics.integration package

Submodules

askomics.libaskomics.integration.AbstractedEntity module

class askomics.libaskomics.integration.AbstractedEntity.**AbstractedEntity**(*identifier*, *prefix*)

Bases: object

An AbstractedEntity represents the classes of the database. It is defined by an uri and a label.

get_turtle()
return the turtle code describing an AbstractedEntity for the abstraction file generation.

get_uri()

askomics.libaskomics.integration.AbstractedRelation module

class askomics.libaskomics.integration.AbstractedRelation.**AbstractedRelation**(*relation_type*, *identifier*, *label*, *identifier_prefix*, *rdfs_domain*, *prefixDomain*, *rdfs_range*, *prefixRange*)

Bases: object

An AbstractedRelation represents the relations of the database. There are two kinds of relations:

- ObjectProperty binds an instance of a class with another.
- DatatypeProperty binds an instance of a class with a string or a numeric value.

In Askomics, an ObjectProperty can be represented as:

- a node on the display graph (relation_type = entity).
- an attribute of a node (relation_type = category).

All DatatypeProperty are represented as nodes attributes. Each relation has an uri composed by the database prefix (:), "**has_**" and an identifier that is the header of the tabulated file being converted. Each relation also has a domain (the class of the source node) and a range (the class of the target). The range is the header of the tabulated file being converted in case of ObjectProperty and a specified class (xsd:string or xsd:numeric) in case of DatatypeProperty.

get_domain()

get_label()

get_range()

get_relation_type()

get_turtle()
return the turtle code describing an AbstractedRelation for the abstraction file generation.

get_uri()

Module contents

askomics.libaskomics.rdfdb package

Submodules

askomics.libaskomics.rdfdb.FederationQueryLauncher module

class askomics.libaskomics.rdfdb.FederationQueryLauncher.**FederationQueryLauncher**(*settings*, *session*, *lendpoints*)

Bases: *askomics.libaskomics.rdfdb.QueryLauncher.QueryLauncher*

The QueryLauncher process sparql queries:

- execute_query send the query to the sparql endpoint specified in params.
- parse_results preformat the query results
- format_results_csv write in the tabulated result file a table obtained from these preformated results using a ResultsBuilder instance.

process_query(*query*)
Execute query and parse the results if exist

askomics.libaskomics.rdfdb.MultipleQueryLauncher module

class askomics.libaskomics.rdfdb.MultipleQueryLauncher.**MultipleQueryLauncher**(*settings*, *session*)

Bases: *askomics.libaskomics.rdfdb.QueryLauncher.QueryLauncher*

The MultipleQueryLauncher process sparql queries. Send SPARQL query on multiple endpoint. Useful to reach several AskOmics Endpoint

process_query(*query*, *lendpoints*, *indexByEndpoint=False*)
Execute query and parse the results if exist

askomics.libaskomics.rdfdb.QueryLauncher module

exception askomics.libaskomics.rdfdb.QueryLauncher.**EndpointError**(*ep_uri*, *msg=''*)

Bases: RuntimeError

exception askomics.libaskomics.rdfdb.QueryLauncher.**NotEndpoint**(*ep_uri*, *msg=''*)

Bases: *askomics.libaskomics.rdfdb.QueryLauncher.EndpointError*

class askomics.libaskomics.rdfdb.QueryLauncher.**QueryLauncher**(*settings*, *session*, *name=None*, *endpoint=None*, *username=None*, *password=None*, *urlupdate=None*, *auth='Basic'*)

Bases: *askomics.libaskomics.rdfdb.QueryLauncher.QueryLauncher_*

Specialization of **QueryLauncher_**. For now, insert/load query are kept here.

Public endpoint may not be writable. This can change in future version.

insert_data(*ttl_string*, *graph*, *ttl_header=''*)
Load a ttl string into the triple store using INSERT DATA method

Parameters

- **ttl_string** – ttl content to load
- **ttl_header** – the ttl header associated with ttl_string

Returns The status

load_data(*url*, *graphName*)
Load a ttl file accessible from http into the triple store using LOAD method

Parameters **url** – URL of the file to load

Returns The status

setUserDatastore()
initialize endpoint with user configuration file

setupSPARQLWrapper()
Setup SPARQLWrapper to reach url endpoint

upload_data(*filename*, *graphName*)
Load a ttl file into the triple store using requests module and Fuseki upload method which allows upload of big data into Fuseki (instead of LOAD method).

Parameters **filename** – name of the file, fp.name from Source.py

Returns response of the request and queryTime

Not working for Virtuoso because there is no upload files url.

class askomics.libaskomics.rdfdb.QueryLauncher.**QueryLauncher_**(*settings*, *session*, *name=None*, *endpoint=None*, *username=None*, *password=None*, *urlupdate=None*)

Bases: *askomics.libaskomics.ParamManager.ParamManager*

The QueryLauncher_ process sparql queries:

- execute_query send the query to the sparql endpoint specified in params.
- parse_results preformat the query results
- format_results_csv write in the tabulated result file a table obtained from these preformated results using a ResultsBuilder instance.

It is a generalization of QueryLauncher, but that fit any endpoint

debug()

format_results_csv(*data*)
write the csv result file from a data ist

Parameters **data** (*list*) – the data to process

Returns The path of the created file

Return type string

parse_results(*json_res*)
parse answer results from TPS

process_query(*query*, *parseResults=True*)
Execute query and parse the results if exist

setUserDatastore()
initialize endpoint with user configuration file

setupSPARQLWrapper()
Setup SPARQLWrapper to reach url endpoint

setup_opener(*proxy_config*)
Sets up a urllib OpenerDirector to be used for requests behind a proxy. :param proxy_config:

- Nothing to do if proxy_config == "auto" (use the system's proxy if any).
- Set an empty ProxyHandler if proxy_config == "noproxy".
- Specify the proxy parameters according to the configuration ini file if proxy_config == "custom". We handle Basic and Digest Auth. No ntlm support, try "noproxy" if the triplestore runs on the same server as AskOmics.

test_endpoint()
Test endpoint existance by sending a dummy query.

exception askomics.libaskomics.rdfdb.QueryLauncher.**SPARQLError**(*response*)
Bases: RuntimeError

The SPARQLError returns an error message when a query sends by requests module encounters an error.

askomics.libaskomics.rdfdb.SparqlQueryAuth module

class askomics.libaskomics.rdfdb.SparqlQueryAuth.**SparqlQueryAuth**(*settings*, *session*)
Bases: *askomics.libaskomics.rdfdb.SparqlQueryBuilder.SparqlQueryBuilder*

This class contain method to build a sparql query to extract data from the users graph

add_apikey(*username*, *keyname*)
Insert a new api key

add_galaxy(*username*, *url*, *key*)
Insert a galaxy instance (url + api key)

check_email_presence(*email*)
Check if an email is present in users graph

check_galaxy(*username*)
Check if user have a galaxy

check_username_presence(*username*)
Check if a username is present in users graph

ckeck_key_belong_user(*username*, *key*)
Chek if a key belong to a user

delete_apikey(*key*)
Delet all info of a user

delete_galaxy(*username*)
delete galaxy triples of a user

get_admin_blocked_by_email(*email*)
get if a user is admin, by his email

get_admin_blocked_by_username(*username*)
get if a user is admin, by his username

get_admins_emails()
Get emails of all admins

get_galaxy_infos(*username*)
Get Galaxy url and apikey of a user

get_number_of_users()
Get the number of users

get_owner_apikey(*key*)
Get the owner of the API key

get_password_with_email(*email*)
Check the password of a user by his email

get_password_with_username(*username*)
Check the password of a user by his username

static get_random_string(*number*)
return a random string of n character

get_user_infos(*username*)
Get infos about one user

get_username_by_email(*email*)
Get usermail of a user by his email

get_users_infos(*username*)
Get users infos exept me

update_mail(*username*, *email*)
update the email of user

update_passwd(*username*, *shapw*, *salt*)
update the email of user

askomics.libaskomics.rdfdb.SparqlQueryBuilder module

class askomics.libaskomics.rdfdb.SparqlQueryBuilder.**SparqlQueryBuilder**(*settings*, *session*)

Bases: *askomics.libaskomics.ParamManager.ParamManager*

SparqlQueryBuilder generate a Sparql query string containing the query corresponding to an AskOmics graph (with load_from_query_json) or the pre-written query of a template file (with load_from_file).

build_query_on_the_fly(*replacement*, *adminrequest=False*, *externalrequest=False*)
Build a query from the private or public template

custom_query(*fromgraph*, *select*, *query*, *externalrequest=False*, *adminrequest=False*)
launch a custom query.

delete_user(*username*)
Delet all info of a user

getExternalServiceEndpoint()
Get all external endpoint finding in all askomics endpoint

getGraphUser(*removeGraph=[]*)

get_delete_metadatas_of_graph(*graph*)
Delte metadata linkd to a graph

get_delete_query_string(*graph*)
clear a graph

get_drop_named_graph(*graph*)
remove a graph

get_graph_of_user(*username*)
Get all subgraph of a user

prepare_query(*template*, *replacement={}*)
Prepare a query from a template and a substitution dictionary. The *$graph* variable is the public graph The *$graph2* variable is user graph or public graph if no user logged

update_admin_status(*admin*, *username*)
Change the admin status of a user!

update_blocked_status(*blocked*, *username*)
hello!

askomics.libaskomics.rdfdb.SparqlQueryGraph module

Classes to query triplestore on property of entity (attributes and relations).

information on build_query_on_the_fly:

- When querying as asdmin : build_query_on_the_fly(QUERY, True)

==> The query have to contains GRAPH ?g { ... } because all data are store on a Graph

- When querying as a classic user : build_query_on_the_fly(QUERY) or build_query_on_the_fly(QUERY, False)

=> The query can not contain the GRAPH keyword because 'FROM' clauses cause all triplets are merged in the unique DEFAULT graph !!

class askomics.libaskomics.rdfdb.SparqlQueryGraph.**SparqlQueryGraph**(*settings*, *session*)
Bases: *askomics.libaskomics.rdfdb.SparqlQueryBuilder.SparqlQueryBuilder*

This class contain method to build a sparql query to extract data from public and private graph It replace the template files

get_abstraction_positionable_entity()
Get all positionable entities

get_all_taxons()
Get the list of all taxon

get_class_info_from_abstraction(*node_class*)
get

get_if_positionable(*uri*)
Get if an entity is positionable

get_isa_relation_entities()
Get the association list of entities and subclass

get_prefix_uri()
Get list of uri defined as metadata for a entities list

get_public_abstraction_attribute_entity()
Get all attributes of an entity

get_public_abstraction_category_entity()
Get the category of an entity

get_public_abstraction_entity()
Get theproperty of an entity

get_public_abstraction_relation(*prop*)
Get the relation of an entity

get_public_graphs()
Get the list of public named graph

get_public_start_point()
Get the start point and in which public graph they are

get_user_abstraction_attribute_entity()
Get all attributes of an entity

get_user_abstraction_category_entity()
Get the category of an entity

get_user_abstraction_entity()
Get theproperty of an entity

get_user_abstraction_relation(*prop*)
Get the relation of an entity

get_user_graph_infos_with_count()
Get infos of all datasets owned by a user

get_user_start_point()
Get the start point and in which private graph they are

query_exemple()
Query exemple. used for testing

askomics.libaskomics.rdfdb.SparqlQueryStats module

Classes to query triplestore on stats data.

information on build_query_on_the_fly:

- When querying as asdmin : build_query_on_the_fly(QUERY, True)

==> The query have to contains GRAPH ?g { ... } because all data are store on a Graph

- When querying as a classic user : build_query_on_the_fly(QUERY) or build_query_on_the_fly(QUERY, False)

=> The query can not contain the GRAPH keyword because 'FROM' clauses cause all triplets are merged in the unique DEFAULT graph !!

class askomics.libaskomics.rdfdb.SparqlQueryStats.**SparqlQueryStats**(*settings*, *session*)
Bases: *askomics.libaskomics.rdfdb.SparqlQueryBuilder.SparqlQueryBuilder*

This class contain method to build a sparql query to extract data from the users graph

condition_query(*access_level*)
return the query according the accessLevel

get_attr_of_classes(*access_level*)
Get all the attributes of a class

get_number_of_classes(*access_level*)
Get number of triples in public graph

get_number_of_entities(*access_level*)
Get number of triples in public graph

get_number_of_subgraph(*access_level*)
Get number of triples in public graph

get_number_of_triples(*access_level*)
Get number of triples in public graph

get_rel_of_classes(*access_level*)
Get all the attributes of a class

get_subgraph_infos(*access_level*)
Get number of triples in public graph

Module contents

askomics.libaskomics.source_file package

Submodules

askomics.libaskomics.source_file.SourceFile module

Classes to import data from source files

class askomics.libaskomics.source_file.SourceFile.**SourceFile**(*settings*, *session*, *path*, *uri_set=None*)

Bases: *askomics.libaskomics.ParamManager.ParamManager*, *askomics.libaskomics.utils.HaveCachedProperties*

Class representing a source file.

get_number_of_lines()
Get the number of line of a tabulated file

Returns number of ligne (int)

get_timestamp()
return the timestamp (use in the tests)

insert_metadatas(*accessL*)
Insert the metadatas into the parent graph

load_data_from_file(*fp*, *urlbase*)
Load a locally created ttl file in the triplestore using http (with load_data(url)) or with the filename for Fuseki (with fuseki_load_data(fp.name)).

Parameters **fp** – a file handle for the file to load

:param urlbase:the base URL of current askomics instance. It is used to let triple stores access some askomics temporary ttl files using http. :return: a dictionnary with information on the success or failure of the operation

persist(*urlbase*, *public*)
Store the current source file in the triple store

Parameters **urlbase** – the base URL of current askomics instance. It is used to let triple stores access some askomics temporary ttl files using http.

Returns a dictionnary with information on the success or failure of the operation

Return type Dict

setGraph(*graph*)

exception askomics.libaskomics.source_file.SourceFile.**SourceFileSyntaxError**
Bases: SyntaxError

askomics.libaskomics.source_file.SourceFileBed module

Classe to import data from a bed source file

class askomics.libaskomics.source_file.SourceFileBed.**SourceFileBed**(*settings*, *session*, *path*, *uri_set=None*)
Bases: *askomics.libaskomics.source_file.SourceFile.SourceFile*

Class representing a BED Source file

get_abstraction()
Get abstraction of a bed file

Returns abstraction in turtle

Return type string

get_domain_knowledge()
Get domain knowledge of a bed file

Returns domain knowledge in turtle

Return type string

get_turtle()
Get turtle content for a bed file

Yield the ttl string

Return type string

open_bed()
Try to parse the file

set_entity_name(*entity*)
set the entity name

set_taxon(*taxon*)
Set the taxon

askomics.libaskomics.source_file.SourceFileGff module

Classes to import data from a gff3 source files

class askomics.libaskomics.source_file.SourceFileGff.**SourceFileGff**(*settings*, *session*, *path*, *uri_set=None*)

Bases: *askomics.libaskomics.source_file.SourceFile.SourceFile*

Class representing a Gff3 Source file

get_abstraction()
Get Abstraction (turtle) of the GFF

get_content_ttl(*entity*)
Get the ttl string for an entity

get_domain_knowledge()
Get Domain Knowledge (turtle) of the GFF

get_entities()
get all the entities present in a gff file

Returns The list of all the entities

Return type List

get_turtle()
Get turtle string for a gff file

set_entities(*entities*)

set_taxon(*taxon*)

askomics.libaskomics.source_file.SourceFileTsv module

Classes to import data from a tsv source files

class askomics.libaskomics.source_file.SourceFileTsv.**SourceFileTsv**(*settings*, *session*, *path*, *preview_limit*, *uri_set=None*)

Bases: *askomics.libaskomics.source_file.SourceFile.SourceFile*

Class representing a Gff3 Source file

category_values
Like @property on a member function, but also cache the calculation in self.__dict__[function name]. The function is called only once since the cache stored as an instance attribute override the property residing in the class attributes. Following accesses cost no more than standard Python attribute access. If the instance attribute is deleted the next access will re-evaluate the function. Source: https://blog.ionelmc.ro/2014/11/04/an-interesting-python-descriptor-quirk/ usage:

class Shape(object):

@cached_property def area(self):

compute value return value

dialect
Like @property on a member function, but also cache the calculation in self.__dict__[function name]. The function is called only once since the cache stored as an instance attribute override the property residing in the class attributes. Following accesses cost no more than standard Python attribute access. If the instance attribute is deleted the next access will re-evaluate the function. Source: https://blog.ionelmc.ro/2014/11/04/an-interesting-python-descriptor-quirk/ usage:

class Shape(object):

@cached_property def area(self):

compute value return value

get_abstraction()
Get the abstraction representing the source file in ttl format

Returns ttl content for the abstraction

get_domain_knowledge()
Get the domain knowledge representing the source file in ttl format

Returns ttl content for the domain knowledge

get_headers_by_file
Like @property on a member function, but also cache the calculation in self.__dict__[function name]. The function is called only once since the cache stored as an instance attribute override the property residing in the class attributes. Following accesses cost no more than standard Python attribute access. If the instance attribute is deleted the next access will re-evaluate the function. Source: https://blog.ionelmc.ro/2014/11/04/an-interesting-python-descriptor-quirk/ usage:

class Shape(object):

@cached_property def area(self):

compute value return value

get_preview_data()
Read and return the values from the first lines of file.

Returns a List of List of column values

Return type List

static get_strand(*strand*)
Get the faldo strand in function of the strand

static get_strand_faldo(*strand*)
Get the faldo strand in function of the strand

get_turtle(*preview_only=False*)
Get the turtle string of a tsv file

guess_values_type(*values*, *header*)
From a list of values, guess the data type

Parameters

- **values** – a List of values to evaluate
- **num** – index of the header

Returns the guessed type ('taxon','ref', 'strand', 'start', 'end', 'numeric', 'text' or 'category', 'goterm')

static is_decimal(*value*)
Determine if given value is a decimal (integer or float) or not

Parameters **value** – the value to evaluate

Returns True if the value is decimal

key_id(*row*)
Get the key id by concatenate all key selected

set_disabled_columns(*disabled_columns*)
Set manually curated types for column

Parameters **disabled_columns** – a List of column ids (0 based) that should not be imported

set_forced_column_types(*types*)
Set manually curated types for column

Parameters **types** – a List of column types ('entity', 'entity_start', 'numeric', 'text' or 'category')

set_headers(*headers*)
Set the headers

Parameters **headers** (*list*) – the headers list

set_key_columns(*key_columns*)
Set all column to build unqiue ID

Parameters **disabled_columns** – a List of column ids (0 based) that should not be imported

askomics.libaskomics.source_file.SourceFileTtl module

Classes to import data from a RDF source files

class askomics.libaskomics.source_file.SourceFileTtl.**SourceFileTtl**(*settings*, *session*, *path*, *file_type='ttl'*)
Bases: *askomics.libaskomics.source_file.SourceFile.SourceFile*

Class representing a ttl Source file

convert_to_ttl(*filepath*, *file_type*)

file_get_contents(*filename*)
get the content of a file

get_preview_ttl()
Return the first 100 lines of a ttl file, text is formated with syntax color

static load_data_from_url(*self*, *url*, *public*)
insert the ttl sourcefile in the TS

persist(*urlbase*, *public*)
insert the ttl sourcefile in the TS

askomics.libaskomics.source_file.SourceFileURL module

Classes to import data from an URL

class askomics.libaskomics.source_file.SourceFileURL.**SourceFileURL**(*settings*, *session*, *url*)

Bases: *askomics.libaskomics.source_file.SourceFile.SourceFile*

Class representing a ttl Source file

load_data_from_url(*url*, *public*)
insert the ttl sourcefile in the TS

Module contents

Submodules

askomics.libaskomics.DatabaseConnector module

class askomics.libaskomics.DatabaseConnector.**DatabaseConnector**(*settings*, *session*)

Bases: *askomics.libaskomics.ParamManager.ParamManager*

Manage Database connection

create_endpoints_table()

create_galaxy_table()

create_integration_table()

create_query_table()

create_user_table()

execute_sql_query(*query*, *variables=None*, *get_id=False*)
execute a sql query

askomics.libaskomics.EndpointManager module

class askomics.libaskomics.EndpointManager.**EndpointManager**(*settings*, *session*)

Bases: *askomics.libaskomics.ParamManager.ParamManager*

disable(*id*, *message*)

disable_by_url(*url*, *message*)

enable(*id*)

list_active_endpoints()

list_endpoints()

remove_endpoint(*id*)

save_endpoint(*name*, *url*, *auth='BASIC'*, *isenable=False*)

askomics.libaskomics.GalaxyConnector module

class askomics.libaskomics.GalaxyConnector.**GalaxyConnector**(*settings*, *session*, *url*, *apikey*)

Bases: *askomics.libaskomics.ParamManager.ParamManager*

Connection with a Galaxy instance

Contain method to connect to a Galaxy instance with URL and APIkey and to get and send datasets to a Galaxy history

check_galaxy_instance()
Check if the galaxy instance

Check if the Galaxy url and the API key :raises: e

get_datasets_and_histories(*allowed_files*, *history_id=None*)
Get Galaxy datasets of current history and all histories of a user

Returns a list of datasets

Return type dict

get_file_content(*dataset_id*)
Get the content of a Galaxy dataset

Parameters **dataset_id** (*string*) – the dataset id

Returns the dataset content

Return type sring

send_json_to_history(*json*)
Send json data into the last used galaxy history

Parameters **json** (*string*) – json data to send

send_to_history(*path*, *name*, *filetype*)
Send a file into the most recent Galaxy history

Parameters **path** (*string*) – path of file to load into Galaxy

upload_files(*files_id*)
Upload galaxy datasets into AskOmics server

Parameters **files_id** (*list*) – Ids of Galaxy datasets to upload

askomics.libaskomics.JobManager module

class askomics.libaskomics.JobManager.**JobManager**(*settings*, *session*)
Bases: *askomics.libaskomics.ParamManager.ParamManager*

Manage Askomics jobs inside a sqlite database

done_integration_job(*jobid*)

done_query_job(*jobid*, *nrows*, *data*, *file*)

list_integration_jobs()

list_query_jobs()

remove_job(*table*, *jobid*)

save_integration_job(*filename*)

save_query_job(*request_graph*, *variates*)

set_error_message(*table*, *message*, *jobid*)

askomics.libaskomics.ParamManager module

class askomics.libaskomics.ParamManager.**ParamManager**(*settings*, *session*)
Bases: object

Manage static file and template sparql queries

static Bool(*result*)

static decode(*toencode*)

static decode_to_rdf_uri(*todecode*, *prefix=''*)

static encode(*toencode*)

static encode_to_rdf_uri(*toencode*, *prefix=None*)

get_directory(*name*, *force_username=None*)
Get a named directory of a user, create it if not exist

get_param(*key*)

get_rdf_directory()

get_rdf_user_directory()

get_sparql_prefixes(*sparqlrequest*)

get_turtle_prefixes(*ttl*)
Parse the ttl string, looking for prefix. add them to ASKOMICS_prefix if they exist. Then return the prefixes as ttl header.

get_turtle_template(*ttl*)

get_upload_directory()
Get the upload directory of a user, create it if not exist

Returns The path of the user upload directory

Return type string

get_user_csv_directory()

header_sparql_config(*sparqlrequest*)

is_defined(*key*)

remove_prefix(*obj*)

reverse_prefix(*uri*)

send_mails(*host_url*, *dests*, *subject*, *text*)

set_param(*key*, *value*)

update_list_prefix(*listPrefix*)

askomics.libaskomics.Security module

class askomics.libaskomics.Security.**Security**(*settings*, *session*, *username*, *email*, *password*, *password2*)
Bases: *askomics.libaskomics.ParamManager.ParamManager*

[summary]

[description]

add_galaxy(*url*, *key*)
Connect a galaxy account to Askomics

add triples for the url of galaxy, and the user api key

Parameters

- **url** (*self;*) – the galaxy url
- **key** (*string*) – the galaxy user api key

admin_user(*new_status*, *username*)
adminify a user

check_email()
Return true if email is a valid one

check_email_in_database()
Check if the email is present in the DB

check_email_password()
check if the password is the good password associate with the email

check_galaxy()
Check if user have a galaxy account

check_password_length()
Return true if password have at least 8 char

check_passwords()
Return true if the 2 passwd are identical

check_username_in_database()
Check if the username is present in the DB

check_username_password()
check if the password is the good password associate with the username

ckeck_key_belong_user(*key*)
Check if a key belong to a user

create_user_graph()
Create a subgraph for the user. All his data will be inserted in this subgraph

delete_galaxy()
Delete galaxy account for the user

delete_user(*username*)
delete a user

get_admin_blocked_by_email()
get the admin status of the user by his username

get_admin_blocked_by_username()
get the admin status of the user by his username

get_admins_emails()
Get all admins emails

get_galaxy_infos()
Get Galaxy url and apikey of a user

get_number_of_users()
get the number of users in the TS

get_owner_of_apikey(*key*)
Get the owner of an API kei

[description] :param key: The API key :type key: string

static get_random_string(*number*)
return a random string of n character

get_sha256_pw()
Get the hashed-salted password

Returns the hashed-salted password

Return type string

get_user_id_by_username()
get user id by is username

get_user_infos()
get all about a user

get_username()
get the username

get_users_infos()
get all about all user

lock_user(*new_status*, *username*)
lock a user

log_user(*request*)
log the user using pyramid's session

persist_user(*host_url*)
Persist all user infos in the TS

renew_apikey()
renew apikey of user

set_admin(*admin*)
set self.admin at True if user is an admin

set_blocked(*blocked*)
set self.blocked at True if user is a blocked

set_galaxy(*galaxy*)
set self.galaxy at True if user has a connected galaxy account

set_username_by_email()
Get the username of a user by his email

update_email()
change the mail of a user

update_passwd()
Change the password of a user, and his randomsalt

askomics.libaskomics.SourceFileConvertor module

class askomics.libaskomics.SourceFileConvertor.**SourceFileConvertor**(*settings*, *session*)

Bases: *askomics.libaskomics.ParamManager.ParamManager*

A SourceFileConvertor instance provides methods to:

- display an overview of the tabulated files the user want to convert in AskOmics.
- **convert the tabulated files in turtle files, taking care of:**
  - the format of the data already in the database (detection of new and missing headers in the user files).
  - the abstraction generation corresponding to the header of the user files.
  - the generation of the part of the domain code that wan be automatically generated.

get_source_files(*selectedFiles*, *forced_type=None*, *uri_set=None*)
Get all source files

Returns a list of source file

Return type list

static guess_file_type(*filepath*)
Guess the file type in function of their extention

Parameters **filepath** (*string*) – path of file

Returns file type

Return type string

askomics.libaskomics.TripleStoreExplorer module

class askomics.libaskomics.TripleStoreExplorer.**TripleStoreExplorer**(*settings*, *session*, *dico={}*)
Bases: *askomics.libaskomics.ParamManager.ParamManager*

Use the different Sparql template queries in order to:

- **get special settings:**
  - relation between two classes (nodes) specified by another class (hidden node).
  - virtual relation adding special Where clauses specified in the database domain.
- get the suggestion list for classes listed as Categories and displayed as node attributes by AskOmics.
- get the startpoints to begin a query building.
- get the neighbor nodes and the attributes of a node.

build_recursive_block(*tabul*, *constraints*)
build SPARQL Block following this grammar : B ==> [A , KEYWORKD] . KEYWORKD is a string prefix for BLOCK (ex: OPTIONAL, SERVICE) A ==> [((B|F),)+] . a list of Block or constraints leafs F ==> [CONSTRAINT1, CONSTRAINT2,. . . .] an array contains only constraints

build_sparql_query_from_json(*list_endpoints*, *typeEndpoints*, *fromgraphs*, *variates*, *constraintes_relations*, *limit*, *send_request_to_tps=True*)
Build a sparql query from JSON constraints

getUserAbstraction()
Get the user abstraction (relation and entity as subject and object)

Returns

Return type

get_prefix_uri()

get_start_points()
Get the possible starting points for your graph.

Returns List of starting points

Return type Node list

askomics.libaskomics.utils module

Some reusable utilities : Bases, decorators, utility functions, etc. Also a place to isolate "python magic" tricks.

askomics.libaskomics.utils.**pformat_generic_object**(*obj*)
Pretty print a object and its attributes to string

askomics.libaskomics.utils.**intersperse_chain**(*delimiter*, *iterators*)
yield delimiter between the elements of each iterators. >>> list(intersperse_chain('sep', [['str1', 'str2'], ['str3', 'str4'], ['str5']])) ['str1', 'str2', 'sep', 'str3', 'str4', 'sep', 'str5']

askomics.libaskomics.utils.**prefix_lines**(*prefix*, *strings*)

```
>>> list(prefix_lines('\t', ['line1', 'line2']))
['\tline1', '\tline2']
```

class askomics.libaskomics.utils.**cached_property**(*func*)
Bases: object

Like @property on a member function, but also cache the calculation in self.__dict__[function name]. The function is called only once since the cache stored as an instance attribute override the property residing in the class attributes. Following accesses cost no more than standard Python attribute access. If the instance attribute is deleted the next access will re-evaluate the function. Source: https://blog.ionelmc.ro/2014/11/04/an-interesting-python-descriptor-quirk/ usage:

class Shape(object):

@cached_property def area(self):

compute value return value

func

class askomics.libaskomics.utils.**HaveCachedProperties**
Bases: object

Provide cache management for classes with @cached_property attributes.

cache
Explicit access (as a dictionary) to all the values cached in the attributes of this object.

get_cache()
Explicit access (as a dictionary) to all the values cached in the attributes of this object. Return the subset of self.__dict__ for keys in self.get_cached_properties().

classmethod get_cached_properties()
cached_property instances are in the class dict with other class attributes

reset_cache()
Equivalent to self.set_cache({}, reset=True) or del self.cache

set_cache(*cache_dict*, *reset=True*)
Set the cache. If reset is True, reset caches not present in cache_dict keys(). With reset = False, a reset is still possible if cache_dict[key] is None

self.cache = cache_dict

is equivalent to: self.set_cache(cache_dict, reset=True)

Module contents

5.2 Submodules

5.3 askomics.ask_view module

class askomics.ask_view.**AskView**(*request*)
Bases: object

This class contains method calling the libaskomics functions using parameters from the js web interface (body variable)

add_endpoint()

api_key()

checkAdminSession()

checkAuthSession()

checkuser()

cleantmpdirectory()

connect_galaxy()

deletCsv()

deleteShortcut()
Delete a shortcut definition into the triplestore

delete_endpoints()

delete_graph()

delete_uploaded_files()

delete_user()
Delete a user from the user graphs, and remove all his data

deljob()
Remove job from database

empty_database()
Delete all named graphs and their metadatas

enable_endpoints()

getSparqlQueryInTextFormat()
Build a request from a json whith the following contents :variates,constraintesRelations

getUserAbstraction()
Get the user asbtraction to manage relation inside javascript

get_data_from_galaxy()

get_galaxy_file_content()

get_my_infos()
Get all infos about a user

get_uploaded_files()

get_users_infos()
For each users store in the triplesore, get their username, email, and admin status

get_value()
Build a request from a json whith the following contents :variates,constraintesRelations

guess_csv_header_type()
Guess the headers type of a csv file

Used for the asko-cli scripts

Returns list of guessed types

Return type dict

importShortcut()
Import a shortcut definition into the triplestore

list_endpoints()
Return a list with all endpoint using by a askomics session.

list_user_graph()
Return a list with all the named graphs of a user.

listjob()
Get all jobs recorded in database

load_bed_into_graph()
Load a BED file into the triplestore

load_data_into_graph()
Load tabulated files to triple store according to the type of the columns set by the user

load_gff_into_graph()
Load GFF file into the triplestore

load_remote_data_into_graph()
Load tabulated files to triple store according to the type of the columns set by the user

load_ttl_into_graph()
Load TTL file into the triplestore

lock_user()
Change a user lock status

login()

login_api()

login_api_gie()

logout()
Log out the user, reset the session

nbUsers()

prefix_uri()
get prefix uri for each entities finded in he header file

preview_ttl()
Convert tabulated files to turtle according to the type of the columns set by the user

renew_apikey()

send2galaxy()

serverinformations()

set_admin()
Change a user admin status

signup()

source_files_overview()
Get preview data for all the available files

start_points()
Get the nodes being query starters

statistics()
Get stats

update_mail()
Chage email of a user

update_passwd()
Change password of a user

uploadCsv()

uploadTtl()

upload_galaxy_file()

5.4 askomics.upload module

class askomics.upload.**FileUpload**(*request*)
Bases: object

delete()

fileinfo(*name*)

filepath(*name*)

get()

get_file_size(*new_file*)

options()

post()

upload()

validate(*new_file*)

5.5 askomics.views module

askomics.views.**my_view**(*request*)

5.6 Module contents

askomics.**main**(*global_config*, ***settings*)
This function returns a Pyramid WSGI application.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V