grano-client Documentation

Release 0.5

Friedrich Lindenberg

October 15, 2014

Contents

grano is extended by a comprehensive Python client library that can be used to edit projects, data schemata, entities and their relations remotely:

```
import granoclient

client = granoclient.Grano()
for project in client.projects:
    print project.label

project = client.get('my-project')
project.label = 'New title'
project.save()

data = {'schemata': ['base'], 'properties': {'name': {'value': 'Alice'}}}
alice = project.entities.create(data)

data = {'schemata': ['base'], 'properties': {'name': {'value': 'Bob'}}}
bob = project.entities.create(data)

rel = {'schema': 'my-schema', 'source': alice, 'target': bob, 'properties': {}}
project.relations.create(rel)

query = project.entities.query().filter('properties-name', 'Alice')
for entity in query:
    print entity.properties.get('name').get('value')
```

CHAPTER 1

Installation

The easiest way to install the client is via the Python package index and pip/easy_install:

```
pip install grano-client
```

If you want to develop the client library's code, check out the repository and set up dependencies etc. with the command:

```
python setup.py develop
```

grano-client depends on requests newer than 2.2.

CHAPTER 2

Configuration

Several aspects of grano-client can be configured, including the host name of the grano server and the API key that is to be used for authentication. To determine these settings, the library will evaluate the following configuration sources in given order:

1. Read the `~/.grano.ini` file in the user's home directory. The file is a simple .ini configuration as detailed below.
2. Check the contents of the following environment variables: `GRANO_HOST`, `GRANO_APIKEY`.
3. Evaluate the keyword arguments passed into the constructor of `granoclient.Grano`.

A simple configuration file for grano-client might look like this:

```
[client]
host = http://beta.grano.cc

# see user profile in grano:
api_key = xxxxxxxxxxxxxxx
```

CHAPTER 3

API

3.1 Grano

class `granoclient.`**`Grano`**(*api_host=None*, *api_key=None*, *api_prefix='/api/1/'*)

Grano client library. This class provides basic access to many of the core APIs for grano, including projects, and a global view of entities and relations (seldomly useful, consider using the project-filtered versions instead.)

The client library can be configured directly, or through a configuration file (`~/.grano.ini`) and a set of environment variables.

Parameters

- **api_host** – (optional) host name URL to connect to, without any path information (e.g. `http://grano.io`).
- **api_key** – (optional) API key of the user which is running the requests.
- **api_prefix** – (optional) path prefix of the grano API, usually `/api/1/`.

`entities`

Returns a `granoclient.EntityCollection` of all available entities in this instance of grano.

Consider using the `entities` property of a specific `granoclient.Project` instead.

`get`(*slug*)

Get a project. Shortcut to `Grano.projects.by_slug()`.

Parameters **slug** – the slug identifying the project to be retrieved.

`projects`

Returns a `granoclient.ProjectCollection` of all available projects in this instance of grano.

`relations`

Returns a `granoclient.RelationCollection` of all available relation in this instance of grano.

Consider using the `relations` property of a specific `granoclient.Project` instead.

3.2 Projects

class `granoclient.`**`ProjectCollection`**(*client*)

Represents all the `granoclient.Project` currently available in this instance of grano. Provides functionality to search for, filter and create elements.

all()
Iterate over all available resources in the collection. This can also be done by just iterating over the collection:

```
for resource in collection:
    ...
```

by_slug(*slug*)
Load a project based on its slug, i.e. its unique designation.

Parameters **slug** – the slug of the project to be retrieved.

create(*data*)
Create a new project.

Parameters **data** – A dictionary with the projects attributes, `slug` and `label` are required.

query(*params=None*)
Begin querying the collection. The query can further be refined using the methods of the returned `granoclient.Query`.

class granoclient.**Project**(**args*, ***kwargs*)
A project within grano. This type serves as a namespace for use cases of the application. Each project has its own schemata, entities and relations.

entities
A collection of the `granoclient.Entity` associated with this project.

relations
A collection of the `granoclient.Relation` associated with this project.

reload()
Reload the resource from the server. This is useful when the resource is a shortened index representation which needs to be traded in for a complete representation of the resource.

save()
Update the server with any local changes, then update the local version with the returned value from the server.

schemata
A collection of the `granoclient.Schema` associated with this project.

3.3 Schemata

class granoclient.**SchemaCollection**(*client*, *project_slug*)
Represents all the `granoclient.Schema` currently available in a given project. Provides functionality to search for, filter and create elements.

all()
Iterate over all available resources in the collection. This can also be done by just iterating over the collection:

```
for resource in collection:
    ...
```

by_name(*name*)
Load a schema based on its name, i.e. its unique designation.

Parameters **name** – the name of the project to be retrieved.

create (*data*)
Create a new schema.

Parameters data – A dictionary with the schemas attributes, `name` and `label` are required, but several `attributes` should be given. See *schema* for details.

query (*params=None*)
Begin querying the collection. The query can further be refined using the methods of the returned `granoclient.Query`.

class granoclient.**Schema** (*client*, *base_endpoint*, *data*)
A schema within grano. Schemata define the types of entities and relations that are stored within a grano project. See also *schema*.

reload ()
Reload the resource from the server. This is useful when the resource is a shortened index representation which needs to be traded in for a complete representation of the resource.

save ()
Update the server with any local changes, then update the local version with the returned value from the server.

3.4 Entities

class granoclient.**EntityCollection** (*client*, *params={}*)
Represents all the `granoclient.Entity` currently available in this instance of grano. Provides functionality to search for, filter and create elements.

all ()
Iterate over all available resources in the collection. This can also be done by just iterating over the collection:

```
for resource in collection:
    ...
```

by_id (*id*)
Load an entity based on its id, i.e. its unique designation.

Parameters id – the id of the entity to be retrieved.

create (*data*)
Create a new entity.

Parameters data – A dictionary with the entity attributes, `schema` and `properties` are required.

query (*params=None*)
Begin querying the collection. The query can further be refined using the methods of the returned `granoclient.Query`.

class granoclient.**Entity** (**args*, ***kwargs*)
An entity within grano. This type serves as a node, which can be used to store data (in the form of properties), and can be part of relations..

inbound
Inbound relations as a filtered `granoclient.RelationCollection`.

outbound
Outbound relations as a filtered `granoclient.RelationCollection`.

project
The `granoclient.Project` to which this entity belongs.

reload()
Reload the resource from the server. This is useful when the resource is a shortened index representation which needs to be traded in for a complete representation of the resource.

save()
Update the server with any local changes, then update the local version with the returned value from the server.

3.5 Relations

class `granoclient.`**RelationCollection**(*client*, *params={}*)
Represents all the `granoclient.Relation` currently available in this instance of grano. Provides functionality to search for, filter and create elements.

all()
Iterate over all available resources in the collection. This can also be done by just iterating over the collection:

```
for resource in collection:
    ...
```

by_id(*id*)
Load a relation based on its id, i.e. its unique designation.

Parameters id – the id of the relation to be retrieved.

create(*data*)
Create a new relation.

Parameters data – A dictionary with the relation attributes, `schema` `source`, `target` and `properties` are required.

query(*params=None*)
Begin querying the collection. The query can further be refined using the methods of the returned `granoclient.Query`.

class `granoclient.`**Relation**(**args*, ***kwargs*)
A relation within grano. This type serves as a connection between two entities, it can also be used to store data (in the form of properties).

project
The `granoclient.Project` to which this relation belongs.

reload()
Reload the resource from the server. This is useful when the resource is a shortened index representation which needs to be traded in for a complete representation of the resource.

save()
Update the server with any local changes, then update the local version with the returned value from the server.

source
The source `granoclient.Entity`.

target
The target `granoclient.Entity`.

3.6 Queries

Queries are re-used whenever a result set needs to be paginated and filtered.

class `granoclient.`**`Query`**(*client*, *clazz*, *endpoint*, *params=None*)
A query is a mechanism to store query state and paginate through result sets returned by the server.

`filter`(*name*, *value*)
Apply a filter to the query and return a modified version.

Parameters

- **name** – the name of the query argument to add.
- **value** – the value of the query argument to add.

`has_next`
Check to see if a next page is available.

`has_prev`
Check to see if a previous page is available.

`next`
Return a derived query for the next page of elements.

`prev`
Return a derived query for the previous page of elements.

`results`
The current page's results.

`total`
The total number of results available (across all pages).

3.7 Exceptions

class `granoclient.`**`GranoException`**(*message*)
An exception produced by the grano client library, possibly as part of it's interaction with the server.

class `granoclient.`**`GranoServerException`**(*response*)
An exception produced by the grano server. The most common types of errors include:

- Missing objects (404), i.e. the client requested data that does not exist on the server.
- Invalid inputs (400), i.e. the data submitted by the client did not pass validation - it may be incomplete.

class `granoclient.`**`NotFound`**(*response*)

class `granoclient.`**`InvalidRequest`**(*response*)

CHAPTER 4

Indices and tables

- *genindex*
- *modindex*
- *search*

Index