django-datastream Documentation

Release 0.2.1

wlan slovenija

Apr 05, 2017

Contents

This Django package provides a RESTful read-only HTTP interface to time-series Datastream API.

CHAPTER 1

Contents

Installation

Inside a virtualenv, using pip simply by doing:

```
pip install django-datastream
```

Or install from source directly.

If installing from source, you can at this point try out *demo*, or proceed with integration with your Django project.

You should add `django_datastream` to `INSTALLED_APPS` in your `settings.py`.

Suggested settings are:

```
INSTALLED_APPS += (
    'tastypie',
    'django_datastream',
)

USE_TZ = True

MONGO_DATABASE_NAME = 'project_name'
MONGO_DATABASE_OPTIONS = {
    'tz_aware': USE_TZ,
}

DATASTREAM_BACKEND = 'datastream.backends.mongodb.Backend'
DATASTREAM_BACKEND_SETTINGS = {
    'database_name': MONGO_DATABASE_NAME,
    'tz_aware': USE_TZ,
}

# JSONP support as well
TASTYPIE_DEFAULT_FORMATS = ('json', 'jsonp', 'xml')
```

Usage

This Django package provides a RESTful read-only HTTP interface by using django-tastypie. It extends it to provide access to Datastream API and you can extend the interface further. Read Tastypie documentation to learn more how to configure and customize the interface.

HTTP API

Note: We assume that API is nested under `/api/` URI prefix.

List of all streams can be obtained at:

```
/api/v1/stream/
```

To get results in JSON format in the browser, you have to append `?format=json`:

```
/api/v1/stream/?format=json
```

You can limit the results by filtering on `tags`:

```
/api/v1/stream/?tags__title=Stream%201
/api/v1/stream/?tags__title__icontains=stream
/api/v1/stream/?tags__label__in=foo,bar
```

Accessing particular stream is through its ID, for example:

```
/api/v1/stream/caa88489-fa0f-4458-bc0b-0d52c7a31715/
```

API is read-only and supports the GET HTTP command for each stream. It returns a list of datapoints stored in a stream. Response contains some additional metadata to allow pagination and automatic interface exploration/discovery easier.

Because of potentially long streams, additional parameters can be specified to limit the interval of datapoints through query string parameters (default is all datapoints):

```
/api/v1/stream/caa88489-fa0f-4458-bc0b-0d52c7a31715/?start=<start timestamp>&end=<end␣
↪timestamp>
```

Timestamps are in seconds since UNIX epoch. If start or end timestamp is missing, this means all datapoints from the beginning of the stream, or all datapoints to the end of the stream, respectively. Start and end timestamps are inclusive. If you want exclusive timestamps, you can use `start_exclusive` and `end_exclusive` query string parameters.

Additionally, paging query string parameters can be used:

```
/api/v1/stream/caa88489-fa0f-4458-bc0b-0d52c7a31715/?limit=<page limit>&ooffset=
↪<offset>
```

Page limit limits absolute number of datapoints returned in this response and offset allows offsetting the datapoints, positive from beginning. You can specify `reverse` to reverse the order of datapoints.

Metadata in the response contains data on how many datapoints would there be otherwise in the response and URIs to previous and next page. Setting page limit to 0 allows simple querying of the URI without retrieving any data. Default page limit is 100 datapoints.

Together with some metadata datapoints are returned as a list of `t` (time) and `v` (value) values or dictionaries, depending on granularity level requested. Which data is returned can be configured with query parameters:

- `granularity` – granularity (`seconds`, `10seconds`, `minutes`, `10minutes`, `hours`, `6hours`, and `days`)
- `v` – value downsamplers, you can specify them to limit returned downsampled values; a comma-separated list or specified multiple times in the query
- `t` – time downsamplers, you can specify them to limit returned downsampled timestamps; a comma-separated list or specified multiple times in the query

For example, to return minutes granularity with only average, minimum, and maximum values:

```
/api/v1/stream/caa88489-fa0f-4458-bc0b-0d52c7a31715/?granularity=minutes&value_
↪downsamplers=mean,max&value_downsamplers=min
```

For all query parameters there exists also shorter forms to allow more complicated queries without having to worry about URI length.

Demo

Together with tests a demo project is provided. If you want to try it out, go to `tests` directory and run `manage.py` there.

To prepare data for a demo project, start MongoDB database, and run:

```
./manage.py dummystream --demo
```

This populates database with three streams and some random datapoints. You can provide different options to the command for different results.

After it finishes initial generation of datapoints and downsamples them, you can additionally run Django development server:

```
./manage.py runserver
```

Open the demo web page where you should see a visualization of three streams you can interact with. This visualization uses HTTP interface this package provides.

CHAPTER 2

Source Code, Issue Tracker and Mailing List

For development GitHub is used, so source code and issue tracker is found there. If you have any questions or if you want to discuss the project, use development mailing list.

CHAPTER 3

Indices and Tables

- genindex
- search