MapRoulette API Wrapper Documentation

Release 0.2.9

Martijn van Exel

February 28, 2016

Contents

This project provides a convenient wrapper around the MapRoulette API. You can create and maintain Challenges and Tasks on a local MapRoulette server or on the main MapRoulette dev and production servers.

CHAPTER 1

Installation

See the project README

CHAPTER 2

Access Credentials

The MapRoulette Challenge administration API endpoints are protected with HTTP basic authentication. This means that you will need to supply a username and password for most operations to work. If you do not have credentials, contact maproulette@maproulette.org. Passwords are transmitted in plain text, so use a password you do not use anywhere else.

CHAPTER 3

Prepare a challenge

You need a few things to create your own MapRoulette challenge:

- Challenge metadata. At the very least you need a title, a slug and some instructions to show the user.
- Tasks.
- Access to a server. It is recommended that you try your challenge on a local development server first, then move on to the main MapRoulette servers.

Once you have those things, you can get to work!

First, we get a MapRoulette server instance:

```
>>> from maproulette import MapRouletteServer
>>> server = MapRouletteServer(
    url='http://localhost:5000/api',
    user=foo,
    password=bar)
```

This will get a `MapRouletteServer` instance that points at a local MapRoulette development server at `http://localhost:5000`.

Let's see if it is alive:

```
>>> server.alive()
True
```

Next, we create a new Challenge on this server:

```
from maproulette import MapRouletteChallenge
challenge = MapRouletteChallenge(
    slug='test-challenge',
    title='Test Challenge')
challenge.create(server)
```

Finally, let's prepare a task and add it to the challenge:

```
from maproulette import MapRouletteTask
from geojson import FeatureCollection, Feature, Point
task = MapRouletteTask(
    challenge,
    identifier='test-task-1',
    geometries=FeatureCollection([Feature(
        geometry=Point((random(), random())))]))
task.create(server))
```

See how we use `geojson.FeatureCollection`, `geojson.Feature` and `geojson.Point` to generate a GeoJSON geometry on the fly. In real life, you would probably get these from another source. Note that MapRoulette requires the task geometry to be wrapped in a FeatureCollection, even if the geometry is just a single point, like in the example above.

You can also use `MapRouletteTaskCollection` to create multiple tasks at once.

CHAPTER 4

API

4.1 MapRoulette Server

4.2 MapRoulette Challenge

4.3 MapRoulette Task

4.4 A Task Collection