maproulette-python-client

Release 1.8.0

Jun 07, 2022

CONTENTS:

1 Getting Started **3**

2 Functionality **5**
- 2.1 Module: project . . . 5
- 2.2 Module: challenge . . . 7
- 2.3 Module: task . . . 11
- 2.4 Module: user . . . 14
- 2.5 Module: configuration . . . 15

3 Project Model **17**

4 Challenge Model **19**

5 Task Model **23**

6 Cooperative Work Model **25**

7 Parent Operation Model **27**

8 Child Operation Model **29**

9 Priority Rule Model **31**

10 Exceptions **33**

11 Indices and tables **35**

Python Module Index **37**

Index **39**

Welcome to MapRoulette's documentation!

CHAPTER

ONE

GETTING STARTED

Install the package (or add it to your requirements.txt file):

```
$ pip install maproulette
```

Import the package:

```
import maproulette
```

From there, create a configuration object. When creating the configuration you can specify a number of of parameters depending on your needs including the hostname, protocol, and client-side certificates. Depending on your use case, you may need to obtain and pass your API key as well. For example:

```
config = maproulette.Configuration(api_key='{YOUR_API_KEY}')
```

Once you have your configuration object we can create an API object using one of several modules depending on the functionality that the user is looking for. For example, creating a Project object allows the user to interact with all of the project-related functionality in the MapRoulette package.

```
api = maproulette.Project(config)
```

Now we have access to the MapRoulette Project API methods. In the example below, I want to find a project by name using a search string:

```
# We want to fetch a project with name 'Health Facilities in India'
my_project_name = 'Health Facilities in India'

# Pretty-print the API response
print(json.dumps(api.find_project(my_project_name), indent=4, sort_keys=True))
```

This returns a nicely printed JSON object representing the project named 'Health Facilities in India':


```
{
    "data": [
        {
            "created": "2019-08-26T06:34:28.655Z",
            "deleted": false,
            "description": "Adding the Hospitals ",
            "displayName": "Health Facilities in India",
            "enabled": true,
            "featured": false,
            "groups": [
                {
                    "created": "2020-03-25T16:23:04.360Z",
```


(continues on next page)

(continued from previous page)


```
                    "groupType": 1,
                    "id": 9273,
                    "modified": "2020-03-25T16:23:04.360Z",
                    "name": "4719_Admin",
                    "projectId": 4719
                },
                {
                    "created": "2020-03-25T16:23:04.360Z",
                    "groupType": 2,
                    "id": 9274,
                    "modified": "2020-03-25T16:23:04.360Z",
                    "name": "4719_Write",
                    "projectId": 4719
                },
                {
                    "created": "2020-03-25T16:23:04.360Z",
                    "groupType": 3,
                    "id": 9275,
                    "modified": "2020-03-25T16:23:04.360Z",
                    "name": "4719_Read",
                    "projectId": 4719
                }
            ],
            "id": 4719,
            "isVirtual": false,
            "modified": "2020-01-30T11:05:44.466Z",
            "name": "health_facilities_in_india",
            "owner": 4785024
        }
    ],
    "status": 200
}
```

CHAPTER

TWO

FUNCTIONALITY

For usage examples check out this link.

2.1 Module: project

This module contains the methods that the user will use directly to interact with MapRoulette projects

class `maproulette.api.project.`**`Project`**(*config*)

Bases: `maproulette.api.maproulette_server.MapRouletteServer`

Class to handle the project-related API requests

`add_challenge_to_project`(*project_id*, *challenge_id*)

Method to add a challenge to a virtual project.

Parameters

- **`project_id`** – the id of the virtual project
- **`challenge_id`** – the id of the challenge being added

Returns the API response from the POST request

`create_project`(*data*)

Method to create a new project

Parameters **`data`** – the data to use to create the new project

Returns the API response to the POST request

`delete_project`(*project_id*, *immediate='false'*)

Method to delete a project.

Parameters

- **`project_id`** – the id of the project being deleted
- **`immediate`** – whether or not the project should be deleted immediately

Returns the API response form the DELETE request

`find_project`(*matcher=''*, *limit=10*, *page=0*, *only_enabled='true'*)

Method to search for projects based on a specific search criteria

Parameters

- **`matcher`** – the search string used to match the project names. Default is empty string
- **`limit`** – the limit to the number of results returned in the response. Default is 10

- **page** – used in conjunction with the limit parameter to page through X number of responses. Default is 0.
- **only_enabled** – flag to set if only wanting enabled projects returned. Default is True.

Returns the API response from the GET request

get_project_by_id(*project_id*)
Method to fetch a project by unique MapRoulette project ID

Parameters **project_id** – the unique project ID

Returns the API response from the GET request

get_project_by_name(*project_name*)
Method to fetch a project by unique MapRoulette project name

Parameters **project_name** – the unique project name

Returns the API response from the GET request

get_project_challenges(*project_id*, *limit=10*, *page=0*)
Method to fetch a list of a project's challenges.

Parameters

- **project_id** – the id of the parent project
- **limit** – the limit to the number of results returned in the response. Default is 10
- **page** – used in conjunction with the limit parameter to page through X number of responses. Default is 0.

Returns the API response from the GET request

get_projects_by_ids(*project_ids*)
Method to retrieve projects from comma separated list of ids

Parameters **project_ids** – comma separated list of project ids to be retrieved

Returns the API response from the GET request

get_random_tasks(*project_id*, *limit=1*, *proximity=-1*, *search=''*)
Method to retrieve random tasks from a project.

Parameters

- **project_id** – the id of the parent project of tasks
- **limit** – limit amount of results returned
- **proximity** – task to find based on proximity of that task
- **search** – a search parameter object stored in a cookie

Returns the API response form the GET request

static is_project_model(*input_object*)
Method to determine whether user input is a valid project model

Parameters **input_object** – the user's input to check

Returns True if instance of model

remove_challenge_from_project(*project_id*, *challenge_id*)
Method to remove a challenge from a virtual project.

Parameters

- **project_id** – the id of the virtual project
- **challenge_id** – the id of the challenge being removed

Returns the API response from the POST request

update_project(*project_id*, *data*)
Method to update an existing project

Parameters

- **project_id** – the id of the project being updated
- **data** – the data to use to update the project

Returns the API response from the PUT request

2.2 Module: challenge

This module contains the methods that the user will use directly to interact with MapRoulette challenges

class maproulette.api.challenge.**Challenge**(*config*)
Bases: maproulette.api.maproulette_server.MapRouletteServer

Class to handle the challenge-related API requests

add_file_tasks_to_challenge(*data*, *challenge_id*, *line_by_line='true'*, *remove_unmatched='false'*, *data_origin_date=''*, *skip_snapshot='false'*)
Method to add tasks to an existing challenge with tasks as GeoJSON

Parameters

- **data** – a GeoJSON containing geometry of tasks to be added to a challenge
- **challenge_id** – the ID corresponding to the challenge that tasks will be added to
- **line_by_line** – whether or not the provided data is in line-by-line GeoJSON format
- **remove_unmatched** – whether or not the JSON provided includes seperate GeoJSON on each line
- **data_origin_date** – the date the data was sourced on
- **skip_snapshot** – whether or not to skip recording a snapshot before proceeding

Returns the API response from the PUT request

add_tasks_to_challenge(*data*, *challenge_id*)
Method to add tasks to an existing challenge

Parameters

- **data** – a GeoJSON containing geometry of tasks to be added to a challenge
- **challenge_id** – the ID corresponding to the challenge that tasks will be added to

Returns the API response from the PUT request

create_challenge(*data*)
Method to create a new challenge

Parameters **data** – a JSON input containing challenge details

Returns the API response to the POST request

create_virtual_challenge(*data*)
Method to create a new virtual challenge

Parameters **data** – a JSON input containing virtual challenge details

Returns the API response to the POST request

delete_challenge(*challenge_id*, *immediate='false'*)
Method to delete a challenge by using the corresponding challenge ID

Parameters

- **challenge_id** – the ID corresponding to the challenge
- **immediate** – whether or not the challenge should be deleted immediately

Returns the API response from the DELETE request

delete_challenge_tasks(*challenge_id*, *status_filters=''*)
Method to delete all existing tasks within a challenge, optionally filtering on current task status

Parameters

- **challenge_id** – the ID corresponding to the challenge
- **status_filters** – a comma separate list of status ID's: 0 = Created, 1 = Fixed, 2 = False Positive, 3 = Skipped, 4 = Deleted, 5 = Already Fixed, 6 = Too Hard

Returns the API response from the DELETE request

extract_challenge_comments(*challenge_id*, *limit=10*, *page=0*)
Method to retrieve all comments for the tasks in a given challenge and respond with a CSV

Parameters

- **challenge_id** – the ID corresponding to the challenge
- **limit** – the limit to the number of results returned in the response. Default is 10
- **page** – used in conjunction with the limit parameter to page through X number of responses. Default is 0.

Returns the API response from the GET request

extract_task_summaries(*challenge_id*, *limit=10*, *page=0*, *status=''*, *review_status=''*, *priority=''*, *export_properties=''*, *task_property_search=''*)
Method to retrieve summaries of all the tasks in a given challenge and respond with a CSV

Parameters

- **challenge_id** – the ID corresponding to the challenge
- **limit** – the limit to the number of results returned in the response. Default is 10
- **page** – used in conjunction with the limit parameter to page through X number of responses. Default is 0.
- **status** – a comma-separated filter for the tasks returned using tasks status values: 0 = Created, 1 = Fixed, 2 = False Positive, 3 = Skipped, 4 = Deleted, 5 = Already Fixed, 6 = Too Hard
- **review_status** – a comma-separated filter for the tasks returned using review status values: 0 - Requested, 1 - Approved, 2 - Rejected, 3 - Assisted, 4 - Disputed
- **priority** – a comma-separated filter for the tasks returned by priority value: 0 - High, 1 - Medium, 2 - Low

- **export_properties** – a comma-separated filter for the properties that should be exported
- **task_property_search** – a filter for the tasks returned using task properties

Returns the API response from the GET request

get_challenge_by_id(*challenge_id*)

Method to retrieve challenge information via the corresponding challenge ID

Parameters **challenge_id** – the ID corresponding to the challenge

Returns the API response from the GET request

get_challenge_by_name(*project_id*, *challenge_name*)

Method to retrieve challenge information via the corresponding challenge name and parent (project) ID

Parameters

- **project_id** – the ID of the parent project
- **challenge_name** – the name corresponding to the challenge

Returns the API response from the GET request

get_challenge_children(*challenge_id*, *limit=10*, *page=0*)

Method to retrieve all children from a given challenge in an expanded list. Unlike the `get_challenge_tasks()` method, this function will wrap the JSON array list inside of the parent challenge object allowing you to see the full hierarchy.

Parameters

- **challenge_id** – the ID corresponding to the challenge
- **limit** – the limit to the number of results returned in the response. Default is 10
- **page** – used in conjunction with the limit parameter to page through X number of responses. Default is 0.

Returns the API response from the GET request

get_challenge_comments(*challenge_id*, *limit=10*, *page=0*)

Method to retrieve all comments for the tasks in a given challenge

Parameters

- **challenge_id** – the ID corresponding to the challenge
- **limit** – the limit to the number of results returned in the response. Default is 10
- **page** – used in conjunction with the limit parameter to page through X number of responses. Default is 0.

Returns the API response from the GET request

get_challenge_geojson(*challenge_id*, *status=''*, *review_status=''*, *priority=''*, *task_property_search=''*)

Method to retrieve the GeoJSON for a challenge that represents all the task children of the challenge

Parameters

- **challenge_id** – the ID corresponding to the challenge
- **status** – a comma-separated filter for the tasks returned using tasks status values: 0 = Created, 1 = Fixed, 2 = False Positive, 3 = Skipped, 4 = Deleted, 5 = Already Fixed, 6 = Too Hard

- **review_status** – a comma-separated filter for the tasks returned using review status values: 0 - Requested, 1 - Approved, 2 - Rejected, 3 - Assisted, 4 - Disputed
- **priority** – a comma-separated filter for the tasks returned by priority value: 0 - High, 1 - Medium, 2 - Low
- **task_property_search** – a filter for the tasks returned using task properties

Returns the API response from the GET request

get_challenge_listing(*project_ids=''*, *limit=10*, *page=0*, *only_enabled='true'*)
Method to retrieve a lightweight list of challenges that belong to the specified project(s)

Parameters

- **project_ids** – a comma-separated list of project IDs for which child challenges are desired. Default is an empty string (i.e. all projects)
- **limit** – the limit to the number of results returned in the response. Default is 10
- **page** – used in conjunction with the limit parameter to page through X number of responses. Default is 0.
- **only_enabled** – whether or not results should be limited to only enabled challenges. Default is true.

Returns the API response from the GET request

get_challenge_statistics_by_id(*challenge_id*)
Method to retrieve statistics for a challenge using its corresponding ID

Parameters **challenge_id** – the ID corresponding to the challenge

Returns the API response to the GET request

get_challenge_tasks(*challenge_id*, *limit=10*, *page=0*)
Method to retrieve all tasks from a given challenge by ID

Parameters

- **challenge_id** – the ID corresponding to the challenge
- **limit** – the limit to the number of results returned in the response. Default is 10
- **page** – used in conjunction with the limit parameter to page through X number of responses. Default is 0.

Returns the API response from the GET request

get_challenges_by_tags(*challenge_tags*, *limit=10*, *page=0*)
Method to retrieve challenge information via tags applied to the challenge

Parameters

- **challenge_tags** – a comma-separated list of tags to search challenges for
- **limit** – the limit to the number of results returned in the response. Default is 10
- **page** – used in conjunction with the limit parameter to page through X number of responses. Default is 0.

Returns the API response from the GET request

get_virtual_challenge_by_id(*challenge_id*)
Method to retrieve an existing virtual challenge

Parameters **challenge_id** – the ID corresponding to the virtual challenge

Returns the API response from the GET request

static is_challenge_model(*input_object*)
Method to determine whether user input is a valid challenge model

Parameters **input_object** – the user's input to check

Returns True if instance of model

rebuild_challenge(*challenge_id*, *remove_unmatched=False*, *skip_snapshot=False*)
Rebuild the challenge by re-creating the tasks from the task data source

Parameters

- **challenge_id** – the ID corresponding to the challenge
- **remove_unmatched** – Used to remove incomplete tasks that have been addressed externally since the last rebuild, assuming the source data represents all tasks outstanding. If set to true, all existing tasks in CREATED or SKIPPED status (only) will be removed prior to rebuilding with the assumption that they will be recreated if they still appear in the updated source data. If set to false, unmatched existing tasks are simply left as-is. Default: False
- **skip_snapshot** – Whether to skip recording a snapshot before proceeding. Default: False

reset_task_instructions(*challenge_id*)
Method to reset all the task instructions so that they revert to the challenge instructions

Parameters **challenge_id** – the ID corresponding to the challenge

Returns the API response from the PUT request

update_challenge(*challenge_id*, *data*)
Method to update a challenge by using the corresponding challenge ID

Parameters

- **challenge_id** – the ID corresponding to the challenge
- **data** – a JSON input containing challenge details

Returns the API response from the PUT request

update_task_priorities(*challenge_id*)
Method to update all the task priorities for a given challenge based on the priority rules in the challenge

Parameters **challenge_id** – the ID corresponding to the challenge

Returns the API response from the PUT request

2.3 Module: task

This module contains the methods that the user will use directly to interact with MapRoulette tasks

class maproulette.api.task.**Task**(*config*)
Bases: maproulette.api.maproulette_server.MapRouletteServer

Class to handle the task-related API requests

static batch_generator(*input_list*, *chunk_size*)
Method to yield successive n-sized chunks from input_list

Parameters

- **input_list** – the list to break into chunks
- **chunk_size** (*int*) – the number of list items to include per chunk

Returns an iterator for the n-sized chunks of the input_list

create_task (*data*)
Method to create a single task using an input JSON of task details.

Parameters **data** – a JSON input containing task details

Returns the API response from the POST request

create_tasks (*data*, *batch_size=5000*)
Method to create a batch of tasks using the specified batch_size.

Parameters

- **data** – a JSON input containing task details
- **batch_size** (*int*) – the number of tasks to post per API call. The default is 5000.

Returns the API response from the POST request

delete_task_tags (*task_id*, *tags*)
Method to delete the supplied tags from a task using the corresponding task ID

Parameters

- **task_id** – the ID corresponding with the task
- **tags** – a comma-separated list of tags to be deleted

Returns the API response from the DELETE request

get_task_by_id (*task_id*)
"Method to retrieve task information using the corresponding task ID

Parameters **task_id** – the ID corresponding with the task

Returns the API response from the GET request

get_task_comments (*task_id*)
Method to retrieve the comments for a task using the corresponding task ID

Parameters **task_id** – the ID corresponding with the task

Returns the API response from the GET request

get_task_history (*task_id*)
Method to retrieve task history using the corresponding task ID

Parameters **task_id** – the ID corresponding with the task

Returns the API response from the GET request

get_task_tags (*task_id*)
Method to retrieve the tags for a task using the corresponding task ID

Parameters **task_id** – the ID corresponding with the task

Returns the API response from the GET request

get_tasks_by_bounding_box (*left*, *bottom*, *right*, *top*, *limit=10000*, *page=0*, *exclude_locked='false'*, *order='ASC'*, *include_total='false'*, *include_geometries='false'*, *include_tags='false'*)
Method to retrieve tasks by a bounding box defined by left, bottom, right, and top lat/long values

Parameters

- **left** – the minimum latitude of the bounding box
- **bottom** – the minimum longitude of the bounding box
- **right** – the maximum latitude of the bounding box
- **top** – the maximum longitude of the bounding box
- **limit** – the limit to the number of results returned in the response. Default is 10,000.
- **page** – used in conjunction with the limit parameter to page through X number of responses. Default is 0.
- **exclude_locked** – boolean indicating whether to exclude locked tasks. Default is 'false'.
- **order** – the order of the results. Default is 'ASC'
- **include_total** – whether to include total or not. Default is 'false'
- **include_geometries** – whether to include geometries or not. Default is 'false'
- **include_tags** – whether to include tags or not. Default is 'false'

Returns the API response from the PUT request

get_tasks_by_tags (*tags*, *limit=10*, *page=0*)

Method to retrieve tasks that have the specified tags

Parameters

- **tags** – a comma-separated list of tags to be searched for
- **limit** – the limit to the number of results returned in the response. Default is 10
- **page** – used in conjunction with the limit parameter to page through X number of responses. Default is 0.

Returns the API response from the GET request

static is_task_model (*input_object*)

Method to determine whether user input is a valid task model

Parameters **input_object** – the user's input to check

Returns True if instance of model

update_task_status (*task_id*, *status*, *comment=None*, *tags=None*, *request_review=None*, *completion_responses=None*)

Method to update a task's status to one of the following: 0 - Created, 1 - Fixed, 2 - False Positive, 3 - Skipped, 4 - Deleted, 5 - Already Fixed, 6 - Too Hard.

Parameters

- **task_id** – the ID corresponding with the task
- **status** – the status to update the task to
- **comment** – optional comment that is provided by the user when setting the status
- **tags** – optional tags to associate with this task
- **request_review** – optional boolean indicating if a review is requested on this task
- **completion_responses** – optional key/value JSON to be stored within this task

Returns the API response from the PUT request

update_task_tags (*task_id*, *tags*)
Method to update a task's tags using the supplied tags and corresponding task ID

Parameters

- **task_id** – the ID corresponding with the task
- **tags** – a comma-separated list of tags to be updated. If empty all tags will be removed.

Returns the API response from the GET request

update_tasks (*data*)
Method to update a batch of tasks

Parameters **data** – a JSON input containing task details

Returns the API response from the PUT request

2.4 Module: user

This module contains the methods that the user will use directly to interact with MapRoulette users

class maproulette.api.user.**User** (*config*)
Bases: maproulette.api.maproulette_server.MapRouletteServer

Class to handle the user-related API requests

add_user_list_to_project (*user_ids*, *project_id*, *group_type*, *is_osm_user_id='true'*)
Method to add a user to a project group

Parameters

- **user_ids** – a list of user IDs to add to the specified project group. IDs should be integers.
- **project_id** – the ID of the project
- **group_type** – the group type to add the user to (1 - Admin, 2 - Write, 3 - Read)
- **is_osm_user_id** – whether or not the specified user ID is an OSM user ID. Default is 'false'.

Returns the API response from the PUT request

add_user_to_project (*user_id*, *project_id*, *group_type*, *is_osm_user_id='true'*)
Method to add a user to a project group

Parameters

- **user_id** – the user ID to add to the specified project group
- **project_id** – the ID of the project
- **group_type** – the group type to add the user to (1 - Admin, 2 - Write, 3 - Read)
- **is_osm_user_id** – whether or not the specified user ID is an OSM user ID. Default is 'false'.

Returns the API response from the POST request

find_user_by_username (*username*, *limit=10*, *page=0*)
Method to search for a user based on a specific username

Parameters

- **username** – the username to search for.
- **limit** – the limit to the number of results returned in the response. Default is 10
- **page** – used in conjunction with the limit parameter to page through X number of responses. Default is 0.

Returns the API response from the GET request

2.5 Module: configuration

This module contains the basic configuration object that is used to communicate with the MapRoulette API.

class `maproulette.api.configuration.`**`Configuration`**(*hostname='maproulette.org', protocol='https', api_version='/api/v2', api_key=None, certs=None, verify=True*)

Configuration object that specifies how to connect to the MapRoulette server.

Parameters

- **hostname** (*str, optional*) – Optional parameter to specify the hostname of the MapRoulette instance being addressed. Do not include the protocol (https, http). Default value is 'maproulette.org'.
- **protocol** (*str, optional*) – Optional parameter to specify the protocol to use for the connection to the MapRoulette instance being addressed such as https or http. Default value is 'https'.
- **api_version** (*str, optional*) – Optional parameter to specify the API version to use. The default is '/api/v2'.
- **api_key** (*str, optional*) – Optional parameter to pass the user-specific API key. This key is necessary for some actions.
- **certs** (*str or tuple, optional*) – Optional parameter to pass the client-side certificate and key if necessary to make connection with the MapRoulette instance being addressed. Can be either a tuple containing the cert and key file paths (in that order) or a string representing the filepath to both the cert and key stored in a single file.
- **verify** (*bool, optional*) – Optional parameter to specify whether to verify SSL certificates for HTTPS requests. Default is True.

__init__(*hostname='maproulette.org', protocol='https', api_version='/api/v2', api_key=None, certs=None, verify=True*)

Initialize self. See help(type(self)) for accurate signature.

__weakref__

list of weak references to the object (if defined)

CHAPTER

THREE

PROJECT MODEL

This module contains the definition of a Project object in MapRoulette.

class `maproulette.models.project.`**`ProjectModel`**(*name*, *id=None*, *description=None*, *groups=None*, *enabled=None*, *is_virtual=None*, *display_name=None*, *featured=None*)

Bases: `object`

Definition for a MapRoulette Project

`READONLY = ['id']`

`property description`
The description for the project

`property display_name`
The friendly name that can be displayed to users

`property enabled`
Whether this project is enabled for use or not

`property featured`
Whether or not the project is featured

`property groups`
The groups that are associated with the project

`property id`
The ID of the project

`property is_virtual`
Whether or not a project is virtual

`property name`
The internal name of the project

`property path`
The path to the project

`to_dict`()
Converts all non-null properties of a project object into a dictionary

`to_json`()
Converts all non-null properties of a project object into a JSON object

CHAPTER

FOUR

CHALLENGE MODEL

This module contains the definition of a Challenge object in MapRoulette.

class `maproulette.models.challenge.`**`ChallengeModel`**(*name*, *id=None*, *description=None*, *parent=None*, *instruction=None*, *difficulty=None*, *blurb=None*, *enabled=None*, *challenge_type=None*, *featured=None*, *overpassQL=None*, *default_priority=None*, *high_priority_rule=None*, *medium_priority_rule=None*, *low_priority_rule=None*, *default_zoom=None*, *min_zoom=None*, *max_zoom=None*, *osm_id_property=None*, *cooperative_type=None*, *popularity=None*, *check_in_comment=None*, *check_in_source=None*, *requires_local=None*, *default_basemap=None*, *default_basemap_id=None*, *custom_basemap=None*, *update_tasks=None*, *exportable_properties=None*, *preferred_tags=None*, *task_styles=None*, *remote_geojson=None*, *keywords=None*)

Bases: `object`

Definition for a MapRoulette Challenge

property blurb
The blurb for the challenge

property challenge_type
The type for this challenge

property check_in_comment
Comment to be associated with changes made by users

property check_in_source
Hashtag appended to changeset comments

property cooperative_type

property custom_basemap
The custom basemap of this challenge

property default_basemap
The default basemap to use for this challenge

property default_basemap_id
The id of the default basemap

property default_priority
The default priority for this challenge

property default_zoom
The default zoom level for this challenge

property description
The description for the challenge

property difficulty
The difficulty setting for the challenge

property enabled
Whether this challenge is enabled for use or not

property exportable_properties
Comma separated list of properties to be exportable

property featured
Whether or not this challenge is featured

property high_priority_rule
The high priority for this challenge

property id
The ID of the challenge

property instruction
The instruction for the challenge

property keywords
A string or list of strings pertaining to the keywords of this challenge

property low_priority_rule
The low priority of this challenge

property max_zoom
The maximum zoom level for this challenge

property medium_priority_rule
The medium priority for this challenge

property min_zoom
The minimum zoom level for this challenge

property name
The internal name of the challenge

property osm_id_property
The id property of an osm feature

property overpassQL
The Overpass query for this challenge

property parent
The parent ID for the challenge

property path
The path to the challenge

property popularity
The popularity of a challenge

property preferred_tags
List of preferred tags the user can use when completing tasks

property remote_geojson
Create a challenge from a GeoJSON URL

property requires_local
Whether or not tasks require local knowledge to complete

property task_styles
Custom task styling based on specific task feature properties

to_dict()
Converts all non-null properties of a challenge object into a dictionary

to_json()
Converts all non-null properties of a challenge object into a JSON object

property update_tasks
Whether or not to periodically delete old tasks

CHAPTER

FIVE

TASK MODEL

This module contains the definition of a Task object in MapRoulette.

class `maproulette.models.task.`**`TaskModel`**(*name*, *parent*, *geometries*, *id=None*, *instruction=None*, *location=None*, *suggested_fix=None*, *status=None*, *mapped_on=None*, *review=None*, *priority=None*, *changeset_id=None*, *completion_responses=None*, *bundle_id=None*, *is_bundle_primary=None*, *mapillary_images=None*, *cooperative_work=None*)

Bases: `object`

Definition for a MapRoulette Task

`READONLY = ['id']`

`property bundle_id`
The bundle ID for this task

`property changeset_id`
The changeset ID for this task

`property completion_responses`
The completion response for this task

`property cooperative_work`
A dict containing cooperative work information which follows the cooperative_work model

`property geometries`
The geometries of the task

`property id`
The ID of the task

`property instruction`
The instruction for the task

`property is_bundle_primary`
Whether or not this task is the bundle primary

`property location`
The location of the task

`property mapillary_images`
The mapillary images for this task

`property mapped_on`
The mapped on date for the task

property name
The internal name of the task

property parent
The parent ID for the task

property path
The path to the task

property priority
The priority of this task

property review
Whether this task needs to be reviewed or not

property status
The status of the task

property suggested_fix
The suggested fix for the task

to_dict()
Converts all non-null properties of a task object into a dictionary

to_json()
Converts all non-null properties of a task object into a JSON object

CHAPTER

SIX

COOPERATIVE WORK MODEL

class maproulette.models.cooperative_work.**CooperativeWorkModel**(*version=2, type=None, parent_operations=None, file_type='xml', file_format='osc', encoding='base64', content=None*)

Bases: `object`

Definition for a Cooperative Work object

property content

A base64-encoded osc changefile to be used in type 2 cooperative work operations

property encoding

The type of encoding used in the changefile for type 2 cooperative work (currently, only base64 encoding is supported)

property file_format

The format of changefile to be used in type 2 cooperative work (currently, only osc files are supported

property file_type

The type of changefile to be used in type 2 cooperative work (currently, only xml files are supported

property parent_operations

A dict containing parent operation details which follows the parent_operation model

to_dict()

to_json()

Converts all non-null properties of a task object into a JSON object

property type

The type of cooperative work operation (either 1 for tag fix or 2 for change file) to be contained in the model

property version

The version of maproulette cooperative work format to be processed (currently, only version 2 is supported)

CHAPTER

SEVEN

PARENT OPERATION MODEL

class maproulette.models.parent_operation.**ParentOperationModel**(*operation_type=None*, *element_type=None*, *osm_id=None*, *child_operations=None*)

Bases: object

property child_operations

The set of operations that will be applied according to the operation type and object specified in the parent relation

property element_type

The element type of the object to which the operation is applied ('way', 'node', 'relation')

property operation_type

The operation type to be applied to the OSM object ('createElement', 'modifyElement', 'deleteElement')

property osm_id

The OSM ID of the object to which the operation is applied

to_dict()

to_json()

Converts all non-null properties of a task object into a JSON object

CHAPTER

EIGHT

CHILD OPERATION MODEL

class maproulette.models.child_operation.**ChildOperationModel**(*operation=None*, *data=None*)

Bases: object

property data

Either a dict (key/value pair) of a tag/tags to be set in a 'setTags' operation, or an array of tag keys (as strings) to be deleted in a deleteTags operation

property operation

The operation to be applied to the OSM object ('setTags', 'deleteTags')

to_dict()

to_json()

Converts all non-null properties of a child operation object into a JSON object

CHAPTER

NINE

PRIORITY RULE MODEL

class maproulette.models.priority_rule.**PriorityRule**(*priority_type*, *priority_operator*, *priority_value*)

Definition for a single priority rule

Parameters

- **priority_type** (*Types or str*) – the data type for the priority rule. The valid options are defined by the `Types` enum.
- **priority_operator** (*NumericOperators or StringOperators or str*) – the operator to use for the priority rule. The valid options for string-type priority rules are defined by the `StringOperators` enum and the valid options for numeric-type priority rules are defined by the `NumericOperators` enum.
- **priority_value** (*str*) – the value to use for the priority rule. This should be formatted like 'highway.footway' to indicate that any task with a 'highway' property equal to 'footway' should be considered for this rule. Multiple values can be specified using commas. Example: 'highway.footway,pedestrian'.

property priority_operator

The operator to use for the priority rule

property priority_type

The type for the priority rule

property priority_value

The value for the priority rule

to_dict()

Converts all properties of a priority rule object into a dictionary

to_json()

Converts all properties of a priority rule object into a JSON object

class maproulette.models.priority_rule.**PriorityRuleModel**(*condition*, *rules*)

A model for a priority rule definition in MapRoulette.

Parameters

- **condition** (*Conditions or str*) – the logical condition to use to string together multiple rules. The valid options for conditions are defined by the `Conditions` enum.
- **rules** (*PriorityRule or list*) – one or more rules to use for the priority rule definition. Rules should be instances of the *PriorityRule* class.

property condition

The condition to use to chain together multiple priority rules

property rules
The list of priority rules to be used in the priority rule model

to_dict()
Converts all properties of a priority rule model object into a dictionary

to_json()
Converts all properties of a priority rule model object into a JSON object

CHAPTER

TEN

EXCEPTIONS

exception maproulette.api.errors.**MapRouletteBaseException**(*message*, *status*, *payload*)

Bases: Exception

MapRoulette Base Exception

exception maproulette.api.errors.**NotFoundError**(*message=None*, *status=None*, *payload=None*)

Bases: *maproulette.api.errors.MapRouletteBaseException*

Resource cannot be found

exception maproulette.api.errors.**ConnectionUnavailableError**(*message=None*, *status=None*, *payload=None*)

Bases: *maproulette.api.errors.MapRouletteBaseException*

A connection error occurred

exception maproulette.api.errors.**UnauthorizedError**(*message=None*, *status=None*, *payload=None*)

Bases: *maproulette.api.errors.MapRouletteBaseException*

The user is not authorized to make this request

exception maproulette.api.errors.**HttpError**(*message=None*, *status=None*, *payload=None*)

Bases: *maproulette.api.errors.MapRouletteBaseException*

An HTTP error occurred

exception maproulette.api.errors.**InvalidJsonError**(*message=None*, *status=None*, *payload=None*)

Bases: *maproulette.api.errors.MapRouletteBaseException*

Errors produced from an invalid JSON object

This client makes it easy for users to communicate with the MapRoulette API from within their Python environment. In the example below, we are able to access a MapRoulette project in just four lines of code:

```
>>> import maproulette
>>> config = maproulette.Configuration()
>>> api = maproulette.Api(config)
>>> api.get_project_by_id(4719)
{'data': {'id': 4719, 'owner': 4785024, 'name': 'health_facilities_in_india',...}
```

CHAPTER

ELEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

m

maproulette.api.challenge, 7
maproulette.api.configuration, 15
maproulette.api.project, 5
maproulette.api.task, 11
maproulette.api.user, 14
maproulette.models.challenge, 19
maproulette.models.child_operation, 29
maproulette.models.cooperative_work, 25
maproulette.models.parent_operation, 27
maproulette.models.project, 17
maproulette.models.task, 23

INDEX

Symbols

__init__() (*maproulette.api.configuration.Configuration method*), 15
__weakref__ (*maproulette.api.configuration.Configuration attribute*), 15

A

add_challenge_to_project() (*maproulette.api.project.Project method*), 5
add_file_tasks_to_challenge() (*maproulette.api.challenge.Challenge method*), 7
add_tasks_to_challenge() (*maproulette.api.challenge.Challenge method*), 7
add_user_list_to_project() (*maproulette.api.user.User method*), 14
add_user_to_project() (*maproulette.api.user.User method*), 14

B

batch_generator() (*maproulette.api.task.Task static method*), 11
blurb() (*maproulette.models.challenge.ChallengeModel property*), 19
bundle_id() (*maproulette.models.task.TaskModel property*), 23

C

Challenge (*class in maproulette.api.challenge*), 7
challenge_type() (*maproulette.models.challenge.ChallengeModel property*), 19
ChallengeModel (*class in maproulette.models.challenge*), 19
changeset_id() (*maproulette.models.task.TaskModel property*), 23
check_in_comment() (*maproulette.models.challenge.ChallengeModel property*), 19
check_in_source() (*maproulette.models.challenge.ChallengeModel property*), 19
child_operations() (*maproulette.models.parent_operation.ParentOperationModel property*), 27
ChildOperationModel (*class in maproulette.models.child_operation*), 29
completion_responses() (*maproulette.models.task.TaskModel property*), 23
condition() (*maproulette.models.priority_rule.PriorityRuleModel property*), 31
Configuration (*class in maproulette.api.configuration*), 15
ConnectionUnavailableError, 33
content() (*maproulette.models.cooperative_work.CooperativeWorkMod property*), 25
cooperative_type() (*maproulette.models.challenge.ChallengeModel property*), 19
cooperative_work() (*maproulette.models.task.TaskModel property*), 23
CooperativeWorkModel (*class in maproulette.models.cooperative_work*), 25
create_challenge() (*maproulette.api.challenge.Challenge method*), 7
create_project() (*maproulette.api.project.Project method*), 5
create_task() (*maproulette.api.task.Task method*), 12
create_tasks() (*maproulette.api.task.Task method*), 12
create_virtual_challenge() (*maproulette.api.challenge.Challenge method*), 7
custom_basemap() (*maproulette.models.challenge.ChallengeModel property*), 20

D

data() (*maproulette.models.child_operation.ChildOperationModel property*), 29
default_basemap() (*maproulette.models.challenge.ChallengeModel property*), 20
default_basemap_id() (*maproulette.models.challenge.ChallengeModel property*), 20
default_priority() (*maproulette.models.challenge.ChallengeModel property*), 20
default_zoom() (*maproulette.models.challenge.ChallengeModel property*), 20
delete_challenge() (*maproulette.api.challenge.Challenge method*), 8
delete_challenge_tasks() (*maproulette.api.challenge.Challenge method*), 8
delete_project() (*maproulette.api.project.Project method*), 5
delete_task_tags() (*maproulette.api.task.Task method*), 12
description() (*maproulette.models.challenge.ChallengeModel property*), 20
description() (*maproulette.models.project.ProjectModel property*), 17
difficulty() (*maproulette.models.challenge.ChallengeModel property*), 20
display_name() (*maproulette.models.project.ProjectModel property*), 17

E

element_type() (*maproulette.models.parent_operation.ParentOperationModel property*), 27
enabled() (*maproulette.models.challenge.ChallengeModel property*), 20
enabled() (*maproulette.models.project.ProjectModel property*), 17
encoding() (*maproulette.models.cooperative_work.CooperativeWorkModel property*), 25
exportable_properties() (*maproulette.models.challenge.ChallengeModel property*), 20
extract_challenge_comments() (*maproulette.api.challenge.Challenge method*), 8
extract_task_summaries() (*maproulette.api.challenge.Challenge method*), 8

F

featured() (*maproulette.models.challenge.ChallengeModel property*), 20
featured() (*maproulette.models.project.ProjectModel property*), 17
file_format() (*maproulette.models.cooperative_work.CooperativeWo property*), 25
file_type() (*maproulette.models.cooperative_work.CooperativeWorkM property*), 25
find_project() (*maproulette.api.project.Project method*), 5
find_user_by_username() (*maproulette.api.user.User method*), 14

G

geometries() (*maproulette.models.task.TaskModel property*), 23
get_challenge_by_id() (*maproulette.api.challenge.Challenge method*), 9
get_challenge_by_name() (*maproulette.api.challenge.Challenge method*), 9
get_challenge_children() (*maproulette.api.challenge.Challenge method*), 9
get_challenge_comments() (*maproulette.api.challenge.Challenge method*), 9
get_challenge_geojson() (*maproulette.api.challenge.Challenge method*), 9
get_challenge_listing() (*maproulette.api.challenge.Challenge method*), 10
get_challenge_statistics_by_id() (*maproulette.api.challenge.Challenge method*), 10
get_challenge_tasks() (*maproulette.api.challenge.Challenge method*), 10
get_challenges_by_tags() (*maproulette.api.challenge.Challenge method*), 10
get_project_by_id() (*maproulette.api.project.Project method*), 6
get_project_by_name() (*maproulette.api.project.Project method*), 6
get_project_challenges() (*maproulette.api.project.Project method*), 6
get_projects_by_ids() (*maproulette.api.project.Project method*), 6

get_random_tasks() (*maproulette.api.project.Project method*), 6
get_task_by_id() (*maproulette.api.task.Task method*), 12
get_task_comments() (*maproulette.api.task.Task method*), 12
get_task_history() (*maproulette.api.task.Task method*), 12
get_task_tags() (*maproulette.api.task.Task method*), 12
get_tasks_by_bounding_box() (*maproulette.api.task.Task method*), 12
get_tasks_by_tags() (*maproulette.api.task.Task method*), 13
get_virtual_challenge_by_id() (*maproulette.api.challenge.Challenge method*), 10
groups() (*maproulette.models.project.ProjectModel property*), 17

H

high_priority_rule() (*maproulette.models.challenge.ChallengeModel property*), 20
HttpError, 33

I

id() (*maproulette.models.challenge.ChallengeModel property*), 20
id() (*maproulette.models.project.ProjectModel property*), 17
id() (*maproulette.models.task.TaskModel property*), 23
instruction() (*maproulette.models.challenge.ChallengeModel property*), 20
instruction() (*maproulette.models.task.TaskModel property*), 23
InvalidJsonError, 33
is_bundle_primary() (*maproulette.models.task.TaskModel property*), 23
is_challenge_model() (*maproulette.api.challenge.Challenge static method*), 11
is_project_model() (*maproulette.api.project.Project static method*), 6
is_task_model() (*maproulette.api.task.Task static method*), 13
is_virtual() (*maproulette.models.project.ProjectModel property*), 17

K

keywords() (*maproulette.models.challenge.ChallengeModel property*), 20

L

location() (*maproulette.models.task.TaskModel property*), 23
low_priority_rule() (*maproulette.models.challenge.ChallengeModel property*), 20

M

mapillary_images() (*maproulette.models.task.TaskModel property*), 23
mapped_on() (*maproulette.models.task.TaskModel property*), 23
maproulette.api.challenge (*module*), 7
maproulette.api.configuration (*module*), 15
maproulette.api.project (*module*), 5
maproulette.api.task (*module*), 11
maproulette.api.user (*module*), 14
maproulette.models.challenge (*module*), 19
maproulette.models.child_operation (*module*), 29
maproulette.models.cooperative_work (*module*), 25
maproulette.models.parent_operation (*module*), 27
maproulette.models.project (*module*), 17
maproulette.models.task (*module*), 23
MapRouletteBaseException, 33
max_zoom() (*maproulette.models.challenge.ChallengeModel property*), 20
medium_priority_rule() (*maproulette.models.challenge.ChallengeModel property*), 20
min_zoom() (*maproulette.models.challenge.ChallengeModel property*), 20

N

name() (*maproulette.models.challenge.ChallengeModel property*), 20
name() (*maproulette.models.project.ProjectModel property*), 17
name() (*maproulette.models.task.TaskModel property*), 23
NotFoundError, 33

O

operation() (*maproulette.models.child_operation.ChildOperationMode property*), 29
operation_type() (*maproulette.models.parent_operation.ParentOpera property*), 27
osm_id() (*maproulette.models.parent_operation.ParentOperationModel property*), 27

osm_id_property() (*maproulette.models.challenge.ChallengeModel property*), 20
overpassQL() (*maproulette.models.challenge.ChallengeModel property*), 20

P

parent() (*maproulette.models.challenge.ChallengeModel property*), 20
parent() (*maproulette.models.task.TaskModel property*), 24
parent_operations() (*maproulette.models.cooperative_work.CooperativeWorkModel property*), 25
ParentOperationModel (*class in maproulette.models.parent_operation*), 27
path() (*maproulette.models.challenge.ChallengeModel property*), 21
path() (*maproulette.models.project.ProjectModel property*), 17
path() (*maproulette.models.task.TaskModel property*), 24
popularity() (*maproulette.models.challenge.ChallengeModel property*), 21
preferred_tags() (*maproulette.models.challenge.ChallengeModel property*), 21
priority() (*maproulette.models.task.TaskModel property*), 24
priority_operator() (*maproulette.models.priority_rule.PriorityRule property*), 31
priority_type() (*maproulette.models.priority_rule.PriorityRule property*), 31
priority_value() (*maproulette.models.priority_rule.PriorityRule property*), 31
PriorityRule (*class in maproulette.models.priority_rule*), 31
PriorityRuleModel (*class in maproulette.models.priority_rule*), 31
Project (*class in maproulette.api.project*), 5
ProjectModel (*class in maproulette.models.project*), 17

R

READONLY (*maproulette.models.project.ProjectModel attribute*), 17
READONLY (*maproulette.models.task.TaskModel attribute*), 23
rebuild_challenge() (*maproulette.api.challenge.Challenge method*), 11
remote_geojson() (*maproulette.models.challenge.ChallengeModel property*), 21
remove_challenge_from_project() (*maproulette.api.project.Project method*), 6
requires_local() (*maproulette.models.challenge.ChallengeModel property*), 21
reset_task_instructions() (*maproulette.api.challenge.Challenge method*), 11
review() (*maproulette.models.task.TaskModel property*), 24
rules() (*maproulette.models.priority_rule.PriorityRuleModel property*), 31

S

status() (*maproulette.models.task.TaskModel property*), 24
suggested_fix() (*maproulette.models.task.TaskModel property*), 24

T

Task (*class in maproulette.api.task*), 11
task_styles() (*maproulette.models.challenge.ChallengeModel property*), 21
TaskModel (*class in maproulette.models.task*), 23
to_dict() (*maproulette.models.challenge.ChallengeModel method*), 21
to_dict() (*maproulette.models.child_operation.ChildOperationModel method*), 29
to_dict() (*maproulette.models.cooperative_work.CooperativeWorkModel method*), 25
to_dict() (*maproulette.models.parent_operation.ParentOperationModel method*), 27
to_dict() (*maproulette.models.priority_rule.PriorityRule method*), 31
to_dict() (*maproulette.models.priority_rule.PriorityRuleModel method*), 32
to_dict() (*maproulette.models.project.ProjectModel method*), 17
to_dict() (*maproulette.models.task.TaskModel method*), 24
to_json() (*maproulette.models.challenge.ChallengeModel method*), 21
to_json() (*maproulette.models.child_operation.ChildOperationModel method*), 29
to_json() (*maproulette.models.cooperative_work.CooperativeWorkModel method*), 25
to_json() (*maproulette.models.parent_operation.ParentOperationModel method*), 27
to_json() (*maproulette.models.priority_rule.PriorityRule method*), 31
to_json() (*maproulette.models.priority_rule.PriorityRuleModel method*), 32
to_json() (*maproulette.models.project.ProjectModel method*), 17

`to_json()` (*maproulette.models.task.TaskModel method*), 24
`type()` (*maproulette.models.cooperative_work.CooperativeWorkModel property*), 25

U

`UnauthorizedError`, 33
`update_challenge()` (*maproulette.api.challenge.Challenge method*), 11
`update_project()` (*maproulette.api.project.Project method*), 7
`update_task_priorities()` (*maproulette.api.challenge.Challenge method*), 11
`update_task_status()` (*maproulette.api.task.Task method*), 13
`update_task_tags()` (*maproulette.api.task.Task method*), 13
`update_tasks()` (*maproulette.api.task.Task method*), 14
`update_tasks()` (*maproulette.models.challenge.ChallengeModel property*), 21
`User` (*class in maproulette.api.user*), 14

V

`version()` (*maproulette.models.cooperative_work.CooperativeWorkModel property*), 25