Snakemine: Redmine API wrapper Documentation

Release 1.0b1

Mark Lee

Sep 27, 2017

Contents

This library is a REST API wrapper for the Redmine project management web application. Its API is directly inspired by Django's settings module and ORM.

CHAPTER 1

Installation

Install-time requirements:

- CPython 2.6 / 2.7 / 3.3 or PyPy (although PyPy is not currently tested on Travis CI due to a strange bug, it is tested regularly via tox)
- If you're using Python 2.6:
  - argparse
  - importlib
- lxml
- python-dateutil
- requests

Run-time requirements:

- A Redmine installation somewhere (this has only been tested with 1.0, I make no guarantees at this point that it works with any newer version)

CHAPTER 2

Example

Let's assume that you have a registered user with an API key.

Here is the Django-esque settings file (named `redmine_settings.py`):

```
BASE_URI = 'https://redmine.example.com'
USERNAME = 'jschmidt'
API_KEY = '1234abcd'
```

Here is a way to retrieve a given issue and change its subject (creatively named `script.py`):

```
from snakemine.issue import Issue

issue = Issue.objects.get(12)
issue.subject = 'Modified subject'
issue.save()
```

You would run the script like so:

```
PYTHONPATH=. SNAKEMINE_SETTINGS_MODULE=redmine_settings python script.py
```

CHAPTER 3

License

The code is licensed under the Apache License 2.0; see the `LICENSE` file for details.

The documentation is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License; see the `LICENSE.docs` file for details.

CHAPTER 4

Contributing

This project is hosted at GitHub. I gladly accept pull requests. If you contribute code, please also create tests for your modifications, otherwise your request will not be accepted as quickly (I will most likely ask you to add tests). It would probably also be in your best interests to add yourself to the `AUTHORS.rst` file if you have not done so already.

A Vagrant environment is available for developing snakemine. Simply run the following command (once Vagrant is installed):

```
$ vagrant up
```

...and it will install all of the Python dependencies in a virtualenv, plus set up a compatible Ruby environment for Redmine. You can then log into the virtual machine:

```
$ vagrant ssh
vagrant@vagrant $ source .virtualenv/bin/activate
vagrant@vagrant $ git clone /vagrant ~/snakemine
```

The last line exists so that it is possible to run `python setup.py sdist` - VirtualBox's remote filesystem module does not support hardlinks, so it fails if you try to run that command or `tox` from the `/vagrant` directory.

CHAPTER 5

Contributors

- Mark Lee

CHAPTER 6

API Documentation

Package: `snakemine`

A Pythonic interface to Redmine's REST API.

Module author: Mark Lee <snakemine@lazymalevolence.com>

Subpackage

Snakemine configuration (`snakemine.conf`)

A settings manager inspired by Django's.

There are two ways to set user-specific settings:

1. Create a module with all of the settings in it and point snakemine to the module via the `SNAKEMINE_SETTINGS_MODULE` environment variable.
2. Use *`Settings.configure`* to set the values via its keyword arguments.

Module author: Mark Lee <snakemine@lazymalevolence.com>

class `snakemine.conf.`**`Settings`**(*default_settings*)

Bases: `object`

Settings object for *`snakemine`*.

Parameters **`default_settings`** (*`dict`*) – The default settings. A copy of this parameter is stored in the created object.

`configure`(*default_settings=None*, ***kwargs*)

Manually configures the user settings instead of via the import module method. This method can only be called if the object has not already had user settings set.

Parameters **`default_settings`** (*`dict`*) – The default settings, which will override the default settings passed in at object creation.

configured
Whether the user settings have been set for the object.

Return type `bool`

Base functionality (`snakemine.base`)

Base for Redmine models.

Module author: Mark Lee <snakemine@lazymalevolence.com>

class `snakemine.base.`**`Manager`**
Bases: `object`

A Django-like model manager for Redmine resources.

`all`()
Retrieves all of the available items for a given resource.

Return type `list` of *`Resource`*

`create`(*data*)
Creates a new *`Resource`*-derived object.

Parameters **`data`** (*`dict`*) – The new resource metadata

Return type *`Resource`*

`delete`(*resource_id*)

`filter`(***kwargs*)

`get`(*resource_id*)
Retrieves a single item for a given resource and ID.

Parameters **`resource_id`** (*`int`*) – The resource's ID

Return type *`Resource`*

`update`(*resource_id, data*)

class `snakemine.base.`**`Resource`**(*response*)
Bases: `object`

An abstract representation of a Redmine resource.

Much like a Django model, a `Resource` typically has a class variable named `objects`, which is an instantiation of the related *`Manager`*.

Parameters **`response`** – the object the represents the metadata of the resource item

`delete`()
Deletes the resource item from Redmine.

`save`()
Creates or updates the resource item in Redmine.

Issue management (`snakemine.issue`)

Handling of Redmine issues.

Module author: Mark Lee <snakemine@lazymalevolence.com>

class `snakemine.issue.`**`Issue`**`(`*response*`)`
Bases: *`snakemine.base.Resource`*

A representation of a Redmine issue.

`objects` **= <snakemine.issue.IssueManager object>**

`parent`
The parent issue, if specified.

Return type *`Issue`* or `None`

`project`
The `Project` associated with the issue.

class `snakemine.issue.`**`IssueManager`**
Bases: *`snakemine.base.Manager`*

A Django-like model manager for Redmine issues.

Project management (`snakemine.project`)

Handling of Redmine projects.

Module author: Mark Lee <snakemine@lazymalevolence.com>

class `snakemine.project.`**`Project`**`(`*response*`)`
Bases: *`snakemine.base.Resource`*

A representation of a Redmine project.

`objects` **= <snakemine.project.ProjectManager object>**

class `snakemine.project.`**`ProjectManager`**
Bases: *`snakemine.base.Manager`*

A Django-like model manager for Redmine projects.

Python Module Index

Index