
rmoq Documentation

Release 1.0.0

Rolf Erik Lekang

February 01, 2015

1	Installation	3
2	Usage	5
3	The Mock class	7
4	Backends	9

A simple request mocker that caches requests responses to files.

Installation

Install it with pip:

```
pip install rmoq
```

Usage

Function decorator

The example below will put the content of fixtures/example.com.txt into the body of the request and if it does not exist the content will be downloaded and stored in fixtures/example.com.txt.

```
@rmoq.activate()
def test_remote_call():
    response = requests.get('http://example.com')
    assert response.body == 'Example'
```

The example below works as the one above it just uses the given path (test_fixtures) instead of the default path.

```
@rmoq.activate('test_fixtures')
def test_remote_call():
    response = requests.get('http://example.com')
    assert response.body == 'Example'
```

With statements

It can also be used in a with statement

```
def test_remote_call():
    with rmoq.Mock():
        response = requests.get('http://example.com')
        assert response.body == 'Example'
```

The mock object can also take a path as an argument.

Class decorator

The decorator will also work for classes, which means you can decorate a whole test-case:

```
@rmoq.activate()
class RemoteTestCase(unittest.TestCase):
    def test_remote_call():
        response = requests.get('http://example.com')
        assert response.body == 'Example'
```

Disable in one test run Setting the environment variable *RMOQ_DISABLED* to *True* will disable rmoq:

```
$ RMOQ_DISABLED=True py.test
```

This can be useful to make sure that the a CI server does not use time on saving new fixtures if you are only using fixtures locally.

The Mock class

```
class rmoq.Mock (prefix='fixtures', backend=<rmoq.backends.FileStorageBackend object at 0x7f85bfdfff10>)
```

The mocker class that mocks requests in rmoq. It supports being used in with-statements and have a method **:method:'activate'** that can be used as a decorator on functions or classes.

```
activate (prefix=None, backend=None)
```

A decorator used to activate the mocker.

Parameters

- **prefix** –
- **backend** – An instance of a storage backend.

Backends

rmoq supports custom storage backends by passing an instance into the backend parameter of `rmoq.activate()` method. A storage backend must inherit from `rmoq.RmoqStorageBackend` and implement the `get()` and `put()` methods.

class `rmoq.RmoqStorageBackend`

Base backend for rmoq backends. All storage backends for rmoq must inherit this backend if it is to be used with rmoq.

static `clean_url(url, replacement='_')`

Cleans the url for protocol prefix and trailing slash and replaces special characters with the given replacement.

Parameters

- **url** – The url of the request.
- **replacement** – A string that is used to replace special characters.

`get(prefix, url)`

Fetches a request response from storage. Should be overridden by subclasses.

Parameters

- **prefix** – A prefix that separates containers of request responses in the storage.
- **url** – The url of the request.

`put(prefix, url, content, content_type)`

Writes a request response in to storage. Should be overridden by subclasses.

Parameters

- **prefix** – A prefix that separates containers of request responses in the storage.
- **url** – The url of the request.
- **content** – The content of the request response.
- **content_type** – The content type header of the request response.

class `rmoq.FileStorageBackend`

Bases: `rmoq.backends.RmoqStorageBackend`

A rmoq backend that reads and writes to the local file system. This is the default backend.

`get(prefix, url)`

`get_filename(prefix, url)`

Creates a file path on the form: `current-working-directory/prefix/cleaned-url.txt`

Parameters

- **prefix** – The prefix from the `.get()` and `.put()` methods.
- **url** – The url of the request.

Returns The created path.

put (*prefix, url, content, content_type*)

class `rmoq.MemcachedStorageBackend` (*servers, **options*)

Bases: `rmoq.backends.RmoqStorageBackend`

Storage backend for rmoq that uses memcached for storage. Takes a the same arguments as python-memcached: a list of servers and options as keyword arguments.

create_key (**parts*)

get (*prefix, url*)

put (*prefix, url, content, content_type*)

A

activate() (rmoq.Mock method), 7

C

clean_url() (rmoq.RmoqStorageBackend static method),
9

create_key() (rmoq.MemcachedStorageBackend
method), 10

F

FileStorageBackend (class in rmoq), 9

G

get() (rmoq.FileStorageBackend method), 9

get() (rmoq.MemcachedStorageBackend method), 10

get() (rmoq.RmoqStorageBackend method), 9

get_filename() (rmoq.FileStorageBackend method), 9

M

MemcachedStorageBackend (class in rmoq), 10

Mock (class in rmoq), 7

P

put() (rmoq.FileStorageBackend method), 10

put() (rmoq.MemcachedStorageBackend method), 10

put() (rmoq.RmoqStorageBackend method), 9

R

RmoqStorageBackend (class in rmoq), 9