
Cassette Documentation

Release 0.3.8

Charles-Axel Dein

April 03, 2015

1	User's Guide	3
1.1	Foreword	3
1.2	Quickstart	3
1.3	Usage	4
2	API Reference	7
2.1	API	7
3	Additional Notes	11
3.1	Development	11
3.2	Changelog for Cassette	11
3.3	License	13
4	Indices and tables	15
	Python Module Index	17

Cassette stores and replays HTTP requests made in your Python app.

Latest documentation: cassette.readthedocs.org/en/latest/

1.1 Foreword

1.1.1 Similar libraries

Cassette took a lot inspiration from the following packages:

- `vcrapy`: HTTP recording and playback library
- `vcr`: in Ruby

1.1.2 Limitations

Cassette should be considered **alpha**:

- Only tested with `urllib2` and `httplib`
- Does not work with `urllib` and `requests`
- The format used is not compatible with `vcr` or `vcrapy`
- Only tested with Python 2.7
- File format **WILL** change.

1.2 Quickstart

- The first time you run your tests, `cassette` will store all the external requests response in a YAML file.
- Next time you run your tests, `cassette` will fetch those responses from the YAML file. You can now run your tests while being offline.

```
import urllib2
```

```
import cassette
```

```
with cassette.play("data/responses.yaml"):
```

```
    # If the request is not already stored in responses.yaml, cassette
    # will request the URL and store its response in the file.
    r = urllib2.urlopen("http://www.internic.net/domain/named.root")
```

```
# This time, the request response must be in the file. The external
# request is not made. cassette retrieves the response from the
# file.
r = urllib2.urlopen("http://www.internic.net/domain/named.root")

assert "A.ROOT-SERVERS.NET" in r.read(10000)
```

1.3 Usage

cassette provides a play context:

```
import cassette

with cassette.play("./data/responses.yaml"):
    urllib2.urlopen("http://...")
```

You can also setup the context manually:

```
import cassette

cassette.insert("./data/responses.yaml")
urllib2.urlopen("http://...")
cassette.eject()
```

1.3.1 Storage backend

New in version 0.3.1: Ability to read from a directory.

New in version 0.3.1: Ability to read from JSON files.

cassette supports multiple storage backend:

- File based (all the requests and responses are in the same file)
- Directory based (each request/response is in a single file)

Two formats are supported, JSON (faster) and YAML.

To read from a directory, just provide the path:

```
cassette.insert("./data/", file_format="json")
```

1.3.2 Report which cassettes are not used

New in version 0.3.7: Ability to report which cassettes are not used.

Here's a way to do it:

```
import urllib2
from cStringIO import StringIO

from cassette.player import Player

def test_report_unused():
```



```
config = {'log_cassette_used': True}
player = Player('./cassette/tests/data/requests/', config=config)
with player.play():
    urllib2.urlopen('http://httpbin.org/get')

content = StringIO()
player.report_unused_cassettes(content)
content.seek(0)
expected = ('httplib_GET_httpbin.org_80__unused.json\n'
            'httplib_GET_httpbin.org_80__unused2.json')
assert content.read() == expected
```

Here's how you would use `teardown` with `pytest` to log those unused cassettes to a file:

```
@pytest.fixture(scope="session", autouse=True)
def report_unused_cassette(request):
    """Report unused cassettes."""
    def func():
        with open('.unused_cassette.log', 'w') as f:
            cassette_player.report_unused_cassettes(f)
    request.addfinalizer(func)
```

You would then `cd` to the directory containing the fixtures, and run:

```
$ xargs rm < ../../../../.unused_cassette.log
```

API Reference

2.1 API

2.1.1 `cassette` module

`cassette.eject` (*exc_type=None, exc_value=None, tb=None*)

Remove cassette, unpatching HTTP requests.

`cassette.insert` (*filename, file_format=''*)

Setup cassette.

Parameters `filename` – path to where requests and responses will be stored.

`cassette.play` (**args, **kws*)

Use cassette.

2.1.2 `cassette_library` module

class `cassette.cassette_library.CassetteLibrary` (*filename, encoder, config=None*)

The `CassetteLibrary` holds the stored requests and manage them.

This is an abstract class that needs to have several methods implemented. In addition, subclasses must store request/response pairs as a keys and values as a dictionary in the property `self.data`

Parameters

- **filename** (*str*) – filename to use when storing and replaying requests.
- **encoder** (*Encoder*) – the instantiated encoder to use

add_response (*cassette_name, response*)

Add a new response to the mocked response.

Parameters

- **cassette_name** (*str*) –
- **response** –

cassette_name_for_httplib_connection (*host, port, method, url, body, headers*)

Create a cassette name from an httplib request.

classmethod **create_new_cassette_library** (*path, file_format, config=None*)

Return an instantiated `CassetteLibrary`.

Use this method to create new a `CassetteLibrary`. It will automatically determine if it should use a file or directory to back the cassette based on the filename. The method assumes that all file names with an extension (e.g. `/file.json`) are files, and all file names without extensions are directories (e.g. `/requests`).

Parameters

- **path** (*str*) – filename of file or directory for storing requests
- **file_format** (*str*) – the file_format to use for storing requests
- **config** (*dict*) – configuration

log_cassette_used (*path*)

Log that a path was used.

report_unused_cassettes (*output=<open file '<stdout>', mode 'w' at 0x7fb73b6dd150>*)

Report unused path to a file.

rewind ()

Restore all responses to a re-seekable state.

save_to_cache (*file_hash, data*)

Save a decoded data object into cache.

write_to_file ()

Write the response data to file.

class `cassette.cassette_library.CassetteName`

A `CassetteName` represents an unique way to retrieve the cassette from the library.

classmethod **from_httplib_connection** (*host, port, method, url, body, headers, will_hash_body=False*)

Create an object from an httplib request.

class `cassette.cassette_library.DirectoryCassetteLibrary` (**args, **kwargs*)

A `CassetteLibrary` that stores and manages requests with directory.

cassette_name_for_httplib_connection (*host, port, method, url, body, headers*)

Create a cassette name from an httplib request.

generate_filename (*cassette_name*)

Generate the filename for a given cassette name.

generate_path_from_cassette_name (*cassette_name*)

Generate the full path to cassette file.

get_all_available ()

Return all available cassette.

write_to_file ()

Write mocked response to a directory of files.

class `cassette.cassette_library.FileCassetteLibrary` (*filename, encoder, config=None*)

Store and manage requests with a single file.

data

Lazily loaded data.

get_all_available ()

Return all available cassette.

load_file ()

Load MockedResponses from YAML file.

`write_to_file()`

Write mocked responses to file.

2.1.3 patcher module

`cassette.patcher.patch(cassette_library)`

Replace standard library.

`cassette.patcher.unpatch()`

Unpatch standard library.

2.1.4 unpatched module

`cassette.unpatched.unpatched_httplib_context(*args, **kws)`

Create a context in which httplib is unpatched.

Additional Notes

3.1 Development

3.1.1 Running cassette tests

The quick and dirty way to run tests is through `setup.py`:

```
$ python setup.py test
```

For more involved development, you can create a virtual environment, install `fabric` (`pip install fabric`) then install requirements:

```
$ fab bootstrap
```

Start the test server and run tests:

```
$ fab test
```

Tests spin up a test server to bounce requests off of, but you can run this server manually:

```
$ fab serve_test_server
```

You will see an error about the test server's address being in use, but this is harmless.

3.2 Changelog for Cassette

3.2.1 0.3.8 (2015-04-03)

- Add compatibility with Python 2.7.9

3.2.2 0.3.7 (2015-03-12)

- Add ability to report on unused cassette.

3.2.3 0.3.6 (2014-10-31)

- Fix `NameError` when using `UL3CassetteHTTPConnection` (thanks to @carolinevdh)

- Fix HTTP Response to use cStringIO, adding Unicode support (thanks to @carolinevdh)

3.2.4 0.3.5 (2014-08-28)

- Fix error closing HTTPConnections directly

3.2.5 0.3.4 (2014-08-27)

- Improve backward compatibility with 0.3.2

3.2.6 0.3.3 (2014-08-27)

- Added support for *requests*. Note that libraries are not necessarily cross compatible-requests cached with *urllib2* may not work with *requests* and vice versa.

3.2.7 0.3.2 (2014-06-26)

- Handle absent headers with httplib (thanks to @blampe)

3.2.8 0.3.1 (2014-06-04)

- Add the ability to read from a directory instead of from a single file (thanks to @anthonysutardja)

3.2.9 0.3 (2014-03-18)

- Respect request headers in cassette name. Requires regenerating cassette files.

3.2.10 0.2 (2013-05-14)

- Get rid of urlopen mocking, mock only at httplib level to circumvent the problem with urlopen raising exceptions when getting non-2XX codes
- Clean up the docs, streamline their structure
- **This is a backward incompatible release**, you'll need to delete your YAML file.

3.2.11 0.1.13 (2013-05-13)

- Fix binary file downloading (thanks to @twolfson)

3.2.12 0.1.12 (2013-04-26)

- Add performance tests (courtesy of @twolfson)
- Cache the loaded file content to achieve significant performance improvements (thanks to @twolfson)

3.2.13 0.1.11 (2013-04-11)

- Lazily load YAML file

3.2.14 0.1.11 (2013-04-11)

- Started tracking changes

3.3 License

Cassette is licensed under a three clause BSD License.

The full license text can be found below (*Cassette License*).

3.3.1 Authors

Cassette is written and maintained by Charles-Axel Dein and various contributors:

Development Lead

- Charles-Axel Dein <charles@uber.com>

Patches and Suggestions

- Todd Wolfson <todd@uber.com>
- Zack Heller <zack@uber.com>

3.3.2 Cassette License

Copyright (c) 2013 by Uber Technologies, Inc. and contributors. See AUTHORS.txt for more details.

All rights reserved.

Redistribution and use in source and binary forms of the software as well as documentation, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- The names of the contributors may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE AND DOCUMENTATION IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF

USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE AND DOCUMENTATION, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Indices and tables

- *genindex*
- *modindex*
- *search*

C

`cassette`, [7](#)
`cassette.cassette_library`, [7](#)
`cassette.patcher`, [9](#)
`cassette.unpatched`, [9](#)

A

`add_response()` (`cassette.cassette_library.CassetteLibrary` method), 7

C

`cassette` (module), 7

`cassette.cassette_library` (module), 7

`cassette.patcher` (module), 9

`cassette.unpatched` (module), 9

`cassette_name_for_httplib_connection()` (`cassette.cassette_library.CassetteLibrary` method), 7

`cassette_name_for_httplib_connection()` (`cassette.cassette_library.DirectoryCassetteLibrary` method), 8

`CassetteLibrary` (class in `cassette.cassette_library`), 7

`CassetteName` (class in `cassette.cassette_library`), 8

`create_new_cassette_library()` (`cassette.cassette_library.CassetteLibrary` class method), 7

D

`data` (`cassette.cassette_library.FileCassetteLibrary` attribute), 8

`DirectoryCassetteLibrary` (class in `cassette.cassette_library`), 8

E

`eject()` (in module `cassette`), 7

F

`FileCassetteLibrary` (class in `cassette.cassette_library`), 8

`from_httplib_connection()` (`cassette.cassette_library.CassetteName` class method), 8

G

`generate_filename()` (`cassette.cassette_library.DirectoryCassetteLibrary` method), 8

`generate_path_from_cassette_name()` (`cassette.cassette_library.DirectoryCassetteLibrary` method), 8

`get_all_available()` (`cassette.cassette_library.DirectoryCassetteLibrary` method), 8

`get_all_available()` (`cassette.cassette_library.FileCassetteLibrary` method), 8

I

`insert()` (in module `cassette`), 7

L

`load_file()` (`cassette.cassette_library.FileCassetteLibrary` method), 8

`log_cassette_used()` (`cassette.cassette_library.CassetteLibrary` method), 8

P

`patch()` (in module `cassette.patcher`), 9

`play()` (in module `cassette`), 7

R

`report_unused_cassettes()` (`cassette.cassette_library.CassetteLibrary` method), 8

`rewind()` (`cassette.cassette_library.CassetteLibrary` method), 8

S

`save_to_cache()` (`cassette.cassette_library.CassetteLibrary` method), 8

U

`unpatch()` (in module `cassette.patcher`), 9

`unpatched_httplib_context()` (in module `cassette.unpatched`), 9

W

`write_to_file()` (`cassette.cassette_library.CassetteLibrary`
method), 8

`write_to_file()` (`cassette.cassette_library.DirectoryCassetteLibrary`
method), 8

`write_to_file()` (`cassette.cassette_library.FileCassetteLibrary`
method), 8