django-async-test Documentation

Release 0.2.2

Alex Hayes

May 09, 2016

Contents

1 Contents 3

2 License 7

3 Author 9

Python Module Index 11

asyncio unit tests with Django transactional support.

It supports Django 1.8+ for Python versions 3.5+ and uses asynctest under the covers to provide support for easily mocking coroutines.

CHAPTER 1

Contents

1.1 Installation

You can install django-async-test either via the Python Package Index (PyPI) or from github.

To install using pip;

```
$ pip install django-async-test
```

From github;

```
$ pip install git+https://github.com/alexhayes/django-async-test.git
```

1.2 Usage

asynctest.TestCase does a great job of mocking coroutines however if you're writing tests that manipulate the database in Django you'll most likely want to ensure that things are cleaned up after your test.

With `django_async_test.TestCase` you have the coroutine support of asynctest combined with the transaction support of Django's django.test.TestCase.

```
import django_async_test

class MyTestCase(django_async_test.TestCase):

    @django_async_test.patch('myapp.my_coroutine')
    def test_foo(self, MockMyCoroutine):

        # Mock our coroutine.
        MockMyCoroutine.return_value = 'Hello World'

        # Create an instance of MyModel
        MyModel.objects.create(...)

        ...
        ...
```

In the above example, the test is run inside a transaction by Django's django.test.TestCase, thus the creation of a MyModel will be rolled back, cleaning up the database.

Also, our co-routine will be patched correctly by asynctest.

Note that decorated `@django_async_test.patch` above actually comes from `asynctest` however to avoid the extra import `django_async_test` imports `asynctest.*` into it's namespace.

1.3 Differences to asynctest

asynctest supports the use of `setUp` and `tearDown` methods as coroutines in your `TestCase`. `django-async-test` does not support this, instead if you can define a `setUpAsync` and/or `tearDownAsync` method that will be called.

This package is mostly just an integration wrapper between Django and asynctest, you should read the asynctest docs.

1.4 Developer Documentation

1.4.1 Contributions

Contributions are more than welcome!

To get setup do the following;

```
mkvirtualenv --python=/usr/bin/python3.5 django-async-test
git clone https://github.com/alexhayes/django-async-test.git
cd django-async-test
pip install -r requirements/dev.txt
```

1.4.2 Running Tests

Once you've checked out you should be able to run the tests.

```
tox
```

1.4.3 Creating Documentation

```
cd docs
make clean html
```

1.5 Internal Module Reference

Release 0.2.2

Date May 09, 2016

1.5.1 django_async_test.testcase

Internal module reference for `django_async_test.testcase`.

class `django_async_test.testcase.`**`TestCase`**(*methodName='runTest', *args, **kwargs*)

Bases: `django.test.testcases.TestCase`, `asynctest.case.TestCase`

A testcase that wraps django.test.TestCase and asynctest.TestCase.

run(*result=None*)

Call django.test.TestCase's `run` method.

setUp()

Override setup method.

Note that `asynctest` supports `setUp` as a coroutine however `django_async_test.TestCase` instead supports a `setUpAsync` method.

tearDown()

Override tearDown method.

Note that `asynctest` supports `tearDown` as a coroutine however `django_async_test.TestCase` instead supports a *tearDownAsync*' method.

CHAPTER 2

License

This software is licensed under the *MIT License*. See the LICENSE.

CHAPTER 3

Author

Alex Hayes <alex@alution.com>

Python Module Index

d

`django_async_test.testcase`, 4

D

django_async_test.testcase (module), 4

R

run() (django_async_test.testcase.TestCase method), 4

S

setUp() (django_async_test.testcase.TestCase method), 5

T

tearDown() (django_async_test.testcase.TestCase method), 5
TestCase (class in django_async_test.testcase), 4