Django PyFixture Documentation

Release 0.0.1

Konstantine Rybnikov

July 06, 2013

CONTENTS

CHAPTER

ONE

RESOURCES

- Project on PyPi
- Project on BitBucket
- Mirror on github

CHAPTER

TWO

INSTALLATION

1. Add *django_pyfixture* into your *INSTALLED_APPS*.
2. Create empty *<appname>/fixtures/__init__.py* file inside your django app called *<appname>*.
3. Create your fixtures in files like *<appname>/fixtures/foo.py* with content similar to this:

```
# file proj/appname/fixtures/foo.py

from django_pyfixture import PyFixtureBase


class FooData(PyFixtureBase):
    def load_data(self):
        # create your python objects here
        pass
```

4. Add inheritance into your base test class like this:

> **Warning:** Make sure you put *PyFixtureTestCase* before *DjangoTestCase*. Seems that Unittest2-guys didn't inherit *object*, so now multi-inheritance via *super()* doesn't work good if order isn't right.

```
# file proj/utils/test_bases.py

from django_pyfixture import PyFixtureTestCase
from django.test import TestCase as DjangoTestCase


class BaseTestCase(PyFixtureTestCase, DjangoTestCase):
    pass
```

5. Add *py_fixtures* list inside your tests like this:

```
    # file proj/appname/tests/foo_tests.py

    class TestFoo(BaseTestCase):
        py_fixtures = ['foo']

        def test_should_get_list_of_foo(self):
            # do something with foo here
            Foo.objects.all()
```

6. To load some data from terminal use

```
python manage.py loaddata_py foo
```

CHAPTER

THREE

ADDITIONAL INFO

Warning: Remember, there's no magical way to clean-up any side-effects you cause, so if you do something beyond transactions (like writing to redis) – make sure you'll add cleanup after test or inside base test case. Cleanup method will be added into *django-pyfixture* later.

CHAPTER

FOUR

INDICES AND TABLES

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