

Speed Up Your Tests with

setUpTestData()

Adam Johnson

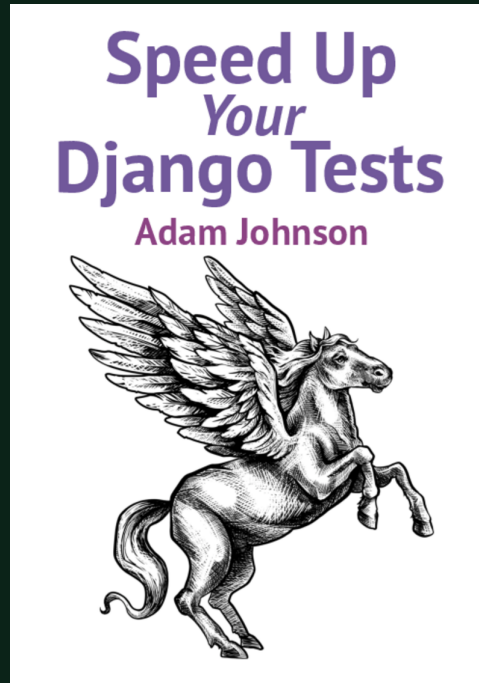


Five part talk

1. unittest's setUp* & tearDown* methods
2. Django's unittest extensions
3. setUpTestData()
4. Django 3.2's setUpTestData() isolation
5. How to convert a TestCase to use
setUpTestData()



Based on my Blog & Book



33% off during Djangocon Europe

Part One

unittest's setUp* & tearDown*
methods



Per-test: setUp(), tearDown()



Per-TestCase: setUpClass(), tearDownClass()



por exemplo:

```
class MyTests(TestCase):
    @classmethod
    def setUpClass(cls):
        super().setUpClass()
        cls.conn = acme.connect()

    @classmethod
    def tearDownClass(cls):
        cls.conn.close()
        super().tearDownClass()

    def setUp(self):
        super().setUp()
        self.user = make_user(self.conn)

    def tearDown(self):
        self.user.delete()
        super().tearDown()
```

Robust use:

```
class MyTests(TestCase):  
    def setUp(self):  
        super().setUp()  
        self.user = make_user(self.conn)  
  
    def tearDown(self):  
        if hasattr(self, "user"):  
            self.user.delete()  
        super().tearDown()
```



...or:



```
self.addCleanup(func)
```



```
cls.addClassCleanup(func) - Python 3.8+
```


Terse & *Robust*:

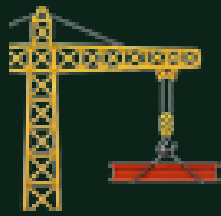
```
class MyTests(TestCase):
    @classmethod
    def setUpClass(cls):
        super().setUpClass()
        cls.conn = acme.connect()
        self.addClassCleanup(cls.conn.close)

    def setUp(self):
        super().setUp()
        self.user = make_user(self.conn)
        self.addCleanup(self.user.delete)
```

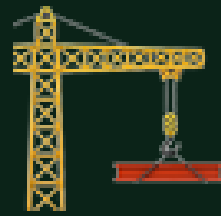


TestCase lifecycle

1. setUpClass ()
2. *Per test:*
 - i. setUp ()
 - ii. *run test*
 - iii. tearDown ()
 - iv. addCleanup () functions
3. tearDownClass ()
4. addClassCleanup () functions



Part Two



Django's unittest extensions

`unittest.TestCase`



`SimpleTestCase`



`TransactionTestCase`



`TestCase`



`(LiveServerTestCase)`

SimpleTestCase

- Blocks DB access
- Adds some assertion methods
- Does Django's own per-test setup *outside of* `setUp()`

TransactionTestCase

- Allows DB access
- Rolls back DB's by wiping and re-adding fixture data (slow)
- Use to test code that commits transactions

TestCase

- Rolls back DB's with **per-class** and **per-test** transactions
- `setUpClass()` calls ✨`setUpTestData()` ✨
- (Many more features!)



Django's TestCase lifecycle

1. `setUpClass()`: tx begin
2. `setUpClass():setUpTestData()`
3. *Per test:*
 - i. `_pre_setup()`: tx begin
 - ii. *run test*
 - iii. `_post_teardown()`: tx rollback
4. `tearDownClass()`: tx rollback



Part Three



setUpTestData()



So, setUpTestData()

- Called by setUpClass()
- @classmethod
- Default empty - yours don't need to call super()

Before:

```
class MyTests(TestCase):  
    def setUp(self):  
        self.book = Book.objects.create(title="1984")  
  
    ...
```



After:

```
class MyTests(TestCase):  
    @classmethod  
    def setUpTestData(cls):  
        cls.book = Book.objects.create(title="1984")  
  
    ...
```



Comparison

- Data creation in `setUp()`: N times
- Data creation in `setUpTestData()`: ✨*once*✨



Muito rápido!





Part Four



Django 3.2's `setUpTestData()`
isolation



Take these tests:

```
class MyTests(TestCase):
    @classmethod
    def setUpTestData(cls):
        cls.book = Book.objects.create(title="Meditations")

    def test_that_changes_title(self):
        self.book.title = "Antifragile"

    def test_that_reads_title_from_db(self):
        db_title = Book.objects.get().title
        assert db_title == "Meditations"

    def test_that_reads_in_memory_title(self):
        assert self.book.title == "Meditations"
```

⚠ Django < 3.2 ⚠

```
$ ./manage.py test example.core.tests
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.F.
=====
FAIL: test_that_reads_in_memory_title (example.core.tests.MyTests)
-----
Traceback (most recent call last):
  File ".../example/core/tests.py", line 19, in test_that_reads_in_m
    assert self.book.title == "Meditations"
AssertionError
-----

Ran 3 tests in 0.002s

FAILED (failures=1)
Destroying test database for alias 'default'...
```


- Database changes rolled back by transaction
- In-memory changes not rolled back
- Docs: re-query in each test - slow and verbose 🥲

👉 django-testdata: automatic per-test copying

```
from testdata import wrap_testdata

class MyTests(TestCase):
    @classmethod
    @wrap_testdata
    def setUpTestData(cls):
        cls.book = Book.objects.create(title="Meditations")

    ...
```



django-testdata merged in Django 3.2



```
class MyTests(TestCase):  
    @classmethod  
    def setUpTestData(cls):  
        cls.book = Book.objects.create(title="Meditations")  
  
    ...
```

Thanks Simon Charette! 👍



Part Five



How to convert a TestCase to
use setUpTestData()

 Four steps

 Blog post: **How to convert a TestCase from setUp() to setUpTestData()**

 Repeat across code base for great gains



por exemplo:

```
class IndexTests(TestCase):
    def setUp(self):
        self.book = Book.objects.create(title="1984")
        self.user = User.objects.create_user(
            username="tester",
            email="test@example.com",
        )
        self.client.force_login(self.user)

    def test_one(self):
        ...

    def test_two(self):
        ...
```



Install django-testdata on Django < 3.2



Search projects



[Help](#)

[Sponsors](#)

[Log in](#)

[Register](#)

django-testdata 1.0.3

`pip install django-testdata`



Latest version

Released: Apr 18, 2021

Django application providing isolation for model instances created during `setUpTestData``.

Navigation

[Project description](#)

[Release history](#)

[Download files](#)

Project links

[Homepage](#)

Statistics

GitHub statistics:

★ Stars: 19

🔗 Forks: 2

📄 Open issues/PRs: 0

View statistics for this project via

Project description

license [MIT](#) | pyPI [v1.0.3](#) | build [passing](#) | coverage [100%](#) | python [2.7](#) | [3.5](#) | [3.6](#) | [3.7](#) | [3.8](#) | [3.9](#) | wheel [yes](#)

Django application providing isolation for model instances created during `setUpTestData``.

Note: This package has been merged into Django and released in version 3.2 (see [PR #12608](#)).

Installation

```
pip install django-testdata
```

Motivation

Django 1.8 introduced `TestCase.setUpTestData`` to allow costly generation of model fixtures to be executed only once per test class in order to speed up testcase instances execution.

One gotcha of `setUpTestData`` though is that test instances all share the same model instances and have to be careful not to alter them to prevent breaking test isolation. Per Django's [documentation](#):

1 Run the target test case

```
$ ./manage.py test --keepdb example.core.tests.IndexTests
Using existing test database for alias 'default'...
System check identified no issues (0 silenced).
..
-----
Ran 2 tests in 0.015s

OK
Preserving test database for alias 'default'...
```



2 Add a stub setUpTestData()

```
class IndexTests(TestCase):  
+     @classmethod  
+     def setUpTestData(cls):  
+  
    def setUp(self):  
        self.book = Book.objects.create(title="1984")  
        self.user = User.objects.create_user(  

```

2 ...on Django < 3.2:

```
+from testdata import wrap_testdata

from example.core.models import Book

class IndexTests(TestCase):
+   @classmethod
+   @wrap_testdata
+   def setUpTestData(cls):
+
    def setUp(self):
        self.book = Book.objects.create(title="1984")
        self.user = User.objects.create_user(
```

3 Move data creation

```
class IndexTests(TestCase):
    @classmethod
    def setUpTestData(cls):
+         cls.book = Book.objects.create(title="1984")
+         cls.user = User.objects.create_user(
+             username="tester",
+             email="test@example.com",
+         )

    def setUp(self):
-         self.book = Book.objects.create(title="1984")
-         self.user = User.objects.create_user(
-             username="tester", email="test@example.com"
-         )
        self.client.force_login(self.user)

    def test_one(self):
```

3 Move data creation

```
class IndexTests(TestCase):
    @classmethod
    def setUpTestData(cls):
        cls.book = Book.objects.create(title="1984")
        cls.user = User.objects.create_user(
            username="tester",
            email="test@example.com",
        )

    def setUp(self):
        self.client.force_login(self.user)

    def test_one(self):
        ...

    def test_two(self):
        ...
```

4 Re-run tests

```
$ ./manage.py test --keepdb example.core.tests.IndexTests
Using existing test database for alias 'default'...
System check identified no issues (0 silenced).
..
-----
Ran 2 tests in 0.012s

OK
Preserving test database for alias 'default'...
```



 Enjoy N  1 performance gain

Thank you! 🙌

- Adam Johnson
- @adamchainz on GitHub & Twitter
- me@adamj.eu
- github.com/adamchainz/talk-speed-up-your-tests-with-setupTestData