

---

# **django-smarttest Documentation**

***Release 0.3.0***

**Kidosoft**

2017-01-25



<b>1</b>	<b>Goal</b>	<b>1</b>
<b>2</b>	<b>Installation</b>	<b>3</b>
<b>3</b>	<b>Configuration</b>	<b>5</b>
<b>4</b>	<b>Usage</b>	<b>7</b>
4.1	Preventing tests from touching database . . . . .	7
4.2	Running only selected test types . . . . .	7
4.3	TestCase for use with switchable splinter driver . . . . .	8
<b>5</b>	<b>Supported Django versions</b>	<b>9</b>
<b>6</b>	<b>Documentation</b>	<b>11</b>
6.1	Decorators . . . . .	11
<b>7</b>	<b>Indices and tables</b>	<b>13</b>
	<b>Python Module Index</b>	<b>15</b>



---

**Goal**

---

Provide code snippets which help running tests in Django.



---

# Installation

---

Install requirements:

```
pip install -r requirements.txt
```

Install Smarttest:

```
pip install django-smarttest
```

or current development version:

```
pip install git+https://github.com/kidosoft/django-smarttest.git
```





---

## Configuration

---

```
INSTALLED_APPS = [  
    ...  
    'smarttest',  
    ...  
]
```



---

## Usage

---

### 4.1 Preventing tests from touching database

```
import unittest

from smarttest.decorators import no_db_testcase

@no_db_testcase
class SomeTestCase(unittest.TestCase):

    def test_some_test(self):
        ...
```

If you'll accidentally write code that tries to run some query on database you'll get exception.

### 4.2 Running only selected test types

```
import unittest

from smarttest.decorators import test_type

@test_type('acceptance')
class SomeAcceptanceTestCase(unittest.TestCase):

    def test_some_acceptance_test(self):
        ...

@test_type('unit')
class SomeUnitTestCase(unittest.TestCase):

    def test_some_unit_test(self):
        ...

class UnspecifiedTypeTestCase(unittest.TestCase):

    def test_some_test(self):
        ...
```

```
$ python -m unittest script
...
```

```
-----
Ran 3 tests in 0.000s

OK
$ IGNORE_TESTS=unit python -m unittest script
.s.
-----
Ran 3 tests in 0.000s

OK (skipped=1)
$ IGNORE_TESTS=acceptance python -m unittest script
s..
-----
Ran 3 tests in 0.000s

OK (skipped=1)
$ IGNORE_TESTS=acceptance,unit python -m unittest script
ss.
-----
Ran 3 tests in 0.000s

OK (skipped=2)
```

Test type can be any selected word. It doesn't have to be “unit” or “acceptance”. You can have different test types for running in different environments if you need.

## 4.3 TestCase for use with switchable splinter driver

Simple TestCase that allows for simple switching between different drivers in tests.

In settings.py:

```
TEST_DRIVER = 'firefox' # or django or any other
```

In test:

```
from smarttest.testcases import SplinterTestCase

class SomeTestCase(SplinterTestCase):

    def test_some_test(self):
        self.browser.visit(self.get_host() + '/')
```

If you run tests continuously (e.g. doing Test Driven Development) TEST\_DRIVER='django' (default setting) is the fastest driver that do not interrupt your workflow. However when you run your acceptance tests you may want to check how it behaves with real browser. Simply set TEST\_DRIVER='firefox' or any other real browser supported by splinter.

---

## Supported Django versions

---

Tested with:

- Django 1.2.7 on python2.7
- Django 1.3.7 on python2.7
- Django 1.4.16 on python2.7
- Django 1.5.11 on python2.7, python3.3, python3.4
- Django 1.6.8 on python2.7, python3.3, python3.4
- Django 1.7.1 on python2.7, python3.3, python3.4
- Django 1.8.8 on python2.7, python3.3, python3.4



---

## Documentation

---

<http://django-smarttest.readthedocs.org/>

### 6.1 Decorators

`smarttest.decorators.no_db_testcase` (*func*)

Decorator that prevents your test touching database.

**Raises `RuntimeError`** – if test try to do database query

`smarttest.decorators.test_type` (*name*)

Function creating decorators that allows skipping tests of declared type.

**Parameters** **name** (*str*) – arbitrary name of test type e.g. acceptance,unit





---

## Indices and tables

---

- `genindex`
- `modindex`
- `search`



## S

`smarttest.decorators`, [11](#)



## N

`no_db_testcase()` (in module `smarttest.decorators`), [11](#)

## S

`smarttest.decorators` (module), [11](#)

## T

`test_type()` (in module `smarttest.decorators`), [11](#)