Django Test Tools Documentation

Release 1.7.6

Luis Carlos Berrocal

Mar 21, 2020

Contents

Contents:

CHAPTER 1

Django Test Tools

Simple tests tools to make testing faster and easier. Most of the tools are to do a quick scaffolding for tests.

The tools presume a naming convention:

- **Tests:** Are named with the convention **TestCaseModelName**. For a model named *Poll* the test would be generated as the testing class would be *TestCasePoll*
- **Factories:** Are named with the convention **ModelName**. For a model named *Poll* the test would be generated as the testing class would be *PollFactory*
- **Serializers:** Are named with the convention **TestCaseSerializer**. For a model named *Poll* the test would be generated as the testing class would be *PollSerializer*

Compatibility matrix:

Python version	Django 1.11.x	Django 2.2.x	Django 3.0.x
3.7	x	x	x
3.6	x	x	x

1.1 Documentation

The full documentation is at https://django-test-tools.readthedocs.io.

1.2 Quickstart

Install Django Test Tools:

```
pip install django-test-tools
```

In your settings.py file add it to your *INSTALLED_APPS*

```
INSTALLED_APPS = (
    ...
    'django_test_tools',
    ...
)
```

Add the settings variable **TEST_OUTPUT_PATH**

```
import environ

ROOT_DIR = (
    environ.Path(__file__) - 3
)  # (alpha_clinic/config/settings/base.py - 3 = alpha_clinic/)
APPS_DIR = ROOT_DIR.path("myapp")
TEST_OUTPUT_PATH = ROOT_DIR.path("output").root
```

1.3 Features

1.3.1 Factory Generator

```
$  python manage.py generate_factories project.app
```

1.3.2 Model Test Case Generator

```
$  python manage.py generate_model_test_cases project.app
```

1.3.3 Serializer Generator

```
$ python manage.py generate_serializers project.app -s ModelSerializer
```

1.4 Running Tests

Does the code actually work?

```
source <YOURVIRTUALENV>/bin/activate
(myenv) $ pip install tox
(myenv) $ tox
```

1.5 Builds

```
$ make patch
```

Check the .travis.yml to make sure the versions of Djago are the latests. Check in https://www.djangoproject.com/download/ for the latest versions.

Close the git-flow release.

Instead of patch you could also use **major** o **minor** depending on the level of the release.

```
make upload
```

1.6 Credits

Tools used in rendering this package:

- Cookiecutter
- cookiecutter-djangopackage

CHAPTER 2

Installation

At the command line:

```
$ easy_install django-test-tools
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv django-test-tools
$ pip install django-test-tools
```

CHAPTER 3

Usage

To use Django Test Tools in a project, add it to your *INSTALLED_APPS*:

```
INSTALLED_APPS = (
    ...
    'django_test_tools.apps.DjangoTestToolsConfig',
    ...
)
```

Add Django Test Tools's URL patterns:

```
from django_test_tools import urls as django_test_tools_urls


urlpatterns = [
    ...
    url(r'^', include(django_test_tools_urls)),
    ...
]
```

CHAPTER 4

django_test_tools package

4.1 Subpackages

4.1.1 django_test_tools.doc_utils package

Submodules

django_test_tools.doc_utils.folder_structure module

Module contents

4.1.2 django_test_tools.flake8 package

Submodules

django_test_tools.flake8.parsers module

class django_test_tools.flake8.parsers.**Flake8Parser**
Bases: object

3 E124 closing bracket does not match visual indentation 6 E127 continuation line over-indented for visual indent 11 E128 continuation line under-indented for visual indent 2 E221 multiple spaces before operator 1 E222 multiple spaces after operator 10 E225 missing whitespace around operator 6 E231 missing whitespace after ',' 2 E251 unexpected spaces around keyword / parameter equals 4 E261 at least two spaces before inline comment 4 E262 inline comment should start with '# ' 8 E265 block comment should start with '# ' 4 E266 too many leading '#' for block comment 2 E271 multiple spaces after keyword 5 E302 expected 2 blank lines, found 1 7 E303 too many blank lines (3) 2 E402 module level import not at top of file 8 E501 line too long (123 > 120 characters) 17 F401 'django.contrib.admin' imported but unused 25 F405 'env' may be undefined, or defined from star imports: .base 1 F811 redefinition of unused 'RemarksManager' from line 3 7 F841 local variable 'response' is assigned to but never used 2 W293 blank line contains whitespace 6 W391 blank line at end of file

parse_summary(*filename*)

write_summary(*source_filename*, *target_filename*)

class django_test_tools.flake8.parsers.**RadonParser**
Bases: object

config/settings/test.py LOC: 61 LLOC: 12 SLOC: 23 Comments: 23 Single comments: 22 Multi: 4 Blank: 12 - Comment Stats

(C % L): 38% (C % S): 100% (C + M % L): 44%

** **Total** ** LOC: 2149 LLOC: 894 SLOC: 1311 Comments: 335 Single comments: 310 Multi: 128 Blank: 400 - Comment Stats

(C % L): 16% (C % S): 26% (C + M % L): 22%

parse_totals(*filename*)

write_totals(*source_filename*, *target_filename*)

Module contents

4.1.3 django_test_tools.generators package

Submodules

django_test_tools.generators.model_test_gen module

django_test_tools.generators.serializer_gen module

class django_test_tools.generators.serializer_gen.**AppSerializerGenerator**(*app*, *serializer_class='ModelSerializ*

Bases: object

class django_test_tools.generators.serializer_gen.**SerializerGenerator**(*model*, *serializer_class='ModelSerializer'*)

Bases: object

Module contents

4.1.4 django_test_tools.git package

Submodules

django_test_tools.git.helpers module

class django_test_tools.git.helpers.**GenericCVS**
Bases: object

classmethod commit(*message*)

classmethod is_usable()

class django_test_tools.git.helpers.**Git**
Bases: *django_test_tools.git.helpers.GenericCVS*

Option Description of Output %H Commit hash %h Abbreviated commit hash %T Tree hash %t Abbreviated tree hash %P Parent hashes %p Abbreviated parent hashes %an Author name %ae Author email %ad Author date (format respects the –date=option) %ar Author date, relative %cn Committer name %ce Committer email %cd Committer date %cr Committer date, relative %s Subject

classmethod add_path(*path*)

classmethod assert_nondirty()

classmethod latest_tag_info()

report()

classmethod tag(*name*, *message*)

Module contents

4.1.5 django_test_tools.management package

Subpackages

django_test_tools.management.commands package

Submodules

django_test_tools.management.commands.generate_factories module

django_test_tools.management.commands.generate_model_test_cases module

django_test_tools.management.commands.generate_serializers module

django_test_tools.management.commands.parse_qc_files module

Module contents

Module contents

4.2 Submodules

4.3 django_test_tools.app_manager module

4.4 django_test_tools.apps module

4.5 django_test_tools.assert_utils module

4.6 django_test_tools.excel module

class django_test_tools.excel.**ExcelAdapter**
Bases: object

classmethod convert_to_dict (*filename*, *sheet_name=None*)
Reads an Excel file and converts every row into a dictionary. All values are converted to strings. Assumes first row contains the name of the attributes. :param filename: <str> Excel filename :param sheet_name: <str> Name of the sheet :return: <list> A list of dictionaries.

classmethod convert_to_list (*filename*, *sheet_name=None*, *has_header=True*)
Reads an Excel file and converts every row into a list of values. :param filename: <str> Excel filename :param sheet_name: <str> Name of the sheet :param has_header: <bool> If true the first row is not included in the list. :return: <list> A list of lists.

4.7 django_test_tools.file_utils module

4.8 django_test_tools.mixins module

4.9 django_test_tools.models module

4.10 django_test_tools.urls module

4.11 django_test_tools.utils module

4.12 Module contents

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/luiscberrocal/django-test-tools/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “feature” is open to whoever wants to implement it.

5.1.4 Write Documentation

Django Test Tools could always use more documentation, whether as part of the official Django Test Tools docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/luiscberrocal/django-test-tools/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *django-test-tools* for local development.

1. Fork the *django-test-tools* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/django-test-tools.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv django-test-tools
$ cd django-test-tools/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 django_test_tools tests
$ python setup.py test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/luiscberrocal/django-test-tools/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_django_test_tools
```

CHAPTER 6

Credits

6.1 Development Lead

- Luis Carlos Berrocal <luis.berrocal.1942@gmail.com>

6.2 Contributors

Issis Itzel Montilla <issis.montilla@gmail.com>

CHAPTER 7

History

7.1 0.1.0 (2017-04-26)

- First release on PyPI.

Python Module Index

d

Index

W