django-ontruck Documentation

Release 0.16.0

ontruck

Dec 09, 2021

Contents

1 django-ontruck **3**
- 1.1 Documentation . . . 3
- 1.2 Quickstart . . . 3
- 1.3 Features . . . 3
- 1.4 Running Tests . . . 4
- 1.5 Generate documentation . . . 4

2 Installation **5**

3 Usage **7**

4 Features **9**
- 4.1 Use cases . . . 9
- 4.2 Events . . . 9
- 4.3 Queries . . . 10
- 4.4 Models . . . 10
- 4.5 Notifiers . . . 10
- 4.6 Views . . . 11
- 4.7 Value Objects . . . 11
- 4.8 Testing . . . 11
- 4.9 Commands . . . 12
- 4.10 Monads . . . 12

5 Contributing **13**
- 5.1 Types of Contributions . . . 13
- 5.2 Get Started! . . . 14
- 5.3 Pull Request Guidelines . . . 15
- 5.4 Tips . . . 15

6 History **17**
- 6.1 0.16.0 (2021-12-9) . . . 17
- 6.2 0.15.0 (2021-10-8) . . . 17
- 6.3 0.14.0 (2021-07-22) . . . 17
- 6.4 0.12.0 (2021-4-15) . . . 17
- 6.5 0.11.2 (2021-2-1) . . . 17
- 6.6 0.11.1 (2020-12-31) . . . 18
- 6.7 0.11.0 (2020-12-09) . . . 18

6.8 0.10.2 (2020-11-27) . . . 18
6.9 0.10.1 (2020-11-27) . . . 18
6.10 0.10.0 (2020-11-26) . . . 18
6.11 0.9.0 (2020-10-27) . . . 18
6.12 0.8.0 (2020-10-26) . . . 18
6.13 0.7.0 (2020-6-10) . . . 18
6.14 0.6.0 (2020-5-29) . . . 19
6.15 0.5.0 (2020-5-17) . . . 19
6.16 0.4.0 (2020-4-29) . . . 19
6.17 0.3.0 (2020-3-31) . . . 19
6.18 0.2.0 (2020-3-23) . . . 19
6.19 0.1.0 (2020-03-20) . . . 19

Contents:

CHAPTER 1

django-ontruck

Django extended by Ontruck-ers

1.1 Documentation

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

1.2 Quickstart

Install django-ontruck:

```
pip install django-ontruck
```

Add it to your *INSTALLED_APPS*:

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

1.3 Features

- Use cases

- Events
- Queries
- Models
- Views
- Testing
- Commands

1.4 Running Tests

Does the code actually work?

Prepare test env

```
source <YOURVIRTUALENV>/bin/activate
(myenv) $ pip install -r requirements_test.txt
```

Run tests

```
(myenv) $ make test
```

Run tests in several python versions using tox

```
(myenv) $ make test-all
```

Run tests getting code coverage

```
(myenv) $ make coverage
```

1.5 Generate documentation

```
(myenv) $ pip install -r requirements_dev.txt
(myenv) $ make docs
```

CHAPTER 2

Installation

At the command line with ssh:

```
$ pip install -e git+ssh://git@github.com/ontruck/django-ontruck.git@master
↪#egg=django-ontruck
```

Or with https:

```
$ pip install -e git+https://github.com/ontruck/django-ontruck.git@master#egg=django-
↪ontruck
```

CHAPTER 3

Usage

To use django-ontruck in a project, add it to your *INSTALLED_APPS*:

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

CHAPTER 4

Features

4.1 Use cases

Use case layer is in charge of handle business logic.

- Manages transaction
- Manages executions dependent of committed transaction
- Returns data to caller (i.e for API responses purposes)
- Hides complexity
- Allows nesting to DRY
- Establishes events as the only way to propagate on_commit execution

```
from django_ontruck.use_cases import UseCaseBase
from .events import FooEvent
from .models import FooModel

class FooUseCase(UseCaseBase):

def execute_in_commit(self, command, executed_by=None):
    example = FooModel.objects.create(**command)
    self.events.append(FooEvent(attr1='attr1_value'))
    return example
```

Events created will be published when outermost transaction is committed

4.2 Events

Abstraction to decouple contexts wrapping django signals

```
from django.dispatch import Signal
from django_ontruck.events import EventBase

class FooEvent(EventBase):
    signal = Signal(['attr1'])
```

Check *show_events* command to *List all events*.

```
python manage.py show_events
```

```
(django-ontruck) juan@juan-XPS-13-7390:~/code/django-ontruck$ python manage.py show_events
[tests.test_app]:
 * FooEvent -> {'attr1'}
   - /home/juan/code/django-ontruck/tests/test_app/events/handlers.py.foo_event_handler
```

4.3 Queries

Analogue to CQRS, a separation between the logic that writes (use cases) from the one that reads (queries).

```
class CountFooQuery(QueryBase):

    def execute(self, command, executed_by=None):
        text = command.get('title')
        return {'count': FooModel.objects.filter(title__contains=text).count()}
```

4.4 Models

Base model to track some CRUD dates and author. Safe delete is implemented.

```
from django.db import models
from django_ontruck.models import BaseModel

class FooModel(BaseModel):
    title = models.CharField(max_length=50)
```

There is a BaseManager in order to filter *deleted=False* elements by default.

Define manager for your Queryset

```
#
FooManager = BaseManager.from_queryset(QuerySet)
```

Set managers including deleted elements or not

```
class FooModel(BaseModel):
    ...
    objects = FooManager(include_deleted=False)
    objects_all = FooManager(include_deleted=True)
```

4.5 Notifiers

Base classes to notify events to 3rd party systems like Slack, Segment, etc.

```
from django.db import models
from django_ontruck.notifiers.segment import SegmentNotifier
from django_ontruck.notifiers import AsyncNotifier

class FooNotifier(segmentNotifier):
    async_class = AsyncNotifier
    event_id = 'test_event'
```

4.6 Views

Collections of DRF views extended to fit with BaseModel and UseCases

4.7 Value Objects

Objects for which equality is determined by their attributes as opposed to by identity. That is, they are **fungible**: one instance of an object can be swapped for any other instance as long as their attributes are the same (much like coins, or stamps.)

4.7.1 Money

The *Money* class represents a monetary value together with its currency.

- The value is stored without rounding until the *allocate* method is invoked. How rounding is performed depends on the currency.

```
from django_ontruck.value_objects.money import euros, pounds, Currencies, money,␣
↪Currency

# commonly used currencies have their own helpers:
two_euros = euros('2.00')  # 50.00 €
one_hundred_pounds = pounds('100.00') # £100.00

# Other currencies can be created using the `money` helper and the currency
twenty_zloty = money(Currencies.PLN, '20')

# Any missing currencies can be created
alt = Currency('ALT', 'Altarian Dollars', '$')
one_altarian_dollar = money(alt, '1')

# We can apply arithmetic operations and the value is stored without rounding.
divided = two_euros / 3
divided.amount  # Decimal('0.66666666666666666666666666667'))

# We can round the value (according to the currency) using `allocate`
divided.allocate().amount  # Decimal('0.67'))
```

4.8 Testing

Utils for testing.

4.8.1 Patch transactions and run transaction.on_commit

After last transaction inside test is exit

Create a fixture

```
@pytest.fixture(autouse=True)
def _run_on_commit_callbacks(request):
    marker = request.node.get_closest_marker("run_on_commit_callbacks")

    if marker:
        with PatchedAtomic():
            yield
    else:
        yield
```

Mark your tests you want to use it

```
@mark.run_on_commit_callbacks
def test_use_case_post_commit(self, mocker, foo_use_case):
    mock_event_send = mocker.patch('django_ontruck.events.EventBase.send')
    foo_use_case.execute({})
    mock_event_send.assert_called_once()
```

4.9 Commands

4.9.1 List all events

Show all events defined in each app and handlers connected.

```
python manage.py show_events
```

```
(django-ontruck) juan@juan-XPS-13-7390:~/code/django-ontruck$ python manage.py show_events
[tests.test_app]:
 * FooEvent -> {'attr1'}
   - /home/juan/code/django-ontruck/tests/test_app/events/handlers.py.foo_event_handler
```

4.9.2 App template

Start app with directory/files structure.

```
python manage.py startontruckapp appname
```

4.10 Monads

We provide an implementation of useful monads.

4.10.1 Result

Result monad is used to encapsulate return results whenever they are sucessful or not in order to treat the responses like a streamlined pipeline.

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/ontruck/django-ontruck/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-ontruck could always use more documentation, whether as part of the official django-ontruck 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/ontruck/django-ontruck/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-ontruck* for local development.

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

```
$ git clone git@github.com:your_name_here/django-ontruck.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-ontruck
$ cd django-ontruck/
$ 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:

```
$ make lint
$ make test
$ make test-all
```

   To get flake8 and tox, just pip install -r requirements_test.txt.

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 3. Check https://travis-ci.org/ontruck/django-ontruck/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ pytest tests/test_events.py::TestEvents::test_event_creation
```

Generate docs with

```
$ make docs
```

CHAPTER 6

History

6.1 0.16.0 (2021-12-9)

- *BaseManager* to allow filtering *deleted=True* elements.

6.2 0.15.0 (2021-10-8)

- Modify *Ok.map_err* and *Error.map_err* implementation to match *Result.map_err*
- Add pipe operator for *Result* monads.

6.3 0.14.0 (2021-07-22)

- Add Result monad

6.4 0.12.0 (2021-4-15)

- Add customer.io notifier

6.5 0.11.2 (2021-2-1)

- Fix DB disconnections in base Consumer class

6.6 0.11.1 (2020-12-31)

- Add NoBrowsableAPIRenderer view mixin class

6.7 0.11.0 (2020-12-09)

- Add base Consumer class

6.8 0.10.2 (2020-11-27)

- Simplify definition of scalar when diffing dicts

6.9 0.10.1 (2020-11-27)

- Account for recursive Sequences (e.g. strings) in diff_dicts

6.10 0.10.0 (2020-11-26)

- Extend diff_dicts method do account for sequences

6.11 0.9.0 (2020-10-27)

- Add diff_dicts method
- Add Money class

6.12 0.8.0 (2020-10-26)

- Remove support for python < 3.8
- Add base classes for notifiers

6.13 0.7.0 (2020-6-10)

- SortedJSONField serializer
- python 3.8 CI upgrade
- Django 2.2.13 CI upgrade
- Drop Django 2.1 CI

6.14 0.6.0 (2020-5-29)

- Queries

6.15 0.5.0 (2020-5-17)

- Implement django signal methods (disconnect, receiver) for Events

6.16 0.4.0 (2020-4-29)

- Extend views

6.17 0.3.0 (2020-3-31)

- startontruckapp command
- Django 2.2 CI upgrade

6.18 0.2.0 (2020-3-23)

- show_events command

6.19 0.1.0 (2020-03-20)

- Use cases
- Events
- Views
- Models
- Testing