Django, from nightmare to dream

with Best Practices

by Stéphane Wirtel
Hello, I am Stéphane

- Python Freelancer
- Open Source = My Passion/Job
- PythonFOSDEM
- CPython contributor
- PSF, EPS members, CSA from PSF
- former core dev of Odoo (Open Source ERP)
- blah blah Python blah
ep2017.europython.org
EuroPython website -> epcon

https://github.com/EuroPython/epcon
with epcon, we can do...

- conference management
with epcon, we can do...

- conference management
- ticket management
with **epcon**, we can do...

- conference management
- ticket management
- statistics (attendees, speakers, (un)assigned or orphan tickets)
with **epcon**, we can do...

- conference management
- ticket management
- statistics (attendees, speakers, (un)assigned or orphan tickets)
- invoice/refund
with epcon, we can do...

- conference management
- ticket management
- statistics (attendees, speakers, (un)assigned or orphan tickets)
- invoice/refund
- helpdesk
with epccon, we can do...

- conference management
- ticket management
- statistics (attendees, speakers, (un)assigned or orphan tickets)
- invoice/refund
- helpdesk
- notifications by email
with **epcon**, we can do...

- conference management
- ticket management
- statistics (attendees, speakers, (un)assigned or orphan tickets)
- invoice/refund
- helpdesk
- notifications by email
- **hotel and room booking**
- **sim card**
epcon is a good tool for EuroPython
but ...
State of Union

- Python 2.7 (support 2020)
State of Union

- Python 2.7 (support 2020)
- Django 1.8 (support april 2018)
State of Union

- Python 2.7 (support 2020)
- Django 1.8 (support april 2018)
- few dependencies are deprecated or no maintainers (incompatible with django 1.11)
State of Union

- Python 2.7 (support 2020)
- Django 1.8 (support april 2018)
- few dependencies are deprecated or no maintainers (incompatible with django 1.11)
- SQLite
State of Union

- Python 2.7 (support 2020)
- Django 1.8 (support april 2018)
- few dependencies are deprecated or no maintainers (incompatible with django 1.11)
- SQLite
- no tests (Python 168 files and 27487 lines of code)
State of Union

- Python 2.7 (support 2020)
- Django 1.8 (support april 2018)
- few dependencies are deprecated or no maintainers (incompatible with django 1.11)
- SQLite
  - no tests (Python 168 files and 27487 lines of code)
  - no documentation, some comments are in Italian
State of Union

- Python 2.7 (support 2020)
- Django 1.8 (support april 2018)
  - few dependencies are deprecated or no maintainers (incompatible with django 1.11)
- SQLite
  - no tests (Python 168 files and 27487 lines of code)
- no documentation, some comments are in Italian
- duplicated/dead code
State of Union

- Python 2.7 (support 2020)
- Django 1.8 (support april 2018)
- few dependencies are deprecated or no maintainers (incompatible with django 1.11)
- SQLite
  - no tests (Python 168 files and 27487 lines of code)
- no documentation, some comments are in Italian
- duplicated/dead code
- no async for jobs
State of Union

- Python 2.7 (support 2020)
- Django 1.8 (support april 2018)
- few dependencies are deprecated or no maintainers (incompatible with django 1.11)
- SQLite
  - no tests (Python 168 files and 27487 lines of code)
- no documentation, some comments are in Italian
- duplicated/dead code
- no async for jobs
- no Continuous Integration / Continuous Deployment
State of Union

- Python 2.7 (support 2020)
- Django **1.8** (support april 2018)
- few dependencies are deprecated or no maintainers (incompatible with django 1.11)
- SQLite
- **no tests** (Python 168 files and 27487 lines of code)
- **no documentation**, some comments are in Italian
- duplicated/dead code
- no async for jobs
- **no Continuous Integration / Continuous Deployment**
- no API for external tools
- need to export data (mobile app, ticket search app, etc...)
State of Union

- Python 2.7 (support 2020)
- Django 1.8 (support april 2018)
- few **dependencies are deprecated** or no maintainers (incompatible with django 1.11)
- SQLite
  - **no tests** (Python 168 files and 27487 lines of code)
- **no documentation**, some comments are in Italian
- duplicated/dead code
- no async for jobs
- **no Continuous Integration** / Continuous Deployment
- no API for external tools
- need to export data (mobile app, ticket search app, etc...)
- no syslog, just send error with exception to the mailing list.
State of Union

- Python 2.7 (support 2020)
- Django 1.8 (support april 2018)

- few **dependencies are deprecated** or no maintainers (incompatible with django 1.11)

- SQLite
- **no tests** (Python 168 files and 27487 lines of code)

- **no documentation**, some comments are in Italian

- duplicated/dead code

- no async for jobs

- **no Continuous Integration** / Continuous Deployment
- no API for external tools
- need to export data (mobile app, ticket search app, etc...)
- no syslog, just send error with exception to the mailing list.
- settings were **hardcoded**, no environment variables
HOUSTON

WE HAVE A PROBLEM!
and seriously

we are not alone

in this case.
But, don't forget one thing!
in fact, there is a lot of things
in fact, there is a lot of things

- Many former developers (it's an open source project)
  - Inherited from PyCon Italia 2009
  - 31 contributors since 2009 (first version)
in fact, there is a lot of things

• Many former developers (it's an open source project)
  ◦ Inherited from PyCon Italia 2009
  ◦ **31 contributors** since 2009 (first version)

• Volunteers
in fact, there is a lot of things

- Many former developers (it's an open source project)
  - Inherited from PyCon Italia 2009
  - 31 contributors since 2009 (first version)

- Volunteers

- Free time
in fact, there is a lot of things

- Many former developers (it's an open source project)
  - Inherited from PyCon Italia 2009
  - 31 contributors since 2009 (first version)

- Volunteers

- Free time

- No web dev, but data scientist, backend dev
We have a real challenge
We have a real challenge

- Continuous Integration / Continuous Deployment
We have a real challenge

- Continuous Integration / Continuous Deployment
- Documentation
We have a real challenge

- Continuous Integration / Continuous Deployment
- Documentation
- Configuration
We have a real challenge

- Continuous Integration / Continuous Deployment
- Documentation
- Configuration
- Tests
We have a real challenge

- Continuous Integration / Continuous Deployment
- Documentation
- Configuration
- Tests
- Write Code/Refactoring
We have a real challenge

- Continuous Integration / Continuous Deployment
- Documentation
- Configuration
- Tests
- Write Code/Refactoring
- Quality of code
We have a real challenge

- Continuous Integration / Continuous Deployment
- Documentation
- Configuration
- Tests
- Write Code/Refactoring
- Quality of code
- Profiling
We have a real challenge

- Continuous Integration / Continuous Deployment
- Documentation
- Configuration
- Tests
- Write Code/Refactoring
- Quality of code
- Profiling
- Deployment
We have a real challenge

- Continuous Integration / Continuous Deployment
- Documentation
- Configuration
- Tests
- Write Code/Refactoring
- Quality of code
- Profiling
- Deployment
- Monitoring
Continuous Integration

- Travis (https://www.travis.org)

```yaml
language: python
python:
  - "2.7"

install:
  - pip install -r requirements-dev.txt

script: "python manage.py compilemessages && python manage.py test"
```
Documentation

- Sphinx
Configuration

- **django-dotenv**

Your `.env` file

```bash
POSTGRES_USER="postgres"
POSTGRES_PASSWORD=""
SECRET_KEY="whoami"
ALLOWED_HOSTS=''
CSRF_COOKIE_SECURE=False
DEBUG=True
ENVIRONMENT='LOCAL'

DJANGO_SETTINGS_MODULE=dev-settings
```

Your `manage.py` file

```python
import dotenv
if __name__ == '__main__':
    dotenv.read_dotenv()
    os.environ.setdefault("DJANGO_SETTINGS_MODULE", "project.settings")
...
```
Tests

- unittest / pytest
- django.test / pytest-django
- mock / pytest-mock
Tests

- coverage / pytest-cov
- django-coverage-plugin
Tests

django-test-without-migrations and pytest-django

```bash
python manage test --nomigrations
# or
pytest --nomigrations
```
Tests

factory_boy and django-factoryboy

definition

```
class TalkFactory(factory.django.DjangoModelFactory):
    class Meta:
        model = 'conference.Talk'

    title = factory.LazyAttribute(lambda talk: fake.sentence(nb_words=6, variable_nb_words=True))
    sub_title = factory.Faker('sentence', nb_words=12, variable_nb_words=True)
    slug = factory.LazyAttribute(lambda talk: slugify(talk.title))
    level = factory.Iterator(conference.models.TALK_LEVEL, getter=lambda x: x[0])
    status = factory.Iterator(conference.models.TALK_STATUS, getter=lambda x: x[0])
    conference = factory.Iterator(conference.models.Conference.objects.all().values_list())
    language = factory.Iterator(TALK_LANGUAGES, getter=lambda x: x[0])
```

Definition of a Factory in app/tests/factories/model.py
Write Code/Refactoring

These tools will help your code

With the tests, you can start to rewrite the bad code

- pyflake8, pylint
- vulture
- django-pdb and --ipdb
- pdbpp (pdb++) (for pytest) or ipdb
Write Code/Refactoring

These tools will help your code

With the tests, you can start to rewrite the bad code

```
mypy with mypy-django

gnrbag.py:84: error: Incompatible types in assignment (expression has type "classmethod", variable has type Callable[[Any], Any])
gnrbag.py:317: error: Name 'json' already defined
gnrbag.py:2874: error: Need type annotation for variable
gnrbag.py:3083: error: Dict entry 3 has incompatible type "str": "str"
```
Write Code/Refactoring

These tools will help your code

With the tests, you can start to rewrite the bad code

- `autoflake --remove-all-unused-imports`
- `isort`
Profiling

django-debug-toolbar
Profiling

django-devserver + line_profiler

[sql] (3ms) 1 queries with 0 duplicates
[profile] Total time to render was 0.03s
[profile] Timer unit: 1e-06 s

Total time: 0.009523 s
File: /home/stephane/.virtualenvs/ep2017/lib/python2.7/site-packages/cms/views.py
Function: details at line 23

<table>
<thead>
<tr>
<th>Line #</th>
<th>Hits</th>
<th>Time</th>
<th>Per Hit</th>
<th>% Time</th>
<th>Line Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>def details(request, slug):</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The main view of the Django-C</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>page.</td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&quot;&quot;</td>
</tr>
</tbody>
</table>
Profiling

`pytest-profiling` with `--profile-svg --durations=2`

```
$ pytest --durations=2 --profile-svg
```

```
======================== slowest 2 test durations ==========================
0.47s setup    assopy/tests/test_reset_password.py::ResetPasswordTestCase::test_reset_password
0.36s call     assopy/tests/test_stripe.py::StripeViewTestCase::test_add_stripe_on_order_test
```

if you are a purist, maybe you will prefer `cProfile`
Profiling

with `--profile-svg` you can get this result
Profiling

cprofilev is a web interface for the cProfile output

```
pip install cprofilev

cprofilev -f prof/test_reset_password.prof
[cProfileV]: cProfile output available at http://127.0.0.1:4000
```

Tue Jul 11 12:11:56 2017 test_reset_password.prof

```
462642 function calls (459672 primitive calls) in 0.368 seconds

Ordered by: cumulative time

<table>
<thead>
<tr>
<th>ncalls</th>
<th>tottime</th>
<th>percall</th>
<th>cumtime</th>
<th>percall</th>
<th>filename:lineno(function)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2/1</td>
<td>0.000</td>
<td>0.000</td>
<td>0.368</td>
<td>0.368</td>
<td>/home/stephane/.virtualenvs/ep2017/lib/python2</td>
</tr>
<tr>
<td>2/1</td>
<td>0.000</td>
<td>0.000</td>
<td>0.368</td>
<td>0.368</td>
<td>/home/stephane/.virtualenvs/ep2017/lib/python2</td>
</tr>
<tr>
<td>1</td>
<td>0.000</td>
<td>0.000</td>
<td>0.368</td>
<td>0.368</td>
<td>/home/stephane/.virtualenvs/ep2017/lib/python2</td>
</tr>
<tr>
<td>1</td>
<td>0.000</td>
<td>0.000</td>
<td>0.368</td>
<td>0.368</td>
<td>/home/stephane/.virtualenvs/ep2017/lib/python2</td>
</tr>
<tr>
<td>1</td>
<td>0.000</td>
<td>0.000</td>
<td>0.368</td>
<td>0.368</td>
<td>/home/stephane/.virtualenvs/ep2017/lib/python2</td>
</tr>
<tr>
<td>1</td>
<td>0.000</td>
<td>0.000</td>
<td>0.367</td>
<td>0.367</td>
<td>/usr/lib64/python2.7/unittest/case.py:430(__call)</td>
</tr>
<tr>
<td>1</td>
<td>0.000</td>
<td>0.000</td>
<td>0.367</td>
<td>0.367</td>
<td>/usr/lib64/python2.7/unittest/case.py:333(__call)</td>
</tr>
<tr>
<td>1</td>
<td>0.000</td>
<td>0.000</td>
<td>0.367</td>
<td>0.367</td>
<td>/home/stephane/src/github.com/EuroPython/epcor</td>
</tr>
<tr>
<td>4</td>
<td>0.000</td>
<td>0.000</td>
<td>0.324</td>
<td>0.081</td>
<td>/home/stephane/.virtualenvs/ep2017/lib/python2</td>
</tr>
<tr>
<td>4</td>
<td>0.000</td>
<td>0.000</td>
<td>0.324</td>
<td>0.081</td>
<td>/home/stephane/.virtualenvs/ep2017/lib/python2</td>
</tr>
</tbody>
</table>
```
Deployment

We already use:

- Docker
- docker-compose
Deployment

We already use:

- Docker
- docker-compose

But for the Continuous Development we need

- Kubernetes / Docker Swarm
Monitoring

- sentry (for the logs)

```
From noreply@europython.eu
Subject [EuroPython-Web-WG] [Django] ERROR (EXTERNAL IP): Internal Server Error: /conference/paper-submission/
To web-wg@europython.eu

Internal Server Error: /conference/paper-submission/
Traceback (most recent call last):
  File "/usr/local/lib/python2.7/site-packages/django/core/handlers/base.py", line 132, in get_response
    response = wrapped_callback(request, *callback_args, **callback_kwargs)
  File "/usr/local/lib/python2.7/site-packages/django/contrib/auth/decorators.py", line 22, in _wrapped_view
    return view_func(request, *args, **kwargs)
  File "/usr/src/ep2017/conference/views.py", line 548, in paper_submission
    talk = form.save()
  File "/usr/src/ep2017/p3/forms.py", line 110, in save
    talk = super(P3SubmissionForm, self).save(*args, **kwargs)
  File "/usr/src/ep2017/conference/forms.py", line 377, in save

```

- supervisord or container orchestration tool

- shinken or nagios
State of Union

We have

- Travis
- Tests (97 tests, before just 3)
- Code coverage (32%, before just 10%)
- Started the port to PostgreSQL 9.6
- Profiling when we develop if needed
State of Union

The next steps

- Add documentation
- Remove the dead code
- Move to Django 1.11.x or Django 2.0
- Use Python $\geq 3.6$
- Use PostgreSQL
- Use Celery
- Provide API (REST) for the mobile and web applications
- Single Page App with ReactJS or VueJS (no idea)
Is there a Django Expert in the room?

WE NEED YOUR HELP!

PLEASE
How to contribute?

Sprint Code on this week-end

Join the Web Team
Share your best practices

https://tinyurl.com/django-best-practices
Questions?

https://wirtel.be

stephane@wirtel.be

@matrixise

github.com/matrixise