Writing a Python web framework in 2021

By Emmanuelle Delescolle

@EmmaDelescolle LevIT
Who am I?

- Emma
- Co-founder of LevIT
- Individual Member of the DSF
- Maintainer of DRF-Schema-Adapter and Cordy
- Maintainer of Ember-cli-crudities
Why?

• The major Python web frameworks are over 10yo. Things have changed
• Explore libraries unavailable/undocumented at the time
• Build API and Websockets into the main code
• Thought exercise
Then Vs Now

Source FreeIMG

Source WikiMedia
Then Vs Now

<table>
<thead>
<tr>
<th>Poor documentation of most libraries</th>
<th>A lot of Python packages are well documented</th>
</tr>
</thead>
<tbody>
<tr>
<td>Little sense of community</td>
<td>Friendly Python community</td>
</tr>
<tr>
<td>Rails was the &quot;reference&quot;</td>
<td>Many Python web frameworks to get inspiration from. As well as Laravel, Spray, etc...</td>
</tr>
<tr>
<td>Server-rendered-pages was the main thing to have in mind</td>
<td>Rest API's and websockets have become primary concerns</td>
</tr>
</tbody>
</table>
What about...

- Sanic
- FastAPI
- Falcon
- Quark
- autobahn
- Starlet
- Tornado
- ....
Let's go on a tour!
Tour: Project Template

django-admin startproject splendid
./manage.py startapp core

cookiecutter gh:Pylons/pyramid-cookiecutter-starter --checkout 2.0-branch

cookiecutter cordy_project
cookiecutter cordy_app

Cookiecutter

A command-line utility that creates projects from cookiecutters (project templates), e.g., creating a Python package project from a Python package project template.

- Documentation: https://cookiecutter.readthedocs.io
- GitHub: https://github.com/cookiecutter/cookiecutter
- PyPI: https://pypi.org/project/cookiecutter/
- Free and open source software: BSD license

We are proud to be an open source sponsor of PyCon 2016.
Tour: ORM

Peewee is a simple and small ORM. It has few (but expressive) concepts, making it easy to learn and intuitive to use.

- a small, expressive ORM
- python 2.7+ and 3.4+ (developed with 3.6)
- supports sqlite, mysql, postgresql and cockroachdb
- tons of extensions

```python
class Deck(Model):
    level = models.IntegerField()
    cards = JSONField()

Deck.objects.filter(level=2)
```

```python
class Deck(Model):
    level = pw.IntegerField()
    cards = JSONField()

Deck.select().filter(level=2)
```
Tour: Template engine

Jinja is a fast, expressive, extensible templating engine. Special placeholders in the template allow writing code similar to Python syntax. Then the template is passed data to render the final document.
Tour: Settings

```python
from django.conf import settings

print(settings.SOME_VAR)
```

Python Simple Settings

```
from simple_settings import settings

print(settings.SOME_VAR)
```

---

A simple way to manage your project settings.

**simple-settings** is inspired by Django's settings system but is generic for any python project.
from route.route import Route

url_map = [
    Route('infos', '/deck/{id}/info', controller='myapp.DeckViewSet', action='info'),
]
from marshmallow import fields, Schema

class UserCreateSerializer(Schema):
    username = fields.String()
    password = fields.String()
Tour:
Request/Response

if request.method == 'GET':
    do_something()
elif request.method == 'POST':
    do_something_else()
return Response(text="Here's the text of the Web page."')

WebOb

WebOb provides objects for HTTP requests and responses. Specifically it does this by wrapping the WSGI request environment and response status/headers/app_iter(body).
Tour: Command-Line

@click.command()
@click.argument('poll_ids', nargs=-1)
def hello(poll_ids=()):
    # do something

Click is a Python package for creating beautiful command line interfaces in a composable way with as little code as necessary. It's the “Command Line Interface Creation Kit”. It's highly configurable but comes with sensible defaults out of the box.
Tour: Middlewares

WSGI middlewares

IE: fancy name function wrappers

Beaker
Lightweight WSGI sessions middleware.

Beaker's starts with the Perl Cache::Cache module, which was ported for use in Myghty. Beaker was then extracted from this code, and has been substantially rewritten and modernized since.

static
This distribution provides an easy way to include static content in your WSGI applications. There is a convenience method for serving files located via pkg_resources. There are also facilities for serving mixed (static and dynamic) content using "magic" file handlers. Python 2.4 string substitution and Kid template support are provided and it is easy to roll your own handlers. Note that this distribution does not require Python 2.4 or Kid unless you want to use those types of templates.
Tour: Special *SGI implementation to support websockets

```python
def application(env, start_response):
    uwsgi.websocket_handshake(env['HTTP_SEC_WEBSOCKET_KEY'], env.get('HTTP_ORIGIN', ''))
    while True:
        msg = uwsgi.websocket_recv()
        uwsgi.websocket_send(msg)
```

The uWSGI project

The uWSGI project aims at developing a full stack for building hosting services.
Tour: Form Builder

vueteify-jsonschema-form

Create beautiful and low-effort forms that output valid data.

Based on Vue.js / Vuetify / JSON Schema.

marshmallow-jsonschema v0.11.1

JSON Schema Draft v7 (http://json-schema.org/) formatting with marshmallow
Put everything in the blender and press power?

Source flickr

Almost... But not exactly!
Missing links

Source: WikiMedia
Missing Links

- CSRF
- Authentication
- Admin?
- "glue"
Missing Links

- CSRF
- Authentication
- Admin
- "glue"

} -> Copy from Django

- -> Use "regular" form handling
- -> Cordy
Léonie, Baroness Cooreman, known by the stage name Annie Cordy, was a Belgian actress and singer. She appeared in more than 50 films from 1954. King Albert II of Belgium bestowed upon her the title of Baroness in recognition for her life's achievements.
What is Cordy?

Cordy is a way to rope-in all the libraries and components mentioned before.

It is a thought experiment

Hopefully it can serve as inspiration for the future of Django
from cordy.auth.models import BaseUser, Group
from cordy.db.models import Model

import peewee as pw

class ToDo(Model):
    description = pw.TextField()
    is_done = pw.BooleanField(null=True)

class User(BaseUser):
    groups = pw.ManyToManyField(Group, backref='users')

UserGroup = User.groups.get_through_model()
What is Cordy?

collectors.py example

class Controller(CordyController):
    
    @action(needs_id=False)
    def index(self):
        return HTMLResponse(content="<h1>Hello World</h1>"

    
class ToDoViewSet(CRUDViewSet):
        
        Model = ToDo
        pagination_class = PageNumberPagination
        page_size = 2
        filter_fields = ['is_done']
        search_fields = ['description', ]

        @authorize_with(AllowAll)
    class ToDoHTML(HTMLCRUDViewSet):
    
        Model = ToDo

        @action(needs_id=False)
        @login_required()
        def index(self, *args, **kwargs):
            return super().index(*args, **kwargs)

    
class WSController(CordyWSController):
        
        def on_connect(self):
            print('WS Connect')

        
        def on_receive(self, data):
            self.send(data['data'])

        
        def on_message(self, message):
            print('Received message:', message)

        
        def on_disconnect(self):
            print('WS Disconnected')
from routes.route import Route
from cordy.auth.controllers import AuthController
from cordy.crud.controllers import OpenAPIView
from cordy.utils import include
from myapp.controllers import Controller, ToDoViewSet, Routes, ToDoHTML

url_map = [
    *Controller.get_routes(prefix=''),
    *Routes.get_routes(),
    include(ToDoViewSet.get_routes(), '/api/v1/'),
    include(OpenAPIView.get_routes(prefix='v1', path='/api/v1/'), '/apidocs'),
    include(ToDoHTML.get_routes(prefix='todo'), ''),
    include(AuthController.get_routes(prefix='auth'), '/api'),
    Route('websocket', '/ws/', controller='myapp.WSController', action='connect'),
    Route('static', '/public/{path_info:.*}', controller='cordy.base.StaticFiles', action='serve'),
]
**What is Cordy?**

**In Action**

<table>
<thead>
<tr>
<th>Description</th>
<th>is_dose</th>
</tr>
</thead>
<tbody>
<tr>
<td>This one too...</td>
<td>✔️</td>
</tr>
<tr>
<td>Also to do</td>
<td></td>
</tr>
<tr>
<td>test broi edited</td>
<td>✔️</td>
</tr>
<tr>
<td>Pros</td>
<td>Cons</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>People involved in Python web libraries are already involved in those libraries</td>
<td>Loss of agency (dependent on library maintainers)</td>
</tr>
<tr>
<td>Resources can be dedicated to the <strong>core</strong> of the framework</td>
<td>Maintaining a framework as a whole is easier</td>
</tr>
<tr>
<td>Overall less work needed</td>
<td>Possible loss of backward compatibility with new library releases</td>
</tr>
</tbody>
</table>
Questions

Root

Curry

Shell