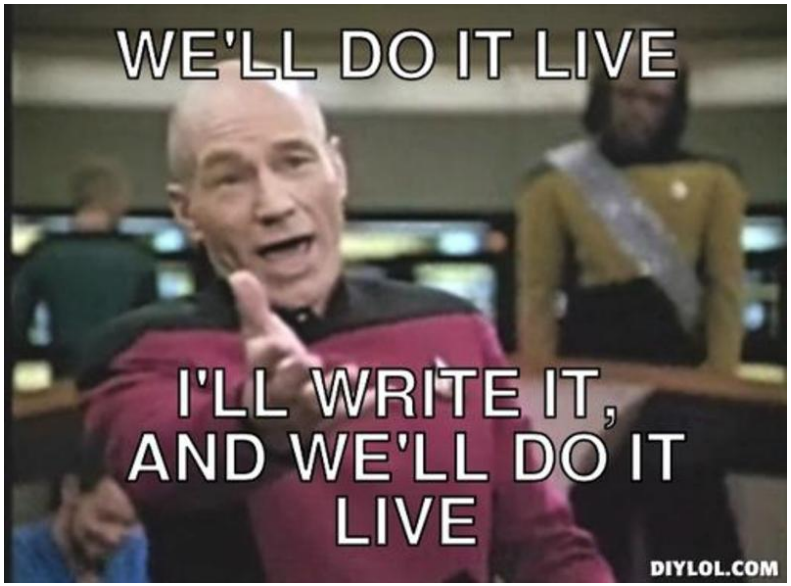
Django-Style Flask

Cody Lee
codylee@wellaware.us

git clone https://github.com/platinummonkey/flask_scale12x.git

SCALE12x
Feb 22, 2014


FAIR
WARNING


WE'LL DO IT LIVE
I'LL WRITE IT,
AND WE'LL DO IT
LIVE
DIYLOL.COM

Introduction

- Senior Network Engineer at WellAware - An oil and gas SaaS and network communications company
- Conference Chairman for the annual Texas Linux Festival
  - Come check us out June 13-14 in Austin, TX!
  - `http://texaslinuxfest.org` for more information
- Yes, I'm really a Mechanical Engineer
  - Bachelors of Science in Mechanical Engineering at Texas A&M University
  - Master's student at University of Texas at San Antonio

Outline

Get Prepared

- git clone `https://github.com/platinummonkey/flask_scale12x.git`
- Run the *viewer_setup.sh* script to setup the helper scripts.
- (Optional*) Setup and Start Titan: run *setup_titan.sh* and *start_titan.sh*.
- (Optional*) Setup virtualenv:

```
mkvirtualenv -a <repo_root> \
    -r <repo_root>/requirements.pip \
    flask_scale12x
```

- Utilize *talk_helper.sh* during the talk to switch between commits easily.

Django

- "Batteries Included"

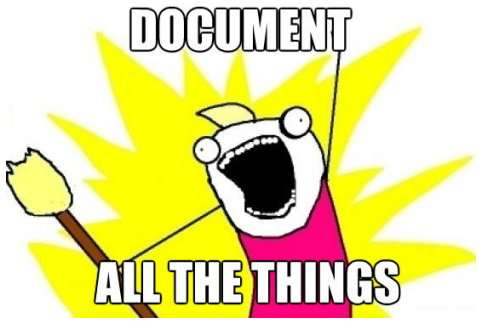
- Early Framework led to early adoption

Django

- Monolithic approach
  - Organization - Everything is an app
  - Url importing and view routing
  - Models - ORM with DB Managers for use with RDBs
  - Admin
  - Builtins
    - Auth & Permissions
    - Testing
    - Configuration

Django

- Documentation
- Templating
- Extensions
- Django-NoSQL project
  - For use with NoSQL DBs [however its just an adapter]



Flask

- Flexible by Design

- No forced organization pattern

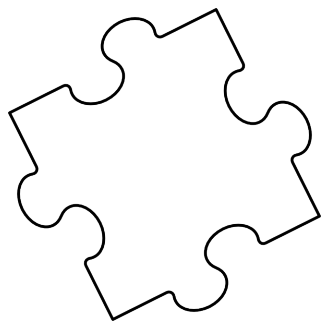
- URL routing happens at the View Definition

Flask

- Documentation
  - Great but could be better
  - Version docs are not separated
  - Changes noted explicitly within the *latest* docs.
- Templating
  - Request object is implicitly context aware
  - Jinja2
    - Done right - Doesn't unecessarily limit programmer
    - Run python code and use callables

Flask

- Extensions
  - Auth - Flask-Login
  - Permissions - Flask-Principal (if simple enough)
  - Configuration - Flask-Environment
  - Testing - Flask-Testing



Flask and Django Side-by-Side

Django

- "Batteries Included"
- Forced Organization Pattern
- Monolithic approach
- Url importing and view routing
- ORM
- Many Builtins
- Great Documentation
- Okay Templating
- Extensions available

Flask

- No Batteries
- No forced organization pattern
- Modular approach
- URL routing happens at the View Definition
- No ORM forced
- Minimal Builtins
- Good Documentation
- Better Templating
- Extensions usually required

Django Sugar

What was really nice to have?

- Organization
- Global Application Context
- Configuration
- URL and View Routing
- ORM & Models
- Auth & Sessions
- Testing
- Permissions

Organization

This where you get to decide how everything gets laid out!

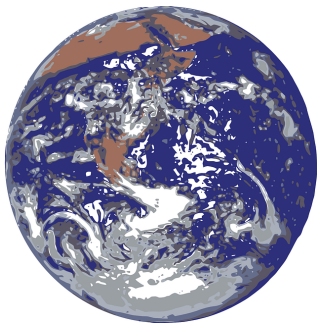
Let's just go with Django's "everything is an app" approach for this context. Feel free to do anything sensible (or not).

<project_root>

- app.py, models.py, requirements.pip, server.py, shell.py, urls.py, version.py, views.py
- auth
  - __init__.py, decorators.py, models.py, urls.py, views.py
  - permissions
    - __init__.py, decorators.py, permissions.py
  - tests...
- config
  - __init__.py, base.py, development.py, production.py, staging.py
- tools
  - apps.py, constants.py, login_manager.py, url_tools.py, sessions.py, ...

Global Application Context

- The main *app.py*!



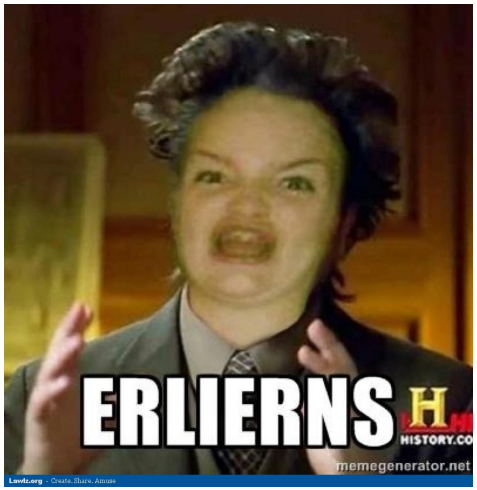
- Pretty common method, with a very slight twist
- TO THE CODE!
  683e32dd75361d4f87687b4f2235de8af455b950

Configuration

- Flask-Environments
- Utilize FLASK_ENVIRONMENT environment variable
- TO THE CODE!
  028b31d21435954cb2579a6e780d579f6d599c00

URL and View Routing

- Here is where things get sugary.
- How does Django do it? How do we want it?
  - *INSTALLED_APPS*
  - tooling
- TO THE CODE!
  ba95b15727bd4f543e907e4e12d06112d750168f



ORM & Models

- Pick a Database (SQL, NoSQL, Graph, Key-Value, ...)
- Hope that it has a sane ORM, if not...
- CREATE ONE!
- We'll use bulbs (a really cool OGM) and factory_boy
- How do we want to integrate it! Just the same way! Utilize INSTALLED_APPS loader
- TO THE CODE!
  a3ca56d346731b9ca953f0c08fda3fb99325d353

Auth & Sessions

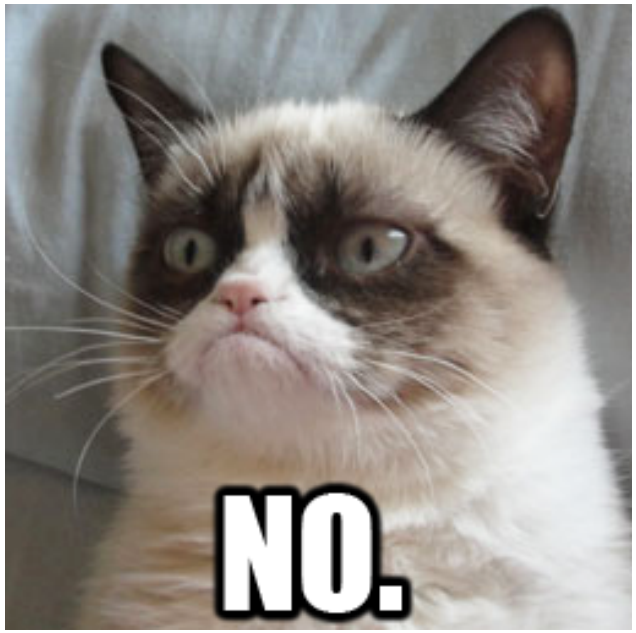
- Auth and sessions are built-in to Django, not so much in Flask
- Create or own *auth app* and models
- Implement Authentication with Flask-Login and Flask-Cache!
- TO THE CODE!
  66c2c5b14e52584892d1a28a9dd33f112d57591c

Testing

- Django has a TestRunner built-in - creates test database and... Flask doesn't
- We still have nose!
- We also have Flask-Testing!
- TO THE CODE!
  66c2c5b14e52584892d1a28a9dd33f112d57591c

Permissions

- Permisisons are built-in to Django (Though Guardian is way better)
- Create our own or use Flask-Prinicpal
- Preview available in code *66c2c5b14e52584892d1a28a9dd33f112d57591c*



Wrapping it All Up

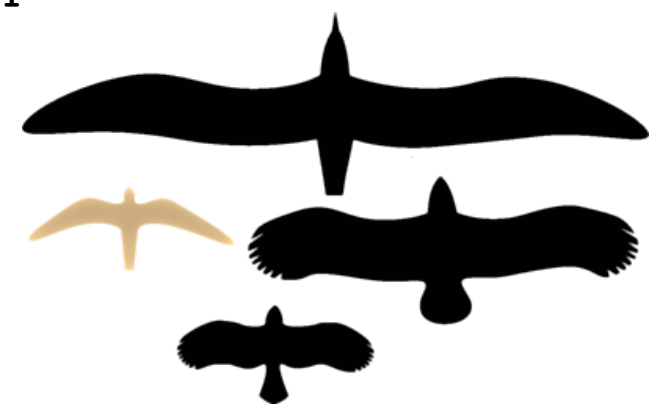
- Whoa! This looks and feels *sorta* like Django!
- What we've done today:
  - Organization
  - Global Application Context
  - Configuration
  - URL and View Routing
  - ORM & Models
  - Auth & Sessions
  - Testing
  - Permissions

So am I using this?

No.

We're moving off of Django to Falcon.

`http://falconframework.org/`



Questions?
Comments?

Please just wake up.