Packaging Django Apps

Erik Swanson

AMBITION

@swans_one

Python Packages

Django Apps

External Apps

Demonstration

Benefits

Python Packages

What is a package

- A collection of python code
- Potentially other data
- Installable through `pip`
- Likely hosted on PyPi (or other service)

setup.py

- Entry point for packaging tasks
- Call into `setuptools.setup`
- From the command line:

```
$python setup.py register sdist upload
```

Django Apps

Break up a Larger Project

- Group related functionality
- Define interfaces
- Just Python Modules

Structure

- model.py
- views.py
- forms.py
- urls.py
- admin.py

- Other python code
- Static Files
- Templates

External Apps

Examples

- south
- crispyforms
- django-debug-toolbar
- django-social-auth
- django-kittens

How Do We Make One?

- Move the code into a separate repository
- Make it a python package
- Extra considerations because of Django

Simulate a Django Environment

File	In Django Project	In External App
README	Yes	Yes
manage.py	Yes	Yes
requirements.txt	Yes	No
setup.py	No	Yes
settings.py	No	Yes
run_tests.py	No	Yes

Demonstration

Inter-App Communication

- Easy to call into external apps
- Settings
- Signals
- Management Commands

Benefits

Benefits

- Testing
- Reusability
- Collaboration
- Reusability + Collaboration = Microservices
- Generalization
- Open Source

Tools

- Migrations
- App Templates
- TravisCI / CircleCi / Jenkins
- Gemfury / DevPy

Apps at Ambition

- Onboarding
- Notifications
- Data Processing
- Competition Framework
- Entity History Tracking
- Business Calendar
- Many More