Flask

A Web Application Micro-Framework in Python

Cody Hess

Coder, Teacher, Career-Switcher

Objectives

- Know the difference between traditional and micro- frameworks
- Become familiar with the "feel" of Flask
- Know when Flask is an appropriate tool
- Write a working JSON API with Flask
- The participant is prepared to learn more independently

Overview

- What is Flask?
- What is a "Micro-Framework"?
- Web Framework Components
- Why use Flask?
- Flask VS Django
- Flask Hello World Walkthrough
- Flask VS Express
    - Hello World
    - Template
    - JSON API
- Mob Programming with Flask
    - Rock Paper Scissors API
    - Rock Paper Scissors with Global Tracking
    - Rock Paper Scissors with Sessions
- Additional Resources

- **No Prerequisites**
- **All Programming is Group Programming**

What is Flask?

Flask is a micro-framework for Python.

What is a Traditional, aka "Full-Stack" Framework?

- routes
- requests
- responses
- sessions
- ORM
- server
- security
- templates
- "Model View Controller"
- "Kitchen Sink"

1. handle the request
2. send the response

What is a Micro-Framework?

- smaller
- faster
- less stuff
- fewer opinions
- "Pay for what you use."

1. handle the request
2. send the response

Benchmarks: One JSON Serialized DB Query

- PHP (raw) 38,963
- Zend 3,965

- Flask 43.307
- Django 26,970

- Sinatra 65,509
- Rails 2,856

- Express 37,835
- Go (raw) 58,631

- C++ 210,219

Benchmarks: Fortune Cookie (Data Munging and Templating)

- PHP (raw) 39.041
- Zend 3,660

- Flask 25,051
- Django 12,483

- Sinatra 45,427
- Rails 1,889

- Express 30,656
- Go (raw) 36,324

- C++ 182,991

Why use Flask?

These are the names of some Python web frameworks:

- Django
- TurboGears
- web2py
- CubicWeb
- Giotto
- Grok
- Pylons
- Reahl
- wheezy.web
- Zope2
- Glashammer
- Karrigell
- Kiss.py
- Lino
- Nagare
- Porcupine
- Pylatte
- Tipfy
- Tornado
- watson
- webapp2
- WebBot
- WebCore
- web.py
- Webware
- Werkzeug
- WHIFF
- Bottle
- CherryPy
- Flask
- Hug
- Pyramid
- Albatross
- Aquarium
- BlueBream
- Circuits
- Falcon
- Growler
- etc
- etc
- etc

Cody's Law !!!

The three acceptable reasons to choose any tool are:

1. It is the best tool for the job.
2. It is a good enough tool that you already know.
3. You just like it more.

Reasons to Use Flask

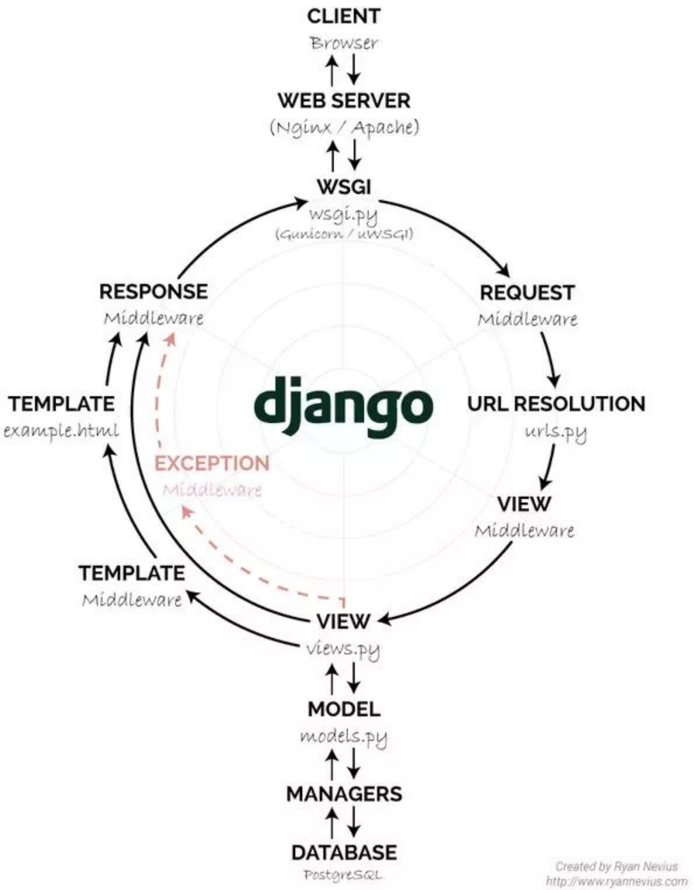
1. You like Python
2. Flask has a good book
3. Simple directory structure
4. You like how Flask "feels"

Flask VS Django

Django Directory Structure

- myproject/
    - manage.py
    - myproject/
        - urls.py
        - wsgi.py
        - settings/
    - appname/
        - models.py
        - managers.py
        - views.py
        - urls.py
        - templates/
        - static/
    - users/
        - models.py
        - views.py
        - urls.py
        - templates/
        - static/
    - static/
    - templates/
    - requirements/

django
Request-Response Cycle



CLIENT
Browser

WEB SERVER
(Nginx / Apache)

WSGI
wsgi.py
(Gunicorn / uWSGI)

REQUEST
Middleware

URL RESOLUTION
urls.py

VIEW
Middleware

VIEW
views.py

MODEL
models.py

MANAGERS

DATABASE
PostgreSQL

TEMPLATE
Middleware

TEMPLATE
example.html

EXCEPTION
Middleware

RESPONSE
Middleware

django

Created by Ryan Nevius
http://www.ryannevius.com

Flask Hello World

```
from flask import Flask

app = Flask(__name__)

@app.route(‘/’)

def index():

    return “Hello world!”
```

Flask VS Express

- Hello World
- Template
- JSON API

Mob Programming

- Rock Paper Scissors API
- Rock Paper Scissors with Global Tracking
- Rock Paper Scissors with Session Tracking

Thanks for coming!

HANDOUT HAS LINKS TO ADDITIONAL RESOURCES