Django and Flask: Understanding why and when to use each

By Eric Schles - Dev Evangelist @ Syncano

About me

- Developer Evangelist @ Syncano
- Professor @ NYU

Enjoys:

-Rock climbing

-Guitar

code for 3 months

PYTHONSYNC

Intro to Flango (Intro to Flask)

Flask: http://flask.pocoo.org/

A brief walk through of flask

A first application - “Hello world” on the fly

A real flask app - a directory for your friends

A first application - “Hello world on the fly

the code: https://github.com/EricSchles/hello_app

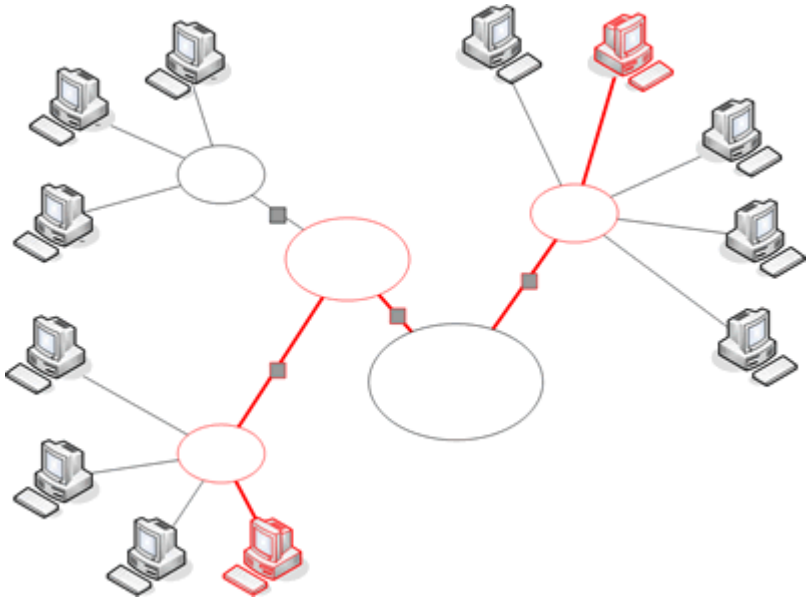
explanation of concepts:

- what’s a flask object?
- what’s a route?
- How do you run a server/what’s a server?

What's a flask object - signature

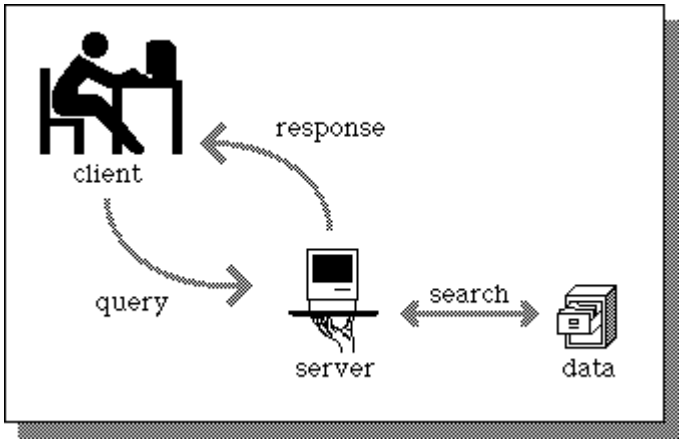
class flask.**Flask**(*import_name*, *static_path=None*, *static_url_path=None*, *static_folder='static'*, *template_folder='templates'*, *instance_path=None*, *instance_relative_config=False*)

What’s a route?



What's a server?

Def: A server is a program that serves content to users. Where users could be end users could be people or other applications.



A real flask app: a directory for your friends

-directory conventions

-Setting up the front end

-Setting up the back end

-Adding pictures - exercise

-Adding search by exact name - exercise

-Improving search by adding full text search - exercise

Directory Conventions

While flask does not enforce directory conventions it is strongly recommended that you follow these:

```
eric@ubuntu:~/Documents/syncanoStuff/directory-o-friends$ touch views.py
eric@ubuntu:~/Documents/syncanoStuff/directory-o-friends$ touch middleware.py
eric@ubuntu:~/Documents/syncanoStuff/directory-o-friends$ touch assets.py
eric@ubuntu:~/Documents/syncanoStuff/directory-o-friends$ touch forms.py
eric@ubuntu:~/Documents/syncanoStuff/directory-o-friends$ touch models.py
eric@ubuntu:~/Documents/syncanoStuff/directory-o-friends$ touch __init__.py
eric@ubuntu:~/Documents/syncanoStuff/directory-o-friends$ mkdir static
eric@ubuntu:~/Documents/syncanoStuff/directory-o-friends$ mkdir templates
eric@ubuntu:~/Documents/syncanoStuff/directory-o-friends$ mkdir static/css
eric@ubuntu:~/Documents/syncanoStuff/directory-o-friends$ mkdir static/img
eric@ubuntu:~/Documents/syncanoStuff/directory-o-friends$ mkdir static/js
eric@ubuntu:~/Documents/syncanoStuff/directory-o-friends$ mkdir templates/layouts
```

Setting up the front end

imports

Routing

Dealing with view functions

Static Files

Rendering Templates

Accessing Request Data

message flashing

understanding imports

run.py:

from app import app

-The first ‘app’ is the directory and the second is the object

runner.py:

from anything import app

routes

As we've seen - setting up routes is one of the easiest things to do in flask.

Here our routes are:

index	info/<username>
signup	directory/<username>/<person>
login	

dealing with view functions

view functions are for passing data to the templated html pages to be shown to the user.

rendering templates

Here you make use of the following functions most heavily:

render_template, request, url_for, make_response

make_response

```
def index():
    response = make_response(render_template('index.html', foo=42))
    response.headers['X-Parachutes'] = 'parachutes are cool'
    return response
```

```
@app.route('/download/<filename>')
def download(filename):
    csv_file = os.path.join(app.config['UPLOAD_FOLDER'],filename)
        with open(csv_file,"r") as f:
            csv = f.read()
    response = make_response(csv)
        response.headers['Content-Disposition'] = "attachment; filename="+filename
        return response
```

message flashing

message flashing is a great dynamic way to let your users know they did something really well, or really poorly.

Note: Please use sparingly - this isn’t ‘99

Static files

These really aren't very exciting examples include:

- pdfs
- text files
- etc.

(disappointed face)

(we don't even put pictures here)

Setting up the backend

Setting up a database

Sending data to the backend

About responses

Sessions

logging

Setting up a database - config

http://flask.pocoo.org/docs/0.10/config/

setting up a database

For SQLite3 - the manual way:

Make a database schema

Make a database file

Connect to a database

Send test data to the database

setting up the database

For SQLAlchemy - the backend agnostic way (recommended)

create your db

intialize your database

create your models in python

--Go play videogames

setting up migration

Migration is used to update your database every time your model changes.

Migration is version control for your schema

Note: If you are using a NoSQL solution you don’t need to do this.

sending data to the backend

If you used SQLAlchemy:

```
u = models.User(name=name,email=email,
password=password,picture=picture)
db.session.add(u)
db.session.commit()
#Go play more video games
```

sending data to the backend

If you used sqlite3

conn = sqlite3.connect(“database.db”)

c = conn.cursor()

c.execute(“INSERT INTO account_holder VALUES (%s,%s,%s,%s)” % name,email,password,picture)

conn.commit()

conn.close()

Adding pictures - exercise

Storing pictures in a database

sending pictures to the front end

Styling pictures

Adding search by exact name - exercise

Setting up the search bar

setting up simple search

testing and verification

adding full text search - exercise

installing necessary extensions

introduction to regex

writing our backend functions

connecting to the front end

test and verification

The downsides of flask

Django

(for perfectionists under a deadline)

Django example

Steps:

- -Setup
- -models
- -views
- -templates

Django v. Flask

VS

flask strengths

-Get things done now

-Almost no setup required

-Great for inexperienced web developers

-GREAT for:

    -hackathons

    -prototyping

    -small little websites for internal use

Django Strengths

-Get things done correctly (Object oriented)

-Set up is baked into django

-Great for experienced web developers

-GREAT for:

    -production level code under a deadline

    -building a website with lots of services

Flask weaknesses

-Other than web development primitives you get nothing

-Not many built-in debugging tools

-Code written in flask does not “scale” well

    -Not strongly object oriented

Django weaknesses

-A TON of setup before you can get going

-Assuming you know what you are doing and you are okay with the way django does things

-Very painful to re-write a lot of their templates, IE the admin panel

    -Yes, I have done this, it was awful!

Asides

Flask:

-top down design is easiest

-Build the views then the templates and finally the views

Django:

-Bottom up is easiest

-Build the models, then views, then templates

Questions….