

Web apps

Web frameworks

Software packages designed for rapid development of web applications

- **Flask**: A 'lightweight' python framework
 - No database layer
 - No (or few) library dependencies
 - Extensions available for more complicated stuff
- **Django**: Full featured python framework
- **Ruby on Rails**: Full featured framework for Ruby

Flask

Flask Includes:

- A Web Server
- API “engine”
- Templating (using Jinja2)
 - * A language for developing dynamic web pages
- Testing tools and Debugger

Installing and running flask

To install:

→ `pip install Flask`

To run (after creating a file or module):

→ By running Python:

→ `python filename.py`

→ By running Flask:

→ `export FLASK_APP=filename.py`

→ `flask run`

More installation info [here](#).
(Don't bother with virtualenv)

A simple 'hello world' app

```
from flask import Flask  
app = Flask(__name__)
```

*the name of the current module (e.g.,
__main__)*

```
@app.route("/")
```

```
def hello():
```

```
    return "Hello World!"
```

*the url that will activate the function
(e.g., hello)*

```
if __name__ == "__main__":
```

```
    app.run()
```

*runs the app on the local server
(args: host, port, debug)*

variable components in urls

```
from flask import Flask
app = Flask(__name__)

@app.route('/hello/<name>')
def hello_name(name):
    return 'Hello %s!' % name

if __name__ == '__main__':
    app.run(debug = True)
```

MVC paradigm

→ Model

- ❖ data models
- ❖ connects to the database
- ❖ handles queries

→ View

- ❖ design of the interface
- ❖ html templates

→ Controller

- ❖ application logic
- ❖ the glue between the model and the view

MVC paradigm

- **Controller:** `app.py` with routes and logic
- **Views:** html templates + static files
- **Model:** database access

flask app structure

→ Application folder:

- View: "static" folder (for js, css, images)
- View: "templates" folder (for html templates)
- Controller: app.py
- Model: flask MYSQL extension

View

- HTML files located in the templates folder
- Uses “special” syntax (Jinja)
 - Includes display logic such as conditions and loops
- Uses template “inheritance” – allowing for reuse of common HTML (e.g. headers, footers)

Model

- `from flask.mysql import MySQL`
 - MySQL is the flask / mysql connector
 - Pass queries to mysql
 - Get results from mysql

```
mysql = MySQL() #Create an instance of a flask mysql object
app.config['MYSQL_DATABASE_USER'] = 'root'
app.config['MYSQL_DATABASE_PASSWORD'] = 'None'
app.config['MYSQL_DATABASE_DB'] = 'flask'
app.config['MYSQL_DATABASE_HOST'] = 'localhost'
mysql.init_app(app)
conn = mysql.connect()
cursor = conn.cursor()
cursor.execute(query) #Executes a mysql query
cursor.fetchall() #Gets all results as a list
```

Creating an app

- Create a project directory: e.g., 311_app
- Create a templates directory under projects
- Create a static directory under projects

Create the controller (and view)

311_app.py

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def home():
    return 'Welcome to the 311 Analysis App!'

if __name__ == '__main__':
    app.run(debug = True)
```

Run it!

Make a home page template home.html (templates)

```
<!doctype html>  
<html lang="en">
```

```
  <head>
```

```
    <title>311 Analysis</title>
```

```
  </head>
```

```
  <body>
```

```
    <h1>311 Data Analysis</h1>
```

```
    {% for link in links %}
```

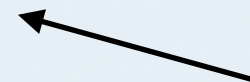
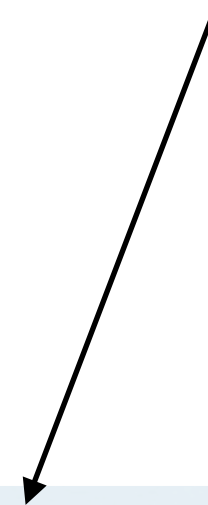
```
      <a href="{{ link.href }}">{{ link.label }}</a><br>
```

```
    {% endfor %}
```

```
  </body>
```

```
</html>
```

*this entire segment will be replaced
by html before sending it out to the
client*



*we need to pass data for each link
from the app (controller) to the
template (view)*

Generating the view

*map must be a function defined in
the app*

the marked up text

```
def get_homepage_links():  
    return [ {"href": url_for('map'), "label": "Draw the Map"},  
             {"href": url_for('analytics'), "label": "Analytics"},  
            ]
```

```
@app.route("/")  
def home():  
    session["data_loaded"] = True  
    return render_template('home.html', links=get_homepage_links())
```

*the variable and value being passed to
the template*

Using a consistent layout

Create layout.html

```
<!doctype html>
<html lang="en">
  <head>
    {% block head %}
      <title>311 Data</title>
      <link rel=stylesheet type=text/css href="{{ url_for('static', filename='style.css') }}">
    {% endblock %}
  </head>
  <body>
    <div class="header">
      <h1>311 Data Analysis</h1>
    </div>
    <div class="page">
      {% block content %}{% endblock %}
    </div>
    <div class="footer">
      {% block footer %}
        <span>All rights reserved</span>
      {% endblock %}
    </div>
```

css stylesheet

body	{ margin:0px; padding:0px;}
.header	{ display:inline-block; width:100%; height:4em; text-align: center}
.header h1	{ color: #0000ff; font-size:2em;margin:0 auto;}
.page	{ margin: 2em auto; width: 100%; padding: 0.8em; background: #fff; text-align: center;}
.page .links	{ color: #000; font-size:1.3em;margin:0 auto;}
.page .map-container	{ width:600px; height:400px; margin:0 auto;text-align: center;}
.page .actions	{ margin-top: 40px;display:inline-block;}
.footer	{ display:inline-block; width:100%; height:1em; padding: 2em 0.25em; text-align: center}
.footer span	{ color: #999; font-size:0.6em;margin:0 auto; }

modify home.html

```
{% extends "layout.html" %}
{% block content %}
    {% if session.data_loaded %}
        <div class="links">
            {% for link in links %}
                <a href="{{link.href}}">{{link.label}}</a><br>
            {% endfor %}
        </div>
    {% else %}
        <div class="app-error"> Oops! Data wasn't loaded properly</div>
    {% endif %}
{% endblock %}
```

Defining a form

*FlaskForm is defined in flask_wtf
install and import flask_wtf*

```
class MapParamsForm(FlaskForm):  
    dtfrom = DateField('DatePicker', format='%Y-%m-%d', default=date(2016,1,1))  
    dtto = DateField('DatePicker', format='%Y-%m-%d', default=date(2016,1,2))
```

from datetime import date

DateField:

from wtforms.fields.html5 import DateField

html page for the form mapparams.html

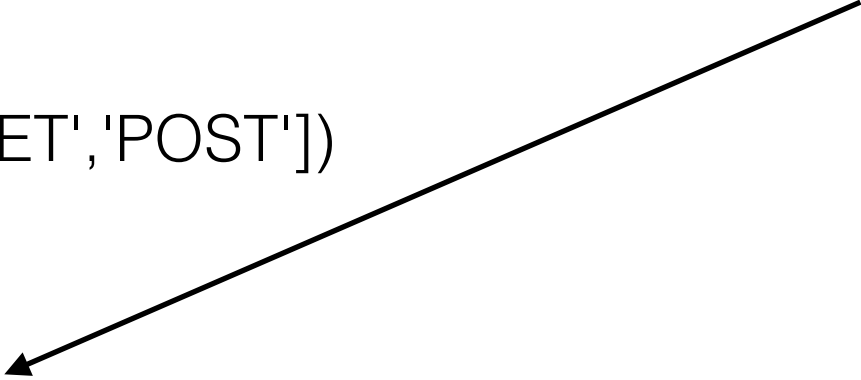
```
{% extends "layout.html" %}
{% block content %}
    <form action="#" method="post">
        {{ form.dtfom(class='datepicker') }}
        {{ form.dtto(class='datepicker') }}
        {{ form.hidden_tag() }}
        <input type="submit"/>
    </form>
{% endblock %}
```

modify the map function to render the form

*when the submit button is clicked,
this will return True*

```
@app.route("/map", methods=['GET','POST'])
def map():
    form = MapParamsForm()
    if form.validate_on_submit():
        pass

    return render_template('mapparams.html', form=form)
```



creating the map

```
if form.validate_on_submit():
    dtfrom = form.dtfrom.data.strftime('%Y-%m-%d')
    dtto = form.dtto.data.strftime('%Y-%m-%d')
    coordinates = get_data(dtfrom, dtto)
    latitudes, longitudes = ([],[])
    if (len(coordinates)>0):
        for pair in coordinates:
            latitudes.append(pair[0])
            longitudes.append(pair[1])
    gmap = gmplot.GoogleMapPlotter.from_geocode("New York",8)
    gmap.heatmap(latitudes, longitudes)
    gmap.draw('templates/mapoutput.html')
    return render_template('map.html', mapfile = 'mapoutput.html')
```

get from to dates from the form

*write a function to get latitudes
and longitudes from the database*

save the map here

map.html

the map generated by gmpplot

```
{% extends "layout.html" %}  
{% block content %}  
    <div class="map-container">  
        {% include mapfile %}  
    </div>  
    <div class="actions">  
        <a href="/map">Back</a>  
    </div>  
{% endblock %}
```


getting data from the database

database setup

```
from flaskext.mysql import MySQL
```

```
from datetime import date
```

```
mysql = MySQL()
```

```
app.config['MYSQL_DATABASE_USER'] = 'root'
```

```
app.config['MYSQL_DATABASE_PASSWORD'] = 'None'
```

```
app.config['MYSQL_DATABASE_DB'] = 'flask'
```

```
app.config['MYSQL_DATABASE_HOST'] = 'localhost'
```

```
mysql.init_app(app)
```

```
conn = mysql.connect()
```

```
cursor = conn.cursor()
```

getting data from the database

```
def get_data(dtfrom, dtto):  
    query = "select latitude, longitude  
            from incidents  
            where created_date >= '" + dtfrom + "'  
            and created_date <= '" + dtto + "';"  
    cursor.execute(query)  
    return cursor.fetchall()
```

Extending the app: selection boxes

Create a form object definition

```
class AnalyticsForm(FlaskForm):  
    attributes = SelectField('Data Attributes', choices=[('Agency', 'Agency'), ('Borough',  
'Borough'), ('Complaint_Type', 'Complaint Type')])
```

from wtforms import SelectField



And render it

```
@app.route('/analytics/', methods=['GET', 'POST'])
def analytics():
    form = AnalyticsForm()
    if form.validate_on_submit():
        pass

    return render_template('analyticsparams.html', form=form)
```

and the template

```
{% extends "layout.html" %}  
{% block content %}  
    <form action="#" method="post">  
        {{ form.attributes }}  
        {{ form.hidden_tag() }}  
        <input type="submit"/>  
    </form>  
{% endblock %}
```

Doing the analytics

```
@app.route('/analytics/', methods=['GET', 'POST'])
def analytics():
    form = AnalyticsForm()
    if form.validate_on_submit():
        import pandas
        df = get_df_data()
        column = request.form.get('attributes')
        group = df.groupby(column)
        ax = group.size().plot(kind='bar')
        fig = ax.get_figure()
        fig.savefig('static/group_by_fig.png')
        return render_template('analyticsoutput.html')

    return render_template('analyticsparams.html', form=form)
```

getting the data (sql query)

```
def get_df_data():  
    import pandas  
    query = "select unique_key, agency, complaint_type, borough  
            from incidents;"  
    cursor.execute(query)  
    data = cursor.fetchall()  
    df = pandas.DataFrame(data=list(data),columns=['Unique_key','Agency',  
        'Complaint Type','Borough'])  
    return df
```

html template with image

```
{% extends "layout.html" %}
{% block content %}
    <div class="image-container">
        
    </div>
    <div class="actions">
        <a href="/analytics">Back</a>
    </div>
{% endblock %}
```