CS1520 Recitation:

Flask 1: Routing

•••

Jeongmin Lee

Routing

Routing

Routing is about how URL and resources are linked!

- URL : Uniform Resource Locator
  e.g., http://cs.pitt.edu/index.html
- Resource: can be anything. image, file, html,
  even a piece of function in Python.

Routing

- Important job of Flask is handling user’s request

Routing

- Important job of Flask is handling user’s request
- Request is coming through URL
- **Routing** is a mechanism that handles **URL to specific function** in your code [<- Important aspect!]

First Application

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

@app.route('/')
def hello_world():
    return 'Hello World!'

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

First Application

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

@app.route('/')
def hello_world():
    return 'Hello World!'

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

First Application

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

@app.route('/')
def hello_world():
    return 'Hello World!'

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

- URL of '/' i bound to the hello_world() function.
- Hence, user requested '/' URL, the hello_world() function will be run and its results will be rendered in the browser.

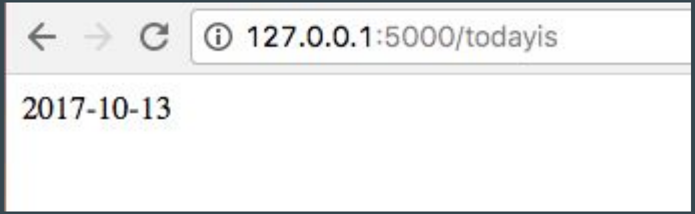
Let’s have more routes!

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

@app.route('/')
def hello_world():
    return 'Hello World!'

@app.route('/todayis')
def get_today_date():
    return str(datetime.date.today())

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



```
2017-10-13
```

Variable Rules

Variable Rules

- Key idea: Build a URL dynamically.

Variable Rules

- Key idea: Build a URL dynamically.
- Use python’s variable to be the part of URL!

Variable Rules

- Key idea: Build a URL dynamically.
- Use python's variable to be the part of URL!
- Variable Rules are passed as a keyword argument to the function with which rule is associated.

Variable Rules

```
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)
```

- Variable Rules are passed as a keyword argument to the function with which rule is associated.

(type: string)

Variable Rules

- You can use other type of variable.
- Integer and Floating Point numbers.
- Path (string with slash ‘/’)

Variable Rules

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

@app.route('/blog/<int:postID>')
def show_blog(postID):
   return 'Blog Number %d' % postID

@app.route('/rev/<float:revNo>')
def revision(revNo):
   return 'Revision Number %f' % revNo

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

Integer and Floating Point

- <int:postID> will cast with coming string part of URL into type of integer.
- Same for Float.

Variable Rules

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

@app.route('/flask')
def hello_flask():
   return 'Hello Flask'

@app.route('/python/')
def hello_python():
   return 'Hello Python'

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

Path:

- In computer world these are different:
  - /python/ and /python

- But Flask will treat them same for trailing slash (/)

Interacting with user’s data: HTML Form + Request Object

Interacting with user

- One way to interact with user is to have a form in HTML and **get what user typed** in the form as variable in your code

Interacting with user

- We can do this with Flask!
  - 1. Render a form
  - 2. Get user’s response with request object
  - 3. Show result back to user

student.html (put into templates/ folder)

```
<html>
   <body>

      <form action = "http://localhost:5000/result" method = "POST">
         <p>Name <input type = "text" name = "name" /></p>
         <p>Email <input type = "text" name = "email" /></p>
         <p>CS score <input type = "text" name = "cs" /></p>
         <p>Math score <input type ="text" name = "math" /></p>
         <p><input type = "submit" value = "submit" /></p>
      </form>

   </body>
</html>
```

Example from: https://pythonspot.com

result.html (put into templates/ folder)

```
<!doctype html>
<html>
   <body>
      <h1>Result </h1>
      <table border = 1>
         {% for key, value in result.items() %}

            <tr>
               <th> {{ key }} </th>
               <td> {{ value }} </td>
            </tr>

         {% endfor %}
      </table>

   </body>
</html>
```

Example from: https://pythonspot.com

main.py

```
from flask import Flask, render_template, request
app = Flask(__name__)

@app.route('/')
def student():
   return render_template('student.html')

@app.route('/result',methods = ['POST', 'GET'])
def result():
   if request.method == 'POST':

      req_dic = request.form.to_dict()

      print('result:{}'.format(req_dic))
      return render_template("result.html", result = req_dic )

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

Example from: https://pythonspot.com

Request object

- Import: from flask import request
- Access
    - request.form : POST Method
        - request.form[‘key_name’]
            - ‘key_name’ : form’s NAME field

Request object

- request.args: GET Method
  - parameters submitted from URL
  - request.args.get(‘key_name’)
    - ‘key_name’: part of URL (google.com/index.html?key=value)