Server Side Languages

Web Design & Development

Day 7

Lecture Overview

- 7.1 Language basics
- 7.2 Flask
- 7.3 URL Routing
- 7.4 Sessions
- 7.5 Files

Python Language

Review of Python basics:

Whitespace (and tabs) matter - indents determine levels. No {} or ;

Import libraries to include other functions and classes

"class" to create user-defined classes

"def" for user-defined functions

Loops "for *iterator* in *collection*:" *iterator.property*

(especially, watch at the end of loops - tabs matter inside/outside loop)

Conditionals "if *truecondition*:" *statement* "else:" *otherstatement*

#comments

Flask Micro Framework

Features:

- built in development server and debugger
- integrated unit testing support
- RESTful request dispatching
- uses Jinja2 templating
- support for secure cookies (client side sessions)
- 100% WSGI 1.0 compliant
- Unicode based
- extensively documented

Install Flask on Mac

Under your Class Exercises directory, create a subdirectory "flask"

Open Terminal prompt, cd to flask directory

```
//do not execute double-slash, comments only
//This installs the virtualenv so Python runs in a separate context
sudo easy_install virtualenv
//This sets up a virtual environment named venv
virtualenv venv
New python executable in venv/bin/python
Installing distribute.............done.
//The next line is dot space venv/bin/activate
. venv/bin/activate
//pip is a package manager that downloads and installs Flask
pip install Flask
```

This has installed the libraries for Flask that we will use in our project.

Exercise 7.1: Create Hello World Flask Application

Open up your text editor, and create a file named "hello.py" in the flask project directory. Make sure to pay attention to tabs (remember Python whitespaces).

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

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

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

Save the file, and launch the application, then browse http://localhost:5000

From Terminal prompt in flask directory:

```
python hello.py
 * Running on http://127.0.0.1:5000/
127.0.0.1 - - [26/Feb/2014 17:48:37] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Feb/2014 17:48:37] "GET /favicon.ico HTTP/1.1" 404 -
```

Importing Libraries

Flask imports Python libraries to extend the functionality of the framework.

```
from flask import Flask
  Flask #Default Flask library
  url_for #Build a URL
  request #Handle HTTP Requests
  make_response #Handle sending HTTP responses, cookies
  render_template #Jinja2 Templates
  Markup #Generate escaped HTML, or wiki-like markups
  escape #Generate escaped HTML (it not using templates)
  redirect #HTTP Redirects
  session #Support for sessions
  os #Filesystem or OS functions
  current_app #app context functions
  sqlalchemy #Database access, ORM
```

URL Routing

As in other frameworks, we need to define routes based on the URI requested (http://localhost/ or http://localhost/user/profile/10)
and HTTP Request method (GET, POST, PUT, DELETE)

We need to import the "request" library to access any request methods.

```
from flask import request
```

Route HTTP Request Example: (implied GET if not specified)

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        do_the_login()
    else:
        show_the_login_form()
```

URL Variables

Add variable segments to your URL.

Delimit variables with <variable>

Optional Convert Variable Types: int, float, path (accepts slashes)

```
@app.route('/user/<username>')
def show_user_profile(username):
    # show the user profile for that user
    return 'User %s' % username

@app.route('/post/<int:post_id>')
def show_post(post_id):
    # show the post with the given id, the id is an integer
    return 'Post %d' % post_id
```

Request GET Parameters

To access GET parameters

Forms posted with GET method or querystring, http://localhost/login?user=bob

Use request.args.get('key')

```
searchword = request.args.get('user')
```

Request POST Parameters

To access POST form fields or GET parameters

Use request.form['keyname'] to access POST fields.

```
@app.route('/login', methods=['POST', 'GET'])
def login():
    error = None
    if request.method == 'POST':
        if valid_login(request.form['username'],
                       request.form['password']):
            return log_the_user_in(request.form['username'])
    else:
        error = 'Invalid username/password'
# the code below is executed if the request method
# was GET or the credentials were invalid
return render_template('login.html', error=error)
```

Exercise 7.3: Redirects

Add redirect for index URL that redirects from / to /hello/Flask

Add the following code:

```
import redirect
@app.route('/')
def index():
   return redirect('/hello/Flask')
```

Sessions

Python does not support HTTP sessions by default, Flask provides a library for sessions.

Sessions stored as a key/value pair

```
import session
session['username']='Joe' #Set session variable
session.pop('username', None) #Unset session variable

if session.has_key('un'):
   ...
```

Uploads

Flask File Upload

```
from flask import request

@app.route('/upload', methods=['GET', 'POST'])
def upload_file():
    if request.method == 'POST':
        f = request.files['the_file']
        f.save('/var/www/uploads/uploaded_file.txt')
```

Questions?

Lab 7: Refactor Your Php Lab 2 in Flask

Refer to FSO for Lab7 and Screencast 7 for this assignment