Python Tutorial 5

MVC, Web Framework, Flask

Haesun Park, haesunrpark@gmail.com,

1. Web Framework
2. MVC
3. Flask
4. example

Web Framework

Flask 맛보기

- 최근 인기가 높은 경량 웹 (어플리케이션) 프레임워크 입니다.(microframework)
- 아나콘다에 배포판에 포함되어 있습니다.

'__main__'

데코레이터

```
from flask import Flask

app = Flask(__name__)

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

app.run()
```

- 브라우저에서 http://localhost:5000/ 에 접속합니다.
- run() 메소드에 host, port 파라메타를 사용하여 서버 아이피와 포트를 지정할 수 있습니다.

Web Framework

https://www.djangoproject.com

http://flask.pocoo.org

http://www.pylonsproject.org

http://bottlepy.org

http://www.tornadoweb.org

http://turbogears.org

like Struts, Spring in Java and Laravel, CodeIgniter in PHP

Web Application Server

http://www.cherrypy.org

https://twistedmatrix.com

http://gunicorn.org

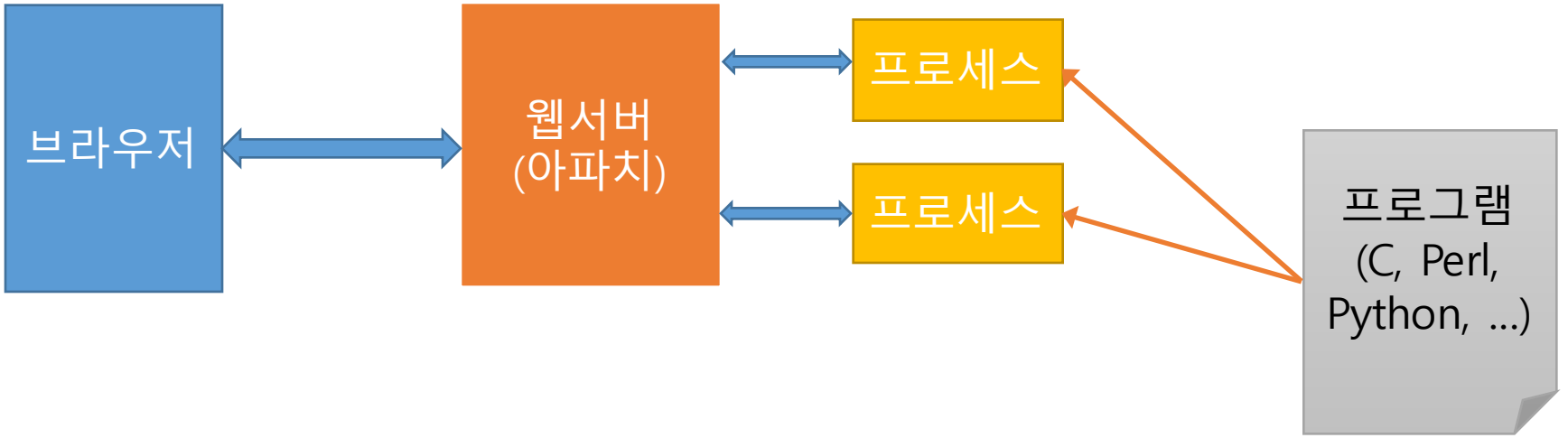
https://uwsgi-docs.readthedocs.io

http://www.tornadoweb.org

like Tomcat, JBoss in Java

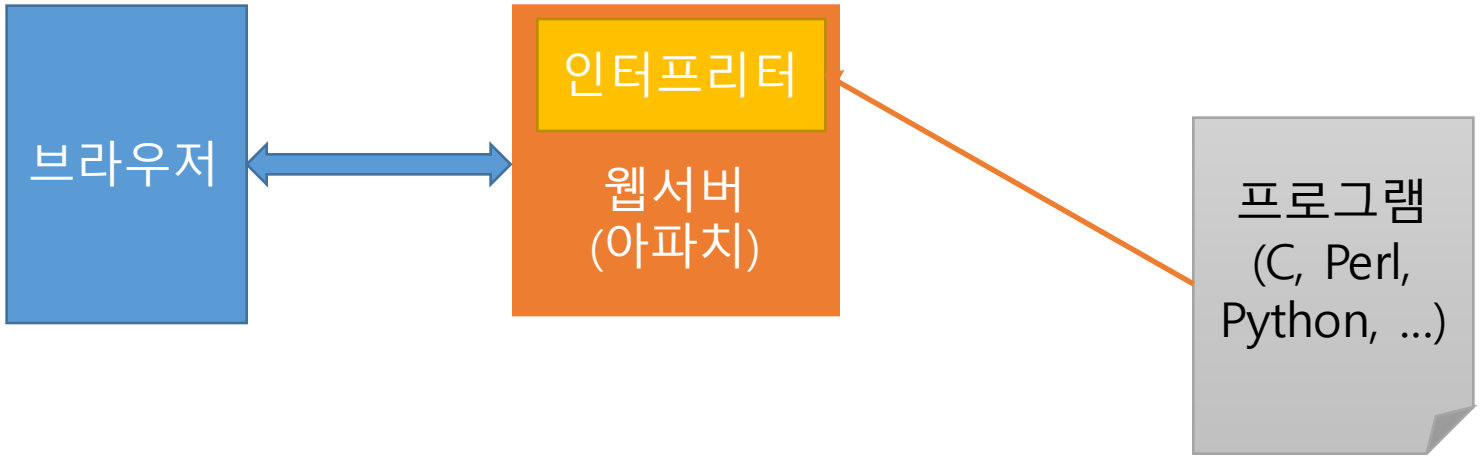
CGI(Common Gateway Interface)

- https://ko.wikipedia.org/wiki/공용_게이트웨이_인터페이스
- 언어에 구애 받지 않습니다.
- 매 호출마다 프로세스가 새로 시작됩니다.
- 리퀘스트 간의 상태가 유지되지 않습니다.
- 이제 사용하지 않습니다.



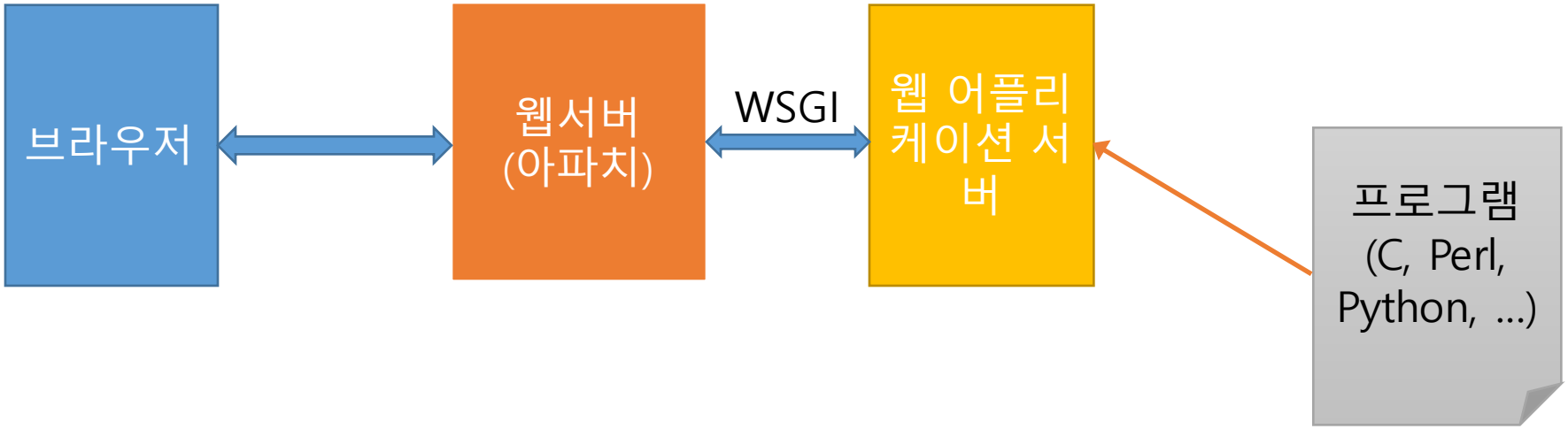
인터프리터 내장

- PHP에서 주로 사용합니다.(mod_php)
- Perl의 경우 사용자층이 매우 작습니다.(mod_perl)
- 인터프리터가 아파치 프로세스에 내장되어 있습니다.
- 매 호출마다 프로세스가 시작되지 않습니다.
- pre-fork 방식일 경우 메모리 사용량이 높습니다.



WSGI(Web Server Gateway Interface)

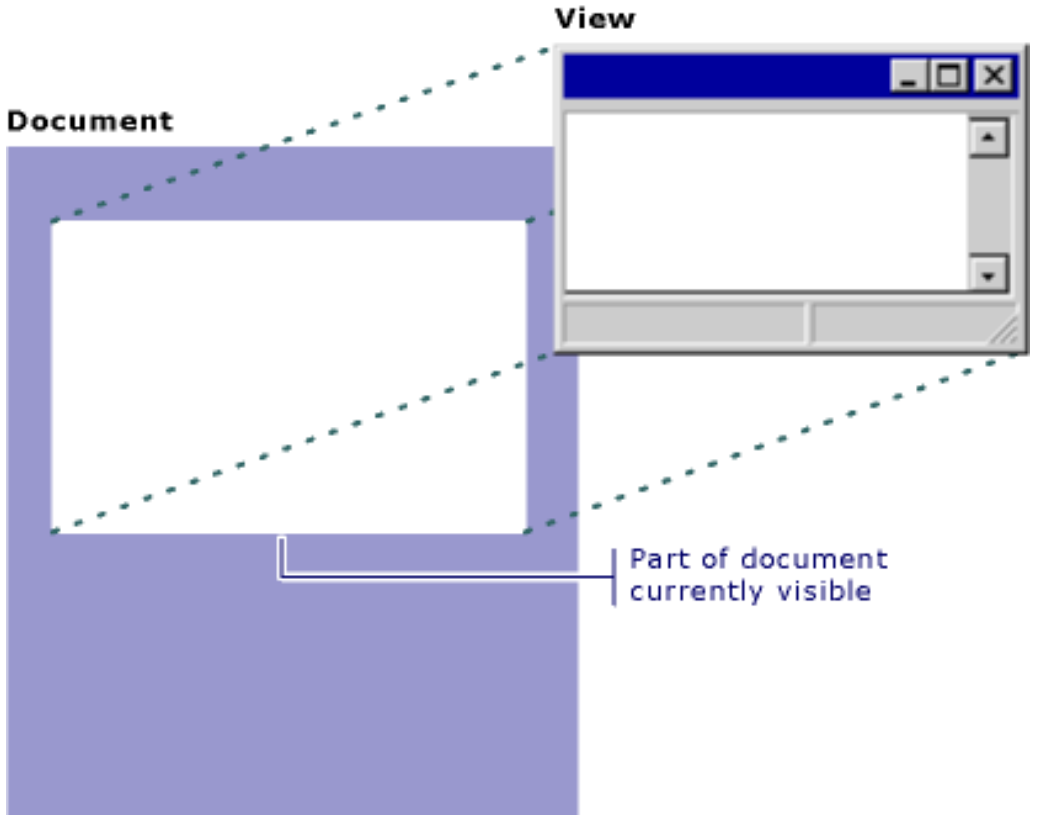
- 파이썬의 웹 어플리케이션 서버(WAS)와 웹서버간의 인터페이스 규약입니다.
- 어플리케이션 서버가 브라우저의 리퀘스트 간의 상태를 관리할 수 있습니다.
- 자바와 파이썬 등에서 대부분의 경우 채택됩니다.
- 웹서버에 mod_wsgi 모듈을 적용합니다.
- 파이썬의 경우 WAS 를 WSGI 서버라고 부르기도 합니다.



MVC

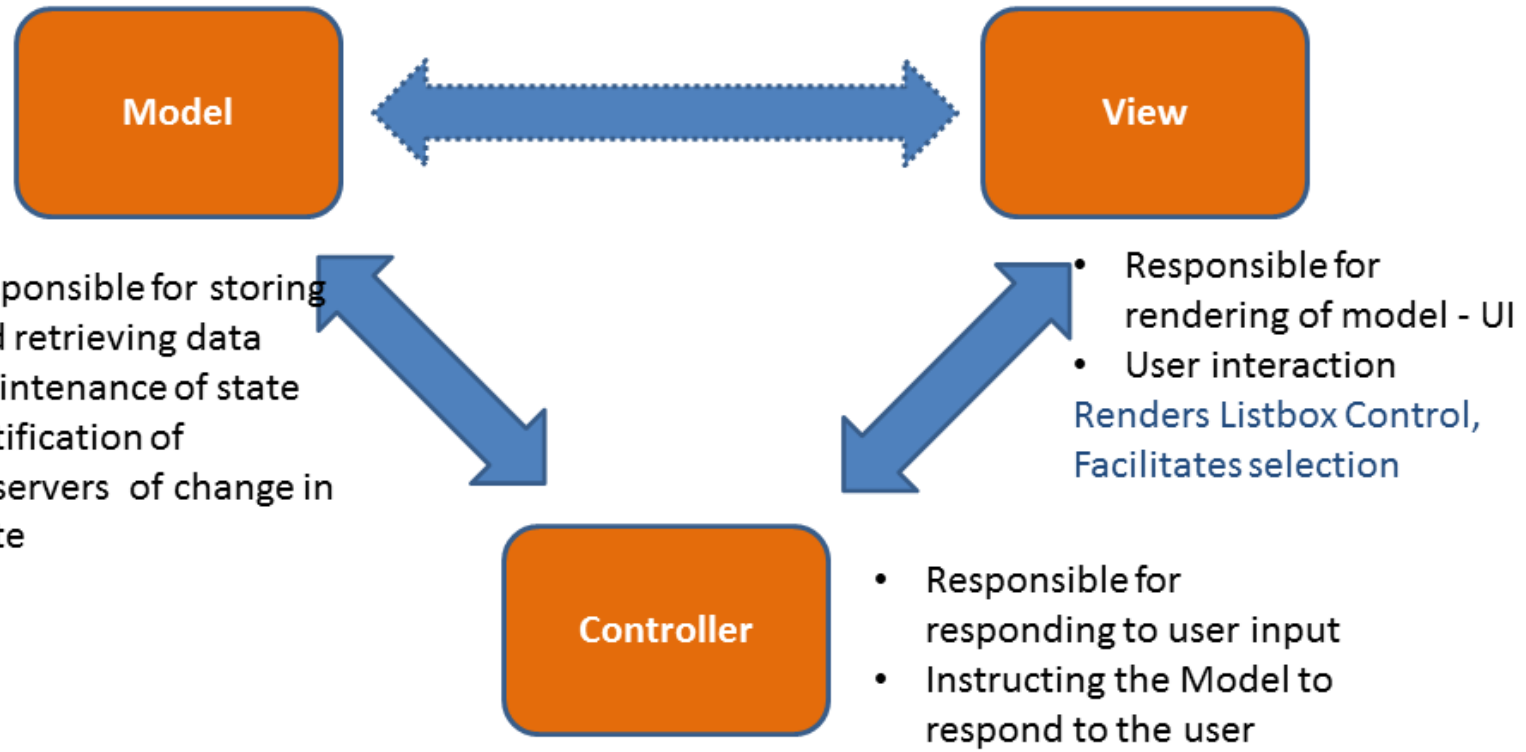
Document/View Pattern

- Microsoft MFC 프로그래밍 아키텍처
- 다큐먼트는 파일을 읽거나 쓰는 등의 데이터와 관련된 작업을 책임집니다.
- 데이터를 화면에 표시하고 사용자에게 인터페이스를 제공합니다.



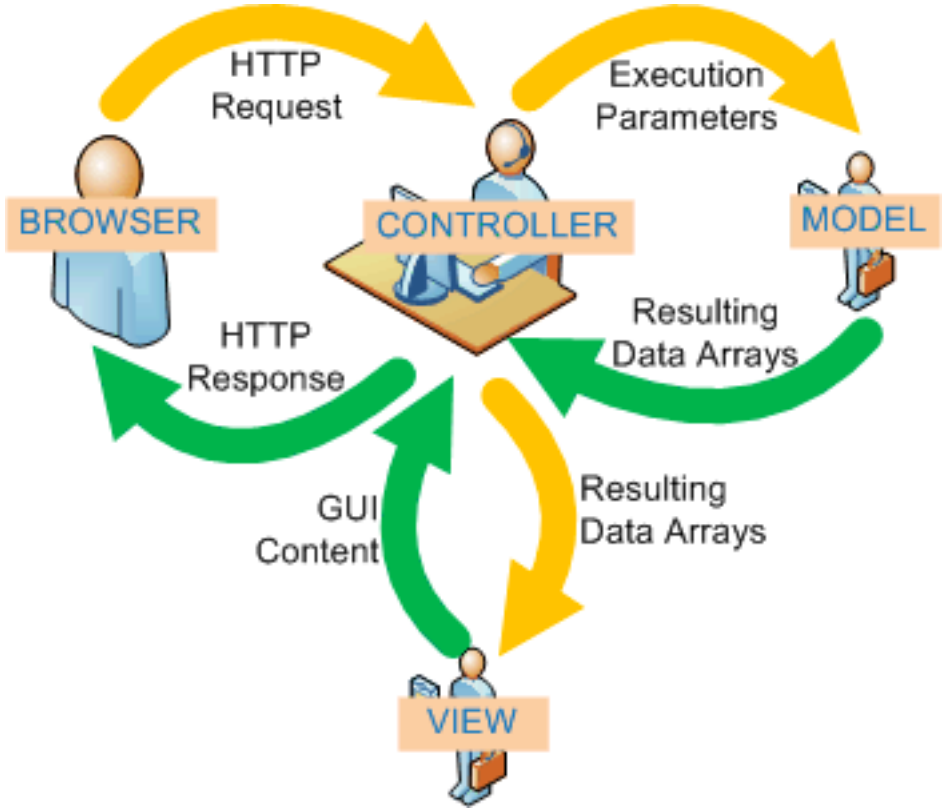
MVC Pattern

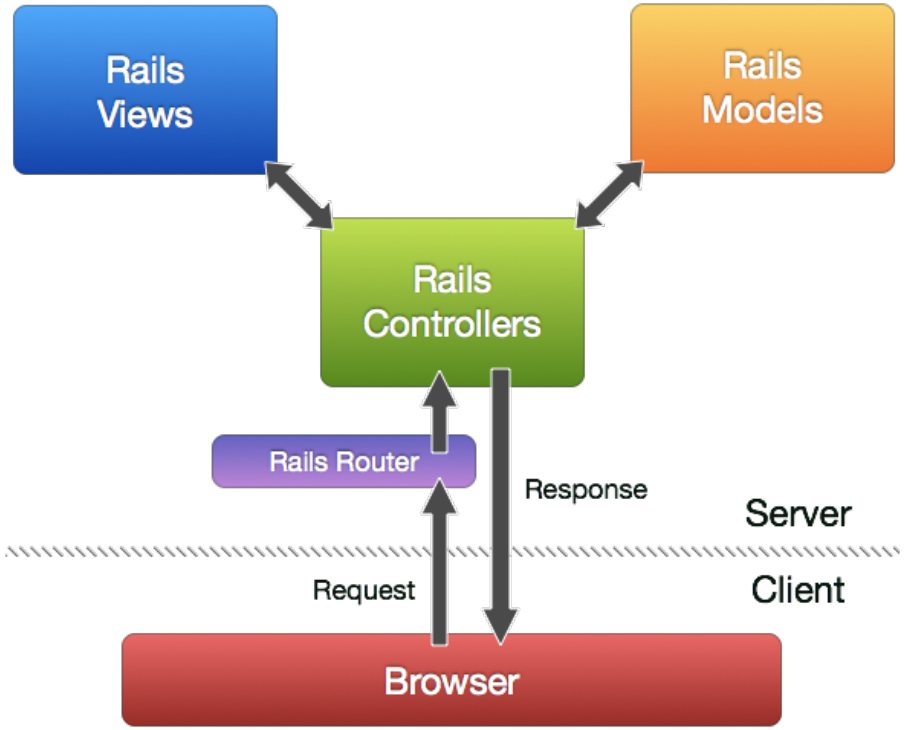
Model View Controller (MVC) Arch Pattern



MVC Architecture

- 웹 프레임워크마다 구현 방식이 조금씩 다를 수 있습니다.
- 대부분 콘트롤러가 모델과 뷰를 중재하는 역할을 담당합니다.
- 자바의 Spring과 루비의 Rails가 채택하면서 널리 알려지게 되었습니다.



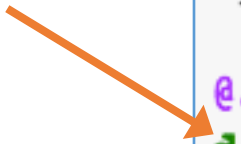


Flask

라우팅

- 브라우저에서 요청한 URL 주소를 플래스크 어플리케이션의 메소드에 연결합니다.
- __name__ : 독자적으로 실행할 땐 '__main__' 이 되고 다른 프로그램에 의해 불려졌을 때는 (모듈) 파일이름이 됩니다.

Controller



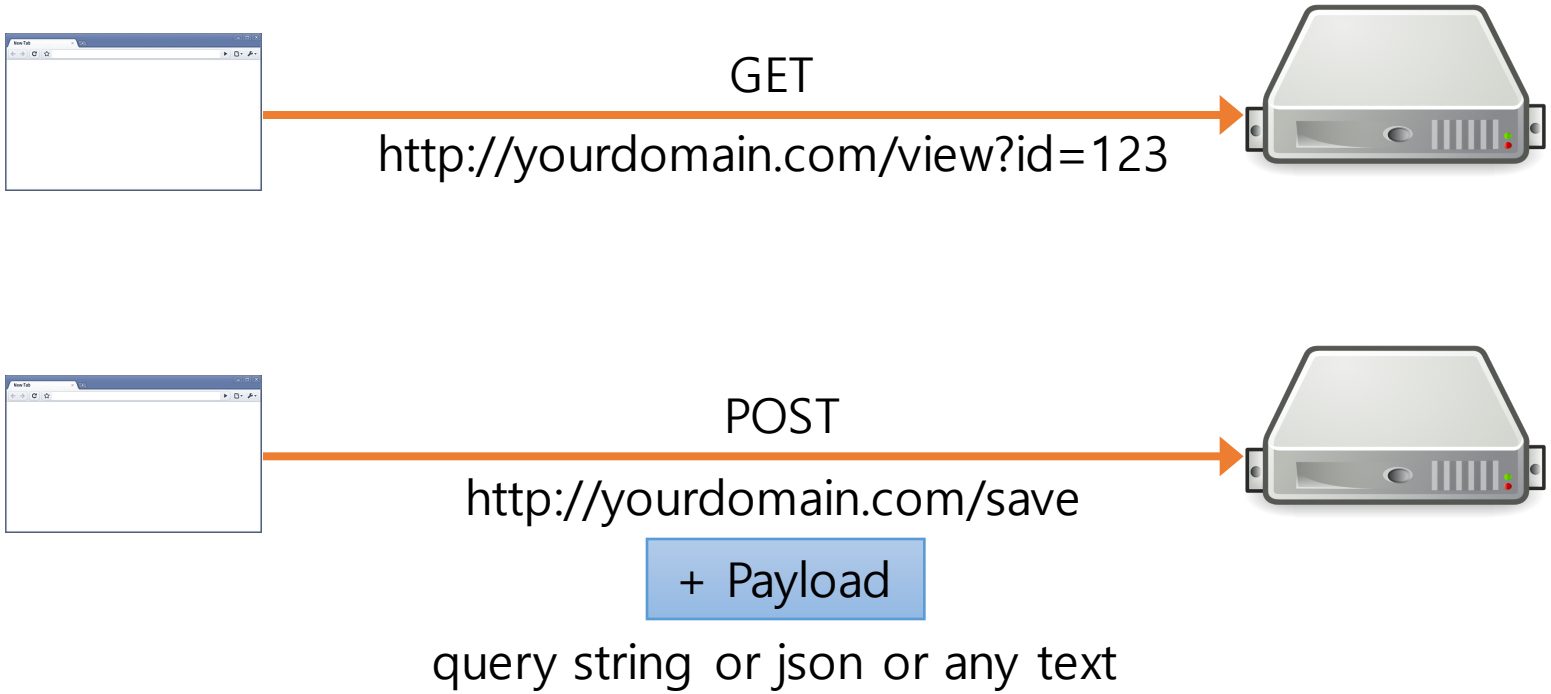
```python
from flask import Flask

app = Flask(__name__)

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

@app.route('/view/<prod_name>')
def view_product(prod_name):
    return 'product name: %s' % prod_name

app.run()
```

변수 매핑

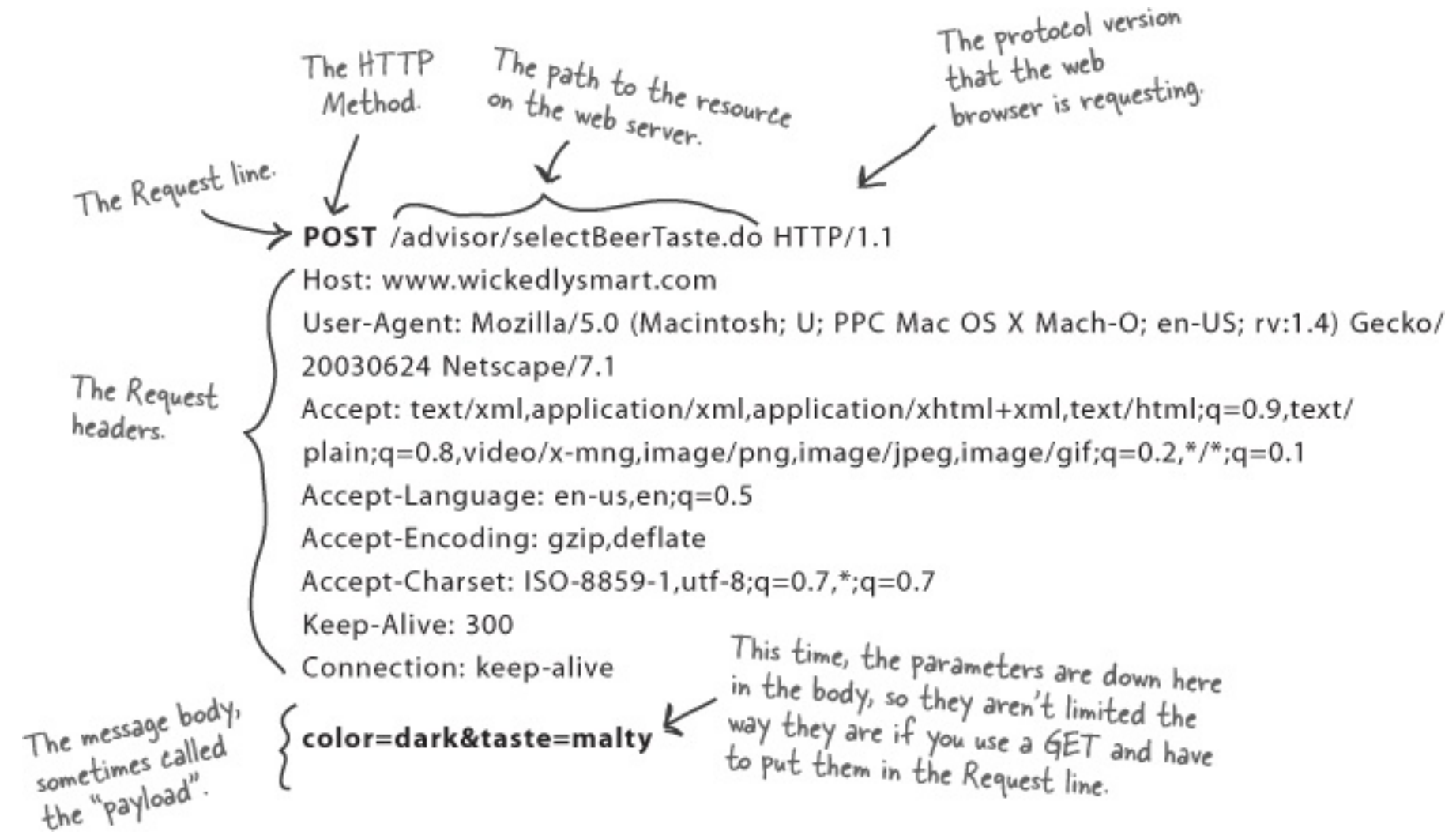
- 변수 타입 지정 가능(string, int, float, path). path로 지정된 변수는 슬래시를 포함할 수 있습니다.
- 주피터 노트북 데모

```
from flask import Flask

app = Flask(__name__)

@app.route('/get/<int:prod_id>')
def get_prod_id(prod_id):
    return 'product id: %d' % prod_id

@app.route('/file/<path:file>')
def view_path(file):
    return 'file path: %s' % file

@app.route('/info/<category>/<prod_name>')
def show_prodinfo(category, prod_name):
    return 'product info: %s(%s)' % (prod_name, category)

app.run()
```

GET & POST


GET
http://yourdomain.com/view?id=123
POST
http://yourdomain.com/save
+ Payload
query string or json or any text

HTTP Request Message

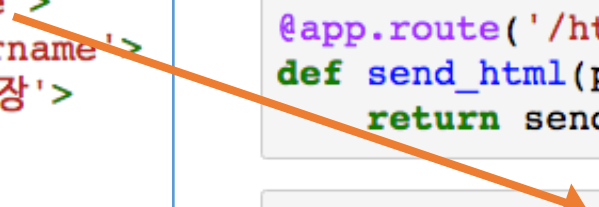


리퀘스트 메소드

- Request Method는 크게 GET과 POST 두가지가 있습니다.
- GET은 HTML 폼의 데이터를 쿼리스트링의 파라메타로 실어 나릅니다.
- POST는 HTML 폼의 데이터를 HTTP 본문에 실어 나릅니다.

post.html

```html
<html>
    <body>
        <form method='POST' action='/save'>
            <input type='text' name='username'>
            <input type='submit' name='저장'>
        </form>
    </body>
</html>
```



```python
from flask import Flask, request, send_from_directory

app = Flask(__name__)

@app.route('/html/<path:path>')
def send_html(path):
    return send_from_directory('html', path)

@app.route('/save', methods=['GET', 'POST'])
def save():
    if request.method == 'POST':
        return 'This is POST Request'
    else:
        return 'This is GET Request'

app.run()
```

폼 데이터 접근

- POST 데이터는 request.form 딕셔너리로 전달됩니다.
- GET 데이터는 request.args 딕셔너리로 전달됩니다.
- 쥬피터 노트북으로 데모

post.html

```html
<html>
    <body>
        <form method='POST' action='/save'>
            <input type='text' name='username'>
            <input type='submit' name='저장'>
        </form>
    </body>
</html>
```

```python
from flask import Flask, request, send_from_directory

app = Flask(__name__)

@app.route('/html/<path:path>')
def send_html(path):
    return send_from_directory('html', path)

@app.route('/save', methods=['GET', 'POST'])
def save():
    if request.method == 'POST':
        return 'This is POST: %s' % request.form.get('username', '')
    else:
        return 'This is GET: %s' % request.args.get('username', '')

app.run()
```

Jinja

- http://jinja.pocoo.org/
- 파이썬 템플릿 엔진
- 표현식은 {%..%} 으로 표시. e.g {% **for** user **in** users %}
- 변수는 {{ .. }} 으로 표시. e.g {{ username }}

post.html

```
<html>
    <body>
        <form method='POST' action='/save'>
            <input type='text' name='username'>
            <input type='submit' name='저장'>
        </form>
    </body>
</html>
```

View

/templates/save_complete.html

```
<!doctype html>
<html>
<body>
<h1>Save Complete</h1>
<h2>Hello {{ data['username'] }}!</h2>
</body>
</html>
```

```
from flask import Flask, request, send_from_directory, render_template

app = Flask(__name__)

@app.route('/html/<path:path>')
def send_html(path):
    return send_from_directory('html', path)

@app.route('/save', methods=['GET', 'POST'])
def save():
    return render_template('save_complete.html', data=request.form)

app.run()
```

sqlite3

- https://www.sqlite.org , Battery Included!
- serverless and embedded database server
- 쥬피터 노트북으로 데모

```
import os
import sqlite3
```

```
from flask import Flask, request, send_from_directory, render_template
```

```
app = Flask(__name__)
app.config.from_object(__name__)
```

```
app.config.update(dict(
        DATABASE=os.path.join(app.root_path, 'flask_test.db')
    ))
```

```
def init_db():
    db = sqlite3.connect(app.config['DATABASE'])
    with app.open_resource('schema.sql', mode='r') as f:
        db.cursor().executescript(f.read())
    db.commit()
    db.close()
init_db()
```

Controller+Model

```
from flask import Flask, request, send_from_directory, \
    render_template, g, redirect, url_for
```

- redirect: 브라우저를 다른 주소로 이동시킴
- url_for: 콘트롤러에 해당하는 URL 생성
- 쥬피터 노트북으로 데모

list.html

```
<!doctype html>
<html>
    <body>
        <h1>전체 사용자<h1>
        <ul>
            {% for user in users %}
            <li>{{ user.username }}</li>
            {% endfor %}
        </ul>
    </body>
</html>
```

```
def connect_db():
    if not hasattr(g, 'db_con'):
        g.db_con = sqlite3.connect(app.config['DATABASE'])
        g.db_con.row_factory = sqlite3.Row
    return g.db_con
```

```
@app.teardown_appcontext
def close_db(error):
    if hasattr(g, 'db_con'):
        g.db_con.close()
```

Global Context

```
@app.route('/html/<path:path>')
def send_html(path):
    return send_from_directory('html', path)
```

```
@app.route('/save', methods=['POST'])
def save():
    db = connect_db()
    db.execute('insert into entries (username) values (?)', \
                [request.form['username']])
    db.commit()
    return redirect(url_for('list'))
```

```
@app.route('/list')
def list():
    db = connect_db()
    cur = db.execute('select username from entries')
    usernames = cur.fetchall()
    return render_template('list.html', users=usernames)
```

```
app.run()
```

템플릿 상속

- redirect: 브라우저를 다른 주소로 이동시킴
- url_for: 콘트롤러에 해당하는 URL 생성
- 쥬피터 노트북으로 데모

layout.html

```
<!doctype html>
<html>
    <body>
        <h1>Flask Test App</h1>
        {% block body %}
        {% endblock %}
    </body>
</html>
```

list2.html

```
{% extends 'layout.html' %}
{% block body %}
<h3>전체 사용자<h3>
<ul>
    {% for user in users %}
    <li>{{ user.username }}</li>
    {% endfor %}
</ul>
{% endblock %}
```

```
def connect_db():
    if not hasattr(g, 'db_con'):
        g.db_con = sqlite3.connect(app.config['DATABASE'])
        g.db_con.row_factory = sqlite3.Row
    return g.db_con
```

```
@app.teardown_appcontext
def close_db(error):
    if hasattr(g, 'db_con'):
        g.db_con.close()
```

```
@app.route('/list')
def list():
    db = connect_db()
    cur = db.execute('select username from entries')
    usernames = cur.fetchall()
    return render_template('list2.html', users=usernames)
```

```
app.run()
```

Model

```python
def connect_db():
    if not hasattr(g, 'db_con'):
        g.db_con = sqlite3.connect(app.config['DATABASE'])
        g.db_con.row_factory = sqlite3.Row
    return g.db_con
```

```python
@app.teardown_appcontext
def close_db(error):
    if hasattr(g, 'db_con'):
        g.db_con.close()
```

```python
@app.route('/list')
def list():
    db = connect_db()
    u = User(db)
    usernames = u.get_list()
    return render_template('list2.html', users=username
```

```python
app.run()
```

- models 폴더 하위에 모델 클래스 생성
- from models.user import User

modes/user.py

```python
class User(object):

    def __init__(self, db):
        self.db = db

    def save(self, name):
        self.db.execute('insert into entries (username) values (?)', [name])
        self.db.commit()
        return True

    def get_list(self):
        cur = self.db.execute('select username from entries')
        return cur.fetchall()
```

ORM

- Object-Relational Mapping
- 데이터베이스의 데이터를 오브젝트화 합니다.
- 하나의 클래스는 하나의 테이블은 대표합니다.
- 보통 하나의 Row가 하나의 인스턴스입니다.

장점

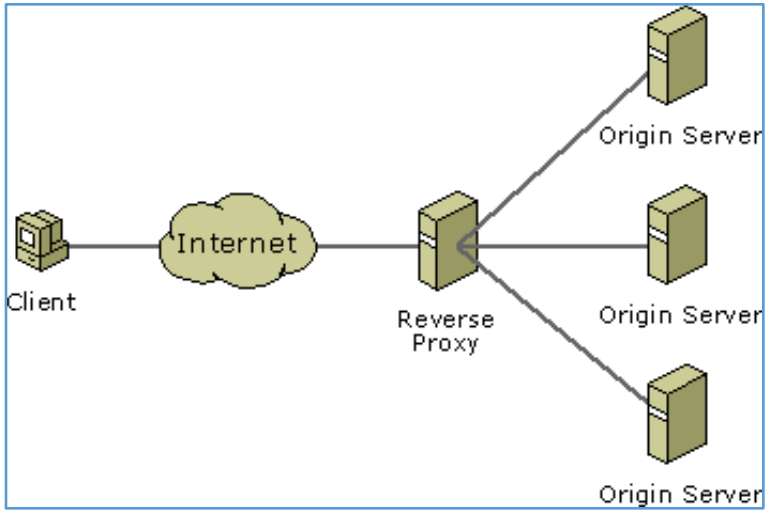
- CRUD 작업을 자동화한다.
- 일반적으로 스키마 반영이 용이하다.
- DB 쿼리 최적화가 평준화된다.

단점

- 대량의 오브젝트를 핸들링할 경우 느리다.
- 라이브러리마다 새로 배워야 한다.
- DB 쿼리를 세밀하게 튜닝하기 어렵다.

WSGI Deploy

gunicorn, gevent, twsited 등을 이용하여 WSGI 서버로 만든 후
엔진엑스 등과 리버스 프록시로 연결합니다.



gevent-server.py

```
from gevent.wsgi import WSGIServer
from flask7 import app

http_server = WSGIServer(('127.0.0.1', 5000), app)
http_server.serve_forever()
```

```
$ python gevent-server.py
```

misc

- 어플리케이션의 폴더 구조는 표준이 없습니다.
- 컨트롤러는 사이트의 메뉴 구조에 주로 영향을 받습니다.
  (e.g. 서브메뉴의 콘틀롤러는 한 폴더에서 관리)
- 모델은 콘트롤러와 인접해 있거나 모델끼리 따로 관리합니다.

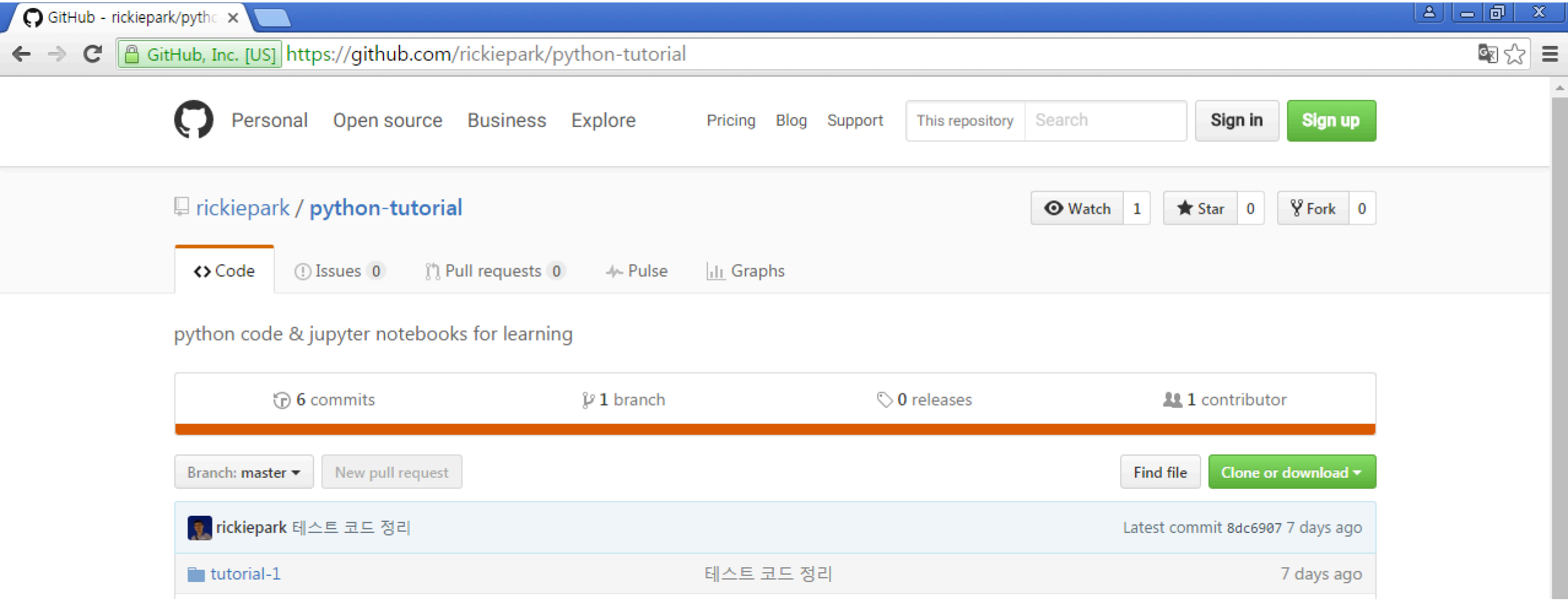
```
~/LargeApp
    |-- run.py
    |-- config.py
    |__ /env              # Virtual Environment
    |__ /app              # Our Application Module
        |-- __init__.py
        |-- /mod_auth   # Our first module, mod_auth
            |-- __init__.py
            |-- controllers.py
            |-- models.py
            |-- forms.py
        |__ /templates
            |-- 404.html
            |__ /auth
                |-- signin.html
        |__ /static
```

디폴트 값을 이용
(Convention over Configuration)

Gitflow http://danielkummer.github.io/git-flow-cheatsheet/index.ko_KR.html

Setup & Example

Github Download



내 문서 밑으로 압축 해제

학생 점수 관리

- 3강에서 했던 학생관리 클래스를 모델로 변환합니다.
- 학생과 점수 데이터를 위한 SQL 문을 생성하고 DB를 만듭니다.
- 학생 리스트, 추가 페이지를 만듭니다.
- 학생 리스트에서 학생 이름을 누르면 각 학생의 점수 데이터를 보여줍니다.
- 학생 데이터를 삭제하는 버튼/링크를 만듭니다.

  (SQL문: delete from *tablename* where id=*id*)

Q&A

Any Question: haesunrpark@gmail.com

github URL: https://github.com/rickiepark/python-tutorial

This slide available at
http://tensorflowkorea.wordpress.com