



Python Programming

Flask 활용

각 절에서 다루는 내용

1. 웹 서버 개요
2. Flask 웹 어플리케이션 설치
3. Flask 웹 어플리케이션 기본 골격
4. 라우팅 설정하기
5. Flask 에 HTML 연동
6. Flask 에 CSS 연동
7. Flask에 JSON 연동

웹서버 개요

■ 웹서버

- 클라이언트가 브라우저를 통해서 요청하는 데이터를 HTTP 프로토콜을 사용해 제공하는 인터넷 서비스 프로그램 (apache, lighttpd, nginx)

■ 웹 클라이언트

- 대표적인 웹 브라우저로 크롬, firefox, explore 등이 있음.

■ HTML (Hyper Text Markup Language)

- 클라이언트가 웹서버에게 데이터를 요청을 할 때 제공하는 정적인 문서를 기술하는 언어

■ CSS (cascading style sheets)

- HTML에서 디자인을 분리하고 여러 HTML로 기술한 문서에 일관된 스타일을 적용
- 글자의 크기, 글자체, 줄 간격, 배경 색상, 배열 위치 등

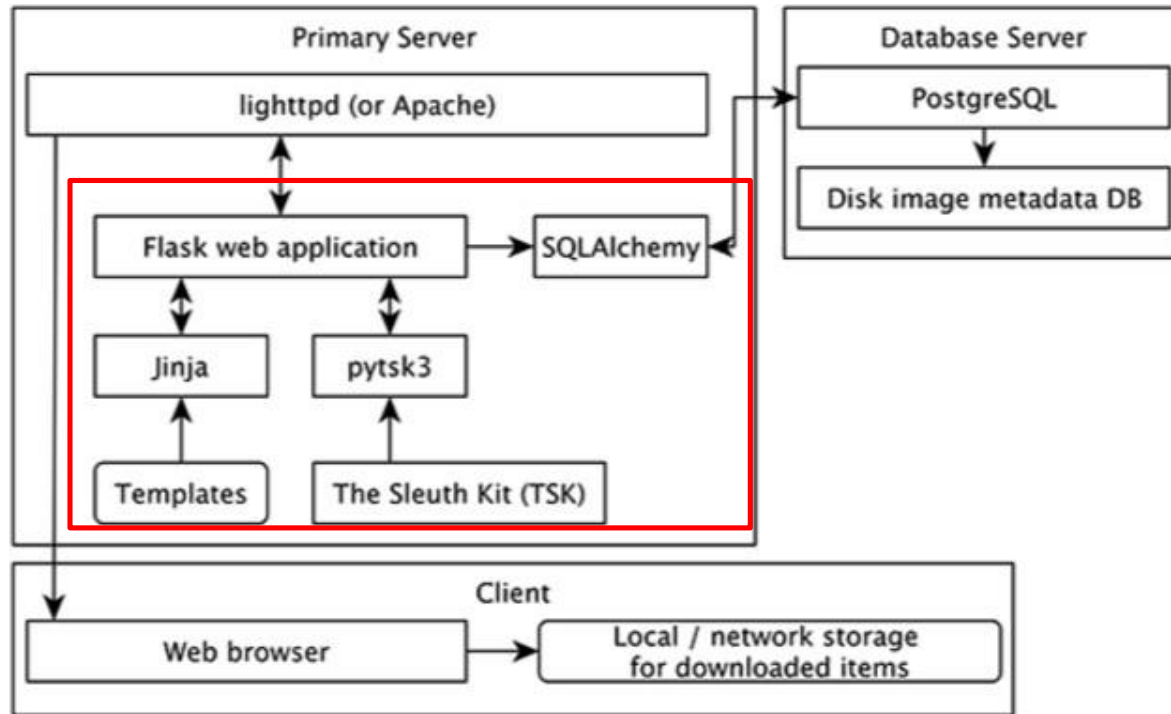
■ 웹 응용 프로그램

- 동적인 웹 콘텐츠를 생성하는 프로그램

■ 웹 응용 프레임워크

- 웹 응용 프로그램의 개발에 필요한 서비스와 자원, API를 제공
- Sping, node.js, Zend framework, bottle, flask, Django 등

FLASK 웹 응용 프레임워크



- 쉬운 웹 개발을 위해 서비스를 제공하는 마이크로 프레임워크 VS Django(full stack)
- Jinja를 통한 지원
 - WSGI:파이썬 웹 서버 인터페이스 (웹 프로그램과 웹브라우저간 통신)
 - URL 라우팅을 지원
 - `app.run()` 함수로 웹 서버를 쉽게 구축하고 실행할 수 있음
- SQLAlchemy를 통한 데이터베이스의 접근을 제공

Flask 설치 및 Flask 웹 어플리케이션 기본 골격

```
sudo apt-get install python3-flask
```

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def index():
    return 'Hello world'
if name == '__main__':
    app.run()
# app.run(debug=True, host='0.0.0.0') 원격 컴퓨터에서 접속 가능한 설정 시작
# app.debug=True
# app.run(host='0.0.0.0', port = 80)
```

Flask 웹 어플리케이션 기본 골격

- `from flask import Flask`
 - flask 클래스를 가져옴
- `app = Flask(__name__)`
 - flask 객체 생성
 - `__name__`: 현재 실행 중인 모듈의 이름을 적달, 임의이 문자열로 변경 가능
- Flask 객체
 - 웹 어플리케이션의 적반에 대해 영향을 끼치는 메인 객체
- `@app.route('/')`
 - URL /로 접속 시 실행할 함수를 지정, 현재는 `index()`로 지정됨
 - `index()`에서 `return` 한 `hello world` 문자열이 클라이언트(브라우저)로 반환됨
- `app.run(debug=True, host='0.0.0.0', port = 80)`
 - Flask 프레임 워크에 포함된 내장 웹 서버를 실행하는 코드
 - `host= ' 0.0.0.0'` # 네트워크의 어떤 디바이스도 접근할 수 있다는 의미
 - `port='80'` # 기본 port는 5000, 변경하고자 하는 경우 입력

라우팅 설정하기

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

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

@app.route('/cakes')
def cakes():
    return 'Yummy cakes!'

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0')
```

- `@app.route('/')`
 - 웹 브라우저에서 `http://127.0.0.1:5000/`로 접속 시에 실행될 `index` 함수 지정
- `@app.route('/cakes')`
 - 웹 브라우저에서 `http://127.0.0.1:5000/cakes`로 접속 시에 실행될 `index` 함수 지정

라우팅 설정하기(Get/Post 구분하기)

```
from flask import Flask

app = Flask(__name__)

@app.route('/method/', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        return "Post"
    else:
        return "Get"

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0')
```

- `@app.route('/method/', methods=['GET', 'POST'])`
 - 라우팅 주소에 `methods` 변수에 GET, POST 문자열 설정한다.
- `request.method == 'POST':`
 - `request.method` 값의 문자열을 반환 받아서 활용한다

라우팅 설정하기(Get 데이터 입력 받기)

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

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

@app.route('/user/<username>/<int:age>')
def show_user_profile_age(username, age):
    return 'Username %s, 나이 %d' % (username, age)

if __name__ == "__main__":
    # app.debug = True
    app.run(host='0.0.0.0', port = 80)
```

- `@app.route('/user/<username>/<int:age>')`
 - route 설정에 변수명 값을 < > 사이에 값을 적는다.
- `def show_user_profile_age(username, age):`
 - 메소드의 매개변수에서 그 변수값을 적어 값을 반환받아 사용한다.

라우팅 설정하기(Post 데이터 입력 받기)

```
from flask import Flask, request

app = Flask(__name__)

@app.route('/postData', methods=['POST'])
def postData():
    if request.method == 'POST':
        uname = request.form['username']
        pw = request.form['password']
        return 'Username : %s, pw : %s' % (uname, pw)

if __name__ == "__main__":
    # app.debug = True
    app.run(host='0.0.0.0', port = 80)
```

- `@app.route('/postData/', methods=['POST'])`
 - `methods` 를 POST 방식으로 설정
- `uname = request.form['username']`
 - `request.form['keyname']` 방식으로 그 값을 반환하여 사용한다.

Flask에 HTML 연동

■ Templates

- Flask는 template 엔진(jinja2)을 사용하여 파이썬 소스파일과 html 문서를 분리하여 관리

```
webapp
├── app.py
└── templates
    └── index.html
```

```
$ mkdir webapp
$ cd webapp
$ mkdir templates
```

- webapp 에 app.py 작성
- templates 디렉토리에 index.html를 작성

Flask에 html 연동

■ render_template

- 웹 브라우저로 템플릿을 보내주기 위한 함수
- 첫번째 인자 : 템플릿 파일(html)의 이름
- 두번째 인자 : 템플릿에서 사용되는 변수명과 변수값을 전달

예문 : `return render_template('index.html', name='kim')`

■ {{ name }}

- 파이썬 app에서 적달한 name 변수 값을 참조하여 출력

예문 : 이름이 있습니다. {{ name }}

- 웹 어플리케이션 실행 (flask에 내장된 웹 서버도 함께 실행됨)

```
$ sudo python3 app.py
```

Flask에 html 연동

```
<html>
<body>
<h1>Hello from a template!</h1>
{%if name %}
    <h1>Hello {{ name }}!</h1>
{%else %}
    <h1>Hello World!</h1>
{%endif %}
</body>
</html>
```

index.html

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

@app.route('/')
def index():
    return render_template('index.html', name='kim')

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80, debug=True)
```

app.py

Flask에 html 연동(리스트 출력)

```
<!DOCTYPE html>
<head>
  <meta charset="UTF-8">  <title>flask list</title>
</head>
<body>
<ul>
{% for item in plist %}
  <li>상품명 : {{ item }} </li>
{% endfor %}
</ul>
</body>
</html>
```

pList.html

← → ↻ 🏠 ⓘ 192.168.0.73/plist/

- 상품명 : pir
- 상품명 : sound
- 상품명 : motor
- 상품명 : door
- 상품명 : servor

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

@app.route("/plist/")
def plist() :
    list = {"motor", "servor", "door", "pir", "sound"}
    return render_template("pList.html", plist = list)

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80, debug=True)
```

app.py

Flask에 html 연동(dictionary 출력)

```
<!DOCTYPE html>
<head>  <meta charset="UTF-8">  <title>flask list</title></head>
<body>
품목 리스트 <br>
<ul>
{% for key, value in ulist.items() %}
  <li>key : {{ key }}, value : {{ value }} </li>
{% endfor %}
</ul>
</body>
</html>
```

list.html

← → ↺ 🏠 ⓘ 192.168.0.73/ulist/

품목 리스트

- key : led1, value : ON
- key : led2, value : OFF
- key : led3, value : ON
- key : temp, value : 32
- key : moter, value : 90

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

@app.route("/ulist/")
def ulist() :
    list = {"led1":"ON","led2":"OFF","led3":"ON","temp":32,"moter":90}
    return render_template("list.html", ulist = list)

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80, debug=True)
```

app.py

Flask 에 CSS 연 동

- CSS 를 위한 static 경로와 style.css 파일 생성

webapp

```
├── app.py
├── templates
│   └── index.html
├── static
│   └── style.css
```

- style.css 편 집

```
body {
    background: red; co
    lor: yellow;
}
```


Flask 에 CSS 연 동

- style.css 편 집

```
<html>
<head>
<link rel="stylesheet" href='/static/style.css' />
</head>
<body>
<h1>Hello from a template!</h1>
</body>
</html>
```

- 실행

```
$ sudo python3 app2.py
```

Flask에서 JSON 출력

```
import json

data = {'key': ['value1', 'value2']}
json_data = json.dumps(data, indent=4, ensure_ascii=False)
# indent 는 들여쓰기, ensure_ascii 는 한글 처리

print(json_data)

python_data = json.loads(json_data)
print(python_data)

print(type(json_data))
print(type(python_data))
```

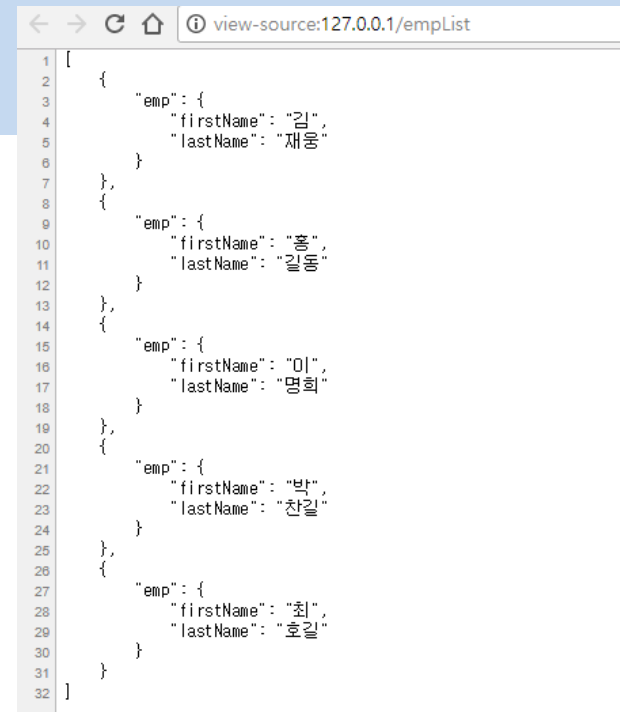
- **import json**
 - 내장 모듈인 json 객체를 import 한다.
- `json_data = json.dumps(data, indent=4)`
 - dumps 함수를 이용해서 json 문자열을 얻어낸다.
- `python_data = json.loads(json_data)`
 - loads 함수를 이용해서 dict 객체를 반환한다.

Flask에서 JSON 출력(class 리스트)

```
class Employee():
    def __init__(self, firstName, lastName):
        self.firstName = firstName
        self.lastName = lastName

    def toJSON(self):
        return {"emp": {'firstName': self.firstName,
                        'lastName': self.lastName}}
```

Employee.py



```
1 [
2   {
3     "emp": {
4       "firstName": "김",
5       "lastName": "재웅"
6     }
7   },
8   {
9     "emp": {
10      "firstName": "홍",
11      "lastName": "길동"
12    }
13  },
14  {
15    "emp": {
16      "firstName": "이",
17      "lastName": "명희"
18    }
19  },
20  {
21    "emp": {
22      "firstName": "박",
23      "lastName": "찬길"
24    }
25  },
26  {
27    "emp": {
28      "firstName": "최",
29      "lastName": "호길"
30    }
31  }
32 ]
```

Flask에서 JSON 출력(class 리스트)

```
from flask import Flask, json
from Employee import Employee
```

app.py

```
app = Flask(__name__)
```

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

```
def getEmployeeList():
```

```
    try:
```

```
        employeeList = []
```

```
        employeeList.append(Employee("김","재웅"))
```

```
        employeeList.append(Employee("홍","길동"))
```

```
        employeeList.append(Employee("이","명희"))
```

```
        employeeList.append(Employee("박","찬길"))
```

```
        employeeList.append(Employee("최","호길"))
```

```
        jsonStr = json.dumps([e.toJSON() for e in employeeList], ensure_ascii=False, indent=4)
```

```
    except :
```

```
        pass
```

```
    return jsonStr
```

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

```
    app.run(host = "0.0.0.0", port=80)
```