Web Studio 2019
3. Python, flask(2)

김상근 / sisobus1@gmail.com

Contents

외부 서버 접속하기

SSH (Secure SHell)

1. 외부 컴퓨터에 접속할 수 있도록 하는 어플리케이션 혹은 프로토콜
2. 강력한 인증 및 암호화를 이용한 안전한 통신 프로토콜
3. Windows에서는 PuTTY, XShell 등을 이용해 할 수 있음
4. Unix 계열 운영체제 (OS X, Linux)에서는 기본으로 설치되어 있음

외부 서버 접속하기

SSH (Secure SHell)

```
$ ssh <USERNAME>@54.180.96.144
<USERNAME>@54.180.96.144's password:
```

Virtualenv 사용하기

Setting, activation

```
$ python -m venv venv
$ . venv/bin/activate
(venv) $
```

Virtualenv 사용하기

Pip 를 이용한 패키지 설치

```
(venv) $ pip install flask
Collecting flask
  Using cached https://files.pythonhosted.org/packages/7f/
e7/08578774ed4536d3242b14dacb4696386634607af824ea997202cd0edb
/Flask-1.0.2-py2.py3-none-any.whl
Collecting Werkzeug>=0.14 (from flask)
…
Installing collected packages: Werkzeug, click, MarkupSafe,
Jinja2, itsdangerous, flask
Successfully installed Jinja2-2.10 MarkupSafe-1.1.1
Werkzeug-0.15.0 click-7.0 flask-1.0.2 itsdangerous-1.1.0
```

RESTful API

API란? 그리고 현대의 웹

1. API(Application Programming Interface)
2. 프로그램을 작성하기 위한 일련의 부 프로그램, 프로토콜 등을 정의하여 상호 작용을 하기 위한 인터페이스 사양(Specification)
3. 현대의 웹은 Javascript를 이용해 Backend API와 HTTP통신으로 상호작용하여 데이터(자원)를 가져옴

RESTful API

RESTful API란?

1. REST(Representational State Transfer)
2. 자원의 표현에 의한 상태를 주고받는 것
3. URI를 통해 자원을 표현하고, HTTP method로 자원에 대한 행위를 표현함
   - (e.g. URI /users/1 가 의미하는 것은 id=1인 user 하나의 자원을 의미함)
   - (e.g. /users/1 을 GET method로 호출하는 것은 id=1인 user하나를 가져오는 것을 의미)
4. HTTP 프로토콜을 사용함

RESTful API

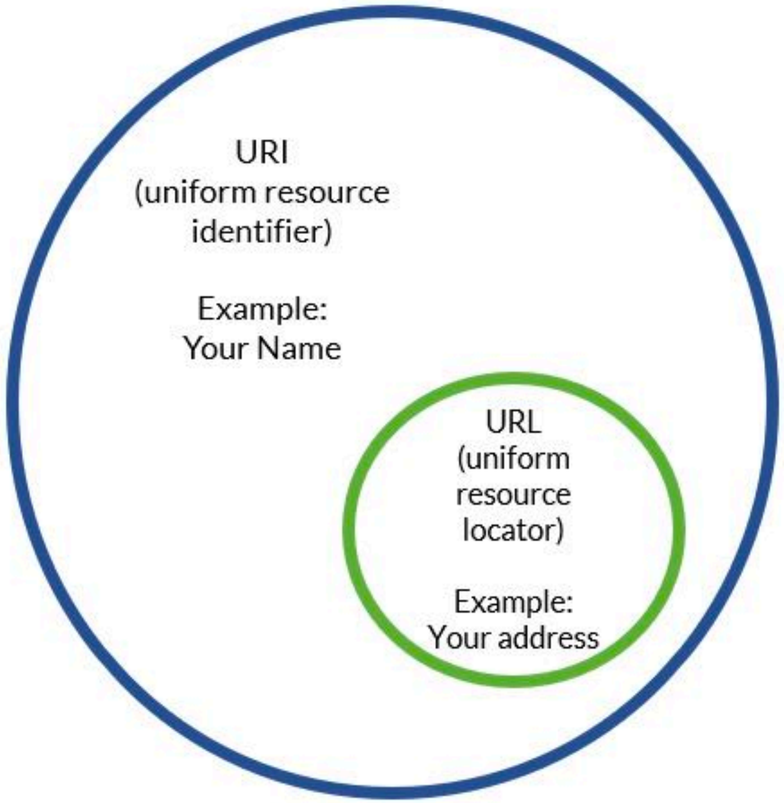
HTTP

1. HTTP(HyperText Transfer Protocol)
2. WWW상에서 정보를 주고 받을 수 있는 프로토콜
3. GET, POST, PUT, DELETE, ~~HEAD, OPTIONS, PATCH, …~~
4. https://developer.mozilla.org/ko/docs/Web/HTTP/Messages

RESTful API

URI? URL?

1. URI (Uniform Resource Identifier)
2. URL (Uniform Resource Locator)
3. 요즈음은 예전과 다르게 주소와 파일 위치가 일치하지 않음
   - Rewrite, Redirection, …



https://dev.to/flippedcoding/what-is-the-difference-between-a-uri-and-a-url-4455

RESTful API

JSON

1. JSON(JavaScript Object Notation)
2. python에는 json 라이브러리가 기본으로 내장되어 있음
3. 여러 언어마다 다른 오브젝트 포멧을 일종의 약속으로 일치시키는 방법
4.

```
{
    “email”: “sisobus@vuno.co",
    “password”: “dksdkffuwnwlfhd”
}
```

RESTful API

JSON 파일 쓰기

```
import json

d = {
    "email": "sisobus@vuno.co",
    "password": "dksdkffuwnwlfhd"
}
with open('auth.json', 'w') as fp:
    fp.write(json.dumps(d))
```

RESTful API

JSON 파일 읽기

```
import json

with open('auth.json', 'r') as fp:
    r = json.loads(fp.read())
print(r)
```

RESTful API

Flask_restful Library

1. flask에서 RESTful API를 쉽게(?) 편하게 만들 수 있도록 도와주는 라이브러리
2. 객체지향적임(Class based)
3. Get, post, put, delete 등의 HTTP method 이름의 메소드를 정의해주면 됨

```
(venv)$ pip search flask-restful
(venv)$ pip install flask-restful
```

RESTful API

User List

1. Class의 get, post, put, delete 메소드를 작성함
2. 테스트는 curl을 이용함

```
$ curl -X POST 54.180.96.144:5000/api/users
"post method"
$ curl -X PUT 54.180.96.144:5000/api/users
"put method"
```

```
from flask import Flask
from flask_restful import Api, Resource

app = Flask(__name__)
api = Api(app)

class UserList(Resource):
    def get(self):
        return 'get method'

    def post(self):
        return 'post method'

    def put(self):
        return 'put method'

    def delete(self):
        return 'delete method'

api.add_resource(UserList, '/api/users')


if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

RESTful API

Sign up (1)

1. request.get_json()으로 받아서
2. users.json에 덮어씀
3. 유저 하나만 저장됨

```
$ curl -X POST -H "Content-Type: application/json" \
  -d '{"email": "sisobus@vuno.co", "password": "1234qwer"}'\
  http://54.180.96.144:5000/api/users
```

```python
from flask import Flask, request
from flask_restful import Api, Resource
import json

app = Flask(__name__)
api = Api(app)

class UserList(Resource):
    def get(self):
        return 'get method'

    def post(self):
        r_json = request.get_json()
        email = r_json['email']
        password = r_json['password']
        with open('users.json', 'w') as fp:
            fp.write(json.dumps([r_json]))
        return 'email: {}, pw: {}'.format(email, password)

    def put(self):
        return 'put method'

    def delete(self):
        return 'delete method'

api.add_resource(UserList, '/api/users')


if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

RESTful API

Sign up (2)

1. 만약 users.json파일이 존재하면
2. 이전 파일을 읽어서 이미 있는 유저인지 체크
3. 없으면 추가해서 덮어씀

```
$ curl -X POST -H "Content-Type: application/json" \
  -d '{"email": "sisobus@vuno.co", "password": "1234qwer"}'\
  http://54.180.96.144:5000/api/users
```

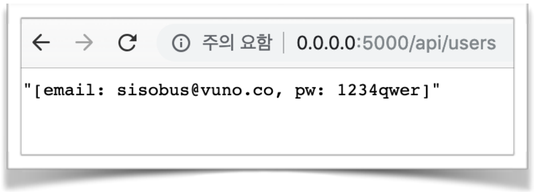
```
class UserList(Resource):
    def get(self):
        return 'get method'

    def post(self):
        r_json = request.get_json()
        email = r_json['email']
        password = r_json['password']
        r = []
        if os.path.exists('users.json'):
            with open('users.json', 'r') as fp:
                r = json.loads(fp.read())
        for d in r:
            if email == d['email']:
                return '{} is aleady exists'.format(email)
        r.append(r_json)
        with open('users.json', 'w') as fp:
            fp.write(json.dumps(r))
        return 'email: {}, pw: {}'.format(email, password)

    def put(self):
        return 'put method'

    def delete(self):
        return 'delete method'
```

RESTful API

User List

1. Get method로 요청하면
   users.json을 읽어서 리스트를 반환

```
$ curl http://54.180.96.144:5000/api/users
```




```python
class UserList(Resource):
    def get(self):
        if not os.path.exists('users.json'):
            return 'users.json is not exists'
        with open('users.json', 'r') as fp:
            r = json.loads(fp.read())
        s = ''
        for d in r:
            email = d['email']
            password = d['password']
            s += '[email: {}, pw: {}]'.format(email, password)
        return s

    def post(self):
        r_json = request.get_json()
        email = r_json['email']
        password = r_json['password']
        r = []
        if os.path.exists('users.json'):
            with open('users.json', 'r') as fp:
                r = json.loads(fp.read())
        for d in r:
            if email == d['email']:
                return '{} is aleady exists'.format(email)
        r.append(r_json)
        with open('users.json', 'w') as fp:
            fp.write(json.dumps(r))
        return 'email: {}, pw: {}'.format(email, password)
```

RESTful API

실습: 비밀번호 변경과 유저 삭제 구현하기(PUT, DELETE)

1. Git clone 〈fork뜬 remote repository〉
   : 제가 실수로 다 지웠어요 ;;
2. Sisobus repository랑 sync 맞추기
3. 3-practice-〈여러분영어이름〉 브랜치 만들기
   : git checkout -b 3-practice-〈여러분 영어이름〉
4. 실습 위치로 이동하기
   : cd 3_flask_2/practice
5. 본인 디렉토리 만들기
   : mkdir 〈여러분영어이름〉

RESTful API

실습: 비밀번호 변경과 유저 삭제 구현하기(PUT, DELETE)

6. 기존 api.py 코드 가져오거나 직접 작성하기
: cp ../source/api.py 〈여러분 영어이름〉
7. 본인 디렉토리로 이동하기 : cd 〈여러분 영어이름〉
8. Pwd 쳐보면 아마 ??? /WebStudio2019/3_flask_2/practice/〈여러분 영어이름〉 이 나옴
9. api.py의 port랑 put, delete 메소드를 수정
10. 아래와 같이 테스트

```
$ curl -X PUT -H "Content-Type: application/json" \
  -d '{"email": "sisobus@vuno.co", "password": "tnwjdgkfqlalfqjsgh"}'\
  http://54.180.96.144:5000/api/users

$ curl -X DELETE -H "Content-Type: application/json" \
  -d '{"email": "sisobus@vuno.co"}'\
  http://54.180.96.144:5000/api/users
```

RESTful API

추가구현

다음 페이지의 bcrypt를 이용하여 비밀번호를 암호화 해봅시다.

RESTful API

Bcrypt

1. 암호화하기 전에 원본에 추가할 문자열을 생성 (보안++)
2. bcrypt.hashpw(원본, salt)를 이용해 암호화한다.
3. 비밀번호 확인은 bcrypt.hashpw(원본, 암호화된 비밀번호)

```
import bcrypt

salt = bcrypt.gensalt()
password = '1234qwer'.encode('utf-8')
hashed_password = bcrypt.hashpw(password, salt)
print(hashed_password)

print(hashed_password == bcrypt.hashpw(password, hashed_password))
```

Q & A

Appendix

Bash 기본 명령어

1. ls : 현재 위치의 파일 리스트
   - ls -al : 숨겨진 파일까지 전부 보기
2. cd : 디렉토리 이동
   - .은 현재디렉토리, ..은 바로 전 디렉토리를 의미함
   - 예를들어 cd .. 이 명령어는 바로 전 디렉토리로 이동
3. rm : 파일 지우기
   - rm -rf : 디렉토리도 지울 수 있음
4. vi <filename> : vim 에디터를 이용한 파일 작성하기
5. mkdir <directoryname> : 디렉토리(폴더) 만들기
6. pwd : 현재 위치 출력하기

Appendix

Vim

1. 최고의 Text editor
2. 이것만 잘써도 코딩 생산성이 상당히 많이 올라감
3. 대부분의 IDE에는 Vim Plugin이 존재함
4. Vim 쓰세요
5. 두번쓰세요
6. 세번쓰세요
7. 평생쓰세요

Appendix

Vim

1. 명령어 모드와 에디터 모드로 나뉨
2. 맨처음 들어갈 때에는 명령어 모드임
3. 에디터 모드에서 명령어 모드로 바꾸는 것은 esc로 함
4. 명령어 모드에서 에디터 모드로 바꾸는 것은 i, o, a로 함 (i만 알아도 상관 없습니다.)

Appendix

Vim 명령어 모드

1. i는 현재 커서에서 에디터모드로 전환
2. yy는 현재 줄 복사
3. p는 복사한 것을 붙여넣기
4. dd는 현재줄 삭제
5. u는 undo
6. ctrl+r는 redo
7. :wq 는 저장후 종료를 의미함
8. /〈search string〉 으로 검색할 수 있음
9. :%s/〈stringA〉/〈stringB〉/g 는 stringA를 stringB로 대치함
   - :%s/^/₩/₩//g : 맨 앞에 //를 붙임
   - :3,12s/ha/hi/g : 3번줄부터 12번째줄의 ha를 hi로 바꿈