7주차(2)_Nginx, Flask 실습

python 가상환경 세팅

1. python 설치

python은 각자 설치 부탁드립니다.
설치 유무를 모르신다면 cmd에 `$ python --version` 혹은 `$ python3 --version` 등의 명령어로 확인 가능합니다.

2. virtualenv 설치

`$ pip install virtualenv`

3. virtualenv 실행

1) 가상환경 생성: `$ python -m venv 생성할가상환경명` ex) `$ python -m venv flask1`

2) 가상환경 실행: `$ 가상환경명\Scripts\acrivate.bat` ex) `$ flask1\Scripts\activate.bat`

2-1) Mac 가상환경 실행: `$ source 가상환경명\bin\activate`

3) 실행 완료: 아래 캡쳐처럼 커맨드라인 앞이 (가상환경명)으로 표시되어야 합니다.

가상환경 내에 flask 설치

`$ pip install flask`

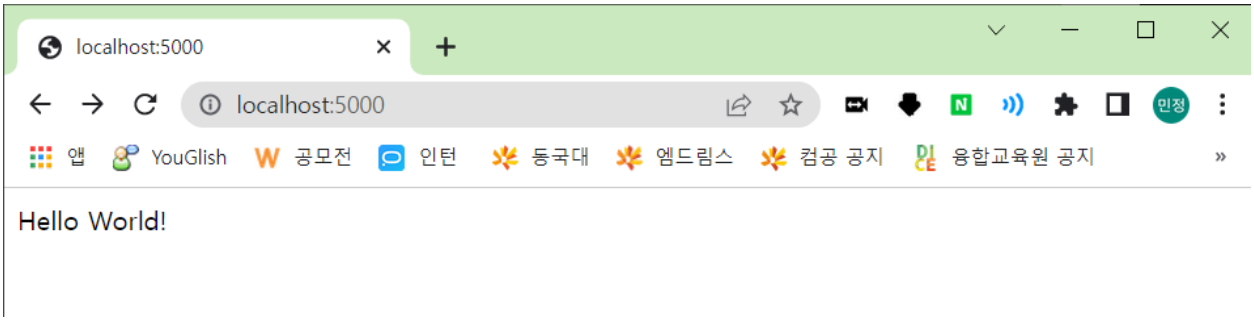
가상환경 종료

`(가상환경명)$ deactivate`

현재 시스템에서의 파이썬 환경은 활성화된 가상환경의 영향을 받기 때문에, 별도의 프로젝트를 진행할 때에는 현재 가상환경을 종료하고 새로운 가상환경을 만들어야 합니다. 프로그램 종료 후 꼭 해주세요!

예제#1 (ex1.py)

http://localhost:5000/ 에 접속했을 때, Hello World!가 뜨는 웹페이지



코드 설명

- Route() 데코레이터를 사용해서 어떤 URL이 작성한 함수를 실행시키는 지 알려주는 부분입니다. 간단히 URL을 설정한다고 생각하시면 됩니다!

  여기서는 '/' 뒤에 아무것도 없으니, 단순히 `ip주소:port번호`의 형태지만, 만약 라면, `localhost:5000/res` 에 접속해야 웹서버에 접속이 가능할거에요.

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

# 만약?
@app.route('/res')
```

- 함수를 생성하고 기능을 적는 부분이에요. 즉, 웹브라우저에 보여줄 메시지를 return하는 부분입니다. 여기서는 Hello World를 웹페이지에 출력할 예정입니다.

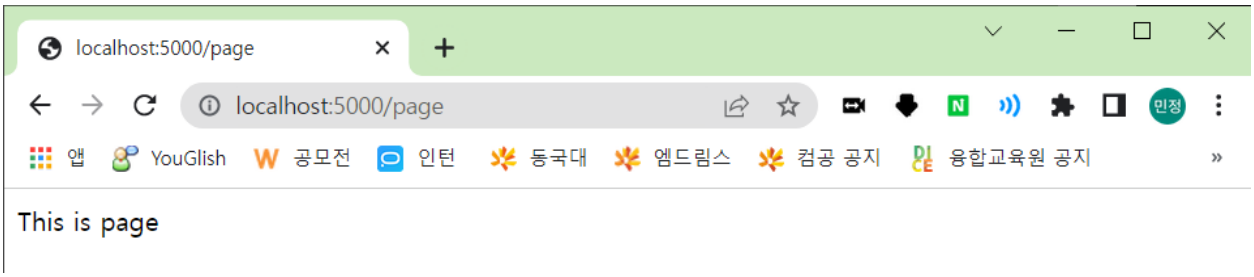
```
def hello_world():
    return ('Hello World!')
```

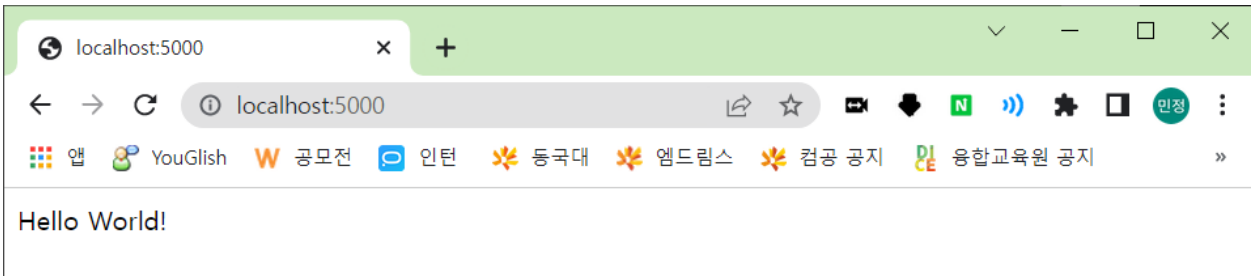
- Python에서 해당 실행 파일 명이 'main.py'일 때만 실행되는 함수입니다. 여기서는 해당 프로그램을 서버에 올리려고 사용합니다.
  보통 Flask의 default port는 5000번인데, 만약 IP주소나 port 번호를 바꾸고 싶다면 아래처럼 형태를 변형하여 접속 가능합니다.

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


# 만약?
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8000)
```

예제#2 (ex2.py)





http://localhost:5000/ 에 접속하면 Hello World가, http://localhost:5000/page 에 접속하면 This is page가 출력되는 웹페이지

코드 설명

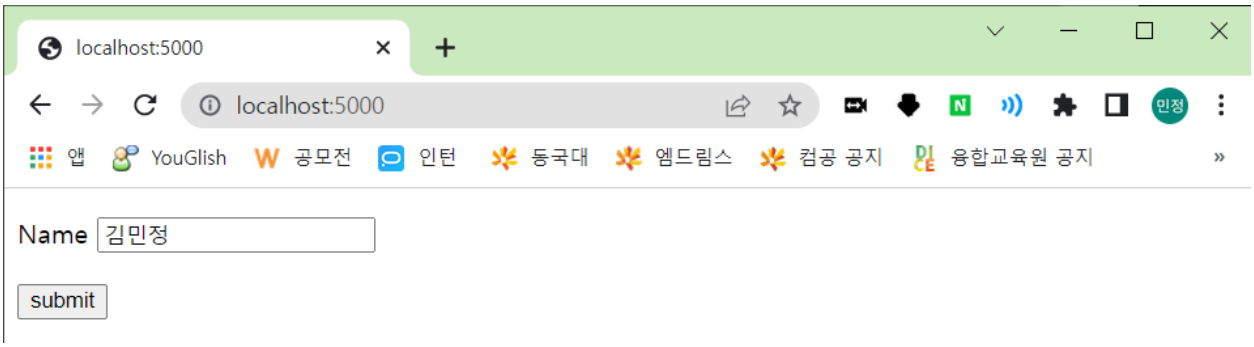
뒤에 page URL이 설정되지 않았다면, hello_world 함수가, page가 추가된다면 print_page 함수가 실행됩니다!

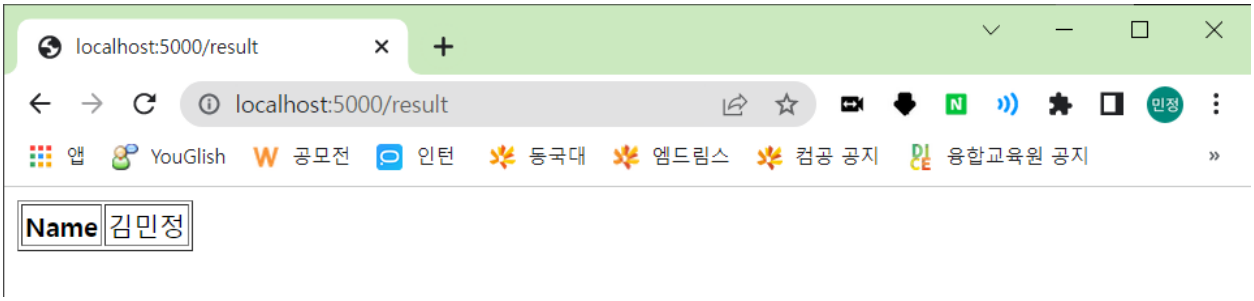
```
@app.route('/')  # default URL
def hello_world():
    return ('Hello World!')

@app.route('/page')  # page URL
def print_page():
    return ('This is page')
```

예제#3 (ex3.py)

main page에서 이름을 입력하고 submit 버튼을 누르면, result page로 넘어가서 Name이 출력됨.





코드 설명

폴더 구조는 아래처럼 html 코드가 templetes라는 폴더 안에 있어야 하고, 해당 templete 폴더는 python 코드와 같은 경로에 있어야 합니다!

```
templetes(디렉토리)
**|_input_info.html
**|_result.html
ex3.py(파일)
```

ex3.py

(1) default page에서는 `input_info.html` 을 실행하게 됩니다. 위에서는 단순하게 hello world를 return하는 정도에 그쳤지만, 여기서는 html 페이지 자체를 return하게 되는 거에요!

(2) result page로 들어오게 되면, 여기선 POST, GET이라는 methods를 사용하게 됩니다. 현재 페이지와 서버가 data를 교환할 때 쓰는 method에요! 즉, 웹사이트에 무언가를 입력하고(GET), 이를 반영하여 페이지를 새로 구성(POST)하고 싶다면 해당 method를 사용해야 합니다!
여기서는 Name을 입력하고, 그걸 result page에서 출력하려고 하니까 사용한거에요!
(3) result page에서는 입력받지 않고, 앞에서 입력받은 걸 출력하고만 싶으니까 POST method 일 때에는 Name을 출력하도록 했습니다.
안쪽 코드는 아래에서 더 자세히 봐볼게요.

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

@app.route('/')
def student():
   return render_template('input_info.html')  ## (1)

@app.route('/result', methods = ['POST', 'GET'])  ## (2)
def result():
   if request.method == 'POST':  ## (3)
      result = dict()
      result['Name'] = request.form.get('Name')
      return render_template("result.html",result = result)

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

(1) result라는 변수를 dictionary 형태로 선언해주었습니다. 이 자료형은 python에서 사용되는 type이고, 보통 result[1] = 1이런식으로 index에 int만 가능한 list와는 다르게 index로 string을 사용할 수 있어요. 실제론 index라고 부르지 않고 key라고 부릅니다.
참고: https://wikidocs.net/16043

(2) 그래서 result['Name']이라는 공간 안에, `request.form.get('Name')`의 값을 저장했어요. 여기서 나오는 Name은 `input_info.html`에서 정의한 Name입니다. 밑 문단에서 자세히 다룰게요!

(3) 이렇게 저장된 result dictionary를 result.html 페이지에 넘겨서 페이지를 렌더링해서 출력해주면, 그게 result page가 되는 겁니다!

```
result = dict()  ## (1)
result['Name'] = request.form.get('Name')  ## (2)
return render_template("result.html",result = result)  ## (3)
```

input_info.html

html는 기본 형태가 있는데, <>안에 쌓인 걸 태그라고 하고, 태그는 꼭 `<태그명>`로 시작해서 `</태그명>`로 끝내줘야 해요!

만약에 한 태그 안에 종료하고 싶으면 `<태그명 내용~ />`형태로 한 번에 close해줘도 됩니다!

그럼 안쪽 코드만 자세히 봐볼게요.

참고: https://velog.io/@inyong_pang/HTMLCSS-HTML-기본-문법

```
<!doctype html>
<html>  <!-- html TAG open -->
   <body>  <!-- body TAG open-->

      <form action = "/result" method = "POST">  <!-- form TAG open -->
         <p>Name <input type = "text" name = "Name" /></p>  <!-- p & input TAG open/close -->
         <p><input type = "submit" value = "submit" /></p>  <!-- p & input TAG open/close -->
      </form> <!-- form TAG close-->

   </body>  <!-- body TAG close -->
</html>  <!-- html TAG close -->
```

(1) result page에 들어갔을 때 수행할 action으로 POST method를 사용합니다. 즉, 여기에 입력된 값들을 ex3.py에 넘겨줘서 결론적으론 result dictionary에 저장되게 합니다.

(2) <p>는 문단을 구분할 때 사용하는 태그입니다. 여기서는 줄을 구분하기 위해 사용했어요.
Name(맨 왼쪽)이라는 모듈은 입력하는 태그(<input>)를 달아줄건데요, <input>을 사용하면 입력하는 모듈을 선택할 수 있는데, 그 중에 “text” type을 선택해준거에요. 참고 링크에 들어가보면 알 수 있는데, 모듈을 굉장히 여러가지가 있고, 여러 모듈로 입력한 값들이 Name(맨 뒤)이라는 이름으로 저장됩니다!

(3) 이 친구는 페이지를 넘기는 버튼입니다! button은 여러 type이 있는데 여기선 submit type을 사용했어요!

```
<form action = "/result" method = "POST">  ## (1)
    <p>Name <input type = "text" name = "Name" /></p>  ## (2)
    <p><input type = "submit" value = "submit" /></p>  ## (3)
</form>
```

참고: https://coding-factory.tistory.com/24

result.html

result page 출력을 위해 제가 드린 코드입니다. 과제 제출을 위해 수정할 필요는 없지만 간략한 정보는 주석으로 달아놓을게요. 과제에서는 그대로 사용하시면 됩니다.

```
<!doctype html>
<html>
   <body>

      <table border = 1>  <!-- 표를 생성함 -->
         {% for key, value in result.items() %}  <!-- 표를 구성하는 값들은 result라는 parameter에서 가져옴 -->

            <tr>
               <th> {{ key }} </th>    <!-- key부분(인덱싱)을 표 왼쪽에 출력-->
               <td> {{ value }} </td>    <!-- value부분(값)을 표 오른쪽에 출력-->
            </tr>

         {% endfor %}
      </table>   <!-- 표 생성 끝 -->

   </body>
</html>
```