

Server-side Dynamic Web Service

제출일	2022.11.16	전공	컴퓨터공학과
과목	컴퓨터네트워크(02)	학번	201602037
담당교수	이영석 교수님	이름	이규정

① 코드 설명

1) Import

```
import json
import urllib
from flask import Flask, Blueprint, render_template, request, redirect

endpoint = 'http://localhost:8080/pastebin/api'

app = Flask(__name__)
```

과제에서는 Blueprint를 사용하여 동작시켰지만, blueprint로 app 함수를 작성하면 404 error가 나와서 해결방법을 찾아보았으나, 도저히 찾을 수가 없어 대체로 app을 사용하여 코드를 작성했습니다.

2) Get_index() , index.html

```
@app.route('/', methods=['GET'])
def get_index():
    count_users = 0
    url = f'{endpoint}/users/'
    data = None
    headers = {'Accept': 'application/json'}
    method = 'GET'
    req = urllib.request.Request(url=url,
                                data=data,
                                headers=headers,
                                method=method)

    with urllib.request.urlopen(req) as f:
        data = json.loads(f.read())
        count_users = len(data)

    count_pastes = 0
    url = f'{endpoint}/pastes/'
    data = None
    headers = {'Accept': 'application/json'}
    method = 'GET'
    req = urllib.request.Request(url=url,
                                data=data,
                                headers=headers,
                                method=method)

    with urllib.request.urlopen(req) as f:
        data = json.loads(f.read())
        count_pastes = len(data)

    return render_template('index.html',
                           count_users=count_users,
                           count_pastes=count_pastes)
```

실습자료에 나와있는 그대로 작성해주었습니다. 처음에 Flask에 대한 지식이 아예 없는채로 시작하여 저 코드를 이해하는 부분에서 상당 시간을 소요했습니다.

Route에 접근할 path를 작성해주고 유저의 수와 메모의 숫자를 나타내주는 함수입니다. url 변수에는 저번주에 작성한 FastAPI에서 user의 상태를 볼 수 있는

<http://localhost:8080/pastebin/api/users/> 를 저장하고 ,Header에는 FastAPI에서 get_users를 실행한 후, 나와있는 accept를 그대로 작성합니다. Method = get을 저장해 get 신호를 app server에 보냅니다. 이제 위에서 작성한 변수들을 request를 이용하여 server에 보내고 위의 url에 담겨있는 data를 읽어 저장합니다. Get_index의 기능은 유저의 수와 메모의 수를 출력 출력하는 것이 data의 길이를 확인하고 저장합니다.

Paste 부분도 위의 코드전개와 동일한 방식으로 작성되었습니다.

7주차 과제인 FastAPI 구현부분이 제대로 이루어지지 않았던 것을 엉뚱한 곳에서 오류를 찾으려고 해서 시간이 굉장히 많이 소요되었습니다. 7주차 과제에서 paste 전체를 불러오는 함수를 작성하지 않아서 아래부분의 코드에서 오류가 났습니다. App 디렉토리의 fastapi 에 함수를 추가해주어서 해결 했습니다.

```

<!DOCTYPE html>
<html>

<title>My Paste Bin - Main</title>

<link rel="stylesheet" href="{{ url_for('static', filename='mystyle.css') }}">

<body>

<h1>Pastes ( {{ count_pastes }} pastes from {{ count_users }} users )</h1>

</body>

</html>
|

```

Index.html 인데 bp를 사용하지 않아서 url_for 부분에 mybp를 삭제해주었습니다.



3) Create_user(), createuser.html

```
@app.route('/createuser', methods=['POST', 'GET'])
def create_user():
    if request.method == 'POST':
        url = f'{endpoint}/users/'
        data = {'username': request.form['username'],
                'password': request.form['password']}
        headers = {'Accept': 'application/json',
                   'Content-type': 'application/json'}
        method = 'POST'
        data = json.dumps(data).encode('utf-8')

        req = urllib.request.Request(url=url,
                                     data=data,
                                     headers=headers,
                                     method=method)

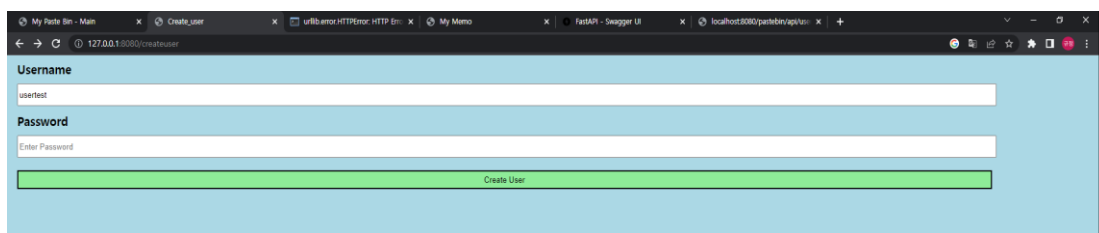
        urllib.request.urlopen(req)

        return redirect('/')

    if request.method == 'GET':
        return render_template('createuser.html')
```

127.0.0.1:8080/createuser 로 접근할 수 있고, method를 2개(POST, GET)형식으로 지정해주었습니다. 함수 안에서 두 가지 상태가 모두 사용됩니다.

GET 신호를 받으면 createuser.html를 보여줍니다.



POST를 받으면 위의 html에서 입력받은 Username과 password를 url에 전달합니다. Input으로 받은 데이터는 request.form[]을 통해서 data에 저장되고 header 정보와 method 정보를 모두 담아 Request를 보내어 데이터베이스에 유저정보를 POST 요청을 합니다. 정상적으로 POST가 되었다면 redirect('/')을 통해 처음 index 화면으로 돌아가게 됩니다.

Createuser.html은 마찬가지로 static 부분만 수정하였습니다.

4) Create_paste(), createpaste.html

```
@app.route('/createpaste', methods=['POST', 'GET'])
def create_paste():
    if request.method == 'POST':
        user_data = {'username': request.form['username'],
                     'password': request.form['password']}

        paste_data = {'title': request.form['title'],
                      'content': request.form['content']}

        url = f'{endpoint}/users/{user_data["username"]}/pastes/?password={user_data["password"]}'

        headers = {'Accept': 'application/json',
                   'Content-type': 'application/json'}
        method = 'POST'
        data = json.dumps(paste_data).encode('utf-8')

        req = urllib.request.Request(url=url,
                                     data=data,
                                     headers=headers,
                                     method=method)

        urllib.request.urlopen(req)

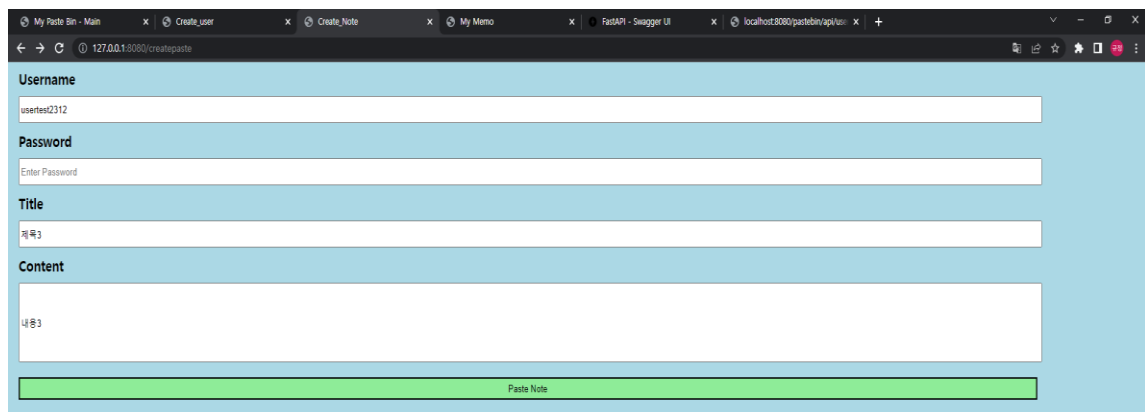
        return redirect('/')

    if request.method == 'GET':
        return render_template('createpaste.html')
```

Paste를 생성하는 함수입니다. Create_user와 같이 POST, GET 둘 다 사용합니다. 약간 다른점은 POST 부분에서 data처리 부분인데, user_data와 paste_data를 분리해서 사용하였습니다. 사용자 검증 부분을 구현하려고 했으나 그 부분은 해결하지 못했고, 대신 DB에 없는 username과 password로 접근하려고 하면 자동으로 error 500이 발생합니다.

url에 접근하려고 user_data 딕셔너리를 사용하여 key값으로 username과 password를 받아와서 url을 받아왔습니다. 그 뒤는 위의 함수들과 동일하게 header와 method 정보를 모아 data와 함께 POST 요청을 보냅니다.

Html은 createuser.html에서 label만 추가해서 작성해주었습니다.



5) Get_paste(),

```
from markupsafe import escape

@app.route('/getpaste/<username>', methods=['GET'])
def get_paste(username):
    count_pastes = 0
    paste_list = {}

    username = escape(username)
    url = f'{endpoint}/users/{username}/pastes/'
    data = None
    headers = {'Accept': 'application/json'}
    method = 'GET'
    req = urllib.request.Request(url=url,
                                data=data,
                                headers=headers,
                                method=method)

    with urllib.request.urlopen(req) as f:
        data = json.loads(f.read())
        count_pastes = len(data)
        for i in range(count_pastes):
            title = data[i]['title']
            content = data[i]['content']
            paste_list[title] = content

    return render_template('getpaste.html',
                           count_pastes=count_pastes,
                           username=username,
                           paste_list=paste_list)
```

마지막 함수인 get_paste(), paste의 정보를 보여주는 함수입니다. DB의 전체 paste를 보여주는 것이 아니라 특정 유저를 선택하여 그 유저의 paste 정보를 나타내주는 것이기 때문에 route에 <username>을 사용해서 접근합니다.

Flask 튜토리얼에 나와있는 escape를 사용하여 username에 접근 할 수 있었습니다. 주소창에 username이 포함된 주소를 입력하여 그 username을 인자로 받아서 url을 다시 만듭니다. 그 이후에는 처음에 작성한 get_index()와 흐름이 똑같습니다. 아래의 request를 열고 데이터를 저장하는 부분은 반복문과 딕셔너리를 사용했는데, 과제에서 요구하는 title, content가 쌍으로 나오게끔 하기 위해서 위와같이 작성해주었습니다. Html에서 paste_list로 접근할 수 있습니다.

```
my < mystyle.css < getpaste.html < createuser.html < main.py < createpaste.html <
<!DOCTYPE html>
<html>

<title> My Memo </title>

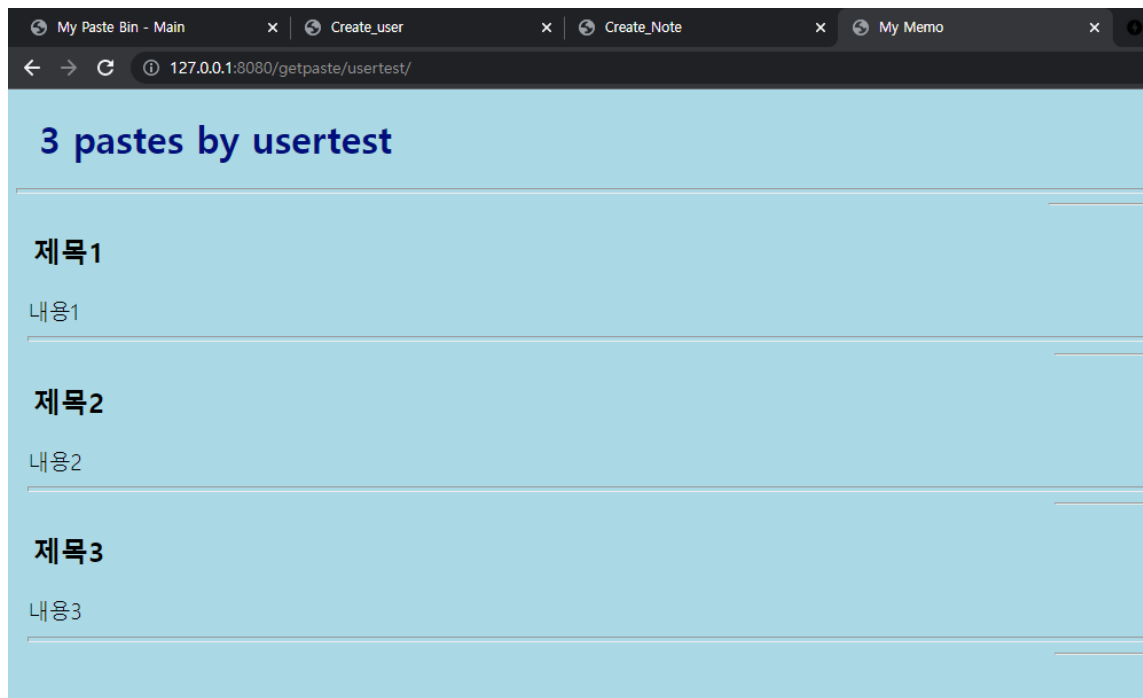
<link rel="stylesheet" href="{ url_for('static', filename='mystyle.css') }">

<body>

<h1> {{ count_pastes }} pastes by {{ username }} </h1>
<hr size="5" <hr width="100px">

{% for key,value in paste_list.items() %}
<h2>
<p>{{ key }} <br>
</h2>
<h3>
    {{ value }}
    <hr size="5" <hr width="100px">
</h3>
<img>
{% endfor %}
</body>
</html>
```

인자로 받은 count_pastes와 username을 나타내주고 반복문을 사용하여 paste_list 사전에서 key와 value를 같이 출력 할 수 있도록 코딩했습니다.




```

body{
    background-color: lightblue;
}
h1{
    color: navy;
    margin-left: 20px;
}
h2{
    color: black;
    margin-left: 15px;
}
h3{
    color: black;
    margin-left: 10px;
    font-weight: lighter
}
input{
    text-align: left;
    width: 90%;
    height: 30px;
    margin: 10px;
}
#content{
    text-align: left;
    width: 90%;
    height: 100px;
    margin: 10px;
}
label{
    font-size: 20px;
    margin: 10px;
}
button{
    width: 90%;
    height: 30px;
    margin: 10px;
    background-color : lightgreen;
}

```

css 함수입니다. 과제 양식에 맞게 간단하게 작성하였습니다.

② 결과 화면

과제 결과 진행화면은 영상으로 첨부합니다.