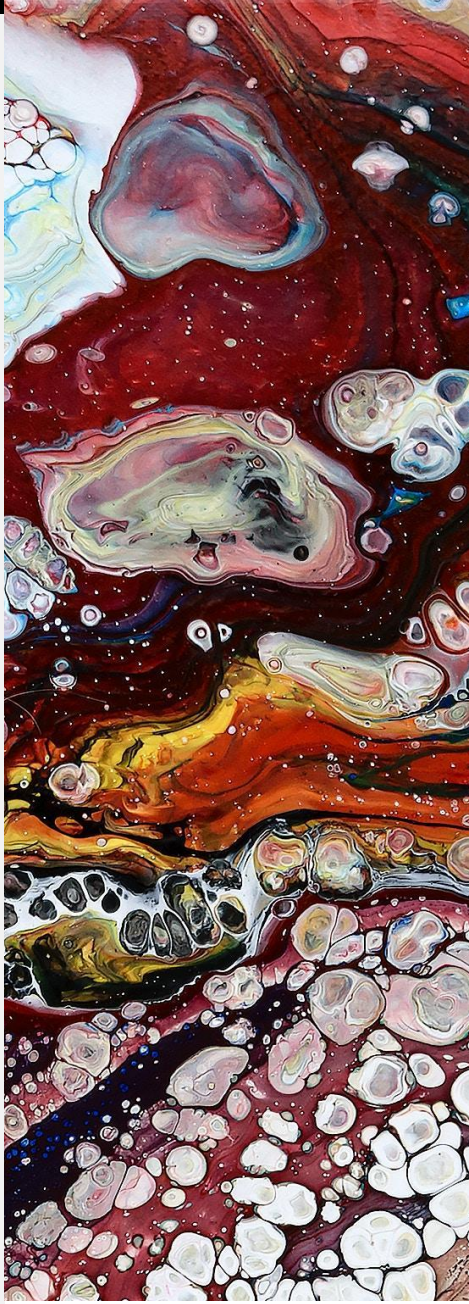
FLASK

웹 프로그래밍

목차



FLASK 웹 어플리케이션 기본 골격

```
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 전달, 임의의 문자열로 변경 가능

Flask 객체
웹 어플리케이션의 전반에 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

FLASK에 HTML 연동

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

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

render_template()함수 이용

FLASK에 CSS, JS, IMAGE 연동

- Static
  - CSS, JS, image 등을 위한 static 경로

```
webapp
    ├── app.py
    └── templates
        └── index.html
    └── static
        └── style.css
        └── script.js
        └── image.jpg
```

라우팅 설정

```
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 설정 (VARIABLE RULES)

```
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)
```

```
<variable_name>
<converter:variable_name>
```

string	accepts any text without a slash (the default)
int	accepts integers
float	like `int` but for floating point values
path	like the default but also accepts slashes
any	matches one of the items provided
uuid	accepts UUID strings

http://127.0.0.1:5000/user/hong

http://127.0.0.1:5000/user/hong/25

REQUEST OBJECT

클라이언트 웹 페이지의 데이터는 global request object를 통해서 서버로 전달됩니다.

request 데이터를 처리하기 위해서는 Flask 모듈로부터 import되어야 합니다.

request 객체의 중요한 속성은 다음과 같습니다.

- **Form** - form 매개변수를 key로, 그 값을 value로 하는 쌍을 보관하는 dictionary입니다.
- **args** - URL 부분 중 물음표(?) 다음에 있는 query 문자열의 내용을 parse합니다.
- **Cookies** - Cookies 이름과 그 값을 가지는 dictionary 객체입니다.
- **files** - 업로드 파일과 관련된 데이터
- **method** - 현재 request method

REQUEST.METHOD

```
from flask import Flask, request

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

REQUEST.ARGS

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

@app.route('/login')
def login():
  user = request.args.get('name')
   return 'User %s' % user

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

http://127.0.0.1:5000/login?name=hong

REQUEST.FORM

form_input.html

```
<!DOCTYPE html>
<html>
   <head>
      <title>flask test</title>
   </head>
   <body>
      <form action="/login" method="POST">
         <input type="text" name=" username " >
         <input type="password" name="password">
         <input type="submit" value="확인">
      </form>

   </body>
</html>
```

REQUEST.FORM

```
from flask import Flask, request

app = Flask(__name__)

@app.route('/')
def index():
    return render_template(form_input.html')

@app.route('/login', methods=['POST'])
def login():
   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)
```

REQUEST.FORM

```
from flask import Flask, request,  render_template

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('form_input.html')

@app.route('/login', methods=['POST'])
def login():
   if request.method == 'POST':
      result = request.form
      return render_template('form_result.html',result=result)

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

REQUEST.FORM

form_result.html

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

     <table border = 1>
        {% for key, value in result.items() %}

           <tr>
              <th> {{ key }} </th>
              <td> {{ value }} </td>
           </tr>

        {% endfor %}
     </table>

  </body>
</html>
```

REQUEST.FILES

upload.html

```
<html>
   <body>
      <form action = "/fileUpload" method = "POST"
           enctype = "multipart/form-data">
         <input type = "file" name = "file" />
         <input type = "submit"/>
      </form>
   </body>
</html>
```

REQUEST.FILES

upload.html

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

app = Flask(__name__)

@app.route('/upload')
def render_file():
   return render_template('upload.html')

@app.route('/fileUpload', methods = ['GET', 'POST'])
def upload_file():
   if request.method == 'POST':
      f = request.files['file']
      dirname=os.path.dirname(__file__) + '/uploads/'+f.filename   #저장할 경로 + 파일명
      print(dirname)
      f.save(dirname)
   return 'uploads 디렉토리 -> 파일 업로드 성공!'

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

URL_FOR() 특정함수를 호출하는 URI찾기

```
from flask import Flask,url_for

app = Flask(__name__)

@app.route('/')
def index():
    pass

@app.route('/login/')
def login():
    pass

@app.route('/user/<username>')
def profile(username):
    pass
```

```
if __name__ == '__main__':
    with app.test_request_context():
        print(url_for('index'))
        print(url_for('login'))
        print(url_for('login',next='/'))
        print(url_for('profile',username='hong'))
```

app.test_request_context() 함수는 플라스크에게 현재 파이썬 쉘에서 테스트를 하고 있음에도 지금 실제로 요청을 처리하고 있는것 처럼 상황을 제공한다.

print(url_for('index'))
index함수에 대한 URL을 화면에 출력

print(url_for('login',next='/'))
next라는 변수에 '/'값을 치환하는데 next변수는 존재하지 않는다. 따라서 알 수 없는 인자는 쿼리 인자로 URL 뒤쪽에 붙는다.

REDIRECT

```
from flask import Flask, redirect, url_for
app = Flask(__name__)

@app.route('/admin')
def hello_admin():
   return 'Hello Admin'

@app.route('/guest/<guest>')
def hello_guest(guest):
   return 'Hello %s as Guest' % guest

@app.route('/user/<name>')
def hello_user(name):
   if name =='admin':
      return redirect(url_for('hello_admin'))
   else:
      return redirect(url_for('hello_guest', guest = name))

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

url_for() 함수는 함수 이름으로 된 종단점(endpoint)의 URL을 생성합니다. 그리고 변수 규칙에 따라서 인자도 전달할 수 있습니다. 그래서 함수명을 첫번째 인자로, 하나 혹은 그 이상의 키워드 인자를 전달하면, 변수 규칙에 맞추어 이에 해당하는 URL을 생성해줍니다.

http://127.0.0.1:5000/user/admin

http://127.0.0.1:5000/user/test

JINJA2

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>My Webpage</title>
</head>
<body>
    <ul id="navigation">
    {% for item in navigation %}
        <li><a href="{{ item.href }}">{{ item.caption }}</a></li>
    {% endfor %}
    </ul>
    <h1>My Webpage</h1>
    {{ a_variable }}
    {# a comment #}
</body>
</html>
```

- `{% ... %}` for Statements
- `{{ ... }}` for Expressions to print to the template output
- `{# ... #}` for Comments not included in the template output
- `#  ... ##` for Line Statements

JINJA2

Variables

```
{{ item.href }} {{ item['href']}}
```

. item 객체 내에서 href라는 속성을 먼저 확인한 후, 없을 경우 href라는 객체가 있는지 확인.

[ ] item 객체 내에서 href라는 객체를 먼저 확인한 후, 없을 경우 href라는 속성이 있는지 확인.

{%- ... %}, {%+ ... %}, {% ... -%} ... : 공백 제거 혹은 유지

Jinja 템플릿 엔진은 템플릿 줄끝의 개행문자를 제외한 나머지 공백은 그래도 출력함.

템플릿 태그 앞뒤에 +를 붙이면 공백유지, -를 붙이면 공백제거

{% raw %} ... {% endraw %} : 이스케이프

```
{{ '{{' }}
```

{% raw %} 와 {% endraw %}사이에 이스케이프할 문자를 넣음.

{{ '{{' }} 와 같이 따옴표로 간단하게 이스케이프 할 수 있음.

JINJA2

For ~ in

{% for <개별요소> in <리스트> %}

<실행코드>

{% endfor %}

{% for item in navigation %}

  {{ item.caption }}

{% endfor %}

JINJA2

For ~ in

변수	내용
loop.index	for 구문이 반복된 횟수(1부터)
loop.index0	for 구문이 반복된 횟수(0부터)
loop.first	반복이 처음일 경우 True
loop.last	마지막 반복일 경우 True
loop.length	전체 반복 횟수
loop.cycle	loop.cycle()안에 넣는 인자를 순서대로 전달 loop.cycle('1','2','3')처럼 사용하면 반복문이 돌때마다 순서대로 '1', '2', '3' 를 반복

JINJA2

if문

```
{% if <조건> %}
    <실행코드>
{% elif <조건> %}
    <실행코드>
{% else %}
    <실행코드>
{% endif %}
```

조건 만족하면 실행코드 그렇치 않으면 거짓일때 실행코드 수행

```
{% <실행코드> if <조건> else <거짓일때 실행코드> %}
```

users값이 None이 아니면 for구문 수행

```
{% for user in users if users %}
    <li>{{ user.username }}</li>
{% endfor %}
```

JINJA2

default.html

```
<!DOCTYPE html>
<html>
<head>
   <title>flask test</title>
</head>
<body>
   {% block body%}
   {% endblock %}
</body>
<footer>
   <span>footer section</span>
</footer>
</html>
```

JINJA2

```
{% extends 'default.html' %}
{% block body %}
<h1>{{name}}</h1>
<ul>
   {% for team in team_list %}
   <li>{{team}}</li>
   {% endfor %}
</ul>
{% endblock %}
```

JINJA2

```
<!DOCTYPE html>
<html lang="en">
<head>
    {% block head %}
    <link rel="stylesheet" href="style.css" />
    <title>{% block title %}{% endblock %} - My Webpage</title>
    {% endblock %}
</head>
<body>
    <div id="content">{% block content %}{% endblock %}</div>
    <div id="footer">
        {% block footer %}
        &copy; Copyright 2008 by <a href="http://domain.invalid/">you</a>.
        {% endblock %}
    </div>
</body>
</html>
```

JINJA2

```
{% extends "base.html" %}
{% block title %}Index{% endblock %}
{% block head %}
    {{ super() }}
    <style type="text/css">
        .important { color: #336699; }
    </style>
{% endblock %}
{% block content %}
    <h1>Index</h1>
    <p class="important">
        Welcome to my awesome homepage.
    </p>
{% endblock %}
```

app.py 분리

app.py

```python
from flask import Flask,render_template,redirect,url_for
from accountApi import account_api

app = Flask(__name__)

app.register_blueprint(account_api, url_prefix='/accounts')

@app.route("/")
def hello():
    return render_template('index.html')

@app.route('/list')
def list():
    return redirect(url_for('account_api.accountList'))

@app.route('/list1')
def list1():
    return redirect('/accounts/account')

if __name__ == "__main__":
    app.run(debug=True)
```

app.py 분리

accountApi.py

```python
from flask import Blueprint,render_template

account_api = Blueprint('account_api', __name__)

@account_api.route("/account")
def accountList():
    return render_template('list.html')
```

app.py 분리

index.html

```
<!DOCTYPE html>
<html>
    <body>
        <a href="{{url_for('list')}}">목록</a>
    </body>
</html>
```

app.py 분리

list.html

```
<!DOCTYPE html>
<html>
    <body>
        <a href="{{url_for('account_api.accountList')}}">계정</a>
        <br>
        <a href="{{url_for('hello')}}">홈</a>
        <br>
        <a href="/accounts/account">계정</a>
        <br>
        <a href="/">홈</a>
    </body>
</html>
```