

9장. 플라스크 & 웹 사이트 구축



flask(플라스크)

플라스크(flask)란?

파이썬으로 제작된 마이크로 웹 프레임워크의 하나이며, 웹 sever를 만들고, 웹 애플리케이션을 제작할 수 있다.



Flask

web development,
one drop at a time

Flask 문서에 오신 것을 환영합니다. 시작하기 [설치](#) 한 다음에 대한 개요 얻을 [빠른 시작을](#). Flask를 사용하여 작지만 완전한 애플리케이션을 만드는 방법을 보여주는 더 자세한 [자습서](#) 도 있습니다. 일반적인 패턴은 [Flask 용 패턴](#) 섹션에 설명되어 있습니다. 나머지 문서에서는 [API](#) 섹션의 전체 참조와 함께 Flask의 각 구성 요소를 자세히 설명합니다.

flask(플라스크)

- 플라스크 설치

Install Flask

Within the activated environment, use the following command to install Flask:

```
$ pip install Flask
```

```
c:\#python>pip install flask
Requirement already satisfied: flask in c:\#users\user\...
Requirement already satisfied: click>=5.1 in c:\#users\user\...
Requirement already satisfied: Werkzeug>=0.15 in c:\#users\user\...
Requirement already satisfied: Jinja2>=2.10.1 in c:\#users\user\...
```

```
C:\#Users\김기용>flask --version
Python 3.8.5
Flask 1.1.2
Werkzeug 1.0.1
```

flask 웹서버 만들기

■ 웹 서버 만들기

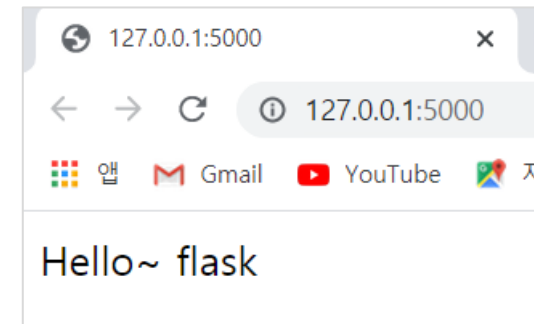
```
from flask import Flask

app = Flask(__name__) # app - Flask 객체 생성

@app.route('/') # 라우트 ('/'는 루트 경로)
def index():
    return "Hello~ flask"

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

```
* Serving Flask app 'app' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```



배포용이 아닌 개발용이다.
WSGI(WebServer GateWay Interface)

flask 웹서버 만들기

■ 웹 페이지 만들기

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

@app.route("/")
def index():
    return '<h1>index 페이지</h1>'

@app.route("/board")
def board():
    return "<h1>게시판 페이지</h1>"
```

← → ↻ ⓘ 127.0.0.1:5000

index 페이지

← → ↻ ⓘ 127.0.0.1:5000/board

게시판 페이지

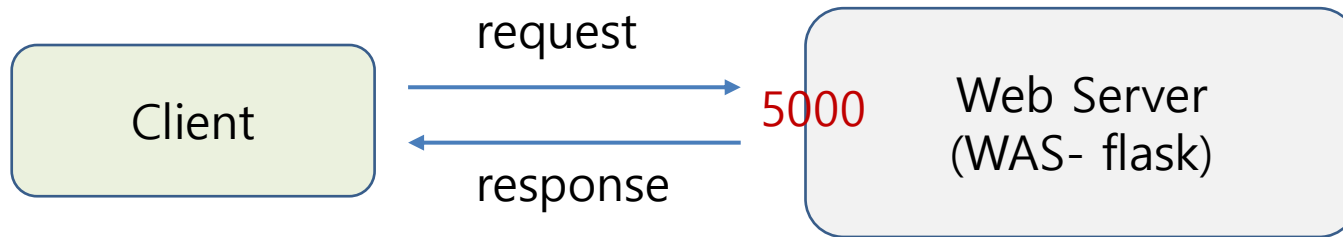
```
@app.route("/register")
def register():
    return "<h1>회원 가입 페이지</h1>"

@app.route("/login")
def login():
    return "<h1>로그인 페이지</h1>"

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

요청 및 처리

- 요청(request) 및 응답(response)



WAS(Web Application Server) : 웹 애플리케이션 서버로 함수를 통한 제어, DB 연동 업무를 수행하고 클라이언트에 응답하는 역할을 한다. 파이썬의 플라스크(flask)와 장고(django), 자바의 스프링(spring) 등의 웹 프레임워크를 가리킨다.

Get과 Post 방식

■ Get

GET은 클라이언트에서 서버로 **정보를 요청**하기 위해 사용되는 메서드이다.

보통 하이퍼링크를 클릭하면 웹 페이지로 이동하는 것을 생각하면 된다.

GET을 통한 요청은 URL 주소 끝에 파라미터로 포함되어 전송된다.

방식은 URL 끝에 " ? " 를 붙이고 그다음 변수명1=값1&변수명2=값2... 형식으로 이어 붙이면 된다.

<https://section.blog.naver.com/BlogHome.naver?directoryNo=0¤tPage=1>

■ Post

POST는 클라이언트에서 서버로 **리소스를 생성하거나 업데이트**하기 위해 데이터를 보낼 때 사용 되는 메서드이다.

예를 들면 회원 가입이나 게시글을 작성할때 사용되는 방식이다.

보안이 필요하거나 용량이 큰 데이터를 전송할 때 사용함.

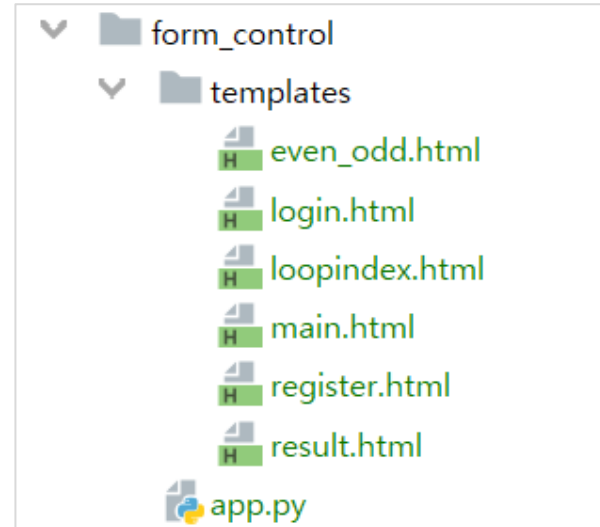
index 페이지 만들기

웹 계층 필수 구조

templates 폴더 -> html 파일

static 폴더 -> css, js, image 파일

start_app.py -> 실행 파일



템플릿으로 질문 목록 페이지 만들기

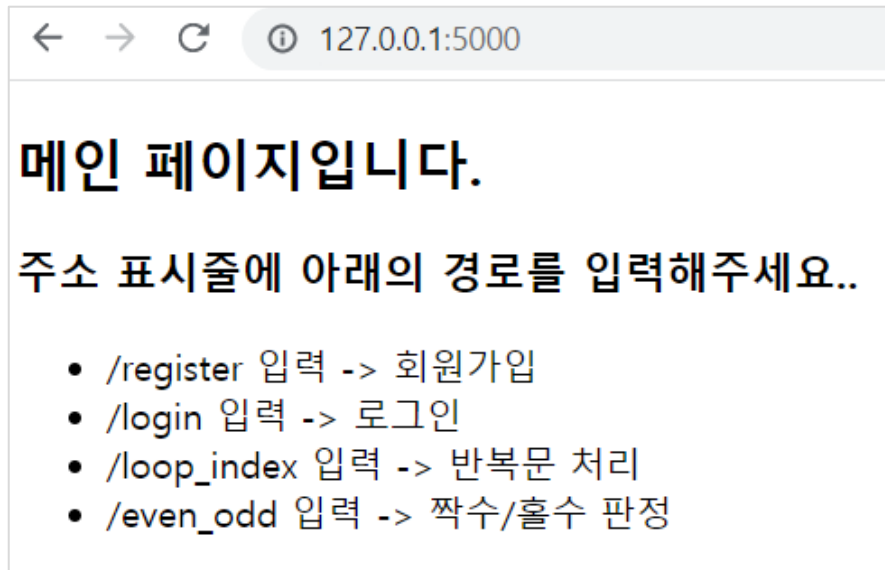
◆ 템플릿(template) 태그

파이썬을 웹에 적용한 언어로 **{% %}** 블록을 사용한다.

템플릿 태그	설 명
<code>{% if item_list %}</code> .. 내용 .. <code>{% endif %}</code>	item_list가 있다면(조건문)
<code>{% for item in item_list %}</code> .. 내용 .. <code>{% endfor %}</code>	item_list를 반복하며 순차적으로 item에 대입(반복문)
<code>{{ id }}</code>	id 출력(출력문)
<code>{{ loop.index }}</code>	loop 객체의 index 출력

메인 페이지

■ 메인페이지 - main.html



메인 페이지

■ 메인페이지 - main.html

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>메인 페이지 입니다.</title>
</head>
<body>
  <h2>메인 페이지입니다.</h2>
  <h3>주소 표시줄에 아래의 경로를 입력해주세요..</h3>
  <ul>
    <li>/register 입력 -> 회원가입</li>
    <li>/login 입력 -> 로그인</li>
    <li>/loop_index 입력 -> 반복문 처리</li>
    <li>/even_odd 입력 -> 짝수/홀수 판정</li>
  </ul>
</body>
</html>
```

회원 가입

- app.py – main() 함수

```
from flask import Flask, render_template, request

app = Flask(__name__)

@app.route("/")
def main():
    return render_template('main.html')
    #return 'Hello~ Flask'
```

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

회원 가입

■ 회원 가입 화면

← → ↻ ⓘ 127.0.0.1:5000/register

회원 가입

아이디

비밀번호

이름

나이



회원 목록

아이디 : 1001

패스워드 : a1234567

패스워드 : 안산

패스워드 : 20

회원 가입

```
<h2>회원 가입</h2>
<form method="post">
  <p>
    <label>아이디 </label>
    <input type="text" name="memberid">
  </p>
  <p>
    <label>비밀번호 </label>
    <input type="password" name="passwd">
  </p>
  <p>
    <label>이름 </label>
    <input type="text" name="name">
  </p>
  <p>
    <label>나이 </label>
    <input type="text" name="age">
  </p>
  <input type="submit" value="가입">
  <input type="reset" value="취소">
</form>
```

register.html

member_list.html

```
<h2>회원 목록</h2>
<p>아이디 : {{ id }}</p>
<p>패스워드 : {{ pwd }}</p>
<p>패스워드 : {{ name }}</p>
<p>패스워드 : {{ age }}</p>
```

회원 가입

▪ app.py – register() 함수

```
# 회원 가입
@app.route("/register", methods=['GET', 'POST'])
def register():
    if request.method == "POST":
        id = request.form['memberid']
        pwd = request.form['passwd']
        name = request.form['name']
        age = request.form['age']
        return render_template('member_list.html', id=id, pwd=pwd, name=name, age=age)
    else:
        return render_template('register.html')
```

로그인

■ 로그인 구현

← → ↻ ⓘ 127.0.0.1:5000/login

아이디 :

패스워드 :



로그인 결과

아이디 : today

패스워드 : t1234

로그인

```
<h2>로그인</h2>
```

```
<form method="post">
```

```
<p><label>아이디 : </label>
```

```
<input type="text" name="uid"></p>
```

```
<p><label>패스워드 : </label>
```

```
<input type="password" name="passwd"></p>
```

```
<input type="submit" value="로그인">
```

```
</form>
```

login.html

login_result.html

```
<h3>로그인 결과</h3>
```

```
<p>아이디 : {{ id }}</p>
```

```
<p>패스워드 : {{ pwd }}</p>
```

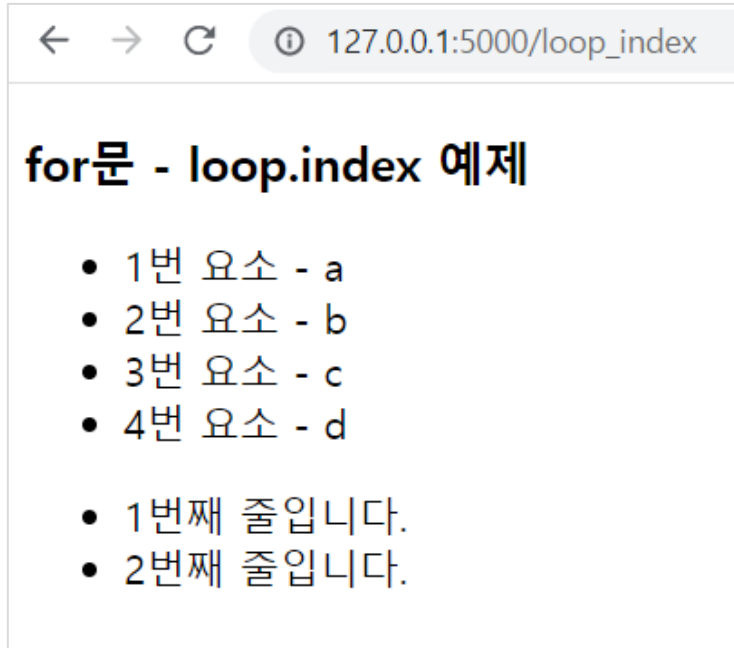
로그인

- app.py – login() 함수

```
# 로그인
@app.route("/login", methods = ['GET', 'POST'])
def login():
    if request.method == 'POST':
        uid = request.form['uid']
        passwd = request.form['passwd']
        return render_template('login_result.html', id=uid, pwd=passwd)
    else:
        return render_template('login.html')
```

loop 인덱스

▪ loop - 인덱스



A screenshot of a web browser window. The address bar shows the URL `127.0.0.1:5000/loop_index`. The main content area has the title **for문 - loop.index 예제** and a bulleted list of items.

for문 - loop.index 예제

- 1번 요소 - a
- 2번 요소 - b
- 3번 요소 - c
- 4번 요소 - d

- 1번째 줄입니다.
- 2번째 줄입니다.

loop 인덱스

loop_index.html

```
<h3>for문 - loop.index 예제</h3>
```

```
<ul>
```

```
  {% for item in items %}
```

```
    <li>
```

```
      {{ loop.index }}번 요소 - {{ item }}
```

```
    </li>
```

```
  {% endfor %}
```

```
</ul>
```

```
<ul>
```

```
  {% for item in items %}
```

```
    {% if loop.index < 3 %}
```

```
      <li>
```

```
        {{ loop.index }}번째 줄입니다.
```

```
      </li>
```

```
    {% endif %}
```

```
  {% endfor %}
```

```
</ul>
```

loop 인덱스

- app.py – loop_index()

```
# loop - index
@app.route("/loop_index", methods = ['GET'])
def get_loopindex():
    items = ['a', 'b', 'c', 'd']
    return render_template('loop_index.html', items = items)
```

짝수 / 홀수 판정

■ 연산 - 짝수 / 홀수 판정

← → ↻ ⓘ 127.0.0.1:5000/even_odd

숫자 입력 :



← → ↻ ⓘ 127.0.0.1:5000/even_odd

짝수/홀수 판정

10는(은) 짝수입니다.

문자를 입력한 경우(오류 처리)

숫자를 입력해주세요

숫자 입력 :

짝수/홀수 판정

■ 짝수 / 홀수 판정

```
<form method="post">
  {% if error_message %}
    <p>{{ error_message }}</p>
  {% endif %}
  <p>
    <label>숫자 입력 : </label>
    <input type="text" name="num">
  </p>
  <input type="submit" value="전송">
</form>
```

even_odd.html

result.html

```
<h2>짝수/홀수 판정</h2>
<h4>{{ num }}는(은) {{ result }}</h4>
```

짝수/홀수 판정

▪ app.py – even_odd()

```
# 짝수 / 홀수 판정
@app.route("/even_odd", methods = ['GET', 'POST'])
def even_odd():
    if request.method == 'POST':
        try:
            num = int(request.form['num'])
        except ValueError:
            error_message = "숫자를 입력해주세요"
            return render_template('even_odd.html', error_message=error_message)
        else:
            if num % 2 == 0:
                result = "짝수입니다."
            else:
                result = "홀수입니다."
            return render_template('calc_result.html', num=num, result=result)
    else:
        return render_template('even_odd.html')
```