

FLASK

WITH DEEL LEARNING

PART I

FLASK FRAMEWORK

FLASK FRAMEWORK

3

◆ Flask 살펴보기

- 파이썬으로 작성된 **마이크로 웹 프레임워크**
- **경량 웹 프레임워크**로 웹 서버로 필 수 기능만 포함
- 필요한 기능 추가로 **쉽게 확장** 가능
- **BSD 라이선스**, 상업적 사용 가능
- 설치 : `conda install -c conda-forge flask`



FLASK FRAMEWORK

4

◆ Flask 살펴보기

▪ 템플릿 엔진(Template Engine)

- 지정된 템플릿 양식과 특정 데이터 합성하여 결과 HTML 문서 출력하는 SW
- 웹 사이트 화면 쉽게 만들도록 도와주는 소프트웨어
- 필요성 : 코드량 감소, 데이터만 바뀌므로 HTML 재사용성, 유지보수 용이
- 분 류 : 서버 사이드 템플릿 엔진, 클라이언트 사이드 템플릿 엔진

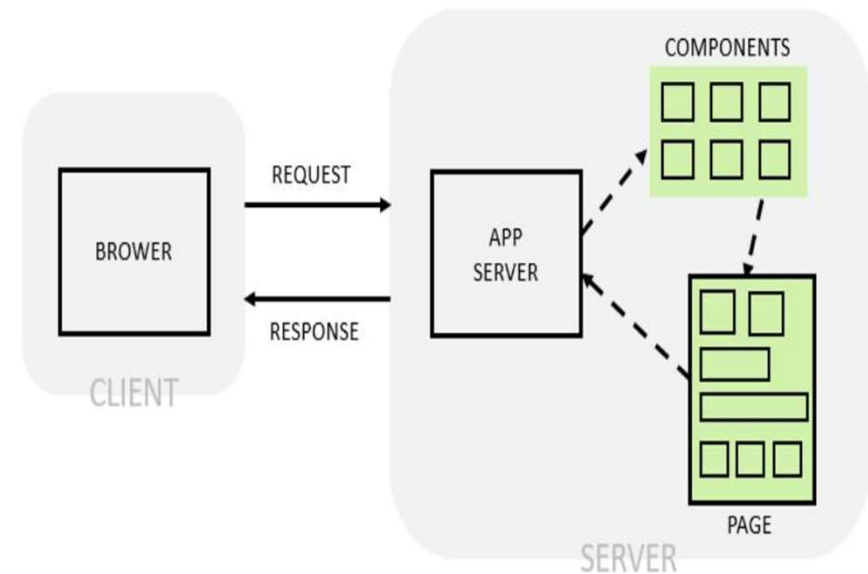
FLASK FRAMEWORK

5

◆ Flask 살펴보기

▪ 템플릿 엔진(Template Engine) : Server Side 템플릿 엔진

- DB 혹은 API에서 가져온 데이터를 미리 정의된 Template에 넣어 HTML 생성 후 클라이언트 전달
- HTML 고정 부분은 템플릿으로 **동적 생성 부분만 끼워 넣기**
- 엔진 : Jinja2, JSP 등등



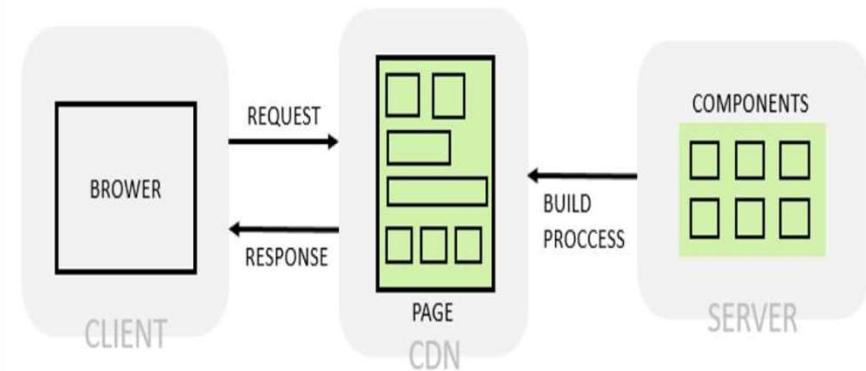
FLASK FRAMEWORK

6

◆ Flask 살펴보기

- 템플릿 엔진(Template Engine) : Client Side 템플릿 엔진

- 서버로부터 데이터 받아 DOM 객체에
동적으로 그려주는 프로세스 담당
- 엔진 : Mustache, Squirrelly, Handlebars



FLASK FRAMEWORK

7

◆ Flask 살펴보기

▪ 소프트웨어 디자인 패턴(Design Patten)

- 소프트웨어 설계에서 **공통 발생하는 문제**에 대해 **자주 쓰이는 설계 방법 정리한 패턴**
- 사용이유 : 개발 효율성, 유지 보수성, 운용성 높아지며 프로그램 최적화에 도움
- 협업 시 **공동작업이 쉬움**
- 서로 영향 주지 않아 **유지보수 더욱 용이**
- 기능 단위 **독립적 테스트** 쉬움

FLASK FRAMEWORK

8

◆ Flask 살펴보기

▪ 소프트웨어 디자인 패턴(Design Patten) - MVC

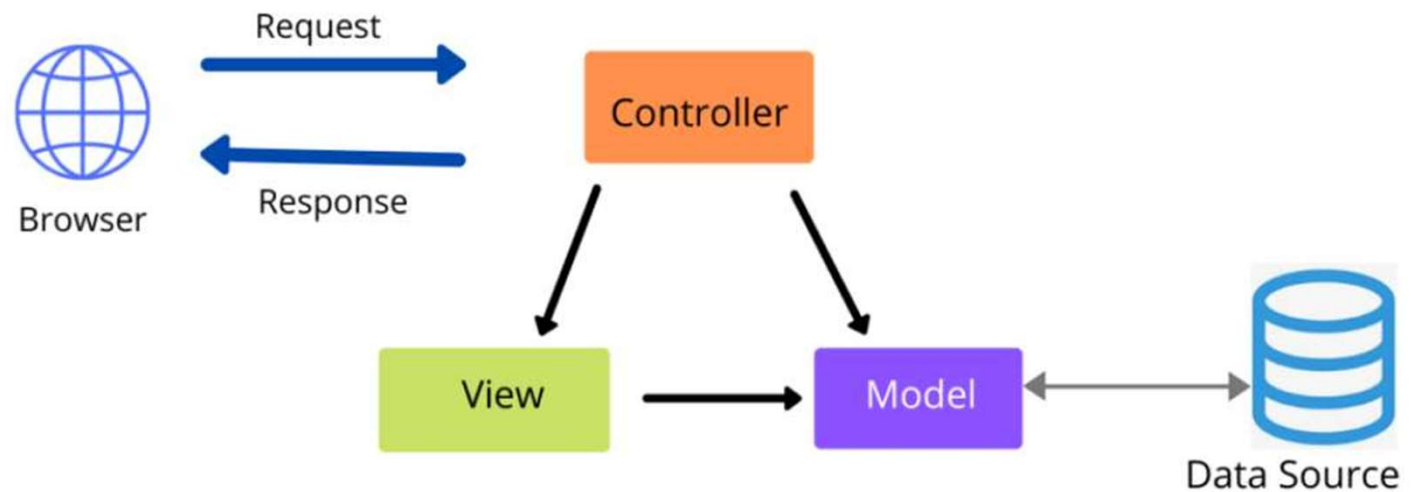
- 모델(model), 뷰(view), 컨트롤러(controller) 세 가지 역할 구분한 패턴
- 모델(model) → 백그라운드 로직, 데이터 조작 즉, DB제어 담당
- 뷰(view) → 사용자가 볼 수 있는 화면, 최종적인 출력 담당
- 컨트롤러(controller) → 요청 데이터 처리, 흐름 제어 (전체적인 관리)

FLASK FRAMEWORK

9

◆ Flask 살펴보기

- 소프트웨어 디자인 패턴(Design Patten) - MVC



FLASK FRAMEWORK

10

◆ Flask 살펴보기

▪ 소프트웨어 디자인 패턴(Design Patten) - MVT

- 모델(model), 뷰(view), 템플릿(Template) 세 가지 역할 구분한 패턴

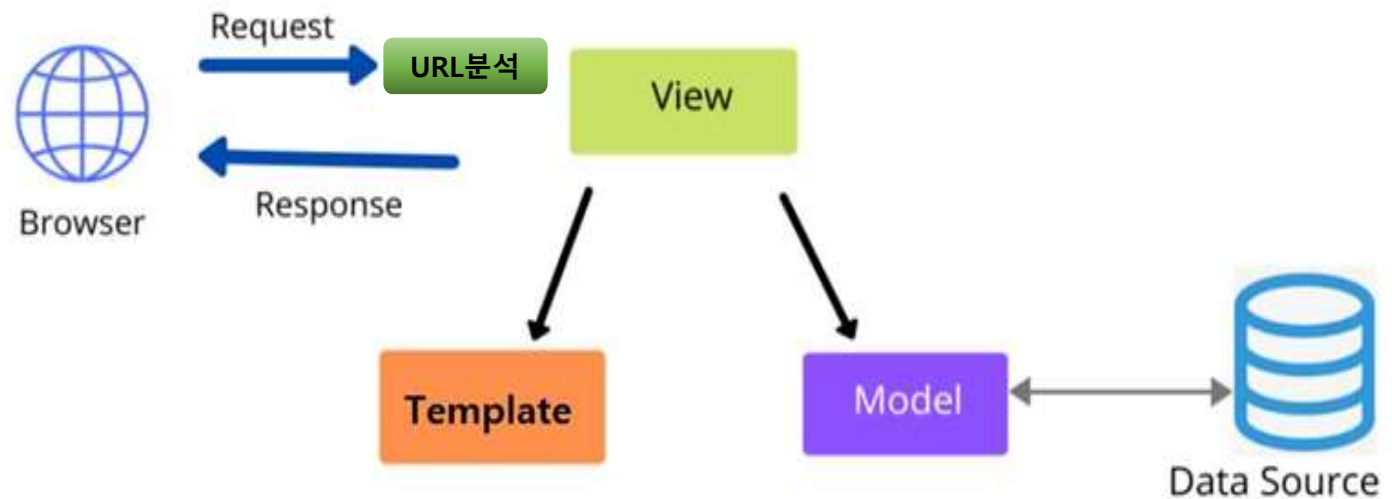
- | | |
|-----------------|-------------------------------|
| • 모델(model) | ➔ 백그라운드 로직, 데이터 조작 즉, DB제어 담당 |
| • 뷰(view) | ➔ 요청 데이터 처리, 흐름 제어 (전체적인 관리) |
| • 템플릿(Template) | ➔ 사용자가 볼 수 있는 화면, 최종적인 출력 담당 |

FLASK FRAMEWORK

11

◆ Flask 살펴보기

- 소프트웨어 디자인 패턴(Design Patten) - MVT



◆ WEB 개념 및 용어

▪ URI (Uniform Resource Identifier)

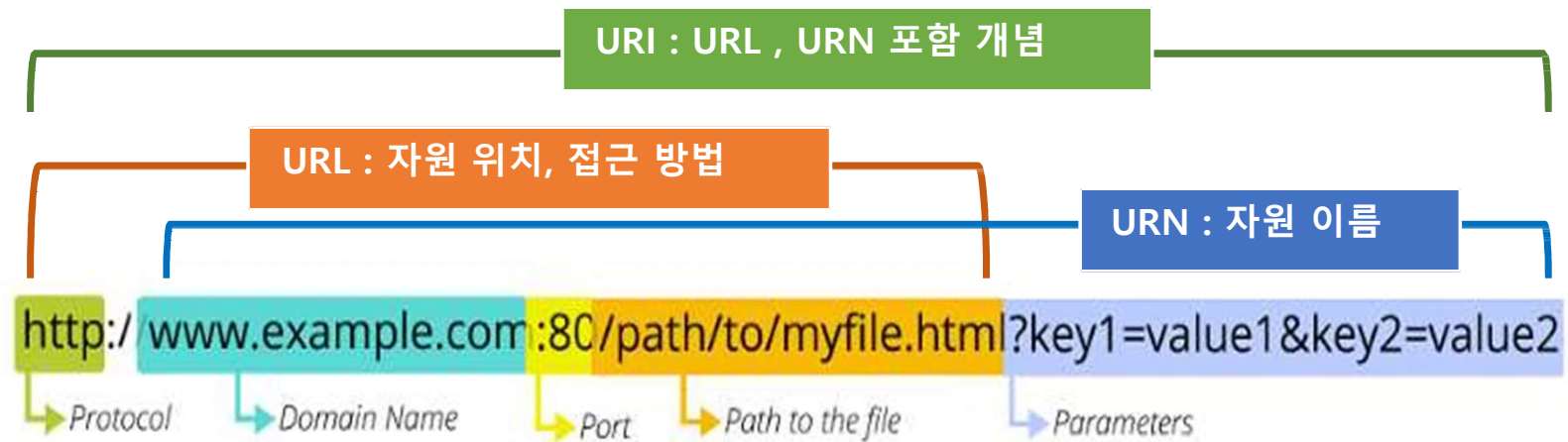
- 통합자원식별자, 인터넷 상에서 자원 식별하기 위한 고유한 문자열
- 하위 개념
 - URL (Uniform Resource Locator) : 자원(웹 페이지) 검색 위해 브라우저 사용 규약 (Where)
 - URN (Uniform Resource Name/Number) : 자원 경로와 식별 이름

FLASK FRAMEWORK

13

◆ WEB 개념 및 용어

- URI (Uniform Resource Identifier)



FLASK FRAMEWORK

14

◆ WEB 개념 및 용어

▪ 127.0.0.1 또는 localhost 의미

- 네트워크에서 사용하는 자신의 **컴퓨터 의미**
 - '**가상으로 인터넷망에 연결되어 있는 것처럼** 할당하는 인터넷 주소'
 - 로컬 컴퓨터를 원격 컴퓨터인 것처럼 통신할 수 있어 **테스트 목적으로 주로 사용**
-
- IPv4 => 127.0.0.1 localhost
 - IPv6 => ::1 localhost

FLASK FRAMEWORK

15

◆ WEB 개념 및 용어

▪ 애플리케이션 루트(root) 의미

- 앱을 실행하는 디렉토리
- 모듈이나 패키지를 읽어들이는 경로
- 패키지 및 프레임워크마다 다름

FLASK FRAMEWORK

16

◆ WEB 개념 및 용어

- 라우팅(Routing) : 사용자가 요청한 URL에 따라 해당 URL에 맞는 페이지를 보여주는 것

- 도메인이름서버(DNS:Domain Name Server) : 도메인 이름에 해당하는 IP 주소 반환 서버

- 인터넷서비스사업자(ISP:Internet Service Provider) : SKT, KT, LGU+ 등 DNS 서버에 도메인명

IP 주소 제공 사업자

FLASK FRAMEWORK

17

◆ WEB 개념 및 용어

▪ Web Server

- 웹 브라우저 클라이언트로부터 HTTP 요청 받아 **정적인 콘텐츠(.html .jpeg .css 등) 제공** SW
- Ex) APACHE SERVER, MICROSOFT IIS, NGINX, Google Web Server ,....

서버 부하 방지
보안, 무중단 운용

▪ WAS : Web Application Server

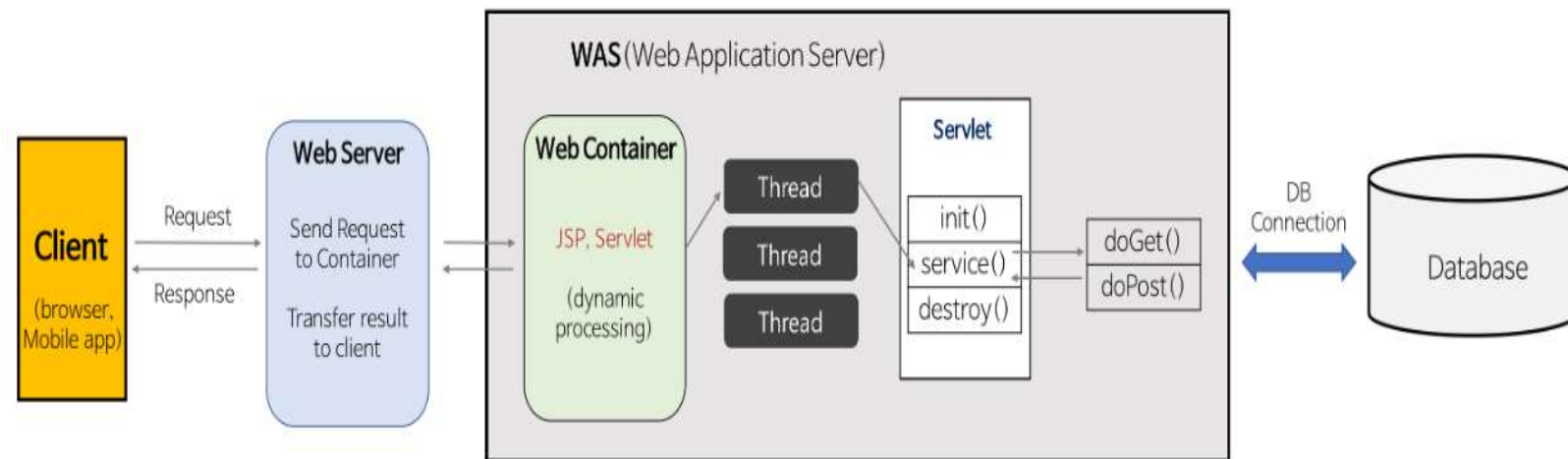
- DB 조회나 다양한 로직 처리를 요구하는 **동적인 콘텐츠 제공**하기 위해 만들어진 SW
- **Web Server 기능**들 구조적으로 **분리하여 처리하고자**하는 목적으로 제시
- Ex) Tomcat, JBoss, Jeus, Web Sphere 등

FLASK FRAMEWORK

18

◆ WEB 개념 및 용어

▪ Web Server 구조



◆ WEB 개념 및 용어

▪ WSGI (Web Server Gateway Interface : 위스키)

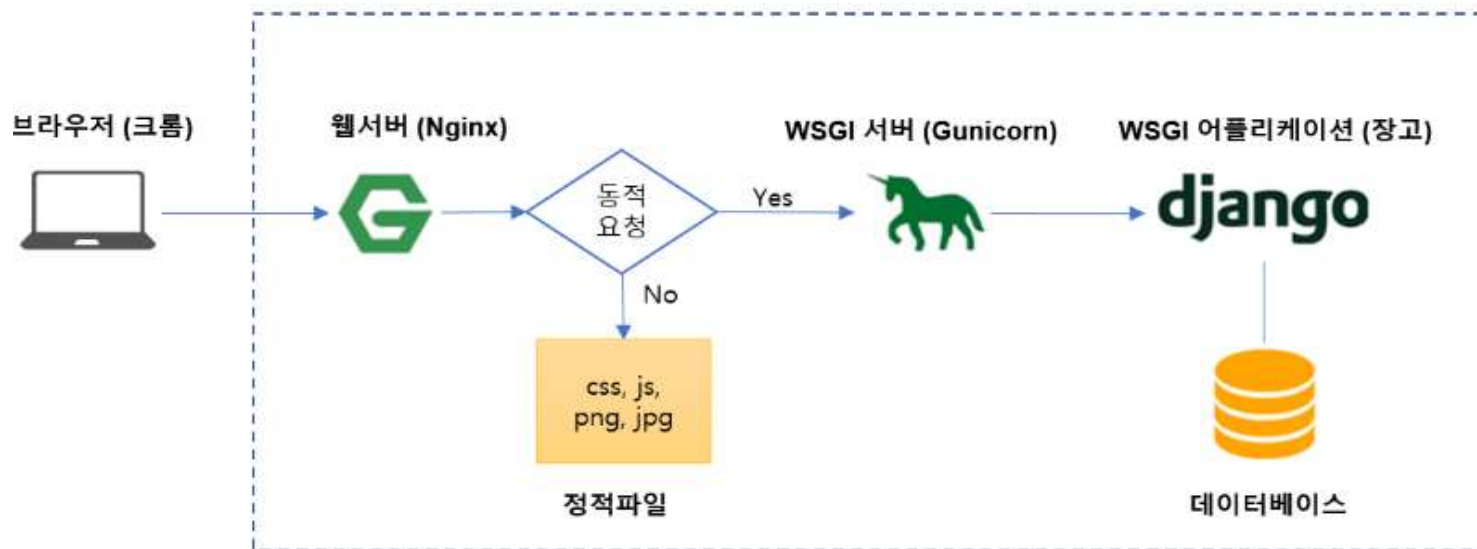
- 웹 서버와 웹 어플리케이션 서버 간의 통신하는 규칙이 필요
 - Web Server에서 요청에 대한 정보를 Application에 전달하기 위해 사용하는 인터페이스
-
- HTTP Request 정보를 가지고 Logic 수행한 뒤에 Callback Function 통해 결과 웹서버에 반환
 - WSGI Interface 구현하는 Web Server나 Application >> WSGI compatible >> WSGI Application

FLASK FRAMEWORK

20

◆ WEB 개념 및 용어

- WSGI (Web Server Gateway Interface : 위스키)



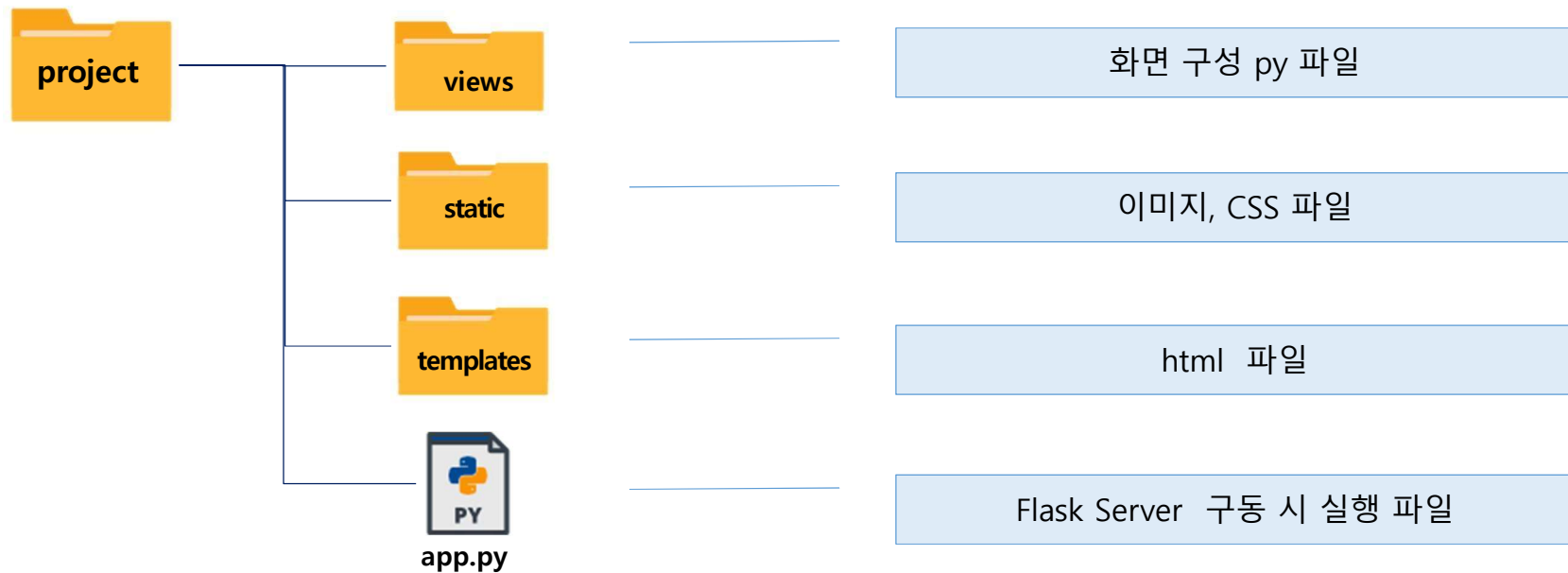
FLASK PROJECT - BASIC

FLASK PROJECT

22

◆ FLASK WEB SERVER 디렉터리 구조

❖ 간단한 기본 구조



FLASK PROJECT

23

◆ FLASK WEB SERVER 디렉터리 구조

❖ WEB SERVER 프로젝트 폴더 생성

```
① > mkdir FLASK_D01  
> cd FLASK_D01
```

```
② FLASK_D01 > mkdir MyWeb  
FLASK_D01 > cd MyWeb
```

← 프로젝트 홈 디렉터리 즉, 애플리케이션 루트

```
③ FLASK_D01 ₩ MyWeb > mkdir static  
FLASK_D01 ₩ MyWeb > mkdir templates  
FLASK_D01 ₩ MyWeb > mkdir views
```

FLASK PROJECT

24

◆ Flask 프로젝트 환경 구축

❖ app.py

```
### ==> 모듈로딩
from flask import Flask

### ==> 전역변수
APP = Flask(__name__)

### ==> 라우팅(Routing) 기능 함수들
@APP.route("/")
def index():
    return "<h1><font color='blue'>Hello MyWeb~!!</font></h2>"

### ==> 조건에 따른 실행 처리
if __name__ == '__main__':
    APP.run()
```


FLASK PROJECT

25

◆ Flask 프로젝트 환경 구축

❖ WEB SERVER 구동

➤ 프로젝트 홈 디렉터리 > flask run

```
(TORCH_NLP38) C:\Users\anece\EXAM_SERVICE_ML\Flask_D01\MyWeb>flask run
```

```
* Debug mode: off
```

```
WARNING: This is a development server. Do not use it in a production deployment.  
instead.
```

```
* Running on http://127.0.0.1:5000
```

```
Press CTRL+C to quit
```

Ctrl + 클 릭

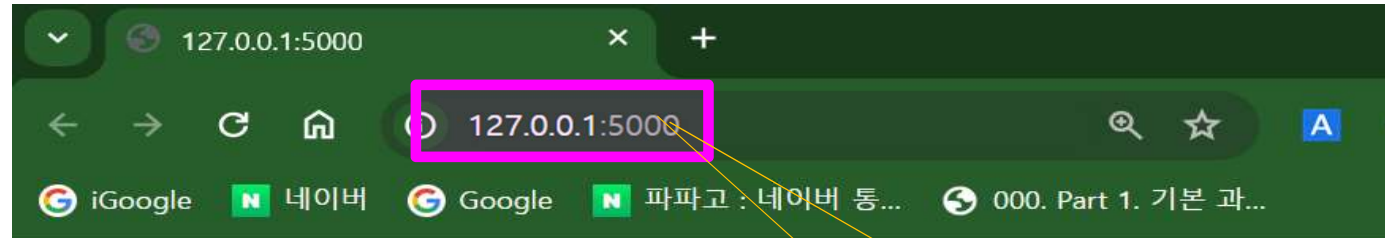
FLASK PROJECT

26

◆ Flask 프로젝트 환경 구축

❖ WEB SERVER 구동

➤ 프로젝트 홈 디렉터리 > flask run



Hello MyWeb~!!

5000 포트번호

루트(/) URL

FLASK PROJECT

27

◆ Flask 프로젝트 환경 구축

❖ flask 명령어

➤ flask [--help](#)

```
(TORCH_NLP38) C:\Users\anece\EXAM_SERVICE_ML\Flask_D01\SimpleWeb>flask --help
Usage: flask [OPTIONS] COMMAND [ARGS]...
```

A general utility script for Flask applications.

An application to load must be given with the '--app' option, 'FLASK_APP' environment variable, or with a 'wsgi.py' or 'app.py' file in the current directory.

Options:

-e, --env-file FILE	Load environment variables from this file. python-dotenv must be installed.
-A, --app IMPORT	The Flask application or factory function to load, in the form 'module:name'. Module can be a dotted import or file path. Name is not required if it is 'app', 'application', 'create_app', or 'make_app', and can be 'name(args)' to pass arguments.
--debug / --no-debug	Set debug mode.
--version	Show the Flask version.
--help	Show this message and exit.

Commands:

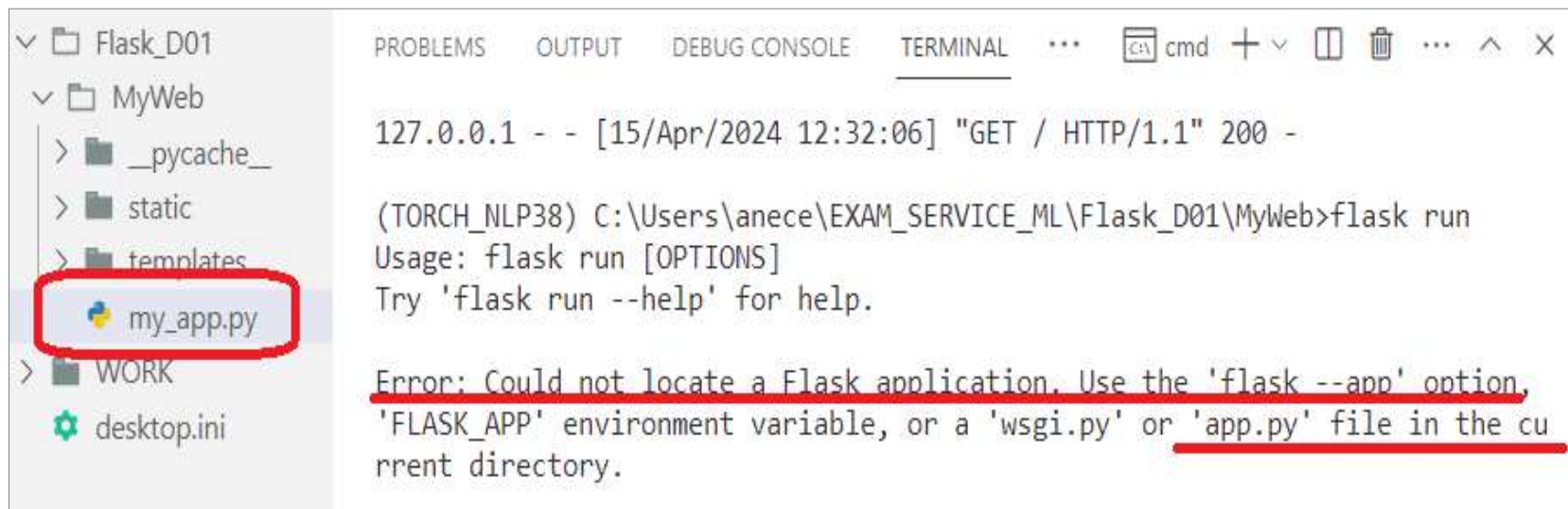
routes	Show the routes for the app.
run	Run a development server.
shell	Run a shell in the app context.

FLASK PROJECT

28

◆ FLASK 환경변수

❖ Flask 실행 시 → app.py 파일명 아닌 경우



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL ... cmd + v [icon] [icon] ... ^ x

127.0.0.1 - - [15/Apr/2024 12:32:06] "GET / HTTP/1.1" 200 -

(TORCH_NLP38) C:\Users\anece\EXAM_SERVICE_ML\Flask_D01\MyWeb>flask run
Usage: flask run [OPTIONS]
Try 'flask run --help' for help.

Error: Could not locate a Flask application. Use the 'flask --app' option,
'FLASK_APP' environment variable, or a 'wsgi.py' or 'app.py' file in the cu
rrent directory.
```

FLASK PROJECT

29

◆ FLASK 환경변수

❖ Flask 실행 시 → app.py 파일명 아닌 경우 : **FLASK_APP** 환경변수 설정

```
MyWeb > set FLASK_APP=my_app.py
```

← 일시적

```
(TORCH_NLP38) C:\Users\anece\EXAM_SERVICE_ML\Flask_D01\MyWeb>set FLASK_APP=my_app.py
```

```
(TORCH_NLP38) C:\Users\anece\EXAM_SERVICE_ML\Flask_D01\MyWeb>flask run
```

```
* Serving Flask app 'my_app.py'
```

```
* Debug mode: off
```

```
WARNING: This is a development server. Do not use it in a production deployment. Use a prod  
stead.
```

```
* Running on http://127.0.0.1:5000
```

```
Press CTRL+C to quit
```

FLASK PROJECT

30

◆ FLASK 환경변수

❖ Flask 실행 시 → app.py 파일명 아닌 경우 : 환경변수 .env 파일 설정

① 패키지 설치 : conda install **python-dotenv**

```
(TORCH_NLP38) C:\Users\anece\EXAM_SERVICE_ML\Flask_D01\MyWeb>conda install python-dotenv
Channels:
  - defaults
  - conda-forge
  - pytorch
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\ProgramData\anaconda3\envs\TORCH_NLP38

  added / updated specs:
    - python-dotenv
```

FLASK PROJECT

31

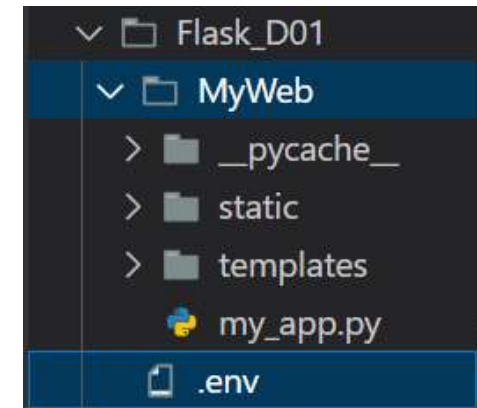
◆ FLASK 환경변수

❖ Flask 실행 시 → app.py 파일명 아닌 경우 : **환경변수 .env 파일 설정**

② .env 파일 생성

```
# Flask Server File
FLASK_APP = MyWeb.my_app.py

# Flask Server Debug On, Auto Reload
FLASK_DEBUG = True
```



FLASK PROJECT

32

◆ ROUTING - URL Binding

❖ 기본 URL

```
app = Flask(__name__)
```

```
@app.route('/')  
def XXX(): return code
```

```
← http://127.0.0.1:5000/
```

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

```
def XXX(): return code  
← http://127.0.0.1:5000/test
```

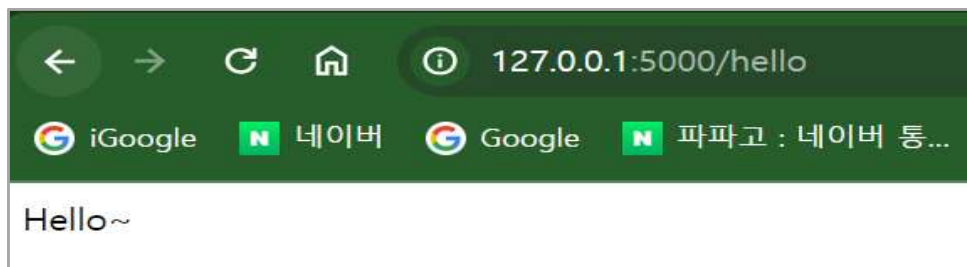
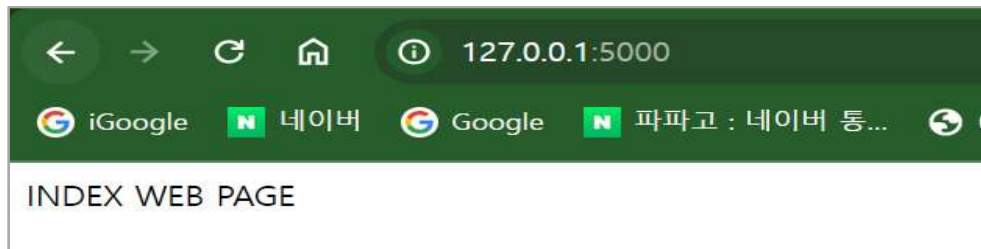
[예] http://127.0.0.1:5000/info

FLASK PROJECT

33

◆ ROUTING - URL Binding

❖ 기본 URL



```
### ==> 모듈로딩
from flask import Flask

### ==> 전역변수
app = Flask(__name__)

### ==> 라우팅 처리
@app.route('/')
def index():
    return 'INDEX WEB PAGE'

@app.route('/hello')
def hello():
    return 'Hello~'
```

◆ ROUTING - URL Binding

❖ 변수 URL

- 형식 : `<converter:variable_name>`
- 타입

string	(기본값) 슬래시 없는 문자열 가능
int	양의 정수만 가능
float	양의 실수만 가능
path	슬래시가 포함된 문자열 가능
uuid	UUID 문자열 가능

```
### ==> 모듈로딩
from flask import Flask

### ==> 라우팅 처리
@app.route('/name')
def hello():
    return 'Name Name'

### ==> 라우팅 처리
@app.route('/<name>')
def hello2(name):
    return f'Name : {name}'

@app.route('/<int:number>')
def show_number(number):
    return f'Select Number :{number}'
```

◆ ROUTING - URL Binding

❖ 리다이렉션 URL

- 다른 웹 페이지로 다시 전달/이동
- 형 식 : URL 뒤에 오는 슬래시(/)
- 메소드 : 인스턴스명.redirect(URL_string)

```
### ==> 라우팅 처리
@app.route('/about')
def about():
    return ' A.B.O.U.T '

@app.route('/projects/')
def show_project():
    return '➔ project ..... '

@app.route('/user_info2')
def user_login2():
    return app.redirect('/')
```

◆ ROUTING - URL Binding

❖ HTML 파일 렌더링

- 메소드 : render_template(HTML 파일)
- 위 치 : templates 폴더 내

```
### 모듈 로딩
from flask import Flask , render_template

### 전역변수
myapp=Flask(__name__)

### 웹 서버의 첫 페이지
@app.route('/')
def index_page():
    return render_template('index.html')
```

FLASK PROJECT - RECOMMENDED

FLASK PROJECT

38

◆ HTTP 프로토콜

❖ HTTP 메서드

방식	특징
GET	서버로 데이터 전송, 자원 조회 , SELECT , QueryString 통해 전달
POST	서버로 데이터 전송, 새로운 데이터 생성, CREATE/INSERT , 보안, KEY-Value BODY 에 담아서 전송, 길이 제한
PUT	전체 데이터 덮어쓰기, 없으면 생성
PATCH	일부 데이터 변경 즉 업데이트
DELETE	자원 삭제

FLASK PROJECT

39

◆ HTTP 프로토콜

❖ REQUEST MESSAGE 객체 - POST 메서드

POST /?id=1 HTTP/1.1

Request line

Host: www.swingvy.com

Content-Type: application/json; charset=utf-8

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:53.0)

Gecko/20100101 Firefox/53.0

Connection: close

Content-Length: 136

Header

Header와 Body message 구분 공백

```
{
  "status": "ok",
  "extended": true,
  "results": [
    {"value": 0, "type": "int64"},
    {"value": 1.0e+3, "type": "decimal"}
  ]
}
```

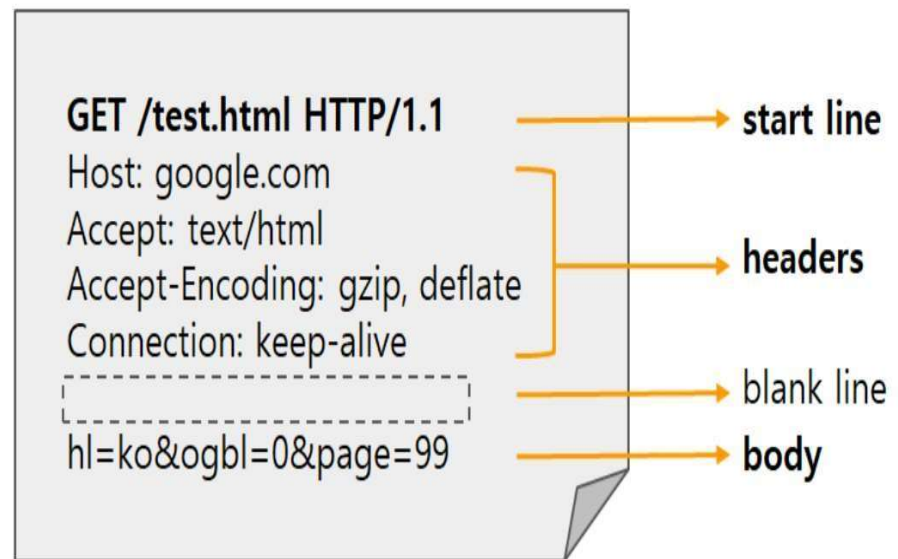
Body message

FLASK PROJECT

40

◆ HTTP 프로토콜

❖ REQUEST MESSAGE 객체 - GET 메서드



◆ HTTP 프로토콜

❖ RESPONSE 코드

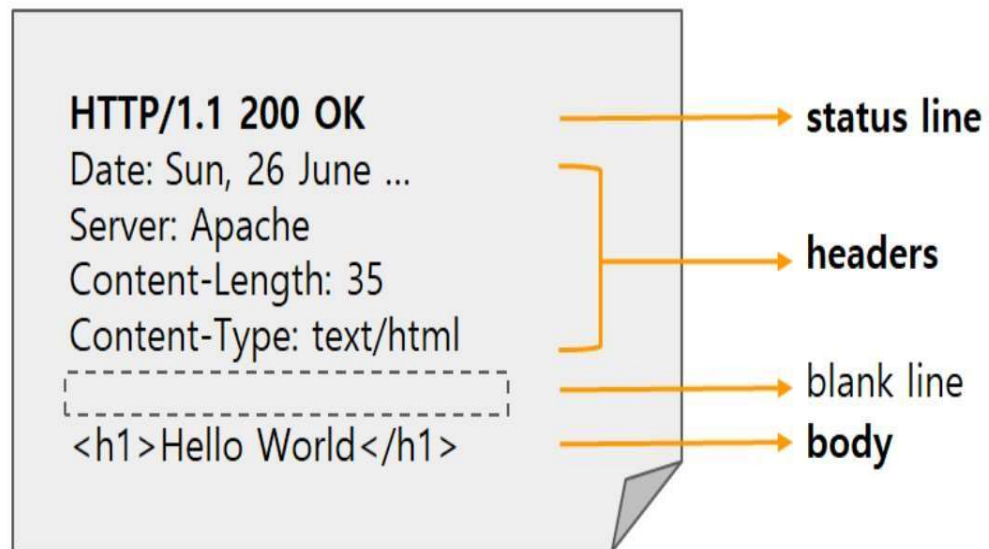
2xx	성공, 요청 정상 처리
3xx	리다이렉션, 요청을 완료하기 위해 다른 주소로 이동
4xx	클라이언트 오류. 올바르지 않은 요청
5xx	서버 오류. 올바른 요청에 대해 서버의 문제로 응답 불가능

FLASK PROJECT

42

◆ HTTP 프로토콜

❖ RESPONSE MESSAGE 객체



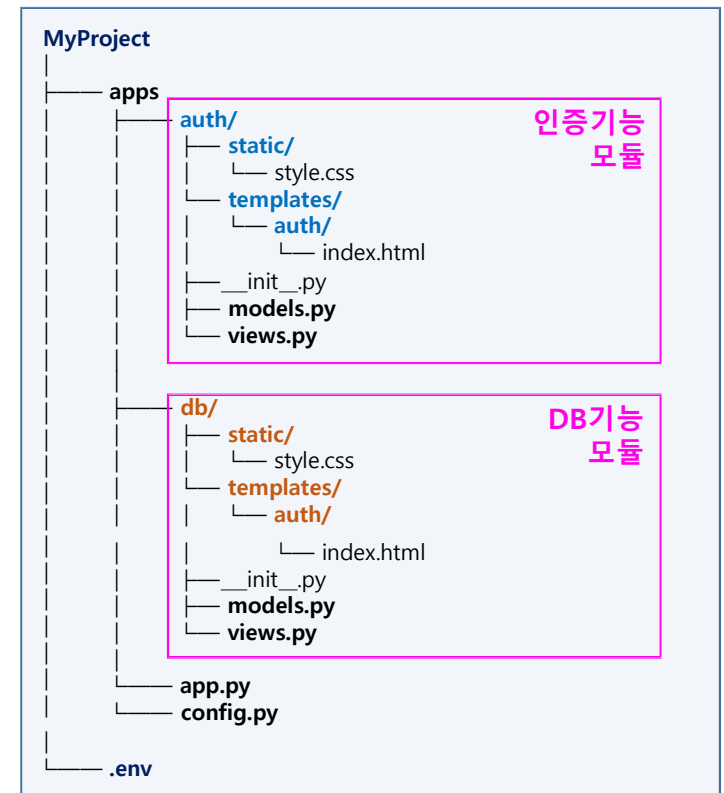
FLASK PROJECT

43

◆ FLASK WEB SERVER 디렉터리 구조

❖ 권장 구조

<https://flask.palletsprojects.com/en/3.0.x/tutorial/layout/>



FLASK PROJECT

44

◆ FLASK WEB SERVER 디렉터리 구조

❖ __init__.py 파일

- 패키지 관련된 설정이나 초기화 코드 작성
- 패키지 폴더 내에 위치하며 해당 폴더가 패키지로 인식되도록 함
 - python 3.3버전부터 없어도 패키지로 인식함
 - 하위 버전 호환을 위해서 파일 생성하는 것이 안전
- 패키지 폴더 내에 모듈 미리 인식하여 사용 간편하도록 설정

FLASK PROJECT

45

◆ FLASK WEB SERVER 디렉터리 구조

❖ config.py 파일

- 프로젝트 환경 설정 파일

❖ .env 파일

- FLASK 환경 설정 파일

FLASK PROJECT

46

◆ FLASK WEB SERVER 디렉터리 구조

❖ modules.py 파일

- DB관련 클래스 및 제어 기능

❖ views.py 파일

- 모듈 객체 생성 및 파우팅 기능

FLASK PROJECT

47

◆ FLASK WEB SERVER 디렉터리 구조

❖ app.py 파일

- Flask 객체 생성 및 모듈 객체 연결
- DB연동 및 초기화
- `__init__.py` 파일명으로 가능



FLASK PROJECT

48

◆ FLASK APPLICATION FACTORY

❖ 순환 참조(circular import) 오류

- A 모듈이 B 모듈을 참조하고 B 모듈이 다시 A 모듈을 참조하는 경우

```
### ==> 전역변수  
APP = Flask(__name__)
```

프로젝트 규모 커질수록 문제 발생 확률 높음

❖ 해결책 : 객체 반환 함수 즉, Application Factory 함수 `create_app()` 사용

FLASK PROJECT

49

◆ FLASK APPLICATION FACTORY

❖ create_app()

- 애플리케이션 팩토리 함수명
- 함수명 다른 이름 변경 불가
- 생성된 Flask 객체 반환

```
def create_app():  
    ### 서버 인스턴스 생성  
    app = Flask(__name__)  
  
    ### 라우팅 함수들  
    @app.route('/')  
    def hello ():  
        return 'Hello ^^'  
  
    ### Flask Server 반환  
    return app
```