AI WITH FLASK

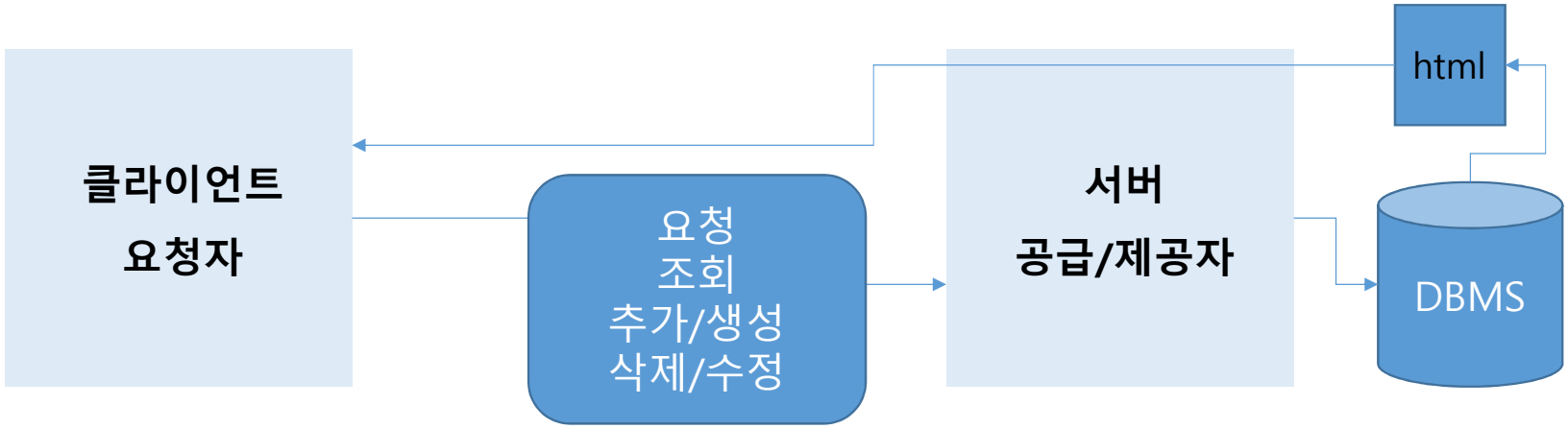
PART I

REQUEST & RESPONSE

REQUEST & RESPONSE

◆ 라우팅

- 클라이언트 요청 URI 처리



REQUEST & RESPONSE

◆ HTTP 메서드

클라이언트와 서버 사이에 요청(Request)과 응답(Response) 데이터를 전송하는 방식

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

REQUEST & RESPONSE

◆ URL Route Registraion

- **endpoint**
  - 웹 애플리케이션에서 클라이언트가 서버에 요청을 보내는 특정 URL 경로 의미
- **route('URL')**
  - 클라이언트 요청 URL에 대한 처리 함수 등록
- **add_url_rule('URL', endpoint, view_func_name)**
  - 클라이언트 요청 URL에 대한 처리 함수 등록
  - route()와 동일

REQUEST & RESPONSE

◆ URL Route Registraion

- **route('URL') & add_url_rule('URL', endpoint, view_func_name)**

```
@app.route('/')
def index():
    return 'route("/") '
```

endpoint 미지정이면 함수 이름을 endpoint로 지정

```
@app.endpoint("index")
def index():
    print('Dssss -> index()')
    return render_template('index.html')

app.add_url_rule("/", view_func=index)
```

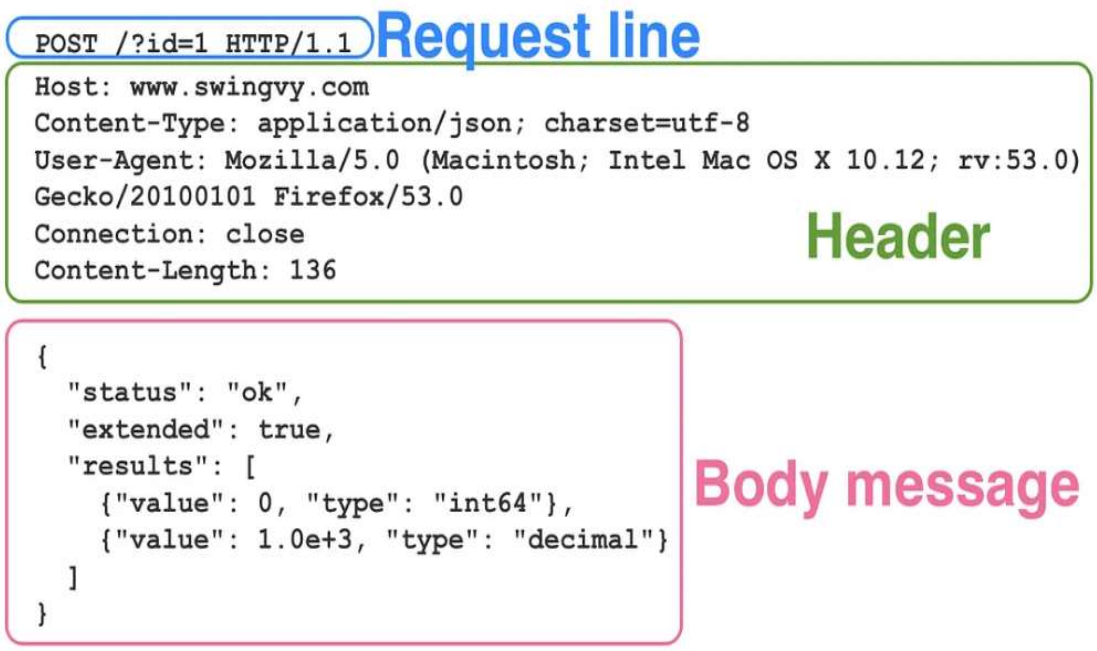
```
app.add_url_rule("/", endpoint='index')

@app.endpoint("index")
def index():
    print('Dssss -> index()')
    return render_template('index.html')
```

REQUEST & RESPONSE

◆ HTTP 프로토콜

- **Request Message 객체**

  By POST Method

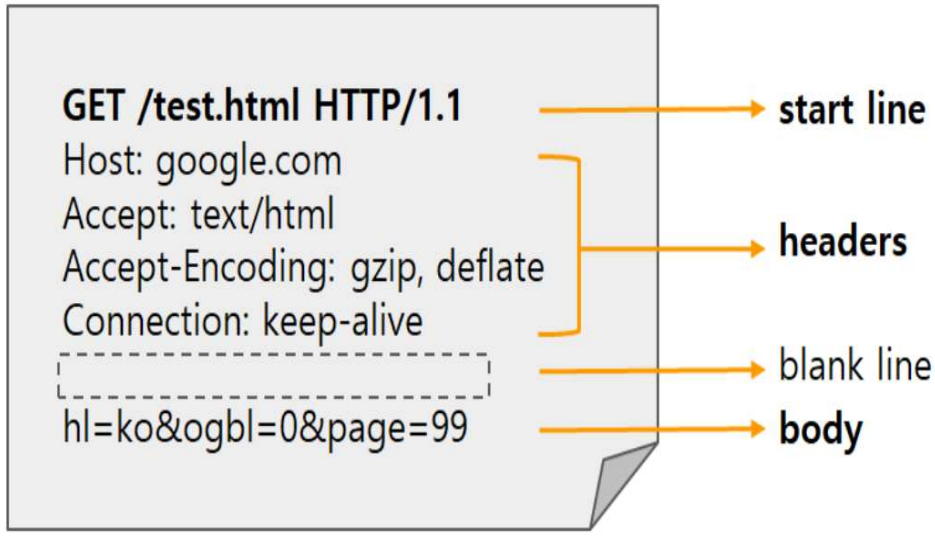


REQUEST & RESPONSE

◆ HTTP 프로토콜

- **Request Message 객체**

  By GET Method



REQUEST & RESPONSE

◆ HTTP 프로토콜

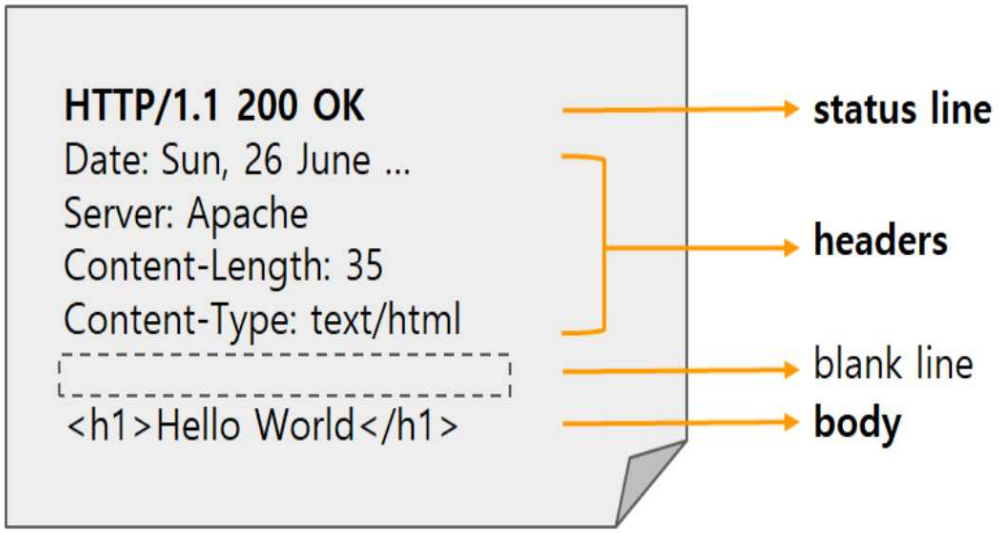
- **Response 코드**

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

REQUEST & RESPONSE

◆ HTTP 프로토콜

- **Response Message 객체**



REQUEST & RESPONSE

◆ HTTP 프로토콜

- **특정**
  - **Connectionless(비연결성)** : 응답(Response)을 보내고 **연결 끊김**

    ➔ HTML1.1 keep-alive 기본

  - **Stateless(무상태)** : 연결이 끊어지는 순간 모든 상태 정보 사라짐

REQUEST & RESPONSE

◆ HTTP 프로토콜

- **단점**
  - 평문 통신이기 때문에 도청이 가능
  - 통신 상대를 확인하지 않기 때문에 위장이 가능
  - 완전성을 증명할 수 없기 때문에 변조가 가능

➔ **HTTPS**

- Secure 강화
- 어떤 웹 서버에 보내는 정보를 다른 사람이 보지 못하도록 암호화
- 접속하려는 사이트가 신뢰할 수 있는 사이트인지 확인하여 수상한 사이트 걸러냄

REQUEST & RESPONSE

◆ 쿠키(Cookie)와 세션(Session)

- Connectionless(비연결성) : 응답(Response)을 보내고 연결 끊김
- Stateless(무상태) : 연결이 끊어지는 순간 모든 상태 정보 사라짐

해결방안

서버는 통신할 때마다 클라이언트가 누구인지 인증 필요

REQUEST & RESPONSE

◆ 쿠키(Cookie)

- 웹사이트 방문 시 해당 웹사이트에서 **사용자의 컴퓨터 또는 장치에 저장되는 작은 파일**
- 인터넷 사용자의 **브라우저에 의해 읽**혀짐 ➔ **접속 클라이언트 인식**
- 해당 웹사이트를 방문할 때마다 **인터넷 사용자의 정보를 수집하고 저장**
- 웹사이트가 인터넷 사용자에 대한 정보 기억, **사용자가 효율적으로 이용할 수 있도록 함**

REQUEST & RESPONSE

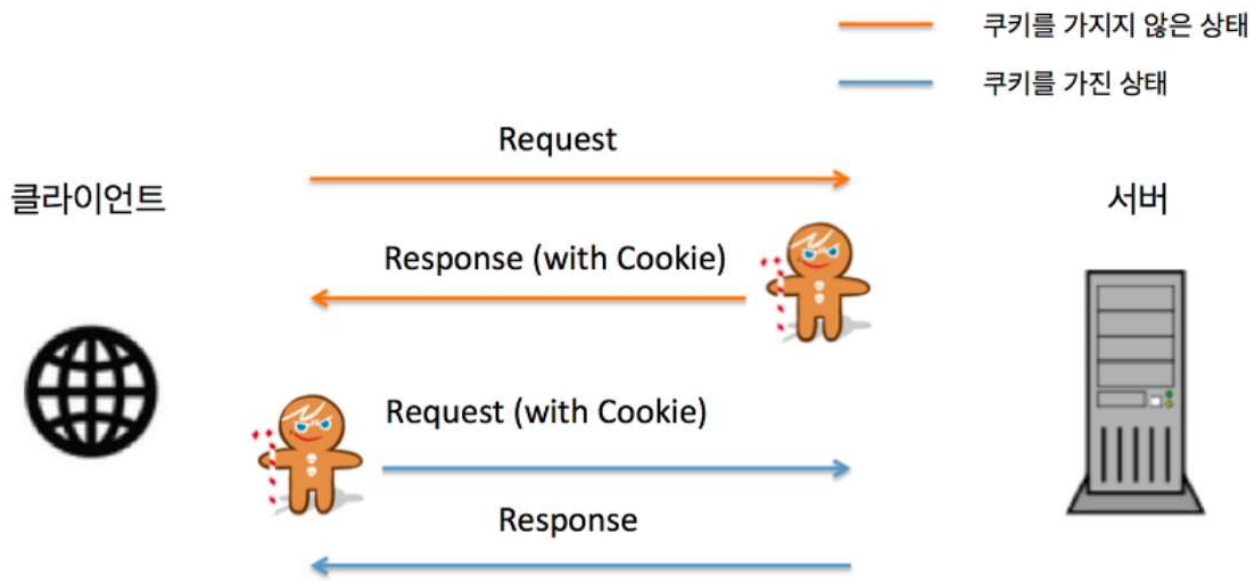
◆ 쿠키(Cookie)

- **역할**
  - **세션관리** : 서버에 저장해야 할 데이터 관리 => 민감 & 보안 정보
  - **개인 맞춤 데이터** : 테마 등과 같은 세팅값
  - **트래킹** : 사용자의 행동 기록 및 분석

REQUEST & RESPONSE

◆ 쿠키(Cookie)

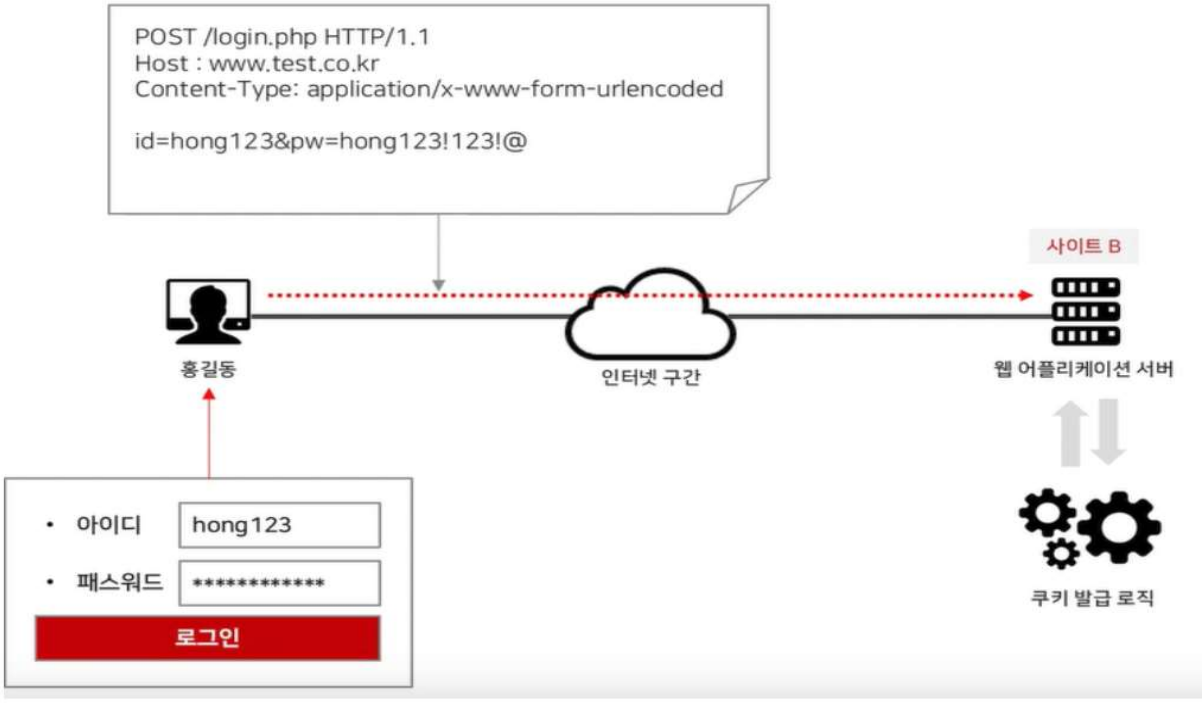
- **동작방식**



REQUEST & RESPONSE

◆ 쿠키(Cookie)

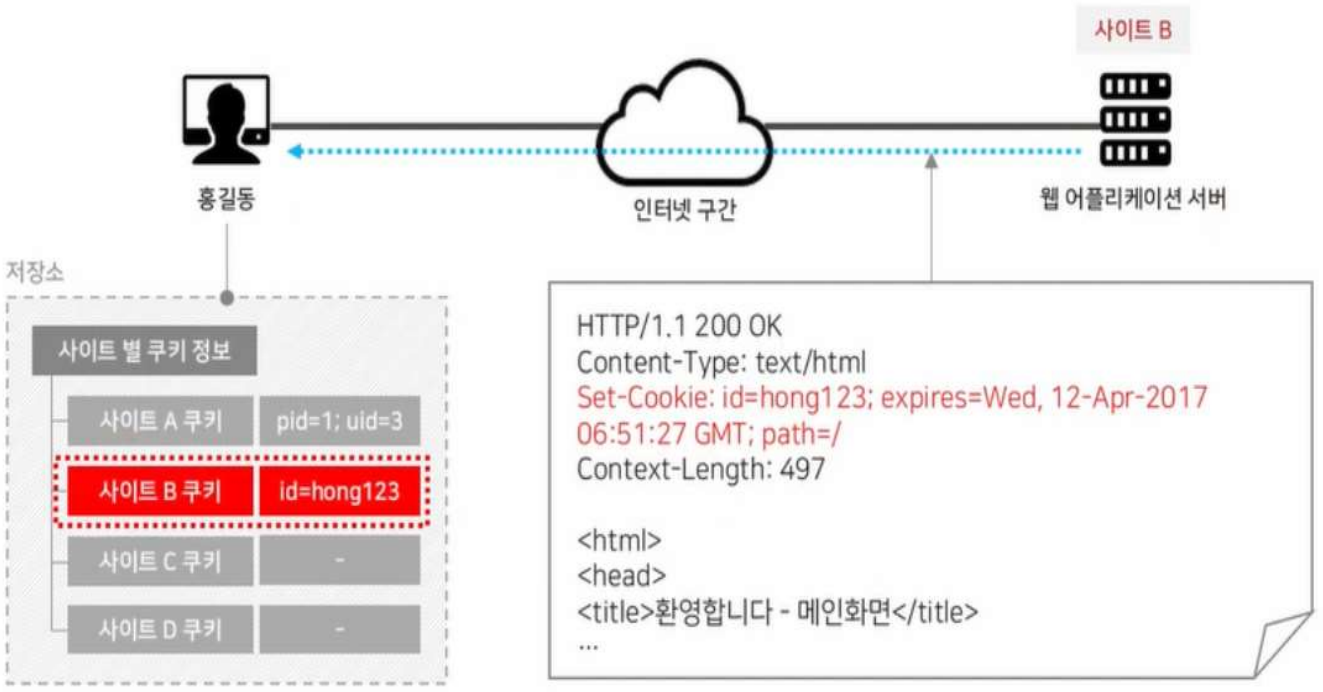
- 최초 방문 시



REQUEST & RESPONSE

◆ 쿠키(Cookie)

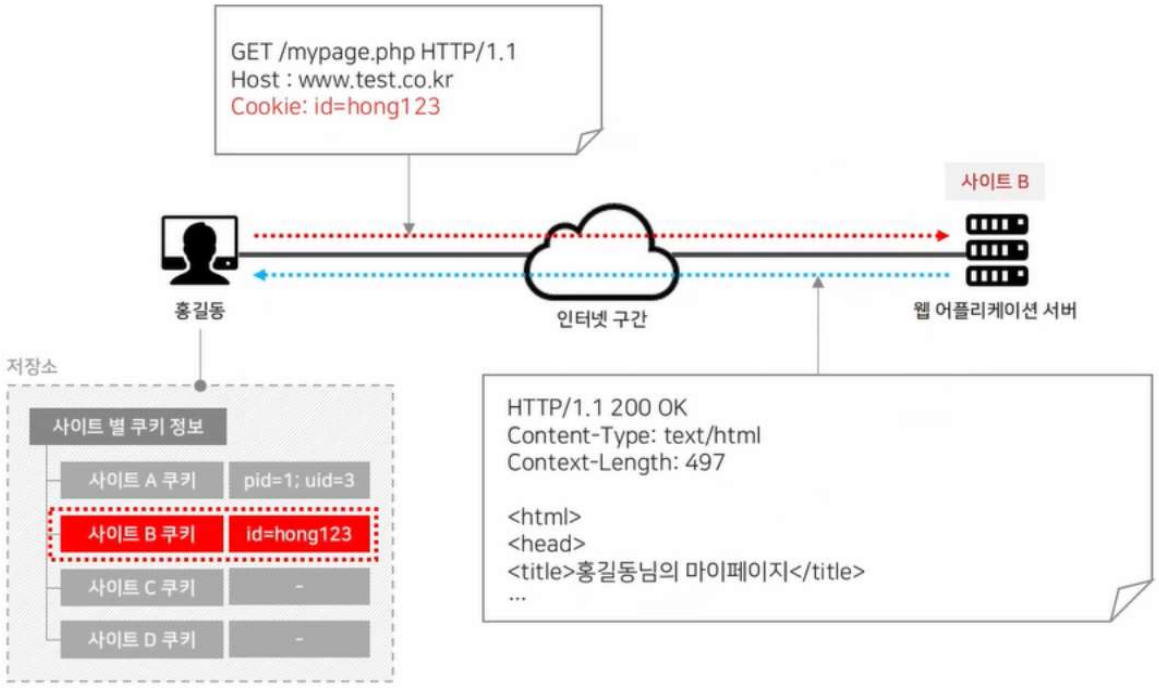
- 최초 방문 시 응답



REQUEST & RESPONSE

◆ 쿠키(Cookie)

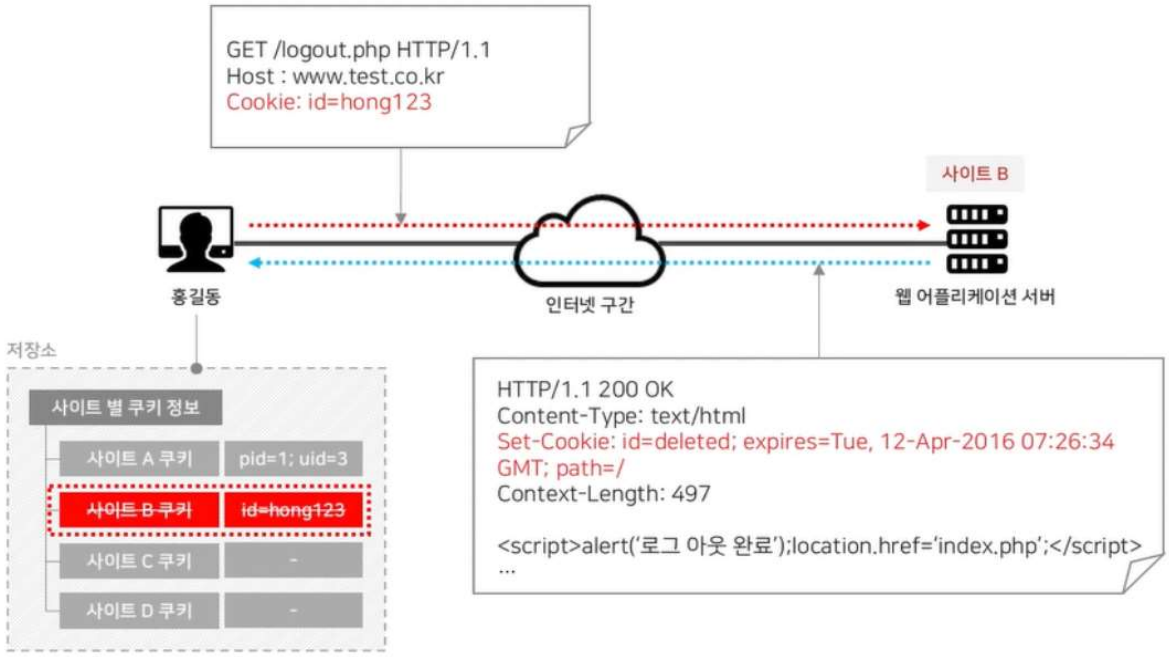
- 다시 방문 시



REQUEST & RESPONSE

◆ 쿠키(Cookie)

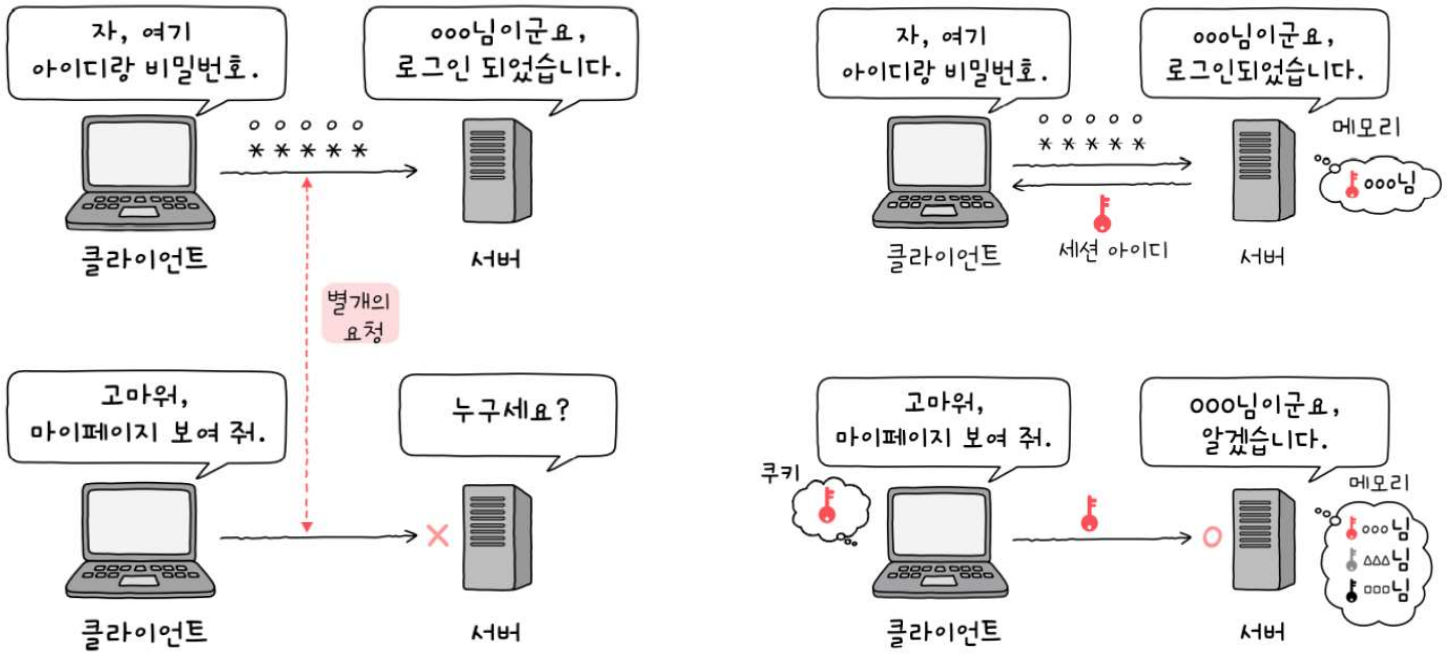
- 로그아웃



REQUEST & RESPONSE

◆ 세션(Session)

- **필요성**



REQUEST & RESPONSE

◆ 세션(Session)

- 세션은 **쿠키 기반**, **사용자 정보 파일을 서버 측에서 관리**
- **서버에서 클라이언트 구분 위해 세션 ID 부여**
- 웹 브라우저가 서버에 접속해서 **브라우저를 종료할 때가지 인증상태 유지**

- **임의의 문자들이 무작위로 나열**된 것으로 **특정 사용자의 세션 추측 어려움**
- 사용자에 대한 정보를 서버에 저장하기 때문에 **보안 면에서 쿠키보다 우수**
- **사용자가 많아질수록 서버에 과부하**를 주게 되므로 성능 저하의 요인

REQUEST & RESPONSE

◆ 세션

- **해결방안 ➔ 세션 아이디**

서버로부터 인증받았음을 증명해 주는 세션이라는 증서가 필요

사용자가 서버에 올바른 아이디와 비밀번호로 로그인 성공

➔ 서버는 세션 아이디 생성

➔ '2sd98dbawix4'와 같은 식으로 알파벳과 숫자가 혼합된 형식

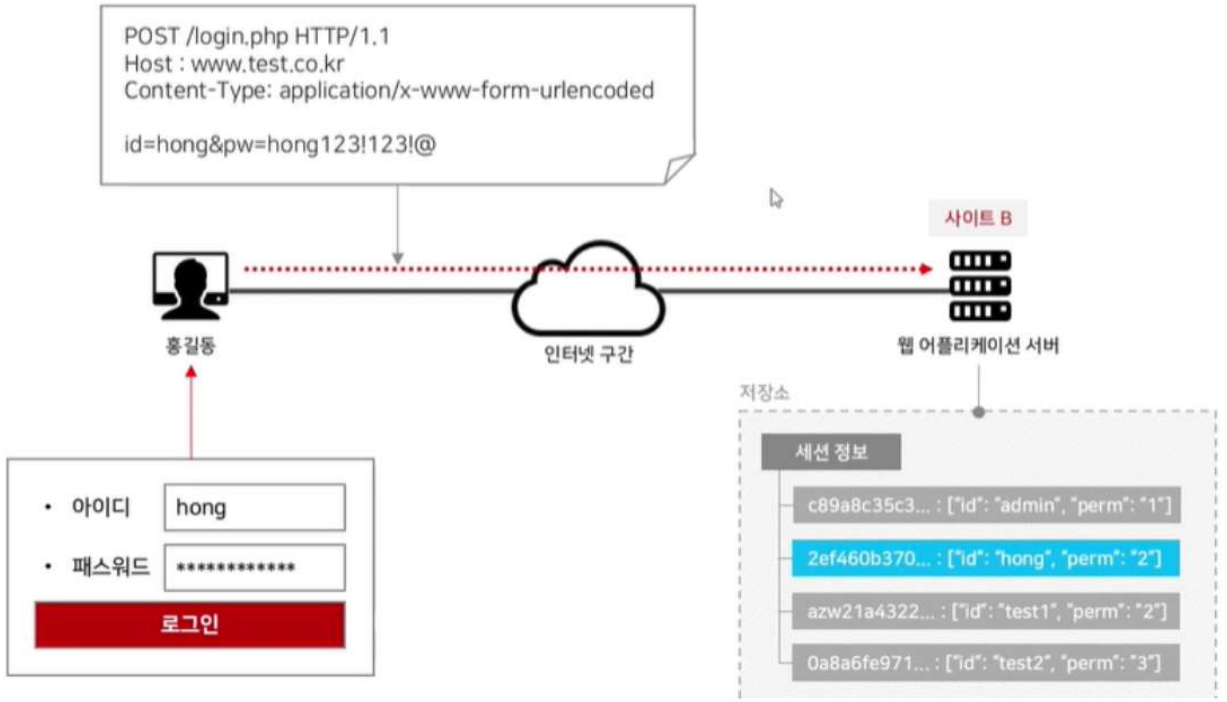
➔ 영화관에서 티켓을 보관용 부분만 찢어 건네주듯 세션 아이디를 사용자에게 전달

메모리에 아이디 사본을 어떤 사용자의 것인지 적어서 보관

REQUEST & RESPONSE

◆ 세션(Session)

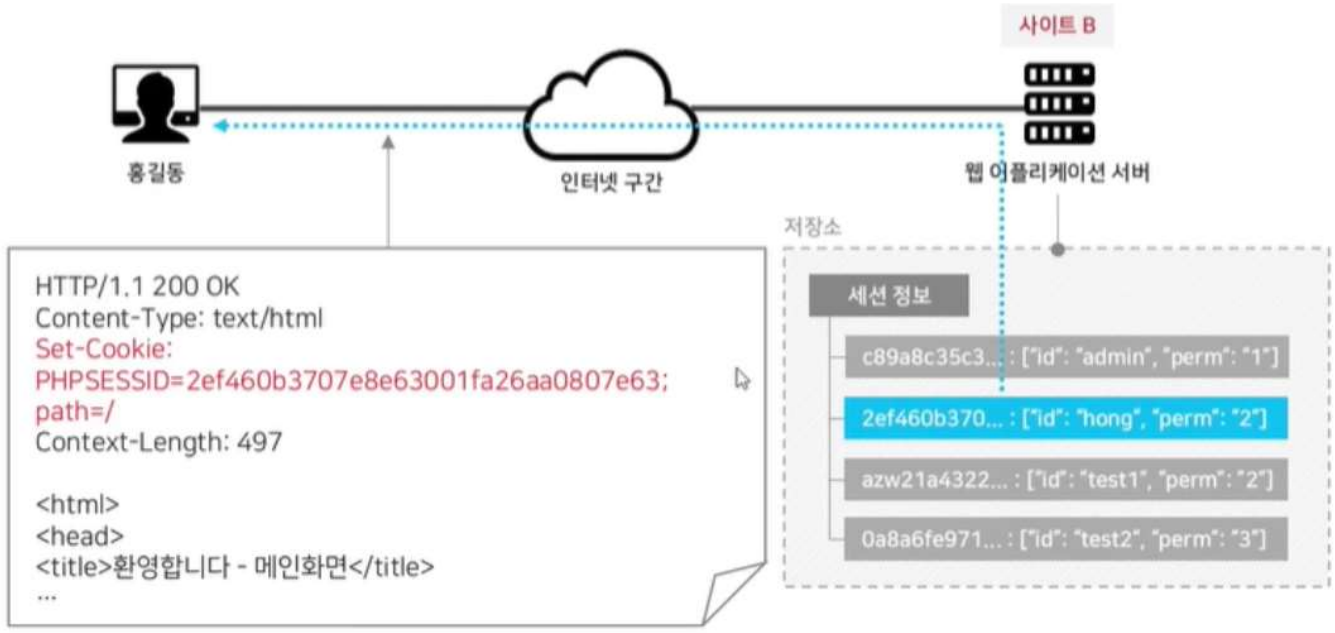
- 발급 과정



REQUEST & RESPONSE

◆ 세션

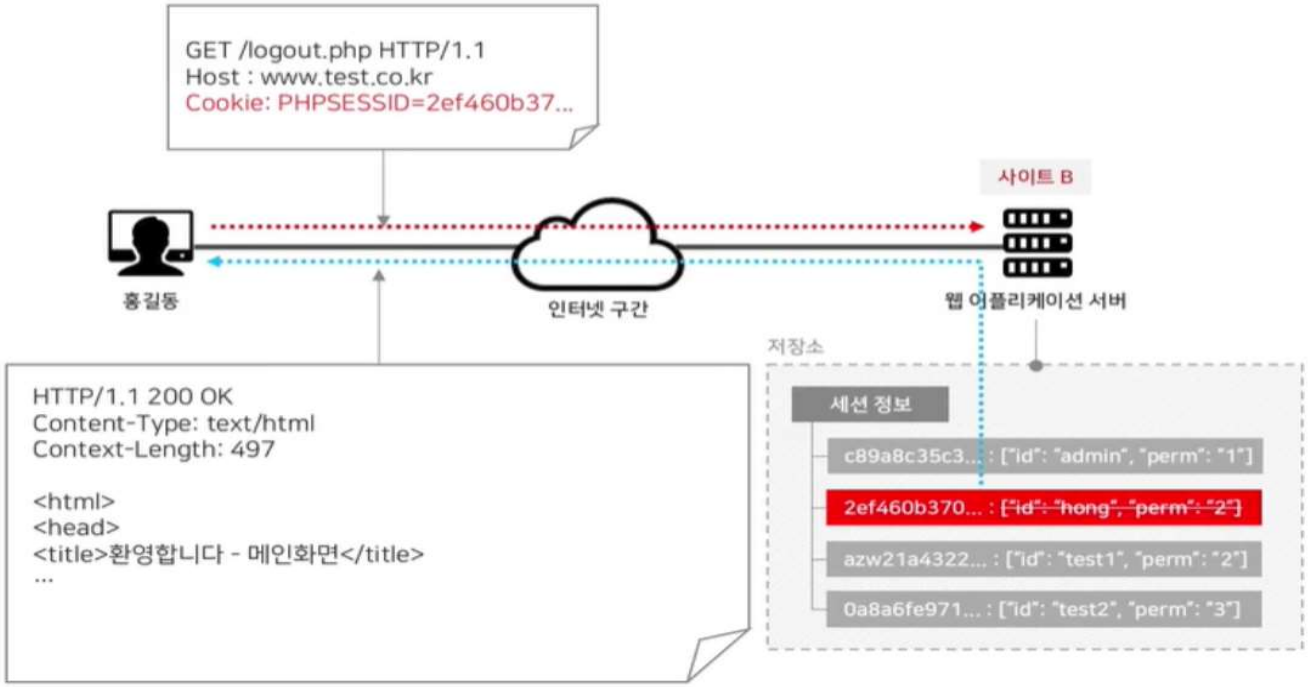
- **발급 과정**



REQUEST & RESPONSE

◆ 세션(Session)

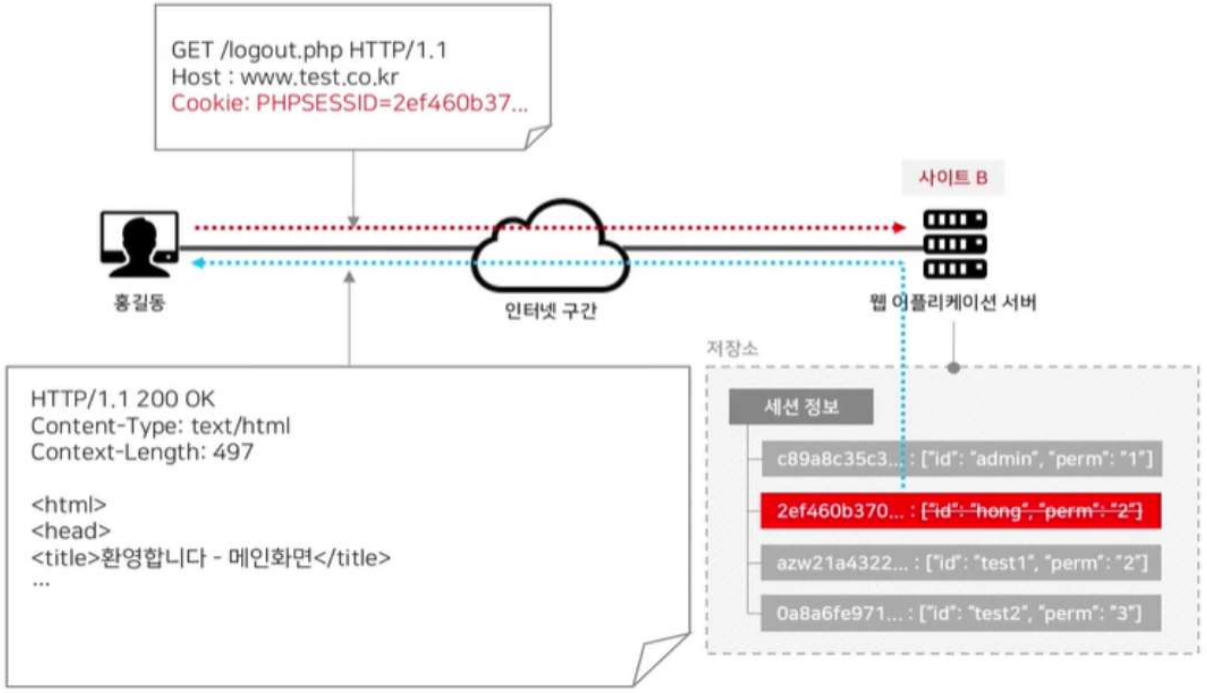
- 재방문



REQUEST & RESPONSE

◆ 세션(Session)

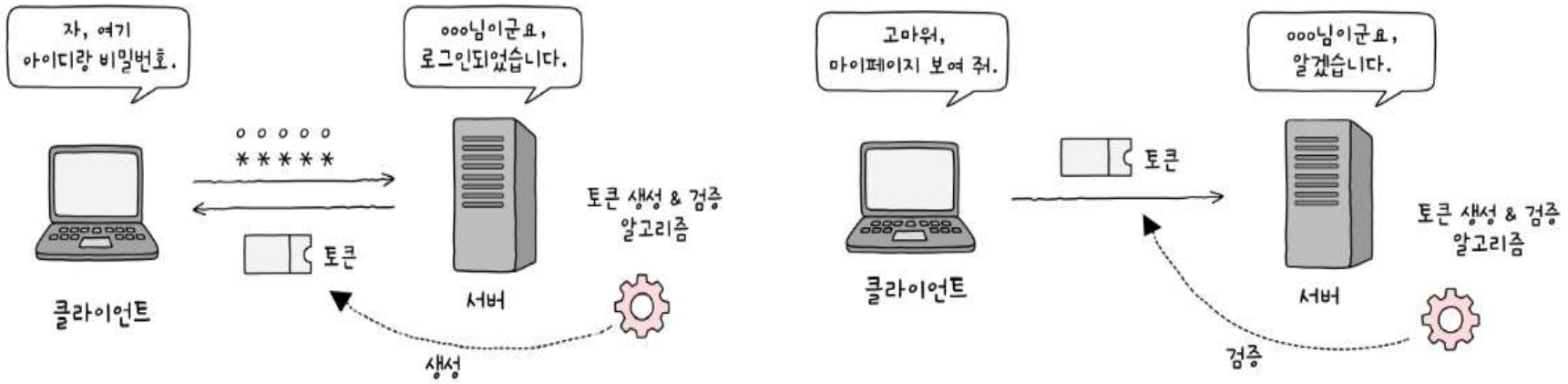
- 삭제



REQUEST & RESPONSE

◆ 토큰(Token)

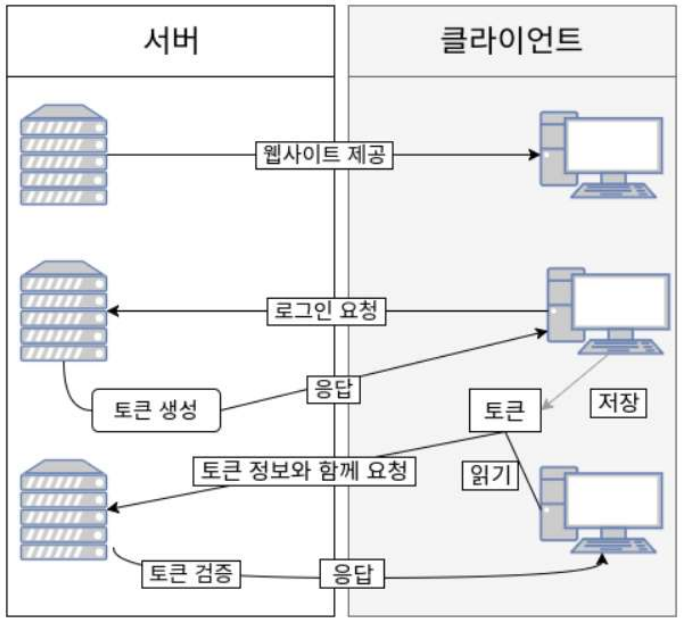
- 메모리 공간 많이 차지하는 **세션 방식대안으로 세션 아이디 대신 발급**
- **특수한 수학적 원리 적용** ➔ 서버만이 유효한 토큰 발행



REQUEST & RESPONSE

◆ 토큰(Token)

① 클라이언트 로그인
② 서버측에서 계정정보 검증
③ 서버측 검증 완료 후 Sigend 토큰 발급
④ 클라이언트 토큰 저장 후 서버에 요청 시 마다 토큰을 함께 전달
① 서버는 토큰 검증 후 요청에 응답



REQUEST & RESPONSE

◆ 메시지 플래싱(Message Flashing)

- **클라이언트에게 피드백 제공 시스템**
- 서버에서 클라이언트에게 **요청 처리 시 생긴 오류 및 처리사항을 HTML에 전달**해주는 기능
- 요청의 끝에 메시지를 기록하고 그 다음 요청에서만 그 메시지에 접근할 수 있게 함
- 보통은 플래싱을 처리하는 **레이아웃 템플릿과 결합되어 사용**

REQUEST & RESPONSE

◆ 메시지 플래싱(Message Flashing)

- **관련 함수**

  - flash() 메소드 : View에서 사용, 사용자에게 피드백 메시지 제공

    카테고리 : info, error, warning

  - get_flashed_messages() 메소드 : Template에서 사용, 메시지 수신

REQUEST & RESPONSE

◆ 메시지 플래싱(Message Flashing)

- **관련 함수**

  - flash() 메소드 : View에서 사용, 사용자에게 피드백 메시지 제공

    카테고리 : info, error, warning

  - get_flashed_messages() 메소드 : Template에서 사용, 메시지 수신

PART II

APP MODULARIZATION

APP MODULARIZATION

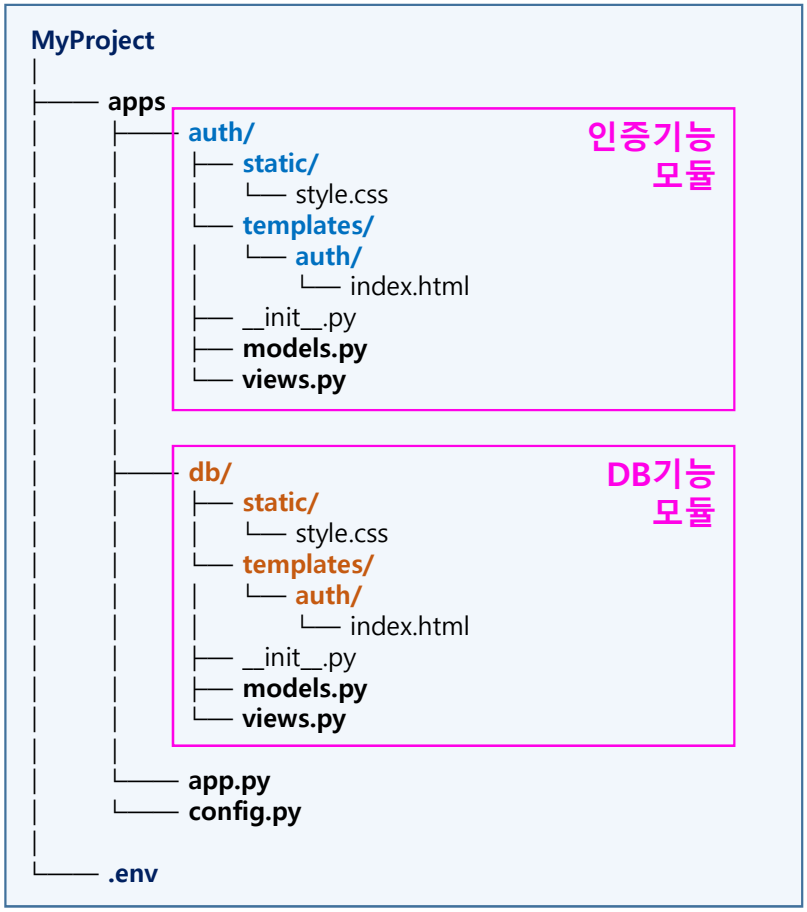
◆ Project 구조

- **모듈 단위기반 프로젝트 구조**

➔ MVT 모델 적용

➔ 유지보수 용이

➔ 표준화



APP MODULARIZATION

◆ Project 구조

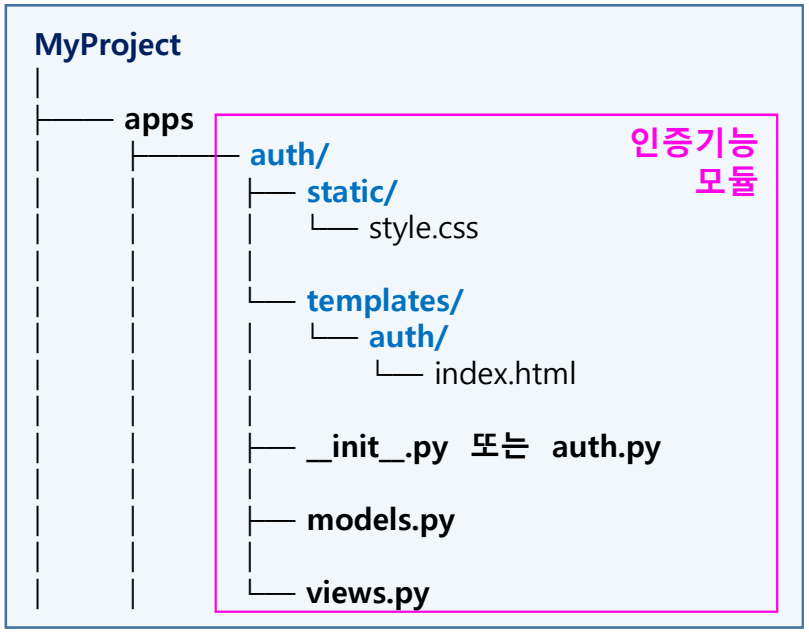
- **모듈 구조**

  ➔ static 폴더 : css, image file, javascript

  ➔ templates 폴더 : html file

  ➔ models.py 파일 : DB 관련 클래스 및 제어 기능

  ➔ views.py 파일 : 모듈 객체 생성 및 라우팅 기능



APP MODULARIZATION

◆ Project 구조

- **모듈 구조**

  ➔ config.py 파일 : 프로젝트의 환경을 설정

  ➔ app.py 파일 : Flask 객체 생성 및 모듈 객체 연결

  DB연동 및 초기화

  __init__.py 파일명으로 가능

  ➔ .env 파일 : Flask 환경설정

APP MODULARIZATION

◆ Project 구조

- **_ _init_ _.py**

① 패키지 관련된 설정이나 초기화 코드 작성

② 패키지 폴더 내에 위치하며 해당 폴더가 패키지로 인식되도록 함

➔ python 3.3버전부터 없어도 패키지로 인식함

➔ 하위 버전 호환을 위해서 파일 생성하는 것이 안전

① 패키지 폴더 내에 모듈 미리 인식하여 사용 간편하도록 설정

APP MODULARIZATION

◆ Application factory

▪ 순환 참조(circular import) 오류

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

```
# 전역변수
app = Flask(__name__)
```

프로젝트 규모가 커질수록 문제 발생 확률 높음

➔ 해결책 : 객체 반환 함수 즉, **application factory 함수 create_app() 사용**

APP MODULARIZATION

◆ 애플리케이션 팩토리(application factory)

- **create_app()**

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

```
from flask import Flask

def create_app():
    app = Flask(__name__)

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

    return app
```

APP MODULARIZATION

◆ Blueprint

- 규모가 큰 앱을 기능 단위로 분활
- 애플리케이션의 모듈화
- 유지보수 향상
- 간결한 코드 및 자원 관리 즉, **모듈별 라우팅 관리**

APP MODULARIZATION

◆ Blueprint

- 모듈 객체 생성

  ➔ View 역할 파일에 작성

```
# ----------------------------------------
# book기능 App ==>  Blueprint 인스턴스 생성
# ----------------------------------------
# 모듈 로딩 ------------------------------
from flask import Blueprint, render_template

# Blueprint 인스턴스 생성 ----------------
# URL => http://127.0.0.1:5000/book
blue_book = Blueprint('book',
                      __name__,
                      url_prefix='/book',
                      template_folder='templates')
```

APP MODULARIZATION

◆ Blueprint

- 모듈 객체 등록

  ➔ 메인 app.py 또는

  __init__.py에 작성

```python
# 애플리케이션 팩토리 함수 -------------
def create_app():
    # Flask Web Server App 생성
    app = Flask(__name__)

    # 기능단위 앱 Blueprint 객체 등록
    # 순환참조 예방위한 import
    from . import book_view, city_view
    app.register_blueprint(book_view.blue_book)
    app.register_blueprint(city_view.blue_city)

    return app
```

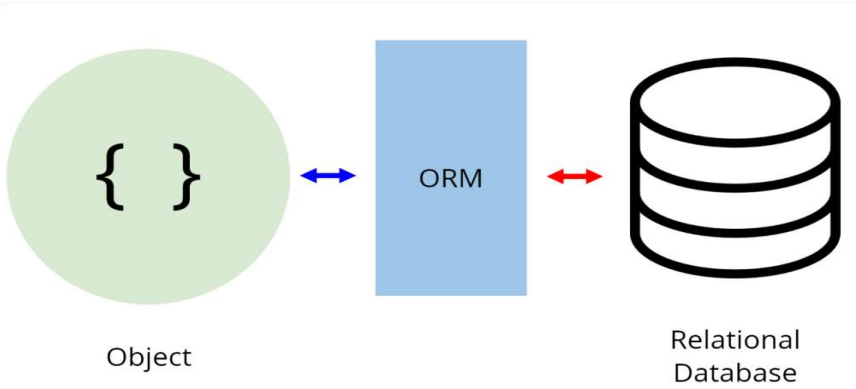
PART II

WITH ORM

WITH ORM

◆ ORM (Object Relational Mapping)

- **객체와 데이터베이스의 관계 매핑 도구**
- 객체와 관계형 데이터베이스 사이 중계자 역할
- MVC 패턴에서 모델(Model) 기술 도구
- 객체와 모델 사이의 관계 기술 도구
- **데이터베이스 종류 상관 없이 일관된 코드** 유지
- 프로그램 유지·보수 편리 및 오류 발생률 줄임

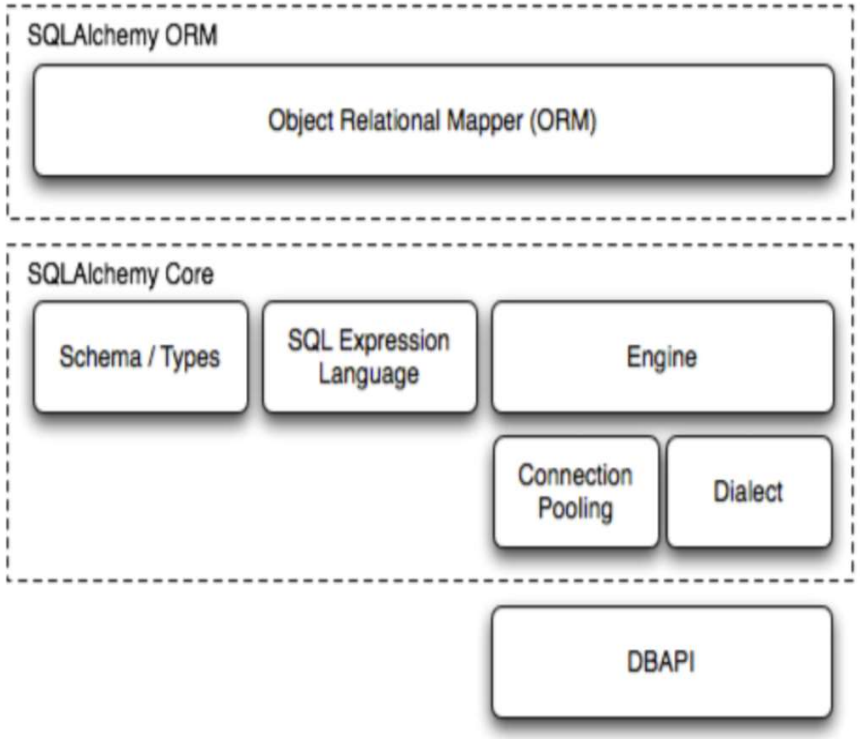


WITH ORM

◆ SQLAlchemy

- Python 대표 **ORM 라이브러리**
- 설치 : **pip install flask_migrate**

  pip install flask_sqlalchemy
- 구성
  - SQLAlchemy ORM
  - SQLAlchemy Core
  - DBAPI (e.g. psycopg2, PyMySQL etc ...)



WITH ORM

◆ SQLAlchemy

- **SQLAlchemy ORM Layer**

  - Core 기능들을 사용하기 쉽게 고수준의 인터페이스 제공해주는 layer
  - python class 통해 데이터베이스 테이블 정의, 관계 정의 및 관리
  - 제공되는 함수 통해서 DML, DDL 명령어 수행 가능

WITH ORM

◆ SQLAlchemy

▪ SQLAlchemy Core Layer

- 가장 핵심적인 역할 하는 계층
- 관계형데이터베이스와의 연결 관리
- 각각의 데이터베이스에 대한 데이터 유형에대한 타입핑 기능(Schema / Types)
- SQL문에 대한 프로그래밍 방식 구현 (SQL Expression Language)

WITH ORM

http://127.0.0.1:5000/db/user_reg

- 보여질 HTML 파일 : user_reg.html
- 회원정보 입력 후 저장

➔ DB에 저장
➔ action= url_for(endpoint) : user_insert
➔ method ='POST'

http://127.0.0.1:5000/db/user_insert

- request 객체에서 데이터 추출
- id, username, email

➔ DB에 저장

http://127.0.0.1:5000/db/user_list

- 보여질 HTML 파일 : user_list.html
- DB에 쿼리 전송
- 모든 회원정보 리스트 출력

WITH ORM

◆ SQLAlchemy

- 관련 설정 ➔ config.py 파일에 추가

```
# SQLALCHEMY DB 관련 설정 ------------------------------------------------------
import os

BASE_DIR = os.path.dirname(__file__)
DB_NAME_SQLITE = 'bpApp.db'

DB_SQLITE_URI = f'sqlite:///{os.path.join(BASE_DIR, DB_NAME_SQLITE)}'
DB_MYSQL_URI = 'mysql+pymysql://root:1234@localhost:3306/testdb'
DB_MARIA_URI = 'mariadb+mariadbconnector://root:root!@127.0.0.1:3308/db_ai'

# 사용할 DBMS 설정
SQLALCHEMY_DATABASE_URI = DB_SQLITE_URI
SQLALCHEMY_TRACK_MODIFICATIONS = False
```

이벤트 처리 옵션

WITH ORM

◆ SQLAlchemy

- **모델 파일 생성 ➔ models.py**

```
from bpApp.app import db

class Question(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    subject = db.Column(db.String(200), nullable=False)
    content = db.Column(db.Text(), nullable=False)
    create_date = db.Column(db.DateTime(), nullable=False)
```

WITH ORM

◆ SQLAlchemy

- 모델 파일 생성 ➔ models.py

```
class Answer(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    question_id = db.Column(db.Integer, db.ForeignKey('question.id', ondelete='CASCADE'))
    question = db.relationship('Question', backref=db.backref('answer_set'))
    content = db.Column(db.Text(), nullable=False)
    create_date = db.Column(db.DateTime(), nullable=False)
```

WITH ORM

◆ SQLAlchemy

- **ORM & Flask 연동**

  ➔ **app.py**

```python
# 모듈 로딩 ---------------------------
from flask import Flask
from flask_migrate import Migrate
from flask_sqlalchemy import SQLAlchemy

# 전역 변수 ---------------------------
db = SQLAlchemy()
migrate = Migrate()

# 애플리케이션 팩토리 함수 -------------
def create_app():
    # Flask Web Server App 생성
    app = Flask(__name__)

    # 설정 로딩
    app.config.from_pyfile("config.py")
    if app.debug :
        print(f'SQLALCHEMY_DATABASE_URI : {app.config["SQLALCHEMY_DATABASE_URI"]}')
```

WITH ORM

◆ SQLAlchemy

- **ORM & Flask 연동**

➔ **app.py 파일**

```
# ORM 연동
db.init_app(app)
migrate.init_app(app, db)

# 기능단위 앱 Blueprint 객체 등록
# 순환참조 예방위한 import
from . import book_view, city_view, question_view, models
app.register_blueprint(book_view.blue_book)
app.register_blueprint(city_view.blue_city)
app.register_blueprint(question_view.blue_question)

return app
```

WITH ORM

◆ SQLAlchemy

- 데이터베이스 관련 명령어
  - > flask db

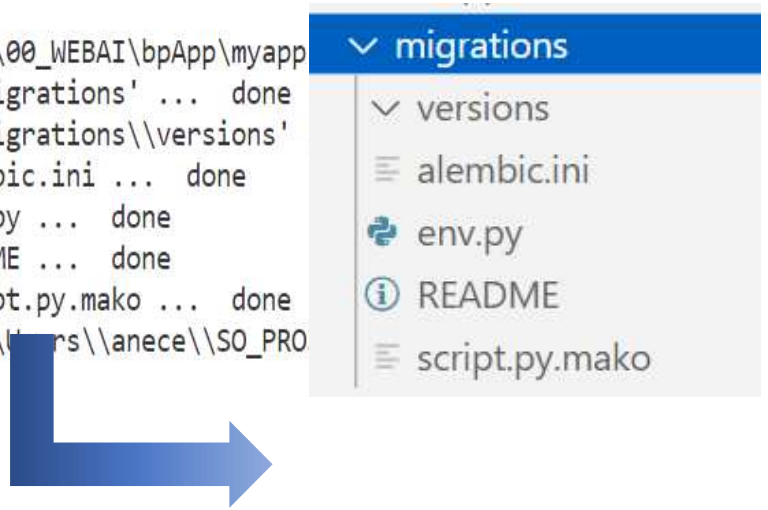
```
Commands:
  branches        Show current branch points
  check           Check if there are any new operations to migrate
  current         Display the current revision for each database.
  downgrade       Revert to a previous version
  edit            Edit a revision file
  heads           Show current available heads in the script directory
  history         List changeset scripts in chronological order.
  init            Creates a new migration repository.
  list-templates  List available templates.
  merge           Merge two revisions together, creating a new revision file
  migrate         Autogenerate a new revision file (Alias for 'revision...
  revision        Create a new revision file.
  show            Show the revision denoted by the given symbol.
  stamp           'stamp' the revision table with the given revision;...
  upgrade         Upgrade to a later version
```

WITH ORM

◆ SQLAlchemy

- 데이터베이스 초기화 / 최초 한번만 실행 ! ➔ flask db init

```
(AI_WEB) C:\Users\anece\SO_PROJECT\00_WEBAI>flask db init
SQLALCHEMY_DATABASE_URI : sqlite:///C:\Users\anece\SO_PROJECT\00_WEBAI\bpApp\myapp
Creating directory 'C:\\Users\\anece\\SO_PROJECT\\00_WEBAI\\migrations' ...  done
Creating directory 'C:\\Users\\anece\\SO_PROJECT\\00_WEBAI\\migrations\\versions'
Generating C:\Users\anece\SO_PROJECT\00_WEBAI\migrations\alembic.ini ...  done
Generating C:\Users\anece\SO_PROJECT\00_WEBAI\migrations\env.py ...  done
Generating C:\Users\anece\SO_PROJECT\00_WEBAI\migrations\README ...  done
Generating C:\Users\anece\SO_PROJECT\00_WEBAI\migrations\script.py.mako ...  done
Please edit configuration/connection/logging settings in 'C:\\Users\\anece\\SO_PRO
\00_WEBAI\\migrations\\alembic.ini' before proceeding.
```



WITH ORM

◆ SQLAlchemy

- **데이터베이스 관련 명령어**

명령어	설명
flask db **migrate**	**모델 새로 생성 또는 변경할** 때 사용 실행하면 작업파일이 생성
flask db **upgrade**	모델 변경 내용을 **실제 데이터베이스에 적용**할 때 사용 생성된 작업파일을 실행하여 데이터베이스 변경

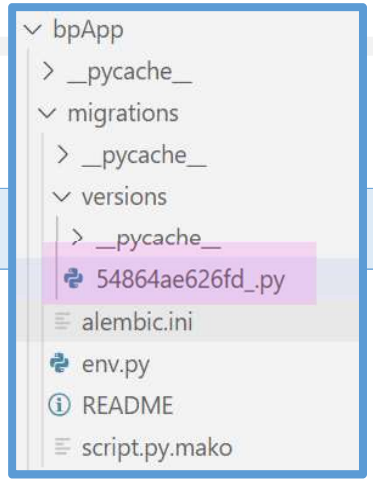
WITH ORM

◆ SQLAlchemy

- 테이블 생성

> flask db migrate

```
(AI_WEB) C:\Users\anece\SO_PROJECT\00_WEBAI\bpApp>flask db migrate
INFO  [alembic.runtime.migration] Context impl SQLiteImpl.
INFO  [alembic.runtime.migration] Will assume non-transactional DDL.
INFO  [alembic.autogenerate.compare] Detected added table 'question'
INFO  [alembic.autogenerate.compare] Detected added table 'answer'
Generating C:\Users\anece\SO_PROJECT\00_WEBAI\bpApp\migrations\versions\54864ae626fd_.py ...  done
```



WITH ORM

◆ SQLAlchemy

- 실제 데이터베이스 적용

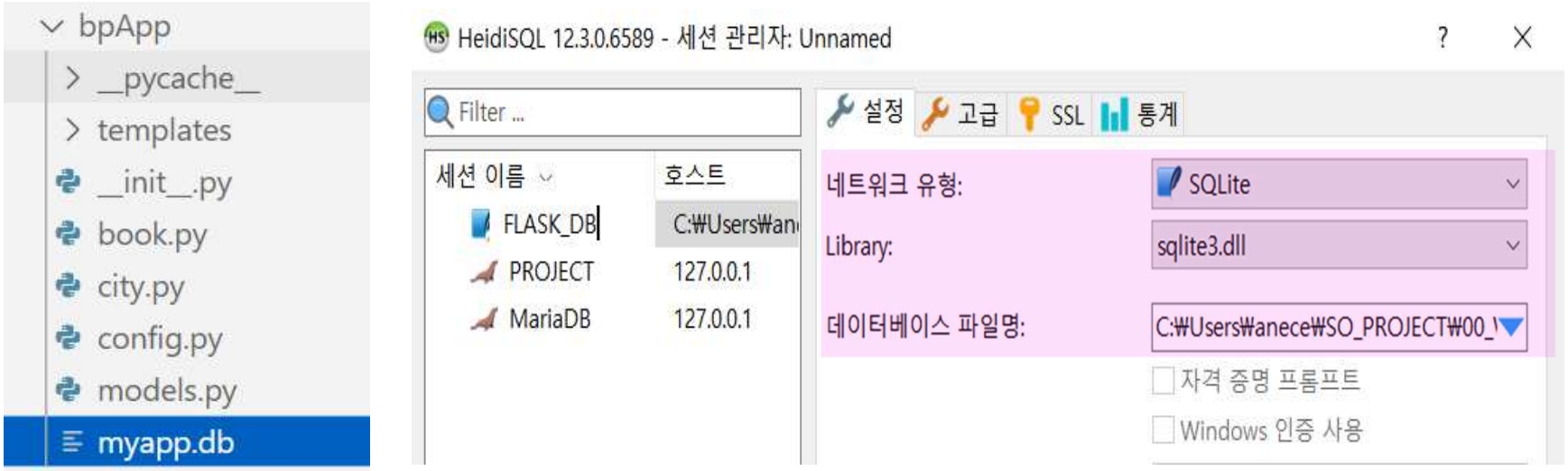
> flask db update

```
(AI_WEB) C:\Users\anece\SO_PROJECT\00_WEBAI\bpApp>flask db upgrade
INFO  [alembic.runtime.migration] Context impl SQLiteImpl.
INFO  [alembic.runtime.migration] Will assume non-transactional DDL.
INFO  [alembic.runtime.migration] Running upgrade  -> 54864ae626fd, empty message
```

WITH ORM

◆ SQLAlchemy

- 데이터베이스 및 테이블 생성 파일 확인



PART II

WITH WFT

WITH WFT

◆ Flask-WTF

- Flask 프레임 워크의 Form 검증 모듈
- 폼 생성 및 json 데이터 상호 작용 위한 검증 도구
- Python 웹 개발을 위한 유연한 양식 검증 및 렌더링 라이브러리
- 웹 프레임워크 및 템플릿 엔진과 함께 작동
- 데이터 검증, CSRF 보호, 국제화(I18N), 파일 업로드 등 지원

WITH WFT

◆ Flask-WTF

- WTF관련 라이브러리들

WTF 관련 패키지	특징
Flask-WTF	• Flask 프레임워크와 통합 • 요청에서 자동으로 데이터 로드, 사용자 선택 로케일 기반 번역 • 전체 애플리케이션 CSRF 등 제공
WTForms-Alchemy	• 확장된 필드 및 유효성 검사기 세트 포함 • 양식을 생성하기 위한 풍부한 지원 제공
WTForms-SQLAlchemy	• SQLAlchemy 모델에서 ORM 지원 필드, 양식 생성 제공
WTForms-Django	• Django 모델에서 ORM 지원 필드와 양식 생성 제공 • Django의 I18N 지원과의 통합 제공

WITH WFT

◆ Flask-WTF

- WTF관련 라이브러리들

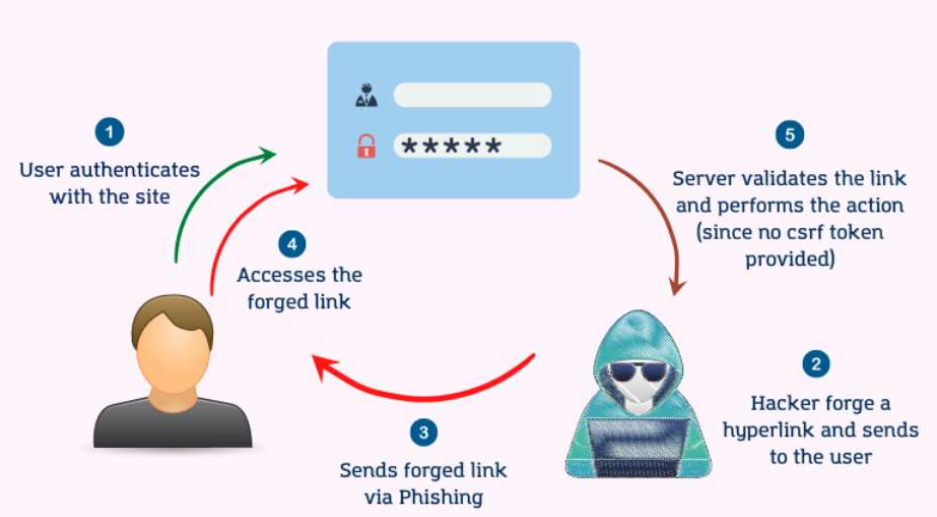
WTF 관련 패키지	특징
Flask-WTF	• Flask 프레임워크와 통합 • 요청에서 자동으로 데이터 로드, 사용자 선택 로케일 기반 번역 • 전체 애플리케이션 CSRF 등 제공
WTForms-Alchemy	• 확장된 필드 및 유효성 검사기 세트 포함 • 양식을 생성하기 위한 풍부한 지원 제공
WTForms-SQLAlchemy	• SQLAlchemy 모델에서 ORM 지원 필드, 양식 생성 제공
WTForms-Django	• Django 모델에서 ORM 지원 필드와 양식 생성 제공 • Django의 I18N 지원과의 통합 제공

WITH WFT

◆ Flask-WTF

▪ CSRF(Cross-site Request Forgery)

웹 어플리케이션 취약점 중 하나
인터넷 사용자가 자신의 의지와는 무관하게
공격자가 의도한 행위(수정, 삭제, 등록 등)
를 특정 웹사이트에 요청하게 만드는 공격



WITH WFT

◆ Flask-WTF

▪ WTF 적용

- **입력 받을 데이터의 형식** : 입력 받을 시 원하는 form 지정 가능
  ➔ 문자열(StringField), 비밀번호(PasswordField), 체크값(BooleanField)…
- **데이터 검증** : Validators 사용하여 입력 받은 값에 대한 유효성 검사 가능
  ➔ 길이(Length), 필수 데이터(DataRequired), 동일 값인지 확인(EqualTo)…
- **CSRF(Cross-site request forgery) 보호**

WITH WFT

◆ Flask-WTF

- 데이터 유효성 검사

```
from flask_wtf import FlaskForm
from wtforms import PasswordField, StringField, SubmitField
from wtforms.validators import DataRequired, Email, Length

class SignUpForm(FlaskForm):
    username = StringField( "사용자명",
                            validators=[
                                DataRequired("사용자명은 필수입니다."),
                                Length(1, 30, "30문자 이내로 입력해 주세요."),
                            ],
    )
```

WITH WFT

◆ Flask-WTF

- **데이터 유효성 검사**

```
email = StringField( "메일 주소",
                    validators=[
                        DataRequired("메일 주소는 필수입니다."),
                        Email("메일 주소의 형식으로 입력해 주세요."),
                    ],
)
password = PasswordField("비밀번호", validators=[DataRequired("비밀번호는 필수입니다.")])
submit = SubmitField("신규 등록")
```

WITH WFT

◆ Flask-WTF

- 데이터 유효성 검사

```
@auth.route("/signup", methods=["GET", "POST"])
def signup():
    # SignUpForm을 인스턴스화한다
    form = SignUpForm()

    if form.validate_on_submit():
        user = User(
            username=form.username.data,
            email=form.email.data,
            password=form.password.data,
        )
```

WITH WFT

◆ Flask-WTF

- **데이터 유효성 검사**

```
# 메일 주소 중복 체크를 한다
if user.is_duplicate_email():
    flash("지정한 이메일 주소는 이미 등록되어 있습니다.")
    return redirect(url_for("auth.signup"))

# 사용자 정보를 등록한다
db.session.add(user)
db.session.commit()

# 사용자 정보를 세션에 저장한다
login_user(user)

# GET 파라미터에 next 키가 존재하고, 값이 없는 경우는
# 사용자의 일람 페이지로 리다이렉트한다
next_ = request.args.get("next")
if next_ is None or not next_.startswith("/"):
    next_ = url_for("detector.index")
return redirect(next_)
```

PART II

BOOTSTRAP

FLASK BOOTSTRAP

◆ 부트스트랩(Bootstrap)?

▪ 정의 및 용도

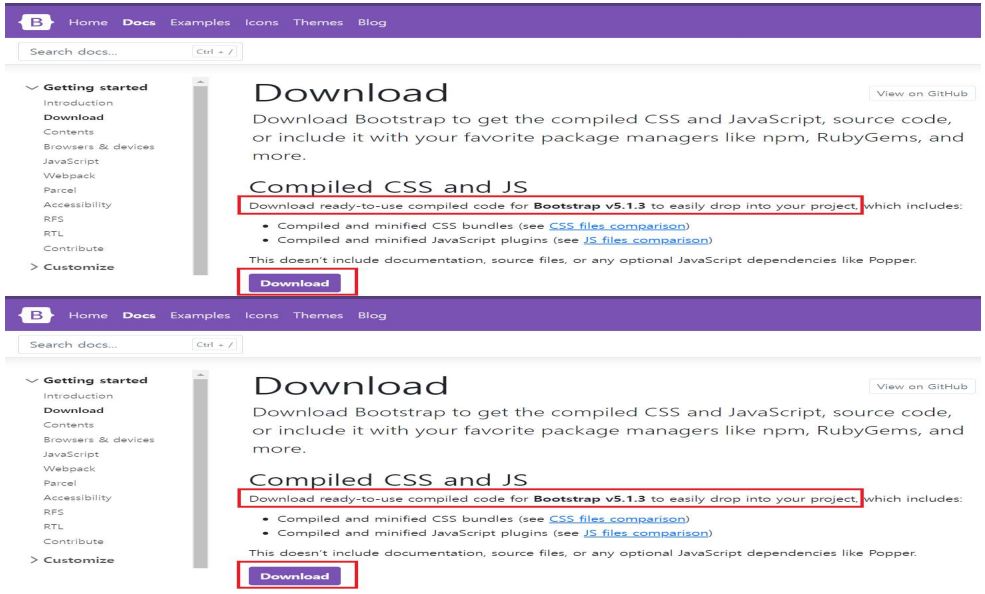
- **프론트엔드 개발 빠르고 쉽게 할 수 있는 프레임워크**
- HTML과 CSS 기반의 **템플릿 양식**, **버튼, 네비게이션 및 기타 페이지 구성하는 요소**
- 자바스크립트를 선택적으로 확장 할 수 있고 **상업적 이용 가능**

FLASK BOOTSTRAP

◆ 부트스트랩(Bootstrap)

- 다운로드

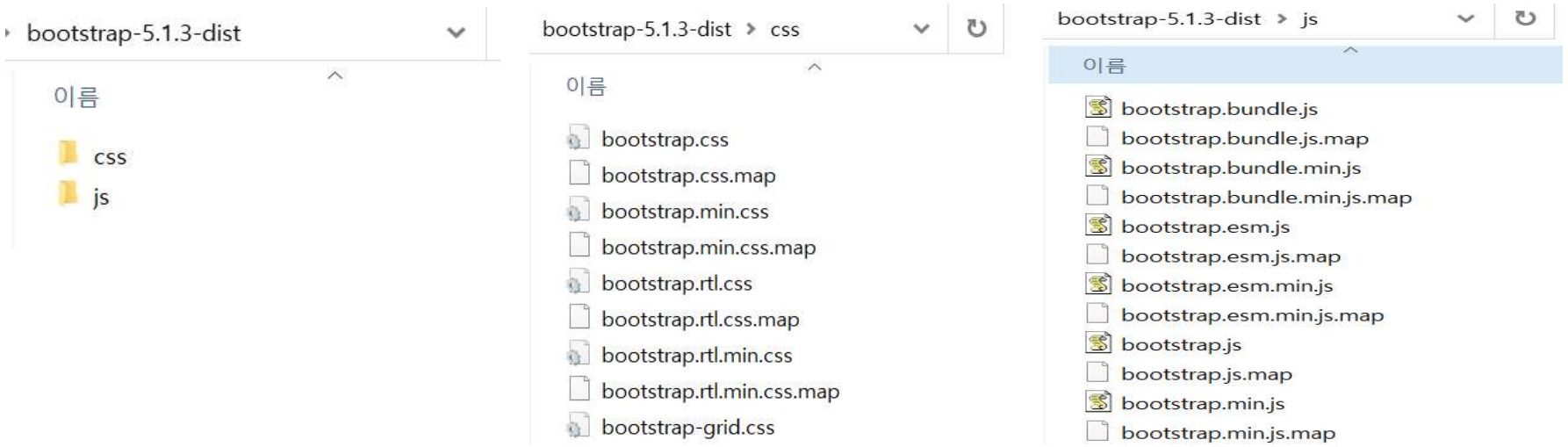
  - https://getbootstrap.com/docs/5.1/getting-started/download/



FLASK BOOTSTRAP

◆ 부트스트랩(Bootstrap)

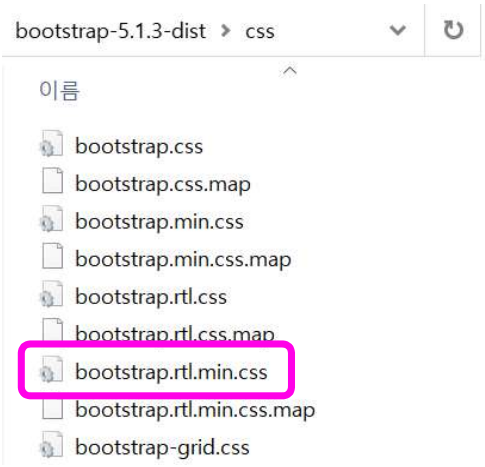
- **bootstrap-5.1.3-dist**



FLASK BOOTSTRAP

◆ 부트스트랩(Bootstrap)

- **HTML 파일 적용**




```
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UTF-8" />
    <title>detector</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/bootstrap.min.css') }}" />
    <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}" />
</head>
```

FLASK BOOTSTRAP

◆ 부트스트랩(Bootstrap)

▪ 주요 클래스

부트스트랩 클래스	설명
card, card-body, card-text	부트스트랩 Card 컴포넌트
badge	부트스트랩 Badge 컴포넌트
form-control, form-label	부트스트랩 Form 컴포넌트
border-bottom	아래방향 테두리 선
my-3	상하 마진값 3
py-2	상하 패딩값 2
p-2	상하좌우 패딩값 2
d-flex justify-content-end	컴포넌트의 우측 정렬
bg-light	연회색 배경
text-dark	검은색 글씨
text-start	좌측 정렬
btn btn-primary	부트스트랩 버튼 컴포넌트