SEEDS SEASON2

Seeds
FIRST STEP
-3 week-

Seeds운영진 **김민성**

INDEX

01 Flask

- Flask는 파이썬 기반 웹 프레임워크중 하나로 자바가 아닌 파이썬으로도 웹 서버를 만들수 있다. 그리고 Flask는 Micro Web Framework라고도 불린다.

- 자바 기반 웹 프레임워크로는 대표적으로 Spring이 있듯이, 파이썬 기반 웹 프레임워크로는 대표적으로 Flask와 Django가 있다.

01 Flask의 특징

- Flask는 직접 원하는 라이브러리 및 패키지를 선택하고 추가해가면서 개발할 수 있고, 별도의 관리자 기능 등이 필수적이지 않은 경우 활용할 수 있는 등 기본 제공 사항이 없어도 자유도가 높다.

- Flask는 타 Framework 기능에 종속되지 않고 자체 Framework를 구현할 수 있으며, Core 기능을 심플하게 유지하여 확장성을 보장하는 형태의 Framework로 Microservice 구현 및 REST API 개발에 특화되어 있다.

- Flask는 DB ORM(Object-Relational Mapping) 구조가 존재하지 않아 특정 DB에 종속되어 사용하지 않아도 된다.

01 Flask의 특징

- REST API를 빠르게 개발할 수 있도록한 확장 패키지인 Flask-RESTful을 활용하여 기존 ORM, Libraries를 그대로 활용하도록 설계된 경량 구조이다.

- Flask는 Django대비 커뮤니티 크기가 낮다고 알려져 있고, 전반적인 인기도 역시 Django보다 낮은 편이지만 개발 자유도에 있어서 압도적으로 우수하다. 또한 가볍고 빠른 설치가 가능한 Framework이므로 빠른 Poc, Prototyping 작업이 용이하다.

- Flask는 Machine Learning, Deep Learning 작업시 원하는 언어와 클라우드 환경에 맞게 구현이 가능하다.

01 Flask 제공기능

- 라우팅 : URL과 함수를 연결하는 라우팅을 지원한다. 이를 통해 웹 어플리케이션의 각 페이지에 대한 URL을 정의하고, 해당 URL로 요청이 들어올 때 실행될 함수를 지정할 수 있다.

- 템플릿 엔진: Flask는 Jinja2 템플릿 엔진을 내장하고 있다. 이를 이용하여 동적인 웹 페이지를 생성할 수 있습니다. Jinja2는 매우 강력한 기능을 가지고 있으며, 템플릿 상속, 조건문, 반복문 등을 지원한다.

- 데이터 베이스 지원: Flask는 다양한 데이터베이스에 대한 ORM(Object Relational Mapping)을 지원한다. SQLAlchemy를 사용하여 데이터베이스와 상호작용하는 코드를 작성할 수 있다.

01 Flask 제공기능

- 세션 관리 : Flask는 세션(Session)을 지원한다. 세션을 이용하여 사용자가 로그인한 상태인지 등을 관리할 수 있습니다.

- HTTP 요청 처리 : Flask는 HTTP요청을 처리하는데 필요한 기능을 제공한다. 예를 들어, 요청 파라미터, 요청 헤더 , 쿠키 등을 처리 할 수 있다.

- 보안 : Flask는 보안에 대한 다양한 기능을 제공한다. 예를 들어 CSRF(Cross-Site Request Forgery) 공격 방어 기능을 제공한다.

01 Flask의 장점

- 경량성 : Flask Framework는 필요한 기능만을 제공하여 빠르게 개발 하는 것을 초점으로 잡고 나왔기 때문에 파일의 용량은 비교적 적다.

- 유연성 : 레고 블록을 작게 조립하는 것 처럼 Flask도 작은 기능 하나로도 빠르게 시작할 수 있기 때문에 다양한 기능들을 붙여서 유연하게 작동시킬수 있게 된다. 필요한 부품을 체크하는 것과 동일하게 프로젝트에서 필요한 기능들을 체크해서 붙여낼 수 있는 장점이 존재한다.

- 초반에는 쉬운 학습 곡선 : 레고 부품으로 간단한 프로그램을 만들 때는 비교적 쉽게 만들 수 있는데 Flask도 그런 작은 부품 단위들을 잘 정리한 것과 같이 쉽게 웹 애플리케이션을 구축할수 있다.

01 Flask의 장점

- 강력한 디버깅 기능 : 내장된 디버깅 기능을 활용하여 버그를 빠르게 제거할 수 있다는 장점이 존재한다.

- 단위 테스트 지원 : 디버깅과 함께 내장된 유닛 테스트 지원으로 테스트 케이스를 쉽게 작성해서 TDD 운용을 보다 쉽고 빠르게 진행할 수 있다.

  *TDD(Test-Driven-Development) : 매우 짧은 개발 사이클을 반복하는 소프트웨어 개발 프로세스 중 하나이며 "테스트 주도 개발" 이라고도 함.

01 Flask의 단점

- Django에 비해 자유도는 높으나, 제공해주는 기능이 Django에 비해 덜 하다.

- 보안 : 경량형을 지향하는 Flask는 보안을 위해서는 자체 라이브러리나 모듈을 만들어서 사용해야 보완성을 지킬 수 있다.

- 대규모 프로젝트 : Flask가 MSA(MicroServiceArchitecture)에 가까운 프레임워크라고 하지만 대규모 프로젝트로 넘어가면 Spring의 인지도나 다른 Framework들의 인지도를 따라가기 애매한 위치에 존재하기 때문에 큰 규모의 프로젝트에서는 조금 버거울 수 있다.

01 Flask의 단점

- 개발자의 스킬 : 기본적으로 제공되는 라이브러리나 Framework들이 내장된 것들이 아니기 때문에 하나의 기능을 추가해도 의존성 체크부터 시작해서 다양한 에러 사항들을 모두 봐야하는 어려움이 있다.

- 성능 : Micro Framework로 빠른 성능을 요구하는 것이지만 최근에는 Node.js의 인지도에 도전하는 Fastapi Framework가 존재하기 때문에 속도적인 측면만 봤을 때 Flask가 빠른 생산성과 속도를 가졌다고 판단하기 어렵다.

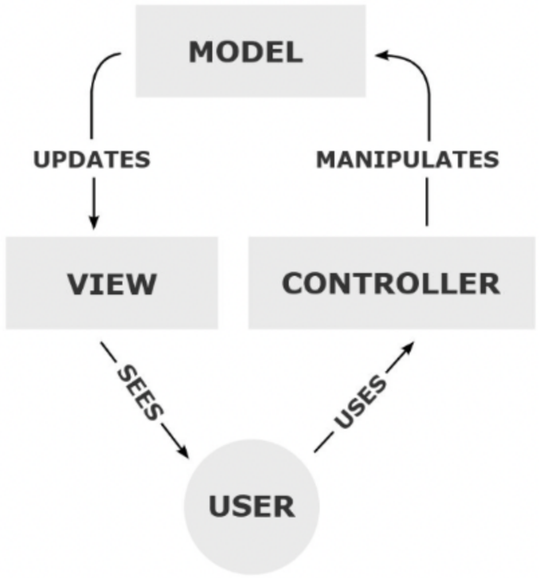
01 Flask 사용시기

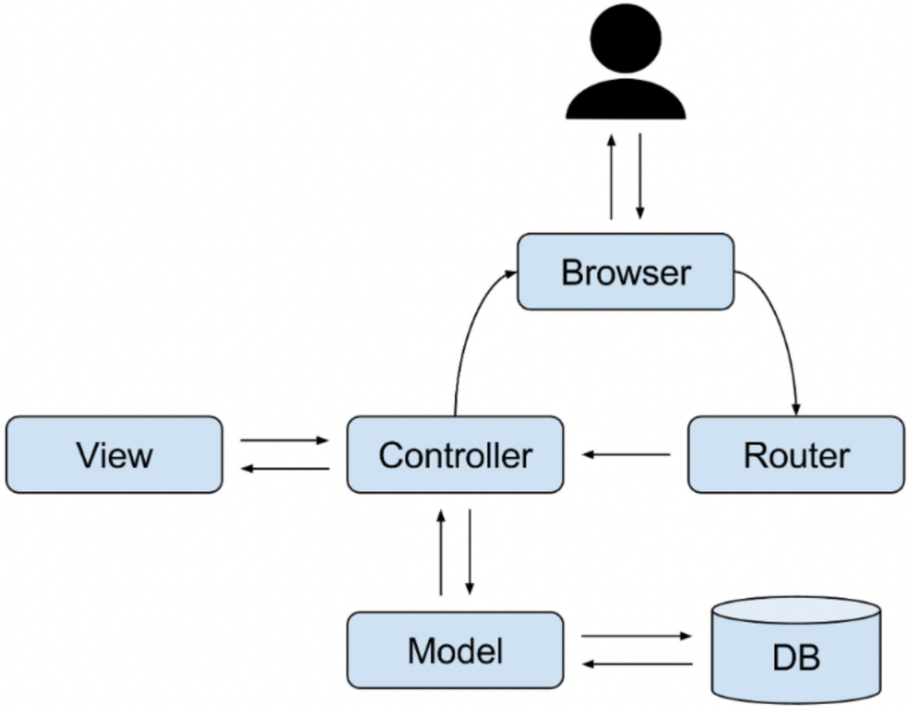
- 파이썬 기반으로 웹 개발을 하고 싶을 때 사용할 수 있다. 파이썬이 Back-End쪽을 담당하고 HTML,CSS,JavaScript 등이 Front-End 쪽을 담당하게 하는 식으로 구현하여 웹 프로젝트를 수행할 수 있다.

- 단독서버로 Flask를 사용할 수도 있고 다른 언어 기반의 서버와도 연동이 가능하다.
  - ex) Spring으로 서비스를 하다가 특정 이미지나 영상은 파이썬 코드로 따로 처리가 가능하다는 뜻.

02 MVC 패턴

- Flask도 MVC 패턴을 따른다. 웬만한 web Framework들은 모두 MVC패턴을 따른다고 봐도 무방하다.





02 MVC패턴

- 사용자가 웹 애플리케이션을 이용하기 위해 URL 요청을 하면, 해당 요청은 Controller에 들어오게 된다.
- Controller은 해당 요청을 담당하는 메소드로 안내하고, 요청에 포함된 정보를 가지고 Model에 접근할 수 있다.
- Model은 데이터베이스 같은 비즈니스 로직을 처리하는 일을 수행하고 Controller로 다시 돌아온다.
- Controller의 처리가 끝났다면 그 결과를 사용자들이 화면을 통해 볼 수 있도록 View로 이동시켜 준다.

03 Flask 사용

- 웹서버 구동코드

```
from flask import Flask
app = Flask(__name__) #Flask 클래스의 인스턴스를 생성
@app.route('/')
def index():
    return 'Flask에 오신걸 환영합니다!'
if __name__ == '__main__':
    app.debug=True #디버그 모드 활성화
    app.run() #Flask 애플리케이션을 로컬 서버에서 실행기본적으로 로컬 호스트의 5000번 포트에서 실행됨.
```

터미널 : python 파일이름.py or flask run

03 Flask 사용

- Flask run vs python app.py 차이
  - flask run : 이 명령은 Flask의 커맨드 라인 인터페이스를 사용하여 애플리케이션을 실행.
    - 이 명령을 사용하면 Flask가 제공하는 개발용 서버를 사용하여 애플리케이션을 실행하고, 서버는 디버깅과 자동 리로드 기능을 제공한다. 또한 FLASK_APP 환경 변수를 통해 애플리케이션을 지정가능
  - python app.py : 이 명령은 python 인터프리터를 사용하여 app.py 스크립트를 직접 실행.
    - 이 명령은 애플리케이션의 시작점 (if __name__ == "__main__" :) 에 정의된 서버를 사용하여 애플리케이션을 실행한다. 디버깅과 자동 리로드기능의 여부는 app.run()함수에 전달된 인자에 따라 달라진다.

- 두 명령의 주요 차이점은 애플리케이션을 실행하는 방식과 환경 설정의 차이. 일반적으로 개발과정에서는 flask run을 사용하여 개발용 서버의 기능을 활용하고, 배포 시에는 WSGI 서버를 활용하여 app.py를 실행하는 것이 일반적

03 Flask 사용

- DB생성 함수

```
from flask import Flask, render_template, request, redirect, url_for
import sqlite3
def init_db():
    with sqlite3.connect('database.db') as db: #database.db 생성
        c = db.cursor() #데이터베이스 연결을 통해 커서를 생성
                        #커서는 SQL 명령을 실행하고 결과를 얻는데 사용
        c.execute('''
            CREATE TABLE IF NOT EXISTS users (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                username TEXT NOT NULL UNIQUE,
            password TEXT NOT NULL
            );
        ''') # 커서를 사용해 SQL 명령을 실행함
```

render_template: HTML 템플릿 파일을 렌더링하는 함수
redirect : 페이지를 다른 URL로 리다이렉트하는 함수
url_for : Flask 웹 애플리케이션에서 사용하는 URL을 동적으로 생성하는 함수

03 Flask 사용

- DB에 유저를 추가하는 함수

```
def add_user(username, password): #데이터베이스에 유저 추가 함수
    with sqlite3.connect('database.db') as db:
        c = db.cursor()
        c.execute("INSERT INTO users (username, password) VALUES (?, ?)", (username,
password))
        db.commit() #SQLite 데이터베이스에서 트랜잭션을 완료하는데 사용되는 메소드입니다.
```

*트랜잭션이란?

- 트랜잭션(Transaction 이하 트랜잭션)이란, 데이터베이스의 상태를 변화시키기 해서 수행하는 작업의 단위를 뜻한다.

DB BrowserforSQLite : https://sqlitebrowser.org/dl/ <- db파일 내부를 볼수있는 프로그램

04 Flask 사용

- 라우팅 함수 정의

```
@app.route('/')
def index():
    return '<h1>Flask에 오신걸 환영합니다.</h1>'
```

04 Flask 사용

- 로그인 페이지 라우팅 함수 정의

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        with sqlite3.connect('database.db') as db:
            c = db.cursor()
            c.execute('SELECT * FROM users WHERE username = ? AND password = ?', (username, password))
            user = c.fetchone() #fetchone() 함수는 SQL 쿼리의 결과에서 다음 행을 가져오는 메서드
            if user is not None:
                return redirect(url_for('login_success')) # 로그인 성공
            else:
                return '로그인 실패' # 로그인 실패
    return render_template('login.html') # GET 요청이면 로그인 폼을 보여줌
```

04 Flask 사용

- login 템플릿 정의

```
<!DOCTYPE html>
<title>로그인</title>
<form action="/login" method="post"> #post 메소드 정의
  <label for="username">사용자 이름</label>
  <input type="text" id="username" name="username" required /> required:
  <label for="password">비밀번호</label>                          해당 필드가 반드시 채워져야 함.
  <input type="password" id="password" name="password" required />
  <input type="submit" value="로그인" />
</form>
```

04 Flask 사용

- 로그인 성공페이지 라우팅 함수 정의

```
@app.route('/login-success')
def login_success():
    return '로그인 성공'
```

과제

로그아웃 기능을 구현해 오세요.