

웹 서버 만들기

노기섭 교수

(kafa46@hongik.ac.kr)

■ Flask 소개

- 특징
- 설치

■ Hello world 서버 만들기

- 웹 애플리케이션 코딩 및 실행
- 라우팅 추가
- MTV 모델 적용

Flask 소개

Flask 소개

■ Flask



- [플라스크](#)는 파이썬으로 작성된 경량 웹 프레임워크
- 간결하고 유연한 웹 애플리케이션 개발을 가능
- Flask는 2010년 아르민 론허([Armin Ronacher](#))가 개발한 마이크로 프레임워크로 시작
- 이후 오픈 소스 프로젝트로 발전하여 많은 개발자들 사이에서 인기를 얻음

Flask 특징

■ 간결함과 유연성

- 최소한의 핵심 기능만 포함
- 개발자가 필요에 따라 기능을 선택적으로 추가

■ 단순한 애플리케이션 구조

- 하나의 파이썬 파일로 작성 가능
- @app.route 데코레이터를 사용하여 편리하게 경로 관리 가능

■ 강력한 확장 기능

- 확장 기능을 통해 강력한 기능 구현 가능
 - 데이터베이스 연동, 폼 처리, 사용자 인증, RESTful API 구축 등 다양한 기능 구축
- 유연성과 신속한 구현 가능

■ 커뮤니티와 지원

- 활발한 커뮤니티와 풍부한 문서 제공
- 다양한 튜토리얼, 예제코드
- [Stack Overflow](https://stackoverflow.com/)와 같은 개발자 커뮤니티에서 질문과 답변을 쉽게 찾을 수 있음

Flask 설치

■ 가상환경 설치

```
python -m venv myenv
```

■ 가상환경 활성화

- Windows

```
myenv\Scripts\activate
```

- macOS 및 Linux

```
source myenv/bin/activate
```

■ Flask 설치

```
pip install Flask
```

Hello world 서버 만들기

프로젝트 폴더를 생성 → app.py 코딩

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
    return "Hello, World!"

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

터미널에서 다음 명령어 실행

`python app.py`

로컬 호스트(127.0.0.1:5000)로 접속

- Flask 클래스: Flask 애플리케이션의 인스턴스를 생성
- @app.route('/'): 특정 URL 경로에 대한 요청을 처리하는 뷰 함수 정의
- def home(): 요청을 처리하는 함수
 - '/' 경로로 요청이 들어오면 "Hello, World!"를 반환
- app.run(debug=True): 애플리케이션을 실행
 - debug=True는 디버그 모드를 활성화

여기서 잠깐!

■ Flask 서버

- 간단한 웹 애플리케이션 개발을 위한 훌륭한 도구이며
- Flask에는 개발을 쉽게 할 수 있도록 기본적으로 제공되는 내장 개발 서버 제공
 - 내장 개발 서버를 사용하면 빠르게 시작할 수 있음
- 내장 서버의 성능과 보안상의 한계 존재
 - uWSGI 또는 Gunicorn 같은 고성능 WSGI (Web Server Gateway Interface) 서버를 사용해야 함

■ WSGI 서버 운용

- 추가로 많은 부분들을 학습해야 함
- 윈도우 시스템에서는 잘 작동되지 않는 경우가 많음
- 우리는 Flask에 내장된 서버를 이용하여 실습을 진행

라우팅 추가

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
    return "Hello, World!"

# 라우팅 추가
@app.route('/about_me')
def about_me():
    return "안녕하세요. 저는 웹 개발자입니다. Flask를 사용하여 웹 애플리케이션을 개발하고 있습니다."

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

터미널에서 다음 명령어 실행

```
python app.py
```

로컬 호스트(127.0.0.1:5000)로 접속

MTV 패턴 적용

■ MTV 패턴

- Model: 데이터베이스 관련 로직
- Template: HTML 파일 생성 및 서비스 로직
- View: 서버 라우팅 및 비즈니스 로직
- 웹 애플리케이션에 따라 각자 개별 패턴을 가지고 있음
 - Java를 사용하는 스프링부트의 경우 "Model-View-Controller (MVC)" 패턴을 사용

■ Model

- 데이터베이스와 상호작용하는 부분
- 데이터베이스의 구조와 데이터를 정의
- Flask에서는 SQLAlchemy를 사용하여 모델을 정의 (객체지향 구현)
- SQLAlchemy
 - Python 언어용 SQL 도구이자 Object-Relational Mapping(ORM) 라이브러리
 - 관계형 데이터베이스와의 상호작용을 단순화
 - 데이터베이스 작업을 더 효율적이고 직관적으로 만들 수 있음

SQLAlchemy 모델 정의

- SQLAlchemy를 설치: `pip install Flask-SQLAlchemy`
- SQLAlchemy 이용하여 모델 정의

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///site.db'
db = SQLAlchemy(app)

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True, nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    def __repr__(self):
        return f'<User {self.username}>'

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

템플릿(HTML) 코딩

```
<!-- templates/index.html -->  
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <title>Home</title>  
</head>  
<body>  
    <h1>Hello, {{ name }}!</h1>  
</body>  
</html>
```

View (서버 라우팅) 코딩

```
from flask import Flask

app = Flask(__name__)

from flask import render_template

@app.route('/')
def home():
    return render_template('index.html', name='World')

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

MTV 전체 코드

```
from flask import Flask, render_template
from flask_sqlalchemy import SQLAlchemy
import os
```

```
app = Flask(__name__) # 플라스크 객체 생성
```

```
# 데이터베이스 파일 경로 설정
```

```
BASE_DIR = os.path.abspath(os.path.dirname(__file__)) # 현재 경로 추출
```

```
DB_PATH = os.path.join(BASE_DIR, 'hello.db')
```

```
app.config['SQLALCHEMY_DATABASE_URI'] = f'sqlite:/// {DB_PATH}'
```

```
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
```

```
# 데이터베이스 객체 생성
```

```
db = SQLAlchemy(app)
```

```
class User(db.Model):
```

```
    id = db.Column(db.Integer, primary_key=True)
```

```
    username = db.Column(db.String(80), unique=True, nullable=False)
```

```
    email = db.Column(db.String(120), unique=True, nullable=False)
```

```
    def __repr__(self):
```

```
        return f'<User {self.username}>'
```

```
def home():
```

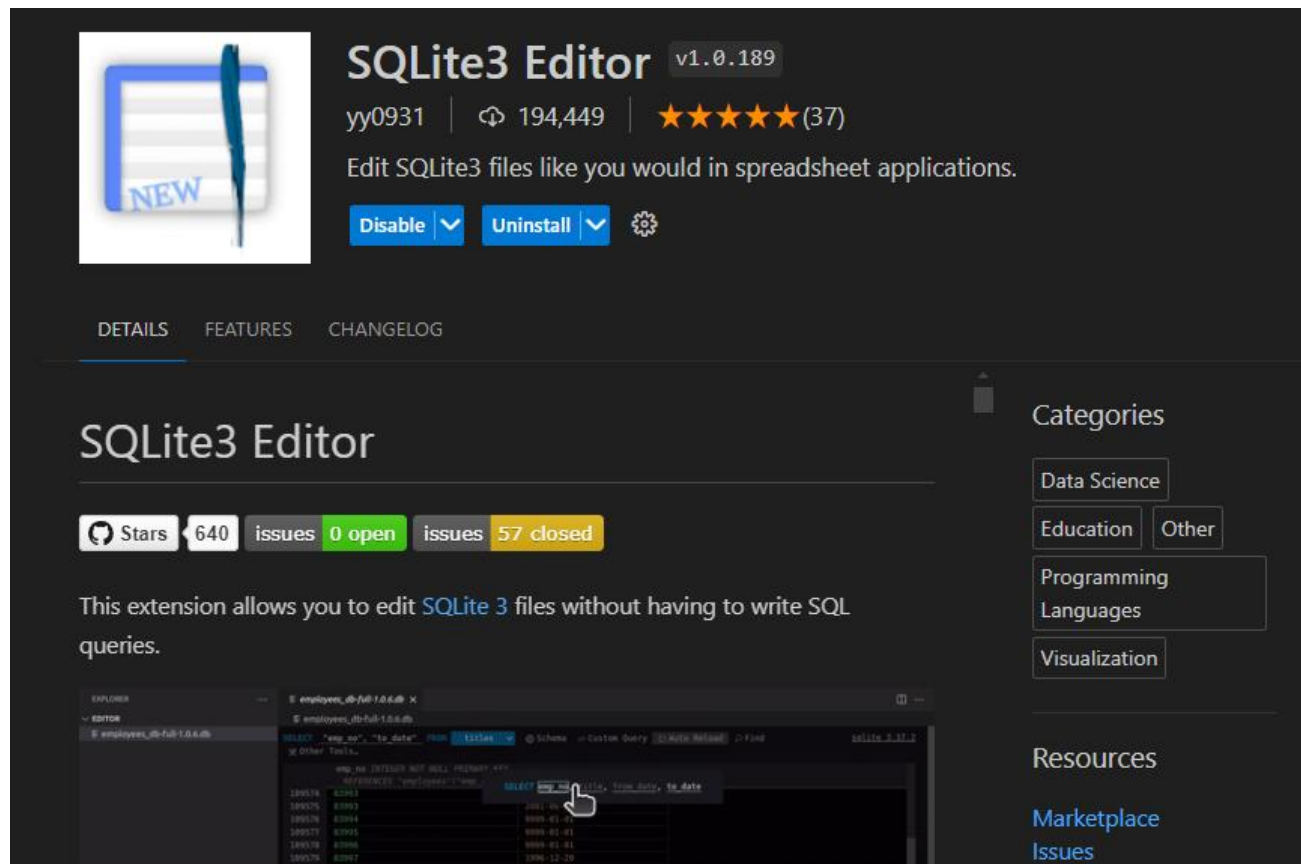
```
    return render_template(
        'index.html', name='World'
    )
```

```
if __name__ == "__main__":
```

```
    with app.app_context():
        db.create_all()
        app.run(debug=True)
```

데이터베이스 확인용 익스텐션

- SQLite 데이터베이스 파일은 VS Code 확장 팩(Extension Pack) 중에서 **SQLite3 Editor**를 설치하면 편리하게 DB 내용을 확인하고 수정 가능



The screenshot shows the VS Code Marketplace page for the 'SQLite3 Editor' extension. The extension is by 'yy0931' and has a version of 'v1.0.189'. It has 194,449 downloads and a 5-star rating from 37 reviews. The description states: 'Edit SQLite3 files like you would in spreadsheet applications.' Below the description are buttons for 'Disable', 'Uninstall', and a settings gear icon. The page also features tabs for 'DETAILS', 'FEATURES', and 'CHANGELOG'. On the right, there are category tags: 'Data Science', 'Education', 'Other', 'Programming Languages', and 'Visualization'. At the bottom right, there are links for 'Marketplace' and 'Issues'. The main content area shows a preview of the extension's interface, which includes a file explorer on the left, a central editor area displaying a SQLite3 database file, and a right-hand pane showing a table of data. The table has columns for 'emp_no', 'emp_name', 'to_date', and 'to_date'. The data rows show employee information, such as '100001', '100001', '100001', '100001', '100001', '100001', '100001', '100001', '100001', '100001'.

SQLite3 Editor v1.0.189
yy0931 | 194,449 | ★★★★★ (37)
Edit SQLite3 files like you would in spreadsheet applications.

Disable | Uninstall | ⚙️

DETAILS | FEATURES | CHANGELOG

SQLite3 Editor

Stars 640 | issues 0 open | issues 57 closed

This extension allows you to edit **SQLite 3** files without having to write SQL queries.

Categories

- Data Science
- Education
- Other
- Programming Languages
- Visualization

Resources

- Marketplace
- Issues

EXPLORER | EDITOR | OUTPUT | DEBUG CONSOLE | DEBUG CONSOLE

employees_db-full-1.0.0.db

SELECT "emp_no", "to_date" FROM employees

emp_no	emp_name	to_date	to_date
100001	100001	1000-01-01	1000-01-01
100002	100002	1000-01-01	1000-01-01
100003	100003	1000-01-01	1000-01-01
100004	100004	1000-01-01	1000-01-01
100005	100005	1000-01-01	1000-01-01
100006	100006	1000-01-01	1000-01-01
100007	100007	1000-01-01	1000-01-01
100008	100008	1000-01-01	1000-01-01
100009	100009	1000-01-01	1000-01-01
100010	100010	1000-01-01	1000-01-01

다음 시간에는...

- 지금까지 작성한 코드를 기반으로
- MTV 패턴 확장하는 실습



수고하셨습니다 ..^^..