웹 페이지에서 Hello, world! 출력하기

목차

작성 날짜 : 2024.08.28. 수요일

작성이 : 김연우 / 김민서 / 국봉호

1. 과제 개요

본 과제의 목표는 Python 의 Flask 프레임워크를 사용하여 간단한 웹 페이지를 생성하고, 웹 페이지에 "Hello, World!"라는 메시지를 출력하는 것입니다. Flask 는 Python 을 사용하여 웹 애플리케이션을 쉽게 개발할 수 있는 마이크로 프레임워크입니다.

Flask 는 이름처럼 가볍고 간단하게 웹 애플리케이션을 시작할 수 있는 프레임워크로, 처음 배우는 사람도 쉽게 웹 서버를 구축할 수 있는 것이 장점입니다. Flask 는 기본적으로 웹 서버를 구축하는데 필수적인 핵심 기능만을 제공하며, 사용자는 이를 기반으로 확장하고자 하는 기능을 자유롭게 추가할 수 있습니다.

Flask 의 주요 특징은 다음과 같습니다:

① 간단한 구조: Flask 는 최소한의 코드로 웹 애플리케이션을 구현할 수 있습니다. Python 에 대한 깊은 이해와 복잡한 설정 없이도 기본적인 웹 서버를 구축할 수 있습니다.
② 확장 가능성: Flask 는 기본적으로 필요한 기능만 제공하므로, 불필요한 부분이 없습니다. 그러나 필요한 기능을 추가로 확장할 수 있는 모듈과 라이브러리가 다양하게 제공됩니다.
③ 빠른 프로토타이핑: Flask 는 작은 규모의 웹 애플리케이션을 빠르게 개발하고 테스트할 수 있도록 설계되었습니다. 몇 줄의 코드만으로도 서버가 실행되기 때문에 초기 프로토타입 개발에 매우 유용합니다.
④ 개발자 친화적인 환경: Flask 는 개발 단계에서의 효율성을 높이기 위해 디버깅 도구와 코드 변경 사항을 실시간으로 반영하는 기능을 지원합니다. 이 덕분에 개발자는 매번 서버를 수동으로 재시작 하지 않아도 됩니다.

이러한 Flask 의 특징을 바탕으로, 간단한 웹 애플리케이션을 직접 작성하고, 이를 통해 웹 개발의 기초를 배우는 것을 목표로 합니다. 특히 Flask 의 기본적인 기능을 사용해 서버를 구축하고 브라우저에 메시지를 출력하는 과정을 통해 웹 애플리케이션 개발의 기초적인 흐름을 이해할 수 있습니다.

2. Flask 및 Python 설치

① Python 설치 과정

먼저, Python 이 시스템에 설치되어 있는지 확인하는 과정이 필요합니다. 이를 위해 터미널을 실행하고 아래의 명령어를 입력했습니다.

```
python3 --version
```

이 명령어를 통해 Python 3.x 버전이 설치되어 있는지 확인할 수 있습니다. 만약 Python 3.x 가 설치되어 있지 않다면, Python 공식 웹사이트에서 Python 3.x 버전을 다운로드하여 설치할 수 있습니다.

이후, 개발 환경을 위해 Visual Studio Code(VSCode)를 사용하였습니다. VSCode 는 다양한 언어의 코드를 작성하고 실행할 수 있는 강력한 코드 에디터로, Flask 애플리케이션을 개발하기 위해 Python Extension 을 추가로 설치하였습니다.
이를 통해 VSCode 에서 Python 코드를 손쉽게 작성하고 실행할 수 있는 환경이 마련되었습니다.

② Flask 설치 방법

Python 이 정상적으로 설치된 것을 확인한 후, Flask 프레임워크를 설치하기 위해 pip3 패키지 관리 도구를 사용하였습니다. 아래 명령어를 통해 Flask 를 설치할 수 있습니다.

```
pip3 install flask
```

이 명령어는 Flask 와 그 외에 필요한 의존성 패키지들을 자동으로 설치해줍니다. 설치가 완료된 후, Flask 가 정상적으로 설치되었는지 확인하기 위해 아래 명령어를 사용하여 설치된 Flask 의 정보를 확인할 수 있습니다.

```
pip3 show flask
```

이 명령어는 Flask 의 설치 경로와 버전 경로를 출력합니다. 이를 통해 Flask 가 시스템에 정상적으로 설치되었는지 확인할 수 있습니다.

추가적으로, Flask 애플리케이션을 실행하기 위해서는 Flask 가 어떤 파일을 실행할지 지정하는 환경 변수가 필요합니다. 이를 설정하기 위해 터미널에서 다음 명령어를 사용하여 'FLASK_APP' 환경 변수를 설정하였습니다.

```
export FLASK_APP=app.py
```

이 환경 변수를 설정함으로써 Flask 가 실행할 애플리케이션 파일을 명확하게 지정할 수 있습니다.

3. Flask 애플리케이션 개발

① 애플리케이션 코드 설명

Flask 애플리케이션은 Python 파일에 간단한 코드로 작성됩니다. 먼저 `app.py`라는 파일을 생성하고, 아래와 같은 코드를 작성합니다:

app.py

```
from flask import Flask

app = Flask(__name__)

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

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

from flask import Flask
Flask 클래스가 포함된 flask 모듈을 임포트합니다. Flask 는 애플리케이션을 만들기 위한 핵심 클래스입니다. 이 모듈을 임포트함으로써 Flask 애플리케이션을 정의하고, 실행하는 데 필요한 모든 기능을 사용할 수 있게 됩니다.

app = Flask(__name__)
Flask 클래스의 인스턴스인 app 을 생성합니다. Flask(__name__)는 애플리케이션을 정의할 때 반드시 필요한 부분입니다. __name__은 현재 실행 중인 모듈의 이름을 나타내며, Flask 는 이 이름을 기반으로 애플리케이션이 어디에서 실행되고 있는지를 파악합니다. 이렇게 하면 Flask 는 애플리케이션이 실행 중인 경로에 맞게 필요한 리소스(정적 파일 등)를 찾을 수 있습니다.

@app.route('/')
이 줄은 Flask 의 라우팅 기능을 설정하는 부분입니다. @app.route('/') 데코레이터는 Flask 에게 어떤 URL 경로로 접속했을 때, 그에 해당하는 함수를 실행할지 알려줍니다. 여기서는 '/'로 설정되어 있기 때문에, 애플리케이션의 루트 URL(예: http://127.0.0.1:5000/)로 접속하면 다음에 정의된 hello_world() 함수가 실행됩니다.

def hello_world():

return 'Hello, World!'
hello_world 함수는 루트 URL에 접속했을 때 실행될 함수입니다. 이 함수는 간단히 "Hello, World!"라는 문자열을 반환합니다. Flask는 반환된 문자열을 HTTP 응답으로 클라이언트(웹 브라우저)에 전달합니다. 그 결과, 사용자가 해당 URL로 접속하면 브라우저 화면에 "Hello, World!"라는 메시지가 표시됩니다.

if __name__ == '__main__': app.run(debug=True)
이 블록은 Python 스크립트가 직접 실행될 때만 Flask 개발 서버를 시작하도록 합니다. if __name__ == '__main__':는 Python에서 일반적으로 사용하는 구문으로, 해당 파일이 다른 모듈에 의해 호출되지 않고 직접 실행되었을 때 실행되도록 보장합니다. app.run(debug=True)는 Flask 애플리케이션을 실행하고, 개발 모드에서 실행되도록 합니다. 개발 모드에서는 애플리케이션이 실행되는 동안 오류 메시지와 디버깅 정보를 쉽게 볼 수 있으며, 코드가 변경될 때마다 서버를 자동으로 다시 시작하는 기능을 제공합니다. debug=True는 디버깅 모드를 활성화합니다. 이 모드를 활성화하면 코드를 수정한 후 서버를 재시작할 필요 없이 자동으로 재시작되며, 오류 발생 시 브라우저에 자세한 디버깅 정보를 표시해줍니다.

② 디렉토리 구조 및 파일 위치 설명

Flask 애플리케이션을 실행하기 위해서는 Flask가 애플리케이션 파일(ex: app.py)를 올바르게 찾아 실행할 수 있어야 합니다. Flask는 기본적으로 'FLASK_APP'이라는 환경 변수를 사용해 실행할 파일을 지정합니다. 따라서 이 파일이 프로젝트의 루트 디렉토리에 있어야 Flask가 이를 쉽게 인식할 수 있습니다.

프로젝트 디렉토리
VScode ㄴ9roomthon_Training | ㄴapp.py

Flask는 특정 파일에서 애플리케이션 인스턴스(즉, 'app = Flask(__name__)')을 찾아야 하며, 이를 통해 애플리케이션을 실행하게 됩니다. 파일이 다른 곳에 위치해 있거나 잘못된 경로로 설정되면 Flask는 이를 인식하지 못해 애플리케이션을 실행할 수 없습니다.

③ 번외 내용: 권장되는 프로젝트 구조

Flaks 프로젝트는 확장성과 유지보수를 고려해 디렉토리 구조를 구성하는 것이 중요합니다. Flask의 기본적인 파일 위치는 다음과 같이 간단히 구성할 수 있습니다.

권장되는 프로젝트 구조
my_flask_app ├ static ├ templates └ app.py

- app.py: Flask 애플리케이션의 핵심 파일로, 서버가 실행되고 라우팅 및 응답 처리를 담당하는 코드가 여기에 작성됩니다.
- static/: 정적 파일을 저장하는 디렉토리입니다. 이미지, CSS, JavaScript 파일을 여기에 저장하며, Flask 가 이 디렉토리에서 정적 파일을 서빙(제공)합니다.
- templates/: HTML 템플릿 파일을 저장하는 디렉토리입니다. Flask 는 템플릿 엔진인 Jinja2 를 사용해 이 디렉토리 내의 HTML 파일을 동적으로 렌더링할 수 있습니다.

이 구조는 Flask 애플리케이션의 가독성과 유지보수성을 높이는 데 도움을 줍니다. 특히, 프로젝트가 커지면 파일을 체계적으로 관리하지 않을 경우 혼란이 발생할 수 있습니다. Flask 는 이러한 디렉토리 구조를 따름으로써 코드의 역할을 명확히 구분하고, 프로젝트 확장성을 고려할 수 있도록 돕습니다.

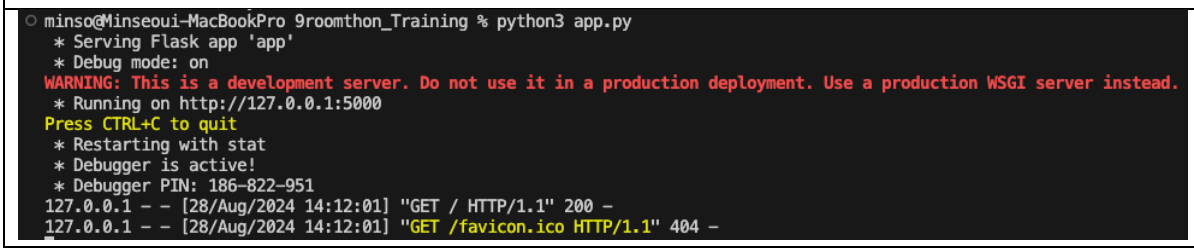
예시로, app.py 는 서버 로직(라우팅, 처리 등)을 담당하고, static/ 디렉토리는 이미지나 CSS 와 같은 정적 자산을 관리하며, templates/는 사용자가 보게 될 HTML 페이지를 관리합니다. 이러한 구조는 Flask 애플리케이션을 개발할 때 일반적으로 권장되는 패턴이며, 애플리케이션이 커질 때 파일을 관리하는 데 매우 유용합니다.

4. 애플리케이션 실행

실행 과정 설명

Flask 애플리케이션을 실행하기 위해서는 터미널에서 `python3 app.py` 명령어를 입력하면 됩니다. 실행 후, Flask 개발 서버가 구동되며 아래와 같은 메시지가 출력됩니다.

```
* Running on http://127.0.0.1:5000
* Press CTRL+C to quit
```

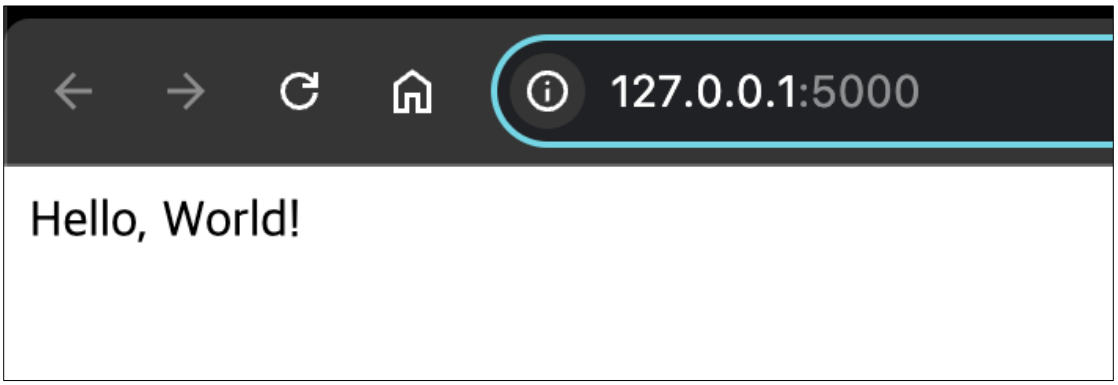


이 메시지는 Flask 서버가 성공적으로 실행되었고, 로컬에서 `http://127.0.0.1:5000` 주소로 접속할 수 있음을 나타냅니다. 브라우저에서 이 주소로 접속하면 웹 페이지가 정상적으로 로드됩니다.

5. 웹 브라우저에서 'Hello, World!' 확인

웹 브라우저에서 출력 확인

서버가 실행된 후, 웹 브라우저에서 `http://127.0.0.1:5000`로 접속하면 "Hello, World!"라는 메시지가 출력된 웹 페이지가 나타납니다. 이는 `app.py` 파일에서 정의한 기본 경로로 요청이 들어왔을 때, `hello_world` 함수가 호출되어 반환된 메시지입니다.



이 화면이 출력되면 Flask 애플리케이션이 정상적으로 동작하고 있음을 의미합니다.

6. 결론

이번 실습에서는 Flask 프레임워크를 사용해 간단한 웹 애플리케이션을 구축하는 방법을 배웠습니다. Flask는 파이썬 기반의 마이크로 프레임워크로, 가벼운 구조와 직관적인 사용법 덕분에 웹 개발 입문자에게 적합합니다. 간단한 코드로 웹 서버를 실행하면서 라우팅, HTTP 응답 처리, 서버 실행에 대한 기본 개념을 익힐 수 있었습니다.

또한, Flask 에서 제공하는 개발 서버는 빠르고 간편한 애플리케이션 테스트에 매우 유용하다는 점도 배울 수 있었습니다. Flask 개발 서버는 코드 수정 시 자동으로 서버를 재시작하는 기능을 제공하여 개발 속도를 크게 향상시키며, 디버깅 도구를 통해 문제를 쉽게 찾고 수정할 수 있는 장점을 가지고 있습니다. 이러한 기능들은 프로토타입 개발과 시범 운영 단계에서 매우 유용하게 사용할 수 있습니다.

그러나 Flask 개발 서버는 몇 가지 중요한 한계를 가지고 있습니다.

첫째, Flask 의 개발 서버는 성능이 제한적입니다. 이는 소규모 트래픽을 처리할 때는 문제가 없지만, 많은 양의 요청을 효율적으로 처리할 수 있는 구조로 설계되지 않았습니다. 따라서, 실제 서비스에서 많은 사용자의 요청이 발생하는 환경에서는 성능 저하가 발생할 수 있습니다.

둘째, 개발 서버는 보안 측면에서 취약할 수 있습니다. Flask 의 개발 서버는 보안 기능이 강화된 프로덕션용 서버와 달리, 기본적인 보안 설정이 되어 있지 않습니다. 예를 들어, 데이터 암호화나 보안 인증 등 중요한 보안 기능이 포함되지 않아, 외부의 악의적인 공격에 노출될 가능성이 큽니다. 이러한 보안적 취약성은 실제 운영 환경에서 심각한 문제로 작용할 수 있습니다.

이처럼 성능과 보안 측면에서의 한계를 감안할 때, Flask 개발 서버는 프로덕션 환경에서 사용하기보다는 프로토타입 및 개발 단계에서 테스트용으로 사용하는 것을 권장합니다. 실제 운영 환경에서는 더 많은 트래픽을 처리하고 보안이 강화된 WSGI(Web Server Gateway Interface) 와 같은 프로덕션 환경에서 사용할 수 있는 서버를 사용하여 애플리케이션을 안정적으로 배포하는 것이 중요합니다.

이번 실습을 통해 Flask의 기본 개념과 활용 방법을 이해했으며, 향후 더 복잡한 기능을 추가하거나 확장된 애플리케이션 개발에 필요한 기초를 다질 수 있었습니다. Flask의 유연성과 확장성을 활용하여 더 복잡한 프로젝트로 발전시킬 가능성을 확인할 수 있었습니다.

참고 문헌 https://flask-docs-kr.readthedocs.io/ko/latest/quickstart.html