목차

1. Django Intro
2. Templates & URLs
3. Model
4. ORM
5. ORM with View
6. Django Form
7. Static
8. Authentication system 1

1. Django Intro

Web application (web service) 개발
인터넷을 통해 사용자에게 제공되는 소프트웨어 프로그램을 구축하는 과정

다양한 디바이스(모바일, 태블릿, PC)에서 웹 브라우저를 통해 접근하고 사용가능

Client
서비스를 요청하는 주체
(웹 사용자의 인터넷이 연결된 장치, 웹 브라우저)

Server
클라이언트의 요청에 응답하는 주체
(웹 페이지, 앱을 저장하는 컴퓨터)

Frontend
사용자 인터페이스(UI)를 구성하고, 사용자가 애플리케이션과 상호작용할 수 있도록 함
(HTML, CSS, JavaScript, 프론트엔드 프레임워크 등)

Backend
서버 측에 동작하며, 클라이언트의 요청에 대한 처리와 데이터베이스와의 상호작용을 담당
(서버 언어, 백엔드 프레임워크, DB, API, 보안 등)

Web Framework
웹 애플리케이션을 빠르게 개발할 수 있도록 도와주는 도구
(개발에 필요한 기본 구조, 규칙, 라이브러리 등을 제공)

django
Python 기반의 대표적인 웹 프레임워크

django 특징

1. 다양성 Python 기반으로 소셜 미디어 및 빅데이터 관리 등 광범위한 서비스 개발에 적합

2. 확장성
   대량의 데이터에 대해 빠르고 유연하게 확장할 수 있는 기능을 제공

3. 보안
   취약점으로부터 보호하는 보안 기능이 기본적으로 내장

4. 커뮤니티 지원
   개발자를 위한 지원, 문서, 업데이트를 제공하는 활성화 된 커뮤니티

가상환경
Python 애플리케이션과 그에 따른 패키지들을 격리하여 관리할 수 있는 독립적인 실행 환경

의존성 패키지
패키지가 다른 패키지의 기능&코드를 사용하기 때문에 그 패키지가 존재해야 제대로 작동하는 관계

가상환경을 사용하는 이유

1. 의존성 관리
   라이브러리 및 패키지를 각 프로젝트마다 독립적으로 사용가능

2. 팀 프로젝트 협업
   모든 팀원이 동일한 환경&의존성 위에서 작업해 버전간 충돌을 방지

LTS (Long-Term Support)
프레임워크&라이브러리 등에서 장기간 지원되는 안정적인 버전

기업 및 대규모 프로젝트에서는 소프트웨어 업그레이드에 많은 비용과 시간이 필요하기에 안정적이고 장기간 지원되는 버전이 필요

디자인 패턴
소프트웨어 설계에서 발생하는 문제 해결을 위한 일반적인 해결책
(공통적인 문제를 해결하는데 쓰이는 형식화된 관행)

MVC 디자인 패턴
Model, View, Controller
애플리케이션을 구조화하는 대표적인 패턴
('데이터 & 사용자 인터페이스 & 비즈니스 로직'을 분리)

-> 시각적 요소와 뒤에서 실행되는 로직을 서로 영향 없이, 독립적이고 쉽게 유지 보수할 수 있기 위함

MTV 디자인 패턴
Model, Template, View
Django에서 애플리케이션을 구조화하는 패턴
(기본 MVC 패턴과 동일, 명칭만 다르게 정의한 것 / View -> Template, Controller -> View)

1. Model
   데이터와 관련된 로직을 관리
   응용프로그램의 데이터 구조를 정의하고 데이터베이스의 기록을 관리

2. Template
   레이아웃과 화면을 처리
   화면상의 사용자 인터페이스 구조와 레이아웃을 정의

3. View
   Model & Template과 관련한 로직을 처리해서 응답을 반환
   클라이언트의 요청에 대해 처리를 분기하는 역할

Django project
애플리케이션의 집합 (DB설정, URL연결, 전체 앱 설정 등을 처리)

Django application
독립적으로 작동하는 기능 단위 모듈 (각자 특정한 기능을 담당하며 다른 앱들과 함께 하나의 프로젝트를 구성)

프로젝트 구조
settings.py : 프로젝트의 모든 설정 관리
urls.py : 요청 들어오는 URL에 따라 해당하는 views를 연결
admin.py : 관리자용 페이지 설정
models.py : DB와 관련된 Model을 정의 (MTV 패턴의 M)
views.py : HTTP 요청을 처리하고 해당 요청에 대한 응답을 반환 (MTV 패턴의 V)

URLs
http://기본주소/index/로 요청이 들어오면 view 함수의 index 함수를 호출한다는 뜻
-> url경로는 반드시 / 로 끝나야함
-> 예시 : path('index/', views.index)

View
특정 경로에 있는 template과 request 객체를 결합해 응답 객체를 반환하는 view 함수를 정의

모든 view 함수는 첫번째 인자로 request 요청 객체를 필수로 받는다.

```
def index(request):
    return render(request, 'articles/index.html')
```

Template
생성 순서

1. articles 앱 폴더 안에 templates 폴더 생성

2. templates 폴더 안에 artilces 폴더 생성

3. articles 폴더 안에 템플릿 파일(html) 생성

render 함수
주어진 템플릿을 주어진 context 데이터와 결합하고 렌더링 된 텍스트와 함께 HttpResponse 응답 객체를 반환하는 함수

예시 : render(request, template_name, context)

1. request : 응답을 생성하는데 사용되는 요청 객체
2. template_name : 템플릿 이름의 경로
3. context : 템플릿에서 사용할 데이터 (딕셔너리 타입으로 작성)

---

2. Templates & URLs

Django Template system
데이터 표현을 제어하면서, 표현과 관련된 부분을 담당

Django Template Language (DTL)
Template에서 조건, 반복, 변수 등의 프로그래밍적 기능을 제공하는 시스템

DTL Syntax
종류 : Variable, Filters, Tags, Comments

1. Variable
   render 함수의 세번째 인자로 딕셔너리 데이터를 사용
   딕셔너리 key에 해당하는 문자열이 template에서 사용가능한 변수명이 됨
   dot(.)을 사용하여 변수 속성에 접근할 수 있음
   예시 : {{ variable }} / {{ variable.attribute }}

2. Filters
   표시할 변수를 수정할 때 사용 (변수 + | + 필터)
   chained(연결)이 가능하며 일부 필터는 인자를 받기도 함
   예시 : {{ variable|filter }} / {{ name|truncatewords:30 }}

3. Tags
   반복 또는 논리를 수행하여 제어흐름을 만듦
   일부 태그는 시작과 종료 태그가 필요
   예시 : {{ tag }} / {% if %} {% endif %}

4. Comments
   DTL에서의 주석

템플릿 상속(Template inheritance)
페이지의 공통요소를 포함하고, **하위 템플릿이 재정의 할 수 있는 공간**을 정의하는 기본 'skeleton' 템플릿을 작성하여 상속 구조를 구축

'extends' tag
{% extends 'path' %}
자식(하위) 템플릿이 부모 템플릿을 확장한다는 것을 알림
(반드시 자식 템플릿 최상단에 1개만 작성해야 함)

'block' tag
{% block name %} {% endblock name %}
하위 템플릿에서 재정의 할 수 있는 블록을 정의
(상위 템플릿에 작성하며 하위 템플릿이 작성할 수 있는 공간을 지정하는 것)

HTML form (요청과 응답)
HTTP 요청을 서버에 보내는 가장 편리한 방법

'form' element
사용자로부터 할당된 데이터를 서버로 전송
-> 웹에서 사용자 정보를 입력하는 여러 방식(text, password, checkbox 등)을 제공

action & method
form의 핵심 속성 2가지
"데이터를 어디(action)로 어떤 방식(method)으로 요청할지"

action
입력 데이터가 전송될 URL을 지정 (목적지)
만약 이 속성을 지정하지 않으면 데이터는 현재 form이 있는 페이지의 URL로 보내짐

method
데이터를 어떤 방식으로 보낼 것인지 정의
데이터의 HTTP request methods (GET, POST)를 지정

'input' element
사용자의 데이터를 입력 받을 수 있는 요소
(type 속성 값에 따라 다양한 유형의 입력 데이터를 받음)

'name' attribute
input의 핵심 속성
입력한 데이터에 붙이는 이름(key)
데이터를 제출했을 때 서버는 name 속성에 설정된 값을 통해서만 사용자가 입력한 데이터에 접근할 수 있음

Query String Parameters
사용자의 입력 데이터를 URL 주소에 파라미터를 통해 서버로 보내는 방법

문자열은 앰퍼샌드(&)로 연결된 key=value 쌍으로 구성되며 기본 URL과는 물음표(?)로 구분됨

예시 : http://host:port/path?key=value&key=value

2일차 교재 41쪽부터 하면 됨

Django 루틴

※ 설계도 작성 전에 모델 변경을 끝내야 한다!

1. 가상환경 생성
   python -m venv venv
2. 가상환경 활성화
   source venv/Scripts/activate
3. Django 설치
   pip install django
   3-1.requirements.txt 전체 설치
   pip install -r requirements.txt
4. 의존성 파일 생성
   pip freeze > requirements.txt
5. Django 프로젝트 생성
   django-admin startproject firstpjt .
6. 앱 생성
   python manage.py startapp articles
7. Django 서버 실행
   python manage.py runserver
8. 앱 등록
   settings.py에서 추가하기
9. 최종 설계도 작성
   python manage.py makemigrations
10. DB에 적용
    python manage.py migrate