웹 서비스 개발

- 모든 걸 직접 만들 필요 없다
- 잘 만들어진 것들을 가져다가 좋은 환경에서 잘 쓰기만 하면 되는 세상

Framework 프레임워크란

- 누군가 만들어 놓은 코드를 재사용 하는 것
- 전 세계의 수많은 개발자들이 이미 수없이 많이 개발해 봤고, 그 과정에서 자주 사용되는 부분들을 재사용 할 수 있게 좋은 구조의 코드로 만들어 두었다.
- 즉 서비스 개발에 필요한 기능들을 미리 구현해서 모아 놓은 것.
- 소프트웨어 프레임워크는 복잡한 문제를 해결하거나 서술하는데 사용되는 기본 개념 구조
- 소프트웨어의 생산성과 품질을 높임 -> 하나하나 만드는것에 비해 시간이 더 여유로워지기 때문.

WWW (world wide web) : 전세계에 퍼져 있는 거미줄 같은 연결망

해저 케이블을 통해 컴퓨터간 통신을 할 수 있다.

결국 인터넷을 사용한다는 것은 전세계의 컴퓨터가 연결되어 있는 하나의 인프라를 이용하는 것.

오늘날 우리가 사용하는 대부분의 웹 서비스는 클라이언트-서버 구조를 기반으로 동작

클라이언트 -> 서버 (request)

서버 -> 클라이언트 (response)

웹 브라우저

- 웹에서 페이지를 찾아 보여주고, 사용자가 하이퍼링크를 통해 다른 페이지로 이동할 수 있도록 하는 프로그램
- 웹 페이지 파일을 우리가 보는 화면으로 바꿔주는 프로그램 (렌더링)
- 우리가 보고 있는 웹 페이지는 사실 HTML 문서 파일 하나임
- 우리가 보는 화면 각각 한 장 한 장이 웹 페이지
- 종류
    - 정적 웹 페이지 : 있는 그대로를 제공하는 것 (요즘은 거의 사용되지 않는다고 함.)
    - 동적 웹 페이지 : 사용자의 요청에 따라 웹 페이지에 추가적인 수정이 되어 클라이언트에게 전달되는 웹 페이지

Design pattern

- 여러 번 짓다보니 자주 사용되는 구조가 있다는 것을 알게 되었고 이를 일반화해서 하나의 공법으로 만들어 둔 것
- 공통정을 묶어서 패턴을 만듦
- 목적
    - 특정 문맥에서 공통적으로 발생하는 문제에 대해 재사용 가능한 해결책을 제시
    - 프로그래머가 어플리케이션이나 시스템을 디자인 할 때 발생되는 공통된 문제들을 해결하는데 형식화 된 가장 좋은 관행
- 장점 : 디자인 패턴을 알고 있다면 서로 복잡한 커뮤니케이션이 매우 간단해짐,
    - 다수의 엔지니어들이 일반화된 패턴으로 소프트웨어 개발을 할 수 있도록 한 규칙, 커뮤니케이션의 효율성을 높이는 기법

Django's Design patter

장고에서 사용하는 디자인 패턴은 MTV(Model, Template, View) 패턴으로 기존의 MVC(Model, View, controller)을 기반으로 조금 변형된 패턴이다.

MVC 란?

- 하나의 큰 프로그램을 세가지 역할로 구분한 개발 방법론
- Model : 데이터와 관련된 로직을 관리
- View : 레이아웃과 화면을 처리
- Controller : 명령을 모델과 뷰 부분으로 연결
- 각 부분을 독립적으로 개발할 수 있어서 하나를 수정하고 싶을 때 모두를 건들지 않아도 된다.
- 즉 개발 효율성 및 유지보수가 쉬워지며, 다수의 멤버로 개발하기 용이하다.

MTV

- model : 데이터와 관련된 로직을 관리(DB) urls.py
- Template : 레이아웃과 화면을 처리 (MVC의 view를 담당) .html
- View : model & Template 와 관련한 로직을 처리해서 응답을 반환 view.py

Django 설치

python -m venv venv

pip install django==3.2.13

pip freeze > requirements.txt

django-admin startproject {{프로젝트 이름}} .

python manage.py startapp {{앱 이름}} <----- 앱 이름은 관행적으로 복수형으로 작성하는 것을 권장

앱 등록시 settings.py의 INSTALLED_APPS 에 등록하기 그리고 콤마 붙이는것 꼭 기억하기!

요청과 응답

기본 순서는 url -> view -> template

URL

- path('{{페이지/}}, view.{{페이지 이름}}, name="{{이름}}")

View

- 함수 작성
- render(request, '{{html이름.html}}', context)

Template

- 실제 내용을 보여주는데 사용되는 파일
- 템플릿 폴더 이름은 반드시 templates 라고 해야하며, settings.py 에서 'DIRS': [BASE_DIR, 'templates',] 이렇게 작성하게 되면 특정 경로를 절대 경로로 편하게 작성할 수 있도록 하게 한다.
- 그리고 상속을 받게 되면 프로젝트가 있는 위치에서 templates 폴더를 만든 후 보통 이름을 base.html 로 작성한 후 상속을 위해 block 태그를 사용해준다.

USE_I18N : 장고의 번역 시스템을 활성화해야 하는지 여부를 지정

Django Template

데이터 표현을 제어하는 도구이자 표현에 관련된 로직

Django Template System : 데이터 표현을 제어하는 도구이자 표현에 관련된 로직을 담당

Django Template Language(DTL)

- 장고 템플릿에서 사용하는 built-in template system
- 조건, 반복, 변수 치환, 필터 등의 기능을 제공
    - 파이썬처럼 일부 프로그래밍 구조(if,for 등) 를 사용할 수 있지만 파이썬 코드로 실행되는 것이 아님!
    - 그래서 장고 템플릿 시스템은 단순히 파이썬이 HTML에 포함된 것이 아니니 주의해야하며
    - 프로그래밍적 로직이 아닌 프레젠테이션을 표현하기 위한 것임을 명심해야함.
- DTL Syntax
    - Variable : {{ variable }}, dot(.) 을 사용하여 변수 속성에 접근할 수 있으며, render의 세번째 인자로 딕셔너리 형태로 넘겨주며 key 값은 template에서 사용가능한 변수명이 된다.
    - Filters : {{ variable:filter }}, 표시할 변수를 수정할 때 사용
    - Tags : {% tag %}, 출력 텍스트를 만들거나, 반복 또는 논리를 수행하여 제어 흐름을 만드는 등 변수보다 복잡한 일들을 수행, 일부 태그(ex. if 태그)는 시작과 종료 태그가 필요
    - Comments : 장고에서 라인의 주석을 표현하기 위해서 사용, 한 줄 주석은 {# 내용 #} 작성 , 여러줄은 {% comment %} 내용 {% endcomment %}

Template inheriance

템플릿 상속

- 템플릿 상속은 기본적으로 코드의 재사용성에 초점을 맞춤
- 템플릿 상속을 사용하면 사이트의 모든 공통 요소를 포함하고, 하위 템플릿에 재정의(override) 할 수 있는 블록을 정의하는 기본 'skeleton' 템플릿을 만들 수 있다.
- 템플릿 상속에 관련된 태그
    - {% extends ' ' %} : 반드시 최상단에 작성되어야 하며 1개만 작성 가능
    - {% block content %} {% endblock content %} : 하위 템플릿에서 재지정(override) 할 수 있는 블록을 정의, 즉 하위 템플릿이 채울 수 있는 공간

Sending form data (Client)

HTML

form element

- 데이터가 전송되는 방법을 정의
- 데이터를 어디(action)로 어떤 방식(method)으로 보낼지
- 핵심 속성
  - action : 입력 데이터가 전송될 URL을 지정
    - 만약 속성을 지정하지 않으면 데이터는 현재 form이 있는 페이지의 URL로 보내짐
  - method : 데이터를 어떻게 보낼 것인지 정의
    - 종류
      - GET
      - POST

input element

- 사용자로부터 데이터를 입력 받기 위해 사용
- type 속성에 따라 동작 방식이 달라진다.
- 타입을 지정하지 않은 경우 default 값은 text
- 핵심 속성 :
  - name : form을 통해 데이터를 제출(submit) 했을 때 name 속성에 설정된 값을 서버로 전송하고, 서버는 name 속성에 설정된 값을 통해 사용자가 입력한 데이터 값에 접근할 수 있다.
    - 주요 용도는 GET/POST 방식으로 서버에 전달하는 것, 파라미터는 name은 key, value는 value로 매핑
    - GET 방식에서는 URL에서 '? key=value&key=value/' 형식으로 데이터를 전달

HTTP request methods

- HTTP(HyperText Transfer Protocol) : HTML 문서와 같은 리소스(데이터, 자원)들을 가져올 수 있도록 해주는 프로토콜(규칙, 규약)
- HTTP는 주어진 리소스가 수행 할 원하는 작업을 나타내는 request methods를 정의
- HTTP Method 예시
  - GET : 서버로부터 정보를 조회하는데 사용, 데이터를 서버로 전송할 때 Query string parameters를 통해 전송
  - POST,
  - PUT,
  - DELETE

Variable routing

Variable routing

- URL 주소를 변수로 사용하는 것을 의미
- 즉 변수 값에 따라 하나의 path()에 여러 페이지를 연결 시킬 수 있음
- 변수는 <>에 정의하며 view 함수의 인자로 할당됨

App URL mapping

App URL mapping : 앱이 많아 졌을때 프로젝트의 urls.py에서 모두 관리하는 것은 프로젝트 유지보수에 좋지 않기에 사용

하나의 프로젝트에 여러 앱이 존재한다면, 각각의 앱 안에 urls.py를 만들고 프로젝트 urls.py에서 각 앱의 urls.py 파일로 URL 매핑을 위탁할 수 있음

그리고 각각의 앱의 urls.py 에서는 app_name 과 urlpatterns = [] 를 꼭 작성해야한다.

include():

- urlpattern 은 언제든지 다른 URLconf 모듈을 포함(include) 할 수 있다
- URL 의 그 시점까지 일치하는 부분을 잘라내고, 남은 문자열 부분의 후속 처리를 위해 include 된 URLconf로 전달

Built-in tag -url

- 주어진 url 패턴 이름 및 선택적 매개 벼수와 일치하는 절대 경로 주소를 반환
- 템플릿에 URL을 하드 코딩하지 않고도 DRY(Don't Repeat Yourself) 원칙을 위반하지 않으면서 링크를 출력하는 방법

장고의 설계 철학(Templates System)

1. 표현과 로직(View)을 분리
2. 중복을 배제

프레임워크의 성격

- 독선적(Opinionated)
- 관용적(Unopinionated)
- 다소 독선적
- 즉 프레임워크는 우리가 하는 개발을 방해하는 것이 아닌 온전히 만들고자 하는 것에만 집중할 수 있게 도와주는 것.