**쟝고 설치**

pip install Django

**쟝고 프로젝트 생성 ( 현재 디렉토리 확인 )**

django-admin startproject web1

**디렉토리 이동 후 확인**

cd web1

dir

**2019-12-30 오전 09:10 <DIR> .**

**2019-12-30 오전 09:10 <DIR> ..**

**2019-12-30 오전 09:10 645 manage.py**

**2019-12-30 오전 09:10 <DIR> web1**

**1개 파일 645 바이트**

**3개 디렉터리 166,770,749,440 바이트 남음**

**앱을 시작한다 – member는 원하는 이름**

django-admin startapp member

**서버 실행 ( 기본 디렉토리의 manage.py를 통해서 )**

python manage.py runserver

**\*\* 프로젝트 폴더에는 urls.py가 자동생성되나 앱 폴더에는 손수 만들어줘야 한다. \*\***

**< member 앱 폴더에 urls.py를 만들어줘라 >**

크롬에 주소를 치면 – urls.py에서 주소를 확인

urls.py에서는 views.py에서 함수를 확인 – **해당 함수가 views.py에 만들어져 어야 한다.**

127.0.0.1:8000/member/index

member : 상위 url, index: 하위 url

하위 url은 web1/member에서상위 url은 총괄하는 web1/web1에서 설정

**web1/web1/urls.py**

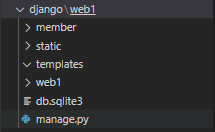
from django.urls import path, include

urlpatterns = [

    path('admin/', admin.site.urls),

    path('member/', include('member.urls'))

]

**web1 -> DB, 상위 url 등 전체 환경**

**member -> 특정 앱**

**static -> CSS, font 등등**

**templates -> HTML**

**templates를 쓰기 위해서는 setting.py에 수정이 필요함. – 경로 잡아주기**

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [ os.path.join(BASE\_DIR,'templates') ],

아래 줄 추가

**settings.py**

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [ os.path.join(BASE\_DIR,'templates') ],

        'APP\_DIRS': True,

        'OPTIONS': {

            'context\_processors': [

                'django.template.context\_processors.debug',

                'django.template.context\_processors.request',

                'django.contrib.auth.context\_processors.auth',

                'django.contrib.messages.context\_processors.messages',

            ],

        },

    },

]

**데이터베이스 초기 설정을 잡아주는 것 – 각종 기본 테이블 등등이 설정된다.**

python manage.py migrate

앞서 했던 것처럼 cursor 잡아주고 sql 생성 후 execute해주면 된다.

**DB 설정을 바꿀 때마다 – setting.py 중간 Database에 변경해주어야 함.**

python manage.py migrate를 해주어야 한다.

setting.py 맨 아래에

STATIC\_URL = '/static/'

STATICFILES\_DIRS = [os.path.join(BASE\_DIR, 'static')]

아래 줄 추가

**부트스트랩 사용**

bootstrap에서 example 중에서 선택 후 **페이지 소스 보기**

**1.** stylesheet부터 확인해야 한다.

**2.** <div></div>를 확인하면서 구분하면서 확인한다. and copy&paste

**<< \*\* 추가 사항 \*\* >>**

a 태그는 **GET**방식, **POST**로 할거면 form 안에 method 설정으로 해주어야 한다.

<form action = "/member/delete" method="POST">

            <input type="submit" value="회원탈퇴" />

        </form>

**---------------------------------------------------------------------------------------------**

**<< \*\* 페이지 추가 시 순서 \*\* >>**

**1.** urls.py에 등록

**2.** views.py에 def로 함수를 선언한다. -> GET, POST에 맞춰서 로직 구성

함수 선언 시 **파라미터는 무조건** **request**

POST 동작 시에 보안관련 decorator 추가해야 함

@csrf\_exempt

**3.** HTML이 필요한 경우 **../templates/[앱명]**에 [페이지명].py로 하면 생성

! > Enter 자동생성하면 기본 틀 만들어짐.

**---------------------------------------------------------------------------------------------**

**<< \*\* 앱 추가 시 순서 \*\* >>**

1. **django-admin startapp [앱 이름] => 상위 도메인!**

**2.** 해당 폴더에 **urls.py** 생성 후 작성 ( 기본관리폴더(프로젝트명)에는 자동생성됨 )  
- urlpatterns는 필요할 때마다 양식에 맞게 추가

from django.urls import path

from . import views

urlpatterns = [

    path('index', views.index, name="index"),

    path('join', views.join, name="join"),

]

**3.** **[프로젝트 명] 폴더/urls.py**의 urlpatterns에 앱(상위 도메인)의 경로 추가

    path('member/', include('member.urls'))

**4.** **views.py**에 import 추가 -> 이후 추가할 하위도메인에 해당하는 함수를 선언

from django.shortcuts import render,redirect

from django.http import HttpResponse

from django.views.decorators.csrf import csrf\_exempt

#SQLite 연결

from django.db import connection

cursor = connection.cursor()

**5.** **templates** 폴더 생성(HTML 모음) -> **[프로젝트 명] 폴더/settings.py** 수정 필요

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [ os.path.join(BASE\_DIR,'templates') ],

....>

**6.** static 폴더 생성 -> css, js, img 폴더 생성 -> **[프로젝트 명] 폴더/settings.py** 수정 필요

STATIC\_URL = '/static/'

STATICFILES\_DIRS = [os.path.join(BASE\_DIR, 'static')]

하면서 시간대도 서울로 수정

# TIME\_ZONE = 'UTC'

TIME\_ZONE = 'Asia/Seoul'

**7.** 데이터베이스(DB) 사용시

7.1 **[프로젝트 명] 폴더/settings.py** 수정 필요

DATABASES = {

    'default': {

        # sqlite DB 사용 시

        # 'ENGINE': 'django.db.backends.sqlite3',

        # 'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),

        # mysql

        # 'ENGINE': 'django.db.backends.mysql',

        # 'NAME': 'DB명',

        # 'USER': '아이디명',

        # 'PASSWORD': '암호',

        # 'HOST': '서버주소',

        # 'PORT': '포트번호'

        # oracle

        'ENGINE': 'django.db.backends.oracle',

        'NAME': 'xe', #SID

        'USER': 'admin',

        'PASSWORD': '1234',

        'HOST': '192.168.99.100',

        'PORT': '32764'

    }

}

추가로 데이터베이스(DB) 설정을 바꿀 때마다 콘솔에 초기 설정을 잡아주어야 함.

**python manage.py migrate**

**---------------------------------------------------------------------------------------------**

**<< \*\* 프로젝트 추가 방법\*\* >>**

1. 기본 베이스로 잡을 경로로 **cd [경로]** 해준다.

2. 경로 확인 후 **django-admin startproject [프로젝트 명]**

3. cd **[프로젝트 명]**

기본적으로 생성되는 폴더는 프로젝트 명과 동일

- 이 폴더는 전체 프로젝트를 총괄하는 폴더

- 기본적인 settings.py 관련은 위의 프로젝트 시작 단계에서 확인

**---------------------------------------------------------------------------------------------**

**<< \*\* 프로젝트 실행 방법 \*\* >>**

1. 서버 작동

**python manage.py runserver**

2. 서버 끄기

**Ctrl + c 여러 번**

**항상 현재 디렉토리가 어디인지 확인**

**---------------------------------------------------------------------------------------------**

**<< \*\* 데이터베이스(DB) 사용 시 \*\* >>**

1. **connection**을 import해준다. -> 선언까지 같이

from django.db import connection

cursor = connection.cursor()

cursor는 sql의 수행단위

2. DB에서 데이터를 가져올 때 (GET에서 가져와서 페이지에 값을 띄울 때)

    sql = "SELECT \* FROM MEMBER ORDER BY ID ASC"

    cursor.execute(sql)

    data = cursor.fetchall();

sql 작성하고 cursor에서 execute해준다.

가져올 라인 수가 1개 => **cursor.fetchone()** => datatype은 ( ) = 튜플

가져올 라인 수가 1개 이상 => **cursor.fetchall()** => datatype은 [( ), ( ), ( )] = (튜플의)리스트

return render(request, 'member/list.html', {"list":data, "title": "회원목록"})

render 시에 파라미터를 dictionary형식으로 전달

3. DB에서 데이터를 가져올 때 (POST로 페이지에서의 값을 데이터베이스로 저장할 때)

        ar = [

            request.POST['name'],

            request.POST['age'],

            request.POST['id']

        ]

HTML의 input태그의 name을 통해서 가져와서 리스트 형식으로 세팅

 sql = """

            UPDATE MEMBER SET NAME=%s, AGE=%s

            WHERE ID=%s

        """

        cursor.execute(sql, ar)

sql 작성 후 **execute(sql, ar) ->** ar에는 %s로 비운 순서대로 나열되어 있어야 한다.

        return redirect("/member/index")

**POST** 형식이기 때문에 **return은 redirect**이다!

**---------------------------------------------------------------------------------------------**

**<< \*\* 모델을 통한 디비 연결 \*\* >>**

**1. models.py**에 데이터베이스 틀을 추가 ( DB에 자동생성된다고 함? )

class Table1(models.Model):

    object  = models.Manager() # vs code 오류 밑줄 제거용, 안써도 된다.

    no      = models.AutoField(primary\_key=True)

    title   = models.CharField(max\_length=200)

    content = models.TextField()

    writer  = models.CharField(max\_length=50)

    hit     = models.IntegerField()    regdate = models.DateField(auto\_now\_add=True)

**2. settings.py**에 설정 추가 - DB 처음에 migrate로 기본 설정을 잡아주는 것이 이 부분

- 여기에 **model을 잡아준 앱 이름**을 넣어주어야 **migrate**로 잡아준다.

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'board',

]

3. 서버 종료 후에 board를 migrate해주는 절차가 필요

\* manage.py의 상태 체크

**python manage.py check**

\* board를 migration에 추가 – **board/migration에 py파일을 추가한다.** ( git의 commit같은 느낌 )

**python manage.py makemigrations board**

\* board를 migrate! – DB에 적용

**python manage.py migrate board**