**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**

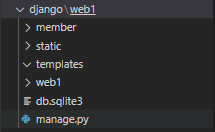
from django.urls import path, include

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 )

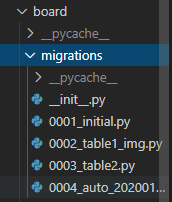
**python manage.py makemigrations board**

\* board를 migrate! – DB에 적용

**python manage.py migrate board**

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

**<< \*\* model을 수정하면? \*\* >>**



앱 아래의 migrations에 로그처럼 쌓인다. 이는 DB의 migrations에서 전체 확인가능

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

**<< \*\* /admin url을 활용하자 \*\* >>**

**=>** 이는 **django 2.2.5버전**으로 해야 가능, **3.1.0으로 가면 버그 발생**

django 버전 확인

**conda list => 알파벳 순이니 d열 가서 확인**

**pip install django==2.2.5 => 버전 변경**

서버 끄고 콘솔에 실행

**python manage.py createsuperuser**

-> **id는 unique**, email은 enter로 생략, password는 짧다고 뭐라해도 걍 y + enter

**127.0.0.1:8000/admin**에서 로그인하면 administor화면 보인다

<http://127.0.0.1:8000/admin>

**모델이 있는 앱 폴더**의 admin.py에 코드 추가

from board.models import Table1

admin.site.register(Table1)

admin에서 해당 앱의 model을 import **[네임]**으로 확인, 수정 가능해진다.

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

**<< \*\* model을 활용한 sql축약 \*\* >>**

# SELECT SUM(math) FROM MEMBER\_TABLE2

test = Table2.objects.aggregate(Sum("math"))

test = Table2.objects.raw("SELECT SUM(math) FROM MEMBER\_TABLE2")

# 출력은 <RawQuerySet: SELECT SUM(math) FROM MEMBER\_TABLE2> 값은 정상적으로 가는 듯

# SELECT NO, NAME FROM MEMBER\_TABLE2

test = Table2.objects.all().values('no','name')

# SELECT \* FROM MEMBER\_TABLE2 ORDER BY name ASC

list = Table2.objects.all().order\_by('name')

#list = Table2.objects.raw("SELECT \* FROM MEMBER\_TABLE2 ORDER BY name ASC")

# 반별 국어, 영어, 수학 합계

# SELECT SUM(kor) AS kor, SUM(eng) AS eng, SUM(math) AS math FROM MEMBER\_TABLE2 GROUP BY CLASSROOM

test = Table2.objects.values('classroom').annotate(kor=Sum('kor'),eng=Sum('eng'),math=Sum('math'))

rows = Table2.objects.filter(classroom=cls)

일반 함수들은 단순히 쓰고 받아오면 쓸 수 있다.

-> **member/view.py > exam\_select**

sqlS = "SELECT 1 as no,SUM(kor) as skor,SUM(eng) as seng,SUM(math) as smath FROM ("+ sql + ")"

sqlA = "SELECT 1 as no,AVG(kor) as akor,AVG(eng) as aeng,AVG(math) as amath FROM ("+ sql + ")"

sum\_list = Table2.objects.raw(sqlS)

avg\_list = Table2.objects.raw(sqlA)

sum\_avg\_list = [

    sum\_list[0].skor,

    sum\_list[0].seng,

    sum\_list[0].smath,

    # avg\_list[0].akor,

    int(avg\_list[0].akor),

    # avg\_list[0].aeng,

    int(avg\_list[0].aeng),

    # avg\_list[0].amath,

    int(avg\_list[0].amath), ]

그룹함수는 다르다. raw 방식으로 sql을 쓰려면 기본키가 있어야 한다.

-> 1 as no(DB의 PK명)으로 주면 PK가 있는 것으로 인식한다.

-> **member/view.py > sum\_avg**

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

**<< \*\* url control \*\* >>**

127.0.0.1:8000/member/exam\_select?**no=1&cls=102 --> GET변수를 담을 때 ? 뒤로 쭉**

**--> 여러개 면 &로 구분**

**- 다른 페이지로 바꾸면서 파라미터를 유지하려면?**

view.[페이지] 의 get방식에서 받고 url render하면서 딕셔너리에 같이 넘겨주고,

[페이지].html에서 ?뒤의 변수로 넘겨준다.

필요한 경우 {% if ~~ %}를 활용한다.

**- redirect**

해당 url을 엔터에 치는 것과 같다. -> 즉, ?붙이고 변수써도 멀쩡히 굴러간다.

return redirect("/member/exam\_select?page=2")

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

**<< \*\* jquery 사용 \*\* >>**

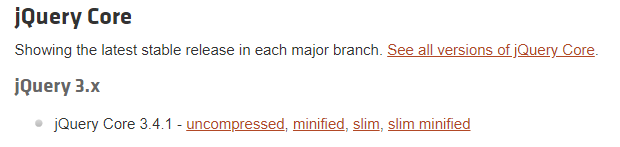
**CDN**

- HTML 내의 <script> 태그를 위해 사용

- 각 브라우저 별 엔진이 다르기 때문에 브라우저별로 다르게 사용해야할 때,

- CDN은 각각에 맵핑을 해준다. (아닐 수도)

우리는 **jquery cdn**을 쓴다. <http://code.jquery.com/>



위의 **uncompressed** 혹은 **minified(저용량)** 버전의 **<script></script>를 HTML**에 넣어주면 끝.

    <script src="http://code.jquery.com/jquery-3.4.1.min.js"

        integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFlBw8HfCJo="

        crossorigin="anonymous">

    </script>

입력하면 script 태그 사용 가능

    <script>

        $(function(){

            $('#name').val('jquery')

            $('#btn\_delete').click(function(){

                $('#form').attr('action', /member/test\_delete)

            })

        })

    </script>

**id가 name**인 곳에 **value를 jquery**로 줘라

$('#name').val('jquery')

**btn\_delete**에 **click**하면 **function**을 실행해라

**form**을 찾아가서 **attribute를 수정**해라, **action 태그를 /member/test\_delete로**

$('#btn\_delete').click(function(){

    $('#form').attr('action', /member/test\_delete)

})

**js\_chart – 다양**

d3가 가장 좋은 형태, but 너무 복잡 and 어렵 / <https://d3js.org/>

c3를 쓴다. 간단, 쉬움 / <https://c3js.org/>

- c3를 위한 script -> 가져다 쓸 경우 **cdn 쓰는 것을 따라 써야 한다. ( 버전도 같게 )**

<link href="https://cdnjs.cloudflare.com/ajax/libs/c3/0.7.3/c3.min.css" rel="stylesheet" />

<script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>

<script src = "https://cdnjs.cloudflare.com/ajax/libs/d3/5.9.7/d3.min.js"></script>

<script src = "https://cdnjs.cloudflare.com/ajax/libs/c3/0.7.3/c3.min.js"></script>

참고 : templates/member/js\_chart.html

**summernote – textarea 전용** <https://summernote.org/getting-started/#compiled-css-js>

Include js/css

여기 코드를 그대로 **copy&paste ( 기존 bootstrap이 있다면 지워야 함, 버전때문에 )**

$(document).ready(**function**() {

$('#summernote').summernote();

});

기본적인 형태, **summernote라는 id로 된 것**을 바꿔준다.

**추가로 아래에 placeholder나 height 예제 코드가 존재**

서머노트로 입력 시 html 태그 형식으로 DB에 저장된다.

- So, 그대로 읽으면 html 형태가 나온다.

- 그래도 찍히는 것이 아니라 **해독하게 만들어줘야 한다.**

- **autoescape** (자동으로 string 출력) 하는 것을 **off로** 해당 파트를 **열고 닫아준다.**

{% autoescape off %}

{{ one.2 | linebreaksbr }}

{% endautoescape %}

**SQL 문은 .md 파일 확인**

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

**<< \*\* why pandas? \*\* >>**

**일반 list**는 2차원이 아니라 1행으로 쭉 이어진 형태

- 특정 열의 데이터를 수정하기가 어렵다.

- 실제 2차원으로 이루어진 데이터타입이 필요하다. So, pandas > dataframe

pandas > **dataframe**

- 실제 2차원을 배치되어 있어 처리가 간편

- 열 별로 데이터 수정, 결측치 바로 수정 등 기능이 좋다.

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

**<< \*\* plt 쓰기 \*\* >>**

# graph

import matplotlib.pyplot as plt

import io # byte로 변환

import base64 # byte배열을 base64로 변경함.

from matplotlib import font\_manager, rc # 한글 폰트 적용

맨 위를 제외하고는 그래프를 HTML로 넘겨주기 위함.

# 폰트 읽기

font\_name = font\_manager.FontProperties\

    (fname="c:/Windows/Fonts/malgun.ttf").get\_name()

# 폰트 적용

rc('font', family=font\_name)

한글 폰트를 위함, 경로는 절대경로로 정확히 줘야 함, **ttf인지 ttc인지**

x = ['a', 'b', 'c']

y = [1, 2, 3]

plt.bar(x,y)

plt.title("test title")

plt.xlabel("x label name")

plt.ylabel("y label name")

# plt.show() # 표시, web에서는 출력 안됨

plt.draw() # 안보이게 그림을 캡처

img = io.BytesIO() # img에 byte배열로 보관

plt.savefig(img, format="png") # png파일 포맷으로 저장

img\_url = base64.b64encode(img.getvalue()).decode()

1. x에 해당 축 네이밍을, y에 데이터를 준비하고

2. plt의 bar 등등 여러 표 함수를 사용하여 title 등을 네이밍 해준다.

3. **plt.show()** -> python 내에서 새로이 창을 띄워서 보여준다.

- **plt.draw()** -> 그래프를 내부에 캡처(저장) – 변환해서 image로 출력 가능

4. 아래 3줄로 값을 img\_url로 만들고

return render(request,"~~~.html", {"graph": "data:;base64,{}".format(img\_url)})

5. 위의 **“data:;base,{}”.format(img\_url)** 여기서 이미지 파일로 변경

6. 여러 개 graph를 하려면 위에서 **리스트로 변환**해서 넘기면 된다.

**\*\* plt.close()를 안해주면 계속 쌓인다. 같은 그래프에 열이 추가된다.**

끝