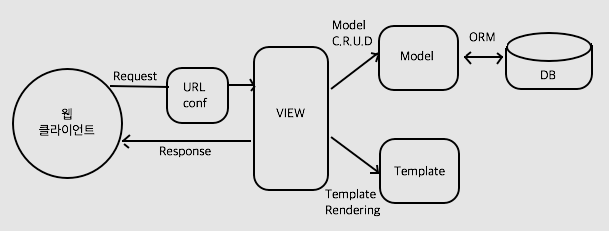
+ Django   
 : 파이선으로 만들어진 무료 오픈소스 웹 어플리케이션 프레임워크  
 : 쉽고 빠르게 웹 어플리케이션을 개발할 수 있도록 돕는 웹 프레임워크  
 (Full Stack Framework)  
  
cf) Flask : micro framework  
  
> 장고 프로젝트

- django-admin startproject 프로젝트명

- 프로젝트에는 여러 개의 app을 만들 수 있음

> MTV (Model Template View) 개발 방식



**- M (Model : models.py) : DBMS 내용** : 장고에서 하나의 모델은 하나의 클래스로 나타낸다.  
 클래스 내부에 데이터의 이름과 데이터 형태를 정의

[자료형]

1) CharField

문자열 정의

옵션 (max\_length : 문자열 최대 길이) | max\_lengh를 적지 않으면 None처리 (무제한)

2) DateField

날짜 양식에 맞게 저장되는 데이터 타입

default - python datatime 라이브러리 기준

3) EmailField

이메일 형식을 가지는 데이터 타입

EmailValidator 를 통해 입력되는 문자열이 이메일 형식인지를 자동으로 체크

4) FileField : 파일을 저장할 수 있는 데이터 타입, 실질적으로 저장되는 것은 파일의 이름

5) TextField : CharField와 거의 같음. 글자 수 제한이 더 크다.

6) IntegerField : 숫자를 저장하는 데이터 타입

7) BooleanField : 참/거짓을 저장하는 데이터 타입

[sqlite]

: PRAGAMA  
 컬럼순서 | 이름 | 자료형 | notnull여부 (1/0) | pk여부 (1/0)

# 1이 True, 0이 false

**- T (Template : templates 경로 생성해서 - HTML) -> 사용자에게 보여줄 화면 관련**



settings.py에 TEMPATES에 관련 정보 있음

[Django 템플릿 엔진]

: DTL(Django Template Language) – 장고의 코어 템플릿 엔진

[Django 템플릿 문법]

ᆞ변수 : {{ 변수 }} # 변수명은 영문자, \_(under bar)로 구성. 공백, 특수문자 사용 X

. 은 변수의 속성에 접근할 때 사용.

. 조회 순서 – 딕셔너리조회 => 속성조회 => 메소드호출 => 리스트의 인덱스 조회

필터 사용 가능. ( | 뒤가 필터) 60개쯤 됨

ex) {{ name | lower }} – name 변수에 lower를 적용시켜서 결과를 소문자로 출력

{{ text | escape | linebreaks }} – 텍스트를 escape하고, 개행함

ᆞ태그 : 어떤 태그는 텍스트를 생성, 어떤 태그는 로직 수행 등 기능이 있는 태그

: 기본형태 {% tag %} 태그 내용 {% endtag %}

ⅰ) for 반복문

<ul>

{% for user in users: %}

<li> {{ user }} </li>

{% endfor %}

</ul>

ⅱ) if 조건문

{% if users %}

users : {{ users | length }}

{% else %}

user가 없어요

{% endif %}

ⅲ) block(막기), extends(탬플릿상속), include(탬플릿, 변수 등 가져오기), …

**- V (View : views.py) -> Logic 관련**

> 주요 파일

- setting.py : 설정파일 - 사용자 정의 파일을 만들면 꼭 여기에서 인식 시켜줘야 한다.

- urls.py : URL conf (URL과 view를 mapping)

- models.py : DBMS 관련 (default : ORM 방식)’

: 여기에 정의된 모델(데이터베이스 테이블)은 migrate 명령을 통해 실제 데이터베이스에 적용  
: migrate 명령  
 1) 모델을 변경 - 필요한 테이블, 컬럼, 데이터타입등을 작성  
 2) python manage.py makemigrations <앱명> - 일종의 설계도  
 3) python manage.py migration <앱명>  
  
[Django 모델 API] - 데이터를 추가/갱신/조회 --> 함수를 쓰면 알아서 SQL동작  
1) insert : 객체 생성 후에 save() 함수를 이용하여 새 객체를 insert  
2) select : Django 모델 클래스는 objects라는 Manager 객체를 자동으로 추가함.  
 이 Manager 객체를 통해 여러 기능을 사용 할 수 있음.  
 - all() : table data 전체  
 - get() : 하나의 row를 가져옴  
 ex) Promary key로 찾아올 경우(*ToDo.objects.get(pk = 1))* - filter() : 특정 조건을 사용하여 row를 가져오는 메소드  
 - exclude() : 특정 조건을 제외한 나머지 row를 가져오는 메소드

- count() : 데이터 개수(row수)를 세어주는 메소드

- order\_by() : 데이터를 특정 컬럼에 따라 정령함. 인수로 컬럼을 사용

ex) ToDo.objects.order\_by(‘id’) # default : 오름차순

ToDo.objects.order\_by(‘-id’) # - : 내림차순

ToDo.objects.order\_by(‘-id’, ‘content’) # n차 정렬도 가능

- distinct() : 중복 제거하는 메소드

- first() : 여러 개의 데이터 중에서 처음에 있는 row를 가져오는 메소드

ex) ToDo.objects.order\_by(‘content’).first()

- last() : 여러 개의 데이터 중에서 마지막 row를 가져오는 메소드

- 이 메소드들은 쿼리 표현식(QuerySet)으로 리턴하므로 함수를 체이닝 방식으로 사용 가능

3) update : 수정할 row 객체를 얻은 후 변경할 필드를 수정. 기본적으로 업데이트X, 덮어씌우기

수정은 save() 함수로 갱신

4) delete : 삭제할 row 객체를 얻은 후 delete()로 삭제

- views.py : 화면 처리 로직, business 로직

- admin.py : app admin 처리

Terminal과 Pycharm또는 SublimeText로 할 예정

Django 설치

C:\WINDOWS\system32>cd \

C:\>cd python\_workspace

C:\python\_workspace>pip install django

설치 확인

C:\python\_workspace>python

>>> import django

>>> print(django.get\_version())

3.2 🡪 확인

ToDoList 프로젝트 생성

C:\python\_workspace>django-admin startproject ToDoList

C:\python\_workspace>cd ToDoList

C:\python\_workspace\ToDoList>dir

C 드라이브의 볼륨에는 이름이 없습니다.

볼륨 일련 번호: 02DF-1B02

C:\python\_workspace\ToDoList 디렉터리

2021-04-23 오전 11:40 <DIR> .

2021-04-23 오전 11:40 <DIR> ..

2021-04-23 오전 11:40 686 manage.py 🡪 이게 관리 파일임

2021-04-23 오전 11:40 <DIR> ToDoList

1개 파일 686 바이트

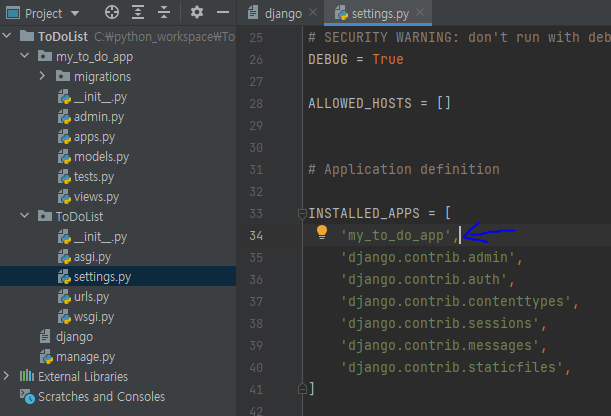
3개 디렉터리 57,373,626,368 바이트 남음

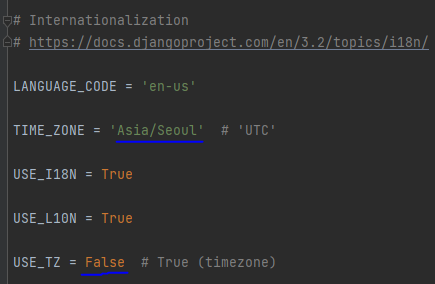
C:\python\_workspace\ToDoList>python manage.py startapp my\_to\_do\_app

* 경로에 파일 생성됨

파이참에서 C\python\_workspace\ToDoList open하기

setting.py 에서 수정할 것, 사용자 정의 추가, 타임존 수정

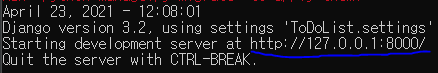


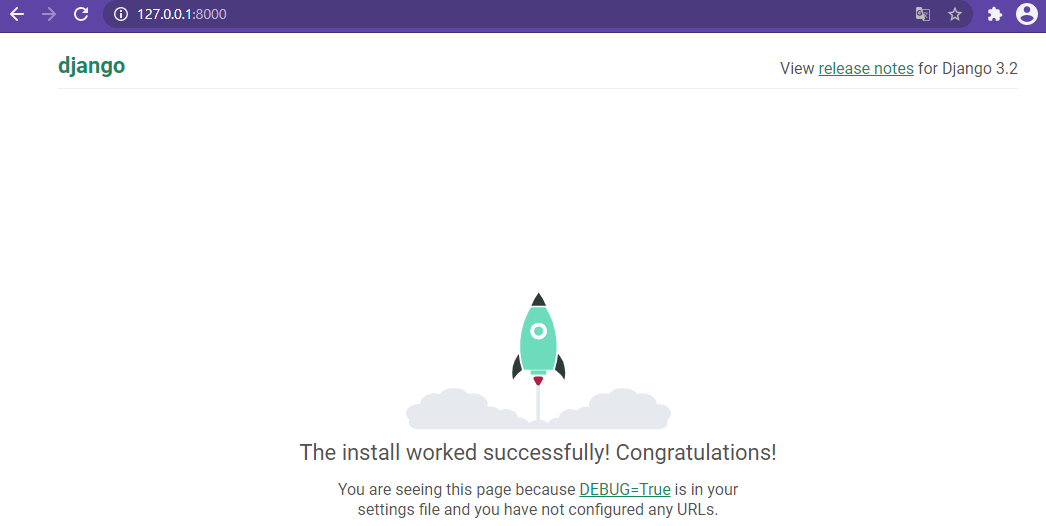


자바는 WAS를 끌어왔으나 Django는 내부에 있음

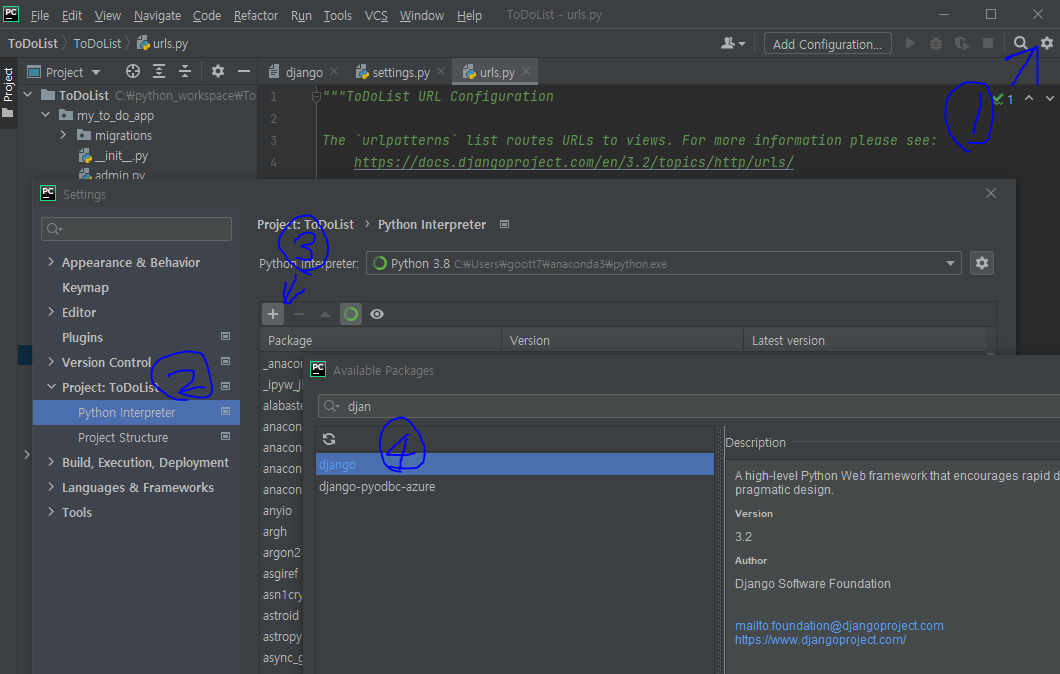
C:\python\_workspace\ToDoList>python manage.py runserver

중간에 빨간줄은 뭐를 안쓰고 있다고 뜨는 것 상관 x

실행주소 확인

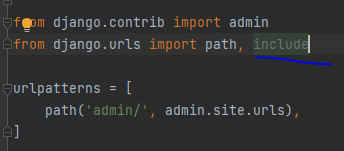


Pycharm에서 다음의 설정이 필요할 수 있음

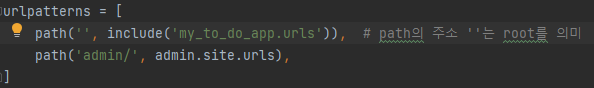


urls.py url 설정, path 추가하기

include 모듈 추가

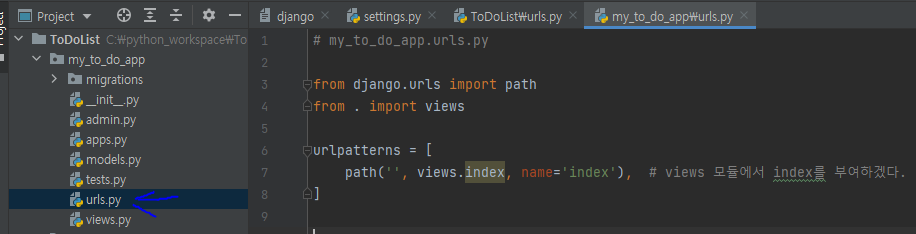


path 추가

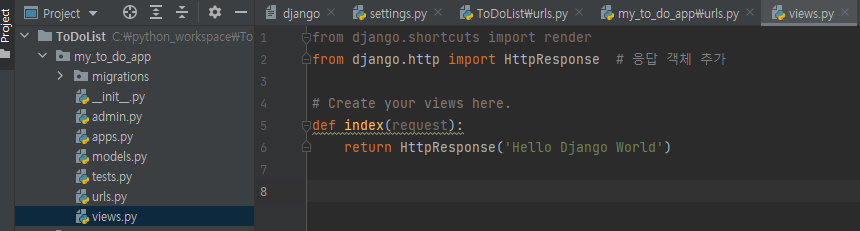


my\_to\_do\_app 사용자 설정 앱

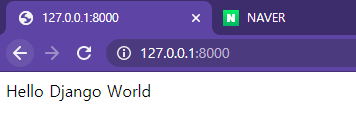
urls.py - 내 앱에 관련된 내 주소를 넣을 파일 생성



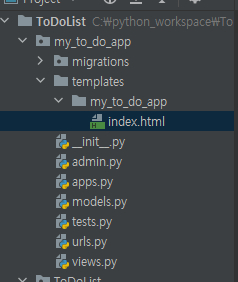
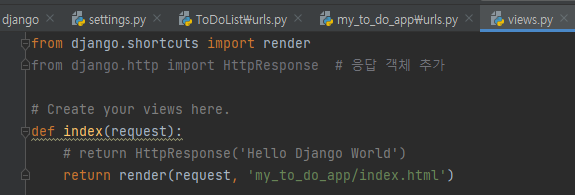
views.py – 응답 객체 추가하고 출력하도록 함수 정의



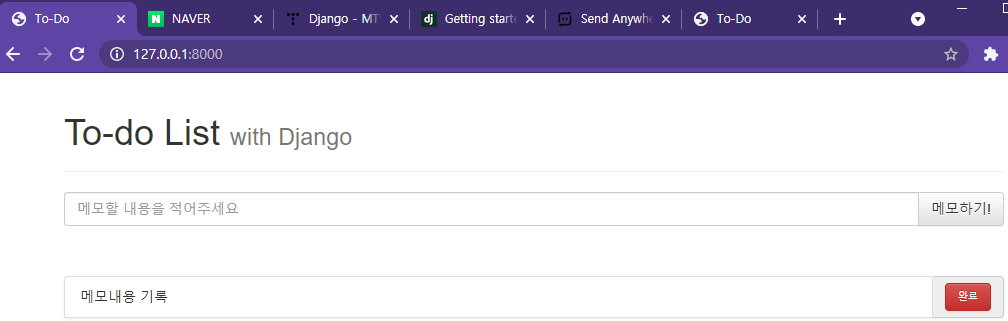
실행시 잘 뜸



view를 위한 경로 설정, 관례적으로 앱과 같은 이름의 경로를 templates 경로에 또 만듬

공유 받은 index 파일을 넣어두고 이걸 띄우기 위해 views.py 수정

 동작 확인

[0423 오후]

Models.py, tests.py, views.py 등 추가된듯 함

SQLite 밑에 다루는 거 간단

C:\python\_workspace\ToDoList>python manage.py dbshell

SQLite version 3.35.4 2021-04-02 15:20:15

Enter ".help" for usage hints.

🡺 기본으로 붙어있는 경량의 DBMS

sqlite> .tables

auth\_group django\_admin\_log

auth\_group\_permissions django\_content\_type

auth\_permission django\_migrations

auth\_user django\_session

auth\_user\_groups my\_to\_do\_app\_todo

auth\_user\_user\_permissions

sqlite>

🡺 기본으로 존재하는 테이블들, migration하고나면 보임

sqlite> select \* from my\_to\_do\_app\_todo;

1|불금엔 칼퇴!

3|장보기

4|힐링하기

5|시험준비 열심히 하기

sqlite>



🡺 DB 정보 있는 것 확인

sqlite> PRAGMA table\_info(my\_to\_do\_app\_todo);

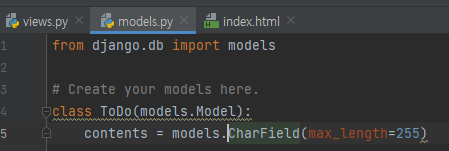
0|id|integer|1||1

1|contents|varchar(255)|1||0

sqlite>

🡺 내 테이블 정보 확인

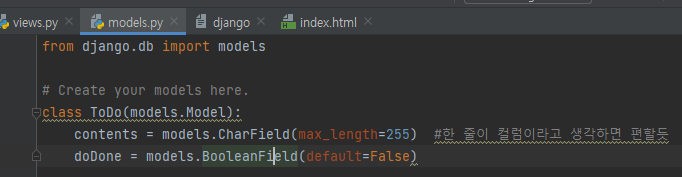
이 DB 관련 작업은 MTV의 ORM이 이미 있기 때문에 Model을 만들고 이를 DB에 넣는 방식



Models.py의 클래스가 하나의 테이블로 자동 생성 됨

my\_to\_do\_app\_todo 🡺 앱명\_클래스명 으로 생성되는 듯 함

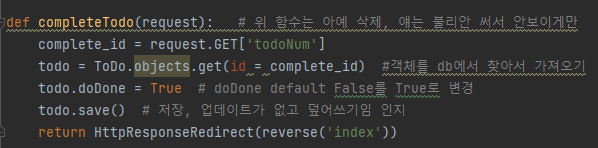
이제 완료 누르면 삭제 말고 안보이게만 처리하는 것 수정



Models.py 수정 🡪 DB 수정

python manage.py makemigrations

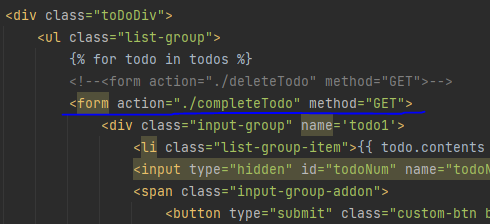
python manage.py migrate



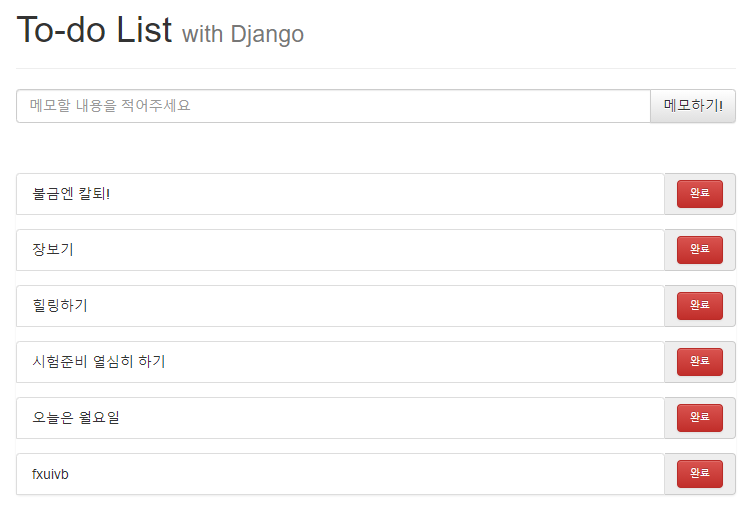
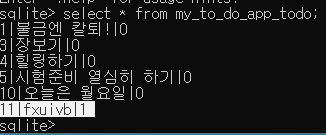
Views.py 에 추가



Urls.py에 접근 추가

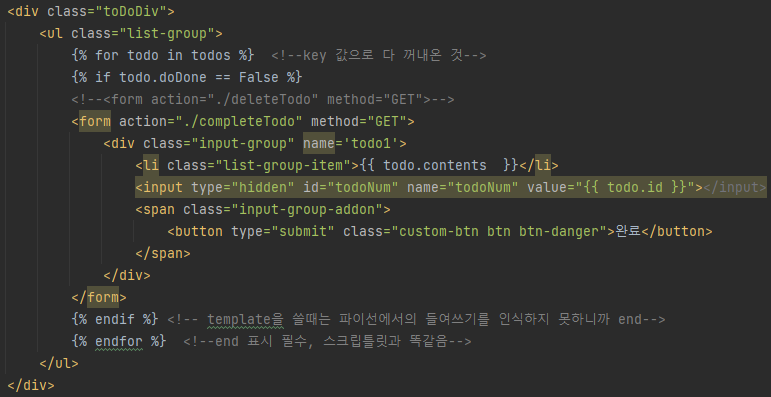


Form action 주소 변경

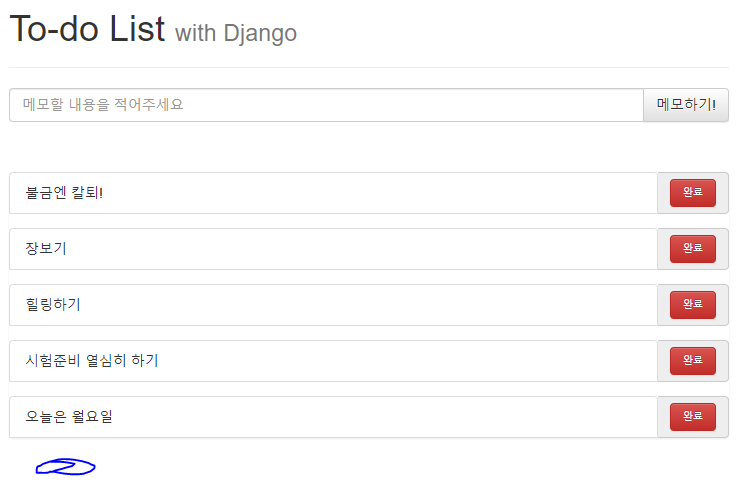
 

항목 하나 추가하고 완료 누르면 True 되는것 확인

이제 완료안된 놈만 출력되도록 처리하면 됨



Index.html에서 if문을 통해 걸러냄



todo.doDone == False인 항목들만 출력됨을 확인

//////////////////////ToDoList 프로젝트 끝////////////////////////////

/// 회원가입, 로그인, 정보변경, excel추출 및 데이터 처리 등 새 프로젝트

/// 이메일을 통한 인증(?)

구성도

/signup /email/send /verifyCode /verify

회원가입 – 이메일 인증 – 인증코드 입력 – 인증코드 확인/회원가입

> signup model

이름 이메일 비번 인증여부

/signin /signin/login / /calculate /result

로그인 – 로그인 성공 – 메인화면 – 엑셀파일업로드/분석 – 결과확인

/loginFail /signError

– 로그인 실패 – 실패화면

excel\_cal\_django파일 받아놓기 🡪 위의 기능들에 쓸 화면 파일임

C:\python\_workspace>pip freeze 🡪 뭐가 설치되어 있는지

pandas @ file:///C:/ci/pandas\_1618365634936/work 확인

C:\python\_workspace>pip install pandas 🡪 없으면 설치

C:\python\_workspace>django-admin startproject ExcelCalculate 🡪 프로젝트 생성

C:\python\_workspace>cd ExcelCalculate

App을 3개 만들것임

C:\python\_workspace\ExcelCalculate>python manage.py startapp main 🡪 메인

C:\python\_workspace\ExcelCalculate>python manage.py startapp sendEmail 🡪 이메일 인증

C:\python\_workspace\ExcelCalculate>python manage.py startapp calculate 🡪 계산

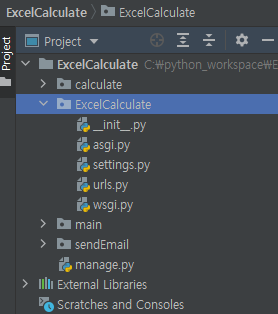
Dir로 디렉토리 생성 확인

2021-04-26 오후 12:04 <DIR> calculate

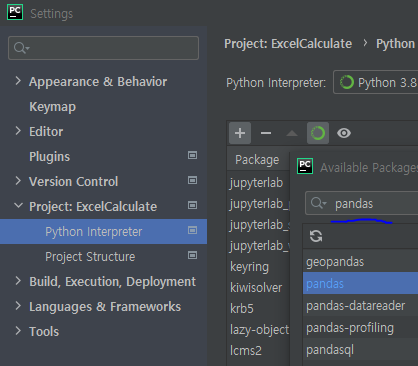
2021-04-26 오후 12:02 <DIR> ExcelCalculate

2021-04-26 오후 12:03 <DIR> main

PyCharm에 프로젝트 열어두기

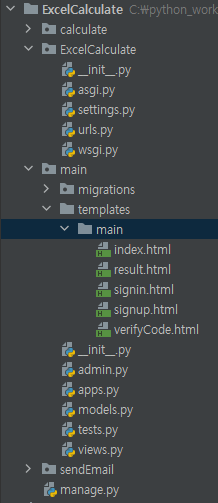


Pandas, Django 유무 확인하고 설치



ExcelCalculate의 settings.py 시간대 수정

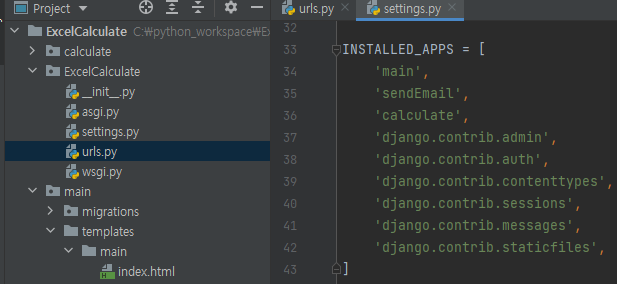
LANGUAGE\_CODE = 'en-us'  
TIME\_ZONE = 'Asia/Seoul' # 'UTC'  
USE\_I18N = True  
USE\_L10N = True  
USE\_TZ = False # True



다운 받아둔 5개 HTML 파일 main\templates\main에 추가

App 아래에 templates 생성 후 밑에 app명으로 경로 추가 생성 해야 인식 잘함

추가한 3개의 앱을 ExcelCalculate(프로젝트)의 setting.py INSTALLED\_APPS에 추가해야 함



3개 앱 ExcelCalculate의 urls.py에 경로 추가

# ExcelCalculate urls.py  
  
from django.contrib import admin  
from django.urls import path, include # include추가  
  
urlpatterns = [  
 path('', include('main.urls'), name='main'),  
 path('calculate/', include('calculate.urls'), name='calculate'),  
 path('email/', include('sendEmail.urls'), name='email'),  
 path('admin/', admin.site.urls),  
]

sendEmail부터 작업, urls.py 추가

# sendEmail urls.py  
  
from django.urls import path  
from . import views  
  
urlpatterns = [  
 path('send', views.send, name='email\_send'), # 이메일 인증 /email/send  
]

sendEmail views.py에 위에서 할당한 동작인 views.send에 대한 작업 수행

from django.shortcuts import render  
from django.http import HttpResponse # 응답 객체  
  
# sendEmail views.py  
  
def send(receiverEmail):  
 return HttpResponse('sendEmail views - sendFunction()')

일단은 그냥 돌아가게만 해두고 기능은 없음, 페이지가 보이는 지만 확인

calculate urls.py 추가

# calculate urls.py  
  
from django.urls import path  
from . import views  
  
urlpatterns = [  
 path('', views.calculate, name='calculate\_cal'), # 엑셀 업로드, 분석 /calculate  
]

calculate views.py 작업

from django.shortcuts import render  
from django.http import HttpResponse # 응답 객체  
  
# calculate views.py  
  
def calculate(request): #요청만 열어둠  
 return HttpResponse('calculate views - calculate function()')

main urls.py 추가

# main urls.py  
  
from django.urls import path  
from . import views  
  
urlpatterns = [  
 path('', views.index, name='main\_index'), # 메인 페이지 / 경로  
 path('signup', views.signup, name='main\_signup'), # /signup  
 path('signin', views.signin, name='main\_signin'), # /signin  
 path('verifyCode', views.verifyCode, name='main\_verifyCode'), # /verifyCode  
 path('verify', views.verify, name='main\_verify'), # /verify  
 path('result', views.result, name='main\_result'), # /result  
]

main views.py 작업

from django.shortcuts import render, redirect # redirect 추가  
  
# main views.py  
  
def index(request):  
 return render(request, 'main/index.html') # templates를 바로 찾아온다  
 # 앱 자신의 템플릿임을 확인키위해 main이름또 쓰는 것  
def signup(request):  
 return render(request, 'main/signup.html')  
  
def signin(request):  
 return render(request, 'main/signin.html')  
  
def verifyCode(request):  
 return render(request, 'main/verifyCode.html')  
  
def verify(request):  
 return redirect('main\_index') # 뷰는 없고 기능만 있으면 되는데, 아직은 메인으로 바로 보냄  
  
def result(request):  
 return render(request, 'main/result.html')

python manage.py runserver 🡪 일단 돌려보면 준비한 주소 작동들 다 확인 가능

- /, /calculate, /email/send, /signup, /signin, /verifyCode, /verify(얘는 메인가게 되어있음), /result

Admin 생성하기

C:\python\_workspace\ExcelCalculate>python manage.py makemigrations

C:\python\_workspace\ExcelCalculate>python manage.py migrate

C:\python\_workspace\ExcelCalculate>python manage.py createsuperuser

Username (leave blank to use 'goott7'): root

Email address: 🡪 엔터쳐서 그냥 넘김

Password: 🡪 12341234로 함

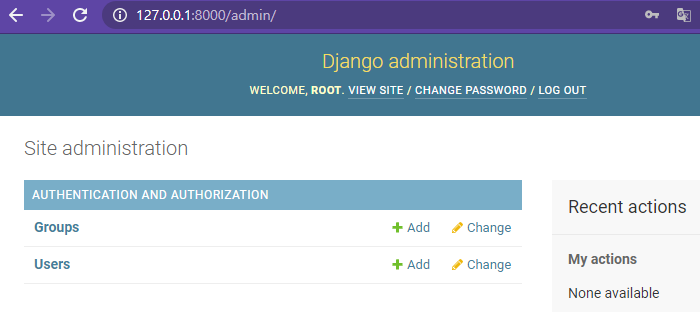
Password (again):

This password is too common.

This password is entirely numeric.

Bypass password validation and create user anyway? [y/N]: y

Superuser created successfully.

 로그인 후

**Signup 관련 모델(회원 테이블), 회원가입까지**

main의 models.py

from django.db import models  
  
# main models.py  
  
class User(models.Model):  
 user\_name = models.CharField(max\_length=20) # 20자 제한  
 user\_email = models.EmailField(unique=True)  
 user\_password = models.CharField(max\_length=100)  
 user\_validate = models.BooleanField(default=False)

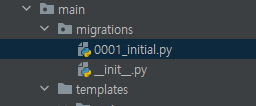
🡪 migration이 되면 ORM이 DB에 등록해줌

C:\python\_workspace\ExcelCalculate>python manage.py makemigrations

Migrations for 'main':

main\migrations\0001\_initial.py

- Create model User

생성 확인



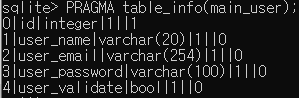
C:\python\_workspace\ExcelCalculate>python manage.py migrate

Operations to perform:

Apply all migrations: admin, auth, contenttypes, main, sessions

Running migrations:

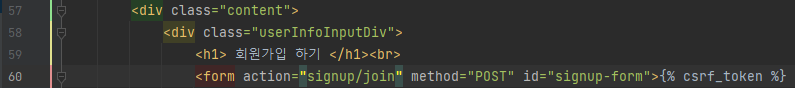
Applying main.0001\_initial... OK

 python manage.py dbshell에서 테이블 생성 확인

회원 가입 부분 DB 처리를 위한 url 설정

signup 페이지에서 join 기능 수행하도록 주소 하나 할당

signup.html의 60줄 form 태그 수정



main urls.py에 다음 path 추가

path('signup/join', views.join, name='main\_join'),

main views.py

def join(request):  
 print('request:', request)  
 #return redirect('main\_verifyCode') # 요청 출력하고 이메일 코드확인 페이지까지  
 return redirect('main\_index')

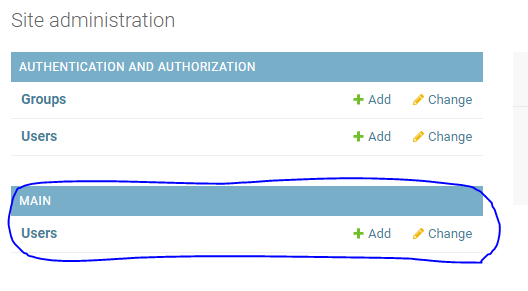
서버에서 request 확인



main admin.py

from django.contrib import admin  
from .models import \*  
  
# main admin.py  
  
admin.site.register(User)

만들어둔 Users가 admin 페이지에 있음



※ redirect에서 오류가 난다면 url.py의 name과 정확히 일치하는지 확인 필요

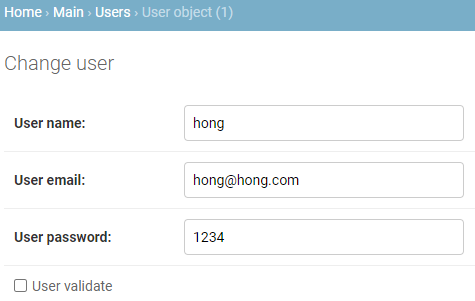
main views.py 수정, 유저 가입 관련

from .models import \* # 추가해서 모델로User 인식하게

def join(request):  
 #print('request:', request)  
 name = request.POST['signupName'] # signup.html에서 주는 이름 그대로  
 email = request.POST['signupEmail']  
 pw = request.POST['signupPW']  
 user = User(user\_name=name, user\_email=email, user\_password=pw) # 모델 데이터 넣기  
 user.save() # db에 넣기  
 return redirect('main\_verifyCode') # 요청 출력하고 이메일 코드확인 페이지까지

유저 가입 후 데이터 확인 가능, sqlite에서, admin에서 모두 가능





**이메일 인증 관련**

main views.py 수정

from random import \* #랜덤 추가

def join(request):  
 #print('request:', request)  
 name = request.POST['signupName'] # signup.html에서 주는 이름 그대로  
 email = request.POST['signupEmail']  
 pw = request.POST['signupPW']  
 user = User(user\_name=name, user\_email=email, user\_password=pw) # 모델 데이터 넣기  
 user.save() # db에 넣기  
   
 # 인증 코드 관련 4자리 정수로 생성 - 랜덤  
 code = randint(1000, 9999)  
 # 쿠키에 저장  
 response = redirect('main\_verifyCode')  
 response.set\_cookie('code', code) # 쿠키는 이름이 필수  
 response.set\_cookie('user\_id', user.id)  
   
 # 인증 코드 - 이메일로 전송  
 #return redirect('main\_verifyCode') # 요청 출력하고 이메일 코드확인 페이지까지  
 return response

이메일 보내기, SMTP(gmail이 naver보다 용이)

이메일 보낼 때 html짜서 css넣고 보내도 그대로 가서 잘 출력된다

🡺 html 인식 시키기 위해 sendEmail밑 templates\sendEmail 경로 생성

위 경로에 email\_format.html 생성

<h1>ExcelCalculate Project 회원가입 인증 코드</h1>  
<h2>다음의 인증 코드를 입력해주세요</h2>  
  
<h3>{{ verifyCode }}</h3> <!--view에서 template으로 들어온 인증용 데이터-->

main views.py 수정2(위 코드에서 인증 코드 – 이메일로 전송 부분 수정)

from sendEmail.views import \* #다른 앱이므로 import 해줌

# 인증 코드 - 이메일로 전송  
# ==> view에서 template으로 나감 다른 앱이므로 import 필요  
send\_result = send(email, code) # receiverEmail, verifyCode  
if send\_result: # 잘 보내지면 True 리턴  
 return response  
else:  
 return HttpResponse('이메일 발송에 실패했습니다.')

뒤에 있던 return response는 주석처리

sendEmail app의 views.py 수정

from django.core.mail import send\_mail, EmailMessage #email  
from django.template.loader import render\_to\_string #로더, 렌더링

def send(receiverEmail, verifyCode): #이메일과 코드 받아서 사용  
 try:  
 content = {'verifyCode':verifyCode} #dict 권장  
 msg\_html = render\_to\_string('sendEmail/email\_format.html', content)  
 msg = EmailMessage(subject='인증코드발송메일', body=msg\_html, from\_email='hyun950609@gmail.com', bcc=[receiverEmail])  
 #제목, 내용바디, 보낸사람, 받는사람 순  
 msg.content\_subtype='html' #html로 보냄 명시

msg.send()  
 return True  
 except:  
 return False

ExcelCalculate 프로젝트의 settings.py 비어있는 공간에 이메일 관련 사항 추가하기

# Email  
EMAIL\_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'  
EMAIL\_HOST = 'smtp.gmail.com'  
EMAIL\_USE\_TLS = True  
EMAIL\_PORT = 587  
EMAIL\_HOST\_USER = 'ajrwk1995@gmail.com'  
EMAIL\_HOST\_PASSWORD = 'qazqw12!@'

DEFAULT\_FROM\_EMAIL = EMAIL\_HOST\_USER

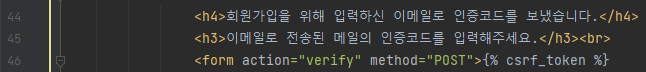
이메일 보내지는지 확인하기





유저 join할 때 랜덤 코드를 생성하고 이메일 보낸 후, 쿠키에 코드랑 아이디를 넣어서 main\_verifyCode로 넘기니까 verifyCode.html 수정 필요

verifyCode에서 코드를 입력받으면 verify로 가서 비교할 테니까 form action 변경



# main urls.py

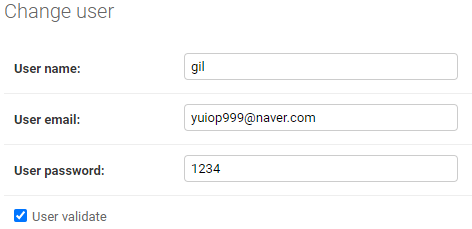
path('verify', views.verify, name='main\_verify'), # /verify

path에 의해 main의 views 추가작업

코드 입력 정상처리되면 쿠키 삭제, 세션에 정보를 넣기까지 함

def verify(request):  
 user\_code = request.POST['verifyCode'] # verifyCode.html에서 저 이름으로 넘어올 것임  
 cookie\_code = request.COOKIES.get('code') # 쿠키에 가입때 넣어둔 랜덤 코드 가져오기  
  
 if user\_code == cookie\_code:  
 user = User.objects.get(id=request.COOKIES.get('user\_id'))  
 user.user\_validate = 1  
 user.save()  
 response = redirect('main\_index')  
 response.delete\_cookie('code')  
 response.delete\_cookie('user\_id')  
 # response.set\_cookie('user', user) # 이제 가입 처리 후, 로그인 정보를 쿠키말고 세션처리하기로  
  
 request.session['user\_email'] = user.user\_email # 회원 가입과 동시에 세션에 넣기  
 request.session['user\_name'] = user.user\_name  
  
 return response  
 else:  
 redirect('main\_verifyCode')

가입 완료되면 validate까지 1됨 확인 가능



////회원가입 끝////

**로그아웃 기능**

index.html의 로그아웃 버튼

<h1>Excel Calculate <small>with Django</small><a href="logout"

위 코드에 의해 path 추가해야됨 main의 urls.py

path('logout', views.logout, name='main\_logout')

main의 views.py에 logout 기능 생성

def logout(request):  
 del request.session['user\_email'] # 세션에서 삭제  
 del request.session['user\_name']  
 return redirect('main\_signin')

**로그인 기능**

signin.html에서 수정

<h3>안녕하세요. 로그인을 진행해주세요.</h3><br>  
<form action="signin/login" method="POST">{% csrf\_token %}

main urls.py

path('signin/login', views.login, name='main\_join'),

main views.py

def login(request):  
 loginEmail = request.POST['loginEmail'] # signin.html에서 name대로 가져옴  
 loginPw = request.POST['loginPW']  
  
 # 회원등록여부 체크  
 try:  
 user = User.objects.get(user\_email=loginEmail) # 없으면 에러날거니까 에러처리로  
 except Exception as e:  
 print('e:', e)  
 return redirect('main\_loginFail')  
  
 # 이메일 체크해서 있으면 넘어오는 곳임, 없으면 에러처리로 실패화면이동함  
 # 비밀번호 일치 여부  
 if user.user\_password == loginPw:  
 request.session['user\_name'] = user.user\_name  
 request.session['user\_email'] = user.user\_email  
 return redirect('main\_index')  
 else:  
 return redirect('main\_loginFail')

로그인 실패 관련 path main urls.py

path('loginFail', views.loginFail, name='main\_loginFail')

main views.py

def loginFail(request):  
 return render(request, 'main/loginFail.html')

loginFail.html생성하고 signin.html의 41번째 줄까지 복사해와서 수정, 로그인실패 추가, 푸터 추가

////로그인 기능 끝////

**메인 화면에서 로그인 여부를 체크하고 아닐경우 로그인으로 가도록 처리**

main views.py 수정

def index(request):  
 # session의 user\_name 세션에 존재 하는지  
 if 'user\_name' in request.session.keys(): # dict의 key로 찾아오는 것  
 return render(request, 'main/index.html') # templates를 바로 찾아온다  
 # 앱 자신의 템플릿임을 확인키위해 main이름또 쓰는 것  
 else:  
 return redirect('main\_signin')

이름 찾고 세션에 있으면 인덱스, 없으면 로그인 실행하도록

**파일 업로드 부**

index.html 수정

<form action="calculate/" method="POST" enctype="multipart/form-data"> {% csrf\_token %}  
 <div class="input-group">  
 하단 버튼을 통해 파일을 업로드 해주세요.(.xls 확장자의 파일만 가능합니다.)<br>

파일 업로드는 반드시 POST, 파일은 쪼개서 가므로 GET못함

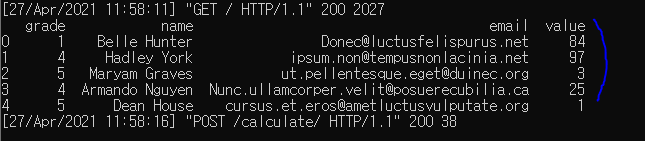
받아온 엑셀을 pandas로 데이터 분석할거임

calculate views.py

import pandas as pd # pandas 추가

def calculate(request): #요청만 열어둠  
 file = request.FILES['fileInput'] # 파일 자체를 업로드한게 아니고 정보를 담아둔 것  
 # 파일은 DB에 올리는거랑 서버에 올리는거랑 다르다.  
  
 # print(file)  
  
 # 받아있는 데이터가 테이블 형태로 잘 정리된 엑셀 파일임  
 # pandas 사용해서 데이터 간단 분석  
 df = pd.read\_excel(file, sheet\_name='Sheet1', header=0)  
 # sheet\_name:엑셀파일 시트명, 가진 파일에는 Sheet1임  
 # header  
 print(df.head(5)) # 적당히 5개 잘라서 출력  
  
 return HttpResponse('calculate views - calculate function()')

파일 상위 5개 잘 읽어옴



이렇게 판다스 사용 해야되는데 오류가 있어서 빅데이터파트부터 진행함

다음장에 수정

def calculate(request): #요청만 열어둠  
 file = request.FILES['fileInput'] # 파일 자체를 업로드한게 아니고 정보를 담아둔 것  
 # 파일은 DB에 올리는거랑 서버에 올리는거랑 다르다.  
  
 # print(file)  
  
 # 받아있는 데이터가 테이블 형태로 잘 정리된 엑셀 파일임  
 # pandas 사용해서 데이터 간단 분석  
 df = pd.read\_excel(file, sheet\_name='Sheet1', header=0)  
 # sheet\_name:엑셀파일 시트명, 가진 파일에는 Sheet1임  
 # header  
 # print(df.head(5)) # 적당히 5개 잘라서 출력  
  
 # grade별 value의 최소값, 최대값, 평균값  
 # grade별 구분  
 grade\_dic = {}  
 total\_row\_num = len(df.index) # 길이 구함  
  
 # for i in range(total\_row\_num):  
 # data = df.loc[[i], ['grade'], ['name'], ['']]  
 #  
 # if not data['grade'] in grade\_dic.keys(): # 데이터는 컬럼 이름으로 찾아올 수 있음  
 # grade\_dic[data['grade']] = [data['value']] # 키(grade)마다 값(value) 넣어줌  
 # else:  
 # grade\_dic[data['grade']].append(data['value'])  
 #  
 # # grade별 value의 최소값, 최대값, 평균값  
 # grade\_calculate\_dic = {}  
 # for key in grade\_dic.keys():  
 # grade\_calculate\_dic[key] = {}  
 # grade\_calculate\_dic[key]['min'] = min(grade\_dic[key])  
 # grade\_calculate\_dic[key]['max'] = max(grade\_dic[key])  
 # grade\_calculate\_dic[key]['avg'] = float(sum(grade\_dic[key])/len(grade\_dic[key]))  
  
  
  
 return HttpResponse('calculate views - calculate function()')

수정

def calculate(request): #요청만 열어둠  
 file = request.FILES['fileInput'] # 파일 자체를 업로드한게 아니고 정보를 담아둔 것  
 # 파일은 DB에 올리는거랑 서버에 올리는거랑 다르다.  
  
 # print(file)  
  
 # 받아있는 데이터가 테이블 형태로 잘 정리된 엑셀 파일임  
 # pandas 사용해서 데이터 간단 분석  
 df = pd.read\_excel(file, sheet\_name='Sheet1', header=0)  
 # sheet\_name:엑셀파일 시트명, 가진 파일에는 Sheet1임  
 # header  
 # print(df.head(5)) # 적당히 5개 잘라서 출력  
  
 # grade별 value 의 최소값, 최대값, 평균값  
 # grade별 구분  
 grade\_dic = {}  
 total\_row\_num = len(df.index)  
  
 for i in range(total\_row\_num):  
 data = df.loc[i]  
  
 if not data['grade'] in grade\_dic.keys():  
 grade\_dic[data['grade']] = [data['value']]  
 else :  
 grade\_dic[data['grade']].append(data['value'])  
  
 print(grade\_dic.keys(), '--', grade\_dic.values())  
  
 # # grade별 value 의 최소값, 최대값, 평균값  
 grade\_calculate\_dic = {}  
 for key in grade\_dic.keys():  
 grade\_calculate\_dic[key] = {}  
 grade\_calculate\_dic[key]['min'] = min(grade\_dic[key])  
 grade\_calculate\_dic[key]['max'] = max(grade\_dic[key])  
 grade\_calculate\_dic[key]['avg'] = float(sum(grade\_dic[key])/len(grade\_dic[key]))  
  
 #print(list(grade\_calculate\_dic.keys()))  
  
 print('------------------------------')  
 grade\_list = list(grade\_calculate\_dic.keys())  
 for key in grade\_list:  
 print('--- grage --- ', key)  
 print('-- min : ', grade\_calculate\_dic[key]['min'], end=' ')  
 print('-- max : ', grade\_calculate\_dic[key]['max'], end=' ')  
 print('-- avg : ', grade\_calculate\_dic[key]['avg'], end='\n')  
  
 print('------------------------------------')  
  
 return HttpResponse('calculate views - calculate function()')

결과 다음과 같이 나옴

아래의 최대최소평균 전에 위의 데이터를 보면 파일을 읽으면서,

등급이 없으면 추가하고 있으면 있는 곳에 붙이는 등 딕셔너리에 추가하는 것 볼수있음

딕셔너리라서 순서 없이 나옴

------------------------------

--- grage --- 1

-- min : 4 -- max : 100 -- avg : 55.38461538461539

--- grage --- 4

-- min : 1 -- max : 97 -- avg : 43.85

--- grage --- 5

-- min : 1 -- max : 99 -- avg : 48.375

--- grage --- 2

-- min : 3 -- max : 95 -- avg : 51.421052631578945

--- grage --- 3

-- min : 0 -- max : 91 -- avg : 39.458333333333336

------------------------------------

[28/Apr/2021 09:49:42] "POST /calculate/ HTTP/1.1" 200 38

아래의 것을 추가해서 순서 적용해도 됨

grade\_list.sort() # 딕셔너리라서 key의 순서가 보장되지 않음

이메일 도메인 추출해서 개수가 몇 개인지, 2개 이상인거 출력까지

# 이메일 도메인별 인원 구하기  
domain\_dic = {}  
  
for i in range(total\_row\_num):  
 data = df.loc[i]  
 # pos = data['email'].find('@') # 데이터에서 email 중 @ 이후의 도메인 수집  
 # domain\_data = data['email'][pos+1:]  
 domain\_data = (data['email'].split('@'))[1] # 데이터에서 email 중 @ 이후의 도메인 수집  
  
 if not domain\_data in domain\_dic.keys():  
 domain\_dic[domain\_data] = 1  
 else:  
 domain\_dic[domain\_data] = domain\_dic[domain\_data] + 1  
  
 print(domain\_dic.keys(), '--', domain\_dic.values())  
  
for key in domain\_dic.keys(): # 도메인 딕셔너리 출력  
 print(key, ':', domain\_dic[key])  
  
for key in domain\_dic.keys(): # 2개 이상인 도메인 출력하기  
 if domain\_dic[key] > 1:  
 print(key)

위의 데이터 분석은 서버 로그로만 뜨지 웹페이지에 뜨지 않으니까 result.html에 보내도록

세션작업 추가함

grade\_calculate\_dic\_session = {}  
for key in grade\_list:  
 grade\_calculate\_dic\_session[int(key)] = {} # key가 int임을 명시  
 grade\_calculate\_dic\_session[int(key)]['min'] = float(grade\_calculate\_dic[key]['min'])  
 grade\_calculate\_dic\_session[int(key)]['max'] = float(grade\_calculate\_dic[key]['max'])  
 grade\_calculate\_dic\_session[int(key)]['avg'] = float(grade\_calculate\_dic[key]['avg'])  
  
request.session['grade\_calculate\_dic'] = grade\_calculate\_dic\_session  
request.session['email\_domain\_dic'] = domain\_dic  
  
# return HttpResponse('calculate views - calculate function()')  
return redirect('/result')

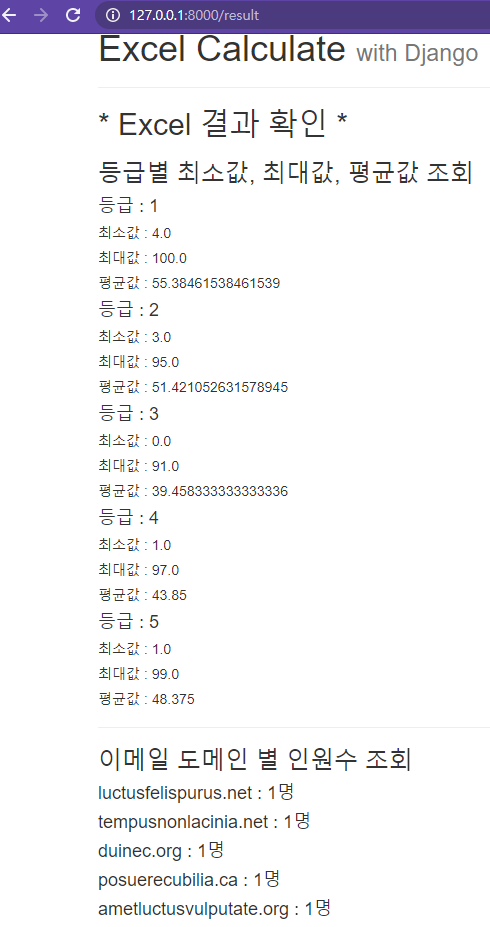
main / 밑의 /result이므로 main\views.py 수정

def result(request):  
 if 'user\_name' in request.session.keys(): #로그인 확인  
 content = {}  
 content['grade\_calculate\_dic'] = request.session['grade\_calculate\_dic'] #session에 넣어둔 분석 데이터 꺼내오기  
 content['email\_domain\_dic'] = request.session['email\_domain\_dic']  
 del request.session['grade\_calculate\_dic'] #session 정리  
 del request.session['email\_domain\_dic']  
 return render(request, 'main/result.html', content) # content넣어서 보냄  
 else:  
 return redirect('main\_signin')

result.html 40번째 줄부터 다음과 같이 수정

<div class='body'>  
 <div class="resultDiv">  
 <h2> \* Excel 결과 확인 \*</h2>  
 <h3>등급별 최소값, 최대값, 평균값 조회</h3>  
 {% for key, value in grade\_calculate\_dic.items %}  
 <h4>등급 : {{ key }}</h4>  
 <h5>최소값 : {{ value.min }}</h5> <!--value에 min이란 이름으로 넣었었음-->  
 <h5>최대값 : {{ value.max }}</h5>  
 <h5>평균값 : {{ value.avg }}</h5>  
 {% endfor %}  
 <hr>  
 <h3>이메일 도메인 별 인원수 조회</h3>  
 {% for key, value in email\_domain\_dic.items %}  
 <h4>{{ key }} : {{ value }}명</h4>  
 {% endfor %}  
   
 </div>  
 <hr>  
</div>

결과 확인



generic view라는 이런거 뷰 만들 때 편하게 해주는게 있다

admin에서 회원 비밀번호를 볼 수 없게 암호화 처리

main\views.py 의 join 수정

import hashlib # SHA256 방식으로 암호화 (내부모듈로 제공되는 것 중 가장 보안성 높음)

join 밑에 추가

# pw 암호화 (SHA 256)  
encode\_pw = pw.encode() # pw 엔코딩  
encrypted\_pw = hashlib.sha256(encode\_pw).hexdigest() # sha256으로 엔코딩된 pw 암호화하고 hex?  
print('encrypted\_pw', encrypted\_pw)

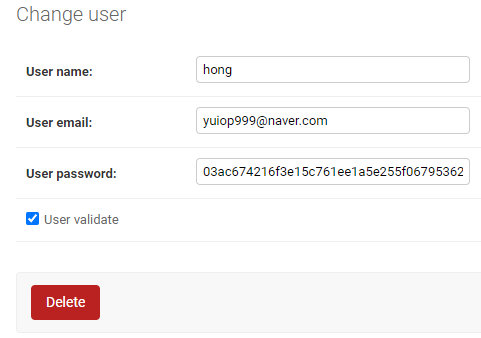
결과로 다음과 같이 로그 찍힘, 1234를 입력했을 때

encrypted\_pw 03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4

회원 db 입력부에 비번으로 넣어줌

user = User(user\_name=name, user\_email=email, user\_password=encrypted\_pw) # 모델 데이터 넣기

관리자도 정확한 비밀번호 모르게



회원가입 후 로그인때도 이 엔코딩 처리가 필요, DB와 매칭

main\views.py login함수 부분 수정

def login(request):  
 loginEmail = request.POST['loginEmail'] # signin.html에서 name대로 가져옴  
 loginPw = request.POST['loginPW']  
  
 # 회원등록여부 체크  
 try:  
 user = User.objects.get(user\_email=loginEmail) # 없으면 에러날거니까 에러처리로  
 except Exception as e:  
 print('e:', e)  
 return redirect('main\_loginFail')  
  
 # 이메일 체크해서 있으면 넘어오는 곳임, 없으면 에러처리로 실패화면이동함  
 # 비밀번호 일치 여부 (암호화 된 비밀번호)  
 encode\_pw = loginPw.encode()  
 encrypted\_pw = hashlib.sha256(encode\_pw).hexdigest()  
 # if user.user\_password == loginPw:  
 if user.user\_password == encrypted\_pw:  
 # 좌측 db의 엔코딩된 비밀번호, 우측 유저가 로그인을 위해 입력한 비번 엔코딩  
 request.session['user\_name'] = user.user\_name  
 request.session['user\_email'] = user.user\_email  
 return redirect('main\_index')  
 else:  
 return redirect('main\_loginFail')

사용자가 홈페이지에 입력한 비번을 또 sha256으로 엔코딩해서 디비와 비교함