2강 장고 초기 설정시 사용한 명령어 징리

django-admin startproject mysite . – 현재 path에 startproject 생성

django-admin startapp todo – 현재 path에 startapp 생성

python manage.py migrate –DB 생성

python manage.py createsuperuser – ID, PS admin으로 관리자 계정 생성

파일 설정 순서

Settings.py

* INSTALLED\_APPS = [] 안에 생성한 Startapp을 넣어야 한다.
* 'todo.apps.TodoConfig',

Templates 경로를 지정하여 준다

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

-

타임존 설정 변경

TIME\_ZONE = 'Asia/Seoul'

USE\_TZ = False

-STATICFILES\_DIRS 추가

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

Models.py

Urls.py

Start Project Urls.py에 아래부분 추가

from django.urls import path, include

path('todo/', include('todo.urls')),

추가

그리고 이전 강의와 다르게 startApp에 위 urls.py 파일을 복사하여 아래와 같이 내용을 입력하였다. (todo/vonly/ 같은 url을 처리하려 하는것인가..? 아직은 정확히 모르겠다.)

from django.urls import path

from . import views

app\_name = 'todo'

urlpatterns = [

    path('vonly/', views.TodoVueOnlyTV.as\_view(), name='vonly'),

]

Views.py

아래 코드 입력

from django.views.generic import TemplateView

# Create your views here.

class TodoVueOnlyTV(TemplateView):

    template\_name = 'todo/todo\_vue\_only.html'

templates

templates/todo/todo\_vue\_only.html 위치시키고 (1강에서 작성하였던 Vue로 작성한 html 파일이다.)

서버를 실행한 후 127.0.0.1:8000/todo/vonly/ 입력하여 출력되는 todo\_vue\_only.html을 보면



처럼 글자가 출력되지 않은걸 볼 수 있다.

그 이유는

Vue.js **머스태쉬** 문법이 Django의 Template 문법과 달라서 발생한 문제이다.

그래서 해결책은 Vue 인스턴스에서

        var vm = new Vue({

            el: '#app',

            delimiters: ['{', '}'],

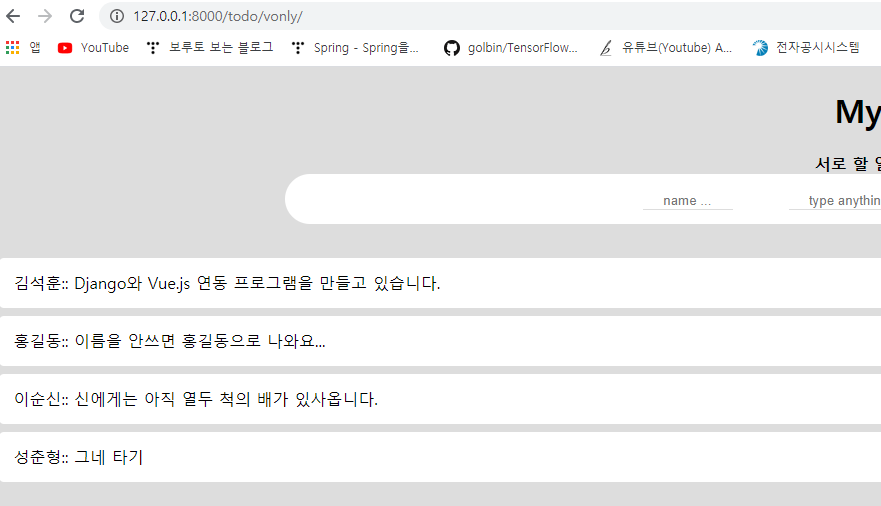
delimiter를 위처럼 바꾸고 (위 라인의 의미는 vue **머스태쉬** 문법에서 중괄호를 하나만 사용하겠다는 의미이다.),

그리고 vue 표현식 적는 부분에서 아래처럼

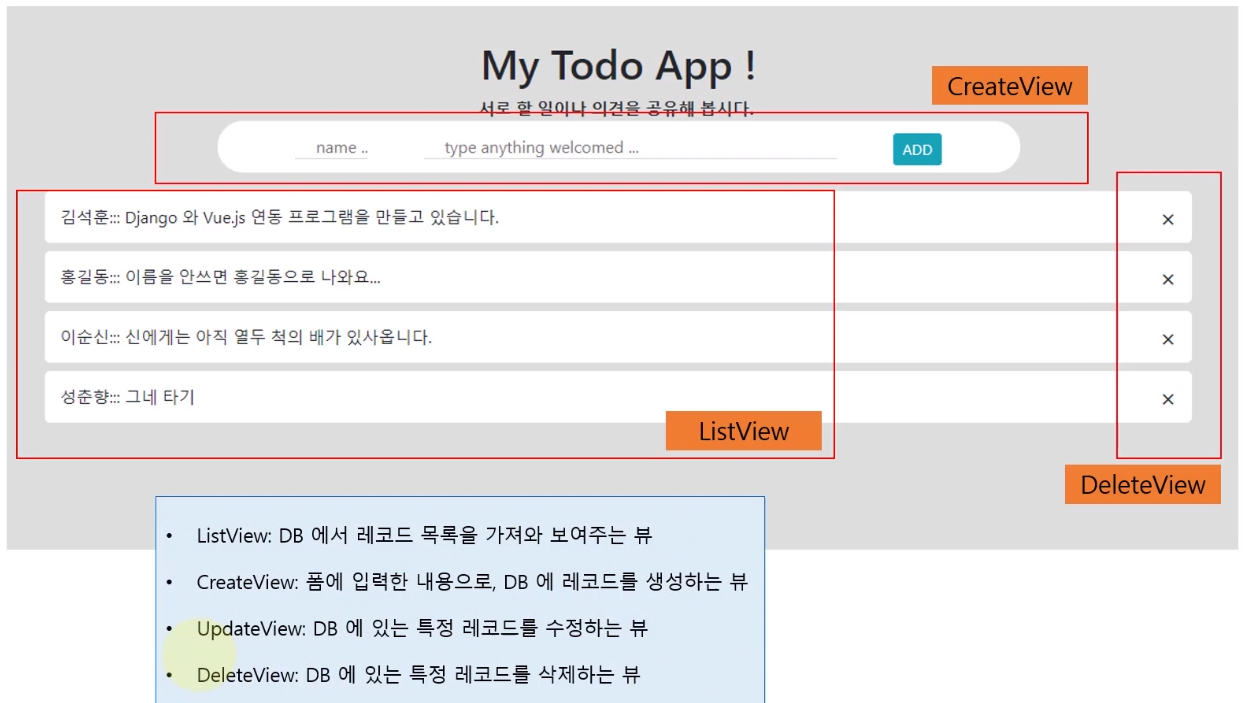
                <span>{{ todo.name }}:: {{todo.item}}</span>

을 아래로 변경하면 의도한대로 html 화면이 보여지는 것을 볼 수 있다.

                <span>{ todo.name }:: {todo.item}</span>



3강



장고에서는 위와 같이 기능을 가지는 클래스형뷰들이 많이 구현되어 있다.

기 구현된 클래스형뷰들을 잘 활용하면 개발 공수를 많이 줄일 수 있다.

Todo/models.py에 아래 테이블을 만든다.

class Todo(models.Model):

    name = models.CharField('NAME', max\_length=5, blank=True)

    todo = models.CharField('TODO', max\_length=50)

    def \_\_str\_\_(self):

        return self.todo

####

테이블을 신규로 정의할 때는 admin 사이트에서도 보일 수 있도록, admin.py 파일에도 등록을 해주어야 한다.

from todo.models import Todo

# Register your models here.

@admin.register(Todo)

class TodoAdmin(admin.ModelAdmin):

    list\_display = ('id', 'name', 'todo')

$%^$#@# 모델에 변경사항이 발생하면 이를 데이터베이스 테이블에 반영을 해주어야 한다.

python manage.py showmigrations – 로 현재 상태를 보고,

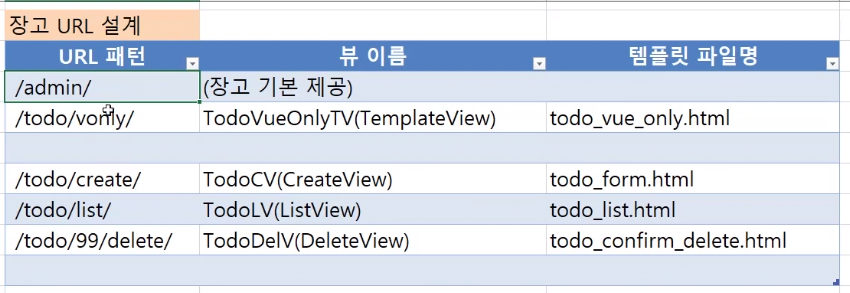
python manage.py makemigrations – 로 migrations 파일을 만든 다음에

python manage.py migrate – 로 데이터베이스에 반영을 해서 테이블을 만든다.

위 같이 Model 작업이 끝난 후에는 admin 페이지에서 model을 확인해 볼 수 있다.



Model 작업이 잘 완료된걸 확인할 수 있음



URL 패턴, 뷰 이름, 템플릿 파일명은 항상 1-1 Mapping 되므로 위처럼 정리를 해놓는게 좋다

위 표를 참조하여 Urls.py에 url 추가.

path('create/', views.TodoCV.as\_view(), name='create'),

    path('list/', views.TodoLV.as\_view(), name='list'),

    path('<int:pk>/delete/', views.TodoDelV.as\_view(), name='delete'),

<int:pk> 이런 표현은 숫자가 들어오면 INT로 변환해서 View로 넘겨주는 역할을 한다.

모든 View의 공통적인 속성(RedirectView 제외)으로 template\_name이 있다.

class TodoCV(CreateView):

    model = Todo

    fields = '\_\_all\_\_'

    template\_name = 'todo/todo\_form.html'

    success\_url = reverse\_lazy('todo:list')

class TodoLV(ListView):

    model = Todo

    template\_name = 'todo/todo\_list.html'

class TodoDelV(DeleteView):

    model = Todo

    template\_name = 'todo/todo\_confirm\_delete.html'

    success\_url = reverse\_lazy('todo:list')

CreateView는 Form 을 만들기 위해서 Model 이 필요하다.

그리고 Form 에 필요한 Field 생성 \_\_all\_\_ 모든 필드 쓴다는 의미

Form 처리가 끝난 후에는 redirect를 할 success\_url이 필요하다. (todo/list)

ListView는 Table에서 가져와야 하기 때문에 마찬가지로 Model을 정의해야 한다.

DeleteView는 특정 table에서 특정 record 하나를 삭제해야 하기 때문에 table을 지정해 주어야 한다.

Deleteview도 삭제 처리가 된 후에는 redirect가 필요하기에 redirect url을 지정한다.

!@#$ success\_url을 지정할 때는 reverse(), reverser\_lazy() 함수 중 하나를 사용해야 하는대, 여기서는 reverse\_lazy()함수를 지정해야 한다. (정확히 이해는 못했지만, 저 라인이 실행되는 시점에는 아직 urls.py 모듈이 실행되지 않은 상태여서 그렇다고 한다.)