# 정리

- models.py : 테이블을 하나의 클래스로 정의함
- admins.py : 정의된 테이블이 Admin 화면에 보이게 함
- python manage.py makemigrations : 데이터베이스에 변경이 필요한 사항을 추출함
- python manage.py migrate : 데이터베이스에 변경사항을 반영함
- python manage.py runserver : 현재까지 작업을 개발용 웹 서버로 확인함

---

게시글 작성하는 페이지 만들기

- posts 라는 앱 만들기

```
python manage.py startapp posts
```

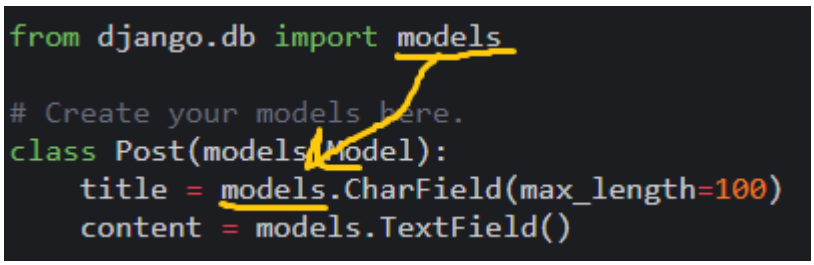
- app 등록하기 (settings.py)

```
INSTALLED_APPS = [ 'posts.apps.PostsConfig',]  # posts 라는 앱폴더 內 apps.py 內 PostsConfig 클래스 의미
```

posts/models.py 열기

- 테이블 만들기 : django.db.models.Model 클래스를 상속받아 테이블을 하나의 class 로 정의

  테이블의 열은 클래스의 변수(속성)



```
from django.db import models

# Create your models here.
class Post(models.Model):      # models = 첫 줄에서 import 한 models
  title = models.CharField(max_length=100)  # max_length 무조건 필수 (100자 길이제한)
  content = models.TextField()


  ### 💘 즉, Post 라는 표에, 1열은 title 이고, 2열은 content
```

- 실제 데이터 베이스에 적용하기 위해서 bash에 명령어 치기

```
python manage.py makemigrations    # 파이썬 코드를 데이터 베이스를 옮기기 위한 role = migrations
                    # 즉, 데이터 베이스 설계도를 만든 거?
                    # 그러므로, 설계가 변경되면, 이 명령어를 다시 쳐줘야 한다

python manage.py migrate          # 설계도를 진짜 데이터베이스로 만들어 주는 것 = 테이블 生
```

여기까지가 데이터베이스를 사용할 준비단계/ 아래부터는 실제 데이터 넣기

---

python (장고관련) 전용 shell 만들기

```
python manage.py shell
```

# CRUD : Create Read Update Delete

```
# Create

>>> from posts.models import Post  # posts 폴더 內 models.py 파일에서 Post 클래스를 import 해
>>> post = Post(title='hello', content='world!') # 여기서 끝내면 아무것도 x save 까지 해야 함
                    # Post 테이블에, 내용 넣을 거임
>>> post
<Post: Post object (None)>
>>> post.title
'hello'
>>> post.save()  # 여기까지 해야 제대로 生
```

```
# Read ( 다 가져오기)
>>> posts = Post.objects.all()    # 데이터 베이스에 저장된 모든 객체를 가져와서, posts 라는 변수에 저장
>>> posts                # 리스트로 리턴된다. 반복문 可
<QuerySet [<Post: Post object (1)>]>

# Read ( 하나만 가져오기)
>>> post = Post.objects.get(pk=1)
>>> post
<Post: Post object (1)>
```

- 조건줘서 가져오기(Read)
  - filter(where)

```
>>> posts = Post.objects.filter(title='hello').all() # 조건 걸리는 거 다 가져와
>>> posts
<QuerySet [<Post: Post object (1)>]>

>>> post = Post.objects.filter(title='hello').first()  # 조건 걸리는 것 중 가장 처음 것만 가져와
>>> post
<Post: Post object (1)>
```

  - __ (LIKE) : 조건을 딱 정하는 게 아니라, 어느 단어 포함하면 出

```
>>> posts = Post.objects.filter(title__contains='he').all()       # title= 이 아니라  ' title__ '

>>> posts
<QuerySet [<Post: Post object (1)>]>
```

- 정렬
  - order_by

```
posts = Post.objects.order_by('title').all()  # 오름차순
posts = Post.objects.order_by('-title').all()  # 내림차순
```

- offset & limit

- 슬라이싱

```
>>> Post.objects.all()[:1]        # 전체 갖고 온 거 에서 [:1] 만 슬라이싱
<QuerySet [<Post: Post object (1)>]>

>>> posts = Post.objects.all()[1:2]
```

- 삭제하기
  - delete

```
>>> post = Post.objects.get(pk=2)
>>> post.delete()
```

- 수정하기 (Update)

```
>>> post = Post.objects.get(pk=1)
>>> post.title
'hello'
>>> post.title = 'hi'   # 인스턴스만 바뀐 거고
>>> post.save()       # 꼭 save 까지 해야함
>>> post = Post.objects.get(pk=1)
>>> post.title
'hi'
```

# 페이지 만들기

crud 라는 프로젝트 폴더 內 `urls.py` 에서

```
from django.contrib import admin
from django.urls import path
from posts import views

urlpatterns = [
  path('create/',views.create),
  path('new/', views.new),
  path('admin/', admin.site.urls),
]
# 이렇게 되어 있었는데, crud 밑의  urls.py 파일은 전체 관리자 느낌이다
# 실질적으로 posts 라는 앱 밑에 있는 catch 와 throw 링크는 admin 과 같은 레벨이 아니니까,
# posts 라는 폴더 內 urls.py 라는 파일을 새로 만들어서  링크를 따로 관리해주자
```

```
# posts/urls.py 生

from django.urls import path
from . import views   # 이 urls.py 는  posts 폴더 내에 있으므로 現 폴더를 뜻하는  . 으로 바꾼다

urlpatterns = [
  path('create/',views.create),
  path('new/', views.new),
]
```

```
# crud 의 urls.py 는 아래의 형태로 바꾼다

from django.contrib import admin
from django.urls import path,include ✔ # include 도 import 해야 한다
from posts import views

urlpatterns = [
  path('posts/', include('posts.urls')),   ✔ 이 형태로
  path('admin/', admin.site.urls),
]
```

```
단, crud 밑 urls 에 있을 땐
링크가
#..../new   이었다면

하위 폴더 만들어서 url 정리 하면 링크가
#...../posts/new 로 바뀐다
```

그러므로 new.html 의 action 링크 수정도 필요하다

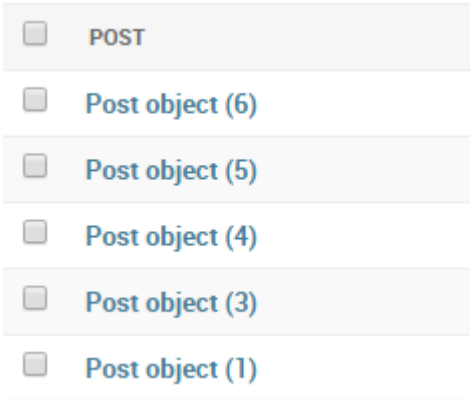
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
</head>
<body>
  <form action="/posts/create/" method="get">   ✅ 이 형태로 수정
    <input type="text" name="message"/>
    <input type="submit" value="제출"/>
  </form>
</body>
</html>
```

# 관리자계정

관리자 계정 커스터마이즈 하기 : `admin.py` 파일

관리자 페이지 주소 : `http://playground-myccpb08.c9users.io:8080/admin/`

```
from django.contrib import admin
from .models import Post

admin.site.register(Post)  # 관리자 페이지 기본 형태  # Post 테이블을 등록해
```



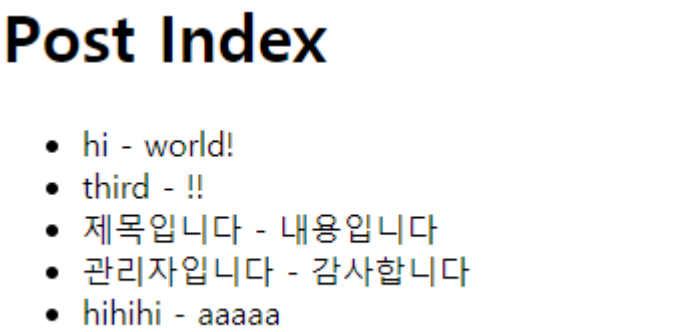
```python
# admin.py

from django.contrib import admin
from .models import Post


class PostAdmin(admin.ModelAdmin):  # admin.ModelAdmin 을 상속받은  PostAdmin 이라는 클래스
  list_display = ('title','content',)  # 관리자 페이지에서, object 목록을  1,2,3,4  가 아니라, 제목과 내용으로 보여주는

admin.site.register(Post, PostAdmin) # 위 클래스명 적어준 거. 이렇게 하면 관리자페이지에서 제목과 내용 보임
```

| TITLE | CONTENT |
| --- | --- |
| hihihi | aaaaa |
| 관리자입니다 | 감사합니다 |
| 제목입니다 | 내용입니다 |
| third | !! |
| hi | world! |

```python
# views.py

from django.shortcuts import render
from .models import Post  # ✅ post 기능 쓰려고 적어 줌

def new(request):
  return render(request,'new.html')

def create(request):
  title = request.GET.get('title')
  content = request.GET.get('content')

  # DB INSERT
  post = Post(title=title, content=content)
  post.save()

  return render(request,'create.html')
```

```python
def index(request):
    # All Post
    posts = Post.objects.all()

    return render(request, 'index.html', {'posts':posts})
```

```
# index.html

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
    <h1>Post Index</h1>        <!-- 관리자 파일 말고, ...일반 페이지에  내가 작성한 콘텐츠 출력하기 -->
    <ul>
    {% for post in posts %}
      <li>{{ post.title }} - {{post.content}}</li>  ✅✅✅✅✅✅✅✅✅✅✅
    {% endfor %}
    </ul>
</body>
</html>
```




관리자 계정 만들기

`python manage.py createsuperuser`

day17 워크샵 4번 문제

```python
# models.py 에

from django.db import models

class student(models.Model):
    name = models.CharField(max_length=100)
    email = models.CharField(max_length=100)
    birthday = models.DateField()
    age = models.IntegerField()

    def __str__(self):  # 객체를 문자열로 표현할 때 사용하는 함수.
        return self.name
```