복습

프로젝트 생성과정 1~6 실행

models.py 작성

-> make migrations

-> python manage.py migrate

라이브러리 설치

pip install ipython

pip install django-extensions

settings.py -> install app

```
INSTALLED_APPS = [
    'articles',

    'django_extensions',

    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

쉘 실행

```
python manage.py shell_plus
```

CRUD 작업 실행

CREATE

3가지 방법

```
1) 생성자 함수 호출 후 저장
```

```
In [2]: article = Article()

In [3]: article.title = "first"

In [4]: article.content = "django!"

In [5]: article.save()

2)초기값과 함께 인스턴스 생성후 저장

In [6]: article = Article(title="second", content ="django!!")

In [7]: article.save()

3) QuerySet API - create메소드   1,2와는 다르게 저장하지 않아도됨

In [8]: Article.objects.create(title="third", content="django!!!")
```

save()

객체를 DB에 저장함

데이터 생성을 해도 save()를 호출하지 않으면 db에 반영되지 않음

모델의 인스턴스를 생성한 후, 반드시 save() 호출

```
모델 완성
class Article(models.Model):
  title = models.CharField(max_length=10)
  content = models.TextField()
  created_at = models.DateTimeField(auto_now_add=True)
  updated_at = models.DateTimeField(auto_now=True)

  def __str__(self):
    return self.title;
```

```
python manage.py makemigrations

python manage.py migrate
```

CRUD - READ

all()

현재 쿼리셋의 복사본을 반환

```
Article.objects.all()
```

get()

주어진 조건에 해당하는 객체를 반환

찾을 수 없는 경우 DoesNotExist

둘 이상 찾은 경우 MultipleObjectsReturned 예외

"딱 하나만 있을 수 있는 목표를 찾는다" -> pk, id같은 "고유한" "Unique"한 데이터를 조회하는데 사용

```
 Article.objects.get(content="django!!")

 Article.objects.get(content="django!!!!")
 DoesNotExist: Article matching query does not exist.
 찾을 수 없는 경우 DoesNotExist

 Article.objects.get(pk=1)
 Article.objects.get(id=1)

In [14]: Article.objects.get(content="django!!!")

MultipleObjectsReturned: get() returned more than one Article -- it returned 2!
```

get() vs filter()

filter -> "여러개를 조회"

```
Article.objects.filter(title="second")

In [16]: Article.objects.filter(content="django!!!")
Out[16]: <QuerySet [<Article: first>, <Article: third>]>
```

CRUD - UPDATE(변경)

인스턴스 생성때와 마찬가지로 인스턴스를 생성후 **변경후 저장**

```
In [8]: article = Article.objects.get(id=1)

In [12]: article.content = "django!!!"

In [13]: article.save()
```

CRUD - DELETE(삭제)

쿼리셋의 모든 행에 대해 삭제 쿼리를 수행후 삭제된 객체들을 포함시킨 딕셔너리를 반환

```
article = Article.objects.get(id=1)
article.delete()
```

어드민 등록

```
python manage.py createsuperuser
```

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

# Register your models here.
admin.site.register(Article)
```

템플릿 상속

폴더 경로에 templates와 base.html 생성


```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap demo</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
rbsA2VBKQhggwzxH7pPCaAqO46MgnOM8OzW1RWuH61DGLwZJEdK2Kadq2F9CUG65"
crossorigin="anonymous">
  </head>
  <body>

    <div class="container">
      {% block content  %}

      {% endblock  %}
    </div>
```

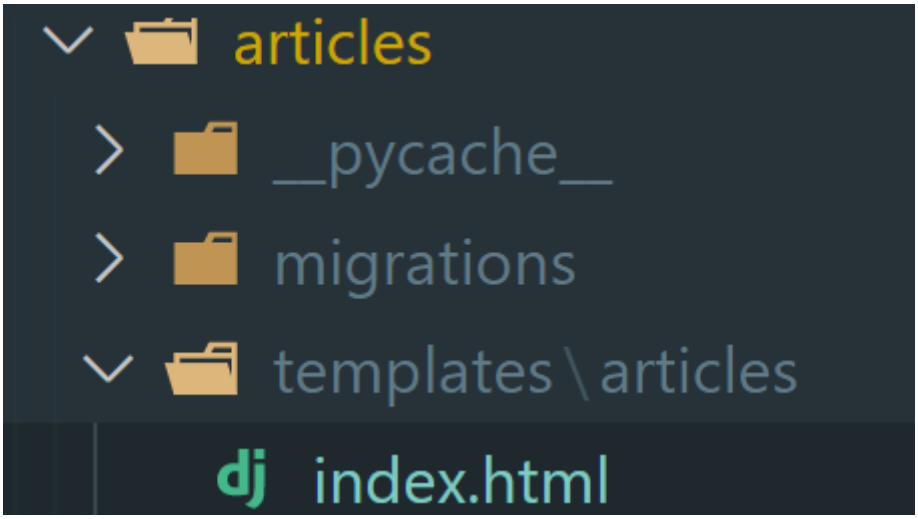
```
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.j
s" integrity="sha384-
kenU1KFdBIe4zVF0s0G1M5b4hcpxyD9F7jL+jjXkk+Q2h455rYXK/7HAuoJl+0I4"
crossorigin="anonymous"></script>
  </body>
</html>
```

settings.py -> 'DIRS' 작성

```
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [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',
            ],
        },
    },
]
```

---

app local templates 생성



index.html

```
{% extends 'base.html' %}

{% block content %}
  <h1 class="text-center">Articles</h1>
{% endblock  %}
```

프로젝트 url

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path("articles/", include('articles.urls'))
]
```

articles url

```
from django.urls import path
from . import views


app_name = 'articles'
urlpatterns = [
    path("", views.index, name="index")
]
```

articles / views.py

```
from django.shortcuts import render

# Create your views here.
def index(request):
  return render(request, 'articles/index.html')
```

CRUD -> 전체 게시글 조회

articles/views.py

```
from django.shortcuts import render
from .models import Article

# Create your views here.
def index(request):
  articles = Article.objects.all()

  context = {
    'articles' : articles
  }

  return render(request, 'articles/index.html', context)
```

articles/templates/articles/index.html

```
{% extends 'base.html' %}

{% block content %}
  <h1 class="text-center">Articles</h1>
  <hr>
  {% for article in articles %}
    <p>글 번호 : {{ article.pk }}</p>
    <p>글 제목 : {{ article.title }}</p>
    <p>글 내용 : {{ article.content }}</p>
    <hr>
  {% endfor %}

{% endblock  %}
```

CRUD - CREATE

articles / urls.py

```
from django.urls import path
from . import views


app_name = 'articles'
urlpatterns = [
    path("", views.index, name="index"),
    path('new/', views.new, name="new")
]
```

articles/views.py

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

articles / new.html

```
{% extends 'base.html' %}


{% block content %}

<h1 class="text-center">NEW</h1>

<form action="#" method="GET">
  <label for="title">Title : </label>
  <input type="text" name="title">
  <br/>
  <label for="content">Content : </label>
  <textarea name="content" cols="30" row="5"></textarea>
  <br>
  <input type="submit">
</form>
<hr>
```

```
<a href="{% url 'articles:index' %}">[BACK]</a>

{% endblock  %}
```

articles/index.html

```
{% extends 'base.html' %}

{% block content %}
  <h1 class="text-center">Articles</h1>
  <a href="{% url 'articles:new' %}">NEW</a>
  <hr>


  {% for article in articles %}
    <p>글 번호 : {{ article.pk }}</p>
    <p>글 제목 : {{ article.title }}</p>
    <p>글 내용 : {{ article.content }}</p>
    <hr>
  {% endfor %}

{% endblock  %}
```

articles/urls.py

```
from django.urls import path
from . import views


app_name = 'articles'
urlpatterns = [
    path("", views.index, name="index"),
    path('new/', views.new, name="new"),
    path('create/', views.create, name="create"),
]
```

articles/ views.py

```
def create(reqeust):
  return render(reqeust, 'articles/create.html')
```

articles / create.html

```
{% extends 'base.html' %}

{% block content %}
  <h1 class="text-center">성공적으로 글이 작성되었습니다</h1>
{% endblock  %}
```

articles/create.html

```
{% extends 'base.html' %}

{% block content %}
  <h1 class="text-center">성공적으로 글이 작성되었습니다</h1>
{% endblock  %}
```

article/ views.py

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

  article = Article()
  article.title = title
  article.content = content
  article.save()
  # 1번

  # article = Article(title=title,content=content)
  # article.save()
  # 2번

  # Article.objects.create(title=title, content=content)
  # 3번

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

전체 조회 내림차순 변경

```
def index(request):
  articles = Article.objects.all()

  # articles = Article.objects.order_by('-pk')
  # articles = Article.objects.order_by('-id')
  # 'db에서' 내림차순으로 해서 가져와 화면에 출력

  # articles = Article.objects.all()[::-1]
  # db에서 가져와서 '파이썬'으로 내림차순으로 변경해서 출력

  context = {
    'articles' : articles
  }
```

```
request.GET.get('title')

GET -> HTTP 메서드
GET -> "특정" 데이터를 요청할때 사용-> 반드시 데이터를 가져와야함
-> DB에 변화를 주지 않음 (CRUD) R -> READ에 해당

POST -> HTTP 메서드
POST -> 서버로 데이터를 전송할때 사용 -> 데이터를 변경/생성 / DB에 변경사항을 발생
CRUD -> C/U/D CREATE,UPDATE,DELETE 담당
```

GET 메서드를 POST메서드로 변경

```
{% extends 'base.html' %}


{% block content %}

<h1 class="text-center">NEW</h1>

<form action="{% url 'articles:create' %}" method="POST">
  <label for="title">Title : </label>
  <input type="text" name="title">
  <br/>
  <label for="content">Content : </label>
  <textarea name="content" cols="30" row="5"></textarea>
  <br>
  <input type="submit">
</form>
<hr>

<a href="{% url 'articles:index' %}">[BACK]</a>

{% endblock  %}
```

CSRF 공격 방어

Security Toekn 방식(CSRF TOKEN)

사용자의 데이터에 임의의 난수를 부여함 -> 매 요청마다 해당 난수값을 포함시켜 전송

-> 서버에서 요청을 받을때마다 전달된 token값이 유효한지 검증

-> POST, PATCH ,DELETE (CREATE, UPDATE, DELETE)등에 적용

Django -> CSRF token 탬플릿을 제공

articles / views.py

```python
def create(request):
  title = request.POST.get('title')
  content = request.POST.get('content')
  # print(request)

  article = Article()
  article.title = title
  article.content = content
  article.save()
  # 1번

  # article = Article(title=title,content=content)
  # article.save()
  # 2번

  # Article.objects.create(title=title, content=content)
  # 3번

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

문제점 발생 : 게시글 작성후 index페이지로 이동하지 않음
