한국폴리텍대학
로봇캠퍼스

웹페이지 만들기

PC기반 제어 실습 (장고와 부트스트랩 활용)
5주차 강의

로봇IT

강의 계획표 / 장고와 부트스트랩 활용

주	주제
1	강의 및 수업의 목표 소개와 개발 환경 구축
2	웹 프런트엔드 기초와 부트스트랩 활용 방법
3	장고의 역할 이해와 프로젝트 만들기
4	장고 프로젝트에서 앱 개발하기
5	**웹 페이지 만들기**
6	정적 파일과 미디어 파일 관리하기
7	페이지 구성 개선하기
8	중간고사
9	테스트 주도 개발법으로 템플릿 모듈화하기
10	다대일 및 다대다 관계 구현
11	폼으로 포스트 작성과 수정 기능 구현
12	외부 라이브러리 활용
13	폼으로 댓글 기능 및 편의 기능 구현
14	홈페이지와 자기소개 페이지 완성 및 공개
15	장고 활용 UI 제작과 모드버스 연동
16	기말고사

URL 설정하기

- **표지판 역할을 하는 urls.py**
  -> 장고 프로젝트 폴더에서 urls.py 파일을 열어서 내용을 확인
  -> 127.0.0.1:8000/admin 페이지의 처리 방법을 확인 (자동 생성된 코드)

실습 파일: do_it_django_prj/urls.py

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

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

path 함수는 URL(웹 주소) 매칭 기능 수행
  > 첫 번째 매개변수는 URL 주소
  > 두 번째 매개변수는 뷰 함수 호출

개발에 필요한 페이지 확인

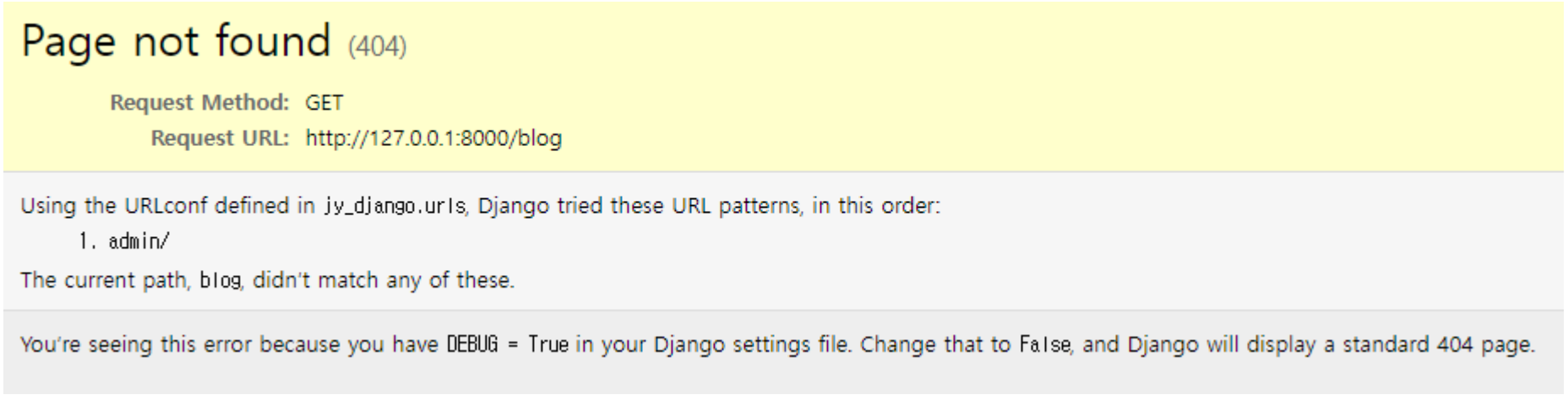
-> 지금까지 초기(index.html), 블로그(blog_list.html), 자기소개(about_me.html) 페이지를 작업했음

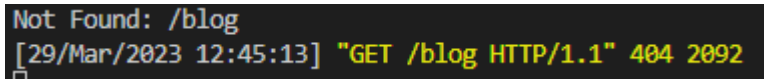
- 초기(대문) 페이지 : IP(도메인) 주소를 입력했을 때 표시하는 화면
- 블로그 페이지 : IP(도메인) 주소 뒤에 /blog를 입력했을 때 표시하는 화면
  포스트(게시글) 목록과 상세 내용을 보여주는 화면으로 구성하고
  포스트 상세 페이지는 /blog/포스트의pk 로 처리
- 자기소개 페이지 : IP(도메인) 주소 뒤에 /about_me를 입력했을 때 표시하는 화면

페이지		URL
대문 페이지		도메인/
블로그 페이지	포스트 목록	도메인/blog/
	포스트 상세	도메인/blog/포스트 pk
자기소개 페이지		도메인/about_me/

블로그 페이지 URL로 접속하기

-> 가상환경에서 장고 서버를 실행 (python manage.py runserver)
-> 웹 브라우저 주소 창에 http://127.0.0.1:8000/blog 입력
- URL을 설정하지 않아서 오류 발생





터미널에서도 오류 확인 가능

- **blog/urls.py 만들기 1**
  -> urls.py 파일에 blog로 접속하는 코드 추가
  -> include 함수를 추가하고 아래와 같이 path를 추가함
  - include() : 다른 앱의 URL을 포함시키는 기능

실습 파일: do_it_django_prj/urls.py

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

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

- **blog/urls.py 만들기 2**
  -> blog 앱의 폴더에는 urls.py 파일이 없으므로, 새로 urls.py 파일을 만들고 아래와 같이 코드 작성
  -> 이 파일에서 127.0.0.1:8000/blog 주소를 처리함

실습 파일: blog/urls.py

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

urlpatterns = [
    # 이 부분을 채울 겁니다!
]
```

이후에 페이지를 만들고 URL을 연결함

FBV로 페이지 만들기

FBV와 CBV

-> urls.py에 추가하는 함수나 클래스 등은 views.py 에서 정의
-> views.py 에 추가하는 코드 방식은 FBV와 CBV가 있음
-> FBV(Function Based View) : 함수를 만들어서 기능을 구현
-> CBV(Class Based View) : 클래스를 만들어서 기능을 구현

※ FBV와 CBV 중 원하는 방식을 사용하면 됨

- **blog/urls.py에 내용 추가**
  -> blog/urls.py 파일을 아래와 같이 코드를 추가
  (views.py에 index 함수를 만들어 구현할 것임)

실습 파일: blog/urls.py

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

urlpatterns = [
    path('', views.index),
]
```

blog/views.py에 index() 함수 정의

-> blog/urls.py 파일을 아래와 같이 코드를 추가
(views.py에 index 함수를 만들어 구현할 것임)

실습 파일: blog/views.py

```
from django.shortcuts import render

def index(request):
    return render(
        request,
        'blog/index.html',
    )
```

-> 코드 추가 후 웹 브라우저에서 127.0.0.1:800/blog 에 접속
※ 코드 수정 및 추가 후 저장하면 소스코드를 다시 빌딩함
-> TemplateDoesNotExist 오류 메시지로 변경됨을 확인 : index.html이 없기 때문

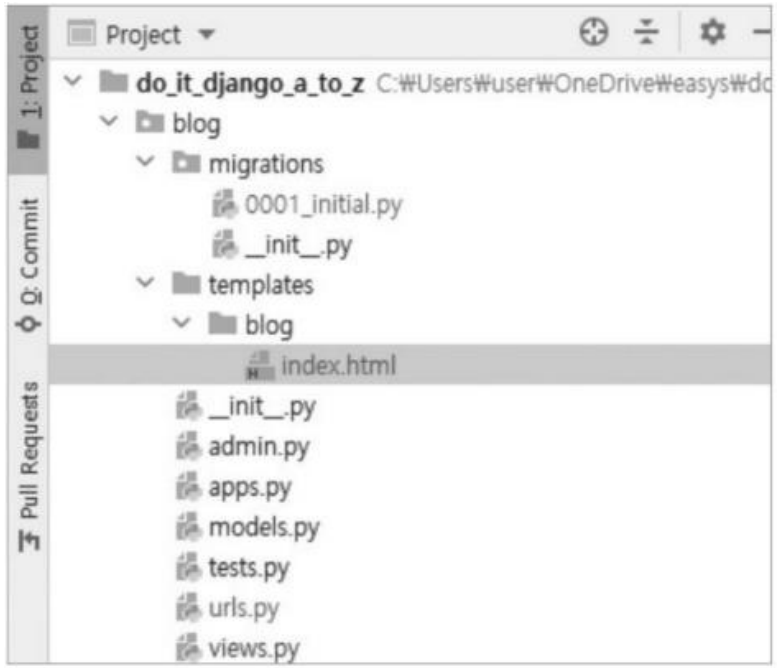
템플릿 파일 만들기 1

-> blog/templates/blog 폴더를 새로 만듦 (주의)
- blog 폴더 안에 templates 폴더를 만들고, templates 폴더 안에 blog 폴더를 만듦

-> blog/templates/blog 폴더에 index.html 파일을 만들고 아래 코드를 입력
- html 태그의 lang 속성에 ko를 입력하여 한국어 페이지로 설정(페이지 검색, 분석, 정렬 방향 등 결정)

실습 파일: blog/templates/blog/index.html

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <title>Blog</title>
</head>
<body>
    <h1>Blog</h1>
</body>
</html>
```



템플릿 파일 만들기 2

-> 웹 브라우저를 새로 고침하여 결과를 확인

-> 아래와 같이 Blog 글자가 나타나지 않는다면, 임의의 py 파일에서 저장 버튼을 누르거나 서버를 다시 시작함

블로그 페이지에 포스트 목록 나열 1

-> views.py 코드를 아래와 같이 수정

- from .models import Post : models.py에 정의된 Post 모델을 포함
- posts = Post.objects.all() : 모든 Post 레코드를 가져와서 posts 변수에 저장
- 'posts' : posts : posts 변수를 딕셔너리로 저장

실습 파일: blog/views.py

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

def index(request):
    posts = Post.objects.all()

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

Post.objects.all() 함수로
데이터베이스에 쿼리를 주고
레코드를 얻을 수 있음

※ 쿼리 : 데이터베이스의 데이터 획득
수정, 삭제 등의 요청

블로그 페이지에 포스트 목록 나열 2

-> index.html 파일을 수정

-> 쿼리로 가져온 Posts 레코드를 for 반복문으로 출력

- for 문은 {% %}로 감싸고, 변수는 {{ }}로 감싸줌 : Python 코드와 구분을 위해
- {% %} 는 파이썬 코드를 실행하는 태그
- {{ }} 변수 사용을 위한 태그

실습 파일: blog/templates/blog/index.html

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <title>Blog</title>
</head>
<body>
    <h1>Blog</h1>

{% for p in posts %}
    <h3>{{ p }}</h3>
{% endfor %}
</body>
</html>
```

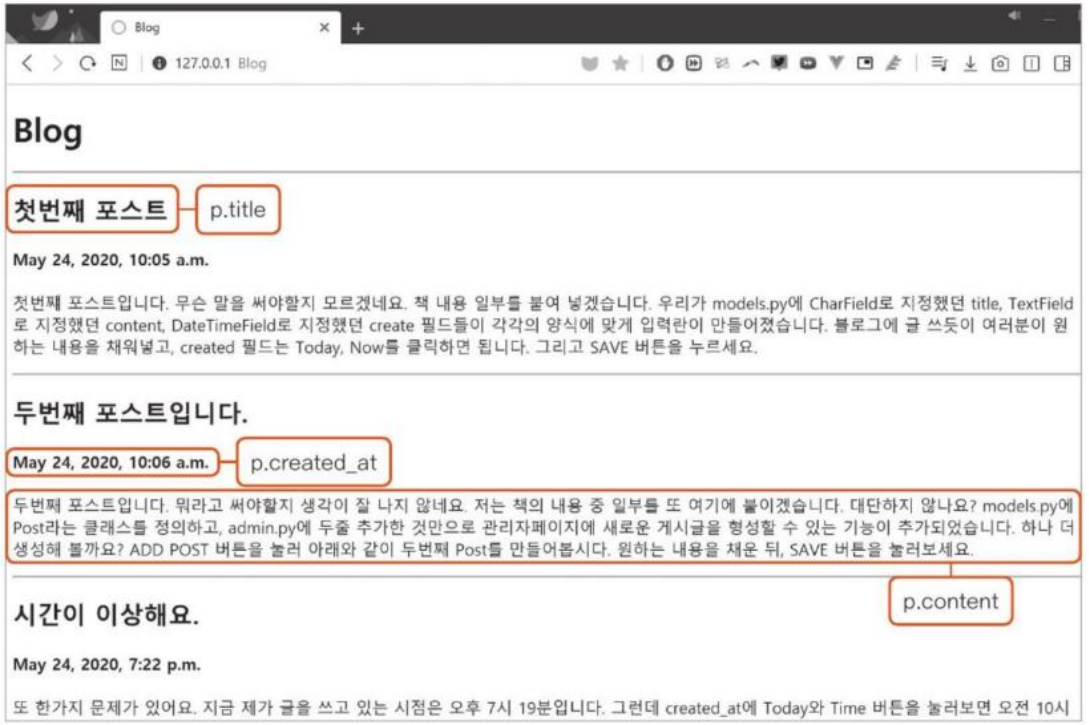
- **블로그 페이지에 포스트 목록 나열 3**
  -> 웹 브라우저에서 결과를 확인함
  -> 이전(4주차)에 models.py 에서 수정한대로, __str__() 함수로 정의한 내용이 출력됨

Post 모델의 필드값 보여주기

-> for 반복문으로 가져온 p 값은 models.py 의 Post 객체들임
-> 데이터의 접근은 닷(.)으로 처리 가능
-> index.html 코드를 아래와 같이 수정 후 결과 확인

실습 파일: blog/templates/blog/index.html

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <title>Blog</title>
</head>
<body>
<h1>Blog</h1>
{% for p in posts %}
    <hr/>
    <h2>{{ p.title }}</h2>
    <h4>{{ p.created_at }}</h4>
    <p>{{ p.content }}</p>
{% endfor %}
</body>
</html>
```

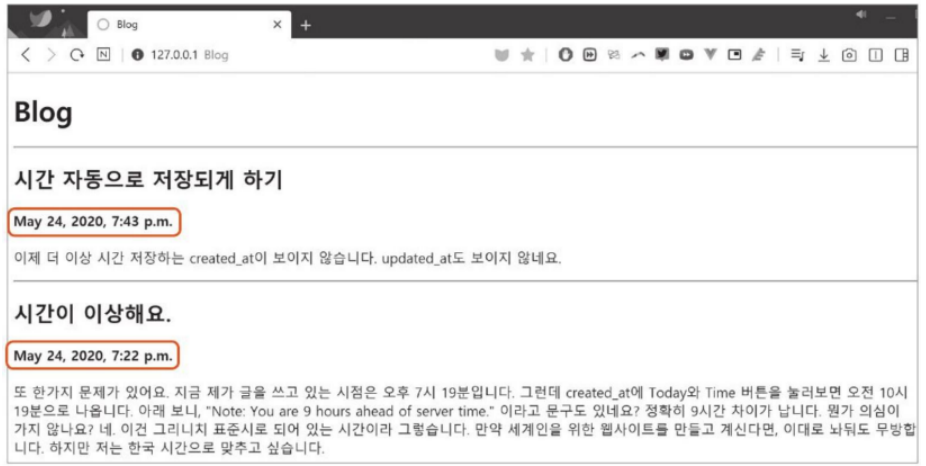


최신 포스트부터 보여주기

-> 가장 최근에 작성된 글부터 보여주기 위해 정렬 기능을 사용
-> 정렬은 order_by 함수를 사용
-> blog/views.py 코드를 아래와 같이 수정 : pk 값의 역순으로 정렬됨 (-pk)

실습 파일: blog/views.py

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

def index(request):
    posts = Post.objects.all().order_by('-pk')

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



실행하여 순서가 바뀌어 있음을 확인

- **지금까지 한 작업을 gitHub에 버전 생성**
  -> 서버를 중단하거나 새 터미널을 실행하여 명령어들을 실행

Cmder

```
C:\github\do_it_django_a_to_z (main -> origin)
(venv) λ git add .
C:\github\do_it_django_a_to_z (main -> origin)
(venv) λ git commit -m "FBV로 블로그 포스트 목록 페이지 만들기"
C:\github\do_it_django_a_to_z (main -> origin)
(venv) λ git push
```

포스트 상세 페이지 URL 정의

-> blog/urls.py 에 아래 코드를 추가

(정리) : 생성한 장고 프로젝트의 urls.py에서 127.0.0.1:8000/blog 주소가 입력될 경우에는 blog/urls.py 에서 처리한다고 작성했음

blog/urls.py 에서는 127.0.0.1:8000/blog/ 주소 뒤에 입력되는 것을 처리함

-> <int:pk> 주소 뒤에 입력되는 정수값을 pk 변수에 담아서 처리한다는 의미

-> 즉, /blog/ 뒤에 정수가 입력되면 single_post_page() 를 처리함

실습 파일: blog/urls.py

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

urlpatterns = [
    path('<int:pk>/', views.single_post_page),
    path('', views.index),
]
```

single_post_page() 함수 정의

-> blog/views.ph 파일에 single_post_page() 함수를 아래와 같이 정의함

실습 파일: blog/views.py

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

def index(request):
    posts = Post.objects.all().order_by('-pk')

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

def single_post_page(request, pk):
    post = Post.objects.get(pk=pk)

    return render(
        request,
        'blog/single_post_page.html',
        {
            'post': post,
        }
    )
```

- request 매개변수에 추가로 pk가 추가됨
- Post.objects.get(pk=pk) 함수는 괄호 안의 조건을 만족하는 Post 레코드를 가져오는 쿼리

URL을 이용한 쿼리 확인

-> 웹 브라우저에서 127.0.0.1:8000/blog/1 이라고 입력하여 결과 확인
- single_post_page.html 을 만들지 않아 TemplateDoesNotExist 오류 발생

-> 웹 브라우저에서 127.0.0.1:8000/blog/100 이라고 입력하여 결과 확인
- Post.objects.get(pk=pk) 에서 가져올 수 있는 레코드가 없어서 오류 발생

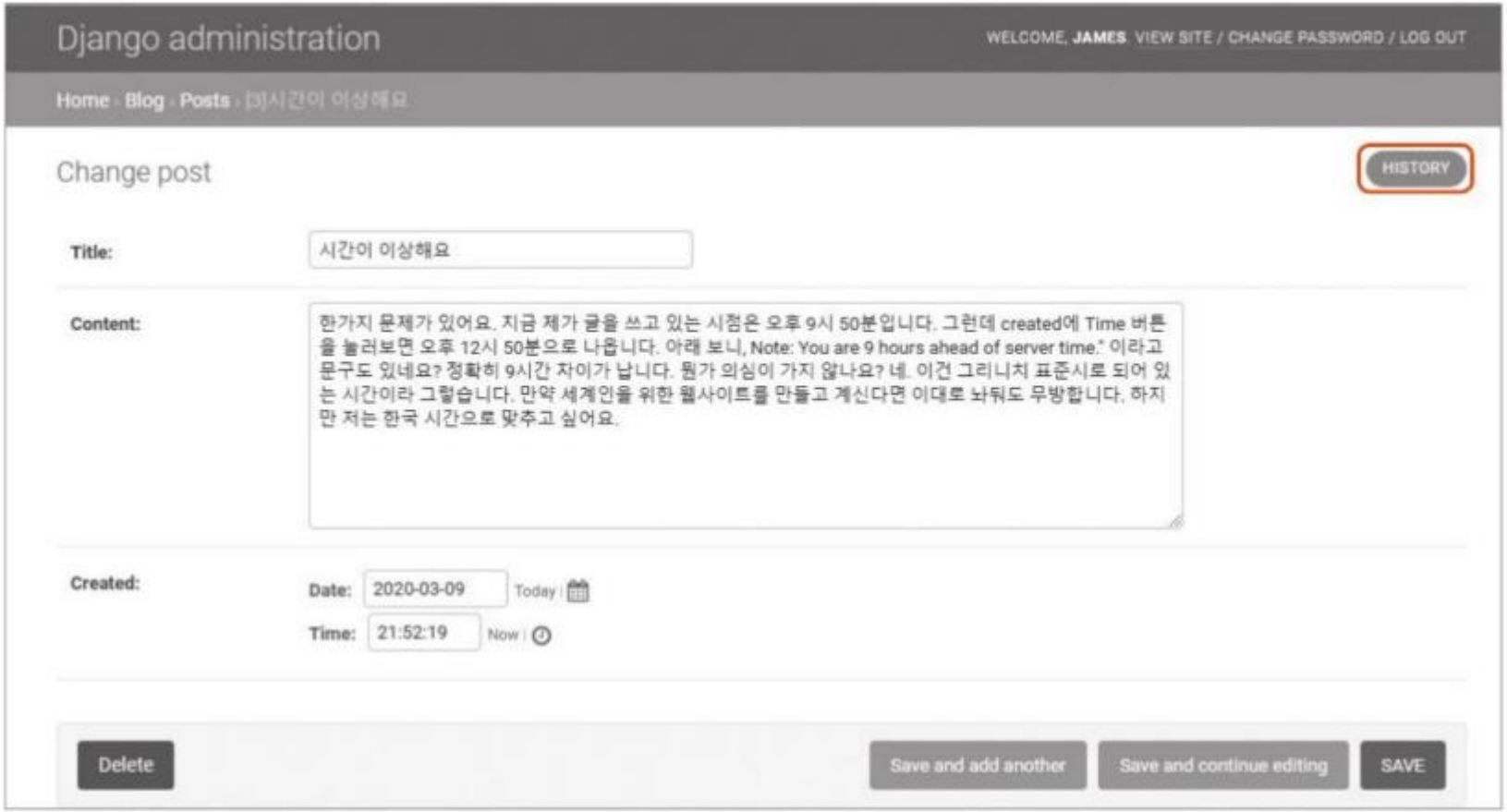
템플릿 파일 만들기

-> blog/templates/blog 폴더에 single_post_page.html 을 아래 코드로 생성
-> 127.0.0.1:8000/blog/1/ 로 접속하여 결과 확인 (포스트 만든 개수 만큼 숫자를 바꿔도 됨)

실습 파일: blog/templates/blog/single_post_page.html

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <title>{{ post.title }} - Blog</title>
</head>
<body>
<nav>
    <a href="/blog/">Blog</a>
</nav>
<h1>{{ post.title }}</h1>
<h4>{{ post.created_at }}</h4>
<p>{{ post.content }}</p>
<hr/>
<h3>여기 댓글이 들어올 수 있겠죠?</h3>
</body>
</html>
```

- **포스트 제목에 링크 만들기 1**
  -> 관리자 페이지(admin)으로 이동해서 포스트를 선택하면 우측 상단에 HISTORY 버튼만 있음



포스트 제목에 링크 만들기 2

-> blog/models.py 파일에 get_absolute_url() 함수 추가

- get_absolute_url 함수는 레코드별 URL 생성 규칙을 정의하는 함수 (약속된 이름)

실습 파일: blog/models.py

```
from django.db import models


class Post(models.Model):
    title = models.CharField(max_length=30)
    content = models.TextField()

    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    # author: 추후 작성 예정

    def __str__(self):
        return f'[{self.pk}]{self.title}'

    def get_absolute_url(self):
        return f'/blog/{self.pk}/'
```

포스트 제목에 링크 만들기

-> 포스트 목록 페이지에서 포스트 제목을 클릭하면 상세 페이지로 이동하는 링크를 생성

-> blog/templates/blog/index.html 파일을 아래와 같이 수정 후 테스트

실습 파일: blog/templates/blog/post_list.html

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <title>Blog</title>
</head>
<body>
<h1>Blog</h1>

{% for p in posts %}
    <hr/>
    <h2><a href="{{ p.get_absolute_url }}">{{ p.title }}</a></h2>
    <h4>{{ p.created }}</h4>
    <p>{{ p.content }}</p>
{% endfor %}
</body>
</html>
```

여기까지 작업된 것을 gitHub에 업로드

single_pages 앱을 위한 URL 지정

-> 첫 페이지는 127.0.0.1:8000 으로 접속했을 때 나타나는 페이지
-> do_it_django_prj/urls.py 파일을 아래와 같이 수정

실습 파일: do_it_django_prj/urls.py

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

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

첫 페이지와 자기소개 페이지의 URL 지정

-> single_pages 앱 폴더에 urls.py 파일을 생성 후 아래 코드를 입력
-> 기본 주소(127.0.0.1:8000) 인 경우는 views.py 의 landing() 함수를 연결
-> 주소 뒤 about_me/ 가 있는 경우 about_me() 함수를 연결

실습 파일: single_pages/urls.py

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

urlpatterns = [
    path('about_me/', views.about_me),
    path('', views.landing),
]
```

views.py에 함수 정의

-> single_pages/views.py 파일에 landing() 과 about_me() 함수를 추가

실습 파일: single_pages/views.py

```
from django.shortcuts import render

def landing(request):
    return render(
        request,
        'single_pages/landing.html'
    )

def about_me(request):
    return render(
        request,
        'single_pages/about_me.html'
    )
```

템플릿 파일 만들기 1

-> 앞서 실습한 것(blog) 처럼, single_pages/templates/single_pages 폴더를 만들고 폴더 안에 landing.html 파일을 생성 후 아래와 같이 입력 (이름은 본인 이름으로)

실습 파일: single_pages/templates/single_pages/landing.html

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <title>달타냥입니다!</title>
</head>
<body>
<nav>
    <a href="/blog/">Blog</a>
    <a href="/about_me/">About me</a>
</nav>

<h1>안녕하세요. 달타냥입니다.</h1>
<h2>대문페이지</h2>
<h3>아직 만들지 않음</h3>
</body>
</html>
```

템플릿 파일 만들기 2

-> single_pages/templates/single_pages 폴더에 about_me.html 파일을 생성 후 아래와 같이 입력 (이름은 본인 이름으로)

실습 파일: single_pages/templates/single_pages/about_me.html

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <title>개발자 달타냥입니다.</title>
</head>
<body>

<nav>
    <a href="/blog/">Blog</a>
    <a href="/about_me/">About me</a>
</nav>

<h1>안녕하세요. 달타냥입니다</h1>
<h2>이력</h2>
<h2>Portfolio</h2>
<h3>아직 공사중입니다. </h3>
</body>
</html>
```

잘 동작하는지 확인 후 gitHub에 업로드

CBV로 페이지 만들기

ListView로 포스트 목록 페이지 만들기

-> CBV는 반복해서 사용하는 기능들을 클래스 형태로 제공 (사용하기 편할 수 있음)
-> 여러 포스트를 나열할 때는 ListView 클래스 활용
-> CBV 실습을 위해 blog/view.py 의 코드는 주석처리하여 실습
-> blog/view.py 파일을 열고 아래와 같이 코드를 입력

실습 파일: blog/views.py

```
from django.shortcuts import render
from django.views.generic import ListView
from .models import Post

class PostList(ListView):
    model = Post

# def index(request):
#     posts = Post.objects.all()
#
#     return render(
#         request,
#         'blog/index.html',
#         {
#             'posts': posts,
#         }
#     )
(...생략...)
```

주석 처리하거나 삭제!

기존코드를 바꿨기 때문에
오류가 발생하나 무시

urls.py 수정

-> blog/urls.py 파일을 열고 아래와 같이 코드를 수정

실습 파일: blog/urls.py

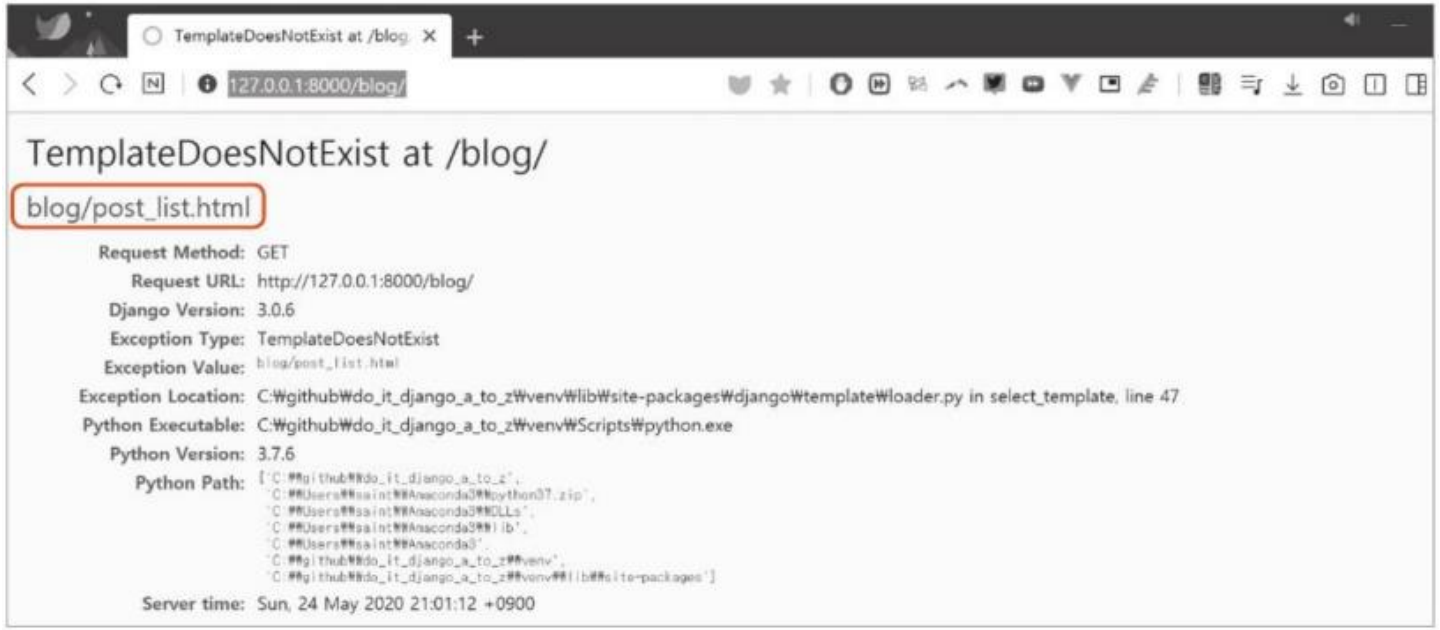
```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.PostList.as_view()),
    # path('', views.index),
  #path('<int:pk>/', views.single_post_page),
]
```

템플릿 파일 지정 1

-> 웹 브라우저에서 127.0.0.1:8000/blog 로 접속하여 결과 확인

-> TemplateDoesNotExist 오류와 blog/post_list.html 파일이 필요한 것을 확인

- ListView 는 모델명 뒤에 _list 가 붙은 html 파일을 기본 템플릿으로 사용
- 즉 Post 모델을 사용하면 post_list.html 파일이 필요
- 또는 PostList 클래스에서 template_name 을 직접 지정할 수 있음



템플릿 파일 지정 2 – template_name을 지정하는 방법 1

-> blog/views.py의 PostList 클래스에 template_name = 'blog/index.html'을 추가 후 결과 확인
-> 작성한 포스트 내용이 보이지 않음
- FBV 사용 시 쿼리로 가져온 데이터를 posts 딕셔너리로 지정했었음

실습 파일: blog/views.py

```
from django.shortcuts import render
from .models import Post
from django.views.generic import ListView

class PostList(ListView):
    model = Post
    template_name = 'blog/index.html'
(...생략...)
```

템플릿 파일 지정 3 – template_name을 지정하는 방법 2

-> ListVIew로 만든 클래스 모델 객체를 가져오려면 object_list 명령어 사용
  - 또는 Post 모델을 사용했으므로 post_list 라고 써도 자동으로 인식함
-> blog/index.html 파일에 작성했던 posts 부분을 object_list (또는 post_list)로 바꿔줌

실습 파일: blog/templates/blog/index.html

```
(...생략...)
<body>
<h1>Blog</h1>

{% for p in post_list %}
    <hr/>
    <h2><a href="{{ p.get_absolute_url }}">{{ p.title }}</a></h2>
    <h4>{{ p.created }}</h4>
    <p>{{ p.content }}</p>
{% endfor %}
</body>
</html>
```

템플릿 파일 지정 4 – post_list.html 파일을 사용하는 방법

-> template_name 을 지정하지 않으면 post_list.html 파일을 자동으로 인식
-> views.py 파일에서 template_name='blog/index.html' 을 삭제 (또는 주석처리)
-> blog/templates/blog/index.html 파일 이름을 post_list.html 로 수정 (마우스 우클릭 - rename)

실습 파일: blog/views.py

```
from django.shortcuts import render
from django.views.generic import ListView
from .models import Post

class PostList(ListView):
    model = Post
    template_name = 'blog/index.html'
(...생략...)
```

최신 포스트부터 보여주기

-> ListView에도 정렬 기능을 제공
-> blog/views.py 파일의 PostList 클래스에 ordering = '-pk' 를 추가함

실습 파일: blog/views.py

```
from django.shortcuts import render
from django.views.generic import ListView
from .models import Post

class PostList(ListView):
    model = Post
    ordering = '-pk'
(...생략...)
```

결과 확인 후
gitHub에 업로드

DetailView로 포스트 상세 페이지 만들기

-> 레코드 내용을 자세히 보여줄 때는 DetailView 클래스를 이용
-> blog/views.py 파일을 열어 아래와 같이 두 줄을 추가함
-> render 함수를 임포트 하는 코드는 삭제 또는 주석처리

실습 파일: blog/views.py

```
~~from django.shortcuts import render~~
from django.views.generic import ListView, DetailView
from .models import Post

class PostList(ListView):
    model = Post
    ordering = '-pk'

class PostDetail(DetailView):
    model = Post
(...생략...)
# def single_post_page(request, pk):
#     post = Post.objects.get(pk=pk)
#
#     return render(
#         request,
#         'blog/single_post_page.html',
#         {
#             'post': post,
#         }
#     )
```

- **urls.py 수정**
  -> blog/urls.py 파일을 열고 path('<int:pk>/', views.PostDetail.as_view()) 코드 추가

실습 파일: blog/urls.py

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

urlpatterns = [
    path('<int:pk>/', views.PostDetail.as_view()),
    path('', views.PostList.as_view()),
    # path('<int:pk>/', views.single_post_page),
    # path('', views.index),
]
```

템플릿 파일 지정

-> 웹 브라우저에서 127.0.0.1:8000/1/을 입력하면 blog/post_detail.html 이 없다는 오류 확인
-> template_name 속성으로 single_post_page.html 을 지정 (또는 파일명을 post_detail.html 로 변경)

결과 확인 후
gitHub에 업로드

다음 강의 예고

정적 파일과 미디어 파일 관리하기

- 정적 파일 관리하기
- 미디어 파일 관리하기

Q&A

감사합니다.