

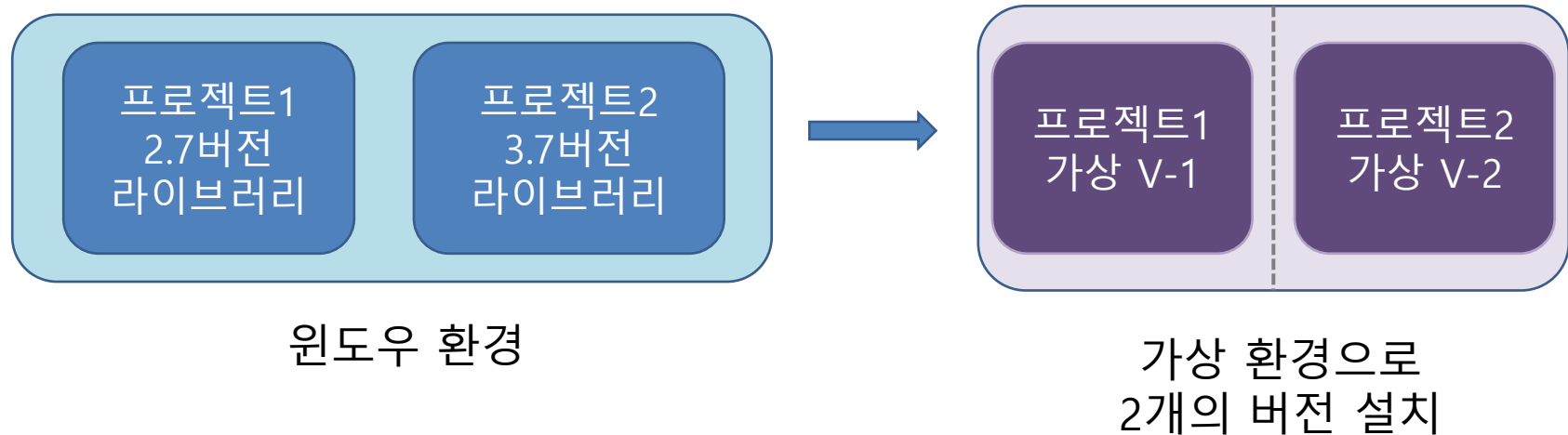
# 1장. blog 사이트(목록, 상세, 등록)



# 가상 환경(venv)

## ❖ 가상화

파이썬 가상 환경은 파이썬 프로젝트를 진행할 때 독립된 환경을 제공한다.  
파이썬 버전이나 라이브러리 버전이 다른 경우 하나의 데스크탑에서 개발할 경우 문제가 발생할 수 있다.



# 가상화 모듈 venv 설치

## ❖ Django 개발 환경 구축

1. venvs 디렉터리와 하위에 myweb 디렉터리를 생성
2. 파이썬 모듈 중 venv를 현재 폴더에 설치(끝에 한 칸 띄고 점 찍음)

```
C:\venvs\myweb>python -m venv .
```

3. 가상 환경에 진입하기

```
C:\venvs\myweb\Scripts>activate  
(myweb) C:\venvs\myweb\Scripts>
```

4. 가상 환경 벗어나기

```
(myweb) C:\venvs\myweb\Scripts>deactivate
```

# 장고(django) 설치하기

## ❖ 장고(django) 설치하기

### 1. 가상환경에서 장고 설치하기

```
(myweb) C:\venvs\ myweb \Scripts>pip install django
```

### 2. 장고 버전 확인하기

```
(myweb) C:\venvs\ myweb \Scripts>python -m django --version
```

# blog 프로젝트 생성하기

## ❖ blog 프로젝트 만들기

1. myblog 프로젝트 생성하기 – 가상진입 > webproject로 이동

```
(myweb) > C:\projects>django-admin startproject myblog
```

2. 서버를 구동해서 127.0.0.1:8000 페이지 확인

```
(myweb) > C:\projects\myblog>python manage.py runserver
```

3. app을 sqlite3에 저장하기 위한 migrate 실행하기

```
(myweb) > C:\projects\myblog>python manage.py migrate
```

# 블로그 사이트 제작

## Blog

### 인공지능 스피커 - 프렌즈

2022년 3월 12일 7:14 오전

네이버 프렌즈



### 파이썬 웹 4기 종강

2022년 3월 12일 5:40 오전

3월 17일 종강

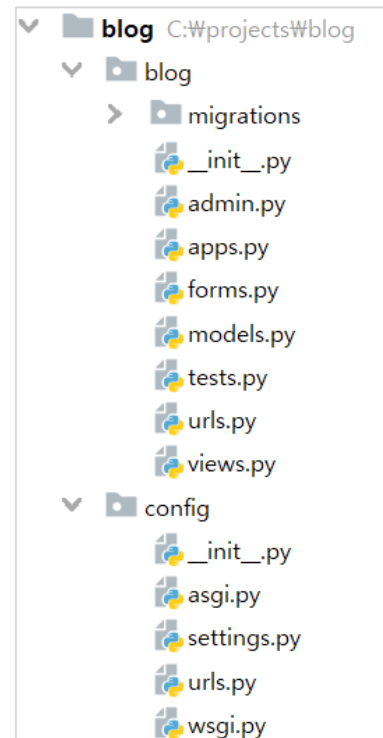
# blog 앱 생성하기

## ❖ blog app(앱) 생성하기

1. blog프로젝트 하위에 blog 앱 생성하기

```
(myweb) > C:\projects\myblog>python manage.py startapp blog
```

2. blog앱 하위에 urls.py 생성하기



# blog 프로젝트 생성하기

## ❖ config/settings.py에서 설정 변경하기

### 1. 언어와 시간 변경하기

```
LANGUAGE_CODE = 'ko-kr'  
  
TIME_ZONE = 'Asia/Seoul'
```

### 2. 'blog'앱 등록하기 – DB 사용을 위한 필수 설정

```
INSTALLED_APPS = [  
    'blog',  
    'django.contrib.admin',  
    'django.contrib.auth',  
]
```



# 127.0.0.1:8000

## ❖ 127.0.0.1:8000 경로 설정

### 1. config/urls.py 경로 설정

```
from blog import views
urlpatterns = [
    path('admin/', admin.site.urls),
    path('blog/', include('blog.urls')),
    path("", views.index), #127.0.0.1:8000
]
```

### 2. blog/urls.py 작성

```
from . import views
urlpatterns = [
    path("", views.index)
]
```

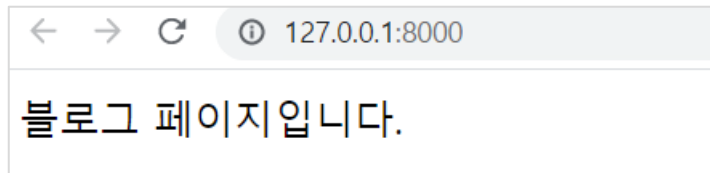
# 127.0.0.1:8000 요청

❖ 127.0.0.1:8000 주소에 **문자 출력하기**

3. blog/views.py 작성

```
from django.http import HttpResponse  
def index(request):  
    return HttpResponse("블로그 페이지입니다.")
```

4. 127.0.0.1:8000 접속



# 템플릿으로 index 페이지 만들기

- 템플릿(templates) 만들기

1. blog 프로젝트 하위에 **templates** 디렉토리를 바로 밑에 만든다.

```
(myweb) > C:\projects\myblog>mkdir templates
```

2. 템플릿 디렉토리 위치를 config/settings.py에 등록하기

```
TEMPLATES = [  
    .....  
    'DIRS': [BASE_DIR / 'templates'],  
    .....  
]
```

# 템플릿으로 index 페이지 만들기

## 3. index 함수 작성하기 - blog/urls.py

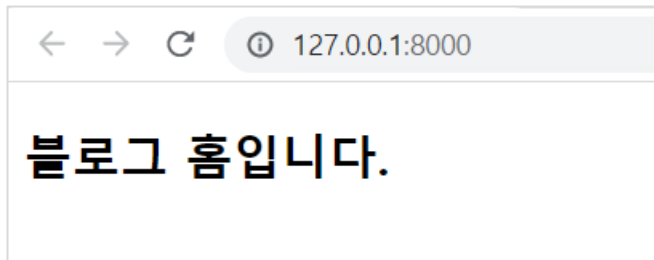
```
app_name = 'blog'
urlpatterns = [
    path("", views.index, name='index'),
]
```

## 4. index 함수 작성하기 - blog/views.py

```
def index(request):
    #블로그 홈
    return render(request, 'blog/post_list.html')
```

# 블로그(홈) 목록 페이지 만들기

3. templates 하위에 blog 디렉터리 생성 후 post\_list.html 만들기
  - templates/blog/index.html



```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <title>블로그 홈</title>
</head>
<body>
  <h2>블로그 홈입니다.</h2>
</body>
</html>
```

# 블로그 DB 작업

## ▪ Post 모델 만들기

1. blog/models.py에 Post 모델 생성

```
from django.db import models

class Post(models.Model):
    title = models.CharField(max_length=100)
    content = models.TextField()
    pub_date = models.DateTimeField()
    modify_date = models.DateTimeField(null=True, blank=True)
    # author는 추후 작성

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

2. 변경된 DB 적용을 위한 makemigrations와 migrate 실행

# 장고 Admin

## ◆ 장고 Admin – superuser 생성

장고는 Admin은 관리자 기능을 갖춘 app이다.  
모델 관리를 shell 환경이 아닌 사이트 화면에서 할 수 있다.  
127.0.0.1:8000/admin으로 접속.  
먼저 슈퍼 유저를 생성해야 함.

```
(myweb) > C:\projects\myblog>python manage.py createsuperuser
```

```
사용자 이름 : admin
```

```
이메일 주소 : admin@test.com
```

```
Password:*****
```

```
Password(again):*****
```

# 장고 Admin

## ◆ localhost:8000/admin 에 접속하기

Django 관리

사용자 이름:

비밀번호:

로그인

admin / 12345

Django 관리

홈 › 인증 및 권한 › 사용자(들)

BLOG	
Posts	+ 추가

인증 및 권한	
그룹	+ 추가
사용자(들)	+ 추가

변경할 사용자 선택

Q  검색

액션:  실행 1 중 아무것도

<input type="checkbox"/>	사용자 이름	이메일 주소
<input type="checkbox"/>	admin	sugu2100@daum.net

1 사용자



# 장고 Admin

- ◆ 장고 Admin에서 모델 관리하기
  - blog/admin.py 에 등록

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

admin.site.register(Post)
```

## Django 관리

### 사이트 관리

#### BLOG

Posts

+ 추가    ✎ 변경

#### 인증 및 권한

그룹

+ 추가    ✎ 변경

사용자(들)

+ 추가    ✎ 변경

# 관리자 페이지에서 포스트하기

## ■ 관리자 페이지에서 post 추가하기

post 추가

**Title:** 21년 화이트 크리스마스

**Content:** 올해는 눈이 많이 온 화이트 크리스마스이다.  
기온은 많이 내려가 최강 한파...

**Pub date:** 날짜: 2021-12-26  
시각: 07:53:24

변경할 post 선택

액션: ----- ▼ 실행 2 중 아무것도 선택되지 않았습니다.

<input type="checkbox"/>	POST
<input type="checkbox"/>	오미크론
<input type="checkbox"/>	화이트 크리스마스

2 posts

# 블로그홈에 포스트 보여주기

- 블로그 홈 페이지



# 블로그홈에 포스트 보여주기

블로그 홈 index() 수정하기 – blog/views.py

```
def index(request):  
    #블로그 홈  
    post_list = Post.objects.order_by('-pub_date')  
    context = {'post_list': post_list}  
    return render(request, 'blog/post_list.html', context)
```

# 블로그홈에 포스트 보여주기

블로그 홈 - templates/blog/post\_list.html

```
<nav>
    <a href="{% url 'blog:index' %}">블로그홈</a>
</nav>
<section>
    <h2>Blog</h2>
    {% if post_list %}
        {% for post in post_list %}
            <hr>
            <h3>{{ post.title }}</h3>
            <h5>{{ post.pub_date }}</h5>
            <p>{{ post.content }}</p>
        {% endfor %}
    {% else %}
        <p>포스트가 없습니다.</p>
    {% endif %}
</section>
```

# 포스트 상세 페이지 만들기

- 포스트 상세 페이지 만들기

1. 메인 템플릿에 상세페이지 링크 추가 – blog/post\_list.html

```
<hr>
<h3><a href="{% url 'blog:detail' post.id %}">{{ post.title }}</a></h3>
    <h3><a href="blog/{{ post.id }}">{{ post.title }}</a></h3>-->
<h5>{{ post.pub_date }}</h5>
<p>{{ post.content }}</p>
```

# 포스트 상세 페이지 만들기

## 2. 상세페이지 URL 매핑 추가 – blog/urls.py

```
urlpatterns = [  
    path('', views.index, name='index'),  
    path('<int:post_id>/', views.detail, name='detail')  
]
```

## 3. detail 함수 추가 – blog/views.py

```
def detail(request, post_id):  
    # 상세 페이지  
    post = Post.objects.get(id=post_id)  
    context = {'post': post}  
    return render(request, 'blog/post_detail.html', context)
```

# 포스트 상세 페이지 만들기

## 4. 상세페이지 템플릿 만들기 – blog/post\_detail.html

```
<head>
  <meta charset="UTF-8">
  <title>{{ post.title }} - Blog</title>
</head>
<body>
  <nav>
    <a href="{% url 'blog:index' %}">블로그홈</a>
  </nav>
  <section>
    <h2>{{ post.title }}</h2>
    <h5>{{ post.pub_date }}</h5>
    <p>{{ post.content }}</p>
    <hr>
  </section>
</body>
```



# 이미지 파일 업로드 하기

블로그 메인입니다.

인공지능 스피커 - 프렌즈

2022년 3월 12일 7:14 오전

네이버 프렌즈



# 미디어 파일 관리하기

- 'media' 폴더 설정 - config/settings.py

```
import os
from pathlib import Path

STATIC_URL = '/static/'

MEDIA_URL = '/media/'
# 프로젝트 폴더 아래 media 폴더가 생성됨
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

# 미디어 파일 관리하기

- 미디어 경로 추가하기 - config/urls.py

```
from django.conf.urls.static import static
from django.conf import settings
```

```
urlpatterns = [
```

```
....
```

```
    path('blog/', include('blog.urls'),
```

```
]
```

```
urlpatterns += static(settings.MEDIA_URL,  
                        document_root=settings.MEDIA_ROOT)
```

# 이미지 파일 업로드 하기

- Post모델에 photo 필드 추가하기 - blog/models

```
class Post(models.Model):  
    title = models.CharField(max_length=100)  
    content = models.TextField()  
    pub_date = models.DateTimeField()  
    modify_date = models.DateTimeField(null=True, blank=True)  
    photo = models.ImageField(upload_to='blog/images/%Y/%m/%d/',  
                              null=True, blank=True)
```

- python manage.py makemigrations
- python manage.py migrate

# 이미지 파일 업로드 하기

- Pillow 모듈 – ImageField() 사용 가능함

```
PS C:\projects\myblog> pip install Pillow
Collecting Pillow
  Downloading Pillow-9.0.1-cp310-cp310-win_amd64.whl (3.2 MB)
    |██████████████████████████████████████| 3.2 MB 3.3 MB/s
Installing collected packages: Pillow
Successfully installed Pillow-9.0.1
```

# 이미지 파일 업로드 하기

- 미디어 파일 표시하기 – templates/blog/post\_list.html

```
{% for post in post_list %}
    <hr>
    <h3><a href="{% url 'blog:detail' post.id %}">
        {{ post.title }}</a></h3>
    <h5>{{ post.pub_date }}</h5>
    <p style="white-space:pre-line; line-height:1.5em">
        {{ post.content }}
    </p>
    {% if post.photo %}
        
    {% endif %}
{% endfor %}
```

# 관리자 페이지에서 포스트 하기

## 방역패스

Title:

방역패스

Content:


방역패스 또는 접종증명·음성확인제는 우리나라에서 시행하고 있는 코로나19 관련 백신패스 제도이다.

Pub date:

날짜: 2021-12-26 오늘 | 

시각: 11:31:31 현재 | 

Modify date:

날짜: 2021-12-26 오늘 | 

시각: 11:31:31 현재 | 

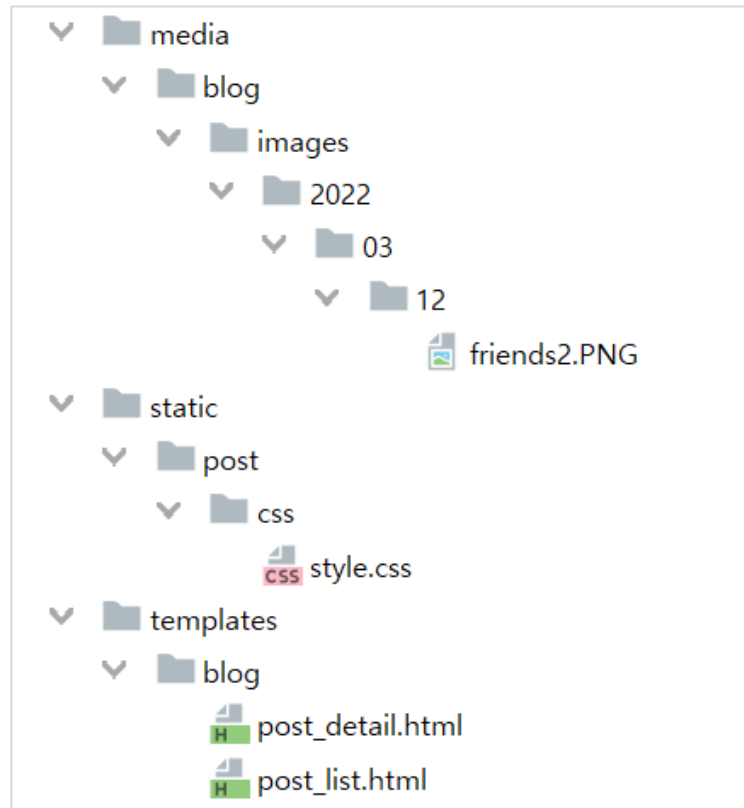
Photo:

현재: <blog/images/2021/12/26/corona1.jpg> ☐ 취소

변경:  선택된 파일 없음

# 프로젝트 파일 관리

- 프로젝트 계층도





# 깃으로 버전(소스) 관리하기

## ➤ 깃으로 버전(소스) 관리하기

깃허브 본인 계정에서 레포지터리 생성 – 이름 : myblog

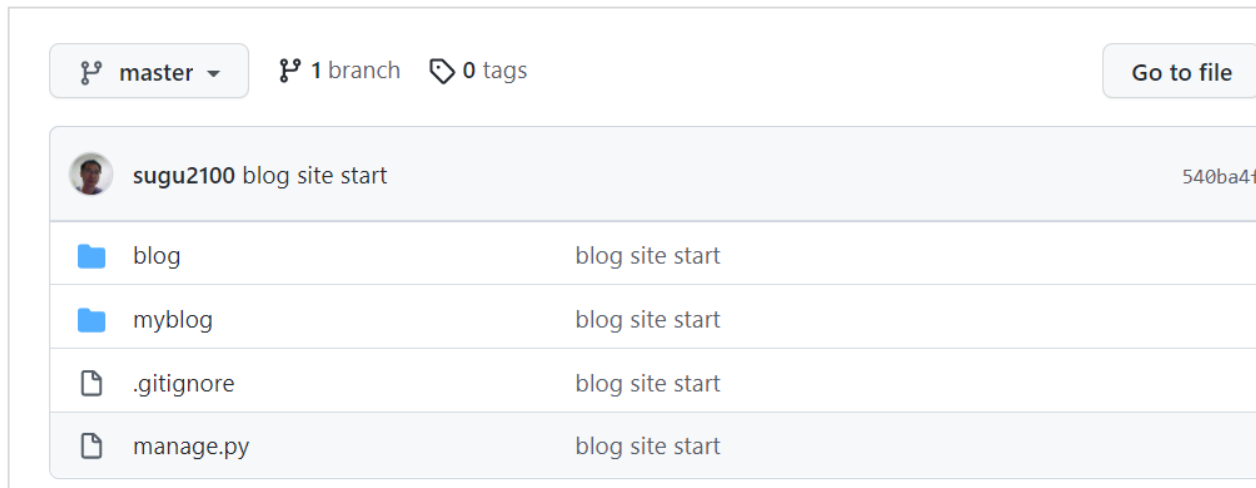
...or create a new repository on the command line

```
echo "# goweb" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/sugu2100/goweb.git
git push -u origin main
```

# 깃으로 버전(소스) 관리하기

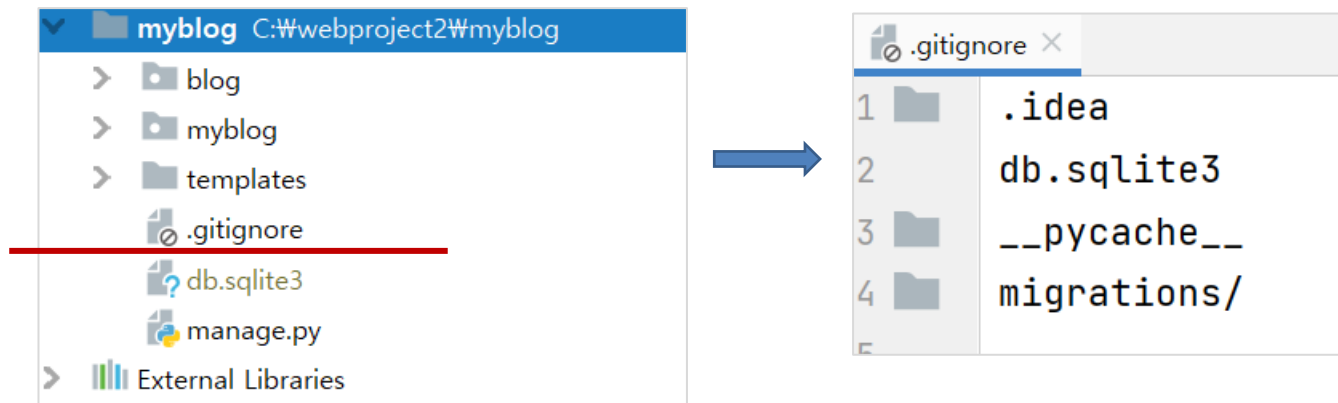
## ➤ 깃으로 버전(소스) 관리하기

깃허브 본인 계정에서 레포지터리 생성 – 이름 : myblog



# .gitignore 파일 만들기

- .gitignore 파일 만들기 – git으로 관리하지 않을 파일 등록



# static – style.css

- static 환경 설정

```
STATIC_URL = 'static/'  
STATICFILES_DIRS = [ BASE_DIR / 'static' ]
```

- style.css 링크하기

```
<!DOCTYPE html>  
<html lang="ko">  
  {% load static %}  
  <head>  
    <meta charset="UTF-8">  
    <title>Blog</title>  
    <link rel="stylesheet" href="{% static 'post/css/style.css' %}">  
  </head>
```

# static – style.css

- style.css

```
#container{width:996px; margin: 20px auto;}

nav ul{list-style:none;}
nav ul li{display:inline-block; margin: 10px;}
```

- post\_list.html

```
<div id="container">
  <header>
    <nav>
      <ul>
        <li><a href="{% url 'blog:index' %}">Home</a></li>
      </ul>
    </nav>
  </header>
  <section>
    <h2>Blog</h2>
```

# 포스트 등록 페이지

## ● 글쓰기 폼

### 포스트 등록

제목:

2021년 화이트 크리스마스..  
그러나 최강한파  
영하 10도..

내용:

사진:  white.jpg

[블로그홈 글쓰기](#)

### Blog

[화이트 크리스마스](#)

2021년 12월 26일 4:23 오후

2021년 화이트 크리스마스..  
그러나 최강한파  
영하 10도..



# 포스트 등록 폼 만들기

- 포스트 등록 폼 만들기

1. 포스트 등록 URL 매핑하기 – blog/urls.py

```
urlpatterns = [  
    ....  
    path('post/create/', views.post_create, name='post_create')  
]
```

# 포스트 등록 폼 만들기

## 2. PostForm 만들기- blog/forms.py

```
class PostForm(forms.ModelForm):  
    class Meta:  
        model = Post  
        fields = ['title', 'content', 'photo']  
        labels = {  
            'title': '제목',  
            'content': '내용',  
            'photo': '사진'  
        }
```



# 포스트 등록 폼 만들기

## 3. post\_create 함수 정의 – blog/views.py

```
def post_create(request):  
    # 포스트 등록  
    if request.method == "POST":  
        # 글과 이미지 각각 가져옴  
        form = PostForm(request.POST, request.FILES)  
        if form.is_valid():  
            post = form.save(commit=False) # 가져장  
            post.pub_date = timezone.now() # 등록일  
            post.save() #db에 저장  
            return redirect('blog:index')  
        else:  
            form = PostForm()  
            context = {'form': form}  
            return render(request, 'blog/post_form.html', context)
```

# 포스트 등록 폼 만들기

## 4. 포스트 등록 템플릿 만들기 – blog/post\_form.html

```
<nav>
  <a href="{% url 'blog:index' %}">블로그홈</a>
  <a href="{% url 'blog:post_create' %}">글쓰기</a>
</nav>
<section>
  <h2>포스트 등록</h2>
  <form action="" method="post" enctype="multipart/form-data">
    <!-- 일반 필드와 이미지필드로 멀티파트로 구성됨-->
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">저장하기</button>
  </form>
</section>
```

# 파일 업로드 / 다운로드

## 쌀밥

라이프스타일

작성자 : cloud (작성일 : 2022년 3월 17일 6:42 오전) (수정일 : 2022년 3월 17일 6:48 오전) [목록](#) [수정](#) [삭제](#)



맛있어 보이는 밥  
파일을 다운로드 하세요

[Download](#) : 

# 파일 업로드 / 다운로드

## 1. Post 모델에 file 필드 추가

```
class Post(models.Model):
    title = models.CharField(max_length=40)
    hook_text = models.CharField(max_length=100, blank=True)
    content = models.TextField()
    create_date = models.DateTimeField(auto_now_add=True)
    update_date = models.DateTimeField(auto_now=True)
    photo = models.ImageField(upload_to='blog/images/%Y/%m/%d/', blank=True)
    file = models.FileField(upload_to='blog/files/%Y/%m/%d/', blank=True)
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    category = models.ForeignKey(Category, null=True, blank=True, on_delete=models.SET_NULL)
```

# 파일 업로드 / 다운로드

## 2. PostForm에 file 필드 추가

```
class PostForm(forms.ModelForm):  
  
    class Meta:  
        model = Post  
        fields = ['title', 'content', 'photo', 'file', 'category']  
        labels = {  
            'title': '제목',  
            'content': '내용',  
            'photo': '사진',  
            'file': '파일',  
            'category': '카테고리'  
        }  
}
```

## 파일 업로드 / 다운로드

### 3. post\_detail에 file 출력

```
<hr>
{% if post.photo %}
    
{% endif %}
<p>{{ post.content | linebreaks }}</p>
{% if post.file %}
    <p><a href="{{ post.file.url }}">
        Download : <i class="far fa-file"></i>
    </a></p>
{% endif %}
<hr>
```

# 파일 업로드 / 다운로드

## 4. 글쓰기 폼 작성

### 글쓰기

제목:

내용: 

맛있어 보이는 밥  
파일 첨부합니다.

사진:  food.jpg

파일:  3장\_blog\_글...카테고리.pptx



카테고리:  ▼

# fontawesome – icon

❖ Fontawesome 사이트에서 무료 아이콘 사용하기


**Take the hassle out of icons  
in your website.**

Font Awesome is the Internet's icon library and toolkit, used by millions of designers, developers, and content creators.

 [Start for Free](#)  [Get More with Pro](#)

**Sign In**  
and let's get you to the Awesome!

Email Address

Password

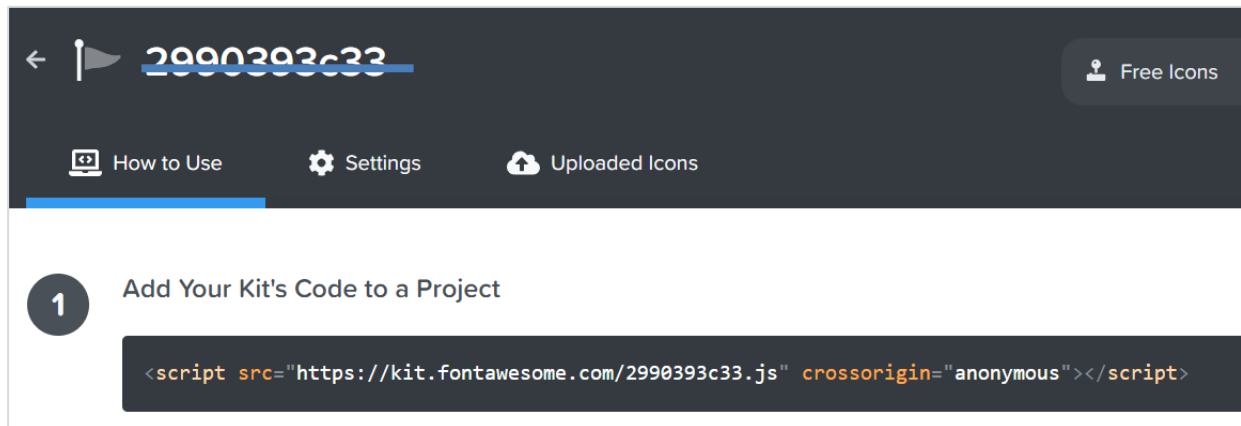
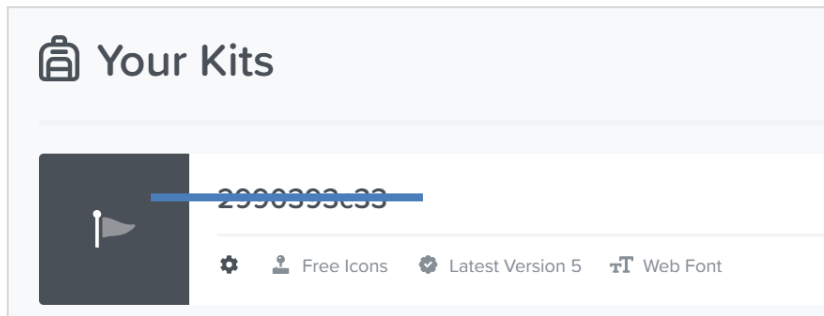
☒ Remember me [Forgot your password?](#)

[Sign In](#) ➔



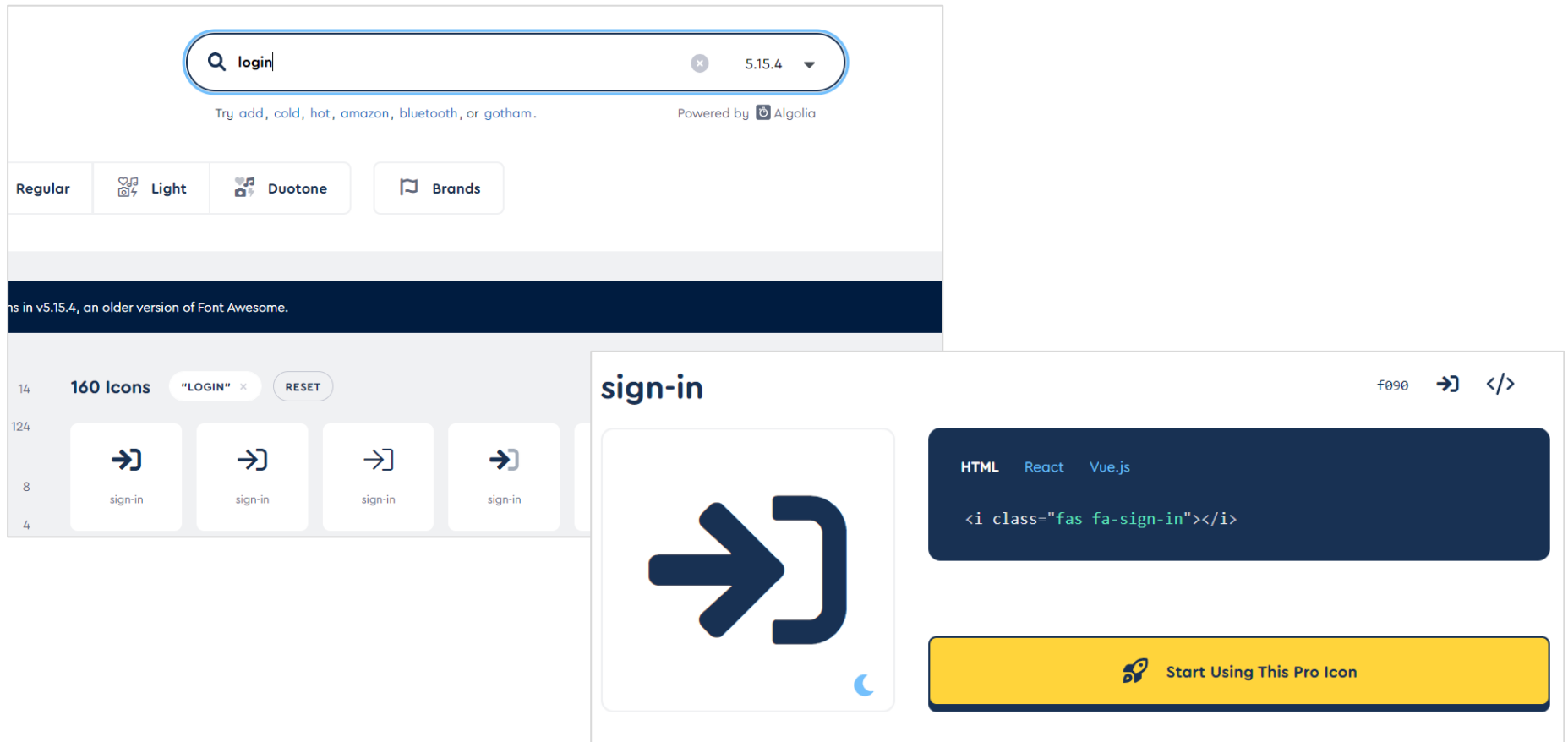
# fontawesome – icon

❖ Fontawesome 사이트에서 무료 아이콘 사용하기



# fontawesome – icon

❖ Fontawesome 사이트에서 무료 아이콘 사용하기



# fontawesome - icon

[Home](#)   [글쓰기](#)   ➡ [로그인](#)   [회원가입](#)

```
{% load static %}
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<title>{% block title %} {% endblock %}</title>
```

```
<link rel="stylesheet" href="{% static 'post/css/style.css' %}">
```

```
<script src="https://kit.fontawesome.com/2990393c33.js" crossorigin="anonymous"></script>
```

```
</head>
```

base.html

navbar.html

```
{% if user.is_authenticated %}
```

```
<li><a href="{% url 'common:logout' %}">({{ user.username }}님)로그아웃</a></li>
```

```
{% else %}
```

```
<li><a href="{% url 'common:login' %}"><i class="fas fa-sign-in-alt"></i> 로그인</a></li>
```

```
{% endif %}
```

```
<li><a href="{% url 'common:signup' %}">회원가입</a></li>
```

```
</ul>
```

```
</nav>
```