■ 파이썬 가상환경 설치
python -m venv [가상환경 이름]
ex)
```
>python -m venv firstproject
```

■ django 설치
```
(firstproject) > pip install django
```

■ django 프로젝트 설치
django-admin startproject [프로젝트 이름] . <----마지막 마침표(.) 입력
ex)
```
(firstproject) > django-admin startproject mysite .
```

현재까지의 디렉토리 구조
```
200django
|-- manage.py
`-- mysite
    |-- __init__.py
    |-- settings.py
    |-- urls.py
    `-- wsgi.py
```

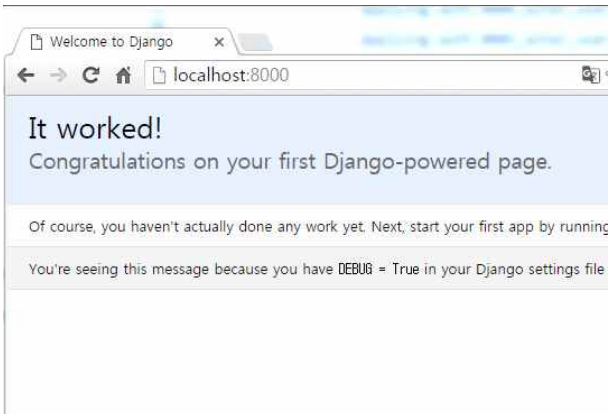
▶ manage.py
스크립트 파일로 사이트 관리를 도와주는 역할을 한다.
이 스크립트로 다른 설치 작업 없이, 컴퓨터에서 웹 서버를 시작할 수 있다.
settings.py는 웹사이트 설정이 있는 파일
urls.py 파일은 urlresolver가 사용하는 패턴 목록을 포함

▶ settings.py 설정 변경
웹사이트에 현재 시간을 설정
TIME_ZONE = 'Asia/Seoul'
정적파일 경로 추가
STATIC_URL = '/static/'
STATIC_ROOT = os.path.join(BASE_DIR, 'static')

■ 데이터베이스 설정하기(기본값 sqlite3)
```
DATABASES = {
        'default': {
            'ENGINE': 'django.db.backends.sqlite3',
            'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
        }
    }
```
■ 데이터베이스 생성
```
(firstproject) > python manage.py migrate
```

■ 서버 구동

```
(firstproject) > python manage.py runserver 0:8000
```



■ Post 모델링

장고 안의 모델은 객체의 특별한 종류로써, 모델을 저장하면 그 내용이 데이터베이스에 저장됨

■ 어플리케이션 제작

python manage.py startapp blog

ex)

```
(firstproject) ~/200django> python manage.py startapp blog
(firstproject) ~/200django>
```

■ 현재까지의 디렉토리 구조

```
200django
├── mysite
|       __init__.py
|       settings.py
|       urls.py
|       wsgi.py
├── manage.py
└── blog
    ├── migrations
    |       __init__.py
    ├── __init__.py
    ├── admin.py
    ├── models.py
    ├── tests.py
    └── views.py
```

■ 프로젝트 파일에 어플리케이션 등록

▶ mysite/settings.py 수정

mysite/settings.py
INSTALLED_APPS = ('django.contrib.admin',

```
        'django.contrib.auth',
        'django.contrib.contenttypes',
        'django.contrib.sessions',
        'django.contrib.messages',
        'django.contrib.staticfiles',
        'blog',
    )
```

■ 블로그 Post(글) 모델링

blog/models.py

```
from django.db import models
from django.utils import timezone


class Post(models.Model):
    author = models.ForeignKey('auth.User')            #작성자
    title = models.CharField(max_length=200)           #제목
    text = models.TextField()                          #본문
    created_date = models.DateTimeField(default=timezone.now)     #작성된 날짜
    published_date = models.DateTimeField(blank=True, null=True) #게시된 날짜

    def publish(self):                                 #출판
        self.published_date = timezone.now()
        self.save()

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

models.CharField - 글자 수가 제한된 텍스트를 정의할 때 사용
models.TextField - 글자 수에 제한이 없는 긴 텍스트를 위한 속성
models.DateTimeField - 날짜와 시간
models.ForeignKey - 다른 모델에 대한 링크

■ Database에 모델링(blog) 추가

▶ DB반영을 위한 파일 생성

```
(firstproject) ~/200django> python manage.py makemigrations blog
```

▶ DB 반영

```
(firstproject) ~/200django> python manage.py migrate blog
```

■ django 관리자

http://localhost:8000/admin 으로 접속

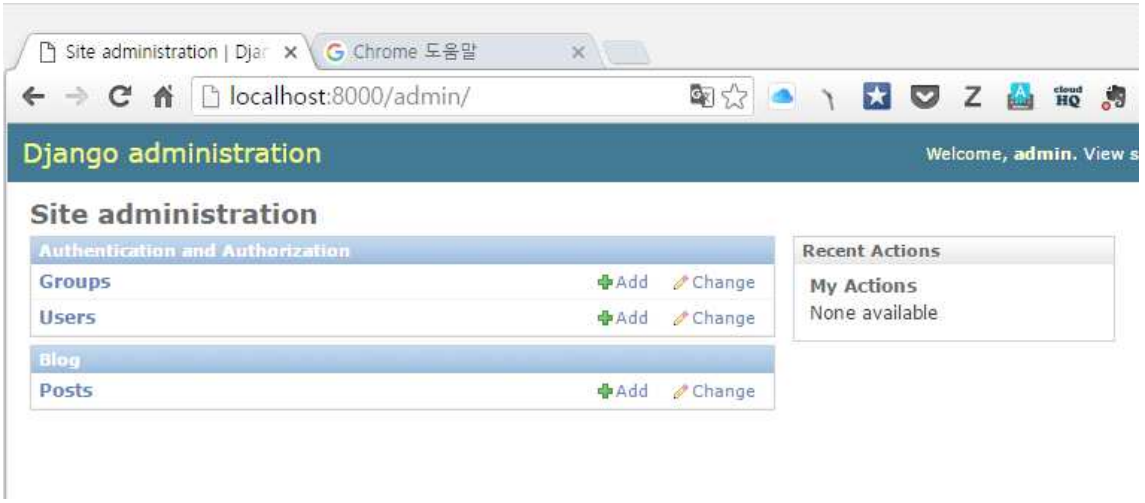
▶ 관리자 계정 등록

```
(firstproject) ~/200django> python manage.py createsuperuser
```

```
(firstproject) C:₩20django>python manage.py createsuperuser
Username (leave blank to use 'sunrin'): admin
Email address: admin@admin.com
Password:
Password (again):
Superuser created successfully.
```

▶ 관리자 계정을 통한 Post 관리등록

blog/admin.py
from django.contrib import admin from .models import Post admin.site.register(Post)



■ Post 작성

1. mysite/settint.py
INSTALLED_APPS = ('django.contrib.admin', 'django.contrib.auth', 'django.contrib.contenttypes', 'django.contrib.sessions', 'django.contrib.messages', 'django.contrib.staticfiles', 'blog',)
2. models.py
from django.db import models

```
from django.utils import timezone


class Post(models.Model):
    author = models.ForeignKey('auth.User')             #작성자
    title = models.CharField(max_length=200)             #제목
    text = models.TextField()                            #본문
    created_date = models.DateTimeField(default=timezone.now)     #작성된 날짜
    published_date = models.DateTimeField(blank=True, null=True) #게시된 날짜

    def publish(self):                                   #출판
        self.published_date = timezone.now()
        self.save()

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

3.admin.py

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

admin.site.register(Post)
```

■ URL 설정

URLconf는 장고에서 URL과 일치하는 뷰를 찾기 위한 패턴들의 집합

mysite/urls.py

```
from django.conf.urls import include, url
    from django.contrib import admin

    urlpatterns = [
        # Examples:
        # url(r'^$', 'mysite.views.home', name='home'),
        # url(r'^blog/', include('blog.urls')),

        url(r'^admin/', include(admin.site.urls)),
        #admin/으로 시작하는 모든 URL을 장고가 view와 대조해 찾아냄

    ]
```

■ 정규표현식

정규표현식
^ 문자열이 시작할 때 $ 문자열이 끝날 때 \d 숫자 + 바로 앞에 나오는 항목이 계속 나올 때 () 패턴의 부분을 저장할 때

예시 자료)
http://www.mysite.com/post/12345/ 여기에서 12345는 글 번호를 의미

뷰마다 모든 글 번호을 작성하지 않고, 정규 표현식으로 url과 매칭되는 글 번호를 뽑을 수 있는 패턴을 만들어 사용

- 정규 표현식 작성 예시> ^post/(\d+)/$.
- ^post/는 장고에게 url 시작점에 (오른쪽부터) post/가 있다는 것을 알려줌 ^)
- (\d+)는 숫자(한 개 또는 여러개) 가 있다는 뜻
- /는 장고에게 /뒤에 문자가 있음 알려줌
- $는 URL의 끝이 방금 전에 있던 /로 끝나야 매칭될 수 있다는 것을 나타냄

mysite/urls.py

```
from django.conf.urls import include, url
    from django.contrib import admin

    urlpatterns = [
        # Examples:
        # url(r'^$', 'mysite.views.home', name='home'),
        # url(r'^blog/', include('blog.urls')),

        url(r'^admin/', include(admin.site.urls)),
        url(r'', include('blog.urls')),    #main url ('')로 blog.urls를 가져옴
]
```

mysite/url.py

↓호출

blog\url.py

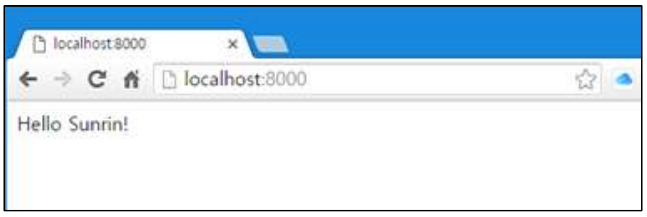
blog/url.py

```
from django.conf.urls import url
from . import views
```

```
urlpatterns = [
    url(r'^$', views.post_list, name='post_list'),
]
```

blog/view.py

```
from django.shortcuts import render
def post_list(request):
        return render(request, 'blog/post_list.html', {})
```



■ 쿼리셋(QuerySet)이란
쿼리셋은 전달받은 모델의 객체 목록
쿼리셋은 데이터베이스로부터 데이터를 읽고, 필터를 걸거나 정렬을 할 수 있다.

■ django shell

```
(firstproject) ~/200django> python manage.py shell
(InteractiveConsole)
>>>
```

▶ 글목록 확인하기

```
>>> from blog.models import Post
>>> Post.objects.all()
[<Post: mysite/settint.py>, <Post: models.py>, <Post: admin.py>]
```

▶ 객체 생성하기

```
>>> from django.contrib.auth.models import User
>>> User.objects.all()
[<User: admin>]
>>> me = User.objects.get(username='ola')
>>> Post.objects.create(author=me, title='sample test',text='sample testing')
<Post: sample test>
>>> Post.objects.all()
[<Post: mysite/settint.py>, <Post: models.py>, <Post: admin.py>, <Post: sample test>]
```

▶ 필터링하기
1) 작성자 기준으로 필터링

```
>>> Post.objects.filter(author=me)
[<Post: mysite/settint.py>, <Post: models.py>, <Post: admin.py>, <Post: sample test>]
```

2) 제목(title) 내용으로 필터링

```
>>> Post.objects.filter( title__contains='.py' )
[<Post: mysite/settint.py>, <Post: models.py>, <Post: admin.py>]
```

3) 출판일 기준으로 필터링

```
>>> from django.utils import timezone
>>> Post.objects.filter(published_date__lte=timezone.now())
[<Post: mysite/settint.py>, <Post: models.py>, <Post: admin.py>]
```

콘솔에서 생성한 글까지 보이게 하기 위해서는

```
>>> post = Post.objects.get(title="sample title")
>>> post.publish()
```

▶ 정렬하기

```
>>> Post.objects.order_by('created_date')   #오름차순 정렬
[<Post: mysite/settint.py>, <Post: models.py>, <Post: admin.py>, <Post: sample test>]
```

```
>>> Post.objects.order_by('-created_date')  #내림차순 정렬
[<Post: sample test>, <Post: admin.py>, <Post: models.py>, <Post: mysite/settint.py>]
```

▶ 쿼리셋(QuerySets) 연결

```
>>> Post.objects.filter(published_date__lte=timezone.now()).order_by('published_date')
```

▶ 쉘(Shell) 종료

```
>>> exit()
```

■ 템플릿의 동적 데이터 반영하기

blog/views.py

```
    from django.shortcuts import render
    from django.utils import timezone    #timezone.now() 함수로 인해 호출
    from .models import Post

    def post_list(request):
        posts = Post.objects.filter(published_date__lte=timezone.now()).order_by
('published_date')
        return render(request, 'blog/post_list.html', {'posts': posts})
```

■ Django 템플릿

▶템플릿 태그

장고 템플릿 태그(Django template tags) 는 파이썬을 HTML로 바꿔주어, 빠르고 쉽게 동적인 웹사이트를 만들 수 있게 도와준다.

▶형식

장고 템플릿 안에 있는 값을 출력하려면, 변수 이름안에 중괄호를 넣어 표시

blog/post_list.html

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">

    </head>
    <body>
<div>
    <h1><a href="/">hello sunrin</a></h1>
</div>
{{ posts }}
    </body>
</html>
```

blog/post_list.html - # 내용 출력하기

```
<div>
    <h1><a href="/">hello sunrin</a></h1>
</div>

{% for post in posts %}
    <div>
        <p>published: {{ post.published_date }}</p>
        <h1><a href="">{{ post.title }}</a></h1>
        <p>{{ post.text|linebreaksbr }}</p>  #|linebreaksbr :파이프 문자(|) 블로그 글 텍스트에서 행이 바뀌면 문단으로 변환하도록 하라는 의미
    </div>
{% endfor %}
```