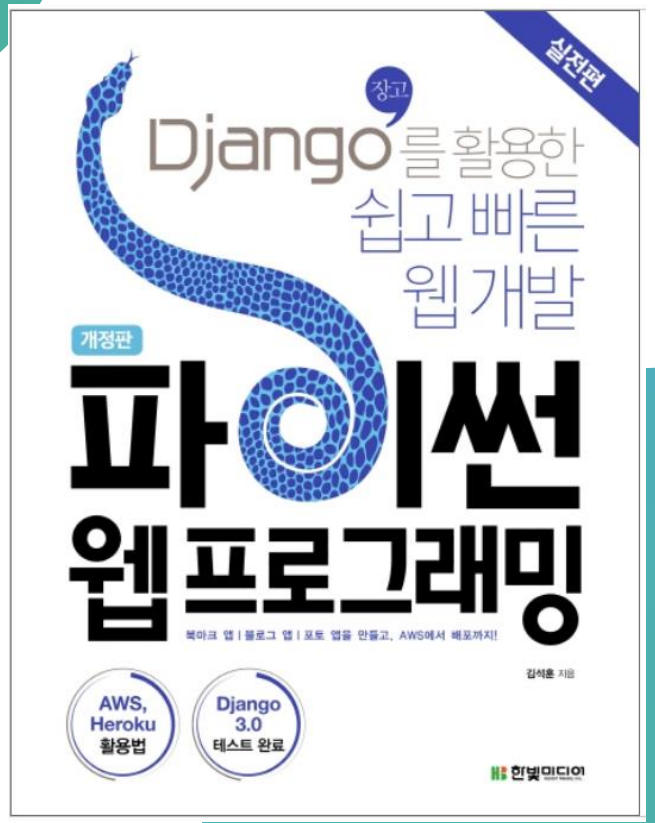
CHAPTER 03 실전 프로그램 개발 – Blog 앱

파이썬 웹프로그래밍



가천대학교 컴퓨터공학과
왕보현

시작하기전에

- **개발 환경**
  - Anaconda
  - Django 3.0.8
  - Python 3.6.10 Anaconda
  - Windows 10
  - 장고 공식 홈페이지 https://docs.djangoproject.com/ (오픈소스)

- **예제 파일 다운로드**
  - http://www.hanbit.co.kr/src/10104

CHAPTER 03 Blog 앱

1. 블로그 앱 생성

```
(vDjBook) (base) C:₩Users₩WBH₩venv₩vDjBook₩ch99>python manage.py startapp blog

(vDjBook) (base) C:₩Users₩WBH₩venv₩vDjBook₩ch99>dir
 C 드라이브의 볼륨: Windows10
 볼륨 일련 번호: C850-E0CA

 C:₩Users₩WBH₩venv₩vDjBook₩ch99 디렉터리

2020-08-17  오후 08:39    <DIR>          .
2020-08-17  오후 08:39    <DIR>          ..
2020-08-17  오후 08:39    <DIR>          blog
2020-08-04  오후 09:15    <DIR>          bookmark
2020-08-04  오후 09:33           139,264 db.sqlite3
2020-08-03  오후 08:53               647 manage.py
2020-08-04  오전 10:48    <DIR>          mysite
               2개 파일             139,911 바이트
               5개 디렉터리  354,078,277,632 바이트 남음
```

2. settings.py에 블로그 앱 등록

```python
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'bookmark.apps.BookmarkConfig',
    'blog.apps.BlogConfig', #추가
]
```

3. 모델 코딩하기- models.py

```python
from django.db import models
from django.urls import reverse


class Post(models.Model):
    title = models.CharField(verbose_name='TITLE',max_length=50)
    slug = models.SlugField('SLUG', unique=True, allow_unicode=True, help_text='one word for title alias.')
    description = models.CharField('DESCRIPTION', max_length=100, blank=True, help_text='simple description text.')
    content = models.TextField('CONTENT')
    create_dt = models.DateTimeField('CREATE DATE', auto_now_add=True)
    modify_dt = models.DateTimeField('MODIFY DATE', auto_now=True)

    class Meta:
        verbose_name = 'post'
        verbose_name_plural = 'posts'
        db_table = 'blog_posts'
        ordering = ('-modify_dt',)

    def __str__(self):
        return self.title

    def get_absolute_url(self):
        return reverse('blog:post_detail', args=(self.slug,))

    def get_previous(self):
        return self.get_previous_by_modify_dt()

    def get_next(self):
        return self.get_next_by_modify_dt()
```

3. 모델 코딩하기- models.py

```
from django.db import models
from django.urls import reverse ①
```

① reverse() 함수를 사용하기 위해 임포트. Reverse() 함수는 URL 패턴을 만들어 주는 장고의 내장함수.
view를 보고 URL을 찾을 수 있는 기능

```
class Post(models.Model):
    title = models.CharField(verbose_name='TITLE', max_length=50) ②
```

② 사용자가 이해하기 쉬운 별칭

```
    slug = models.SlugField('SLUG', unique=True, allow_unicode=True, help_text='one word for title alias.')
```

③ 슬러그 : 페이지나 포스트를 설명하는 핵심 단어의 집합. 슬러그는 보통 제목의 단어들을 하이픈으로 연결해 생성.
슬러그는 페이지나 포스트의 제목에서 조사, 전치사, 쉼표, 마침표 등을 빼고 띄어쓰기는 하이픈으로 대체해서 만들며
URL에 사용됨.

3. 모델 코딩하기- models.py

```
description = models.CharField('DESCRIPTION', max_length=100, blank=True, help_text='simple description text.')
```
④

④ blank=True : 빈칸도 가능 , help_text : 도움말 , Admin 사이트에서 확인 가능

```
content = models.TextField('CONTENT') ⑤
create_dt = models.DateTimeField('CREATE DATE', auto_now_add=True) ⑥
modify_dt = models.DateTimeField('MODIFY DATE', auto_now=True) ⑦
```

⑤ TextField여서 여러줄 입력 가능

⑥ 날짜와 시간을 입력하는 DateTimeField이며, auto_now_add속성은 객체가 생성될 때의 시각을 자동으로 기록

⑦ auto_now속성은 객체가 데이터베이스에 저장될 때의 시각을 자동으로 기록. 즉, 객체가 변경될 때의 시각이 기록

3. 모델 코딩하기- models.py

```
class Meta: ⑧
    verbose_name = 'post' ⑨
    verbose_name_plural = 'posts' ⑩
    db_table = 'blog_posts'    ⑪
    ordering = ('-modify_dt',) ⑫
```

⑧ 필드 속성 외에 필요한 파라미터가 있으면, Meta 내부 클래스로 정의

⑨ 테이블의 별칭은 단수와 복수로 가질 수 있는데, 단수 별칭을 post로 함

⑩ 테이블의 복수 별칭을 posts로 함.

⑪ 데이터베이스에 저장되는 테이블의 이름을 blog_posts로 지정. 이 항목을 지정하지 않았다면 blog_post(앱이름_테이블명소문자)가 됨.

⑫ modify_dt 컬럼을 기준으로 내림차순으로 정렬

```
def __str__(self):    ⑬
    return self.title

def get_absolute_url(self): ⑭
    return reverse('blog:post_detail', args=(self.slug,))
```

⑬ 객체의 문자열 표현 메소드. 객체의 문자열을 객체.title 속성으로 표시

⑭ 이 메소드가 정의된 객체를 지칭하는 URL을 반환함. 메소드 내에서는 장고의 내장 함수인 reverse()를 호출함.

3. 모델 코딩하기- models.py

```
    def get_previous(self):   ⑮
        return self.get_previous_by_modify_dt()

    def get_next(self): ⑯
        return self.get_next_by_modify_dt()
```

⑮ modify_dt 컬럼을 중심으로 이전 데이터 반환 : 현재 modify_dt의 내림차순으로 정렬되어 있으므로 최신 포스트 반환

⑯ 현재 modify_dt의 내림차순으로 정렬되어 있으므로 예전 포스트 반환

4. 모델 코딩하기 - admin.py 에 등록

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

@admin.register(Post)
class PostAdmin(admin.ModelAdmin):   ①
    list_display  = ('id', 'title', 'modify_dt') ②
    list_filter   = ('modify_dt',)               ③
    search_fields = ('title', 'content')         ④
    prepopulated_fields = {'slug': ('title',)}   ⑤
```

① PostAdmin 클래스는 Post테이블이 Admin 싸이트에서 어떤 모습으로 보여줄 지를 정의하는 클래스

② Post의 객체를 보여줄 때 id, title, modify_dt를 화면에 출력하라고 지정

③ modify_dt 컬럼을 사용하는 필터 사이드 바를 보여주도록 지정

④ 검색박스를 표시하고 입력된 단어는 title과 content 컬럼에서 검색하도록 함

⑤ slug 필드는 title 필드를 사용해 미리 채워지도록 함

5. 모델 migrate하기

```
(vDjBook) (base) C:₩Users₩WBH₩venv₩vDjBook₩ch99>python manage.py makemigrations blog
Migrations for 'blog':
  blog₩migrations₩0001_initial.py
    - Create model Post

(vDjBook) (base) C:₩Users₩WBH₩venv₩vDjBook₩ch99>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, blog, bookmark, contenttypes, sessions
Running migrations:
  Applying blog.0001_initial... OK

(vDjBook) (base) C:₩Users₩WBH₩venv₩vDjBook₩ch99>
```

6. URLconf

※ urls.py 파일이 저장된 폴더 꼭 확인

ch99\mysite\urls.py

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


urlpatterns = [
    path('admin/', admin.site.urls),
    # shkim
    path('bookmark/', include('bookmark.urls')),    ①
    path('blog/', include('blog.urls')),            ②
]
```

① include 함수를 이용하여 bookmark 앱의 urls.py로 처리를 위임

② include 함수를 이용하여 blog 앱의 urls.py로 처리를 위임

6. URLconf

ch99\bookmark\urls.py ➔ 새로 만들어 주어야 하는 파일

```
from django.urls import path
from bookmark.views import BookmarkLV, BookmarkDV                    ①

app_name = 'bookmark'                                                ②추가
urlpatterns = [                                                      ③
    path('', BookmarkLV.as_view(), name='index'),                    ④
    path('<int:pk>/', BookmarkDV.as_view(), name='detail'),          ⑤
]
```

① URL conf에서 뷰를 호출하므로, 뷰 모듈의 관련 클래스를 임포트

② 애플리케이션 이름공간을 bookmark로 지정. 이 이름은 URL 패턴의 이름을 정하는 데 사용 예) bookmark.detail
각 애플리케이션 폴더에 생성한 urls.py 파일에는 app_name을 반드시 지정해 줄 것.

③ path()함수의 URL 스트링 부분만 변경되어 있음 URL의 /bookmark/부분은 mysite/urls.py에서 정의 했으므로 여기서는 생략

④ URL /bookmark/ 요청을 처리할 뷰 클래스를 지정. URL 패턴의 이름은 네임스페이스를 포함해 bookmark:index가 됨

⑤ URL /bookmark/숫자/ 요청을 처리할 뷰 클래스를 지정.
숫자 자리에는 레코드의 기본키가 들어감.
URL 패턴의 이름은 네임스페이스를 포함해 bookmark:detail이 됨
URL 스트링에서 추출된 파라미터가 뷰클래스의 메소드에 인자로(예 pk=99) 전달

6. URLconf

ch99\blog\urls.py

```python
from django.urls import path, re_path
from blog import views

app_name = 'blog'
urlpatterns = [

 # Example: /blog/
 path('', views.PostLV.as_view(), name='index'),

 # Example: /blog/post/ (same as /blog/)
 path('post/', views.PostLV.as_view(), name='post_list'),

 # Example: /blog/post/django-example/
 re_path(r'^post/(?P<slug>[-\w]+)/$' , views.PostDV.as_view(), name='post_detail'),

 # Example: /blog/archive/
 path('archive/', views.PostAV.as_view(), name='post_archive'),

 # Example: /blog/archive/2019/
 path('archive/<int:year>/', views.PostYAV.as_view(), name='post_year_archive'),

 # Example: /blog/archive/2019/nov/
 path('archive/<int:year>/<str:month>/', views.PostMAV.as_view(), name='post_month_archive'),

 # Example: /blog/archive/2019/nov/10/
 path('archive/<int:year>/<str:month>/<int:day>/', views.PostDAV.as_view(), name='post_day_archive'),

 # Example: /blog/archive/today/
 path('archive/today/', views.PostTAV.as_view(), name='post_today_archive'),
]
```

① URL의 정규 표현식
다음 페이지 참고

```
r ' ^  post/  (?P  <slug>  [-  \w] + ) / $ '
예) blog/post/I-<am>-boy/
```

② 인자가 두 개 있는 경우

URL 의 정규표현식

```
(?P  <slug>  .  )
```

```
‘ ^ post/ (?P  <slug> [-\w]+ )/ $
blog/post/I-<am>-boy/
```

^ 문자열의 시작
$ 문자열의 마지막

.	아무 글자나 한글자
[]	대괄호 안에 있는 문자들 중 한글자
[a-z]	a부터 z까지 임의의 한글자
₩b	공백, 영문자, 숫자가 아닌 문자 하나
₩B	₩b의 반대문자 하나
₩d	숫자 하나 ([0-9]와 동일)
₩D	₩d의 반대문자 하나
₩w	영문, 숫자, _ 중 하나
₩W	₩w의 반대문자중 하나
₩s	공백, 탭중 하나
₩S	₩s와 반대문자중 하나

*	최소 0번 이상 반복 ({0, } 과 동일)
+	최소 1번 이상 반복 ({1, } 과 동일)
?	0번 또는 1번 ({0, 1} 과 동일)
|	또는 (A|B : A 또는 B)
r‘ ’	RAW모드 (작은따옴표 사이에 정규식을 작성)
	자동 이스케이프
	<, >, “, ‘ 등의 문자를 html 표현식으로 변환

{n}	n번 반복
{m, n}	최소 m번에서 최대 n번 반복 (n 생략시 m번이상 반복)

[출처] https://m.blog.naver.com/PostView.nhn?blogId=kgw1988&logNo=220944485372&proxyReferer=https:%2F%2Fwww.google.com%2F

[출처]https://velog.io/@devzunky/TIL-no.40-Django-Basic-4-url-mapping-2

URL 의 정규표현식

```
re_path(r ‘ ^  articles/  ( ?P <year>  [0-9] {4}  ) / $‘  )
```

정규표현식을 사용해서 url을 읽을 때는

일반 path함수가 아니라 re_path함수를 사용합니다.

그리고 첫번째 인자에 string을 적기 전에

r로 먼저 시작한뒤 string을 적어줍니다.

'^articles/ : article로 시작한다.

?P : 파라메터 이다.

<year> : year 변수명

[0-9] : 0에서 9 중 한 글자

{4} : [0-9]를 4번 반복한다.

$: 종료 문자

```
r‘^article/(?P<id>\d+)/$’ ➔  r‘  ^  article/  (?P  <id>  \d  + )  /  $  ’
```

```
r'^article/(?P<id>.*)/$’  ➔  r‘  ^  article/  (?P  <id>  .  * )  /  $  ’
```

[출처] https://m.blog.naver.com/PostView.nhn?blogId=kgw1988&logNo=220944485372&proxyReferer=https:%2F%2Fwww.google.com%2F

7. views.py

ch99\blog\views.py

```
from django.views.generic import ListView, DetailView
from django.views.generic import ArchiveIndexView, YearArchiveView, MonthArchiveView
from django.views.generic import DayArchiveView, TodayArchiveView

from blog.models import Post

#--- ListView
class PostLV(ListView):
    model = Post
    template_name = 'blog/post_all.html'
    context_object_name = 'posts'
    paginate_by = 2

#--- DetailView
class PostDV(DetailView):
    model = Post

#--- ArchiveView
class PostAV(ArchiveIndexView):
    model = Post
    date_field = 'modify_dt'

class PostYAV(YearArchiveView):
    model = Post
    date_field = 'modify_dt'
    make_object_list = True
```

select * from Post

select * from Post where slug = <slug>

select modify_dt from Post order by modify_dt desc

select substring(modify_dt, 5,2) ➔ 월
from Post where substring(modify_dt,1,4) = <year>

select *
from Post where substring(modify_dt,1,4) = <year>

select substring(modify_dt, 7,2) ➔ 일
from Post
where substring(modify_dt,1,4) = <year>
    and substring(modify_dt,5,2) = <month>

```
class PostMAV(MonthArchiveView):
    model = Post
    date_field = 'modify_dt'

class PostDAV(DayArchiveView):
    model = Post
    date_field = 'modify_dt'

class PostTAV(TodayArchiveView):
    model = Post
    date_field = 'modify_dt'
```

select *
from Post
where substring(modify_dt,1,4) = <year>
    and substring(modify_dt,5,2) = <month>
    and substring(modify_dt,7,2) = <day>

select *
from Post
where modify_dt = today

7. views.py

ch99\blog\views.py

```
from django.views.generic import ListView, DetailView
from django.views.generic import ArchiveIndexView, YearArchiveView, MonthArchiveView      ①
from django.views.generic import DayArchiveView, TodayArchiveView
```

① 뷰 작성에 필요한 클래스형 제네릭 뷰를 import 함

```
from blog.models import Post          ②
```

② 테이블 조회를 위해 Post 모델 클래스를 import 함

7. views.py

ch99\blog\views.py

```
#--- ListView
class PostLV(ListView): ③
    model = Post     ④
    template_name = 'blog/post_all.html' ⑤
    context_object_name = 'posts'   ⑥
    paginate_by = 2      ⑦
```

③ ListView 제네릭 뷰를 상속받아 PostLV 클래스형 뷰를 정의함. ListView 제네릭 뷰는 테이블로부터 객체(테이블 레코드) 리스트를 가져와 그 리스트를 컨텍스트 변수에 저장함.

④ PostLV 클래스의 대상 테이블을 Post로 함

⑤ 템플릿 파일을 post_all.html로 지정. 만일 지정하지 않으면 디폴트 템플릿 파일명은 post_list.html 로 됨

⑥ 템플릿 파일로 넘겨주는 컨텍스트 변수명을 posts로 함.
이것을 지정하지 않더라도 디폴트 컨텍스트 변수명인 object_list 역시 사용 가능

⑦ 한 페이지에 보여주는 객체(테이블의 레코드) 리스트의 숫자는 2 임.
클래스형 뷰에서는 이렇게 paginate_by 속성을 정의하는 것만으로도 장고가 제공하는 페이징 기능을 사용 가능.
페이징 기능이 활성화되면 객체 리스트 하단에 페이지를 이동할 수 있는 버튼을 만들 수 있음.

7. views.py

ch99\blog\views.py

```
#--- DetailView
class PostDV(DetailView):  ⑧
    model = Post     ⑨
```

⑧ DetailView 제네릭 뷰를 상속받아 PostDV 클래스형 뷰를 정의함. DetailView 제네릭 뷰는 테이블로부터 특정 객체(테이블의 특정 레코드)를 가져와 그 객체의 상세 정보를 컨텍스트 변수에 저장함. 테이블에서 특정 객체를 조회하기 위한 키는 기본 키 대신 slug 속성을 사용하고 있음. 이 slug 파라미터는 URL conf에서 추출해서 뷰로 넘겨줌.

⑨ PostDV 클래스의 대상 테이블을 Post로 함.
다른 속성들을 지정하지 않았기 때문에 컨텍스트 변수는 object이고
템플릿 파일은 post_detail.html 임.

7. views.py

ch99\blog\views.py

```
#--- ArchiveView
class PostAV(ArchiveIndexView):  ⑩
    model = Post                 ⑪
    date_field = 'modify_dt'     ⑫
```

⑩ ArchiveIndexView 제네릭 뷰를 상속받아 PostAV 클래스형 뷰를 정의. ArchiveIndexView 제네릭 뷰는 테이블로부터 객체(레코드) 리스트를 가져와, 날짜 필드를 기준으로 최신 객체를 먼저 출력함.

⑪ PostAV 클래스의 대상 테이블은 Post 테이블

⑫ 기준이 되는 날짜 필드는 modify_dt 컬럼을 사용함. 즉, 변경 날짜가 최근인 포스트를 먼저 출력

7. views.py

ch99\blog\views.py

```
class PostYAV(YearArchiveView):  ⑬
    model = Post                 ⑭
    date_field = 'modify_dt'     ⑮
    make_object_list = True      ⑯
```

⑬ YearArchiveView 제네릭 뷰를 상속받아 PostYAV 클래스형 뷰를 정의. YearArchiveView 제네릭 뷰는 테이블로부터 날짜 필드의 연도를 기준으로 객체 리스트를 가져와 그 객체들이 속한 월을 리스트로 출력함.
날짜 필드의 연도 파라미터는 URLconf에서 추출해 뷰로 넘겨줌.

⑭ PostYAV 클래스의 대상 테이블은 Post 테이블

⑮ 기준이 되는 날짜 필드는 modify_dt 컬럼을 사용. 즉 변경 날짜가 YYYY 연도인 포스트를 검색해, 그 포스트들의 변경 월을 컨텍스트 변수에 저장

⑯make_object_list 속성이 True면 해당 연도에 해당하는 객체의 리스트를 만들어서 템플릿에 넘겨줌. 즉, 템플릿 파일에서 object_list 컨텍스트 변수를 사용할 수 있음. 디폴트는 false

7. views.py

ch99\blog\views.py

```
class PostMAV(MonthArchiveView):    ⑰
    model = Post                    ⑱
    date_field = 'modify_dt'        ⑲
```

⑰ MonthArchiveView 제네릭 뷰를 상속받아 PostMAV 클래스형 뷰를 정의함. MonthArchiveView 제네릭 뷰는 테이블로부터 날짜 필드의 연월을 기준으로 객체 리스트를 가져와 그 리스트를 출력함.
날짜 필드의 연도 및 월 파라미터는 URLconf에서 추출해 뷰로 넘겨줌.

⑱ PostMAV 클래스의 대상 테이블을 Post로 함.

⑲ 기준이 되는 날짜 필드는 modify_dt 컬럼을 사용. 즉, 변경 날짜의 연월을 기준으로 포스트를 검색해 그 포스트들의 리스트를 출력함

7. views.py

ch99\blog\views.py

```
class PostDAV(DayArchiveView):      ⑳
    model = Post                    ㉑
    date_field = 'modify_dt'        ㉒
```

⑳ DayArchiveView 제네릭 뷰를 상속받아 PostDAV 클래스형 뷰를 정의함. DayArchiveView 제네릭 뷰는 테이블로부터 날짜 필드의 연월일을 기준으로 객체 리스트를 가져와 그 리스트를 출력함.
날짜 필드의 연도 , 월 , 일 파라미터는 URLconf에서 추출해 뷰로 넘겨줌.

㉑ PostDAV 클래스의 대상 테이블을 Post로 함.

㉒ 기준이 되는 날짜 필드는 modify_dt 컬럼을 사용. 즉, 변경 날짜의 연월일을 기준으로 포스트를 검색해 그 포스트들의 리스트를 출력함

7. views.py

ch99\blog\views.py

```
class PostTAV(TodayArchiveView):    ㉓
    model = Post                    ㉔
    date_field = 'modify_dt'        ㉕
```

㉓ TodayArchiveView 제네릭 뷰를 상속받아 PostTAV 클래스형 뷰를 정의함. TodayArchiveView 제네릭 뷰는 테이블로부터 날짜 필드가 오늘인 객체 리스트를 가져와 그 리스트를 출력함. TodayArchiveView는 오늘 날짜를 기준 연월일로 지정한다는 점 이외에는 DayArchiveView와 동일함

㉔ PostTAV 클래스의 대상 테이블을 Post로 함.

㉕ 기준이 되는 날짜 필드는 modify_dt 컬럼을 사용. 즉, 변경 날짜가 오늘인 포스트를 검색해 그 포스트들의 리스트를 출력함

8. templates 파일들

ch99\blog\templates\blog\post_all.html

```
<h1>Blog List</h1>
<br>

{% for post in posts %}
    <h3><a href='{{ post.get_absolute_url }}'>{{ post.title }}</a></h3>
    {{ post.modify_dt|date:"N d, Y" }}
    <p>{{ post.description }}</p>
{% endfor %}

<br>

<div>
    <span>
        {% if page_obj.has_previous %}
            <a href="?page={{ page_obj.previous_page_number }}">PreviousPage</a>
        {% endif %}

        Page {{ page_obj.number }} of {{ page_obj.paginator.num_pages }}

        {% if page_obj.has_next %}
            <a href="?page={{ page_obj.next_page_number }}">NextPage</a>
        {% endif %}
    </span>

</div>
```

※ get_absolute_url() 메소드는 모델 클래스의 메소드로 정의되어 있어야 사용 가능함. 이 메소드를 정의할 때 reverse() 함수를 사용하고 reverse()함수의 인자로 URL 패턴명을 사용하고 있음

① date 필드를 이용 표시 형식은 "N d, Y" 임 (예 : July 05, 2019)

> >

8. templates 파일들

ch99\blog\templates\blog\post_all.html

Blog List

Django

Sept. 05, 2020

Django

Bohyun Wang Positive

Aug. 17, 2020

Mrs. Wang need to positive thinking

Page 1 of 2 NextPage

8. templates 파일들

views.py

```python
#--- ListView
class PostLV(ListView):
    model = Post
    template_name = 'blog/post_all.html'
    context_object_name = 'posts'
    paginate_by = 2
```

post_all.html

```
<h1>Blog List</h1>
<br>

{% for post in posts %}
    <h3><a href='{{ post.get_absolute_url }}'>{{ post.title }}</a></h3>
    {{ post.modify_dt|date:"N d, Y" }}
    <p>{{ post.description }}</p>
{% endfor %}

<br>
```

posts 변수에는
Post 테이블의 모든
레코드가 저장

models.py

```python
def get_absolute_url(self):
    return reverse('blog:post_detail', args=(self.slug,))
```

8. templates 파일들

ch99\blog\templates\blog\post_all.html

```
<div>
    <span>
        {% if page_obj.has_previous %}
            <a href="?page={{ page_obj.previous_page_number }}">PreviousPage</a>
        {% endif %}

        Page {{ page_obj.number }} of {{ page_obj.paginator.num_pages }}

        {% if page_obj.has_next %}
            <a href="?page={{ page_obj.next_page_number }}">NextPage</a>
        {% endif %}
    </span>

</div>
```

② page_obj : 장고의 Page 객체가 들어 있는 컨텍스트 변수. 현재 페이지를 기준으로 이전 페이지가 있는지 확인

③ previous_page_number 라는 텍스트를 출력하고, 이 텍스트에 URL 링크를 연결 예 > ?page=3

④ page_obj.number 는 현재 페이지 번호

⑤ page_obj.paginator.num_pages 는 총 페이지 수

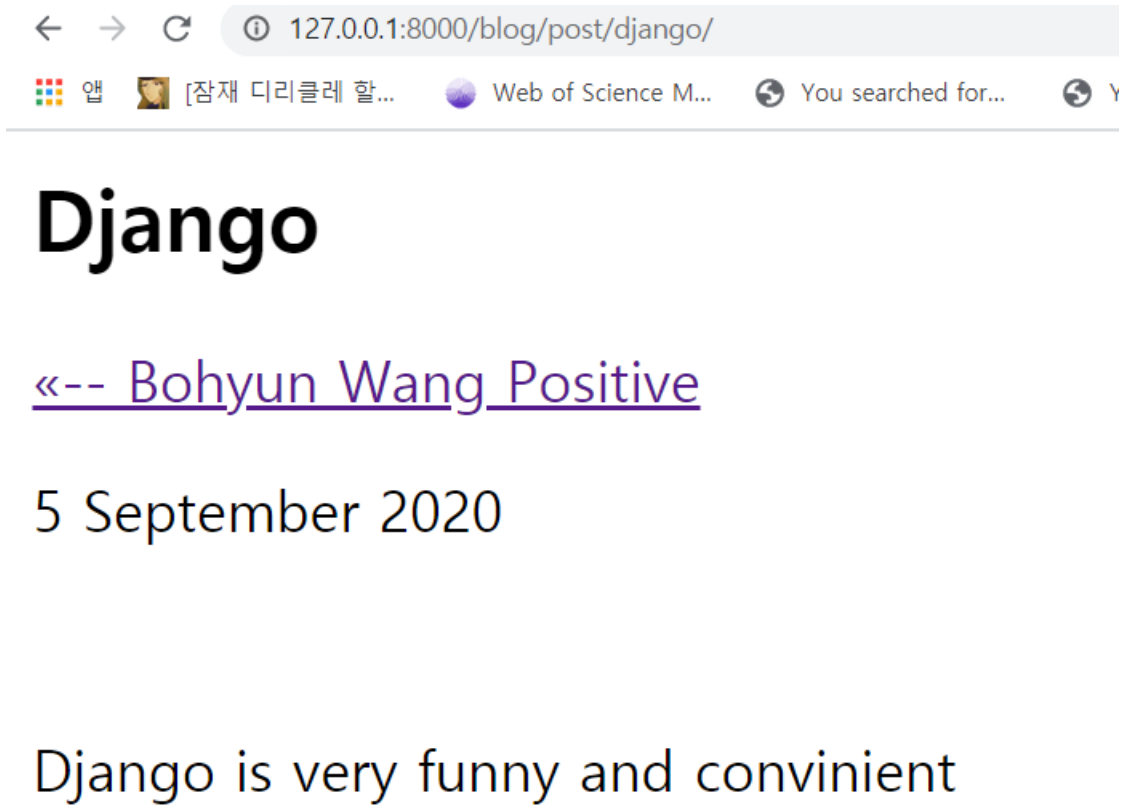
8. templates 파일들

ch99\blog\templates\blog\post_detail.html

```
<h2>{{ object.title }}</h2>

<p>
    {% if object.get_previous %}
    <a href="{{ object.get_previous.get_absolute_url }}" title="View previous post">
        &laquo;-- {{ object.get_previous }}
    </a>
    {% endif %}

    {% if object.get_next %}
    | <a href="{{ object.get_next.get_absolute_url }}" title="View next post">
    {{ object.get_next }} --&raquo;
    </a>
    {% endif %}
</p>

<p>{{ object.modify_dt|date:"j F Y" }}</p>
<br>

<div>
    {{ object.content|linebreaks }}
</div>
```

8. templates 파일들

ch99\blog\templates\blog\post_detail.html



8. templates 파일들

ch99\blog\templates\blog\post_detail.html

```
<h2>{{ object.title }}</h2>
<p>
    {% if object.get_previous %}
    <a href="{{ object.get_previous.get_absolute_url }}" title="View previous post">
        &laquo;-- {{ object.get_previous }}
    </a>
    {% endif %}
```

① object 객체는 PostDV 클래스형 뷰에서 컨텍스트 변수로 넘겨주는 Post 클래스의 특정 객체

② get_previous 함수는 modify_dt 컬럼 기준으로 이전 객체를 반환함. 즉, 변경 날짜가 현재 객체보다 더 최신 객체가 있는지 확인함.

③ get_previous 함수는 이전 객체(포스트)를, get_previous.get_absolute_url 함수는 이전 객체를 지칭하는 URL 패턴을 반환

`{{ object.get_previous }}` 코드는 models.py 파일의

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

코드에 의해 title이 출력됨.

8. templates 파일들

ch99\blog\templates\blog\post_detail.html

```
    {% if object.get_next %}   ④
    | <a href="{{ object.get_next.get_absolute_url }}" title="View next post">
    {{ object.get_next }} --&raquo;   ⑤
    </a>
    {% endif %}
</p>
```

④ get_next 함수는 modify_dt 컬럼 기준으로 다음 객체를 반환함. 즉, 변경 날짜가 현재 객체보다 더 오래된 객체가 있는지 확인함.

⑤ get_enxt 함수는 다음 객체(포스트)를, get_next.get_absolute_url 함수는 다음 객체를 지칭하는 URL 패턴을 반환

`{{ object.get_next }}` 코드는 `models.py` 파일의

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

코드에 의해 `title`이 출력됨.

8. templates 파일들

ch99\blog\templates\blog\post_detail.html

```
<p>{{ object.modify_dt|date:"j F Y" }}</p>  ⑥
<br>

<div>
    {{ object.content|linebreaks }}  ⑦
</div>
```

⑥ 객체의 modify_dt 속성값을 "j F Y" 포맷으로 출력함 (예: 12 July 2015)

⑦ 포스트 객체의 내용을 출력함. linebreaks 템플릿 필터는 ₩n을 인식하게 함.

장고 날짜 시간 포맷 문자

https://docs.djangoproject.com/en/3.1/ref/templates/builtins/#ref-templates-builtins-filters

Format character	Description	Example output
Day		
d	Day of the month, 2 digits with leading zeros.	**'01'** to **'31'**
j	Day of the month without leading zeros.	**'1'** to **'31'**
D	Day of the week, textual, 3 letters.	**'Fri'**
l	Day of the week, textual, long.	**'Friday'**
S	English ordinal suffix for day of the month, 2 characters.	**'st'**, **'nd'**, **'rd'** or **'th'**
w	Day of the week, digits without leading zeros.	**'0'** (Sunday) to **'6'** (Saturday)
z	Day of the year.	**1** to **366**
Week		
W	ISO-8601 week number of year, with weeks starting on Monday.	**1**, **53**

장고 날짜 시간 포맷 문자

https://docs.djangoproject.com/en/3.1/ref/templates/builtins/#ref-templates-builtins-filters

Format character	Description	Example output
Month		
`m`	Month, 2 digits with leading zeros.	`'01'` to `'12'`
`n`	Month without leading zeros.	`'1'` to `'12'`
`M`	Month, textual, 3 letters.	`'Jan'`
`b`	Month, textual, 3 letters, lowercase.	`'jan'`
`E`	Month, locale specific alternative representation usually used for long date representation.	`'listopada'` (for Polish locale, as opposed to `'Listopad'`)
`F`	Month, textual, long.	`'January'`
`N`	Month abbreviation in Associated Press style. Proprietary extension.	`'Jan.'`, `'Feb.'`, `'March'`, `'May'`
`t`	Number of days in the given month.	`28` to `31`

장고 날짜 시간 포맷 문자

https://docs.djangoproject.com/en/3.1/ref/templates/builtins/#ref-templates-builtins-filters

Format character	Description	Example output
Year		
y	Year, 2 digits.	`'99'`
Y	Year, 4 digits.	`'1999'`
L	Boolean for whether it's a leap year.	`True` or `False`
o	ISO-8601 week-numbering year, corresponding to the ISO-8601 week number (W) which uses leap weeks. See Y for the more common year format.	`'1999'`

장고 날짜 시간 포맷 문자

Format character	Description	Example output
Time		
g	Hour, 12-hour format without leading zeros.	'1' to '12'
G	Hour, 24-hour format without leading zeros.	'0' to '23'
h	Hour, 12-hour format.	'01' to '12'
H	Hour, 24-hour format.	'00' to '23'
i	Minutes.	'00' to '59'
s	Seconds, 2 digits with leading zeros.	'00' to '59'
u	Microseconds.	000000 to 999999
a	'a.m.' or 'p.m.' (Note that this is slightly different than PHP's output, because this includes periods to match Associated Press style.)	'a.m.'
A	'AM' or 'PM'.	'AM'
f	Time, in 12-hour hours and minutes, with minutes left off if they're zero. Proprietary extension.	'1', '1:30'
P	Time, in 12-hour hours, minutes and 'a.m.'/'p.m.', with minutes left off if they're zero and the special-case strings 'midnight' and 'noon' if appropriate. Proprietary extension.	'1 a.m.', '1:30 p.m.', 'midnight', 'noon', '12:30 p.m.'

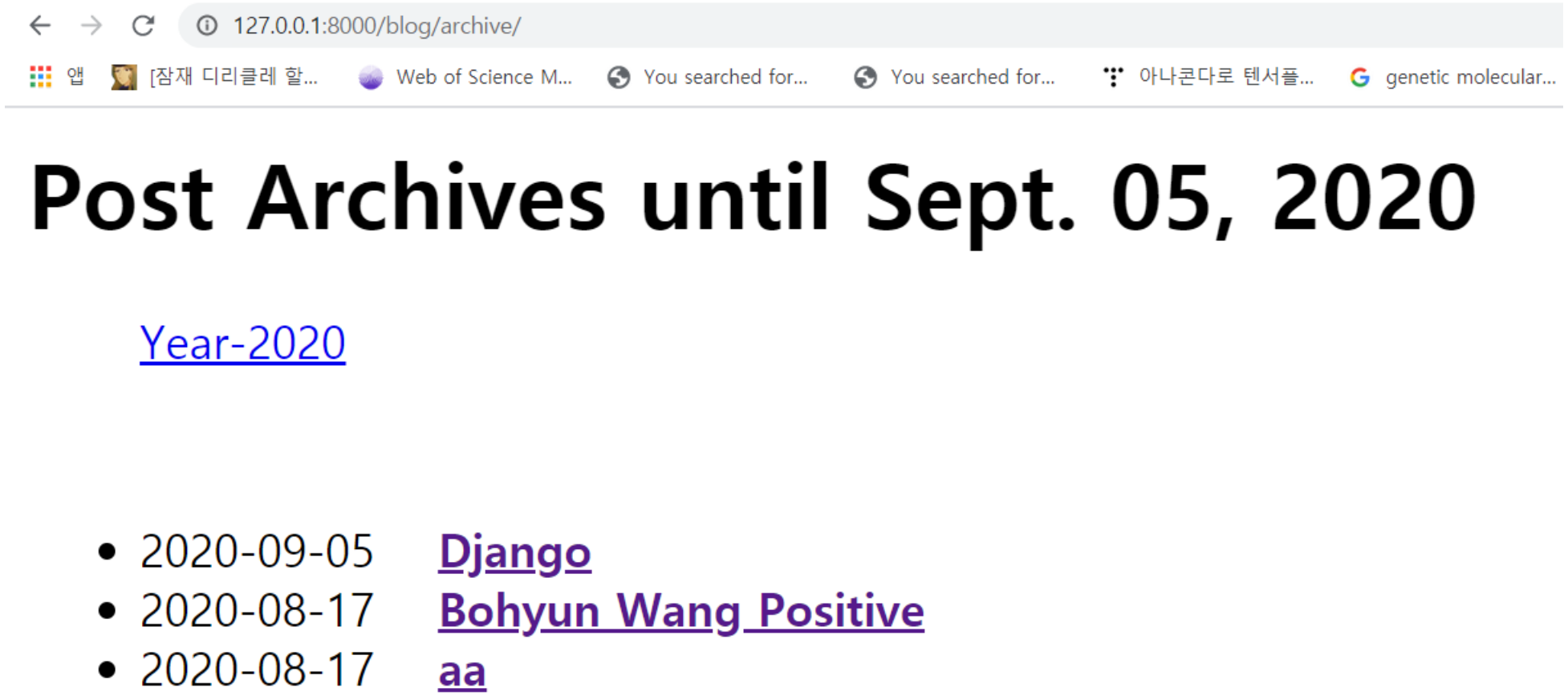
8. templates 파일들

ch99\blog\templates\blog\post_archive.html

```
<h1>Post Archives until {% now "N d, Y" %}</h1>
<ul>
    {% for date in date_list %}
    <li style="display: inline;">
        <a href="{% url 'blog:post_year_archive' date|date:'Y' %}">Year-{{ date|date:"Y" }}</a></li>
    {% endfor %}
</ul>
<br/>

<div>
    <ul>
        {% for post in object_list %}
        <li>{{ post.modify_dt|date:"Y-m-d" }}   
        <a href="{{ post.get_absolute_url }}"><strong>{{ post.title }}</strong></a></li>
        {% endfor %}
    </ul>
</div>
```

8. templates 파일들

ch99\blog\templates\blog\post_archive.html



8. templates 파일들

ch99\blog\templates\blog\post_archive.html

```
<h1>Post Archives until {% now "N d, Y" %}</h1>①
<ul>
    {% for date in date_list %}                ②
    <li style="display: inline;">
        <a href="{% url 'blog:post_year_archive' date|date:'Y' %}">Year-{{ date|date:"Y" }}</a></li> ③
    {% endfor %}
</ul>
<br/>
```

① {% now %} 템플릿 태그는 현재의 날짜와 시간을 원하는 포맷으로 출력. 포맷 문자열을 인자로 받음.
"N d, Y" 포맷 문자열은 July 18, 2015 형식

② data_list 컨텍스트 변수는 DateQuerySet 객체 리스트를 담고 있음. DateQuerySet 객체 리스트는 QuerySet 객체 리스트에서 날짜 정보만을 추출해 담고 있는 객체 리스트임. DateQuerySet에 들어 있는 객체는 datetime.date 타입의 객체임

③ 년도 메뉴는 "Y", 즉 YYYY 형식의 텍스트로, 해당 년도에 생성 또는 수정된(modify_dt) 포스트를 보여주는 URL이 링크되어 있음.

8. templates 파일들

ch99\blog\templates\blog\post_archive.html

```
<div>
    <ul>
        {% for post in object_list %}  ④
        <li>{{ post.modify_dt|date:"Y-m-d" }}     ⑤
        <a href="{{ post.get_absolute_url }}"><strong>{{ post.title }}</strong></a></li>  ⑥
        {% endfor %}
    </ul>
</div>
```

④ 디폴트 컨텍스트 변수로 object_list와 latest 둘 다 가능하고 여기에는 뷰에서 넘겨준 객체 리스트가 저장되어 있음

⑤ <li> : 순서 없는 리스트 , Y-m-d 포맷 문자열은 2019-05-05 형식으로 출력, 세 개의 공백 후 title 보여줌

⑥ title에 get_absolute_url() 메소드를 사용해 해당 포스트를 지정하는 URL 을 링크 함.

8. templates 파일들

ch99\blog\templates\blog\post_archive_year.html

```
<h1>Post Archives for {{ year|date:"Y" }}</h1>

<ul>
    {% for date in date_list %}
    <li style="display: inline;">
        <a href="{% url 'blog:post_month_archive' year|date:'Y' date|date:'b' %}">{{ date|date:"F" }}</a></li>
    {% endfor %}
</ul>
<br>

<div>
    <ul>
        {% for post in object_list %}
        <li>{{ post.modify_dt|date:"Y-m-d" }}   
        <a href="{{ post.get_absolute_url }}"><strong>{{ post.title }}</strong></a></li>
        {% endfor %}
    </ul>
</div>
```

8. templates 파일들

ch99\blog\templates\blog\post_archive_year.html

Post Archives for 2020

August September

- 2020-09-05 Django
- 2020-08-17 Bohyun Wang Positive
- 2020-08-17 aa

8. templates 파일들

ch99\blog\templates\blog\post_archive_year.html

```
<h1>Post Archives for {{ year|date:"Y" }}</h1>  ①

<ul>
    {% for date in date_list %}                 ②
    <li style="display: inline;">
        <a href="{% url 'blog:post_month_archive' year|date:'Y' date|date:'b' %}">{{ date|date:"F" }}</a></li> ③
    {% endfor %}
</ul>
<br>
```

① year 컨텍스트 변수는 해당 연도에 대한 datatime.date 타입의 객체. Y 포맷 문자열은 2019 형식임.

② data_list 컨텍스트 변수는 DateQuerySet 객체 리스트를 담고 있음. DateQuerySet 객체 리스트는 QuerySet 객체 리스트에서 날짜 정보만을 추출해 담고 있는 객체 리스트임. DateQuerySet에 들어 있는 객체는 datetime.date 타입의 객체임

③ 월 메뉴는 "F", 즉 July 형식의 텍스트로, 해당 연월에 생성 또는 수정된(modify_dt) 포스트를 보여주는 URL이 링크되어 있음.

8. templates 파일들

ch99\blog\templates\blog\post_archive_month.html

```
<h1>Post Archives for {{ month|date:"N, Y" }}</h1>

<div>
    <ul>
        {% for post in object_list %}
        <li>{{ post.modify_dt|date:"Y-m-d" }}   
        <a href="{{ post.get_absolute_url }}"><strong>{{ post.title }}</strong></a></li>
        {% endfor %}
    </ul>
</div>
```

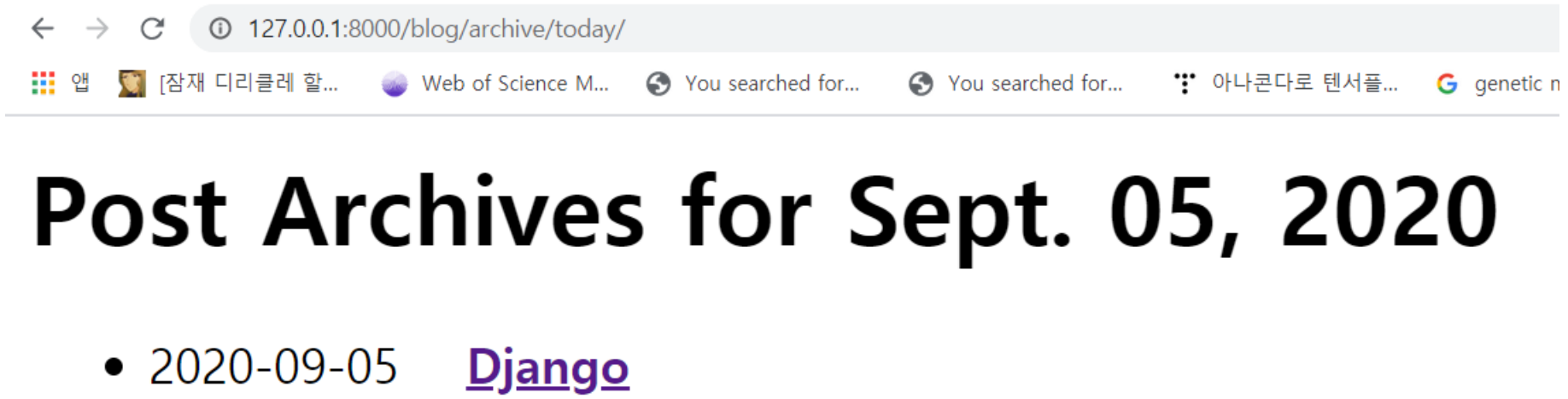
8. templates 파일들

ch99\blog\templates\blog\post_archive_month.html

8. templates 파일들

ch99\blog\templates\blog\post_archive_day.html

```
<h1>Post Archives for {{ day|date:"N d, Y" }}</h1>

<div>
    <ul>
        {% for post in object_list %}
        <li>{{ post.modify_dt|date:"Y-m-d" }}   
        <a href="{{ post.get_absolute_url }}"><strong>{{ post.title }}</strong></a></li>
        {% endfor %}
    </ul>
</div>
```



9. 제네릭 view 클래스들

(1) 제네릭 view 종류