05장 실습 예제 확장하기

- 3장의 예제를 확장하여 클래스형 뷰를 사용하는 애플리케이션을 만들어 본다.

- books라는 새로운 애플리케이션을 만들어 보면서 클래스형 뷰의 사용법을 익히고, 그 다음에 함수형 뷰로 되어 있는 기존의 polls 애플리케이션을 클래스형 뷰로 변경하는 예제를 실습한다.

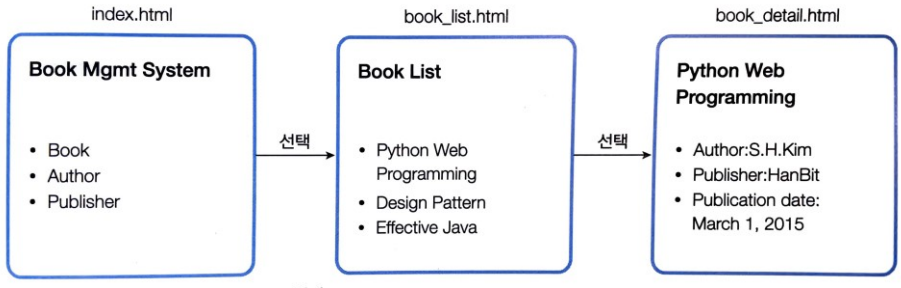
5.1 새로운 애플리케이션 만들기

- 이번 장에서는 books라는 새로운 애플리케이션을 코딩한다. books 애플리케이션은 책을 출판하는데 필요한 정보들인 책, 저자, 출판사에 대한 정보들을 관리하는 웹 애플리케이션이다.

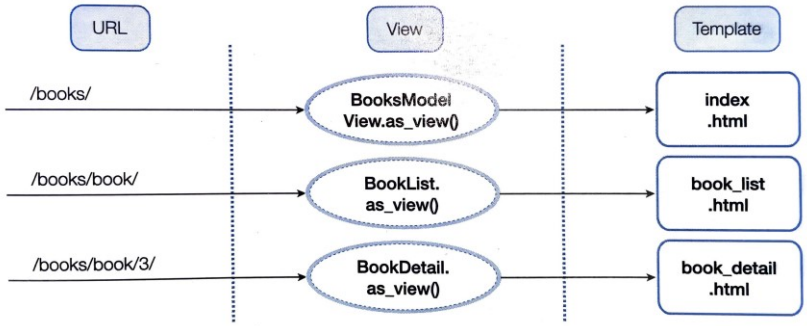
- 순서
  - 애플리케이션 설계하기
  - 프로젝트 뼈대 만들기
  - 애플리케이션 - Model 코딩하기
  - 애플리케이션 - URLconf 코딩하기
  - 애플리케이션 - View 코딩하기
  - 애플리케이션 - Template 코딩하기

5.1.1 애플리케이션 설계하기

- books 애플리케이션 - UI 설계



- books 애플리케이션 - 뷰 흐름 설계



5.1.2 프로젝트 뼈대 만들기 - 애플리케이션 추가

■ 다음 명령를 실행하여 애플리케이션을 생성한다.

```
D:\dev\workspace\django1\ch5Lab>python manage.py startapp books

D:\dev\workspace\django1\ch5Lab>dir
 D 드라이브의 볼륨: disk2
 볼륨 일련 번호: 8493-73C8

 D:\dev\workspace\django1\ch5Lab 디렉터리

2021-11-12  오후 03:46    <DIR>          .
2021-11-12  오후 03:46    <DIR>          ..
2021-11-12  오후 03:46    <DIR>          books
2021-11-11  오후 10:22           143,360 db.sqlite3
2021-11-11  오전 11:59               684 manage.py
2021-11-12  오후 03:44    <DIR>          mysite
2021-11-12  오후 03:44    <DIR>          polls
               2개 파일             144,044 바이트
               5개 디렉터리  240,847,958,016 바이트 남음
```

■ mysite/settings.py 파일에 books 애프리케이션 등록
- 11: 프로젝트에 포함되는 모든 애플리케이션은 설정 파일에 등록해야 한다.

[ch5Lab/mysite/settings.py]

```
01  ...(생략)...
02
03  INSTALLED_APPS = [
04      'django.contrib.admin',
05      'django.contrib.auth',
06      'django.contrib.contenttypes',
07      'django.contrib.sessions',
08      'django.contrib.messages',
09      'django.contrib.staticfiles',
10      'polls.apps.PollsConfig',          # 추가
11      'books.apps.BooksConfig',          # 추가
12  ]
13
14  ..(생략)...
```

5.1.3 애플리케이션 - Model 코딩하기

■ 1. 테이블에 대해 설계된 내용에 따라 models.py 파일에 테이블을 정의한다.

[ch5Lab/books/models.py]

```
01  from django.db import models
02
03
04  class Book(models.Model):
05      title = models.CharField(max_length=100)
06      authors = models.ManyToManyField('Author')
07      publisher = models.ForeignKey('Publisher', on_delete=models.CASCADE)
08      publication_date = models.DateField()
09
```

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


class Author(models.Model):
    name = models.CharField(max_length=50)
    salutation = models.CharField(max_length=100)
    email = models.EmailField()

    def __str__(self):
        return self.name


class Publisher(models.Model):
    name = models.CharField(max_length=50)
    address = models.CharField(max_length=100)
    website = models.URLField()

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

■ 2. Admin 사이트에 보이도록 테이블을 admin.py 파일에도 등록해준다.

[ch5Lab/books/admin.py]

```
from django.contrib import admin
from books.models import Book, Author, Publisher

admin.site.register(Book)
admin.site.register(Author)
admin.site.register(Publisher)
```

■ 3. 방금 정의한 테이블을 데이터베이스에 반영한다.

```
# 데이터베이스에 변경이 필요한 사항을 추출함
D:\dev\workspace\django1\ch5Lab>python manage.py makemigrations
Migrations for 'books':
  books\migrations\0001_initial.py
    - Create model Author
    - Create model Publisher
    - Create model Book

# 데이터베이스에 변경사항을 반영함
D:\dev\workspace\django1\ch5Lab>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, books, contenttypes, polls, sessions
Running migrations:
  Applying books.0001_initial... OK

# 현재까지 작업을 개발용 웹서버에 확인함
D:\dev\workspace\django1\ch5Lab>python manage.py runserver 0.0.0.0:8000
```

5.1.4 애플리케이션 - URLconf 코딩하기

■ 기존의 mysite/urls.py 파일에 books 앱의 URL 설정을 불러오는 내용을 추가한다.

[ch5Lab/mysite/urls.py]

```
01  from django.contrib import admin
02  from django.urls import path, include
03  from mysite import views    # 추가
04
05
06  urlpatterns = [
07      path('admin/', admin.site.urls),
08      path('', views.HomeView.as_view(), name='home'),    # 추가
09      path('polls/', include('polls.urls')),
10      path('books/', include('books.urls')),
11  ]
```

■ books/urls.py 파일에는 /book/과 /author/ 및 /publisher/ URL도 같이 정의해준다.
- 08: 함수형 뷰가 아니라 클래스형 뷰로 정의하기 위해 각 URL에 따른 뷰 클래스 및 as_view() 메소드를 지정한다.

[ch5Lab/books/urls.py]

```
01  from django.urls import path
02  from . import views
03
04
05  app_name = 'books'
06  urlpatterns = [
07      # /books/
08      path('', views.BooksModelView.as_view(), name='index'),
09
10      # /books/book/
11      path('book/', views.BookList.as_view(), name='book_list'),
12
13      # /books/author/
14      path('author/', views.AuthorList.as_view(), name='author_list'),
15
16      # /books/publisher/
17      path('publisher/', views.PublisherList.as_view(), name='publisher_list'),
18
19      # /books/book/99/
20      path('book/<int:pk>/', views.BookDetail.as_view(), name='book_detail'),
21
22      # /books/author/99/
23      path('author/<int:pk>/', views.AuthorDetail.as_view(), name='author_detail'),
24
25      # /books/publisher/99/
26      path('publisher/<int:pk>/', views.PublisherDetail.as_view(), name='publisher_detail'),
27  ]
```

5.1.5 애플리케이션 - 클래스형 View 코딩하기

■ books/views.py 파일에 아래 내용을 입력한다.
- 01~03: 클래스형 제네릭 뷰를 사용하기 위해 TemplateView, ListView, DetailView 클래스를 임포트한다.
- 04: 테이블 조회를 위하여 모델 클래스들을 임포트한다.
- 08: BooksModelView는 books 애플리케이션의 첫 화면을 보여주기 위한 뷰이다. 특별한 로직이 없고 템플릿 파일만을 렌더링하는 경우에는 이처럼 TemplateView 제네릭 뷰를 상속받아 사용하면 간단하다. TemplateView를 사용하는 경우에는 필수적으로

template_name 클래스 벼수를 오버라이딩해서 지정해줘야 한다.

- 12: get_context_data() 메소드를 정의할 때는 반드시 첫 줄에 super() 메소드를 호출해야 한다.
- 13: books 애플리케이션의 첫 화면에 테이블 리스트를 보여주기 위해 컨텍스트 변수 model_list에 담아서 템플릿 시스템에 넘겨주고 있다.

[ch5Lab/books/views.py] 클래스형 뷰 코딩

```
01  from django.views.generic.base import TemplateView
02  from django.views.generic import ListView
03  from django.views.generic import DetailView
04  from books.models import Book, Author, Publisher
05
06
07  #--- TemplateView
08  class BooksModelView(TemplateView):
09      template_name = 'books/index.html'
10
11      def get_context_data(self, **kwargs):
12          context = super().get_context_data(**kwargs)
13          context['model_list'] = ['Book', 'Author', 'Publisher']
14          return context
15
16
17  #--- ListView
18  class BookList(ListView):
19      model = Book
20
21
22  class AuthorList(ListView):
23      model = Author
24
25
26  class PublisherList(ListView):
27      model = Publisher
28
29
30  #--- DetailView
31  class BookDetail(DetailView):
32      model = Book
33
34
35  class AuthorDetail(DetailView):
36      model = Author
37
38
39  class PublisherDetail(DetailView):
40      model = Publisher
```

5.1.6 애플리케이션 - Template 코딩하기

■ books 애플리케이션 - URL/뷰/템플릿 매핑

URL 패턴	뷰 클래스명	템플릿 파일명	템플릿 설명
/books/	BooksModelView	index.html	books 애플리케이션 첫 화면
/books/book/	BookList	book_list.html	책의 리스트를 보여줌
/books/author/	AuthorList	author_list.html	저자의 리스트를 보여줌
/books/publisher/	PublisherList	publisher_list.html	출판사의 리스트를 보여줌
/booksbook/3/	BookDetail	book_detail.html	특정 책의 상세 정보를 보여줌

/books/author/3/	AuthorDetail	author_detail.html	특정 저자의 상세 정보를 보여줌
/books/publisher/3/	PublisherDetail	publisher_detail.html	특정 출판사의 상세 정보를 보여줌

■ index.html 템플릿

- base_books.html 템플릿을 상속받아서 content 블록만을 재정의하였고, 나머지 블록은 부모 템플릿 내용을 그대로 사용하고 있다.
- 07: add 및 lower 템플릿 필터를 사용하여 모델명을 소문자로 변환하고, 필요한 문자열을 붙여준다. 예) 모델명이 Author라면 urlvar는 books:author_list가 될 것이다.

[ch5Lab/books/templates/books/index.html]

```
01  {% extends "base_books.html" %}
02
03  {% block content %}
04      <h2>Books Management System</h2>
05      <ul>
06          {% for modelname in model_list %}
07  {% with "books:"|add:modelname|lower|add:"_list" as urlvar %}
08              <li><a href="{% url urlvar %}">{{ modelname }}</a></li>
09  {% endwith %}
10          {% endfor %}
11      </ul>
12  {% endblock content %}
```

■ book_list.html 템플릿

- 06~08: 뷰로부터 object_list 컨텍스트 변수를 전달받아서 object_list에 들어 있는 객체들을 순회하면서 하나씩 보여주고 있다.

[ch5Lab/books/templates/books/book_list.html]

```
01  {% extends "base_books.html" %}
02
03  {% block content %}
04      <h2>Book List</h2>
05      <ul>
06          {% for book in object_list %}
07              <li><a href="{% url 'books:book_detail' book.id %}">{{ book.title }}</a></li>
08          {% endfor %}
09      </ul>
10  {% endblock content %}
```

■ author_list.html 템플릿

[ch5Lab/books/templates/books/author_list.html]

```
01  {% extends "base_books.html" %}
02
03  {% block content %}
04      <h2>Author List</h2>
05      <ul>
06          {% for author in object_list %}
07              <li><a href="{% url 'books:author_detail' author.id %}">{{ author.name }}</a></li>
08          {% endfor %}
09      </ul>
10  {% endblock content %}
```

5.1.7 애플리케이션 - Template 상속 기능 추가

■ base.html 템플릿
- 05: {% load static %} 템플릿 태그는 static이라는 사용자 정의 태그를 로딩해 준다.
- 06: {% static %} 사용자 정의 태그를 통해 admin/css/base.css 스타일시트 파일을 찾게 된다.

[ch5Lab/templates/base.html]

```
01  <!DOCTYPE html>
02  <html lang="en">
03  <head>
04      {% load static %}
05      <link rel="stylesheet" href="{% static 'admin/css/base.css' %}" />
06      <title>{% block title %}My Amazing Site{% endblock %}</title>
07  </head>
08
09  <body>
10      <div id="sidebar">
11          {% block sidebar %}
12          <ul>
13              <li><a href="/">Project_Home</a></li>
14              <li><a href="/admin/">Admin</a></li>
15          </ul>
16          {% endblock %}
17          <br>
18      </div>
19
20      <div id="content">
21          {% block content %}{% endblock %}
22      </div>
23  </body>
24  </html>
```

■ base_books.html 템플릿
- base.html 템플릿을 상속받아서 이 중 title 블록과 sidebar 블록을 재정의하고 있다.
- 06: 이 템플릿 변수의 의미는 부모 base.html 템플릿에서 정의한 내용을 하위 base_books.html 템플릿에서 재사용한다는 의미이다.

[ch5Lab/templates/base_books.html]

```
01  {% extends "base.html" %}
02
03  {% block title %}Books Application Site{% endblock %}
04
05  {% block sidebar %}
06  {{ block.super }}
07  <ul>
08      <li><a href="/books/">Books_Home</a></li>
09  </ul>
10  {% endblock %}
```

5.1.8 지금까지 작업 확인하기

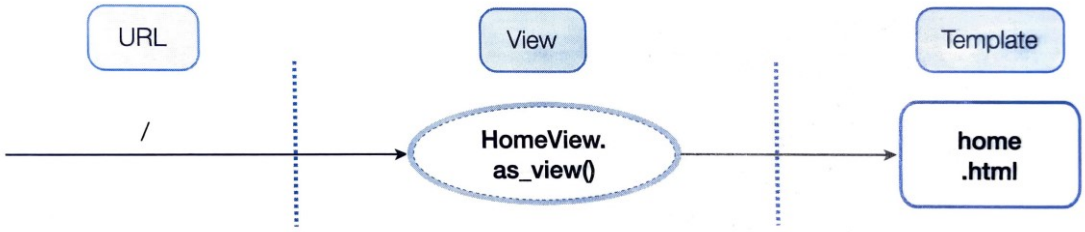
■ runserver를 실행하고 브라우저로 Admin 사이트에 접속한다.

5.2 프로젝트 첫 페이지 만들기

■ 프로젝트 첫 페이지인 루트(/) URL에 대한 처리 로직을 코딩한다.

5.2.1 프로젝트 첫 페이지 설계

■ 프로젝트 첫 페이지 - 뷰 흐름 설계



5.2.2 URLconf 코딩하기

■ 프로젝트에 대한 URL이므로 mysite/urls.py 파일에 루트(/) URL 및 임포트 문장, 두 줄만 추가하면 된다.

[ch5Lab/mysite/urls.py]

```
01  from django.contrib import admin
02  from django.urls import path, include
03  from mysite import views    # 추가
04
05
06  urlpatterns = [
07      path('admin/', admin.site.urls),
08      path('', views.HomeView.as_view(), name='home'),    # 추가
09      path('polls/', include('polls.urls')),
10      path('books/', include('books.urls')),
11  ]
```

5.2.3 View 코딩하기

■ 뷰 이름은 앞에서 URLconf를 코딩하면서 HomeView라고 정의하였다. 파일의 위치는 애플리케이션이 아니라 프로젝트와 관련된 뷰이므로, mysite/views.py 파일에 코딩하는 것이 적절하다.

-
- 09: get_context_data() 메소드를 정의할 때는 반드시 첫 줄에 super() 메소드를 호출해야 한다.
- 11: mysite 프로젝트 하위에 있는 애플리케이션들의 리스트를 보여주기 위해 컨텍스트 변수 app_list에 담아서 템플릿 시스템에 넘겨준다.

[ch5Lab/mysite/views.py]

```
01  from django.views.generic.base import TemplateView
02
03
04  #--- TemplateView
05  class HomeView(TemplateView):
06
07      template_name = 'home.html'
08
09      def get_context_data(self, **kwargs):
10          context = super().get_context_data(**kwargs)
11          context['app_list'] = ['polls', 'books']
12          return context
```

5.2.4 Template 코딩하기

- home.html 템플릿은 개별 애플리케이션 템플릿이 아니라 프로젝트 템플릿이므로, 상속에 사용하는 부모 템플릿의 위치와 동일하게 'ch05Lab/templates/' 디렉토리에 생성한다.
  - 06: 뷰로부터 app_list 컨텍스트 변수를 전달받아서 app_list에 들어있는 애플리케이션명(appname)들을 하나씩 순회하면서 화면에 보여준다.
  - 07: add 템플릿 필터를 사용하여 애플리케이션명에 필요한 문자열을 붙여주고 있다. 예) 애플리케이션명이 'books'라면 urlvar는 'books:index'가 된다.

[ch5Lab/templates/home.html]

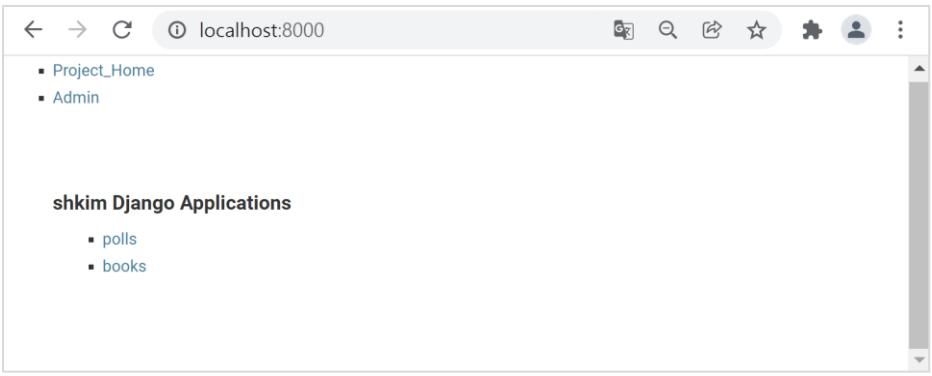
```
01  {% extends "base.html" %}
02
03  {% block content %}
04      <h2>shkim Django Applications</h2>
05      <ul>
06          {% for appname in app_list %}
07  {% with appname|add:":"|add:"index" as urlvar %}
08              <li><a href="{% url urlvar %}">{{ appname }}</a></li>
09  {% endwith %}
10          {% endfor %}
11      </ul>
12  {% endblock content %}
```

5.2.5 지금까지 작업 확인하기

- runserver를 실행한 후에 브라우저를 통해 루트(/) URL로 접속한다.

```
D:\dev\workspace\django1\ch5Lab>python manage.py runserver 0.0.0.0:8000
```

- 프로젝트 첫 페이지 - Project_Home 성공 화면



5.2.6 apps.py 활용 맛보기

■ 프로젝트의 전반적인 항목들을 설정하는 곳은 settings.py 파일고, 각 앱마다 필요한 항목을 설정할 수 있는 곳은 apps.py 파일이다.

- 07: books 앱의 설정 클래스인 BooksConfig의 속성 중 하나인 verbose_name(앱 이름에 대한 별칭)을 정의한다.

[ch5Lab/books/apps.py]

```
01  from django.apps import AppConfig
02
03
04  class BooksConfig(AppConfig):
05      # default_auto_field = 'django.db.models.BigAutoField'
06      name = 'books'
07      verbose_name = 'Book-Author-Publisher App'  # 추가
```

■ 이외에도 path. label 등의 속성이 있는데 이들을 활용하는 코드는 mysite/views.py 파일에 작성한다.

- 15: apps 객체의 get_app_configs() 메소드를 호출하면, settings.py 파일의 INSTALLED_APPS에 등록된 각 앱의 설정 클래스들을 담은 리스트를 반환한다.
- 16: app은 각 앱의 설정 클래스를 의미하므로, app.path는 각 설정 클래스의 path 속성으로 애플리케이션 디렉토리의 물리적 경로를 뜻한다. 예를 들어 books 앱의 물리적 경로는 D:\dev\workspace\django1\ch5Lab\books 이다. 물리적 경로에 site-packages 문자열이 들어 있으면 외부 라이브러리 앱을 의미하므로 if 문장에서 이런 앱을 제외한다.
- 17: 설정 클래스의 label 속성값을 키(key)로 verbose_name 속성값을 값(value)으로 해서 dictVerbose 딕셔너리에 담는다. books 앱의 경우 label=books, verbose_name=Book-Author-Publisher App
- 18: for 문장이 완료된 후에 verbose_dict 컨텍스트 변수에 dictVerbose 딕셔너리를 대입한다.

[ch5Lab/mysite/views.py] BooksConfig 속성 활용 코드

```
01  from django import apps
02  from django.views.generic.base import TemplateView
03  from django.apps import apps  # 추가
04
05
06  #--- TemplateView
07  class HomeView(TemplateView):
08
09      template_name = 'home.html'
```

```
10
11      def get_context_data(self, **kwargs):
12          context = super().get_context_data(**kwargs)
13          # context['app_list'] = ['polls', 'books']
14          dictVerbose = {}
15          for app in apps.get_app_configs():
16              if 'site-packages' not in app.path:
17                  dictVerbose[app.label] = app.verbose_name
18          context['verbose_dict'] = dictVerbose
19          return context
```

■ templates/home.html을 아래와 같이 수정한다.

[ch5Lab/templates/home.html]

```
01  {% extends "base.html" %}
02
03  {% block content %}
04      <h2>shkim Django Applications</h2>
05      <ul>
06          {% for key, value in verbose_dict.items %}
07              <li><a href="{% url key|add:':index' %}">{{ value }}</a></li>
08          {% endfor %}
09      </ul>
10  {% endblock content %}
```

■ mysite/settings.py 수정
- 04: INSTALLED_APPS 항목에 앱을 등록 시 모듈명이 아니라 설정 클래스로 등록한다.

[ch5Lab/mysite/settings.py]

```
01  INSTALLED_APPS = [
02      ...(생략)...
03      # 'books.apps.BooksConfig',                 # 추가
04      'books',    # 추가
05  ]
```

5.3 polls 애플리케이션 - 클래스형 뷰로 변경하기

5.3.1 URLconf 코딩하기
5.3.2 View 코딩하기
5.3.3 Template 코딩하기
5.3.4 로그 추가하기
5.3.5 지금까지 작업 확인하기