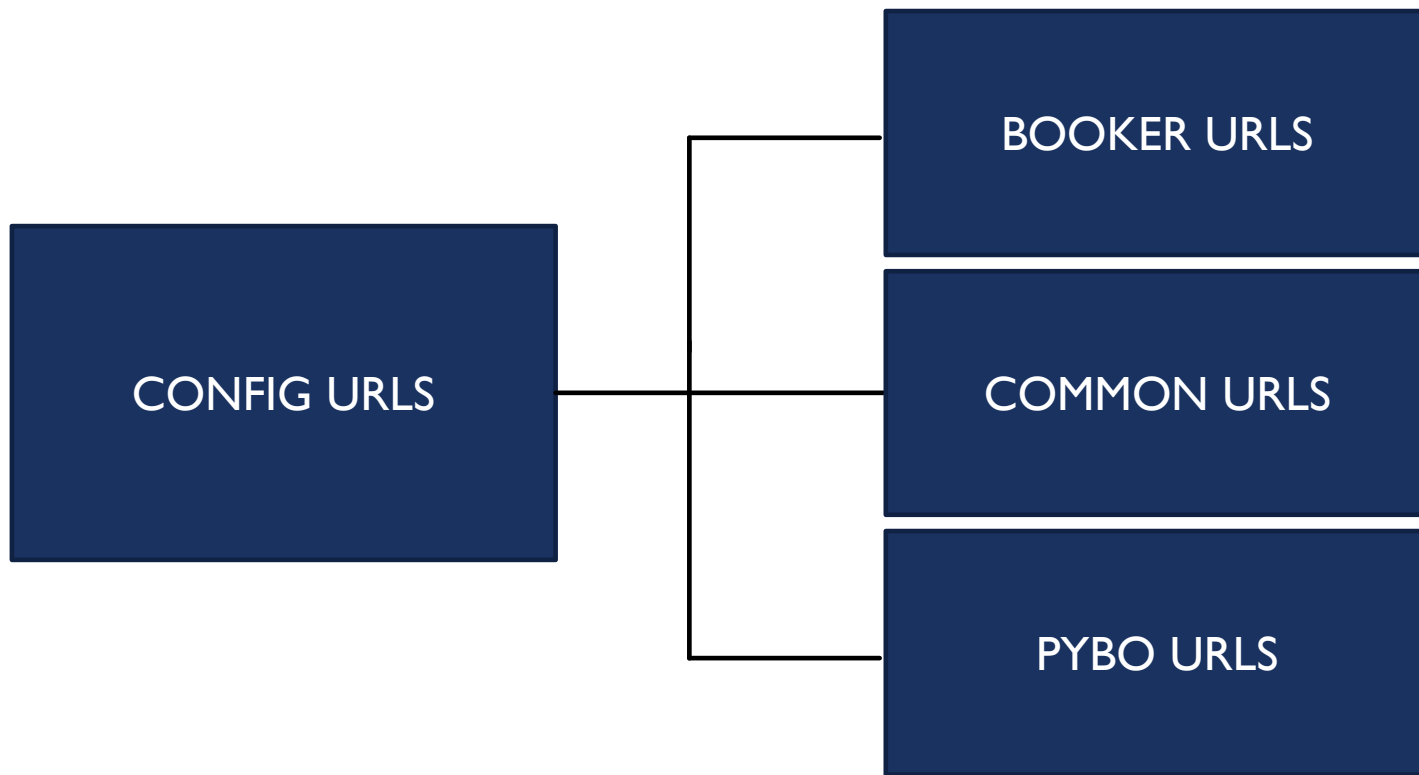




PYBO-BOOKER WEB APP

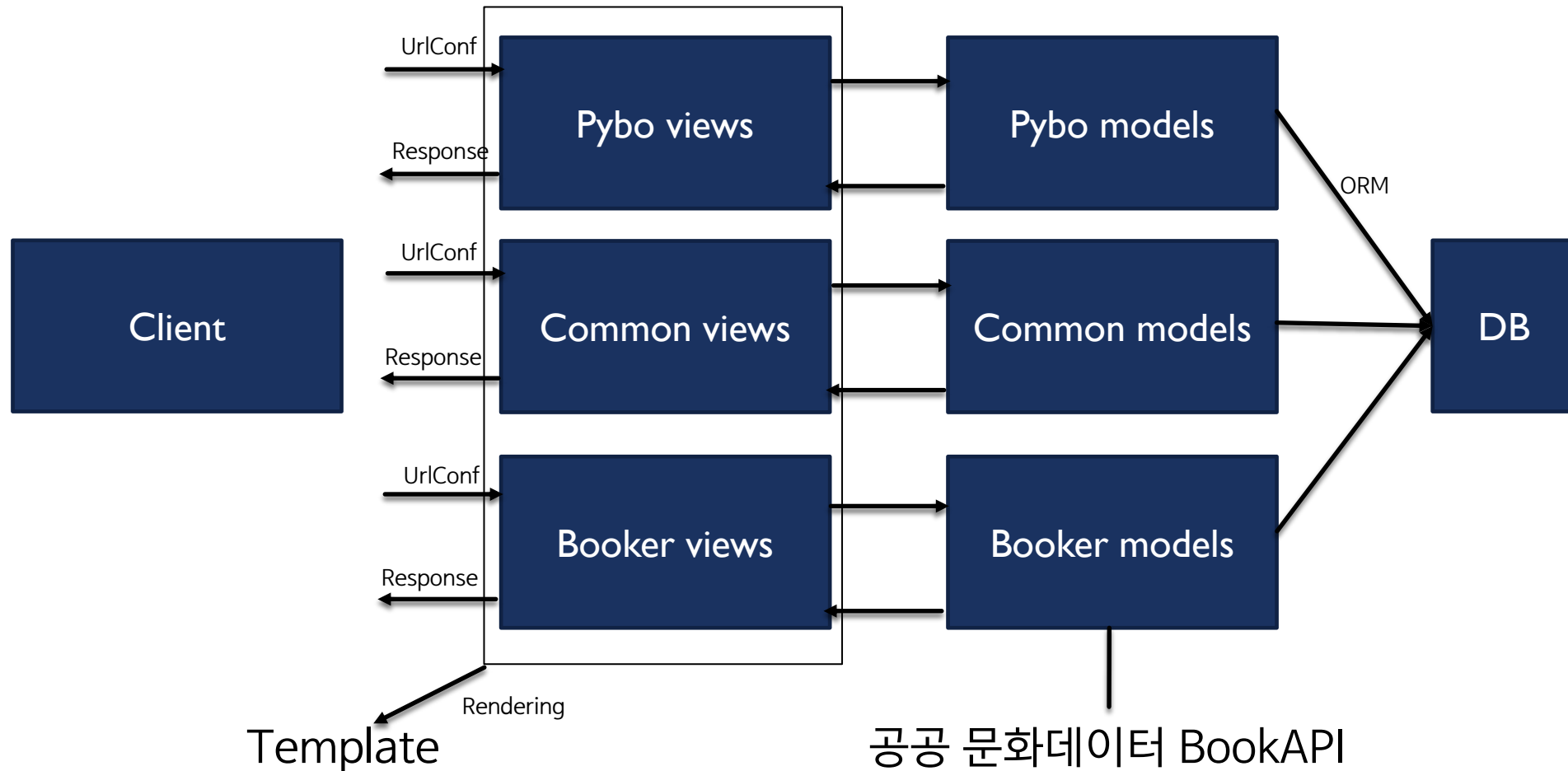
DEVELOPED BY 박태욱



메인 프로젝트는 config로 하여,
프로젝트 설정 및 데이터베이스 설정
어드민 설정 등등을 담당합니다.
메인 프로젝트는 설정을 담당하기
때문에 이름을 config 로
설정하였습니다.

URL 구조

애플리케이션 구조



프로젝트 설정 구상 - CONFIG

- 메인 페이지 '/' 주소를 렌더링한다
- JSON 파일을 사용하여 숨겨야 하는 정보들을 os 모듈을 통해 불러온다
- 이메일을 사용하기 위해 이메일 설정을 정의한다
- 개발시에는 dbsqlite3를 사용, 배포시에는 aws rds등 데이터베이스와 연동하기 위해 mysql설정만 정의, 로컬 mysql과 연결하여 migrate이 잘되는지 확인만 한다
- 구글 로그인을 사용하기 위해 구글 로그인과 관련된 Django all-auth 라이브러리를 설치하고 설정을 정의한다.
- urlConf 로 각각 애플리케이션들과 연결시킨다
- common.views의 404페이지 렌더링과 연결하여 404 페이지를 처리한다

애플리케이션 구상 - COMMON

- Django auth를 통해 회원가입이 가능하도록 한다
- Django all-auth 라이브러리를 통해 로그인 가능하도록 한다/ 구글로그인 구현
- 이메일 인증을 통해 비밀번호 초기화를 구현한다
- 404페이지를 처리한다

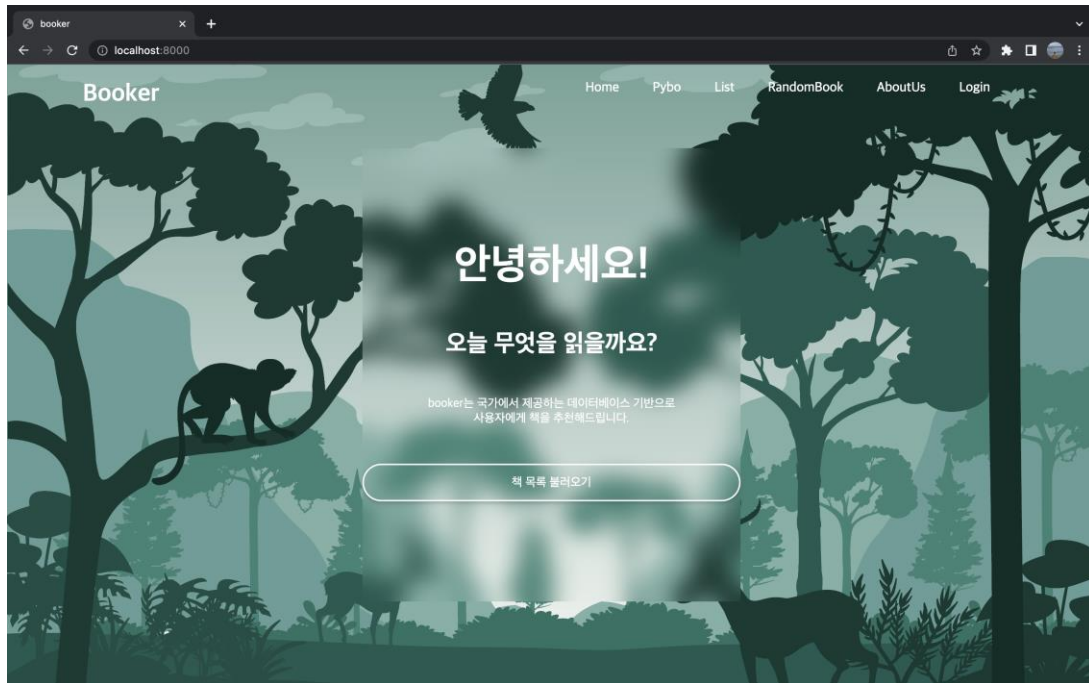
애플리케이션 구상 - BOOKER

- 문화 공공데이터 광장에서 책 API 를 통해 책 데이터를 받아온다 requests 라이브러리 사용
- 책 데이터를 리스트로 나열하고 원하는 방식에 따라 정렬이 가능하도록 하고 페이지를 구현한다
- 책 상세정보를 구현하고 상세정보 창에서 책에 댓글을 달거나, 좋아요를 누를 수 있게 한다
- 책 상세정보 창에서 댓글을 삭제하거나 수정하는 폼으로 이동이 가능하도록 한다
- 랜덤 책 뽑기를 통해 책을 랜덤으로 추천 받을 수 있도록 한다
- 내가 댓글을 단 책과 좋아요를 누른 책을 확인할 수 있도록 한다
- 개발자 정보/프로필을 남긴다

애플리케이션 구상 - PYBO

- 로그인한 사용자에게 한해 새로운 자기소개 게시글을 받을 수 있다
- 게시글들을 리스트로 표현이 가능하다
- 내가 작성한 게시글 이라면 수정과 삭제가 가능하도록 한다
- 내가 작성한 댓글은 수정과 삭제가 가능하도록 한다
- 게시글 상세 페이지에서 게시글 상세 정보를 표시하고 댓글 기능을 추가한다
- 내가 작성한 게시글 확인이 가능하도록 한다

BOOKER/MAIN HOME



- 메인 페이지로 책 목록 불러오기를 클릭시 list로 이동한다
- 로고 클릭시 다시 메인/booker 홈으로 이동한다
- Home을 클릭시 다시 홈으로 이동한다
- Pybo클릭시 파이보 앱으로 이동한다
- RandomBook클릭시 랜덤 책추천 으로 이동한다
- AboutUs 클릭시 개발자 상세 정보로 이동한다
- 로그인 클릭시 로그인 페이지로 이동한다



CONFIG

SETTINGS.PY - CONFIG

LOGIN_REDIRECT_URL='/'

LOGOUT_REDIRECT_URL='/'

EMAIL_HOST = 'smtp.gmail.com'

EMAIL_PORT = '587'

EMAIL_HOST_USER = 'xodnrxo17@gmail.com'

EMAIL_HOST_PASSWORD =
get_secret('EMAIL_PASSWORD')

EMAIL_USE_TLS = True

DEFAULT_FROM_EMAIL = EMAIL_HOST_USER

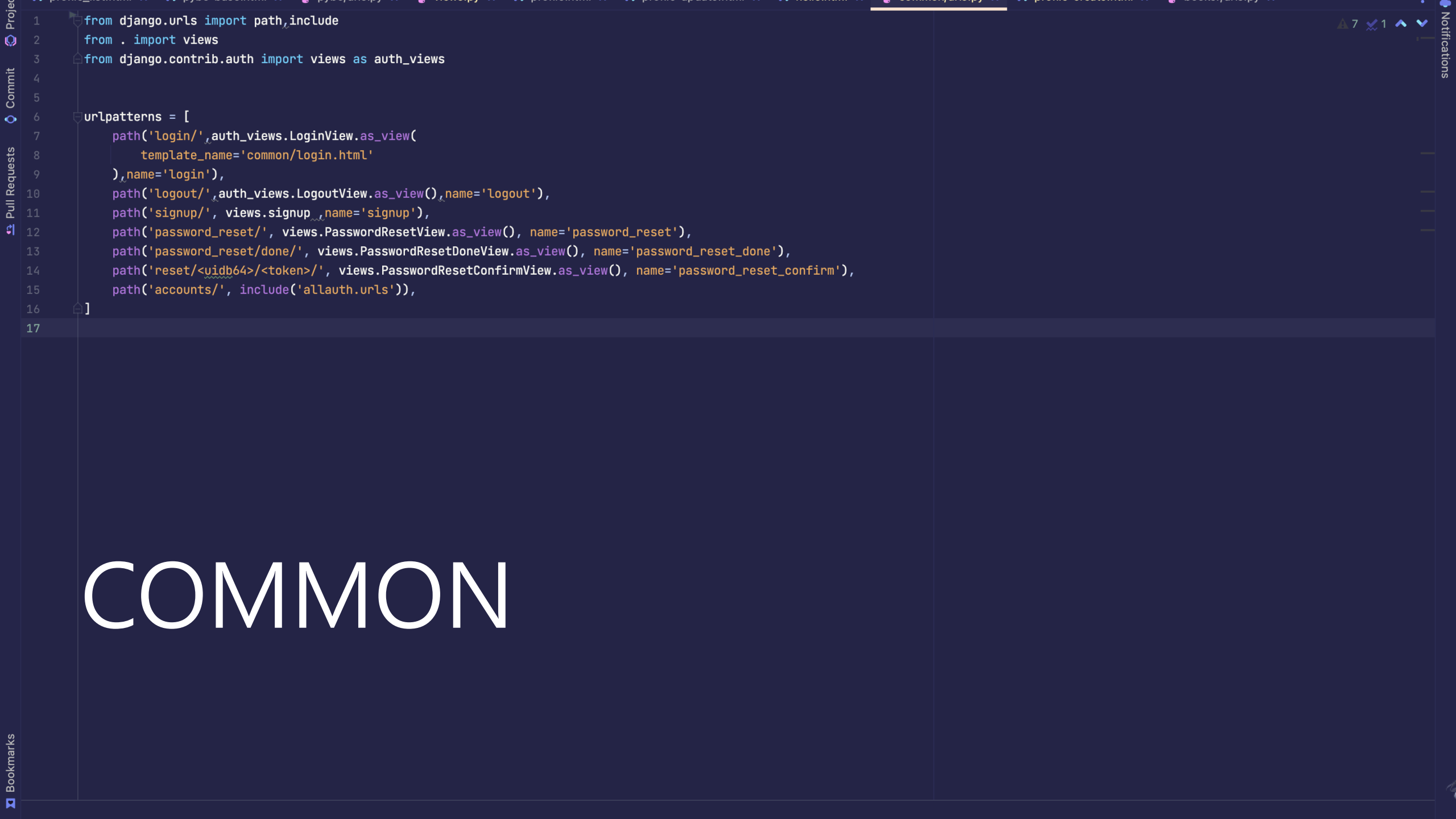
SITE_ID=1

- 로그인 및 로그아웃 시 '/' 로 리디렉트 되도록 설정
- 이메일 호스트를 gmail로 하였다
- 이메일 포트를 587번 포트로 지정하였다
- 이메일 호스트 유저는 개발자 이메일을 정의한다
- 이메일 패스워드는 개인정보이므로 가린다
- 이메일 TLS 보안은 참으로 설정한다
- 사이트 id 는 1로 지정하여 all auth 설정을 한다

SETTINGS.PY - CONFIG

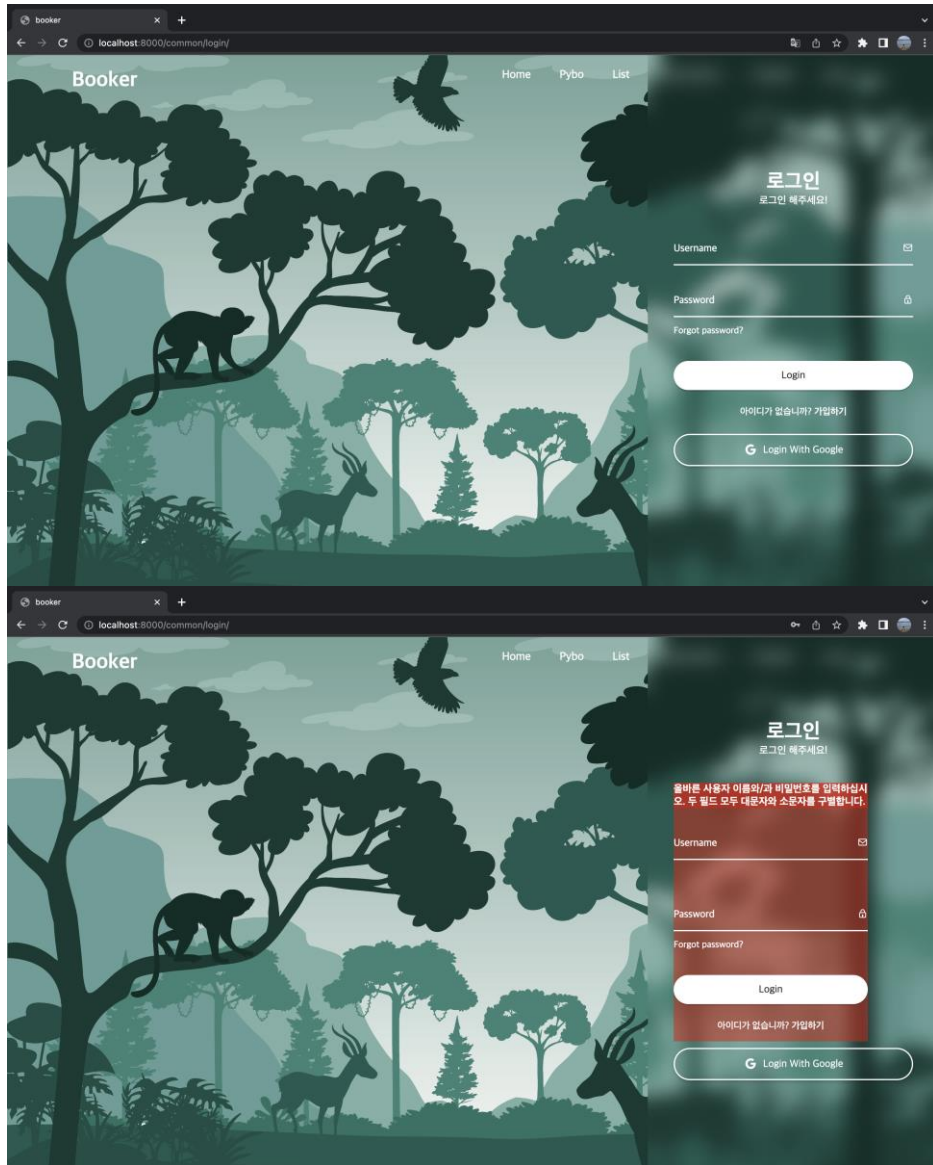
```
INSTALLED_APPS = [  
    'common.apps.CommonConfig',  
    'pybo.apps.PyboConfig',  
    'booker.apps.BookerConfig',  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'django.contrib.sites',  
  
    # allauth  
    'allauth',  
    'allauth.account',  
    'allauth.socialaccount',  
    'allauth.socialaccount.providers.openid',  
  
    #provider  
    'allauth.socialaccount.providers.google'  
]
```

- Common 앱과 Booker 그리고 Pybo 앱을 지정한다
- Django.contrib.site를 지정하여 8000번 포트를 사이트로 지정
- allauth 즉 소셜 로그인을 사용하기 위해 인스톨드 앱에 필요한 항목들을 정의한다
- 구글 로그인을 사용하기 위해 제공자에 구글을 포함한다



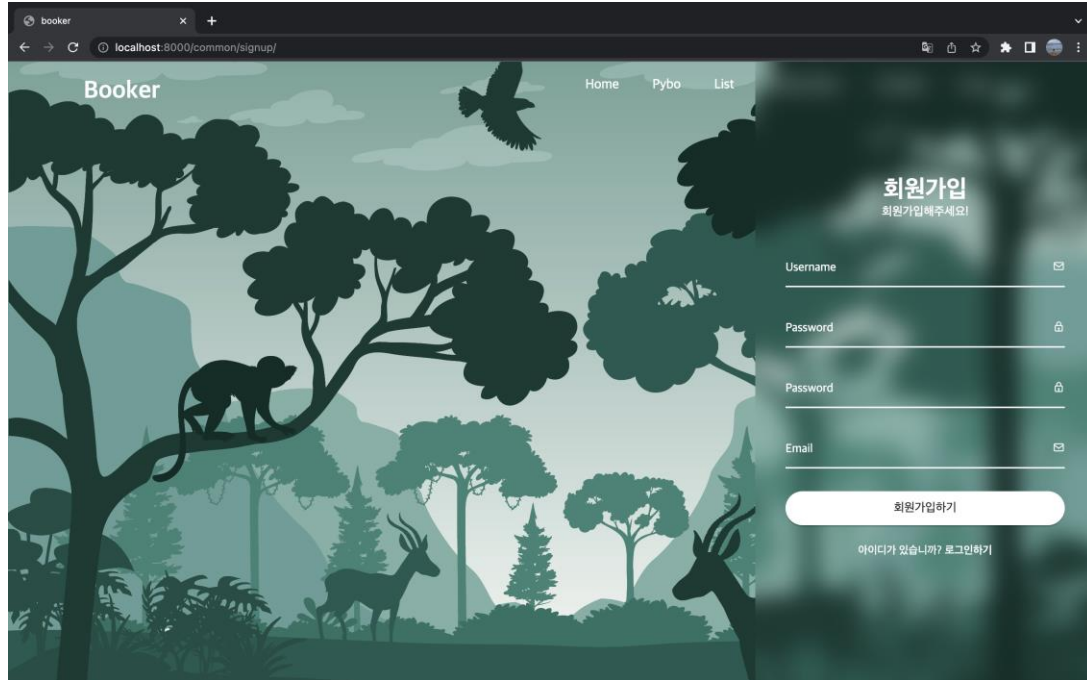
COMMON

LOGIN - COMMON



- 로그인 페이지로 폼을 통해 유저이름과 비밀번호를 받는다
- 유저이름과 비밀번호 입력란에 문제가 있을 시 아래 사진처럼 오류를 표시한다
- 로그인 버튼을 클릭하여 입력한 정보를 보낸다
- Forgot 패스워드 클릭 시 비밀번호 초기화 페이지로 이동한다
- 가입하기 버튼 클릭 시 가입 화면으로 이동한다
- Login with google 클릭 시 구글 로그인 페이지로 이동한다

REGISTER - COMMON



- 회원가입 페이지로 유저이름과 비밀번호 이메일을 받는다
- 폼에 이상이 있을 시 오류를 표시한다
- 회원가입하기 버튼을 클릭하여 회원가입이 가능하다
- 로그인하기 버튼을 눌러서 로그인 창으로 이동이 가능하다

PASSWORD RESET - COMMON



●비밀번호 초기화 페이지로 사용자 이름과 매칭되는 이메일을 받는다

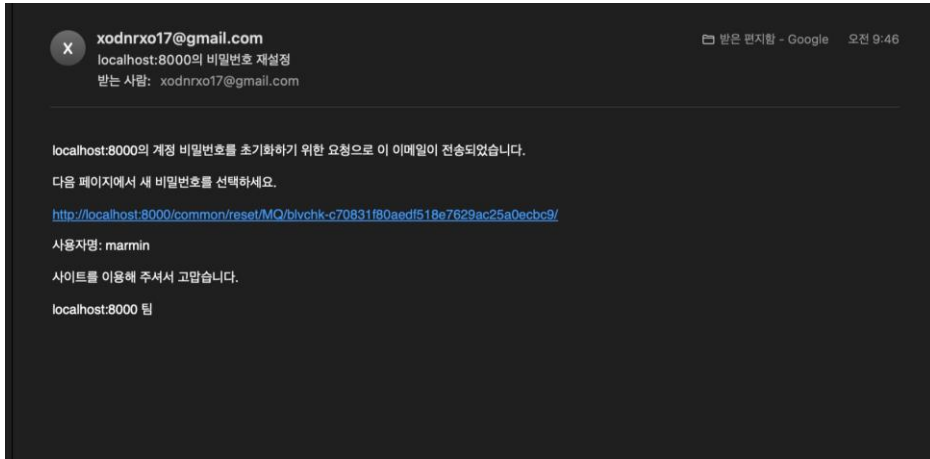
●폼에 이상이 있을 시 오류를 표시한다

●이메일 전송 버튼을 눌러 이메일을 전송한다.

●아래 이메일 전송 완료 창으로 리디렉션 된다

●전송된 이메일의 url로 이동시 새로운 비밀번호 입력이 가능하다

PASSWORD RESET - COMMON



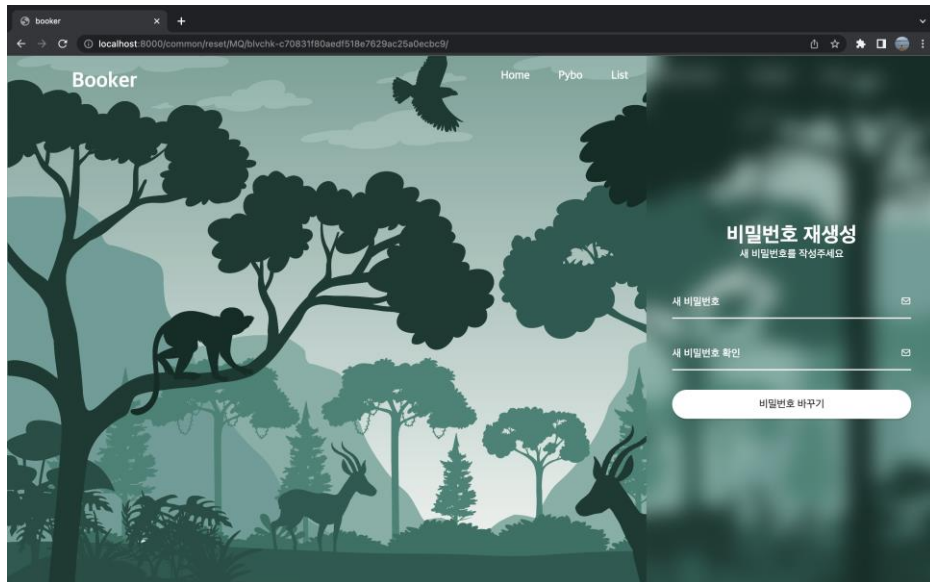
●이메일로 전송된 url로 이동하여 비밀번호 재생성이 가능하다

●이메일로 전송된 url이 손상되면 초기화가 불가능하다/
토큰값과 uid값이 일치해야 초기화가 가능하다

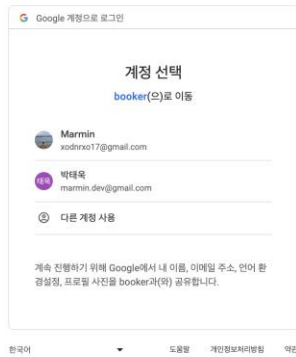
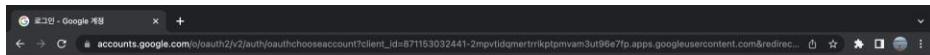
●비밀번호 재생성으로 이동하여 비밀번호를 재생성한다

●비밀번호 바꾸기 버튼 클릭시 로그인된 상태로 홈으로 이동한다

●폼에 이상이 있을 시 오류를 표시한다



OAUTH2.0- COMMON

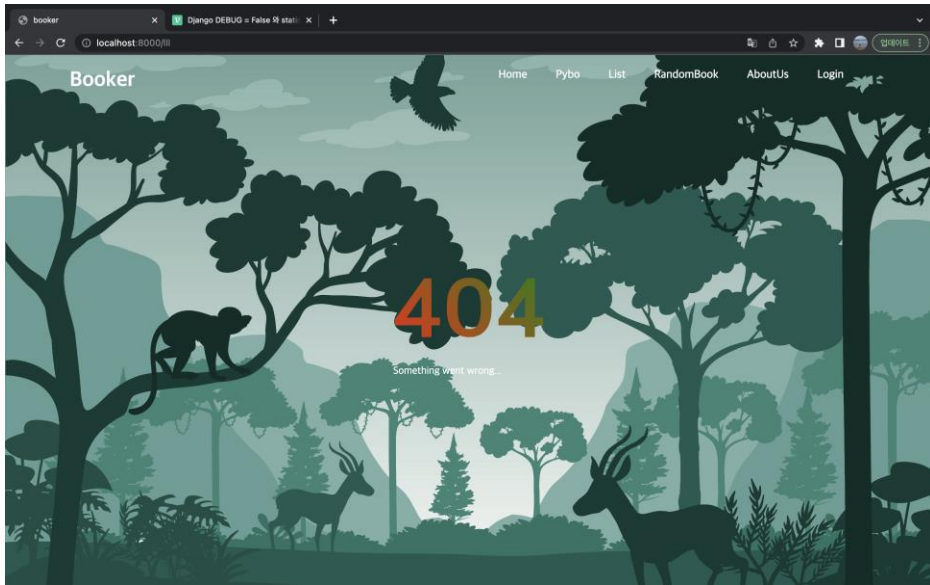


●구글 로그인이 가능하다

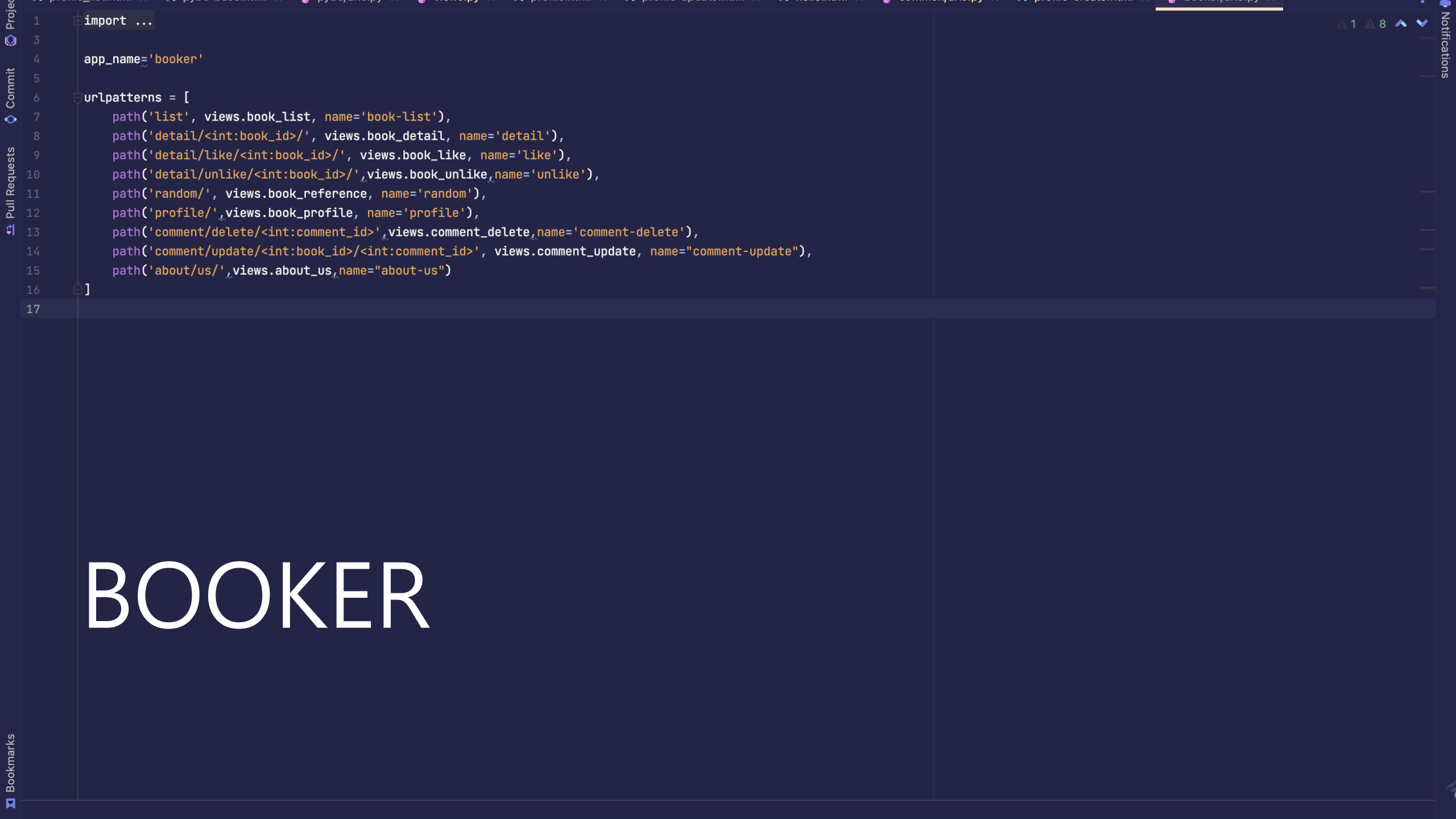
●처음 접속하는 사용자일 경우 사용자이름을 받는다

●두번째 접속부터는 바로 로그인이 가능하다

404- COMMON



- 잘못된 경로로 이동하였을 시 나타날 페이지이다



```
1 import ...
3
4 app_name='booker'
5
6 urlpatterns = [
7     path('list', views.book_list, name='book-list'),
8     path('detail/<int:book_id>/', views.book_detail, name='detail'),
9     path('detail/like/<int:book_id>/', views.book_like, name='like'),
10    path('detail/unlike/<int:book_id>/', views.book_unlike, name='unlike'),
11    path('random/', views.book_reference, name='random'),
12    path('profile/', views.book_profile, name='profile'),
13    path('comment/delete/<int:comment_id>', views.comment_delete, name='comment-delete'),
14    path('comment/update/<int:book_id>/<int:comment_id>', views.comment_update, name="comment-update"),
15    path('about/us/', views.about_us, name="about-us")
16 ]
17
```

BOOKER

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

```
2 from django.contrib.auth.models import User
```

BOOKER MODELS.PY- BOOKER

```
3 class Book(models.Model):
```

```
4     title = models.CharField('제목', max_length=50, unique=True, null=False)
```

```
5     extent=models.CharField('출판사', max_length=50)
```

```
6     description=models.TextField('소개', max_length=500)
```

```
7     charge = models.CharField('가격', max_length=10)
```

```
8     cover = models.CharField('책표지', max_length=100)
```

```
9     rights = models.CharField('저자', max_length=50)
```

```
10    issued_at = models.CharField('출판일자', max_length=20)
```

```
11    like = models.ManyToManyField(User, blank=True)
```

```
12
```

```
13
```

```
14    def __str__(self):
```

```
15        return self.title
```

```
16
```

```
17
```

```
18
```

```
19 class Comment(models.Model):
```

```
20     author = models.ForeignKey(User, on_delete=models.CASCADE)
```

```
21     content = models.CharField('내용', max_length=255)
```

```
22     post = models.ForeignKey(Book, on_delete=models.CASCADE)
```

```
23     create_at = models.DateTimeField()
```

```
24
```

```
25
```

```
26    def __str__(self):
```

```
27        return self.content
```

```
28
```

```
29
```

```
class Meta:
```

```
    ordering = ['-create_at']
```

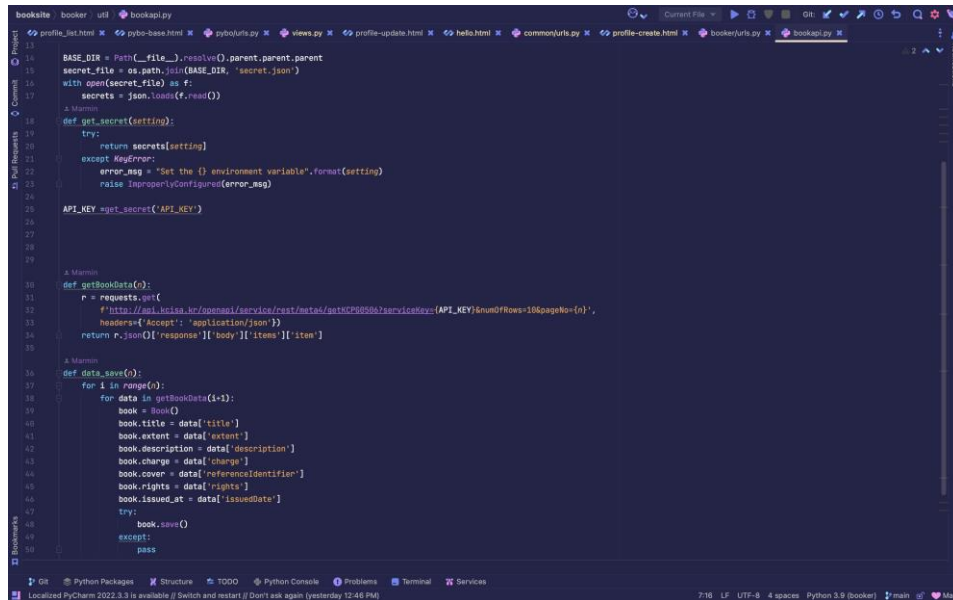
● Book 모델 중 like 를 뺀 나머지 필드는 request라이브러리를 이용한 유틸프로그램으로 데이터가 자동으로 채워진다

● like는 many to many 필드를 사용하여 유저와 다대다 매핑을 통해 추천을 받을 수 있도록 하였다

● comment는 사용자가 입력하는 정보에 따라 입력되는 모델이다

● comment 필드 중 content 필드는 사용자의 입력을 받아 채워지고 나머지는 로그인 정보 및 책, 그리고 생성 일자가 자동으로 입력된다

REQUEST UTIL - BOOKER



```
13
14 BASE_DIR = Path(__file__).parent.parent.parent
15 secret_file = os.path.join(BASE_DIR, 'secret.json')
16 with open(secret_file) as f:
17     secrets = json.loads(f.read())
18
19 A Message
20 def get_secret(setting):
21     try:
22         return secrets[setting]
23     except KeyError:
24         error_msg = "Set the {} environment variable".format(setting)
25         raise ImproperlyConfigured(error_msg)
26
27 API_KEY = get_secret('API_KEY')
28
29 A Message
30 def getBookData(n):
31     r = requests.get(
32         f'http://api.kobis.co.kr/openapi/service/rest/metadata/getKCPROB000?apiKey={API_KEY}&numOfRows=10&pageNo={n}',
33         headers={'Accept': 'application/json'})
34     return r.json()['response']['body']['items']['item']
35
36 A Message
37 def data_save(n):
38     for i in range(n):
39         for data in getBookData(i+1):
40             book = Book()
41             book.title = data['title']
42             book.extent = data['extent']
43             book.description = data['description']
44             book.charge = data['charge']
45             book.cover = data['referenceIdentifier']
46             book.rights = data['rights']
47             book.issued_at = data['issuedDate']
48             try:
49                 book.save()
50             except:
51                 pass
```

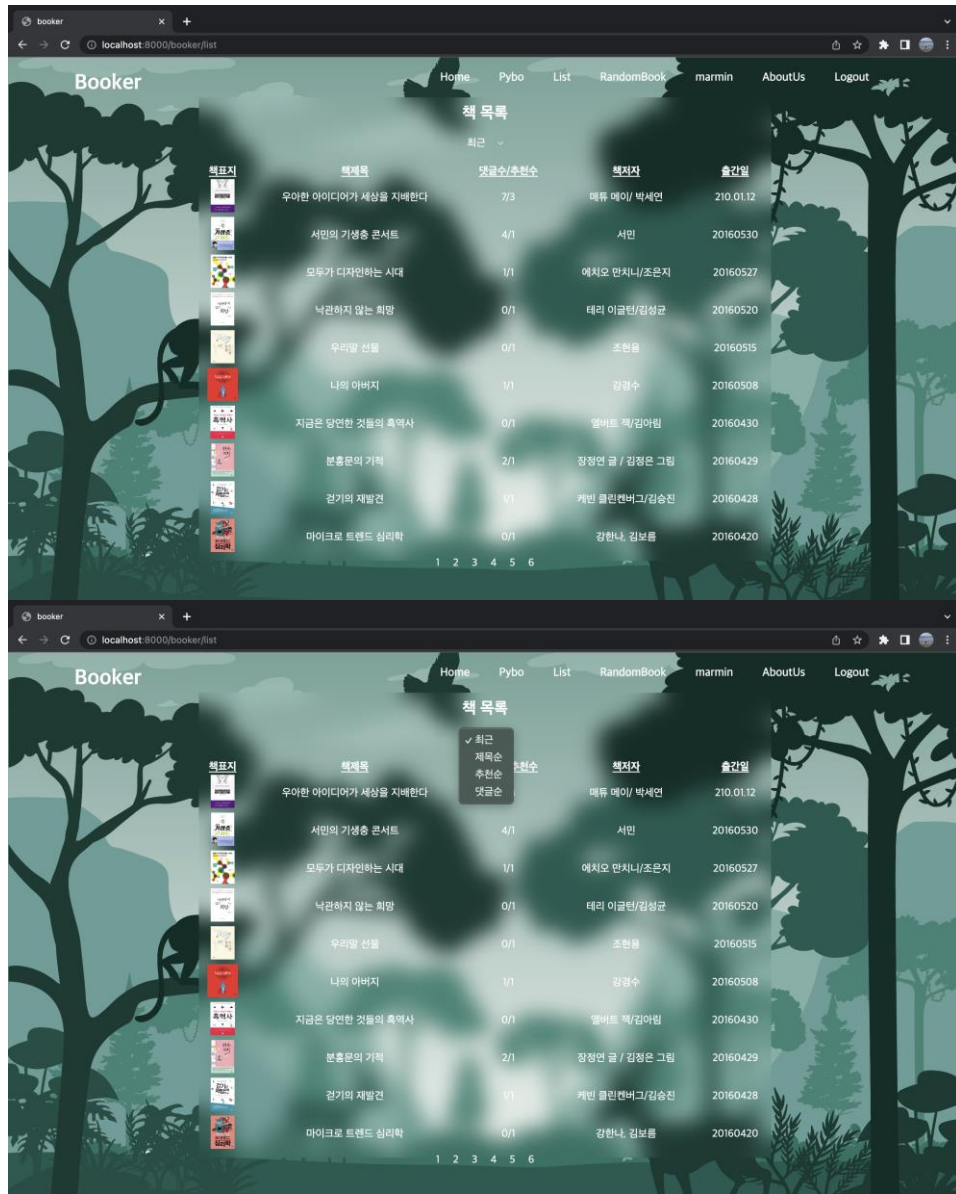
●문화데이터 광장에서 API키를 받아 따로 저장된 API키를 불러온다

●저장된 API키로 문화데이터 광장에 request라이브러리를 통해 데이터를 요청한다

●JSON 형식으로 받은 데이터를 필요한 데이터만 필터링하여 데이터 베이스에 저장하고 자동으로 인덱싱 된다

●View에서 바로바로 요청하여 사용도 가능하나 인덱스 번호와 데이터의 중복 등을 체크하기 위해 데이터 베이스로 옮기는 방식으로 구현하였다

BOOK LIST - BOOKER



● 받아온 데이터를 리스트로 나열한다 기본으로 최근 출간 된 순으로 나열이 되도록 정렬하였다

● 받아온 데이터를 10개씩 불러와 나열하고 페이지를 구현하여 페이지를 넘겨 다음 10개 데이터를 확인이 가능하다

● 표지사진, 제목, 추천 수, 댓글 수, 저자, 출간일 을 나열하였다

● 추천 순/ 댓글 순/ 제목 순 정렬이 가능하도록 설정하였다

● 정렬기준을 바꾼 뒤 페이지를 넘기면 생기는 정렬기준이 다시 디폴트 설정으로 돌아가는 현상은 자바스크립트로 해결하였다

● JQuery를 통해 정렬기준과 페이지를 서버로 전송한다 JQuery 는 base.html 템플릿을 지정하여 cdn으로 불러왔다

BOOK LIST - BOOKER

```
<script type="text/javascript">
  $(document).ready(function(){
    $(".page-link").on('click',function(){
      $("#page").val($(this).data("page"));
      $("#searchForm").submit();
    });
    $("#ac").on('change',function(){
      $("#ca").val($(this).val());
      $("#page").val(1);
      $("#searchForm").submit();
    });
  });
</script>
```

● 데이터를 전달하기 위한 Jquery코드이다

페이지의 항목 중 카테고리나 페이지를 id와 class값으로 찾아와 찾아온 데이터를 query string 으로 보내는 항목이다

BOOK LIST - BOOKER

```
<script>
const url = new URL(window.location.href);
const urlParams = url.searchParams;
const valCa = urlParams.get('ca');

if (valCa == "title") {
  let title = document.getElementById("ca-title");
  title.setAttribute("selected", true);
} else if (valCa == "liked") {
  let liked = document.getElementById("ca-liked");
  liked.setAttribute("selected", true);
} else if (valCa == "comment") {
  let comment = document.getElementById("ca-comment");
  comment.setAttribute("selected", true);
} else {
  let recent = document.getElementById("ca-recent");
  recent.setAttribute("selected", true);
}

const ca = document.getElementById('ca')
ca.value = valCa
</script>
```

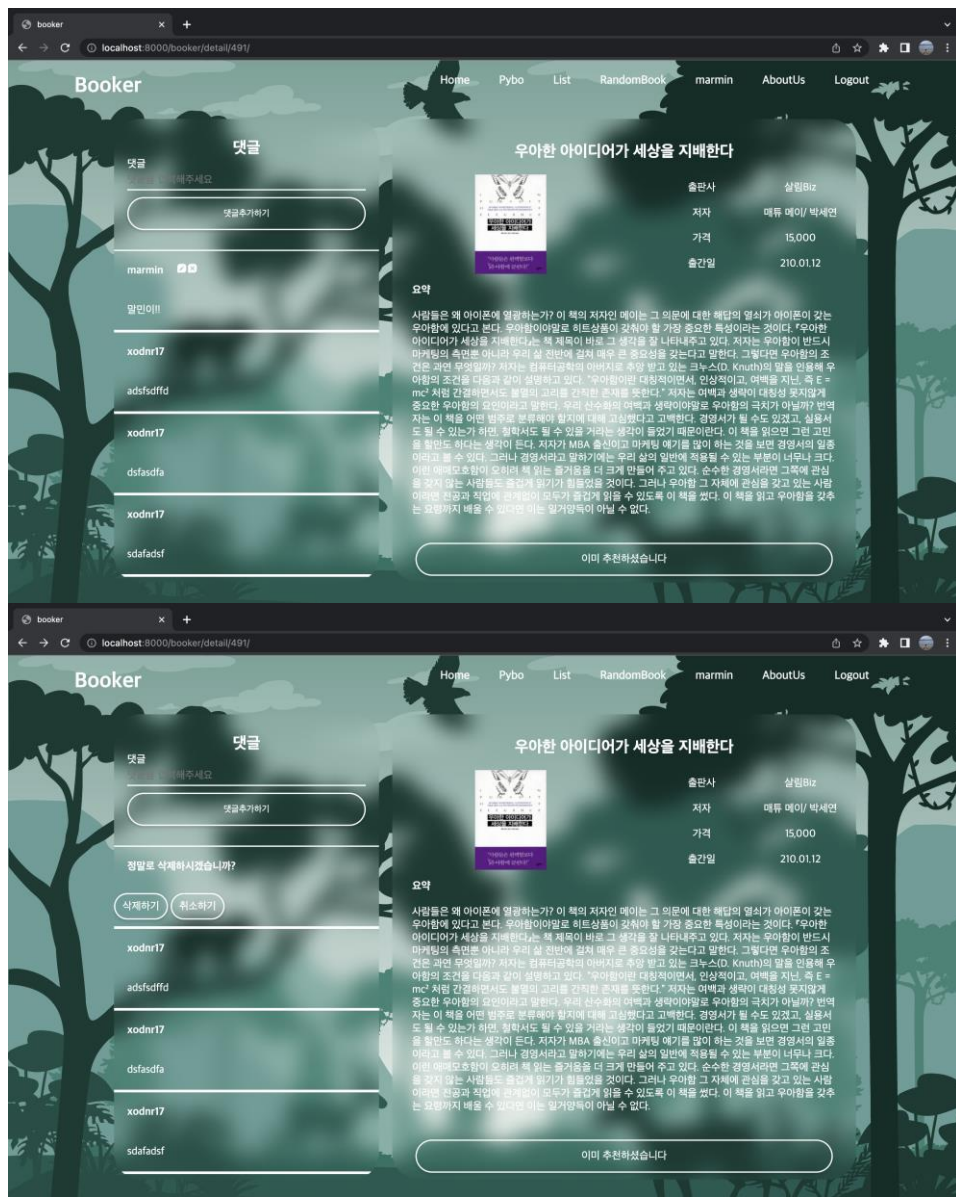
● 책 카테고리를 지정하는 코드이다.

Options과 select 를 사용하여 원하는 카테고리 선택 시 풀리지 않도록 구현하였다.

url 에서 카테고리 쿼리스트링을 검사하여 그 쿼리가 맞다면 옵션 항목을 선택하는 속성을 부여하는 코드이다.

마지막 코드 두 줄은 페이지 변경 시 카테고리 속성이 풀리는 것을 방지하기 위한 코드로 이 코드로 인해 페이지를 변경해도 카테고리가 변경되지 않는다

BOOK DETAIL - BOOKER



●책 데이터의 상세 정보를 보여준다. 좌측은 책에 대한 댓글을 표시하고 우측은 책의 상세정보를 보여준다

●좌측 상단의 입력창에서 댓글을 입력받아 댓글을 추가가 가능하도록 로그인 되어있지 않다면 자동으로 로그인으로 이동한다

●추가한 댓글은 아이디가 일치하면 옆에 수정, 삭제 아이콘이 나타나는데, 수정아이콘은 수정 폼으로 이동한다

●삭제 아이콘 클릭 시 아래와 같이 삭제 확인 창이 나타난다. 이는 자바스크립트를 통해 class항목을 조회하는 방식으로 구현하였다.

●우측은 책의 타이틀, 표지사진, 책의 요약, 출간일, 저자, 가격을 나타내는 상세 페이지이다.

●아래 추천하기/ 이미 추천하셨습니다 버튼을 클릭하여 추천/ 추천해제가 가능하다.

BOOK DETAIL - BOOKER

```
function deleteModal(n) {
    var cc = document.getElementsByClassName(n);
    var ci = document.getElementById(n);
    for (var i = 0; i < cc.length; i++) {
        cc[i].style.display = "none";
    }
    ci.style.display = "block";
}

function unDeleteModal(n) {
    var cc = document.getElementsByClassName(n);
    var ci = document.getElementById(n);
    for (var i = 0; i < cc.length; i++) {
        cc[i].style.display = "block";
    }
    ci.style.display = "none";
}
```

● BookDetail에서 삭제 확인 창을 띄우기 위한 코드이다

아이디를 장고 템플릿 코드를 이용하여 각각 다르게 지정되게 하고 for 문을 돌아 id 값이 맞는 항목의 삭제 확인 창을 띄우는 로직이다.

취소 버튼을 누를시 같은 로직으로 삭제 확인 창을 사라지게 하는 코드이다.

BOOK DETAIL - BOOKER

```
def book_detail(request, book_id):
```

```
    Comment.objects.filter(post=book).order_by(['-create_at'])
```

```
    if request.method == "POST":
```

```
        if request.user.is_authenticated:
```

```
            form = CommentForm(request.POST)
```

```
            if form.is_valid():
```

```
                comment = form.save(commit=False)
```

```
                comment.author = request.user
```

```
                comment.create_at = timezone.now()
```

```
                comment.post = book
```

```
                comment.save()
```

```
                return redirect('booker:detail', book_id)
```

```
        else:
```

```
            return redirect('login')
```

```
    else:
```

```
        form = CommentForm()
```

```
        liked = False
```

```
        if request.user in book.like.all():
```

```
            liked = True
```

```
    context = {'book':book, 'form':form, 'liked':liked}
```

```
    return render(request, 'booker/book-detail.html', context)
```

● BookDetail에서 삭제 확인 창을 띄우기 위한 코드이다

댓글은 최근 목록부터 나타나도록 하였다
만약 HTTP POST메서드로 요청이 들어올 시
유저가 로그인 되어있고, 폼이 유효하다면 폼 정보를
저장하고 다시 그 디테일 페이지로 리디렉트한다

만약 POST 메서드가 아닐 시 좋아요 여부를 검사,
댓글 리스트를 불러와 페이지에 내용으로 담고 책
정보를 담아 템플릿에 렌더링 한다.

BOOK DETAIL - BOOKER

```
@login_required(login_url='login')
def book_like(request, book_id):
    """
    책 추천하기 버튼 누르기
    """
    book = get_object_or_404(Book, pk=book_id)
    book.like.add(request.user)
    book.save()
    return redirect('booker:detail',book_id)
```

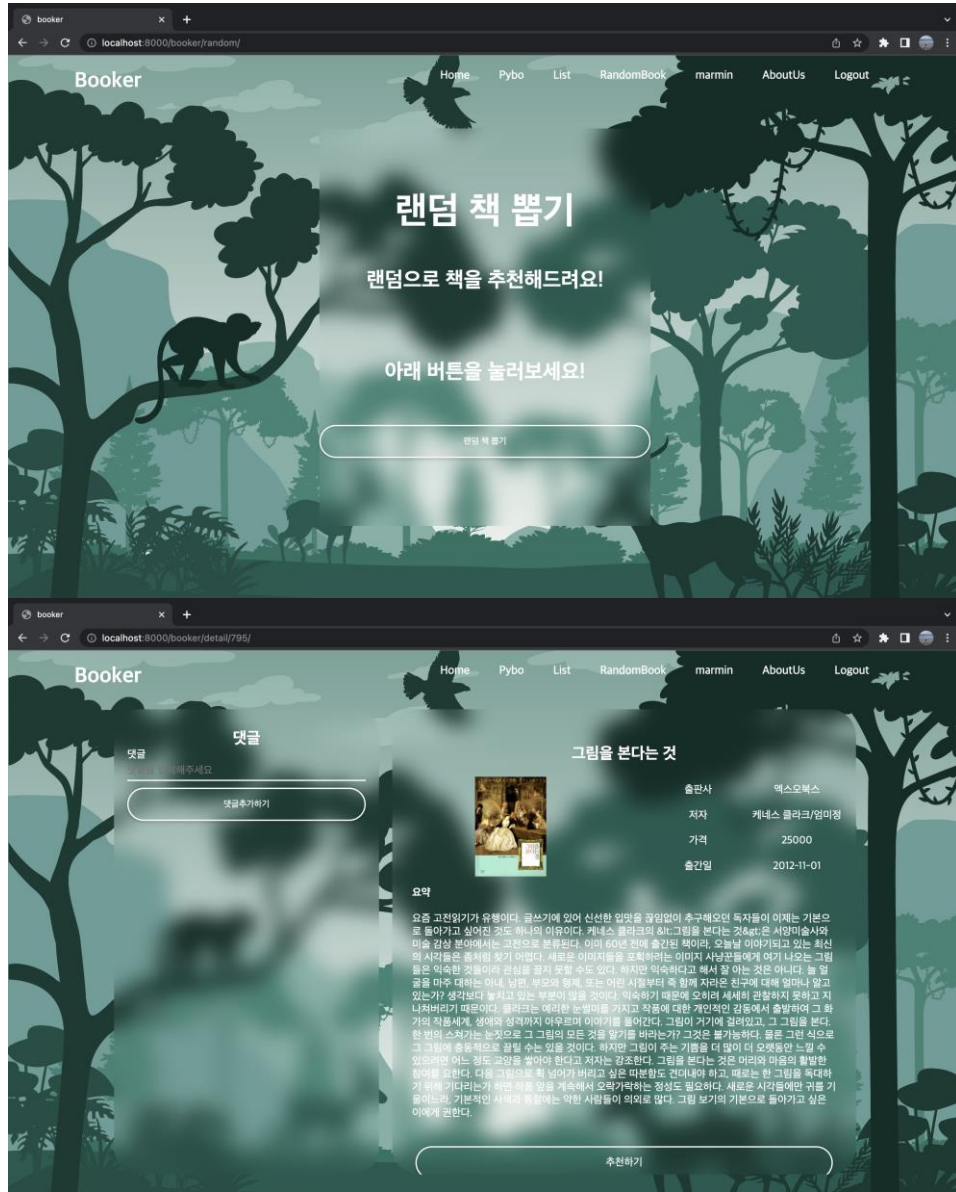
```
@login_required(login_url='login')
def book_unlike(request, book_id):
    """
    책 추천 풀기
    """
    book = get_object_or_404(Book, pk=book_id)
    book.like.remove(request.user)
    book.save()
    return redirect('booker:detail',book_id)
```

● BookDetail에서 추천 버튼을 구현하기 위한 코드다

책 추천버튼을 누를 경우 책을 불러와 like 에
추가한다. 추가된 정보를 데이터 베이스에 저장한다
그리고 다시 리디렉트된다

같은 로직으로 책 추천 풀기 코드이다.

RANDOM BOOK - BOOKER



●랜덤 책 뽑기로 인덱스 번호를 사용하여 랜덤 책 상세페이지로 이동하는 기능이다. 아래 버튼을 눌러 이동한다

●로그인이 되어있지 않다면 자동으로 로그인으로 이동한다

●Random모듈을 사용하여 구현하였다

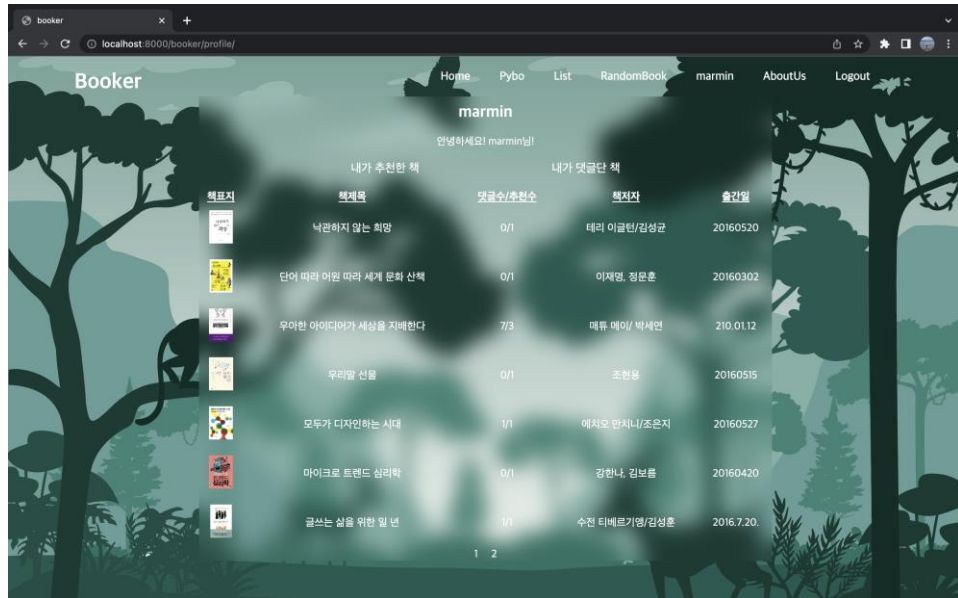
RANDOM BOOK - BOOKER

```
def book_reference(request):  
    """  
    랜덤 책 뽑기  
    """  
    book_id = random.randrange(1, 819)  
    if request.method == 'POST':  
        if request.user.is_authenticated:  
            return redirect('booker:detail', book_id)  
        else:  
            return redirect('login')  
    return render(request, 'booker/book-random.html')
```

● 랜덤 책 뽑기 코드이다

책 아이디는 1부터 819번 책의 마지막 인덱스 번호까지로 설정하였으며 len()으로 대체 가능하다.
POST와 GET의 로그인 여부가 다르므로 @loginrequired 대신 내부에서 로그인여부를 검증하는 코드를 사용하였다

BOOK PROFILE - BOOKER



● 내가 좋아요를 누른 항목과 댓글을 단 항목을 확인 가능한 페이지이다.

● 내가 추천한 책과 댓글을 단 책을 클릭하여 확인이 가능하고, JQuery 를 통해 데이터를 전송한다

● 내가 추천한 책 제목을 클릭하여 책 상세 페이지로 이동이 가능하다.

● 페이지 네이션을 구현하여 8개씩 데이터가 표시되고 다음장으로 넘기면 다음 8 개 데이터가 나타난다

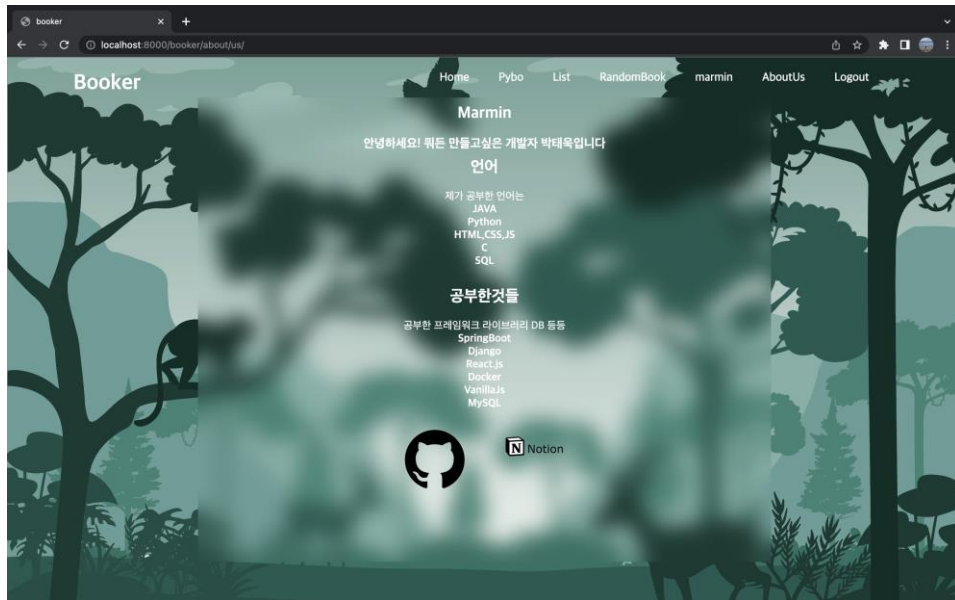
BOOK PROFILE - BOOKER

```
def book_profile(request):
    """
    프로필 리스트 불러오기
    """
    sa = request.GET.get('sa','liked')
    page = request.GET.get('page','1')
    user = request.user
    if sa == 'comment':
        comments = Comment.objects.filter(author=request.user)
        book_list = [comment.post for comment in comments]
        book_title_set = set()
        books = []
        for book in book_list:
            if book.title not in book_title_set:
                book_title_set.add(book.title)
                books.append(book)
    else:
        books = Book.objects.filter(like=user)
    paginator = Paginator(books,7)
    page_obj = paginator.get_page(page)
    context = {'books': page_obj}
    return render(request, 'booker/profile.html', context)
```

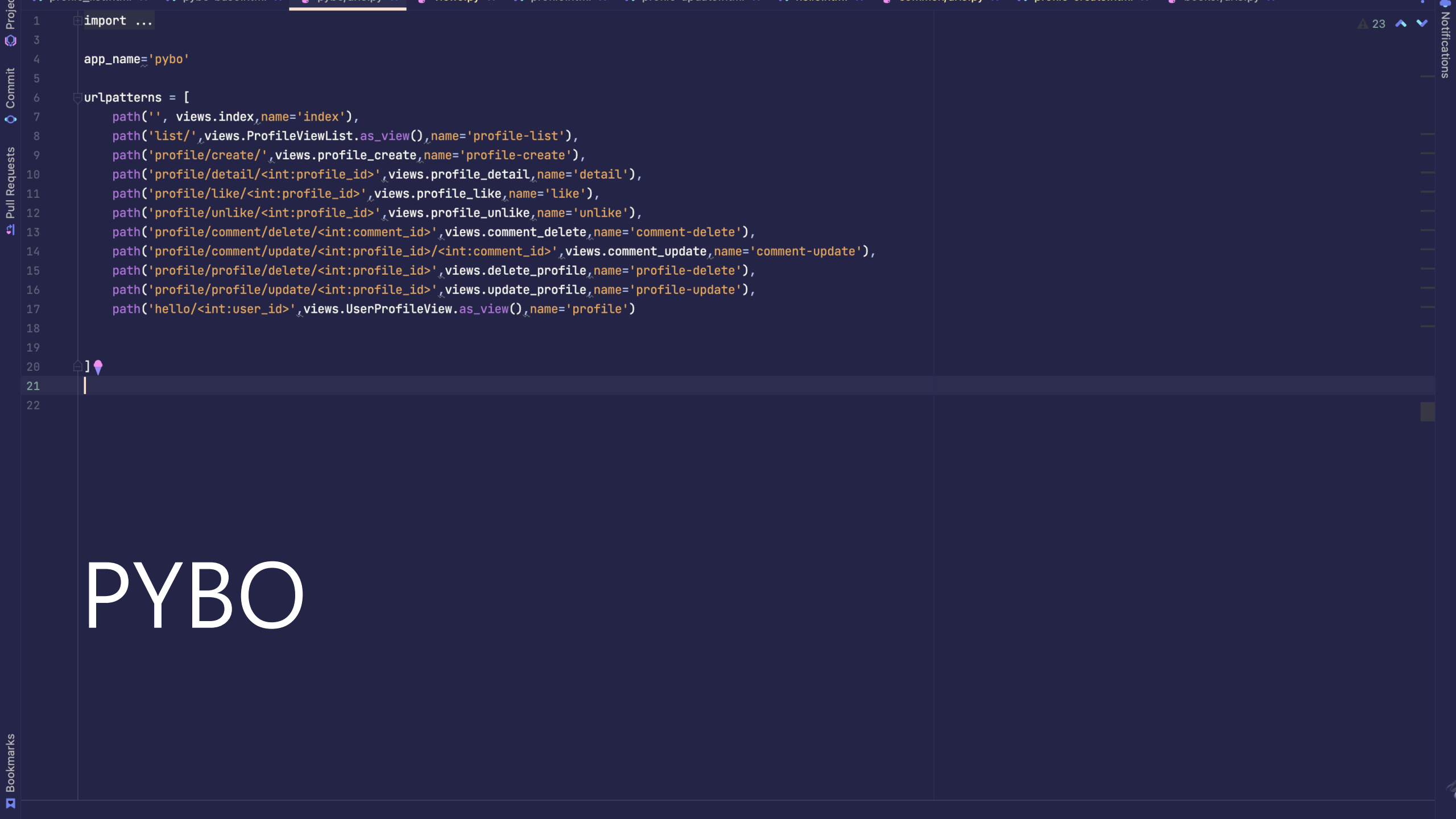
● 내 프로필 코드이다.

내가 작성한 댓글이나 좋아요 한 책을 구분지어서 볼 수 있도록
쿼리스트링으로 받고 받은 쿼리스트링을 통해
ORM 을 각각 지정하여 다른 항목을 불러오도록 한다
또한 페이지네이터를 구현하여 페이지징을 가능하도록
하였다

ABOUT US - BOOKER



- about us 페이지에는 개발자 정보를 넣었다
- 여태 공부한 언어와 프레임워크 등등 여태 공부한 것들을 소개하는 페이지이다
- 깃허브와 노션의 아이콘을 넣고 a 태그에 포함시켜 해당 아이콘을 클릭할 경우 개발자의 깃허브와 노션으로 이동하도록 하였다



PYBO

```
1 from django.db import models
2 from django.contrib.auth.models import User
3
4 PYBO MODELS.PY - PYBO
5
6 class Profile(models.Model):
7     title = models.CharField(max_length=100)
8     author = models.CharField(max_length=100)
9     content = models.TextField(max_length=1000)
10    create_at = models.DateTimeField()
11    like = models.ManyToManyField(User, blank=True)
12
13    Marmin
14
15    def __str__(self):
16        return self.title
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

- Profile 모델은 사용자에게 입력받아서 저장하는 모델로, Title, author, content 필드는 사용자가 폼에서 입력하여 저장한다

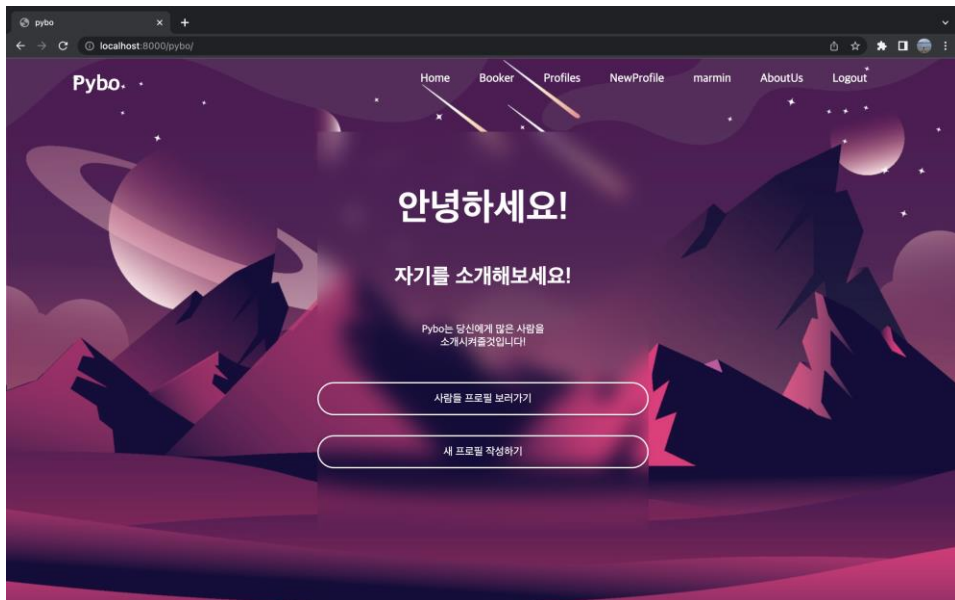
- like는 many to many 필드를 사용하여 유저와 다대다 매핑을 통해 추천을 받을 수 있도록 하였다

- create_at은 저장하는 순간 현재 시간이 자동으로 저장된다

- 프로필 커멘트는 content 필드를 폼으로 입력받고 게시물, 작성자, 작성일시는 자동으로 등록된다.

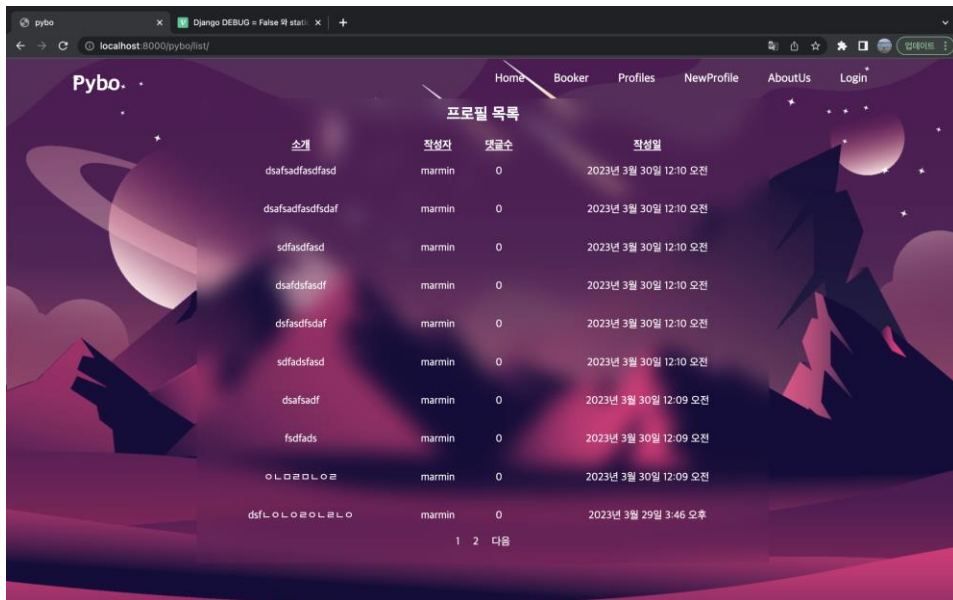
- 추후 댓글 좋아요 기능을 추가하기 위해 like 필드를 추가하였다

PYBO HOME - PYBO



- 자기 소개 게시물을 작성하는 파이보 애플리케이션 메인 페이지다
- 사람들 프로필 보러가기 버튼 클릭시 게시물 리스트 페이지로 이동한다
- 새 프로필 작성하기 버튼 클릭시 새로운 게시물 폼으로 이동한다
- Booker클릭시 Booker 앱으로 이동한다
- NewProfile클릭시 새로운 게시물 폼으로 이동한다
- about us 클릭 시 개발자 정보가 포함된 페이지로 이동한다

PROFILE LIST - PYBO



- 사람들이 올린 게시물을 볼 수 있는 페이지이다.
- 제목을 클릭시 원하는 게시물로 이동한다
- 제목, 작성자 댓글 수 , 작성 일시를 확인이 가능하다
- 페이지 네이션을 구현하여 다음 페이지로 넘어가기가 가능하다
- 화면당 10개의 페이지가 표시된다

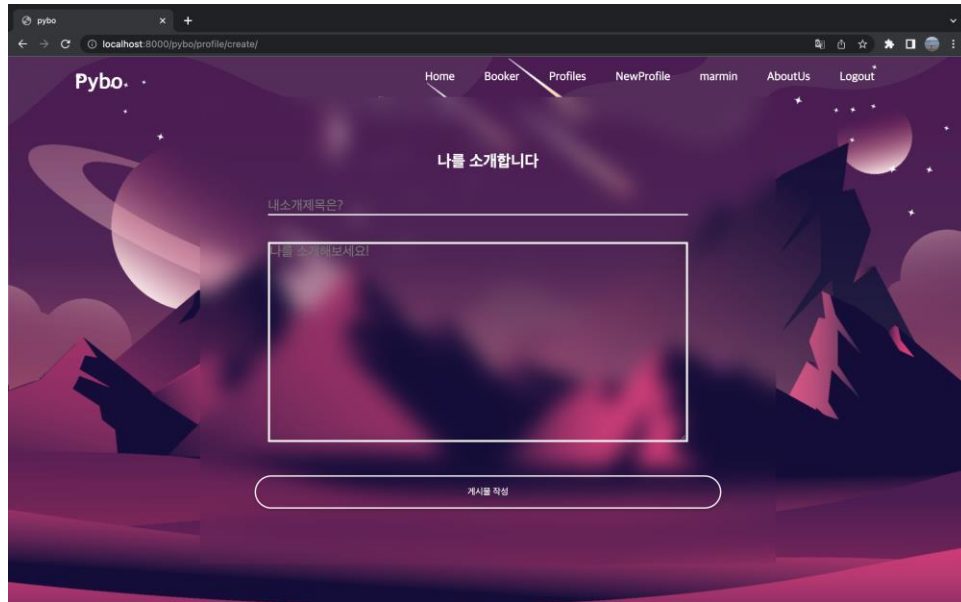
PROFILE LIST - PYBO

```
class ProfileViewList(ListView):  
    model = Profile  
    template_name = 'profile_list.html'  
    context_object_name = 'profiles'  
    ordering = ['-create_at']  
    paginate_by = 10
```

- 리스트뷰로 정의한 프로필 리스트이다

Booker 에서 사용한 함수형 보다 간단한 형식이며 Django.generic.view 의 ListView를 상속받아 구현하였다. Profile 모델을 사용하고 템플릿 이름, 답을 내용, 쿼리문 몇 개의 항목을 불러와 페이지네이션 할것인지에 대해 구현하였다

PROFILE CREATE - PYBO



- 새 게시물을 작성하는 페이지이다. 작성후 게시물 작성 버튼을 클릭시 게시물이 저장된다
- 맨 위 필드는 제목을 입력하는 필드이다
- 아래 텍스트 에리어는 글의 내용을 적는 필드이다

PROFILE CREATE - PYBO

```
@login_required(login_url='login')
def profile_create(request):
    if request.method == "POST":
        form = ProfileForm(request.POST)
        if form.is_valid():
            profile = form.save(commit=False)
            profile.author = request.user
            profile.create_at = timezone.now()
            profile.save()
            return redirect('pybo:profile-list')
    else:
        form=ProfileForm()
    context = {'form':form}
    return render(request,'pybo/profile-create.html',context)
```

● 새 게시물을 작성하기 위한 로직이다.

이 페이지는 GET, POST 모두 로그인을 해야 가능하도록 Login_required 어노테이션을 추가하여 로그인한 사용자만 메서드 요청을 가능하도록 하였다.

만약 로그인 되어있지 않다면 로그인 창으로 넘어가게 된다

HTTP POST 메서드로 입력된다면 POST 폼이 유효한지 아닌지 검증 후 게시물을 저장하고 프로필 리스트로 리디렉트한다 아니라면 폼을 전송해 입력이 가능하도록 한다

PROFILE DETAIL - PYBO



- 사용자들이 올린 게시물의 상세 페이지를 볼 수 있는 페이지이다
- 좌측은 댓글을 추가 할 수 있는 폼이 제공되고 우측은 상세 정보를 표시하는 페이지이다.
- 좌측 댓글에는 내가 추가한 댓글일 경우 삭제와 수정 아이콘이 나타나며 삭제 버튼 클릭시 삭제 확인 창이 나타난다
- 우측 상세 페이지에는 제목, 작성자, 내용, 작성일이 표시되며 작성자는 이 페이지에서 삭제 및 수정 버튼을 확인가능하다
- 글 추천하기를 클릭 시 이미 추천하셨습니다 버튼으로 교체되며 버튼을 다시 클릭시 추천하기로 돌아온다.
- 글 삭제하기 버튼을 클릭 시 삭제 확인 버튼으로 교체되며 다시 클릭하면 게시물이 삭제된다
- 게시글 수정하기 버튼 클릭시 게시물을 수정하는 폼으로 돌아간다

PROFILE DETAIL - PYBO

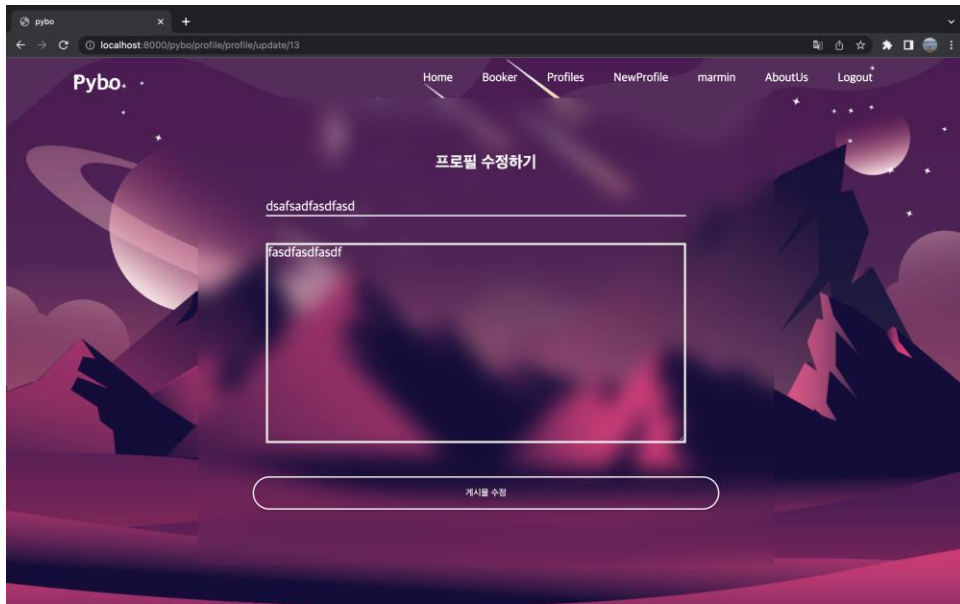
```
def profile_detail(request, profile_id):
    profile = get_object_or_404(Profile, pk=profile_id)
    # comment = Comment.objects.filter(post=book).order_by(['-
create_at'])
    if request.method == "POST":
        if request.user.is_authenticated:
            form = ProfileCommentForm(request.POST)
            if form.is_valid():
                comment = form.save(commit=False)
                comment.author = request.user
                comment.create_at = timezone.now()
                comment.profile = profile
                comment.save()
                return redirect('pybo:detail', profile_id)
            else:
                return redirect('login')
        else:
            form = ProfileCommentForm()
            liked = False
            if request.user in profile.like.all():
                liked = True
    context = {'profile': profile, 'form': form, 'liked': liked}
    return render(request, 'pybo/detail.html', context)
```

● 게시물의 상세정보를 보기 위한 로직이다

게시물을 보기 위한 로직으로 POST 방식으로 댓글을 추가할 수 있다. 이 로직은 POST 방식으로 댓글을 추가하기 위해서는 로그인을 해야하기 때문에 내부에서 로그인 여부를 확인하고, 로그인 되어 있지 않다면 로그인 url로 리디렉트 된다.

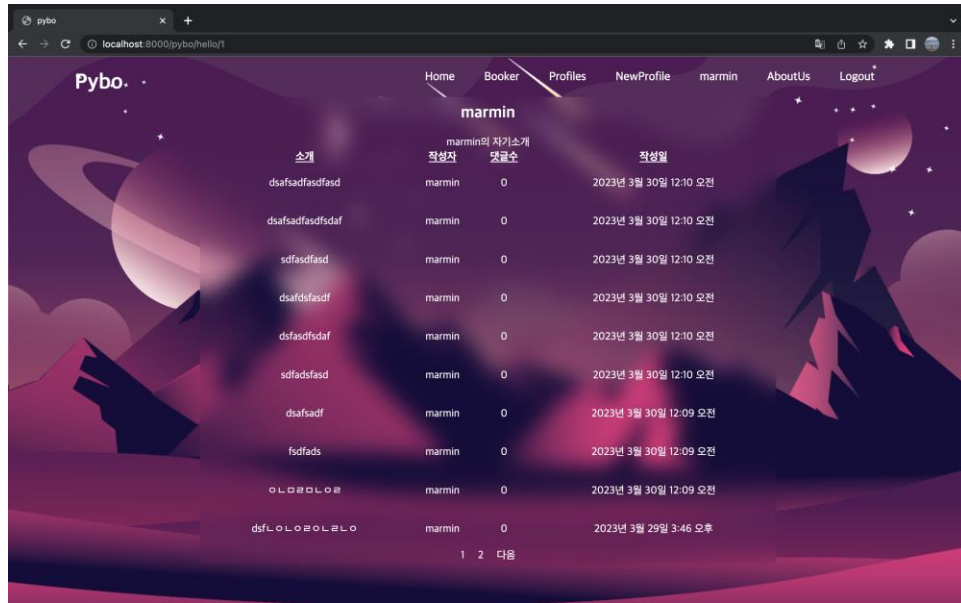
GET 방식은 로그인이 필요없도록 구현하여 상세정보와 댓글을 확인이 가능하다

PROFILE UPDATE - PYBO



- 게시물을 수정하는 폼이다.
- 맨 위 필드는 제목을 입력하는 필드이다
- 아래 텍스트 에리어는 글의 내용을 적는 필드이다

PROFILE USER - PYBO



- 내가 작성한 자기소개 게시글을 볼 수 있는 페이지이다
- 맨 위에는 사용자의 이름이 표시된다
- 데이터를 10개씩 불러오며 페이지네이션을 구현하여 다음 페이지를 확인이 가능하도록 하였다
- 제목을 클릭 할 경우 해당 게시물의 상세 페이지로 이동한다

PROFILE USER - PYBO

```
class UserProfileView(ListView):
    model = Profile
    template_name = 'pybo/hello.html'
    context_object_name = 'profiles'
    paginate_by = 10
    def get_queryset(self):
        queryset = super().get_queryset()
        return queryset.filter(author=self.request.user).
order_by('-create_at')
```

● 내 프로필을 보여주는 함수이다

ListView를 상속받아서 구현하였다

페이지네이션, 답을 내용등을 정의하였고, 쿼리셋을 직접
정의하여 원하는 정보만 불러올 수 있도록 정의하였다



PYBO-BOOKER WEB APP