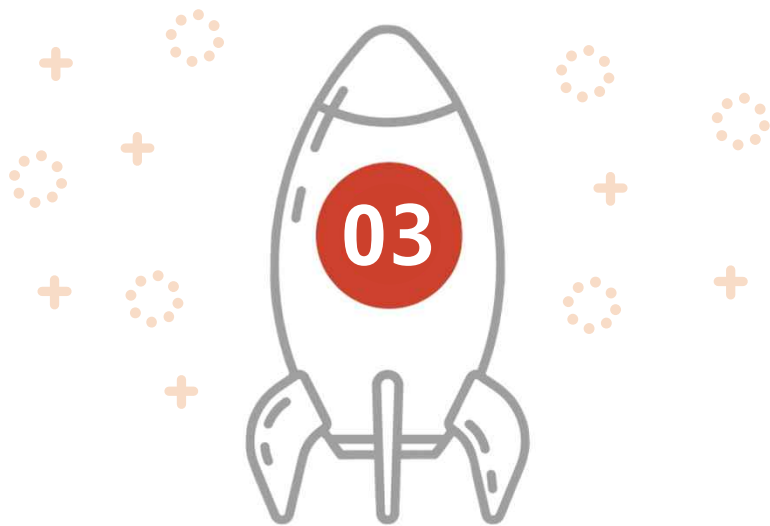


5주차 강의자료

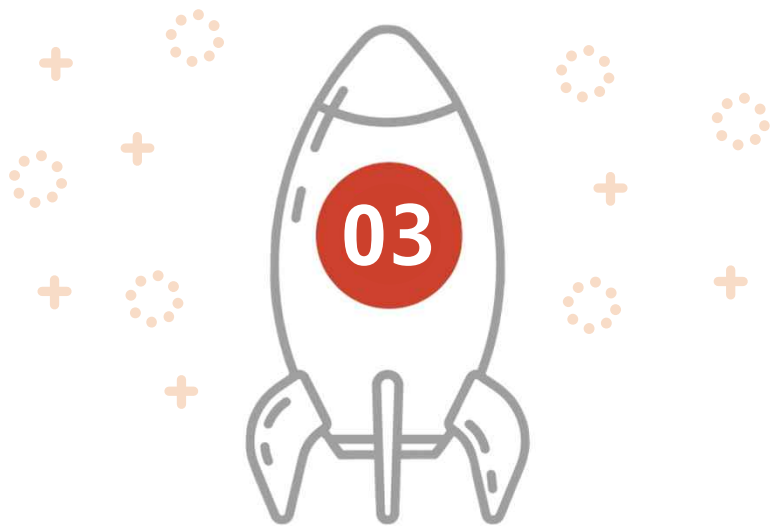
# 웹서버컴퓨팅



**파이보 서비스 개발!**

# INDEX

- 03-1 내비게이션 기능 추가하기
- 03-2 게시판 페이징 기능 추가하기
- 03-3 템플릿 필터 직접 만들어 보기
- 03-4 질문에 달린 답변 개수 표시하기
- 03-5 로그인, 로그아웃 구현하기
- 03-6 회원가입 구현하기



파이보 서비스 개발!

# INDEX

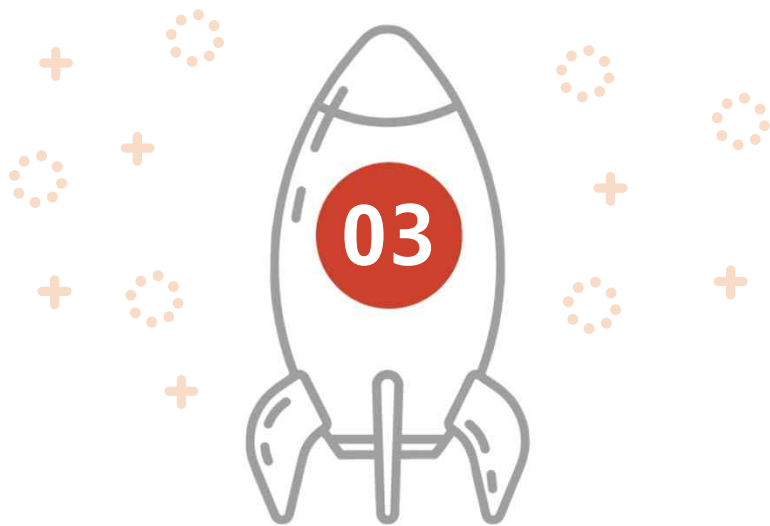
- 03-7 모델에 글쓰기 추가하기
- 03-8 글쓰기 표시하기
- 03-9 게시물 수정 & 삭제 기능 추가하기
- 03-10 댓글 기능 추가하기
- 03-11 views.py 파일 분리하기



**파이보 서비스 개발!**

## INDEX

- 03-12 추천 기능 추가하기
- 03-13 스크롤 초기화 문제점 해결하기
- 03-14 마크다운 기능 적용하기
- 03-15 검색, 정렬 기능 추가하기
- 03-16 도전! 저자 추천 파이보 추가 기능



## 파이보 서비스 개발!

### 이장의 목표

- ✓ 파이보를 상용 게시판 수준으로 개발한다.
- ✓ 부트스트랩을 적용하여 서비스를 더 아름답게 만든다.
- ✓ 게시물 등록, 삭제, 수정부터 로그인, 로그아웃, 페이지, 검색까지 게시판을 완벽하게 만든다.

## 03-1 내비게이션바

▶ <https://wikidocs.net/71108> 웹사이트 내용 참조

<https://github.com/pahkey/djangobook/tree/3-01>

Do it!  
실습

### 내비게이션바 추가하기

#### 01단계 내비게이션의 필요성

▶ . 2-10 에서 보면, 메인 페이지로 돌아가는 장치가 없  
→ 내비게이션의 필요성

▶ . 내비게이션은 모든 페이지에서 보여야 함 →  
base.html에 내비게이션 기능 작성

번호	제목	작성일시
1	<a href="#">질문 등록 테스트입니다4</a>	2023년 4월 3일 12:58 오후
2	<a href="#">질문 등록 테스트입니다3</a>	2023년 4월 3일 12:53 오후
3	<a href="#">질문 등록 테스트입니다1</a>	2023년 4월 3일 11:39 오전
4	<a href="#">질문 등록 테스트입니다</a>	2023년 4월 3일 11:26 오전
5	<a href="#">점프 투 알고리즘도 있나요?</a>	2020년 5월 7일 9:52 오전
6	<a href="#">장고로 만들어진 유명한 사이트가 있나요?</a>	2020년 5월 7일 9:01 오전
7	<a href="#">Django Model Question</a>	2020년 5월 7일 8:51 오전

질문 등록하기

## 03-1 내비게이션바

▶ <https://wikidocs.net/71108> 웹사이트 내용 참조

Do it!  
실습

### 내비게이션바 추가하기

#### 01단계 부트스트랩 내비게이션 디자인 사용

▶ <https://getbootstrap.kr/docs/5.1/components/navbar/>

Navbar Home Link Dropdown ▾ Disabled Search

<https://github.com/pahkey/djangobook/tree/3-01>

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbar!"
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="#">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Link</a>
        </li>
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-bs-t
            Dropdown
          </a>
          <ul class="dropdown-menu" aria-labelledby="navbarDropdown">
            <li><a class="dropdown-item" href="#">Action</a></li>
            <li><a class="dropdown-item" href="#">Another action</a></li>
            <li><hr class="dropdown-divider"></li>
            <li><a class="dropdown-item" href="#">Something else here</a></li>
          </ul>
        </li>
        <li class="nav-item">
          <a class="nav-link disabled" href="#">Disabled</a>
        </li>
      </ul>
      <form class="d-flex">
        <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">
        <button class="btn btn-outline-success" type="submit">Search</button>
      </form>
    </div>
  </div>
</nav>
```

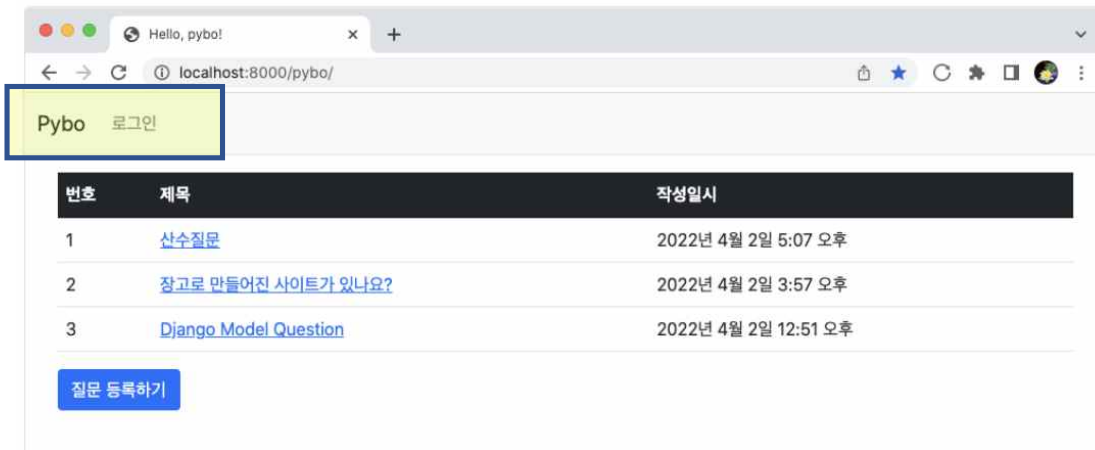
## 03-1 내비게이션바

▶ <https://wikidocs.net/71108> 웹사이트 내용 참조

Do it!  
실습

### 내비게이션바 추가하기

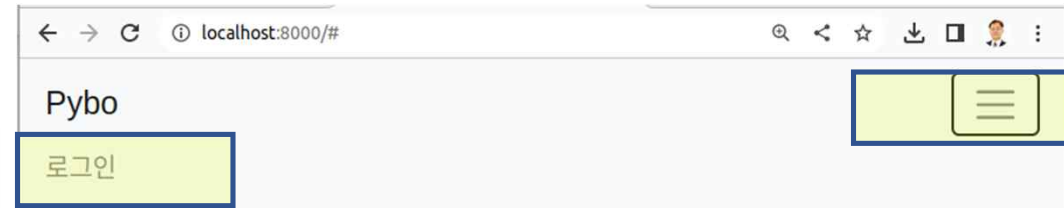
#### 01단계 로고, 로그인 링크 추가하기



▶ 항상 pybo:index 페이지로 이동해 주는 'Pybo' 로고를 가장 왼쪽에 배치했고, 오른쪽에는 '로그인' 링크를 추가했다. (로그인 기능은 나중에 구현한다).

<https://github.com/pahkey/djangobook/tree/3-01>

#### 02단계 반응형 웹 만들기



▶ 부트스트랩은 화면 크기가 작은 기기를 고려한 '반응형 웹' 까지 적용됨



## 03-1 내비게이션바

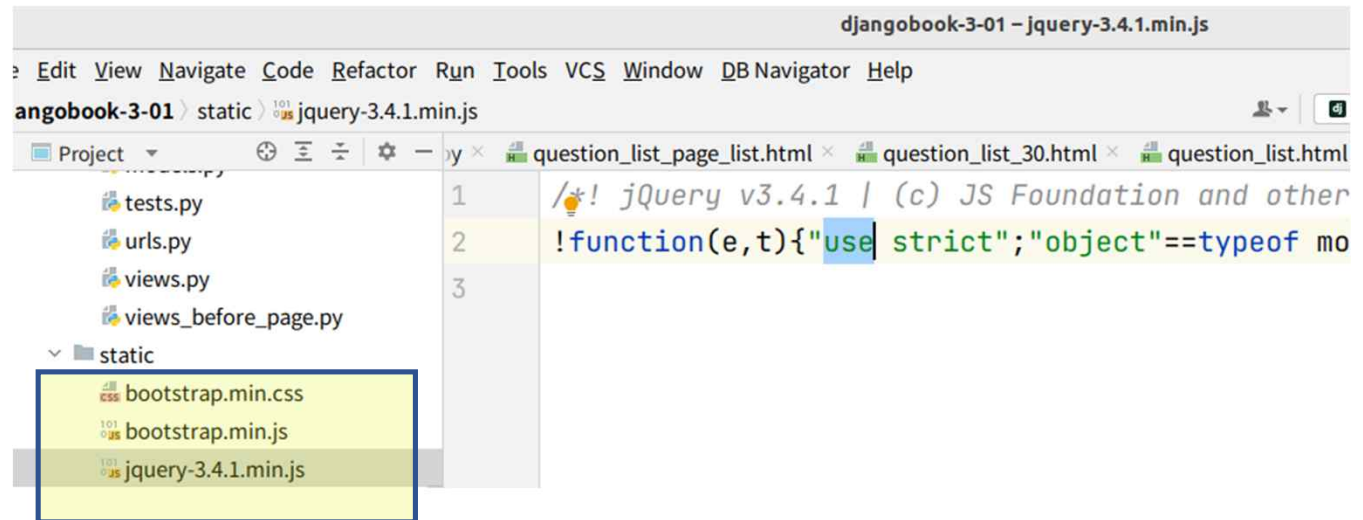
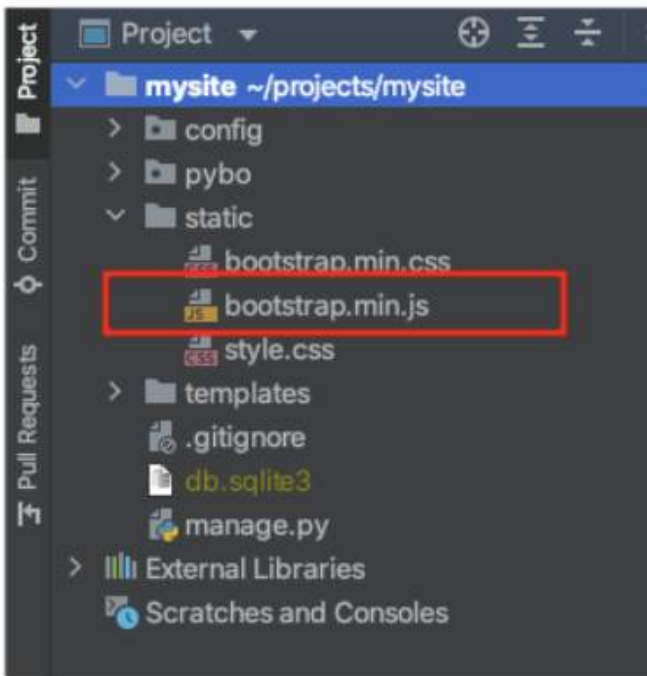
Do it!  
실습

<https://wikidocs.net/71108> 웹사이트 참조

<https://github.com/pahkey/djangobook/tree/3-01>

03단계 부트스트랩에 필요한 파일 추가하기 : 자바스크립트 파일

04단계 부트스트랩에 필요한 파일 추가하기 : 제이쿼리



- ▶ 다음처럼 projects\mysite\static 디렉터리 하위에 bootstrap.min.js 파일이 위치해 있어야 한다.

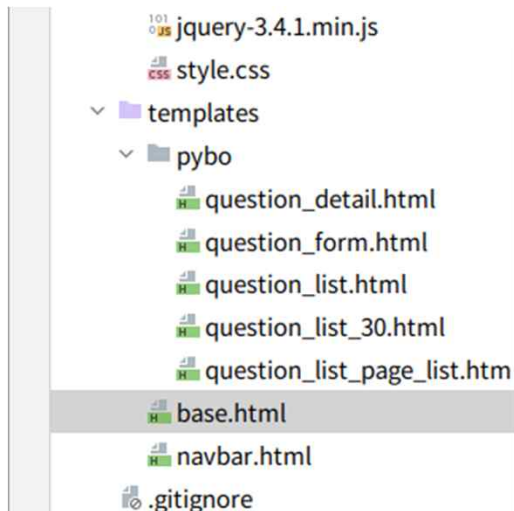
## 03-1 내비게이션바

Do it!  
실습

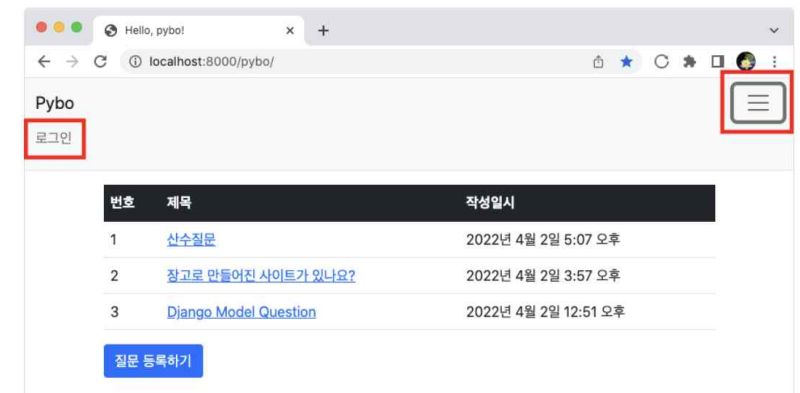
<https://wikidocs.net/71108> 웹사이트 참조

<https://github.com/pahkey/djangobook/tree/3-01>

05단계 templates/base.html에 파일 추가하기



```
19 <!-- 기본 템플릿 안에 삽입될 내용 End -->
20 <!-- jQuery JS -->
21 <script src="{% static 'jquery-3.4.1.min.js' %}"></script>
22 <!-- Bootstrap JS -->
23 <script src="{% static 'bootstrap.min.js' %}"></script>
24 </body>
25 </html>
```



- ▶ 파일 위치를 확인하고 base.html 파일을 열어 </body> 태그 바로 위에 코드를 추가함
- ▶ 햄버거 메뉴 버튼을 누르면 숨어 있는 링크가 표시됨

## 03-1 내비게이션바

Do it!  
실습

include 기능으로 내비게이션바 추가

01단계 templates/navbar.html 생성하고 코드 작성하기

- ▶ navbar.html 파일 코드는 base.html 파일에 작성했던 내비게이션을 위한 HTML과 같다
- ▶ 내비게이션바와 관련된 코드를 분리했다고 생각하면 된다

[파일명: projects\mysite\templates\navbar.html]

<https://github.com/pahkey/djangobook/tree/3-01>

```
<!-- 네비게이션바 -->
<nav class="navbar navbar-expand-lg navbar-light bg-light border-bottom">
  <div class="container-fluid">
    <a class="navbar-brand" href="{% url 'pybo:index' %}">Pybo</a>
    <button class="navbar-toggler" type="button"
      data-bs-toggle="collapse"
      data-bs-target="#navbarSupportedContent"
      aria-controls="navbarSupportedContent"
      aria-expanded="false"
      aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link" href="#">로그인</a>
        </li>
      </ul>
    </div>
  </div>
</nav>
```

## 03-1 내비게이션바

[파일명: projects\mysite\templates\base.html]

Do it!  
실습

### include 기능으로 내비게이션바 추가

#### 02단계 templates/base.html 에 include 적용하기

- ▶ navbar.html 파일 코드는 base.html 파일에 작성했던 내비게이션을 위한 HTML과 같다
- ▶ 내비게이션바와 관련된 코드를 분리했다고 생각하면 된다
- ▶ 네비게이션바 HTML 코드들을 삭제하고 navbar.html 템플릿을 include 태그로 포함시켰다.
- ▶ navbar.html 파일은 다른 템플릿들에서 중복되어 사용되지는 않지만 독립된 하나의 템플릿으로 관리하는 것이 유지 보수에 유리하므로 분리하였다.

<https://github.com/pahkey/djangobook/tree/3-01>

```
{% load static %}
<!doctype html>
<html lang="ko">
<head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <!-- Bootstrap CSS -->
  <link rel="stylesheet" type="text/css" href="{% static 'bootstrap.min.css' %}">
  <!-- pybo CSS -->
  <link rel="stylesheet" type="text/css" href="{% static 'style.css' %}">
  <title>Hello, pybo!</title>
</head>
<body>
  <!-- 네비게이션바 -->
  {% include "navbar.html" %}
  <!-- 기본 템플릿 안에 삽입될 내용 Start -->
  {% block content %}
  {% endblock %}
  <!-- 기본 템플릿 안에 삽입될 내용 End -->
  <!-- Bootstrap JS -->
  <script src="{% static 'bootstrap.min.js' %}"></script>
</body>
</html>
```

## 03-2 게시판 페이징 기능 추가하기

Do it!  
실습

### 임시 질문 데이터 300개 생성하기

<https://github.com/pahkey/djangobook/tree/3-02>

#### 01단계 장고 셸 실행하고 필요한 모듈 임포트하기

```
(mysite) c:\projects\mysite>python manage.py shell
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>>
```

- ▶ 이어서 질문 데이터를 생성하기 위한 모듈을 임포트하자.

```
>>> from pybo.models import Question
>>> from django.utils import timezone
```

#### 02단계 for문으로 테스트 데이터 300개 만들기

```
>>> for i in range(300):
...     q = Question(subject='테스트 데이터입니다:[%03d]' % i, content='내용무', create_date=timezone.now())
...     q.save()
... 
```

## 03-2 게시판 페이징 기능 추가하기

Do it!  
실습

### 임시 질문 데이터 300개 생성하기

#### 02단계 for문으로 테스트 데이터 300개 만들기

- ▶ 장고 셸로 등록한 테스트 데이터가 보일 것이다.
- ▶ 그리고 300개 이상의 데이터가 한 페이지 전부 보여지는 것을 확인할 수 있다.
- ▶ 이러한 이유로 페이징은 반드시 필요하다.

localhost:8000/pybo/			https://github.com/pahkey/djangobook/tree/3-02
Pybo 로그인			
번호	제목	작성일시	
1	<a href="#">테스트 데이터입니다:[299]</a>	2022년 4월 2일 5:48 오	
2	<a href="#">테스트 데이터입니다:[298]</a>	2022년 4월 2일 5:48 오	
3	<a href="#">테스트 데이터입니다:[297]</a>	2022년 4월 2일 5:48 오	
4	<a href="#">테스트 데이터입니다:[296]</a>	2022년 4월 2일 5:48 오	
5	<a href="#">테스트 데이터입니다:[295]</a>	2022년 4월 2일 5:48 오	
6	<a href="#">테스트 데이터입니다:[294]</a>	2022년 4월 2일 5:48 오	
7	<a href="#">테스트 데이터입니다:[293]</a>	2022년 4월 2일 5:48 오	
8	<a href="#">테스트 데이터입니다:[292]</a>	2022년 4월 2일 5:48 오	
9	<a href="#">테스트 데이터입니다:[291]</a>	2022년 4월 2일 5:48 오	
10	<a href="#">테스트 데이터입니다:[290]</a>	2022년 4월 2일 5:48 오	
11	<a href="#">테스트 데이터입니다:[289]</a>	2022년 4월 2일 5:48 오	



## 03-2 게시판 페이징 기능 추가하기

<https://github.com/pahkey/djangobook/tree/3-02>

Do it!  
실습

### 페이징 기능 살짝 구현해 보기

#### 01단계 Paginator

- ▶ 장고에서 페이징을 위해 사용하는 클래스는 Paginator이다.
- ▶ Paginator 클래스를 사용하여 다음과 같이 index 함수에 페이징 기능을 적용해 보자.

[파일명: projects\mysite\pybo\views.py]

```
from django.shortcuts import render, get_object_or_404, redirect
from django.utils import timezone
from .models import Question
from .forms import QuestionForm, AnswerForm
from django.core.paginator import Paginator

def index(request):
    page = request.GET.get('page', '1') # 페이지
    question_list = Question.objects.order_by('-create_date')
    paginator = Paginator(question_list, 10) # 페이지당 10개씩 보여주기
    page_obj = paginator.get_page(page)
    context = {'question_list': page_obj}
    return render(request, 'pybo/question_list.html', context)

(... 생략 ...)
```

## 03-2 게시판 페이징 기능 추가하기

Do it!  
실습

### 페이징 기능 살짝 구현해 보기

#### 01단계 Paginator

▶ 장고에서 페이징을 위해 사용하는 클래스는 Paginator이다.

▶ index 함수를 자세히 살펴보자.

▶ `page = request.GET.get('page', '1')`은

`http://localhost:8000/pybo/?page=1` 처럼 GET 방식으로 호출된 URL에서 page값을 가져올 때 사용한다.

▶ 만약 `http://localhost:8000/pybo/` 처럼 page값 없이 호출된 경우에는 디폴트로 1이라는 값을 설정한다

▶ 그리고 Paginator 클래스를 다음처럼 사용했다.

```
paginator = Paginator(question_list, 10) # 페이지당 10개씩 보여 주기
```

▶ 첫 번째 파라미터 `question_list`는 게시물 전체를 의미하는 데이터이고 두번째 파라미터 10은 페이지당 보여줄 게시물의 개수이다.

<https://github.com/pahkey/djangobook/tree/3-02>

[파일명: `projects\mysite\pybo\views.py`]

```
from django.shortcuts import render, get_object_or_404, redirect
from django.utils import timezone
from .models import Question
from .forms import QuestionForm, AnswerForm
from django.core.paginator import Paginator

def index(request):
    page = request.GET.get('page', '1') # 페이지
    question_list = Question.objects.order_by('-create_date')
    paginator = Paginator(question_list, 10) # 페이지당 10개씩 보여주기
    page_obj = paginator.get_page(page)
    context = {'question_list': page_obj}
    return render(request, 'pybo/question_list.html', context)

(... 생략 ...)
```



## 03-2 게시판 페이징 기능 추가하기

<https://github.com/pahkey/djangobook/tree/3-02>

Do it!  
실습

### 페이징 기능 살짝 구현해 보기

#### 01단계 Paginator

- ▶ paginator를 이용하여 요청된 페이지(page)에 해당되는 페이징 객체(page\_obj)를 생성했다.
- ▶ 이렇게 하면 장고 내부적으로는 데이터 전체를 조회하지 않고 해당 페이지의 데이터만 조회하도록 쿼리가 변경된다.

항목	설명
paginator.count	전체 게시물 개수
paginator.per_page	페이지당 보여줄 게시물 개수
paginator.page_range	페이지 범위
number	현재 페이지 번호
previous_page_number	이전 페이지 번호
next_page_number	다음 페이지 번호
has_previous	이전 페이지 유무
has_next	다음 페이지 유무
start_index	현재 페이지 시작 인덱스(1부터 시작)
end_index	현재 페이지의 끝 인덱스(1부터 시작)

[파일명: projects\mysite\pybo\views.py]

```
from django.shortcuts import render, get_object_or_404, redirect
from django.utils import timezone
from .models import Question
from .forms import QuestionForm, AnswerForm
from django.core.paginator import Paginator

def index(request):
    page = request.GET.get('page', '1') # 페이지
    question_list = Question.objects.order_by('-create_date')
    paginator = Paginator(question_list, 10) # 페이지당 10개씩 보여주기
    page_obj = paginator.get_page(page)
    context = {'question_list': page_obj}
    return render(request, 'pybo/question_list.html', context)

(... 생략 ...)
```

## 03-2 게시판 페이징 기능 추가하기

<https://github.com/pahkey/djangobook/tree/3-02>

Do it!  
실습

### 페이징 기능 살짝 구현해 보기

#### 01단계 Paginator

- ▶ 아직 까지는 10개만 보여주고 있다, 나머지 290개는 보여주 못하고 있다.

로그인

번호	제목	작성일시
1	<a href="#">테스트 데이터입니다:[299]</a>	2023년 4월 4일 3:12 오후
2	<a href="#">테스트 데이터입니다:[298]</a>	2023년 4월 4일 3:12 오후
3	<a href="#">테스트 데이터입니다:[297]</a>	2023년 4월 4일 3:12 오후
4	<a href="#">테스트 데이터입니다:[296]</a>	2023년 4월 4일 3:12 오후
5	<a href="#">테스트 데이터입니다:[295]</a>	2023년 4월 4일 3:12 오후
6	<a href="#">테스트 데이터입니다:[294]</a>	2023년 4월 4일 3:12 오후
7	<a href="#">테스트 데이터입니다:[293]</a>	2023년 4월 4일 3:12 오후
8	<a href="#">테스트 데이터입니다:[292]</a>	2023년 4월 4일 3:12 오후
9	<a href="#">테스트 데이터입니다:[291]</a>	2023년 4월 4일 3:12 오후
10	<a href="#">테스트 데이터입니다:[290]</a>	2023년 4월 4일 3:12 오후

질문 등록하기

## 03-2 게시판 페이징 기능 추가하기

Do it!  
실습

### 페이징 적용하기

#### 01단계 질문 목록 템플릿에 페이징 기능적용하기

- ▶ 상당히 많은 양의 HTML코드가 추가되었지만 어렵지 않으니 찬찬히 살펴보자.
- ▶ 이전 페이지가 있는 경우에는 "이전" 링크가 활성화되게 하였고 이전 페이지가 없는 경우에는 "이전" 링크가 비활성화되도록 하였다. (다음페이지의 경우도 마찬가지로 방법으로 적용되었다.)
- ▶ 그리고 페이지 리스트를 루프 돌면서 해당 페이지로 이동할 수 있는 링크를 생성하였다.
- ▶ 이때 현재 페이지와 같을 경우에는 active클래스를 적용하여 강조표시도 해 주었다..

[파일명: projects\mysite\templates\pybo\question\_list.html]

```
(... 생략 ...)  
</table>  
<!-- 페이징처리 시작 -->  
<ul class="pagination justify-content-center">  
  <!-- 이전페이지 -->  
  {% if question_list.has_previous %}  
  <li class="page-item">  
    <a class="page-link" href="?page={{ question_list.previous_page_number }}">이전</a>  
  </li>  
  {% else %}  
  <li class="page-item disabled">  
    <a class="page-link" tabindex="-1" aria-disabled="true" href="#">이전</a>  
  </li>  
  {% endif %}  
  <!-- 페이지리스트 -->  
  {% for page_number in question_list.paginator.page_range %}  
  {% if page_number == question_list.number %}  
  <li class="page-item active" aria-current="page">  
    <a class="page-link" href="?page={{ page_number }}">{{ page_number }}</a>  
  </li>  
  {% else %}  
  <li class="page-item">  
    <a class="page-link" href="?page={{ page_number }}">{{ page_number }}</a>  
  </li>  
  {% endif %}  
  {% endfor %}  
  <!-- 다음페이지 -->  
  {% if question_list.has_next %}  
  <li class="page-item">  
    <a class="page-link" href="?page={{ question_list.next_page_number }}">다음</a>  
  </li>  
  {% else %}  
  <li class="page-item disabled">  
    <a class="page-link" tabindex="-1" aria-disabled="true" href="#">다음</a>  
  </li>  
  {% endif %}  
</ul>  
<!-- 페이징처리 끝 -->  
  <a href="{% url 'pybo:question_create' %}" class="btn btn-primary">질문 등록하기</a>  
</div>  
{% endblock %}
```

<https://github.com/pahkey/djangobook/tree/3-02>

## 03-2 게시판 페이징 기능 추가하기

<https://github.com/pahkey/djangobook/tree/3-02>

Do it!  
실습

### 페이징 적용하기

01단계

#### 질문 목록 템플릿에 페이징 기능적용하기

- ▶ 위 템플릿에 사용된 주요 페이징 기능을 표로 정리해 보았다.

- ▶ 그리고 페이지 리스트를 보기 좋게 표시하기 위해 부트스트랩의 pagination 컴포넌트를 이용하였다.
- ▶ 템플릿에 사용한 pagination, page-item, page-link 등이 부트스트랩 pagination 컴포넌트의 클래스이다.

페이징 기능	코드
이전 페이지가 있는지 체크	<code>{% if question_list.has_previous %}</code>
이전 페이지 번호	<code>{{ question_list.previous_page_number }}</code>
다음 페이지가 있는지 체크	<code>{% if question_list.has_next %}</code>
다음 페이지 번호	<code>{{ question_list.next_page_number }}</code>
페이지 리스트 루프	<code>{% for page_number in question_list.paginator.page_range %}</code>
현재 페이지와 같은지 체크	<code>{% if page_number == question_list.number %}</code>

## 03-2 게시판 페이징 기능 추가하기

<https://github.com/pahkey/djangobook/tree/3-02>

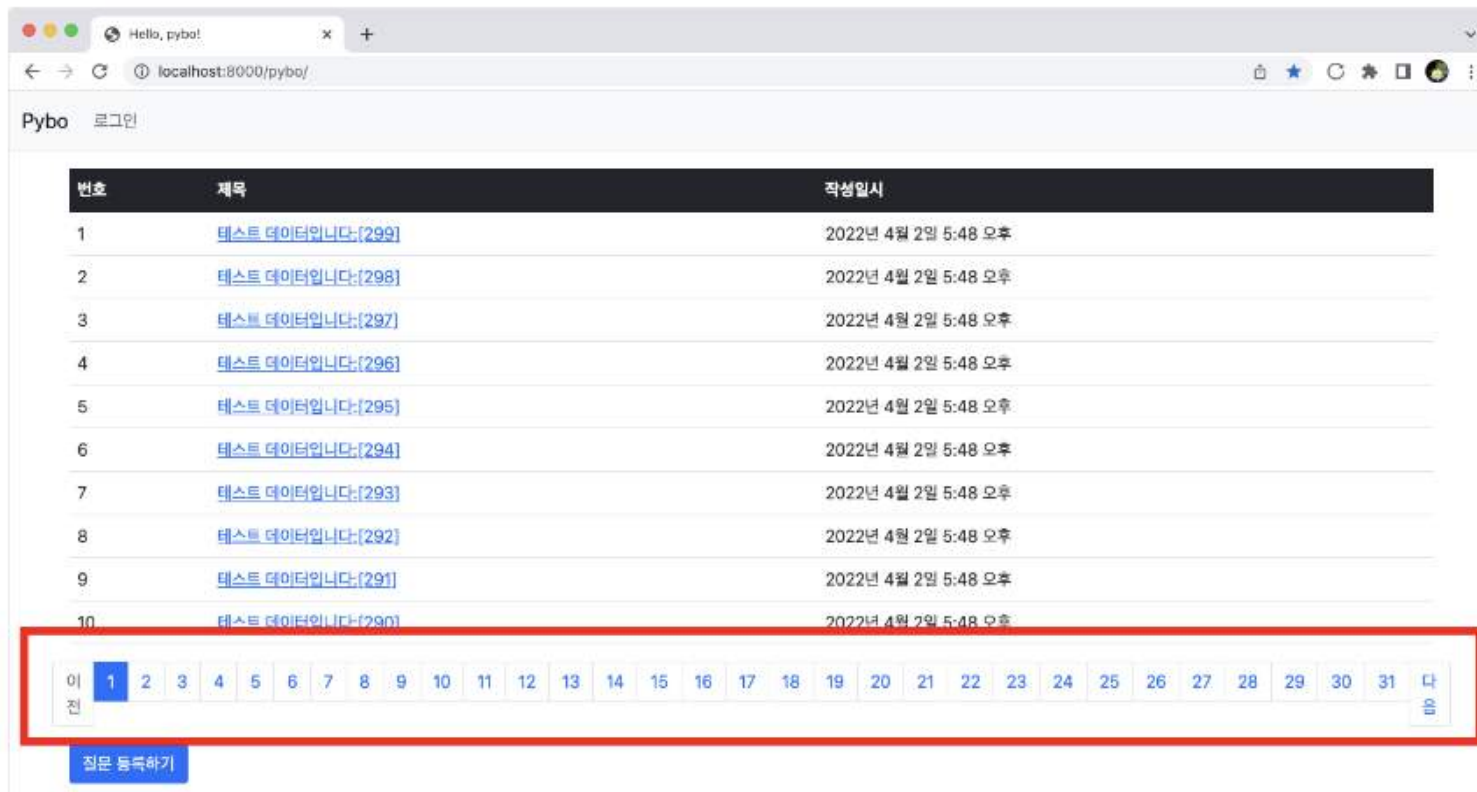
Do it!  
실습

### 페이징 적용하기

#### 01단계 페이지 리스트

- 여기까지 수정하고 질문 목록을 다시 조회해 보자.

- ▶ 페이징 처리는 잘 되었지만 한 가지 문제를 발견할 수 있다.
- ▶ 문제는 위에서 보듯이 이동할 수 있는 페이지가 모두 표시된다는 점이다.



## 03-2 게시판 페이징 기능 추가하기

<https://github.com/pahkey/djangobook/tree/3-02>

Do it!  
실습

### 페이징 적용하기

#### 01단계 페이지 리스트

- ▶ 이 문제를 해결하기 위해 다음과 같이 템플릿을 수정하자.

[파일명: projects\mysite\templates\pybo\question\_list.html]

```
(... 생략 ...)  
<!-- 페이지리스트 -->  
{% for page_number in question_list.paginator.page_range %}  
    {% if page_number >= question_list.number|add:-5 and page_number <= question_list.number|add:5 %}  
    {% if page_number == question_list.number %}  
    <li class="page-item active" aria-current="page">  
        <a class="page-link" href="?page={{ page_number }}">{{ page_number }}</a>  
    </li>  
    {% else %}  
    <li class="page-item">  
        <a class="page-link" href="?page={{ page_number }}">{{ page_number }}</a>  
    </li>  
    {% endif %}  
    {% endif %}  
{% endfor %}  
(... 생략 ...)
```



## 03-2 게시판 페이징 기능 추가하기

<https://github.com/pahkey/djangobook/tree/3-02>

Do it!  
실습

### 페이징 적용하기

#### 01단계 페이지 리스트

- ▶ 다음의 코드를 삽입하여 페이지 표시 제한 기능을 구현했다.

```
{% if page_number >= question_list.number|add:-5 and page_number <= question_list.number|add:5 %}  
(... 생략 ...)  
{% endif %}
```

여기서 사용한 `|add:-5`, `|add:5` 는 템플릿 필터이다. `|add:-5` 는 5만큼 빼라는 의미이고 `|add:5` 는 5만큼 더하라는 의미이다.

- ▶ 위 코드는 페이지 리스트가 현재 페이지 기준으로 좌우 5개씩 보이도록 만든다.
- ▶ 현재 페이지를 의미하는 `question_list.number`보다 5만큼 크거나 작은 값만 표시되도록 만든 것이다.

## 03-2 게시판 페이징 기능 추가하기

<https://github.com/pahkey/djangobook/tree/3-02>

Do it!  
실습

### 페이징 적용하기

#### 01단계 페이지 리스트

- ▶ 만약 현재 페이지가 15라면 다음처럼 페이지 리스트가 표시될 것이다.
- ▶ 축하한다! 페이징 기능이 완성되었다.
- ▶ 페이징은 사실 구현하기 까다로운 기술이다. ▶
- ▶ Paginator 클래스의 도움이 없었다면 아마 이렇게 쉽게 해내기는 힘들었을 것이다.

번호	제목	작성일시
1	<a href="#">테스트 데이터입니다:[159]</a>	2022년 4월 2일 5:48 오후
2	<a href="#">테스트 데이터입니다:[158]</a>	2022년 4월 2일 5:48 오후
3	<a href="#">테스트 데이터입니다:[157]</a>	2022년 4월 2일 5:48 오후
4	<a href="#">테스트 데이터입니다:[156]</a>	2022년 4월 2일 5:48 오후
5	<a href="#">테스트 데이터입니다:[155]</a>	2022년 4월 2일 5:48 오후
6	<a href="#">테스트 데이터입니다:[154]</a>	2022년 4월 2일 5:48 오후
7	<a href="#">테스트 데이터입니다:[153]</a>	2022년 4월 2일 5:48 오후
8	<a href="#">테스트 데이터입니다:[152]</a>	2022년 4월 2일 5:48 오후
9	<a href="#">테스트 데이터입니다:[151]</a>	2022년 4월 2일 5:48 오후
10	<a href="#">테스트 데이터입니다:[150]</a>	2022년 4월 2일 5:48 오후

이전 10 11 12 13 14 15 16 17 18 19 20 다음

질문 등록하기



## 03-3 템플릿 필터 직접 만들어 보기

• 완성 소스 [github.com/pahkey/djangobook/tree/3-03](https://github.com/pahkey/djangobook/tree/3-03)

Do it!  
실습

### 항상 1로 시작하는 게시물 번호 문제 해결하기

#### 01단계 게시물 번호 공식 만들기

게시물 일련번호 공식 만들기

일련번호 = 전체 게시물 개수 - 시작 인덱스 - 현재 인덱스 + 1

- ▶ 질문 게시물의 번호를 역순으로 정렬하려면 다음과 같은 공식을 적용해야 한다.
- ▶ 시작 인덱스 : 페이지당 시작되는 게시물의 시작 번호  
현재 인덱스 : 페이지에 보여지는 게시물 개수만큼 0부터 1씩 증가되는 번호

#### 02단계 템플릿 필터 디렉터리 만들기

템플릿 필터 디렉터리 위치 확인

C:\projects\mysite\pybo\templatetags

- ▶ 템플릿 필터 파일은 pybo/templatetags 디렉터리를 새로 만들어 저장해야 한다.

C:\ 명령 프롬프트

(mysite) C:\projects\mysite\pybo>mkdir templatetags

- ▶ pybo/templatetags 디렉터리는 반드시 앱 디렉터리 아래에 생성해야 한다.

## 03-3 템플릿 필터 직접 만들어 보기

• 완성 소스 [github.com/pahkey/djangobook/tree/3-03](https://github.com/pahkey/djangobook/tree/3-03)

Do it!  
실습

### 항상 1로 시작하는 게시물 번호 문제 해결하기

#### 03단계 템플릿 필터 작성하기

```
파일 이름 C:/projects/mysite/pybo/templatetags/pybo_filter.py

from django import template

register = template.Library()

@register.filter
def sub(value, arg):
    return value - arg
```

- ▶ sub 함수에 `@register.filter`라는 애너테이션을 적용하면 템플릿에서 해당 함수를 필터로 사용할 수 있게 된다.

#### 04단계 템플릿 필터 사용하기

```
템플릿 파일에서 템플릿 필터 파일 로드 예

{% load pybo_filter %}
```

- ▶ 직접 만든 템플릿 필터를 사용하려면 템플릿 파일 맨 위에 `{% load pybo_filter %}`와 같이 템플릿 필터 파일을 로드해야 한다.

## 03-3 템플릿 필터 직접 만들어 보기

• 완성 소스 [github.com/pahkey/djangobook/tree/3-03](https://github.com/pahkey/djangobook/tree/3-03)

Do it!  
실습

항상 1로 시작하는 게시물 번호 문제 해결하기

### 04단계 템플릿 필터 사용하기

파일 이름 C:/projects/mysite/templates/pybo/question\_list.html

```
{% extends 'base.html' %}
{% load pybo_filter %}
(... 생략 ...)
<td>{{ question_list.paginator.count|sub:question_list.start_index|sub:forloop.counter0|add:1 }}</td>
(... 생략 ...)
```

forloop.counter 지우고 입력

공식 요소	설명
question_list.paginator.count	전체 게시물 개수
question_list.start_index	시작 인덱스(1부터 시작)
forloop.counter0	루프 내의 현재 인덱스(forloop.counter0는 0부터, forloop.counter는 1부터 시작)

▶ 게시물 일련번호 공식에 사용된 코드를 정리한 표

일련번호 제대로 표시됨

제대로 표시된 일련번호

## 03-4 질문에 달린 답변 개수 표시하기

• 완성 소스 [github.com/pahkey/djangobook/tree/3-04](https://github.com/pahkey/djangobook/tree/3-04)

Do it!  
실습

### 질문에 달린 답변의 개수 표시하기

```
파일 이름 C:/projects/mysite/templates/pybo/question_list.html

(... 생략 ...)
<td>
  <a href="{% url 'pybo:detail' question.id %}">
    {{ question.subject }}
    {% if question.answer_set.count > 0 %}
    <span class="text-danger small ml-2">
      {{ question.answer_set.count }}
    </span>
    {% endif %}
  </a>
</td>
(... 생략 ...)
```



댓글 개수가 나타남

- ▶ {% if question.answer\_set.count > 0 %}로 답변이 있는 경우를 검사하고, {{ question.answer\_set.count }}로 답변 개수를 표시했다.

## 03-5 로그인 · 로그아웃 구현하기

• 완성 소스 [github.com/pahkey/djangobook/tree/3-05](https://github.com/pahkey/djangobook/tree/3-05)

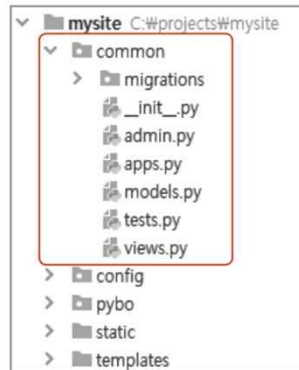
Do it!  
실습

### common 앱 생성 후 초기 설정 작업하기

#### 01단계 common 앱 생성하기

```
C:\> 명령 프롬프트
(mysite) C:\projects\mysite>django-admin startapp common
```

- ▶ projects/mysite 디렉터리 위치에서 다음 명령으로 common 앱을 생성하자.



mysite/common 디렉터리 구조 확인

#### 02단계 설정 파일에 common 앱 등록하기

```
파일 이름 C:/projects/mysite/config/settings.py

(... 생략 ...)
INSTALLED_APPS = [
    'common.apps.CommonConfig',
    (... 생략 ...)
]
(... 생략 ...)
```

```
파일 이름 C:/projects/mysite/config/urls.py

from django.contrib import admin
from django.urls import include, path
from pybo import views

urlpatterns = [
    (... 생략 ...)
    path('common/', include('common.urls')),
]
```

config 디렉터리의 urls.py 파일임!

## 03-5 로그인 · 로그아웃 구현하기

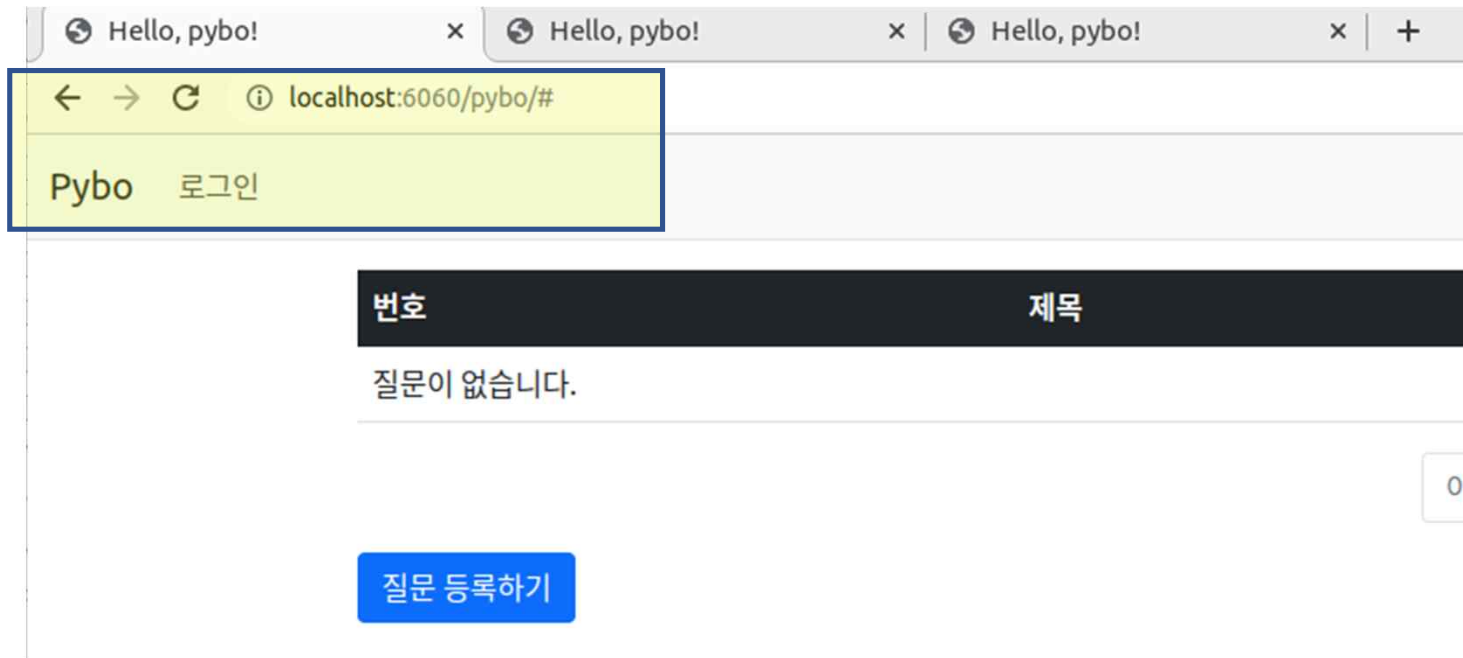
Do it!  
실습

### common 앱 생성 후 초기 설정 작업하기

• 완성 소스 [github.com/pahkey/djangobook/tree/3-05](https://github.com/pahkey/djangobook/tree/3-05)

#### 01단계 3-04 로그인 화면과 비교

- ▶ <https://github.com/pahkey/jump2django/tree/3-04> 로그인 수행화면
- ▶ <http://localhost:6060/pybo/#> 으로 6060 포트 서버 오픈
- ▶ 로그인 버튼을 누르면 # 페이지 (아직 만들지 않은 #페이지로 이동), 공백 페이지로 이동
- ▶ 3-01 챕터에서 로그인 버튼에 # 링크를 추가했다.
- ▶ pybo 내비게이션 바를 만들면서 pybo 버튼 옆에 로그인 버튼도 추가했고 버튼에 대한 링크인 #을 추가했다



## 03-5 로그인 • 로그아웃 구현하기

Do it!  
실습

### common 앱 생성 후 초기 설정 작업하기

• 완성 소스 [github.com/pahkey/djangobook/tree/3-05](https://github.com/pahkey/djangobook/tree/3-05)

#### 01단계 3-05 로그인 화면과 비교

- ▶ <https://github.com/pahkey/jump2django/tree/3-05> 로그인 수행화면
- ▶ <http://localhost:6070/common/login/> 으로 6070 포트 서버 오픈
- ▶ 로그인 버튼을 누르면 /common/login/ url 페이지로 이동하면서 로그인 폼이 보임

Browser tabs: Hello, pybo! x Hello, pybo! x Hello, pybo! x +

Address bar: localhost:6070/common/login/

Page title: Pybo 로그인

Form fields:

- 사용자ID:
- 비밀번호:

Button: 로그인

## 03-5 로그인 • 로그아웃 구현하기

Do it!  
실습

### common 앱 생성 후 초기 설정 작업하기

• 완성 소스 [github.com/pahkey/djangobook/tree/3-05](https://github.com/pahkey/djangobook/tree/3-05)

#### 01단계 3-05 로그인 화면과 비교

- ▶ <https://github.com/pahkey/jump2django/tree/3-06> 로그인 버튼과 회원가입 버튼이 있다.
- ▶ <http://localhost:6080/common/login/> 6080 포트 오픈
- ▶ 회원 가입 버튼을 누르면 `common/signup` url 화면으로 이동함
- ▶ 회원가입을 하기 위한 폼이 생성됨

The screenshot shows a web browser window with three tabs, all titled 'Hello, pybo!'. The active tab displays the URL 'localhost:6080/common/login/'. The page content includes the text 'Pybo' followed by links for '로그인' (Login) and '회원가입' (Sign Up). Below these links are two input fields: '사용자ID' (User ID) and '비밀번호' (Password). At the bottom of the form is a blue button labeled '로그인' (Login).



## 03-5 로그인 • 로그아웃 구현하기

Do it!  
실습

### common 앱 생성 후 초기 설정 작업하기

• 완성 소스 [github.com/pahkey/djangobook/tree/3-05](https://github.com/pahkey/djangobook/tree/3-05)

#### 01단계 3-05 로그인 수행화면 깃허브 코드 비교

- ▶ <https://github.com/pahkey/jump2django/tree/3-04> 에서 36 번 커밋 히스토리에 들어가 05번째 커밋의 내용과 04와 비교한다.

github.com/pahkey/jump2django/tree/3-04

Search or jump to...

Pull requests Issues Codespaces Marketplace

/jump2django Public

Issues Pull requests Actions Projects Security Ir

3-04

34 branches 0 tags

This branch is 36 commits behind main.

## 03-5 로그인 • 로그아웃 구현하기

Do it!  
실습

### common 앱 생성 후 초기 설정 작업하기

• 완성 소스 [github.com/pahkey/djangobook/tree/3-05](https://github.com/pahkey/djangobook/tree/3-05)

#### 01단계 3-05 로그인 수행화면 깃허브 코드 비교

- ▶ <https://github.com/pahkey/jump2django/tree/3-04> 에서 36 번 커밋 히스토리에 들어가 05번째 커밋의 내용과 04와 비교한다.

3-05  
pahkey committed on Apr 1, 2022

3-06  
pahkey committed on Apr 1, 2022

3  
common/admin.py

... -0,0 +1,3

6  
common/apps.py

... -0,0 +1,6

```
1 + from django.contrib import admin
2 +
3 + # Register your models here.
```

```
1 + from django.apps import AppConfig
2 +
3 +
4 + class CommonConfig(AppConfig):
5 +     default_auto_field = 'django.db.models.BigAutoField'
6 +     name = 'common'
```

## 03-5 로그인 • 로그아웃 구현하기

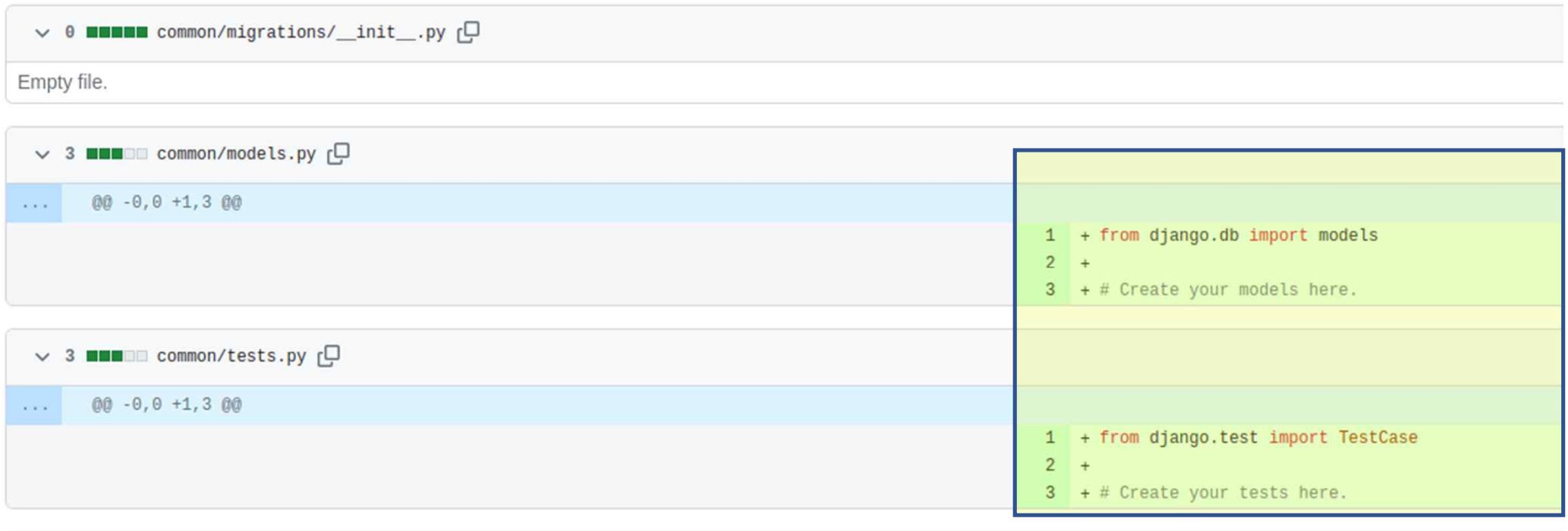
Do it!  
실습

### common 앱 생성 후 초기 설정 작업하기

• 완성 소스 [github.com/pahkey/djangobook/tree/3-05](https://github.com/pahkey/djangobook/tree/3-05)

#### 01단계 3-05 로그인 수행화면 깃허브 코드 비교

- ▶ common 디렉토리가 새로 생성되어 왼편에는 코딩이 없고, 오른편에는 새롭게 생성된 common 디렉토리의 파일들이 보인다.



```
0 common/migrations/__init__.py
Empty file.

3 common/models.py
... @@ -0,0 +1,3 @@

3 common/tests.py
... @@ -0,0 +1,3 @@

1 + from django.db import models
2 +
3 + # Create your models here.

1 + from django.test import TestCase
2 +
3 + # Create your tests here.
```

## 03-5 로그인 • 로그아웃 구현하기

Do it!  
실습

### common 앱 생성 후 초기 설정 작업하기

• 완성 소스 [github.com/pahkey/djangobook/tree/3-05](https://github.com/pahkey/djangobook/tree/3-05)

#### 01단계 3-05 로그인 수행화면 깃허브 코드 비교

- ▶ `path('login/', auth_views.LoginView.as_view(template_name='common/login.html')), name='login')`,
- ▶ `login/` url 이벤트 발생하면, view에서 이벤트핸들러 처리를 진행, logind은 `login/` url의 별칭, 하드 url을 사용하지 않고, 소프트 별칭 url 사용.

9 common/urls.py

... @@ -0,0 +1,9 @@

```
1 + from django.urls import path
2 + from django.contrib.auth import views as auth_views
3 +
4 + app_name = 'common'
5 +
6 + urlpatterns = [
7 +     path('login/', auth_views.LoginView.as_view(template_name='common/login.html'),
8 +         name='login'),
9 +     path('logout/', auth_views.LogoutView.as_view(), name='logout'),
10 + ]
```

## 03-5 로그인 • 로그아웃 구현하기

Do it!  
실습

### common 앱 생성 후 초기 설정 작업하기

• 완성 소스 [github.com/pahkey/djangobook/tree/3-05](https://github.com/pahkey/djangobook/tree/3-05)

#### 01단계 3-05 로그인 수행화면 깃허브 코드 비교

- ▶ # 로그인 성공후 이동하는 URL : LOGIN\_REDIRECT\_URL = '/'
- ▶ # 로그아웃시 이동하는 URL : LOGOUT\_REDIRECT\_URL = '/'

```
config/settings.py

31 # Application definition
32
33 INSTALLED_APPS = [
34     'pybo.apps.PyboConfig',
35     'django.contrib.admin',
36     'django.contrib.auth',
37
38     ...
39
125 # https://docs.djangoproject.com/en/4.0/ref/settings/#default-auto-field
126
127 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

31 # Application definition
32
33 INSTALLED_APPS = [
34     + 'common.apps.CommonConfig',
35     'pybo.apps.PyboConfig',
36     'django.contrib.admin',
37     'django.contrib.auth',
38
39     ...
40
126 # https://docs.djangoproject.com/en/4.0/ref/settings/#default-auto-field
127
128 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
129 +
130 + # 로그인 성공후 이동하는 URL
131 + LOGIN_REDIRECT_URL = '/'
132 +
133 + # 로그아웃시 이동하는 URL
134 + LOGOUT_REDIRECT_URL = '/'
```

## 03-5 로그인 • 로그아웃 구현하기

Do it!  
실습

### common 앱 생성 후 초기 설정 작업하기

• 완성 소스 [github.com/pahkey/djangobook/tree/3-05](https://github.com/pahkey/djangobook/tree/3-05)

#### 01단계 3-05 로그인 수행화면 깃허브 코드 비교

- ▶ urlpatterns = [path('common/', include('common.urls')), 포함

3 config/urls.py

```
... 00 -1,7 +1,10 00
1  from django.contrib import admin
2  from django.urls import path, include
3
4  urlpatterns = [
5      path('admin/', admin.site.urls),
6      path('pybo/', include('pybo.urls')),
7  ]
```

```
1  from django.contrib import admin
2  from django.urls import path, include
3  + from pybo import views
4
5  urlpatterns = [
6      path('admin/', admin.site.urls),
7      path('pybo/', include('pybo.urls')),
8  + path('common/', include('common.urls')),
9  + path('', views.index, name='index'), # '/' 에 해당되는 path
10 ]
```

## 03-5 로그인 • 로그아웃 구현하기

Do it!  
실습

### common 앱 생성 후 초기 설정 작업하기

• 완성 소스 [github.com/pahkey/djangobook/tree/3-05](https://github.com/pahkey/djangobook/tree/3-05)

#### 01단계 3-05 로그인 수행화면 깃허브 코드 비교

- ▶ login.html 내부에 form\_errors.html"이 있음을 확인

common/login.html

```
1 + {% extends "base.html" %}
2 + {% block content %}
3 + <div class="container my-3">
4 +   <form method="post" action="{% url 'common:login' %}">
5 +     {% csrf_token %}
6 +     {% include "form_errors.html" %}
7 +     <div class="mb-3">
8 +       <label for="username">사용자ID</label>
9 +       <input type="text" class="form-control" name="username" id="username"
10 +         value="{% form.username.value|default_if_none:'' %}">
11 +     </div>
12 +     <div class="mb-3">
13 +       <label for="password">비밀번호</label>
14 +       <input type="password" class="form-control" name="password" id="password"
15 +         value="{% form.password.value|default_if_none:'' %}">
16 +     </div>
17 +     <button type="submit" class="btn btn-primary">로그인</button>
18 +   </form>
19 + </div>
20 + {% endblock %}
```

## 03-5 로그인 • 로그아웃 구현하기

Do it!  
실습

### common 앱 생성 후 초기 설정 작업하기

• 완성 소스 [github.com/pahkey/djangobook/tree/3-05](https://github.com/pahkey/djangobook/tree/3-05)

#### 01단계 3-05 로그인 수행화면 깃허브 코드 비교

▶ form\_errors.html 정의

/form\_errors.html 

```
1 + {% if form.errors %}
2 + <div class="alert alert-danger">
3 +     {% for field in form %}
4 +     {% for error in field.errors %} <!-- 필드 오류를 출력한다. -->
5 +     <div>
6 +         <strong>{{ field.label }}</strong>
7 +         {{ error }}
8 +     </div>
9 +     {% endfor %}
10 + {% endfor %}
11 + {% for error in form.non_field_errors %} <!-- 년필드 오류를 출력한다.
12 + <div>
13 +     <strong>{{ error }}</strong>
14 + </div>
15 + {% endfor %}
16 + </div>
17 + {% endif %}
```



## 03-5 로그인 • 로그아웃 구현하기

Do it!  
실습

### common 앱 생성 후 초기 설정 작업하기

• 완성 소스 [github.com/pahkey/djangobook/tree/3-05](https://github.com/pahkey/djangobook/tree/3-05)

#### 01단계 3-05 로그인 수행화면 깃허브 코드 비교

- ▶ 내비게이션 바 수정

templates/navbar.html

1,7 +12,11 @@

```
<div class="collapse navbar-collapse" id="navbarSupportedContent">
  <ul class="navbar-nav me-auto mb-2 mb-lg-0">
    <li class="nav-item">
      <a class="nav-link" href="#">로그인</a>

    </li>
  </ul>
</div>
```

```
12      <div class="collapse navbar-collapse" id="navbarSupportedContent">
13        <ul class="navbar-nav me-auto mb-2 mb-lg-0">
14          <li class="nav-item">
15            +      {% if user.is_authenticated %}
16            +      <a class="nav-link" href="{% url 'common:logout' %}">{{ user.username }} (로그아웃)</a>
17            +      {% else %}
18            +      <a class="nav-link" href="{% url 'common:login' %}">로그인</a>
19            +      {% endif %}
20          </li>
21        </ul>
22      </div>
```

## 03-5 로그인 • 로그아웃 구현하기

• 완성 소스 [github.com/pahkey/djangobook/tree/3-05](https://github.com/pahkey/djangobook/tree/3-05)

Do it!  
실습

### common 앱 생성 후 초기 설정 작업하기

#### 03단계 common/urls.py 생성하기

파일 이름 C:/projects/mysite/common/urls.py

```
app_name = 'common'

urlpatterns = [
]
```

- ▶ common/urls.py 파일을 새로 생성하고 다음과 같이 작성하자.
- ▶ 아직 common 앱에 어떤 기능도 구현하지 않았으므로 urlpatterns는 빈 상태로 놔두자.

## 03-5 로그인 • 로그아웃 구현하기

• 완성 소스 [github.com/pahkey/djangobook/tree/3-05](https://github.com/pahkey/djangobook/tree/3-05)

Do it!  
실습

### 로그인 구현하기

#### 01단계 네비게이션바 수정하고 URL 매핑 추가하기

파일 이름 C:/projects/mysite/templates/navbar.html

```
(... 생략 ...)  
<a class="nav-link" href="{% url 'common:login' %}">로그인</a>  
(... 생략 ...)
```

- ▶ 템플릿 태그로 {% url 'common:login' %}를 사용했으므로 common/urls.py 파일에 URL 매핑을 추가하자.

파일 이름 C:/projects/mysite/common/urls.py

```
from django.urls import path  
from django.contrib.auth import views as auth_views  
  
app_name = 'common'  
  
urlpatterns = [  
    path('login/', auth_views.LoginView.as_view(), name='login'),  
]
```

django.contrib.auth 앱의 LoginView 클래스를 활용  
했으므로 별도의 views.py 파일 수정이 필요 없음!

- ▶ 로그인 기능은 django.contrib.auth 앱을 사용할 것이므로 common/views.py 파일은 수정할 필요가 없다.

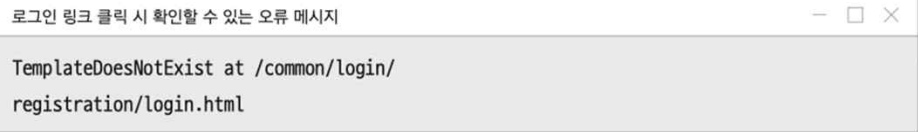
## 03-5 로그인 • 로그아웃 구현하기

• 완성 소스 [github.com/pahkey/djangobook/tree/3-05](https://github.com/pahkey/djangobook/tree/3-05)

Do it!  
실습

### 로그인 구현하기

#### 02단계 로그인 템플릿 만들기



로그인 링크 클릭 시 확인할 수 있는 오류 메시지

TemplateDoesNotExist at /common/login/  
registration/login.html

- ▶ 위 오류는 registration 디렉터리에 login.html 파일이 없음을 의미한다.
- ▶ 이 오류를 해결하려면 registration/login.html 템플릿 파일을 작성해야만 한다.



```
(... 생략 ...)  
urlpatterns = [  
    path('login/', auth_views.LoginView.as_view(  
        template_name='common/login.html'), name='login'),  
]
```

- ▶ as\_view함수에 template\_name으로 'common/login.html'을 설정하면 registration 디렉터리가 아닌 common 디렉터리에서 login.html 파일을 참조하게 된다.

## 03-5 로그인 • 로그아웃 구현하기

• 완성 소스 [github.com/pahkey/djangobook/tree/3-05](https://github.com/pahkey/djangobook/tree/3-05)

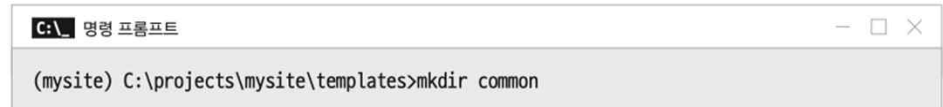
Do it!  
실습

### 로그인 구현하기



- ▶ common 디렉터리에 login.html 파일이 없어서 오류가 발생하였다.

### 03단계 common 디렉터리 생성 후 login.html 생성하기



- ▶ common 디렉터를 생성하고 login.html 파일을 생성한 후 다음처럼 코드를 작성하자.

## 03-5 로그인 • 로그아웃 구현하기

• 완성 소스 [github.com/pahkey/djangobook/tree/3-05](https://github.com/pahkey/djangobook/tree/3-05)

Do it!  
실습

### 로그인 구현하기

파일 이름 C:/projects/mysite/templates/common/login.html

```
{% extends "base.html" %}
{% block content %}
<div class="container my-3">
  <form method="post" class="post-form" action="{% url 'common:login' %}">
    {% csrf_token %}
    {% include "form_errors.html" %}
    <div class="form-group">
      <label for="username">사용자ID</label>
      <input type="text" class="form-control" name="username" id="username"
        value="{% form.username.value|default_if_none:'' %}">
    </div>
    <div class="form-group">
      <label for="password">비밀번호</label>
      <input type="password" class="form-control" name="password" id="password"
        value="{% form.password.value|default_if_none:'' %}">
    </div>
  </div>
</div>
```

```
    <button type="submit" class="btn btn-primary">로그인</button>
  </form>
</div>
{% endblock %}
```

- ▶ 입력 항목 username과 password는 모두 `django.contrib.auth` 앱에서 요구하는 필수 항목이다.

## 03-5 로그인 • 로그아웃 구현하기

• 완성 소스 [github.com/pahkey/djangobook/tree/3-05](https://github.com/pahkey/djangobook/tree/3-05)

Do it!  
실습

### 로그인 구현하기

#### 04단계 form\_error.html 만들어 작성하기

파일 이름 C:/projects/mysite/templates/form\_errors.html

```
{% if form.errors %}
    {% for field in form %}
        {% for error in field.errors %} <!-- 필드 오류를 출력한다. -->
            <div class="alert alert-danger">
                <strong>{{ field.label }}</strong>
                {{ error }}
            </div>
        {% endfor %}
    {% endfor %}
    {% for error in form.non_field_errors %} <!-- 년필드 오류를 출력한다. -->
        <div class="alert alert-danger">
            <strong>{{ error }}</strong>
        </div>
    {% endfor %}
{% endif %}
```

디렉터리 확인!!

폼 오류 2가지

- 필드 오류(입력값이 누락되었거나 형식에 맞지 않음)
- 년필드 오류(입력값과 관계없이 발생한 오류)

- ▶ form\_errors.html 파일은 로그인 실패 시 로그인이 실패한 원인을 알려 준다.

## 03-5 로그인 • 로그아웃 구현하기

• 완성 소스 [github.com/pahkey/djangobook/tree/3-05](https://github.com/pahkey/djangobook/tree/3-05)

Do it!  
실습

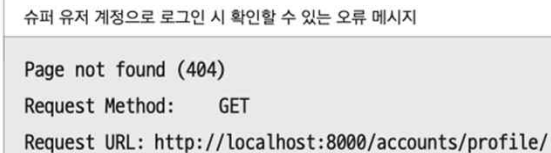
### 로그인 구현하기

#### 05단계 엉뚱한 값으로 로그인해 보기



- ▶ 입력값을 누락하거나 엉뚱한 값을 입력한 다음 <로그인>을 누르면 적절한 오류 메시지가 표시될 것이다.

#### 06단계 제대로 로그인해 보기



슈퍼 유저 계정으로 로그인 시 확인할 수 있는 오류 메시지

Page not found (404)  
Request Method: GET  
Request URL: http://localhost:8000/accounts/profile/

- ▶ 오류가 발생한 이유는 `django.contrib.auth` 앱이 로그인 성공 후 `/accounts/profile/`이라는 URL로 페이지를 리다이렉트 했기 때문인데, 아직 이 URL에 대한 준비를 하지 않았다.



## 03-5 로그인 • 로그아웃 구현하기

• 완성 소스 [github.com/pahkey/djangobook/tree/3-05](https://github.com/pahkey/djangobook/tree/3-05)

Do it!  
실습

### 로그인 구현하기

#### 07단계 로그인 성공 시 이동할 페이지 등록하기

파일 이름 C:/projects/mysite/templates/form\_errors.html

```
{% if form.errors %}
    {% for field in form %}
        {% for error in field.errors %} <!-- 필드 오류를 출력한다. -->
            <div class="alert alert-danger">
                <strong>{{ field.label }}</strong>
                {{ error }}
            </div>
        {% endfor %}
    {% endfor %}
    {% for error in form.non_field_errors %} <!-- 년필드 오류를 출력한다. -->
        <div class="alert alert-danger">
            <strong>{{ error }}</strong>
        </div>
    {% endfor %}
{% endif %}
```

디렉터리 확인!

파일 이름 C:/projects/mysite/config/settings.py

```
(... 생략 ...)
# 로그인 성공 시 자동으로 이동할 URL
LOGIN_REDIRECT_URL = '/'
```

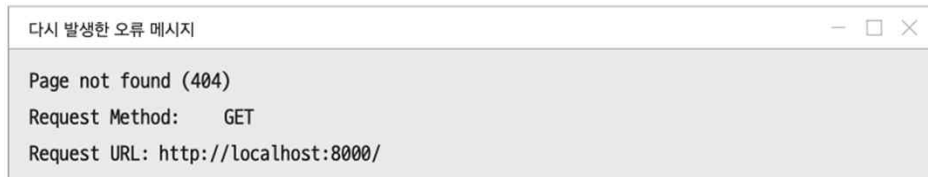
- ▶ 로그인 성공 시 / 페이지로 이동할 수 있도록 config/settings.py 파일을 수정하자.

## 03-5 로그인 • 로그아웃 구현하기

• 완성 소스 [github.com/pahkey/djangobook/tree/3-05](https://github.com/pahkey/djangobook/tree/3-05)

Do it!  
실습

### 로그인 구현하기



/ 페이지에 대한 매핑이 없어 발생한 오류

- ▶ 이 오류 메시지는 / 페이지에 대한 매핑이 없음을 의미한다.

### 08단계 config/urls.py 수정하기



- ▶ / 페이지 요청에 대해 `path('', views.index, name='index')`가 작동하여 `pybo/views.py` 파일의 `index` 함수가 실행된다.

## 03-5 로그인 • 로그아웃 구현하기

• 완성 소스 [github.com/pahkey/djangobook/tree/3-05](https://github.com/pahkey/djangobook/tree/3-05)

Do it!  
실습

### 로그아웃 구현하기

#### 01단계 내비게이션바 수정하기

파일 이름 C:/projects/mysite/templates/navbar.html

```
(... 생략 ...)
<li class="nav-item">
  {% if user.is_authenticated %}
  <a class="nav-link" href="{% url 'common:logout' %}">
    {{ user.username }} (로그아웃)
  </a>
  {% else %}
  <a class="nav-link" href="{% url 'common:login' %}">로그인</a>
  {% endif %}
</li>
(... 생략 ...)
```

- ▶ navbar.html 템플릿 파일에서 로그인 링크 부분을 다음과 같이 수정하자.

#### 02단계 로그아웃 URL 매핑하기

파일 이름 C:/projects/mysite/common/urls.py

```
urlpatterns = [
    path('login/', auth_views.LoginView.as_view(
        template_name='common/login.html'), name='login'),
    path('logout/', auth_views.LogoutView.as_view(), name='logout'),
]
```

- ▶ 로그아웃 링크가 추가 되었으므로 {% url 'common:logout' %}에 대응하는 URL 매핑을 common/urls.py파일에 추가하자.

## 03-5 로그인 • 로그아웃 구현하기

• 완성 소스 [github.com/pahkey/djangobook/tree/3-05](https://github.com/pahkey/djangobook/tree/3-05)

Do it!  
실습

### 로그아웃 구현하기

#### 03단계 로그아웃 성공 시 이동할 페이지 등록하기

파일 이름 C:/projects/mysite/config/settings.py

```
(... 생략 ...)  
# 로그인 로그아웃 성공 시 이동할 URL  
LOGIN_REDIRECT_URL = '/'  
LOGOUT_REDIRECT_URL = '/'
```

- ▶ 로그아웃 성공 시 / 페이지로 이동하기 위한 LOGOUT\_REDIRECT\_URL = '/'을 설정했다.

**Thank You**