

2024

응용소프트웨어개발 11주차 템플릿 적용하기

JEONBUK NATIONAL UNIVERSITY



전북대학교
JEONBUK NATIONAL UNIVERSITY

Startbootstrap 템플릿 이용하기

- 최신버전의 bootstrap를 쓰고 있다면, 아래 링크를 이용하세요.
 - <https://github.com/StartBootstrap/startbootstrap-blog-home>
 - <https://github.com/StartBootstrap/startbootstrap-blog-post>
 - 두 프로젝트 모두 dist 폴더만 참고하면 됩니다.
- 교재를 따라했다면, 아래 링크를 이용하세요.
 - startbootstrap-blog-home: [이 링크](#)
 - startbootstrap-blog-post: [이 링크](#)

Startbootstrap 템플릿 이용하기 (blog home)

01단계 Blog Home 예제의 카드 구성 살펴보기

부트스트랩에서 제공하는 카드 기능을 사용하면 지정한 이미지, 헤더, 푸터 등을 카드 형태로 구성할 수 있습니다. Blog Home 예제에서는 포스트와 검색 창, 카테고리 목록을 카드에 담아 화면을 구성하고 있습니다.

앞으로 블로그 게시물을 '포스트'라고 부르겠습니다.

카드의 자세한 사용법은 부트스트랩 공식 웹 사이트에서 [Documentation > Components > Card] 문서를 참고하세요.

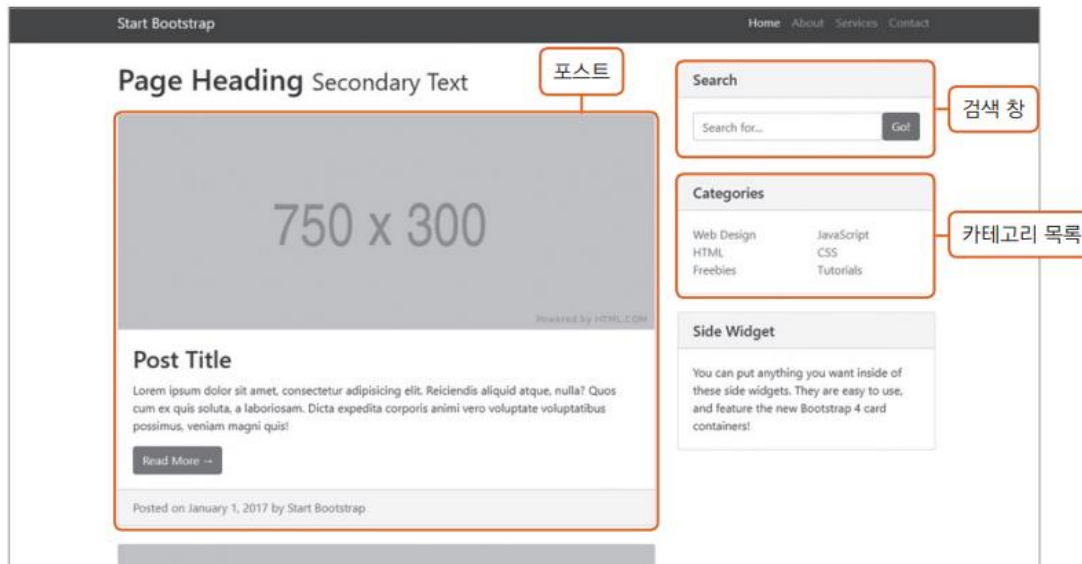


그림 4-33 포스트, 검색 창, 카테고리 목록을 카드에 담아 구성하는 Blog Home 템플릿

Startbootstrap 템플릿 사용하기 (blog home)

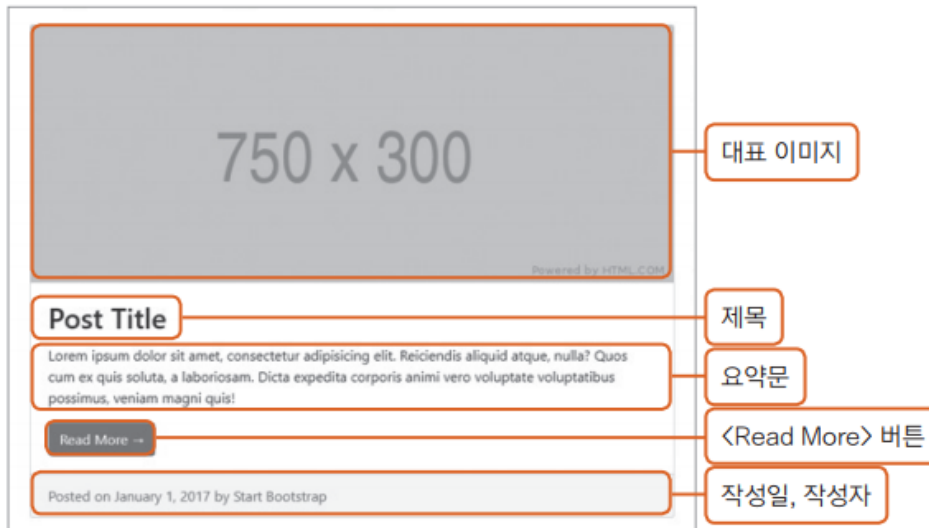


그림 4-34 포스트 카드의 구성

```

62
63 <!-- Blog Post -->
64 <div class="card mb-4">
65   
66   <div class="card-body">
67     <h2 class="card-title">Post Title</h2>
68     <p class="card-text">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Reiciendis aliquida atque, nulla? Quos cum ex quis soluta, a laboriosam. Dicta expedita corporis animi vero voluptate voluptatibus possimus, veniam magni quis!</p>
69     <a href="#" class="btn btn-primary">Read More &rarr;</a>
70   </div>
71   <div class="card-footer text-muted">
72     Posted on January 1, 2020 by
73     <a href="#">Start Bootstrap</a>
74   </div>
75 </div>
76

```

그림 4-35 포스트 카드의 HTML

Startbootstrap 템플릿 이용하기 (blog home)



placeholder.com을 사용해 임시 이미지 넣기

포스트 카드의 HTML을 보면 태그에 삽입할 이미지 경로로 'http://placeholder.it/750x300'이 입력되어 있습니다. 이 값은 웹 개발을 할 때 유용하게 사용할 수 있는 placeholder.com이라는 서비스에서 제공하는 주소입니다. 원하는 폭과 높이를 지정하면 그 크기에 맞는 그림을 보내줍니다.

이 서비스는 사진을 넣을 때 어떤 모습일지 먼저 확인하고 싶을 때 사용하면 좋습니다. 이 부분을 blog_list.html의 왼쪽 부분에 반복해서 붙여주면 똑같은 모양을 만들 수 있겠죠.

Startbootstrap 템플릿 사용하기 (blog home)

03단계 검색 창 카드와 카테고리 카드 살펴보기

이번에는 Blog Home 예제의 오른쪽에 있는 검색 창 카드와 카테고리 카드를 살펴봅시다. 아까 포스트를 구성하는 카드와 달리 제목(header)이 있네요. 카드 내용과 구분할 수 있게 음영 처리가 되어 있습니다.

먼저 검색 창 카드의 소스 코드를 살펴볼까요? 오른쪽 카드와 비슷하지만 조금 다릅니다. 먼저 `<h5>` 태그 안에 `class`로 `card-header`를 지정했네요. 검색 창은 `class`가 `card-body`인 `<div>` 태그 안에 들어 있습니다.

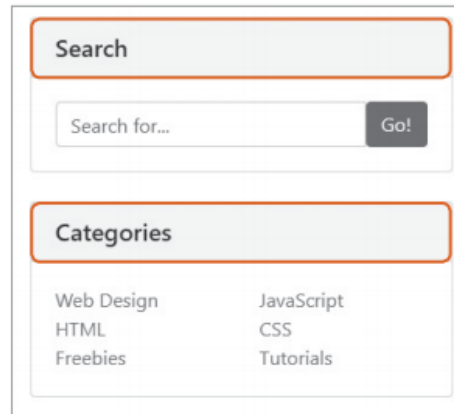


그림 4-36 검색 창 카드와 카테고리 카드

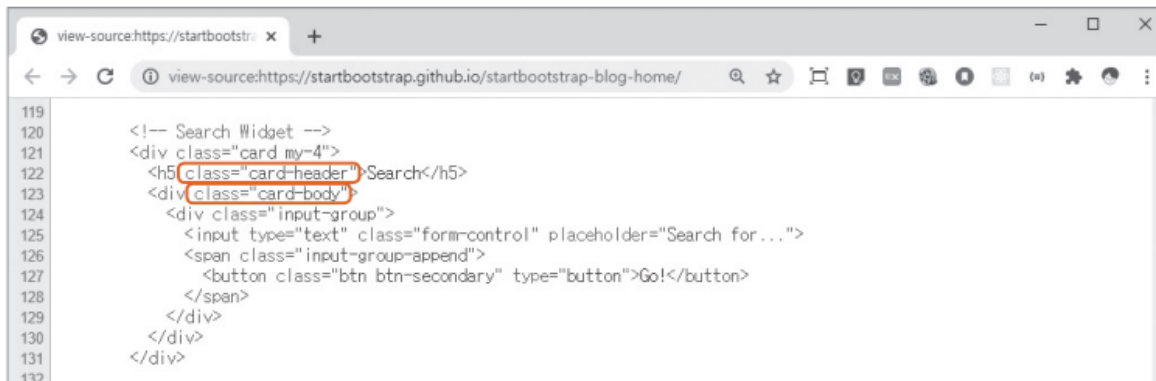


그림 4-37 검색 창 카드의 HTML

Startbootstrap 템플릿 사용하기 (blog home)

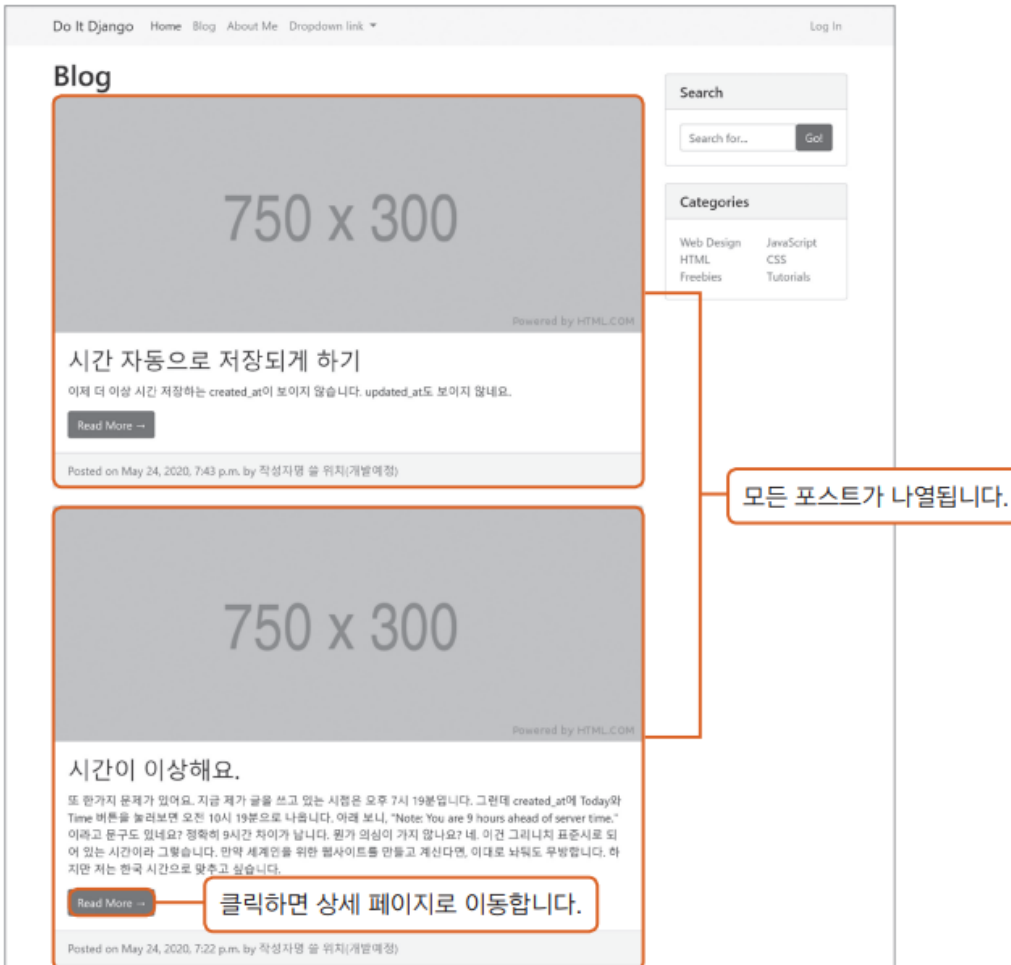


그림 9-5 실제 포스트 내용 표시

Startbootstrap 템플릿 사용하기 (blog post)

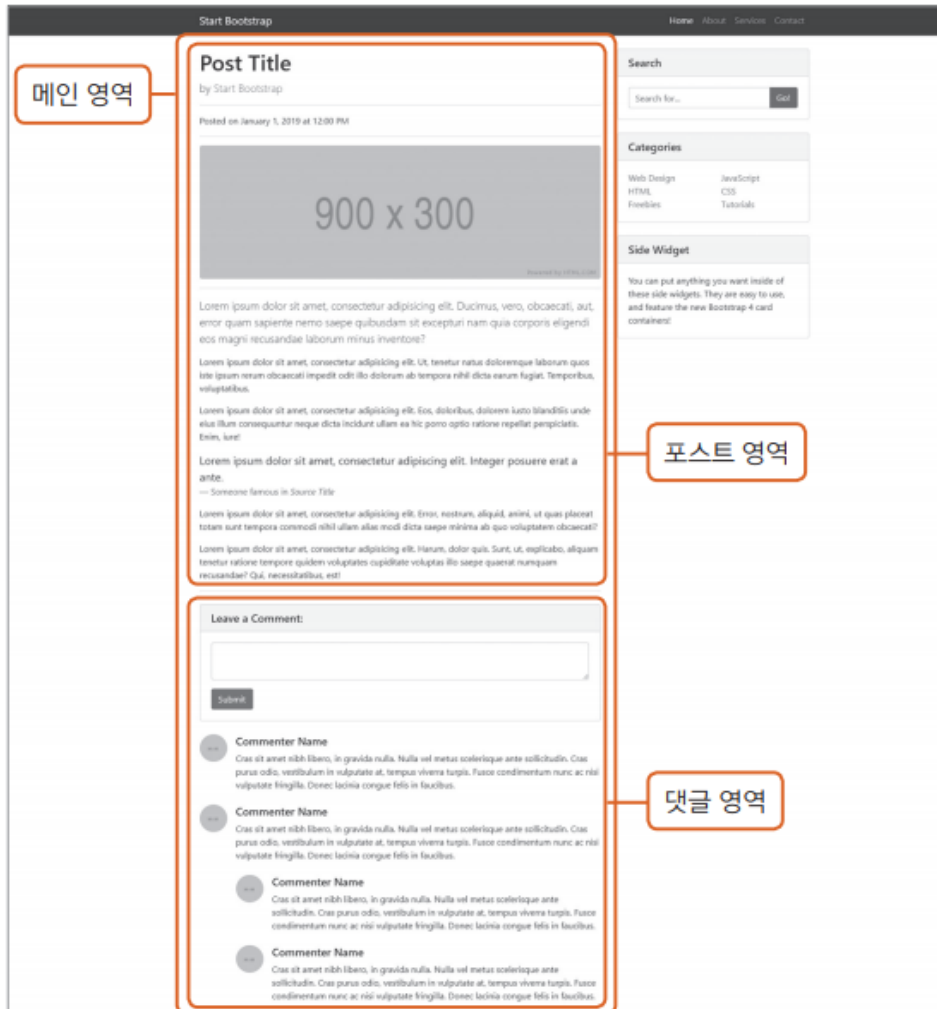


그림 9-8 포스트 상세 페이지의 구조

Startbootstrap 템플릿 사용하기 (blog post)

실습 파일: blog/templates/blog/post_detail.html

```
<!DOCTYPE html>
{% load static %}
<html lang="ko">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="description" content="">
    <meta name="author" content="">

    <title>{{ post.title }} - Blog</title>

    <!-- Bootstrap core CSS -->
    <link rel="stylesheet" href="{% static 'blog/bootstrap/bootstrap.min.css' %}"
media="screen">
```

```
<!-- Custom styles for this template -->
<link href="css/blog-post.css" rel="stylesheet">
</head>

(...생략...)

<!-- Bootstrap core JavaScript -->
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
    integrity="sha384-Dfxdz2htPH0lsSSs5nCTpuj/zy4C+0GpamoFVy38MVBnE+IbbVYUew
+0rCXaRkfj"
    crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.
min.js"
    integrity="sha384-9/reFTGAW83EW2RDu2S0VKAizap3H66lZ81PoYlFhbGU+6BZp6G7n
iu735Sk7lN"
    crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.
min.js"
    integrity="sha384-w1Q4orYjBQndcko6MimVbzY0tgp4pWB41Z7lr30Wkz0vr/aWKhXdBN
mNb5D92v7s"
    crossorigin="anonymous"></script>

</body>
</html>
```

Startbootstrap 템플릿 사용하기 (blog post)

03단계 내비게이션 바에 페이지 윗부분이 가려지는 문제 해결하기

그런데 포스트 제목이 내비게이션 바에 가려서 보이지 않습니다. 아래로 페이지를 내려도 내비게이션 바가 계속 웹 브라우저 상단에 고정되어 페이지 윗부분을 가립니다.



실습 파일: blog/templates/blog/post_list.html

(...생략...)

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
```

(...생략...)

실습 파일: blog/templates/blog/post_detail.html

(...생략...)

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">
```

(...생략...)

Startbootstrap 템플릿 이용하기 (blog post)

blog-post.css는 어떤 역할을 할까요? 파일 내용을 살펴보니 <body> 태그의 padding-top을 56px로 설정하네요. 페이지의 윗부분이 내비게이션 바에 가려지므로 그 가려지는 크기만큼 body 요소에 패딩을 주는 간단한 해결책입니다.

실습 파일: blog/static/blog/css/blog-post.css

```
body {
  padding-top: 56px;
}
```

이 css 파일을 사용하기 위해 blog/templates/blog/post_detail.html 파일을 다음과 같이 수정해 주세요. 광고에서 정적 파일을 어떻게 사용하는지 이제 조금씩 감이 잡히나요?

실습 파일: blog/templates/blog/post_detail.html

```
(...생략...)
<!-- Bootstrap core CSS -->
<link rel="stylesheet" href="{% static 'blog/bootstrap/bootstrap.min.css' %}" media=
"screen">

<!-- Custom styles for this template -->
<link rel="stylesheet" href="{% static 'blog/css/blog-post.css' %}" media="screen">
(...생략...)
```

블로그 다듬기 (글 미리보기)

01단계 출력 글자 수 제한하기

다음처럼 p.content 뒤에 | truncatewords:45를 입력하여 앞에서부터 45개 단어만 출력하도록 설정합니다. 글자 수로 나누고 싶은 분들은 truncatechars를 입력하면 됩니다.

실습 파일: blog/templates/blog/post_list.html

```
(...생략...)
<div class="card-body">
  <h2 class="card-title">{{ p.title}}</h2>
  <p class="card-text">{{ p.content | truncatewords:45 }}</p>
  <a href="{{ p.get_absolute_url }}" class="btn btn-primary">Read More &rarr;</a>
</div>
<div class="card-footer text-muted">
  Posted on {{ p.created_at}} by
  <a href="#">작성자명 쓸 위치(개발예정)</a>
(...생략...)
```

이제 웹 브라우저로 127.0.0.1:8000/blog/에 가봅시다. 다음과 같이 포스트의 내용이 45개의 단어가 넘는 경우에는 앞 45개 단어만 표시하고 뒤에는 말줄임표(...)로 처리된 것을 볼 수 있습니다.

🔥 출바꿈이 반영되지 않아 읽기가 어려운 문제는 나중에 해결하겠습니다.



그림 10-12 truncatewords 필드를 사용해 포스트 목록 페이지에서는 본문 중 앞 45개의 단어만 출력

블로그 다듬기 (글 요약문 표시)

02단계 요약문 필드 만들기

여기에서 한 발 더 나아가 포스트의 요약문을 보여주는 `hook_text`라는 새 필드를 만들고자 합니다. 허핑턴포스트 같은 뉴스 웹 사이트가 기사 제목 아래 사람들의 관심을 끄는 메시지를 보여주는 방식과 비슷하게 말이죠.

`models.py`를 열어 `hook_text` 필드가 비어 있지 않을 때는 `hook_text` 필드 값을 보여주도록 하겠습니다. `CharField`를 사용해 `hook_text`의 글자 수는 100자까지만 작성할 수 있는 것으로 제한을 두었습니다.

👤 필드명을 `abstract`(요약하다)로 하지 않은 이유는 프로그래밍의 `abstract`라는 개념과 헷갈릴 수 있기 때문입니다.

실습 파일: `blog/models.py`

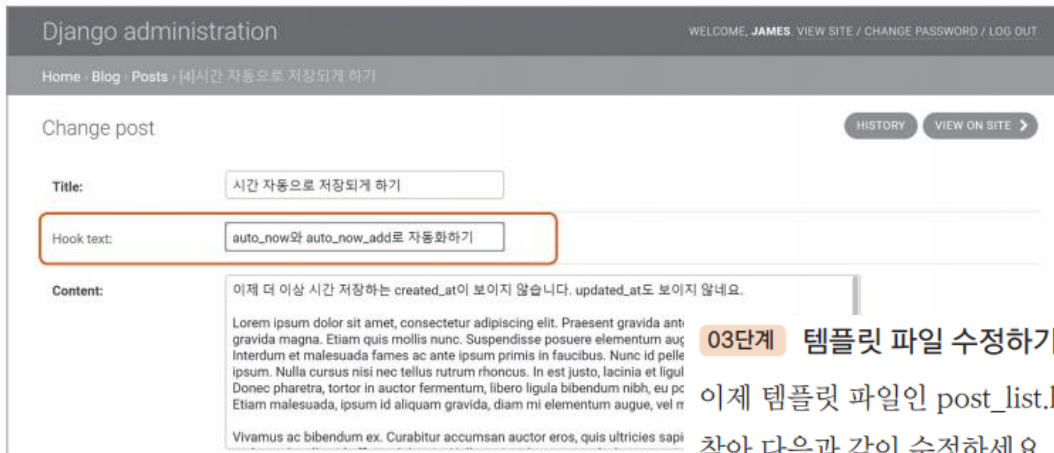
```
(...생략...)
class Post(models.Model):
    title = models.CharField(max_length=30)
    hook_text = models.CharField(max_length=100, blank=True)
    content = models.TextField()

    head_image = models.ImageField(upload_to='blog/images/%Y/%m/%d/', blank=True)
```

Cmdr

```
C:\github\do_it_django_a_to_z (main -> origin)
(venv) λ python manage.py makemigrations
C:\github\do_it_django_a_to_z (main -> origin)
(venv) λ python manage.py migrate
```

블로그 다듬기 (글 요약문 표시)



Django administration

WELCOME, JAMES. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home - Blog - Posts - [시간 자동으로 저장되게 하기]

Change post

HISTORY VIEW ON SITE

Title: 시간 자동으로 저장되게 하기

Hook text: auto_now와 auto_now_add로 자동화하기

Content: 이제 더 이상 시간 저장하는 created_at이 보이지 않습니다. updated_at도 보이지 않네요.

03단계 템플릿 파일 수정하기

그림 10-13 추가된 Hook text 입력란에 요약문 입력

이제 템플릿 파일인 `post_list.html`을 수정할 차례입니다. `class="card-body"`인 `div` 요소를 찾아 다음과 같이 수정하세요. 이때 태그 안에서 `class="text-muted"`를 추가하면 글씨가 회색으로 나타납니다. 부트스트랩에서 미리 `class`의 스타일을 지정해 뒀기 때문입니다.

실습 파일: `blog/templates/blog/post_list.html`

```
(...생략...)
<div class="card-body">
  <h2 class="card-title">{{ p.title }}</h2>
  {% if p.hook_text %}
    <h5 class="text-muted">{{ p.hook_text }}</h5>
  {% endif %}
  <p class="card-text">{{ p.content | truncatewords:45 }}</p>
  <a href="{{ p.get_absolute_url }}" class="btn btn-primary">Read More &rarr;</a>
</div>
(...생략...)
```

블로그 다듬기 (글 요약문 표시)

04단계 포스트 상세 페이지 수정하기

포스트 상세 페이지도 수정합니다. `<h1>{{ post.title }}` 바로 아래에 포스트 목록 페이지와 마찬가지로 작성합니다. 한 가지 차이가 있다면 포스트 목록 페이지에서는 `p.hook_text`였는데, 여기서는 `post.hook_text`로 작성한다는 점 정도뿐입니다.

실습 파일: `blog/templates/blog/post_list.html`

```
(...생략...)
<!-- Page Content -->
<div class="container">

  <div class="row">

    <!-- Post Content Column -->
    <div class="col-lg-8">

      <!-- Title -->
      <h1 class="mt-4">{{ post.title }}</h1>
      <h5 class="text-muted">{{ post.hook_text }}</h5>

      <!-- Author -->
      <p class="lead">
        by
        <a href="#">작성자명 쓸 위치(개발예정)</a>
      </p>

      <hr>
    </div>
  </div>
</div>
(...생략...)
```

메인영역 모듈화하기

왜 템플릿 요소를 모듈화할까?

다음 그림의 왼쪽은 포스트 목록 페이지를 캡처한 모습이고, 오른쪽은 포스트 상세 페이지를 캡처한 모습입니다. 맨 위에는 내비게이션 바가 보이고 그 아래 왼쪽에는 메인 영역, 오른쪽에는 사이드 영역이 있습니다. 맨 밑에는 푸터가 있죠.

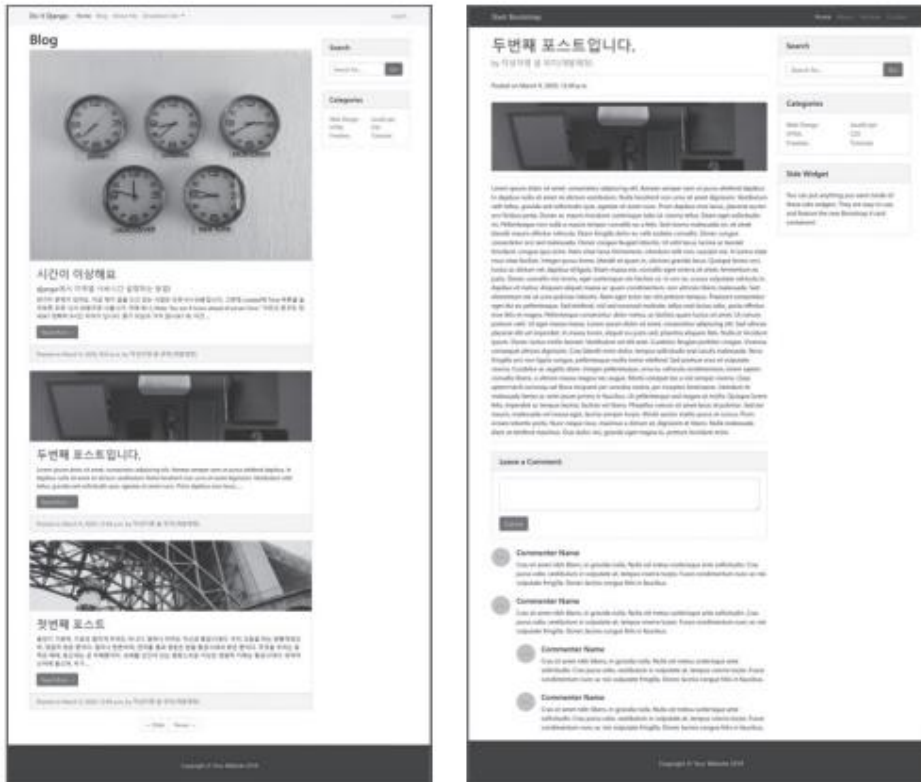


그림 12-1 포스트 목록 페이지(왼쪽)와 포스트 상세 페이지(오른쪽) 비교

post_list.html 모듈화하기

01단계 base.html 만들기

먼저 post_list.html을 복사한 다음 파일명을 base.html로 수정합니다. base.html은 공통 영역만 남기기 위한 파일입니다.

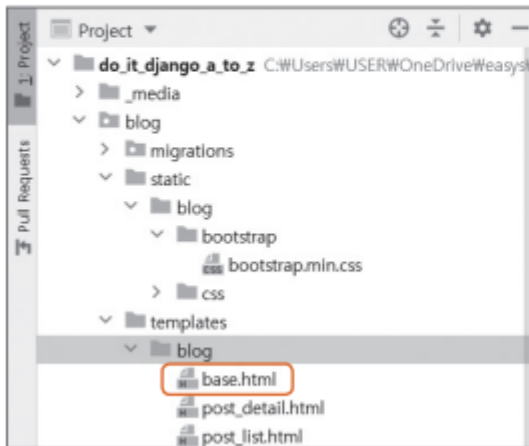


그림 12-2 post_list.html 복사 후 파일명을 base.html로 수정

실습 파일: blog/templates/blog/base.html

```
(...생략...)

<div class="container my-3">
  <div class="row">
    <div class="col-md-8 col-lg-9" id="main-area">
      <h1>Blog</h1>
      (...생략...)
      </h1>
      {% block main_area %}
      {% endblock %}
    </div>

    <div class="col-md-4 col-lg-3">

      (...생략...)
```

post_list.html 모듈화하기

02단계 base.html을 확장해 post_list.html 넣기

이제 post_list.html에는 block 안에 들어가는 요소만 있으면 되므로 base.html에서 지웠던 부분만 남기고 나머지를 전부 지워주세요. 그리고 맨 위에 `{% extends 'blog/base.html' %}`를 추가하고, 남긴 요소 앞 뒤로 `{% block main_area %}`와 `{% endblock %}`으로 블록의 시작과 끝을 알려주세요. 그럼 앞에서 만든 base.html의 main_area 블록을 post_list.html의 블록에 들어 있는 내용으로 채웁니다. 다음 코드는 post_list.html 파일 전체입니다. 길었던 post_list.html이 이렇게 짧아졌습니다.

실습 파일: blog/templates/blog/post_list.html

```
{% extends 'blog/base.html' %}
```

```
{% block main_area %}
```

```
<h1>Blog</h1>
```

```
{% if post_list.exists %}
```

```
    {% for p in post_list %}
```

```
        <!-- Blog Post -->
```

```
        <div class="card mb-4">
```

```
            {% if p.head_image %}
```

```
<!-- Pagination -->
```

```
<ul class="pagination justify-content-center mb-4">
```

```
    <li class="page-item">
```

```
        <a class="page-link" href="#">&larr; Older</a>
```

```
    </li>
```

```
    <li class="page-item disabled">
```

```
        <a class="page-link" href="#">Newer &rarr;</a>
```

```
    </li>
```

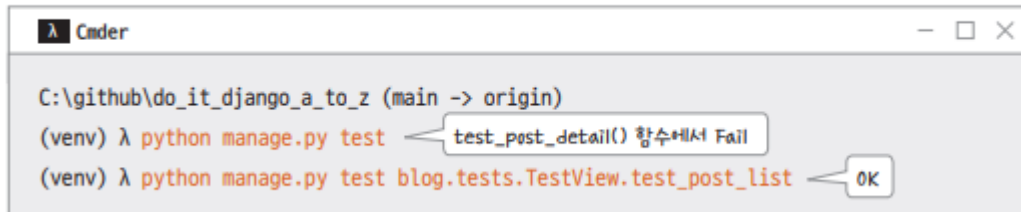
```
</ul>
```

```
{% endblock %}
```

수정결과 확인하기

03단계 수정 결과 확인하기

수정한 내용에 이상이 없다면 `test_post_list()` 함수의 테스트 항목은 모두 통과할 겁니다. 터미널에서 `python manage.py test`로 테스트하면 `test_post_detail()` 함수로 테스트할 때 Fail이 나올테니 `test_post_list()` 함수만 테스트하겠습니다. 터미널에서 `python manage.py test blog.tests.TestView.test_post_list`로 테스트해 보세요. 이번에는 OK가 나올 겁니다. 웹 브라우저에서도 확인해 보세요.



```

C:\github\do_it_django_a_to_z (main -> origin)
(venv) λ python manage.py test test_post_detail() 함수에서 Fail
(venv) λ python manage.py test blog.tests.TestView.test_post_list OK
    
```

post_detail.html 모듈화하기

Do it
실습

post_detail.html 모듈화하기

01단계 base.html을 확장해 post_detail.html 넣기

이제 post_detail.html도 base.html을 이용할 수 있게 수정하겠습니다. 방법은 동일합니다. post_detail.html에서 메인 영역에 해당하는 부분만 남기고 다 지운 후 블록을 지정하면 됩니다. 현재 post_detail.html은 <div class="container"> 안에 <div class="row">가 있고, <div class="col-lg-8">과 <div class="col-md-4">로 8:4로 나뉘어진 구조입니다. 이 중 <div class="col-lg-8"> 안에 있는 내용만 남기면 됩니다.

실습 파일: blog/templates/blog/post_detail.html

```
{% extends 'blog/base.html' %}

{% block main_area %}

    <!-- Title -->
    <h1 class="mt-4">{{ post.title }}</h1>
    <h5 class="text-muted">{{ post.hook_text }}</h5>
    <!-- Author -->
    <p class="lead">
        by
        <a href="#">작성자명 쓸 위치(개발예정)</a>
```

post_detail.html 모듈화하기

```

C:\github\do_it_django_a_to_z (main -> origin)
(venv) λ python manage.py test
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
F.

=====
FAIL: test_post_detail (blog.tests.TestView)
-----

Traceback (most recent call last):
  File "C:\github\do_it_django_a_to_z\blog\tests.py", line 70, in test_post_detail
    self.assertIn(post_001.title, soup.title)
AssertionError: '첫번째포스트입니다.' not found in 'Blog'

-----

Ran 2 tests in 0.276s

FAILED (failures=1)
Destroying test database for alias 'default'...
    
```

즉, 웹 브라우저 위쪽에 나타나는 타이틀(첫 번째 포스트의 제목)이 없다는 것입니다. 웹 브라우저의 타이틀은 <head> 태그 안에 있고, <head> 태그는 base.html에 있습니다. 이 부분은 post_list.html과 공유하고 있는데 어떻게 해야 할까요? post_detail.html에 다음과 같이 블록을 하나 더 만들면 됩니다.

post_detail.html 모듈화하기

즉, 웹 브라우저 위쪽에 나타나는 타이틀(첫 번째 포스트의 제목)이 없다는 것입니다. 웹 브라우저의 타이틀은 <head> 태그 안에 있고, <head> 태그는 base.html에 있습니다. 이 부분은 post_list.html과 공유하고 있는데 어떻게 해야 할까요? post_detail.html에 다음과 같이 블록을 하나 더 만들면 됩니다.

실습 파일: blog/templates/blog/post_detail.html

```
{% extends 'blog/base.html' %}

{% block head_title %}
    {{ post.title }} - Blog
{% endblock %}

{% block main_area %}
    <!-- Title -->
    <h1 class="mt-4">{{ post.title }}</h1>
    <h5 class="text-muted">{{ post.hook_text }}</h5>

    (...생략...)
```

실습 파일: blog/templates/blog/base.html

```
<!DOCTYPE html>
{% load static %}
<html lang="ko">
<html>

<head>
    <title>{% block head_title %}Blog{% endblock %}</title>
    <link rel="stylesheet" href="{% static 'blog/bootstrap/bootstrap.min.css' %}"
        media="screen">

    <script src="https://kit.fontawesome.com/*****.js"
        crossorigin="anonymous"></script>
</head>

(...생략...)
```

네비게이션 바 포함하기

01단계 내비게이션 바를 navbar.html로 모듈화하기

먼저 blog/templates/blog/ 폴더에 navbar.html을 만듭니다. 그리고 base.html의 <nav> 태그부터 그 아래 modal에 관련된 코드까지 잘라 navbar.html에 붙여 넣습니다.

실습 파일: blog/templates/blog/navbar.html

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container">
    <a class="navbar-brand" href="/">Do It Django</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNavDropdown" aria-controls="navbarNavDropdown" aria-expanded="false"
aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>

    (...생략...)

  </div>
</nav>
```

실습 파일: blog/templates/blog/base.html

```
(...생략...)
<body>

{% include 'blog/navbar.html' %}

<div class="container my-3">
  <div class="row">
    (...생략...)
```

푸터 포함하기

02단계 푸터를 footer.html로 모듈화하기

마찬가지로 푸터도 모듈화 해 봅시다. base.html에서 <footer> 태그 내용을 잘라내고 다음 코드로 대체하세요.

실습 파일: blog/templates/blog/base.html

```
(...생략...)
{% include 'blog/footer.html' %}
(...생략...)
```

실습 파일: blog/templates/blog/footer.html

```
<!-- Footer -->
<footer class="py-5 bg-dark">
  <div class="container">
    <p class="m-0 text-center text-white">
      Copyright &copy; Do It Django A to Z 2021
    </p>
  </div>
<!-- /.container -->
</footer>
```


과제 #12

1. 링크를 참조하여 주어진 템플릿으로 홈페이지를 만들어보시오.
2. 과제 12용 폴더를 새로 만들고, Startbootstrap 템플릿으로 게시물 목록과 게시물을 표시하도록 하시오. 추가적으로 single_pages를 구현해도 됩니다.

요구사항. 과제는 이메일로 제출할 것(제목/파일명 엄수). 제출기한 엄수 (2024.11.17. 05:00까지)

이메일 제목: [응소] 이름_이름 과제 #12_startbootstrap_템플릿 제출합니다.

첨부파일: 응소_과제12_김철수_홍길동_startbootstrap_템플릿 .mp4

이메일 내용: github 저장소 url

동영상은 변경된 부분을 강조하여 녹화하고, 정상적으로 동작하는 홈페이지를 간단하게 보여주면 됩니다.

받는사람: 교수님(taegon@jbnu.ac.kr), TA석승원(champ9162@naver.com)

테스트 주도 개발(TDD, Test Driven Development)

테스트 주도 개발을 왜 적용할까?

왜 테스트 주도 개발을 적용하는지 그 이유를 알아보기 위해 지금까지는 어떤 방식으로 웹 사이트를 개발했는지 오른쪽 그림과 같이 표현해 보았습니다. 각 과정에서 했던 일을 자세히 정리해 보면 다음과 같습니다.

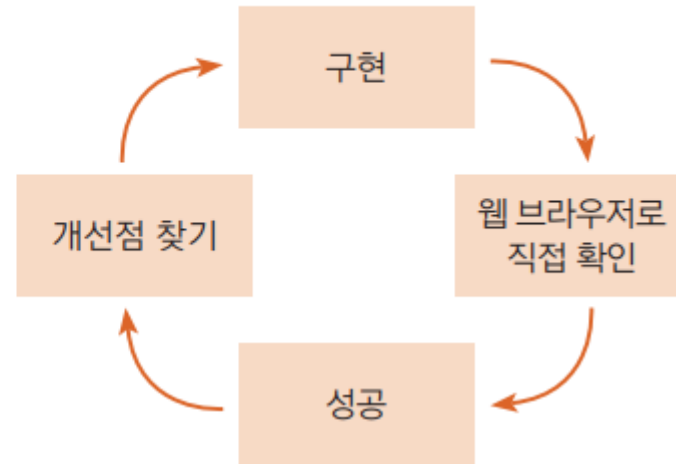


그림 11-1 웹 사이트 제작의 기본 과정

- 구현: 만들고 싶은 요소를 떠올리고 소스 코드를 작성합니다.
- 웹 브라우저로 직접 확인: 웹 브라우저로 들어가서 잘 작동하는지 일일이 테스트합니다.
- 성공: 제대로 작동하지 않으면 다시 소스 코드를 들여다보고 수정하면서 성공시킵니다.
- 개선점 찾기: 다음에는 무엇을 더 개선해야 할지, 무엇을 개발해야 할지 고민을 한 후 다시 구현해 봅니다.

테스트 주도 개발(TDD, Test Driven Development)

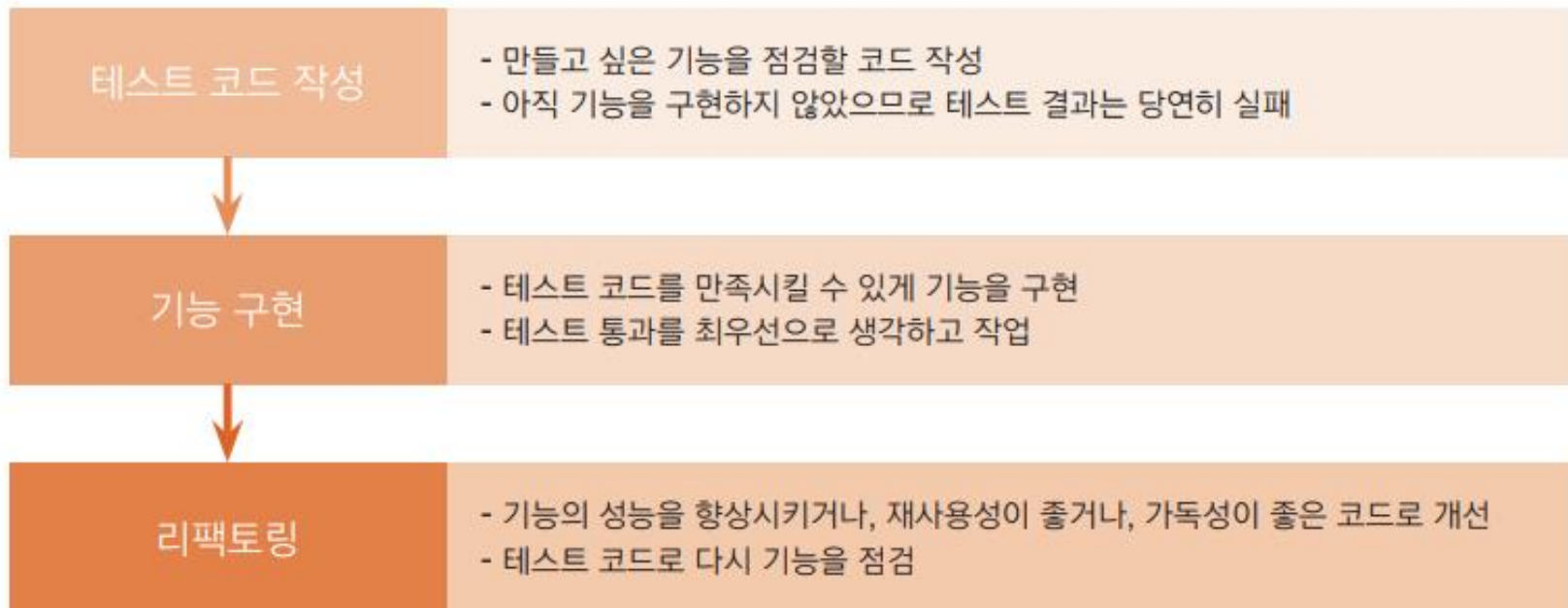
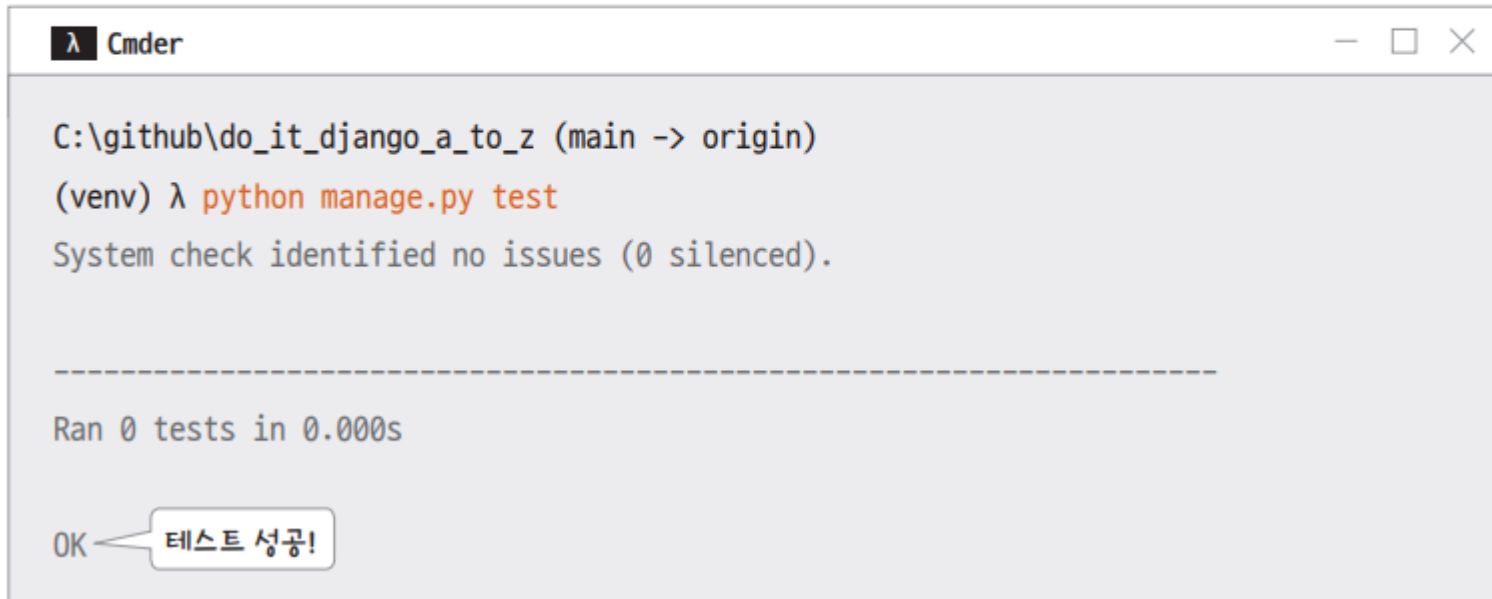


그림 11-2 테스트 주도 개발 과정

테스트 주도 개발(TDD, Test Driven Development)

01단계 테스트 코드 사용 연습하기

터미널에서 `python manage.py test`라고 입력하세요. 0개의 테스트를 실행한 결과 OK가 나왔습니다. 당연한 결과입니다. 아무런 테스트 미션을 주지 않았으니까요.



```
λ Cmder

C:\github\do_it_django_a_to_z (main -> origin)
(venv) λ python manage.py test
System check identified no issues (0 silenced).

-----

Ran 0 tests in 0.000s

OK
```

테스트 성공!

테스트 주도 개발(TDD, Test Driven Development)

실습 파일: blog/tests.py

```
from django.test import TestCase

class TestView(TestCase):
    def test_post_list(self):
        self.assertEqual(2, 3)
```

λ Cmder

C:\github\do_it_django_a_to_z (main -> origin)

(venv) λ python manage.py test

(...생략...)

Traceback (most recent call last):

File "C:\github\do_it_django_a_to_z\blog\tests.py", line 6, in test_post_list

self.assertEqual(2, 3)

AssertionError: 2 != 3

Ran 1 test in 0.038s

FAILED (failures=1)

테스트 실패!

Destroying test database for alias 'default'...

테스트 주도 개발(TDD, Test Driven Development)

Do it
실습

포스트 목록 페이지 테스트 코드 작성하기

01단계 tests.py에 테스트할 내용 나열하기

blog/tests.py에 앞에서 연습 삼아 작성해 본 코드는 지우고 다음과 같이 내용을 작성하세요. 하나의 TestCase 내에서 기본적으로 설정되어야 하는 내용이 있으면 setUp() 함수에서 정의를 하면 됩니다. 현재는 setUp() 함수 내에 Client()를 사용하겠다는 내용만 담았습니다.

실습 파일: blog/tests.py

```
from django.test import TestCase, Client

class TestView(TestCase):
    def setUp(self):
        self.client = Client()

    def test_post_list(self):
        # 1.1 포스트 목록 페이지를 가져온다.
        # 1.2 정상적으로 페이지가 로드된다.
        # 1.3 페이지 타이틀은 'Blog'이다.
        # 1.4 내비게이션 바가 있다.
        # 1.5 Blog, About Me라는 문구가 내비게이션 바에 있다.

        # 2.1 메인 영역에 게시물이 하나도 없다면
        # 2.2 '아직 게시물이 없습니다'라는 문구가 보인다.

        # 3.1 게시물이 2개 있다면
        # 3.2 포스트 목록 페이지를 새로고침했을 때
        # 3.3 메인 영역에 포스트 2개의 타이틀이 존재한다.
        # 3.4 '아직 게시물이 없습니다'라는 문구는 더 이상 보이지 않는다.
```

이제 앞에서 작성한 시나리오를 점검할 수 있는 테스트 코드를 다음과 같이 작성합니다.

실습 파일: blog/tests.py

```
from django.test import TestCase, Client
from bs4 import BeautifulSoup
from .models import Post

class TestView(TestCase):
    def setUp(self):
        self.client = Client()

    def test_post_list(self):
        # 1.1. 포스트 목록 페이지를 가져온다.
        response = self.client.get('/blog/')
        # 1.2. 정상적으로 페이지가 로드된다.
        self.assertEqual(response.status_code, 200)
        # 1.3. 페이지 타이틀은 'Blog'이다.
        soup = BeautifulSoup(response.content, 'html.parser')
        self.assertEqual(soup.title.text, 'Blog')
        # 1.4. 내비게이션 바가 있다.
        navbar = soup.nav
        # 1.5. Blog, About Me라는 문구가 내비게이션 바에 있다.
        self.assertIn('Blog', navbar.text)
        self.assertIn('About Me', navbar.text)

        # 2.1. 포스트(게시물)가 하나도 없다면
        self.assertEqual(Post.objects.count(), 0)
        # 2.2. main area에 '아직 게시물이 없습니다'라는 문구가 나타난다.
```

TDD로 개발하기

- https://github.com/saintdragon2/do_it_django_a_to_z/commits/master/blog/tests.py

실습 파일: blog/tests.py

```
from django.test import TestCase, Client

class TestView(TestCase):
    def setUp(self):
        self.client = Client()

    def test_post_list(self):
        # 1.1 포스트 목록 페이지를 가져온다.
        # 1.2 정상적으로 페이지가 로드된다.
        # 1.3 페이지 타이틀은 'Blog'이다.
        # 1.4 내비게이션 바가 있다.
        # 1.5 Blog, About Me라는 문구가 내비게이션 바에 있다.

        # 2.1 메인 영역에 게시물이 하나도 없다면
        # 2.2 '아직 게시물이 없습니다'라는 문구가 보인다.

        # 3.1 게시물이 2개 있다면
        # 3.2 포스트 목록 페이지를 새로고침했을 때
        # 3.3 메인 영역에 포스트 2개의 타이틀이 존재한다.
        # 3.4 '아직 게시물이 없습니다'라는 문구는 더 이상 보이지 않는다.
```