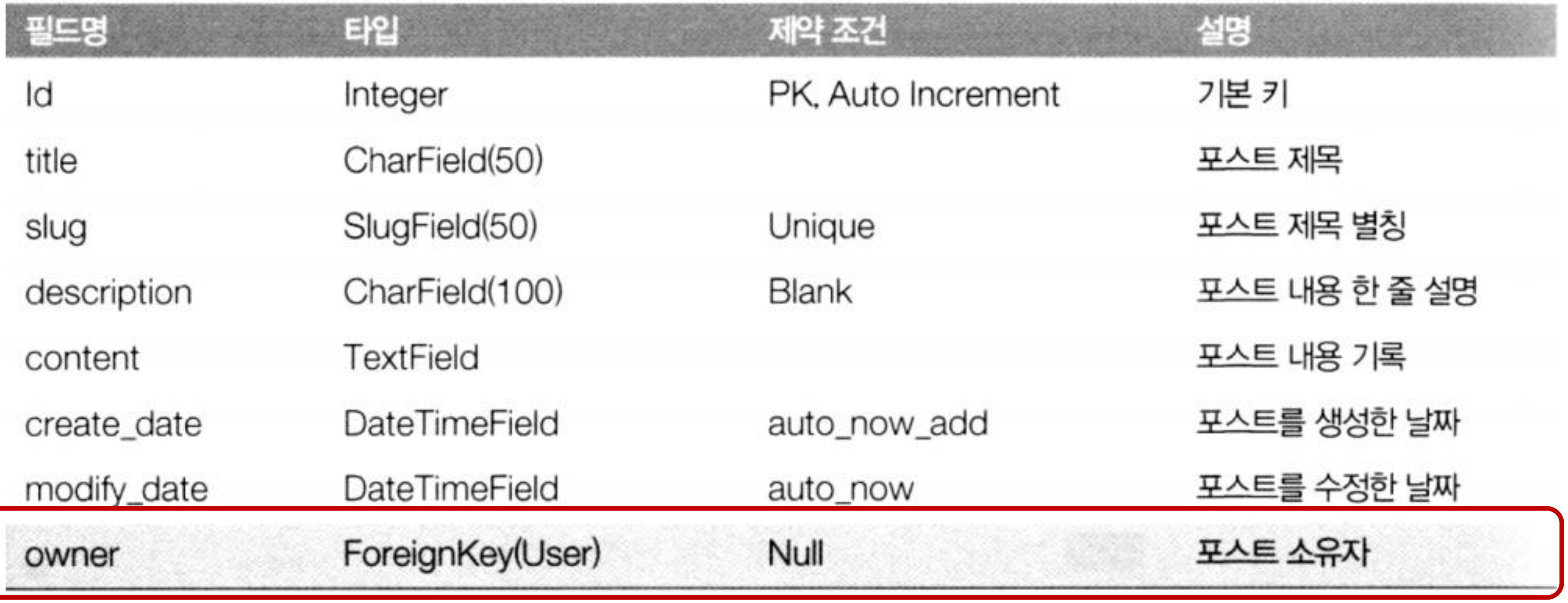
실전 프로그램 개발
- 콘텐츠 편집 기능 -

(Blog)

애플리케이션 설계하기

❖ 모델 설계

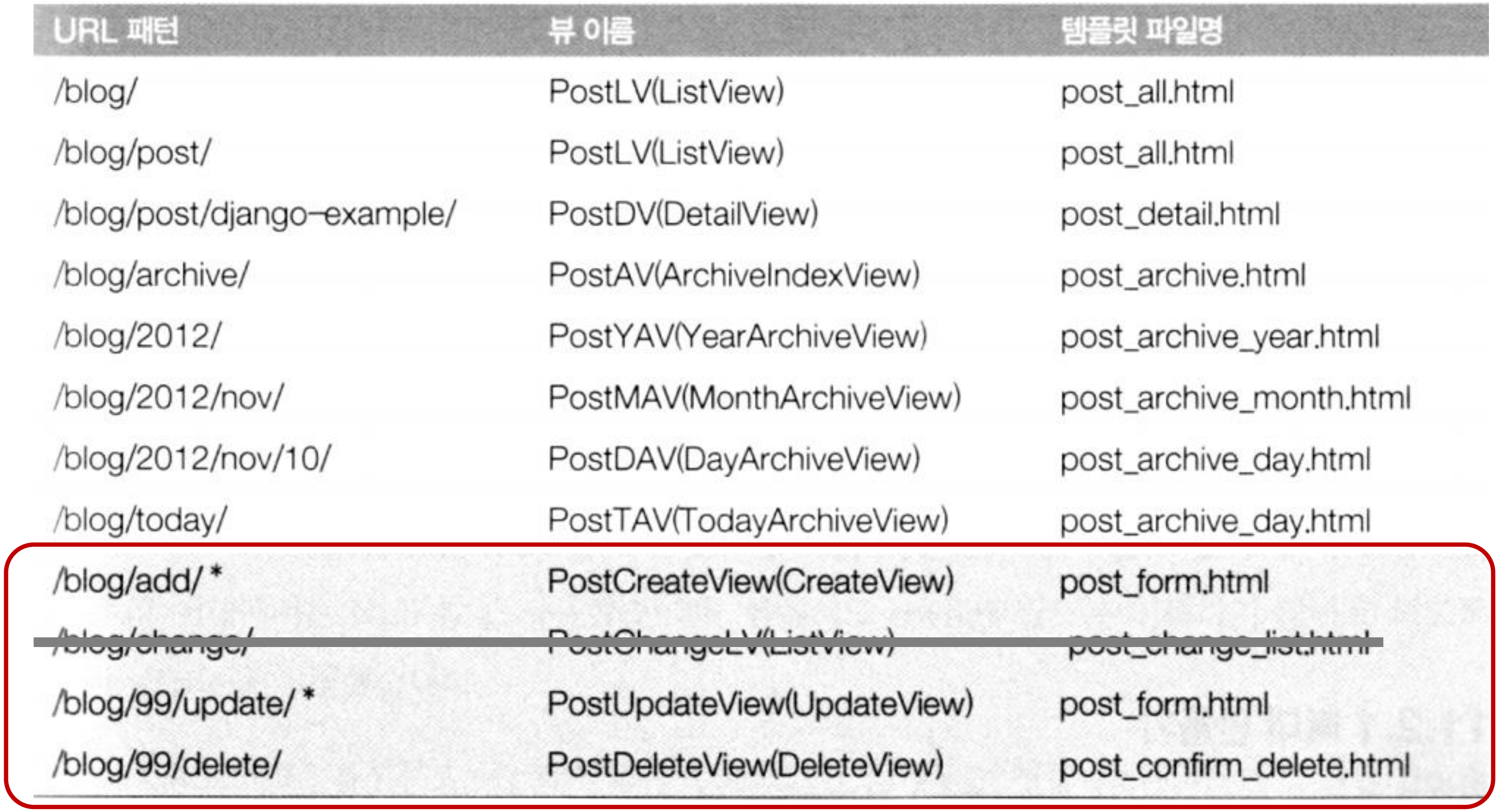
- Post 모델 클래스



필드명	타입	제약 조건	설명
Id	Integer	PK, Auto Increment	기본 키
title	CharField(50)		포스트 제목
slug	SlugField(50)	Unique	포스트 제목 별칭
description	CharField(100)	Blank	포스트 내용 한 줄 설명
content	TextField		포스트 내용 기록
create_date	DateTimeField	auto_now_add	포스트를 생성한 날짜
modify_date	DateTimeField	auto_now	포스트를 수정한 날짜
owner	ForeignKey(User)	Null	포스트 소유자

애플리케이션 설계하기

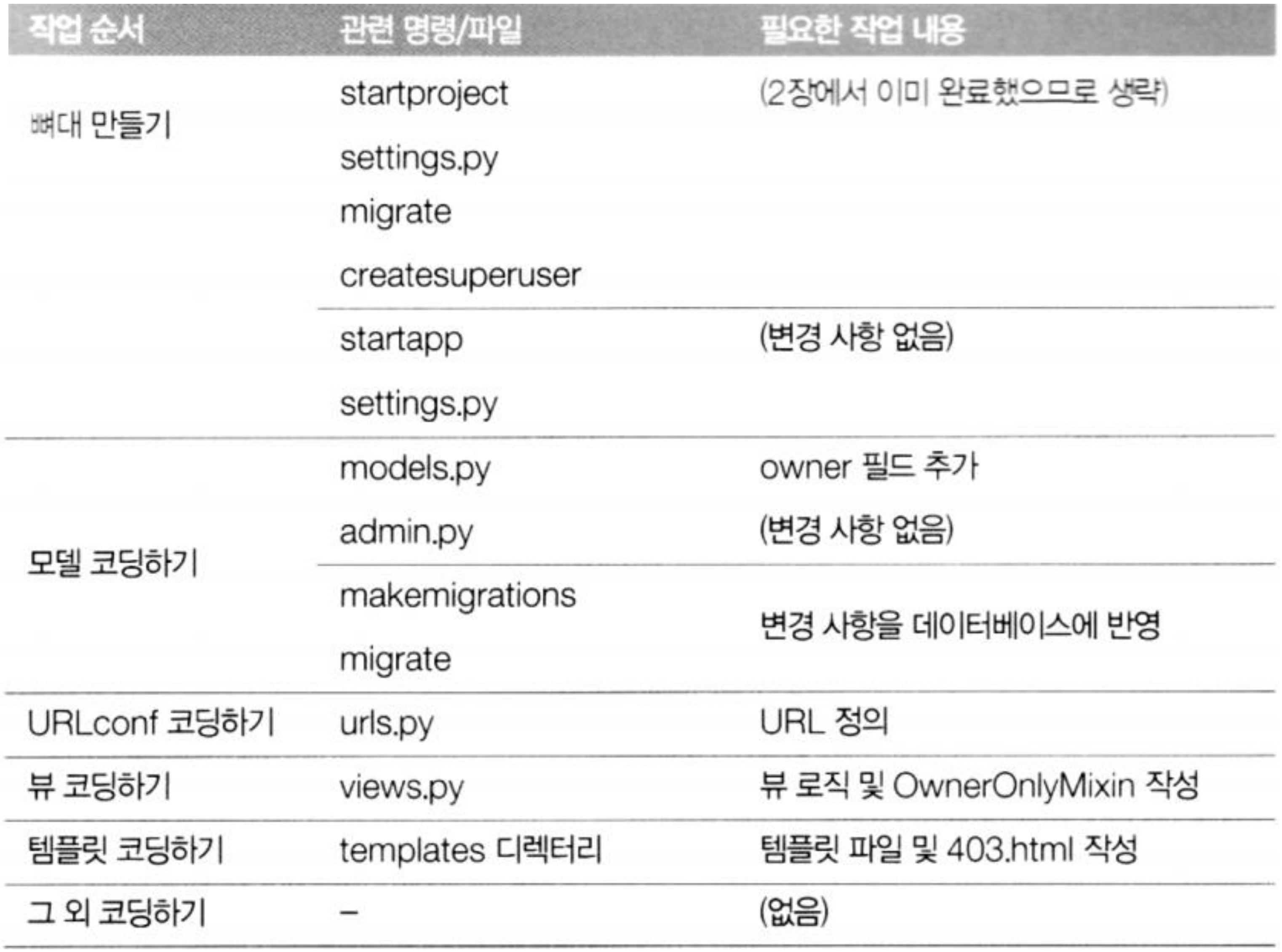
❖ Blog URL 설계



URL 패턴	뷰 이름	템플릿 파일명
/blog/	PostLV(ListView)	post_all.html
/blog/post/	PostLV(ListView)	post_all.html
/blog/post/django-example/	PostDV(DetailView)	post_detail.html
/blog/archive/	PostAV(ArchiveIndexView)	post_archive.html
/blog/2012/	PostYAV(YearArchiveView)	post_archive_year.html
/blog/2012/nov/	PostMAV(MonthArchiveView)	post_archive_month.html
/blog/2012/nov/10/	PostDAV(DayArchiveView)	post_archive_day.html
/blog/today/	PostTAV(TodayArchiveView)	post_archive_day.html
/blog/add/ *	PostCreateView(CreateView)	post_form.html
~~/blog/change/~~	~~PostChangeLV(ListView)~~	~~post_change_list.html~~
/blog/99/update/ *	PostUpdateView(UpdateView)	post_form.html
/blog/99/delete/	PostDeleteView(DeleteView)	post_confirm_delete.html

애플리케이션 설계하기

❖ 작업 순서



작업 순서	관련 명령/파일	필요한 작업 내용
뼈대 만들기	startproject	(2장에서 이미 완료했으므로 생략)
	settings.py	
	migrate	
	createsuperuser	
	startapp	(변경 사항 없음)
	settings.py	
모델 코딩하기	models.py	owner 필드 추가
	admin.py	(변경 사항 없음)
	makemigrations	변경 사항을 데이터베이스에 반영
	migrate	
URLconf 코딩하기	urls.py	URL 정의
뷰 코딩하기	views.py	뷰 로직 및 OwnerOnlyMixin 작성
템플릿 코딩하기	templates 디렉터리	템플릿 파일 및 403.html 작성
그 외 코딩하기	–	(없음)

개발 코딩하기

❖ **blog/models.py**

```
from django.db import models
from django.urls import reverse
from taggit.managers import TaggableManager
from django.contrib.auth.models import User
from django.utils.text import slugify


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

  owner = models.ForeignKey(User, on_delete=models.CASCADE,
                            verbose_name='OWNER', blank=True, null=True)
```

개발 코딩하기

❖ blog/models.py

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

def __str__(self):
  return self.title

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

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

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

def save(self, *args, **kwargs):
  self.slug = slugify(self.title, allow_unicode=True)
  super().save(*args, **kwargs)
```

개발 코딩하기

❖ 데이터베이스 반영

```
$ python manage.py makemigrations blog
$ python manage.py migrate
```

개발 코딩하기

❖ blog/views.py

```
:
from django.views.generic import CreateView, UpdateView, DeleteView
from django.contrib.auth.mixins import LoginRequiredMixin
from django.urls import reverse_lazy
from mysite.views import OwnerOnlyMixin

:

class PostCreateView(LoginRequiredMixin, CreateView):
  model = Post
  fields = ['title', 'slug', 'description', 'content', 'tags']
  initial = {'slug': 'auto-filling-do-not-input'}
  success_url = reverse_lazy('blog:index')

  def form_valid(self, form):
    form.instance.owner = self.request.user
    return super().form_valid(form)
```

개발 코딩하기

❖ blog/views.py

```
class PostUpdateView(OwnerOnlyMixin, UpdateView):
  model = Post
  fields = ['title', 'slug', 'description', 'content', 'tags']
  success_url = reverse_lazy('blog:index')


class PostDeleteView(OwnerOnlyMixin, DeleteView) :
  model = Post
  success_url = reverse_lazy('blog:index')
```

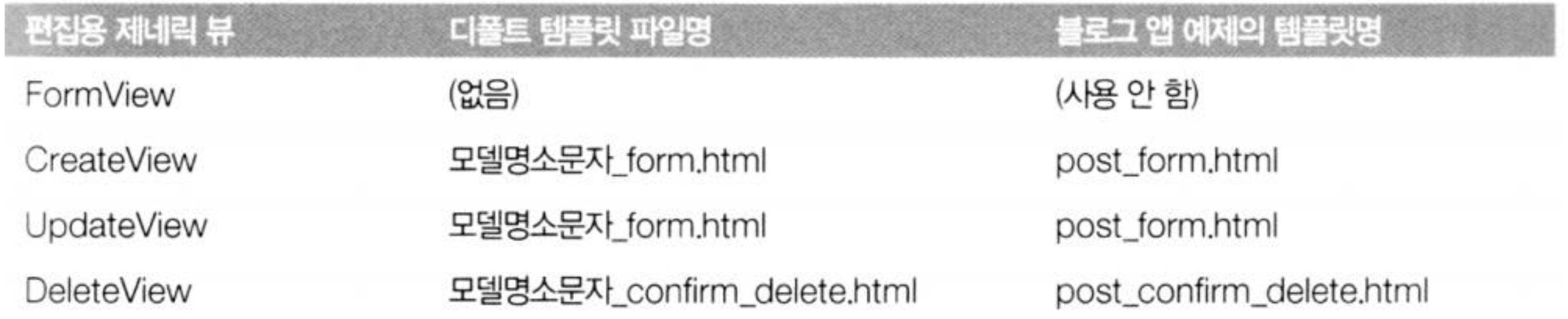
개발 코딩하기

❖ blog/urls.py

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

app_name = 'blog'
urlpatterns = [
    :
  # Example: /blog/add/
  path('add/', PostCreateView.as_view(), name="add"),

  # Example: /blog/99/update/
  path('<int:pk>/update/', PostUpdateView.as_view(), name="update"),

  # Example: /blog/99/delete/
  path('<int:pk>/delete/', PostDeleteView.as_view(), name="delete"),
]
```

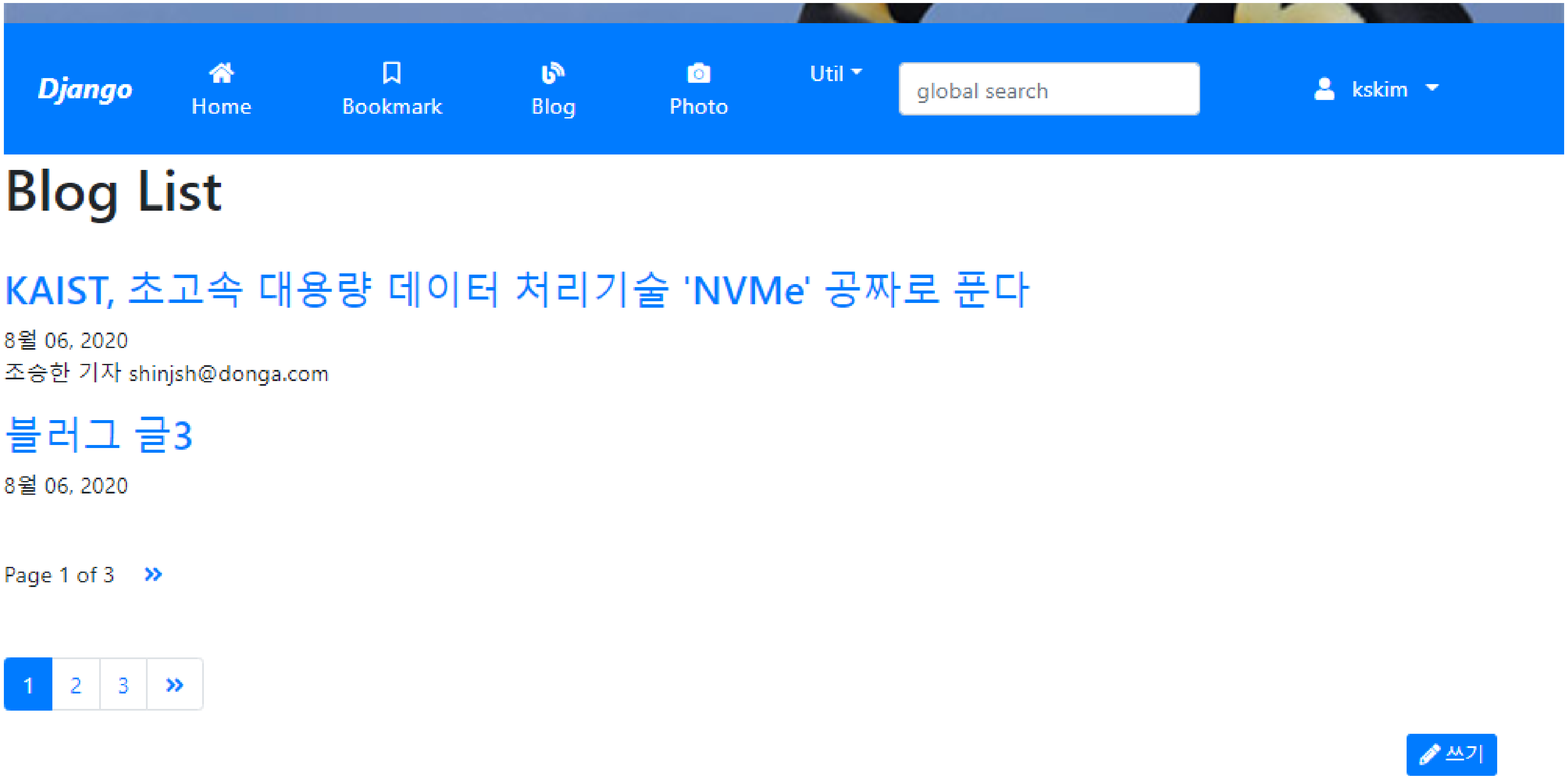
개발 코딩하기

❖ 템플릿



편집용 제네릭 뷰	디폴트 템플릿 파일명	블로그 앱 예제의 템플릿명
FormView	(없음)	(사용 안 함)
CreateView	모델명소문자_form.html	post_form.html
UpdateView	모델명소문자_form.html	post_form.html
DeleteView	모델명소문자_confirm_delete.html	post_confirm_delete.html

❖ 글 쓰기 인터페이스



Blog List

KAIST, 초고속 대용량 데이터 처리기술 'NVMe' 공짜로 푼다

8월 06, 2020

조승한 기자 shinjsh@donga.com

블러그 글3

8월 06, 2020

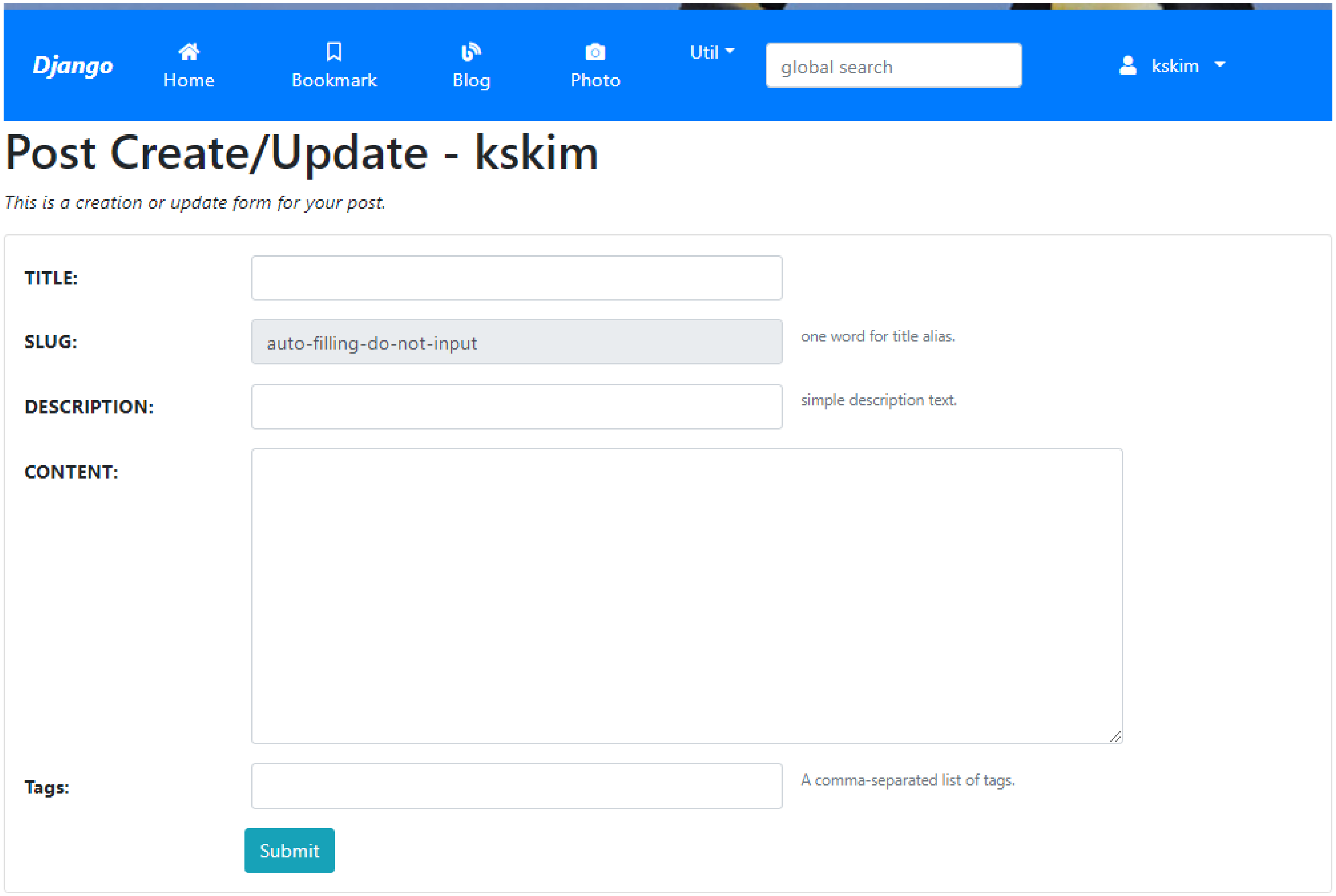
Page 1 of 3 »

개발 코딩하기

❖ **blog/templates/blog/post_all.html**

```
:
  {% include "pagination.html" %}

  {% if user.is_active %}
  <div class="text-right mr-3">
     <a href="{%url 'blog:add' %}" class="btn btn-primary btn-sm">
        <i class="fas fa-pencil-alt"></i> 쓰기</a>
  </div>
  {% endif %}

{% endblock %}
```

❖ 글 쓰기/수정 화면



Post Create/Update - kskim

This is a creation or update form for your post.

TITLE:		
SLUG:	auto-filling-do-not-input	one word for title alias.
DESCRIPTION:		simple description text.
CONTENT:		
Tags:		A comma-separated list of tags.

개발 코딩하기

❖ blog/templates/blog/post_form.html

```
{% extends "base.html" %}
{% load widget_tweaks %}

{% block title %}post_form.html{% endblock %}

{% block content %}
  <h1>Post Create/Update - {{user}}</h1>
  <p class="font-italic">This is a creation or update form for your post.</p>

  {% if form.errors %}
  <div class="alert alert-danger">
    <div class="font-weight-bold">
        Wrong! Please correct the error(s) below.</div>
    {{ form.errors }}
  </div>
  {% endif %}
```

개발 코딩하기

❖ **blog/templates/blog/post_form.html**

```
<form action="." method="post" class="card pt-3">{% csrf_token %}
  <div class="form-group row">
    {{ form.title|
       add_label_class:"col-form-label col-sm-2 ml-3 font-weight-bold" }}
    <div class="col-sm-5">
      {{ form.title|add_class:"form-control"|attr:"autofocus" }}
    </div>
  </div>

  <div class="form-group row">
    {{ form.slug|
       add_label_class:"col-form-label col-sm-2 ml-3 font-weight-bold" }}
    <div class="col-sm-5">
      {{ form.slug|add_class:"form-control"|attr:"readonly" }}
    </div>
    <small class="form-text text-muted">{{ form.slug.help_text }}</small>
  </div>
```

개발 코딩하기

❖ **blog/templates/blog/post_form.html**

```
    <div class="form-group row">
      {{ form.description|
         add_label_class:"col-form-label col-sm-2 ml-3 font-weight-bold" }}
      <div class="col-sm-5">
        {{ form.description|add_class:"form-control" }}
      </div>
      <small class="form-text text-muted">
        {{ form.description.help_text }}</small>
    </div>

    <div class="form-group row">
      {{ form.content|
         add_label_class:"col-form-label col-sm-2 ml-3 font-weight-bold" }}
      <div class="col-sm-8">
        {{ form.content|add_class:"form-control" }}
      </div>
    </div>
```

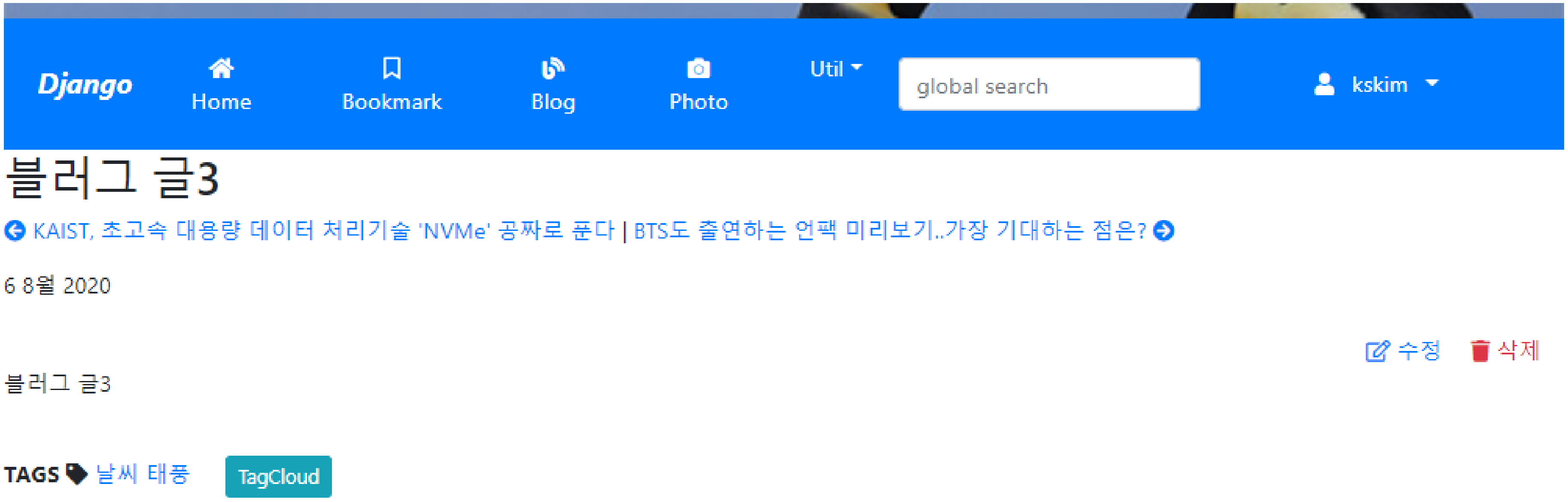
개발 코딩하기

❖ **blog/templates/blog/post_form.html**

```
    <div class="form-group row">
      {{ form.tags|
         add_label_class:"col-form-label col-sm-2 ml-3 font-weight-bold" }}
      <div class="col-sm-5">
        {{ form.tags|add_class:"form-control" }}
      </div>
      <small class="form-text text-muted">{{ form.tags.help_text }}</small>
    </div>

    <div class="form-group">
      <div class="offset-sm-2 col-sm-5">
        <input type="submit" value="Submit" class="btn btn-info"/>
      </div>
    </div>

  </form>

{% endblock %}
```

개발 코딩하기

❖ 글 수정/삭제 인터페이스



블러그 글3

KAIST, 초고속 대용량 데이터 처리기술 'NVMe' 공짜로 푼다 | BTS도 출연하는 언팩 미리보기..가장 기대하는 점은?

6 8월 2020

수정 삭제

블러그 글3

TAGS 날씨 태풍 TagCloud

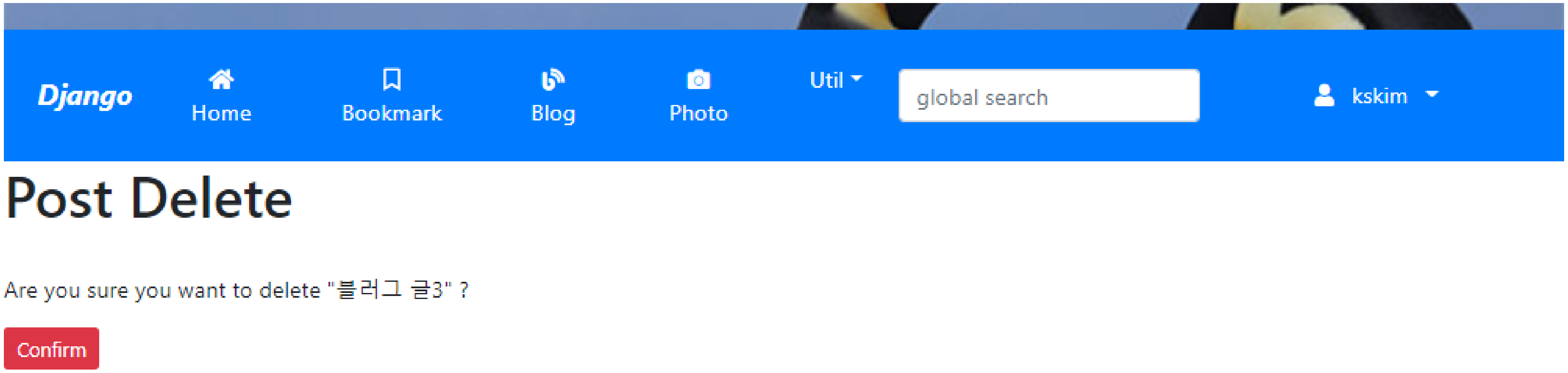
개발 코딩하기

❖ **blog/templates/blog/post_detail.html**

```
:

<div class="text-right">
  <a href="{% url 'blog:update' post.id %}" class="mr-3">
    <i class="far fa-edit"></i> 수정</a>
  <a href="{% url 'blog:delete' post.id %}" class="text-danger mr-3">
    <i class="fas fa-trash"></i> 삭제</a>
</div>


<div>
  {{ post.content|linebreaks }}
</div>

:
```

개발 코딩하기

❖ 글 삭제



Post Delete

Are you sure you want to delete "블러그 글3" ?

개발 코딩하기

❖ **blog/temaplates/blog/post_confirm_delete.html**

```
{% extends "base.html" %}

{% block title %}post_confirm_delete.html{% endblock %}

{% block content %}

  <h1>Post Delete</h1>
  <br>

  <form action="." method="post">{% csrf_token %}
    <p>Are you sure you want to delete "{{ object }}" ?</p>
    <input type="submit" value="Confirm" class="btn btn-danger btn-sm" />
  </form>

{% endblock %}
```