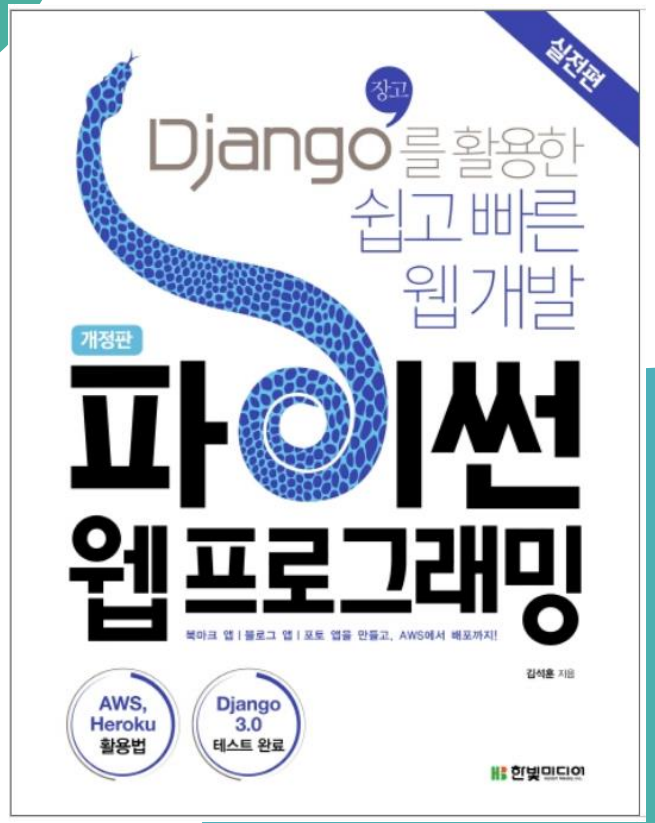
CHAPTER 12

파이썬 웹프로그래밍

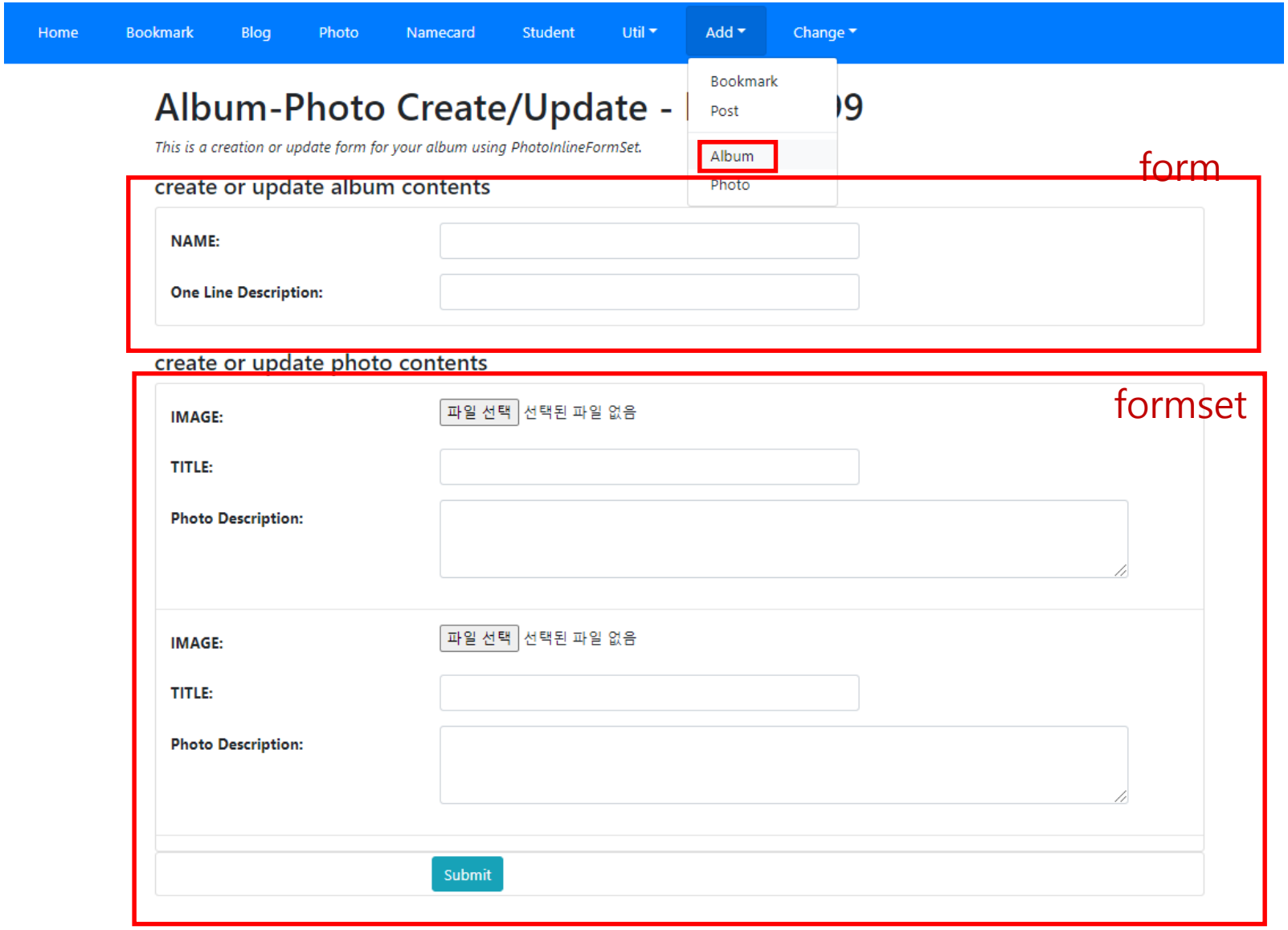


가천대학교 컴퓨터공학과
왕보현

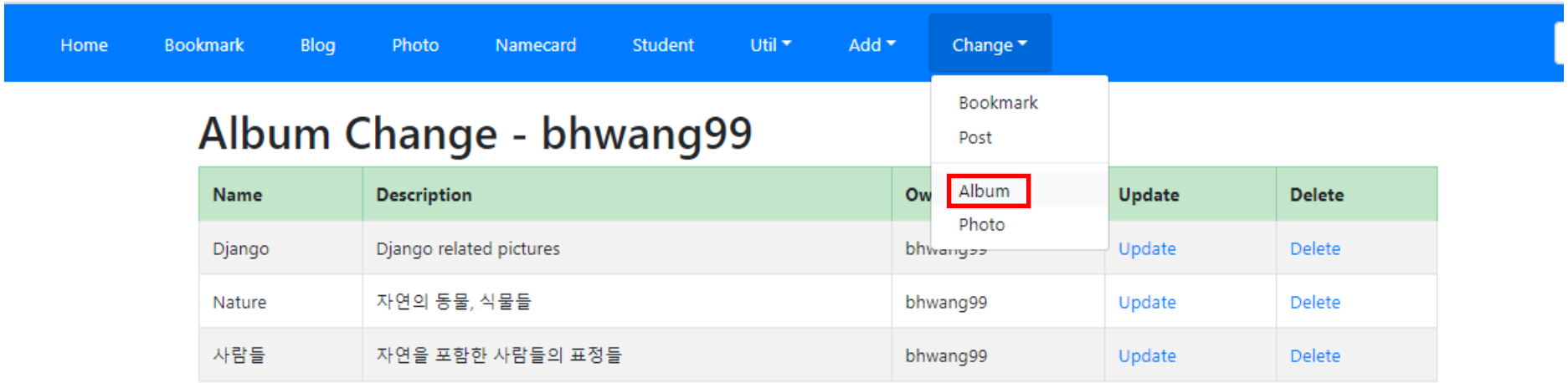
CHAPTER 12 실전 프로그램 개발 – 콘텐츠 편집 기능(Photo 앱)

- 11장과 비슷하나 Album과 Photo 테이블이 1:N 관계로 연결되어 있어, 생성이나 수정 폼 화면에서 이런 관계가 어떻게 구현되는지 이해하는 것이 중요
- 콘텐츠에 대한 열람은 모든 사용자가 가능
- 콘텐츠를 새로 생성하는 것은 로그인한 사용자만 가능
- 콘텐츠를 수정 또는 삭제하는 작업은 그 콘텐츠를 생성한 사용자만 가능

album_form.html create 화면



album_change_list.html 화면



album_form.html update 화면

Album Change - bhwang99

Name	Description	Owner	Update	Delete
Django	Django related pictures	bhwang99	Update	Delete
Nature	자연의 동물, 식물들	bhwang99	Update	Delete
사람들	자연을 포함한 사람들의 표정들	bhwang99	Update	Delete

Album-Photo Create/Update - bhwang99

This is a creation or update form for your album using PhotoInlineFormSet.

create or update album contents

NAME: Django

One Line Description: Django related pictures

reate or update photo contents

IMAGE: Currently: photo/2020/10/django-cloud.jpg
Change:
파일 선택 선택된 파일 없음

TITLE: Django Book Tag Cloud

Photo Description: 장고 책의 주요 단어로 만든 태그 클라우드 그래픽.

IMAGE: Currently: photo/2020/10/django-big.jpg
Change:
파일 선택 선택된 파일 없음

TITLE: Django Movie Starts

Photo Description: Django-unchanged 영화 포스터.
주요 배역들이 나와 있음.

IMAGE: Currently: photo/2020/10/그림2.png
Change:
파일 선택 선택된 파일 없음

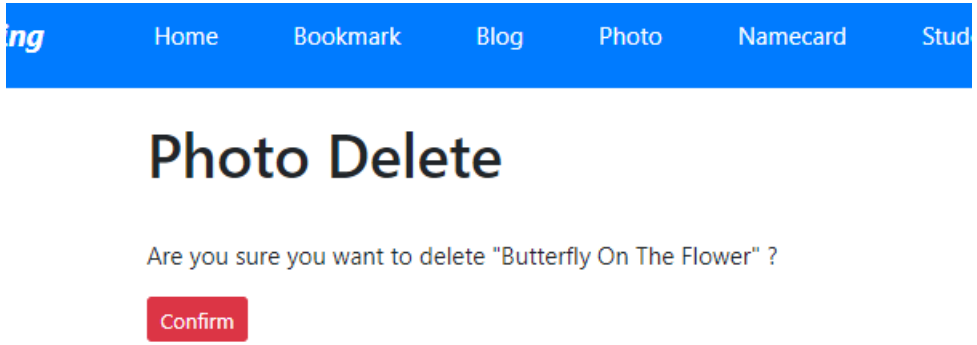
TITLE: Django 관련

Photo Description: ㅇㄴㄹㅇ

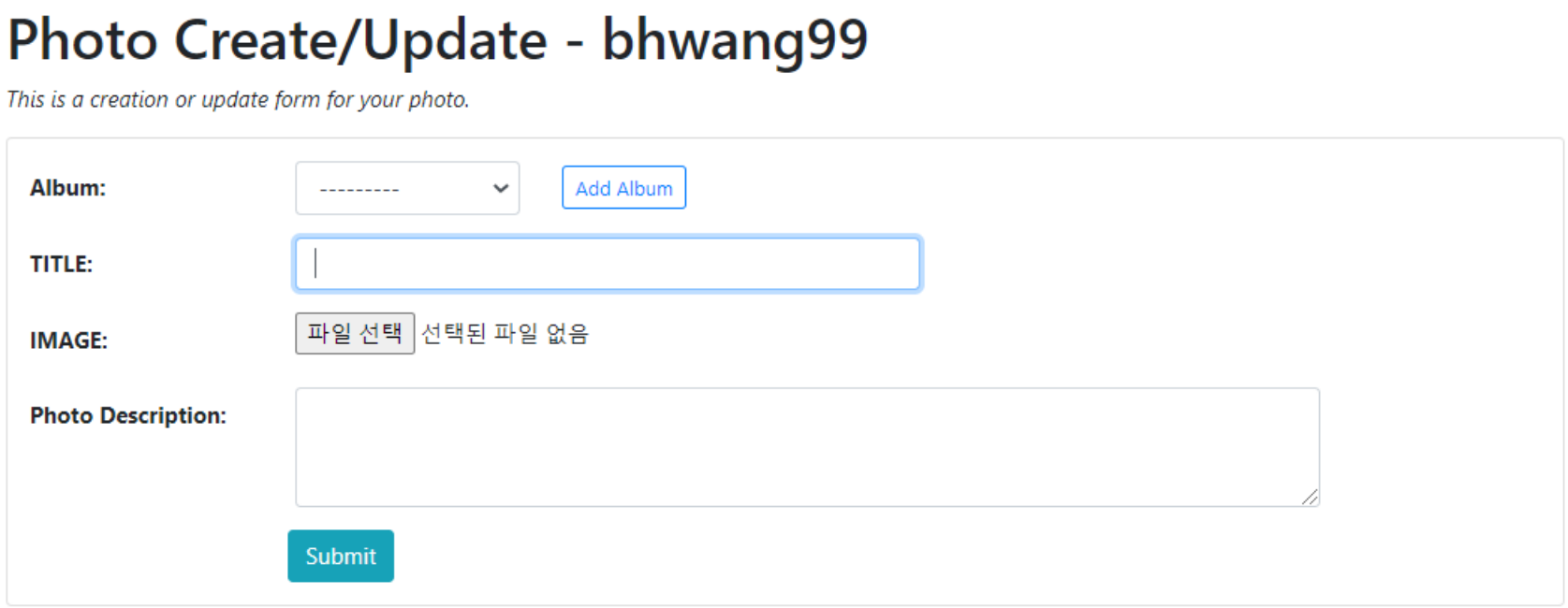
IMAGE: Currently: photo/2020/10/bg_login.jpg
Change:
파일 선택 선택된 파일 없음

TITLE: test

photo_confirm_delete.html 화면



photo_form.html 화면



2. URL 설계

URL 설계

URL 패턴	뷰 이름	템플릿 파일명
/photo/	AlbumLV(ListView)	album_list.html
/photo/album/	AlbumLV(ListView)	album_list.html
/photo/album/99	AlbumDV(DetailView)	album_detail.html
/photo/photo/99	PhotoDV(DetailView)	photo_detail.html
/photo/album/add/	AlbumPhotoCV(CreateView)	album_form.html
/photo/album/change/	AlbumChangeLV(ListView)	album_change_list.html
/photo/album/99/update/	AlbumPhotoUV(UpdateView)	album_form.html
/photo/album/99/delete/	AlbumDeleteView(DeleteView)	album_confirm_delete.html
/photo/photo/add/	PhotoCreateView(CreateView)	photo_form.html
/photo/photo/change/	PhotoChangeView(ListView)	photo_change_list.html
/photo/photo/99/update/	PhotoUpdateView(UpdateView)	photo_form.html
/photo/photo/99/delete/	PhotoDeleteView(DeleteView)	photo_confirm_delete.html

1. 테이블 설계

ch99₩photo₩models.py - 테이블 수정. 기존 album과 photo 테이블에 owner 컬럼 추가

```
from django.db import models
from django.urls import reverse

from photo.fields import ThumbnailImageField


class Album(models.Model):
    name = models.CharField('NAME', max_length=30)
    description = models.CharField('One Line Description', max_length=100, blank=True)
    owner = models.ForeignKey('auth.User', on_delete=models.CASCADE, verbose_name='OWNER', blank=True, null=True) #추가 ①

     class Meta:
        ordering = ('name',)

    def __str__(self):
        return self.name

    def get_absolute_url(self):
        return reverse('photo:album_detail', args=(self.id,))
```

① 로그인한 사람의 id가 입력되는 컬럼 , User Table에 있는 id 값 중 하나가 입력되어야 함.
on_delete=models.CASCADE 는 User Table의 login ID가 삭제가 되면 그 ID로 입력된 bookmark 테이블의 모든 레코드도 삭제하라는 의미

null=True의 이유는 기존 이미 입력된 자료는 owner 컬럼 값이 null이어야 하기 때문
만약 migrate로 컬럼 추가가 안 된다면 테이블의 모든 자료를 지우고 추가 해 볼것

owner 컬럼이 ForeignKey 이므로 테이블에서 _id가 붙어서 컬럼명이 설정 ➔ owner_id

1. 테이블 설계

ch99₩photo₩models.py - 테이블 수정. 9장의 기존 album과 photo 테이블에 owner 컬럼 추가

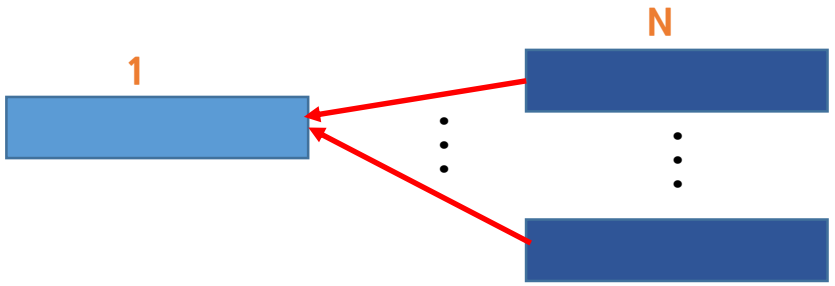
```
class Photo(models.Model):
    album = models.ForeignKey(Album, on_delete=models.CASCADE)
    title = models.CharField('TITLE', max_length=30)
    description = models.TextField('Photo Description', blank=True)
    image = ThumbnailImageField('IMAGE', upload_to='photo/%Y/%m')
    upload_dt = models.DateTimeField('UPLOAD DATE', auto_now_add=True)
    owner = models.ForeignKey('auth.User', on_delete=models.CASCADE, verbose_name='OWNER', blank=True, null=True) # 추가 ①
```

① Album과 User 테이블 간 관계 및 Photo와 User 테이블 간 관계는 모두 N : 1 관계

makemigrations , migrate

※ ForeignKey에 대해 : 외부 테이블의 키를 참조

User 테이블 : Photo 테이블 ➔ 1 : N의 관계



User (1)

테이블(T): auth_user

Filter in any column

	id	password	last_login	is_superuser	username	first_name	
	…	필터	필터	필터	필터	필터	필터
1	1	pbkdf2_sha256$180000$bMRSBtjjR3A3…	2020-10-15 08:33:37.254645	1	bhwang99		bhwa
2	2	pbkdf2_sha256$180000$BGliyuaDn0r3$…	NULL	0	test		
3	3	pbkdf2_sha256$180000$8YnJluLdcVM…	2020-10-15 14:54:01.446936	0	testtest		

Photo (N)

id	title	description	image	upload_dt	album_id	owner_id
…	필터	필터	필터	필터	필터	필터
1	장고책 기본편 - 개정판	장고책 표지 그림.…	photo/2020/10/djangobook-1.jpg	2020-10-17 19:41:14.999478	1	1
2	장고책 실전편	장고책-실전편 (외면, 파랑책) 표지 …	photo/2020/10/djangobook-2.PNG	2020-10-17 19:41:15.017444	1	NULL

2. URL 설계

ch99₩photo₩urls.py

```python
from django.urls import path
from photo import views

app_name = 'photo'
urlpatterns = [
    # Example: /photo/
    path('', views.AlbumLV.as_view(), name='index'),
    # Example: /photo/album/, same as /photo/
    path('album', views.AlbumLV.as_view(), name='album_list'),
    # Example: /photo/album/99/
    path('album/<int:pk>/', views.AlbumDV.as_view(), name='album_detail'),
    # Example: /photo/photo/99/
    path('photo/<int:pk>/', views.PhotoDV.as_view(), name='photo_detail'),

    ### 9장의 urls.py 파일에 이하 추가
     # Example: /photo/album/add/
    path('album/add/', views.AlbumPhotoCV.as_view(), name='album_add'),

    # Example: /photo/album/change/
    path('album/change/', views.AlbumChangeLV.as_view(), name='album_change'),

    # Example: /photo/album/99/update/
    path('album/<int:pk>/update/', views.AlbumPhotoUV.as_view(), name='album_update'),

    # Example: /photo/album/99/delete/
    path('album/<int:pk>/delete/', views.AlbumDelV.as_view(), name='album_delete'),

    # Example: /photo/photo/add/
    path('photo/add/', views.PhotoCV.as_view(), name='photo_add'),

    # Example: /photo/photo/change/
    path('photo/change/', views.PhotoChangeLV.as_view(), name='photo_change'),

    # Example: /photo/photo/99/update/
    path('photo/<int:pk>/update/', views.PhotoUV.as_view(), name='photo_update'),

    # Example: /photo/photo/99/delete/
    path('photo/<int:pk>/delete/', views.PhotoDelV.as_view(), name='photo_delete'),
]
```

2. forms 설계

ch99₩photo₩forms.py

```
from django.forms import inlineformset_factory   ①
from photo.models import Album, Photo            ②

PhotoInlineFormSet = inlineformset_factory(Album, Photo, fields = ['image', 'title', 'description'], extra = 2) ③
```

① 인라인 폼셋을 반환하는 inlineformset_factory() 함수를 임포트

② 폼셋에 사용할 모델을 임포트

③ 1:N 관계인 Album과 Photo 테이블을 이용해 사진 인라인 폼셋을 생성
fields를 통해 Photo 모델에서 폼셋에 사용하는 필드를 지정
폼셋에 들어 있는 빈 폼의 개수는 2개로 지정

inlineformset_factory

https://docs.djangoproject.com/en/3.1/ref/forms/models/

inlineformset_factory(parent_model, model, form=ModelForm, formset=BaseInlineFormSet, fk_name=None, fields=None, exclude=None, extra=3, can_order=False, can_delete=True, max_num=None, formfield_callback=None, widgets=None, validate_max=False, localized_fields=None, labels=None, help_texts=None, error_messages=None, min_num=None, validate_min=False, field_classes=None)

Returns an **InlineFormSet** using **modelformset_factory()** with defaults of **formset=BaseInlineFormSet**, **can_delete=True**, and **extra=3**.

If your model has more than one **ForeignKey** to the **parent_model**, you must specify a **fk_name**.

See Inline formsets for example usage.

3. View 설계

ch99₩photo₩views.py

```
from django.views.generic import ListView, DetailView
from photo.models import Album, Photo

## 하단 6줄 추가
from django.contrib.auth.mixins import LoginRequiredMixin
from django.shortcuts import redirect    ① 리다이렉트를 위한 단축 함수
from django.urls import reverse_lazy        를 임포트
from django.views.generic import CreateView, UpdateView, DeleteView
from mysite.views import OwnerOnlyMixin
from photo.forms import PhotoInlineFormSet


class AlbumLV(ListView):
    model = Album

class AlbumDV(DetailView):
    model = Album

class PhotoDV(DetailView):
    model = Photo
```

```
# 이하 추가
#--- Create/Change-list/Update/Delete for Photo
class PhotoCV(LoginRequiredMixin, CreateView):
    model = Photo
    fields = ('album', 'title', 'image', 'description')
    success_url = reverse_lazy('photo:index')

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


class PhotoChangeLV(LoginRequiredMixin, ListView):
    model = Photo
    template_name = 'photo/photo_change_list.html'

    def get_queryset(self):
        return Photo.objects.filter(owner=self.request.user)


class PhotoUV(OwnerOnlyMixin, UpdateView):
    model = Photo
    fields = ('album', 'title', 'image', 'description')
    success_url = reverse_lazy('photo:index')


class PhotoDelV(OwnerOnlyMixin, DeleteView):
    model = Photo
    success_url = reverse_lazy('photo:index')
```

3. View 설계

ch99₩photo₩views.py

```
#이하 추가
#--- Change-list/Delete for Album
class AlbumChangeLV(LoginRequiredMixin, ListView):
    model = Album
    template_name = 'photo/album_change_list.html'

    def get_queryset(self):
        return Album.objects.filter(owner=self.request.user)


class AlbumDelV(OwnerOnlyMixin, DeleteView):
    model = Album
    success_url = reverse_lazy('photo:index')
```

```
#--- (InlineFormSet) Create/Update for Album
class AlbumPhotoCV(LoginRequiredMixin, CreateView):
    model = Album
    fields = ('name', 'description')
    success_url = reverse_lazy('photo:index')

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        if self.request.POST:
            context['formset'] = PhotoInlineFormSet(self.request.POST, self.request.FILES)
        else:
            context['formset'] = PhotoInlineFormSet()
        return context

    def form_valid(self, form):
        form.instance.owner = self.request.user
        context = self.get_context_data()
        formset = context['formset']
        for photoform in formset:
            photoform.instance.owner = self.request.user
        if formset.is_valid():
            self.object = form.save()
            formset.instance = self.object
            formset.save()
            return redirect(self.get_success_url())
        else:
            return self.render_to_response(self.get_context_data(form=form))
```

3. View 설계

ch99₩photo₩views.py - AlbumPhotoCV 설명

```
def get_context_data(self, **kwargs):    ②
    context = super().get_context_data(**kwargs)  ③
    if self.request.POST:          ④
        context['formset'] = PhotoInlineFormSet(self.request.POST, self.request.FILES)
    else:
        context['formset'] = PhotoInlineFormSet()  ⑤
    return context    ⑥

def form_valid(self, form):
    form.instance.owner = self.request.user
    context = self.get_context_data()
    formset = context['formset']
    for photoform in formset:
        photoform.instance.owner = self.request.user
    if formset.is_valid():
        self.object = form.save()
        formset.instance = self.object
        formset.save()
        return redirect(self.get_success_url())
    else:
        return self.render_to_response(self.get_context_data(form=form))
```

② 장고에서 제공하는 디폴트 컨텍스트 변수 이외에 추가적인 컨텍스트 변수를 정의하기 위해 get_context_data() 메소드를 오버라이딩 정의함. 여기서는 formset 컨텍스트 변수를 추가

③ AlbumPhotoCV 부모 클래스의 get_context_data() 메소드를 호출해 기본 컨텍스트 변수를 설정. 이 예제에서는 기본 컨텍스트 변수 이외에 메인 폼도 컨텍스트 변수에 추가함.

④ POST 요청인 경우, formset 컨텍스트 변수를 request.POST와 request.FILES 파라미터를 사용해 지정함. request.FILES 파라미터를 추가한 이유는 파일 업로드가 이뤄지기 때문

업로드 되는 파일의 정보가 request.FILES에 저장

⑤ GET 요청인 경우에는 formset 컨텍스트틑 변수에 빈 폼셋을 지정함.

⑥ context 라는 context 변수 사전을 반환

참고 : 어떤 값을 넣을지 모르는데 *args는 값을 넣으면 함수에 변수가 튜플형태로 입력되는 것이고, **kwargs는 딕셔너리 형태로 입력되는 것이라고 보면 된다.
출처: https://toughbear.tistory.com/entry/python-args와-kwargs-의미와-사용 [Toughbear의 비개발자를 위한 개발 이야기]

3. View 설계

ch99₩photo₩views.py - AlbumPhotoCV 설명

```
def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)
    if self.request.POST:
        context['formset'] = PhotoInlineFormSet(self.request.POST, self.request.FILES)
    else:
        context['formset'] = PhotoInlineFormSet()
    return context

def form_valid(self, form):      ⑦
    form.instance.owner = self.request.user ⑧
    context = self.get_context_data()          ⑨
    formset = context['formset']               ⑨
    for photoform in formset:
        photoform.instance.owner = self.request.user ⑩
    if formset.is_valid():                     ⑪
        self.object = form.save()              ⑫
        formset.instance = self.object         ⑬
        formset.save()                         ⑭
        return redirect(self.get_success_url()) ⑮
    else:
        return self.render_to_response(self.get_context_data(form=form))⑯
```

⑦ 폼에 입력된 내용에 대해 유효성 검사를 수행해 오류가 없는 경우 form_valid() 메소드를 호출

⑧ 폼에 연결된 owner 필드에는 현재 login된 user id를 입력

⑨ 앞에서 정의한 get_context_data()메소드를 호출해 formset 객체를 구함.
이 시점에서 formset 데이터는 유효성 검사 전이고, form 데이터는 유효성 검사를 통과한 데이터이다.

⑩ 폼셋에 들어 있는 각 폼의 owner 필드에 현재 login된 user id 를 입력

⑪ 폼셋에 들어 있는 각 사진 폼의 데이터가 모두 유효한지 확인함

⑫ form.save()를 호출해 폼의 데이터를 테이블에 저장. 즉 앨범 레코드를 하나 생성

⑬ 폼셋의 메인 객체를 방금 테이블에 저장한 객체로 지정
즉, Photo테이블의 album 컬럼에 상단에서 입력한 Album 레코드의 id를 foreign key로 저장

⑭ formset.save()를 호출해 폼셋의 데이터를 테이블에 저장. 즉 12번에서 생성한 1:N 관계로 연결된 여러 개의 사진 레코드를 테이블에 저장

⑮ 마지막으로 get_success_url() 메소드 및 redirect() 단축 함수를 호출해 페이지를 이동. 즉 앨범 리스트 페이지로 리다리렉트

⑯ 폼셋의 데이터가 유효하지 않으면, 다시 메인 폼 및 인라인 폼셋을 출력함.
이때의 폼과 폼셋에는 직전에 사용자가 입력한 데이터를 다시 보여줌.

3. View 설계

ch99₩photo₩views.py

- GET, POST
  - GET과 POST는 HTTP프로토콜을 이용해서 서버에 무언가를 전달할 때 사용하는 방식
  - GET은 주소줄에 값이 ?뒤에 쌍으로 이어 붙고 POST는 숨겨져서(body안에) 보내진다.
  - GET은 URL에 이어붙기 때문에 길이제한이 있어서 많은양의 데이터는 보내기 어렵고 POST는 많은 양의 보내기에도 적합하다.(용량제한은 있음)
  - 즉 http://url/bbslist.html?id=5&pagenum=2 같이 하는 것이 GET방식
  - id를 넘겨서 게시판의 리스트를 가져온다고 하면 당연히 GET을 쓸 것이고 글을 작성한다고 하면 POST를 작성하는 것이 일반적.
  - 전달해야 될 양이 많을 경우에는 고민 없이 POST를 사용. 양이 많지 않은 경우에는 GET도 되고 POST도 됨.
  - GET은 가져오는 것이고 POST는 수행하는 것입니다.
  - GET은 Select적인 성향, POST는 서버의 값이나 상태를 바꾸기 위해서 사용
  - GET은 서버에서 어떤 데이터를 가져와서 보여준다거나 하는 용도이지 서버의 값이나 상태등을 바꾸지 않음 게시판의 리스트라던지 글보기 기능 같은 것이 이에 해당
  - 글쓰기를 하면 글의 내용이 디비에 저장이 되고 수정을 하면 디비값이 수정. 이럴 경우에 POST를 사용

출처 : https://blog.outsider.ne.kr/312

3. View 설계

ch99₩photo₩views.py

url 요청

photo/album/add

request.GET

view.py 파일

AlbumPhotoCV.as_view()

```
def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)
    if self.request.POST:
        context['formset'] = PhotoInlineFormSet(self.request.POST, self.request.FILES)
    else:
        context['formset'] = PhotoInlineFormSet()
    return context
```

album_form.html 문서 보여주기

html 문서 form에 자료 입력 후 submit 버튼 클릭

request
POST, FILES

form의 유효성 검사

def form_valid() 함수 실행

```
def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)
    if self.request.POST:
        context['formset'] = PhotoInlineFormSet(self.request.POST, self.request.FILES)
    else:
        context['formset'] = PhotoInlineFormSet()
    return context
```

성공하면 success_url로

3. View 설계

ch99₩photo₩views.py

```python
class AlbumPhotoUV(OwnerOnlyMixin, UpdateView):
    model = Album
    fields = ('name', 'description')
    success_url = reverse_lazy('photo:index')

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        if self.request.POST:
            context['formset'] = PhotoInlineFormSet(self.request.POST, self.request.FILES, instance=self.object)
        else:
            context['formset'] = PhotoInlineFormSet(instance=self.object)
        return context

    def form_valid(self, form):
        context = self.get_context_data()
        formset = context['formset']
        if formset.is_valid():
            self.object = form.save()
            formset.instance = self.object
            formset.save()
            return redirect(self.get_success_url())
        else:
            return self.render_to_response(self.get_context_data(form=form))
```

4. 템플릿 코딩

ch99₩mysite₩base.html

```
<li class="nav-item dropdown mx-1 btn btn-primary">
    <a class="nav-link dropdown-toggle text-white" href="#" data-toggle="dropdown">Util</a>
    <div class="dropdown-menu">
        <a class="dropdown-item" href="{% url 'admin:index' %}">Admin</a>
        <div class="dropdown-divider"></div>
        <a class="dropdown-item" href="{% url 'blog:post_archive' %}">Archive</a>
        <a class="dropdown-item" href="{% url 'blog:search' %}">Search</a>
    </div>
</li>

#하단 코드 수정
 <li class="nav-item dropdown mx-1 btn btn-primary">
    <a class="nav-link dropdown-toggle text-white" href="#" data-toggle="dropdown">Add</a>
    <div class="dropdown-menu">
        <a class="dropdown-item" href="{% url 'bookmark:add' %}">Bookmark</a>
        <a class="dropdown-item" href="{% url 'blog:add' %}">Post</a>
        <div class="dropdown-divider"></div>
        <a class="dropdown-item" href="{% url  'photo:album_add' %}">Album</a>
        <a class="dropdown-item" href="{% url  'photo:photo_add' %}">Photo</a>
    </div>
</li>

 <li class="nav-item dropdown mx-1 btn btn-primary">
    <a class="nav-link dropdown-toggle text-white" href="#" data-toggle="dropdown">Change</a>
    <div class="dropdown-menu">
        <a class="dropdown-item" href="{% url 'bookmark:change' %}">Bookmark</a>
        <a class="dropdown-item" href="{% url 'blog:change' %}">Post</a>
        <div class="dropdown-divider"></div>
        <a class="dropdown-item" href="{% url  'photo:album_change' %}">Album</a>
        <a class="dropdown-item" href="{% url  'photo:photo_change' %}">Photo</a>
    </div>
</li>
```

4. 템플릿 코딩

ch99₩photo₩templates₩photo₩photo_form.html

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

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

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

  {% if form.is_multipart %}  ①
  <form enctype="multipart/form-data" action="" method="post" class="card pt-3">
  {% else %}
  <form action="." method="post" class="card pt-3">
  {% endif %}
  {% csrf_token %}

    <div class="form-group row">
      {{ form.album|add_label_class:"col-form-label col-sm-2 ml-3 font-weight-bold" }}
      <div class="col-sm-2">
        {{ form.album|add_class:"form-control" }}
      </div>
      <div class="col-sm-2 my-auto">
        <a href="{% url 'photo:album_add' %}" class="btn btn-outline-primary btn-sm">
        Add Album</a>
      </div>
    </div>
```

```
    <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.image|add_label_class:"col-form-label col-sm-2 ml-3 font-weight-bold" }}
      <div class="col-sm-5">
        {{ form.image|add_class:"form-control-file" }}
      </div>
    </div>

    <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-8">
        {{ form.description|add_class:"form-control"|attr:"rows:3" }}
      </div>
    </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 %}
```

4. 템플릿 코딩
ch99₩photo₩templates₩photo₩photo_form.html

① is_multipart() 메소드는 폼이나 폼셋을 미리 체크해 multipart 인코딩이 필요한지 여부를 알려줍니다.
반환값이 True면 enctype=multipart/form-data로 지정해야 합니다. 이 예제에서는 폼에 이미지 필드가 있으므로 True를 반환합니다.
참고로 enctype 속성은 다음 표에서 보는 것처럼, 폼 데이터를 서버로 전송할 때 어떤 방식으로 데이터를 인코딩할 것인지 결정합니다.
POST 방식인 경우에만 사용하는 속성.

enctype 속성 값	인코딩 방식 설명
application/x-www-form-urlencoded	디폴트 값입니다. 빈칸은 +기호로 변환되고 $, # 등의 특수문자들도 아스키 16진수 값으로 변환됩니다.
multipart/form-data	데이터가 변환되지 않고 그대로 서버로 전송됩니다. 파일이나 이미지 등의 바이너리 파일을 업로드할 때 사용합니다.
text/plain	빈칸은 + 기호로 변환되지만 특수문자들은 변환되지 않습니다.

4. 템플릿 코딩

ch99₩photo₩templates₩photo₩photo_change_list.html

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

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

{% block content %}

    <h1>Photo Change - {{user}}</h1>

    <table class="table table-striped table-bordered table-condensed">

        <thead>
        <tr class="table-info">
            <th>Album</th>
            <th>Title</th>
            <th>Description</th>
            <th>Owner</th>
            <th>Update</th>
            <th>Delete</th>
        </tr>
        </thead>

        <tbody>
        {% for item in object_list %}
        <tr>
            <td>{{ item.album }}</td>
            <td>{{ item.title }}</td>
            <td>{{ item.description }}</td>
            <td>{{ item.owner }}</td>
            <td><a href="{% url 'photo:photo_update' item.id %}">Update</a></td>
            <td><a href="{% url 'photo:photo_delete' item.id %}">Delete</a></td>
        </tr>
        {% endfor %}
        </tbody>

    </table>

{% endblock %}
```

4. 템플릿 코딩

ch99₩photo₩templates₩

photo₩photo_confirm_delete.html

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

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

{% block content %}

    <h1>Photo 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 %}
```

4. 템플릿 코딩

ch99₩photo₩templates₩photo₩album_form.html

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

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

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

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

    {% if formset.errors %}
    <div class="alert alert-warning">
        <div class="font-weight-bold">Wrong! Please correct the formset error(s) below.</div>
        {% for formerrors in formset.errors %}
            {{ formerrors }}
        {% endfor %}
    </div>
    {% endif %}

    {% if form.is_multipart or formset.is_multipart %}
    <form enctype="multipart/form-data" action="" method="post">
    {% else %}
    <form action="." method="post">
    {% endif %}
    {% csrf_token %}
```

```
<h4>create or update album contents</h4>

<fieldset class="card pt-3">

    <div class="form-group row">
        {{ form.name|add_label_class:"col-form-label col-sm-3 ml-3 font-weight-bold" }}
        <div class="col-sm-5">
            {{ form.name|add_class:"form-control"|attr:"autofocus" }}
        </div>
    </div>

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

</fieldset>
<br>

<h4>create or update photo contents</h4>

<fieldset class="card pt-3">

    {{ formset.management_form }} ①
    {% for form in formset %}
        {{ form.id }}
        {# form.album #}
    <div class="form-group row">
        {{ form.image|add_label_class:"col-form-label col-sm-3 ml-3 font-weight-bold" }}
        <div class="col-sm-5">
            {{ form.image|add_class:"form-control-file" }}
        </div>
    </div>

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

4. 템플릿 코딩

ch99₩photo₩templates₩photo₩album_form.html

① 폼 셋에 들어 있는 각 폼을 다루는 경우 {{ formset.management_form }} 변수를 반드시 추가해야 함.
장고의 템플릿 엔진이 폼셋에 들어 있는 폼의 개수 등을 알 수 있어야 하기 때문

4. 템플릿 코딩

ch99₩photo₩templates₩photo₩album_form.html

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

        <hr>
        {% endfor %}

    </fieldset>

    <div class="form-group card py-1">
        <div class="offset-sm-3 col-sm-5">
            <input type="submit" value="Submit" class="btn btn-info"/>
        </div>
    </div>

</form>

{% endblock %}
```

4. 템플릿 코딩

ch99₩photo₩templates₩photo₩album_change_list.html

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

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

{% block content %}

    <h1>Album Change - {{user}}</h1>

    <table class="table table-striped table-bordered table-condensed">

        <thead>
        <tr class="table-success">
            <th>Name</th>
            <th>Description</th>
            <th>Owner</th>
            <th>Update</th>
            <th>Delete</th>
        </tr>
        </thead>

        <tbody>
        {% for item in object_list %}
        <tr>
            <td>{{ item.name }}</td>
            <td>{{ item.description }}</td>
            <td>{{ item.owner }}</td>
            <td><a href="{% url 'photo:album_update' item.id %}">Update</a></td>
            <td><a href="{% url 'photo:album_delete' item.id %}">Delete</a></td>
        </tr>
        {% endfor %}
        </tbody>

    </table>

{% endblock %}
```

4. 템플릿 코딩

ch99₩photo₩templates₩photo₩album_confirm_delete.html

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

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

{% block content %}

    <h1>Album Delete <small class="font-italic">(including related photos)</small></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 %}
```