**关系数据库管理系统**

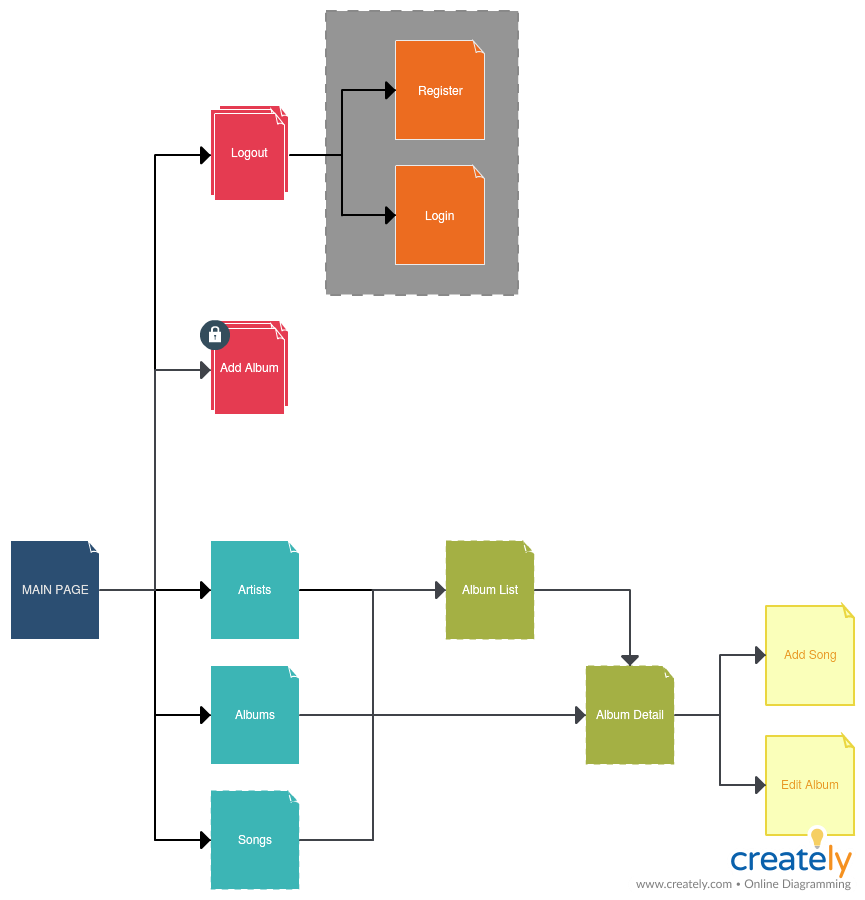
总结报告

周美廷-76066002

# 实现环境

此website用了Django（python）环境以及SQLite3实现的

# 系统功能结构图



# 基本表的定义，主外码等完整性约束定义，索引的定义

-----------用户-----------

CREATE TABLE auth\_user (

id INTEGER NOT NULL

PRIMARY KEY AUTOINCREMENT,

password VARCHAR (128) NOT NULL,

last\_login DATETIME,

is\_superuser BOOL NOT NULL,

username VARCHAR (150) NOT NULL

UNIQUE,

first\_name VARCHAR (30) NOT NULL,

email VARCHAR (254) NOT NULL,

is\_staff BOOL NOT NULL,

is\_active BOOL NOT NULL,

date\_joined DATETIME NOT NULL,

last\_name VARCHAR (150) NOT NULL

);

-----------用户权限-----------

CREATE TABLE auth\_permission (

id INTEGER NOT NULL

PRIMARY KEY AUTOINCREMENT,

content\_type\_id INTEGER NOT NULL

REFERENCES django\_content\_type (id) DEFERRABLE INITIALLY DEFERRED,

codename VARCHAR (100) NOT NULL,

name VARCHAR (255) NOT NULL

);

-----------专辑-----------

CREATE TABLE music\_album (

id INTEGER NOT NULL

PRIMARY KEY AUTOINCREMENT,

artist\_id INTEGER NOT NULL

REFERENCES music\_artist (id) DEFERRABLE INITIALLY DEFERRED,

album\_title VARCHAR (500) NOT NULL,

genre\_id INTEGER NOT NULL

REFERENCES music\_genre (id) DEFERRABLE INITIALLY DEFERRED,

album\_logo VARCHAR (1000) NOT NULL,

is\_favorite BOOL NOT NULL,

user\_id INTEGER NOT NULL

REFERENCES auth\_user (id) DEFERRABLE INITIALLY DEFERRED

);

-----------歌手-----------

CREATE TABLE music\_artist (

id INTEGER NOT NULL

PRIMARY KEY AUTOINCREMENT,

gender VARCHAR (100) NOT NULL,

artist\_name VARCHAR (250) NOT NULL

);

-----------类型-----------

CREATE TABLE music\_genre (

id INTEGER NOT NULL

PRIMARY KEY AUTOINCREMENT,

genre\_name VARCHAR (250) NOT NULL

);

-----------歌曲-----------

CREATE TABLE music\_song (

id INTEGER NOT NULL

PRIMARY KEY AUTOINCREMENT,

file\_type VARCHAR (10) NOT NULL,

song\_title VARCHAR (250) NOT NULL,

album\_id INTEGER NOT NULL

REFERENCES music\_album (id) DEFERRABLE INITIALLY DEFERRED,

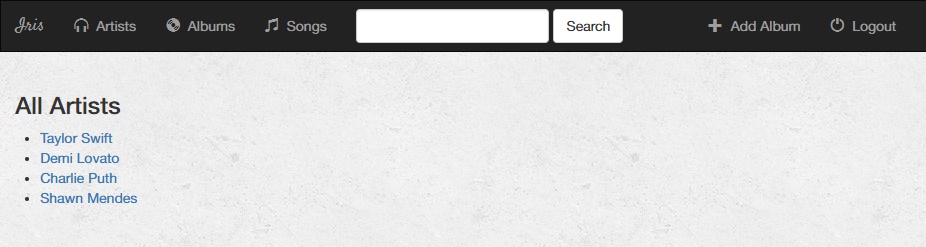
is\_favorite BOOL NOT NULL

);

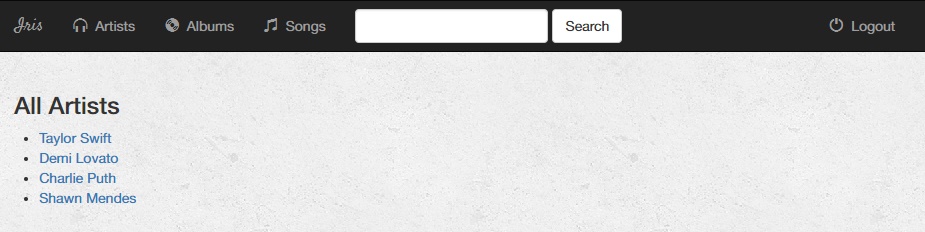
# 系统的安全性设计，不同人员的外模式及相关权限

管理员可添加新专辑而普通用户不可：

管理员：



普通用户：



# 存储过程、触发器和函数的代码说明

存储过程：

-- Table: auth\_user

CREATE TABLE "auth\_user" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "password" varchar(128) NOT NULL, "last\_login" datetime NULL, "is\_superuser" bool NOT NULL, "username" varchar(150) NOT NULL UNIQUE, "first\_name" varchar(30) NOT NULL, "email" varchar(254) NOT NULL, "is\_staff" bool NOT NULL, "is\_active" bool NOT NULL, "date\_joined" datetime NOT NULL, "last\_name" varchar(150) NOT NULL);

INSERT INTO auth\_user (id, password, last\_login, is\_superuser, username, first\_name, email, is\_staff, is\_active, date\_joined, last\_name) VALUES (1, 'pbkdf2\_sha256$120000$neaL9ghU0wqa$cEO1EIpftiIoo1OnOoysYDTi5bzQOaILUnocr4UgkFE=', '2018-12-27 15:16:32.652915', 1, 'admin', '', 'admin@example.com', 1, 1, '2018-12-20 18:13:05.381754', '');

INSERT INTO auth\_user (id, password, last\_login, is\_superuser, username, first\_name, email, is\_staff, is\_active, date\_joined, last\_name) VALUES (2, 'pbkdf2\_sha256$120000$U64H1e5P8gLs$e93iIAxlbvCZ1Ziv+/j6/KqoFtdeErEcmTR9hT3J8hg=', '2018-12-20 22:29:35.745367', 0, 'test', '', 'test@example.com', 0, 1, '2018-12-20 22:29:35.018195', '');

INSERT INTO auth\_user (id, password, last\_login, is\_superuser, username, first\_name, email, is\_staff, is\_active, date\_joined, last\_name) VALUES (3, 'pbkdf2\_sha256$120000$eGN4SCBM6cKZ$1F4X71QLh7LRn1ceu4/P3wGXE6ZYPSbLRa66GSZI8P4=', '2018-12-27 15:28:03.310200', 0, 'JavieraStephanie', '', 'javierastephanie@yahoo.com', 0, 1, '2018-12-21 04:35:14.510898', '');

-- Table: music\_album

CREATE TABLE "music\_album" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "artist\_id" integer NOT NULL REFERENCES "music\_artist" ("id") DEFERRABLE INITIALLY DEFERRED, "album\_title" varchar(500) NOT NULL, "genre\_id" integer NOT NULL REFERENCES "music\_genre" ("id") DEFERRABLE INITIALLY DEFERRED, "album\_logo" varchar(1000) NOT NULL, "is\_favorite" bool NOT NULL, "user\_id" integer NOT NULL REFERENCES "auth\_user" ("id") DEFERRABLE INITIALLY DEFERRED);

INSERT INTO music\_album (id, artist\_id, album\_title, genre\_id, album\_logo, is\_favorite, user\_id) VALUES (1, 1, 'Fearless', 1, 'https://upload.wikimedia.org/wikipedia/en/thumb/8/86/Taylor\_Swift\_-\_Fearless.png/220px-Taylor\_Swift\_-\_Fearless.png', 0, 1);

INSERT INTO music\_album (id, artist\_id, album\_title, genre\_id, album\_logo, is\_favorite, user\_id) VALUES (5, 1, '1989', 1, 'https://upload.wikimedia.org/wikipedia/en/thumb/f/f6/Taylor\_Swift\_-\_1989.png/220px-Taylor\_Swift\_-\_1989.png', 0, 1);

INSERT INTO music\_album (id, artist\_id, album\_title, genre\_id, album\_logo, is\_favorite, user\_id) VALUES (13, 2, 'Give Your Heart a Break', 1, 'https://upload.wikimedia.org/wikipedia/en/thumb/f/f6/Taylor\_Swift\_-\_1989.png/220px-Taylor\_Swift\_-\_1989.png', 0, 1);

INSERT INTO music\_album (id, artist\_id, album\_title, genre\_id, album\_logo, is\_favorite, user\_id) VALUES (14, 4, 'Stiches', 1, 'https://t2.genius.com/unsafe/220x220/https%3A%2F%2Fimages.rapgenius.com%2F3b4e501efa5af6642e9f15e1452b8026.600x600x1.jpg', 0, 1);

INSERT INTO music\_album (id, artist\_id, album\_title, genre\_id, album\_logo, is\_favorite, user\_id) VALUES (15, 1, 'Red', 1, 'https://upload.wikimedia.org/wikipedia/en/thumb/c/c0/Taylor\_Swift\_-\_Red\_%28Single%29.png/220px-Taylor\_Swift\_-\_Red\_%28Single%29.png', 0, 1);

INSERT INTO music\_album (id, artist\_id, album\_title, genre\_id, album\_logo, is\_favorite, user\_id) VALUES (17, 3, 'Nine Track Mind', 1, 'https://direct.rhapsody.com/imageserver/images/Alb.243008322/500x500.jpg', 0, 1);

-- Table: music\_artist

CREATE TABLE "music\_artist" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "gender" varchar(100) NOT NULL, "artist\_name" varchar(250) NOT NULL);

INSERT INTO music\_artist (id, gender, artist\_name) VALUES (1, 'Female', 'Taylor Swift');

INSERT INTO music\_artist (id, gender, artist\_name) VALUES (2, 'Female', 'Demi Lovato');

INSERT INTO music\_artist (id, gender, artist\_name) VALUES (3, 'Male', 'Charlie Puth');

INSERT INTO music\_artist (id, gender, artist\_name) VALUES (4, 'Male', 'Shawn Mendes');

-- Table: music\_genre

CREATE TABLE "music\_genre" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "genre\_name" varchar(250) NOT NULL);

INSERT INTO music\_genre (id, genre\_name) VALUES (1, 'Pop');

INSERT INTO music\_genre (id, genre\_name) VALUES (2, 'Classical');

INSERT INTO music\_genre (id, genre\_name) VALUES (3, 'Jazz');

INSERT INTO music\_genre (id, genre\_name) VALUES (4, 'Country');

-- Table: music\_song

CREATE TABLE "music\_song" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "file\_type" varchar(10) NOT NULL, "song\_title" varchar(250) NOT NULL, "album\_id" integer NOT NULL REFERENCES "music\_album" ("id") DEFERRABLE INITIALLY DEFERRED, "is\_favorite" bool NOT NULL);

INSERT INTO music\_song (id, file\_type, song\_title, album\_id, is\_favorite) VALUES (4, '', 'You Belong With Me', 1, 0);

INSERT INTO music\_song (id, file\_type, song\_title, album\_id, is\_favorite) VALUES (6, '', 'Give Your Heart A Break', 13, 0);

INSERT INTO music\_song (id, file\_type, song\_title, album\_id, is\_favorite) VALUES (7, '', 'Stitches', 14, 0);

INSERT INTO music\_song (id, file\_type, song\_title, album\_id, is\_favorite) VALUES (8, '', 'Love Story', 1, 0);

触发器：

**Insert**

def create\_album(request):

if not request.user.is\_authenticated:

return render(request, 'music/login.html')

else:

form = AlbumForm(request.POST or None)

if form.is\_valid():

album = form.save(commit=False)

album.user = request.user

album.save()

return render(request, 'music/albums.html')

context = {

"form": form,

}

return render(request, 'music/album\_form.html', context)

**Update**

class AlbumUpdate(UpdateView):

model = Album

fields = ['artist', 'album\_title', 'genre', 'album\_logo']

class BaseUpdateView(ModelFormMixin, ProcessFormView):

"""

Base view for updating an existing object.

Using this base class requires subclassing to provide a response mixin.

"""

def get(self, request, \*args, \*\*kwargs):

self.object = self.get\_object()

return super().get(request, \*args, \*\*kwargs)

def post(self, request, \*args, \*\*kwargs):

self.object = self.get\_object()

return super().post(request, \*args, \*\*kwargs)

class UpdateView(SingleObjectTemplateResponseMixin, BaseUpdateView):

"""View for updating an object, with a response rendered by a template."""

template\_name\_suffix = '\_form'

**Delete**

def delete\_album(request, artist\_id, album\_id):

artist = get\_object\_or\_404(Artist, pk=artist\_id)

album = Album.objects.get(pk=album\_id)

album.delete()

albums = Album.objects.filter(artist=artist)

return render(request, 'music/detail.html', {'artist': artist, 'albums': albums})

函数：

def index(request):

def detail(request, artist\_id):

def favorite\_album(request, artist\_id, album\_id):

def albumdetail(request, artist\_id, album\_id):

def create\_album(request):

def favorite(request, song\_id):

def create\_song(request, artist\_id, album\_id):

def delete\_album(request, artist\_id, album\_id):

def delete\_song(request, artist\_id, album\_id, song\_id):

def logout\_user(request):

def register(request):

def login\_user(request):

def albums(request, filter\_by):

def songs(request, filter\_by):

# 实现过程中主要技术论述

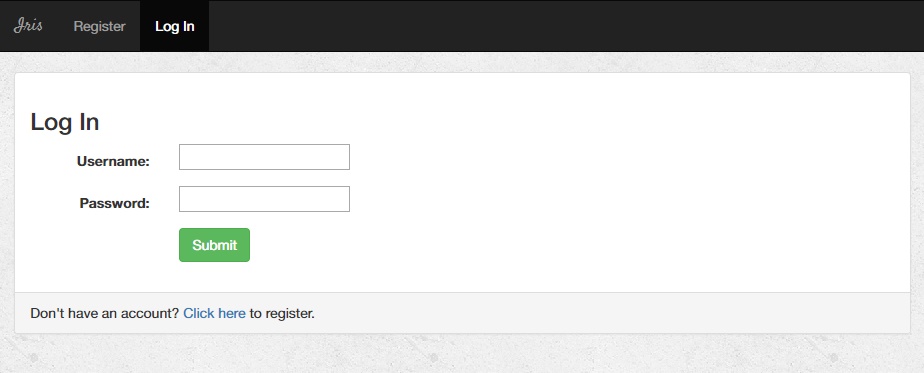
通过使用django，webapp和数据库实现变成很简单了。首先要做的是先把主url给列出来。一个website可有多个app。而在每个app里有自己的url。所以主website和app的url都要分开列出来。之后没有主website的什么事了。

App中，在这儿我只用一个app: music。把所有的实体以及属性都列出来。列出url时各个url有自己的页面（html页面），所以这html也要先写下来，后在url列表代码里与对应的url绑定。每个url也会调用对应的函数功能。比如首页调用列出歌手的函数。那么这些函数定义便是主要功能了。

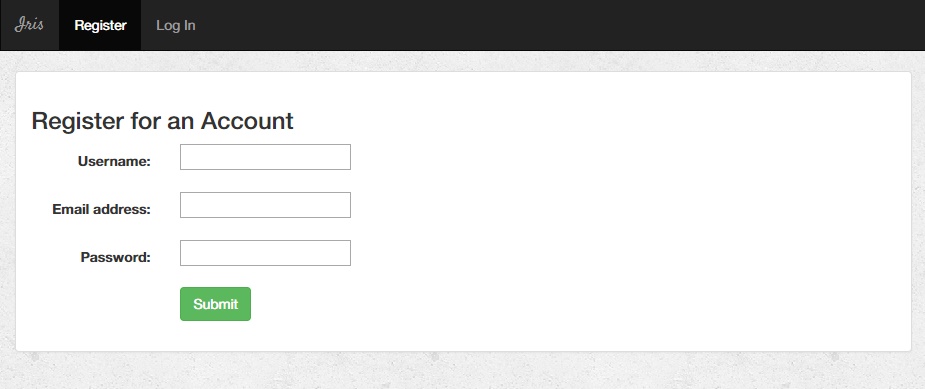
最主要的功能实现便是在views.py了。那边可以定义简单地流出所有实体内容，或者查找功能，或者注册登陆功能，删除功能等等。后在terminal中调用py manage.py runserver 8080执行代码（8080是选填，那是因为我的8000port无法使用）。

# 若干展示系统功能的运行实例

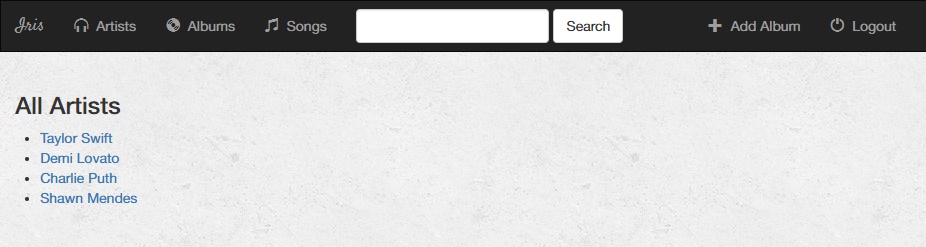
登陆：



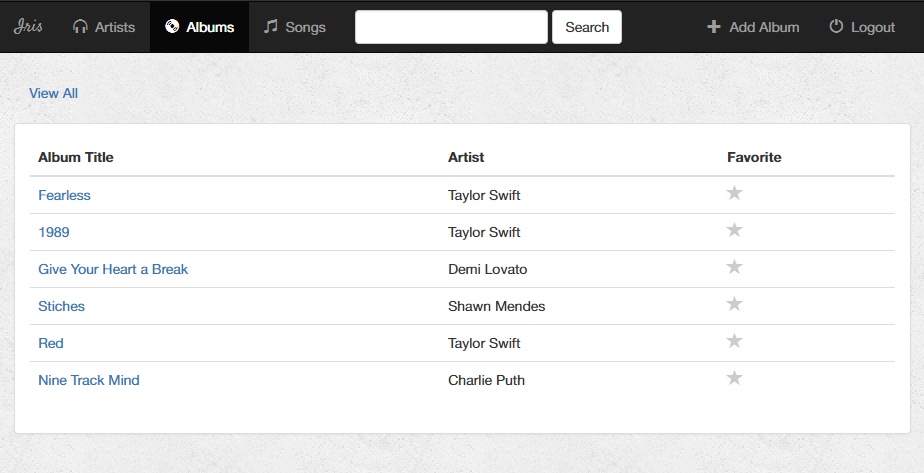
注册：



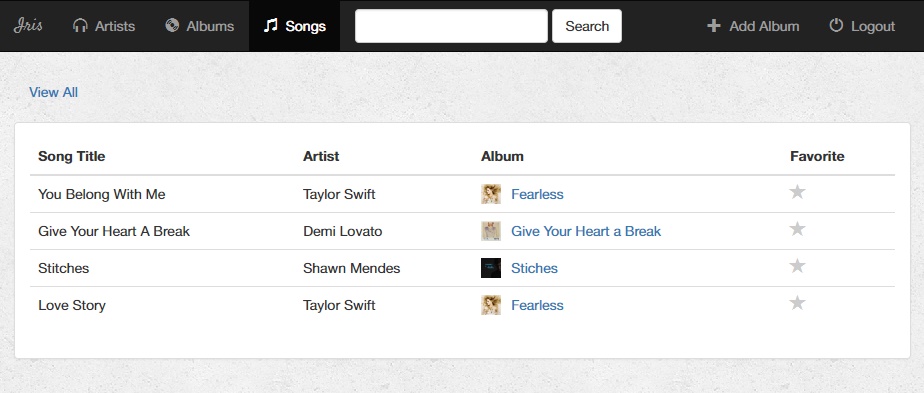
歌手列表（首页）：



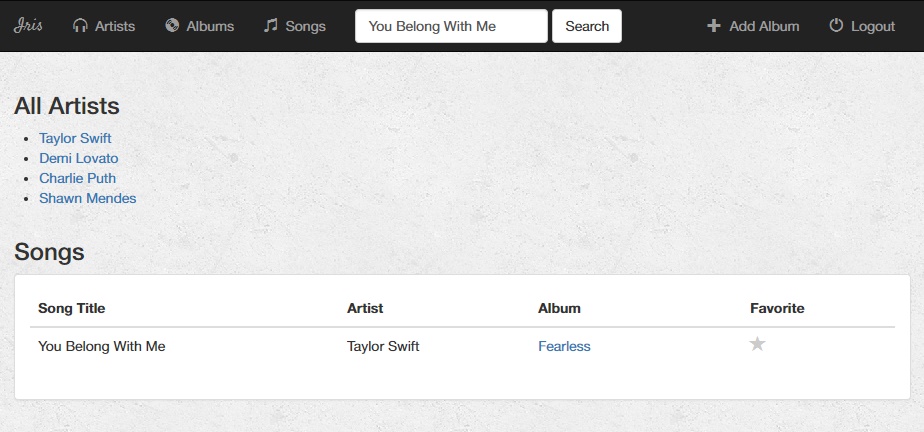
专辑列表（cross-artist）：



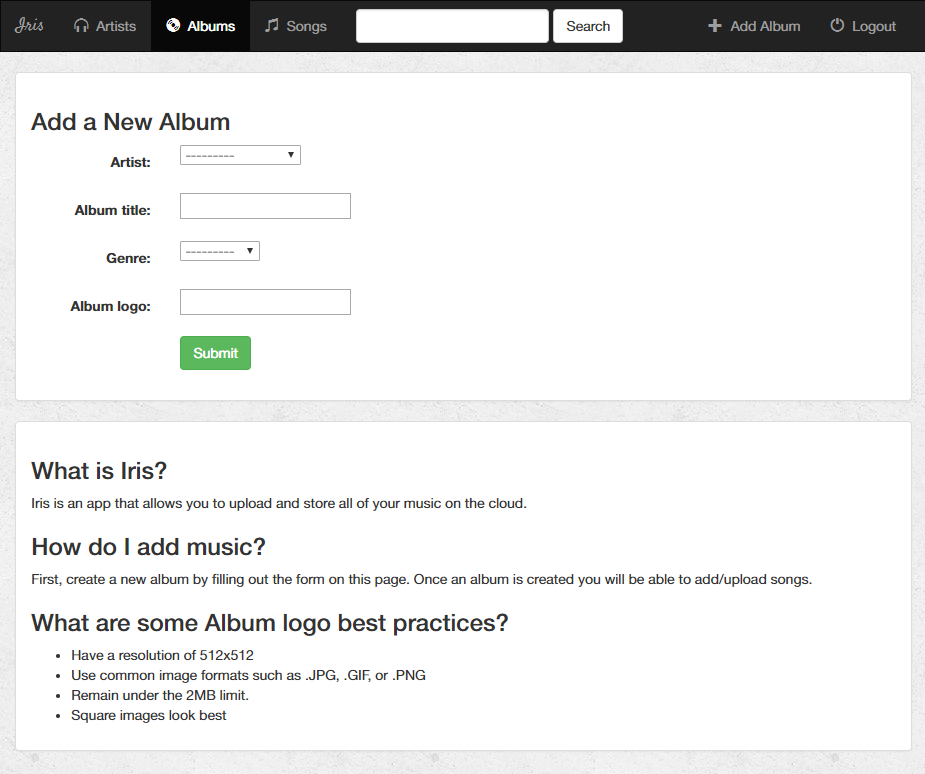
歌曲列表（cross-artist & cross-album）：



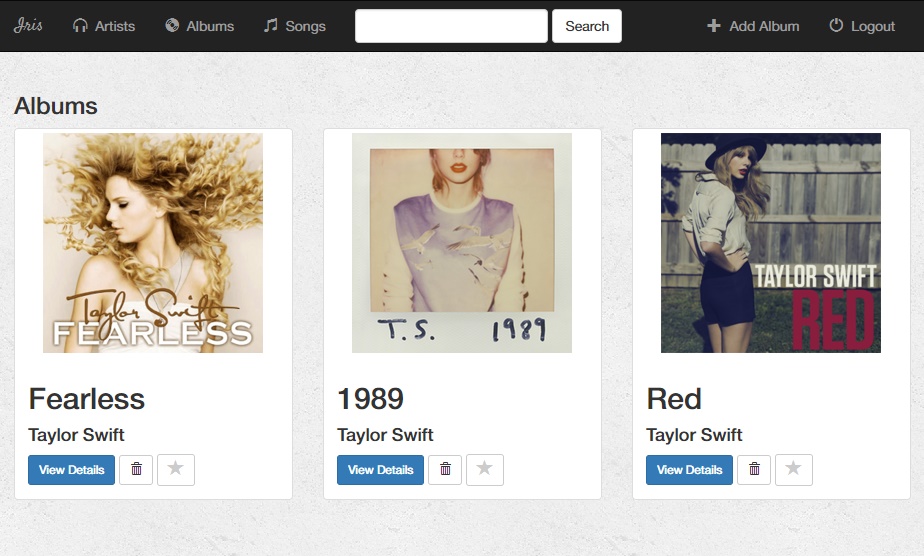
查找结果：



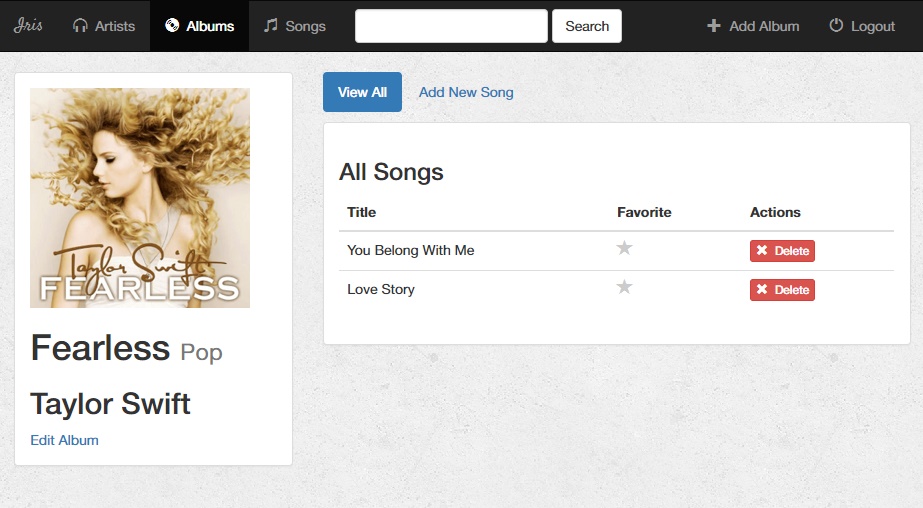
添加新专辑：



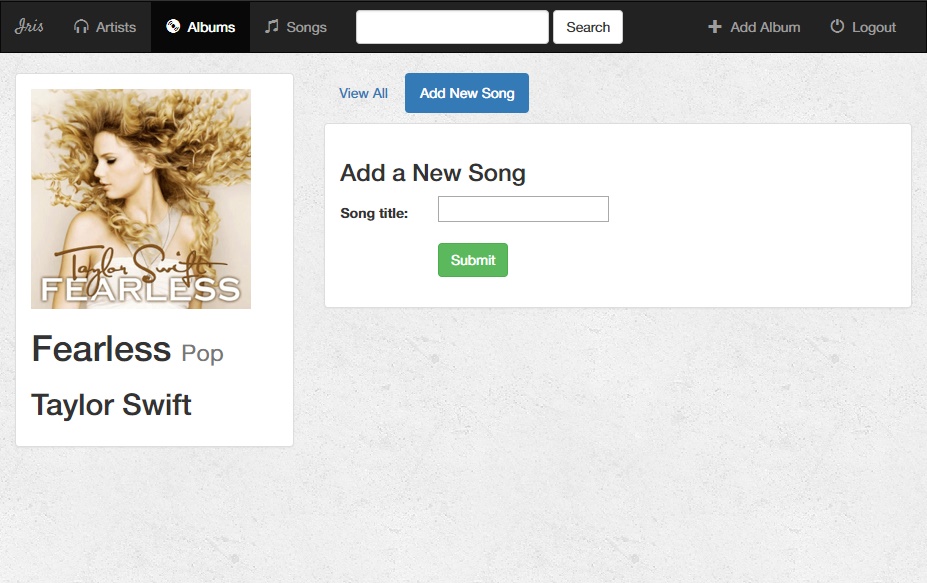
某个歌手专辑列表：



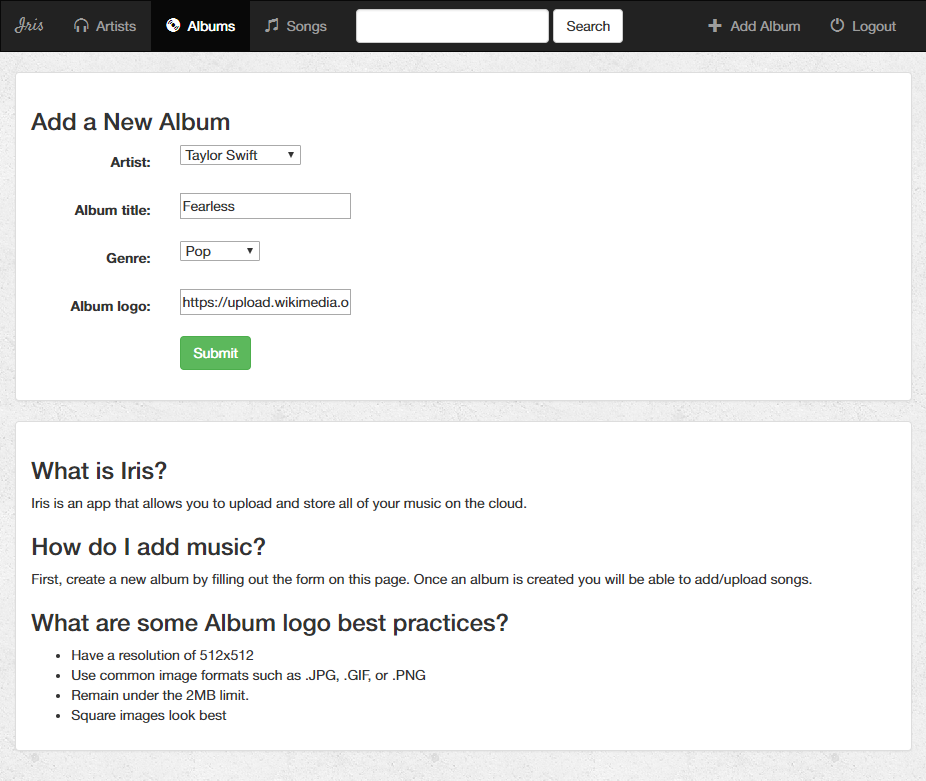
某个专辑歌曲列表：



添加新歌曲：



编辑专辑：



# 源程序简要说明

实体定义：models.py

from django.contrib.auth.models import Permission, User

from django.db import models

from django.urls import reverse

# Create your models here.

class Artist(models.Model):

artist\_name = models.CharField(max\_length=250)

gender = models.CharField(max\_length=100)

def \_\_str\_\_(self):

return self.artist\_name

class Genre(models.Model):

genre\_name = models.CharField(max\_length=250)

def \_\_str\_\_(self):

return self.genre\_name

class Album(models.Model):

user = models.ForeignKey(User, default=1, on\_delete=models.CASCADE)

artist = models.ForeignKey(Artist, on\_delete=models.CASCADE)

album\_title = models.CharField(max\_length=500)

genre = models.ForeignKey(Genre, on\_delete=models.CASCADE)

album\_logo = models.CharField(max\_length=1000)

is\_favorite = models.BooleanField(default=False)

def get\_absolute\_url(self):

return reverse('index')

def \_\_str\_\_(self):

return self.album\_title

class Song(models.Model):

album = models.ForeignKey(Album, on\_delete=models.CASCADE)

file\_type = models.CharField(max\_length=10)

song\_title = models.CharField(max\_length=250)

is\_favorite = models.BooleanField(default=False)

def \_\_str\_\_(self):

return self.song\_title

添加新数据的表格定义：forms.py

from django import forms

from django.contrib.auth.models import User

from .models import Album, Song

class AlbumForm(forms.ModelForm):

class Meta:

model = Album

fields = ['artist', 'album\_title', 'genre', 'album\_logo']

class SongForm(forms.ModelForm):

class Meta:

model = Song

fields = ['song\_title']

class UserForm(forms.ModelForm):

password = forms.CharField(widget=forms.PasswordInput)

class Meta:

model = User

fields = ['username', 'email', 'password']

实际功能实现（query）：views.py

from django.views import generic

from django.views.generic.edit import CreateView, UpdateView, DeleteView

from django.urls import reverse\_lazy

from django.http import JsonResponse

from django.shortcuts import render, get\_object\_or\_404

from django.contrib.auth import authenticate, login, logout

from django.db.models import Q

from .models import Artist, Album, Song

from .forms import AlbumForm, SongForm, UserForm

AUDIO\_FILE\_TYPES = ['wav', 'mp3', 'ogg']

IMAGE\_FILE\_TYPES = ['png', 'jpg', 'jpeg']

def index(request):

all\_artists = Artist.objects.all()

song\_results = Song.objects.all()

query = request.GET.get("q")

if query:

song\_results = song\_results.filter(

Q(song\_title\_\_icontains=query)

).distinct()

return render(request, 'music/index.html', {

'all\_artists': all\_artists,

'songs': song\_results,

})

else:

return render(request, 'music/index.html', {'all\_artists': all\_artists})

def detail(request, artist\_id):

if not request.user.is\_authenticated:

return render(request, 'music/login.html')

else:

user = request.user

artist = get\_object\_or\_404(Artist, pk=artist\_id)

albums = Album.objects.filter(artist=artist)

song\_results = Song.objects.all()

query = request.GET.get("q")

if query:

albums = albums.filter(

Q(album\_title\_\_icontains=query) |

Q(artist\_\_icontains=query)

).distinct()

song\_results = song\_results.filter(

Q(song\_title\_\_icontains=query)

).distinct()

return render(request, 'music/detail.html', {

'artist': artist,

'albums': albums,

'songs': song\_results,

})

else:

return render(request, 'music/detail.html', {'artist': artist, 'albums': albums, 'user': user})

def favorite\_album(request, artist\_id, album\_id):

artist = get\_object\_or\_404(Artist, pk=artist\_id)

album = get\_object\_or\_404(Album, pk=album\_id)

try:

if album.is\_favorite:

album.is\_favorite = False

else:

album.is\_favorite = True

album.save()

except (KeyError, Album.DoesNotExist):

return JsonResponse({'success': False})

else:

return JsonResponse({'success': True})

def albumdetail(request, artist\_id, album\_id):

artist = get\_object\_or\_404(Artist, pk=artist\_id)

album = get\_object\_or\_404(Album, pk=album\_id)

return render(request, 'music/albumdetail.html', {'artist': artist, 'album': album})

def create\_album(request):

if not request.user.is\_authenticated:

return render(request, 'music/login.html')

else:

form = AlbumForm(request.POST or None)

if form.is\_valid():

album = form.save(commit=False)

album.user = request.user

album.save()

return render(request, 'music/albums.html')

context = {

"form": form,

}

return render(request, 'music/album\_form.html', context)

class AlbumUpdate(UpdateView):

model = Album

fields = ['artist', 'album\_title', 'genre', 'album\_logo']

def favorite(request, song\_id):

song = get\_object\_or\_404(Song, pk=song\_id)

try:

if song.is\_favorite:

song.is\_favorite = False

else:

song.is\_favorite = True

song.save()

except (KeyError, Song.DoesNotExist):

return JsonResponse({'success': False})

else:

return JsonResponse({'success': True})

def create\_song(request, artist\_id, album\_id):

form = SongForm(request.POST or None)

artist = get\_object\_or\_404(Artist, pk=artist\_id)

album = get\_object\_or\_404(Album, pk=album\_id)

if form.is\_valid():

albums\_songs = album.song\_set.all()

for s in albums\_songs:

if s.song\_title == form.cleaned\_data.get("song\_title"):

context = {

'artist': artist,

'album': album,

'form': form,

'error\_message': 'You already added that song',

}

return render(request, 'music/create\_song.html', context)

song = form.save(commit=False)

song.album = album

song.save()

return render(request, 'music/detail.html', {'artist': artist, 'album': album})

context = {

'artist': artist,

'album': album,

'form': form,

}

return render(request, 'music/create\_song.html', context)

def delete\_album(request, artist\_id, album\_id):

artist = get\_object\_or\_404(Artist, pk=artist\_id)

album = Album.objects.get(pk=album\_id)

album.delete()

albums = Album.objects.filter(artist=artist)

return render(request, 'music/detail.html', {'artist': artist, 'albums': albums})

def delete\_song(request, artist\_id, album\_id, song\_id):

artist = get\_object\_or\_404(Artist, pk=artist\_id)

album = get\_object\_or\_404(Album, pk=album\_id)

song = Song.objects.get(pk=song\_id)

song.delete()

return render(request, 'music/albumdetail.html', {'artist': artist, 'album': album})

def logout\_user(request):

logout(request)

form = UserForm(request.POST or None)

context = {

"form": form,

}

return render(request, 'music/login.html', context)

def register(request):

form = UserForm(request.POST or None)

if form.is\_valid():

user = form.save(commit=False)

username = form.cleaned\_data['username']

password = form.cleaned\_data['password']

user.set\_password(password)

user.save()

user = authenticate(username=username, password=password)

if user is not None:

if user.is\_active:

login(request, user)

albums = Album.objects.filter(user=request.user)

return render(request, 'music/index.html', {'albums': albums})

context = {

"form": form,

}

return render(request, 'music/registration\_form.html', context)

def login\_user(request):

if request.method == "POST":

username = request.POST['username']

password = request.POST['password']

user = authenticate(username=username, password=password)

if user is not None:

if user.is\_active:

login(request, user)

albums = Album.objects.filter(user=request.user)

return render(request, 'music/detail.html', {'albums': albums})

else:

return render(request, 'music/login.html', {'error\_message': 'Your account has been disabled'})

else:

return render(request, 'music/login.html', {'error\_message': 'Invalid login'})

return render(request, 'music/login.html')

def albums(request, filter\_by):

album\_list = Album.objects.all()

return render(request, 'music/albums.html', {'album\_list': album\_list})

def songs(request, filter\_by):

users\_songs = Song.objects.all()

return render(request, 'music/songs.html', {'song\_list': users\_songs})

# 收获和体会

通过这次课设大作业我才能真正的从头建立一个手写并能用数据库存储的webapp，收获很多。日后将能做更多的webapp啦。