实验报告

一、网页系统功能

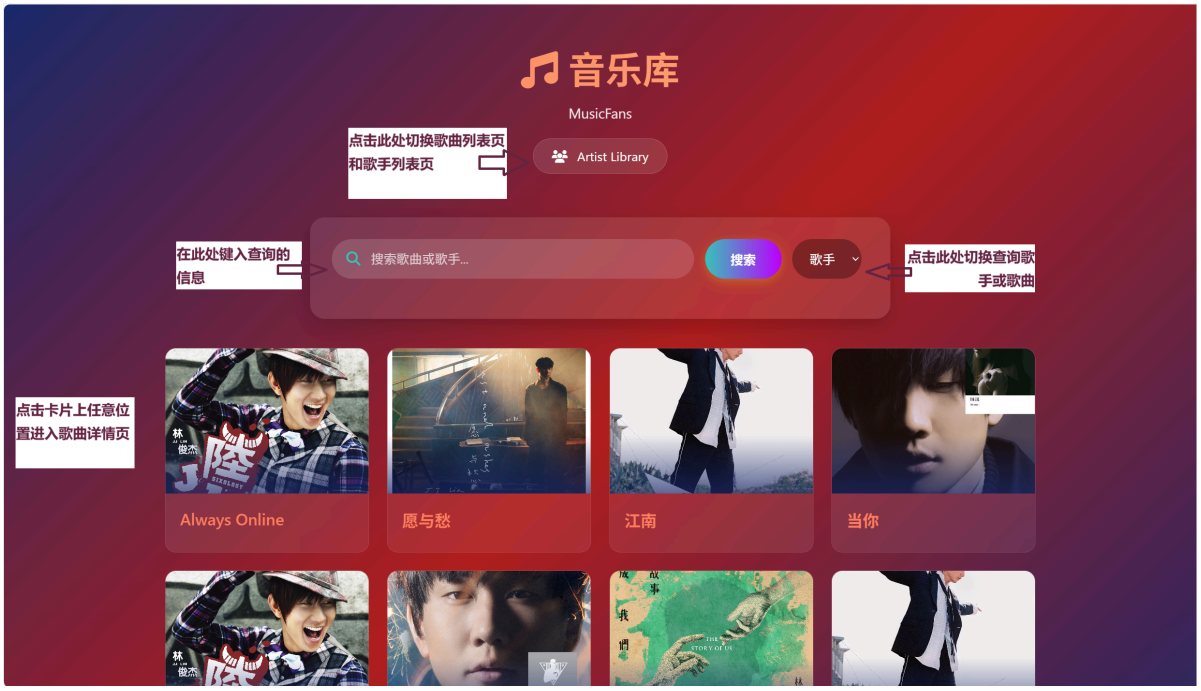
(i) class

```python
from django.db import models
from django.utils import timezone


class Song(models.Model):
    num = models.IntegerField()
    name = models.CharField()
    artist = models.CharField()
    lyric = models.CharField()
    url = models.CharField()
    pic_url = models.CharField()
    artist_url = models.CharField()


class Singer(models.Model):
    num = models.IntegerField()
    name = models.CharField()
    pic_url = models.CharField()
    desc = models.CharField()
    url = models.CharField()


class Comment(models.Model):
    num = models.IntegerField()
    context = models.CharField()
    created_at = models.DateTimeField(default=timezone.now)

```

这是在django中models.py定义的数据类型。歌曲类成员变量有num(编号)，name(歌曲名)，artist(歌手名)，lyric(歌词)，url(原网站链接)，pic_url(歌曲图片url)，artist_url(歌手图片的url)。歌手类成员变量有num(编号)，name(歌手名)，pic_url(图片的url)，url(原网站链接)，desc(歌手简介)。评论类成员变量有num(对应歌曲的编号)，context(正文)，created_at(时间)。

(ii)页面

1.主页（歌曲列表页）



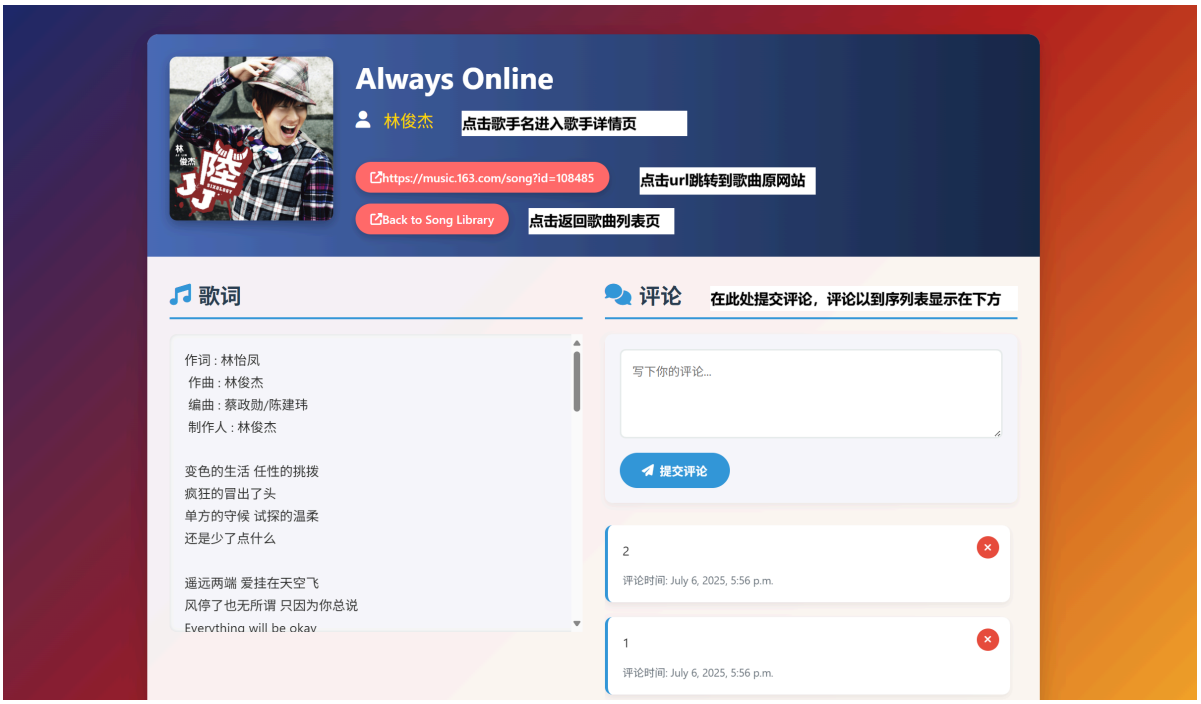
该页面为歌曲列表页。点击“Artist Library”可以切换到歌手列表页。此页面上具有搜索功能，输入非空搜索内容后点击搜索之后的画面就是搜索结果页。可以从此页面跳转到歌手列表页和歌曲详情页。分页跳转功能支持输入页数和按键增加减少两种方式。背景为45度的渐变颜色，搜索按钮颜色为初音未来色和清华紫的渐变。图标样式来自于font-awesome。

```
def show_mainpage(request):
    """歌曲详情页

    每页显示8首歌曲,如果用户输入的页数不是整数就跳转到第一页，如果超出范围则跳转到最近的页数
    """
    song_list = Song.objects.all()
    paginator = Paginator(song_list, 8)

    page = request.GET.get("page")
    try:
        songs = paginator.page(page)
    except PageNotAnInteger:
```

```
        songs = paginator.page(1)
    except EmptyPage:
        songs = paginator.page(paginator.num_pages)

    base_url = "/index/MUSIC"
    return render(
        request,
        "index.html",
        {"songs": songs, "base_url": base_url, "MEDIA_URL":
settings.MEDIA_URL},
    )
```

获得所有歌曲，并进行分页，解析index.html并传入一些参数，base_url为跳转服务，MEDIA_URL为本地资源的文件夹

2、歌曲详情页

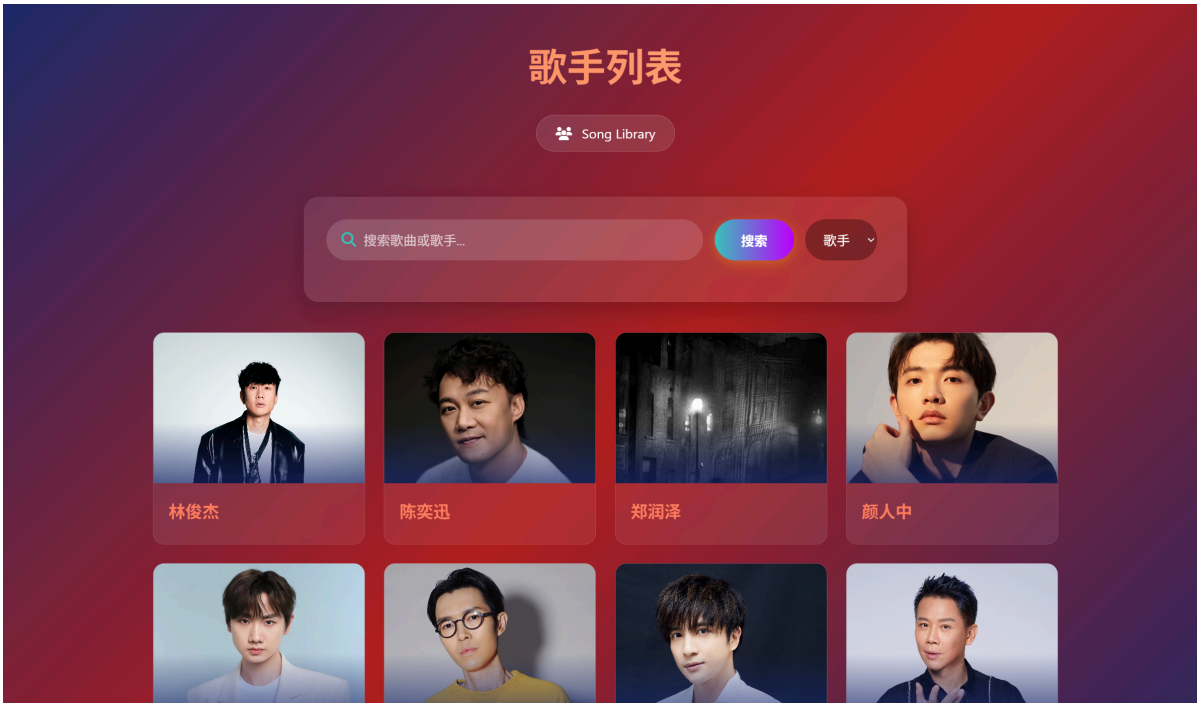


此页面点击歌手名可以跳转到歌手详情页，点击url跳转到原网站，点击Back返回歌曲列表页。评论提交内容不能为空。

```
def show_song(request, id):
    """展示歌曲信息,在数据库中滤出这首歌的所有评论数据"""
    song = Song.objects.get(id=id)
    artist = Singer.objects.get(name=song.artist)
    artistid = artist.id
    comments = Comment.objects.filter(num=id)

    return render(
        request,
        "index1.html",
        {
            "song": song,
            "MEDIA_URL": settings.MEDIA_URL,
            "artistid": artistid,
            "comments": comments[::-1],
        },
    )
```

获取这一首歌曲，及对应的评论，转换为倒序列表作为参数，解析index1.html

3、歌手列表页



页面功能和歌曲列表页一样

```
def show_artistlist(request):
    """歌手列表页

    每页显示8位歌手,如果用户输入的页数不是整数就跳转到第一页，如果超出范围则跳转到最近的页数
    """
    singer_list = Singer.objects.all()
```

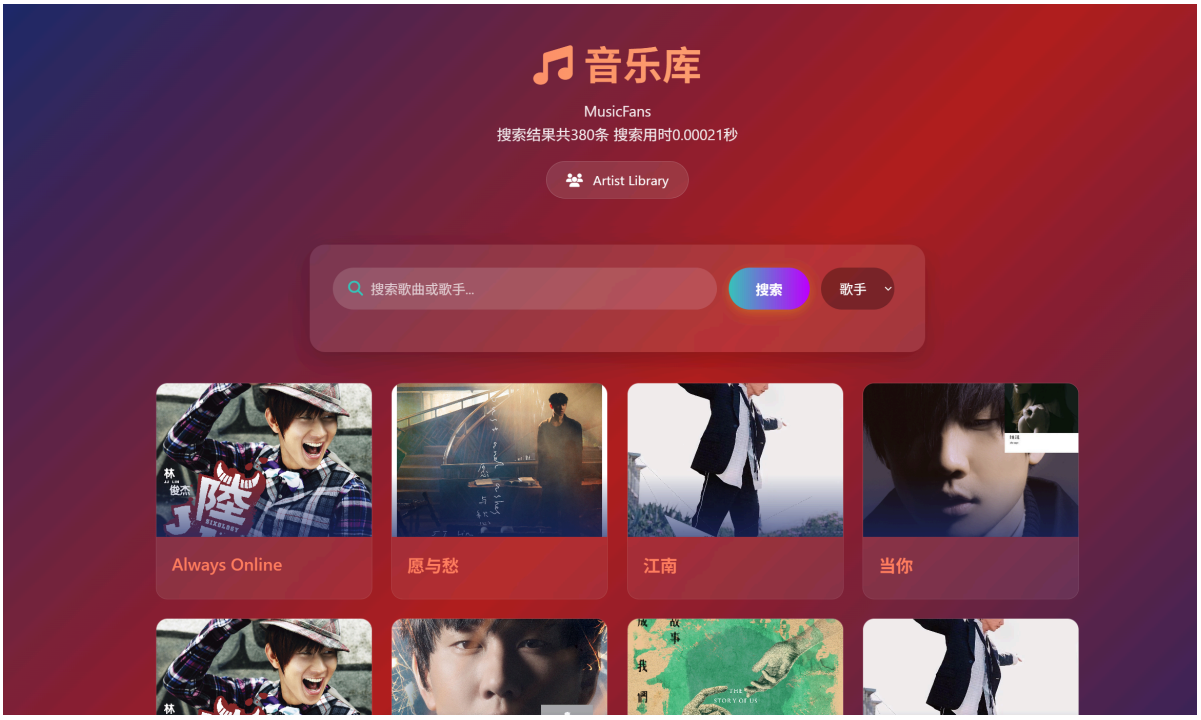
```
    paginator = Paginator(singer_list, 8)

    page = request.GET.get("page")
    try:
        singers = paginator.page(page)
    except PageNotAnInteger:
        singers = paginator.page(1)
    except EmptyPage:
        singers = paginator.page(paginator.num_pages)

    base_url = "/index/MUSIC/artist"

    return render(
        request,
        "index2.html",
        {"singers": singers, "base_url": base_url, "MEDIA_URL":
settings.MEDIA_URL},
    )
```

4、歌手详情页

点击url跳转到原歌手网页，点击Back返回歌手列表页。点击歌曲封面和歌曲名可以跳转到歌曲列表页

```python
def show_singer(request, id):
    """展示歌手信息,在数据库中滤出歌手的歌曲"""
    singer = Singer.objects.get(id=id)
    song_list = Song.objects.filter(artist=singer.name)

    return render(
        request,
        "index3.html",
        {"singer": singer, "MEDIA_URL": settings.MEDIA_URL, "song_list": song_list},
    )
```

5、搜索结果页



与歌曲列表页和歌手列表页一致，仅增加了搜索结果与搜索时间。点击对应卡片可以跳转到详情页

```python
def search(request):
    """搜索

    搜索时间为filter的查询时间
    通过传进来的type参数，进行不同的搜索，并且解析不同的html页面
    """
    query = request.GET.get("q")
    search_type = request.GET.get("type")
    if search_type == "artist":
        start_time = time.time()
        singer_list = Singer.objects.filter(
            Q(name__contains=query) | Q(desc__contains=query)
        )
        end_time = time.time()
        lens = len(singer_list)

```

```
        paginator = Paginator(singer_list, 8)

        page = request.GET.get("page")
        try:
            singers = paginator.page(page)
        except PageNotAnInteger:
            singers = paginator.page(1)
        except EmptyPage:
            singers = paginator.page(paginator.num_pages)

        base_url = "/index/search?type=artist&q=" + query

        return render(
            request,
            "index4.html",
            {
                "singers": singers,
                "base_url": base_url,
                "MEDIA_URL": settings.MEDIA_URL,
                "query": query,
                "time": "{:.5f}".format(end_time - start_time),
                "len": lens,
            },
        )

    if search_type == "song":
        start_time = time.time()
        song_list = Song.objects.filter(
            Q(name__contains=query)
            | Q(artist__contains=query)
            | Q(lyric__contains=query)
        )
        end_time = time.time()
        lens = len(song_list)
        paginator = Paginator(song_list, 8)

        page = request.GET.get("page")
        try:
            songs = paginator.page(page)
        except PageNotAnInteger:
            songs = paginator.page(1)
        except EmptyPage:
            songs = paginator.page(paginator.num_pages)

        base_url = "/index/search?type=song&q=" + query

        return render(
            request,
            "index5.html",
            {
                "songs": songs,
                "base_url": base_url,
                "MEDIA_URL": settings.MEDIA_URL,
                "query": query,
                "time": "{:.5f}".format(end_time - start_time),
                "len": lens,
```

```
            },
        )
```

通过搜索页面传进来的类型参数在数据库中搜索结果，搜索时间认为是filter函数的时间，保留5位小数，len为结果的数量

二、数据量

歌曲：2026首；歌手100位

三、使用的技术与算法

（1）爬虫技术

```
def get_info():
    """爬虫的主函数

    将爬取的内容写在指定文件中
    """
    singercsvfile = open(
        r"C:\Users\14395\Desktop\git\MusicInfo\singer.csv", "a", errors="ignore"
    )
    songcsvfile = open(
        r"C:\Users\14395\Desktop\git\MusicInfo\song.csv", "a", errors="ignore"
    )

    singerwriter = csv.writer(singercsvfile)
    songwriter = csv.writer(songcsvfile)

    singerwriter.writerow(
        ("artist_id", "artist_name", "artist_desc", "artist_url", "artist_pic_url")
    )

    songwriter.writerow(
        ("song_id", "song_name", "artist_name", "song_url", "lyric", "song_pic")
    )

    # ls1 = [1001, 1002, 1003, 2001, 2002, 2003, 6001, 6002, 6003, 7001, 7002, 7003, 4001, 4002, 4003]    # id的值
    ls1 = [1001]
    # ls2 = [-1, 0, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90] # initial的值
    ls2 = [-1]

    # 网易云歌手主页面，这里ls1和ls2只有一项是为了只爬取一页的100位歌手

    for i in ls1:
        for j in ls2:
            url = (
                "http://music.163.com/discover/artist/cat?id="
                + str(i)
```

```
                + "&initial="
                + str(j)
            )

            response = requests.get(url, headers=headers)
            response.encoding = "utf-8"
            soup = BeautifulSoup(response.text, "lxml")  # soup为热门华语男歌手页面

            list = soup.find_all("a", attrs={"class": "nm nm-icn f-thide s-fc0"})

            for artist in list:  # 可以list[a:b]选定范围，多次爬虫，避免被反爬
                artist_name = artist.string
                artist_id = artist["href"].replace("/artist?id=", "").strip()

                url1 = "https://music.163.com/artist/desc?id=" + artist_id
                response1 = requests.get(url1, headers=headers)
                response1.encoding = "utf-8"
                soup1 = BeautifulSoup(response1.text, "lxml")  # soup1为某一歌手介绍主页
                div = soup1.find("div", class_="n-artdesc")  # 按class查找
                p = div.find("p")  # 找第一个段落
                artist_desc = p.get_text(strip=False).strip()

                url2 = "https://music.163.com/artist/?id=" + artist_id
                response2 = requests.get(url2, headers=headers)
                response2.encoding = "utf-8"
                soup2 = BeautifulSoup(response2.text, "lxml")  # soup2为某一歌手主页
                artist_pic = soup2.find("meta", attrs={"property": "og:image"})[
                    "content"
                ]

                download_img(
                    artist_pic,
                    r"C:\Users\14395\Desktop\git\MusicInfo\singerpics",
                    str(artist_id) + ".jpg",
                )

                singerwriter.writerow(
                    (artist_id, artist_name, artist_desc, url1, artist_pic)
                )

                song_list_ul = soup2.find("ul", class_="f-hide")

                for item in song_list_ul.find_all(
                    "li"
                ):  # 可以list[a:b]选定范围，多次爬虫，避免被反爬
                    song_link = item.find("a")
                    song_href = song_link.get("href", "")
                    song_id = re.search(r"id=(\d+)", song_href)
                    song_id = song_id.group(1) if song_id else ""
                    song_name = song_link.get_text(strip=True)
```

```
                    song_url = "https://music.163.com/song?id=" + song_id
                    response3 = requests.get(song_url, headers=headers)
                    soup3 = BeautifulSoup(response3.text, "lxml")  # soup3为某一歌曲页面

                    song_pic = soup3.find("meta", attrs={"property": "og:image"})[
                        "content"
                    ]

                    lyric = get_lyric(song_id)

                    songwriter.writerow(
                        (song_id, song_name, artist_name, song_url, lyric, song_pic)
                    )

                time.sleep(random.uniform(0.3, 0.7))


get_info()

# 直接运行这个函数可能会因为网易云严格的反爬措施而报错，建议在一些地方调整参数，分多次爬取或者每次爬取一部分内容
```

访问网易云网站发现歌手榜单的格式，通过设置id和initial获取热门华语男歌手列表（现在打开热门华语男歌手列表其中的歌手会有不同）。搜索标签获得歌手列表，提取出歌手名和歌手id，然后用其访问歌手描述的网页，获得歌手简介。并继续用歌手id访问歌手页面，提取歌手图片url。用这个url下载歌手图片。将这些所有数据写入singer.csv中。在歌手页面获取其所有的歌曲，获取歌曲名和歌曲id。进入歌曲页面获取歌曲图片的url。将这些所有数据写入song.csv中。

（2）下载歌曲图片

```
import csv
import requests
import os

# 下载图片


def download_songimg():
    """爬取歌曲的图片,由于歌曲数量较多,为了避免被反爬虫,爬下一部分数据之后再下载图片"""
    with open(r"C:\Users\14395\Desktop\git\MusicInfo\song.csv", mode="r") as csv_file:
        csv_reader = csv.DictReader(csv_file)
        for row_num, row in enumerate(csv_reader):
            if row_num % 2 == 0:  # csv中隔行有数据
                continue
            image_url = row["song_pic"].strip()
            image_id = row["song_id"].strip()
            save_path = os.path.join(
                r"C:\Users\14395\Desktop\git\MusicInfo\songpics", image_id + ".jpg"
            )
```

```
            response = requests.get(image_url, headers=headers)
            with open(save_path, "wb") as f:
                for chunk in response.iter_content(1024):
                    f.write(chunk)


download_songimg()
```

从song.csv中获取歌曲图片url并下载

(3)使用songimporter.py和singerimporter.py向数据库中导入数据

```
import os
import sys
import csv
import django


BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
sys.path.append(BASE_DIR)
os.environ.setdefault("DJANGO_SETTINGS_MODULE", "project1.settings")
django.setup()

from MUSIC.models import Singer


def func():
    """将歌手信息导入数据库"""

    singerfile = open(r"C:\Users\14395\Desktop\git\MusicInfo\singer.csv",
"r")

    singreader = csv.DictReader(singerfile)
    for row_num, row in enumerate(singreader):
        if row_num % 2 == 0:
            continue

        singer = Singer(
            num=row["artist_id"],
            name=row["artist_name"],
            pic_url=row["artist_pic_url"],
            desc=row["artist_desc"],
            url=row["artist_url"],
        )

        singer.save()


func()
print(f"成功导入 {Singer.objects.count()} 个歌手")

```

```
import os
import sys
```

```
import csv
import django


BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
sys.path.append(BASE_DIR)
os.environ.setdefault("DJANGO_SETTINGS_MODULE", "project1.settings")
django.setup()

from MUSIC.models import Song


def func():
    """将歌曲信息导入数据库"""
    csvfile = open(r"C:\Users\14395\Desktop\git\MusicInfo\song.csv", "r")
    singerfile = open(r"C:\Users\14395\Desktop\git\MusicInfo\singer.csv",
"r")
    reader = csv.DictReader(csvfile)
    singreader = csv.DictReader(singerfile)
    for row_num, row in enumerate(reader):
        if row_num % 2 == 0:
            continue

        tmp = row["artist_name"]
        singer_url = ""
        for hang in singreader:
            if hang["artist_name"] == tmp:
                singer_url = hang["artist_url"]
                break

        song = Song(
            num=row["song_id"],
            name=row["song_name"],
            artist=row["artist_name"],
            lyric=row["lyric"],
            url=row["song_url"],
            pic_url=row["song_pic"],
            artist_url=singer_url,
        )
        song.save()


func()
print(f"成功导入 {Song.objects.count()} 首歌曲")
```

(4)评论采用网页重定向

```
def comment(request, id):
    """评论函数"""
    content = request.POST.get("content")
    obj = Comment(num=id, context=content)
    obj.full_clean()  # 进行数据验证
    obj.save()
    return HttpResponseRedirect(f"/index/MUSIC/{id}")

```

```
def delcomment(request, id):
    """删除评论"""
    iden = request.POST.get("delete_comment")
    comment = Comment.objects.get(id=iden)
    comment.delete()
    return HttpResponseRedirect(f"/index/MUSIC/{id}")
```

(5) urls.py

```
from django.urls import path, include
import MUSIC.views as views

urlpatterns = [
    path("MUSIC/<int:id>", views.show_song),
    path("MUSIC", views.show_mainpage),
    path("MUSIC/artist", views.show_artistlist),
    path("MUSIC/artist/<int:id>", views.show_singer),
    path("comment/<int:id>", views.comment),
    path("delcomment/<int:id>", views.delcomment),
    path("search", views.search),
]
```

切换界面就重新进入新的网址，调用对应的函数渲染不同的html

(6) 利用jieba库处理歌词

```
def segment_text(text):
    """由于歌曲的信息各不相同，所以前面处理歌词的函数会有遗漏。手动根据词云结果添加停用词

    Keyword arguments:
    text -- 字符串

    返回处理后的字符串
    """
    stopwords = [
        "母带",
        "h3R3",
        "总监",
        "录音",
        "制作",
        "录音师",
        "作词",
        "出品人",
        "网易",
        "法老",
        "陶喆",
        "Producer",
        "编曲",
        "SBMS",
        "配唱",
        "版权",
        "国际",
        "弦乐",
```

```
        "贝斯",
        "Studio21A",
        "统筹",
        "录音室",
        "编写",
        "键盘",
        "NEWBAND",
        "公司",
        "音频编辑",
        "钢琴",
        "维伴",
        "首席",
        "乐队",
        "音响",
        "原唱",
        "编曲",
        "作曲",
        "Ltd",
        "打击乐",
        "Sound",
        "rapper",
        "Music",
        "混音",
        "牛班",
        "吉他",
        "音乐",
        "Studio",
        "Engineer",
        "Publishing",
        "何飚",
        "don",
        "PGM",
        "Vocal",
        "有限公司",
        "工作室",
        "林俊杰",
        "队长",
        "Asen",
        "罗言",
        "re",
        "乐团",
        "人声",
        "企划",
        "MUSIC",
        "项目",
        "Wiz",
        "OP",
        "录音棚",
        "汪苏",
        "郎梓朔",
        "营销",
        "发行",
        "Program",
        "编辑",
        "石行",
        "改编",
```

```
            "工程师",
            "爱乐乐团",
            "监制",
            "Mixing",
            "说唱",
            "SP",
            "Mastering",
            "Chan",
            "张子",
            "陈楚生",
            "刘卓",
            "索尼",
            "林梦洋",
            "设计",
        ]
        words = jieba.cut(text)
        return " ".join([word for word in words if word not in stopwords and len(word) > 1])
```

要求词不在停用词中且长度大于1，为了避免一些无意义的词如（“的”，“地”，"我"）

(7)利用adjustText库处理散点图的名称标签，避免重合

```
texts = []
for i, artist in enumerate(artists):
    texts.append(plt.text(x.iloc[i], y.iloc[i], artist, fontsize=6, alpha=1))


adjust_text(
    texts,
    expand_points=(2, 2),
    expand_text=(1.2, 1.2),
    force_text=(0.5, 0.5),
    only_move={"points": "y", "text": "y"},
)
```

expand_points扩大点的范围，标签不会出现在这一范围内。expand_text扩大标签的范围，避免重叠。force_text为标签之间互相排斥的强度。only_move表明点和标签只能在y方向移动。

（8）数据分析使用的相关代码见数据分析报告

(9)其他技术：pandas，matplotlib，re，requests，BeautifulSoup等技术

四、实验用时与感想

(1)爬虫且完整获得数据：36h

(2)网页系统设计：36h

(3)数据分析：24h

感想：这是第一个我完成的计算机技术工程，颇有成就感。从一开始畏惧网页复杂的html源码，到熟练的找到所需内容，从一开始总是写成c++到掌握python的基础语法，第一次写html构建网站都让我收获良多。总的来说，这次实验的内容非常不错，让我掌握了许多重要的技术和技能。任务量有一点小大:(