网站地址：http://139.196.15.209/

中山大學
SUN YAT-SEN UNIVERSITY

《数据库系统原理》

期末大作业报告

学 院 名 称 ： 数据科学与计算机学院

专业（班级）： 17 计算机科学与技术

姓名学号 ：

17341067 江金昱

17341072 赖少玄

17341063 黄镇

17341073 蓝靖瑜

目录

一、小组人员及分工

姓名	分工
江金昱（组长）	后端、前端、报告撰写
黄镇	云服务器部署、报告撰写
赖少玄	数据库、后端、报告撰写
蓝靖瑜	需求分析、网站功能测试

二、应用需求介绍：

如今音乐已经成为了人们生活必不可分的一部分，各大音乐流派层出不穷，每个人都能找到属于自己喜欢的音乐类型。而如此繁杂的音乐体系，需要进行良好的管理。因此，本次我们选择实现“音乐管理系统“，旨在实现音乐数据的增删改查等操作，并紧密结合数据库，建立了可视化的 web 网页，实现了不同层次的操作。需求如下：

1. 实现音乐数据的方便的增加、删除、修改操作。

2. 搜索功能。当用户输入关键字的时候，如歌手名字，可输出相关信息

3. 可视化的 web 网页。通过主界面可进入“歌手“、”专辑“、”作曲人“、”作词人“、”歌曲“页面。主界面提供网站概览，提供网站创作者相关资料等基本信息。

4. 歌手界面。展示热门歌手，点击热门歌手可显示歌手的资料，并提供搜索歌手信息的功能。

5. 专辑界面。展示热门专辑，点击热门专辑可显示专辑的资料，并提供搜索专辑信息的功能，并提供专辑的分类功能。

6. 作曲人界面。展示知名作曲人，点击知名作曲人可显示作曲人的资料，并提供搜索作曲人信息的功能。

7. 作词人界面。展示知名作词人，点击知名作词人可显示作词人的资料，并提供搜索作词人信息的功能。

8. 歌曲界面。展示热门歌曲，点击歌曲图片可现实歌曲的相关资料，并提供搜索歌曲的

功能，同时还实现了歌曲的分类功能。

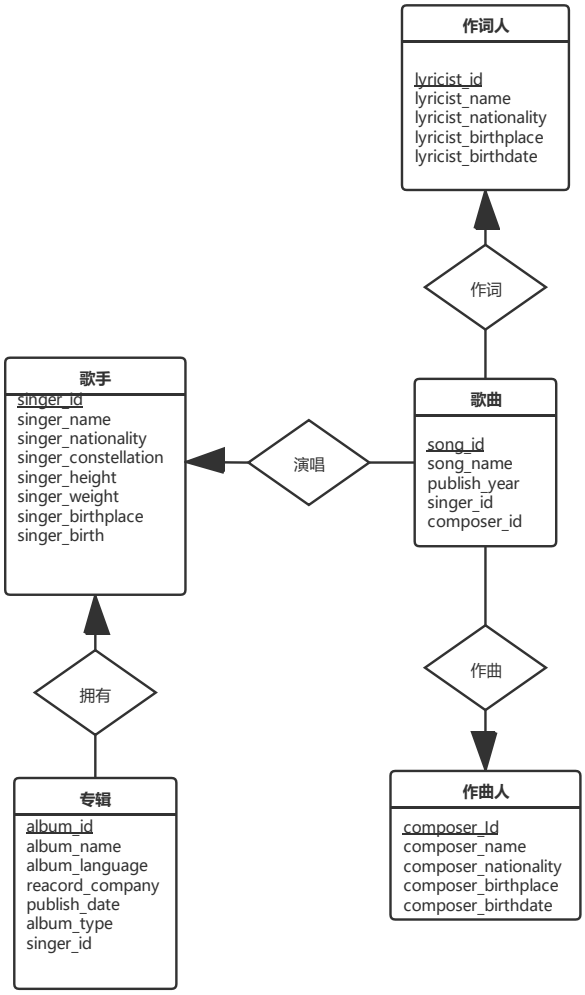
三、应用系统的设计（数据库设计、ER 图）：

1. 通过需求分析建立实体集以及联系集

我们建立了 5 个实体集，这 5 个实体集分别是“Singers”（歌手）、“Albums”（专辑）、“Composers”（作曲人）、“Lyricists”（作词人）、“Songs”（歌曲）。

各个实体间的联系是：专辑是歌手的作品，歌曲由歌手演唱，歌曲由作曲人作曲，歌曲由作词人作词，每首歌曲都属于一个专辑。

2. ER 图



其中加了下划线的部分是主键

3. 数据库关系表格

每个表格标明了主键(PK)和外键(FK)

singers（歌手）		
字段名	含义	类型
singer_id(PK)	歌手 id，用作主键	char(8)
singer_name	姓名	varchar(30)
singer_nationnality	国籍	varchar(20)
singer_constellation	星座	varchar(10)
singer_height	身高	decimal(4, 1)
singer_weight	体重	decimal(4, 1)
singer_birthplace	出生地	varchar(30)
singer_birth	生日	date

composers（作曲人）		
字段名	含义	类型
composer_id(PK)	作曲人 id，用作主键	char(8)
composer_name	姓名	varchar(30)
composer_nationality	国籍	varchar(20)
composer_birthplace	出生地	varchar(30)
composer_birth	生日	date

lyricists（作词人）		
字段名	含义	类型
lyricist_id(PK)	作词人 id，用作主键	char(8)
lyricist_name	姓名	varchar(30)
lyricist_nationality	国籍	varchar(20)

lyricist_birthplace	出生地	varchar(30)
lyricist_birth	生日	date

albums（专辑）		
字段名	含义	类型
album_id(PK)	专辑 id，用作主键	char(8)
album_name	专辑名	varchar(30)
album_language	专辑语言	varchar(20)
record_company	唱片公司	varchar(30)
publish_date	发行日期	date
album_type	专辑类型	varchar(20)
singer_id(FK-singers 表的 singer_id)	歌手 id	char(8)

songs（歌曲）		
字段名	含义	类型
song_id(PK)	歌曲 id，用作主键	char(8)
song_name	歌曲名	varchar(30)
publish_year	发行年份	char(4)
singer_id(FK-singers 表的 singer_id)	歌手 id	char(8)
composer_id(FK-composers 表的 composer_id)	作曲人 id	char(8)
lyricist_id(FK-lyricists 的 lyricist_id)	作词人 id	char(8)
album_id(FK-albums 的 album_id)	专辑 id	char(8)

四、系统的实现（包括系统主要运行界面）：

1. 使用工具

数据库：mySQL

后端：Django

前端：html+bootstrap

2. 数据库的实现

控制台连接 mysql 服务器，使用命令 create database music 创建数据库，use music 切换数据库后，执行创建五个表的语句（示例为创建 singers 的语句）：

```
create table singers(
singer_id char(8) primary key,
singer_name varchar(30) not null,
singer_nationality varchar(20),
singer_constellation varchar(10),
singer_height decimal(4, 1),
singer_weight decimal(4, 1),
singer_birthplace varchar(30),
singer_birth date
);
……
```

创建完表后，（后端实现部分）使用 django 的命令 python manage.py inspectdb > models.py 导出模型文件，复制到的 app 对应的目录下即可。

3. 后端实现

1. 使用 Django 创建项目，并且创立 APP

新建一个叫 music_management_system 的项目

django-admin startproject music_management_system

以下为生成的文件结构

```
music_management_system /
    manage.py（用来管理 Django 项目的命令行工具）
    music_management_system /（项目文件夹）
        __init__.py（空文件，告诉 Python 这个目录应该被认为是一个 python 包）
        settings.py（项目的配置文件，包含 app、数据库 等）
        urls.py （网站的目录，比如 http://127.0.0.1/XXXX/，这个文件就保存 XXX/）
        wsgi.py（运行在与 wsgi 兼容的 web 服务器的入口）
```

然后再新建一个叫 music 的 APP

```
py manage.py startapp music
```

2. 数据库模型的导入（models.py）

我们需要在 music 这个 APP 中设置模型，模型可以手动编写，或者由数据库建立的关系后使用命令导入。我们这里使用从数据库中导入模型的实现，敲击以下命令

```
python manage.py inspectdb > music/models.py
```

正如我们在应用系统设计中提到的那样，我们设计了 5 个实体，分别对应：

```
class Albums(models.Model):   # 专辑
class Composers(models.Model): # 作曲人
class Lyricists(models.Model): # 作词人
class Singers(models.Model): # 歌手
class Songs(models.Model): # 歌曲
```

下面我们仅展示其中专辑实体的代码实现，其他的实体相似，不作重复介绍。

每个关系都需要定义主键，如果有联系，则需要添加相应的外键。

```
class Albums(models.Model):
    album_id = models.CharField(primary_key=True, max_length=8)
```

```
    album_name = models.CharField(max_length=30)
    album_language = models.CharField(max_length=20, blank=True, null=True)
    record_company = models.CharField(max_length=30, blank=True, null=True)
    publish_date = models.DateField(blank=True, null=True)
    album_type = models.CharField(max_length=20, blank=True, null=True)
    singer = models.ForeignKey('Singers', models.DO_NOTHING, blank=True,
null=True)
    class Meta:
        managed = False
        db_table = 'albums'
    def __str__(self):
        return self.album_name
```

3. 网站链接的设置（urls.py）

根据我们的需求，我们需要实现“主页”、“歌手页”，“作曲人页”，“作词者页”，“专辑页”，“歌曲页”，因此我们需要为每一个页面添加链接，并且填写好对应的视图函数，并指定链接的名称，以方便引用，避免硬编码。

```
urlpatterns = [
    path('', views.index, name='index'),
    path('singer/',views.singer,name='singer'),
    path('composer/',views.composer,name='composer'),
    path('lyricist/',views.lyricist,name='lyricist'),
    path('album/',views.album,name='album'),
    path('song/',views.song,name='song')
]
```

4. 视图的设置

视图的用途是：接受 https 请求（request），对 https 请求处理后将处理完毕的 request 请求传递给 html 网页，并且渲染相应的网页。下面展示了“主页”（index）的视图函数设置。同时提供了异常处理机制，将代码封装到 Try-Except 当中，出错即向前端返回一个 404 页面。

```
def index(request):
    try:
        singers_list = models.Singers.objects.order_by('-singer_name')[:5]
        context={'singers_list':singers_list}
    except:
        raise Http404("Ooooooooooo!!!!!!!!!!! You have error access this
website, please contract 673624736@qq.com!")
    return render(request, 'music/index.html', context)
```

index 视图函数实现的功能有，模型中的 Singer 关系中获取歌手的名字，并且将其添加到环境的上下文中，以字典形式存储，然后将上下文和 http 请求传递给 html 模板 index.html

5. 搜索功能的实现

根据我们的需求，我们需要实现搜索功能。当用户输入关键字的时候，如歌手名字，可输出相关信息。

以实现搜索“歌手”信息为例。为了从数据库中查询歌手相关的信息，我们首先需要在歌手主页对应的视图函数中，从 request 对象获取用户输入的关键字，然后从模型中根据关键字筛选出符合条件的数据项，最后将搜索出来的数据项保存到上下文字典中，传回给 html 模板进行处理。

```
def singer(request):
    # 查看用户的输入是否为空
    if request.GET.get("search_text") == None or
request.GET.get("search_text") == '':
        search_result = []
    else:
        # 从 Singer 关系中过滤出符合条件的数据项
        search_result =
Singers.objects.filter(singer_name__contains=request.GET.get("search_text"))
        if not search_result:
            search_result = ['N']
    context={'search_result':search_result}
    # 传递上下文并且渲染 html
    return render(request, 'music/singer.html', context)
```

6. 管理员界面的实现

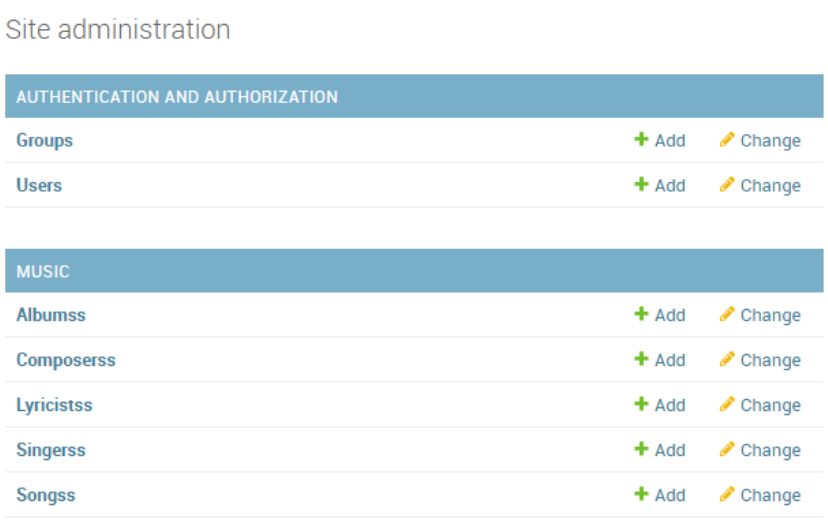
django 内置已实现了后台管理员用户界面。我们可以使用后台的管理员用户界面对数据库进行增、删、改、查等操作。

为了使用后台，首先需要创建后台管理者:

```
python manage.py createsuperuser
```

创建完毕后，我们在浏览器中输入 XX.XX.XX.XX/admin/ （XX.XX.XX.XX 指 IP）以进入管理员界面。

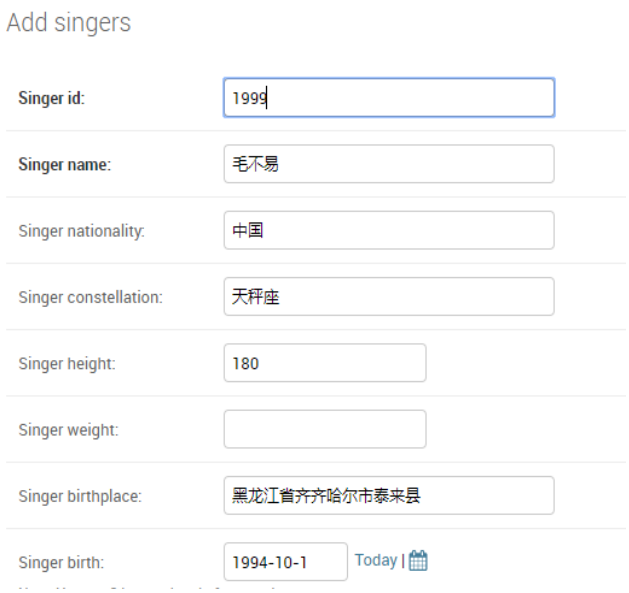
同时，为了显示的优雅，我们需要对 views.py 中的函数添加 def __str__(self)方法，使得实体的里的元素以名字字段的内容显示。



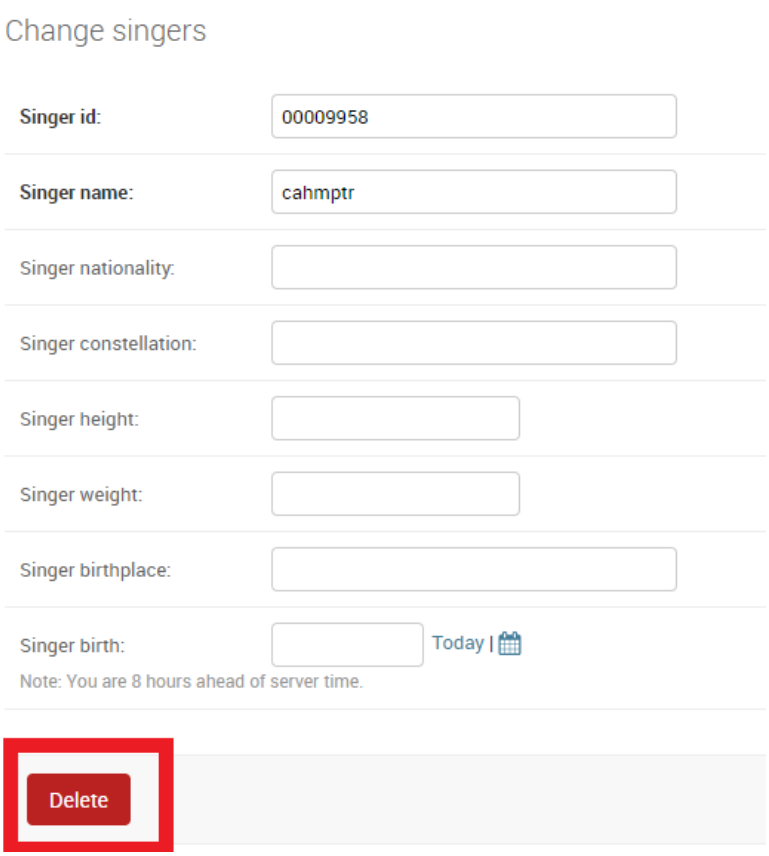
可以看到，此时后台界面中已经出现了我们添加的 5 个实体关系。

以 Singers 这个实体关系为例，演示数据库的增删改查。

① 增加数据项

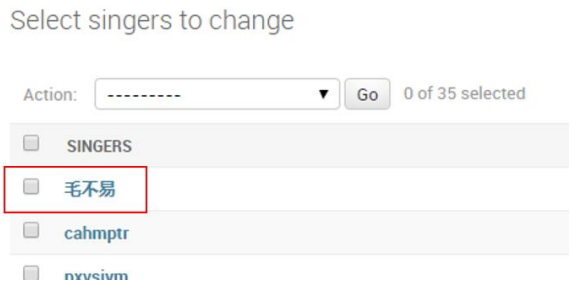


② 删除数据项

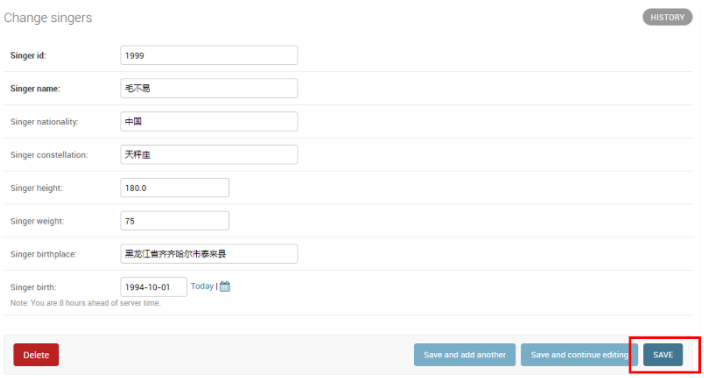


③ 修改数据项

点击添加的“毛不易”数据项



更改后点击 Save 按钮，即可完成更改



4. 前端实现

前端实现在 django 生成的目录下的 music/templates/music/文件夹下，

样式表在 django 生成的目录下的 music/static/music/文件夹下

由于网页可根据不同的功能分成不同的板块，因此我们可以使用 对不同的功能模块进行分块处理。

1. 使用工具说明

html 语言+javascript 语言与 bootstrap 包

2. 网页具体实现

① 静态样式表与 javascript 的载入

css 的载入：

```
<!--载入静态样式表-->
<link rel="stylesheet" href="{% static 'music/owl-
carousel/owl.carousel.css' %}">
<link rel="stylesheet" href="{% static 'music/owl-
carousel/owl.theme.css'%}"   >
<link rel="stylesheet" href="{% static 'music/owl-
carousel/owl.transitions.css' %}">
<link rel="stylesheet" href="{% static 'music/css/font-awesome.min.css' %}">
<link rel="stylesheet" href="{% static 'music/css/main.css' %}">
<link rel="stylesheet" href="{% static 'music/css/color1.css' %}">
<link rel="apple-touch-icon" href="apple-touch-icon.png">
<link rel="stylesheet" href="{% static 'music/css/bootstrap.min.css' %}">
```

javascript 的载入

```
<!--外部 javascript-->
<script src="{% static 'music/js/vendor/modernizr-2.8.3-respond-
1.4.2.min.js' %}"></script>
```

② 导航栏的实现

应用了 bootsrap 的“navbar”类，使用无序列表对链接进行组织。

通过导航栏，我们可以进入“主页”、“歌手”、“专辑“，”作词人“，”作曲人“，”歌曲“页面

```
<!-- Navigation 标签定义导航的链接 -->
<nav id="mainNav" class="navbar navbar-default navbar-custom navbar-fixed-
```

```
top">
    <div class="container">
        <!--导航栏-->
        <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-
1" >
            <ul class="nav navbar-nav text-center" >
                <li><a href="{% url 'index' %}">>主页</a></li>
                <li><a href="{% url 'singer' %}">歌手</a></li>
                <li><a href="{% url 'album' %}">专辑</a></li>
                <li><a href="{% url 'lyricist' %}">作词人</a></li>
                <li><a href="{% url 'composer' %}">作曲人</a></li>
                <li><a href="{% url 'song' %}">歌曲</a></li>
            </ul>
        </div>
    </div>
</nav>
```

③ 滚动展示图片类的编写

```
<div class="layer1">
    <div class="container">
        <div id="main-banner" class="main-banner">
            <div class="intro-text">
                <div class="intro-heading"><h1><span>欢迎来到</span>音乐管理系
统<span>created by 江金昱_赖少玄_黄镇_蓝靖瑜</span></h1></div>
            </div>

            <div class="intro-text">
                <div class="intro-heading"><h1><span>我们的</span>小组成员学号
<span>17341067_17341072_17341063_17341073</span></h1></div>
            </div>
        </div>
    </div>
    <div class="banner-layer layer2"></div>
    <div class="banner-layer layer3"></div>
</div>
```

③ 显示搜索结果-动态渲染机制

下面的代码首先判断从后端视图函数传来的 context 字典（上下文）中的字段“search_result”是否为空，如果非空则显示搜索结果，搜索结果采用遍历的方式输出，以列表的方式呈现。

```
{% if search_result %}
<center>
```

```
<section id="search-results">
    {% ifequal search_result|first "N" %}
    <p>No results</p>
    {% else %}
    <div class="table-responsive">
        <table class="table-fill">
            <tr>
                <th>姓名</th>
                <th>国籍</th>
                <th>星座</th>
                <th>身高</th>
                <th>体重</th>
                <th>出生地</th>
                <th>出生日期</th>
            </tr>
            {% for r in search_result %}
            <tr>
                <td>{{r.singer_name}}</td>
                <td>{{r.singer_nationality}}</td>
                <td>{{r.singer_constellation}}</td>
                <td>{{r.singer_height}}</td>
                <td>{{r.singer_weight}}</td>
                <td>{{r.singer_birthplace}}</td>
                <td>{{r.singer_birth}}</td>
            </tr>
            {% endfor %}
        </table>
        <br/>
    </div>
    {% endifequal %}
</section>
</center>
{% endif %}
```

④ 分类功能的实现（以专辑为例）

思想方法是，为每一种类别的专辑设定一个属性（如 `data-filter=".category-1"`）当添加专辑的时候，就将对应专辑的 class 属性添加上对应类别的类别即可（如 `class="col-md-3 col-sm-6 portfolio-item mix category-1"`），如下代码所示

```
<div class="col-md-12 text-center">
    <ul class="cat-list">
        <li><button class="filter" data-filter="all">所有专辑</button></li>
```

```
        <li><button class="filter" data-filter=".category-1">流行乐
</button></li>
        <li> <button class="filter" data-filter=".category-2">乡村乐
</button></li>
        <li> <button class="filter" data-filter=".category-3">摇滚乐
</button></li>
        <li> <button class="filter" data-filter=".category-4">原声带
</button></li>
    </ul>
</div>
```

添加专辑进对应的类别：

```
<div class="col-md-3 col-sm-6 portfolio-item mix category-2" data-value="4">
    <a href="album/?search_text=red" class="portfolio-link">
        <div class="portfolio-hover">
            <div class="portfolio-hover-content">
                <p>Red</p>
                <h4>泰勒斯威夫特</h4>
            </div>
        </div>
        <img src="{% static 'music/images/album/red.jpg' %}" class="img-
responsive" alt="">
    </a>
</div>
```

五、将网站部署到阿里云轻量应用服务器上

我们购买了阿里云学生机的轻量应用服务器，操作系统选的是 centos7.3。我们将网站部署到服务器上的过程如下：

1. 更新软件

```
yum install update
```

2. 安装 python3

```
yum install python3
```

3. 安装 django

```
pip3 install django
```

4. 安装 pymysql

```
pip3 install pymysql
```

5. 安装 mysql

```
wget http://dev.mysql.com/get/mysql57-community-release-el7-8.noarch.rpm
yum localinstall mysql57-community-release-el7-8.noarch.rpm
yum install mysql-devel
yum install mysql-community-server
```

6. 启动 mysql

```
systemctl start mysqld
```

7. 修改 mysql 密码

先用 grep 'temporary password' /var/log/mysqld.log 得到临时密码

然后执行 mysql -u root -p 后输入临时密码登录

然后用 set password for 'root'@'localhost'=password('xxxxxxxx'); 修改密码

8. 用 scp 将本地的文件夹传到服务器上

scp -r music_management_system root@服务器地址:/root

注意要修改 settings.py 里的密码，并且要把端口那行注释掉

9. 导入数据库

先执行 mysql -uroot -p 后输入密码

```
create database music;
use music;
source /root/music_management_system/music.sql
```

我们第一次导入的时候导入失败了，上网搜索后发现是因为版本不兼容，我们在本地将 sql 文件里的 utf8mb4_0900_ai_ci 全部替换为 utf8_general_ci，然后将 utf8mb4 替换为 utf8，然后重新上传后成功导入数据库

10. 运行网站

我们首先尝试执行 python3 manage.py runserver 来启动服务器，结果发现报错，报错内容为：django.core.exceptions.ImproperlyConfigured: mysqlclient 1.x.xx or newer is required; you have 0.9.3. 我们将报错信息作为关键字在网上搜索，得到了解决方法：根据错误提示，

找到 base.py 的文件路径，将里面判断版本号的两行代码注释掉即可。

这时我们继续尝试启动服务器，结果还是报错，根据之前的经验，我们先后执行了 python3 manage.py makemigrations 和 python manage.py migrate 两条命令之后，再运行就不会报错了。但是这时用我们自己的电脑还是打不开我们的阿里云服务器上的网站，根据网上的资料，我们先退出之前的进程，然后执行 python3 manage.py runserver 0.0.0.0:80，这时我们就可以成功用自己的电脑打开自己的网站了。

然而这时还有最后一个问题：如果我把 ssh 窗口关掉了，那网站也上不去了。根据经验，我尝试在命令后面加上&之后在执行，发现还是不行。然后我查阅网上的资料，我尝试执行 (python3 manage.py runserver 0.0.0.0:80 &)，就是不仅加&，还加括号。我关掉 ssh 窗口，发现能上，终于成功了。

11. 文件名后缀名大小写的问题

最后再说一个部署网站时遇到的问题，网站部署好之后，虽然能上了，并且我自己看起来很正常，但是细心的舍友发现，在歌手那一页，周杰伦和花泽香菜的照片没有显示出来。我用浏览器右键查看网页源代码，找到原本应该显示的图片的文件地址，然后我再去文件夹里面找这个文件，发现明明是有照片的啊，怎么又会没有呢？这时我突然注意到：文件夹里的文件的后缀名.JPG 是大写的，而网页源代码的文件的后缀名.jpg 是小写的。如果在 windows 环境下运行就没问题，但是 Linux 环境下是分大小写的。然后我用 ssh 把服务器上的这两张照片的后缀名改成小写，就可以了。

六、可运行的程序网址与网站功能截图

1. 可直接访问的运行网址（可访问时间 2019 年 12 月 20 日至 2020 年 1 月 19 日）

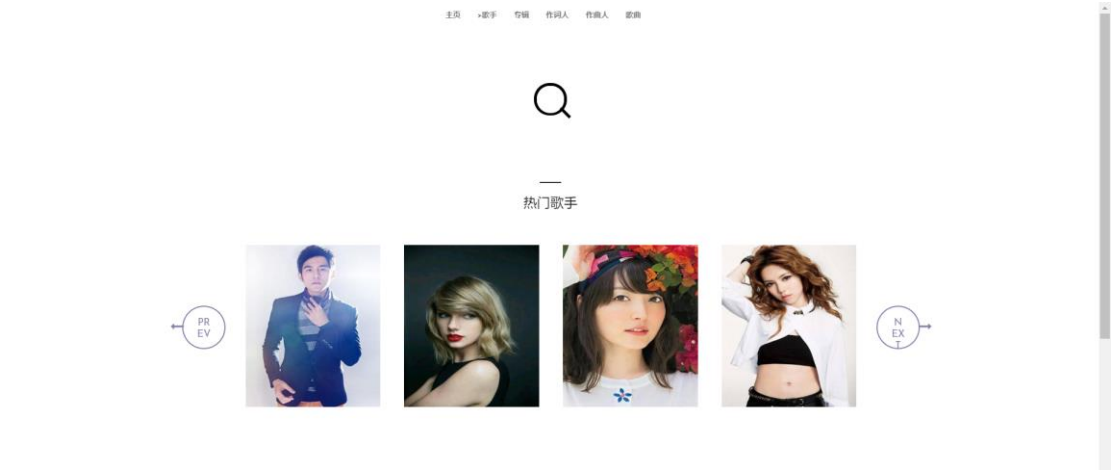
http://139.196.15.209/ （打不开请联系 673624736@qq.com 或者群里私戳江金昱）

2. 网站主页以及各页面展示

在浏览器中，输入网站地址，进入网站主页，如下图。

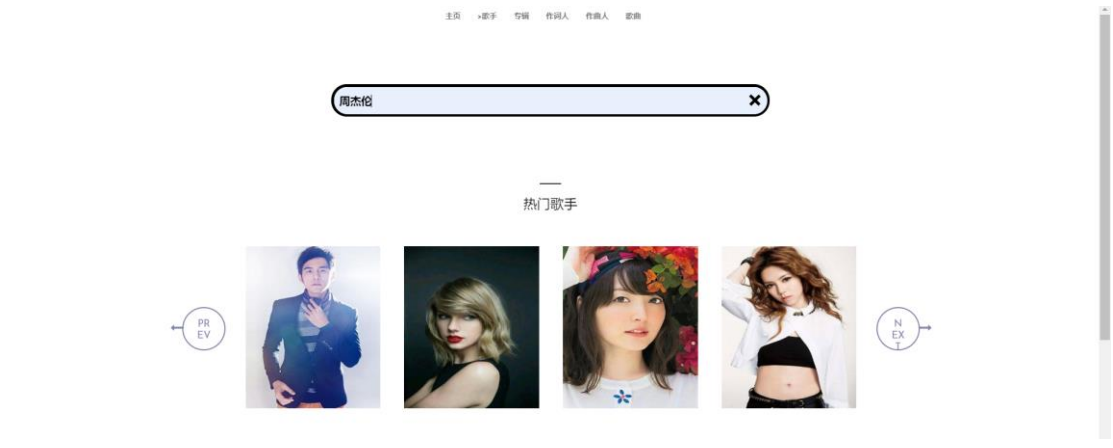
点击网站上方的导航栏可以进入“主页”、“歌手”、“专辑”、“作词人”、“作曲人”、“歌曲”界面。

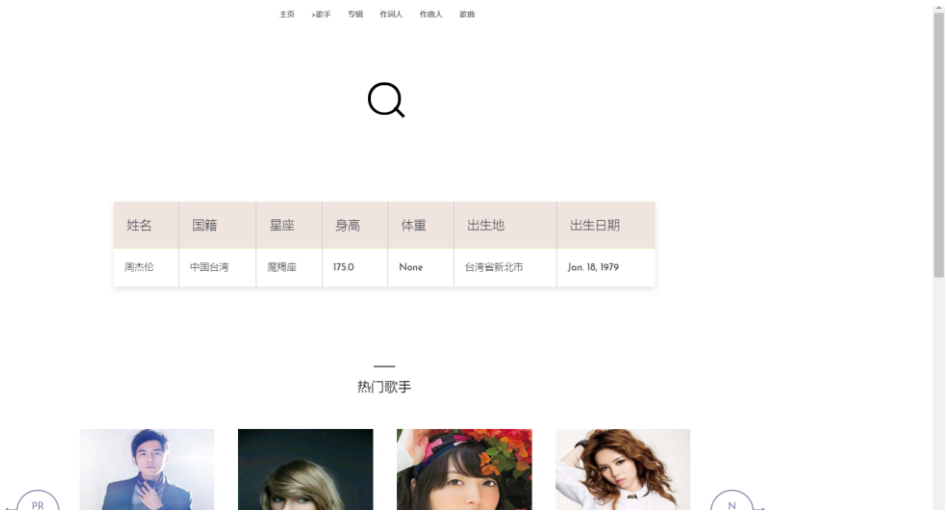
如，我们点击“歌手“，进入歌手界面，如下图：



3. 搜索功能展示

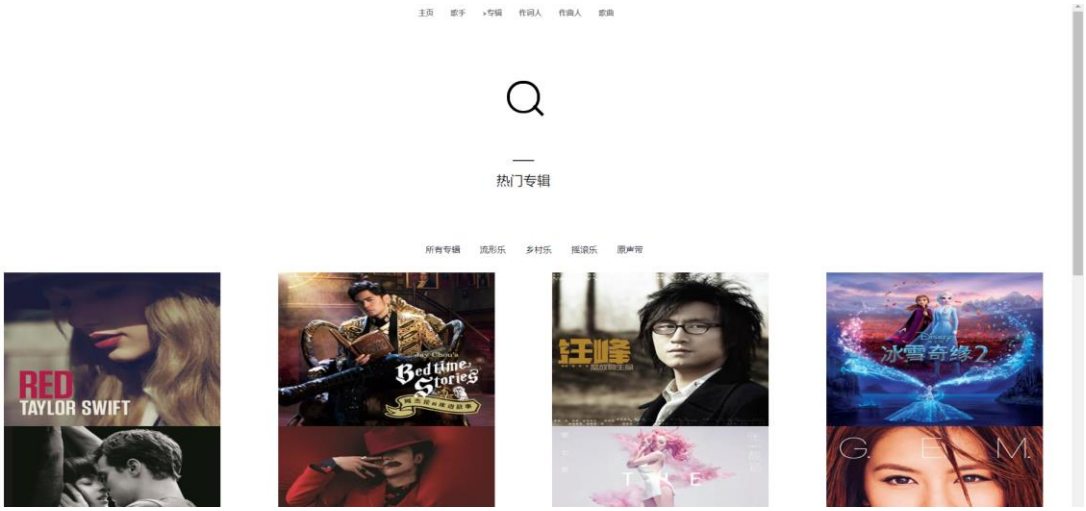
点击搜索框，搜索框会展开，输入歌手的名字可显示歌手的信息，其他页面同理，如下图。



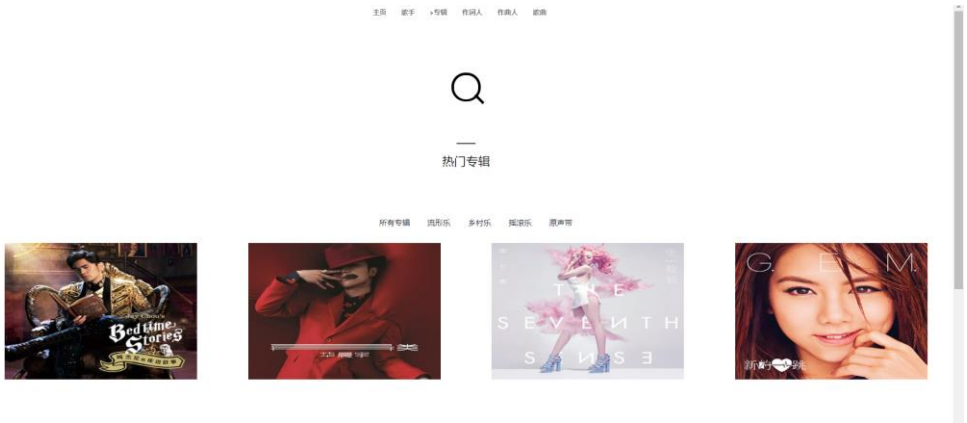


4. 热门专辑分类功能展示

进入专辑页面，可看到，专辑的分类有“所有专辑”、“流行乐”、“乡村乐”、“摇滚乐”、“原声带“。

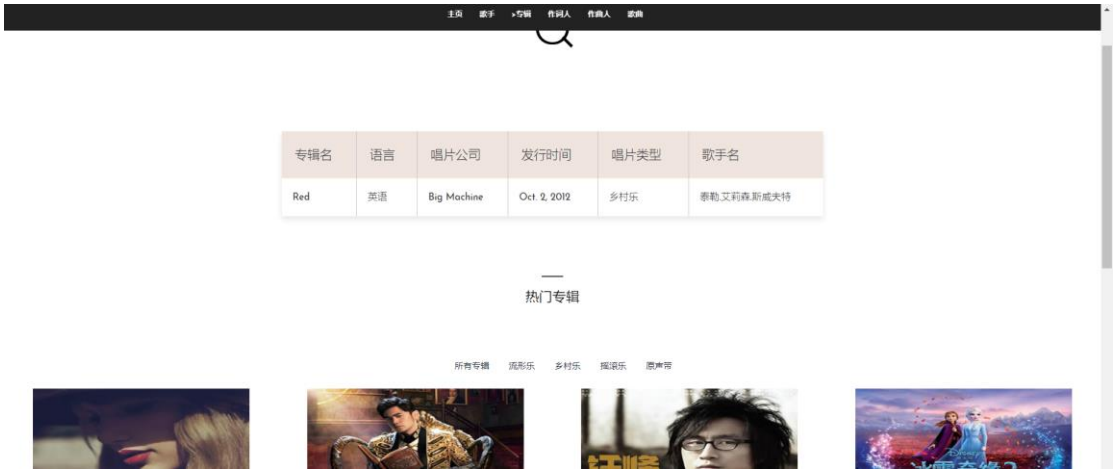


点击专辑类别，可显示相应专辑类别的专辑，如下图所示：



5. 点击图片显示专辑/歌手/作曲人/作词人 信息展示

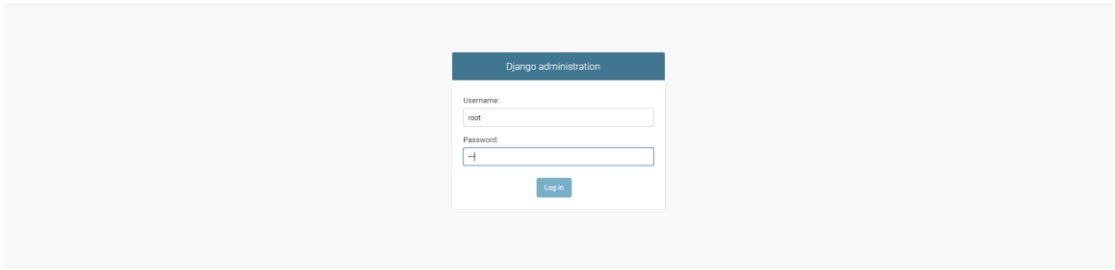
以专辑为例，我们点击 TaylorSwift 的专辑“Red“（第一张），可直接显示专辑信息



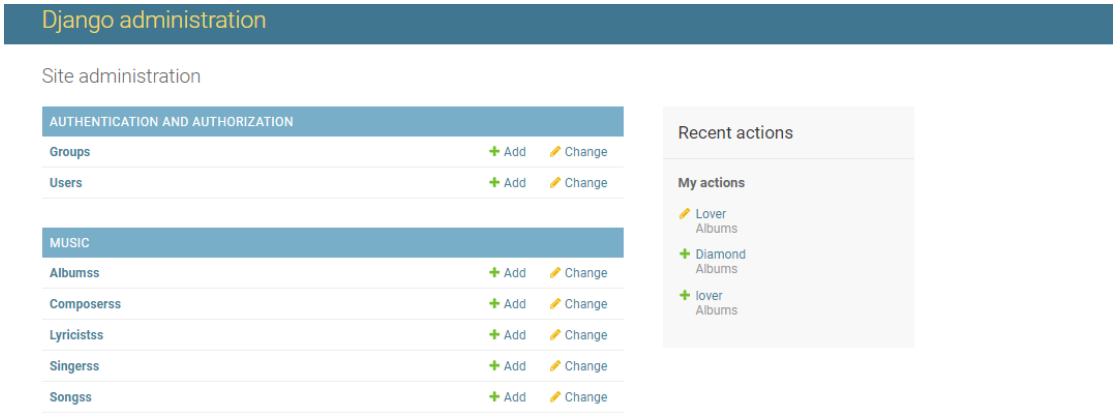
6. 管理员界面展示（网站数据的增删改）

登陆 http://139.196.15.209/admin/

输入用户名：root　密码：123



可以看到我们添加的 5 个实体



点击 Albums，可以看到我们添加的专辑名字（一些是随机生成的，下拉到最后可以看到真

实的数据），点击右上方可以添加专辑。

点击专辑名字，可以对专辑进行删除（左下方）和更改（右下方）