# 微信小程序设计文档

题 目： 音乐播放器

学 号： 2008030455

姓 名： 庄凯林

指导教师： 高建文

# 一、配置环境

## 1.1项目开发环境

本项目开发使用环境配置如下：

运行环境：Windows10

开发工具：python（3.7）、mysql（5.9）、微信开发者工具、PyCharm（python开发工具）、SQLyog

数据库：mysql

服务器后端：FastAPI

## 1.2项目介绍

### 1.2.1项目基本信息介绍

微信小程序名称：音乐播放器

功能：用户的登录\注册、音乐的搜索、播放音乐、获取音乐基本介绍、下载音乐

模块划分：用户模块、音乐模块

### 1.2.2模块介绍

#### 1 用户模块

用户模块包含了用户的登录、注册，前端显示为小程序页面，在页面输入信息后提交到后端服务器进行登录。后端收到信息后端进行以下判断：

**登录请求：**判断验证码是否正确，错误时返回错误信息，正确时在mysql数据库查找用户名/电话号码—密码 相等的数据，查找成功时返回token，否则，返回提示信息。

**注册请求：**判断验证码是否正确，错误时返回错误信息，正确时在mysql数据库查找用户名相等的数据，如果查找成功，返回错误信息，否则进行用户注册。

#### 2 音乐模块

音乐模块包含了音乐的搜索、播放、下载。

搜索：在用户未登录情况下，用户拥有搜索的权限，在【分类】页面，输入搜索信息返回搜索数据。

播放：未登录情况下可以进行音乐播放。

下载：下载音乐需要用户在已登录状态，否则将不执行下载。

## 三、项目详细介绍

### 3.1 后端介绍

#### 3.1.1 项目功能层次划分

小程序后端服务器功能划分如下图所示：

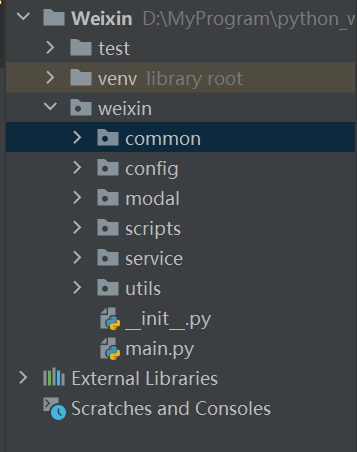


图 3.1.1-1 功能划分结

以下为项目相关介绍：

* Wexin：项目根目录
* test：模块测试目录
* venv：项目配置库
* weixin：主要业务模块目录
  + common：返回给前端数据的公共目录，里面设置统一的响应数据
  + config：配置目录（主要用户配置mysql数据库连接信息）
  + modal：实体类（mysql数据表转换的python类）
  + scripts：python爬虫目录（获取网络数据到mysql数据表中）
  + service：服务类目录（对外提供服务的目录，如用户登录、注册）
  + utils：工具类目录（独立无关的功能类）

#### 3.1.2 项目功能代码介绍

**1. 公共响应类**

from pydantic import BaseModel  
from typing import Any  
from enum import IntEnum  
  
  
class CodeStatus(IntEnum):  
 SUCCESS = 1  
 FAIL = 0  
  
  
class CommonResult(BaseModel):  
 code: CodeStatus  
 info: str  
 data: Any

code：响应状态（0-失败，1-成功）

info：响应信息

data：响应数据

**2.数据库配置**

# MySQL数据库配置  
mysql\_config = {  
 "host": "localhost",  
 "port": 3306,  
 "user": "root",  
 "password": "密码",  
 "database": "db\_weixin"  
}

数据库配置用户连接数据库时的身份校验

**3.实体类**

音乐信息类

from pydantic import BaseModel, AnyUrl  
from typing import Union  
  
  
class MusicInfo(BaseModel):  
 artist: str  
 pic: Union[AnyUrl, None] = None  
 url: AnyUrl  
 duration: str  
 release\_date: Union[str, None] = None  
 album: Union[str, None] = None  
 song\_time\_minutes: str  
 sname: str

相关属性介绍：

* artist：歌手
* pic：歌曲图片链接
* url：音源链接
* duration：总时长
* release\_time：发行时间
* album：所属专辑
* song\_time\_minutes：【分钟：秒】形式的总时长
* sname：歌名

用户信息类：

from typing import Union  
  
from pydantic import BaseModel, EmailStr  
  
  
class UserBase(BaseModel):  
 nickname: str  
 phone: Union[str, None] = None  
 passwd: str  
  
  
class LoginUser(UserBase):  
 pass  
  
  
class RegisterUser(UserBase):  
 pass  
  
  
class Profile(BaseModel):  
 uid: int  
 email: Union[EmailStr, None] = ""  
 gender: int = 0  
 city: Union[str, None] = ""  
 province: Union[str, None] = ""  
 lang: Union[str, None] = ""  
 commentary: Union[str, None] = ""

相关信息介绍如下：

UserBase类：

* nickname：用户昵称
* phone：联系电话
* passwd：密码

Profile类：

* uid：用户id
* email：电子邮箱
* gender：性别（0-男，1-女）
* city：城市
* province：省份
* lang：语言
* commentary：个人简介

登录业务：

@app.post("/login", response\_model=CommonResult)  
async def login(co: str = Body(embed=True), user: LoginUser = Body(embed=True)) -> Any:  
 print(f"用户登录：{user}")  
 global code, g\_token  
  
 if co != code:  
 resp = CommonResult(code=CodeStatus.FAIL, info="验证码错误", data=None)  
 else:  
 conn, cursor = mysqlTemplate.get\_conn()  
 sql = "select id from t\_user where nickname=%s and passwd=%s"  
 rows = mysqlTemplate.query(sql, cursor, (user.nickname, user.passwd))  
 resp = CommonResult(code=CodeStatus.FAIL, info="用户不存在", data=None)  
  
 g\_token = get\_token()  
  
 if len(rows) > 0:  
 resp = CommonResult(code=CodeStatus.SUCCESS, info="登录成功",  
 data={"token": g\_token, "id": rows[0]["id"]})  
 else:  
 sql = "select id from t\_user where phone=%s and passwd=%s"  
 rows = mysqlTemplate.query(sql, cursor, (user.nickname, user.passwd))  
 if len(rows) > 0:  
 resp = CommonResult(code=CodeStatus.SUCCESS, info="登录成功",  
 data={"token": g\_token, "id": rows[0]["id"]})  
 mysqlTemplate.close(conn, cursor)  
 return resp

参数：

* co：验证码
* user：用户信息

登录业务流程：

1. 判断验证码
   * + 1. 验证码错误，返回错误信息
       2. 校验成功，连接数据库，查询是否存在用户
          1. 存在，返回成功信息、token和uid
          2. 不存在，返回错误信息

注册业务：

@app.post("/register", response\_model=CommonResult)  
async def register(co: str = Body(embed=True), user: RegisterUser = Body(embed=True)) -> Any:  
 print(f"用户注册：{user}")  
 global code  
  
 if co != code:  
 resp = CommonResult(code=CodeStatus.FAIL, info="验证码错误", data=False)  
 else:  
 conn, cursor = mysqlTemplate.get\_conn()  
 sql = "select count(\*) from t\_user where nickname=%s"  
 rows = mysqlTemplate.query(sql, cursor, (user.nickname,))[0]["count(\*)"]  
 if rows > 0:  
 return CommonResult(code=CodeStatus.FAIL, info="用户已存在", data=None)  
  
 sql = "insert into t\_user(nickname, phone, passwd) values (%s, %s, %s)"  
 rows = mysqlTemplate.execute(sql, cursor, (user.nickname, user.phone, user.passwd))  
 mysqlTemplate.commit(conn)  
  
 if rows > 0:  
 resp = CommonResult(code=CodeStatus.SUCCESS, info="注册成功", data=True)  
 sql = "select id from t\_user where nickname=%s and passwd=%s"  
 rows = mysqlTemplate.query(sql, cursor, (user.nickname, user.passwd))[0]["id"]  
 add\_profile(uid=rows)  
 else:  
 resp = CommonResult(code=CodeStatus.FAIL, info="注册失败", data=False)  
 mysqlTemplate.close(conn, cursor)  
 return resp

参数介绍：

* co：验证码
* user：用户信息

注册业务流程：

1. 判断验证码
   * + 1. 验证码错误，返回错误信息
       2. 校验成功，连接数据库，查询是否存在用户
          1. 存在，返回错误信息
          2. 不存在，注册用户

获取验证码：

@app.get("/code")  
async def get\_code() -> str:  
 global code  
 code = "".join(random.choice(string.digits) for \_ in range(6))  
 return code

获取验证码函数返回六位数字组成的字符串。

获取用户简介业务：

@app.post("/profile", response\_model=CommonResult)  
async def get\_profile(uid: int = Body(embed=True), token: str = Body(embed=True)) -> Any:  
 global g\_token  
 if token != g\_token:  
 return CommonResult(code=CodeStatus.FAIL, info="非法访问", data=None)  
  
 conn, cursor = mysqlTemplate.get\_conn()  
 sql = "select \* from t\_user\_detail where uid=%s"  
 result = mysqlTemplate.query(sql, cursor, (uid,))  
 return CommonResult(code=CodeStatus.SUCCESS, info="查询成功", data=result)

参数介绍：

* uid：用户id
* token：登录成功时返回的token

获取用户简介流程：

1. 判断token
   * + 1. token错误，返回错误信息
       2. 成功，从数据库获取数据并返回

更新个人简介业务：

@app.post("/profile/update", response\_model=CommonResult)  
async def update\_profile(profile: Profile = Body(embed=True), token: str = Body(embed=True)) -> Any:  
 global g\_token  
 if token != g\_token:  
 return CommonResult(code=CodeStatus.FAIL, info="非法访问", data=None)  
  
 print(profile)  
 conn, cursor = mysqlTemplate.get\_conn()  
 sql = f"update t\_user\_detail set email=%s, gender=%s, city=%s, province=%s, commentary=%s where uid=%s"  
 data = (profile.email, profile.gender, profile.city, profile.province, profile.commentary, profile.uid)  
 count = mysqlTemplate.execute(sql, cursor, data)  
 mysqlTemplate.commit(conn)  
 mysqlTemplate.close(conn, cursor)  
  
 if count >= 0:  
 resp = CommonResult(code=CodeStatus.SUCCESS, info="保存成功", data=None)  
 else:  
 resp = CommonResult(code=CodeStatus.FAIL, info="保存失败", data=None)  
 return resp

参数介绍：

* profile：个人简介
* token：token

更新个人简介流程：

1. 判断token
   * + 1. token错误，返回错误信息
       2. 正确，连接数据库，更新个人简介数据

添加个人简介：

def add\_profile(uid: int) -> None:  
 conn, cursor = mysqlTemplate.get\_conn()  
 sql = f"insert into t\_user\_detail(uid) values (%s)"  
 mysqlTemplate.execute(sql, cursor, (uid,))  
 mysqlTemplate.commit(conn)  
 mysqlTemplate.close(conn, cursor)

该函数为指定uid的用户创建个人简介初始数据表

token获取：

def get\_token(count=32) -> str:  
 token = string.ascii\_letters + string.digits  
 return "".join(random.choice(token) for \_ in range(count))

返回由大小写字母和数字组成的字符序列，默认长度为32

获取音乐业务：

@app.get("/query", response\_model=CommonResult)  
async def query\_music(key: Union[str, None] = None, limit\_start: int = 0, limit\_len: int = 30) -> Any:  
 conn, cursor = mysqlTemplate.get\_conn()  
  
 if key is None:  
 sql = f"select \* from t\_music limit %s,%s"  
 result = mysqlTemplate.query(sql, cursor, (limit\_start, limit\_len))  
 else:  
 sql = """  
 select \* from t\_music   
 where sname like %s or artist like %s   
 limit %s,%s  
 """  
 result = mysqlTemplate.query(sql, cursor, (f"%{key}%", f"%{key}%", limit\_start, limit\_len))  
  
 resp = CommonResult(code=CodeStatus.SUCCESS, info="查询成功", data=result)  
 mysqlTemplate.close(conn, cursor)  
 return resp

参数介绍：

* key：查询字符串
* limit\_start：数据库指定开始位置
* limit\_len：获取数据长度

获取音乐流程：

1. 判断key是否为空
   * + 1. key为空，返回未过滤数据
       2. key不为空，返回过滤数据

main.py：项目主运行程序

### 3.2 前端介绍

index.wxml



type.wxml:





Mine.wxml:

