|  |  |  |
| --- | --- | --- |
| **文件状态**  **[ ]草稿**  **[ ]讨论稿**  **[X]正式发布** | **文档类型** | **详细设计说明书** |
| **文件标识** | **004-XS10** |
| **版 本** | **1.0** |
| **作 者** | **马鑫宇 等** |
| **完成日期** | **2016/4/14** |

项目名称：娱音绕梁

文档名称：软件详细设计说明书

文档修订

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **更改人** | **描述** |
| 0.5 | 2016.04.14 | 张伟 | 拟定草稿 |
| 1.0 | 2016.04.14 | 马鑫宇 | 正式定稿 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

批准人签字

|  |  |  |
| --- | --- | --- |
| **职务** | **姓名** | **日期** |
| 项目经理 | 张伟 | 2014/04/14 |
| 客户代表 | 张谨 | 2014/04/14 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# 目录

**[一、 概述 1](#_Toc12330)**

[1.1 目的 1](#_Toc23249)

[1.2范围 1](#_Toc32550)

[1.3术语定义 1](#_Toc17784)

**[二、 总体架构设计 3](#_Toc12709)**

[2.1系统内外部关系 3](#_Toc11253)

[2.2功能模块说明 3](#_Toc13934)

[2.3系统流程 5](#_Toc23861)

[2.4运行环境和第三方组件依赖 5](#_Toc16121)

[2.5依赖的外部接口 5](#_Toc23890)

[2.6对外提供的接口 5](#_Toc9606)

**[三、 前台代码设计 6](#_Toc22688)**

[3.1 Activity包 6](#_Toc31716)

[3.2 Adapter包 8](#_Toc32079)

[3.3 Constants包 9](#_Toc5276)

[3.4 File包 10](#_Toc9335)

[3.5 Helper包 12](#_Toc14927)

[3.6 Util包 13](#_Toc5241)

**[四、后台代码设计 15](#_Toc12055)**

[4.1 后台接口代码 15](#_Toc12121)

[4.2 后台管理页面代码 16](#_Toc7154)

**[五、数据存储设计 17](#_Toc13568)**

[5.1数据库设计 17](#_Toc6266)

[5.2视图的设计 17](#_Toc16525)

[5.3数据的物理存储 19](#_Toc30079)

**[六、用户界面设计 20](#_Toc8472)**

**[七、参考资料 20](#_Toc2358)**

# 

# 概述

## 1.1 目的

本文档的目的旨在推动软件工程的规范化，使设计人员遵循统一的详细设计书写规范，节省制作文档的时间，降低系统实现的风险，做到系统设计资料的规范性与全面性，以利于系统的实现、测试、维护、版本升级等。

详细设计的详细程度，应达到可以编写程序的水平。

## 1.2范围

本文档用于软件设计阶段的详细设计，它的上游（依据的基线）是概要设计说明书，它的下游是源程序清单及单元测试计划，并为单元测试报告提供测试依据。

软件详细设计的范围是：各子系统的公用模块实现设计、专用模块实现设计、存储过程实现设计、触发器实现设计、外部接口实现设计、角色授权设计、其它详细设计等。

按照二层结构（C/S）的布局，详细设计应从下面两个方面进行：

数据库服务器(S)上的面向数据的设计：数据字典物理设计、基本表物理设计、中间表物理设计（报表设计）、临时表物理设计、视图物理设计；

客户端(C)的面向对象的设计：音乐列表界面设计、音乐详情页面设计、挑战列表设计、挑战详情页面设计、社区设计、登录注册界面设计。

## 1.3术语定义

需求文档中所涉及术语此处不再赘述，如有冲突以需求文档为准。

|  |  |
| --- | --- |
| 术语/定义 | 说明 |
| 默认成功响应 | 客户端的所有请求，在服务器端成功时所给出的统一响应。 |
| 雾霾图 | 用户接受挑战前提供给用户的，模糊后的图像。 |
| 视图 | 计算机数据库中的视图是一个虚拟表，其内容由查询定义。同真实的表一样，视图包含一系列带有名称的列和行数据。但是，视图并不在数据库中以存储的数据值集形式存在。行和列数据来自由定义视图的查询所引用的表，并且在引用视图时动态生成。 |
| 实例 | Android中，一个Activity所依附于的Application(应用)，这个应用是个单例模式，可以被其他Activity随时调用。 |
| 上下文 | 在Android中，很多操作都需要传入实例。在非Activity类中不能够直接通过Activity获取Application，所以需要传入该实例，也称作“上下文”。 |

# 总体架构设计

## 2.1系统内外部关系

本系统分为前台安卓端和后台服务器两个部分。安卓端界面组件提供界面的展示，用户通过在界面的操作出发调度程序组件，调度程序组件向后台服务器发出请求，用户相关操作接口接受请求，将具体业务逻辑交给数据库操作模块组件处理，最后将处理结果发送给调度程序，调度程序再转交给安卓端界面，界面做出相应的显示，实现用户的交互。

前台构件主要负责完成捕捉用户操作的方式，从而触发不同的事件。这些事件有可能会造成后台Service的启动，有可能造成Activity页面切换，也有可能造成页面向服务器发送请求。

其中，Android内部使用SQLLite数据库进行客户端数据持久化，使用Broadcast Reciver进行Service与Activity之间的通信，使用Android Support V7兼容包使低版本Android可以使用Android 5.0新控件。

数据库端使用PHP+MySQL进行服务端数据持久化，使用PHP编写与客户端交互的接口，接受前端发送字符串类型和其他多媒体类型的文件。

## 2.2功能模块说明

**(1) 用户模块**

* + 用户身份验证

在使用APP的过程中，用户身份验证是最重要的部分之一，这关系到用户是否合法地使用APP，同时还有助于防止恶意用户的行为，所以用户身份验证是必不可少的部分。

本系统使用Token (令牌) 来校验用户的身份有效性。用户每一次登录会在服务器端生成一个用户身份令牌，并将其返回给用户。服务器端的Token有一定期限的有效时间。用户每提交一次数据都会发送一个客户端的Token，服务器在进行操作之前会对二者进行比较，如果有效才会返回值。如果用户登录状态已经失效，则要求用户重新登录。

* + 用户登录注册

用户通过填写登录名和登录密码完成登录，登录完成之后，用户将获得一个Token，相当于整个APP的通行证。以后的数据请求都需要该Token。

* + 用户个人信息模块

用户可以在APP的用户信息管理模块看到自己的信息，同时可以对自己的信息进行修改。由于本系统引入等级机制，故会有一些项目是和用户当前经验值相关且不可修改。

**(2) 音乐模块**

* + 音乐播放

用户可以在线播放音乐，同时可以在播放的时候跳转到上一首或者下一首。在播放音乐的时候可以看到音乐对应的详细信息。

* + 根据音乐发起挑战

用户在音乐播放详情页面，可以看到当前正在播放的音乐对应的几个标签。通过点击这些标签可以进入基于这些标签的挑战发起页面。由于系统设定每一个音乐对应一定数量的标签，所以每个音乐所对应的标签会各不相同。

**(3) 挑战模块**

* + 查看挑战列表

用户可以查看当前已经存在的挑战，列表里面每一个挑战都可以看到发起挑战的人的账户、发起时间、以及挑战的简介。

* + 挑战详情和接受挑战

通过点击挑战列表里面不同的列表项，会进入不同挑战详情页面。用户会首先看到一个被扭曲的图像。如果确认接收挑战就会看到对应的清晰图片。用户被要求在一定时间限制内猜出该图片对应的标签名称。如果猜测正确将有一定的经验奖励。

**(4) 社区模块**

* + 查看版块和帖子

用户可以查看不同版块下的不同的帖子，每一个帖子对应一种主题，如求助、灌水等等。用户可以在不同的版块下浏览不同的帖子。

* + 发布帖子

用户登录之后可以发布帖子，帖子要求发布到特定版块之下。用户发布帖子将有一定的格式要求，用户成功发布帖子之后可以在相应版块下看到自己发布的帖子。发布帖子将有一定的经验奖励。

**(5) 经验模块**

用户的某些动作会造成经验值的增加，增加经验值之后可以有不同的头像和不同的头衔。这些头像和头衔将会被现实在某些交互的地方（如用户发布了评论，则别人可以看到该用户的头像和头衔）。

本系统具备后台网页管理系统，如果后台管理员发现用户发送的帖子不符合标准，会根据情况作出提示或者进行删帖处理，酌情减少经验值。

## 2.3系统流程

以《需求说明书》为准。

## 2.4运行环境和第三方组件依赖

以《需求说明书》为准。

## 2.5依赖的外部接口

以《概要设计文档》 为准。

## 2.6对外提供的接口

以《概要设计文档》 为准。

# 前台代码设计

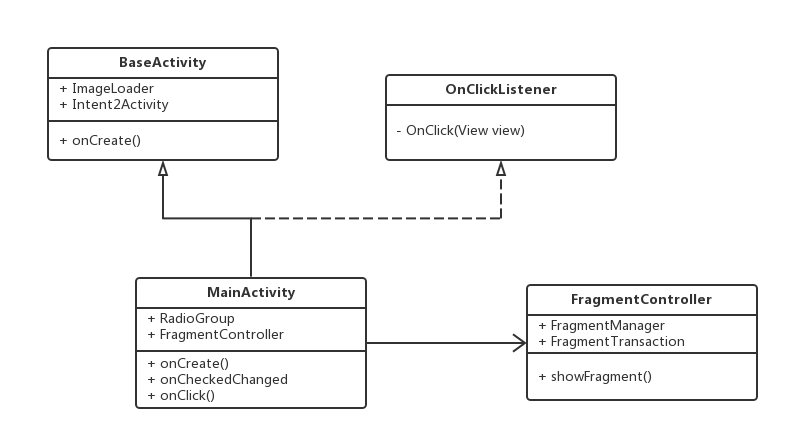
为了方便查看代码编写的要求，下面将以包名(Package)为分类，对前台每一个Package里面的主要代码文件的功能和实现进行阐述。

# 3.1 Activity包

1. **MainActivity.java**

功能：完成主界面Fragment的切换，为各个Fragment提供基础服务。

依赖关系：



变量、函数说明：

RadioGroup : 用于完成Fragment状态切换

FragmentController: Fragment切换助手类，用于完成详细的Fragment切换

OnCreate: 用于初始化界面

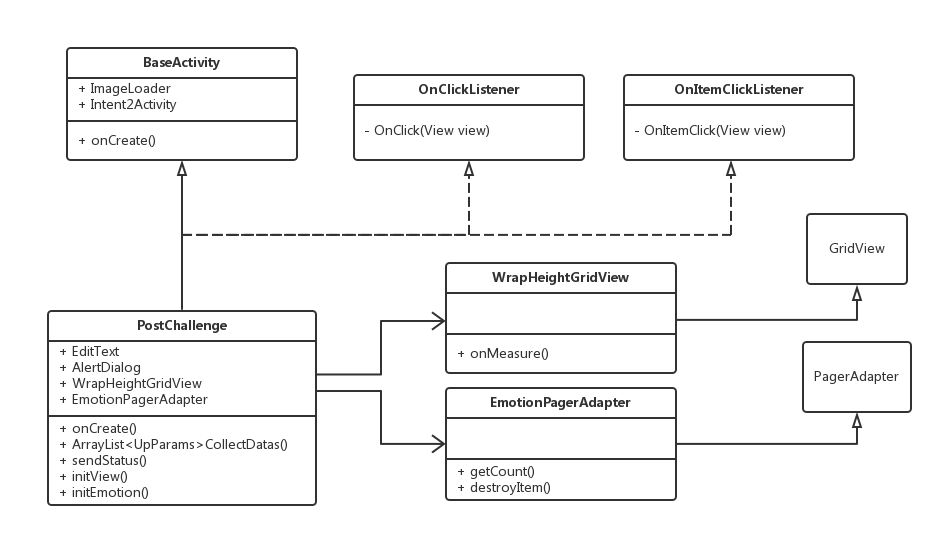
OnClickListener: 用于注册点击事件

OnCheckedChanged: 监听RadioGroup变化

1. **PostChallenge.java**

功能：完成挑战上传的功能

依赖关系：

****

变量、函数说明：

EditText: 用于完成提交挑战页面的输入

AlertDialog: 用于完成页面提交时候的弹框提示信息

WrapHeightGridView: 用于完成挑战内容提交的时候，显示挑战的图片

EmotionPagerAdapter: 用于表情输入框的容纳

OnCreate() 页面初始化

CollectDatas() 收集需要上传的信息，存为键值对

SendStatus() 向服务器发送

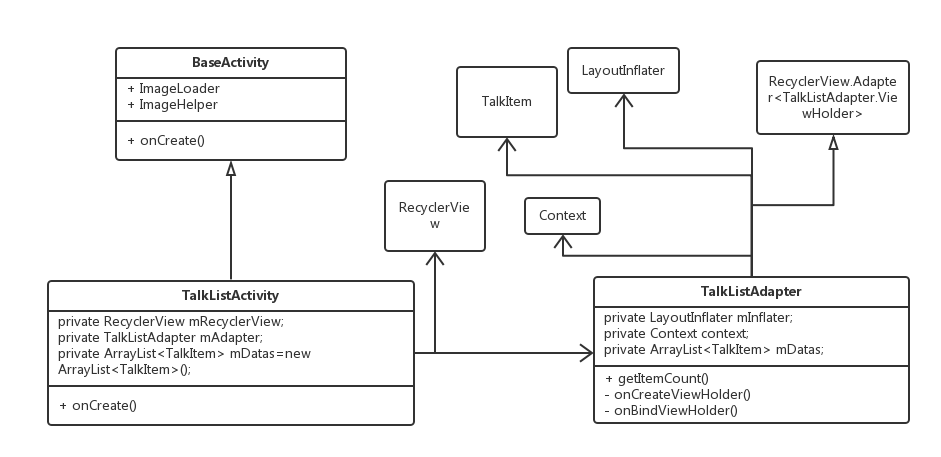
initView() 初始化页面

initEmotion() 初始化表情面板

1. **TalkListActivity.java**

功能：展示特定版块下的帖子

依赖关系：



变量、函数说明：

RecycleView：Android 5.0基类，用于实现列表效果

TalkListAdapter：基础Adapter类，用于完成列表样式和数据的绑定

ArrayList<TalkItem>：需要绑定的数据

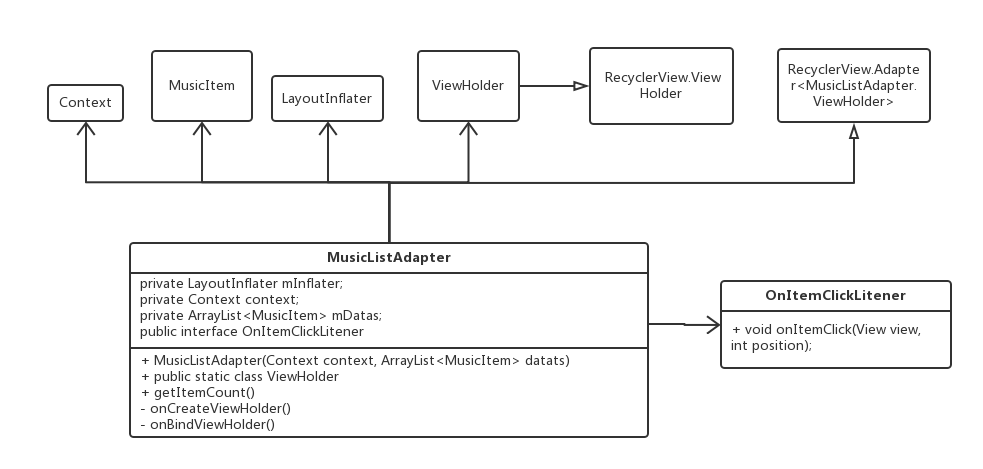
OnCreate()：初始化样式

# 3.2 Adapter包

1. **MusicListAdapter.java**

功能：实验音乐列表数据和音乐列表样式的绑定

依赖关系：



变量、函数说明

LayoutInflater: 实现样式绑定功能

Context：上下文

mData：需要绑定的数据

OnItemClickListener：MusicListAdapter开放出来的接口，用于从外部调用的时候响应点击列表数据的位置。

MusicListAdapter()：构造函数

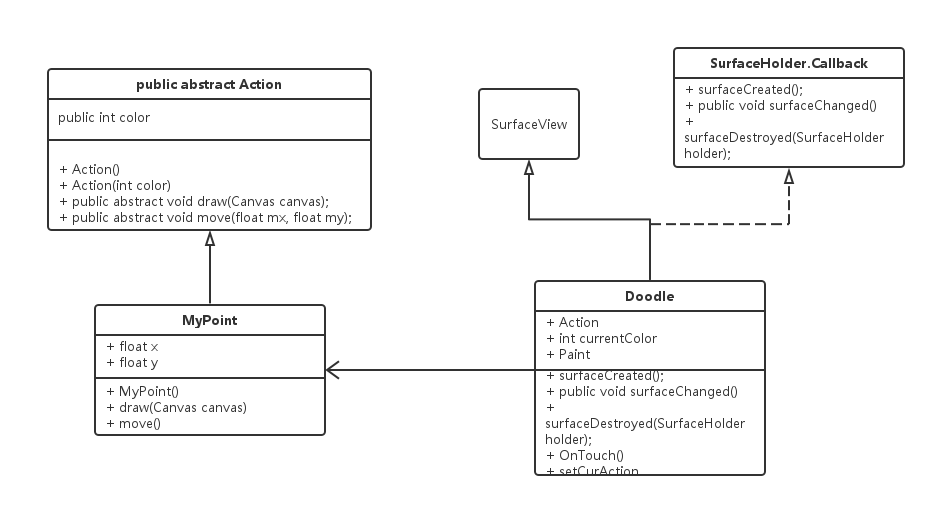
ViewHolder: 内部类，用于定义每个列表元素里面都包含哪些子元素

# 3.3 Constants包

**(1) Action.java**

功能：实现画图的基本动作监视

依赖关系：



变量、函数说明：

Action

color：颜色的值

Action() 动作的初始化

Draw() 绘图的动作开始的时候触发的函数

Move() 绘图的动作进行时触发的函数

MyPoint

Float x 绘制的x坐标

Float y 绘制的y坐标

MyPoint() 初始化

Draw 刚要进行绘制的时候触发的函数

Movw 正在绘制的时候触发的函数

Doodle

Action 每个绘图对象(Doodle) 都具备一个Action对象用于监听其动作

surfaceCreated 绘图区域创建的时候触发

SurfaceChanged 绘图区域发生变化的时候触发

SurfaceDestroyed 绘图对象销毁调用的对象

OnTouch 主画板被触摸的时候触发

SetCurAction 设置画笔的形状

1. **MusicItem.java**

功能：实现接收网络数据的实例化。这是一个众多通用的类派生出来类的一种，也即：所有从网络获取的数据（评论、挑战等）也都是利用类似的\*Item.java来存储。

依赖关系：无

变量、函数说明：

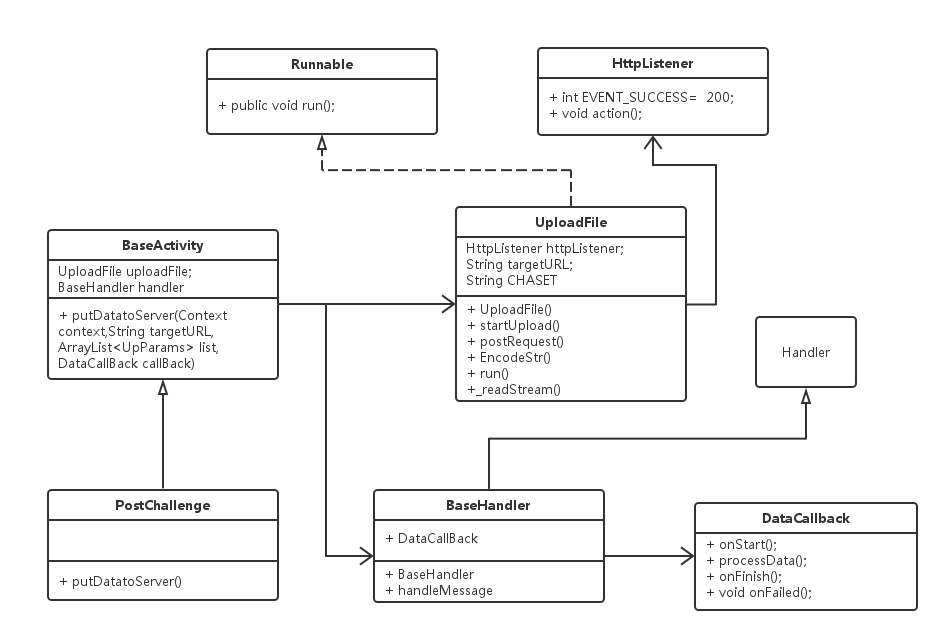
private int id; //音乐id  
 private String name; //音乐的名称  
 private String cover; //音乐的封面  
 private String path; //音乐文件的路径  
 private String singer; //音乐的歌手名  
 private String type; //音乐的类型

# 3.4 File包

1. **UploadFile.java**

功能说明：用于Android端向服务器发送非字符串型数据的时候(例如向服务器发送图片等)所使用到的上传类。

依赖关系：

****

变量、函数说明：

UploadFile继承Runnable，依附于BaseActivity而存在，所以UploadFile不能够独立于Activity而完成其功能。

UploadFile

HttpListener: 用于定义Http请求返回的代号

targetURL: 用于定义需要发送请求的URL

CHASET 编码格式

UploadFile() 构造函数

StartUpload 开始上传

postRequest 发送请求

EncodeStr 将字符串以URLEncode编码

Run 实现Runnable

ReadStream 读取接口返回的字符

BaseHandler

DataCallback 用于根据存储接口传回字符对应的状态，改变listener属性

BaseHandler 构造函数

HandleMessage 继承自Handler所需要实现的

DataCallBack 请求的状态

onStart() 开始

onProcess() 进行中

onFinish() 结束

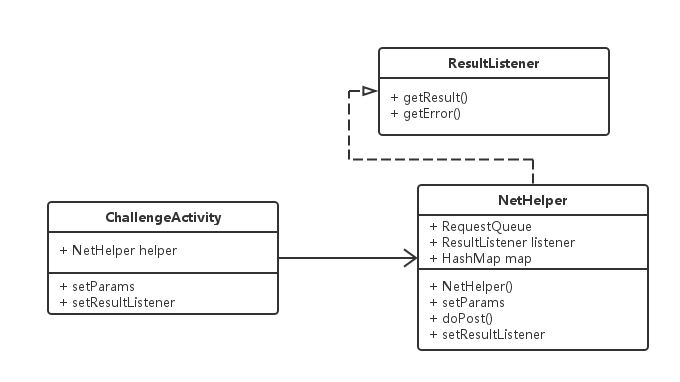
onFailed() 失败

# 3.5 Helper包

1. **NetHelper.java**

功能：用于处理字符串类请求

依赖关系：



变量、函数说明：

NetHelper用于发起网络请求（仅限于字符串请求），不能独立于Activity单独实现请求，需要在Activity中引用

NetHelper

RequestQueue 请求队列

ResultListener 请求结果监听

HashMap map中存放了用于发送到服务器端的键值对

NetHelper()构造函数

SetParams() 用于添加传入服务器的键值对

doPost() 向服务器发起请求

SetResultListener() 为NetHelper对象设置监听器

ResultListener

getResult() 网络请求正常的时候获取的结果

getError() 网络请求错误的时候

# 3.6 Util包

Util包中大多封装了工厂模式的一些方法，这些方法可以从外部直接调用，通过传入不同的数据可以返回不同的值。所有Util中的文件大多遵循这一点，下面举几个有代表性的例子。

**(1) TimeUtils.java**

功能：用于完成对时间的处理，比如获取当前时间，或者获取两个时间之间的时间差

依赖关系：无

变量：无

函数说明：

TimeUtils

getDay() 获取当前的日期

getCurrentTime() 获取当前系统时间，返回一串字符串表示当前时间距离1970年1月1日的时间长度

getDateDifftime(Date date1,Date now) 获取当前系统时间now与输入时间date1的时间差，返回字符串将以“ \*\*\* 之前”的形式表示。如：“2小时 之前”。

1. **NetUtils.java**

功能：用于完成对网络状态的判断，比如获取当前网络是否可用。

依赖关系：无

变量：无

函数说明：

isNetworkAvailable() 用于判断网络状态的可用性

1. **TitleBuilder**

功能：用于完成对ActionBar(界面上最上方位置的控件)的初始化。

依赖关系：无

变量：

private TextView tvTitle; 标题文字

private ImageView ivLeft; 左侧图像  
private ImageView ivRight; 左侧文字  
private TextView tvLeft; 右侧图像  
private TextView tvRight; 右侧文字

函数说明：

TitleBuilder() 构造函数

SetTitleText() 设置标题文字

SetLeftImage() 设置左侧图片

SetLeftText() 设置左侧文字

SetRightImage() 设置右侧图像

SetRightText() 设置右侧文字

SetLeftOnClickListener() 设置左侧按钮点击事件

SetRightOnClickListener() 设置右侧按钮点击事件

# 四、后台代码设计

# 4.1 后台接口代码

1. **getChallengeDetail.php**

获得某一个用户发起的某一个“你画我猜”挑战的具体信息，是安卓端的接口，在安卓端用户点击某个挑战时调用。

1. **getChallengeList.php**

获得所有正在进行的“你画我猜”挑战，是安卓端的接口，在安卓端用户进入挑战页面时调用。

1. **getMusic.php**

获得当前用户需要播放的音乐的歌曲文件和封面图片，是安卓端的接口，在安卓端用户选择歌曲点击播放时调用。

1. **getTags.php**

获得当前用户选择播放的音乐的标签，是安卓端的接口，在安卓端用户选择歌曲点击播放时调用。

1. **getTalkList.php**

获得社区所有用户发出的帖子信息，是安卓端的接口，在安卓端用户进入社区时调用。

1. **jsonHelper.php**

json字符串的工具类，用于生成和解析json字符串，在每次向安卓端返回数据时调用。

1. **sql.php**

数据库连接的配置文件，被每个需要调用数据库的接口类引用。

1. **upChallenge.php**

接受安卓端用户发布的一个“你画我猜”挑战数据，将相关图片文件和具体数据保存到服务器和数据库。

# 4.2 后台管理页面代码

1. **Index.php**

后台管理系统主页，用于后台管理人员上传音乐进行测试。

1. **upMusic.php**

接收后台管理系统上传的音乐及其图片，进行相关数据处理操作。

1. **checkFinish.php**

检查歌曲及其封面图是否上传完成，并获取相应参数信息存入数据库，在后台管理系统上传歌曲时调用。

## 五、数据存储设计

## 5.1数据库设计

详见《软件概要设计文档》所述。

## 5.2视图的设计

1. **getChallengeDetail视图**

视图含义：获取挑战详情，供用户接受挑战使用

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段含义 | 字段类型 |
| id | 挑战的id | Int |
| Frog\_path | 雾霾图的路径 | Varchar |
| Path | 清晰挑战图的路径 | Varchar |
| User\_id | 发起挑战的用户id | Int |
| Account | 用户账户 | Varchar |
| Avatar | 用户等级对应的头像 | Varchar |
| Introduce | 用户简介 | Varchar |
| Tag | 挑战图对应的标签，也是进行猜测的正确答案 | Varchar |
| Music | 挑战对应的歌曲 | Varchar |
| Singer | 歌曲的歌手名 | Varchar |
| Type | 歌曲的类型 | Varchar |

1. **getChallengeList视图**

视图含义：获取挑战列表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段含义 | 字段类型 |
| id | 挑战的id | Int |
| Challenge\_name | 挑战的一句话简介 | Varchar |
| Upper\_time | 挑战上传的时间 | Varchar |
| Avatar | 上传挑战的人的头像 | Varchar |
| Upper\_name | 上传挑战的人的名字 | Varchar |

1. **getCommentNum 视图**

视图功能：获取每一条帖子对应的回复数量

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段含义 | 字段类型 |
| id | 帖子的id | Int |
| Comment\_num | 帖子评论的数量 | Int |

1. **getTags视图**

视图功能：获取标签列表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段含义 | 字段类型 |
| id | 标签的id | Int |
| Tag | 标签的名称 | Varchar |
| Music | 标签对应的音乐 | Varchar |

1. **getTalkList视图**

视图功能：获取帖子的列表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段含义 | 字段类型 |
| id | 帖子的id | Int |
| Section\_id | 版块的id | Int |
| User\_id | 发布帖子的人的id | Int |
| Comment\_num | 评论数量 | Int |
| Account | 用户账户 | Varchar |
| Avatar | 发帖人头像 | Varchar |
| Time | 发帖时间 | Varchar |
| Title | 帖子标题 | Varchar |
| Content | 帖子内容 | Varchar |

1. **getUserLevel视图**

视图功能：获取用户当前经验值对应的等级

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段含义 | 字段类型 |
| User\_id | 用户的id | Int |
| level\_id | 等级的id | Int |

1. **getUserLevelInfo视图**

视图功能：获取当前用户基于经验值的个人信息

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段含义 | 字段类型 |
| id | 用户的id | Int |
| Current\_exp | 用户当前的经验值 | Int |
| Level\_exp | 用户当前等级对应的最低经验值 | Int |
| Avatar | 用户当前经验值对应的头像 | Varchar |
| Brand | 用户当前经验值对应的头衔 | Varchar |

## 5.3数据的物理存储

所有数据表统一存储在服务器的数据库。客户端启动后第一次请求时，服务器给出完整响应，同时客户端在本地留存一份副本。以后进行相同的请求时，客户端一边向服务器发出请求，一边显示本地副本；当收到服务器的响应后，如果与本地副本存在差异，则同时更新本地副本和显示。

这样可以有效提高响应速度，改善性能。

# 六、用户界面设计

参考《软件概要设计文档》所述。

# 七、参考资料

|  |  |  |
| --- | --- | --- |
| **序号** | **编号** | **文档名称** |
| 1 | 004-XM10 | “娱音绕梁”项目开发任务书 |
| 2 | 004-XQ10 | “娱音绕梁”项目需求说明书 |
| 3 | 004-GS10 | “娱音绕梁”项目概要设计文档 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |