目录

[一、通用模块-common 1](#_Toc479763117)

[1.1 功能简述 1](#_Toc479763118)

[1.1.1 注册账户 1](#_Toc479763119)

[1.1.2 注销账户 2](#_Toc479763120)

[1.1.3 修改账户信息 2](#_Toc479763121)

[1.1.4 查询账户信息 3](#_Toc479763122)

[1.1.5 登录验证 4](#_Toc479763123)

[1.2 数据库设计 5](#_Toc479763124)

[1.2.1 数据库架构图 5](#_Toc479763125)

[1.2.2 数据库详细说明 5](#_Toc479763126)

[附录A 状态码 6](#_Toc479763127)

[模块标识码 6](#_Toc479763128)

[功能标识码 6](#_Toc479763129)

# 一、通用模块-common

## 1.1 功能简述

该系统采用的是RBAC（role-based access control，基于角色的访问控制），因此账户模块被融入到通用模块之中(不包含商家账户)。

### 1.1.1 注册账户

该模块针对底层实现来说是对账户的添加操作，对使用者而言就是注册操作。

**注册必填信息包括**：

* 账户名
* 密码
* 性别
* 手机号

**注册选填信息包括**：

* 邮箱
* 头像图片

**系统生成信息包括**：

* 账户ID(UUID)
* 注册时间
* 账户状态

注意事项：邮箱没有默认值，而头像图片有指定的默认值。手机号是固定长度的11位数字，注册时间是禁止修改的。

无法注册情况：

* 账户名已存在
* 账户名为手机号
* 账户名为邮箱

#### 返回信息格式(JSON)

{"code":"0","message":"success","data":{}} //code状态码，message信息，data返回数据

{"code":"0","message":"success","data":[]} // data 为 array 的情况

#### 状态码及对应信息

|  |  |  |
| --- | --- | --- |
| **状态码** | **信息** | **返回data格式** |
| 000100 | 账户注册成功 |  |
| 000101 | 账户已存在，注册失败 |  |
| 000102 | 账户名为手机号，注册失败 |  |
| 000103 | 账户名称为邮箱，注册失败 |  |
| 000104 | 系统异常导致账户注册失败 |  |

### 1.1.2 注销账户

该模块针对底层实现来说是对账户的删除操作，对使用者而言就是注销操作。

删除操作提供三类操作：

* 删除权限：当某种权限不再需要的时候
* 删除角色：当某种角色不再需要的时候，或者说用户的某个角色注销的时候
* 删除账户：单独某个账户的清除，包含其权限和角色信息。可通过账户的ID，账户名称两种方式进行删除操作。

#### 返回信息格式(JSON)

{"code":"0","message":"success","data":{}} //code状态码，message信息，data返回数据

{"code":"0","message":"success","data":[]} // data 为 array 的情况

#### 状态码及对应信息

|  |  |  |
| --- | --- | --- |
| **状态码** | **信息** | **返回data格式** |
| 000200 | 账户移除成功 |  |
| 000201 | 账户的角色移除成功 |  |
| 000202 | 权限不足，无法移除账户的角色 |  |
| 000203 | 权限不足，无法删除账户 |  |
| 000204 | 该类角色移除成功(保留账户) |  |
| 000205 | 该类角色的账户移除成功(保留角色) |  |
| 000206 | 该类角色及其账户移除成功 |  |
| 000207 | 系统异常导致账户移除失败 |  |
| 000208 | 系统异常导致账户的角色移除失败 |  |
| 000209 | 系统异常导致导致该类角色移除失败 |  |
| 000210 | 系统异常导致该类角色的账户移除失败 |  |
| 000211 | 系统异常导致该类角色的账户移除失败 |  |

### 1.1.3 修改账户信息

该模块针对底层实现来说是对账户的更新操作，对使用者而言就是修改个人信息操作。

**可修改的个人信息包括**：

* 密码
* 性别
* 手机号
* 邮箱、
* 头像图片以及
* 账号的状态
* 角色

**禁止修改的个人信息包括**：

* 账户ID、
* 账户名称
* 注册时间

#### 返回信息格式(JSON)

{"code":"0","message":"success","data":{}} //code状态码，message信息，data返回数据

{"code":"0","message":"success","data":[]} // data 为 array 的情况

#### 状态码及对应信息

|  |  |  |
| --- | --- | --- |
| **状态码** | **信息** | **返回data格式** |
| 000300 | 账户信息修改成功 |  |
| 000301 | 包含禁止修改信息，修改失败 |  |
| 000302 | 账户密码修改成功 |  |
| 000303 | 账户角色修改成功 |  |
| 000304 | 账户基本信息修改成功 |  |
| 000305 | 密码修改异常导致修改失败 |  |
| 000306 | 角色修改异常导致修改失败 |  |
| 000307 | 基本信息异常导致修改失败 |  |
| 000308 | 系统异常导致修改失败 |  |

### 1.1.4 查询账户信息

该模块针对底层实现来说是对账户的查询操作，对使用者而言就是浏览账户信息操作。该模块是进行账户查询操作，返回的是账户的信息。

#### 返回信息格式(JSON)

{"code":"0","message":"success","data":{}} //code状态码，message信息，data返回数据

{"code":"0","message":"success","data":[]} // data 为 array 的情况

#### 状态码及对应信息

|  |  |  |
| --- | --- | --- |
| **状态码** | **信息** | **返回data格式** |
| 000400 |  |  |
| 000401 |  |  |

### 1.1.5 登录验证

该模块针对底层实现来说是对账户的查询操作，对使用者而言就是登录账户验证操作。只不过这个操作是一种比较常用的查询账户信息操作，因此将其单独进行设计，与上面的查询账户信息模块不同之处是：该模块只针对单独账户，返回的只是该角色的权限及基本信息。

验证方式可以通过手机号、邮箱、账户这三种方式进行登录验证，而用户无须手动选择，只需要填写登录信息即可，这部分是由后台进行区分并处理的。

如果登录验证成功，会向t\_logininfo插入一条登录信息

#### 返回信息格式(JSON)

{"code":"0","message":"success","data":{}} //code状态码，message信息，data返回数据

{"code":"0","message":"success","data":[]} // data 为 array 的情况

#### 状态码及对应信息

|  |  |  |
| --- | --- | --- |
| **状态码** | **信息** | **返回data格式** |
| 000500 | 账号验证成功 |  |
| 000501 | 账户验证失败：账号不存在 |  |
| 000502 | 账户验证失败：密码错误 |  |
| 000503 | 系统异常导致账号验证失败 |  |
| 000504 | 账号验证成功，但是登录信息保存失败 |  |

### 1.1.6 登录信息记录

该模块主要功能是记录用户所有登录操作，无论登录账号是否存在，密码是否正确。该表的主要目的：

* 筛选出长期不登录账号
* 检测异常登录IP及设备

#### 返回信息格式(JSON)

{"code":"0","message":"success","data":{}} //code状态码，message信息，data返回数据

{"code":"0","message":"success","data":[]} // data 为 array 的情况

#### 状态码及对应信息

|  |  |  |
| --- | --- | --- |
| **状态码** | **信息** | **返回data格式** |
| 000600 | 账号密码匹配正确：登录信息记录成功 |  |
| 000601 | 账号存在：密码错误，登录信息记录成功 |  |
| 000602 | 账号不存在：登录信息记录成功 |  |
| 000603 | 系统异常导致登录信息记录失败 |  |

### 1.1.7 账户信息修改记录

该模块主要功能是记录账户信息的修改等。由于账户可能被删除，但是这些记录信息是不可以消除的，所以数据库底层将他们并没有直接关联，但是他们确是高度同步的。

#### 返回信息格式(JSON)

{"code":"0","message":"success","data":{}} //code状态码，message信息，data返回数据

{"code":"0","message":"success","data":[]} // data 为 array 的情况

#### 状态码及对应信息

|  |  |  |
| --- | --- | --- |
| **状态码** | **信息** | **返回data格式** |
| 000700 | 密码修改成功:账户修改信息记录成功 |  |
| 000701 | 性别修改成功:账户修改信息记录成功 |  |
| 000702 | 手机号修改成功:账户修改信息记录成功 |  |
| 000703 | 邮箱修改成功:账户修改信息记录成功 |  |
| 000704 | 头像修改该成功:账户修改信息记录成功 |  |
| 000705 | 账号状态修改成功:账户修改信息记录成功 |  |
| 000706 | 角色修改成功:账户修改信息记录成功 |  |
| 000707 | 修改禁止修改信息：账户修改信息记录成功 |  |
| 000708 | 系统异常导致账户修改信息记录失败 |  |

## 1.2 数据库设计

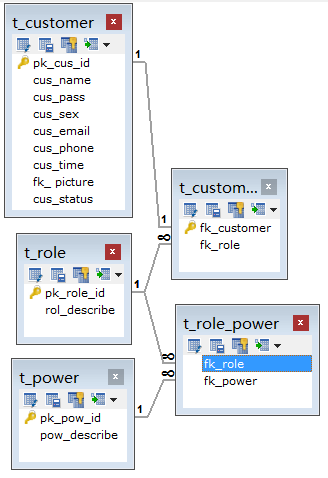
### 1.2.1 数据库设计思路

系统采用的是RBAC（role-based access control，基于角色的访问控制）权限管理，每个人可以有多个角色，每个角色又以有多种权限，反过来同一个权限可以对应多个角色，同一个角色可以对应多个人，因此这三者是多对多的关系，因此需要采用中间表进行设计。

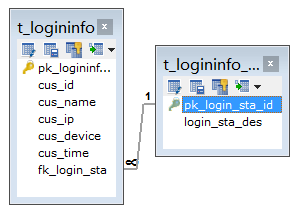
登录是最常用的操作，对登录信息进行记录也是必要的。设计初衷是检测异常登录，以及剔除长期未登录用户。

### 1.2.2 数据库架构图

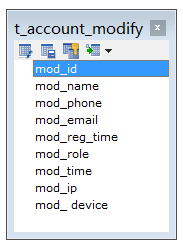
#### <1> RBAC结构



#### <2> 登录信息记录



#### <3> 账户信息修改记录



### 1.2.3 数据库详细说明

t\_customer为账户表，保存的是账户的基本信息；t\_role为角色表，保存的只是该系统包含的角色；t\_power，保存的是账户的权限。

t\_custome\_role是账户与角色的关联表；t\_role\_power是角色与权限的关联表。

t\_logininfo与t\_account\_modify是信息记录表，主要是记录用户的登录信息，以及账户修改信息记录。

# 附录A 状态码

状态码采用六位数字构成，其由三部分组成，每部分又由两个数字构成。

其构成格式为：模块标识码+功能标识码+信息标识码

注：信息标识码可详见各模块的功能说明。

## 模块标识码

|  |  |
| --- | --- |
| **模块** | **标识码** |
| common | 00 |

## 功能标识码

|  |  |
| --- | --- |
| **模块** | **标识码** |
| 注册账户 | 01 |
| 注销账户 | 02 |
| 修改账户信息 | 03 |
| 查询账户信息 | 04 |
| 登录验证 | 05 |
| 登录信息记录 | 06 |
| 账户信息修改记录 | 07 |