目录

[一、通用模块-common 1](#_Toc480361609)

[1.1 PO类设计 1](#_Toc480361610)

# 通用模块-common

## 1.1 包结构

org.common.RBAC：该包下是关于RBAC（基于角色的访问控制）的所有操作

org.common.RBAC.domain：该包下是所有数据库表对应的PO(持久化类)类

org.common.RBAC.dao：该包下保存的是所有DAO组件的接口

org.common.RBAC.dao.impl：该包下保存的是所有DAO组件的实现

## 1.2 PO(持久化)类设计

### 1.2.1 RBAC模块

从数据库建模角度来看账户、角色、权限三者中，账户与角色是N-N关系，而角色与权限是N-N关系。

从面相对象建模角度来看，账户管理与角色的关联关系，角色管理与权限的关联关系，因此账户与角色是单向N-N关系，角色和权限是单向N-N关系。

* Account：账户基本信息，对应底层数据库的t\_customer表
* Role：角色信息，对应底层的t\_role表
* Power：权限信息，对应底层的t\_power表

账户、角色、权限三者是基于中间表进行关联的，而使用Account管理与Role的关联，而Role管理与Power的关联。

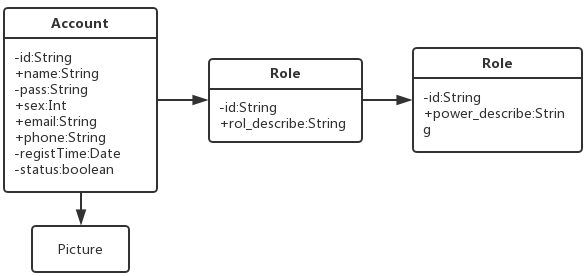


图 1

Account账户还单向关联图片表实体，用来保存用户头像。

### 1.2.2 信息记录模块

#### <1>登录信息记录

从数据库建模角度来看，这部分不与其它表进行关联。而不与账户关联的主要目的是记录那些不存在的账户以及已经删除的账户所产生的登录历史。

从面相对象建模角度来看，LoginInfo与LoginInfoStatus是单向N对1的关系。

* LoginInfo：登录信息记录，对应底层数据库的t\_logininfo表
* LoginInfoStatus：登录信息中记录登录状态，对应底层数据库的t\_logininfo\_ status表

LoginInfoStatus存在的意义就是用来区别登录信息所产生的状态，例如：账号不存在，密码错误等。

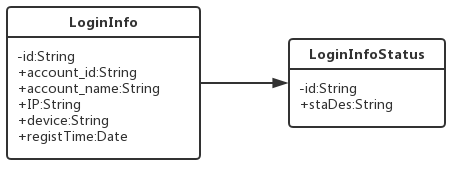


图 2

#### <2>账户信息修改记录

从数据库建模来看，这部分不与其它表进行关联。其不与账户关联的原因，与上面一样。只不过该部分针对的角度是账户信息修改。

* account\_modify

## 1.3 DAO组件设计

### 1.3.1 RBAC模块

采用面向接口的方式进行定义，并将其中将所有通用的操作提取出来。下面是各接口的详细定义：

#### <1>BaseDao接口

该接口包含了所有的通用的操作，其包含以下几个方法：

* T get(Class<T> entityClazz, Serializable id)：根据ID加载实体
* Serializable save(T entity)：保存实体
* update(T entity)：更新实体
* void delete(T entity)：删除实体
* void delete(Class<T> entityClazz, Serializable id)：根据ID删除实体
* List<T> findAll(Class<T> entityClazz)：获取所有实体
* long findCount(Class<T> entityClazz)：获取所有实体总数

注：所有org.common.RBAC.dao下定义的接口都继承这个接口