

“课堂银行” 系统设计文档

200110611 王志铭 | 200110607 吴泽楷 | 200110610 刘颜铭

一、项目结构设计

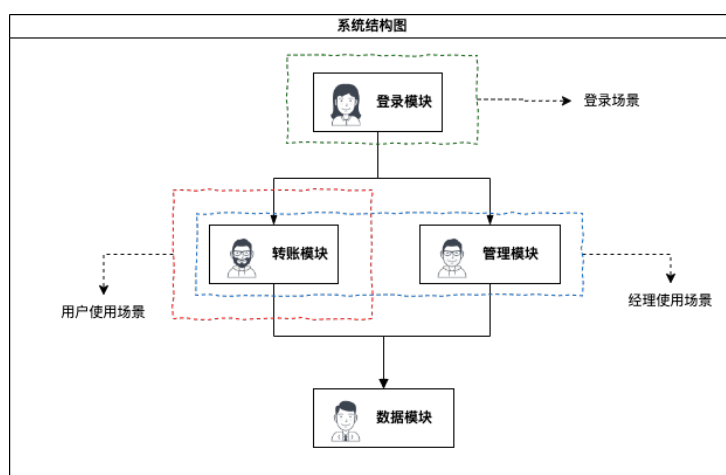
我们根据系统的场景来设计本项目模块的结构。

银行系统设计了两类交互场景：登录场景、使用场景。特别注意的是，使用场景还细分两类：面向用户的场景功能为转账，而面向经理的场景功能为管理、转账。

基于以上的场景设计，我们可以抽象出三个功能模块：登录模块、转账模块、管理模块，以及一个底层的数据模块。

登录模块主要负责账户的校验和用户信息保存；转账模块主要负责实现发起转账、查看转账明细；管理模块主要负责账户的查询、注册、删除等操作；而数据模块则主要是数据库表结构的设计。

项目模块结构示意图如下所示。



二、项目具体实现

基于上面的设计，我们自底向上地介绍不同模块的具体实现：

1. 数据模块

首先是用户模型，主要记录了用户的帐户信息。我们经过讨论后认为，普通用户和经理的区别主要在于账户管理权限，其他的功能是相同的。因此设计为同一模型，区别在于 role 字段的不同。这样实现也有利于登录页面的统一。

然后是转账记录模型，包含了一条转账的 from 方、to 方、金额、时间。

2. 转账模块

转账模块实现了从 A 到 B 的转账操作。

此模块并非简单的 A 账户扣钱 B 帐户加钱的操作，更重要的是在于保持数据的一致性，也就是不允许出现 A 扣钱 B 不加钱，或者 A 不扣 B 加钱的情况。

我们通过事务来实现了上面提到的特性。由于项目使用了 SpringBoot 框架，我们最终选择通过 @Transactional 在业务层实现事务，避免了与数据库层的业务耦合。

3. 管理模块

管理模块是仅限经理使用的模块。

经理登录后可以看到管理菜单，而用户只能看到个人信息和转账菜单。

基于扁平化管理结构的考量，我们规定一个经理的管理面向了所有的经理和用户。在具体实现时，经理通过发起查询/增加/删除请求，带上对应的 id，在 controller 当中进行对应的数据库 select/update/delete。在具体的方案上，我们使用到了 MyBatisPlus。

注册时会进行 username 的检查，在业务层保证其唯一性。

4. 登录模块

基于数据模块的设计，我们把用户和经理的登录做了统一。

在登录逻辑上，由于 username 是唯一的，我们通过 (username, password) 确定一个帐号。

具体实现上要简单一些，在 user 表当中查询 (username, password) 对应的帐号，判断是否登录成功。

三、额外功能实现

1. 基于 Session 的状态保存

在登录时，我们将用户的信息保存在 Session 当中。这样使得不同的交互模块都能复用到登录时的用户信息（如 id 等），既避免了用户非法输入，也避免了转账时的信息重填，提升用户体验。

Key	Value
CurUser	{"rid":2,"name":"manager2","age":32,
CurUserType	manager

2. 动态 menu 实现

在整个界面设计当中，我们遵循了「高内聚、低耦合」的设计思想。用户和经理使用的页面菜单略有不同，我们设计了 menu 模型，为每个子菜单设计了 menuRight 权限。相比于双界面固定菜单，这样的设计一是有高扩展性，不需要再修改 base page，二是方便鉴权，普通用户无法访问到经理的界面。

3. 管理模块功能扩展

除了项目书要求的注册、删除，我们还添加了修改功能。并且先做一次预查询，方便了经理只修改少部分信息的情况。

4. 提前检验余额

转账时经常出现 from 方余额不足的情况，此时我们先前设计的事务已经能够保证回滚。不过我们还额外设计了提前余额检验，将转账流程的粒度细化，不必每次都开事务，提升了系统服务的性能。

5. 刷新保留页面

项目使用了 Nginx，在实现刷新保留页面时，我们把 menu 存到 local storage 里（用插件），监听 reload 时 menu 变化时再设置一次动态路由，然后服务端部署 nginx 配置文件里允许奇怪的 location。

6. 转账明细查询鉴权

在查询转账明细时，我们额外设置了权限约束：用户看不到与自己无关的明细，只有经理能看到所有。具体的实现是通过附带 id 的数据库查询。