**灯具公司进销存系统**

LSLS(Lighting company Longevity system)

体系结构描述文档

V3.0 正式版

南京大学SIF4工作组

2017-11-12

目录

[更新历史 2](#_Toc496468776)

[1.引言 2](#_Toc496468777)

[1.1编制目的 2](#_Toc496468778)

[1.2词汇表 2](#_Toc496468779)

[1.3参考资料 2](#_Toc496468780)

[2.产品概述 2](#_Toc496468781)

[3.逻辑视角 2](#_Toc496468782)

[4.组合视角 4](#_Toc496468783)

[4.1开发包图 4](#_Toc496468784)

[4.2运行时进程 7](#_Toc496468785)

[4.3物理部署 7](#_Toc496468786)

[5.接口视角 8](#_Toc496468787)

[5.1模块的职责 8](#_Toc496468788)

[5.2用户界面层的分解 11](#_Toc496468789)

[5.2.1 用户界面层模块的职责 13](#_Toc496468790)

[5.2.2 用户界面层模块的接口规范 13](#_Toc496468791)

[5.2.3 用户界面模块设计原理 13](#_Toc496468792)

[5.3业务逻辑层的分解 13](#_Toc496468793)

[5.3.1 业务逻辑层模块的职责 14](#_Toc496468794)

[5.3.2 业务逻辑层模块的接口规范 14](#_Toc496468795)

[5.4数据层的分解 16](#_Toc496468796)

[5.4.1 数据层模块的职责 16](#_Toc496468797)

[5.4.2 数据层模块的接口规范 17](#_Toc496468798)

[6.接口视角 17](#_Toc496468799)

[6.1数据持久化对象 17](#_Toc496468800)

[6.2 Txt持久化格式 18](#_Toc496468801)

[6.3 数据库表 18](#_Toc496468802)

# 更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| **修改人员** | **日期** | **变更原因** | **版本号** |
| SIF4 | 2017-10-19 | 最初设计 | V1.0 正式版 |
| SIF4 | 2017-11-07 | 接口描述完善 | V2.0 正式版 |
| SIF4 | 2017-11-12 | Data层接口补充 | V3.0 正式版 |

# 1.引言

### 1.1编制目的

本报告详细完成对灯具公司进销存系统的概要设计，达到指导详细设计和开发的目的，同时实现测试人员及用户的沟通。

本报告面向开发人员、测试人员及最终用户编写，是了解系统的导航。

### 1.2词汇表

|  |  |  |
| --- | --- | --- |
| 词汇名称 | 词汇含义 | 备注 |
| LSLS | 灯具公司进销存系统 | …… |
| …… | …… | …… |

### 1.3参考资料

# 2.产品概述

参考灯具公司进销存系统用例文档和灯具公司进销存系统软件需求规格说明中对产品的概括描述。

# 3.逻辑视角

灯具公司进销存系统中，选择了分层体系结构风格，将系统分为3层（展示层、业务逻辑层、数据层）能够很好地示意整个高层抽象。展示层包含GUI界面的实现，业务逻辑层包含业务逻辑处理的实现，数据层负责数据的持久化和访问。分层体系结构的逻辑视角和逻辑设计方案如图1和图2所示。

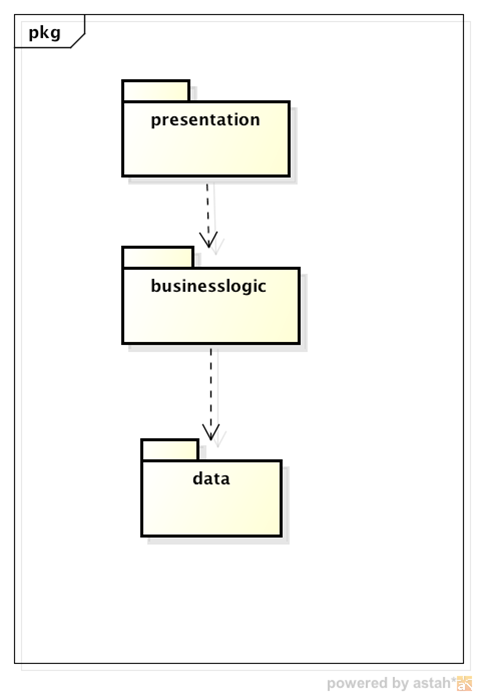


图1 参照体系结构风格的包图表达逻辑视角

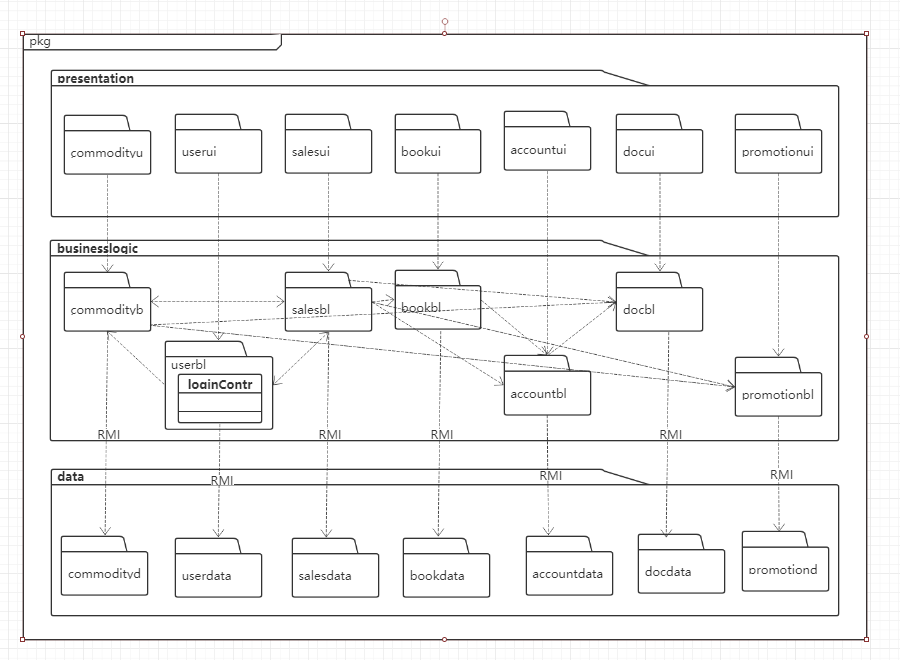


图2 软件体系结构逻辑设计方案

# 4.组合视角

### 4.1开发包图

LSLS的最终开发包设计如表1所示。

表1 LSLS的最终开发包设计

|  |  |
| --- | --- |
| 开发（物理）包 | 依赖的其他开发包 |
| mainui | userui, salesui, commodityui, bookui, accountui, docui, promotionui,vo |
| salesui | salesblservice,界面类库包,vo |
| salesblservice |  |
| salesbl | salesblservice,salesdataservice,po,promotionbl,userbl |
| salesdataservice | Java RMI,po |
| salesdata | databaseutility,po,salesdataservice |
| commodityui | commodityblservice,界面类库包 |
| commodityblservice |  |
| commoditybl | commodityblservice, commoditydataservice,po,salesbl |
| commoditydataservice | Java RMI,po |
| commoditydata | Java RMI,po,databaseutility |
| promotionui | promotionblservice,界面类库包 |
| promotionblservice |  |
| promotionbl | promotionblservice,promotiondataservice,vo |
| promotiondataservice | Java RMI,po |
| promotiondata | Java RMI,po,databaseutility |
| userui | userblservice,界面类库包 |
| userblservice |  |
| userbl | UserInterface,UserDataClient,UserPO |
| userdataservice | Java RMI,po |
| userdata | RMI,po,databaseutility |
| bookui | bookblservice,界面类库包 |
| bookblservice |  |
| bookbl | bookblservice,bookdataservice,vo |
| bookdataservice | Java RMI,po |
| bookdata | Java RMI,po,databaseutility |
| accountui | accountblservice,界面类库包 |
| accountblservice |  |
| accountbl | accountblservice, accountdataservice,vo |
| accountdataservice | Java RMI,po |
| accountdata | Java RMI,po,databaseutility |
| docui | docblservice,界面类库包 |
| docblservice |  |
| docbl | docblservice, docdataservice,vo |
| docdataservice | Java RMI,po |
| docdata | Java RMI,po,databaseutility |
| vo |  |
| po |  |
| utilitybl |  |
| 界面类库包 |  |
| Java RMI |  |
| databaseutility | JDBC |

LSLS客户端开发包图如图3所示，服务器端开发包图如图4所示。

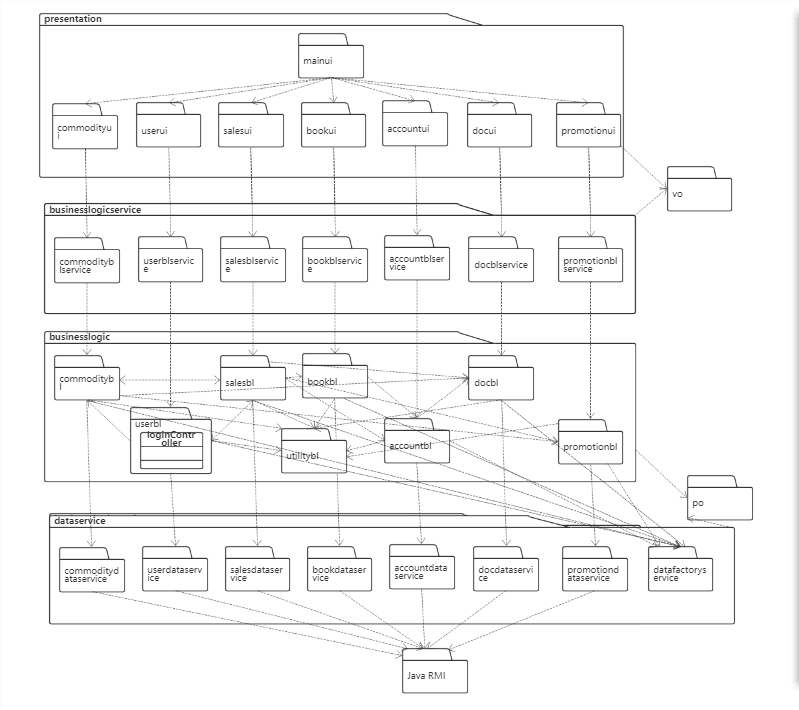


图3 LSLS客户端开发包图

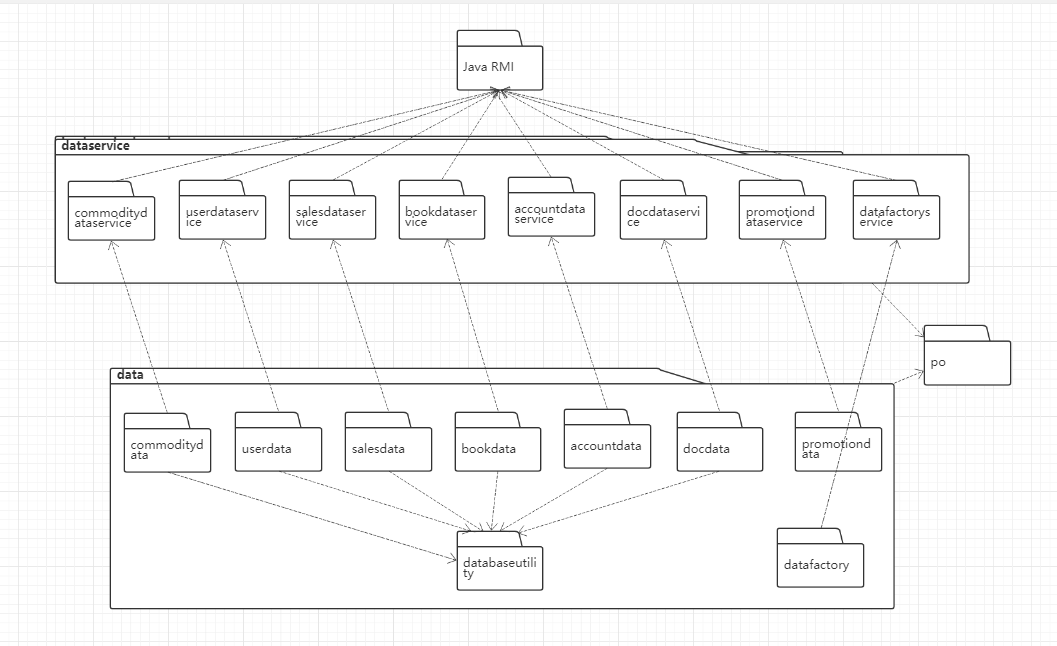


图4 LSLS服务器端开发包图

### 4.2运行时进程

在LSLS中，会有多个客户端进程和一个服务器端进程，其进程图如图5所示。结合部署图，客户端进程是在客户端机器上运行服务器端进程是在服务器端机器上运行。

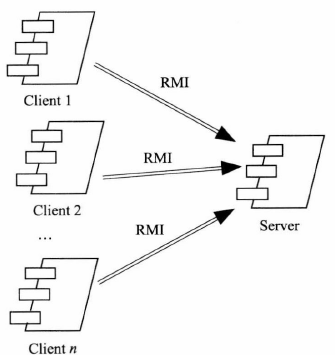


图5 进程图

### 4.3物理部署

LSLS中客户端构件是放在客户端机器上，服务器端构件是放在服务器端机器上。在客户端节点上，还要部署RMIStub构件。由于Java RMI构件属于JDK 6.0的一部分。所以，在系统JDK环境已经设置好的情况下，不需要再独立部署。部署图如图6所示。

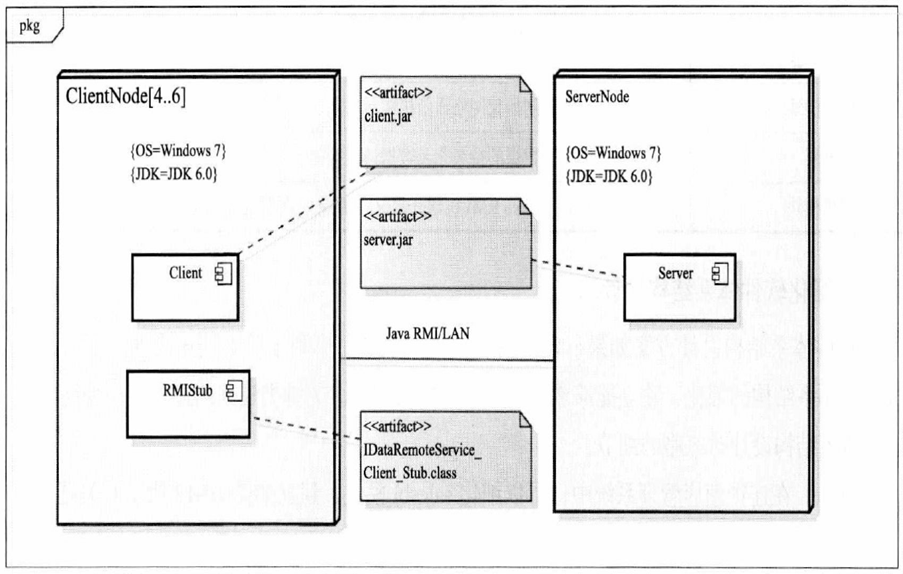


图6 部署图

# 5.接口视角

### 5.1模块的职责

客户端模块和服务器端模块分别如图7和图8所示。客户端各层和服务器端各层的职责分别如表2和表3所示。

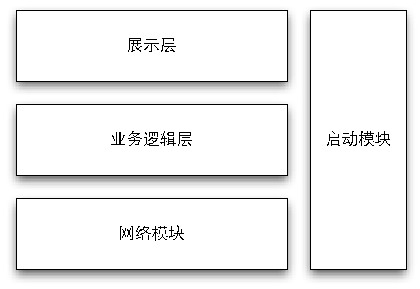


图7 客户端模块视图

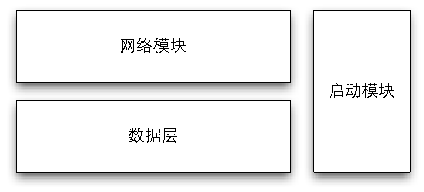


图8 服务器端模块视图

**表2 客户端各层的职责**

|  |  |
| --- | --- |
| **层** | **职责** |
| 启动模块 | 负责初始化网络通信机制，启动用户界面 |
| 用户界面层 | 基于创刊的LSLS用户界面 |
| 业务逻辑层 | 对于用户界面的输入进行响应并进行业务逻辑处理 |
| 客户端网络模块 | 利用Java RMI机制查找RMI服务 |

**表3 服务器端各层的职责**

|  |  |
| --- | --- |
| **层** | **职责** |
| 启动模块 | 负责初始化网络通信机制，启动用户界面 |
| 数据层 | 负责数据的持久化及数据访问接口 |
| 服务器端网络模块 | 利用Java RMI机制开启RMI服务，注册RMI服务 |

每一层只是使用下方直接接触的层。层与层之间仅仅是通过接口的调用来完成的。层直接调用的接口如表4所示。

**表4 层之间调用的接口**

|  |  |  |
| --- | --- | --- |
| **接口** | **服务调用方** | **服务提供方** |
| CommodityBLService  LoginBLService  SalesBLService  PromotionBLService  BookBLService  AccountBLService  DocBLService  UserBLService | 客户端展示层 | 客户端业务逻辑层 |
| CommodityDataService  LoginDataService  SalesDataService  PromotionDataService  BookDataService  AccountDataService  DocDataService  UserDataService | 客户端业务逻辑层 | 服务器端数据层 |

借用Sales用例来说明层之间的调用，如图9所示。每一层之间都是由上层依赖了一个接口（需接口），而下层实现了这个接口（供接口）。SalesBLService提了Sales界面所需要的所有业务逻辑功能。SalesDataService提供了对数据库的增、删、改、查等操作。这样的实现就大大降低了层与层之间的耦合。

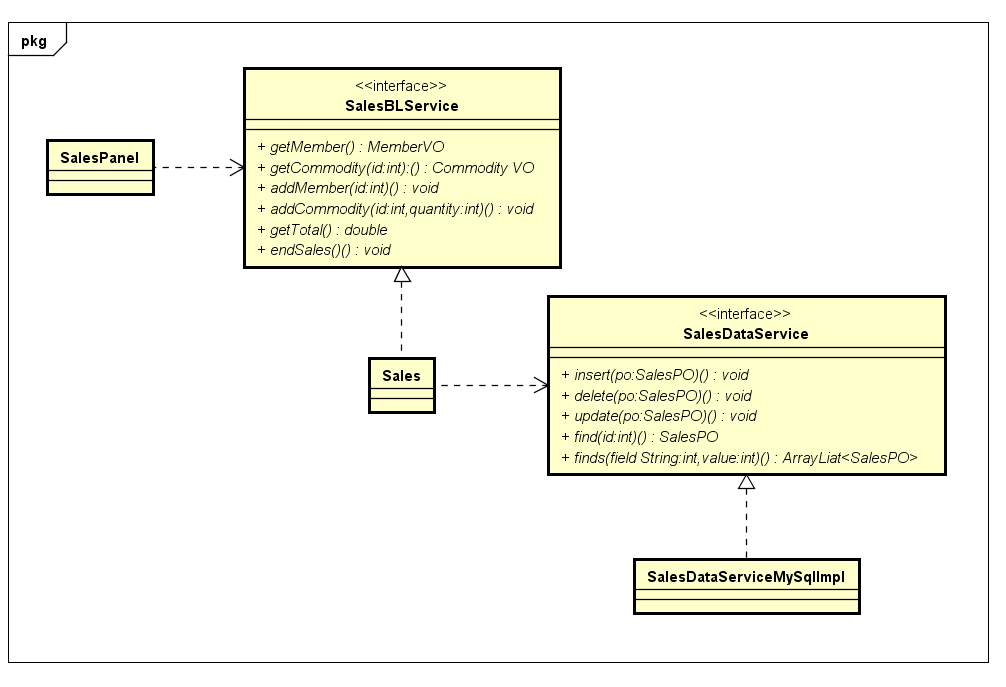


图9 Sales用例层之间调用的接口

### 5.2用户界面层的分解

根据需求，系统存在18个用户界面：登录界面、库存管理人员主界面、进货销售人员主界面、财务人员主界面、总经理主界面、管理员主界面、商品及其分类管理界面、库存管理界面、库存分析界面、客户管理界面、销售界面、退货界面、建账界面、账表查看界面、账户管理界面、审批单据界面、制定销售单据界面、调整用户界面。界面跳转如图10所示。

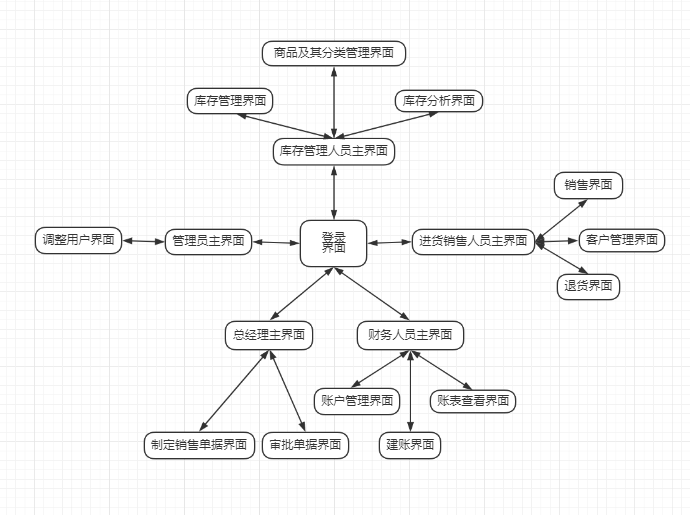


图10 用户界面跳转

服务器端和客户端的用户界面设计接口是一致的，只是具体的页面不一样。用户界面类如图11所示。

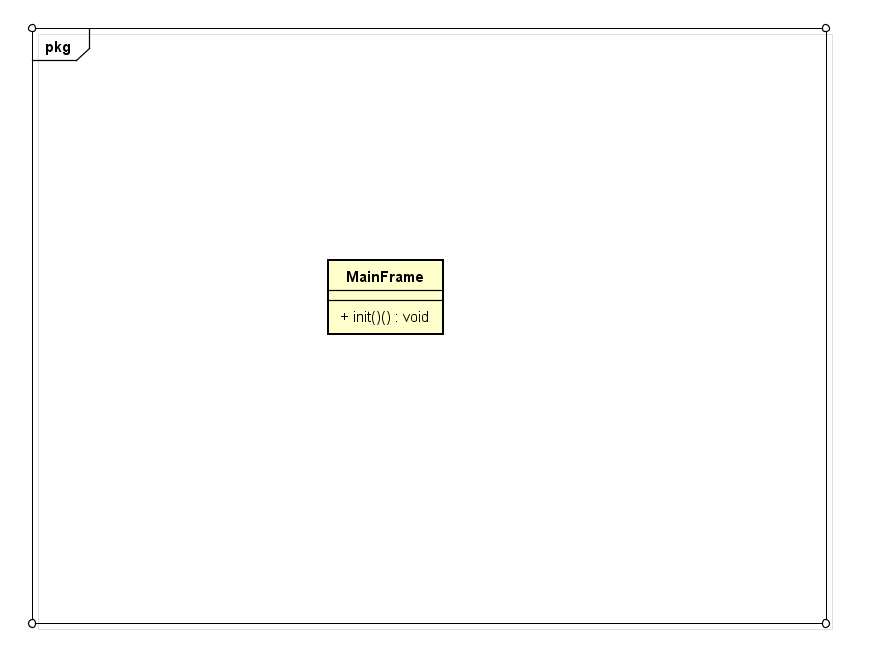


图11 用户界面类

### 5.2.1 用户界面层模块的职责

如表5所示为用户界面层模块的职责。

**表5 用户界面层模块的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| MainFrame | 界面Frame,负责界面的显示和跳转 |

### 5.2.2 用户界面层模块的接口规范

用户界面层模块的接口规范如表6所示。

**表6 用户界面层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| MainFrame | 语法 | Int(args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示Frame以及LoginPanel |

用户界面层需要的服务接口如表7所示。

**表7 用户界面层模块需要的服务接口**

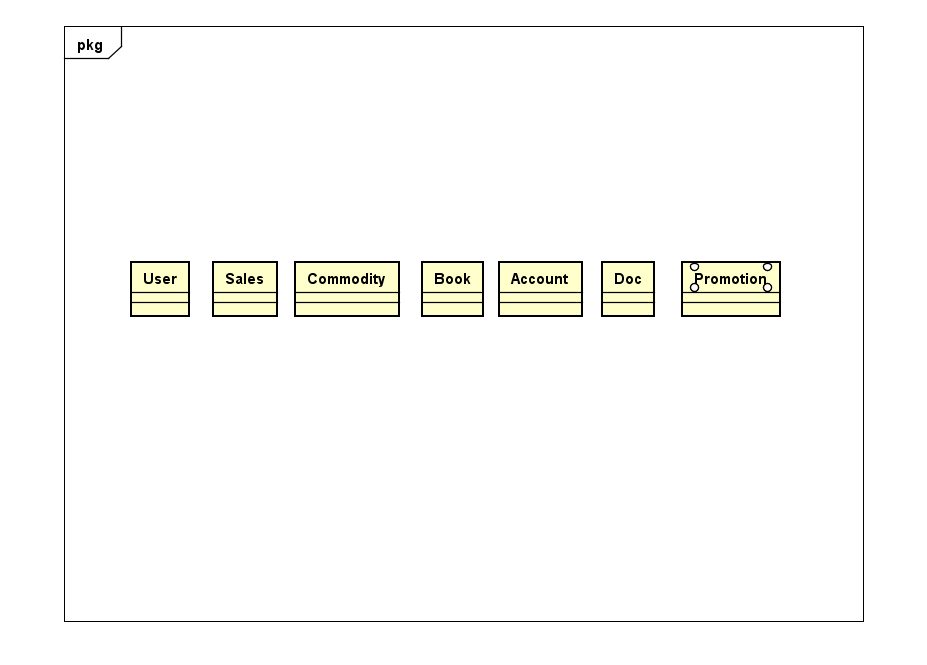
|  |  |
| --- | --- |
| **服务名** | **服务** |
| businesslogicservice.LoginBLService | 登录界面的逻辑接口 |
| businesslogicservice.\*BLService | 每个界面都有一个相应的业务逻辑接口 |

### 5.2.3 用户界面模块设计原理

用户界面利用Java的Swing和AWT库来实习。

### 5.3业务逻辑层的分解

业务逻辑层包括多个针对界面的业务逻辑处理对象。例如，User对象负责处理登录界面的业务逻辑；Sales对象负责销售界面的业务逻辑。业务逻辑层的设计如图12所示。

**图12 业务逻辑层的设计**

### 5.3.1 业务逻辑层模块的职责

业务逻辑层模块的职责如表8所示。

**表8 业务逻辑层模块的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| userbl | 负责实现对应与登陆界面所需要的服务 |
| salesbl | 负责实现销售界面所需要的服务 |
| commoditybl | 负责实现库存界面所需要的服务 |
| bookbl | 负责实现账簿界面所需要的服务 |
| accountbl | 负责实现账户界面所需要的服务 |
| docbl | 负责实现单据界面所需要的服务 |
| promotionbl | 负责实现销售策略界面所需要的服务 |

### 5.3.2 业务逻辑层模块的接口规范

userbl和accountbl模块的接口规范分别如表9和表10所示。

**表9 userbl模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| User.login | 语法 | | public boolean login(String id,String password); |
| 前置条件 | | password符合输入规则 |
| 后置条件 | | 查找是否存在相应的User，根据输入的password返回登陆验证的结果 |
| User.new | 语法 | | public static boolean userNew(); |
| 前置条件 | | 管理员成功登陆 |
| 后置条件 | | 新建成功则在数据库中添加新的User记录，否则返回相应出错信息 |
| User.edit | 语法 | | public static boolean userEdit(User opUser)； |
| 前置条件 | | 管理员已经选定某一User |
| 后置条件 | | 若修改操作正常进行则根据管理员的修改对相应User信息进行修改，否则返回出错信息 |
| User.delete | 语法 | | public static boolean userDel(User delUser); |
| 前置条件 | | 管理员已经选定某一User |
| 后置条件 | | 在数据库中将对应用户记录删除 |
| 需要的服务 | | | |
| 服务名 | | 服务 | |
| DatabaseFactory.getUserDatabase | | 得到User数据库的服务的引用 | |
| UserDataService.check(UserPO po) | | 在数据库中查找UserPO对象 | |
| UserDataService.new(UserPO po) | | 在数据库中新建UserPO对象 | |
| UserDataService.edit(UserPO po) | | 在数据库中修改引用的UserPO对象 | |

**表10 accountbl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Zhanghuguanli.NameChange | 语法 | public void NameChange(String a) ; |
| 前置条件 | 财务人员进行更改账户名称操作 |
| 后置条件 | 根据账户名称确定是否存在账户，并根据反馈进行更改操作 |
| Zhanghuguanli.NameDelete | 语法 | public void NameDelete(String a) ; |
| 前置条件 | 财务人员进行删除账户操作 |
| 后置条件 | 根据账户名称确定是否存在账户，若存在则删除该用户相关内容 |
| Zhanghuguanli. NameAdd | 语法 | public void NameAdd(String a) ; |
| 前置条件 | 财务人员进行增加账户操作 |
| 后置条件 | 根据账户名称确定是否存在账户，若不存在则增加用户 |
| Zhanghuguanli.NameCheck | 语法 | public void NameCheck(String a) ; |
| 前置条件 | 财务人员进行查看账户操作 |
| 后置条件 | 根据账户名称确定账户，并显示账户名称，余额等内容 |
| Zhanghuguanli. JinEchange | 语法 | public void JinEchange(float a) ; |
| 前置条件 | 财务人员进行更改金额操作 |
| 后置条件 | 根据账户名称确定金额数目，并根据更改数据进行更改金额 |
| Zhanghuguanli. YuEchange | 语法 | public void YuEchange(float a) ; |
| 前置条件 | 付款单和收款单导入分析成功 |
| 后置条件 | 根据账户名称确定账户，系统计算并显示该账户余额 |
| Zhidingshoukuandan. GetS | 语法 | Public void GetS(String Num,String keHu,String Oprater,String zhuanZhangLieBiao,float jinE); |
| 前置条件 | 财务人员必须已经被识别和授权,并进行制定操作 |
| 后置条件 | 收款单通过审批后更改客户的应付数据，同时财务人员会收到一条消息，根据该信息财务人员完成现实银行账户操作。 |
| ZhiDingFuKuanDan. GetF | 语法 | public void GetF(String Num,String zhangHu,String Oprater,String tiaoMuQingDan,float jinE); |
| 前置条件 | 财务人员必须已经被识别和授权,并进行制定操作 |
| 后置条件 | 付款单通过审批后更改客户的应收数据，同时财务人员会收到一条消息，根据该信息财务人员完成现实银行账户操作。 |

以下各表分别为salesbl、commoditybl、bookbl、docbl以及promotionbl模块的接口规范。

**表 salesbl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| saleRun.play | 语法 | public boolean Play(); |
| 前置条件 | 进货销售人员对销售单商品列表进行了操作 |
| 后置条件 | 商品列表所添加商品仓库存在 |
| saleRun.confirm | 语法 | public boolean Confirm()； |
| 前置条件 | 进货销售人员完成了销售操作并准备提交 |
| 后置条件 | 销售单核实无误 |
| stockRun.play | 语法 | public boolean Play(); |
| 前置条件 | 进货销售人员对进货单商品列表进行了操作 |
| 后置条件 | 商品列表所添加商品类别合法 |
| stockRun.confirm | 语法 | public boolean Confirm()； |
| 前置条件 | 进货销售人员完成了进货操作并准备提交 |
| 后置条件 | 进货单核实无误 |
| saleReturnRun.play | 语法 | public boolean Play(); |
| 前置条件 | 进货销售人员对销售退货单商品列表进行了操作 |
| 后置条件 | 商品列表符合初始销售单 |
| saleReturnRun.confirm | 语法 | public boolean Confirm()； |
| 前置条件 | 进货销售人员完成了销售退货操作并准备提交 |
| 后置条件 | 销售退货单核实无误 |
| saleReturnRun.findSaleForm | 语法 | public boolean findSaleForm(String ID) |
| 前置条件 | 销售过程发生过，销售单记录存在 |
| 后置条件 | ID号正确 |
| saleReturnRun.findReturnForm | 语法 | public boolean findReturnForm(String ID)； |
| 前置条件 | 销售过程发生过，退货单记录存在 |
| 后置条件 | 销售退货单ID正确 |
| stockReturnRun.play | 语法 | public boolean Play(); |
| 前置条件 | 进货销售人员对退货单商品列表进行了操作 |
| 后置条件 | 商品列表所添加退货商品仓库存在且充足 |
| stockReturnRun.confirm | 语法 | public boolean Confirm()； |
| 前置条件 | 进货销售人员完成了退货记录并准备提交 |
| 后置条件 | 退货单核实无误 |
| 需要的服务 | | |
| 服务名 | 服务 | |
| GetGoods(String ID); | 输入商品ID号获取商品的包括单价、类别、型号的详细信息 | |
| GetStock(String ID); | 输入商品ID号获取商品的在仓库的库存数量以及仓库库号 | |
| ChangeStock(String ID;Dword number); | 更改仓库库存 | |

**表** commoditybl**模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| goodsClassification. Attribute. addClassification | 语法 | public void addClassification(String num, String str) |
| 前置条件 | 库存管理人员进行增加分类的操作 |
| 后置条件 | 根据分类名称确定是否已存在该类，若不存在则根据输入进行添加操作 |
| goodsClassification.Attribute. modClassification | 语法 | public void modClassification(int n, String num, String str) |
| 前置条件 | 库存管理人员进行修改分类操作 |
| 后置条件 | 根据类别名称或编号确定是否存在该类，并根据需求进行更改操作 |
| goodsClassification.Attribute. subClassification | 语法 | public void subClassification(int n) |
| 前置条件 | 库存管理人员进行删除分类操作 |
| 后置条件 | 根据类别名称或编号确定是否存在该类，若存在则进行删除分类操作 |
| goodsClassification.SubClass | 语法 | public boolean Judgement(String name) |
| 前置条件 | 库存管理人员已被识别或授权 |
| 后置条件 | 根据商品分类的主类判断子类的存在是否合理 |
| goodsClassification.SuperClass | 语法 | public void SuperClass |
| 前置条件 | 库存管理人员已被识别或授权 |
| 后置条件 | 将商品分类的主类数据进行存储 |
| goodsManagement.Main.Searching | 语法 | public void Searching(String name, String category, String num) |
| 前置条件 | 库存管理人员进行查看操作 |
| 后置条件 | 根据商品的名称、编号或类别确定是否存在该商品，若存在则查看商品信息 |
| goodsManagement.Main.addGoods | 语法 | public void addGoods(String name, String type, int stock, double bprice, double sprice) |
| 前置条件 | 库存管理人员进行增加商品的操作 |
| 后置条件 | 根据商品名称或编号确定是否已存在该商品，若不存在则输入商品信息进行添加操作 |
| goodsManagement.Main.subGoods | 语法 | public void subGoods(String name, String num) |
| 前置条件 | 库存管理人员进行删除商品的操作 |
| 后置条件 | 根据商品名称或编号确定是否已存在该商品，若存在则进行删除操作 |
| goodsManagement.Main.modGoods | 语法 | public void modGoods(String name, String num, int stock, double bprice, double sprice) |
| 前置条件 | 库存管理人员进行修改商品的操作 |
| 后置条件 | 根据商品名称或编号确定是否已存在该商品，若存在则进行修改操作 |
| goodsManagement.Main. ErrorJudgement | 语法 | public boolean ErrorJudgement() |
| 前置条件 | 库存管理人员已被识别或授权 |
| 后置条件 | 进行输入判断，包含非法输入判断、重复查询判断、中断查询 |
| inventoryAnalysis.Analysis. Searching | 语法 | public boolean Searching |
| 前置条件 | 库存管理人员已被识别或授权 |
| 后置条件 | 库存管理人员进行查看库存状况的操作 |
| inventoryAnalysis.Analysis.Date | 语法 | public void Date |
| 前置条件 | 库存管理人员已被识别或授权 |
| 后置条件 | 系统根据管理员输入的时间段显示该时间段内的库存变化情况 |
| inventoryAnalysis.Documents. GivingDocument | 语法 | public void GivingDocument |
| 前置条件 | 库存管理人员根据库存分析提供库存赠送单 |
| 后置条件 | 系统生成库存赠送单并显示 |
| inventoryAnalysis.Documents. RemainingDocument | 语法 | public void RemainingDocument |
| 前置条件 | 库存管理人员根据库存分析提供库存报溢单 |
| 后置条件 | 系统生成库存报溢单并显示 |
| inventoryAnalysis.Documents. LackingDocument | 语法 | public void LackingDocument |
| 前置条件 | 库存管理人员根据库存分析提供库存报损单 |
| 后置条件 | 系统生成库存报损单并显示 |
| inventoryAnalysis.Documents. WarningDocument | 语法 | public void WarningDocument |
| 前置条件 | 库存管理人员根据库存分析提供库存报警单 |
| 后置条件 | 系统生成库存报警单并显示 |
| inventoryManagement.Viewing. Searching | 语法 | public void Searching |
| 前置条件 | 库存管理人员已被识别或授权 |
| 后置条件 | 库存管理人员进行查看库存状况的操作 |
| inventoryManagement.Viewing. Date | 语法 | public void Date |
| 前置条件 | 库存管理人员已被识别或授权 |
| 后置条件 | 系统根据管理员输入的时间段显示该时间段内的库存变化情况 |
| inventoryManagement.Checking | 语法 | public void checking |
| 前置条件 | 库存管理人员查看当日的库存快照 |
| 后置条件 | 系统显示当日的库存快照 |

**表 bookbl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| chaKanXiaoShouMingXiBiao. Timelog | 语法 | public void Timelog(String Time); |
| 前置条件 | 财务人员按时间区间进行筛选查询 |
| 后置条件 | 系统显示有关时间区间的类 |
| chaKanXiaoShouMingXiBiao.goodsName | 语法 | public void goodsName(String Goods); |
| 前置条件 | 财务人员按商品名进行筛选查询 |
| 后置条件 | 系统显示有关商品名的类 |
| chaKanXiaoShouMingXiBiao. Client | 语法 | public void Client(String Name); |
| 前置条件 | 财务人员按客户进行筛选查询 |
| 后置条件 | 系统显示有关客户的类 |
| chaKanXiaoShouMingXiBiao.Operater | 语法 | public void Operater(String opName); |
| 前置条件 | 财务人员按业务员进行筛选查询 |
| 后置条件 | 系统显示有关业务员的类 |
| chaKanXiaoShouMingXiBiao. cangKu | 语法 | public void cangKu(String place); |
| 前置条件 | 财务人员按仓库进行筛选查询 |
| 后置条件 | 系统显示有关仓库的类 |
| chaKanXiaoShouMingXiBiao. out | 语法 | public void out(); |
| 前置条件 | 财务人员按时间区间，商品名，客户，业务员，仓库五种筛选条件进行查询 |
| 后置条件 | 系统显示符合上述条件的商品销售记录，以列表形式显示时间，商品名，型号，数量，单价，总额 |
| chaKanJingYingLiChengBiao. Xiao  ShouChuHuoDan | 语法 | public void xiaoShouChuHuoDan(); |
| 前置条件 | 财务人员选择销售类单据并选择销售出货单 |
| 后置条件 | 系统显示销售出货单的单据内容 |
| chaKanJingYingLiChengBiao. Xiao  ShouTuiHuoDan | 语法 | public void xiaoShouTuiHuoDan(); |
| 前置条件 | 财务人员选择销售类单据并选择销售退货单 |
| 后置条件 | 系统显示销售退货单的单据内容 |
| chaKanJingYingLiChengBiao. jinHuo  Dan | 语法 | public void jinHuoDan(); |
| 前置条件 | 财务人员选择销售类单据并选择进货单 |
| 后置条件 | 系统显示进货单的单据内容 |
| chaKanJingYingLiChengBiao. jinHuoT  uiHuoDan | 语法 | public void jinHuoTuiHuoDan(); |
| 前置条件 | 财务人员选择销售类单据并选择进货退货单 |
| 后置条件 | 系统显示进货退货单的单据内容 |
| chaKanJingYingLiChengBiao. fuKuan  Dan | 语法 | public void fuKuanDan(); |
| 前置条件 | 财务人员选择销售类单据并选择付款单 |
| 后置条件 | 系统显示付款单的单据内容 |
| chaKanJingYingLiChengBiao. ShouKu  anDan | 语法 | public void shouKuanDan(); |
| 前置条件 | 财务人员选择销售类单据并选择收款单 |
| 后置条件 | 系统显示收款单的单据内容 |
| chaKanJingYingLiChengBiao. xianJin  FeiYongDan | 语法 | public void xianJinFeiYongDan(); |
| 前置条件 | 财务人员选择销售类单据并选择现金费用单 |
| 后置条件 | 系统显示现金费用单的单据内容 |
| chaKanJingYingLiChengBiao. baoYi  Dan | 语法 | public void baoYiDan(); |
| 前置条件 | 财务人员选择销售类单据并选择报溢单 |
| 后置条件 | 系统显示报溢单的单据内容 |
| chaKanJingYingLiChengBiao. baoShun  Dan | 语法 | public void baoShunDan(); |
| 前置条件 | 财务人员选择销售类单据并选择报损单 |
| 后置条件 | 系统显示报损单的单据内容 |
| chaKanJingYingLiChengBiao. zengSon  gDan | 语法 | public void zengSongDan(); |
| 前置条件 | 财务人员选择销售类单据并选择赠送单 |
| 后置条件 | 系统显示销售赠送单的单据内容 |

**表** docbl**模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| DocCheck.display | 语法 | | public void display(ArrayList<DocCheck> docs); |
| 前置条件 | | 总经理成功登陆系统 |
| 后置条件 | | 列表显示待审批的单据 |
| DocCheck. view | 语法 | | public void view(DocCheck doc); |
| 前置条件 | | 总经理制定某一待审批单据 |
| 后置条件 | | 显示指定单据的详细信息 |
| DocCheck. edit | 语法 | | public boolean edit(DocCheck doc); |
| 前置条件 | | 界面显示着某一单据的详细信息 |
| 后置条件 | | 若修改操作正常进行则根据总经理的修改对相应单据信息进行修改，否则返回false |
| 需要的服务 | | | |
| 服务名 | | 服务 | |
| DatabaseFactory.getDocDatabase | | 得到Doc数据库的服务的引用 | |
| UserDataService.check(DocPO po) | | 在数据库中查找DocPO对象 | |
| UserDataService.edit(DocPO po) | | 在数据库中修改引用的DocPO对象 | |

**表** promotionbl**模块的接口规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| PromotionDev.SPP | 语法 | | public static boolean SPP\_Dev(ArrayList<Integer> goodsList,ArrayList<Double> SPList) |
| 前置条件 | | 总经理已被授权（成功登陆即授权） |
| 后置条件 | | 在与已有促销策略不冲突时，将新的商品特价包添加到PromotionDataBase，否则提示出错信息 |
| PromotionDev.SP | 语法 | | public static boolean SP\_Dev(double sum,ArrayList<Integer> giftsList,double voucher,double discount) |
| 前置条件 | | 总经理已被授权（成功登陆即授权） |
| 后置条件 | | 与已有促销策略不冲突时，将新的总价促销策略添加到PromotionDataBase，否则提示出错信息 |
| PromotionDev.CP | 语法 | | public static boolean CP\_Dev(int userLevel,ArrayList<Integer> giftsList,double voucher,double discount) |
| 前置条件 | | 总经理已被授权（成功登陆即授权） |
| 后置条件 | | 与已有促销策略不冲突时，将新的客户促销策略添加到PromotionDataBase，否则提示出错信息 |
| 需要的服务 | | | |
| 服务名 | | 服务 | |
| DatabaseFactory.getPromotionDatabase | | 得到Promotion数据库的服务的引用 | |
| PromotionDataService.check(PromotionPO po) | | 在数据库中查找PromotionPO对象 | |
| PromotionDataService.new(PromotionPO po) | | 在数据库中添加PromotionPO对象 | |

### 5.4数据层的分解

数据层主要给业务逻辑层提供数据访问服务包括对于持久化的数据的增、删、改、查。Sales业务逻辑需要的服务由SalesDataService接口提供。由于持久化数据的保存可能存在多种形式：Txt文件、序列化文件、数据库等，所示抽象了数据服务。数据层模块的描述具体如图13所示。

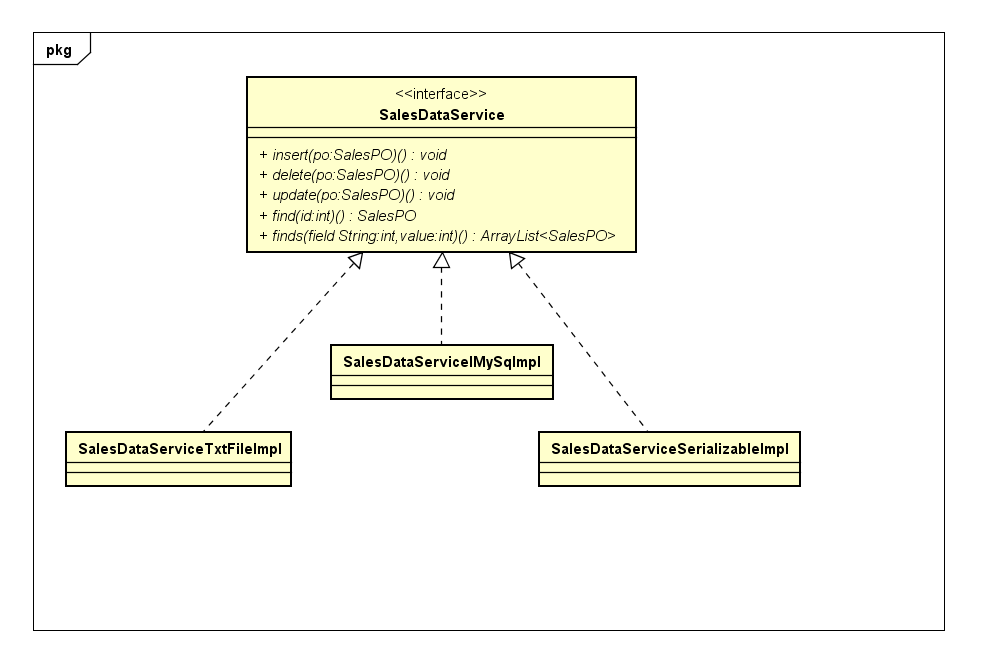


图13 数据层模块的描述

### 5.4.1 数据层模块的职责

数据层模块的职责如表11所示。

**表11 数据层模块的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| SalesDataService | 持久化数据库的接口，提供集体载入、集体保存、增删改查服务 |
| SalesDataServiceTxtFileImpl | 基于Txt文件的持久化数据库的接口，提供集体载入、集体保存、增删改查服务 |
| SalesDataServiceSerializableFileImpl | 基于序列化文件的持久化数据库的接口，提供集体载入、集体保存、增删改查服务 |
| SalesDataServiceMySqlImpl | 基于MySql数据库的持久化数据库的接口，提供集体载入、集体保存、增删改查服务 |

### 5.4.2 数据层模块的接口规范

数据层的接口规范如表12所示。

**表12 数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| SalesDataService.find | 语法 | public SalesPO find(long id) throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 按ID进行查找返回相应的SalesPO结果 |
| SalesDataService.insert | 语法 | public void insert(SalesPO po) throws RemoteException; |
| 前置条件 | 同样ID的po在Mapper中不存在 |
| 后置条件 | 在数据库中增加一个po记录 |
| SalesDataService.delete | 语法 | public void delete(SalesPO po) throws RemoteException; |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 删除一个po |
| SalesDataService.updata | 语法 | public void update(SalesPO po) throws RemoteException; |
| 前置条件 | 在数据库中存在同样ID的po |
| 后置条件 | 更新一个po |
| SalesDataService.init | 语法 | public void init() throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 初始化持久化数据库 |
| SalesDataService.finish | 语法 | public void finish() throws RemoteException; |
| 前置条件 | 无 |
| 后置条件 | 结束持久化数据库的使用 |

**表 Data层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| DataBaseFactory. getUserDataservice | 语法 | UserDataService getUserDatabase(); |
| 前置条件 | 持久化数据库成功链接 |
| 后置条件 | 利用接口进行数据的获取和更新 |
| DataBaseFactory. getCommodityService | 语法 | CommodityService getCommodityService(); |
| 前置条件 | 持久化数据库成功链接 |
| 后置条件 | 利用接口进行数据的获取和更新 |
| DataBaseFactory. getClientService | 语法 | ClientService getClientService(); |
| 前置条件 | 持久化数据库成功链接 |
| 后置条件 | 利用接口进行数据的获取和更新 |
| DataBaseFactory. getPromotionService | 语法 | PromotionService getPromotionService(); |
| 前置条件 | 持久化数据库成功链接 |
| 后置条件 | 利用接口进行数据的获取和更新 |
| DataBaseFactory. getAccountService | 语法 | AccountService getAccountService(); |
| 前置条件 | 持久化数据库成功链接 |
| 后置条件 | 利用接口进行数据的获取和更新 |

# 6.接口视角

### 6.1数据持久化对象

系统的PO类就是对应的相关的实体类，在此只做简单的介绍。

UserPO类包含用户的用户名、密码属性

CommodityPO类包含商品的编号、价格、数量和名字属性。

MemberPO类包含客户的类（进货商、销售商）、级别（五级，一级普通用户，五级VIP客户）、姓名、电话、地址、邮编、电子邮箱、应收额度、应收、应付、默认业务员

SalesPO类保存销售时数据，包括编号、客户编号、商品列表、仓库、总额合计

PromotionPO类保存商品特价包、总额或客户级别促销策略

DocPO类包括的单据的类型及审批状态

BookPO类记录账簿的相关信息，包括销售明细、经营历程、经营情况

AccountPO类保存账户数据，包括账户账号以及密码属性

持久化用户对象UserPO的定义如图14所示。

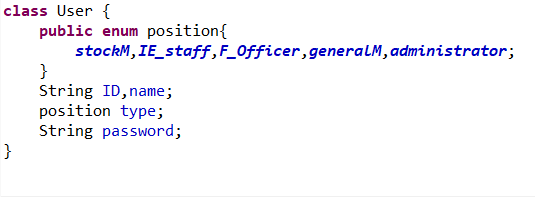


图14 持久化用户对象UserPO的定义

### 6.2 Txt持久化格式

Txt数据保持格式以Commodity.txt为例。每行分别对应货号、商品名称、价格、数量。中间用“：”隔开。如下所示：

123：LED灯：10：32

### 6.3 数据库表

数据库中包含User表、Commodity表、Member表、Sales表、Promotion表、Doc表、Book表、Account表。