袁楚宏 袁阳阳 张鑫龙 周小帆

China Nanjing University

快递物流系统 软件详细设计描述文档

目录

[1. 引言 1](#_Toc434340954)

[1.1 编制目的 1](#_Toc434340955)

[1.2 词汇表 2](#_Toc434340956)

[1.3 参考资料 2](#_Toc434340957)

[3. 体系结构设计概述 2](#_Toc434340958)

[4. 结构视角 2](#_Toc434340959)

[4.1 业务逻辑层分解 2](#_Toc434340960)

[4.1.1 userLogic模块 2](#_Toc434340961)

[（1） 模块概述 2](#_Toc434340962)

[（2） 整体结构 2](#_Toc434340963)

[4.1.2 transitInfoLogic模块 2](#_Toc434340964)

[（1） 模块概述 2](#_Toc434340965)

[（2） 整体结构 2](#_Toc434340966)

[（3） 模块内部类的接口规范 3](#_Toc434340967)

[（4） 业务逻辑层的动态模型 4](#_Toc434340968)

[（5） 业务逻辑层的设计原理 4](#_Toc434340969)

[4.1.3 workOrgManLogic模块 4](#_Toc434340970)

[（1） 模块概述 4](#_Toc434340971)

[（2） 整体结构 4](#_Toc434340972)

[（3） 模块内部类的接口规范 5](#_Toc434340973)

[（4） 业务逻辑层的动态模型 9](#_Toc434340974)

[（5） 业务逻辑层的设计原理 9](#_Toc434340975)

[4.1.4 accountLogic模块 9](#_Toc434340976)

[（1） 模块概述 9](#_Toc434340977)

[（2） 整体结构 9](#_Toc434340978)

[（3） 模块内部类的接口规范 10](#_Toc434340979)

[（4） 业务逻辑层的动态模型 11](#_Toc434340980)

[（5） 业务逻辑层的设计原理 11](#_Toc434340981)

[4.1.5 stockLogic模块 11](#_Toc434340982)

[（1） 模块概述 11](#_Toc434340983)

[（2） 整体结构 11](#_Toc434340984)

[（3） 模块内部类的接口规范 12](#_Toc434340985)

[（4） 业务逻辑层的动态模型 14](#_Toc434340986)

[（5） 业务逻辑层的设计原理 14](#_Toc434340987)

[5. 依赖视角 14](#_Toc434340988)

1. 引言
   1. 编制目的

本报告详细完成对连锁商店管理系统的详细设计，达到指导后续软件构造的目的，同时实现和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而编写，是了解系统的导航。

* 1. 词汇表

|  |  |  |
| --- | --- | --- |
| 词汇名称 | 词汇含义 | 备注 |
| ELS | 快递物流系统 |  |
| RMI | 远程接口调用 | 一般使用JavaRMI包 |

* 1. 参考资料

1. 快递物流系统软件体系结构描述文档
2. 快递物流系统用例文档
3. 快递物流系统需求规格文档
4. 产品概述

参考快递物流系统用例文档和快递物流系统需求规格说明文档中对产品的概括描述。

1. 体系结构设计概述

参考快递物流系统软件体系结构描述文档中体系结构设计的描述。

1. 结构视角
   1. 业务逻辑层分解

业务逻辑层的开发包图参见软件体系结构文档图3。

* + 1. userLogic模块

1. 模块概述

userLogic模块承担的需求参见需求规格说明文档

1. 整体结构
   * 1. transitInfoLogic模块
2. 模块概述

transitInfoLogic模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

transitInfoLogic模块的职责及接口参见软件体系结构描述文档5.3.2中transitInfo模块的接口规范

1. 整体结构

根据体系结构的设计，分层结构分为界面、逻辑和数据层。层与层之间添加Service接口。界面层和数据层之间添加CourierLSer作为快递员查询订单信息的接口，CustomerLogicSer作为查询物流信息的接口。逻辑层和数据层之间添加CourierDSer作为快递员查询订单信息的接口，CustomerDSer作为查询物流信息的接口。ExpressPO存放持久化的订单数据，ResultMessage存放操作是否成功和订单的值对象ExpressVO。

transitInfoLogic模块的设计如图所示。

transitInfoLogic模块各个类的职责如表所示。

transitInfoLogic模块各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| CourierLogic | 负责实现订单信息查询界面所需要的数据处理 |
| CustomerLogic | 负责实现物流信息查询界面所需要的数据处理 |

1. 模块内部类的接口规范

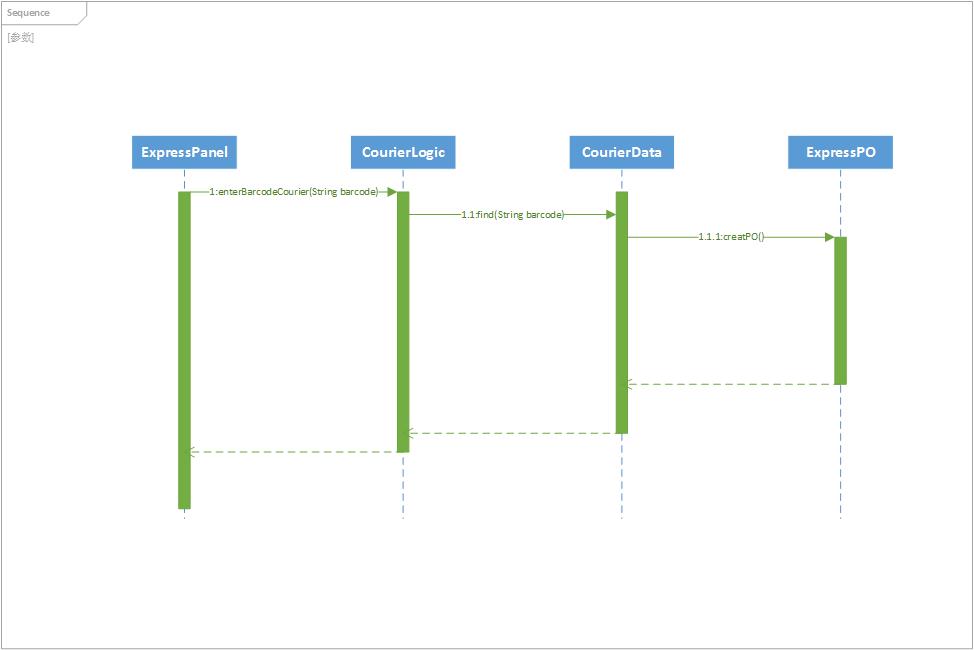
CourierLogic的接口规范如表所示。

CourierLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CourierLogic.enterBarcodeCourier | 语法 | public ResultMessage enterBarcodeCourier(String barcode) |
| 前置条件 | 已输入并确认订单条形码号 |
| 后置条件 | 根据条形码号从数据层获取订单信息 |
| 需要的服务（需接口） | | |
| CourierDSer.find(String barcode) | 根据barcode查找订单信息，返回查找结果 | |
|  |  | |

CustomerLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CustomerLogic.enterBarcodeCustomer | 语法 | public ResultMessage enterBarcodeCourier(String barcode) |
| 前置条件 | 已输入并确认订单条形码号 |
| 后置条件 | 根据条形码号从数据层获取订单的物流信息 |
| 需要的服务（需接口） | | |
| Customer.findTransit(String barcode) | 根据barcode查找订单物流信息，返回查找结果 | |

1. 业务逻辑层的动态模型

上图表明了快递物流系统中，当快递员输入快递单号之后，快递员订单信息查询逻辑的相关对象之间的协作。

1. 业务逻辑层的设计原理

利用委托式控制风格，每个界面需要访问的业务逻辑由相应领域的对象负责。

* + 1. workOrgManLogic模块

1. 模块概述

workOrgManLogic模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

workOrgManLogic模块的职责及接口参见软件体系结构描述文档5.3.2中workOrgManLogic模块的接口规范

1. 整体结构

根据体系结构的设计，分层结构分为界面、逻辑和数据层。界面层和逻辑层之间添加DriverManLSer作为司机管理界面的逻辑层接口，OrgManLSer作为机构管理界面的逻辑层接口，VanManLSer作为车辆管理界面的逻辑层接口，WageManLSer作为薪水管理界面的逻辑层接口，WorkManLSer作为人员管理界面的逻辑层接口。逻辑层和数据层之间添加DriverManDSer作为司机管理逻辑的数据层接口，OrgManDSer作为机构管理逻辑的数据层接口，VanManDSer作为车辆管理逻辑的数据层接口，WageManDSer作为薪水管理逻辑的数据层接口，WorkManDSer作为人员管理逻辑的数据层接口。DriverVO,OrgVO,VanVO,WageVO,WorkVO分别作为司机、机构、车辆、薪水和人员的界面值对象。相应的PO分别作为相应持久化对象。UtilityLogic作为逻辑层共有的初始化操作的集合，依赖于数据层的UtilityDSer接口。CityPO含有城市信息。

workOrgManLogic模块的设计如图所示。

workOrgManLogic模块各个类的职责如表所示

workOrgManLogic模块各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| DriverManLogic | 负责司机管理界面所需要的业务逻辑 |
| OrgManLogic | 负责机构管理界面所需要的业务逻辑 |
| VanManLogic | 负责车辆管理界面所需要的业务逻辑 |
| WageManLogic | 负责薪水管理界面所需要的业务逻辑 |
| WorkManLogic | 负责人员管理界面所需要的业务逻辑 |

1. 模块内部类的接口规范

DriverManLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| DriverManLogic.addDriver(DriverVO Driver) | 语法 | public boolean addDriver(DriverVO Driver) |
| 前置条件 | 已输入完整的司机信息，确认新增 |
| 后置条件 | 将司机信息发送到数据层并添加到数据库 |
| DriverManLogic.deleteDriver(String id) | 语法 | public boolean deleteDriver(String id) |
| 前置条件 | 选择司机，确认删除 |
| 后置条件 | 将ID发送到数据层并从库中删除相应司机的信息 |
| DriverManLogic.reviseDriver(DriverVO Driver) | 语法 | public boolean reviseDriver(DriverVO Driver) |
| 前置条件 | 确认修改司机信息 |
| 后置条件 | 将司机信息发送到数据层，并修改库中相应信息 |
| DriverManLogic.checkDriver(String id) | 语法 | public ResultMessage checkDriver(String id) |
| 前置条件 | 从司机列表中选择司机选择查看 |
| 后置条件 | 从数据层获得相应司机的信息返回 |
| DriverManLogic.searchDriver(String keyword) | 语法 | public ResultMessage searchDriver(String keyword) |
| 前置条件 | 打开司机管理界面或输入关键词检索 |
| 后置条件 | 从数据层获取司机列表或含有相应关键词的司机列表返回 |
| 需要的服务（需接口） | | |
| DriverManDSer.addDriver(DriverPO driver) | 将PO中的司机信息添加进数据库 | |
| DriverManDSer.deleteDriver(String id) | 将数据库中相应ID的司机信息删除 | |
| DriverManDSer.reviseDriver(DriverPO driver) | 在数据库中修改PO中信息相应的司机信息 | |
| DriverManDSer.checkDriver(String id) | 在数据库中获得相应ID司机的信息并返回 | |
| DriverManDSer.findDriver(String keyword) | 在数据库中获得含有相应关键词（若关键词为空则返回全部）的司机信息列表返回。 | |

OrgManLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrgManLogic.addOrg(OrgVO org) | 语法 | public boolean addOrg(OrgVO org) |
| 前置条件 | 已输入完整的机构信息，确认新增 |
| 后置条件 | 将机构信息发送到数据层并添加到数据库 |
| OrgManLogic.deleteOrg(String id) | 语法 | public boolean deleteOrg(String id) |
| 前置条件 | 选择机构，确认删除 |
| 后置条件 | 将ID发送到数据层并从库中删除相应机构的信息 |
| OrgManLogic.reviseOrg(OrgVO org) | 语法 | public boolean reviseOrg(OrgVO org) |
| 前置条件 | 确认修改机构信息 |
| 后置条件 | 将机构信息发送到数据层，并修改库中相应信息 |
| OrgManLogic.searchOrg(String keyword) | 语法 | public ResultMessage searchOrg(String keyword) |
| 前置条件 | 打开机构管理界面或输入关键词检索 |
| 后置条件 | 从数据层获取机构列表或含有相应关键词的机构列表返回 |
| OrgManLogic.getCities() | 语法 | public ResultMessage getCities() |
| 前置条件 | 打开机构管理界面 |
| 后置条件 | 获得城市信息列表返回 |
| 需要的服务（需接口） | | |
| OrgManDSer.addOrg(OrgPO Org) | 将PO中的机构信息添加进数据库 | |
| OrgManDSer.deleteOrg(String id) | 将数据库中相应ID的机构信息删除 | |
| OrgManDSer.reviseOrg(OrgPO Org) | 在数据库中修改PO中信息相应的机构信息 | |
| OrgManDSer.findOrg(String keyword) | 在数据库中获得含有相应关键词（若关键词为空则返回全部）的机构信息列表返回。 | |
| UtilityDSer.getCities() | 在数据库中获得城市信息列表并返回 | |

VanManLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| VanManLogic.addVan(VanVO Van) | 语法 | public boolean addVan(VanVO Van) |
| 前置条件 | 已输入完整的车辆信息，确认新增 |
| 后置条件 | 将车辆信息发送到数据层并添加到数据库 |
| VanManLogic.deleteVan(String id) | 语法 | public boolean deleteVan(String id) |
| 前置条件 | 选择车辆，确认删除 |
| 后置条件 | 将ID发送到数据层并从库中删除相应车辆的信息 |
| VanManLogic.reviseVan(VanVO Van) | 语法 | public boolean reviseVan(VanVO Van) |
| 前置条件 | 确认修改车辆信息 |
| 后置条件 | 将车辆信息发送到数据层，并修改库中相应信息 |
| VanManLogic.checkVan(String id) | 语法 | public ResultMessage checkVan(String id) |
| 前置条件 | 从车辆列表中选择车辆选择查看 |
| 后置条件 | 从数据层获得相应车辆的信息返回 |
| VanManLogic.searchVan(String keyword) | 语法 | public ResultMessage searchVan(String keyword) |
| 前置条件 | 打开车辆管理界面或输入关键词检索 |
| 后置条件 | 从数据层获取车辆列表或含有相应关键词的车辆列表返回 |
| 需要的服务（需接口） | | |
| VanManDSer.addVan(VanPO Van) | 将PO中的车辆信息添加进数据库 | |
| VanManDSer.deleteVan(String id) | 将数据库中相应ID的车辆信息删除 | |
| VanManDSer.reviseVan(VanPO Van) | 在数据库中修改PO中信息相应的车辆信息 | |
| VanManDSer.checkVan(String id) | 在数据库中获得相应ID车辆的信息并返回 | |
| VanManDSer.findVan(String keyword) | 在数据库中获得含有相应关键词（若关键词为空则返回全部）的车辆信息列表返回。 | |

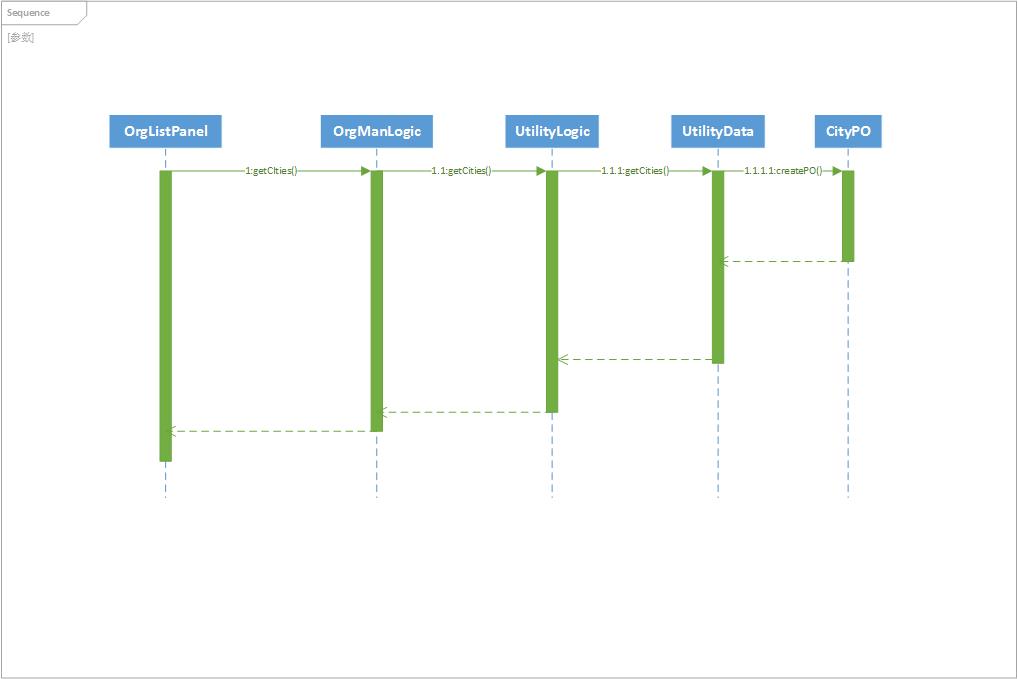
WageManLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| WageManLogic.searchWorker(String keyword) | 语法 | public ResultMessage searchWorker(String keyword) |
| 前置条件 | 打开薪水管理界面 |
| 后置条件 | 从数据库获得人员薪水列表返回 |
| WageManLogic.reviseWage(WageVO wage) | 语法 | public boolean reviseWage(WageVO wage) |
| 前置条件 | 确认薪水策略制定 |
| 后置条件 | 将薪水策略发送到数据层并修改数据库中相应人员的薪水策略 |
| 需要的服务（需接口） | | |
| WageManDSer.findWork(String keyword) | 在数据库中获得含有相应关键词（若关键词为空则返回全部）的人员薪水信息列表返回。 | |
| WageManDSer.reviseVan(WagePO Van) | 在数据库中修改PO中相应人员的薪水策略信息 | |

WorkManLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| WorkManLogic.addWork(WorkVO work) | 语法 | public boolean addWork(WorkVO Work) |
| 前置条件 | 已输入完整的人员信息，确认新增 |
| 后置条件 | 将人员信息发送到数据层并添加到数据库 |
| WorkManLogic.deleteWork(String id) | 语法 | public boolean deleteWork(String id) |
| 前置条件 | 选择人员，确认删除 |
| 后置条件 | 将ID发送到数据层并从库中删除相应人员的信息 |
| WorkManLogic.reviseWork(WorkVO work) | 语法 | public boolean reviseWork(WorkVO Work) |
| 前置条件 | 确认修改人员信息 |
| 后置条件 | 将人员信息发送到数据层，并修改库中相应信息 |
| WorkManLogic.searchWork(String keyword) | 语法 | public ResultMessage searchWork(String keyword) |
| 前置条件 | 打开机构管理界面或输入关键词检索 |
| 后置条件 | 从数据层获取机构列表或含有相应关键词的机构列表返回 |
| WorkManLogic.orgList() | 语法 | public ResultMessage orgList() |
| 前置条件 | 打开人员管理界面 |
| 后置条件 | 获得机构信息列表返回 |
| 需要的服务（需接口） | | |
| WorkManDSer.addWork(WorkPO Work) | 将PO中的人员信息添加进数据库 | |
| WorkManDSer.deleteWork(String id) | 将数据库中相应ID的人员信息删除 | |
| WorkManDSer.reviseWork(WorkPO Work) | 在数据库中修改PO中信息相应的人员信息 | |
| WorkManDSer.findWork(String keyword) | 在数据库中获得含有相应关键词（若关键词为空则返回全部）的人员信息列表返回。 | |
| UtilityDSer.getOrgs() | 在数据库中获得机构信息列表并返回 | |

1. 业务逻辑层的动态模型

下图表明了在打开机构管理界面后，获得城市信息列表时相关对象之间的协作。

1. 业务逻辑层的设计原理

见4.1.1.（5）

* + 1. accountLogic模块

1. 模块概述

accountLogic模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

accountLogic模块的职责及接口参见软件体系结构描述文档5.3.2中accountLogic模块的接口规范

1. 整体结构

根据体系结构的设计，分层结构分为界面、逻辑和数据层。界面和逻辑层间添加AccountLSer作为账户管理界面的业务逻辑接口，CountLSer作为账管理界面的业务逻辑接口。逻辑层和数据层间添加AccountDSer作为账户管理逻辑的数据层接口，CountDSer作为账管理逻辑的数据层接口。AccountVO和CountVO分别为账户和账的界面值对象，相应的PO为持久化对象。ResultMessage存放操作是否成功和要返回界面的信息。UtilityLogic负责处理一些逻辑初始化操作。

accountLogic模块各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| AccountLogic | 负责账户管理界面所需要的业务逻辑 |
| CountLogic | 负责期初建账和账信息界面所需要的业务逻辑 |

1. 模块内部类的接口规范

AccountLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| AccountLogic.addAccount(AccountVO account) | 语法 | public boolean addAccount(AccountVO account) |
| 前置条件 | 已输入完整的账户信息，确认新增 |
| 后置条件 | 将账户信息发送到数据层并添加到数据库 |
| AccountLogic.deleteAccount(String id) | 语法 | public boolean deleteAccount(String id) |
| 前置条件 | 选择账户，确认删除 |
| 后置条件 | 将ID发送到数据层并从库中删除相应账户的信息 |
| AccountLogic.reviseAccount(AccountVO account) | 语法 | public boolean reviseAccount(AccountVO account) |
| 前置条件 | 确认修改账户信息 |
| 后置条件 | 将账户信息发送到数据层，并修改库中相应信息 |
| AccountLogic.searchAccount(String keyword) | 语法 | public ResultMessage searchAccount(String keyword) |
| 前置条件 | 打开账户管理界面或输入关键词检索 |
| 后置条件 | 从数据层获取账户列表或含有相应关键词的账户列表返回 |
| 需要的服务（需接口） | | |
| AccountDSer.addAccount(AccountPO account) | 将PO中的账户信息添加进数据库 | |
| AccountDSer.deleteAccount(String id) | 将数据库中相应ID的账户信息删除 | |
| AccountDSer.reviseAccount(AccountPO account) | 在数据库中修改PO中信息相应的账户信息 | |
| AccountDSer.findAccount(String keyword) | 在数据库中获得含有相应关键词（若关键词为空则返回全部）的账户信息列表返回。 | |

CountLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CountLogic.newCount() | 语法 | public boolean newCount() |
| 前置条件 | 用户选择了新增期初信息 |
| 后置条件 | 获得当前所有机构、人员、车辆、库存和账户信息，将其处理存储为期初信息。 |
| CountLogic.checkInitInfo() | 语法 | public ResultMessage checkInitInfo() |
| 前置条件 | 用户选择账，选择查看期初信息 |
| 后置条件 | 从数据层获得期初信息返回给界面 |
| 需要的服务（需接口） | | |
| CountDSer.addCount(CountPO count) | 将PO中的账信息添加到数据库 | |
| CountDSer.findInitInfo() | 从数据库中获得期初信息 | |
| CountDSer.findOrg() | 从数据库中获得所有机构的信息 | |
| CountDSer.getWorkers() | 从数据库中获得所有人员的信息 | |
| CountDSer.getVans() | 从数据库中获得所有车辆的信息 | |
| CountDSer.getStocks() | 从数据库中获得所有库存的信息 | |
| CountDSer.getAccount() | 从数据库中获得所有账户的信息 | |

1. 业务逻辑层的动态模型
2. 业务逻辑层的设计原理

见4.1.1.（5）

* + 1. stockLogic模块

1. 模块概述

stockLogic模块承担的需求参见需求规格说明文档功能需求及相关非功能需求。

stockLogic模块的职责及接口参见软件体系结构描述文档5.3.2中stockLogic模块的接口规范

1. 整体结构

根据体系结构的设计，分层结构分为界面、逻辑和数据层。界面和逻辑层间添加StockCheckLSer作为库存查看界面的业务逻辑接口，StockCheckNowLSer作为库存盘点界面的业务逻辑接口，StockCheckWarnLSer作为库存报警界面的业务逻辑接口，StockDivideLSer作为库存分区界面的业务逻辑接口，StockWarningLSer作为库存报警设置界面的业务逻辑接口。逻辑层和数据层之间添加StockCheckDSer作为库存查看逻辑的数据层接口，StockCheckNowDSer作为库存盘点业务逻辑的数据层接口，StockCheckWarnDSer作为库存报警业务逻辑的数据层接口，StockDivideDSer作为库存分区业务逻辑的数据层接口，StockWarningDSer作为库存报警设置业务逻辑的数据层接口。StockVO,ShelfVO,GoodVO,StockInVO,StockOutVO分别存储库存、架、货物、入库单和出库单的值对象，相应的PO存储相应的持久化对象。ResultMessage存储操作是否成功和返回的信息。StockInOrderLogic是入库单生成逻辑，需要通过StockCheckWarnLogic检测是否需要报警。

stockLogic模块各个类的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| StockCheckLogic | 负责库存查看界面所需要的业务逻辑 |
| StockCheckNowLogic | 负责库存盘点界面所需要的业务逻辑 |
| StockCheckWarnLogic | 负责库存报警界面所需要的业务逻辑 |
| StockDivideLogic | 负责库存分区界面所需要的业务逻辑 |
| StockWarningLogic | 负责库存报警设置界面所需要的业务逻辑 |

1. 模块内部类的接口规范

StockCheckLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockCheckLogic.checkStockIn(Date start, Date end, String id) | 语法 | public ResultMessage checkStockIn(Date start, Date end, String id) |
| 前置条件 | 输入了开始日期和结束日期 |
| 后置条件 | 从数据层获取开始和结束日期之间的入库单 |
| StockCheckLogic.checkStockOut(Date start, Date end, String id) | 语法 | public ResultMessage checkStockOut(Date start, Date end, String id) |
| 前置条件 | 输入了开始日期和结束日期 |
| 后置条件 | 从数据层获取开始和结束日期之间的出库单 |
| 需要的服务（需接口） | | |
| StockCheckDSer.getIn(Date start,Date end,String id) | 从数据库中获得开始和结束日期之间的入库单 | |
| StockCheckDSer.getOut(Date start,Date end,String id) | 将数据库中获得开始和结束日期之间的出库单 | |

StockCheckNowLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockCheckNowLogic.checkNow(String id) | 语法 | public ResultMessage checkNow(String id) |
| 前置条件 | 打开库存盘点界面 |
| 后置条件 | 从数据层获取当前库存快照信息 |
| StockCheckLogic.setPoint(String id, Date date) | 语法 | public boolean setPoint(String id, Date date) |
| 前置条件 | 用户确认盘点 |
| 后置条件 | 将当前时间作为截止点传往数据层，存入数据库 |
| StockCheckLogic.outputExcel(String location, StockVO stock) | 语法 | public boolean outputExcel(String location, StockVO stock) |
| 前置条件 | 用户选择路径并确认导出为Excel |
| 后置条件 | 导出Excel并打开，返回操作是否成功 |
| 需要的服务（需接口） | | |
| StockCheckNowDSer.getStock(String id) | 从数据库中获得当前库存快照 | |
| StockCheckDNowSer.setPoint(String id,Date date) | 在数据库中添加库存盘点截止点 | |

StockCheckWarnLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockCheckWarnLogic.checkWarning (String id) | 语法 | public ResultMessage checkWarning(String id) |
| 前置条件 | 新建了一个入库单 |
| 后置条件 | 比较当前库存量和库存警戒线比例，返回是否超出警戒线 |
| 需要的服务（需接口） | | |
| StockCheckWarnDSer.getWarning(String id) | 从数据库中获得库存警戒线 | |
| StockCheckWarnDSer.getStock() | 从数据库获得当前库存情况 | |

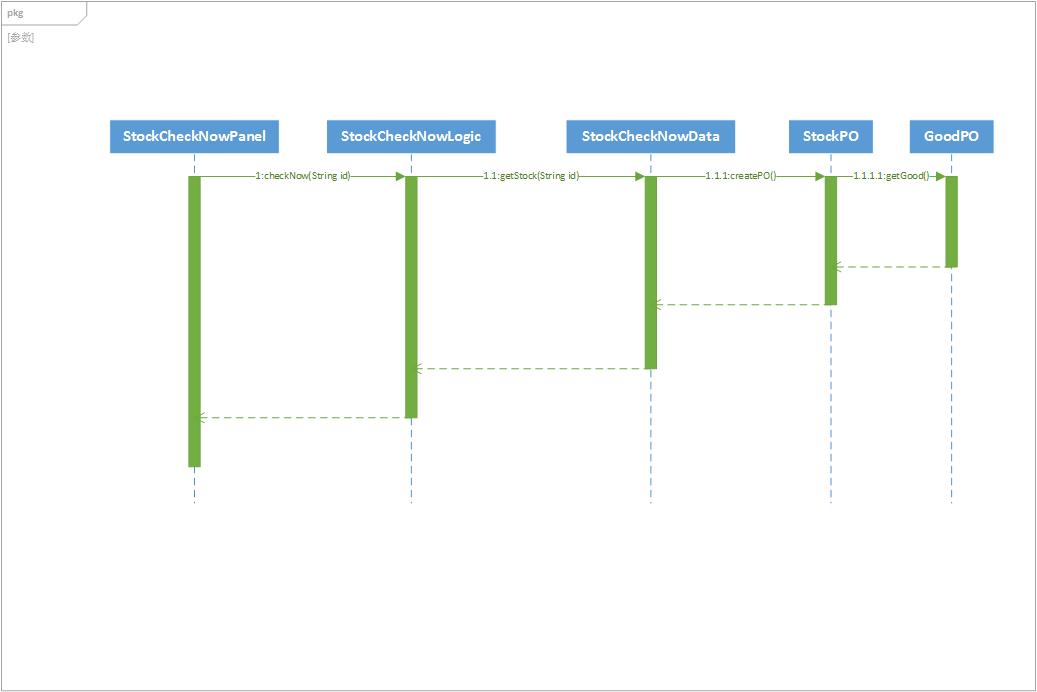
StockDivideLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockDivideLogic.searchShelf(String id,String keyword) | 语法 | public ResultMessage searchShelf(String id,String keyword) |
| 前置条件 | 打开库存分区界面或输入关键词搜索架 |
| 后置条件 | 从数据库获得架列表或含有关键词的架列表 |
| StockDivideLogic.addShelf(ShelfVO shelf) | 语法 | public boolean addShelf(ShelfVO shelf) |
| 前置条件 | 输入了完整的架信息，选择添加 |
| 后置条件 | 将架信息发送到数据层并添加入库 |
| StockDivideLogic.deleteShelf(String id) | 语法 | public boolean deleteShelf(String id) |
| 前置条件 | 选择一个架，确认删除 |
| 后置条件 | 从数据库中将相应ID的架信息清除 |
| StockDivideLogic.reviseShelf(ShelfVO shelf) | 语法 | public boolean reviseShelf(ShelfVO shelf) |
| 前置条件 | 修改一个架的信息，确认修改 |
| 后置条件 | 将架信息发送到数据层并修改库中的相应信息 |
| 需要的服务（需接口） | | |
| StockDivideDSer.getShelves(String id,String keyword) | 从数据库获得架列表或含有关键字的架列表 | |
| StockDivideDSer.addShelf(ShelfPO shelf) | 向数据库中添加架信息 | |
| StockDivideDSer.deleteShelf(String id) | 从数据库中删除ID相应的架信息 | |
| StockDivideDSer.reviseShelf(ShelfPO shelf) | 在数据库中修改相应ID的架信息 | |

StockWarningLogic的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockWarningLogic.setWarning(int w, Part part, String id) | 语法 | public boolean setWarning(int w, Part part, String id) |
| 前置条件 | 设置了一个区的警戒线 |
| 后置条件 | 将警戒线发送到数据层并修改数据库中的相应警戒线 |
| StockWarningLogic.getWarning(String id) | 语法 | public ResultMessage getWarning(String id) |
| 前置条件 | 打开库存分区界面 |
| 后置条件 | 从数据层中获得库存各区的警戒线 |
| 需要的服务（需接口） | | |
| StockWarningDSer.getWarning(String id) | 从数据库中获得库存警戒线 | |
| StockWarningDSer.setWarning(int w,Part part,String id) | 修改数据库中相应区的警戒线 | |

1. 业务逻辑层的动态模型

如图为执行库存盘点操作后，相关类的协作顺序图。

下图表现了库存报警的状态转换情况。

1. 业务逻辑层的设计原理

见4.1.1。（5）

1. 依赖视角