

gITmining软件体系结构描述文档

V1.0



2015-12-23

|  |  |  |  |
| --- | --- | --- | --- |
| 变更历史 | | | |
| 版本号 | **更新时间** | **更新人** | **更新摘要** |
| V1.0 | 2016/2/28 | 董本超 | 完成体系结构描述文档概要 |
|  |  |  |  |

目录

[1引言 3](#_Toc438668479)

[1.1编制目的 3](#_Toc438668480)

[1.2词汇表 3](#_Toc438668481)

[1.3参考资料 3](#_Toc438668482)

[2产品概述 3](#_Toc438668483)

[3逻辑视角 3](#_Toc438668484)

[4组合视角 6](#_Toc438668485)

[4.1开发包图 6](#_Toc438668486)

[4.2运行时进程 9](#_Toc438668487)

[4.3物理部署 9](#_Toc438668488)

[5接口视角 10](#_Toc438668489)

[5.1模块的职责 10](#_Toc438668490)

[5.2用户界面层的分解 15](#_Toc438668491)

[5.2.1用户界面层模块的职责 16](#_Toc438668492)

[5.2.2用户界面层模块的接口规范 17](#_Toc438668493)

[5.2.3用户界面模块设计原理 17](#_Toc438668494)

[5.3业务逻辑层的分解 17](#_Toc438668495)

[5.3.1业务逻辑层模块的职责 18](#_Toc438668496)

[5.3.2业务逻辑层模块的接口规范 18](#_Toc438668497)

[5.4数据层的分解 31](#_Toc438668498)

[5.4.1数据层模块的职责 35](#_Toc438668499)

[5.4.2数据层模块的接口规范 35](#_Toc438668500)

[6信息视角 40](#_Toc438668501)

[6.1数据持久化对象 40](#_Toc438668502)

[6.2数据库表 43](#_Toc438668503)

# 1引言

## 1.1编制目的

本报告详细完成对GitMining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。

## 1.2词汇表

|  |  |  |
| --- | --- | --- |
| **词汇名称** | **词汇含义** | **备注** |
| GitMining | 代码挖掘 | 挖掘并分析github网站上的用户和仓库数据 |

## 1.3参考资料

1.软件工程与计算(卷三) 团队与软件开发实践, 机械工业出版社, 2012;

2.GitMining迭代一用例文档;

3.GitMining迭代一需求规格说明文档;

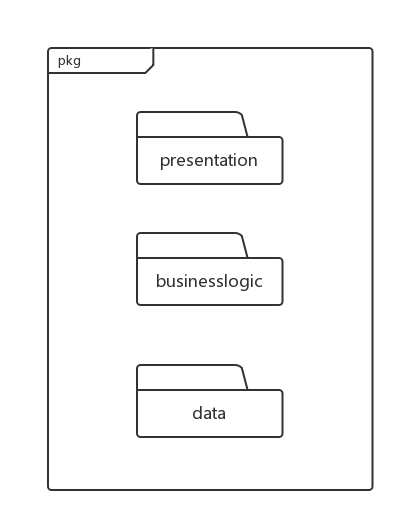
# 2产品概述

GitMining系统是获取并分析github网站上公开的用户信息和仓库信息并分析其相互之间关系、从而进行数据统计和归纳的系统。

该系统在后期迭代二或三中也将根据用户的兴趣进行分类数据检索和推荐，从而提供系统的实用性。

# 3逻辑视角

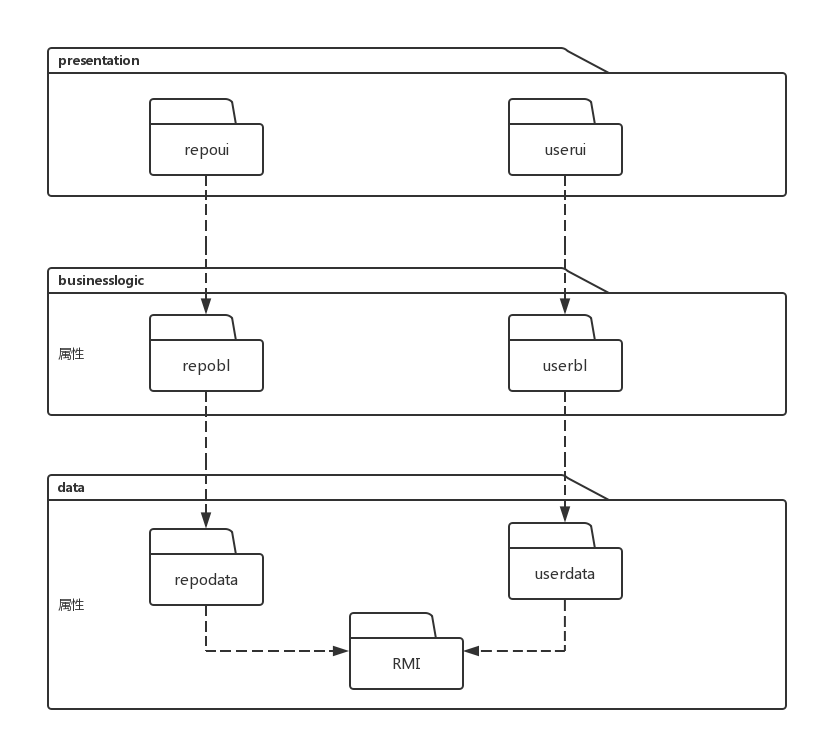
在迭代一阶段，系统暂且不需要服务器端。客户端将选择分层体系结构，将系统分为3层（展示层、业务逻辑层、数据层）。展示层包含GUI的实现，业务逻辑层包含业务逻辑处理的实现，数据层负责数据访问(注意: 迭代一阶段数据层不存储数据，只通过已有的api进行数据获取)。分层体系结构的逻辑视角如图1所示。

图1 参考体系结构风格的包图表达逻辑视角

GitMining系统在迭代一阶段的主要功能包括GitHub仓库信息查询和用户信息查询，它们对应的逻辑包如下：

|  |  |
| --- | --- |
| **功能** | **对应逻辑包** |
| 仓库信息查询 | repoui, repobl, repodata |
| 用户信息查询 | userui, userbl, userdata |

由上表信息可以生成软件体系结构逻辑设计方案，如图2所示。

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

GitMining迭代一阶段的开发包设计如表1所示。

**表1 GitMining开发包设计**

|  |  |
| --- | --- |
| **开发（物理）包** | **依赖的其他开发包** |
| repoui | repoblservice |
| userui | userblservice |
| repoblservice | vo |
| userblservice | vo |
| repobl | repoblservice，repodataservice |
| userbl | userblservice， userdataservice |
| repodataservice | repodata |
| userdataservice | userdata |
| repodata |  |
| userdata |  |
| vo |  |

Gitmining系统的开发包图如图3所示

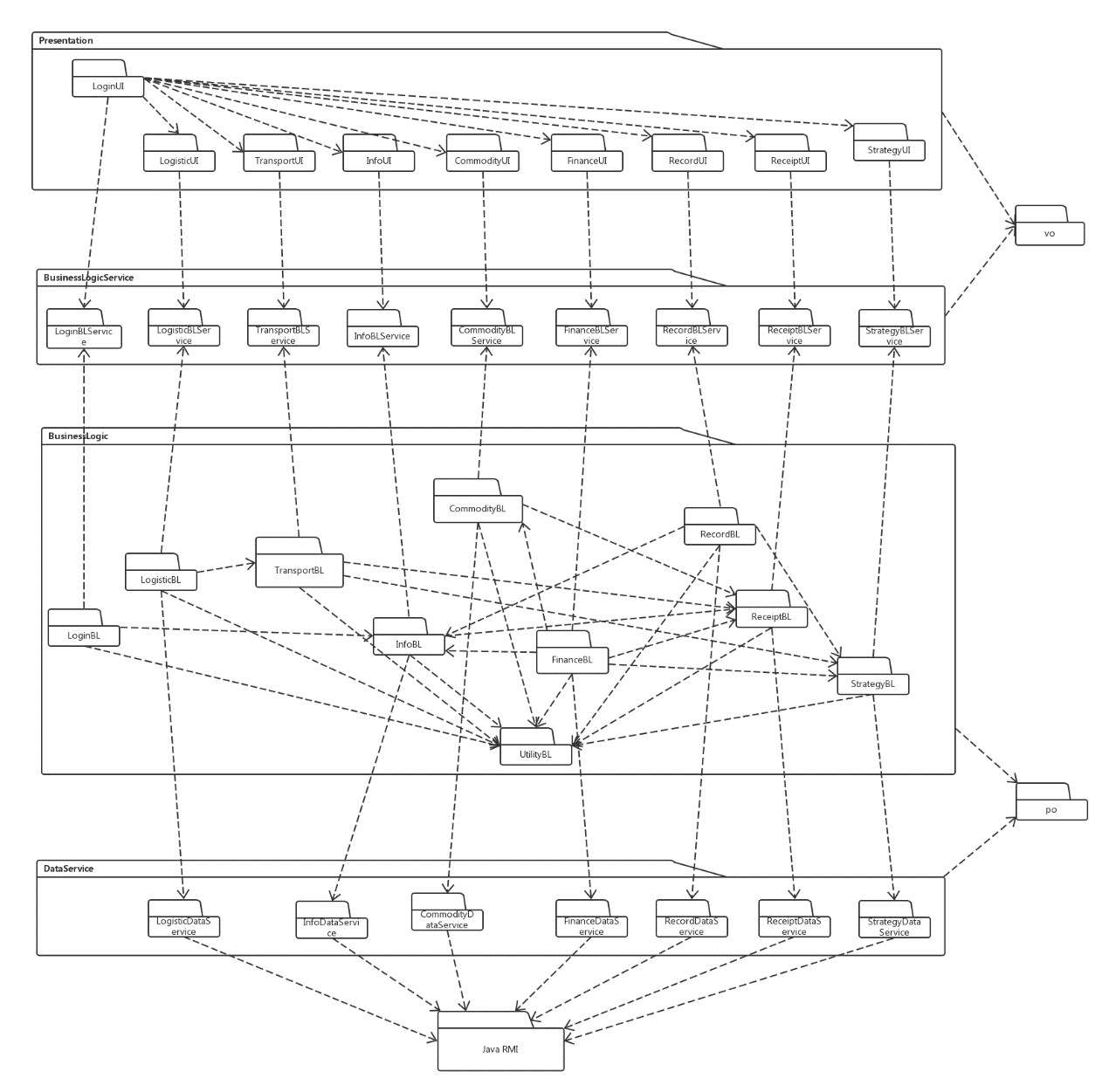
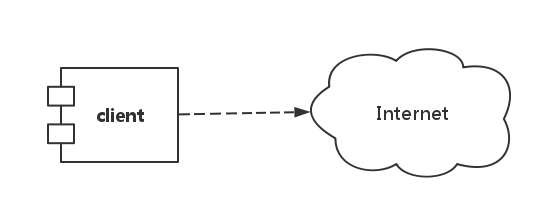


图3 GitMining系统开发包图

## 4.2运行时进程

在GitMining系统运行过程中，系统需要通过api从互联网获取信息，因此系统运行时必须联网，进程图如图4.

图4 进程图

# 5接口视角

## 5.1模块的职责

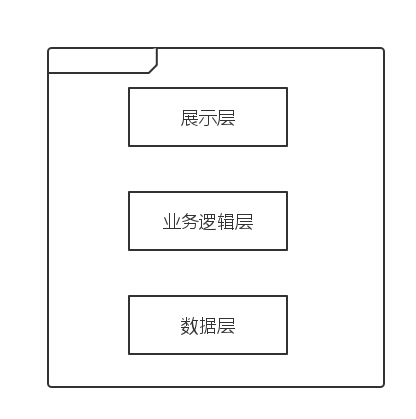
 GitMining系统的模块视图如图5所示。各层的职责如表2所示。

图5 客户端模块视图

**表2 GitMining系统各层的职责**

|  |  |
| --- | --- |
| **层** | **职责** |
| 展示层 | 将仓库或用户信息展示给系统的用户 |
| 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，访问数据层 |
| 数据层 | 利用已有的api联网获取数据并传给逻辑层 |

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。层之间调用的接口如表4所示。

**表3 层之间调用的接口**

|  |  |  |
| --- | --- | --- |
| **接口** | **服务调用方** | **服务提供方** |
| repoblservice  userblservice | 展示层 | 业务逻辑层 |
| repodataservice  userdataservice | 业务逻辑层 | 数据层 |

## 5.2用户界面层的分解

根据需求，系统存在？？？个用户界面：？？？。界面跳转如图6所示。

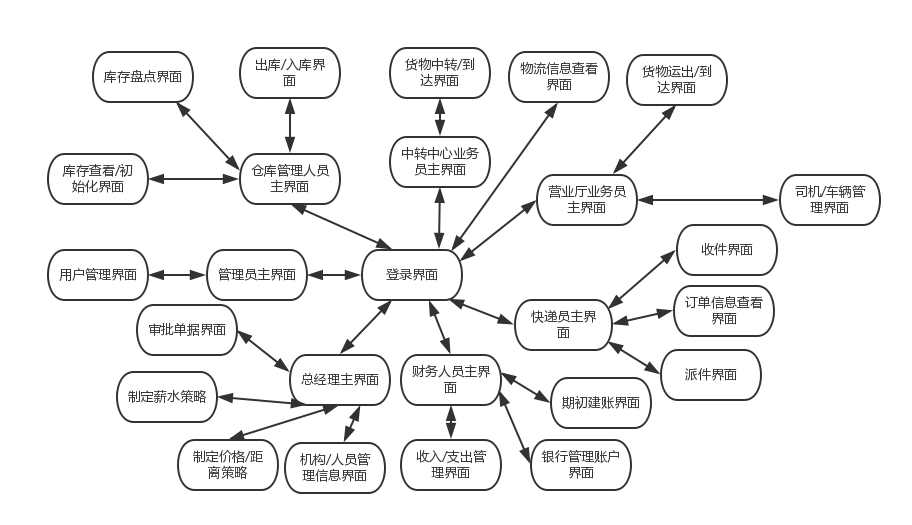


图6 用户界面跳转

## 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，repobl负责处理用户请求查询仓库信息的业务逻辑。具体职责说明如下。

### 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4所示。

**表4 业务逻辑层模块的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| repobl | 负责处理用户查询仓库信息的业务逻辑 |
| userbl | 负责处理用户查询github用户信息的业务逻辑 |

### 5.3.2业务逻辑层模块的接口规范

待讨论！！！！

## 5.4数据层的分解

由于该系统的迭代一阶段不涉及数据的本地存储，所有数据均直接从互联网上获得并及时解析，所以数据层暂且功能很少，只有从互联网获取数据并进行初步解析的功能。未来在迭代二或三阶段数据层将增加更多功能点。

### 5.4.1数据层模块的职责

由于各个功能模块对应数据层模块的职责基本相同，故这里只用一张表总结描述，如表18所示。

**表18 数据层模块的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| \*DataService | 持久化数据操作的接口，提供持久化数据的增、删、改、查服务 |

### 5.4.2数据层模块的接口规范

表19到表25为各个功能数据层模块的接口规范。

**表19 Logistic数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| LogisticDataService.update | 语法 | Public void update(LogisticPO po) throws RemoteException, SQLException; |
| 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 |
| LogisticDataService.read | 语法 | Public ArrayList<LogisticPO> read(String num) throws RemoteException, SQLException; |
| 前置条件 | 无 |
| 后置条件 | 读取物流信息 |
| LogisticDataService.remove | 语法 | Public void remove(String num) throws RemoteException, SQLException; |
| 前置条件 | 无 |
| 后置条件 | 删除物流信息持久化数据，被update调用 |

**表20 Agency数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| InfoDataService.getList | 语法 | public ArrayList<AgencyPO> getList() throws RemoteExceptoin, SQLException; |
| 前置条件 | 无 |
| 后置条件 | 返回AgencyPO列表 |
| InfoDataService.addItem | 语法 | public void addItem(AgencyPO item) throws RemoteException, InfoBLException, SQLException; |
| 前置条件 | 相同id的po在列表中不存在 |
| 后置条件 | 在持久化数据中增加一个po条目 |
| InfoDataService.deleteItem | 语法 | public void deleteItem(String id) throws RemoteException, SQLException |
| 前置条件 | 持久化数据列表中存在一个相同id的po |
| 后置条件 | 删除一个po |
| InfoDataService.update | 语法 | public void update(String id，AgencyPO item) throws RemoteException, InfoBLException, SQLException |
| 前置条件 | 在持久化数据列表中存在相同id的po |
| 后置条件 | 更新一个po |

BankAccount数据层，Driver数据层，Staff数据层，Truck数据层，UserAccount数据层与Agency类似，故不一一列举。

**表21 Commodity数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| CommodityDataService.check | 语法 | Public ArrayList<CommodityPO> check(String TransferNum) throws RemoteException, SQLException; |
| 前置条件 | 无 |
| 后置条件 | 返回指定中转中心仓库的库存信息 |
| CommodityDataService.getAll | 语法 | Public ArrayList<CommodityPO> getAll() throws RemoteException, SQLException |
| 前置条件 | 无 |
| 后置条件 | 返回所有中转中心的库存信息 |
| CommodityDataServicec.renew | 语法 | Public void renew()throws RemoteException, SQLException |
| 前置条件 | 无 |
| 后置条件 | 初始化库存数据块 |
| CommodityDataService.add | 语法 | Public void add(CommodityPO po) throws RemoteException, SQLException |
| 前置条件 | 无 |
| 后置条件 | 库存数据块更新库存信息 |
| CommodityDataServicec.delete | 语法 | Public void delete(String expressNum)throws RemoteException, SQLException |
| 前置条件 | 无 |
| 后置条件 | 初始化库存数据块 |

**表22 Record数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| RecordDataService.getRecord | 语法 | public ArrayList<RecordPO> getRecord() throws RemoteException, SQLException; |
| 前置条件 | 无 |
| 后置条件 | 返回系统日志 |
| RecordDataService.record | 语法 | public void record(RecordPO po) throws RemoteException, SQLException |
| 前置条件 | 系统有主要操作发生 |
| 后置条件 | 在系统日志中记录操作 |

**表23 ChargeReceipt数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| ChargeReceiptDataService.getList | 语法 | public ArrayList<ChargeReceiptPO> getList(Date fromTime,Date toTime) throws RemoteExceptoin, SQLException; |
| 前置条件 | 无 |
| 后置条件 | 按对应time返回相应ChargeReceiptPO列表 |
| ChargeReceiptDataService.addItem | 语法 | public void addItem(ChargeReceiptPO item) throws RemoteException, SQLException; |
| 前置条件 | 相同id的po在列表中不存在 |
| 后置条件 | 在持久化数据中增加一个po条目 |
| ChargeReceiptDataService.deleteAll | 语法 | public void deleteAll() throws RemoteException, SQLException |
| 前置条件 | 无 |
| 后置条件 | 清空数据库中所有chargeReceipt |

其他单据类似，故不一一列举。

**表24** **Finance数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| FinanceDataService.add | 语法 | Public void add (FinancePO po,int year) throws RemoteException, FileNotFoundException, IOException |
| 前置条件 | 无 |
| 后置条件 | 在账目数据库中增加一个持久化对象 |
| FinanceDataService.find | 语法 | Public FinancePO find(int year) throws RemoteException, FileNotFoundException, ClassNotFoundException, IOException |
| 前置条件 | 无 |
| 后置条件 | 根据年份信息返回账目数据库中的一个持久化对象 |
| FinanceDataService.addIncome(double income) | 语法 | Public void addIncome(double income) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 更新账目数据库中的总收入数据 |
| FinanceDataService.renewIncome() | 语法 | Public void renewIncome()throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 初始化账目数据库中的总收入数据 |
| FinanceDataService.getIncome() | 语法 | Public BigDecimal getIncome() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回账目数据库中的总收入数据 |
| FinanceDataService.addOutcome(double  outcome) | 语法 | Public void updateOutcome(double outcome) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 更新账目数据库中的总支出数据 |
| FinanceDataService.renewOutcome() | 语法 | Public void renewOutcome() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 初始化账目数据库中的总支出数据 |
| FinanceDataService.getOutcome() | 语法 | Public BigDecimal getOutcome() throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 返回账目数据库中的总支出数据 |

**表25 Strategy数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| StrategyDataService.getExpres  sFee | 语法 | public ExpressFeePO getExpressFee() throws RemoteException, SQLException |
| 前置条件 | 请求获得计算快递费用的策略 |
| 后置条件 | 返回计算快递费用的策略 |
| StrategyDataService.update  ExpressFeeStrategy | 语法 | public void updateExpressFeeStrategy  (ExpressFeePO efpo) throws RemoteException, SQLException |
| 前置条件 | 无 |
| 后置条件 | 更新计算快递费用的策略 |
| StrategyDataService.getCarriage  Fee | 语法 | Public CarriagePO getCarriageFee() throws RemoteException, SQLException |
| 前置条件 | 请求获得计算运费的策略 |
| 后置条件 | 返回计算运费的策略 |
| StrategyDataService.updateCarr  iageFee | 语法 | public void updateCarriageFeeStrategy  (CarriageFeePO cfpo) throws RemoteException, SQLException |
| 前置条件 | 无 |
| 后置条件 | 更新计算运费的策略 |
| StrategyDataService.getSalary | 语法 | public SalaryPO getSalary() throws RemoteException |
| 前置条件 | 请求获得薪水策略 |
| 后置条件 | 返回薪水策略 |
| StrategyDataService.update  SalaryStrategy | 语法 | public void updateSalaryStrategy(SalaryPO po) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 更新薪水策略 |
| StrategyDataService.getDistance | 语法 | public double getDistance(String city1,String city2) throws RemoteException,FileNotFoundException, ClassNotFoundException, IOException; |
| 前置条件 | 请求获得两城市间的距离 |
| 后置条件 | 返回两城市间的距离 |
| StrategyDataService.getDiatanceStrategy | 语法 | public DistancePO getDistanceStrategy() throws RemoteException, FileNotFoundException,ClassNotFoundException,IOException; |
| 前置条件 | 请求获得城市间距离策略 |
| 后置条件 | 返回城市间距离策略 |
| StrategyDataService.updateDistanceStrategy | 语法 | Public void updateDistanceStrategy(DistancePO po) throws RemoteException, FileNotFoundException, IOException; |
| 前置条件 | 无 |
| 后置条件 | 更新城市间距离策略 |

# 6信息视角

## 6.1数据持久化对象

系统的PO就是对应的需要持久化的类，所有的po类的介绍如下所示。

* LogisticPO类包含托运订单号、到达时间、货运状态（收件、到达寄件人营业厅、到达寄件人中转中心、到达收件人中转中心、到达收件人营业厅、派件中)。
* InfoPO类是车辆、司机、人员、机构、用户账户、银行账户的持久化信息父类，其对应的子类包括TruckPO、DriverPO、StaffPO、AgencyPO、UserAccountPO、BankAccountPO。

TruckPO类包含车辆代号、发动机号、车牌号、底盘号、购买时间、服役时间；DriverPO类包含司机编号、姓名、出生日期、身份证号、手机号、性别、行驶证期限；StaffPO类包含工号、姓名、性别、出生年月、所属部门、基本工资、快递员派件数、司机运货次数；AgencyPO类包含机构名称、机构类别（中转中心、营业厅）、机构编号、位置、占地面积、土地租金（元／平方米／月）；UserAccountPO类包含工号、姓名、所属部门、用户名、密码；BankAccountPO类包含账户名。

* CommodityPO类包含入库数量、金额、存储位置、库存变化数量。
* RecordPO类包含操作名称、发生时间。
* ReceiptPO类是寄件单、派件单、装车单、中转单、营业厅到达单、中转中心到达单、入库单、出库单、收款单、付款单的持久化信息父类，其对应的子类包括SendReceiptPO、DespatchReceiptPO、EntruckReceiptPO、TransferReceiptPO、StoreArrivalReceiptPO、HubArrivalReceiptPO、DepotInReceiptPO、DepotOutReceiptPO、ChargeReceiptPO、PayReceiptPO。

SendReceiptPO类包含寄件人姓名、住址、单位、手机，收件人姓名、住址、单位、手机；DespatchReceiptPO类包含到达日期、托运订单条形码号、派送员；EntruckReceiptPO类包含装车日期、营业厅／中转中心汽运编号、到达地、车辆代号、本次装箱所有订单条形码号、运费；

TransferReceiptPO类包含中转方式、中转日期、中转单编号、（飞机／铁路）班次或者汽车的车牌号、出发地、到达地、货柜号、本次装箱所有货物的单号；StoreArrivalReceiptPO类包含到达日期、（中转单编号）、出发地、货物到达状态（损坏、完整、丢失）；

HubArrivalReceiptPO类包含到达日期、中转单编号、出发地、货物到达状态（损坏、完整、丢失）；

DepotInReceiptPO类包含快递编号、目的地、入库日期、区号（汽运区区号、铁运区区号、航运区区号）、排号、架号、位号；

DepotOutReceiptPO类包含快递编号、出库日期、目的地、装运形式（火车、飞机、汽车）、中转单编号或者汽运编号；

ChargeReceiptPO类包含收款日期、收款金额、收款快递员、对应的所有快递订单条形码号;

PayReceiptPO类包含付款日期、付款金额、付款人、付款账号、条目（租金（按年收）、运费（按次计算）、人员工资（按月统计）、奖励（一次性））、备注。

* FinancePO类包含机构、人员、车辆、库存、银行账户信息（名称、余额）、总收入、总支出。
* StrategyPO类是价格策略、薪水策略的持久化信息的父类，其对应的子类包括和ExpressFeeVO、CarriageVO、SalaryVO、DistanceVO。
* SalaryFeePO类包含员工ID，姓名，职位，工资。

持久化对象ReceiveReceiptPO的定义如下所示，其他po对象详见代码

public class ReceiveReceiptPO extends ReceiptPO{

private static final long serialVersionUID = 1L;

private String receiveNum;

private String receiver;

private myTime receiveTime;

public ReceiveReceiptPO(String receiveNum,String receiver,myTime receiveTime) {

super(ReceiptType.RECEIVE);

this.setReceiveNum(receiveNum);

this.setReceiver(receiver);

this.setReceiveTime(receiveTime);

}

public String getReceiveNum() {

return receiveNum;

}

public void setReceiveNum(String receiveNum) {

this.receiveNum = receiveNum;

}

public String getReceiver() {

return receiver;

}

public void setReceiver(String receiver) {

this.receiver = receiver;

}

public myTime getReceiveTime() {

return receiveTime;

}

public void setReceiveTime(myTime receiveTime) {

this.receiveTime = receiveTime;

}

}

## 6.2数据库表

数据库表中包含agency表，bankaccount表，carriagefee表，chargereceipt表，commodity表，depotinreceipt表，depotoutreceipt表，despatchreceipt表，driver表，entruckreceipt表，expressfee表，hubarrivalreceipt表，logistic表，payreceipt表，profit表，receivereceipt表，record表，salary表，sendreceipt表，staff表，storearrivalreceipt表，transferreceipt表，truck表，useraccount表。