|  |  |
| --- | --- |
| Clientui | Clientblservice 界面库类包 |
| Clientbl | Clientblservice Clientdataservice |
| Clientblservice | Po |
| Clientdata | Java RMI,Po, Databaseutility |
| Clientdataservice | Java RMI,Po |
| Stockui | Stockblservice 界面库类包 |
| Stockbl | Stockblservice Stockdataservice |
| Stockblservice | Po |
| Stockdata | Java RMI,Po, Databaseutility |
| Stockdataservice | Java RMI,Po |
| Saleui | Saleblservice 界面库类包 |
| Salebl | Saleblservice Saledataservice |
| Saleblservice | Po |
| Saledata | Java RMI,Po, Databaseutility |
| Saledataservice | Java RMI,Po |

表16 ClienBl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Client.addClient | 语法 | public int addClient(ClientPO clientPO) throws Exception; |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 添加客户 |
| Client.deleteClient | 语法 | public void deleteClient(int id) throws Exception; |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 查找是否存在客户ID，存在则删除客户，否则抛出异常 |
| Client.modifyClient | 语法 | public void modifyClient(ClientPO newClientPO) throws Exception; |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 查找是否存在客户ID，存在则修改客户，否则抛出异常 |
| Client.queryClient | 语法 | public List<ClientPO> queryClient(Map<String, Object> filters) throws Exception; |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 查找是否存在满足过滤条件的客户，如果存在则添加到类表中，否则返回空列表 |
| Client.queryClientById | 语法 | public ClientPO queryClientById(int id) throws Exception; |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 查找是否存在此客户编号，如果存在则添加到类表中，否则返回空列表 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| ClientData.addClientData(ClientPO client) | 将新客户插入数据库 |
| ClientData.deleteClientData(int id) | 将客户从数据库移除 |
| ClientData.modifyClientData(ClientPO newClientPO) | 修改数据库中的客户数据 |
| ClientData.queryClientDataById(int id) | 根据ID查找数据库中的客户资料 |
| ClientData.queryClientData(Map<String, Object> filters) | 根据过滤条件查找客户 |

表17 StockBl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Stock.makeReceipt | 语法 | public void makeReceipt(StockReceiptPO receiptPO) throws Exception; |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 生成进货（退货）单据 |
| Stock.queryReceipt | 语法 | public List<StockReceiptPO> queryReceipt(ReceiptConditionVO filter) throws Exception; |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 返回符合过滤条件的单据 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| StockData.makeReceiptData(StockReceiptPO receipt) | 向数据库中插入进货（退货）单据 |
| StockData.queryReceiptData(ReceiptConditionVO filter) | 根据过滤条件查询单据 |

表18 SaleBl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Sale.makeReceipt | 语法 | public void makeReceipt(SaleReceiptPO receipt) throws Exception; |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 生成销售（退货）单据 |
| Sale.queryReceipt | 语法 | public Vector<SaleReceiptPO> queryReceipt(ReceiptConditionVO filter) throws Exception; |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 返回符合过滤条件的单据 |
| Sale.querySaleRecord | 语法 | public List<SaleReceiptPO> queryReceipt(ReceiptConditionVO filter) throws Exception; |
| 前置条件 | 用于已登录并被授权 |
| 后置条件 | 返回符合过滤条件的商品销售记录 |

|  |  |
| --- | --- |
| 需要的服务（需接口） | |
| 服务名 | 服务 |
| SaleData.makeReceiptData(SaleReceiptPO receipt) | 数据库中插入销售（退货）单据 |
| SaleData.queryReceiptData(ReceiptConditionVO filter) | 根据过滤条件查询单据 |
| SaleData. querySaleRecord(SaleConditionVO filter) | 根据过滤条件查询商品销售记录 |

表31 ClientData模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| ClientDataService.addClient | 语法 | public ResultMessage<Integer> addClient(ClientPO clientPO) throws RemoteException; |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 数据库中插入ClientPO对象 |
| ClientDataService.deleteClient | 语法 | public ResultMessage deleteClient(int id) throws RemoteException; |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 数据库中移除ClientPO对象 |
| ClientDataService.modifyClient | 语法 | public ResultMessage modifyClient(ClientPO newClientPO) throws RemoteException; |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 根据newClient修改orignClient |
| ClientDataService.queryClientById | 语法 | public ResultMessage<ClientPO> queryClientById(int id) throws RemoteException; |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 若此客户编号存在则返回ClientPO对象，否则返回null |
| ClientDataService.queryClient | 语法 | public ResultMessage<List<ClientPO>> queryClient(Map<String, Object> filters) throws RemoteException; |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 若有满足过滤条件的客户存在则返回ClientPO数组对象，否则返回空列表 |

表32 StockData模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockDataService.makeReceipt | 语法 | public ResultMessage makeReceipt(StockReceiptPO receiptPO) throws RemoteException; |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 数据库中插入StockReceiptPO 对象 |
| StockDataService.queryReceipt | 语法 | public ResultMessage<List<StockReceiptPO>> queryReceipt(ReceiptConditionVO filter) throws RemoteException; |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 返回符合条件的单据 |

表33 SaleData模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| SaleDataService.makeReceipt | 语法 | public ResultMessage makeReceipt(SaleReceiptPO receiptPO) throws RemoteException; |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 数据库中插入SaleReceiptPO 对象 |
| SaleDataService.queryReceipt | 语法 | public ResultMessage<List<SaleReceiptPO>> queryReceipt(ReceiptConditionVO filter)  throws RemoteException; |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 返回符合条件的单据 |
| SaleDataService.querySaleRecord | 语法 | public ResultMessage<List<GoodsRecordVO>> querySaleRecord(SaleConditionVO filter) throws RemoteException; |
| 前置条件 | 用户已登录并被授权 |
| 后置条件 | 返回符合条件的商品销售记录 |

信息视角，数据库表的定义：

create table commoditycategory

(

pkey int unsigned not null auto\_increment,

pid int unsigned not null,

name varchar(32) not null,

typesofar varchar(128) not null,

primary key(pkey),

unique(typesofar)

) ENGINE=MEMORY;

create table commodity

(

pkey int unsigned not null auto\_increment,

categoryId int unsigned not null,

serial\_number varchar(128) not null,

name varchar(32) not null,

type varchar(32) not null,

`signal` int unsigned not null,

model varchar(32) not null,

everhas char(1) not null,

gift char(1) not null,

amount int unsigned not null,

inprice decimal(12,2) not null,

outprice decimal(12,2) not null,

lastinprice decimal(12,2) not null,

lastoutprice decimal(12,2) not null,

primary key(pkey),

foreign key(categoryId) references commoditycategory(pkey)

) ENGINE=MEMORY;

create table client

(

pkey int unsigned not null auto\_increment,

type enum('STOCKER', 'SELLER') not null,

level enum('LEVEL1', 'LEVEL2', 'LEVEL3', 'LEVEL4', 'LEVEL5') not null,

name varchar(32) not null,

phone char(11) not null,

address varchar(255) not null,

zip varchar(32) not null,

email varchar(255) not null,

upperBound decimal(12,2) not null,

toReceive decimal(12,2) not null,

toPay decimal(12,2) not null,

defaultSalesMan varchar(32) not null,

primary key(pkey)

) ENGINE=MEMORY;

create table receipt

(

pkey int unsigned not null auto\_increment,

statement enum('approve', 'wait', 'disapprove') not null,

serial\_number char(20) not null,

type enum('STOCK\_ACCEPT', 'STOCK\_REJECTION', 'SALE\_ACCEPT', 'SALE\_REJECTION', 'RECEIVE', 'PAYMENT', 'CASH', 'REPORECEIPT', 'GOODSRECEIPT'),

time date not null,

operator varchar(32) not null,

primary key(pkey),

unique(serial\_number)

) ENGINE=MEMORY;

create table stockreceipt

(

pkey int unsigned not null auto\_increment,

pid int unsigned not null,

clientId int unsigned not null,

repository varchar(32) not null,

comment varchar(255) not null,

total decimal(12,2) not null,

primary key(pkey),

foreign key(pid) references receipt(pkey),

foreign key(clientId) references client(pkey)

) ENGINE=MEMORY;

create table salereceipt

(

pkey int unsigned not null auto\_increment,

pid int unsigned not null,

clientId int unsigned not null,

strategyId int unsigned not null,

salesman varchar(32) not null,

repository varchar(32) not null,

total decimal(12,2) not null,

allowance decimal(12,2) not null,

coupon decimal(12,2) not null,

actualValue decimal(12,2) not null,

comment varchar(255) not null,

primary key(pkey),

foreign key(pid) references receipt(pkey),

foreign key(clientId) references client(pkey),

foreign key(strategyId) references strategy(pkey)

) ENGINE=MEMORY;

create table productsreceipt

(

pkey int unsigned not null auto\_increment,

pid int unsigned not null,

commodityId int unsigned not null,

amount int unsigned not null,

price decimal(12,2) not null,

total decimal(12,2) not null,

comment varchar(255) not null,

primary key(pkey),

foreign key(pid) references receipt(pkey),

foreign key(commodityId) references commodity(pkey)

) ENGINE=MEMORY;

create table bankaccount

(

pkey int unsigned not null auto\_increment,

name varchar(32) not null,

price decimal(12,2) not null,

comment varchar(255) not null,

primary key(pkey),

unique (name)

) ENGINE=MEMORY;

create table recpayreceipt

(

pkey int unsigned not null auto\_increment,

pid int unsigned not null,

clientId int unsigned not null,

total decimal(12,2) not null,

state enum('DEAL', 'UNDEAL') not null,

primary key(pkey),

foreign key(pid) references receipt(pkey),

foreign key(clientId) references client(pkey)

) ENGINE=MEMORY;

create table transfer

(

pkey int unsigned not null auto\_increment,

pid int unsigned not null,

accountId int unsigned not null,

price decimal(12,2) not null,

comment varchar(255) not null,

primary key(pkey),

foreign key(pid) references receipt(pkey),

foreign key(accountId) references bankaccount(pkey)

) ENGINE=MEMORY;

create table admin

(

pkey int unsigned not null auto\_increment,

id int unsigned not null,

name varchar(32) not null,

password varchar(32) not null,

role varchar(32) not null,

primary key(pkey),

unique (name)

) ENGINE=MEMORY;

create table cash

(

pkey int unsigned not null auto\_increment,

pid int unsigned not null,

accountId int unsigned not null,

total decimal(12,2) not null,

state enum('DEAL', 'UNDEAL') not null,

primary key(pkey),

foreign key(pid) references receipt(pkey),

foreign key(accountId) references bankaccount(pkey)

) ENGINE=MEMORY;

create table cashitem

(

pkey int unsigned not null auto\_increment,

pid int unsigned not null,

name varchar(32) not null,

price decimal(12,2) not null,

comment varchar(255) not null,

primary key(pkey),

foreign key(pid) references receipt(pkey)

) ENGINE=MEMORY;

create table strategy

(

pkey int unsigned not null auto\_increment,

condition\_type enum('CUSTOMERLEVEL', 'TOTALPRICE', 'COMPOSITION') not null,

privilege\_type enum('GIVE', 'DISCOUNT', 'COUPON') not null,

begin date not null,

end date not null,

primary key(pkey)

) ENGINE=MEMORY;

create table strategycondition

(

pkey int unsigned not null auto\_increment,

pid int unsigned not null,

data int unsigned not null,

primary key(pkey),

foreign key (pid) references strategy(pkey)

) ENGINE=MEMORY;

create table privilege

(

pkey int unsigned not null auto\_increment,

pid int unsigned not null,

data int unsigned not null,

primary key(pkey),

foreign key (pid) references strategy(pkey)

) ENGINE=MEMORY;

create table reporeceipt

(

pkey int unsigned not null auto\_increment,

pid int unsigned not null,

commodityId int unsigned not null,

actual int unsigned not null,

statistic int unsigned not null,

primary key(pkey),

foreign key(pid) references receipt(pkey),

foreign key(commodityId) references commodity(pkey)

) ENGINE=MEMORY;

create table goodreceipt

(

pkey int unsigned not null auto\_increment,

pid int unsigned not null,

sum decimal(12,2) not null,

primary key(pkey),

foreign key(pid) references receipt(pkey)

) ENGINE=MEMORY;

create table goods

(

pkey int unsigned not null auto\_increment,

pid int unsigned not null,

commodityId int unsigned not null,

inprice decimal(12,2) not null,

outprice decimal(12,2) not null,

amount int unsigned not null,

primary key(pkey),

foreign key(pid) references receipt(pkey),

foreign key(commodityId) references commodity(pkey)

) ENGINE=MEMORY;

create table account

(

pkey int unsigned not null auto\_increment,

name varchar(32) not null,

primary key(pkey),

unique (name)

) ENGINE = MEMORY;

create table repo

(

pkey int unsigned not null auto\_increment,

batchNum varchar(32) not null,

time date not null,

primary key(pkey)

) ENGINE = MEMORY;

create table repogoods

(

pkey int unsigned not null auto\_increment,

pid int unsigned not null,

commodityId int unsigned not null,

amount int unsigned not null,

aver\_price decimal(12,2) not null,

primary key(pkey),

foreign key(pid) references repo(pkey),

foreign key(commodityId) references commodity(pkey)

) ENGINE = MEMORY;