数据层模块的接口规范

UserDSer的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UserManDSer.addUser | 语法 | public Result addUser(UserPO user) throws RemoteException |
| 前置条件 | 同样ID的po在数据库中不存在 |
| 后置条件 | 在数据库中增加一个PO记录 |
| UserManDSer.deleteUser | 语法 | public Result deleteUser(String id) throws RemoteException |
| 前置条件 | 数据库中存在同样ID的用户 |
| 后置条件 | 在数据库中删除一个PO记录 |
| UserManDSer.reviseUser | 语法 | public Result reviseUser(UserPO user) throws RemoteException |
| 前置条件 | 在数据库中存在同样ID的用户 |
| 后置条件 | 在数据库中修改该用户的数据 |
| UserManDSer.findUser | 语法 | public ArrayList<UserPO> findUser(String keyword) throws RemoteException |
| 前置条件 | 数据库中存在含关键字的用户（关键字为空不检索） |
| 后置条件 | 返回数据库中含关键字用户数据的列表（关键字为空返回全部） |
| LoginDSer.findUser | 语法 | public UserPO findUser(String id,String password) throws RemoteException |
| 前置条件 | 用户输入用户名密码登录时 |
| 后置条件 | 返回数据库中含关键字用户PO信息 |

TransitInfoDSer的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| Courier.find | 语法 | public ExpressPO find(String barcode) throws RemoteException |
| 前置条件 | 需要根据barcode获取快递单信息 |
| 后置条件 | 按barcode进行查找返回相应的PO结果 |
| Customers.findTransit | 语法 | public ArrayList<String> findTransit(String barcode) throws RemoteException |
| 前置条件 | 需要通过barcode获取物流轨迹列表 |
| 后置条件 | 按barcode进行查找返回相应订单的物流轨迹和货运信息 |

CityDSer的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| CityDSer.addCity | 语法 | public Result addCity (CityPO city) throws RemoteException |
| 前置条件 | 该city在数据库中不存在 |
| 后置条件 | 在数据库中增加一个city记录 |
| CityDSer.reviseCity | 语法 | public Result reviseCity(CityPO city) throws RemoteException |
| 前置条件 | 该city在数据库中已存在 |
| 后置条件 | 在数据库中更新该city |
| CityDSer.deleteCity | 语法 | public Result deleteCity(String cityname) throws RemoteException |
| 前置条件 | Cityname在数据库中存在 |
| 后置条件 | 在数据库中删除该city |
| CityDSer.addCities | 语法 | public Result addCities(CitiesPO cities) throws RemoteException |
| 前置条件 | 该cities在数据库中不存在 |
| 后置条件 | 在数据库中添加该Cities |
| CityDSer.deleteCities | 语法 | public Result deleteCities(CitiesPO cities) throws RemoteException |
| 前置条件 | 该CitiesPO在数据库中已存在 |
| 后置条件 | 在数据库中删除该citiesPO信息 |
| CityDSer.reviseCities | 语法 | public Result reviseCities(CityPO cities) throws RemoteException |
| 前置条件 | 该citiesPO在数据库中已存在 |
| 后置条件 | 改变城市之间距离和价格 |
| CityDSer.getCitiesInfo | 语法 | public ArrayList<CityPO> getCitiesInfo () throws RemoteException |
| 前置条件 | 需要获取城市间信息 |
| 后置条件 | 返回城市间信息列表 |
| CityDSer.isCityUsed | 语法 | public Result isCityUsed(String name,String id) throws Remote Exception |
| 前置条件 | 创建或修改城市信息后 |
| 后置条件 | 检查城市间信息是否被使用并返回结果 |
| ConstantDSer.getConstant | 语法 | public ConstantPO getConstant() throws Remote Exception |
| 前置条件 | 需要获取常量时 |
| 后置条件 | 从数据库中获取并返回常量信息 |
| ConstantDSer.setConstant | 语法 | public Result setConstant (ConstantPO constant) throws Remote Exception |
| 前置条件 | 设置常量后 |
| 后置条件 | 在数据库中更新常量信息 |

AccountDSer的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| AccountDSer.findAccount | 语法 | public ArrayList<AccountPO> findAccount(String keyword) throws RemoteException |
| 前置条件 | 无 |
| 后置条件 | 按关键字查找返回符合条件的账户列表 |
| AccountDSer.addAccount | 语法 | public Result addAccount(AccountPO account) throws RemoteException |
| 前置条件 | 同样账户名的账户在数据库中不存在 |
| 后置条件 | 在数据库中增加一个po记录 |
| AccountDSer.deleteAccount | 语法 | public Result deleteAccount(int id) throws RemoteException |
| 前置条件 | 该id对应的账户在数据库中存在 |
| 后置条件 | 在数据库中删除一个po记录 |
| AccountDSer.reviseAccount | 语法 | public Result reviseAccount(AccountPO po) throws RemoteException |
| 前置条件 | 该账户在数据库中存在 |
| 后置条件 | 在数据库中更新该账户信息 |
| CountDSer.addCount | 语法 | public Result addCount(CountPO count) throws RemoteException |
| 前置条件 | 同样账户名的账户在数据库中不存在 |
| 后置条件 | 在数据库中增加一个po记录 |
| CountDSer.getCount | 语法 | public ArrayList<CountPO> getCount() throws RemoteException |
| 前置条件 | 需要获取期初信息 |
| 后置条件 | 在数据库中获取并返回期初信息 |
| CountDSer.deleteCount | 语法 | public Result deleteCount(int id) throws RemoteException |
| 前置条件 | Id信息在数据库中存在 |
| 后置条件 | 在数据库中删除id对应的账的信息 |

OrderNewDSer的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderNewDSer.ArriveOrderDSer.create | 语法 | public Result create (ArrivePO order) throws RemoteException |
| 前置条件 | 单据已生成，且信息完整 |
| 后置条件 | 在数据库中新增一个ArriveOrder记录 |
| OrderNewDSer.CenterLoadingOrderDSer.create | 语法 | public Result create(CenterLoadingPO order) throws RemoteException |
| 前置条件 | 单据已生成，且信息完整 |
| 后置条件 | 在数据库中新增一个CenterLoadingOrder记录 |
| OrderNewDSer. \*.getOrderSize | 语法 | public int getOrderSize(String partId)throws RemoteException |
| 前置条件 | 需要获得当天生成单据数量 |
| 后置条件 | 在数据库中获取并返回当天单据数量 |
| OrderNewDSer.ExpressOrderDSer.create | 语法 | public boolean create (ExpressPO order) throws RemoteException |
| 前置条件 | 单据已生成，且信息完整 |
| 后置条件 | 在数据库中新增一个Order记录 |
| OrderNewDSer.HallLoadingOrderDSer.create | 语法 | public boolean create (HallLoadingPO order) throws RemoteException |
| 前置条件 | 单据已生成，且信息完整 |
| 后置条件 | 在数据库中新增一个Order记录 |
| OrderNewDSer.IncomeOrderDSer.create | 语法 | public boolean create (IncomePO order) throws RemoteException |
| 前置条件 | 单据已生成，且信息完整 |
| 后置条件 | 在数据库中新增一个Order记录 |
| OrderNewDSer.PaymentOrderDSer.create | 语法 | public boolean create (PaymentPO order) throws RemoteException |
| 前置条件 | 单据已生成，且信息完整 |
| 后置条件 | 在数据库中新增一个Order记录 |
| OrderNewDSer.ReceiveOrderDSer.create | 语法 | public boolean create (ReceivePO order) throws RemoteException |
| 前置条件 | 单据已生成，且信息完整 |
| 后置条件 | 在数据库中新增一个Order记录 |
| OrderNewDSer.ReceiveOrderDSer.getExpress | 语法 | public ExpressPO getExpress (String id) throws RemoteException |
| 前置条件 | 单据已生成，且信息完整 |
| 后置条件 | 在数据库中获取并返回物流信息 |
| OrderNewDSer.RecipientOrderDSer.create | 语法 | public boolean create (RecipientPO order) throws RemoteException |
| 前置条件 | 单据已生成，且信息完整 |
| 后置条件 | 在数据库中新增一个Order记录 |
| OrderNewDSer.SendOrderDSer.create | 语法 | public boolean create (SendPO order) throws RemoteException |
| 前置条件 | 单据已生成，且信息完整 |
| 后置条件 | 在数据库中新增一个Order记录 |
| OrderNewDSer.SendOrderDSer.getCouriers | 语法 | public ArrayList<String> getCouriers () throws RemoteException |
| 前置条件 | 单据已生成，且信息完整 |
| 后置条件 | 在数据库中获取并返回快递员信息 |
| OrderNewDSer.StockInOrderDSer.create | 语法 | public boolean create (StockInPO order) throws RemoteException |
| 前置条件 | 单据已生成，且信息完整 |
| 后置条件 | 在数据库中新增一个Order记录 |
| OrderNewDSer.StockInOrderDSer.getShelves | 语法 | public ArrayList<ShelfPO> getShelves() throws RemoteException |
| 前置条件 | 单据新建逻辑请求架信息 |
| 后置条件 | 在数据库中获取并返回架信息 |
| OrderNewDSer.StockInOrderDSer.addGood | 语法 | public Result addGood(GoodPO good)throws RemoteException |
| 前置条件 | 入库单生成后 |
| 后置条件 | 在数据库中添加货物的储存信息 |
| OrderNewDSer.StockOutOrderDSer.create | 语法 | public Result create (StockOutPO order) throws RemoteException |
| 前置条件 | 单据已生成，且信息完整 |
| 后置条件 | 在数据库中新增一个Order记录 |
| OrderNewDSer.StockOutOrderDSer.removeGood | 语法 | public Result removeGood(String id) throws RemoteException |
| 前置条件 | 出库单创建成功 |
| 后置条件 | 在数据库中删除货物并返回结果 |
| OrderNewDSer.TransitOrderDSer.create | 语法 | public Result create (TransitPO order) throws RemoteException |
| 前置条件 | 单据已生成，且信息完整 |
| 后置条件 | 在数据库中新增一个Order记录 |
| OrderNewDSer.UpdateTransitOrderDSer.addInfo | 语法 | public Result addInfo(TransitInfoPO info) throws RemoteException |
| 前置条件 | 快递已寄件 |
| 后置条件 | 创建快递的物流信息并返回结果 |
| OrderNewDSer.UpdateTransitOrderDSer.update | 语法 | public Result update(TransitInfoPO info) throws RemoteException |
| 前置条件 | 新单据已创建成功 |
| 后置条件 | 在该快递物流信息中添加新的物流轨迹 |
| OrderNewDSer.UpdateTranStateDSer. updateTranState | 语法 | public Result updateTranState(UpdateTranState order) throws RemoteException |
| 前置条件 | 快递有新的订单生成 |
| 后置条件 | 创建单据的物流状态并返回结果 |
| OrderNewDSer. UpdateTranStateDSer.update | 语法 | public List<String> getOrdersHere (LocationFind order) throws RemoteException |
| 前置条件 | 需要获得当前机构所有订单 |
| 后置条件 | 根据当前机构ID从数据库中获得所有订单列表并返回 |

workOrgManDSer的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| DriverManDSer.findDriver | 语法 | public ArrayList<DriverPO> findDriver(String hallId) throws RemoteException |
| 前置条件 | 需要获得所有的Driver信息 |
| 后置条件 | 按营业厅Id查找并返回相应的DriverPO列表 |
| DriverManDSer.addDriver | 语法 | public Result addDriver (DriverPO po) throws RemoteException |
| 前置条件 | 新人员信息已输入完全并确认 |
| 后置条件 | 在数据库中增加一个PO记录 |
| DriverManDSer.deleteDriver | 语法 | public Result deleteDriver (String id) throws RemoteException |
| 前置条件 | 数据库中存在相应ID的司机 |
| 后置条件 | 在数据库中删除相应的司机 |
| DriverManDSer.reviseDriver | 语法 | public Result reviseDriver(DriverPO) throws RemoteException |
| 前置条件 | 数据库中存在相应ID的司机 |
| 后置条件 | 在数据库中更新相应ID司机的信息 |
| OrgManDSer.findOrg | 语法 | public ArrayList<OrgPO> findOrg(String keywords) throws RemoteException |
| 前置条件 | 逻辑层需要搜索获取机构PO信息 |
| 后置条件 | 按关键词查找并返回相应的OrgPO结果 |
| OrgManDSer.addOrg | 语法 | public Result addOrg(OrgPO po) throws RemoteException |
| 前置条件 | 新机构信息已输入完全并确认 |
| 后置条件 | 在数据库中增加一个PO记录 |
| OrgManDSer.deleteOrg | 语法 | public Result deleteOrg(String id) throws RemoteException |
| 前置条件 | 数据库中存在相应ID的机构 |
| 后置条件 | 在数据库中删除相应机构的信息 |
| OrgManDSer.reviseOrg | 语法 | public Result reviseOrg(OrgPO) throws RemoteException |
| 前置条件 | 数据库中存在相应ID的机构 |
| 后置条件 | 在数据库中更新相应机构的信息 |
| VanManDSer.findVan | 语法 | public ArrayList<VanPO> findVan(String keyword) throws RemoteException |
| 前置条件 | 逻辑层需要搜索获取相应的车辆信息 |
| 后置条件 | 按关键词查找并返回相应的VanPO结果 |
| VanManDSer.addVan | 语法 | public Result addVan(VanPO Van) throws RemoteException |
| 前置条件 | 新货车信息已输入完全并确认 |
| 后置条件 | 在数据库中增加一个PO记录 |
| VanManDSer.deleteVan | 语法 | public Result deleteVan(String id) throws RemoteException |
| 前置条件 | 数据库中存在相应ID的货车 |
| 后置条件 | 在数据库中删除相应货车的信息 |
| VanManDSer.reviseVan | 语法 | public Result reviseOrg(VanPO Van) throws RemoteException |
| 前置条件 | 数据库中存在相应ID的货车 |
| 后置条件 | 在数据库中更新相应机构的信息 |
| WageManDSer.searchWork | 语法 | public List<WorkPO> searchWork (String keyword) throws RemoteException |
| 前置条件 | 逻辑层需要获取关键字的人员信息 |
| 后置条件 | 数据库中获取并返回关键字有关的人员信息 |
| WageManDSer.updateWork | 语法 | public Result updateWork (WorkPO work) throws RemoteException |
| 前置条件 | 逻辑层更新了人员薪水制定信息 |
| 后置条件 | 在数据库中更新人员的薪水信息 |
| WorkManDSer.findWork | 语法 | public ArrayList<WorkPO> findWork(String keyword) throws RemoteException |
| 前置条件 | 逻辑层需要获取关键字相关的人员信息 |
| 后置条件 | 按关键词查找并返回相应的WorkPO列表 |
| WorkManDSer.addWork | 语法 | public Result addWork(WorkPO work) throws RemoteException |
| 前置条件 | 新工人信息已输入完全并确认 |
| 后置条件 | 在数据库中增加一个PO记录 |
| WorkManDSer.deleteWork | 语法 | public Result deleteWork(String id) throws RemoteException |
| 前置条件 | 数据库中存在相应ID的工人 |
| 后置条件 | 在数据库中删除相应工人的信息 |
| WorkManDSer.reviseWork | 语法 | public Result reviseWork(WorkPO work) throws RemoteException |
| 前置条件 | 数据库中存在相应ID的工人 |
| 后置条件 | 在数据库中更新相应工人的信息 |

StockDSer的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| StockCheckDSer.getIn | 语法 | public ArrayList<StockInPO> getIn(Calendar start, Calendar end,String id) throws RemoteException |
| 前置条件 | 逻辑层需要获取入库单列表 |
| 后置条件 | 返回数据库中开始时间和结束时间之间的入库单列表 |
| StockCheckDSer.getOut | 语法 | Public ArrayList<StockOutPO> getOut (Calendar start, Calendar end,String id) throws RemoteException |
| 前置条件 | 逻辑层需要获取出库单列表 |
| 后置条件 | 返回数据库中开始时间和结束时间之间的出库单列表 |
| StockCheckNowDSer.getStock | 语法 | public StockPO getStock(String id)throws RemoteException |
| 前置条件 | 逻辑层需要获取库存情况 |
| 后置条件 | 返回数据库中库存的情况，包括各个架上的货物 |
| StockDivideDSer.getShelves | 语法 | public ArrayList<ShelfPO> getShelves(String id,String keyword)throws RemoteException |
| 前置条件 | 逻辑层需要获取架的信息 |
| 后置条件 | 返回数据库中所有架的信息 |
| StockDivideDSer.addShelf | 语法 | public Result addShelf(ShelfPO shelf) throws RemoteException |
| 前置条件 | 数据库中不含有同样ID的架 |
| 后置条件 | 在数据库中增加一个架的信息 |
| StockDivideDSer.deleteShelf | 语法 | public Result deleteShelf(String id) throws RemoteException |
| 前置条件 | 数据库中含有同样ID的架 |
| 后置条件 | 在数据库中删除相应的架 |
| StockDivideDSer.reviseShelf | 语法 | public Result reviseShelf(ShelfPO shelf) throws RemoteException |
| 前置条件 | 数据库中含有同样ID的架 |
| 后置条件 | 在数据库中修改相应的架 |
| StockWarningDSer.getWarning | 语法 | public ArrayList<Integer> getWarning(String id) throws RemoteException |
| 前置条件 | 逻辑层需要获取当前警戒线比例 |
| 后置条件 | 根据仓库ID返回数据库中航运，铁运，汽运和机动的警戒线比例 |
| StockWarningDSer.setWarning | 语法 | public Result setWarning(int w,Part part,String id) throws RemoteException |
| 前置条件 | 逻辑层设置改变警戒线比例后 |
| 后置条件 | 根据仓库ID更新数据库中航运，铁运，汽运和机动的警戒线比例，返回结果 |
| StockCheckWarnDSer.getWarning | 语法 | public ArrayList<Integer> getWarning(String id) throws RemoteException |
| 前置条件 | 逻辑层需要获取当前警戒线比例 |
| 后置条件 | 根据仓库ID返回数据库中航运，铁运，汽运和机动的警戒线比例 |
| StockCheckWarnDSer.getShelves | 语法 | public ArrayList<ShelfPO> getShelves(String id,String keyword)throws RemoteException |
| 前置条件 | 逻辑层需要获取架的信息 |
| 后置条件 | 返回数据库中所有架的信息 |
| StockCheckWarnDSer.getStock | 语法 | public StockPO getStock(String id)throws RemoteException |
| 前置条件 | 逻辑层需要获取库存情况 |
| 后置条件 | 返回数据库中库存的情况，包括各个架上的货物 |

OrderApproveDSer模块接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| OrderApproveDSer.getExamine | 语法 | public ArrayList<OrderPO> getExamine () throws RemoteException |
| 前置条件 | 逻辑层开启单据审批 |
| 后置条件 | 从数据库获得并返回需要审批单据列表 |
| OrderApproveDSer.update | 语法 | public Result update(Boolean isPassed, ArrayList<String>id, Order kind) throws RemoteException |
| 前置条件 | id存在于数据库中 |
| 后置条件 | 在数据库中更新订单审批状态 |
| OrderApproveDSer.getOrder | 语法 | public PO getOrder(String id,Order kind) throws RemoteException |
| 前置条件 | 逻辑层开启单据审批 |
| 后置条件 | 根据订单种类从数据库获得并返回相应的单据信息 |
| OrderApproveDSer.getUser | 语法 | public UserPO getUser (String userId) throws RemoteException |
| 前置条件 | 逻辑层需要用户信息 |
| 后置条件 | 根据用户id从数据库获得并返回相应的用户信息 |
| OrderResubmitDSer.getOrders | 语法 | public ArrayList<KindGetter> getOrders(String userId) throws RemoteException |
| 前置条件 | 逻辑层需要获得相应的单据列表 |
| 后置条件 | 根据用户ID从数据库获得并返回相应的单据列表 |

CheckDSer模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| DiaryDSer.seeDiary | 语法 | public ArrayList<DiaryPO> seeDiary(Calendar start, Calendar end) throws RemoteException |
| 前置条件 | 逻辑层需要日志列表 |
| 后置条件 | 根据输入的开始和结束日期从数据库中获得日志记录，返回相应的数据 |
| IncomeListDSer.getCollectRec | 语法 | public ArrayList<IncomePO> getCollectRec (Calendar date,String id) throws RemoteException |
| 前置条件 | 逻辑层需要获得收款记录 |
| 后置条件 | 根据日期和营业厅名，从数据库中获取并返回该营业厅的所有收款单记录 |
| BusiCircumDSer.findIncome | 语法 | public ArrayList<IncomePO> findIncome (Calendar start，Calendar end) throws RemoteException |
| 前置条件 | 逻辑层需要获得收款记录列表 |
| 后置条件 | 根据开始结束时段从数据库中获取并返回所有的收款记录 |
| BusiCircumDSer.findPayment | 语法 | public ArrayList<PaymentPO> findPayment (Calendar start，Calendar end) throws RemoteException |
| 前置条件 | 逻辑层需要获得付款记录列表 |
| 后置条件 | 根据开始结束时段从数据库中获取并返回所有的付款记录 |
| CostBeneDSer.getIncomes | 语法 | public ArrayList<IncomePO> getIncomes () throws RemoteException |
| 前置条件 | 逻辑层需要获得收款记录列表 |
| 后置条件 | 从数据库中获取并返回所有的收款记录 |
| CostBeneDSer.getPayments | 语法 | public ArrayList<PaymentPO> getPayments () throws RemoteException |
| 前置条件 | 逻辑层需要获得付款记录列表 |
| 后置条件 | 从数据库中获取并返回所有的付款记录 |