**日照钢厂调度中心分货**

**开发文档**

**京创智汇（上海）物流科技有限公司**

**2020年05月**

修订记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **日期** | **作者** | **版本** | **修改原因** | **主要修改内容** |
| 2020.05.18 | 刘伽椰 | 1.0 | 新建 | 框架设计 |
| 2020.05.19 | 刘伽椰 | 1.1 | 修改 | 全文添加内容 |
| 2020.05.19 | 邵鲁玉 | 1.2 | 修改 | 内容补充 |
| 2020.05.20 | 刘伽椰  廖家俊 | 1.3 | 修改 | 全文可读性修改 |
| 2020.05.20 | 刘伽椰  廖家俊  刘子豪 | 1.4 | 修改 | 框架结构修改；缺省内容补充 |
| 2020.05.21 | 刘伽椰  廖家俊  刘子豪 | 1.5 | 修改 | 修改需求说明与整体框架，补充代码分支说明 |
| 2020.05.22 | 邵鲁玉  刘伽椰 | 1.6 | 修改 | 调整接口参数；  增加功能流程说明图 |

**目录**

[1、概述 4](#_Toc40953698)

[2、需求背景 4](#_Toc40953699)

[2.1业务规则 4](#_Toc40953700)

[3、系统功能 4](#_Toc40953701)

[3.1 系统流程 4](#_Toc40953702)

[3.1.1人工流程 4](#_Toc40953703)

[3.1.2系统流程： 5](#_Toc40953704)

[3.2功能描述 5](#_Toc40953705)

[4、系统设计 6](#_Toc40953706)

[4.1总体架构 6](#_Toc40953707)

[4.2业务模型设计 7](#_Toc40953708)

[4.3业务逻辑设计 11](#_Toc40953709)

[4.4外部接口 16](#_Toc40953710)

[4.5代码管理 17](#_Toc40953711)

[附录 18](#_Toc40953712)

[A工具描述 18](#_Toc40953713)

[B 基类描述 18](#_Toc40953714)

# 1、概述

本分货模块针对钢厂类型，提供分货到车次的功能。根据可发量情况表，结合调度中心和西门开单的需求和业务规则，通过算法程序进行分配优化，生成标载车次和甩货列表。

# 2、需求背景

当前大宗商品产业链中存在物流资源（例如，运输车辆）短缺、仓库库存过多、货物超期严重等现象，这些现象导致了物流的严重阻塞以及物流成本的急剧增长。在物流供需不平衡的场景下，物资如何分配到车队、车辆是平衡物流资源的重要环节。分货策略不仅仅是决定单辆车辆的运载情况的配载策略，而且是对于长期整体物流效率与成本的把控。好的分货策略能够在车载重量、车辆数量限制的情况下，充分利用资源提高发运效率、降低库存。合理的分货策略能够在有限运输资源的条件下降低物流成本、提高服务质量，对整体物流带来经济效益的提升。

## 2.1业务规则

1、优先发运：主要包括客户催货、超期清理，其中客户催货是第一优先级

2、甩货最少：在保证处理优先发运的前提下进行合理搭配，做到同流向甩货最少

3、装卸方便：优先一装一卸，其次是两装一卸、一装两卸，特殊情况会出现两装两卸，属于清理甩货的方式，调度人员与司机达成一致

4、车次载重最大化：为保证车次不亏吨，尽可能配到满载上限

# 3、系统功能

## 3.1 系统流程

### 3.1.1人工流程

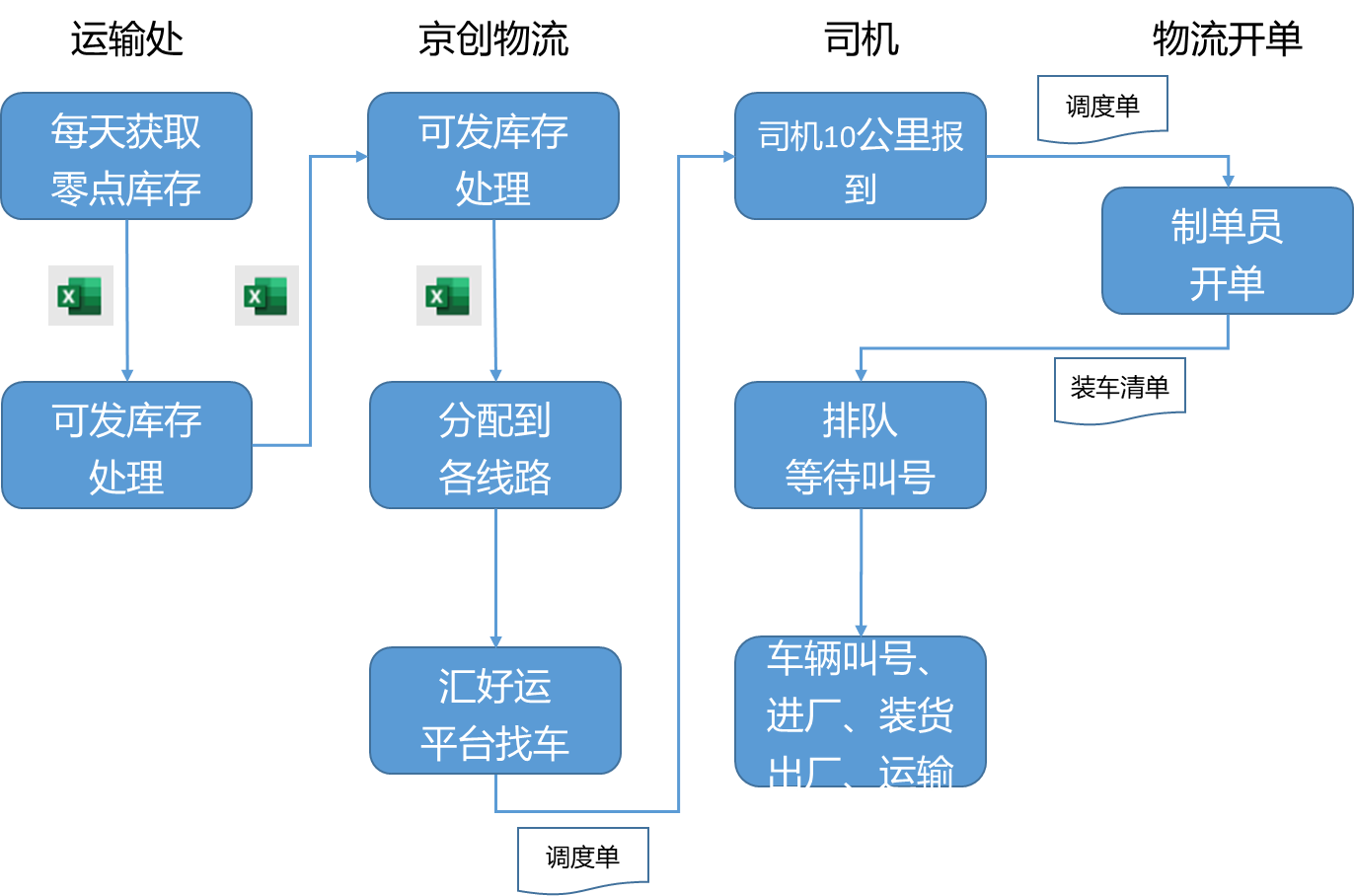


图 1 人工流程图

### 3.1.2系统流程：

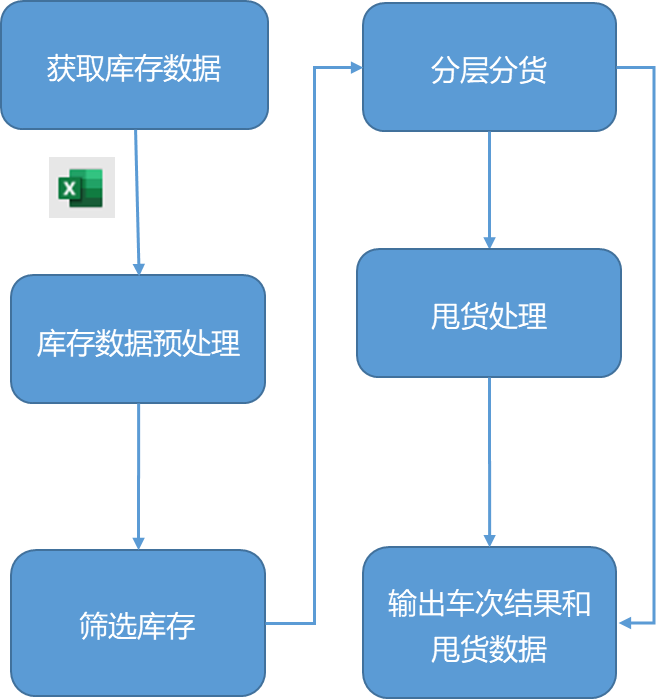


图 2 系统流程图

## 3.2功能描述

1、数据处理：获取可发量情况表，对可发重量和需开单重量进行合并，可发件数和需开单件数进行合并；需短溢重量计算处理；窄带捆包数计算处理；卸货地址标准化；大于标载的库存货物拆分

2、车次配货：将符合标载的数据直接生成车次；将优先发运和最新挂单时间进行正序排序；

四级筛选过滤，将带匹配货物进行线性规划，求解成功生成车次；最终产生不同装卸类型的车次列表和甩货列表。

功能流程图如下说明。



图 3 功能流程图

# 4、系统设计

## 4.1总体架构

该项目的总体架构如下图所示。

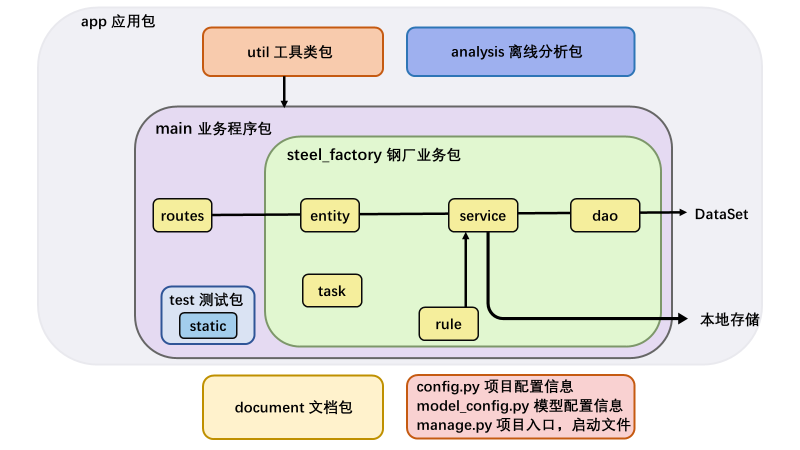


图 3 app应用解析

analysis离线分析包：暂时未使用。后续会包含本地数据分析、特征分析等任务。

util工具类包：包含各种计算工具、动态规划算法、连接池、基类等。

main业务程序包：业务程序包内按不同的厂区业务划分成不同的业务包，如steel\_factory钢厂业务包。其内部结构是一致的，均含有entity业务实体包（存放数据对象类，如订单、车次等）、dao数据连接包（存放数据库访问逻辑）、service业务逻辑包（存放业务处理逻辑，如取库存预处理与分货等）、task定时任务包、rule规则包（存放重量规则、品类规则、拼货规则等，一类规则对应一个文件）、test测试包（存放测试代码。static包指静态数据，比如测试时读入、输出的excel表）、routes路由包（存放对外输入输出接口，负责把外部数据进行相应的提取操作并转化成项目里的对象数据，项目处理逻辑完结后再由该包负责对外输出）。后续的开发如果涉及到不同的厂，业务逻辑大体不变的情况下，编写不同的rule规则即可，别的类一般不需要修改。

document文档包：包含项目的文档资料

Config.py项目配置信息：包含项目运行所需要的程序参数，如数据库连接的端口信息等。

model\_Config.py 模型配置信息：包含模型处理时会用到的参数，比如重量上下限，可合并的品类等参数

manage.py 项目启动入口

## 4.2业务模型设计

该项目中使用的业务实体类如下类图所示。



图 4 业务实体类图

该项目中的数据结构设计与现实业务对象映射关系如下图，所有数据结构都存储在entity业务实体包中。



图 5 类与业务实体映射关系

LoadTask类对应实体业务中的车次，两者为一对一关系，一个LoadTask对象对应一个车次实体。业务实体库存在数据库中以发货通知单为元数据（单位）进行存储，Stock类对应业务实体中的库存，与发货通知单是多对一的关系，一个或多个Stock对象对应一条发货通知单记录。

（1）LoadTask类

继承BaseEntity类（详见附录）

车次类，一个对象表示一个车次，一个车次绑定一辆车，只能存放一个发货通知单中的内容

成员变量

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 含义 | 初始值 | 示例 |
| load\_task\_id | string | 所属车次号 | None | 1 |
| priority | int | 优先级， | 0 | 客户催货 |
| priority\_grade | string | 优先级对应的ABCD等级 | None | ‘A’ |
| load\_task\_type | string | 装卸类型 | None | 一装一卸 |
| total\_weight | float | 总重量，当前车次的货物清单总重量 | 0.0 | 32.5 |
| notice\_num | string | 发货通知单号 | None | F000000001 |
| oritem\_num | string | 订单项次号， | None | DH0000001 |
| weight | float | 发货通知单重量 | 0.0 | 32.5 |
| count | int | 发货通知单件数 | 0 | 2 |
| city | string | 城市 | None | 青岛市 |
| end\_point | string | 区县 | None | 黄岛区 |
| big\_commodity | string | 大品种，货物品种的大分类 | None | 黑卷 |
| commodity | string | 品种，货物品种的小分类 | None | 热轧卷板 |
| standard | string | 规格 | None | 10\*20 |
| sgsign | string | 材质 | None | Q235B |
| outstock\_code | string | 出库仓库，取货的仓库 | None | F2-山东联储中间库 |
| instock\_code | string | 入库仓库，若货物运输的目的地需要其他仓库作为中转站，则存在入库仓库，表示该中转站 | None | ‘-’ |
| receive\_address | string | 收货地址，货物运输目的地的详细地址 | None | 山东省青岛市黄岛区茂山路373号 |
| price\_per\_ton | float | 当前车次的吨单价 | 0.0 | 0.0 |
| total\_price | flaot | 当前车次的总价 | 0.0 | 0.0 |
| latest\_order\_time | string | 最新挂单时间，该发货通知单中的货物生产结束进入仓库的时间 | None | 2020-05-05 |
| remark | string | 注释 | None |  |

（2）Stock库存类

继承BaseEntity类

一个Stock对象对应一个库存子项（发货通知单子项），多个库存子项构成一个完整的库存数据（一条完整发货通知单记录），关系结构如下图所示。



图 6 Stock拆分结构示意图

成员变量

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 含义 | 初始值 | 示例 |
| Stock\_id | string | 库存对象编号 | None | 1 |
| Parent\_stock\_id | string | 父ID | None | 1 |
| Delivery | string | 发货通知单号 | None | F00000001 |
| Order | string | 订单号 | None | DH000001 |
| Priority | string | 发运的优先等级 | None | 客户催货 |
| Consumer | string | 收货用户 | None | 青岛德泰鑫源贸易有限公司 |
| Small\_product\_name | string | 品种，货物品种的小分类 | None | 热轧卷板 |
| Big\_product\_name | string | 大品种，货物品种的大分类 | None | 黑卷 |
| Actual\_weight | int | 实际可发重量 | 0 | 2 |
| Actual\_number | int | 实际可发件数 | 0 | 32 |
| Piece\_weight | int | 件重 | 0 | 16 |
| mark | string | 牌号 | None | Q235B |
| specs | string | 规格 | None | 10\*20 |
| Pack\_form | string | 包装形式 | None |  |
| Warehouse\_out | string | 出库仓库，取货的仓库 | None | F2-山东联储中间库 |
| Warehouse\_in | string | 入库仓库，若货物运输的目的地需要其他仓库作为中转站，则存在入库仓库，表示该中转站 | None | ‘-’ |
| Province | string | 省份，货物目的地的省份 | None | 山东省 |
| City | string | 城市，货物目的地的地级市 | None | 青岛市 |
| Actual\_end\_point | string | 区县（存在部分数据为入库仓库） | None | 城阳区 |
| End\_point | string | 区县 | None | 城阳区 |
| Address | string | 卸货地址，详细地址 | None | 山东省青岛市城阳区流亭街道苇山社区苇山货场 |
| Standard\_address | string | 合并卸货地址 | None | 山东省青岛市城阳区流亭街道苇山社区苇山货场 |
| Delivery\_date | string | 合同约定交货日期 | None | 20200505 |
| Latest\_order\_time | string | 最新挂单时间，该发货通知单中的货物生产结束进入仓库的时间 | None | 20200505 |
| Logistics | string | 物流公司类型 | None | 省内2 |

## 4.3业务逻辑设计

业务工程实现的逻辑结构如下图所示，所有工程逻辑都存储在service业务逻辑包和rules钢厂分货规则包中，service包中包含stock\_service、steel\_dispatch\_service、generate\_excel\_service三个模块，rules钢厂分货规则包中包含dispatch\_filter和pulp\_solve两个模块。

程序主流程中，首先调用steel\_dispatch\_service.dispatch()方法，得到分货后的车次结果，再调用generate\_excel\_service.generate\_excel()方法将结果写进本地的excel文件中。



图 7 工程实现逻辑结构

rules钢厂分货规则包中包含dispatch\_filter和pulp\_solve两个模块以实现拼货逻辑，工程实现的逻辑结构如下图所示。



图 8 拼货实现逻辑结构

以下给出详细描述。

（1）stock\_service模块

库存获取与预处理服务

* get\_stock()方法

获取库存列表，通过读取静态的excel文件《sheet1.xls》（文件在test.static目录下）获取库存数据。

* address\_latitude\_and\_longitude()方法

获取数据表中地址经纬度信息，使用查询sql（string类型存放sql语句）从db\_sys数据库模式中获取经纬度。

* deal\_stock()方法

输入:

stock\_list 以Stock对象为元数据的列表

功能与流程：

1）调用get\_stock()方法,获取库存列表（一条发货通知单记录对应一个Stock对象）

2）对获取的库存数据进行数据预处理，其中包括：

1.将数据库中一个发货通知单中的不同重量内容按照规则进行合并，得到实际可发的重量与件数。过滤掉实际可发重量/件数小于等于0和最新挂单时间为空（及当前没有货物可分）的数据。

2.数据库中需要合并的字段：可发重量、需开单重量、需短溢重量合并为实际可发重量；可发件数、需开单件数、需短溢重量合并为实际可发件数，规则如下：

* 可发重量/件数和需开单重量/件数，进行数据量级上的统一（统一重量单位）后直接相加
* 当需短溢重量大于0时，实际可发重量直接减去需短溢重量，根据实际可发重量和单件重量来计算件数的约数作为实际可发件数。
* 特殊品种的件数和重量处理——品种为窄带的发货通知单数据，将窄带的捆包数改为可发件数；

3.品种为开平板的发货通知单数据，将根据不同的存储仓库（“出库仓库”字段）修改品种名称以区分，达到新老区开平板不拼货的效果。

4.卸货地址标准化，由于存在差别在几个字左右的不同地址，但其实是一个地址的情况

3）拆分库存数据记录，以33吨为标准，将件数较多的库存（Stock对象）拆分成多个，当一条发货通知单总重量不达到33吨或单件货物就超过33吨时不进行拆分。拆分后，一个Stock对象只能存储一个发货通知单子项，但多个Stock对象才对应一个完整的发货通知单。最后输出拆分后的数据库记录，数据结构为：以Stock对象为元数据的列表。

* rename\_pd()方法

更改数据列名，统一格式

（2）steel\_dispatch\_service模块

钢厂分货服务

* dispatch()方法

输入：stock\_list:：以Stock对象为元数据的列表

功能与流程：

1）获取库存数据（发货通知单）并进行处理，调用stock\_service.deal\_stock()方法

2）将库存数据区分处理。

1.提取重量在31以上的记录直接进行一对一的车次封装；

2.对所有小于31吨的库存记录进行多次拼车判断:

* 是否能增加到1）中的车次中，同时保证不超重
* 根据分货筛选规则（调用dispatch\_filter.dispatch()方法，详见4.2.3算法描述）进行拼车判断

3）无法分成完整车次（无法拼车达到重量下限）的货物合并在车次号为-1的车次中；

4）返回分货后的车次结果列表，数据结构为以LoadTask对象为元数据的列表。

（3）generate\_excel\_service模块

车次结果数据处理、写本地文件服务

* generate\_excel()方法

输入：

load task list 以车次对象为元数据的列表

功能与流程：

1）对原始的分货结果做一些简单的数据分析与合并处理，对分货车次进行汇总计算，统计区县的车次数量和配载拼品种情况。结果如下（部分示例），其中混装配载建议为分货结果中出现的混装配载（品种拼车）情况的统计。



图 4 汇总结果示意图

2）将原始的分货结果写进excel中，示例入下，存储在“分货结果.xlsx”文件中，分货结果存储在工作簿“分货车次明细”中，分货结果的统计分析（步骤1）存储在工作簿“分货车次汇总”中。



图 5 分货结果文件示意图

* rename\_load\_result()方法

分货结果字段重命名，统一格式

* rename\_collet\_result()方法

分货汇总结果重命名，统一格式

* drop\_duplicate\_list()方法

二位数组内部去除重复数组

* concat\_string()方法

拼接数组内品种字符

（4）dispatch\_filter模块

* dispatch()方法

分货筛选包含四层筛选，第一层一装一卸筛选器，第二层两装一卸（同区仓库）筛选器，第三层两装一卸（非同区仓库）筛选器，第四层一装两卸筛选器。

第一层：调用first\_deal\_general\_stock()方法

遍历货物列表，对每件货物进行同仓库同卸货地址的筛选，使用背包过滤方法对筛选后的货物列表进行二次计算（调用goods\_filter方法），获得最终可拼一车的所有货物列表后打包为一个车次（调用calculate方法）

第二层：调用second\_deal\_general\_stock()

遍历货物列表，对每件货物进行可拼同区仓库（对所有可拼情况进行遍历）和同卸货地址的筛选，使用背包过滤方法对筛选后的货物列表进行二次计算（调用goods\_filter方法），获得最终可拼一车的所有货物列表后打包为一个车次（调用calculate方法）

第三层：调用third\_deal\_general\_stock()

遍历货物列表，对每件货物进行可拼非同区仓库（对所有可拼情况进行遍历）同卸货地址的筛选，使用背包过滤方法对筛选后的货物列表进行二次计算（调用goods\_filter方法），获得最终可拼一车的所有货物列表后打包为一个车次（调用calculate方法）

第四层：调用fourth\_deal\_general\_stock()

遍历货物列表，对每件货物进行同仓库可拼卸货地址（对所有可拼情况进行遍历）的筛选，使用背包过滤方法对筛选后的货物列表进行二次计算（调用goods\_filter方法），获得最终可拼一车的所有货物列表后打包为一个车次（调用calculate方法）

* goods\_filter()方法

背包过滤方法

将库存数据转换为调用背包算法需要的简单参数结构——（货物重量列表，货物价值==货物重量，装载剩余上限），调用pulp\_solve.pulp\_pack()方法计算得到最终选取货物的方案

* calculate()方法

将一组货物数据（以Stock对象为元数据的列表）打包为一个车次数据（LoadTask对象）。

（5）pulp\_solve模块

* pulp\_pack()方法

调用pulp包解决最优化问题，使用线性规划的方法求解最优解，优化目标与限制条件如下：

其中w表示单件货物的重量，x表示该货物的选择情况，，c为阈值表示装载上限。优化目标为最大化选择的货物总重量，限制条件为选择的货物总重量小于阈值。

因此方法结果求解，得到实数列x。最终需要得到货物是否被选择（0表示不选择，1表示选择）的结果列表，因此程序使用0.5为阈值将x进行01离散化处理。

## 4.4外部接口

（1）测试接口链接:[http://192.168.1.27:9238/order](http://192.168.1.44/#/system/goods-allocation/goods-allocation/billOfManage)

（2）功能说明：根据输入的可发量情况表，按照分货配载规则分发到车次，产生预装车清单列表

（3）接口说明：

|  |  |  |  |
| --- | --- | --- | --- |
| **请求协议** | **请求方式** | **编码格式** | **返回格式** |
| HTTP | POST | UTF-8 | JSON |

（4）接口参数：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **参数名字** | | **类型** | **是否必须** | **默认值** | **说明** |
| type | | int | 是 | 无 | 业务类型，可选值：0，1，2  当type=1时，执行钢厂智能分货 |
| data | | list | 是 | 无 | 订单号 |
|  | notice\_num | string | 是 | 无 | 发货通知单号 |
|  | oritem\_num | string | 是 | 无 | 订单号 |
|  | devperiod | string | 是 | 无 | 合同交期 |
|  | deliware\_house | string | 是 | 无 | 出库仓库 |
|  | trans\_way | string | 是 | 无 | 运输方式 |
|  | commodity\_name | string | 是 | 无 | 品名 |
|  | big\_commodity\_name | string | 是 | 无 | 大品名 |
|  | priority | string | 是 | 无 | 优先发运：客户催货或超期清理 |
|  | pack | string | 是 | 无 | 包装方式 |
|  | quality | string | 是 | 无 | 品质 |
|  | material | string | 是 | 无 | 材质 |
|  | standard | string | 是 | 无 | 销售规格 |
|  | deliware | string | 是 | 无 | 入库仓库 |
|  | portnum | string | 是 | 无 | 港口编号 |
|  | detail\_address | string | 是 | 无 | 详细地址 |
|  | province | string | 是 | 无 | 省 |
|  | city | string | 是 | 无 | 市 |
|  | waint\_fordel\_number | int | 是 | 无 | 待发件数 |
|  | waint\_fordel\_weight | float | 是 | 无 | 待发重量 |
|  | can\_send\_number | int | 是 | 无 | 可发件数 |
|  | can\_send\_weight | float | 是 | 无 | 可发重量 |
|  | dlv\_spot\_name\_end | string | 是 | 无 | 终点 |
|  | deliware | string | 是 | 无 | 入库仓库 |
|  | pack\_number | int | 是 | 无 | 窄带捆包数 |
|  | need\_lading\_num | int | 是 | 无 | 需开单件数 |
|  | need\_lading\_wt | float | 是 | 无 | 需开单重量 |
|  | over\_flow\_wt | float | 是 | 无 | 需短溢重量 |
|  | logistics\_company\_type | string | 是 | 无 | 物流公司类型 |
|  | latest\_order\_time | string | 是 | 无 | 最新挂单时间 |
|  | port\_name\_end | string | 是 | 无 | 到货码头 |
|  | bill\_plan\_wt | float | 是 | 无 | 发货通知单计划重量 |
|  | order\_wt | float | 是 | 无 | 订单重量 |
|  | contract\_not\_deli\_wt | float | 是 | 无 | 合同未发总重量 |

（5）返回结果

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名字** | | **类型** | **说明** |
| code | | int | 状态码 |
| msg | | string | 返回提示信息 |
| data | | list | 车次信息列表 |
|  | load\_task\_id | string | 车次号 |
| priority | string | 优先发运 |
| priority\_grade | string | 优先级对应的ABC等级 |
| load\_task\_type | string | 装卸类型:一装一卸，两装一卸（同区仓库），  两装一卸（非同区仓库），一装两卸，两装两卸，甩货 |
| total\_weight | float | 总重量，当前车次的货物清单总重量 |
| notice\_num | string | 发货通知单号 |
| oritem\_num | string | 订单项次号 |
| weight | float | 发货通知单重量 |
| count | int | 发货通知单件数 |
| city | string | 城市 |
| end\_point | string | 终点 |
| big\_commodity | string | 大品种，货物品种的大分类 |
| commodity | string | 品种，货物品种的小分类 |
| standard | string | 规格 |
| sgsign | string | 材质 |
| outstock\_code | string | 出库仓库 |
| instock\_code | string | 入库仓库 |
| receive\_address | string | 收货地址 |
| price\_per\_ton | float | 当前车次的吨单价 |
| total\_price | flaot | 当前车次的总价 |
| latest\_order\_time | string | 最新挂单时间 |
| remark | string | 注释 |

## 

## 4.5代码管理

一、项目Git地址：<https://git.jczh56.com/models/gc-goods-allocation>

二、分支说明:

1.dev\_1.0为项目主发布分支，在该分支上构建开发分支

2.dev\_steel为钢厂开发分支，

3.dev\_pipe为管厂开发分支

三、开发分支管理规范

1.开发新功能时选择钢厂或管厂分支拉到本地进行开发；

2.在开发分支进行功能开发和测试，同步提交相关文档；

3.测试完成后准备发布时将代码合并到dev\_1.0并打上对应的版本tag；

# 附录

## A工具描述

util包db\_pool.py数据库连接池

connection（pymsql中的数据库连接对象）统一存放在该工具模块中

## B 基类描述

BaseEntity类

位于app.main.entity.base\_entity.py，实体类的基类，子类继承可调用或重写以下通用方法，当前工程中类继承关系如下图。



包含方法as\_dict、get\_attr\_str、set\_attr

as\_dict将对象（成员变量）转换为dict数据类型返回

get\_attr\_str读取类成员变量，将其统一转换为string数据类型

set\_attr根据传入的对象设置成员变量，将传入的dict数据类型转换为成员变量