1. Gflux系统设计文档

北京网感至察科技有限公司

2015年03月09日

* 1. 1.基本介绍
     + - 1. 1.1文档目的

本文档的目的为说明Gflux系统及WEB服务的实现原理与主要接口。

* + - * 1. 1.2相关文档
* A1Gflux系统部署文档
  1. 2.总体设计
     + - 1. 2.1运行环境

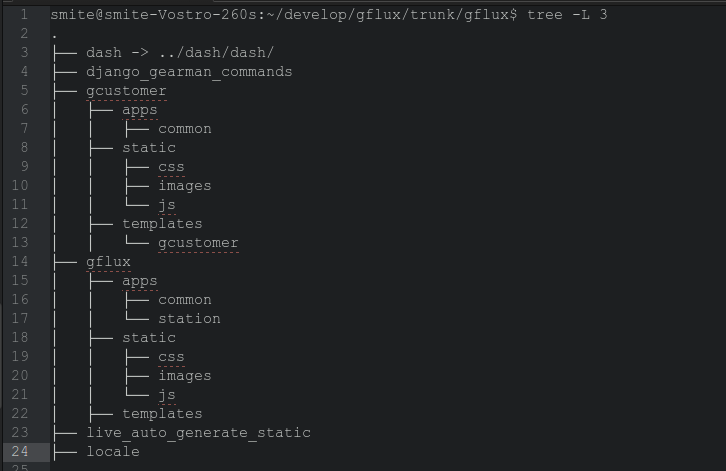
本系统使用Django, jQuery,HTML5及Python语言开发而成；后端数据库为PostgreSQL。

Django是一个基于MVC的Web开发框架。Django中将用户输入以模型（Model）、模板(Template)和视图（Views）建模，其中模型处理与数据存储相关的事务；模板处理与表现相关的决定，即如何在页面中进行显示；视图存取模型及调取恰当模板的相关逻辑。

本系统服务器端在Ubuntu 12.0.4上测试通过；本系统浏览器客户端在Chrome 31.0上测试通过。

* + - * 1. 2.2程序基本结构

程序目录基本结构如下：



其中编程定制开发的主要目录文件为：

1. dash ，gcustomer，gflux/apps

存有后端实现的程序文件

1. gflux/static, gcustomer/static

存有前端实现的脚本文件

1. gflux/templates,gcustomer/templates

存有前端实现的模板文件

1. gflux/apps/common/management/commands,gflux/apps/station/management/commands

存有后端数据库的初始化脚本,gearman worker 脚本

除此之外，live\_auto\_generate\_static 是django自动聚合的所有静态文件目录， locale 是django多语言的文件夹。

* + - * 1. 2.3数据对象与存储

在上述后端脚本中包含多个models.py文件，定义有系统所需要的所有数据对象，以及它们在数据库中对应的表格。

数据对象包括：

|  |  |  |
| --- | --- | --- |
| 名称 | 介绍 | 对应表格 |
| DimChinaProvinceCityDistrict | 中国行政规划 | dim\_chinaprovincecitydistrict |
| SiteDayBatch | 油站逻辑天定义 | sitedaybatch |
| DimDateHour | 时间年月日定义 | dim\_datehour |
| Station | 站点 | station |
| FuelType | 汽油类型 | fuel\_type |
| FuelTypeRelation | 站级油品与系统油品关系 | Fuel\_type\_relation |
| Location | 加油站大区 | Location |
| StationDailyStat | 油站每天的状态量 | station\_daily\_stat |
| StationMonthStat | 油站每月的状态量 | station\_month\_stat |
| StationDailyFuelSales | 油站每天油品状态量 | station\_daily\_fuel\_sales |
| Trans | 交易流水 | fact\_trans |
| Card | 客户忠诚度 | card |
| Item | 商品表 | Item |
| StationItemAssoc | 站点商品之间的相关性 | station\_item\_assoc |
| ItemAssoc | 所有商品之间的相关性 | item\_assoc |
| User | 网站用户 | station\_user |
| File | 用户上传的文件 | station\_file |
| UserStation | 用户与油站关系 | station\_userstatino |
| StatinoNoneFuelTop | 油站非油品排行 | station\_stationnonefueltop |
| StationFuelType | 油站油品 | statino\_stationfueltype |
| TargetAudience | 用户关注的人群 | gcustomer\_target\_audience |
| SiteProfile | 站点画像 | gcustomer\_site\_profile |
| CustomerGroup | 站点聚合 | gcustomer\_group |
| CustomerProfile | 人物画像 | gcustomer\_user\_profile |
| DefaultPromotionArgs | 优惠活动模板 | gcustomer\_default\_promotion\_args |
| Promotion | 优惠活动 | gcustomer\_promotion |
| PromotionResponse | 用户参加优惠活动 | gcustomer\_promotion\_response |
| ShardStat | 分片状态 | shard\_stat |
| SiteShard | 站点分片状态 | site\_shard |

* 1. 3. 算法设计
     + - 1. 3.1 数据采集

大规模数据上传可以采取以下两种方式进行：

客户主动上传

1.大客户将站级服务器的数据库备份文件，通常目录名为Office，其中有GDB文件，将其打包以其它媒介上传到我方服务器上，由我方负责导出数据然后导入数据库。

2.客户将按要求准备好的文件，通过web上传，直接导入数据库。

客户被动上传（未实现）

在站级服务器上安装软件，定期实时进行数据上传，相关代码在win32/installer，具体过程为：

1.安装marketcloud软件

2.安装过程中配置odbc，远程内网服务器，计划任务设置

3.计划运行时会从内网服务器上下载配置文件

4.脚本根据配置文件动态导出数据库并上传到内网mysql中

为实现此目的，需要搭建一台高性能的数据库服务器，站级服务器上的数据导入软件会采取如下方式进行数据上传：

1.定期连接远程公网服务器下载配置文件

2.定期导出mysql中数据并上传到网站数据库中

* + - * 1. 3.2分布式存储

各个油站的数据采取分布式的方式进行存储，每个加油站的数据被存储在一台服务器上。

决定每个站点属于哪台服务器的具体流程如下:

* + - 1. 从shard\_stat表格中选取出压力最小的服务器（存储的站点数最少）
      2. 将站点与服务器映射结果存储在site\_shard表格中
         1. 3.3报表生成

图表基本上都基于对fact\_trans表格的查询来进行，主要相关的目录和文件以及其基本功能包括:

1. dash/core/backends/sql/cubes.py

Cube类根据参数生成sqlalchemy的select expression进行数据库查询，并简单组织了一下返回结果。

2. dash/core/backends/sql/models.py

提供了分布式存储支持，能根据站名取得对应的数据库服务器连接

3. dash/core/reports.py

Report类提供了获取所有图表数据的基础封装，包含了请求路由，查询数据库，返回结果的一整个流程。

以站点最近一个月每天的加油量趋势为例，报表结果json数据的生成过程如下：

1. 根据Report类中urlpattern方法提供的路由，请求将直接调用as\_view函数

2. as\_view函数调用get函数返回Http Response

3. get函数调用handle函数进行计算

4. handle函数过程如下：

1. 验证用户权限

2. 验证请求参数是否合法

3. 调用子类的report函数进行计算并返回结果

* + - * 1. 3.4报表渲染

报表的数据查询是在dash中的report.py文件中将报表的结果数据查询统计出来，已经在上面一小节进行了介绍。报表渲染的基本介绍如下：

1. dash/core/ui.py封装了html页面所有组件的父类
2. ui\_portal是最顶层的，接下来依次是ui\_page,ui\_dash,ui\_dashlet
3. 其中除了ui\_dashlet没有外部路由，其他组件都有其对应的路由，也即可以通过url进行访问

以站点最近一个月每天的加油量趋势为例，从HTTP请求到HTTP回答的渲染过程如下：

1. 底层UIView类根据路由调用as\_view函数
2. as\_view函数调用子类的render函数
3. render函数首先进行权限检查，然后准备包含的模块，调用django的template进行渲染
4. 浏览器取得页面后开始执行ajax请求取得报表的json数据
5. 调用js的框架渲染报表
   * + 1. 3.5 后端服务

为了将计算量较大或比较耗时的函数从web server中分离出来，我们将如下的计算剥离出来，通过gearman采取分布式的方式在其它机器上进行计算，包括如下服务：

1. 数据导入服务, gearman\_worker\_import\_data.py

1. 解析文件，将流水存储到数据库

2. 更新站点信息，站点描述，油枪数，油枪号等

3. 计算所有商品之间的相关性

4. 按天统计油品的销售数据

1. 邮件发送服务，gearman\_worker\_send\_email.py

1. 连接settings.py中提供的邮件服务器

2. 登陆邮件服务器

3. 发送邮件

3. compute\_item\_assoc,计算指定时段商品相关性

1.按公式以某个概率采样数据

2.使用科学计算库计算相关性

3.抽取计算结果并更新数据库

4. deal\_with\_hb\_shihua\_data,预处理广阳数据

5. deal\_with\_senmei\_data,预处理森美数据

6. gearman\_worker\_compute\_fuel\_daybatch,计算每天油品汇总数据

1.处理数据的起止日期

2.计算每天油品汇总量,根据sha1存入或更新到StationDailyFuelSales

7. gearman\_worker\_compute\_station\_daybatch,计算每天油站汇总数据

1.处理数据的起止日期

2.计算每天油站汇总数据,根据sha1存入或更新到StationDailyStat

8. gearman\_worker\_import\_senmei\_data,导入森美数据

1. 根据deal\_with\_senmei\_data产生的预处理文件，将流水存储到数据库

2. 更新站点信息，站点描述，油枪数，油枪号等

3. 计算所有商品之间的相关性

4. 按天统计油品的销售数据

9. gearman\_worker\_import\_sitedaybatch\_data,导入站点逻辑天

10. gearman\_worker\_import\_sp\_excel\_data,导入sp数据

1. 解析文件，将流水存储到数据库

2. 更新站点信息，站点描述，油枪数，油枪号等

3. 计算所有商品之间的相关性

4. 按天统计油品的销售数据

11. gearman\_worker\_import\_ycshell\_excel\_data,导入延长壳牌数据

1. 解析文件，将流水存储到数据库

2. 更新站点信息，站点描述，油枪数，油枪号等

3. 计算所有商品之间的相关性

4. 按天统计油品的销售数据

12. gearman\_worker\_update\_vip,更新支付类型脚本

13. import\_csv\_data\_card\_all,导入card和all分开的CSV格式的数据

1. 解析文件，将流水存储到数据库

这个是以前文件的导入脚本,也需要加上gearman\_worker\_import\_data.py的2~4

14. init\_fuel\_type\_relation,初始化油品类型

15. compute\_gilbarco\_index计算网感指数脚本,与 precompute\_all\_gilbarco\_indexes配套

* 1. 4. 详细设计
     + - 1. 4.1 所有的图表请求

|  |  |
| --- | --- |
| 涉及数据对象 | Trans |
| 前端请求路径 | dash库根据后端report类名自动生成唯一路径 |
| 前端请求参数 | fuel\_type=0&site=BJBJBJ&date=2013-04-08&payment\_type=0 |
| 后端实现函数 | 每一个自定义的report类 |
| 后端返回结果示例 | {"status": "OK",  "extra": [  {"center": [50, 0],  "data": [  {"y": 69.4,  "name": "92\u53f7 \u8f66\u7528\u6c7d\u6cb9(\u4eac\u2164)"},  {"y": 7.11,  "name": "95\u53f7 \u8f66\u7528\u6c7d\u6cb9(\u4eac\u2164)"},  {"y": 23.49,  "name": "0\u53f7 \u8f66\u7528\u67f4\u6cb9(\u4eac\u2164)"}],  "name": "\u6bd4\u4f8b",  "dataLabels": {"distance": 5, "style": {"color": "black", "fontWeight": "bold"},  "enabled": true,  "format": "{point.y:.1f}%"},  "type": "pie", "  showInLegend": false,  "size": 100}],  "categories": ["0 - 1", "1 - 2", "2 - 3", "3 - 4", "4 - 5", "5 - 6", "6 - 7", "7 - 8", "8 - 9", "9 - 10", "10 - 11", "11 - 12", "12 - 13", "13 - 14", "14 - 15", "15 - 16", "16 - 17", "17 - 18", "18 - 19", "19 - 20", "20 - 21", "21 - 22", "22 - 23", "23 - 24"],  "dataset": [  {"stacking": "normal", "type": "area", "data": [0, 0, 0, 0, 0, 0, 0, 1344.85, 1779.77, 1353.69, 1505.02, 675.52, 100.0, 578.97, 50.0, 161.38, 726.6, 396.0, 0, 0, 0, 0, 0, 0], "name": "92\u53f7 \u8f66\u7528\u6c7d\u6cb9(\u4eac\u2164)"}, {"stacking": "normal", "type": "area", "data": [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 300.0, 545.0, 0, 0, 0, 43.05, 0, 0, 0, 0, 0, 0], "name": "95\u53f7 \u8f66\u7528\u6c7d\u6cb9(\u4eac\u2164)"}, {"stacking": "normal", "type": "area", "data": [0, 0, 0, 0, 0, 0, 0, 0, 0, 70.0, 428.0, 2048.13, 0, 0, 389.7, 0, 0, 0, 0, 0, 0, 0, 0, 0], "name": "0\u53f7 \u8f66\u7528\u67f4\u6cb9(\u4eac\u2164)"}]} |
| 实现流程说明 | 使用python元编程扫描对应子类的成员变量情况，自动绑定路由，继承父类自动生成查询语句，自动生成返回结果 |

* + - * 1. 4.2 更改密码

|  |  |
| --- | --- |
| 涉及数据对象 | User |
| 前端请求路径 | ajax/update\_password/ |
| 前端请求参数示例 | 1. old\_password=123abc&new\_password=123123 |
| 后端实现函数 | UpdatePassword |
| 后端返回结果示例 | {"info": "password error", "ret": "0003"} |
| 实现流程说明 | 判断权限，判断新密码格式，更改后返回 |

* + - * 1. 4.3 取得用户站点

|  |  |
| --- | --- |
| 涉及数据对象 | UserStation |
| 前端请求路径 | ajax/get\_user\_station/ |
| 前端请求参数示例 | 1. user\_id=1 |
| 后端实现函数 | GetUserStation |
| 后端返回结果示例 | {"info": "Everything is OK", "data": [  {"name": "BJBJBJ", "desc": "\u4f60\u597dsb123"}, {"name": "BJBJT", "desc": "\u4f60\u597d2"}, {"name": "BJ\_CY", "desc": "CY"},  {"name": "BJ\_HD\_SDDD", "desc": "\u4f60\u54c8"}, {"name": "BJ\_XC", "desc": "XC"},  {"name": "CN\_BJ\_HD", "desc": "HD"},  {"name": "CN\_TEST", "desc": "TEST"},  {"name": "CN\_TTT", "desc": "TTT"},  {"name": "CQCQ", "desc": "\u91cd\u5e86"}],  "ret": "1101"} |
| 实现流程说明 | 判断权限，查询数据库返回 |

* + - * 1. 4.4 删除用户站点

|  |  |
| --- | --- |
| 涉及数据对象 | UserStation |
| 前端请求路径 | Ajax/remove\_user\_site/ |
| 前端请求参数示例 | 1. user\_id=1&site\_name=BJBJBJ |
| 后端实现函数 | RemoveUserStation |
| 后端返回结果示例 | {"info": "Everything is OK", "ret": "1101"} |
| 实现流程说明 | 判断权限，删除后返回 |

* + - * 1. 4.5 添加用户站点

|  |  |
| --- | --- |
| 涉及数据对象 | UserStatino |
| 前端请求路径 | Ajax/add\_user\_site/ |
| 前端请求参数示例 | 1. user\_id=1&site\_name=BJBJBJ |
| 后端实现函数 | AddUserStation |
| 后端返回结果示例 | {"info": "Everything is OK", "ret": "1101"} |
| 实现流程说明 | 判断权限，增加后返回 |

* + - * 1. 4.6 取得油站的油品列表

|  |  |
| --- | --- |
| 涉及数据对象 | FuelTypeRelation |
| 前端请求路径 | ajax/get\_station\_fuel\_type/ |
| 前端请求参数示例 | 1. site=BJBJBJ |
| 后端实现函数 | GetStationFuelType |
| 后端返回结果示例 | {"info": "Everything is OK", "fuel\_types": [{"barcode": 300586, "name": "95\u53f7 \u8f66\u7528\u6c7d\u6cb9(\u4eac\u2164)"}, {"barcode": 300603, "name": "0\u53f7 \u8f66\u7528\u67f4\u6cb9(\u4eac\u2164)"}, {"barcode": 300585, "name": "92\u53f7 \u8f66\u7528\u6c7d\u6cb9(\u4eac\u2164)"}], "ret": "1101"} |
| 实现流程说明 | 判断权限，查询数据库返回数据 |

* + - * 1. 4.7 高级报表查询

|  |  |
| --- | --- |
| 涉及数据对象 | Trans |
| 前端请求路径 | ajax/professional\_analysis/ |
| 前端请求参数示例 | 1. site=BJBJT&count\_type=L&time\_type=ONE&date=2013-04-08&start\_date=2013-03-20&end\_date=2013-04-09&series\_category%5B%5D=fuel-type&base\_time=hour |
| 后端实现函数 | ProfessionalAnalysis |
| 后端返回结果示例 | {"info": "Everything is OK", "categories": ["00:00 - 01:00", "01:00 - 02:00", "02:00 - 03:00", "03:00 - 04:00", "04:00 - 05:00", "05:00 - 06:00", "06:00 - 07:00", "07:00 - 08:00", "08:00 - 09:00", "09:00 - 10:00", "10:00 - 11:00", "11:00 - 12:00", "12:00 - 13:00", "13:00 - 14:00", "14:00 - 15:00", "15:00 - 16:00", "16:00 - 17:00", "17:00 - 18:00", "18:00 - 19:00", "19:00 - 20:00", "20:00 - 21:00", "21:00 - 22:00", "22:00 - 23:00", "23:00 - 24:00"], "ret": "1101", "dataset": [{"data": [0, 0, 0, 0, 0, 0, 0, 0, 0, 9.01, 55.08, 276.08, 0, 0, 51.48, 0, 0, 0, 0, 0, 0, 0, 0, 0], "name": " \u6cb9\u54c1\u7c7b\u578b:0\u53f7 \u8f66\u7528\u67f4\u6cb9(\u4eac\u2164)"}, {"data": [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 36.1, 66.79, 0, 0, 0, 5.18, 0, 0, 0, 0, 0, 0], "name": " \u6cb9\u54c1\u7c7b\u578b:95\u53f7 \u8f66\u7528\u6c7d\u6cb9(\u4eac\u2164)"}, {"data": [0, 0, 0, 0, 0, 0, 0, 176.29, 230.14, 176.28, 195.78, 89.74, 12.82, 75.0, 6.41, 20.69, 95.4, 51.28, 0, 0, 0, 0, 0, 0], "name": " \u6cb9\u54c1\u7c7b\u578b:92\u53f7 \u8f66\u7528\u6c7d\u6cb9(\u4eac\u2164)"}]} |
| 实现流程说明 | 检查权限，查询数据库，返回结果 |

* + - * 1. 4.8 登陆

|  |  |
| --- | --- |
| 涉及数据对象 | User |
| 前端请求路径 | Ajax/check\_login/ |
| 前端请求参数示例 | username=smite&password=chow |
| 后端实现函数 | checkLoginAjaxRequest |
| 后端返回结果示例 | {"info": "Everything is OK", "ret": "1101"} |
| 实现流程说明 | 判断用户名密码是否正确，返回结果 |

* + - * 1. 4.9 用户注册

|  |  |
| --- | --- |
| 涉及数据对象 | User |
| 前端请求路径 | Ajax/check\_register/ |
| 前端请求参数示例 | 1. username=aaa&password=aaaaaa&email=aaa%40aa.aa&company=aaa&district=110101 |
| 后端实现函数 | checkRegisterAjaxRequest |
| 后端返回结果示例 | {"info": "Everything is OK", "ret": "1101"} |
| 实现流程说明 | 检查是否注册，保存数据返回 |

* + - * 1. 4.10 取得用户列表

|  |  |
| --- | --- |
| 涉及数据对象 | User |
| 前端请求路径 | Ajax/check\_users/ |
| 前端请求参数示例 | 1. type=-1 |
| 后端实现函数 | checkUsersAjaxRequest |
| 后端返回结果示例 | {"info": "Everything is OK", "data": [{"company": "test2", "type": 4, "name": "test2", "time": "2014-12-12"}, {"company": "dfa", "type": 0, "name": "test", "time": "2015-2-4"}, {"company": null, "type": 4, "name": "tao", "time": "2014-11-22"}, {"company": "aaa", "type": 0, "name": "aaa", "time": "2015-3-9"}], "ret": "1101"} |
| 实现流程说明 | 检查权限，查询数据库，返回结果 |

* + - * 1. 4.11 上传文件

|  |  |
| --- | --- |
| 涉及数据对象 | File |
| 前端请求路径 | ajax/upload\_file/ |
| 前端请求参数示例 | 1. qqfile=card1000.txt |
| 后端实现函数 | uploadFileAjaxRequest |
| 后端返回结果示例 | {"info": "Everything is OK", "ret": "1101"} |
| 实现流程说明 | 检查用户是否达到上传限制，保存文件，返回结果 |

* + - * 1. 4.12 已上传文件列表

|  |  |
| --- | --- |
| 涉及数据对象 | File |
| 前端请求路径 | ajax/check\_uploaded\_files/  HTTP请求路径为/yunji/ajax/import\_data\_spots/ |
| 前端请求参数示例 | 无 |
| 后端实现函数 | checkFilesAjaxRequest |
| 后端返回结果示例 | {"info": "Everything is OK", "data": [], "ret": "1101"} |
| 实现流程说明 | 判断权限，返回结果 |

* + - * 1. 4.13 获取站点最新最旧数据时间

|  |  |
| --- | --- |
| 涉及数据对象 | Station |
| 前端请求路径 | Ajax/get\_station\_latest\_earliest\_date/ |
| 前端请求参数示例 | site=B |
| 后端实现函数 | GetStationLatestEarliestDate |
| 后端返回结果示例 | {"info": "Everything is OK",  "latest\_date": "2013-04-09",  "latest\_year\_date": "2012-04-08",  "ret": "1101",  "latest\_month\_date": "2013-03-09", "latest\_quarter\_date": "2013-01-09", "lastest\_second\_date": "2013-04-08",  "earliest\_date": "2013-03-20"} |
| 实现流程说明 | 判断权限，返回数据 |

* + - * 1. 4.14 修改用户类型

|  |  |
| --- | --- |
| 涉及数据对象 | User |
| 前端请求路径 | Ajax/update\_user\_type/ |
| 前端请求参数示例 | 1. user\_name=test2&user\_type=0 |
| 后端实现函数 | UpdateUserType |
| 后端返回结果示例 | {"info": "success", "ret": "ok"} |
| 实现流程说明 | 判断权限，更新数据库，返回 |

* + - * 1. 4.15 登出

|  |  |
| --- | --- |
| 涉及数据对象 | 无 |
| 前端请求路径 | ajax/logout/ |
| 前端请求参数示例 | 无 |
| 后端实现函数 | LogoutRequest |
| 后端返回结果示例 | {"info": "Everything is OK", "ret": "1101"} |
| 实现流程说明 | 判断权限，清空session，返回 |

* + - * 1. 4.16 导入数据

|  |  |
| --- | --- |
| 涉及数据对象 | Trans |
| 前端请求路径 | ajax/importData/ |
| 前端请求参数示例 | 1. type=1&site=BJBJT&site\_desc=%E4%BD%A0%E5%A5%BD2&location=CN\_BJ&location\_desc=CNBJ&all\_file=all1000.txt&card\_file=card1000.txt&province=&city=&district= |
| 后端实现函数 | ImportData |
| 后端返回结果示例 | {"info": "Everything is OK", "import\_flag\_string": "CN\_BJ@BJBJT@447883", "ret": "1101"} |
| 实现流程说明 | 检查参数，提交gearman任务，返回 |

* + - * 1. 4.17 删除文件

|  |  |
| --- | --- |
| 涉及数据对象 | File |
| 前端请求路径 | Ajax/deleteFile/ |
| 前端请求参数示例 | 1. id=1 |
| 后端实现函数 | DeleteFile |
| 后端返回结果示例 | {"info": "Everything is OK", "ret": "1101"} |
| 实现流程说明 | 检查权限，删除数据，返回 |

* + - * 1. 4.18 导入状态查询

|  |  |
| --- | --- |
| 涉及数据对象 | 无 |
| 前端请求路径 | Ajax/show\_import\_data\_process/ |
| 前端请求参数示例 | 无 |
| 后端实现函数 | ShowImportDataProcess |
| 后端返回结果示例 | [2015-03-09 08:57:54+00:00]success uploaded file:all1000.txt  [2015-03-09 08:57:57+00:00]success uploaded file:card1000.txt  [2015-03-09 08:58:00+00:00]success submit import site:BJBJT's data task |
| 实现流程说明 | 检查权限，返回 |

* + - * 1. 4.19 检查站点代码和地区代码唯一性

|  |  |
| --- | --- |
| 涉及数据对象 | Station Location |
| 前端请求路径 | Ajax/check\_unique/ |
| 前端请求参数示例 | 1. type=2&name=DFAD |
| 后端实现函数 | checkUnique |
| 后端返回结果示例 | {"info": "Everything is OK", "ret": "1101"} |
| 实现流程说明 | 判断是否存在，返回 |

* + - * 1. 4.20 设置用户网站语言

|  |  |
| --- | --- |
| 涉及数据对象 | User |
| 前端请求路径 | Ajax/set\_language |
| 前端请求参数示例 | 1. type=zh-cn |
| 后端实现函数 | setLanguage |
| 后端返回结果示例 | {"info": "Everything is OK", "ret": "1101"} |
| 实现流程说明 | 检查权限，更改，返回 |

* + - * 1. 4.21 取得站点油枪列表

|  |  |
| --- | --- |
| 涉及数据对象 | Station |
| 前端请求路径 | Ajax/get\_guns\_id/ |
| 前端请求参数示例 | 1. site=BJBJT |
| 后端实现函数 | getGusIdBySite |
| 后端返回结果示例 | {"info": "Everything is OK", "data": [6, 12, 8, 9, 2, 1, 7, 10, 3, 11, 4, 5], "ret": "1101"} |
| 实现流程说明 | 检查权限，查询数据库，返回 |

* + - * 1. 4.22 增加油机

|  |  |
| --- | --- |
| 涉及数据对象 | Station |
| 前端请求路径 | Ajax/add\_channel\_machine/ |
| 前端请求参数示例 | 1. site=BJBJT&name=dfd&guns%5B%5D=1&type=2 |
| 后端实现函数 | views.py文件中的ajax\_get\_pro\_type函数 |
| 后端返回结果示例 | {"info": "Everything is OK", "ret": "1101"} |
| 实现流程说明 | 检查权限，更新，返回 |

* + - * 1. 4.23取得油机列表

|  |  |
| --- | --- |
| 涉及数据对象 | Statino |
| 前端请求路径 | Ajax/get\_channel\_machine/ |
| 前端请求参数示例 | 1. site=BJBJT |
| 后端实现函数 | getChannelAndMachineAjax |
| 后端返回结果示例 | {"info": "Everything is OK", "data": {"machines": [{"name": "dfd", "value": ["1"]}]}, "ret": "1101"} |
| 实现流程说明 | 检查权限，返回 |

* + - * 1. 4.24 删除通道

|  |  |
| --- | --- |
| 涉及数据对象 | Station |
| 前端请求路径 | Ajax/del\_channel/ |
| 前端请求参数示例 | 1. site=BJBJT&name=ddfdf |
| 后端实现函数 | delChannelAjax |
| 后端返回结果示例 | {"info": "Everything is OK", "ret": "1101"} |
| 实现流程说明 | 检查权限，删除数据，返回 |

* + - * 1. 4.25 删除油机

|  |  |
| --- | --- |
| 涉及数据对象 | Station |
| 前端请求路径 | Ajax/del\_machine/ |
| 前端请求参数示例 | 1. site=BJBJT&name=ddfdf |
| 后端实现函数 | delMachineAjax |
| 后端返回结果示例 | {"info": "Everything is OK", "ret": "1101"} |
| 实现流程说明 | 检查权限，删除数据，返回 |

* + - * 1. 4.26删除油位

|  |  |
| --- | --- |
| 涉及数据对象 | Station |
| 前端请求路径 | Ajax/del\_level/ |
| 前端请求参数示例 | 1. site=BJBJT&name=ddfdf |
| 后端实现函数 | delLevelAjax |
| 后端返回结果示例 | {"info": "Everything is OK", "ret": "1101"} |
| 实现流程说明 | 检查权限，删除数据，返回 |

* + - * 1. 4.27 检查是否存在油机油位通道名

|  |  |
| --- | --- |
| 涉及数据对象 | Scenic Station |
| 前端请求路径 | Ajax/check\_passage\_machine\_level\_name/ |
| 前端请求参数示例 | 1. type=0&name=ddfdf&site=BJBJT |
| 后端实现函数 | checkPassageMachineLevelName |
| 后端返回结果示例 | {"info": "Everything is OK", "ret": "1101"} |
| 实现流程说明 | 检查权限，判断是否存在，返回 |

* + - * 1. 4.28 取得中国行政区划信息

|  |  |
| --- | --- |
| 1. 涉及数据对象 | 1. DimChinaProvinceCityDistrict |
| 1. 前端请求路径 | 1. ajax/get\_china\_location/ |
| 1. 前端请求参数示例 | 1. parent=0&level=1 |
| 1. 后端实现函数 | 1. get\_china\_location |
| 1. 后端返回结果示例 | 1. {"info": "Everything is OK", "ret": "1101", 2. "dict\_city": [ 3. [130100, "\u77f3\u5bb6\u5e84\u5e02"], 4. [130200, "\u5510\u5c71\u5e02"], 5. [130300, "\u79e6\u7687\u5c9b\u5e02"], 6. [130400, "\u90af\u90f8\u5e02"], 7. [130500, "\u90a2\u53f0\u5e02"], 8. [130600, "\u4fdd\u5b9a\u5e02"], 9. [130700, "\u5f20\u5bb6\u53e3\u5e02"], 10. [130800, "\u627f\u5fb7\u5e02"], 11. [130900, "\u6ca7\u5dde\u5e02"], 12. [131000, "\u5eca\u574a\u5e02"], 13. [131100, "\u8861\u6c34\u5e02"]]} |
| 1. 实现流程说明 | 1. 查询数据库，返回 |

* + - * 1. 4.29 根据油站代码取得油站信息

|  |  |
| --- | --- |
| 1. 涉及数据对象 | 1. Station |
| * + 1. 前端请求路径 | 1. Ajax/get\_info\_by\_sitename/ |
| 1. 前端请求参数示例 | 1. site\_name=ddd |
| 1. 后端实现函数 | 1. getInfoBySiteName |
| 1. 后端返回结果示例 | 1. {"info": "Everything is OK", 2. "province": 110000, "provinces": [ 3. {"id": 110000, "name": "\u5317\u4eac\u5e02"}, 4. {"id": 120000, "name": "\u5929\u6d25\u5e02"}], "district": 110102, "city": 110100, "ret": "1101", "districts": [{"id": 110101, "name": "\u4e1c\u57ce\u533a"}, {"id": 110102, "name": "\u897f\u57ce\u533a"}, {"id": 110105, "name": "\u671d\u9633\u533a"}, {"id": 110106, "name": "\u4e30\u53f0\u533a"}, {"id": 110107, "name": "\u77f3\u666f\u5c71\u533a"}, {"id": 110108, "name": "\u6d77\u6dc0\u533a"}, {"id": 110109, "name": "\u95e8\u5934\u6c9f\u533a"}, {"id": 110111, "name": "\u623f\u5c71\u533a"}, {"id": 110112, "name": "\u901a\u5dde\u533a"}, {"id": 110113, "name": "\u987a\u4e49\u533a"}, {"id": 110114, "name": "\u660c\u5e73\u533a"}, {"id": 110115, "name": "\u5927\u5174\u533a"}, {"id": 110116, "name": "\u6000\u67d4\u533a"}, {"id": 110117, "name": "\u5e73\u8c37\u533a"}], "citys": [{"id": 110100, "name": "\u5e02\u8f96\u533a"}, {"id": 110200, "name": "\u53bf"}], "data": "\u4f60\u597d2"} |
| 1. 实现流程说明 | 1. 检查权限，查询，返回 |

* + - * 1. 4.30 更新油站信息

|  |  |
| --- | --- |
| 1. 涉及数据对象 | 1. Statino |
| * + 1. 前端请求路径 | 1. Ajax/save\_station\_info/ |
| 1. 前端请求参数示例 | 1. site\_name=BJBJT&description=%E4%BD%A0%E5%A5%BD2&province=110000&city=110100&district=110102 |
| 1. 后端实现函数 | 1. saveStationInfo |
| 1. 后端返回结果示例 | 1. {"info": "Everything is OK", "ret": "1101"} |
| 1. 实现流程说明 | 1. 检查权限，保存，返回 |