|  |  |
| --- | --- |
| **HiDM CIM开发平台开发手册** | |
|  | |
|  | |
| **Filename 文档名:**  **版本:** | HiDM CIM开发平台开发手册.docx  V1.4 |
|  |  |
| **Prepared By 编制:** | Xu,Tao  HiDM CIM |

文档更改记录

下表包含对本文档所做更改的历史记录，包括更改日期，创建日期，生成的版本号以及更改摘要。

|  |  |  |
| --- | --- | --- |
| Date  更改日期 | Author(s)  编者 | Description of Change  更改描述 |
| 2018/07/03 | Xu,Tao | 初始版本 |
| 2018/07/26 | Xu,Tao | V1.1，添加Chart封装使用方法 |
| 2018/07/27 | Xu,Tao | V1.2，引入Bootstrap布局 |
| 2018/08/01 | Xu,Tao | V1.3，引入Select2多选及AutoComplete控件，添加必选参数验证逻辑 |
| 2018/09/10 | Xu,Tao | V1.4，添加Windows Service开发和Job管控 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

目录

[1. 系统简介 6](#_Toc521488878)

[2. 源码解决方案结构 6](#_Toc521488879)

[2.1 简介 6](#_Toc521488880)

[2.2 项目结构 7](#_Toc521488881)

[2.2.1 Web.UI 7](#_Toc521488882)

[3. 功能开发流程 9](#_Toc521488883)

[3.1 需求概要 9](#_Toc521488884)

[3.2.1 查询功能 10](#_Toc521488885)

[3.3 程序设计 11](#_Toc521488886)

[3.3.1 Model创建 11](#_Toc521488887)

[3.3.2 Data Access创建 12](#_Toc521488888)

[3.3.3 Business创建 13](#_Toc521488889)

[3.3.4 View创建 14](#_Toc521488890)

[3.4 菜单配置 18](#_Toc521488891)

[3.4.1 模块配置 18](#_Toc521488892)

[3.4.2 按钮配置 19](#_Toc521488893)

[3.4.3 权限分配 20](#_Toc521488894)

[4. 关键逻辑 22](#_Toc521488895)

[4.1 Chart 生成 22](#_Toc521488896)

[4.1.1 SQL 22](#_Toc521488897)

[4.1.2 Business 22](#_Toc521488898)

[4.1.3 JavaScript 23](#_Toc521488899)

[4.1.4 效果展示 23](#_Toc521488900)

[4.2 Bootstrap布局 23](#_Toc521488901)

[4.2.1 Javascript 24](#_Toc521488902)

[4.2.2 页面布局 24](#_Toc521488903)

[4.2.3 效果展示 25](#_Toc521488904)

[4.3 AutoComplete和多选控件 27](#_Toc521488905)

[4.3.1 Html 27](#_Toc521488906)

[4.3.2 JavaScript 27](#_Toc521488907)

[4.3.3 Controller 28](#_Toc521488908)

[4.3.4 控件联动 28](#_Toc521488909)

[4.4 必选参数验证 29](#_Toc521488910)

[4.4.1 Html 29](#_Toc521488911)

[4.4.2 后台验证 29](#_Toc521488912)

[4.4.3 前台验证 29](#_Toc521488913)

[4.5 复写方法 29](#_Toc521488914)

[4.5.1 查询 30](#_Toc521488915)

[4.5.2 重置 30](#_Toc521488916)

**Figures**

[Figure 1源代码解决方案架构 6](#_Toc521488917)

[Figure 2各项目关联关系 7](#_Toc521488918)

[Figure 3 Web.UI结构 7](#_Toc521488919)

[Figure 4功能开发流程 9](#_Toc521488920)

[Figure 5 批量Release功能分析 10](#_Toc521488921)

[Figure 6 程序设计 11](#_Toc521488922)

[Figure 7 LotHoldInfoInput 11](#_Toc521488923)

[Figure 8 BaseHoldInfo 12](#_Toc521488924)

[Figure 9 ReleaseLotInput 12](#_Toc521488925)

[Figure 10 LotHoldInfoRepository 13](#_Toc521488926)

[Figure 11 UI Layout 15](#_Toc521488927)

[Figure 12 MainForm JS 16](#_Toc521488928)

[Figure 13 Popup Form Js 17](#_Toc521488929)

[Figure 14 JS模块注册 17](#_Toc521488930)

[Figure 15模块配置 18](#_Toc521488931)

[Figure 16按钮配置 19](#_Toc521488932)

[Figure 17模块权限分配 20](#_Toc521488933)

[Figure 18 按钮权限分配 21](#_Toc521488934)

[Figure 19 Chart SQL事例 22](#_Toc521488935)

[Figure 20 Chart Bussines 代码事例 23](#_Toc521488936)

[Figure 21 Chart Javascript代码事例 23](#_Toc521488937)

[Figure 22 JS依赖引入 24](#_Toc521488938)

[Figure 23 Chart & Grid Size自动调整 24](#_Toc521488939)

[Figure 24 Bootstrap 效果展示 25](#_Toc521488940)

[Figure 25自适应效果 26](#_Toc521488941)

[Figure 26 Select控件 27](#_Toc521488942)

[Figure 27 Select2初始化 27](#_Toc521488943)

[Figure 28 Select2后台API 28](#_Toc521488944)

[Figure 29 控件联动参数设定 28](#_Toc521488945)

[Figure 30控件联动事件 29](#_Toc521488946)

[Figure 31 必输栏位标志 29](#_Toc521488947)

[Figure 32必输参数后台判断 29](#_Toc521488948)

[Figure 33 必输参数前端台判断 29](#_Toc521488949)

[Figure 34 获取echarts实例 30](#_Toc521488950)

[Figure 35 查询逻辑流程 30](#_Toc521488951)

[Figure 36 重置逻辑流程 30](#_Toc521488952)

# 系统简介

系统设计为基于Asp.net MVC开发的通用平台,可用作基础开发框架，用于添加CIM相关功能,具有成熟的数据访问(Dapper)、作业调度(Quartz.net)、日志记录(Log4Net)、权限控制，权限管控精确到按钮级别。

# 源码解决方案结构

* 1. 简介

系统源码解决方案共由四个项目（EIP.Web/HiDM.Reporting.Business/HiDM.Reporting.DataAccess/HiDM.Reporting.Models）组成。

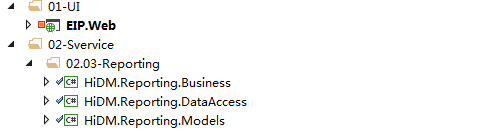


Figure 1源代码解决方案架构

1. EIP.Web为页面表示层，分为前端界面和后端Controller，主要负责前端页面展示及用户交互，使用Ajax将用户的请求通过Jason格式提交给后端Controller，后端Controller调用Business层对用户请求的数据进行处理；
2. HiDM.Reporting.Business为业务层，主要负责将表示层的传递的参数进行业务逻辑处理，调用MES等其他系统的API进行作业，或调用DataAccess层将数据持久化至数据库；
3. HiDM.Reporting.DataAccess为数据持久化层，主要负责对Service层处理好的数据进行增删改查操作；
4. HiDM.Reporting.Models为数据模型层，主要用于数据库持久化ORM操作或各业务层之间的数据传递。

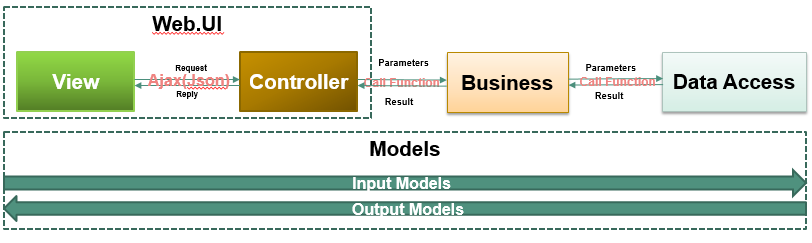


Figure 2各项目关联关系

* 1. 项目结构
     1. Web.UI

Web UI用于界面展示，由cshtml、javascript、CSS和后端Controller组成。

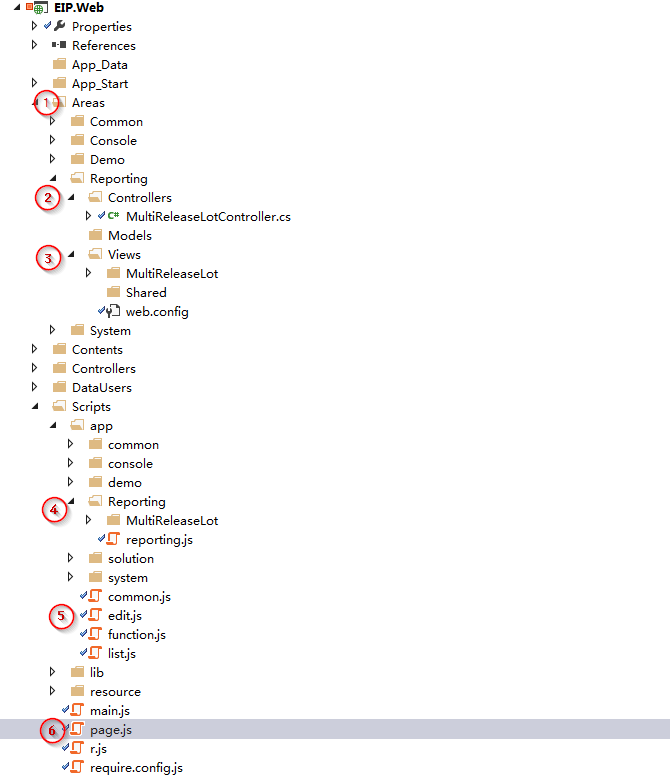


Figure 3 Web.UI结构

1. Areas

MVC本身提倡的就是关注点分离。但是当项目本身的业务逻辑足够复杂，如果所有的业务逻辑都写个Controller文件夹下面的时候，就会看到非常庞大的各种命名的Controller，这个时候Area的作用就非常明显了，将不同的模块通过区域加以区分，简单明了。本系统根据不同的模块将System、Reporting等Area，目前开发功能请在Reporting下添加。

1. Controller

本系统中Controller有如下两种作用，可以根据功能添加Controller。

1. 响应用户页面请求，返回相应的View给用户；
2. 响应用户的操作请求，对用户的输入进行处理，并返回结果；
3. View

用户界面，最终展示给用户的UI，主要通过html、Javascript和CSS实现，一般功能会包括一个主界面和多个popup界面，已Controller为单位建立View文件夹。

1. Scripts

浏览器端javascript脚本，主要用于前端样式处理、数据提交与展示等，包括引用的第三方脚本、自定义Function和页面脚本，一般情况下，只需要根据功能在App-Reporting下建立相应文件夹，并在其中分别对主界面和popup界面建立对应的js文件。

1. Common Javascripts

目前系统已有的共用js，如弹出提示框、根据权限获取界面按钮、验证Form等。

1. 模块javascripts注册

功能JS创建完成后，需要在page.js中进行注册，以便界面可以调用。

# 功能开发流程

在本系统中开发新功能的流程大致如下，下面将以实际需求为例讲解如何实现所要功能。



Figure 4功能开发流程

* 1. 需求概要

根据Lot ID或Hold To Owner Department查询当前Hold的信息（展示Hold列表，并显示各Hold Code的饼状分布），同时提供批量Release功能。

* 1. 需求分析

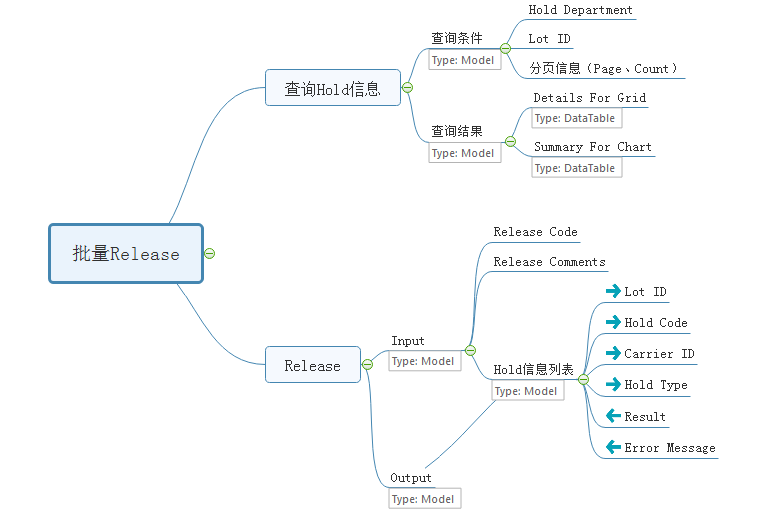
本需求中共两个主要功能点：查询和批量Release

Figure 5 批量Release功能分析

* + 1. 查询功能

1. 输入：Lot ID（文本框）、 Hold To Owner Department（下拉框）、分页信息；
2. 业务逻辑：根据输入的条件，从MES数据库中查询Hold信息；
3. 输出：Hold列表（表格）、Hold Code的饼状分布（图表）；
   * 1. 批量Release
4. 输入：选中的Lot ID和Hold Code的组合列表、Release Code、Comments；
5. 业务逻辑：根据用户选择的Lot和Hold Code的组合列表、Release Code及Comments，调用MES API进行Release操作；
6. 输出：每个Lot和Hold Code组合的成功或失败消息，显示回Grid中；
   1. 程序设计

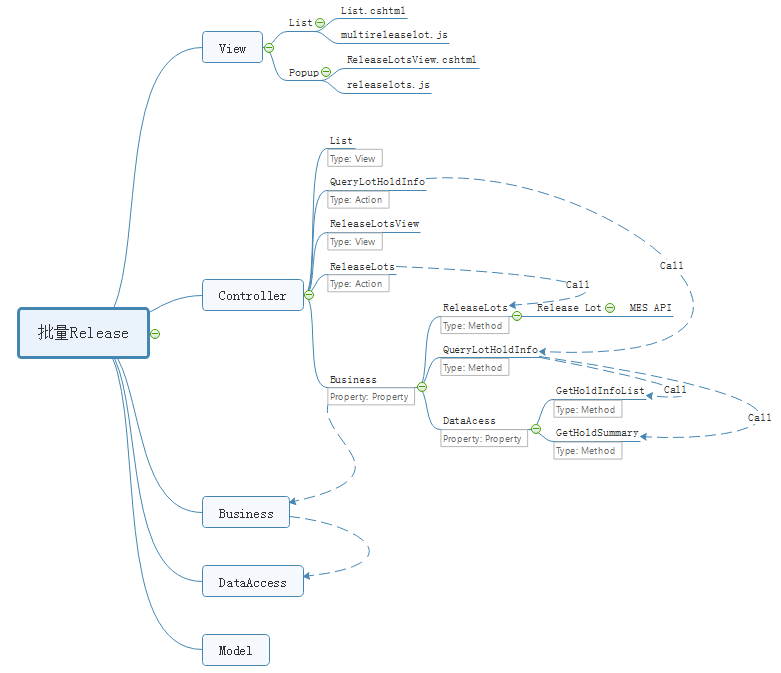


Figure 6 程序设计

* + 1. Model创建

根据功能，在HiDM.Reporting.Models项目中建立Demo文件夹，且由上节需求分析可知，需要建立两个Model用于分别对应查询和Release输入。

1. 查询条件输入

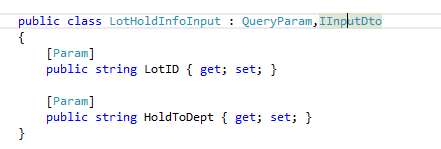


Figure 7 LotHoldInfoInput

新建LotHoldInfoInput,继承QueryParam类（封装定义分页信息和获取查询参数）和IInputDto接口，并将查询参数设定Param Attribute。

1. Release输入
2. BaseHoldInfo 选中的Hold信息参数类

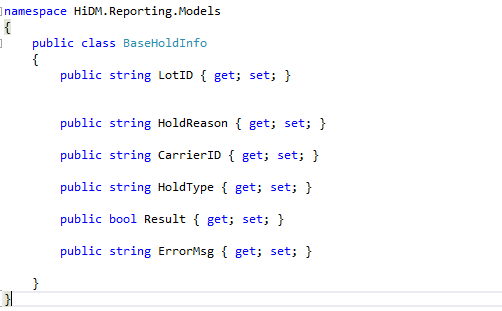


Figure 8 BaseHoldInfo

1. ReleaseLotInput Release输入参数类型，包括选中的Hold信息列表、UserID、Release Reason、Comments。

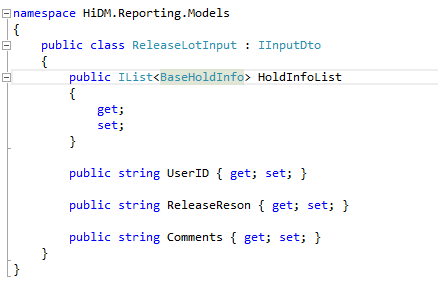


Figure 9 ReleaseLotInput

* + 1. Data Access创建

本需求中，数据查询主要分为两个部分：Hold信息明细和Hold Code分布查询，其中Hold信息明细会作为Grid进行展示，Hold Code分布用于图表展示。

1. Interface

新建接口ILotHoldInfoRepository,用于定义查询两类信息的方法接口：GetHoldInfoList和GetHoldSummary，两个方法的参数相同，但是由于GetHoldInfoList为Grid，需要讲分页信息传递给客户端，所以返回值类型为PagedResults，而GetHoldSummary只需要数据，则直接返回DataTable数据集。

1. Implement

新建LotHoldInfoRepository，继承ReportingRepository并实现接口ILotHoldInfoRepository方法:

1. 实现GetHoldInfoList方法，调用base.PagingQueryAsyncDataTable查询明细；
2. 实现GetHoldSummary，调用QueryAsyncDataTable查询图表信息。



Figure 10 LotHoldInfoRepository

* + 1. Business创建

本需求中，业务逻辑主要分为两个部分：数据查询、批量Release。

1. Interface

新建接口IMultiReleaseLotLogic,用于定义查询（QueryLotHoldInfo）和Release（ReleaseLots）的方法接口。

1. QueryLotHoldInfo:用于查询信息，参数类型为LotHoldInfoInput，返回值类型为PagedResults，由于客户端点击查询按钮后需要同时查询两类数据，所以在实现时，会将图表数据防止与PagedResults的ExtraDatatables属性中，与Grid数据同时返回给客户端。
2. ReleaseLots：用于批量Release Lot，输入参数类型为ReleaseLotInput，其中包含需要Release的Hold信息列表和Release Reason信息，由于调用MES API时为单个Release，每一行选中的Hold信息都会有单独的结果信息，所以需要在调用完MES API之后将结果写回到BaseHoldInfo供客户端展示，因此返回值为BaseHoldInfo的列表
3. Implement

新建MultiReleaseLotLogic，实现接口IMultiReleaseLotLogic方法:

1. 定义ILotHoldInfoRepository类型的字段\_HoldLotInfoRepository，用于数据库操作。
2. 实现创建参数为ILotHoldInfoRepository的构造函数，并将参数赋值给\_HoldLotInfoRepository字段；
3. 实现QueryLotHoldInfo方法：

* 首先调用\_HoldLotInfoRepository.GetHoldInfoList查询Hold信息明细，将返回值赋值给pageResults临时变量；
* 调用\_HoldLotInfoRepository.GetHoldSummary(input)，将返回值添加到pageResults的ExtraDataTables中。

1. 实现ReleaseLots，循环参数ReleaseLotInput中的Hold信息列表，调用MES API进行Release列表，并将结果信息返回到Hold信息的Result和ErrorMsg中。
   * 1. View创建
2. Controller创建

在Areas->Reporting->Controller中建立MultiReleaseLotController，继承BaseController。

1. 定义IMultiReleaseLotLogic类型的字段\_ \_MultiReleaseLotLogicLogic，用于业务操作。
2. 实现创建参数为IMultiReleaseLotLogic的构造函数，并将参数赋值给\_MultiReleaseLotLogicLogic字段；
3. 创建List方法，用于查询主界面访问；
4. 创建QueryLotHoldInfo方法，用于调用\_MultiReleaseLotLogicLogic.QueryLotHoldInfo，返回查询结果；
5. 创建ReleaseLotsView方法，弹出页面访问；
6. 创建ReleaseLots方法，用于调用\_MultiReleaseLotLogicLogic.ReleaseLots进行Release Lot；
7. View创建

在Areas->Reporting->View目录下，建立与Controller同名的文件夹（MVC机制，去除”Controller”）,并建立如下两个View：

1. List ：功能主界面，用于查询结果、选中结果、弹出Release Reason选择界面（ReleaseLotsView），界面布局如下图：

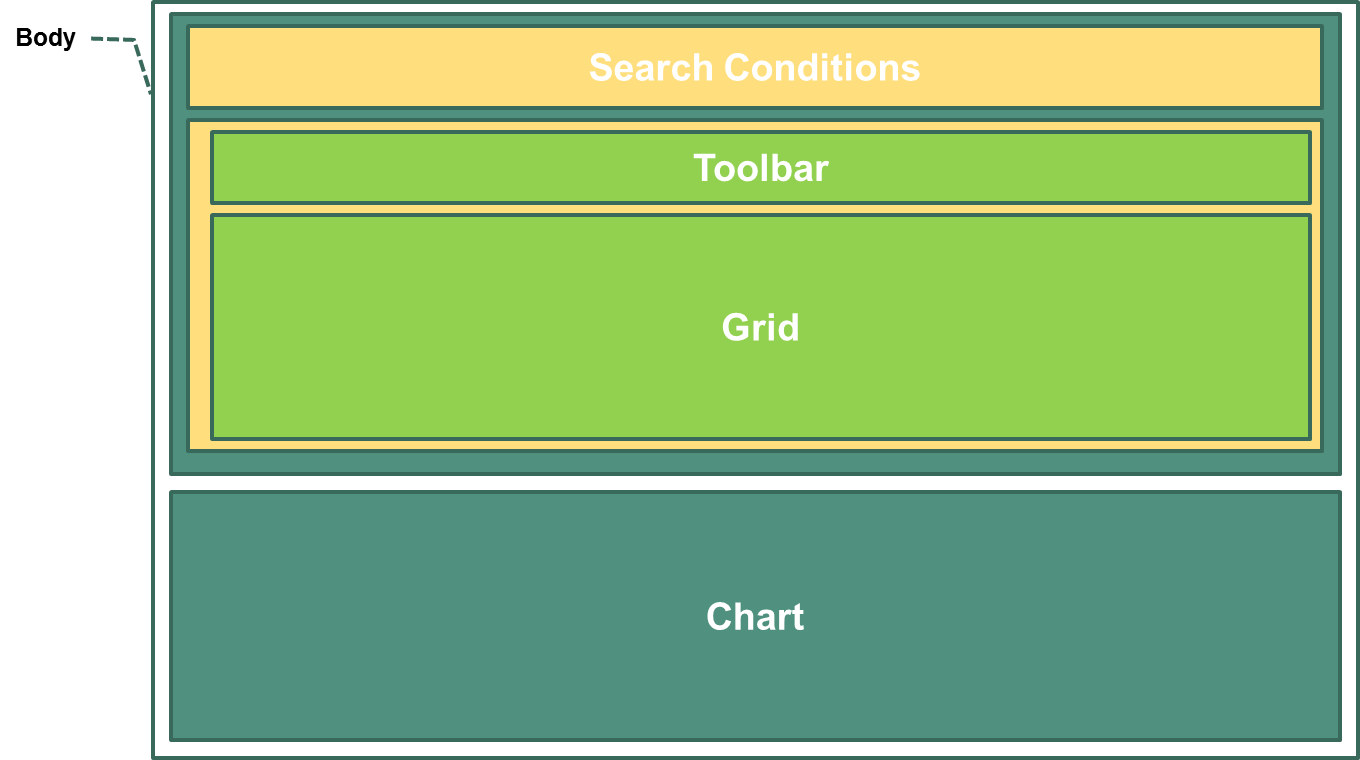


Figure 11 UI Layout

1. 页面中使用通过赋值ViewBag.Require，指定使用的Javascript模块；
2. 通过赋值Layout，指定使用的模板页，主界面使用："~/Views/Shared/\_LayoutPageBaseList.cshtml"
3. 使用css class：ui-layout-center/ ui-layout-south/ ui-layout-north/ ui-layout-east/ ui-layout-west进行布局；
4. @{ Html.RenderPartial("~/Areas/Common/Views/Shared/\_PartialButton.cshtml"); }用于获取在后台动态配置的页面按钮；
5. [使用@Html.DropDownListSQL](mailto:使用@Html.DropDownListSQL)并指定“ConnectionString、SQL、Name、HtmlAttributes”属性初始化下拉选框；
6. 搜索按钮的Name必须为：btn\_select\_box；重置按钮的Name必须为btn\_reset\_box；
7. 定义id为list的table，用于展示列表信息；
8. 定义id为page的div，用于展示分页信息；
9. 定义id为browserAnalysis的div，用于展示Chart信息；
10. 主界面javascript（multireleaselot.js）
11. 使用以下代码指定依赖的js模块，并执行界面加载完成后初始化的function；

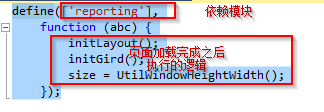


Figure 12 MainForm JS

1. initLayout,初始化界面Layout；
2. initGrid，初始化Grid：指定查询数据请求的url，columns，分页信息等，并指定grid数据加载完成之后，调用loadGraph展示chart信息；
3. UtilWindowHeightWidth：设定页面高度；
4. loadGraph，初始化并展示Chart信息；
5. releaseLots，弹出Release弹出框，实现定义好function，按钮在后台动态配置，方便权限管控；
6. releaseComplete，Release完成之后执行的回调函数，用于展示Release结果；
7. getSeletedHoldInfo，用于在Release弹出框获取当前选择的Lot信息；
8. ReleaseLotView

选中hold信息之后，弹出此页面，供选择Release Reason和Comments。

设定属性Layout = "~/Views/Shared/\_LayoutPageEditSubmit.cshtml";

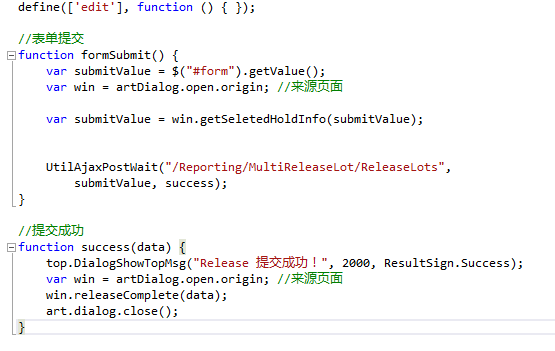
1. Releaselots.

Figure 13 Popup Form Js

1. define:定义依赖项和初始化执行方法；
2. formSubmit:获取选择的Reason、Comment及hold信息列表，通过ajax提交，提供完成之后执行回调函数success；
3. 回调函数success，提示执行成功，主界面显示结果，关闭当前弹出框；
4. JS注册

打开page.js，注册主界面和弹出界面的js，即界面data-require对应js路径。

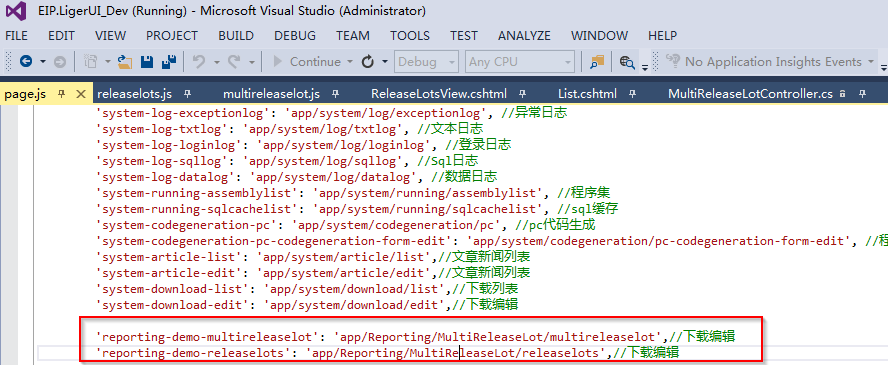


Figure 14 JS模块注册

* 1. 菜单配置
     1. 模块配置

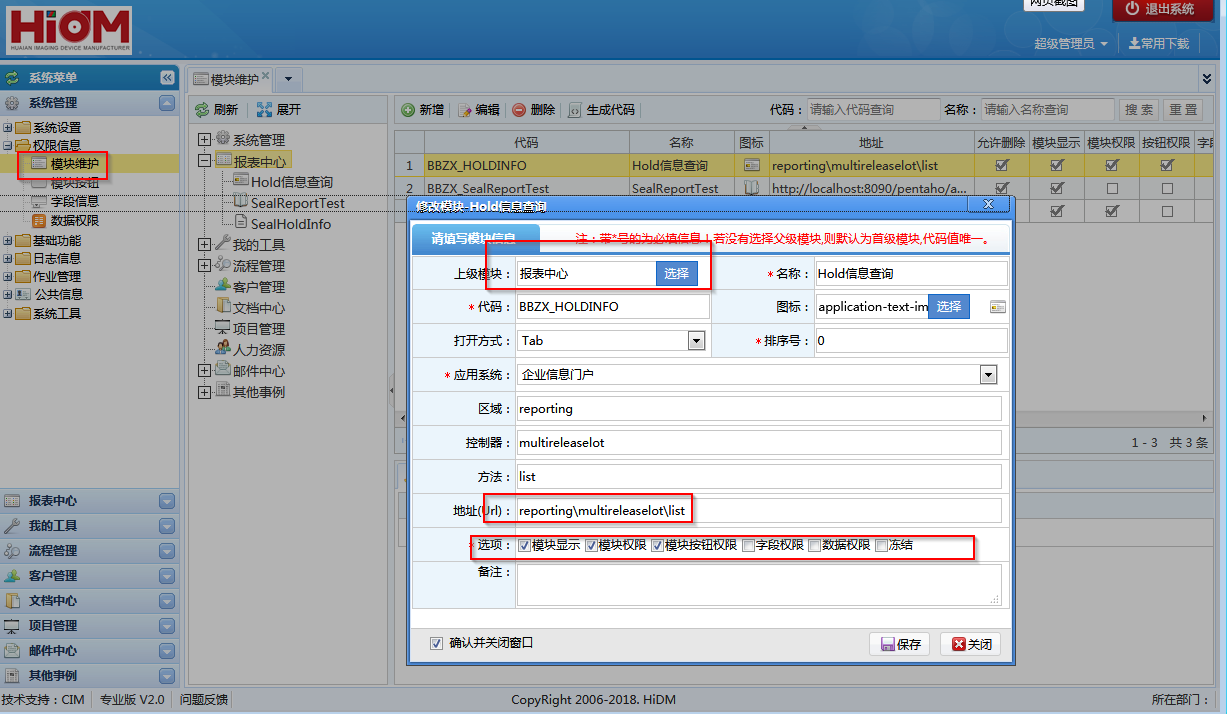


Figure 15模块配置

1. 打开路径：系统管理 – 权限信息 – 模块维护 – 新增；
2. 指定上级模块，菜单名称，图标，打开方式，路径等信息进行保存；
   * 1. 按钮配置

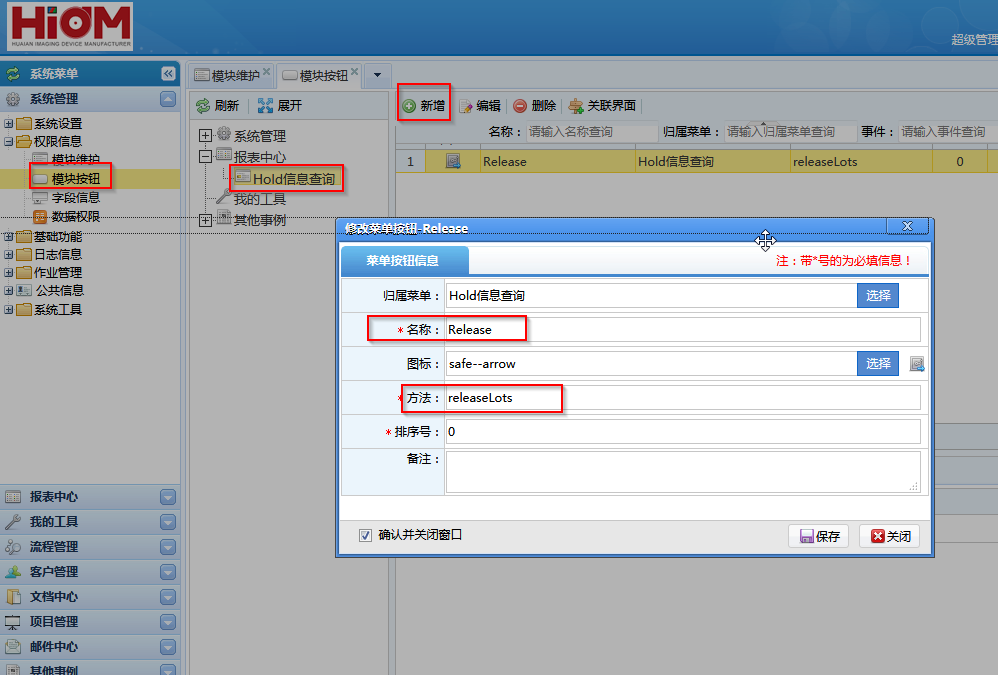


Figure 16按钮配置

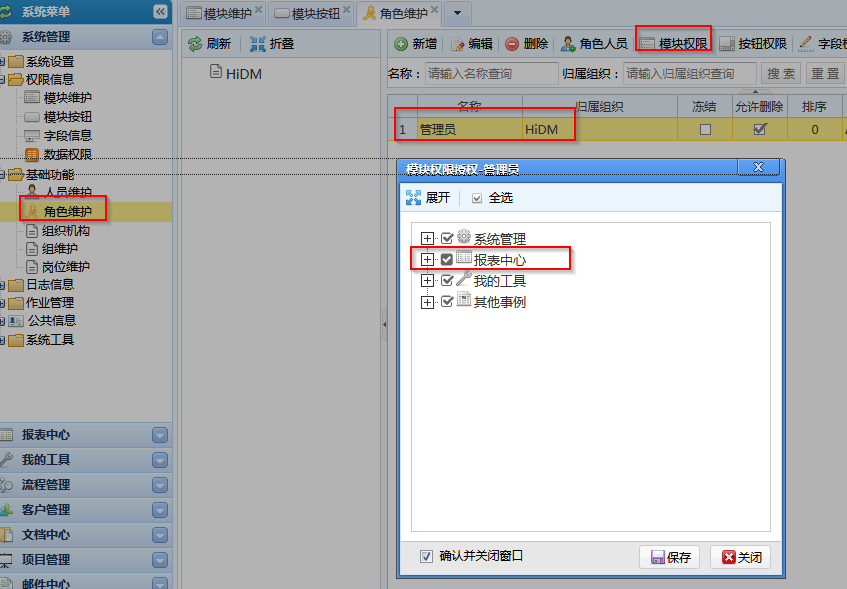
1. 打开路径：系统管理 – 权限信息 – 模块按钮 – 新增；
2. 指定归属菜单，按钮名称，图标，执行js中的方法等信息进行保存；
   * 1. 权限分配
3. 模块权限
4. 打开方式：系统管理 – 基础功能 – 角色维护 – 选择角色 – 模块权限；
5. 勾选需要赋予的权限，保存；

Figure 17模块权限分配

1. 按钮权限
2. 打开方式：系统管理 – 基础功能 – 角色维护 – 选择角色 – 按钮权限；
3. 选择模块，勾选需要赋予的权限的当前模块的按钮，保存；

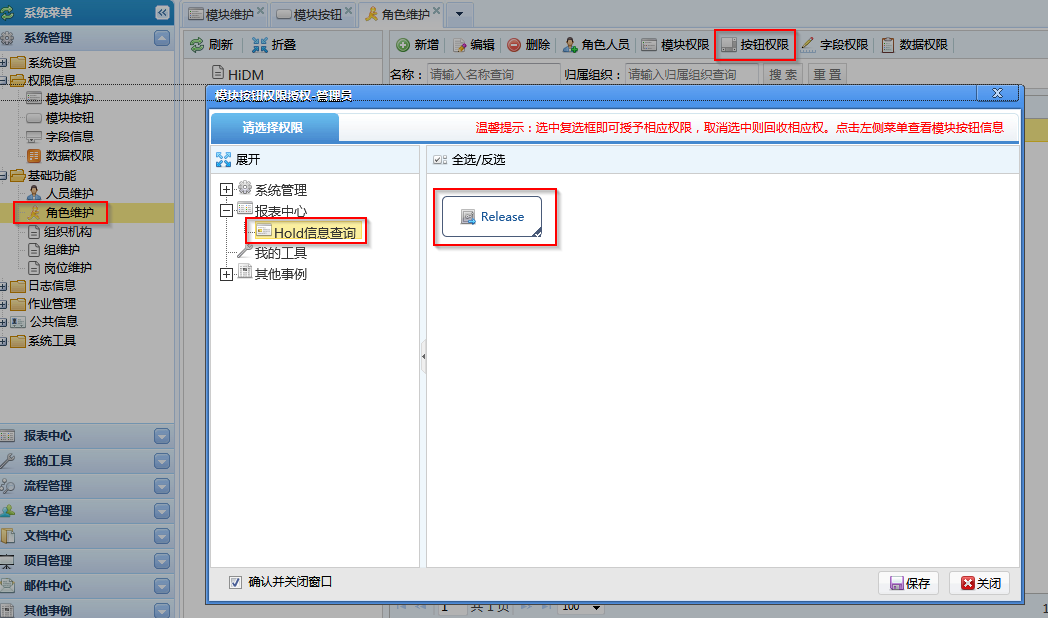


Figure 18 按钮权限分配

# 关键逻辑

* 1. Chart 生成

目前对Chart封装只支持Pie、Bar、Line，其他需求可以自定义或逐步优化

* + 1. SQL

用于图表生成的SQL需包含三个信息：axis、series、value，如下图SQL,WIP产品状态分布图，axis为状态，series为产品，数量为value.

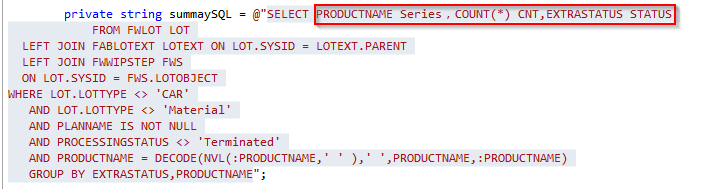


Figure 19 Chart SQL事例

* + 1. Business

在Bussiness查询出DataTable后，调用DataTable拓展方法如下方法将DataTable转为EChart所需Option：

ToChartOption(ChartType chartType,string Title, string subTitle,string columnAxis,string columnSeries, string columnValue, string stackName,ChartDirection chartDirection)，方法具体阐述如下：

/// <summary>

/// 将DataTable转换为EChart Option

/// </summary>

/// <param name="dtData">DataTable</param>

/// <param name="chartType">Chart 类型：line,bar,pie</param>

/// <param name="Title">Chart Title</param>

/// <param name="subTitle">Chart Sub Title</param>

/// <param name="columnAxis">DataTable中用于作为坐标轴的列名</param>

/// <param name="columnSeries">DataTable中用于作为Series的列名</param>

/// <param name="columnValue">DataTable中用于作为值的列名</param>

/// <param name="stackName">Stack名</param>

/// <param name="chartDirection">Chart 方向</param>

/// <returns></returns>

上节中的SQL对应Business代码事例如下：

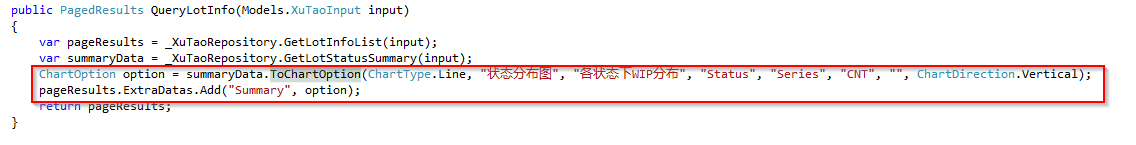


Figure 20 Chart Bussines 代码事例

* + 1. JavaScript

由于后端传递的参数已经符合现有Echart所有要求（不只是Data），所以可以直接通过以下方式绑定到Chart上（如有特殊需求，需对传递回来的参数在JS中进行修改）。

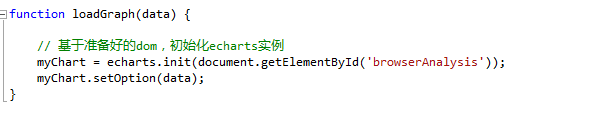
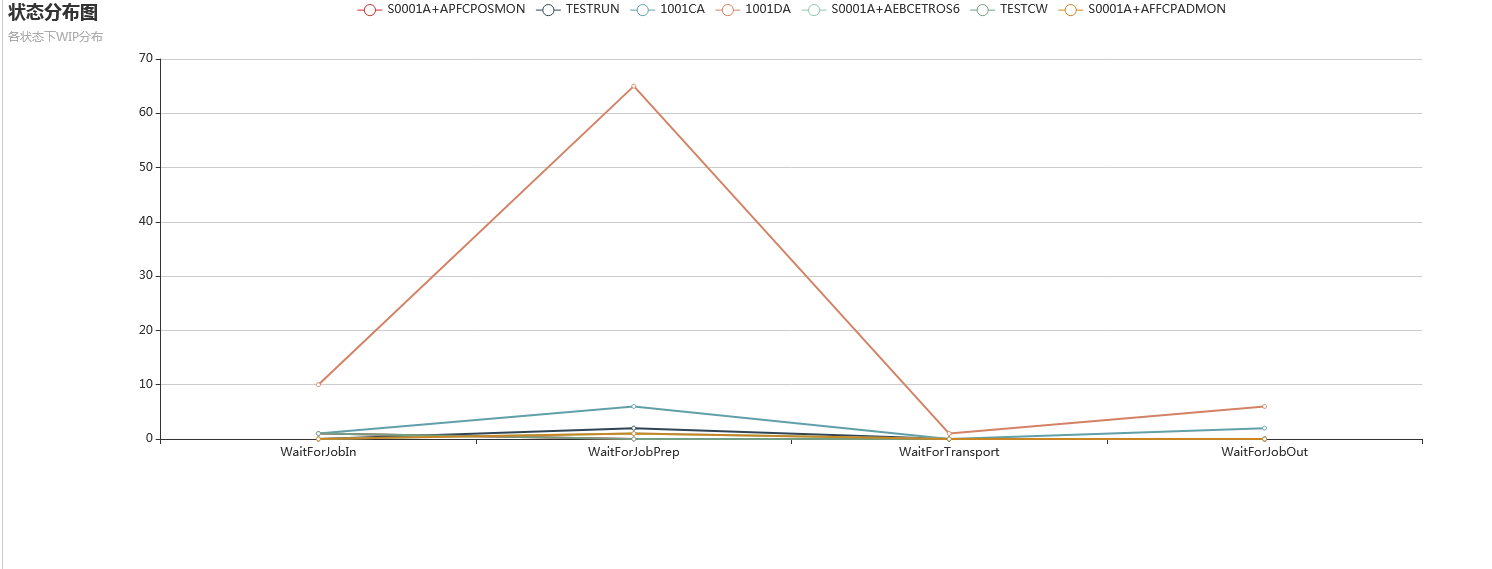


Figure 21 Chart Javascript代码事例

* + 1. 效果展示



* 1. Bootstrap布局

Bootstrap是基于HTML、CSS、JavaScript 开发的简洁、直观、强悍的前端开发框架，使得 Web 开发更加快捷。主要包含以下内容：

 **基本结构**：Bootstrap 提供了一个带有网格系统、链接样式、背景的基本结构。

 **CSS**：Bootstrap 自带以下特性：全局的 CSS设置、定义基本的 HTML 元素样式、可扩展的 class，以及一个先进的网格系统。这将在Bootstrap CSS部分详细讲解。

 **组件**：Bootstrap 包含了十几个可重用的组件，用于创建图像、下拉菜单、导航、警告框、弹出框等等。这将在布局组件部分详细讲解。

 **JavaScript 插件**：Bootstrap包含了十几个自定义的jQuery 插件。您可以直接包含所有的插件，也可以逐个包含这些插件。这将在Bootstrap插件部分详细讲解。

 **定制**：您可以定制Bootstrap的组件、LESS 变量和jQuery 插件来得到您自己的版本

本项目中主要使用Bootstrap进行报表页面布局，实现快速布局和页面自适应。

* + 1. Javascript

1. 在页面对应js中由原先的依赖’reporting’模块修改为依赖’reportingbootstrap’模块，如下图展示：

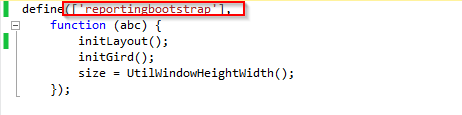


Figure 22 JS依赖引入

1. Chart及Grid Size自动调整

在页面js中定义如下方法，并在方法内写入窗体大小调整后Chart和Grid跟随调整的方法，如下图展示：

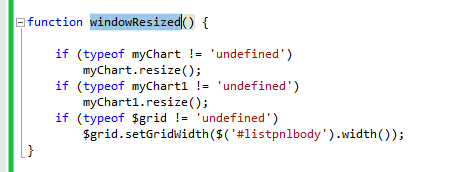


Figure 23 Chart & Grid Size自动调整

* + 1. 页面布局

页面布局主要使用bootstrap的栏栅系统、Panel和form控件进行样式定义，具体可以集合bootstrap及“XuTao\Index.cshtml”事例进行学习。

* + 1. 效果展示

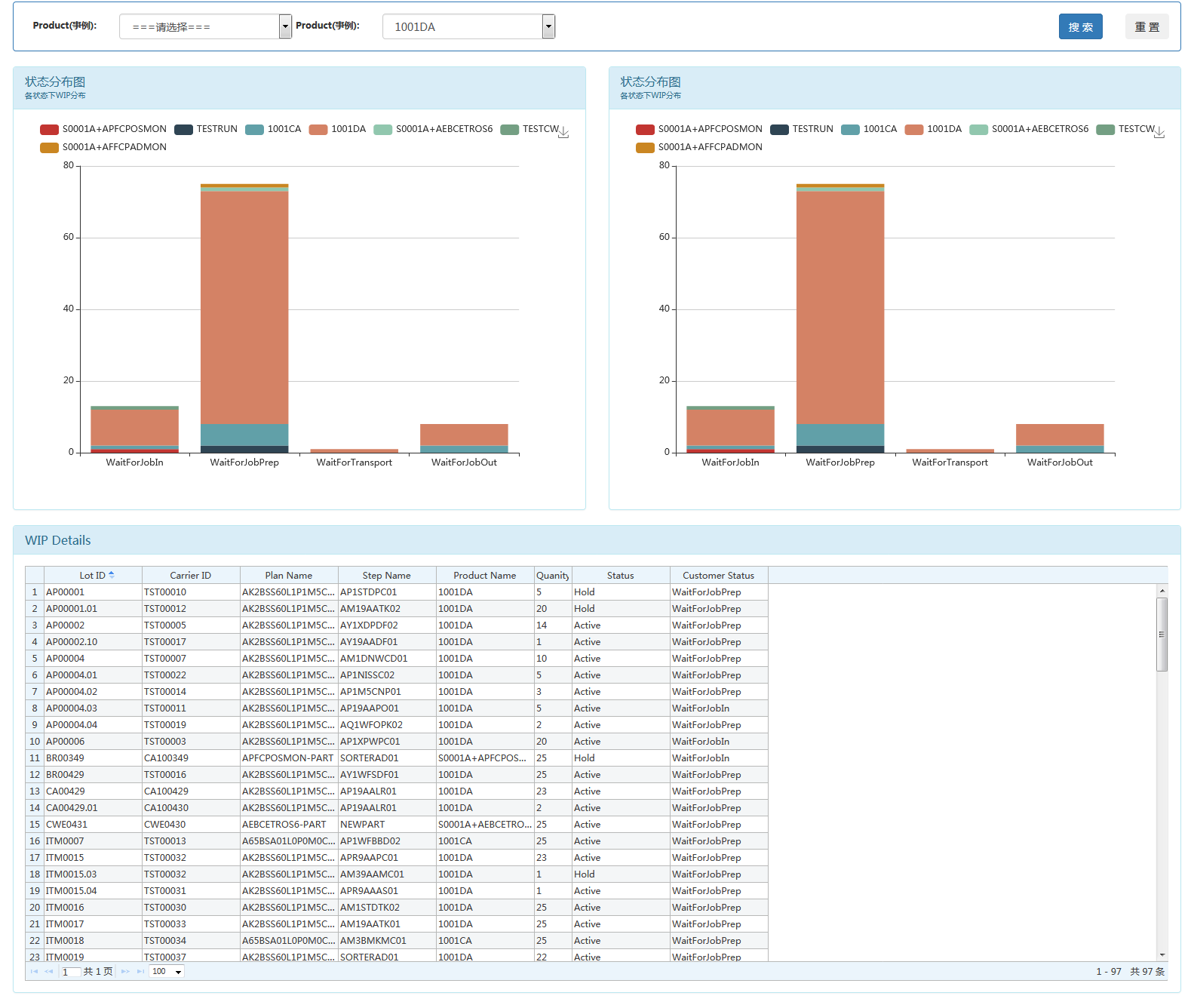


Figure 24 Bootstrap 效果展示

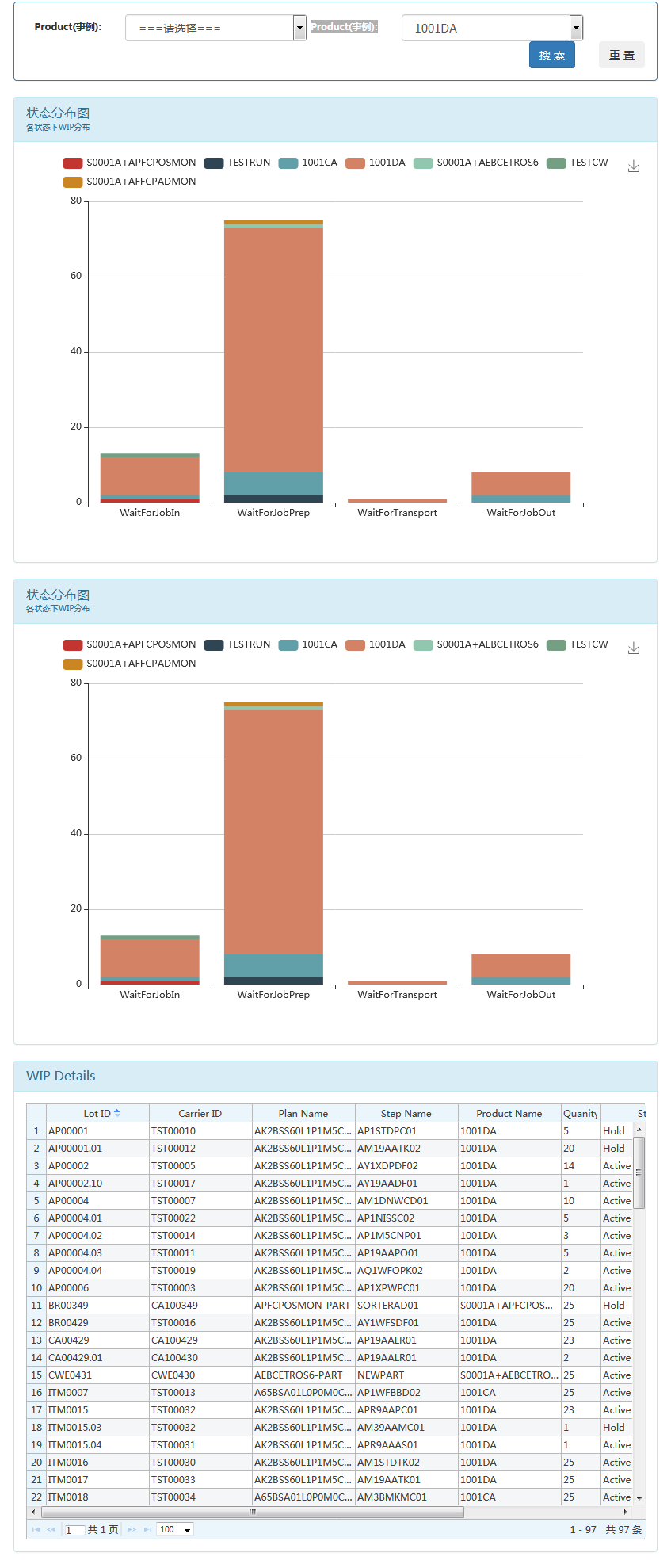


Figure 25自适应效果

* 1. AutoComplete和多选控件

系统中使用select2作为下拉选择控件，支持AutoComplete和多选。该控件使用方法如下：

* + 1. Html

在页面中添加select控件，添加class（必须）: manualfilter select2。



Figure 26 Select控件

* + 1. JavaScript

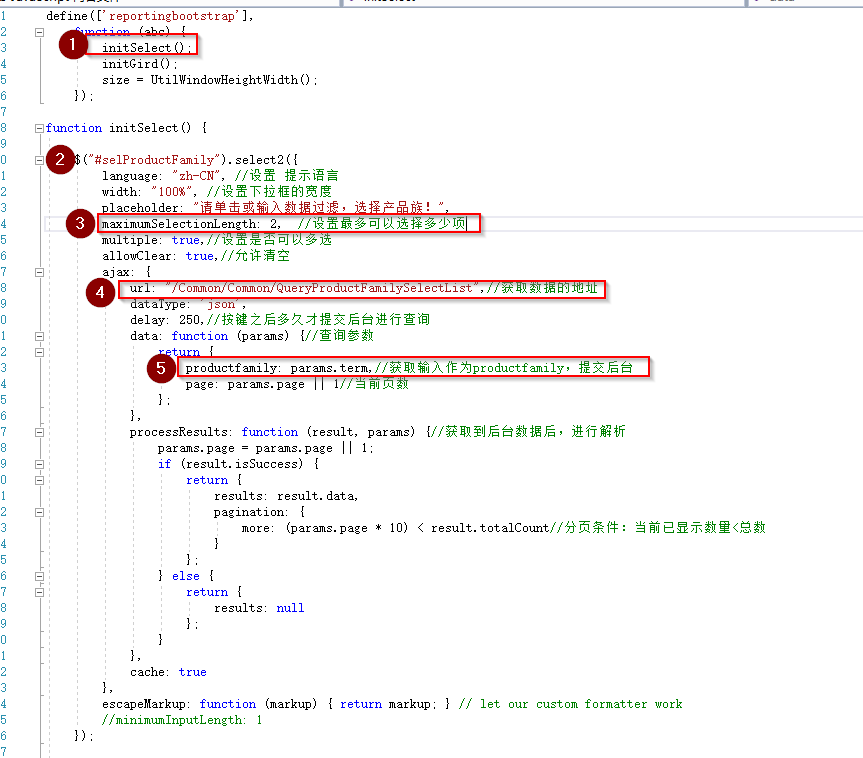


Figure 27 Select2初始化

1. 在页面Javascript中依赖项使用reportingbootstrap，并定义initSelect方法，用于初始化select控件。
2. 使用select2方法初始化选择控件；
3. 如果是多选，通过maxinumSelectionLength属性设定最大可选项；
4. 通过url属性，设定获取下拉选择内容的API地址；
5. params.term是存放当前用户已输入字符的属性，可以取出赋值到自定义变量名中（与后台API参数名一致）；
   * 1. Controller

在Areas\Common\Controller\CommonController.cs中建立新的方法，对应客户端设定的url，用于根据客户端传递的参数，获取下拉数据。

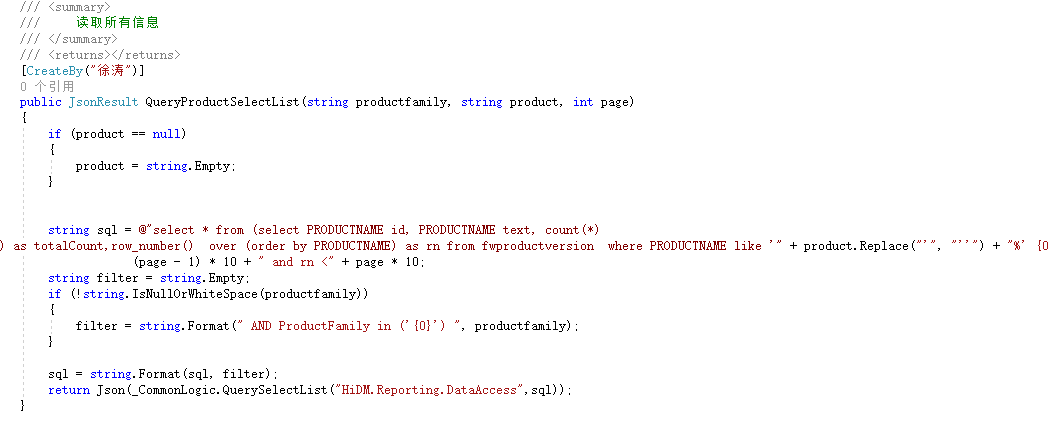


Figure 28 Select2后台API

* + 1. 控件联动
  1. 在子控件ajax.data中，指定提交给后台API的参数时，获取父控件的值，作为参数之一传递给后台API，在后台进行过滤。



Figure 29 控件联动参数设定

* 1. 同时设定父控件的选择和取消选择事件，在选择和取消选择时，清空子控件的值。

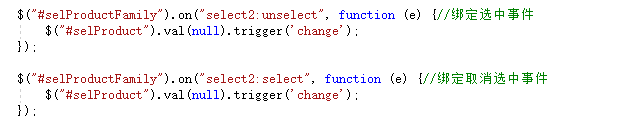


Figure 30控件联动事件

* 1. 必选参数验证
     1. Html

可以给控件对应的label中添加以下元素，实现控件名称前添加红色“\*”,表示必输栏位。



Figure 31 必输栏位标志

* + 1. 后台验证

在Business中添加判断必选参数是否为空的逻辑，如果为空，则如下图抛出ParameterRequiredException.

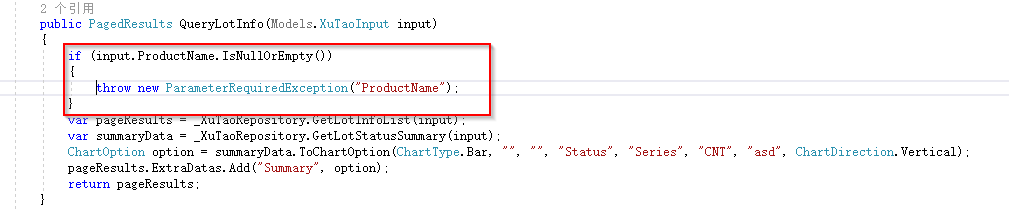


Figure 32必输参数后台判断

* + 1. 前台验证

在页面JS中，添加如下方法：

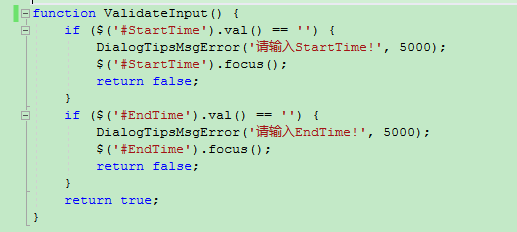


Figure 33 必输参数前端台判断

* 1. 复写方法
     1. 查询

1. 所有grid需要添加“grid”class，以便在点击按钮后，底层获取到所有的grid并执行reload事件；
2. 所有的chart需要添加“chart”class，以便在界面初始化时，底层进行chart初始化，同时在获取chart控件进行赋值时，由之前的

Echarts.init 修改为echarts. getInstanceByDom,如下图所示：



Figure 34 获取echarts实例

1. 目前系统中查询按钮点击之后逻辑顺序如下图所示，其中”ValidateInput”、”BeforeSearchClick”、”AfterSearchClick”为暴露方法，可以在页面js中进行自定义；

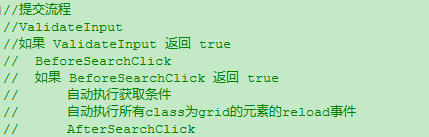


Figure 35 查询逻辑流程

* + 1. 重置

目前系统中重置按钮点击后，逻辑顺序如下图所示，其中ResetBefore、AdditionalReset为暴露方法，可以在页面js中进行自定义

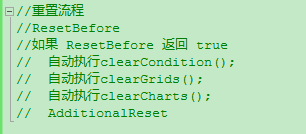


Figure 36 重置逻辑流程

# Windows Service开发

Microsoft Windows 服务(过去称为 NT 服务)允许用户创建可在其自身的 Windows 会话中长时间运行的可执行应用程序。可以跟随系统启动，并在后台运行。多用作系统服务器端承载或需要长期在后台执行的程序。本系统中使用TopShelf开源框架，仅需几部即可快速开发一个新的Windows Service。

* 1. Service创建
     1. HiDM.SMFG.WindowsService.Service

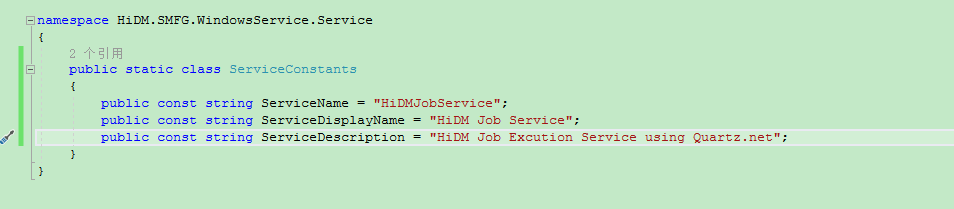
1. 新建Service

在HiDM.SMFG.WindowsService.Service中新建一个Class，继承ServiceControl接口，并实现其Start和Stop方法，将当前Service所需实现的功能代码写在Start方法中，并在Stop中编写，关闭Service时需要执行的逻辑。如JobService中Start时，启动Scheduler，Stop时关闭Scheduler。



1. 修改ServiceName

在ServiceConstants中修改ServiceName/DisplayName/Description



* + 1. HiDM.SMFG.WindowsService

修改Service Class名称，并更新逻辑需要在app.config中添加配置参数；



* 1. 安装及卸载

将所有生成的文件复制到服务器某个目录下，在目录下执行以下bat文件进行安装、启动、停止、卸载。

Install Service.bat：安装并启动服务；

Start Service.bat ：启动服务；

Stop Service.bat ：停止服务；

Uninstall Service.bat ：卸载服务；



# Job创建

本系统使用quartz.net实现任务调度，通过topshelf开发的windows service作为调度承载，并将调度管理继承至报表开发平台，可是对调度任务进行实时管理。Quartz.NET是一个开源的作业调度框架，非常适合在平时的工作中，定时轮询数据库同步，定时邮件通知，定时处理数据等。

Quartz.NET允许开发人员根据时间间隔（或天）来调度作业。它实现了作业和触发器的多对多关系，还能把多个作业与不同的触发器关联。

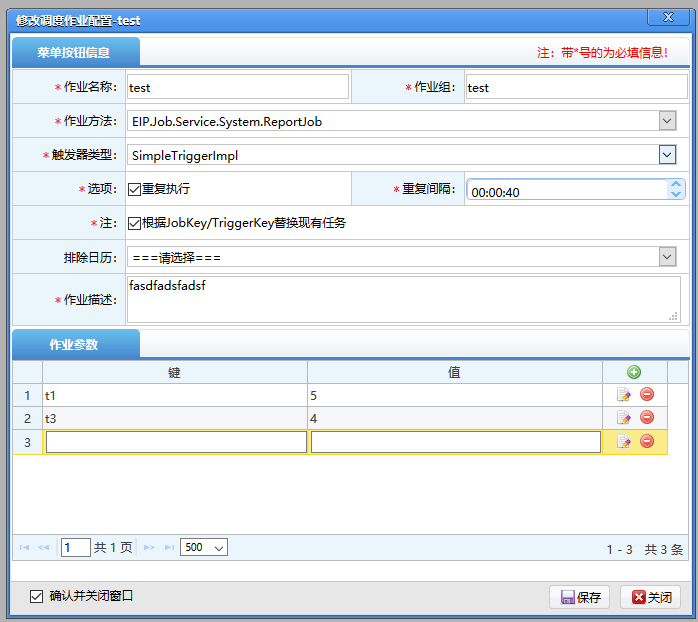
整合了 Quartz.NET的应用程序可以重用来自不同事件的作业，还可以为一个事件组合多个作业。

* 1. Job创建

在EIP.Job.Service中建立Class，并继承接口IJob并实现其Excute方法，编写需要定期执行的业务逻辑，可以通过context中MergedJobDataMap属性获取到配置的参数。

* 1. Job配置

1. Job Class建立完成之后，生成EIP.Job.Service，并将生成的dll复制到报表平台bin目录和Job Windows Service的安装目录下；
2. 登录报表平台，系统管理-作业管理-作业配置管理
3. 点击新增



作业名称：当前作业的名称；

作业组：自定义作业组，分组使用；

作业方法：选择6.1中建立的Class，需要定期执行的逻辑；

触发器类型：

1. SimpleTriggerImpl:简单触发器，需要执行重复执行的时间间隔；
2. CronTriggerImpl:Cron表达式触发器，通过cron表达式指定具体时间点的重复触发，如：每分钟的第几秒、每小时的第几分钟第几秒、每天的某个时间点或者某几个时间点等，无需编写cron表达式，可视化操作；

排除日历：可在日历维护界面先使用cron表达式维护日历，再在此处选择，在排除的日期，job将不执行；

作业参数：当前作业的可配置参数，如6.1中所示，该参数可通过context的MergedJobDataMap属性在代码中获取。

1. 保存，点击保存后，报表平台就会通过TCP连接，将Job信息发送至Job Service进行保存和执行。