|  |
| --- |
|  |
| 问题汇总 |
|  |
|  |
| ©MetaShare Inc. |
| **8/3/2017** |

|  |
| --- |
| 本文eService开发过程中的问题 |

目录

[1. 引言 1](#_Toc489561971)

[2. 第五期问题 1](#_Toc489561972)

[2.1 Service不通过 1](#_Toc489561973)

[2.2 前台时间花费较长 1](#_Toc489561974)

# 引言

Eservice项目从2017-05-11开始对整个架构需要调整，本文包括每天早会后对早会上讨论事情的汇总，包括早会讨论了什么事情，每天做什么改进，接下来要安排什么会议讨论什么事情等等，以供大家查看和改进。

# 指导原则

## 计算方法的开发——鬲文娟

计算方法的开发应该用测试驱动开发

先写计算逻辑一是很多时候想不清楚，再就是写完了并不知道对错。如果一开始侥幸写对了，就会放弃写测试，直接进入应用部分的使用。后面一旦出现问题，就会非常难找。另外code review也很难做。这次的计算是非常简单，但接下来会有更复杂的计算。

## 命名规范问题——鬲文娟

数据库的命名没有研究透彻eservice之前的命名规则，命名出现问题

方法命名和变量命名一定要准确，要让别人读代码时容易联系到它的上下文和目的。除非在操作低层基类和构建通用逻辑时，不许使用抽象名字，如id, element, list。命名时必须考虑上下文因素。大多数的时候，我们是从个例开始构建，逐步抽象或归纳出基类，在重构时，需要根据上下文的变化，更改名字。

## Service的书写规范——鬲文娟

1. 在写Service时没有遵守正确的原则，要先写一个简单的让它工作以后再写复杂的。一层一层从下至上测试跑通，做出一个完整的工作例子，然后再复制，快速把其他实现。
2. 要分清哪些是聚合，哪些是引用，否则会造成死循环的。
3. 在提交前要确认他们的单元测试都是通过的。

## 异常处理——赵宝红

异常处理分为两种情况

1. 一种是不可能到达的情况，需要直接准确抛出异常：

例如方法参数的为空的情况，数组越界等等。

1. 另一种代码逻辑错误引起的异常，需要改正代码，杜绝发生异常的可能：

例如是程序中逻辑中允许有可能有为空的情况，而又没有在判空的情况下使用了空对象而引发的异常，这种是需要代码判空处理的异常，不能直接抛出异常。

针对所有的异常，在开发环境中可以抛出，以便开发人员调试及解决问题。而在生产环境下用户都不希望看到这些异常，所以需要针对各种异常，需要异常捕获，给用户以友好的界面提示，而不是直接抛出异常。

## Dao层的书写规范——林鑫

1. 首次写这类的Dao就参照之前的简单的写了下，不知道写的有不足之处和存在的问题。
2. 在书写Dao时有些细节问题没处理好较为费时，如从数据库取值和赋值的方法对数据的判断校验又不健全或者写的有问题。
3. 在daoTest仿写测试时，发现dao 的书写或者数据库建表字段存在问题，将问题解决后没问题提交。后面有类似复制修改调整后快速实现其他的，字段多的要细心书写。

## 测试数据的书写规范——叶靖

本期任务中，包含大量的测试数据，测试数据之间存在各种关系，在写测试数据时对数据之间的关系分析理解不到位直接进行编写，造成了很多问题。下次再写之前应先分析清楚再进行编写。

## Database Script命名

请清理一下Database Script下的脚本文件，对于现在数据表的修改和新建，可以用表名命名脚本文件，对于其他目的的脚本，命名时写清目的。

# 需求类和管理类问题

## 任务划分不够细——Bella

对testdata/service/dao/database的任务没有区分开，本次实现是需要将页面用Mockservice先展示出来，所以测试数据应该是先写的，应该放在每个任务下面去实现。

## 需求传递不够全面和及时——Bella

有些时候需求更新并整理后未与整个团队沟通达成共识，导致有些人对新需求理解不够。

## 对任务的时间紧迫性要求不够——Bella

对于已经预估时间的任务，应时刻提醒开发员注意时间，并在需求更改后及时调整时间。

## 项目初期的估算——Bella

项目初期可能需求不是很明确，但是从一开始的估算就是一个整体，哪怕客户只有一句话，也要给出基于这一句话的信息的估算。比如，客户问做一财务系统要多少钱？你要根据你的经验给出一个大概的估算。随着需求获取的信息越来越多，整个计划也迭代得越来越清晰，估算会上上下下浮动。随着经验的增加，你对大粒度需求的估算会越来越准确。当有一个需求没很多详细信息的时候，你也要把它作为一个用户故事放在那里，加上一个预估的大小。

## 任务组织不合理——Bella

比如： SY01.UC028 Update Rig Information。 这是关于eService中的Rig information部分的修改，应该只是一个user story，你们把它分割成5个，在理解时造成割裂，而且层次不清楚。合理划分应该是：

实现rig 参考数据(Controller/Gateway/Service/DAO/Database)，这个在Rig Board Specification的文档中第四页最后明确写出，“Derived Master Data User Story: Create Rig Database”在你的backlog，SY05.UC013.US01命名为”Add new rig”，这是完全不对的，造成命名重复，理解混乱。而在这个user story中，没有验证，所以SY05.UC013.US02是不成立的。造成这个混乱的原因是你在建立user story的时候，引用的需求描述是rig board前台的需求。SPB-Product Backlog 和 Burndown chart中的backlog不一致，

1. eService WinForm, 扩展Rig Information 部分。(SY01.028.\*)

2. eService Online – Rig Board– 维护Rig Information

a. 新增rig (SY06.UC013.US01),按你的命名，应该还有(SY05.UC013.US01)

b. 更新rig信息 （SY05.UC032.US01）

c. 改变rig 状态（SY05.UC032.US01）, 由于与各个操作稍有不同，它们可以也是不同的User Story，但在实现时，可以放在一起统一实现。

从这个问题看，你们的backlog管理方法有问题，结构划分本身是错的，把task和user story混为一谈了。User story是当是站在用户观点的需求描述，然后转换成若干系统级的Use Case来实现，再进而根据设计划分成子系统级的Use Case，再进一步划分成组件级的任务，从浅入深，相互关联，有据可查。同时明确任务之件的依赖关系，团队才明确什么先做什么后做。

SubSystem划分也有问题，不能简单划分成前台和后台，因为前台后台定义不明确。应该按OnlineDesignModel文档中的组件来分，而且可能会在组件中再细划分，如ReferenceData。有时间的时候，把eService Express也整合到那个文档中，让团队都理解整体架构。考虑问题的时候，练习全局思维。

## 业务用例的提取——Bella

提取业务用例时要区分，如下面它不是一个业务用例。

BUC.ES.CON.01 Create client consultant database, and hook up with call sheet data entry.

至少可以看做是两组业务用例。

1. Client Consultant reference data management. 下面可以细分CRUD, database, service,等等

2. Rig job client consultant operation,下面可以细分，add client consultant to rig job, remove client consultant from rig job. 问题三中菜单中的第二个菜单项可以去掉。Add and remove组合实现了change consultant.

第二个用例实现依赖于第一个用例实现。在你的分析中，依赖关系应当同时识别出来并记录下来。

## 讨论过程规范化——Bella

对于设计类问题或者一些比较复杂的情况，团队需要坐在一起在白板上画一下，大家brainstorm统一思想，然后拍照片存档，如果有必要需画出设计图上传到SVN。

## 对于开发过程的完善——Bella

1. 对于英文表述的ticket, 将其翻译成中文并调整成为你们理解后的翻译。加到英文之后。如果是因为英文水平的原因，我会发现并指正。
2. 程序员在改动代码前，必须和业务分析员确认需求理解。
3. 程序员在改动代码前，如果有架构上的改动和复杂设计，必须和架构师确认设计。
4. 测试数据的完备性：测试数据需要提前准备。

## 开发质量如何保证

分析原因：

* 人员切换导致；
* 任务没有预估；
* 对任务没有分析，按照自己的理解去做，也没有写TestCase
* 给Adam发邮件应该更清晰，注重表达思路
* 自身主观意识不强

如何提高开发质量：

* 提高C#基本语法知识；
* 严格遵守代码规范，尤其是命名规范；
* 用心去写TestCase，花时间训练
* 根据TestCase互相Review已经完成的任务
* 严格按照开发流程去开发
* 如果有问题及时反馈
* 树立团队荣辱感

需要做什么：

* 刘斌将整个系统串一遍；
* 吴嘉将开发流程细化，用活动图画

# 使用Syncufsion控件——柴志雄

1. 本次前台使用Syncfusion控件，由于对控件的机制不熟悉，导致在使用过程中找不到修改的地方，非常浪费时间。
2. 抽时间对Syncfusion进行研究并给大家做Presentation确保整个团队对Syncfusion常用的控件掌握使用方法。

## 问题汇总

#### 点击Save验证后再次点击重复保存数据

【问题描述】：为保证点击save有效，在页面加载完成后再给save绑定点击事件，使用document绑定，点取消的时候没有清除document绑定事件导致重复提交

【解决方案】：每次点击save前都清除一次document绑定事件

$(document).unbind("click").on('click', "#" + dialogId + "BtnYes", function () {

$(this).attr("disabled", "disabled");

EServiceExpress.Common.Instance.CloseDialog(dialogId);

})

#### 点击Save后多个验证弹框次序错误

【问题描述】：由于是多次弹框同时使用Syncfusion弹框控件，且使用后未删除隐藏弹框，导致

Syncfusion弹框控件误判页面z-index值，出现弹框被覆盖异常

【解决方案】：每次关闭弹框同时删除隐藏弹框

EServiceExpress.Common.prototype.CloseDialog = function (dialogId) {

$("#" + dialogId).ejDialog("close");

$("#" + dialogId + "\_wrapper").remove();

$("#" + dialogId + "Footer").remove();

}

#### 使用Syncfusion数据传递的处理花费时间较长——林鑫

【问题描述】：由于对Syncfusion插件的修改导致原有功能无法将数据传递到后台,从而使操作实际未能成功; ( ProductHaul 新增,修改,删除 操作)

【解决方案】：删除和修改时 将要传递的值先隐藏在表格上,再写script中的方法将选中数据的值传递到后台. 新增 在保存时直接调用前台function方法 异步将数据提交后台, 前台的第一值必须与Model定义属性一样才能保证传递过去的Model数据存在相应值.

<e-columns>

<e-column field="CallSheetId" Visible="false"></e-column>

<e-column field="Company" header-text="Client Company" text-align="Left" width="180" ></e-column>

<e-column field="WellLocation" header-text="Downhole Location" text-align="Left" width="160"></e-column>

</e-columns>

Verify.VerifyDailog.prototype.deleteFun = function (itemUrl) {

$.ajax({

type: 'post',

url: itemUrl,

data: {

'productHaulId': $("#FlatGrid table tr[aria-selected='true'] td").eq(1).text()

},

success: function (data) {

history.go(0);

}

});

}

## Grid控件及使用

首先确保项目中是否引入所有必需的依赖关系包，将必要的脚本和CSS文件从wwwroot - > lib - > syncfusion-javascript文件夹中引用到\_Layout.cshtml页面。

<link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.css" />

<link rel="stylesheet" href="~/css/site.css" />

<link href="~/lib/syncfusion-javascript/Content/ej/web/default-theme/ej.web.all.min.css" rel="stylesheet" />

<script src="~/lib/jquery/dist/jquery.js"></script>

<script src="~/lib/bootstrap/dist/js/bootstrap.js"></script>

<script src="~/js/site.js" asp-append-version="true"></script>

<script src="~/lib/syncfusion-javascript/Scripts/jsrender.min.js"></script>

<script src="~/lib/syncfusion-javascript/Scripts/ej/web/ej.web.all.min.js"></script>

在\_viewImports.cshtml页面中定义以下命名空间

@using Syncfusion.JavaScript

@addTagHelper "\*, Syncfusion.EJ"

在页面定义表格

<ej-grid id="FlatGrid" datasource="ViewBag.DataSource">

<e-columns>

<e-column field="OrderID" header-text="Order ID" ></e-column>

<e-column field="CustomerID" header-text="Customer ID" ></e-column>

<e-column field="EmployeeID" header-text="Employee ID" ></e-column>

<e-column field="Freight" header-text="Freight" ></e-column>

<e-column field="OrderDate" header-text="Order Date" ></e-column>

<e-column field="ShipCity" header-text="Ship City" ></e-column>

</e-columns>

</ej-grid>

#### 自定义工具栏

<e-toolbar-settings show-toolbar="true" toolbar-items="@(new List<string>() {"add", "edit", "delete"})">

</e-toolbar-settings>

#### 控制工具栏功能

<e-edit-settings allow-adding="true" dialog-editor-template-id="#template" edit-mode="dialogTemplate">

</e-edit-settings>

#### 分页

<ej-grid id="FlatGrid" allow-paging="true" > </ej-grid>

#### 调节列宽

<ej-grid id="FlatGrid" allow-resize-to-fit="true" > </ej-grid>

#### 排序

<ej-grid id="FlatGrid" allow-sorting="true" > </ej-grid>

#### 控制行选中

<ej-grid id="FlatGrid" allow-selection="false">

## 鼠标事件

#### 绑定

$("#FlatGrid\_edit").trigger("click");

#### 右击

$(document)

.delegate(obj,

'mousedown',

function(e) {

if (e.which == 3) {

if (typeof callback == 'function') {

callback();

}

}

});

#### 双击

$(document)

.on('dblclick',

"#FlatGrid .e-gridcontent table tr",

function () {

$("#FlatGrid\_edit").trigger("click");

});

## 网页打开新窗口

#### Ajax

1、一定要把 window 放在success外面

2、一定要用同步请求

var result="";

jQuery.ajax({

      "type":"post",

      "url":"http://www.baidu.com",

      "async":"false",

      "success":function(rel){

           if(rel.isSuccess){

   result=rel.url;

               //window.open(rel.url,"\_blank");

    }

     }

});

if(result.length>0){

      window.open(result,"\_blank");

}

#### a标签

<a href='http://www.test.com' target='\_blank' >test</a>

#### javascript

window.open('http://www.test.com',"\_blank");

## 下拉框——刘江波

#### 普通下拉框

#### 带自动联想的下拉框

## Syncfusion控件库问题

目前我们使用的是Syncfusion EJ1，目前EJ2已经发布。需要考虑兼容问题。

Combobox在EJ1中的最新版中已经有了，但在我们引用的版本中没有。所以把我们的引用升级到最新版本应该可以解决这个问题。

现在的问题是eServiceOnline Solution是一个Asp .Net core的工程，但是由于引用到的基于.Net Framework的库，所以它又不是不个纯粹的.Net core应用。交叉引用会带来升级的障碍。目前微软的库结构正在处于进化过程中，依赖非常不稳定，也给我们开发带来了不必要的麻烦。微软引入了一个新产品.Net STANDARD, 它做为.NET core和.NET Framework的共同基类库，这对我们架构的要求也提出了新的挑战，如何正确升级我们的工程？这个问题留给你们架构团队去考虑。

Syncfusion控件不满足需求，应该如何处理？

复查开发过程，总结存在的问题，提出改进措施。

我们处理的流程是：

1. 在Syncfusion官网没有找到满足条件的控件，然后参照了天际项目曾使用过的类似方法实现了这个功能；

以后的改进措施

1. 首先查找Syncfusion官网，如果找不到类似控件，在论坛提问；
2. 在问题回复之后如果有类似控件可以满足需求，那就使用控件；
   1. 如果控件与当前使用的类库一致，使用控件遇到问题再解决；
   2. 如果控件与当前使用的类库不一致，将你需要用的类库引入到项目中；
3. 问题回复之后如果没有类似控件，需要将问题反馈给Adam，经讨论后再决定如何处理；

## EJ1和EJ2兼容性问题

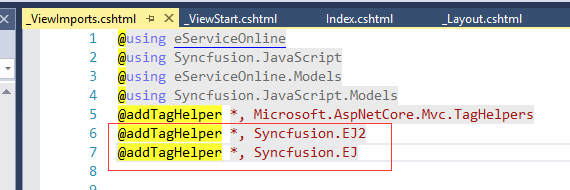
#### 问题描述

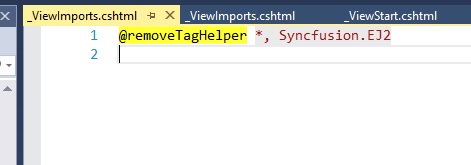
在EService项目中新做的Crew Board页面使用EJ2完成，由于在项目中引入了EJ2的库，导致原来使用EJ1库的页面不能正常运行，是因为同时添加了EJ1和EJ2的TagHelper，程序在运行的时候会默认去使用EJ2的库，使用EJ1的页面就会报错。

#### 解决方案

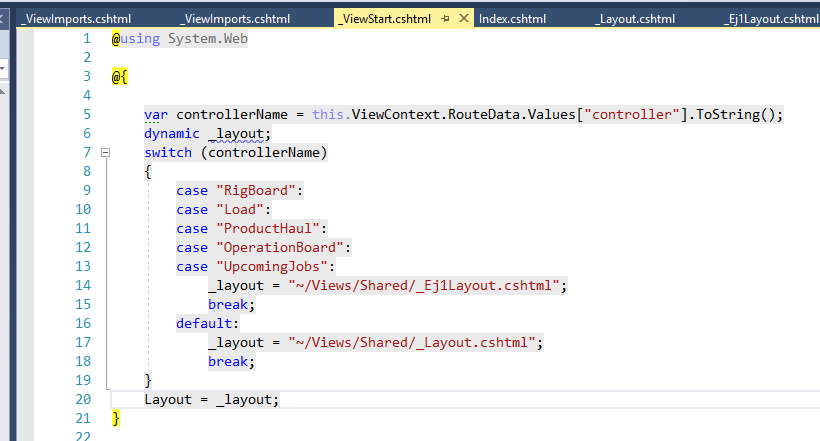
在View根目录下的\_ViewImports中加载EJ和EJ2的TagHelper，然后在每个具体的Controller的View目录下，需要移除不需要的那个版本的TagHelper。

View根目录下的\_ViewImports截图：



Controller下View中的\_ViewImports截图：

需要加载不同的Layout，这个操作在页面加载的时候访问\_ViewStart中去完成Layout的切换。



# 环境问题

## 环境搭建及部署——叶靖

1. IIS组件安装需选择完整，避免引起错误后不好查找引起错误的原因。
2. Oracle数据库客户端未安装，无法配置tnsnames.ora文件，导致无法链接数据库，部署前需检查是否安装。
3. 将.Net Core项目发布到IIS需安装AspNetCoreModule模块，否则无法发布成功。
4. 项目发布到IIS时首先需注意权限问题，不赋予权限，访问应用时会提示HTTP 500.19(0x80070005)，发布时先应确认权限是否存在问题。
5. 确认防火墙配置正确，明确需要用到的端口，配置端口开放策略。

## 代码逻辑问题

#### ProductHaul.CopyFromCallSheetHeader()方法引发严重错误

一个错误发现如下问题：

1)域层次问题。由于域的设计是一个层级结构，每一层对只能对底层有依赖，而不能反向，否则会造成循环依赖。

2)静态方法与属性方法

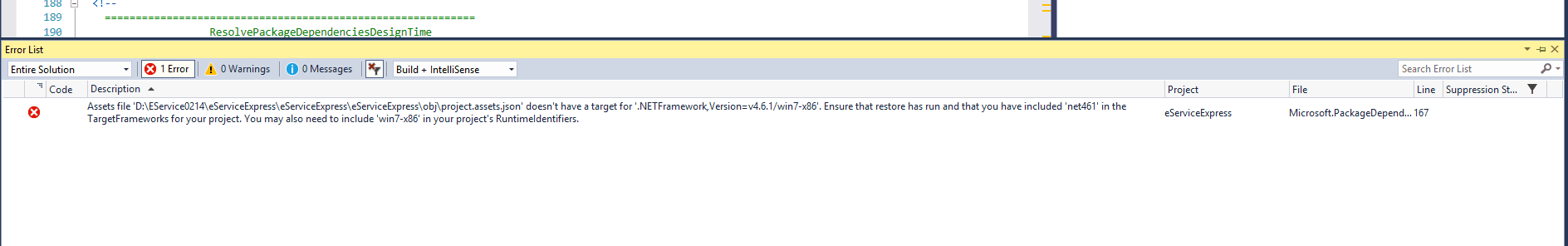
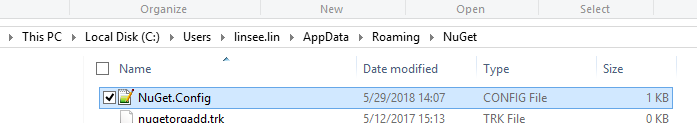
3)值传递与引用传递

#### ProductHaulService.CreateProductHaul()中的检查existingProductHaul的逻辑的问题

ProductHaulService.CreateProductHaul()中的检查existingProductHaul的逻辑的问题，我之前说过。而且逻辑上反反复复地做着没有意义的检查和重新赋值。你们自己检查一下。

## 数据库连接超时 所有基本设置为改变,但连接请求超时,如下图所示: 解决方法: 远程连接到数据库所在的服务器主机,打开服务窗口运行并重新启动oracle相关服务. 按“WIN+R”键，打开运行窗口；输入”services.msc”，回车；

## NuGet 配置有问题导致项目编译失败

  
项目编译失败报以上错误信息提示，Toots -> Nuget Package Manger 中的设置不起作用，在本地电脑找到NuGet.Config文件对其修改。（修改内容是其他人编译没有问题的NuGet.Config文件内容）  


# 设计类问题

## MicroService的设计——林鑫

每一个接口，最重要的是输入和输出。一系列的接口应该是可以连接起来的，每一个接口的输入，要能够为后一个接口的输入提供必要的数据。返回值是一个道理。另外，web程序是stateless的，web api的接口，一般不可能是对象参数。

<http://blog.christianposta.com/microservices/low-risk-monolith-to-microservice-evolution/>

<http://blog.christianposta.com/microservices/low-risk-monolith-to-microservice-evolution-part-ii/>

<http://blog.christianposta.com/microservices/low-risk-monolith-to-microservice-evolution-part-iii/>

### 设计思路

原有的Service 操作的基本都是大对象，不满太适合本次功能的实现使用。因此在大家的讨论下提出MicroService的设计，主要是在本次操作中以当前最小对象，没有调用之前的繁琐。MicroService的方法还是调用EService之前已有或者新增的基础放法，将本次操作的对象最小化。

## RigBoardController的设计——林鑫

今天我发现在RigBoardController，GetStyleCell这个方法，是用硬代码的形式实现前台效果。这个原型不是抛弃型的原型，要按着设计去做，不要写垃圾代码。我们设计得很清楚，实体的状态用前台的样式来表达，在构建原型的时候不是把这个效果显示出来就完了，而是要把实体的状态转换成前台样式。在构建原型的过程中，也是你们理解需求的过程，理解实体的定义和实体之间的关系的过程。

因为这样割裂地去考虑问题，所以就想不到，一个实体的多个状态叠加时的表示，就会有了GetContextMenus方法中Company的ContextMenu对style的依赖。前期讨论中对style的设计完全丢失了，等后面越做越多，会带来极大的重构成本。

另外一个例子，昨天改LSD数据访问逻辑，对有无Downhole Well Location是从字符串进行判断，而没有使用实体中的标志数据。

改进建议：根据真实数据建立RigJobTestData，根据每天分析的深入，不断完善这个测试数据。这样将整个项目分成两部分，一是前台，所有开发基于RigJobTestData，及时发现和反馈不明确的需求和设计。二是后台的数据提取和转换用RigJobTestData做为单元测试的结果部分。

### 设计思路

硬代码的实现效果是没有比较长远的考虑，当时只是先把要展示的先展示大概。对数据的处理也欠缺考虑，经过小葛的重构使得整体框架得以成形。上述遇到的问题主要有：方法实现考虑不足，还有实现时考虑不足只是实现了基本描述没有深入考虑。

## ViewModel的引入——林鑫

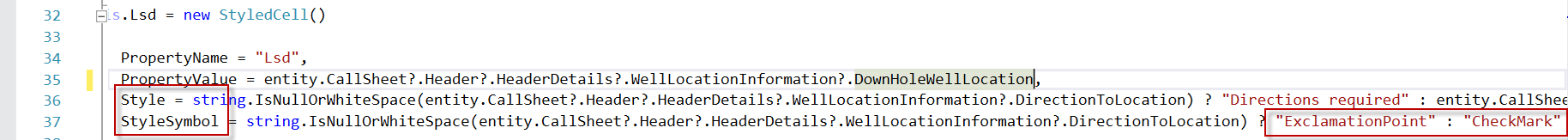
第六期主要实现Rig Board，在这个页面根据不同的属性的值展示不同的样式，以方便用户查看。这里要用到ViewModel。

ViewModel和View的模型应该是等价的，也就是说View中要实现正确表示所需要的所有数据，ViewModel应该是全部都提供的，至少是基本数据可供View中生成派生数据（这就是javascript那个版本）。我们从ViewModel提供样式名这个地方入手，那么ViewModel的每个属型的类型就不应该再是string了，而是一个新的类型包含了样式类的名字。我们把它叫做StyledCell。它包括要显示的值Value, 样式Style，可能还需要保留一个对应业务实体数据属性的名字，因为在前台进行操作的时候，我们需要知道是那个单元上的操作。样式产生的规则应当是一个方法，根据业务数据中状态数据来产生。这样需要一系列从业务实体数据到ViewModel的转换方法，这些方法需要合理的组织以方便维护。在StyledCell类中随着前台需求的不同，也许会增加更多的属性，甚至包括回调的方法。

### 设计思路

ViewModel的使用中，有时候方法名字命名有误导致大家理解有误。对整体布局认识的不够深刻，业务数据的处理做的也不到位。

## 设计问题——林鑫



### 设计思路

在看到Excel表格和要实现的结果，我想的是定义一个存放样式的类在前台展示时使用。名字命名有点不符合规范，还有字符串的使用不当，因该定义为字符串常亮以便其他地方再次使用。

## WebContext的重构——鬲文娟

经过思考与分析，首先online的gateway只是很薄的一层数据访问层，也就是说gateway仅仅负责从微服务取数据，那么我们获取rigJob的相关内容的逻辑，是应该放在rigjob的service还是在online的gateway上再加一层，去处理这个逻辑。

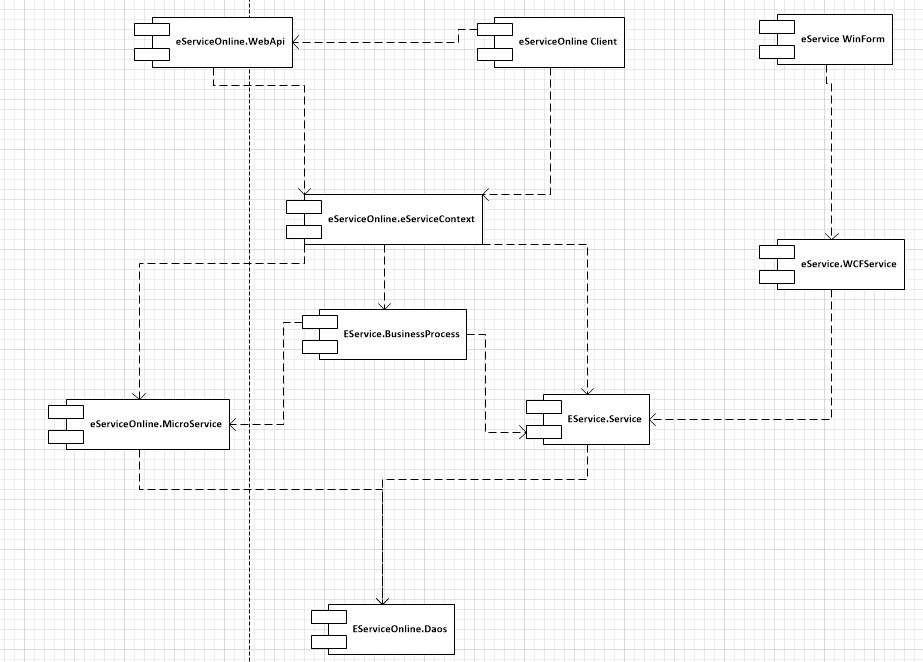
首先，我们要澄清一个问题，这里所说的gateway是个概念上的术语，不是指eServiceOnlineGateway。我们已经看到，使用原来eServiceGateway，可以从开发上减少任务，但是由于它不是为web应用而设计的，所以带来了性能问题。现在你们创建了一个EserviceWebGateway，但好象没有搞明白它和其他组件的关系。实际上它和eServiceWebContext是一回事。如果你们认为它们不是一回事的话，就应当是eServiceWebContext调用EserviceWebGateway， 另外还有其他 的Gateway，在这里决定调哪一个Gateway.

Gateway又名网关，就是门的意思，只是一个通道， 这里没有任何业务逻辑。然后考虑通Gateway传递的消息是什么，这决定了怎么样分布逻辑，逻辑的分布对性能的影响很大。至于获取rigjob的相关内容的逻辑，具体是放在Controller中，还是放service中，甚至放在Dao中，需要根据实际情况去考量和设计。

你们所谓的思考和分析是没有过程的，至少团队要坐在一起在白板上画一下，大家brainstorm统一一下思想，然后拍张照片存档。现在你们没有任何一张图，只是在脑子里想，用嘴说，还说不明白。以后要求，所有设计必须提供白板照片。

### 设计思路

每一层都有存在的价值与意义。



## Chart架构原型——刘斌

第一次做原型的时候，Ajax请求后台获取实时数据，在Ajax里面又套了一层Ajax，用来修改轴的参数，但是Ajax本身就是异步，如果异步里面再套异步，这样会导致不可控制。

解决方法：在一次Ajax中完成获取数据和修改轴参数。

当在编程中遇到ID的时候，一个基本的概念便是它的唯一性， Chart Title如果没有唯一的命名机制，是不能做Id的。

做原型的原则是尽可能的把模块分开，变成更多独立的小的单元来协作解决问题，这样在实际解决方案的实现时才会有更多的选择。

文档中的文字表述问题：在描述轴的参数的时候应该严格遵循Syncfusion本身的描述，因为描述不正确的话，在后面原型复杂的时候可能会导致思维混乱。

### 迭代过程：

先理解需求，然后看着需求把实现过程分成若干小步，对于每一步都需要进行验证，然后有验证结果，迭代过程中遇到的问题需要记录下来，这是对自己积累经验很好的一种方式。对于这样做的好处是如果方向有偏差，可以及时回退，不会导致大面积的Revert。

迭代的过程是不断改进的过程，也就是说，每一步完成都应当是一个持续工作的解决方案。不是每一次都要推倒重来。跳跃性的修改，如果遇到不能解决的问题，很难回退和追踪，只能回滚。

迭代的方法，3F(Focus, Feedback, Fix it)

**Focus**，聚焦，专注。专注于你当前要解决的问题，不要有太多的其他顾虑。把想法付诸实现。

**Feedback**，反馈，抬头看路。审视当前解决方案与目标方向是否有偏差，是否有新的问题产生。实现中是否存在错误。

**Fix it**，修正存在错误。提出进一步要解决的问题，确定下一个小目标，进入下一轮迭代。

### 确认问题或需求：

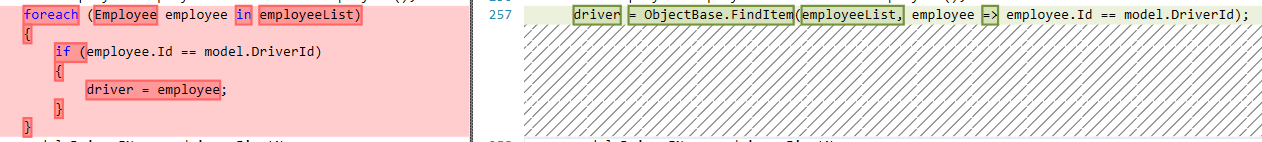
在确认需求或问题的时候，要提出不同的场景或者是解决方案，而不是提出一个非常确定的问题，这样会强制自己开动脑筋去思考，并且客户会根据提出的场景给出合理的反馈。

# 后台代码实现类问题

## 获取集合中指定ID的某一个元素——刘斌

问题描述：通过满足条件的元素

底层已提供方法，所以不需要我们再去写foreach去查询，正确的方式如右图：

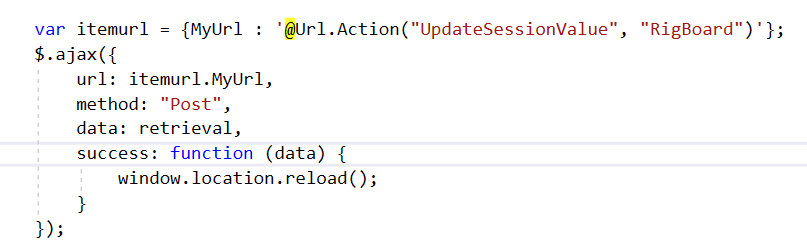


## URL的书写方式——林鑫

### 问题描述

部署后发现service point过滤器不工作，经过检查发现，是javascript中的调用路径不对。这个问题发现过，你们好象没改彻底。我们的部署环境不是在IIS的根目录下，而是在eServiceOnline的子目录下的一个应用。访问路径是 http://hostname/eServiceOnline/RigBoard.

我们找了个方案替换了一下，目前是工作的，但没做彻底检查，你们统一一下方案，全面检查一下。这是在\_Retrieval.cshtml中修改的例子



### 解决方案

在页面script中写URL开始的书写为： Insert/Load直接用的链接地址不符合要求，之后将页面上的URL改为： @(this.Url.Action("Insert", "Load"))。由于script中不支持带参数的C#代码如：@(this.Url.Action("Insert", "Load", new{ id = callSheetId })),所以有的传带参数的URL还拼接如下:

@(this.Url.Action("Index", "Load"))' + "? id =" + $(this).find("td:last").text();

JS中的Url就用传参的方式将URL传递进去。

## 增加新实体或方法——鬲文娟

### 问题描述

当在项目中增加新实体或是新方法的的时候，应当从本身工作域开始，如有必要再移到Common Library里面。不要直接向Common Library直接增加方法，这样会产生不必要的依赖。如ClientCompanyShortName产生的一系列方法。这个实现中有一个假设是CallSheet.CompanyInfoSection数据中保存了ShortName这个字段，实际上是没保存的。接下来的后续后果是，你会去改大量的逻辑来让CallShee. CompanyInfoSection保存这个字段。

### 解决方案

问题的提出，是因为我们在做已有项目的再次开发添加新的功能时，新加的实体和方法应该只是从当前的工作域进行添加域更改，否则可能会违反开闭原则，在开发的过程中发现确实应该放在原有的类里，再进行迁移。

## 大对象复用问题——鬲文娟

### 问题描述

在RigJob实体中引用了两个大对象CompanyInformation和WellLocationInformation，增加了对这两个实体的依赖，而我们需要的并不是这个结构的信息，而只是其中的一部分。读的时候觉着用起来方便，但是写的时候就麻烦了。其实最大的问题是导致你及后来使用它的人造成了概念上的理解错误。在DomainModel里面，RigJob与Job Alert, Call Sheet, Job 三个实体关联， 它相当于是这三个的基类(另一个角度可以说是某一时刻的截图)，Job Alert并没有这个信息结构。

### 解决方案

我们需要的只是一个字符串或其他简单类型去用作显示，不应引用大对象。

## WinForm中控件的逻辑控制——叶靖

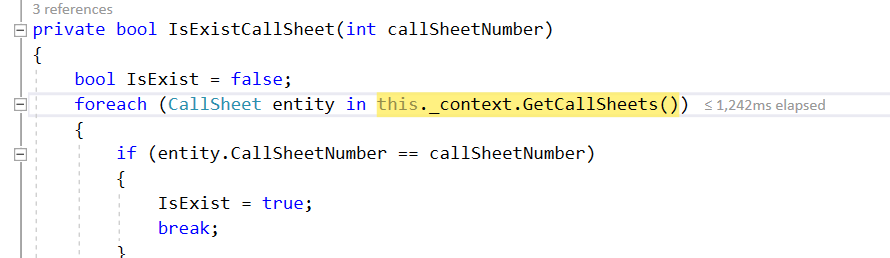
### 问题描述

eService中用户界面上的enable和read-only，标准实现模式是不在控件中对设置，因为很多时候，它的设置是与逻辑相关的。一般是在TypeDescriptor中去设置。这是对前台效果的配置。如果它控制不了的，则用逻辑来控制。今天发现新增加的RigName等控件，调式过程中想修改控制逻辑，结果发现它的readonly设置为true.

### 解决方案

## ProductHaul创建不成功——林鑫

### 问题描述

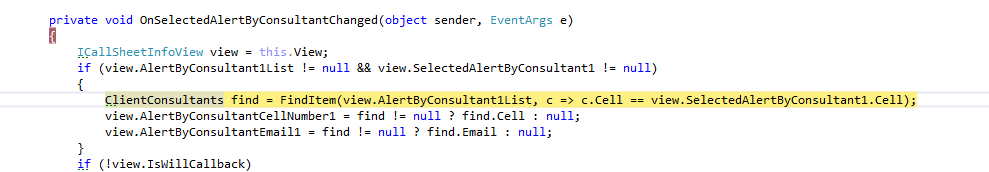


### 解决方案

出现该问题，主要是在从集合遍历时的集合取得不正确导致数据操作有误。  
直接根据参数提供的CallSheetNumber去查询callsheet是否存在，不需要去从集合中判断是否存在。

## 神逻辑——叶靖

### 问题描述



### 解决方案

## ClientConsultants 命名错误——刘斌

### 问题描述

在命名ClientConsultant的时候，之前给命名成ClientConsultants了，用了复数。造成概念混淆，工作不稳定，导致大规模重构。而且重构时，因为不确定哪些应该是Collection，哪些是单个对象，导致大量手工修改。

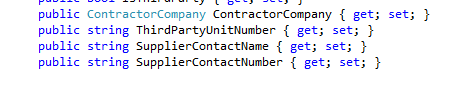
### 解决方案

## 循环依赖问题——刘斌

### 问题描述

在实现Add Third Party Unit in productHaul功能中，在ProductHaul实体中定义了ThirdPartyUnitSection实体属性，造成了循环依赖。

### 解决方案



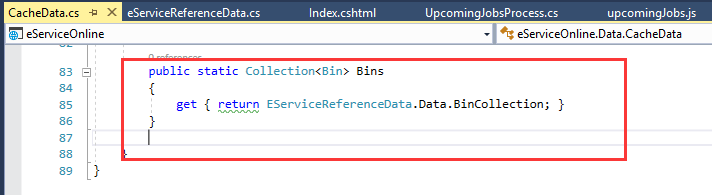
1. 可以在ThirdPartyUnitSection中添加ProductHaulId属性。

## 数据读取问题 —— 林鑫

### 问题描述

在取bin数据时一直参照之前的取数据方式从服务中直接去读取信息,没有利用到CacheData中数据的方式读取数据。

### 解决方案

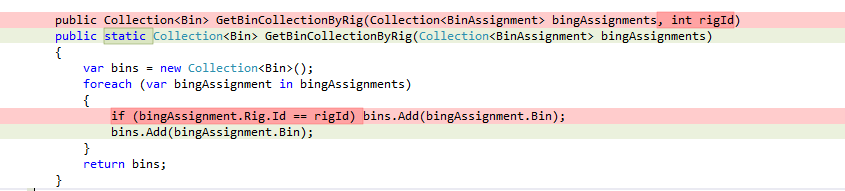
在用到bin参考数据时可以写到CacheData中，直接从其中取有效数据

## bin assignment集合到bin的集合的转换问题——林鑫

### 问题描述

在写转换的时候把部分逻辑代入到底层代码中。

### 解决方案

底层只处理转换不要加逻辑处理，去除之前的逻辑处理。  


## bin assignment 逻辑处理问题——林鑫

### 问题描述

在处理binType哪对应的bin时逻辑过于繁琐,导致程序中存在不必要的数据读取,再加上之前的数据从服务中取还有bin assignment集合到bin 的集合的转换问题使得代码执行效果特别慢.

### 解决方案

简化之前取值逻辑和转化逻辑,写完代码后再找人review一下改代码处理是否得当

## 提交代码问题——林鑫

### 问题描述

提交不必要的文件代码影响其他人员的编译

### 解决方案

在提交代码时注意自己写的测试代码,不要直接提交上去.

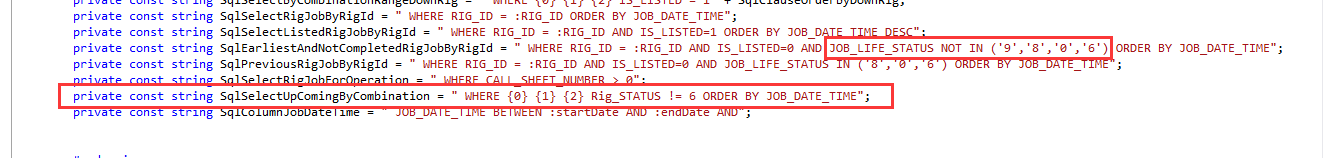
## 在实现接口的时候把逻辑推到的DAO层——林鑫

### 问题描述

再取数据时,将数据取值条件在Sql语句中直接拼死,

### 解决方案

将取值条件当做参数在上层定义传参到DAO层,再进行SQL的组装



## Test Case 书写问题——林鑫

### 问题描述

在书写Test case时情况考虑不够细致,没法用来测试对应的功能.

### 解决方案

在书写前跟其他人员交流,尽可能的将所有可能都考虑到;同时在书写时候也写的细致点,有利于其他人员按照Testcase测试功能.

## 命名问题——林鑫

### 问题描述

在命名时大体定义该名称得法过于笼统,不适合当前方法名称

### 解决方案

更准确细致的描述出当前方法,不要一味参照之前的方法定义方法名称和变量名称

# 前台代码实现类问题

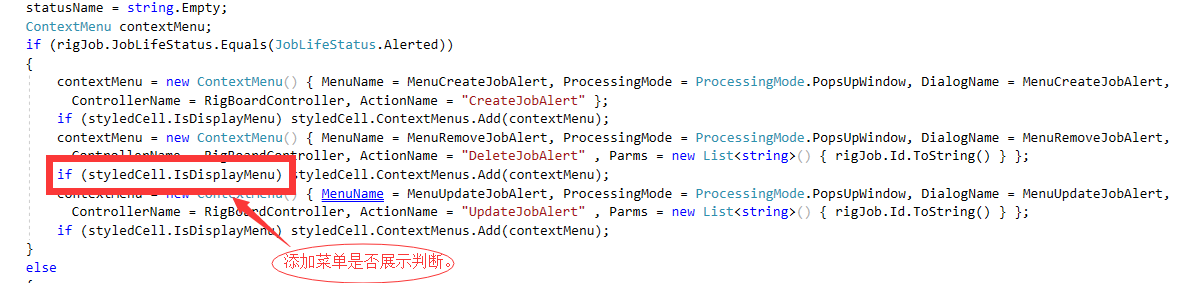
## eServiceOnline 前台菜单展示问题

### 在RigBoard 页面上菜单显示一部分

### 问题描述：

进入Rig Board 页面，对每一列的右击操作菜单，原本不应该显示的菜单右击可以显示出来；

**解决方案：**

之前给Rig Board 页面的没个菜单都加入了权限，所以在新增菜单展示时，也要把权限显示逻辑加上。如下代码所示：  


## eServiceExpress 前台页面布局问题 —— Linsee Lin

### 加载存在Syncfusion控件的局部视图页面

### 问题描述：

在Current Job页面点击菜单加载包含Syncfusion控件的局部视图页面时，没法解析控件导致页面加载失败。  
 **解决方案：** 在要加载包含Syncfusion控件的局部视图页面上添加，下行代码：

@Html.EJ().ScriptManager()

如果该局部视图页面存在script脚本时，不需要使用  
@section Scripts{ }   
包含script脚本，否则脚本不生效；或者将要写的脚本放置要加载局部视图页面的页面上

# 安全控制部分问题

## 获取用户权限问题——刘斌

### 问题描述

使用了SanjelSecurityService中原有的机制读取整个应用的安全规则，造成性能问题。

### 解决方案

利用Dao层原有方法，在MicroService中重新定义接口，获取一个用户的安全数据，解决性能问题。

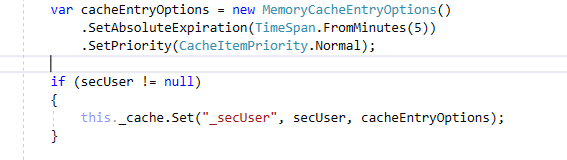
## 缓存用户权限问题——刘斌

### 问题描述

1. 将获取到的用户权限做成静态数据。这样会导致其他用户访问时不重新读取用户权限记录，不支持多用户使用。
2. 将数据保存到Session中，数据过大的话，可能会带来新的性能问题。

### 解决方案

将获取到的用户安全数据缓存到MemoryCache中，并定时刷新缓存的数据，长期不使用的缓存数据定期清理。



## 获取用户名问题——刘斌

### 问题描述



使用上面这种方法获取用户名，开发中可以，但是应用部署后，这种方式只能获取到应用名，导致加入安全控制部署后不能访问应用。

### 解决方案

string loggedUser = htmlHelper.ViewContext.HttpContext.User.Identity.Name.Split('\\')[1];



## Win&lInux 文件路径not found问题——stone

### 问题描述

获取到的filename或filepath 存在’\\’ .linux 不能识别path中的‘\\’，导致文件无法读取

DataCacheBase 类GetDataPath 方法 （第272行）

string dataFileName = GetDataFileName(cacheKey) ;

ReadData 方法（第85行）

string filePath = GetDataFileName(cacheKey);

路径拼接：

GetDataPath (第273行) int lastIndexOf = dataFileName.LastIndexOf(@"\", StringComparison.Ordinal);

GetDataFilePath (第296行) return System.IO.Path.Combine(Path, string.Join(@"\", cacheKey.GetPaths()));

GetBackupDataFilePath(第308行) return System.IO.Path.Combine(buckupCache.Path, string.Join(@"\", cacheKey.GetPaths()));

### 解决方案

Windows 和 linux 都可以识别路径中‘/’因此将 路径中的‘\\’替换为‘/’

修改结果：

string dataFileName = GetDataFileName(cacheKey).Replace('\\', '/');

string filePath = GetDataFileName(cacheKey).Replace('\\', '/');

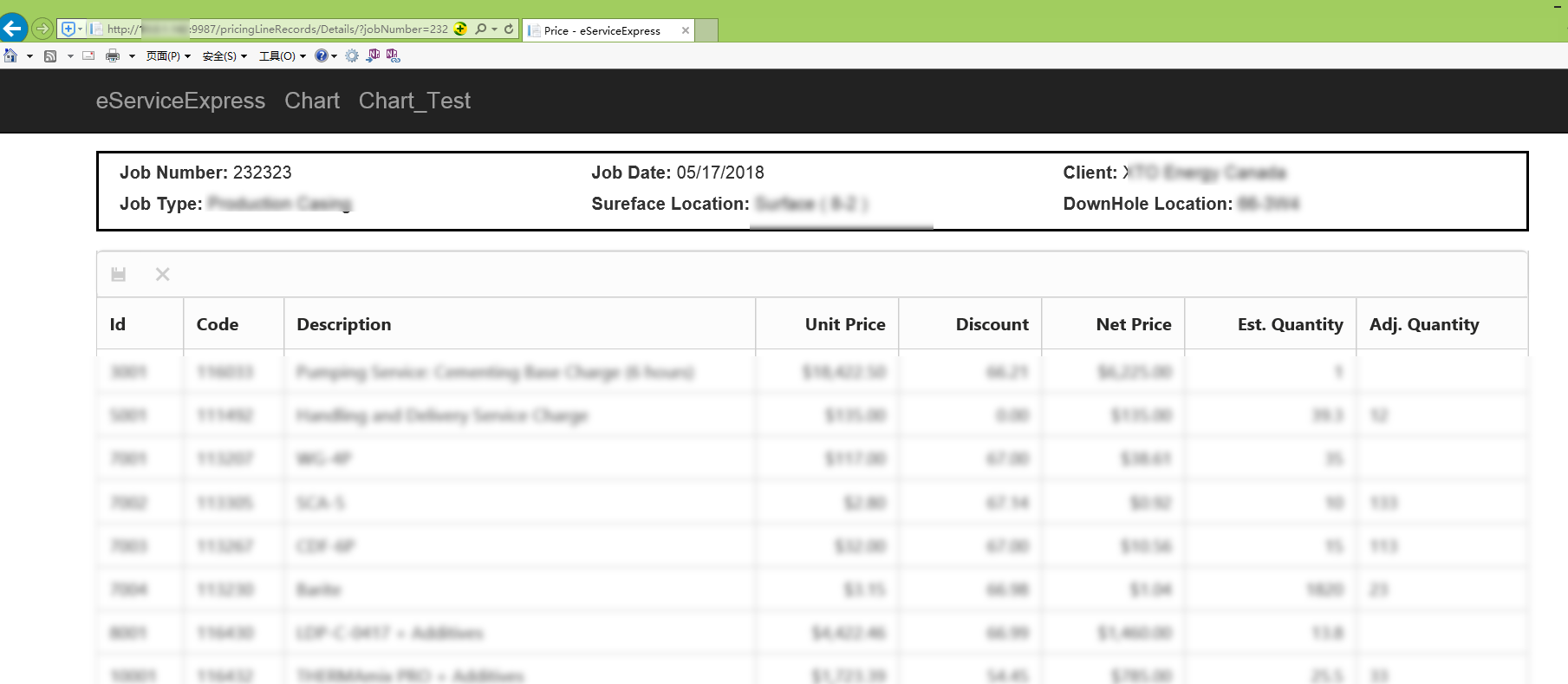
路径拼接使用‘/’

GetDataPath (第273行) int lastIndexOf = dataFileName.LastIndexOf(@"/", StringComparison.Ordinal);

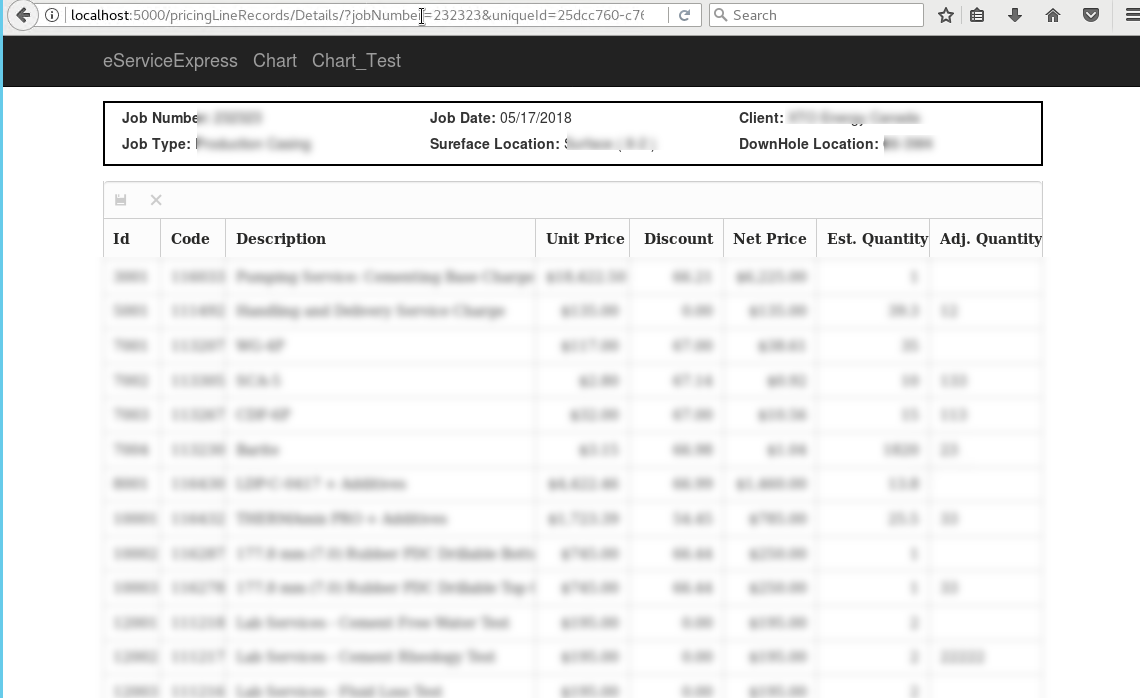
GetDataFilePath (第296行) return System.IO.Path.Combine(Path, string.Join(@"/", cacheKey.GetPaths()));

GetBackupDataFilePath(第308行) return System.IO.Path.Combine(buckupCache.Path, string.Join(@"/", cacheKey.GetPaths()));

本地测试可用 window：



Linux ：

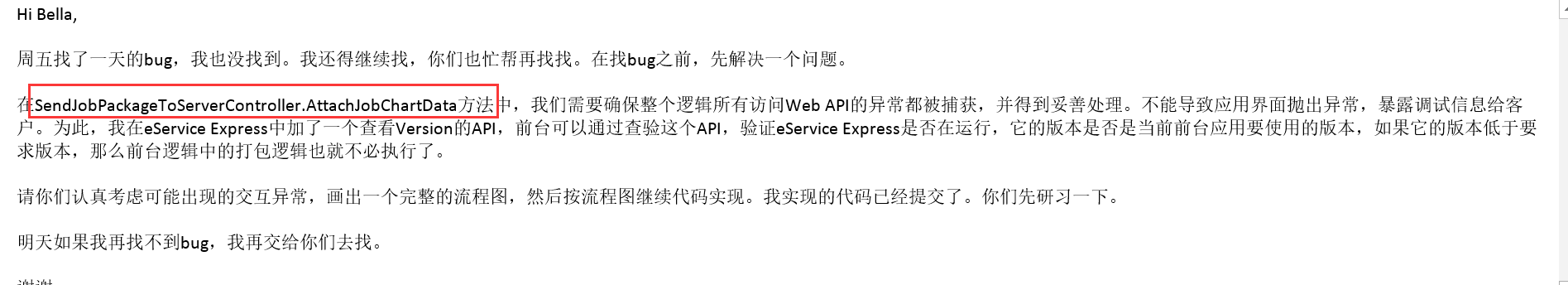


# 开发过程

## 跟客户沟通

### 问题描述

20181211由于Adam的邮件没有仔细阅读，导致做错了任务。邮件中已明确指出需要修改的方法但是Bright和Colin依然改错了。



### 解决方案

对于Adam发来的邮件必须认真阅读，如果由于自身原因导致出现问题责任自己承担，对此我们制定了惩罚策略，每次因为读邮件不仔细已经非常明确到任务却出了差错，责任人每次罚款10元，项目经理因为review不到位每次罚款10元。

## 跟BA沟通

## 测试用例覆盖不全面

# 重构

## 重构ProdcutHaul和ProductHaulLoad

重构中你遇到的第一个问题是，除了实体外，引用Product Haul的程序都不在CommonLibrary中。导致不能通过resharper直接对SanjelEservice中的代码直接在实体变动时进行重构。所以你直接把CommonLibrary中的实体重构了。因为代码同期变动量太小，我无法判断你是通过重构，还是直接拷贝了原来的ProductLoad的代码进行修改，但我猜后者的可能性比较大。因为如果是用工具重构的话，BlendSectionId，AllProductLoadList，  不会同时出现在ProductHaul和ProductHaulLoad中。如果是这样的话，就没有达到练习重构的目的。

重构中常用的技术是通过编译错误进行修正，但首先要保证基础结构进行正确重构。

另外发现的一个问题是CallSheetHeader.CopyFromCallSheetHeader(ProductHaul productHaul, CallSheetHeader callSheetHeader)方法放置的位置不对，尽管它是一个静态方法，但从语义层面它表达了依赖关系，这么做暗含的依赖关系是反的。所以重构时也会给你造成障碍。所以要先把它移到ProductHaul类中去。

昨天通话中的静态模型你没有更新，不知道你是没有提交，还是没有画图。首先是并没有搞清楚哪些属性应该分离出去，没有计划重构的步骤。

从目前代码看， 只把一部分属性移了出去，还剩下一部分仍在ProductHaul里面。我今天试着重新重构了一下，感觉要你接着我的重构继续的话，可能难度更大。所以决定让你继续在你的代码基础上做。步骤如下：

### 属性重构

1. 把没有移出的属性继续移出。
2. 将CallSheetHeader.CopyFromCallSheetHeader(ProductHaul productHaul, CallSheetHeader callSheetHeader)移到ProductHaulLoad中去，参数也要从ProductHaul变成ProductHaulLoad.
3. 把GetProductLoadListFromAll， GetBlendChemicalDictionary改成静态方法，放到ProductLoadSection。
4. Product Haul和RigJobs两个文件夹下面的实体，不应该放在Sanjel.Common.BusinessEntities工程中，应该移到Sanjel.BusinessEnitities工程中。
5. ProductHaulLoad从AggregatedObject继承，以此保留Root ID和Parent ID属性

至此，完成ProductHaulLoad的实体创建和分离。所有测试和eservice online应用测试应该全部通过，且功能不变。

### 根据ProductHaulLoad的更新

1. 完善ProductHaulLoadDao及MicroService，服务方法不要过多设计，能满足本步要求即可。
2. 完善数据库表
3. 完善数据库转换脚本
4. 完善独立测试
5. 重构ProductHaulService中的聚合设置，载入ProductHaulLoad数据。

至此完成ProductHaul与ProductHaulLoad一对一关系的数据实现，所有测试和eservice online应用测试应该全部通过，且功能不变。

### 实现ProductHaul与ProductHaulLoad一对多关系的重构

1. 重构ProductHaul.ProductHaulLoad成为ProductHaul.ProductHaulLoadList
2. 原有前台逻辑中的操作暂用下标0，也就是第一个记录即可

所有测试和eservice online应用测试应该全部通过，且功能不变。

### 前台菜单重构

1. 按需求将菜单从两层结构，改为三层结构。
2. 识别数据读取请求接口，并实现。此时我们的数据仍然是一对一的，显示都应只有一条记录。但是目标是可以显示多条记录。注意请求是递进的，不要一下读出一个大对象出来。

至此，历史数据读取正确。打印正确。

### 创建功能实现

1. 按新需求重构Schedule Product Haul 页面， 正确保存。
2. 实现Schedule Product Haul with Existing Haul,识别新接口并实现，更新相关逻辑。
3. 重构Product Haul Tab 页显示，两级显示改为三级显示。这个用例在前面应该没有识别出来，把它加进去。
4. 完成Re-Schedule, delete schedule，等其他逻辑的实现。

# 打印部分问题

## 打印Chart图表时的数据越界问题——吴嘉

### 问题描述

打印时，如果数据越界，也就是曲线数据出现大于Y轴最大值，或是小于Y轴最小值，超出部分就不打印了。屏幕图表的效果是对的，目前打印是没有处理的，在这种情况下，它会把线打到其他图表上去。

### 解决方案

在ChartModel中添加bool类型的参数IsPrintCrossingPoint（是否打印越界点），来控制打印时是否打印越界的点。可以在应用层Express里通过这个参数来控制是否打印越界的点。当IsPrintCrossingPoint为 true时，不对打印数据做任何处理，如实打印；当IsPrintCrossingPoint为false时，不打印越界的点（处理方式：去掉越界的点，不打印越界的点和相邻的未越界的点之间的连线）。

# 架构部分问题—季少龙

### 扩展API层问题描述

首先，对整个eservice的架构不熟悉，导致在扩展API层时候的问题比较多，eservice整体架构中涉及的设计模式以及反射等等语法比较多，理解起来比较困难。

### 解决方案

整体的设计模式和高级语法以及设计上的问题不是一朝一夕就能掌握的，需要在以后的工作以及工作之后自己投入时间学习去联系，目前扩展出来的架构照猫画虎的模仿以前的架构进行设计的，但是实现的 很简单，比较简单易懂，保证程序可以正常调用API去实现业务逻辑。

# 从架构部分问题扩展出应该加强c#高阶语法以及设计模式学习—季少龙

### 从架构问题引出的问题

对于我们这些初级程序员来说，目前掌握的技能还不是很全面， 只是了解一些基本的语法知识以及浅理解的OOP思想。目前的状态是对高阶语法以及设计模式的理解缺失，导致很多源码阅读起来非常困难，阅读源码不精，就会导致理解起来非常困难，进而影响下一步的开发任务以及后续的开发工作。

### 提高自己的能力 的建议

不管通过什么途径，提高自己的语法基础，不能局限于基本语法之内，要对整体 的语法进行一个全面的认识和理解。

设计模式可以用代码去实现每个模式的Demo以及理解每个设计模式之中的含义以及优缺点。运用起来才可以得心应手。

注：小龙同学的反思很好，这样才能进步。但是对于提出这个问题的角度，和具体的问题，不是描述得很清楚，所以解决方案有些笼统。其实这个问题本身也是相关的。

首先，什么是C#高阶语法？具体指哪些？

其次，设计模式的理解的确很难，特别是eservice 架构中一些机制的实现。这需要时间和经验，遇到读不懂的代码，花些时间去分析，去请教宝红和小鬲，一起分析研究。

第三，其实学习任何事情，都要建立起知识框架，有广度和深度，理解细节的时候，要先理解大的框架，然后再钻研细节，如果一直纠结于细节中，往往会形成一叶障目的局面。

回头说C#语法提高的问题，其实没那么复杂。首先要夯实基础，基本的类型，基本的操作，基本的逻辑，基本的算法，基本的数据结构。实现时基本一点不是问题，比如你只会一种循环的语法就足够了，关键是理解如何去遍历一个集合，什么是结束条件。这是根本。回到最基本的编程教科书， 如果你没读过C 语言程序设计 （The C Programming Language），建议你买一本放在手边，一定要是最经典的版本。

The C Programming Language is a computer programming book written by Brian Kernighan and Dennis Ritchie, the latter of whom originally designed and implemented the language, as well as co-designed the Unix operating system with which development of the language was closely intertwined。

# Calendar上Context Menu上URL路径问题

### 1.1.1 问题描述

在新一期开发中，在实现calendar上context menu时，应为实现机制的改变，改为异步加载菜单时，将菜单的URL在JS文件中写成了绝对路径，导致部署成为子网站时候出现链接找不到的问题。

### 1.1.2 解决方案

重构calendar上的context menu，将url作为参数传入到js方法中，前台只是负责数据的传递参数的角色，将所有的处理都丢给服务器去解决。不仅划分清楚了作用域，也能直观的知道每个作用域的职责。

# 程序运行的基本Context

### 1.1.1 问题描述

在eService Express上做增加Supervisor任务的时候直接从Server端取的数据，忽略了eService Express的基本Context：eService Express是本地的一个脱机服务，与Server没有任何联系。

### 1.1.2 解决方案

需要从OfflineCommonObjectGateway中获取Employee的数据，不与Server产生关系。

反思：一定要清楚eService各个应用之间如何耦合，每个应用的运行Context，什么时候通过Service调用，什么时候通过Web API 调用，什么时候通过WCF Service调用？这样才能避免出现同样的错误。