|  |
| --- |
|  |
| Express Requirements |
| Sanjel项目 |
|  |
| ©MetaShare Inc. |
| **12/3/2018** |

|  |
| --- |
| 本文记录第11期Express项目的需求信息 |

目录

[1 引言 1](#_Toc525124223)

[1.1 背景 1](#_Toc525124224)

[1.2 要求 1](#_Toc525124225)

[1.3 说明 2](#_Toc525124226)

[1.3.1 为什么要使用MDD 2](#_Toc525124227)

[1.3.2 为什么不使用原有的数据结构和项目信息 2](#_Toc525124228)

[2 需求 2](#_Toc525124229)

[2.1 目标 2](#_Toc525124230)

# 引言

## 背景

根据客户反馈继续完善Express的需求。

## 目标

### Master Data Management系统

* 管理Reference Data信息
* 查看Reference Data数据的历史修改记录
* 提供Reference Data的WebApi访问接口

### CommonLibary扩展

本次需要对CommonLibary进行扩展，这次扩展对SanjeleService，SanjelProgramManagement应用程序影响应该不是很大。毕竟只集中在Gateway部分，还有应用程序里的配置文件需要扩展，Web API的注册机制。安全授权部分本期先不考虑。

值得注意的是BlendFluidType和AdditiveType，它们要在新的里面实现，但在应用程序里不使用，程序里使用原来的CommonType的定义。我会关注这个问题，看是否可能把这两个类型转成新定义，但估计影响面会比较大。

1. 本次扩展主要工作是根据原有的架构扩展出调用API的架构层。

### 根据MDM项目修改commonlibrary的实体变动以及一些逻辑的变动

原有的实体被MDM实体替换之后，需要重构commonlibrary中跟以前实体相耦合的业务逻辑以及测试数据等等

### SanjelProgramManagement项目与MDM集成

在集成过程中解决数据的不完整性以及TestData数据与数据库数据不一致问题

# 实现说明

## Eservice Express上加"Pressure Test"

1、在Local Jobs 界面，“Set Current Job" 按钮后面加一个按钮，"Pressure Test"。

2、点击”Pressure Test" 按钮，系统自动产生一个job tag.

Job Number: 自动生成序列号， 格式如下： PT[YY][MM][DD][SN] ， SN 自动增加，从01到99。

Job Date: 点击按钮时获取的当前时间。

Client Company: Sanjel

Job Type：Pressure Test

Job Status: 三个状态：Pending, InProgress, Complete。

Pending: 当Pressure Test Job已生成，但是没有进行Job Set Up 的时候。

In Progress： 当Pressure Test Job已经选好Unit Number，开始时间已经设好，但是没有结束时间。

Complete: 当Pressure Test Job已经设置结束时间。

Surface Location： 计算机名字。

这个job tag只是一个设置标签，它没有eService local job与之相关联，所以它没有job unique id。如果在逻辑中需要unique id, 可以在生成的时候，生成一个unique id。

3、Pressure Test 可以被选中为Current Job。选中时，实时读取SCM\_PLC数据中最后5秒钟的数据，提取其中的Unit Number，实例化Job Monitor Setting和WITS Setting。这与原来从Unit Section 读取的逻辑相同，只是数据源不同。

4、Monitor可以根据Pressure Test Job tag中的设置进行实时绘图。

5、Pressure Test Job，可以对其数据进行打印。表头数据项的数据值待打印出来的，我再改进需求进行调整。

## #258 Update Y Axis label printing format

## #284 WITS数据化格式

本来是想把原来WITSSetting这个类改成为jobWITSSetting类，让这个类去做JobTag中WITSsetting的设置（也就是JobTag表中WITSsetting字段的json）

然后新的WITSsetting这个类来实现数据格式化（实现新的json格式数据）

但是在查看以前的代码时候，思考之后，觉得按上面的方法去改动代码的话会涉及的地方非常多，而且修改起来可能会出问题。为了不能影响其他功能正常运行，

我目前新建了一个ConfigWITSsetting类去把WITSsetting转化为新的数据结构。然后把格式化好的ConfigWITSsetting类当做WITSsetting类的属性在WITSsettingDao层序列化后保存到了Config数据库的Witssetting表的UNIT\_WITS\_SETTING字段中（WITSsetting表是linsee以前做的功能，以前unit\_wits\_setting字段存的json格式和jobtag中WITSsetting字段存的json格式一样）.虽然目前数据格式保存的和需求一样，但是感觉这样实现不太好。

所以你实现后，但是感觉这样实现不太好。 这个感觉是对的。需要你好好学习一下适配器模式。

我在FDAS\_Analysis Document.vsdx文件中加了一页解释适配器模式。记住这是一个思考方式，大到系统之间通讯，小到类之间的协做，方法的复用，都用得到。很多时候，别人的代码你不完全明白的时候，不要轻易去动，特别是交互接口。

具体到这个实现。Config数据库是Eservice Express与其他软件的交互接口，格式是其他软件定义的。当你定义ConfigWITSSetting这个类的时候，实际上是对这个接口进行描述，包括DAO的实现。这一部份与EServiceExpress的JobTag是不在一个域里面的，它们是完全独立的两个系统。是不能混在一起的。它应该是一个独立的project，并且可以独立测试，你可把它认为是其他软件的API库。

然后，在EServiceExpress中去构建适配器将两部分连通。

## 完成了公式部分的UnitNumber实例化代码，引入了Session机制，将JobTag保存在了Session里。

## EService Win Form 里面加一个功能

第一步，点击“Attach File”按钮

第二步，点击“Add”按钮

第三步，选择Type， 在选择项中会加入一个新的选项，“FDAS Data”, 开发时可以用”Chart”暂时代替。选择后， “Browse”按钮Disable。

第四步，点击“Upload”按钮，做以下事情

将SCM\_PLC数据库的自上次数据上传后，到当前的数据，序列化成为JOSN文本, 压缩成ZIP文件，命名为” ”[计算机名字] \_[Job Number]\_SCM\_PLC.zip

按照原有的手工添加FDAS Zip文件的逻辑，添加到Attachment表，自动形成Description，格式为”[计算机名字] FDAS SCM\_PLC Data [起始时间] – [终止时间]”

在Config数据库中建一个数据表Upload\_History， 记录每次打包的时间，Job Number, Job Unique Id, 起始时间，终止时间，这个数据供A步骤中使用。

验证：一个job只能添加一个FDAS Data的记录。

## 增加功能按钮。在Monitor页面，压缩一下，jobinformation部分的宽度，后面加两个按钮。

”Print”, 与Local Jobs页面的打印功能一样。

“End Job”, 自动添充当前时间到End Time的两个框内，并保存，更新页面。相当于，输入EndTime日期、时间并点 击“Update”

## 在eServiceOnline中新增一项主菜单, “Completed Jobs“，其布局与”Upcoming Jobs”类似，但是数据源是Job Package。状态是Approved。显示数据列包括Client, Well Location, Rig, Job Type, Job Date。

## 在过滤条件区，加Service Point过滤器，同Upcoming Jobs，再加一个下拉框”Client”。对数据进行组合条件过滤

## # 画 online中的权限实现流程图 以及完善online的架构图

## Scheduler控件的研究

在完成今天发送的任务以后，开始对Syncfusion的Scheduler控件进行研究。在eServiceOnline中有OperationBoard这个功能，但目前没有用。它之前实现了一个简单的Scheduler的显示。我们下一步要在Scheduler上做更复杂的功能。但不是Scheduler中已经有的那些功能。对Scheduler只要掌握它的Appointment的工作原理和它的基本属性。然后研究ContextMenu，它的实现机制应当与目前RigBoard上面的ContextMenu基本一致，如果不一致，需要提出存在的问题。

## 按着eservice架构设计，实现架构原型。把各层接口都实现出来。架构原型实际上就是根据控件的操作接口，反向推演出各层抽象的接口、所需实体的设计。

应用场景：在Calendar控件上对Employee的工作任务进行预订（Schedule)、改期(Re-Schedule)、取消(Cancel)

在此基础上，实现一个团队（Crew）的实体，在Calendar控件上对Crew的工作任务进行预订（Schedule)、改期(Re-Schedule)、取消(Cancel)。

eService Online中有Operation Board，它是一个Calendar Control，它应该是你原型的起点。

对Scheduler只要掌握它的Appointment的工作原理和它的基本属性。然后研究ContextMenu。

按着eservice架构设计，实现架构原型。把各层接口都实现出来。架构原型实际上就是根据控件的操作接口，反向推演出各层抽象的接口、所需实体的设计。

结合应用场景实现Employee Scheduler和Crew Scheduler。是要你把Employee和Crew的实体关系识别出来，进而将它们的Schedule的关系识别出来。并通过调度crew而产生相应的employee的调度。

## 以前功能的修改

1. # Current job –>Monitor中，job start time, job end time两个时间日期相差较大（2018-6-28 10:50——2018-08-08 15:00）系统直接报内存溢出错误了，分析原因是时间周期过长一次获取的点数过多导致的，实时绘图时是不会存在这种场景的

1. # X轴标签显示没有问题，是Interval没有同步更新的问题，当把起始时间都设置了，Duration自动更新了，但Interval没有按比例放大。
2. # 每次在Monitor页面选中或取消选中一个chart或曲线，要保存到Job Tag中，这样下次重新加载时和上次结束时的结果是一样的。同时打印也是按着选择来的。
3. # 各个chart的Y轴要对齐。因为Y轴的标签长度不一，导致了Y轴的不整齐。把Y轴标签按最宽的标签预留出来，标签右对齐。这样Y轴就对齐了
4. # 由于图表勾选框部分的先择比较多，以后可能会更多，需要增加折叠功能。在这部分上面加一个标题 “Chart Selection”, 最右侧加上一个按钮，展开时，显示减号按钮；折叠时显示加号按钮
5. # 目前，我们用不到“Job Information”,“Event Log”和”wrap up”这三个菜单。暂时把它隐藏起来
6. # local jobs页面不设置 Current job 直接点击到current job页面 会抛出异常
7. # 现在不设置endTime JS会直接报错
8. # 当Job End Time的日期被清楚，时间置为00：00（因为它不能被清除），这个时候保存的数据应该是DateTime.MinValue.
9. # CurrentJob页面，左侧菜单中的”Job Set Up”按钮有bug，只有鼠标移到字上才可以点击。应该是整个按钮都可以。
10. # Current Job页面， “Job Set Up”页面有如下几个问题：

a. 加载速度过慢。从加载过程中视觉上看，是先加载了一遍，然后又加载的状态。

b. 右侧的几个区块，高度定义方式有问题，好象是固定最大高度，这样如果内容多了就会溢出。

在测试用机上，会发现最后Pressure的Radio按钮是点不了的，在开发机上的高分辨率显示器上，都没有问题。可是到了1024X768的工业用机上，就不行了。这是个强约束。我们又在回头讲约束的问题。这些区域应该是动态增长的。如果需要，Units中可以显示横向三组。也就是一行就行了。我们现在设计的最大处理能力是三个unit.

c. Units中的IP Address去掉不要了。

# Job Tag在第一次初始化时，需要做如下默认设置。

d. Units中的显示项，不显示所有的Units,需要过滤一下，过滤条件是Unit\_Sub\_Type\_Id是（70， 71， 238）

e. 过滤出的Units全部默认勾选。

f. WITS中，所 有的RATE 1，全部勾选。第一个Unit的Density和Pressure选中。

g. 初始化JobMonitorSetting时，各条曲线的颜色要有曲别。如Rate图表中，如果有三个unit，那么会有四条线，一个Total Rate和三个unit rate。主基调是蓝色，但是要自动为它们调整一个色系不同的深度。

h. Rate图标的Y Axis的设置动态初始化。1 truck unit 最大值1.5; 2 truck units 最大值3; 3 truck truck units 最大值4

i. Density 最大值2500

k. 切记，这些只是第一次初始化的设置。一旦保存后，以后以保存的设置为准。

1. # Density-Temp图表中，第二个Y轴不显示刻度线，现在两个Y轴的刻度线都显示，太乱了。
2. # Monitor页面中的Interval自动计算，不允许修改。大致比例按12：1，向整数据靠齐。初始Duration 60分钟，interval为5分钟。如果Duration是120分钟，则Interval是10分钟。如果最后Job 结束时间是149分钟，就近取齐，Interval还是10分钟。如果最后Job 结束时间是179分钟，就近取齐，Interval就是15分钟。大概是这样一个算法。
3. # Monitor界面，增加两个快捷键。”\_/-” 键，收起Chart Selection，”+/=”键，展开Chart Selection。只需要按那个键，不需要按Shift键。
4. # Monitor页面，实现监控。

当Duration, interval的自动调整。每当画展线满了的时候，Duration要自动加倍。Interval是通过算法固定下来，用户不能调整Invterval。之前Linsee简单粗暴地改过一次，根本就不对，我给回滚了，改了一下。但是现在有时并不工作。比如说手动将Druation设置为20分种，Interval自动调整为5分钟（Interval的计算由一个单独方法进行，如果以后要调整这个算法，只改这一个方法就行），线画满以后，不能自动调整。

如果Duration初始设置是60分钟，图形自动调整好像是工作的，但是上面的设置框里并没有被更新。

如果设置框被更新了，设置变化应当自动保存在数据库中。

1. # 启动Data Generator, 模拟两个unit。然后点击”Pressure Test”, 进入Job Set Up界面，可以看到两个unit。勾掉一个，保存后进入monitor界面，再回去，发现被勾掉的一个不见了。正常job 没有问题。
2. # 在画线部分，由于时间截取的误差导致有的时候1秒钟之间没有数据，线上会出线一个断点，但可能只是晚千分之三秒， 如图所示。你们看一下是否有更好的改进方法保证连续。
3. # 在EService Express中Monitor页面中，Chart Selection的右侧，有一个”-“(减号)。点击它，Chart Selection会隐藏起来。隐藏后，减号图标应变成加号图标，再点击，Chart Selection会显示出来。
4. 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。
5. Chart打印的Legend标签位置有问题。现在Legend标签是打印在Chart的下面，用户要求打在Chart上面。我改了一下xml文件中的type，”Bottom” 改为”Top“.  结果标签打印不出来。
6. 改成每个图表上只打一条曲线。如RATE图表上有445058 RATE和Total Rate两条线，原来是两条线上打到一起和显示是一样的，现在要求分开打。至于打还是不打，按JobTag中保存的设置决定。
7. 增加功能按钮。在Monitor页面，”Print”, 与Local Jobs页面的打印功能一样。
8. 打印图表，表格线最右侧要封口，不能开口
9. 打印需要时间相对会较长。现在点击打印按钮，没有任何反应。即便是打印成功，用户也不知道。做如下改善：点击打印窗口后，弹出一个动图，显示"Printing ...."
10. 最后四个Chart，Temperature和Density是分开的的，所以

打印Density的时候，就不应该打第二个Y轴和刻度线了。

打印 Temperature的时候，应该变成第一个Y轴和其相应的刻度线。