|  |
| --- |
|  |
| Master Data Management |
| Sanjel项目 |
|  |
| ©MetaShare Inc. |
| **9/19/2018** |

|  |
| --- |
| 本文记录Master Data Management项目的需求信息 |

目录

[1 引言 1](#_Toc531613731)

[1.1 背景 1](#_Toc531613732)

[1.2 目标 1](#_Toc531613733)

[1.2.1 Master Data Management系统 1](#_Toc531613734)

[1.2.2 CommonLibary扩展 2](#_Toc531613735)

[1.2.3 根据MDM项目修改commonlibrary的实体变动以及一些逻辑的变动 2](#_Toc531613736)

[1.2.4 SanjelProgramManagement项目与MDM集成 2](#_Toc531613737)

[1.3 说明 2](#_Toc531613738)

[1.3.1 为什么要使用MDD 2](#_Toc531613739)

[1.3.2 为什么不使用原有的数据结构和项目信息 2](#_Toc531613740)

[2 实现说明 3](#_Toc531613741)

[2.1 将部分数据从TestData到MDM实体的表中 3](#_Toc531613742)

[2.2 整理MDM实体关系 3](#_Toc531613743)

[2.3 MDM引用数据整理 3](#_Toc531613744)

[2.4 研究外挂项目 3](#_Toc531613745)

# 引言

## 背景

Master Data Management项目要做的不是切换，而是面向下一代产品的基础工作。目前的WinForm应用程序的架构是十五年前的设计。当时基于WCF的SOA架构是比较先进的。当时由于互联网远没有现在发达，不管是覆盖率，还是网速。所以基于离线应用场景的需求，将参考数据一次性下载到本地，应用程序启动时加全部加载到内存，这个解决方案是合理的。而且还将继续应用在离线客户端。当时只有一个应用即eService，在总体架构上讲，是单体程序。

随着近年来互联网技术的飞速发展，基于RESTful Web API的技术成为主流，客户应用程序也从单体程序转向网页客户端面，移动客户端，对于离线应用场景，.Net Core也提供了自宿主(self-hosted) Web Server的架构。对于我们来说也发展成多种形式的客户端的应用程序体系。做为服务提供层，我们需要一个容易搭建、容易使用的通用接口。所以Web API成为了必然的选择。

基于Web API的数据访问将按需分配(On Demand)，每一个用例所需要的引用数据一般来讲不不会很多。如果象eService这样的单体程序，它的加载是一次性的，与目前从本地加载差距并不会很大。在系统集成方面，我们并不局限于一种方式。

## 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

## 说明

### 为什么要使用MDD

根据Eservices以前的架构，如果我们要开发数据维护界面，需要开发员去一个一个写，这种工作方式简单、重复、低效。而且还会存在人为错误，质量保证问题。充分利用你们公司的优势，自动生成维护界面。同时我们的需求中还有一个很重要的需求，是要保存修改历史，做到真正的版本控制。

### 为什么不使用原有的数据结构和项目信息

扩展WebApi的方式需要对原有系统进行重构，这对原有应用来讲是个系统性的风险。管理难度，风控难度要大得多。因此我们需要重新生成一套系统即 Master Data Management 用于管理Eservice各子系统引用数据信息。各个子系统采用WebApi的方式访问数据。可以参考原有数据结构，但必须要能提供与目前的数据访问接口一致。

# 实现说明

## 将部分数据从TestData到MDM实体的表中

## 整理MDM实体关系

SVN路径：

<https://184.69.193.254:9880/SanjelDocuments/trunk/Projects/D-Design/MDM/Design/CIM--Model.vsdx>

## MDM引用数据整理

一开始是通过CSV文件来整理，

后面为了避免人工导数据引起的问题，决定通过SQL脚本直接导数据。

SVN路径：

[https://184.69.193.254:9880/SanjelDocuments/trunk/Projects/D-Design/MDM/Data Prepareation/Eservice Migrate Data To MDM.sql](https://184.69.193.254:9880/SanjelDocuments/trunk/Projects/D-Design/MDM/Data%20Prepareation/Eservice%20Migrate%20Data%20To%20MDM.sql)

## 研究外挂项目

外挂项目：项目需要的修改在外挂项目上面进行，每次项目的更新，直接用新版本替换掉旧版本，减少工作量，更高效的应对版本更换。

对于MDM外挂项目的研究方案（后面需要继续研究完善）：

SVN路径为：

<https://184.69.193.254:9880/SanjelDocuments/trunk/Projects/D-Design/MDM/MDM外挂项目的研究.docx>