LMES组件开发技术规范

一、编写目的

- 统一技术规范，使应用程序的结构和编码风格标准化。
- 开发强大的功能，一流的代码，优秀的设计，让项目开发速度提升。
- 专注领域复用能力建设，抽象通用能力+开放设计。

二、 开发过程

1、学习基座API

参阅 CMS.Plugin.Sample 示例代码。

2、使用模板进行后端开发

参阅《CMS2.0使用模板进行后端开发.pdf》。

3、如何实现领域驱动设计

参阅《实现领域驱动设计.pdf》，建议默认采用DDD分层架构开发组件。

4、如何进行单元测试

参阅《单元测试技术.pdf》，建议核心业务逻辑编写单元测试，确保代码质量。

三、 技术规范

3.1、组件设计规范

1、所有标准组件开发前，都需要进行方案的设计。

- 设计文档和源码一起提交，受源码管理器托管。
- 基于DDD分层架构的组件需要输出 **领域模型设计文档**。《领域模型设计.pdf》
- 组件设计需要考虑业务场景对组件性能的要求，并在设计文档中体现（**比如分库分表方案**）。《数据库性能优化方案.pdf》
- 组件设计可根据需要选择合理的 **数据存储方案**，提前规划**生产数据**和**配置数据**的分离，以便后续工程备份还原。
- 设计方案需要经过**评审**，已确保方案的合理性。

2、每个组件开发前需要考虑以下三个方面：

- 组件抽象：例如"**CMS.Plugin.OrderManagement.Abstractions**" 组件设计好不好，组件抽象起决定性作用，组件抽象需要保持稳定，避免频繁更新。

  `打包发布为Nuget包。(.nupkg)`，提供`TryAddOrderManagement()`扩展方法给其他组件尝试注册默认实现。（不允许**依赖**其他组件抽象）

- 组件实现：例如"**CMS.Plugin.OrderManagement**"，"**CMS.Plugin.OrderManagement_V2**"，**一个** 组件抽象 可以有 **多个** 组件实现。

`打包发布为CMS插件包（.cmsplugs）`，可以依赖多个组件抽象，当定制需求与标准组件差异过大，无法满足时，**经评估**可Clone**组件实现**代码，定制开发。

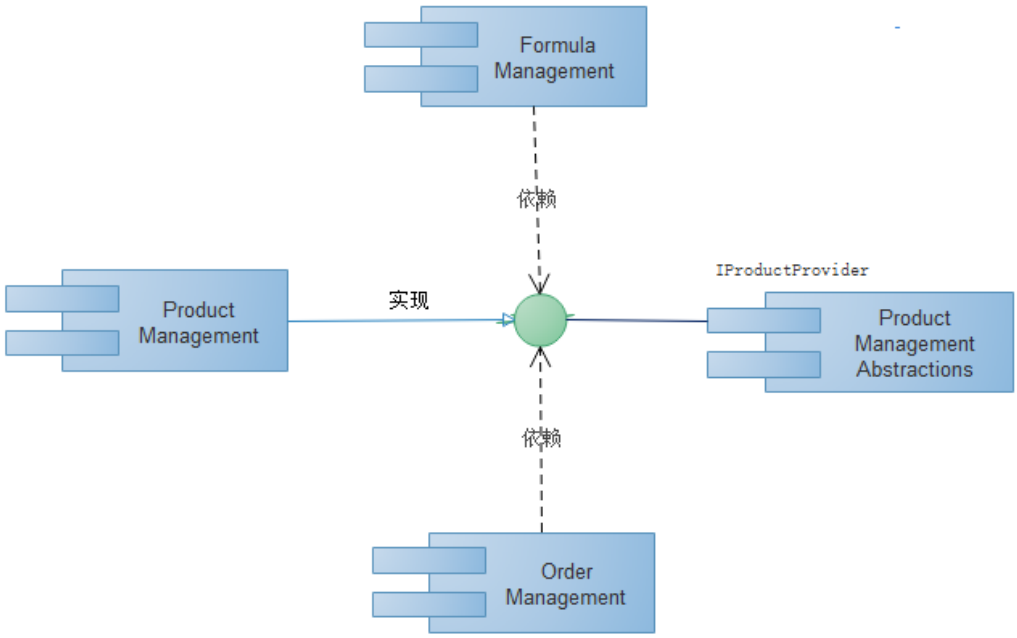
- 组件选项：例如"**OrderManagementOptions**"，一般由 组件名 + Options 命名，用于配置组件运行参数，通过调整参数适配不同业务场景。属于组件抽象的一部分。通常配置选项在 `Startup` 类的 `ConfigureServices` 方法中. 但由于组件已经提供了模块化基础设施,因此可以在组件的 `ConfigureServices` 方法配置选项. 也可以从 `appsettings.json` 文件中读取配置值，例如：

```
{
  "OrderManagementOptions": {
    "ActiveState": "0", // 工单激活状态：0=禁用,1=启用
    "FihishMode": "1" , // 0=人工判断结束,1=满足条件自动结束
    "ProductionMode": "1",  // 0=手动下发生产,1=自动按序生产
  }
}
```

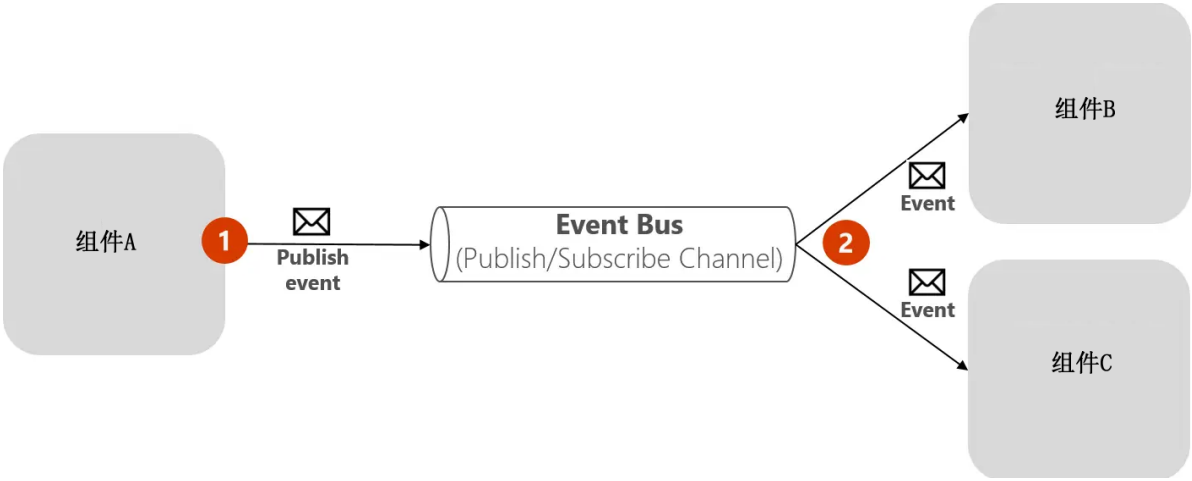
```
public override void ConfigureServices(ServiceConfigurationContext context)
{
    var configuration = context.Services.GetConfiguration();
    Configure<OrderManagementOptions>(configuration);
}
```

3、组件间的相互依赖和解耦

- **建议组件和组件间不要直接依赖**，可通过 **组件抽象** 来封装一系列的对象交互。



- **领域事件和事件总线**： 领域事件是领域模型中非常重要的部分，用来表示领域中发生的事件。一个领域事件将导致进一步的业务操作，有助于形成完整的业务闭环。领域事件主要用于解耦组件，各个组件间不再是强一致性，而是基于事件的最终一致性。事件总线是将消息从发送方传输到接收方的中介. 它在对象,服务和应用程序之间提供了一种松散耦合的通信方式.



- **创建C# API 客户端代理**：通过客户端代理来调用其他组件提供的HTTP服务(REST APIS)，通过这种方式,你不需要通过 `HttpClient` 或者其他低级的HTTP功能调用远程服务并获取数据。注意：此方式适用于对于性能要求不高的场景。快速上手 | WebApiClient

3.2、组件开发规范

- 无特殊情况尽可能的使用**模板**进行后端开发，模板中已经提供了代码分层规范，命名规范等各种示例，参阅示例代码开发。模板已集成ABP 框架 大部分的基础设施，ABP 框架 提供了非常多的开箱即用的功能，开发代码前先阅读Abp文档，不要重复自己一次又一次地实现所有这些常见的东西.专注于你的业务代码,并让ABP按照约定自动执行。
- **数据库设计规范：**主从表命名采用下划线分隔，如：主表：scms_worksections**（工序）**，从表：scms_worksection_processparameters（**工序采集参数，1:n**），以遍快速识别主从关系。
- 前端开发：参阅《前端规范及教程》
- 代码规范与质量：CodeReview是代码质量保障的关键一环，作为CodeReviewer我们要坚守团队的统一规范，严格把控每一份代码中的质量和规范等问题，牢牢的把控好代码质量关口；同时作为被CodeReviewer我们也要尊重别人的时间和意见，共同维护团队的代码规范，从CodeReview中学习别人的意见和设计思想，促进自身的快速成长。《一文浅谈CodeReview中的一些思考》

3.3、外部接口规范

参阅《CMS2.0 外部接口文档》，所有外部接口保持通用的接口信息，比如接口验证方式，编码方式，响应格式。

- **标准套件接口统一URL：** http://**ip:port**/api/v1/**messuite**/external/
- **定制外部接口统一URL：** http://**ip:port**/api/v1/**project**/external/ **project：为对应项目编码**
- **对接第三方外部接口**： 使用统一的 WebApiClient，该组件特性：
  - 支持编译时代理类生成包，提高运行时性能和兼容性
  - 支持 OAuth2 与 token 管理扩展包，方便实现身份认证和授权
  - 支持 Json.Net 扩展包，提供灵活的 Json 序列化和反序列化
  - 支持 JsonRpc 调用扩展包，支持使用 JsonRpc 协议进行远程过程调用
  - 支持将本地或远程 OpenApi 文档解析生成 WebApiClientCore 接口代码的 dotnet tool，简化接口声明的工作量
  - 提供接口声明的语法分析与提示，帮助开发者避免使用不当的语法

四、标准组件

目前LMES标准化组件有如下：

- **工序管理：**创建工序，定义工序功能，配置工序的采集参数、下发参数。

抽象：CMS.Plugin.ProcessManagement.Abstractions

实现：CMS.Plugin.ProcessManagement

选项：**ProcessManagementOptions**

- **工单管理**：工单流程，工单管理，工单记录。

  抽象：CMS.Plugin.OrderManagement.Abstractions

  实现：CMS.Plugin.OrderManagement

  选项：**OrderManagementOptions**

- **产品管理**：创建产品，维护产品对应信息，生产时选择对应产品，调用工艺路线中对应的工序及其对应配方进行生产。

  抽象：CMS.Plugin.OrderManagement.Abstractions

  实现：CMS.Plugin.OrderManagement

  选项：**OrderManagementOptions**

- **物料管理**：BOM管理-以订单维度更新或新增bom，自动更新、手动更新BOM信息。

  抽象：CMS.Plugin.MaterialManagement.Abstractions

  实现：CMS.Plugin.MaterialManagement

  选项：**MaterialManagementOptions**

- **条码管理**：条码管理-激光打码规则

  抽象：CMS.Plugin.BarcodeManagement.Abstractions

  实现：CMS.Plugin.BarcodeManagement

  选项：**BarcodeManagementOptions**

- **配方管理**：创建配方，关联工序，定义具体工序的参数数值，上下限，组合工序形成工艺路线，关联对应产品。

  抽象：CMS.Plugin.FormulaManagement.Abstractions

  实现：CMS.Plugin.FormulaManagement

  选项：**FormulaManagementOptions**

- **追溯管理**：追溯报表，品生产完成，记录生产时的各种生产参数，形成报表进行产品追溯。

  抽象：CMS.Plugin.TraceManagement.Abstractions

  实现：CMS.Plugin.TraceManagement

  选项：**TraceManagementOptions**

- **质量管理**：不合格管理，产品判定，工单质量问题记录。

  抽象：CMS.Plugin.QualityManagement.Abstractions

  实现：CMS.Plugin.QualityManagement

  选项：**QualityManagementOptions**

- **流程管理**：生产过程处理流程，流程的可视化管理。

  抽象：CMS.Plugin.FlowManagement.Abstractions

  实现：CMS.Plugin.FlowManagement

  选项：**FlowManagementOptions**

五、参考文档

- CMS2.0 使用指南：https://docs.syc-cms.com:8443/cms/tutorial/
- CMS2.0 开发文档：https://docs.syc-cms.com:8443/cms/develop/
- CMS2.0 API参考：https://docs.syc-cms.com:8443/cms/api/)
- CMS2.0 Q&A：https://docs.syc-cms.com:8443/cms/qa/