**注意：在阅读该文档前，请确保已经阅读过《附2.EC-Central中BizEntity的5个基础接口的使用说明.docx》，对我们的多语言的业务模式有了认识和了解。**

1. **基本目标**

我们需要实现2个层次的多语言，即软件产品本身的多语言支持以及业务数据的多语言支持；

1. **第一个层次多语言：软件产品本身的多语言支持**

对于此种情况的语言信息，都是作为非业务的上下文环境信息传递的，所以在我们的Service端的类方法设计上，是不会传递这个语言信息的（最直观的就是方法参数上没有该语言信息相关的数据），在Portal端使用时也没有显示的传递该语言信息到服务端；（只有在服务端的帮助工具类的内部实现时可能需要通过当前线程上下文来获取该语言信息以做一些和多语言相关的处理逻辑）；

而该语言信息从Portal端到服务端的过程是由系统框架底层来自动传递和处理的，对开发人员完全透明；

1. 界面控件的文本标签元素的多语言支持
2. Portal端Silverlight的UI框架及框架功能界面的多语言支持

Silverlight的UI框架现在默认支持中文简体、中文繁体和英文三种语言，如果以后要新增新的语言支持，那么需要添加对应语言的资源文件，并且框架代码中某些点的代码需要修改（因为UI框架的代码有地方是写死了当前支持的三种语言的代码）；

1. ECCentral各个功能界面的多语言支持

为不同语言提供相应的resx资源文件来存储对应的文本信息数据，所以界面上的文本标签等元素的文本信息都必须通过资源文件来管理，并且需要支持多语言模式；如果以后要新增新的语言支持，那么需要添加对应语言的资源文件即可；

1. 界面控件绑定的固定数据和配置数据的多语言支持
2. 枚举显示文本

每个Domain在Portal端针对每种语言都有一个专门的固定的resx文件，用来存储所有的枚举所对应的显示文本信息；如果以后要新增新的语言支持，那么需要添加对应语言的资源文件即可；

1. 配置数据显示文本

因为配置数据都放在服务端的，所以Portal都是通过rest service来获取到相关配置数据，在Service端会根据上下文环境中的语言信息来获取相关语言的配置文本数据；这也就要求我们在服务端的配置数据需要支持多语言。当前我们用的CodeNamePair是支持多语言的配置数据的，如果以后要新增新的语言支持，那么需要添加对应语言的CodeNamePair配置文件即可。

1. 提示信息的多语言支持
2. 由Portal端管理的提示信息，那么其文本信息都放在Portal端的resx资源文件里管理即可，同1) 界面控件的文本标签元素的多语言支持的处理；
3. 由服务端管理的提示信息（主要是业务异常提示信息），都是在服务端通过MessageResources进行配置管理的，支持多语言模式；如果以后要新增新的语言支持，那么需要添加对应语言的MessageResources配置文件即可；
4. **第二个层次的多语言：业务数据的多语言支持**
5. 设计基本原则
6. 按照文档《附2.EC-Central中BizEntity的5个基础接口的使用说明.docx》中对ILanguage接口的使用指南来识别出哪些模型需要ILanguage接口，并为相应的BizEntity加上ILanguage的接口；
7. 对于这种业务数据的语言信息，我们都应该当做业务数据来对待和处理，而不应该认为其只是一个上下文环境变量（比如上面的软件产品本身的语言信息），所以：
8. 我们在设计相关的Service和API时，需要把Language作为入参（或者入参实体的属性）
9. 我们在设计UI界面时，在需要多语言的业务数据处理的地方，都需要把Language的选择设计为一个控件元素，可以让用户来操作（阅读或录入等）；（**注：因为当前阶段暂时不用在UI界面级别支持多语言，所以这个控件元素可以先忽略不做**）
10. 具体设计方式
11. Portal UI功能界面上，在需要支持多语言的业务数据的处理功能上，需要提供让用户选择指定语言的能力，比如提供一个语言选择的下拉框，但当前阶段，可以先不这样做；当前阶段定制中蛋默认只有一种语言（zh-CN），所以我们会在Portal端的ECCentral.Portal.Common.ConstValue里定义一个const常量：

/// <summary>

/// 全局的业务多语言代码，由于目前一期不在Portal端实现多语言，

/// 但是Service端的业务模型是支持多语言的，所以Portal端需要传输Biz LanguageCode的地方请取本值。

/// </summary>

public const string BizLanguageCode = "zh-CN";

凡是在有需要传递到服务端Service里来指定业务数据的语言的地方，都统一使用该常量即可；

1. 一个业务模型，如果其部分属性的数据需要多语言支持，部分数据又是有语言无关的，那么此时可有2种做法：
2. 如果需要多语言支持的属性比较多（5个或5个以上）或属性的业务相关性比较强，那么可以把这些属性单独抽取出来，设计为一个单独的类模型，并从ILanguage派生，然后主业务模型聚合该新独立出来的类对象或对象集合（因为多语言了，所以可能变为1对多的聚合方式了）即可；
3. 如果需要多语言支持的属性的业务相关性不强并且数量少于5个，那么可以把这些属性的类型从string改为一个专用的类型LanguageContent；LanguageContent内部就是一个language和string的键值对的集合；
4. 在DataAccess持久化的时候，如果是在Mapping中蛋网的DB结构，那么需要注意2点：
5. 中蛋网的DB结构里，几乎所有的表都添加了LanguageCode的字段，如果假设遇到我们的某个业务模型可能和Language无关的，可以跨Language使用的，那么该模型不需要Language属性，但其所对应的数据库表又有LanguageCode字段，那么在做写入操作时，可以直接给表的LanguageCode字段填入一个默认值（比如空字符串之类的）；在做读操作时则直接忽略数据库表的该字段，不读取即可；
6. 中蛋网的DB结构基本上不支持业务数据多语言模式（如果要支持很多字段之间或表之间可能都要改为1对多的关系了），所以我们在针对业务数据多语言的相关模型的读取或存储的时候，直接mapping到原有DB结构上即可（加载的业务数据的LanguageCode都默认用zh-CN），不用考虑业务数据多语言模式的问题；待以后重构DB真正支持业务数据多语言后，再更新相应的DataAccess的存储mapping关系即可；
7. **对于所发送的业务邮件的多语言处理（程序异常报警邮件等非业务邮件不在此范围内）**

把系统所发送的业务邮件按照收件人可分为2种：

1. **发送给内部运营工作人员的邮件**

针对这种邮件，我们不用专门指定语言编码了，直接由邮件发送模块自己获取当前程序运行线程的上下文的语言信息即可；

1. **发送给外部消费顾客的邮件**

对于这种邮件，我们需要先获取到发送对象Customer的FavoriteLanguageCode（可以通过ICustomerBizInteract接口的方法CustomerBasicInfo GetCustomerBasicInfo(int customerSysNo)来获取一个CustomerBasicInfo的对象实体，取其属性FavoriteLanguageCode即可），然后在邮件发送的接口方法上把该Customer的FavoriteLanguageCode传入以决定使用哪种语言的邮件模板；

1. **对于打印模板的多语言处理**

不用指定语言编码了，直接由打印模块自己获取当前程序运行线程的上下文的语言信息，然后找到对应语言的打印模板文件，读取其内容再打印；