           在Service中，当我们的业务逻辑发现了相关条件不满足等逻辑错误时，一般是需要把这种错误信息返回给UI上，并在UI上展示出来给用户看的，以便用户对操作做出调整。

           首先来看看，作为一个类的方法，如何将异常信息返回给调用者上呢？一般会有2种做法：

1. 使用方法的返回值或者out参数，将异常的code或description返回给调用者
2. 定义相应的Exception类型，在出现错误的地方直接throw这种Exception

           在没有Exception机制的语言里，比如C语言，都是使用的1的方式，通过返回值或out参数来告诉调用者发生了什么，调用者需要自己通过if等语句测试返回值来判断。这样做有几个缺点：首先，增加的条件语句可能会带来更多的错误；其次，条件语句是分支点，会增加测试难度；另外，构造函数没有返回值，返回错误代码是不可能的。另外对于发生异常的方法，方法也需要自己来控制发生异常后的中断返回。如果是多层的嵌套调用，需要把最里面方法的异常返回给最外面的调用者，那需要所有方法都耦合这样的异常处理机制（即在方法声明时就需要考虑异常的处理，使用返回值还是用out参数什么的），给编写代码带来麻烦。

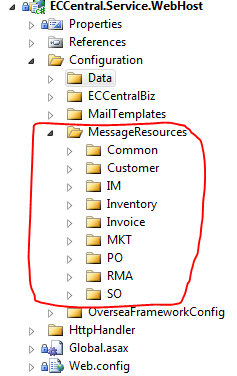
           而在Java、C#等引入了异常机制后，更多的使用throw Exception的方式，这样的方式能够更好的将 异常中断、异常处理和正常的业务代码分离，我们在写方法代码时只需要考虑正常的业务逻辑和算法流程即可，在发生业务异常需要通知外面调用方时，只要throw一个相关的Exception即可，方法执行自然会中断。而只有关心该异常的 外部调用方才Catch就好了，其他嵌套调用的方法都不用关心甚至不用感知和异常有关的任何逻辑。

           而通过不同的异常类型，我们可以表述不同的异常情况。在我们项目的Service端里，会使用第2种抛出异常的方式来将业务异常信息返回给UI端。所以**我们使用了一个专门的Exception类型：ECCentral.BizEntity.BizException来表示业务异常；在Service端抛出该类型的异常，会在客户端Silverlight中弹出友好的业务相关提示，并展示出BizException的Message中的文本，而在Service端抛出其他的Exception则会导致客户端Silverlight中弹出系统Error的警告提示**（现在是展示Exception.ToString()的文本，方便开发人员debug，在后期正式上线后这个可以改为一个统一的服务器错误的提示信息，而不用显示detail的exception信息）。

所以可以看到，我们在Service端的业务异常信息，凡是需要通知到客户端并弹出提示显示出来的，都需要抛出BizException，并将提示的文本放入BizException的Message属性中（通过BizException的构造函数）。

那么这里就会涉及到Message的文本的多语言问题，这里我们开始新的话题：如何在服务端获得当前线程语言的固定message文件信息，需要做2步动作：

1. 事先将所需要的Message文本放到专门的xml配置文件里。配置文件放在Service端的WebHost的Configuration的MessageResources目录下的对应Domain目录里：



（1）。文件命名：[自定义名字].[语言代码].xml，比如：CustomerMaintain.zh-cn.xml，CustomerMaintain.en-us.xml，CustomerMaintain.zh-tw.xml

可以看出，是为每种语言定义一个xml文件，它们的文件名有相同的[自定义名字]，但有各自的[语言代码]

（2）。文件内容的xml格式，在一个xml文件里，Message节点的name不能有相同的（不区分大小写的），它是会在程序里用来定位message的标示之一：

xxx.en-us.xml:

<?xml version="1.0" encoding="utf-8" ?>

<MessageResource>

  <MessageList>

    <Message name="CustomerCreateFailed">Insert Customer Number Failed!(CustomerNumber generated:{0})&#10;Please Have A Check!&#10;</Message>

    <Message name="MemoInsertFailed">Memo Info Insert Failed!{0}&#10;</Message>

    <Message name="PaymentInsertFailed">The payment info insert failed! Excetion message:{0}&#10;</Message>

<Message name="ShippingInsertFaild">The shipping info insert failed! Excetion message:{0}&#10;</Message>

... ...

  </MessageList>

</MessageResource>

xxx.zh-cn.xml:

<?xml version="1.0" encoding="utf-8" ?>

<MessageResource>

  <MessageList>

    <Message name="CustomerCreateFailed">插入用户编号失败!(生成的用户编号:{0})&#10;请做检查!&#10;</Message>

    <Message name="MemoInsertFailed">插入备注失败! 具体异常信息:{0}&#10;</Message>

    <Message name="PaymentInsertFailed">插入支付信息失败! 具体异常信息:{0}&#10;</Message>

<Message name="ShippingInsertFaild">插入收获地址失败! 具体异常信息:{0}&#10;</Message>

... ...

  </MessageList>

</MessageResource>

1. 获取xml配置里的文本内容：

throw new BizException(ResouceManager.GetMessageString("Customer.Point.CustomerPoint", "AddPointFailed"));

使用ECCentral.Service.Utility. ResouceManager类，其有2个public重载方法：

1.

public static string GetMessageString(string resouceFileTitle, string keyName, string inputLanguageCode)

**参数resouceFileTitle**资源文件相对于Configuration/MessageResources目录的相对路径 + 文件名的[自定义名字]部分（注：用“.”来分割路径的目录和文件）;

例如：

放在WebHost的Configuration的MessageResources目录下Customer目录里的CustomerMaintain.zh-cn.xml，那么其resouceFileTitle就为Customer.CustomerMaintain

放在WebHost的Configuration的MessageResources目录下Customer目录下Gift目录里的CustomerGift.zh-cn.xml，那么其resouceFileTitle就为Customer.Gift.CustomerGift

这里不区分大小写，比如Customer.Gift.CustomerGift和 customer.GIFT.CustomerGift是等价一样的效果。

**参数keyName**则是资源文件中Message这个Xml节点的name属性，也不区分大小写。

**参数inputLanguageCode**则是指定资源文件名称中的[语言代码]

2.

public static string GetMessageString(string resouceFileTitle, string keyName)

该重载不用传入语言代码，会自动获取线程上下文的语言代码。

**所以我们在Service端，一般都采用2这个方法重载。**