**EC-Central项目**

**配置文件说明**

Benny.T.Yang

2012-3-30

# 一、 编写目的

本文档的目的是用于指导开发人员了解整个项目中相关配置文件的情况，包括配置文件的作用、位置、规范等，主要从Service端和UI Portal端来分别说明。

# 二、Service端配置文件说明

## 1. 概述

Service端的配置文件全部都放置在ECCentral.Service.WebHost工程下面，主要分为2种配置：

* Web.config中的配置，主要都为基础配置，都和业务无关的
* Configuration目录下的配置，主要都是具体应用配置和业务相关配置

下面将从这2个配置的方面做分别的说明。

## 2. Web.config中的配置说明

总共有10个自定义的配置项，分别为：

1. 定义与SQL及DB相关的配置文件的存放的位置
2. 需要先在<configSections>节点下增加节点：

|  |
| --- |
| <section name="dataAccess" type="ECCentral.Service.Utility.DataAccess.Config.DataAccessSection, ECCentral.Service.Utility.DataAccess"/> |

1. 然后在<configuration>节点下增加节点：

|  |
| --- |
| <dataAccess sqlConfigListFile="Configuration\Data\DbCommandFiles.config" databaseListFile="Configuration\Data\Database.config" /> |

节点说明：

|  |  |  |
| --- | --- | --- |
| 节点 | 属性 | 说明 |
| dataAccess | sqlConfigListFile | 用来指定DataAccess组件中需要用到的SQL脚本配置文件的索引文件的路径；如果该属性没有配置，或者值为空字符串，那么程序会使用默认地址为WebHost根目录下的Configuration\Data\DbCommandFiles.config |
| databaseListFile | 用来指定DataAccess组件中需要的数据库列表（连接字符串列表）配置文件的路径；如果该属性没有配置，或者值为空字符串，那么程序会使用默认地址为WebHost根目录下的Configuration\Data\Database.config |

注：这里的两个路径都支持绝对路径或者相对于WebHost根目录的相对路径的配置；

1. 定义和日志相关的配置
2. 需要先在<configSections>节点下增加节点：

|  |
| --- |
| <section name="log" type="ECCentral.Service.Utility.LogSection, ECCentral.Service.Utility"/> |

1. 然后在<configuration>节点下增加节点：

|  |
| --- |
| <log globalRegionName="EC-Central" localRegionName="Service">  <webservice param="http://localhost:777/Service/Framework/V50/LogService.svc" />  <!--<text param="日志文件的目录，支持相对路径和绝对路径" />-->  <!--<msmq param="消息队列的地址，多个的话用英文逗号隔开" />-->  <!--<custom type="" param="" />-->  </log> |

节点说明：

|  |  |  |
| --- | --- | --- |
| 节点 | 属性 | 说明 |
| log | globalRegionName | 用来指定DataAccess组件中需要用到的SQL脚本配置文件的索引文件的路径 |
| localRegionName | 用来指定DataAccess组件中需要的Database配置列表文件的路径 |
| log/webservice | param | 指定oversea的log webservice的地址（在ECCentral中，这个log webservice架设在portal的host中），支持配置多个地址，用;隔开即可 |
| log/text | param | 添加了text节点就会增加写本地日志功能，param属性用来指定本地存放日志文件的路径，支持绝对路径或者相对于WebHost根目录的相对路径； |
| log/msmq | param | 添加了msmq节点就会增加写日志到MSMQ队列里的功能，param属性用来指定MSMQ地址，支持配置多个地址，用;隔开即可 |
| Log/custom | type | 自定义log处理程序的类型，需要实现接口ECCentral.Service.Utility.ILogEmitter |
| param | 自定义log处理程序的接口方法void Init(string configParam) 的入参，可提供给自定义的初始化log处理对象实例逻辑来使用 |

1. 定义CodeNamePair配置文件的存放的位置
2. 需要先在<configSections>节点下增加节点：

|  |
| --- |
| <section name="codeNamePair" type="ECCentral.Service.Utility.CodeNamePairSection, ECCentral.Service.Utility"/> |

1. 然后在<configuration>节点下增加节点：

|  |
| --- |
| <codeNamePair baseFolder="Configuration\CodeNamePairs">  <appendItems>  <appendItem languageCode="zh-cn" selectAppendItem="--请选择--" allAppendItem="--所有--" />  <appendItem languageCode="zh-tw" selectAppendItem="--請選擇--" allAppendItem="--所有--" />  <appendItem languageCode="en-us" selectAppendItem="--Please choose--" allAppendItem="--All--" />  </appendItems>  </codeNamePair> |

节点说明：

|  |  |  |
| --- | --- | --- |
| 节点 | 属性 | 说明 |
| codeNamePair | baseFolder | 来指定存放CodeNamePair的配置文件的目录的路径，支持绝对路径或者相对于WebHost根目录的相对路径；如果该属性没有配置，或者值为空字符串，那么程序会使用默认地址为WebHost根目录下的Configuration\CodeNamePairs |
| codeNamePair/appendItems/appendItem | languageCode | 该节点用来配置全局的2个CodeNamePair的追加项，该属性用来说明追加的项的文本语言 |
| selectAppendItem | 该属性表示“请选择”的追加项的显示文本 |
| allAppendItem | 该属性表示“选择所有”的追加项的显示文本 |

1. 定义设置网站启动时和网站关闭时自动执行的任务的配置文件的位置

在<appSettings>节点下添加节点：

|  |
| --- |
| <add key="AutorunConfigPath" value="Configuration/Autorun.config"/> |

其中key属性的值是固定的，value属性的值则表示设置网站启动时和网站关闭时自动执行的任务的配置文件路径， 支持绝对路径或相对于WebHost根目录的路径；如果该节点没有配置，或者value属性的值为空字符串，那么程序会使用默认地址为WebHost根目录下的Configuration\Autorun.config；

1. 定义Restful Service的配置文件的位置

在<appSettings>节点下添加节点：

|  |
| --- |
| <add key="RestServiceConfigPath" value="Configuration/RestService.config"/> |

其中key属性的值是固定的，value属性的值则表示设置Restful Service的配置文件的路径， 支持绝对路径或相对于WebHost根目录的路径；如果该节点没有配置，或者value属性的值为空字符串，那么程序会使用默认地址为WebHost根目录下的Configuration\RestService.config；

1. 定义存放应用配置AppSettings的配置文件的目录的位置

在<appSettings>节点下添加节点：

|  |
| --- |
| <add key="AppSettingsPath" value="Configuration/AppSettings"/> |

其中key属性的值是固定的，value属性的值则表示存放应用配置config文件的目录的路径，支持绝对路径或相对于WebHost根目录的路径；如果该节点没有配置，或者value属性的值为空字符串，那么程序会使用默认地址为WebHost根目录下的Configuration\AppSettings；

1. 定义存放异常信息的xml资源文件的目录的位置

在<appSettings>节点下添加节点：

|  |
| --- |
| <add key=" MessageResourcesPath" value="Configuration/ MessageResources"/> |

其中key属性的值是固定的，value属性的值则表示存放异常信息的xml资源文件的目录的路径，支持绝对路径或相对于WebHost根目录的路径；如果该节点没有配置，或者value属性的值为空字符串，那么程序会使用默认地址为WebHost根目录下的Configuration\MessageResources；

1. 定义Web打印配置文件的位置

在<appSettings>节点下添加节点：

|  |
| --- |
| <add key="WebPrintConfigPath" value="Configuration\PrintTemplates\WebPrinter.config"/> |

其中key属性的值是固定的，value属性的值则表示Web打印配置文件的路径，支持绝对路径或相对于WebHost根目录的路径；如果该节点没有配置，或者value属性的值为空字符串，那么程序会使用默认地址为WebHost根目录下的Configuration\PrintTemplates\WebPrinter.config；

1. 定义邮件模板配置文件的位置

在<appSettings>节点下添加节点：

|  |
| --- |
| <add key="MailTemplatesFilesConfigPath" value="Configuration\MailTemplates\MailTemplatesFiles.config"/> |

其中key属性的值是固定的，value属性的值则表示具体邮件模板文件的索引文件路径，支持绝对路径或相对于WebHost根目录的路径；如果该节点没有配置，或者value属性的值为空字符串，那么程序会使用默认地址为WebHost根目录下的Configuration\MailTemplates\MailTemplatesFiles.config；

1. 定义和缓存相关的配置
2. 首先需要先在<configSections>节点下增加节点：

|  |
| --- |
| <section name="cache" type="ECCentral.Service.Utility.CacheSection, ECCentral.Service.Utility"/> |

1. 然后在<configuration>节点下增加节点：

|  |
| --- |
| <cache default="distributed">  <item name="local" type="ECCentral.Service.Utility.LocalMemoryCache, ECCentral.Service.Utility">  <parameters>  <add key="pollingInterval" value="" />  <add key="cacheMemoryLimitMegabytes" value="" />  <add key="physicalMemoryLimitPercentage" value="" />  </parameters>  </item>  <item name="distributed" type="ECCentral.Service.Utility.MemcachedCache, ECCentral.Service.Utility">  <parameters>  <add key="serverList" value="10.16.83.72:11211; 10.16.83.73:11211; 10.16.83.74:11211"/>  <add key="keyPrefix" value="ECCentral\_2012.05.05\_"/>  <add key="maxPoolSize" value="10"/>  <add key="minPoolSize" value="5"/>  <add key="sendReceiveTimeout" value="00:00:02"/> <!-- 2秒 -->  <add key="connectTimeout" value="00:00:02"/> <!-- 2秒 -->  <add key="socketRecycleAge" value="00:30:00"/> <!-- 30分钟 -->  <add key="compressionThreshold" value="131072"/> <!-- 128k -->  <add key="localCacheExpiry" value="00:01:00"/> <!-- 1分钟 -->  <add key="localCacheName" value="local"/>  </parameters>  </item>  </cache> |

节点说明：

|  |  |  |
| --- | --- | --- |
| 节点 | 属性 | 说明 |
| cache | default | 用来指定在程序中通过CacheFactory获取默认Cache时使用的Name |
| log/item | name | 指定该缓存模块的名称，对应到程序中的CacheName，用CacheFactory通过指定该名称即可获取该缓存的Provider |
| type | 指定该缓存的Provider的实现类型的名（AssemblyQualifiedName） |
| log/item/parameters/add | key | 参数名（在缓存Provider初始化时，该参数集会被传入ICache.InitFromConfig方法用于配置初始化信息） |
| value | 参数值 |

通过配置log/item/parameters的参数集合的方式，每个缓存模块的Provider实现类，都可以定制自己的初始化参数；下面分别说明一下LocalMemoryCache和MemcachedCache的配置参数：

|  |  |  |
| --- | --- | --- |
| **LocalMemoryCache** | | |
| 参数名 | 是否必须 | 说明 |
| pollingInterval | 否 | 指定将当前内存使用情况与所配置的绝对最大内存配额限制和内存百分比配额限制进行比较所采用的时间间隔。 该值以“HH:MM:SS”格式输入，默认为2分钟 |
| cacheMemoryLimitMegabytes | 否 | MemoryCache 对象的实例可增长到的最大内存大小（以兆字节为单位）。 默认值是 0，表示默认使用 MemoryCache 类的自动调整大小试探法。 |
| physicalMemoryLimitPercentage | 否 | 缓存可使用的物理内存的百分比。 默认值是 0，表示默认使用 MemoryCache 类的自动调整大小试探法。 |

|  |  |  |
| --- | --- | --- |
| **MemcachedCache** | | |
| 参数名 | 是否必须 | 说明 |
| serverList | **是** | 配置缓存服务器的地址信息，如果有多个缓存服务器，可以用英文半角的逗号或分号隔开；该配置必填，否则程序会抛出异常 |
| keyPrefix | 否 | 设置该缓存Provider将统一在缓存的Key上添加的前缀，虽然该配置是可选配置，但建议还是一定要配置，其可以带来2个作用，一个作用是是可以多个应用共享缓存服务器，只要每个应用都配置了自己的缓存Key前缀，就不会出现Key的冲突；另一个作用是只要修改该配置，就可以直接整体的重置该应用的所有该缓存数据 |
| maxPoolSize | 否 | 因为性能考虑，所以打开与缓存服务器的TCP连接的Socket会缓存，该节点配置最多会缓存多少Socket连接，默认为10；如果同时使用Socket连接超过了该配置数，则多出的Socket连接将会在使用完后立即关闭释放，不会再缓存到缓冲池中 |
| minPoolSize | 否 | 因为性能考虑，所以打开与缓存服务器的TCP连接的Socket会缓存，该节点配置最少会缓存多少Socket连接，默认为5 |
| sendReceiveTimeout | 否 | 通过Scoket向缓存服务器发送或接收数据的超时时间，配置的格式为“HH:MM:SS”，默认为2秒钟 |
| connectTimeout | 否 | 通过Scoket连接缓存服务器的超时时间，配置的格式为“HH:MM:SS”，默认为2秒钟 |
| socketRecycleAge | 否 | 因为性能考虑，所以打开与缓存服务器的TCP连接的Socket会缓存，该配置节点设置Socket缓存的时间，配置的格式为“HH:MM:SS”，默认为30分钟 |
| compressionThreshold | 否 | 对数据将采取压缩的最小字节数，也就是说当缓存的数据超过这个值时，就会先在本地做压缩，然后再发送到缓存服务器，该配置节点默认值为131072（128k） |
| localCacheExpiry | 否 | 本地一级缓存的过期时间，配置的格式为“HH:MM:SS”，默认为1分钟 |
| localCacheName | 否 | 设置本地一级缓存的name（对应到配置中某个item的name），如果没有设置或设置为空，则将不会启用本地一级缓存 |

1. 定义关于数据文件导出的配置文件目录的位置

在<appSettings>节点下添加节点：

|  |
| --- |
| <add key="FileExporterConfigPath" value="Configuration/FileExporter.config"/> |

其中key属性的值是固定的，value属性的值则表示数据文件导出的配置文件的路径，支持绝对路径或相对于WebHost根目录的路径；如果该节点没有配置，或者value属性的值为空字符串，那么程序会使用默认地址为WebHost根目录下的Configuration\FileExporter.config；

1. 定义http上传的文件存放的目录位置

在<appSettings>节点下添加节点：

|  |
| --- |
| <add key="UploadFileBaseFolder" value="UploadFiles"/> |

其中key属性的值是固定的，value属性的值则表示http方式上传的文件所存放的目录路径，支持绝对路径或相对于WebHost根目录的路径；如果该节点没有配置，或者value属性的值为空字符串，那么程序会使用默认地址为WebHost根目录下的UploadFiles；

1. 定义中文分词组件配置文件的位置

在<appSettings>节点下添加节点：

|  |
| --- |
| <add key=" WordSegmentConfigPath" value=" Configuration/KTDictSeg.config"/> |

其中key属性的值是固定的，value属性的值则表示分词组件的配置文件的路径，支持绝对路径或相对于WebHost根目录的路径；如果该节点没有配置，或者value属性的值为空字符串，那么程序会使用默认地址为WebHost根目录下的Configuration/KTDictSeg.config；

1. 定义ObjectFactory注入实现类的配置文件的位置

在<appSettings>节点下添加节点：

|  |
| --- |
| <add key=" TypeVersionConfigFilePath" value="Configuration\TypeVersion.config"/> |

其中key属性的值是固定的，value属性的值则表示ObjectFactory注入实现类的配置文件的路径，支持绝对路径或相对于WebHost根目录的路径；如果该节点没有配置，或者value属性的值为空字符串，那么程序会使用默认地址为WebHost根目录下的Configuration/TypeVersion.config；

1. 定义消息与订阅者关系的配置文件的位置

在<appSettings>节点下添加节点：

|  |
| --- |
| <add key=" EventConsumerConfigFilePath" value="Configuration\ EventConsumer.config"/> |

其中key属性的值是固定的，value属性的值则表示消息与订阅者关系的配置文件的路径，支持绝对路径或相对于WebHost根目录的路径；如果该节点没有配置，或者value属性的值为空字符串，那么程序会使用默认地址为WebHost根目录下的Configuration/ EventConsumer.config；

1. HttpHandler的相关配置

总共需要注册4个HttpHandler，其中有3个是Service端框架的：

FileUpload：公共上传模块的服务端程序，用来接收上传的文件数据，并保存到服务器硬盘上；

HtmlView：用来接收Post过来的文本信息，并直接输出这些文本（其实就是html）到浏览器上；

WebPrinter：公共打印模块的服务端程序，用来创建相应的具体打印工具以及获取数据，最终输出html到浏览器上；

另外1个是第三方的：

ReportViewerWebControl：微软的ReportView Web报表需要的；

为了兼容IIS 7.x的集成模式和IIS 7.x的经典模式（及IIS 6），需要在2个地方分别都注册这4个HttpHandler；（注意每个HttpHandler的Path都不能改变）

(1). 针对IIS 7.x的集成模式

在<system.web>节点下添加如下节点：

|  |
| --- |
| <httpHandlers>  <add path="Reserved.ReportViewerWebControl.axd" verb="\*" type="System.ServiceModel.Activation.HttpHandler, System.ServiceModel.Activation, Version=4.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" validate="false"/>  <add path="FileUpload.ashx" verb="\*" type="ECCentral.Service.Utility.FileUploadHandler, ECCentral.Service.Utility" validate="false"/>  <add path="HtmlView.ashx" verb="\*" type="ECCentral.Service.Utility.HtmlViewHandler, ECCentral.Service.Utility" validate="false"/>  <add path="WebPrinter.ashx" verb="\*" type="ECCentral.Service.Utility.WebPrintHandler, ECCentral.Service.Utility" validate="false"/>  </httpHandlers> |

(2). 针对IIS 7.x的经典模式以及IIS 6.0

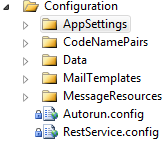
在<system.webServer>节点下添加如下节点：

|  |
| --- |
| <validation validateIntegratedModeConfiguration="false" />  <handlers>  <add name="ReportViewerWebControl" path="Reserved.ReportViewerWebControl.axd" verb="\*" type="System.ServiceModel.Activation.HttpHandler, System.ServiceModel.Activation, Version=4.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" preCondition="integratedMode"/>  <add name="FileUpload" path="FileUpload.ashx" verb="\*" type="ECCentral.Service.Utility.FileUploadHandler, ECCentral.Service.Utility" preCondition="integratedMode"/>  <add name="HtmlView" path="HtmlView.ashx" verb="\*" type="ECCentral.Service.Utility.HtmlViewHandler, ECCentral.Service.Utility" preCondition="integratedMode"/>  <add name="WebPrinter" path="WebPrinter.ashx" verb="\*" type="ECCentral.Service.Utility.WebPrintHandler, ECCentral.Service.Utility" preCondition="integratedMode"/>  </handlers> |

其中节点<validation validateIntegratedModeConfiguration="false" />是用来指定在IIS 7.X的集成模式时，忽略掉上面为兼容IIS 7.x的经典模式以及IIS 6.0而在<system.web>节点里定义的<httpHandlers>节点里的内容；

## 3. Configuration目录下的配置说明

该目录下配置文件的整体结构图如下：



1. 应用设置AppSettings目录

每个Domain在AppSettings目录下都会有个对应的配置文件，命名规则为：[Domain名字].config，文件的xml结构如下：（每个key对应到一个value，在程序中通过key来获取对应的value）

|  |
| --- |
| <?xml version="1.0"?>  <appSettings>  <!--<add key="" value=""/>-->  <add key="MaxPointExpiringDate" value="2"/>  </appSettings> |

具体的使用，请参见VSTS中$\EC\_Products\ECCentral\02\_Design\02\_工具类帮助说明 下的《AppSettings应用配置项的使用说明.docx》文档；

1. 键值对列表设置CodeNamePairs目录

在目录里，每个Domain会有一个自己的键值对数据配置文件。因为需要支持多语言，所以每个Domain的配置文件名的规则为[Domain简名].[语言编码].config，如Customer.zh-cn.config；每个Domain的CodeNamePair的配置针对每一种语言就只有一个文件（所有的键值对列表都放在这个文件里）。文件的xml结构如下：

|  |
| --- |
| <?xml version="1.0"?>  <codeNamePairList>  <codeNamePair key="TimeRange" selectAppendItem="请选择" allAppendItem="全部">  <item code="1" name="上午"/>  <item code="2" name="下午"/>  </codeNamePair>  <codeNamePair key="Compare">  <item code="Greater" name="&gt;"/>  <item code="Equal" name="="/>  <item code="Less" name="&lt;"/>  </codeNamePair>  </codeNamePairList> |

具体的使用，请参见VSTS中$\EC\_Products\ECCentral\02\_Design\02\_工具类帮助说明 下的《CodeNamePair键值对的配置项的使用说明.docx》文档；

1. SQL和DB相关配置Data目录

该目录下的配置分为三种配置文件：

1. 数据库列表（连接字符串列表）配置文件

就一个配置文件，放在Data目录下，包含了所有系统中数据访问层需要用到的SQL Server连接字符串（具体的连接字符串都是加密了的），其文件xml格式如下：

|  |
| --- |
| <?xml version="1.0"?>  <databaseList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://oversea.newegg.com/DatabaseList">  <database name="QueryKeystoneDB">  <connectionString>Hn3t+ao774 =</connectionString>  </database>  <database name="OverseaLocalControlPanel">  <connectionString>Hn3t+ao774=</connectionString>  </database>  </databaseList > |

1. SQL脚本配置文件的索引文件

就一个配置文件，放在Data目录下，包含了所有SQL脚本配置文件的路径，支持绝对路径或者相对于该索引文件所在目录的相对路径，其文件xml格式如下：

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <dataCommandFiles xmlns="http://oversea.newegg.com/DbCommandFiles">  <file name="Customer/CustomerCalling.config" />  <file name="Customer/CallingQuery.config" />  <file name="d://Test/Customer\_Agent.config" />  </dataCommandFiles > |

1. SQL脚本配置文件

在Data目录下，每个Domain会有个一个自己的目录（以Domain名字为目录名），在Domain目录里存放Domain自己需要用到的SQL脚本配置文件，文件数量和文件划分规则没有限制和规则，基本都按照业务模块来分文件，每个文件中配置相关的数据持久化所需要的SQL脚本信息，其文件xml格式如下：

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <dataOperations xmlns="http://oversea.newegg.com/DataOperation">  <dataCommand name="IsAreaNameDuplicate" database="NCService" commandType="Text">  <commandText>  <![CDATA[  SELECT SysNo FROM IPP3.dbo.Area  #StrWhere#  ]]>  </commandText>  </dataCommand>  <dataCommand name="GetAreaOriginStatus" database="NCService" commandType="Text">  <commandText>  <![CDATA[  SELECT [Status]  FROM [IPP3].[dbo].[Area]  WHERE [SysNo]=@SysNo AND [CompanyCode]=@CompanyCode  ]]>  </commandText>  <parameters>  <param name="@SysNo" dbType="Int32"/>  <param name="@CompanyCode" dbType="AnsiStringFixedLength" size="50"/>  </parameters>  </dataCommand>  </dataOperations> |

1. Web打印相关配置和模板存放PrintTemplates目录

和Data目录里的配置很类似，该目录下是两种文件：

1. Web打印的配置文件（包括了打印模板的索引）

就一个配置文件，放在PrintTemplates目录下，包含了打印生成类的配置，以及每个打印需求的数据构建类以及需要用到的打印模板文件的路径，该文件支持绝对路径或者相对于该配置文件所在目录的相对路径（需要支持多语言，所以按语言编码进行分组），其文件xml格式如下：

|  |
| --- |
| <?xml version="1.0"?>  <webPrinter>  <handlerRegisters>  <handler extention="htm, html" type="ECCentral.Service.Utility.HtmlPrintHandler, ECCentral.Service.Utility" />  <handler extention="rdlc" type="ECCentral.Service.Utility.RdlcPrintHandler, ECCentral.Service.Utility" />  <handler extention="rpt" type="ECCentral.Service.Utility.RptPrintHandler, ECCentral.Service.Utility" />  </handlerRegisters>  <printerSettings>  <printer name="PrintRequestLabel" dataBuilder="ECCentral.Service.RMA.AppService.RequestLabelPrintData, ECCentral.Service.RMA.AppService">  <template languageCode="zh-cn" path="RMA/PrintRequestLabel.zh-cn.rdlc" />  <template languageCode="en-us" path="RMA/PrintRequestLabel.en-us.rdlc" />  </printer>  <printer name="PrintRefund" dataBuilder="ECCentral.Service.RMA.AppService.RefundPrintData, ECCentral.Service.RMA.AppService">  <template languageCode="zh-cn" path="RMA/PrintRefund.zh-CN.htm" />  <template languageCode="en-us" path="RMA/PrintRefund.en-us.htm" />  </printer>  </printerSettings>  </webPrinter> |

1. 打印模板文件

把打印模板文件都统一放在PrintTemplates目录下，按照Domain来划分目录，每个Domain一个目录，存放自己需要用到的打印模板；打印模板现在支持html和rdlc两种；

1. 邮件模板配置MailTemplates目录

和PrintTemplates目录里的配置很类似，该目录下是两种文件：

1. 邮件模板文件的索引文件

就一个配置文件，放在MailTemplates目录下，包含了所有邮件模板文件的路径，支持绝对路径或者相对于该索引文件所在目录的相对路径（需要支持多语言，所以按语言编码进行分组），其文件xml格式如下：

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <mailTemplates>  <templateList languageCode="zh-CN">  <template path="Templates/MailTemplate\_GiftCard.zh-CN.config"/>  <template path ="Templates/CustomerGift\_Notify.zh-CN.config"/>  <template path ="Templates/CustomerGift\_Remind.zh-CN.config"/>  <template path ="Templates/CustomerGift\_Void.zh-CN.config"/>  <template path ="Templates/RMAOutBound\_DunMail.zh-CN.config"/>  </templateList>  <templateList languageCode="en-US">  <template path="Templates/MailTemplate\_GiftCard.en-US.config"/>  <template path ="Templates/CustomerGift\_Notify.en-US.config"/>  <template path ="Templates/CustomerGift\_Remind.en-US.config"/>  <template path ="Templates/CustomerGift\_Void.en-US.config"/>  <template path ="Templates/RMAOutBound\_DunMail.en-US.config"/>  </templateList>  </mailTemplates> |

1. 邮件模板文件

放在MailTemplates目录下的Templates目录里，根据业务模块来划分文件，每个模板文件配置某个模块所需要的所有邮件模板信息，其文件xml格式如下：

|  |
| --- |
| <?xml version="1.0"?>  <mailTemplate>  <id>CustomerGift\_Notify</id>  <sender>admin@xxx.cn</sender>  <isHtmlType>1</isHtmlType>  <mailPriority>1</mailPriority>  <subject>  <![CDATA[  获奖通知  ]]>  </subject>  <body>  <template reference="shared/header.htm"/>  <template>  <![CDATA[  <html xmlns="http://www.w3.org/1999/xhtml">……</html>  ]]>  </template>  <template reference="shared/footer.htm"/>  </body>  </mailTemplate> |

1. 服务端异常信息配置MessageResources目录

在该目录下按照Domain来划分目录存放配置文件，每个目录都对应一个Domain，目录名就为Domain名字，每个目录下的具体配置文件的数量和划分没有具体的规则和限制，基本上按照开发人员自己理解的业务模块划分来划分建立文件，其文件xml格式如下：

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <MessageResource>  <MessageList>  <Message name="CSIsNull">请指定要添加的CS用户</Message>  <Message name="ExistsCS">指定用户已存在</Message>  <Message name="CantSerCsSameAsLeader">不能指定该CS同时为组长</Message>  </MessageList>  </MessageResource> |

具体的使用，请参见VSTS中$\EC\_Products\ECCentral\02\_Design\02\_工具类帮助说明 下的ResouceHelper服务端异常信息多语言.docx》文档；

1. 自动运行组件配置Autorun.config文件

直接放在Configuration目录下，用来配置需要在服务端系统启动或关闭时，需要自动运行的一些程序（比如在服务端系统启动时需要做一下初始化准备工作，那么可以做一个实现ECCentral.Service.Utility.IStartup接口的类，并配置到该配置文件中，同理还有个ECCentral.Service.Utility. IShutdown接口），其文件xml格式如下：

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <autorun>  <startup>  <module type="ECCentral.Service.Utility.WCF.RestWebServiceStartup, ECCentral.Service.Utility.WCF">  <constructor>  <arg>ECCentral.BizEntity.BizException, ECCentral.BizEntity</arg>  </constructor>  </module>  <module type="ECCentral.Service.Utility.ResouceManagerAutorun, ECCentral.Service.Utility" />  </startup>  <shutdown>  <module type="ECCentral.Service.Utility.ResouceManagerAutorun, ECCentral.Service.Utility" />  </shutdown>  </autorun> |

1. WCF Restful服务配置RestService.config文件

直接放在Configuration目录下，用来配置需要启动哪些WCF Rest服务的，其文件xml格式如下：

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <serviceList>  <service path="IMService" type="ECCentral.Service.IM.Restful.IMService, ECCentral.Service.IM.Restful" />  <service path="POService" type="ECCentral.Service.PO.Restful.POService, ECCentral.Service.PO.Restful" />  <service path="SOService" type="ECCentral.Service.SO.Restful.SOService, ECCentral.Service.SO.Restful" />  </serviceList> |

1. 数据文件导出配置FileExporter.config文件

直接放在Configuration目录下，用来配置文件导出的处理类的：

|  |
| --- |
| <?xml version="1.0"?>  <fileExporter default="ExportWord" expiry="00:05:00" maxRowCountLimit="10000"  baseFolder="x://tmp\_files/" virtualPath="ExportFiles">  <add name="ExportCustomer" type="ECCentral.Service.Customer.AppService.CustomerExcelFileExporter, ECCentral.Service.Customer.AppService" />  <add name="ExportWord" type="WordExporter.ExporterV1, WordExporter" />  </fileExporter> |

* **根节点fileExporter**，有五个属性：

**default属性：**表示当Portal端在调用导出方法时没有指定exporterName的时候，Service端默认使用的导出实现类型，该属性值只能为下面某个add子节点的name属性的值；如果没有配置该属性或该属性为空，那么默认使用ECCentral.Service.Utility.ExcelFileExporter类型，将导出Excel文件；如果该属性配置的name不为空但又不在其下自己点中存在，那么还是会默认使用ECCentral.Service.Utility.ExcelFileExporter类型来导出Excel文件；

**expiry属性：**设置导出文件保存在服务器端后多长时间就可以删除了，其格式为“HH:mm:ss”；注意该属性只是设置了多长时间后文件可以删除，但不是说到了那个时间后文件就一定会删除；因为删除动作是由每次导出动作触发后才会伴随执行的（并没有背后线程来执行），也就是说如果一直没有导出动作触发，那么文件将一直保留在服务器硬盘上，即使早已过了设定的expiry时间；如果该属性没有设置，或者设置为空或者设置格式不正确，那么都会使用默认5分钟；

**maxRowCountLimit属性**：用来设置一个Sheet里最大允许导出的记录条数，默认为10000；

**baseFolder属性**：用来设置待导出文件存放的目录，支持绝对路径或者是相对于当前WebHost根目录的相对路径；默认使用WebHost根目录下的TempFiles目录；

**virtualPath属性**：用来设置文件存放的目录所映射到的网站http相对路径（相对于WebHost这层的路径）；默认使用WebHost下的TempFiles；

**（baseFolder**和**virtualPath**在大部分情况下都不用配置，但在这样的一种场景可以派上用场：我们的Service是架设了多台服务器，采用load balance，那么我们每个Service的**baseFolder**都指向同一个局域网共享目录，然后在每个Service下都建一个同名的虚拟目录指向这个局域网共享目录，并将**virtualPath**设置为该虚拟目录的名字**）**

* **fileExporter下面的add子节点**就用来配置自定义的数据文件导出类型，name属性的值需要唯一，不能有重复，否则会报配置异常，Portal端的导出方法上的exportName参数就是对应到这里配置的name属性，用来告诉Service，应该使用哪个导出类来负责处理文件导出工作；

1. 分词组件配置KTDictSeg.config文件

用来设置分词组件的属性的值，具体可参加配置文件中的说明；

1. 分词组件的字典 SegmentDict目录

存放分词组件的字典数据；

1. ObjectFactory注入实现类的配置TypeVersion.config文件

用来为ObjectFactory定义相应的接口或基类所使用的实现类的版本；

|  |
| --- |
| <?xml version="1.0"?>  <typeVersionMaps globalDefaultVersion="1.0" assemblyFolder="">  <map type="ECCentral.Service.Invoice.BizProcessor.PostPayProcessor, ECCentral.Service.Invoice.BizProcessor" version="2.0.0.0" />  <map type=" ECCentral.Service.IBizInteract.ISOBizInteract, ECCentral.Service.IBizInteract" version="2.0.0.0" />  </typeVersionMaps> |

节点说明：

|  |  |  |
| --- | --- | --- |
| 节点 | 属性 | 说明 |
| typeVersionMaps | globalDefaultVersion | 指定那些没有配置map节点的接口或基类，在调用ObjectFactory<T>.Instance以及其他的没有指定version的GetInstance重载方法时，在创建实现类实例的时候，所默认反射创建的类型的version，是匹配实现类上定义的VersionExport这个Attribute的Version属性的 |
| assemblyFolder | 指定ObjectFactory创建实现类时，需要搜索dll的目录，可以指定多个目录，用英文的分号或逗号隔开，支持绝对路径，也支持相对于应用程序根目录的相对路径，这样我们可以把一些实现类的dll分开到不同的目录来存放 |
| typeVersionMaps/map | type | 指定需要创建的接口或基类的类型，也就是对应到ObjectFactory<T>的泛型参数T |
| version | 在调用ObjectFactory<T>.Instance以及其他的没有指定version的GetInstance重载方法时，在创建实现类实例的时候，所默认反射创建的类型的version，是匹配实现类上定义的VersionExport这个Attribute的Version属性的 |

关于ObjectFactory的相关使用，请参看*$/ECCentral/02\_Design/02\_工具类帮助说明*目录下的文档《ObjectFactory使用说明.docx》；

1. 消息与订阅者关系的配置EventConsumer.config文件

用来定义消息和订阅了该消息的订阅者直接的关系；

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <subscription>  <event type="ECCentral.Service.EventMessage.SampleMessage\_1, ECCentral.Service.EventMessage">  <consumer type="ECCentral.Service.EventConsumer.SampleMessageConsumer\_A, ECCentral.Service.EventConsumer" />  <consumer type="ECCentral.Service.EventConsumer.SampleMessageConsumer\_B, ECCentral.Service.EventConsumer" />  </event>  <event type="ECCentral.Service.EventMessage.SampleMessage\_2, ECCentral.Service.EventMessage">  <consumer type="ECCentral.Service.EventConsumer.SampleMessageConsumer\_C, ECCentral.Service.EventConsumer">  <constructor>  <arg>1234</arg>  <arg>test</arg>  </constructor>  </consumer>  </event>  </subscription> |

节点说明：

|  |  |  |
| --- | --- | --- |
| 节点 | 属性 | 说明 |
| subscription/event | type | 指定一个消息的类型，该类型必须实现了接口IEventMessage |
| subscription/event/consumer | type | 指定订阅者的类型，该类型必须实现IConsumer<TMsg>接口，这个泛型TMsg必须为<event></event>节点上的type属性所指定的类型 |
| subscription/event/consumer/ constructor/arg |  | consumer节点的type属性所指定的类型的构造函数的参数，在程序反射创建该consumer订阅者类型的对象实例时，会用这些参数来传入构造函数 |

# 三、UI Portal端配置文件说明

*待整理…*