# 一、整体思路

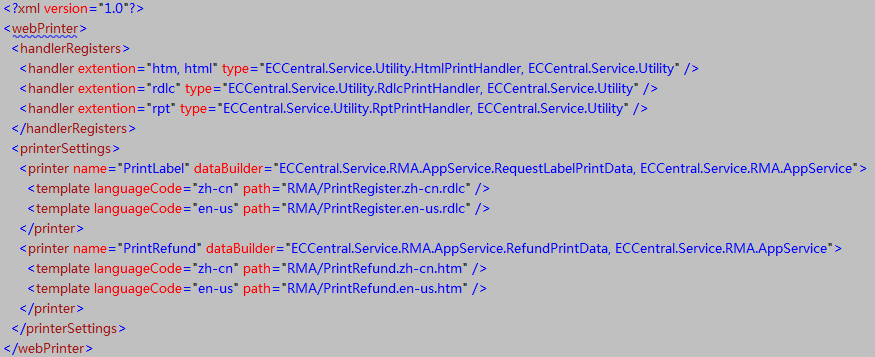
希望能够统一ECCentral中的打印的做法，可以支持扩展更多的Web打印方式（如：直接拼接html直接打印、Web上的水晶报表、Web上的ReportView等方式），但Portal端不用感知到扩展或变化，依然能使用原有的方式进行打印预览调用即可；  
整体的思路如下面的时序图所展示，整个打印模组分为Portal端的API和Service端的相应模块

# 二、Service端部分（2个接口 + 1个配置）

Service端主要会有2个标准接口IPrintDataBuild和IWebPrint；  
    public interface IPrintDataBuild  
    {  
        void BuildData(NameValueCollection requestPostData, out PrintKeyValueVariables variables, out PrintKeyDataSourceVariables tableVariables);  
    }

    public interface IWebPrint  
    {  
        void RenderHtmlForPrint(HttpContext context, string templateFileFullPath, PrintKeyValueVariables variables, PrintKeyDataSourceVariables tableVariables);  
    }

* 其中IPrintDataBuild的职责是用来构建打印所需要的数据，并将数据放到标准的数据容器类型对象KeyValueVariables和KeyTableVariables中，所以IPrintDataBuild的具体实现类的创建则是和具体需求点紧密相关的，不同的需求会需要构建不同的数据；所以在Service端需要有相应的配置，来指定每个打印需求所使用的具体的IPrintDataBuild的实现类；
* 而IWebPrint的职责是根据打印模板以及构建好的标准数据，进行打印预览的html的生成，所以IWebPrint的具体实现类的创建是和具体的打印方式相关的，也就是和模板类型紧密相关的，不同的模板类型可能需要不同的html生成模式，比如直接拼接html的就直接读取模板文件里的html字符串，然后用相应字符串模板引擎来处理数据替换就好了，而水晶报表则需要使用相应的web控件，然后设置模板地址、相应DataSource等再来Render出html；所以在Service端需要有相应的配置，来指定每种打印模板类型（根据模板文件的后缀名来区分）所使用的具体的IWebPrint的实现类；
* 针对两个接口的配置，以及打印模板路径的配置需求，在Service端有如下的配置结构：



该配置分为2个部分：  
A). <handlerRegisters>部分，用来配置IWebPrint接口，也就是设定某种后缀名的模板由哪种报表技术的处理类来处理，以后扩展新的报表技术后，就通过这里来添加新的模板后缀名和新的报表生成类  
B). <printerSettings>部分，用来配置每一个具体的打印需求，一个打印需求就是一个<printer>节点，其中通过dataBuilder属性来指定了一个IPrintDataBuild的实现类，也就是指定了数据的来源；其下的<template>节点用来指定每种语言代码所使用的模板文件的地址（支持在绝对地址，也支持相对于该配置文件的相对路径）

**所以可以看到，我们的打印模板是支持多语言的！**

（对于拼接Html方式的打印，所使用的html打印模板的格式规则，请参见文档《邮件模板及打印模板的统一规则.docx》;对于具体的配置文件以及相关的打印模板文件等配置信息请参加文档《6.EC-Central项目配置文件说明.docx》中相关部分）

# 三、Portal端部分（就一个标准的静态类方法）

直接调用方法：

HtmlViewHelper.WebPrintPreview("Customer", "PrintRefund")

方法第一个参数为Domain名称，第二参数为上面Service端的配置里设置的<printerSettings>节点下<printer>的name属性，用来指定具体的打印；

调用该方法后，将会弹出浏览器，并在浏览器中展示打印预览，并可以通过浏览器来实现最终打印；