开发SOAP服务组件

> REF: https://docs.intersystems.com/healthconnectlatest/csp/docbook/Doc.View.cls?KEY=ESOAP_web_service

这里有3个要点：

1. 使用EnsLib.SOAP.Service预置组件

   EnsLib.SOAP.Service继承%SOAP.WebService和Ens.BusinessService，前者提供Web服务的功能，而后者使之成为了Produciton的一个组件，可以像其他组件发送消息，并且拥有Ensemble的监控和错误日志，使用组件的参数项配置。在文档中您会看到直接使用%SOAP.WebService创建SOAP服务的例子，这通常用于更底层的IRIS编程。除非必须，开发Ensemble组件时用EnsLib.SOAP.Service更合适。

   注意，**由EnsLib.SOAP.Service创建的Web服务组件需要加入Production，并且Production处于运行状态才能提供SOAP服务**。

2. 不要是用SOAP Inbound Adapter

   因为性能和安全的原因，生产环境推荐使用的Web服务机制是"CSP处理机制", 关于什么是“CSP处理机制”，请参考Web Gateway 介绍。 EnsLib.SOAP.Service组件使用了SOAP InboundAdapter. 这个适配器监听用户定义的TCP接口，从一个特定的TCP接口接收处理HTTP请求，这种方式不是我们需要的"CSP处理机制“，因此在使用它，也就是创建子类继承EnsLib.SOAP.Service时，要覆盖父类的Adapter属性，强制保证不要用SOAP inbound Adapter。

3. 生产环境需要安装Web Server. 系统内嵌的PWS(Private Web Server)不能提供生产环境需要的性能和安全。简单说， 如果您正使用57772或者类似的InterSystems默认使用的端口号方法Web服务，别，请安装正式的IIS，Apache,或者Nginx。

Step 1: 创建一个提供Soap服务的类

以下是一个最简单的SOAP服务的提供类的示例：

```
Class Test.SOAP.ServiceDemo Extends EnsLib.SOAP.Service
{
///For this business service, ADAPTER should be "" so that we use the normal SOAP processing
Parameter ADAPTER =
Parameter SERVICENAME = "MyService";
Parameter NAMESPACE = "http://www.myhospital.org";
Parameter USECLASSNAMESPACES = 1;

Method HIPMessageServer(pInput As %String) As %Status [ WebMethod ]
  {    Set tRequest = ##class(Ens.StringRequest).%New()
       set tRequest.StringValue = pInput
```

```
12        set tStatus = ..SendRequestSync("TargetBusinessHostName", tRequest,
  .tResponse)
13        Quit tStatus
14      }
15    }
```

参数说明：

从CSP页面访问Soap Service

上面的类是一个.cls的类。在InterySystems的studio下可以进行简单的测试: 点击Studio工具栏的小地球(View Web Page), 可以打开这个服务的访问页面：

- 服务描述：
- HIPMessageServer: **此链接可以测试该方法，仅限于入参是字符串和数字类型**。对于流和复杂类型的入参，还是要用专门的测试工具测试。

这里我们要复习一下Caché Web Application的概念。我们刚刚编写的类继承了EnsLib.SOAP.Service, 它其实是一个CSP页面，而CSP界面又是Caché web application的一部分。当点击Studio的"View Web Page"的时候，studio打开了默认的浏览器来访问这个类，或者说CSP Page，使用的URL是当前namespace的缺省Web Application地址，加本类的类名(其实是组件名，后面解释)， 也就是上面地址栏的“csp/healthshare/demo/Test.SOAP.ServiceDemo.cls"。因此，要得到上面的结果， 需要两个条件：

1. Web Application "/csp/heathshare/demo"必须已经配置好，有Web访问的权限。
2. 默认访问这个页面是禁止的，会出现"非法csp请求"。开启这个测试页面要打开一个开关， 在Terminal中，在%SYS命名空间下执行以下命令。 其中webapplicationname是web应用的名称，但后面要加入“/”。例如web应用名是“/soap/myservice”， webapplicationname应该为”/soap/myservice/”

   set ^SYS("Security","CSP","AllowClass",webapplicationname,"%SOAP.WebServiceInfo")=1 set ^SYS("Security","CSP","AllowClass",webapplicationname,"%SOAP.WebServiceInvoke")=1

以下是真实的操作，注意Web Application的名字是"/csp/healthshare/demo", 但执行上面命令时的参数用的是"/csp/healthshare/demo/"

```
  ENSPRACTICE>zn "%sys"

  %SYS>zw ^SYS("Security","CSP","AllowClass")
  ^SYS("Security","CSP","AllowClass","/api/atelier/","%Api.Atelier")=1
  ^SYS("Security","CSP","AllowClass","/api/deepsee/","%Api.DeepSee")=1
  ^SYS("Security","CSP","AllowClass","/api/document/","%Api.Document")=1
  ^SYS("Security","CSP","AllowClass","/csp/ensdemo/","%CSP.UI.Portal.About")=1
  ^SYS("Security","CSP","AllowClass","/csp/samples/","%CSP.UI.Portal.About")=1
  ^SYS("Security","CSP","AllowClass","/csp/samples/","%SOAP.WebServiceInfo")=1
  ^SYS("Security","CSP","AllowClass","/csp/samples/","%SOAP.WebServiceInvoke")=1

  %SYS>s
^SYS("Security","CSP","AllowClass","/csp/enspractice/","%SOAP.WebServiceInvoke")=1
  %SYS>s
^SYS("Security","CSP","AllowClass","/csp/enspractice/","%SOAP.WebServiceInfo")=1
```

修改后不用重启任何组件，刷新浏览器, 您将看到正确的显示。

Step 2: 添加组件到Production, 用SOAPUI测试

上面使用的URL: "csp/healthshare/demo/Test.SOAP.ServiceDemo.cls"是Web Application加这个类在Production在BS组件的配置名字。或者说：如果您将类Test.SOAP.ServiceDemo.cls加到Prodcution, 组件名字也是Test.SOAp.ServiceDemo，那么就用这个URL访问。 如果您要配置自己的组件名字，比如有两个BS都用到这个类，配置的组件名字分别是"Service1"和"Service2", 那么访问这两个服务的URL是:

```
service1的URL:
/csp/healthshare/demo/Test.SOAP.ServiceDemo.cls?cfgitem=service1
service2的URL:
/csp/healthshare/demo/Test.SOAP.ServiceDemo.cls?cfgitem=service2
```

如果Web Application配置了使用password， URL需要加上如下的字段:
&CacheUserName=yourUserName&CachePassword=yourPassword

组件的配置项

配置

- pool size = 0
- SOAP会话支持: Ensemble web service 可以在web client 和web service 之间维护会话（session ID），以便再次调用。•将类参数SOAPSESSION设置为1， 默认值是0

SOAPUI测试

同样， 导入WSDL时也需要配置authentication.

没有配置用户authentication时soap service会发出错误消息：

Error loading [http://localhost:57772/csp/learning/Practice.BS.MyWebService.CLS?WSDL]: org.apache.xmlbeans.XmlException: org.apache.xmlbeans.XmlException: error: Unexpected character encountered (lex state 3): '&'

Advanced Skills

服务方法的入参构成

上面例子中的入参是字符串。定义服务规范的时候， 您还可以选择其他的类型，比如

- 流， 比如

```
Method HIPMessageServer(action As %String, message as %Stream.Object) As %Status [ WebMethod ]
```

- XML-enabled 对象， 也就是对象类要继承%XML.Adapter。这样服务请求里的xml结构才可以映射到类对象。

```
Method HIPMessageServer(action As %String, message as ESOAP.SOAPRequest) As %Status [ WebMethod ]
```

假设其中的ESOP.SOAPRequest是这么定义的。

```
Class ESOAP.SOAPResponse Extends (%RegisteredObject, %XML.Adaptor)
{
Property Name As %String;
Property Sex As %String;
}
```

当然，类里的属性可以有其他的类或者集合属性，但无论是什么元素，所有的内容都要是XML-enabled的。

返回消息的构建

上面代码的中HIPMessageServer()返回的是%Status, 这是ObjectScript语言中的一个状态对象。真实场景中通常您需要构建一个真实的返回消息，它可以是一个字符串，一个流，或者一个自定义的结构，比如。

Method HIPMessageServer(pInput As %String) As %String [WebMethod]

或者

Method HIPMessageServer(pInput As %String) As %Stream.GlobalCharacter [WebMethod]

或者如在线文档的例子：

Method HIPMessageServer(pInput As %String) As ESOAP.SOAPResponse [WebMethod]

其中ESOAP.SOAPResponse是您定义的任意类，文档上的定义只是一个示意：

```
Class ESOAP.SOAPResponse Extends (%RegisteredObject, %XML.Adaptor)
{
Property CustomerID As %Numeric;
Property Name As %String;
Property Street As %String;
Property City As %String;
Property State As %String;
Property Zip As %Numeric;
}
```

注意的是两点：

1. 不需要使用持久化类%Persistent，上面用的就是%RegisteredObject
2. 继承%XML.Adaptor，这样数据才可以打包成SOAP Response的格式。

返回错误信息

默认错误发生，web service 返回的SOAP 消息不包含错误发生的详细信息。如果您需要返回详细的错误信息，需要调用EnsLib.SOAP.Service的两个方法，**ReturnMethodStatusFault()** or **ReturnStatusFault()** 。

两个方法的签名是：

```
ClassMethod ReturnStatusFault(pCode As %String, pStatus As %Status)
ClassMethod ReturnMethodStatusFault(pStatus As %Status)
```

其中pCode是WSDL定义的, pStatus是详细的错误信息。

以下是修改后的代码。

```
Method HIPMessageServer(pInput As %String) As %Status [ WebMethod ]
  {   Set tRequest = ##class(Ens.StringRequest).%New()
      set tRequest.StringValue = pInput
      set tStatus = ..SendRequestSync("TargetBusinessHostName", tRequest,
.tResponse)
      if $$$ISERR(sc) do ..ReturnMethodStatusFault(tStatus)

      //如果成功收到tResponse,用tResponse的数据构造soapresponse
      set soapresponse=##class(ESOAP.SOAPResponse).%New()
      set soapresponse.CustomerID=response.CustomerID
      Quit soapresponse
  }
}
```

匿名访问还是用户名密码

如果测试时，浏览器中页面显示错误 “An error occurred with the CSP application and has been logged to system error log",很有可能是您的Web Application不支持您所做的匿名访问。 尤其是如果使用的命名空间的缺省Web Application, 通常是不合适把它改成不要求密码的。

如果不需要匿名访问，可以用下面的请求参数发送用户名密码

/csp/healthshare/demo/Test.SOAP.ServiceDemo.cls?cfgitem=service1&CacheUserName=uname&CachePassword=passwd

其中”uname"和"passwd"是有访问权限的

如果需要匿名访问，最好重新创建一个Web Application, 比如"/test", 那么您访问的URL将是。注意您必须确认Web服务器/Web Gateway认识此URL并把它转发给这个Web Application。 这可能需要您查看Web Gateway的配置，并在Web Gateway配置页面上做跟踪。

/test/Test.SOAP.ServiceDemo.cls?cfgitem=service1

SOAP请求的Debug

开启 SOAP日志需要在terminal执行以下命令：

```
set ^ISCSOAP("Log") = “ios”        //设置SOAP日志打开，i:记录Inbound消息；o:记录outbound消息；s:记录安全相关信息
set ^ISCSOAP("LogFile") = “c:\temp\a.log”    //设置文件保存路径和文件名
```

调试结束后应关闭SOAP 日志，执行：

```
kill ^ISCSOAP("Log")
```

END