Article
Jonathan Schulman · Sep 14, 2016 4m read

Creating a Dummy SOAP Web Service

A "Dummy" SOAP Web Service

When dealing with SOAP in Caché, it is sometimes necessary to debug errors by directly accessing (and sometimes editing) the XML which is sent, i.e. the SOAP request and subsequent SOAP response. If you're debugging a Caché web service, it is often useful to use a tool such as SoapUI (https://www.soapui.org/) to manually create and control the SOAP request, so that the effect of adjustments can easily be seen on the Caché web service.

But what if you have a web service (possibly not Caché), and you want to debug the associated Caché web client? You may have the SOAP response XML saved in a file (e.g. a Caché SOAP log) – what you need is a 'dummy' web service to send it to your Caché web client, behaving like the actual web service.

As I often work on debugging customer Caché web client issues in Support, I created such a 'dummy' web service – see below:

```
Class JSUtil.DummyWebService Extends %CSP.Page
{

/// Mimic a SOAP Web Service by sending the specified SOAP Response XML.
/// Typically this XML will be copied-and-pasted from a SOAP log.
Parameter CONTENTTYPE = "text/xml";

/// File containing the SOAP Response XML:
Parameter XMLFILENAME = "C:\data\soapresponse.txt";

ClassMethod OnPage() As %Status
   {
                set XML=""
                set stream = ##class(%Stream.FileCharacter).%New()
                set sc = stream.LinkToFile(..#XMLFILENAME)

                while 'stream.AtEnd {
                 set XML = XML_stream.Read()
                }

    write XML
   quit $$$OK
   }

}
```

To use class JSUtil.DummyWebService:

0. Change the value of Parameter XMLFILENAME to be the location of the XML containing the SOAP response from the web service. Typically this file will be created by manually cutting and pasting the response XML from the Caché SOAP log.
0. Get the URL for this CSP page by using Studio's 'View Web Page' – it should display the response XML.
0. Paste the above URL in the LOCATION parameter of the Caché web client.

Now when you invoke the Caché web client, it receives the SOAP request XML pointed to by parameter XMLFILENAME.

I have used this technique many times to help debug Caché web clients. Consider the following example:

The SOAP log contains the following error in the 'Input to Web client':

ERROR #6203: Unexpected Element

(The complete SOAP log can be found here: *https://github.com/ISC-schulman/InterSystems/raw/master/soaplog.txt*)

We also have the WSDL for the web service (*https://github.com/ISC-schulman/InterSystems/raw/master/example.wsdl*), but do not have access to the web service itself. (While this is common for InterSystems Support, admittedly it is probably unusual for a customer – this is just a simple example to illustrate how DummyWebService can be used.)

First, using DummyWebService we reproduce the error:

0. Use the SOAP Wizard to generate a web client from the provided WSDL. Take all the defaults (though you can specify a Package name.)
0. Look at the SOAP log, and cut-and-paste the web service response XML ('Input to Web client') to a file:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:req="http://test.systelab.com/">
<soapenv:Header/>
<soapenv:Body><req:createAsynchronuosRequestResponseElement>OK</req:createAsynchronuosRequestResponseElement>
</soapenv:Body>
</soapenv:Envelope>
```

0. Point to this file via parameter XMLFILENAME in JSUtil.DummyWebService.
0. Use Studio ' Display Web Page' to view DummyWebService – it should display the contents of the file you created in Step 2.
0. Cut-and-paste the URL from Step 4 into the LOCATION parameter of the web client generated in Step 1.
0. Invoke the web client, e.g.

```
set client = ##class(MyWebService.RequestWSSoapHttpPort).%New()
do client.createAsynchronuosRequest("x")
quit
```

0. Verify (e.g. via a SOAP log) that the same error occurs, namely 'ERROR #6203: Unexpected Element'.

Next we eliminate this error. Here is where some experience or trial-and-error is used, but again this is only meant to illustrate how one uses DummyWebService.

0. Use the SOAP Wizard and the provided WSDL as before to generate a web client – specify a different Package name to prevent overwriting the first web client. Otherwise, take all the defaults, except one:

   select "Use unwrapped message format for document style web methods" in Step 3 of the SOAP Wizard.

0. Repeat Steps 5 and 6 above.
0. Verify (e.g. via a SOAP log) that the web client error has been eliminated.

(Note: the use of 'unwrapped message format' is a common solution for web client problems – see "Using the SOAP Wizard" in our documentation for more information.)

Summary

One can use the class DummyWebService to send a specified SOAP response (e.g. from a SOAP log) to a Caché web client, mimicking the response of a web service.

#SOAP #Caché

Source URL: https://community.intersystems.com/post/creating-dummy-soap-web-service