

Vetverslunin Direct.is

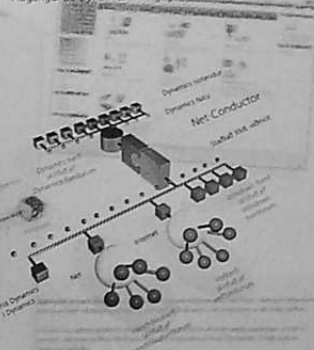
Direct er vetverslun með fjölskráðum vörum af tilloft- og bakþjónu með áherslu á gætt þjónu og gættum viðskiptum. Þetta er áhrifamikill og áhrifamikill vefur sem hefur verið byggður á .NET og Dynamics NAV. Þetta er áhrifamikill og áhrifamikill vefur sem hefur verið byggður á .NET og Dynamics NAV.

Viðskipta- og vetverslun Direct.is

Direct.is er áhrifamikill og áhrifamikill vefur sem hefur verið byggður á .NET og Dynamics NAV. Þetta er áhrifamikill og áhrifamikill vefur sem hefur verið byggður á .NET og Dynamics NAV.

NET-Conductor í hnotskurn

- Ný leið til að tengast Dynamics NAV
- Stöðug leið til samskipta við Dynamics NAV byggð á XML vefþjónustum
- Aðgangur að ferlum og virkni sem þegar er til staðar í Dynamics NAV
- Auðveld leið til bygga ofan á fyrirliggjandi Dynamics NAV kerfi með nýjustu tækni í hugbúnaðargæð
- Einföld leið til að tengja Dynamics NAV við aðrar tæknir í flöru Microsoft-eyra sem Sharepoint Server
- Aðgangur að Dynamics NAV gegnum vefþjónmót

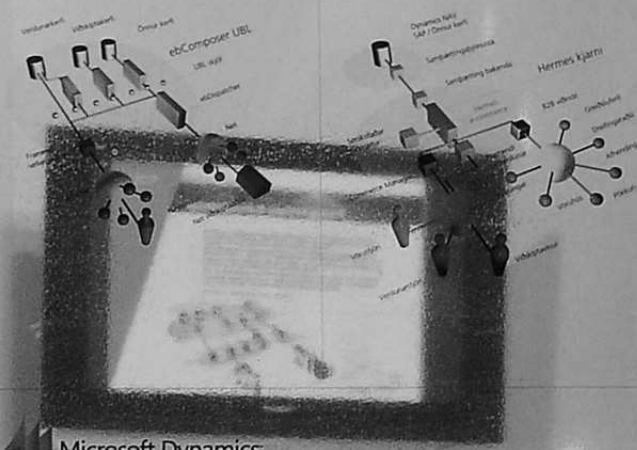


ebComposer UBL í hnotskurn

- Samskipti með rafræn viðskiptaskýlið samkvæmt viðurkenndum stöðli
- Úttæsla á hinum nýja UBL stöðli sem mun koma í stað EDI
- Leið fyrir viðskiptakerfi til að skiptast á stöðluðum rafrænum skýlum eins og þöntunum eða reikningum
- Kerfisningar sem auðvelt er að innleiða inn í fyrirliggjandi umhverfi
- Mun styrkt þróunartími en reikna mætti með
- Fullkominn stuðningur við staðalinn UBL 2.0 og NES

Hermes í hnotskurn

- Aðrir vefir eru tengdir við miklum sveigjanlegum og áhrifamikill kerfi
- Viðskipta- og vetverslunir bæði við innviðskipta kerfi og samstarfsaðila
- Eðlileg kerfi og eiginleikar sem tryggja áhrifamikill og áhrifamikill vefur
- Byggt á stöðluðum grunn og viðurkenndri tækni
- Auðveld vör- og verslunartjón með sérstök stjórnkerfi
- Hönnun mótur að því að gera sem flesta viðskiptaferla sjálfvirkni með því að tengjast bein viðskiptaferum sem fyrir eru



Microsoft Dynamics

Microsoft Dynamics NAV í hnotskurn

- Aðhliða viðskiptahugbúnaður fyrir allar gerðir fyrirtækja
- Notendur eru á heimavelli þar sem vörur eru eins og í öðrum Microsoft hugbúnaði og hægt að aðlaga að þörfum hvers og eins
- Auðveld samþætting við önnur kerfi í fyrirliggjandi umhverfi
- Notendur ná árangri með samverkandi viðskiptaferum
- Akvarðanatáka er auðveldu með góðu aðgengi að öllum upplýsingum
- Rue de Net sérhæfir sig í ráðgjöf og innleiðingu á Dynamics NAV



Microsoft Dynamics NAV er áhrifamikill og áhrifamikill vefur sem hefur verið byggður á .NET og Dynamics NAV.

NETConductor Invoker Samples

Index

NETConductor – Invoker Samples	3
ShakesphereTest	3
EsynamsTest	5
StockQuoteTest	7
GoogleSearchTest.....	9

Documentation:

29/05/2008 Alfred B. Thordarson
Created original document

29/05/2008 Eva Hrönn Guðnadóttir
Layout improved for Rue de Net

03/06/2008 Ellen Margrét Bæhrenz
Merging original document and layout

29/05/2008 Alfred B. Thordarson
Added text to the samples and overhaul of the style.

NETConductor – Invoker Samples

ShakesphereTest

This function uses a public web service

(<http://www.xmlme.com/WSShakespeare.asmx?op=GetSpeech>) to retrieve information about quote from Shakespheres plays.

```
// Variables:
ws [Automation NETConductor Invoker]
xml [Automation NETConductor XmlContainer]
Dlg Dialog
q Text
play Text
speaker Text
quote Text

// C/AL Source Code:
IF NOT CONFIRM('Run ShakespeareTest?') THEN EXIT;
q:='oyster';

// Our Dialog:
dlg.OPEN('Quote part: #1#####\'+i.e. 'oyster', 'sweet
sorrow' and then press Enter',q);
dlg.UPDATE();
dlg.INPUT(1,q);
dlg.CLOSE();
IF ISCLEAR(ws) THEN BEGIN
    CREATE(ws);
    // Initilize our NETConductor license:
    ws.InitLicense('C:\Program Files\Rue de Net\NETConductor\DT','C:\Program
Files\Rue de Net\NETConductor\DT');
END;

// Attach to the public web service and write out the service description
ws.AttachToWebService('http://www.xmlme.com/WSShakespeare.asmx');
ws.WriteOutService('c:\temp\WS_Info_Shakespeare.txt');

// Attach to the operation GetSpeech and set the parameters of the operation
ws.AttachToWebOperation('GetSpeech');
ws.SetParameter('Request', q);
IF NOT ws.InvokeWebOperation() THEN ERROR(ws.GetInvokeError());
xml:=ws.GetXmlReturnValue();

// Save the reply to and XML file:
xml.SaveXml('C:\temp\WS_Result_Shakespeare.xml');
xml.SetRoot('SPEECH');

// Retrieve play info and show confirmation window with the reply information.
play:=xml.GetText('PLAY');
speaker:=xml.GetText('SPEAKER');
quote:=xml.GetText('.');
IF NOT CONFIRM(play+' '+speaker+' : '+quote) THEN EXIT;
```

When we run this test we first get a confirmation window asking us if we want to run this test. We press Yes and then we get this window:

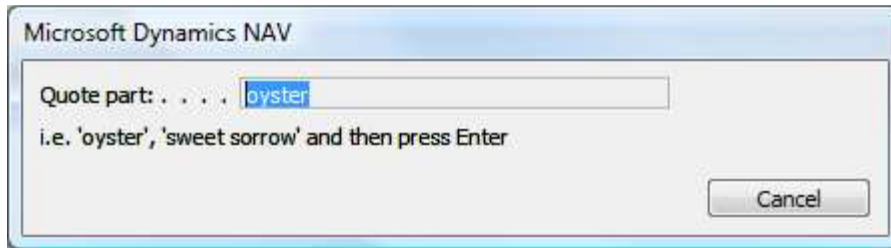


Image 1: Shakesphere sample, Quote dialog.

When we press enter the web service searches for that phrase from one of Shakespeare's plays. The speech, speaker, and play will be returned as an XML string. We put that result into a confirmation windows that appears to the user.

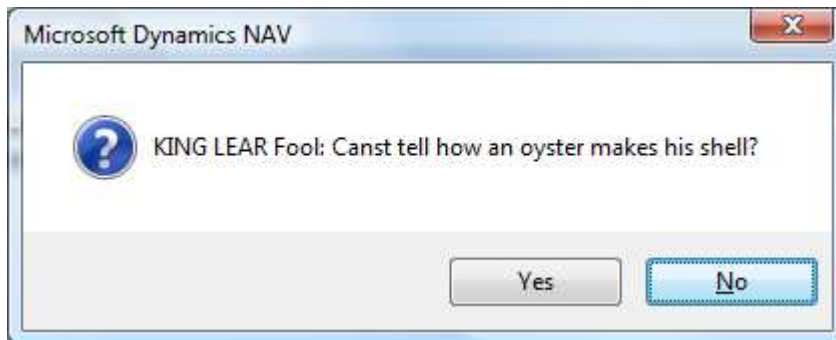


Image 2: Shakesphere sample, Web Service reply

The resulting XML from the web service looks like this:

```
<SPEECH>
  <PLAY>KING LEAR</PLAY>
  <SPEAKER>Fool</SPEAKER>
  Canst tell how an oyster makes his shell?
</SPEECH>
```

EsynamsTest

This function uses a public web service

(<http://www.esynaps.com/WebServices/SearchWS.asmx?op=Search>) to search ASMX/WSDL files on the web.

```
// Variables:
ws [Automation NETConductor Invoker]
xml [Automation NETConductor XmlContainer]
Dlg Dialog
keyword Text
url Text

// C/AL Source Code:
IF NOT CONFIRM('Run EsynamsTest?') THEN EXIT;
keyword:='Microsoft';

// Our Dialog
dlg.OPEN('Keyword: #1#####\'+
        'i.e. 'Microsoft', 'Google' and then press Enter',keyword);
dlg.UPDATE();
dlg.INPUT(1,keyword);
dlg.CLOSE();

IF ISCLEAR(ws) THEN BEGIN
    CREATE(ws);
// Initialize our NETConductor license
    ws.InitLicense('C:\Program Files\Rue de Net\NETConductor\DT','C:\Program
Files\Rue de Net\NETConductor\DT');
END;

// Attach to our web service and web operation and set the parameters
ws.AttachToWebService('http://www.esynaps.com/WebServices/SearchWS.asmx');
ws.WriteOutService('c:\temp\WS_Info_Esynaps.txt');
ws.AttachToWebOperation('Search');
ws.SetParameter('KeyWord', keyword);

// Invoke the web service operation and save the reply to XML.
IF NOT ws.InvokeWebOperation() THEN ERROR(ws.GetInvokeError());
xml:=ws.GetXmlReturnValue();
xml.SaveXml('C:\temp\WS_Result_Esynaps.xml');

IF xml.SetRoot('//url') THEN REPEAT
    url:=xml.GetText('.');
    IF NOT CONFIRM('Esynaps: '+url+', continue?') THEN EXIT;
UNTIL xml.NextRoot()=FALSE;
```

The first dialog we see is the Keyword dialog. We will use the web service to search by the Keyword we type in here:

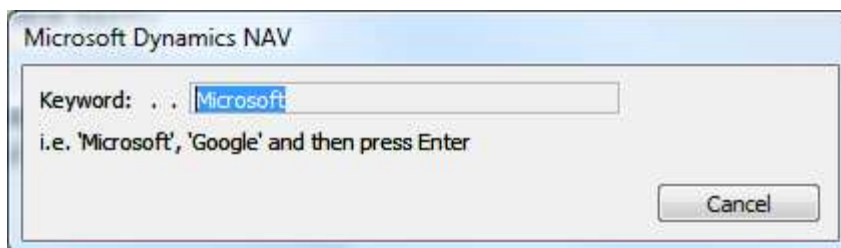


Image 3: EsynapsTest, Keyword Dialog

Then the web service will search for all ASMX and WSDL files connected to that Keyword on the web. We display each file we find in a confirmation dialog:

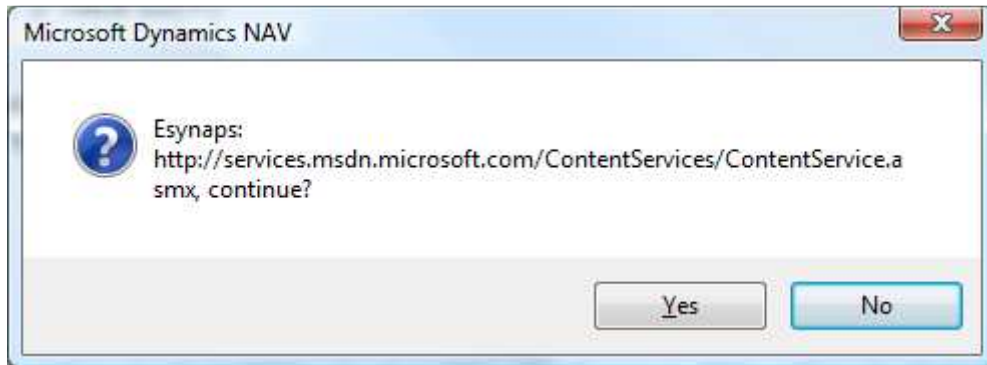


Image 4: EsynapsTest, Web Service result

The XML we receive looks like this:

```
<Results>
  <url>
    http://social.msdn.microsoft.com/Forums/en-US/asmxandxml/thread/ba85e1b3-25ea
    4fcf-91c4-9b1c3ca61ac9
  </url>
  <url>
    http://social.msdn.microsoft.com/Forums/en-US/asmxandxml/thread/36731836-9cc4
    4cbd-8a88-6ec03a03567f
  </url>
  <url>
    http://services.msdn.microsoft.com/ContentServices/ContentService.asmx
  </url>
  <url>http://teraserver-usa.com/TerraService2.asmx</url>
  <url>http://teraserver-usa.com/TerraService2.asmx?WSDL</url>
  <url>http://www.seroundtable.com/archives/006437.html</url>
  <url>
    http://blog.svencipido.be/Blog/How+To+Host+An+Asmx+In+A+Windows+
    Service+Without+Using+IIS+.aspx
  </url>
  <url>http://www.sharepointblogs.com/forums/t/16071.aspx</url>
  <url>
    http://www.nabble.com/How-to-Access-Microsoft-Reporting-Services-(RS)
    Webservices--(reportservice2005.asmx-wsdl)-td23083561.html
  </url>
  <url>
    http://www.eggheadcafe.com/software/aspnet/33643731/creating-stub-class-for
    s.aspx
  </url>
</Results>
```

StockQuoteTest

This function uses a public web service

(<http://www.websvcex.com/stockquote.asmx?op=GetQuote>) to receive latest status of stock for given company.

```
// Variables:
ws [Automation NETConductor Invoker]
xml [Automation NETConductor XmlContainer]
Dlg Dialog
symbol Text
last Text
date Text
time Text

// C/AL Source Code:
IF NOT CONFIRM('Run StockQuoteTest?') THEN EXIT;
symbol:='MSFT';
// Our Dialog
dlg.OPEN('Symbol: #1#####\'+
        'i.e. 'MSFT', 'GOOG' and then press Enter',symbol);
dlg.UPDATE();
dlg.INPUT(1,symbol);
dlg.CLOSE();

IF ISCLEAR(ws) THEN BEGIN
    CREATE(ws);
    // Initialize our NETConductor license.
    ws.InitLicense('C:\Program Files\Rue de Net\NETConductor\DT','C:\Program
Files\Rue de Net\NETConductor\DT');
END;

// Attach to the web service and operation and set the parameters.
ws.AttachToWebService('http://www.websvcex.com/stockquote.asmx');
ws.WriteOutService('c:\temp\WS_Info_StockQuote.txt');
ws.AttachToWebOperation('GetQuote');
ws.SetParameter('symbol', symbol);

// Invoke the web operation. Retrieve XML save it to file.
IF NOT ws.InvokeWebOperation() THEN ERROR(ws.GetInvokeError());
xml:=ws.GetXmlReturnValue();
xml.SaveXml('C:\temp\WS_Result_StockQuote.xml');

// Display the info from the StockQuote in a confirmation window.
xml.SetRoot('Stock');
symbol:=xml.GetText('Symbol');
last:=xml.GetText('Last');
date:=xml.GetText('Date');
time:=xml.GetText('Time');

IF NOT CONFIRM(symbol+' '+last+' @ '+date+' '+time) THEN EXIT;
```

The first dialog we see is the Symbol dialog. We will use the web service to search by the Symbol we type in here:

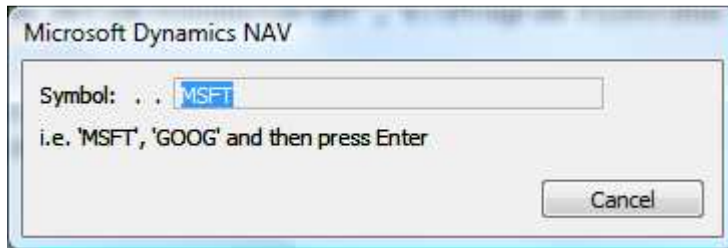


Image 5: StockQuoteTest, Symbol Dialog

After we press enter the web service will search for information related to the Symbol we provided earlier.

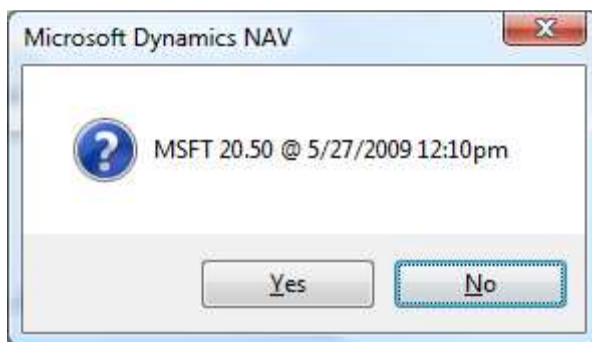


Image 6: StockQuoteTest, Stock Information Reply

The XML we receive back looks like this:

```
<StockQuotes>
  <Stock>
    <Symbol>MSFT</Symbol>
    <Last>20.50</Last>
    <Date>5/27/2009</Date>
    <Time>12:10pm</Time>
    <Change>+0.16</Change>
    <Open>20.25</Open>
    <High>20.60</High>
    <Low>20.24</Low>
    <Volume>14178582</Volume>
    <MktCap>182.4B</MktCap>
    <PreviousClose>20.34</PreviousClose>
    <PercentageChange>+0.79%</PercentageChange>
    <AnnRange>14.87 - 29.57</AnnRange>
    <Earnings>1.737</Earnings>
    <P-E>11.71</P-E>
    <Name>Microsoft Corpora</Name>
  </Stock>
</StockQuotes>
```

GoogleSearchTest

This function uses a web service from Google (<http://api.google.com/search/beta2>) to search Google.

```
// Variables:
ws [Automation NETConductor Invoker]
xml [Automation NETConductor XmlContainer]
complex [Automation NETConductor ComplexContainer]
resultElement [Automation NETConductor ComplexContainer]
result [Automation NETConductor ComplexContainer]
urltitle [Record 84930 NET Privileges]
Dlg Dialog
searchString Text
i Integer
time Text

// C/AL Source Code:
IF NOT CONFIRM('Run GoogleSearchTest?') THEN EXIT;

searchString := 'Rue de Net';
dlg.OPEN('Search String: #1#####\'+
        'i.e. ' 'Rue de Net', 'GOOGLE' and then press Enter',searchString);
dlg.UPDATE();
dlg.INPUT(1,searchString);
dlg.CLOSE();

// Initialize the NETConductor license
IF ISCLEAR(ws) THEN BEGIN
    CREATE(ws);
    ws.InitLicense('C:\Program Files\Rue de Net\NETConductor\DT','C:\Program
Files\Rue de Net\NETConductor\DT');
END;

// Attach to our web service and operation and set our parameters.
ws.AttachToWebService_2('http://www.soapclient.com/xml/googlesearch.wsdl',
'http://api.google.com/search/beta2');
ws.AttachToWebOperation('doGoogleSearch');
// The Google search API developers key, gotten from
http://www.soapclient.com/googleSearch.html.
ws.SetParameter('key', 'firp7ewYCi3NrIIRyuT9Gr6bycyt7AHV');
ws.SetParameter('q', searchString);
ws.SetParameter('start', '0');
ws.SetParameter('maxResults', '10');
ws.SetParameter('filter', 'false');
ws.SetParameter('restrict', '0');
ws.SetParameter('safeSearch', 'true');
ws.SetParameter('lr', 'lang_en');
ws.SetParameter('ie', '');
ws.SetParameter('oe', '');

// Write out service description.
ws.WriteOutService('c:\temp\WS_Info_Google.txt');

IF NOT ws.InvokeWebOperation() THEN ERROR(ws.GetInvokeError());

// Receive XML reply and receive complex value back(class).
xml:=ws.GetXmlReturnValue();
xml.SaveXml('C:\temp\WS_Result_GoogleSearch.xml');
complex := ws.GetComplexReturnValue();
// Receive array of complex values from our original complex value.
resultElement := complex.GetComplexValue('resultElements');

i := 1;

// Loop through the results and add them to a temporary table.
```

```

IF CONFIRM ('Continue?')
THEN REPEAT
  // Receive next result from the array of complex values
  result := resultElement.GetArrayComplexValue(i);
  urltitle.Name:=FORMAT(i);
  urltitle.Description:=COPYSTR(result.GetValue('URL'),1,50);
  urltitle.INSERT;
  i += 1;
UNTIL i > resultElement.GetArrayLength()-1
ELSE BEGIN
  MESSAGE('Break');
  END;

// Display our data in a form.
FORM.RUNMODAL(84931,urltitle);

```

The first dialog we see is the Search String dialog. We will use the web service to search Google by the Search String we type in here:

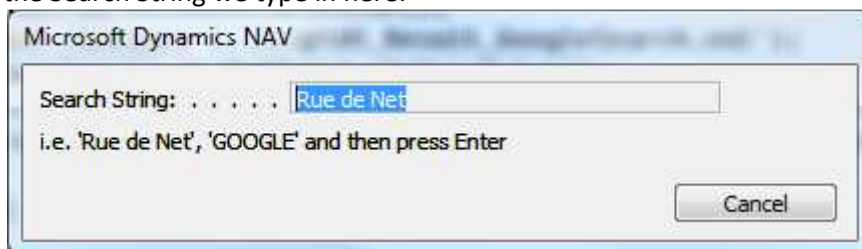


Image 7: Google Search Test, Search String Dialog

When you have typed in your search string you can continue. The search results should be like this:

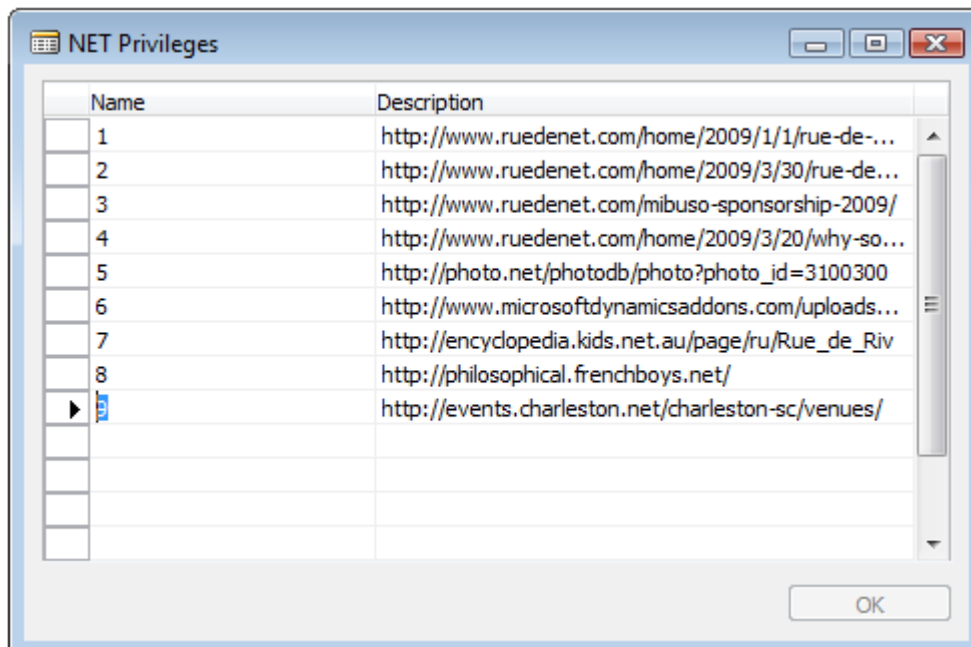


Image 8: Google Search Test, Result Form

ComplexContainer

The ComplexContainer type is used in Invoker to represent classes and arrays. The classes we receive from the google search look like this:

```
Class GoogleSearchResult
  System.Boolean documentFiltering
  System.String searchComments
  System.Int32 estimatedTotalResultsCount
  System.Boolean estimateIsExact
  ResultElement[] resultElements (*2)
  System.String searchQuery
  System.Int32 startIndex
  System.Int32 endIndex
  System.String searchTips
  DirectoryCategory[] directoryCategories (*3)
  System.Double searchTime
```

```
Class ResultElement
  System.String summary
  System.String URL
  System.String snippet
  System.String title
  System.String cachedSize
  System.Boolean relatedInformationPresent
  System.String hostName
  DirectoryCategory directoryCategory (*3)
  System.String directoryTitle
```

```
Class DirectoryCategory
  System.String fullViewableName
```

So we can see that the class GoogleSearchResult has a couple of variables. To receive that class from the Invoker, we use the operation GetComplexReturnValue().

```
complex := ws.GetComplexReturnValue();
```

Here we get the original class from the web method into our own ComplexContainer type. The GoogleSearchResult class also has an array of ResultElement. To get the array to a ComplexContainer variable we use the same method as above.

```
resultElement := complex.GetComplexValue('resultElements');
```

Now the resultElement variable is an array of ResultElement. To get individual ResultElement from the ComplexContainer variable we use the GetArrayComplexValue(Int index);

```
result := resultElement.GetArrayComplexValue(i);
```

Now the result variable holds an instance of the ResultElement class. We can use that variable now to get information about the individual search hits. This line gets the URL value from the search hit.

```
result.GetValue('URL');
```

XMLContainer

The XMLContainer is used by the Invoker to represent an XML document. The StockQuoteTest uses a public web service and receives an XML reply. The XML looks like this:

```
<StockQuotes>
  <Stock>
    <Symbol>MSFT</Symbol>
    <Last>20.50</Last>
    <Date>5/27/2009</Date>
    <Time>12:10pm</Time>
    <Change>+0.16</Change>
    <Open>20.25</Open>
    <High>20.60</High>
    <Low>20.24</Low>
    <Volume>14178582</Volume>
    <MktCap>182.4B</MktCap>
    <PreviousClose>20.34</PreviousClose>
    <PercentageChange>+0.79%</PercentageChange>
    <AnnRange>14.87 - 29.57</AnnRange>
    <Earns>1.737</Earns>
    <P-E>11.71</P-E>
    <Name>Microsoft Corpora</Name>
  </Stock>
</StockQuotes>
```

To navigate through the XML with the Invoker we use a couple of methods. The first method we look at is GetXmlReturnValue(). This method returns the XML reply from the web service to our XMLContainer variable.

```
xml := ws.GetXmlReturnValue();
```

Next we want to navigate through our XML.

To do that we can use the SetRoot(String rootName). This method sets the current root of the XML according to the rootName parameter. This operation uses XPath. Since the first Node in our reply XML is StockQuotes and it has no siblings we can set the root to it's first and only child Stock.

```
xml.SetRoot('Stock');
```

Now the current root is Stock and from there we want to navigate to the children it has.

We use the GetText(String node) method.

The parameter node is a node or an attribute.

We can see that Stock (the current root now) has a child named Symbol and we can therefore retrieve the value from that node by using the GetText method:

```
symbol := xml.GetText('Symbol');
```

Now we can loop through all these children in the same way and then we would have navigated the XML document.



Vesturgata 3, 101 Reykjavik, Iceland
ruedenet@ruedenet.com
Tel: (+354) 414 5050
Fax: (+354) 414 5051