

Vetverslunin Direct.is

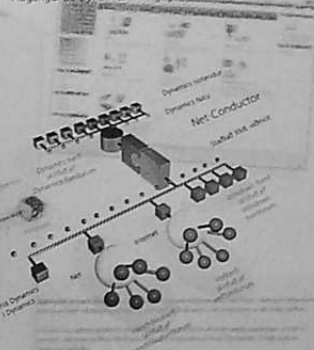
Direct er vetverslun með fjölbreytt vörurval af fíttæk- og bakþjónu með áherslu á gætt þjónu og gættum viðskiptum. Direct er að styrta leið milli framleiðenda og kaupenda með fæm milliðum auk þess að tryggja við umgjöngu og þingfærni sem takar sér í lagra vöruskiptum. Þú þess að halda einni kassa er markmiðið hjá Direct að sem flestir viðskiptafólkur séu til.

Viðskipta- og vetverslun Direct.is

Beitendur þess eru þeir fólk sem þess að tryggja gættum viðskiptum með áherslu á gættum þjónu og gættum viðskiptum. Direct er að styrta leið milli framleiðenda og kaupenda með fæm milliðum auk þess að tryggja við umgjöngu og þingfærni sem takar sér í lagra vöruskiptum. Þú þess að halda einni kassa er markmiðið hjá Direct að sem flestir viðskiptafólkur séu til.

NET-Conductor í hnotskurn

- Ný leið til að tengast Dynamics NAV
- Stöðugt 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 hugbúnaðargæði
- Einföld leið til að tengja Dynamics NAV við aðrar tækjur í flöru Microsoft-eyra sem Sharepoint Server
- Aðgangur að Dynamics NAV gegnum vefþjónu

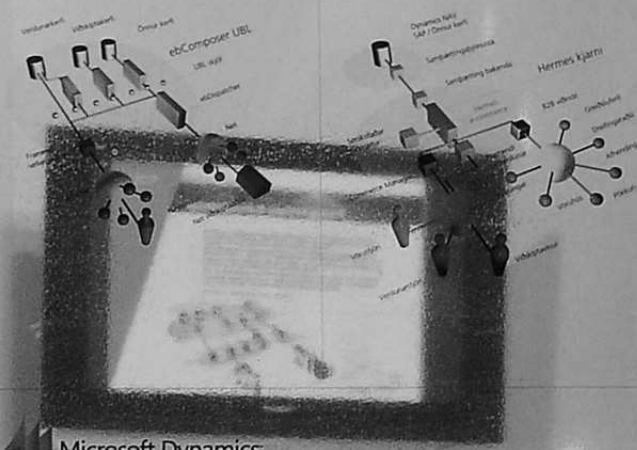


ebComposer UBL í hnotskurn

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

Hermes í hnotskurn

- Aðrir vefþjónustur og málum sveigjanleg og samþætta kerfi
- Viðtakari þess til samskipta bæði við innviðtakari kerfi og samstarfsaðila
- Eðlileg kerfi og eiginleikar sem tryggja áreiðanlegu kaupfun viðskiptavina
- Byggt á stöðluðum grunni og viðurkenndri tækni
- Auðveld vörur- og verslunartjón með sérstök stjórkerfi
- Hönnun mátt að því að gera sem flesta viðskiptafólkur sjá hvernig með því að tengjast bein viðskiptakerfum sem fyrir eru



Microsoft Dynamics

Microsoft Dynamics NAV í hnotskurn

- Aðhliða viðskiptahugbúnaður fyrir allar gerðir fyrirtækja
- Notendur eru á heimavelli þar sem vörur er 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 viðskiptakerfum
- Akvarðanataka er auðveldu með góðu aðgengi að öllum upplýsingum
- Rue de Net sérhæfir sig í ráðgjöf, hönnun og innleiðingu á Dynamics NAV



Microsoft Dynamics NAV er áður Microsoft Business Solution, Revision Edition og þar áður Revision

NETConductor

PHP Script Sample

Index

NETConductor – PHP Script Sample 3

Step 1: The NETConductor Web Service 3

Step 2: Creating the PHP script 5

Step 3: Working with the data 6

Documentation:

22/05/2009 Richard Ottó O'Brien
Created document

NETConductor – PHP Script Sample

This tutorial describes how to create a PHP script that consumes and uses the NETConductor web service. This document is divided into three main steps.

The first step explains where to find the demo NETConductor web service and invokes an operation from the web browser. In this step you also see what the replies from the NETConductor web service look like.

The second step demonstrates how to create a PHP script, how to consume the NETConductor web service and shows sample code that connects and gets the data from the service.

The final step has to do with working with the data gotten from the web service.

Step 1: The NETConductor Web Service

The first thing we do is to visit the Rue de Net website at www.ruedenet.com.

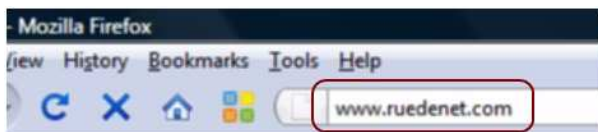


Image 1: The Rue de Net website

From there we click the link on the left labeled: „NETConductor“

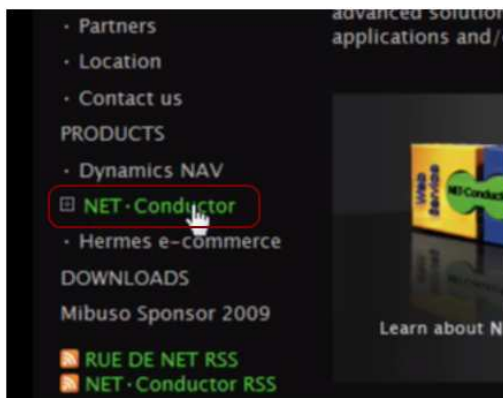


Image 2: The Rue de Net website, NETConductor link

Now we see the NETConductor page and in here we click on „Demos“.

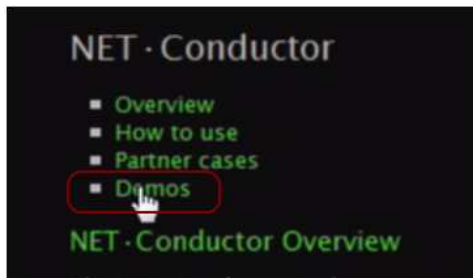


Image 3: The Rue de Net web site, Demos link

When the Demos page has been loaded, we scroll down and find the link labeled „Basics NETConductor Web Service“. We click on that link.



Image 4: The Rue de Net web site, Basics Web Service link

Now we are looking at a NETConductor web service!

What we want to do is press the „RunCodeUnit“ link:

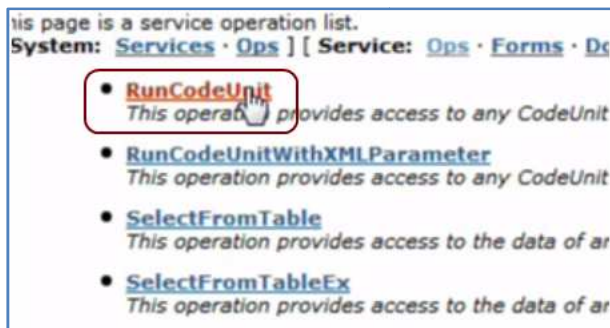


Image 5: The NETConductor basics web service.

Next we press the „Invoke“ button. When that button is pressed we call the method RunCodeUnit with the default parameters, ServiceName is empty, CodeUnit No is 84922 and the function we want to call within CodeUnit 84922 is the function “Test“.

This call is basically `RunCodeUnit(“”, 84922, “\Function\”=“Test\”““)`.

CodeUnit 84922 contains NETConductor samples and with the above call we ask the NETConductor to run the function Test which fetches data from NAV and shows it to the user.

Parameter	Value
serviceName (CAPITAL String)	<input type="text"/>
	The name of the NET-Conductor DT service to use for Navision work (i.e. CRONUS.DT). If the 'serviceName' is left empty the operation will read the 'DefaultDataTier' setting in the 'Web.config' configuration file.
no * (Int)	<input type="text" value="84922"/>
	Number of the CodeUnit to run.
parameters (String)	<input type="text" value='Function="Test"'/>
	Parameters to send to the OnRun function of the CodeUnit. The format of this parameter is: "p1"="v1", "p2"="v2",... where p1 is the name of a parameter and v1 is its value. If it is left empty then no parameters are sent to the OnRun function.
	<input type="button" value="Invoke"/>
	Returns: The XML returned from the CodeUnit, which may hold multiple strings, hashtables, record sets or error messages.
	* are required parameters.

Image 6: The NETConductor basics web service, RunCodeUnit method call.

When the RunCodeUnit method has finished fetching data from NAV it displays the data in the explorer with XML.

This is how the XML looks like. Actual data from NAV represented in XML.

```

- <ReplySet name="6337827463150646829">
  <Reply type="String" name="Test" value="This test provides a string, a
  <Reply type="String" name="Test" value="Second test string."/ >
  <Reply type="String" name="Test" value="Third test string."/ >
  - <Reply type="Hashtable" name="House of Flying Daggers">
    <KeyPair key="Director" value="Yimou Zhang"/>
    <KeyPair key="Writer" value="Bin Wang"/>
    <KeyPair key="Jm" value="Takeshi Kaneshiro"/>
    <KeyPair key="Leo" value="Andy Lau"/>
    <KeyPair key="Mei" value="Ziyi Zhang"/>
    <KeyPair key="Yee" value="Dandan Song"/>
  </Reply>
  - <Reply type="Hashtable" name="Shi mian mai fu">

```

Image 7: The NETConductor basics web service, XML reply.

Step 2: Creating the PHP script

Now we want to use the data we fetched with the NETConductor web service in step 1. In order to use the methods the web service has to offer we must somehow communicate with it. To do so we will use SOAP to consume the web service.

We must make sure we have enabled the PHP SOAP extension. To do this it is best to create a script containing the *phpinfo()* and running that script in a browser. This will display all sorts of information about our PHP server environment.

soap	
Soap Client	enabled
Soap Server	enabled

Image8: Information about our PHP server environment.

Soap Client and *Soap Server* must both be enable. If they are not you must go to PHP.net (www.php.net) and download a build with all the extensions. Instructions on how to set it up are provided on their page.

Lets start our sample by creating a new PHP script file and instantiating the *SoapClient* class and speciyng the location of our web service WSDL. The *SoapClient* class is rather new feature of PHP, and enables us to specify a web service and it extracts all its methods. We can than as will be shown later in this manual, simply call these methods as if they were standard PHP functions.

```
$client = new SoapClient
(
    'http://gaffall/NETConductor/Basics.asmx?WSDL',
    array("trace" => 1,
        "exceptions" => 1)
);
```

Then we call the CodeUnit via our SoapClient instance, the parameters for the web method we call are placed within a array, and construct a DOM document from the XML reply coming from the NETConductor.

```
// first call the codeunit
$result = $client->RunCodeUnit
(
    array
    (
        "serviceName" => "ncdev.dt",
        "no" => 84922,
        "parameters" => "\"Function\"=\"Test\""
    )
);

// Now construct a DOM to be able to go over our results:
$DOMResult = new DomDocument;
$DOMResult->loadXML($result->RunCodeUnitResult->any);
```

We have now successfully created a PHP script which consumes our web service.

Step 3: Working with the data

To be able to manipulate and present the data we use *DOM XPath* to enable us to fetch data which we want from our XML reply. To do this we perform a *XPath* query and specify which nodes we want.

```
// Construct a query to fetch specific nodes
$Xpath = new DomXPath($DOMResult);
$Nodes = $Xpath->query("*/G_L_Account/*");
```

Now the variable contains a *DOMNodeList* with all the nodes we specified in our *XPATH* query. With a simple for-loop we extract the *innerText* of an XML node and then print the data.

```
for ($i = 0; $i < $Nodes->length; $i += 3)
{
```

```

printf(
    "%s %s %s<br/>",
    DOMInnerHTML($Nodes->item($i)),
    DOMInnerHTML($Nodes->item($i + 1)),
    DOMInnerHTML($Nodes->item($i + 2))
);
}

```

We must create a function to extract the our data from the XML. Below is a trivial method which does exactly this. It accepts as a parameter a XML node and cuts out the text contained inside the node (for example <node>THIS TEXT RIGHT HERE</node>).

```

function DOMInnerHTML($element)
{
    $innerHTML = "";
    $children = $element->childNodes;
    foreach ($children as $child)
    {
        $tmp_dom = new DOMDocument();
        $tmp_dom->appendChild($tmp_dom->importNode($child, true));
        $innerHTML.=trim($tmp_dom->saveHTML());
    }
    return $innerHTML;
}

```

Now all is done and we should be able to try out and see whether our script works as we hoped.

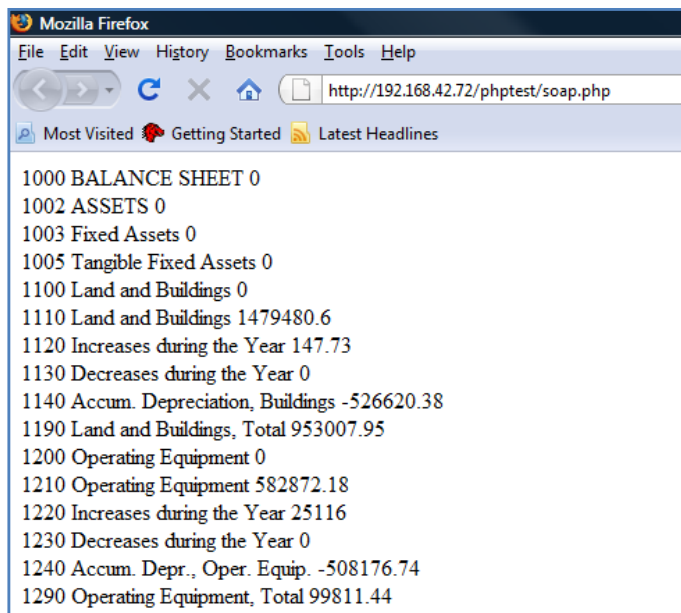


Image 9: The results from the created PHP script

If all is right you should see the same.

You have now consumed a NETConductor web service and displayed data from NAV with a PHP script.

Congratulations!



Vesturgata 3, 101 Reykjavik, Iceland

ruedenet@ruedenet.com

Tel: (+354) 414 5050

Fax: (+354) 414 5051