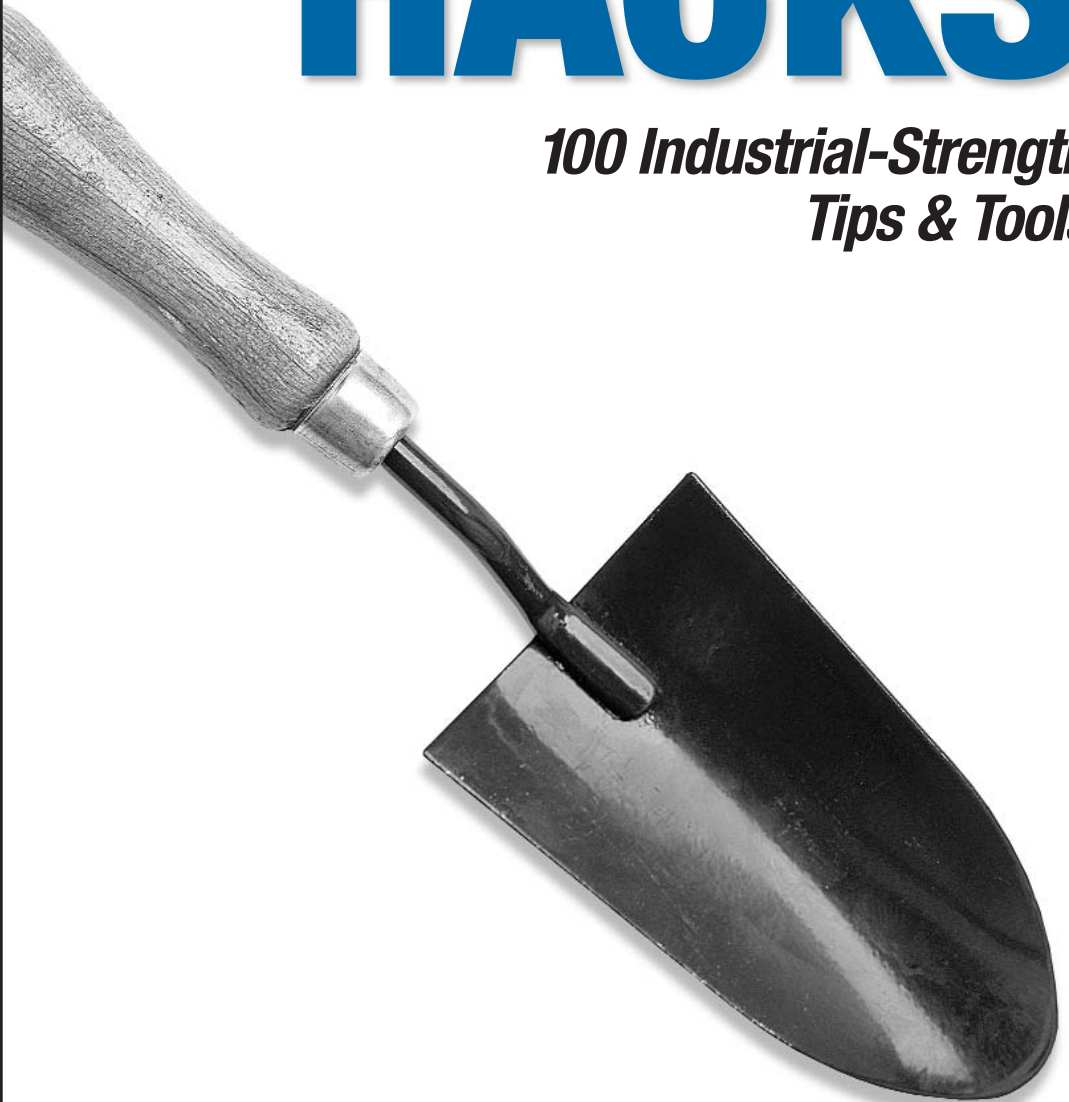


# EXCEL HACKS™

*100 Industrial-Strength  
Tips & Tools*



O'REILLY®

*David Hawley & Raina Hawley*

HACK  
#99

## Access SOAP Web Services from Excel

If your spreadsheet needs to access constantly updated data, or if you need to access services hosted on another computer, Excel's Web Services support will enable you to get connected.

SOAP-based Web Services have been a key part of Microsoft's plans for .NET, as well as a common feature of toolkits from other vendors. SOAP (the acronym doesn't mean anything) is a protocol that uses XML to transmit information between systems. In the case you'll explore here, it's used to call procedures and return values. A companion specification, Web Service Definition Language (WSDL), describes Web Services so that applications can connect to them easily. Microsoft's Web Services Reference Tool can take a WSDL file and generate VBA code your application can use to access SOAP-based web services.



This hack uses Excel features that are available only in Excel XP and Excel 2003 on Windows. Earlier versions of Excel do not support this, and neither do current or announced Macintosh versions of Excel.

Making this work requires downloading the Office Web Services Toolkit. As its location has changed a few times, it's easiest to go to <http://www.microsoft.com/downloads/search.aspx> and search for "Office Web Services Toolkit". Separate versions are available for Office XP and Office 2003. You'll need to install this toolkit, using the directions that come with it, before proceeding with this hack.

Once you've installed the toolkit, you can start connecting your spreadsheet to web services. To get to the Web Service References Tool (its name inside of Excel), you'll need to select Tools → Macro → Visual Basic Editor. On the Tools menu of the VBE, you'll find Web Services References.... Selecting this brings up the dialog box shown in Figure 8-21.

You can use the search features in the top left of this dialog to find services through Microsoft's Universal Discovery, Description and Integration (UDDI) service, or you can enter a URL for the WSDL file at the lower left. You can find a listing of public services at <http://xmethods.net/>, though you should definitely test to make sure the services still work before you integrate them with your spreadsheets. Many services also require license keys and sometimes license payments, but for this example you'll use one that is available for free. It returns the IP address for a given domain name.

Start by telling Excel which service you want to use—in this case, <http://www.cosme.nu/services/dns.php?wsdl>. Enter that value in the URL: box at the

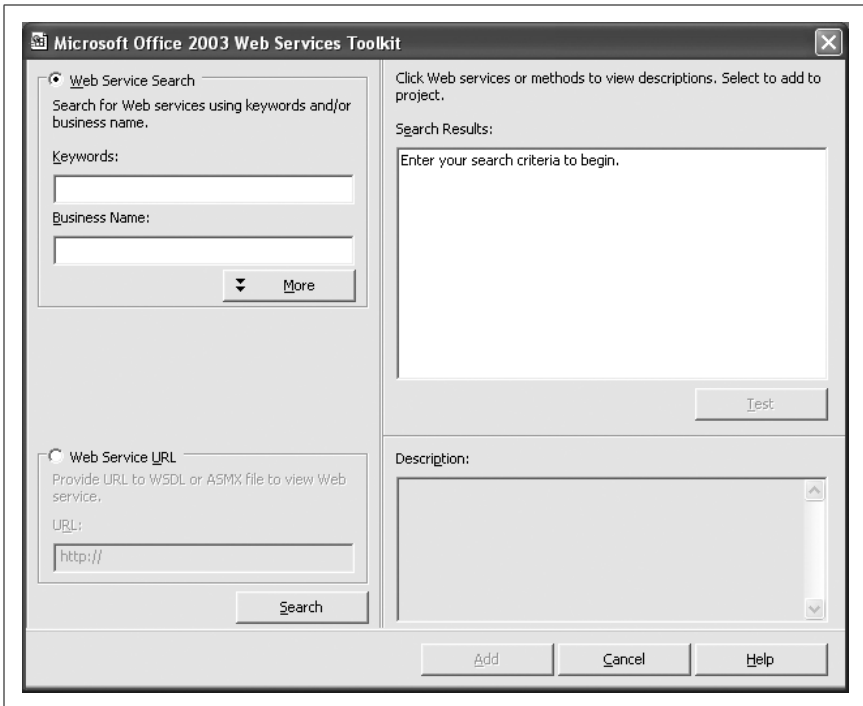


Figure 8-21. The Microsoft Office Web Services Toolkit in action

bottom left and click Search. A search result of dns will appear in the top right, as shown in Figure 8-22. Check the box to its left.

Clicking the Add button will make Excel generate VBA code for the service, as shown in Figure 8-23.

Next, close the VBE and set up a very simple spreadsheet such as the one shown in Figure 8-24.

To add a button for calling the service, display the Control toolbar by right-clicking a toolbar and choosing Control Toolbox from the pop-up menu. Click the button icon, and then click the spreadsheet wherever you want the button to go. Right-click the button, and choose Properties from the pop-up menu. Under Name, enter **GetData**; under Caption, enter **Get IP Address**. Close the Properties dialog box, and your spreadsheet should look something like that shown in Figure 8-25.

To add the final piece, right-click the button you added and choose View Code. In the window that appears, enter this subroutine:

```
Private Sub GetData_Click()  
    Dim info As New clsws_dns
```

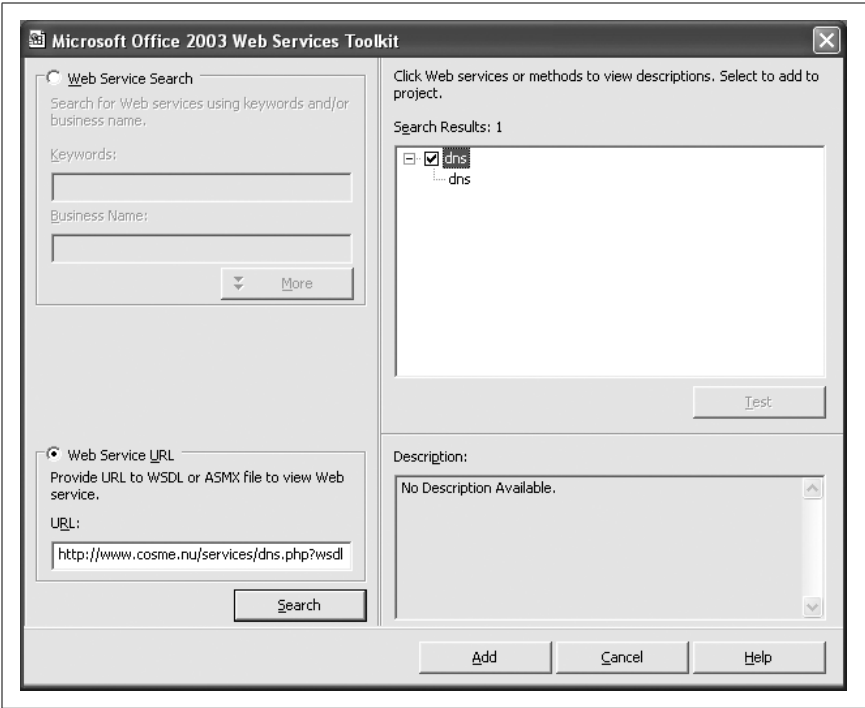


Figure 8-22. Telling the Web Services Toolkit to generate code for a web service

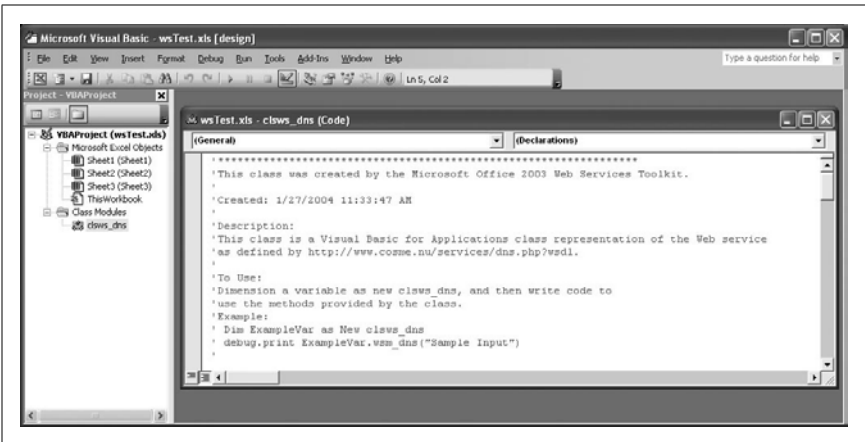


Figure 8-23. VBA code generated by the Web Services Toolkit for accessing the dns service

```

Dim name As String
Dim IP As String

name = Range("B2").Text
    
```

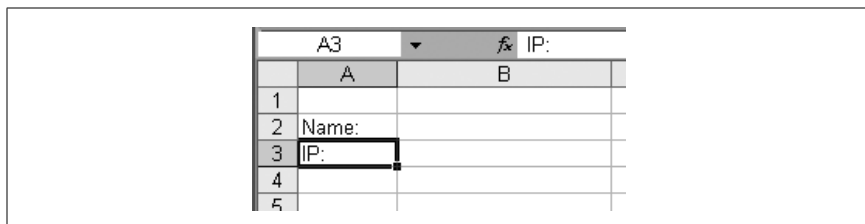


Figure 8-24. A spreadsheet for adding web services

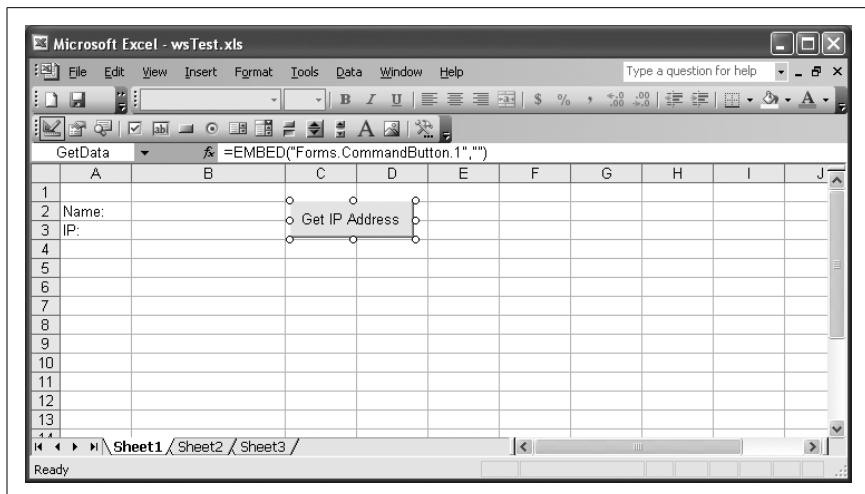


Figure 8-25. Spreadsheet with button for calling web services

```
IP = info.wsm_dns(name)
```

```
Set IPRange = Range("B3")
```

```
IPRange.Value = IP
```

```
End Sub
```

This code is pretty simple. It references the object the toolkit created for the web service, and creates variables for the name and IP address. It collects the name from cell B2, calls the web service with the name as an argument, and then puts the value returned into cell B3. Once you've entered this code and closed the VBE, you can leave design mode by making sure the triangle and ruler icon at the left of the Control toolbar isn't highlighted. The spreadsheet will now enable you to enter a domain name in cell B2. Clicking the Get IP Address button will put the IP address corresponding to that domain name in cell B3. Figures 8-26 and 8-27 show this spreadsheet in action with different domain names.

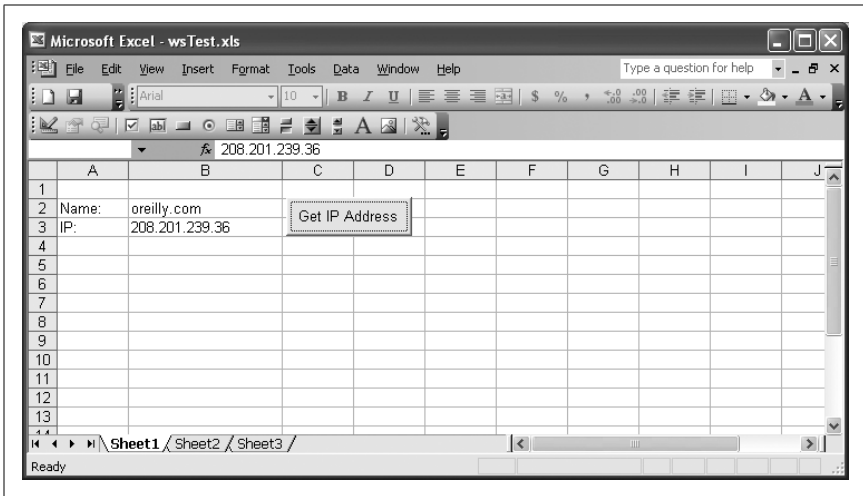


Figure 8-26. A retrieved IP address for oreilly.com

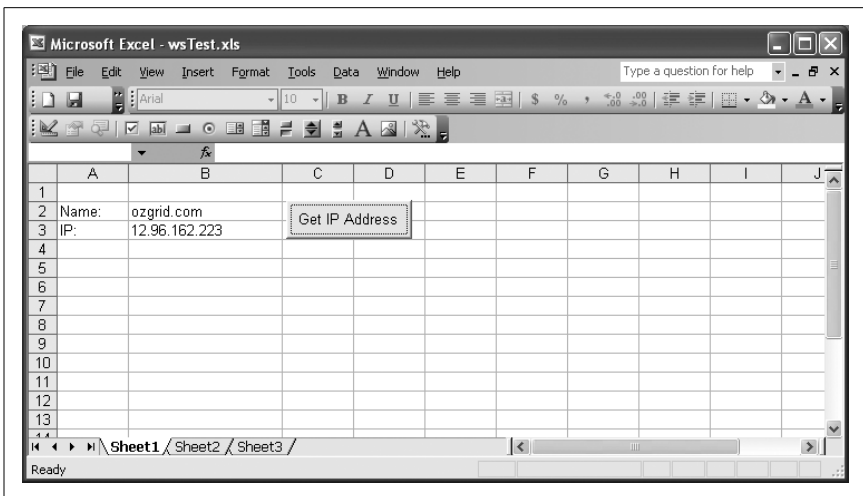


Figure 8-27. A retrieved IP address for ozgrid.com

IP address resolution is one of the simpler services out there, but many times services this simple can be very useful in a spreadsheet—for instance, for currency converters, price retrieval, postal code processing, and much more. You don’t even need to learn about SOAP or WSDL to use these services, as the Web Services Toolkit takes care of all of that for you.

A few caveats are worth mentioning, however. First, the computer has to be connected to a network for a web service to work. You probably don’t want to create spreadsheets that depend heavily on web services if their users will

be working on them at 30,000 feet and will be thoroughly disconnected. (Spreadsheets such as this one, which uses a web service to populate fields but doesn't need to be connected constantly, are probably OK.)

The other major issue with web services generally is that the field is in significant flux. At the time of this writing, SOAP had moved from Version 1.1 to 1.2, and a new version of WSDL was under development; what's more, many people feel UDDI might eventually be replaced with other technologies. For now, be certain to test the services you use, and keep an eye out for new versions of the Office Web Services Toolkit.

—*Simon St.Laurent*