Consuming XML Web Services in VFP 8

Doug Hennig

XML Web services are becoming very important for application development, and VFP 8 makes it easier than ever to use them. This month, Doug explores consuming XML Web services in VFP applications, including how to use the new XML Web Service Builder to bind controls to XML Web services results with very little code.

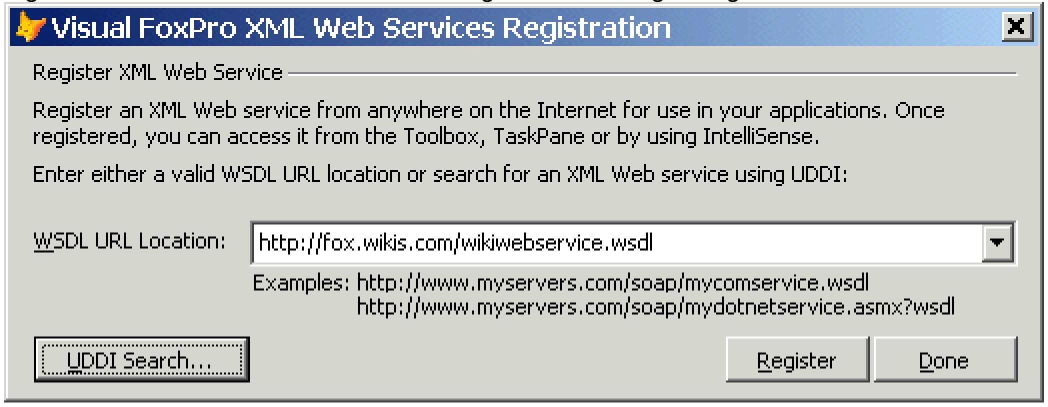
Microsoft's vision of software development strongly embraces XML Web services. A Web service is nothing more than a component that sits on a Web server somewhere. A client calls a function of the component, using SOAP (Simple Object Access Protocol) as the transport mechanism for the function call, parameters, and return value, and does something with the results, which are usually in the form of XML. The reason Web services are becoming more important for application development is they can form the building blocks of distributed applications.

In my June 2002 FoxTalk article "Searching With Web Services", I discussed using Web services in VFP 7 to provide a means of searching various Web sites for information. This month, we'll revisit this topic, but using the new tools that come with VFP 8. We won't discuss the plumbing involved in Web services, such as SOAP, WSDL, and UDDI. There are numerous white papers available, including many on the MSDN Web site (http://msdn.microsoft.com), that go into great detail on that topic.

Registering a Web service

The first step to using a Web service is to register it with VFP. Although this isn't strictly necessary, it's easier to work with a Web services if you do. To register a Web service, launch the XML Web Services Registration dialog (shown in **Figure 1**) by clicking on the Web Services button in the Types page of the IntelliSense Manager (this was the only way to launch this dialog in VFP 7), clicking on the Register link in the My XML Web Services category of the new Toolbox, choosing the Register an XML Web Service link in the XML Web Services pane of the new Task Pane Manager, or selecting the New button in the XML Web Services Manager, which is also available from the XML Web Services pane of the Task Pane Manager.

Figure 1. Use the XML Web Services Registration dialog to register a Web service in VFP.


Visual FoxPro XML Web Services Registration

Register XML Web Service

Register an XML Web service from anywhere on the Internet for use in your applications. Once registered, you can access it from the Toolbox, TaskPane or by using IntelliSense.

Enter either a valid WSDL URL location or search for an XML Web service using UDDI:

WSDL URL Location: http://fox.wikis.com/wikiwebservice.wsdl

Examples: http://www.myservers.com/soap/mycomservice.wsdl
http://www.myservers.com/soap/mydotnetservice.asmx?wsdl

Enter the URL, including the http:// prefix, for the WSDL of the Web service you want to register, and click on the Register button. If you don't know the URL, click on the UDDI Search button. In the UDDI Search dialog, type a name and click the Search button. The combo box is populated with a list of matching Web services from the UDDI database and the edit box below it shows information about the selected Web service.

For the examples in this article, we'll work with the Web service that provides searching on the FoxWiki (a great FoxPro resource created by Steven Black, http://fox.wikis.com), so register the following WSDL file: http://fox.wikis.com/wikiwebservice.wsdl.

Using Web services in VFP 8

The way you use registered Web services in VFP 8 is totally different than it was in VFP 7. Rather than selecting a Web service from IntelliSense when you type LOCAL SomeVariable AS, you drag a Web service from the Toolbox to either a code window, which generates code, or a container (such as a form) in the Form or Class Designers, which adds an object to the container. Let's look at both of these mechanisms.

Using Web services programmatically

Here's the code generated when you drag the FoxWiki Web service from the Toolbox to a code window:

```
LOCAL lowikiwebservice AS "MSSOAP.SoapClient30"
LOCAL lowikiwebservice AS "XML Web Service"
* Do not remove or alter following line. It is used to support IntelliSense for
* your XML Web service.
*__VFPWSDef__: lowikiwebservice = http://fox.wikis.com/wikiwebservice.wsdl ,
*    wikiwebservice , wikiwebserviceSoapPort
LOCAL loException, lcErrorMsg, loWSHandler
TRY
  loWSHandler = NEWOBJECT("WSHandler", IIF(VERSION(2) = 0, "", ;
    HOME() + "FFC\") + "_ws3client.vcx")
  lowikiwebservice = loWSHandler.SetupClient( ;
    "http://fox.wikis.com/wikiwebservice.wsdl", "wikiwebservice", ;
    "wikiwebserviceSoapPort")
* Call your XML Web service here.  ex: leResult = lowikiwebservice.SomeMethod()
CATCH TO loException
  lcErrorMsg = "Error: " + TRANSFORM(loException.Errorno) + " - " + ;
    loException.Message
  DO CASE
    CASE VARTYPE(lowikiwebservice)#"O"
      * Handle SOAP error connecting to web service
    CASE !EMPTY(lowikiwebservice.FaultCode)
      * Handle SOAP error calling method
      lcErrorMsg = lcErrorMsg + CHR(13) + lowikiwebservice.Detail
    OTHERWISE
      * Handle other error
  ENDCASE
  * Use for debugging purposes
  MESSAGEBOX(lcErrorMsg)
FINALLY
ENDTRY
```

This code uses the new structured error handling in VFP 8 (see my January 2003 article, "CATCH Me If You Can", for details on this wonderful new feature) so it handles errors gracefully. Also, if you used Web services in VFP 7, you'll notice this code uses a different class, WSHandler in _WS3Client.VCX rather than WSClient in _WebServices.VCX, to act as the Web service proxy.

Add the following three lines of code to the code generated by the Toolbox (just above the CATCH statement), or run TestWikiWebService.PRG, included with this month's Subscriber Downloads.

```
lcXML = loWikiWebService.GetTextSearch('Stonefield')
xmltocursor(lcXML)
browse
```

This codes searches the FoxWiki for all occurrences of "Stonefield" (substitute your favorite search string), and displays them in a browse window, complete with title and URL for the full text. Not bad for three lines of code! But wait until you see what we can do with just two lines!

Using Web services visually

Dragging a Web service from the Toolbox to the Form or Class Designers adds an instance of the same WSHandler class to the container and fills in several properties with information about the selected Web service. The great thing about this class is that VFP provides a builder for it that allows you to configure the object so it not only calls a particular method of the Web service, but also obtains any parameters needed for the method in a variety of ways (such as prompting the user or from controls in the form) and binds the results to other controls.

Let's build a form that allows us to search the FoxWiki and displays the results in a grid. Create a form called WikiSearch.SCX. (This form is included with this month's Subscriber Downloads if you want to see the finished results.) Drop a label, a textbox, a command button, and a grid on it, laid out as shown in **Figure 6**, and set the properties of these controls as shown in **Table 1**.

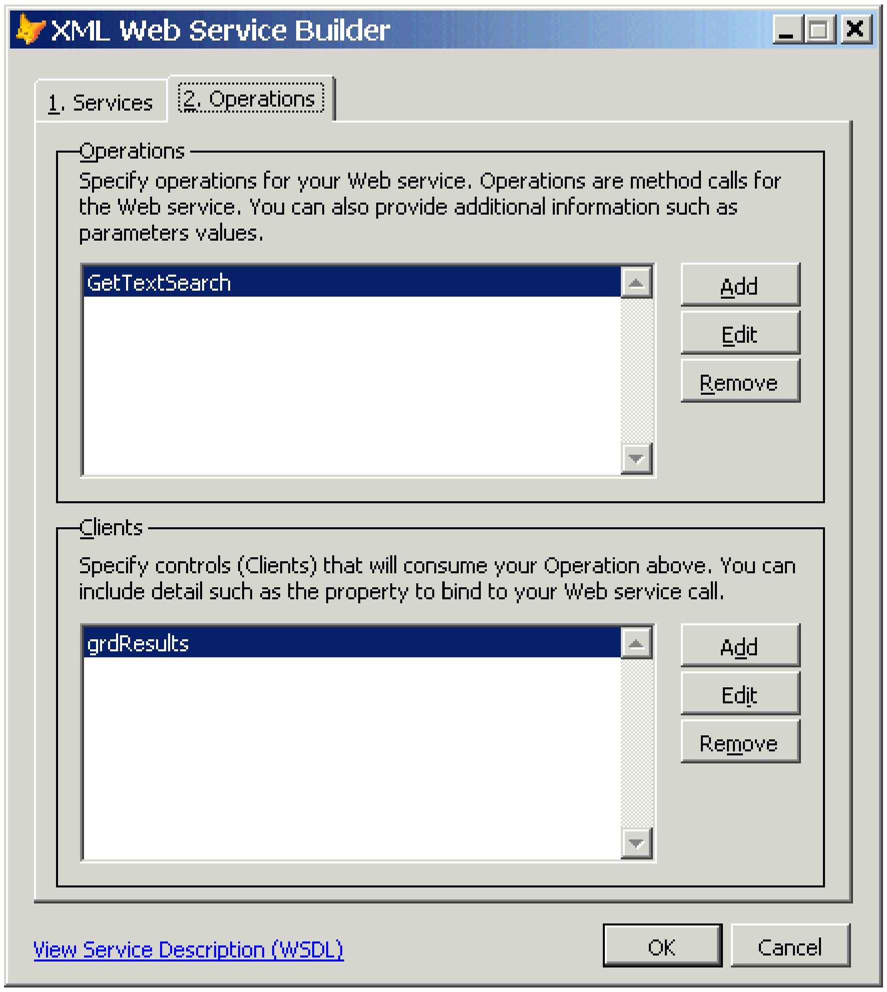
Control	**Property**	**Value**
Form	AutoCenter	.T.
	Caption	Wiki Search
	DataSession	2-Private
Label	Name	lblSearch
	AutoSize	.T.
	Caption	Search for:
TextBox	Name	txtSearch
CommandButton	Name	cmdSearch
	Caption	Search
	Default	.T.
Grid	Name	grdResults
	ColumnCount	2
	DeleteMark	.F.
	RecordMark	.F.
	SplitBar	.F.
	Column1.Header1.Caption	Title
	Column2.Header1.Caption	Link

Table 1. Set the properties of the controls in the form to these values.

Bring up the Toolbox from the Tools menu or the VFP toolbar, click on the My XML Web Services category, and drag the icon for the WikiWebService to the form. That will add a WSHandler object to the form and automatically launch the XML Web Service Builder.

The Operations page of the builder is automatically selected because the Web service to use has already been defined in the object. (The Services page allows you to define which Web service you're calling.) The Operations page allows you to define the operations (which methods of the Web service are called) and which clients (other controls) will consume the results of each operation. **Figure 2** shows this page.

Figure 2. The Operations page of the XML Web Service Builder allows you to define Web service operations and clients of those operations.


XML Web Service Builder

Operations

Specify operations for your Web service. Operations are method calls for the Web service. You can also provide additional information such as parameters values.

GetTextSearch

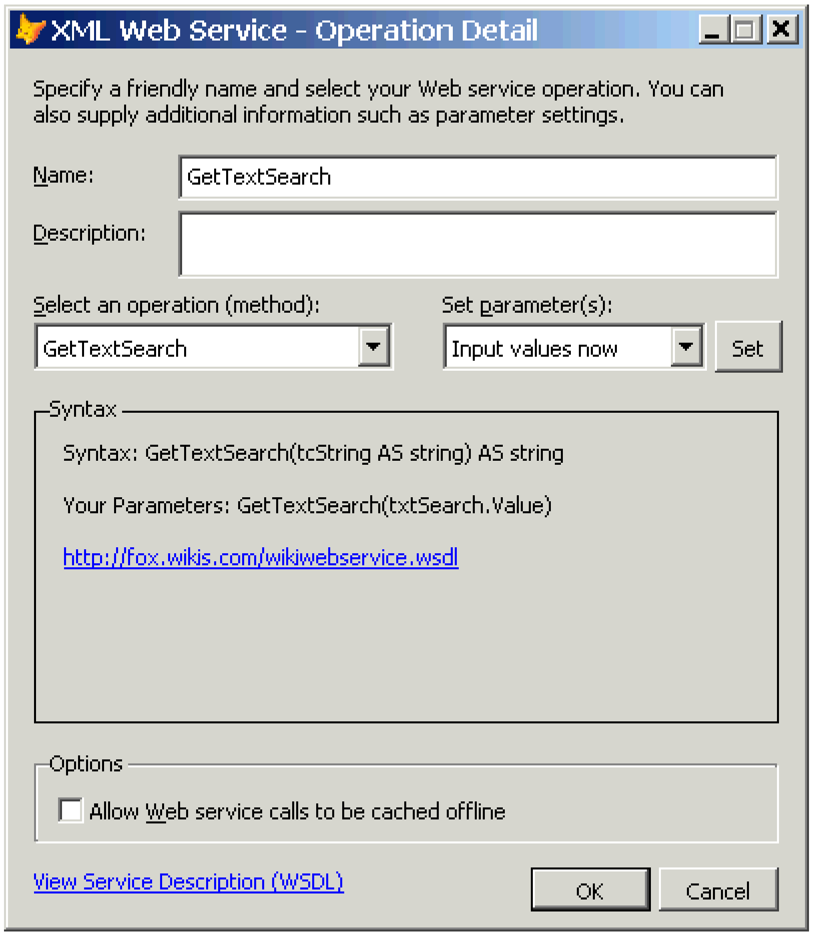
Clients

Specify controls (Clients) that will consume your Operation above. You can include detail such as the property to bind to your Web service call.

grdResults

To define an operation, click on the Add button. There are several properties you can set for an operation in the Operation Detail dialog (shown in **Figure 3**), including a friendly name and description. To select the Web service method to call, choose it from the methods combo box; for this example, choose the GetTextSearch method. If you turn on the Allow Web service calls to be cached offline option, WSHandler will cache the results of a Web service call and use those results when the Web service isn't available for a future call.

Figure 3. Use the Operation Detail page to specify which method of the Web service to call and what parameters to pass it.


XML Web Service - Operation Detail

Specify a friendly name and select your Web service operation. You can also supply additional information such as parameter settings.

Name: GetTextSearch

Description:

Select an operation (method): GetTextSearch

Set parameter(s): Input values now

Syntax

Syntax: GetTextSearch(tcString AS string) AS string

Your Parameters: GetTextSearch(txtSearch.Value)

http://fox.wikis.com/wikiwebservice.wsdl

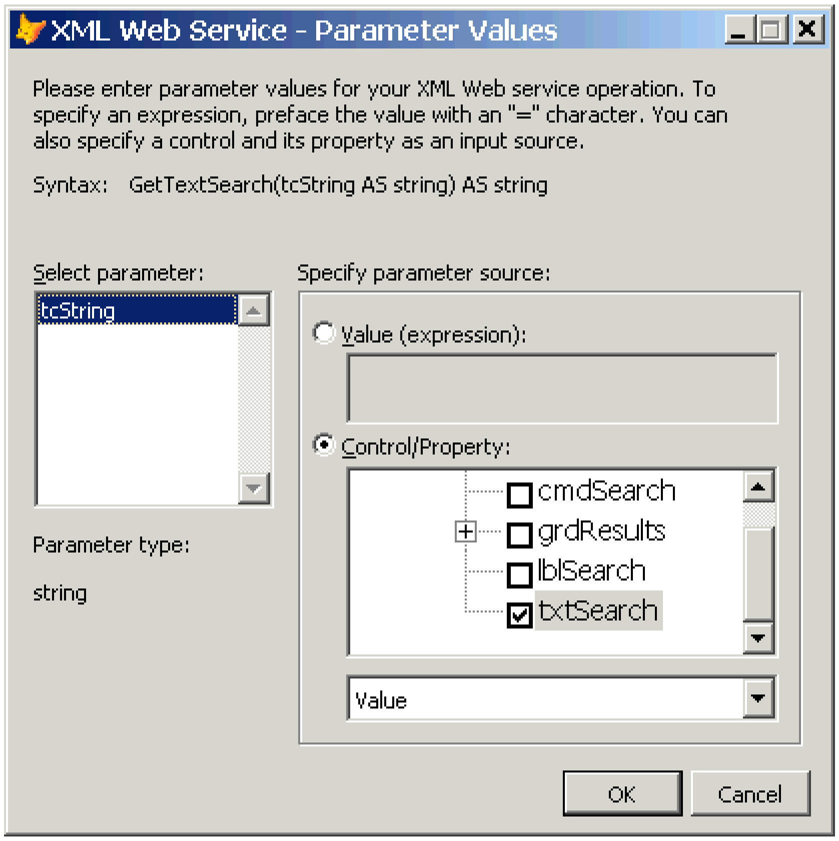
Options

Allow Web service calls to be cached offline

View Service Description (WSDL)

Use the Set parameter(s) combo box to specify how the values for the parameters passed to it are obtained. The choices are Input values now, Programmatically set at run-time, and Prompt at run-time. Programmatically set at run-time means you will set the value in code. Use Prompt at run-time if you want VFP to display a dialog in which the user can enter the values; this probably is more useful for developer tools rather than end-user applications. The term "Input values now" is somewhat misleading, since it doesn't necessarily mean you enter hard-coded values (although you certainly can do that). When you select that choice, the Set button is enabled. Click on it to display the Parameter Values dialog (see **Figure 4**).

Figure 4. In the Parameter Values page, you specify the source of each parameter for the Web service method.



XML Web Service - Parameter Values

Please enter parameter values for your XML Web service operation. To specify an expression, preface the value with an "=" character. You can also specify a control and its property as an input source.

Syntax: GetTextSearch(tcString AS string) AS string

Select parameter: tcString

Specify parameter source:

Value (expression): [✓ Control/Property]

cmdSearch, grdResults, lblSearch, [✓ txtSearch]

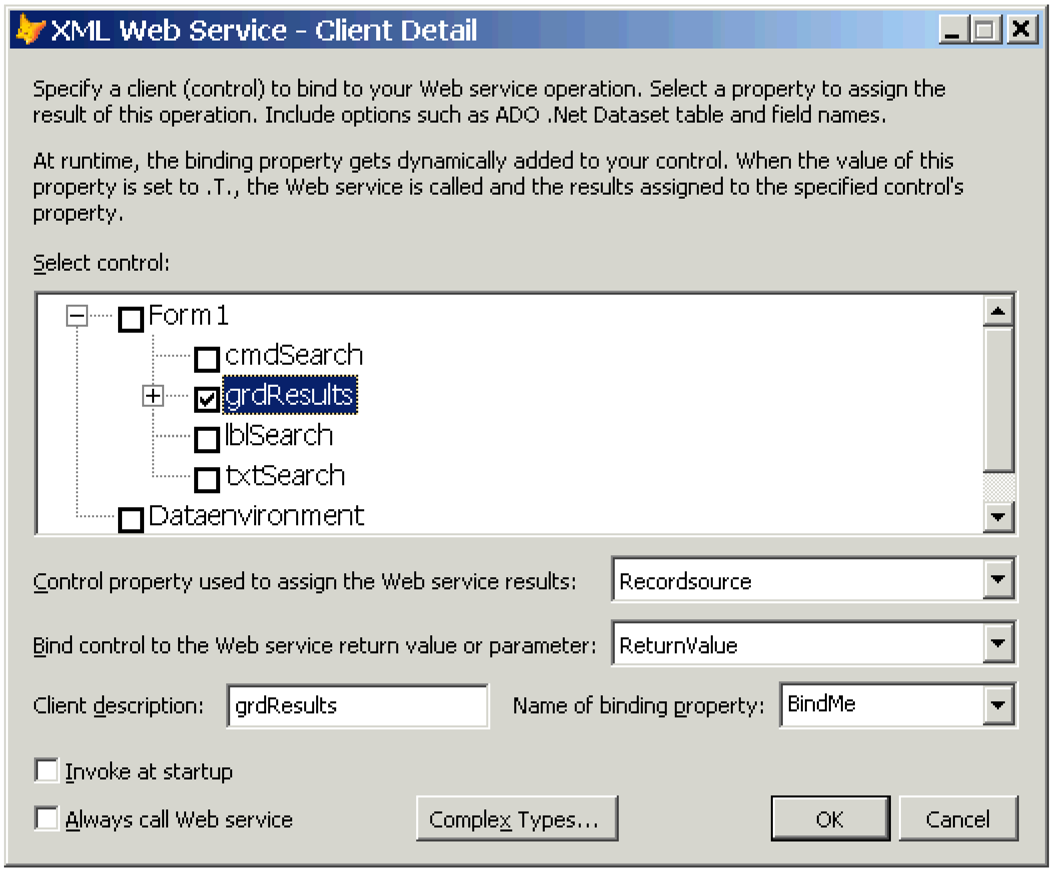
Value

Parameter type: string

To specify a value for a parameter, select it in the Select parameter list, choose Value (expression) as the parameter type, and enter the value into the edit box below this option. To indicate the value is an expression that should be evaluated at runtime, precede it with "=". To obtain the parameter value from another control, choose Control/Property, put a checkmark in front of the appropriate control, and select the property that holds the value from the combo box below the control list. For the WikiWebService, we'll bind the tcString parameter to the Value property of the txtSearch textbox. That way, the Web service will automatically search for whatever we type into the textbox.

To define a client for an operation, click on the Add button beside the Clients list in the Operations page. Use the Client Detail dialog, shown in **Figure 5**, to specify which control in the container is the client for the operation. You can then indicate which property of the control receives the result, and whether the property is bound to a parameter or the return value of the Web service method.

Figure 5. The Client Detail page is used to specify where the results of the Web service operation should go.


XML Web Service - Client Detail

Specify a client (control) to bind to your Web service operation. Select a property to assign the result of this operation. Include options such as ADO .Net Dataset table and field names.

At runtime, the binding property gets dynamically added to your control. When the value of this property is set to .T., the Web service is called and the results assigned to the specified control's property.

Select control:

- [ ] Form1
  - [ ] cmdSearch
  - [✓] grdResults
  - [ ] lblSearch
  - [ ] txtSearch
- [ ] Dataenvironment

Control property used to assign the Web service results: Recordsource

Bind control to the Web service return value or parameter: ReturnValue

Client description: grdResults

Name of binding property: BindMe

[ ] Invoke at startup

[ ] Always call Web service

If you turn on the Invoke at startup option, the operation executes when the container instantiates. Otherwise, you'll have to invoke it manually. If the client object is a CursorAdapter and you turn on Invoke at startup, you're warned you should set the form's BindControls property to False to prevent any problem with controls bound to the CursorAdapter. WSHandler automatically sets BindControls to True after the CursorAdapter has been filled with data from the Web service so data binding can occur.

If you turn on the Always call Web service option, the client will always call the Web service method. Otherwise, WSHandler will reuse the results when it detects multiple clients call the same operation at the same time (such as when the container instantiates).

WSHandler adds the property you specify in the Name of binding property option (the default is "BindMe" but you can also choose any custom property or method of the client control) to the client control at runtime if it doesn't already exist. The property acts like a switch: WSHandler uses BINDEVENTS() (a new function in VFP 8 that provides event binding between VFP objects) so it's notified when the property's value changes. Setting its value to .T. causes the WSHandler object to call the Web service method and put the results into the client control.

If the Web service you want to call returns complex XML, such as objects serialized as XML, ADO.NET DataSets, or VFP cursors converted to XML via CURSORTOXML(), click on the Complex Types button. In the Complex Types dialog, click on the Query button to call the Web service method immediately; the builder will then examine the return value and fill in the rest of the controls. For example, if the Web service returns an ADO.NET DataSet, the Dataset option is selected, the Dataset tables combo box is filled with the names of the tables in the DataSet, and the Table fields combo box shows the fields in the selected table. (In the case of a grid client, Table fields is ignored and the grid is bound to the selected table.) If the Web service returns an XML string that can be converted into a VFP cursor (as is the case with the WikiWebService used in these examples), you'll be informed of that and asked if you want to view the cursor's schema. If you choose Yes, the Dataset tables and Table fields combo boxes are filled in. The Attach button is enabled if the client object is a CursorAdapter and you click on Query to query the Web service. In that case, if you click on the Attach button and choose Yes in the subsequent dialog, the builder will update the CursorSchema property of the CursorAdapter to match the schema of the selected table.

For our example form, choose grdResults in the Client Detail page so the results will be displayed in a grid. The WSHandler object will automatically convert the XML returned from the GetTextSearch method of the Web service into a cursor and bind the grid to that cursor.

That's it for the builder, so choose OK in each open dialog until the builder is closed. The last thing we have to do is tell the WSHandler object when to call the Web service. That should happen when we click on the Search button, so put the following code into the Click method of that button:

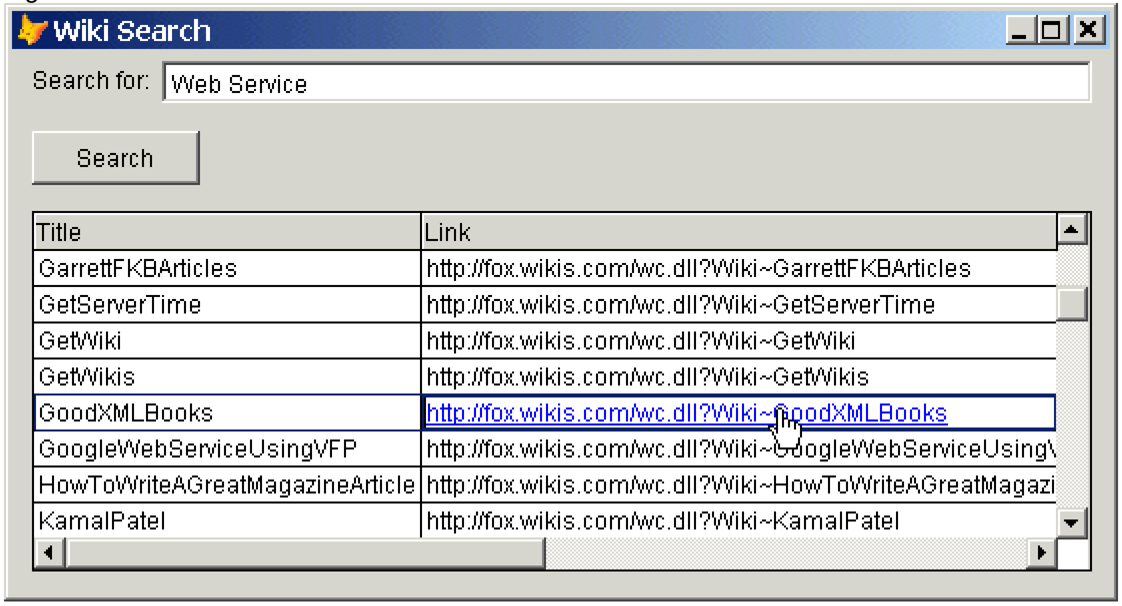
```
Thisform.grdResults.BindMe = .T.
Thisform.grdResults.AutoFit()
```

Setting the BindMe property of grdResults (remember, the WSHandler object will add this property to the grid when the form is instantiated) to .T. starts the operation because that was the property specified in the builder. The WSHandler object calls the GetTextSearch method of the FoxWiki Web service, passing it the contents of the Value property of the txtSearch text box, creates a cursor from the returned XML, and puts the name of that cursor into the RecordSource property of the grdResults grid. Calling the grid's AutoFit method ensures the grid columns are adjusted to fit the received data.

Run the form, enter a search string, and click on the Search button. After a few seconds, the search results are shown in a grid that auto-fits the columns to its contents and provides hyperlinks to the Wiki articles; Ctrl-Click on the hyperlink to bring up the document in your browser. See **Figure 6** for an example of what this form looks like. Not bad for just two lines of code!

Note: if you didn't build this form from scratch but want to use the one provided with the Subscriber Downloads, instead of running this form directly, run WikiSearch.PRG; it ensures the path to the WSHandler object in WikiSearch.SCX is correct for your system.

Figure 6. WikiSearch.SCX searches the FoxWiki for a particular string and displays the results in a grid.



Web Service

Title	Link
GarrettFKBArticles	http://fox.wikis.com/wc.dll?Wiki~GarrettFKBArticles
GetServerTime	http://fox.wikis.com/wc.dll?Wiki~GetServerTime
GetWiki	http://fox.wikis.com/wc.dll?Wiki~GetWiki
GetWikis	http://fox.wikis.com/wc.dll?Wiki~GetWikis
GoodXMLBooks	http://fox.wikis.com/wc.dll?Wiki~GoodXMLBooks
GoogleWebServiceUsingVFP	http://fox.wikis.com/wc.dll?Wiki~GoogleWebServiceUsing\
HowToWriteAGreatMagazineArticle	http://fox.wikis.com/wc.dll?Wiki~HowToWriteAGreatMagazi
KamalPatel	http://fox.wikis.com/wc.dll?Wiki~KamalPatel

Summary

XML Web services are easier to use than ever in VFP 8. The WSHandler class allows you to consume XML Web services both programmatically and visually, including handling the wiring of form controls to a Web service. Start considering what role XML Web services will take in your distributed applications.

Doug Hennig is a partner with Stonefield Systems Group Inc. He is the author of the award-winning Stonefield Database Toolkit (SDT) and Stonefield Query, author of the CursorAdapter and DataEnvironment builders that come

with VFP 8, and co-author of “What’s New in Visual FoxPro 8.0”, “What’s New in Visual FoxPro 7.0”, and “The Hacker’s Guide to Visual FoxPro 7.0”, from Hentzenwerke Publishing. Doug has spoken at every Microsoft FoxPro Developers Conference (DevCon) since 1997 and at user groups and developer conferences all over North America. He is a Microsoft Most Valuable Professional (MVP) and Certified Professional (MCP). Web: www.stonefield.com and www.stonefieldquery.com Email: dhennig@stonefield.com