Course XXX-07

Using Visual InterDev and Visual FoxPro

By Whil Hentzen

Access to Data

Access to data on the Web works differently than in the traditional LAN-based architecture that you're used to. Let's examine why, and then discuss the ramifications.

HTTP: An analogy

Unlike the LAN, where you are either extricably linked to the data (in a pessimistic locking scenario) or a mere keystroke away (in an optimistic locking scenario), the browser has no direct connection to the database.

This relationship is similar to a word processor that retrieves a file from a network. You Open the .DOC file, and now it's in memory on your own PC, regardless of where the original.DOC file was. There isn't a "live" or "continuous" connection between the words you're editing and the file on disk. You need to execute a File Save in order make the connection again.

And during the File|Save, what happens is that you are actually creating a brand new connection. You can prove this because if you took out your handy wire cutters, you could be working on the .DOC in your PC's memory, and have a pal snip the wire between your PC and the rest of the network. You might not notice until you went to save – you'd then find out that your connection to the network was gone.

Same thing with the Web – your connection is temporary – lasting only long enough to grab the goodies and get out. Let's be more specific:

1. The browser (the "client") sends a request to the web server. This is analogous to executing File Open in our word processor.

2. The web server (a piece of software such as IIS running on a separate box) recieves the request, and goes to fetch the data as described in the request.

3. This data may be on the same box, in a simple format like a set of DBFs or an MDB, or it may be on yet another separate box – the database server. In either case, the data is returned to the web server.

4. The web server returns the data to the client (browser), and the connection is broken.

5. The browser gets a file/chunk of data from the server, and displays it for the user, just as Word can display the contents of a .DOC file in a pleasing format for the user.

Obviously, you're going to through a paradigm shift when it comes to providing access to data on the Web. What are the ramifications?

1. You'll be working with set-based data sets instead of the whole enchilada.

2. The interface capabilities are significantly more limited than what you're using to in VFP.

3. HTTP protocol is disconnected so you don't have the same performance capabilities.

Set based data sets

The Web is very much a client-server mechanism, and the connection can be very tenuous at times. Even more so than a traditional client-server relationship, you want to minimize traffic over the wire. Get used to working with one or a few records, as opposed to being able to SKIP or BROWSE through a couple hundred thousand records (and their myriad relatives.)

Limited interface capabilities

During the early 1990's, we bemoaned the limitations of the FoxPro 1.0 and 2.x user interfaces – a minimal number of events and limited control over what the user was doing and what you were able to control. A far cry from the incessant READs of dBASE and FoxBASE interfaces, to be sure, but we spent an incredible amount of time devising workarounds. With the form designer tools of VFP (and similar tools like VB), a whole new world opened up to us: extremely granular control over every aspect of the interface.

As they said in the movie Mad Max Beyond Thunderdome, "No matter where you go, there you are." And here we are, back where we were. While accelerating, the capabiltiies of today's browsers simply don't match what we have on the desktop with our other tools. Many have compared them to dBASE III or FoxBASE. I won't argue where we stand on the ladder of evolution – but we've taken a few steps back for the time being.

HTPP protocol

You'll have to rework your data access mindset and strategies, being more careful about what you're asking for and when. With VFP's blinding speed, you could get away with being lazy – don't have the right data? Just grab another batch through a SQL SELECT. Don't know what you want? No matter – grab as much as you want, and then winnow down the result set later.

You'll approach your data differently now.

Plumbing

Data access architecture

Tools available in Visual InterDev

Database projects. A project that includes tools to build and manage your databases separately from the rest of your Web pages. Included as a component of a Visual InterDev solution.

Data View window. Provides a live view of the data to which you're currently connected.

Visual Database Tools. Tools that allow the management and querying of a database graphically. Database Designer handles SQL Server and Oracle databases; Query Designer creates SQL statements; View Designer creates Views.

Data Environment. Connection strings are descriptions of how to connect to a specific database. Data commands represent specific record sets that you'll view in your Web page. Most useful for VFP developers accessing DBF-based data.

Data-bound controls. Controls put on a Web page that are automatically bound to fields in a database record.

Source control for database objects.

Production site requirements

What you need and where you can get it.

First, start off with a clean machine – all the way down to formatting the drive and installing everything from scratch. It's simply too easy for remnants of something that was supposed to have been completely uninstalled to be left around and cause trouble when you least expect it.

Second, install NT Server 4.0. Internet Information Server 2.0 comes with NT Server.

Third, install Service Pack 3 for NT 4.0. This includes IIS 3.0. SP3 is available for download at www.microsoft.com, among other places. (Of course, you need NT 4.0 to use it.) You'll now see Programs/Microsoft Internet Server (Common) on the Start menu.

Fourth, Visual Studio Enterprise 6. You can get SP3 and IE4 from this package as well. Then you are asked which type you want to install: Custom, Products or Server Applications. Select Server. This will install a number of items, including NT Option Pack 4.0, Front Page Server Extensions, Visual FoxPro Server, Visual InterDev Server, and MS Data Access Components.

NT Option Pack includes updates to some components, including from IIS3 to IIS4.

You'll now see Programs/Windows NT 4.0 Option Pack/Microsoft Internet Information Server/Internet Service Manager, which brings up Microsoft Management Console (MMC).

Development software - Server

Same as Production Server

Development software - Workstation

Now let's look at what you'll want to do for your workstation.

First, again, start off with a clean machine. This is probably more important since you're more likely to have installed something goofy on a workstation. It's also more of a nuisance, again, because you're more likely to have a ton of things already installed.

Install Windows NT Workstation 4.0, and Service Pack 3 for NT 4.0.

Install Visual Studio Enterprise, Custom. You'll want to select Internet Explorer 4.0 as well as Visual FoxPro and Visual InterDev.

You'll also want to install MSDN – comes with Visual Studio – be sure to have several gigabytes free, because you'll want to install the whole thing.

Hints

Use a server-workstation network setup. You can play around with everything on a single machine, but for actual development, you'll want to use a local and a master.

Start with a clean install. FDISK is your best friend when preventing problems from the get-go.

Mentally get ready to install more than once. Get all of your files ready, and keep them in a single place. You probably won't need to, but better to be prepared.

Document what you do, what happens, and what you've accomplished. It's very easy to install something the third or fourth time, and not realize that you forgot Step 7. Be particularly sure you document which version of a tool you're installing.

What is Visual InterDev?

Visual InterDev is to an HTML editor as what Visual FoxPro is to EDLIN.

From the documentation:

"Visual InterDev is a Web development tool designed for programmers and Web teams who want to create:

- Data-driven Web applications using any data source supported by ODBC or OLE DB. This covers all the databases from major vendors, including Microsoft, Oracle, Informix, and Sybase.

- Broad-reach Web pages using HTML and script in Web applications that take advantage of the latest advances in browser technology such as Microsoft® Internet Explorer 4.0, Dynamic HTML and multimedia features.

- Integrated solutions that can include applets or server-side COM components created in Microsoft Visual Basic®, Visual C++®, Visual J++™ and Visual FoxPro®. "

Just as FoxPro has provided a complete environment for building LAN applications for the past decade, with a project manager, tools to build forms, menus, and reports, and then allowed the user to package a deliverable, Visual InterDev does the same – and more – for high-end data-driven Web applications.

But there's more. In addition to providing a project manager that connects an HTML editor, data connections, and COM components, Visual InterDev also does a great job coordinating the work of multiple developers, and delivering finished applications to a production server.

The Visual InterDev Interface and Tools

Toolbox. Three views accessed via tabs in the bottom of the window. Document Outline displays a map of the documents in the project. Toolbox contains five areas: General, HTML, Design-Time Controls, ActiveX Controls, and Server Objects. Script Outline .

Project Explorer. Similar in purpose to the Project Manager in VFP. Displays a hierarchical listing of all components of a solution/project.

Property Window. Similar in purpose to the Properties Window in VFP. Displays a list of properties for the currently selected component.

HTML Editor. Three views: Design, Source, and Quick View.

Visual InterDev Structure

This entire discussion is predicated on the assumption that you'll be working on a workstation, that your workstation is connected to a server; and both are running the appropriate Visual InterDev components.

Application Hierarchy

A Visual InterDev application is called a solution, which may be made up of one or more projects. For our purposes, we'll be talking about a single project for the remainder of this discussion.

A project consists of, on the top level, a default page (either HTML or ASP), a GLOBAL.ASA file, and a SEARCH.HTM file. A number of directories are also automatically created for a Visual InterDev project, depending on which options are selected. These include _private, _scriptlibrary and images.

Plumbing

The server maintains a master copy of the entire solution. When you open the project on your local workstation, you download a copy of the entire set of files to your workstation. Thus, you are working on a mirror image of the entire project. Other developers can also download the project to their workstation, and work on their own local version independently.

When you finish, you send your changes back to the server, which then incorporates changes into the master project and reconciles changes as needed.

Thus, components of Visual InterDev are installed both on the Server and on the Workstation. You'll also want other development tools installed on your workstation as you need them. For example, you may want Visual FoxPro, Visual Basic and Visual C++ to build components.

There are, as of this writing, some issues about where the actual database is located. You may find it useful to keep identical copies of the database in the same fixed path both on the server and on the workstation during development. More on this when we talk about connection strings.

Creating a Project

In order to create a new Visual InterDev project, start up both the Server and the workstation, and making sure the Web Server is running on the server.

Visual InterDev will create a series of files on your server under the Web Server's home directory. You can specify location this by turning to your server, selecting Start, Programs, Windows NT 4.0 Option Pack, Microsoft Internet Information Server, Internet Service Manager to bring up the Microsoft Management Console (MMC – learn this acronym well!). In the left pane tree view, you can drill down from Console Root, Internet Information Server, <name of your server>, Default Web Site, and right-click on Default Web Site. Select the Properties menu option, and then the Home Directory tab. If you've chosen the first radio button (content should come from a directory located on this computer), you can type in or browse a local path to select a directory or subdirectory. This will be the "root" of the Web Server, and all files will be located here.

[Note: Later on, when retrieving a page from the web server, you'll use this location as the starting point, simply prefixing the page name with the name of the server. For example, suppose your web server (machine) name is HERMAN, you've specified the D:\APPS\WEBSTUFF as the local path, and the page you want to access is called IT.HTML. You would place IT.HTML in the WEBSTUFF subdirectory. Then, in your browser on your workstation, you'd enter the URL: http://HERMAN/IT.HTML. The web server's home directory setting knows that IT.HTML is in D:\APPS\WEBSTUFF.]

Starting up a new project

Bring up Visual InterDev, and select the File, New Project menu option.

The New Project Wizard displays, asking you if you'd like a Visual Studio or Visual InterDev project, and if you've selected the Visual InterDev project icon, whether you'd like a New Web Project or the Sample App Wizard. Select the New Web Project icon. A name for the project will be displayed along with a location.

[Note: I've found it much less confusing to keep a directory structure on the local workstation that is identical to the web server machine.]

Rename the project if desired, and select the Open button. Now the fun begins. The Web Project Wizard displays, and asks for the server that you'd like to use.

Where is that server?

One of the most annoying events in this whole process is when you can't find your server. While not exhaustive, here are a few possible situations you might find yourself in.

First, if you type in the wrong name, you'll get a grey dialog box "Contacting Web server…" and eventually, a message box stating "Unable to contact web server http://name. You'll get an OK button; resist the temptation to say, "NO! It's NOT OK" because doing so won't do any good. You can find the name of your server by looking in the MMC tree view, and seeing what the name of the server is under the Internet Information Server node.

A second possibility is that you don't have Front Page Server Extensions installed. In this case, you'll swear that you're entering the right server name, but Visual InterDev still isn't finding it.

To install Front Page Server Extensions, bring up Program Files\Windows NT 4.0 Option Pack\Microsoft Internet Information Server\FrontPage Server Adminstrator, then select button in upper left corner and install them.

A third possibility is that you don't have sharing set up properly. Right-click on the directory you've specified as your Web Server Home Directory and make sure that Web Sharing is enabled.

A fourth possibility is simply that your network isn't set up properly, or that you simply haven't rebooted everything after changing settings.

The other question you'll have to answer is which mode you want to work in – Master or Local. Master mode means that you'll be working on a local set of files, but Visual InterDev will automatically keep the files in sync.

The second step is to decide whether you want to create a new Web application or to connect to an existing application. In this case, you'll create a new application.

The third and fourth steps are to select a theme and a layout – both are optional, and you may want to bypass these for your first few projects, as they are additional, time-consuming steps.

Finally, you'll select Finish, and you'll see a series of messages indicating that the project is being created. You'll also see a list of files in the Project Explorer window grow. What is happening is an entire project shell is being copied to the server's Home Directory.

Finally, you'll want to create a home page. Right-click on the server name/project name node in the Project Explorer window project tree view, select Add…, and then identify the type of object you want to add – an HTML page, an ASP page, or whatever. Name the page, and it will open an HTML editor window. You can identify this page as the start page by right-clicking on it's node in the Project Explorer window tree view, and selecting the Set As Start Page menu option.

Select File, Save All (unless you're brave or foolhardy), and you've now got a brand new project ready to work on.

Connecting to Data

First things first. You gotta have some data. We're going to use a stand-alone Visual FoxPro table to begin with, in order to show what the procedure is and how the mechanism works. We'll make it more complex later. We've got a single .DBF in the Home Directory both on the local machine and on the web server. We need it on the local machine in order to create a connection string, and on the server because that's where the web server is going to look for it when you request the page.

The first thing we're going to do is create a mechanism for Visual InterDev and our web application to be able to identify the database and determine how to connect to it.

Creating a DSN

First, you'll need to create a Data Source Name (DSN) that identifies how to connect to the database. (You'll see a number of existing DSNs when you create Connection Strings later – don't use them! They're there for other applications (like Office) to use. You'll need to use your own!)

Use Control Panel / ODBC and create a **File** DSN and point to the database that you're interested in. This DSN will be saved in Program Files/Common Files/ODBC/System Files. Remember that this DSN points to a specific database in a specific location. If you move it, you'll need a new DSN!

In the Control Panel, select the ODBC applet, and go to the File DSN tab.

Press the Add button, select the driver from Create New Data Source dialog, note the difference between FoxPro Driver and Visual FoxPro Driver, then Browse to the name of the DSN – NOT THE DBF. I think the wording in the dialog is confusing. You are creating a DSN in Program Files\Common Files\ODBC\… Then Finish.

Next, you'll get a ODBC Microsoft FoxPro Setup dialog. Use this to select the directory your DBF is in. You're back at ODBC … Administrator. Click OK.

A DSN is simply a text file – you can open it up in Notepad and take a look at it if you desire.

The DSN contains properties that we're going to use in our Connection String, which can be thought of as a variable in Visual InterDev. This variable contains the location to the database, and the web application hands this to the web server to tell it where to find the data.

Creating a Data Connection object

Add a Data Connection by right-clicking on the GLOBAL.ASA file in the Project Explorer window and selecting Add Data Connection.

You'll get a Select Data Source dialog with two tabs. The first tab says "File Data Source" and voila! You should see your DSN you just created in the list. Pick it!

Next, you'll get a Connection1 Properties dialog. The Connection Name text box has "Connection1" in it, and the Use Connection String option button is selected. You'll see the driver for the File DSN you selected in the previous step.

You can rename the connection string to something else if you like – I usually use a name that matches up to the project name. Press the Test Connection button, just to be sure, and then Apply and OK.

Now the Connection String is part of the Visual InterDev project, so the Web application knows where the data is, and how to connect to it. As a point of interest, the DSN from which the Connection String was created is no longer of any interest. You could actually even delete the .DSN file from Program Files/Common Files/ODBC/System Files (using Control Panel, of course) if you wanted to.

Note that if you used a System or Machine DSN, the application would be bound to the DSN, and thus, you would have to recreate the DSN on every development machine working on the application – and the Web Server machine itself.

Also note that you can have multiple data connections – if you need to attach to multiple data stores, you can add a data connection for each of them.

Creating a Data Command

A data command is a reusable object that defines a set of data to work with. This set of data may be represented by a table, a query, or a view, for example.

Your blank ASP page, created earlier, should look something like this when you view the HTML source (if you're in the Design tab or Quick View tab in the HTML editor, well, it would be pretty uninteresting):

```
<%@ LANGUAGE=VBScript %>
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
</HEAD>
<BODY>

<P> </P>

</BODY>
</HTML>
```

Put your cursor in between the BODY and /BODY tags. Then open the Design-Time Controls tab in the Toolbox, and double-click on RecordSet. You'll be prompted as to whether or not you want to enable the Object Model, and by gum, you sure do. The RecordSet control will be placed on the page. It will be named "DTCRecordset1."

Now set the properties for this Data Command. Set the Connection to the connection you created earlier, the Database object to Tables, and the object name to the name of the table you have selected. Close the properties window.

Placing data objects on page

Now add a couple of controls, say, a text box and a check box. To do so, place your cursor under the Recordset control, and then double-click on the text box and check box controls in the Toolbox. You'll get one of each.

Set the properties for these controls. First, make sure that the recordset is set to DTCRecordset1. Map the text box to a character field in the table and map the check box to a logical field in the table.

Running

Save and preview in your browser. The first record in your dataset will be displayed.

Manipulating Data

Navigating through existing data

One of Visual InterDev's controls is a RecordsetNavBar control that provides the ability to move from record to record in the recordset. To use it, place your cursor in the page where you want to NavBar control to appear, and then double-click on the RecordsetNavBar control in the Toolbox. Then set the Recordset property of the control to the Recordset used earlier.

Native Editing/Adding/Deleting

Automatic updates based on navigation

Simply set the RecordsetNavBar updateOnMove property to True. Note that this essentially does a save regardless of whether the data was changed, which may cause unwanted overhead.

Adding

First, you'll add a button to the page, such as "Add." Then you'll initialize a new record in the click event of the button. At this point, the user can enter data as they desire. Finally, in order to save, the user will press a Save button, and run the same process that they do when saving an edited record.

Deleting

First, you'll add a button to the page, such as "Delete." Then call the recordset's delete method to eliminate the current record. The next record in the recordset will be shown by default.

Editing/Adding/Deleting options with script

The Script Outline tab in the Toolbox allows access to a variety of controls and their methods. Double-clicking on a method places a blank subroutine in the HTML editor for you to enter your own custom script.

Forcing updates through script

Forcing updates manually requires the use of script in methods attached to the recordset object. First, you'll call the setValue method in order to copy data values into the current record – for every field, similar to when you used to SCATTER/GATHER on a field by field basis. Then you'll call the updateRecord method in order to save the current record to the database.

```
Sub btnSave_onclick

  rsUsers.fields.setValue("FirstName", txtFirstName.value)

  rsUsers.fields.setValue("LastName", txtLastName.value)

  rsUsers.fields.setValue("OrgName", txtOrgName.value)
```

```
  rsUsers.fields.setValue("LastUpdated", date)

  rsUsers.updateRecord

End sub
```

Adding records

In the click method of the Add button, you can call the recordset's addRecord to add a new blank record in the recordset. Then, initialize the fields on the form. For example:

```
Sub btnAdd_onclick

  rsUsers.addRecord

  rsUsers.fields.setValue("LastName", "")

  rsUsers.fields.setValue("OrgName", "")

  rsUsers.updateRecord

End sub
```

Note that we haven't populated the LastUpdated field with a value. This is because the Save method will do that for us. All we're doing is essentially SCATTER MEMVAR BLANK.

Deleting records

The code for deleting a record, and then moving to the next record, looks like this:

```
Sub btnDelete_onclick

  rsUsers.deleteRecord

End sub
```

Note that there is no code to move to the next record. This is native behavior. You could force the positioning of the record pointer by inserting code to do so. For example,

```
DTCRecordset1.deleteRecord;

DTCRecordset1.movePrevious;
```