

Frequently Asked Questions (FAQs) for *Murach's VB.NET Database Programming with ADO.NET*

Why can't I attach the databases provided with the book to MSDE?

The batch file (DB_Attach.bat) that's included in the download for our book that attaches the sample databases to MSDE was designed to work with a server that was installed from the Visual Studio 2002 CDs. In that case, the server is installed with a name that consists of your computer name, followed by \VSDotNET. If you didn't install MSDE from the Visual Studio 2002 CDs, you may need to change the server name in the batch file.

If you're using Visual Studio 2003, for example, you have to download MSDE from Microsoft's web site. Then, you have to install MSDE by entering a command at the command prompt. One of the parameters on that command is INSTANCENAME. Microsoft suggests that you specify VSDOTNET for this parameter. If you do that, the server will be given a name that consists of your computer name, followed by \VSDOTNET. In that case, the batch file we provide will work fine.

If you specify another value for the INSTANCENAME parameter, however, or if you omit this parameter, you'll need to edit the DB_Attach.bat file before you execute it. To do that, open the file in Notepad. As you'll see, it consists of a single command:

```
OSQL -S %COMPUTERNAME%\VSDotNET -E /i db_attach.sql
```

To change the server name, replace the VSDotNET specification with the name you specified on the INSTANCENAME parameter. Or, if you omitted this parameter, just omit the \VSDotNET specification. Then, close and save the file. Now, when you execute this file, it should attach the sample databases.

If you installed MSDE from another source, or if you want to attach the databases to an instance of SQL Server that you have access to, you'll want to find out what the server name is so you can change the batch file accordingly. The easiest way to do that is to use the Server Explorer from within Visual Studio as shown on page 535 of our book. After you expand the SQL Servers node, you should see the name of your server. Then, you can replace the server name specification in the DB_Attach.bat file (%COMPUTERNAME%\VSDotNET) with that name.

If you're using our student materials, you'll also need to modify the DB_Attach.bat and DB_Restore.bat files that attach and restore the databases used by the exercises and the projects. If you're using our instructor materials, you'll need to modify the DB_AttachMurach.bat, DB_RestoreMurach.bat, DB_AttachPayables.bat, and DB_RestorePayables.bat files that attach and restore the databases used by the applications in the book; you'll need to modify the DB_AttachMurachBooks.bat and DB_RestoreMurachBooks.bat files that attach and restore the database used by the exercises; and you'll need to modify the DB_AttachTechSupport.bat and DB_RestoreTechSupport.bat files that attach and restore the database used by the projects.

How can I bind a data grid to a relationship through code?

To bind a data grid to a relationship through code, you set the DataSource property to the dataset that contains the relationship, and you set the DataMember property to the relationship. The trick is that you must refer to the relationship through its parent table. For example, suppose you define a relationship named VendorsInvoices that relates rows in a parent table named Vendors to rows in a child table named Invoices. Then, if the relationship and tables are stored in a dataset named DsPayables1, you can bind a data grid named grdInvoices to the relationship using this code:

```
grdInvoices.DataSource = DsPayables1
grdInvoices.DataMember = "Vendors.VendorsInvoices"
```

Why do I get a “Login failed” error message when I try to connect to a SQL Server database from a web application?

This error typically occurs when you use Windows NT Integrated security to connect to the database. To understand why, you need to realize that ASP.NET runs as a separate Windows process with its own login name: ASPNET. When you try to access a SQL Server database from a web application, then, ASP.NET will try to use this login name to log in to SQL Server. For that to work, you first have to create a Windows user named ASPNET, and you have to create a SQL Server user account named ASPNET and grant it access to the database. A simpler solution, however, is to not use Windows NT Integrated security. Instead, you can name the specific SQL Server account you want to use when you create the connection. If you’re using MSDE on your own system, for example, you can probably just use the default system administrator account, sa. Then, your connection string will look something like this:

```
Dim sConnectionString As String _
    = "Data Source=Anne\VSDOTNET;" _
    & "Initial Catalog=Payables;" _
    & "Integrated Security=False;" _
    & "User ID=sa"
```

If you make this change and you continue to get a “Login failed” error message, it’s probably because SQL Server authentication isn’t enabled on your server. In that case, you’ll need to enable it before you can access the database using the sa account. To do that, you’ll need to either reinstall MSDE and specify the appropriate switch to enable SQL Server authentication, or you’ll need to modify the registry. For information on using either of these techniques, please see the Microsoft Knowledge Base article at <http://support.microsoft.com/default.aspx?scid=kb;en-us;321698>.

When should I use Crystal Reports to join data from two or more tables?

The sample Crystal report in the book joins data from the Vendors and Invoices table within the report definition. If you’re using the push model, however, it’s more efficient to code a join in the Select statement that defines the data source. That’s because the join will be performed on the server. In contrast, when you join tables within a report definition, the join is performed on the client, which means that more data must be retrieved from the server. The only time you should join data within a report definition, then, is if you’re using the pull model. The report definition in the book includes a join simply for illustrative purposes.

How do I add text to a Crystal report?

To add text to a Crystal report, you use the Text Object component that's available from the Crystal Reports tab of the Toolbox when a report is displayed in the Crystal Report Designer. Just drag this component to the section of the report where you want it to appear, and then click in the box that appears and type the text you want to include in the report. You can also use the Line Object component in the Toolbox to add a line to a report, and you can use the Box Object component to add a box to a report.

How can I format data in a bound control?

You can format data in a bound control by handling the Format event of the control's Binding object. This event is raised after the data has been retrieved from the data source, but before it's assigned to the control. That lets you modify the value before it's displayed.

Before you can code an event procedure for the Format event, you must create a module-level variable for the Binding object that includes the WithEvents keyword like this:

```
Dim WithEvents UnitPriceBinding As Binding
```

Then, in the procedure for the Load event of the form, you assign the Binding object for the control that's bound to the data you want to format to this variable like this:

```
UnitPriceBinding = txtUnitPrice.DataBindings("Text")
```

Here, the Binding object for the Text property of a text box control named txtUnitPrice is assigned to the UnitPriceBinding variable.

Next, you create an event procedure for the Format event of the Binding object using the drop-down lists in the Code Editor window. The event procedure for the Format event of the txtUnitPrice text box, for example, would look something like this:

```
Private Sub UnitPriceBinding_Format(ByVal sender As Object, _  
    ByVal e As System.Windows.Forms.ConvertEventArgs) _  
    Handles UnitPriceBinding.Format  
    e.Value = FormatCurrency(e.Value)  
End Sub
```

Here, the value being sent to the bound control, which is passed to the event handler via the Value property of the e argument, is formatted as currency.

If the data from a bound control will be used to update the data source, you can use the Parse event of the control's Binding object to modify the data before it's sent to the data source. This event is raised when the user modifies the contents of a bound control, but before the data source is updated. Then, within the event procedure for this event, you can remove any formatting that you added in the Format event procedure. For example, to remove the leading dollar sign that was added to the value in the txtUnitPrice text box, you could code an event procedure like this:

```
Private Sub UnitPriceBinding_Parse(ByVal sender As Object, _  
    ByVal e As System.Windows.Forms.ConvertEventArgs) _  
    Handles UnitPriceBinding.Parse  
    Dim s As String = e.Value  
    If s.Substring(0, 1) = "$" Then  
        e.Value = s.Substring(1)  
    End If  
End Sub
```

This procedure starts by assigning the value that's being sent to the data source, which is passed to the event procedure via the Value property of the e argument, to a String object. Then, it uses

the Substring method of the String object to retrieve the first character of the string value. If the first character is a dollar sign, the Substring method is used to remove this character from the string value. Then, the modified value is used to update the data source.

Will the applications in this book work with Visual Basic 2003?

Although the applications in this book were written using Visual Basic 2002, for the most part, they work equally well with Visual Basic 2003. The exception is a problem with the Vendor Maintenance programs in chapters 4, 5, and 6 that was brought to our attention by one of our customers. If you delete a vendor using the Vendor Maintenance program in one of these chapters, you'll notice that the buttons on the form aren't enabled and disabled properly. That's because when you deleted a vendor in the 2002 version of these programs, the SelectedIndexChanged event of the Vendors combo box was fired, which caused the buttons to be set properly. In Visual Basic 2003, however, this event doesn't fire. Because of that, you need to set the buttons in the procedure that handles the Click event of the Delete button. Specifically, you need to add this code to the Delete procedure:

```
Me.SetMaintenanceButtons(True)  
btnUpdateDB.Enabled = DsPayables1.HasChanges
```

(In chapter 6, the name of the dataset is dsPayables instead of DsPayables1.) Then, the programs will work properly.

In addition to this problem, we have discovered only one minor difference between Visual Basic 2002 and Visual Basic 2003 that affects our book. This difference comes into play when you create a query using the Data Adapter Configuration Wizard as described on pages 50 and 51. With Visual Basic 2002, if you create a query that performs an update or delete operation and you don't include the primary key field in the query, the Configuration Wizard adds it for you automatically. With Visual Basic 2003, a dialog box is displayed asking if you want to add the primary key.