CHAPTER 7

Codeless Database Programming

In this chapter, I'm going to show you how to create a working Visual Basic database program without writing a single line of code. I'll use the ADO Data Control and some common controls found in nearly every Visual Basic Program.

Data Binding

One of the most powerful concepts in Visual Basic is the concept of data binding. By using data binding, you can delegate many of the details of moving data between your database and your program.

What is data binding?

Data binding is a technique that allows a data source to be tied to a data consumer. Then when the data values associated with the data source change, the updated information is reflected in the data consumer. Likewise, data values that are updated in the data consumer are passed back to the data source for updating.

In the case of a Visual Basic program, both the data source and data consumer are typically ActiveX controls, although other types of COM objects may be used (see Part III, Hardcore ADO, for more information about this subject). The classic data source in a Visual Basic program is the data control, while the classic data consumer is a text box control.

Note

One data control doesn't fit all: There are three different data controls, one for each object model (DAO, RDO, and ADO). The DAO control is simply known as the Data Control, while the RDO data control is also known as the RemoteData control. The ADO Data Control is also called the ADODC. You should pick the data control corresponding to the object model you wish to use and if you don't have a specific need to use the DAO or RDO object models, you should use the ADO Data Control to take advantage of the enhancements and features in the ADO object model.

How does data binding work?

Data binding is a two-step process. First, you create a data source and provide the information necessary for it to connect to the database server. Then you create one or more data consumers that are in turn connected to a specific database field returned by the data source. Since the data consumers are typically common controls such as the text box control, they are also known as bound controls.

The binding information is handled by setting various property values in the bound controls. Two properties in particular are very important when binding a control to a data source. The `DataSource` property identifies the name of the data source control, while the `DataField` property identifies which database field will supply the data for the bound control.

Then, when your application begins, the data control establishes a connection with the database server and each of the bound controls establishes a connection with the data control. Whenever the data control moves to a new record, the information in each of the bound controls can be updated automatically. Likewise, whenever the user changes a value in a bound control, the information is passed back to the data control, which in turn will automatically update the database when the user moves to a different record.

Connecting to the database

After binding the controls the user will interact with the data control, which needs to be connected to the database. Again, this is handled via a set of properties. The key property is the `ConnectionString` property, which holds the information needed to connect to the database server. This is a `String` value that contains four main pieces of information. The name of the database server and the name of the data provider are used to create a vehicle that can be used to link the program to the database server. Then the user name and password are used to authenticate the user and determine the user's privileges in the database.

Once you have a valid `ConnectionString`, you need to specify the source of the data. This is kept in the `RecordSource` property. This value can be the name of a database table, an SQL **Select** statement, or a stored procedure. Typically with a data control, you'll want to specify a table name, since it will make all of the records in the table available to the user. Also, by specifying a table name, you won't run into any complications with adding records to your database.

Cross-Reference

See Chapter 4 for details about **Select** statements.

Intrinsic bound controls

Visual Basic includes two main types of controls: *intrinsic* and *ActiveX*. Intrinsic controls are included with the Visual Basic runtime library and are always available to the Visual Basic programmer. While these controls are not true COM objects, they are much more efficient and the most frequently used. Many of these controls can be bound to a data control, including:

- **CheckBox**
- **ComboBox**
- **Image**
- **Label**
- **ListBox**
- **PictureBox**
- **TextBox**

Of these controls, probably the one you'll use most often is the `TextBox` control, since this control makes it easy to display a database value to a user and allows them to modify it. Other controls you might find yourself using are an `Image` or `Picture` control when you want to display a picture on your form and the `CheckBox` control when you want to display a `Boolean` value from your database. While you might think that the `ComboBox` control might also be heavily used, there is a more database-friendly ActiveX control called the `DataCombo` control that you will find yourself using in place of the `ComboBox` control in most applications.

ActiveX bound controls

Unlike the intrinsic bound control, the ActiveX bound controls are true COM objects and are external to the Visual Basic runtime libraries. Also, unlike the intrinsic bound controls, the ActiveX controls are more complex and have the ability to work with more than one database field at a time.

- **DataList**
- **DataCombo**
- **DataGrid**
- **DateTimePicker**
- **ImageCombo**
- **MaskEdit**
- **MonthView**
- **MSFlexGrid**
- **MSChart**
- **RichTextBox**

Of these controls, probably the most useful are the `DataCombo`, the `DateTimePicker`, the `MaskEdit`, and the `MonthView` control. All of these controls have one common feature: they make it harder for a user to enter an incorrect value into the program.

Keep the Garbage Out

You're probably familiar with the old expression "Garbage in, garbage out". One of the most important goals of a database programmer is to prevent bad data from getting into the database. When you get bad data in your database, you may find it hard to isolate and correct. I remember a situation where one slightly corrupted field in a database prevented a financial application from closing the books at the end of a fiscal year. It took nearly a week to track down and correct the bad piece of data. In the meantime, none of the other programs in the application would run correctly, and other processing involving the general ledger came to a complete halt. In a small business this may not be a big problem, but in a billion dollar a year organization, you can believe the top-level management wasn't very happy. So think about it this way, keeping bad data out of your database is a good way of keeping yourself happily employed.

Building the Codeless Program

Building a codeless program is an interesting exercise and one that is probably worth your time, especially if you are relatively new to Visual Basic. Bound controls will make your life much easier, especially in more complex applications. They definitely reduce the amount of code you have to write in your application and anything that reduces the amount of code you have to write appeals to the lazy programmer in everyone. For the rest of this chapter, I'm going to show you how to build a codeless program that allows you to access the Customers table in the database I designed in Chapter 3.

On the CD-ROM

It's your turn: This program is available on the CD-ROM as `\VB6DB\Chapter07\NoCode\Project1.vbp`. To run this program, simply create a Data Link File as described in Configuring the ADO Data Control below for your `ADODC1` control that reflects your database server, database name, user name, and password information. Then run the program.

Preparing your project

When you start Visual Basic, you have a number of different project templates you can choose from. These templates allow you to build many different types of Visual Basic programs. For this program, I'm going to take the Standard EXE project and add everything I need to build the program (see Figure 7-1).

The next step is to add the ADO Data Control to the project. This is done using the Components dialog box. To open this dialog box, choose Project ➪ Components from the main menu. Then select the Controls tab if it isn't already selected and scroll down the list of controls until you find the Microsoft ADO Data Control (see Figure 7-2). Click on the checkbox and press OK to add the control to your Toolbox.

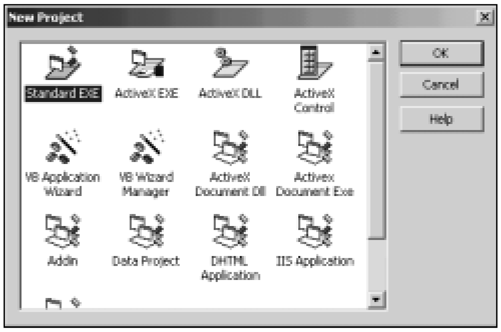
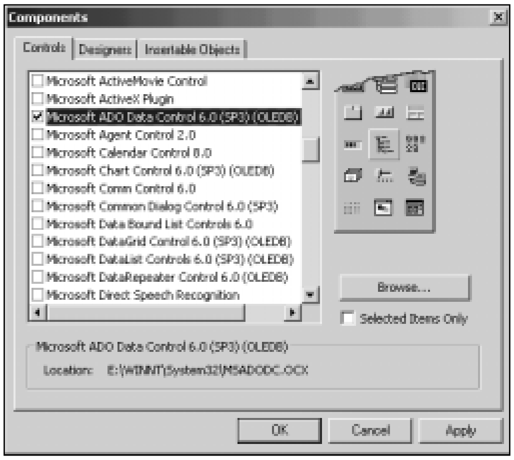


Figure 7-1: Selecting a Standard EXE project

Note

The service pack game: Figure 7-2 reflects the names of the controls that were updated using Visual Studio Service Pack 3. This is identified by (`SP3`) in the name of the control.


Components

Controls | Designers | Insertable Objects

- ☐ Microsoft ActiveMovie Control
- ☐ Microsoft ActiveX Plugin
- ☑ Microsoft ADO Data Control 6.0 (SP3) (OLEDB)
- ☐ Microsoft Agent Control 2.0
- ☐ Microsoft Calendar Control 8.0
- ☐ Microsoft Chart Control 6.0 (SP3) (OLEDB)
- ☐ Microsoft Comm Control 6.0
- ☐ Microsoft Common Dialog Control 6.0 (SP3)
- ☐ Microsoft Data Bound List Controls 6.0
- ☐ Microsoft DataGrid Control 6.0 (SP3) (OLEDB)
- ☐ Microsoft DataList Controls 6.0 (SP3) (OLEDB)
- ☐ Microsoft DataRepeater Control 6.0 (OLEDB)
- ☐ Microsoft Direct Speech Recognition

Microsoft ADO Data Control 6.0 (SP3) (OLEDB)
Location: E:\WINNT\System32\MSADODC.OCX

Figure 7-2: Adding the ADO Data Control to the project

Configuring the ADO Data Control

Once you add the ADO Data Control to your project, you need to place a copy of the data control on your form and set the properties so that it will access your database. You can either set the properties though the Visual Basic Properties Window, or you can right click on the control and select ADODC Properties from the popup menu. This will display a set of Property Pages for the control (see Figure 7-3).

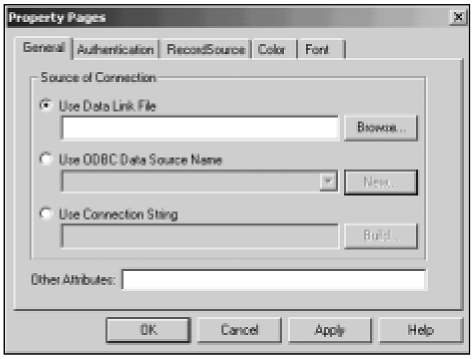


Figure 7-3: Setting the data control's properties

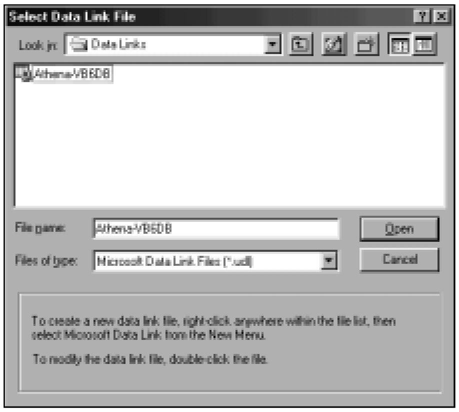
On the General tab, you'll see that there are three different ways to connect to a database server. You can use a Data Link File, an ODBC Data Source Name, or a Connection string. I'm going to show you how to build a Data Link File in this chapter and you can see how to create a Connection String in Chapter 9.

Note

OLE DB, not ODBC: The ADO Data Control is based on the OLE DB architecture, which is more efficient and offers a more flexible architecture. While you can create an ODBC Data Source, you would be much better off using a Data Link File or creating a Connection string.

Selecting a Data Link File

After selecting Use Data Link File as the Source of Connection, press the Browse button to either select an existing Data Link File or create a new one. The Select Data Link File dialog box (see Figure 7-4) will be displayed with the default directory for data link files (`\Program Files\Common Files\System\OLE DB\Data Links\`). If you have an existing Data Link File, simply select it and press Open. If not, you can create one by right clicking in the file area of the dialog box and choosing New ➪ Microsoft Data Link. A new file called New Microsoft Data Link will be created.



File name: Athena-VB6DB

Files of type: Microsoft Data Link Files (*.udl)

To create a new data link file, right-click anywhere within the file list, then select Microsoft Data Link from the New Menu.

To modify the data link file, double-click the file.

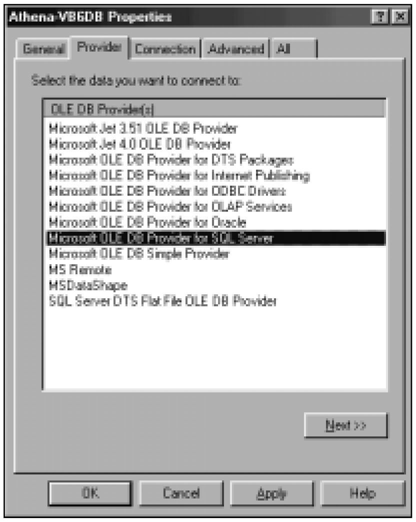
Figure 7-4: Choosing a Data Link File

Tip

Name of the game: I like to include the database server and the default database name in the name of my Data Link files. This makes it easy to identify the connection information.

Choosing an OLE DB provider

You can edit the properties in a Data Link File by right clicking on its icon and selecting Properties from the popup menu. This will display the Properties dialog box for the data link file. Select the Provider tab of the Properties dialog box to begin configuring your data link (see Figure 7-5).



Select the data you want to connect to:

OLE DB Provider(s)
- Microsoft Jet 3.51 OLE DB Provider
- Microsoft Jet 4.0 OLE DB Provider
- Microsoft OLE DB Provider for DTS Packages
- Microsoft OLE DB Provider for Internet Publishing
- Microsoft OLE DB Provider for ODBC Drivers
- Microsoft OLE DB Provider for OLAP Services
- Microsoft OLE DB Provider for Oracle
- Microsoft OLE DB Provider for SQL Server
- Microsoft OLE DB Simple Provider
- MS Remote
- MSDataShape
- SQL Server DTS Flat File OLE DB Provider

Figure 7-5: Viewing the properties of a Data Link File

Note

Define me before you use me: If you just created this file, you must edit the properties before it can be used.

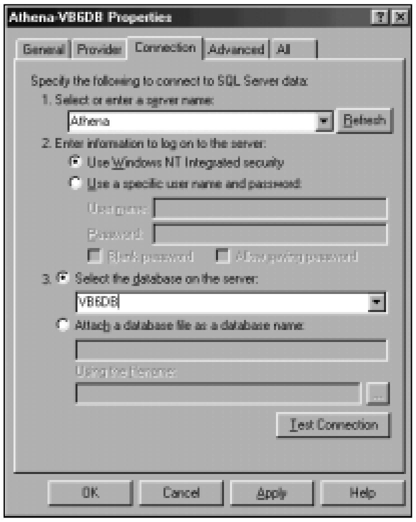
There are a number of choices for the data provider. For best performance, you should always choose the OLE DB provider for your database system. If you can't locate one for your specific database management system, then choose Microsoft OLE DB Provider for ODBC Drivers. Once you've chosen your provider, press the Next button.

Tip

It isn't there: Microsoft only supplies OLE DB providers for SQL Server and Oracle. If you are using a different database system, contact your database vendor to get their OLE DB provider.

Entering provider-specific information

Each OLE DB provider has a list of information it needs in order to connect to a database server. Figure 7-6 shows the information required for the Microsoft OLE DB Provider for SQL Server.


Athena-VB6DB Properties

General | Provider | Connection | Advanced | All

Specify the following to connect to SQL Server data:

1. Select or enter a server name: Athena
2. Enter information to log on to the server:
   - [✓ Use Windows NT Integrated security]
   - Use a specific user name and password:
     - User name:
     - Password:
     - Blank password
     - Allow saving password
3. [✓ Select the database on the server:] VB6DB
   - Attach a database file as a database name:
   - Using the filename:

Figure 7-6: Selecting the database server and default database

There are three basic pieces of information needed in this form: the name of the database server, login information, and the name of the default database. In this example, I'm using `Athena` as my database server, Windows NT Authentication for my login information and `VB6DB` as the default database.

Testing the connection

Once you've finished entering the properties, press the Test Connection button to verify that you can connect to the database server. If the information you specify is correct, you should see a message box saying, "Test connection succeeded". If there is a problem, you will see an error message describing the problem. You should then correct the information you provided and try it again.

After you are able to test the connection successfully, you should press the OK button to close the Properties dialog box, and then press the Open button on the Select Data Link File dialog box to choose the data link file. This will return you to the data control's Properties window.

Choosing a RecordSource

The last step of configuring the data control is to choose a source of data. This information is stored in the RecordSource property. You can edit this property on the RecordSource tab of the Properties window. In this case, since you want to make the Customers available via this program, you should choose a CommandType of adCmdTable and then select the Customers table from the drop-down list found immediately below that field (see Figure 7-7). This will automatically be entered into the RecordSource property. Then I can press OK to close the Property Pages dialog box and save these values in the data control.

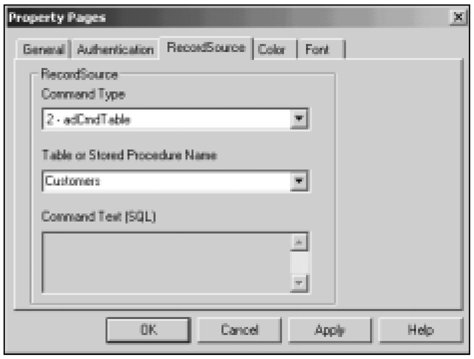

Property Pages
RecordSource
Command Type: 2 - adCmdTable
Table or Stored Procedure Name: Customers
Command Text (SQL)

Figure 7-7: Selecting the Customers table as the RecordSource

Adding bound controls

After adding and configuring the ADO Data Control, it's time to add some bound controls. I'm going to start by adding a text box for the Name field and binding it to the data control (see Figure 7-8).

Drag a text box onto your form, and size it to hold a person's full name. Then view the Properties window and scroll it so that the DataSource and DataField properties are both visible. Select the DataSource property. A drop-down arrow will appear at the end of the property's value field. Press it and select the data control from the drop-down list. It should be the only item on the list.

Next, select the DataField property and press the drop-down arrow. A list of all of the fields in the table will be listed. Select the Name field to bind the text box to the Name column in the Customers table.

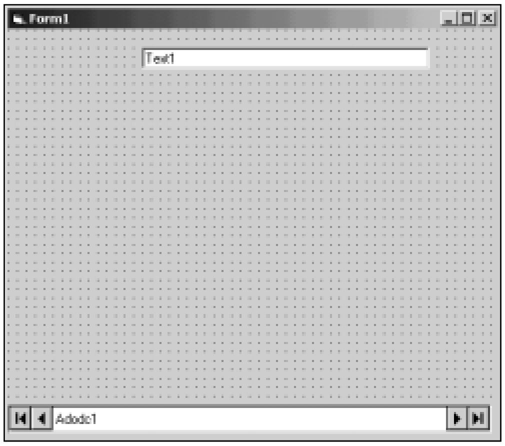


Figure 7-8: Adding the text box for the Name field to your form

Tip

The Database Connection: You may not have given this much thought at this point, but the Visual Basic development environment automatically uses the connection information you specified in the ADO Data Control to gather information about your database at development time. It then uses this information to help you avoid mistakes when you enter values into the various properties. Of course, you can enter all of the information for the bound controls manually and then configure the data control, but that isn't the lazy programmer's way of doing things.

Testing your program

Even though this program can only display the person's name from the Customers table, it is worth running it to verify that the program actually works. After choosing Run ➪ Start from the main menu, the program will load, establish a connection to the database, retrieve the first row from the Customers table, and display the value from the Name field in the text box (see Figure 7-9).

You can press the Next Row button on the data control to display the name from the second row in the table and you can return to the first row by pressing the Previous Row button. Pressing the Last Row button will take you to the last row of the table, while pressing the First Row button will return you to the first row in the table.

Finishing your program

Of course, a program that lists one field from a database table is a significant accomplishment in terms of all the little steps necessary to make it work, but it isn't a terribly useful program. So at this point, you need to add controls for the rest of the fields in the table. Also, you should take the time to place labels beside each field so that the user will understand the information displayed. When you've finished, your form might look like the one shown in Figure 7-10.

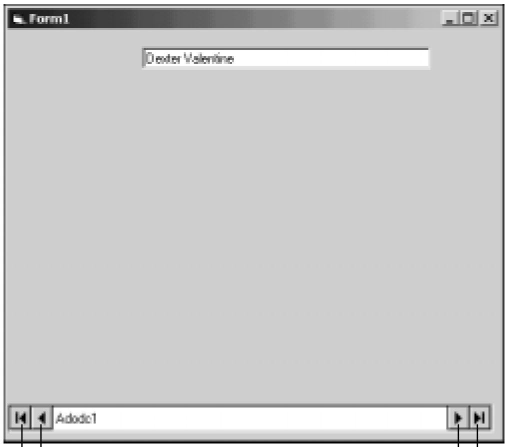
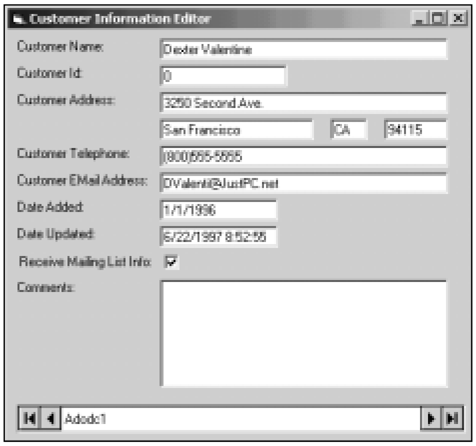


Figure 7-9: Running your codeless program for the first time



Customer Name:	Dexter Valentine
Customer Id:	0
Customer Address:	3250 Second Ave.
	San Francisco CA 94115
Customer Telephone:	(800)555-5555
Customer EMail Address:	DValenti@JustPC.net
Date Added:	1/1/1996
Date Updated:	6/22/1997 8:52:55
Receive Mailing List Info:	✓
Comments:	

Figure 7-10: Finishing your codeless program

Updating database information

To change a value in the database using this program, simply change the value in the field and move to another row. As you change the data displayed on the form, the bound controls pass the changes back to the data control, which will automatically update the row when it moves to another row. Of course, if there is an error with the update process, a message box will appear and you will remain on the current row until you correct the problem.

Using numeric fields

You may remember that the Zip field stores its information using a 32-bit integer rather than a character string. While you might think that some special code might be necessary to handle the data conversion from `Long` to `String` and back again, you would be wrong. The conversion is handled automatically by the text box control.

One limitation of using text boxes to display numeric information is that you can enter non-numeric information into the text box without immediately triggering an error. Of course, attempting to update the row will trigger a runtime error because you can't put a non-numeric value into a numeric field.

Using Boolean fields

`Boolean` fields are a natural fit for the `CheckBox` control. You set the `DataSource` and `DataField` properties just like the `TextBox` control. The control will display a `TRUE` value by placing a check mark in the check box and will clear the check box when the database value is `FALSE`.

Using Datetime fields

Like numeric values, date and time values also undergo a dynamic conversion process when information is displayed in the control. As long as the date and time information is properly formatted and legal, the control will automatically handle the conversion. If there is an error, the old Operation canceled message will be displayed, and the original values will be restored.

Adding new records

By default, the ADO Data Control doesn't allow you to add new records to the database. However, by changing one property value, you can easily include this capability in your codeless program. The `EOFAction` property determines what happens when the user attempts to move beyond the last row in the table.

By default, the data control will simply move the user back to the last row in the table if the user attempts to move beyond it. However, if you set `EOFAction` to `adDoAddNew` (2), the data control will automatically add a new blank row anytime the user attempts to move beyond the end of the file.

Once the user adds a record and enters the proper values into each of the fields, moving to another row will save the new row to the database. Of course, when you add a record using this technique, the final data must meet the database rules for acceptability. Any invalid values must be corrected before you can move to another row.

Thoughts on Codeless Programming

Okay, so a codeless program may not be perfect, but it will work and can often be useful in the early stages of building an application. If you're willing to add a little code to handle non-database functions, such as edit checking and menu management, Visual Basic offers a much better alternative to building a database application in Access.

Access is primarily an easy to use database forms generator with some scripting capabilities added. While it allows you to build simple database applications quickly, when building more complex applications you constantly run into situations that aren't easy to handle because of the many limitations in Access. However, Visual Basic is a true programming language and doesn't suffer from the same limitations.

The primary drawback to using a true programming language like Visual Basic is that it can be difficult to create programs that perform a complex task like accessing a database. However, by using bound controls and other tools in Visual Basic, you can easily create database programs that require very little code and that offer more power and flexibility than their Access counterparts.

Caution

I've added a record and I can't get out: One downside to using a codeless program to add records to your database is that there isn't a convenient method to abort the add process. In fact, the only way to abort the add process is to end the program, which is pretty drastic. Of course, this is a situation where a little code can go a long way towards making the program much easier to use.

Summary

In this chapter you learned the following:

- What data binding is and how it works.
- Which controls you can bind in your Visual Basic program.
- How you can use bound controls to build a meaningful program without any code.
- How to choose an OLE DB provider and build an ADO connection string to connect to your database.

✦ ✦ ✦