CHAPTER 5

Introducing Visual Basic

In This Chapter

Why use Visual Basic?

Visual Basic editions

Types of Visual Basic applications

In this chapter, I'm going to discuss why you should be using Visual Basic to develop your database applications. Then I'm going to quickly review the types of Visual Basic applications.

Why Use Visual Basic?

The easy answer to this question is that Visual Basic offers the best combination of features, efficiency, ease of use, and reliability of any programming language on the market today. However, the easy answer isn't always the obvious answer, so let's look at why you should be using Visual Basic for your database applications.

Basic history

Visual Basic is Microsoft's latest implementation of a language known as BASIC (Beginners All-purpose Symbolic Instruction Code) that was developed at Dartmouth College in 1964. It was designed to be an easy to use programming language for teaching students how to use a computer.

Jurassic BASIC

In the 1960's and 1970's, versions of BASIC were written for nearly every computer platform available, including everything from large-scale IBM mainframes and Cray supercomputers to the smallest mini-computers. It was extremely popular on the small mini-computers of that time, because both the computers and BASIC were interactive.

Most programming languages at that time were compiled in batch. In other words, you entered a bunch of statements into the computer and ran a compiler that checked the syntax and then ran the program if there were no syntax errors. Most often, you had to punch the statements onto punch cards using a special device known as a keypunch. Once you had created a deck of cards with your program on them, you had to submit the cards using a card reader to a faceless mainframe computer for executing. You then had to wait for the computer to run your batch job, at which time you got a printout that listed your errors. If you made a syntax error in the first line of the program, you got dozens of pages of errors, which had to be fixed before you could try again. If you were lucky, you could get two or three runs an hour, and if you were a lousy typist, you might spend several hours before you got far enough to actually run your program for the first time.

The key to BASIC's success was that BASIC would check each line of code when it was entered into the computer and report any errors immediately. This helped most programmers find their syntax errors and correct them before they tried to run the program. Then when the programmers were ready to run the program, they sat in front of a terminal and were able to immediately interact with the program. This helped immensely when testing a program, because you could test your program, find an error, and retest it in a matter of a minute or two, which allowed you to get meaningful results much quicker.

Of course this efficiency came with a cost. Most BASIC programs ran very slowly when compared to programs written in other programming languages. For the most part, this meant that the BASICs of that time were used mostly for teaching students how to program. After learning BASIC, most people would move onto real languages like COBOL or FORTRAN, which were more suited to building complex programs. This is where BASIC got a reputation for being a toy programming language that wasn't suitable for serious work.

BASIC jumps out of the Gates

In 1976, a small company known as MITS developed a computer from a handful of electronic parts, including a general-purpose calculator chip from a small, little-known company known as Intel. This computer was known as the Altair 8800. However, without a programming language, this computer was pretty useless. It did, however, provide an opportunity for a couple of Harvard students named Bill Gates and Paul Allen, who formed a company known as Microsoft. Their first product was a BASIC interpreter for the Altair that ran in 4,096 bytes of memory.

As other personal computers — such as the Apple II, the Commodore, the Atari, and a host of long-forgotten others — were designed, Microsoft was called upon to develop versions of BASIC for them as well. Before long, Microsoft was the standard of the industry, and every company that introduced a computer needed a Microsoft BASIC interpreter.

When IBM chose to enter the personal computer marketplace, they asked Microsoft to create a BASIC interpreter for their new machine. When discussions with another company for an operating system fell through, Microsoft was asked to create an operating system for the new computer also. This new operating system became known as PC-DOS, and later as MS-DOS.

By now Microsoft's BASIC had grown to the point where many people were using it to develop serious applications. The ability to read and write random files was a standard feature of the language, though the concept of database access was years away. BASIC had become a key product for the success of Microsoft and this generation of the personal computer industry.

Before BASIC was Visual

Over time, Microsoft realized that a compiler was necessary to improve performance of BASIC programs, so a new product known as QuickBasic was created. Another version called the Professional Development System was created for professional application developers and included a number of tools to make their life easier.

When Microsoft began building Windows, they decided to standardize on the C programming language. In order to build a serious program in the early versions of Windows, you really had to use C because that was the only practical way to access the operating system functions.

Microsoft didn't forget its roots, however, and created a special version of BASIC for Windows known as Visual Basic. It allowed programmers to include graphical interfaces for their BASIC applications. Version 1.0 of Visual Basic was the first and only version of Visual Basic to run on both DOS and Windows.

Data basics

Microsoft Access was developed to compete with similar products from Borland called Paradox, as well as products from Ashton-Tate called DBase. These products allowed people to build simple database applications quickly and easily. This was possible because of the functions provided by the databases that were at the heart of these tools.

Access came with a database known as Microsoft Jet. Unlike the databases used in the other tools, Jet was a true relational database. Since Jet was independent of Access, it was possible to use it in other programming languages. When Visual Basic 3 was released in 1993, it included an enhanced version of the Jet database, known as Jet 1.1. This tool became very popular with developers, and there are some people that are still using the original Visual Basic 3/Jet 1.1 combination to develop applications.

As time marched on, Microsoft released Visual Basic 4, which was the first version of Visual Basic to support 32-bit application development. It was also the last version of Visual Basic to support 16-bit application development. This dual nature was accomplished by two independent implementations of Visual Basic. However, the two versions were slightly incompatible with each other, because some of the controls weren't available in both versions. This usually caused a problem because Visual Basic 3 programmers couldn't move up to the 32-bit Visual Basic 4 compiler.

Another difference between the 16 and 32-bit version of Visual Basic was the Jet 2.5 and Jet 3.0 databases. Jet 2.5 was an improved version of the 16-bit Jet database included with Visual Basic 3, while the Jet 3.5 database was the same 32-bit database used by Access 95.

In Visual Basic 5, the 16-bit version was dropped, but numerous enhancements were made to the 32-bit version. The biggest enhancement was the ability to create your own ActiveX controls. Previously, if you wanted to create your own controls, you needed to write them in C++, which was time-consuming and complicated and often beyond the ability of the average Visual Basic programmer. However, with Visual Basic 5, anyone familiar with Visual Basic class objects could create their own controls.

The key to making ActiveX controls feasible was a native-mode compiler. Previous versions of Visual Basic compiled their programs into an intermediate code, which was executed at runtime by a special interpreter. The new native-mode compiler used components from the Visual C++ compiler to create the actual code. Now Visual Basic applications were almost as fast as Visual C++ applications.

BASIC today

Visual Basic 6 is the most recent version of Visual Basic. I like to think of Visual Basic 6 as Visual Basic 5 with every available factory option. While a few minor changes were made to the language itself, most of the changes added new features to the development environment. While Visual Basic 5 had only one type of application, Visual Basic 6 adds specialized applications that run on a Web server (IIS Application) and an application that runs on a Web browser (DHTML Application). Also, Visual Basic 6 includes a number of enhancements in the database area. These enhancements include the ability to drag and drop database definitions onto a form, a stored procedure debugger, and the ability to design databases directly in Visual Basic.

Database integration

The inclusion of the Jet database in Visual Basic doesn't necessarily mean that it's easy to use. However, the engineers at Microsoft worked hard to integrate database support into the Visual Basic language. This is accomplished through a few key features, such as a database object model, the ability to bind regular controls to the results of a database request, and specialized database controls.

Database object model

One of Visual Basic's strengths is the ability to create and use objects. Microsoft took advantage of this ability and created a set of objects that you can use to access the database. This isolates all of the low-level details from the average application and makes it easier to use the database.

Visual Basic currently supports three different object models:

- Data Access Objects (DAO)
- Remote Data Objects (RDO)
- ActiveX Data Objects (ADO)

DAO was originally released in Visual Basic 3 to support the Jet database and its design reflects that. While you could use DAO to access other databases, you had to define the other databases to Jet and access them through Jet. This extra overhead made a big impact on performance.

RDO was developed to eliminate the overhead and allow programmers to directly connect to SQL Server databases. The RDO object model was also much simpler than the DAO object model, making it easier to use. RDO was first introduced with Visual Basic 5.

Since neither DAO nor RDO offered a universal solution to database access, Microsoft developed the ADO object model with the intent to make it work everywhere. ADO was first shipped with Visual Basic 6, though you could download a copy from the Internet for use with Visual Basic 5. The ADO object model is similar to the RDO object model; however, some of the restrictions imposed by the RDO object model were removed, making it more flexible and easy to use.

Cross-Reference

Part III of this book is dedicated to the topic of ADO.

Bound controls

Most of the controls in Visual Basic can be tied to an open database, and automatically display the information in the current row of a table. The user can alter the contents of the bound control and the information will automatically be saved into the database. Using bound controls reduces the amount of code needed to display information from a database. And in general, the less code a programmer has to write, the more stable the program.

Database-oriented tools

Visual Basic 6 now includes a set of tools that helps the programmer perform database functions:

- **Data Environment Designer** helps you drag and drop database design information onto a form to quickly create database applications.
- **Data View Window** is available to view the contents of a database table or query at design time.
- **SQL Editor** is also available to help you create and debug stored procedures.
- **T-SQL Debugger** helps you debug stored procedures in SQL Server and Oracle8i systems.
- **Data Reporter** helps you build reports using information from the Data Environment Designer.

The Visual Database Tools integrated into Visual Basic, allowing you to create and modify database structures without leaving the Visual Basic development environment.

Database-oriented controls

Visual Basic also includes several controls that are designed primarily with database access in mind. The Data Control (which is available for both ADO and DAO) allows you to quickly build an application, which can scroll through the values in a database table or access the results of a database query.

Microsoft also took some common controls, such as the ComboBox and ListBox, and created database-specific versions of them with enhanced functions. These controls take their data directly from the database rather than requiring the programmer to load them explicitly. Microsoft also created a special control that allows a programmer to populate a spreadsheet grid with data from a database and allows the user to directly edit the values. These controls are available for both the ADO object libraries and the DAO object library.

A special control known as the DataRepeater control takes your custom ActiveX control, which displays information from a single row in a table, and repeats it as many times as necessary to display the results of a database query.

Visual Basic Editions

There are several different editions of Visual Basic that offer various features targeted at several audiences:

- Learning Edition
- Professional Edition

- Enterprise Edition
- Other flavors of Visual Basic

Tip

Fixes before the fact: Like most software today, Visual Basic isn't perfect. Since Visual Basic was first released, Microsoft has found and fixed several problems and put the fixes together in a service pack. The service pack can be downloaded from Microsoft's Web site at `www.Microsoft.com/Vbasic` by following the links to product updates and downloads. There, you will be able to download the service pack. While hardly any of these problems should affect you, it is a good idea to download and apply the service pack to prevent any known problems from happening.

Learning Edition

The Visual Basic Learning Edition is targeted at people who want to learn how to use Visual Basic. Only the most core features are found in this edition. While you can develop simple database applications using this edition, the lack of features will force you to upgrade to the Professional Edition rather quickly.

Professional Edition

The Professional Edition is the most common edition of Visual Basic. This edition includes such features at the Data Environment Designer, the Data View Window, which allows you to view the information found in a database table while designing your application, the special database controls described above. The Visual Database Tools and the SQL Editor described above are included only with the Enterprise Edition.

Tip

Tools, tools and more tools: In general, I prefer to use the tools found with the particular database system, rather than the Visual Database Tools. This is because the Database Tools are missing certain features that would eventually force you to learn the tools included with the database system anyway.

A full set of ActiveX tools are included in the Professional Edition that perform a wide range of functions useful for building your database, such as the TreeView and ListView controls, as well as tools that allow your programs to access resources over the Internet.

You can choose to build a number of different types of programs with the Professional Edition. You can create your own ActiveX controls, your own ActiveX objects, Web server applications and Web browser applications in addition to the normal executable programs. This edition also has a number of wizards that help you build applications more quickly, as well as a number of templates that you can easily modify to perform routine functions.

Enterprise Edition

The Enterprise Edition of Visual Basic includes all of the features found in the Professional Edition, but adds the Visual Database Tools and the SQL Editor to create stored procedures. A number of other facilities, such as the Visual Component Manager and Visual SourceSafe, are included to facilitate group-programming projects. An application Performance Explorer tool is also available to help you determine the efficiency of your application.

The other big difference between the Professional Edition and the Enterprise Edition of Visual Basic are the collection of tools that are included with the package. Personal versions of Internet Information Server, and Microsoft Transaction Server and SQL Server are included to help developers set up a test environment on their own computers.

Also, the Remote Data Objects are available only with the Enterprise Edition. This was true with Visual Basic 5 as well as Visual Basic 6, but is only important if you had older programs that used RDO. Otherwise, this really isn't that critical of a feature.

Other Variations

Microsoft is so strongly committed to Visual Basic that they find ways to incorporate it into many products, often without your knowledge. Besides the editions of Visual Basic that I've just introduced, you can find additional variations in many other places.

Visual Basic for Applications (VBA)

Most Office products include a product known as Visual Basic for Applications (VBA). This product is based on the same Visual Basic language that is used to develop application programs. The primary purpose of this variation is to help Office users build macros that make a particular application easier to use. The macros are really just calls to the object model exposed by the application itself. However, an application programmer can use this object model to create some rather complex programs that can be run from Word or Excel.

Visual Basic Script (VBScript)

Another variation of Visual Basic is known as Visual Basic Script or VBScript. VBScript is designed as a lightweight macro language. Many of the features found in regular Visual Basic and VBA are missing from this language. VBScript was originally implemented as part of Internet Explorer as an alternative to JavaScript; however, it can be incorporated into your own application programs as well.

Caution

VBScript and viruses: Recently, several different viruses that were written using VBScript have made the news. These viruses were included in e-mail messages as macros, which are automatically executed when someone opens the message. Just because VBScript is now associated with viruses doesn't mean that it's bad, but you should take appropriate precautions and treat all files that you receive from e-mail or otherwise as a potential virus source.

Windows Host Scripting (WHS)

The final variation of Visual Basic is really a variant of VBScript, known as Windows Host Scripting (WHS). WHS is used as a macro language for Windows itself. It allows you to build complex command line programs similar to those built by Unix programmers using csh or other Unix command interpreters. The best way to think of WHS is that it is really just a replacement for the old batch file procedures that people used to write for DOS.

Types of Visual Basic Programs

There are four main types of Visual Basic programs:

- ✥ Standard EXEs
- ✥ ActiveX Controls/DLLs/EXEs
- ✥ IIS Applications
- ✥ DHTML Applications

Many applications will use a combination of these program types in order to fully exploit the power of Visual Basic.

Standard EXEs

People have been building standard EXE applications for years. In this type of application, the programmer creates an EXE file that is run by the application's user, which in turn interacts with the user to perform a specific function.

When building database applications, I often refer to the standard EXE as the traditional client/server program, because it communicates with a database server using the client/server application model. This helps to distinguish this type of program from other Web-based application programs, such as IIS Applications and DHTML Applications and other multi-tier applications built using ActiveX components.

ActiveX DLLs/Controls/EXEs

ActiveX programs come in three flavors:

- ✥ ActiveX DLLs
- ✥ ActiveX Controls
- ✥ ActiveX EXEs

Each of these flavors represents a collection of Component Object Model (COM) objects that have been compiled and can be used by other programs.

ActiveX DLLs

ActiveX DLLs are simply a collection of COM objects that can be used by other programs. Since the objects are stored in a Dynamic Link Library (DLL), you need to create a program that can use them. They can't be used standalone. However, ActiveX DLLs are a great place to locate common elements that would be shared among multiple programs.

COM+ Applications are really just ActiveX DLL programs that have been designed to run under control of the Windows 2000 COM+ Transaction Server. The primary purpose of COM+ transactions is to create a middle tier of processing that is independent of both the client computer and the database server. While you can accomplish this by using an ActiveX EXE, the COM+ Transaction Server is far more scalable and reliable and can be extremely useful if you have a large application with a high volume of activity.

Note

MTS + 1 = COM+: The COM+ Transaction Server is a major update to the old Microsoft Transaction Server version 2.0. While many of the features are similar, COM+ handles more of the details, making it easier to use than MTS.

ActiveX Controls

ActiveX Controls are nearly identical to ActiveX DLLs, except an ActiveX Control provides a visual presence on a Visual Basic form. You design an ActiveX Control by drawing a series of controls onto a form-like surface and adding the code necessary to manage the control.

ActiveX Controls are useful mostly when you want to put a combination of other controls and access them as a single unit. When combined with the DataRepeater control, you can display multiple rows of information on a form using the same format.

ActiveX EXEs

An ActiveX EXE combines the features of a Standard EXE with the objects found in an ActiveX DLL. Many programs you are familiar with are really implemented as an ActiveX EXE, such as Microsoft Word and Excel.

One advantage of an ActiveX EXE is that you can place the program on one computer and access the objects managed by the program from another program running on a different computer on the network. This program can offload work from both your database server and your client computer by adding a middle or third tier of processing.

IIS Applications

In today's world, many people are moving away from traditional client/server database applications to Web-based applications. These applications are really just programs that run on the Web server that return a string of HTML tags in response to a request from the Web browser.

An *IIS Application* is a special type of Visual Basic program that doesn't have any graphical components. Instead, it receives requests that originate from a Web browser via the same object model used by Active Server Programs (ASP) created by Visual Interdev. Then, the IIS Application constructs a Web page and returns it back to the Web browser for further processing by the user.

Since the IIS Application imposes very few requirements on the Web browser, it doesn't matter if the user has Internet Explorer, Netscape Navigator, or nearly any other type of Web browser running on any type of computer. This is the only type of Visual Basic program that will run on a Macintosh or a Sun workstation. However, as its name implies, IIS Applications will only work with an IIS Web server, so you must have a Windows 2000/NT server to run your IIS Application.

DHTML Applications

A *DHTML Application* is a Web page that contains an embedded ActiveX DLL, which can communicate with a database server or access other resources over the Internet. Its primary advantage is that it doesn't involve the Web server once the Web page and DLL have been downloaded. However, DHTML Applications only run on Internet Explorer 4.01 or higher.

Thoughts on Visual Basic

Microsoft claims that over three million people program in some form of Visual Basic. While this number undoubtedly includes people who write macros in Word and Excel, the true number of real Visual Basic programmers still exceeds one million individuals. I'm proud to be one of them.

I've been creating programs for over 25 years. I've used traditional programming languages like Fortran, COBOL, and Pascal for years. I've used specialized languages such as LISP, Prolog, GASP and SAS. I've used more assembly and systems programming languages than I can remember. I even used C before there was a C++. I've written compilers for my own programming languages, which were used by no one but myself. Yet I prefer Visual Basic to all of them.

There are many reasons why I prefer Visual Basic, but the most important one is that I'm more productive in Visual Basic than any other language, especially when writing Windows programs. I can start with nothing and end up with a comprehensive program faster in Visual Basic than any other language.

I wouldn't use Visual Basic to build a compiler, nor would I use Visual Basic to write an operating system, but I would use Visual Basic to write nearly any other type of program. And when it comes to database programs, Visual Basic is a clear winner to me. It is easy to create both traditional client/server and Web-based database programs, and by using ActiveX DLLs and COM+ transactions, you can gain a great deal of flexibility in how you actually create these programs.

Summary

In this chapter you learned that:

- ✦ Microsoft's first commercial product was an implementation of a programming language called BASIC.
- ✦ Visual Basic comes in multiple editions, of which only the Professional Edition and the Enterprise Edition have complete database support.
- ✦ You can create standard EXEs, ActiveX Controls, ActiveX DLLs and ActiveX EXEs using Visual Basic.
- ✦ You can develop Web server-based Visual Basic applications using IIS Applications.
- ✦ You can develop browser-based Visual Basic applications using DHTML Applications.
- ✦ You can develop Visual Basic to build n-tier applications using COM+ transactions.

✦ ✦ ✦