Programming a Bulletproof User Interface

Now we're going to continue the theme of the last chapter by enhancing the way our data control operates with our bound controls and - through the recordset - with the underlying database. We will be building a more robust and efficient interface, and preparing to encapsulate all the code that we'll build so that we can reuse it in other applications. Our form will become a great vehicle for data entry and manipulation.

The Data Control - Right or Wrong?

The data control has several methods associated with it, and we'll use these to operate the data control from code.

Some database programmers stay away from the data control - they feel that it is too slow and takes up too many resources. And it's true that once you get beyond smaller scale desktop applications these reservations have some force - Jet and Access are not designed for larger-scale operations. However, there are good reasons for us to use the data control in our examples here:

- It's ideal for small-scale applications
- When you consider what we did with just a few lines of code in the last chapter, the data control is just too powerful to ignore
- In terms of functionality, there are only a few things that we can't do compared to other DAO or ADO controls when working with recordsets

Most importantly, we're using the data control and Access as learning tools; we're exploring VB programming techniques and concepts that will add understanding, power and flexibility to your programming - whichever data access technology you are using to mediate between your VB program and the underlying data.

So, instead of discarding the data control, we're going to move on and enhance it. Once we have made our user interface mean and lean, we are going to encapsulate its functionality in a class that will allow us to reuse all of our code - I think you'll find this a very useful approach.
Here's how we'll progress through this chapter:

- We'll look briefly at classes and reusability
- We'll examine the properties and methods of the data control – in detail
- We'll look closely at the properties and methods of the recordset
- We'll examine the state-machine concept, which 'monitors' what our application is doing and decides what the user should be allowed to do next
- We'll add code to our user interface to make the user and the controls work better together

Let's begin by thinking about what we'll do to enhance our user interface.

Building an Enhanced User Interface

As mentioned previously, if we want to add, delete or scroll through records quickly, or if we want to find records, validate entries and perform all of the tasks that a professional data-entry screen uses, we must program the data control. The best way to show you how to do this is to enhance the program we have been working on. How? We will put some buttons on the form and permit the user to update and navigate using these. Then we'll significantly enhance the default processes for navigation, and for adding, editing and deleting records. We'll add the successive enhancements bit by bit, explaining what each of them does as we go. At the end of the chapter we'll run all of the code that we've built into our application and see the results. However, before we go any further, let's take a close look at the important properties and methods of the data control. In this chapter, we will be programming the data control, so let's see its built-in functionality.

The Class System

Once we've got our user interface form up and running, if we wanted to add data access to our existing project or to a separate project, we could copy our original VB form so that we could use it as a base to build on. There would be minor differences between the two data access forms, but the bulk of the code would be the same. Of course, we want to eliminate any duplicate code wherever possible. One way to eliminate code redundancy is to encapsulate common code into a class module. Using a class module, we can hide all of the details of how the code works. We will demonstrate how to do that shortly - and when we complete the next exercise, we will have a solid example of code reuse to take forward with us when we start using classes in detail in chapter 7.

Back to the Data Control...

Remember that we initialized the data control by setting its DatabaseName and RecordSource properties. The data control then has enough information to return a recordset object, which in turn has its own methods and properties. We customize the data control object by referring to its properties using the syntax Datal.property and to the returned recordset object by using Datal.Recordset.property.