CHAPTER 3

Creating ADO Objects

While many of you know (or think you know) how to construct ADO or other COM objects, bear with me for a minute while I walk you through some basics. I’ve seen too many suspect examples in publications whose technical editors should know better, to skip over this. I talk below about setting the references, or writing the code on an ASP to do so, and about a couple of mistakes that many developers are still making.

NOTE The techniques discussed here apply to COM—all COM objects—not just ADO. However, they do not apply to ADO.NET or any Visual Basic .NET objects except those accessed through COM interop, as I will discuss in the .NET part of this book (beginning with Chapter 13).

Instantiating ADO Objects

Okay, I want to get down to the technical coding stuff, but before you can use any of the ADO objects, you have to get Visual Basic to instantiate the objects—but you knew that. If you are working in the Visual Basic IDE, this process is drop-dead easy. All you have to do is reference the ADO library—and this might have already been done for you if you start a “Data” project template or reference the ADO Data Control component.

NOTE When you install Visual Basic 6.0 SP5, the new default ADO version changes to 2.5 from 2.1. None of the previous SPs affected this setting—you had to manually set a newer version of ADO.

If you’re working in the Visual Basic IDE, you can select an appropriate ADO library by using the Project/References menu. Simply select Microsoft ActiveX Data Objects 2.x Library (where x is the version of ADO 2 that you want to use) in the dialog box shown in Figure 3-1. After the library is referenced, Visual Basic examines the typelibs and fills in your application’s symbol table. This means that as you code, Visual Basic fills in the objects, properties, methods, and (especially) enumerated arguments. This single feature in Visual Basic dramatically improves
your performance—your code-writing performance that is. It can verify that you have typed the object names correctly and that the objects will exist at run time.

![References - DataProject](image)

**Figure 3-1. The Visual Basic IDE Project/References dialog box.**

---

**TIP**  What about the "Microsoft ActiveX Data Objects Recordset 2.x Library"? Isn’t this supposed to bring in a smaller, lighter, more limited version of ADO? Well, at one time it did—but no longer. It’s now just a type lib that exposes a subset of the full ADODB functionality, but it’s really referencing the same MSADO15.DLL that’s used by all versions of ADO.

Okay, so how do you create these objects in code? All too many MSDN and documentation examples show the very common syntax:

```vbnet
Dim cn as New ADODB.Connection ' This is the wrong way...
```

---

1. Guilty: I also used the `Dim xx as New yy` syntax in early revisions of the *Hitchhiker’s Guide*. When I discovered how this syntax impacted performance, I corrected them. Sorry ‘bout that.