CHAPTER 2

Introduction to ADO

I STRUGGLED WITH THIS CHAPTER in the sense that I didn't want to bore you with material that you already understood. But the trainer in me said to lay some solid foundations before digging right into the coding techniques. So, this chapter has a few theoretical sections that describe ADO and where it fits in the bigger scheme of things. If you want to skip down to the techie stuff, go ahead. I start talking about cool coding stuff in the Chapter 3. However, if you need a few more words on ADO, stick around.

ADO and OLE DB

When I start talking about getting at data from ADO, you might want to know a little more about what's going on behind the scenes. Well, as you know (or should know) ADO is a Component Object Model (COM) interface to OLE DB providers, and OLE DB expects to be accessed by consumers such as ADO. So, formally, an OLE DB Consumer is any piece of system or application code that consumes an OLE DB interface. This includes OLE DB components themselves. Figure 2-1 illustrates how ADO interfaces with OLE DB.

A provider is any software component that exposes an OLE DB interface. OLE DB providers can be classified broadly into two classes: data providers and service components.

A data provider is any OLE DB provider that owns data and exposes its data in a tabular form as a rowset, which is defined later in this chapter. Examples of data providers include relational DBMSs, storage managers, spreadsheets, ISAMs, and e-mail data sources.

A service component is any OLE DB component that does not own its own data, but encapsulates some service by producing and consuming data through OLE DB interfaces. A service component is both a consumer and a provider. For example, a heterogeneous query processor is a service component—it has to draw data from one source, restructure it, and pass it up the food chain to the requesting component, the consumer. Suppose a consumer asks to join data from tables in two different data sources. In its role as a consumer, the query processor retrieves rows from rowsets created over each of the base tables. In its role as a provider, the query processor creates a rowset from these rows and returns it to the consumer.
A database management system (DBMS) is a type of data source whose job it is to return information in one form or another as an OLE DB data provider. In some implementations, a DBMS is segmented into functional pieces (components)—each handing a specific job. In theory, component DBMSs offer greater efficiency than traditional DBMSs because consumers generally require only a portion of the database management functionality offered, thereby reducing resource overhead. OLE DB enables simple tabular data providers to implement functionality native to their data store: at a minimum, they can use only the interfaces that expose data as tables. This opens opportunities for the development of query processor components, such as SQL or geographical query processors. These can consume tabular information from any provider that exposes its data through OLE DB. In addition, SQL DBMSs can expose their functionality in a more layered manner by using the OLE DB interfaces.

Okay, so much for the theory. What's really going on is this: ADO is a COM-based programming interface—like DAO or RDO. Instead of ODBC or Jet drivers that connect to the data sources, ADO uses OLE DB. It turns out that one of those OLE DB providers speaks ODBC, enabling you to connect to any ODBC data source (assuming you have the right ODBC driver). These drivers are called "data providers" in OLE DB speak.