Accessing Data Using ADO.NET

ADO.NET and its predecessor ADO have a long history of facilitating direct data access for a variety of applications built using Microsoft technologies. ADO.NET has evolved tremendously since it was first introduced almost a decade ago. The biggest change in ADO.NET, however, was the addition of Entity Framework. The ability to work with a conceptual model against the native providers has captured the imagination of developers who want to use the .NET Framework for application development. Interacting with the conceptual model provided the abstraction level that programmers always wanted in their applications. In addition, this also reduced the number of lines of code that programmers had to write to fetch data from different sources, which made the technology extremely popular.

In Chapter 1, you were briefly introduced to the latest additions to ADO.NET. In this chapter, we will dive more deeply into the features of ADO.NET, but we will refrain from discussing Entity Framework, since it is covered in detail in Chapter 7. Specifically, we will talk about the following:

- How to read and modify a database using core ADO.NET concepts.
- How DataSource Controls are used to abstract the process of querying a database and perform manipulations using DataSet and DataReader in a declarative fashion.
- The different data type mappings between .NET Framework types and the native database.
- How to design a secure data access layer using ADO.NET.

ADO.NET Architecture

Traditionally, ADO.NET served as a platform that let applications connect to data stores and perform data queries and manipulations. That hasn’t changed much. However, a significant investment has gone into making ADO.NET more intelligent and to perform better with a variety of options to allow data manipulation in a disconnected fashion. As DataSets were the only available option previously, with Entity Framework you now have quite a few choices. Figure 4-1 illustrates the different components of ADO.NET and how they relate. You read an overview of these components in Chapter 1.
Although there have been significant changes to ADO.NET since it was first released with .NET Framework 1.0, the database providers have seen subtle changes mostly involving performance and security. The book *Beginning ASP.NET 2.0 Databases: From Novice to Professional* by Damien Foggon (Apress, 2006) covers the database providers in great detail for .NET Framework 2.0. In this chapter, we will first start by exploring ADO.NET for reading and modifying a database and then we will discuss what’s new in the .NET Framework 4.5 release.

**Reading and Modifying a Database Using ADO.NET Data Providers**

The data providers in ADO.NET are designed for fast forward-only read-only access to the data, and they perform manipulations in the form of inserts, updates, and deletes. Generally, these are referred to as **CRUD** (Create, Read, Update, and Delete) operations. CRUD operations can be performed via inline queries or stored procedures.

---

**Note**  
For reasonably complex queries, it is a good idea to use stored procedures for CRUD operations for performance and security benefits. By using stored procedures, you can help prevent SQL injection attacks, and since they are precompiled after the first run, there would be a boost in performance on subsequent data access operations.

There are native providers based on the database drivers you are using to access data. However, they all follow a common architecture. Figure 4-2 illustrates this.