Chapter 2: Creating Connections

Understanding Connections

- Connections are responsible for handling the physical communication between a data store and a .NET application. Because the Connection object is part of a Data Provider, each Data Provider implements its own version.

- The two Data Providers supported by the .NET Framework implement the **OleDbConnection** in the System.Data.OleDB namespace and the **SqlConnection** in the System.Data.SqlClient namespace, respectively.

- **Note** It's important to understand that if you're using a Connection object implemented by another Data Provider, the details of the implementation may vary from those described here. The OleDbConnection, not surprisingly, uses OLE DB and can be used with any OLE DB provider, including Microsoft SQL Server. The SqlConnection goes directly to SQL Server without going through the OLE DB provider and so is more efficient.

- The Connection object provides two overloaded versions of its constructor, giving you the option of passing in the ConnectionString, as shown in below Table:

Table 2-1: Connection Constructors

Method	Description
New()	Creates a Connection with the ConnectionString property set to an empty string
New(ConnectionString)	Creates a Connection with the ConnectionString property specified

The ConnectionString is used by the Connection object to connect to the data source. We'll explore it in detail in the next section of this chapter.

- ***Using Connection Properties***

The significant properties of the OleDbConnection and SqlDbConnection objects are shown in Table 2-2 and Table 2-3, respectively.

Table 2-2: OleDbConnection Properties

Property	Meaning	Default
ConnectionString	The string used to connect to the data source when the ***Open*** method is executed	Empty
ConnectionTimeout	The maximum time the Connection object will Continue attempting to make the connection before throwing an exception	15 seconds
Database	The name of the database to be opened once a connection is opened	Empty
DataSource	The location and file containing the database	Empty
Provider	The name of the OLE DB Data Provider	Empty
ServerVersion	The version of the server, as provided by the OLE DB Data Provider	Empty
State	A Connection State value indicating the current state of the Connection	Closed

Table 2-2: SqlConnection Properties

Property	Meaning	Default
ConnectionString	The string used to connect to the data source when the ***Open*** method is executed	Empty
ConnectionTimeout	The maximum time the Connection object will Continue attempting to make the connection before throwing an exception	15 seconds
Database	The name of the database to be opened once a connection is opened	Empty
DataSource	The location and file containing the database	Empty
PacketSize	The size of network packets used to communicate with SQL Server	8192 bytes
ServerVersion	The version of SQL Server	Empty
State	A Connection State value indicating the current state of the Connection	Closed
WorkStationId	A string identifying the database client, or, if that is not specified, the name of the workstation	Empty

As you can see, the two versions of the Connection object expose a slightly different set of properties: The SqlDbConnection doesn't have a Provider property, and the OleDbConnection doesn't expose PacketSize or WorkStationID. To make matters worse, not all OLE DB Data Providers support all of the OleDbConnection properties, and if you're working with a custom Data Provider, all bets are off.

What this means in real terms is that we still can't quite write code that is completely data source-independent unless we're prepared to give up the optimization of specific Data Providers. However, as we'll see, the problem isn't as bad as it might at first seem, since the .NET Framework provides a number of ways to accommodate run-time configuration. Rather more tedious to deal with are the different names of the objects, but using an intermediate variable can minimize the impact, as we'll see later in this chapter.

- ***Connection Methods***

Table 2-4: Connection Methods

Method	Description
BeginTransaction	Begins a database transaction
ChangeDatabase	Changes the current database on an open Connection
Close	Closes the connection to the data source
CreateCommand	Creates and returns a Data Command associated with the Connection
Open	Establishes a connection to the data source

- ***Handling Connection Events***

Both the OLE DB and the SQL Server Connection objects provide two events: StateChange and InfoMessage

StateChange Events

Not surprisingly, the StateChange event fires whenever the state of the Connection object changes. The event passes a StateChangeEventArgs to its handler, which, in turn, has two properties: OriginalState and CurrentState. The possible values for OriginalState and CurrentState are shown in Table 2-5.

Table 2-5: Connection States

State	Description
Broken	The Connection is open, but not functional. It may be closed and reopened
Closed	The Connection is closed
Connecting	Connecting The Connection is in the process of connecting, but has not yet been opened
Executing	The Connection is executing a command
Fetching	The Connection is retrieving data
Open	The Connection is open

- **InfoMessage Events**

The InfoMessage event is triggered when the data source returns warnings. The information passed to the event handler depends on the Data Provider.