**Introduction**

ADO.NET is an object-oriented set of libraries that allows you to interact with data sources.  Commonly, the data source is a database, but it could also be a text file, an Excel spreadsheet, or an XML file.  For the purposes of this tutorial, we will look at ADO.NET as a way to interact with a data base.

As you are probably aware, there are many different types of databases available.  For example, there is Microsoft SQL Server, Microsoft Access, Oracle, Borland Interbase, and IBM DB2, just to name a few.  To further refine the scope of this tutorial, all of the examples will use SQL Server.

  Therefore, everything you learn in this tutorial and all code will work with SQL Server.  The examples will use the Northwind database.  This is a tutorial is specifically for ADO.NET

**Data Providers**

We know that ADO.NET allows us to interact with different types of data sources and different types of databases.  However, there isn't a single set of classes that allow you to accomplish this universally.  Since different data sources expose different protocols, we need a way to communicate with the right data source using the right protocol.  Some older data sources use the ODBC protocol, many newer data sources use the OleDb protocol, and there are more data sources every day that allow you to communicate with them directly through .NET ADO.NET class libraries.

ADO.NET provides a relatively common way to interact with data sources, but comes in different sets of libraries for each way you can talk to a data source.  These libraries are called Data Providers and are usually named for the protocol or data source type they allow you to interact with.  table 1 lists some well known data providers, the API prefix they use, and the type of data source they allow you to interact with.

**table 1.  ADO.NET Data Providers are class libraries that allow a common way to interact with specific data sources or protocols.  The library APIs have prefixes that indicate which provider they support.**

|  |  |  |
| --- | --- | --- |
| **Provider Name** | **API prefix** | **Data Source Description** |
| ODBC Data Provider | Odbc | Data Sources with an ODBC interface.  Normally older data bases. |
| OleDb Data Provider | OleDb | Data Sources that expose an OleDb interface, i.e. Access or Excel. |
| Oracle Data Provider | Oracle | For Oracle Databases. |
| SQL Data Provider | Sql | For interacting with Microsoft SQL Server. |
| Borland Data Provider | Bdp | Generic access to many databases such as Interbase, SQL Server, IBM DB2, and Oracle. |

**ADO.NET Objects**

ADO.NET includes many objects you can use to work with data.  This section introduces some of the primary objects you will use.  Over the course of this tutorial, you'll be exposed to many more ADO.NET objects from the perspective of how they are used in a particular lesson.  The objects below are the ones you must know.  Learning about them will give you an idea of the types of things you can do with data when using ADO.NET.

**The SqlConnection Object**

To interact with a database, you must have a connection to it.  The connection helps identify the database server, the database name, user name, password, and other parameters that are required for connecting to the data base.  A connection object is used by command objects so they will know which database to execute the command on.

**The SqlCommand Object**

The process of interacting with a database means that you must specify the actions you want to occur.  This is done with a command object.  You use a command object to send SQL statements to the database.  A command object uses a connection object to figure out which database to communicate with.  You can use a command object alone, to execute a command directly, or assign a reference to a command object to an SqlDataAdapter, which holds a set of commands that work on a group of data as described below.

**The SqlDataReader Object**

Many data operations require that you only get a stream of data for reading.  The data reader object allows you to obtain the results of a SELECT statement from a command object.  For performance reasons, the data returned from a data reader is a fast forward-only stream of data.  This means that you can only pull the data from the stream in a sequential manner.  This is good for speed, but if you need to manipulate data, then a DataSet is a better object to work with.

**The DataSet Object**

DataSet objects are in-memory representations of data.  They contain multiple Datatable objects, which contain columns and rows, just like normal database tables.  You can even define relations between tables to create parent-child relationships.  The DataSet is specifically designed to help manage data in memory and to support disconnected operations on data

**The SqlDataAdapter Object**

Sometimes the data you work with is primarily read-only and you rarely need to make changes to the underlying data source.  Some situations also call for caching data in memory to minimize the number of database calls for data that does not change.  The data adapter makes it easy for you to accomplish these things by helping to manage data in a disconnected mode.  The data adapter fills a DataSet object when reading the data and writes in a single batch when persisting changes back to the database.  A data adapter contains a reference to the connection object and opens and closes the connection automatically when reading from or writing to the database.  Additionally, the data adapter contains command object references for SELECT, INSERT, UPDATE, and DELETE operations on the data.  You will have a data adapter defined for each table in a DataSet and it will take care of all communication with the database for you.  All you need to do is tell the data adapter when to load from or write to the database.