Introduction to ADO.NET

What is ADO.NET?

ADO stands for Microsoft ActiveX Data Objects. ADO.NET is one of Microsoft's Data Access technologies that we use to communicate with different data sources. It is a part of the .Net Framework and is used to establish a connection between a .NET Application and the data source.

ADO.NET is the bridge between your application and the Database where you persist your data.

Data sources can be Azure Databases, SQL Server, Oracle, MySQL, etc. ADO.NET consists of a set of classes that can be used to connect to a Db then perform CRUD operations on it. ADO.NET mainly uses System.Data.dll and System.Xml.dll.

What types of Applications use ADO.NET?

ADO.NET can be used to develop any type of .NET application. The following are some of the .NET applications where you can use ADO.NET to interact with a data source.

- ASP.NET Web Form Applications
- Windows Applications
- ASP.NET MVC Application
- Console Applications
- ASP.NET Web API Applications

What is .NET Data Providers?

The Database cannot directly execute C# code. It only understands SQL. So, if a .NET application needs to perform CRUD operations from or to a database, then the .NET application needs to:

1. Connect to the Database
2. Prepare an SQL Command
3. Execute the Command
4. Retrieve the results
5. Map the results to C# Class instances
6. Display the results in the application

And this is possible with the help of .NET Data Providers.

A **.NET Framework data provider** is used for connecting to a database, executing commands, and retrieving results. Those results are either processed directly, placed in a DataSet to be exposed to the user as needed, combined with data from multiple sources, or passed between application tiers (layers). .NET Framework data providers are lightweight. They create a minimal

layer between the data source and the code. This increases performance without sacrificing functionality.

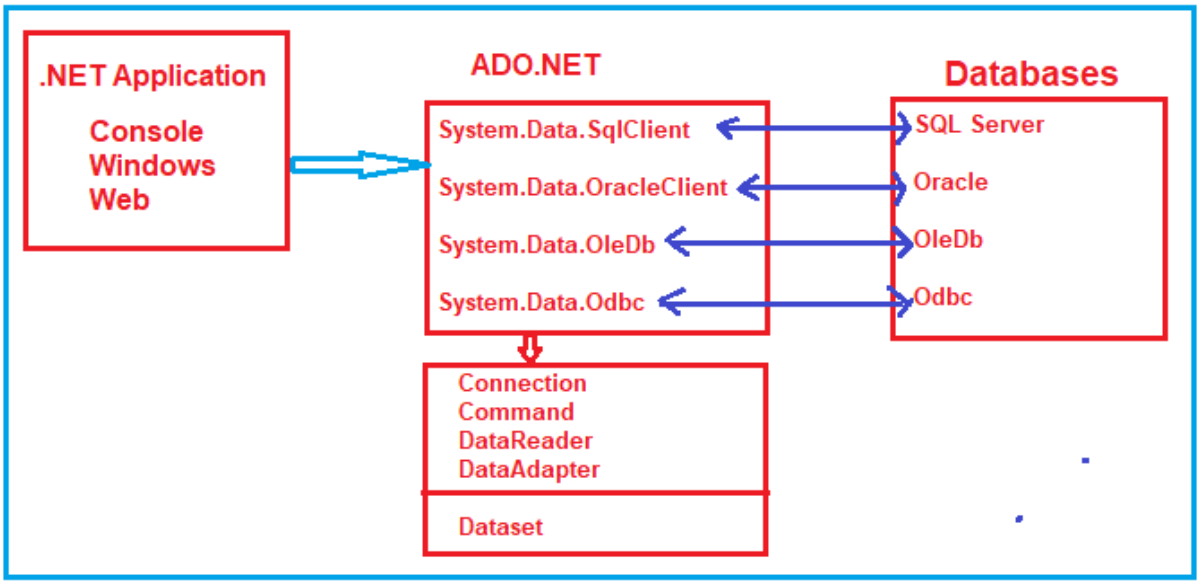
ADO.NET Data Providers

Different .NET Data Providers have been created for different Databases

SQL Server Data Provider: System.Data.SqlClient
Oracle Data Provider: System.Data.OracleClient
OLEDB Data Provider: System.Data.OleDb
ODBC Data Provider: System.Data.Odbc

Based on the data source being used, a different Data Provider must be used in your application to connect with it. The left, middle, and right sections of the below graph show:

- Left - The .NET Application types,
- Middle - The .NET Data Providers available,
- Right - The different data sources that ADO.NET can communicate with.



Components of ADO.NET

The four Components of ADO.NET are designed for data manipulation and fast data access.

- Connection:
    - Establishes a connection to the specified data source. The base class for all Connection objects is the DbConnection class.
- Command
    - Executes a command against the data source. Exposes Parameters and can execute in the scope of a Transaction from a Connection. The base class for all Command objects is the DbCommand class.
- DataReader
    - Reads a forward-only, read-only stream of data from a data source. The base class for all DataReader objects is the DbDataReader class.
- DataAdapter
    - Populates a DataSet and resolves updates with the data source. The base class for all DataAdapter objects is the DbDataAdapter class.

Depending on the provider, the ADO.NET components (Connection, Command, DataReader, and DataAdapter) have a different prefix.

	SQL Server	Oracle	OLE DB	ODBC
Connection	SQLConnection	OracleConnection	OleDbConnection	OdbcConnection
Command	SQLCommand	OracleCommand	OleDbCommand	OdbcCommand
DataReader	SQLDataReader	OracleDataReader	OleDbDataReader	OdbcDataReader
DataAdapter	SQLDataAdapter	OracleDataAdapter	OleDbDataAdapter	OdbcDataAdapter

The data provider we will use is System.Data.SqlClient. **System.Data.SqlClient** is recommended for middle-tier applications that use Microsoft SQL Server. It's also recommended for single-tier applications that use Microsoft Database Engine (MSDE) or SQL Server.

ADO.NET code to connect to SQL Server Database

The following image shows sample ADO.NET code that connects to SQL Server Database and retrieves data. We use System.Data.SqlClient classes SQLConnection, SQLCommand, and SQLDataReader.

SQL Server and ADO.NET:
https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/sql/

These classes are called 'Provider classes' and are used to interact with the database and perform CRUD operations.

```
SqlConnection connection = new SqlConnection("data source=.; database=TestDB; integrated security=SSPI");
SqlCommand command = new SqlCommand("Select * from Customers", connection);
connection.Open();
SqlDataReader myReader = command.ExecuteReader();

while (myReader.Read())
{
    Console.WriteLine("\t{0}\t{1}", myReader.GetInt32(0), myReader.GetString(1));
}

connection.Close();
```

**Notice all the classes used above are prefixed with the word SQL. This means these classes all interact with a SQL Server database.

.ExecuteReader(); - Sends the CommandText to the Connection and builds a SqlDataReader.

.ExecuteNonQuery() - Executes a SQL statement against the connection and returns the number of rows affected.

.AddWithValue(paramstring, value); - Adds a value to the end of the SqlParameterCollection.

```
// These are valid ways of adding values to the parameters in the connection string.
command.Parameters.Add("@ID", SqlDbType.Int);
command.Parameters["@ID"].Value = customerID;

// Use AddWithValue() to assign a value to the demographics parameter.
// SQL Server will implicitly convert strings into XML.
command.Parameters.AddWithValue("@demographics", demoXml);
```

DataSet:

The DataSet object in ADO.NET is not provider specific. Once you connect to a database and execute the command, a SqlDataReader is returned. Use the .Read() method of the SqlDataReader Class Object to retrieve the row data returned from the query. The data can then be stored in a DataSet and interacted with independent from the database. The DataSet contains a collection of one or more DataTable objects that you queried.

SQL Injection
https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ef/security-considerations
https://www.codeproject.com/Articles/732429/ADO-NET-How-to-Prevent-SQL-Injection-Attack-2

Sql Connection Example.

```
SqlConnection conn = new
SqlConnection("Server=tcp:p1rebuild.database.windows.net,1433;Initial
Catalog=071822_batch_Db;Persist Security Info=False;User
ID=p1rebuild;Password=Have1pie;MultipleActiveResultSets=False;Encrypt=True;TrustServerCer
tificate=False;Connection Timeout=30;");
```