Comparison between ADO and ADO.NET

ADO	ADO.Net
ADO is base on **COM** : **Component Object Modelling** based.	ADO.Net is based on **CLR** : **Common Language Runtime** based.
ADO stores data in **binary** format.	ADO.Net stores data in **XML** format i.e. parsing of data.
ADO can't be integrated with XML because ADO have limited access of XML.	ADO.Net can be integrated with XML as having robust support of XML.
In ADO, data is provided by **RecordSet**.	In ADO.Net data is provided by **DataSet** or **DataAdapter**.
ADO is **connection** oriented means it requires continuous active connection.	ADO.Net is **disconnected**, does not need continuous connection.
ADO gives rows as single table view, it scans sequentially the rows using **MoveNext** method.	ADO.Net gives rows as collections so you can access any record and also can go through a table via loop.
In ADO, You can create only **Client** side cursor.	In ADO.Net, You can create both **Client & Server** side cursor.
Using a single connection instance, ADO can not handle multiple transactions.	Using a single connection instance, ADO.Net can handle multiple transactions.

benefits offered by ADO.NET

Interoperability

As XML becomes more widely accepted, the benefits of the XML-based persistence and transmission of ADO.NET become more obvious. ADO.NET is designed to take advantage of the acceptance of XML. Because the work being done out-of-sight, out-of-mind is in XML, any component that can parse XML can use ADO.NET. Programming languages and lack of compatibility no longer stand in the way. With ADO.NET, a PERL component can pass an ADO.NET DataSet to a C# component, which can manipulate it and pass it to an ASP.NET Web Form written in Visual Basic.NET. Throughout the whole process, the data is transmitted as an XML file and materialized as a DataSet without need for concern.

Even if the component being used cannot parse XML, that component can still play a part. Since the DataSet is being transmitted as an XML file, it can be saved to the file system or retransmitted to another component without doing any damage to the file.

Maintainability

The Web and Internet exchange have come a long way since they became widely accepted by the non-programmer community. Every day more and more people get on the Internet, and every day businesses around the world need to find ways to maintain their Web site infrastructure to handle the increased traffic. Commonly, they do this by dividing the Web site structure into layers to even out the processing demand across multiple servers. Perhaps the presentation processing, business logic, and data services are all moved to separate servers, as in typical three- tier architecture. Maybe the business logic layer is subdivided to multiple servers.

Using ADO.NET makes this type of division a little easier. The layers of the solution must be able to communicate. With ADO.NET, they can communicate through XML-based DataSet s, making the communication a little less painful.

Performance

Web applications are inherently disconnected; the client does not maintain an active connection to the server, and the server does not maintain an active connection to the data source.
ADO.NET DataSet s offer performance advantages over traditional ADO disconnected RecordSet s. In the past, when data was passed from one component to another, the application's performance took a heavy hit due to the cost of COM marshalling. The values in the RecordSet had to be converted to data types recognized by COM. As a result, there was a significant processing cost. In ADO.NET, this data type conversion isn't required because you're simply passing an XML file.

Scalability

In any Web solution, scalability is a huge concern. Everyone wants Web traffic to increase, and as it does the solution needs to accommodate the higher levels of throughput. Often, applications will perform well for a small number of users, instilling a false sense of confidence. As the user base grows, application performance can suffer because the users are fighting over a limited number of resources, such as database connections, records locked by other users, etc.

ADO.NET inherently encourages you to work in a disconnected manner, retrieving the data that is required and releasing the resources immediately. Because you can pull your data into a disconnected DataSet in the form of an XML file, scalability is increased without the need for additional hardware. Database locks and active connections are released, giving additional users access to the resources.

ADO.NET Namespaces

The following are the important namespaces related with ADO.NET

System.Data.
This namespace is the core of ADO.NET framework. DataSet is the most import class of this namespace. It also contains classes to represent tables, columns, rows, relation and the constraint.

System.Data.Common
As its name suggests, it provides common classes, those works as base class. These classes are used by all data providers. Examples are DbConnection and DbDataAdapter

System.Data.OleDb
It provides classes that work with OLE-DB data sources using the .NET OleDb data provider. If you are

using the Microsoft Access as a database, then System.Data.OleDb namespace will be used.

Example of these classes are as follows:

- OleDbConnection
- OleDbCommand

System.Data.Odbc
This namespace defines classes that work with the ODBC data sources using the .NET ODBC data provider. It is collection of classes used to access an ODBC data source. It contains classes such as

- OdbcConnection
- OdbcCommand
- OdbcDataReader

System.Data.SqlClient
It defines the data provider for the SQL Server database. It contains classes that are used to access a SQL Server database. Examples are:

- SqlConnection
- SqlCommand
- SqlDataReader

System.Data.SqlTypes
This namespace provides classes for specific data types for the SQL Server database.

Data providers

Data provider is used to connect to the database, execute commands and retrieve the record. It is lightweight component with better performance. It also allows us to place the data into DataSet to use it further in our application.

The .NET Framework provides the following data providers that we can use in our application.

.NET Framework data provider	Description
.NET Framework Data Provider for SQL Server	It provides data access for Microsoft SQL Server. It requires the **System.Data.SqlClient** namespace.
.NET Framework Data Provider for OLE DB	It is used to connect with OLE DB. It requires the **System.Data.OleDb** namespace.

.NET Framework Data Provider for ODBC	It is used to connect to data sources by using ODBC. It requires the **System.Data.Odbc** namespace.
.NET Framework Data Provider for Oracle	It is used for Oracle data sources. It uses the **System.Data.OracleClient** namespace.
EntityClient Provider	It provides data access for Entity Data Model applications. It requires the **System.Data.EntityClient** namespace.
.NET Framework Data Provider for SQL Server Compact 4.0.	It provides data access for Microsoft SQL Server Compact 4.0. It requires the **System.Data.SqlServerCe** namespace.

.NET Framework Data Providers Objects

Following are the core object of Data Providers.

Object	Description
Connection	It is used to establish a connection to a specific data source.
Command	It is used to execute queries to perform database operations.
DataReader	It is used to read data from data source. The DbDataReader is a base class for all DataReader objects.
DataAdapter	It populates a DataSet and resolves updates with the data source. The base class for all DataAdapter objects is the DbDataAdapter class.

NET Framework Data Provider for SQL Server

Data provider for SQL Server is a lightweight component. It provides better performance because it directly access SQL Server without any middle connectivity layer. In early versions, it interacts with ODBC layer before connecting to the SQL Server that created performance issues.

The .NET Framework Data Provider for SQL Server classes is located in the **System.Data.SqlClient** namespace. We can include this namespace in our C# application by using the following syntax.

1. using System.Data.SqlClient;

This namespace contains the following important classes.

Class	Description
SqlConnection	It is used to create SQL Server connection. This class cannot be inherited.
SqlCommand	It is used to execute database queries. This class cannot be inherited.
SqlDataAdapter	It represents a set of data commands and a database connection that are used to fill the DataSet. This class cannot be inherited.
SqlDataReader	It is used to read rows from a SQL Server database. This class cannot be inherited.
SqlException	This class is used to throw SQL exceptions. It throws an exception when an error is occurred. This class cannot be inherited.