**What is ADO.NET?**

**ADO.NET** is a set of computer software components that can be used by programmers to access data and data services. It is a part of the [base class library](http://library) that is included with the [Microsoft .NET Framework](http://framework). It is commonly used by programmers to access and modify data stored in [relational database systems](http://dbms), though it can also be used to access data in non-relational sources. ADO.NET is sometimes considered an evolution of [ActiveX Data Objects](http://objects) (ADO) technology, but was changed so extensively that it can be considered an entirely new product.

**What are the benefits of ADO.NET?**

1>Scalability:-

ADO.NET works on DataSet that can represent a whole database or even a data table as a disconnected object and thereby eliminates the problem of the constraints of number of databases being connected. In this way scalability is achieved.

2>Data Source Independence:-

In ADO.NET DataSet is completely independent of data source and no way DataSet is controlled by the data source as it happens in case of RecordSet.

3>Interoperability:-

As ADO.NET transmits the data using the format of XML which is not dependent on ADO.NET or windows platform.

4>Strongly Typed Fields:-

It supports strongly typed fields by using CTS.

5>Performance:-

The performance in ADO.NET is higher in comparison to ADO that uses COM marshalling.

6>Firewall:

As in ADO.NET transmission is via XML format, therefore it can pass through firewalls.

**What are the differences between ADO and ADO.NET?**

ADO relied on a connection based model. In the connected approach, the client had to be connected with the server and remain connected till the whole procedure or transaction was completed. Time resources and bandwidth became major constraints on such architecture.

To solve this problem the latter version of ADO used RecordSet. All the contents from the data source were copied into RecordSet. This allows clients to get disconnected from the server, work on the RecordSet and copy the changes back to the data source again. This approach did not succeed much because it requires COM marshalling to transmit disconnected data, it support only those datatypes that were defined by the COM standards and hence required type conversion.

ADO.NET can be used to access data sources using new .NET data providers as well as existing OLEDB data providers using the OLEDB.NET data provider.

**Discuss the architecture of ADO.NET.**

There are two components of ADO.NET that you can use to access and manipulate data:

* .NET Framework data providers
* The [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset(v=VS.80).aspx)

### *.NET Framework Data Providers*

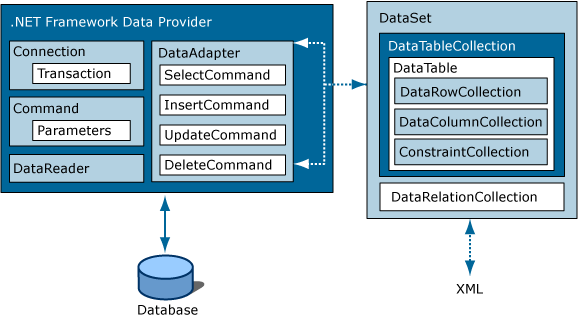
The .NET Framework Data Providers are components that have been explicitly designed for data manipulation and fast, forward-only, read-only access to data. The **Connection** object provides connectivity to a data source. The **Command** object enables access to database commands to return data, modify data, run stored procedures, and send or retrieve parameter information. The **DataReader** provides a high-performance stream of data from the data source. Finally, the [DataAdapter](http://msdn.microsoft.com/en-us/library/system.data.common.dataadapter(v=VS.80).aspx) provides the bridge between the **DataSet** object and the data source. The **DataAdapter** uses **Command** objects to execute SQL commands at the data source to both load the **DataSet** with data, and reconcile changes made to the data in the **DataSet** back to the data source.

### *The DataSet*

The [ADO.NET DataSet](http://msdn.microsoft.com/en-us/library/zb0sdh0b(v=VS.80).aspx) is explicitly designed for data access *independent of any data source*. As a result, it can be used with multiple and differing data sources, used with XML data, or used to manage data local to the application. The **DataSet** contains a collection of one or more [DataTable](http://msdn.microsoft.com/en-us/library/system.data.datatable(v=VS.80).aspx) objects made up of rows and columns of data, as well as primary key, foreign key, constraint, and relation information about the data in the **DataTable** objects.

The following diagram illustrates the relationship between a .NET Framework data provider and a **DataSet**.

*ADO.NET architecture*

**

**What are the components of .NET Data Provider?**

The following table outlines the four core objects that make up a .NET Framework data provider.

| Object | Description |
| --- | --- |
| **Connection** | Establishes a connection to a specific data source. |
| **Command** | Executes a command against a data source. Exposes **Parameters** and can execute within the scope of a **Transaction** from a **Connection**. |
| **DataReader** | Reads a forward-only, read-only stream of data from a data source. |
| **DataAdapter** | Populates a **DataSet** and resolves updates with the data source. |

**What is .NET Framework** **Data Provider? Why is it used?**

The System.Data.SqlClient namespace is the.NET Framework Data Provider for SQL Server.

The.NET Framework Data Provider for SQL Server describes a collection of classes used to access a SQL Server database in the managed space. Using the [SqlDataAdapter](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldataadapter(v=VS.100).aspx), you can fill a memory-resident [Dataset](http://msdn.microsoft.com/en-us/library/system.data.dataset(v=VS.100).aspx) that you can use to query and update the database.

**Classes**

|  | Class | Description |
| --- | --- | --- |
|  | [SqlBulkCopy](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlbulkcopy(v=VS.100).aspx) | Lets you efficiently bulk load a SQL Server table with data from another source. |
|  | [SqlBulkCopyColumnMapping](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlbulkcopycolumnmapping(v=VS.100).aspx) | Defines the mapping between a column in a [SqlBulkCopy](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlbulkcopy(v=VS.100).aspx) instance's data source and a column in the instance's destination table. |
|  | [SqlBulkCopyColumnMappingCollection](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlbulkcopycolumnmappingcollection(v=VS.100).aspx) | Collection of [SqlBulkCopyColumnMapping](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlbulkcopycolumnmapping(v=VS.100).aspx) objects that inherits from [CollectionBase](http://msdn.microsoft.com/en-us/library/system.collections.collectionbase(v=VS.100).aspx). |
|  | [SqlClientFactory](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlclientfactory(v=VS.100).aspx) | Represents a set of methods for creating instances of the System.Data.SqlClient provider's implementation of the data source classes. |
|  | [SqlClientMetaDataCollectionNames](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlclientmetadatacollectionnames(v=VS.100).aspx) | Provides a list of constants for use with the **GetSchema** method to retrieve metadata collections. |
|  | [SqlClientPermission](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlclientpermission(v=VS.100).aspx) | Enables the .NET Framework Data Provider for SQL Server to help make sure that a user has a security level sufficient to access a data source. |
|  | [SqlClientPermissionAttribute](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlclientpermissionattribute(v=VS.100).aspx) | Associates a security action with a custom security attribute. |
|  | [SqlCommand](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand(v=VS.100).aspx) | Represents a Transact-SQL statement or stored procedure to execute against a SQL Server database. This class cannot be inherited. |
|  | [SqlCommandBuilder](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommandbuilder(v=VS.100).aspx) | Automatically generates single-table commands that are used to reconcile changes made to a [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset(v=VS.100).aspx) with the associated SQL Server database. This class cannot be inherited. |
|  | [SqlConnection](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnection(v=VS.100).aspx) | Represents an open connection to a SQL Server database. This class cannot be inherited. |
|  | [SqlConnectionStringBuilder](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnectionstringbuilder(v=VS.100).aspx) | Provides a simple way to create and manage the contents of connection strings used by the [SqlConnection](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnection(v=VS.100).aspx) class. |
|  | [SqlDataAdapter](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldataadapter(v=VS.100).aspx) | Represents a set of data commands and a database connection that are used to fill the [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset(v=VS.100).aspx) and update a SQL Server database. This class cannot be inherited. |
|  | [SqlDataReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldatareader(v=VS.100).aspx) | Provides a way of reading a forward-only stream of rows from a SQL Server database. This class cannot be inherited. |
|  | [SQLDebugging](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldebugging(v=VS.100).aspx) | Infrastructure. Included to support debugging applications. Not intended for direct use. |
|  | [SqlDependency](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldependency(v=VS.100).aspx) | The [SqlDependency](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldependency(v=VS.100).aspx) object represents a query notification dependency between an application and an instance of SQL Server 2005. An application can create a [SqlDependency](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldependency(v=VS.100).aspx) object and register to receive notifications via the [OnChangeEventHandler](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.onchangeeventhandler(v=VS.100).aspx) event handler. |
|  | [SqlError](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlerror(v=VS.100).aspx) | Collects information relevant to a warning or error returned by SQL Server. |
|  | [SqlErrorCollection](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlerrorcollection(v=VS.100).aspx) | Collects all errors generated by the .NET Framework Data Provider for SQL Server. This class cannot be inherited. |
|  | [SqlException](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlexception(v=VS.100).aspx) | The exception that is thrown when SQL Server returns a warning or error. This class cannot be inherited. |
|  | [SqlInfoMessageEventArgs](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlinfomessageeventargs(v=VS.100).aspx) | Provides data for the [InfoMessage](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnection.infomessage(v=VS.100).aspx) event. |
|  | [SqlNotificationEventArgs](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlnotificationeventargs(v=VS.100).aspx) | Represents the set of arguments passed to the notification event handler. |
|  | [SqlParameter](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlparameter(v=VS.100).aspx) | Represents a parameter to a [SqlCommand](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand(v=VS.100).aspx) and optionally its mapping to [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset(v=VS.100).aspx) columns. This class cannot be inherited. |
|  | [SqlParameterCollection](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlparametercollection(v=VS.100).aspx) | Represents a collection of parameters associated with a [SqlCommand](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand(v=VS.100).aspx) and their respective mappings to columns in a [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset(v=VS.100).aspx). This class cannot be inherited. |
|  | [SqlRowsCopiedEventArgs](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlrowscopiedeventargs(v=VS.100).aspx) | Represents the set of arguments passed to the [SqlRowsCopiedEventHandler](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlrowscopiedeventhandler(v=VS.100).aspx). |
|  | [SqlRowUpdatedEventArgs](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlrowupdatedeventargs(v=VS.100).aspx) | Provides data for the [RowUpdated](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldataadapter.rowupdated(v=VS.100).aspx) event. |
|  | [SqlRowUpdatingEventArgs](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlrowupdatingeventargs(v=VS.100).aspx) | Provides data for the [RowUpdating](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldataadapter.rowupdating(v=VS.100).aspx) event. |
|  | [SqlTransaction](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqltransaction(v=VS.100).aspx) | Represents a Transact-SQL transaction to be made in a SQL Server database. This class cannot be inherited. |

**Delegates**

|  | Delegate | Description |
| --- | --- | --- |
|  | [OnChangeEventHandler](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.onchangeeventhandler(v=VS.100).aspx) | Handles the [OnChange](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldependency.onchange(v=VS.100).aspx) event that is fired when a notification is received for any of the commands associated with a [SqlDependency](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldependency(v=VS.100).aspx) object. |
|  | [SqlInfoMessageEventHandler](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlinfomessageeventhandler(v=VS.100).aspx) | Represents the method that will handle the [InfoMessage](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnection.infomessage(v=VS.100).aspx) event of a [SqlConnection](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnection(v=VS.100).aspx). |
|  | [SqlRowsCopiedEventHandler](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlrowscopiedeventhandler(v=VS.100).aspx) | Represents the method that handles the [SqlRowsCopied](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlbulkcopy.sqlrowscopied(v=VS.100).aspx) event of a [SqlBulkCopy](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlbulkcopy(v=VS.100).aspx). |
|  | [SqlRowUpdatedEventHandler](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlrowupdatedeventhandler(v=VS.100).aspx) | Represents the method that will handle the [RowUpdated](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldataadapter.rowupdated(v=VS.100).aspx) event of a [SqlDataAdapter](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldataadapter(v=VS.100).aspx). |
|  | [SqlRowUpdatingEventHandler](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlrowupdatingeventhandler(v=VS.100).aspx) | Represents the method that will handle the [RowUpdating](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldataadapter.rowupdating(v=VS.100).aspx) event of a [SqlDataAdapter](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldataadapter(v=VS.100).aspx). |

**Enumerations**

|  | Enumeration | Description |
| --- | --- | --- |
|  | [SortOrder](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sortorder(v=VS.100).aspx) | Specifies how rows of data are sorted. |
|  | [SqlBulkCopyOptions](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlbulkcopyoptions(v=VS.100).aspx) | Bitwise flag that specifies one or more options to use with an instance of [SqlBulkCopy](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlbulkcopy(v=VS.100).aspx). |
|  | [SqlNotificationInfo](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlnotificationinfo(v=VS.100).aspx) | This enumeration provides additional information about the different notifications that can be received by the dependency event handler. |
|  | [SqlNotificationSource](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlnotificationsource(v=VS.100).aspx) | Indicates the source of the notification received by the dependency event handler. |
|  | [SqlNotificationType](http://msdn.microsoft.com/en-us/library/z0fkxc6y(v=VS.100).aspx) | Describes the different notification types that can be received by an [OnChangeEventHandler](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.onchangeeventhandler(v=VS.100).aspx) event handler through the [SqlNotificationEventArgs](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlnotificationeventargs(v=VS.100).aspx) parameter. |

**What is a connection object? How to connect to a database?**

A SqlConnection object represents a unique session to a SQL Server data source. With a client/server database system, it is equivalent to a network connection to the server. SqlConnection is used together with [SqlDataAdapter](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldataadapter.aspx) and [SqlCommand](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.aspx) to increase performance when connecting to a Microsoft SQL Server database. For all third-party SQL server products, and other OLE DB-supported data sources, use [OleDbConnection](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbconnection.aspx).

When you create an instance of SqlConnection, all properties are set to their initial values. For a list of these values, see the SqlConnection constructor.

If the SqlConnection goes out of scope, it won't be closed. Therefore, you must explicitly close the connection by calling Close or Dispose. Close and Dispose are functionally equivalent. If the connection pooling value Pooling is set to true or yes, the underlying connection is returned back to the connection pool. On the other hand, if Pooling is set to false or no, the underlying connection to the server is actually closed.

**What is connection pooling? Why the concept of connection pooling is used? Or what are the benefits of using connection pooling?**

Connecting to a database server typically consists of several time-consuming steps. A physical channel such as a socket or a named pipe must be established, the initial handshake with the server must occur, the connection string information must be parsed, the connection must be authenticated by the server, checks must be run for enlisting in the current transaction, and so on.

In practice, most applications use only one or a few different configurations for connections. This means that during application execution, many identical connections will be repeatedly opened and closed. To minimize the cost of opening connections, ADO.NET uses an *optimization technique called connection pooling.*

*Benefits*

* *Connection pooling reduces the number of times that new connections must be opened.* The pooler maintains ownership of the physical connection. It manages connections by keeping alive a set of active connections for each given connection configuration. Whenever a user calls Open on a connection, the pooler looks for an available connection in the pool. If a pooled connection is available, it returns it to the caller instead of opening a new connection. When the application calls Close on the connection, the pooler returns it to the pooled set of active connections instead of closing it. Once the connection is returned to the pool, it is ready to be reused on the next Open call.

Only connections with the same configuration can be pooled. ADO.NET keeps several pools at the same time, one for each configuration. Connections are separated into pools by connection string and by Windows identity when integrated security is used. Connections are also pooled based on whether they are enlisted in a transaction.

* *Pooling connections can significantly enhance the performance and scalability of your application.* By default, connection pooling is enabled in ADO.NET. Unless you explicitly disable it, the pooler optimizes the connections as they are opened and closed in your application. You can also supply several connection string modifiers to control connection pooling behavior.

**How a connection pool is created?**When a connection is first opened, a connection pool is created based on an exact matching algorithm that associates the pool with the connection string in the connection. *Each connection pool is associated with a distinct connection string.* When a new connection is opened, if the connection string is not an exact match to an existing pool, a new pool is created. *Connections are pooled per process, per application domain, per connection string and when integrated security is used, per Windows identity.* Connection strings must also be an exact match; keywords supplied in a different order for the same connection will be pooled separately.

In the following C# example, three new [*SqlConnection*](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnection.aspx)objects are created, but only two connection pools are required to manage them. Note that the first and second connection strings differ by the value assigned for Initial Catalog.

using (SqlConnection connection = new SqlConnection(

"Integrated Security=SSPI;Initial Catalog=Northwind"))

{

connection.Open();

// Pool A is created.

}

using (SqlConnection connection = new SqlConnection(

"Integrated Security=SSPI;Initial Catalog=pubs"))

{

connection.Open();

// Pool B is created because the connection strings differ.

}

using (SqlConnection connection = new SqlConnection(

"Integrated Security=SSPI;Initial Catalog=Northwind"))

{

connection.Open();

// The connection string matches pool A.

}

**How to add a connection to connection pool?**

A connection pool is created for each unique connection string. When a pool is created, multiple connection objects are created and added to the pool so that the minimum pool size requirement is satisfied. Connections are added to the pool as needed, up to the maximum pool size specified (100 is the default). Connections are released back into the pool when they are closed or disposed.

When a [*SqlConnection*](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnection.aspx) object is requested, it is obtained from the pool if a usable connection is available. To be usable, a connection must be unused, have a matching transaction context or be unassociated with any transaction context, and have a valid link to the server.

The connection pooler satisfies requests for connections by reallocating connections as they are released back into the pool. If the maximum pool size has been reached and no usable connection is available, the request is queued. The pooler then tries to reclaim any connections until the time-out is reached (the default is 15 seconds). If the pooler cannot satisfy the request before the connection times out, an exception is thrown.

We strongly recommend that you always close the connection when you are finished using it so that the connection will be returned to the pool. You can do this using either the Close or Dispose methods of the Connection object, or by opening all connections inside a using statement in C#, or a Using statement.

**How to remove a connection from connection pool?**

The connection pooler removes a connection from the pool after it has been idle for a long time, or if the pooler detects that the connection with the server has been severed. Note that a severed connection can be detected only after attempting to communicate with the server. If a connection is found that is no longer connected to the server, it is marked as invalid. Invalid connections are removed from the connection pool only when they are closed or reclaimed.

If a connection exists to a server that has disappeared, this connection can be drawn from the pool even if the connection pooler has not detected the severed connection and marked it as invalid. This is the case because the overhead of checking that the connection is still valid would eliminate the benefits of having a pooler by causing another round trip to the server to occur. When this occurs, the first attempt to use the connection will detect that the connection has been severed, and an exception is thrown.

**What is the purpose of connection pooling in ADO.NET?**

Connecting to a data source can be time consuming. To minimize the cost of opening connections, ADO.NET uses an optimization technique called connection pooling, which minimizes the cost of repeatedly opening and closing connections. Connection pooling is handled differently for the .NET Framework data providers.

**What is a command object in ADO.NET? How to use it?**

SqlCommand Class represents a Transact-SQL statement or stored procedure to execute against a SQL Server database. This class cannot be inherited.

SqlCommand features the following methods for executing commands at a SQL Server database:

| Item | Description |
| --- | --- |
| [BeginExecuteNonQuery](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.beginexecutenonquery.aspx) | Initiates the asynchronous execution of the Transact-SQL statement or stored procedure that is described by this SqlCommand, generally executing commands such as INSERT, DELETE, UPDATE, and SET statements. Each call to [BeginExecuteNonQuery](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.beginexecutenonquery.aspx) must be paired with a call to [EndExecuteNonQuery](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.endexecutenonquery.aspx) which finishes the operation, typically on a separate thread. |
| [BeginExecuteReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.beginexecutereader.aspx) | Initiates the asynchronous execution of the Transact-SQL statement or stored procedure that is described by this SqlCommand and retrieves one or more results sets from the server. Each call to [BeginExecuteReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.beginexecutereader.aspx) must be paired with a call to [EndExecuteReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.endexecutereader.aspx) which finishes the operation, typically on a separate thread. |
| [BeginExecuteXmlReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.beginexecutexmlreader.aspx) | Initiates the asynchronous execution of the Transact-SQL statement or stored procedure that is described by this SqlCommand. Each call to BeginExecuteXmlReader must be paired with a call to EndExecuteXmlReader, which finishes the operation, typically on a separate thread, and returns an [XmlReader](http://msdn.microsoft.com/en-us/library/system.xml.xmlreader.aspx) object. |
| [ExecuteReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.executereader.aspx) | Executes commands that return rows. For increased performance, [ExecuteReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.executereader.aspx) invokes commands using the Transact-SQL sp\_executesql system stored procedure. Therefore, [ExecuteReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.executereader.aspx) might not have the effect that you want if used to execute commands such as Transact-SQL SET statements. |
| [ExecuteNonQuery](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.executenonquery.aspx) | Executes commands such as Transact-SQL INSERT, DELETE, UPDATE, and SET statements. |
| [ExecuteScalar](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.executescalar.aspx) | Retrieves a single value (for example, an aggregate value) from a database. |
| [ExecuteXmlReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.executexmlreader.aspx) | Sends the [CommandText](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.commandtext.aspx) to the [Connection](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.connection.aspx) and builds an [XmlReader](http://msdn.microsoft.com/en-us/library/system.xml.xmlreader.aspx) object. |

You can reset the [CommandText](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.commandtext.aspx) property and reuse the SqlCommand object. However, you must close the [SqlDataReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldatareader.aspx) before you can execute a new or previous command.

**Give an example showing how to use ExecuteReader method.**The following example creates a [SqlConnection](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnection.aspx), a SqlCommand, and a [SqlDataReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldatareader.aspx). The example reads through the data, writing it to the console. Finally, the example closes the [SqlDataReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldatareader.aspx) and then the [SqlConnection](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnection.aspx) as it exits the Using code blocks.

private static void ReadOrderData(string connectionString)

{

string queryString =

"SELECT OrderID, CustomerID FROM dbo.Orders;";

using (SqlConnection connection = new SqlConnection(

connectionString))

{

SqlCommand command = new SqlCommand(

queryString, connection);

connection.Open();

SqlDataReader reader = command.ExecuteReader();

try

{

while (reader.Read())

{

Console.WriteLine(String.Format("{0}, {1}",

reader[0], reader[1]));

}

}

finally

{

// Always call Close when done reading.

reader.Close();

}

}

}

**How to run/execute a stored procedure from .NET code?**

**What is a DataSet?**

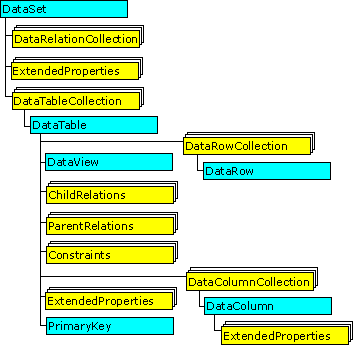
The Dataset is the central object in ADO.NET. DataSet is a logical container of data. It represents a set of data tables referenced as one unit in the application.

With this object you can get all the data you need from each table quickly, examine and change it while you are disconnected from the server and then update the server with changes in an efficient operation.

The DataSet object is central to supporting disconnected, distributed data scenarios with ADO.NET. The DataSet is a memory-resident representation of data that provides a consistent relational programming model regardless of the data source. It can be used with multiple and differing data sources, used with XML data, or used to manage data local to the application. The DataSet represents a complete set of data including related tables, constraints, and relationships among the tables.

**Discuss the DataSet object model.**

The following illustration shows the **DataSet** object model.



The methods and objects in a **DataSet** are consistent with those in the relational database model.

The **DataSet** can also persist and reload its contents as XML and its schema as XML Schema definition language (XSD) schema.

*The DataTableCollection*

An ADO.NET **DataSet** contains a collection of zero or more tables represented by **DataTable** objects. The **DataTableCollection** contains all the **DataTable** objects in a **DataSet**.

A **DataTable** is defined in the **System.Data** namespace and represents a single table of memory-resident data. It contains a collection of columns represented by a **DataColumnCollection**, and constraints represented by a **ConstraintCollection**, which together define the schema of the table. A **DataTable** also contains a collection of rows represented by the **DataRowCollection**, which contains the data in the table. Along with its current state, a **DataRow** retains both its current and original versions to identify changes to the values stored in the row.

*The DataRelationCollection*

A **DataSet** contains relationships in its **DataRelationCollection** object. A relationship, represented by the **DataRelation** object, associates rows in one **DataTable** with rows in another **DataTable**. It is analogous to a join path that might exist between primary and foreign key columns in a relational database. A **DataRelation** identifies matching columns in two tables of a **DataSet**.

Relationships enable navigation from one table to another within a **DataSet**. The essential elements of a **DataRelation** are the name of the relationship, the name of the tables being related, and the related columns in each table. Relationships can be built with more than one column per table by specifying an array of **DataColumn** objects as the key columns. When a relationship is added to the **DataRelationCollection**, it may optionally add a **UniqueKeyConstraint** and a **ForeignKeyConstraint** to enforce integrity constraints when changes are made to related column values.

*ExtendedProperties*

The **DataSet** (as well as the **DataTable** and **DataColumn**) has an **ExtendedProperties** property. **ExtendedProperties** is a **PropertyCollection** where you can place customized information, such as the SELECT statement that was used to generate the resultset, or a date/time stamp of when the data was generated. The **ExtendedProperties** collection is persisted with the schema information for the **DataSet** (as well as the **DataTable** and **DataColumn**).

**How to fill DataSet with data?**

To fill DataSet with data we have to use Fill() method of DataAdapter object.

Fill() has several overloads. But the simple one is

Fill(DataSet, DataTable)

The first parameter will take the name of the dataset to be filled and the second parameter specifies the name of the DataTable in the DataSet which will contain the data.

**What is the use of DataSet?**

DataSet acts as a virtual table. The value is retrieved from the database

and stores in the DataSet. Then we can pass the data to the DataGrid through DataSet.

**How can you update the records in database using datareader?**

Well, You cannot update. DataReader is just used for reading the data in forward only mode. You can achieve this using Dataset but not by DataReader.

**What is Dataview in ADO Dot Net ?**

The DataView provides different views of the data stored in a DataTable. That is we can customize the views of data from a DataTable. DataView can be used to sort, filter, and search the data in a DataTable , additionally we can add new rows and modify the content in a DataTable.

We can create DataView in two ways. Either we can use the DataView constructor, or we can create a reference to the DefaultView property of the DataTable.

DataView dView = new DataView(dTbl);

dView = dataSet.Tables(0).DefaultView;

**Stored procedure return more than one resut set and datareader used to fetch record, how you fetch second result set using datareader?**

Use DataReader.NextResult()

True - It Check more result remain to read and reader will forward to next result.

False - No more result set found

**Name the classes that are contained in System.Data NameSpace?**

DataSet

DataTable

DataColumn

DataRow

DataRealation

Constraint

**Name the classes are found in System.Data.Common NameSpace?**

1)DataColumnMapping

2)DataTableMapping

**Which is the default Provider Name of the Providers used to access the DataBase?**

**Select from following answers:**

1. **System.Data.SqlClient**
2. System.Data.Odbc
3. System.Data.Oledb
4. System.Data.OracleClient

**Which ADO.NET Object's fill method is used to fill the data from database into either DataSet or DataTable?**

**DataAdapter**

**To perform asynchronous data access, what must be added to the connection string?**

You must set Asynchronous=true.

**You are working with a DataSet and want to be able to display data, sorted different ways. How do you do so?**

**Use a DataView object for each sort.**

**Which is the base class for TypedDataset?**

**Dataset**

**To improve the performance and scalability of your .NET application. Which one of the following techniques would help?.**

**Select from following answers:**

1. Connection Strings
2. **Connection Pooling**
3. SQL Injection
4. Data Adapter

**Which method is used to Get a value indicating whether the column contains non-existent or missing values?**

**GetName ()**

**Which method is used to commit all changes in the DataSet or DataTable?**

**Select from following answers:**

1. Update()
2. **AcceptChanges()**
3. GetChanges()
4. None

Calling the AcceptChanges method or AcceptChanges method will commit all changes in the DataSet or DataTable. If either of these methods are called before the Update method is called, no changes will be committed when the Update method is called, unless further changes have been made since AcceptChanges was called.

**Which method is used to create a new row in a Table?**

**Select from following answers:**

1. **NewRow()**
2. Add()
3. Read()
4. ExecuteReader()

After you create a DataTable and define its structure using columns and constraints, you can add new rows of data to the table. To add a new row, declare a new variable as type DataRow. A new DataRow object is returned when you call the NewRow method. The DataTable then creates the DataRow object based on the structure of the table, as defined by the DataColumnCollection.

**Best Method to retrieve two values from Database (SQL Server)**

**Select from following answers:**

1. ExecuteDataSet()
2. ExecuteScalar()
3. ExecuteReader()
4. **ExecuteNonQuery()**

**How to add auto increment column in the DataTable?**

To add auto increment column in the DataTable, we can set the AutoIncrement property of the DataColumn object as true and specify the seed value after adding that field into the DataTable object.

// create columns for the DataTable

DataTable dTable = new DataTable();

DataColumn auto = new DataColumn("AutoID", typeof(System.Int32));

dTable.Columns.Add(auto);

// specify it as auto increment field

auto.AutoIncrement = true;

auto.AutoIncrementSeed = 1;

auto.ReadOnly = true;

**How to create a column in the DataTable?**

To create a column in the DataTable we can use the Columns.Add method of the DataTable object and pass DataColumn object as parameter.

DataTable dTable = new DataTable();

// create another column

DataColumn name = new DataColumn("Name", typeof(string));

dTable.Columns.Add(name);

**How to add a new row in DataTable?**

To add a new row in DataTable, we can use NewRow() method of the DataTable object.(Here assume that there are two columns Name, Address in the DataTable.

DataTable dTable = new DataTable();

DataRow row = null;

for (int i = 0; i < 5; i++)

{

row = dTable.NewRow ();

row["Name"] = i + " - Raja";

row["Address"] = "USA";

dTable.Rows.Add(row);

}

**Difference between OLEDB Provider and SqlClient ?**

SQLClient .NET classes are highly optimized for the .net / sqlserver combination and achieve optimal results. The SqlClient data provider is fast. It's faster than the Oracle provider, and faster than accessing database via the OleDb layer. It's faster because it accesses the native library (which automatically gives you better performance), and it was written with lots of help from the SQL Server team.

**What are the advantage of ADO.Net?**

• ADO.NET Does Not Depend On Continuously Live Connections

• Database Interactions Are Performed Using Data Commands

• Data Can Be Cached in Datasets

• Datasets Are Independent of Data Sources

• Data Is Persisted as XML

• Schemas Define Data Structures

**Explain about Data access objects or DAO?**

DAO is used for database access on windows platform. It creates a work space object in which applications or operations are performed. There are two types of database engines they are Jet database engine and ODBC direct database engine.

**Explain about the relationship of XML and ADO.NET?**

ADO.NET utilizes the power of XML by providing disconnected access to data. This is designed with the help of XML classes in .NET Framework which form the components of single architecture.

**Name some of the top level objects which ADO consists?**

1. Connection object is responsible for creating a connection to the database.

2. Record object represents data which is not from the database.

3. Parameter object represents a sql parameter

4. Stream object is responsible to represent data from a text page or web page.

**How to create a DataView from DataTable?**

In order to create a DataView from a DataTable, use instantiate the DataView object by passing DataTable as parameter in the constructor.

eg.

DataView dView = new DataView(dTable);

Can DataAdapter object accept DataTable as parameter in Fill method?

Posted by: Raja | Show/Hide AnswerYes,

DataAdapter object can accept either DataTable or DataSet as parameter to fill data from database.

eg.

SqlDataAdapter dAd = new SqlDataAdapter();

DataTable dTable = new DataTable();

DataSet dSet = new DataSet();

----

---

dAd.Fill(dTable); // will also work

dAd.Fill(dSet); // will also work

We should only use DataSet as parameter when we are expecting more than one result set is being returned from database.

**What is a Linked Server?**

A linked server configuration enables SQL Server to execute commands against OLE DB data sources on remote servers. Linked Servers is a concept in SQL Server by which we can add other SQL Server to a Group and query both the SQL Server dbs using T-SQL Statements.

Linked servers offer the following advantages:

1. Remote server access.

2. The ability to issue distributed queries, updates, commands, and transactions on heterogeneous data sources across the enterprise.

3. The ability to address diverse data sources similarly.

With a linked server, you can create very clean, easy to follow, SQL statements that allow remote data to be retrieved, joined and combined with local data. Stored Procedure sp\_addlinkedserver, sp\_addlinkedsrvlogin will be used add new Linked Server.

**If you have more than one lakh rows in database table,while printing that table in frontend performence will be degraded. for that what do you do to improve the performence?**

Increase the execution time out in web.config (or)increase the connection time out in connection string

**Diff Data Grid and Repeater**

Datagrid is

\* one which has advanced features and lets you do lot many things like paging and sorting your data without much effort.

\* DataGrid can hold text data, but not linked or embedded objects.

Whereas a DataRepeater is

\* which does not have the paging feature but we can do it by coding.

\* one which can hold other controls and can embed objects.

\* It can embed a Datagrid within it but not viceversa.

Apart from this a Data Repeater

--is used in places where you need more control over the rendering of your data

-- have very flexible templates that give you total control over the formatting of your data

**Can you edit data in the Repeater control?**

No, it just reads the information from its data source

**Which ADO.NET object is very fast in getting data from database?**

SqlDataReader object. (Note: Even datasets also use SqlDataReader objects internally for retriving data from database.)

**Can you edit data in Repeater control?**

No, it is readonly and forward only control so we can't edit data in repeater control.

**How do I insert records using data reader?**

The procedure for updating records using INSERT commands is very similar to the one we presented in the previous example (of SELECT) except that here the command does not return anything and thus the method to call on the SqlCommand object is called ExecuteNonQuery().  
  
C# Version  
  
string connString = "server=FARAZ; database=programmersheaven;" +  
"uid=sa; pwd=";  
SqlConnection conn = new SqlConnection(connString);  
  
// INSERT Query  
string cmdString ="INSERT INTO Author " +  
"(authorId, name) " +  
"VALUES(3, 'Anders Hejlsberg')";  
  
SqlCommand cmd = new SqlCommand(cmdString, conn);  
conn.Open();  
cmd.ExecuteNonQuery();  
conn.Close();

A Data Reader is a Forword Only and Read Only Cursor, which is use to fetch data from the database. It cannot be use to Insert any data.  
  
The above answer is OK but they are not using any datareader object. They are just associating a sql query to the command objects and executed command.ExecuteNonQuery() which does'nt use any DataReader object.

**What does it mean by connected data access architecture of ADO.Net?**

In the connected environment, it is your responsibility to open and close the database connection. You first establish the database connection, perform the interested operations to the database and when you are done, close the database connection. All the changes are done directly to the database and no local (memory) buffer is maintained.

In the connected environment, we use Connection Object, Transaction Object, CommandObject, Parameter Object, DataReader Object and DataAdaptor object. Means we only use these object and do not cache a local memory object i.e., known as Dataset.

**Difference between SqlCommand and SqlCommandBuilder.**

sqlcommand is a class to writing the queries like select/delete/update/insert and sqlcommandbuilder is also class but we can update/modify the transection of data on the database at run time.  
da.update(sqlcommandbulder queries);

**What is the difference between ADO and ADO.NET?**

1.Ado is created using COM technology,whereas Ado.net is implemented by using .NET framework technology.  
  
2.In ADO we can store only one table, whereas in ado.net we can store multiple tables in the dataset.  
  
3.In ADO we can't generate relations whereas in ADO.NET we can generate relations.  
  
4.In ADO we can't generate SQL statements whereas in ADO.NET we can generate SQL statements.  
  
5.ADO is connection-oriented,but ADO.NET is connectionless.

6.In ADO we can use recordset on the other hand in ADO.NET we can use Data set. The ADO Recordset object is used to hold a set of records from a single table

**Can we connect two datareader to same data source using single connection at same time?**

Yes, you can connect two datareader to the same datasource, but one mainthing is close the first datareader before using second one then only it's possible.

The point is, we can have any number of datareaders to the same datasource, but only one of them could have an active connection at any point of time.

It is possible using MARS (Multi Active Record Set) in VS 2005. We have to set MARS =True in the connection string, so that we can use 2 datareaders for a single connection

**What are good ADO.NET object(s) to replace the ADO Recordset object?**

The differences includes  
In ADO, the in-memory representation of data is the Recordset.  
In ADO.net, it is the dataset  
  
A recordset looks like a single table in ADO  
In contrast, a dataset is a collection of one or more tables in ADO.net  
  
ADO is designed primarily for connected access  
ADO.net the disconnected access to the database is used  
  
In ADO you communicate with the database by making calls to an OLE DB provider.  
In ADO.NET you communicate with the database through a data adapter (an OleDbDataAdapter, SqlDataAdapter, OdbcDataAdapter, or OracleDataAdapter object), which makes calls to an OLE DB provider or the APIs provided by the underlying data source.  
  
In ADO you cant update the database from the recordset. ADO.NET the data adapter allows you to control how the changes to the dataset are transmitted to the database.

**How to copy the contents from one table to another table and how to delete the source table in ado.net?**

SELECT \* INTO newTable FROM oldTable.  
  
This will copy the table structure and data in it  
  
If you require the structure of the table one[i.e. no data] then  
  
SELECT \* INTO newTable FROM oldTable WHERE 0 = 1

Now, execute the "drop" command on old table.

**How to select data set or data reader?**

The data reader is more useful when you need to work with large number of tables, database in non-uniform pattern and you need not execute the large no. of queries on few particular table.  
When you need to work on fewer no. of tables and most of the time you need to execute queries on these fewer tables, you should go for the dataset.  
  
It also depends on the nature of application. If multiple users are using the database and the database needs to be updated every time, you must not use the dataset. For this, .Net provides the connection oriented architecture. But in the scenarios where instant update of database is not required, dataset provides optimal performance by making the changes locally and connecting to database later to update a whole batch of data. This also reduces the network bandwidth if the database is accessed through network.  
  
Disconnected data access is suited most to read only services. On the down side, disconnected data access architecture is not designed to be used in the networked environment where multiple users are updating data simultaneously and each of them needs to be aware of current state of database at any time (e.g., Airline Reservation System).

**What is the Dot Net Framework data provider for SQL Server?**

The dot net framework data provider for SQL Server is the optimized data provider for Microsoft SQL Server 7 or later. It is recommended to use SQL Server data provider to access the SQL Server DB than general provider like OLEDB. The classes for this provider are present in the System.Data.SqlClient namespace.From there we can access the provider classes:- SqlConnection, SqlCommand, SqlParameter, SqlTransaction, SqlDataReader, SqlDataAdapter.

**How do I read data (or records) from database using data reader?**

To read data from the database, you first make a connection object (SqlConnection, etc) and open it.

string connString = "server=FARAZ; database=programmersheaven;" +  
"uid=sa; pwd=";  
SqlConnection conn = new SqlConnection(connString);  
conn.Open();  
  
Then you create a command using this connection and the command text.  
  
string cmdString = "select \* from author";  
SqlCommand cmd = new SqlCommand(cmdString, conn);  
  
Then you execute the command with the command object?s ExecuteReader() method. The ExecuteReader method returns the object of type IDataReader  
  
SqlDataReader reader = cmd.ExecuteReader();  
  
Now you read the individual records using this data reader. To advance to the next record, you call its Read() method which returns Boolean to indicate if there exists a next row. If the DataReader?s Read() method returns true then the DataReader acts as a database row (record). Now you can access the fields of this particular row specifying the column names (or integral indexes) in its indexers.  
  
while(reader.Read())  
{  
txtData.Text += reader["authorId"].ToString();  
txtData.Text += ", ";   
txtData.Text += reader["name"].ToString();  
txtData.Text += "  
";  
}  
  
Finally, you need to close the database connection opened before performing the database operation (SELECT, in our case)  
  
conn.Close();  
  
Let?s look at the complete code now for review

string connString = "server=siraj; database=programmersheaven;" +  
"uid=sa; pwd=";  
SqlConnection conn = new SqlConnection(connString);  
string cmdString = "select \* from author";  
SqlCommand cmd = new SqlCommand(cmdString, conn);  
conn.Open();  
SqlDataReader reader = cmd.ExecuteReader();  
while(reader.Read())  
{  
txtData.Text += reader["authorId"].ToString();  
txtData.Text += ", ";   
txtData.Text += reader["name"].ToString();  
txtData.Text += "  
";  
}  
conn.Close();  
  
**How do I write common code for different dot net framework data providers?**

The System.Data namespace contains the interfaces implemented by different dot net framework data providers, such as:  
  
\* IDbConnection implemented by SqlConnection, OracleConnection, OleDbConnection, OdbcConnection classes represents a connection with the database server  
\* IDbCommand implemented by SqlCommand, OracleCommand, OleDbCommand, OdbcCommand classes represents an SQL command passed to the database server  
\* IDbDataAdapter implemented by SqlDataAdapter, OracleDataAdapter, OleDbDataAdapter, OdbcDataAdapter classes represents a data adapter used to fill the data set in the disconnected environment of the ADO.Net  
\* IDataReader implemented by SqlDataReader, OracleDataReader, OleDbDataReader, OdbcDataReader classes represents a data reader used to read records from the database server, analogous to read only, forward only cursor  
\* IDbTransaction implemented by SqlTransaction, OracleTransaction, OleDbTransaction, OdbcTransaction classes represents a transaction established with the database server  
  
We strongly recommend the readers to use the references of these interface type to perform the database operations wherever possible. Using these, you can write a code that is data provider independent. Consider a data access module which is supplied the database connection and which performs the database operations using this connection. This module does not know which data provider the connection belongs and uses the interface approach. Following code demonstrate this data access module  
  
internal class DataAccessModule  
{  
private IDbConnection conn;  
private IDbCommand cmd;  
private const string GetValueCmdText ="SELECT value FROM MyTable WHERE name = '";  
  
public DataAccessModule(IDbConnection conn)  
{  
this.conn = conn;  
cmd = conn.CreateCommand();  
conn.Open();  
}  
public string GetValue(string name)  
{  
cmd.CommandText = GetValueCmdText + name + "'";  
IDataReader reader = cmd.ExecuteReader();  
if(reader.Read())  
{  
return reader["value"].ToString();  
}  
else  
{  
return null;  
}  
}  
// more functions...  
}

Create your own Layer using the data Interface and instantiate the appropriate data providers as and when required.  
IDbConnection, IDbCommand, IDbDataAdapter, IDataReader, IDbTransaction.  
  
Create a Base class say for example: BaseDataProvider.cs and use the above mentioned Interfaces.  
  
In App.Config file, Create the Key DataBase ? 1 Value=? MQSQLServer?, 2 ? for Oracle and So on. And based on the Database type, return the appropriate DataProvider Object.

**What does it mean by disconnected data access architecture of ADO.Net?**

ADO.Net introduces the concept of disconnected data architecture. In traditional data access components, you make a connection to the database system and then interact with it through SQL queries using the connection. The application stays connected to the DB system even when it is not using DB services. This commonly wastes the valuable and expensive database resource as most of the time applications only query and view the persistent data. ADO.Net solves this problem by managing a local buffer of persistent data called data set. Your application automatically connects to the database server when it needs to pass some query and then disconnects immediately after getting the result back and storing it in dataset. This design of ADO.Net is called disconnected data architecture and is very much similar to the connection less services of http over the internet. It should be noted that ADO.Net also provides the connection oriented traditional data access services.

**What's the difference between accessing data with dataset or data reader?**

The dataset is generally used when you like to employ the disconnected architecture of the ADO.Net. It reads the data into the local memory buffer and perform the data operations (update, insert, delete) locally to this buffer.  
The data reader, on the other hand, is directly connected to the database management system. It passes all the queries to the database management system, which executes them and returns the result back to the application.  
Since no memory buffer is maintained by the data reader, it takes up fewer resources and performs more efficiently with small number of data operations. The dataset, on the other hand is more efficient when large number of updates are to be made to the database. All the updates are done in the local memory and are updated to the database in a batch. Since database connection remains open for the short time, the database management system does not get flooded with the incoming requests.

**Does SQLClient and OLEdb class share the same functionality?**

No, each have its own functionality,  
  
ex : for sql client , there is SqlConnection object  
  
and for oledb client , there is OleDBConnection

**How do I define a data adapter?**

The data adapter stores your command (query) and connection and using these connect to the database when asked, fetch the result of query and store it in the local dataset.  
  
The DataAdapter class (SqlDataAdapter, OracleDataAdapter, OleDbDataAdapter, OdbcDataAdapter) may be instantiated in three ways:  
  
1. by supplying the command string (SQL Select command) and connection string  
2. by supplying the command string (SQL Select command) and a connection object  
3. by supplying the command object (SqlCommand, OracleCommand, OleDbCommand, OdbcCommand)  
  
For example, with SQL Server, the data adapter is created as  
// for Sql Server  
SqlDataAdapter dataAdapter = new SqlDataAdapter(commandString, conn);

Here we have created a new instance of data adapter and supplied it command string and connection object in the constructor call.  
  
For Access, the data adapter is created like  
// for MS Access  
OleDbDataAdapter dataAdapter = new OleDbDataAdapter(commandString, connectionString);

Here we have created a new instance of data adapter and supplied it command string and connection string in the constructor call.

**What are the performance considerations when using dataset?**

Since no memory buffer is maintained by the data reader, it takes up fewer resources and performs more efficiently with small number of data operations. The dataset, on the other hand is more efficient when large number of updates are to be made to the database. All the updates are done in the local memory and are updated to the database in a batch. Since database connection remains open for the short time, the database management system does not get flooded with the incoming requests.  
However, since the dataset stores the records in the local buffer in the hierarchical form, it does take up more resources and may affect the overall performance of the application.

**How XML is supported in ADO.Net?**

The dataset is represented in the memory as an XML document. You can fill the dataset by XML and can also get the result in the form of XML. Since XML is an international and widely accepted standard, you can read the data using the ADO.Net in the XML form and pass it to other applications using Web Service. These data consuming application need not be the essentially Dot Net based. They may be written with Java, C++ or any other programming language and running on any platform.

**What is a data adapter?**

A data adapter is the component that exists between the local repository (dataset) and the physical database. It contains the four different commands (SELECT, INSERT, UPDATE and DELETE). It uses these commands to fetch the data from the DB and fill into the dataset and to perform updates done in the dataset to the physical database. It is the data adapter that is responsible for opening and closing the database connection and communicates with the dataset.

**What does it mean by Dot Net Framework Data Provider?**

Dot Net Framework Data Provider is a set of classes that establishes the database communication between an application and the database management system based on the standards of ADO.Net framework. Different data providers provide specialized and optimized connectivity to particular database management system or to a particular class of DBMS. For example, the MS SQL Server data provider provides the optimized connectivity between dot net application and MS SQL Server DBMS while the OLEDB data provider provides the uniform connectivity between dot net application and the OLEDB databases.

**How do I update a record in the table using ADO.Net dataset?**

Once you have the UpdateCommand prepared in the data adapter, you can update individual records simply by updating the field values in the data table?s rows. The above code demonstrate how we can update the ?lines? field of the third record of the table ?Article?  
DataTable dt = ds.Tables["Article"];  
dt.Rows[2]["lines"] = 600;  
da.Update(ds, "Article");  
ds.AcceptChanges();

**How do you get records number from 5 to 15 in a dataset of 100 records? Write code.**

OleDbConnection1.Open()  
OleDbDataAdapter1.Fill(DataSet21, 5, 15, ?tab?)

**What is ADO.Net?**

Most of the today?s applications need to interact with database systems to persist, edit or view data. In .Net data access service is provided through ADO.Net (ActiveX Data Object in Dot Net) components. ADO.Net is an object oriented framework that allows you to interact with database systems. We usually interact with database systems through SQL queries or stored procedures. ADO.Net encapsulates our queries and commands to provide a uniform access to various database management systems.  
  
http://www.programmersheaven.com/articles/images/faq/image001.gif  
  
ADO.Net is a successor of ADO (ActiveX Data Object). The prime features of ADO.Net are its disconnected data access architecture and XML integration.

**What is the difference between data reader and data adapter?**

DateReader is an forward only and read only cursor type if you are accessing data through DataRead it shows the data on the web form/control but you can not perform the paging feature on that record(because it's forward only type). Reader is best fit to show the Data (where no need to work on data)  
  
DataAdapter is not only connect with the Databse(through Command object) it provide four types of command (InsertCommand, UpdateCommand, DeleteCommand, SelectCommand), It supports to the disconnected Architecture of .NET show we can populate the records to the DataSet. where as Dataadapter is best fit to work on data.

**What is a dataset?**

A dataset is the local repository of the data used to store the tables and disconnected record set. When using disconnected architecture, all the updates are made locally to dataset and then the updates are performed to the database as a batch.

Dataset is a class of using System.Data; it is used to store the multiple table and is called disconnected architecture and used to fill the table in dataset using dataadapter.

**Maximum number of parameters you can pass in a Store procedure in SQL Server 2005?**

2100.

**What is basic difference between Dataset and Typed Dataset. When to use Typed dataset?**

TypeDataSet - You have to define the type of the column when creating the TypeDataset. It is useful when you know the type of the data retrieved from database. You can use type dataset as the business entity. It is physically available. Developed at design time. Performance is issue when you implement SOA.  
  
Dataset - It is normal dataset. created runtime. No need to worry about the type. Any type of data you can fill it. Creates all the column dynamically.

**Explain about rowstate in a dataset table?**

It is the current state of the row w.r.t. its relationship to the DataRowCollection.THe value of Rowstate is dependent on the method DataRow.AcceptChanges

**What is difference in Record set and Dataset?**

Recordset provides data one row at a time.

Dataset is a data structure which represents the complete table data at same time.

Recordset has the logic to update and manipulate data

Dataset is just a data store and manipulation is done through DataAdapters in .NET.

**How can we identify the controls which can be used for binding data?**

Control which drives from BaseDataBoundControl Class

Serves as the base class for controls that bind to data using an ASP.NET data source control.

**How to bind the controls(best practice) comboboxes to the data in the dataset ?**

ComboBox1.DataSorce = ds;

ComboBox1.DataTextField = "ColumnNameOfTheTable";

ComboBox1.DataBind();

**How can i identify the updated rows in a dataset?**

If the RowState property of the DataRow is "Modified" then that DataRow can be treated as updated.  
  
If dr.RowState=DataRowState.Modified then  
' u r logic here  
End if

**What is the difference between Close and Dispose?**

There is one major difference between calling the Close and Dispose methods on database connections. Close leaves the connection in a closed state; but, it is reusable—all properties, *etc.* can be accessed and Open can be called. On the other hand, after calling Dispose on a database connection—as with any object—, the connection object can no longer be accessed.

However, calling Dispose does *not* remove the connection from the connection pool.

The basic difference between Close() and Dispose() is, when a Close() method is called, any managed resource can be temporarily closed and can be opened once again. It means that, with the same object the resource can be reopened or used. Where as Dispose() method permanently removes any resource ((un)managed) from memory for cleanup and the resource no longer exists for any further processing.

*Visual C# Best Practice*

Only call the Close method on a Stream or a database connection if the object will be reused. Otherwise, use the Dispose method.

The Dispose method may be called on any instance of any type implementing the IDisposable interface. This is supported by the C# using statement which makes calling Dispose automatically easy.

*Example showing difference between Close() and Dispose() Method:*  
using System;  
using System.Data;  
using System.Data.SqlClient;  
public class Test  
{  
private string connString = "Data Source=COMP3;Initial Catalog=Northwind;UserId=sa;Password=pass";  
private SqlConnection connection;  
public Test()  
{  
connection = new SqlConnection(connString);  
}  
private static void Main()  
{  
Test t = new Test();  
t.ConnectionStatus();  
Console.ReadLine();  
}  
public void ConnectionStatus()  
{  
try  
{  
if(connection.State == ConnectionState.Closed)  
{  
connection.Open();  
Console.WriteLine("Connection opened..");  
}  
  
if(connection.State == ConnectionState.Open)  
{  
connection.Close();  
Console.WriteLine("Connection closed..");  
}  
// connection.Dispose();  
  
if(connection.State == ConnectionState.Closed)  
{  
connection.Open();  
Console.WriteLine("Connection again opened..");  
}  
}  
catch(SqlException ex)  
{  
Console.WriteLine(ex.Message+"\n"+ex.StackTrace);  
}  
catch(Exception ey)  
{  
Console.WriteLine(ey.Message+"\n"+ey.StackTrace);  
}  
finally  
{  
Console.WriteLine("Connection closed and disposed..");  
connection.Dispose();  
}  
}  
}  
  
In the above example if you uncomment the "connection.Dispose()" method and execute, you will get an exception as, "The ConnectionString property has not been initialized.".This is the difference between Close() and Dispose().