DOT NET TECHNOLOGIES

UNIT-4

ADO.NET Introduction:

It is a module of .Net Framework which is used to establish connection between application and data sources. Data sources can be such as SQL Server and XML. ADO.NET consists of classes that can be used to connect, retrieve, insert and delete data.

All the ADO.NET classes are located into System.Data.dll and integrated with XML classes located into System.Xml.dll.

Components of ADO.NET:

1. 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.

These are the components that are designed for data manipulation and fast access to data. It provides various objects such as **Connection, Command, DataReader** and **DataAdapter** that are used to perform database operations.

2. The DataSet:

It is a collection of data tables that contain the data. It is used to fetch data without interacting with a Data Source that's why, it also known as **disconnected** data access method. It is an in-memory data store that can hold more than one table at the same time. We can use DataRelation object to relate these tables. The DataSet can also be used to read and write data as XML document.

ADO.NET provides a DataSet class that can be used to create DataSet object. It contains constructors and methods to perform data related operations.

It is used to access data independently from any data resource. DataSet contains a collection of one or more DataTable objects of data.

Objects of ADO.NET:

1. Connection

The Connection object is the first component of ADO.NET. The connection object opens a connection to your data source.

Connection object helps in accessing and manipulating a database. Database transactions are also dependent upon the Connection object.

The following are the commonly used connections in the ADO.NET

- SqlConnection
- OleDbConnection
- OdbcConnection

2. Command

The Command object is used to perform an action on the data source. Command object can execute stored procedures and T-SQL commands.

You can execute SQL queries to return data in a DataSet or a DataReader object. Command object performs the standard Select, Insert, Delete, and Update T-SQL operations.

3. DataReader

The DataReader is built as a way to retrieve and examine the rows returned in response to your query as quickly as possible.

The data returned by a DataReader is always read-only. This class was built to be a lightweight forward-only, read-only, way to run through data quickly (this was called a **firehose cursor** in ADO).

However, if you need to manipulate schema or use some advanced display features such as automatic paging, you must use a DataAdapter and DataSet.

DataReader object works in a connected model.

4. DataAdapter

The DataAdapter takes the results of a database query from a Command object and pushes them into a DataSet using the DataAdapter.Fill() method. Additionally, the DataAdapter.Update() method will negotiate any changes to a DataSet back to the original data source.

DataAdapter object works in a connected model. DataAdapter performs the five following steps:

- Create/open the connection
- Fetch the data as per command specified
- Generate XML file of data

- Fill data into DataSet.
- Close connection.

5. Command Builder

It is used to save changes made in an in-memory cache of data on the backend. The work of Command Builder is to generate Command as per changes in DataRows.

Command Builder generates command on basis of row state. There is a five-row state:

1. Unchanged
2. Added
3. Deleted
4. Modified
5. Detached

Command Builder works on add, delete, and modified row state only.

Detached is used when an object is not created from row state.

6. Transaction

The Transaction object is used to execute the backend transaction. Transactions are used to ensure that multiple changes to database rows occur as a single unit of work.

The Connection class has a **BeginTransaction** method that can be used to create a Transaction.

A definite best practice is to ensure that Transactions are placed in Using statements for rapid cleanup if they are not committed. Otherwise, the objects (and any internal locks that may be needed) will remain active until the GC gets around to cleaning it up.

7. Parameters

Parameter object is used to solve the SQL Injection attack problem while dealing with the user input parameters.

Parameter object allows passing parameters into a Command object the Parameter class allows you to quickly put parameters into a query without string concatenation.

ADO.NET DataTable:

DataTable represents relational data into tabular form. ADO.NET provides a DataTable class to create and use data table independently. It can also be used with DataSet also. Initially, when we create DataTable, it does not have table schema. We can create table schema by adding columns and constraints to the table. After defining table schema, we can add rows to the table.

We must include System.Data namespace before creating DataTable.

DataTable Properties:

1. **Columns:** It is used to get the collection of columns that belong to this table.
2. **Constraints:** It is used to get the collection of constraints maintained by this table.
3. **DataSet:** It is used to get the DataSet to which this table belongs.
4. **Default View:** It is used to get a customized view of the table that may include a filtered view.
5. **Has Errors:** It is used to get a value indicating whether there are errors in any of the rows in the table of the DataSet.
6. **Minimum Capacity:** It is used to get or set the initial starting size for this table.
7. **Primary Key:** It is used to get or set an array of columns that function as primary keys for the data table.
8. **Rows:** It is used to get the collection of rows that belong to this table.
9. **Table Name:** It is used to get or set the name of the DataTable.

DataTable Constructors:

The following table contains the DataTable class constructors.

Constructors	Description
DataTable()	It is used to initialize a new instance of the DataTable class with no arguments.
DataTable(String)	It is used to initialize a new instance of the DataTable class with the specified table name.
DataTable(SerializationInfo, StreamingContext)	It is used to initialize a new instance of the DataTable class with the SerializationInfo and the StreamingContext.
DataTable(String, String)	It is used to initialize a new instance of the DataTable class using the specified table name and namespace.

DataTable Methods:

The following table contains the DataTable class methods.

Method	Description
AcceptChanges()	It is used to commit all the changes made to this table.
Clear()	It is used to clear the DataTable of all data.

Clone()	It is used to clone the structure of the DataTable.
Copy()	It is used to copy both the structure and data of the DataTable.
CreateDataReader()	It is used to returns a DataTableReader corresponding to the data within this DataTable.
CreateInstance()	It is used to create a new instance of DataTable.
GetRowType()	It is used to get the row type.
GetSchema()	It is used to get schema of the table.
ImportRow(DataRow)	It is used to copy a DataRow into a DataTable.
Load(IDataReader)	It is used to fill a DataTable with values from a data source using the supplied IDataReader.
Merge(DataTable, Boolean)	It is used to merge the specified DataTable with the current DataTable.
NewRow()	It is used to create a new DataRow with the same schema as the table.
Select()	It is used to get an array of all DataRow objects.
WriteXml(String)	It is used to write the current contents of the DataTable as XML using the specified file.

ADO.NET DataAdapter:

The DataAdapter works as a bridge between a DataSet and a data source to retrieve data. DataAdapter is a class that represents a set of SQL commands and a database connection. It can be used to fill the DataSet and update the data source.

DataAdapter Class Signature:

public class DataAdapter : System.ComponentModel.Component, System.Data.IDataAdapter

DataAdapter Constructors:

Constructors	**Description**
DataAdapter()	It is used to initialize a new instance of a DataAdapter class.

DataAdapter(DataAdapter)	It is used to initializes a new instance of a DataAdapter class from an existing object of the same type.

Methods:

Method	Description
CloneInternals()	It is used to create a copy of this instance of DataAdapter.
Dispose(Boolean)	It is used to release the unmanaged resources used by the DataAdapter.
Fill(DataSet)	It is used to add rows in the DataSet to match those in the data source.
FillSchema(DataSet, SchemaType, String, IDataReader)	It is used to add a DataTable to the specified DataSet.
GetFillParameters()	It is used to get the parameters set by the user when executing an SQL SELECT statement.
ResetFillLoadOption()	It is used to reset FillLoadOption to its default state.
ShouldSerializeAcceptChangesDuringFill()	It determines whether the AcceptChangesDuringFill property should be persisted or not.
ShouldSerializeFillLoadOption()	It determines whether the FillLoadOption property should be persisted or not.
ShouldSerializeTableMappings()	It determines whether one or more DataTableMapping objects exist or not.
Update(DataSet)	It is used to call the respective INSERT, UPDATE, or DELETE statements.

The DataSet:

It is a collection of data tables that contain the data. It is used to fetch data without interacting with a Data Source that's why, it also known as **disconnected** data access method. It is an in-memory data store that can hold more than one table at the same time. We can use DataRelation object to relate these tables. The DataSet can also be used to read and write data as XML document.

ADO.NET provides a DataSet class that can be used to create DataSet object. It contains constructors and methods to perform data related operations.

It is used to access data independently from any data resource. DataSet contains a collection of one or more DataTable objects of data.

DataSet Constructors:

Constructor	Description
DataSet()	It is used to initialize a new instance of the DataSet class.
DataSet(String)	It is used to initialize a new instance of a DataSet class with the given name.
DataSet(SerializationInfo, StreamingContext)	It is used to initialize a new instance of a DataSet class that has the given serialization information and context.
DataSet(SerializationInfo, StreamingContext, Boolean)	It is used to initialize a new instance of the DataSet class.

DataSet Properties:

1. **Case Sensitive:** It is used to check whether DataTable objects are case-sensitive or not.
2. **Dataset Name:** It is used to get or set name of the current DataSet.
3. **Default View Manager:** It is used to get a custom view of the data contained in the DataSet to allow filtering and searching.
4. **Has Errors:** It is used to check whether there are errors in any of the DataTable objects within this DataSet.
5. **Is Initialized:** It is used to check whether the DataSet is initialized or not.
6. **Locale:** It is used to get or set the locale information used to compare strings within the table.
7. **Namespace:** It is used to get or set the namespace of the DataSet.
8. **Site:** It is used to get or set an ISite for the DataSet.
9. **Tables:** It is used to get the collection of tables contained in the DataSet.

DataSet Methods:

The following table contains some commonly used methods of DataSet.

Method	Description
BeginInit()	It is used to begin the initialization of a DataSet that is used on a form.

Clear()	It is used to clear the DataSet of any data by removing all rows in all tables.
Clone()	It is used to copy the structure of the DataSet.
Copy()	It is used to copy both the structure and data for this DataSet.
CreateDataReader(DataTable[])	It returns a DataTableReader with one result set per DataTable.
CreateDataReader()	It returns a DataTableReader with one result set per DataTable.
EndInit()	It ends the initialization of a DataSet that is used on a form.
GetXml()	It returns the XML representation of the data stored in the DataSet.
GetXmlSchema()	It returns the XML Schema for the XML representation of the data stored in the DataSet.
Load(IDataReader, LoadOption, DataTable[])	It is used to fill a DataSet with values from a data source using the supplied IDataReader.
Merge(DataSet)	It is used to merge a specified DataSet and its schema into the current DataSet.
Merge(DataTable)	It is used to merge a specified DataTable and its schema into the current DataSet.
ReadXml(XmlReader, XmlReadMode)	It is used to read XML schema and data into the DataSet using the specified XmlReader and XmlReadMode.
Reset()	It is used to clear all tables and removes all relations, foreign constraints, and tables from the DataSet.
WriteXml(XmlWriter, XmlWriteMode)	It is used to write the current data and optionally the schema for the DataSet using the specified XmlWriter and XmlWriteMode.

SQL DataReader:

This class is used to read data from SQL Server database. It reads data in forward-only stream of rows from a SQL Server database. it is sealed class so that cannot be inherited. It inherits DbDataReader class and implements IDisposable interface.

SQL DataReader Signature:

public class SqlDataReader : System.Data.Common.DbDataReader, IDisposable

SQL DataReader Properties:

1. **Connection:** It is used to get the SQL Connection associated with the SqlDataReader.
2. **Depth:** It is used to get a value that indicates the depth of nesting for the current row.
3. **Field Count:** It is used to get the number of columns in the current row.
4. **Has Rows:** It is used to get a value that indicates whether the SqlDataReader contains one or more rows.
5. **Is Closed:** It is used to retrieve a boolean value that indicates whether the specified SqlDataReader instance has been closed.
6. **Item[String]:** It is used to get the value of the specified column in its native format given the column name.
7. **Item [Int32]:** It is used to get the value of the specified column in its native format given the column ordinal.
8. **Records Affected:** It is used to get the number of rows changed, inserted or deleted by execution of the Transact-SQL statement.
9. **Visible Field Count:** it is used to get the number of fields in the SqlDataReader that are not hidden.

Methods:

Method	Description
Close()	It is used to closes the SqlDataReader object.
GetBoolean(Int32)	It is used to get the value of the specified column as a Boolean.
GetByte(Int32)	It is used to get the value of the specified column as a byte.
GetChar(Int32)	It is used to get the value of the specified column as a single character.
GetDateTime(Int32)	It is used to get the value of the specified column as a DateTime object.
GetDecimal(Int32)	It is used to get the value of the specified column as a Decimal object.

GetDouble(Int32)	It is used to get the value of the specified column as a double-precision floating point number.
GetFloat(Int32)	It is used to get the value of the specified column as a single-precision floating point number.
GetName(Int32)	It is used to get the name of the specified column.
GetSchemaTable()	It is used to get a DataTable that describes the column metadata of the SqlDataReader.
GetValue(Int32)	It is used to get the value of the specified column in its native format.
GetValues(Object[])	It is used to populate an array of objects with the column values of the current row.
NextResult()	It is used to get the next result, when reading the results of SQL statements.
Read()	It is used to read record from the SQL Server database.

Data View:

A DataView represents a view a DataSet object. You can set filters on the data or sort on data in the DataSet through different DataViews and produce different views of the data.

For example, you can create a DataSet with three tables and create three different DataView objects for each table. Once you have a Data View object, you can attach it with any data-bound Control. Such as a DataGrid or a ComboBox control using data-bound control using data-bound control's DataSource property.

OLEDB:

An OleDb Connection object represents a unique connection to a data source. With a client/server database system, it is equivalent to a network connection to the server. Depending on the functionality supported by the native OLE DB provider, some methods or properties of an OleDbConnection object may not be available.

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

You can open more than one DataReader on a single OleDbConnection. If the OLE DB provider you use does not support more than one DataReader on a single connection, the provider implicitly opens an additional connection for each.

If the OleDbConnection goes out of scope, it is not closed. Therefore, you must explicitly close the connection by calling Close or Dispose, or by using the OleDbConnection object within a Using statement.

OLEDB Connection String Keywords:

Data Source: The name of the database or physical location of the database file.

File Name: The physical location of a file that contains the real connection string.

Persist Security Info: If set to true, retrieving the connection string returns the complete connection string that was originally provided. If set to false, the connection string will contain the information that was originally provided, minus the security information.

Provider: The vendor-specific driver to use for connecting to the data store.

The following providers have been tested with ADO.NET.

Driver	Provider
SQLOLEDB	Microsoft OLE DB provider for SQL Server
MSDAORA	Microsoft OLE DB provider for Oracle
Microsoft.Jet.OLEDB.4.0	OLE DB provider for Microsoft Jet

Manage Providers:

Managed Providers are the bridge from a data store to a .NET application

The Managed Providers have four core components:

Connection—The Connection represents a unique session to a data store. This may be manifested as a network connection in a client/server database application.

Command—The Command represents a SQL statement to be executed on a data store.

DataReader—The DataReader is a forward-only, read-only stream of data records from a data store to a client.

DataAdapter—The DataAdapter represents a set of Commands and a Connection which are used to retrieve data from a data store and fill a DataSet.

The Two Managed Providers

ADO.NET, the successor to Microsoft's highly successful ActiveX Data Objects (ADO), offers two Managed Providers. These providers are similar in their object model, but are chosen at design-time based on the data provider being used. The SQL Managed Provider offers a direct link into Microsoft's SQL Server database application (version 7.0 or higher), while the OleDb Managed Provider is used for all other data providers. Following is a brief description of each of the

Managed Providers. Throughout this chapter we will show you how the Managed Providers work, and specify when a particular object, property, method or event is proprietary to only one of the Managed Providers.

OleDb Managed Provider

The OleDb Managed Provider uses native OLEDB and COM Interop to establish a connection to a data store and negotiate commands. The OleDb Managed Provider is the data access provider to use when you are working with data from any data source that is not Microsoft's SQL Server 7.0 or higher. To use the OleDb Managed Provider, you must import the System.Data.OleDb namespace.

SQL Managed Provider

The SQL Managed Provider is designed to work directly with Microsoft SQL Server 7.0 or greater. It connects and negotiates directly with SQL Server without using OLEDB. This provides a better performance model than the OleDb Managed Provider, but it's restricted to use with Microsoft SQL Server 7.0 or higher. To use the SQL Managed Provider, you must import the System.Data.SqlClient namespace.