

Unit 5

ADO.NET

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**.

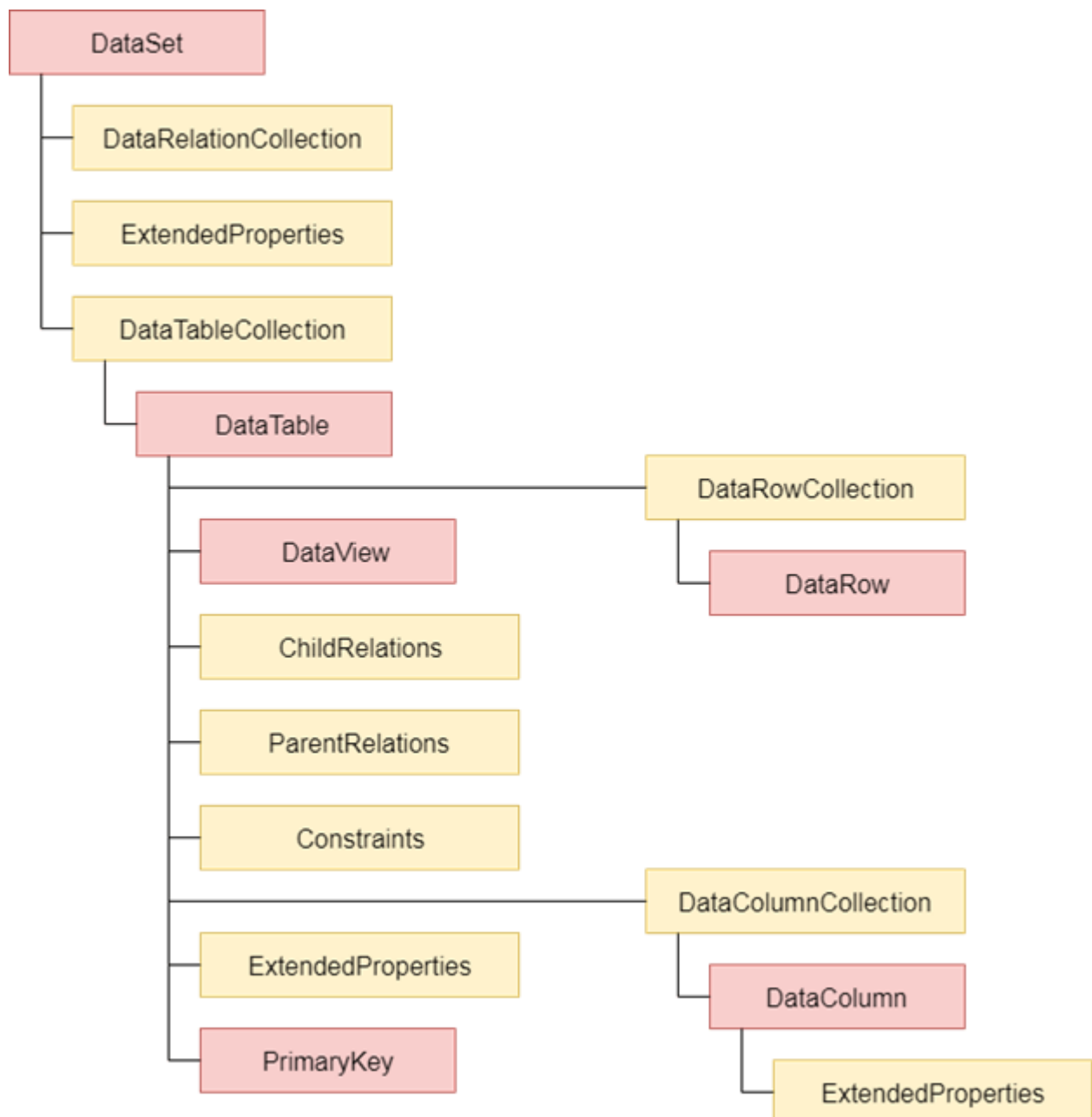
ADO.NET has two main components that are used for accessing and manipulating data are the .NET Framework data provider and the DataSet.

.NET Framework Data Providers

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. We will have a detailed discussion about **Data Providers** in new topic.

The DataSet

It is used to access data independently from any data resource. DataSet contains a collection of one or more DataTable objects of data. The following diagram shows the relationship between .NET Framework data provider and DataSet.



ADO Object Model

ADO.NET (Active Data Objects for .NET) is a data access technology provided by Microsoft as part of the .NET Framework. It allows developers to interact with data from various data sources, such as databases and XML files, in a managed and efficient manner. ADO.NET provides a rich set of classes and components for data access, and its object model consists of several key components:

Connection (**System.Data.SqlClient.SqlConnection**)

- Represents a connection to a data source, such as a database server.
- Manages the connection state (open or closed).

- Provides properties for connection string, timeout, and more.

Command (System.Data.SqlClient.SqlCommand):

- Represents a SQL command or stored procedure to be executed against a data source.
- Can be used to execute SQL queries, insert, update, delete records, and call stored procedures.
- Provides parameters for passing values to queries.

DataReader (System.Data.SqlClient.SqlDataReader):

- Provides a forward-only, read-only stream of data from a data source, typically after executing a query.
- Efficient for retrieving large result sets with low memory overhead.
- Requires an open connection while reading data.

DataAdapter (System.Data.SqlClient.SqlDataAdapter):

- Acts as a bridge between a DataSet and a data source.
- Fills a DataSet with data from the data source (via the Fill method) and updates the data source with changes made to the DataSet (via the Update method).

DataSet (System.Data.DataSet):

- Represents an in-memory, disconnected cache of data.
- Consists of one or more DataTables, which contain rows and columns of data.
- Provides methods for querying, updating, and synchronizing data with a data source.

DataTable (System.Data.DataTable):

- Represents a single table of data within a DataSet.
- Contains DataColumn and DataRow objects.
- Supports filtering, sorting, and searching operations.

DataColumn (System.Data.DataColumn):

- Represents a column of data within a DataTable.
- Defines the data type and constraints for the column.

DataRow (System.Data.DataRow):

- Represents a single row of data within a DataTable.
- Allows access to individual data values through indexing or column names.

Parameter (System.Data.SqlClient.SqlParameter):

- Represents a parameter to be passed to a SQL command or stored procedure.
- Helps prevent SQL injection by separating data from SQL commands.

DataView (System.Data.DataView):

- Provides a customized view of a DataTable that can be sorted, filtered, and searched.
- Useful for presenting data in a specific order or applying complex filters.

Connection Pooling:

- ADO.NET includes built-in connection pooling to efficiently manage database connections, improving performance and resource utilization.

Transaction (System.Data.SqlClient.SqlTransaction):

- Represents a database transaction that can include multiple SQL commands.
- Ensures data integrity by allowing either all or none of the changes to be committed to the database.
- ADO.NET is database-agnostic, which means it can work with various database management systems like Microsoft SQL Server, Oracle, MySQL, and more. Developers can choose the appropriate data provider (e.g., System.Data.SqlClient for SQL Server) while using the same ADO.NET programming model. This flexibility makes ADO.NET a widely used data access technology in the .NET ecosystem for building data-driven applications.

Difference between ADO and ADO.NET

This article will learn the difference between the ADO and ADO.NET technology in detail. Firstly we will learn the basic introduction of both the technologies after that we will see the differences between both of them.

ADO:

ADO stands for ActiveX Data Objects. In this the programmers use a generic set of objects, no matter the underlying data sources. It requires locking of database resources and a lengthy connections for applications. In ADO connection is creating by using a single Connection object.

ADO.NET:

ADO.NET is an advance version of ADO. It interacts with databases and have a consistent access to various data sources. It does not requires locking and a lengthy connections for its applications. ADO.NET can have a separate objects that represent connections to different data sources which make access faster and more efficient.

It uses a multilayered architecture that includes a few concepts such as Connection, Command, and DataSet objects. ADO.NET is truly the data access technology for current and future applications.

Head to Head differences	ADO	ADO.NET
1 Data Representation	In ADO, data is represented by a RecordSet object, which can hold data from one data source at a time.	In ADO.NET, data is represented by a DataSet object, which can hold data from various data sources, integrate it, and after combining all the data it can write it back to one or several data sources.
2 Data Reads	In ADO, data is read in sequential manner.	In ADO.NET, data is read either in a sequential or non-sequential manner.
3 Disconnected Access	ADO first makes a call to an OLE DB provider. It supports the connected access, represented by the Connection with a database.	It uses standardized calls such as DataSetCommand object to communicate with a database and with the OLE DB provider.
4 Creating Cursors	It allows us to create client-side cursor only.	ADO.NET either created client-side or server-side cursors.

5 Locking	ADO has the feature of locking.&	This features is not used in ADO.NET.
6 XML integration	This feature is not used in ADO.	This feature is used in ADO.NET.
7 Relationship between Multiple Tables	It requires SQL Join and UNIONs to combine data from multiple tables in a single RecordSet.	It uses DataRelation objects for combining data from multiple tables without the use of SQL Join and UNIONs.
8 Conversion	ADO requires data to be converted to data types supported by receiving system.	ADO.NET does not require complex conversions that wasted processor time.
9 Data Storage	In ADO data is stored in a binary form.	In ADO.NET, data is stored in XML form. With the use of XML, it stores data more easily.
10 Firewalls	In ADO, firewalls are typically configured to prevent system-level requests, such as COM marshaling.	In ADO.NET firewalls are supported because ADO.NET DataSet objects use XML for representing data.
11 Communication	In this, objects are communicate in a binary mode.	It uses XML for communication.
12 Programmability	It uses a Connection object to transfer commands to a data source with defined data constructs.	ADO.NET does not require data constructs because it uses strongly typed characteristics of XML.
13 Connection Model	ADO supports only one model of connection i.e. Connection Oriented Data Access Architecture Model.	ADO.NET supports two models of connection i.e. Connection-Oriented Data Access Architecture and Disconnected Oriented Data Access Architecture.
14. Environment	ADO was geared for a two-tiered architecture.	ADO.NET address a multi-tiered architecture.

15. Metadata	It derives information automatically at runtime based on metadata.	In ADO.NET derives information at design time on metadata in order to provide better runtime performance.
16. Multiple Transactions	In ADO multiple transactions cannot be send using a single connection.	In ADO.NET multiple transactions can be send using a single connection.
17. Data Types	It has a less number of data types.	It has a huge and rich collection of data types.
18. Serialization	ADO serialized data in a proprietary protocol.	ADO.NET serialized data using XML.
19. Security	ADO is less secured.	ADO.NET much secured as compared to ADO.
20 Based on	ADO is based on Component Object Modelling.	ADO.NET is a Common Language Runtime based library.
21 Scalability	ADO technology have less scalable.	ADO.NET have more scalability with the use of locking mechanism.
22. Resources	ADO technology use more resources as compared to ADO.NET.	ADO.NET conserves limited resources.
23. Performance	ADO have a poor performance.	In ADO.NET performance is much better as compared to ADO.

Connected Architecture of ADO.NET

The ADO.NET is one of the Microsoft data access technologies which is used to establish a connection between the .NET Application (Console, WCF, WPF, Windows, MVC, Web Form, etc.) and different data sources such as SQL Server, Oracle, MySQL, XML, etc. The ADO.NET framework access the data from data sources in two different ways. The models are Connection Oriented Data Access Architecture and Disconnected Data Access Architecture. In this article, I will explain both these architectures in detail with Examples.

Types of Architecture to Access the Data using ADO.NET

The Architecture supported by ADO.NET for communicating with data sources is categorized into two models. They are as follows:

1. Connected Oriented Architecture
2. Disconnected Oriented Architecture

So, ADO.NET supports both Connection-Oriented Architectures as well as Disconnection-Oriented Architecture. Depending upon the functionality or business requirement of an application, we can make it either Connection-Oriented or Disconnection-Oriented. Even, it is also possible to use both Architectures together in a single .NET application to communicate with different data sources.

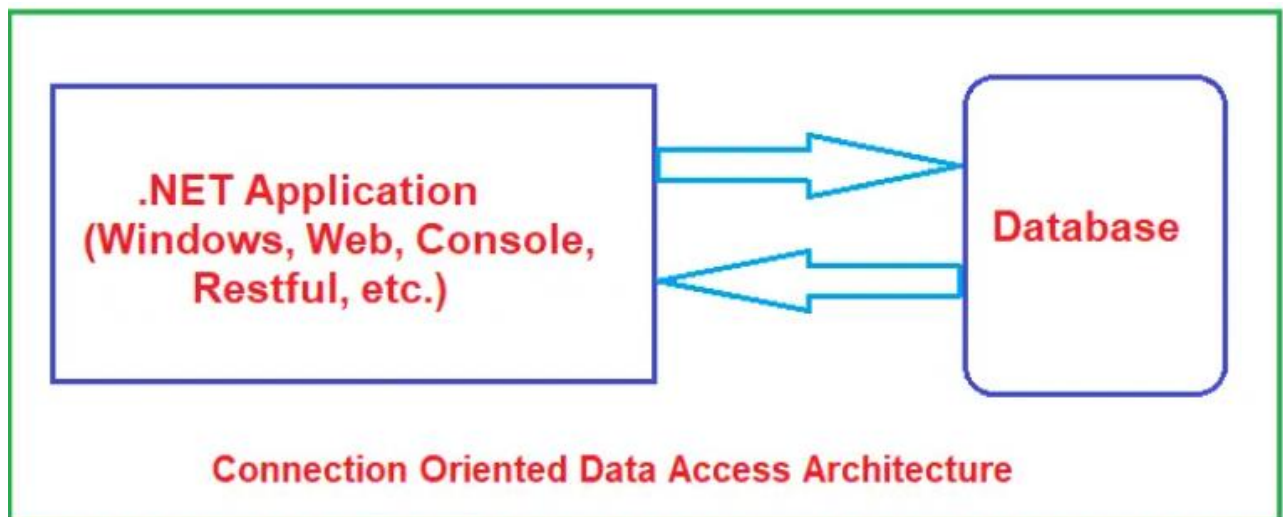
ADO.NET Connection-Oriented Data Access Architecture

In the case of Connection Oriented Data Access Architecture, always an open and active connection is required in between the .NET Application and the database. An example is Data Reader and when we are accessing the data from the database, the Data Reader object requires an active and open connection to access the data, If the connection is closed then we cannot access the data from the database and in that case, we will get the runtime error.

The Connection Oriented Data Access Architecture is always forward only. That means using this architecture mode, we can only access the data in the forward direction. Once we read a row, then it will move to the next data row and there is no chance to move back to the previous row.

The Connection Oriented Data Access Architecture is read-only. This means using this architecture we can only read the data. We cannot modify the data i.e. we cannot update and delete the data row.

For Connection Oriented Architecture, we generally use the object of the ADO.NET DataReader class. The DataReader object is used to retrieve the data from the database and it also ensures that an open and active connection should be there while accessing the data from the database. In Connection Oriented Architecture, the .NET Application is directly linked with the corresponding Database.



ADO.NET DataReader in Connected-Oriented Architecture

The ADO.NET DataReader object is used to read the data from the database using Connected Oriented Architecture. It works in forward only and only mode. It requires an active and open connection while reading the data from the database.

Example to Understand Connection-Oriented Architecture

We are going to use the following Employee tables to understand Connection-Oriented Architecture as well as Disconnection-Oriented Architecture using ADO.NET.

Id	Name	Email	Mobile
100	Anurag	Anurag@dotnettutorial.net	1234567890
101	Priyanka	Priyanka@dotnettutorial.net	2233445566
102	Preety	Preety@dotnettutorial.net	6655443322
103	Sambit	Sambit@dotnettutorial.net	9876543210

Please use the following SQL Script to create the EmployeeDB and populate the Employee table with the required sample data.

```
CREATE DATABASE EmployeeDB;

GO

USE EmployeeDB;

GO

CREATE TABLE Employee(
Id INT IDENTITY(100, 1) PRIMARY KEY,
Name VARCHAR(100),
Email VARCHAR(50),
Mobile VARCHAR(50),
)

GO

INSERT INTO Employee VALUES
('Anurag','Anurag@dotnettutorial.net','1234567890')

INSERT INTO Employee VALUES
('Priyanka','Priyanka@dotnettutorial.net','2233445566')

INSERT INTO Employee VALUES ('Preety','Preety@dotnettutorial.net','6655443322')

INSERT INTO Employee VALUES ('Sambit','Sambit@dotnettutorial.net','9876543210')

GO
```

Using ADO.NET Data Reader to Fetch the Data from the Database

In the below example, I am using the ADO.NET Data Reader object to Fetch the Data from the Database. As Data Reader in ADO.NET works on Connection-Oriented Architecture, so it always requires an active and open connection to access the data from the database.

```
using System;

using System.Data.SqlClient;

namespace ConnectionOrientedArchitecture
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                string ConString = @"data source=LAPTOP-ICA2LCQL\SQLEXPRESS;
database=EmployeeDB; integrated security=SSPI";

                using (SqlConnection connection = new SqlConnection(ConString))
                {
                    // Creating the command object

                    SqlCommand cmd = new SqlCommand("select Name, Email, Mobile from Employee",
                    connection);

                    // Opening Connection

                    connection.Open();

                    // Executing the SQL query

                    SqlDataReader sdr = cmd.ExecuteReader();

                    //Looping through each record

                    //SqlDataReader works in Connection Oriented Architecture

                    //So, it requires an active and open connection while reading the data

                    //from the database

                    while (sdr.Read())
```

```

{
Console.WriteLine(sdr["Name"] + ", " + sdr["Email"] + ", " + sdr["Mobile"]);
}

//Here, the connection is going to be closed automatically
}

catch (Exception ex)
{
Console.WriteLine($"Exception Occurred: {ex.Message}");
}

Console.ReadKey();
}
}
}

```

Output

Now, let us do one thing. After reading the first row from the database, let us close the connection and see what happens. In the below example, you can see, within the while loop, after reading the first row, we are closing the database connection by calling the Close method.

```

using System;

using System.Data.SqlClient;

namespace ConnectionOrientedArchitecture
{
class Program
{
static void Main(string[] args)
{

```

```
try
{
    string ConString = @"data source=LAPTOP-ICA2LCQL\SQLEXPRESS;
    database=EmployeeDB; integrated security=SSPI";

    using (SqlConnection connection = new SqlConnection(ConString))
    {
        // Creating the command object

        SqlCommand cmd = new SqlCommand("select Name, Email, Mobile from Employee",
        connection);

        // Opening Connection

        connection.Open();

        // Executing the SQL query

        SqlDataReader sdr = cmd.ExecuteReader();

        //Looping through each record

        //SqlDataReader works in Connection Oriented Architecture

        //So, it requires an active and open connection while reading the data

        //from the database

        while (sdr.Read())
        {
            //Read-only, you cannot modify the data

            //sdr["Name"] = "PKR";

            Console.WriteLine(sdr["Name"] + ", " + sdr["Email"] + ", " + sdr["Mobile"]);

            connection.Close();//Here, the connection is closed
        }
    }
}
```

```
catch (Exception ex)
{
    Console.WriteLine($"Exception Occurred: {ex.Message}");
}

Console.ReadKey();
}
}
}
```

ADO.NET Disconnection-Oriented Data Access Architecture

In the case of Disconnection Oriented Data Access Architecture, always an open and active connection is not required in between the .NET Application and the database. In this architecture, Connectivity is required only to read the data from the database and to update the data within the database.

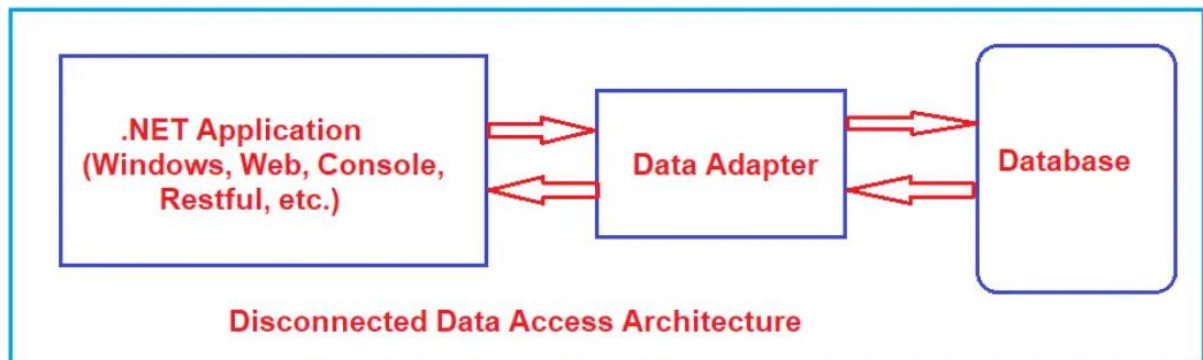
An example is DataAdapter and DataSet or DataTable classes. Here, using the DataAdapter object and using an active and open connection, we can retrieve the data from the database and store the data in a DataSet or DataTable. The DataSets or DataTables are in-memory objects or you can say they store the data temporarily within .NET Application. Then whenever required in our .NET Application, we can fetch the data from the dataset or data table and process the data. Here, we can modify the data, we can insert new data, can delete the data from within the dataset or data tables. So, while processing the data within the .NET Application using DataSet or Datatable, we do not require an active and open connection.

And finally, when we processed the data in our .NET Application, then if we want to update the modified data which is stored inside the dataset or Datatable into the database, then we need to establish the connection again and we need to update the data in the database. This is how Disconnection Oriented Data Access Architecture works.

ADO.NET DataAdapter in Disconnection-Oriented Architecture

The ADO.NET DataAdapter object acts as an interface between the .NET application and the database. The Data Adapter object fills the Dataset or DataTable which helps the user to perform the operations on the data. And once we modify the DataSet or DataTable, then we need to pass the modified DataSet

or DataTable to the DataAdapter which will update the modified data into the database. The DataAdapter object will internally manage the connection i.e. when to establish the connection and when to terminate the connection.



The ADO.NET DataAdapter establishes a connection with the corresponding database and then retrieves the data from the database and fills the retrieved data into the DataSet or DataTable. And finally, when the task is completed i.e. the Data is processed by the application i.e. the data is modified by the application, and modified data is stored in the DataSet or DataTable. Then the DataAdapter takes the modified data from the DataSet or DataTable and updates it into the database by again establishing the connection.

So, we can say that DataAdapter acts as a mediator between the Application and database which allows the interaction in disconnection-oriented architecture.

Example to Understand Disconnected-Oriented Architecture in ADO.NET

In the below example, I am using the ADO.NET Data Adapter object to Fetch the Data from the Database and fill the DataTable. Then data table stores the data in memory and then we modified the data table data, and finally, we provided the modified data table with the Data Adapter object which will update the modified data within the database.

```
using System;

using System.Data;

using System.Data.SqlClient;

namespace BatchOperationUsingSqlDataAdapter
{
    class Program
```

```
{  
  
static void Main(string[] args)  
  
{  
  
try  
  
{  
  
// Connection string.  
  
string connectionString = @"data source=LAPTOP-ICA2LCQL\SQLEXPRESS;  
database=EmployeeDB; integrated security=SSPI";  
  
// Connect to the EmployeeDB database.  
  
using (SqlConnection connection = new SqlConnection(connectionString))  
  
{  
  
// Create a SqlDataAdapter  
  
SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM EMPLOYEE",  
connection);  
  
//Fetch the Employee Data and Store it in the DataTable  
  
DataTable dataTable = new DataTable();  
  
//The Fill method will open the connection, fetch the data, fill the data in  
  
//the data table and close the connection automatically  
  
adapter.Fill(dataTable);  
  
// Set the UPDATE command and parameters.  
  
string UpdateQuery = "UPDATE Employee SET Name=@Name, Email=@Email,  
Mobile=@Mobile WHERE ID=@EmployeeID;";  
  
adapter.UpdateCommand = new SqlCommand(UpdateQuery, connection);  
  
adapter.UpdateCommand.Parameters.Add("@Name", SqlDbType.NVarChar, 50,  
"Name");  
  
adapter.UpdateCommand.Parameters.Add("@Email", SqlDbType.NVarChar, 50,  
"Email");  
  
}  
}  
}
```



```
adapter.UpdateCommand.Parameters.Add("@Mobile", SqlDbType.NVarChar, 50,
"Mobile");

adapter.UpdateCommand.Parameters.Add("@EmployeeID", SqlDbType.Int, 4, "ID");

//Set UpdatedRowSource value as None

//Any Returned parameters or rows are Ignored.

adapter.UpdateCommand.UpdatedRowSource = UpdateRowSource.None;

//Change the Column Values of Few Rows

DataRow Row1 = dataTable.Rows[0];

Row1["Name"] = "Name Changed";

DataRow Row2 = dataTable.Rows[1];

Row2["Email"] = "Email Changed";

DataRow Row3 = dataTable.Rows[2];

Row2["Mobile"] = "Mobile Changed";

// Execute the update.

//The Update method will open the connection, execute the Update command by takking

//the data table data and then close the connection automatically

adapter.Update(dataTable);

Console.WriteLine($"Updated Data Saved into the DataBase");

}

}

catch (Exception ex)

{

Console.WriteLine($"Exception Occurred: {ex.Message}");

}

Console.ReadKey();

}
```

```
}  
  
}
```

This is how connection-oriented architecture works. Now, you can verify the Employee data and you should see the updated data as shown in the below image.

Id	Name	Email	Mobile
100	Name Changed	Anurag@dotnettutorial.net	1234567890
101	Priyanka	Email Changed	Mobile Changed
102	Preety	Preety@dotnettutorial.net	6655443322
103	Sambit	Sambit@dotnettutorial.net	9876543210

Differences between Connected-Oriented Architecture and Disconnected-Oriented Architecture

Let us see the Differences between ADO.NET Connected Oriented Architecture and Disconnected Oriented Architecture.

ADO.NET Connected Oriented Architecture

1. It is connection-oriented data access architecture. It means always an active and open connection is required.
2. Using the DataReader object we can implement the Connected Oriented Architecture.
3. Connected Oriented Architecture gives a faster performance.
4. Connected Oriented Architecture can hold the data of a single table only.
5. You can access the data in a forward-only and read-only manner.
6. Using Data Reader, we cannot persist the data in the database.

ADO.NET Disconnected Oriented Architecture

1. It is disconnection-oriented data access architecture. It means always an active and open connection is not required.
2. Using DataAdapter and DataSet or Datatable we can implement Disconnected Oriented Architecture.
3. Disconnected Oriented Architecture gives a lower performance.

4. Disconnected Oriented Architecture can hold the data of multiple tables using dataset and single table data using Datatable.
5. You can access the data in forward and backward directions and you can also modify the data.
6. Using Data Adapter, we can persist the DataSet or DataTable data into the database.

In the next article, I am going to discuss **How to Load XML data to SQL Server Database Table using ADO.NET DataTable** with Examples. Here, in this article, I try to explain **Connected and Disconnected Architecture in ADO.NET** with Examples. I hope you enjoy this Connected and Disconnected Architecture in ADO.NET article.

Working with Grid Control

GridView control is used to display whole table data on web page. In GridView control each column define a field or title, while each rows represents a record or data.

The GridView control display data with rows and columns wise, we can display whole table in GridView and we also display only required columns from table in GridView control in asp.net.

In GridView control we can easily do sorting, paging and inline editing.

Also we can perform editing and deleting operation on displayed data with GridView control.

GridView Control Syntax

```
<asp:GridView ID="GridView1" runat="server">  
</asp:GridView>
```

GridView Example in ASP.Net C#

Now let's take an asp.net example to bind data in to GridView.

Here, we learn how to bind and display data in GridView control from sql server database in asp.net c#.

GridView Control Example in ASP.Net C#

Step 1 – Open the Visual Studio → Create a new empty Web application.

Step 2 – Create a New web page.

Step 3 – Drag and drop GridView Control on web page from toolbox.

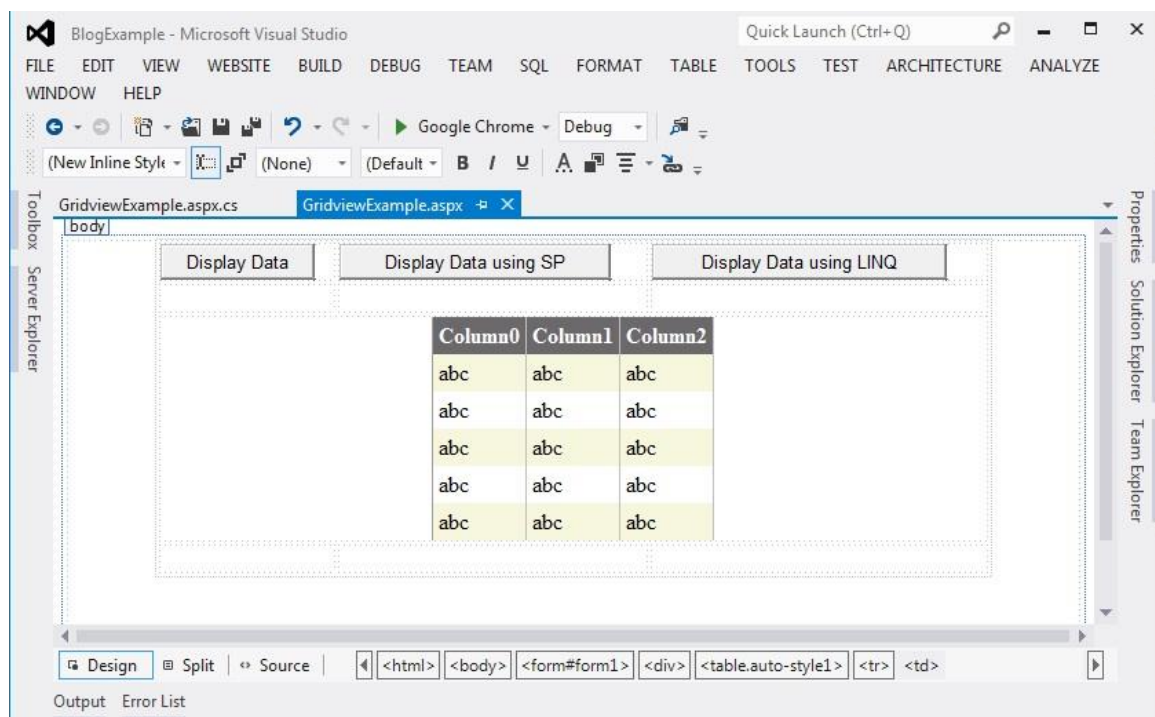
Step 4 – Create New Database in SQL Server

Step 5 – Make connection between Database and web application.

In this asp.net example we will bind data to GridView from SQL server database. So, we need to do connection between our web application and sql server database. There are several method for make connection, we will explain three different connection method in our example.

1. Bind data using SQL Connection and SQL DataAdapter
2. Bind Data using DataSet and Table Adapter
3. Bind Data using LINQ method.

Design asp.net web form with GridView control along with three button as show s in below figure.



ASP.Net GridView Control.

First add namespace on code page.

```
using System.Data.SqlClient;  
using System.Data;
```

Bind data using SQL Connection and SQL DataAdapter

Now, write below code on 'Display Data' button. This connection method we use SqlConnection and SqlDataAdapter object and write sql query in code behind page. This sql connection method are basic connection method.

```
protected void btnview_Click(object sender, EventArgs e)
{
    SqlConnection SQLConn = new SqlConnection("Data
    Source=.\SQLEXPRESS;Initial Catalog='Blog';Integrated Security=True");
    SqlDataAdapter SQLAdapter = new SqlDataAdapter("Select * from UserMst",
    SQLConn);
    DataTable DT = new DataTable();

    SQLAdapter.Fill(DT);

    GridView1.DataSource = DT;
    GridView1.DataBind();
}
```

Bind Data using DataSet and Table Adapter

Write below code on 'Display Data using SP' Button.

In this method we use DataSet and SQL Stored Procedure for sql connection. Use Table Adapter instead of DataAdapter.

```
protected void btnviewdataSP_Click(object sender, EventArgs e)
{
    DS_USER.USERMST_SELECTDataTable UDT = new
    DS_USER.USERMST_SELECTDataTable();
    DS_USERTableAdapters.USERMST_SELECTTableAdapter UAdapter = new
    DS_USERTableAdapters.USERMST_SELECTTableAdapter();
    UDT = UAdapter.SelectData();
    GridView1.DataSource = UDT;
    GridView1.DataBind();
}
```

Bind Data using LINQ method

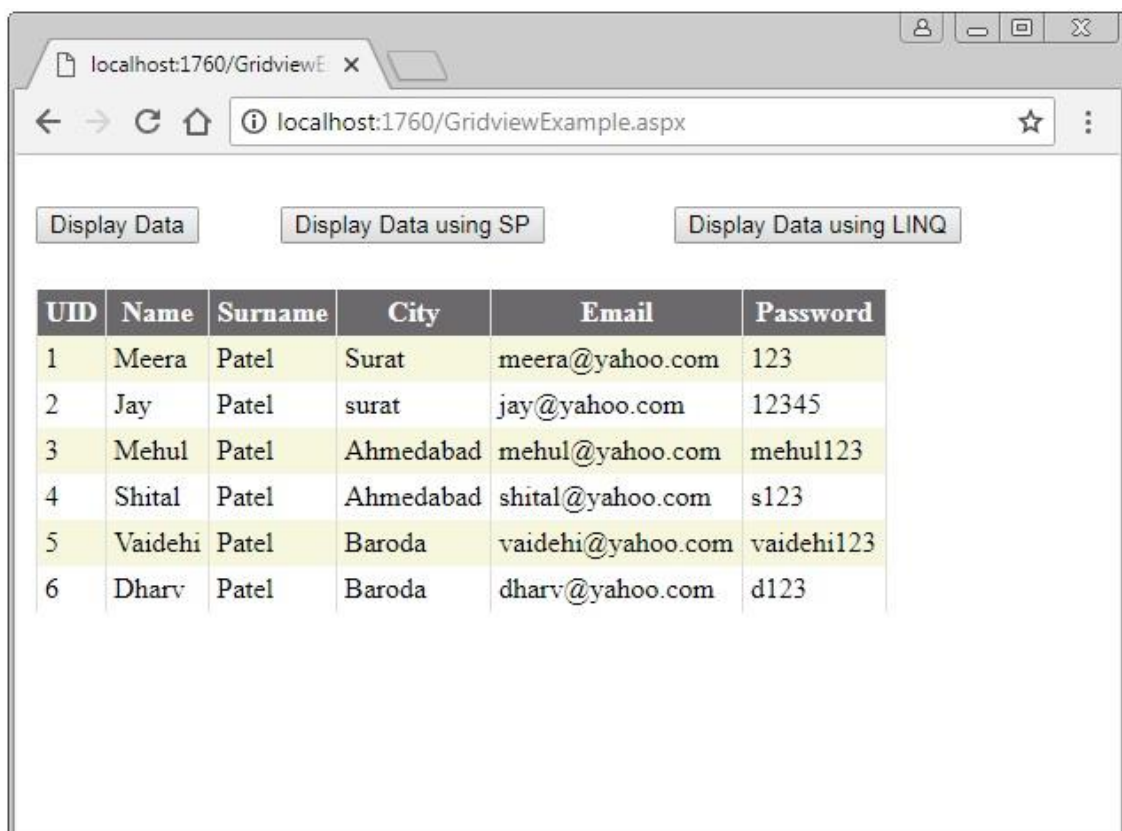
Write below code on 'Display Data using LINQ' button.

Here, we use LINQ method to fetch data from sql server and Display data in to GridView control.

```
protected void btnviewLINQ_Click(object sender, EventArgs e)
{
    DataClassesDataContext Ctx = new DataClassesDataContext();

    GridView1.DataSource = Ctx.USERMST_SELECT();
    GridView1.DataBind();
}
```

Here is the result of above GridView example.



The screenshot shows a web browser window with the address bar displaying 'localhost:1760/GridviewExample.aspx'. The page contains three buttons: 'Display Data', 'Display Data using SP', and 'Display Data using LINQ'. Below the buttons is a GridView control displaying a table of user data. The table has six columns: UID, Name, Surname, City, Email, and Password. The data is as follows:

UID	Name	Surname	City	Email	Password
1	Meera	Patel	Surat	meera@yahoo.com	123
2	Jay	Patel	surat	jay@yahoo.com	12345
3	Mehul	Patel	Ahmedabad	mehul@yahoo.com	mehul123
4	Shital	Patel	Ahmedabad	shital@yahoo.com	s123
5	Vaidehi	Patel	Baroda	vaidehi@yahoo.com	vaidehi123
6	Dharv	Patel	Baroda	dharv@yahoo.com	d123

ASP.Net GridView Control.

Working with Crystal Report Control

Working with the Crystal Reports control in ASP.NET allows you to integrate powerful reporting capabilities into your web applications. Crystal Reports is a widely-used reporting tool that enables you to create, design, and display reports from various data sources. To get started with the Crystal Reports control in ASP.NET, follow these steps:

Step 1: Install Crystal Reports

Before you can work with the Crystal Reports control, ensure that you have Crystal Reports for Visual Studio installed. You can download and install it as an extension to Visual Studio.

Step 2: Create a Crystal Report

In Visual Studio, go to "File" > "New" > "Project."

Choose a web project template (e.g., ASP.NET Web Forms) and click "OK."

Right-click on the project in Solution Explorer and select "Add" > "New Item."

Choose "Crystal Reports" and give your report a name (e.g., "SampleReport.rpt").

The Crystal Reports designer will open. Design your report by adding data sources, fields, headers, footers, and formatting as needed.

Step 3: Add the CrystalReportViewer Control

Open or create a web page where you want to display the Crystal Report.

Drag and drop the **CrystalReportViewer** control from the Toolbox onto your web page.

Step 4: Configure the CrystalReportViewer Control

In the code-behind of your web page, you'll need to configure the **CrystalReportViewer** control to display your report. Here's a basic example:

csharpCopy code

```
using CrystalDecisions.CrystalReports.Engine; using CrystalDecisions.Shared;
public partial class YourPage : System.Web.UI.Page { protected void
Page_Load(object sender, EventArgs e) { if (!IsPostBack) { // Create an instance
of your Crystal Report ReportDocument report = new ReportDocument();
report.Load(Server.MapPath("SampleReport.rpt")); // Path to your report file
// Set the report's data source (e.g., a dataset or datatable) //
report.SetDataSource(yourDataSource); // Set the report document to the
CrystalReportViewer control CrystalReportViewer1.ReportSource = report; } }
```

Ensure that you replace **"SampleReport.rpt"** with the correct path to your Crystal Report file and provide an appropriate data source using **SetDataSource**.

Step 5: Run Your Web Application

Build and run your web application to view the Crystal Report in the **CrystalReportViewer** control.

Note:

Crystal Reports offers extensive features for designing and formatting reports. You can add parameters, subreports, group data, and more to create complex reports.

Ensure that you have the necessary Crystal Reports runtime installed on your web server to display reports properly.

Be cautious with the data sources and connections you use in your reports, as you may need to configure database access and security settings accordingly.

By following these steps, you can integrate Crystal Reports into your ASP.NET web application and display reports with the **CrystalReportViewer** control.