ADO .NET

{ BASICS

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.**
- 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.

- 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.

DataSet

We should consider the following points to use DataSet.

- It caches data locally at our application, so we can manipulate it.
- It interacts with data dynamically such as binding to windows forms control.
- It allows performing processing on data without an open connection. It means it can work while connection is **disconnected.**
- If we required some other functionality mentioned above, we can use **DataReader** to improve performance of our application.
- DataReader does not perform in disconnected mode. It requires DataReader object to be **connected.**

Which one should we use DataReader or DataSet?

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

ADO.NET Framework Data Providers

.NET Framework Data Providers Objects

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

.NET Framework Data Provider for SQL Server

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

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

- using System.Data.SqlClient;

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

.NET Framework Data Provider for SQL Server

. NET Framework Data Provider for Oracle

- It is used to connect with Oracle database through Oracle client. The data provider supports Oracle client software version 8.1.7 or a later version. This data provider supports both local and distributed transactions.

- Oracle Data Provider classes are located into **System.Data.OracleClient** namespace. We must use both **System.Data.OracleClient** and **System.data** to connect our application with the Oracle database.

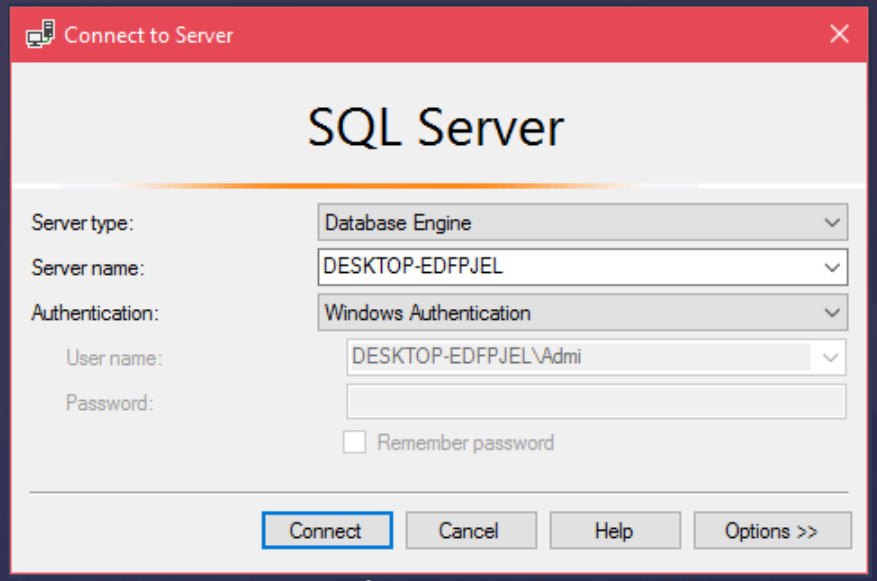
- using System.Data;
- using System.Data.OracleClient;

Data Provider	Note
.NET Framework Data Provider for SQL Server	It is good for middle-tier applications, single-tier applications that use Microsoft SQL Server.
.NET Framework Data Provider for OLE DB	It is good for single-tier applications that use Microsoft Access databases.
.NET Framework Data Provider for ODBC	It is good for middle and single-tier applications that use ODBC data sources.
.NET Framework Data Provider for Oracle	It is good for middle and single-tier applications that use Oracle data sources.

.NET Framework Data Provider is better

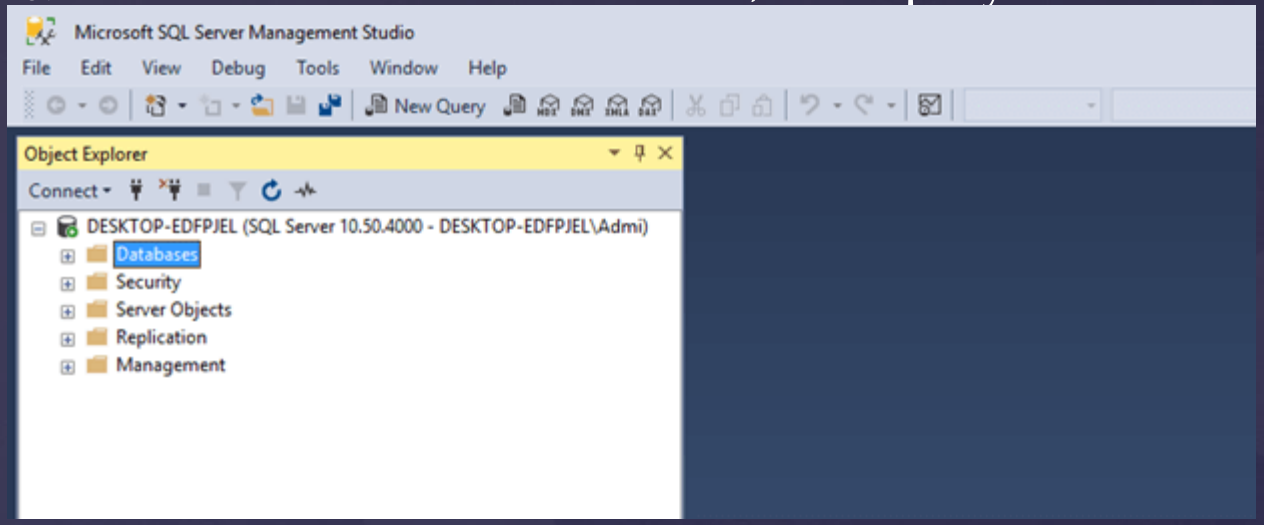
Sql server connectivity

- ADO.NET SQL Server Connection
- To connect with SQL Server, we must have it installed in our system. We are using Microsoft SQL Server Management Tool to connect with the SQL Server. We can use this tool to handle database. Now, follow the following steps to connect with SQL Server.



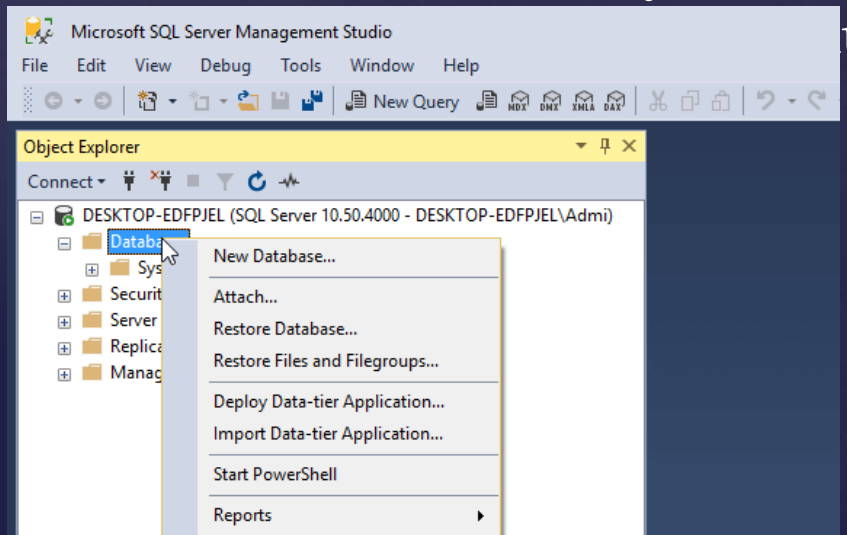
- **Open Microsoft SQL Server Management Tool**
- It will prompt for database connection. Provide the server name and authentication.
- After successful connection, it displays the following window.

- After successful connection, it displays the following window.

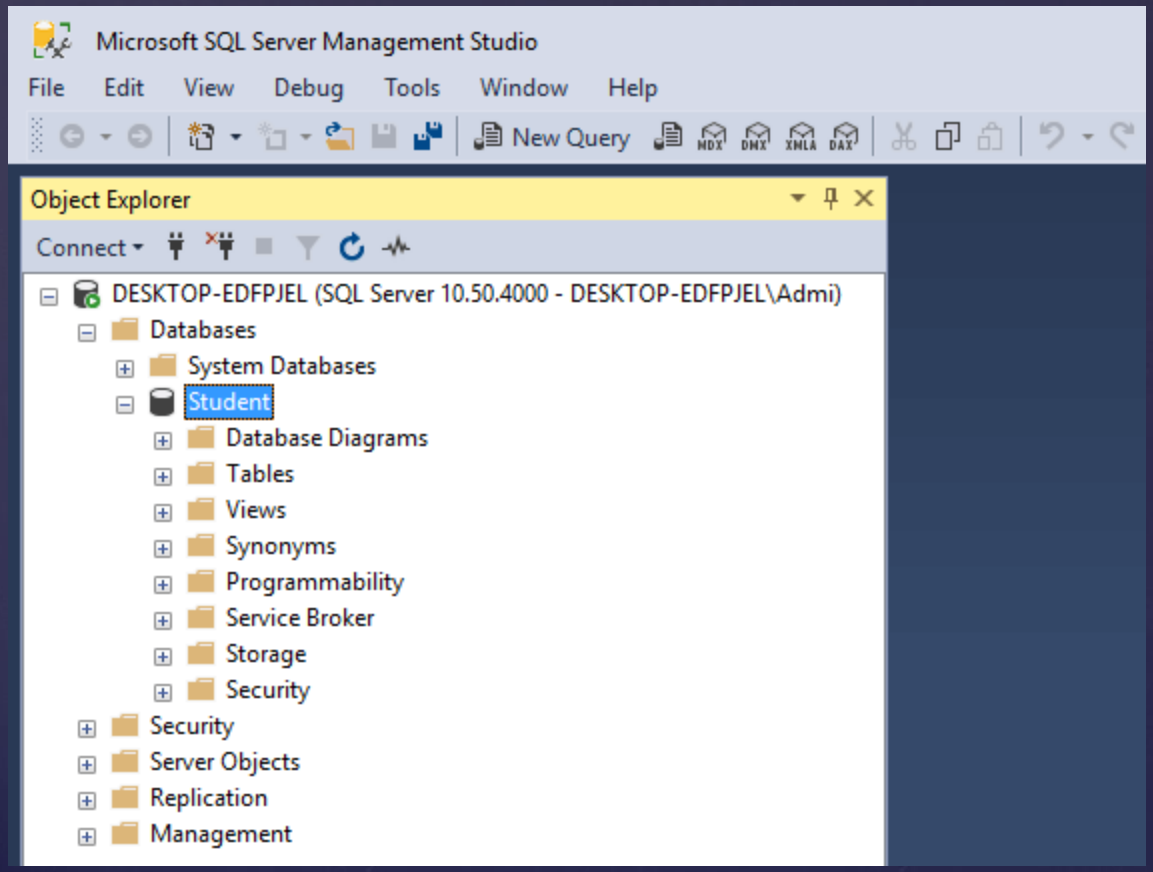


Creating Database

- Now, create database by selecting database option then right click u and provides couple of options.



Click on the Ok button then it will create a database that we can see in the left window of the below screenshot.

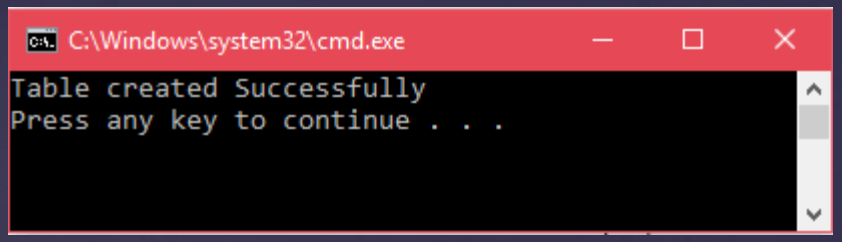


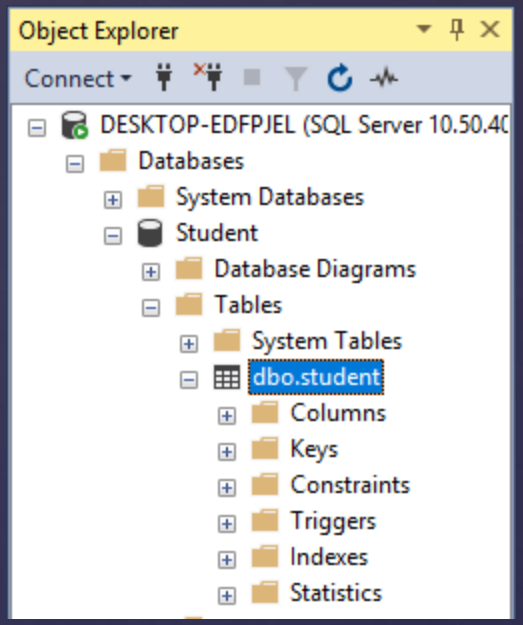
- After creating database, now, let's create a table by using the following C# code. In this source code, we are using created **student** database to connect.
- In visual studio 2017, we created a .NET console application project that contains the following C# code.

Establish connection and create a table

Program.cs

```
using System;
using System.Data.SqlClient;
namespace AdoNetConsoleApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            new Program().CreateTable();
        }
        public void CreateTable()
        {
            SqlConnection con = null;
            try
            {
                // Creating Connection
                 con = new SqlConnection("data source=.; database=student; integrated secu
rity=SSPI");
                // writing sql query
                SqlCommand cm = new SqlCommand("create table student(id int not null,
                name varchar(100), email varchar(50), join_date date)", con);
                // Opening Connection
                con.Open();
                // Executing the SQL query
                cm.ExecuteNonQuery();
                // Displaying a message
                Console.WriteLine("Table created Successfully");
            }
            catch (Exception e)
            {
                Console.WriteLine("OOPs, something went wrong."+e);
            }
            // Closing the connection
            finally
            {
                con.Close();
            }
        }
    }
}
```





Will be created

Run Program.cs

Insert data

```
using System;
using System.Data.SqlClient;
namespace AdoNetConsoleApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            new Program().CreateTable();
        }
        public void CreateTable()
        {
            SqlConnection con = null;
            try
            {
                // Creating Connection
                con = new SqlConnection("data source=.; database=student;
 integrated security=SSPI");
                // writing sql query
                SqlCommand cm = new SqlCommand("insert into student
                (id, name, email, join_date)values('101','Ronald Trump','ronal
d@example.com','1/12/2017')", con);
                // Opening Connection
                con.Open();
                // Executing the SQL query
                cm.ExecuteNonQuery();
                // Displaying a message
                Console.WriteLine("Record Inserted Successfully");
            }
            catch (Exception e)
            {
                Console.WriteLine("OOPs, something went wrong."+e);
            }
            // Closing the connection
            finally
            {
                con.Close();
            }
        }
    }
}
```

Retrieve data

```
using System;
using System.Data.SqlClient;
namespace AdoNetConsoleApplication
{
   class Program
   {
      static void Main(string[] args)
      {
         new Program().CreateTable();
      }
      public void CreateTable()
      {
         SqlConnection con = null;
         try
         {
            // Creating Connection
            con = new SqlConnection("data source=.; database=student; integrated security=SSPI");
            // writing sql query
            SqlCommand cm = new SqlCommand("Select * from student", con);
            // Opening Connection
            con.Open();
            // Executing the SQL query
            SqlDataReader sdr = cm.ExecuteReader();
            // Iterating Data
            while (sdr.Read())
            {
               Console.WriteLine(sdr["id"] + " " + sdr["name"]+" "+sdr["email"]); // Displaying Record
            }
         }
         catch (Exception e)
         {
            Console.WriteLine("OOPs, something went wrong.\n"+e);
         }
         // Closing the connection
         finally
         {
            con.Close();
         }
      }
   }
}
```

Delete data

```
using System;
using System.Data.SqlClient;
namespace AdoNetConsoleApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            new Program().CreateTable();
        }
        public void CreateTable()
        {
            SqlConnection con = null;
            try
            {
                // Creating Connection
                con = new SqlConnection("data source=.; database=student; integrated security=SSPI
");
                // writing sql query
                SqlCommand cm = new SqlCommand("delete from student where id = '101'", con);
                // Opening Connection
                con.Open();
                // Executing the SQL query
                cm.ExecuteNonQuery();
                Console.WriteLine("Record Deleted Successfully");
            }
            catch (Exception e)
            {
                Console.WriteLine("OOPs, something went wrong.\n"+e);
            }
            // Closing the connection
            finally
            {
                con.Close();
            }
        }
    }
}
```