# .NET Technologies (01CE1602)

6th Semester

4 Credits

Prof. Ravikumar R N

Dept. of Computer Engineering

# Objective

- Net Technologies are blend of technologies supported by Microsoft .Net Framework that allows user to create various applications.
- Students will be able to work with various technologies provided by Microsoft .NET platform.

# Course Outcomes

- To Review the components of .Net Framework
- To practice Web based application
- To create web applications using MVC framework
- **To practice basic database application using ADO.net**
- To designing, developing, and deploying APIs

# Unit 4
## Working with ADO.Net

# Contents

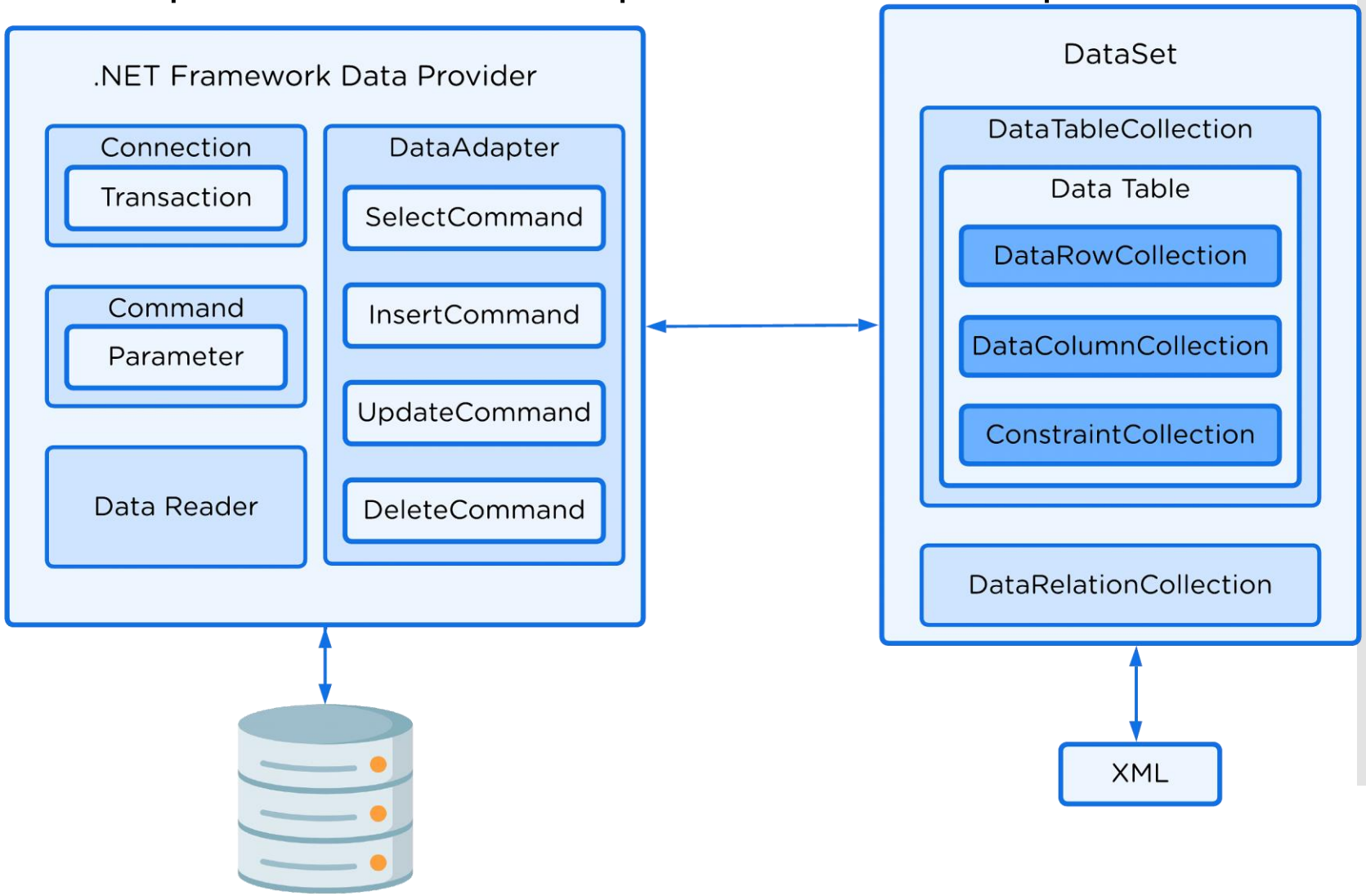
- ADO.Net Architecture
- Characteristics of ADO.Net
- Data Namespaces
- ADO.Net Object Model
- DataSet
- DataTable
- DataRelation
- Connection object
- Command Object
- Data Reader
- Object
- DataAdapter Object
- Data Controls (Repeater, DataList, DataGrid)
- Binding data with Crystal Report
- Performing CRUD operations

# What is ADO.NET?

- ADO.NET is a set of classes (a framework) to **interact with data sources** such as databases and XML files. ADO is the acronym for ActiveX Data Object. It is created by **Microsoft** as part of its .NET framework.
- It allows us to connect to underlying data or databases. It has classes and methods to retrieve and manipulate data.
- The following are a few of the .NET applications that use ADO.NET to connect to a database, execute commands and retrieve data from the database.
  - ASP.NET Web Applications.
  - Console Applications.
  - Windows Applications.
- All the ADO.NET classes are located into System.Data.dll and integrated with XML classes located into System.Xml.dll.

# ADO.NET Architecture / Object Model

- 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.



# Classes

| Class | Description |
| --- | --- |
| **SqlConnection** | It is used to create SQL Server connection. This class cannot be inherited. |
| **SqlCommand** | It is used to execute database queries. This class cannot be inherited. (ExecuteNonQuery) |
| **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. |
| **DataSet** | The ADO.NET DataReader object is one of the two techniques provided by ADO.NET. This object is utilized to obtain data from the data store. |

# ADO.NET Data Namespaces

- Data Provider is used to connect with the database and then retrieve data as per command execution. It's like a lightweight component with multiple roles.
- Some of the popular data providers and frameworks are listed below:
- ADO.NET Data Providers:
  - OleDb (System.Data.OleDb) - Object Linking and Embedding, Database
  - **SqlClient (System.Data.SqlClient)**
  - Odbc (System.Data.Odbc)
  - OracleClient (System.Data.OracleClient)
- ADO.NET Entity Framework:
  - LINQ to Entities
  - EntityClient (System.Data.EntityClient)
  - Typed ObjectQuery

# Various Connection Architectures

- There are the following two types of connection architectures:
- Connected architecture: the application remains connected with the database throughout the processing. **(DataReader)**
- Disconnected architecture: the application automatically connects/disconnects during the processing. The application uses temporary data on the application side called a **DataSet**.

# ADO.NET Features

- Maintainability
- The multilayered architecture enables building application logic in a separate layer – it mitigates the risk and simplifies adding new features.
- Interoperability
- Data is transferred to and from data sources and internal data representations in an XML format.
- Programmability
- Strongly typed data and IntelliSense support in Microsoft Visual Studio simplifies writing statements.
- Scalability
- High application scalability is possible thanks to a disconnected mode that enables users to work with the data without retaining an open connection. It conserves resources and allows multiple users to access the data simultaneously.
- Performance
- Both connected and disconnected classes are optimized for performance, and the proper use of those classes enables maintaining application performance at a high level.

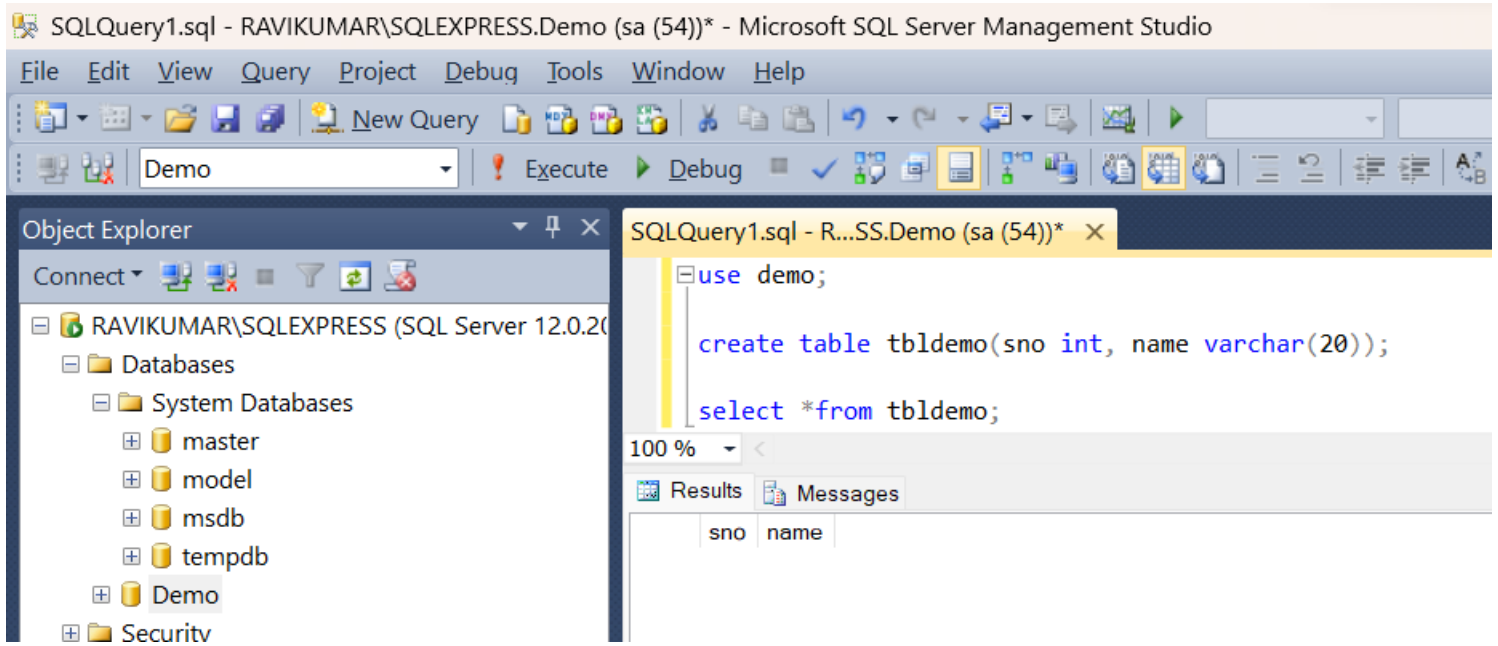
# ADO.NET SqlConnection Class

- It is used to establish an open connection to the SQL Server database.
- It is a sealed class so that cannot be inherited.
- SqlConnection class uses SqlDataAdapter and SqlCommand classes together to increase performance when connecting to a Microsoft SQL Server database.
- Connection does not close explicitly even it goes out of scope. Therefore, you must explicitly close the connection by calling Close() method.

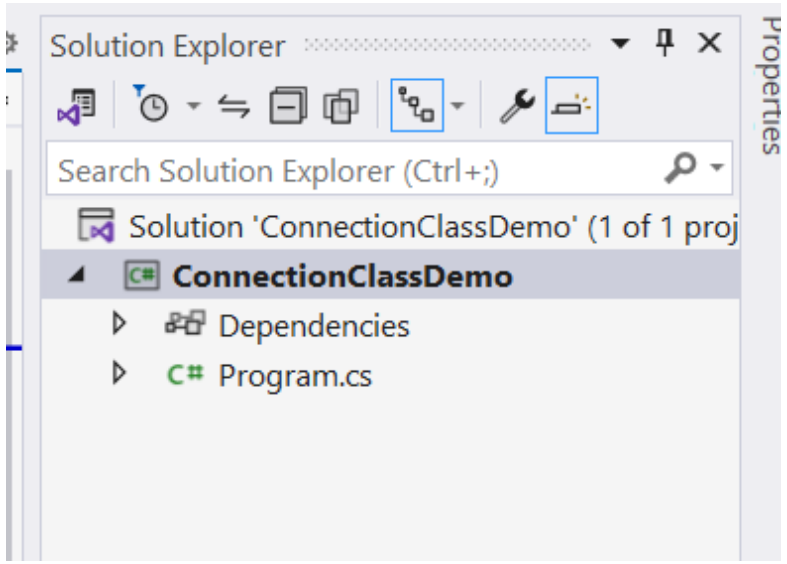
# ADO.NET SqlConnection Class Methods

| Method | Description |
|---|---|
| BeginTransaction() | It is used to start a database transaction. |
| ChangeDatabase(String) | It is used to change the current database for an open SqlConnection. |
| ChangePassword(String, String) | It changes the SQL Server password for the user indicated in the connection string. |
| Close() | It is used to close the connection to the database. |
| CreateCommand() | It enlists in the specified transaction as a distributed transaction. |
| GetSchema() | It returns schema information for the data source of this SqlConnection. |
| Open() | It is used to open a database connection. |
| ResetStatistics() | It resets all values if statistics gathering is enabled. |

# Console app Demo



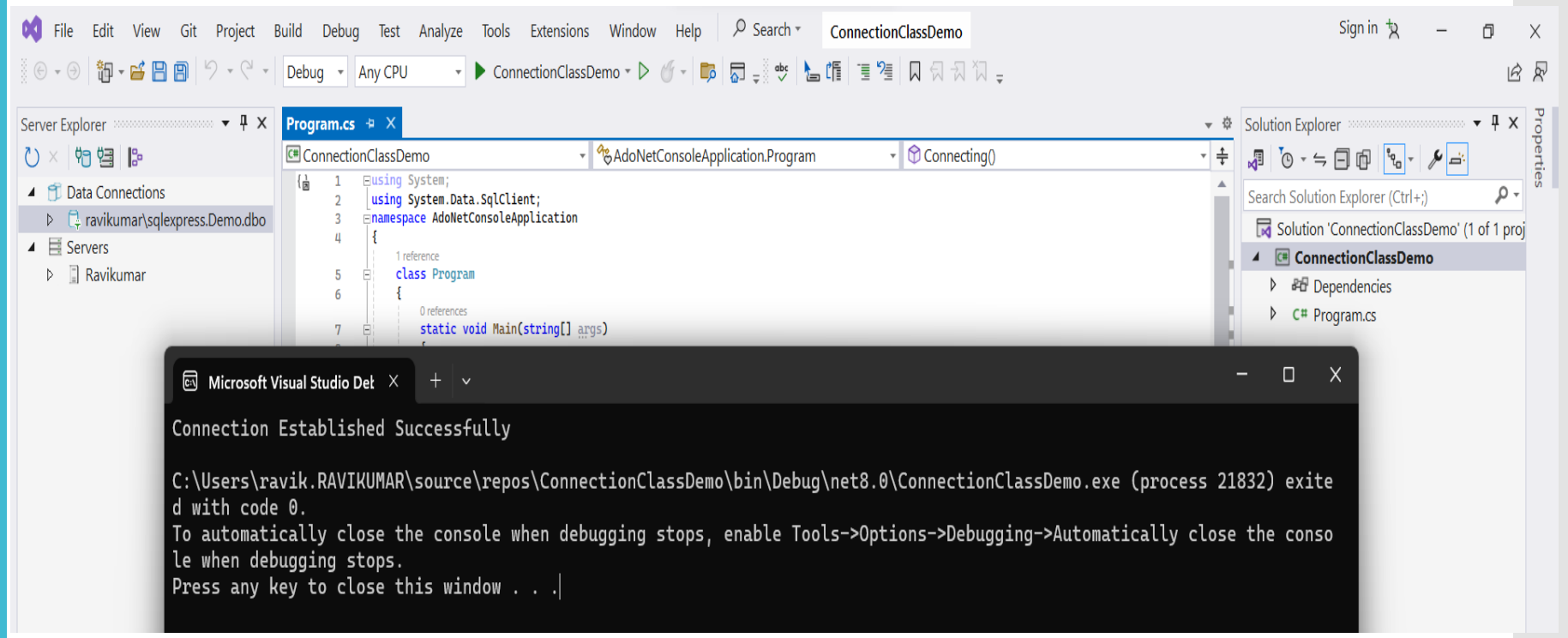

- Console App



Console app Demo using Block

```
using System;
using System.Data.SqlClient;
namespace AdoNetConsoleApplication{
    class Program{
        static void Main(string[] args)
        {
            new Program().Connecting();
        }
        public void Connecting(){
            using (
                    // Creating Connection
                    SqlConnection con = new SqlConnection("data source=RAVIKUMAR\\SQLEXPRESS; database=Demo; integrated security=SSPI")
                )
            {
                con.Open();
                Console.WriteLine("Connection Established Successfully");}}}}
```

# Console app Demo



# Console App using try catch

```
using System;
using System.Data.SqlClient;
namespace AdoNetConsoleApplication {
   class Program {
      static void Main(string[] args)  {
         new Program().Connecting();
      }
      public void Connecting()  {
         SqlConnection con = null;
         try {
            // Creating Connection
            con = new SqlConnection("data source=.; database=student; integrated security=SSPI");
            con.Open();
            Console.WriteLine("Connection Established Successfully");
         }
         catch (Exception e) {  Console.WriteLine("OOPs, something went wrong.\n"+e);    }
         finally {  con.Close();          }       }    } }
```

# ADO.NET SqlCommand Class

- This class is used to store and execute SQL statement for SQL Server database. It is a sealed class so that cannot be inherited.

| Constructor | Description |
|---|---|
| SqlCommand() | It is used to initialize a new instance of the SqlCommand class. |
| SqlCommand(String) | It is used to initialize a new instance of the SqlCommand class with a string parameter. |
| SqlCommand(String, SqlConnection) | It is used to initialize a new instance of the SqlCommand class. It takes two parameters, first is query string and second is connection string. |
| SqlCommand(String, SqlConnection, SqlTransaction) | It is used to initialize a new instance of the SqlCommand class. It takes three parameters query, connection and transaction string respectively. |
| SqlCommand(String, SqlConnection, SqlTransaction, SqlCommandColumnEncryptionSetting) | It Initializes a new instance of the SqlCommand class with specified command text, connection, transaction, and encryption setting. |

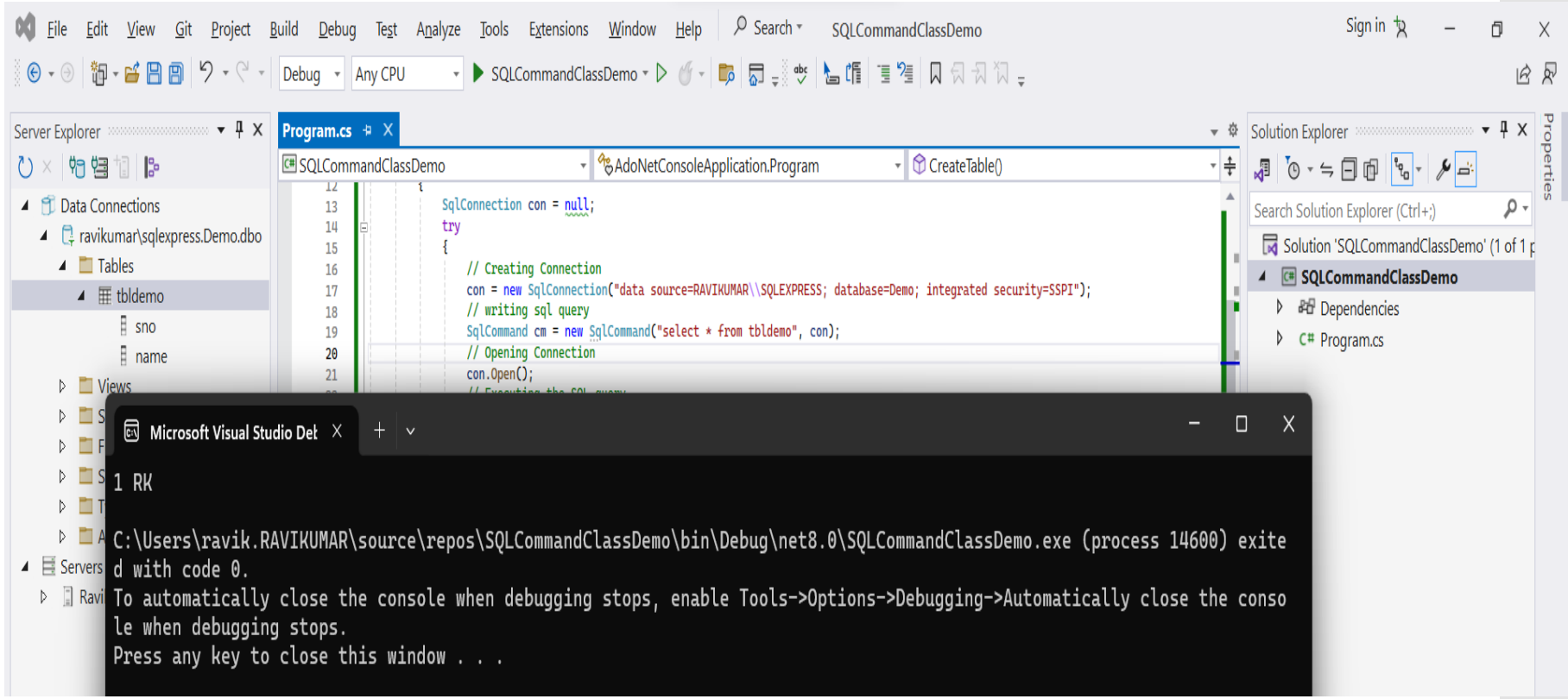
# ADO.NET SqlCommand Class

| Method | Description |
|---|---|
| BeginExecuteNonQuery() | It is used to Initiate the asynchronous execution of the SQL statement described by this SqlCommand. |
| Cancel() | It tries to cancel the execution of a SqlCommand. |
| Clone() | It creates a new SqlCommand object that is a copy of the current instance. |
| CreateParameter() | It creates a new instance of a SqlParameter object. |
| ExecuteReader() | It is used to send the CommandText to the Connection and builds a SqlDataReader. |
| ExecuteXmlReader() | It is used to send the CommandText to the Connection and builds an XmlReader object. |
| ExecuteScalar() | It executes the query and returns the first column of the first row in the result set. Additional columns or rows are ignored. |
| Prepare() | It is used to create a prepared version of the command by using the instance of SQL Server. |
| ResetCommandTimeout() | It is used to reset the CommandTimeout property to its default value. |

# ADO.NET SqlCommand Class and SqlDataReader Class

```
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{

con = new SqlConnection("data source=RAVIKUMAR\\SQLEXPRESS;
database=Demo; integrated security=SSPI");
SqlCommand cm = new SqlCommand("select * from tbldemo", con);
con.Open();
SqlDataReader sdr = cm.ExecuteReader();
while (sdr.Read())
{
  Console.WriteLine(sdr["sno"] + " " + sdr["name"]);
                }
            }
            catch (Exception e){
            Console.WriteLine("OOPs, something went wrong." + e);
            }
finally{  con.Close();   } } } }
```

# ADO.NET SqlCommand Class and SqlDataReader Class

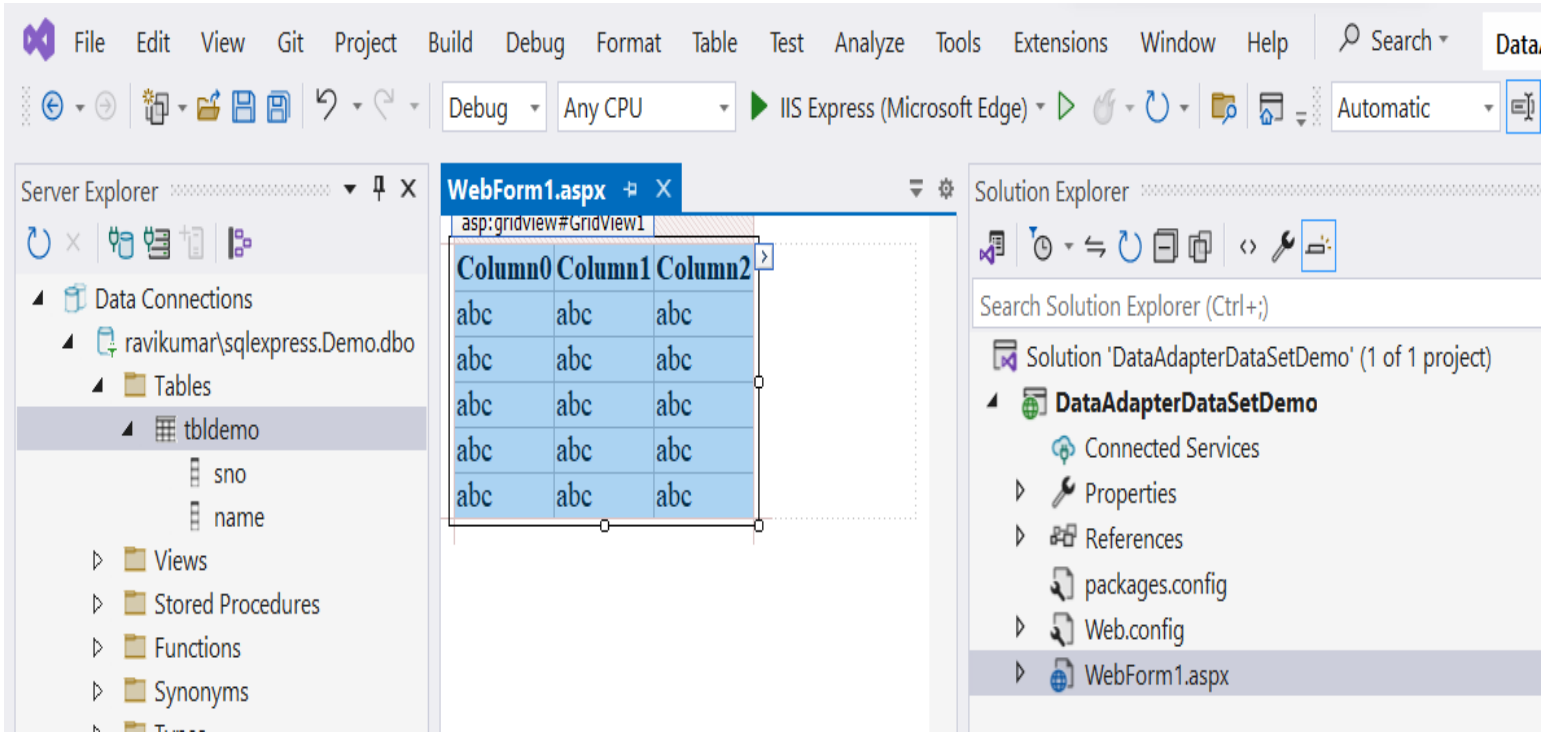


# DataSet and DataAdapter

- **DataSet 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.

- **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.

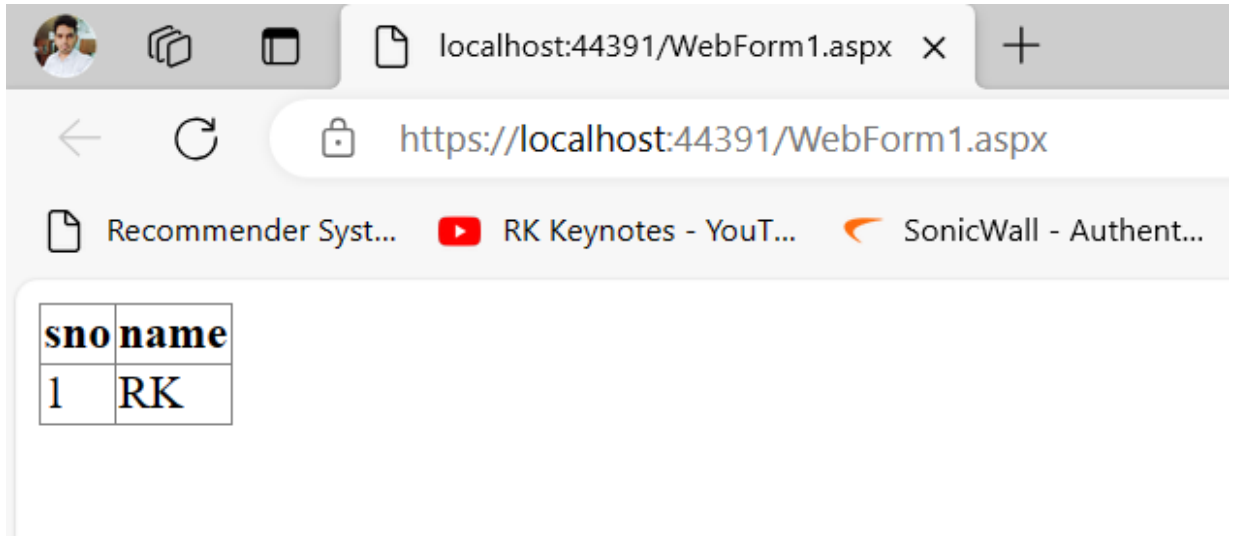
# DataSet and DataAdapter



# DataSet and DataAdapter

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

namespace DataAdapterDataSetDemo

{

    public partial class WebForm1 : System.Web.UI.Page

    {

        protected void Page_Load(object sender, EventArgs e)

        {

            using (SqlConnection con = new SqlConnection("data source=RAVIKUMAR\\SQLEXPRESS; database=Demo; integrated security=SSPI"))

            {

                SqlDataAdapter sde = new SqlDataAdapter("Select * from tbldemo", con);

                DataSet ds = new DataSet();

                sde.Fill(ds);

                GridView1.DataSource = ds;

                GridView1.DataBind(); }}}}
```

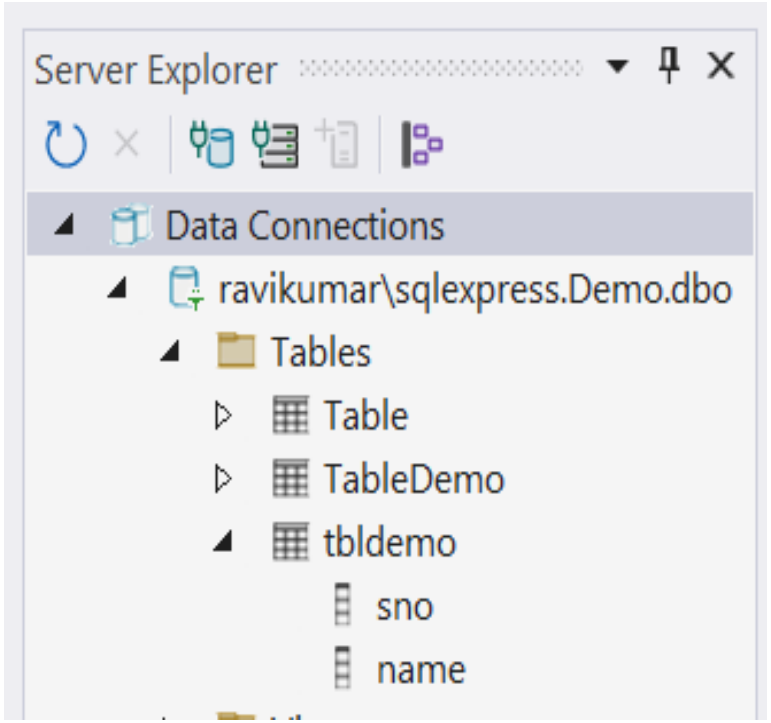
# DataSet and DataAdapter

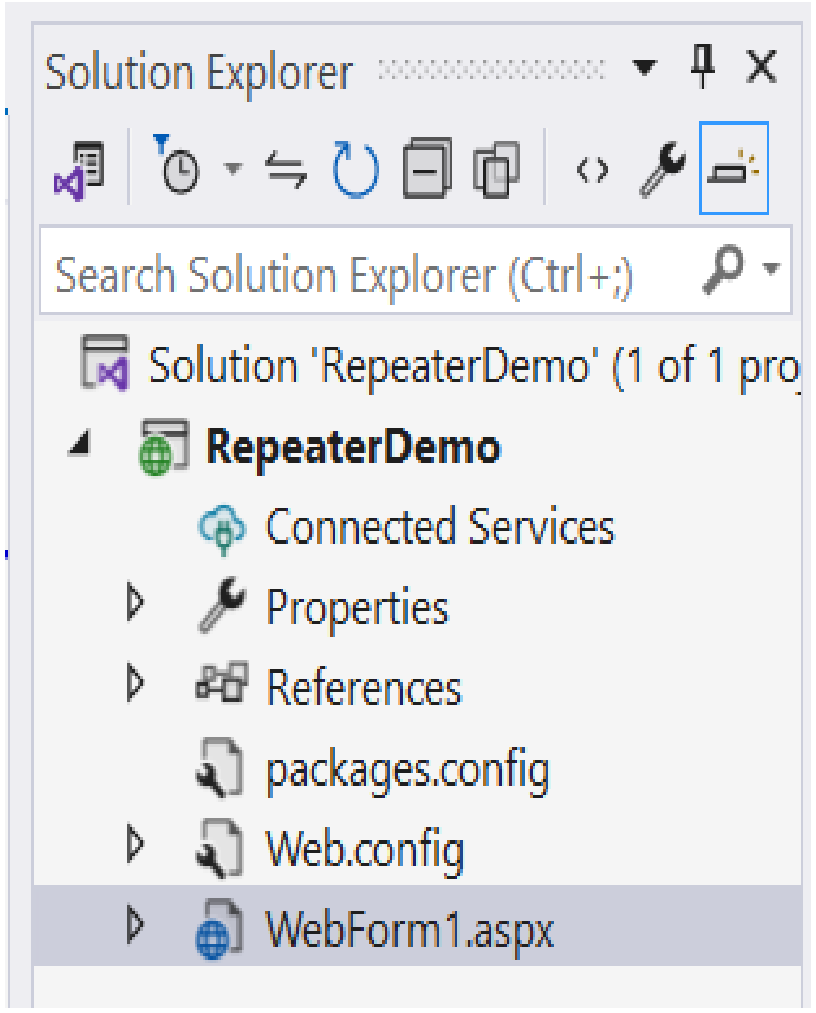


# Repeater

- The main use of Repeater Control is for **displaying a repeated list of items** bound to the control.
- A Repeater Control is faster and lightweight for displaying data compared to a GridView or DataGrid.
- With the Repeater control we can display data in a **custom format**.
- The main drawback of a Repeater Control is that it doesn't support paging and sorting.
- The Repeater Control has the following types of template fields:
- **Item Template – (describing how each item will be rendered from the data source collection.)**
- Alternating Item Template – (color and style)
- Header Template – (display header element)
- Footer Template – (display footer)
- Separator Template – (separate each item)

# Repeater


Server Explorer
Data Connections
ravikumar\sqlexpress.Demo.dbo
Tables
Table
TableDemo
tbldemo
sno
name


Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'RepeaterDemo' (1 of 1 pro
RepeaterDemo
Connected Services
Properties
References
packages.config
Web.config
WebForm1.aspx

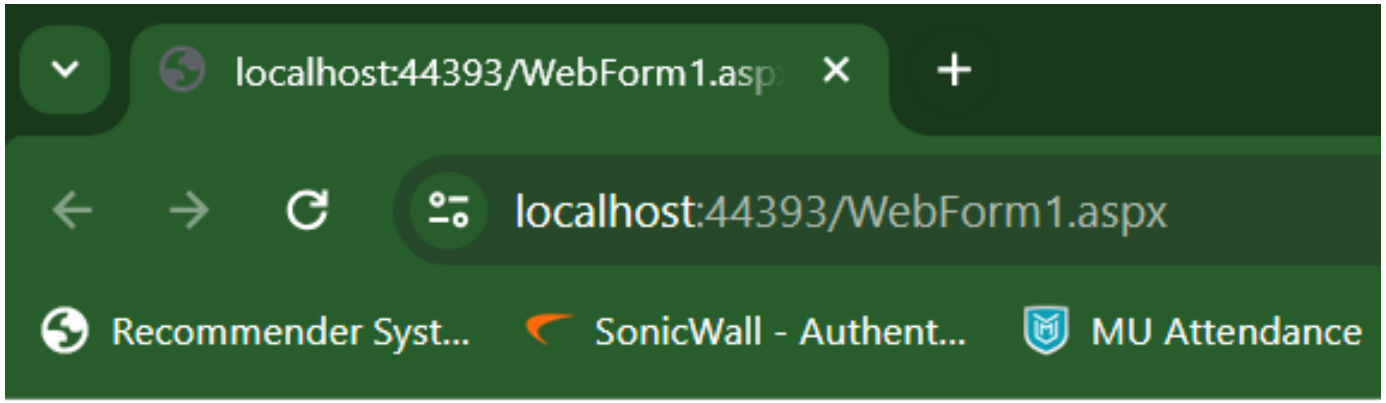
# Repeater

```
//WebForm1.aspx
<body>
    <form id="form1" runat="server">
        <asp:Repeater ID="Repeater1" runat="server">
            <ItemTemplate>
                <div>
                    <table>
                        <tr>
                            <td>SNO:</td>
                            <th>Student<%#Eval("sno")%></th>
                        </tr>
                        <tr>
                            <td>Name:</td>
                            <td><%#Eval("name")%></td>
                        </tr>

                    </table>
                </div>
            </ItemTemplate>
        </asp:Repeater>

    </form>
</body>
```

# Repeater

```
//WebForm1.aspx.cs

protected void Page_Load(object sender, EventArgs e)
        {
            SqlConnection con = new SqlConnection("Data
Source=RAVIKUMAR\\SQLEXPRESS;Database=Demo;Integrated
Security=True");

     SqlDataAdapter sda = new SqlDataAdapter("select * from
tbldemo", con);
            DataTable dt = new DataTable();
            sda.Fill(dt);
            Repeater1.DataSource = dt;
            Repeater1.DataBind();
        }
```

# Repeater


localhost:44393/WebForm1.asp
localhost:44393/WebForm1.aspx
Recommender Syst...
SonicWall - Authent...
MU Attendance
SNO: Student1
Name: RK
SNO: Student2
Name: RKK

# DataList

- The ASP.NET DataList control is a light weight server side control that works as a container for data items. It is used to display data into a list format to the web pages.
- It displays the data from the data source and rows in various layouts, like arranging them in either rows or columns (arrange data items horizontally or vertically using property RepeatDirection).
- **Performance is slow as compared to Repeater** and faster when compared to GridView.

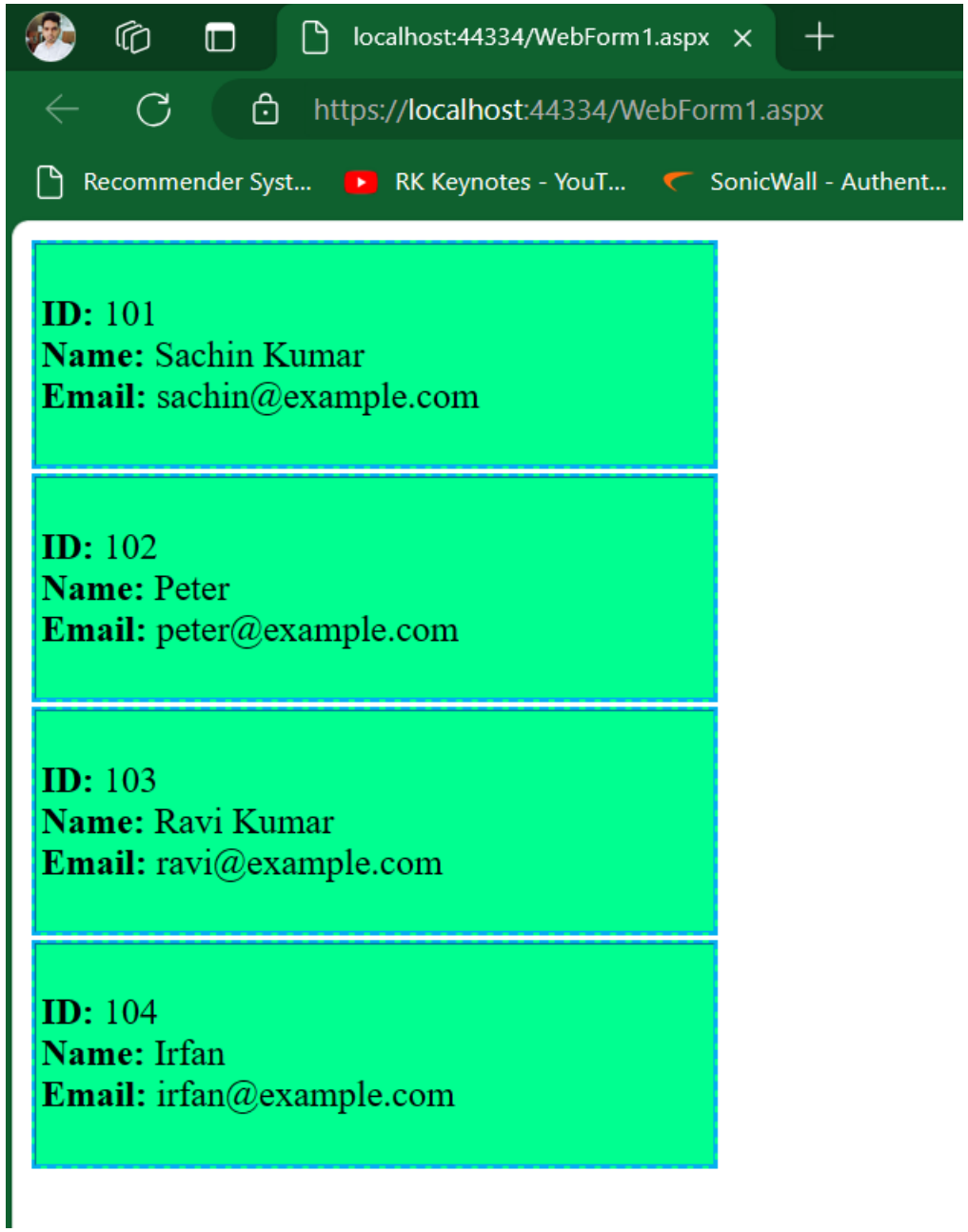
# DataList

```
<body>
    <form id="form1" runat="server">
        <div>
            <asp:DataList ID="DataList1" runat="server">
                <ItemTemplate>
                <table cellpadding="2" cellspacing="0" border="1" style="width: 300px; height: 100px;
                border: dashed 2px #04AFEF; background-color: #00ff90">
                    <tr>
                            <td>
                                <b>ID: </b><span class="id"><%# Eval("ID") %></span><br />
                                <b>Name: </b><span class="name"><%# Eval("Name") %></span><br />
                                <b>Email: </b><span class="email"><%# Eval("Email")%></span><br />
                            </td>
                    </tr>
                </table>
            </ItemTemplate>
            </asp:DataList>
        </div>
    </form>
</body>
```

# DataList

```
protected void Page_Load(object sender, EventArgs e)
    {
        DataTable table = new DataTable();
        table.Columns.Add("ID");
        table.Columns.Add("Name");
        table.Columns.Add("Email");
        table.Rows.Add("101", "Sachin Kumar", "sachin@example.com");
        table.Rows.Add("102", "Peter", "peter@example.com");
        table.Rows.Add("103", "Ravi Kumar", "ravi@example.com");
        table.Rows.Add("104", "Irfan", "irfan@example.com");
        DataList1.DataSource = table;
        DataList1.DataBind();

    }
}
```

# DataList


localhost:44334/WebForm1.aspx
https://localhost:44334/WebForm1.aspx
Recommender Syst...
RK Keynotes - YouT...
SonicWall - Authent...
ID: 101
Name: Sachin Kumar
Email: sachin@example.com
ID: 102
Name: Peter
Email: peter@example.com
ID: 103
Name: Ravi Kumar
Email: ravi@example.com
ID: 104
Name: Irfan
Email: irfan@example.com

# DataGrid

- .NET Framework provides DataGrid control to display data on the web page. It was introduced in .NET 1.0 and now has been deprecated.
- DataGrid is used to display data in scrollable grid. It requires data source to populate data in the grid.
- It is a server side control and can be dragged from the toolbox to the web form.
- Data Source for the DataGrid can be either a DataTable or a database.

# DataGrid

- .NET Framework provides DataGrid control to display data on the web page. It was introduced in .NET 1.0 and now has been deprecated.
- DataGrid is used to display data in scrollable grid. It requires data source to populate data in the grid.
- It is a server side control and can be dragged from the toolbox to the web form.
- Data Source for the DataGrid can be either a DataTable or a database.
- **Already covered in slide number 24.**

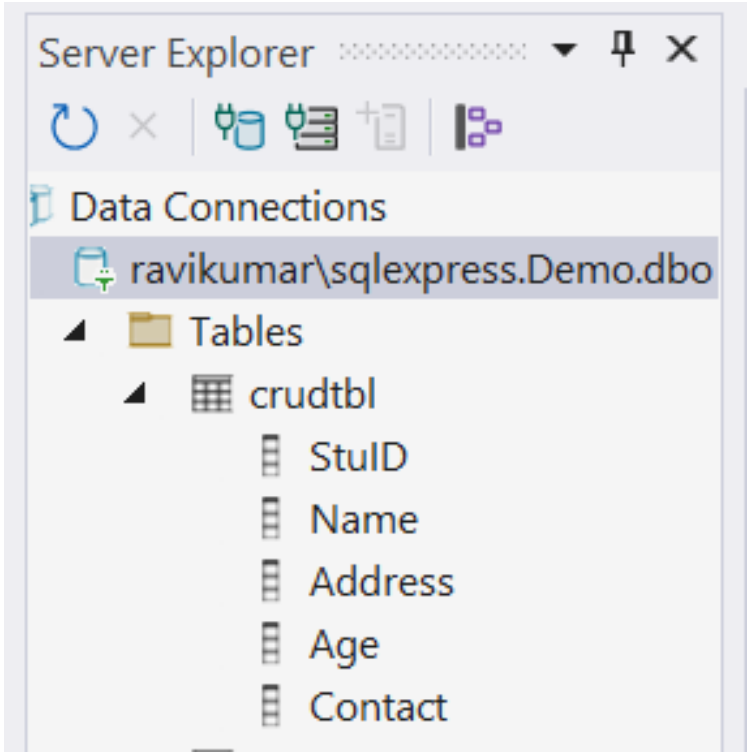
# Repeater vs DataList vs GridView

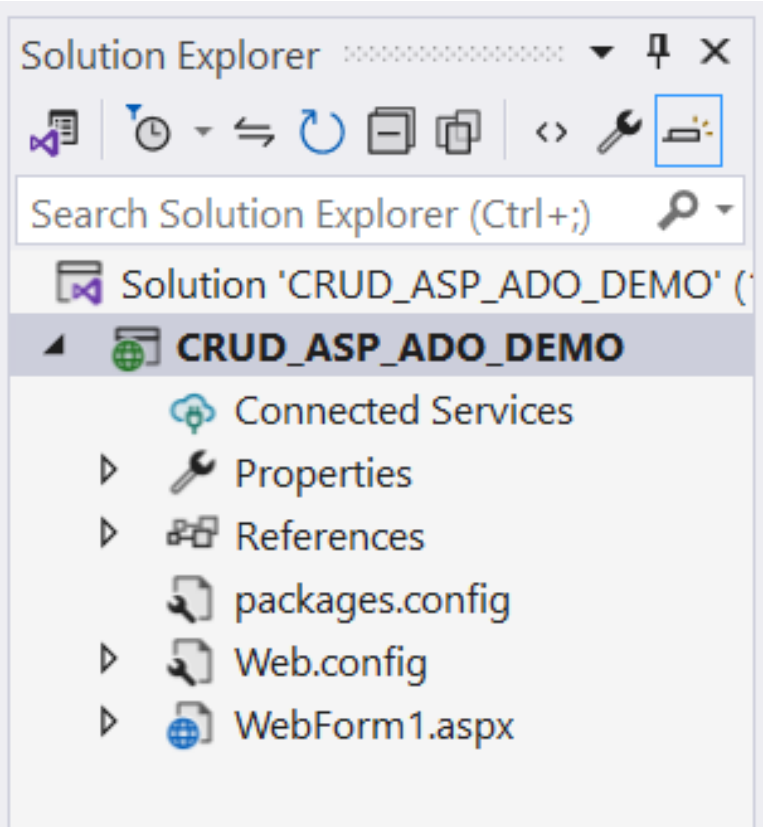
| Repeater | DataList | GridView |
|---|---|---|
| Template driven. | Rendered as Table. | Introduced with Asp.Net 2.0. |
| This feature is not supported (automatic column generation). | Automatically generates columns from the data source. | Built-in Paging and Sorting is provided. |
| Selection of row is not supported. | Selection of row is supported. | Built-in supports for Update and Delete operations. |
| Editing of contents is not supported. | Editing of contents is supported. | Supports auto format or style features. |
| This feature is not supported (arrange data items with RepeatDirection). | You can arrange data items horizontally or vertically in DataList by using property RepeatDirection. | RepeatDirection property is not supported. |
| This is very lightweight and fast data control among all the data control. | Performance is slow as compared to Repeater | Doesn't support customizable row separator. Performance is slow as compared to DataList. |

| S.No. | ADO | ADO.NET |
|---|---|---|
| 1. | It is based on COM (Component Object Modelling). | It is a CLR (Common Language Runtime) based library. |
| 2. | It works only when data store is connected. | It does not needs active connection to access data from data store. |
| 3. | It has feature of locking. | It does not have feature of locking. |
| 4. | It access and store data from data source by recordset object. | It access and store data from data source by dataset object. |
| 5. | XML integration is not feasible in ADO. | XML integration is feasible in ADO.NET. |
| 6. | In ADO, data is stored in binary form. | While in this, data is stored in XML. |
| 7. | It allow us to create client side cursors only. | It give us the choice of using weather client side and server side cursors. |
| 8. | It requires SQL JOINs and UNIONs to combine data from multiple tables in a single result table. | It uses DataRelational objects, for combining data from multiple tables without requiring JOINs and UNIONs. |
| 9. | It supports sequential access of rows in a RecordSet. | It allows completely non-sequential data access in DataSet through collection based hierarchy. |

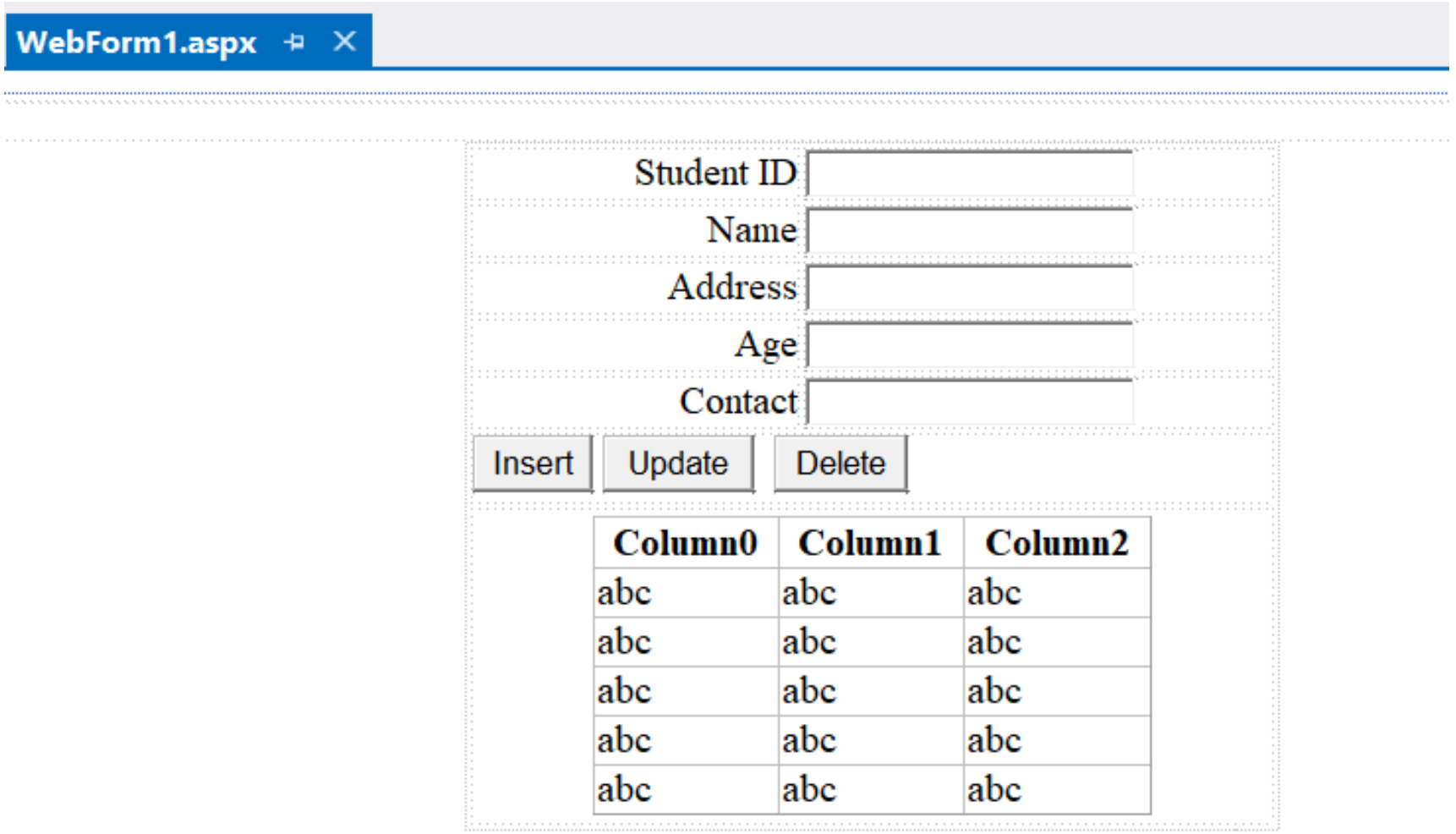
# Crystal Report

# CRUD Operation with ADO.NET



# CRUD Operation with ADO.NET



# CRUD Operation with ADO.NET

```
using System; using System.Web.UI; using System.Data;  using System.Data.SqlClient;

namespace CRUD_ASP_ADO_DEMO{

    public partial class WebForm1 : System.Web.UI.Page{

        protected void Page_Load(object sender, EventArgs e){

            if (!IsPostBack)

            {

                LoadRecord();

            }  }

        SqlConnection con = new SqlConnection("Data Source = ravikumar\\sqlexpress;
Initial Catalog = Demo; User ID = sa; Password=123;Integrated Security=True");

        protected void Button1_Click(object sender, EventArgs e)

        {

            con.Open();

            SqlCommand comm = new SqlCommand("Insert into crudtbl values('" +
TextBox1.Text + "','" + TextBox2.Text + "','" + TextBox3.Text + "','" + TextBox4.Text +
"','" + TextBox5.Text + "')", con);

            comm.ExecuteNonQuery();

            con.Close();

            ScriptManager.RegisterStartupScript(this, this.GetType(), "script",
"alert('Successfully Inserted');", true);

            LoadRecord();     ClearData();         }
```

CRUD Operation with ADO.NET

```
void LoadRecord()
        {
            SqlCommand comm = new SqlCommand("select * from crudtbl", con);
            SqlDataAdapter d = new SqlDataAdapter(comm);
            DataTable dt = new DataTable();
            d.Fill(dt);
            GridView1.DataSource = dt;
            GridView1.DataBind();
        }
        void ClearData()
        {
            TextBox1.Text = "";
            TextBox2.Text = "";
            TextBox3.Text = "";
            TextBox4.Text = "";
           TextBox5.Text = "";        }
```

# CRUD Operation with ADO.NET

```
protected void Button2_Click(object sender, EventArgs e)
    {
        con.Open();
        SqlCommand comm = new SqlCommand("update crudtbl set Name
'" + TextBox2.Text + "', Address= '"+TextBox3.Text+"',
Age='"+TextBox4.Text+"', Contact='"+TextBox5.Text+"' where StuID=
'"+TextBox1.Text+"' ", con);
        comm.ExecuteNonQuery();
        con.Close();
        ScriptManager.RegisterStartupScript(this, this.GetType(),
"script", "alert('Successfully Updated');", true);
        LoadRecord();
        ClearData();
    }
```

# CRUD Operation with ADO.NET

```
 protected void Button3_Click(object sender, EventArgs e)
        {
            con.Open();
            SqlCommand comm = new SqlCommand("delete crudtbl where StuID =
+ TextBox1.Text + "' "", con);
            comm.ExecuteNonQuery();
            con.Close();
            ScriptManager.RegisterStartupScript(this, this.GetType(),
"script", "alert('Successfully Deleted');", true);
            LoadRecord();
            ClearData();

        }
    }
}
```

# CRUD Operation with ADO.NET

- Demo

# Summary

- ADO.Net Architecture
- Characteristics of ADO.Net
- Data Namespaces
- ADO.Net Object Model
- DataSet
- DataTable
- DataRelation
- Connection object
- Command Object
- Data Reader
- Object
- DataAdapter Object
- Data Controls (Repeater, DataList, DataGrid)
- Binding data with Crystal Report
- Performing CRUD operations

# Up next

- ASP.Net Web API and Security
- Introduction
- Understanding HTTP methods
- Creating API controller
- API configuration
- Routing
- Parameter binding
- request and response
- API filters
- Form Based Security
- Windows based security

- End of unit 4