ADO**ActiveX Data Objects**.NET

Prepared for Vth semester DDU-CE students
2022-23 WAD

Apurva A Mehta

Memory

- ____ is the capital of India.
- Which of the following city is not part of Gujarat?
  - Amdavad
  - Baroda
  - Pune
  - Rajkot
- Day before exam



What is ADO.NET ?

- It is a part of the base class library that is included with the Microsoft .NET framework.
- It is used by programmers to access and modify data stored in relational database systems.
- ADO.NET is a technology designed to let an ASP.NET program (or any other .NET program, for that matter) access data .

Cont.

- ADO.NET relies on the functionality in a small set of core classes
  - Container Class
    - Used to contain and manage data
      - DataSet, DataTable, DataRow and DataRelation
  - Connector Class
    - Used to connect to a specific data source
      - Connection, Command and DataReader

Important Namespaces

- System.Data
- System.Data.SqlClient
- System.Data.SqlTypes

Interaction with Database

- Direct Database Access
  - You don't keep a copy of the information in memory.
  - Instead, you work with it for a brief period of time while the database connection is open, and then close the connection as soon as possible.

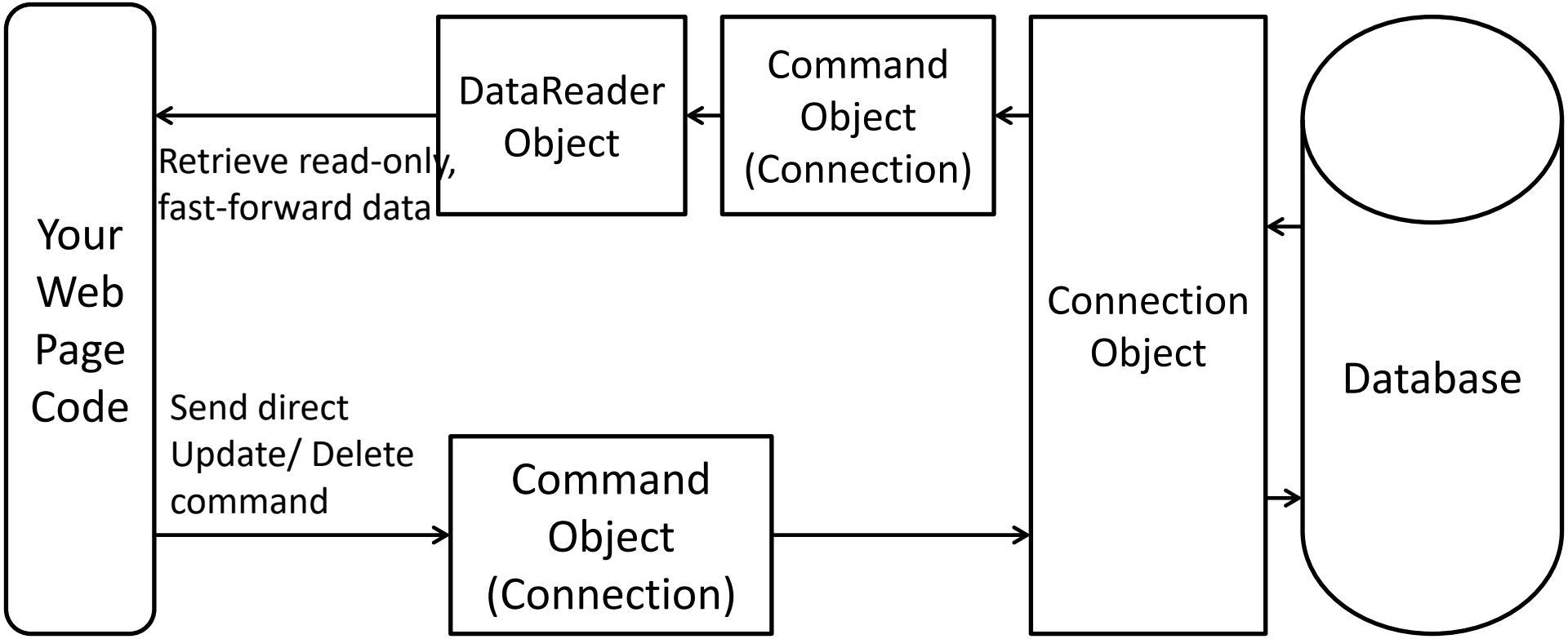
Cont.

- Disconnected Access
  - You keep a copy of the data in the Dataset object so you can work with it after the database connection has been closed.

Cont.

- The direct data model is well suited to ASP.NET web pages
  - No need to keep a copy of their data in memory for long periods of time
  - ASP.NET Page Load →Request
  - ASP.NET Page Shutdown → Response
- Performance improvement with Disconnected access
  - Could get the product catalog from a database once, and keep that data in memory on the web server so you can reuse it when someone else requests the same page
    - Caching

Direct Data Access



Steps Involved

- To query information
  - Create Connection, Command, and DataReader objects.
  - Use the DataReader to retrieve information from the database, and display it in a control on a web form.
  - Close your connection
  - Send the page to the user.
    - At this point, the information your user sees and the information in the database no longer have any connection, and all the ADO.NET objects have been destroyed.

Steps Involved

- To add/ update information
  - Create new Connection and Command objects.
  - Execute the command

Connection Object

- The first thing that we will have to do, when working with databases is to create a connection object.
- 2 Ways of creating connection object
- Do not hard code connection string: Web.config

Ways to Specify Connection String

```
SqlConnection con = new SqlConnection(@"Data
    Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=D:\Personal\WDDN\C# Codes 2019-
    20\ADONET_Demo1\ADONET_Demo1\App_Data\Test.mdf;Integrated Security=True");
```

```
SqlConnection con = new SqlConnection();
con.ConnectionString = @"Data
    Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=D:\Personal\WDDN\C# Codes 2019-
    20\ADONET_Demo1\ADONET_Demo1\App_Data\Test.mdf;Integrated Security=True";
```

```
<configuration>
  <connectionStrings>
    <add name="ConTest" connectionString="Data Source=(LocalDB)\MSSQLLocalDB;
         AttachDbFilename=D:\Personal\WDDN\C# Codes 2019-20\ADONET_Demo1\ADONET_Demo1\App_Data\Test.mdf;
         Integrated Security=True"/>
  </connectionStrings>
  <system.web>
    <compilation debug="true" targetFramework="4.5.2"/>
    <httpRuntime targetFramework="4.5.2"/>
  </system.web>
  <system.codedom>
```


```
<connectionStrings>
    <add name="ConTest"
    connectionString="Data
    Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=D:\Personal\W
    DDN\C# Codes 2019-
    20\ADONET_Demo1\ADONET_Demo1\App_Data\Test.mdf;Integrated
    Security=True"/>
  </connectionStrings>
```

Access Connection String from Web.config

using System.Web.Configuration;

SqlConnection con = new SqlConnection();

con.ConnectionString = WebConfigurationManager.ConnectionStrings["ConTest"].ConnectionString;

```
SqlConnection con = new SqlConnection();
con.ConnectionString = WebConfigurationManager.ConnectionStrings["ConTest"].ConnectionString;
```

ConnectionString and Properties

- Key/Value pairs that has the information required to create a connection object
- Properties
  - Data Source
  - Initial Catalog
  - SSIP
  - Connection Timeout

Open and Close Connection

- Create Connection
  - SqlConnection con = new SqlConnection();
- Open Connection
  - con.Open()
- Do your tasks

  //
- Close your Connection
  - con.Close()

Alternatives

```
using(object)
 {
    …
 }
```

- Disposable Object
- CLR
  - Dispose()
- Dispose()←→Close()
- Shorten database code
- No need of finally block

Command Object

- A Command object executes the SQL statement

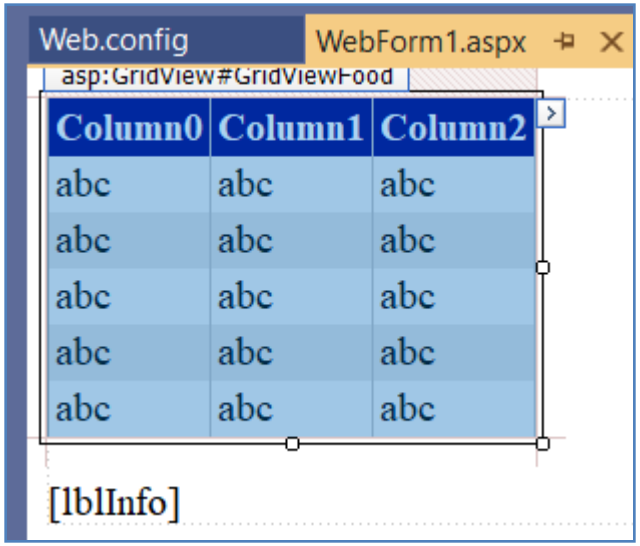
```
string command = "Select * from Food";
SqlCommand cmd = new SqlCommand(command,
   con);
```

OR

```
SqlCommand cmd = new SqlCommand()
cmd.Conenction = con;
cmd.CommandText = “Select * from Food”
```

```
SqlConnection con = new SqlConnection();
con.ConnectionString = WebConfigurationManager.ConnectionStrings["ConTest"].ConnectionString;
try
{
    using (con)
    {
        string command = "Select * from Food";
        SqlCommand cmd = new SqlCommand(command, con);
        con.Open();
        lblInfo.Text = "<b> Server Version: </b>" + con.ServerVersion;
        lblInfo.Text += "<br/> <b> Connection was: </b>" + con.State.ToString();
        SqlDataReader rdr = cmd.ExecuteReader();
        GridViewFood.DataSource = rdr;
        GridViewFood.DataBind();
    }
}
catch (Exception err)
{
    //Handle execeptions if any
    lblInfo.Text = "Error reading the datastore: ";
    lblInfo.Text += err.Message;
}
lblInfo.Text += "<b> Now Connection is: </b>";
lblInfo.Text += con.State.ToString();
```

```
<configuration>
  <connectionStrings>
    <add name="ConTest" connectionString="Data Source=(LocalDB)\MSSQLLocalDB;
         AttachDbFilename=D:\Personal\WDDN\C# Codes 2019-20\ADONET_Demo1\ADONET_Demo1\App_Data\Test.mdf;
         Integrated Security=True"/>
  </connectionStrings>
  <system.web>
    <compilation debug="true" targetFramework="4.5.2"/>
    <httpRuntime targetFramework="4.5.2"/>
  </system.web>
  <system.codedom>
    <compilers>
      <compiler language="c#;cs;csharp" extension=".cs"
        type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.CSharpCodeProvider, Microsoft.CodeDom.Pr
        warningLevel="4" compilerOptions="/langversion:6 /nowarn:1659;1699;1701"/>
      <compiler language="vb;vbs;visualbasic;vbscript" extension=".vb"
        type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.VBCodeProvider, Microsoft.CodeDom.Provid
```



Server Explorer

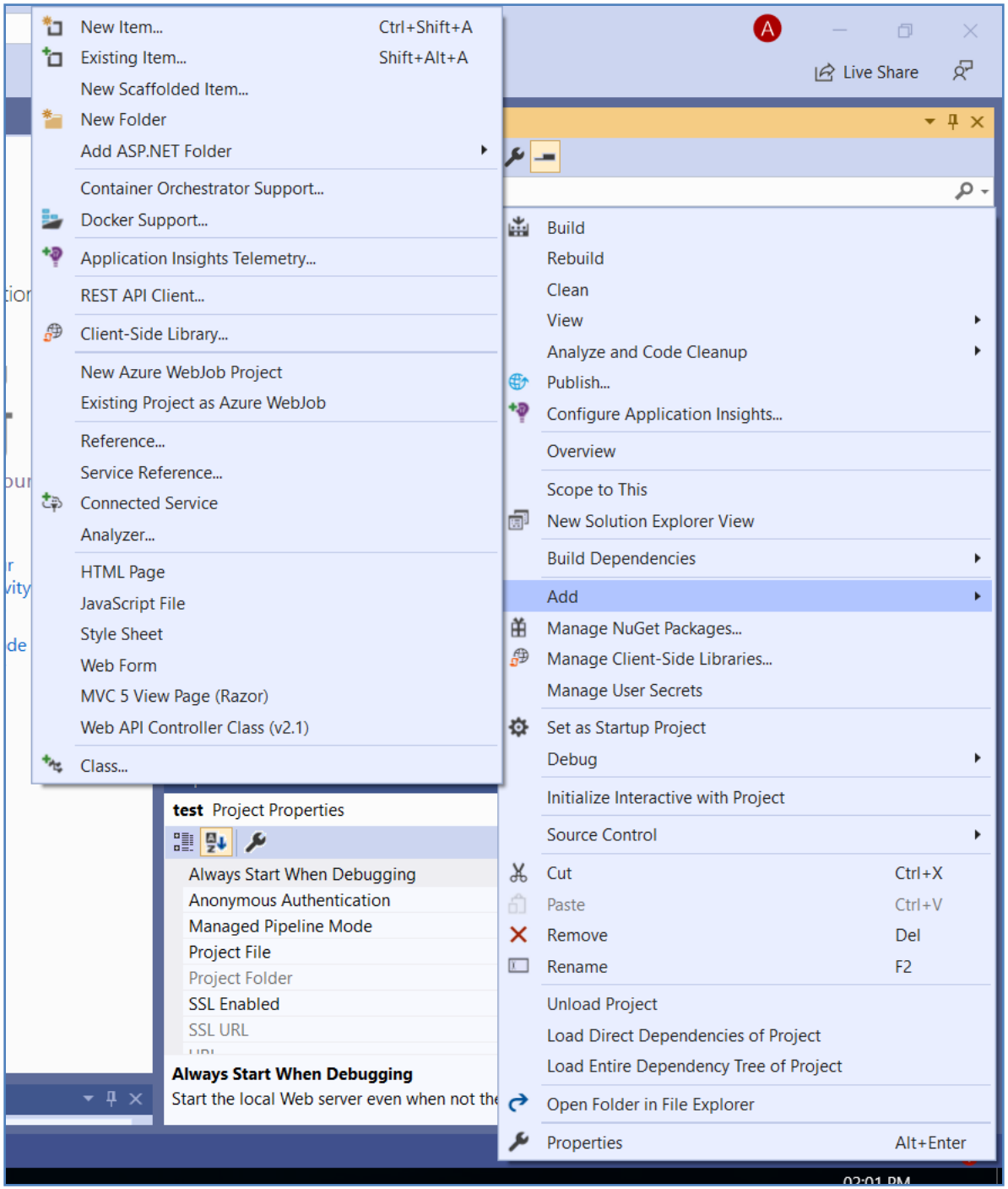
- Azure (apurvamehta2015@hotmail.com
- Data Connections
  - Publication.mdf
  - Test.mdf
    - Tables
      - Food
        - Id
        - Name
        - Day
        - Type
    - Views
    - Stored Procedures
    - Functions
    - Synonyms
    - Types
    - Assemblies

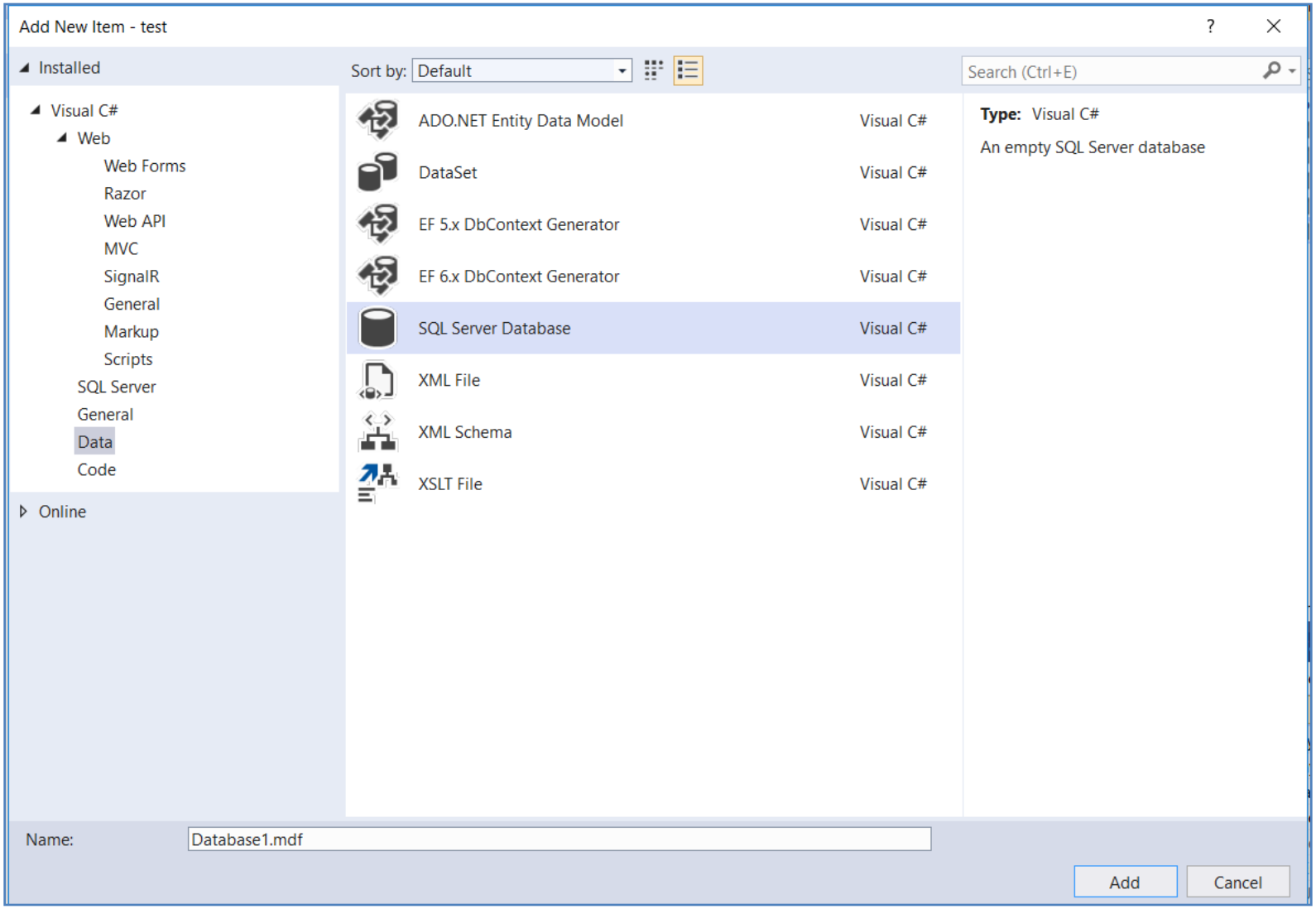
dbo.Food [Data] | Web.config | WebForm1.aspx | WebForm1

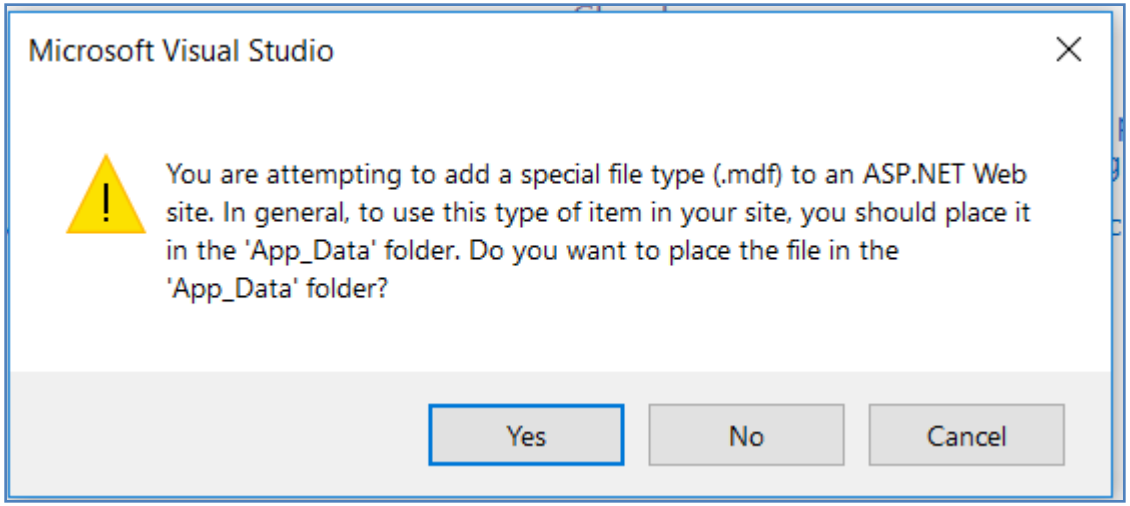
Max Rows: 1000

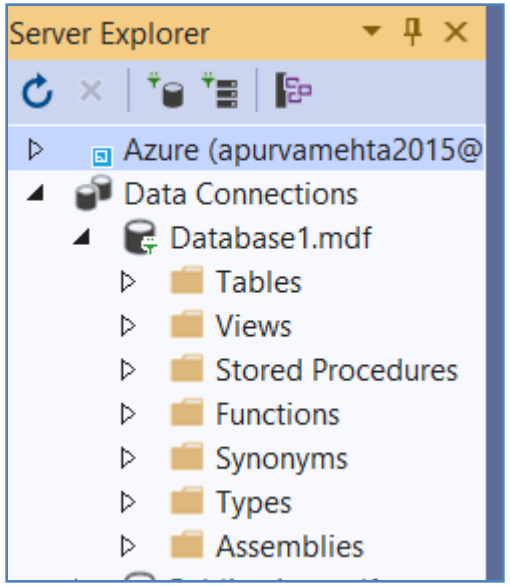
Id	Name	Day	Type
1	Potato Variation...	Monday	Lunch
2	Chola	Tuesday	Lunch
3	Moong	Wednesday	Lunch
4	Leafy Vegetable	Thursday	Lunch
5	Chana	Friday	Lunch
6	Mixdal	Saturday	Lunch
NULL	NULL	NULL	NULL

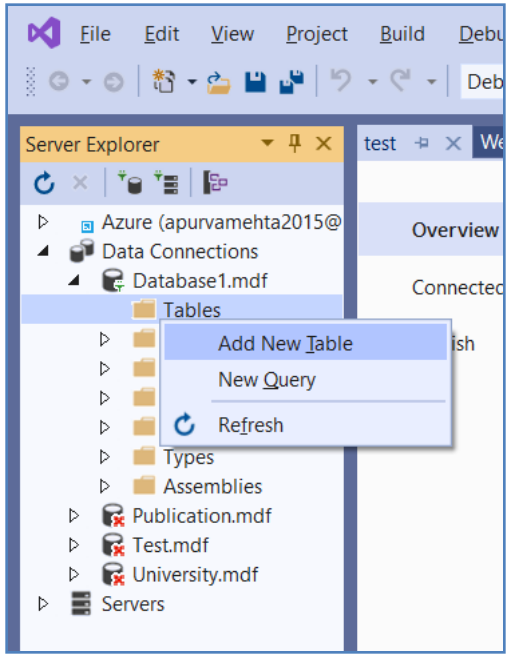
- **ADONET_Demo1**
  - Connected Services
  - Properties
  - References
  - App_Data
    - Test.mdf
  - packages.config
  - Web.config
  - WebForm1.aspx
    - WebForm1.aspx.cs
    - WebForm1.aspx.designer.cs

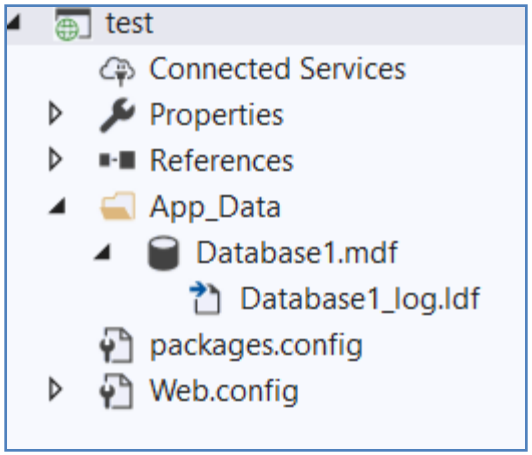

New Item... Ctrl+Shift+A
Existing Item... Shift+Alt+A
New Scaffolded Item...
New Folder
Add ASP.NET Folder
Container Orchestrator Support...
Docker Support...
Application Insights Telemetry...
REST API Client...
Client-Side Library...
New Azure WebJob Project
Existing Project as Azure WebJob
Reference...
Service Reference...
Connected Service
Analyzer...
HTML Page
JavaScript File
Style Sheet
Web Form
MVC 5 View Page (Razor)
Web API Controller Class (v2.1)
Class...
test Project Properties
Always Start When Debugging
Anonymous Authentication
Managed Pipeline Mode
Project File
Project Folder
SSL Enabled
SSL URL
Always Start When Debugging
Start the local Web server even when not the
Live Share
Build
Rebuild
Clean
View
Analyze and Code Cleanup
Publish...
Configure Application Insights...
Overview
Scope to This
New Solution Explorer View
Build Dependencies
Add
Manage NuGet Packages...
Manage Client-Side Libraries...
Manage User Secrets
Set as Startup Project
Debug
Initialize Interactive with Project
Source Control
Cut Ctrl+X
Paste Ctrl+V
Remove Del
Rename F2
Unload Project
Load Direct Dependencies of Project
Load Entire Dependency Tree of Project
Open Folder in File Explorer
Properties Alt+Enter


Add New Item - test
Installed
Sort by: Default
Search (Ctrl+E)
Visual C#
Web
Web Forms
Razor
Web API
MVC
SignalR
General
Markup
Scripts
SQL Server
General
Data
Code
Online
ADO.NET Entity Data Model
Visual C#
DataSet
Visual C#
EF 5.x DbContext Generator
Visual C#
EF 6.x DbContext Generator
Visual C#
SQL Server Database
Visual C#
XML File
Visual C#
XML Schema
Visual C#
XSLT File
Visual C#
Type: Visual C#
An empty SQL Server database
Name:
Database1.mdf
Add
Cancel


Microsoft Visual Studio
You are attempting to add a special file type (.mdf) to an ASP.NET Web site. In general, to use this type of item in your site, you should place it in the 'App_Data' folder. Do you want to place the file in the 'App_Data' folder?
Yes
No
Cancel


Server Explorer
Azure (apurvamehta2015@
Data Connections
Database1.mdf
Tables
Views
Stored Procedures
Functions
Synonyms
Types
Assemblies


File
Edit
View
Project
Build
Server Explorer
test
Overview
Azure (apurvamehta2015@
Data Connections
Database1.mdf
Tables
Add New Table
New Query
Refresh
Types
Assemblies
Publication.mdf
Test.mdf
University.mdf
Servers


test
Connected Services
Properties
References
App_Data
Database1.mdf
Database1_log.ldf
packages.config
Web.config

dbo.Customer [Design] | WebForm1.aspx.cs | Output

Update | Script File: dbo.Customer.sql

	Name	Data Type	Allow Nulls	Default
	CustomerId	int	☐	
	Name	nvarchar(50)	☐	
			☐	

- Test.mdf
  - Tables
    - Customer
      - CustomerId
      - Name
    - Inventory
      - CarId
      - Make
      - Color
    - Order
      - OrderId
      - CarId
      - CustomerId

Properties

CustomerId Column

Allow Nulls	**False**
Collation	
⊞ Computed Column Specific	
⊞ Data Classification	
Data Type	**int**
Default Value or Binding	
Description	
⊞ Full Text Specification	False
⊟ Identity Specification	True
(Is Identity)	**True**
Identity Increment	**1**
Identity Seed	**1**
Is Column Set	False
Is File Stream	False
Is ROWGUID Column	False
Is Sparse	False
Not For Replication	False
Primary Key	True

dbo.Order [Design]* | dbo.Customer [Design] | WebForm1.aspx.cs | Output

Update | Script File: dbo.Order.sql*

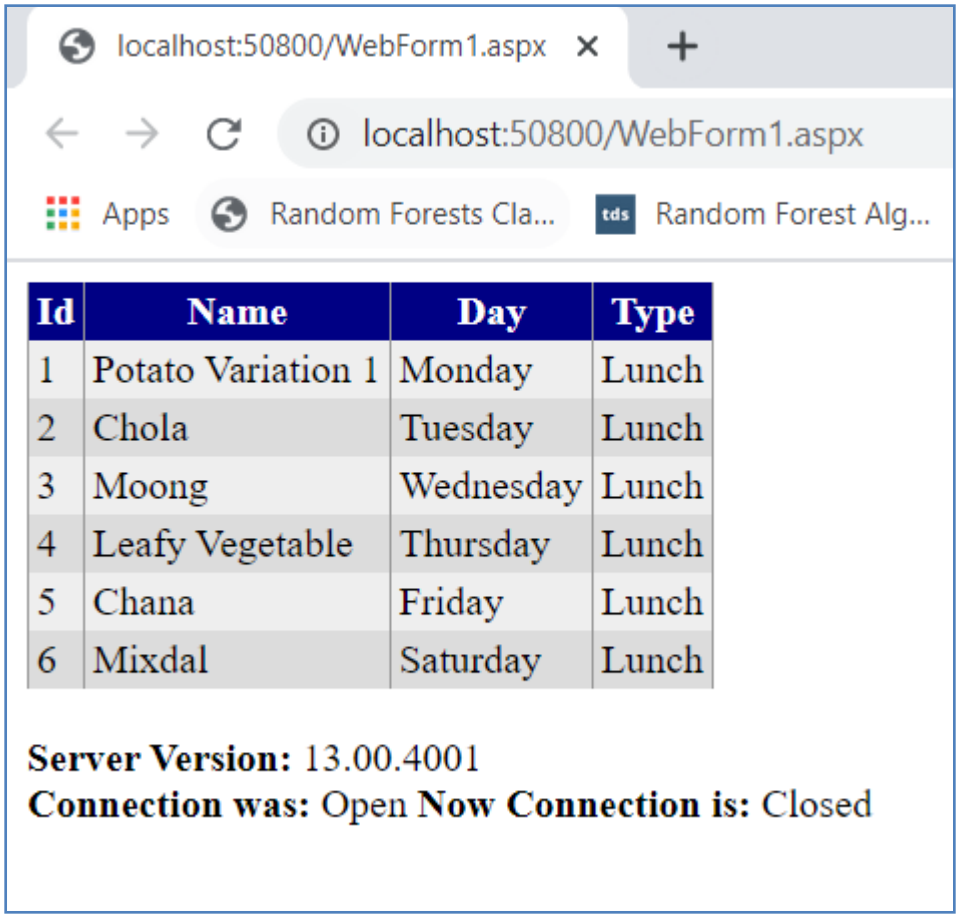
	Name	Data Type	Allow Nulls	Default
	OrderId	int	☐	
	CarId	int	☐	
	CustomerId	int	☐	
			☐	

- **Keys** (1)
  - <unnamed> (Primary Key, Clustered: OrderId)
- **Check Constraints** (0)
- **Indexes** (0)
- **Foreign Keys** (2)
  - FK_Order_Customer (CustomerId)
  - FK_Order_Inventory (CarId)
- **Triggers** (0)

Design | T-SQL

```sql
CREATE TABLE [dbo].[Order] (
    [OrderId]    INT IDENTITY (1, 1) NOT NULL,
    [CarId]      INT NOT NULL,
    [CustomerId] INT NOT NULL,
    PRIMARY KEY CLUSTERED ([OrderId] ASC),
    CONSTRAINT [FK_Order_Customer] FOREIGN KEY ([CustomerId]) REFERENCES [dbo].[Customer] ([CustomerId]),
    CONSTRAINT [FK_Order_Inventory] FOREIGN KEY ([CarId]) REFERENCES [dbo].[Inventory] ([CarId])
);
```

```
SqlConnection con = new SqlConnection();
con.ConnectionString = WebConfigurationManager.ConnectionStrings["ConTest"].ConnectionString;
try
{
    using (con)
    {
        string command = "Select * from Food";
        SqlCommand cmd = new SqlCommand(command, con);
        con.Open();
        lblInfo.Text = "<b> Server Version: </b>" + con.ServerVersion;
        lblInfo.Text += "<br/> <b> Connection was: </b>" + con.State.ToString();
        SqlDataReader rdr = cmd.ExecuteReader();
        GridViewFood.DataSource = rdr;
        GridViewFood.DataBind();
    }
}
catch (Exception err)
{
    //Handle execeptions if any
    lblInfo.Text = "Error reading the datastore: ";
    lblInfo.Text += err.Message;
}
lblInfo.Text += "<b> Now Connection is: </b>";
lblInfo.Text += con.State.ToString();
```



Id	Name	Day	Type
1	Potato Variation 1	Monday	Lunch
2	Chola	Tuesday	Lunch
3	Moong	Wednesday	Lunch
4	Leafy Vegetable	Thursday	Lunch
5	Chana	Friday	Lunch
6	Mixdal	Saturday	Lunch

- Connection is closed first and then exception-handling code is triggered
- It’s always good to close database connections as soon as possible

DataReader

- After command is defined
- Quickly retrieve all your results
  - Fast-forward-only read-only access

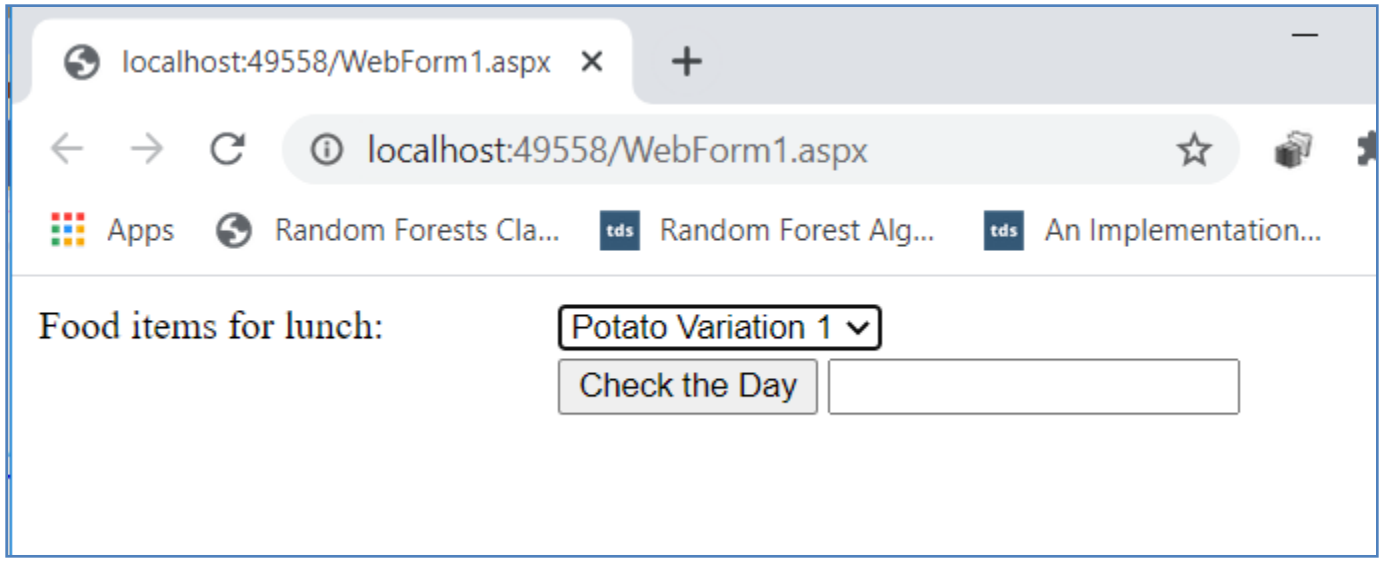
```
con.Open();
SqlDataReader rdr = cmd.ExecuteReader();
rdr.Read();
```

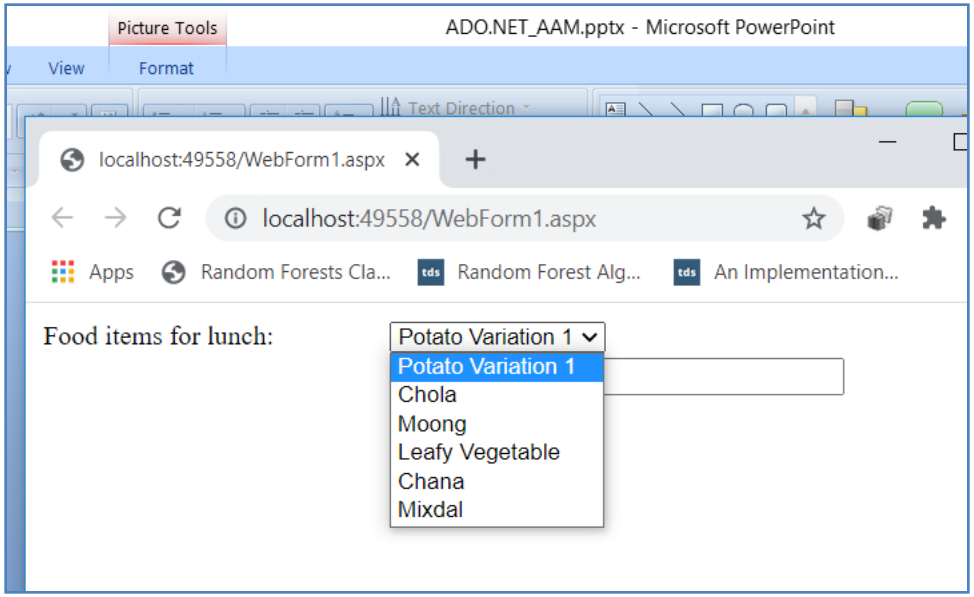
```xml
<configuration>
  <connectionStrings>
    <add name="Connection1" connectionString="Data Source=(LocalDB)\MSSQLLocalDB;
         AttachDbFilename=D:\C# Codes 2019-20\ADONET_Demo1\ADONET_Demo1\App_Data\Test.mdf;
         Integrated Security=True"/>
  </connectionStrings>
  <system.web>
    <compilation debug="true" targetFramework="4.5.2"/>
    <httpRuntime targetFramework="4.5.2"/>
  </system.web>
  <system.codedom>
```

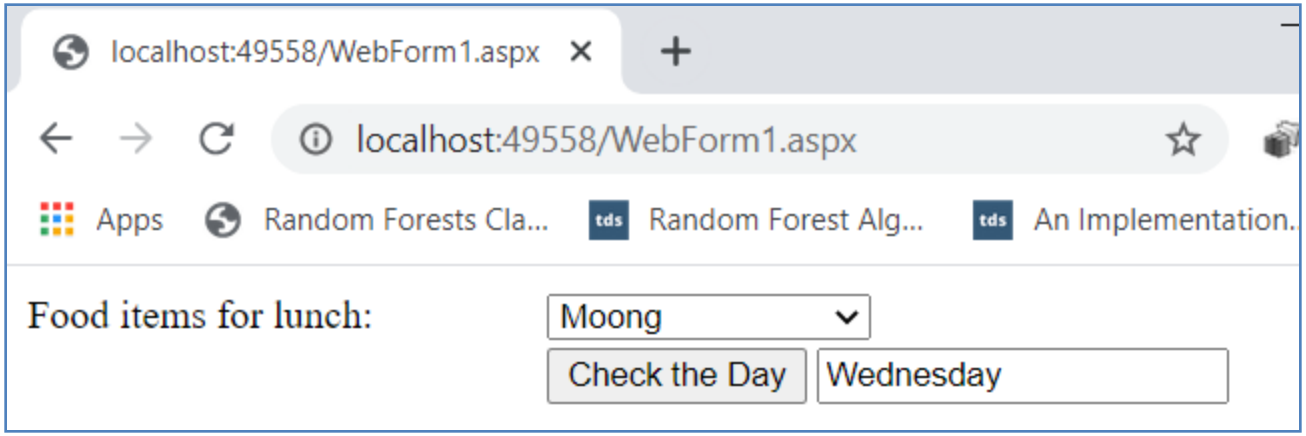
```xml
<tr>
    <td>Food items for lunch:</td>
    <td>
        <asp:DropDownList ID="ddlLunch" runat="server">
        </asp:DropDownList>
    </td>
</tr>
<tr>
    <td> </td>
    <td>
        <asp:Button ID="btnCheck" runat="server" OnClick="btnCheck_Click"
            Text="Check the Day" />
        <asp:TextBox ID="txtDay" runat="server"></asp:TextBox>
    </td>
</tr>
<tr>
    <td>
        <asp:Label ID="lblInfo" runat="server"></asp:Label>
    </td>
    <td> </td>
</tr>
```

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        SqlConnection con = new SqlConnection();
        con.ConnectionString =
            WebConfigurationManager.ConnectionStrings["Connection1"].ConnectionString;
        try
        {
            using (con)
            {
                string command = "Select * from Food";
                SqlCommand cmd = new SqlCommand(command, con);
                con.Open();
                SqlDataReader rdr = cmd.ExecuteReader();
                while (rdr.Read())
                {
                    ListItem item = new ListItem();
                    item.Text = rdr["Name"].ToString();
                    ddlLunch.Items.Add(item);
                }
                rdr.Close();
            }
        }
        catch (Exception err)
        {
            //Handle execeptions if any
            lblInfo.Text = "Error reading the datastore: ";
            lblInfo.Text += err.Message;
        }
    }
}
```

```
protected void btnCheck_Click(object sender, EventArgs e)
{
    string input = ddlLunch.SelectedItem.Text;
    SqlConnection con = new SqlConnection();
    con.ConnectionString =
        WebConfigurationManager.ConnectionStrings["Connection1"].ConnectionString;
    try
    {
        using (con)
        {
            string command = "Select Day from Food where Name='" + input + "'  ";
            SqlCommand cmd = new SqlCommand(command, con);
            con.Open();
            SqlDataReader rdr = cmd.ExecuteReader();
            while (rdr.Read())
            {
                txtDay.Text = rdr["Day"].ToString();
            }
            rdr.Close();
        }
    }
    catch (Exception err)
    {
        //Handle execeptions if any
        lblInfo.Text = "Error reading the datastore: ";
        lblInfo.Text += err.Message;
    }
}
}
```


localhost:49558/WebForm1.aspx
localhost:49558/WebForm1.aspx
Apps
Random Forests Cla...
Random Forest Alg...
An Implementation...
Food items for lunch:
Potato Variation 1
Check the Day

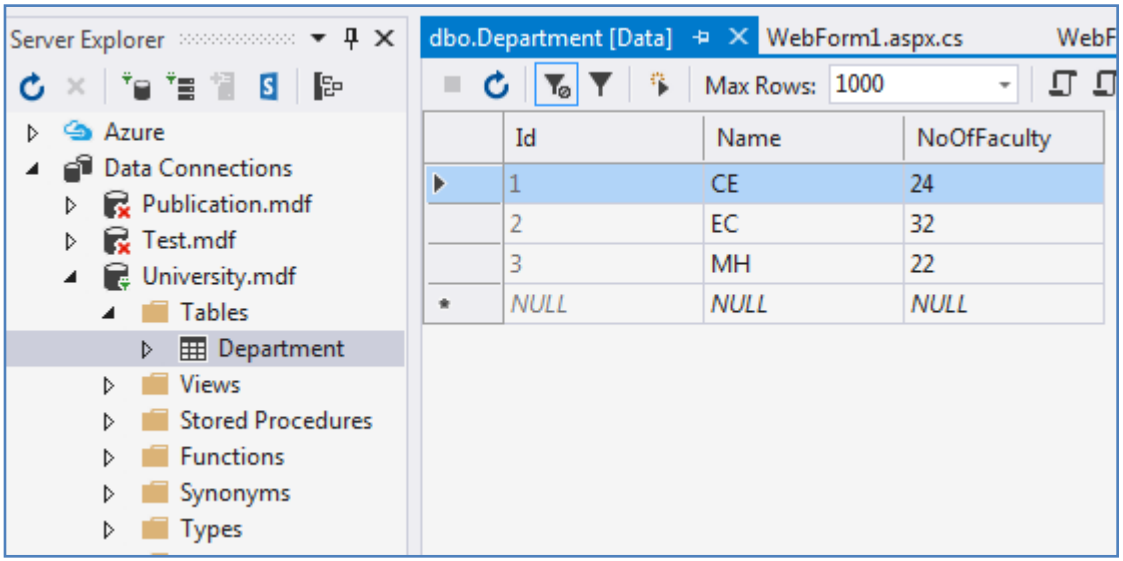

Picture Tools
ADO.NET_AAM.pptx - Microsoft PowerPoint
View
Format
Text Direction
localhost:49558/WebForm1.aspx
localhost:49558/WebForm1.aspx
Apps
Random Forests Cla...
Random Forest Alg...
An Implementation...
Food items for lunch:
Potato Variation 1
Potato Variation 1
Chola
Moong
Leafy Vegetable
Chana
Mixdal


localhost:49558/WebForm1.aspx
localhost:49558/WebForm1.aspx
Apps
Random Forests Cla...
Random Forest Alg...
An Implementation.
Food items for lunch:
Moong
Check the Day
Wednesday

Application

Select Department | Select ▼ | Display Faculty Count

No. of Faculty | None



Select Department | CE ▼ | Display Faculty Count

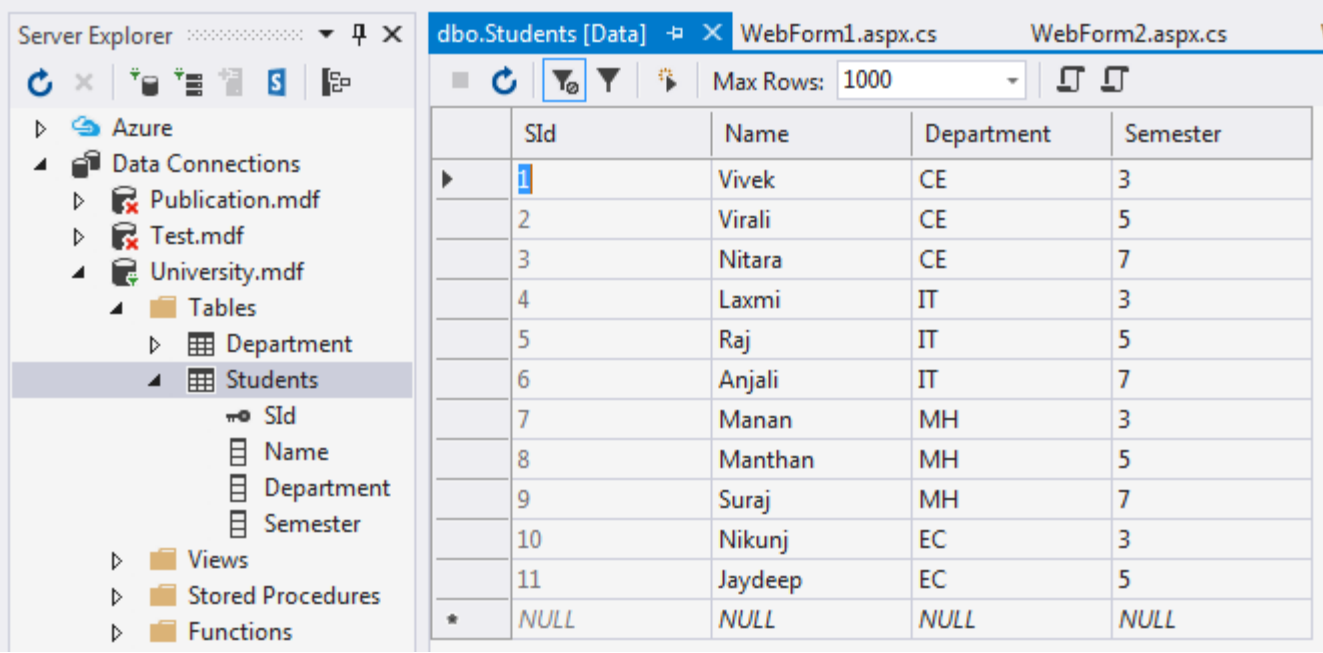
No. of Faculty | 24

Application

Select Department Select ▼

No. of Students None Display Student Count

Students Information



SId	Name	Department	Semester
1	Vivek	CE	3
2	Virali	CE	5
3	Nitara	CE	7
4	Laxmi	IT	3
5	Raj	IT	5
6	Anjali	IT	7
7	Manan	MH	3
8	Manthan	MH	5
9	Suraj	MH	7
10	Nikunj	EC	3
11	Jaydeep	EC	5
NULL	NULL	NULL	NULL

Select Department EC ▼

No. of Students 2 Display Student Count

Students Information

SId	Name	Department	Semester
10	Nikunj	EC	3
11	Jaydeep	EC	5

ExecuteReader()

- **ExecuteReader** gives you a data reader back which will allow you to read all of the columns of the results a row at a time.
- Return type is DataReader.
- Return value is compulsory and should be assigned to an another object DataReader.

ExecuteNonQuery()

- ExecuteNonQuery is used to execute SQL Statement and it returns number of rows affected.
- We cannot use ExecuteNonQuery while we are expecting some data from the SQL query.
- We can use ExecuteNonQuery in INSERT, UPDATE AND DELETE queries where they are not returning any result.
- Return type is int.
- Return value is optional and can be assigned to an integer variable.

Sample Sql Statements

```
string query = "INSERT INTO Products (Name, Price, Date) VALUES(@Name, @Price, @Date)";
cmd.Parameters.AddWithValue("@Name", "USB Keyboard");
cmd.Parameters.AddWithValue("@Price", "20");
cmd.Parameters.AddWithValue("@Date", "25 May 2017");
```

Cont.

UPDATE Customers SET Name = @Name, Country = @Country WHERE CustomerId = @CustomerId

DELETE FROM Customers WHERE CustomerId = @CustomerId

Disconnected Data Access

- Use the DataSet to keep a copy of your data in memory
  - You need to do something time-consuming with the data
    - By dumping it into a DataSet first, you ensure that the database connection is kept open for as little time as possible.
  - You want to navigate backward and forward through your data while you're processing it.
    - This isn't possible with the DataReader, which goes in one direction only—forward.

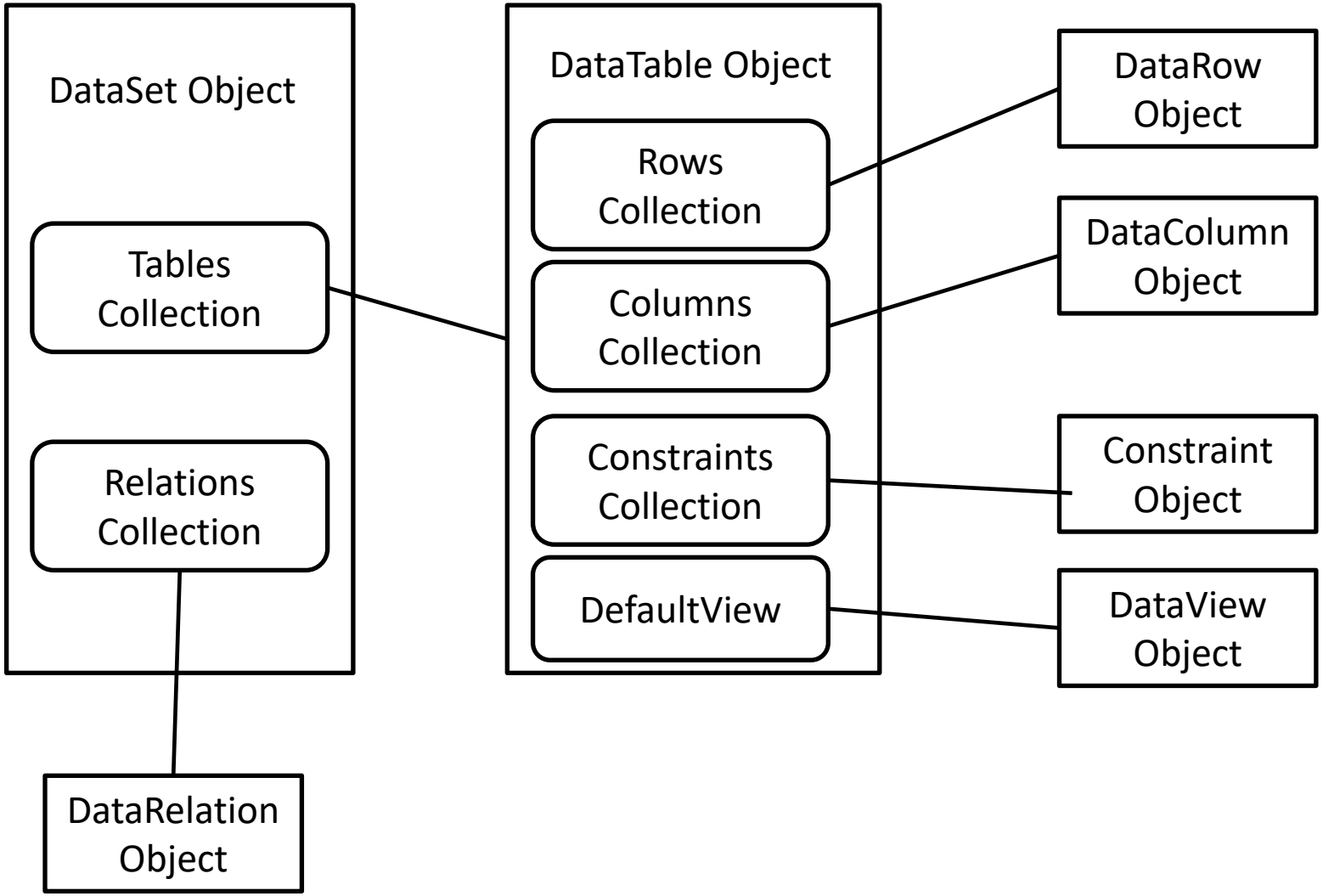
Cont.

- You want to navigate from one table to another.
  - Using the DataSet, you can store several tables of information.
  - You can even define relationships that allow you to browse through them more efficiently.
- You want to save the data to a file for later use.
  - The DataSet includes two methods—
    - WriteXml() and ReadXml()—that allow you to dump the content to a file and convert it back to a live database object later

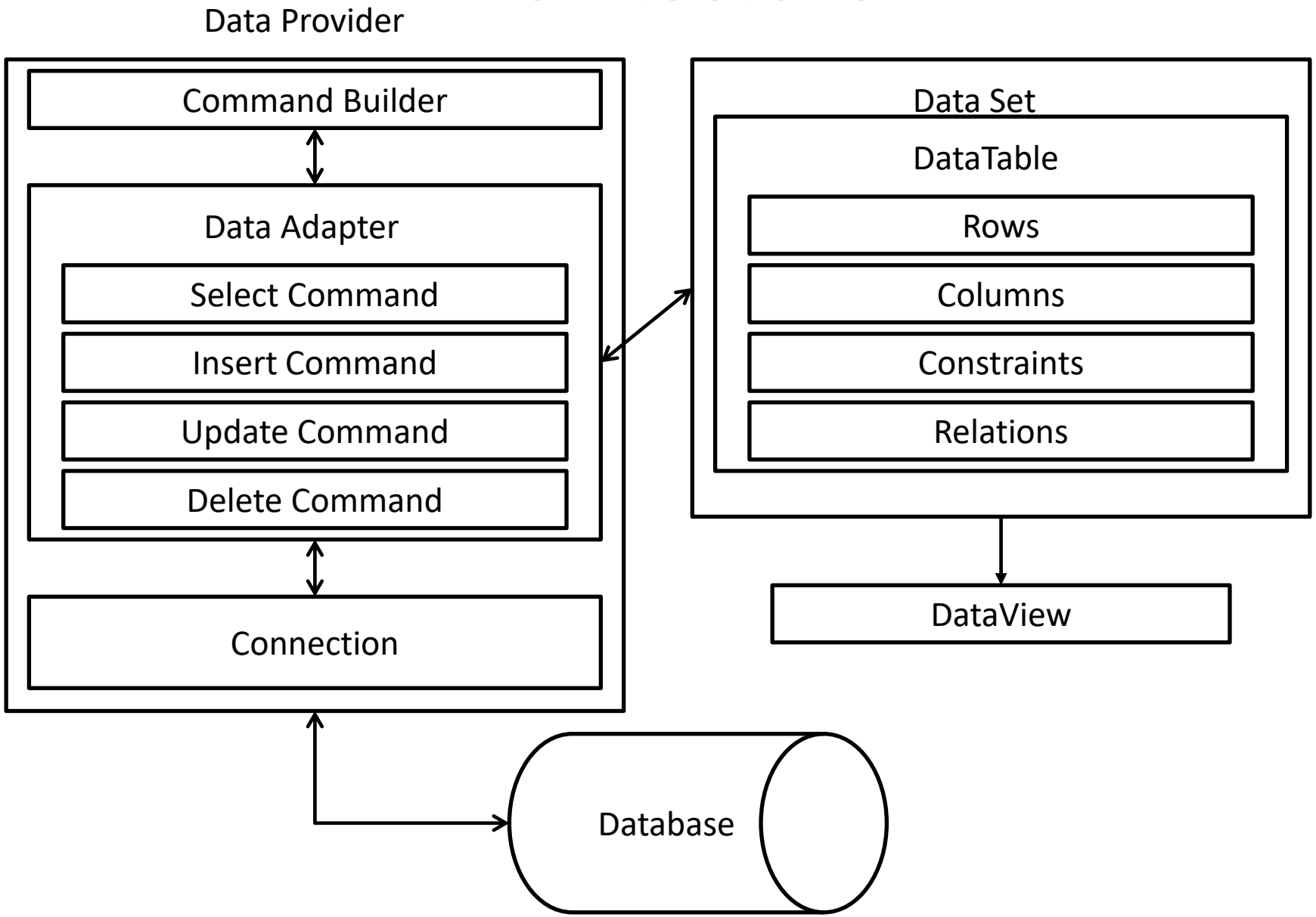
Updating Disconnected Data

- The DataSet tracks the changes you make to the records inside.
  - This allows you to use the DataSet to update records.
  - The basic principle is simple.
    - You fill a DataSet in the normal way, modify one or more records, and then apply your update by using a DataAdapter.

Dataset family of Obejcts



Architecture



```
<configuration>
  <connectionStrings>
    <add name="Connection1" connectionString="Data Source=(LocalDB)\MSSQLLocalDB;
         AttachDbFilename=
         D:\C# Codes 2019-20\ADO_Disconnected_Class_Demo\ADO_Disconnected_Class_Demo
         \App_Data\Test.mdf;
         Integrated Security=True"/>
  </connectionStrings>
  <system.web>
    <compilation debug="true" targetFramework="4.5.2"/>
    <httpRuntime targetFramework="4.5.2"/>
  </system.web>
  <system.codedom>
    <compilers>
      <compiler language="c#;cs;csharp" extension=".cs"
        type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.CSharpCodeProvider, M
```

```
public partial class WebForm1 : System.Web.UI.Page
{
    DataSet ds;
    0 references
    protected void Page_Load(object sender, EventArgs e)
    {
        string constring = WebConfigurationManager.
            ConnectionStrings["Connection1"].ConnectionString;
        SqlConnection con = new SqlConnection(constring);
        SqlCommand cmd = new SqlCommand();
        cmd.Connection = con;
        cmd.CommandText = "select * from Info";
        SqlDataAdapter da = new SqlDataAdapter(cmd);
        ds = new DataSet();
        try
        {
            using (con)
            {
                con.Open();
                da.Fill(ds, "Info");
            }
        }
```

```
        catch(Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
        //GridViewDetails.DataSource = ds;
        GridViewDetails.DataSource = ds.Tables["Info"];
        GridViewDetails.DataBind();
    }

    0 references
    protected void btnCity_Click(object sender, EventArgs e)
    {
        ddlCity.Items.Clear();
        foreach (DataRow row in ds.Tables["Info"].Rows)
        {
            ddlCity.Items.Add(row["City"].ToString());
        }
    }
}
```

```
<tr>
    <td>
        <asp:GridView ID="GridViewDetails" runat="server">
        </asp:GridView>
    </td>
    <td> </td>
</tr>
<tr>
    <td> </td>
    <td> </td>
</tr>
<tr>
    <td>
        <asp:Button ID="btnCity" runat="server" OnClick="btnCity_Click" Text="City" />
    </td>
    <td>
        <asp:DropDownList ID="ddlCity" runat="server">
        </asp:DropDownList>
    </td>
</tr>
<tr>
```

localhost:53311/WebForm1.aspx

localhost:53311/WebForm1.aspx

Apps

Id	Name	City
2	Bhavi	Jetpur
3	Lakshmi	Kadi
5	Test	Nadiad
6	Jyoti	Modasa
7	Vyom	Kheda

City

Apps

Id	Name	City
2	Bhavi	Jetpur
3	Lakshmi	Kadi
5	Test	Nadiad
6	Jyoti	Modasa
7	Vyom	Kheda

City

Jetpur

Jetpur
Kadi
Nadiad
Modasa
Kheda

Output | WebForm2.aspx.cs | Web.config | dbo.Info [Data]

Max Rows: 1000

Id	Name	City
2	Bhavi	Jetpur
3	Lakshmi	Kadi
5	Test	Nadiad
6	Jyoti	Modasa
7	Vyom	Kheda
NULL	NULL	NULL

```
<tr>
    <td>
        <asp:Button ID="btnAdd" runat="server" OnClick="btnAdd_Click" Text="Add" />
        <asp:TextBox ID="txtId" runat="server"></asp:TextBox>
        <asp:Button ID="btnModify" runat="server" Text="Modify"
            OnClick="btnModify_Click" />
    </td>
    <td> </td>
    <td> </td>
</tr>
```

```
<asp:Panel ID="Panel1" runat="server">
    <table class="auto-style1">
        <tr>
            <td>Name</td>
            <td>
                <asp:TextBox ID="txtName" runat="server"></asp:TextBox>
            </td>
        </tr>
        <tr>
            <td>City</td>
            <td>
                <asp:TextBox ID="txtCity" runat="server"></asp:TextBox>
            </td>
        </tr>
        <tr>
            <td colspan="2">
                <asp:Button ID="btnSubmit" runat="server" Text="Submit"
                    OnClick="btnSubmit_Click" />
            </td>
        </tr>
    </table>
</asp:Panel>
```

```
<asp:Panel ID="Panel2" runat="server">
    <table class="auto-style1">
        <tr>
            <td>Name:</td>
            <td>
                <asp:TextBox ID="txtName1" runat="server"></asp:TextBox>
            </td>
        </tr>
        <tr>
            <td>City:</td>
            <td>
                <asp:TextBox ID="txtCity1" runat="server"></asp:TextBox>
            </td>
        </tr>
        <tr>
            <td>
                <asp:Button ID="btnUpdate" runat="server" OnClick="btnUpdate_Click" Text="Update"
                <asp:Button ID="btnDelete" runat="server" OnClick="btnDelete_Click" Text="Delete"
            </td>
            <td>
                <asp:DetailsView ID="DetailsView1" runat="server" Height="50px" Width="125px">
                </asp:DetailsView>
            </td>
        </tr>
    </table>
</asp:Panel>
```

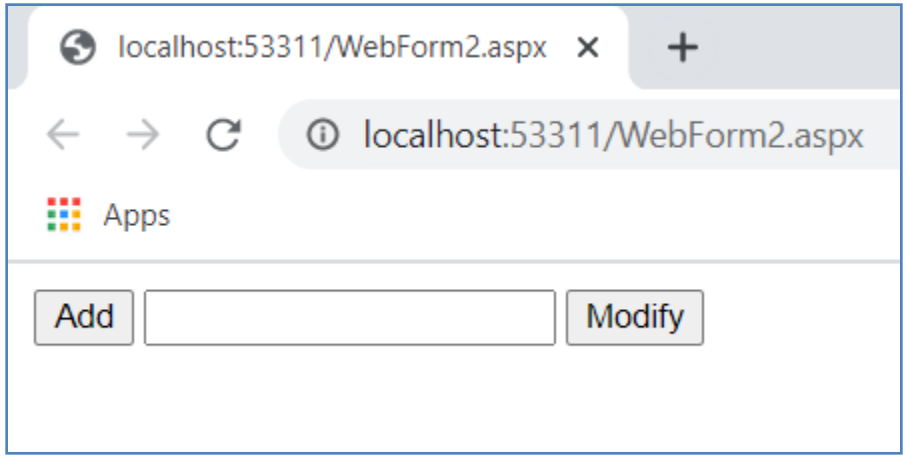
```
public partial class WebForm2 : System.Web.UI.Page
{
    DataSet ds;
    SqlConnection con;
    SqlCommand cmd;
    SqlDataAdapter da;
    string constring;
    SqlCommandBuilder builder;
    0 references
    protected void Page_Load(object sender, EventArgs e)
    {
        Panel1.Visible = false;
        Panel2.Visible = false;
    }

    3 references
    void db_init()
    {
        constring = WebConfigurationManager.
            ConnectionStrings["Connection1"].ConnectionString;
        con = new SqlConnection(constring);
        cmd = new SqlCommand();
        cmd.Connection = con;
```

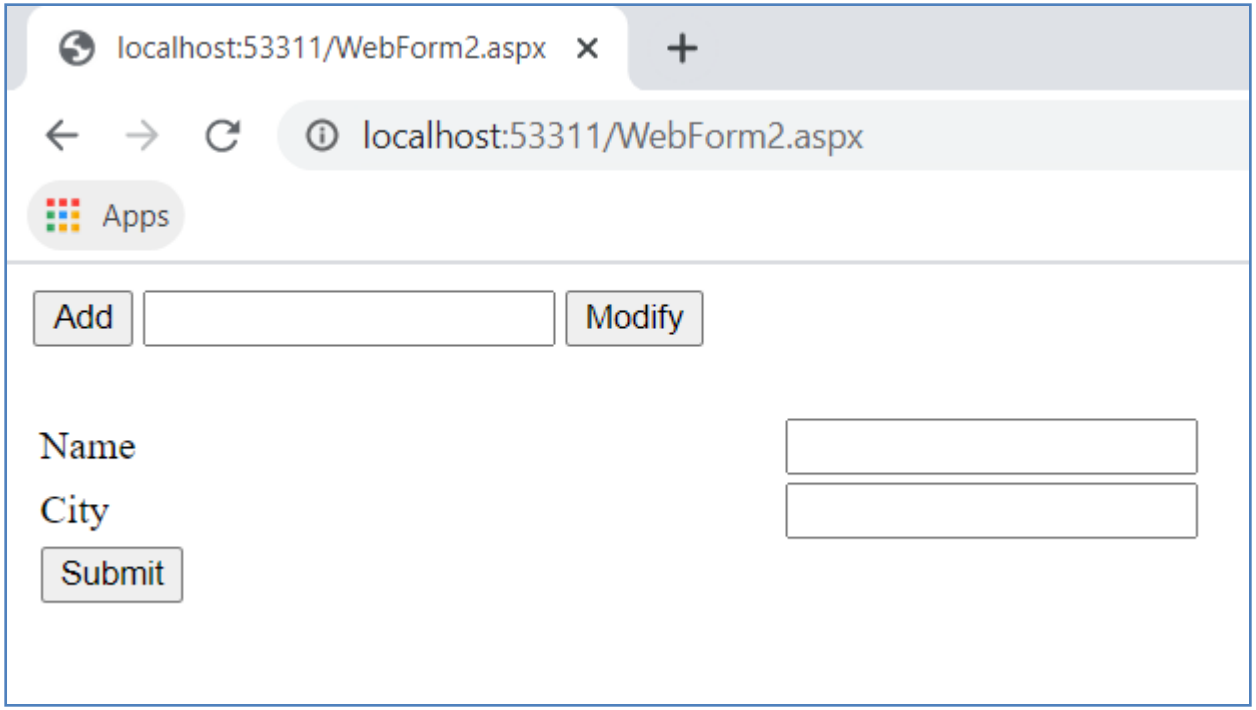
```
void db_init()
{
    constring = WebConfigurationManager.
        ConnectionStrings["Connection1"].ConnectionString;
    con = new SqlConnection(constring);
    cmd = new SqlCommand();
    cmd.Connection = con;
    cmd.CommandText = "select * from Info where Id='" + txtId.Text + "'";
    da = new SqlDataAdapter(cmd);
    ds = new DataSet();
    builder = new SqlCommandBuilder(da);
}
0 references
protected void btnAdd_Click(object sender, EventArgs e)
{
    Panel1.Visible = true;
}

0 references
protected void btnSubmit_Click(object sender, EventArgs e)
{
    string constring = WebConfigurationManager.
        ConnectionStrings["Connection1"].ConnectionString;
    SqlConnection conn = new SqlConnection(constring);
```

```
protected void btnSubmit_Click(object sender, EventArgs e)
{
    string constring = WebConfigurationManager.
        ConnectionStrings["Connection1"].ConnectionString;
    SqlConnection conn = new SqlConnection(constring);
    SqlCommand cmd = new SqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = "select * from Info";
    SqlDataAdapter da = new SqlDataAdapter(cmd);
    ds = new DataSet();
    SqlCommandBuilder builder = new SqlCommandBuilder(da);
    conn.Open();
    da.Fill(ds, "Info");
    conn.Close();
    DataTable dt = ds.Tables["Info"];
    DataRow dr = dt.NewRow();
    dr["Name"] = txtName.Text;
    dr["City"] = txtCity.Text;
    dt.Rows.Add(dr);
    da.Update(ds, "Info");
}
```

```
protected void btnUpdate_Click(object sender, EventArgs e)
{
    db_init();
    con.Open();
    da.Fill(ds, "Info");
    con.Close();
    DataTable dt = ds.Tables["Info"];
    DataRow row = dt.Rows[0];
    row["City"] = txtCity1.Text;
    row["Name"] = txtName1.Text;
    da.Update(ds, "Info");
}
0 references
protected void btnModify_Click(object sender, EventArgs e)
{
    Panel2.Visible = true;
    db_init();
    string filter = @"id=" + txtId.Text; // If you are selecting whole table
    con.Open();
    da.Fill(ds, "Info");
    con.Close();
    DataTable dt = ds.Tables["Info"];
    DataRow row = dt.Rows[0];
    txtCity1.Text = row["City"].ToString(); // Text Box value
    txtName1.Text = row["Name"].ToString();
}
```

```
protected void btnDelete_Click(object sender, EventArgs e)
{
    db_init();
    con.Open();
    da.Fill(ds, "Info");
    con.Close();
    int id = Convert.ToInt32(txtId.Text);
    DataRow rowToDelete = ds.Tables["Info"].
        AsEnumerable().FirstOrDefault(row => row.Field<int>("Id") == id);
    if (rowToDelete != null)
        rowToDelete.Delete();
    da.Update(ds, "Info");
    ds.AcceptChanges();
}
```


localhost:53311/WebForm2.aspx
localhost:53311/WebForm2.aspx
Apps
Add
Modify


localhost:53311/WebForm2.aspx
localhost:53311/WebForm2.aspx
Apps
Add
Modify
Name
City
Submit

localhost:53311/WebForm2.aspx

localhost:53311/WebForm2.aspx

Apps

Add Modify

Name Sur

City Varanasi

Submit

Output | WebForm2.aspx | WebForm2.aspx.cs | Web.config | dbo.Info [Data]

Max Rows: 1000

Id	Name	City
2	Bhavi	Jetpur
3	Lakshmi	Kadi
5	Test	Nadiad
6	Jyoti	Modasa
7	Vyom	Kheda
15	Sur	Varanasi
NULL	NULL	NULL

localhost:53311/WebForm2.aspx

Apps

Add | 15 | Modify

Name: Sur

City: Varanasi

Update | Delete

localhost:53311/WebForm2.aspx

Apps

Add | 15 | Modify

Name: Sur

City: Banaras

Update | Delete

Max Rows: 1000

	Id	Name	City
▶	2	Bhavi	Jetpur
	3	Lakshmi	Kadi
	5	Test	Nadiad
	6	Jyoti	Modasa
	7	Vyom	Kheda
	15	Sur	Banaras
*	NULL	NULL	NULL

Max Rows: 1000

Id	Name	City
2	Bhavi	Jetpur
3	Lakshmi	Kadi
5	Test	Nadiad
6	Jyoti	Modasa
7	Vyom	Kheda
15	Sur	Banaras
NULL	NULL	NULL

localhost:53311/WebForm2.aspx

localhost:53311/WebForm2.aspx

Apps

Add 5 Modify

Name: Test

City: Nadiad

Update Delete

Id	Name	City
2	Bhavi	Jetpur
3	Lakshmi	Kadi
6	Jyoti	Modasa
7	Vyom	Kheda
15	Sur	Banaras
NULL	NULL	NULL