What is ASP.Net?

It is the latest version of Classic ASP; it is a Web APPLICATION Framework developed by Microsoft to build dynamic data driven web applications and web services.

ASP.Net is a subset of .Net Framework. In simple terms framework is a collection of classes. ASP.Net is the successor to classic ASP (Active Server Pages).

What other technologies can be used to build web applications?

JS, Classic ASP, PHP, Python etc.

What is a web Application?

A web app is an app that runs on a web server unlike computer based Softwares that run locally on the operating system of that device.

What are the advantages of web applications?

1. Installation on web server.
2. Installation on client side is not required, only a browser is required to access web applications.
3. All its Maintenance and web support is done on web server.
4. It can be accessed by one form anywhere if Internet and Browser are there. Same Web app can be used on multiple operating platforms (Cross Platform).

Difference between web applications and windows forms application?

|  |  |
| --- | --- |
| Web App | Windows Forms App |
| Installed on web server | It is installed in windows platform using windows operating system |
| It can be accessed from anywhere using web browser and internet connection | It can be accessed only from the system in which it is installed |
| It supports cross platform | It can run only on windows |
| The web app is independent of type of system | It is specific to bit, if designed for 32 bit will not run 64 bit OS |

How web app works??

Web app works on client/Server architecture. Server will convert web app pages into html so the client’s browser can understand.

On the server side the web app runs under IIS in windows platform.

Creating your first ASP.Net website?

Visual Studio provides built-in stand-alone IIS for the developers to run and test their web application without publishing to IIS.

In classic ASP we used to have .asp extension pages, in ASP.Net each ASP.Net page will be divided into three files as follows:

1. PageName.aspx (Frontend Layout File – HTML, CSS, JS) – VS Provides two types of views (Designer View, Code View)
2. PageName.cs (Code File – Backend Event Processing File)
3. PageName.designer.cs (Generates automatically by VS)

ASP.Net is event-based programming.

Controls Type:

1. Standard HTML Controls
2. ASP.Net Server Controls
   1. All the ASP.Net Controls will be converted into Standard HTML Controls by IIS before sending to the Client Browser.
   2. ASP.Net provides built-in Server Controls for developing purpose.
   3. Also 100s of third party Server Controls are available which can be used in ASP.Net.
   4. All the server controls will be executed on Server.
   5. All the Server controls provides Event Handler.
   6. Server Controls provides more functionalities as compared to the Standard HTML controls which saves development time.
   7. Server Controls preserve data across postbacks. (Retain State).

Stateless Protocol

Web Apps work on HTTP Protocol. HTTP Protocol is a **stateless protocol**, meaning it does not retain state (Posted Data) between user requests.

Get Request:

Client Send the Request (AddNewBook.asp) 🡪 Request Processed by Web Server (IIS, Apache etc.) 🡪 Create the new instance of the requested URL/Page 🡪 Process and send the New Created instance to Client 🡪 Client sees that Requested URL/Page as a New Page

ASP.net Events Handle Types

1. Cached Event (By Default) – Fired after the form Post Back like Button Click, Text Box Changed.
2. PostBack Event – Without Postback form (AutoPostBack)

Example in SimpleCounter.aspx Web Form Events

* Total 4 Events
  + 1 Event – Page Load
  + 1 Event – TextBo1 textchange
  + 1 Event – Click Button 1
  + 1 Event – Click Button 2

Client Send Request for Page SimpleCounter.aspx 🡪 IIS Will process this page. Check the Number of Events and their corresponding controls to be handle and stores these info in the ViewState 🡪 Generate HTML with the hidden ViewState which has Events handle detail 🡪Process Non Post Back Events like page load etc 🡪 Send the HTML to the clients

IsPostBack Property:

This is a Page Property and will return True if Page is Posted, otherwise Return False if Page is Not Posted.

Focus() method

onClientClick – For Java Script onclick Event

**ADO.NET – ActiveX Data Objects**

Available Framework/Methods of working with Databases in Visual Studio Application.

* ADO.Net (Microsoft)
* Entity Framework (Microsoft)
* Dapper (Third Party)
* Etc….

In Simple terms, you can think of ADO.Net as a set of classes (Framework), that can be used to interact with data sources like Databases and XML Files etc. This data can then be consumed in any .net application. (Web App, Windows Form, Console App)

XML is standard data format which is used to share data between different platform.

What are .Net Data Providers?

Databases only understand SQL. If a .net App has to retrieve data, then the application needs to

1. Connect to the Database (Build Connection String + Open Connection)
2. CRUD Operation – Prepare SQL Command
3. Execute the Command
4. Retrieve the results and display in the application

.Net has provided different classes to perform various database providers operation.

Different .Net Data Providers

* SQL Server
* Oracle
* OLEDB
* ODBC

Depending on the provider, the following ADO.net objects have a different prefix.

1. Connection (System.Data)
   1. SqlClient.SQLConnection
   2. OracleClient.OracleConnection
   3. OleDb.OleDbConnection
   4. Odbc.OdbcConnection
2. Build SQL Command
   1. SQLCommand
   2. OracleCommand
   3. OleDbCommand
   4. OdbcCommand
3. Execute Command will return Data
   1. SQLDataReader / SQLDataAdapter
   2. OracleDataReader / OracleDataAdapter
   3. OleDbDataReader / OleDbDataAdapter
   4. OdbcDataReader / OdbcDataAdapter
4. Another method to Data Store
   1. DataSet (Newer version of RecordSet) – Compatible for all data providers.

SQLConnection is ADO.Net (System.Data.SqlClient.SqlConnection)

By using this class, we can connect to our database and can open the database.

* ConnectionString (Set the SQL Server Connection Setting)
* Open (It will connect to the Database)
* Important: Always close the connection once finish the task. It will release the occupied memory used by connection object.
* Method: State – Returns the State of Connection either it’s Open or Closed.

SQLCommand

We can define SQL Commands using this Class

SqlReader

SqoCommand will execute the SQL Command and returns the data as an SQLReader Type

* SqlDataReader is Forward Only Data Type.
* Initially Record pointer will be on 1st Record if any record available.
* It will move to next record automatically whenever we read data from DataReader
* Method to Read Data (record) – Read – Returns True or False if any data is available for Read
* Property HasRows returns true or false if reader object has data.
* After Read need to get columns value – ReaderReference(“FieldName”) or (ColumnIndex)

SqlConnection

* Open Connection
* Execute Command
* Get Result from Execute Command
* Execute Another Command
* Now ADO.Net will not perform SQL Command bcz First command is already associated with that open Connection.
* By default Connection will only execute One Command.
* Solution 1: Close the connection after executing command and Re-Open Connection again before executing another command. (This solution will use more resources and will slow down the App performance)
* Solution 2: Tell Ado.net that Connection will be executed multiple Commands. (Connection String)
* using Block – Close the connection automatically.

Fill Drop Down using OOP Approach instead of Modular Programming Approach.

Application Configuration File

* web.config (In ASP.Net Web APP)
* app.config (In Windows Form App)

There are several usage of config file.

Storing Connection String in the Config File.

There are two methods

1. Store using Key / Value Pair
2. Using ConnectionStrings Block (Recommended)

SqlCommand.ExecuteReader

* Execute SQL Query stored in SqlCommand.CommandText and returns the result as SqlDatareader
* SQL Server supports multiple SQL Query Execution in one Statement like SELECT AuthorId, AuthorName FROM ListAuthor; SELECT \* FROM ListCategory;
* In the above scenario ExecuteReader will return multiple Result Sets and SqlDataReader has the ability to store Multiple Reset Sets automatically.
* SqlDataReader.NextResult will return true if any Result is available and it will move to the Next Result.

Class Author

* Method GetAllAuthors will returns the list of Author
* Add Property Count which will contains the Total Authors Count
* “SELECT AuthorID, AuthorName from ListAuthors;Select Count(\*) AS CountAuthor from ListAuthors”
* First Populate Author List from 1st Result Set
* Reader.NextResult()
* This.Count = Reader(“CountAuthro”)

SqlCommand

* ExecuteReader
  + If Result Set has Multiple Rows and Multiple Columns
  + Select LoginID, LoginEmail from LoginUsers where LoginEmail=”abc” – Returns Single Row and 2 Columns
  + Return SqlDataReader
* ExecuteNonQuery
  + If want to perform INSERT, DELETE, UPDATE Query
  + Returns Number of Rows Affected. If >0 Means task completed successfully
  + “Update LoginUsers Set LoggedDateTime =’ 09/23/2021 10:20:15’ Where LoginId=1”
* ExecuteScalar
  + If Result Set has Single Row and Single Column.
  + Select Count(\*) from ListAuthors
  + Select LoginID from LoginUsers where ……….
  + Return Single Object

Books

1. BooksModel – Fields
2. Books – Methods
   1. Add(BooksModel book) – Responsible to Insert New Book into SQL DB
   2. Add Method Validation
   3. After Validation Insert Values into SQL DB

Web form –

* AddNewBook.aspx . All Inputs for Add New Book will be in the Form Controls.
* Save Button – Validation - Store Forms values into BooksModel Intance and Pass Instance into Book.Add () Method
* Validation on Form Server Side.
* Validation on Client Side using JS

**Data Binding**

Beauty of ADO.Net. It saves lots of developer time to populate data into Web/Form Controls. Data Bind automatically populate Data with Web/Form Controls.

Most of the Controls has Data Source and Data Member Properties.

Bind Dropdown Control with SqlDataReader

ListAuthor.DataTextField = "AuthorName";

ListAuthor.DataValueField = "AuthorId";

ListAuthor.DataSource = ReaderAuthor; //SqlDataReader Object

ListAuthor.DataBind(); //Populate - Iterate DataSource Property Object

Bind Dropdown Control with Custom Type (Class)

Author author = new Author();

List<AuthorModel> ListAuthor = new List<AuthorModel>();

ListAuthor = author.GetAllAuthors();

this.ListAuthor.DataSource = ListAuthor;

this.ListAuthor.DataTextField = "AuthorName";

this.ListAuthor.DataValueField = "AuthorId";

this.ListAuthor.DataBind();

Bind Dropdown control with Custom Type (Class) Return SqlDataReader

Author author = new Author();

this.ListAuthor.DataSource = author.GetAllAuthors2();

this.ListAuthor.DataTextField = "AuthorName";

this.ListAuthor.DataValueField = "AuthorId";

this.ListAuthor.DataBind();

How to repeat Web Form Section in others Web Forms like Header, Footer etc.

1. Master Pages (Discuss Later)
2. Active Server Control Extension (ASCX File) similar to Web Form (ASPX) file.

Sample LMS Project

* ASP.Net
  + Class Library which performs all the CRUD Operation
* WinForm
  + Class Library which performs all the CRUD Operation
* Currently in both project Class Library is exactly same and performs the same operation.
* It’s not a good practice to create Class libraries multiple times in multiple projects.
* The best solution is to create one Common Class Library and use this library in any type of project.

Types of ADO.Net Architecture

1. Connected
   1. sqlDataReader is Connection Oriented Architecture
   2. Forward Only
   3. Read Only
   4. Application issues query then reads back results and process them.
   5. Always requires OPEN CONNECTION.
   6. Update, Delete, Read and Select Operation can be performed as the data is accessed in the database, so that a connection must be maintained. This can cause traffic problems.
2. Disconnected
   1. Application issues query then retrieves from DB, and stores results for processing.
   2. Move forward and backward
   3. Not a Read Only, means Result can be updated.
   4. Stored Result can be deleted, edited, inserted, select (CRUD Operations) can be performed.
   5. After all the operations, Stored Result can be directly updated with DB.
   6. Minimizes time connected to database.
   7. sqlDataAdapter and DataSet Objects. DataSet is an enhanced version of RecordSet (Asp Classic)

Disconnected Architecture Flow

* Connection
* Command
* SqlDataAdapter
* DataSet/DataTable

DataSet vs DataTable

* DataTable similar to Table in any DB means collection of rows and columns.
* DataSet similar to DB, means collection of Tables, Views & Relation.

**SQLDataAdapter**

* SqlDataAdapter fetch the Resultset from the DB and stores in DataSet/DataTable.

DataSet 🡪 DataTables 🡪 DataRow 🡪 DataColumn

Scope Identity:

INSERT INTO LoginUsersDetail (LoginEmail, UserFullName, LoginPassword, UserRoleId, IsActive) Values('xyzb@xyz.com','xyaz','xyza123',2,0);SELECT SCOPE\_IDENTITY()

public Object AddLogin2()

{

string CS = ConfigurationManager.ConnectionStrings["SqlDbCS"].ConnectionString;

using (SqlConnection conn = new SqlConnection(CS))

{

using (SqlCommand cmd = new SqlCommand())

{

cmd.CommandText = "INSERT INTO LoginUsersDetail (LoginEmail, UserFullName, LoginPassword, UserRoleId, IsActive) Values('Abcxyz@xyz.com','xyz','xyz123',2,0);SELECT SCOPE\_IDENTITY()";

cmd.CommandType = System.Data.CommandType.Text;

cmd.Connection = conn;

//cmd.CommandText = "spInsertLogin";

//cmd.CommandType = System.Data.CommandType.StoredProcedure;

//cmd.Connection = conn;

//cmd.Parameters.AddWithValue("@pLoginEmail", login.LoginEmail);

//cmd.Parameters.AddWithValue("@pUserName", login.UserFullName);

//cmd.Parameters.AddWithValue("@pLoginPassword", login.LoginPassword);

//cmd.Parameters.AddWithValue("@pRoleId", login.RoleId);

//cmd.Parameters.AddWithValue("@pIsActive", login.IsActive);

if (conn.State == System.Data.ConnectionState.Closed)

{

conn.Open();

}

Object obj = cmd.ExecuteScalar();

//int RowsAffected = (int)(obj);

return obj;

}

}

}

* BulletedList (Same as other List Controls like Dropdown, ListBox, Check etc).
  + Bullet Style Property – Set Bullet Style
  + FirstBulletNumber
  + DisplayMode (Text, HyperLink, LinkButton). Default Text.
    - Hyperlink (value must be url)
    - Target – Use to control Link Page Location
  + LinkButton (Item Selection)
* ASP.net List Controls (ListItem Object Collection)
  + DropdownList
  + CheckBoxList
  + RadioButtonList
  + ListBox
  + BulletedList
* Server.MapPath Method
  + This method returns the physical path for a given virtual path.
* Response.Write("<br><br>");
* Response.Write(". (Current Folder) returns = " + Server.MapPath("."));
* Response.Write("<br><br>");
* Response.Write(".. (Parent Folder) returns = " + Server.MapPath(".."));
* Response.Write("<br><br>");
* Response.Write("../.. (Parent of Parent Folder) returns = " + Server.MapPath("../.."));
* Response.Write("<br><br>");
* Response.Write("~ (Root Path) returns = " + Server.MapPath("~"));
* XML File
* XML is a software- and hardware-independent tool for storing and transporting data.
  + Earlier CSV format used to exchange Data. Now a days data can be exchange in XML and JSON format. (Platform independent)
  + Similar as HTML means it uses TAGS.

<?xml version="1.0"?>

<catalog>

<book id="bk101">

<author>Gambardella, Matthew</author>

<title>XML Developer's Guide</title>

<genre>Computer</genre>

<price>44.95</price>

<publish\_date>2000-10-01</publish\_date>

<description>An in-depth look at creating applications

with XML.</description>

</book>

<book id="bk102">

<author>Ralls, Kim</author>

<title>Midnight Rain</title>

<genre>Fantasy</genre>

<price>5.95</price>

<publish\_date>2000-12-16</publish\_date>

<description>A former architect battles corporate zombies,

an evil sorceress, and her own childhood to become queen

of the world.</description>

</book>

<book id="bk103">

<author>Corets, Eva</author>

<title>Maeve Ascendant</title>

<genre>Fantasy</genre>

<price>5.95</price>

<publish\_date>2000-11-17</publish\_date>

<description>After the collapse of a nanotechnology

society in England, the young survivors lay the

foundation for a new society.</description>

</book>

<book id="bk104">

<author>Corets, Eva</author>

<title>Oberon's Legacy</title>

<genre>Fantasy</genre>

<price>5.95</price>

<publish\_date>2001-03-10</publish\_date>

<description>In post-apocalypse England, the mysterious

agent known only as Oberon helps to create a new life

for the inhabitants of London. Sequel to Maeve

Ascendant.</description>

</book>

</catalog>

* **FileUpload Control**
  + Normally we use two techniques to store Files after posting file.
    - Store into DB
      * File stores as binary
      * Disadvantage: Lets suppose we do have Student Profile Table which contains 15 columns for Profile (Size: 500 Bytes), Additional we do have a column to store Picture (Max. Limit 1 MB)
    - Store into Web Server Virtual Directory (Copy file into Web Server Directory) host/appname/uploads/<file>

In Windows App, store files into a Physical Directory. (Network Folder(multi-user) or Local Folder(single user)).

* + - * Use FilePath column in DB, and store the Virutal/Absolute Path in that column.
  + SaveAs Method – Save the uploaded files into the given path.
  + FileName - Returns Uploaded File Name
  + HasFile - Returns True/False
  + PostedFile.ContentLength - Returns File Size in Bytes
  + PostedFiles – returns collection of files selected
* Adrotator Control (Display Advertisement Banner Randomly – Static one at a time)
  + Works with XML and Database
  + Ad Attributes
    - ImageUrl nvarchar
    - NavigateUrl nvarchar
    - AlternateText nvarchar
    - Keyword (Filter ad data) nvarchar
    - Impressions (likelihood/priority for ad) int
    - Width int
    - Height int
  + Table in SQL
    - ID - int Primary
* **Calendar Control**
  + Used for selecting Date rather than typing in the text box.
  + It will only show the Calendar, requires any control like Textbox, Label etc. to store the selected date from the calendar.
  + HTML5 DateTime Input type, cannot be customize

|  |  |
| --- | --- |
| Format | Result |
| DateTime.Now.ToString("MM/dd/yyyy") | 05/29/2015 |
| DateTime.Now.ToString("dddd, dd MMMM yyyy") | Friday, 29 May 2015 |
| DateTime.Now.ToString("dddd, dd MMMM yyyy") | Friday, 29 May 2015 05:50 |
| DateTime.Now.ToString("dddd, dd MMMM yyyy") | Friday, 29 May 2015 05:50 AM |
| DateTime.Now.ToString("dddd, dd MMMM yyyy") | Friday, 29 May 2015 5:50 |
| DateTime.Now.ToString("dddd, dd MMMM yyyy") | Friday, 29 May 2015 5:50 AM |
| DateTime.Now.ToString("dddd, dd MMMM yyyy HH:mm:ss") | Friday, 29 May 2015 05:50:06 |
| DateTime.Now.ToString("MM/dd/yyyy HH:mm") | 05/29/2015 05:50 |
| DateTime.Now.ToString("MM/dd/yyyy hh:mm tt") | 05/29/2015 05:50 AM |
| DateTime.Now.ToString("MM/dd/yyyy H:mm") | 05/29/2015 5:50 |
| DateTime.Now.ToString("MM/dd/yyyy h:mm tt") | 05/29/2015 5:50 AM |
| DateTime.Now.ToString("MM/dd/yyyy HH:mm:ss") | 05/29/2015 05:50:06 |
| DateTime.Now.ToString("MMMM dd") | May 29 |
| DateTime.Now.ToString("yyyy’-‘MM’-‘dd’T’HH’:’mm’:’ss.fffffffK") | 2015-05-16T05:50:06.7199222-04:00 |
| DateTime.Now.ToString("ddd, dd MMM yyy HH’:’mm’:’ss ‘GMT’") | Fri, 16 May 2015 05:50:06 GMT |
| DateTime.Now.ToString("yyyy’-‘MM’-‘dd’T’HH’:’mm’:’ss") | 2015-05-16T05:50:06 |
| DateTime.Now.ToString("HH:mm") | 05:50 |
| DateTime.Now.ToString("hh:mm tt") | 05:50 AM |
| DateTime.Now.ToString("H:mm") | 5:50 |
| DateTime.Now.ToString("h:mm tt") | 5:50 AM |
| DateTime.Now.ToString("HH:mm:ss") | 05:50:06 |
| DateTime.Now.ToString("yyyy MMMM") | 2015 May |

1. **d** -> Represents the day of the month as a number from 1 through 31.
2. **dd** -> Represents the day of the month as a number from 01 through 31.
3. **ddd**-> Represents the abbreviated name of the day (Mon, Tues, Wed, etc).
4. **dddd**-> Represents the full name of the day (Monday, Tuesday, etc).
5. **h**-> 12-hour clock hour (e.g. 4).
6. **hh**-> 12-hour clock, with a leading 0 (e.g. 06)
7. **H**-> 24-hour clock hour (e.g. 15)
8. **HH**-> 24-hour clock hour, with a leading 0 (e.g. 22)
9. **m**-> Minutes
10. **mm**-> Minutes with a leading zero
11. **M**-> Month number(eg.3)
12. **MM**-> Month number with leading zero(eg.04)
13. **MMM**-> Abbreviated Month Name (e.g. Dec)
14. **MMMM**-> Full month name (e.g. December)
15. **s**-> Seconds
16. **ss**-> Seconds with leading zero
17. **t**-> Abbreviated AM / PM (e.g. A or P)
18. **tt**-> AM / PM (e.g. AM or PM
19. **y**-> Year, no leading zero (e.g. 2015 would be 15)
20. **yy**-> Year, leading zero (e.g. 2015 would be 015)
21. **yyy**-> Year, (e.g. 2015)
22. **yyyy**-> Year, (e.g. 2015)
23. **K**-> Represents the time zone information of a date and time value (e.g. +05:00)
24. **z**-> With DateTime values represents the signed offset of the local operating system's time zone from  
      
    Coordinated Universal Time (UTC), measured in hours. (e.g. +6)
25. **zz**-> As z, but with leading zero (e.g. +06)
26. **zzz**-> With DateTime values represents the signed offset of the local operating system's time zone from UTC, measured in hours and minutes. (e.g. +06:00)
27. **f**-> Represents the most significant digit of the seconds' fraction; that is, it represents the tenths of a second in a date and time value.
28. **ff**-> Represents the two most significant digits of the seconds' fraction in date and time
29. **fff**-> Represents the three most significant digits of the seconds' fraction; that is, it represents the milliseconds in a date and time value.
30. **ffff**-> Represents the four most significant digits of the seconds' fraction; that is, it represents the ten-thousandths of a second in a date and time value. While it is possible to display the ten-thousandths of a second component of a time value, that value may not be meaningful.
31. **fffff**-> Represents the five most significant digits of the seconds' fraction; that is, it represents the hundred-thousandths of a second in a date and time value.
32. **ffffff**-> Represents the six most significant digits of the seconds' fraction; that is, it represents the millionths of a second in a date and time value.
33. **fffffff**-> Represents the seven most significant digits of the second's fraction; that is, it represents the ten-millionths of a second in a date and time value.

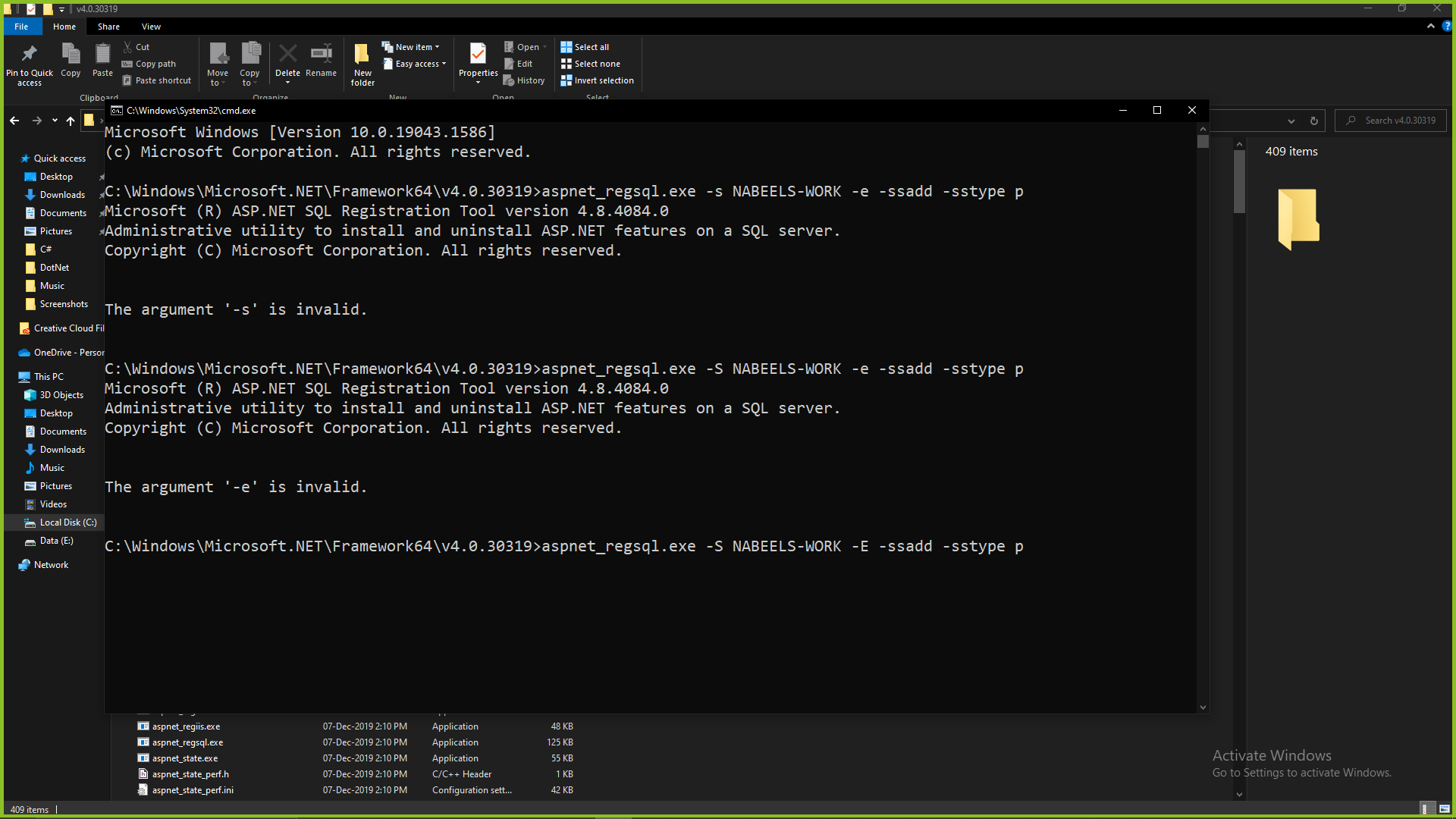
* **Hidden Field**
  + Same as HTML Input Type Hidden
  + It is a server control, it can be treated same as Textbox.
  + Data will be store as plain text. Note: Don’t store any confidential data in Hidden Field.
  + Hidden field control can be accessible by Java Script using its id name.
  + It will retain it’s value on post back.
* **Multiview**
  + Student Profile table has 100 columns, 100 controls will not be good to place in one web page due to its readability issue.
  + To solve this problem, we can create multiple web pages (each page has 15 columns) and we can have the navigation control to move those pages.
  + Disadvantage: Data handling across multiple pages. For .eg. once you move to the 2nd page, you need to store all data in page 1 and take it to the page 2.
  + To resolve this issue, Asp.net introduces Multiview. In Multiview you can place all the 100 columns control in one page but you can create sections in multiveiw. Only one section at a time will be visible.
* **Wizard Control in ASP.Net**
  + Same as Multi-View but has more advance functions like built-in-Navigation Buttons.
  + Properties
    - DisplaySideBar (Enable/Disable Left Side Bar)
    - ActiveStepIndex – Used to set or get the ActiveStepIndex of the Wizard Control.
    - DisplayCancelButton – By Default Off
    - CancelButtonImageUrl
    - CancelButtonStyle
    - CancelButtonText
    - CancelButtonType
    - CancelDestinationPageUrl
    - Finish,Next,Previous,Complete Buttons Properties
    - HeaderStyle
    - NavigationButtonStyle
    - NavigationStyle
    - SideBarButtonStyle
    - SideBarStyle
  + Events
    - Next
    - Previous
    - Finish
    - Cancel
    - ActiveStepChanged
* **Container (**It can be useful to group controls/Create Sections)
  + **Panel** (Panel uses DIV Element) <div><img>…….</div>
  + **Placeholder** (Uses nothing) <img>….
* **Literal**
  + Same as Label, but in Literal no styles can be applied. Literal Control rendered as plain text but Label control rendered as <Span> Element.
* **Adding Controls Dynamically**
  + Dynamically controls can only be added in the container.
  + Always added in the end of container.
  + Panel, Placeholder, Form, TD etc. container can also be used.
  + Dynamic created control cannot be access directly in the code bcz it’s not available at design time.
  + Later we will discuss how to access dynamically created controls and raise events.
* **Validation Controls.**
  + Controls can be used to validate Input Data.
  + ASP.Net Framework provides 6 built-in-validation controls.
  + These controls can be used to perform both client side and server side validation.
  + Developer used to write JS to perform Client-Side validation, but ASP.net validation controls automatically generate JS.
  + Client side validation is not enough because user can disable JS on the browsers so it’s a good practice to have Server Side also.
  + Set Property Page.Isvalid to true if all the validation has been passed in client and server.
* **RequiredField Validator**
  + It is used to validate NULL values.
  + Properties
    - Control To Validate
    - Error Message
    - Initial Value (Initial value to be considered as NULL)
* **RangeValidator (Validate Range)**
  + Property
    - Type (Data Type)
    - MinimumValue
    - MaximumValue
    - ErrorMessage
    - ContorlToValidate
    - Display (Static: will occupy the space, Dynamic: will not occupy the space)
* **CompareValidator**
  + Property
    - ControlToCompare
    - ValueToCompare
    - ControlToValidate
    - Type
    - Operator (=,>…)
    - ErrorMessage
    - Display
* **RegularExpression**
  + Property
    - ControlToValidate
    - ValidationExpression
    - ErrorMessage
    - Enable Client Script (by Default true, false mean it will not generate client side js).
    - Text (If this property has a value, ErrorMessage property will be disable)
  + **ValidationSumary (It will display all error message in one place)**
    - HeaderText
    - ShowSummary (True/False)
    - ShowMessageBox
    - DisplayMode (List, BulletList, SingleParagraph)
* ValidationGroup
  + If we have more than one Submit Button, .Net will perform all validation controls and display error messages.
  + TO avoid this issue, we need to group validation Controls.
  + All Validation Control has a property: ValidationGroup, we can set any unique name as well as in the Submit Buttons.
* **CustomValidator**
  + CustomValidator can be used to write our own method which can validate our inputs.
* **Different Page Navigation Techniques in ASP.Net**
  + Hyperlink Control
    - Same as Anchor Tag in HTML
    - Doesn’t generate any server side events.
  + Response.Redirect
    - Same as Hyperlink (Anchor Tag)
    - It can be handled on any Server Side event.
    - Request/Response Cycle: Example: Button 🡪 Click Event 🡪 use Response.redirect(“”)
      * Client 🡪Server (Postback)
      * Server 🡪 Client 🡪 (Response Header)
      * Client 🡪 Server (Get Request for the new page)
      * Server 🡪 Client (Web Server Response for new page)
    - Can be used to navigate Local/Virtual and External Web sites.
    - Form Data is not preserved while response.redirect.
    - Response.redirect shows the destination page url in the location bar.
    - It maintains browser history.
  + Server.Transfer
    - Cycle: Server.Transfer(“”)
      * Client 🡪Server (Postback)
      * Server 🡪 Client (Web Server Response for new page)
    - Faster than Response.Redirect
    - Maintain browser History
    - It will not show the new Page URL.
    - Preserves Form Data.
      * Method 01 - to retrieve submitted form values using System.Collections.Specialized.NameValueCollection and Request.Form.
      * Method 02 - to retrieve submitted form values using previous page property - Page.PreviousPage
    - It will not redirect to external URL.
  + Server.Execute
    - Server.Execute and Server.Transfer are almost same.
    - Cycle – One Way
    - Preserves Form Data
    - URL will not show.
    - The only difference is New Page will be rendered after the Current Page.
  + Cross-Page-postback
    - By default ASP.Net Form posts to itself.
    - Cross-Page-postback means, posts the form to another page.
    - Preserves Forms Values.
    - Reading form values using Page.Previous Page property, need to use FindControl Method. If we misspell the control name then runtime exception will through. To resolve this better to use Strongly Typed Reference.
    - Strongly Type Reference generates error in compile time.
    - To Obtain Strong Typed Reference, first method is to create Properties of your controls what you want to expose.
  + Window.Open (Using Java Script)
* **Send data from one page to another page**
  + Request.Form (Form Post)
  + Page.PreviousPage Property (Form Post)
  + Cross Page Postback.
  + QueryString
    - QS are name/value collection pairs
    - Using QS, is a very common way to send data from one web form to another.
    - QS are appended to the page URL.
    - ? sign indicates the beginning of QS and its value.
    - Subsequent Qs can be appended to the URL using the & sign.
    - There is a limit on the QS length, Hence, QS cannot be used to send very long data.
    - QS are visible to the user
    - Hence should not be used to send sensitive information unless encrypted
    - To Read the QS, use Requst.Querystring Property
    - If values contains & sign, then QS values will be compromised. To resolve this issue, Asp.net provides Server.UrlEncode() method to property encode the URL.
    - Second option, is to replace the & sign with HTML Code. HTML Code of & sgn is %26.
    - .Net provides lots of Encryption Algorithm
  + Cookies
    - Cookies can also be used to send data from one webform to another.
    - Cookies can be used to store user preferences or other information that is client-specific.
    - Cookies will be stored in client computer/device.
    - Cookies must be enabled in client device to store.
    - Cookies types
      * Persistent cookies – Remain on the computer, even after the browser is closed. But you can configure how long the cookies remain using the expires property.
      * Non-Persistent cookies – if you don’t set the expires property, then the cookie is called as Non-Persistent cookie. Non-Persistent cookies only remain in memory until the browser is closed.
    - How to check if cookies are enabled or disabled.
      * Browser supports cookies (Yes/No)
        + Request.Browser.Cookie – Property Returns True/False
      * Cookie Enabled (Yes/No) (In Yes/No, Browser supports must be Yes)
        + Write a Temp. Cookie
        + Redirect to the same page
        + Read Temp Cookie
        + If cookie present means Cookie is Enabled
        + Else Cookie is Disabled.
  + Session State
    - Session variables can also be used to send data from one page to another.
    - Session State Variables are stored on web server by default and are kept for the life time of a session.
    - Session Variables are client specific means each user will have it’s own Session State.
    - Session State will remove from the server after the closure of client browser.
    - All Session State can also be removed if Web server or the process where IIS stored Session State is crashed or restarted.
    - The default session state mode is InProc.
    - The life time of a session is determined by the timeout value. By default it’s 20 minutes. In Every 20 minutes IIS automatically clears all the session state variables.
      * Application Level Set Timeout in web config.file.
* <sessionState mode="InProc" timeout="10"></sessionState>
  + - Session State variables are available across all pages in a single Application, but only for a given single session. Session variables are like single user global data.
    - It is always a good practice to check, if a session state variable is null before accessing or calling any methods such as .ToString() on Session Objects, otherwise it may run into runtime Null Exceptions.
    - Application Performance can be improved by disabling Session State, if its not required.
      * Page Level
* <%@ Page Language="C#" AutoEventWireup="true" CodeBehind="C34\_SessionState.aspx.cs" Inherits="CsharpWebAppDemo.C34\_SessionState" EnableSessionState="False"%>
  + - * Application Level – in web .config file under <system.web>
        + <sessionState mode="Off"></sessionState>
    - Cookie Less Sessions
      * By Default Sessions use Cookies.
      * In Cookies, ASP.Net stores Client Session-id. (Unique Id)
      * Client (Client Session Id) 🡪 Server (Data + Client Session Id)
      * This Session-id, is then used by the web server to identify if the request if coming from the same user or different user.
      * Lets suppose, client has disabled the cookies, it means session variables cannot be created.
      * To avoid this problem, Asp.Net provides cookie less Session.
      * Set cookieless attribute to true in web.config file.
        + <SessionState mode=”InProc” cookieless=”true”>
      * For CookieLess sessions to work property, only relative URLs must be used in the application to navigate from one page to another.
        + Relative URL: response.redirect(“WebForm2.aspx”)  
          response.redirect(“~/Data/Webform2.aspx”)
        + Absolute URL: response.redirect(“http://localhost/Data/Webform2.aspx”)
    - Session State Mode: (Session Values stored in Web Server by default). By using mode developer can decide where ASP.Net will store the Session Objects.
      * Off (Session State will be off in the entire application)
      * InProc (Default)
        + Session State Memory is kept on the web server within the ASP.net worker process (w3wp.exe)
        + Very easy to implement
        + Dis: Session Data can be lost if Work Process (W3wp.ee) crashed or failure.
        + Good for Single server hosting.
        + Dis: Not suitable for Web Farms and Web Gardens.
        + Example: If any application hosted on multiple servers (Load Balancing), then InProc Mode will not work properly.
      * StateServer
        + The session state variables are stored in a process, called as asp.net state service (Service Program which runs in Windows Service).
        + This process is different from the asp.net worker process.
        + The asp.net state service can be present on a web server or a dedicated machine.
        + Web Farm and Web Garden can be supported.
        + Dis: Slower than inProc
        + Dis: If the StateServer machine or the service goes down, all the sessions are lost.
        + How to Configure:

StateServer Service must be on under windows services (ASP.Net State Service)

Web Config – Define Mode & StateService Server.

<sessionState mode=StateServer” stateConnectionString=”tcpip=localhost:42424” timeout=”20”></sessionState>

Localhost can be replaced with any server name or IP Address.

* + - * SQLServer (To store all the sessions values in SQLServer DB) – Process & service independent.
        + How to configure:
        + 
    - **Web Config.**

<!--below connection string is of windows authentication-->

<!--<sessionState mode="SQLServer" sqlConnectionString="data source=NABEELS-WORK;integrated security=SSPI"></sessionState>-->

<!--connection string using sql server authentication-->

* + - <sessionState mode="SQLServer" sqlConnectionString="Server=NABEELS-WORK;User Id=Demo;Password=Nabeel30;MultipleActiveResultSets=True;"></sessionState>
    - SQLServer is the most reliable option.
    - Can be used with web farms and web gardens.
    - More scalable than InProc and State Server
    - Dis: Slower than InProc and StateServer
  + Application State
    - Application state variables are available across all pages and across all sessions.
    - Application State variables are like multi-user global data.
    - Application state variables are stored on the web server.
    - Application state variables are cleared, only when the process hosting application is restarted, that is when the application ends.
    - Application state variables are not shared across a web farm or a web garden.
    - Application state variables are not thread safe, Lock and Unlock methods of the application class must be used to protect again race conditions, deadlocks and access violation.
* Encyption
  + EncryptBase64
  + EncryptAES
  + EncryptHash
  + DecryptBase64
  + DecryptAES
  + DecryptHash
  + Encrypt - Use it all the project
    - Use any Alogorithm
  + Decrypt
* **Master & Content Pages**
  + Master pages can be used to have a same design in all the pages so developer doesn’t need to re-write the same code again and again in every Web Form. For example if all the web forms we do require the same Header and Footer so better to keep in Master Page.
  + Best Practice to design the layout of the web application so can understand what part do we require in Master Pages.
  + How to Create
    - Add Master page (Header, Footer and Main Content Layout)
    - Add ContentPlaceHolder in Master Page Main Content Area
    - Add Content Page by right click the Master Page.
      * In Content Page, ContentPlaceHolder Section design your normal web form.
* **Nested Master Pages**
  + **Master Page 1**
  + **Master Page 2 from Master Page 1**
  + **Content Page for Master Page 2 (It will contain Master Page1 and Master Page 2 design)**

**Authentications in ASP.Net Web Application**

* Authentication is the process of identifying users.
* Authorization is the process of granting access to those users based on identity.
* Together authentication and authorization secure our Web Application.

**Types of Authentications in ASP.Net Web Apps.**

* **Anonymous Authentication**
* **Windows Authentication**
* **Forms Authentication.**

**Anonymous Authentication**

* Anonymous Authentication allows users to access the public areas of the web site, without prompting the users for a username and password. Like YouTube, google, etc…
* By Default ASP.Net creates Anonymous Authentication Web Site.

**How to enable Anonymous Authentication**

* Visual Studio provides built-in IIS and it uses it’s built-in IIS to develop any Application.
* Above Authentications to work, need to publish /host web sites in External IIS. There are two methods to host our application in External IIS.
  + Using Visual Studio
    - Project Properties
    - Web 🡪 Local IIS Server 🡪 Create Virtual Directory.
  + Using IIS
    - Open IIS
    - New Application Under Default Web Sites
* There are two methods to enable
  + Using IIS (Internet Information Service). IIS is used to host/publish Web Sites.
    - IIS 🡪 WebApplication 🡪 Authentication 🡪 Anonymous Authentication (Enable/Disable)
  + Using Web Application Web Config File.
    - Under <system.web>

<authorization>

<allow users="?"/>

</authorization>

Here, ? = Anonymous Users

<authorization>

<deny users="?"/>

</authorization>

Here, it doesn’t allow anonymous users.

Symbols:

‘?’ = Anonymous

‘\*’ = All Users

**Windows Authentication in ASP.Net**

* Anonymous authentication is fine for web site that contains public information that everyone can see.
* However, if the web site contains private information or performs tasks such as booking tickets, payments, etc. then the users need to be authenticated and authorized.
* Windows authentication is best suited for Intranet (Local/Private Network, such as Offices etc) Web Applications.
* The advantage of Windows authentication is that the web application can use the exact same security scheme that applies to your corporate network. User names, passwords, and permissions are the same for network resources and web applications.

employee

Server

Web App Hosted

employee

employee

employee

Note: Windows authentication enabled in IIS and Anonymous authentication disabled using web.config, allow all windows users.

<authorization>

<deny users="?"/>

</authorization>

Allow Specific Windows Users for Authentication.

<authorization>

<allow users = “NABEELS-WORK\Nabeel's PC, NABEELS-WORK\hassan” />

<deny users="\*"/> //Note: Deny All users wither Anonymous or Windows

</authorization>

Note: Order of Allow and Deny is very important.

Allow/Deny Windows Group

<authorization>

<allow roles=”Administrators” />

<deny users="\*"/>

</authorization>

**Folder Level Authentication in Windows Authentication using Multiple Web Configs.**

* Root Level Web.config will apply the settings to the entire project.
* Folder Level Web.Config will apply the settings to the current folder only and will override the root level web config settings for this folder.

**Forms Authentication in ASP.net**

* **Using Web.config users list**
* **Using DB level Users list.**

**Forms Authentication in ASP.Net using Web.config users list.**

* Define Users List in Web.config
* <authentication mode="Forms">
* <forms loginUrl="Login.aspx" defaultUrl="Dashboard.aspx" timeout="30" protection="All">
* <credentials passwordFormat="Clear">
* <user name="nabeel" password="nabeel123" />
* <user name="hassaan" password="hassaan123" />
* <user name="umair" password="umair123" />
* </credentials>
* </forms>
* </authentication>
* //disable anonymous
* <authorization>
* <deny users="?" />
* </authorization>

if(FormsAuthentication.Authenticate(TextBox1.Text, TextBox2.Text))

{

//create the authentication cookie and redirect the user to the dashboard

FormsAuthentication.RedirectFromLoginPage(TextBox1.Text, CheckBox1.Checked);

//Response.Redirect("Dashboard.aspx");

}

else

{

Label1.Text = "Invalid username or password";

}

**Do not store Plain Text Passwords into DB**

* Standard: Password cannot be Decrypted
* Best way to Encrypt Password is using HASH Technology. (Using any Unique Key)

string str = FormsAuthentication.HashPasswordForStoringInConfigFile("Nabeel", "MD5");

|  |  |  |
| --- | --- | --- |
| Clear | 0 | Specifies that passwords are not encrypted. This field is constant. |
| MD5 | 2 | Specifies that passwords are encrypted using the MD5 hash algorithm. This field is constant.  Due to collision problems with MD5, Microsoft recommends a security model based on SHA256 or better. |
| SHA1 | 1 | Specifies that passwords are encrypted using the SHA1 hash algorithm. This field is constant.  Due to collision problems with SHA1, Microsoft recommends a security model based on SHA256 or better. |
| SHA256 | 3 | Specifies that passwords are encrypted using the SHA256 hash algorithm. This field is constant. |
| SHA384 | 4 | Specifies that passwords are encrypted using the SHA384 hash algorithm. This field is constant. |
| SHA512 | 5 | Specifies that passwords are encrypted using the SHA512 hash algorithm. This field is constant. |

**ASP.Net GridView**

* Populate Data in a Grid (Table Rows/Columns)

**ASP.Net Data Sources**

* **SqlDataSource (SQl Server)**
* **ObjectDataSource (Custom Class)**
* **AccessDataSource (MS-Access)**
* **XMLDataSource (XML)**
* **LinqDataSource (Linq)**
* **EntityDataSource (Entity Framework)**

Without using Data Sources, developer has to do the following to perform any CRUD Operation.

* Read Connection String
* Create Connection Object
* Create SQL Command Object
* Execute the Command
* Retrieve and bind the result to the data-bound controls like Gridview, Dropdown etc.

Using ASP.Net built-in Data Sources Controls, we don’t need to write a single line of code.

Table:

1. EmployeeID
2. FirstName
3. LastName
4. DateofBirth
5. AnnualSalary
6. Gender
7. DepartmentName

Formatting GridView Control

* Hide Column Field
  + Don’t include in your Data Source
  + Remove Column from GridView
  + Set Visible = false
* Customize Column Header
  + Use HeaderText Property
* Customize Date Field
  + <https://docs.microsoft.com/en-us/dotnet/api/system.web.ui.webcontrols.boundfield.dataformatstring?view=netframework-4.8>
  + <https://docs.microsoft.com/en-us/dotnet/standard/base-types/standard-numeric-format-strings>
* Customize Numeric Field
  + {0:c} for Currency
  + {0:c2} Currency with 2 Decimal.
  + By default, it will use the default Culture setting. To override the default culture use the following methods
    - In Web Config (Application Level)
      * <system.web>  
        <globalization culture=”en-US”/>  
        </System.Web>
    - Page Level
      * <@Page Culture =”en-GB”/>
    - Page Level Using Code
      * System.Threading.Thread.CurrentThread.CurrentCulture = new System.Globalization.CultureInfo(“en-IN”)
* Customize Gridview Column Value at runtime
  + GridView Event – RowDataBound
* Gridview Styles
  + Control Level
    - Apply to all Grid
  + Row Level
  + Alternate Row Level
  + Header
  + Footer
  + Edit Mode Level
  + Built-in-Themes
  + Run time using code
* Get Grid Cell Values
  + Using RowDataBound e.Row.Cells[3] argument
    - Disadvantage: Program will cause error if column position will change in the grid.
  + Using Column Name
    - DataBinder.Eval(e.Row.DataItem,”Salary”)
* Using SQL Stored Procedures with SqlDataSource Control.
  + Required
    - Stored Procedure in SQL
    - SqlDataSource and Configure with SP
    - GridView and configure with sqlDataSource
* Using Stored Procedures with ObjectDataSource Control
  + Requried
    - Custom Objects (Class with Methods)
    - Stored Procedures in SQL
    - ObjectDataSource and onfigure with Custom Object Class
    - GridView / DropDown / Text Box etc and configure with ObjectDataSource
* Delete data from gridview using sqldatasource control
  + Required
    - Table in SQLServer
    - Configure sqlDataSource with Delete Query
    - GridView and configure with sqlDataSource
    - Enable Delete from GridView Properties
    - Use Java Script in GridView RowData Bound Event for Client confirmation, if required.
  + Optimistic Concurrency (Conflict Detection)
    - In a Normal Case, how do we write Delete Query?
      * DELETE FROM ListAuthor where AuthorID = @PID
    - The above is not a good practice to write DELTE Query
    - Reason?
      * If one user has already opened the record, in the mean while another user has changed the same record but in the first user screen the data is not refreshed. In this case first user will not know that his data has been changed by another user.
    - To resolve this , better to use Optimistic Concurrency, means to compare all the values along with Primary Key in Delete or Update Query like
      * Delete from ListAuthor where AuthorId=@piD AND AuhthorName=pAuthorName
      * P\_{0} means original displayed valued in the ifrst user screen.

SQL Server Additional Data types

* Binary (To store Binary Data like MS-Word, Ms-Excel, Any Image etc.)
* Identity – Unique in a Table Column (Integer)
* UniqueIdentifier – 16 Byte Binary and globally unique across tables, databases and servers. Lets suppose in a DB, there are five tables which contains Primary Key as UniqueIdentifier. In this case SQL Server will generate Unique 16 Bytes Binary data for each table column.

Encrypt Password using SHA1 Encryption Method with Unique Key