Web

PAGE NO.
DATE 6/12/2024

Create new project.

C# | All Platforms | Web

ASP.NET Web Application (.NET FRAMEWORK)

NEXT

click ↓
EMPTY

CREATE

Project → Web Application
Connected Services.
properties
References
packages.config
Web.config

→ Program :-

```
<!DOCTYPE html>

<html>
<head runat = 'server'>
  <title> </title>
</head>
< body>
   <form id = "form1" runat = " server">
      <div>
      </div>
   </form>
</body>
</html>
```

→ Names

(windows)	(web)
combobox	DropDown List

Assignment :-

Label (Name)	Textbox
Label (Gender)	Radio Button
Label (City)	DropDown List
~~Label (Subject)~~	~~Checkbox~~
Label (State)	~~Li · Checklistbox~~ List Box
Label (Subject)	Checkbox

Website → Static
Application → Dynamic

PAGE No.
DATE / /

ASP.NET :-

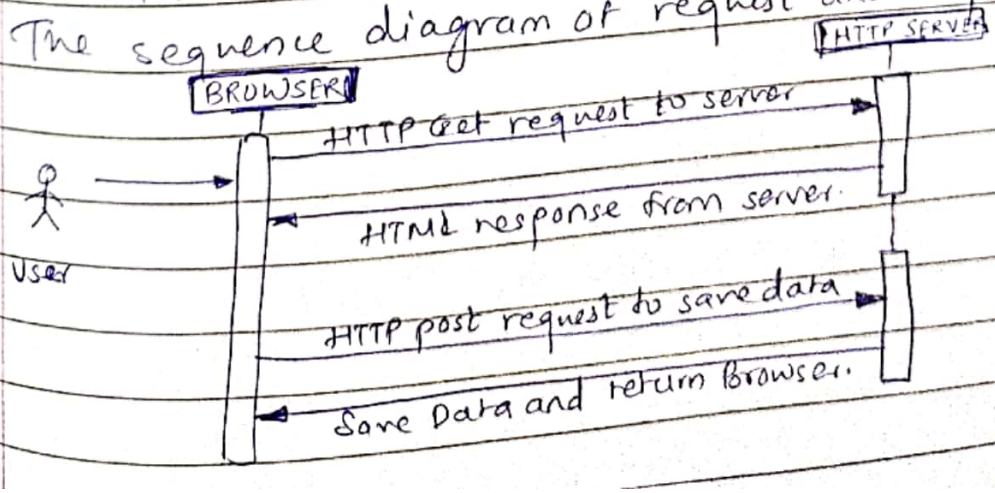
- The full form of ASP is Active Server Pages.
- ASP.NET is a web application framework developed and marketed by Microsoft to allow programmers to build dynamic web sites.
- ASP.NET was first released in the year 2002.
- ASP.NET applications can also be written in a variety of .NET languages. These include C#, VB.NET, and J#.
- ASP.NET is a high speed and low-cost programming language that is widely used to create websites and applications. It is very easy to learn and requires minimal setup and resources. Moreover, it is a widely used and very popular programming language. There are huge opportunities available for .NET programmers worldwide. Therefore, it is a very good option for beginner programmers to learn.

ASP.NET Page Life Cycle :-

How to initiate a request?

→ When you type a URL in Browser address bar and press enter, the Browser sent a page request to the server.

The sequence diagram of request and response.



1) Start (user)
2) Page Request (B→S)
3) Load Requested page in server (S)
4) Processing requested page (S)
5) Send page to the browser (S → B)
6) Unload page from the server memory (obj destroy :) (S)

Important:

```
<html>
  <head>
    <title> </title>
      <link /> ← External CSS
      <style> </style> ← Internal CSS
      <script> </script> ← Javascript fun
  </head>
  <body>
    ..
  </body>
</html>
```

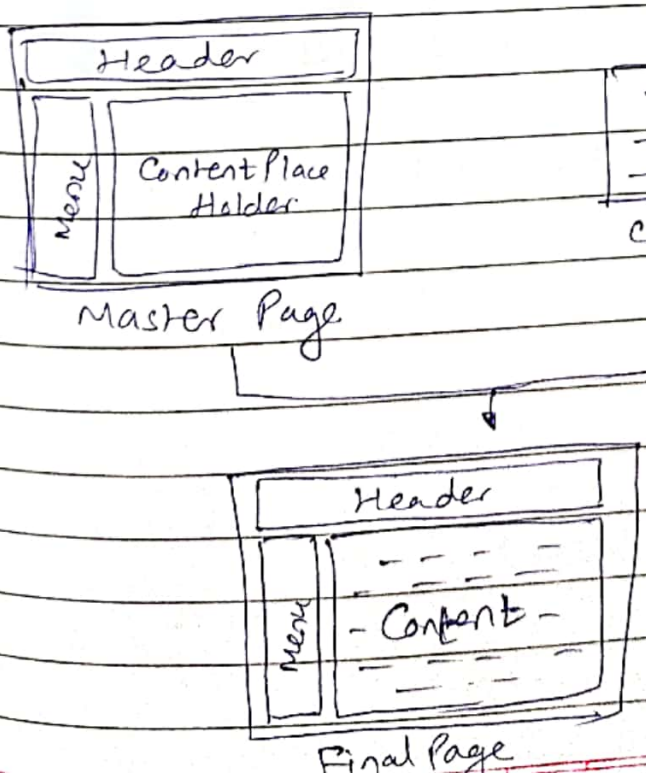
Inline CSS

```
<h1 style = " " >
  ..
</h1>
```

Master Pages:-

- Master pages allow you to create a consistent look and behaviour for all the pages (or group of pages) in your web application.
- A master page provides a template for other pages, with shared layout and functionality. The master page defines placeholders for the content, which can be overridden by content pages. The output result is a combination of the master page and the content page.
- The content pages contain the content you want to display.
- When users request the content page, ASP.NET merges the pages to produce output that combines the layout of the master page with the content of the content page.

Add → new item → Web Forms ~~Web~~ Master page



Add Content Page :-

- ↳ click on master page that we created.
  - ↳ select Add content page.

OR

- ↳ Click on Project
  - ↳ Add → new item
    - ↳ Web forms with master page
      - ↳ window pop up (Select Master Page)
        - ↳ Select the master page that you want to select
        - ↳ click OK.

- To show page.

↳ Select a .aspx. file that you want to show first
↳ Right click on that
↳ Set as start page

showing pages on menu strips click.

```
<asp:MenuItem Text="Home" Value="Home" NavigateUrl=
"~/frm_home.aspx">
</asp:MenuItem>
```

Content :

Content 1 → Head related

Content 2 → Body related.

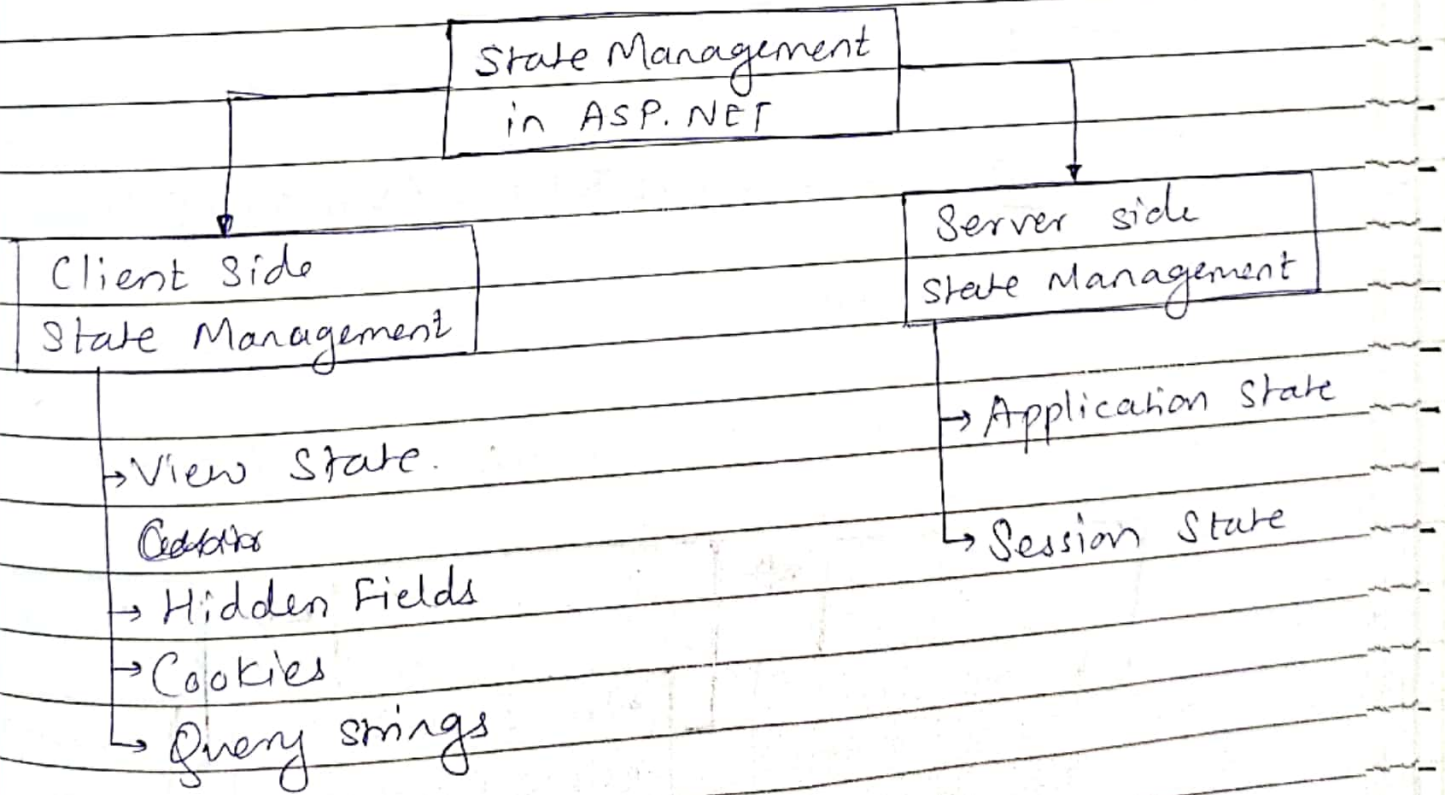
State Management Techniques :-

State management is the process by which you can maintain state and page information over multiple requests for the same or different pages.

Web applications are stateless i.e When we used request to the server then output gets displayed on browser, but after that if we send a second request then browser will not preserve the state of previous application or website.

Types of State Management

1. Client side state management
2. Server side state management



Client Side State Management :-

View State :-

- It is technique to preserve value of page and controls between round trip.

- It is used to store a small amount of value within a same page.

[20 bytes]

viewstate assign =

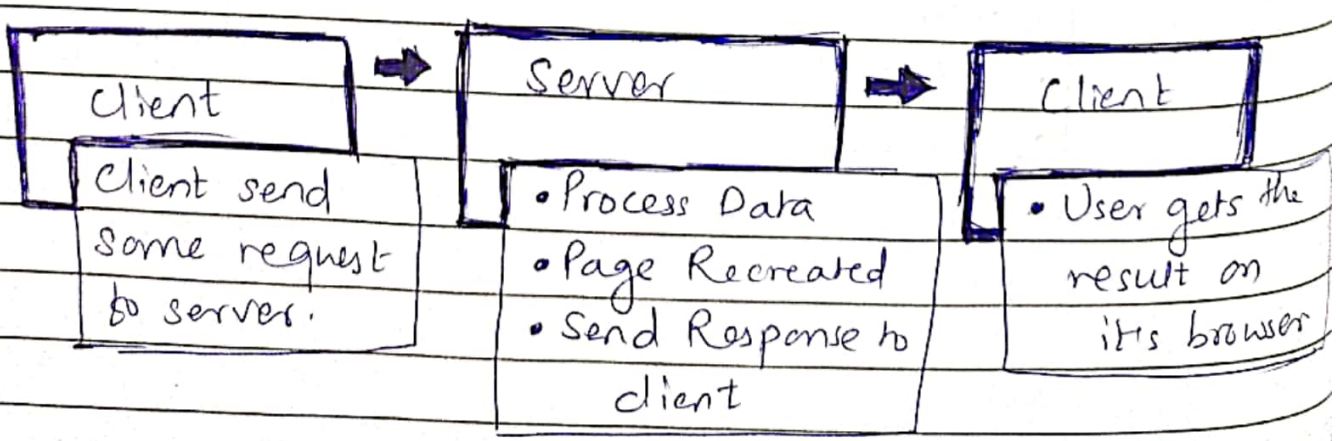
```
ViewState ["(variableName)"] = (value);
```

Access =

```
int --- = Convert.ToInt32 (ViewState ["(variable Name)"]);
```

Clear viewstate

```
ViewState ["(variable Name)"] = Null / " ";
```



Hidden Field :-

In .aspx page

```
<div>
    <asp:HiddenField ID="hf_data" runat="server" />
</div>
```

A Hidden field is used for storing small amounts of data on the client side. In most simple words it's just a container of some objects. It is invisible in the browser. To hold the data, value property should be set.

eg. Increment counter value on click

```
Hiddenfield.value = 0;
```

Hiddenfield — control name; value — Property

Query String :-

These are used for holding some value from a different page and move these values to the different page. '?' symbol is used to seperate the character strings. Not used to send large data.

(Send parameter to another page)

Set =>

```
Respons.Redirect("Webform.aspx?password=" + txt_pass.Text +
                 "&Name=" + "Sushant" + "&ID" +
                  Key       value     "ggg");
```

Access =>

```
txt_nm.Text = Request.QueryString["Name"].ToString();
txt_id.Text = Request.QueryString["ID"].ToString();
txt_pass.Text = Request.QueryString["Password"].ToString();
```

Clear =>

Cookies:-

It is used to store small amount of data on client side.
To generate cookies 'HttpCookie' class is used.

Create =>

```
HttpCookie obj = new HttpCookie("Info");

obj["Id"] = TextBox1.Text;
obj["Name"] = TextBox2.Text;
```

Set =>

```
Response.Cookies.Add(obj);
Response.Redirect("WebForm.aspx");
```

Access =>

```
HttpCookie objReq = Request.Cookies["Info"];
Label1.Text = objReq["ID"];
Label2.Text = objReq["Name"];
```

clear =>

```
objReq =
```

Server Side State Management :-

Session State Management :-

Session state variables are available across the all pages. Session state variables are stored on web server memory. To create session variables key-value format is used. Sessions are stored till the browser is open.

set ⇒

```
Session["ID"] = "100";
Response.Redirect("WebForm.aspx");
```

Access ⇒

```
txt_id.Text = Convert.ToString(Session["ID"]);
```

Clear →

```
Session["ID"] = null;
```

For DataTable :-

Set ⇒

```
DataTable dt = new DataTable();
Session["dtDetails"] = dt;
```

~~Datatable~~

PAGE No.
DATE / /

Access=>

DataTable dtEmp = Sesian["dtDetails"] as DataTable;

Application State Management :-

Application state variables are also available across all the pages.
Application variables life time or cleared only when process hosting to application is restarted (till system is on).

set=>

~~Applicatian["ID"] =~~
Application["Name"] = "ABCD";
Response.Redirect("WebForm.aspx");

Access=>

txt_name.Text = Convert.ToString(Application["Name"]);

Validation:- (It is wed client side validation.)

RequiredField Validator	Range Regular Expression Validator	Range	Compare	Summary
ID	ID	ID	ID	ID
runat	runat	runat	runat	runat
ErrorMessage	ErrorMessage	ErrorMessage	ErrorMessage	ErrorMessage
ControlToValidate	ControlToValidate	ControlToValidate	ControlToValidate	ControlToValidate
Display="Dynmic"	Display=	Display	Display	Display
ValidationGroup	ValidationGroup	ValidationGroup	ValidationGroup	ValidationGroup
ForeColor	ForeColor	ForeColor	ForeColor	
	ValidationExpression ("^[0-9]-{10}")	Type-	ControlToCompare	
		MinimumValue	ControlToValidate	
		MaximumValue		

Validation
↳ ValidationGroup = ValidationGroup ← Button
(Value) (Value)

NOTE => In web.config

```
<Configuration>
  <appSettings>
    <add key="ValidationSettings:UnobtrusiveValidationMode"
         value="None" />
  </appSettings>
  ---
  --
</Configuration>
```

dll - Dynamic link library

Ajax :-

(Asynchronous Javascript And xml)

- used for update the panel.

Create new project
↳ Take new web form
↳ Add references
↳ References Manager
↳ Browse (click)
↳ Take file (AjaxControlToolkit.dll)
↳ Click on Add.
↳ Click on OK
↳ Rebuild Solution.

ToolBox
↳ Right on Toolbox
↳ Choose Items
↳ Popup window of 'choose ToolBox Items'
↳ Click on Browse
↳ Take file (AjaxControlToolkit.dll)
↳ Click on open
↳ Click on OK
↳ Rebuild Solution.

NOTE = attribute in calenderExtender
[TargetControlID= TextBox]

(If we using external file for project)

NOTE

<%@ Register Assembly = "AjaxControlToolkit" Namespace = "AjaxControlToolkit" TagPrefix="asp" %>

```
<body>
  <form ...>
    (Mandatory) <asp:ScriptManager ID="ScriptManager1" runat="server"></asp:ScriptManager>

    <div>
      <asp:TextBox ...></asp:TextBox>
      <asp:CalenderExtender ID="CalenderExtender1"
          runat="server" TargetControlID="TextBox1">
      </asp:CalenderExtender>
    </div>
  </form>
</body>
```

.cs File.

```
txt_dt_time.Attributes.Add("ReadOnly","True");
```

Time :- (Ajax Extensions)

(Timer) <scriptmanager>

```
<div>
  <asp:Timer ID="Timer1" runat="server" onTick="Timer1_Tick"
      interval="1000"></asp:Timer>

  <asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <Triggers> (Fires Event)
      <asp:AsyncPostBackTrigger ControlID="Timer1" />
    </Triggers>
```

```
<ContentTemplate>
  <asp:Label ID="lbl_dt_time" runat="server" Text=""></asp:Label>
</ContentTemplate>
</asp:UpdatePanel>
</div>
```

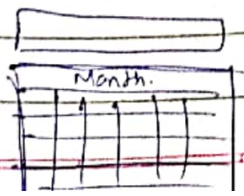
.cs

```
protected void Timer1_Tick(object sender, EventArgs e)
{
    lbl_dt_time.Text = DateTime.Now.ToString();
}
```

Calender → Button :-

```
<Register ------ >
<div>
  <asp:ScriptManager ID="ScriptManager1" runat="server">
  </asp:ScriptManager>
  <asp:TextBox ID="txt_dt" runat="server"></asp:TextBox>
  <asp:CalenderExtender runat="server"
      TargetControlID="txt_dt"
      PopupButtonID="Button1"
      Format="dd-MM-yyyy">
  </asp:CalenderExtender>
  <asp:Button ID="Button1" runat="server" Text="Calender" />
</div>
```

UI ⇒



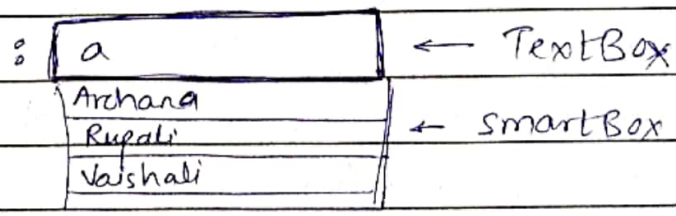
on button click display calender.

```
protected void Page_load(---)
{
    txt_dt.Attributes.Add("Readonly", "true");
}
```

Web Service

Auto Complete Extender :-

```
<asp:ScriptManager ---->
<div>
  <asp:TextBox ID="txt_name" runat="server">
                                    </asp:TextBox>
  <div id="div_name_list"></div>
  <asp:AutoCompleteExtender
      ID="AutoCompleteExtender1"
      runat="server"
      TargetControlID="txt_name"
      ServicePath="~/SmartWebService.asmx"
      ServiceMethod="WebUserName"
      MinimumPrefixLength="1"
      CompletionListElementID="div_name_list">
  </asp:AutoCompleteExtender>
</div>
```



Add web service file.

Right click on project
↳ Add
↳ New item
↳ Web service (ASMX)
↳ Rename

After opening the file

- To allow this web service to be called from script, using ASP.NET AJAX, uncomment the following line.

[System.Web.Script.Services.ScriptService]

cs.file ⇒

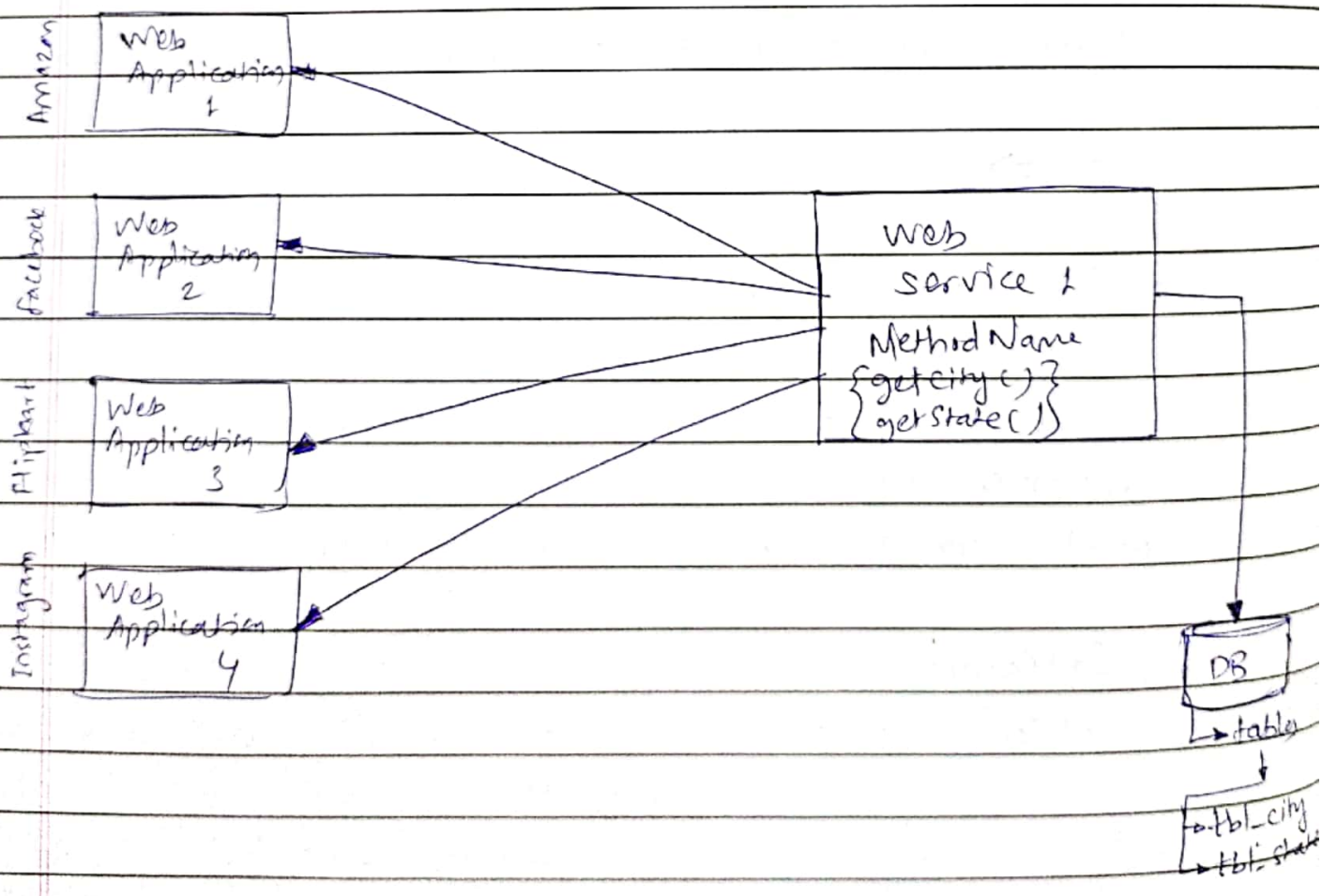
public class SmartWebService : ...

// web service for autosuggest text property
[WebMethod]
public string[] webUserName (string prefixText)
{
SqlConnection conn = ...
SqlCommand cmd = new SqlCommand ("SELECT studName
FROM tbl_student_details where
studName like "%" + prefixText + "%", conn)

cmd.Connection.Open();

```
SqlDataReader sdr = cmd.ExecuteReader();
List<string> fname = new List<string>();
while (sdr.Read())
{
    fname.Add(sdr["studName"].ToString());
}
cmd.Connection.Close();
return fname.ToArray();
}
```

WebService.



Linq

PAGE NO.
DATE / /

tools
↳ Get tools and features.
↳ Individual components
↳ Linq to SQL tools.
↳ Download All, then install.

LINQ :-

LINQ that stands for Language Integrated Query (pronounced as 'Link') is a .NET language extension that supports data retrieval from different data sources like XML document, databases and ~~collec~~ collections. It was introduced in the .NET 3.5 framework. VB and C# are the languages that have LINQ capabilities.

Add
↳ New item
↳ LinQ to SQL Classes.
↳ dbml file.

Start of the program

```
using system.Linq;

public partial class Webform1 : _
{
    protected void Page_load ( )
    {
        DataClasses2DataContext objDC = new DataClasses2Data
                                                    Context();
        GridView1.DataSource = from objtbl in objDC.tbl_emps
                               where objtbl.EMPNO == 3
                               Select objtbl ;

        GridView1.DataBind ();
    }
}
```

→ Select new { userName = objtbl.ENAME, no = objtbl.EMPNO }

#

```
Protected void Page_load ()
{
    var nameList = objDC.tbl_emps.Select (x => new
                        { studName = x.ENAME }).ToList();
    GridView1.DataSource = nameList;
    GridView1.DataBind ();
}
```

Tab Container:-

```
<asp: ScriptManager - - /> </asp: ScriptManager>
<div>
  <asp : Tab Container ID = "TabContainer1" runat = "server"
AutoPostBack: "True" onActiveTabChanged = " ". ActiveTabIndex = "1">
    <asp : TabPanel TabIndex = "0" runat = "server">
      <Header Template > List </HeaderTemplate>
      < ContentTemplate>
        <asp: Label ID= "Label1" runat = 'server"
                    Text = "Employee List' > </asp: label>
        <asp:GridView ID = "GridView1" runat = "server">
                    </asp: GridView>
      </ContentTemplate>
    </asp : TabPanel >

    <asp: TabPanel TabIndex: "1" HeaderText = " Add"
      runat = "server" >
      <Header Template > Add </HeaderTemplate>
      < Content Template>
        <asp: Label ID= "Label2" runat = "server"
                    Text = "Employee Details:">
                    </asp : Label>

        <asp: Label ID = "Label3" runat = "server"
                    Text: " Employee Name": >
                    </asp : label>

        < asp: TextBox ID = "TextBox1" runat = "server">
                    </asp : TextBox>

      </content Template>
    </asp: TabPanel>
  </asp: Tab Container>
</div>
```

.cs File

```
protected void Page_load()
{
    TabContainer1.ActiveTabIndex = 1;
}

protected void TabContainer1_ActiveTabChanged()
{
    if (TabContainer1.ActiveTabIndex == 0)
    {
        TextBox1.Text = "";
    }
            OR
    if (TabContainer1.ActiveTab.HeaderText == "Add")
    {
        TextBox2.Text = "";
    }
}
```

ghya samjun hot kahitari

Footer Template:-

<asp:GridView ---- ShowHeaderWhenEmpty="true" ShowFooter="true">
<EditTemplate>
...
</EditTemplate>

<FooterTemplate>
<asp:Button ID="btn_add" runat="" OnClick="btn_add_click" />

</FooterTemplate>

LINQ :-

LINQ that stands for Language Integrated Query. is a .NET language extension that supports data retrieval from different data sources like XML document, databases and collections. It was introduced in the .NET 3.5 framework. VB and C# are the languages that have LINQ capabilities.

→ Advantages :-

+ Familiar Language - Developers don't have to learn a new query language for each type of data source or data format.

+ Less Coding - It reduces the amount of code to be written as compared with a more traditional approach.

+ Readable code - LINQ makes the code more readable. so other developers can easily understand and maintain it.

+ Standardized way of querying multiple data sources :- The same LINQ syntax can be used to query multiple data sources.

+ Compile time safety of queries - It provides type checking of objects at compile time.

PAGE No.
DATE / /

* Intellisense Support - LINQ provides Intellisense for generic collections.

AJAX :-

AJAX is about updating parts of a webpage, without reloading the whole page.

* What is AJAX?

→ AJAX = Asynchronous Javascript and XML

- AJAX is a technique for creating fast and dynamic web pages.
- AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the ~~scen~~ scenes. This means that it is possible to update parts of a web page, without reloading the whole page
- Classic web pages, (which do not use AJAX) must reload the entire page if the content should change.
- ex of applications using AJAX → Google Maps, Gmail, YouTube, & facebook tabs

Page Life Cycle :-

Page stage	Event Raised	Used For
Page Request		ASP.NET decides to send a cached page or new page (page lifecycle is triggered)
Start	PreInit	isPostBack property : to see if the page is postback or new.
		Create Dynamic Controls.
		Set masterpage (if Dynamic) and theme
Initialization	Init	Controls on the page are available. Control Init is set before page init. postback data is not yet available also control values not yet set from viewstate.
	InitComplete.	Tracking of viewstate is enabled. That means programatic viewstate changes can be made. Use this event to make changes to view state that you want to make sure are persisted after the next postback.

Load	PreLoad		Viewstate loaded. Preload is raised after viewstate is loaded and postback data is processed
	On load	PageLoad	is called.

Page Life cycle events (First time page call).

1) PreInit
2) Init
3) Init Complete
4) PreLoad
5) Load
6) Load Complete
7) PreRender
8) PreRender Complete
9) SaveState
10) Render
11) Unload

Page life cycle events sequence, after button click

1) PreInit
2) Init
3) Init Complete
4) PreLoad
5) Load
6) PostBackEventHandling (Button Click)
7) Load Complete
8) PreRender
9) PreRender Complete
10) SaveState
11) Render
12) Unload

PAGE No.
DATE / /

Web Services:-

A web service is a resource that is available over the internet. It's valuable because it provides functionality other applications can use, such as payment processing, logins and database storage. This collection of protocols and standards is typically used to exchange data between apps or systems.

What is An API?

→ An application programming interface (or API for short) is a software component that enables two otherwise unrelated applications to communicate with each other. The result of this communication is increased functionality. An API consists of standardized rules and functions that ~~dictate~~ determine what data may be taken or modified within an application and how the process occurs.

Fiddler → ① use for API testing
~~② to pass input~~ ② To pass input to API & view result in fiddler.

GET → get response from API
POST → use to pass parameter.

[{"ID":19,"Name":"Rutvi","City":"Jalna"}]

API

Choose API project

Asp.NET Web Application (.NET Framework)
↳ Next button
↳ Create project (Name, Location etc) → Create button
↳ Empty → Add folder & core refe
↳ Web API ☑
↳ Create button.

~~Dicsan~~

Deserialize

JSON format → Key : Value pair

Program:-

```
namespace apiExample.Controllers.
{
    public class EmployeeController : ApiController
    {
        SqlConnection strcon : _ _ .

        [HttpGet]
        [Route("api/Emp/get")]
```

Validation:-

1) Required Field Validator:-

Ensure that the required field is not emply.

2) Regular Expression Validator:-

Regular expression allows input text in a specific pattern.

3) Range Validator:-

Range validator used input value within given range

4) Compare Validator:-

The compare validator compares value one control with a fixed value another control.

5) Validation Summary:-

Validation summary shows all error of the page

ReadOnly keyword	Const keyword
① In C#, readonly fields can be created using readonly keyword	① In C#, constant fields are created using const keyword
② ReadOnly is a runtime constant	② Const is a compile time constant
③ The value of readonly field can be changed	③ The value of the const field can not be changed.
④ It cannot be declared inside the method	④ It can be declared inside the method
⑤ In readonly fields, we can assign values in declaration and in the the constructor part	⑤ In const fields, we can only assign values in declaration part.
⑥ It can be used with static modifiers	⑥ It cannot be used with static modifiers.

Partial Class -

Program/ code is manage in piece of code is known as partial ~~cet~~ class.

MVC :-

- ASP.NET MVC :-

- ASP.NET is a free web framework for building websites and web applications on .NET Framework.
- ASP.NET MVC 5 is a web framework based on Model - View - Controller (MVC) architecture.
- It is an open - source software from Microsoft
- It's web development framework combines the features of MVC architecture, the most up-to-date ideas and techniques
- The whole idea behind using the ~~model~~ Model View Controller design pattern is that you maintain a seperation of concerns.

MVC Architecture :-

MVC stands for Model, View and Controller. MVC seperates an application into three components - Model, View and Controller

- **Model** : Model Represents the data

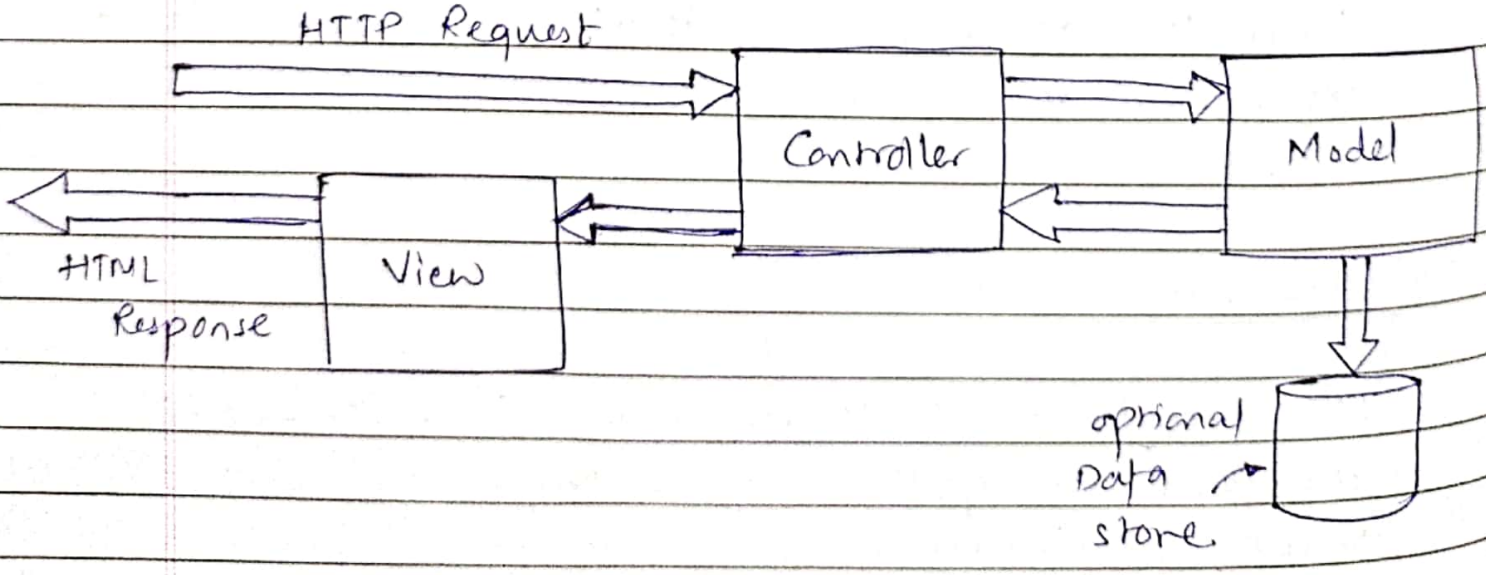
A class in C# is used to describe a model. Model objects store data retrieved from the database.

- **View** : View is the User Interface

View in MVC is a user interface. View display model data to the user and also enables them to modify them. View in ASP.NET MVC is HTML, CSS and some special syntax that makes it easy to communicate with the model and the controller.

- **Controller** : Controller is the request handler

The controller handles the user request. Typically, the user uses the view and raises an HTTP request, which will be handled by the controller. The controller processes the request and returns the appropriate view as a response.



PAGE No.
DATE / /

Controller handles the user interaction ~~requst~~ ~~requst~~ request & select appropriate action handler to execute and passes the query string values to the model, which in turn can query database based on information received from controller.