**ASP dotnet**

Asp .net is a application framework from Microsoft for developing web applications to produce dynamic web pages.

Applications are divided into :

1. Desktop application

To consume them we need to install them into system.

1. Mobile Application

Designed for mobile devices.

To consume them they have to be installed in mobile

1. Web Application

Installed in a centralised server

Then we consume these applications from the server

After installing the application in server we can access it from any device provided we have the internet and a browser

**How to develop a web application?**

Asp -> active server pages

Asp .net -> successor to asp.

Php

Jsp, servlets

Using these with the help of a language we develop applications

Asp .net uses c# or vb ,net

**What do we need to develop a web application**

HTML, CSS, Javascript/Jquery, ASP .net with c#, Web server( IIS)

We develop web applications we need to develop web pages. Web applications is a collection of web pages.

**Working with IIS server**

To host the web application we need server, and IIS server serves the purpose

IIS : Internet Information services

Generally comes with the OS

**Server Controls**

In asp .net we don’t use HTML controls, because these cant be directly accessed in the server side, so asp .net has their own server controls

Attribute to tell this runs at server : runat=”server”

<asp: TextBox id=”Textbox1” runat=”server">

Maintains the state of values, submits the value to the server.

Asp.net server controls can be directly used in our code.

To get the data of html control( attribute name should be name not id)

<input type=”text” name=”TextBox2”/>

To access it in server side code : string str2= Request.Form[“TextBox2”];

To write the data : Response.Write(str2);

Rendering : the process of converting our asp .net code into html code and send it to the browser. This is done by the web server

Controls have three properties :

1. Properties

Attribute of a control which mainly has its impact on the look of the control.

These tells the behaviour of the control, also we can style the control using these properties.

Id : is important to recognise a particular control

We can modify properties by specifying in the property window or even in the source view.

1. Methods

An action that has to be performed.

Every control has some control associated with it.

By calling a method, we tell what task a particular control has to do.

Methods can be called only in the code.

Example : TextBox1.focus()

1. Events

A time period which tells when an action has to be performed.

Example : after clicking on a button a particular task has to be performed

Onclick=”btn\_click” -> event( this is written in the server control.

Every control has many events associated with it.

Every control has default event ( button -> click event., textbox-> textchanged event)

**PostBacks**

Is a process of page submitting back to itself. Sending request back to itself.

If we write something in a textbox and click on a button, then the page submits to itself, the server receives the data that is associated with the control.

Class has a property called IsPostBack.

This.IsPostBack -> true ( if it is next or postback request) , false ( first request)

Because of this property we are able to access the data written at the client side.

In a postback all the data associated with the form and controls will be submitted back to the server.

Who is capable of submitting a page : by default button control does the submitting the data to the server.

Whereas other controls like textbox, checkbox, radiobutton, etc can also be used to submit the page to server only when the AutoPostBack property is set as true, default is false.

**Postback events and cached events**

Page has a series of events, those events will fire everytime the page is loaded either in the first request of the page as well as in the postback request page.

Control events run only in the Postback event

Postback events run only when the page is posted back to the server.

When we define any event handlers to controls which are not capable of submitting page to the server, when the events fire based on client action gets stored in the cache memory and fires after the page is submitted to the server.

First the cached events will get fired before the postback events.

**PostBack vs Cross Page Post Back**

Postback : page calling back itself. Submitting data to itself

Crosspagepostback : page calling and submitting data to another page.

We can specify PostBackUrl to send the postback to another page

Also we can use respone.redirect or serverTransfer to send the control to new page

**Server.transfer and response.redirect**

In server.transfer, the transfer is done internally in the server but the browser doesn’t know. It is faster as it happens on the server. Can transfer only to .aspx pages. Previous page object is available in the memory and can be accessed in the new page.

While using Server.transfer we can use the previous page values like:

Page pp=PreviousPage

Control ctrl= pp.FindControl(“txtName”);

TextBox tb=(textbox)ctrl;

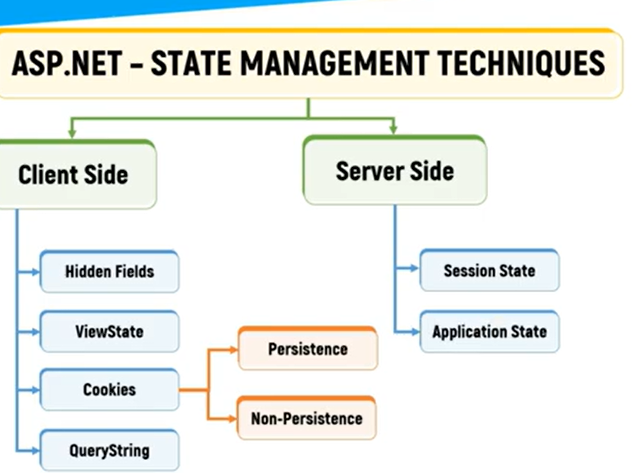
String name=tb.TemplateSourceDirectory;

While in response.redirect, a response is sent to browser informing it, a fresh request has to be sent to the server for the Failure.aspx. slower, because there is additional roundtrip. Can transfer to both aspx or html pages. Previous page object is not available because the request comes from browser ( and no state is maintained)

In response.redirect we can pass data using the query string

http:localhost//Success?id=1&name=Nikhita&age=21

**STATE MANAGEMENT**



**Hidden field:**

Control provided by asp .net, used to store small amount of data on the client

Hidden filed control is not rendered to the browser and is invisible on the browser

<asp:HiddenField Id=”HiddenField1” runat=”server”/>

Sensitive data donot use the hidden field

Stored and read from the page.

**Query String:**

Pass information between the pages by appending data to URL.

Technique used for transferring small amounts of non-sensitive data

// Redirecting to another page with query string

Response.Redirect("Page2.aspx?username=John");

// Retrieving query string on the destination page

string username = Request.QueryString["username"];

**Cookies:**

Small pieces of data stored on the clients machine.

Can be used to store user-specific info such as login credentials

// Setting a cookie

HttpCookie cookie = new HttpCookie("Username", "John");

Response.Cookies.Add(cookie);

// Retrieving a cookie

HttpCookie cookie = Request.Cookies["Username"];

if(cookie != null)

{

string username = cookie.Value;

}

**View State and control state**

Used to persist data between postbacks for a single page in ASP .net web forms.

Automatically maintained by asp .net for each control on the page.

Can be used for storing small amounts of data but can increase the size of page if large amounts of data are stored.

View state values are stored in the browser itself.

protected void Page\_Load(object sender, EventArgs e)

{

if(!IsPostBack)

{

ViewState["Name"] = "John";

}

}

protected void btnSubmit\_Click(object sender, EventArgs e)

{

string name = (string)ViewState["Name"];

// Use the name variable

}

**Session State**

Alows you to store user-specific data that needs to be persisted across multiple requests. It's stored on the server and identified by a unique session ID, which is usually stored as a cookie on the client's machine.

// Setting session variable

Session["Username"] = "John";

// Retrieving session variable

string username = (string)Session["Username"];

**Application State**

Application State allows you to store data that is shared across all users of an application. It's stored on the server and remains available as long as the application is running.

// Setting application variable

Application["TotalUsers"] = 100;

// Retrieving application variable

int totalUsers = (int)Application["TotalUsers"];

**ASP .net MVC**

Based on MVC design pattern that allows to develop software solutions

MVC design pattern:

* Allows to develop web applications with loosely coupled components
* Enables separating data access, business and presentation logic from each other

Tight Coupling

* Asp .net web forms are tightly coupled.
* Each .aspx page is dependent on its code behind page
* Automated testing is difficult

Loose coupling

* ASP .net mvc is loosely coupled
* Logic is separated into 3 parts

MVC : Model + view + controller

Model

* Classes in c#
* Database operations such as fetch, update, insert are performed
* Represents info about domain that can be the application data of web application

View

* End user GUI(graphical user interface) with which user interacts
* Represents the presentation logic to provide the data of model

Controller

* Classes in c#
* Contain logic and provide link b/w model and view
* Logic responsible for interacting the view and model
* Handles the requests to the application by the user
* Get data from database
* Specify the view that returns a response to client

Benefits of MVC :

* Separation of concerns
* Simplified testing and maintenance
* Extensibility