ASP.NET

(Active Server Pages)

Web Applications work on a HTTP protocol.Http protocol is a stateless protocol meaning it doesn’t retain state between user requests.

**ViewState** variables- are used to preserve data across page postback(ViewState of one webform is not available in other webform)

**SessionState** variables – are available across all pages, but only for a given single session.Session variables are like single user global data. Only the current session has access to it’s session state.

**ApplicationState** variables – are available across all pages and across all sessions.Application State variables are like multi-user global data. All sessions can read and write ApplicationState variables.

**IsPostBack**-determines if the page is being loaded in response to a client postback or if it is being loaded and accessed for the first time

**IIS-**Internet Information Services-delivers web pages to clients using Hypertext Transfer Protocol(HTTP)

To check if IIS is installed ->Start->run->inetmgr

DropDownList – Set the DataTextField and DataValueField before DataSource and DataBind().

**Lists** :DropDownList,CheckBoxList,RadioButtonList,ListBox and BulletedList

**AdRotator**:is used to display random ads this ads information can be stored in a xml file or in a db

**Calendar**

**Hidden Field**-is used to store a value that needs to be persisted across posts to the server, but you don’t want the control or it’s value visible to the user.

Alternatives for hidden field (ViewState,QueryStrings,SessionState and Cookies)

**MultiView** –is made up of multiple view controls, and each view control in turn can have controls inside it(ActiveViewIndex property of the MultiView control is used to determine the view that is visible or active)

**Wizard**- enables creation of multi-step user interface .Wizard control provides with build-in previous /next functionality

Panel Control – is used as a container for other controls, and it can be used to show/hide all the controls in the group or for adding controls dynamically to the webform.

**Validators**

**RequiredFieldValidators-** is used to ensure that the data entered by the user is valid .

**RangeValidator-** is used to check if the value is within a specific range of values

**CompareValidator-**is used to compare the value of one control with the value of another contrl or a constant value.(Comparision operators:Equals,GreaterThan,GreaterThanEqual,LessThan,LessThanEqual,NotEqual,DataTypeCheck)

**RegularExpressionValidator-** is used to check if the value of an associated input control matches the pattern specified by a regular expression.(ValidationExpression property of this validator is used)

**CustomValidator**- allows us to write a method with a custom logic to handle the validation of the value entered .

**ValidationSummary-**is used to display a summary of all validation errors occurred in a web page at one place.(In a real time app it is common do display a red star symbol next to the input control where the error occurred)

**ValidationGroups**-we group the validators in orther to validate the group and not only one validator

**Different Page navigation Tehniques**

**HyperLink –**is used to navigate to another page , the page you want to navigate to is specified by the navigate URL

**Reponse.Redirect**-user Button,LinkButton or ImageButton to intercept a click event and call Response.Redirect() method .When the method is called automatically issues a new Get request to the web server. The web server will then server the new page.(so in short Response.Redirect causes 2 requests/response cycles.)

**Server.Transfer**-is used to navigate to other pages/sites running on the same web server.it can’t be used to navigate to pages/sites on other servers.Is faster than Response.Redirect() because there is only 1 Request/Response.Server.Transfer preserve the variabiles from the original form.

**Serve.Execute-**the url in the browser doesn’t changes.The difference between Server.Transfer and Server.Execute is that Server.Transfer terminates the execution of the current page and starts the execution of the new page , where as Server.Execute process the second web form without leaving the first webform .After completing the execution of the first webform the control returns to the second form

**CrossPagePosting-**allows to post one page to another page (if u want to PostBack on a button click set the PostBackURL of the button to the page that you want to post to(Page.IsCrossPagePostBack property is used to indicate wheter the page is involved in a cross-page postback .(to strongly type reference in CrossPagePostBack by creating propertys and referring to those propertys (ex: public string Name{get{return txtName.Text;}})

**Opening a new window using JavaScript** –Window.Open(URL,name,features,replace)

**Tehniques to send data from one web form to another**

1.CrossPagePostBack ; 2.Context.Handler object; 3.Query Strings; 4.Cookies; 5.Session State;6.Application State

**Context.Handler**- return webForm1 as the previous page only the first time when you land on webform2 from webform1 .If there is a button on webform2 and if you click the button the page will postback and Context.Handler will return webForm2 instead of webForm1.

**QueryString**-are name/value collection pairs.QueryStrings are appended to the URL page(? Indicates the beginning of a query string and it’s value.Subsquent query strings can pe appended to the URL using the & symbol. There is a limit on the Query string length. Hence query strings can’t be used to send very long data. Query Strings are available to the user hence should not be sent sensitive information, unless encrypted. To read the query string use the Request onbject’s Query String property)We can use Server.UrlEncode(txtName.Text) or txtName.Text.Replace(“&”,%26”) to encode the query string

**Cookie**-store small amounts of information on the client’s machine in general web sites , use cookies to store user preferences or other information that is client-specific. Cookies are classified into 2 types:**1.Persitant Cookies**(Remain on the client computer , even after the browser is closed.To configure how lomg the cookies remain using the expires property of the HttpCookie object.**2.Non-Persistant Cookies**(if you don’t se the Expire’s property then the cookie is called as a Non-Persistant cookie and remain in memory until the browser is closed.)

**SessionState variables-**can also be used to send data from one webform to another(Are stored on the web server by default, and are kept for the life time of a session. The default mode is InProc. The timeout value determines the lifetime of a session state variable. Session state variables are available across all pages but only for a single given session(are like single user global data). Is always good to check if session variable IsNull before calling any of it’s methods. Session state can be turned off at page or application level, set SessionState mode=false is web.config file.<%Page EnableSessionState=”false”>

**SessionState Mode Management-**

**WebFarm=web application deployed on multiple server**

**WebGarden=web application deployed on a server with multiple processors**

1.**Off**-disables session state for the entire application

2.**InProc** –session state variables are stored on the web server memory inside the asp.net worker process.(Will perform best because the session state memory is kept on the webserver, within the ASP.NET worker process(w3wp.exe). Suitable for webapplications hosted on a single server. Objects can be added without serialization.DISADVANTAGES –session state data is lost when the worker process or application pool is recycled.Not suitable for web farms or web gardens.Scalability could be an issue)

3.**StateServer session state mode-**the session state variables are stored in a process called as asp.net state service.The asp.net state service can be present on a web server or a dedicated machine.(1.start->run->services.msc->ASP.NET StateService->Start. 2.In web.config set session state mode=”StateServer”.3.Set stateConnectionString=”tcpip=StateServer:42424)ADVANTAGES(Survives worker process restart. Can be used with web farms and web gardens. State server offers more scalability than InProc)DISADVANTAGES(State Server is slower than InProc. Complex objects need to be serialized and deserialized . If SessionServer is on a dedicated machine and if the server goes down all session variables are lost)

4.**SQLServer session state mode –**the session state variables are stored on a SQLServer database. To configure 1.Create the ASPState database using aspnet\_regsql.exe tool from the cmd(cmd-commands-msdn.microsoft.com/en-us/library/ms229862(v=vs.100).aspx).2.Set the session state mdoe =”SQLServer” and sqlConnectionstring=”data source=.;integrated security=SSPI”. ADVANTAGES(SQLServer is the most reliable option. Survives worker process recycling and SQLServer restarts. Can be used with web farms and web gardens. More scalable than State server and InProc session state modes.) DISADVANTAGES(Slower than StateServer and InProc session state modes. Complex objects need to be serialized and deserialized) -> start->run->inetmgr->Application pools->right click ASP.NET 4 ->Advance settings->Identity->Local System->if we are running on local system)

**ApplicationState**-are available across all pages and across all sessions.(are like multi user global data). Are stored on the web server . Application state variables are cleared, only when the process hosting the 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 against race conditions, deadlocks and access violations. User application state variables when u need to have global access and when you need them for entire time, during the life time of an application.Cache object can be used as an alternative

**Exception Handling-** try{}catch(){} finally{release resources} throw – uset to raise exception

**Error Events –** 1.Page\_Error – this event is raised at page level, when there is an unhandled exception on the page .The event handler resides on the page

2.Applicaiont\_Error – this event is raised at the application level, when there is an unhandled exception at an application level. The event handler resides in Global.asax file.

**Custom Errors -**  for a list of all available HTTP status : <http://en.wikipedia.org/wiki/List_Of_HTTP_status_codes> .To specify the custom error pages at an application level, use customErrors element in web.config.

<customErrors mode=”on” defaultRedirect=”DefaultErrorPage.aspx”>

<error statusCode=”401” redirect=”UnauthorizedErrorPage.aspx”/>

<error statusCode=”404” redirect=”PageNotFoundErrorPage.aspx”/>

<error statusCode=”500” redirect=”InternalServerErroPage.aspx”/>

</customErrors>

The mode attribute determines when a custom error page is displayed over the yellow screen, exception page Mode attribute can have on, off or RemoteOnly. Default is RemoteOnly.

**Windows event viewer –** RightClick ->My Computer->Manage->EventViewer or Start->Run->Eventvwr

**Logging exceptions to database**

**Sending Emails using asp.net-**Configurating the capability of sending emails to web config

<appSettings>

<add key=”SendEmail” value=”true”/>

</appSettings>

**Tracing –** enables us to view diagnostic information about a request and is very usefull when debugging application problems. Tracing can be turned on or off: 1.At the application level 2.At the page level

In the url after we launch the webform ->localhost:56414/Traces.axd – to see the trace file

To append the trace messages at the end of the page set pageOutput=”true”

**Application Pools –** An application pool can contain one or more applications. In IIS it is possible to create one or more application pools. Deploying applications to different application pools enables us to achieve the degree of application isolation that we need , in term of availability and security.

Start->run->inetmgr . Asp.Net applications execute inside asp.net worked process called w3wp.exe. The applicaions are executed by the worker process, using a window identity. The windows identit is that is used independent on the application pool identity. The application pool identity can by of the following built in accounts. **1.LocalService 2.Local System 3.Network System 4.ApplicationPoolIdentity**

In addition tot thes built-in accounts we can also use a custom account by specifying the username and password.

**Local System**-Completely trusted account and has very high privileges and can also access network resources

**Network Services-**Restricted or limited service account that is generally uset to run, standard least-priveleged Services. This account has less privileges than Local System account. This account can access local resources

**Local Service –** Restricted or limited service account that is very similar to Network Service and meant to run standard least-priviledged services. This account can’t acces network resources.

**ApplicationPoolIdentity**- When a new application pool is created, IIS creates a virtual account with the name of the new Application Pool and run the application pool’s worker processes under this account. This is also least privileged account.

Running an application using a low-priveleged account is a good security practice.

**Anonymous Authentication –** Authentication-Who is the user? Authorization-What rights the user has?

Anonymous authentication allows users to access the public ares of the web site, without prompting the users ofr a user name or password.

By default, the application pool identity is used to execute application code. To disable anonymous authentication: click”Details” link unde actions in the right hand side panel in IIS or add the following cfg

<authorization>

<deny users=”?”/>

<allow users=”\*”/>

</authorization>

**Anonymous Authentication and Impersonation –** to enable impersonation set impersonate=”true” for the identity element in web.config.

1.If IMPERSONATION is disabled, then , the application pool identity is used to execute the application code.

2.If IMPERSONATION is enabled, then , ‘NT AUTHORITY\IUSR’ account is used to execute the application code

If there are 2 or more websites hosted on a machine, with IUSR as the anonymous account, then they can access each other’s content. If we want to isolate , each applications content, the applications can be deployed to different application pools, and the NTFS file permissions can be set for the respective application pool identity.

<configuration>

<system.web>

<compilation debug=”true” targetFramework=”4.0”/>

<identity impersonate=”true”/>

</system.web>

</configuration>

**Forms authentication using user names list in web.config**

Anonymous authentication – is fine for websites that contain public information that every one can use

Windows authentication – is used for intranet web applications where the users are part of a windows domain-based network

Forms authentication is used for internet web applications. The advantage of Forms authentication is that users don’t have to be member of a domain – based network to have access to your applications.(forms authentication classes are present in System.Web.Security)

**Secure Socket Layer** –

HTTP stands for Hyper Text Transfer Protocol and HTTPS stands for Hyper Text Transfer Protocol Secure.When the server and the client communicate using HTTP the messages are not encrypted. With HTTPS the messages are encrypted and very secure.

An SSL certificate contains a public key and certificate issuer.

Extra processing time is required for HTTPS, for key negotiation is also termed as SSL handshake. The handshake allows the server to authenticate itself to the client.

**Implementing SSL**

Creating a self signed certificate. 2 of the easier options available are : 1.Using IIS ; 2.Using MakeCert.exe <http://msdn.microsoft.com/en-us/library/bfsktky3.aspx>

Ex 1 .Using IIS ->Server cetificates->Create Self –>Signed Certificates->Pc Name->ok

Ex. 2.Start->All Prog->Visual Studio->Visual Studio Tools->VisualStudio cmd-> makecert –r –pe –n “CN=YourComputerName” –b 01/01/2000 -e 01/01/2100 –ss my –sr localMachine –sky exchange –sp “Microsoft RSA SChannel Cryptographic Provider” –sy 12

The certificates that we create with IIS are automatically installed in the trusted root authority store

If you want to dis-allow access over HTTP protocol there are 2 ways :

1.Remove HTTP binding at the IIS Server level. This Option will prevent all the web applications running on that server to use only HTTPS binding

2.Let both the bindings be available at the server level and configure SSL settings at an application or web site level(chose require SSL)

Note: Use import and export features of IIS to import and export certificates

**Redirect HTTP to HTTPS**

Step1. Please download and install “URLRedirect” module from the following link <http://www.iis.net/downloads/microsoft/url-write>

Step2 . Uncheck “Require SSL” option from “SSL Settings” for the web application in IIS

Step3.Add ReWrite rules in web.config file These rules can also be created in IIS directly using the “URLRewrite” module.

After installing the “URLRewrite” paste the rules in the root web.config

<system.webserver>

<httpRedirect enabled=”false” destination=”” httpResponseStatus=”Found”/>

<rewrite>

<rules>

<rule name=”HTTP to HTTPS Redirection” stopProcessing=”true”/>

<match url=”(.\*)”/>

<conditions>

<add input=”{HTTPS}” pattern =”off”/>

</conditions>

<action type=”Redirect” url=https://{HTTP\_HOST}{REQUEST\_URI} redirectType=”Found”/>

</rule>

</rewrite>

</system.webserver>

**User Controls** –Web server controls combine one or more server or HTML control on a web user control page, which can in turn be used on web form as a single control

**Custom Server Controls –**Unlike UserControls , composite controls are created completely in code.