**ASP.NET2019**

**Q-1[A]ONE MARK:**

1]asp stand for-Active Server Pages

2]clr stand for-common language runtime

3]extension of user control is- **.** ascx

4]which validation control is used to check whether the user entered some value or not.

a. RegularExpressionValidator

**Q-1[B]TWO MARK:**

1]file upload control

a. The FileUpload control allows the user to browse for and select the file to be uploaded, providing a browse button and a text box for entering the filename.

Once, the user has entered the filename in the text box by typing the name or browsing, the SaveAs method of the FileUpload control can be called to save the file to the disk.

The basic syntax of FileUpload is:

<asp:FileUpload ID= "Uploader" runat = "server" />

The FileUpload class is derived from the WebControl class, and inherits all its members. Apart from those, the FileUpload class has the following read-only properties:

|  |  |
| --- | --- |
| **Properties** | **Description** |
| FileBytes | Returns an array of the bytes in a file to be uploaded. |
| FileContent | Returns the stream object pointing to the file to be uploaded. |
| FileName | Returns the name of the file to be uploaded. |
| HasFile | Specifies whether the control has a file to upload. |
| PostedFile | Returns a reference to the uploaded file. |

2]textbox control with properties

a. This is an input control which is used to take user input. To create TextBox either we can write code or use the drag and drop facility of visual studio IDE.

This is server side control, asp provides own tag to create it. The example is given below.

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

Server renders it as the HTML control and produces the following code to the browser.

<input name="TextBox1" id="TextBox1" type="text">

This control has its own properties that are tabled below.

|  |  |
| --- | --- |
| **Property** | **Description** |
| AccessKey | It is used to set keyboard shortcut for the control. |
| TabIndex | The tab order of the control. |
| BackColor | It is used to set background color of the control. |
| BorderColor | It is used to set border color of the control. |
| BorderWidth | It is used to set width of border of the control. |
| Font | It is used to set font for the control text. |
| ForeColor | It is used to set color of the control text. |
| Text | It is used to set text to be shown for the control. |
| ToolTip | It displays the text when mouse is over the control. |
| Visible | To set visibility of control on the form. |
| Height | It is used to set height of the control. |
| Width | It is used to set width of the control. |
| MaxLength | It is used to set maximum number of characters that can be entered. |
| Readonly | It is used to make control readonly. |

**Q-1[C]THREE MARK:**

1]types of files in asp.net

a. 1.) .asax --> It refers to the Global.asax file containing code that drives from the HttpApplication class.It resides Application root directory.

2.) .ascx --> It refers to a web user control file. It resides Application root directory or a subdirectory.

3.) .aspx --> It refers to a ASP.NET Web forms.It resides Application root directory or a subdirectory.

4.) .asmx --> It refers to an xml web services file that contains classes and methods. It resides Application root directory or a subdirectory.

.5.) .axd --> It refers to a handler file that is used to website administration requests.It resides Application root directory.

6.) .cd --> It refers to a class diagram file. It resides Application root directory or a subdirectory.

7.) .compile --> It refers to a precompiled stub file that point to an assembly representing a compiled website file.Fir example .aspx,.ascx ,.master file are precompiled. It resides Bin subdirectory.

8.) .browser --> It refers to a browser definition file used to identify the features of client browsers. It redies App\_Browsers Subdirectory.

9.) .dll --> It refers to a compiled class library files (assembly file).It resides Bin subdirectory.

10.) .cs,.vb,.jsl --> It refers to a Source -code file that contains application logic.It resides App\_Code Subdirectory or same directory as web page.

11.) .csproj,.vbproj,.vjsproj --> It refers to a project file for a visual studio client application project.It resides Project Directory of visual studio.

12.) .master --> It refers to a master page that defines the layout of web page in a web application.It resides Application root or subdirectory.

13.) .mdf,.sdf --> It refers to a SQL database file. It resides App\_Data subdirectory.

14.) .mdb,ldb --> It refers to a Access database file. It resides App\_Data subdirectory.

15.) .msgx,svc --> It refers to an indigo messaging framework (MFx) service file.It resides Application root or a subdirectory.

16.) .soap --> It refers to a SOAP extension file .It resides Application root or a subdirectory.

17.) .sin --> It refers to a Solution file for visual web developer project. it resides Visual web Developer project directory.

18.) .skin --> It refers to a skin file.It is used for consistent formatting in web controls. It resides App\_Themes subdirectory.

19.) .sitemap --> It refers to a site-map file that containing the structure of the website. It resides Application root directory.

20) .rem --> It refers to a handler file which implements remoting concepts in web application.It resides Application root or a subdirectory.

2]installation steps of IIS server

## a. Step 1: Start Server Manager

As with all Windows Server roles, we have to go to the Server Manager to begin the installation. Hit your “**Windows**” key and search for **Server Manager** if it is not already opened. Once open, click on “**Add Roles and Features**“

Step 2: Click Next on Wizard

On the first page of the “**Add Roles and Features Wizard**“, click “**Next**“

Step 3: Select Installation Type

In the “**Select Installation type page**“, select “**Role-based or feature-based-installation**” and click “**Next**“

Step 4: Choose Destination Server

Select the server you will install NFS on and click “**Next**“

Step 5: Select Roles to install

In this “**Select server roles**” part check the “**WebServer (IIS)**” box then a pop-up window will come up.

Step 6: Add IIS Features

In the pop-up window, just click on “**Add Features**” then hit “**Next**“. After that click “**Next**” on the next three consecutive windows as illustrated below.

Step 7: Confirm Selections

On the “**Confirm installation selections**” page simply click on “**Install**” and afford it some time to finish after which you just click “**Close**“.

Step 8: Prove the Web Server is running

Open your browser either within the server or on a computer that can access your IIS Server network and input its IP Address on the browser’s search as shown below. If it loads, then we are good to go.

Step 9: Configure Default Site

**Q-1[D]FIVE MARK:**

1]explain two validation control with example

a. ASP.NET provides controls that automatically check user input and require no code. We can also create custom validation for our application.

**Validator Description**

*CompareValidator*-It is used to compare the value of an input control against a value of another input control.

*RangeValidator*-It evaluates the value of an input control to check the specified range.

*RegularExpressionValidato*r-It evaluates the value of an input control to determine whether it matches a pattern defined by a regular expression.

*RequiredFieldValidator*-It is used to make a control required.

*ValidationSummary*-It displays a list of all validation errors on the Web page.

* **ASP.NET CompareValidator Control**

This validator evaluates the value of an input control against another input control on the basis of specified operator.

We can use comparison operators like: less than, equal to, greater than etc.

## CompareValidator Properties

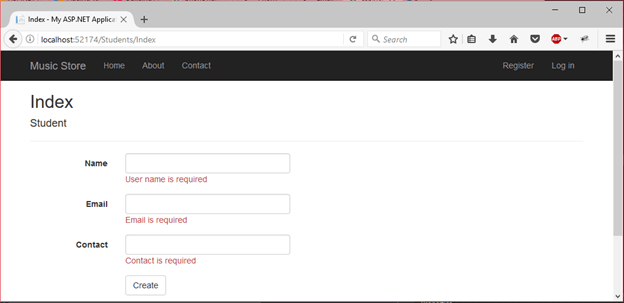
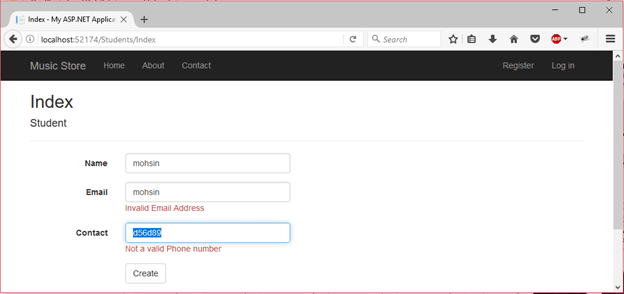
|  |  |
| --- | --- |
| **Property** | **Description** |
| AccessKey | It is used to set keyboard shortcut for the control. |
| TabIndex | The tab order of the control. |
| BackColor | It is used to set background color of the control. |
| BorderColor | It is used to set border color of the control. |
| BorderWidth | It is used to set width of border of the control. |
| Font | It is used to set font for the control text. |
| ForeColor | It is used to set color of the control text. |
| Text | It is used to set text to be shown for the control. |

Here, in the following example, we are validating user input by using CompareValidator controller. Source code of the example is given below.

**// compare\_validator\_demo.aspx**

1. **<**%@ Page Language="C#" AutoEventWireup="true" CodeBehind="compare\_validator\_demo.aspx.cs"
2. Inherits="asp.netexample.compare\_validator\_demo" %**>**
3. <!DOCTYPE html**>**
4. **<html** xmlns="http://www.w3.org/1999/xhtml"**>**
5. **<head** runat="server"**>**
6. **<title></title>**
7. **<style** type="text/css"**>**
8. .auto-style1 {
9. width: 100%;
10. }
11. .auto-style2 {
12. height: 26px;
13. }
14. .auto-style3 {
15. height: 26px;
16. width: 93px;
17. }
18. .auto-style4 {
19. width: 93px;
20. }
21. **</style>**
22. **</head>**
23. **<body>**
24. **<form** id="form1" runat="server"**>**
25. **<table** class="auto-style1"**>**
26. **<tr>**
27. **<td** class="auto-style3"**>**
28. First value**</td>**
29. **<td** class="auto-style2"**>**
30. **<asp:TextBox** ID="firstval" runat="server" required="true"**></asp:TextBox>**
31. **</td>**
32. **</tr>**
33. **<tr>**
34. **<td** class="auto-style4"**>**
35. Second value**</td>**
36. **<td>**
37. **<asp:TextBox** ID="secondval" runat="server"**></asp:TextBox>**
38. It should be greater than first value**</td>**
39. **</tr>**
40. **<tr>**
41. **<td** class="auto-style4"**></td>**
42. **<td>**
43. **<asp:Button** ID="Button1" runat="server" Text="save"**/>**
44. **</td>**
45. **</tr>**
46. **</table>**
47. **<** **asp:CompareValidator** ID="CompareValidator1" runat="server" ControlToCompare="secondval"
48. ControlToValidate="firstval" Display="Dynamic" ErrorMessage="Enter valid value" ForeColor="Red"
49. Operator="LessThan" Type="Integer"**></asp:CompareValidator>**
50. **</form>**
51. **</body>**
52. **</html>**

Output:

RangeValidator Control

The RangeValidator control verifies that the input value falls within a predetermined range.

It has three specific properties:

|  |  |
| --- | --- |
| **Properties** | **Description** |
| Type | It defines the type of the data. The available values are: Currency, Date, Double, Integer, and String. |
| MinimumValue | It specifies the minimum value of the range. |
| MaximumValue | It specifies the maximum value of the range. |

The syntax of the control is as given:

<asp:RangeValidator ID="rvclass" runat="server" ControlToValidate="txtclass"

ErrorMessage="Enter your class (6 - 12)" MaximumValue="12"

MinimumValue="6" Type="Integer">

</asp:RangeValidator>

2]features of asp.net framework

a. ASP.NET is full of features and provides an awesome platform to create and develop web application. Here, we are discussing these features of Web Forms.

* Server Controls
* Master Pages
* Working with data
* Membership
* Client Script and Client Frameworks
* Routing
* State Management
* Security
* Performance
* Error Handling

Server Controls

Web Forms provides rich set of server controls. These controls are objects that run when the page is requested and render markup to the browser. Some Web server controls are similar to familiar HTML elements, such as buttons and text boxes. It also provides controls that we can use to connect to data sources and display data.

Master Pages

It allowsus to create a consistent layout for the pages in our application. This page defines the look and feel and standard behavior that we want for all of the pages in our application. When users request the content pages, they merge with the master page to produce output that combines the layout of the master page with the content from the content page.

Working with Data

In an ASP.NET Web Forms application, we use data-bound controls to automate the presentation or input of data in web page UI elements such as tables and text boxes and drop-down lists.

Membership

Project's Account folder contains the files that implement the various parts of membership: registering, logging in, changing a password, and authorizing access. Additionally, ASP.NET Web Forms supports OAuth and OpenID. These authentication enhancements allow users to log into your site using existing credentials, from such accounts as Facebook, Twitter and Google.

Client Script and Client Frameworks

We can enhance the server-based features of ASP.NET by including client-script functionality in ASP.NET Web Form pages. We can use client script to provide a richer, more responsive user interface to the users. We can also use client script to make asynchronous calls to the Web server while a page is running in the browser.

Routing

We can configure URL routing of our application. A request URL is simply the URL a user enters into their browser to find a page on our web site. We use routing to define URLs that are semantically meaningful to users and that can help with search-engine optimization (SEO).

State Management

ASP.NET Web Forms includes several options that help you preserve data on both a per-page basis and an application-wide basis.

Security

Developing a secure application is most important aspect of software development process. ASP.NET Web Forms allow us to add extensibility points and configuration options that enable us to customize various security behaviors in the application.

Performance

Web Forms provides good performance and allows us to modify performance related to page and server control processing, state management, data access, application configuration and loading, and efficient coding practices.

Debugging and Error Handling

We can diagnose problems that occur in our Web Forms application. Debugging and error handling are well supported within ASP.NET Web Forms so that our applications compile and run effectively.

**Q-2[A]ONE MARK:**

1]you can have multiple global.asax file in your web application.true/false

a. You cannot have multiple Global. asax, unless you convert folders to Areas which is probebly overkill ( false )

2]which is small text data stored in client machine?

a. Cookie

3]HTTP is stateless protocol?true/false

a.true

4]abandon method used with cookies?true/false

a.

**Q-2[B]TWO MARK:**

1]query string

a. A is a collection of characters input to a computer or web browser. A Query String is helpful when we want to transfer a value from one page to another. ... A Query String Collection is used to retrieve the variable values in the HTTP query string query string.

If we want to transfer a large amount of data then we can't use the Request.QueryString. Query Strings are also generated by form submission or can be used by a user typing a query into the address bar of the browsers. Query Strings are contained in request headers.

Syntax of Query String

Request.QueryString(variable)[(index).count].

2]explain cookies

a. ASP.NET Cookie is a small bit of text that is used to store user-specific information. This information can be read by the web application whenever user visits the site.

When a user requests for a web page, web server sends not just a page, but also a cookie containing the date and time. This cookie stores in a folder on the user's hard disk.

When the user requests for the web page again, browser looks on the hard drive for the cookie associated with the web page. Browser stores separate cookie for each different sites user visited.

here are two ways to store cookies in ASP.NET application.

* Cookies collection
* HttpCookie

**Q-2[C]THREE MARK:**

1]global.asax file

* a. The Global.asax is also known as the ASP.NET application file and is used to serve application-level and session-level events.
* It allows us to write code that response to global application events raised by ASP.NET or by HttpModules.
* These events fire at various points during the lifetime of a web application, including when the application domain is first created.
* The Global.asax file resides in the root directory of an ASP.NET-based application
* At run time, global.asax is parsed and compiled into a dynamically generated .NET Framework class derived from the HttpApplication base classThe Global.asax file is optional. If you do not define the file, the ASP.NET page framework assumes that you have not defined any application or session event handlers

**the Global.asax file contains the following events:**

**Application\_Init:** Fired when an application initializes or is first called. It is invoked for all HttpApplication object instances.  
  
**Application\_Disposed:** Fired just before an application is destroyed. This is the ideal location for cleaning up previously used resources.  
  
**Application\_Error:** Fired when an unhandled exception is encountered within the application.  
  
**Application\_Start:** Fired when the first instance of the HttpApplication class is created. It allows you to create objects that are accessible by all HttpApplication instances.  
  
**Application\_End:** Fired when the last instance of an HttpApplication class is destroyed. It is fired only once during an application's lifetime.  
  
**Application\_BeginRequest:** Fired when an application request is received. It is the first event fired for a request, which is often a page request (URL) that a user enters.  
  
**Application\_EndRequest:** The last event fired for an application request.  
  
**Application\_PreRequestHandlerExecute:** Fired before the ASP.NET page framework begins executing an event handler like a page or Web service.  
  
**Application\_PostRequestHandlerExecute:** Fired when the ASP.NET page framework has finished executing an event handler.  
  
**Applcation\_PreSendRequestHeaders:** Fired before the ASP.NET page framework sends HTTP headers to a requesting client (browser).  
  
**Application\_PreSendContent:** Fired before the ASP.NET page framework send content to a requesting client (browser).  
  
**Application\_AcquireRequestState:** Fired when the ASP.NET page framework gets the current state (Session state) related to the current request.  
  
**Application\_ReleaseRequestState:** Fired when the ASP.NET page framework completes execution of all event handlers. This results in all state modules to save their current state data.  
  
**Application\_ResolveRequestCache:** Fired when the ASP.NET page framework completes an authorization request. It allows caching modules to serve the request from the cache, thus bypassing handler execution.  
  
**Application\_UpdateRequestCache:** Fired when the ASP.NET page framework completes handler execution to allow caching modules to store responses to be used to handle subsequent requests.  
  
**Application\_AuthenticateRequest:** Fired when the security module has established the current user's identity as valid. At this point, the user's credentials have been validated.  
  
**Application\_AuthorizeRequest:** Fired when the security module has verified that a user can access resources.  
  
**Session\_Start:** Fired when a new user visits the application Web site.  
**Session\_End:** Fired when a user's session times out, ends, or they leave the application Web site.

2]explain view state

a. Viewstate is a very useful client side property. It is used for page level state management. Viewstate stores any type of data and used for sending and receiving information,

When We Should Use View State

-When the data to be stored is small.

-Try to avoid secure data.

It is implemented using a view state object defined by the StateBag class which defines a collection of view state items. The state bag is a data structure containing attribute value pairs, stored as strings associated with objects.

The StateBag class has the following properties:

Properties Description

Item(name) The value of the view state item with the specified name. This is the default property of the StateBag class.

Count The number of items in the view state collection.

Keys Collection of keys for all the items in the collection.

Values Collection of values for all the items in the collection.

The StateBag class has the following methods:

Methods Description

Add(name, value) Adds an item to the view state collection and existing item is updated.

Clear Removes all the items from the collection.

Equals(Object) Determines whether the specified object is equal to the current object.

Finalize Allows it to free resources and perform other cleanup operations.

GetEnumerator Returns an enumerator that iterates over all the key/value pairs of the StateItem objects stored in the StateBag object.

GetType Gets the type of the current instance.

IsItemDirty Checks a StateItem object stored in the StateBag object to evaluate whether it has been modified.

Remove(name) Removes the specified item.

SetDirty Sets the state of the StateBag object as well as the Dirty property of each of the StateItem objects contained by it.

SetItemDirty Sets the Dirty property for the specified StateItem object in the StateBag object.

ToString Returns a string representing the state bag object.

**Q-2[D]FIVE MARK:**

1]session state with example

a. In ASP.NET session is a state that is used to store and retrieve values of a user.

It helps to identify requests from the same browser during a time period (session). It is used to store value for the particular time session. By default, ASP.NET session state is enabled for all ASP.NET applications.

Each created session is stored in SessionStateItemCollection object. We can get current session value by using Session property of Page object.

In the following example, we are creating a session and storing user email. This example contains the following files.

### // Default.aspx

1. **<**%@ Page Title="Home Page" Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
2. Inherits="SessionExample.\_Default" %**>**
3. **<head>**
4. **<style** type="text/css"**>**
5. .auto-style1 {
6. width: 100%;
7. }
8. .auto-style2 {
9. width: 105px;
10. }
11. **</style>**
12. **</head>**
13. **<form** id="form1" runat="server"**>**
14. **<p>**Provide Following Details**</p>**
15. **<table** class="auto-style1"**>**
16. **<tr>**
17. **<td** class="auto-style2"**>**Email**</td>**
18. **<td>**
19. **<asp:TextBox** ID="email" runat="server" TextMode="Email"**></asp:TextBox>**
20. **</td>**
21. **</tr>**
22. **<tr>**
23. **<td** class="auto-style2"**>**Password**</td>**
24. **<td>**
25. **<asp:TextBox** ID="password" runat="server" TextMode="Password"**></asp:TextBox>**
26. **</td>**
27. **</tr>**
28. **<tr>**
29. **<td** class="auto-style2"**>** **</td>**
30. **<td>**
31. **<asp:Button** ID="login" runat="server" Text="Login" OnClick="login\_Click" **/>**
32. **</td>**
33. **</tr>**
34. **</table>**
35. **<br** **/>**
36. **<asp:Label** ID="Label3" runat="server"**></asp:Label>**
37. **<br** **/>**
38. **<asp:Label** ID="Label4" runat="server"**></asp:Label>**
39. **</form>**

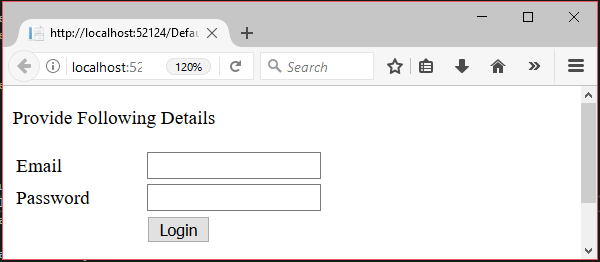
## Code

### // Default.aspx.cs

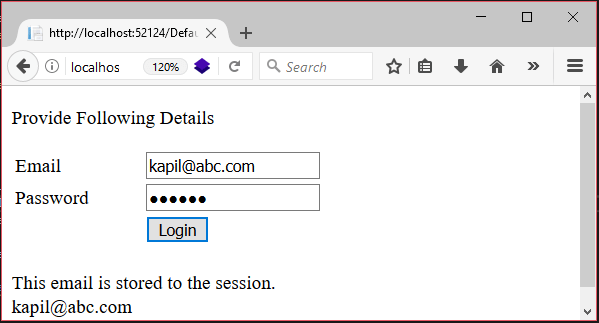
1. **using** System;
2. **using** System.Web.UI;
3. **namespace** SessionExample
4. {
5. **public** partial **class** \_Default : Page
6. {
7. **protected** **void** login\_Click(**object** sender, EventArgs e)
8. {
9. **if** (password.Text=="qwe123")
10. {
11. // Storing email to Session variable
12. Session["email"] = email.Text;
13. }
14. // Checking Session variable is not empty
15. **if** (Session["email"] != **null**)
16. {
17. // Displaying stored email
18. Label3.Text = "This email is stored to the session.";
19. Label4.Text = Session["email"].ToString();
20. }
21. }
22. }
23. }

Output:

This application will store user email to the session when user login.



It will show stored session value, user email.



2]what is state?explain client side and server side state management.

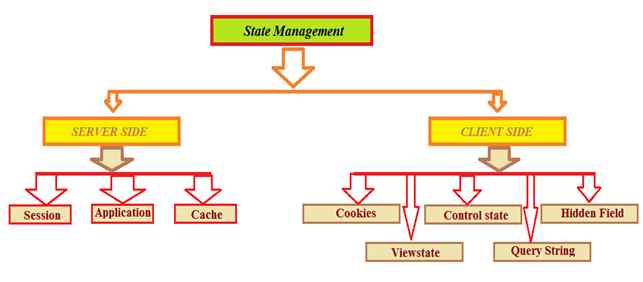
a. State management is implemented in order to retain information about the user requests. Web pages are stateless. Each request creates new page without retaining any previous information about the user requests. ASP.NET supports several State management techniques to maintain state information.

State management in ASP.NET can be classified into

1. Client-side state management

2. Server-side state management

Two types of State Management techniques are available in ASP.NET as in the following figure,

State Management Part

* Server side

Session

Session is a very important technique to maintain state. Normally session is used to store information and identity. The server stores information using Sessionid.

## Application

Application State is a server side management state. It is also called application level state management. In this mainly store user activity in server memory and application event shown in Global.asax file.

## Cache

Cache is stored on server side. It implements Page Caching and data caching. Cache is use to set expiration polices  
  
Response.Cache.SetExpiresTime(DateTime.Now.AddDays(1));

# Client Side

Now here I am explaining client side state management one by one:

Also state management has the following four important parts available on the client side,

## Cookie

Cookie is a small and an important part of ASP.NET. In this store user information, session and application. It can be created constant and temporary and they work with browser request. Cookies are store on client side. The server can read cookies and abstract data.  
  
Two types of cookies are available,  
  
**Persistence**  
  
This type of cookie works with Date and time.

1. Response.Cookies["CookieName"].Value = "Test Cookies";
2. //set expire time
3. Response.Cookies["CookieName"].Expires = DateTime.Today.AddHours(1);

**Non-Persistence**  
  
This is a temporary cookie. It is created with access application and discards the close application.

1. Response.Cookies["CookieName"].Value = "Test Cookies";

## Control state

Control state technique is developed to maintain data work properly in order. We can use view state but suppose view state is disabled by the user, the control will not work as expected. For expected results of the control we have to use Control State. In application, the Viewstate is by default true. Sometimes we need to use custom control to manage application properly.

## Hidden Field

Hidden fields are used to store value to client side. Hidden field is not displayed on the browser, but it works on a request.

1. **if** (HiddenField1.Value != **null**)
2. {
3. **int** val = Convert.ToInt32(HiddenField1.Value) + 1;
4. HiddenField1.Value = val.ToString();
5. Label1.Text = val.ToString();
6. }

## View state

Viewstate is a very useful client side property. It is used for page level state management. Viewstate stores any type of data and used for sending and receiving information,

## Query String

Query string stores the value in URL.  
  
Response.Redirect("ShowStringValue.aspx?Username=" + txtUsername.Text);  
  
It is visible to all the users in url as in the following link,

**Q-3[A]ONE MARK:**

1]which namespace use when we connect our asp.net page to sql server?

a. SqlClient namespace

2]which is the execute method which is used to read data from the table?

a. executeReader

3]full form ADO-ActiveX Data Object

4]in which architecture we are connected with the database throughout our opration?

a.

**Q-3[B]TWO MARK:**

1]datareader class

a. A data reader provides an easy way for the programmer to read data from a database as if it were coming from a stream.

This class is used to read data from SQL Server database. It reads data in forward-only stream of rows from a SQL Server database. it is sealed class so that cannot be inherited. It inherits DbDataReader class and implements IDisposable interface.

The DataReader properties

|  |  |
| --- | --- |
| PROPERTY | DESCRIPTION |
| Depth | Indicates the depth of nesting for row |
| FieldCount | Returns number of columns in a row |
| IsClosed | Indicates whether a data reader is closed |
| Item | Gets the value of a column in native format |
| RecordsAffected | Number of row affected after a transaction |

**Table 5-27.**The DataReader methods

|  |  |
| --- | --- |
| METHOD | DESCRIPTION |
| Close | Closes a DataRaeder object. |
| Read | Reads next record in the data reader. |
| NextResult | Advances the data reader to the next result during batch transactions. |
| Getxxx | There are dozens of Getxxx methods. These methods read a specific data type value from a column. For example. GetChar will return a column value as a character and GetString as a string. |

2]difference:executescaler() v/s execute no query()

a. **ExecuteScalar():**

1. will work with Non-Action Queries that contain aggregate functions.
2. Return the first row and first column value of the query result.
3. Return type is object.
4. Return value is compulsory and should be assigned to a variable of required type.

**ExecuteNonQuery():**

1. will work with Action Queries only (Create,Alter,Drop,Insert,Update,Delete).
2. Returns the count of rows effected by the Query.
3. Return type is int
4. Return value is optional and can be assigned to an integer variable.

**Q-3[C]THREE MARK:**

1]difference:connected architecture v/s disconnected architecture

| **Connected** | **Disconnected** |
| --- | --- |
| It is connection oriented. | It is dis\_connection oriented. |
| Connected methods gives faster performance | Disconnected get low in speed and performance. |
| Datareader | DataSet |
| Connected can hold the data of single table | Disconnected can hold data of multiple tables |
| Connected you need to use a read only forward only data reader | In Disconnected you cannot use. |
| Data Reader can’t persist the data | Data Set can persist the data |
| It is Read only, we can’t update the data. | We can update the data. |

^aa side coonected and >disconnected

2]difference:data table v/s data set

a. **DataTable:**  
1.*Meaning:*A DataTable is an in-memory representation of a single database table which has collection of rows and columns  
  
2.*Number of rows retrieved at a time:*DataTable fetches only one TableRow at a time  
  
3.*Provision of DataRelation Objects:*As DataTable is a single database table, so there is no DataRelation object in it.  
  
4.*Enforcing Data Integrity*:In DataTable, there is no UniqueConstraint and ForeignKeyConstraint objects available.  
  
5.*DataSource can be Serialized or Not:*In DataTable, DataSource cannot be serialized.  
  
6.T*o know the example for DataTable*, please try the URL: [http://www.dotnetperls.com/datatable](http://www.dotnetobject.com/expage.php?exurl=http://www.dotnetperls.com/datatable)  
  
**DataSet:**  
  
1.*Meaning:*A DataSet is an in-memory representation of a database-like structure which has collection of DataTables.  
  
2.*Number of rows retrieved at a time:*DataSet can fetch multiple TableRows at a time  
  
3.*Provision of DataRelation Objects:* In DataSet, DataTable objects can be related to each other with DataRelation objects.  
  
4.*Enforcing Data Integrity:*In DataSet, data integrity is enforced by using the UniqueConstraint and ForeignKeyConstraint objects.  
  
5.*DataSource can be Serialized or Not:*DataSet is serialized DataSource .That is why web services can always returns DataSet as the result but not the DataTables.  
  
6.*To know the example for DataSet*, please try the URL: [http://www.dotnetperls.com/dataset](http://www.dotnetobject.com/expage.php?exurl=http://www.dotnetperls.com/dataset)

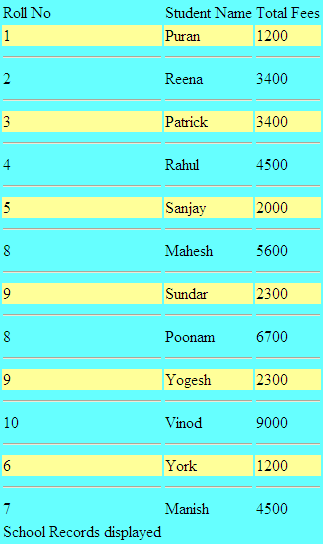
**Q-3[D]FIVE MARK:**

1]repeater control

a. . The Repeater control is used to display a repeated list of items that are bound to the control. The Repeater control may be bound to a database table, an XML file, or another list of items.  
  
Repeater is a Data Bind Control. Data Bind Controls are container controls. Data Binding is the process of creating a link between the data source and the presentation UI to display the data. ASP .Net provides rich and wide variety of controls, which can be bound to the data.  
  
**Repeater has 5 inline template to format it:**  
1. <HeaderTemplate>  
2. <FooterTemplate>  
3. <ItemTemplate>  
4. <AlternatingItemTemplate>  
5. <SeperatorTemplate>  
6. <AlternatingItemTemplate>  
  
**HeaderTemplate:** This template is used for elements that you want to render once before your ItemTemplate section.  
  
**FooterTemplate:** - This template is used for elements that you want to render once after your ItemTemplate section.  
  
**ItemTemplate:**This template is used for elements that are rendered once per row of data. It is used to display records  
  
**AlternatingItemTemplate:** This template is used for elements that are rendered every second row of data. This allows you to alternate background colors. It works on even number of records only.  
  
**SeperatorTemplate:** It is used for elements to render between each row, such as line breaks.  
  
**Some point about Repeater Control**

* It is used to display backend result set. It is used to display multiple tuple.
* It is an unformatted control. The Repeater control is a basic templated data-bound list. It has no built-in layout or styles, so you must explicitly declare all layout, formatting, and style tags within the control's templates.
* The Repeater control is the only Web control that allows you to split markup tags across the templates. To create a table using templates, include the begin table tag (<table>) in the HeaderTemplate, a single table row tag (<tr>) in the ItemTemplate, and the end table tag (</table>) in the FooterTemplate.

The Repeater control has no built-in selection capabilities or editing support. You can use the ItemCommand event to process control events that are raised from the templates to the control

The ASP.NET Repeater Control will not render the result unless you bound it to a data source through its DataSource property.  
 ****

2]gridview control

a. . The GridView control displays the values of a data source in a table. Each column represents a field, while each row represents a record. The GridView control supports the following features:

* Binding to data source controls, such as SqlDataSource.
* Built-in sort capabilities.
* Built-in update and delete capabilities.
* Built-in paging capabilities.
* Built-in row selection capabilities.
* Programmatic access to the GridView object model to dynamically set properties, handle events, and so on.
* Multiple key fields.
* Multiple data fields for the hyperlink columns.
* Customizable appearance through themes and styles.

Creating a GridView

1. <asp:GridView ID="gridService" runat="server">
2. </asp:GridView>

GridView Control Example in ASP.Net C#

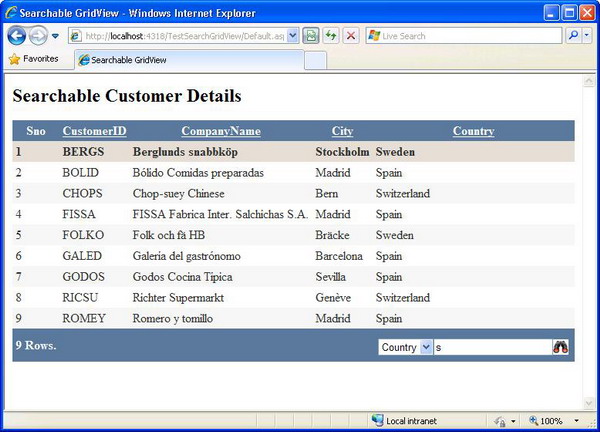
**Step 1 –** Open the Visual Studio –> Create a new empty Web application.

**Step 2 –** Create a New web page.

**Step 3 –** Drag and drop GridView Control on web page from toolbox.

**Step 4 –** Create New Database in SQL Server

**Step 5 –** Make connection between Database and web application.



**Q-4[A]ONE MARK:**

1]skin file extension

a. .skin file

2]master page can run individually.true/false

a.

3]which type of caching is very useful for static page?

a. Page output caching.

4]extension of master page is…

a. .MASTER

**Q-4[B]TWO MARK:**

1]partial page caching

a. Partial page caching enables you to cache portion of a page. It is also called as control caching or fragment caching. Normally partial caching is performed using the "User Control". You can do caching with the help of user control as same as page output caching but you have to do caching in user control.

 Similar to output caching, partial page caching allows you to cache certain blocks of your website. You can, for example, only cache the center of the page. Partial page caching is achieved with the caching of user controls. You can build your ASP.NET pages consisting of numerous user controls and then apply output caching on the user controls you select. This will cache only parts of the page that you want, leaving other parts of the page outside the reach of caching. This is a very nice feature, and if done correctly, it can lead to pages that perform better.

2]difference:theme v/s css

a. **CSS:**  
1.Applies to all HTML Controls   
2.Is applied on the Client Side in the Browser   
3.We can apply multiple style sheets to a single page  
4.The CSS supports cascading  
5.The CSS cannot override the property values defined for a control.  
6.Cannot be applied through the configuration files  
7.Can be used directly via a reference to the css file location  
8.Do not require any other resource like Skin files  
9.In case of CSS you can define only style properties   
  
**Themes:**  
1.Applies to all the server controls   
2.Is applied on the server rather than in the browser   
3.But we cannot apply multiple themes to a single page. Only one theme we can apply for a single page.  
4.But themes does not support cascading  
5.But any property values defined in a theme, the theme property overrides the property values declaratively set on a control, unless we explicitly apply by using the StyleSheetTheme property.  
6.Can be applied through Configuration Files.  
7.All theme and Skin files should be placed in a special Asp.net folder called the “App\_Themes” in order for the themes to work and behave normally.  
8.Each theme should be associated with at least one Skin file.  
9.But a theme can define multiple properties of a control not just style properties such as we can specify the graphics property for a control, template layout of a GridView control etc.

**Q-4[C]THREE MARK:**

1]data caching

a. Data caching means caching data from a data source. As long as the cache is not expired, a request for the data will be fulfilled from the cache. When the cache is expired, fresh data is obtained by the data source and the cache is refilled.

These controls derive from the abstract class DataSourceControl and have the following inherited properties for implementing caching:

CacheDuration - It sets the number of seconds for which the data source will cache data.

CacheExpirationPolicy - It defines the cache behavior when the data in cache has expired.

CacheKeyDependency - It identifies a key for the controls that auto-expires the content of its cache when removed.

EnableCaching - It specifies whether or not to cache the data.

### Example

To demonstrate data caching, create a new website and add a new web form on it. Add a SqlDataSource control with the database connection already used in the data access tutorials.

For this example, add a label to the page, which would show the response time for the page.

<asp:Label ID="lbltime" runat="server"></asp:Label>

Apart from the label, the content page is same as in the data access tutorial. Add an event handler for the page load event:

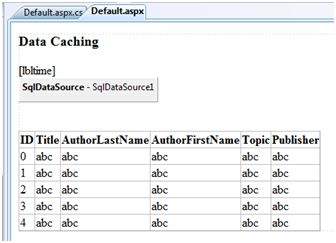
protected void Page\_Load(object sender, EventArgs e)

{

lbltime.Text = String.Format("Page posted at: {0}", DateTime.Now.ToLongTimeString());

}

The designed page should look as shown:



2]master page in detail.

a. . Master pages provide templates for other pages on your web site.

Requirements;

Master pages allow you to create a consistent look and behavior 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.

**Q-4[D]FIVE MARK:**

1]what is caching?explain page output caching.

a. Caching is a technique of storing frequently used data/information in memory, so that, when the same data/information is needed next time, it could be directly retrieved from the memory instead of being generated by the application.

Caching is extremely important for performance boosting in ASP.NET, as the pages and controls are dynamically generated here. It is especially important for data related transactions, as these are expensive in terms of response time.

Page output caching

This is a form of caching in ASP.Net that stores a copy of your Web page in the memory cache so that subsequent requests for the same Web page can be fetched directly from the cache -- the cached output is sent to the application. This improves the application's performance considerably. The following code snippet shows how you can implement page output caching.

i.e. the HTML and client script that the server sends to browsers for rendering. When a visitor views a page, the server caches the output code in the application's memory. Until the cache expires, the system displays the page using the cached output. As a result, the application can quickly serve requests without running the page code or communicating with SQL servers.

**Q-5[A]ONE MARK:**

1]XML stand for- eXtensible Markup Language

2]wsdl stand for- Web Services Description Language (WSDL)

3]which is process to check for any type of bugs that our application may have?

a.

4]SOAP stand for- Simple Object Access Protocol

**Q-5[B]TWO MARK:**

1]web.config file

a. ASP.NET *Web.config* allows you to define or revise the configuration settings at the time of developing the application or at the time of deployment or even after deployment. The following are brief points that can be understood about the *Web.config* file:

* *Web.config* files are stored in XML format which makes us easier to work with.
* You can have any number of *Web.config* files for an application. Each *Web.config* applies settings to its own directory and all the child directories below it.
* All the *Web.config* files inherit the root *Web.config* file available at the following location *systemroot\Microsoft.NET\Framework\versionNumber\CONFIG\Web.config* location
* IIS is configured in such a way that it prevents the *Web.config* file access from the browser.

The changes in *Web.config* don’t require the reboot of the web server

2]what is XML?

A. XML is a software- and hardware-independent tool for storing and transporting data.

* XML stands for eXtensible Markup Language
* XML is a markup language much like HTML
* XML was designed to store and transport data
* XML was designed to be self-descriptive
* XML is a W3C Recommendation

**Q-5[C]THREE MARK:**

1]explain page level tracing and application level tracing.

a. ASP.NET tracing enables you to view diagnostic information about a single request for an ASP.NET page

**(i) Page Level Tracing**

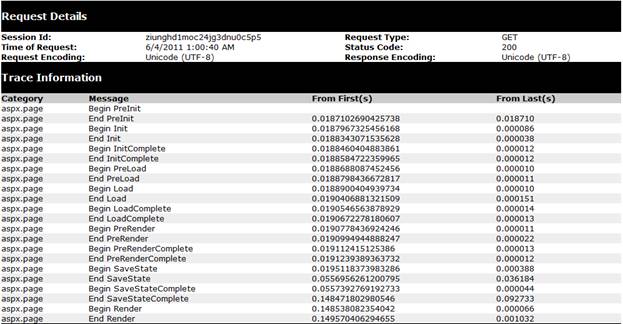
We can control whether tracing is enabled or disabled for individual pages. If tracing is enabled, when the page is requested, ASP.NET appends to the page a series of tables containing execution details about the page request. Tracing is disabled by default in an ASP.NET application.

**Application Level Tracing**

 Instead of enabling tracing for individual pages, you can enable it for your entire application. In that case, every page in your application displays trace information. Application tracing is useful when you are developing an application because you can easily enable it and disable it without editing individual pages. When your application is complete, you can turn off tracing for all pages at once.

 When you enable tracing for an application, ASP.NET collects trace information for each request to the application, up to the maximum number of requests you specify. The default number of requests is 10. You can view trace information with the trace viewer.

 By default, when the trace viewer reaches its request limit, the application stops storing trace requests. However, you can configure application-level tracing to always store the most recent tracing data, discarding the oldest data when the maximum number of requests is reached.



2]authorization in setail

a. **Authorization is** the process of allowing an authenticated user access to resources.

Authorization refers to the process that determines what a user is able to do. For example, an administrative user is allowed to create a document library, add documents, edit documents, and delete them. A non-administrative user working with the library is only authorized to read the documents.

Authorization is orthogonal and independent from authentication. However, authorization requires an authentication mechanism. Authentication is the process of ascertaining who a user is. Authentication may create one or more identities for the current user.

Types:7.45

**Q-5[D]FIVE MARK:**

1]custom error handling in asp.net

a.

2]explain web services in detail

a.