**ASP.NET2018**

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

1]CLR stand for -Common Language Runtime (CLR)

2]code behind files which extension?

a. **.** aspx

3]which extension used to match white space character?

a. \s

4]using which property we can set multiple lines in textbox?

a. TextBox.Rows Property

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

1]explain listbox control

a. ListBox control is an asp.net web server control. ListBox control used to store the multiple items and allow user to select multiple item from listbox control. ListBox control is same as dropdownlist control.

In a ListBox control there is a **SelectionMode property** to change the mode of section from single to multiple. By default listbox control selection mode is single if you want to select multiple items from listbox, then just change the SelectionMode property to multiple.

**Some important Properties of ListBox Control.**

1. **Items.Count:** Returns the total number of items in the list box.
2. **Items.Clear:** Clears all the items from the list box.
3. **SelectedItem.Text:** Returns the text of the selected item.
4. **Items.Remove(“name”):** Removes the item with text “name”.
5. **Items.RemoveAt(int index):** Removes the item present at the given index.

2]different types of files?

a. . There are some file types managed by ASP.NET as given below:-

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.

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

1]file upload control

A. The posted file is encapsulated in an object of type HttpPostedFile, which could be accessed through the PostedFile property of the FileUpload class.

The HttpPostedFile class has the following frequently used properties:

|  |  |
| --- | --- |
| **Properties** | **Description** |
| ContentLength | Returns the size of the uploaded file in bytes. |
| ContentType | Returns the MIME type of the uploaded file. |
| FileName | Returns the full filename. |
| InputStream | Returns a stream object pointing to the uploaded file. |

## Example

The following example demonstrates the FileUpload control and its properties. The form has a FileUpload control along with a save button and a label control for displaying the file name, file type, and file length.

In the design view, the form looks as follows:



The content file code is as given:

<body>

<form id="form1" runat="server">

<div>

<h3> File Upload:</h3>

<br />

<asp:FileUpload ID="FileUpload1" runat="server" />

<br /><br />

<asp:Button ID="btnsave" runat="server" onclick="btnsave\_Click" Text="Save" style="width:85px" />

<br /><br />

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

</div>

</form>

</body>

2]compare validator with example

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

It specifies the value of the input control to compare with. It specifies the constant value to compare with.

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

|  |  |
| --- | --- |
| **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. |
| ControlToCompare | It takes ID of control to compare with. |
| ControlToValidate | It takes ID of control to validate. |
| ErrorMessage | It is used to display error message when validation failed. |
| Operator | It is used set comparison operator. |

The basic syntax of the control is as follows:

<asp:CompareValidator ID="CompareValidator1" runat="server"

ErrorMessage="CompareValidator">

</asp:CompareValidator>

<table align="center" class="style1" style="border: thin solid #008080">

<tr>

<td colspan="2"

style="border-bottom: thin solid #008080; font-weight: 700; text-align: center; color: #990033;"

class="style2">

CompareValidator Control in ASP.Net</td>

</tr>

<tr>

<td>

&nbsp;</td>

<td>

&nbsp;</td>

</tr>

<tr>

<td style="font-weight: 700; text-align: right">

Name :</td>

<td>

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

</td>

</tr>

<tr>

<td style="font-weight: 700; text-align: right">

Password :</td>

<td>

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

</td>

</tr>

<tr>

<td style="font-weight: 700; text-align: right">

Confirm - Pass :</td>

<td>

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

<asp:CompareValidator ID="CompareValidator1" runat="server"

ControlToCompare="txtpassword" ControlToValidate="txtconfipass"

ErrorMessage="Password not same !!"></asp:CompareValidator>

</td>

</tr>

<tr>

<td>

&nbsp;</td>

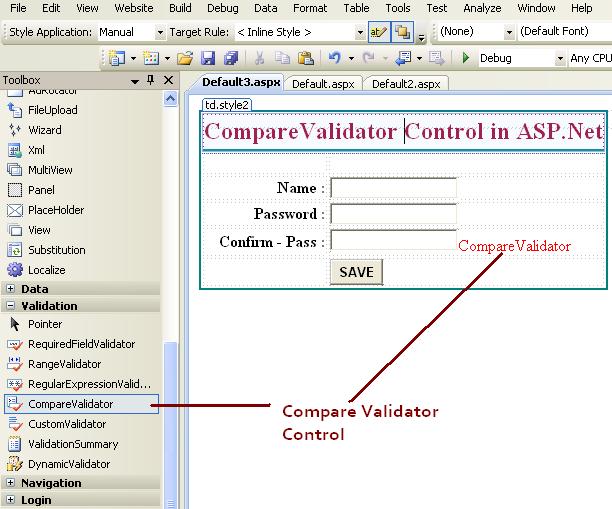
<td>

<asp:Button ID="Button1" runat="server" Font-Bold="True" Text="SAVE" />

</td>

</tr>

</table>

[](https://meeraacademy.com/wp-content/uploads/2012/06/compare.jpg)

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

1]what is asp.net?explain advantages

a. ASP.NET stands for Active Server Pages .NET

ASP.NET is a web development platform, which provides a programming model, a comprehensive software infrastructure and various services required to build up robust web applications for PC, as well as mobile devices.

ASP.NET works on top of the HTTP protocol, and uses the HTTP commands and policies to set a browser-to-server bilateral communication and cooperation.

ASP.NET is a part of Microsoft .Net platform. ASP.NET applications are compiled codes, written using the extensible and reusable components or objects present in .Net framework. These codes can use the entire hierarchy of classes in .Net framework.

The ASP.NET application codes can be written in any of the following languages:

* C#
* Visual Basic.Net
* Jscript
* J#

ASP.NET is used to produce interactive, data-driven web applications over the internet. It consists of a large number of controls such as text boxes, buttons, and labels for assembling, configuring, and manipulating code to create HTML pages.

Here are some of the advantages of using ASP.NET:

* Asp.Net is purely server-side technology, so the code is processed on the windows server before it is displayed in the web browser. Therefore, ASP.net applications execute more quickly than interpreted scripts.
* ASP.Net features such as early binding, JIT compilation, caching services and native optimization supports to get high level of performance. With .NET you are not limited to JIT but have the option AOT if you want to eliminate startup delays.
* Asp.Net framework is language independent, means you can choose any programming language (C#, J#, VB, etc) which best suited to your application.
* The Common Language Specification data types in all .Net applications are similar, so no Type conversion is necessary when calling.Net methods, C++, C# from Visual Basic, or Vice Versa.
* ASP.NET provides full support for XML, CSS and other new as well as established web standards.
* Introduction of view state helps in maintaining state of the controls automatically between the postbacks events.
* With the built-in configuration information, Asp.Net is easy to deploy. There is no need to register components because the configuration information is built-in.
* Finally, Asp.Net reduces the line of code needed to develop large applications.

2]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

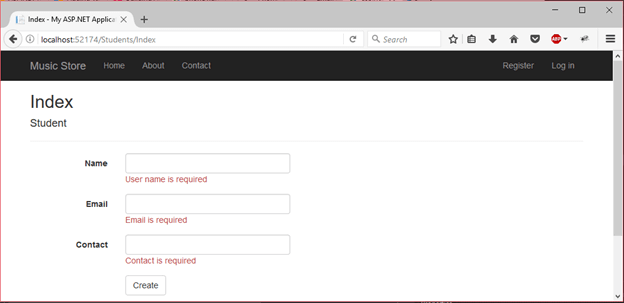
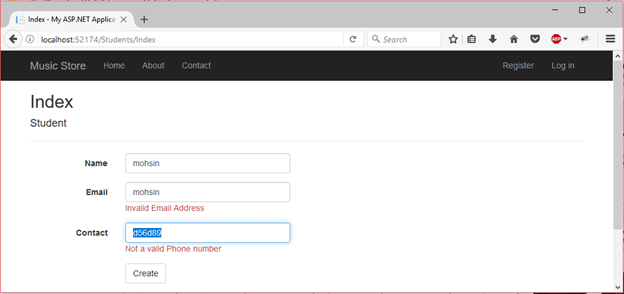
|  |  |
| --- | --- |
| **Property** | **Description** |
| AccessKey | It is used to set keyboard shortcut for the control. |
| TabIndex | The tab order of the control. |
| BackColor | It is used to set background color of the control. |
| BorderColor | It is used to set border color of the control. |
| 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:

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

1]user define cookies is fetch using which object?

a. Response.Cookies object for create Cookie.

2]which is default state of web page control?

a. View State

3]which used to send information from one page to another via url

a. Session State.

4]we can cancel the current session via which method?

a. Clear() and Abadon()

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

1]explain querystring

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 hiddenfield

a. HiddenField, as name implies, is hidden. This is non visual control in ASP.NET where you can save the value. This is one of the types of client-side state management tools. It stores the value between the roundtrip. Anyone can see HiddenField details by simply viewing the source of document.

HiddenFields are not encrypted or protected and can be changed by anyone. However, from a security point of view, this is not suggested. ASP.NET uses HiddenField control for managing the ViewState. So, don’t store any important or confidential data like password and credit card details with this control.

<asp:HiddenFieldID="HiddenField1" runat="server" />

* Use of HiddenField

We developers mostly do not show an ID value of table like ProductID, MemberID because users are not concerned with this kind of data. We store that information in HiddenFields and complete our process very easily.

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

1]explain application state with “global .asax” file.

a. Application state is a global storage mechanism that used to stored data on the server .

Application state is used to maintain the state on the server side. Its state is available through out all the users of the application. It means these state values are maintained on the application level and not on the user level and user session level.

Application state does not have default expiry or you can not set an expiry of the application state.

The Global. asax file, also known as the ASP.NET application file, is an optional file that contains code for responding to application-level and session-level events raised by ASP.NET or by HTTP modules. Global. asax is the asp.net application file.

The Global.asax file resides in the root directory of an asp.net web application.Events and states such as session state and Application state,are specified in the Global.asax file.

Global.asax file is used to maintain session and application specific event within an application.We can have only one Global.asax file in a asp.net application.

2]difference cookies v/s session

* a. Cookies are client-side files that contain user information, whereas Sessions are server-side files that contain user information.
* Cookie is not dependent on session, but Session is dependent on Cookie.
* Cookie expires depending on the lifetime you set for it, while a Session ends when a user closes his/her browser.
* The maximum cookie size is 4KB whereas in session, you can store as much data as you like.
* Cookie does not have a function named unsetcookie() while in Session you can use Session\_destroy(); which is used to destroy all registered data or to unset some.

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

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

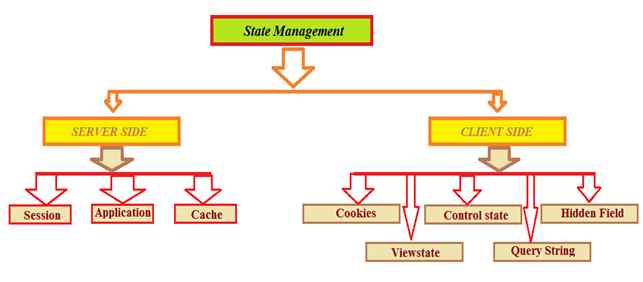
a. State management is implemented in order to retain information about the user requests. Web pages are stateless. Each request creates new page without retaining any previous information about the user requests. ASP.NET supports several State management techniques to maintain state information.

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,

2]explain session in detail

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>**

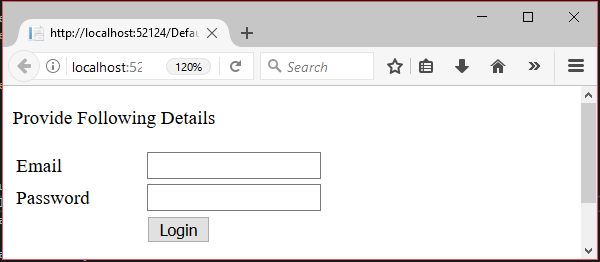
## Code

### // Default.aspx.cs

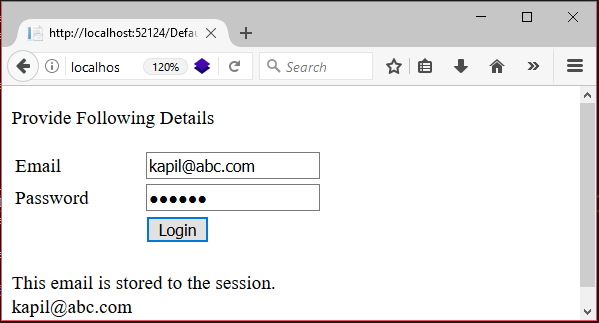
1. **using** System;
2. **using** System.Web.UI;
3. **namespace** SessionExample
4. {
5. **public** partial **class** \_Default : Page
6. {
7. **protected** **void** login\_Click(**object** sender, EventArgs e)
8. {
9. **if** (password.Text=="qwe123")
10. {
11. // Storing email to Session variable
12. 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.



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

1]which property gets primary key field value in gridview control

a. DataKeyNames property

2]full form ADO- ActiveX Data Objects

3]executereader() used to return rows for reading.true/false

a.

4]which method return the number of row affected by the column?

a. ExecuteNonQuery

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

1]give meaning:1]executenonquery() 2]executescaler()

* returns the number of rows affected. If you run the SQL from your question in a SqlCommand and check the return value of ExecuteNonQuery it should tell you how many records were affected.
* The ExecuteScalar() in C# SqlCommand Object is using for retrieve a single value from Database after the execution of the SQL Statement. The ExecuteScalar() executes SQL statements as well as Stored Procedure and returned a scalar value on first column of first row in the returned Result Set.

2]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. |

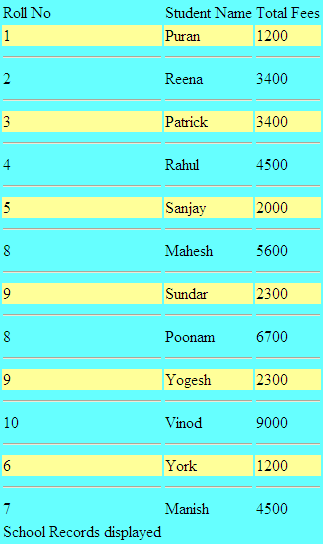
**Q-3[C]THREE MARK:**

1]explain repeater control with example

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]dataadapter class with example

a. The DataAdapter works as a bridge between a DataSet and a data source to retrieve data. DataAdapter is a class that represents a set of SQL commands and a database connection. It can be used to fill the DataSet and update the data source.

DataAdapterExample.aspx

<%@ Page Language="C#" %>

<%@ Import Namespace="System.Data" %>

<%@ Import Namespace="System.Data.SqlClient" %>

<%@ Import Namespace="System.Configuration" %>

**<!DOCTYPE html>**

<script runat="server">

**protected** **void** **Page\_Load**(**object** sender, System.EventArgs e) {

**if** (!Page.IsPostBack) {

SqlConnection MyConnection;

SqlCommand MyCommand;

SqlDataAdapter MyAdapter;

DataTable MyTable;

MyConnection = **new** SqlConnection();

MyConnection.ConnectionString = ConfigurationManager.ConnectionStrings["AppConnectionString1"].ConnectionString;

MyCommand = **new** SqlCommand();

MyCommand.CommandText = "SELECT TOP 8 \* FROM PRODUCTS";

MyCommand.CommandType = CommandType.Text;

MyCommand.Connection = MyConnection;

MyTable = **new** DataTable();

MyAdapter = **new** SqlDataAdapter();

MyAdapter.SelectCommand = MyCommand;

MyAdapter.Fill(MyTable);

GridView1.DataSource = MyTable.DefaultView;

GridView1.DataBind();

MyAdapter.Dispose();

MyCommand.Dispose();

MyConnection.Dispose();

}

}

</script>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title>DataAdapter example: how to use DataAdapter in asp.net</title>

</head>

<body>

<form id="form1" runat="server">

<div>

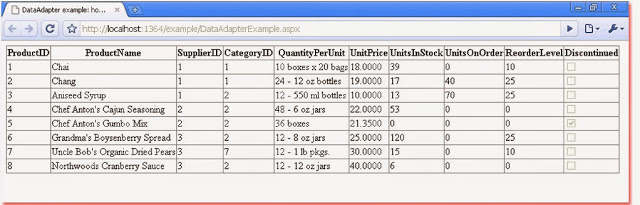
<asp:GridView ID="GridView1" runat="server"></asp:GridView>

</div>

</form>

</body>

</html>



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

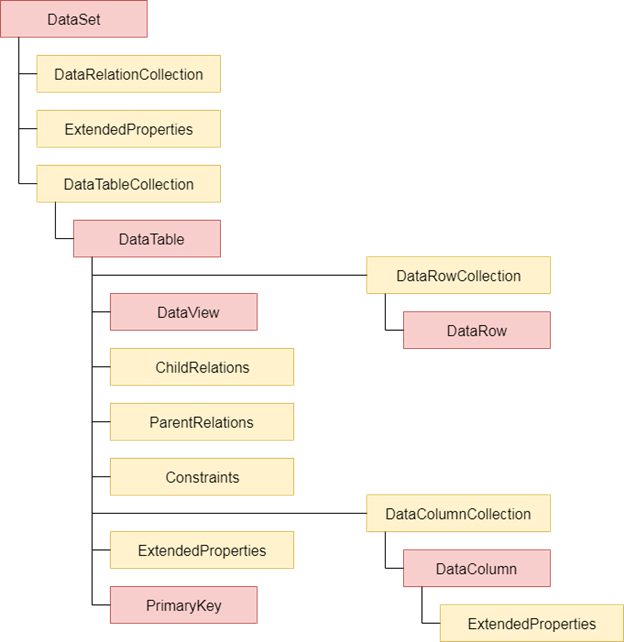
1]architecture of ado.net 5.4 book ma proper che

A. It is a module of .Net Framework which is used to establish connection between application and data sources. Data sources can be such as SQL Server and XML. ADO.NET consists of classes that can be used to connect, retrieve, insert and delete data.

All the ADO.NET classes are located into System.Data.dll and integrated with XML classes located into System.Xml.dll.

ADO.NET has two main components that are used for accessing and manipulating data are the .NET Framework data provider and the DataSet.

## .



**Fig:** ADO.NET Architecture

* The DataSet Class

The dataset represents a subset of the database. It does not have a continuous connection to the database. To update the database a reconnection is required. The DataSet contains DataTable objects and DataRelation objects. The DataRelation objects represent the relationship between two tables.

Following table shows some important properties of the DataSet class:

Properties Description

CaseSensitive Indicates whether string comparisons within the data tables are case-sensitive.

Container Gets the container for the component.

DataSetName Gets or sets the name of the current data set.

DefaultViewManager Returns a view of data in the data set.

DesignMode Indicates whether the component is currently in design mode.

EnforceConstraints Indicates whether constraint rules are followed when attempting any update operation.

Events Gets the list of event handlers that are attached to this component.

ExtendedProperties Gets the collection of customized user information associated with the DataSet.

HasErrors Indicates if there are any errors.

* The DataTable Class

The DataTable class represents the tables in the database. It has the following important properties; most of these properties are read only properties except the PrimaryKey property:

Properties Description

ChildRelations Returns the collection of child relationship.

Columns Returns the Columns collection.

Constraints Returns the Constraints collection.

DataSet Returns the parent DataSet.

DefaultView Returns a view of the table.

ParentRelations Returns the ParentRelations collection.

PrimaryKey Gets or sets an array of columns as the primary key for the table.

Rows Returns the Rows collection

* The DataRow Class

The DataRow object represents a row in a table. It has the following important properties:

Properties Description

HasErrors Indicates if there are any errors.

Items Gets or sets the data stored in a specific column.

ItemArrays Gets or sets all the values for the row.

Table Returns the parent table.

2]explain 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#

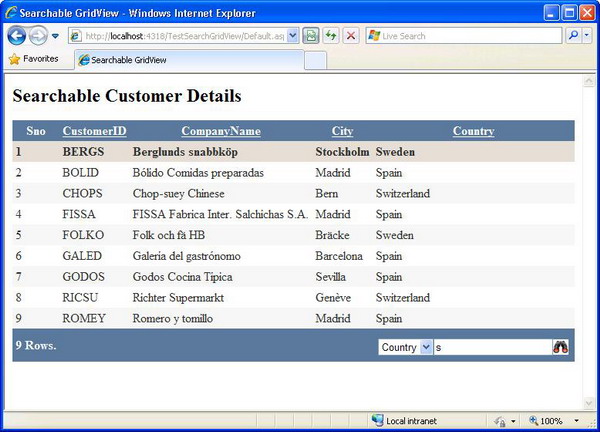
**Step 1 –** Open the Visual Studio –> Create a new empty Web application.

**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]one page contais multiple master page :true/false

a.

2]skin file extension…

a. .skin file

3]used to define the time duration of web page to be cached

a. Data Caching&cache profile

4]master page can run directly on browser.true/false

a.false

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

1]explain absolute expiration

a. Absolute Expiration allows us to specify the duration of the cache, starting from the time the cache is activated.

The cache is set to expire exactly two minutes after the user has retrieved the data.

2]page output caching

a. Output cache stores a copy of the finally rendered HTML pages or part of pages sent to the client.

When the next client requests for this page, instead of regenerating the page, a cached copy of the page is sent, thus saving time.

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

1]difference themes 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.

2]explain master page?and its requirements

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 theme?explain elements

a. ASP.NET themes are a collection of properties that define the appearance of pages and controls in your Web site. A theme can include skin files, which define property settings for ASP.NET Web server controls, and can also include cascading style sheet files

2]what is caching?explain any two types

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.

ASP.NET provides the following different types of caching:

* **Output Caching** : Output cache stores a copy of the finally rendered HTML pages or part of pages sent to the client. When the next client requests for this page, instead of regenerating the page, a cached copy of the page is sent, thus saving time.
* **Data Caching** : 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.
* **Object Caching** : Object caching is caching the objects on a page, such as data-bound controls. The cached data is stored in server memory.
* **Class Caching** : Web pages or web services are compiled into a page class in the assembly, when run for the first time. Then the assembly is cached in the server. Next time when a request is made for the page or service, the cached assembly is referred to. When the source code is changed, the CLR recompiles the assembly.
* **Configuration Caching** : Application wide configuration information is stored in a configuration file. Configuration caching stores the configuration information in the server memory

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

1]full form WSDL- Web Services Description Language

2]each application have only and must one “web.config”file.true/false

a.

3]you can write the trace information using ……..or ………..metod

a.tracing and debugging.  
  
4]which section in “wweb.config” used to display our own customized error?

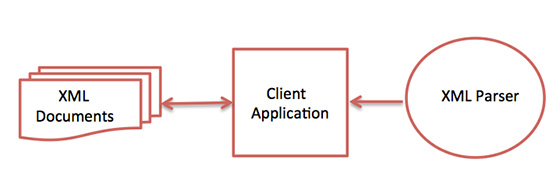
a. customErrors section

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

1]what is XMLparser

a. XML parser is a software library or a package that provides interface for client applications to work with XML documents. It checks for proper format of the XML document and may also validate the XML documents. ... The goal of a parser is to transform XML into a readable code.

Following diagram shows how XML parser interacts with XML document −



2]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

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

1]explain tracing with page llevel and application level

A. ASP.NET tracing enables you to view diagnostic information about a single request for an ASP.NET page. ASP.NET tracing enables you to follow a page's execution path, display diagnostic information at run time, and debug your application. ASP.NET tracing can be integrated with system-level tracing to provide multiple levels of tracing output in distributed and multi-tier applications.

 There are two ways:

 (i) Page Level Tracing

(ii) Application Level Tracing

**(i) Page Level Tracing**

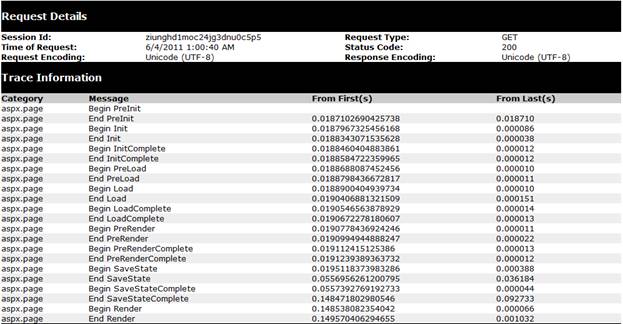
We can control whether tracing is enabled or disabled for individual pages. If tracing is enabled, when the page is requested, ASP.NET appends to the page a series of tables containing execution details about the page request. Tracing is disabled by default in an ASP.NET application.

**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.

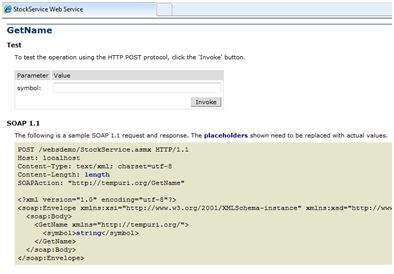
 By default, when the trace viewer reaches its request limit, the application stops storing trace requests. However, you can configure application-level tracing to always store the most recent tracing data, discarding the oldest data when the maximum number of requests is reached.



2]what is web service?explain with example

a. . A Web Service is a software program that uses XML to exchange information with other software via common internet protocols. In a simple sense, Web Services are a way for interacting with objects over the Internet.

A web service is

* Language Independent.
* Protocol Independent.
* Platform Independent.
* It assumes a stateless service architecture.
* Scalable (e.g. multiplying two numbers together to an entire customer-relationship management system).
* Programmable (encapsulates a task).
* Based on XML (open, text-based standard).
* Self-describing (metadata for access and use).
* Discoverable (search and locate in registries)- ability of applications and developers to search for and locate desired Web services through registries. This is based on UDDI.
* Take the following steps to create the web service:
* **Step (1)** : Select File -> New -> Web Site in Visual Studio, and then select ASP.NET Web Service.
* **Step (2)** : A web service file called Service.asmx and its code behind file, Service.cs is created in the App\_Code directory of the project.
* **Step (3)** : Change the names of the files to StockService.asmx and StockService.cs.
* **Step (4)** : The .asmx file has simply a WebService directive on it:
* <%@ WebService Language="C#" CodeBehind="~/App\_Code/StockService.cs" Class="StockService" %>
* **Step (5)** : Open the StockService.cs file, the code generated in it is the basic Hello World service. The default web service code behind file looks like the following:
* using System;
* using System.Collections;
* using System.ComponentModel;
* using System.Data;
* using System.Linq;
* using System.Web;
* using System.Web.Services;
* using System.Web.Services.Protocols;
* using System.Xml.Linq;
* namespace StockService
* {
* // <summary>
* // Summary description for Service1
* // <summary>
* [WebService(Namespace = "http://tempuri.org/")]
* [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]
* [ToolboxItem(false)]
* // To allow this Web Service to be called from script,
* // using ASP.NET AJAX, uncomment the following line.
* // [System.Web.Script.Services.ScriptService]
* public class Service1 : System.Web.Services.WebService
* {
* [WebMethod]
* public string HelloWorld()
* {
* return "Hello World";
* }
* }
* }
* **Step (6)** : Change the code behind file to add the two dimensional array of strings for stock symbol, name and price and two web methods for getting the stock information.
* using System;
* using System.Linq;
* using System.Web;
* using System.Web.Services;
* using System.Web.Services.Protocols;
* using System.Xml.Linq;
* [WebService(Namespace = "http://tempuri.org/")]
* [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]
* // To allow this Web Service to be called from script,
* // using ASP.NET AJAX, uncomment the following line.
* // [System.Web.Script.Services.ScriptService]
* public class StockService : System.Web.Services.WebService
* {
* public StockService () {
* //Uncomment the following if using designed components
* //InitializeComponent();
* }
* string[,] stocks =
* {
* {"RELIND", "Reliance Industries", "1060.15"},
* {"ICICI", "ICICI Bank", "911.55"},
* {"JSW", "JSW Steel", "1201.25"},
* {"WIPRO", "Wipro Limited", "1194.65"},
* {"SATYAM", "Satyam Computers", "91.10"}
* };
* [WebMethod]
* public string HelloWorld() {
* return "Hello World";
* }
* [WebMethod]
* public double GetPrice(string symbol)
* {
* //it takes the symbol as parameter and returns price
* for (int i = 0; i < stocks.GetLength(0); i++)
* {
* if (String.Compare(symbol, stocks[i, 0], true) == 0)
* return Convert.ToDouble(stocks[i, 2]);
* }
* return 0;
* }
* [WebMethod]
* public string GetName(string symbol)
* {
* // It takes the symbol as parameter and
* // returns name of the stock
* for (int i = 0; i < stocks.GetLength(0); i++)
* {
* if (String.Compare(symbol, stocks[i, 0], true) == 0)
* return stocks[i, 1];
* }
* return "Stock Not Found";
* }
* }
* **Step (7)** : Running the web service application gives a web service test page, which allows testing the service methods.
* 
* **Step (8)** : Click on a method name, and check whether it runs properly.
* 
* **Step (9)** : For testing the GetName method, provide one of the stock symbols, which are hard coded, it returns the name of the stock
* 

## Consuming the Web Service

* For using the web service, create a web site under the same solution. This could be done by right clicking on the Solution name in the Solution Explorer. The web page calling the web service should have a label control to display the returned results and two button controls one for post back and another for calling the service.
* The content file for the web application is as follows:
* <%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="wsclient.\_Default" %>
* <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
* <html xmlns="http://www.w3.org/1999/xhtml" >
* <head runat="server">
* <title>
* Untitled Page
* </title>
* </head>
* <body>
* <form id="form1" runat="server">
* <div>
* <h3>Using the Stock Service</h3>
* <br /> <br />
* <asp:Label ID="lblmessage" runat="server"></asp:Label>
* <br /> <br />
* <asp:Button ID="btnpostback" runat="server" onclick="Button1\_Click" Text="Post Back" style="width:132px" />
* <asp:Button ID="btnservice" runat="server" onclick="btnservice\_Click" Text="Get Stock" style="width:99px" />
* </div>
* </form>
* </body>
* </html>
* The code behind file for the web application is as follows:
* using System;
* using System.Collections;
* using System.Configuration;
* using System.Data;
* using System.Linq;
* using System.Web;
* using System.Web.Security;
* using System.Web.UI;
* using System.Web.UI.HtmlControls;
* using System.Web.UI.WebControls;
* using System.Web.UI.WebControls.WebParts;
* using System.Xml.Linq;
* //this is the proxy
* using localhost;
* namespace wsclient
* {
* public partial class \_Default : System.Web.UI.Page
* {
* protected void Page\_Load(object sender, EventArgs e)
* {
* if (!IsPostBack)
* {
* lblmessage.Text = "First Loading Time: " + DateTime.Now.ToLongTimeString
* }
* else
* {
* lblmessage.Text = "PostBack at: " + DateTime.Now.ToLongTimeString();
* }
* }
* protected void btnservice\_Click(object sender, EventArgs e)
* {
* StockService proxy = new StockService();
* lblmessage.Text = String.Format("Current SATYAM Price:{0}",
* proxy.GetPrice("SATYAM").ToString());
* }
* }
* }

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

1]explain reading & writing dataset with XML.

a. Here in this blog we will see how to first write data from data set into a XML file and then how to read that XML file data into a Dataset.

To elaborate this we will use two grids on the page.

**Step1:**

Create a web page and put two Button

1. Button to write XML ( named btnImport)
2. Button to read XML ( named btnExport)

and also two grids

1. To show written data  (named grdImport)
2. To show Read data  ( named grdExport)

Here is the HTML Code (ReadWriteXML.aspx page) for the program

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="ReadWriteXML.aspx.cs" Inherits="Pages\_ReadWriteXML" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head id="Head1" runat="server">

    <title></title>

</head>

<body>

    <form id="form1" runat="server">

    <div>

        <table>

            <tr>

                <td>

                    <asp:Button ID="btnImport" runat="server" Text="Write XML" OnClick="btnImport\_Click" />

                </td>

            </tr>

            <tr>

                <td>

                </td>

            </tr>

            <tr>

                <td>

                    <asp:GridView ID="grdImport" runat="server">

                    </asp:GridView>

                </td>

            </tr>

            <tr>

                <td>

                </td>

            </tr>

            <tr>

                <td>

                    <asp:Button ID="btnExport" runat="server" Text="Read XML" OnClick="btnExport\_Click" />

                </td>

            </tr>

            <tr>

                <td>

                </td>

            </tr>

            <tr>

                <td>

                    <asp:GridView ID="grdExport" runat="server">

                    </asp:GridView>

                </td>

            </tr>

        </table>

    </div>

    </form>

</body>

</html>

**Code for ReadWriteXML.aspx.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

using System.Data.SqlClient;

using Database.DAL;

public partial class Pages\_ReadWriteXML : System.Web.UI.Page

{

    Common objCommon = new Common();

    DataSet DS;

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

    {

    }

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

    {

        string FileName = "TEST\_XML.xml";

        string FilePath = Server.MapPath("~/XML\_Files/");

        using (SqlCommand cmd = new SqlCommand())

        {

            cmd.CommandText = "READ\_XML\_EMP";

            DS = DBHelper.SqlExecuteAdapter(cmd);

        }

        objCommon.Write\_XML(DS, FileName, FilePath);

        grdImport.DataSource = DS;

        grdImport.DataBind();

    }

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

    {

        string FileName = "TEST\_XML.xml";

        string FilePath = Server.MapPath("~/XML\_Files/");

        DS = objCommon.Read\_XML(FileName, FilePath);

        grdExport.DataSource = DS;

        grdExport.DataBind();

    }

}

**DBHelper Class**

using System;

using System.Data.SqlClient;

using System.Data;

using System.Security.Cryptography;

using System.IO;

using System.Web;

using System.Web.UI;

namespace Database.DAL

{

    /// <summary>

    /// Summary description for SqlHelper

    /// </summary>

    public class DBHelper

    {

        private DBHelper()

        {

            //

            // TODO: Add constructor logic here

            //

        }

        #region Connection String

        //Pick connection string from web.config

        /\*

               which could be:

               <connectionStrings>

               <add name="Connection" connectionString="Data Source=TESTSQL;Initial Catalog=TESTDB;uid=sa; pwd=TEST123;Connect Timeout=400;pooling='true'; MaxPool Size=100;" providerName="System.Data.SqlClient;"/>

               </connectionStrings>

        \*/

        private static string conn = System.Configuration.ConfigurationManager.ConnectionStrings["Connection"].ToString();

        #endregion

        #region Execute Adapter

        public static DataSet SqlExecuteAdapter(SqlCommand cmd)

        {

            DataSet ds = new DataSet();

            cmd.CommandType = CommandType.StoredProcedure;

            using (SqlConnection con = new SqlConnection(conn))

            {

                cmd.Connection = con;

                con.Open();

                SqlDataAdapter adapt = new SqlDataAdapter(cmd);

                adapt.Fill(ds);

                con.Close();

            }

            return ds;

        }

        #endregion

    }

}

**Common class**

using System;

using System.Data;

using System.Configuration;

using System.Collections;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Web.UI.HtmlControls;

using System.Data.SqlClient;

using Database.DAL;

using System.Xml;

using System.Xml.Serialization;

using System.IO;

namespace Database.DAL

{

    public class Common

    {

        public Common()

        {

            //

            // TODO: Add constructor logic here

            //

        }

        #region READ WRITE XML

        public void Write\_XML(DataSet DS, string FileName, string FilePath)

        {

            StreamWriter XMLFile = new StreamWriter(FilePath + FileName, false);

            DS.WriteXml(XMLFile);

            XMLFile.Close();

        }

        public DataSet Read\_XML(string FileName, string FilePath)

        {

            DataSet DS = new DataSet();

            DS.ReadXml(FilePath + FileName);

            return DS;

        }

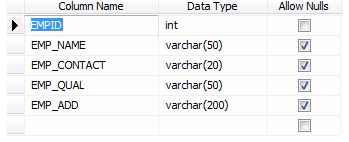
        #endregion

    }

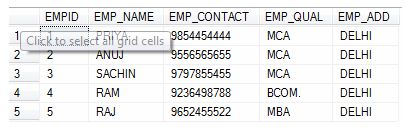
}

**DB Structure**

**TEST\_XML table**



INSERT VALUES IN TABLE LIKE



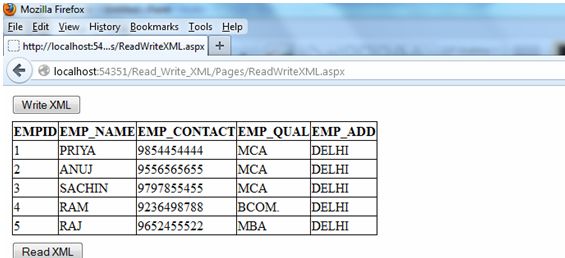
READ\_XML\_EMP PROCEDUER

USE [TESTDB]

GO  
/\*\*\*\*\*\* Object:  StoredProcedure [dbo].[READ\_XML\_EMP]    Script Date: 07/11/2013 17:11:26 \*\*\*\*\*\*/  
SET ANSI\_NULLS ON  
GO  
SET QUOTED\_IDENTIFIER ON  
GO  
CREATE PROC [dbo].[READ\_XML\_EMP]  
AS  
BEGIN  
SELECT \* FROM TEST\_XML  
END  
GO

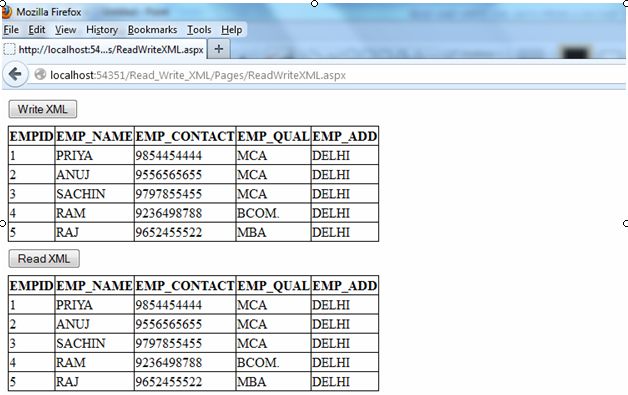
FIRST CLICK ON WRITE XML Button to Write Data INTO XML File

**Output will be:**



Click on Export button to read data from XML file

**Output Will be:**



2]what is authentication & authorization

* a. *Authentication* is knowing the identity of the user. For example, Alice logs in with her username and password, and the server uses the password to authenticate Alice.
* *Authorization* is deciding whether a user is allowed to perform an action. For example, Alice has permission to get a resource but not create a resource.
* Windows Authentication
* Forms Authentication
* Passport Authentication
* Custom Authentication

The windows Authentication provider lets you authenticates users based on their windows accounts. This provider uses IIS to perform the authentication and then passes the authenticated identity to your code. This is the default provided for ASP.net.

The passport authentication provider uses Microsoft's passport service to authenticate users.

The forms authentication provider uses custom HTML forms to collect authentication information and lets you use your own logic to authenticate users. The user's credentials are stored in a cookie for use during the session.

**Windows authentication and IIS**

If you select windows authentication for your ASP.NET application, you also have to configure authentication within IIS. This is because IIS provides Windows authentication. IIS gives you a choice for four different authentication methods:

 Anonymous, basic digest, and windows integrated

 If you select anonymous authentication, IIS doesn't perform any authentication, Any one is allowed to access the ASP.NET application.

If you select basic authentication, users must provide a windows username and password to connect. How ever this information is sent over the network in clear text, which makes basic authentication very much insecure over the internet.

If you select digest authentication, users must still provide a windows user name and password to connect. However the password is hashed before it is sent across the network. Digest authentication requires that all users be running Internet Explorer 5 or later and that windows accounts to stored in active directory.

 Passport authentication

Passport authentication lets you to use Microsoft's passport service to authenticate users of your application. If your users have signed up with passport, and you configure the authentication mode of the application to the passport authentication, all authentication duties are offloaded to the passport servers.

Passport uses an encrypted cookie mechanism to indicate authenticated users. If users have already signed into passport when they visit your site, they'll be considered authenticated by ASP.NET. Otherwise they'll be redirected to the passport servers to log in. When they are successfully log in, they'll be redirected back to your site.

To use passport authentication you have to download the Passport Software Development Kit (SDK) and install it on your server. The SDK can be found at http://msdn.microdoft.com/library/default.asp?url=/downloads/list/websrvpass.aps. It includes full details of implementing passport authentication in your own applications.

Forms authentication

Forms authentication provides you with a way to handle authentication using your own custom logic with in an ASP.NET application. The following applies if you choose forms authentication.

1. When a user requests a page for the application, ASP.NET checks for the presence of a special session cookie. If the cookie is present, ASP.NET assumes the user is authenticated and processes the request.
2. If the cookie isn't present, ASP.NET redirects the user to a web form you provide
3. You can carry out whatever authentication, checks you like in your form. When the user is authenticated, you indicate this to ASP.NET by setting a property, which creates the special cookie to handle subsequent requests.

Configuring Authorization

After your application has authenticated users, you can proceed to authorize their access to resources. But there is a question to answer first: Just who is the user to whom your are grating access? It turns out that there are different answers to that question, depending on whether you implement impersonation. Impersonation is a technique that allows the ASP.NET process to act as the authenticated user, or as an arbitrary specified user.

ASP.NET impersonation is controlled by entries in the applications web.config file. The default setting is "no impersonation". You can explicitly specify that ASP.NET shouldn't use impersonation by including the following code in the file

*<identity impersonate="false"/>*

With this setting ASP.NET does not perform impersonation. It means that ASP.NET will runs with its own privileges. By default ASP.NET runs as an unprivileged account named ASPNET. You can change this by making a setting in the processModel section of the machine.config file. When you make this setting, it automatically applies to every site on the server. To user a high-privileged system account instead of a low-privileged, set the userName attribute of the processModel element to SYSTEM. Using this setting is a definite security risk, as it elevates the privileges of the ASP.NET process to a point where it can do bad things to the operating system.

When you disable impersonation, all the request will run in the context of the account running ASP.NET: either the ASPNET account or the system account. This is true when you are using anonymous access or authenticating users in some fashion. After the user has been authenticated, ASP.NET uses it own identity to request access to resources.

The second possible setting is to turn on impersonation.

*<identity impersonate="true"/>*

In this case, ASP.NET takes on the identity IIS passes to it. If you are allowing anonymous access in IIS, this means ASP.NET will impersonate the IUSR\_ComputerName account that IIS itself uses. If you aren't allowing anonymous access,ASP.NET will take on the credentials of the authenticated user and make requests for resources as if it were that user. Thus by turning impersonation on and using a non-anonymous method of authentication in IIS, you can let users log on and use their identities within your ASP.NET application.

Finally, you can specify a particular identity to use for all authenticated requests

1. <identity impersonate="true" username="DOMAIN\username" password="password"/>

With this setting, all the requests are made as the specified user (Assuming the password it correct in the configuration file). So, for example you could designate a user for a single application, and use that user's identity every time someone authenticates to the application. The drawback to this technique is that you must embed the user's password in the web.config file in plain text. Although ASP.NET won't allow anyone to download this file, this is still a security risk if anyone can get the file by other means.

**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