

1ST Edition

ASP.NET With C#

HandBook For B.Sc.IT Student (Mumbai University)

BY



KAMAL THAKUR

- THIS IS THE FIRST EDITION.
- THIS BOOK IS ESPECIALLY FOR "MUMBAI UNIVERSITY – B.SC. (IT)" STUDENT.
- YOU'LL GET ALL QUESTION + SOLUTION WHICH ARE ASK IN "MUMBAI UNIVERSITY" EXAMINATION.
- THIS HANDBOOK IS FOR ALL COURSE/SYLLABUS (75:25 PATTERN | REVISED COURSE | 60:40 PATTERN).
- UNIT WISE + YEAR WISE + QUESTION WISE + SYLLABUS WISE
- ALL PREVIOUS YEAR QUESTION PAPER WITH SOLUTION SET (2013 – 2018).
- EASY INDEX MADE FOR SEARCHING ANY QUESTION.
- IF YOU'VE A FREE TIME THEN PLEASE VISIT MY BLOG @ "WWW.MUMBAIBSCITSTUDY.COM".



- 🌐 **Official Sites:** → www.mumbaibscitstudy.com
- 📘 **Facebook:** → <https://facebook.com/mumbaibscitstudy>
- 📷 **Instagram:** → <https://instagram.com/mumbaibscitstudy>
- 📺 **YouTube:** → <http://bit.do/KamalT>
- 👤 **Google+:** → <https://plus.google.com/+KamalTUniverse>
- 📄 **Blogger:** → <https://mumbaibscitstudy.blogspot.com/>
- ✉ **Gmail:** → kamalthakurbcsit@gmail.com

ASP.NET With C#

HandBook For B.Sc.IT Student (Mumbai University)

BY

Kamal Thakur

B.Sc.IT (Mumbai University)

Web Designer | Blogger | YouTuber | E-Books Designer & Maker

Contact & Follow Us

Email ID: kamalthakurbcit@gmail.com

WhatsApp No.: +91 – 8454975016

Websites:

1. **Blogger:** www.mumbaibscitstudy.blogspot.com
2. **HostFree:** www.mumbaibscitstudy.hostfree.pw
3. **Tumblr:** www.kamaltuniverse.tumblr.com

Social Networking:

1. **Facebook:** www.facebook.com/mumbaibscitstudy
2. **Instagram:** www.instagram.com/mumbaibscitstudy

YouTube Channel:

Mumbai B.Sc.IT Study: <http://bit.do/KamalT>

Contact Me @

Facebook: <https://facebook.com/kamaltuniverse>

Instagram: <https://www.instagram.com/mumpsytance>

Twitter: <https://twitter.com/kamaltuniverse>

Tumblr: <https://kamaltuniverse.tumblr.com>

Pinterest: <https://in.pinterest.com/kamaltuniverse>

LinkedIn: <https://linkedin.com/in/skmkamal>

Created and Made . . .

With The Help & Idea Of . . .

My Father. . .

Jawahar Thakur

G.D Architecture

Preface

- This Book is especially for "Mumbai University – B.Sc. (IT)" Student.
- This is the first edition.
- You'll get All Question + Solution which are ask in "Mumbai University" Examination.
- This Handbook is for all Course/Syllabus (75:25 Pattern | Revised Course | 60:40 Pattern).
- Unit Wise + Year Wise + Question Wise + Syllabus Wise
- All Previous Year Question Paper with Solution Set (2013 – 2018).
- Easy Index made for searching any question.
- If you've a free time then please visit my Blog @ "www.MumbaiBscitStudy.com".
- This book also contain some Programming Question with Solution.
- This book will also help for Interview Question of ASP.NET with C#.

Reference

<https://agile-code.com>
<https://docs.microsoft.com>
<https://msdn.microsoft.com>
<https://c-sharpcorner.com>
<https://way2tutorial.com>
<https://allinterview.com>
<https://javatpoint.com>
<https://programiz.com>
<https://differencebtw.com>
<https://coursehero.com>
<https://w3schools.com>
<https://mindstick.com>
<https://freefeast.info>
<https://quora.com>
<https://telerik.com>
<https://gsys.biz>
<https://social.msdn.microsoft.com>
<https://dotnettricks.com>
<http://tutorialsteacher.com>
<http://dotnetfunda.com>
<http://claudiobernasconi.ch>
<http://triconsole.com>
<http://onlydifferencefaqs.blogspot.com>
<http://csharp.net-informations.com>
<http://helpingdotnet.blogspot.com>
<http://sandeep-tada.blogspot.com>
<https://api.jquery.com>



— : SYLLABUS : —

UNIT

REVIEW OF .NET FRAMEWORKS | INTRODUCTION TO C# | C# REFERENCE TYPES

I

- ⇒ Review Of .NET Frameworks
- ⇒ Introduction To C#
- ⇒ Variables And Expressions
- ⇒ Flow Controls
- ⇒ Functions
- ⇒ Debugging And Error Handling
- ⇒ OOPs With C#
- ⇒ Defining Classes And Class Members

UNIT

WINDOWS PROGRAMMING

II

- ⇒ Assembly
 - Components Of Assembly
 - Private And Shared Assembly
- ⇒ Garbage Collector
- ⇒ JIT Compiler
- ⇒ Namespaces Collections
- ⇒ Delegates And Events
- ⇒ Introduction To ASP.NET 4
 - Microsoft.NET Framework
 - ASP.NET Lifecycle

UNIT

ASP.NET APPLICATION | CSS | ASP.NET SERVER CONTROLS

III

- ⇒ ASP.NET Server Controls
 - Introduction
 - How To Work With Button Controls
 - Textboxes
 - Labels
 - Checkboxes And Radio Buttons
 - List Controls And Other Web
 - Server Controls
 - web.config and global.asax Files
- ⇒ Programming ASP.NET Web Pages
 - Introduction
 - Data Types And Variables
 - Statements
 - Organizing Code
 - Object Oriented Basics
- ⇒ Cascading Style Sheet (CSS)
 - Need Of CSS
 - Introduction To CSS
 - Working With CSS With Visual Developer

**UNIT****IV****PROGRAMMING ASP.NET WEB PAGES | NAVIGATION AND USER CONTROL**

- ⇒ **Validation Control**
 - Introduction
 - Basic Validation Controls
 - Validation Techniques
 - Using Advanced Validation Controls
- ⇒ **State Management**
 - Using View State
 - Using Session State
 - Using Application State
 - Using Cookies And URL Encoding
- ⇒ **Master Pages**
 - Creating Master Pages
 - Content Pages
 - Nesting Master Pages
 - Accessing Master Page Controls From A Content Page
- ⇒ **Navigation**
 - Introduction To Use The Site Navigation
 - Using Site Navigation Controls

UNIT**V****DATABASE AND ADO.NET | ASP.NET SECURITY**

- ⇒ **Databases**
 - Introduction
 - Using SQL Data Sources
 - GridView Control
 - DetailsView And FormView Controls
 - ListView And DataPager Controls
 - Using Object Datasources
- ⇒ **ASP.NET Security**
 - Authentication
 - Authorization
 - Impersonation
 - ASP.NET Provider Model

UNIT**VI****ASP.NET AJAX | JQUERY**

- ⇒ **LINQ**
 - Operators
 - Implementations
 - LINQ To Objects
 - XML
 - ADO.NET
 - Query Syntax
- ⇒ **ASP.NET AJAX**
 - Introducing AJAX
 - Working Of AJAX
 - Using ASP.NET AJAX Server Controls
- ⇒ **jQuery**
 - Introduction To JQuery
 - JQuery UI Library
 - Working Of JQuery

**— : CONTENTS : —**

UNIT	REVIEW OF .NET FRAMEWORKS INTRODUCTION TO C# C# REFERENCE TYPES	12 – 75
I		
Q.1.	Explain the three Layer Architecture of ASP.NET.....	14
Q.2.	Draw and Explain .NET Framework Architecture.....	15
Q.3.	What is .NET Framework? Explain various components of .NET Framework 4.0.....	15
Q.4.	Write a short notes on .NET Framework.....	15
Q.5.	What is .NET Framework? What is in the .NET Framework?.....	15
Q.6.	Write short note on Common Language Runtime (CLR) in .NET.....	16
Q.7.	Describe the roles of CLR in .NET Framework.....	16
Q.8.	What is Microsoft Intermediate Language (MSIL)? Why is important?.....	17
Q.9.	Write a short note on CIL Compiler.....	17
Q.10.	Explain the terms JIT Compiler.....	18
Q.11.	Explain the different parts that constitute ASP.NET application.....	18
Q.12.	Explain Framework Base Class Library.....	19
Q.13.	What is Namespace?.....	20
Q.14.	Explain System Namespace.....	20
Q.15.	How to create namespace and its alias?.....	20
Q.16.	Explain the concept of Namespaces with suitable example.....	20
Q.17.	List and explain the use of any five Namespaces in C#.....	20
Q.18.	Write a short note on the System.Collections Namespace.....	20
Q.19.	What is Assembly? Explain the purpose of Assemblies in .NET Framework.....	22
Q.20.	Write a short note on Assembly.....	22
Q.21.	Explain the components of Assembly.....	23
Q.22.	What is the significance of Assemblies in .NET?.....	23
Q.23.	Explain the concept of Private and Shared Assembly.....	24
Q.24.	Differentiate between Private Assembly and Shared Assembly.....	24
Q.25.	What is Garbage Collection?.....	25
Q.26.	How Garbage Collection works?.....	25
Q.27.	Explain three types of generations in Garbage Collection.....	25
Q.28.	Explain Compiling and Execution of a C# Program.....	26
Q.29.	List and Explain the various Primitive Data Types used in C#.....	26
Q.30.	What are enumerations? Explain it with the help of an example.....	28
Q.31.	Why is there a necessity of Enumerator Data Type in C#?.....	29
Q.32.	Explain the concept of Boxing and Unboxing.....	29
Q.33.	Is it a must to Unbox a Boxed Variable? Explain.....	29
Q.34.	Explain Boxing and Unboxing with Reference to Value Type and References Type.....	29
Q.35.	What are Boxing and Unboxing? How is it achieved?.....	29
Q.36.	What are Indexers? Explain with example.....	30
Q.37.	Write the Naming Rules and Conventions of the Variable.....	31
Q.38.	Explain the Comparison and Logical Operators in C#.....	32
Q.39.	What is Regular Expression? Explain Regex Patterns.....	34
Q.40.	What is Flow Control?.....	35
Q.41.	Explain use of Break and Continue Statements in loop.....	35
Q.42.	Explain the different keywords used for interrupting the Loops.....	35
Q.43.	What is the difference between for loop and foreach loop?.....	37
Q.44.	Explain foreach loop with proper syntax and example.....	38



Q.45.	Give general form of Switch Statement. Explain with example.....	39
Q.46.	Explain Switch Statement. What is Fallthrough in Switch? Is Fallthrough permitted in C#?.....	41
Q.47.	Explain Write() and WriteLine() Methods with examples.....	42
Q.48.	Are multiple Main() methods allowed in C#? Justify with an example.....	43
Q.49.	What are Sealed Methods? Why are they used?.....	43
Q.50.	What is the difference between Read() and ReadLine()? Explain with an Example.....	44
Q.51.	Explain the difference between Response.Redirect() and Server.Transfer() method in ASP.NET.....	45
Q.52.	What is an Exception? Explain Exception Handling in C#.....	46
Q.53.	Explain briefly try, catch and throw statements in C#.....	46
Q.54.	What is Exception Handling? Explain the Syntax of Exception Handling Code. What is the use of finally Block?.....	46
Q.55.	Write a short note on Exception Handling in C#.....	46
Q.56.	Why Exception Handling is required? Write syntax for user define Exception?.....	48
Q.57.	What is a User Defined Exception? Explain with Syntax.....	48
Q.58.	Explain Variable Sized Arrays with suitable example.....	49
Q.59.	What is a Jagged Array? Explain with example.....	49
Q.60.	Write a note on "Array Of Arrays".....	49
Q.61.	What is ArrayList? State its methods and properties.....	50
Q.62.	What is the difference between Arrays and ArrayLists? Explain with a program.....	51
Q.63.	Explain References Parameters and Output Parameters with a program.....	52
Q.64.	Explain the different Access Modifiers.....	53
Q.65.	Define each of the following terms:..... (i) Derived Class (ii) Abstract Class (iii) Static Class (iv) Sealed Class (v) Partial Class	54
Q.66.	What are Sealed Classes? Why are they used?.....	55
Q.67.	List and Explain Properties of Page Class.....	55
Q.68.	What is the scope of protected Data Members of a class?.....	56
Q.69.	Differentiate between Structures and Classes.....	57
Q.70.	Explain Request and Response Objects of ASP.NET.....	57
Q.71.	What are Interfaces? Write a program to show the implementation of Multiple Interfaces.....	59
Q.72.	Distinguish between Abstract Class and Interface.....	61
Q.73.	Explain the similarities between Interfaces and Abstract Classes.....	61
Q.74.	When is Explicit Interface necessary?.....	61
Q.75.	What is Constructor? Explain Parameterized Constructor with suitable example.....	62
Q.76.	What are the different types of constructors in C#?.....	62
Q.77.	Explain Static Constructor with example.....	62
Q.78.	What are the rules in defining a Constructor?.....	62
Q.79.	Define Inheritance and Polymorphism.....	67
Q.80.	Explain how Multiple Inheritance is supported by classes in C#.....	67
Q.81.	Explain how Multiple Inheritance is achieved using interfaces.....	67
Q.82.	What is Polymorphism? Explain Runtime Polymorphism in C#.....	67
Q.83.	Explain with an example the concept of Inheritance.....	67
Q.84.	Explain with an example the concept of Method Overloading?.....	71
Q.85.	What are the steps involved for the selection of a method in Method Overloading?....	71
Q.86.	What is Method Overriding? Give an example of it.....	73
Q.87.	What is the difference between Overriding Methods and Hiding Methods?.....	74
Q.88.	Explain Method Hiding with example.....	74
Q.89.	Differentiate between Function Overloading and Function Overriding.....	75
Q.90.	Write a short note on Properties.....	75
Q.91.	Name and describe any 5 basic Properties in ASP.NET.....	75
Q.92.	What is a Property? Why are they referred to as Smart Fields?.....	75



| UNIT |
II

COLLECTION | CONVERSIONS | DELEGATE | EVENT
WINDOWS PROGRAMMING

79 – 107

Q.93.	What are the different types of Collections in .NET? Explain.....	80
Q.94.	What are Collections? Write different Collections support by C#.....	80
Q.95.	Explain HashTable Collection with an example.....	81
Q.96.	What is Type Casting?.....	82
Q.97.	Explain Implicit and Explicit Type Conversion with examples.....	82
Q.98.	What is Type Conversion in C#? How is it carried out in C#?.....	82
Q.99.	Write a short note on Conversions.....	82
Q.100.	What is Delegate?.....	85
Q.101.	State its Declaration, Instantiation and Invocation Process.....	85
Q.102.	Explain the steps to implement Delegates in C# .NET.....	85
Q.103.	What are the two types of delegates? Explain each with an example.....	85
Q.104.	Delegates in C# are used for Event Handling.....	85
Q.105.	What is an Event? What are the differences between Delegate and Event?.....	89
Q.106.	Write a short note on GUI. State its advantages.....	89
Q.107.	What is the difference between Button, LinkButton and ImageButton? Explain any three Common Button Attributes.....	90
Q.108.	Explain LinkButton Controls and ImageButton Controls.....	90
Q.109.	What is the difference between Button and LinkButton Web Server Controls?.....	90
Q.110.	Explain the TextBox Control with any Five Properties.....	92
Q.111.	List and explain any four TextBox Attributes.....	92
Q.112.	Explain Label Control.....	91
Q.113.	Explain CheckBox Controls.....	93
Q.114.	Explain the following properties/events of CheckBox control:..... (1) ID (2) AccessKey (3) Text (4) Checked (5) CheckedChanged()	94
Q.115.	Explain RadioButton Control.....	94
Q.116.	Explain GroupBox Control.....	96
Q.117.	What is the difference between CheckBox and RadioButton Control? What are the common attributes associated with these controls?.....	97
Q.118.	What is the difference between RadioButton and CheckButton Controls?.....	97
Q.119.	When is CheckedChanged Event of a CheckBox fired?..... <i>Describe the following properties:</i> (a) GroupName property of a RadioButton (b) Text property of a Label (c) TextMode property of a TextBox (d) Checked property of a RadioButton	97
Q.120.	Explain properties of ListBox Controls.....	98
Q.121.	Explain the properties of DropDownList Control.....	99
Q.122.	Explain RadioButtonList Control with its properties and methods.....	100
Q.123.	What is the difference between RadioButton and RadioButtonList Control?.....	101
Q.124.	Explain the similarities and differences between ListBox and DropDownList.....	102
Q.125.	What is the difference between ListBox and DropDownList? List and explain any three common properties of these Controls.....	102
Q.126.	Differentiate between ListBox and ComboBox Controls.....	103
Q.127.	Explain any five Methods / Properties of ListItem Collection Object.....	103
Q.128.	What is Common DialogBox? Explain FontDialog with suitable example.....	104
Q.129.	What are the advantages of using Common DialogBox?.....	104
Q.130.	What is the use of Menus and Toolbars in Windows Application? Explain.....	105
Q.131.	List and explain the properties of ToolStripMenuItem.....	105
Q.132.	Explain StatusStrip Control.....	106



Q.133. Explain the concept of MDI in windows Application. Also explain how to define child forms?.....	107
Q.134. What is the difference between SDI and MDI?.....	107
Q.135. Write a short note on MDI.....	107

UNIT III	ASP.NET APPLICATIONS CSS (CASCAIDING STYLE SHEET) ASP.NET SERVER CONTROLS	109 – 149
-------------------------------	--	------------------

Q.136. Explain the ASP.NET Life Cycle.....	111
Q.137. Explain the Event Life Cycle of ASP.NET?.....	111
Q.138. Explain ASP.NET Page Life Cycle.....	111
Q.139. Explain the different Directives supported by ASP.NET.....	114
Q.140. What is the place of Page Directive? Explain its attributes.....	116
Q.141. Explain src, tagprefix and tagname attributes of @ register directive.....	116
Q.142. Explain the Code Render Block with suitable example.....	117
Q.143. What is the difference between Themes and Skins in ASP.NET?.....	118
Q.144. What are Skins and why are they used?.....	118
Q.145. What are Themes? Explain its different types.....	118
Q.146. Explain procedure of creating and using named theme in a website.....	119
Q.147. What is CSS?.....	120
Q.148. Explain Inline Style Sheet.....	120
Q.149. Explain External Style Sheet.....	120
Q.150. Explain Embedded Style Sheet.....	120
Q.151. How would you link an HTML Page to an External Style Sheet?.....	120
Q.152. Explain the different types of CSS in ASP.NET.....	120
Q.153. What is the advantages and disadvantages of CSS?.....	120
Q.154. What is the need of CSS in ASP.NET?.....	120
Q.155. List and explain the advantages of CSS over HTML.....	120
Q.156. What is Selector in CSS? Explain various types of CSS Selectors.....	123
Q.157. What are the different types of Selectors present in jQuery? Explain.....	123
Q.158. Explain Class Selector and ID Selector with the help of an example.....	123
Q.159. How are the Basic Selector of CSS used in jQuery?.....	128
Q.160. How to create and use External Style Sheet using Visual Studio Developer?.....	129
Q.161. Explain working with CSS using Visual Studio Developer.....	129
Q.162. Compare CSS with HTML.....	300
Q.163. How does CSS fix HTML formatting problems?.....	300
Q.164. Explain <LINK> Tag with example.....	300
Q.165. What are User Controls?.....	131
Q.166. How do we create a User Control in ASP.NET?.....	131
Q.167. What are the different ways to redirect a user to another page programmatically? What's the difference between them?.....	132
Q.168. Explain the ListView Control.....	133
Q.169. List and explain various templates provided by ListView Control.....	133
Q.170. Explain the basic Attributes of ListView Control.....	133
Q.171. Differentiate between HTML Server Controls and Web Server Controls.....	134
Q.172. What is the difference between Server Controls and User Controls?.....	134
Q.173. Explain any five Common properties of Web Server Controls.....	135
Q.174. Write a short note on State Management.....	135
Q.175. Write a short note on Server Side State Management.....	135
Q.176. Explain Sessions in ASP.NET.....	135
Q.177. What are four application Events when an Application is executed?.....	135



Q.178. Explain the work of Session State in ASP.NET.....	138
Q.179. Write short descriptions on Session State Variables.....	139
Q.180. What is the difference between Application State and Session State in ASP.NET?.....	139
Q.181. What is ViewState? How it works in ASP.NET?.....	140
Q.182. What is ViewState? Where is the ViewState stored after the page postback?.....	140
Q.183. List the different places in a Web Application where ViewState Field can be Disabled?	140
Q.184. What is the significance of ViewState in ASP.NET?.....	140
Q.185. What is View State? How to preserve View State for user's own data.....	140
Q.186. Explain in brief about Cookies.....	142
Q.187. Explain the role of Cookies in ASP.NET.....	142
Q.188. What are the advantages and disadvantages of using Cookies? Explain how server sets a Cookie and retrieves it.....	142
Q.189. Explain various properties of HttpCookie Class.....	143
Q.190. What is use QueryString? Explain Encoding and Decoding of QueryString.....	144
Q.191. Explain QueryString with example.....	144
Q.192. What is the need of Query String in ASP.NET?.....	145
Q.193. Explain HiddenField Control with example.....	146
Q.194. Explain the global.asax files in ASP.NET Application.....	147
Q.195. List and explain the major events in global.asax file.....	147
Q.196. Explain the use of global.asax files in ASP.NET Application.....	147
Q.197. What are the Event Handlers that we can have in global.asax file?.....	148
Q.198. What is web.config file? Explain its Structure.....	149
Q.199. Explain the use of web.config files in ASP.NET Application.....	149
Q.200. Write a short note on Web.Cong File.....	149
Q.201. What is the importance of web.config File in any Web Application?.....	149

<p>UNIT</p> <hr style="width: 50%; margin: 0 auto;"/> <p>IV</p>	<p>PROGRAMMING ASP.NET WEB PAGES</p> <p>NAVIGATION AND USER CONTROLS</p>	<p>151 – 177</p>
---	--	------------------

Q.202. What is an event? What is an Event Handler? How is it designed?.....	152
Q.203. What is PostBack Event? Explain IsPostBack with suitable example.....	153
Q.204. What is PostBack Event? Explain with suitable example.....	153
Q.205. Explain AutoPostBack.....	154
Q.206. What are the uses of AutoPostBack and runat properties?.....	154
Q.207. Explain AutoEventWiredUp.....	155
Q.208. What is Code-Behind Model in ASP.NET? How is it different from Single File Model?..	155
Q.209. What is the difference between Inline Code and Code Behind?.....	156
Q.210. What is the Code Behind and Inline Code?.....	156
Q.211. What is the difference between .aspx file and .cs file? Explain with an example for each.....	156
Q.212. Explain relationship between Content Page and Master Page.....	157
Q.213. What is the use of MasterPages in ASP.NET? How a Content Page can be added to a Master Page.....	157
Q.214. What are Content Pages?.....	157
Q.215. What is the significance of Master pages in ASP.NET? What is the name given to pages other than Master Pages? Explain in detail.....	157
Q.216. What are the advantages of a Master Pages? Explain about ContentPlaceHolder's in Master Pages.....	557
Q.217. Explain the role of Master Pages in ASP.NET.....	157
Q.218. Explain the steps of creating Master Page in Web Development.....	159
Q.219. What are the three different ways to store data in the Cache Object?.....	162
Q.220. What is a Caching? Explain its types.....	162



Q.221. Define Caching in ASP.NET. List the advantages and disadvantages.....	162
Q.222. Explain Menu Site Navigation Controls.....	163
Q.223. Explain any two Site Navigation Controls in ASP.NET.....	163
Q.224. Explain the TreeView Control.....	164
Q.225. When do we have to use TreeView Control in ASP.NET?.....	164
Q.226. Explain TreeView Site Navigation Controls.....	164
Q.227. Explain any two Site Navigation Controls in ASP.NET.....	164
Q.228. What is Website Navigation?.....	166
Q.229. Explain structure of web.sitemap file with suitable example.....	166
Q.230. Explain a syntax for creating sitemap file with suitable example.....	166
Q.231. List the steps for creating Sitemap File.....	166
Q.232. What is a Sitemap File?.....	166
Q.233. What is SiteMapPath Navigation Controls?.....	168
Q.234. What is the role of the SiteMapDataSource Control?.....	168
Q.235. Why UrlEncode() and UrlDecode() methods are used in ASP.NET? Explain.....	169
Q.236. Explain URL Encoding in detail.....	169
Q.237. Explain UrlEncode() and UrlDecode() methods in ASP.NET.....	169
Q.238. What is difference between Server.Redirect and Server.Transfer? Explain with suitable example.....	170
Q.239. How do we create a User Control in ASP.NET?.....	171
Q.240. Explain Validation Controls with example.....	172
Q.241. Explain RegularExpressionValidator with the help of an example.....	172
Q.242. What is RangeValidator? Describe any four properties of it.....	172
Q.243. Explain CustomValidator control with suitable example.....	172
Q.244. What is the use of Compare Validators? Explain it along with its properties?.....	172
Q.245. How to Use the ASP.NET Validation Control to Validate the User Input.....	175
Q.246. Describe Common Validator properties for different Validation Controls.....	176
Q.247. Write necessary properties which are common for all validation controls.....	176
Q.248. Define the steps to set up Validation Control in an ASP.NET Web Page.....	176
Q.249. Explain CauseValidation properties.....	177

 UNIT V	DATABASES AND ADO.NET LINQ ASP.NET SECURITY	178 – 215
---------------------------------	--	------------------

Q.250. Explain type of joins in SQL Server.....	180
Q.251. Write the necessary steps for connecting to the SQL Server Database.....	181
Q.252. Explain ADO.NET Object Model with help of suitable diagram.....	185
Q.253. Draw and explain the architecture of ADO.NET in brief.....	185
Q.254. Explain SqlDataAdapter Class with properties and methods.....	185
Q.255. Explain SqlConnection Objects in ADO.NET.....	185
Q.256. What is SqlDataSource Control used in ADO.Net?.....	185
Q.257. Explain SqlCommand Object in ADO.NET.....	185
Q.258. Describe DataReader Object of ADO.NET with example.....	185
Q.259. Explain ExecuteReader, ExecuteNonQuery and ExecuteScalar methods.....	185
Q.260. Explain ASP.NET Provider Model with its diagram.....	188
Q.261. What is the difference between DataReader and DataAdapter? Explain.....	189
Q.262. Differentiate between DataSet and DataReader.....	189
Q.263. What is the difference between ExecuteScalar and ExecuteNonQuery?.....	190
Q.264. List all the steps in order to access a database through ADO.NET.....	190
Q.265. Explain in brief Connected and Disconnected Mode of ADO.NET.....	191
Q.266. What is Data Binding? Explain its different types.....	192



Q.267. What is DataBound Control?.....	193
Q.268. Explain in brief about Single Item Control.....	193
Q.269. Explain GridView Control with example.....	194
Q.270. What is the use of GridView Control in ASP.NET?.....	194
Q.271. What is GridView Control? Explain how to enable Row Selection, Paging and Sorting features of GridView.....	194
Q.272. Explain any four properties of GridView Control.....	194
Q.273. Explain in brief about Paging Control.....	194
Q.274. What is a Data Source? Explain various types of Data Sources in ASP.NET.....	195
Q.275. Explain DetailsView Control.....	196
Q.276. Explain FormView Control.....	196
Q.277. Briefly explain FormView Control. How is it different from DetailsView?.....	196
Q.278. What is the difference between ListView and GridView Control?.....	197
Q.279. Write short note on XPath.....	197
Q.280. State the ways of deployment of website in ASP.NET.....	198
Q.281. What is LINQ?.....	199
Q.282. Write the basic syntax of a LINQ Query.....	199
Q.283. Explain its syntax and give its advantages.....	199
Q.284. Explain the LINQ Query Syntax with an example Query.....	199
Q.285. Explain the Structure of a LINQ Query.....	199
Q.286. What are advantages and disadvantages of LINQ?.....	199
Q.287. What are different types of operations in LINQ?.....	201
Q.288. Explain the terms Take, Skip, TakeWhile, SkipWhile, First, FirstOrDefault, Last, LastOrDefault with respect to LINQ.....	204
Q.289. Write a note on first, FirstOrDefault, Last and LastOrDefault.....	204
Q.290. Explain the Standard Query Operators "Select", "From", "OrderBy" and "Where" in LINQ.....	204
Q.291. Explain any four Standard Query Operators in LINQ.....	204
Q.292. What are the different ways to implement LINQ?.....	206
Q.293. How can Query Syntax and Extension Methods be mixed? Explain.....	208
Q.294. What is LINQ to Objects? Explain.....	208
Q.295. Explain with an example LINQ to XML.....	209
Q.296. Explain LINQ to SQL with the help of a Query that performs equijoin.....	210
Q.297. What is difference between SQL and LINQ?.....	211
Q.298. What do you mean by Authentication?.....	211
Q.299. Explain types of Authentication.....	211
Q.300. Explain Passport Authentication in ASP.NET.....	211
Q.301. What are the Authentication Mode in ASP.NET for security?.....	211
Q.302. Explain Windows Authentication in ASP.NET.....	211
Q.303. Write the steps with sample code to use Windows Authentication.....	211
Q.304. What is the Authorization in ASP.NET?.....	214
Q.305. How Authorization is used in case of providing Security to Web Application.....	214
Q.306. What is the difference between Authorisation and Impersonation in terms of Security in ASP.NET?.....	215
Q.307. Write a note on Impersonation.....	215

**| UNIT |****VI****AJAX
JQUERY****216 – 240**

Q.308.	What is AJAX?.....	217
Q.309.	How is the processing of a web page without AJAX different from the processing of a Web Page with AJAX?.....	218
Q.310.	List the advantages and disadvantages of AJAX.....	218
Q.311.	What are the benefits of using AJAX?.....	219
Q.312.	What is the significance of AJAX Websites?.....	220
Q.313.	Explain the working of AJAX.....	220
Q.314.	Write a short note on Architecture of AJAX.....	220
Q.315.	List any five applications where AJAX is incorporated.....	220
Q.316.	Explain the difference between AJAX Page Processing and Traditional Page Processing.....	221
Q.317.	Explain the basic steps for creating AJAX application with ASP.NET.....	222
Q.318.	Explain UpdatePanel Control with example.....	223
Q.319.	Explain about ScriptManager Control.....	224
Q.320.	What role does the ScriptManager Play?.....	224
Q.321.	Explain UpdateProgress Control in Ajax.....	225
Q.322.	Explain the Timer Control with its Attributes.....	226
Q.323.	Explain the use of Timer Control in AJAX.....	226
Q.324.	What are the application services provided in ASP.NET? Explain.....	227
Q.325.	What is Web Service? Explain the basic steps to create a Web Service using ASP.NET with C#.....	228
Q.326.	What are the advantages of Web Services?.....	228
Q.327.	List the necessary steps to publish the website in ASP.NET.....	229
Q.328.	Explain XMLHttpRequest Object with its properties and methods.....	230
Q.329.	What is jQuery? How to use jQuery in ASP Pages?.....	231
Q.330.	Brief the concept of jQuery.....	232
Q.331.	Explain jQuery Expression with example.....	232
Q.332.	What are the features of jQuery?.....	233
Q.333.	What does Dollar Symbols (\$) mean in jQuery? Explain with example.....	233
Q.334.	Explain the use of document.ready() function in jQuery.....	234
Q.335.	Write short note on jQuery Event Functions.....	235
Q.336.	What is the importance of jQuery in the Development of web Applications?.....	235
Q.337.	List and explain the benefits of using jQuery.....	236
Q.338.	What is jQuery Selector? Write some examples.....	236
Q.339.	Explain DOM Manipulation Methods in jQuery.....	240

| Program |**I | II | III | IV | V | VI****241 – 256**

Q.340.	Write a windows application to open the website mu.ac.in upon click the link. Write the necessary Events and steps required for implementation. [Hint: Use LinkLabel]...	241
Q.341.	Write ASP.NET Code to display selected elements from the CheckBoxList on a label control. Elements on the label must be separated by a whitespace.....	241
Q.342.	Write ASP.NET Code to send data entered in two textboxes from one web page to another web page. Display the data on two separate labels.....	242
Q.343.	Write a code to insert and update data into a SqlServer Database an ASP.NET Web Page.....	242
Q.344.	Write a code to display all the number in array greater than 10 using LINQ.....	243
Q.345.	Write jQuery code to demonstrate the use of hide() and slideUp() functions on <p> element.....	243



- Q.346.** Write jQuery program that changes the background color of a paragraph to red and font color to yellow when mouse enters over it. Also set the background color to white and font color to black when mouse leaves the paragraph..... 244
- Q.347.** Write a code that shows how to write a "LOCAL" cookie to a client's computer. The "LOCAL" cookie, stores:..... 245
- *FirstName*
 - *LastName*
- Q.348.** Create a web page to read student's seat number, name add contact number using text boxes. Use appropriate validation controls to validate the following:
Student roll number should be from 1 to 120, name is compulsory and contact number must be of 10 digits. Write HTML and code behind code..... 245
- Q.349.** Name, Address, ContactNo, City in SQL Server. Write code to display get all record from database into GridView control..... 246
- Q.350.** Create a web page to declare and initialize array of 10 integers and display the numbers greater than 30 using LINQ. Code should execute on button's click event in a web page..... 247
- Q.351.** Write a code to achieve overriding using virtual method. Use comments whenever necessary..... 247
- Q.352.** Write program using Overloaded Constructors..... 248
- Q.353.** Write a program to create a new cookie with the name "Username" and add it to the HttpResponseMessage Object on the click of a button. Set the expiry date of the cookie to One year from Now..... 248
- Q.354.** Write a program using any five Methods / Property of ArrayList Class..... 249
- Q.355.** Create a string array of names. Write a program with LINQ query to display all names from the array that contain the letter "S" and order them in ascending order. Display the query result in a Label..... 249
- Q.356.** Write a program using jQuery that hides a paragraph on Click of a Button..... 250
- Q.357.** Create a delegate with two init parameter and a return type. Create a class with two Delegate methods multiply and divide. Write a program to implement the Delegate. 250
- Q.358.** Write a C# program to do the following:..... 251
- > Initialize an array A with 10 elements.
 - > Initialize an array B with 7 elements.
 - > Divide each element of array A with each element of array B that is a[0] / b[0], a[1] / b[1] etc.
 - > Implement the same to handle Divide by zero error and Index out of bound error.
- Q.359.** In a web form the details of an employee is filled in after he/she is employed in the company. In the design there is a Text Box to fill the First Name, a Text Box to fill the Last Name, a Text Box to fill the Surname, a text box to enter Email Address, a Text Box to enter the Email Address again to confirm the same, a text box to fill the age. What are the Validation Controls that have to be added for each field in the form?..... 252
- Q.360.** Write a program to illustrate Functional Overloading..... 253
- Q.361.** Create a window Form application in C# with a Listbox, a Textbox and three Buttons Add, Delete and clear. Add button will add the text from the Textbox to the Listbox. The delete button will remove the selected text from the Listbox and the clear button will clear the Listbox..... 254
- Q.362.** Create a Console Application in C# to handle an Event using Timer Object..... 254
- Q.363.** Write a program that adds a cookie to the cookie collection of HttpResponseMessage at the click event of a button. Also set the expiry date of the cookie to 2 days from now. 255
- Q.364.** Write a program that Hides a paragraph when Mouse is Hovered Over it..... 255
- Q.365.** Write a code to redirect the page to google.co.in when typed as google.com..... 256
- Q.366.** Write a jQuery Application to create an animation..... 256



UNIT

I

REVIEW OF .NET FRAMEWORKS

INTRODUCTION TO C#

C# REFERENCE TYPES

Topic

NET FRAMEWORK

- ⇒ THREE LAYER ARCHITECTURE OF ASP.NET
 - > PRESENTATION LAYER
 - > MIDDLE LAYER
 - > DATABASE LAYER
- ⇒ .NET FRAMEWORK ARCHITECTURE
- ⇒ FRAMEWORK CLASS LIBRARY (FCL)
- ⇒ COMMON LANGUAGE RUNTIME (CLR)
- ⇒ COMPONENTS OF CLR
 - > COMMON TYPE SYSTEM (CTS)
 - > COMMON LANGUAGE SPECIFICATION (CLS)
- ⇒ ROLES OF COMMON LANGUAGE RUNTIME
- ⇒ MICROSOFT INTERMEDIATE LANGUAGE (MSIL)
- ⇒ JUST-IN-TIME (JIT) COMPILER
 - > TYPES OF JIT COMPIERS
 - *PRE-JIT COMPLIER*
 - *ECONO JIT COMPLIER*
 - *NORMAL JIT COMPLIER*
- ⇒ BASE CLASS LIBRARY (BCL)
 - > NAMESPACES
 - *SYSTEM NAMESPACE*
 - *SYSTEM.LINQ NAMESPACE*
 - *SYSTEM.WEB NAMESPACE*
 - *SYSTEM.WEB.UI NAMESPACE*
 - *SYSTEM.COLLECTIONS NAMESPACE*
 - > ASSEMBLY
 - *PURPOSE OF ASSEMBLY*
 - COMPONENTS OF ASSEMBLY
 - MANIFEST
 - MODULE
 - TYPE
 - *SIGNIFICANCE OF ASSEMBLIES*
 - TYPES OF ASSEMBLY
 - PRIVATE ASSEMBLY
 - SHARED/PUBLIC ASSEMBLY

INTRODUCTION TO C#

- ⇒ COMPILING AND EXECUTING OF C# PROGRAM
- ⇒ DATA TYPES
 - > PRIMITIVE DATA TYPES
 - > ENUMERATOR (ENUM) DATA TYPES
 - > NECESSITY OF ENUMERATOR DATA TYPE
- ⇒ BOXING AND UNBOXING
- ⇒ NAMING RULES AND CONVENTIONS OF VARIABLE
- ⇒ INDEXERS
- ⇒ OPERATORS
 - > COMPARISON OPERATORS
 - > LOGICAL OPERATORS
- ⇒ REGULAR EXPRESSION
 - > REGEX PATTERNS
- ⇒ FLOW CONTROLS
 - > BREAK STATEMENT
 - > CONTINUE STATEMENT
 - > FOR LOOP
 - > FOREACH LOOP
 - > FOR LOOP Vs. FOREACH LOOP
 - > SWITCH STATEMENT
 - > FALLTHROUGH IN SWITCH
 - > GENERAL FORM OF SWITCH STATEMENT
 - > BASIC STRUCTURE OF A SWITCH STATEMENT
- ⇒ METHODS
 - > WRITE ()
 - > WRITELINE ()
 - > MAIN ()
 - > READ ()
 - > READLINE ()
 - > SEALED METHODS
 - > READ () Vs. READLINE ()
 - > RESPONSE.REDIRECT ()
 - > SERVER.TRANSFER ()



> RESPONSEREDIRECT () VS. SERVERTRANSFER ()

⇒ EXCEPTION

> EXCEPTION HANDLING

- TRY
- CATCH
- FINALLY
- THROW

⇒ PROPERTIES/SMART FIELDS

> STATIC PROPERTIES
> INHERITANCE PROPERTIES
> POLYMORPHISM PROPERTIES
> ABSTRACT PROPERTIES

- PRIVATE ASSEMBLY VS. SHARED/PUBLIC ASSEMBLY

⇒ GARBAGE COLLECTOR

> WORKING OF GARBAGE COLLECTION

> SIMILARITIES-INTERFACES & ABSTRACT CLASSES

> NECESSARY OF EXPLICIT INTERFACE

⇒ CONSTRUCTOR

> TYPES OF CONSTRUCTORS IN C#
> DEFAULT CONSTRUCTORS
> PARAMETERIZED CONSTRUCTOR
> COPY CONSTRUCTOR
> STATIC CONSTRUCTOR
> PRIVATE CONSTRUCTOR

OOPS WITH C#

⇒ ARRAY

> JAGGED ARRAY
> VARIABLE SIZED ARRAYS
> ARRAY OF ARRAYS
> ARRAYLIST
> ARRAY VS. ARRAYLIST

⇒ PARAMETERS

> REFERENCES PARAMETERS
> OUTPUT PARAMETERS

⇒ ACCESS MODIFIERS

> TYPES OF ACCESS MODIFIERS

- PUBLIC
- PRIVATE
- PROTECTED
- INTERNAL
- PROTECTED INTERNAL

⇒ CLASS

> DERIVED CLASS
> ABSTRACT CLASS
> STATIC CLASS
> SEALED CLASS
> PAGE CLASS
> PARTIAL CLASS
> PROTECTED DATA MEMBER
> STRUCTURE VS. CLASSES

⇒ OBJECTS

> REQUEST OBJECT
> RESPONSE OBJECT

⇒ INTERFACES

> ABSTRACT CLASS VS. INTERFACE

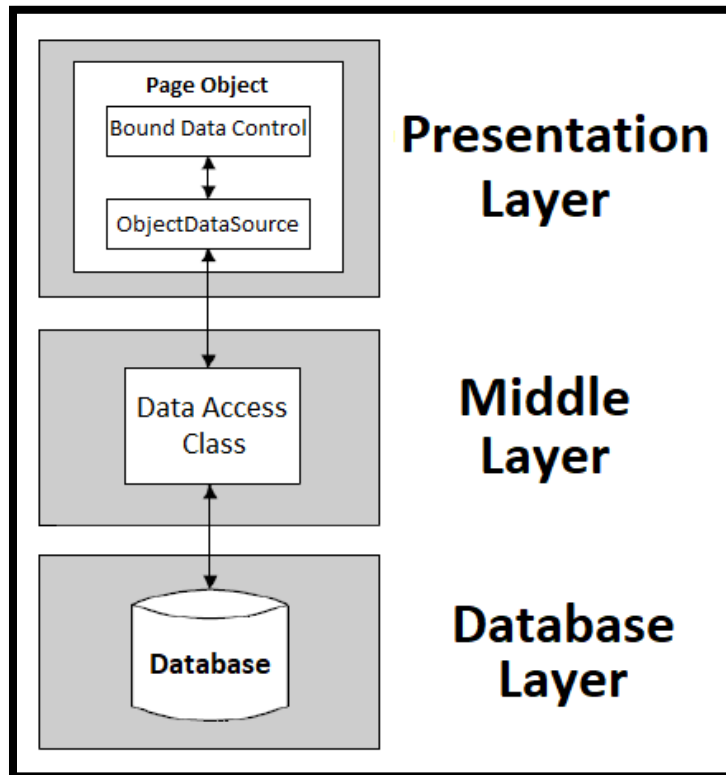
**NET FRAMEWORK****Q.1. Explain the three Layer Architecture of ASP.NET.**

ASKED YEAR →

CBSGS: Nov – 2014

SOLUTION**Three Layer Architecture Of ASP.NET:**

Three tier architecture means dividing our project into three layers that is Presentation Layer (UI Layer), Business Layer (Logic Code Layer) and Data Layer (Layer which connects to database).

**Presentation Layer:**

- ⇒ Presentation Layer is our user interface layer where we can design our interface using HTML Web Controls or Windows Controls or Mobile Controls.
- ⇒ It can be our website or Windows Application or Mobile Application.
- ⇒ This layer only communicates with Business Logic.

Middle Layer:

- ⇒ Application layer OR Business Layer is the middle layer or bridge layer that connects database layer and presentation layer.
- ⇒ In this layer we can write our business logic (logic written as per business requirement) or any validation code.
- ⇒ This layer communicates with database layer and presentation layer.

Database Layer:

- ⇒ Database Layer makes the connection to the database server.
- ⇒ In this layer you can write database connection and SQL queries or stored procedures.
- ⇒ This layer only communicates with business layer.
- ⇒ When user post any data from user interface (presentation page), data first goes to the business layer there data will be validated data is posted to database layer to insert into the database.



- ⇒ When user request any data from the database then the request is first processed by the business layer validates the request and sends to database layer than database layer forwards it to the database it to the database server and fetches necessary records.
- ⇒ Found records are loaded by the database layer and passes it back to the business layer then business layer those records to the presentation layer.

Q.2.

- **Draw and Explain .NET Framework Architecture.** [CBSEGS: Apr – 2014] | [IDOL: Dec/May – 2017 | Oct – 2012]
- **What is .NET Framework? Explain various components of .NET Framework 4.0.** [CBSEGS: Nov – 2017]
- **Write a short notes on .NET Framework.** [IDOL: May – 2018 | Apr – 2014]
- **What is .NET Framework? What is in the .NET Framework?** [CBSEGS: Oct – 2013]

ASKED YEAR →

IDOL: May – 2018 | Dec/May – 2017 | Apr – 2014 | Oct – 2012

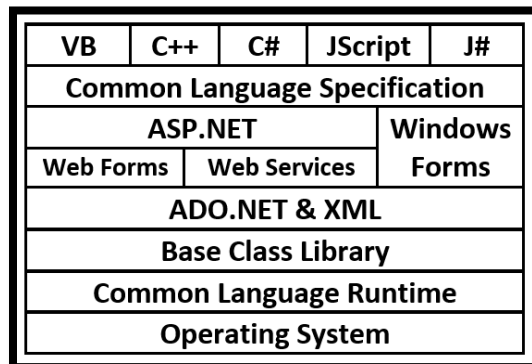
CBSEGS: Nov – 2017 | Apr – 2014 | Oct – 2013

SOLUTION

.NET Framework Architecture:

- ⇒ The architecture of .NET comprises multiple layers starting with an operating system up to the different programming languages.
- ⇒ At the base of the diagram is the operating system, which technically can be any platform but typically Microsoft Windows.
 - Common Language Specification (CLS)
 - Common Language Runtime (CLR)
 - Framework Class Library (FCL)

Components Of .NET Framework Architecture



Common Language Specification (CLS):

- ⇒ The CLS is a common platform that integrates code and components from multiple .NET programming languages.
- ⇒ A .NET application can be written in multiple programming languages with no extra work by the developer.
- ⇒ CLS ensures complete interoperability among applications, regardless of the language used to create the application.
- ⇒ These applications written in different programming languages get compiled to common intermediate language and can work together in the same application.

Framework Class Library (FCL):

- ⇒ The .NET Framework Class Library (FCL) provides the core functionality of .NET Framework Architecture.
- ⇒ The .NET Framework Class Library (FCL) includes a huge collection of reusable classes, interfaces, and value types that expedite & optimize the development process & provide access to system functionality.
- ⇒ The .NET Framework Class Library (FCL) organized in a hierarchical tree structure is divided into Namespaces. Namespace is a logical grouping of types for the purpose of identification.
- ⇒ The .NET Framework Class Library (FCL) provides the consistent base type that are used across all .NET enabled languages. The classes are accessed by namespaces, which reside within Assemblies.
- ⇒ The system Namespace is the root for type in the .NET Framework. The .NET Framework Class Library (FCL) are managed classes that provide access to system services.
- ⇒ The .NET Framework Class Library (FCL) classes are object oriented and easy to use in program developments. Moreover, third-party components can integrate with the classes in .NET Framework.

Common Language Runtime (CLR) :

- ⇒ .Net Framework provides runtime environment called Common Language Runtime (CLR).
- ⇒ It provides an environment to run all the .Net Programs.
- ⇒ The code which runs under the CLR is called as Managed Code.
- ⇒ Programmers need not to worry on managing the memory if the programs are running under the CLR as it provides memory management and thread management.
- ⇒ Programmatically, when our program needs memory, CLR allocates the memory for scope and de-allocates the memory if the scope is completed.
- ⇒ Language Compilers (e.g. C#, VB.Net, J#) will convert the Code/Program to Microsoft Intermediate Language (MSIL) intern this will be converted to Native Code by CLR.

.NET Languages :

- ⇒ .NET includes new object oriented programming languages such as C#, VB.NET, J# and managed C++.
- ⇒ These languages all compile to MSIL and work together in a single application.

.NET Tools - Visual Studio.net :

- ⇒ Microsoft's flagship tool for developing Windows Software.
- ⇒ Studio provides an Integrated Developing Environment (IDE) for developers to create Standalone Windows Applications, Interactive Web Sites, Web Applications, and Web Services running on any platform that supports .NET.

COMMON LANGUAGE RUNTIME (CLR)**Q.3.**

- **Write short note on Common Language Runtime (CLR) in .NET. [CBSSGS: Nov – 2017]**
- **Describe the roles of CLR in .NET Framework. [CBSSGS: Nov – 2017 | Apr – 2016]**

ASKED YEAR ⇒

CBSSGS: Nov – 2017 | Apr – 2016

SOLUTIONCommon Language Runtime (CLR):

- ⇒ The Common Language Runtime (CLR) is the common Execution (Runtime) Environment.
- ⇒ It works as a layer between Operating Systems and the applications written in .NET compliant languages.
- ⇒ The main function of Common Language Runtime (CLR) is to convert the Managed Code into Native Code and then execute the program.
- ⇒ Common Language Runtime (CLR) is the programming (Virtual Machine component) that manages the execution of programs written in any language that uses the .NET Framework, for example C#, VB.Net, F# and so on.
- ⇒ It provides an environment to run all the .Net Programs.
- ⇒ The code which runs under the CLR is called as Managed Code.
- ⇒ Common Language Runtime (CLR) provides many services like:
 - Code Management
 - Memory Management
 - Verification of Type Safety
 - Conversion of MSIL to Native Code
 - Loading and Execution of managed Code
 - Handling Cross-Language Exceptions

Components Of Common Language Runtime (CLR):Common Type System (CTS) :

- ⇒ Common Type System (CTS) describes a set of types that can be used in different .Net languages in common. That is, the Common Type System (CTS) ensure that objects written in different .Net languages can interact with each other. For Communicating between programs written in any .NET complaint language, the types have to be compatible on the basic level.
- ⇒ These types can be Value Types or Reference Types. The Value Types are passed by values and stored in the stack. The Reference Types are passed by references and stored in the heap.

Common Language Specification (CLS) :

- ⇒ It is a sub set of CTS and it specifies a set of rules that needs to be adhered or satisfied by all language compilers targeting CLR.
- ⇒ It helps in cross language inheritance and cross language debugging.

Microsoft Intermediate Language (MSIL) :

- ⇒ It is language independent code. When you compile code that uses the .NET Framework library, you don't immediately create operating system - specific native code.
- ⇒ Instead, you compile your code into Microsoft Intermediate Language (MSIL) code.
- ⇒ The MSIL code is not specific to any operating system or to any language.

Roles Of CLR:

- ⇒ CLR Stands for Common Language Runtime, is the environment where all programs using .net technologies over executed.
- ⇒ CLR allows the execution of code across different platform using transmitting code IL (Intermediate Language).
- ⇒ Intermediate Language (IL) is converted into Machine Language during execution by JIT (Just In Time) Compiler.
- ⇒ During JIT Compilation the code is also check or type safety.
- ⇒ Type safety ensure that object are always accessed in compatible way. If value of 8 byte given to 4 byte then error will occur by CLR and error will attempt to trap.
- ⇒ CLR consist of common rules that must be followed by all language using .net framework these set of rules are called as Common Language Specification (CLS).
- ⇒ The CLS enables an object or an application to interact with objects or application of their languages.
- ⇒ One of the specification defined in CLS is Common Type System (CTS).
- ⇒ CTS provides a type system that is common across all languages.
- ⇒ CTS defines rules that ensure that the data object written in various languages are able to interact with each other.

MICROSOFT INTERMEDIATE LANGUAGE (MSIL)**Q.4.**

- **What is Microsoft Intermediate Language (MSIL)? Why is important? [CBSGS: Apr – 2016]**
- **Write a short note on CIL Compiler. [IDOL: May – 2017]**

ASKED YEAR ⇒

IDOL: May – 2017

CBSGS: Apr – 2016

SOLUTIONMicrosoft Intermediate Language (MSIL):

- ⇒ MSIL stands for Microsoft Intermediate Language. We can call it as Intermediate Language (IL) or Common Intermediate Language (CIL).
- ⇒ During the compile time, the compiler convert the source code into Microsoft Intermediate Language.
- ⇒ MSIL is a CPU independent set of instructions that can be efficiently converted to the native code.
- ⇒ During the runtime the Common Language Runtime (CLR)'s Just In Time Compiler (JIT) converts the MSIL code into native code to the operating system.
- ⇒ When a compiler produces MSIL it also produces Metadata. The MSIL and Metadata are contained in a Portable Execution (PE) file.

Important of MSIL:

MSIL includes instructions for loading, storing, initializing, and calling method on objects, as well as instructions for arithmetic and logical operations, control flow, direct memory access, exception handling and other operations.

**JUST-IN-TIME (JIT) COMPILER****Q.5. Explain the terms JIT Compiler.**ASKED YEAR ⇒ IDOL: May – 2017 | Oct – 2016 | Apr – 2015CBSGS: Apr – 2017**SOLUTION****JIT Compiler:**

- ⇒ The JIT Compiler is part of the Common Language Runtime (CLR).
- ⇒ The CLR manages the execution of all .NET applications.
- ⇒ In addition to JIT compilation at runtime, the CLR is also responsible for garbage collection, type safety and for exception handling.

Types Of JIT Compilers:

- 1) Pre-JIT Compiler
- 2) Econo JIT Compiler
- 3) Normal JIT Compiler

Pre-JIT Compiler:

- ⇒ Pre-JIT compiles complete source code into native code in a single compilation cycle.
- ⇒ This is done at the time of deployment of the application.

Econo-JIT Compiler:

- ⇒ Econo-JIT compiles only those methods that are called at runtime.
- ⇒ However, these compiled methods are removed when they are not required.

Normal-JIT Compiler:

- ⇒ Normal-JIT compiles only those methods that are called at runtime.
- ⇒ These methods are compiled the first time they are called, and then they are stored in cache.
- ⇒ When the same methods are called again, the compiled code from cache is used for execution.
- ⇒ These methods are compiled the first time they are called, and then they are stored in cache.
- ⇒ When the same methods are called again, the compiled code from cache is used for execution.

Q.6. Explain the different parts that constitute ASP.NET application.

ASKED YEAR ⇒

CBSGS: Apr – 2015**SOLUTION****Different Parts That Constitute ASP.NET Application:**

Following are the different parts that constitute ASP.NET application:

- ⇒ **Web Form Site (.aspx)**: Web form represent the pages that your users view in their browser.
- ⇒ **Master Page (.master)**: It enable us to define the global structure and the look and feel of a web site.
- ⇒ **Web User Control (.ascx)**: It contains page fragments that can be reused in multiple pages in your site.
- ⇒ **HTML Page (.html/.htm)**: It can be used to display static HTML in our web site.
- ⇒ **Style Sheet (.css)**: It contains global configuration information that is used throughout the site.
- ⇒ **Site Map (.sitemap)**: It contains a hierarchical representation of files in our site in an XML format.
- ⇒ **Jscript File (.js)**: It contains JavaScript that can be executed in the client's browser.
- ⇒ **Skin File (.skin)**: It contains design information for controls in your web site.

**BASE CLASS LIBRARY (BCL)****Q.7. Explain Framework Base Class Library.**ASKED YEAR → **IDOL:** May – 2016**CBSGS:** Apr – 2016 | Apr – 2015**SOLUTION****Base Class Library (BCL):**

- ⇒ The .NET framework provides a set of base class libraries which provide functions and features which can be used with any programming language which implements .NET, such as Visual Basic, C# (or course), Visual C++, etc.
- ⇒ It provides the functionality like ADO.NET, XML, Threading, IO, Security, Diagnostics, Resources, Globalization, collections etc.
- ⇒ The .NET base class library exists in order to encapsulate a huge number of common functions and makes them easily accessible to the developer.
- ⇒ It is a library of classes, interfaces, and value types.
- ⇒ This library system functionality and is the foundation of .NET Framework applications, components, and controls are built.
- ⇒ It serves as the main point of interaction between developer and runtime.

Base Class Library Namespace:

- **System:** It contains fundamental classes and base classes that define commonly-used value and reference data types, events and event handlers, interfaces, attributes and processing exceptions.
- **System.Activities:** It contains all the classes necessary to create and work with activities in Windows Workflow Foundation.
- **System.Collections:** It contains types that define various standard, specialized and generic collection objects.
- **System.Configuration:** It contains types that define various standard, specialized and generic collection objects.
- **System.Data:** It contains classes for accessing and managing data from diverse sources.
- **System.Deployment:** It contains types that define support deployment of ClickOnce application.
- **System.EnterpriseServices:** It contains types that define the com + services architecture, which provides an infrastructure for enterprise application.
- **System.Globalization:** It contains classes that define culture-related information, including language, country/region, calendars in use, currency.
- **System.IO:** It contains types that support input and output, including the ability to read and write data to streams either synchronously or asynchronously.
- **System.Linq:** It contains types that support queries which is used Language-Integrated Query (LINQ). It includes types that represent queries as objects in expression trees.
- **System.Management:** It contains a type that provides access to management information and management events about the system, devices and applications instrumented to the Windows Management Instrumentation (WMI) infrastructure.
- **System.Net:** It contains classes that provide a simple programming interface for a number of network protocols.
- **System.Printing:** It contains types that support printing. It provides access to the properties of print system objects.
- **System.Runtime:** It contains types that support an application's interaction with the common language runtime.
- **System.Security:** It contains classes that represent the .Net framework security system and permissions.
- **System.Windows:** It contains types used in Windows Presentation Foundation (WPF) application, including Animation Clients, User Interface Controls, and Data Binding.

**NAMESPACE****Q.8.**

- **What is Namespace?** [IDOL: Dec – 2017 | May – 2017 | May – 2016] | [CBSGS: Apr/Nov – 2016 | Oct – 2012]
- **Explain System Namespace.** [IDOL: Dec – 2017 | May – 2016] | [CBSGS: Apr – 2016 | Oct – 2012]
- **How to create namespace and its alias?** [CBSGS: Nov – 2016]
- **Explain the concept of Namespaces with suitable example.** [IDOL: Oct – 2016]
- **List and explain the use of any five Namespaces in C#.** [IDOL: May – 2017] | [CBSGS: Oct – 2013]
- **Write a short note on the System.Collections Namespace.** [IDOL: May – 2016]

ASKED YEAR →

[IDOL: May/Dec – 2017 | Oct/May – 2016 | Oct – 2012]

[CBSGS: Nov/Apr – 2016 | Oct – 2013]

SOLUTION**Namespaces:**

- ⇒ Namespaces are C# program elements which help to organize programs.
- ⇒ In Microsoft .NET the namespace is like container for objects, structure, delegates, etc.
- ⇒ In .net every program is created with default namespace called as global namespace however a program can declare any number of namespaces.
- ⇒ Namespaces are used to create logical groups of related classes and interfaces that can be used by any language targeting the .NET Framework.
- ⇒ Namespaces are used to avoid conflicts between classes that have the same names. For example, one can use two classes of the same name in an application provided they belong to different namespaces.
- ⇒ One can access the classes belonging to a namespace by simply importing the namespace into an application. '.' (dot) is used as a delimiter between classes and namespaces. For example `System.Console` represents the console class of the System namespace.
- ⇒ There are more than 80 in-built namespaces in .NET Framework that are use frequently.
- ⇒ Popular in-built Namespaces that are used frequently are `System.IO`, `System.Drawing`, `System.Windows.Forms` etc.

Defining a Namespace:

A namespace definition begins with the keyword namespace followed by the namespace name as follows:

Syntax:

```
namespace namespace_name
{
// code declarations
}
Syntax for alias:
using alias_name=Namespace_name;
```

Example:

```
using System;
using Cat = System.Text.StringBuilder; // Cat is alia
class Program
{
static void Main()
{
Cat cat = new Cat();
cat.Append("sparky");
cat.Append(100);
Console.WriteLine(cat);
}
}
```

Output:



sparky100

System Namespace:

- ⇒ The namespace can be different in different project but each of them should be placed under "system" namespace.
- ⇒ System namespace contains fundamental classes and base classes that define commonly used value and reference data types, events and Event Handlers, Interfaces, Attributes and Processing Exceptions.
- ⇒ Others classes provide services supporting data type conversion, method parameter manipulation, mathematics, remote and local program invocation, application environment management and supervision of managed and unmanaged applications.
- ⇒ System namespace is specified with using system directive.

Example Classes:

```
Console  
Convert
```

System.LINQ Namespace:

It provides classes and interfaces that support queries that use Language Integrated Query (LINQ).

Example Classes:

```
Enumerable  
Queryable
```

System.Web Namespace:

It contains extension class for HTTP context.

Example Classes:

```
HttpContextBaseExtensions  
HttpContextExtensions
```

System.Web.UI Namespace:

It provides classes and interfaces that enable one to create ASP.NET Server controls and ASP.NET Web pages for the user interface of one's ASP.NET Web Applications.

Example Classes:

```
Control  
MasterPage
```

System.Collections Namespace:

It contains interfaces and classes that define various collections of object such as lists, queues, bit arrays, hash tables and dictionaries.

Example Classes:

```
ArrayList  
SortedList
```


**ASSEMBLY****Q.9.**

- **What is Assembly? Explain the purpose of Assemblies in .NET Framework. [CBSGS: Oct – 2012]**
➤ **Write a short note on Assembly. [IDOL: May – 2017 | Oct – 2016] | [CBSGS: Nov – 2014]**

ASKED YEAR →

IDOL: May – 2017 | Oct – 2016

CBSGS: Oct – 2012 | Nov – 2014

SOLUTION**Assembly:**

- ⇒ Assemblies are the building blocks of .NET Framework applications; they form the fundamental unit of deployment.
- ⇒ Assembly is a compiled code library for deployment, versioning and security purpose.
- ⇒ An assembly is a single deployable unit that contains all the information about the classes, structures and interfaces.
- ⇒ When an application is compiled, The MSIL Code created is stored in an assembly.
- ⇒ With the MSIL Code, metadata (information about the information contained in the assembly) is also created after the first compilation.
- ⇒ An assembly stores all the information about itself. This information is called metadata and includes the name and version number of the assembly and the security information.
- ⇒ That means an assembly contains the MSIL Code, which the common language runtime executes (JIT), and the type metadata.
- ⇒ The metadata enables the assemblies to be fully descriptive.
- ⇒ Assemblies and metadata provides the CLR with the information required for executing an application. For Example, if an application uses a component, the assembly keeps track of the version number of the component used in the application. The assembly provides this information to the CLR while the application is being executed.
- ⇒ There are two types of assemblies, namely Private Assemblies and Shared Assemblies.
- ⇒ To make a shared assembly, we need to register it with GAC (Global Assembly Cache) whereas private assemblies reside in applications directory.

Purpose of Assembly in .NET Framework:

- ⇒ Assemblies in .NET are a solution to the dll hell problem as one can use different versions of same assembly in different applications at the same time.
- ⇒ Assemblies are designed to simplify application deployment and to solve versioning problem that can occur with component-based applications.
- ⇒ To solve versioning problems, as well as the remaining problems that lead to DLL conflicts, the runtime uses assemblies to do the following:
 - Enable developers to specify version rules between different software components.
 - Provide the infrastructure to enforce versioning rules.
 - Provide the infrastructure to enforce versioning rules.
 - Provide the infrastructure to allow multiple versions of a component to be run simultaneously (called side-by-side execution).
 - Simplify uninstalling and replicating applications.

**Q.10. Explain the components of Assembly.**

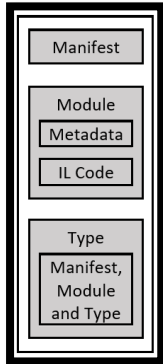
ASKED YEAR ⇒

IDOL: May – 2017 | Apr – 2015

CBSGS: Nov – 2016 | Apr – 2014

SOLUTION**Components Of Assembly:**

Assembly consists of Manifest, Module and Type.

**Manifest:**

- ⇒ It describes the assembly.
- ⇒ It contains:
 - Name and Version number of the assembly.
 - Its interaction information with other assemblies.
 - Security permissions required by the assembly.

Module:

- ⇒ Module is either a DLL or EXE File.
- ⇒ It contains:
 - Compiled Code or Intermediate Language Code.
 - Metadata is associated with Module.

Type:

- ⇒ Type is a class that contains data and logic affecting the data.
- ⇒ This class show the information using properties, fields and methods.

Q.11. What is the significance of Assemblies in .NET?

ASKED YEAR ⇒

CBSGS: Oct – 2013

SOLUTION**Significance Of Assemblies:**

- ⇒ Assemblies are main building blocks. An assembly maybe defined as a unit of deployment. A single assembly is a collection of types, and resources. The CLR does not understand any types that are outside assemblies. The CLR executes the code in assemblies as they contain MSIL Code. They define type, version and security boundaries.
- ⇒ Assemblies in .NET are a solution to the DLL hell problem as one can use different versions of same assembly in different applications at the same time. To make a Shared Assembly, we need to register it with GAC where as private assemblies reside in applications directory.
- ⇒ Assemblies are designed to simplify application deployment and to solve versioning problems that can occur with component-based applications.
- ⇒ To solve versioning problems, as well as the remaining problems that lead to DLL conflicts, the runtime uses assemblies to do the following:
 - Enable developers to specify version rules between different software components.
 - It provide the infrastructure to enforce versioning rules.
 - It provide the infrastructure to allow multiple versions of a component to be run simultaneously (called side-by-side execution).
 - Simplify uninstalling and replicating applications.

TYPES OF ASSEMBLY**Q.12.** Explain the concept of Private and Shared Assembly.

ASKED YEAR → IDOL: May – 2018 | Dec – 2017

SOLUTIONPrivate Assembly:

- ⇒ A private assembly is an assembly that is deployed with an application and is available for the exclusive use of that application.
- ⇒ That is, other applications do not share the private assembly.
- ⇒ Private assemblies are one of the methods that can be used to create isolated applications.
- ⇒ Private assemblies must be designed to work side-by-side with other versions of the assembly on the system.
- ⇒ Private assemblies must be accompanied by an assembly manifest.
- ⇒ Private assemblies are installed in a folder of the application's directory structure.
- ⇒ A private assembly is not required to be signed, and `publicKeyToken` is not required in the `assemblyIdentity` element of the assembly manifest.
- ⇒ Private assemblies can be installed into the application's folder using any installation technology. Private assemblies are not required to be installed using the Windows Installer.

Shared/Public Assembly:

- ⇒ A shared assembly is an assembly available for use by multiple applications on the computer.
- ⇒ Shared side-by-side assemblies are not registered globally on the system, but they are globally available to applications that specify a dependence on the assembly in manifests.
- ⇒ Multiple versions of side-by-side assemblies can be shared by different applications running at the same time.
- ⇒ Shared side-by-side assemblies are installed in the `WinSxS` folder.
- ⇒ Shared side-by-side assemblies can be installed by an operating system update, or by a Windows Installer package that installs or updates an application.

PRIVATE ASSEMBLY VS. SHARED/PUBLIC ASSEMBLY**Q.13.** Differentiate between Private Assembly and Shared Assembly.

ASKED YEAR →

CBSGS: Nov/Apr – 2017

SOLUTIONPrivate Assembly Vs. Shared/Public Assembly:

<u>Shared/Public Assembly</u>	<u>Private Assembly</u>
Public assembly can be used by multiple applications.	Private assembly can be used by only one application.
Public assembly is stored in GAC (Global Assembly Cache).	Private assembly will be stored in the specific application's directory or sub-directory.
Public assembly is also termed as shared assembly.	There is no other name for private assembly.
Strong name has to be created for Public Assembly.	Strong name is not required for Private Assembly.
Public assembly should strictly enforce version constraint.	Private assembly doesn't have any version constraint.

**GARBAGE COLLECTOR****Q.14.**

- **How Garbage Collection works?** [CBSGS: Nov – 2017 | Apr/Nov – 2016 | Apr – 2015 | Apr – 2014] | [IDOL: May – 2016]
- **What is Garbage Collection?** [CBSGS: Apr – 2017 | Apr/Nov – 2016 | Apr – 2015] | [IDOL: May – 2017 | May – 2016]
- **Explain three types of generations in Garbage Collection.** [CBSGS: Apr – 2017]

ASKED YEAR ⇒

IDOL: May – 2016

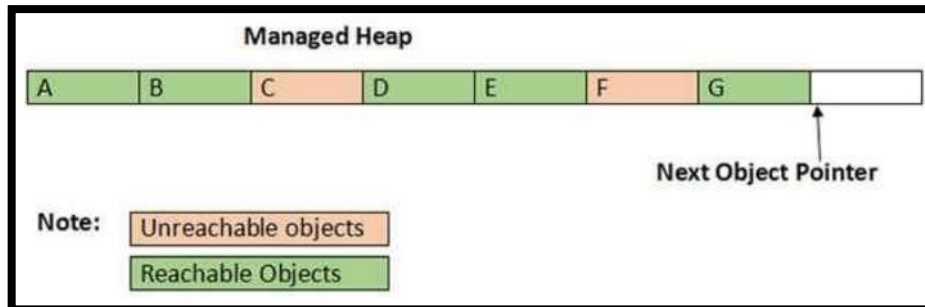
CBSGS: Nov/Apr – 2017 | Nov/Apr – 2016 | Apr – 2015 | Apr – 2014

SOLUTION**Garbage Collector:**

- ⇒ The Garbage Collector (GC) is the part of the .NET Framework that allocates and releases memory for your .NET applications.
- ⇒ The Common Language Runtime (CLR) manages allocation and deallocation of a managed object in memory.
- ⇒ C# programmers never do this directly; there is no delete keyword in the C# language. It relies on the garbage collector.

Example:

Assume the managed heap contains a set of objects named A, B, C, D, E, F and G. During garbage collection, these objects are examined for active roots. After the graph has been constructed, unreachable objects (that we will assume are objects C and F) are marked as garbage in reddish color in the following diagram.

**Working Of Garbage Collection:**

- ⇒ Garbage Collection works on managed heap, which is nothing but a block of memory to store objects, when garbage collection process is put in motion, it checks for dead objects and the objects which are no longer used, then it compacts the space of live object and tries to free more memory.
- ⇒ Basically, heap is managed by different 'Generations', it stores and handles long-lived and short-lived objects. There are three generations of objects on the heap; see the below generations of Heap:

Generation 0:

- ⇒ This generation holds short-lived objects, e.g., Temporary objects. GC initiates garbage collection process frequently in this generation.
- ⇒ Newly allocated objects form a new generation of objects and are implicitly generation 0 collections, unless they are large objects, in which case they go on the large object heap in a generation 2 collection.
- ⇒ Most objects are reclaimed for garbage collection in generation 0 and do not survive to the next generation.

Generation 1:

- ⇒ This generation contains short-lived objects and serves as a buffer between short-lived objects and long-lived objects.

Generation 2:

- ⇒ This generation contains long-lived objects. An example of a long-lived object is an object in a server application that contains static data that is live for the duration of the process.
- ⇒ Objects which are not collected in generation Zero, are then moved to generation 1, such objects are known as survivors, similarly objects which are not collected in generation One, are then moved to generation 2 and from there onwards objects remain in the same generation.

**COMPILING AND EXECUTING OF C# PROGRAM****Q.15. Explain Compiling and Execution of a C# Program.**ASKED YEAR → **IDOL:** May – 2016**SOLUTION****Compiling And Executing Of C# Program:**

If we are using Visual Studio.Net for compiling and executing C# programs, take the following steps –

- STEP 1:** Start Visual Studio.
- STEP 2:** On the menu bar, choose File → New → Project.
- STEP 3:** Choose Visual C# from templates, and then choose Windows.
- STEP 4:** Choose Console Application.
- STEP 5:** Specify a name for your project and click OK button.
- STEP 6:** This creates a new project in Solution Explorer.
- STEP 7:** Write code in the Code Editor.
- STEP 8:** Click the Run button or press F5 key to execute the project. A Command Prompt window appears that contains the line Hello World.

Compile a C# program by using the command-line:

- STEP 1:** Open a text editor and add the above-mentioned code.
- STEP 2:** Save the file as "helloworld.cs".
- STEP 3:** Open the command prompt tool and go to the directory where you saved the file.
- STEP 4:** Type "helloworld.cs" and press enter to compile your code.
- STEP 5:** If there are no errors in your code, the command prompt takes you to the next line and generates "helloworld.exe" executable file.
- STEP 6:** Type "helloworld" to execute your program.
- STEP 7:** You can see the output Hello World printed on the screen.

PRIMITIVE DATA TYPES**Q.16. List and Explain the various Primitive Data Types used in C#.**ASKED YEAR → **IDOL:** Apr – 2015 | Apr – 2013**SOLUTION****Primitive Data Types:****Boolean (bool):**

Boolean data type has two possible values (i.e. true or false). Default value is false. Boolean variables are generally used to check conditions such as in if statements, loops, etc.

Signed Integral:

These data types hold integer values (both positive and negative). Out of the total available bits, one bit is used for sign.

- 1) **sbyte:**
 - Size: 8 bits
 - Range: -128 to 127.
 - Default Value: 0
- 2) **short:**
 - Size: 16 bits
 - Range: -32,768 to 32,767
 - Default value: 0
- 3) **int:**
 - Size: 32 bits
 - Range: -231 to 231-1



- **Default value:** 0
- 4) **long:**
- **Size:** 64 bits
 - **Range:** -263 to 263-1
 - **Default value:** 0L [L at the end represent the value is of long type]

Unsigned Integral:

These data types only hold values equal to or greater than 0. We generally use these data types to store values when we are sure, we won't have negative values.

- 1) **byte:**
- **Size:** 8 bits
 - **Range:** 0 to 255.
 - **Default value:** 0
- 2) **ushort:**
- **Size:** 16 bits
 - **Range:** 0 to 65,535
 - **Default value:** 0
- 3) **uint:**
- **Size:** 32 bits
 - **Range:** 0 to 232-1
 - **Default value:** 0
- 4) **ulong:**
- **Size:** 64 bits
 - **Range:** 0 to 264-1
 - **Default value:** 0

Floating Point:

These data types hold floating point values i.e. numbers containing decimal values. For example, 12.36, -92.17, etc.

- 1) **float:** Single-precision floating point type
- **Size:** 32 bits
 - **Range:** 1.5×10^{-45} to 3.4×10^{38}
 - **Default value:** 0.0F [F at the end represent the value is of float type]
- 2) **Double:** Double-precision floating point type. What is the difference between single and double precision floating point?
- **Size:** 64 bits
 - **Range:** 5.0×10^{-324} to 1.7×10^{308}
 - **Default value:** 0.0D [D at the end represent the value is of double type]

Character (char):

It represents a 16 bit unicode character.

- **Size:** 16 bits
- **Default value:** '\0'
- **Range:** U+0000 ('\u0000') to U+FFFF ('\uffff')

Decimal:

Decimal type has more precision and a smaller range as compared to floating point types (double and float). So it is appropriate for monetary calculations.

- **Size:** 128 bits
- **Default value:** 0.0M [M at the end represent the value is of decimal type]
- **Range:** $(-7.9 \times 10^{28}$ to $7.9 \times 10^{28}) / (100$ to $28)$



ENUMERATION (ENUMS)

Q.17. What are enumerations? Explain it with the help of an example.

ASKED YEAR → IDOL: Apr – 2013

SOLUTION

Enumerations:

- ⇒ An enumeration type (also named an enumeration or an enum) provides an efficient way to define a set of named integral constants that may be assigned to a variable.
- ⇒ An enum type is a distinct type that declares a set of named constants.
- ⇒ They are unique types that allow to declare symbolic names to integral values.
- ⇒ Enums are value types.

```
enum<typeName>
{
    <value1>,
    <value2>,
    <value3>,
    ...
    <valueN>,
}
```

- ⇒ Assume that you have to define a variable whose value will represent a day of the week. There are only seven meaningful values which that variable will ever store. To define those values, you can use an enumeration type, which is declared by using the enum keyword.

EXAMPLE:

```
enum Days
{
    Sunday,
    Monday,
    Tuesday,
    Wednesday,
    Thursday,
    Friday,
    Saturday,
}
enum Months: byte
{Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec};
```

- ⇒ By default the underlying type of each element in the enum is int. We can specify another integral numeric type by using a colon, as shown in the previous example.
- ⇒ We can verify the underlying numeric values by casting to the underlying type, as the following example shows.

```
Days today=Days.Monday;
int dayNumber=(int)today;
Console.WriteLine("{0} is day number #{1}.", today, dayNumber);
Months thisMonth=Months.Dec;
byte monthNumber=(byte)thisMonth;
Console.WriteLine("{0} is month number #{1}.", thisMonth, monthNumber);
```

OUTPUT:

```
Monday is day number #1.
Dec is month number #11.
```

**Q.18. Why is there a necessity of Enumerator Data Type in C#?**

ASKED YEAR → IDOL: Apr – 2014

SOLUTION**Necessity Of Enumerator Data Type:**

- ⇒ An enum is a value type with a set of related named constants often referred to as an enumerator list.
- ⇒ The enum keyword is used to declare an enumeration. It is a primitive data type, which is user defined.
- ⇒ enum type can be integer (float, int, byte, double etc.). But if you used beside int it has to be cast.
- ⇒ enum is used to create numeric constants in .NET framework.
- ⇒ There must be a numeric value for each enum type.
- ⇒ The default underlying type of the enumeration elements is int. By default, the first enumerator has the value 0, and the value of each successive enumerator is increased by 1.

EXAMPLE:

```
enum Dow {Sat, Sun, Mon, Tue, Wed, Thu, Fri};
```

BOXING AND UNBOXING**Q.19.**

- Explain the concept of Boxing and Unboxing. [CBSGS: Apr – 2016 | Apr – 2015] | [IDOL: May – 2018]
- Is it a must to Unbox a Boxed Variable? Explain. [IDOL: Apr – 2014]
- Explain Boxing and Unboxing with Reference to Value Type and References Type. [CBSGS: Nov – 2014]
- What are Boxing and Unboxing? How is it achieved? [CBSGS: Apr – 2017]

ASKED YEAR → IDOL: May – 2018 | Apr – 2014

CBSGS: Apr – 2017 | Apr – 2016 | Apr – 2015 | Nov – 2014

SOLUTION**Concept of Boxing and Unboxing:**

C# allows us to convert a value type to a reference type, and back again to value types. The operation of converting value type to a reference type is called boxing and the reverse operation is called unboxing.

Boxing:

```
int val=1;
object obj=val //creates box to hold val
```

- ⇒ The first line we created a value type val and assigned a value to val.
- ⇒ The second line, we created an instance of object obj and assign value of val to obj.
- ⇒ When executed this code creates a temporary reference type "box" for the object on heap. These types of operation is called Boxing.
- ⇒ Now both the variable val and obj exist but the value of obj resides on the heap. This means that values of are independent of each other (i.e. when a code changes the value of val, the value of obj is not affected).

Unboxing:

```
int val=1;
object obj=val; //Boxing
int i=(int) obj; //Unboxing
```

- ⇒ The first two line shows how to Box a value type.
- ⇒ The next line shows extracts the value of the type from the object. This is converting a value of a reference type into a value of value type. This operation is called unboxing.



INDEXERS

Q.20. What are Indexers? Explain with example.

ASKED YEAR ⇒ IDOL: Dec – 2017 | Oct – 2012

SOLUTION

Indexer:

- ⇒ An indexer is a member that enables an object to be indexed in the same way as an array.
- ⇒ Indexers allow instances of a class or struct to be indexed just like arrays.
- ⇒ Indexers resemble properties except that their accessors take parameters.
- ⇒ The main use of indexers is to support the creation of specialized arrays that are subject to one or more constraints.
- ⇒ However, you can use an indexer for any purpose for which an array-like syntax is beneficial. Indexers can have one or more dimensions.

Features of Indexer:

- ⇒ Indexers enable objects to be indexed in a similar manner to arrays.
- ⇒ A get accessor returns a value. A set accessor assigns a value.
- ⇒ This keyword is used to define the indexers.
- ⇒ The value keyword is used to define the value being assigned by the set indexer.
- ⇒ Indexers do not have to be indexed by an integer value; it is up to you how to define the specific look-up mechanism.
- ⇒ Indexers can be overloaded.
- ⇒ Indexers can have more than one formal parameter, for example, when accessing a two-dimensional array.
- ⇒ A simple One-Dimensional Indexer has this general form:

```
element-type this[int index]
{
    // The get accessor
    get
    {
        // return the value specified by index
    }
    // The set accessor
    set
    {
        // set the value specified by index
    }
}
```

EXAMPLE:

```
using System;
namespace IndexerExample
{
    class MyPreviousExp
    {
        private string[] myCompanies = new string[10];
        //index creation
        public string this[int index]
        {
            get
            {
                if(index<0 || index>=6)
                    return "null";
                else
                    return myCompanies[index];
            }
            set
            {

```




```
if(!(index<0 || index>=10))
myCompanies[index] = value;
}
}
}
class myMainClass
{
public static void Main()
{
MyPreviousExp indexerObj = new MyPreviousExp();
indexerObj[0] = "AMS";
indexerObj[3] = "HCL";
indexerObj[5] = "ACC" ;
for(int i=0; i<10; i++)
{
Console.WriteLine(" My Companies{0} : {1} ",i,indexerObj[i]);
}
}
}
}
```

Output:

```
myCompanies 0 : AMS
myCompanies 1 :
myCompanies 2 :
myCompanies 3 : HCL
myCompanies 4 :
myCompanies 5 : ACC
myCompanies 6 : null
myCompanies 7 : null
myCompanies 8 : null
myCompanies 9 : null
```

NAMING RULES AND CONVENTIONS OF VARIABLE

Q.21. Write the Naming Rules and Conventions of the Variable.

ASKED YEAR ⇒ IDOL: May – 2017

SOLUTION

Naming Rules And Conventions:

Word Choice:

- ✓ DO choose easily readable identifier names.
- ✓ DO favour readability over brevity.
- X DO NOT use underscores, hyphens, or any other nonalphanumeric characters.
- X DO NOT use Hungarian notation.
- X AVOID using identifiers that conflict with keywords of widely used programming languages.

Using Abbreviations and Acronyms:

- X DO NOT use abbreviations or contractions as part of identifier names.
- X DO NOT use any acronyms that are not widely accepted, and even if they are, only when necessary.

Avoiding Language-Specific Names:

- ✓ DO use semantically interesting names rather than language-specific keywords for type names.
- ✓ DO use a generic CLR type name, rather than a language-specific name, in the rare cases when an identifier has no semantic meaning beyond its type.



✓ DO use a common name, such as value or item, rather than repeating the type name, in the rare cases when an identifier has no semantic meaning and the type of the parameter is not important.

OPERATORS

Q.22. Explain the Comparison and Logical Operators in C#.

ASKED YEAR ⇒ IDOL: May – 2017

CBSGS: Nov – 2014

SOLUTION

Comparison Operators:

- ⇒ These operators compare two expressions to determine whether or not they are equal, and if not, how they differ.
- ⇒ Comparison operators are used when you want to compare two values to make a decision.
- ⇒ Comparison operators are most commonly used in conjunction with "If...Then" and "While something is true do this..." statements, otherwise known as conditional statements.
- ⇒ The items that are most often compared are numbers.
- ⇒ The result of a comparison operator is either TRUE or FALSE.

List of Comparison Operators:

- == – Checks if the values of two operands are equal or not, if yes then condition becomes true.
- != – Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.
- > – Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.
- < – Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.
- >= – Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.
- <= – Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.

EXAMPLE:

```
using System;
class Program
{
    static void Main(string[] args)
    {
        int a = 21;
        int b = 10;
        if (a == b)
        {
            Console.WriteLine("Line 1 - a is equal to b");
        }
        else
        {
            Console.WriteLine("Line 1 - a is not equal to b");
        }
        if (a < b)
        {
            Console.WriteLine("Line 2 - a is less than b");
        }
        else
        {
            Console.WriteLine("Line 2 - a is not less than b");
        }
        if (a > b)
        {
            Console.WriteLine("Line 3 - a is greater than b");
        }
        else
```



```
{
Console.WriteLine("Line 3 - a is not greater than b");
}
/* Lets change value of a and b */
a = 5;
b = 20;
if (a <= b)
{
Console.WriteLine("Line 4 - a is either less than or equal to b");
}
if (b >= a)
{
Console.WriteLine("Line 5-b is either greater than or equal to b");
}
}
}
```

OUTPUT:

```
Line 1 - a is not equal to b
Line 2 - a is not less than b
Line 3 - a is greater than b
Line 4 - a is either less than or equal to b
Line 5 - b is either greater than or equal to b
```

Logical Operators:

- ⇒ Logical operators are used to perform logical operation such as and, or.
- ⇒ Logical operators operates on Boolean expressions (true and false) and returns Boolean values.
- ⇒ Logical operators are used in decision making and loops.

The following logical operators operate on Boolean or Integral Operands:

- & – Returns True when both expressions result in a True value
- | – Returns True if at least one expression results in a True value.
- ! – Reverses the outcome of an expression.
- && – Enables you to short-circuit your logical And condition checks.
- || – Enables you to short-circuit your logical Or condition checks.

EXAMPLE: The following example demonstrates all the logical operators available in C# –

```
using System;
namespace OperatorsAppl
{
class Program
{
static void Main(string[] args)
{
bool a = true;
bool b = true;
if (a && b)
{
Console.WriteLine("Line 1 - Condition is true");
}
if (a || b)
{
Console.WriteLine("Line 2 - Condition is true");
}
}
}
/* lets change the value of a and b */
a = false;
b = true;
if (a && b)
```



```
{
Console.WriteLine("Line 3 - Condition is true");
}
else
{
Console.WriteLine("Line 3 - Condition is not true");
}
if (!(a && b))
{
Console.WriteLine("Line 4 - Condition is true");
}
Console.ReadLine();
}
}}
```

OUTPUT:

```
Line 1 - Condition is true
Line 2 - Condition is true
Line 3 - Condition is not true
Line 4 - Condition is true
```

REGULAR EXPRESSION**Q.23. What is Regular Expression? Explain Regex Patterns.**

ASKED YEAR →

IDOL: May – 2017 | May – 2016 | Oct – 2012

SOLUTION**Regular Expression:**

- ⇒ In writing programs or web pages that manipulate text, it is frequently necessary to locate strings that match complex patterns. Regular expressions were invented to describe such patterns.
- ⇒ Thus, a regular expression is just a shorthand code for a pattern.
- ⇒ For example, the pattern "\w+" is a concise way to say "match any non-null strings of alphanumeric characters".
- ⇒ The .NET framework provides a powerful class library that makes it easy to include regular expressions in your applications.
- ⇒ With this library, you can readily search and replace text, decode complex headers, parse languages, or validate text.

Regex Patterns:

- . – Match any character except newline.
- \w – Match any alphanumeric character.
- \s – Match any whitespace character.
- \d – Match any digit.
- \b – Match the beginning or end of a word.
- ^ – Match the beginning of the string.
- \$ – Match the end of the string.
- * – Repeat any number of times.
- + – Repeat one or more times
- ? – Repeat zero or one time.
- {n} – Repeat n times.
- {n,m} – Repeat at least n, but no more than m times.
- {n,} – Repeat at least n times.
- W – Match any character that is NOT alphanumeric.
- \S – Match any character that is NOT whitespace.
- \D – Match any character that is NOT a digit.
- \B – Match a position that is NOT the beginning or end of a word.
- [^x] – Match any character that is NOT x.
- [^aeiou] – Match any character that is NOT one of the characters aeiou.



FLOW CONTROLS

Q.24.

- **What is Flow Control?** [CBSGS: Nov – 2016 | Apr – 2014]
- **Explain use of Break and Continue Statements in loop.** [CBSGS: Nov – 2016 | Apr – 2014]
- **Explain the different keywords used for interrupting the Loops.** [IDOL: May – 2018 | Dec – 2017]

ASKED YEAR ⇒

IDOL: May – 2018 | Dec – 2017

CBSGS: Nov – 2016 | Apr – 2014

SOLUTION

Flow Controls:

- ⇒ The statements inside your source files are generally executed from top to bottom, in the order that they appear.
- ⇒ Control flow statements, however, break up the flow of execution by employing decision making, looping, and branching, enabling your program to conditionally execute particular blocks of code.

Keywords Used For interrupting The Loops:

Break Statement:

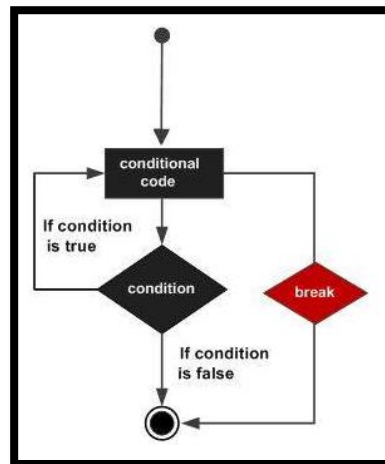
- ⇒ Terminates the loop or switch statement and transfers execution to the statement immediately following the loop or switch.
- ⇒ The break statement in C# has following two usage –
 - When the break statement is encountered inside a loop, the loop is immediately terminated and program control resumes at the next statement following the loop.
 - It can be used to terminate a case in the switch statement.
- ⇒ If you are using nested loops (i.e., one loop inside another loop), the break statement will stop the execution of the innermost loop and start executing the next line of code after the block.

SYNTAX:

The syntax for a break statement in C# is as follows –

```
break;
```

Flow Diagram:



EXAMPLE:

```

using System;
namespace Loops
{
class Program
{
static void Main(string[] args)
{
/* local variable definition */

```



```
int a = 10;
/* do loop execution */
do
{
if (a == 15)
{
/* skip the iteration */
a = a + 1;
continue;
}
Console.WriteLine("value of a: {0}", a);
a++;
}
while (a < 20);
Console.ReadLine();
}
```

OUTPUT:

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
```

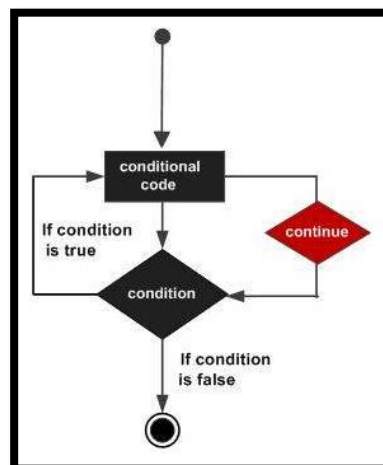
Continue Statement:

- ⇒ It causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.
- ⇒ The continue statement in C# works somewhat like the break statement. Instead of forcing termination, however, continue forces the next iteration of the loop to take place, skipping any code in between.
- ⇒ For the for loop, continue statement causes the conditional test and increment portions of the loop to execute.
- ⇒ For the while and do...while loops, continue statement causes the program control passes to the conditional tests.

SYNTAX:

The syntax for a continue statement in C# is as follows –

```
continue;
```

Flow Diagram:

**EXAMPLE:**

```
using System;
namespace Loops
{
class Program
{
static void Main(string[] args)
{
/* local variable definition */
int a = 10;
/* while loop execution */
while (a < 20)
{
Console.WriteLine("value of a: {0}", a);
a++;
if (a > 15)
{
/* terminate the loop using break statement */
break;
}
}
Console.ReadLine();
}
}
}
```

OUTPUT:

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

FOR LOOP VS. FOREACH LOOP**Q.25.****What is the difference between for loop and foreach loop?**

ASKED YEAR ⇒

IDOL: Apr – 2014

CBSGS: Nov/Apr – 2017

SOLUTION**For Loop Vs. Foreach Loop:**

<u>For Loop</u>	<u>Foreach Loop</u>
In case of for the control variable of the loop is always be int only.	In case of foreach the control variable of the loop while be same as the type of values under the array.
The for loop executes the statement or block of statements repeatedly until specified expression evaluates to false.	The foreach statement repeats a group of embedded statements for each element in an array or an object collection.
There is need to specify the loop Bounds (Minimum, Maximum).	We do not need to specify the loop bounds (Minimum, Maximum).
forloop is complex than foreach loop.	foreach loop is simple than for loop.

**EXAMPLE:**

```
using sytem;
class class1
{
static void Main()
{
int j=0;
for(int i=0; i<=10;i++)
{
j=j+1;
}
Console.ReadLine();
}
}
```

EXAMPLE:

```
using sytem;
class class1
{
static void Main()
{
int j=0;
int[] arr=new int[]
{0,3,5,2,5,34,6,3,42,23};
foreach(int i in arr)
{
j=j+1;
}
Console.ReadLine();
}
}
```

FOREACH LOOP

Q.26.**Explain foreach loop with proper syntax and example.**

ASKED YEAR ⇒

IDOL: Apr – 2013

CBSGS: Nov – 2015 | Nov – 2014

SOLUTION

Foreach Loop:

⇒ A foreach loop enables you to address each element in an array using this simple syntax as follow:

```
foreach (<baseType> <name> in <array>)
{
// can use <name> for each element
}
```

- ⇒ This loop will cycle through each element, placing it in the variable <name> in turn, without danger of accessing illegal elements.
- ⇒ You don't have to worry about how many elements are in the array, and you can be sure that you'll get to use each one in the loop.
- ⇒ The main difference between using this method and a standard for loop is that foreach gives you read-only access to the array contents, so you can't change the values of any of the elements.

EXAMPLE:

```
static void Main(string[] args)
{
string[] friendNames = { "Robert Barwell", "Mike Parry", "Jeremy Beacock" };
Console.WriteLine("Here are {0} of my friends:", friendNames.Length);
foreach (string friendName in friendNames)
{
Console.WriteLine(friendName);
}
Console.ReadKey();
}
```



SWITCH STATEMENT

Q.27. Give general form of Switch Statement. Explain with example.

ASKED YEAR →

CBSGS: Nov – 2014

SOLUTION

Switch Statement:

- ⇒ The Switch Statement is similar to the `if` statement in that it executes code conditionally based on the value of a test.
- ⇒ However, switch enables you to test for multiple values of a test variable in one go, rather than just a single condition.
- ⇒ This test is limited to discrete values, rather than clauses such as "greater than X", so its use is slightly different; but it can be a powerful technique.

General Form of the Switch Statement:

```
switch (expression)
{
    case value-1:
        block-1
        break;
    case value-2:
        block-2
        break;
    ...
    default:
        default-block
        break;
}
statement-x;
```

- ⇒ The expression must be an integer type or char or string type.
- ⇒ value-1, value-2 ... are constants or constant expressions and are known as case labels.
- ⇒ Each of these values should be unique within a switch statement.
- ⇒ block-1, block-2 ... are statement lists and may contain zero or more statements.
- ⇒ There is no need to put braces around these blocks but it is important to note that case labels end with a colon (:).

Basic Structure Of A Switch Statement:

```
switch (<testVar>)
{
    case <comparisonVal1>:
        <code to execute if <testVar> == <comparisonVal1> >
        break;
    case <comparisonVal2>:
        <code to execute if <testVar> == <comparisonVal2> >
        break;
    . . .
    case <comparisonValN>:
        <code to execute if <testVar> == <comparisonValN> >
        break;
    default:
        <code to execute if <testVar> != comparisonVals>
        break;
}
```

EXAMPLE:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```



```
namespace Switch
{
class Program
{
static void Main(string[] args)
{
Console.WriteLine("Enter the month in number");
int m=Convert.ToInt32(Console.ReadLine());
switch (m)
{
case 1:
Console.WriteLine("Jan");
break;
case 2:
Console.WriteLine("Feb");
break;
case 3:
Console.WriteLine("Mar");
break;
case 4:
Console.WriteLine("Apr");
break;
case 5:
Console.WriteLine("May");
break;
case 6:
Console.WriteLine("Jun");
break;
case 7:
Console.WriteLine("Jul");
break;
case 8:
Console.WriteLine("Aug");
break;
case 9:
Console.WriteLine("Sep");
break;
case 10:
Console.WriteLine("Oct");
break;
case 11:
Console.WriteLine("Nov");
break;
case 12:
Console.WriteLine("Dec");
break;
default:
case 2:
Console.WriteLine("Wrong input");
break;
}
Console.ReadKey();
}
}
}
```

**Q.28. Explain Switch Statement. What is Fallthrough in Switch? Is Fallthrough permitted in C#?**

ASKED YEAR →

IDOL: Oct/May – 2016 | Apr – 2013 | Oct – 2012

CBSGS: Nov – 2017

SOLUTION**Switch Statement:**

- ⇒ C# has a built-in multi-way decision statement known as a switch.
- ⇒ The Switch Statement tests the value of a given variable (or expression) against a list of case values and when a match is found, a block of statements associated with that case is executed.

The Switch Statement is executed in the following order:

- ⇒ The expression is evaluated first.
- ⇒ The value of the expression is successively compared against the values, value-1, value-2, ... If a case is found whose value matches the value of the expression, then the block of statements that follows the case are executed.
- ⇒ The break statement at the end of each block signals the end of a particular case and causes an exit from the switch statement, transferring the control to the statement-x following the switch.
- ⇒ The default is an optional case. When present, it will be executed if the value of the expression does not match any of the case values. If not present, no action takes place when all matches fail and the control goes to the statement-x.

Fallthrough In Switch Statement:

- ⇒ In the absence of the break statement in a case block, if the control moves to the next case block without any problem, it is known as 'fallthrough'.
- ⇒ Fallthrough is permitted in C, C++ and Java, but C# does not permit automatic fallthrough, if the case block contains executable code.
- ⇒ However, it is allowed if the case block is empty. For instance the following is an error in C#.

```
switch (m)
{
  case 1:
    x = y;
  case 2:
    x = y + m;
  default:
    x = y - m;
}
```

- ⇒ If one want two consecutive case blocks to be executed continuously, one has to force the process by using the goto statement.

EXAMPLE:

```
switch (m)
{
  case 1:
    x = y;
    goto case 2;
  case 2:
    x = y + m;
    goto default;
  default:
    x = y - m;
    break;
}
```

- ⇒ The go to mechanism enables us to jump backward and forward between cases and therefore arrange labels arbitrarily.

EXAMPLE: The example below is perfectly valid –

```
switch (m)
{
  default:
    x = y-m;
}
```



```
break;
case 2:
x = y + m;
goto default;
case 1:
x = y;
goto case 2;
}
```

METHODS

Q.29. Explain Write () and WriteLine () Methods with examples.

ASKED YEAR ⇒ IDOL: May – 2018 | Dec – 2017 | Oct – 2012

SOLUTION

Write () :

- ⇒ The Write () method outputs one or more values to the screen without a new line character.
- ⇒ Write method resides inside the "Console" class which itself resides inside the System namespace.
- ⇒ The Console class provides basic support for applications that read and write characters to and from the console.
- ⇒ The Write () method outputs one or more values on the screen without a new line character.
- ⇒ This means any subsequent output will be printed in the same line.

WriteLine () :

- ⇒ The WriteLine () always appends a new line character to the end of the string. This means any subsequent output will start on a new line.
- ⇒ WriteLine method also resides inside the "Console" class of the System namespace.
- ⇒ The WriteLine () method prints one or more object on a single line with a new line character inserted at the end.
- ⇒ This means any subsequent output will be printed on a new line.

EXAMPLE:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System;
class Program
{
static void Main()
{
Console.Write("Box");
Console.Write("Table");
Console.Write("chair");
Console.WriteLine();
Console.WriteLine("desk");
}
}
```

OUTPUT:

```
BoxTablechair desk
```

**Q.30. Are multiple Main () methods allowed in C#? Justify with an example.**

ASKED YEAR →

IDOL: May – 2018 | April – 2014

SOLUTION**Multiple Main () Allowed In C#:**

We can use more than one Main Method in C# program, But there will only one Main Method which will act as entry point for the program.

```
using System; namespace ConsoleApplication2
{
    class A
    {
        static void Main(string[] args)
        {
            Console.WriteLine("I am from Class A");
            Console.ReadLine();
        }
    }
    class B
    {
        static void Main(string[] args)
        {
            Console.WriteLine("I am from Class B");
            Console.ReadLine();
        }
    }
}
```

Try to execute Program as given below:

```
csc filename.cs /main:classname
```

As in given program there are two classes A and B in which each containing A Main Method. We may write as.

```
csc filename.cs /main:A [ for Class A Main Execution ] or,
csc filename.cs /main:B [ for Class B Main Execution ]
```

SEALED METHODS**Q.31. What are Sealed Methods? Why are they used?**

ASKED YEAR →

IDOL: Dec – 2017 | Oct – 2012

SOLUTION**Sealed Methods:**

- ⇒ The methods declared with a sealed keyword in its header are known as sealed methods. Such method provides a complete implementation to its base class virtual method using the override keyword.
- ⇒ A method cannot be defined as sealed unless that method is an override of a method in its base class.
- ⇒ A sealed method cannot be overridden further.
- ⇒ Sealed methods are useful to avoid problems caused by overriding the existing functionality.
- ⇒ It prevents the user from changing the internal functionality of a class.

EXAMPLE:

```
using System;
class A
{
    public virtual void F()
    {
```



```
Console.WriteLine("A.F");
}
public virtual void G()
{
    Console.WriteLine("A.G");
}
}
class B: A
{
    sealed override public void F()
    {
        Console.WriteLine("B.F");
    }
    override public void G()
    {
        Console.WriteLine("B.G");
    }
}
class C: B
{
    override public void G()
    {
        Console.WriteLine("C.G");
    }
}
```

⇒ The class B provides two override methods: an F method that has the sealed modifier and a G method that does not. B's use of the sealed modifier prevents C from further overriding F.

READ() VS. READLINE()

Q.32.**What is the difference between Read () and ReadLine () ? Explain with an Example.**

ASKED YEAR ⇒

IDOL: May – 2016

SOLUTION

Read () Vs. ReadLine () :

Read () :

- ⇒ The Read () reads the next characters from the standard input stream.
- ⇒ Accept the string value and return the string value.
- ⇒ console.read () – Reads only one character from the standard input.

```
int a = Console.Read()
Console.WriteLine(a);
```

ReadLine () :

- ⇒ It reads the next line of characters from the standard input stream.
- ⇒ Accept the string and return Integer.
- ⇒ console.readline () – Reads all characters in the line from the standard input.

EXAMPLE:

```
using System;
class Program
{
    static void Main()
    {
        int x = 10;
        Console.WriteLine(x);
        Console.Write("\nPress any key to continue... ");
    }
}
```




```
Console.ReadLine();  
}  
}
```

OUTPUT:

```
10  
Press any key to continue...
```

RESPONSE.REDIRECT() VS. SERVER.TRANSFER()**Q.33.****Explain the difference between `Response.Redirect()` and `Server.Transfer()` method in ASP.NET.**ASKED YEAR ⇒ IDOL: Dec – 2017CBSGS: Apr – 2017 | Apr – 2016**SOLUTION**`Response.Redirect()` :

This method redirects a client to a new URL. Specifies a new URL and whether execution of the current page should terminate.

SYNTAX:

```
public void Redirect(string url, bool endResponse)
```

- ⇒ Here `url` is a string value specifies the location of the target
- ⇒ `endResponse` is a boolean value indicates whether execution of the current pages should terminate.

EXAMPLE:

```
Response.Redirect(https://www.google.com, false);
```

- ⇒ The above code will redirect you to google page & second argument "`false`" indicates we do not want an exception to be thrown to terminate the page.
- ⇒ The `Response.Redirect` method redirects a request to a new URL and specifies the new URL.

`Server.Transfer()` :

- ⇒ `server.transfer` method for the current request, terminates execution of the current page and starts execution of a new page using the specifies URL path of the page.
- ⇒ You can't use `transfer()` to send the user to another website or to a non-ASP.net page (such as an HTML page). The `transfer()` method only allows you to jump from one ASP.NET page to another, in the same web application.
- ⇒ When you use `transfer()` the user won't have any idea that another page has taken over, because the browser will still show the original URL.

**EXCEPTION****Q.34.**

- **What is an Exception? Explain Exception Handling in C#. [CBSGS: Nov – 2017]**
- **Explain briefly try, catch and throw statements in C#. [CBSGS: Nov/Apr – 2016]**
- **What is Exception Handling? Explain the Syntax of Exception Handling Code. What is the use of finally Block? [IDOL: May – 2018]**
- **Write a short note on Exception Handling in C#. [IDOL: Oct – 2016]**

ASKED YEAR ⇒

IDOL: May – 2018 | Oct – 2016

CBSGS: Nov – 2017 | Nov/Apr – 2016

SOLUTION**Exception:**

- ⇒ An exception is a problem that arises during the execution of a program.
- ⇒ A C# exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero.
- ⇒ Exceptions provide a way to transfer control from one part of a program to another.

Exception Handling:

C# exception handling is built upon four keywords: try, catch, finally, and throw.

Try:

- ⇒ A try block identifies a block of code for which particular exceptions is activated.
- ⇒ It is followed by one or more catch blocks.

Catch:

- ⇒ A program catches an exception with an exception handler at the place in a program where you want to handle the problem.
- ⇒ The catch keyword indicates the catching of an exception.

Finally:

- ⇒ The finally block is used to execute a given set of statements, whether an exception is thrown or not thrown.
- ⇒ For example, if you open a file, it must be closed whether an exception is raised or not.

Throw:

- ⇒ A program throws an exception when a problem shows up.
- ⇒ This is done using a throw keyword.

SYNTAX: Syntax for using try/catch looks like the following –

```
try
{
// statements causing exception
}
catch( ExceptionName e1 )
{
// error handling code
}
catch( ExceptionName e2 )
{
// error handling code
}
catch( ExceptionName eN )
{
// error handling code
}
```



```
finally
{
// statements to be executed
}
```

EXAMPLE:

```
using System;
namespace ErrorHandlingApplication
{
class DivNumbers
{
int result;
DivNumbers()
{
result = 0;
}
public void division(int num1, int num2)
{
try
{
result = num1 / num2;
}
catch (DivideByZeroException e)
{
Console.WriteLine("Exception caught: {0}", e);
}
Finally
{
Console.WriteLine("Result: {0}", result);
}
}
static void Main(string[] args)
{
DivNumbers d = new DivNumbers();
d.division(25, 0);
Console.ReadKey();
}
}
}
```

OUTPUT:

```
Exception caught: System.DivideByZeroException: Attempted to divide by zero.
at ...
Result: 0
```



Q.35.

- Why Exception Handling is required? Write syntax for user define Exception? [CBSGS: Apr – 2015]
- What is a User Defined Exception? Explain with Syntax. [CBSGS: Apr – 2017]

ASKED YEAR →

CBSGS: Apr – 2017 | Apr – 2015

SOLUTION

Why Exception Handling Is Required?

- ⇒ When our application encounters an exceptional circumstance, such as division by zero or low memory warning, an exception is generated.
- ⇒ Use a try block around the statements that might throw exceptions.
- ⇒ If a catch block defines an exception variable, we can use it to get more information on the type of exception that occurred.
- ⇒ Actions that may result in an exception are executed with the try keyword.
- ⇒ An exception handler is a block of code that is executed when an exception occurs. In C#, the catch keyword is used to define an exceptional handler.
- ⇒ Exceptions can be explicitly generated by a program using the throw keyword.
- ⇒ Exception object contain detailed information about the error, including the state of the call stack and a text description of the error.
- ⇒ Code in a finally block is executed even if an exception is thrown, thus allowing a program to release resources.
- ⇒ That is why to deal with any unexpected or exceptional situations that arise while program is running, exception handling is required.

Syntax for User Defined Exceptions:

- ⇒ To throw the user defined exceptions. 'throw' keyword is used.
`throw new Throwable_subclass;`
- ⇒ For this, a class is to be created which is inherited from class Exception.

EXAMPLE:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication2
{
    using System;
    class MyException:Exception
    {
    public MyException(string str):base(str)
    {
    }
    }
    class MyClient
    {
    public static void Main()
    {
    int x=5, y=1000;
    try
    {
    float z=(float)x / (float)y;
    if(z<0.1)
    throw new MyException("Number is too small...");
    }
    catch (MyException e)
    {
    Console.WriteLine("Exception caught here...." + e.Message);
    }
    Console.WriteLine("LAST STATEMENT");
    Console.ReadKey();
    }
```



```

}
}
}

```

JAGGED ARRAY / VARIABLE SIZED ARRAYS / ARRAY OF ARRAYS

Q.36.

- Explain Variable Sized Arrays with suitable example. [IDOL: May – 2018 | April – 2014]
- What is a Jagged Array? Explain with example. [CBSGS: Nov – 2015]
- Write a note on "Array Of Arrays". [IDOL: Apr – 2013]

ASKED YEAR ⇒

IDOL: May – 2018 | April – 2014 | Apr – 2013

CBSGS: Nov – 2015

SOLUTION

Jagged Array / Variable Sized Arrays / Array of Arrays:

- ⇒ A Jagged Array is an array whose elements are arrays.
- ⇒ Variable Sized Arrays are called as Jagged Array.
- ⇒ The elements of a jagged array can be of different dimensions and sizes.
- ⇒ A Jagged Array is sometimes called an "array of arrays".
- ⇒ A Jagged Array is an array of an array in which the length of each array index can differ.
- ⇒ We can declare a jagged array named scores of type int as –

```
int [][] scores;
```

- ⇒ Declaring an array, does not create the array in memory. To create the above array –

```
int[][] scores = new int[5][];
for (int i = 0; i < scores.Length; i++)
{
    scores[i] = new int[4];
}
```

- ⇒ You can initialize a jagged array as –

```
int[][] scores = new int[2][]{new int[]{92,93,94},new int[]{85,66,87,88}};
```

- ⇒ Where, scores is an array of two arrays of integers - scores[0] is an array of 3 integers and scores[1] is an array of 4 integers.

EXAMPLE:

The following example illustrates using a jagged array –

```
using System;
namespace ArrayApplication
{
    class MyArray
    {
        static void Main(string[] args)
        {
            /* a jagged array of 5 array of integers*/
            int[][] a = new int[][]{new int[]{0,0},new int[]{1,2},
            new int[]{2,4},new int[]{ 3, 6 }, new int[]{ 4, 8 } };
            int i, j;
            /* output each array element's value */
            for (i = 0; i < 5; i++)
            {
                for (j = 0; j < 2; j++)
                {
                    Console.WriteLine("a[{0}][{1}] = {2}", i, j, a[i][j]);
                }
            }
            Console.ReadKey();
        }
    }
}
```



```
}
```

OUTPUT:

```
a[0][0]: 0
a[0][1]: 0
a[1][0]: 1
a[1][1]: 2
a[2][0]: 2
a[2][1]: 4
a[3][0]: 3
a[3][1]: 6
a[4][0]: 4
a[4][1]: 8
```

ARRAYLIST

Q.37.**What is ArrayList? State its methods and properties.**

ASKED YEAR ⇒

IDOL: May – 2018

SOLUTION

ArrayList:

- ⇒ ArrayList is a non-generic type of collection in C#.
- ⇒ It can contain elements of any data types.
- ⇒ It is similar to an array, except that it grows automatically as you add items in it.
- ⇒ Unlike an array, you don't need to specify the size of ArrayList.

EXAMPLE: Initialize ArrayList

```
ArrayList myArrayList = new ArrayList();
```

Property of ArrayList:

- **Capacity:** Gets or sets the number of elements that the ArrayList can contain.
- **Count:** Gets the number of elements actually contained in the ArrayList.
- **IsFixedSize:** Gets a value indicating whether the ArrayList has a fixed size.
- **IsReadOnly:** Gets a value indicating whether the ArrayList is read-only.
- **Item:** Gets or sets the element at the specified index.

Methods of ArrayList:

- **Add() / AddRange() :** Add() method adds single elements at the end of ArrayList.
- **AddRange() :** method adds all the elements from the specified collection into ArrayList.
- **Insert() / InsertRange() :** Insert() method insert a single elements at the specified index in ArrayList.
- **InsertRange() :** Method insert all the elements of the specified collection starting from specified index in ArrayList.
- **Remove() / RemoveRange() :** Remove() method removes the specified element from the ArrayList.
- **RemoveRange() :** method removes a range of elements from the ArrayList.
- **RemoveAt() :** Removes the element at the specified index from the ArrayList.
- **Sort() :** Sorts entire elements of the ArrayList.
- **Reverse() :** Reverses the order of the elements in the entire ArrayList.
- **Contains:** Checks whether specified element exists in the ArrayList or not. Returns true if exists otherwise false.
- **Clear:** Removes all the elements in ArrayList.
- **CopyTo:** Copies all the elements or range of elements to compatible Array.
- **GetRange:** Returns specified number of elements from specified index from ArrayList.
- **IndexOf:** Search specified element and returns zero based index if found. Returns -1 if element not found.
- **ToArray:** Returns compatible array from an ArrayList.

**EXAMPLE:**

```

using System;
using System.Collections;
namespace CollectionApplication
{
class Program
{
static void Main(string[] args)
{
ArrayList al = new ArrayList();
Console.WriteLine("Adding some numbers:");
al.Add(45);
al.Add(78);
al.Add(33);
al.Add(56);
al.Add(12);
al.Add(23);
al.Add(9);
Console.WriteLine("Capacity: {0} ", al.Capacity);
Console.WriteLine("Count: {0}", al.Count);
Console.Write("Content: ");
foreach (int i in al)
{
Console.Write(i + " ");
}
Console.WriteLine();
Console.Write("Sorted Content: ");
al.Sort();
foreach (int i in al)
{
Console.Write(i + " ");
}
Console.WriteLine();
Console.ReadKey();
}
}
}

```

Output:

```

Adding some numbers:
Capacity: 8
Count: 7
Content: 45 78 33 56 12 23 9
Content: 9 12 23 33 45 56 78

```

ARRAY VS. ARRAYLIST**Q.38.****What is the difference between Arrays and ArrayLists? Explain with a program.**

ASKED YEAR ⇒

IDOL: Apr – 2014

SOLUTION**Array Vs. ArrayList:**

<u>Array</u>	<u>ArrayList</u>
Array is strongly typed. This means that an array can store only specific type of items/elements.	ArrayList can store any type of items/elements.
Array stores fixed number of elements. Size of an Array must be specified at the time of initialization.	ArrayList grows automatically and you don't need to specify size.



No need to cast elements of an array while retrieving because it is strongly type and stores specific type of items only.	Items of ArrayList need to be cast to appropriate data type while retrieving.
Use static helper class Array to perform different tasks on the array.	ArrayList itself includes various utility methods for various tasks.
In arrays we can store only one datatype either int, string, char etc.	In ArrayList we can store all the datatype values.
Array can't accept null.	ArrayList collection accepts null.
Arrays belong to System.Array namespace: <code>using System;</code>	ArrayList belongs to System.Collection namespaces: <code>using System.Collections;</code>
EXAMPLE: <code>int[] intArray=new int[]{2}; intArray[0] = 1; intArray[2] = 2;</code>	EXAMPLE: <code>ArrayList Arrlst = new ArrayList(); Arrlst.Add("Sagar"); Arrlst.Add(1); Arrlst.Add(null);</code>

PARAMETERS

Q.39. Explain References Parameters and Output Parameters with a program.

ASKED YEAR → IDOL: Apr – 2014 | Apr – 2013

CBSGS: Nov – 2015

SOLUTION

References Parameters:

- ⇒ It is used as a call by reference in C#.
- ⇒ It is used in both places; when we declare a method and when we call the method.
- ⇒ In this example we create a function (cube) and pass a ref parameter in it, now we look at the value before we call the function and after we call the function.

EXAMPLE:

```
class TryRef
{
public void cube(ref int x)
{
x= x * x * x;
}
}
class Program
{
static void Main(string[] args)
{
TryRef tr = new TryRef();
int y = 5;
Console.WriteLine("The value of y before the function call: " + y);
tr.cube(ref y);
Console.WriteLine("The value of y after the function call: " + y);
Console.ReadLine();
}
}
```

OUTPUT:

```
The value of y before the function call: 5
The value of y after the function call: 125
```



Output Parameters:

- ⇒ Sometimes we do not want to give an initial value to the parameter; in this case we use the out parameter.
- ⇒ The declaration of the out parameter is the same as the ref parameter.
- ⇒ But it is used to pass the value out of a method.

EXAMPLE:

```
class tryout
{
public int mul(int a, out int b)
{
b = a * a;
return b;
}
}
class Program
{
static void Main(string[] args)
{
tryout to = new tryout();
int x,y;
x = to.mul(10, out y);
Console.WriteLine("The output is: "+x);
Console.ReadLine();
}
}
```

OUTPUT:

```
The output is: 100
```

ACCESS MODIFIERS

Q.40. Explain the different Access Modifiers.

ASKED YEAR ⇒ IDOL: May – 2017

SOLUTION

Access Modifiers:

- ⇒ Access Modifiers (Access Specifiers) describes as the scope of accessibility of an Object and its members.
- ⇒ All C# types and type members have an accessibility level.
- ⇒ We can control the scope of the member object of a class using access specifiers.
- ⇒ We are using access modifiers for providing security of our applications.
- ⇒ When we specify the accessibility of a type or member we have to declare it by using any of the access modifiers provided by C# language.

Types Of Access Modifiers:

C# provide five access specifiers, they are as follows:

- 1) *public*
- 2) *private*
- 3) *protected*
- 4) *internal*
- 5) *protected internal*

public:

- ⇒ Public is the most common access specifier in C#.
- ⇒ It can be access from anywhere, which means there is no restriction on accessibility.



- ⇒ The scope of the accessibility is inside class as well as outside.
- ⇒ The type or member can be accessed by any other code in the same assembly or another assembly that references it.

private:

- ⇒ The scope of the accessibility is limited only inside the classes or struct in which they are declared.
- ⇒ The private members cannot be accessed outside the class and it is the least permissive access level.

protected:

- ⇒ The scope of accessibility is limited within the class or struct and the class derived (Inherited) from this class.

internal:

- ⇒ The internal access modifiers can access within the program that contain its declarations and also access within the same assembly level but not from another assembly.

protected internal:

- ⇒ Protected internal is the same access levels of both protected and internal.
- ⇒ It can access anywhere in the same assembly and in the same class also the classes inherited from the same class.

CLASS

Q.41.

Define each of the following terms:

- *Derived Class*
- *Abstract Class*
- *Static Class*
- *Sealed Class*
- *Partial Class*

ASKED YEAR →

CBSGS: Nov – 2015

SOLUTION

Derived Class:

- ⇒ Derived Class is a subclass which is created base class or the main class.
- ⇒ The initial class that is used as a basis class or superclass.
- ⇒ The derived class incorporates all the data of base class.

Abstract Class:

- ⇒ In hierarchical applications one base class simply acts as a base for others and is not useful on its own.
- ⇒ We can do this by creating the base class abstract by using abstract modifies.

Static Class:

- ⇒ A static class is a class whose objects cannot be created and must contain only static members.

Sealed Class:

- ⇒ A class that cannot be subclassed is called sealed class.
- ⇒ This is achieved by using modifier sealed.

Partial Class:

- ⇒ Each class in C# resides in a separate physical file with a .cs extension.
- ⇒ C# provides the ability to have a single class implementation in multiple .cs files using the partial modifier keyword.
- ⇒ The partial modifier can be applied to a class, method, interface or structure.



SEALED CLASS

Q.42. What are Sealed Classes? Why are they used?

ASKED YEAR → **IDOL:** Dec – 2017 | Oct – 2012

SOLUTION

Sealed Classes:

- ⇒ Generally if we create classes we can inherit the properties of that created class in any class without having any restrictions.
- ⇒ In some situation we will get requirement like we don't want to give permission for the users to derive the classes from it or don't allow users to inherit the properties from particular class.
- ⇒ For that purpose we have keyword called "Sealed" in OOPS.
- ⇒ When we defined class with keyword "Sealed" then we don't have a chance to derive that particular class and we don't have permission to inherit the properties from that particular class.
- ⇒ A sealed class cannot be inherited.
- ⇒ It is an error to use a sealed class as a base class.
- ⇒ Use the sealed modifier in a class declaration to prevent inheritance of the class.
- ⇒ It is not permitted to use the abstract modifier with a sealed class.
- ⇒ Structs are implicitly sealed; therefore, they cannot be inherited.

EXAMPLE:

```
using System;
sealed class MyClass
{
    public int x;
    public int y;
}
class MainClass
{
    public static void Main()
    {
        MyClass mC = new MyClass();
        mC.x = 110;
        mC.y = 150;
        Console.WriteLine("x = {0}, y = {1}", mC.x, mC.y);
    }
}
```

PAGE CLASS

Q.43. List and Explain Properties of Page Class.

ASKED YEAR → **IDOL:** Apr – 2015

SOLUTION

Properties of Page Class:

- **Session:** Gets the current Session object provided by ASP.NET.
- **Application:** Gets the `HttpApplicationState` object for the current Web request.
- **Cache:** Gets the Cache object associated with the application in which the page resides.
- **Request:** Gets the `HttpRequest` object for the requested page.
- **Response:** Gets the `HttpResponse` object associated with the Page object. This object allows you to send HTTP response data to a client and contains information about that response.
- **Server:** Gets the Server object, which is an instance of the `HttpServerUtility` class.
- **User:** Gets information about the user making the page request.
- **Trace:** Gets the `TraceContext` object for the current Web request.

**PROTECTED DATA MEMBER****Q.44.** What is the scope of protected Data Members of a class?

ASKED YEAR →

CBSGS: Nov – 2015

SOLUTION**Protected Data Member:**

- ⇒ A protected member variable or function is very similar to a private member but it provided one additional benefit that they can be accessed in child classes which are called derived classes.
- ⇒ The protected members is visible only to its own class and its derived class.

EXAMPLE:

```
#include <iostream>
using namespace std;
class Box
{
protected:
double width;
};
class SmallBox:Box
{
// SmallBox is the derived class.
public:
void setSmallWidth( double wid );
double getSmallWidth( void );
};
// Member functions of child class
double SmallBox::getSmallWidth(void)
{
return width ;
}
void SmallBox::setSmallWidth( double wid )
{
width = wid;
}
// Main function for the program
int main()
{
SmallBox box;
// set box width using member function
box.setSmallWidth(5.0);
cout << "Width of box : "<< box.getSmallWidth() << endl;
return 0;
}
```

OUTPUT:**Width of box : 5**

STRUCTURE VS. CLASSES**Q.45. Differentiate between Structures and Classes.**

ASKED YEAR ⇒

IDOL: May – 2018 | Apr – 2014

SOLUTIONStructure Vs. Classes:

<u>Structure</u>	<u>Classes</u>
The struct is value type in C# and it inherits from <code>System.ValueType</code> .	The class is reference type in C# and it inherits from the <code>System.Object</code> Type
The struct value will be stored on the stack memory.	The class object is stored on the heap memory.
The struct can only inherit the interfaces.	The class can inherit the interfaces, abstract classes.
The struct can have only constructor.	The class can have the constructor and destructor.
The struct can instantiated without using the new keyword.	The new keyword should be used to create the object for the class.
The struct can't have the default constructor.	The class will have the default constructor.
The struct is by default sealed class hence it will not allow to inherit. It can't use the abstract, sealed, base keyword.	The class can be declared as abstract, sealed class.

OBJECT**Q.46. Explain Request and Response Objects of ASP.NET.**

ASKED YEAR ⇒

IDOL: May – 2018 | Oct – 2016 | Oct – 2012

SOLUTIONRequest Object:

- ⇒ This object is mainly used to retrieve the information from the form in a HTML Page.
- ⇒ The request object is an instance of the `System.Web.HttpRequest` class.
- ⇒ It represents the values and properties of the HTTP request that makes the page loading into the browser.
- ⇒ The information presented by this object is wrapped by the higher level abstractions (the web control model).
- ⇒ However, this object helps in checking some information such as the client browser and cookies.

Properties Of Request Object:

- **AcceptTypes**: Gets a string array of client-supported MIME accept types.
- **ApplicationPath**: Gets the ASP.NET application's virtual application root path on the server.
- **Browser**: Gets or sets information about the requesting client's browser capabilities.
- **Cookies**: Gets a collection of cookies sent by the client.
- **HttpMethod**: Gets the HTTP data transfer method (such as GET, POST, or HEAD) used by the client.
- **FilePath**: Gets the virtual path of the current request.
- **Files**: Gets the collection of files uploaded by the client, in multipart MIME format.
- **Form**: Gets a collection of form variables.
- **Headers**: Gets a collection of HTTP headers.
- **HttpMethod**: Gets the HTTP data transfer method (such as GET, POST, or HEAD) used by the client.
- **RequestType**: Gets or sets the HTTP data transfer method (GET or POST) used by the client.
- **Url**: Gets information about the URL of the current request.

Method Of Request Object:

- **BinaryRead**: Performs a binary read of a specified number of bytes from the current input stream.
- **Equals (Object)**: Determines whether the specified object is equal to the current object. (Inherited from object.)
- **GetType**: Gets the Type of the current instance.



- **MapImageCoordinates** : Maps an incoming image-field form parameter to appropriate x-coordinate and y-coordinate values.
- **MapPath (String)** : Maps the specified virtual path to a physical path.
- **SaveAs** : Saves an HTTP request to disk.
- **ToString** : Returns a String that represents the current object.
- **ValidateInput** : Causes validation to occur for the collections accessed through the Cookies, Form, and `QueryString` properties.

Response Object:

- ⇒ The Response Object represents the server's response to the client request.
- ⇒ It is an instance of the `System.Web.HttpResponse` class.
- ⇒ In ASP.NET, the response object does not play any vital role in sending HTML text to the client, because the server-side controls have nested, object oriented methods for rendering themselves.
- ⇒ However, the `HttpResponse` object still provides some important functionalities, like the cookie feature and the `Redirect()` method.
- ⇒ The `Response.Redirect()` method allows transferring the user to another page, inside as well as outside the application.
- ⇒ It requires a round trip.

Property Of Response Object:

- **Buffer** : Gets or sets a value indicating whether to buffer the output and send it after the complete response is finished processing.
- **BufferOutput** : Gets or sets a value indicating whether to buffer the output and send it after the complete page is finished processing.
- **Charset** : Gets or sets the HTTP character set of the output stream.
- **ContentEncoding** : Gets or sets the HTTP character set of the output stream.
- **ContentType** : Gets or sets the HTTP MIME type of the output stream.
- **Cookies** : Gets the response cookie collection.
- **Expires** : Gets or sets the number of minutes before a page cached on a browser expires.
- **ExpiresAbsolute** : Gets or sets the absolute date and time at which to remove cached information from the cache.
- **HeaderEncoding** : Gets or sets an encoding object that represents the encoding for the current header output stream.
- **Headers** : Gets the collection of response headers.
- **IsClientConnected** : Gets a value indicating whether the client is still connected to the server.
- **Output** : Enables output of text to the outgoing HTTP response stream.
- **OutputStream** : Enables binary output to the outgoing HTTP content body.
- **RedirectLocation** : Gets or sets the value of the Http Location header.
- **Status** : Sets the status line that is returned to the client.

Methods Of Response Object:

- **AddHeader** : Adds an HTTP header to the output stream. `AddHeader` is provided for compatibility with earlier versions of ASP.
- **AppendCookie** : Infrastructure adds an HTTP cookie to the intrinsic cookie collection.
- **AppendHeader** : Adds an HTTP header to the output stream.
- **AppendToLog** : Adds custom log information to the InterNET Information Services (IIS) log file.
- **BinaryWrite** : Writes a string of binary characters to the HTTP output stream.
- **ClearContent** : Clears all content output from the buffer stream.
- **Close** : Closes the socket connection to a client.
- **End** : Sends all currently buffered output to the client, stops execution of the page, and raises the `EndRequest` event.
- **GetType** : Gets the Type of the current instance.
- **Pics** : Appends a HTTP PICS-Label header to the output stream.
- **Redirect (String)** : Redirects a request to a new URL and specifies the new URL.
- `Redirect(String, Boolean)` : Redirects a client to a new URL. Specifies the new URL and whether execution of the current page should terminate.



- **SetCookie**: Updates an existing cookie in the cookie collection.
- **ToString**: Returns a String that represents the current Object.

SYNTAX:

The syntax to use these method is

```
Response.Method
```

EXAMPLE:

```
Response.Redirect("newpage.html")
```

⇒ The following statement will write the string inside parenthesis on the browser screen.

```
<% Response.Write("color=red">Text from Write Method")%>
```

INTERFACES

Q.47. What are Interfaces? Write a program to show the implementation of Multiple Interfaces.

ASKED YEAR ⇒

IDOL: Dec – 2017

SOLUTION

Interfaces:

- ⇒ An interface is defined as a syntactical contract that all the classes inheriting the interface should follow.
- ⇒ The interface defines the 'what' part of the syntactical contract and the deriving classes define the 'how' part of the syntactical contract.
- ⇒ Interfaces define properties, methods, and events, which are the members of the interface.
- ⇒ Interfaces contain only the declaration of the members.
- ⇒ It is the responsibility of the deriving class to define the members.
- ⇒ It often helps in providing a standard structure that the deriving classes would follow.
- ⇒ Abstract classes to some extent serve the same purpose, however, they are mostly used when only few methods are to be declared by the base class and the deriving class implements the functionalities.

Declaring Interfaces:

- ⇒ Interfaces are declared using the interface keyword.
- ⇒ It is similar to class declaration.
- ⇒ Interface statements are public by default.
- ⇒ Following is an example of an interface declaration –

EXAMPLE:

```
public interface ITransactions
{
    // interface members
    void showTransaction();
    double getAmount();
}
```

Example: The following example demonstrates implementation of the above interface –

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System;
namespace InterfaceApplication
{
    public interface ITransactions
    {
        // interface members
        void showTransaction();
    }
}
```




```
double getAmount();
}
public class Transaction : ITransactions
{
    private string tCode;
    private string date;
    private double amount;
    public Transaction()
    {
        tCode = " ";
        date = " ";
        amount = 0.0;
    }
    public Transaction(string c, string d, double a)
    {
        tCode = c;
        date = d;
        amount = a;
    }
    public double getAmount()
    {
        return amount;
    }
    public void showTransaction()
    {
        Console.WriteLine("Transaction: {0}", tCode);
        Console.WriteLine("Date: {0}", date);
        Console.WriteLine("Amount: {0}", getAmount());
    }
}
class Tester
{
    static void Main(string[] args)
    {
        Transaction t1 = new Transaction("001", "8/10/2012", 78900.00);
        Transaction t2 = new Transaction("002", "9/10/2012", 451900.00);
        t1.showTransaction();
        t2.showTransaction();
        Console.ReadKey();
    }
}
```

OUTPUT:

```
Transaction: 001
Date: 8/10/2012
Amount: 78900
Transaction: 002
Date: 9/10/2012
Amount: 451900
```

**ABSTRACT CLASS VS. INTERFACE****Q.48. Distinguish between Abstract Class and Interface.**

ASKED YEAR →

IDOL: Dec – 2017 | Apr – 2015

CBSGS: Apr – 2017 | Nov – 2014

SOLUTION**Abstract Class Vs. Interface:**

<u>Abstract Class</u>	<u>Interface</u>
A class may inherit only one Abstract Class.	A class may inherit several interfaces.
It can provide complete default code and/or just the details that have to be overridden.	An interface cannot provide any code just the signature.
It requires less time to find the actual method.	It requires more time to find the actual method in the corresponding classes.
An Abstract Class can contain access modifiers for the functions, properties, etc.	An interface cannot have access modifiers for the functions, properties etc. Everything is assumed as public.
An Abstract Class defines the lose identity of a class & there it is used for objects of the same type.	Interfaces are used to define the peripheral abilities of a class.
If various implementations are of the same bind and use common behaviours or status, then abstract class is better to use.	If various implementations only share method signatures then it is better to use interfaces.

Q.49. Explain the similarities between Interfaces and Abstract Classes.

ASKED YEAR →

CBSGS: Nov – 2014

SOLUTION**Similarities Of Interfaces And Abstract Classes:**

- ⇒ Neither Abstract classes nor Interfaces can be instantiated.
- ⇒ Both abstract classes and interfaces may contain members that can be inherited by a derived class.
- ⇒ Neither interfaces nor abstract classes may be directly instantiated, but you can declare variables of these types.
- ⇒ If you do, you can use polymorphism to assign objects that inherit from these types to variables of these types.
- ⇒ In both cases, you can then use the members of these types through these variables, although you don't have direct access to the other members of the derived object.

Q.50. When is Explicit Interface necessary?

ASKED YEAR →

IDOL: Apr – 2014

SOLUTION**Necessary Of Explicit Interface:**

- ⇒ C# does not support multiple inheritance, but a class has the option of implementing one or more interfaces.
- ⇒ One challenge with interfaces is that they may include methods that have the same signatures as existing class members or members of other interfaces.
- ⇒ Explicit interface implementations can be used to disambiguate class and interface methods that would otherwise conflict.
- ⇒ Explicit interfaces can also be used to hide the details of an interface that the class developer considers private.
- ⇒ To explicitly implement an interface member, just use it's fully qualified name in the declaration.
- ⇒ A fully qualified interface name takes the form `InterfaceName . MemberName`

EXAMPLE:

```
interface I1
{
void A();
}
interface I2
```



```
{
void A();
}
//here we add an explicit implementation of I1's A() method.
//Class C explicitly implements I1.A().
class C : I1, I2 { public void A()
{
Console.WriteLine("C.A()");
}
void I1.A()
{
Console.WriteLine("I1.A()");
}
}
```

CONSTRUCTOR

Q.51.

- **What is Constructor? Explain Parameterized Constructor with suitable example. [CBSGS: Nov – 2016]**
- **What are the different types of constructors in C#? [IDOL: Apr – 2014]**
- **Explain Static Constructor with example. [CBSGS: Nov – 2015]**
- **What are the rules in defining a Constructor? [CBSGS: Nov – 2015]**

ASKED YEAR ⇒

IDOL: Apr – 2014

CBSGS: Nov – 2016 | Nov – 2015

SOLUTION

Constructor:

- ⇒ Constructor is a special method of the class that will be automatically invoked when an instance of the class is created is called a constructor.
- ⇒ The main use of constructors is to initialize private fields of the class while creating an instance for the class. When you have not created a constructor in the class, the compiler will automatically create a default constructor in the class.
- ⇒ The default constructor initializes all numeric fields in the class to zero and all string and object fields to null.

Rules in defining a Constructor:

- ⇒ A constructor should have the same name as that of the class.
- ⇒ Constructors can take only number of arguments constructors do not return values.
- ⇒ Constructors are usually public.

Types Of Constructors In C#:

Constructors can be divided into 5 types:

- 1) *Default Constructor*
- 2) *Parametrized Constructor*
- 3) *Copy Constructor*
- 4) *Static Constructor*
- 5) *Private Constructor*

Default Constructors:

- ⇒ A constructor without any parameters is called a default constructor; in other words this type of constructor does not take parameters.
- ⇒ The drawback of a default constructor is that every instance of the class will be initialized to the same values and it is not possible to initialize each instance of the class to different values.
- ⇒ The default constructor initializes:
 - *All numeric fields in the class to zero.*
 - *All string and object fields to null.*

**EXAMPLE:**

```
using System;
namespace DefaultConstructor
{
class addition
{
int a, b;
public addition()    //default constructor
{
a = 100;
b = 175;
}
public static void Main()
{
addition obj = new addition();
//an object is created , constructor is called
Console.WriteLine(obj.a);
Console.WriteLine(obj.b);
Console.Read();
}
}
}
```

OUTPUT:

```
100
175
```

Parameterized Constructor:

- ⇒ A constructor with at least one parameter is called a parametrized constructor.
- ⇒ The advantage of a parametrized constructor is that you can initialize each instance of the class to different values.

EXAMPLE:

```
using System;
namespace Constructor
{
class paraconstructor
{
public int a, b;
public paraconstructor(int x, int y)
// decalaring Parametrized Constructor with ing x,y parameter
{
a = x;
b = y;
}
}
class MainClass
{
static void Main()
{
paraconstructor v = new paraconstructor(100, 175);    // Creating object of Parameterized
Constructor and ing values
Console.WriteLine("-----parameterized constructor example-----");
Console.WriteLine("\t");
Console.WriteLine("value of a=" + v.a );
Console.WriteLine("value of b=" + v.b);
Console.Read();
}
}
```



}

OUTPUT:

```
-----parameterized constructor example-----  
value of a=100  
value of b=175
```

Copy Constructor:

- ⇒ The constructor which creates an object by copying variables from another object is called a copy constructor.
- ⇒ The purpose of a copy constructor is to initialize a new instance to the values of an existing instance.

SYNTAX:

```
public employee(employee emp)  
{  
    name=emp.name;  
    age=emp.age;  
}
```

EXAMPLE:

```
using System;  
namespace copyConstructor  
{  
    class employee  
    {  
        private string name;  
        private int age;  
        public employee(employee emp) // declaring Copy constructor.  
        {  
            name = emp.name;  
            age = emp.age;  
        }  
        public employee(string name, int age) // Instance constructor.  
        {  
            this.name = name;  
            this.age = age;  
        }  
        public string Details // Get deatils of employee  
        {  
            get  
            {  
                return " The age of " + name + " is " + age.ToString();  
            }  
        }  
    }  
    class empdetail  
    {  
        static void Main()  
        {  
            employee emp1 = new employee("bscit", 23); // Create a new employee object.  
            employee emp2 = new employee(emp1); // here is emp1 details is copied to emp2.  
            Console.WriteLine(emp2.Details);  
            Console.ReadLine();  
        }  
    }  
}
```

**OUTPUT:**

```
The age of bscit is 23
```

Static Constructor:

- ⇒ When a constructor is created as static, it will be invoked only once for all of instances of the class and it is invoked during the creation of the first instance of the class or the first reference to a static member in the class. A static constructor is used to initialize static fields of the class and to write the code that needs to be executed only once.
- ⇒ A static constructor does not take access modifiers or have parameters.
- ⇒ A static constructor is called automatically to initialize the class before the first instance is created or any static members are referenced.
- ⇒ A static constructor cannot be called directly.
- ⇒ The user has no control on when the static constructor is executed in the program.
- ⇒ A typical use of static constructors is when the class is using a log file and the constructor is used to write entries to this file.

SYNTAX:

```
class employee
{
// Static constructor
static employee()
}
```

EXAMPLE:

```
using System;
namespace staticConstructor
{
public class employee
{
static employee() // Static constructor declaration
{
Console.WriteLine("The static constructor ");
}
public static void Salary()
{
Console.WriteLine();
Console.WriteLine("The Salary method");
}
}
class details
{
static void Main()
{
Console.WriteLine("-----Static constructor example-----");
Console.WriteLine();
employee.Salary();
Console.ReadLine();
}
}
}
```

OUTPUT:

```
-----Static constructor example-----
The static constructor
The Salary method
```



Private Constructor:

- ⇒ Private constructors are used to restrict the instantiation of object using 'new' operator.
- ⇒ A private constructor is a special instance constructor. It is commonly used in classes that contain static members only.
- ⇒ This type of constructors is mainly used for creating singleton object.
- ⇒ If you don't want the class to be inherited we declare its constructor private.
- ⇒ We can't initialize the class outside the class or the instance of class can't be created outside if its constructor is declared private.
- ⇒ We have to take help of nested class (Inner Class) or static method to initialize a class having private constructor.

EXAMPLE:

```
using System;
namespace defaultConstructor
{
    public class Counter
    {
        private Counter()    //private constructor declaration
        {
        }
        public static int currentview;
        public static int visitedCount()
        {
            return ++ currentview;
        }
    }
    class viewCountedetails
    {
        static void Main()
        {
            // Counter
            aCounter = new Counter();// Error
            Console.WriteLine("-----Private constructor example-----");
            Console.WriteLine();
            Counter.currentview = 500;
            Counter.visitedCount();
            Console.WriteLine("Now the view count is: {0}", Counter.currentview);
            Console.ReadLine();
        }
    }
}
```

OUTPUT:

```
-----Private constructor example-----
Now the view count is: 501
```



INHERITANCE AND POLYMORPHISM

Q.52.

- Define Inheritance and Polymorphism. [IDOL: May – 2018] | [CBSGS: Apr – 2015]
- Explain how Multiple Inheritance is supported by classes in C#. [IDOL: May – 2018]
- Explain how Multiple Inheritance is achieved using interfaces. [CBSGS: Apr – 2014]
- What is Polymorphism? Explain Runtime Polymorphism in C#. [CBSGS: Nov – 2017]
- Explain with an example the concept of Inheritance. [IDOL: Oct – 2016]

ASKED YEAR ⇒

IDOL: May – 2018 | Oct – 2016

CBSGS: Nov – 2017 | Apr – 2015 | Apr – 2014

SOLUTION

Inheritance:

- ⇒ Inheritance allows us to define a class in terms of another class, which makes it easier to create and maintain an application.
- ⇒ This also provides an opportunity to reuse the code functionality and speeds up implementation time.
- ⇒ When creating a class, instead of writing completely new data members and member functions, the programmer can designate that the new class should inherit the members of an existing class.
- ⇒ This existing class is called the base class, and the new class is referred to as the derived class.

Base And Derived Classes:

- ⇒ A class can be derived from more than one class or interface, which means that it can inherit data and functions from multiple base classes or interfaces.
- ⇒ The syntax used in C# for creating derived classes is as follows –

```
<access-specifier> class <base_class>
{
  ...
}
class <derived_class> : <base_class>
{
  ...
}
```

EXAMPLE:

Consider a base class Shape and its derived class Rectangle –

```
using System;
namespace InheritanceApplication
{
class Shape
{
public void setWidth(int w)
{
width = w;
}
public void setHeight(int h)
{
height = h;
}
protected int width;
protected int height;
}
// Derived class
class Rectangle: Shape
{
public int getArea()
{
return (width * height);
}
}
```




```
class RectangleTester
{
    static void Main(string[] args)
    {
        Rectangle Rect = new Rectangle();
        Rect.setWidth(5);
        Rect.setHeight(7);
        // Print the area of the object.
        Console.WriteLine("Total area: {0}", Rect.getArea());
        Console.ReadKey();
    }
}
```

OUTPUT:

```
Total area: 35
```

Initializing Base Class:

- ⇒ The derived class inherits the base class member variables and member methods.
- ⇒ Therefore the super class object should be created before the subclass is created.
- ⇒ We can give instructions for superclass initialization in the member initialization list.

EXAMPLE:

The following program demonstrates this –

```
using System;
namespace RectangleApplication
{
    class Rectangle
    {
        //member variables
        protected double length;
        protected double width;
        public Rectangle(double l, double w)
        {
            length = l;
            width = w;
        }
        public double GetArea()
        {
            return length * width;
        }
        public void Display()
        {
            Console.WriteLine("Length: {0}", length);
            Console.WriteLine("Width: {0}", width);
            Console.WriteLine("Area: {0}", GetArea());
        }
    }
    //end class Rectangle
    class Tabletop : Rectangle
    {
        private double cost;
        public Tabletop(double l, double w) : base(l, w) { }
        public double GetCost()
        {
            double cost;
            cost = GetArea() * 70;
            return cost;
        }
    }
}
```



```
public void Display()
{
    base.Display();
    Console.WriteLine("Cost: {0}", GetCost());
}
}
class ExecuteRectangle
{
    static void Main(string[] args)
    {
        Tabletop t = new Tabletop(4.5, 7.5);
        t.Display();
        Console.ReadLine();
    }
}
}
```

OUTPUT:

```
Length: 4.5
Width: 7.5
Area: 33.75
Cost: 2362.5
```

Multiple Inheritance In C#:

C# does not support multiple inheritance. However, we can use interfaces to implement multiple inheritance.

EXAMPLE: The following program demonstrates this.

```
using System;
namespace InheritanceApplication
{
    class Shape
    {
        public void setWidth(int w)
        {
            width = w;
        }
        public void setHeight(int h)
        {
            height = h;
        }
        protected int width;
        protected int height;
    }
    // Base class PaintCost
    public interface PaintCost
    {
        int getCost(int area);
    }
    // Derived class
    class Rectangle : Shape, PaintCost
    {
        public int getArea()
        {
            return (width * height);
        }
        public int getCost(int area)
        {
            return area * 70;
        }
    }
}
```



```
class RectangleTester
{
    static void Main(string[] args)
    {
        Rectangle Rect = new Rectangle();
        int area;
        Rect.setWidth(5);
        Rect.setHeight(7);
        area = Rect.getArea();
        // Print the area of the object.
        Console.WriteLine("Total area: {0}", Rect.getArea());
        Console.WriteLine("Total paint cost: ${0}" , Rect.getCost(area));
        Console.ReadKey();
    }
}
```

OUTPUT:

```
Total area: 35
Total paint cost: $2450
```

Polymorphism:

- ⇒ Through Inheritance, a class can be used as more than one type; it can be used as its own type, any base types, or any interface type if it implements interfaces. This is called polymorphism.
- ⇒ In C#, every type is polymorphic. Types can be used as their own type or as object instance, because any type automatically treats Object as a base type.
- ⇒ Polymorphism means having more than one form.
- ⇒ Overloading and overriding are used to implement polymorphism.
- ⇒ Polymorphism is classified into compile time polymorphism or early binding or static binding or static binding and Runtime polymorphism or late binding or dynamic binding.

Compile Time Polymorphism OR Early Binding:

- ⇒ The Polymorphism in which the compiler identifies which Polymorphic form it has to execute at compile time itself is called as compile time Polymorphism or early binding.
- ⇒ Advantage of early binding is execution will be fast. Because everything about the method is known to the compiler during compilation itself and disadvantage is lack of flexibility.
- ⇒ Examples of early binding are overloaded methods, overloaded operators and overridden methods that are called directly by using objects.

Runtime Polymorphism OR Late Binding:

- ⇒ It is also known as Inclusion polymorphism and achieved through the use of virtual functions.
- ⇒ Assume that the class A implements a virtual method M & classes B & C that are derived from A override from A the virtual method M. When B is cast to A, a call to the method M from A is dispatched to B. Similarly when C is cast to A, a call to M is dispatched to C. The decision on exactly which method to call is delayed until runtime & therefore it is also known as runtime polymorphism.
- ⇒ Since the method is linked with a particular class much later after compilation, this process is termed as late binding.

EXAMPLE:

```
using System;
public class A
{
    public virtual void Print()
    {
        System.Console.WriteLine("Virtual Print method from A");
    }
}
```



```
public class B:A
{
public override void Print()
{
System.Console.WriteLine("Override Print method from B");
}
}
class Program
{
static void Main(string[] args)
{
A a=new A();
a.Print();
}
}
```

OUTPUT:

Virtual Print method from A

METHOD OVERLOADING

Q.53.

- Explain with an example the concept of Method Overloading? [CBSE: Nov – 2015 | Oct – 2013] | [IDOL: Oct – 2016 | Apr – 2015 | Oct – 2012]
- What are the steps involved for the selection of a method in Method Overloading? [CBSE: Oct – 2013]

ASKED YEAR →

IDOL: Oct – 2016 | Apr – 2015 | Oct – 2012

CBSE: Nov – 2015 | Oct – 2013

SOLUTION

Method Overloading:

- ⇒ Method Overloading provides the programmer with the capability to create multiple methods with the same name, but each working with different parameters.
- ⇒ In C#, two or more methods within the same class can share the same name, as long as their parameter declarations are different.
- ⇒ When this is the case, the methods are said to be overloaded, and the process is referred to as method overloading.
- ⇒ Method overloading is one of the ways that C# implements polymorphism.
- ⇒ In general, to overload a method, simply declare different versions of it.
- ⇒ The compiler takes care of the rest.
- ⇒ One important restriction: the type and/or number of the parameters of each overloaded method must differ.
- ⇒ It is not sufficient for two methods to differ only in their return types. They must differ in the types or number of their parameters.
- ⇒ Return types do not provide sufficient information in all cases for C# to decide which method to use.
- ⇒ Overloaded methods may differ in their return types, too.
- ⇒ When an overloaded method is called, the version of the method executed is the one whose parameters match the arguments.

EXAMPLE:

```
using System;
class Overload
{
public void ovlDemo()
{
Console.WriteLine("No parameters");
}
// Overload ovlDemo for one integer parameter.
public void ovlDemo(int a)
{
```



```
Console.WriteLine("One parameter: " + a);
}
// Overload ovlDemo for two integer parameters.
public int ovlDemo(int a, int b)
{
    Console.WriteLine("Two parameters: " + a + " " + b);
    return a + b;
}
// Overload ovlDemo for two double parameters.
public double ovlDemo(double a, double b)
{
    Console.WriteLine("Two double parameters: " + a + " " + b);
    return a + b;
}
}
class OverloadDemo
{
    public static void Main()
    {
        Overload ob = new Overload();
        int resI;
        double resD;
        // call all versions of ovlDemo()
        ob.ovlDemo();
        Console.WriteLine();
        ob.ovlDemo(2);
        Console.WriteLine();
        resI = ob.ovlDemo(4, 6);
        Console.WriteLine("Result of ob.ovlDemo(4, 6): " +
            resI);
        Console.WriteLine();
        resD = ob.ovlDemo(1.1, 2.32);
        Console.WriteLine("Result of ob.ovlDemo(1.1, 2.32): " +
            resD);
    }
}
```

OUTPUT:

```
No parameters
One parameter: 2
Two parameters: 4 6
Result of ob.ovlDemo(4, 6): 10
Two double parameters: 1.1 2.32
Result of ob.ovlDemo(1.1, 2.32): 3.42
```

Steps Involved In The Selection Of Method:

- ⇒ Whenever a method is called, the methods signature (the number of arguments and their data type and the sequence) is checked with the list of methods available.
- ⇒ In the following example, Main method first calls Disp(10) method of abc class is called and so on.
- ⇒ Out of the list the matched method is invoked.

EXAMPLE:

```
class abc
{
    public void Disp(int a)
    {
        Console.WriteLine(a);
    }
    public void Disp(int a, int b)
```



```
{
Console.WriteLine("{0}{1}", a,b);
}
}
class program
{
static void Main(string[] args)
{
abc al=newabc();
al.Disp(10);
al.Disp(10,20);
}
}
```

METHOD OVERRIDING

Q.54. What is Method Overriding? Give an example of it.

ASKED YEAR ⇒

IDOL: Apr – 2014

CBSGS: Nov – 2014

SOLUTION

Method Overriding:

- ⇒ Method Overriding in C# is a feature like the virtual function in C++.
- ⇒ Method Overriding is a feature that allows you to invoke functions (that have the same signatures) that belong to different classes in the same hierarchy of inheritance using the base class reference.
- ⇒ C# makes use of two keywords: "virtual" and "overrides" to accomplish Method Overriding.
- ⇒ Let's understand this through following examples.

EXAMPLE:

```
Class BC
{
public virtual void Display()
{
System.Console.WriteLine("BC::Display");
}
}
class DC:BC
{
public override void Display()
{
System.Console.WriteLine("DC::Display");
}
}
class Demo
{
public static void Main()
{
BC b;
b=new BC();
b.Display();
b=new DC();
b.Display();
}
}
```

**OUTPUT:**

```
Hide Copy Code
BC::Display
DC::Display
```

OVERRIDING METHODS VS. METHODS HIDING**Q.55.**

- What is the difference between Overriding Methods and Hiding Methods? [CBSGS: Oct – 2013]
- Explain Method Hiding with example. [CBSGS: Nov – 2015]

ASKED YEAR →

CBSGS: Nov – 2015 | Oct – 2013

SOLUTION**Overriding Methods Vs. Methods Hiding:**

<u>Method Overriding</u>	<u>Method Hiding</u>
An instance method in a subclass with the same signature (name, plus the number and the type of its parameters) and return type as an instance method in the superclass overrides the superclass's method.	A method introduced in a class hides the base class method with the same signature (method name and parameter count, modifiers and types).
In method overriding, a base class reference variable pointing to a child class object, will invoke the overridden method in the child class.	In Hiding Methods, a base class reference variable pointing to a child class object, will invoke the hidden method in the base class.
The base class method should be "virtual" and the "override" keyword is used with the child class method.	The base class method may or may not be virtual but the child class method is in written with "new" keyword.
EXAMPLE: <pre>public class Base { public virtual void SomeMethod() { } } public class Derived: Base { public override void SomeMethod() { } } ... Base b=new Derived(); b.SomeMethod();</pre> <p>will end up calling Derived.SomeMethod of that overrides Base.SomeMethod.</p>	EXAMPLE: <pre>public class Base { public virtual void SomeOtherMethod() { } public new void SomeOtherMethod() { } } ... Base b=new Derived(); Derived d=new Derived(); b.SomeOtherMethod(); d.SomeOtherMethod();</pre> <p>will first call Base.SomeOtherMethod, then Derived.SomeOtherMethod. They're effectively two entirely separate methods which happen to have the same name, rather than the derived method overriding the base method.</p>

**FUNCTION OVERRIDING VS. FUNCTION OVERLOADING****Q.56. Differentiate between Function Overloading and Function Overriding.**

ASKED YEAR →

CBSGS: Nov – 2016 | Apr – 2014

SOLUTION**Function Overriding Vs. Function Overloading:**

<u>Function Overriding</u>	<u>Function Overloading</u>
Methods name and signatures must be same.	Having same method name with different Signatures.
Overriding is the concept of runtime polymorphism.	Overloading is the concept of compile time polymorphism.
When a function of base class is re-defined in the derived class called as Overriding.	Two functions having same name and return type, but with different type and/or number of arguments is called as Overloading.
It needs inheritance.	It doesn't need inheritance.
Method should have same data type.	Method can have different data types.
Method should be public.	Method can be different access specifies.

PROPERTIES/SMART FIELDS**Q.57.**

- Write a short note on Properties. [IDOL: Dec – 2017]
- Name and describe any 5 basic Properties in ASP.NET. [IDOL: Dec – 2017]
- What is a Property? Why are they referred to as Smart Fields? [IDOL: Apr – 2015]

ASKED YEAR →

IDOL: Dec – 2017 | Apr – 2015

SOLUTION**Properties/Smart Fields:**

- ⇒ In C#, properties are nothing but natural extension of data fields.
- ⇒ They are usually known as "smart fields" in C# community.
- ⇒ Data Encapsulation and Hiding are the two fundamental characteristics of any Object-Oriented Programming Language. In C#, Data Encapsulation is possible through either classes or structures.
- ⇒ By using various access modifiers like private, public, protected, internal etc., it is possible to control the accessibility of the class members.
- ⇒ Usually inside a class, we declare a data field as private and will provide a set of public SET and GET methods to access the data fields. This is a good programming practice, since the data fields are not directly accessible outside the class. We must use the set/get methods to access the data fields.

Static Properties:

- ⇒ C# also supports static properties, which belongs to the class rather than to the objects of the class.
- ⇒ All the rules applicable to a static member are applicable to static properties also.
- ⇒ Remember that set/get accessor of static property can access only other static members of the class.
- ⇒ Also static properties are invoking by using the class name.
- ⇒ The following program shows a class with a static property.

```
using System;
class MyClass
{
private static int x;
public static int X
{
get
{
```




```
return x;
}
set
{
x = value;
}
}
}
class MyClient
{
public static void Main()
{
MyClass.X = 10;
int xVal = MyClass.X;
Console.WriteLine(xVal); //Displays 10
}
}
```

Inheritance Properties:

- ⇒ The properties of a Base class can be inherited to a Derived class.
- ⇒ The inheritance of properties is just like inheritance any other member.
- ⇒ The below program is very straightforward.

```
using System;
class Base
{
public int X
{
get
{
Console.Write("Base GET");
return 10;
}
set
{
Console.Write("Base SET");
}
}
}
class Derived : Base
{
}
class MyClient
{
public static void Main()
{
Derived d1 = new Derived();
d1.X = 10;
Console.WriteLine(d1.X); //Displays 'Base SET Base GET 10'
}
}
```

Polymorphism Properties:

- ⇒ A Base class property can be polymorphically overridden in a Derived class.
- ⇒ But remember that the modifiers like virtual, override etc. are using at property level, not at accessor level.

EXAMPLE:

```
using System;
class Base
```



```
{
public virtual int X
{
get
{
Console.WriteLine("Base GET");
return 10;
}
set
{
Console.WriteLine("Base SET");
}
}
}
class Derived : Base
{
public override int X
{
get
{
Console.WriteLine("Derived GET");
return 10;
}
set
{
Console.WriteLine("Derived SET");
}
}
}
class MyClient
{
public static void Main()
{
Base b1 = new Derived();
b1.X = 10;
Console.WriteLine(b1.X); //Displays 'Derived SET Derived GET 10'
}
}
```

Abstract Properties:

- ⇒ A property inside a class can be declared as abstract by using the keyword abstract.
- ⇒ Remember that an abstract property in a class carries no code at all.
- ⇒ The get/set accessors are simply represented with a semicolon.
- ⇒ In the derived class we must implement both set and get assessors.
- ⇒ If the abstract class contains only set accessor, we can implement only set in the derived class.

EXAMPLE: The following program shows an abstract property in action.

```
using System;
abstract class Abstract
{
public abstract int X
{
get;
set;
}
}
class Concrete : Abstract
{
public override int X
{
```



```
get
{
Console.Write(" GET");
return 10;
}
set
{
Console.Write(" SET");
}
}
}
class MyClient
{
public static void Main()
{
Concrete c1 = new Concrete();
c1.X = 10;
Console.WriteLine(c1.X); //Displays 'SET GET 10'
}
}
```



| UNIT |



Collection | Conversions | Delegate | Event

Windows Programming

Topic

COLLECTION | CONVERSIONS | DELEGATE | EVENT

- ⇒ COLLECTION
 - > HASHTABLE COLLECTION
- ⇒ CONVERSIONS / TYPE CASTING
 - > CONVERSIONS
 - > TYPES OF CONVERSIONS
 - *Implicit Conversion*
 - *Explicit Type Conversion*
 - CASTING
 - CONVERTING
 - PARSING
- ⇒ DELEGATE
 - > TYPES OF DELEGATES
 - *Simple Delegate*
 - *Multicast Delegates*
 - > STEPS TO IMPLEMENT DELEGATES
 - *Declaring Delegates*
 - *Instantiating Delegates*
 - *Invocation Process*
- ⇒ EVENT
 - > EVENT VS. DELEGATE

WINDOWS PROGRAMMING

- ⇒ GRAPHICAL USER INTERFACE (GUI)
- ⇒ BUTTON CONTROL
 - > LINK BUTTON
 - > IMAGE BUTTON
- ⇒ TEXTBOX
- ⇒ LABEL CONTROL
- ⇒ CHECKBOX CONTROL
- ⇒ RADIOBUTTON CONTROL
- ⇒ GROUPBOX CONTROL
- ⇒ CHECKBOX VS. RADIOBUTTON CONTROL
- ⇒ LIST CONTROLS
 - > LISTBOX
 - > DROPDOWNLIST
 - > RADIOBUTTONLIST
- ⇒ RADIOBUTTONLIST VS. RADIOBUTTON
- ⇒ LISTBOX CONTROLS VS. DROPDOWNLIST CONTROLS
- ⇒ LISTBOX VS. COMBOBOX CONTROLS
- ⇒ LISTITEMCOLLECTION
- ⇒ COMMONDIALOG BOX
 - > FONTDIALOG
- ⇒ MENU CONTROL
- ⇒ TOOLBAR CONTROL
- ⇒ TOOLSTRIPMENUITEM CONTROL
- ⇒ STATUSSTRIP CONTROL
- ⇒ SDI
- ⇒ MDI
- ⇒ SDI VS. MDI

**COLLECTIONS****Q.1.**

- **What are the different types of Collections in .NET? Explain. [IDOL: Dec/May – 2017 | Apr – 2015]**
- **What are Collections? Write different Collections support by C#. [IDOL: Oct – 2016 | Oct – 2012]**

ASKED YEAR → **IDOL: Dec/May – 2017 | Oct – 2016 | Apr – 2015 | Oct – 2012****SOLUTION****Collection:**

- ⇒ Collections are basically group of records which can be treated as a one logical unit.
- ⇒ .NET Collections are divided in to four important categories as follows:

- 1) *Indexed Based*
- 2) *Key Value Pair*
- 3) *Prioritized Collection*
- 4) *Specialized Collection*

Indexed Based:

It helps you to access the value of row by using the internal generated index number by the collection.

Example:

- **ArrayList:** A simple resizable, index-based collection of objects.

Key Value Pair:

It helps you to access value by the user defined key.

Example:

- **SortedList:** A sorted collection of name/value pairs of objects.
- **Hashtable:** A collection of name/value pairs of objects that allows retrieval by name or index.
- **NameValueCollection:** A collection of name/values pairs of strings that allows retrieval by name or index.

Prioritized Collection:

It helps us to get the element in a particular sequence.

Example:

- **Queue:** A first-in, first-out collection of objects.
- **Stack:** A last-in, first-out collection of objects.

Specialized Collection:

It is very specific collections which are meant for very specific purpose like hybrid dictionary that start as list and become hashtable.

Example:

- **StringCollection:** A simple resizable collection of strings.
- **StringDictionary:** A collection of name/values pairs of strings that allows retrieval by name or index.
- **ListDictionary:** An efficient collection to store small lists of objects.
- **HybridDictionary:** A collection that uses a ListDictionary for storage when the number of items in the collection is small, and then migrates the items to a Hashtable for large collections.



HASH TABLE COLLECTION

Q.2. Explain HashTable Collection with an example.

ASKED YEAR → IDOL: Apr – 2015

SOLUTION

Hashtable Collection:

- ⇒ The Hashtable class represents a collection of key-and-value pairs that are organized based on the hash code of the key.
- ⇒ It uses the key to access the elements in the collection.
- ⇒ A hash table is used when you need to access elements by using key, and you can identify a useful key value.
- ⇒ Each item in the hash table has a key/value pair.
- ⇒ The key is used to access the items in the collection.

Property of Hashtable class:

- **Count:** Gets the number of key-and-value pairs contained in the Hashtable.
- **IsFixedSize:** Gets a value indicating whether the Hashtable has a fixed size.
- **IsReadOnly:** Gets a value indicating whether the Hashtable is read-only.
- **Item:** Gets or sets the value associated with the specified key.
- **Keys:** Gets an ICollection containing the keys in the Hashtable.
- **Values:** Gets an ICollection containing the values in the Hashtable.

Method of Hashtable class:

- **public virtual void Add(object key, object value);** – Adds an element with the specified key and value into the Hashtable.
- **public virtual void Clear();** – Removes all elements from the Hashtable.
- **public virtual bool ContainsKey(object key);** – Determines whether the Hashtable contains a specific key.
- **public virtual bool ContainsValue(object value);** – Determines whether the Hashtable contains a specific value.
- **public virtual void Remove(object key);** – Removes the element with the specified key from the Hashtable.

Example:

```
using System;
using System.Collections;
namespace CollectionsApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            Hashtable ht = new Hashtable();
            ht.Add("001", "Zara Ali");
            ht.Add("002", "Abida Rehman");
            ht.Add("003", "Joe Holzner");
            ht.Add("004", "Mausam Benazir Nur");
            ht.Add("005", "M. Amlan");
            ht.Add("006", "M. Arif");
            ht.Add("007", "Ritesh Saikia");
            if (ht.ContainsValue("Nuha Ali"))
            {
                Console.WriteLine("This student name is already in the list");
            }
            else
            {
                ht.Add("008", "Nuha Ali");
            }
            // Get a collection of the keys.
        }
    }
}
```



```

ICollection key = ht.Keys;
foreach (string k in key) {
Console.WriteLine(k + " : " + ht[k]);
}
Console.ReadKey();
}
}
}

```

Output:

```

001: Zara Ali
002: Abida Rehman
003: Joe Holzner
004: Mausam Benazir Nur
005: M. Amlan
006: M. Arif
007: Ritesh Saikia
008: Nuha Ali

```

CONVERSIONS / TYPE CASTING**Q.3.**

- **What is Type Casting?** [CBSGS: Apr – 2015]
- **Explain Implicit and Explicit Type Conversion with examples.** [IDOL: Apr – 2013]
- **What is Type Conversion in C#? How is it carried out in C#?** [IDOL: Oct – 2016]
- **Write a short note on Conversions.** [IDOL: Oct – 2016]

ASKED YEAR → [IDOL: Oct – 2016 | Apr – 2014 | Apr – 2013]

CBSGS: Apr – 2015

SOLUTION**Conversions:**

- ⇒ It is converting one type of data to another type. It is also known as Type Casting.
- ⇒ Type Conversion takes two forms:
 - (i) *Implicit Type Conversion*
 - (ii) *Explicit Type Conversion*

Types Of Conversions:Implicit Conversion:

- ⇒ This Conversion is performed without any loss of data.
- ⇒ For Example, Short type can be implicitly converted to int because the short range is a subset of int range.

```

short b=75;
int a=b;

```
- ⇒ This Conversion occurs automatically.
- ⇒ An Implicit Conversion is also known as an Automatic Type Conversion.
- ⇒ The process of assigning a smaller type to a larger one is known as Implicit Conversion or Widening or promotion.

Example:

```

byte a1=75;
short a2=a1;
int a3=a2;
long a4=a3;
float a5=a4;
decimal a6=a4;

```

Explicit Type Conversion:



- ⇒ There are many Conversion that cannot be implicitly made between types.
- ⇒ If one attempts such Conversions the compiler gives an error message.
- ⇒ The process of assigning a larger type to a smaller one is known as explicit Conversion or Narrowing.
- ⇒ The following Conversions cannot be made implicitly.

```
int to short
int to uint
uint to int
float to int
decimal to any numeric type
any numeric type to char
```

- ⇒ However, such Conversions can be carried out explicitly using casting, converting and parsing.

Casting:

```
type variable1=(type)variable2;
int m=50;
byte n=(byte)m;
casting into a smaller type results in the loss of data.
```

Converting:

C# provides the methods to convert between Numeric Values and Strings

```
int m=100;
string s=m.ToString();
string s=Console.ReadLine();
int s1=Convert.ToInt32(s),
```

Parsing:

It returns the Numeric representation of String Data.

```
string s="100";
int s1=int.parse(s);
int m=s1+500;
```

Example: The following program demonstrates some type conversions that require casts. It also shows some situations in which the casts cause data to be lost.

```
using System;
class CastDemo
{
    static void Main()
    {
        double x, y;
        byte b;
        int i;
        char ch;
        uint u;
        short s;
        long l;
        x = 10.0;
        y = 3.0;
        // Cast double to int, fractional component lost.
        i = (int) (x / y);
        Console.WriteLine("Integer outcome of x / y: " + i);
        Console.WriteLine();
        // Cast an int into a byte, no data lost.
        i = 255;
        b = (byte) i;
        Console.WriteLine("b after assigning 255: " + b + " -- no data lost.");
        // Cast an int into a byte, data lost.
        i = 257;
```




```
b = (byte) i;
Console.WriteLine("b after assigning 257: " + b + " -- data lost.");
Console.WriteLine();
// Cast a uint into a short, no data lost.
u = 32000;
s = (short) u;
Console.WriteLine("s after assigning 32000: " + s + " -- no data lost.");
// Cast a uint into a short, data lost.
u = 64000;
s = (short) u;
Console.WriteLine("s after assigning 64000: " + s + " -- data lost.");
Console.WriteLine();
// Cast a long into a uint, no data lost.
l = 64000;
u = (uint) l;
Console.WriteLine("u after assigning 64000: " + u + " -- no data lost.");
// Cast a long into a uint, data lost.
l = -12;
u = (uint) l;
Console.WriteLine("u after assigning -12: " + u + " -- data lost.");
Console.WriteLine();
// Cast an int into a char.
b = 88; // ASCII code for X
ch = (char) b;
Console.WriteLine("ch after assigning 88: " + ch);
}
}
```

Output:

```
Integer outcome of x / y: 3
b after assigning 255: 255 -- no data lost.
b after assigning 257: 1 -- data lost.
s after assigning 32000: 32000 -- no data lost.
s after assigning 64000: -1536 -- data lost.
u after assigning 64000: 64000 -- no data lost.
u after assigning -12: 4294967284 -- data lost.
ch after assigning 88: X
```

**DELEGATE****Q.4.**

- **What is Delegate? [IDOL: Dec/May – 2017] | [CBSGS: Nov – 2016 | Apr – 2015]**
- **State its Declaration, Instantiation and Invocation Process. [IDOL: Dec – 2017]**
- **Explain the steps to implement Delegates in C# .NET. [IDOL: May – 2017 | May – 2016 | Oct – 2016] | [CBSGS: Nov – 2016 | Apr – 2015]**
- **What are the two types of delegates? Explain each with an example. [IDOL: Oct – 2012]**
- **Delegates in C# are used for Event Handling. [IDOL: Apr – 2014] | [CBSGS: Oct – 2013]**

ASKED YEAR ⇒

[IDOL: Dec/May – 2017 | Oct/May – 2016 | Apr – 2014 | Oct – 2012]

[CBSGS: Nov – 2016 | Apr – 2015 | Oct – 2013]

SOLUTION**Delegate:**

- ⇒ A Delegate is a type that enables you to store references to functions.
- ⇒ A Delegate is a type which holds the method(s) reference in an object. It is also referred to as a type safe function pointer.
- ⇒ Delegates are used to pass methods as arguments to other methods.
- ⇒ Any method from any accessible class that matches the delegate's signature, which consists of the return type and parameters, can be assigned to the delegate. The method can be either static or an instance method.

Types Of Delegates:

There are two types of Delegates as follows:

- 1) *Simple Delegate*
- 2) *Multicast Delegates*

Simple Delegate:

One delegate can invoke only one method whose reference is encapsulated into the delegate.

Example:

```
using system;
{
public delegate void CalculateDelegate(int a, int b);
class program
{
public static void Add(int a, int b)
{
int c=a+b;
Console.WriteLine(c);
}
}
class Program1
{
public static void Main(string[] args)
{
calculateDelegate cd=new calculateDelegate(program.Add);
cd(10,20);
console.ReadKey();
}
}
}
```

Multicast Delegates:

- ⇒ It is possible for certain delegates to hold and invoke multiple methods. Such delegates are called as multicast delegates.
- ⇒ They are also known as combinable delegates.
- ⇒ They must satisfy the following conditions:
 - The return type of the delegate must be void.



- None of the parameters of the delegate type can be declared as output parameters, using out keyword.

Example:

```
namespace Multicast_Delegates
{
    Delegate void MDelegate();
    class MD
    {
        public static void Display()
        {
            Console.WriteLine("New Delhi");
        }
        public static void Print()
        {
            Console.WriteLine("New York");
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            MDelegate m1=newMDelegate(MD.Display);
            MDelegate m2=newMDelegate(MD.Print);
            MDelegate m3=m1+m2;
            MDelegate m4=m2+m1;
            MDelegate m5=m3-m2;
            //invoking delegates
            m3 ();
            m4 ();
            m5 ();
            Console.ReadKey();
        }
    }
}
```

Steps To Implement Delegates In ASP.Net C#:**STEP 1: Declaring Delegates:**

- ⇒ Delegate declaration determines the methods that can be referenced by the delegate.
- ⇒ A delegate can refer to a method, which has the same signature as that of the delegate.
- ⇒ For example, consider a delegate –

```
public delegate int MyDelegate (string s);
```

- ⇒ The preceding delegate can be used to reference any method that has a single string parameter and returns an int type variable.
- ⇒ Syntax for delegate declaration is –

```
delegate <return type> <delegate-name> <parameter list>
```

STEP 2: Instantiating Delegates:

- ⇒ Once a delegate type is declared, a delegate object must be created with the new keyword and be associated with a particular method.
- ⇒ When creating a delegate, the argument passed to the new expression is written similar to a method call, but without the arguments to the method.
- ⇒ For example –

```
public delegate void printString(string s);
...
printString ps1 = new printString(WriteToScreen);
printString ps2 = new printString(WriteToFile);
```

**STEP 3: Invocation Process:**

- ⇒ When a method is referenced or associated with an instance of the delegate then this is the last step in which we invoke that method by calling the associated instance of the delegate and assign values to its parameters, if the referenced or associated method takes parameters.
- ⇒ The following is the general syntax to invoke a method using the referenced or associated delegate instance.
`ReturnType instanceName (ParametersList);`
- ⇒ In the preceding declaration the `ReturnType` represents the return type of the referenced or associated method, the `instanceName` represents the name of the delegate instance and the `ParametersList` represents the number of parameters of the referenced or associated method.

Example: Following example demonstrates declaration, instantiation, and use of a delegate that can be used to reference methods that take an integer parameter and returns an integer value.

```
using System;
delegate int NumberChanger(int n);
namespace DelegateAppl
{
    class TestDelegate
    {
        static int num = 10;
        public static int AddNum(int p)
        {
            num += p;
            return num;
        }
        public static int MultNum(int q)
        {
            num *= q;
            return num;
        }
        public static int getNum()
        {
            return num;
        }
        static void Main(string[] args)
        {
            //create delegate instances
            NumberChanger nc1 = new NumberChanger (AddNum);
            NumberChanger nc2 = new NumberChanger (MultNum);
            //calling the methods using the delegate objects
            nc1(25);
            Console.WriteLine("Value of Num: {0}", getNum());
            nc2(5);
            Console.WriteLine("Value of Num: {0}", getNum());
            Console.ReadKey();
        }
    }
}
```

Output:

```
Value of Num: 35
Value of Num: 175
```

Delegates In C# Are Used For Event Handling:

- ⇒ An event is a mechanism via which a class can notify its clients when something happens.
- ⇒ For example, when a button is clicked, a button-click-event notification is sent to the window hosting the button.
- ⇒ Events are declared using delegates.
- ⇒ A button is a class, when clicked, the click event fires.



- ⇒ A Timer is a class, every millisecond a tick event fires.
- ⇒ Following class has an event: SomeEvent.
- ⇒ Events are variables of type delegates. If an event is to be declared, just declare a variable of type delegate and put event keyword before the declaration, like this:

```
public event MyEventHandler SomeEvent
```

- ⇒ Where MyEventHandler is the delegate name.
- ⇒ The syntax for initiating an event handler for some event:

```
object of the class containing the event.event_name +=new delegate_name (method)
MyEvent evt=new MyEvent();
//Add handler() to the event list.
evt.SomeEvent+=new MyEventHandler (abc);
```

- ⇒ And Notice the use of += instead of simply=.
- ⇒ It's because delegates are special objects that can hold references to more than one object (here, reference to more than one function).
- ⇒ For example, if you had another function named handler1 with the same signature as handler, both of the functions could be referenced in the following example way:

Example:

```
namespace delegate_and_event
{
    // Declare a delegate for an event
    delegate void MyEventHandler();
    // Declare an event class.
    class MyEvent
    {
        public event MyEventHandler SomeEvent;
        // This is called to fire the event.
        public void OnSomeEvent()
        {
            if(SomeEvent !=null)
                SomeEvent();
        }
    }
    public class EventDemo
    {
        // An event handler.
        static void handler()
        {
            Console.WriteLine("Event occurred");
        }
        public static void Main()
        {
            MyEvent evt=new MyEvent();
            // Add handler() to the event list.
            evt.SomeEvent += new MyEventHandler(handler);
            // Fire the event.
            evt.OnSomeEvent();
        }
    }
}
```

EVENT VS. DELEGATE**Q.5. What is an Event? What are the differences between Delegate and Event?**

ASKED YEAR →

CBSGS: Apr – 2017

SOLUTIONEvent:

- ⇒ Events enable a class or object to notify other classes or objects when something of in latest access.
- ⇒ The class that sends (or raises) the event is called the publisher and the classes that receive (or handle) the event are called subscribers.

Event Vs. Delegate:

<u>Event</u>	<u>Delegate</u>
It is Data Member of a type (class / structure).	It is a datatype (reference type) that holds references of methods with some signature also called as function pointer.
It is declared inside a type (class / structure).	It may or may not be declared inside a class.
It is used to generate notifications which are then passed to methods through delegates.	It is used as the return type of an event & used in passing messages from event to methods.

GRAPHICAL USER INTERFACE (GUI)**Q.6. Write a short note on GUI. State its advantages.**

ASKED YEAR →

IDOL: May – 2016

SOLUTIONGraphical User Interface (GUI):

- ⇒ In C#, the most rapid and convenient way to create your user interface is to do so visually, using the Windows Forms Designer and Toolbox.
- ⇒ Windows Forms controls are reusable components that encapsulate user interface functionality and are used in client side Windows based applications.
- ⇒ A control is a component on a form used to display information or accept user input. The Control class provides the base functionality for all controls that are displayed on a Form.

Advantages Of GUIs:

- ⇒ A major advantage of GUIs is that they make computer operation more intuitive, and thus easier to learn and use.
- ⇒ GUIs generally provide users with immediate, visual feedback about the effect of each action.
- ⇒ GUI allows multiple programs and/or instances to be displayed simultaneously.
- ⇒ Users do not need to know any programming languages.



BUTTON CONTROL

Q.7.

- What is the difference between Button, LinkButton and ImageButton? Explain any three Common Button Attributes. [CBSGS: Nov – 2014]
- Explain LinkButton Controls and ImageButton Controls. [CBSGS: Nov – 2017]
- What is the difference between Button and LinkButton Web Server Controls? [IDOL: Apr – 2015]

ASKED YEAR ⇒

IDOL: Apr – 2015

CBSGS: Nov – 2014 | Nov – 2017

SOLUTION

Button Control:

- ⇒ It displays a push button control on the Web page.
- ⇒ The Push button can be a command button or a submit button.
- ⇒ A command button has a command name and permits us to create multiple buttons on a page.
- ⇒ When a command button is clicked, we can also implement an event handler to control the actions to be performed.
- ⇒ A Submit button does not have a command name and when the button is clicked, it posts the web page back to the server.
- ⇒ When a submit button is clicked, we can also implement an event handler to control the actions to be performed.

Properties of a Button Control:

- **CausesValidation:** This indicates whether validation will be performed when this button will be clicked. Here, the Value can be set as true or false.
- **PostBackUrl:** This specifies the URL of the page to post to from the current page when a button is clicked.
- **ValidationGroup:** It enables you to specify which validators on the page are called when the button is clicked. If no validation groups are established, a button click calls all of the validators that are on the page.
- **OnClickClientClick:** Attach a client side (JavaScript) event that will fire when this button will be clicked.
- **OnClick:** Raises the Click event of the Button control.

Attributes of Button Control:

- **Text:** (Button and LinkButton controls only) The text displayed by the button. For a LinkButton control, the text can be coded as content between the start and end tags or as the value of the Text attribute.
- **ImageUrl:** (ImageButton control only) The image to be displayed for the button.
- **AlternateText:** (ImageButton control only) The text to be displayed if the browser can't display the image.
- **CausesValidation:** Determines whether page validation occurs when you click button. The default is True.
- **CommandName:** A string value that's passed to the Command event when a user clicks the button.
- **CommandArgument:** A string value that's passed to the Command event when a user clicks the button.
- **PostBackUrl:** The URL of the page that should be requested when the user clicks the button.

LinkButton Control:

- ⇒ The LinkButton control is used to create a hyperlink-style button on the Web page.
- ⇒ This control looks like a Hyperlink control but almost has the functionality of the Button control.
- ⇒ Despite being a hyperlink, you can't specify the target URL.
- ⇒ There is no UserSubmitBehavior property like the Button control with this control.

Properties of LinkButton Control:

- **CausesValidation:** This indicates whether validation will be performed when this button will be clicked. Here, the Value can be set as true or false.
- **PostBackUrl:** This Specifies the URL of the page to post to from the current page when the LinkButton control is clicked.
- **ValidationGroup:** The group of controls for which the LinkButton control causes validation when it posts back to the server.
- **OnClick:** Attach a server side method that will fire when this button will be clicked.
- **OnClickClientClick:** Attach a client side (JavaScript) method that will fire when this button will be clicked.



ImageButton Control:

- ⇒ It's like an ASP Button control, the only difference is; we have the ability to place our own image as a button.
- ⇒ We use an image Button when we want our button to look different than the plain rectangular button.
- ⇒ Any image can be a button.

Properties of an ImageButton Control:

- **ImageUrl**: Gets or Sets the location of the image to display as button control.
- **CausesValidation**: This indicates whether validation will be performed when this button will be clicked. Here, the Value can be set as true or false.
- **PostBackUrl**: This Specifies the URL of the page to post to from the current page when the LinkButton control is clicked.
- **OnClick**: Attach a server side method that will fire when this button will be clicked.
- **OnClientClick**: Attach a client side (JavaScript) method that will fire when this button will be clicked.

Example: The following .aspx Code example demonstrates how to design Button, LinkButton and ImageButton controls.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="buttons.aspx.cs" Inherits="buttons" %>
<!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></title>
</head>
<body>
<form id="form1" runat="server">
<asp:Button ID="Button1" runat="server" onclick="Button1_Click" style="z-index: 1; left: 23px;
top: 56px; position: absolute" Text="click!" />
<asp:LinkButton ID="LinkButton1" runat="server" EnableTheming="True"
onclick="LinkButton1_Click" style="z-index: 1; left: 24px; top: 194px; position: absolute"
ToolTip="Link to another page">LinkButton</asp:LinkButton>
<p style="height: 669px">
<asp:ImageButton ID="ImageButton1" runat="server" ImageUrl="~/images/smiley-guy.jpg"
onclick="ImageButton1_Click" style="z-index: 1; left: 16px; top: 294px; position: absolute;
height: 80px; width: 88px; right: 1177px; bottom: 421px" />
<asp:TextBox ID="TextBox1" runat="server" style="z-index: 1; left: 20px; top: 383px; position:
absolute"></asp:TextBox>
</p>
<p>&nbsp;</p>
</form>
</body>
</html>
```

.aspx.cs Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
public partial class buttons : System.Web.UI.Page
{
protected void Page_Load(object sender, EventArgs e)
{
}
protected void Button1_Click(object sender, EventArgs e)
{
Button1.Text = "You clicked the button!";
}
protected void LinkButton1_Click(object sender, EventArgs e)
{
}
```




```
Response.Redirect("default.aspx");//move to the next link i.e default.aspx
}
protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
{
    TextBox1.Text="keep smiling";
}
}
```

TEXTBOX CONTROL

Q.8.

- Explain the **TextBox** Control with any **Five Properties**. [IDOL: May – 2017 | Oct/May – 2016 | Apr – 2015 | Apr – 2014]
 ➤ List and explain any four **TextBox Attributes**. [CBSGS: Oct – 2013]

ASKED YEAR ⇒

IDOL: May – 2017 | Oct/May – 2016 | Apr – 2015 | Apr – 2014

CBSGS: Oct – 2013

SOLUTION

TextBox Control:

- ⇒ **TextBox** Control are typically used to accept input from the user.
- ⇒ **TextBox** Control can accept one or more lines of text depending upon the settings of the **TextMode** attribute.

Basic Syntax of TextBox Control:

```
<asp:TextBox ID="txtstate" runat="server" ></asp:TextBox>
```

Property of TextBox Control:

- **TextMode**: It specifies the type of text box. **SingleLine** creates a standard text box, **MultiLine** creates a **TextBox** that accepts more than one line of text and the **Password** causes the characters that are entered to be masked. The default is **SingleLine**.
- **Text**: The text content of the **TextBox**.
- **MaxLength**: The maximum number of characters that can be entered into the **TextBox**.
- **Wrap**: It determines whether or not text wraps automatically for multi-line **TextBox**; default is **true**.
- **ReadOnly**: It determines whether the user can change the text in the box; default is **false** (i.e. the user cannot change the text).
- **Columns**: The width of the **TextBox** in characters. The actual width is determined based on the font that is used for the text entry.
- **Rows**: The height of a multi-line text box in lines. The default value is 0, means a single line text box.

LABEL CONTROL

Q.9.**Explain Label Control.**

ASKED YEAR ⇒

IDOL: Apr – 2014

SOLUTION

Label Controls:

- ⇒ **Label** Controls provide an easy way to display text which can be changed from one execution of a page to the next.
- ⇒ If we want to display text that does not change, we use the literal text.

Property of Label Control:

- **TextMode**: It specifies the type of text box. **SingleLine** creates a standard text box, **MultiLine** creates a text box that accepts more than one line of text and the **Password** causes the characters that are entered to be masked. The default is **SingleLine**.
- **Text**: The text content of the **TextBox**.
- **MaxLength**: The maximum number of characters that can be entered into the **TextBox**.



- **Wrap**: It determines whether or not text wraps automatically for multi-line text box; default is true.
- **ReadOnly**: It determines whether the user can change the text in the box; default is false (i.e. the user cannot change the text).
- **Columns**: The width of the `TextBox` in characters. The actual width is determined based on the font that is used for the text entry.
- **Rows**: The height of a multi-line `TextBox` in lines. The default value is 0, means a single line `TextBox`.

CHECKBOX CONTROL

Q.10.

- **Explain CheckBox Controls. [IDOL: Dec – 2017 | Oct – 2016 | Apr – 2013] | [CBSGS: Apr – 2015]**
- **Explain the following properties/events of CheckBox control: [CBSGS: Apr – 2017]**
 - (1) ID
 - (2) AccessKey
 - (3) Text
 - (4) Checked
 - (5) CheckedChanged ()

ASKED YEAR →

IDOL: Dec – 2017 | Oct – 2016 | Apr – 2013

CBSGS: Apr – 2017 | Apr – 2015

SOLUTION

CheckBox Control:

- ⇒ A `CheckBox` displays a single option that the user can either check or uncheck and radio buttons present a group of options from which the user can select just one option.
- ⇒ In it, user can select one or more than one options at a time.
- ⇒ Its main property is "value" and it is 1, if `CheckBox` is checked otherwise false.

Syntax:

```
<asp:CheckBox ID= "chkoption" runat= "Server">  
</asp:CheckBox>
```

CheckBox Properties:

- **AutoPostBack**: Specifies whether the form should be posted immediately after the `Checked` property has changed or not. Default is false.
- **AccessKey**: It is used to set keyboard shortcut for the control.
- **CausesValidation**: Specifies if a page is validated when a `Button` control is clicked.
- **Checked**: Specifies whether the `CheckBox` is checked or not.
- **Id**: A unique id for the control.
- **Runat**: Specifies that the control is a server control. Must be set to "server".
- **Text**: The text next to the `CheckBox`.

CheckBox Event:

- **OnCheckedChanged**: The name of the function to be executed when the checked property has changed.

Example:

```
using System;  
using System.Text;  
using System.Windows.Forms;  
namespace WindowsFormsApplication1  
{  
    public partial class Form4 : Form  
    {  
        public Form4 ()
```



```

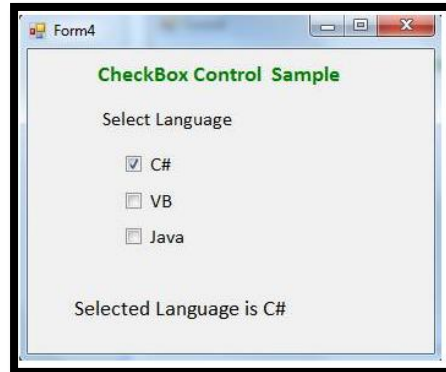
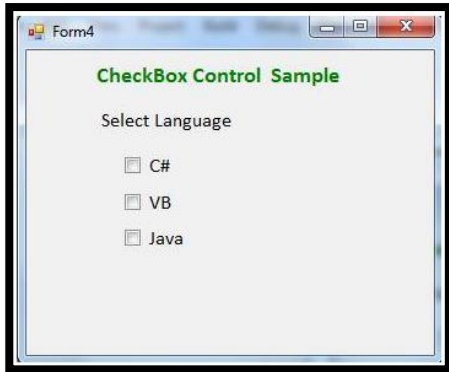
{
InitializeComponent();
}
private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
//display checkbox value when checkbox1 is checked
label1.Text = "Selected Language is " + checkBox1.Text;
}
private void checkBox2_CheckedChanged(object sender, EventArgs e)
{
//display checkbox value when checkbox2 is checked
label1.Text = "Selected Language is " + checkBox2.Text;
}
private void checkBox3_CheckedChanged(object sender, EventArgs e)
{
//display checkbox value when checkbox3 is checked
label1.Text = "Selected Language is " + checkBox3.Text;
}
}
}

```

Output:

It produces the following output to the browser.

When you select the given option then CheckedChanged event will fire and selected value will show in Label.

**RADIOBUTTON CONTROL****Q.11. Explain RadioButton Control.**

ASKED YEAR ⇒ **IDOL:** Dec – 2017 | Apr – 2013

CBSGS: Apr – 2015

SOLUTION**RadioButton Control:**

- ⇒ RadioButton Control is used, when we want the user to select only one option from the available choices.
- ⇒ RadioButton are used to select single option from multiple options.
- ⇒ For example, the gender of a person. A person can be "Male" or "Female". He/she cannot be both.
- ⇒ So, if the user has first selected "Male" and if he/she tries to select "Female", the initial "Male" selection he made should automatically get de-selected.

Syntax:

```

<asp:RadioButton ID= "rdboption" runat= "Server">
</asp:RadioButton>

```



Property of RadioButton Control:

- **AutoPostBack**: A Boolean value that specifies whether the form should be posted immediately after the Checked property has changed or not. Default is false.
- **Checked**: A Boolean value that specifies whether the RadioButton is checked or not.
- **Id**: A unique id for the control.
- **GroupName**: The name of the group to which this RadioButton belongs.
- **OnCheckedChanged**: The name of the function to be executed when the checked property has changed.
- **Runat**: It specifies that the Control is a server control. Must be set to "server".
- **Text**: The text next to the RadioButton.
- **TextAlign**: On which side of the RadioButton the text should appear (right or left).

Example:


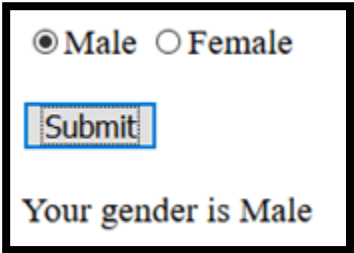
In this example, we are creating two radio buttons and putting in a group named gender.

```
// WebControls.aspx
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebControls.aspx.cs"
Inherits="WebFormsControlls.WebControls" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title>RadioButton Control</title>
</head>
<body>
<form id="form1" runat="server">
<div>
<asp:RadioButton ID="RadioButton1" runat="server" Text="Male" GroupName="gender"/>
<asp:RadioButton ID="RadioButton2" runat="server" Text="Female" GroupName="gender"/>
</div>
<p>
<asp:Button ID="Button1" runat="server" Text="Submit" OnClick="Button1_Click" style="width:
61px"/>
</p>
</form>
<asp:Label runat="server" id="genderId"></asp:Label>
</body>
</html>
```

Code Behind:

```
// WebControls.aspx.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace WebFormsControlls
{
public partial class WebControls : System.Web.UI.Page
{
protected void Button1_Click(object sender, EventArgs e)
{
genderId.Text = "";
if (RadioButton1.Checked)
{
genderId.Text = "Your gender is "+RadioButton1.Text;
}
else genderId.Text = "Your gender is "+RadioButton2.Text;
}
}
}
```

**Output:**

It produces the following output to the browser.	It responds back to the client when user select the gender.
	

GROUPBOX CONTROL**Q.12. Explain GroupBox Control.**

ASKED YEAR ⇒

IDOL: Dec – 2017

SOLUTIONGroupBox Control:

- ⇒ A GroupBox Control is a container control that is used to place Windows Forms child controls in a group.
- ⇒ The purpose of a GroupBox is to define user interfaces where we can categories related controls in a group.

How to use GroupBox Control:

- ⇒ Drag and drop GroupBox control from toolbox on the window Form.
- ⇒ When you drag and drop GroupBox, it has a border by default, and its caption is set to the name of the control.
- ⇒ We can change its caption.
- ⇒ We can insert other control in side GroupBox control.

Properties of GroupBox:

- **AutoSize:** Gets or sets a value that indicates whether the GroupBox resizes based on its contents.
- **BackColor:** Gets or sets the background color for the control.
- **ForeColor:** ForeColor of all control in side GroupBox can be change at a time.

Example:

```
private void GroupBox_Load(object sender, EventArgs e)
{
    //Change
    Fore color of all control in side GroupBox
    groupBox1.ForeColor= Color.RoyalBlue;
}
```

Fore color of all control in side GroupBox will be change when application run.

Output:

CHECKBOX CONTROL VS. RADIOBUTTON CONTROL

Q.13.

- What is the difference between CheckBox and RadioButton Control? What are the common attributes associated with these controls? [CBSGS: Oct – 2013]
- What is the difference between RadioButton and CheckButton Controls? [IDOL: Apr – 2015]

ASKED YEAR ⇒

IDOL: Apr – 2015

CBSGS: Oct – 2013

SOLUTIONCheckBox Control Vs. RadioButton Control:

<u>CheckBox Control</u>	<u>RadioButton Control</u>
In it, user can select one or more than one options at a time.	In it, user can select only one option at a time.
Its main property is "value" and it is 1, if CheckBox is checked otherwise false.	Its main property is "checked" and it is true, if the control is checked otherwise false.
Example: Select Hobbies <input checked="" type="checkbox"/> Playing Cricket <input type="checkbox"/> Studying <input type="checkbox"/> Swimming <input checked="" type="checkbox"/> Reading Here, we can select multiple option.	Example: Select Gender <input checked="" type="radio"/> Male <input type="radio"/> Female Here, we can select only one option.

Common Attributes associated with these controls:

- **Text:** To display the text for the control.
- **Checked:** To check whether the radio button or checkbox is checked (selected) or not.
- **TextAlign:** Gets or sets the alignment of the text label associated with the control.
- **enableViewState:** Gets or sets the value indicating whether the server control persists its ViewState.
- **ValidationGroup:** Gets or sets the group of controls for which these controls causes validation when they postback to the server.

Q.14.

When is CheckedChanged Event of a CheckBox fired?**Describe the following properties:**

- (a) *GroupName property of a RadioButton*
- (b) *Text property of a Label*
- (c) *TextMode property of a TextBox*
- (d) *Checked property of a RadioButton*

ASKED YEAR ⇒

CBSGS: Nov – 2015

SOLUTIONCheckedChanged Event of CheckBox:

For a CheckBox the checked changed event is raised whenever the checked status at the control changes.

GroupName property of a RadioButton:

To create a group of radio buttons simply specify the same name for the `GroupName` attribute of each radio button in the group. If we want to create two or more groups of radio buttons on a single form just use a different `GroupName` for each group.

Text property of a Label:

It specifies the text content at the Label control.

TextMode property of a TextBox:

It specifies the type of `TextBox`. Single line creates a standard text multiline creates a `TextBox` that accepts more than one line of text and password causes the characters that are entered to be masked. The default is single line.



Checked property of a RadioButton:

It indicates whether `RadioButton` is selected or not. Default is false.

LISTBOX CONTROLS

Q.15. Explain properties of `Listbox` Controls.

ASKED YEAR → IDOL: Dec/ May – 2017 | Oct – 2012

CBSGS: Nov – 2017 | Nov – 2016 | Oct – 2013 | Apr – 2014

SOLUTION

Properties of `Listbox` Controls:

`AutoPostBack`:

- ⇒ Gets or sets a value indicating whether a postback to the server automatically occurs when the user changes the list selection (Inherited from `ListControl`).
- ⇒ If we set `AutoPostBack` property of the `Listbox` to `True`, then form containing `Listbox` is submitted automatically whenever a new item is selected.

`Items`:

- ⇒ This property enables you to obtain a reference to the list of items that are currently stored in the `Listbox`.
- ⇒ With this reference, you can add items, remove items, and obtain a count of the items in the collection.

Methods:

```
ListBox1.Items.Clear();  
ListBox1.Items.Add(new ListItem(textBox1.Text));
```

`Sorted`:

- ⇒ The `Sorted` property set to `true`, the `Listbox` items are sorted.
- ⇒ The following code snippet sorts the `Listbox` items.

```
listBox1.Sorted = true;
```

`Selected Mode`:

- ⇒ `SelectionMode` property defines how items are selected in a `Listbox`.
- ⇒ The `SelectionMode` value can be one of the following four `SelectionMode` enumeration values.
 - **None** – No item can be selected.
 - **One** – Only one item can be selected.
 - **MultiSimple** – Multiple items can be selected.
 - **MultiExtended** – Multiple items can be selected, and the user can use the SHIFT, CTRL, and arrow keys to make selections.
- ⇒ To select an item in a `Listbox`, we can use the `SetSelected` method that takes item index and a true or false value where true value represent the item to be selected.
- ⇒ The following code snippet make a `Listbox` multiple selection and selects second and third item in the list.

```
listBox1.SelectionMode = SelectionMode.MultiSimple;  
listBox1.SetSelected(1, true);  
listBox1.SetSelected(2, true);
```

`MultiColumn`:

- ⇒ A multicolumn `Listbox` places items into as many columns as are needed to make vertical scrolling unnecessary.
- ⇒ The user can use the keyboard to navigate to columns that are not currently visible.
- ⇒ Set the `HorizontalScrollbar` property to `true` to display a horizontal scroll bar that enables the user to scroll to columns that are not currently shown in the visible region of the `Listbox`.
- ⇒ The value of the `ColumnWidth` property determines the width of each column.
- ⇒ The following code snippet makes a `Listbox` with multiple columns:



```
listBox1.MultiColumn = true;
```

SelectedItem:

⇒ Gets or sets the currently selected item in the `ListBox`.

⇒ We can get text associated with currently selected item by using `Items` property.

```
string selectedItem = listBox1.Items[listBox1.SelectedIndex].ToString();
```

SelectedIndex:

⇒ Gets or sets the zero-based index of the currently selected item in a `ListBox`.

⇒ The following code snippet sets the Index to 1

```
listBox1.SelectedIndex = 1;
```

DROPDOWNLIST CONTROL

Q.16. Explain the properties of `DropDownList` Control.

ASKED YEAR → IDOL: Apr – 2015

CBSGS: Oct – 2013

SOLUTION

Properties of `DropDownList` Control:

- **SelectedValue**: Get the value of the selected item from the dropdown box.
- **SelectedIndex**: Gets or Sets the index of the selected item in the dropdown box.
- **SelectedItem**: Gets the selected item from the list.
- **Items**: Gets the collection of items from the dropdown box.
- **DataTextField**: Name of the data source field to supply the text of the items. (No need to set when you are adding items directly into .aspx page.)
- **DataValueField**: Name of the data source field to supply the value of the items. (No need to set when you are adding items directly into .aspx page.)
- **DataSourceID**: ID of the datasource component to provide data. (Only used when you have any `DataSource` component on the page, like `SqlDataSource`, `AccessDataSource` etc.)
- **DataSource**: The datasource that populates the items in the dropdown box. (Generally used when you are dynamically generating the items from Database.)
- **AutoPostBack**: true or false. If true, the form is automatically posted back to the server when user changes the dropdown list selection. It will also fire `OnSelectedIndexChanged` method.
- **AppendDataBoundItems**: true or false. If true, the statically added item (added from .aspx page) is maintained when adding items dynamically (from code behind file) or items are cleared.
- **OnSelectedIndexChanged**: Method name that fires when user changes the selection of the dropdown box. (Fires only when `AutoPostBack=true`.)



RADIOBUTTONLIST CONTROL

Q.17. Explain RadioButtonList Control with its properties and methods.

ASKED YEAR →

CBSGS: Apr – 2016

SOLUTION

RadioButtonList Control:

- ⇒ The RadioButtonList control, like the DropDownList control, enables a user to select only one list item at a time.
- ⇒ The RadioButttonList control displays a list of radio buttons that can be arranged either horizontally or vertically.
- ⇒ The RadioButtonList control is a single control that groups a collection of a radiobuttons.
- ⇒ It provides Single Selection Checked List.

Properties of RadioButtonList Control:

- **Items**: Represents the collection of items in the list.
- **TextAlign**: Determines how text label is aligned relative to RadioButton. Possible values are Left, and Right.
- **AutoPostBack**: If true, the form is automatically posted when a new RadioButton is selected from the list.
- **SelectedItem**: Represents the selected RadioButton.
- **SelectedIndex**: Specifies the index number of currently checked RadioButton.
- **RepeatColumns**: Determines the number of columns sued to display RadioButton.
- **RepeatDirection**: Indicates the direction in which RadioButton must be repeated.
- **RepeatLayout**: Determines how RadioButton are formatted. Can be Table (default), or Flow.
- **DataSource**: Identifies the data source for the items listed in the list.
- **DataMember**: Identifies the particular table in a data source to bind to the control.
- **DataTextField**: Identifies the field from the data source to use for RadioButton labels.
- **DataValueField**: Identifies the field from the data source to use for RadioButton values.
- **DataTextFormatString**: Gets or sets a format string that determines how DataTextField is displayed.
- **CellPadding**: Sets number of pixels between the border and a particular RadioButton.
- **CellSpacing**: Sets number of pixels between RadioButton in the list.

Methods of RadioButtonList Control:

- **DataBind**: Binds the list to data source. Loads items from data source into the items collection.
- **OnSelectedIndexChanged**: Raises the SelectedIndexChanged event.

Example: The following program illustrates how you can use the RadioButtonList control to display a list of movie titles.

```
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<script runat="server">
protected void btnSubmit_Click(object sender, EventArgs e)
{
    lblMovie.Text = rblMovies.SelectedItem.Text;
}
</script>
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
<title>Show RadioButtonList</title>
</head>
<body>
<form id="form1" runat="server">
<div>
<asp:RadioButtonList id="rblMovies" DataSourceID="srcMovies" DataTextField="Title"
DataValueField="Id" RepeatColumns="3" Runat="server"/>
<asp:Button id="btnSubmit" Text="Submit" Runat="server" OnClick="btnSubmit_Click"/>
<hr/>
<asp:Label id="lblMovie" Runat="server"/>
```



```
<asp:SqlDataSource id="srcMovies" SelectCommand="SELECT Id, Title FROM Movies"
ConnectionString="<%$ ConnectionStrings:Movies %>" Runat="server"/>
</div>
</form>
</body>
</html>
```

Output:

Titanic Ghost Shrek
 Star Wars Forrest Gump Independence Day
 Jurassic Park Ice Age The Ring
 Jaws

 Independence Day

RADIOBUTTONLIST VS. RADIOBUTTON**Q.18. What is the difference between RadioButton and RadioButtonList Control?**

ASKED YEAR →

IDOL: Oct - 2016

SOLUTION**RadioButtonList Vs. RadioButton:**

<u>RadioButtonList:</u>	<u>RadioButton</u>
RadioButtonList is a single control with a list of RadioButton.	RadioButton is a single control.
It is derived from ListControl class. This will work similar to other list controls like ListBox, DropDownList and CheckBoxList.	It is derived from CheckBox Class.
For giving a caption for buttons you can use the Text property.	We have to set the GroupName property to identify a group.
Using the "SelectedIndexChanged" event you will get the selected buttons value ("RadioButtonList1.SelectedValue").	Also the event handler for the event "CheckedChanged" will help us to do some job.
It is easy to bind this to a DataSource.	Another one thing is you have to write separate handlers for each radio button.
Example: <pre><asp:RadioButtonList ID="RadioButtonList1" runat="server" RepeatDirection="Horizontal" onselectedindexchanged= "RadioButtonList1_SelectedIndexChanged"> <asp:ListItem Text="Male" Value="1" ></asp:ListItem> <asp:ListItem Text="Female" Value="2" ></asp:ListItem> </asp:RadioButtonList></pre>	Example: <pre><asp:RadioButton ID="RadioButton1" runat="server" GroupName="Gender" AutoPostBack="true" oncheckedchanged= "RadioButton1_CheckedChanged" Text="Male" /> <asp:RadioButton ID="RadioButton2" runat="server" GroupName="Gender" AutoPostBack="true" oncheckedchanged= "RadioButton2_CheckedChanged" Text="Female" /></pre>

LISTBOX CONTROLS VS. DROPDOWNLIST CONTROLS**Q.19.**

- Explain the similarities and differences between `Listbox` and `DropDownList`. [CBSSGS: Nov – 2015]
- What is the difference between `Listbox` and `DropDownList`? List and explain any three common properties of these Controls. [CBSSGS: Nov – 2014]

ASKED YEAR →

CBSSGS: Nov – 2015 | Nov – 2014

SOLUTIONListbox Vs. DropDownList:

<u>Listbox</u>	<u>DropDown List</u>
A <code>Listbox</code> are used in cases where there are small number of items to be selected.	In contrast, <code>DropDownList</code> are typically used with larger list so that they don't take up much space on the page.
A <code>Listbox</code> lets a user select one or more items from the list of items.	A <code>DropDownList</code> lets a user choose an item from the <code>DropDownList</code> of items.
In <code>Listbox</code> does not allow the user to edit the data inside the list.	<code>DropDown</code> box allows the data to be entered.
<code>Listbox</code> control have properties like <code>Row</code> and <code>SelectionMode</code> (single or multiple).	<code>DropDownList</code> does not have these properties.
The selected <code>Index</code> property of <code>Listbox</code> is set - 1.	The selected <code>Index</code> property of <code>DropDownList</code> it is set to zero (0).

Similarities between Listbox and DropDown:

- ⇒ `Listbox` and `DropDownList` have common properties like `Text`, `Value` and `Selected`.
- ⇒ Both allow the user to display a list of values.

Common Properties:

- **Items**: The collection of `Listitem` objects that represents the items in the control. This property returns of type `ListitemCollection`.
- **Rows**: The number of items that are displayed in a `Listbox` at one time. If the list contains more rows than can be displayed, a scroll bar is added automatically.
- **SelectedItem**: The `Listitem` object for the currently selected item, or the `Listitem` object for the item with the lowest index if more than one item is selected in a `Listbox`.
- **SelectedIndex**: The index of the currently selected item, or the index of the first selected item if more than one item is selected in a `Listbox`. If no item is selected in a `Listbox`, the value of this property is -1.
- **SelectedValue**: The value of the currently selected item, or the value of the first selected item if more than one item is selected in a `Listbox`. If no item is selected in a `Listbox`, the value of this property is an empty string ("").
- **SelectionMode**: Indicates whether a `Listbox` allows single selections (`single`) or multiple selections (`multiple`).

LISTBOX CONTROLS VS. COMBOBOX CONTROLS**Q.20. Differentiate between ListBox and ComboBox Controls.**ASKED YEAR → IDOL: May – 2018 | May – 2016SOLUTIONListBox Vs. ComboBox Controls:

<u>ListBox:</u>	<u>ComboBox:</u>
We can select multiple option from list.	We can select only one option from list.
We can't add information at run time.	We can add information at run time.
In ListBox we can only select item.	In ComboBox we can write/search and select item.
We have no other styles for ListBox.	<u>We have 3 Styles:</u> 1) DropDown Combo 2) Simple Combo 3) DropDown List
ListBox is only listed Items Box.	ComboBox is combination of TextBox and ListBox.
In ListBox, we have Scroll Down and Scroll Up facility.	In ComboBox we have only DropDown facility.
We can use CheckBox within ListBox.	We can't use CheckBox within a ComboBox.
ListBox is much easier to handle.	ComboBox is not easier to handle as compare to ListBox.
We can't add image item in ListBox.	We can add image item in ComboBox.
It Occupies more space but shows more than one value.	It Occupies less space but shows only one value for visibility.

LISTITEM COLLECTION**Q.21. Explain any five Methods / Properties of ListItem Collection Object.**

ASKED YEAR →

CBSGS: Nov – 2014SOLUTIONMethods / Properties of ListItem Collection Object:Property:

- **Capacity**: Gets or sets the maximum number of items that the ListItemCollection can store.
- **Count**: Gets the number of ListItem objects in the collection.
- **IsReadOnly**: Gets a value indicating whether the ListItemCollection is read-only.
- **IsSynchronized**: Gets a value indicating whether access to the ListItemCollection is synchronized (thread-safe).
- **Item[Int32]**: Gets a ListItem at the specified index in the collection.
- **SyncRoot**: Gets the object that can be used to synchronize access to the ListItemCollection.

Method:

- **Add(string)**: Adds a new item to the end of the collection, and assigns the specified string value to both the Text and Value properties of the item.
- **Add(ListItem)**: Adds the specified ListItem to the end of the collection.
- **Insert(integer, string)**: Inserts an item at the specified index location in the collection, and assigns the specified string value to the Text property of the item.
- **Insert(integer, ListItem)**: Inserts the specified list item at the specified index location in the collection.
- **Remove(ListItem)**: Removes the specified list item from the collection.
- **RemoveAt(integer)**: Removes the item at the specified index location from the collection.
- **Clear()**: Removes all the items from the collection.
- **FindByValue(string)**: Returns the ListItem whose Value property has the specified value.
- **FindByText(string)**: Returns the ListItem whose Text property has the specified value.



COMMON DIALOGBOX CONTROLS

Q.22.

- What is Common DialogBox? Explain FontDialog with suitable example.
➤ What are the advantages of using Common DialogBox? [IDOL: May – 2016]

ASKED YEAR →

IDOL: May – 2018 | May – 2016 | Oct – 2012

SOLUTION

Common DialogBox:

- ⇒ Windows implements a variety of reusable dialog boxes that are common to all applications, including dialog boxes for opening files, saving files, and printing.
- ⇒ Since these dialog boxes are implemented by the operating system, they can be shared among all the applications that run on the operating system, which helps user experience consistency; when users are familiar with the use of an operating system-provided dialog box in one application, they don't need to learn how to use that dialog box in other applications.
- ⇒ Because these dialog boxes are available to all applications and because they help provide a consistent user experience, they are known as common dialog boxes.

FontDialog Box:

- ⇒ FontDialog Box represents a common DialogBox that displays a list of fonts that are currently installed on the system.
- ⇒ The FontDialog box lets the user choose attributes for a logical font, such as font family and associated font style, point size, effects, and a script.

Example: The following C# program invites a FontDialog Box and retrieve the selected Font Name and Font Size.

```
using System;
using System.Drawing;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            FontDialog dlg = new FontDialog();
            dlg.ShowDialog();
            if (dlg.ShowDialog() == DialogResult.OK)
            {
                string fontName;
                float fontSize;
                fontName = dlg.Font.Name;
                fontSize = dlg.Font.Size;
                MessageBox.Show(fontName + " " + fontSize );
            }
        }
    }
}
```



MENUS AND TOOLBARS CONTROL

Q.23. What is the use of Menus and Toolbars in Windows Application? Explain.

ASKED YEAR → IDOL: May – 2018 | May – 2017 | Oct – 2012

SOLUTION

Use of Menus and Toolbars in Windows Application:

Menu :

- ⇒ The Menu control has two modes of display: static and dynamic.
- ⇒ Static display means that the Menu control is fully expanded all the time. The entire structure is visible, and a user can click on any part.
- ⇒ In a dynamically displayed menu, only the portions you specify are static, while their child menu items are displayed when the user holds the mouse pointer over the parent node.
- ⇒ We can configure the contents of the Menu control directly in the control, or we can specify the contents by binding the control to a data source.
- ⇒ Without writing any code, we can control the appearance, orientation, and content of an ASP.NET Menu control.
- ⇒ In addition to the visual properties exposed by the control, the control supports ASP.NET control skins and themes.

ToolBar :

- ⇒ The Windows Forms `ToolBar` control is used on forms as a control bar that displays a row of drop-down menus and bitmapped buttons that activate commands. Thus, clicking a toolbar button can be an equivalent to choosing a menu command.
- ⇒ The buttons can be configured to appear and behave as pushbuttons, drop-down menus, or separators.
- ⇒ Typically, a `ToolBar` contains buttons and menus that correspond to items in an application's menu structure, providing quick access to an application's most frequently used functions and commands.
- ⇒ The `ToolBar` control allows you to create toolbars by adding `Button` objects to a `Buttons` collection.
- ⇒ We can use the `Collection Editor` to add buttons to a `ToolBar` control; each `Button` object should have text or an image assigned, although you can assign both.
- ⇒ The image is supplied by an associated `ImageList` component. At run time, we can add or remove buttons from the `ToolBar.ToolBarButtonCollection` using the `Add` and `Remove` methods.
- ⇒ To program the buttons of a `ToolBar`, add code to the `ButtonClick` events of the `ToolBar`, using the `Button` property of the `ToolBar.ButtonClickEventArgs` class to determine which button was clicked.

TOOLSTRIPMENUITEM

Q.24. List and explain the properties of `ToolStripMenuItem`.

ASKED YEAR → IDOL: Apr – 2014

SOLUTION

Properties of the ToolStripMenuItem Control:

- **Checked**: Gets or sets a value indicating whether the `ToolStripMenuItem` is checked.
- **CheckOnClick**: Gets or sets a value indicating whether the `ToolStripMenuItem` should automatically appear checked and unchecked when clicked.
- **CheckState**: Gets or sets a value indicating whether a `ToolStripMenuItem` is in the checked, unchecked, or indeterminate state.
- **Enabled**: Gets or sets a value indicating whether the control is enabled.
- **IsMdiWindowListEntry**: Gets a value indicating whether the `ToolStripMenuItem` appears on a multiple document interface (MDI) window list.
- **ShortcutKeyDisplayString**: Gets or sets the shortcut key text.
- **ShortcutKeys**: Gets or sets the shortcut keys associated with the `ToolStripMenuItem`.



- **ShowShortcutKeys**: Gets or sets a value indicating whether the shortcut keys that are associated with the ToolStripMenuItem are displayed next to the ToolStripMenuItem.

STATUSSTRIP CONTROL

Q.25. Explain StatusStrip Control.

ASKED YEAR → **IDOL**: Apr – 2015

SOLUTION

StatusStrip Control:

- ⇒ StatusStrip control is a powerful StatusBar control.
- ⇒ StatusStrip provides ProgressBar, DropDownButton, SplitButton, and Label features, you can add a progress bar, a drop down button, a SplitButton, or a label control to the StatusStrip itself.

StatusStrip Properties:

- ⇒ The StatusStrip class is inherited from the ToolStrip → ScrollableControl → Control classes and hence has all of the common properties supported by a Windows Forms control.
- ⇒ Some of these common properties include Font, BackColor, Dock, BackgroundImage, and TextDirection.
- ⇒ We can set the StatusStrip Control properties using the Properties Window.

How to use StatusStrip Control:

- ⇒ Drag and drop StatusStrip control from toolbox on the window Form.
- ⇒ Select control which we want to show in StatusStrip.
- ⇒ Here progress bar is added. By default maximum value of ProgressBar is 100 and minimum value is 0.

Code:

```
private void frmStatusStrip_Load(object sender, EventArgs e)
{
    while (toolStripProgressBar1.Value < 100)
    {
        // Progressbar value will increase 5 untill its value will not reach to 100.
        toolStripProgressBar1.Value += 5;
    }
}
```

Output: ProgressBar will show when Application Run



**MULTIPLE DOCUMENT INTERFACE (MDI)****Q.26.** Explain the concept of MDI in windows Application. Also explain how to define child forms?ASKED YEAR → **IDOL:** Apr – 2013**SOLUTION****Multiple Document Interface (MDI):**

- ⇒ An MDI (Multiple Document Interface) application is an application in which we can view and work with several documents at once like Microsoft Excel or Visual Studio 2010.
- ⇒ MDI applications having a feature of MDI child forms and its very essential element of it.

Step to Create and Implement MDI Child Form:

STEP 1: Create a MDI parent form with a `MenuStrip` which containing the values New and Windows and Close in the sub menu of New.

STEP 2: Now go to Properties Window, select the all menu items and set the `MdiList` property to true.

STEP 3: Now add MDI child forms by going in Solution Explorer, right-click the project, select Add New Item from Add Menu.

STEP 4: A Dialog box appear, now select Windows Form give the name as Form2 and click on open button. A New Form Name as Form2 added.

STEP 5: Add a `RichTextBox` Control to the Form from Toolbox.

STEP 6: Now go on Property Window and Set Properties for the `RichTextBox`. Set Anchor property to Top, Left and the Dock property to fill.

STEP 7: Click on New Menu Item and generate a Click Event Handler on it.

STEP 8: Now Double click on New Menu Item and Write down the following Code.

```
protectedvoid MDIChildNew_Click(object sender, System.EventArgs e)
{
    Form2 newMDIChild = newForm2();
    // Set the Parent Form of the Child window.
    newMDIChild.MdiParent =this;
    // Display the new form.
    newMDIChild.Show();
}
```

STEP 9: Now Run the application, And from output window you can create a new MDI child form by clicking on New Menu Item.

SDI Vs. MDI**Q.27.**

- What is the difference between SDI and MDI? [**IDOL:** Apr – 2014]
- Write a short note on MDI. [**IDOL:** Oct – 2016]

ASKED YEAR → **IDOL:** Oct – 2016 | Apr – 2014**SOLUTION****MDI Vs. SDI:**

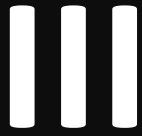
<u>MDI</u>	<u>SDI</u>
MDI stands for "Multiple Document Interface".	SDI stands for "Single Document Interface".
Child Windows per document are allowed in MDI.	One document per window is enforced in SDI
MDI is a container control while.	SDI is not container control.
MDI contain multiple document at a time appeared as child window.	SDI contains one window only at a time.
MDI supports many interfaces means we can handle many applications at a time according to user's requirement.	SDI supports one interface means you can handle only one application at a time.
For switching between documents MDI uses special interface inside the parent window	For switching between documents SDI uses Task Manager.



In MDI grouping is implemented naturally.	In SDI grouping is possible through special window managers.
For maximizing all documents, parent window is maximized by MDI.	SDI it is implemented through special code or window manager.
Switch focus to specific document can be easily handled in MDI.	It is difficult to implement in SDI.



UNIT



ASP.NET Applications CSS (Cascading Style Sheet) ASP.NET Server Controls

Topic

ASP.NET APPLICATIONS

- ⇒ ASP.NET LIFE CYCLE
- ⇒ ASP.NET APPLICATION LIFE CYCLE
- ⇒ ASP.NET PAGE LIFE CYCLE
- ⇒ ASP.NET PAGE LIFE CYCLE PHASES
- ⇒ ASP.NET PAGE LIFE CYCLE STAGES
- ⇒ ASP.NET PAGE LIFE CYCLE EVENTS
- ⇒ DIRECTIVES
- ⇒ TYPES OF ASP.NET DIRECTIVES
 - > PAGE DIRECTIVE
 - > APPLICATION DIRECTIVE
 - > ASSEMBLY DIRECTIVE
 - > CONTROL DIRECTIVE
 - > IMPLEMENTS DIRECTIVE
 - > IMPORT DIRECTIVE
 - > MASTER DIRECTIVE
 - > MASTERTYPE DIRECTIVE
 - > OUTPUTCACHE DIRECTIVE
 - > PREVIOUSPAGE TYPE DIRECTIVE
 - > REFERENCE DIRECTIVE
 - > REGISTER DIRECTIVE
- ⇒ CODE RENDER BLOCKS
- ⇒ INLINE CODE RENDER BLOCKS
- ⇒ INLINE EXPRESSION RENDER BLOCKS
- ⇒ THEMES IN ASP.NET
- ⇒ TYPES OF THEMES
 - > CUSTOMIZATION THEMES
 - > STYLESHEET THEMES
- ⇒ USES OF THEMES AND SKINS / THEMES Vs. SKINS
- ⇒ CREATING AND USING NAMED THEME
- ⇒ SKINS IN ASP.NET

⇒ TYPES OF CONTROL SKINS

- > DEFAULT SKIN
- > NAMED SKIN

CASCADING STYLE SHEET (CSS)

- ⇒ NEED OF CSS
- ⇒ ADVANTAGES OF CSS
- ⇒ DISADVANTAGES OF CSS
- ⇒ TYPES OF CSS
 - > INLINE STYLE SHEET
 - > INTERNAL/EMBEDDED STYLE SHEET
 - > EXTERNAL STYLE SHEET
- ⇒ CSS SELECTOR
- ⇒ TYPES OF CSS SELECTOR
 - > UNIVERSAL SELECTOR
 - > ELEMENT TYPE SELECTOR / TYPE SELECTOR
 - > ID SELECTOR
 - > CLASS SELECTOR
 - > GROUPING SELECTOR
- ⇒ BASIC SELECTOR OF CSS
- ⇒ STEPS OF CREATING CSS STYLE SHEET
- ⇒ CSS Vs. HTML
- ⇒ CSS FIX HTML FORMATTING PROBLEMS
- ⇒ <LINK> TAG

ASP.NET SERVER CONTROLS

- ⇒ USER CONTROL
 - > CREATING A USER CONTROL IN ASP.NET
 - > STEPS IN CREATING USER CONTROL
 - > REDIRECT USERS TO ANOTHER PAGE USING THE BROWSER
 - > REDIRECT USERS TO ANOTHER PAGE USING A SERVER-SIDE METHOD



- ⇒ LISTVIEW CONTROL
- ⇒ SERVER CONTROL
- ⇒ HTML SERVER CONTROLS VS. WEB SERVER
- ⇒ WEB USER CONTROL VS. WEB SERVER CONTROL
- ⇒ STATE MANAGEMENT
- ⇒ TYPES OF STATE MANAGEMENT
 - > SESSION - SERVER SIDE
 - > APPLICATION - SERVER SIDE
 - > COOKIES - CLIENT SIDE
 - > VIEWSTATE - CLIENT SIDE
 - > QUERYSTRING - CLIENT SIDE
 - > HIDDENFIELD - CLIENT SIDE
- ⇒ APPLICATION EVENTS WHEN AN APPLICATION IS EXECUTED
- ⇒ ADVANTAGES AND DISADVANTAGES OF SESSION STATE
- ⇒ CONFIGURING SESSION TIMEOUT
- ⇒ WORKING OF SESSION STATE
- ⇒ SESSION STATE VARIABLES
- ⇒ WORKING OF VIEWSTATE
- ⇒ SIGNIFICANCE OF VIEWSTATE
- ⇒ APPLICATION STATE VS. SESSION STATE
- ⇒ SECURING VIEWSTATE
- ⇒ VIEWSTATE FIELD
- ⇒ ROLES OF COOKIES
- ⇒ HOW SERVER SETS A COOKIE AND RETRIEVES IT?
- ⇒ ADVANTAGES AND DISADVANTAGES OF COOKIES
- ⇒ PROPERTIES OF HTTPCOOKIE CLASS
- ⇒ ENCODING AND DECODING OF QUERY STRING
- ⇒ NEED OF QUERY STRING IN ASP.NET
- ⇒ GLOBAL.ASAX
 - > GLOBAL.ASAX EVENTS
 - > EVENTS WHICH ARE FIRED FOR EVERY REQUEST
 - > EVENTS WHICH ARE NOT FIRED FOR EVERY REQUEST
 - > MAJOR EVENTS IN GLOBAL.ASAX FILE
 - > EVENT HANDLERS IN GLOBAL.ASAX FILE
- ⇒ ASP.NET CONFIGURATION FILES
 - > WEB.CONFIG FILE
 - > USES / IMPORTANCE OF WEB.CONFIG FILE
 - > STRUCTURE / GENERAL FORM OF WEB.CONFIG

**ASP.NET LIFE CYCLE****Q.1.**

- Explain the ASP.NET Life Cycle. [IDOL: May – 2017 | Apr – 2014]
- Explain the Event Life Cycle of ASP.NET? [IDOL: Apr – 2013] | [CBSGS: Nov – 2015]
- Explain ASP.NET Page Life Cycle. [IDOL: Dec – 2017 | Oct/May – 2016 | Oct – 2012] | [CBSGS: Nov/Apr – 2016 | Apr – 2014]

ASKED YEAR ⇒

[IDOL: Dec/May – 2017 | Oct/May – 2016 | Apr – 2014 | Apr – 2013 | Oct – 2012

| [CBSGS: Nov/Apr – 2016 | Nov – 2015 | Apr – 2014

SOLUTION**ASP.NET Life Cycle:** (IDOL: May – 2017 | Apr – 2014)

- ⇒ When a page is requested, it is loaded into the server memory, processed and sent to the browser.
- ⇒ Then it is unloaded from the memory.
- ⇒ At each of these steps, methods and events are available, which could be overridden according to the need of the application.
- ⇒ The ASP.NET Life Cycle could be divided into two groups:
 - 1) *Application Life Cycle*
 - 2) *Page Life Cycle*

ASP.NET Application Life Cycle:

- ⇒ The application life cycle has the following stages:
- ⇒ User makes a request for accessing application resource, a page. Browser sends this request to the web server.
- ⇒ A unified pipeline receives the first request and the following events take place:
 - An object of the class `ApplicationManager` is created.
 - An object of the class `HostingEnvironment` is created to provide information regarding the resources.
 - Top level items in the application are compiled.
- ⇒ Response objects are created. The application objects such as `HttpContext`, `HttpRequest` and `HttpResponse` are created and initialized.
- ⇒ An instance of the `HttpApplication` object is created and assigned to the request.
- ⇒ The request is processed by the `HttpApplication` class. Different events are raised by this class for processing the request.

ASP.NET Page Life Cycle: (IDOL: Dec – 2017 | Oct/May – 2016 | Oct – 2012) | (CBSGS: Nov/Apr – 2016 | Nov – 2015 | Apr – 2014)

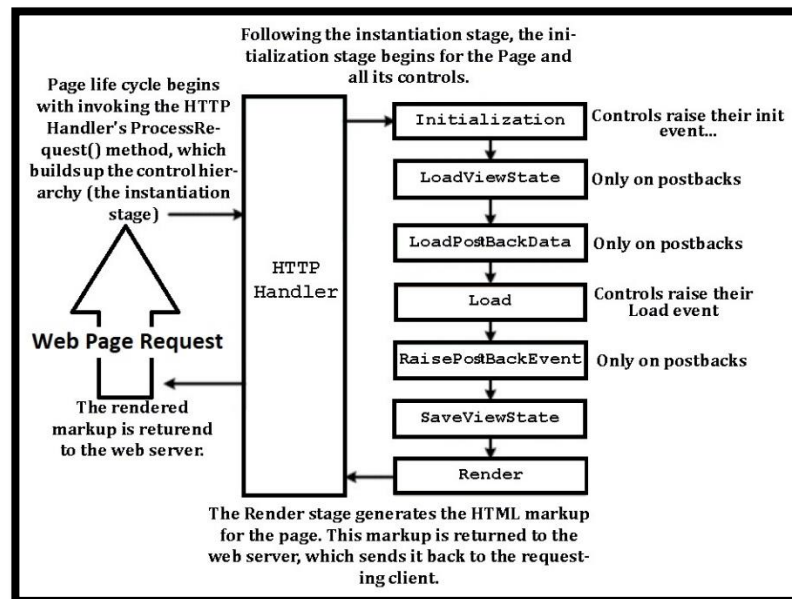
- ⇒ When a page is requested, it is loaded into the server memory, processed, and sent to the browser.
- ⇒ Then it is unloaded from the memory. At each of these steps, methods and events are available, which could be overridden according to the need of the application.

The Page Life Cycle Phases are:

- *Initialization*
- *Instantiation of the Controls on the Page*
- *Restoration and Maintenance of the State*
- *Execution of the Event Handler Codes*
- *Page Rendering*

Stages of ASP.NET Page Life Cycle:

Following are the different stages of an ASP.NET page:



STAGE 0: Page Request

- ⇒ When a POST request is initiated from the Client Side (web browser), the web server gets the request and it is routed to the HTTP pipeline where the HTTP Page handler class is identified and the `ProcessRequest()` method is called which ultimately fires the different page events in the life cycle of any web page.

STAGE 1: Page Initialization

- ⇒ The first stage in the Page Life Cycle is Initialization and the associated event is `Page_Init` that is triggered in the page life cycle.
- ⇒ It initializes all controls that are statically declared in the .aspx file with the default values.
- ⇒ Controls can use the `Page_Init` event to initialize some of the settings that can be used throughout the lifetime of the incoming web request.
- ⇒ It also initializes all the server controls of the web page with their default values.
- ⇒ ViewState information for the page will not be available at this stage.

STAGE 2: View State Loading

- ⇒ Next, the `LoadViewState` method is called. This stage only happens when the page has been posted back.
- ⇒ ViewState is a collection of name/value pairs, where control's and page itself store information that is persistent among web requests.
- ⇒ At this stage, the view state data saved from the previous page visit is loaded and recursively populated into the Page's control hierarchy.
- ⇒ This method restores the ViewState information of a web page that was last saved using the `SaveViewState` method.
- ⇒ By overriding the `LoadViewState()` method, component developer can understand how ViewState is restored.

STAGE 3: Postback Data Processing

- ⇒ Once the ViewState is restored, control will be updated with the client side posted data values.
- ⇒ This is done by `LoadPostData` event.

STAGE 4 & 5: Page Loading & PostBack Change Notification

- ⇒ And then, at the end of the posted data changes event, controls will be reflected with changes done on the client.
- ⇒ At this point, `Page_Load` event and `RaisePostBackDataChangedEvent` are fired.

STAGE 6: PostBack Event Handling

- ⇒ Key event in the ASP.NET Page Life Cycle is when the server-side code associated with an event is triggered on the client – When the user clicks on the button, the page posts back.



⇒ Page framework calls the `RaisePostBackEvent` that looks up for the event handler and runs the associated delegate.

STAGE 7: Page Pre Rendering Phase

⇒ Then, Page prepares for rendering due to `Page_PreRender` event.

⇒ In this stage, user does the update operations before the `ViewState` is stored and output is rendered.

STAGE 8: View State Saving

⇒ Then the `SaveViewState` event is called where all the values of the controls will be saved to their own `ViewState` collection.

⇒ The resultant `ViewState` is serialized, hashed, base24 encoded and associated with the `_ViewState` Hidden Field.

STAGE 9: Page Rendering

⇒ Next the `Page_Render` method is called which takes the `HtmlWriter` object so as to collect all HTML text to be generated for the control.

⇒ For each control, the web page calls the render method and caches the HTML output. The rendering mechanism can be altered by overriding this render method.

STAGE 10: Page Unloading

⇒ This is the final stage of the ASP.NET Page Life Cycle which calls the `Page_UnLoad` to do the final cleanup of the page objects or resources.

⇒ This event releases the complex resources such as database connections, files, graphical objects etc.

⇒ After this event, browser receives the HTTP response packet and displays the output page.

ASP.NET Page Life Cycle Events: (IDOL: Apr – 2013) | (CBSGS: Nov – 2015)

- **PreInit:** `PreInit` is the first event in page life cycle. It checks the `IsPostBack` property and determines whether the page is a postback. It sets the themes and master pages, creates dynamic controls and gets and sets profile property values. This event can be handled by overloading the `OnPreInit` method or creating a `Page_PreInit` handler.
- **Init:** `Init` event initializes the control property and the control tree is built. This event can be handled by overloading the `OnInit` method or creating a `Page_Init` handler.
- **InitComplete:** `InitComplete` event allows tracking of view state. All the controls turn on view-state tracking.
- **LoadViewState:** `LoadViewState` event allows loading view state information into the controls.
- **LoadPostData:** During this phase, the contents of all the input fields defined with the `<form>` tag are processed.
- **PreLoad:** `PreLoad` occurs before the post back data is loaded in the controls. This event can be handled by overloading the `OnPreLoad` method or creating a `Page_PreLoad` handler.
- **Load:** The `Load` event is raised for the page first and then recursively for all child controls. The controls in the control tree are created. This event can be handled by overloading the `OnLoad` method or creating a `Page_Load` handler.
- **LoadComplete:** The loading process is completed, control event handlers are run and page validation takes place. This event can be handled by overloading the `OnLoadComplete` method or creating a `Page_LoadComplete` handler.
- **PreRender:** The `PreRender` event occurs just before the output is rendered. By handling this event, pages and controls can perform any updates before the output is rendered.
- **PreRenderComplete:** As the `PreRender` event is recursively fired for all child controls, this event ensures the completion of the pre-rendering phase.
- **SaveStateComplete:** State of control on the page is saved. Personalization, control state and view state information is saved. The HTML markup is generated. This stage can be handled by overriding the `Render` method or creating a `Page_Render` handler.
- **Unload:** The `Unload` phase is the last phase of the page life cycle. It raises the `Unload` event for all controls recursively and lastly for the page itself. Final cleanup is done and all resources and references, such as database connections, are freed. This event can be handled by modifying the `OnUnload` method or creating a `Page_UnLoad` handler.



DIRECTIVES

Q.2. Explain the different Directives supported by ASP.NET.

ASKED YEAR → IDOL: Dec – 2017 | May – 2016

SOLUTION

ASP.NET Directives:

- ⇒ ASP.NET directives are instructions to specify optional settings, such as registering a custom control and page language.
- ⇒ These settings describe how the web forms (.aspx) or user controls (.ascx) pages are processed by the .Net framework.

The syntax for declaring a directive is:

```
<%@ directive_name attribute=value [attribute=value] %>
```

Types of ASP.NET Directives:

The Application Directive:

- ⇒ The Application directive defines application-specific attributes.
- ⇒ It is provide .
- ⇒ at the top of the global.aspx file.

The basic syntax of Application directive is:

```
<%@ Application Language="C#" %>
```

The Assembly Directive:

- ⇒ The Assembly directive links an assembly to the page or the application at parse time.
- ⇒ This could appear either in the global.asax file for application-wide linking, in the page file, a user control file for linking to a page or user control.

The basic syntax of Assembly directive is:

```
<%@ Assembly Name ="myassembly" %>
```

The Control Directive:

- ⇒ The control directive is used with the user controls and appears in the user control (.ascx) files.

The basic syntax of Control directive is:

```
<%@ Control Language="C#" EnableViewState="false" %>
```

The Implements Directive:

- ⇒ The Implement directive indicates that the web page, master page or user control page must implement the specified .Net framework interface.

The basic syntax for implements directive is:

```
<%@ Implements Interface="interface_name" %>
```

The Import Directive:

- ⇒ The Import directive imports a namespace into a web page, user control page of application.
- ⇒ If the Import directive is specified in the global.asax file, then it is applied to the entire application.
- ⇒ If it is in a page of user control page, then it is applied to that page or control.

The basic syntax for import directive is:

```
<%@ namespace="System.Drawing" %>
```



The Master Directive:

⇒ The Master directive specifies a page file as being the mater page.

The basic syntax of sample MasterPage directive is:

```
<%@ MasterPage Language="C#" AutoEventWireup="true" CodeFile="SiteMater.master.cs"
Inherits="SiteMaster" %>
```

The MasterType Directive:

⇒ The MasterType directive assigns a class name to the Master property of a page, to make it strongly typed.

The basic syntax of MasterType directive is:

```
<%@ MasterType attribute="value" [attribute="value" ...] %>
```

The OutputCache Directive:

⇒ The OutputCache directive controls the output caching policies of a web page or a user control.

The basic syntax of OutputCache directive is:

```
<%@ OutputCache Duration="15" VaryByParam="None" %>
```

The Page Directive:

⇒ The Page directive defines the attributes specific to the page file for the page parser and the compiler.

The basic syntax of Page directive is:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default"
Trace="true" %>
```

The PreviousPageType Directive:

⇒ The PreviousPageType directive assigns a class to a page, so that the page is strongly typed.

The basic syntax for a sample PreviousPageType directive is:

```
<%@ PreviousPageType attribute="value" [attribute="value" ...] %>
```

The Reference Directive:

⇒ The Reference directive indicates that another page or user control should be compiled and linked to the current page.

The basic syntax of Reference directive is:

```
<%@ Reference Page ="somepage.aspx" %>
```

The Register Directive:

⇒ The Register derivative is used for registering the custom server controls and user controls.

The basic syntax of Register directive is:

```
<%@ Register Src="~/footer.ascx" TagName="footer" TagPrefix="Tfooter" %>
```


**Q.3. What is the place of Page Directive? Explain its attributes.**

ASKED YEAR ⇒

CBSGS: Apr – 2016

SOLUTION**Page Directive:**

- ⇒ Page Directive allows you to specify many configuration options for the page.
- ⇒ Page directives are instructions, inserted at the top of an ASP.Net page, to control the behavior of the asp.net pages.

The basic syntax of page directive is –

```
<%@page Language="C#" AutoEventWireup="true" codeFile="Default.aspx.cs" Inherits="_Default" Trace="true"%>
```

The attributes of the page directive are:

- **AutoEventWireup** – The Boolean value that enables or disables page events that are being automatically bound to methods e.g. page_load.
- **EnableViewState** – The Boolean value that enables or disables view state across page requests.
- **Inherits** – The name of the code behind or other class.

The methods of the page directive are:

- **OnSelectedIndexChanged** – Raises the SelectedIndexChanged event. This allows you to provide a custom handler for the event.
- **OnLoad (EventArgs)** – Handles the control Load event.
- **GetData ()** – Retrieves a DataSourceView object that the data-bound control uses to perform data operations.
- **Dispose ()** – Enables a server control to perform final clean up before it is released from memory.
- **Focus ()** – Sets input focus to a control.
- **Language** – The programming language for code.
- **Codefile** – The name of the code behind file.
- **Trace** – If enables or disables tracing.
- **Transaction** – It indicates if transaction are supported.

Q.4. Explain src, tagprefix and tagname attributes of @ register directive.

ASKED YEAR ⇒

IDOL: Apr – 2013

SOLUTION**The Register Directive:**

- ⇒ The Register derivative is used for registering the custom server controls and user controls.
- ⇒ When you create a user control and you drag that user control onto your page then you will see the @Register directive.
- ⇒ This directive registers your user control on the page so that the control can be accessed by the page.

Example:

```
<%@ Register TagPrefix="MayTag Namespace="MyName.MyNameSpace" Assembly="MyAssembly"%>
```

Attributes of Register Directive:

- **Assembly:** It specifies the assembly we want to associating with the TagPrefix.
- **Namespace:** Specifies the namespace to relate with TagPrefix.
- **Src:** Specifies the user control.
- **TagName:** Specifies alias name to relate to your class name.
- **TagPrefix:** Specifies alias name to relate to the namespace.



CODE RENDER BLOCKS

Q.5. Explain the Code Render Block with suitable example.

ASKED YEAR → IDOL: May – 2016

SOLUTION

Code Render Blocks:

- ⇒ We can use Code Render Blocks to define inline code or expressions that will execute when a page is rendered.
- ⇒ Code within a code render block is executed immediately when it is encountered—usually when the page is loaded or rendered.
- ⇒ On the other hand, code within a code declaration block (within `<script>` tags) is executed only when it is called or triggered by user or page interactions.
- ⇒ There are two types of Code Render Blocks—Inline Code, and Inline Expressions — both of which are typically written within the body of the ASP.NET page.

Inline Code Render Blocks:

- ⇒ Inline Code Render Blocks execute one or more statements, and are placed directly inside a page's HTML between `<%` and `>` delimiters.
- ⇒ In our example, the following is a code render block:

```
<% string Title = "This is generated by a code render block."; %>
```
- ⇒ These code blocks simply declare a String variable called Title, and assign it the value this is generated by a code render block.

Inline Expression Render Blocks:

- ⇒ Inline Expression render blocks can be compared to Response.Write in classic ASP.
- ⇒ They start with `<%=` and end with `>`, and are used to display the values of variables and methods on a page.
- ⇒ In our example, an inline expression appears immediately after our inline code block:

```
<%= Title %>
```
- ⇒ If we are familiar with classic ASP, we'll know what this code does: it simply outputs the value of the variable Title that we declared in the previous inline code block.

Example:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_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></div>
<% for (int i = 0; i <= 6; i++)
{ %>
<div>
</div>
<font size="<%=i%>"> Welcome to ASP.NET 3.5 </font>
<% } %>
</form>
</body>
</html>
```



THEMES AND SKINS

Q.6.

- **What is the difference between Themes and Skins in ASP.NET?** [IDOL: May – 2018 | Apr – 2014]
- **What are Skins and why are they used?** [IDOL: Apr – 2015 | Apr – 2013]
- **What are Themes? Explain its different types.** [IDOL: May – 2017 | Oct – 2016]

ASKED YEAR ⇒

IDOL: May – 2018 | May – 2017 | Oct – 2016 | Apr – 2015 | Apr – 2014 | Apr – 2013

SOLUTION

Themes :

- ⇒ A theme is a collection of property settings that allow us to define the look of pages and controls, and then apply the look consistently across pages in a Web application, across an entire Web application, or across all Web applications on a server.
- ⇒ Themes are made up of a set of elements such as skins, cascading style sheets (CSS), images, and other resources.
- ⇒ At a minimum, a theme will contain skins.
- ⇒ Themes are defined in special directories in our Web site or on our Web server.
- ⇒ A theme can also include a cascading style sheet ".css" file. When we put a ".css" file in the theme folder, the style sheet is applied automatically as part of the theme. We define a style sheet using the file name extension ".css" in the theme folder.
- ⇒ Themes can also include graphics and other resources, such as script files or sound files.
- ⇒ We can define themes for a single Web application, or as global themes that can be used by all applications on a Web server.

Types of Themes:

Customization Themes:

These types of themes are applied after the properties of the control are applied, meaning that the properties of the themes override the properties of the control itself.

Stylesheet Themes:

We can apply this type of theme to a page in exactly the same manner as a customization theme.

However, stylesheet themes don't override control properties, thus allowing the control to use the theme properties or override them.

Skins:

- ⇒ A Skin file has the file name extension .skin and contains property settings for individual controls such as Button, Label, TextBox, or Calendar controls.
- ⇒ Control skin settings are like the control markup itself, but contain only the properties you want to set as part of the theme.
- ⇒ For example, the following is a control skin for a Button control:

```
<asp:button runat="server" BackColor="lightblue" ForeColor="black" />
```
- ⇒ We create ".skin" files in the Theme folder. A ".skin" file can contain one or more control skins for one or more control types.
- ⇒ We can define skins in a separate file for each control or define all the skins for a theme in a single file.

Types of Control Skins:

There are two types of Control Skins – Default Skins and Named Skins:

Default Skin:

- ⇒ A Default Skin automatically applies to all controls of the same type when a theme is applied to a page.
- ⇒ A control skin is a default skin if it does not have a SkinID attribute.

Named Skin:

- ⇒ A Named Skin is a control skin with a SkinID property set.



- ⇒ Named skins do not automatically apply to controls by type. Instead, we explicitly apply a named skin to a control by setting the control's `SkinID` property.
- ⇒ Creating Named Skins allows you to set different skins for different instances of the same control in an application.

Uses of Themes and Skins / Themes Vs. Skins:

Themes :

- ⇒ Since themes can contain CSS files, images and skins, you can change colors, fonts, positioning and images simply by applying the desired themes.
- ⇒ You can have as many themes as you want and you can switch between them by setting a single attribute in the web.config file or an individual aspx page. Also you can switch between themes programmatically.
- ⇒ Setting the themes programmatically, you are offering your users a quick and easy way to change the page to their likings.
- ⇒ Themes allow you to improve the usability of your site by giving users with vision problems the option to select a high contrast theme with a large font size.

Skins :

- ⇒ Skins can be used to apply the common style to the individual ASP.Net controls.
- ⇒ You can apply a style to the specific control.
- ⇒ It easy to remind which skin is for which control because inside the skin file ASP.Net controls are directly used and in .aspx pages it automatically appears to that control id style.
- ⇒ Gives the same consistency to the multiple controls in the web application.

Q.7.**Explain procedure of creating and using named theme in a website.**

ASKED YEAR ⇒

IDOL: May – 2016

SOLUTION

Applying Themes to a Web Site:

We can apply a theme to an entire Web site, which means we do not need to apply the theme to individual pages. (If we want, we can override the themes settings on a page.)

To set a theme for a Web Site:

STEP 1: Open the Web.config file.

STEP 2: In the pages element, add a theme attribute and set its value to the name of theme that you want to apply to the entire Web site, as in the following example:

```
<pages theme="sampleTheme">
```

STEP 3: Save and close the Web.config file.

STEP 4: Switch to or open the Default.aspx file, and then switch to Source view.

STEP 5: Remove the theme attribute (`theme="themeName"`) from the @ Page declaration.

STEP 6: Press CTRL+F5 to run Default.aspx.

The page is now displayed with the theme you specified in the Web.config file.

If you choose to specify a theme name in your page declaration, it will override any theme specified in the Web.config file.]

**CSS (CASCADING STYLE SHEET)**

Q.8.

- **What is CSS?** [IDOL: May – 2018 | Dec/May – 2017 | Apr – 2015 | Oct – 2012 | Apr – 2013] | [CBSGS: Apr – 2016 | Apr – 2015]
- **Explain Inline Style Sheet.** [IDOL: May – 2018 | Dec/May – 2017 | Oct – 2012]
- **Explain External Style Sheet.** [IDOL: Apr – 2014] | [CBSGS: Nov – 2015]
- **Explain Embedded Style Sheet.** [IDOL: May – 2018 | Dec/May – 2017 | Apr – 2014 | Oct – 2012]
- **How would you link an HTML Page to an External Style Sheet?** [CBSGS: Nov – 2015]
- **Explain the different types of CSS in ASP.NET.** [CBSGS: Apr – 2015 | Apr/Nov – 2014 | Oct – 2013]
- **What is the advantages and disadvantages of CSS?** [CBSGS: Apr – 2015]
- **What is the need of CSS in ASP.NET?** [CBSGS: Oct – 2013]
- **List and explain the advantages of CSS over HTML.** [IDOL: May – 2016] | [CBSGS: Apr – 2015]

ASKED YEAR ⇒

IDOL: May – 2018 | Dec/May – 2017 | May – 2016 | Oct – 2012

CBSGS: Apr – 2016 | Nov/Apr – 2015 | Nov/Apr – 2014 | Oct – 2013

SOLUTION**Cascading Style Sheet (CSS):** (IDOL: May – 2018 | Dec/May – 2017 | Apr – 2015 | Oct – 2012) | (CBSGS: Apr – 2016 | Apr – 2015)

- ⇒ CSS stands for Cascading Style Sheet.
- ⇒ It is a Style sheet language.
- ⇒ It describes the presentation of an HTML or XML document.
- ⇒ CSS is used to control the web document in a simple and easy way.

Need of CSS: (CBSGS: Oct – 2013)

- ⇒ HTML has very limited set of tags or options to style the web pages.
- ⇒ Its feature set severely limits the formatting possibilities that the page requires.
- ⇒ **Example:** `<p face="arial" color="red" size="4">` this is read text in an arial font and of size 4.
- ⇒ In this case the data and presentation are mixed in the same file.
- ⇒ The required formatting tags and attributes make the pages larger and slower to load and display.
- ⇒ **CSS fixes the formatting problems:**
 - CSS offers a rich set of options to change every little aspect of the web site.
 - CSS is understood by all the browsers and so is the language for visual presentation of web pages.
 - It overcomes the problem of mixed data and presentation by enabling the programmer to define all formatting information in external files.
 - Another benefit of separate style sheet is the decrease in bandwidth that is required for the site. Style sheets don't change the each request, so a browser saves a local copy of the style the first time it downloads it.

Advantages of CSS: (CBSGS: Apr – 2015)

- ⇒ **Write Once Use Anywhere/Reusability:** You can write one style sheet with a number of properties and attach this style sheet to any number of pages on your site.
- ⇒ **Workflow:** CSS allows you to maintain the integrity of your data.
- ⇒ **Reduced Bandwidth Costs:** CSS is downloaded only once in the external style sheet document. When surfing the rest of our site, the CSS will be cached on the user's computer, and therefore speeds up the loading time.
- ⇒ **Higher Search Engine Rankings:** Cleaner code is easier for search engines to index.
- ⇒ **Faster Download:** CSS requires less code and it can control the order of the elements that are downloaded such as content before images.
- ⇒ **Better Usability:** We can give your pages a distinct look and a uniform "branded" style without the use of graphics.
- ⇒ **Increased Reach:** CSS website is compatible with many different devices.

Disadvantages of CSS: (CBSGS: Apr – 2015)

- ⇒ **Cannot Create Layout Effects:** CSS is not a layout or page-description language. It cannot create effects like multiple columns, frames or force the text to flow from one point to other as like HTML's `<marquee>` tag.



- ⇒ **Doesn't give absolute control over a page's appearance:** We cannot be assured that our document will look similar on different web browsers.
- ⇒ **Doesn't guarantee any kind of absolute pixel control:** We cannot be assured that a particular image or line of text will be aligned at a particular point over a background image.

Types of CSS: (CBSE: Apr – 2015 | Nov/Apr – 2014 | Oct – 2013)

There are three different types of style sheets:

- 1) *Inline Style Sheet*
- 2) *Embedded or Internal Style Sheet*
- 3) *External Style Sheet*

Inline Style Sheet: (IDOL: May – 2018 | Dec/May – 2017 | Oct – 2012)

- ⇒ In this type, style specifications are placed right within the html elements.
- ⇒ They affect using the style attribute of html element.
- ⇒ The inline style is used when you want to apply style declarations to an individual element in a particular HTML document.
- ⇒ Inline style sheet is NOT an efficient way of using CSS in two cases:
 - **CASE 1:** If you want to apply same style declaration to different elements every now and then in such cases, don't use inline style-sheet.
 - **CASE 2:** Inline style-sheet mixes the content with the presentation which is what style sheets were designed to avoid.

Syntax:

```
<H1 style>...</h1>
```

Example:

```
<html>
<head>
<title>Inline Style Sheet</title>
<meta http-equiv="content-style-type" content="text/css">
</head>
<body style="background:orange">
<h1 style="color:white; font-family:arial; font-size:14pt; text-transform:uppercase; text-align:left;"> This is an example of inline css</h1>
</body>
</html>
```

Output:

THIS IS AN EXAMPLE OF INLINE CSS

Internal/Embedded Style Sheet: (IDOL: May – 2018 | Dec/May – 2017 | Apr – 2014 | Oct – 2012)

- ⇒ It is also called Internal Style Sheet.
- ⇒ Embedded styles are styles that are embedded in the head of the document.
- ⇒ Embedded styles affect only the tags on the page they are embedded in.
- ⇒ They affect using style element (tag) of HTML.
- ⇒ While using <STYLE> tag it must include TYPE attribute.
- ⇒ TYPE attribute specifies what type of style is included in the document.

The attributes of <STYLE> element tag are:

- 1) **type:**
 - It specifies the internal type of the style language.
 - In CSS, the value of type is "text/CSS".
- 2) **media:**



- Media specifies the medium on which style sheet is applied.
- Its default value is screen.
- Its value can be screen/print/projection.

Syntax:

```
<head> <style>...</style> </head>
```

Example:

```
<html>
<head>
<title> Embedded CSS</title>
<style type="text/CSS">
body {background-color:#ccffff;}
h1 {color:purple; font-family:arial; font-size:30 px; text-transform:uppercase; text-align:left;}
</style>
</head>
<body>
<h1> This is an example of embedded CSS</h1>
<h1> B E </h1>
</body>
</html>
```

Output:

THIS IS AN EXAMPLE OF EMBEDDED CSS
B E

External Style Sheet: (CBSE: Apr – 2014) | (CBSE: Nov – 2015)

- ⇒ In external style sheets, the CSS files are kept separately from an HTML document.
- ⇒ External CSS file contains only CSS code and it is saved with a ".css" extension.
- ⇒ The main advantage of External style sheet is that external CSS is a "true separation" of style and content.
- ⇒ It is easier to reuse CSS code in any separate file.
- ⇒ The CSS file is used as an external style sheet file in HTML document by using a <LINK> tag instead of <STYLE> tag.
- ⇒ The <LINK> tag is placed in the <HEAD> section of the HTML document.

Link an HTML Page to an External Style Sheet: (CBSE: Nov – 2015)

- ⇒ The first way to add CSS style sheets to your web pages is through the <link> element that points to an external CSS file.
- ⇒ For example the following <link> shows what options you have when embedding a style sheet in your page:

```
<link href="StyleSheet.css" rel="stylesheet" type="text/css" media="screen" />
```
- ⇒ The href property points to a file within your site, just as you saw in the previous chapter when you created links between two pages.
- ⇒ The rel and type attributes tell the browser that the linked file is in fact a cascading style sheet.
- ⇒ type attribute is not used in META tag. It specifies which type of style language is used. The value of the type attribute is "text/CSS".
- ⇒ title attributes is optional. It indicates the name of the style sheet.
- ⇒ The media attribute is quite interesting: it enables you to target different devices, including the screen, printer, handheld devices, and even Braille and aural support tools for visually impaired visitors.
- ⇒ The default for the media attribute is screen, so it's OK to omit the attribute if you're targeting standard desktop browsers.

EXAMPLE:**CODE:** externalstylesheet.css

```
body { background:#ccffff;}
h2,p {
```




```

color: green;
font-family:arial;
text-align:center;
}
p i{
color: orange;
border-style: solid;
font-weight: lighter;
}
.ex{
color:purple
}

```

CODE: externalstylesheet.html

```

<html>
<head><title>Extenal style sheet</title>
<link rel= "stylesheet" type= "text/CSS" href="external.css">
</head>
<body>
<h2> It is an example of External style sheet</h2>
<p class="ex"> This is a "true separation" of style and content</p>
<p><i> External CSS </i> </p>
</body>
</html>

```

Output:



CSS SELECTORS

Q.9.

- **What is Selector in CSS? Explain various types of CSS Selectors.** [CBSSGS: Nov – 2017]
- **What are the different types of Selectors present in jQuery? Explain.** [CBSSGS: Oct – 2013]
- **Explain Class Selector and ID Selector with the help of an example.** [IDOL: Apr – 2013]

ASKED YEAR ⇒

IDOL: Dec/May – 2017 | Oct – 2016 | Apr – 2015 | Apr – 2013 CBSSGS: Nov – 2017 | Nov – 2016 | Nov/Apr – 2015 | Nov – 2014 | Oct – 2013

SOLUTION

Selector In CSS:

- ⇒ Selectors are one of the most important aspects of CSS that allow you to select and manipulate HTML elements.
- ⇒ Basically selectors are the CSS rule set that actually find or select the HTML elements you want to style based on their id, classes, attributes, types, values of attributes and much more.
- ⇒ In other words CSS selectors are the rule set that are used to select the HTML content on HTML pages so that they can be styled as we choose.

Types of CSS Selectors:

There are five types of selectors are as follows:

- 1) *Universal Selector*
- 2) *Element Type Selector / Type Selector*
- 3) *ID Selector*
- 4) *Class Selector*



5) Grouping Selector

Universal Selector:

- ⇒ The universal selector plays the role of a wild card character that selects all the elements on the page.
- ⇒ The HTML pages are made up of various tags and using this selector, all the tags are affected according to the style specified in the style block.
- ⇒ The universal selector is declared using an asterisk (*), so we can also use this selector in combination with another selector.

Syntax:

```
*{
color: green;
font-family:
arial black;
text-align: center;
}
```

Example:

```
<html>
<head>
<title>Universal selector</title>
<style>
*{
color:green;
font-family:arial black;
text-align:center;
}
</style>
</head>
<body>
<p>This selector will affect all the tags..</p>
</body>
</html>
```

Output:

This selector will affect all the tags..

Element Type Selector / Type Selector:

- ⇒ This selector is also known as a "Type Selector" that selects the elements based on the same name of elements.
- ⇒ Thus, the selector name <p> would match all the HTML <p> elements or selector name would match all the HTML elements.

Syntax:

```
p{color:green;
font-family:arial black;}

ul{list-style-type: circle;
border: solid 1px #ccc;
color: red;}
```

Example:

```
<html>
<head>
<title>Element type selector</title>
<style>
```



```
p{color:green;
font-family:arial black;}
ul{list-style-type: circle;
border: solid 1px #ccc;
color: red;}
</style>
</head>
<body>
<p>This selector will affect the tags which have the same name of elements..</p>
<ul>
<li>Animals</li>
<li>Fruits</li>
<li>Electronics</li>
</ul>
</body>
</html>
```

Output:

This selector will affect the tags which have the same name of elements..

- Animals
- Fruits
- Electronics

ID Selector:

- ⇒ The id selector is declared using the "hash" character followed by the id of the element.
- ⇒ The id selector uses the id attribute of an HTML tags to find the specific element.
- ⇒ This selector searches the id attribute that should match the id specified in the style block.
- ⇒ An id should be unique within the page so that you can use the id selector when you want to find the single and unique element.
- ⇒ The "hash" symbol is ignored during matching of the element.

Syntax:

```
#para2{color:green;
font-family:arial black;}
#list{list-style-type: circle;
border: solid 1px #ccc;
color: red;}
```

Example:

```
<html>
<head>
<title>ID selector</title>
<style>
#para1{color:blue;
font-family:arial black;}
#para2{color:green;
font-family:arial black;}
#list{list-style-type: circle;
border: solid 1px #ccc;
color: red;}
</style>
</head>
<body>
```



```
<p id="para1">This selector will only affect the tags which have the id attribute of
elements..</p>
<p id="para2">This tag is also very helpful</p>
<p>This paragraph is not affected..</p>
<h1>This is ID selector</h1>
<ul id="list">
<li>Animals</li>
<li>Fruits</li>
<li>Electronics</li>
</ul>
<ul>
<li>Boy</li>
<li>Girl</li>
</ul>
</body>
</html>
```

Output:

This selector will only affect the tags which have the id attribute of elements..

This tag is also very helpful

- Animals
- Fruits
- Electronics

Class Selector:

- ⇒ The class selector is one of the most useful selectors.
- ⇒ Its declaration is done with a dot preceding a string of one or more characters.
- ⇒ Just as is the case with an ID selector, this string of characters is defined by the developer.
- ⇒ The class selector also matches all elements on the page that have their class attribute set to the same value as the class, minus the dot.

Syntax:

```
.center{color:blue;
font-family:arial black;
text-align: center;}
.list{list-style-type: circle;
border: solid 1px #ccc;
color: red;}
```

Example:

```
<html>
<head>
<title>Class selector</title>
<style>
.center{color:blue;
font-family:arial black;
text-align: center;}
.list{list-style-type: circle;
border: solid 1px #ccc;
color: red;}
</style>
</head>
<body>
```



```
<p class="center">This selector will only affect the tags which have the class attribute of
elements..</p>
<ul class="list">
<li>Animals</li>
<li>Fruits</li>
<li>Electronics</li>
</ul>
</body>
</html>
```

Output:

This selector will only affect the tags which have the class attribute of elements..

- Animals
- Fruits
- Electronics

Grouping Selector:

⇒ If you have elements with the same style definitions, like this –

```
p{color:blue;
font-family:arial black;
text-align: center;}
h1{color:blue;
font-family:arial black;
text-align: center;}
h4{color:blue;
font-family:arial black;
text-align: center;}
```

⇒ It will be better to group the selectors, to minimize the code.

⇒ To group selectors, separate each selector with a comma.

Example:

```
<html>
<head>
<title>Grouping selector</title>
<style>
p,h1,h5{color:blue;
font-family:arial black;
text-align: center;}
</style>
</head>
<body>
<p>There are two groups of selectors..</p>
<h1>This is Grouping selector</h1>
<h5>Hello grouped selectors..!!</h5>
</body>
</html>
```

Output:

There are two groups of selectors..

This is Grouping selector

Hello grouped selectors..!!

**Q.10. How are the Basic Selector of CSS used in jQuery?**

ASKED YEAR → IDOL: Apr – 2013

SOLUTION**Basic Selectors used in jQuery:**The Universal Selector:

Just as its CSS counterpart, the universal selectors matches all elements in the page.

Example:

To set the font-family of each element in the page to Arial, the code is all follows:

```
$('.*').css('font-family', 'Arial');
```

The ID Selector:

This selector finds and retrieves an element by its id.

Example:

To set the CSS class a button called Button1, the code is as follows:

```
$('#Button1').addClass('NewClassName');
```

The CSS style NewClassName is applied to 'Button'.

The Element Selector:

This selector gets a reference to zero or more elements that match a specific tag name.

Example:

The following code turns the text color of all headings at level two or blue.

```
$('#h2').css('color', 'blue');
```

The Class Selector:

The class selector gets a reference to zero or more elements that match a specific class name.

Example:

```
<h1 class="Highlight">Heading 1</h1>
<h2>Heading 2</h2>
<p class="Highlight">First Paragraph</p>
<p>Second Paragraph</p>
```

The following jQuery code changes the background color of the first heading and the first paragraph to red, leaving the other elements unmodified.

```
$('.Highlight').css('background-color', 'red');
```

Grouped and Combined Selectors:

Grouped: Just as with CSS, one can group and combine selectors. The following grouped selector changes the text color of all h1 and h2 elements in the page.

Example:

```
$('#h1', 'h2').css('color', 'orange');
```

Combined: With a combined selector, one can find specific elements that fall within some others.

Example:

```
$('#MainContent p').css('border', '1px solid red');
```



This code changes the border only that paragraph which fall within the `MainContent` element, leaving all other paragraphs unchanged.

Q.11. How to create and use External Style Sheet using Visual Studio Developer?

ASKED YEAR →

CBSGS: Nov – 2017

SOLUTION**Steps creating a CSS Style Sheet:**

STEP 1: In Solution Explorer, right-click the name of the website, then choose Add New Item...

STEP 2: From Add New Item... Dialog Box, select SyleSheet. You can change name of the style sheet from name box. Default name is StyleSheet.css.

STEP 3: Click Add button. Style sheet has been added in the website. Add formatting elements in style sheet file.

Using Style Sheet:

```
<head runat="server">
<link rel="Stylesheet" href="styletname.css" type="text/css" />
</head>
```

Q.12. Explain working with CSS using Visual Studio Developer.

ASKED YEAR →

CBSGS: Apr – 2017

SOLUTION**Working with CSS using Visual Studio Developer:**

- ⇒ CSS contain style rules that are applied to elements in a webpage.
- ⇒ These styles define how elements are displayed & where they are positioned on the page.
- ⇒ Visual Studio provides that you use to work with CSS.
- ⇒ The Visual Web Developer Web Development tool gives complete support for creating & applying styles to the text, elements, and controls on Web Pages.

Tasks Include:

- ⇒ Setting inline styles for individual elements.
- ⇒ Creating a style block for a page.
- ⇒ Creating a cascading style sheet (.css file), and then applying the ".css" file to one or more pages in our site.
- ⇒ Changing style sheet references programmatically.

Steps to set inline styles in Design View:

STEP 1: Switch Design View.

STEP 2: Right-Click caption-label & then click style. The style Builder dialog box appears. The style Builder helps us to set style confirmation by organizing style information into logical categories & offering us appropriate values and options for each style setting.

STEP 3: Click font, click family, then click ellipsis (...) that is located to the right of the textbox. Select Arial as selected fonts. Size 1.2.

STEP 4: Click Background, click the ellipsis (...) that is located to the right of the color box.

STEP 5: Click OK to close the style Builder dialog box.



CSS Vs. HTML

Q.13. Compare CSS with HTML.

ASKED YEAR →

CBSGS: Apr – 2016

SOLUTION

CSS Vs. HTML:

- ⇒ HTML is a standard language to create web page CSS is a style-sheet document to be applied any HTML document.
- ⇒ HTML has few keywords used for formatting and hence cannot be useful for bigger pages CSS was developed to simplify code in much bigger pages.
- ⇒ CSS creates proper customer tags that can define the font, size, color margins.
- ⇒ HTML is very simple to learn and write where as CSS is little difficult.
- ⇒ CSS makes a web page creation a lot easier than HTML and lot easier to do and trouble shoot.

Q.14. How does CSS fix HTML formatting problems?

ASKED YEAR →

IDOL: Apr – 2013

SOLUTION

CSS Fix HTML Formatting Problems:

- ⇒ CSS offers a rich set of options to change every little aspect of the web site.
- ⇒ CSS is understood by all the browsers and so is the language for visual presentation of web pages.
- ⇒ It overcomes the problem of mixed data and presentation by enabling the programmer to define all formatting information in external files.
- ⇒ Another benefit of separate style sheet is the decrease in bandwidth that is required for the site. Style sheets don't change with the each request, so a browser saves a local copy of the style sheet the first time to download it.

<LINK> TAG

Q.15. Explain <LINK> Tag with example.

ASKED YEAR →

CBSGS: Apr – 2015

SOLUTION

<LINK> Tag:

- ⇒ The <LINK> tag defines a link between a document and an external resource.
- ⇒ The <LINK> tag is used to link to external style sheets.
- ⇒ <LINK> tag is used to connect to external Cascading Style Sheet File.
- ⇒ The styles which are mentioned in external css file, can applied to the complete web site.
- ⇒ External CSS file has to be created in where the styles are written.
- ⇒ <LINK> tag should be written in the tag of other web pages in which this external styles have to be applied.

Syntax:

```
<link rel="stylesheet" type="text/css" href="default.css" />
```

Where,

- **rel**: can be used to specify the relationship of the target of the link to the current page.
type (content-type)
- **type**: This attribute Provides information about the content type of the destination resource, telling whether it's an HTML document, a JPG image, an Excel document, etc.
- **href**: The "href" attribute specifies the destination resource, which the element is linking to. It may specify a resource in the same website or in an external one.

Example:

**File: style.css**

```
p
{
background-color:yellow;
letter-spacing: 10;
line-height: 6;
}
```

File: Webpage.html

```
<html>
<head>
<link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
<p>
This is a paragraph.
</p>
</body>
```

Now this paragraph will have the styles which are mentioned in "style.css" file.

USER CONTROLS

Q.16.➤ **What are User Controls?** [IDOL: Apr – 2013]➤ **How do we create a User Control in ASP.NET?** [IDOL: May – 2018 | May – 2016 | Apr – 2014]

ASKED YEAR →

IDOL: May – 2018 | May – 2016 | Apr – 2014 | Apr – 2013

SOLUTION

User Controls:

- ⇒ User controls behaves like miniature ASP.NET pages or web forms, which could be used by many other pages.
- ⇒ These are derived from the `System.Web.UI.UserControl` class.

Characteristics of User Controls:

- They have an .ascx extension.
- They may not contain any `<html>`, `<body>` OR `<form>` tags.
- They have a Control directive instead of a Page directive.

Creating A User Control In ASP.NET:

- ⇒ User control declarative syntax is very similar to syntax used to create an ASP.NET Web page.
- ⇒ The primary differences are that the user controls use an @ Control Directive in place of an @ Page directive, and that the user controls do not include the html, body, and form elements around the content.

Steps in Creating User Control:

STEP 1: Create a new file and give it a name with the extension ".ascx".

For example, name your user control "DisplayName.ascx".

STEP 2: Create an @ Control directive at the top of the page and specify the programming language you want to use for the control (if any).

STEP 3: Add controls that you want the user control to display.

STEP 4: Add code for the tasks that the user control will perform, such as handling control events or reading data from a data source.



STEP 5: Create properties in the control if you want to be able to share information between the user control and the hosting page. You create properties as you would for any class, either as public members or with get and set accessors.

Example:

The following example shows an ASP.NET Web page that contains a user control. The user control is in the file "Spinner.ascx" in the Controls folder. In the page, the control is registered to use the prefix uc and the tag name Spinner. The user control properties MinValue and MaxValue are set declaratively.

```
<%@ Page Language="C#" %>
<%@ Register TagPrefix="uc" TagName="Spinner" Src="~/Controls/Spinner.ascx" %>
<html>
<body>
<form runat="server">
<uc:Spinner id="Spinner1" runat="server" MinValue="1" MaxValue="10" />
</form>
</body>
```

Q.17.

What are the different ways to redirect a user to another page programmatically? What's the difference between them?

ASKED YEAR → IDOL: May – 2018

SOLUTION

Redirect A User To Another Page By Using The Browser:

STEP 1: Set the Response object's BufferOutput property to true.

STEP 2: Call the Response object's Redirect method, passing it the URL of the page to which you want to redirect users.

STEP 3: The following code example shows how to redirect a page based on the contents of a local variable, UserLanguage, which is set elsewhere.

Example:

```
Response.BufferOutput = true;
if (UserLanguage == "English")
{
Response.Redirect("http://www.microsoft.com/gohere/look.htm");
}
else if (UserLanguage == "Deutsch")
{
Response.Redirect("http://www.microsoft.com/gohere/look_deu.htm");
}
else if (UserLanguage == "Español")
{
Response.Redirect("http://www.microsoft.com/gohere/look_esp.htm");
}
```

Redirect Users To Another Page By Using A Server-Side Method:

STEP 1: Call the Transfer method, passing it the name of the page to which you want to redirect users.

STEP 2: The following code example shows how to redirect to another page.

Example:

```
protected void Button1_Click(object sender, System.EventArgs e)
{
Server.Transfer("Page2.aspx", true);
}
```



LISTVIEW CONTROL

Q.18.

- Explain the ListView Control. [IDOL: Oct – 2016 | Apr – 2013] | [CBSGS: Apr – 2017 | Nov – 2014]
- List and explain various templates provided by ListView Control. [CBSGS: Apr – 2014]
- Explain the basic Attributes of ListView Control. [IDOL: Oct – 2016]

ASKED YEAR ⇒

IDOL: Oct – 2016 | Apr – 2013

CBSGS: Apr – 2017 | Nov/Apr – 2014

SOLUTION

ListView Control:

- ⇒ It display data from a data source using templates.
- ⇒ Like other controls, it is also used to Edit, Delete and Insert Data.
- ⇒ The template elements define the formatting that is used to display data. These templates are generates automatically when you configure a ListView Control.
- ⇒ The Layout Template defines the overall layout of the control.
- ⇒ To enable paging, a DataPager control is added to the ListView Control.

Basic Attributes of ListView Control:

- **ID:** The ID of the control.
- **Runat:** Must specify "server".
- **DataSourceID:** The ID of the data source to bind to.
- **DataKeyNames:** The names of the primary key fields separated by commas.
- **InsertItemPosition:** The location within the ListView Control where the InsertItem template is rendered. We can specify FirstItem, LastItem, or None.

Various Templates provided by ListView Control:

- **LayoutTemplate:** It defines the basic layout of the control.
- **ItemTemplate:** The template used for each item in the Data Source.
- **ItemSeparatorTemplate:** The template used to separate items in the Data Source.
- **AlternatingItemTemplate:** The template used for alternating items in the data source.
- **EditItemTemplate:** The template used when a row is being edited.
- **InsertItemTemplate:** The template used for inserting a row.
- **EmptyDataTemplate:** The template used when the data source is empty.
- **SelectedItemTemplate:** The template used when a row is selected.
- **GroupTemplate:** The template used to define a group layout.
- **GroupSeparatorTemplate:** The template used to separate groups of items.
- **EmptyItemTemplate:** The template used for empty items in a group.

**HTML SERVER CONTROLS VS. WEB SERVER/ASP.NET CONTROLS****Q.19. Differentiate between HTML Server Controls and Web Server Controls.**

ASKED YEAR

CBSGS: Apr – 2017

SOLUTION**HTML Server Controls Vs. Web Server/ASP.Net Controls:**

<u>HTML Server Controls</u>	<u>Web Server Controls</u>
HTML control runs at client side.	ASP.Net controls run at server side.
We can run HTML controls at server side by adding attribute runat="server".	We cannot run ASP.Net Controls on client side as these controls have this attribute runat="server" by default.
HTML controls are client side controls, so it does not provide STATE management.	ASP.Net Controls are Server side controls, provides STATE management.
HTML control does not require rendering.	ASP.Net controls require rendering.
As HTML controls runs on client side, execution is fast.	As ASP.Net controls run on server side, execution is slow.
HTML controls do not have any separate class for its controls.	ASP.Net controls have separate class for its each control.
HTML controls does not support Object Oriented paradigm.	With ASP.Net controls, you have full support of Object oriented paradigm.
HTML controls cannot be accessed form code behind files.	ASP.Net controls can be directly worked and accessed from code behind files.
HTML control have limited set of properties and/or methods.	ASP.Net controls have rich set of properties and/or methods.

WEB USER CONTROL VS. WEB SERVER CONTROL**Q.20. What is the difference between Server Controls and User Controls?**

ASKED YEAR ⇒

IDOL: Apr – 2015

SOLUTION**Web User Control Vs. Web Server Control:**

<u>Web User Control</u>	<u>Web Server Control</u>
User Control is a page file with extension .ascx which can only be used within single application.	Custom Control are assemblies (.dll files) that can be used in multiple application.
User Control cannot be added to the toolbox of visual studio.NET. To use a user control with in an aspx page you have to drag the user control from the solution explorer to designer Page.	It can be added to toolbox of VS.NET.
User Control can be viewed as a sort of generic controls during the design time. The proper GUI of user controls can be viewed only during the run time.	It can be viewed during the design time.

Web User Control:

- ⇒ Web user controls are easy to make, but they can be less convenient to use.
- ⇒ Web user controls are compiled dynamically at run time they cannot be added to the Toolbox.
- ⇒ The only way to share the user control between applications is to put a separate copy in each application, which takes more maintenance if you make changes to the control.
- ⇒ If our control has a lot of static layout, a user control might make sense.

Web Custom Control:

- ⇒ Web custom controls are compiled code, which makes them easier to use but more difficult to create.



- ⇒ Once we have created the control, however, we can add it to the Toolbox and display it in a visual designer with full Properties window support and all the other design-time features of ASP.NET server controls.
- ⇒ In addition, we can install a single copy of the Web custom control in the global assembly cache and share it between applications, which makes maintenance easier.

PROPERTIES OF WEB SERVER CONTROLS

Q.21. Explain any five Common properties of Web Server Controls.

ASKED YEAR ⇒

CBSGS: Apr – 2016 | Nov – 2015

SOLUTION

Common property of Web Server Controls:

- **Border Color**: This property is used to change the border color as a control.
- **CSS Class**: It enables us to set the style sheet class for this control.
- **Font**: It enables us to change font settings.
- **Enabled**: It determines whether the control is enabled or not. If control is disabled user cannot interact with it.
- **Visible**: It determines whether the control is visible or not.
- **ID** – Identifier for the control.
- **Enabled** – Indicates whether the control is grayed out.
- **ForeColor** – Sets the foreground color for the control.
- **BackColor** – Sets the background color for the control.
- **Visible** – It indicates whether a server control is visible.

STATE MANAGEMENT

Q.22.

- Write a short note on State Management. [IDOL: Oct/May – 2016 | Apr – 2013] | [CBSGS: Nov – 2016]
- Write a short note on Server Side State Management. [IDOL: May – 2017]
- Explain Sessions in ASP.NET. [IDOL: Apr – 2013]
- What are four application Events when an Application is executed? [IDOL: Apr – 2015]

ASKED YEAR ⇒

IDOL: May – 2017 | Oct/May – 2016 | April – 2013 | Oct – 2012

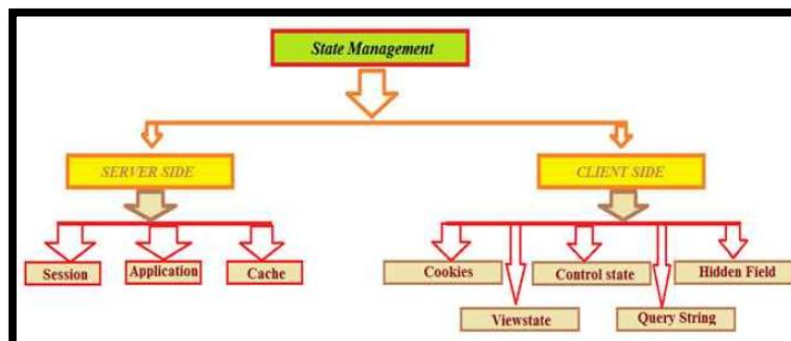
CBSGS: Nov – 2016

SOLUTION

State Management:

- ⇒ ASP.NET is based on the stateless HTTP protocol.
- ⇒ So each request from the client browser to the web server is understood as an independent request.
- ⇒ State management is a technique used to maintain the state information for ASP.NET web pages across multiple requests.
- ⇒ ASP.NET comes with built in support for state management, but at the server and client ends.

Types of State Management:





Server-Side State Management:

- ⇒ State Management is the technique that is used to maintain user and page information over multiple requests while browsing the web.
- ⇒ There are three types of Server-Side State Management:
 - 1) *Application*
 - 2) *Session*
 - 3) *Cache*

Application State:

- ⇒ Application State is a server side management state.
- ⇒ It is also called application level state management.
- ⇒ ASP.NET allows you to save values using application state, a global storage mechanism that is accessible from all pages in the Web application.
- ⇒ Application state is stored in the Application key/value dictionary.
- ⇒ Once we add our application-specific information to application state, the server manages it, and it is never exposed to the client.
- ⇒ Application state is a great place to store information that is not user-specific.
- ⇒ By storing it in the application state, all pages can access data from a single location in memory, rather than keeping separate copies of the data.
- ⇒ Data stored in the Application object is not permanent and is lost any time the application is restarted.

Application Events when an Application is executed:

1. **Application_Start:** This event occurs when the first user visits a page of the application.
2. **Application_End:** This event occurs when there are no more users of the application.
3. **Application_BeginRequest:** This occurs at the beginning of each request to the server.
4. **Application_EndRequest:** occurs at the end of each request to the server.

Application Start: This event Start with domain start.

```
Void Application_Start(object sender, EventArgs e)
{
Application["AppstartMessage"] = "Welcome to CSharp Corner Developer Communtiy";
}
```

Application Error: In this section manage unhandled exception error.

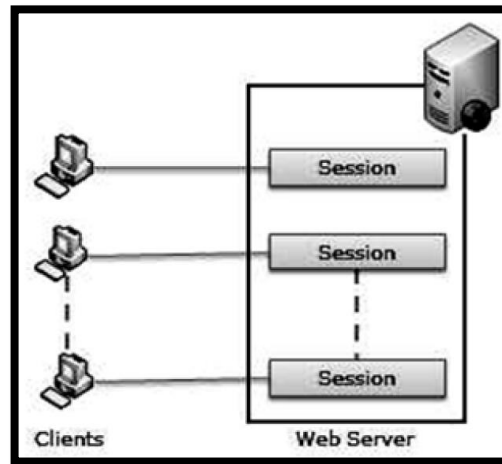
```
void Application_Error(object sender, EventArgs e)
{
// Write an unhandled error code exception
}
```

Application End: This ends with domain or restarts IIS.

```
Void Application_End(object sender, EventArgs e)
{
Application["AppEndMessage"] = "Application Closed";
}
```

Session State:

- ⇒ ASP.NET uses session state to track the state of each user of an application.
- ⇒ To do this, it creates a session state object that contains a session ID.
- ⇒ This ID is passes to the browser then back to the server with the text request so the server can identify the session associated with that request.
- ⇒ To work with the data in the session state, the HttpSessionState class is used which defines a collection of session state items.
- ⇒ To access the session state object. Session property of the Page is used.



Attributes of Session State element in the web.config file:

- **Mode:** It specifies where to store the session state. It has four options: off, InProc, StateServer and SQLServer
 - **off** – Disables session state management.
 - **InProc** – Session state is stored locally in memory of ASP.NET worker process.
 - **StateServer** – Session State is stored outside ASP.NET worker process and is managed by Windows Service (ASP.NET State Service). Location of this service is specified by stateConnectionString attribute.
 - **SQLServer** – Session State is stored outside ASP.NET worker process in SQL Server database. Location of this database is represented by sqlConnectionString attribute.
- **Cookieless:** Specifies whether cookieless sessions are used.
- **Timeout:** The number of minutes the session should be maintained without any use activity. The default is 20.
- **stateConnectionString:** The server name or IP Address and port number (24242) of the server that runs the ASP.NET state service.

Configuring Session Timeout:

It indicates the time in minutes that the session can be idle before it is abandoned. Default value is 20 mins.

```
(web.config)
<configuration>
<sessionState mode="InProc" cookieless="false" timeout="40"/>
</configuration>
```

To clear the session:

- **Session.Abandon()** – If method is called, Session_End event gets fired.
- **Session.Clear()** – Just clears the session data without killing the session.

Enabling and Disabling Session State:

- ⇒ Page-Level Session state – <%@Page EnableSessionState="False">
- ⇒ To disable the session state for page that do not require access to session data by using the EnableSessionState to false in the page directive of that page.

Application Level Session State - in web.config

```
<configuration>
<system.web>
<pages enableSessionState="false">
</system.web>
</configuration>
```



Advantages And Disadvantages Of Session State:

Advantages :

- It helps maintain user state and data all over the application.
- It is easy to implement and we can store any kind of object.
- Stores client data separately.
- Session is secure and transparent from the user.

Disadvantages:

- Performance overhead in case of large volumes of data/user, because session data is stored in server memory.

WORKING OF SESSION STATE

Q.23. Explain the work of Session State in ASP.NET.

ASKED YEAR → IDQI: May – 2018 | May – 2016 | Oct – 2012

CBSGS: Oct – 2013

SOLUTION

Working of Session State:

- ⇒ ASP.NET uses session state to track the state of each user of an application.
- ⇒ To do this, it creates a session state object that contains a session ID.
- ⇒ This ID is passed to the browser then back to the server with the next request so the server can identify the session associated with that request.
- ⇒ To work with the data in the session state, the `HttpSessionState` class is used which defines a collection of session state items.
- ⇒ To access the session state object, session property of the page is used.
- ⇒ It helps maintain user state and data all over the application.
- ⇒ It is easy to implement and we can store any kind of object.
- ⇒ Stores client data separately.
- ⇒ Session is secure and transparent from the user.

Example:

```
//Storing UserName in Session
Session["UserName"]=txtUser.Text;
//how to retrieve values from session:
//check whether session variable null or not
if (Session["UserName"]!=null)
{
//Retrieving UserName from Session
lblWelcome.Text="Welcome:"+Session["UserName"];
}
else
{
//Do Something else
}
```

**SESSION STATE VARIABLES****Q.24.** Write short descriptions on Session State Variables.

ASKED YEAR →

CBSGS: Nov – 2015

SOLUTION**Session State Variable:**

- ⇒ When a user connects to an ASP.NET website a new session object is created.
- ⇒ When session state is turned on, a new session state object is created for each new request.
- ⇒ This object becomes part of the context and it is available through the web page.
- ⇒ Sessions are identified and tracked with a session id which is passed from client to server.

Example:

To create a session variable:

```
Session ["var"]=value;
```

To retrieve a session variable value:

```
Lblresult.text=session ["var"];
```

APPLICATION STATE VS. SESSION STATE**Q.25.** What is the difference between Application State and Session State in ASP.NET?

ASKED YEAR →

IDOL: Apr – 2014

SOLUTION**Application State Vs. Session State:**

<u>Session State</u>	<u>Application State</u>
Session state is user and browser specific.	Application state is application specific.
Session state can be stored in memory on the server as well as client's cookies.	Application state is stored only in the memory on the server.
If client has disabled cookies in his browser then session state will be stored in URL.	Application state does not track client's cookies or URL.
Session state has scope to the current browser only.	Application state has no scope to the current browser.
If we change the browser session id is changed.	If we change the browser application id remains same.

**VIEWSTATE****Q.26.**

- **What is ViewState? How it works in ASP.NET?** [IDOL: May – 2018 | Oct – 2012] | [CBSGS: Apr – 2017 | Nov – 2014 | Oct – 2013]
- **What is ViewState? Where is the ViewState stored after the page postback?** [CBSGS: Apr – 2017]
- **List the different places in a Web Application where ViewState Field can be Disabled?** [IDOL: Apr – 2013]
- **What is the significance of ViewState in ASP.NET?** [IDOL: Apr – 2014]
- **What is View State? How to preserve View State for user's own data.** [IDOL: Apr – 2015]

ASKED YEAR ⇒

IDOL: May – 2018 | Apr – 2015 | Apr – 2014 | Apr – 2013 | Oct – 2012

CBSGS: Nov/Apr – 2017 | Nov – 2014 | Oct – 2013

SOLUTION**ViewState:**

- ⇒ ViewState is an ASP.NET feature that provides for retaining the values of page and control properties that change from one execution of a page to another.
- ⇒ ViewState does not hold the page's controls, it holds the control IDs and the corresponding values that would otherwise have been lost due to a postback to the web server.
- ⇒ ViewState represents the state of a page when it was last processed on the web server.
- ⇒ ViewState is a property of all server controls and is stored as a key-value pair using the `System.Web.UI.StateBag` object.

Working of ViewState:

- ⇒ It works with the primitive types.
- ⇒ Before ASP.NET sends a page back to the client, it determines what changes the program has made to the properties of the page and its controls. These changes are encoded into a string that is assigned to the value of a hidden input field name `__ViewState`.
- ⇒ When the page is posted back to the server, the `__ViewState` field is sent back to the server along with the HTTP request. Then ASP.NET retrieves the property values from the `__ViewState` field and uses them to restore the page and control properties.

Enabling ViewState at a Page Level:

```
<%@ Page EnableViewstate="False/true"%>
// Enabling viewstate at control level
<asp:Label id="label1" runat="server" EnableViewState="false">
// Enabling viewstate at an Application level - in web.config
<pages enableViewState="false/true">
```

Example:

Here is a simple example of using the ViewState to carry values between postbacks:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_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>ViewState</title>
</head>
<body>
<form id="form1" runat="server">
<asp:TextBox runat="server" id="NameField" />
<asp:Button runat="server" id="SubmitForm" onclick="SubmitForm_Click" text="Submit & set name" />
<asp:Button runat="server" id="RefreshPage" text="Just submit" />
<br /><br />
Name retrieved from ViewState: <asp:Label runat="server" id="NameLabel" />
</form>
</body>
```



```
</html>
```

ViewState is stored after PagePostBack:

- ⇒ When a page is sent back to the client for output, the changes in the properties of the page and its controls are determined and stored in the value of a hidden input field name `_VIEWSTATE`.
- ⇒ When the page is again post back `_VIEWSTATE` field is sent to the server with the HTTP request.

Significance of ViewState in ASP.NET:

- ⇒ ViewState is one of the most important and useful client side state management mechanisms.
- ⇒ It can store the page value at the time of post back (Sending and Receiving information from Server) of your page.
- ⇒ It is easy to implement.
- ⇒ No server resources are required.
- ⇒ Enhanced security features, like it can be encoded and compressed.
- ⇒ ASP.NET pages provide the ViewState property as a built-in structure for automatically storing values between multiple requests for the same page.
- ⇒ ViewState use Hidden field to store its information in an encoding format.

ViewState Field can be disabled in the Web Application:

ViewState Field can be disabled at the following places in the Web Application:

- 1) Page
- 2) Control
- 3) Application
- 4) Machine

Page:

```
<%@Page.EnableViewState="false"/>
```

Control:

```
<asp:TextBox id="Name" runat="server" EnableViewState="false"/>
```

For Application:

```
<Page enableViewState="false"/>
```

For Machine:

```
<Page enableViewState="false" enableViewStateMac="false"/>
```

Securing ViewState:

- ⇒ By default, view state data is stored in the page in a hidden field and is encoded using base64 encoding.
- ⇒ In addition, a hash of the view state data is created from the data by using a machine authentication code (MAC) key.
- ⇒ The hash value is added to the encoded view state data and the resulting string is stored in the page.
- ⇒ When the page is posted back to the server, the ASP.NET page framework re-computes the hash value and compares it with the value stored in view state.
- ⇒ If the hash values do not match, an exception is raised that indicates that view state data might be invalid.
- ⇒ By creating a hash value, the ASP.NET page framework can test whether the view state data has been corrupted or tampered with.
- ⇒ However, even if it is not tampered with, view state data can still be intercepted and read by malicious users.



COOKIES

Q.27.

- Explain in brief about Cookies. [CBSGS: Nov – 2017]
- Explain the role of Cookies in ASP.NET. [IDOL: Dec – 2017 | Apr – 2014]
- What are the advantages and disadvantages of using Cookies? Explain how server sets a Cookie and retrieves it. [CBSGS: Nov – 2015]

ASKED YEAR ⇒

IDOL: Dec – 2017 | Apr – 2015 | Apr – 2014

CBSGS: Nov – 2017 | Nov – 2015

SOLUTION

Cookies in ASP.NET: (CBSGS: Nov – 2017)

- ⇒ Cookie is a small amount of data that server creates on the client.
- ⇒ When a web server creates a cookie, an additional HTTP header is sent to the browser when a page is served to the browser.
- ⇒ There are two type of Cookies as follows:
 - **Persist Cookie:** A cookie has not have expired time which is called as Persist Cookie.
 - **Non-Persist Cookie:** A cookie has expired time which is called as Non-Persist Cookie.

Roles of Cookies: (IDOL: Dec – 2017 | Apr – 2014)

- ⇒ Common use of cookies is to remember users between visits.
- ⇒ Practically, cookie is a small text file sent by web server and saved by web browser on client machine.
- ⇒ Cookies may be used for authentication, identification of a user session, user's preferences, shopping cart contents, or anything else that can be accomplished through storing text data.
- ⇒ Cookies can also be used for travelling of data from one page to another.

How Server Sets a Cookie and Retrieves it: (CBSGS: Nov – 2015)

Syntax for Creating Cookie:

- **Syntax:**

```
Response.Cookies["cookiename"]:value="some value";
```

- **Example:**

```
Response.Cookies["uname"]:value="text";
```

Syntax for Reading Cookie:

- **Syntax:**

```
string str=Request.Cookies["cookiename"].Value;
```

- **Example:**

```
string str=Request.Cookies["userName"].Value;
```

Common Property of Cookies:

- **Domain:** It is used to associate cookies to domain.
- **Secure:** It can enable secure cookie to set true (HTTPS).
- **Value:** It can manipulate individual cookie.
- **Values:** It can manipulate cookies with key/value pair.
- **Expires:** It is used to set expire date for the cookies.

Advantages and Disadvantages of Cookies: (CBSGS: Nov – 2015)

Advantages:

- Its clear text so user can able to read it.
- We can store user preference information on the client machine.
- Its easy way to maintain.
- Fast accessing.

**Disadvantages:**

- If user clear cookie information we can't get it back.
- No security.
- Each request will have cookie information with page.
- Users can edit or delete cookies files.
- It stores data in text files and hence not reliable in storing sensitive data.
- It can store only small amount of data.

Q.28. Explain various properties of HttpCookie Class.

ASKED YEAR →

CBSGS: Nov – 2017

SOLUTION**Properties of the HttpCookie Class:**

⇒ **Domain:** The Domain property is a string containing the domain of the cookie.

Syntax:

```
String Domain
```

Example:

```
Response.Cookies["myNameCookie"].Domain = "triconsole.com";
```

⇒ **Expires:** The Expires property is used to specify a date at which a cookie becomes invalid.

Syntax:

```
DateTime Expires
```

Example:

```
DateTime toExpire = new DateTime(2006, 12, 1);  
Response.Cookies["myNewCookie"].Expires = toExpire;
```

⇒ **HasKeys:** The HasKeys property is True if the cookie has subkeys.

Syntax:

```
Boolean HasKeys
```

Example:

```
Request.Cookies["myNewCookie"].HasKeys;
```

⇒ **Name:** The Name property contains the name of the cookie.

Syntax:

```
String Name
```

Example:

```
string cookieName = Request.Cookies["myNewCookie"].Name;
```

⇒ **Path:** The Path property contains the virtual path associated with the current cookie.

Syntax:

```
String Path
```

Example:

```
string cookiePath = Request.Cookies["myNewCookie"].Path;
```

⇒ **Secure:** The Secure property is True if the cookie should be sent only over a secure connection.

Syntax:

```
Boolean Secure
```

Example:

```
Boolean transmitSecure = Request.Cookies["myNewCookie"].Secure;
```

⇒ **Value:** The Value property contains the actual text value of the cookie.

Syntax:



```
String Value
```

▪ **Example:**

```
Response.Cookies["myNewCookie"].Value = "Test Value";
```

⇒ **Values:** The Values property is a collection of names-value pairs that enable a cookie to contain subvalues.

▪ **Syntax:**

```
NameValueCollection Values
```

▪ **Example:**

```
Response.Cookies["myNewCookie"].Values["SubCookie1"] = "value1";
Response.Cookies["myNewCookie"].Values["SubCookie2"] = "value2";
string myResponse = Request.Cookies["myNewCookie"].Values["SubCookie1"] + " ";
myResponse += Request.Cookies["myNewCookie"].Values["SubCookie2"];
```

QUERY STRING

Q.29.

➤ What is use **QueryString**? Explain Encoding and Decoding of **QueryString**. [CBSSGS: Apr – 2014]

➤ Explain **QueryString** with example. [CBSSGS: Nov – 2016]

ASKED YEAR ⇒

CBSSGS: Nov – 2016 | Apr – 2014

SOLUTION

QueryString:

- ⇒ A QueryString 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.
- ⇒ When we need to pass content between the HTML pages or aspx Web Forms in the context of ASP.NET, a Query String is very easy to use and the Query String follows a separating character, usually a Question Mark (?).
- ⇒ It is basically used for identifying data appearing after this separating symbol.
- ⇒ A Query String Collection is used to retrieve the variable values in the HTTP 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.
- ⇒ A Query String collection is a parsed version of the QUERY_STRING variable in the Server Variables collection.
- ⇒ It enable us to retrieve the QUERY_STRING variable by name.
- ⇒ When we use parameters with Request.QueryString, the server parses the parameters sent to the request and returns the effective or specified data.

Syntax of Query String:

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

Creating Query String:

We can create a new writeable instance of `HttpValueCollection` by calling `System.Web.HttpUtility.ParseQueryString(string.Empty)`.

```
NameValueCollection queryString = System.Web.HttpUtility.ParseQueryString(string.Empty);
queryString["param1"] = "paramValue1";
queryString["param2"] = "paramValue2";
```

Retrieving Query String:

The `QueryString` collection retrieves the values of the variables in the HTTP query string and it is specified by the values following the ? (question mark).

```
protected void Page_Load(object sender, EventArgs e)
{
```



```
string param1 = Request.QueryString["param1"];
string param2 = Request.QueryString["param2"];
}
```

Encoding and Decoding of QueryString:

- ⇒ The `HtmlEncode()` method is particularly useful if we are retrieving values from a database and we aren't sure if the text is valid HTML.
- ⇒ We can use the `HtmlDecode()` method to revert the text to its normal form if we need to perform additional operations or comparisons with it in your code.
- ⇒ The `UrlEncode()` method changes text into a form that can be used in a URL, escaping spaces and other special characters.
- ⇒ This step is usually performed with information you want to add to the query string.

```
Label1.Text = Server.HtmlEncode("To bold text use the <b> tag.");
```

Q.30. What is the need of Query String in ASP.NET?

ASKED YEAR ⇒ IDOL: Apr – 2014

SOLUTION

Need of Query String in ASP.NET:

- ⇒ This is the most simple and efficient way of maintaining information across requests.
- ⇒ The information you want to maintain will be sent along with the URL.
- ⇒ A typical URL with a query string looks like:

```
www.somewebsite.com/search.aspx?query=foo
```

- ⇒ The URL part which comes after the ? symbol is called a QueryString.
- ⇒ QueryString has two parts, a key and a value. In the above example, query is the key and foo is its value.
- ⇒ You can send multiple values through QueryString, separated by the & symbol.
- ⇒ The following code shows sending multiple values to the "foo.aspx" page:

```
Response.Redirect("foo.aspx?id=1&name=foo");
```

- ⇒ The "foo.aspx" page will get the values like in the following table:

Key	Value
id	1
name	foo

- ⇒ The following code shows reading the QueryString values in foo.aspx

```
string id = Request.QueryString["id"];
string name = Request.QueryString["name"];
```

- ⇒ If you try to get a value which is not in the QueryString collection, you will get a NULL reference

Pros and Cons:

- ⇒ Query string is lightweight and will not consume any server resources.
- ⇒ It is very easy to use and it is the most efficient state management technique. However, it has many disadvantages.
- ⇒ You can pass information only as a string.
- ⇒ URL length has limitations. So you can't send much information through URL.
- ⇒ Information passed is clearly visible to everyone and can be easily altered.



HIDDEN FIELD

Q.31. Explain HiddenField Control with example.

ASKED YEAR →

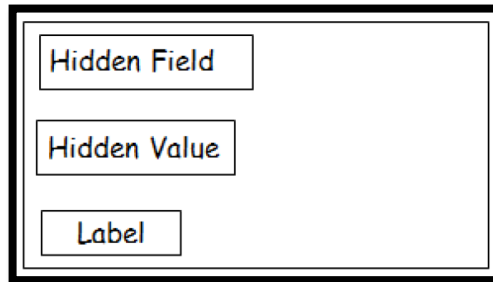
CBSGS: Apr – 2016

SOLUTION

HiddenField:

- ⇒ HiddenField Control is used to store a value that needs to be persisted across posts to the server.
- ⇒ Normally View State, Session State and Cookies are used to maintain the state of web forms page.
- ⇒ However, if these methods are disabled or are not available, you can use the Hidden field control to store state values.
- ⇒ To specify the value for HiddenField Control, use the value property.
- ⇒ You can provide a routine that gets called everytime the value of HiddenField Control changes between posts to the server by creating an even handdler for the ValueChanged Event.

Example: In this example when you click the Hidden value Button the label displayed a value. When you click the button again & again the value will decrease by 1.



HiddenField.aspx

```
<asp:content ID="C1" ContentPlaceHolderID="head" runat="server">
</asp:content>
<asp:content ID="C2" contentPlaceHolderID="C3" runat="server">
<div>
<asp:HiddenFiled ID="h1" runat="server"/>
<br/>
<asp:Button ID="button" runat="server" text="Hidden value" onclick="Button_click"/>
<asp:Label ID="Label" runat="server" font-bold="True"/>
</div>
</asp:content>
```

HiddenField.aspx.cs

```
protected void button click(object sender, EventArgs e)
{
if (h1.value==string.Empty)
h1.value="101";
h1.value=convert.ToInd32(h1.value) - D.Tostring;
label.Text=h1.value;
}
```

GLOBAL.ASAX FILES**Q.32.**

- Explain the global.asax files in ASP.NET Application. [IDOL: May – 2017 | Apr – 2015 | Apr – 2014] | [CBSGS: Apr – 2014]
- List and explain the major events in global.asax file. [CBSGS: Nov – 2017]
- Explain the use of global.asax files in ASP.NET Application. [IDOL: Dec – 2017 | Oct – 2012]

ASKED YEAR ⇒

IDOL: Dec/May – 2017 | Apr – 2015 | Apr – 2014 | Oct – 2012

CBSGS: Nov – 2017 | Apr – 2014

SOLUTIONglobal.asax File:

- ⇒ The global.asax file, also known as the ASP.NET Application File, is an optional file that contains code for responding to application-level events raised by ASP.NET or by HttpModules.
- ⇒ The global.asax file resides in the root directory of an ASP.NET-based application.
- ⇒ The global.asax file is parsed and dynamically compiled by ASP.NET.
- ⇒ The global.asax file itself is configured so that any direct URL request for it is automatically rejected; external users cannot download or view the code written within it.
- ⇒ The global.asax file does not need recompilation if no changes have been made to it.
- ⇒ There can be only one global.asax file per application and it should be located in the application's root directory only.

Types of global.asax events:

There are two types of global.asax events as follows below:

- 1) Events which are fired for every request
- 2) Events which are not fired for every request

Events which are fired for every request:

- **Application_BeginRequest()** – This event raised at the start of every request for the web application.
- **Application_AuthenticateRequest** – This event rose just before the user credentials are authenticated. We can specify our own authentication logic here to provide custom authentication.
- **Application_AuthorizeRequest()** – This event raised after successful completion of authentication with user's credentials. This event is used to determine user permissions. You can use this method to give authorization rights to user.
- **Application_ResolveRequestCache()** – This event raised after completion of an authorization request and this event used in conjunction with output caching. With output caching, the rendered HTML of a page is reused without executing its code.
- **Application_AcquireRequestState()** – This event raised just before session-specific data is retrieved for the client and is used to populate Session Collection for current request.
- **Application_PreRequestHandlerExecute()** – This event called before the appropriate HTTP handler executes the request.
- **Application_PostRequestHandlerExecute()** – This event called just after the request is handled by its appropriate HTTP handler.
- **Application_ReleaseRequestState()** – This event raised when session specific information is about to serialized from the session collection.
- **Application_UpdateRequestCache()** – This event raised just before information is added to output cache of the page.
- **Application_EndRequest()** – This event raised at the end of each request right before the objects released.

Events which are not fired for every request:

- **Application_Start()** – This event raised when the application starts up and application domain is created.
- **Session_Start()** – This event raised for each time a new session begins, This is a good place to put code that is session-specific.
- **Application_Error()** – This event raised whenever an unhandled exception occurs in the application. This provides an opportunity to implement generic application-wide error handling.
- **Session_End()** – This event called when session of user ends.
- **Application_End()** – This event raised just before when web application ends.



- **Application_Disposed()** – This event fired after the web application is destroyed and this event is used to reclaim the memory it occupies.

Major Events in global.asax File:

- Application_Init()
- Application_Disposed()
- Application_Error()
- Application_Start()
- Application_End()
- Application_BeginRequest()
- Application_EndRequest()
- Session_Start()
- Session_End()
- Application_AuthorizationRequest()

Q.33.**What are the Event Handlers that we can have in global.asax file?**

ASKED YEAR →

CBSGS: Apr – 2017

SOLUTION

Event Handlers in global.asax File:

The purpose of these event handlers in the Global.asax file is given below:

- **Application_Init:** The Application_Init event is fired when an application initializes the first time.
- **Application_Start:** The Application_Start event is fired the first time when an application starts.
- **Session_Start:** The Session_Start event is fired the first time when a user's session is started. This typically contains for session initialization logic code.
- **Application_BeginRequest:** The Application_BeginRequest event is fired each time a new request comes in.
- **Application_EndRequest:** The Application_EndRequest event is fired when the application terminates.
- **Application_AuthenticateRequest:** The Application_AuthenticateRequest event indicates that a request is ready to be authenticated. If you are using Forms Authentication, this event can be used to check for the user's roles and rights.
- **Application_Error:** The Application_Error event is fired when an unhandled error occurs within the application.
- **Session_End:** The Session_End Event is fired whenever a single user Session ends or times out.
- **Application_End:** The Application_End event is last event of its kind that is fired when the application ends or times out. It typically contains application cleanup logic.

**WEB.CONFIG FILE****Q.34.**

- **What is web.config file? Explain its Structure.** [CBSSGS: Apr – 2016 | Nov – 2016]
- **Explain the use of web.config files in ASP.NET Application.** [IDOL: Dec – 2017 | Oct – 2012]
- **Write a short note on Web.Cong File.** [IDOL: May – 2016]
- **What is the importance of web.config File in any Web Application?** [IDOL: Apr – 2015]

ASKED YEAR ⇒

IDOL: Dec – 2017 | May – 2016 | Apr – 2015 | Oct – 2012

CBSSGS: Apr – 2016 | Nov – 2016

SOLUTIONweb.config File:

- ⇒ The web.config file is an XML based file configuration that contains application wide settings.
- ⇒ It is present in applications root directory we can have more than web.config file for application but we can only machine.config file.
- ⇒ web.config is the main settings and configuration file for web application.
- ⇒ It is an XML document that resides in the root directory of the site or application and contains data about how the web application will act.
- ⇒ This information controls module loading, security configuration, session state configuration, and application language and compilation settings.
- ⇒ web.config files can also contain application specific items such as database connection strings.

Importance / Uses:

- ⇒ The changes in web.config don't require the reboot of the web server.
- ⇒ Store Database Connections, Session States, Error Handling, Security configurations in web.config file.
- ⇒ The changes in web.config don't require the reboot of the web server.
- ⇒ Store Database Connections, Session States, Error Handling, Security configurations in The Official Microsoft ASP.NET Site web.config file.

Structure / General Form:

```
<configuration>
<system.web>
<! Specify Compilation, Custom Error Authentication, Authorization/>
</System.web>
<appsettings>
<! specify file path, connection string, servername, custom setting/>
</appsettings>
</configuration>
```

We can use web.configuration for following:

- (i) Uses <pages> section to specify whether session or view state is enabled or disabled.

Example:

```
<configuration>
<system.web>
<pages enable sessionstate="true"/>
</system.web>
</configuration>
```

- (ii) Set Authentication Mode

Example:

```
<authentication mode="windows"/>
<authorization>
<allow roles="users"/>
<deny users="*/>
</authorization>
```



(iii) We can use <custom Errors> to specify Application Wide Error Information.

Example:

```
<custom Error mode="on">  
<error states code="You" redirect="FileUnavailable.aspx"/>  
</custom errors>
```

(iv) Can specify Key-Value-Pair

Example:

```
<appsettings>  
<add key="key" value="value"/>  
</appsettings>
```

(v) To specify Session Storage Mode

Example:

```
<sessionstate mode="InProc"/>
```



| UNIT |

IV

Programming ASP.NET Web Pages

Navigation And User Controls

Topic

PROGRAMMING ASP.NET WEB PAGES

- ⇒ EVENTS
- ⇒ EVENT HANDLERS
- ⇒ POSTBACK EVENT
- ⇒ ISPOSTBACK
- ⇒ AUTOPOSTBACK
- ⇒ AUTOEVENTWIREDUP
- ⇒ USES OF AUTOPOSTBACK AND RUNAT PROPERTIES
- ⇒ ORGANIZING CODE
 - > CODE BEHIND
 - > INLINE CODE
 - > INLINE CODE Vs. CODE BEHIND
 - > CODE BEHIND Vs. SINGLE FILE
 - > .ASPX Vs .CS
- ⇒ MASTER PAGES AND CONTENT PAGES
 - > MASTER PAGES
 - > CONTENT PAGES
 - > SIGNIFICANCE OF MASTER PAGE IN ASP.NET
 - > ADVANTAGES OF MASTER PAGE
 - > CONTENTPLACEHOLDER'S IN MASTER PAGE
 - > STEPS FOR CREATING THE MASTER PAGE
- ⇒ CACHING
 - > OUTPUT CACHING
 - > DATA CACHING
 - > OBJECT CACHING
 - > ADVANTAGE OF CACHING
 - > DISADVANTAGES OF CACHING

NAVIGATION CONTROLS

- ⇒ MENU CONTROL
- ⇒ TREEVIEW CONTROL
 - > USE OF TREEVIEW CONTROL IN ASP.NET
- ⇒ WEBSITE NAVIGATION
 - > SITEMAP FILE
 - > STRUCTURE OF WEB SITEMAP FILE
 - > STEPS FOR CREATING SITEMAP FILE
 - > SITEMAPPATH CONTROL
 - > SITEMAPDATASOURCE CONTROL
- ⇒ URLENCODE() AND URLDECODE() METHODS
 - > URLENCODE ()
 - > URLDECODE ()
 - > SERVER.REDIRECT Vs. SERVER.TRANSFER
- ⇒ USER CONTROLS
- ⇒ STEPS IN CREATING USER CONTROL
- ⇒ VALIDATION CONTROLS
 - > TYPES OF VALIDATION CONTROLS
 - *CompareValidator*
 - *CustomValidator*
 - *RangeValidator*
 - *RegularExpressionValidator*
 - *RequiredFieldValidator*
- ⇒ USE OF ASP.NET VALIDATION CONTROL
- ⇒ COMMON PROPERTIES FOR ALL VALIDATION CONTROLS
- ⇒ STEPS TO SET UP VALIDATION CONTROL
- ⇒ CAUSEVALIDATION



EVENTS AND EVENT HANDLERS

Q.1. What is an event? What is an Event Handler? How is it designed?

ASKED YEAR →

IDOL: Apr – 2015

SOLUTION

Event:

- ⇒ An event is an action or occurrence such as a mouse click, a key press, mouse movements, or any system-generated notification. A process communicates through events. For example, interrupts are system-generated events. When events occur, the application should be able to respond to it and manage it.
- ⇒ Events in ASP.NET raised at the client machine, and handled at the server machine. For example, a user clicks a button displayed in the browser. A Click event is raised. The browser handles this client-side event by posting it to the server.
- ⇒ The server has a subroutine describing what to do when the event is raised; it is called the event-handler. Therefore, when the event message is transmitted to the server, it checks whether the Click event has an associated event handler.
- ⇒ If it has, the event handler is executed.

Event Handlers:

- ⇒ ASP.NET event handlers generally take two parameters and return void.
- ⇒ The first parameter represents the object raising the event and the second parameter is event argument.
- ⇒ The general syntax of an event is:

```
private void EventName (object sender, EventArgs e);
```

Event Handling Using Controls:

- ⇒ All ASP.NET controls are implemented as classes, and they have events which are fired when a user performs a certain action on them.
- ⇒ For example, when a user clicks a button the 'Click' event is generated. For handling events, there are in-built attributes and event handlers.
- ⇒ Event handler is coded to respond to an event, and take appropriate action on it.
- ⇒ By default, Visual Studio creates an event handler by including a Handles clause on the Sub procedure. This clause names the control and event that the procedure handles.

The ASP tag for a button control:

```
<asp:Button ID="btnCancel" runat="server" Text="Cancel" />
```

The event handler for the Click event:

```
Protected Sub btnCancel_Click(ByVal sender As Object, ByVal e As System.EventArgs)  
Handles btnCancel.Click  
End Sub
```

- ⇒ An event can also be coded without Handles clause.
- ⇒ Then, the handler must be named according to the appropriate event attribute of the control.

The ASP tag for a button control:

```
<asp:Button ID="btnCancel" runat="server" Text="Cancel" Onclick="btnCancel_Click" />
```

The event handler for the Click event:

```
Protected Sub btnCancel_Click(ByVal sender As Object, ByVal e As System.EventArgs)  
End Sub
```

- ⇒ Some events cause the form to be posted back to the server immediately, these are called the postback events. For example, the click event such as, `Button.Click`.
- ⇒ Some events are not posted back to the server immediately, these are called non-postback events.



- ⇒ For example, the change events or selection events such as `TextBox.TextChanged` or `CheckBox.CheckedChanged`. The `nonpostback` events could be made to post back immediately by setting their `AutoPostBack` property to `true`.

POSTBACK EVENT

Q.2.

- What is PostBack Event? Explain IsPostBack with suitable example. [CBSGS: Apr – 2015]
 ➤ What is PostBack Event? Explain with suitable example. [IDOL: May – 2018]

ASKED YEAR ⇒

IDOL: May – 2018

CBSGS: Apr – 2015

SOLUTION

PostBack:

- ⇒ PostBack is an event that is triggered when an action is performed by an ASP.Net Control.
 ⇒ For example when we click on an asp button, the data on the page is posted back to the server for processing.
 ⇒ PostBack is the name given to the process of submitting an ASP.NET page to the server for processing.

IsPostBack:

- ⇒ "IsPostBack" is normally used on page_load event to detect whether page is going to reload (i.e. postback or refresh) due to any asp control of page.
 ⇒ IsPostBack having a Boolean value; thus, the first time that the page loads the IsPostBack flag is false. Each time a PostBack occurs, the entire page including the Page_Load is "posted back" and executed.
 ⇒ IsPostBack is used to check that a page is submitted to server first time or second time rendering, if code are written in "IsPostBack" block,
 ⇒ If (isPostBack())

```
{
//----- your code here-----//
}
```

- ⇒ This code will execute only once when this page runs first time. On any other submit event this page will be submitted to server but this block will not execute.

Example:

```
<html>
<head runat="server">
<script runat="server">
protected void Page_Load(object sender, EventArgs e)
{
if(!IsPostBack)
{
Label1.Text="Page is loaded first time"
}
else
{
Label1.Text="Page is not loaded first time"
}
}
</script>
</head>
<body>
<form id="form1" runat="server">
<asp:Label id="Label1" runat="server" Text="" />
<asp:Button id="Button1" runat="server" Text="Submit" />
</form>
</body>
</html>
```



AUTOPOSTBACK

Q.3. Explain AutoPostBack.

ASKED YEAR →

CBSGS: Apr/Nov – 2016

SOLUTION

AutoPostBack:

- ⇒ AutoPostBack means that if any change happens on a control by the client, it should postback to the server to handle that change at server side.
- ⇒ If this property is set to TRUE the automatic post back is enabled, otherwise FALSE. Default is FALSE.

Example:

```
<asp:TextBox id= "t1" AutoPostBack="true" runat="server" />
```

- ⇒ Web server controls are created on the server and they require a `runat="server"` attribute to work.
- ⇒ This attribute indicates that the control should be treated as a server control.

AutoPostBack Property:

- ⇒ Set this property to true if the server needs to capture the selection as soon as it is made.
- ⇒ For Example, other controls on the web page can be automatically filled depending on the user selection from a list control.
- ⇒ This property can be used to allow automatic population of other controls on the web page based on a user's selection from a list. The value of this property is stored in view state.

Q.4. What are the uses of AutoPostBack and runat properties?

ASKED YEAR →

CBSGS: Apr – 2014

SOLUTION

Uses of AutoPostBack and Runat properties:

- ⇒ AutoPostBack or Postback is nothing but submitting page to server. AutoPostBack is webpage going to server, Server processes the values and sends back to same page or redirects to different page.
- ⇒ HTML elements in ASP.NET files are, by default, treated as text. To make these elements programmable, add a `runat="server"` attribute to the HTML element. This attribute indicates that the element should be treated as a server control.
- ⇒ The `runat="server"` attribute indicates that the form should be processed on the server. It also indicates that the enclosed controls can be accessed by server scripts.
- ⇒ AutoPostBack means that if any change happens on a control by the client, it should postback to the server to handle that change at server side. If this property is set to TRUE the automatic post back is enabled, otherwise FALSE. Default is FALSE.

Example:

```
<asp:TextBox id= " t1" AutoPostBack="true" runat="server" />
```

Web Server Controls are created on the server and they require a `runat="server"` attribute to work. This attribute indicates that the control should be treated as a Server Control.

Example:

```
<asp:Button id="button1" Text="Click me!" runat="server" />
```

AUTOEVENTWIREDUP**Q.5.** Explain AutoEventWiredUp.

ASKED YEAR →

CBSGS: Nov – 2016

SOLUTIONAutoEventWiredUp:

- ⇒ AutoEventWireup is an attribute in Page directive.
- ⇒ It is a Boolean attribute that indicates whether the ASP.NET pages events are auto-wired.
- ⇒ By default it is true.

Example:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" %>
```

CODE-BEHIND MODEL**Q.6.** What is Code-Behind Model in ASP.NET? How is it different from Single File Model?

ASKED YEAR →

CBSGS: Nov – 2015

SOLUTIONCode-Behind Model in ASP.NET:

- ⇒ Code behind refers to the code for your ASP.NET page that is contained within a separate class file.
- ⇒ This allows a clean separation of HTML from the presentation logic.
- ⇒ The file that contains the programming logic (.aspx, .cs file) is a code behind file where only server controls can interact with it.

Code Behind Vs. Single File:

<u>Code Behind</u>	<u>Single File</u>
HTML and controls are in ".aspx" file and code is in separate "aspx.cs".	Code is in <script> block in same ".aspx" file that contains HTML and controls.
Code for the page is compiled into a separate class file from which ".aspx" file are derived.	".aspx" File Derives Page File.
Program logic is separated from user interface design code so it is easy to understand.	Both program logic and user interface design code are placed in same ".aspx" file.

**INLINE CODE VS. CODE BEHIND****Q.7.**

➤ **What is the difference between Inline Code and Code Behind?** [IDOL: May – 2016 | Apr – 2013] | [CBSGS: Apr – 2017 | Nov – 2016 | Apr – 2014]

➤ **What is the Code Behind and Inline Code?** [CBSGS: Nov – 2017]

ASKED YEAR ⇒

IDOL: May – 2016 | Apr – 2013

CBSGS: Nov/Apr – 2017 | Nov – 2016 | Apr – 2014

SOLUTION**Inline Code Vs. Code Behind:**

<u>Inline Code</u>	<u>Code Behind</u>
It is a page model where server side code is stored in the same file as markup.	A page model where server side code is stored in a separate code file.
In inline code model, the code is in <script> blocks in the same .aspx file that contains the HTML and controls.	In Code Behind, the HTML and Controls are in the .aspx file, and the code is in a separate .aspx.vb or .aspx.cs file.
In this, the .aspx file derives from the Page class.	In Code Behind, the code for the page is compiled into a separate class from which the .aspx file derives.
In this, when the page is deployed, the source code is deployed along with the Web Forms page, because it is physically in the .aspx file. However, one does not see the code, only the results are rendered when the page runs.	In Code Behind, all project class files (without the .aspx file itself) are compiled into a .dll file, which is deployed to the server without any source code. When a request for the page is received, then an instance of the project .dll file is created and executed.
Example: <pre><html xmlns="http://www.w3.org/1999/xhtml"> <head runat="server"> <title></title> <body> <form runat="server"> <div> <asp:Label runat="server" Text="Label"> </div> </form> </body> </html></pre>	Example: <pre>public partial class Default: System.Web.UI.Page { protected void Page_Load(object sender, EventArgs e) { Label1.Text="Hello....."; } }</pre>

ASPX VS .CS**Q.8.**

What is the difference between .aspx file and .cs file? Explain with an example for each.

ASKED YEAR ⇒

CBSGS: Oct – 2013

SOLUTION**.aspx Vs .cs:**

<u>.aspx</u>	<u>.CS</u>
It is a page model where server side code is stored in the same file as markup.	A page model where server side code is stored in a separate code file.
ASPX files usually will have the User Interface and which has usually HTML tags, ASP.NET server control embed code (which ultimately produce some HTML markups).	ASPX.CS file (usually called the code behind) will have server side coding in C#.
In this, the .aspx file derives from the Page class.	In Code Behind, the code for the page is compiled into a separate class from which the .aspx file derives.
In this, when the page is deployed, the source code is deployed along with the Web Forms page, because it is physically in the	In Code Behind, all project class files (without the .aspx file itself) are compiled into a dll file which is deployed to the



".aspx" file. However, one does not see the code, only the results are rendered when the page runs.

server without any source code. When a request for the page is received, then an instance of the project .dll file is created and executed.

Example:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
</head>
<body>
<form runat="server">
<div>
<asp:Label runat="server" Text="Label">
</div>
</form>
</body>
</html>
```

Example:

```
public partial class_Default:
System.Web.UI.Page
{
protected void Page_Load(object sender,
EventArgs e)
{
Label1.Text="Hello.....";
}
}
```

MASTER PAGE AND CONTENT PAGE**Q.9.**

- **Explain relationship between Content Page and Master Page. [IDOL: May – 2017 | May – 2016 | Oct – 2012] | [CBSGS: Apr – 2017 | Nov/Apr – 2016 | Apr – 2015 | Apr – 2014]**
- **What is the use of MasterPages in ASP.NET? How a Content Page can be added to a Master Page. [CBSGS: Nov – 2014]**
- **What are Content Pages? [IDOL: Apr – 2013]**
- **What is the significance of Master pages in ASP.NET? What is the name given to pages other than Master Pages? Explain in detail. [CBSGS: Oct – 2013]**
- **What are the advantages of a Master Pages? Explain about ContentPlaceHolder's in Master Pages. [CBSGS: Nov – 2015]**
- **Explain the role of Master Pages in ASP.NET. [IDOL: Dec – 2017 | Apr – 2014]**

ASKED YEAR →

IDOL: Dec/MAY – 2017 | MAY – 2016 | Apr – 2014 | OCT – 2012

CBSGS: APR – 2017 | NOV/APR – 2016 | APR – 2015 | NOV/APR – 2014 | Oct – 2013

SOLUTION**Master Pages:**

- ⇒ 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.

Example:

```
<%@ Master %>
<html>
<body>
<h1>Standard Header From Masterpage</h1>
<asp:ContentPlaceHolder id="CPH1" runat="server">
</asp:ContentPlaceHolder>
</body>
</html>
```

- ⇒ The master page above is a normal HTML page designed as a template for other pages.
- ⇒ The @ **Master** directive defines it as a master page.



- ⇒ The master page contains a placeholder tag `<asp:ContentPlaceholder>` for individual content.
- ⇒ The `id="CPH1"` attribute identifies the placeholder, allowing many placeholders in the same master page.
- ⇒ This master page was saved with the name "`master1.master`".

Significance of Master Page in ASP.NET:

- ⇒ The Master Page serves as a template for the other content pages on the site.
- ⇒ The Master Page has some code-behind methods, and it could auto-generate the page titles for the pages and the other tags which are mentioned in it for each content page.
- ⇒ Master Page stores global page elements that occur on every content page.
 - Extension: ".Master"
- ⇒ Content Page stores page-specific elements that are put into the master.
 - Extension: ".aspx"
- ⇒ Master Page code behind can change master page after it acquires content.
 - Extension: ".aspx.cs"

Advantages of a Master Page:

- **Enhanced User Experience:** As all content pages are inserted within the same master page web interface will provide more consistent look.
- **Common Controls:** Keeping the common controls inside the master page the control updating will be at one location.
- **Less Code:** A simple web site may only require code in its master page, so the content page don't have any code and easier to work with.
- **Less Storage Requirements:** As most of the web site interface code can be placed in the master page the content pages will be much small.

ContentPlaceholder's in Master Page:

- ⇒ Every master page should include at least one ContentPlaceholder control, this is a placeholder for the content from the content page.

Example:

```
<asp:ContentPlaceholder id="ContentPlaceholder" runat="server">
</asp:ContentPlaceholder>
```

- ⇒ Inside the content pages by default content server control is get added. While designing the content page we have to add controls to the content server control.

Example:

```
<asp:content id="content" ContentPlaceHolderId="ContentPlaceholder1" runat="server">
</asp:content>
```

Content Page:

- ⇒ Once a master page is created, the content pages can be developed.
- ⇒ Choose the website a add New item a select the web form and enter the name of the form, check the select Master Page check box and click add. When the select a Master Page dialog box appears, select the master page.
- ⇒ The Page Directive in the aspx code for a content page includes a MasterPagefile attributes that specifies the name of the master page.

Example:

```
<%@ Page MasterPageFile="master1.master" %>
<asp:Content ContentPlaceHolderId="CPH1" runat="server">
<h2>Individual Content</h2>
<p>Paragraph 1</p>
<p>Paragraph 2</p>
</asp:Content>
```

- ⇒ The content page above is one of the individual content pages of the web.
- ⇒ The `@ Page` directive defines it as a standard content page.



- ⇒ The content page contains a content tag `<asp:Content>` with a reference to the master page (`ContentPlaceHolderId="CPH1"`).
- ⇒ This content page was saved with the name "mypage1.aspx".
- ⇒ When the user requests this page, ASP.NET merges the content page with the master page.

Q.10. Explain the steps of creating Master Page in Web Development.ASKED YEAR ⇒ IDOL: Oct - 2016**SOLUTION****Steps for Creating the Master Page in Web Development:****STEP 1: Open New Project in Visual Studio**

- ⇒ New project → Installed → Web → ASP.NET Web Application.
- ⇒ After clicking OK button in the Window → Select Empty
- ⇒ After clicking OK button, project "masterpage" opens but no file is there.

STEP 2: Add New File in to our Project

- ⇒ Add the Master Page into our project.
- ⇒ Right Click Project → Add → New Item
- ⇒ After clicking on New Item, Window will open, select Web Form → Web Forms Master Page
- ⇒ After clicking the Add Button, Master Page "site1.master" adds to our project.
- ⇒ Click on "site1.master" into Solution Explorer.

STEP 3: Design the Master Page, using HTML**HTML code of my master page is as follows:**

```
<%@ Master Language="C#" AutoEventWireup="true" CodeBehind="Site1.master.cs"
Inherits="masterpage.Site1" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title>c# corner</title>
<link href="css/my.css" rel="stylesheet" />
<asp:ContentPlaceHolder ID="head" runat="server">
</asp:ContentPlaceHolder>
</head>
<body>
<!DOCTYPE html>
<html>
<head>
<title>my layout</title>
<link rel="stylesheet" type="text/css" href="my.css">
</head>
<body>
<header id="header">
<h1>c# corner</h1>
</header>
<nav id="nav">
<ul>
<li><a href="home.aspx">Home</a></li>
<li><a href="#">About</a></li>
<li><a href="#">Article</a></li>
<li><a href="#">Contact</a></li>
</ul>
</nav>
<aside id="side">
<h1>news</h1>
<a href="#"><p>creating html website</p></a>
<a href="#"><p>learn css</p></a>
```



```
<a href="#">learn c#</a>
</aside>
<div id="con">
<asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
</asp:ContentPlaceHolder>
</div>
<footer id="footer">
copyright @c# corner
</footer>
</body>
</html>
<form id="form1" runat="server">
</form>
</body>
</html>
```

CSS Code:

```
#header{
color: #247BA0;
text-align: center;
font-size: 20px;
}
#nav{
background-color:#FF1654;
padding: 5px;
}
ul{
list-style-type: none;
}
li a {
color: #F1FAEE;
font-size: 30px;
column-width: 5%;
}
li
{
display: inline;
padding-left: 2px;
column-width: 20px;
}
a{
text-decoration: none;
margin-left:20px
}
li a:hover{
background-color: #F3FFBD;
color: #FF1654;
padding:1%;
}
#side{
text-align: center;
float: right;
width: 15%;
padding-bottom: 79%;
background-color: #F1FAEE;
}
#article{
background-color: #EEF5DB;
padding: 10px;
padding-bottom: 75%;
}
```



```
#footer{
background-color: #C7EFCF;
text-align:center;
padding-bottom: 5%;
font-size: 20px;
}
#con{
border:double;
border-color:burllywood;
}
```

Our master page is designed. Move to the next step.

STEP 4: Add Web Form in to our Project.

- ⇒ Right click on the project → Add → New Item
- ⇒ Select Web Form with the Master Page.
- ⇒ After clicking on that, add the Button Window.
- ⇒ Open the selected Masterpage → "Site1.Master" and click OK.
- ⇒ Now, design our Homepage.
- ⇒ Here, we write home page only.

Home.aspx

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site1.Master" AutoEventWireup="true"
CodeBehind="home.aspx.cs" Inherits="masterpage.home" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
<h1>Home page</h1>
</asp:Content>
```

Finally, our Master Page is created; build and run the project.

The Master Page looks as shown in the below picture:





CACHING

Q.11.

- **What are the three different ways to store data in the Cache Object?** [IDOL: Dec – 2017]
- **What is a Caching? Explain its types.** [IDOL: Apr – 2015]
- **Define Caching in ASP.NET. List the advantages and disadvantages.** [IDOL: May – 2016 | Apr – 2013]

ASKED YEAR ⇒

IDOL: Dec – 2017 | May – 2016 | Apr – 2015 | Apr – 2013

SOLUTION

Caching:

- ⇒ 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.
- ⇒ Caching places frequently used data in quickly accessed media such as the random access memory of the computer. The ASP.NET runtime includes a key-value map of CLR objects called cache. This resides with the application and is available via the HttpContext and System.Web.UI.Page.
- ⇒ In some respect, caching is similar to storing the state objects. However, the storing information in state objects is deterministic, i.e., you can count on the data being stored there, and caching of data is nondeterministic.
- ⇒ The data will not be available in the following cases:
 - If its lifetime expires,
 - If the application releases its memory,
 - If caching does not take place for some reason.

Different ways to store data in the Cache Object:

Output Caching:

- ⇒ Rendering a page may involve some complex processes such as, database access, rendering complex controls etc. Output caching allows bypassing the round trips to server by caching data in memory. Even the whole page could be cached.
- ⇒ The OutputCache directive is responsible of output caching. It enables output caching and provides certain control over its behaviour.

Syntax for OutputCache directive:

```
<%@ OutputCache Duration="15" VaryByParam="None" %>
```

Data Caching:

- ⇒ The main aspect of data caching is caching the data source controls.
- ⇒ 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.

Object Caching:

- ⇒ Object caching is caching the objects on a page, such as data-bound controls.
- ⇒ The cached data is stored in server memory.
- ⇒ Object Caching
- ⇒ Object caching provides more flexibility than other cache techniques.
- ⇒ We can use object caching to place any object in the cache.
- ⇒ The object can be of any type - a data type, a web control, a class, a dataset object, etc.



⇒ The item is added to the cache simply by assigning a new key name, shown as follows Like:

```
Cache["key"] = item;
```

Advantage of Caching:

- ⇒ It reduces the overhead from server resources.
- ⇒ It increases the performance of the application by serving user with cached output.
- ⇒ It decreases server round trips for fetching data from the database by persisting data in the memory.
- ⇒ The page download is faster.
- ⇒ It can make or break the performance of your computer system.
- ⇒ It takes the heavy load from the server for the repeated operations.
- ⇒ It reduces the load on the web services or database.
- ⇒ It increases reliability.

Disadvantages of Caching:

- ⇒ In terms of Caching approach, it will not be coherent with the actual data content in the delayed writes.
- ⇒ Increased Maintenance.
- ⇒ Scalability Issue.
- ⇒ Could run into issues syncing caches.
- ⇒ As information store in cache so it make page heavy.
- ⇒ May be updated information is not show.

NAVIGATION CONTROLS

Q.12.

- **Explain Menu Site Navigation Controls. [IDOL: May – 2017 | Oct – 2016 | Apr – 2014] | [CBSGS: Oct – 2013 | Nov – 2014]**
- **Explain any two Site Navigation Controls in ASP.NET. [IDOL: Apr – 2015]**

ASKED YEAR ⇒

[IDOL: May – 2017 | Oct – 2016 | Apr – 2014]

[CBSGS: Oct – 2013 | Nov – 2014]

SOLUTION

Menu Control:

- ⇒ The Menu Control displays site navigation information in a menu. Submenus automatically appear when the user hovers the mouse over a menu item that has a submenu.
- ⇒ To display an application's navigation structure, the Menu control must be bound to a `SiteMapDataSource` control.
- ⇒ The menu control has many formatting attributes. We can quickly apply a coordinated set of formatting attributes by clicking the Smart Tag icon for the Menu Control and choosing Auto Format from the menu that appears.
- ⇒ Then, we can select one of several predefined schemes for the menu.

Attributes of Menu Control:

- **ID:** The ID of the control.
- **Runat:** Most specify "server".
- **DataSourceID:** The ID of the `SiteMapDataSource` control menu should be bound to.
- **CssClass:** It enables us to set a CSS class attribute that applies to the entire control.
- **StaticEnableDefaultPopOutImage:** A Boolean that determines whether images are used to indicate submenus on the top-level menu items.
- **DisappearAfter:** It determines the time in milliseconds that menu items will remain visible after you move your mouse away from them.
- **DataSourceID:** The ID of a `SiteMapDataSource` control that supplies the data for the menu from the Web sitemap file.
- **IncludeStyleBlock:** If true, the CSS that formats the menu is included in a style element in the rendered HTML. If false, no style element is generated. The default is true.
- **ItemWarp:** If True, words in the menu items will be word-wrapped if necessary. The default is False.
- **MaximumDynamicDisplay:** The number of levels of dynamic submenus to display.



- **Orientation:** It determines whether to use a horizontal menu with dropout submenus, or a vertical menu with fold-out submenus. The default is Vertical.
- **RenderingMode:** It determines whether the control presents itself using tables and inline styles or unordered list and CSS styles.
- **StaticDisplayLevels:** The number of levels that should always be displayed. The default is 1.
- **StaticEnableDefaultPopOutImage:** If true, an arrow graphic is displayed next to any menu item that has a pop-out submenu. If false, the arrow graphic is not displayed. The default is true.

ASPX Code for Menu Control:

```
<asp:Menu id="menu1" Orientation="Horizontal" runat="server" DataSourceID="SiteMapDataSource1">
</asp:Menu>
```

TREE VIEW CONTROL**Q.13.**

- **Explain the TreeView Control.** [IDOL: Apr – 2013] | [CBSGS: Nov – 2017 | Nov – 2016]
- **When do we have to use TreeView Control in ASP.NET?** [IDOL: May – 2018 | Apr – 2014]
- **Explain TreeView Site Navigation Controls.** [IDOL: May – 2017 | Apr – 2013] | [CBSGS: Nov – 2014 | Oct – 2013]
- **Explain any two Site Navigation Controls in ASP.NET.** [IDOL: Apr – 2015]

ASKED YEAR ⇒ [IDOL: May – 2018 | May – 2017 | Apr – 2015 | Apr – 2014 | April – 2013]

[CBSGS: Nov – 2017 | Nov – 2016]

SOLUTIONTreeView Control:

- ⇒ A TreeView is capable of displaying a hierarchical list of items, similar to how the tree in Windows Explorer looks.
- ⇒ Items can be expanded and collapsed with the small plus and minus icons in front of items that contain child elements.
- ⇒ The data used by the TreeView Control is not limited to the Web.sitemap file, however. We can also bind it to regular XML files and even create a TreeView or its items (called nodes) programmatically.

Attributes of TreeView Control:

- **ID:** The ID of the control.
- **Runat:** Must specify "server".
- **DataSourceID:** The ID of the SiteMapDataSource control the tree should be bound to.
- **ExpandDepth:** The number of levels to be automatically expanded when the tree is initially displayed. The default is FullyExpand.
- **MaxDepthDataBind:** Limits the maximum depth of the tree. The default is -1, which places no limit.
- **NodeIndent:** The number of pixels to indent each level. The default is 20.
- **NodeWrap:** Set to True to word-wrap the text of each node. The default is False.
- **ShowExpandCollapse:** Set to False if we want to hide the Expand/Collapse buttons. The default is True.
- **ShowLines:** Set to True to include lines that show the hierarchical structure. The default is False.

Common properties of TreeView:

- **CssClass:** It enables to set a CSS class attribute that applies to the entire control.
- **CollapseImageUrl:** The image that collapses a part of the tree when clicked. The default is an icon with a minus symbol on it.
- **ExpandImageUrl:** The image that expands a part of the tree when clicked. The default is an icon with a plus symbol on it.
- **CollapseImageToolTip:** The tooltip that is shown when a hovers over a collapsible menu item.
- **ExpandImageToolTip:** The tooltip that is shown when a user hovers over an expandable menu item.

To create a TreeView control follows the following steps:

- ⇒ Open the web/master page in design View and add the TreeView control from the Navigation Toolbox to the page.

**Syntax:**

```
<asp:TreeView ID="TreeView1" runat="server"></asp:TreeView>
```

Example:

```
<asp:TreeView ExpandDepth="1" runat="server">
<Nodes>
<asp:TreeNode Text="Employees">
<asp:TreeNode Text="Bradley" Value="ID-1234" />
<asp:TreeNode Text="Whitney" Value="ID-5678" />
<asp:TreeNode Text="Barbara" Value="ID-9101" />
</asp:TreeNode>
</Nodes>
</asp:TreeView>
```

Use of TreeView Control in ASP.NET:

- ⇒ ASP.NET TreeView Web control is used to display hierarchical data (such as a table of contents) in a tree structure in a Web page.
- ⇒ The TreeView control is made up of TreeNode objects. The TreeView control can be bound to data.

It supports the following features:

- ⇒ Automatic data binding, which allows the nodes of the control to be bound to hierarchical data, such as an XML document.
- ⇒ Site Navigation support through integration with the SiteMapDataSource control.
- ⇒ Node text that can be displayed as either selectable text or hyperlinks.
- ⇒ Customizable appearance through themes, user-defined images, and styles.
- ⇒ Programmatic access to the TreeView object model, which allows you to dynamically create trees, populate nodes, set properties, and so on.
- ⇒ Node population through client-side callbacks to the server (on supported browsers).
- ⇒ The ability to display a check box next to each node.

```
<asp:TreeView ExpandDepth="1" runat="server">
<Nodes>
<asp:TreeNode Text="Employees">
<asp:TreeNode Text="Bradley" Value="ID-1234" />
<asp:TreeNode Text="Whitney" Value="ID-5678" />
<asp:TreeNode Text="Barbara" Value="ID-9101" />
</asp:TreeNode>
</Nodes>
</asp:TreeView>
```

- ⇒ Each node in the Tree is represented by a name/value pair (not necessarily unique), defined by the Text and Value properties of TreeNode, respectively. The text of a node is rendered, whereas the value of a node is not rendered and is typically used as additional data for handling postback events.
- ⇒ This example also uses the ExpandDepth property of TreeView to automatically expand the tree 1 level deep when it is first rendered.
- ⇒ The TreeView control is made up of one or more nodes. Each entry in the tree is called a node and is represented by a TreeNode object. The following table describes the three different node types.

Type of Node:

- **Root:** A node that has no parent node and one or more child nodes.
- **Parent:** A node that has a parent node and one or more child nodes.
- **Leaf:** A node that has no child nodes.



WEBSITE NAVIGATION

Q.14.

- **What is Website Navigation?** [CBSGS: Apr – 2017]
- **Explain structure of web.sitemap file with suitable example.** [CBSGS: Apr – 2017]
- **Explain a syntax for creating sitemap file with suitable example.** [IDOL: Dec – 2017]
- **List the steps for creating Sitemap File.** [IDOL: Dec – 2017]
- **What is a Sitemap File?** [CBSGS: Apr – 2016]

ASKED YEAR ⇒

IDOL: Dec – 2017

CBSGS: Apr – 2017 | Apr – 2016

SOLUTION

Website Navigation:

ASP.NET Site navigation enables us to store links to all of our pages in a central location, and render those links in lists or navigation menus on each page by including a specific web server control.

SiteMap File:

- ⇒ All the navigation controls are uses an XML-based file which is known as site map.
- ⇒ Site maps are XML files which are used to describe the logical structure of web application.
- ⇒ It is used to define the layout of all pages in web application and how they relate to each other. Site map files are defined with sitemap extension.
- ⇒ The root nodes of sitemap file is <sitemap> element within the element, there is a element. This is generally the start page of the application.
- ⇒ <siteMapNode> element have the following attributes:
 - **Title:** It provides a textual description of the link.
 - **Description:** It is used for Tool/Tip of the link.
 - **URL:** It gives the location of the valid physical file.

Structure of Web SiteMap File:

```
<Sitemap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-file-1.0">
<SiteMapNode url="~//" title="home" description="MUMBAI UNIVERSITY KALINA">
<SiteMapNode url="Default.aspx" title="Home" description="Go to the homepage"/>
<SiteMapNode url="IT.aspx" title="IT Department" description="BSCIT INFORMATION"/>
<SiteMapNode>
<SiteMapNode url="COMMERCE.aspx" title="COMMERCE DEPARTMENT" description="COMMERCE DEPARTMENT
INFORMATION">
<SiteMapNode url="BFM.aspx" tite="BFM" description="BFM INFORMATION"/>
<SiteMapNode url="BBI.aspx" title="BBI" description="BBI INFORMATION"/>
</SiteMapNode>
</SiteMap>
```

Steps for creating Sitemap File:

- STEP 1:** Create a new ASP.NET Empty Web Application
- STEP 2:** Add a new Web Form named Home.aspx,
- STEP 3:** Add new Web Form named ProductGroups.aspx,
- STEP 4:** Add a new Web Form named PlasticItems.aspx,
- STEP 5:** Add a new Web Form named Furniture.aspx
- STEP 6:** Add a new Web Form named AboutUs.aspx



STEP 7: Add a new Web Form named ContactUs.aspx,

Till now, we added the following WebForm files,

1. AboutUs.aspx
2. ContactUs.aspx
3. Furnitures.aspx
4. Home.aspx
5. PlasticItems.aspx
6. ProductGroups.aspx

STEP 8: Add Sitemap file. Right click on Project and select,

STEP 9: Double on Web.sitemap file type the following code:

```
<?xmlversionxmlversion="1.0"encoding="utf-8" ?>
<siteMapxmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">
  <siteMapNodeurl="Home.aspx" title="Home" description="Home Page">
    <siteMapNodeurl="ProductGroups.aspx" title="Product Group" description="Product
    Group Page">
      <siteMapNodeurl="Furnitures.aspx" title="Furnitures" description="Furniture Items
      Page" />
      <siteMapNodeurl="PlasticItems.aspx" title="Plastics" description="Plastic Items
      Page" />
    </siteMapNode>
  <siteMapNodeurl="Aboutus.aspx" title="About Us" description="About Us Page" />
  <siteMapNodeurl="Contactus.aspx" title="Contact Us" description="Contact Us Page"
  />
</siteMapNode>
</siteMap>
```

STEP 10: Explanation of Web.sitemap code

Web.sitemap is XML base file.

In this file there is one main SiteMapNode that is HOME - Home.aspx. Under that we have the following two sections:

1. ProductGroup section -- (ProductGroups.aspx) having following child link:

- a. Furniture.aspx
 - b. PlasticItems.aspx
2. Two Independent Page.
- a. AboutUs.aspx
 - b. ContactUs.aspx

STEP 11: Now, Drag and Drop the SiteMapPath control from ToolBox inside Navigation group,

Inside following files, write some sample text in every page.

1. AboutUs.aspx
2. ContactUs.aspx
3. Furnitures.aspx
4. Home.aspx
5. PlasticItems.aspx
6. ProductGroups.aspx

STEP 12: Now run application. I had typed Furniture.aspx in address bar.

You can see furnitures.aspx page shown. You came here via Home, Product Group, Furnitures.



**SITEMAPPATH CONTROL****Q.15. What is SiteMapPath Navigation Controls?**

ASKED YEAR → IDOL: Oct – 2016 | Apr – 2013

CBSGS: Nov – 2016 | Nov – 2015

SOLUTION**SiteMapPath Control:**

- ⇒ The SiteMapPath Control shows you where you are in the site's structure.
- ⇒ It presents itself as a series of links, often referred to as breadcrumb.
- ⇒ Just like the Menu and TreeView it has a number of style properties you use to change the look of elements like the current node, a normal node, and the path separator.
- ⇒ SiteMapPath will automatically show itself in line with the other links in the page.

Property of SiteMapPath Control:

- **PathDirection:** It supports two values: `RootToCurrent` and `CurrentToRoot`. The first setting shows the root element on the left, intermediate levels in the middle, and the current page at the right of the path.
- **PathSeparator:** It defines the symbol or text to show between the different elements of the path. The default is the greater than symbol (>) but you can change it to something like the pipe character (|).
- **RenderCurrentNodeAsLink:** It determines whether the last element of path (the Current page) is rendered as a text link or as plain text. The default is `False`, which is usually fine, because you are already on the page that element is representing, so there's no real need for a link.
- **ShowToolTips:** It determines whether the control displays tooltips (retrieved from the description attribute of the `siteMapNode` elements in the Web.sitemap file) when the user hovers over the elements in the path. The default is `True`, which means the tooltips are shown by default.

SITEMAPPATH CONTROL**Q.16. What is the role of the SiteMapDataSource Control?**

ASKED YEAR →

CBSGS: Apr – 2016

SOLUTION**SiteMapDataSource**

- ⇒ The `SiteMapDataSource` control is a data source to the site map data that is stored by the site map providers that are configured for your site.
- ⇒ The `SiteMapDataSource` enables Web server controls that are not specifically site navigation controls, such as the `TreeView`, `Menu`, and `DropDownList` controls, to bind to hierarchical site map data.
- ⇒ You can use these Web server controls to display a site map as a table of contents or to actively navigate a site. Alternatively, you can use the `SiteMapPath` control, which is designed specifically as a site navigation control and therefore does not need an instance of the `SiteMapDataSource` control.
- ⇒ Sitemap data is retrieved from an `SiteMapProvider` object, such as `XmlSiteMapProvider`, which is the default site map provider for ASP.NET. You can specify any provider that is configured for your site to provide the site map data to the `SiteMapDataSource` and can obtain the list of available providers by accessing the `SiteMap Providers` collection.



URLENCODE() AND URLDECODE() METHODS

Q.17.

- Why `UrlEncode ()` and `UrlDecode ()` methods are used in ASP.NET? Explain. [CBSGS: Nov – 2016]
- Explain URL Encoding in detail. [CBSGS: Nov – 2015]
- Explain `UrlEncode ()` and `UrlDecode ()` methods in ASP.NET. [CBSGS: Apr – 2015]

ASKED YEAR →

CBSGS: Nov – 2016 | Nov/Apr – 2015

SOLUTION

UrlEncode() and UrlDecode() Methods in ASP.NET:

- ⇒ We cannot send special characters through query string. All special characters should be encoded when you pass them through the query string. The encoded string must be decoded at the receiver.
- ⇒ There are two methods to achieve this – `UrlEncode()` and `UrlDecode()`
- ⇒ The main purpose of these two methods is to encode and decode the URL respectively.
- ⇒ We need to encode the URL because some of the characters are not safe sending those across browser.
- ⇒ Some characters are being misunderstood by the browser and that leads to data mismatch on the receiving end.
- ⇒ Characters such as a question mark (?), ampersand (&), slash mark (/), and spaces might be truncated or corrupted by some browsers.

UrlEncode () :

- ⇒ It is used for encoding data that will be passed through a query string variable.
- ⇒ `UrlEncode` replaces unsafe ASCII characters with a "%" followed by two hexadecimal digits.
- ⇒ A query string variable is a data that follows question mark (?) in the URL field of your browser.
- ⇒ We can create a query string variable at the time of redirect operation or while building a hyperlink to another page.

Syntax:

```
UrlEncode (string str)
```

For Encoding:

- ⇒ `UrlEncode ()` method is used.
- ⇒ It is the process of converting string into valid URL format.

```
string UrlEncode = Server.UrlEncode("Redirect.aspx? Name = bscit&ID = 5");  
Response.Write("Encoding URL:-" + UrlEncode);  
//shows like this Redirect.aspx%3f+Name+%3d+bscit%26ID+%3d+5
```

UrlDecode ()

- ⇒ This method is used to decode the encoded URL string. Decodes any %## encoding in the given string.

Syntax:

```
UrlDecode (string str)
```

For Decoding:

- ⇒ `UrlDecode()` method is used.
- ⇒ **UrlDecode** – URL decodes a string and returns the decoded string.

```
string UrlDecode = Server.UrlDecode(UrlEncode);  
Response.Write("Decoding URL:-" + UrlDecode);  
//shows like this Redirect.aspx? Name = bscit&ID = 5
```

SERVER.REDIRECT VS. SERVER.TRANSFER**Q.18.** What is difference between `Server.Redirect` and `Server.Transfer`? Explain with suitable example.

ASKED YEAR →

IDOL: May – 2016

SOLUTIONServer.Redirect Vs. Server.Transfer:

Server.Redirect	Server.Transfer
<code>Response.Redirect()</code> will send you to a new page, update the address bar and add it to the Browser History. On your browser you can click back.	<code>Server.Transfer()</code> does not change the address bar, we cannot hit back. One should use <code>Server.Transfer()</code> when he/she doesn't want the user to see where he is going. Sometime on a "loading" type page.
It redirects the request to some plain HTML pages on our server or to some other web server.	It transfers current page request to another .aspx page on the same server.
It causes additional roundtrips to the server on each request.	It preserves server resources and avoids the unnecessary roundtrips to the server.
It doesn't preserve Query String and Form Variables from the original request.	It preserves Query String and Form Variables (optionally).
It enables to see the new redirected URL where it is redirected in the browser (and be able to bookmark it if it's necessary).	It doesn't show the real URL where it redirects the request in the users Web Browser.
<code>Response.Redirect</code> simply sends a message down to the (HTTP 302) browser.	<code>Server.Transfer</code> happens without the browser knowing anything, the browser request a page, but the server returns the content of another.

Using the Code:

When you use `Server.Transfer`, then the previous page also exists in server memory while in the `Response.Redirect` method, the previous page is removed from server memory and loads a new page in memory. Let's see an example.

Example:

We add some items in the context of the first page "`UserRegister.aspx`" and thereafter the user transfers to another page "`UserDetail.aspx`" using `Server.Transfer` on a button click. The user will then be able to request data that was in the context because the previous page also exists in memory. Let's see the code.

Code of the UserRegister.aspx.cs page:

```
using System;
public partial class UserRegister : System.Web.UI.Page
{
    protected void btn_Detail_Click(object sender, EventArgs e)
    {
        Context.Items.Add("Name", "Sandeep Singh Shekhawat");
        Context.Items.Add("Email", "sandeep.shekhawat88@gmail.com");
        Server.Transfer("UserDetail.aspx");
    }
}
```

Code of the UserDetail.aspx.cs page:

```
using System;
public partial class UserDetail : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        try
```



```
{
Response.Write(string.Format("My name is {0} and email address is {1}",
Context.Items["Name"].ToString(),
Context.Items["Email"].ToString()));
}
catch (NullReferenceException ex)
{
Response.Write(ex.Message);
}
}
}
```

USER CONTROLS

Q.19. How do we create a User Control in ASP.NET?

ASKED YEAR → IDOL: May – 2018 | May – 2016 | Apr – 2014

SOLUTION

Creating a User Control in ASP.NET:

- ⇒ User control declarative syntax is very similar to syntax used to create an ASP.NET Web page.
- ⇒ The primary differences are that the user controls use an @ Control Directive in place of an @ Page directive, and that the user controls do not include the html, body, and form elements around the content.

Steps in Creating User Control:

STEP 1: Create a new file and give it a name with the extension ".ascx".

For example, name your user control "DisplayName.ascx".

STEP 2: Create an @ Control directive at the top of the page and specify the programming language you want to use for the control (if any).

STEP 3: Add controls that you want the user control to display.

STEP 4: Add code for the tasks that the user control will perform, such as handling control events or reading data from a data source.

STEP 5: Create properties in the control if you want to be able to share information between the user control and the hosting page. You create properties as you would for any class, either as public members or with get and set accessors.

Example:

The following example shows an ASP.NET Web page that contains a user control. The user control is in the file "Spinner.ascx" in the Controls folder. In the page, the control is registered to use the prefix uc and the tag name Spinner. The user control properties MinValue and MaxValue are set declaratively.

```
<%@ Page Language="C#" %>
<%@ Register TagPrefix="uc" TagName="Spinner" Src="~/Controls/Spinner.ascx" %>
<html>
<body>
<form runat="server">
<uc:Spinner id="Spinner1" runat="server" MinValue="1" MaxValue="10" />
</form>
</body>
```


**VALIDATION CONTROLS****Q.20.**

- Explain Validation Controls with example. [IDOL: May – 2018 | May – 2017 | Oct/May – 2016 | April – 2015 | Oct – 2012] | [CBSGS: Nov/Apr – 2016 | Apr – 2014]
- Explain RegularExpressionValidator with the help of an example. [IDOL: April – 2013]
- What is RangeValidator? Describe any four properties of it. [IDOL: April – 2013] | [CBSGS: Nov – 2015]
- Explain CustomValidator control with suitable example. [CBSGS: Apr – 2015]
- What is the use of Compare Validators? Explain it along with its properties? [CBSGS: Nov – 2014]

ASKED YEAR →

IDOL: May – 2018 | May – 2017 | Oct/ May – 2016 | April – 2015 | April – 2013 CBSGS: Nov/Apr – 2016 | Apr/Nov – 2015 | Nov/Apr – 2014
| Oct – 2012

SOLUTION**Validation Controls:**

- ⇒ An important aspect of creating ASP.NET Web pages for user input is to be able to check that the information users enter is valid.
- ⇒ ASP.NET provides a set of validation controls that provide an easy-to-use but powerful way to check for errors and, if necessary, display messages to the user.
- ⇒ ASP.NET validation controls validate the user input data to ensure that useless, unauthenticated, or contradictory data don't get stored.

Types of Validation Controls:

ASP.NET provides the following Validation Controls:

- 1) RequiredFieldValidator
- 2) RangeValidator
- 3) CompareValidator
- 4) RegularExpressionValidator
- 5) CustomValidator
- 6) ValidationSummary

RequiredFieldValidator:**Syntax:** <asp:RequiredFieldValidator>

- ⇒ It checks that the validated control contains a value.
- ⇒ It cannot be empty. It can be used in conjunction with other validators on a control to trap empty values.
- ⇒ Simply, we can use it to check if the input control has any value.
- ⇒ The most important property in the RequiredFieldValidator is `InitialValue`.

Example:

```
<asp:RequiredFieldValidator runat="server" controlToValidate="txtName" errorMessage="Name must be entered" display="static">
</asp:RequiredFieldValidator>
```

Additional Property:

- **InitialValue** – The initial value of the control that is validated.

RangeValidator:**Syntax:** <asp:RangeValidator>

- ⇒ It checks if the input control's value is within a specified range.
- ⇒ In other words, it checks that the value in the validated control is within the specified text or numeric range.
- ⇒ If the validated control is empty, no validation takes place.
- ⇒ The most important properties in the Range Validator are `MaximumValue`, `MinimumValue`, and `type`.

**Example:**

```
<asp:RangeValidator runat="server" display="static" controlToValidate="txtDependents"
errorMessage="Must be from 0 to 10" type="Integer" minimumValue="0" maximumValue="10">
</asp:RangeValidator>
```

Properties of RangeValidator Control:

- **ControlToValidate:** The id of the control to validate.
- **ErrorMessage:** The message to display in the control when validation fails.
- **Maximum:** It specifies the Maximum values of the input control.
- **Minimum:** It specifies the Minimum values of the input control.
- **Type:** It specifies the data type of the value to check. The valid types are Date, double, integer string, currency.

CompareValidator:**Syntax:** <asp:CompareValidator>

- ⇒ It checks if the value is acceptable compared to a given value or compared to the content of another control.
- ⇒ In other words, it checks that the value in the validated control matches the value in another control or a specific value. The data type and comparison operation can be specified.
- ⇒ If the validated control is empty, no validation takes place.
- ⇒ The most important properties is ValueToCompare, ControlToCompare, Operator and type.

Properties of Compare Validator:

- **ValueToCompare:** The value that the control specified in the ControlToValidate property should be compared to.
- **Operator:** The type of comparison to perform (Equal, NotEqual, GreaterThan, GreaterThanEqual, LessThan, LessThanEqual, or DataTypeCheck).
- **Type:** The data type to use for the comparison (String, Integer, Double, Date or Currency).
- **ControlToCompare:** The ID of the control that the value of the control specified in the ControlToValidate property should be compared to.

A compare Validator that checks for a value greater than zero:

```
<asp:TextBox id="txtQuantity" runat="server">
</asp:TextBox>&nbsp;
<asp:CompareValidator ID="CompareValidator1" runat="server" ControlToValidate="txtQuantity"
Type="Integer" Operator="GreaterThan" ValueToCompare=""0 ErrorMessage="Quantity must be greater
than zero.">
</asp:CompareValidator>
```

A compare Validator that checks for an integer value:

```
<asp:TextBox id="txtQuantity" runat="server">
</asp:TextBox>&nbsp;
<asp:CompareValidator ID="CompareValidator2" runat="server" ControlToValidate="txtQuantity"
Operator="DataTypeCheck" Type="Integer" ErrorMessage="Quantity must be an integer.">
</asp:CompareValidator>
```

RegularExpressionValidator:**Syntax:** <asp:RegularExpressionValidator>

- ⇒ It checks the value against a regular expression (pattern).
- ⇒ It checks that the value in the control matches a specified regular expression.
- ⇒ If the validated control is empty, no validation takes place.
- ⇒ The most important property is ValidationExpression.

Example:

```
<asp:RegularExpressionValidator runat="server" display="static" controlToValidate="txtH"
errorMessage="Hours must be 1-3 digits only" validationExpression="\d{1,3}">
</asp:RegularExpressionValidator>
```



CustomValidator:

Syntax: <asp:CustomValidator>

- ⇒ It allows us to develop custom validation.
- ⇒ It performs user-defined validation on an input using a specified function (client-side, server-side, or both).
- ⇒ If the validated control is empty, no validation takes place.
- ⇒ The most important property is ClientValidationFunction.

Properties of CustomValidator Control:

- **ClientValidationFunction:** Specifies the name of the client-side validation script function to be executed.
- **ControlToValidate:** The id of the control to validate
- **ErrorMessage:** The text to display in the ValidationSummary control when validation fails.
- **id:** A unique id for the control
- **IsValid:** A Boolean value that indicates whether the control specified by ControlToValidate is determined to be valid
- **OnServerValidate:** Specifies the name of the server-side validation script function to be executed
- **runat:** Specifies that the control is a server control. Must be set to "server"

Example:

```
protected void CustomValidator1:ServerValidate(object source, ServerValidateEventArgs args)
{
    string str = args.Value;
    args.IsValid = false;
    //checking for input length greater than 6 and less than 25 characters
    if (str.Length < 7 || str.Length > 20)
    {
        return;
    }
    //checking for a atleast a single capital letter
    bool capital = false;
    foreach (char ch in str)
    {
        if (ch >= 'A' && ch <= 'Z')
        {
            capital = true;
            break;
        }
    }
    if (!capital)
    {
        return;
    }
    bool digit = false;
    foreach (char ch in str)
    {
        if (ch >= '0' && ch <= '9')
        {
            digit = true;
            break;
        }
    }
    if (!digit)
    {
        return;
    }
    args.IsValid = true;
}
}
```



USE OF ASP.NET VALIDATION CONTROL

Q.21. How to Use the ASP.NET Validation Control to Validate the User Input.

ASKED YEAR → IDOL: Dec – 2017 | Oct – 2012

SOLUTION

Use of ASP.NET Validation Control:

To validate user input, there are many validation controls available in ASP.NET

- *CompareValidator*
- *CustomerValidator*
- *RangeValidator*
- *RegularExpressionValidator*
- *RequiredFieldValidator*

The following program shows how to use validation controls in ASP.NET.

PROGRAM:

```
<table>
<tr>
<td runat="server">First Name</td>
<td>
<asp:RequiredFieldValidator runat="server" ControlToValidate="TextBox1" ErrorMessage="First
Name cannot be blank" CssClass="errormsg">*</asp:RequiredFieldValidator>
</td>
</tr>
<tr>
<td runat="server">Middle Name<td>
<td>
<asp:RequiredFieldValidator runat="server" ControlToValidate="TextBox2" ErrorMessage="Middle
Name cannot be Blank" CssClass="errormsg">*</asp:RequiredFieldValidator>
</td>
<tr>
<tr>
<td runat="server">Sir Name</td>
<td>
<asp:RequiredFieldValidator runat="server" ControlToValidate="TextBox3" ErrorMessage="Sir Name
cannot be Blank" CssClass="errormsg">*</asp:RequiredFieldValidator>
</td>
</tr>
<tr>
<td>E-mail Address
<td>
<asp:TextBox runat="server">
</td>
<td>
<asp:RequiredFieldValidator runat="server" ControlToValidate="TextBox4" ErrorMessage="E-mail
address cannot be Blank">*</asp:RequiredFieldValidator>
<asp:RegularExpressionValidator runat="server" ControlToValidate="TextBox4" ErrorMessage="Enter
the valid Email id" ValidationExpression="\w+([-+.']\w+)*@\w+([-.\w+)*\.\w+([-
.] \w+)*">*</asp:RegularExpressionValidator>
</td>
</tr>
<tr>
<td>E-Mail Address again
<td>
<asp:TextBox runat="server">
</td>
<td>
```



```

<asp:RequiredFieldValidator runat="server" ControlToValidate="TextBox5" ErrorMessage="Email id
cannot be blank">*</asp:RequiredFieldValidator>
<asp:CompareValidator runat="server" ControlToCompare="TextBox4" ControlToValidate="TextBox5"
ErrorMessage="Wrong value">*</asp:CompareValidator>
</td>
</tr>
<tr>
<td>Age
<td>
<asp:TextBox runat="server">
</td>
<td>
<asp:RangeValidator runat="server" ControlToValidate="TextBox6" ErrorMessage="Please Enter the
age between 21 and 60" MaximumValue="60" MinimumValue="21" Type="Integer"
CssClass="errormsg">*</asp:RangeValidator>
<td>
</tr>
</table>

```

COMMON PROPERTIES FOR ALL VALIDATION CONTROLS

Q.22.

- Describe Common Validator properties for different Validation Controls. [IDOL: Oct – 2016]
- Write necessary properties which are common for all validation controls. [CBSGS: Nov – 2017]

ASKED YEAR ⇒

IDOL: Oct – 2016

CBSGS: Nov – 2017

SOLUTION

Common Properties for all Validation Controls:

- **ControlToValidate:** The id of the control to validate.
- **ErrorMessage:** The message to display in the control when validation fails.
- **Text:** Displays a text for validation control before validation.

STEPS TO SET UP VALIDATION CONTROL IN AN ASP.NET WEB PAGE

Q.23.

Define the steps to set up Validation Control in an ASP.NET Web Page.

ASKED YEAR ⇒

CBSGS: Apr – 2017

SOLUTION

Steps to set up Validation Control in an ASP.NET Web Page:

- STEP 1:** Draw a Validation Control on a Web form.
- STEP 2:** Set the `controlToValidate` property to the control you want to validate.
- STEP 3:** Specify the `ControlToCompare` property with compare Validator Control.
- STEP 4:** Specify Error Message to the validation control's `ErrorMessage` property.
- STEP 5:** Specify Error Message to the validation control's `Text` property, if you want to display a message in the Error Message property.
- STEP 6:** Draw a `ValidationSummary` Control on the Web form to display the error messages from the validation controls in one place.

CAUSEVALIDATION**Q.24.** Explain CauseValidation properties.

ASKED YEAR →

CBSGS: Apr – 2016

SOLUTIONCauseValidation:

- ⇒ User the CauseValidation property to determine whether validation is performed on both the client and the server when a TextBox Control is set to Validate when a postBack occurs.
- ⇒ Page Validation determines whether the input controls associated with a validation control on the page all pass the validation rules specified by the validation control.
- ⇒ By default, a TextBox Control cause page validation when the control focus. To set the TextBox control to validate when a postBack occurs, set the CauseValidation property to true & the AutoPostBack property to true.



| UNIT |

V

Databases And ADO.NET

LINQ

ASP.NET Security

TopicDATABASE AND ADO.NET

- ⇒ USING SQL TO WORK WITH DATABASE
- ⇒ TYPES OF SQL JOINS
 - > INNER JOIN
 - > OUTER JOIN
 - Left Outer Join
 - Right Outer Join
 - Full Outer Join
 - > CROSS JOIN
 - > SELF JOIN
- ⇒ STEPS FOR CONNECTING TO SQL SERVER DATABASE
- ⇒ ADO.NET OBJECTS
- ⇒ COMPONENTS OF ADO.NET OBJECTS
- ⇒ DATASET
 - > DATATABLE
- ⇒ DATA PROVIDER
 - > THE SQLCONNECTION CLASS
 - > THE SQLCOMMAND CLASS
 - > THE SQLDATAADAPTER CLASS
 - > THE SQLDATAREADER CLASS
 - > Provider Model
 - > DATAREADER Vs. DATAADAPTER
 - > DATASET Vs. DATAREADER
 - > EXECUTESCALAR Vs. EXECUTENONQUERY
- ⇒ ADO.NET
 - > MODE OF ADO.NET
 - Connected Mode Of ADO.NET
 - Disconnected Mode Of ADO.NET
 - > ACCESS DATABASE THROUGH ADO.NET
- ⇒ DATA BINDING
 - > TYPES OF DATA BINDING
 - > SINGLE-VALUE DATA BINDING
 - > REPEATED-VALUE DATA BINDING

- ⇒ DATABOUND CONTROL
- ⇒ TYPES OF DATABOUND
 - > LISTCONTROL
 - Multiple Item Control
 - ListBox Control
 - CheckBoxLayout Control
 - Single Item Control
 - DropDownList Control
 - RadioButtonList Control
 - BulletedList Control
- ⇒ GRIDVIEW CONTROL
- ⇒ DATA SOURCE CONTROL
 - > TYPES OF DATA SOURCE CONTROLS
 - Hierarchical Data Source Controls
 - Table-Based Data Source Controls
- ⇒ DETAILSVIEW CONTROL
- ⇒ FORMVIEW CONTROL
- ⇒ FORMVIEW CONTROL Vs. DETAILVIEW CONTROL
- ⇒ LISTVIEW Vs. GRIDVIEW

LANGUAGE INTEGRATED QUERY (LINQ)

- ⇒ LINQ QUERY SYNTAX
- ⇒ ADVANTAGES OF LINQ
- ⇒ DISADVANTAGES OF LINQ
- ⇒ TYPES OF LINQ OPERATIONS
 - > FILTERING OPERATORS
 - > JOIN OPERATORS
 - > PROJECTION OPERATIONS
 - > SORTING OPERATORS
 - > GROUPING OPERATORS
 - > CONVERSIONS
 - > CONCATENATION
 - > AGGREGATION
 - > QUANTIFIER OPERATIONS



- > ADVANTAGES OF DATA BINDING
- > DISADVANTAGES OF DATA BINDING
- ⇒ DEPLOYMENT
- ⇒ XPATH

- > PARTITION OPERATIONS
- > GENERATION OPERATIONS
- > SET OPERATIONS
- > EQUALITY
- > ELEMENT OPERATORS

- ⇒ LINQ QUERY OPERATORS
- ⇒ LINQ STANDARD QUERY OPERATORS
 - > SELECT
 - > FROM
 - > ORDERBY
 - > WHERE
- ⇒ IMPLEMENTATIONS
 - > QUERY EXPRESSION (QUERY SYNTAX)
 - > METHOD INVOCATION (METHOD SYNTAX)
 - > MIXED SYNTAX
- ⇒ QUERY SYNTAX AND EXTENSION METHODS CAN BE MIXED
- ⇒ LINQ TO OBJECTS
- ⇒ LINQ TO XML
- ⇒ LINQ TO SQL
- ⇒ LINQ VS. SQL

ASP.NET SECURITY

- ⇒ AUTHENTICATION
 - > TYPES OF AUTHENTICATION
 - Windows Authentication
 - Basic Authentication
 - Anonymous Authentication
 - Digest Authentication
 - Integrated Windows Authentication
 - Forms Authentication
 - Passport Authentication
- ⇒ AUTHORIZATION
 - > FILE AUTHORIZATION
 - > URL AUTHORIZATION
- ⇒ IMPERSONATION
- ⇒ AUTHORISATION VS. IMPERSONATION

**USING SQL TO WORK WITH DATABASE****Q.1. Explain type of joins in SQL Server.**ASKED YEAR → IDOL: Dec – 2017**SOLUTION****Types Of SQL Joins:**

- ⇒ In SQL Server we have only three types of joins.
- ⇒ Using these joins we fetch the data from multiple tables based on condition.

(1) Inner Join:

Inner join returns only those records/rows that match/exists in both the tables.

Syntax for Inner Join is as

```
Select * from table_1 as t1
inner join table_2 as t2
on t1.IDcol=t2.IDcol
```

(2) Outer Join:

We have three types of Outer Join.

1) Left Outer Join:

- ⇒ Left outer join returns all records/rows from left table and from right table returns only matched records.
- ⇒ If there are no columns matching in the right table, it returns NULL values.

Syntax for Left outer Join is as:

```
Select * from table_1 as t1
left outer join table_2 as t2
on t1.IDcol=t2.IDcol
```

2) Right Outer Join:

- ⇒ Right outer join returns all records/rows from right table and from left table returns only matched records.
- ⇒ If there are no columns matching in the left table, it returns NULL values.

Syntax for Right Outer Join is as:

```
Select * from table_1 as t1
right outer join table_2 as t2
on t1.IDcol=t2.IDcol
```

3) Full Outer Join:

- ⇒ Full outer join combines left outer join and right outer join. This join returns all records/rows from both the tables.
- ⇒ If there are no columns matching in the both tables, it returns NULL values.

Syntax for Full Outer Join is as:

```
Select * from table_1 as t1
full outer join table_2 as t2
on t1.IDcol=t2.IDcol
```

(3) Cross Join:

- ⇒ Cross join is a Cartesian join means Cartesian product of both the tables.
- ⇒ This join does not need any condition to join two tables.



⇒ This join returns records/rows that are multiplication of record number from both the tables means each row on left table will related to each row of right table.

Syntax for Cross Outer Join is as:

```
Select * from table_1  
cross join table_2
```

(4) Self Join:

- ⇒ Self Join is used to join a database table to itself, particularly when the table has a foreign key that references its own Primary Key.
- ⇒ Basically we have only three types of joins: Inner join, Outer join and Cross join.
- ⇒ We use any of these three JOINS to join a table to itself.
- ⇒ Hence Self join is not a type of SQL join.

STEPS FOR CONNECTING TO SQL SERVER DATABASE

Q.2. Write the necessary steps for connecting to the SQL Server Database.

ASKED YEAR ⇒ IDOL: May – 2018

SOLUTION

Steps For Connecting To The SQL Server Database:

STEP 1:

⇒ Open Visual Studio, go to the File >> New >> Project or use the shortcut key "Ctrl + Shift + N".

STEP 2:

⇒ Here, select Visual C# Web >> ASP.NET Web Application. Finally, click "OK" button.

STEP 3:

- ⇒ Here, we can select the template for our ASP.NET Application.
- ⇒ We are choosing "Empty" here.
- ⇒ Now, click "OK" button.

STEP 4:

- ⇒ Now, open the project and look for the Solution Explorer.
- ⇒ Here, open the "default.aspx". If we want a WebForm, we can add the page (Web Form).
- ⇒ Add + New item (Ctrl + Shift + A).
- ⇒ Now, we can create a login page, using ASP.NET code. We need to follow the drag and drop method. Here, we already created the login page.

STEP 5:

- ⇒ Open the project, if we want a SQL Server Database, we can add the page (SQL Server database) Add + New item (Ctrl + Shift + A).
- ⇒ Here, we can select Visual C# and choose SQL Server Database. Afterwards, click "OK" button.
- ⇒ Now, open the new Window and click "YES" button.
- ⇒ Now, add this to the database in our project.

STEP 6:

- ⇒ Go to the "Server Explorer" and add your Database. Click the "Tables and afterwards", click "Add New Table".
- ⇒ Open the New Table and you can fill the data, which is like (studentname, password) and afterwards, you click the "Update".



- ⇒ Here, click Database in update and subsequently click "update the database".
- ⇒ Here, the Database is updated.
- ⇒ Here, the Database and data are added.
- ⇒ Now, click on the "Right Click" and Click "Show Table Data".
- ⇒ Now, the data is stored.

STEP 7:

- ⇒ Add "SQL Data Source". Drag and drop method. Here, click "Configure Data Source".
- ⇒ Choose your database and click "NEXT" button.
- ⇒ Select the "ConnectionString" and Click "NEXT" button.
- ⇒ Choose Specify columns from a table or view and afterwards, click "Next" button.
- ⇒ Now, click "Test Query". Here, add the data and click "Finish" button.

STEP 8:

- ⇒ Now, we can go to CS (C# Code) page and we will write the C# code.

CODE:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data.SqlClient;
using System.Configuration;
namespace DatabaseConnectivity
{
    public partial class loginpage : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (IsPostBack)
            {
                SqlConnection conn = new SqlConnection
                (ConfigurationManager.ConnectionStrings["RegiConnectionString"].ConnectionString);
                conn.Open();
                string checkuser = "select count(*) from RegisterDataBase where
                StudentName='"+TextBox1.Text+"'";
                SqlCommand cmd = new SqlCommand(checkuser, conn);
                int temp = Convert.ToInt32(cmd.ExecuteScalar().ToString());
                if (temp == 1)
                {
                    Response.Write("Student Already Exist");
                }
                conn.Close();
            }
        }
        protected void Button1_Click(object sender, EventArgs e)
        {
            try
            {
                SqlConnection conn = new SqlConnection
                (ConfigurationManager.ConnectionStrings["RegiConnectionString"].ConnectionString);
                conn.Open();
                string insertQuery = "insert into
                RegisterDataBase (StudentName, Passwords, EmailId, Department, College) values
                (@studentname, @passwords, @emailid, @department, @college)";
                SqlCommand cmd = new SqlCommand(insertQuery, conn);
```



```
cmd.Parameters.AddWithValue("@studentname", TextBox1.Text);
cmd.Parameters.AddWithValue("@passwords", TextBox2.Text);
cmd.Parameters.AddWithValue("@emailid", TextBox3.Text);
cmd.Parameters.AddWithValue("@department", TextBox4.Text);
cmd.Parameters.AddWithValue("@college", TextBox5.Text);
cmd.ExecuteNonQuery();
Response.Write("Student registration Successfully!!!thank you");
conn.Close();
}
catch (Exception ex)
{
Response.Write("error" + ex.ToString());
}
}
}
```

CODE: Now, you can see the Loginpage.aspx code.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="loginpage.aspx.cs"
Inherits="DatabaseConnectivity.loginpage" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
<link href="stylepage.css" type="text/css" rel="stylesheet" />
<style type="text/css">
.auto-style1 {width 100%;}
</style>
</head>
<body>
<form id="form1" runat="server">
<div id="title">
<h1>REGISTER PAGE</h1>
</div>
<div id="teble"></div>
<table class="auto-style1">
<tr>
<td>
<aspLabel ID="Label1" runat="server" Text="StudentName"></aspLabel></td>
<td>
<aspTextBox ID="TextBox1" runat="server"></aspTextBox></td>
</tr>
<tr>
<td>
<aspLabel ID="Label2" runat="server" Text="Password"></aspLabel></td>
<td>
<aspTextBox ID="TextBox2" runat="server"></aspTextBox></td>
</tr>
<tr>
<td>
<aspLabel ID="Label3" runat="server" Text="EmailId"></aspLabel></td>
<td>
<aspTextBox ID="TextBox3" runat="server"></aspTextBox></td>
</tr>
<tr>
<td>
<aspLabel ID="Label4" runat="server" Text="Department"></aspLabel></td>
<td>
<aspTextBox ID="TextBox4" runat="server"></aspTextBox></td>
</tr>
</table>
</div>
</form>
</body>
</html>
```



```
<td>
<aspLabel ID="Label5" runat="server" Text="College"></aspLabel></td>
<td>
<aspTextBox ID="TextBox5" runat="server"></aspTextBox></td>
</tr>
</table>
<div id="button">
<aspButton ID="Button1" runat="server" Text="submit" OnClick="Button1_Click" BackColor="Yellow"
/>
</div>
<div id="sim"></div>
<aspSqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%=
ConnectionStringRegiConnectionString %>" SelectCommand="SELECT * FROM
[RegisterDataBase]"></aspSqlDataSource>
<div id="grid">
<aspGridView ID="GridView1" runat="server" AllowPaging="True" AllowSorting="True"
AutoGenerateColumns="False" CellPadding="4" DataSourceID="SqlDataSource1" ForeColor="#333333"
GridLines="None">
<AlternatingRowStyle BackColor="White" ForeColor="#284775" />
<Columns>
<aspBoundField DataField="Id" HeaderText="Id" SortExpression="Id" />
<aspBoundField DataField="StudentName" HeaderText="StudentName" SortExpression="StudentName" />
<aspBoundField DataField="Passwords" HeaderText="Passwords" SortExpression="Passwords" />
<aspBoundField DataField="EmailId" HeaderText="EmailId" SortExpression="EmailId" />
<aspBoundField DataField="Department" HeaderText="Department" SortExpression="Department" />
<aspBoundField DataField="College" HeaderText="College" SortExpression="College" />
</Columns>
<EditRowStyle BackColor="#999999" />
<FooterStyle BackColor="#5D7B9D" -Bold="True" ForeColor="White" />
<HeaderStyle BackColor="#5D7B9D" -Bold="True" ForeColor="White" />
<PagerStyle BackColor="#284775" ForeColor="White" HorizontalAlign="Center" />
<RowStyle BackColor="#F7F6F3" ForeColor="#333333" />
<SelectedRowStyle BackColor="#E2DED6" -Bold="True" ForeColor="#333333" />
<SortedAscendingCellStyle BackColor="#E9E7E2" />
<SortedAscendingHeaderStyle BackColor="#506C8C" />
<SortedDescendingCellStyle BackColor="#FFFDF8" />
<SortedDescendingHeaderStyle BackColor="#6F8DAE" />
</aspGridView>
</div>
<div id="last">
<h3>Developed by B.Sc.IT</h3>
</div>
</form>
</body>
</html>
```

STEP 9:

- ⇒ Here, you need to run any Browser and after a few minutes, you will get some output.
- ⇒ Now, we can insert the data into your Database.

STEP 10:

- ⇒ Now, we can added the GridView.
- ⇒ Drag and drop method needs to be used.
- ⇒ Now, you can choose the data source, it is SqlDataSource.
- ⇒ Here, the database data is added to GridView.

STEP 11:

Here, you need to run any Browser and after a few minutes, we will get an output. Now, we can view the data.

**ADO.NET OBJECTS****Q.3.**

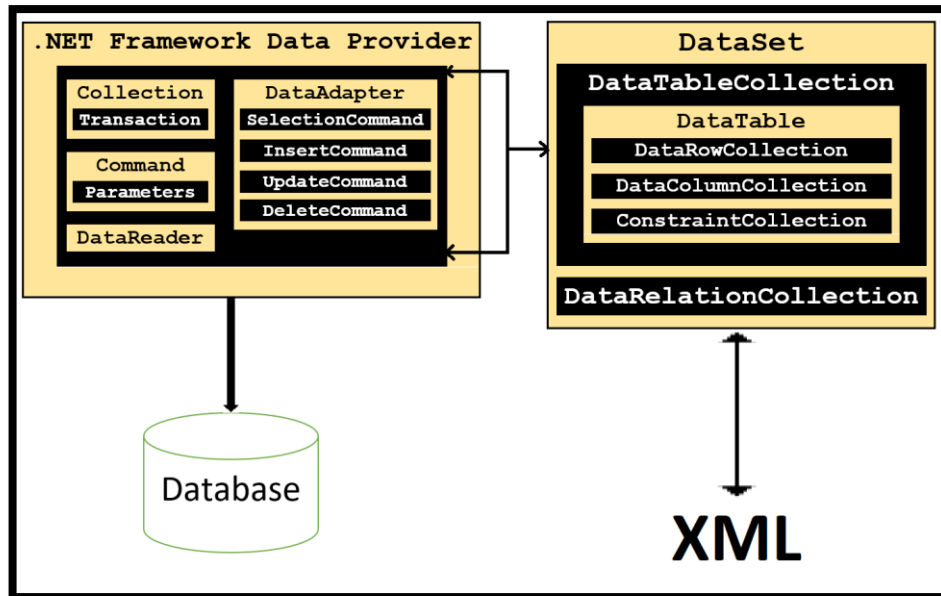
- Explain ADO.NET Object Model with help of suitable diagram. [IDOL: May – 2018 | May – 2017 | May – 2016 | Apr – 2015 | Apr – 2014 | Apr – 2013 | Oct – 2012]
- Draw and explain the architecture of ADO.NET in brief. [CBSGS: Nov – 2016]
- Explain SqlDataAdapter Class with properties and methods. [CBSGS: Nov/Apr – 2017 | Nov – 2016 | Nov – 2015 | Apr – 2014]
- Explain SqlConnection Objects in ADO.NET. [CBSGS: Nov – 2016 | Nov – 2015]
- What is SqlDataSource Control used in ADO.Net? [IDOL: Oct – 2016]
- Explain SqlCommand Object in ADO.NET. [CBSGS: Apr – 2017 | Nov/Apr – 2015 | Apr – 2014] | [IDOL: May – 2016]
- Describe DataReader Object of ADO.NET with example. [CBSGS: Apr – 2016 | Nov/Apr – 2015]
- Explain ExecuteReader, ExecuteNonQuery and ExecuteScalar methods. [CBSGS: Apr – 2015] | [IDOL: Dec – 2017]

ASKED YEAR ⇒

IDOL: May – 2018 | Dec/May – 2017 | Oct/May – 2016 | Apr – 2015 | CBSGS: Nov/Apr – 2017 | Apr/Nov – 2016 | Nov/Apr – 2015 | Apr – 2014 | Apr – 2013 | Oct – 2012

SOLUTION**ADO.NET Objects:**

Data Access in ADO.NET relies on Two Components: DataSet and Data Provider

**DataSet:**

- ⇒ The DataSet is a disconnected, in-memory representation of data.
- ⇒ It can be considered as a local copy of the relevant portions of the database.
- ⇒ The DataSet is persisted in memory and the data in it can be manipulated and updated independent of database.
- ⇒ When the use of this DataSet is finished, changes can be made back to the central database for updating.
- ⇒ The data in DataSet can be loaded from any valid data source like Microsoft SQL Server database, an Oracle database or from a Microsoft Access database. It contains one or more tables, in addition to their relationship and constraint information.

DataTable:

- ⇒ It used to represent a table in the DataSet.
- ⇒ It consists of collection of rows and columns.
- ⇒ Required columns can be added to the DataTable using the Columns collection.



DataProvider:

- ⇒ The `DataProvider` is responsible for providing and maintaining the connection to the database.
- ⇒ A `DataProvider` is a set of related components that work together to provide data in an efficient and performance driven manner.
- ⇒ The .NET Framework currently comes with two `DataProvider`: the `SQL DataProvider` which is designed only to work with Microsoft's SQL Server 7.0 or later and the `OleDb DataProvider` which allows us to connect to other types of databases like Access and Oracle.
- ⇒ Each `DataProvider` consists of the following components classes:
 - *The `SqlConnection Class`*
 - *The `SqlCommand Class`*
 - *The `SqlDataAdapter Class`*
 - *The `SqlDataReader Class`*

The `SqlConnection Class`:

- ⇒ The `SqlConnection Class` is used to create the connection to the database.
- ⇒ Microsoft Visual Studio .NET provides two types of Connection Classes: the `SqlConnection Object`, which is designed specifically to connect to Microsoft SQL Server 7.0 or later, and the `OleDbConnection object`, which can provide connections to a wide range of database types like Microsoft Access and Oracle. The connection object contains all of the information required to open a connection to the database.
- ⇒ Common property of the `SqlConnection Class` are – `ConnectionString`, `Open ()` and `Close ()`

Properties of `SqlConnection Class`:

- **AccessToken:** Gets or sets the access token for the connection.
- **ConnectionString:** Gets or sets the string used to open a SQL Server database.
- **ConnectionTimeout:** Gets the time to wait while trying to establish a connection before terminating the attempt and generating an error.
- **Container:** Gets the `IContainer` that contains the Component.
- **Database:** Gets the name of the current database or the database to be used after a connection is opened.

The `SqlCommand Class`:

- ⇒ The `SqlCommand` object class is represented by two corresponding classes: `SqlCommand` and `OleDbCommand`.
- ⇒ Command Objects are used to execute SQL Commands against a SQL Server Database.
- ⇒ Command Objects provide three methods (i.e. `ExecuteNonQuery`, `ExecuteScalar` & `ExecuteReader`) that are used to execute commands on the database.

Properties of `SqlCommand Class`:

- **CommandText:** Contains the text of a SQL query.
- **CommandTimeout:** Contains the length of the timeout of a query, in seconds.
- **CommandType:** Specifies the type of command to be executed.
- **Connection:** Specifies the connection to the database.
- **Parameters:** Specifies a collection of parameters for the SQL query.
- **Transaction:** Specifies a transaction object, which enables developers to run queries in a transaction.

Methods of `SqlCommand Class`:

`SqlCommand.ExecuteReader ()` – The `ExecuteReader ()` method executes the command text against the database specified in the Connection object and returns a `SqlDataReader` object with the results of the query.

Syntax:

```
SqlDataReader ExecuteReader ()  
SqlDataReader ExecuteReader (CommandBehavior behavior)
```

Example:

```
SqlConnection conn = new SqlConnection("Data Source=localhost;
```



```
User Id=sa; Password=; Initial Catalog=northwind");
conn.Open();
SqlCommand cmd = new SqlCommand("SELECT * FROM Customers", conn);
SqlDataReader reader = cmd.ExecuteReader();
```

`SqlCommand.ExecuteScalar()` - The `ExecuteScalar()` method executes the command text against the database specified in the Connection object and returns a single object. The `ExecuteScalar()` method exists because it is wasteful to return a dataset for a single value. The overhead for the dataset would be much larger than the actual value being returned.

Syntax:

```
Object ExecuteScalar()
```

Example:

```
SqlConnection conn = new SqlConnection("Data Source=localhost;
User Id=sa; Password=; Initial Catalog=northwind");
conn.Open();
SqlCommand cmd = new SqlCommand("SELECT count(*) FROM Customers", conn);
Int32 customerCount = (Int32)cmd.ExecuteScalar();
msg.Text = "There are "+customerCount.ToString()+" customers in the database.";
```

`SqlCommand.ExecuteNonQuery()` - The `ExecuteNonQuery()` method executes the command text against the database specified in the Connection object. This method is optimized for queries that do not return any information (for example, DELETE and UPDATE queries).

Syntax:

```
Int32 ExecuteNonQuery()
```

Example:

```
SqlConnection conn = new SqlConnection("Data Source=localhost;
User Id=sa; Password=; Initial Catalog=northwind");
conn.Open();
SqlCommand cmd = new SqlCommand("DELETE FROM Customers WHERE LastName='Jones'", conn);
cmd.ExecuteNonQuery();
```

The SqlDataAdapter Class:

- ⇒ The `SqlDataAdapter` is the class at the core of ADO.NET's disconnected data access.
- ⇒ The `DataAdapter` object which populates a disconnected `DataSet` with data and performs update.
- ⇒ It is essentially the middleman facilitating all communication between the database and a `DataSet`.
- ⇒ The `DataAdapter` is used either to fill a `DataTable` or `DataSet` with data from the database with its `Fill` Method.
- ⇒ After the memory-resident data has been manipulated, the `DataAdapter` can commit the changes to the database by calling the `Update` method.

Properties:

- **DeleteCommand**: Represents a DELETE statement or stored procedure for deleting records from the Data Source.
- **InsertCommand**: Represents an INSERT statement or stored procedure for inserting a new record to the Data Source.
- **SelectCommand**: Represents a SELECT statement or stored procedure can be used to select records from a data source.
- **UpdateCommand**: Represents an UPDATE statement or stored procedure for Updating recording in a Data Source.

Methods:

- **Fill()**: This method fills data records from a `DataAdapter` to a `DataSet` object.
- **FillSchema()**: This method adds a `DataTable` to a `DataSet`.

The SqlDataReader Class:

- ⇒ It is connected, forward, read-only, stream of Data (i.e. used to read sequential collection of records from database).
- ⇒ It is faster than `DataSet` but requires an open connection.



- ⇒ The `DataReader` Object provides a connection oriented data access to the Data Sources.
- ⇒ A Connection Object can contain only one `DataReader` at a time and the connection in the `DataReader` remains open, also it cannot be used for any other purpose while data is being accessed.
- ⇒ When we started to read from a `DataReader` it should always be open and positioned prior to the first record.
- ⇒ The `Read()` method in the `DataReader` is used to read the rows from `DataReader` and it always moves forward to a new valid row, if any row exist .

PROVIDER MODEL

Q.4. Explain ASP.NET Provider Model with its diagram.

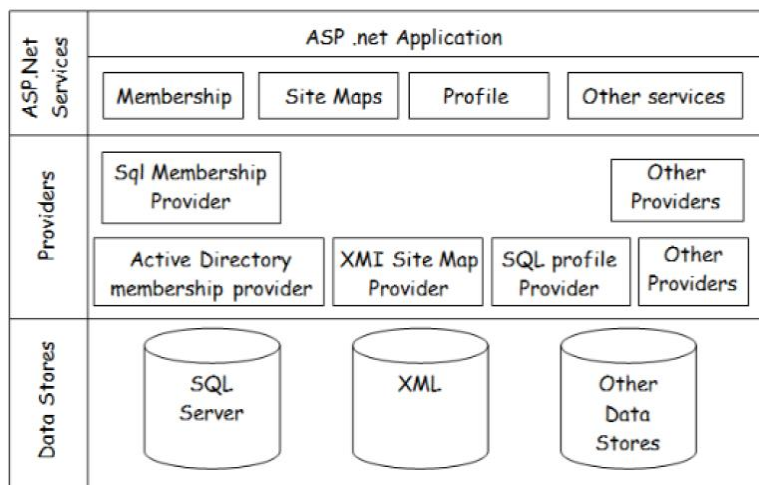
ASKED YEAR ⇒

IDOL: May – 2016

CBSGS: Apr – 2016 | Apr – 2015 | Apr – 2014

SOLUTION

Provider Model:



The following are the built-in providers of the ASP.NET Provider Model:

- Membership Providers
- Role Management Providers
- Profile Providers
- Site Map Providers
- Session State Providers
- Web Event Providers

Membership Providers:

- ⇒ ASP.NET Membership gives you a built in a way to validate and store user credentials.
- ⇒ The membership provider is used to manage the users of the application and their respective roles.
- ⇒ It is extensible, but can work with SQL Server Database and Active Directory.

Role Management Providers:

- ⇒ A role provider is responsible for retrieval and storage of roles information to and from a data store.
- ⇒ You've the SQL Role Provider that is responsible for storing roles information in the SQL Server Database.

Profile Providers:

- ⇒ Profile Providers can be used to store users profile information.
- ⇒ It provides the interface between ASP.NETs profile service & Profile Data Sources.

Custom Providers:

- ⇒ Custom Providers are there that we create generally by extending the existing providers to suit our specific requirements.



⇒ The following are the benefits of creating custom providers:

- Storage of membership information across any database or other data sources that are not supported by default.
- Managing the membership information as per your own schema.

DATA READER VS. DATA ADAPTER

Q.5. What is the difference between DataReader and DataAdapter? Explain.

ASKED YEAR ⇒ IDOL: May – 2018 | Apr – 2014

CBSGS: Apr – 2015 | Nov – 2014

SOLUTION

DataReader Vs. DataAdapter:

<u>DataReader</u>	<u>DataAdapter</u>
<u>Database Connecting Mode</u>	
Connected mode.	Disconnected mode.
<u>Data Navigation</u>	
Unidirectional (i.e. Forward Only).	Bidirectional (i.e. data can navigate back and forth).
<u>Read/Write</u>	
Only Read operation can be carried out.	Both Read and Write operations are possible.
<u>Data Handling</u>	
Handles Database table.	Handles text file, XML file and Database table.
<u>Storage Capacity</u>	
No storage.	Temporary Storage Capacity (i.e. in-memory cache of data).
<u>Accessibility</u>	
Can access one table at a time.	Can access multiple table at a time.
<u>Speed</u>	
Much faster than DataSet.	Less faster than DataReader.

DATA SET VS. DATA READER

Q.6. Differentiate between DataSet and DataReader.

ASKED YEAR ⇒ IDOL: May – 2016 | Apr – 2013

CBSGS: Apr – 2017 | Apr – 2016 | Apr – 2015

SOLUTION

DataSet Vs. DataReader:

<u>DataSet</u>	<u>DataReader</u>
DataSet Used in a disconnected architecture.	DataReader Used in a connected architecture.
DataSet Provides lower performance.	DataReader Provides better performance.
DataSet Object has read/write access.	DataReader Object has read-only access.
DataSet Object supports multiple tables from various databases.	DataReader Object supports a single table based on a single SQL query of one database.
DataSet Object is bound to multiple controls.	DataReader Object is bound to a single control.
DataSet Object has slower access to data.	DataReader Object has faster access to data.
DataSet Object is supported by Visual Studio tools.	DataReader Object must be manually coded.
We can create relations in a DataSet.	We can't create a relation in a DataReader.
DataSet supports integration with XML Dataset communicates with the Data Adapter only.	Whereas a DataReader doesn't support data reader communicates with the command object.
DataSet can modify data.	DataReader cannot modify data.



EXECUTE SCALAR VS. EXECUTE NON QUERY

Q.7. What is the difference between ExecuteScalar and ExecuteNonQuery?

ASKED YEAR →

CBSGS: Nov – 2017

SOLUTION

ExecuteScalar Vs. ExecuteNonQuery:

ExecuteScalar () :

- ⇒ ExecuteScalar will return single row single column value i.e. single value, on execution of SQL Query or Stored procedure using command object. It's very fast to retrieve single values from database. Used to execute SQL Select command which is used to return a single value.
- ⇒ ExecuteScalar only returns the value from the first column of the first row of your query.

Example:

```
string result = (string)cmd.ExecuteScalar();
```

Where cmd is an object of SqlCommand class.

ExecuteNonQuery () :

- ⇒ ExecuteNonQuery method will return number of rows effected with INSERT, DELETE or UPDATE operations.
- ⇒ This ExecuteNonQuery method will be used only for insert, update and delete, Create, and SET statements.
- ⇒ ExecuteNonQuery does not return data at all. It returns only the number of rows affected by an insert, update, or delete.

Example:

```
int result= cmd.ExecuteNonQuery();
```

Where cmd is an object of SqlCommand class.

STEPS TO ACCESS DATABASE THROUGH ADO.NET

Q.8. List all the steps in order to access a database through ADO.NET.

ASKED YEAR →

IDOL: Dec – 2017 | Apr – 2014 | Apr – 2013 | Oct – 2012

CBSGS: May – 2017

SOLUTION

Steps in order to access a Database through ADO.NET:

Import the namespace using `System.Data.SqlClient;`

STEP 1: Create the SQL Connection class's Object

```
SqlConnection conn=new SqlConnection();
```

STEP 2: In the Button_Click or Page_Load event set the Connection String property of connection object in the following way.

```
conn.ConnectionString=("Data Source=ACER-PC;  
Initial Catalog=master;Integrated SQLDataAdapt
```

STEP 3: Write SQL Query and store in a string variable.

```
statmnt="Select * from Product1";
```

STEP 4: Instantiate the SqlDataAdapter class and set the SelectCommand Property

```
SqlDataAdapter adapter=new SqlDataAdapter();  
adapter.SelectCommand=new SqlCommand(statmnt, conn);
```

**STEP 5: Instantiate the DataSet Class**

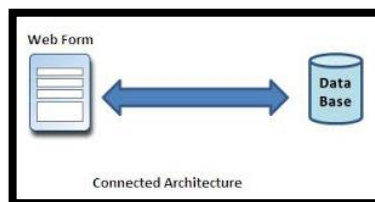
```
DataSet ds=new DataSet
```

STEP 6: In the Button_Click event or Page_Load event, write the following code and design the DataGridView control in the Design View.

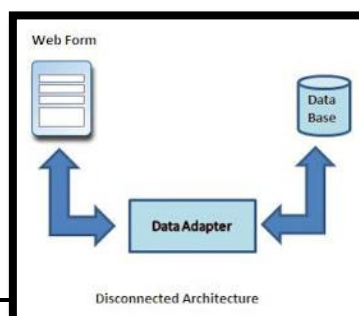
```
using System.Data.SqlClient;
//using System.Data;
public partial class_Default: System.Web.UI.Page
{
    string statmnt="";
    SqlDataAdapter adapter=new SqlDataAdapter();
    SqlConnection conn=new SqlConnection();
    DataSet ds=new DataSet();
    protected void Button1_Click(object sender, EventArgs e)
    {
        conn.ConnectionString="Data Source=ACER-PC;InitialCatalog=master;IntegratedSecurity=True";
        statmnt="Select * from Product1";
        adapter.SelectCommand=new SqlCommand(statmnt, conn);
        conn.Open();
        adapter.Fill(ds, "Prd");
        conn.Close();
        GridView3.DataSource=ds;
        GridView3.DataBind();
    }
}
```

MODE OF ADO.NET**Q.9. Explain in brief Connected and Disconnected Mode of ADO.NET.**

ASKED YEAR → IDOL: Oct – 2016

SOLUTION**Connected Architecture of ADO.NET:**

- ⇒ The architecture of ADO.net, in which connection must be opened to access the data retrieved from database is called as connected architecture.
- ⇒ Connected architecture was built on the classes connection, command, DataReader and transaction.
- ⇒ Connected architecture is when you constantly make trips to the database for any CRUD (Create, Read, Update and Delete) operation you wish to do.
- ⇒ This creates more traffic to the database but is normally much faster as you should be doing smaller transactions.
- ⇒ DataReader is Connected Architecture since it keeps the connection open until all rows are fetched one by one.

Disconnected Architecture In ADO.NET:



- ⇒ The architecture of ADO.net in which data retrieved from database can be accessed even when connection to database was closed is called as disconnected architecture.
- ⇒ Disconnected architecture of ADO.net was built on classes `connection`, `DataAdapter`, `commandbuilder` and `dataset` and `dataview`.
- ⇒ Disconnected architecture is a method of retrieving a record set from the database and storing it giving you the ability to do many CRUD (Create, Read, Update and Delete) operations on the data in memory, then it can be re-synchronized with the database when reconnecting.
- ⇒ A method of using disconnected architecture is using a Dataset.
- ⇒ `DataSet` is Disconnected Architecture since all the records are brought at once and there is no need to keep the connection alive.

DATA BINDING

Q.10. What is Data Binding? Explain its different types.

ASKED YEAR ⇒

IDOL: Apr – 2015

CBSGS: Apr – 2016

SOLUTION

Data Binding:

- ⇒ Data Binding is binding controls to data from databases. With data binding we can bind a control to a particular column in a table from the database or we can bind the whole table to the data grid.
- ⇒ Data binding provides simple, convenient, and powerful way to create a read/write link between the controls on a form and the data in their application.
- ⇒ Data binding allows you to take the results of properties, collection, method calls, and database queries and integrate them with your ASP.NET code.
- ⇒ We can combine data binding with Web control rendering to relieve much of the programming burden surrounding Web control creation.
- ⇒ We can also use data binding with ADO.NET and Web controls to populate control contents from SQL select statements or stored procedures.

Types Of Data Binding:

There are two types of Data Binding exist:

- (i) Single-Value Data Binding
- (ii) Repeated-Value Data Binding

Single-Value Data Binding:

- ⇒ You can use Single-Value Data Binding to add information anywhere on ASP.NET page. You can even place information into a control property or as plain text inside an HTML tag.
- ⇒ Single-Value Data Binding allows you to take a variable, a property or an expression and insert it dynamically into a page.
- ⇒ Single-Value Data Binding is really just a different approach to dynamic text. To use it, you add special data binding expressions into your .aspx files. These expressions have the following format:
- ⇒ `<%# expression_goes_here %>`

Repeated-Value Data Binding:

- ⇒ Repeated-value data binding allows you to display an entire table (or just a single field from a table). Unlike single-value data binding, this type of data binding requires a special control that supports it (e.g. list control such as `checkBoxList` or `ListBox`).
- ⇒ A control supports `repeated_value` data binding if it provides a `DataSource` property.
- ⇒ You can use `repeated_value` binding to bind data from a collection or an array.



Advantages And Disadvantages Of Data Binding:

Advantages

- Databinding in .NET can be used to write data driven applications quickly. .NET data binding allows you to write less code with fast execution but still get the work done in the best way.
- .NET automatically writes a lot of databinding code for you in the background (you can see it in "Windows Generated Code" section), so the developer does not have to spend time writing code for basic databinding, but still has the flexibility of modifying any code that he would like to. We get the benefits of bound as well as unbound approach.
- Control over the Databinding process by using events. This is discussed in more detail later in the article.

Disadvantages

- More optimized code can be written by using the unbound or traditional methods.
- Complete flexibility can only be achieved by using the unbound approach.

DATABOUND CONTROL

Q.11.➤ **What is DataBound Control?** [IDOL: Apr – 2013] | [CBSGS: Nov – 2016]➤ **Explain in brief about Single Item Control.** [IDOL: Dec – 2017 | Apr – 2014]

ASKED YEAR →

IDOL: Dec – 2017 | Apr – 2014 | Apr – 2013

CBSGS: Nov – 2016

SOLUTION

DataBound Control:

DataBound web server controls are controls that can be bound to a data source control to make it easy to display and modify data in our web application. All of these controls provide a variety of properties that we can set to control the appearance of the UI that they generate. DataBound web server controls are composite controls that combine other ASP.NET web controls, such as Label and TextBox controls, into a single layout.

Types Of DataBound:

- **Simple** – inherit from ListControl
- **Composite** – inherit from CompositeDataBoundControl, e.g. GridView, DetailsView, FormView
- **Hierarchical** – Menu and TreeView controls
- **Visualisation** – ChartControl

ListControl:

- ⇒ Not designed to work with pages of data or provide complicated editing.
- ⇒ Provide a list of items on which user can operate.
- ⇒ Based on abstract ListControl base class.
- ⇒ ListControl exposed Items collection which contains ListItem objects.
- ⇒ Each ListItem has Text property displayed to user and Value property posted back to server.
- ⇒ Can add items to Items collection via code or markup.
- ⇒ ListControl can be bound to data sources.
- ⇒ Can specify which fields will bind to the Text and Value properties either through declarative markup or the DataTextField and DataValueField properties.

Multiple Item Control:

ListBox Control:

- ⇒ Displays longer list and allows user to (optionally via SelectionMode property) select multiple values.
- ⇒ Rows property controls how many rows are displayed on screen at a time.
- ⇒ Determine which items selected by iterating over Items collection and examining the Selected property for each element.

CheckBoxList Control:

- ⇒ Display lists allowing users to make selections via a checkbox.
- ⇒ CheckBoxList allows for multiple selections.
- ⇒ The RepeatColumns property indicates number of columns to be displayed.
- ⇒ The RepeatDirection (Vertical by default) indicates whether data should be rendered horizontally or vertically.

Single Item Control:DropDownList Control:

- ⇒ Displays list and allows user to make a single selection.
- ⇒ SelectedValue indicates which entry has been chosen.

RadioButtonList Control:

- ⇒ RadioButtonList supports single selection.
- ⇒ The RepeatColumns property indicates number of columns to be displayed.
- ⇒ The RepeatDirection (Vertical by default) indicates whether data should be rendered horizontally or vertically.
- ⇒ The RadioButtonList exposed SelectedValue to indicate which item has been selected.

BulletedList Control:

- ⇒ Displays list as ordered or unordered list in HTML.
- ⇒ BulletStyle property controls whether rendered as bulleted (disc, circle or square) or numbered (LowerAlpha, UpperAlpha, LowerRoman, UpperRoman).
- ⇒ FirstBulletNumber controls starting number of sequence.
- ⇒ DisplayMode can take a value of Text, LinkButton or HyperLink – latter two causing postback to raise a Click event.

GRIDVIEW CONTROL

Q.12.

- Explain GridView Control with example. [CBSGS: Nov – 2016]
- What is the use of GridView Control in ASP.NET? [CBSGS: Oct – 2013]
- What is GridView Control? Explain how to enable Row Selection, Paging and Sorting features of GridView. [CBSGS: Nov – 2015]
- Explain any four properties of GridView Control. [IDOL: Apr – 2015]
- Explain in brief about Paging Control. [IDOL: Dec – 2017 | Apr – 2014]

ASKED YEAR ⇒

IDOL: Dec – 2017 | May/Oct – 2016 | Apr – 2015 | April – 2014

CBSGS: Nov – 2016 | Nov – 2015 | Oct – 2013

SOLUTIONGridView Control:

- ⇒ GridView control is a successor to the ASP.NET 1.X DataGrid Control.
- ⇒ It provides more flexibility in displaying and working with data from your database in comparison with any other controls.
- ⇒ The GridView control enables you to connect to a datasource and display data in tabular format, however you have bunch of options to customize the look and feel.
- ⇒ When it is rendered on the page, generally it is implemented through <table> HTML tag.
- ⇒ Its properties like BackColor, ForeColor, BorderColor, BorderStyle, BorderWidth, Height etc. are implemented through style properties of <table> tag.

Properties of GridView Control:

- **AllowPaging:** true/false. Indicate whether the control should support paging.
- **AllowSorting:** true/false. Indicate whether the control should support sorting.
- **SortExpression:** Gets the current sort expression (field name) that determines the order of the row.
- **SortDirection:** Gets the sorting direction of the column sorted currently (Ascending/Descending).
- **DataSource:** Gets or sets the data source object that contains the data to populate the control.



To enable paging use the properties:

- **AllowPaging**: It is Boolean value indicating whether the GridView will provide paging with page size.
- **PageSize**: It is the size of the page to be included on each page of data.

To enable sorting use the properties:

- **AllowSorting**: It is Boolean value indicating whether the GridView data is sorted. Set value to true.

To enable Row Selection use Enable Selection property to true:

Example:

```
<asp:gridview AllowSorting="true" AllowPaging="true" PageSize="5" ID="Gridview1" runat="server"
DataKeyNames="pid" DataSourceID="SqlDS" >
<Columns>
<asp:BoundField DataField="pname" HeaderText="PRODUCT NAME"
SortExpression="pname"> </asp:BoundField>
</Columns>
</asp:gridview>
```

DATA SOURCE CONTROL

Q.13. What is a Data Source? Explain various types of Data Sources in ASP.NET.

ASKED YEAR →

CBSGS: Nov – 2015

SOLUTION

Data Source Control:

- ⇒ A data source control interacts with the data-bound controls and hides the complex data binding processes.
- ⇒ These are the tools that provide data to the data bound controls and support execution of operations like insertions, deletions, sorting, and updates.
- ⇒ Each data source control wraps a particular data provider-relational databases, XML documents, or custom classes and helps in:
 - *Managing connection*
 - *Selecting data*
 - *Managing presentation aspects like paging, caching, etc.*
 - *Manipulating data*
- ⇒ There are many data source controls available in ASP.NET for accessing data from SQL Server, from ODBC or OLE DB servers, from XML files, and from business objects.

Types Of Data Source Controls:

Based on type of data, these controls could be divided into two categories –

- 1) *Hierarchical Data Source Controls*
- 2) *Table-Based Data Source Controls*

Hierarchical Data Source Controls:

The data source controls used for hierarchical data are –

- **XMLDataSource**: It allows binding to XML files and strings with or without schema information.
- **SiteMapDataSource**: It allows binding to a provider that supplies site map information.

Table-Based Data Source Controls:

The data source controls used for tabular data are –

- **SqlDataSource**: It represents a connection to an ADO.NET data provider that returns SQL data, including data sources accessible via OLEDB and ODBC.
- **ObjectDataSource**: It allows binding to a custom .Net business object that returns data.



- **LinqDataSource**: It allows binding to the results of a Linq-to-SQL query (supported by ASP.NET 3.5 only).
- **AccessDataSource**: It represents connection to a Microsoft Access database.

DETAILSVIEW CONTROL

Q.14. Explain DetailsView Control.

ASKED YEAR →

CBSGS: Nov – 2014

SOLUTION

DetailsView:

- ⇒ The DetailsView Control displays data for a single row of a data source. It is usually used in combination with a drop-down list or GridView control that is used to select the item to be displayed.
- ⇒ The DetailsView element includes a Fields element that contains a BoundField element for each field retrieved from the data source.
- ⇒ You can edit the Fields collection by choosing Edit Fields from the smart tag menu of a DetailsView Control.

Three Modes of the DetailsView Control:

- **ReadOnly**: Used to display an item from the data source.
- **Edit**: Used to edit an item in the data source.
- **Insert**: Used to insert a new item into the data source.

DetailsView Control Attributes:

- **ID**: The ID of this control.
- **Runat**: Must specify "server".
- **DataSourceID**: The ID of the Data source to bind the DetailsView control to.
- **DataKeyNames**: A list of field names that form the primary key for the data source.
- **AutoGenerateRows**: If True, a row is automatically generated for each field in the data source. If False, you must define the rows in the Fields element.
- **DefaultMode**: Sets the initial mode of the DetailsView control. Valid options are Edit, Insert, or ReadOnly.
- **AllowPaging**: Set to True to allow paging.

FORMVIEW CONTROL

Q.15.

➤ Explain FormView Control. [CBSGS: Oct – 2013]

➤ Briefly explain FormView Control. How is it different from DetailsView? [CBSGS: Nov – 2015]

ASKED YEAR →

CBSGS: Nov – 2015 | Oct – 2013

SOLUTION

FormView Control:

- ⇒ The FormView Control uses templates to render all of the fields as a single row by default.
- ⇒ It displays one record at a time.
- ⇒ **Syntax**: `<asp:FormView id="formView1" runat="server">`
- ⇒ FormView comes with the template fields.
- ⇒ The FormView control can use only templates with data binding expressions to display bound data.
- ⇒ It displays the data in a tabular format.

Properties of the FormView Control:

- **EditItemTemplate**: The template that is used when a record is being edited.
- **InsertItemTemplate**: The template that is used when a record is being created.
- **ItemTemplate**: The template that is used to render the record to display only.

Methods of the FormView Control:

- **ChangeMode:** ReadOnly/Insert/Edit. It Change the working mode of the control from the current to the defined FormViewMode type.
- **InsertItem:** It used to insert the record into database. This method must be called when the DetailsView control is in insert mode.
- **UpdateItem:** It used to update the current record into database. This method must be called when DetailsView control is in edit mode.
- **DeleteItem:** It used to delete the current record from database.

FormView Control Vs. DetailView Control:

<u>FormView Control</u>	<u>DetailView Control</u>
FormView Provides more formatting and layout options.	DetailView Provide less formatting and layout options.
FormView can use only templates to display bound data.	DetailView use <BoundField> to display bound data.
The FormView renders all fields in a single table row.	The DetailView displays each field as a table row.
FormView control provides more control over the layout.	DetailView control provides less control over the layout.

LISTVIEW VS. GRIDVIEW**Q.16. What is the difference between ListView and GridView Control?**

ASKED YEAR ⇒

CBSGS: Nov – 2014

SOLUTIONListView Vs. GridView:

<u>ListView</u>	<u>GridView</u>
It was introduced with ASP.NET 3.5.	It was introduced with ASP.NET 2.0.
Template driven.	Rendered as Table.
Built-in supports for Data grouping.	Need to write custom code.
Built-in supports for Insert Operation.	Need to write custom code.
Provides Flexible Layout to your Data.	Need to write custom code.
Performance is fast is compared to GridView.	Performance is slow as compared to ListView.

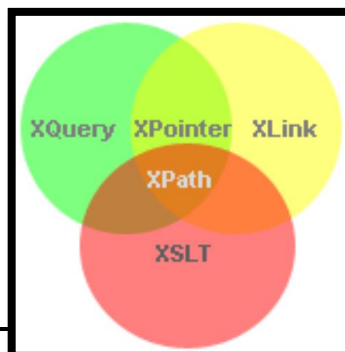
XPATH**Q.17. Write short note on XPath.**

ASKED YEAR ⇒

IDOL: Dec – 2017 | Oct – 2012

SOLUTIONXPath:

- ⇒ XPath stands for XML Path Language.
- ⇒ XPath uses "path like" syntax to identify and navigate nodes in an XML document.
- ⇒ XPath contains over 200 built-in functions.
- ⇒ XPath is a major element in the XSLT standard.
- ⇒ XPath is a W3C recommendation.





XPath Expression:

- ⇒ An XPath expression generally defines a pattern in order to select a set of nodes. These patterns are used by XSLT to perform transformations or by XPointer for addressing purpose.
- ⇒ XPath uses a path expression to select node or a list of nodes from an XML document.
- ⇒ XPath specification specifies seven types of nodes which can be the output of execution of the XPath expression.
 - 1) Root
 - 2) Element
 - 3) Text
 - 4) Attribute
 - 5) Comment
 - 6) Processing Instruction
 - 7) Namespace
- ⇒ Following is the list of useful paths and expression to select any node/ list of nodes from an XML document:
 - **node-name** – Select all nodes with the given name "nodename".
 - **/** – Selection starts from the root node.
 - **//** – Selection starts from the current node that match the selection.
 - **.** – Selects the current node.
 - **..** – Selects the parent of the current node.
 - **@** – Selects attributes.
 - **student** – Selects all nodes with the name "student".
 - **class/student** – Selects all student elements that are children of class.
 - **//student** – Selects all student elements no matter where they are in the document.

DEPLOYMENT

Q.18. State the ways of deployment of website in ASP.NET.

ASKED YEAR → IDOL: May – 2018

SOLUTION

Deployment:

There are two categories of ASP.NET deployment:

- 1) **Local Deployment:** In this case, the entire application is contained within a virtual directory and all the contents and assemblies are contained within it and available to the application.
- 2) **Global Deployment:** In this case, assemblies are available to every application running on the server.

Different Techniques used for Deployment:

There are different techniques used for deployment, following are most common and easiest ways of deployment:

- XCOPY Deployment
- Copying a Website
- Creating a Set Up Project

XCOPY Deployment:

- ⇒ XCOPY deployment means making recursive copies of all the files to the target folder on the target machine.
- ⇒ We can use any of the commonly used techniques:
 - FTP transfer
 - Using Server management tools that provide replication on a remote site
 - MSI installer application
- ⇒ XCOPY deployment simply copies the application file to the production server and sets a virtual directory there.
- ⇒ We need to set a virtual directory using the Internet Information Manager Microsoft Management Console (MMC snap-in).

Copying a Website:

- ⇒ The Copy Web Site option is available in Visual Studio.



- ⇒ It is available from the Website -> Copy Web Site menu option.
- ⇒ This menu item allows copying the current web site to another local or remote location.
- ⇒ It is a sort of integrated FTP tool.
- ⇒ Using this option, we connect to the target destination, select the desired copy mode:
 - Overwrite
 - Source to Target Files
 - Sync UP Source And Target Projects
- ⇒ Then proceed with copying the files physically.
- ⇒ Unlike the XCOPY deployment, this process of deployment is done from Visual Studio environment.
- ⇒ However, there are following problems with both the above deployment methods:
 - You pass on your source code.
 - There is no pre-compilation and related error checking for the files.
 - The initial page load will be slow.

Creating a Setup Project:

- ⇒ In this method, we use Windows Installer and package to our web applications, so it is ready to deploy on the production server.
- ⇒ Visual Studio allows us to build Deployment Packages.
- ⇒ Let us test this on one of our existing project, say the data binding project.

LANGUAGE INTEGRATED QUERY (LINQ)

Q.19.

- **What is LINQ?** [IDOL: May – 2016 | Oct – 2012 | Apr – 2015 | Apr – 2014 | Apr – 2013 | Oct – 2012] | [CBSGS: Nov – 2017 | Nov – 2016 | Apr – 2014 | Oct – 2013]
- **Write the basic syntax of a LINQ Query.** [CBSGS: Nov – 2017]
- **Explain its syntax and give its advantages.** [IDOL: Oct – 2012]
- **Explain the LINQ Query Syntax with an example Query.** [CBSGS: Oct – 2013]
- **Explain the Structure of a LINQ Query.** [IDOL: Apr – 2015 | Apr – 2013]
- **What are advantages and disadvantages of LINQ?** [IDOL: May – 2016 | Apr – 2014 | Oct – 2012] | [CBSGS: Nov – 2016 | Apr – 2014]

ASKED YEAR ⇒

IDOL: May – 2016 | Apr – 2015 | Apr – 2014 | Apr – 2013 | Oct – 2012

CBSGS: Nov/Apr – 2017 | Apr/Nov – 2016 | Apr – 2014 | Oct – 2013

SOLUTION

Language Integrated Query (LINQ):

- ⇒ LINQ stands for Language Integrated Query.
- ⇒ It is a technique for querying data that is integrated into .NET languages such as C# and VB.
- ⇒ It has a single syntax for querying multiple data sources such as relational data and XML data.
- ⇒ It is extensible; talented developers can write providers that allow LINQ to Query any arbitrary data source.
- ⇒ It uses a declarative syntax that allows developers to tell the compiler or provider what to do, not how to do it.
- ⇒ It is composable, in that the results of one query can be used by a second query, and one query can be a subclass of another query. In many cases, this can be done without forcing the execution of any query until the developer wants that execution to take place.
- ⇒ LINQ includes three basic type of implementation:
 - **LINQ to Objects:** It is used to query almost any kind of collection that exists in the .NET Framework.
 - **LINQ to XML:** It is used to query XML data directly in your web application.
 - **LINQ to ADP.NET:** It is used to access data from SQL server and many other different end of data sources.

LINQ Query Syntax:

- ⇒ LINQ Query Syntax is a set of query keywords built into the .NET framework that allow the developer to write SQL style commands in line straight in the code editor, without the use of quotes.



⇒ The .NET Framework introduces us to the following query keywords:

- **from / in** – Specifies the data source.
- **where** – Conditional boolean expression (e.g. `i == 0`).
- **orderby (ascending/descending)** – Sorts the results into ascending or descending order.
- **select** – Adds the result to the return type.
- **group / by** – Groups the results based on a given key.

```
from [identifier] in [source collection]
let [expression]
where [boolean expression]
order by [[expression](ascending/descending)], [optionally repeat]
select [expression]
group [expression] by [expression] into [expression]
```

Example: For the following Generic list called characters,

```
private readonly List<Character> Characters = new List<Character>
{
    //Format: Character Name, Gender, Number of episodes
    new Character("Charlie", Gender.Male, 162),
    new Character("Alan", Gender.Male, 162),
    new Character("Berta", Gender.Female, 117),
    new Character("Jake", Gender.Male, 162),
    new Character("Evelyn", Gender.Female, 78),
    new Character("Judith", Gender.Female, 69) };
```

For Example:

```
IEnumerable<Character> results = from character in Characters
where character.Episodes > 120
select character;
```

- Specify the data source (Characters) and a temporary variable for each element within the data source (in this case a Generic List).
- Create a condition with a Boolean result used to query the data source.
- If the condition is met, select the temporary variable and add it to our results collection (IEnumerable<Type>)

Advantages:

- ⇒ LINQ offers syntax highlighting that proves helpful to find out mistakes during design time.
- ⇒ LINQ offers IntelliSense which means writing more accurate queries easily.
- ⇒ Writing codes is quite faster in LINQ and thus development time also gets reduced significantly.
- ⇒ LINQ makes easy debugging due to its integration in the C# language.
- ⇒ Viewing relationship between two tables is easy with LINQ due to its hierarchical feature and this enables composing queries joining multiple tables in less time.
- ⇒ LINQ allows usage of a single LINQ syntax while querying many diverse data sources and this is mainly because of its unitive foundation.
- ⇒ LINQ is extensible that means it is possible to use knowledge of LINQ to querying new data source types.
- ⇒ LINQ offers the facility of joining several data sources in a single query as well as breaking complex problems into a set of short queries easy to debug.
- ⇒ LINQ offers easy transformation for conversion of one data type to another like transforming SQL data to XML data.

Disadvantages:

- ⇒ Performance is degraded if LINQ queries not written in correct manner.
- ⇒ If there has been a change in LINQ query, then assembly needs to be recompiled and redeployed.
- ⇒ LINQ is generic, whereas queries written in database can take full advantage of the complete database features.
- ⇒ It's much easier to restrict access to tables in database using queries than through LINQ.
- ⇒ LINQ statements are not precompiled whereas SQL stored procedures are precompiled, so stored procedures are faster in performance as compared to traditional LINQ.



- ⇒ LINQ needs to process the complete query, which might have a performance impact in case of complex queries against stored procedures which only need serialize procedure name and argument data over the network.
- ⇒ Sometimes it is hard to understand advance LINQ queries statements.

TYPES OF LINQ OPERATIONS

Q.20. What are different types of operations in LINQ?

ASKED YEAR ⇒

IDOL: May – 2018 | May – 2017 | Oct – 2016 | Apr – 2015

CBSGS: Nov – 2016

SOLUTION

Types Of LINQ Operations:

- ⇒ A set of extension methods forming a query pattern is known as LINQ Standard Query Operators.
- ⇒ As building blocks of LINQ query expressions, these operators offer a range of query capabilities like filtering, sorting, projection, aggregation, etc.
- ⇒ LINQ standard query operators can be categorized into the following ones on the basis of their functionality.
 - 1) Filtering Operators
 - 2) Join Operators
 - 3) Projection Operations
 - 4) Sorting Operators
 - 5) Grouping Operators
 - 6) Conversions
 - 7) Concatenation
 - 8) Aggregation
 - 9) Quantifier Operations
 - 10) Partition Operations
 - 11) Generation Operations
 - 12) Set Operations
 - 13) Equality
 - 14) Element Operators

Filtering Operators:

Filtering is an operation to restrict the result set such that it has only selected elements satisfying a particular condition.

Operator:

- **where**: Filter values based on a predicate function.
- **OfType**: Filter values based on their ability to be as a specified type.

Join Operators:

Joining refers to an operation in which data sources with difficult to follow relationships with each other in a direct way are targeted.

Operator:

- **Join**: The operator join two sequences on basis of matching keys.
- **GroupJoin**: Join two sequences and group the matching elements.

Projection Operations:

Projection is an operation in which an object is transformed into an altogether new form with only specific properties.

Operator:

- **Select**: The operator projects values on basis of a transform function.
- **SelectMany**: The operator project the sequences of values which are based on a transform function as well as flattens them into a single sequence.



Sorting Operators:

A sorting operation allows ordering the elements of a sequence on basis of a single or more attributes.

Operator:

- **OrderBy**: The operator sort values in an ascending order.
- **OrderByDescending**: The operator sort values in a descending order.
- **ThenBy**: Executes a secondary sorting in an ascending order.
- **ThenByDescending**: Executes a secondary sorting in a descending order.
- **Reverse**: Performs a reversal of the order of the elements in a collection.

Grouping Operators:

The operators put data into some groups based on a common shared attribute.

Operator:

- **GroupBy**: Organize a sequence of items in groups and return them as an IEnumerable collection of type IGrouping<key, element>.
- **ToLookup**: Execute a grouping operation in which a sequence of key pairs are returned.

Conversions:

The operators change the type of input objects and are used in a diverse range of applications.

Operator:

- **AsEnumerable**: Returns the input typed as IEnumerable<T>.
- **AsQueryable**: A (generic) IEnumerable is converted to a (generic) IQueryable.
- **Cast**: Performs casting of elements of a collection to a specified type.
- **OfType**: Filters values on basis of their, depending on their capability to be cast to a particular type.
- **ToArray**: Forces query execution and does conversion of a collection to an array.
- **ToDictionary**: On basis of a key selector function set elements into a Dictionary<TKey, TValue> and forces execution of a LINQ query.
- **ToList**: Forces execution of a query by converting a collection to a List<T>.
- **ToLookup**: Forces execution of a query and put elements into a Lookup<TKey, TElement> on basis of a key selector function.

Concatenation:

Performs concatenation of two sequences and is quite similar to the Union operator in terms of its operation except of the fact that this does not remove duplicates.

Operator:

- **Concat**: Two sequences are concatenated for the formation of a single one sequence.

Aggregation:

Performs any type of desired aggregation and allows creating custom aggregations in LINQ.

Operator:

- **Aggregate**: Operates on the values of a collection to perform custom aggregation operation.
- **Average**: Average value of a collection of values is calculated.
- **Count**: Counts the elements satisfying a predicate function within collection.
- **LongCount**: Counts the elements satisfying a predicate function within a huge collection.
- **Max**: Find out the maximum value within a collection.
- **Min**: Find out the minimum value existing within a collection.
- **Sum**: Find out the sum of a values within a collection.



Quantifier Operations:

These operators return a Boolean value i.e. True or False when some or all elements within a sequence satisfy a specific condition.

Operator:

- **All**: Returns a value 'True' if all elements of a sequence satisfy a predicate condition.
- **Any**: Determines by searching a sequence that whether any element of the same satisfy a specified condition.
- **Contains**: Returns a 'True' value if finds that a specific element is there in a sequence if the sequence do not contains that specific element, 'false' value is returned.

Partition Operators:

Divide an input sequence into two separate sections without rearranging the elements of the sequence and then returning one of them.

Operator:

- **Skip**: Skips some specified number of elements within a sequence and returns the remaining ones.
- **SkipWhile**: Same as that of Skip with the only exception that number of elements to skip are specified by a Boolean condition.
- **Take**: Take a specified number of elements from a sequence and skip the remaining ones.
- **TakeWhile**: Same as that of Take accept the fact that number of elements to take are specified by a Boolean condition.

Generation Operations:

A new sequence of values is created by generational operators.

Operator:

- **DefaultIfEmpty**: When applied to an empty sequence, generate a default element within a sequence.
- **Empty**: Returns an empty sequence of values and is the simplest generational operator.
- **Range**: Generates a collection having a sequence of integers or numbers Not Applicable.
- **Repeat**: Generates a sequence containing repeated values of a specific length.

Set Operations:

There are four operators for the set operations, each yielding a result based on different criteria.

Operator:

- **Distinct**: Results a list of unique values from a collection by filtering duplicate data if any.
- **Except**: Compares the values of two collections and return the ones from one collection who are not in the other collection.
- **Intersect**: Returns the set of values found to be identical in two separate collections.
- **Union**: Combines content of two different collections into a single list that too without any duplicate content.

Equality:

Compares two sentences (enumerable) and determine are they an exact match or not.

Operator:

- **SequenceEqual**: Results a Boolean value if two sequences are found to be identical to each other.

Element Operators:

Except the DefaultIfEmpty, all the rest eight standard query element operators return a single element from a collection.

Operator:

- **ElementAt**: Returns an element present within a specific index in a collection.
- **ElementAtOrDefault**: Same as ElementAt except of the fact that it also returns a default value in case the specific index is out of range.
- **First**: Retrieves the first element within a collection or the first element satisfying a specific condition.



- **FirstOrDefault**: Same as First except the fact that it also returns a default value in case there is no existence of such elements.
- **Last**: Retrieves the last element present in a collection or the last element satisfying a specific condition.
- **LastOrDefault**: Same as Last except the fact that it also returns a default value in case there is no existence of any such element.
- **Single**: Returns the lone element of a collection or the lone element that satisfy a certain condition.
- **SingleOrDefault**: Same as Single except that it also returns a default value if there is no existence of any such lone element.
- **DefaultIfEmpty**: Returns a default value if the collection or list is empty or null.

LINQ QUERY OPERATORS

Q.21.

- Explain the terms Take, Skip, TakeWhile, SkipWhile, First, FirstOrDefault, Last, LastOrDefault with respect to LINQ. [CBSGS: Oct – 2013]
- Write a note on first, FirstOrDefault, Last and LastOrDefault. [IDOL: Apr – 2013]

ASKED YEAR →

IDOL: Apr – 2013

CBSGS: Oct – 2013

SOLUTION

LINQ Query Operators:

- **Skip**: Skips elements up to a specified position in a sequence.
- **SkipWhile**: Skips elements based on a predicate function until an element does not satisfy the condition.
- **Take**: Takes elements up to a specified position in a sequence.
- **TakeWhile**: Takes elements based on a predicate function until an element does not satisfy the condition.
- **First**: Returns the first element of a collection, or the first element that satisfies a condition.
- **FirstOrDefault**: Returns the first element of a collection, or the first element that satisfies a condition. Returns a default value if no such element exists.
- **Last**: Returns the last element of a collection, or the last element that satisfies a condition.
- **LastOrDefault**: Returns the last element of a collection, or the last element that satisfies a condition. Returns a default value if no such element exists.

LINQ STANDARD QUERY OPERATORS

Q.22.

- Explain the Standard Query Operators "Select", "From", "OrderBy" and "Where" in LINQ. [IDOL: May – 2018 | Apr – 2013] | [CBSGS: Nov – 2015 | Nov – 2014]
- Explain any four Standard Query Operators in LINQ. [IDOL: May – 2018]

ASKED YEAR →

IDOL: May – 2018 | Apr – 2013

CBSGS: Nov – 2015 | Nov – 2014

SOLUTION

LINQ Standard Query Operators:

Select:

- ⇒ There are two projection operators available in LINQ (i.e. Select and SelectMany).
- ⇒ The Select operator always returns an IEnumerable collection which contains elements based on a transformation function.
- ⇒ It is similar to the Select clause of SQL that produces a flat result set.

Example: The following example of the select clause returns a collection of anonymous type containing the Name and Age property.

```

IList<Student> studentList = new List<Student>()
{
    new Student() { StudentID = 1, StudentName = "John", Age = 13 },
    new Student() { StudentID = 2, StudentName = "Moin", Age = 21 },
    new Student() { StudentID = 3, StudentName = "Bill", Age = 18 },
    new Student() { StudentID = 4, StudentName = "Ram", Age = 20 },
    new Student() { StudentID = 5, StudentName = "Ron", Age = 15 }
}

```



```
};  
// returns collection of anonymous objects with Name and Age property  
var selectResult = from s in studentList  
select new { Name = "Mr. " + s.StudentName, Age = s.Age };  
// iterate selectResult  
foreach (var item in selectResult)  
Console.WriteLine("Student Name: {0}, Age: {1}", item.Name, item.Age);
```

Output:

```
Student Name: Mr. John, Age: 13  
Student Name: Mr. Moin, Age: 21  
Student Name: Mr. Bill, Age: 18  
Student Name: Mr. Ram, Age: 20  
Student Name: Mr. Ron, Age: 15
```

From:

- ⇒ It defines the collection or data source that the query must act upon.
- ⇒ The From operator outputs the result in a variable mentioned after the "From" clause.

OrderBy:

- ⇒ It sorts values in ascending order.
- ⇒ It can sort the items in the result collection.
- ⇒ It is followed by an optional ascending and descending keyword to specify sort order.
- ⇒ We can specify multiple criteria by separating them with a comma.

Example:

```
IList<Student> studentList = new List<Student>()  
{  
    new Student() { StudentID = 1, StudentName = "John", Age = 18 },  
    new Student() { StudentID = 2, StudentName = "Steve", Age = 15 },  
    new Student() { StudentID = 3, StudentName = "Bill", Age = 25 },  
    new Student() { StudentID = 4, StudentName = "Ram", Age = 20 },  
    new Student() { StudentID = 5, StudentName = "Ron", Age = 19 }  
};  
var orderByResult = from s in studentList  
orderby s.StudentName  
select s;  
  
var orderByDescendingResult = from s in studentList  
orderby s.StudentName descending  
select s;
```

Output:

orderByResult in the above example would contain following elements after execution:

```
Bill  
John  
Ram  
Ron  
Steve
```

orderByDescendingResult in the above example would contain following elements after execution:

```
Steve  
Ron  
Ram  
John  
Bill
```

Where:

- ⇒ It filters the collection based on a given criteria expression and returns a new collection.
- ⇒ The criteria can be specified as lambda expression or Func delegate type.
- ⇒ The Where extension method has following two overloads:

```
public static IEnumerable<TSource> Where<TSource>(this IEnumerable<TSource> source,
Func<TSource, bool> predicate);
```

```
public static IEnumerable<TSource> Where<TSource>(this IEnumerable<TSource> source,
Func<TSource, int, bool> predicate);
```

- ⇒ Both overload methods accepts a Func delegate type parameter.
- ⇒ One overload required Func<TSource, bool> input parameter and second overload method required Func<TSource, int, bool> input parameter where int is for index:

Example: The following query sample uses a Where operator to filter the students who is teen ager from the given collection (sequence). It uses a lambda expression as a predicate function.

```
IList<Student> studentList = new List<Student>()
{
    new Student() { StudentID = 1, StudentName = "John", Age = 13 } ,
    new Student() { StudentID = 2, StudentName = "Moin", Age = 21 } ,
    new Student() { StudentID = 3, StudentName = "Bill", Age = 18 } ,
    new Student() { StudentID = 4, StudentName = "Ram" , Age = 20 } ,
    new Student() { StudentID = 5, StudentName = "Ron" , Age = 15 }
};
var filteredResult = from s in studentList
where s.Age > 12 && s.Age < 20
select s.StudentName;
```

Output:

```
John
Bill
Ron
```

IMPLEMENTATIONS**Q.23. What are the different ways to implement LINQ?**

ASKED YEAR ⇒ IDOL: May – 2017

SOLUTION**Different ways to implement LINQ:**

- ⇒ LINQ provides a uniform programming model (i.e. common query syntax) to query data sources (like SQL databases, XML documents, ADO.NET Datasets, Various Web services and any other objects such as Collections, Generics etc).
- ⇒ LINQ provides you three different ways to write a LINQ query in C# or VB.

Query Expression (Query Syntax)

- ⇒ Query expression syntax is like as SQL query syntax with just a few minor deviations.
- ⇒ The result of a query expression is a query object, which is usually a collection of type IEnumerable<T> or IQueryable<T>.
- ⇒ This syntax is easy to read and write and at compile time, query expression is converted into Method Invocation.

Query Expression Syntax:

```
from [identifier]
in [source collection]
let [expression]
where [boolean expression]
order by [expression(ascending/descending)]
```



```
select [expression]
group [expression] by [expression]
into [expression]
```

Query Expression Example:

```
// Datasource
List<int> numbers = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
// Query based syntax
IEnumerable query =
from num in numbers
where num > 5 && num < 10
select num;
//result: 6,7,8,9
```

Method Invocation (Method Syntax):

- ⇒ Method syntax is complex as compared to Query expression since it uses lambda expression to write LINQ query.
- ⇒ It is easily understood by .NET CLR.
- ⇒ Hence at compile time, Query expression is converted into Method Invocation.
- ⇒ The result of a Method syntax is also a query object, which is usually a collection of type IEnumerable<T> or IQueryable<T>.

```
[source collection]
.Where [boolean expression]
.OrderBy [expression(ascending/descending)]
.Select [expression]
.GroupBy [expression]
```

Method Syntax Example:

```
// Datasource
List<int> numbers = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
// Method based syntax
IEnumerable<int> query =numbers.Where(num => num > 5 && num < 10)
//result: 6,7,8,9
```

Mixed Syntax:

- ⇒ We can use a mixture of both syntax by enclosing a query expression inside parentheses and make a call to method.

Mixed Syntax Example:

```
// Datasource
List<int> numbers = new List<int>() { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
// Mixed syntax
int query = (from num in numbers
where num > 5 && num < 10
select num).Count();
//result: 4
```

**Q.24. How can Query Syntax and Extension Methods be mixed? Explain.**ASKED YEAR ⇒ **IDOL:** Oct – 2012**SOLUTION****Query Syntax and Extension Methods Can Be Mixed:**

- ⇒ The System.Linq namespace also gives us the ability to use LINQ's built in extension methods on the results of our query expressions.
- ⇒ For example, instead of storing the results of the query in a variable and then calling the Count extension method and storing the result in another variable, we can just do it all right on a single line;

```
int count = (from character in Characters
where character.Episodes > 120
select character).Count();
```

- ⇒ All we need to do to achieve this is wrap our expression in parenthesis.

LINQ To OBJECTS**Q.25. What is LINQ to Objects? Explain.**ASKED YEAR ⇒ **IDOL:** Dec – 2017**CBSGS:** Apr – 2014 | Nov – 2015**SOLUTION****LINQ To Objects:**

- ⇒ Queries in LINQ to Objects return variables of type usually IEnumerable<T> only.
- ⇒ In short, LINQ to Objects offers a fresh approach to collections as earlier, it was vital to write long coding (foreach loops of much complexity) for retrieval of data from a collection which is now replaced by writing declarative code which clearly describes the desired data that is required to retrieve.
- ⇒ There are also many advantages of LINQ to Objects over traditional foreach loops like more readability, powerful filtering, capability of grouping, enhanced ordering with minimal application coding.
- ⇒ Such LINQ queries are also more compact in nature and are portable to any other data sources without any modification or with just a little modification.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace LINQtoObjects
{
class Program
{
static void Main(string[] args)
{
string[] tools = { "Tablesaw", "Bandsaw", "Planer", "Jointer", "Drill", "Sander" };
var list = from t in tools select t;
StringBuilder sb = new StringBuilder();
foreach (string s in list)
{
sb.Append(s + Environment.NewLine);
}
Console.WriteLine(sb.ToString(), "Tools");
Console.ReadLine();
}
}
}
```

In the example, an array of strings (tools) is used as the collection of objects to be queried using LINQ to Objects.



```
Objects query is:  
var list = from t in tools select t;
```

Output:

```
Tablesaw  
Bandsaw  
Planer  
Jointer  
Drill  
Sander
```

LINQ To XML

Q.26. Explain with an example LINQ to XML.ASKED YEAR → **IDOL: Oct – 2016**

SOLUTION

LINQ To XML:

- ⇒ LINQ to XML offers easy accessibility to all LINQ functionalities like standard query operators, programming interface, etc.
- ⇒ Integrated in the .NET framework, LINQ to XML also makes the best use of .NET framework functionalities like debugging, compile-time checking, strong typing and many more to say.
- ⇒ While using LINQ to XML, loading XML documents into memory is easy and easier is querying and document modification.
- ⇒ It is also possible to save XML documents existing in memory to disk and to serialize them.
- ⇒ It eliminates the need for a developer to learn the XML query language which is somewhat complex.
- ⇒ LINQ to XML has its power in the `System.Xml.Linq` namespace.
- ⇒ This has all the 19 necessary classes to work with XML.
- ⇒ These classes are the following ones:
 - 1) XAttribute
 - 2) XCDATA
 - 3) XComment
 - 4) XContainer
 - 5) XDeclaration
 - 6) XDocument
 - 7) XDocumentType
 - 8) XElement
 - 9) XName
 - 10) XNamespace
 - 11) XNode
 - 12) XNodeDocumentOrderComparer
 - 13) XNodeEqualityComparer
 - 14) XObject
 - 15) XObjectChange
 - 16) XObjectChangeEventArgs
 - 17) XObjectEventHandler
 - 18) XProcessingInstruction
 - 19) XText

Example: Read an XML File using LINQ

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Xml.Linq;  
namespace LINQtoXML  
{  
class ExampleOfXML  
{
```



```
static void Main(string[] args)
{
    string myXML = @"<Departments>
<Department>Account</Department>
<Department>Sales</Department>
<Department>Pre-Sales</Department>
<Department>Marketing</Department>
</Departments>";
    XDocument xdoc = new XDocument();
    xdoc = XDocument.Parse(myXML);
    var result = xdoc.Element("Departments").Descendants();
    foreach (XElement item in result)
    {
        Console.WriteLine("Department Name - " + item.Value);
    }
    Console.WriteLine("\nPress any key to continue.");
    Console.ReadKey();
}
}
```

Output:

```
Department Name - Account
Department Name - Sales
Department Name - Pre-Sales
Department Name - Marketing

Press any key to continue.
```

LINQ To XML

Q.27.**Explain LINQ to SQL with the help of a Query that performs equijoin.**

ASKED YEAR →

CBSGS: Nov – 2015

SOLUTION

LINQ To SQL:

- ⇒ It is used to access data from SQL Server and many other different kinds of data sources. Using LINQ to ADO.NET we can Reform LINQ to dataset, LINQ to SQL, and LINQ to entities.
- ⇒ It allows writing oriented queries against SQL server databases.
- ⇒ It allows writing queries against the dataset.

Steps to create LINQ to ADO.NET:**STEP 1:** Add GridView control to the webpage.**STEP 2:** Add LINQ to SQL classes file in our website using Website Menu. Add new item. Name it DataClasses dbml**STEP 3:** From server explorer connect to db. Drag and drop any table from the db to the dbml file.**STEP 4:** Add following code.

```
protected void page_load()
{
    DataclassesDataContext dc=new DataClasses DataContext(),
    varquery=from m in dc.custs select m;
    Gridview1.datasource=query;
    Gridview1.dataBind();
}
```



LINQ Vs. SQL

Q.28. What is difference between SQL and LINQ?

ASKED YEAR →

CBSGS: Apr – 2014

SOLUTION

LINQ Vs. SQL:

- ⇒ LINQ is Simpler, Tidier and Higher-Level.
- ⇒ LINQ enabling you to access and query a wide variety of sources including collections in your own code, XML files, .NET Datasets and databases from your VB.NET or C# code.
- ⇒ SQL language is used to create, modify and retrieve information from RDBMS. LINQ is a uniform programming model for any kind of Data Access.

AUTHENTICATION

Q.29.

- **What do you mean by Authentication?** [CBSGS: Nov – 2014] | [IDOL: May – 2017 | Oct/May – 2016]
- **Explain types of Authentication.** [CBSGS: Nov – 2014] | [IDOL: May – 2017 | Oct/May – 2016]
- **Explain Passport Authentication in ASP.NET.** [IDOL: May – 2016]
- **What are the Authentication Mode in ASP.NET for security?** [CBSGS: Nov/Apr – 2017]
- **Explain Windows Authentication in ASP.NET.** [CBSGS: Nov/Apr – 2016 | Apr – 2015]
- **Write the steps with sample code to use Windows Authentication.** [CBSGS: Apr – 2014]

ASKED YEAR →

IDOL: May – 2017 | Oct/May – 2016

CBSGS: Nov/Apr – 2017 | Nov/Apr – 2016 | Apr – 2015 | Nov/Apr – 2014

SOLUTION

Authentication:

- ⇒ Authentication is the process of obtaining identification credentials such as name and password from a user and validating those credentials against some authority.
- ⇒ If the credentials are valid, the entity that submitted the credentials is considered an authenticated identity.
- ⇒ To enable a specified authentication provider for an ASP.NET application, you must create an entry in the application's configuration file as follows:

```
//web.config file
<authentication mode="[Windows/Forms/Passport/None]">
</authentication>
```

Types of Authentication:

There are three ways of doing authentication and authorization in ASP.NET:

- 1) *Windows Authentication*
- 2) *Forms Authentication*
- 3) *Passport Authentication*

Windows Authentication:

- ⇒ The Windows authentication provider is the default provider for ASP.NET.
- ⇒ Windows Authentication authenticates the user based on the user's windows accounts.
- ⇒ Windows authentication in ASP.NET actually relies on IIS to do the authentication.
- ⇒ If the user attempts to access a page and is not authenticated then user will be shown a dialog box asking them to enter their username & password.
- ⇒ This information is then passed to the web server & checked against the list of users in the domain.
- ⇒ It causes the browser to display a login dialog box when the user attempts to access restricted page.
- ⇒ To configure ASP.NET to use Windows authentication use the following code:

```
<system.web>
<authentication mode="Windows"/>
```




```
<authorization>
<allow users="*" />
</authorization>
</system.web>
<allow users="*" />
```

⇒ Statement specifies permissions are provided to authorized users as well anonymous users.

Types Of Windows Authentication:

There are four different kinds of Windows Authentication option available that can be configured in IIS:

1) **Basic Authentication:**

- ⇒ This form of authentication is supported by all browsers.
- ⇒ When a website requests client authentication using Basic authentication, the web browser displays a login dialog box from the user name and password.
- ⇒ After a user provides built-in Windows user account information, the data is transmitted to a web server.
- ⇒ Once IIS receives the authentication data, it attempts to authenticate the user with the corresponding Windows account.
- ⇒ This password is encoded using Base64 and sent to server.
- ⇒ It is important to note that the Base64 encoding is not encryption.
- ⇒ So the drawback of this mechanism is that the user name and password are sent in clear text (unencrypted) during communication.
- ⇒ MITM attack can be executed by a malicious hacker and crucial credentials can be capture in between the communication.

2) **Anonymous Authentication:**

- ⇒ A remote user is not required to supply credentials to access a file when Anonymous Authentication is enabled.
- ⇒ By default, the configured anonymous access account is the IUSR account created when IIS is installed.
- ⇒ Anonymous Authentication can be configured from IIS manages.
- ⇒ It is important to note that if we enable more than one authentication option.
- ⇒ The client will use the strongest authentication method as long as anonymous authentication is not enabled.
- ⇒ If anonymous authentication is enabled, the client will access the website anonymously.
- ⇒ So it is suggested that you disable anonymous authentication during more than one authentication implementation.

3) **Digest Authentication:**

- ⇒ Digest Authentication, like Basic Authentication requires the user to provide account information using a login dialog box that is displayed by the browser.
- ⇒ Unlike the basic authentication, the user name and password are not transmitted in clear text.
- ⇒ Instead, a cryptographically secure hash with this information is sent.
- ⇒ We can implement this authentication by simply enabling this option in IIS.
- ⇒ Digest Authentication involves hashing the user's password using the MD5 algorithm.
- ⇒ Windows is unable to store MD5 hashes of password for local accounts (SAM database) thus the limitation of Digest Authentication is that in IIS, it function only when the virtual directory being authenticated or controlled by Windows Active Directory Domain Controller.
- ⇒ Digest Authentication protects users and applications against a variety of malicious attacks by incorporating a piece of information about the request as inputs to the hashing algorithm.
- ⇒ Enabling and disabling digest authentication can also be performed programmatically. We can enable this authentication using AppCmd command as follows:

```
appcmd.exe set config /section:digestAuthentication /enable:true
```

4) **Integrated Windows Authentication:**

- ⇒ Integrated Windows Authentication is the most reasonable mechanism for LAN-WAN-based applications.
- ⇒ For this authentication to work properly, both client and server must be on same network.
- ⇒ In fact, integrated authentication does not transmit any credential information.
- ⇒ Instead, it coordinates with the domain server where it logged in and gets that computer to send the authentication information to the server.
- ⇒ It performs authentication without any client interactions.



- ⇒ When IIS asks the client to authenticate itself, the browser sends a token that represents the Windows account of the current user.
- ⇒ Technically, this authentication incorporates two authentication mechanisms: NTLM and Kerberos.
- ⇒ Enabling integrated authentication via IIS manager typically enables supports for both of these two mechanism.

Steps to use Windows Authentication with Sample Code:

- ⇒ When you configure your ASP.NET application as windows authentication it will use local windows user and groups to do authentication and authorization for your ASP.NET pages.
- ⇒ In 'web.config' file set the authentication mode to 'Windows' as shown in the below code snippets.

```
<authentication mode="Windows"/>
```

- ⇒ We also need to ensure that all users are denied except authorized users. The below code snippet inside the authorization tag that all users are denied. '?' indicates any

```
<authorization>  
<deny users="?"/>  
</authorization>
```

- ⇒ We also need to specify the authorization part. We need to insert the below snippet in the 'web.config' file stating that only 'Administrator' users will have access to

```
<location path="Admin.aspx">  
<system.web>  
<authorization>  
<allow roles="questpon-srize2\Administrator"/>  
<deny users="*" />  
</authorization>  
</system.web>  
</location>
```

- ⇒ The next step is to compile the project and upload the same on an IIS virtual directory. On the IIS virtual directory we need to ensure to remove anonymous access and check the integrated windows authentication.
- ⇒ Now if you run the web application you will be popped with a userid and password box.

Forms Authentication:

- ⇒ This is a cookie Based Authentication where username and password are stored on client machines as cookie files or they are sent through URL for every request.
- ⇒ Forms-based Authentication presents the user with an HTML-based Web page that prompts the user for credentials.

```
<configuration>  
<system.web>  
<authentication mode="[Windows/Forms/Password/None]">  
</authentication>  
</system.web>  
</configuration>
```

Passport Authentication:

- ⇒ This is an authentication strategy that makes use of Microsoft Password service to authenticate the users of an application.
- ⇒ It allows users to create a Single Sign-IN (SSI) name and Password that they can use to access any site that has implemented the Password SSI Service.
- ⇒ It is secure as it is based on cryptography and eliminates credentials-management hassles in the application.

```
<configuration>  
<system.web>  
<authentication mode="Password">  
<passportredirect Url="loginform.aspx"/>  
</authentication>  
<authorization>  
<deny users="2">  
</authorization>  
</system.web>  
</configuration>
```

AUTHORIZATION**Q.30.**

- **What is the Authorization in ASP.NET?** [IDOL: May – 2018 | May – 2017 | Oct – 2016 | Apr – 2015] | [CBSGS: Nov – 2017]
 ➤ **How Authorization is used in case of providing Security to Web Application.** [IDOL: Apr – 2015]

ASKED YEAR ⇒

IDOL: May – 2018 | May – 2017 | Oct – 2016 | Apr – 2015

CBSGS: Nov – 2017

SOLUTIONAuthorization:

- ⇒ Authorization determines whether an identity should be granted access to a specific resource.
- ⇒ In ASP.NET, there are two ways to authorize access to a given resource:
 - File Authorization
 - URL Authorization

File Authorization:

- ⇒ File authorization is performed by the `FileAuthorizationModule`.
- ⇒ It checks the access control list (ACL) of the ".aspx" or ".asmx" handler file to determine whether a user should have access to the file.
- ⇒ ACL permissions are verified for the user's Windows identity (if Windows authentication is enabled) or for the Windows identity of the ASP.NET process.

URL Authorization:

- ⇒ URL authorization is performed by the `UrlAuthorizationModule`, which maps users and roles to URLs in ASP.NET applications.
- ⇒ This module can be used to selectively allow or deny access to arbitrary parts of an application (typically directories) for specific users or roles.

Using URL Authorization:

- ⇒ With URL authorization, we explicitly allow or deny access to a particular directory by user name or role.
- ⇒ To do so, we create an authorization section in the configuration file for that directory.
- ⇒ To enable URL authorization, you specify a list of users or roles in the `allow` or `deny` elements of the authorization section of a configuration file.
- ⇒ The permissions established for a directory also apply to its subdirectories, unless configuration files in a subdirectory override them.

The following shows the syntax for the authorization section:

```
<authorization>
<[allow|deny] users roles verbs />
</authorization>
```

- The `allow` or `deny` element is required.
- We must specify either the `users` or the `roles` attribute.
- Both can be included, but both are not required.
- The `verbs` attribute is optional.

AUTHORISATION VS. IMPERSONATION**Q.31.****What is the difference between Authorisation and Impersonation in terms of Security in ASP.NET?**

ASKED YEAR ⇒

CBSGS: Oct – 2013

SOLUTIONAuthorisation Vs. Impersonation:AuthorisationImpersonation



Authorization determines whether an identity should be granted access to a specific resource.

By default ASP.NET executes in the security context of a restricted user account on the local machine. Sometimes one needs to access network resources such as a file on a shared drive, which requires additional permissions. One way to overcome this restriction is to impersonation. With impersonation, ASP.NET can execute the request using the identity of the client who is making the request or ASP.NET can impersonate a specific account you specify in web.config.

Example:

```
web.config
<configuration>
<system.web>
<authorization>
<allow users="*" />
</authorization>
</system.web>
</configuration>
```

Example:

```
web.config
<configuration>
<system.web>
<identity impersonate="true">
</system.web>
</configuration>
```

IMPERSONATION

Q.32. Write a note on Impersonation.

ASKED YEAR ⇒

IDOL: May – 2018 | May – 2017 | Oct – 2016

CBSGS: Nov – 2016

SOLUTION

Impersonation:

- ⇒ Impersonation is the process of executing code in the framework of another user identity.
- ⇒ By default all ASP.NET code is executed using a fixed machine-specific account.
- ⇒ To execute code using another identity you can use the built-in impersonation capabilities of ASP.NET.
- ⇒ Impersonation is a technique that allows the ASP.NET process act as the authenticated user or as an arbitrary specified user.
- ⇒ ASP.NET impersonation is controlled by <identity> tag in the applications web.config file.
- ⇒ The default setting is impersonation as false. You also can explicitly specify that
- ⇒ ASP.NET should not use impersonation by including the following code in file:

```
<identity impersonate="false" />
```

- ⇒ Now ASP.NET does not perform impersonation. It means that ASP.NET will runs with its own privileges. The second possible setting is to turn on impersonation.

```
<identity impersonate="true" />
```

- ⇒ Now ASP.NET takes on the identity IIS passes to it. 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.
- ⇒ To impersonate a specific user for all the requests on all pages of an ASP.NET application, you can specify the username and password attributes in the <identity> tag of the Web.config file for that application as follows:

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



| UNIT |

VI

AJAX

jQuery

Topic

AJAX (ASYNCHRONOUS JAVASCRIPT AND XML)

- ⇒ AJAX
- ⇒ ADVANTAGES OF AJAX
- ⇒ DISADVANTAGES OF AJAX
- ⇒ WORKING OF AJAX
- ⇒ ARCHITECTURE OF AJAX
- ⇒ APPLICATIONS OF AJAX
- ⇒ AJAX WEB APPLICATION MODEL
- ⇒ AJAX PAGE VS. TRADITIONAL PAGE
- ⇒ STEPS FOR CREATING AJAX APPLICATION
- ⇒ AJAX CONTROL
 - > UPDATEPANEL CONTROL
 - > SCRIPTMANAGER CONTROL
 - > UPDATEPROGRESS CONTROL
 - > TIMER CONTROL
- ⇒ WEB SERVICES
 - > STEPS FOR CREATING WEB SERVICE
 - > ADVANTAGES OF WEB SERVICES
 - > STEPS TO PUBLISH THE WEBSITE
- ⇒ XMLHTTPREQUEST

JQUERY

- ⇒ JQUERY IN ASP.NET
- ⇒ USE OF JQUERY
- ⇒ CONCEPT OF JQUERY
- ⇒ SYNTAX OF QUERY
- ⇒ FEATURES OF JQUERY
- ⇒ DOLLAR SYMBOLS (\$) IN JQUERY
- ⇒ DOCUMENTREADY FUNCTION
- ⇒ JQUERY EVENT FUNCTIONS
- ⇒ IMPORTANCE OF JQUERY
- ⇒ BENEFITS OF USING JQUERY
- ⇒ JQUERY SELECTOR
 - > ELEMENT SELECTOR
 - > ID SELECTOR
 - > CLASS SELECTOR
 - > UNIVERSAL SELECTOR
 - > MULTIPLE ELEMENTS SELECTOR
- ⇒ DOM MANIPULATION METHODS

**AJAX (ASYNCHRONOUS JAVASCRIPT AND XML)****Q.1.**

- **What is AJAX?** [CBSGS: Apr – 2015 | Nov – 2014]
- **How is the processing of a web page without AJAX different from the processing of a Web Page with AJAX?** [CBSGS: Nov – 2014]

ASKED YEAR →

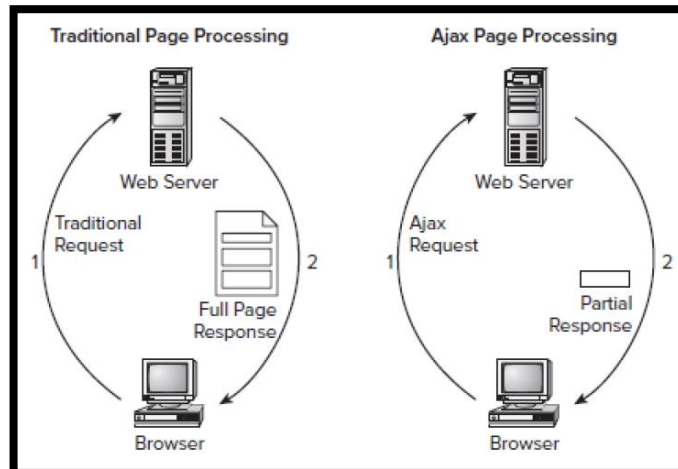
CBSGS: Apr – 2015 | Nov – 2014

SOLUTION**AJAX:**

- ⇒ AJAX stands for Asynchronous JavaScript and XML, enables your client-side web pages to exchange data with the server through asynchronous calls. Probably the most popular feature driven by Ajax is the flicker-free page that enables you to perform a postback to the server without refreshing the entire page.
- ⇒ In traditional page processing the first drawback is entire page is loaded after a postback, the HTML sent to the browser is much larger than it needs to be.
- ⇒ The second drawback of a full page reload has to do with the way the browser renders the page. Because the entire page is replaced, the browser has to dismiss the old one and then draw the new one. This causes the page to "flicker", which results in an unattractive user experience.

ASP.NET AJAX enables you to:

- ⇒ Create flicker-free pages that enable you to refresh portions of the page without a full reload and without affecting other parts of the page
- ⇒ Provide feedback to your users during these page refreshes
- ⇒ Update sections of a page and call server-side code on a scheduled basis using a timer
- ⇒ Access Server-Side Web Services and page methods and work with the data they return
- ⇒ Use the rich, client-side programming framework to access and modify elements in your page, and get access to a code model and type system that looks similar to that of the .NET Framework.



**Q.2.**

- **List the advantages and disadvantages of AJAX.** [IDOL: May – 2016 | Apr – 2015] | [CBSGS: Apr – 2017]
➤ **What are the benefits of using AJAX?** [CBSGS: Nov – 2017 | Nov – 2015] | [IDOL: Apr – 2014]

ASKED YEAR → IDOL: May – 2016 | Apr – 2015 | Apr – 2014

CBSGS: Apr/Nov – 2017 | Nov – 2015

SOLUTION

Advantages Of AJAX:

- ⇒ **AJAX Learn:** Ajax is easy to learn and understand.
- ⇒ **Speed:** Reduce the Server Traffic in both side request to or from. Also reduce the time consuming on both side responses.
- ⇒ **Interaction:** AJAX is much responsive, whole page (small amount) data transfer at a time.
- ⇒ **XMLHttpRequest:** It can be used by Web browser scripting languages, such as JavaScript to transfer XML and other text data to and from a Web server using HTTP. It is used for making requests to the non-Ajax pages.
- ⇒ **Asynchronous Calls:** AJAX allows for the ability to make asynchronous calls to a web server. This allows the client browser to avoid waiting for all data to arrive before allowing the user to act once more.
- ⇒ **Form Validation:** This is the biggest advantage. Form validation should be instant and properly. AJAX gives you all of that, and more. Form validation is the easiest way to get a proper validation.
- ⇒ **Bandwidth Usage:** No require to completely reload page you are really using less server bandwidth, and when you are in countries or places that charge for your surfing bandwidth, it really is something to think about.

Disadvantages Of AJAX:

- ⇒ Clicking the browser's "back" button may not return the user to an earlier state of the Ajax-enabled page.
- ⇒ Dynamic web page updates also caused some troubles for a user to bookmark a particular state of the application.
- ⇒ Ajax opens up another attack vector for malicious code that web developers might not expected for.
- ⇒ Any user whose browser does not support Ajax or JavaScript, or simply has JavaScript disabled, will not be able to use its functionality.
- ⇒ ActiveX requests are enabled only in I.E. or never latest browser.
- ⇒ It is built on JavaScript.
- ⇒ Open Source.

Q.3.**What is the significance of AJAX Websites?**

ASKED YEAR → IDOL: Apr – 2015

SOLUTION

Significance Of AJAX Websites:

- ⇒ AJAX stands for Asynchronous JavaScript and XML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script.
- ⇒ Ajax uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic content display.
- ⇒ Conventional web applications transmit information to and from the sever using synchronous requests. It means you fill out a form, hit submit, and get directed to a new page with new information from the server.
- ⇒ With AJAX, when you hit submit, JavaScript will make a request to the server, interpret the results, and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.
- ⇒ XML is commonly used as the format for receiving server data, although any format, including plain text, can be used.
- ⇒ AJAX is a web browser technology independent of web server software.
- ⇒ A user can continue to use the application while the client program requests information from the server in the background.
- ⇒ Intuitive and natural user interaction. Clicking is not required, mouse movement is a sufficient event trigger.
- ⇒ Data-driven as opposed to page-driven.



WORKING OF AJAX

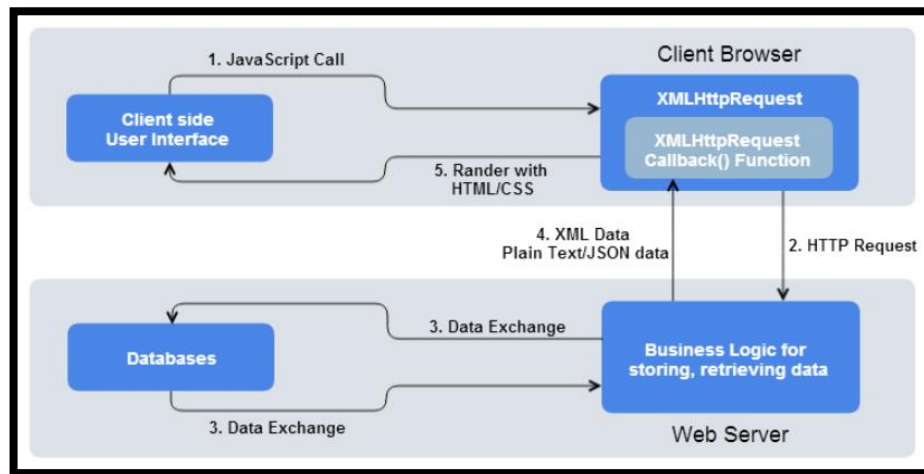
Q.4. Explain the working of AJAX.

ASKED YEAR → IDOL: Dec/May – 2017 | Oct – 2012

SOLUTION

Working Of AJAX:

Below figure (visual diagram) illustrates how AJAX technologies work together to handle user action. User action are triggers an AJAX response.



- ⇒ Client side user perform action to generate event that event call to a JavaScript function.
- ⇒ JavaScript function create XMLHttpRequest object, XMLHttpRequest object specify the JavaScript callback function.
- ⇒ JavaScript XMLHttpRequest object call as an asynchronous HTTP request to the Server.
- ⇒ Web Server process the request and return XML contain data.
- ⇒ XMLHttpRequest object calls to a callback function along with response from the web server.
- ⇒ Client browser updates the HTML DOM representing the web page along with new data.

Example:

```
<asp:ScriptManger runat="server">
</asp:ScriptManger>
<asp:UpdatePanel runat="server">
<ContentTemplate>
<asp:Label runat="server">
<asp:Button runat="server" Text="Button" onClick="Button1_Click"/>
</ContentTemplate>
</asp:UpdatePanel>
```

Code Behind:

```
protected void Page_Load(object sender, EventArgs e)
{
    Labell1.Text=DateTime.New.ToString();
}
```




ARCHITECTURE OF AJAX

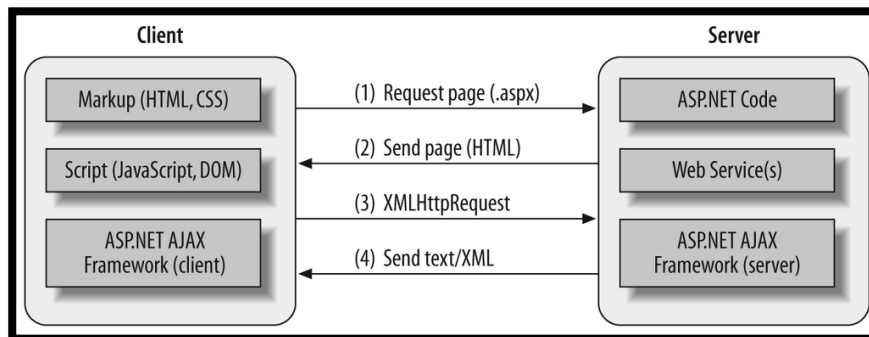
Q.5. Write a short note on Architecture of AJAX.

ASKED YEAR → IDOL: Oct – 2016

SOLUTION

Architecture Of AJAX:

- ⇒ ASP.NET AJAX consists of both server and client components. It is possible to use only the server components, or only the client components.
- ⇒ There is one exception: every ASP.NET AJAX application will need the `ScriptManager` server control. Usually, you will want to use both the server and client components.
- ⇒ The roles the client and server components play in an ASP.NET AJAX project will become clearer when we take a closer look at how Ajax applications that use `XMLHttpRequest` really work.
- ⇒ Below Figure shows the basic structure of ASP.NET AJAX. Whereas standard web pages consist of only two parts—one request and one response—Ajax-enabled web pages can continuously exchange data with the server.
- ⇒ ASP.NET AJAX helps on both ends of the wire. Client script libraries (which, as you will soon see are dynamically loaded by the `ScriptManager` control) facilitate communication between browser and web server and make client coding easier.
- ⇒ The code implemented in the ASP.NET AJAX server assembly takes care of accepting and handling `XMLHttpRequest` calls and also implements some convenient server web controls that we will cover later in the book.
- ⇒ As a result, client and server components can exchange data with very little effort by the programmer.



- ⇒ The ASP.NET AJAX client framework (bottom layer of the client component in above Figure) is sent to the browser from the server the first time an ASP.NET AJAX-enabled page is requested (steps 1 and 2 in Figure).
- ⇒ Subsequent requests to the server from the same page in an Ajax application can then be made with HTTP requests that return text and XML (steps 3 and 4 in Figure).
- ⇒ An ASP.NET web page might use full-page postbacks and asynchronous requests for different tasks.

Q.6. List any five applications where AJAX is incorporated.

ASKED YEAR → IDOL: Dec – 2017 | Apr – 2014

SOLUTION

Five Application where AJAX is Incorporated:

- ⇒ Familiar interactive UI elements such as progress indicators, tooltips, and pop-up windows.
- ⇒ Improved efficiency for Web Forms application, because significant parts of a Web page's processing can be performed in the browser.
- ⇒ Partial-page updates that refresh only the parts of the Web page that have changed.
- ⇒ Client integration with ASP.NET application services for forms authentication, roles, and user profiles.
- ⇒ Auto-generated proxy classes that simplify calling Web service methods from client script.
- ⇒ The ability to customize server controls to include client capabilities.
- ⇒ Support for the most popular browsers, which includes Microsoft Internet Explorer, Mozilla Firefox, and Apple Safari.



AJAX PAGE PROCESSING VS. TRADITIONAL PAGE PROCESSING

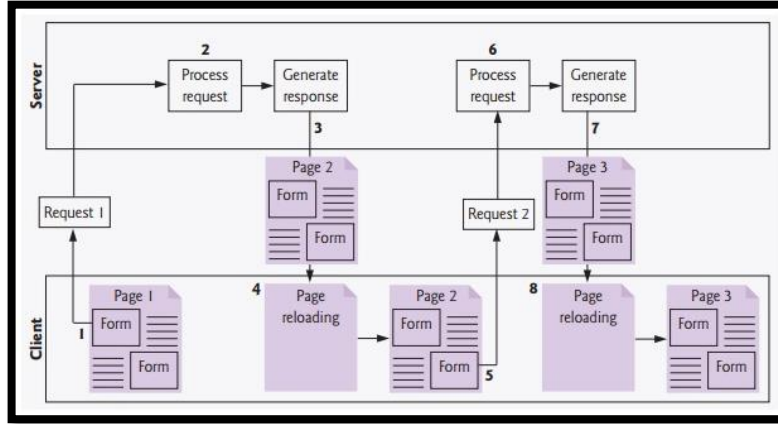
Q.7. Explain the difference between AJAX Page Processing and Traditional Page Processing.

ASKED YEAR → IDOL: May – 2018 | Dec – 2017

SOLUTION

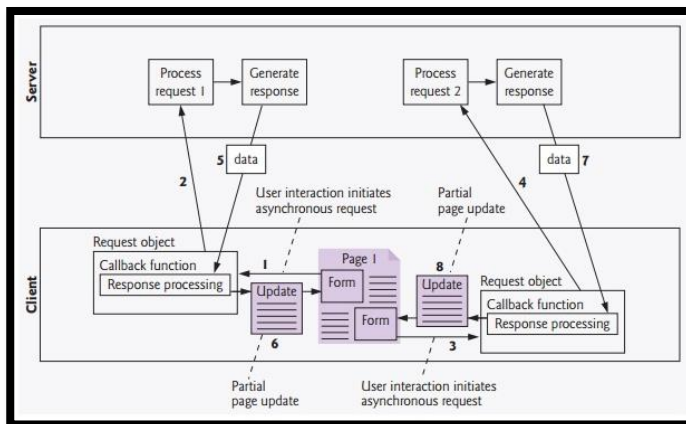
AJAX Page Processing Vs. Traditional Page Processing:

Traditional Web Applications:



- ⇒ Above Figure show presents the typical interactions between the client and the server in a traditional web application, such as one that uses a user registration form.
- ⇒ First, the user fills in the form's fields, then submits the form.
- ⇒ The browser generates a request to the server, which receives the request and processes it.
- ⇒ The server generates and sends a response containing the exact page that the browser will render, which causes the browser to load the new page (Step 4) and temporarily makes the browser window blank.
- ⇒ Note that the client waits for the server to respond and reloads the entire page with the data from the response.
- ⇒ While such a synchronous request is being processed on the server, the user cannot interact with the client web page. Frequent long periods of waiting, due perhaps to Internet congestion, have led some users to refer to the World Wide Web as the "World Wide Wait".
- ⇒ If the user interacts with and submits an-other form, the process begins again.
- ⇒ This model was originally designed for a web of hypertext documents—what some people call the "brochure web". As the web evolved into a full-scale applications platform, the model shown in Figure yielded "choppy" application performance.
- ⇒ Every full-page refresh required users to re-establish their understanding of the full-page contents.
- ⇒ Users began to demand a model that would yield the responsive feel of desktop applications.

Ajax Web Applications:





- ⇒ Ajax applications add a layer between the client and the server to manage communication between the two (Figure).
- ⇒ When the user interacts with the page, the client creates an XMLHttpRequest object to manage a request.
- ⇒ The XMLHttpRequest object sends the request to the server and awaits the response.
- ⇒ The requests are asynchronous, so the user can continue interacting with the application on the client-side while the server processes the earlier request concurrently.
- ⇒ Other user interactions could result in additional requests to the server.
- ⇒ Once the server responds to the original request, the XMLHttpRequest object that issued the request calls a client-side function to process the data returned by the server.
- ⇒ This function known as a callback function uses partial page updates to display the data in the existing web page without re-loading the entire page.
- ⇒ At the same time, the server may be responding to the second re-quest and the client-side may be starting to do another partial page update.
- ⇒ The callback function updates only a designated part of the page.
- ⇒ Such partial page up-dates help make web applications more responsive, making them feel more like desktop applications.
- ⇒ The web application does not load a new page while the user interacts with it.

STEPS FOR CREATING AJAX APPLICATION

Q.8. Explain the basic steps for creating AJAX application with ASP.NET.

ASKED YEAR → IDOL: Oct – 2012

SOLUTION

Steps For Creating AJAX Application With ASP.NET:

STEP 1: Create a new web Form called UpdatePanel.aspx.

STEP 2: Add the label control in the design view.

STEP 3: Drag and Drop the ScriptManager Control.

STEP 4: Drag and Drop the UpdatePanel Control. In the UpdatePanel, place the above Label and the Button Control.

STEP 5: Compile and Execute the form.

STEP 6: Whenever one of the controls within the UpdatePanel causes a postback to the server, only the contents within that UpdatePanel is refreshed. That means when the button is clicked, the contents on the Label get refreshed, but the other controls which are not in the UpdatePanel don't get refreshed.

That means the basic steps for creating AJAX application are:

STEP 1: Create a web form.

STEP 2: Drag and Drop ScriptManager and UpdatePanel Controls.

STEP 3: Then add the controls to be refreshed after postback, in the UpdatePanel.

Example:

```
<asp:ScriptManager runat="server">
</asp:ScriptManager>
<asp:UpdatePanel runat="server">
<ContentTemplate>
<asp:Label runat="server">
<asp:Button runat="server" Text="Button" onClick="Button1_Click"/>
</ContentTemplate>
</asp:UpdatePanel>
```

**Code Behind:**

```
protected void Page_Load(object sender, EventArgs e)
{
    Label1.Text=DateTime.Now.ToString();
}
```

AJAX CONTROLS**Q.9. Explain UpdatePanel Control with example.**

ASKED YEAR ⇒

IDOL: May – 2018 | May – 2016

CBSGS: Apr – 2017 | Nov – 2016 | Nov/Apr – 2015 | Apr – 2014

SOLUTION**UpdatePanel Control:**

- ⇒ ASP.NET UpdatePanel controls enable you to build rich, client-centric Web applications. By using UpdatePanel controls, you can refresh selected parts of the page instead of refreshing the whole page with a postback. This is referred to as performing a partial-page update.
- ⇒ The UpdatePanel control is a container control and derives from the Control class. It acts as a container for the child controls within it and does not have its own interface. When a control inside it triggers a post back, the UpdatePanel intervenes to initiate the post asynchronously and update just that portion of the page.
- ⇒ For example, if a button control is inside the update panel and it is clicked, only the controls within the update panel will be affected, the controls on the other parts of the page will not be affected. This is called the partial post back or the asynchronous post back.

Example: Add an AJAX web form in your application. It contains the script manager control by default. Insert an update panel. Place a button control along with a label control within the update panel control. Place another set of button and label outside the panel.

```
<form id="form1" runat="server">
<div>
<asp:ScriptManager ID="ScriptManager1" runat="server" />
</div>
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
<ContentTemplate>
<asp:Button ID="btnpartial" runat="server" onclick="btnpartial_Click" Text="Partial PostBack"/>
<br/>
<br/>
<asp:Label ID="lblpartial" runat="server"></asp:Label>
</ContentTemplate>
</asp:UpdatePanel>
<p> </p>
<p>Outside the Update Panel</p>
<p>
<asp:Button ID="btntotal" runat="server" onclick="btntotal_Click" Text="Total PostBack" />
</p>
<asp:Label ID="lbltotal" runat="server"></asp:Label>
</form>
```

Properties of the UpdatePanel Control:

- **ChildrenAsTriggers:** This property indicates whether the post backs are coming from the child controls, which cause the UpdatePanel to refresh.
- **ContentTemplate:** It is the content template and defines what appears in the UpdatePanel when it is rendered.
- **ContentTemplateContainer:** Retrieves the dynamically created template container object and used for adding child controls programmatically.
- **IsInPartialRendering:** Indicates whether the panel is being updated as part of the partial post back.
- **RenderMode:** Shows the render modes. The available modes are Block and Inline.
- **UpdateMode:** Gets or sets the rendering mode by determining some conditions.



- **Triggers** : Defines the collection trigger objects each corresponding to an event causing the panel to refresh automatically.

Methods of the UpdatePanel Control:

- **CreateContentTemplateContainer** : Creates a Control object that acts as a container for child controls that define the UpdatePanel Control's content.
- **CreateControlCollection** : Returns the collection of all controls that are contained in the UpdatePanel Control.
- **Initialize** : Initializes the UpdatePanel Control trigger collection if partial-page rendering is enabled.
- **Update** : Causes an update of the content of an UpdatePanel Control.

Q.10.

➤ Explain about ScriptManager Control. [IDOL: May – 2018]

➤ What role does the ScriptManager Play? [CBSGS: April – 2017]

ASKED YEAR →

IDOL: May – 2018

CBSGS: Nov – 2015 | April – 2017

SOLUTION

ScriptManager Control:

- ⇒ The ScriptManager Control is the most important control and must be present on the page for other controls to work.
- ⇒ It has the basic syntax:


```
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
```
- ⇒ If you create an 'Ajax Enabled site' or add an 'AJAX Web Form' from the 'Add Item' dialog box, the web form automatically contains the ScriptManager Control.
- ⇒ The ScriptManager Control takes care of the client-side script for all the Server Side Controls.

Properties of ScriptManager:

- **AllowCustomErrorsRedirect** : Gets or sets a value that determines whether custom errors section of the web.config file is used during error.
- **AsyncPostBackErrorMessage** : Gets or sets the error message that is sent to the client when an unhandled server exception occurs during an asynchronous postback.
- **AsyncPostBackTimeout** : Gets or sets the time in seconds before asynchronous postbacks time if our no response is received.
- **ClientID** : Gets the server control identifier generated by ASP.NET (The id for this control that is rendered on the page)
- **EnablePageMethods** : Gets or sets whether public static page methods in ASP.NET page can be called from client script.
- **EnableViewState** : Gets or sets a value that indicates whether server control persists its ViewState and the ViewState of its child control if any.
- **IsInAsyncPostBack** : Gets or sets a value that indicates whether the current postback is being executed in partial rendering mode.

Role Of ScriptManager:

- ⇒ The ScriptManager control is central to Ajax functionality in ASP.NET. The control manages all ASP.NET Ajax resources on a page. This includes downloading Microsoft Ajax Library scripts to the browser and coordinating partial-page updates that are enabled by using UpdatePanel controls. In addition, the ScriptManager control enables you to do the following:
- ⇒ Register script that is compatible with partial-page updates. In order to manage dependencies between your script and the core library, any script that you register is loaded after the Microsoft Ajax Library script.
- ⇒ Specify whether release or debug scripts are sent to the browser.
- ⇒ Provide access to Web service methods from script by registering Web services with the ScriptManager control.
- ⇒ Provide access to ASP.NET authentication, role, and profile application services from client script by registering these services with the ScriptManager control.
- ⇒ Enable culture-specific display of ECMAScript (JavaScript) Date, Number, and String functions in the browser.
- ⇒ Access localization resources for embedded script files or for stand-alone script files by using the ResourceUICultures property of the ScriptReference control.
- ⇒ Register server controls that implement the IExtenderControl or IScriptControl interfaces with the ScriptManager control so that script required by client components and behaviors is rendered.

**Example:**

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="how-to-use-ScriptManager.aspx.vb"
Inherits="how_to_use_ScriptManager" %>
<!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>How to Use ScriptManager</title>
</head>
<body>
<form id="form1" runat="server">
<asp:ScriptManager ID="ScriptManager1" runat="server" />
<asp:UpdatePanel runat="server" id="UpdatePanel1">
<ContentTemplate>
<asp:Button runat="server" id="Button1" text="Update" onclick="Button1_Click" />
<br /><br />
<asp:Label runat="server" id="Label1" />
</ContentTemplate>
</asp:UpdatePanel>
</form>
</body>
</html>
```

Code Behind:

```
Partial Class how_to_use_ScriptManager
Inherits System.Web.UI.Page
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles
Button1.Click
Label1.Text = "Random Number : " & New Random().Next().ToString()
End Sub
End Class
```

Q.11. Explain UpdateProgress Control in Ajax.

ASKED YEAR ⇒

IDOL: May – 2017

CBSGS: Nov – 2017 | Nov – 2014 | Oct – 2013

SOLUTION**UpdateProgress Control:**

- ⇒ To keep the user updated on the progress while the message is delivered to the server, we should add an UpdateProgress control.
- ⇒ The UpdateProgress control provides a sort of feedback on the browser while one or more update panel controls are being updated.
- ⇒ For example, while a user logs in or waits for server response while performing some database oriented job.
- ⇒ It provides a visual acknowledgement like "Loading page...", indicating the work is in progress.

The syntax for the UpdateProgress Control is:

```
<asp:UpdateProgress ID="UpdateProgress1" runat="server" DynamicLayout="true"
AssociatedUpdatePanelID="UpdatePanel1">
<ProgressTemplate>
Loading...
</ProgressTemplate>
</asp:UpdateProgress>
```

- ⇒ The above snippet shows a simple message within the ProgressTemplate tag.
- ⇒ However, it could be an image or other relevant controls.
- ⇒ The UpdateProgress control displays for every asynchronous postback unless it is assigned to a single update panel using the AssociatedUpdatePanelID property.

Properties of the UpdateProgress Control:

The following table shows the properties of the UpdateProgress Control:

- **AssociatedUpdatePanelID:** Gets and sets the ID of the update panel with which this control is associated.
- **Attributes:** Gets or sets the cascading style sheet (CSS) attributes of the UpdateProgress control.
- **DisplayAfter:** Gets and sets the time in milliseconds after which the progress template is displayed. The default is 500.
- **DynamicLayout:** Indicates whether the progress template is dynamically rendered.
- **ProgressTemplate:** Indicates the template displayed during an asynchronous post back which takes more time than the DisplayAfter time.

Methods of the UpdateProgress Control:

The following table shows the methods of the UpdateProgress Control:

- **GetScriptDescriptors:** Returns a list of components, behaviors, and client controls that are required for the UpdateProgress control's client functionality.
- **GetScriptReferences:** Returns a list of client script library dependencies for the UpdateProgress control.

Example:

```
<form id="form1" runat="server">
<asp:ScriptManager ID="ScriptManager1" runat="server" />
<asp:UpdateProgress runat="server" id="PageUpdateProgress">
<ProgressTemplate>
Loading...
</ProgressTemplate>
</asp:UpdateProgress>
<asp:UpdatePanel runat="server" id="Panel">
<ContentTemplate>
<asp:Button runat="server" id="UpdateButton" onclick="UpdateButton_Click" text="Update" />
</ContentTemplate>
</asp:UpdatePanel>
</form>
Code-Behind:
protected void UpdateButton_Click(object sender, EventArgs e)
{
System.Threading.Thread.Sleep(5000);
}
```

Q.12.

➤ Explain the Timer Control with its Attributes. [IDOL: May – 2017] | [CBSGS: Nov – 2016]

➤ Explain the use of Timer Control in AJAX. [IDOL: Oct – 2016] | [CBSGS: Apr – 2016 | Oct – 2013]

ASKED YEAR ⇒

IDOL: May – 2017 | Oct – 2016

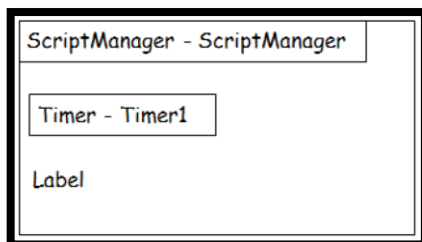
CBSGS: Nov/Apr – 2016 | Oct – 2013

SOLUTIONTimer Control:

- ⇒ The Timer Control is used to initiate the post back automatically.
- ⇒ If you want page refresh without user's reaction to the page, use a timer control.
- ⇒ Timer control performs an action which you will be writing under timer - click even offer every particular interval of time.

Example: Include a ScriptManager control and UpdatePanel control in a webpage also add timer control as shown below.

Add time control UpdatePanel.





It will generate the following source code:

```
<asp:ScriptManager id="SC1" runat="server">
</asp:ScriptManager>
<asp:UpdatePanel id="UP1" runat="server">
</asp:UpdatePanel>
<asp:UpdatePanel>
<asp:UpdatePanel id="UP2" runat="server">
<contentTemplate>
<asp:Timer id="t1" runat="server" Interval="6000">
<br/>
<asp:label id="l1" runat="server" text="Label">
</asp:label>
</contentTemplate>
</asp:UpdatePanel>
```

Then add following code to your webpage.

```
protected void T1_click(object sender, EventArgs e)
{
    l1.Text=System.DateTime.Now.ToString();
}
```

The above code updates Label with current date & time after every 6000 milliseconds.

Q.13.

What are the application services provided in ASP.NET? Explain.

ASKED YEAR

CBSGS: Oct – 2012

SOLUTION

Application Services:

- ⇒ ASP.NET Applications Services are built-in Web Services that provide access to features such as forms authentication, roles, and profile properties.
- ⇒ These services are part of a Service-Oriented Architecture (SOA), in which an application consists of one or more services provided on the server, and one or more clients.
- ⇒ An important feature of ASP.NET Application Services is that they are available to a variety of client applications, not just ASP.NET Web applications.
- ⇒ ASP.NET application services are available to any client that is based on the .NET Framework. In addition, any client application that can send and receive messages in SOAP format can use ASP.NET application services.
- ⇒ Client applications for ASP.NET Applications Services can be of different types and can run on different Operating Systems. These include the following types of clients:
 - **AJAX Clients:** These are ASP.NET Web pages (.aspx files) that run in the browser and that access application services from client script.
 - **.NET Framework Clients:** These are .NET Framework Windows applications that access application services over HTTP by using the provider model infrastructure.
 - **SOAP Clients:** These are clients that can access application services through SOAP 1.1.
- ⇒ The application services provided by ASP.NET enable client applications to access and share information that is part of a Web Application. ASP.NET makes available the following application services:
 - **Authentication Service:** This service enables you to let users log onto an application. The service accepts user credentials and returns an authentication ticket (cookie).
 - **Roles Service:** This service determines the application-related roles for an authenticated user, based on information that is made available by an ASP.NET roles provider.
 - **Profile Service:** This service provides per-user information as a user profile that is stored on the server. This enables your application to access a user's settings at different times and from different client UI Components.



WEB SERVICES

Q.14. What is Web Service? Explain the basic steps to create a Web Service using ASP.NET with C#.

ASKED YEAR ⇒ IDOL: May – 2018 | Dec – 2017 | Oct – 2016 | Apr – 2014 | Oct – 2012

SOLUTION

Web Service:

- ⇒ Web Services can convert your application into a Web-Application, which can publish its function or message to the rest of the world.
- ⇒ The basic Web Services platform is XML + HTTP.
- ⇒ A Web Service is a Web Application which is basically a class consisting of methods that could be used by other applications.
- ⇒ It also follows a Code-Behind Architecture like the ASP.Net Web Pages, although it does not have a User Interface.

Basic Steps For Creating 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". (Any names can be there depending on the example)

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.

STEP 6: Change the code behind file according to the service requirements.

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.

ADVANTAGES OF WEB SERVICES

Q.15. What are the advantages of Web Services?

ASKED YEAR ⇒ IDOL: Apr – 2015

SOLUTION

Advantages Of Web Services:

Interoperability:

This is the most important benefit of Web Services. Web Services typically work outside of private networks, offering developers a non-proprietary route to their solutions. Services developed are likely, therefore, to have a longer life-span, offering better return on investment of the developed service. Web Services also let developers use their preferred programming languages. In addition, thanks to the use of standards-based communications methods, Web Services are virtually platform-independent.

Usability:

Web Services allow the business logic of many different systems to be exposed over the Web. This gives your applications the freedom to choose the Web Services that they need. Instead of re-inventing the wheel for each client, you need only include



additional application-specific business logic on the client-side. This allows you to develop services and/or client-side code using the languages and tools that you want.

Reusability:

Web Services provide not a component-based model of application development, but the closest thing possible to zero-coding deployment of such services. This makes it easy to reuse Web Service components as appropriate in other services. It also makes it easy to deploy legacy code as a Web Service.

Deployability:

Web Services are deployed over standard Internet technologies. This makes it possible to deploy Web Services even over the fire wall to servers running on the Internet on the other side of the globe. Also thanks to the use of proven community standards, underlying security (such as SSL) is already built-in.

Q.16. List the necessary steps to publish the website in ASP.NET.

ASKED YEAR → IDOL: Oct – 2012

SOLUTION

Steps To Publish The Website In ASP.NET:

STEP 1:

- ⇒ On the Build Menu, Click Publish Web Site.
- ⇒ The Publish Web Site Dialog Box appears.

STEP 2:

- ⇒ In the Target Location Box, enter the directory of our choice, say for example `c:\CompiledSite`.
- ⇒ The Allow this precompiled site to be updatable option specifies that all program code is compiled into assemblies, but that ".aspx" files (including single-file ASP.NET Web pages) are copied as-is to the target folder.
- ⇒ In this walkthrough, we will not select that option.

STEP 3:

- ⇒ Click OK.
- ⇒ Visual Web Developer precompiles the contents of the Web site and writes the output to the folder that we specified.
- ⇒ The Output Window Displays Progress Messages.
- ⇒ If an error occurs during compilation, it is reported in the Output Window.

STEP 4:

- ⇒ If errors occur during publishing, fix the errors, and then repeat STEP 1.

Publishing by Copying:

STEP 1: Copy the Web Folders

- ⇒ Copy our website (all folders and content) from our development computer to an application folder on our Remote Hosting Computer (server).
- ⇒ If our App_Data folder contains test data, don't copy the App_Data folder

STEP 2: Copy the DLL Files

- ⇒ On the Remote Server, create a bin folder in the root of our application. (if we have installed Helpers, we already have a bin folder)
- ⇒ Copy everything from our folder:
`C:\Program Files (x86)\Microsoft ASP.NET\ASP.NET Web Pages\v1.0\Assemblies`
- ⇒ To our applications bin folder on the Remote Server.

**STEP 3: Copy the SQL Server Compact DLL Files**

- ⇒ If our application has a SQL Server Compact Database (an ".sdf" file in App_Data folder), we must copy the SQL Server Compact DLL files.
- ⇒ Copy everything from our folder:
C:\Program Files (x86)\Microsoft SQL Server Compact Edition\v4.0\Private
- ⇒ To our applications bin folder on the Remote Server.

STEP 4:

- ⇒ Copy SQL Server Compact Data.

XMLHttpRequest**Q.17. Explain XMLHttpRequest Object with its properties and methods.**ASKED YEAR ⇒ IDOL: Apr – 2015**SOLUTION****XMLHttpRequest:**

- ⇒ The XMLHttpRequest object is the key to AJAX. It has been available ever since Internet Explorer 5.5 was released in July 2000, but was not fully discovered until AJAX and Web 2.0 in 2005 became popular.
- ⇒ XMLHttpRequest (XHR) is an API that can be used by JavaScript, JScript, VBScript, and other web browser scripting languages to transfer and manipulate XML data to and from a webserver using HTTP, establishing an independent connection channel between a webpage's Client-Side and Server-Side.
- ⇒ The data returned from XMLHttpRequest calls will often be provided by back-end databases.
- ⇒ Besides XML, XMLHttpRequest can be used to fetch data in other formats, e.g. JSON or even plain text.

Properties of XMLHttpRequest:

- **onreadystatechange**: An event handler for an event that fires at every state change.
- **readyState**: The readyState property defines the current state of the XMLHttpRequest object.
- **responseText**: Returns the response as a string.
- **responseXML**: Returns the response as XML. This property returns an XML document object, which can be examined and parsed using the W3C DOM node tree methods and properties.
- **status**: Returns the status as a number (e.g., 404 for "Not Found" and 200 for "OK").
- **statusText**: Returns the status as a string (e.g., "Not Found" or "OK").

Methods of XMLHttpRequest:

- **abort()** – Cancels the current request.
- **getAllResponseHeaders()** – Returns the complete set of HTTP headers as a string.
- **getResponseHeader(headerName)** – Returns the value of the specified HTTP header.
- **open(method, URL, async, userName, password)** – Specifies the method, URL, and other optional attributes of a request. The method parameter can have a value of "GET", "POST", or "HEAD". Other HTTP methods such as "PUT" and "DELETE" (primarily used in REST applications) may be possible. The "async" parameter specifies whether the request should be handled asynchronously or not. "true" means that the script processing carries on after the send() method without waiting for a response, and "false" means that the script waits for a response before continuing script processing.
- **send(content)** – Sends the request.
- **setRequestHeader(label, value)** – Adds a label/value pair to the HTTP header to be sent.

**JQUERY****Q.18. What is jQuery? How to use jQuery in ASP Pages?**

ASKED YEAR →

IDOL: Apr – 2013

CBSGS: Apr – 2016

SOLUTION**jQuery:**

- ⇒ jQuery emphasize the communication between JavaScript and HTML.
- ⇒ It provides fast and easy way of HTML DOM traversing and manipulation, event handling and client side animation etc.
- ⇒ jQuery also supports an efficient easy to implement AJAX applications.
- ⇒ The purpose of jQuery is to make it much easier to use JavaScript on our website.
- ⇒ jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that we can calls with a single line of code.
- ⇒ jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM Manipulation.
- ⇒ The jQuery library contains the following features:
 - *HTML/DOM Manipulation*
 - *CSS Manipulation*
 - *HTML Event Methods*
 - *Effects and Animations*

Use Of jQuery In ASP.NET Pages:**Adding jQuery to our Web Pages:**

There are several ways to start jQuery on our web site. We can download the jQuery library from jQuery.com

Downloading jQuery:

There are two versions of jQuery available for downloading. It can be downloaded from jQuery.com.

The jQuery library is a single JavaScript file, and we reference it with the following code:

```
<asp:ScriptManager runat="server">
<scripts>
<asp:ScriptReference path="~/Scripts/jquery-1.4.1.min.js"/>
</scripts>
</asp:ScriptManager>
```

The downloaded file should be placed in the same directory as the pages where we wish to use it.

```
<body>
<form runat="server">
<div>
</div>
<asp:ScriptManager runat="server">
<scripts>
<asp:ScriptManager path="~/Scripts/jquery-1.4.1.min.js"/>
</scripts>
</asp:ScriptManager>
<asp:Button runat="server" text="Button"/>
<script type="text/javascript">
$(document).ready(function() {
$('h2').css('color','red')
});
</script>
<h2> This is header Level 2.</h2>
</form>
</body>
</html>
```

**Q.19.**

- **Brief the concept of jQuery.** [IDOL: May – 2018]
- **Explain jQuery Expression with example.** [CBSGS: Apr – 2015]

ASKED YEAR ⇒ IDOL: May – 2018 | May – 2017 | Oct – 2012

SOLUTION

Concept Of jQuery:

- ⇒ jQuery is a lightweight, "write less, do more", JavaScript Library.
- ⇒ jQuery is easy to learn.
- ⇒ The purpose of jQuery is to make it much easier to use JavaScript on your website.
- ⇒ jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you call with a single line of code.
- ⇒ jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.

The jQuery Library contains the following features:

- *HTMT/DOM Manipulation*
- *CSS Manipulation*
- *HTML Event Methods*
- *Effects and Animations*

jQuery Syntax: (CBSGS: Apr – 2015)

- ⇒ With jQuery select (query) HTML elements and perform "actions" on them.
- ⇒ The jQuery syntax is tailor made for selecting HTML elements and performing some action on the element(s).

Basic Syntax:

```
$(selector).action()
```

- ⇒ A \$ sign to define/access jQuery.
- ⇒ A (selector) to "query (or find)" HTML elements.
- ⇒ A jQuery action () to be performed on the element (s).

Examples:

- `$(this).hide()` – Hides the current element.
- `$("p").hide()` – Hides all <p> elements.
- `$(".test").hide()` – Hides all elements with class="test".
- `$("#test").hide()` – Hides the element with id="test".

Example:

```
<html>
<head runat="server">
<script type="text/JavaScript" src="jquery.js">
$(document).ready(function()
{
$("button").click(function()
{
$("table").hide();
});
});
</script>
</head>
<body runat="server">
<table border="1">
<tr><td>1234</td></tr>
</table>
<input type="button" value="hide"/>
</body>
</html>
```



FEATURES OF JQUERY

Q.20. What are the features of jQuery?

ASKED YEAR ⇒ **IDOL:** Oct – 2016

SOLUTION

Features Of jQuery:

- ⇒ **DOM Manipulation:** The jQuery made it easy to select DOM elements, negotiate them and modifying their content by using cross-browser open source selector engine called Sizzle.
- ⇒ **Event Handling:** The jQuery offers an elegant way to capture a wide variety of events, such as a user clicking on a link, without the need to clutter the HTML code itself with event handlers.
- ⇒ **AJAX Support:** The jQuery helps you a lot to develop a responsive and feature rich site using AJAX technology.
- ⇒ **Animations:** The jQuery comes with plenty of built-in animation effects which you can use in your websites.
- ⇒ **Lightweight:** The jQuery is very lightweight library – about 19KB in size (Minified and zipped).
- ⇒ **Cross Browser Support:** The jQuery has cross-browser support, and works well in IE 6.0+, FF 2.0+, Safari 3.0+, Chrome and Opera 9.0+
- ⇒ **Latest Technology:** The jQuery supports CSS3 selectors and basic XPath syntax.

DOLLAR SYMBOLS (\$) IN JQUERY

Q.21. What does Dollar Symbols (\$) mean in jQuery? Explain with example.

ASKED YEAR ⇒ **IDOL:** Dec – 2017

SOLUTION

Dollar Symbols (\$) In jQuery:

The jQuery syntax is specially made for selecting HTML elements and performing some action on these elements.

Syntax:

```
$(selector).action()
```

where,

\$ sign defines jQuery

(selector) finds HTML elements

action() performs some actions on the elements

Example: `$(this).hide()` – Hides Current Element.

```
<html>
<head>
<script type="text/javascript" src="jquery.js"></script>
<script type="text/javascript">
$(document).ready(function() {
$("button").click(function() {
$(this).hide();
});
});
</script>
</head>
<body>
<button>Hide Me</button>
</body>
</html>
```



DOCUMENTREADY FUNCTION

Q.22. Explain the use of `document.ready()` function in jQuery.

ASKED YEAR →

CBSGS: Nov/Apr – 2017 | Apr – 2016 | Apr – 2015 | Apr – 2014

SOLUTION

DocumentReady Function:

- ⇒ Most of the jQuery code will be executed when the browser is done loading the page.
- ⇒ It is important to wait with executing our code until the page is done loading the Document Object Model which is a hierarchical representation of the web page and contains a tree like structure of all HTML elements.
- ⇒ If the jQuery is executed too early, then the DOM may not have been loaded the elements we are referring to in our script. So, we may get errors.
- ⇒ Fortunately, it is easy to postpone the execution of our code until the DOM is ready function in jQuery.
- ⇒ It is used to prevent any jQuery code from running before the document is finished loading there are few functions which can fail if they run before the document is loaded.

Syntax:

```
$(document).ready(function(){  
  //jQuery methods go here...  
});
```

Example:

```
<html>  
<head runat="server">  
<script type="text/JavaScript" src="jquery.js">  
$(document).ready(function()  
{  
  $("button").click(function()  
{  
    $("table").hide();  
  });  
});  
</script>  
</head>  
<body runat="server">  
<table border="1">  
<tr><td>1234</td></tr>  
</table>  
<input type="button" value="hide"/>  
</body>  
</html>
```



JQUERY EVENT FUNCTIONS

Q.23. Write short note on jQuery Event Functions.

ASKED YEAR ⇒ IDOL: May – 2016

CBSGS: Apr – 2016 | Apr – 2015

SOLUTION

jQuery Event Functions:

- ⇒ Event method trigger or attach a function to an event handler for the selected elements.
- ⇒ These methods are used to register behaviour to take place effect when the user interacts with the browser and to further manipulate there registered behaviour.

Commonly used jQuery Event Functions:

- **\$(document).ready()** – This method allows us to execute a function when the document is fully loaded.
- **click()** – This method is executed when the user clicks on the HTML element.
- **bind()** – This method attaches one or more event handlers for selected elements, and specifies a function to run when the event occurs.
- **die()** – This method removes one or more event handlers, added with the `live()` method, for the selected elements.
- **change()** – This event occurs when the value of an element has been changed (only in `<input>`).
- **blur()** – The `blur()` event occurs when an element loses focus.
- **error()** – The `error()` method triggers the error event, or attaches a function to run when an error event occurs.

IMPORTANCE OF JQUERY

Q.24. What is the importance of jQuery in the Development of web Applications?

ASKED YEAR ⇒ IDOL: Apr – 2014

SOLUTION

Importance Of jQuery:

- ⇒ **Ease of Use:** This is pretty much the main advantage of using JQuery, it is a lot more easy to use compared to standard JavaScript and other JavaScript libraries. Apart from simple syntax, it also requires much less lines of code to achieve the same feature in comparison.
- ⇒ **Large Library:** JQuery enables you to perform hordes of functions in comparison to other JavaScript libraries.
- ⇒ **Strong Opensource Community (Several jQuery Plugins Available):** JQuery, while relatively new, has a following that religiously devote their time to develop and enhance the functionality of JQuery. Thus there are hundreds of prewritten plugins available for download to instantly speed up your development process. Another advantage behind this is the efficiency and security of the script.
- ⇒ **Great Documentation and Tutorials:** The jQuery website has a comprehensive documentation and tutorials to get even an absolute beginner in programming to get the ball rolling with this library.
- ⇒ **Ajax Support:** JQuery lets you develop Ajax templates with ease, Ajax enables a sleeker interface where actions can be performed on pages without requiring the entire page to be reloaded.



BENEFITS OF USING JQUERY

Q.25. List and explain the benefits of using jQuery.

ASKED YEAR → IDOL: Apr – 2013

SOLUTION

Benefits Of Using jQuery:

- **Large Library**: jQuery provides a large collection of functions in comparison to other JavaScript libraries.
- **Save Time**: Number of lines of code we need to write is comparatively very small with jQuery.
- **Cross Browser Friendly**: jQuery is currently the most popular JavaScript library and works in all browsers.
- **Plug-ins**: There are an abundance of plug-ins on the web that make creating special effects simple and fast for web developers.
- **That was easy!** – jQuery has easy implementation for web developers in comparison to other applications.
- **FREE!** – It is free, open source software.
- **Events**: It presents a single events Application Program Interface (API).
- **Effects**: It provides the effects such as fading, sliding and many more.
- **Mobile Devices**: jQuery is supported by any mobile device whose web browser supports JavaScript. A lot of mobile devices like iPads and iPhones don't run Flash at all.
- **Simplifies AJAX**: jQuery lets us to develop AJAX templates with ease, AJAX enables a sleeker interface where actions can be performed on pages without requiring the entire page to be reloaded.
- **Wow Factor**: Web developers use jQuery to make web pages more exciting, interactive, cleaner, and more user friendly. Make our users go WOW!

JQUERY SELECTOR

Q.26. What is jQuery Selector? Write some examples.

ASKED YEAR → IDOL: Dec/May – 2017 | May – 2016 | Apr – 2014 | Apr – 2013

CBSGS: Nov – 2016 | Oct – 2013

SOLUTION

jQuery Selector:

- ⇒ jQuery selectors allow you to select and manipulate HTML element(s).
- ⇒ jQuery selectors are used to "find" (or select) HTML elements based on their id, classes, types, attributes, values of attributes and much more.
- ⇒ It's based on the existing CSS Selectors, and in addition, it has some own custom selectors.
- ⇒ All selectors in jQuery start with the dollar sign and parentheses: \$ () .

Selectors present in JQuery:

Element Selector ("element"):

- ⇒ The jQuery element selector selects elements based on the element name. \$ ("p")
- ⇒ JavaScript `getElementsByTagName ()` function is called to return the appropriate elements when this expression is used.

Syntax:

```
jQuery("element")
```

Where,

Element – An element to search for. Refers to the tagName of DOM nodes.

Example:

```
<html>
<head>
<title>element demo</title>
<style>
div, span
```



```
{
width: 60px;
height: 60px;
float: left;
padding: 10px;
margin: 10px;
background-color: #eee;
}
</style>
<script src="https://code.jquery.com/jquery-1.10.2.js"></script>
</head>
<body>
<div>DIV1</div>
<div>DIV2</div>
<span>SPAN</span>
<script>
$( "div" ).css( "border", "9px solid red" );
</script>
</body>
</html>
```

ID Selector ("#id"):

- ⇒ The jQuery #id selector uses the id attribute of an HTML tag to find the specific element.
- ⇒ An id should be unique within a page, so you should use the #id selector when you want to find a single, unique element.
- ⇒ For id selectors, jQuery uses the JavaScript function `document.getElementById()`, which is extremely efficient. When another selector is attached to the id selector, such as `h2#pageTitle`, jQuery performs an additional check before identifying the element as a match.
- ⇒ Calling `jQuery()` (or `$()`) with an id selector as its argument will return a jQuery object containing a collection of either zero or one DOM element.
- ⇒ Each id value must be used only once within a document. If more than one element has been assigned the same ID, queries that use that ID will only select the first matched element in the DOM. This behavior should not be relied on, however; a document with more than one element using the same ID is invalid.
- ⇒ If the id contains characters like periods or colons you have to escape those characters with backslashes.

Syntax:

```
jQuery("#id")
```

Where,

Id – An ID to search for, specified via the id attribute of an element.

Example:

```
<html>
<head>
<title>id demo</title>
<style>
div
{
width: 90px;
height: 90px;
float: left;
padding: 5px;
margin: 5px;
background-color: #eee;
}
</style>
<script src="https://code.jquery.com/jquery-1.10.2.js"></script>
</head>
<body>
<div id="notMe"><p>id="notMe"</p></div>
```



```
<div id="myDiv">id="myDiv"</div>
<script>
$( "#myDiv" ).css( "border", "3px solid red" );
</script>
</body>
</html>
```

Class Selector (".class"):

- ⇒ Selects all elements with the given class.
- ⇒ To find elements with a specific class, write a period character, followed by the name of the class: \$(".test")
- ⇒ For class selectors, jQuery uses JavaScript's native `getElementsByClassName()` function if the browser supports it.

Syntax:

```
jQuery(".class")
```

Where,

class – A class to search for. An element can have multiple classes; only one of them must match.

Example:

```
<html>
<head>
<title>class demo</title>
<style>
div, span
{
width: 120px;
height: 40px;
float: left;
padding: 10px;
margin: 10px;
background-color: #EEEEEE;
}
</style>
<script src="https://code.jquery.com/jquery-1.10.2.js"></script>
</head>
<body>
<div class="notMe">div class="notMe"</div>
<div class="myClass">div class="myClass"</div>
<span class="myClass">span class="myClass"</span>
<script>
$( ".myClass" ).css( "border", "3px solid red" );
</script>
</body>
</html>
```

Universal Selector (*):

- ⇒ Selects all elements available in a DOM.
- ⇒ The universal selector selects all the elements available in the document.

Syntax:

```
$('*')
```

Where,

***** – A symbolic star

Example:

```
<html>
<head>
<title>The Selector Example</title>
```



```
<script type = "text/javascript"
src = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
</script>
<script type = "text/javascript" language = "javascript">
$(document).ready(function()
{
/* This would select all the elements */
$("").css("background-color", "yellow");
});
</script>
</head>
<body>
<div class = "big" id = "div1">
<p>This is first division of the DOM.</p>
</div>
<div class = "medium" id = "div2">
<p>This is second division of the DOM.</p>
</div>
<div class = "small" id = "div3">
<p>This is third division of the DOM</p>
</div>
</body>
</html>
```

Multiple Elements Selector ("selector1, selector2, selectorN"):

- ⇒ Selects the combined results of all the specified selectors.
- ⇒ We can specify any number of selectors to combine into a single result.
- ⇒ This multiple expression combinator is an efficient way to select disparate elements.
- ⇒ The order of the DOM elements in the returned jQuery object may not be identical, as they will be in document order.
- ⇒ An alternative to this combinator is the `.add()` method.

Syntax:

```
jQuery("selector1, selector2, selectorN")
```

Where,

- **selector1:** Any valid selector.
- **selector2:** Another valid selector.
- **selectorN:** As many more valid selectors as you like.

Example:

```
<html>
<head>
<title>multiple demo</title>
<style>
div, span, p
{
width: 126px;
height: 60px;
float: left;
padding: 3px;
margin: 2px;
background-color: #eee;
font-size: 14px;
}
</style>
<script src="https://code.jquery.com/jquery-1.10.2.js"></script>
</head>
<body>
<div>div</div>
<p class="myClass">p class="myClass"</p>
```



```
<p class="notMyClass">p class="notMyClass"</p>
<span>span</span>
<script>
$("div, span, p.myClass").css("border", "3px solid red");
</script>
</body>
</html>
```

DOM MANIPULATION METHODS

Q.27. Explain DOM Manipulation Methods in jQuery.

ASKED YEAR →

IDOL: May – 2018 | May – 2017 | Oct/ May – 2016 | Apr – 2015 | Apr – 2013 | Oct – 2012

CBSGS: Apr – 2017

SOLUTION

DOM Manipulation Methods:

Following are the lists of all the methods which we can use to manipulate DOM elements:

- **after (content)** : Insert content after each of the matched elements.
- **append (content)** : Append content to the inside of every matched element.
- **appendTo (selector)** : Append all of the matched elements to another, specified, set of elements.
- **before (content)** : Insert content before each of the matched elements.
- **clone (bool)** : Clone matched DOM Elements, and all their event handlers, and select the clones.
- **clone ()** : Clone matched DOM Elements and select the clones.
- **empty ()** : Remove all child nodes from the set of matched elements.
- **html (val)** : Set the html contents of every matched element.
- **html ()** : Get the html contents (innerHTML) of the first matched element.
- **insertAfter (selector)** : Insert all of the matched elements after another, specified, set of elements.
- **insertBefore (selector)** : Insert all of the matched elements before another, specified, set of elements.
- **prepend (content)** : Prepend content to the inside of every matched element.
- **prependTo (selector)** : Prepend all of the matched elements to another, specified, set of elements.
- **remove (expr)** : Removes all matched elements from the DOM.
- **replaceAll (selector)** : Replaces the elements matched by the specified selector with the matched elements.
- **replaceWith (content)** : Replaces all matched elements with the specified HTML or DOM elements.
- **text (val)** : Set the text contents of all matched elements.
- **text ()** : Get the combined text contents of all matched elements.
- **wrap (elem)** : Wrap each matched element with the specified element.
- **wrap (html)** : Wrap each matched element with the specified HTML content.
- **wrapAll (elem)** : Wrap all the elements in the matched set into a single wrapper element.
- **wrapAll (html)** : Wrap all the elements in the matched set into a single wrapper element.
- **wrapInner (elem)** : Wrap the inner child contents of each matched element (including text nodes) with a DOM element.
- **wrapInner (html)** : Wrap the inner child contents of each matched element (including text nodes) with an HTML structure.

**PROGRAM**

I | II | III | IV | V | VI

Q.1.**Write a windows application to open the website mu.ac.in upon click the link. Write the necessary Events and steps required for implementation. [Hint: Use LinkLabel]**

ASKED YEAR →

IDOL: December – 2017

SOLUTION**PROGRAM:****STEP 1:** Create a new windows application.**STEP 2:** Drag and drop LinkLabel control from the toolbox on the designer window.**STEP 3:** Double click on the event and write the following event.

```
private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedArgs e)
{
    System.Diagnostics.Process.Start("www.mu.ac.in");
}
```

STEP 4: Execute the application, and click the link.**STEP 5:** www.mu.ac.in site will be visible.**Q.2.****Write ASP.NET Code to display selected elements from the CheckBoxList on a label control. Elements on the label must be separated by a whitespace.**

ASKED YEAR →

CBSGS: November – 2017

SOLUTION**PROGRAM:**

```
protected void btShow_Click(object sender, EventArgs e)
{
    string strAllSelectect="";
    foreach (ListItem val in chkSubjectsList.Items)
    {
        if (val.Selected)
        {
            strAllSelectect += val.Value + " ";
        }
    }
    lblShowSelected.Text = strAllSelectect;
}
```

OUTPUT :

Select desired options from following list

- ASP.NET
- Linux
- Java
- Network
- Software

Show

ASP.NET Java Network

**Q.3.**

**Write ASP.NET Code to send data entered in two textboxes from one web page to another web page.
Display the data on two separate labels.**

ASKED YEAR →

CBSGS: November – 2017

SOLUTION**PROGRAM:**

The screenshot shows a web form with two textboxes and a button. The first textbox is labeled "Enter your first name:" and contains the text "John". The second textbox is labeled "Enter contact number:" and contains the text "1234567890". Below the textboxes is a button labeled "Show on next page".

Code(.cs file):

```
protected void btShowOnNextPage_Click(object sender, EventArgs e)
{
    String name = Server.UrlEncode(txtName.Text);
    String contact = Server.UrlEncode(txtContactNo.Text);
    String Url = "DisplayData.aspx?" + "nm=" + name + "&contact=" + contact;
    Response.Redirect(Url);
}
```

Code for next page(.cs file):

```
protected void Page_Load(object sender, EventArgs e)
{
    String strName = Server.UrlDecode(Request.QueryString["nm"]);
    String strContact = Server.UrlDecode(Request.QueryString["contact"]);
    lblName.Text = "User Name: " + strName;
    lblContact.Text = "Number: " + strContact;
}
```

Output:

```
User Name: John
Number: 1234567890
```

Q.4.

Write a code to insert and update data into a SqlServer Database an ASP.NET Web Page.

ASKED YEAR →

CBSGS: November – 2017

SOLUTION**PROGRAM:**

```
protected void cmdAdd_Click(object sender, EventArgs e)
{
    SqlConnection conn = new SqlConnection();
    conn.ConnectionString = ("Data Source=ADMIN-PC\\SQLEXPRESS;Initial Catalog=TYIT;Integrated Security=True");
    string query = "INSERT INTO Student (RollNo,FName,Marks,Address) VALUES (' " + txtRollNo.Text + " ',' " + txtFName.Text + " ',' " + txtMarks.Text + " ',' " + txtAdd.Text + " ' )";
    try
    {
        conn.Open();
        SqlCommand cmd = new SqlCommand(query, conn);
        cmd.ExecuteNonQuery();
        lblMessage.Text="Data Added Successfully";
    }
    catch (Exception ex)
```



```
{
lblMessage.Text="Problem in connection or in database";
}
finally
{
conn.Close();
}
}
```

Q.5.**Write a code to display all the number in array greater than 10 using LINQ.**

ASKED YEAR ⇒

CBSGS: November – 2017

SOLUTION**PROGRAM:**

```
static void Main()
{
// array as data source.
int[] arr = { 34, 7, 82, 8, 75, 6,12,3,38 };
// Query Expression.
IEnumerable<int> resultQuery =
from n in arr
where n > 10
select score;
// Execute the query to produce the results
foreach (int num in resultQuery)
{
Console.WriteLine(num);
}
}
```

Output:

```
34
82
75
12
38
```

Q.6.**Write jQuery code to demonstrate the use of hide () and slideUp () functions on <p> element.**

ASKED YEAR ⇒

CBSGS: November – 2017

SOLUTION**PROGRAM:****Use of Hide() Method:**

```
<!DOCTYPE html>
<html> <head>
<style>
p { color:red; margin:5px; cursor:pointer; }
p:hover { background:yellow; }
</style>
<script src="http://code.jquery.com/jquery-latest.js"></script>
</head>
<body>
<p>First Paragraph</p>
<p>Second Paragraph</p>
<p>Yet one more Paragraph</p>
<script>
$("p").click(function () { $(this).hide(); });
```




```
</script>
</body>
</html>
```

Use of SlideUp() Method:

```
<!DOCTYPE html>
<html>
<head>
<style>
p { color:red; margin:5px; cursor:pointer; }
p:hover { background:yellow; }
</style>
<script src="http://code.jquery.com/jquery-latest.js">
</script>
</head>
<body>
<p>First Paragraph</p>
<p>Second Paragraph</p>
<p>Yet one more Paragraph</p>
<script>
$("p").click(function () { $(this).slideUp(); });
</script>
</body>
</html>
```

Q.7.

Write jQuery program that changes the background color of a paragraph to red and font color to yellow when mouse enters over it. Also set the background color to white and font color to black when mouse leaves the paragraph.

ASKED YEAR →

CBSSGS: November – 2015

SOLUTION

PROGRAM:

```
<html>
<head>
<script src="scripts/jquery-1.4.1.js" type="text/JavaScript">
</script>
$(document).ready(function()
{
$("p").focus()
{
$(this).CSS("background-color:red");
$(this).CSS("color","yellow");
});
$("p").mouseover()
{
$(this).CSS("background-color:white");
$(this).CSS("color","black");
};
};
</script>
</head>
<body>
<form id="forms" runat="server">
<p>simple para-1</p>
</form>
</body>
</html>
```



Q.8.

Write a code that shows how to write a "LOCAL" cookie to a client's computer. The "LOCAL" cookie, stores:

- *FirstName*
- *LastName*

ASKED YEAR →

CBSGS: April – 2015

SOLUTION

PROGRAM:

STEP 1: Create the user layout to enter FirstName and LastName (i.e. WebForm).

STEP 2: Code behind (.cs code)

WriteCookieButton_Click event handler in the code behind file has the code required to write the cookie to the client computer as shown below:

```
protected void WriteCookieButton_Click(object sender, EventArgs e)
{
    // Create an instance of HttpCookie class
    HttpCookie LocalCookie = new HttpCookie("LOCAL");
    LocalCookie["FirstName"] = FirstNameTextBox.Text;
    LocalCookie["LastName"] = LastNameTextBox.Text;
    // Write the cookie to the client computer
    Response.Cookies.Add(LocalCookie);
}
```

Following is optional:

ReadCookieButton_Click event handler in the code behind may have code to read the cookie from the client computer as shown below:

```
protected void ReadCookieButton_Click(object sender, EventArgs e)
{
    // Check if the "LOCAL" cookie exists on the client computer
    if (Request.Cookies["LOCAL"] != null)
    {
        //Retrieve the "LOCAL" cookie into a cookie object
        HttpCookie LocalCookie = Request.Cookies["LOCAL"];
        //Write FirstName, LastName and LastVisit values
        Response.Write("First Name = " + LocalCookie["FirstName"] + "");
        Response.Write("Last Name = " + LocalCookie["LastName"] + "");
    }
}
```

Q.9.

Create a web page to read student's seat number, name add contact number using text boxes. Use appropriate validation controls to validate the following:

Student roll number should be from 1 to 120, name is compulsory and contact number must be of 10 digits.

Write HTML and code behind code

ASKED YEAR →

CBSGS: April – 2015

SOLUTION

PROGRAM:

```
<html>
<head>
</head>
<body>
<table>
<td>Roll Number</td>
<asp:TextBox runat="server">
</td>
```



```
<td>
<asp:RangeValidator runat="server" ControlToValidate="TextBox6" ErrorMessage="Enter the Roll
number between land 120" MaximumValue="120" MinimumValue="1" Type="Integer"
CssClass="errormsg">*</asp:RangeValidator>
</td>
</tr>
<td>Namer
<td>runat="server">
</td>
<td>
<asp:RequiredFieldValidator runat="server" ControlToValidate="TextBox1" ErrorMessage="First
Name cannot be blank" CssClass="errormsg">*</asp:RequiredFieldValidator>
</td>
<asp:RegularExpressionValidator runat="server" ControlToValidate="TextBox1" ErrorMessage="Wrong
entry" ValidationExpression="\d{10}">*</asp:RegularExpressionValidator>
<asp:Button runat="server" onClick="Button1_Click" Text="Button"/>
</tr>
</table>
</body>
</html>
```

Q.10.

Name, Address, ContactNo, City in SQL Server. Write code to display get all record from database into GridView control.

ASKED YEAR →

CBSGS: April – 2015

SOLUTION

PROGRAM:

```
SqlConnection conn = new SqlConnection();
conn.ConnectionString = (@"Data Source=ADMIN-PC\SQLEXPRESS;Initial
Catalog=Database_Name;Integrated Security=True");
string query = "SELECT * FROM StudentsInfo ";
try
{
conn.Open();
SqlCommand cmd = new SqlCommand(query, conn);
SqlDataAdapter da = new SqlDataAdapter();
da.SelectCommand = cmd;
DataSet myDS = new DataSet();
da.Fill(myDS, "Student");
GridView1.DataSource = myDS;
GridView1.DataBind();
}
catch (Exception ex)
{
lblMessage.Text = "Problem in connection or in database";
}
finally
{
conn.Close();
}
```

**Q.11.**

Create a web page to declare and initialize array of 10 integers and display the numbers greater than 30 using LINQ. Code should execute on button's click event in a web page.

ASKED YEAR →

CBSGS: April – 2015

SOLUTION**PROGRAM:**

```
Button1_Click()
{
    // Data source.
    int[] numbers = {90, 8, 71, 82, 93, 75, 82, 45, 9, 2};
    // Query Expression.
    IEnumerable<int> searchQuery = //query variable
    from numbers in numbers //required
    where score > 30 // optional
    select numbers; //must end with select or group
    // Execute the query to produce the results
    foreach (int n in searchQuery)
    {
        Console.WriteLine(n);
    }
}
```

Output:

```
90 71 82 93 75 45
```

Q.12.

Write a code to achieve overriding using virtual method. Use comments whenever necessary.

ASKED YEAR →

CBSGS: April – 2015

SOLUTION**PROGRAM:**

```
class Base
// Base Class
{
    public virtual void Display()
    //virtual method in base class
    {
        System.Console.WriteLine("Display - Base");
    }
}
class Derived : Base
{
    public override void Display()
    //derived class base class Display() is overridden here
    {
        System.Console.WriteLine("Display - Derived");
    }
}
class Demo
{
    public static void Main()
    {
        Base b;
        b = new Base ();
        b.Display();
        b = new Derived ();
        b.Display();
    }
}
```

**Output:**

```
Display - Base
Display - Derived
```

Q.13. Write program using Overloaded Constructors.

ASKED YEAR ⇒

CBSGS: November – 2014

SOLUTION**PROGRAM:**

```
using System;
class Room
{
public double length;
public double breadth;
public Room(double x, double y)
{
length=x;
breadth=y;
} public Room( double x)
{
length=breadth=x;
}
public void Area()
{
double a=length*breadth;
Console.WriteLine("Area is" + a);
}
class Mainclass
{
public static void Main(string[] args )
{
Room room1=new Room(25.0,15.0);
Room room2= new Room(20.0);
room1.Area();
room2.Area();
Console.ReadKey();
}
}
```

Q.14.

Write a program to create a new cookie with the name "Username" and add it to the HttpResponseMessage Object on the click of a button. Set the expiry date of the cookie to One year from Now.

ASKED YEAR ⇒

CBSGS: November – 2014

SOLUTION**PROGRAM:**

```
protected void Button1_Click(object sender, EventArgs e)
{
HttpCookie Username = new HttpCookie("UserName", "WELCOME");
Username.Expires=DateTime.Now.AddYears(1);
Response.Cookies.Add(Username);
}
```

**Q.15. Write a program using any five Methods / Property of ArrayList Class.**

ASKED YEAR →

CBSGS: November – 2014

SOLUTION**PROGRAM:**

```
using System;
using System.Collections;
using System.Text;
class Program
{
public static void Main( string[] args)
{
Arraylist n = new ArrayList();
n.Add("Mumbai");
n.Add("Pune");
n.Add("Kolkatta");
n.Add("Delhi");
n.Add("Chennai");
Console.WriteLine("ArrayList has capacity : " +n.Capacity);
Console.WriteLine("ArrayList has count : " + n.Count);
Console.WriteLine();
n.Sort();
for(int i=0; i<n.Count;i++)
{
Console.WriteLine(n[i]);
}
n.RemoveAt(3);
for(int i=0; i<n.Count;i++)
{
Console.WriteLine(n[i]);
}
Console.ReadKey();
}
}
```

Q.16.**Create a string array of names. Write a program with LINQ query to display all names from the array that contain the letter "S" and order them in ascending order. Display the query result in a Label.**

ASKED YEAR →

CBSGS: November – 2014

SOLUTION**PROGRAM:**

```
public partial class_Default: System.Web.UI.Page
{
protected void Page_Load(object sender, EventArgs e)
{
Label1.Text="";
//Creating Array
string[] StringArray=new string[5];
StringArray[0]="abc";
StringArray[1]="aac";
StringArray[2]="abd";
StringArray[3]="bad";
StringArray[4]="test";
//Select all String Starting with 'a' using LINQ
IEnumerable<string>result=from str in StringArray orderby str select str;
//Display result in List Box
foreach(string str in result)
{
Label1.Text+=str+"\n";
}
}
```



```
}  
}  
}
```

Q.17. Write a program using jQuery that hides a paragraph on Click of a Button.

ASKED YEAR →

CBSGS: November – 2014

SOLUTION

PROGRAM:

```
<html>  
<head>  
<title>jQuery that hides a paragraph on Click of a Button</title>  
<script src="Scripts/jquery-1.4.1.js" type="text/javascript"></script>  
<script type="text/javascript">  
$(document).ready(function ()  
{  
$("p").click(function ()  
{  
$(this).hide();  
});  
});  
</script>  
</head>  
<body>  
<p> ASP.NET with C# is simple </p>  
</body>  
</html>
```

Q.18.

Create a delegate with two init parameter and a return type. Create a class with two Delegate methods multiply and divide. Write a program to implement the Delegate.

ASKED YEAR →

CBSGS: November – 2014

SOLUTION

PROGRAM:

```
class Program  
{  
delegate double ProcessDelegate(double param1, double param2);  
static double Multiply(double param1, double param2)  
{  
return param1 * param2;  
}  
static double Divide(double param1, double param2)  
{  
return param1 / param2;  
}  
static void Main(string[] args)  
{  
ProcessDelegate process1= new ProcessDelegate(Multiply);  
ProcessDelegate process2= new ProcessDelegate(Divide);  
int r1=process1(8,2);  
int r2=process2(8,2);  
Console.WriteLine("Result1 = " + r1);  
Console.WriteLine("Result2 = " + r2);  
Console.ReadKey();  
}  
}
```



Q.19.

Write a C# program to do the following:

- Initialize an array A with 10 elements.
- Initialize an array B with 7 elements.
- Divide each element of array A with each element of array B that is $a[0] / b[0]$, $a[1] / b[1]$ etc.
- Implement the same to handle Divide by zero error and Index out of bound error.

ASKED YEAR →

CBSSGS: October – 2013

SOLUTION**PROGRAM:**

```
class Program
{
    static void Main(string[] args)
    {
        int[] A={1,2,3,4,5,6,7,8,9,10};
        int[] B={1,2,3,0,5,6,7};
        for (int i=0; i<10;i++)
        {
            try
            {
                float ans=A[i]/B[i];
                Console.WriteLine("The output is {0}", A[i]/B[i]);
            }
            catch(DivideByZeroException e)
            {
                Console.WriteLine(e.Message);
            }
            catch(IndexOutOfRangeException e)
            {
                Console.WriteLine(e.Message);
            }
        }
        Console.ReadKey();
    }
}
```


**Q.20.**

In a web form the details of an employee is filled in after he/she is employed in the company. In the design there is a Text Box to fill the First Name, a Text Box to fill the Last Name, a Text Box to fill the Surname, a text box to enter Email Address, a Text Box to enter the Email Address again to confirm the same, a text box to fill the age. What are the Validation Controls that have to be added for each field in the form?

ASKED YEAR →

CBSGS: October – 2013

SOLUTION**PROGRAM:**

<u>Controls</u>	<u>Validator</u>
First Name – text box	RequiredFieldValidator
Middle Name – text box	RequiredFieldValidator
Sir Name – text box	RequiredFieldValidator
Email Id – text box	RequiredFieldValidator RegularExpressionValidator
Email id confirm – text box	RequiredFieldValidator RegularExpressionValidator CompareValidator
Age – text box	RequiredFieldValidator RangeValidator

Example:

```
<table>
<tr>
<td runat="server">First Name</td>
<td>
<asp:RequiredFieldValidator runat="server" ControlToValidate="TextBox1" ErrorMessage="First
Name cannot be blank" CssClass="errormsg">*</asp:RequiredFieldValidator>
</td>
</tr>
<tr>
<td runat="server">Middle Name<td>
<td>
<asp:RequiredFieldValidator runat="server" ControlToValidate="TextBox2" ErrorMessage="Middle
Name cannot be Blank" CssClass="errormsg">*</asp:RequiredFieldValidator>
</td>
</tr>
<tr>
<td runat="server">Sir Name</td>
<td>
<asp:RequiredFieldValidator runat="server" ControlToValidate="TextBox3" ErrorMessage="Sir Name
cannot be Blank" CssClass="errormsg">*</asp:RequiredFieldValidator>
</td>
</tr>
<tr>
<td>E-mail Address
<td>
<asp:TextBox runat="server">
</td>
<td>
<asp:RequiredFieldValidator runat="server" ControlToValidate="TextBox4" ErrorMessage="E-mail
address cannot be Blank">*</asp:RequiredFieldValidator>
<asp:RegularExpressionValidator runat="server" ControlToValidate="TextBox4" ErrorMessage="Enter
the valid Email id" ValidationExpression="\w+([-+.']\w+)*@\w+([-.\w+([-
.\w+)]\w+)*">*</asp:RegularExpressionValidator>
</td>
```



```
</tr>
<tr>
<td>E-Mail Address again
<td>
<asp:TextBox runat="server">
</td>
<td>
<asp:RequiredFieldValidator runt="server" ControlToValidate="TextBox5" ErrorMessage="Email id
cannot be blank">*</asp:RequiredFieldValidator>
<asp:CompareValidator runat="server" ControlToCompare="TextBox4" ControlToValidate="TextBox5"
ErrorMessage="Wrong value">*</asp:CompareValidator>
</td>
</tr>
<tr>
<td>Age
<td>
<asp:TextBox runat="server">
</td>
<td>
<asp:RangeValidator runat="server" ControlToValidate="TextBox6" ErrorMessage="Please Enter the
age between 21 and 60" MaximumValue="60" MinimumValue="21" Type="Integer"
CssClass="errormsg">*</asp:RangeValidator>
<td>
</tr>
</table>
```

Q.21.**Write a program to illustrate Functional Overloading.**

ASKED YEAR →

IDOL: April – 2013

SOLUTION**PROGRAM:**

```
class abc
{
public void Disp(int a)
{
Console.WriteLine(a);
}
public void Disp(int a, int b)
{
Console.WriteLine("{0}{1}",a,b);
}
}
class program
{
static void Main(string[] args)
{
abc a1=newabc();
a1.Disp (10);
a1.Disp (10,20);
}
}
```

Output:

```
10
10 20
```

**Q.22.**

Create a window Form application in C# with a Listbox, a Textbox and three Buttons Add, Delete and clear. Add button will add the text from the Textbox to the Listbox. The delete button will remove the selected text from the Listbox and the clear button will clear the Listbox.

ASKED YEAR → IDOL: April – 2013**SOLUTION****PROGRAM:**

```
using System;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
public partial class _Default: System.Web.UI.Page
{
protected void btnAdd_Click(object sender, EventArgs e)
{
ListBox1.Items.Add(TextBox1.Text);
}
protected void btnDelete_Click(object sender, EventArgs e)
{
ListBox1.Items.Remove(ListBox1.SelectedItem);
}
protected void btnClear_Click(object sender, EventArgs e)
{
ListBox1.Items.Clear();
}
}
```

Q.23.

Create a Console Application in C# to handle an Event using Timer Object.

ASKED YEAR → IDOL: April – 2013**SOLUTION****PROGRAM:**

```
using System;
using System.Timers;
namespace aaa
{
class Program
{
static int counter=0;
static string displayString="This string will appear one letter at a time.";
static void Main(string[] args)
{
Timer myTimer=new Timer(100);
myTimer.Elapsed+=new ElapsedEventHandler(WriteChar);
myTimer.Start();
Console.ReadKey();
}
static void WriteChar(object source, ElapsedEventArgs e)
{
Console.Write(displayString[counter++% displayString.Length]);
}
}
}
```

**Q.24.**

Write a program that adds a cookie to the cookie collection of HttpResponse at the click event of a button. Also set the expiry date of the cookie to 2 days from now.

ASKED YEAR →

IDOL: April – 2013

SOLUTION**PROGRAM:**

```
protected void Button1_Click(object sender, EventArgs e)
{
    HttpCookie userCookie1;
    userCookie1=new HttpCookie("UserInfol");
    userCookie1["username"]=TextBox1.Text;
    userCookie1["password"]=TextBox2.Text;
    userCookie1.Expires=DateTime.Now.AddDays(2);
    Response.Cookies.Add(userCookie1);
    Label1.Text="Cookie create successful!<br>";
}
```

Q.25.

Write a program that Hides a paragraph when Mouse is Hovered Over it.

ASKED YEAR →

IDOL: April – 2013

SOLUTION**PROGRAM:**

```
<html>
<head>
<title>Mouse over Hover Effects</title>
<script src="https://code.jquery.com/jquery-git.js"></script>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width">
<title>Change the color of any paragraph to red on mouseover event.</title>
<style>
p {
color: blue;
font-size: 24px
}
</style>
<script>
$( "p" ).on( "mouseover", function() {
$( this ).css( "color", "red" );
});
</script>
</head>
<body>
<p>jQuery Exercises, Practice and Solution.</p>
<p>Ajax Tutorial</p>
</body>
</html>
```

**Q.26. Write a code to redirect the page to google.co.in when typed as google.com.**ASKED YEAR → **IDOL:** October – 2012**SOLUTION****PROGRAM:****STEP 1:** Create a New Web Application.**STEP 2:** Drag and Drop TextBox and Button Controls.**STEP 3:** Write the following Button_Click event as follows:

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (TextBox1.Text=="google.com")
    {
        Response.Redirect("http://www.google.co.in");
    }
}
```

Q.27. Write a jQuery Application to create an animation.ASKED YEAR → **IDOL:** October – 2012**SOLUTION****jQuery Application to create an Animation:**

- ⇒ The jQuery animate() method lets you create custom animations.
- ⇒ The jQuery animate() method is used to create custom animations.

Syntax:

```
$(selector).animate({params}, speed, callback);
```

- ⇒ The required params parameter defines the CSS properties to be animated.
- ⇒ The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.
- ⇒ The optional callback parameter is the name of a function to be executed after the animation completes.
- ⇒ The following example demonstrates a simple use of the animate() method; it moves a <div> element to the left, until it has reached a left property of 250px:

Example:

```
$("#button").click(function()
{
    $("#div").animate({left:'250px'});
});
$("#button").click(function()
{
    var div=$("#div");
    div.animate({height:'300px',opacity:'0.4'}, "slow");
    div.animate({width:'300px',opacity:'0.8'}, "slow");
    div.animate({height:'100px',opacity:'0.4'}, "slow");
    div.animate({width:'100px',opacity:'0.8'}, "slow");
});
```

OUR OTHER'S E-BOOK

B.Sc.IT: SEMESTER – V

PAPER – I: NETWORK SECURITY

Network Security HandBook
Network Security Paper Solution
Network Security Question Paper + Practical Question
Network Security Question Bank

PAPER – II: ASP.NET WITH C#

ASP.NET With C# HandBook
ASP.NET With C# Paper Solution
ASP.NET With C# Question Paper + Practical Question
ASP.NET With C# Question Bank

PAPER – III: SOFTWARE TESTING

Software Testing HandBook
Software Testing Paper Solution
Software Testing Question Paper + Practical Question
Software Testing Question Bank

PAPER – VI: ADVANCED JAVA

Advanced Java HandBook
Advanced Java Paper Solution
Advanced Java Question Paper + Practical Question
Advanced Java Question Bank

PAPER – V: LINUX ADMINISTRATION

Linux Administration HandBook
Linux Administration Paper Solution
Linux Administration Question Paper
Linux Administration Question Bank

B.Sc.IT: SEMESTER – VI

PAPER – I: INTERNET TECHNOLOGY

Internet Technology HandBook
Internet Technology Paper Solution
Internet Technology Question Paper + Practical Question
Internet Technology Question Bank

PAPER – II: PROJECT MANAGEMENT

Project Management HandBook
Project Management Paper Solution
Project Management Question Paper + Practical Question
Project Management Question Bank

PAPER – III: DATA WAREHOUSING

Data Warehousing HandBook
Data Warehousing Paper Solution
Data Warehousing Question Paper + Practical Question
Data Warehousing Question Bank

PAPER – VI: IPR AND CYBER LAWS (ELECTIVE)

IPR and Cyber Laws HandBook
IPR and Cyber Laws Paper Solution
IPR and Cyber Laws Question Paper + Practical Question
IPR and Cyber Laws Question Bank

PAPER – VI: DIGITAL SIGNAL AND SYSTEMS (ELECTIVE)

Digital Signal and Systems HandBook
Digital Signal and Systems Paper Solution
Digital Signal and Systems Question Paper
Digital Signal and Systems Question Bank

PAPER – VI: GEOGRAPHIC INFORMATION SYSTEM (ELECTIVE)

Geographic Information System HandBook
Geographic Information System Paper Solution
Geographic Information System Question Paper
Geographic Information System Question Bank



🌐 **Official Sites:** → www.mumbaibscitstudy.com
📘 **Facebook:** → <https://facebook.com/mumbaibscitstudy>
📷 **Instagram:** → <https://instagram.com/mumbaibscitstudy>
📺 **YouTube:** → <http://bit.do/KamalT>
👤 **Google+:** → <https://plus.google.com/+KamalTUniverse>
📄 **Blogger:** → <https://mumbaibscitstudy.blogspot.com/>
✉ **Gmail:** → kamalthakurbscit@gmail.com