BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)

ADVANCED WEB PROGRAMMING

CBCGS(NOV - 2019) Q.P.Code: 75427

Q1. a) What are .NET languages? Explain various features of C# languages. (5)

The Microsoft .Net Framework provides the necessary foundation for developing windows as well as web applications. The unique feature of .NET is the multi-language support that it provides. To help create languages for the .NET Framework, Microsoft created the Common Language Infrastructure specification (CLI). The CLI describes the features that each language must provide in order to use the .NET Framework and common language runtime. The .NET platform supports many languages and the list is growing day by day. The C# compiler is considered to be the most efficient compiler in the .NET family

C# is object oriented programming language. It provides a lot of features that are given below:-

1. **Simple :-** C# is a simple language in the sense that it provides structured approach, rich set of library functions, data types etc. Unsafe operations such as direct memory manipulation are not allowed.
2. **Modern :-** C# programming is based upon the current trend and it is very powerful and simple for building scalable, interoperable and robust applications. C# programming is based upon the current trend and it is very powerful and simple for building scalable, interoperable and robust applications.
3. **Object Oriented Programming :-** C# is object oriented programming language i.e it supports data encapsulation, inheritance, polymorphism, interfaces. OOP's makes development and maintenance easier where as in Procedure-oriented programming language it is not easy to manage if code grows as project size grow.
4. **Type Safe :-** C# type safe code can only access the memory location that it has permission to execute. Therefore it improves a security of the program.
5. **Scalable and Updatable :-** C# is automatic scalable and updateable programming language. For updating our application we delete the old files and update them with new ones.
6. **Component Oriented :-** C# is component oriented programming language. It is the predominant software development methodology used to develop more robust and highly scalable applications.

Q1. b) What is property? Explain read-write property with proper syntax and example. (5)

Properties are the special type of class members that provides a flexible mechanism to read, write, or compute the value of a private field. Properties are named members of classes, structures, and interfaces. Member variables or methods in a class or structures are called Fields. Properties are an extension of fields and are accessed using the same syntax. Properties can be used as if they are public data members, but they are actually special methods called Accessors They use accessors through which the values of the private fields can be read, written or manipulated. Properties do not name the storage locations. Instead,

MUQuestionPapers.com

they have accessors that read, write, or compute their values. A property is similar to a variable, it's a way to access a piece of data that's associated with an object. It uses pre-defined methods which are "get" and "set" methods which help to access and modify the properties.

read-write property :- read-write means that both a get and a set method exist; the opposite is read-only. Normally the only time we had explicitly declare a property read-write is in a class extension for a class where the public interface declares the property read-only. So that it's publicly read-only, but internally we can both get and set.

Syntax :-

```
<access_modifier> <return_type> <property_name>
{
    get { // body }
    set { // body }
}
```

Example – This sample shows a Person class that has two properties: Name (string) and Age (int). Both properties provide get and set accessors, so they are considered read/write properties.

```
class Person
{  private string _name = "N/A";
   private int _age = 0;
   public string Name
   {
      get
      {
         return _name;
      }
      set
      {
         _name = value;
      }
   }
   public int Age
   {
      get
      {
         return _age;
      }

      set
      {
         _age = value;
      }
   }

   public override string ToString()
   {
      return "Name = " + Name + ", Age = " + Age;
   }
}
```

MUQuestionPapers.com

```
class TestPerson
{
   static void Main()
   {
      Person person = new Person();
      Console.WriteLine("Person details - {0}", person);
      person.Name = "Joe";
      person.Age = 99;
      Console.WriteLine("Person details - {0}", person);
      person.Age += 1;
      Console.WriteLine("Person details - {0}", person);
      Console.WriteLine("Press any key to exit.");
      Console.ReadKey();
   }
}
```

Output:

```
Person details - Name = N/A, Age = 0
Person details - Name = Joe, Age = 99
Person details - Name = Joe, Age = 100
```

Q1. c) Explain static constructor and copy constructor with example. (5)

Static Constructor :- A static constructor is used to initialize static variables of a class. It is declared using the static keyword. A class can have only one static constructor.

Syntax of a static constructor :-

```
static classname()
{
static member initialization;
}
```

A static constructor is parameter less and no access modifiers are allowed with its declaration. The exact timing of static constructor execution is implementation-dependent, but is subject to the following rules:-

- The static constructor for a class executes before any instance of the class is created.
- The static constructor for a class executes before any of the static members for the class are referenced.
- The static constructor for a class executes after the static fields initializes for the class.
- The static constructor for a class executes at most one time during a single program execution.

Example :-

```
class A
{
static A()
{
Console.WriteLine("static constructor A is called");
```


MUQuestionPapers.com

```
}
public static void Main() { }
}
```

Copy Constructor :- A copy constructor copies the data from one object to another. In C#, there is no built in support for copy constructor. So we must create a user defined copy constructor if we wish to add this feature to a class.

The following example illustrate the use of a copy constructor:

```
class complex_number
{
private int real, img;
        public complex_number()
        {
                real=0; img=0;
        }
        public complex_number(complex_number c)                    //copy constructor
        {
                real=c.real;
                img=c.img;
        }
        public void disp()
        {
                System.Console.WriteLine(real);
                System.Console.WriteLine(img);
        }
        public static void Main()
        {
                complex_number c1=new complex_number(2,3);
                complex_number c2=new complex_number(c1);
                c2.disp();
        }
}
```

The above constructor is invoked by instantiating an object of type complex_number and passing it the object to be copied.

Q1. d) Explain one dimensional and two dimensional arrays with proper syntax and example. (5)

- **Creating a one-dimensional array :-** A one-dimensional array is used to store a list of values.

The general syntax for declaring a one-dimensional array is :

```
datatype[ ] arrayname=new datatype[number of elements];
```

Example :-

int[] x=new int[5];

Initializing one-dimensional arrays :-

Syntax :- arrayname[index]=value;

Example :-

int[] x=new int[3]; x[0]=100; x[1]=23; x[2]=440;

An array can be initialized at the time of its declaration in the following manner.
int[] x={100,23,440};
or
int[] x=new int[3]{100,23,440};
or
int[] x=new int[]{100,23,440};

- **Creating a two-dimensional array :-** The two-dimensional arrays are used to store table of values.

The general syntax for declaring a two-dimensional array is :

datatype[,] arrayname=new datatype[row size, column size];

Example :- int[,] x=new int[5,4];

The above statement creates an array of 20 cells (Similar to a table of five rows and four columns).

Initializing two-dimensional arrays :-

Syntax :- arrayname[rowindex, colindex]=value;

Example :- int[,] x=new int[3, 2]; x[0,0]=100; x[0,1]=23;

The initialization of a two-dimensional array at the time of declaration is done in the following manner.
int[,] x={{1,2},{2,4},{5,6}};
or
int[,] x=new int[2,3]{ {1,2},{2,4},{5,6}};

Q1. e) Define inheritance. Explain single inheritance with example. (5)

In C#, inheritance is a process in which one object acquires all the properties and behaviours of its parent object automatically. In such way, we can reuse, extend or modify the attributes and behaviours which is defined in other class. In C#, the class which inherits the members of another class is called derived class and the class whose members are inherited is called base class. The derived class is the specialized class for the base class.

MUQuestionPapers.com

Single Inheritance :- In single inheritance, subclasses inherit the features of one superclass. It is the simplest of all inheritance. In this type of inheritance there is only one base class and one 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 -

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

- **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 of single inheritance -

```
using System;
  public class Animal
   {
     public void eat() { Console.WriteLine("Eating..."); }
   }
  public class Dog: Animal
  {
     public void bark() { Console.WriteLine("Barking..."); }
  }
  class TestInheritance{
     public static void Main(string[] args)
      {
         Dog d = new Dog();
         d.eat();
         d.bark();
      }
   }
```

Output :

```
Eating...
Barking...
```

Q1. f) List various reference types in c#. Also explain boxing operation with example. (5)

The reference type hold references to the values stored somewhere in memory. The reference types do not contain the actual data stored in a variable, but they contain a reference to the variables. Using multiple variables, the reference types can refer to a memory location. If the

MUQuestionPapers.com

data in the memory location is changed by one of the variables, the other variable automatically reflects this change in value.

List of reference types :-

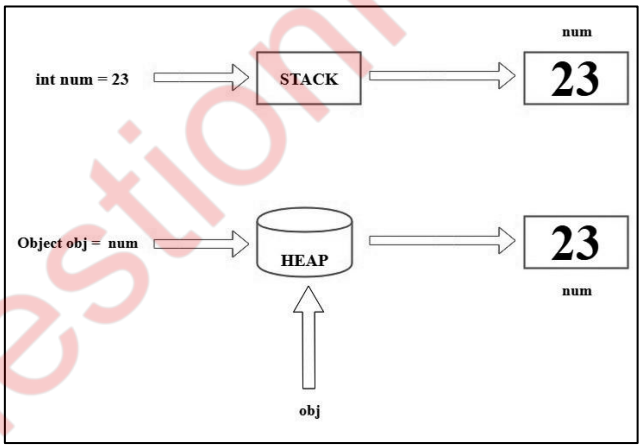
- interface type
- delegate type
- array type
- class type
- strings

Boxing Operation :- The process of Converting a Value Type (char, int etc.) to a Reference Type(object) is called Boxing. Boxing is implicit conversion process in which object type (super type) is used. The Value type is always stored in Stack. The Referenced Type is stored in Heap.

Example :

```
int num = 23;                    // 23 will assigned to num
Object Obj = num;                // Boxing
```

First declare a value type variable (num), which is integer type and assigned it with value 23. Now create a references object type (obj) and applied Explicit operation which results in num value type to be copied and stored in object reference type obj as shown in below figure :



Example of Boxing with a C# programming code :

```
using System;
class GFG {
   static public void Main()                          // Main Method
   {
      int num = 2020;                                 // assigned int value 2020 to num
      object obj = num;                                  // boxing
      num = 100;                                         // value of num to be change
      System.Console.WriteLine("Value - type value of num is : {0}", num);
      System.Console.WriteLine("Object - type value of obj is : {0}", obj);
   }
}
```

MUQuestionPapers.com

Output:

Value - type value of num is : 100
Object - type value of obj is : 2020

Q2. a) What are .aspx files? Explain code behind class file in detail. (5)

- An ASPX file is an Active Server Page Extended (ASPX) file, which is a webpage generated by web servers running the Microsoft ASP.NET framework.
- It contains one or more scripts written in VBScript or C# code that are processed by the web server into HTML, which is sent to the user's web browser.
- ASPX pages are also called ".NET Web forms".
- The program logic is written in a separate file called code behind file.
- The code behind file has the extension .aspx.cs. There is a code behind file associated with each page of a website.
- As the program logic is separated from user interface design code, this model is easy to understand.
- The code for the page is compiled into a separate class from which the .aspx file derives. Only server controls can interact with the code behind the page; not the HTML controls.
- In the code-behind file, we create a class (which can be any class derived from the Page class) that serves as the base class for the web page we create in the .aspx file.
- This relationship between our class and the web page is established by a Page directive at the top of the .aspx file.
- When users point their browsers to the .ASPX page, the code-behind class file runs and dynamically produces the output for the page. When the page is requested, the code-behind file is loaded and treated as an integral part of the page.
- We enable the ASP.NET code-behind feature by adding a couple of attributes, namely Inherits and Src, to the @Page directive -
  <%@ Page Language="C#" Inherits="MyBasePageClass" Src="MyBasePage.cs" %>
- The Inherits attribute contains the name of the code-behind class from which the current page inherits.
- The code-behind class can be any class derived from the Page class. The Src attribute represents the URL of the source file defining the class.
- When the ASP.NET HTTP run time processes a page, it first analyses the @Page directive.
- If Src is present, the code-behind file is loaded and compiled. Then the run time uses the contents of Inherits to get the name of the class to instantiate so that it can serve the ongoing page request.

Q2. b) Explain each of the following in brief: (5)

i. **Web forms**
ii. **Post back**
iii. **Page rendering**
iv. **Page Load event**
v. **Page PreRender event**

i) **Web forms :-** The Web Forms are web pages built on the ASP.NET Technology. Web Forms are pages that users request using their browser. These pages can be written using a combination of HTML, client-script, server controls, and server code.

ii) **Post back :-** The process of submitting an ASP.NET page to the server for processing is termed as post back. The post back originates from the client-side browser. The Page object has an IsPostBack property, which is tells whether the request is a fresh request or a post back.

iii) **Page rendering :-** At this stage, view state for the page and all controls are saved. The page calls the Render method for each control and the output of rendering is written to the OutputStream class of the Response property of page.

iv) **Page Load event :-** Load is raised for page first and recursively for all child controls. In the event handler associated with this event you can write code that needs to be executed for each postback of the web page.

v) **Page PreRender event :-** 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.

Q2. c) List any four category of server controls in asp.net? Explain common properties of web server controls. (5)

Category of server controls :-

- Standard controls
- Data controls
- Validation controls
- Navigation controls

Properties of web server controls :-

1. AccessKey :- Enables you to set a key with which a control can be accessed at the client by pressing the associated letter.
2. BackColor, ForeColor :- Enables you to change background color and text color of a control.
3. BorderColor :- This property is used to change the border color of a control.
4. BorderStyle :- Using this property border Style can be set to none, dotted, dashed, solid double, groove etc.
5. BorderWidth :- Enables you to change border width of a control.
6. Enabled :- Determines whether the control is enabled or not. If the control is disabled user cannot interact with it.
7. Font :- Enables you to change the Font settings.
8. Height :- Enables you to set the height of the control.
9. Width :- Enables you to set the width of the control.
10. Visible :- Determines whether the control is visible or not.

MUQuestionPapers.com

Q2. d) Explain the basic functionality of each of the following webserver controls. (5)

i. **CheckBox**
ii. **TextBox**
iii. **DropDownList**
iv. **Menu**

i) **CheckBox :-** A CheckBox displays a single option that the user can either check or uncheck. If we want check box or radio button to be selected when the form is initially displayed, set its Checked attribute to true.

Example –

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

ii) **TextBox :-** TextBox controls are typically used to accept input from the user. A text box control can accept one or more lines of text depending upon the settings of the TextMode attribute.

Example –

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

iii) **DropDownList :** The DropDownList is a web server control which is used to create an HTML Select component. It allows us to select an option from the dropdown list. It can contain any number of items.

Example –

```
<asp:DropDownList id="DropDownList1" runat="server"
    DataSource="<% databindingexpression %>"
    DataTextField="DataSourceField"
    DataValueField="DataSourceField"
    AutoPostBack="True|False"
    OnSelectedIndexChanged="OnSelectedIndexChangedMethod">
  <asp:ListItem value="value" selected="True|False">
    Text
  </asp:ListItem>
</asp:DropDownList>
```

iv) **Menu :-** The ASP.NET Menu control allows us to develop both statically and dynamically displayed menus for our ASP.NET Web pages. 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.

Example –

```
<asp:Menu ID="Menu1" runat="server"></asp:Menu>
```

MUQuestionPapers.com

Q2. e) Write a brief note on various validator controls. (5)

They are web server controls used for the validation of form fields. These controls are included in System.Web.UI.WebControls namespace. They can perform both server-side and client-side validation.

1) **RequiredFieldValidator :-** The RequiredFieldValidator control ensures that the required field is not empty. It is generally tied to a text box to force input into the text box.
2) **RangeValidator Control :-** The RangeValidator control is used to check whether the user enters an input value that is within a specified range. Applicable for range of numbers, dates, and characters.
3) **CompareValidator Control :-** The CompareValidator control compares a value in one control with a fixed value or a value in another control.
4) **RegularExpressionValidator :-** The RegularExpressionValidator allows validating the input text by matching against a pattern of a regular expression. Patterns consists of literal characters and special characters
5) **CustomValidator** :- The CustomValidator control allows writing application specific custom validation routines for both the client side and the server side validation.
6) **ValidationSummary** :- The ValidationSummary control does not perform any validation but shows a summary of all errors in the page. The summary displays the values of the ErrorMessage property of all validation controls that failed validation.

Q2. f) What are rich controls? Explain about Calendar and AdRotator controls in brief. (5)

Rich controls are web controls that model complex user interface elements. A typical rich control can be programmed as a single object but renders itself using a complex sequence of HTML elements. Rich controls can also react to user actions and raise more-meaningful events that our code can respond to on the web server. In other words, rich controls give we a way to create advanced user interfaces in our web pages without writing lines of convoluted HTML.

The Calendar Control :- The Calendar control presents a miniature calendar that we can place in any web page. The Calendar can be programmed as a single object, but it renders itself with dozens of lines of HTML output.

Example –

```
<asp:Calendar id="MyCalendar" runat="server" />
```

The Calendar control presents a single-month view. The user can navigate from month to month by using the navigational arrows, at which point the page is posted back and ASP.NET automatically provides a new page with the correct month values.

MUQuestionPapers.com

❖ **Properties for Calendar Control :-**

a) DayHeaderStyle :- The style for the section of the Calendar that displays the days of the week (as column headers).
b) DayStyle:- The default style for the dates in the current month.
c) NextPrevStyle :- The style for the navigation controls in the title section that move from month to month.
d) OtherMonthDayStyle :- The style for the dates that aren't in the currently displayed month. These dates are used to "fill in" the calendar grid. For example, the first few cells in the topmost row may display the last few days from the previous month.
e) SelectedDayStyle :- The style for the selected dates on the calendar.
f) TodayDayStyle :- The style for the date designated as today (represented by the TodaysDate property of the Calendar control).
g) WeekendDayStyle :- The style for dates that fall on the weekend.

The AdRotator Control :- The basic purpose of the AdRotator is to provide a graphic on a page that is chosen randomly from a group of possible images. In other words, every time the page is requested, an image is selected at random and displayed, which is the rotation indicated by the name AdRotator. One use of the AdRotator is to show banner-style advertisements on a page, but we can use it anytime we want to vary an image randomly.

The Advertisement File(.xml file) -

```
<Advertisements>
    <Ad>
    <ImageUrl>prosetech.jpg</ImageUrl>
    <NavigateUrl>http://www.prosetech.com</NavigateUrl>
    <AlternateText>ProseTech Site</AlternateText>
    <Impressions>1</Impressions>
    <Keyword>Computer</Keyword>
    </Ad>
</Advertisements>
```

❖ **Advertisement File Elements :-**

a) ImageUrl :- The image that will be displayed. This can be a relative or a fully qualified Internet URL.
b) NavigateUrl :- The link that will be followed if the user clicks the banner.
c) AlternateText :- The text that will be displayed instead of the picture if it cannot be displayed.
d) Impressions :- A number that sets how often an advertisement will appear. This number is relative to the numbers specified for other ads.

MUQuestionPapers.com

e) Keyword :- A keyword that identifies a group of advertisements. We can use this for filtering.

Q3. a) Explain exception handling mechanism with proper syntax and example. (5)

- An exception is an event that interrupts normal program execution.
- In other words, when an error occurs in our application, the .NET Framework creates an exception object that represents the problem.
- We can catch this object by using an exception handler. If we fail to use an exception handler, our code will be aborted, and the user will see an error page.
- If we catch the exception, we can notify the user, attempt to resolve the problem, or simply ignore the issue and allow our web page code to keep running.
- There are two aspects involved in the processing of exceptions: raising (or throwing) the exception and handling the exception.
- The steps involved in exception handling are:
    a. Detect problem-causing exception (hit the exception).
    b. Inform that an error has occurred (throw the exception).
    c. Receive error information (catch the exception).
    d. Take corrective actions (handle the exception).
- In C#, the try and catch block used for handling exception has the following syntax :-

1) Try with multiple catch block :-

```
try {
        statements; }
catch(ExceptionType obj) {
        statements; }
catch(ExceptionType obj) {
        statements; }
```

2) Try block with catch and finally block :-

```
try {
        statements; }
catch(ExceptionType obj) {
        statements; }
finally {
        statements; }
```

3) Try block with only finally block :-

```
try {
        statements; }
finally {
        statements; }
```

- A try block must have a catch block or a finally block.
- There can be multiple catch blocks but a single finally block for a try block.
- The normal scope rules apply for the try block. For instance any local variable defined in the try block will not be accessible outside the block.

Eaxmple of exception handling –

```
try {
        int n=Convert.ToInt32(Console.ReadLine());
        System.Console.WriteLine("n="+n);
    }
catch(FormatException e) { }
```

Q3. b) Describe session state variables in asp.net. (5)

- Session-state management is one of ASP.NET's premiere features.
- It allows us to store any type of data in memory on the server.
- The information is protected, because it is never transmitted to the client, and it's uniquely bound to a specific session.
- Session state is ideal for storing information such as the items in the current user's shopping basket when the user browses from one page to another.
- ASP.NET tracks each session by using a unique identifier.
- ASP.NET uses a proprietary algorithm to generate this value, thereby guaranteeing (statistically speaking) that the number is unique and it's random.
- When the client presents the session ID, ASP.NET looks up the corresponding session, retrieves the objects we stored previously, and places them into a special collection so they can be accessed in our code.
- The session state object is created from the HttpSessionState class, which defines a collection of session state items.
- We can interact with session state by using the System.Web.SessionState.HttpSessionState class, which is provided in an ASP.NET web page as the built-in Session object.
- The syntax for adding items to the collection and retrieving them is basically the same as for adding items to a page's view state.
- **Adding session state variables :-**

```
Session["session_variable_name"]=value;
```

Example :-

```
Session["x"] ="Tata";
Session["y"] = "atat";
```

The session variable can also be added using add() method.

Example :-

```
string s1 = TextBox1.Text;
Session.Add("z", s1);
```

- **Retrieving Session Variables :-**

```
String variable=Session["session_variable_name"].ToString();
```

Example :-

```
string x = Session["z"].ToString();
```

Q3. c) Explain state management through persistent cookies. (5)

- A cookie is a technique used for client side state management.
- It helps Web applications to store user-specific information.
- It is a text file stored on the client side by the browser. These files store name value pairs.
- One advantage of cookies is that they work transparently, without the user being aware that information needs to be stored.
- A Web application can read the information stored on the cookie whenever the user visits the site.
- Cookies are mainly classified in the following two types -
  i) Persistent :- As the name itself suggests, these cookies remain persistent in the client's memory even after the browser is closed. They remain permanently in memory until they are explicitly removed or their expiration is reached.
  ii) Non-persistent :- These cookies do not remain in the client's memory and are lost after the browser is closed.
- Before we can use cookies, we should import the System.Net namespace so we can easily work with the appropriate types:

```
using System.Net;
```

- Cookies are fairly easy to use. Both the Request and Response objects (which are provided through Page properties) provide a Cookies collection.

❖ **Creating and Reading Cookies using Request and Response Objects :-**

Syntax for Creating a Cookie :-

```
Response.Cookies[“cookie name”].Value=“specify value”;
```

Example :-

```
Response.Cookies["uname"].Value = TextBox1.Text;
```

Syntax for Reading a Cookie :-

```
String str=Request.Cookies[“cookie name”].Vaue;
```

Example :-

```
string str= Request.Cookies["pwd"].Value;
```

Example of Cookies –

```
public void SetPersistentCookies(string name, string value)
{
   HttpCookie cookie = new HttpCookie(name);
   cookie.Value = value;
   cookie.Expires = Convert.ToDateTime("12/12/2008″);
   Response.Cookies.Add(cookie);
}
```

Q3. d) Describe the features and benefits of master pages in asp.net. (5)

- Master page provides a framework (common content as well as the layout) within which the content from other pages can be displayed.
- Master pages are similar to ordinary ASP.NET pages.
- Master pages are text files that can contain HTML, web controls, and code.
- However, master pages have a different file extension i.e .master and they can't be viewed directly by a browser.
- It provides elements such as headers, footers, style definitions, or navigation bars that are common to all pages in our web site.
- Instead, master pages must be used by other pages, which are known as content pages.
- When users request the content page, ASP.NET merges the layout of the master page with the content of the content page and produce output.
- So the Content Pages need not have to duplicate code for shared elements within our Web site. It gives a consistent look and feel for all pages in our application.
- The master page layout consist of regions where the content from each content page should be displayed.
- These regions can be set using ContentPlaceHolder server controls.
- These are the regions where we are going to have dynamic content in our page.
- A derived page also known as a content page is simply a collection of blocks the runtime will use to fill the regions in the master.

❖ **Advantages of Master Page :-**

- ➢ It is very easy to implement.
- ➢ They allow us to centralize the common functionality of our pages so that we can make updates in just one place.
- ➢ Master pages enable consistent and standardized layout of the website.
- ➢ They make it easy to create one set of controls and code and apply the results to a set of pages. For example, we can use controls on the master page to create a menu that applies to all pages.
- ➢ They give us fine-grained control over the layout of the final page by allowing us to control how the placeholder controls are rendered.
- ➢ They provide an object model that allows us to customize the master page from individual content pages.

MUQuestionPapers.com

Q3. e) What is a theme? How does it work? **(5)**

- A theme decides the look and feel of the website. It is a collection of files that define the looks of a page. It can include skin files, CSS files & images.
- One of the neat features of ASP.NET is themes, which enable us to define the appearance of a set of controls once and apply the appearance to our entire web application.
- Themes make it simple to customize the appearance of a site or page using the same design tools and methods used when developing the page itself, thus obviating the need to learn any special tools or techniques to add and apply themes to a site.
- We can apply themes to controls, pages, and even entire sites. We can leverage this feature to customize parts of a web site while retaining the identity of the other parts of the site.
- Themes allow all visual properties to be customized, thus ensuring that when themed, pages and controls can achieve a consistent style.
- All themes are application specific. To use a theme in a web application, we need to create a folder that defines it.
- This folder needs to be placed in the App_Themes folder, which must be placed inside the top-level directory for our web application.
- An application can contain definitions for multiple themes as long as each theme is in a separate folder.
- Only one theme can be active on a given page at a time.
- To actually make our theme accomplish anything, we need to create at least one skin file in the theme folder.
- A skin file is a text file with the .skin extension. ASP.NET never serves skin files directly; instead, they're used behind the scenes to define a theme.
- A skin file is a list of control tags. The control tags in a skin file don't need to completely define the control; they need only set the properties that we want to use.
- When we add a control tag for the ListBox in the skin file, it might look like following markup: <asp:ListBox runat = "server" ForeColor = "White" BackColor = "Orange"/>
- The runat = "server" portion is always required. Everything else is optional. We should avoid setting the ID attribute in our skin file because the page that contains the ListBox needs to define a unique name for the control in the actual web page.

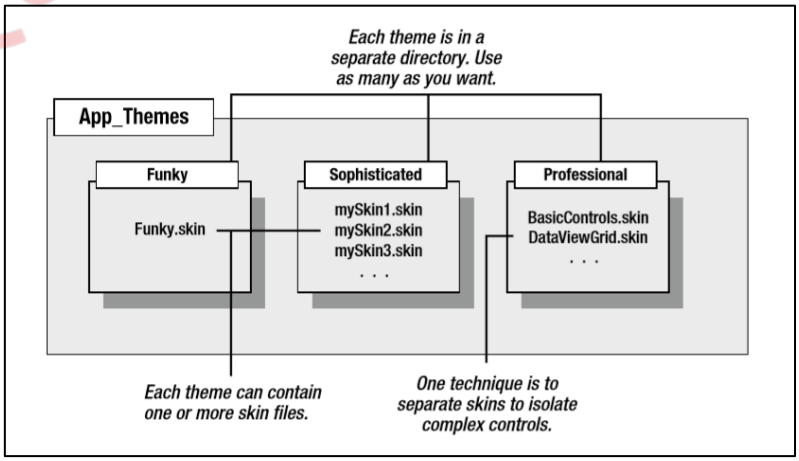


Figure - Themes and skin

MUQuestionPapers.com

Q3. f) What are the three different ways to use styles on web pages? Explain various category of style setting in New Style dialog box. (5)

Web pages can use styles in three different ways :-

1) **Inline style :-** An inline style is a style that's placed directly inside an HTML tag. This can get messy, but it's a reasonable approach for one-time formatting. We can remove the style and put it in a style sheet later.
2) **Internal style sheet :-** An internal style sheet is a collection of styles that are placed in the <head> section of our web page markup. We can then use the styles from this style sheet to format the web controls on that page.
3) **External style sheet :-** An external style sheet is similar to an internal style sheet, except it's placed in a completely separate file. This is the most powerful approach because it gives us a way to apply the same style rules to many pages.

Various category of style setting in New Style dialog box :-

1. Font :- It allows us to choose the font family, font size, and text color, and apply other font characteristics.
2. Block :- It allows us to fine-tune additional text settings, such as the height of lines in a paragraph, the way text is aligned, the amount of indent in the first list, and the amount of spacing between letters and words.
3. Background :- It allows us to set a background color or image.
4. Border :- It allows us to define borders on one or more edges of the element. We can specify the border style, thickness, and color of each edge.
5. Box :- It allows us to define the margin and the padding.
6. Position :- It allows us to set a fixed width and height for our element, and use absolute positioning to place our element at a specific position on the page.

Q4. a) What is data binding? Explain repeated value data binding with example. (5)

Data binding is the process how the user interface controls of a client application are configured to retrieve from, or update data into, a data source. It provides a simple and consistent way for applications to present and interact with data. If the user modifies the data in a data-bound control, our program can update the corresponding record in the database. ASP.NET data binding works in one direction only. Information moves from a data object into a control. ASP.NET data binding is much more flexible than old-style data binding.

Repeated value data binding :-

- Repeated-value data binding works with the ASP.NET list controls.
- Repeated-value data binding allows us to display an entire table.
- To use repeated-value binding, we link one of these controls to a data source (such as a field in a data table).
- When we call DataBind(), the control automatically creates a full list by using all the corresponding values.

- With repeated-value data binding, we can write a data-binding expression in our .aspx file, or we can apply the data binding by setting control properties.
- In some ways, data binding to a list control is the simplest kind of data binding. We need to follow only three steps:
  1. Create and fill some kind of data object - We have numerous options, including an array, the basic ArrayList and Hashtable collections, the strongly typed List and Dictionary collections, and the DataTable and DataSet objects.
  2. Link the object to the appropriate control - To do this, we need to set only a couple of properties, including DataSource. If we're binding to a full DataSet, we'll also need to set the DataMember property to identify the appropriate table we want to use.
  3. Activate the binding - As with single-value binding, we activate data binding by using the DataBind() method, either for the specific control or for all contained controls at once by using the DataBind() method for the current page.

Example –

```
public List<string> GetCarsList()
{
        List<string> cars = new List<string>();
        cars.Add("Santro");
        cars.Add("Marazzo");
        cars.Add("i20");
         cars.Add("Ertiga");
        cars.Add("Brezza");
        cars.Add("Tigor");
        cars.Add("Creta");
        return cars;
}
protected void Page_Load(object sender, EventArgs e)
{
        lstItems.DataSource=GetCarsList();
        this.DataBind();
}
```

Q4. b) Write any three similarities between FormView and DetailsView. Explain about item templates of FormView. (5)

Similarities between formview and detailsview :-

1) Like DetailsView, FormView also displays a single record from the data source at a time.
2) Both show a single record at a time but can include optional pager buttons that let us step through a series of records (showing one per page).
3) Both controls can be used to display, edit, insert and delete database records but one at a time. And both support templates, but the FormView requires them.

MUQuestionPapers.com

4) Both have paging feature and hence support backward and forward traversal.

ItemTemplate of FormView :-

- ItemTemplate is used to display the data from data the source in ReadOnly mode.
- The controls included in an ItemTemplate are controls that only display data, such as a Label control.
- The template can also contain command buttons to change the FormView control's mode to insert or edit, or to delete the current record.
- The ItemTemplate template can include Link Button controls that change the mode of the FormView control based on the Command Name value.
- A CommandName value of New puts the record into insert mode and loads the InsertItemTemplate, which allows the user to enter values for a new record.
- We can add a button with a CommandName value of Edit to the ItemTemplate template to put the FormView control into edit mode.
- A button with a CommandName value of Delete can be added to the ItemTemplate template to allow users to delete the current record from the data source.

Q4. c) Explain data binding to a Dictionary collection. (5)

- A dictionary collection is a special kind of collection in which every item is indexed with a specific key (or dictionary word).
- This is similar to the way that built-in ASP.NET collections such as Session, Application, and Cache work.
- Dictionary collections always need keys. This makes it easier to retrieve the item we want.
- In ordinary collections, such as the List, we need to find the item we want by its index number position, or more often by traveling through the whole collection until we come across the right item.
- With a dictionary collection, we retrieve the item we want by using its key.
- Generally, ordinary collections make sense when we need to work with all the items at once, while dictionary collections make sense when we frequently retrieve a single specific item.
- We can use two basic dictionary-style collections in .NET.
- The Hashtable collection allows us to store any type of object and use any type of object for the key values.
- The Dictionary collection uses generics to provide the same "locking in" behavior as the List collection.
- We choose the item type and the key type upfront to prevent errors and reduce the amount of casting code we need to write.
- Example –

```
protected void Page_Load(Object sender, EventArgs e)
{
    if (!this.IsPostBack)
```

```
    {
        // Use integers to index each item. Each item is a string.
        Dictionary<int, string> fruit = new Dictionary<int, string>();
        fruit.Add(1, "Kiwi");
        fruit.Add(2, "Pear");
        fruit.Add(3, "Mango");
        fruit.Add(4, "Blueberry");
        fruit.Add(5, "Apricot");
        fruit.Add(6, "Banana");
        fruit.Add(7, "Peach");
        fruit.Add(8, "Plum");
        MyListBox.DataSource = fruit;          // Define the binding for the list controls.
        MyListBox.DataTextField = "Value";
        this.DataBind();                                    // Activate the binding.
    }
}
```

Q4. d) Explain various style properties of GridView control. (5)

- HeaderStyle :- It configures the appearance of the header row that contains column titles, if we've chosen to show it (if ShowHeader is true).
- RowStyle :- It configures the appearance of every data row. AlternatingRowStyle If set, applies additional formatting to every other row. This formatting acts in addition to the RowStyle formatting.
- SelectedRowStyle :- It configures the appearance of the row that's currently selected. This formatting acts in addition to the RowStyle formatting.
- EditRowStyle :- It configures the appearance of the row that's in edit mode. This formatting acts in addition to the RowStyle formatting.
- EmptyDataRowStyle :- It configures the style that's used for the single empty row in the special case where the bound data object contains no rows.
- FooterStyle :- It configures the appearance of the footer row at the bottom of the GridView, if we've chosen to show it (if ShowFooter is true).
- PagerStyle :- It configures the appearance of the row with the page links, if we've enabled paging (set AllowPaging to true).
- BorderStyle :- It configures the appearance of border style of the Web server control.
- AlternatingRowStyle :- It configures the appearance of TableItemStyle object that enables us to set the appearance of alternating data rows in a GridView control.
- SelectedValue :- Gets the data key value of the selected row in a GridView control.

Q4. e) Briefly explain following features of GridView control. (5)

i. **Sorting**
ii. **Paging**
iii. **Selecting a row**

iv. Editing with the GridView

i) **Sorting :-** The GridView sorting features allow the user to reorder the results in the GridView by clicking a column header. It's convenient and easy to implement. To sort GridView data, following are the steps :
   - Set AllowSorting attribute of GridView to True.
   - Also set SortExpression property of columns to respective field from database to sort.

ii) **Paging :-** Paging refers to the ability of GridView control to display bound data one page at a time.The users can arbitrarily select pages from a GridView. To enable paging feature of a GridView :
   - Set AllowPaging property of GridView control to true.
   - Set PageSize property to no of records we want in each page.

iii) **Selecting a row :-** Selecting an item refers to the ability to click a row and have it change color (or become highlighted) to indicate that the user is currently working with this record. The SelectedRowStyle determines how the selected row or cell will appear. To find out what item is currently selected (or to change the selection), we can use the SelectedIndex property. It will be -1 if no item is currently selected. Also, we can react to the SelectedIndexChanged event to handle any additional related tasks.

iv) **Editing with the GridView :-** The GridView provides support for editing that's almost as convenient as its support for selection. This tasks can take place automatically if we use specialized button types. To add edit controls, we use the CommandField column and we set ShowEditButton to true. Set AutoGenerateEditButton property of GridView to true to get Edit button on each row.

Q4. f) Describe asp.net provider model and direct data access method. (5)

❖ **Data Provider model :-** ADO.NET has a set of classes that are used to connect to a specific data source. Each set of data interaction classes is called an ADO. NET data provider. ADO.NET data provider includes SQL Server Data Provider, Oracle Data Provider, OLEDB Data Provider and ODBC Data Provider. The classes used to connect to a specific data source includes Connection, Command, DataReader and DataAdapter.
   - Connection is used to establish a connection to a specific data source.
   - Command is used to execute SQL statements against the data source.
   - DataReader is used to read data from data source.
   - DataAdapter populates a DataSet and resolves updates with the data source.

   Each provider designates its own prefix for naming classes. Thus, the SQL Server provider includes SqlConnection and SqlCommand classes, and the Oracle provider includes OracleConnection and OracleCommand classes. Internally, these classes work quite differently, because they need to connect to different databases by using different low-level protocols. Externally, however, these classes look quite similar and provide an identical set of basic methods because they implement the same common interfaces.

MUQuestionPapers.com

❖ **Direct data access :-** The most straightforward way to interact with a database is to use direct data access. When we use direct data access, we're in charge of building an SQL command (like the ones we considered earlier in this chapter) and executing it. We use commands to query, insert, update, and delete information. The direct data model is well suited to ASP.NET web pages, which don't need to keep a copy of their data in memory for long periods of time.

- To query information with simple data access, follow these steps :
  1. Create Connection, Command, and DataReader objects.
  2. Use the DataReader to retrieve information from the database, and display it in a control on a web form.
  3. Close your connection.
  4. Send the page to the user. At this point, the information your user sees and the information in the database no longer have any connection, and all the ADO.NET objects have been destroyed.
- To add or update information, follow these steps :
  1. Create new Connection and Command objects.
  2. Execute the Command (with the appropriate SQL statement).

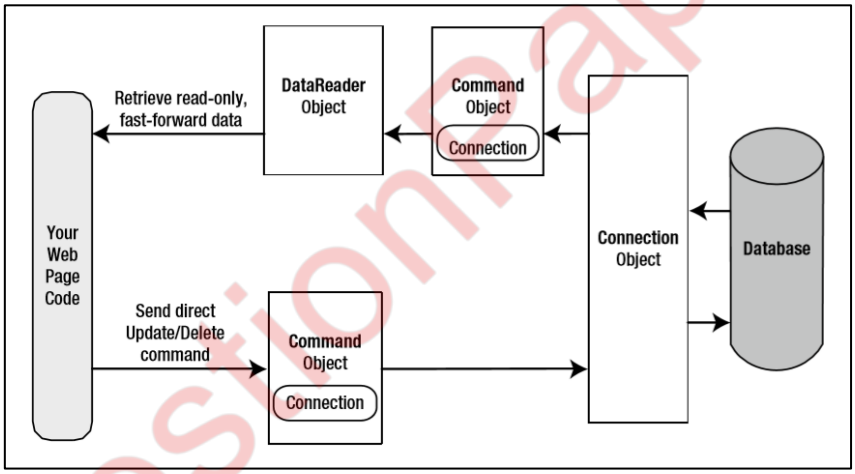


Figure - Direct data access with ADO.NET

Q5. a) Briefly explain different types of authentication in asp.net. (5)

Authentication is the process of determining users identities and forcing those users to prove they are who they claim to be. Usually, this involves entering credentials into some sort of login page or window.

We can use two types of authentication to secure an ASP.NET website :

1. Forms authentication
2. Windows authentication

1. Forms Authentication :-

In Forms authentication, ASP.NET is in charge of authenticating users, tracking them, and authorizing every request. Usually, forms authentication works in conjunction with a database where we store user information, but we have complete flexibility.

To implement forms-based security, follow these three steps:

1. Set the authentication mode to forms authentication in the web.config file.
2. Restrict anonymous users from a specific page or directory in our application.
3. Create the login page.

❖ **Web.config Settings** :- We define the type of security in the web.config file by using the <authentication> tag.

Example -

```
<configuration>
        . . .
        <system.web>
                . . .
                <authentication mode="Forms">
                        <forms name="MyAppCookie"
                                loginUrl="~/Login.aspx"  protection="All"
                                timeout="30" path="/" />
                </authentication>
        </system.web>
</configuration>
```

2. Windows Authentication :-

- In Windows authentication, the web server forces every user to log in as a Windows user.
- This system required that all user have windows user account on the server.
- With Windows authentication, the web server takes care of the authentication process.
- ASP.NET simply makes this identity available to our code for our security checking.
- When we use Windows authentication, we force users to log into IIS before they're allowed to access secure content in our website.
- The user login information can be transmitted in several ways, but the end result is that the user is authenticated using a local Windows account.

To implement Windows-based security with known users, we need to follow three steps:

1. Set the authentication mode to Windows authentication in the web.config file.
2. Disable anonymous access for a directory by using an authorization rule.
3. Configure the Windows user accounts on our web server.

❖ **Web.config Settings** :- To use Windows authentication, we need to make sure the <authentication> element is set accordingly in our web.config file.

Example –

```
<configuration>
        <system.web>
                . . .
```

MUQuestionPapers.com

```
                <authentication mode="Windows" />
                <authorization>
                        <deny users="?" />
                </authorization>
        </system.web>
</configuration>
```

- At the moment, there's only one authorization rule, which uses the question mark to refuse all anonymous users. This step is critical for Windows authentication (as it is for forms authentication). Without this step, the user will never be forced to log in.
- We can also add <allow> and <deny> elements to specifically allow or restrict users from specific files or directories. Unlike with forms authentication, we need to specify the name of the server or domain where the account exists.
  Example –
  <allow users = "WebServer\pqr" />

Q5. b) What is XML? What are its basic characteristics? (5)

XML is short for Extensible Markup Language. It is a very widely used format for exchanging data, mainly because it's easy readable for both humans and machines. If we have ever written a website in HTML, XML will look very familiar to us, as it's basically a stricter version of HTML. XML is made up of tags, attributes and values.
Following are basic characteristics of XML :-

- XML elements are composed of a start tag (like <Name>) and an end tag (like </Name>). Content is placed between the start and end tags. If we include a start tag, we must also include a corresponding end tag. The only other option is to combine the two by creating an empty element, which includes a forward slash at the end and has no content (like <Name />). This is similar to the syntax for ASP.NET controls.
- Whitespace between elements is ignored. That means we can freely use tabs and hard returns to properly align our information.
- We can use only valid characters in the content for an element. We can't enter special characters, such as the angle brackets (< >) and the ampersand (&), as content. Instead, we have to use the entity equivalents (such as < and > for angle brackets, and & for the ampersand). These equivalents will be automatically converted to the original characters when we read them into our program with the appropriate .NET classes.
- XML elements are case sensitive, so <ID> and <id> are completely different elements.
- All elements must be nested in a root element. In the SuperProProductList example, the root element is <SuperProProductList>. As soon as the root element is closed, the document is finished, and we cannot add anything else after it. In other words, if we omit the <SuperProProductList> element and start with a <Product> element, we'll be able to enter information for only one product; this is because as soon as we add the closing </Product>, the document is complete.
- Every element must be fully enclosed. In other words, when we open a subelement, we need to close it before we can close the parent. <Product> <ID> </ID> </Product> is valid, but <Product> <ID> </Product> </ID> isn't. As a general rule, indent when we open a new

element because this will allow us to see the document's structure and notice if we accidentally close the wrong element first.

- XML documents usually start with an XML declaration like <?xml version=• "1.0"?>. This signals that the document contains XML and indicates any special text encoding. However, many XML parsers work fine even if this detail is omitted.

Q5. c) What is the use of XMLTextWriter class? Explain various methods of this class. (5)

- One of the simplest ways to create any XML document is to use the basic XmlTextWriter class.
- The XmlTextWriter class allows us to write to XML files.
- Exceptions thrown by the XmlTextWriter can disclose path information that we do not want bubbled up to the application. Our applications must catch exceptions and process them appropriately.
- When we pass the XmlTextWriter to another application the underlying stream is exposed to that application. If we need to pass the XmlTextWriter to a semi-trusted application, we should use an XmlWriter object created by the Create method instead.
- The XmlTextWriter does not validate any data that is passed to the WriteDocType or WriteRaw methods. We should not pass arbitrary data to these methods.
- Constructor for creating an instance of the XmlTextWriter class is:
  XmlTextWriter (Stream, Encoding)

Methods of XMLTextWriter class :-

1. WriteStartDocument() :- This method writes the XML declaration with the version "1.0"
2. WriteEndDocument() :- This method closes any open elements or attributes and puts the writer back in the Start state.
3. WriteStartElement(String) :- This method writes out a start tag with the specified local name.
4. WriteEndElement() :- This method closes an element.
5. WriteString(String) :- This method writes the given text content.
6. WriteAttributeString(String, String) :- This method writes out the attribute with the specified local name and value.
7. WriteComment(String) :- This method writes out a comment <!--...--> containing the specified text.

Q5. d) What is authorization? Explain adding authorization rules in web.config file. (5)

Once a user is authenticated, authorization is the process of determining whether that user has sufficient permission to perform a given action (such as viewing a page or retrieving information from a database). Windows imposes some authorization checks (for example, when we open a file), but our code will probably want to impose its own checks (for example, when a user performs a task in our web application such as submitting an order, assigning a project, or giving a promotion).

MUQuestionPapers.com

Authorization Rules :- To control who can and can't access our website, we need to add access-control rules to the <authorization> section of our web.config file.

Example –

```
<configuration>
    . . .
    <system.web>
        . . .
        <authentication mode="Forms">
            <forms loginUrl="~/Login.aspx" />
        </authentication>
        <authorization>
            <allow users="*" />
        </authorization>
    </system.web>
</configuration>
```

The asterisk (*) is a wildcard character that explicitly permits all users to use the application even those who haven't been authenticated.

To change this behavior, we need to explicitly add a more restrictive rule, as shown here:

```
<authorization>
    <deny users="?" />
</authorization>
```

The question mark (?) is a wildcard character that matches all anonymous users. By including following rule in applications web.config file, we specify that anonymous users are not allowed.

Q5. e) Explain the working of update progress control in detail. **(5)**

- The UpdateProgress control works in conjunction with the UpdatePanel.
- Essentially, the UpdateProgress control allows us to show a message while a time-consuming update is under way.
- When we add the UpdateProgress control to a page, we get the ability to specify some content that will appear as soon as an asynchronous request is started and will disappear as soon as the request is finished.
- The UpdatePanel performs its work asynchronously in the background. While the asynchronous request is under way, the user won't necessarily realize that anything is happening.
- If the asynchronous request takes some time, the user may assume the page is broken or click the same button multiple times.
- This creates needless extra work for our web application and slowing down the process further. The UpdateProgress control can be used to overcome this problem.
- The UpdateProgress control allows us to show a message while a time-consuming update is under way. It provides a wait message showing that the last request is still being processed.

Example –

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


MUQuestionPapers.com

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager> <br /> <br />
<asp:UpdatePanel ID = "UpdatePanel1" runat = "server">
        <ContentTemplate>
                <asp:Label ID = "Label1" runat = "server" Font-Bold = "True" >
                </asp:Label> <br /><br />
                <asp:Button ID = "Button1" runat = "server" Text = "Click Here to Start"
                onclick="Button1_Click" />
        </ContentTemplate>
</asp:UpdatePanel><br />
<asp:UpdateProgress runat = "server">
        <ProgressTemplate>
                <b>Loading please wait …</b>
        </ProgressTemplate>
</asp:UpdateProgress>
</div>
</form>
```

Q5. f) Explain about implementation of timed refreshes of update panel using timer. (5)

- The Timer control is a server control that embeds a JavaScript component into the Web page.
- The JavaScript component initiates the postback from the browser when the interval that is defined in the Interval property has elapsed.
- We set the properties for the Timer control in code that runs on the server and those properties are passed to the JavaScript component.
- In this walkthrough, we will use a Timer control to update the contents of two UpdatePanel controls.
- The Timer control will be positioned outside the two UpdatePanel controls and it will be configured as a trigger for both panels.
- Timer control can be used to refresh an UpdatePanel at regular intervals without waiting for a user action.
- The Timer control is refreshingly straightforward. We simply add it to a page and set its Interval property to the maximum number of milliseconds that should elapse before an update. For example, if we set Interval to 60000, the timer will force a postback after one minute elapses.

```
<asp:Timer ID = "Timer1" runat = "server" Interval = "1000" />
```

- The Timer control works by rendering a bit of client-side script that starts a JavaScript timer.
- When the JavaScript timer fires (at the interval we specify), the client-side script triggers a postback and raises a server-side Tick event.
- To use the timer with partial rendering, wrap the updateable portions of the page in UpdatePanel controls with the UpdateMode property set to Conditional.
- Then add a trigger that forces the UpdatePanel to update itself whenever the Tick event of Timer occurs.

MUQuestionPapers.com

- Following is the markup for the same:

```
<asp:UpdatePanel ID = "UpdatePanel1" runat = "server" UpdateMode = "Conditional">
    <ContentTemplate>
    ...
    </ContentTemplate>
    <Triggers>
        <asp:AsyncPostBackTrigger ControlID = "Timer1" EventName = "Tick" />
    </Triggers>
</asp:UpdatePanel>
```

- To stop the timer, we simply need to set the Enabled property to false in server-side code.

MUQuestionPapers.com