T.Y. BSC. (IT): Sem. V

Advanced Web Programming

Mumbai University Examination Paper Solution: NOV-18

Q.P. Code: 57833

Q1. Attempt any Three of the Following. **[15]**

Q1(a) What is namespace? Explain with the help of an example. **(5)**

Ans. A namespace is designed for providing a way to way to keep one set of names separate from another. The class names declared in one namespace does not conflict with the same class declared in another.

Define a Namespace= A namespace definition begins with the keyword namespace followed by the namespace name as follows:

```
Namespace namespace_name
{
  //code declaration
}
```

To call the namespace-enabled version of either function or variable, prepend the namespace name as follows:

```
Namespace_name.item_name;
```

Namespaces have the following properties:

- They origin large code projects.
- They are delimited by using the "." Operator.
- The using directive obviates the requirement to specify the name of the namespace for every class.
- The global namespace is the "root" namespace: global::System will always refer to the .NET Framework namespace System.
- Namespace can be nested within other Namespace.

Example:

```
using System;
namespace nestednamespace
{
 namespace n1
 {
   public class A
  {
    public void f1()
   {
     Console.WriteLine("f1 of A of n1");
    }
  }
 }
 namespace n2
{
  public class A
  {
    public void f1()
   {
     Console.WriteLine("f1 of A of n2");
    }
  }
 }
 class program
 {
  static void Main(string[] args)
  {
    n2.A a1 = new n2.A();
    a1.f1();
    n1.A a2= new n1.A();
    a2.f1();
```

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

Output:

f1 of A of n2

f1 of A of n1

Q1(b) Explain jagged array with an example. (5)

Ans. A jagged array is an array of arrays. You can declare a jagged array named scores of type in 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]; }
```

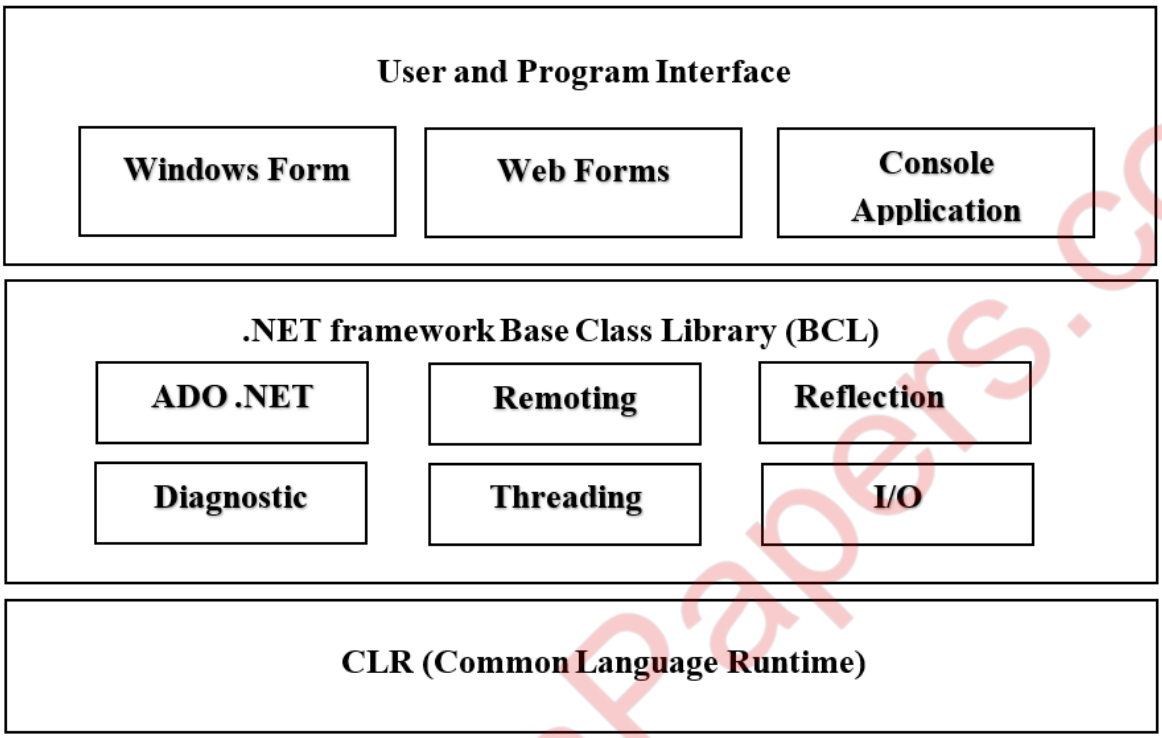
You can initialize a jagged array as –

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

Where, score is an array of integers – scores [0] is an array of 3 integers and score[1] is an array of 4 integers.

Q1(c) What is .NET Framework? Explain its architecture in brief. (5)

Ans.



Following are the components of .Net Framework:

A. User and program Interface:

- It specifies the type of application that can generated using .Net.
- We can create following:
  - i. Console application:
    - — These are the application where input and output are through console command.
    - — There are no forms, no internet connectivity required.
  - ii. Windows application:
    - — Here input and output are through windows form.
    - — No internet connectivity required.
  - iii. Web application:
    - — Here input and output are through web forms.
    - — Int needs remote[server] processing, needs internet connectivity.
    - — Net primarily used to create web application.

B. Base Class Libraries:

- .NET frame supports rich base class libraries to provide access to system functionalities.
- BCL is the foundation on which .NET framework application its components and controls are build.
- Using BCL we can create types of application such as windows application, console application, web application, etc.
- Developers just need to import the BCL in their language code and uses its predefine methods to create application.

C. CLR [Common Language Runtime]:

- It is virtual machine component of .NET framework.
- It manages creation, compilation, memory allotment, execution, garbage collection of .NET application.
- It ensures conversion of sources code to MSIL [Microsoft Intermediate Language] by compiler.
- It also ensures the JIT conversion of your MSIL code to machine code.
- CLR provides additional service like memory management, execution handling garbage collection, security and thread management.
- All programs written in .NET framework irrespective of language are managed by CLR.

Q1(d) Write a program in C# to demonstrate multiple inheritance using interface. (5)

Ans.

```
using System;
    Namespace ConsoleApplication1
    {
     interface calc1
     {
        int add(int a, int b);
     }
     interface calc2
     {
        int sub(int a, int b);
     }
```

```
interface calc3
{
   int mul(int a, int b);
}
interface calc4
{
   int div(int a, int b);
}
class Calculation : calc1, calc2, calc3, clac4
{
   public int result1;
   public int add(int a, int b)
   {
         return result1 =a+b;
    }
    public int result2;
    public int sub(int x, int y)
   {
         return result2 =x-y;
    }
    public int result3;
    public int mul(int p, int q)
   {
         return result3 =p*q;
    }
   public int result4;
   public int div(int r, int s)
   {
         return result4 =r/s;
    }
}
class MultipleInheritance
{
         static void Main(string[] args)
```

```
        {
            Calculation c= new Calculation();
            c.add(3,2);
            c.sub(9,5);
            c.mul(5,3);
            c.div(20,10);
            Console.WriteLine("Multiple Inheritance using Interface: \n");
            Console.WriteLine("Addition:" + c.result1);
            Console.WriteLine("Subtraction:" + c.result2);
            Console.WriteLine("Multiplication:" + c.result3);
            Console.WriteLine("Division:" + c.result4);
            Console.ReadKey();
        }
    }
}
```

Output:

Multiple Inheritance Concept Using Interfaces:

Addition: 5

Subtraction: 4

Multiplication: 15

Division: 2

Q1(e) Explain various types of constructors in C#. **(5)**

Ans. Constructor is a method which gets executed automatically when we create or instantiate object of that class having constructor.

Types of Constructors in C#:

There are five types of constructor in C# as listed below:

- Default Constructor:
  - — A constructor without any parameter is called as default constructor means a constructor which does not have any input parameter is known as default constructor.

    E.g.:

    public test() //Default Constructor

    {……..}

- Parameterized Constructor:
  - A constructor having one or more parameters is called as parameterized constructor means a constructor which having single input parameter or multiple input parameters of same data types or different data types are known as parameterized constructor.

    E.g.:

    ```
    public test(double l, double b)  // Parameterized Constructor
    {……}
    ```
- Copy Constructor:
  - A constructor that contains a parameter of some class type is called as copy constructor.
  - C# does not provide a copy constructor.
  - A copy constructor enables you to copy the data stored in the member variables of an object of the class into another new object means it helps to copy data stored on one object into another new object of the same instance.

    E.g.:

    ```
    class test
    {
        Public test(test t1)  // Copy Constructor
        {
            length = t1.length;
            breath = t1.breath;
        }
    }
    ```
- Static Constructor:
  - Static Constructor should be parameter less means should not contain any input parameter.
  - Program will not execute if static constructor is having any input parameter.
  - Static constructor can be invoked once for any number instances are created and it is invoked only during the first initialization of instance.
  - It is used to initialized static fields of the class static constructor is created using a static keyword as shown below.

E.g.:

```
static int x;
static test()
{
        Console.WriteLine("static Constructor");
        X=10; //set the values for static member here
}
```

- Private Constructor:
  - — Private Constructor are used to restrict the instantiation object using 'new' operator.
  - — A private constructor id special instance constructor.
  - — It is commonly used in classes that contain static members only.
  - — This type of constructor is mainly used for creating singleton object.
  - — If you don't want the class to be inherited we declare its constructor private.

E.g.:

```
private test()  //private constructor
{
        Console.WriteLine("private constructor");
}
```

Q1(f) What is delegate? Explain Multicast delegate with an example. (5)

Ans. A delegate is like a point to function. It is reference type data type and it holds the reference of method. All the delegates are implicitly derived from Sysytem.Delegate class.

A delegate can be declared using delegate keyword followed by function signature as shown below:

Delegate Syntax:

<access modifier> delegate <return type> <delegate_name>(<parameters>)

Multicast Delegate:

The delegate can point to multiple methods. A delegate that points multiple methods is called a multicast delegate. The "+" operator adds a function to the delegate object and the "-" operator removes an existing function from a delegate object.

Example:

```
public delegate void MyDelegate();
class Abc
```

```
{
        public void Show()
        {
                Console.WriteLine("London");
        }
        public void Display()
        {
                Console.WriteLine("New York");
        }
}
class Xyz
{
        public static void Main()
        {
                Abc a1= new Abc();
                MyDelegate m1=  new MyDelegate(a1.Show);
                MyDelegate m2=  new MyDelegate(a1.Display);
                M1=m1+m2+m1+m2-m1;
                m1();
                Console.Read();
        }
}
```

Q2. Attempt any Three of the Following. **[15]**

Q2(a) What is the difference between ListBox and Drop Down List? List and explain any three common properties of these controls. **(5)**

Ans. The basic difference between DropDownList and ListBox in ASP.NET are following:

1. Only one items of DropDownList is visible when it renders on the page. More than one item of the ListBox is visible when it renders on the page.
2. Only one item can be selected on DropDownList. More than one item can be selected in ListBox provided SelectionMode is Multiple as in the below code snippets.

Both controls are rendered as "<select>" html tag in the HTML.

DROPDOWNLIST

```
<asp:DropDownList ID= "drop1" runat= "server">
```

<asp:ListItem Text = “One” Value= “1”/>

<asp:ListItem Text = “Two” Value= “2”/>

</asp:DropDownList>

LISTBOX

<asp:ListBox ID= “list1” runat= “server” SelectionMode= “Multiple”>

<asp:ListItem Text = “One” Value= “1”/>

<asp:ListItem Text = “Two” Value= “2”/>

</asp:ListBox>

Property	Description
Items	The collection of ListIem objects that represents the items in the control. This property returns an object of type ListItemCollection.
Rows	Specifies the number of items displayed in the box. If actual list contains more rows than displayed then a scroll bar is added.
SelectedIndex	The index of the currently selected item. If more than one item is selected, then the index of the first selected item. If no item is selected, the value of this property is -1.
SelectedValue	The value of the currently selected item. If more than one item is selected, then the value of the first selected item. If no items is selected , the value of this property is an empty string(“”).
SelectionMode	Indicates whether a list box allows single selections or multiple selection.

Q2(b) Explain AdRotator control with an example. (5)

Ans. The AdRotator is one of the rich web server control of asp.net. AdRotator control is used to display a sequence of advertisement images as per given priority of image.

Adrotator control display the sequence of images, which is specified in the external XML file. In a xml file we indicate the images to display with some other attributes, like image impressions, NavigateUrl, ImageUrl, AlternateText.

In a Adrotator control images will be changed each time while refreshing the web page.

The Advertisement File:

```
<Advertisements>
<Ad>
<!-- First ad here. -->
</Ad>
<Ad>
<!-- Second ad here. -->
</Ad>
</Advertisement>
```

Advertisement File elements:

Properties	Description
ImageUrl	It is describe the path of image which will be displayed in adrotator control.
NavigateUrl	Describe the navigation webpage link when user click on ad Image.
AlternateText	The text will be displayed when image can not be displayed.
Impressions	Describe the display rate of image or priority of image
Keyword	It is a category of Ad.

The AdRotator Class:

The actual AdRotator class provides a limited set of properties you specify both the appropriate advertisement file in the AdvertisementFile property and the type of window that the link should follow (the Target window).

Target	Description
_blank	The link opens a new unframed window.
_parent	The link opens in the parent of the current frame
_self	The link opens in the current frame.
_top	The link opens in the topmost frame of the current window.

Q2(c) List and explain any four types of validation controls used in ASP.NET. (5)

Ans. ASP.NET validation controls validate the user input data to ensure that useless, unauthenticated, or contradictory data don't get started.

ASP.NET provides the following validation controls:

- RequiredFieldValidator
- CompareValidator
- CustomValidator
- RangeValidator
- RegularExpressionValidator
- ValidationSummar

- RequiredFieldValidator Control:

  The RequiredFieldValidator control is simple validation control, which checks to see if the data is entered for the input control. We can have a RequiredFieldValidator control for each from element on which you wish to enforce mandatory field rule.

  The syntax of the control is given:

  <asp:RequiredFiledValidator ID= "rfvcandidate" runat= "server"
  ErrorMessage= "Please enter name!" ControToValidate= "txtName">
  </asp:RequiredFieldValidator>

- RangeValidator Control:

  The rangeValidator server control is another validator control, which checks to see if a control value is within a valid range. The attributes that to this control are: maximumValue, MinimumValue, and Type.

  The syntax of the control is as given:

  <asp:RangeValidator ID= "rvclass" runat= "server"
  ControlToValidate= "txtclass" 58
  ErrorMessag= "Enter your class (6-12)" MaximumValur="12"
  MinimumValue= "6" Type= "Integer">
  </asp:RangeValidator>

- RegularExpressionValidator:

  The RegularExpressionValidator allows validating the input text by matching against a pattern of a regular expression. The regular expression is set in the ValidationExpression property.

  The syntax of the control is as given:

  <asp: RegularExpressionValidator ID= "string" runat= "server"
  ErrorMessage= "string" ValidationExpression= "string"

ValidationGroup= “string”>

</asp:RegularExpressionValidator>

- CompareValidator Control:

  The CompareValidator control allows you to make comparison to compare data entered in an input control with a constant value or a value in a different control.

  It can most commonly be used when you need to confirm password entered by the user at the registration time. The data is always case sensitive

  It has the following specific properties:

  The basic syntax of the control is as follows:

  <asp:CompareValidator ID= “CompareValidator1” runat= “server”

  ErrorMessage= “CompareValidator”>

  </asp:CompareValidator>

Q2(d) Explain Calendar control with an example in ASP.NET. (5)

Ans. The Calendar control present a miniature calendar that you can place in any web page. Like most rich controls, the calendar can be programmed as a single object, but it renders itself with dozens of lines of HTML output.

<asp:Calendar id= “MyCalendar” runat= “server” />

The Calendar control presents a single-month view.

When the user clicks a date, the date becomes highlighted in gray box. You can retried the selected day in your code as a DateTime object from the Calendar.SelectedDate property.

Properties:

Property	Description
SlectionMode	Specifies the type of selection that can be made. Acceptable values are Day, DayWee.
SelectedDate	The currently selected date if single date was selected, or the first selected date if m.
SelectedDates	A collection that contributes all of the selected dates in sequence. To determine the property of this selection.

aspx code for the calendar control:

<asp:CalendarID= “calArrival” runat= “server” visible=”false”

BorderColor= “Black” BorderStyle= “Solid”>

<TodaydayStyleBackColor= “Blue” Font-Bold= “true” ForeColor= “White” />

<TitleStyleBackColor= “Blue” Font-Bold= “true” ForeColor= “White” />

<NextPrevStyleForeColor= “Wheat” />

</asp:Calendar>

The SelectionChanged event handler for calendar control.

This occur immediately the user selects a new day or browser to a new month.

You can react to these events and update other portions of the web page to correspond to the current calendar month.

You can react to these events and update other portions of the web page to correspond to the current calendar month.

```
protected void calArrival_SelectionChnaged
(object sender, EventArgs e)
{
        ddlMonth.SelectedValue =  calArrival.SelectedDate.Month.ToString();
        ddlDay.SelectedValue =  calArrival.SelectedDate.Day.ToString();
        calArrival.Visible= false;
}
```

Q2(e) Write short note on page class. **(5)**

Ans. Every web page is custom class that inherits from System.Web.UI.Page. By inheriting from this class, your web page class acquires a number of properties and methods that your code can use. These includes properties for enabling Caching, validation, and tracing.

Property	**Description**
IsPostBack	This Boolean property indicates whether this is first time the page is being run(false) or whether the page is being resubmitted in response to a control event, typically with stored view state information(true)
EnableViewState	When set of false, this overrides the EnableViewState property of the contained controls, thereby ensuring that no controls will maintain state information.
Application	This collection holds information that’s shared between all users in your website.
Session	This collection holds information for single user, so it can be used in different pages.

Cache	This collection allows you to store objects that are time-consuming to create so they can be reused in other pages or for other clients. This technique, when implemented property, can improve performance of your web page.
Request	This refers to an HttpRequest object that contains information about the current web request. You can use the HttpRequest object to get information about the user's browser, although you will probably prefer to leave these details to ASP.NET.
Response	This refers to an Httpresponse object that represents the response ASP.NET will send to the user's browser.
Server	This refers to an HttpServerUtility object that allows you to perform a few miscellaneous tasks.
User	If the uses have been authenticated, this property will be initialized with user information.

Q2(f) Explain SiteMapPath control in ASP.NET. (5)

Ans. The SiteMapPath control basically is used to access web pages of the website from one webpage to another. It is a navigation control and displays the map of the site related to its web pages. This map includes the pages in the particular website and displays the name of those pages. We can click on that particular page in the Site Map to navigate to that page.

We can say that the SiteMapPath control displays links for connecting to URLs of other pages. The SiteMapPath provides breadcrumb navigation, which means it shows the users current location and allows the user to navigate up the hierarchy to a higher level by using links. The SiteMapPath control is useful because it provides both an at-a-glance view that shows the current position and a way to move up the hierarchy.

The SiteMapPath control uses a property called SiteMapProvider for accessing data from database and it stores the information in data source.

The representation of the SiteMapPath control is as follows:

- Root Node-> Child Node.
- Common attributes of SiteMapPath control:
  - ParentLevelsDisplayed: It specifies the number of levels of parent nodes and then displays the control accordingly related to the currently displayed node.

- RenderCurrentNodeAsLink: It specifies whether or not the site navigation node that represents the currently displayed page is rendered as a hyperlink.
- PathSeperator: It specifies the string that displays the SiteMapPath nodes in the rendered navigation path.

➢ Style properties of the SiteMapPath:

- CurrentNodeStyle: It specifies the style used gor th display text for the current node.
- RootNodeStyle: It specifies the style for the root node style text.
- NodeStyle: It specifies the style used the display text for all nodes in the site navigation path.

Q3. Attempt any Three of the Following. **[15]**

Q3(a) What is user-defined exception? Explain with example. **(5)**

Ans. We have seen built-in exception classes however, we often like to raise an exception when the business rule of our application gets violated. So, for this we can create a custom exception class by deriving Exception or ApplicationException class. User can also define their own exception objects to represent custom error conditions. All you need to do is create an instance of the appropriate exception class and then use the throw statement.

Example:

```
using System;
namespace UserException
{
        class Program
        {
                Static void Main (String [] args)
                {
                        Int bal= 20000;
                        int wit;
                        {
                                Console.WriteLine("Amount to Withdraw");
                                Wit= c=Convert.ToInt32(Console.ReadLine(b));
                                If (bal-wit > =1000)
                                Console.WriteLine("New Balance=" + (bal[]wit));
                                else
                                throw new NoMoney();
```

```
                    }
                    catch(No money e)
                    {
                        Console.WriteLine("No money Left");
                    }
                    Catch(Exception e)
                    {}
                    finaly
                    {
                        Console.WriteLine("Continues");
                    }
                    Console.ReadLine();
                }
                class Nomeney: Exception
                {}
        }
}
```

Q3(b) What is debugging? Explain the process of debugging in detail. (5)

Ans. Debugging allows the developers to see how the code works in step-by-step manner, how the values of the variables change, how the objects are created and destroyed, etc.

When the site is executed for the first time, visual studio displays a prompt asking whether it should be enabled for debugging. When debugging is enabled, the following lines of codes are shown in the web.config:

```
<system.web>
    <compilation debug= "true">
        <assembiles>
        ……..
        </assembiles>
    </compilation>
</system.web>
```

The debug toolbar provides all the tools available for debugging:

- Breakpoints: Breakpoints specifies the runtime to sun a specific line of code and then stop execution so that the code could be examined and perform various debugging jobs such as, changing the value of the variables, step through the codes, moving in and out of functions and methods etc.
- To set a breakpoint, right click on the code and choose insert break point. A red dot appears on the left margin and the line of code is highlighted.
- Next when you execute the code, you can observe its behavior.
- The debug windows: Visual Studio provides the following debug windows, each of which some program information, the following table listed the windows:

Window	Description
Immediate	Displays variables and expression.
Autos	Displays all variables in the current and previous statements.
Locals	Displays all variables in the current context.
Watch	Displays up to four different sets of variables.
Call Stack	Displays all methods in the call stack.
Threads	Displays and control threads.

Q3(c) Write short note on cookies in ASP.NET. (5)

Ans. Cookies provide another way to store state information for later use. Cookies are small files that are created in the web browser's memory or on the client's hard drive. One advantage of cookies is that they work transparently, without the user begin aware that information needs to be stored. Cookies are easily accessible and readable if the user finds and opens the corresponding file. Some users disable cookies on their browser, which will cause problem for web applications that require them.

Step to use cookies:

1) One should import the System.Net namespace so you can easily work with the appropriate types:

   Using System.Net;

2) To set a cookie, just create a new HttpCookie object. It can be then filled with string information and can be attached to the current web response:

```
// Create the cookie object.
HttpCookie cookie = new HttpCookie("priority");
// Set a value in it.
Cookie["SubjectPriiority"] = "ASP";
// Add another value.
cookie ["Country"] = "INDIA";
// Add it to the current web response.
Response.Cookie.Add(cookie);
```

3) A cookie added in this way will persist until the user closes the browser and will be sent with every request. To create a longer-lived cookie, you can set an expiration date:

```
// This cookie lives for one year.
cookie.Expires = DateTime.Now.AddYears(1);
```

4) You retrieve cookies by cookie name, using the Request.Cookies collection:

```
HttpCookie cookie = new HttpCookie("Priority");
```

5) The only way to remove a cookie is by replacing it with a cookie that has an expiration date that has already passed. This code demonstrates the technique:

```
HttpCookie cookie =  new HttpCookie("Priority");
Cookie.Expires = DateTime.Now.AddYears(-1);
Response.Cookie.Add(cookie);
```

Cookie's common property

1. Domain => Which is used to associate cookies to domain.
2. Secure => We can enable secure cookie to set true(HTTPs).
3. Value => We can manipulate individual cookie.
4. Values => We can manipulate cookies with key/value pair.
5. Expires => Which is used to set expire date for the cookies.

Advantages of Cookie:

1. Its clear text so user can able to read it.
2. We can store user preference information on the client machine.
3. Its easy way to maintain.
4. Fast accessing.

Disadvantages of Cookie:

1. If user clear cookie information we can't get it back.
2. No security.
3. Each request will have cookie information with page.

Q3(d) What is ViewState in ASP.NET? State its advantages and disadvantages. (5)

Ans. ViewState is one of the most import and useful client-side state management mechanism. It can store the page value at the time of post back of your page. ASP.NET pages provide the ViewState property as built-in structure for automatically storing values between requests for the same page.

Example: Of you want to add one variable in ViewState

Hide Copy Code ViewState[“Var”] = Count;

For Retrieving information from ViewState

Hide Copy Code

string Test=ViewState[“TestVal”];

sometimes you may need to type cast ViewState value to retrive. As I give an example to strore and retrieve object in view state in the last of this article.

Advantages:

- Easy to implement.
- No server resources are required.
- Enhanced security features, like it can be encoded and compressed.

Disadvantages:

- It can be performed overheated if we are going to storage to state large amount of data, because it is associated with page only.
- Its stored in hidden field in hashed format sill it can be easily trapped.
- It does not have any support on mobile devices.

Q3(e) Create a web application to demonstrate use of Master Page with applying styles and themes for page beautification. Write necessary steps with code for the same. (5)

Ans. Steps to add a master page to a project:

Choose website -> Add New Item command ->choose Master Page -> specify the name of the master page -> click on OK.

Two ways to create a new content page:

Choose Website -> Add New Item command -> select web form -> check the select master page check box -> click add. When the select a Master Page dialog box appears, select the master page you want and click OK.

Select the master page in the solution explore, then choose the website -> Add content page command.

CODE : StyleSheet.css body

```
{
        border-style: dashed
        background -color: Lime;
        font-family: Arial, Helvetica, sans-serif;
        font-size: 18px;
}
```

Theme1-SkinFile.skin

<asp:Lable runat= “server” ForeColor= “red” FontSize= “14pt” Font-Names= “Verdana” SkinID= “LableSkin” />

<asp:Lable runat= “server” ForeColor= “Blue” FontSize= 25pt” Font-Names= “Verdana” Skin />

<asp:button runat= “server” Borderstyle= “solid” BorderWidth= “2px” Bordercolor= “Blue” Backcolor= “Blue”/>

MasterPage.master

<%@ Master Language = “C#” AutoEventWiredup= “true” CodeFile= “MasterPage.master.cs” Inherits= “MasterPage” %>

<html xmlns= “ ”>

<head runat= “server”>

<title> TyBScIT </title>

<asp: ContentPlaceHolder id= “head” runat = “server”>

</asp:ContentPlaceHolder>

<link herf= “Styles.stylesheet.css” rel= “stylesheet” type= “text/css” />

</head>

<body>

<form id= “form1” runat= “server”>

<h1> TyBScIT Sem V Subjects</h1>

<asp:ContentPlaceHolder id= “ContentPlaceHolder1” runat= “server”>

```
</asp:ContentPlaceHolder>
</div>
</form>
</body>
</html>
```

ContentPage.aspx

```
<%@ Page Title= " " Language= "C#" MasterPageFiles= "~//MasterPage.master"
AutoEventWiredup= "true" CodeFile= "ContentPage.aspx.cs"
Inherits= "Default2" Theme= "Theme1" %>
<asp:Content ID= "Content1" ContentPlaceHolderID= "head" Runat= "Server">
</asp:Content>
<asp:Content ID= "Content2" ContentPlaceHolderID= "ContentPlaceHolder1" Runat=
"Server">
<asp:Label ID= "Label1" runat= "server" Text= "SPM">
</asp:Label>
<br/>
<br/>
<asp:Label ID= "Label2" runat= "server" Text= "AWP">
</asp:Label>
<br/>
<asp:Label ID= "Label3" runat= "server" Text= "IOT">
</asp:Label>
<br/>
<asp:Label ID= "Label4" runat= "server" Text= "EJ">
</asp:Label>
<br/>
<asp:Label ID= "Label5" runat= "server" Text= "NGT">
</asp:Label>
<br/>
<br/>
<asp:Button ID= "Button1" runat= "server" Text="show" onclick= "Button1_Click" />
</asp: Content>
```

Q3(f) Explain the four most important selectors present in CSS. (5)

Ans. Following are the selectors present in CSS:

1. Universal Selector:

   The universal selector, indicated by an asterisk (*), applies to all element in your page. The universal selector can be used to set global settings like a font family. The following rule set changes the font for all elements in our page to Arial:

   ```
   *{
    Font-family: Arial;
    }
   ```

2. Type Selector:

   The type selector enables us to point to an HTML element of a specific type. With a type selector all HTML elements of that type will be styled accordingly.

   ```
   h1
   {
   Color:Green;
    }
   ```

3. ID Selector:

   Th id selector always prefixed by a hash symbol (#) and enables us to refer to a single element in the page. Within an HTML or ASPX page, we can give an element a unique ID using the id attribute. With the ID selector, we can change the behavior for that single element, for example:

   ```
   #IntroText
   {
   Font-style: italic;
    }
   ```

   Because we can reuse this ID across multiple pages in our site, you can use this rule to quickly change the appearance of an element that use once per page, but more than once in our site, for example with the following HTML code:

   ```
   <p id= “IntroText”>I am italic because I have right ID </p>
   ```

4. Class Selector:

   The class selector enables us to style multiple HTNL elements through the class attribute. This is handy when we want to give the same type of formatting to several unrelated HTML elements. The following rule changes the text to red and bold for all HTML elements that have their class attributes set to highlight:

.Highlight

{ font-weight:bold; color:Red; }

- The following code snippet use the Highlight class to make the contents of a element and a link (<a>) appear with a bold typeface:

This is normal text but this link is Red and Bold as well.

This is also normal text but

 this link is red and bold as well

Q4. Attempt any Three of the Following. **[15]**

Q4(a) List and explain ADO.NET object. **(5)**

Ans. ADO.NET includes many objects we can use to work with data. Some important objects of ADO.NET are:

- Connection: To interact with a database, we must have a connection to it. The connection helps identify the database server, the name, user name, password, and other parameters that are required for connection to the data base.

  A connection object is used by command objects so they will know which database to execute the command on.
- Command: The command object is one of the basic components of ADO.NET. the command object uses the connection object to execute SQL quires.

  The quires can be in the form of Inline text, stored procedures or direct table access. An important feature of command object is that it can be used to execute quires and stored procedures with parameters. If a select query is issued, the result set it returns in usually stored in either a DataSet or a DataReader object.
- DataReader: Many data operation require that we only get a stream of data for reading. The data reader object allows us to obtain the result of a SELECT statement from a command object. For performance reasons, the data returned from a data reader is a fast forward-only stream of data.

  This means that we can only pull the data from the stream in a sequential manner this is good for speed, but if we need to manipulate data, then a DataSet is a better object to work with.
- DataSet: DataSet objects are in-memory representation of data. They contain multiple Datatable objects, which contain columns and rows, just like normal database tables. We can even define relations between tables to create parent-child relationships.

The DataSet is specifically designed to help manage data in memory and to support disconnected operations on data, when such a scenario make sense.

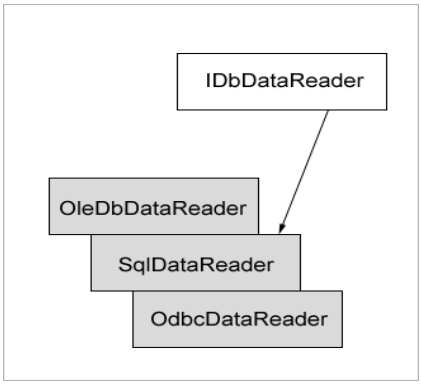
- DataAdapter: The data adapter fills a DataSet object when reading the data and writers in a single batch when persisting changes back to the database. A dada adapter contains a reference to the connection object and opens and close the connection automatically when reading from or writing to the database.

Q4(b) What is Data Reader in ADO.NET? Explain with an example. (5)

Ans. A data reader provides an easy way for the programmer to read data from a database as if it were coming from a stream. The DataReader is the solution for forward streaming data through ADO.NET. The data reader is also called a firehose cursor or forward read-only cursor because it moves forward through the data. The data reader not only allows you to move forward through each record of database, but it also enables you to parse the data from each column. The DataReader class represents a data reader in ADO.NET.

Similar to other ADO.NET objects, each data provider has a data reader class for example; OleDbDataReader is the data reader class for OleDb data providers. Similarly, SqlDataReader and ODBC DataReader are data reader classes for SQL and ODBC data providers, respectively.

The IDataReader interface defines the functionally of a data reader and works as the base class for all data provider-specific data reader classes such as OleDataReader. SqlDataReader, and OdbcDataReader. Figure 5-36 shows some of the classes that implement IDbDataReader.



The DataReader properties,

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

The DataReader methods,

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

Code for DataReader:

```
SqlCommand myCMD = new SqlCommand("SELECT CategoryID, CategoryName FROM Categories;" + "SELECT EmployeeID, LastName FROM Employees", nwindConn);

nwindConn.Open();

SqlDataReader myReader = myCMD.ExecuteReader();

do {

   Console.WriteLine("\t{0}\t{1}", myReader.GetName(0), myReader.GetName(1));

  while (myReader.Read()) Console.WriteLine("\t{0}\t{1}", myReader.GetInt32(0), myReader.GetString(1));  }

while (myReader.NextResult());

myReader.Close();

nwindConn.Close();
```

Q4(c) Explain SqolDataSouece in ADO.NET. (5)

Ans. The SqlDataSource control represents a connection to a relational database such as AQL server or Oracle database, or data accessible through OLEDB or Open Database Connectivity(ODBC). Connection to data is mode through teo important properties ConnectionString and ProviderName.

SqlDataSource control attributes:

Attribute	Description
ID	The ID for the SqlDataSource control.
Runat	Must specify "server".
ConnectionString	The connection string. In most cases, you should use a <%$ expression to specify the name of connection string saved in the web.config file.
ProviderName	The name of the provider used to access the database. Values can be System.Data.Odbc, System.data.Oledb, System.Data.OracaleClient, or System.DataSqlClient. The default is Sysytem.Data.SqulClient.
SelectCommand	The SQL select statement executed by the data source to retrieve data.

The following code provides the basic syntax of the control:

<asp:SqldataSource runat= "server" ID== "MySqlSource" ConnectionString= '<%$ ConnectionString:Empconstr %>' SelectionCommand= "SELECT* FROM EMPLOYEES" />

The following code shows a data source control enabled for data manipulation.

<asp:SqldataSource runat= "server" ID= "MySqlSource" ConnectionString= '<%$ ConnectionString:Empconstr %>' SelectionCommand= "SELECT* FROM EMPLOYEES" UpdateCommand= "UPDATE EMPLOYEES SET LASTNAME=@lame" DeleteCommand= "DELETE FROM EMPLOYEES WHERE EMPLOYEEID=@eid"

FilterExpression= "EMPLOYEEID >10" >

</asp:SqlDataSource>

Q4(d) What is GridView Control? Explain with example. **(5)**

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

- Binding to data source controls, such as SqlDataSource.
- Built–in sort capabilities.
- Built-in update and delete capabilities.
- Built-in paging capabilities.
- Built-in row selection capabilities.
- Multiple key fields.
- Multiple data fields for the hyperlink columns.
- Customizable appearance through themes and styles.

Sorting allows the user to sort the items in the GridView control with respect to specific column by clicking on the columns header. To enable, sorting, set the AllowSorting property to true.

AllowSorting=”True”

Instead of displaying all the records in the data source at the same time, the GridView control can automatically break the records up into page. To enable paging, set the AllowPaging=”True”

Also we can set how many rows we want to see in page.

PageSize-“4”

Example:

```
string constr = ConfigurationManager.ConnectionStrings[“constr”]
ConnectionString;
using (SqlConnection con = new SqlConnection(str))
{
using (SqlCommand cmd = new SqlCommand(“SELECT * FROM Customer”))
{
```

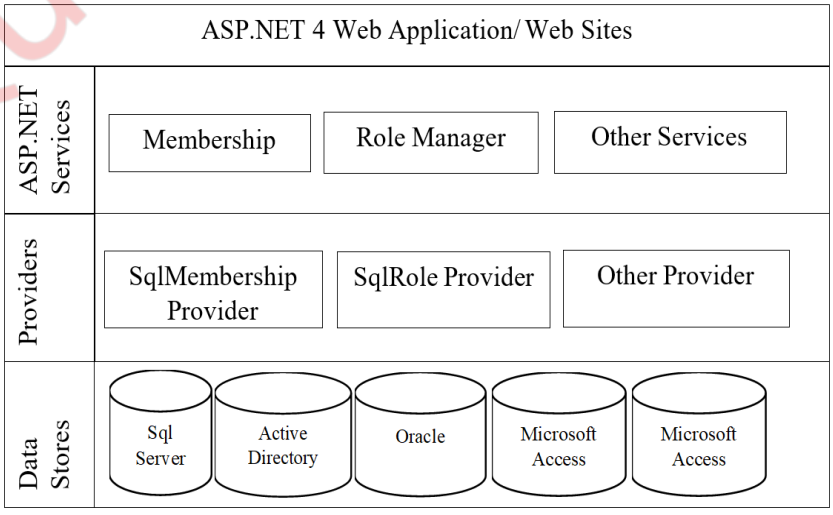
```
cmd.Connection = con;

using (SqlDataAdapter Sda = new SqlDataAdapter(cmd) )

{

DataTable dt = new dataTable();

Sda.fill(dt);

GridView1.DataSource = dt;

GridVie1.dataBind();

}

}

}
```

Q4(e) What are the application service provider in ASP.NET? Explain. (5)

Ans. ASP.NET 4 ships with a number of application services, of which the most important ones are:

- Membership: Enables us to manage and work with user accounts in our system.
- Roles: Enables us to manage the roles that your users can be assigned to.
- Profile: Enables us to store user-specific data in a back-end database.

Overview of these services and how they are related to our web site and the underlying data stores that the services may use show are as follows:



A provider is software that provides a standardized interface between a service and a data source. ASP.NET providers are as follows:

- ✓ Membership
- ✓ Role Management
- ✓ Sitemap
- ✓ Profile
- ✓ Session state etc.

At the top of the diagram you see the ASP.NET 4 web sites and web application that represent the web sites that you build. These web sites can contain controls like the login controls that it turns can talk to the ASP.NET application services such as membership and profile. To create a flexible solution, these services don't talk to an underlying data source directly, but instead talk to a configured provider.

A provider is an interchangeable piece of software that is designed for a specific task. For example, in the case of the work with users in the underlying data store. You can configure different providers for some application service depending on your need.

Q4(f) Difference between FormView and DetailsView in ASP.NET. (5)

Ans. Following are some points which shows the difference between FormView and DetailsView:

- Like DetailView, FormView also displays a single record from the data source at a time.
- Both controls can be used to display, edit, insert and delete database records but one at a time. Both have paging features and hence support backward and forward traversal.
- FormView is a new data-bound control that is nothing but a templated version of detailsView control.
- The major difference between DetailsView and FormView is, here user need to define the rendering template for each item.
- The FormView control provides more formatting and layout options than DetailsView.
- The DetailsView control can uses <BoundField> elements or <TemplateField> elements to display bound data whereas FormView can use only templates to display bound data.
- The FormView control renders all fields in a single table row whereas the DetailsView control displays each field as a table row.

- When compare to DetailsView, the FormView control provides more control over the layout.

A FormView control after a data source has been assigned

A DetailsView control that displays data

Code	Code1
Name	Name1
Description	Description1
Edit New	
1 2 3 4 5	

Q5. Attempt any Three of the Following. **[15]**

Q5(a) Explain XmlTextReader and XmlTextWriter with an example. **(5)**

Ans: XmlTextReader:

Provides forward-only, read-only access to a stream of XML data. The current node refers to the node on which the reader is positioned. The reader is advanced using any of the read methods and properties reflect the value of the current node. The syntax to declare an object of this class is as follows:

XmlTextReader reader = new XmlTextReader ("XML1.xml");

XmlTextReader provides the following functionality:

- Enforces the rules of well-formed XML.
- XmlTextReader does not provide data validation.
- Checks that DocumentType nodes are well-formed. XmlTextReader checks the DTD for well-formedness, but does not validate using the DTD.
- For nodes where NodeType is XmlNodeType.EntityReference, a single empty EntityReference node is returned (that is, the Value property is String.Empty).
- Does not expand default attributes.

Example:

```
String xmlNode= "~\XMLFile.xml";

XmlReader creader = XmlReader.Create(xmlNode);

While (xreader.Read())

{

    Switch (xreader.NodeType)

    {

        Case XmlNodeType.Element;

        ListBox1.Items.Add("<" + xreader.Name + ">");

        Break;

        Case XmalNodeType.Text;

        ListBox1.Add(xreader.value);

        Break;

        Case XmlNode Type.EndElement;

        ListBox1.Items.Add("<"+ wxreader.Name + ">");
```

```
            Break;
        }
}
```

XmlTextWriter:

One of the simplest ways to create or read any XML document is to use the basic XmlTestWriter and XmlTextReader classes. You create the entire XML document by calling the methods of the XmlTextWriter, in the right order. To start a document, you always begin by calling WriteStartDocument(). To end it, you call WriteEndDocument().

You write the elements you need, in three steps. First, you write the start tag by calling WriteStartElement(). Then you write attributes, elements, and text content inside. Finally, you write the and tag by calling WriteEndElement().

The methods you see always work with the current element, so if you call WriteAttributeString() and follow it up with a call to WriteAttributeString(), you are adding an attribute to that element. Similarly, if you use WriteString(), you insert text content inside the current element, and if you use WriteStartelement() again, you write another element, nested inside the current element.

Example:

```
XmlTextWriter writer= new XmlTextWriter("~\ XMLFile.xml",null);
{
    Writer.WriterStartDocument();
    writer.WriterStartElement("Details", "");
    writer.WriterStartElement("First name", "wer");
    writer.WriterStartElement("Last name", "wedw");
    writer.WriterStartElement("College", "ABC");
```

```
        writer.WriteEndElement();

        writer.WriteEndDocument();

        writer.Clode();

}
```

Q5(b) What is Xelement? Explain with an example. (5)

Ans. The XElement class represents an XML element in System.Xml.XLing. XElement loads and parse XML. It allows us to remove lots of old code and eliminate the possibility of bugs and typos. The XAttribute class represents an attribute of an element. XElement.Save method save the contents of XElement to a XML file. You can create an element and supply text xontent that should be placed inside using code like this: XElement element = new XElement("price", 23.99); this is already better than the XmlTextWriter, which forces you to start an element, insert its content, and close it with three separate statements.

Heres how it works. Both the XDocument and XElement class include a constructor that takes a parameter array holds a list of nested nodes. You can extend this technique to create an entire XML document, complete with elements, text content, attributes, and comments. For example, here's the complete code that creates the SuperProProductList.xml document in memory. When the document is completely build, the code saves it to a file using the XDocument.Save() methods.

Example:

```
XDocument doc = new XDocument(new XDeclaretion("1.0", null, "yes"),

new XComment("Created with the XDocument Class"),

new  XElement("SuperProProductList"),

new Xelement("Product", new XAtrtribute("ID",1),

new XAttribute("name", "chair"),

new XElement("price", 49.33) ),
```

```
new xElement("Product",
new XAttribute("ID",2),
new XAttribute("Name", "Fresh Fruit Basket"),
new XElement("Price", 49.99),
)
new xElement("Product",
new XAttribute("ID",3),
new XAttribute("Name", "car"),
new XElement("Price", 445.657),
)
)
);
Doc.Save(file);
```

Q5(c) What do you mean by authentication? Explain its types. (5)

Ans. Authentication:

- In ASP.NET there are many different types of authentication procedures for web applications. If you want to specify your own authentication methods then that also is possible.
- The different authentication modes are accepted through settings that can be applied to the applications web.config file.
- The web.config file is XML-bsed file which allows changing of ASP.NET behavior easily.
- ASP.NET provides three different authentication provides:

- Windows
- From
- Passport

✓ Windows: It allows to authenticate user based on their windows accounts. This provider uses IIS to perform the authentication and then asses the authenticated identity to your code. This is the default provides for ASP.NET. Syntax: <authencation mode= "windows">

✓ Forms: It uses custom HTML forms to collect authentication information and lets you use your own logic to authenticate users. The users identification are stored in a cookie for use during the session. Syntax: <authencation mode= "forms">

✓ Passport: it uses Microsoft passport service to authenticate users. It maintain only username and password. Syntax: <authencation mode= "password">

Q5(d) What do you mean by Impersonation in ASP.NET? Explain. (5)

Ans: ASP.NET impersonation used to control the execution of the code in authenticated and authorized client. It is not a default process in the ASP.NET application. It is used for executing the local thread in an application.ASP.NET impersonation is used to control the execution of the code in authenticated and authorized client. It is not a default process in the ASP.NET application. It is used for executing the local thread in an application. If the code changes the thread, the new thread executes the process identity by default.

The impersonation can be enabled in two ways as mentioned below :

- Impersonation enabled: ASP.NET impersonates the token passed to it by IIS, can be an authenticated user or an internet user account.
- The syntax for enabling is as shown below:

  <identity impersonate="true" />
- Impersonation enabled for a specific identity: ASP.NET impersonates the token generated using the identity specified in the Web.config file.

  <identity impersonate="true"
  userName="domain\user"
  password="password"/>
- Impersonation disabled: It is the default setting for the ASP.NET application.
- The process identity of the application worker process is the ASP.NET account.

  <identity impersonate="false" />

Q5(e) Explain ASP.NET AJAX control toolkit. (5)

Ans. The ASP.NET AJAX control toolkit is a joint project between Microsoft and the ASP.NET community. It contains of dozens of controls that use the ASP.NET AJAX libraries to create sophisticated effects. The ASP.NET AJAX control toolkit is completely free. It includes full source code, which helpful if you are ambitious enough to want to create your own custom controls that use ASP.NET AJAX features.

It uses extenders that enhance the standard AP.NET web controls. That way, you don't have to replace all the controls on your web pages instead, you simply plug in the new bits of functionality that your need. Installing the ASP.NET AJAX control toolkit. You can download the ASP.NET AJAX control toolkit from http://ajaxcontroltoolkit.codeplex.com. Click the download tab, and find the toolkit for the latest version of .NET. After you downloaded this ZIP file, you can extract the files it contains to a more permeant location on your hard drive.

After installation of toolkit, all the component from AjaxControlToolkit.dll will appear in the list in the choose toolbox items dialog box. ASP.NET AJAX control toolkit, currently includes about 50 components. The easiest way to start experimenting with other controls is to surf to www.asp.net/ajaxlibarary/act_tutorials.ashx, where you will find a reference that describe each control.

Some of the controls are like AutoComplete, Color Picker, calendar, Watermark, Modal Popup extender, slideshow Extender and more of the useful control. The ASP.NET AJAX control toolkit is now maintained by DevExpress team. The current version of ASP.NET AJAX toolkit is v16.1.0.0. there are lot of new enhancement in the current version from new controls to bug fixes in all controls.

Q5(f) Create a web application to demonstrate the use of HTMLEditorExtender Ajax Control. Write the code of default.aspx and required property settings for the same. (5)

Ans:

```
<%@Page Language="C#" AutoEventWireup="true"
    CodeFile="default.aspx.cs" Inherits="_default" %>
    <%@RegisterAssembly="AjaxControlToolkit"
    Namespace="AjaxControlToolkit.HtmlEditor.Sanitizer"TagPrefix="ajaxToolkit"
    %>
    <!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">
<div>
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
<asp:TextBox runat="server" ID="txtBox1" TextMode="MultiLine" Columns="50"
Rows="10" Text="Hello <b>world!</b>" /><br />
<ajaxToolkit:HtmlEditorExtender      ID="htmlEditorExtender1"
TargetControlID="txtBox1" runat="server"  DisplaySourceTab="true">
<Toolbar>
<ajaxToolkit:Undo />
<ajaxToolkit:Bold/>
<ajaxToolkit:Italic />
<ajaxToolkit:Underline />
<ajaxToolkit:Copy />
<ajaxToolkit:JustifyLeft/>
<ajaxToolkit:JustifyCenter/>
<ajaxToolkit:JustifyRight/>
<ajaxToolkit:Delete />
<ajaxToolkit:SelectAll />
<ajaxToolkit:FontColorSelector />
<ajaxToolkit:InsertImage />
</Toolbar>
</ajaxToolkit:HtmlEditorExtender>
</div>
</form>
</body>
</html>
```