UNIT V: ASP.NET WITH C#

UNIVERSITY QUESTIONS WITH ANSWER

Q1. Explain XmlTextReader and XmlTextWriter with an example.
XmlTextReader:

The XmlTextReader moves through your document from top to bottom, one node at a time. You call the Read() method to move to the next node. This method returns true if there are more nodes to read or false once it has read the final node. The current node is provided through the properties of the XmlTextReader class, such as NodeType and Name.
A node is a designation that includes comments, whitespace, opening tags, closing tags, content, and even
the XML declaration at the top of your file. To get a quick understanding of nodes, you can use the XmlTextReader to run through your entire document from start to finish and display every node it encounters.
Example:

```
string file = Path.Combine(Request.PhysicalApplicationPath,
 @"App_Data\SuperProProductList.xml");
FileStreamfs = new FileStream(file, FileMode.Open);
XmlTextReader r = new XmlTextReader(fs);
// Use a StringWriter to build up a string of HTML that
// describes the information read from the XML document.
StringWriter writer = new StringWriter();
// Parse the file, and read each node.
while (r.Read())
{
 // Skip whitespace.
if (r.NodeType == XmlNodeType.Whitespace) continue;
writer.Write("<b>Type:</b> ");
writer.Write(r.NodeType.ToString());
writer.Write("<br>");

 // The name is available when reading the opening and closing tags
 // for an element. It's not available when reading the inner content.
if (r.Name != "")
 {
writer.Write("<b>Name:</b> ");
writer.Write(r.Name);
writer.Write("<br>");
 }
 // The value is when reading the inner content.
```

```
if (r.Value != "")
 {
writer.Write("<b>Value:</b> ");
writer.Write(r.Value);
writer.Write("<br>");
 }
if (r.AttributeCount> 0)
 {
writer.Write("<b>Attributes:</b> ");
for (int i = 0; i <r.AttributeCount; i++)
 {
writer.Write(" ");
writer.Write(r.GetAttribute(i));
writer.Write(" ");
 }
writer.Write("<br>");
 }
writer.Write("<br>");
}
fs.Close();
// Copy the string content into a label to display it.
lblXml.Text = writer.ToString();
```

XmlTextWritter

One of the simplest ways to create or read any XML document is to use the basic XmlTextWriter and XmlTextReader classes. These classes work like their StreamWriter and StreamReader relatives, except that they write and read XML documents instead of ordinary text files. First, you create or open the file. Then, you write to it or read from it, moving from top to bottom.

Example:

```
using System.IO;
usingSystem.Xml;
```

Here's an example that creates a simple version of the SuperProProductList document:

```
// Place the file in the App_Data subfolder of the current website.
// The System.IO.Path class makes it easy to build the full file name.
string file = Path.Combine(Request.PhysicalApplicationPath,
 @"App_Data\SuperProProductList.xml");
FileStreamfs = new FileStream(file, FileMode.Create);
XmlTextWriter w = new XmlTextWriter(fs, null);
w.WriteStartDocument();
w.WriteStartElement("SuperProProductList");
w.WriteComment("This file generated by the XmlTextWriter class.");
// Write the first product.
w.WriteStartElement("Product");
w.WriteAttributeString("ID", "1");
w.WriteAttributeString("Name", "Chair");
w.WriteStartElement("Price");
w.WriteString("49.33");
```

```
w.WriteEndElement();
w.WriteEndElement();
// Write the second product.
w.WriteStartElement("Product");
w.WriteAttributeString("ID", "2");
w.WriteAttributeString("Name", "Car");
w.WriteStartElement("Price");
w.WriteString("43399.55");
w.WriteEndElement();
w.WriteEndElement();
// Write the third product.
w.WriteStartElement("Product");
w.WriteAttributeString("ID", "3");
w.WriteAttributeString("Name", "Fresh Fruit Basket");
w.WriteStartElement("Price");
w.WriteString("49.99");
w.WriteEndElement();
w.WriteEndElement();
// Close the root element.
w.WriteEndElement();
w.WriteEndDocument();
w.Close();
```

Q2. What is XElement? Explain with an Example.

Answer.

The XElement class is one of the fundamental classes in LINQ to XML. It represents an XML element. You can use this class to create elements; change the content of the element; add, change, or delete child elements; add attributes to an element; or serialize the contents of an element in text form. You can also interoperate with other classes in System.Xml, such as XmlReader, XmlWriter, and XslCompiledTransform.
Example:

```
XElement xmlTree1 = new XElement("Root",

newXElement("Child1", 1),

newXElement("Child2", 2),

newXElement("Child3", 3),

newXElement("Child4", 4),

newXElement("Child5", 5),

newXElement("Child6", 6)
```

```
);
XElement xmlTree2 = new XElement("Root",
from el in xmlTree1.Elements()
where((int)el >= 3 && (int)el <= 5)
select el
);
Console.WriteLine(xmlTree2);
```

Output:

```
<Root>
<Child3>3</Child3>
<Child4>4</Child4>
<Child5>5</Child5>
</Root>
```

Q3. What do you mean by Authentication? Explain its types.

Authentication

Authentication is the process of verifying the identity of a user by obtaining some sort of credentials and using those credentials to verify the user's identity. If the credentials are valid, the authorization process starts. Authentication process always proceeds to Authorization process.

ASP.NET allows four types of authentications:

1. **Windows Authentication**
2. **Forms Authentication**
3. **Passport Authentication**
4. **Custom Authentication**

Windows-based authentication:
It causes the browser to display a login dialog box when the user attempts to access restricted page.
It is supported by most browsers.
It is configured through the IIS management console.
It uses windows user accounts and directory rights to grant access to restricted pages.

Forms-based authentication:
Developer codes a login form that gets the user name and password.
The username and password entered by user are encrypted if the login page uses a secure connection.
It doesn't reply on windows user account.
Windows Live ID authentication:
It is centralized authentication service offered by Microsoft.
The advantage is that the user only has one maintain one username and password.

Q4. What do you mean by Impersonation in ASP.NET? Explain.

Answer.

Impersonation is the process of executing code in the context of another user identity. By default, all ASP.NET code is executed using a fixed machine-specific account. To execute code using another identity we can use the built-in impersonation capabilities of ASP.NET. We can use a predefined user account or user's identity, if the user has already been authenticated using a windows account.

We can use the impersonation in this two scenarios:

1. To give each web application different permissions.
2. To use existing Windows user permission.

These two scenario are fundamentally different. In the first one, impersonation defines a single, specific account. In this case, no matter what user access the application, and no matter what type of user-level security you use, the code will run under the account you've set. In the second one, the user must be authenticated by IIS. The web-page code will then execute under the identity of the appropriate user.

Implement Impersonation:

Impersonate the Microsoft IIS Authenticated Account or User:

To impersonate the IIS authenticating user on every request for every page in an ASP.NET application, we must include an <identity> tag in the Web.config file of this application and set the impersonate attribute to true.

```
<identityimpersonate="true"/>
```

Impersonate a Specific User:

To impersonate a specific user for all the requests on all pages of an ASP.NET application, you can specify the userName and password attributes in the <identity> tag of the Web.config file for that application.

```
<identityimpersonate="true"userName="AccountNAME"password="Password"/>
```

Q5. Explain ASP.NET AJAX Control Toolkit.

Ajax Control Toolkit is an open source library for web development. The ASP.net Ajax Control toolkit contains highly rich web development controls for creating responsive and interactive AJAX enabled web applications.

ASP.Net Ajax Control Toolkit contains 40 + ready controls which is easy to use for fast productivity.

controls are like AutoComplete, Color Picker, Calendar, Watermark, Modal Popup Extender, Slideshow Extender and more of the useful controls.

The AJAX Control Toolkit contains more than 30 controls that enable you to easily create rich, interactive web pages.

Browser Support:

It supports all new, updated and stable versions of Firefox, Chrome, Opera, Apple Safari and internet Explorer > 8 Version.

Visual Studio Support:

It supports Visual studio version greater than 2010

System Requirements:

.Net Framework 4.0 or higher is required.

Visual Studio 2010 or higher with Editions enlisted in visual studio support.

Q6. Create a web application to demonstrate the use of HTMLEditorExtender Ajax Control.
Write the code of default.aspx and required property for the same.

Answer.
HTLEditorExtender Ajax control example:

1. Add the Script Manager Control on your page
2. Add a TextBox (columns = 80, rows =20 textmode = multiline)
3. Select that textbox control
4. Now add HtmlEditorExtender control from AJAX control toolkit.

Default.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" %>
<%@ Register assembly="AjaxControlToolkit" namespace="AjaxControlToolkit" 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 ID="TextBox1" runat="server" Columns="80" Rows="20" TextMode="MultiLine"></asp:TextBox>
<ajaxToolkit:HtmlEditorExtender ID="TextBox1_HtmlEditorExtender" runat="server" TargetControlID="TextBox1" EnableSanitization="false">
<Toolbar>
<ajaxToolkit:Undo /><ajaxToolkit:Redo />
<ajaxToolkit:Bold /><ajaxToolkit:Italic />
<ajaxToolkit:SelectAll /><ajaxToolkit:UnSelect />
<ajaxToolkit:Delete />
<ajaxToolkit:Copy />
<ajaxToolkit:Paste />
<ajaxToolkit:Cut />
</Toolbar>
</ajaxToolkit:HtmlEditorExtender>
</div>
```

Q7. Write about note on the XML Text Reader class.

The XmlTextReader moves through your document from top to bottom, one node at a time. You call the Read() method to move to the next node. This method returns true if there are more nodes to read or false once it has read the final node. The current node is provided through the properties of the XmlTextReader class, such as NodeType and Name.
A node is a designation that includes comments, whitespace, opening tags, closing tags, content, and even
the XML declaration at the top of your file. To get a quick understanding of nodes, you can use the XmlTextReader to run through your entire document from start to finish and display every node it encounters.
Example:

```
string file = Path.Combine(Request.PhysicalApplicationPath,
 @"App_Data\SuperProProductList.xml");
FileStreamfs = new FileStream(file, FileMode.Open);
XmlTextReader r = new XmlTextReader(fs);
// Use a StringWriter to build up a string of HTML that
// describes the information read from the XML document.
StringWriter writer = new StringWriter();
// Parse the file, and read each node.
while (r.Read())
{
 // Skip whitespace.
if (r.NodeType == XmlNodeType.Whitespace) continue;
writer.Write("<b>Type:</b> ");
writer.Write(r.NodeType.ToString());
writer.Write("<br>");
 // The name is available when reading the opening and closing tags
 // for an element. It's not available when reading the inner content.
if (r.Name != "")
 {
writer.Write("<b>Name:</b> ");
writer.Write(r.Name);
writer.Write("<br>");
 }
 // The value is when reading the inner content.
if (r.Value != "")
 {
writer.Write("<b>Value:</b> ");
writer.Write(r.Value);
writer.Write("<br>");
 }
if (r.AttributeCount> 0)
```

```
 {
writer.Write("<b>Attributes:</b> ");
for (int i = 0; i <r.AttributeCount; i++)
 {
writer.Write(" ");
writer.Write(r.GetAttribute(i));
writer.Write(" ");
 }
writer.Write("<br>");
 }
writer.Write("<br>");
}
fs.Close();
// Copy the string content into a label to display it.
lblXml.Text = writer.ToString();
```

Q8. Explain the Reading process from an XML Document with example.

Reading the XML document in your code is just as easy with the corresponding XmlTextReaderclass.

There are mainly two methods for reading XML with C#: The XmlDocument class and the XmlReader class. XmlDocument reads the entire XML content into memory and then lets you navigate back and forward in it as you please, or even query the document using the XPath technology.

The XmlReader is a faster and less memory-consuming alternative. It lets you run through the XML content one element at a time, while allowing you to look at the value, and then moves on to the next element. By doing so, it obviously consumes very little memory because it only holds the current element, and because you have to manually check the values, you will only get the relevant elements, making it very fast.

```
stringfile=Path.Combine(Request.PhysicalApplicationPath,
@"App_Data\SuperProProductList.xml");
FileStreamfs=newFileStream(file,FileMode.Open);XmlTextReaderr=ne
wXmlTextReader(fs);
//UseaStringWriter tobuildupa stringofHTMLthat
//describestheinformationreadfromtheXMLdocument.
StringWriterwriter=newStringWriter();
//Parsethefile,andreadeachnode.while(r.Red())
        {
```

```
//Skipwhitespace.
if(r.NodeType==XmlNodeType.Whitespace)continue;writer.
Write("<b>Type:</b>");writer.Write(r.NodeType.ToString());
writer.Write("<br>");
//Thenameisavailablewhenreadingtheopeningandclosingtags//for
anelement.It'snot availablewhenreadingtheinnercontent.
if(r.Name!="")
{
writer.Write("<b>Name:</b>");writer
.Write(r.Name);writer.Write("<br>");
}

//Thevalueiswhenreadingtheinnercontent.if(
r.Value!="")
{

writer.Write("<b>Value:</b>");writer.
Write(r.Value);writer.Write("<br>");
}

if(r.AttributeCount>0)
{

writer.Write("<b>Attributes:</b>");for(inti=0;i<r.AttributeCount;i++)
{

writer.Write("");writer.Write(r.GetAttribute(i));writer.Write("");
}

writer.Write("<br>");
}

writer.Write("<br>");
}

fs.Close();
//Copythestringcontentintoalabeltodisplayit.lblXml.Text=writer.ToString();
```

Q9. Define Authentication and Authorization. Give types of Authentication.

Authentication

Authentication is the process of verifying the identity of a user by obtaining some sort of credentials and using those credentials to verify the user's identity. If the credentials are valid, the authorization process starts. Authentication process always proceeds to Authorization process.

Example:

```
<authenticationmode="Forms">

    <formsloginUrl="LoginPage.aspx"protection="All"timeout="30">
```

```
        <credentialspasswordFormat="Clear">
        <username="admin"password="adminpwd"/>
        <username="coder"password="coderpwd"/>
        </credentials>
        </forms>
</authentication>
```

Authorization

Authorization is the process of allowing an authenticated user to access the resources by checking whether the user has access rights to the system. Authorization helps you to control access rights by granting or denying specific permissions to an authenticated user.

Example:

```
<authorization>

        < allowusers="admin"/>
        < denyusers="coder"/>
</authorization>
```

ASP.NET allows four types of authentications:

1. **Windows Authentication**
2. **Forms Authentication**
3. **Passport Authentication**
4. **Custom Authentication**

Windows-based authentication:

- It causes the browser to display a login dialog box when the user attempts to access restricted page.
- It is supported by most browsers.
- It is configured through the IIS management console.
- It uses windows user accounts and directory rights to grant access to restricted pages.

Forms-based authentication:

- Developer codes a login form that gets the user name and password.
- The username and password entered by user are encrypted if the login page uses a secure connection.
- It doesn't reply on windows user account.
- Windows Live ID authentication:
- It is centralized authentication service offered by Microsoft.

The advantage is that the user only has one maintain one username and password.

Q10. What is Windows Authentication? Give details.

Windows authentication:

Windows-based authentication is manipulated between the Windows server and the client machine.

The ASP.NET applications resides in Internet Information Server (IIS). Any user's web request goes directly to the IIS server and it provides the authentication process in a Windows-based authentication model.

- When you configure your ASP.NET application as windows authentication it will use local windows user and groups to do authentication and authorization for your ASP.NET pages.
- In 'web.config' file set the authentication mode to 'Windows' as shown in the below code snippets.

```
<authentication mode="Windows"/>
```

- We also need to ensure that all users are denied except authorized users. The below code snippet inside the authorization tag that all users are denied. '?' indicates any

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

- We also need to specify the authorization part. We need to insert the below snippet in the 'web.config' file .stating that only 'Administrator' users will have access to

```
<location path="Admin.aspx">
<system.web>
<authorization>
<allow roles="questpon-srize2\Administrator"/>
<deny users="*"/>
</authorization>
```

Advantage of Windows Authentication

- It relies and allows the user to use existing Windows Accounts.
- Establishes the foundation for Uniform Authentication model for multiple types of applications.
- From developer's point of view, easy to implement.

Disadvantage of Windows Authentication

- Applicable to Microsoft platforms only.

- No custom control over this platform provided authentication process.

Q11. Explain AJAX with its Advantages and Disadvantages.

- Ajax, which stands for Asynchronous JavaScript And XML, enables your client- side web pages to exchange data with the server through asynchronous calls.
- Probably the most popular feature driven by Ajax is the flicker-free page that enables you to perform a postback to the server without refreshing the entire page.
- Although a number of different Ajax frameworks are available for ASP.NET, the most obvious one is Microsoft ASF.NETAJAX, because it comes fully integrated with the.NET 4 Framework and Visual Web Developer 2010.

Advantages of AJAX

- Reduce the traffic travels between the client and the server.
- Response time is faster so increases performance and speed.
- You can use JSON (JavaScript Object Notation) which is alternative to XML. JSON is key value pair and works like an array.
- Ready Open source JavaScript libraries available for use – JQuery, etc.
- AJAX communicates over HTTP Protocol.

Disadvantages of AJAX

- It can increase design and development time
- More complex than building classic web application
- Security is less in AJAX application as all files are downloaded at client side.
- Search Engine like Google cannot index AJAX pages.
- JavaScript disabled browsers cannot use the application.
- Due to security constraints, you can only use it to access information from the host that served the initial page. If you need to display information from another server, it is not possible within the AJAX.

Q12. Give Brief information on Accordion control with appropriate properties.

The accordion is a graphical control element comprising a vertically stacked list of items, such as labels or thumbnails. Each item can be "expanded" or "collapsed" to reveal the content associated with that item. There can be zero expanded items, exactly one, or more than one item expanded at a time, depending on the configuration.

The term stems from the musical accordion in which sections of the bellows can be expanded by pulling outward.

Accordion Web Control contains multiple panes and shows one at a time. The panes contain content of images and text having wide range of supported properties from formatting of the panes and text, like header and content templates, also it formats the selected header template.

Accordion contains multiple collapsible panes while one can be shown at a time and the extended features of accordion also has support data source which points to the DataSourceID and binds through .DataBind() method and also you need to specify the Header and Content template to show the data

Accordion is also an extender control which can be placed in a container like div and so on. Extender controls are those controls which extend the functionality of your current work just as an example you have lot of paragraphs in a single page and you don't have much space and want to summarize in small space then you need to embed to the div or some container which can hold all the information in accordion collapsible pans.

A common example of an accordion is the Show/Hide operation of a box region, but extended to have multiple sections in a list.

An accordion is similar in purpose to a tabbed interface, a list of items where exactly one item is expanded into a panel (i.e. list items are shortcuts to access separate panels)

General properties and set the following parameters:

- Name: Enter the name of the Control. The name must be unique on the window.
- Caption: Enter the caption for the Control.
- Status Text: Enter a tip to display in the browser's status bar when the mouse pointer is over the control. You can use {} syntax to substitute values from controls here.
- Please note: the user must have their status bar visible for this to be shown.
- Tooltip text: Enter the text to display when the mouse pointer is held over the control for more than a second.
- Data Source: Select An XPath statement that defines the data element for this control. By default it is set to ${windowelement} - the Data Source used by the window the control is on.
- Enable: Choose whether the panel is enabled (Y) or not (N). Default is Y.
- Visible: Choose whether the panel is visible (Y) or not (N). Default is Y.