CS-32: Programming with ASP.NET

UNIT – 5
Working with XML ASP.NET Application Configuration and Deployment of Application

Dharmesh Kapadiya
Dhruvita Savaliya

TOPICS :

- Reading Datasets From XML
- Writing DataSets With XML
- WebServices (Introduction, HTTP, SOAP, UDDI, XML, Creating a Web Service, Consuming a Web Service)
- Introduction To Web.Config
- Common Configuration Sections
- AppSettings
- Tracing
- Custom Errors
- Authentication And Authorization
- Deployment of Application in web server

XML BASICS :

- XML is a cross-platform, hardware and software independent, text based markup language, which enables you to store data in a structured format by using meaningful tags.
- XML stores structured data in XML documents that are similar to databases.
- Notice that unlike Databases, XML documents store data in the form of plain text, which can be used across platforms.
- You design your own customized markup languages for many types of documents.
- It is a subset of Standard Generallzed Markup Language (SGML), which is a more generalized form of markup language definition.

- In an XML document, you specify the structure of the data by creating a DTD (Document type definition) or an XML schema.
- When you include a DTD in an XML document, the software checks the structure of the XML document against the DTD.
- This process of checking the structure of the XML document is called *validating.*
- *The software performing the task of validating is called a validating parser.*
- When you define your XML language, you define the names of the attributes and elements, the order in which elements appear, and the way elements are nested.
- These definitions are written in a document called a **Document Type Declaration (DTD) or an XML** schema.
- An XML document that is well formed and follows the rules in the corresponding DTD or XML schema is said to be valid.
- XML documents are usually **read and processed by a parser.**
- This may be a piece of software, or, in the case of .NET, a series of classes that are programmed to load, read, and manipulate the XML document.

- **Syntax of writing data to XML File :**

```
<root>
      <child>
            <subchild>.....</subchild>
      </child>
</root>
```

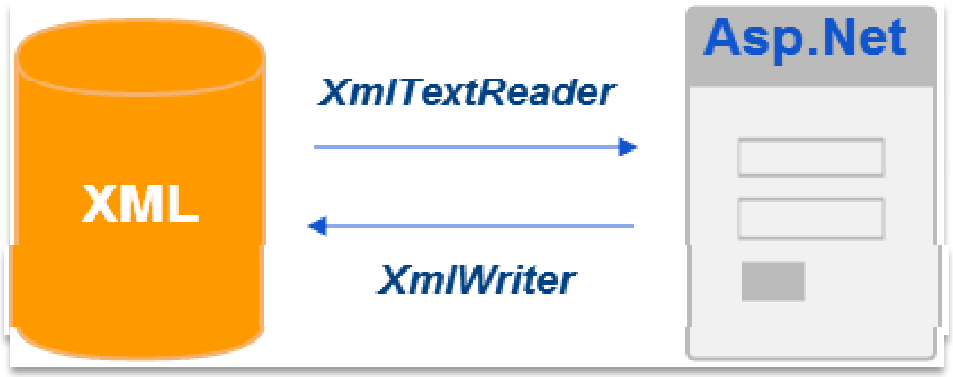
- **XML FILE :**

```
<?xml version="1.0"
   standalone="yes"?>
<NewDataSet>
 <Table>
   <empid>1</empid>
   <empnm>Raj</empnm>
   <empcity>Amreli</empcity>
    <empmobile>9526013452
    </empmobile>
 </Table>
 <Table>
   <empid>2</empid>
   <empnm>Jatin</empnm>
   <empcity>Babra</empcity>
   <empmobile>9876543212
    </empmobile>
 </Table>
</NewDataSet>
```

Note : XML can have a file called DTD or XSD.
Both of these files are used to specify rules for XML data.

DTD	**XSD**
DTD stands for **Document Type Definition**.	XSD stands for XML Schema Definition.
DTDs are derived from **SGML** syntax.	XSDs are written in XML.
DTD **doesn't support datatypes**.	XSD **supports datatypes** for elements and attributes.
DTD **doesn't support namespace**.	XSD **supports namespace**.
DTD **doesn't define order** for child elements.	XSD **defines order** for child elements.
DTD is **not extensible**.	XSD is **extensible**.
DTD is **not simple to learn**.	XSD is **simple to learn** because you don't need to learn new language.
DTD provides **less control** on XML structure.	XSD provides **more control** on XML structure.

- **XML PARSER**
- XML parser is software which checks whether our coding of XML file and its tags, attributes, etc. are proper or not.
- All modern browsers have a built-in XML parser.
- XML parser converts XML data and displays in the browser.
- Common XML parsers are REXML, MSXML, FastXML, etc.



READING AND WRITING XML FILE :

- The namespace System.xml provides the basic support for processing XML in Asp.Net.
- **XmlWriter :**
- The XmlWriter class is a base class that provides a forwardonly, read-only process to generate XML stream.
- It provides methods to write data in an XML document.
- since in our example, we are using various methods of this class to write data and it is simple.
- **XmlTextReader :**
- The XmlTextReader is the next class we will use to read the data from an XML document.
- This class has some very nice properties and methods.

Reading Datasets From XML :

- ADO.NET provides simple methods for working with XML data.
- The ReadXml method provides a way to read either data only, or both data and schema into a DataSet from an XML document, whereas the ReadXmlSchema method reads only the schema. To read both data and schema, use one of the ReadXML overloads that includes the mode parameter, and set its value to ReadSchema.
- In this walkthrough, you create a Windows application that loads XML data into a dataset. The dataset is then displayed in a DataGridView control. Finally, an XML Schema based on the contents of the XML file is displayed in a text box.
- For More(Reading Datasets From XML) :
- https://learn.microsoft.com/en-us/visualstudio/data-tools/read-xml-data-into-a-dataset?view=vs-2022&tabs=csharp

File.aspx :

```
<form id="form1" runat="server">
   <div>
      <asp:GridView ID="gvCustomers" runat="server" AutoGenerateColumns="false">
         <Columns>
            <asp:BoundField DataField="customerId" HeaderText="customerId" />
            <asp:BoundField DataField="name" HeaderText="name" />
            <asp:BoundField DataField="country" HeaderText="country" />
         </Columns>
      </asp:GridView>
   </div>
   </form>
```

File.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
   {
      if(!this.IsPostBack)
      {
         DataSet ds=new DataSet();

   ds.ReadXml(Server.MapPath("/xml Files/Customers.xml"));
         gvCustomers.DataSource = ds;
         gvCustomers.DataBind();
      }
   }
```

Writing DataSets With XML :

- In ADO.NET you can write an XML representation of a DataSet, with or without its schema. If schema information is included inline with the XML, it is written using the XML Schema definition language (XSD). The schema contains the table definitions of the DataSet as well as the relation and constraint definitions.
- When a DataSet is written as XML data, the rows in the DataSet are written in their current versions. However, the DataSet can also be written as a DiffGram so that both the current and the original values of the rows will be included.
- The XML representation of the DataSet can be written to a file, a stream, an **XmlWriter**, or a string. These choices provide great flexibility for how you transport the XML representation of the DataSet.

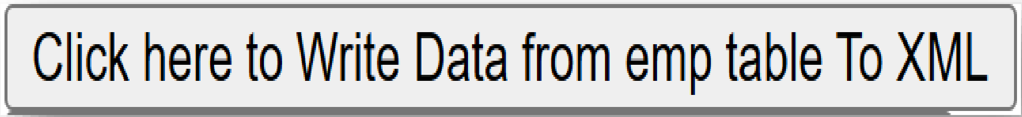
File.aspx :

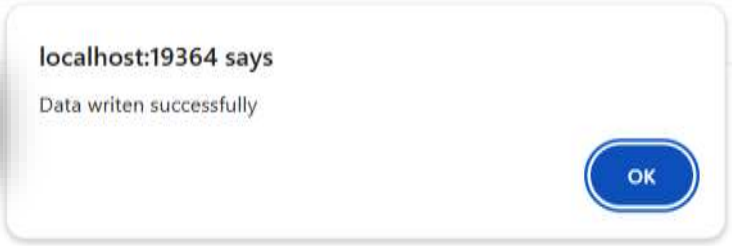
```
<asp:Button ID="btnwritedata" runat="server" Text="Click here to
  Write Data from emp table To XML" OnClick="btnwritedata_Click" />
```

File.aspx.cs :

```
protected void btnwritedata_Click(object sender, EventArgs e)
  {
     con = new
  SqlConnection(ConfigurationManager.ConnectionStrings["constr"].Co
  nnectionString);
     SqlDataAdapter da = new SqlDataAdapter("select * from emp", con);
     DataSet ds = new DataSet();
     da.Fill(ds);
    ds.WriteXml(MapPath("/xmlFiles/empData.xml"));
     Response.Write("<script>alert('Data writen
  successfully');</script>");
  }
```

- Now Click on this button :




localhost:19364 says

Data writen successfully

Now, Refresh empData.xml file.

```
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
 <Table>
  <empid>1</empid>
  <empnm>ram</empnm>
  <empcity>amreli</empcity>
  <empmobile>9123456789</empmobile>
 </Table>
 <Table>
  <empid>2</empid>
  <empnm>shyam</empnm>
  <empcity>surat</empcity>
  <empmobile>981234674 </empmobile>
 </Table></NewDataSet>
```

- **Next Example :**
- Dynamic Insert record into a table.
- Write data into a XML file.
- Read data from XML file.

File.aspx :

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

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
</head>
<body>
  <form id="form1" runat="server">
  <div>
    Emp Name :<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox><br />
    Emp City :<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox><br />
    Emp Mobile :<asp:TextBox ID="TextBox3" runat="server"></asp:TextBox><br />
   <asp:Button ID="insert_record" runat="server" Text="Click here to Insert a record into emp table"
    OnClick="insert_record_Click" />
   <asp:Button ID="btnwritedata" runat="server" Text="Click here to Write Data from emp table To XML"
    OnClick="btnwritedata_Click" />
    <asp:Button ID="btnreadData" runat="server" Text="Click here to Read Data From Emp XML File"
    OnClick="btnreadData_Click"/>
  </div>
    <asp:GridView ID="grdEmp" runat="server"></asp:GridView>
  </form>
</body>
</html>
```

File.aspx.cs :

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;

public partial class readwriteXMLwithDatabase :
    System.Web.UI.Page
{
   SqlConnection con;
   SqlCommand cmd = new SqlCommand();
  protected void btnwritedata_Click(object sender,
    EventArgs e)
   {
     con = new
    SqlConnection(ConfigurationManager.Connection
    Strings["constr"].ConnectionString);
     SqlDataAdapter da = new SqlDataAdapter("select
    * from emp", con);
     DataSet ds = new DataSet();
     da.Fill(ds);
    ds.WriteXml(MapPath("/xmlFiles/empData.xml
    "));
     Response.Write("<script>alert('Data writen
    successfully');</script>");
   }
   protected void btnreadData_Click(object sender,
    EventArgs e)
   {
     DataSet ds = new DataSet();

    ds.ReadXml(MapPath("/xmlFiles/empData.xml"))
    ;
     grdEmp.DataSource = ds;
     grdEmp.DataBind();
   }
   protected void insert_record_Click(object sender,
    EventArgs e)
   {
     con = new
    SqlConnection(ConfigurationManager.Connection
    Strings["constr"].ConnectionString);
     SqlDataAdapter da = new SqlDataAdapter("insert
    into emp (empnm,empcity,empmobile)
    values('"+TextBox1.Text+"','"+TextBox2.Text+"','"+
    TextBox3.Text+"')", con);
     DataSet ds = new DataSet();
     da.Fill(ds);
   }
}
```

WEB SERVICE :

- Web Service means one website requests to other website for providing particular service.
- Web Service is an application that is designed to interact directly with other applications over the internet.
- Web services are small units of code and independent of operations system as well as programming language.
- Web service client is an application which uses a Web service.
- Once Web service is developed it uploads to the server then it is used by **web methods**.
- For example, Yahoo provides web service of whether forecasting, news updates, stock market status, etc. We can call these services by calling methods of it and display the information in our website.
- Some website provides paid webservice.
- Main advantage of Web service is Web service messages are formatted as XML, a standard way for communication between two incompatible systems. And this message is sent via HTTP so that they can reach to any machine on the internet without being blocked by firewall.

- **Components of Web Service :**
- **HTTP :** Hyper Text Transfer Protocol
- **XML :** Extensible Markup Language
- **SOAP :** Simple Object Access Protocol
- **WSDL :** Web Services Description Language
- **UDDI :** Universal Description Discovery and Integration

XML (Extensible Markup Language) :

- The basic thing that works behind any web service is XML and HTTP.
- HTTP is a standard protocol which is used for sharing information over the internet.
- XML is a standard language which is used for sharing information among different platforms as it is platform independent.
- Web method that we create are same like normal methods which have arguments and which also returns values.
- These arguments and return values are given in form of XML which is platform independent.
- As the data is shared via XML thus it makes the data also platform independent.
- XML provides standard data representation format which can be understood by any platform.

SOAP (Simple Object Access Protocol) :

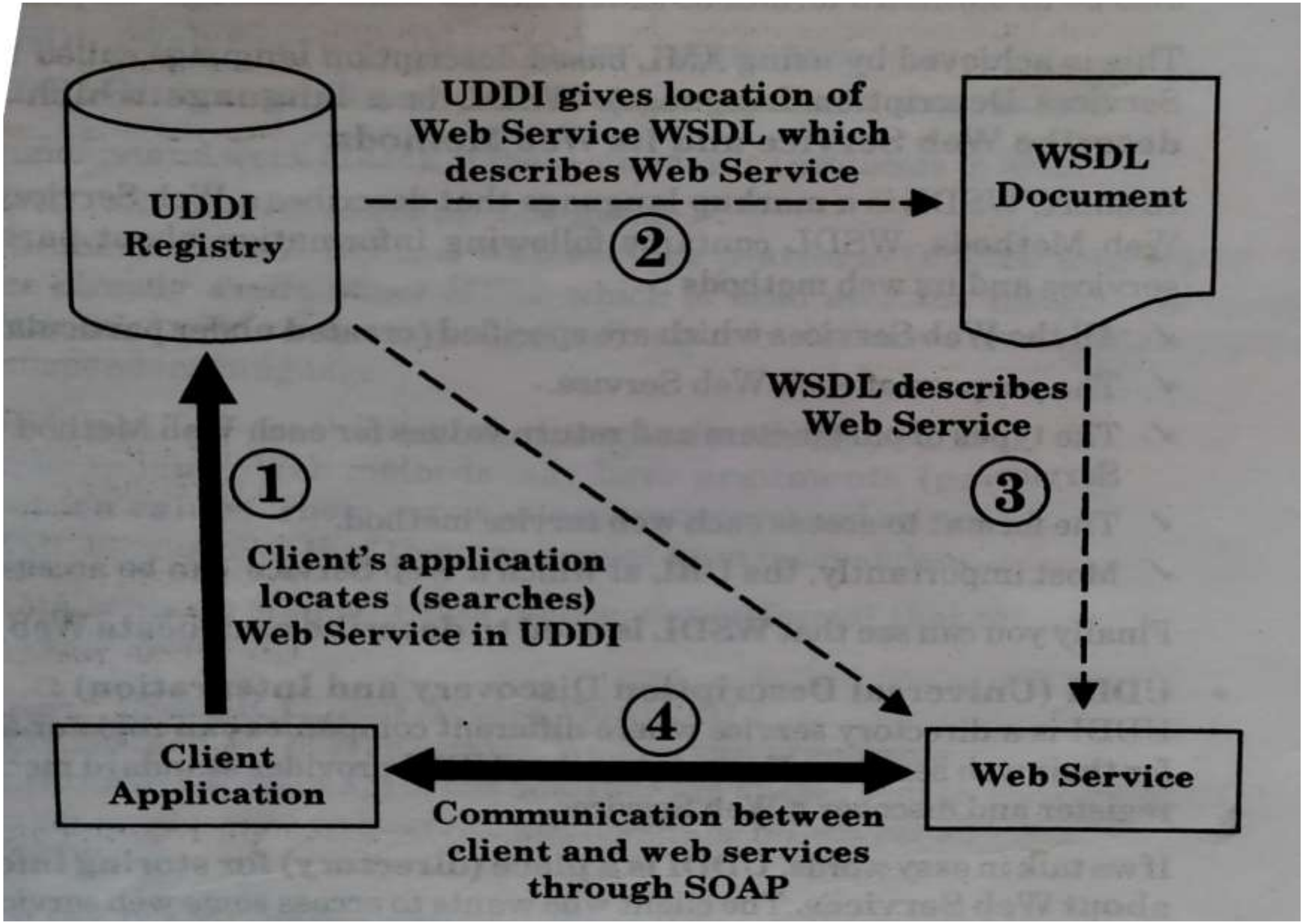
- SOAP stands for Simple Object Access Protocol.
- SOAP is an XML-based protocol for accessing web services over HTTP.
- SOAP is responsible to transfer XML data from one location to another location.
- SOAP is a standard communication protocol for interchanging information in a structured format in distributed environment.
- When SOAP client makes request for Web method SOAP packet is created. This packet contains name of the Web method and list of parameters to be passed to Web method in as XML format.
- When SOAP packet reaches at the Web server where Web service exists, the information of SOAP packet is extract. After this Web method is invoked and again return value is return back to the client using SOAP packet.
- SOAP is a communication protocol designed communicate via internet.
- SOAP provides data transport for Web services.
- SOAP can be used for broadcasting a message.
- SOAP is platform and language independent.
- SOAP defines a uniform way of passing XML-encoded data.

WSDL (Web Services Description Language) :

- WSDL stands for Web Services Description Language.
- To call Web methods the developers must be known information regarding Web methods like parameters and return value of Method.
- Web services are platform independent.
- So explanation of Web method should be in standard format so that it can be shared amongst all platforms.
- This can be done by using WSDL. WSDL is a language which is used to describe Web service and its Web methods.
- WSDL is an XML-based language used to define web services and to describe how to access them.
- WSDL is an xml document.
- WSDL contains every details regarding using web service.
- WSDL specifies the location, purpose, methods and format of the methods of the Web service.
- It contains tags like <types>, <message>, <portType> and <binding> to specify Web Service.

UDDI (Universal Description, Discovery and Integration) :

- UDDI stands for Universal Description, Discovery and Integration.
- UDDI is a directory service where businesses can register and search for web services.
- UDDI is a directory for storing information about Web services.
- UDDI allows you to find web services by connecting to a directory.
- A description of the selected Web service is returned to the client application as a WSDL file.
- UDDI is platform independent, open framework.
- UDDI can communicate via SOAP, CORBA, Java RMI Protocol.
- UDDI is built into the Microsoft .NET platform.


UDDI Registry
UDDI gives location of Web Service WSDL which describes Web Service
2
WSDL Document
WSDL describes Web Service
3
1
Client's application locates (searches) Web Service in UDDI
4
Client Application
Web Service
Communication between client and web services through SOAP

- Following is the detailed description of above diagram:[

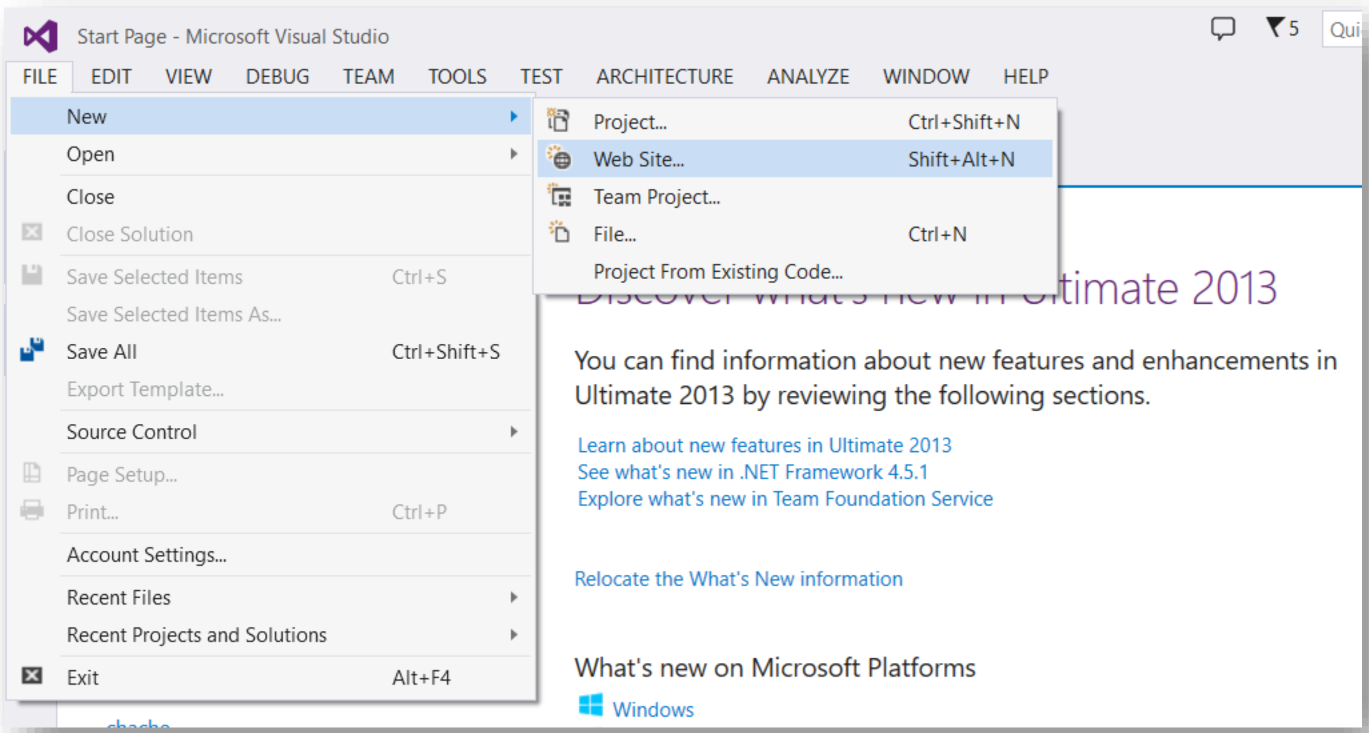
1. First of all client application searches for the web application. We have already discussed that after creating Web Services, they are registered under UDDI. So when any client's application wants to access any web service, it searches for Web service under UDDI.
2. If Web Service is registered, UDDI gives its location in a format of WSDL document. We have already discussed that WSDL is XML based descriptor which describes web application. Location of WSDL is given by UDDI.
3. WSDL describes Web Service and its embedded Web Methods. WSDL describes argument list (parameters) and return values of each web method within web service. So that client can get the list of each of web methods.
4. We discussed that UDDI gives location of Web Service and description in form of WSDL. Now finally when client gets physical location of WSDL, client can access Web Service via SOAP. SOAP is a communication protocol which allows you to access Web Service and its Web Methods.

- **Namespaces used for Web Services:**
- There are two important namespaces which are used for creating and consuming Web Services.

1. System.Web.ServicesSystem
2. Web.Services.Protocol

- **System.Web.Services:**
- If you want to create web service, your class must inherit WebService class. System.Web.Services is a namespace which has "WebService" class. It allows you to inherit WebService class in your class.
- **System.Web.Services.Protocol:**
- We have already discussed that different protocols as SOAP, etc. works behind Web Service. This namespace allows you to utilize such protocols in order to use Web Service defined.
- **Web Service related files:**There are two types of files that work along with Web Services. They are,
- **.asmx file (Web Service Class):** This is the class file for Web Service. Web Service classes are stored as asmx file. This file is the entry point to any Web Service.
- **.asmx.cs/.asmx.vb file (Web Service Code Behind File):** This is the code behind file for Web Service and its Web methods. Web Services contain Web Method which will have some code.

Step 1 : open visual studio 2013.



Ultimate 2013

You can find information about new features and enhancements in Ultimate 2013 by reviewing the following sections.

Learn about new features in Ultimate 2013
See what's new in .NET Framework 4.5.1
Explore what's new in Team Foundation Service

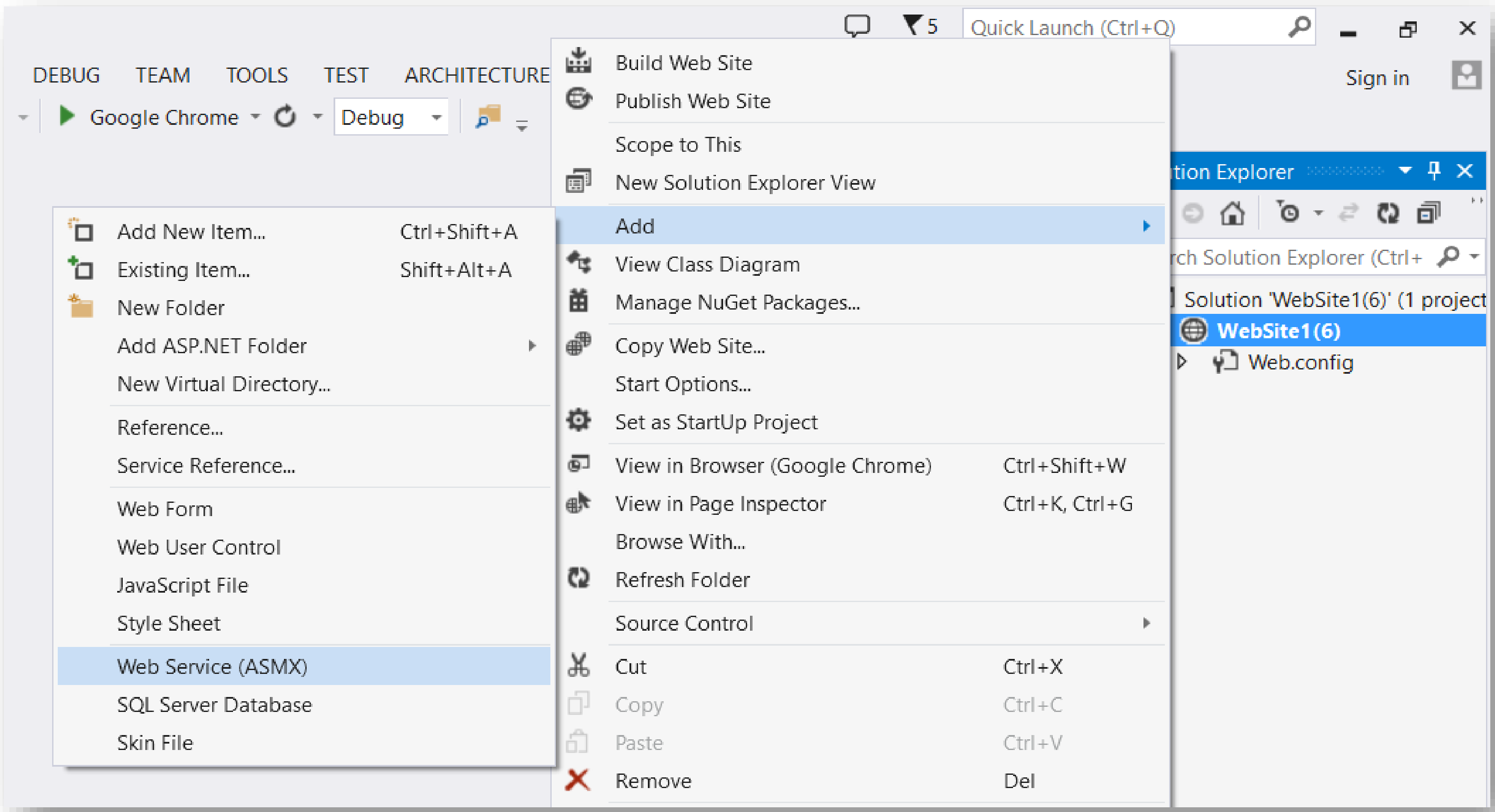
Relocate the What's New information

What's new on Microsoft Platforms

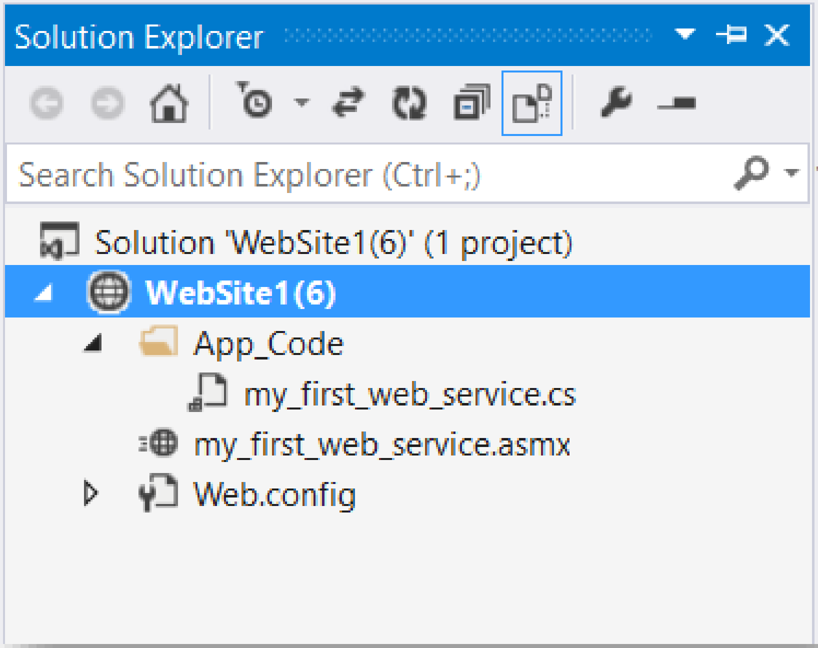
Windows

Step 2 : Then , Templates → visual C# → ASP.NET Empty Web Site

Strep 3 : Right click on your project → Add → **Web Service (ASMX)** or you can also find Web Service in Add New Item



- **Step 3 :** check project :
- It Will add this types of directory :
- App_Code for all .cs file of web services
- Out side of this folder you can find .asmx file



```
Solution 'WebSite1(6)' (1 project)
◢ WebSite1(6)
    ◢ App_Code
        my_first_web_service.cs
      my_first_web_service.asmx
    ▷ Web.config
```

Step 4 : add code in my_first_web_service.cs code

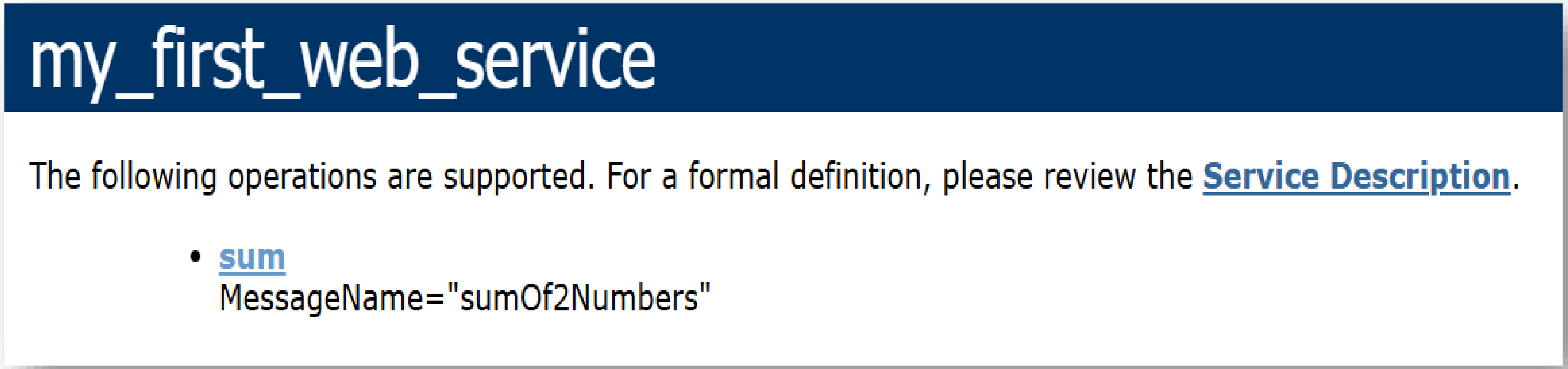
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;
/// <summary>
/// Summary description for my_first_web_service
/// </summary>
[WebService(Namespace = "http://demo.org/")]        //http://tempuri.org  by default
    namespace
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
// To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.
// [System.Web.Script.Services.ScriptService]
public class my_first_web_service : System.Web.Services.WebService {
   public my_first_web_service () {

      //Uncomment the following line if using designed components
      //InitializeComponent();
   }

    [WebMethod(MessageName="sumOf2Numbers")]
    public int sum(int a,int b) {
       return a+b;
    }

}
```

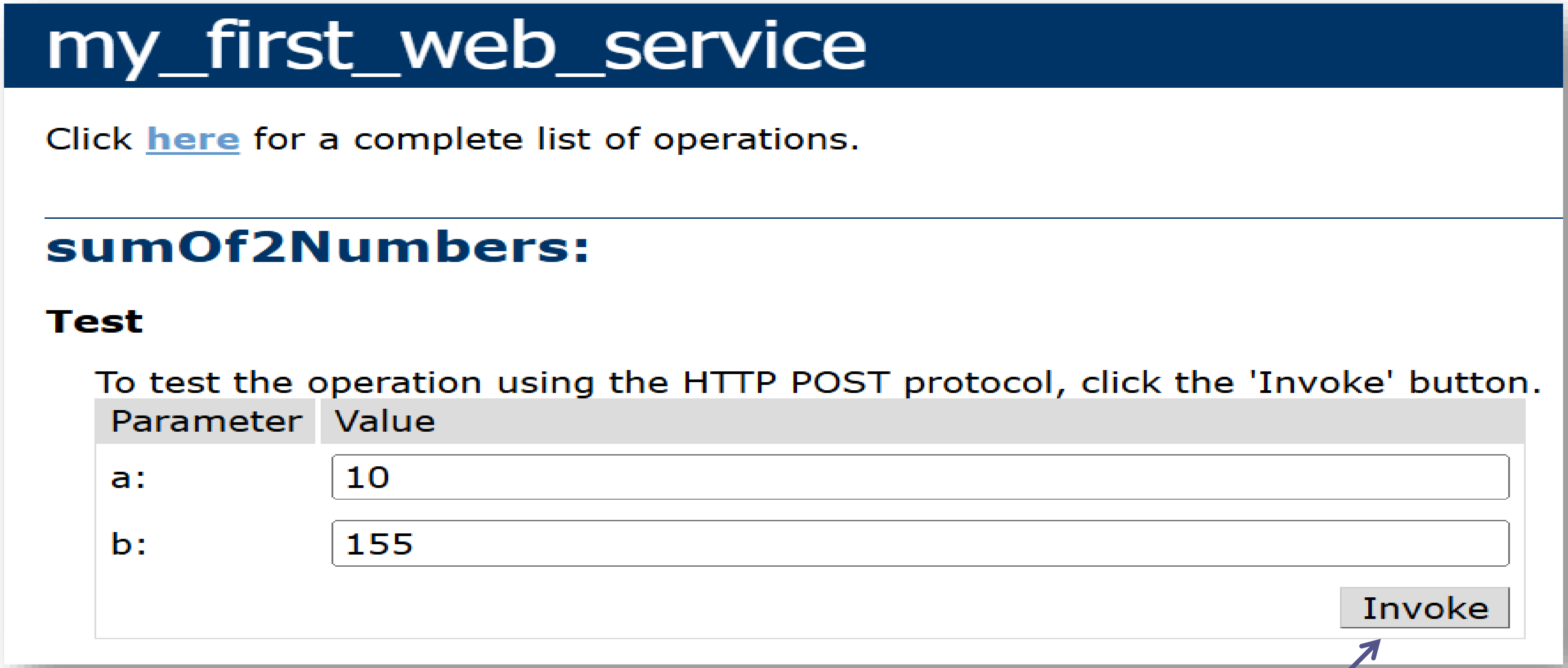
- **Step 5 :** run your web service.


my_first_web_service

The following operations are supported. For a formal definition, please review the Service Description.

- sum
  MessageName="sumOf2Numbers"

- **Step 6 :** Add parameter value.



my_first_web_service

Click here for a complete list of operations.

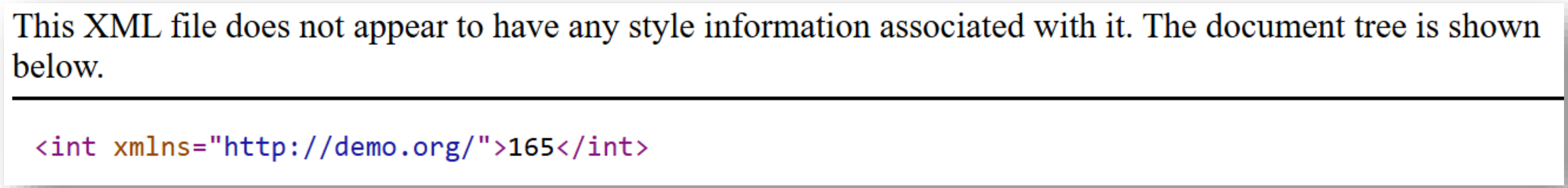
sumOf2Numbers:

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
a:	10
b:	155

- **Step 7 :** click on Invoke button to check result



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<int xmlns="http://demo.org/">165</int>
```

Select Name from database :

```
[WebMethod]
   public string serch_name(int id)
   {
      SqlConnection con = new SqlConnection(@"Data
   Source=(LocalDB)\v11.0;AttachDbFilename=E:\KAMANI\TY\ASP.NET
   2024-25\UNIT - 5\WebSite1\App_Data\Database.mdf;Integrated
   Security=True");
      SqlCommand cmd = new SqlCommand("select empnm from emp
   where empid='"+id+"'",con);
      con.Open();
      object name = cmd.ExecuteScalar();
      con.Close();
      if (name == null)
         return "No data found";
      else
         return name.ToString();
   }
```

DEPLOYMENT OF APPLICATION or Configuration files :

- The behavior of an ASP.Net application is affected by different settings in the configuration files:

1. **Machine.config**
2. **Web.config.**

- The machine.config file contains default and the machine-specific value for all supported settings. The machine settings are controlled by the system administrator and applications are generally not given access to this file.
- An application however, can override the default values by creating web.config file for each folder.
- Visual Studio generates a default web.config file for each project.
- Every machine which has ASP.NET installed, they has one *machine.config file that is created in "C:\WINDOWS\Microsoft.NET\Framework\[Version]\CONFIG", which mainly defines:*
  - Supported configuration file sections,
  - the ASP.NET worker process configuration, and
  - registers different providers that are used for advanced features such as profiles, membership, and role based security.

WEB.CONFIG FILE :

- Each and every ASP.Net application has its own copy of configuration settings stored in a file called Web.Config.
- The XML based Web.Config file is used to specify the application wide settings for your entire application.
- The Web.config file is basically XML-based text file.
- There can be multiple config files in your application.
- If the web application spans multiple folders, each sub folder has its own Web.Config file that inherits or overrides the parent's file settings.
- The settings in a file apply to the directory in which it is located and all child directories.
- Web.Config file contains application wide data like custom setting and error messages, database connection string, user authentication and authorization, web page language options, debug options.
- The root element of **Web.Config is <configuration>.**

- <configuration> section consists of many other sub elements which are :
- **<configSections> :** This allows you to configure different sections of web application like JavaScript, Web Services, Resources etc.
- **<appSettings> :** This allows you to add any application related variable or settings. You can <add>, <remove> or <clear> any application specific variable or settings.
- **<connectionStrings> :** This allows you to create connectionString variables which can be used in any Web Form of your web application.
- **<system.web> :** This is important element used to configure your web application. This section allows you to do most of configuration as follows:

<compilation>	<authentication>	<caching>
<customErrors>	<deployment>	<roleManager>
<trace>	<sessionState>	<webServices>

- **<system.webServer> :** This section stores information and configuration settings for web servers which will be used for your deploying and running your web application.
- **<runtime> :** This section is used to specify some run time settings for your web application.

Configuration Settings :

- **Tracing Web Application:**
- Tracing allows you to check for any type of bugs that your web application may have. Along with tracing your web application, you can also maintain log files which give you some additional information. These log files can help you to find out some problems that occur during web application usage.
- **Customizing Errors:**
- While using web site, sometimes different error occur. You are not aware that which error is going to come. Web Site users feel uncomfortable when they get some unexpected screen. By customizing errors, you can redirect your error to specific page which can display error but in your design specific way.

- **Authentication and Authorization:**
- Authenticated users are requirement for most of the web sites. To allow authenticated users, you must authorize users in such a way that any even any unauthorized user visits your web site, should not be able to harm your information. This can be done with the help of combining Authentication and Authorization.
- **Enabling Role Manager:**
- Enabling Role Manager is a feature which is used along with Authentication and Authorization. This feature can be used to allow some specific users. For example, some of the web page of your website are important, you want only "Administrator" level user can access. "Guest" users should not be able to access these pages.
- **Session Configuration:**
- Session Configuration allows you to specify session settings like Timeout Period, Session Mode, Cookie Based Session or CookielessSession, SqlCommandTimeout, NetworkTimeout, etc.

- **Trust Levels:**
- There Can be different types of web site Users. For example, Web master will have access to all the web pages of web site. Web Developer will have some limited access to web pages of Website; Administrator will have access to all the back end pages for web site administration where at last level is Users who will have access to only those pages which are informative. So different types of trust levels can be defined using trust levels.
- **Web Service Configuration :**
- Web services are configured to lock some resources from being accessed by some specific users.
- **Caching:**
- Caching in short means, keeping your output or data into cache memory so that it can be accessed in a faster way instead of gathering output from web Server or searching data from Database.

WHAT IS TRACING?

- **Tracing is a process that monitors the execution route** and shows diagnostic data about a particular application or web page running on the web server.
- Both the production and the development environment can have tracing enabled.
- These details might assist you in looking into faults or undesirable outcomes that occur while ASP.NET processes a page request.
- When trace is activated, you have the option to utilize the trace viewer, which allows you to inspect the trace information collected and cached by ASP.NET at the bottom of each page.

- **FEATURES OF TRACING**
- There are many features of tracing in .NET. Some of them are mentioned below.
- With the help of tracing, the most recent data trace is visible.
- We have programmatic access to and control over trace messages.
- The debug statement allows us to view the web page and the application's execution route.
- **ADVANTAGE OF TRACING**
- Application development, debugging, and performance optimization all heavily rely on tracing in .NET.
- This leads to better code quality, performance, and general application stability by giving developers insightful knowledge of the behavior of their applications during runtime.

TYPES OF TRACING :

- There are two types of tracing in ASP.NET.

1. **Page level tracing**
2. **Application-level tracing**

- **Page level tracing**
- In ASP.NET, **page level tracing allows developers to gather** and analyze diagnostic information on a specific web page
- during its execution. It offers thorough insights into the page's data values, control events, and execution flow.
- Certain modifications need to be implemented to enable pagelevel tracing in your web application using ASP.NET:

  <%@ Page Trace="true" %>

- You can also optimize the trace output by changing the attribute of the @Page directive.

- **<%@ Page Trace="true" TraceMode="SortByTime" TracePageOutput="true" %>**
- **TraceMode:** Specifies the order in which the trace messages appear, e.g., SortByTime, SortByCategory, SortByDuration.
- **TracePageOutput:** Determines whether the trace output is displayed at the bottom of the page.
- Page-level tracing is a useful tool for analyzing and improving the behavior of certain ASP.NET pages.
- It's crucial to keep in mind that activating tracing in a production environment might have an impact on performance; for this reason, it's normally advised to utilize it only in development or debugging settings.

- **Example Page level tracing:**
- **Default.aspx :**

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

- **Default.aspx.cs :**

```
 protected void Page_Load(object sender, EventArgs e)
{
                    Trace.Write("hello");
                    Trace.Warn("Sorry");
 }
```

APPLICATION-LEVEL TRACING :

- The ASP.NET capability, known as **application-level tracing, enables programmers to gather and examine** diagnostic data for an entire web-based application. It gives developers insights into how the program behaves throughout several pages, allowing them to monitor the execution flow, spot problems, and gather performance data.
- Certain modifications need to be implemented to enable application-level tracing in your web application using ASP.NET:
- Navigate the Web.config file in your ASP.NET application.
- Find the <**system.web> in the Web.config file.**
- To enable tracing, insert the following configuration elements within the <system.web> section**:**

<trace enabled="true"/>

- **<system.web> <trace enabled="true" requestLimit="40”** pageOutput="false" traceMode="SortByTime“ localOnly="true"/> **</system.web>**
- **enabled:** Set it to true to enable tracing for the application.
- **requestLimit:** Defines the upper limit for the number of trace entries to be stored per request.
- **pageOutput:** Set it to false to prevent trace output from being displayed at the bottom of each page.
- **traceMode:** Determines the order in which the trace messages are displayed. Common values include SortByTime, SortByCategory, or SortByDuration.
- **localOnly:** Enable this setting to restrict tracing information visibility solely to the server by setting it to true.
- Application-level tracing proves to be a highly effective technique for evaluating the overall behavior and performance of an ASP.NET application.
- It assists in following the execution path, finding problems affecting numerous pages, and learning about the behavior of the program during runtime.
- However, due to potential performance effects and security issues, tracing should only be used sparingly and should not be enabled in production systems.

- **web.config :**

```
<configuration>
   <system.web>
     <compilation debug="true" targetFramework="4.5" />
     <httpRuntime targetFramework="4.5" />
          <trace enabled="true"/>
   </system.web>
</configuration>
```

- **Default.aspx.cs :**

```
    protected void Page_Load(object sender, EventArgs e)
  {
                    Trace.Write("hello");
                    Trace.Warn("Sorry");
   }
```

- Here by default pageOutput=“false”
- That’s why if you want to check tracing then you can write into your URL : **localhost:port_number/trace.axd**

CUSTOM ERRORS IN ASP.NET :

- When a run-time or design-time error occurs in an application, ASP.NET shows a default Yellowish-Red error page that gives a brief description of the error along with some status code.
- If this page is viewed by the developer, then he can analysis the actual reason by its status code, but if a user or anyone other than the developer is viewing this error page, they won't get the actual idea of why this error is coming.
- So to provide a friendly error message page we need to set custom error in web.config.

- SOME OF THE BASIC COMMON ERRORS WITH STATUS CODES ARE:
- **Error Code Description :**
- **400** Bad Request The request cannot be fulfilled due to bad Syntax.
- **403** Forbidden The request was a legal request, but the server is refusing to respond to it.
- **404** Not Found Page Not Found
- **405** Method not allowed A request was made of a page using a request method not supported by that page
- **408** Timeout server time out

```
<system.web>
        <customErrors mode="On"
        defaultRedirect="=Default2.aspx">
        <error statusCode="404"
        redirect="filenotfound.aspx"/>
        </customErrors>
</system.web>
```

- Here, mode has its three values :
- **OFF:** ASP.NET uses its default error page for both local and remote users in case of an error.
- **ON:** Custom error will be enabled for both local as well as remote client.
- **REMOTE ONLY:** The custom error page will be shown to a remote user only, the local user will get the built-in error page.

1. page level
2. application level
3. gobal.asax file level

- Page level :
- <%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" **ErrorPage**="~/filenotfound.aspx" Inherits="_Default" %>

AUTHENTICATION VS. AUTHORIZATION :

- **Authentication**
- Authentication refers to the process of verifying the identity of a user or entity.
- It ensures that the user is who they claim to be.
- In ASP.NET, authentication is typically performed by validating credentials, such as a username and password, provided by the user.
- The authentication process establishes the identity of the user, allowing them to access protected resources or perform certain actions within the application.

- The ASP.NET authentication scheme that is used to identify users who view an ASP.NET application. An ASP.net application has two separate authentication levels because all requests coming through IIS before it handled by ASP.NET. After IIS authentication schemes ASP.NET implements additional authentication schemes. They are :
- **None Authentication (No Authentication)**
- **Windows Authentication**
- **Forms Authentication**
- **Passport Authentication**
- The mode attribute specifies the authentication scheme.

  <authentication mode="[Windows|Forms|Passport|None]" >
- **None Authentication :**
- You can specify "None" as the authentication provider when requests are not authenticated at all or if you plan to develop custom authentication code.
- When you need "None" authentication, use the followingbWeb.config configuration:

```
<system.web>
        <authentication mode="None" />
</system.web>
```

AUTHORIZATION :

- Authorization, on the other hand, deals with granting or denying access to specific resources or functionality based on the authenticated user's privileges.
- Once a user's identity is established through authentication, authorization determines what the user is allowed to do within the application.
- It involves checking whether the authenticated user has the necessary permissions or roles to access certain pages, perform specific operations, or view confidential data.
- Authorization helps ensure that users can only access the resources they are entitled to based on their assigned roles or privileges.

- **ASP.NET AUTHORIZATION**
- ASP.NET allows two ways to authorize access to a given resources, they are URL authorization and File authorization
- **URL authorization**
- URL authorization maps users and roles to URLs in ASP.NET applications
- **File authorization**
- File authorization validate the ACL (access control list) of the .aspx or .asmx handler file to determine whether a user should have access to the file.
- **How to implement Authorization ?**
- The following example shows a sample implementation of Authorization logic in web.config file.

```
<authorization>
        <allow roles="Administrators" />
                <deny users="*" />
</authorization>
```

ASP.NET DEPLOYMENT :

- There are two categories of ASP.NET deployment:
- **Local deployment : In this case, the entire application is** contained within a virtual directory and all the contents and assemblies are contained within it and available to the application.
- **Global deployment : In this case, assemblies are available** to every application running on the server.
- There are different techniques used for deployment, however, we will discuss the following most common and easiest ways of deployment:
- XCOPY deployment
- Copying a Website
- Creating a set up project

- **XCOPY Deployment :**
- XCOPY deployment means making recursive copies of all the files to the target folder on the target machine. You can use any of the commonly used techniques:
- FTP transfer
- Using Server management tools that provide replication on a remote site
- MSI installer application
- XCOPY deployment simply copies the application file to the production server and sets a virtual directory there. You need to set a virtual directory using the Internet Information Manager Microsoft Management Console (MMC snap-in).

- **Copying a Website :**
- The Copy Web Site option is available in Visual Studio. It is available from the Website→Copy Web Site menu option. This menu item allows copying the current web site to another local or remote location. It is a sort of integrated FTP tool.
- Using this option, you connect to the target destination, select the desired copy mode:

  Overwrite

  Source to Target Files

  Sync UP Source And Target Projects
- Then proceed with copying the files physically. Unlike the XCOPY deployment, this process of deployment is done from Visual Studio environment. However, there are following problems with both the above deployment methods:

  You pass on your source code.

  There is no pre-compilation and related error checking for the files.

  The initial page load will be slow.

- **Creating a Setup Project :**
- In this method, you use Windows Installer and package your web applications so it is ready to deploy on the production server. Visual Studio allows you to build deployment packages. Let us test this on one of our existing project, say the data binding project.
- **Visit this site for creating a setup project steps :**
- https://medium.com/@binaridissanayake/a-step-by-step-guide-to-deploying-your-asp-net-project-in-iis-3bf5be6366ef

- More about WebService :
- https://www.tpointtech.com/web-services-in-c-sharp

Some Questions for Exam :

- Sort questions like : fullform of UDDI WSDL XML XDS DTD etc…
- What is XML parser ?
- Explain web.config file.
- What is tracing ? Explain application level tracing..
- Explain web service with example.
- Explain reading and write data into XML file using Dataset.
- Explain Authentication and Authorization.
- Explain Custom Error with example.
- Explain deployment.