

By :

Dr. Deepak Mathur

**WEB SERVICE 384**

---

# WEBSITE

- ✗ What is website ????
- ✗ **The Target is :**
- ✗ In Web, it is typically difficult to extract, examine, and use the data programmatically that has been used by web site.
- ✗ **The problem is :**
  - + Data is presented in a format such that it is easy for a human to read but relatively difficult and error prone for applications to read and process reliably.

# **UNDERSTANDING WEB SERVICES**

- ✖ Web services are all about delivering distributed applications via programmable components that are accessible over the Web.

**What is the need???????**

Example in next slide

# SHIPPING COMPANY EXAMPLE

---

- ✖ Total 3 parties
- ✖ First party : Company
- ✖ Second Party: Regulatory govt agencies
- ✖ Third party :???????

# DEFINITION OF WEB SERVICE

---

- ✖ 1. Web service is a programmable URL.
- ✖ 2. Web service is an application component that is remotely callable using standard Internet protocols such as HTTP and XML.
- ✖ 3. Web services use a text-based messaging model to communicate, allowing them to operate effectively on these many different platforms.



# APPLICATIONS OF WEB SERVICES

- ✖ **Services** that are either too difficult or too expensive to implement yourself.  
For example, credit card validation, financial account management, stockquotes, and so on.
- ✖ **Services** that provide commonly needed functionality for other services. For example, user authentication..
- ✖ **Services** that aggregate distributed, discrete services .  
A good example of this type of service would be travel booking.
- ✖ **Services** that integrate your business systems with your partners (or other business systems within your own organization).

# WEB SERVICE DEPENDENCY

- ✗ Does not depend on
  - ✗ specific operating system
  - ✗ object model
  - ✗ programming language.

# REPRESENTING DATA

- ✘ How do you ensure a consistent and accurate interpretation of the data when the service and consumer may reside on different platforms, operating systems, object models, and/or programming languages????????
- ✘ Any guess.....



# XML IS USED

---

- ✖ XML has the ability to describe data using a simple grammar that is highly interoperable among the many heterogeneous systems that are connected to the Internet.
- ✖ XML provides a standards-based method for describing data (also known as metadata).

# STRENGTHS OF XML.....>

- ✖ 1. XML is a text-based language, which makes it easily readable and more portable than binary data formats.
- ✖ 2. XML gives you the ability to define your own tags to describe data and its relationships to other data (hence, the word extensible in its name).
- ✖ 3. XML strictly enforces its language syntax, unlike HTML.
- ✖ 4. Parsers are widely available to accurately parse and validate XML structures that you define, which means you don't have to do it yourself!

# XML DOCUMENT STRUCTURE

- ✕ An XML document consists of
  - + a prolog
  - + document elements
  - + optional attributes that model a logically related set of data.
  
- + Let us see an example?>>>>>>>>>

```
✕ <?xml version="1.0" encoding="UTF-8"?>
✕ <weather>
  + <location city="St. Louis, MO USA">
    ✕ <forecast date="2001-07-15">
      ✕ <temperature units="F">80</temperature>
      ✕ <humidity units="%">55</humidity>
      ✕ <skies>Cloudy, 40% chance of showers</skies>
    ✕ </forecast>
  + </location>
✕ </weather>
```



# A XML REALATED QUESITON

- ✗ Can XML be used to form a LIGHT WEIGHT DATABASE ?

```
<ROOM>  
  <ST>  
    <ROLLNO> 1</ROLLNO>  
    <NAME>A</NAME>  
  </ST>  
</ROOM>
```

+ HAVE U GOT THE ANSWER??????????????

# XSD

---

- ✖ XML parser uses strict rules to ensure that the XML document is well formed.
- ✖ The XSD (XML SCHEMA DEFINITION) language defines rules for describing the *valid combinations and relationships of elements, attributes, and data types* that can appear in an XML document.

- ✗ <xsd:element name="forecast">
- ✗ <xsd:complexType>
- ✗ <xsd:all>
- ✗ <xsd:element ref="temperature" />
- ✗ <xsd:element ref="humidity" />
- ✗ <xsd:element ref="skies" />
- ✗ </xsd:all>
- ✗ <xsd:attribute ref="date" />
- ✗ </xsd:complexType>
- ✗ </xsd:element>

```
✕ <?xml version="1.0" encoding="UTF-8"?>
✕ <weather>
  + <location city="St. Louis, MO USA">
    ✕ <forecast date="2001-07-15">
      ✕ <temperature units="F">80</temperature>
      ✕ <humidity units="%">55</humidity>
      ✕ <skies>Cloudy, 40% chance of showers</skies>
    ✕ </forecast>
  + </location>
✕ </weather>
```



# EXCHANGING MESSAGES

- ✖ Web services communicate in the form of messages.
- ✖ 1. A request message delivers information about an operation to be executed and any data required to carry out that operation. Request messages flow from clients to Web services.
- ✖ 2. A response message delivers information about the results of the operation execution. Response messages flow from Web services to clients.

# MESSAGE EXCHANGE WITH SOAP

- ✖ The Simple Object Access Protocol (SOAP) is an industry-standard protocol that enables message-based communication for Web services. SOAP implements a message format based on XML to exchange **1.Function requests and 2. Function responses.**

# 1. FUNCTION REQUESTS EXAMPLE

- ✗ <soap:Body>
- ✗ <ConvertTemperature  
xmlns="http://tempuri.org/">
- ✗ <Temperature>{decimal}</Temperature>
- ✗ <FromUnits>{string}</FromUnits>
- ✗ <ToUnits>{string}</ToUnits>
- ✗ </ConvertTemperature>
- ✗ </soap:Body>

**Table 22-2: CTemp method arguments and data types**

Argument	Data Type
Temperature	Decimal
FromUnits	String
ToUnits	String

For ex.

Temperature=98.6

FromUnits=F

ToUnits=C



# WSDL

- ✖ The WSDL (Web Service Description Language) document defines the message formats and message exchange patterns that a Web service can process . < Discussed in previous 2 slides

**Request and Response>**

# ***PUBLISHING WEB SERVICES***

- ✖ We must now consider how a potential consumer of a Web service will locate a WSDL document on a target Web server.
- ✖ Because if a potential consumer locates WSDL successfully then .....??????

What the consumer would be able to do ?

# WEB SERVICE DISCOVERY

- ✖ The process in which a web service consumer learns that a Web service exists and where to find the Web service's WSDL document.
- ✖ The Microsoft .NET Framework provides a tool named **disco.exe** to enable Web service discovery.
- ✖ What the disco.exe can be like :  
`<wsdl:contractRef ref="http://jdc7200cte/Services/CTemp/CTemp.asmx?WSDL"/>`

# FINDING WEB SERVICES

✗ Who will find ?

**consumer** is the correct answer or not?

- ✗ Similar to the search engine approach that is used to query and locate Web pages (google.com ) the
- ✗ **Universal Description, Discovery, and Integration (UDDI)** specification defines a logically centralized web site or engine to find the Web services that they offer.
- ✗ <http://www.uddi.org>



The Find page is shown in Figure 22.2.



# UDDI

---

- ✖ UDDI defines, classifies and stores three basic types of information:
- ✖ **White Pages:** Describes address, contact, and other standard business demographic information.
- ✖ **Yellow Pages:** Describes industrial categorizations for businesses based on standard categories.
- ✖ **Green Pages:** Describes the technical specification for Web services .

# WEB SERVICES INFRASTRUCTURE

- ✖ Web service infrastructure means that what are the Microsoft technologies and tools that provide the essential architectural elements for
  - ✖ 1.executing
  - ✖ 2.creating
  - ✖ 3.consuming web services.

# ***THE MICROSOFT .NET FRAMEWORK***

- ✕ The Key features of the .NET Framework that provide support for Web services include:

The Common Language Runtime (CLR)

The .NET Framework Class Library(BCL)

ASP.NET



# ADVANTAGES OF USING ASP.NET

**ASP.NET** applications are fully compiled .NET applications, providing superior performance characteristics.

- ✘ **ASP.NET** supports WYSIWYG HTML editors and programming environments such as Visual Studio .NET. This enables you to be very productive when developing ASP.NET applications and enables you to leverage the many features of these tools.
  - ✘ **ASP.NET** applications support extensive configuration capabilities based on XML configuration files.
  - ✘ **ASP.NET** provides flexible, advanced, and easy-to-use application and session state management
- ASP.NET** implements multiple authentication and authorization schemes.

# WEB SERVICES INFRASTRUCTURE

---

- ✖ 4 primary infrastructure needed to support Web services:
  - ✖ 1. **Web Service Directories**, which provide a means to locate providers of Web services
  - ✖ 2. **Web Service Discovery**, which provides the capability to locate Web services
  - ✖ 3. **Web Service Description**, which enables Web service capabilities to be described
  - ✖ 4. **Web Service Wire Formats**, which allow Web services to exchange data and messages

# 1. WEB SERVICE DIRECTORIES

- ✖ Web service directories provide a centralized, Internet-accessible location that consumers can use to **find Web services**.
- ✖ You can think of Web service directories as a type of Web portal or "Yellow Pages" specifically suited for listing and locating Web services.

- ✖ Using Web service directories, you can search for Web services using a variety of structured criteria. <http://www.uddi.org>
- ✖ Visual Studio has the ability to search these online UDDI directories automatically.



## 2. WEB SERVICE DISCOVERY

- ✖ Web service discovery is the process of locating one or more related documents that describe a specific Web service such as WSDL.
- ✖ The implementation of the discovery process is also embodied in the .NET Framework's **namespace.**

### 3. WEB SERVICE DESCRIPTION

- ✖ A Web service description is an XML document that defines the capabilities of a Web service.
- ✖ That is in terms of the **request messages** that a Web service will accept as well as the **response messages** a Web service can return

## 4. WEB SERVICE WIRE FORMATS

- ✗ Wire formats define the method by which Web service request and response messages are encoded and transported between the Web service and any consumer.
- ✗ ASP.NET Web Services support three wire formats:

HTTP-GET

HTTP -POST

HTTP-SOAP

▪

# HTTP-GET

- ✖ The HTTP-GET protocol encodes Web service operation requests and arguments **in the URL to the Web service.**
- ✖ Example:
- ✖ <http://localhost/ctemp/ctemp.asmx/ctemp?Temperature= 32&FromUnits=F&ToUnits=C>



# HTTP-POST

- ✖ The HTTP-POST protocol encodes Web service operation requests and arguments **within the payload area of the HTTP-POST request as name/value pairs.**
- ✖ **Which is safe GET or POST**

# HTTP-SOAP

---

- ✖ HTTP-SOAP is the default ASP.NET Web Service wire format.
- ✖ Web service request and response messages are encoded into SOAP messages.
- ✖ SOAP messages are encoded in XML using the SOAP vocabulary defined in the specification.
- ✖ Explained Earlier in slide no 17 and 18

# SECURITY<sup>331</sup>

---

- ✖ What is security ?
- ✖ Need of security?
  - ✖ transfer of sensitive data over the Internet
  - ✖ exchange of sensitive data between Web applications
  - ✖ risks of hackers to steal or misuse the crucial financial data
- ✖ One fact:
- ✖ For the ASP.NET applications, the underlying Web server is Microsoft Internet Information Services (IIS).

# ASP.NET SECURITY

---

## ✕ Authentication

- ✕ example

## ✕ Authorization

- ✕ example

## ✕ Impersonation

- ✕ example



# AUTHENTICATION

---

- ✖ different types of authentication can be implemented by using the IIS server:
- ✖ Anonymous Authentication
- ✖ Basic Authentication
- ✖ Integrated Windows Authentication
- ✖ Digest Authentication

# ANONYMOUS AUTHENTICATION

---

- ✖ Allows all users to browse the Web site
- ✖ No prompting for a username and password
  - ✖ Then what is the need ???
  - ✖ Any Example :
    - ✖ Static website

# BASIC AUTHENTICATION

- ✗ Requires the users to enter username/password combination
- ✗ The password is sent over the network in an **unencrypted form**, making it possible for unauthorized users / hackers to grab the data.

# INTEGRATED WINDOWS AUTHENTICATION:

✗ Basic Authentication

+

Valid Windows users

✗ IIS will verify the username and password with a Windows Domain Controller



# DIGEST AUTHENTICATION

✖ Basic Authentication

+

Hash Value

✖ Password as a hash value over the network rather than the password. The hash value cannot be decrypted and hence the original text cannot be deciphered.

What is hash value???

# HASH FUNCTION

- ✘ A complex hash function may look like :::
- ✘  $F(x) = e^{45} + 3 * \log_x(3245 * x^2 + 75 * x^3 + 45) / 3 + \sqrt{3 \log x^2 * 56 / 9.5} * \log_{10} x$
- ✘ Now input x and find F(x) ??????
- ✘ Easy or Tuff

# AUTHORIZATION

- ✖ The access permissions for example:
- ✖ **Read**: Allows users to retrieve and read the content stored in the virtual directory.
- ✖ **Write**: Allows users to retrieve and modify the content stored in the virtual directory. Example typical example of this would be a virtual directory that stores the files that are uploaded as attachments to e-mail messages.

- ✖ **Script source access:** Allows users to view the source code of any server-side program.
- ✖ **Directory browsing:** Allows users to view the contents of the entire virtual directory.
- ✖ **Log visits:** Keeps track of the number of users who visit the site, and records information about various details, such as the IP address of the client and the resources that are requested for.
- ✖ **Index:** Uses Microsoft Index Server to index the virtual directory. .



- ✖ The following are the different access permissions that can be assigned to users and groups for the files and directories on the server:
- ✖ **Full Control:** Allows users to have complete control on files and/or directories.
- ✖ **Modify:** Allows users to modify the contents of files and/or directories. However, users will not be able to delete files and/or directories.

- ✖ **Read & Execute:** Allows users to read the contents of the existing files and/or directories and execute any application stored in that folder. However, users will not be able to modify the contents of the files and/or directories.
- ✖ **List Folder Contents:** Allows users to view the contents of the folder. However, users will neither be able to read the contents of any file in the folder nor modify any contents.
- ✖ **Write:** Allows users to make changes to files and/or directories.
- ✖ **No Access:** Does not allow any access to files and/or directories.

# ASP.NET AUTHENTICATION OPTIONS

- ✖ An ASP.NET Web application can be provided with one of the following types of security:
- ✖ **Windows:** The application is secured by using **Integrated Windows Authentication (slide no 43 )**  
In this method, access to a Web application is allowed only to those users who are able to verify their Windows credentials.

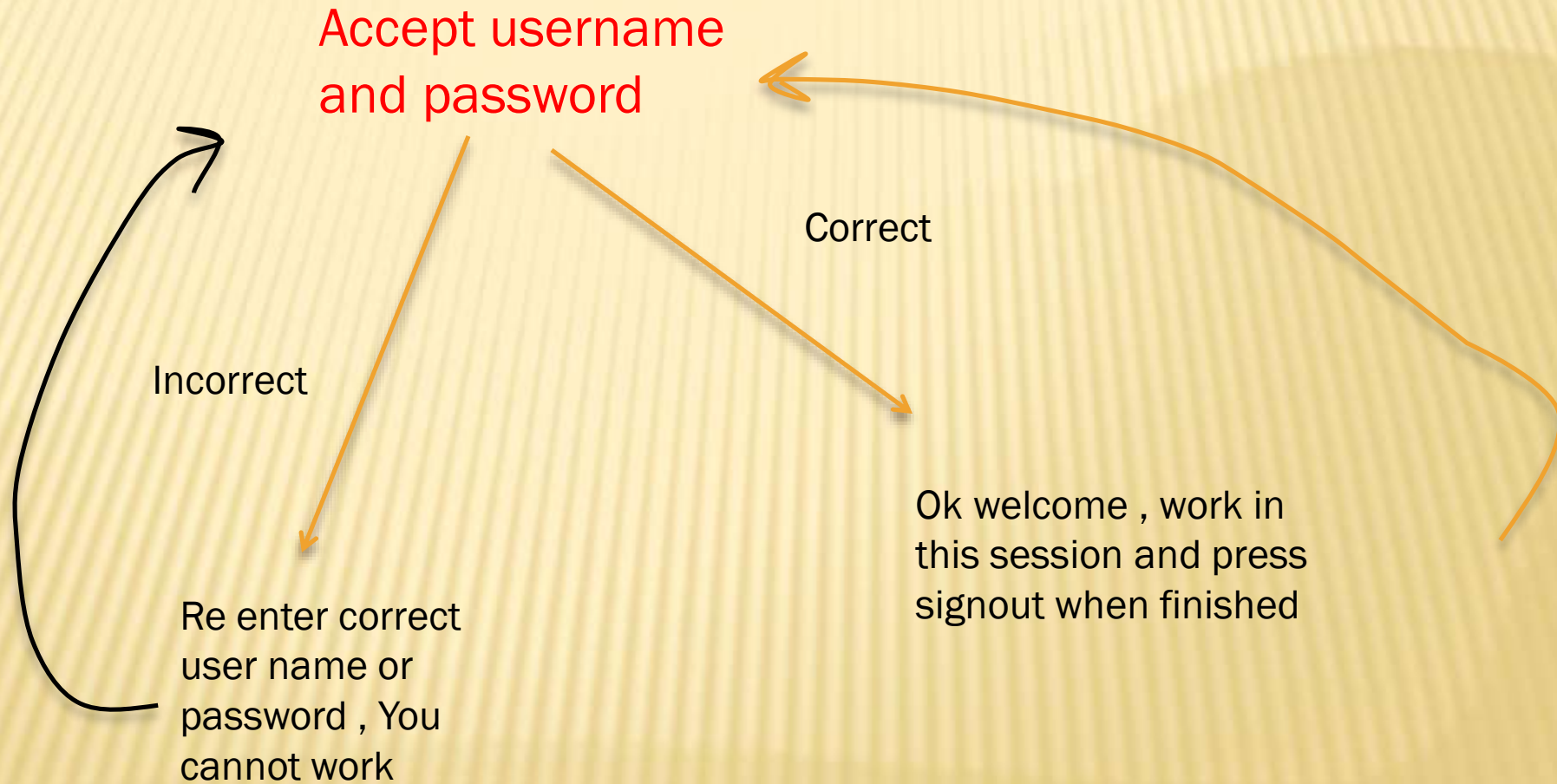
- ✖ **Passport:** The application is secured by using Microsoft Passport authentication. Passport is a single-sign-on technology developed by Microsoft for use on the Web. For more information on using Microsoft Passport, visit <http://www.passport.com/business/>.
- ✖ **Forms:** The application is secured by using a custom authentication model with cookie support. ( this we would try to implement here )
- ✖ **None:** The application is not secured; access to the application does not require authentication.



# FORMS-BASED AUTHENTICATION

- ✖ Built-in feature
- ✖ Here when a user is determined to be unauthenticated, the user is automatically redirected to the login page.( Has this happened to u????)
- ✖ Now to implement in asp.net we use web.config file>>>>>

- ✗ The following is a sample code in the web.config file used to enable forms-based authentication:
- ✗ <configuration>
- ✗ <system.web>
- ✗ <authentication mode="Forms">
- ✗ <forms name=".B2CBuySiteAuthCookie" loginUrl=
- ✗ "userauth.aspx" protection="All" timeout="10" />
- ✗ </authentication>
  
- ✗ **Now we will code userauth.aspx>>>>>**



# "USERAUTH.ASPX"

```
× <%@ Import Namespace="System.Web.Security " %>
× <html>
× <script language="VB" runat=server>
× Sub Login_Click(Src As Object, E As EventArgs)
× If UserName.Value = "john" And UserPass.Value = "secret"
× 'Credentials are ok, redirect back to the page that forced
× ' authentication, pass the user name
× FormsAuthentication.RedirectFromLoginPage(UserName.Value,False)
× Else
× Msg.Text = "Invalid user name or password: Please try again"
× End If
× End Sub
× </script>
```

???? Why  
false



## "USERAUTH.ASPX" CONTINUES.....

Now in this file in body section :

```
<asp:button text="Login" OnClick="Login_Click"  
runat=server/>
```





Which function is this  
?????

Now we will code our that

```
file(a.aspx)>>>>>>>>>>
```

- ✖ Remember it would go to file **a.aspx only when username and password are correctly entered** by user

# A.ASPX

- ✗ <%@ Import Namespace="System.Web.Security " %>
  - ✗ <html>
  - ✗ <script language="VB" runat=server> When it would be displayed
  - ✗ Sub Page\_Load(Src As Object, E As EventArgs)
  - ✗ Welcome.Text = "Hello, " + User.Identity.Name 
  - ✗ End Sub
  - ✗ Sub Signout\_Click(Src As Object, E As EventArgs)
  - ✗ FormsAuthentication.SignOut()
  - ✗ Response.Redirect("userauth.aspx")
  - ✗ End Sub
  - ✗ </script> 
- Would go back on userauth.aspx file

## A.ASPX COTINUED .....

- ✗ <body>
- ✗ Using Forms Authentication
- ✗ <form runat=server>
- ✗ <asp:label id="label1" runat=server/>
- ✗ <asp:button text="Signout"  
OnClick="Signout\_Click"
- ✗ runat=server/>
- ✗ </form>
- ✗ </body>
- ✗ </html>



**OUTPUT 1: IF USERNAME AND PASSWORD ARE CORRECTLY ENTERED THEN IT WOULD GO ON A.ASPX ELSE WOULD REMAIN HERE ONLY**



**OUTPUT 2 : THIS IS A.ASPX FILE AND WHEN USER WOULD PRESS SIGNOUT BUTTON IT WOULD GOTO USERAUTH.ASPX**



# ASP.NET APPLICATION CONFIGURATION <sup>236</sup>

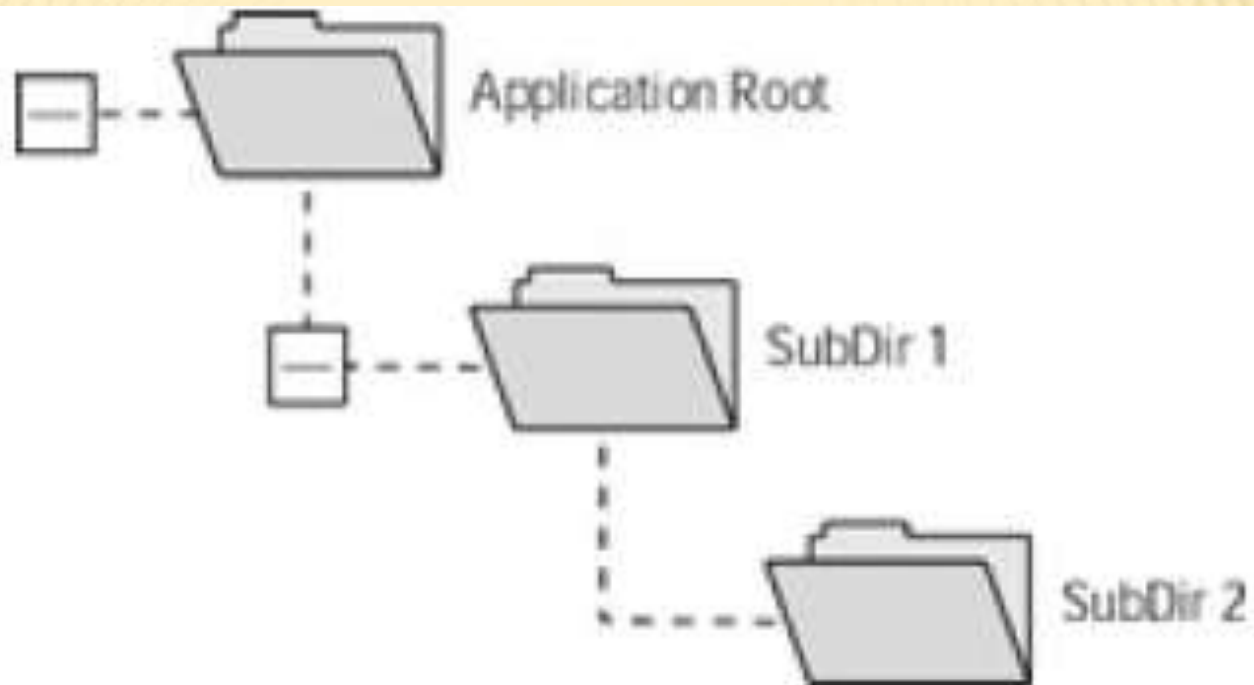
- ✘ The deployment process includes installation and **configuration** of the Web application (Web site) on an application Web Server.
- ✘ **Configuring** a Web site requires implementation of settings according to the server's capabilities and requirements.
- ✘ At a later stage, the site administrator might
- ✘ need to change the settings of the site or the server so **configuration** file would be a help.

# WEB.CONFIG

- ✘ The configuration information for the entire ASP.NET application is defined and contained in *configuration files*. *These files are written in XML and is named Web.config*
- ✘ ASP.NET uses a *hierarchical configuration architecture that uses an XML format*.
- ✘ *In the hierarchical configuration architecture, whenever a client makes a request for an ASP.NET application or a specific ASP.NET resource, ASP.NET checks the settings for the URL requested by the client in a hierarchical fashion*

The check is carried out using the configuration files located in the path for the requested URL





**Figure 14-1: File structure of a Web site**

# ADVANTAGES OF WEB.CONFIG

- ✖ The configuration information for the ASP.NET applications is stored in XMLbased configuration files, which makes it easy to read and write.
- ✖ The configuration information files are stored in the same directory tree as the rest of the application files, thus making the installation of ASP.NET applications easy.

- ✖ The configuration system is highly flexible and allows developers to store customized configuration criteria and settings in the configuration system.
- ✖ The configuration information contained in the XML file is applied hierarchically

# BASIC STRUCTURE OF AN ASP.NET CONFIGURATION FILE

PAGE1

```
<configuration>

  <configSections>
    <section name="appSettings" type = "System.Web.
Configuration.NameValueSectionHandler" />
    <section name="httpModules" type = "System.Web.
Config.HttpModulesConfigHandler"/>
    <section name="httpHandlers" type = " System.Web.
Config.HttpHandlerConfigHandler " />
    <section name="sessionState" type = " System.Web.
Config.SessionStateConfigHandler " />
    <section name=" globalization " type = " System.Web.
Config.GlobalizationConfigHandler " />

    <!-- Additional configsection declarations go here -->
  </configSections>

  <appSettings>
    <!--custom application settings go here-->
  </appSettings>
```



# PAGE2

```
<system.web>
```

```
<compilation defaultLanguage="vb" debug="true">
```

```
<!-- all compiltion config is here -->
```

```
</compilation>
```

```
<customErrors mode="RemoteOnly">
```

```
<!-- error config goes here -->
```

```
</customErrors>
```

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

```
<!-- authentication settings controlled here -->
```

```
</authentication>
```

# PAGE3

```
<authorization>
```

```
<!-- Allow/Deny all users , roles-->
```

```
</authorization>
```

```
<trace enabled="false" requestLimit="10" pageOutput="false"
```

```
traceMode="SortByTime" localOnly="true" />
```

```
<!-- control trace settings for this web application -->
```

```
<sessionState>
```

```
<!-- configure session state for this web application -->
```

```
</sessionState>
```

```
<httpHandlers>
```

```
<!--configure HTTP Handlers for this web application-->
```

```
</httpHandlers>
```

```
<httpModules>
```

```
<!--configure HTTP Modules for this web application-->
```

```
</httpModules>
```

```
<globalization/>
```

```
<!-- configure globalization settings -->
```

```
</system.web>
```

```
</configuration>
```

# ASP.NET CONFIGURATION SECTIONS

- ✗ **<configuration> section**
- ✗ The **<configuration>** section is the root configuration section for all the ASP.NET configuration files. This is a special tag that encapsulates all other sections in the file.
- ✗ The syntax is as follows:
  - ✗ **<configuration>**
  - ✗ **</configuration>**

# <CONFIGSECTIONS> SECTION

- ✖ <configSections> section
- ✖ The <configSections> section contains a list of the configuration section handlers associated with each configuration section. When you want to devise your own section handlers, you must declare them in the <configSections> section. The syntax is as follows:
  - ✖ <configSections>
  - ✖ <section name="config section element name" type = "Type"/>
  - ✖ </configSections>
- ✖ The two attributes Name and Type are described as follows:
  - ✖ § Name: Used to specify the name of the element that will contain the configuration data.
  - ✖ § Type: Used to specify the configuration section handler class to be associated with the element specified in the Name attribute.



# <APPSETTINGS> SECTION

- ✖ <appSettings> section
- ✖ The <appSettings> section of the Web.config file provides a way to define custom application settings for an application.
- ✖ **Example**
- ✖ <appSettings>
- ✖ <add Key="dsn"
- ✖ Value="localhost;uid=user1;pwd=password"/>
- ✖ </appSettings>

# <BROWSERCAPS> SECTION

- ✗ <browserCaps> section
- ✗ The <browserCaps> section of the Web.config file enables you to specify the configuration settings for the browser capabilities component.

# EXAMPLE

```
<browserCaps>
<result type="System.Web.HttpBrowserCapabilities" />
<use var="Environment Variable" />
browser="type"
version=browser version
majorver=0
minorver=0
frames=false/true
tables=false/true
<filter>
<case match="Name of operating systems to match">
platform="Current OS"
</case>
...
</filter>
</browserCaps>
```

# ANOTHER ONE

```
<browserCaps>  
<result type="System.Web.HttpBrowserCapabilities" />  
<use var="HTTP_USER_AGENT" >  
  browser="Unknown"  
  version=0.0  
  majorver=0  
  minorver=0  
  frames=false  
  tables=false />  
<filter>  
  <case match="Windows 95 | Win95">  
    platform=Win95  
  </case>  
  <case match="Windows NT | WinNT">  
    platform=WinNT  
  </case>  
</filter>  
</browserCaps>
```



# <COMPILATION> SECTION

```
<compilation debug="true/false">
```

```
× <compilers defaultLanguage="[Lang]">
```

```
× <compiler
```

```
× language="[Lang]"
```

```
× extension="[ Ext]"
```

```
× type="Type[,assemblyName]"/>
```

```
× </compiler>
```

```
× <assemblies>
```

```
× <add assembly="[Assembly] " />
```

```
× <remove assembly="[Assembly]" />
```

```
× <clear />
```

```
× </assemblies>
```

```
× <namespaces>
```

```
× <add namespace="[namespace]"/>
```

```
× <remove namespace="[namespace]"/>
```

```
× <clear/>
```

```
× </namespaces>
```

```
</compilation>
```

# <CUSTOMERRORS> SECTION

✖ The <customErrors> section provides a means for defining custom error messages for an ASP.NET application.

✖ example

```
<customErrors defaultRedirect="customerror.htm"  
  mode="On">
```

```
<error statusCode="500"  
  redirect="CustomInternalError.htm"/>  
</customErrors>
```

# <GLOBALIZATION> SECTION

- ✗ The <globalization> section is used to configure the globalization settings of the application. The syntax is as follows:

```
<globalization  
requestEncoding="[any valid encoding string]"  
responseEncoding="[any valid encoding string]"  
fileEncoding="[any valid encoding string]"  
culture="[any valid culture string]"  
uiCulture="[any valid culture string]"  
>
```

## ✖ Example

```
<globalization  
fileEncoding="utf-8"  
requestEncoding="utf-8"  
responseEncoding="utf-8"  
>
```



# <HTTPHANDLERS> SECTION

- ✖ The <httpHandlers> section maps the incoming URL requests to the classes that implement IHttpHandler or IHttpHandler interfaces  
<httpHandlers>

- ✖ Syntax

<httpHandlers>

<add verb="[verb list]"

path="[path/wildcard]"

type="Type[,assemblyName]" />

<remove verb="[verb list]"

path="[path/wildcard]" />

<clear />

</httpHandlers>

# <HTTPMODULES> SECTION

- ✖ The <httpModules> section is used for adding, removing, or clearing HTTP modules in an application. The syntax is as follows:

<httpModules>

<Add Type="Type [,assemblyName]" name="module name"/>

<Remove Type="Type [,assemblyName]" name="module name"/>

<Clear />

</httpModules>

# <AUTHENTICATION> SECTION

```
<authentication mode="[Windows/Forms/Passport]">  
<Forms name="[name]" loginurl="[url]" protection=  
"All/None/Encryption/Validation" timeout="30" path="/">  
<credentials passwordformat="[Clear/ SHA1/ MD5]">  
<user name="[UserName]" password="[password]"/>  
</credentials>  
</forms>  
<passport redirectUrl="url"/>  
</authentication>
```

# <AUTHENTICATION> SECTION CONTD..

```
<authentication mode="Cookie">
<forms name="401kApp" loginUrl="/Firstlogin.aspx"
protection="All">
<credentials passwordformat="SHA1">
<user name="Marie" password="GAF97FSA3223NTT"/>
<user name="Caste" password="DF^$3GFDX443BSD99"/>
<user name="RockMen"
password="IDCJMWAFLKSTGDLS##"/>
</credentials>
</forms>
</authentication>
```



# <AUTHORIZATION> SECTION

<authorization>

<allow users="[comma separated list of users]"  
roles="[comma separated list of roles]"/>

<deny users="[comma separated list of users]"  
roles="[comma separated list of roles]"/>

</authorization>

# <PROCESSMODEL> SECTION

- ✖ The <processModel> section is responsible for configuring the ASP.NET process model settings on an IIS Web server. The process model settings are used for defining how the threads within a process should work.

# <SESSIONSTATE> SECTION

- ✖ The <sessionState> section provides a means to configure the session state HttpModule of the ASP.NET application. The syntax is as follows:

```
<sessionstate  
mode = "mode="Off|Inproc|StateServer|SqlServer"  
cookieless="true|false"  
timeout="number of minutes"  
connectionString="server name:port number"  
sqlConnectionString="sql connection string"/>
```

# <TRACE> SECTION

- ✖ The <trace> section allows the configuration of the ASP.NET trace service. The syntax is as follows:

<trace

enabled="[true/false]"

requestLimit="[Integer]"

pageOutput="[true/false]"

✖ />



# <WEBSERVICES> SECTION

- ✖ The <webServices> section is used to control the settings of the ASP.NET WebServices.

# GLOBAL.ASAX

---

- ✗ Global.asax is an optional file which is used to handling higher level application events .
- ✗ For example:
  - ✗ Application\_Start
  - ✗ Application\_End
  - ✗ Session\_Start
  - ✗ Session\_End                      etc
- ✗ It is also popularly known as ASP.NET Application File.
- ✗ This file resides in the root directory of an ASP.NET-based application.

- ✖ Global.asax contains a Class representing your application as a whole.
- ✖ At run time, this file is parsed and compiled into a dynamically generated .NET Framework class derived from the `HttpApplication` base class.
- ✖ You can deploy this file as an assembly in the `\bin` directory of an ASP.NET application.
- ✖ The `Global.asax` file itself is configured so that if a user requests the file, the request is rejected. External users cannot download or view the code written within it.

# PROCESS TO CREATE GLOBAL.ASAX

Visual Studio>>>>>>>>create new  
website>>>>>>>>>>>go to solution  
explorer>>>>>>>add new  
item>>>>>>>>>Global application  
class>>>>>Add



- ✖ Unit 5 ends
- ✖ THANK U

```
x <%@ Application Language="C#" %>
x <script runat="server">
x     void Application_Start(object sender, EventArgs e)
x     {
x         // Code that runs on application startup
x     }
x     void Application_End(object sender, EventArgs e)
x     {
x         // Code that runs on application shutdown
x     }
x     void Application_Error(object sender, EventArgs e)
x     {
x         // Code that runs when an unhandled error occurs
x     }
x     void Session_Start(object sender, EventArgs e)
x     {
x         // Code that runs when a new session is started
x     }
x     void Session_End(object sender, EventArgs e)
x     { // Code that runs when a session ends. The Session_End event is raised only when the
sessionstate mode is set to InProc in the Web.config file. If session mode is set to StateServer or SQLServer,
the event is not raised.
x     }
x </script>
```