

## Unit 3: Advance ASP.NET

### 3.1 Web.ConfigFile

**1 Purpose:**

Web.config is a configuration file used to manage a website's settings in ASP.NET applications.

**2 Format:**

It uses XML format where specific tags have predefined meanings.

**3 Separation of Concerns:**

Settings are kept **separate from application code**, promoting clean architecture and easy maintenance.

**4 Usage by IIS and ASP.NET Core Module:**

Both **IIS (Internet Information Services)** and the **ASP.NET Core Module** read the web.config file to configure the hosting environment.

**5 Location:**

A single Web.config file is usually located in the **root directory** of the application. However, **subfolders** within the application can also contain their own Web.config files for folder-specific configurations.

**6 Centralized Configuration:**

Web.config file Acts as a **central repository** for application-level settings accessible by web pages.

**7 Connection Strings:**

Web.config file commonly used to store **database connection strings** in a centralized location.

If the connection string changes, you only need to update it in **one place**.

**8 Other Configuration Settings:**

The file can hold various settings such as:

1. Database connection strings
2. Caching options
3. Session state configuration
4. Error handling policies
5. Security settings

**ForExample**

```
<connectionStrings>
    <addname="myCon"connectionString="server=MyServer;database=student;
    uid=studadmin; pwd=admin123" />
</connectionStrings>
```

The above code represents the **connection strings** available to the website, which define how the application connects to various databases.

To add **custom application settings** as **name-value pairs** (for read-only access), the<appSettings> Section is used in the Web.config file.

### Example of using<appSettings>:

```
<configuration>
  <appSettings>
    <addkey="appCategory"value="Mythology"/>
    <addkey="appBook"value="BhagwadGeeta">
  </appSettings>
</configuration>
```

### Advantages of Using XML-Based Configuration Files in ASP.NET:

1. **Extensible Configuration System:**
  - The ASP.NET configuration system is **easily expandable**, allowing convenient storage and retrieval of application-specific data.
2. **Human-Readable Format:**
  - XML-based configuration files are **easy to read and understand** for developers and administrators.
3. **No Server Restart Required:**
  - When configuration file settings are modified, the **web server does not need to be restarted**.
4. **Automatic Change Detection:**
  - ASP.NET **automatically detects** changes in the configuration file and applies them to the running application.
5. **Flexible Editing:**
  - You can create and edit ASP.NET configuration files using **any standard text editor or XML parser**.

### 3.2 SiteMapPath control:

The **SiteMapPath** control is used to display a **bread crumb-style navigation** for a website. Its primary purpose is to **help users understand their current location** within the site's structure and easily navigate back to previous sections.

#### Key Purposes:

1. **Breadcrumb Navigation:**
  - Shows a trail of links from the homepage to the current page.
  - Helps users backtrack without needing to use the browser's back button.
2. **Enhanced User Experience:**
  - Provides clear context of where the user is within the website.
  - Makes site navigation more intuitive, especially in large websites.
3. **Centralized Navigation Management:**
  - Pulls navigation structure from a .sitemap XML file.

- Any changes to the site structure in the sitemap are automatically reflected in the breadcrumb navigation.
- 4. **Support for SEO and Accessibility:**
  - Breadcrumb navigation is beneficial for **search engines** to understand site structure.
  - Improves accessibility by allowing screen readers to convey page hierarchy.
- 5. **Consistent Navigation UI:**
  - Ensures that all pages display navigation paths consistently across the website.

**ASP.NET tag** is `<asp:SiteMapPath ID="SiteMapPath1" runat="server"> </asp:SiteMapPath>`

#### Properties

Property	Description
PathDirection	Gets or sets the order that the navigation path nodes are rendered in.
Path Separator	Gets or sets the string that delimits SiteMapPath nodes in the rendered navigation path.
NodeStyle	Gets the style used for the display text for all nodes in the site navigation path.
RootNodeStyle	Gets the style for the root node display text.
CurrentNodeStyle	Gets the style used for the display text for the current node.

#### Sample.sitemapFile (Web.sitemap):

Save this file in the root of your project as Web.sitemap:

```
<?xmlversion="1.0"encoding="utf-8"?>
<siteMapxmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">

  <siteMapNode      url="http://localhost:52745/Demo/WebServCtl.aspx"
  title="Home" description="">
    <siteMapNode      url="~/DtBind.aspx"      title="WebServerControls"
    description="Services"/>
    <siteMapNodeurl="~/FormView1.aspx"          title="Computer"
    description="Computer"/>
  </siteMapNode>
</siteMap>
```

Now put sitemappath control on WebServCtl.aspx, DtBind.aspx and FormView1.aspx.

### 3.3 WebServices

A **Web Service** is a software component that uses **XML** to communicate with other applications over standard Internet protocols. In essence, it enables applications to interact and exchange information over the web, regardless of the platform or programming language they are built on.

#### 3.3.1 Basics of Web Services

A Web Service is a web-based functionality that can be accessed using standard web protocols (like HTTP). It is:

- **Language-independent**
- **Platform-independent**
- **Protocol-independent**

Web Services are typically **stateless**, meaning they do not store any information about previous interactions. They are also **discoverable**, meaning developers and applications can use service registries to locate and identify available Web Services.

#### Key Technologies Used in Web Services

- **SOAP (Simple Object Access Protocol):** Enables communication between applications and services using XML-based messages.
- **XML (eXtensible Markup Language):** Describes data in a format readable by any application that supports XML, regardless of the platform or language.
- **WSDL (Web Services Description Language):** A standard XML format for describing Web Services and how to access them.
- **UDDI (Universal Description, Discovery, and Integration):** Allows the creation and management of searchable registries for Web Services.

#### Advantages of Web Services

- Promote **modular programming**, allowing different organizations to use the same Web Service components.
- Use **open, text-based standards** to enable communication between components written in various languages and running on different platforms.
- Can be **implemented gradually**, reducing costs and minimizing disruptions during technology transitions.
- Are **easy and cost-effective to create** since they often repurpose existing infrastructure and applications.
- Help **lower costs** associated with enterprise application integration and B2B (business-to-business) communications.

## Limitations of Web Services

- **Interoperability issues** may arise between different platforms and implementations.
- **Performance limitations**, especially in high-demand environments.
- **Increased network traffic** due to the verbosity of XML and SOAP.
- **Web Services standards are still evolving**, leading to compatibility concerns.
- Some organizations seek to **maintain proprietary control** over specific Web Services standards, limiting open adoption.

### 3.3.2 Interacting with Web Services

The extension of web service file in [asp.net](#) is .asmx. It contains the methods that are to be used as web services. To declare the method as web service <WebMethod> attribute is used before method declaration.

You need to create two applications:

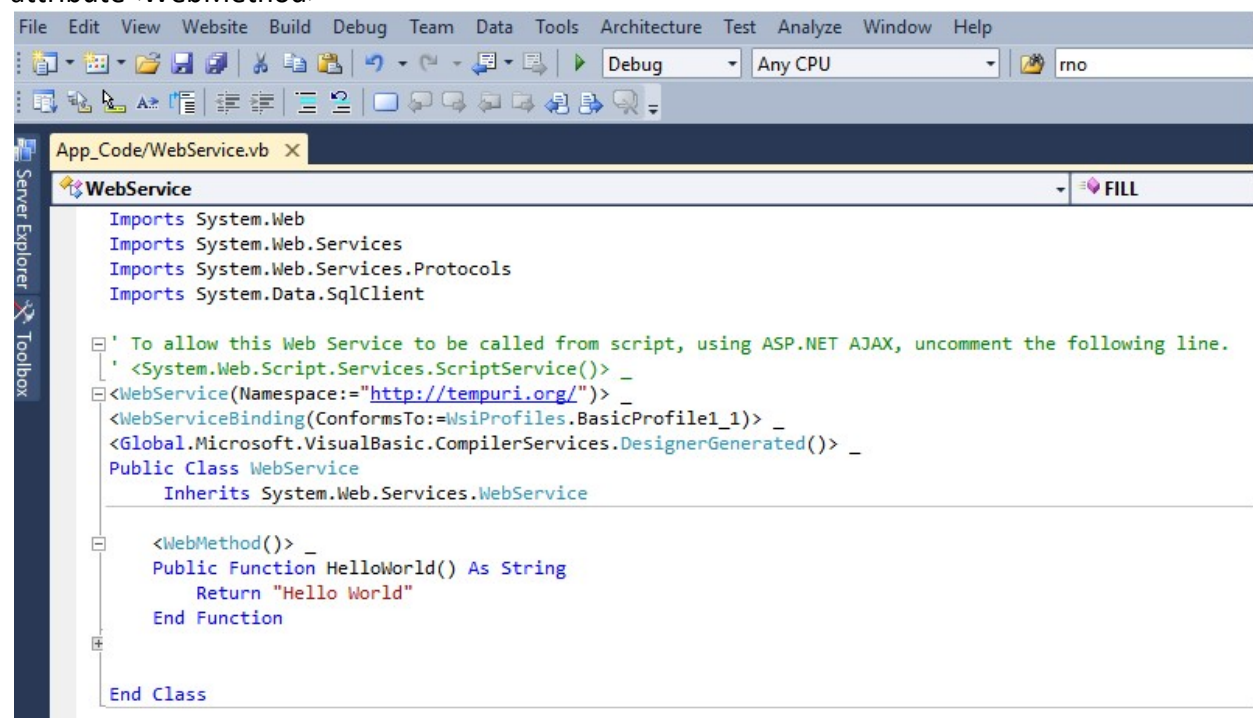
1. Web Service
2. Web Application to access Web Service

#### Creating Web Service

From File Menu->Select New->Select Web Site...->Select Web Service from the wizard -> Give appropriate name ->Click on Ok button.

There will be two default files one is service.asmx and other is service.vb, which is stored in app\_code folder.

Now in service.vb, there will be one method by default which is prefixed with the attribute<WebMethod>



```
File Edit View Website Build Debug Team Data Tools Architecture Test Analyze Window Help
Debug Any CPU rno
App_Code/WebService.vb
WebService
Imports System.Web
Imports System.Web.Services
Imports System.Web.Services.Protocols
Imports System.Data.SqlClient

' To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.
' <System.Web.Script.Services.ScriptService()> _
<WebService(Namespace:="http://tempuri.org/")> _
<WebServiceBinding(ConformsTo:=WsiProfiles.BasicProfile1_1)> _
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Public Class WebService
    Inherits System.Web.Services.WebService

    <WebMethod()> _
    Public Function HelloWorld() As String
        Return "Hello World"
    End Function
End Class
```

Add another function below the HelloWorld Method of Service.vb file.

### Example

```
<WebMethod>
Public Function GreetUser(ByVal name As String) As String
    Return "Hello " & name
End Function
```

Save and execute the Web Service.

Service.asmx file will be executed in web browser.

### Web Application to access Web Service

TO consume web service, you need to provide web reference of web service in your web application.

From File Menu->Select New->Select Web Site..-> Select ASP.NET Web Site from the wizard  
-> Give appropriate name->Click on Ok button.

Place textbox (txtName) to take input, button (btnDisplayName) to call Web Service and label (lblName) for message and Label (lblGreetUser) to greet the user on the page

Add Web Reference in the Web Application:

- Right Click on the Web Application in Solution Explorer
- Select Add Web Reference from the Popup menu
- Copy the URL of Web Service from the browser
- Paste it in the URL bar of the **Add** Web Reference Wizard.
- Click on go button.

Here the Web Reference is the reference of another Web Service that is added in the Web Application to access the Web Service.

By default, you will get the Web Reference name as localhost.

Click on Add Reference Button to add reference to the Web Application. **Web Services**

Write the following code on the btnDisplayName\_click event of the Default.aspx.vb page:

### Example

```
Protected Sub btnDisplayName Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnDisplayName.Click
    Dim s1 As New localhost.Service
    lblGreetUser.Text = s1.GreetUser(txtName.Text)
End Sub
```