

.NET FRAMEWORK & ASP.NET

Compiled By: Afaq Alam Khan

.Net Framework

- The .Net Framework is a software framework that runs primarily on Microsoft windows.
- It Includes a large library and supports several programming languages which allow language interoperability (each language can use code written in other language).
- The .Net library is available to all the programming languages that .Net supports .
- Programmes written for the .Net Framework execute in software environment known as **common language runtime (CLR)**, an application virtual machine that provides important services such as security, memory management and exception handling.
- CLR includes the Common Type System (CTS) for cross-language type compatibility and the Common Language Specification (CLS) for ensuring that third-party libraries can be used from all .NET-enabled languages.

- To support hardware and operating-system independence, Microsoft developed the Microsoft Intermediate Language (MSIL, or just IL). IL is a CPU-independent machine language style instruction set into which .NET Framework programs are compiled.
- IL programs are compiled to the actual machine language on the target platform prior to execution (known as just-in-time, or JIT, compiling).
- .Net Framework is freely distributed as part of OS
- .Net Framework is not O.S
- .Net Framework is not a programming Language

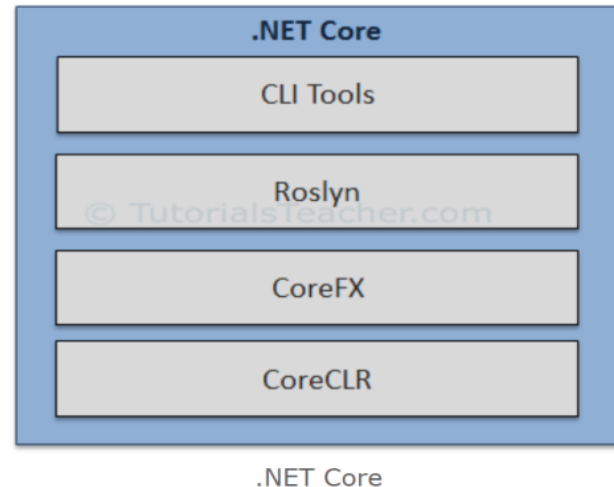
Dot Net Core

- **Dot Net Core:** .NET Core is a new version of .NET Framework, which is a free, open-source, general-purpose development platform maintained by Microsoft
- **.Net Core Characteristics:**
 - Open source
 - Cross Platform
 - Consistent across Architectures
 - Wide Range of applications
 - Support Multiple Languages
 - Modular Architecture
 - Flexible Deployment
 - Compatibility

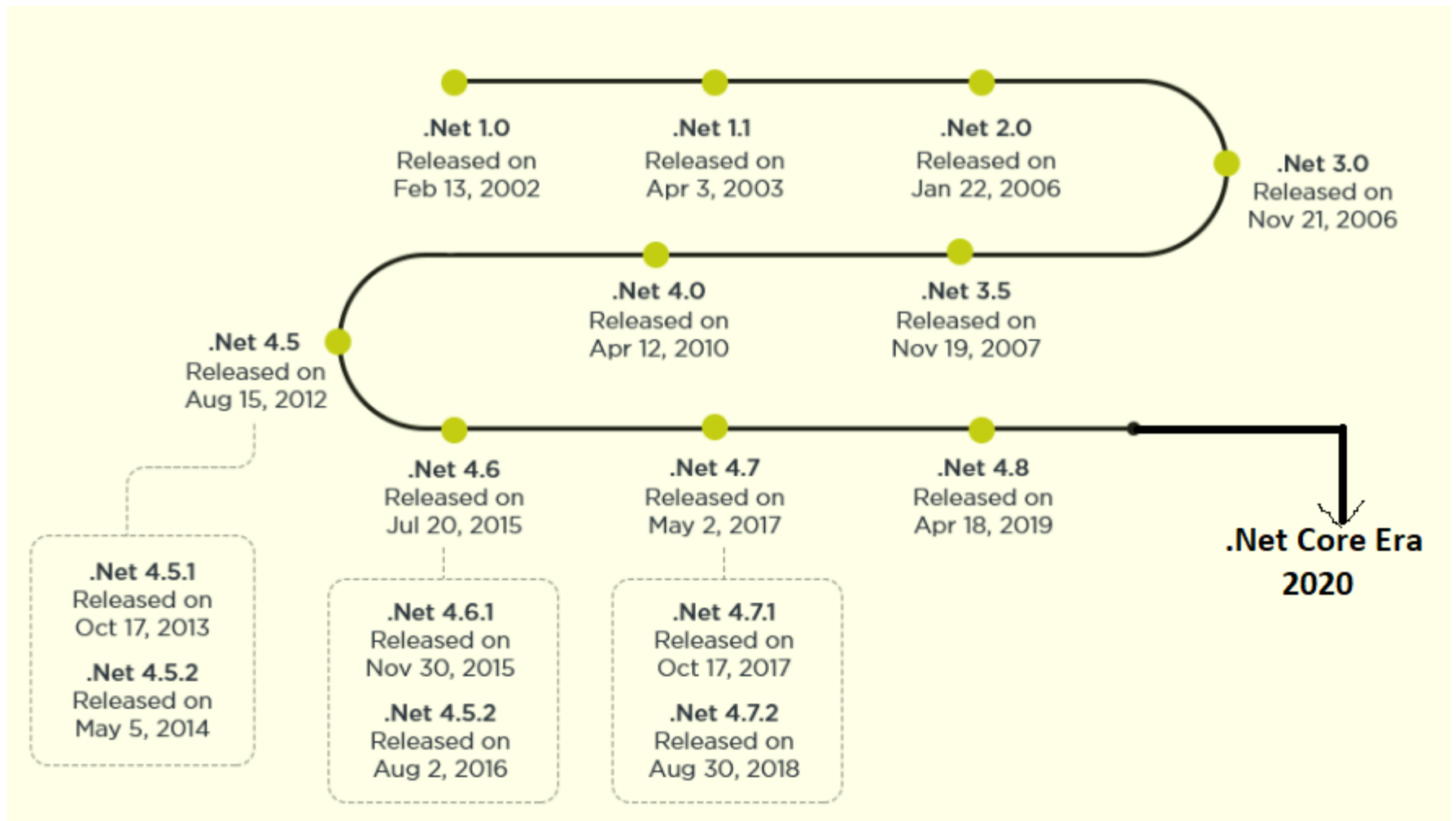
.Net Core Composition

- CLI Tools: A set of tooling for development and deployment.
- Roslyn: Language compiler for C# and Visual Basic
- CoreFX: Set of framework libraries.
- CoreCLR: A JIT based CLR (Command Language Runtime).

{<https://www.tutorialsteacher.com/core/dotnet-core>}



Journey of .Net Framework



Release History

Version No	CLR	Date
1.0	1.0	2002
1.1	1.1	2003
2.0	2.0	2005
3.0	2.0	2006
3.5	2.0	2007
4.0	4	2010
4.5	4	2012
4.5.1	4	2013
4.5.2	4	2014

Version No	CLR	Date
4.6	4	2015
4.6.1	4	2016
4.6.2	4	2016
4.7.2	4	30/4/2018
4.8	4	18/4/2019

Version (Core)	Date
.Net Core 1.1.13	14/4/2019
.Net Core 2.1.17	20/3/2020
.Net Core 3.1.3	24/3/2020
.Net 5	Preview

.Net Products

- Microsoft Visual Studio .NET represents the best development environment for the .NET platform.
- It allows developing applications of the following types:
- **Web Services:** Components that can be accessed over the Internet very easily.
- **Web Forms:** HTML based applications (Web Sites).
- **Windows Forms:** Rich Windows GUI applications. Windows form applications can take advantage of controls, mouse and keyboard events and can talk directly to the underlying OS.
- **Windows Console Applications:** Compilers, utilities and tools are typically implemented as console applications.
- **Windows Services:** It is possible to build service applications controllable via the Windows Service Control Manager (SCM) using the .NET Framework.
- **Component Library:** .NET Framework allows you to build standalone components (types) that may be easily incorporated into any of the above mentioned application types.

Features of .Net [1]

- **Rich Functionality out of the box:** Contains hundreds of classes that provide variety of functionality ready to use in applications.
- **Easy development of web applications:** ASP.NET is a technology available on .NET platform which provides an event driven programming model, similar to Visual Basic 6, that simplifies development of web pages (now called as web forms) with complex user interface.
- **OOPs Support:** The philosophy of .NET is – “Object is mother of all.” Languages like Visual Basic.NET now support many of the OO features that were lacking traditionally. Even primitive types like integer and characters can be treated as objects – something not available even in OO languages like C++.

Features of .Net [2]

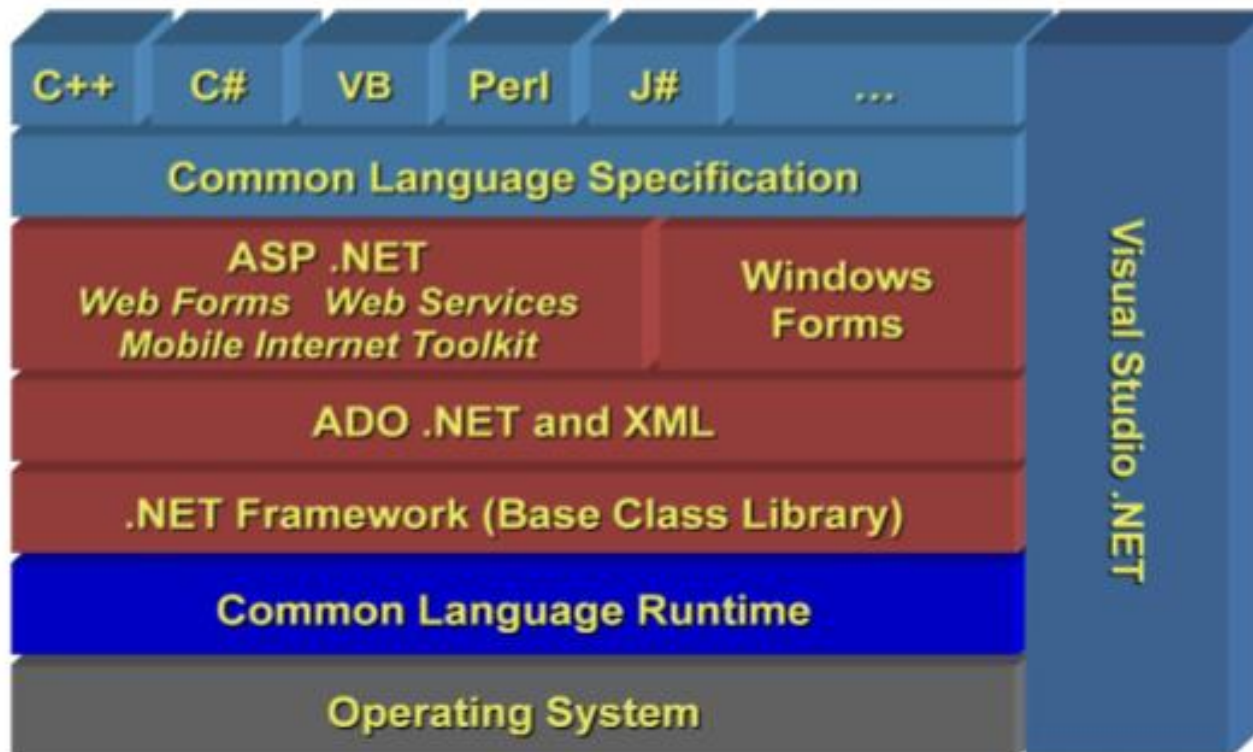
- **Multi-Language Support:** Currently four languages are available right out of the box namely – Visual Basic.NET, C# (pronounced as C-sharp), Jscript.NET and Managed C++ (a dialect of Visual C++). The beauty of multi language support lies in the fact that even though the syntax of each language is different, the basic capabilities of each language remain at par with one another.
- **Multi-Device Support :** Mobile and wireless devices such as PDAs, mobiles and handheld PCs, Smartphones have gained popularity. .NET provides promising platform for programming such devices. .NET Compact Framework and Mobile Internet Toolkit are step ahead in this direction.
- **Automatic memory management:** .NET handles memory management on its own, thus relieving the developer. The garbage collector takes care of freeing unused objects at appropriate intervals.

Features of .Net [3]

- **Compatibility with COM and COM+:** Before the introduction of .NET, COM was the de-facto standard for componentized software development. .NET still relies on COM+ for features like transaction management and object pooling.
- **No more DLL Hell:** .NET ends DLL hell by allowing applications to use their own copy of dependent DLLs. Also, .NET components do not require any kind of registration in system registry.
- **Strong XML support:** .NET tries to harness power of XML in every possible way. In addition to providing support for manipulating and transforming XML documents, .NET provides XML web services that are based on standards like HTTP, XML and SOAP.
- **Security:** Microsoft has taken great efforts to make .NET platform safe and secure for enterprise applications. Features such as type safety, code access security and role based authentication make overall application more robust and secure.

Architecture

- .Net Framework consists of CLR, Class Library CLS and other components as shown:



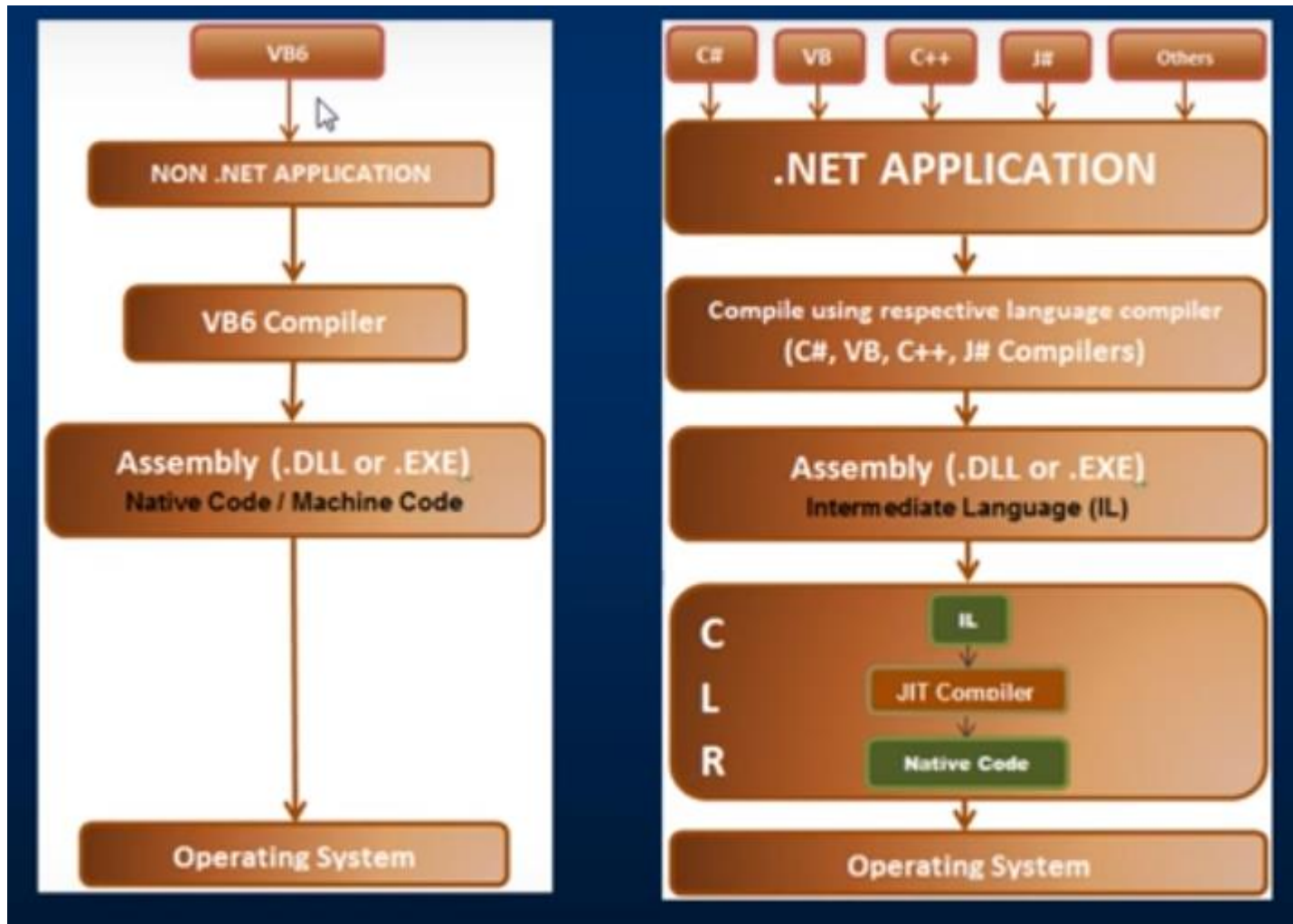
Components

- Common Language Run Time

Any .Net Language → **LC** → MSIL → **JIT** → Executable Native Code

- Class Libraries
- ASP.Net/ Web Forms
- Language Support
- ADO.Net

Application Execution



Assemblies[1]

- When we compile managed code we get an assembly.
- Output looks like EXE or DLL
 - ▣ Actually contains compiled IL and information about assembly called Metadata
 - ▣ Metadata contains manifest that describes the Assembly, procedures and types it exports and other assemblies it requires
- Metadata
 - ▣ Version, name, culture, security requirement, list of other files in assembly, list of external required reference.

Assembly [2]

- There are two types of Assemblies:
 - ▣ **Private:** can be used by its own application and its location is normally
ApplicationName/bin/debug/pvtAssemblyName
 - ▣ **Public:** can be used for multiple applications and its location is c/windows/assembly
 - Following steps are to be followed while creating public assembly
 - Create an assembly
 - Generate strong name key and associate it with the assembly
 - Place assembly in Global Access Cache (GAC)

- **ILDASM:** tool to de assemble our application



ASP.Net

[WebForms]

ASP.Net

- ASP.NET is a technology for developing Web applications based on the Microsoft .NET Framework.
 - It was first released in January 2002 with version 1.0 of the .NET Framework, and is the successor to Microsoft's Active Server Pages (ASP) technology.
 - ASP.NET is built on the Common Language Runtime (CLR), allowing programmers to write ASP.NET code using any supported .NET language, such as C# and VB.NET.
 - ASP.NET web pages or webpage, known officially as **Web Forms**, are the main building block for application development. Web forms are contained in files with an “.aspx” extension.
- Like its predecessor (ASP), ASP.NET is server-side i.e., it runs on the Web server.
 - Instead of being interpreted by the client, server-side code is interpreted by the Web server.
 - In the case of ASP.NET, the code in the page is read by the server and used dynamically to generate standard HTML/JavaScript/CSS that is then sent to the browser.
 - As all processing of ASP.NET code occurs on the server, it's called a server-side technology.

Programming Models

- ASP.NET supports a number of programming models for building web applications some of the popular models are:
 - ASP.Net WebForms ←
 - ASP.Net MVC
 - ASP.Net Web Pages
 - ASP.Net Web API
 - ASP.Net Core (MVC)

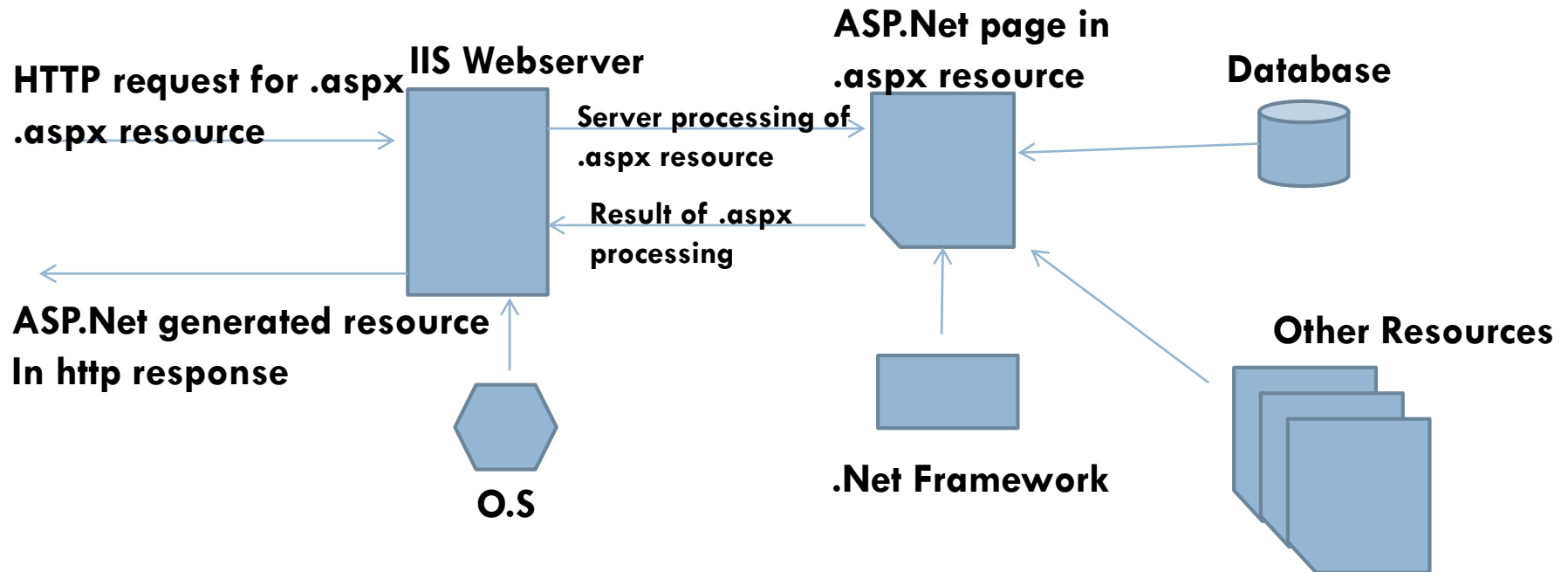
Features

- **ASP.NET lets you use your favorite programming language, or at least one that's really close to it. The .NET Framework currently supports over twenty languages, four of which may be used to build ASP.NET Websites.**
- **ASP.NET pages are compiled, not interpreted. Instead of reading and interpreting your code every time a dynamic page is requested, ASP.NET compiles dynamic pages into efficient binary files that the server can execute very quickly.**
- **ASP.NET has full access to the functionality of the .NET Framework. Support for XML, Web Services, database interaction, email, regular expressions, and many other technologies are built right into .NET, which saves you from having to reinvent the wheel.**
- **ASP.NET allows you to separate the server-side code in your pages from the HTML layout. When you're working with a team composed of programmers and design specialists, this separation is a great help, as it lets programmers modify the server-side code without stepping on the designers' carefully crafted HTML—and vice versa.**

Execution of ASP.Net Files

- ❑ The browser contacts the Web server specified in the address URL and makes a request for the page by formulating a HTTP request, which is sent to the Web server.
- ❑ The Web server on receiving the request determines the file type requested and passes processing to the appropriate handler.
- ❑ ASP.NET files are compiled, if necessary, into .NET Page classes and then executed, with the results sent to the client's browser.
 - **Compilation means that on first load ASP.NET applications take longer to display than previous versions of ASP, but once compiled they are noticeably faster.**

Execution of ASP.Net Files



Compiling and Delivering ASP.NET Pages

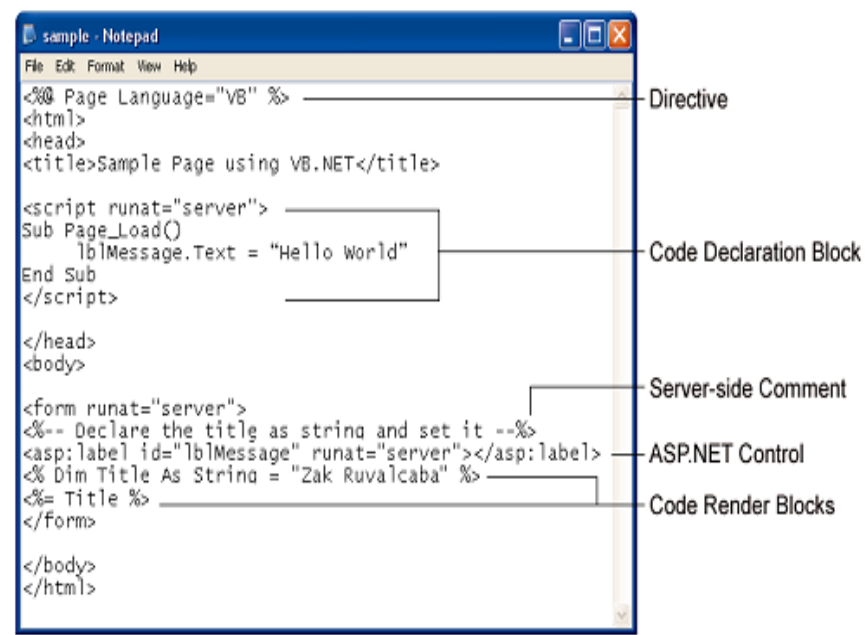
- The process of compiling and delivering ASP.NET pages goes through the following stages:
 - 1. IIS matches the URL in the request against a file on the physical file system (hard disk) by translating the virtual path (for example, /site/ index.aspx) into a path relative to the site's Web root (for example, d:\domains\thisSite\wwwroot\site\index.aspx).
 - 2. Once the file is found, the file extension (.aspx) is matched against a list of known file types for either sending on to the visitor or for processing.
 - 3. If this is first visit to the page since the file was last changed, the ASP code is compiled into an assembly using the Common Language Runtime compiler, into MSIL, and then into machine-specific binary code for execution.
 - 4. The binary code is a .NET class .dll and is stored in a temporary location.
 - 5. Next time the page is requested the server will check to see if the code has changed. If the code is the same, then the compilation step is skipped and the previously compiled class code is executed; otherwise, the class is deleted and recompiled from the new source.

- 6. The compiled code is executed and the request values are interpreted, such as form input fields or URL parameters.
- 7. If the developer has used Web forms, then the server can detect what software the visitor is using and render pages that are tailored to the visitors requirements, for example, returning Netscape specific code, or Wireless Markup Language (WML) code for mobiles.
- 8. Any results are delivered back to the visitor's Web browser.
- 9. Form elements are converted into client side markup and script,HTML and JavaScript for Web browsers,and WML and WMLScript for mobiles, for example.

ASP.NET Page Structure

- ASP.NET pages are simply text files with the .aspx file name extension that can be placed on an IIS server equipped with ASP.NET.
- When a browser requests an ASP.NET page, the ASP.NET runtime parses and compiles the target file into a .NET Framework class.
- An ASP.NET page consists of the following elements:

- **Directives**
- **Code declaration blocks**
- **Code render blocks**
- **ASP.NET server controls**
- **Server-side comments**
- **Server-side include directives**
- **Literal text and HTML tags**
- **Client-side scripts**



The screenshot shows a Notepad window titled 'sample - Notepad' containing the following ASP.NET code:

```
<%@ Page Language="VB" %>
<html>
<head>
<title>Sample Page using VB.NET</title>
<script runat="server">
Sub Page_Load()
    lblMessage.Text = "Hello World"
End Sub
</script>
</head>
<body>
<form runat="server">
<!-- Declare the title as string and set it -->
<asp:label id="lblMessage" runat="server"></asp:label>
<% Dim Title As String = "Zak Ruvalcaba" %>
<%= Title %>
</form>
</body>
</html>
```

Annotations on the right side of the code block identify the following elements:

- Directive**: Points to the opening page directive `<%@ Page Language="VB" %>`.
- Code Declaration Block**: Points to the `Sub Page_Load()` block.
- Server-side Comment**: Points to the `<!-- Declare the title as string and set it -->` comment.
- ASP.NET Control**: Points to the `<asp:label id="lblMessage" runat="server"></asp:label>` control.
- Code Render Blocks**: Points to the `<% Dim Title As String = "Zak Ruvalcaba" %>` and `<%= Title %>` lines.

ASP.Net Page Structure

□ Directives

- Page directives are used to set various attributes about a page. The ASP Engine and the compiler follow these directives to prepare a page.
- Directives start with the sequence `<%@`, followed by the directive name, plus any attributes and their corresponding values, then end with `%>`.
- There are many kinds of directives. The most frequently ones are the following: `@Page`, `@Import`, `@Implements`, `@Register`, `@OutputCache` and `@Assembly` directives. These directives can be placed anywhere in a page, however, these are typically placed at the top.

@ Page

We may use this directive to declare many page-related attributes about a particular page. For example, we use this directive to declare the language to be used in a page, such as `<%@ Page Language="VB" Debug="true" %>` page. There are numerous attributes of this directive. Some of the frequently used ones are these: *AutoEventWireup*, *Buffer*, *ClientTarget*, *EnableSessionState*, *ErrorMessage*, *Debug*, *Trace*, *TraceMode*, and so on.

@ Import

We use this directive to import a namespace in the page class file. For example, in the following directive, we are importing the *System.Data.OleDb* namespace in our page: `<%@ Import Namespace="System.Data.OleDb" %>`.

ASP.Net Page Structure

- **Code Declaration Blocks**

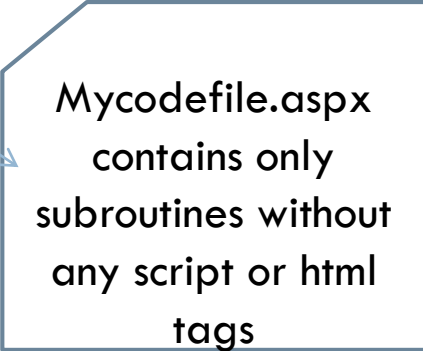
Code declaration blocks are one method to contain all the application logic of your ASP.NET page. This application logic defines variables, subroutines, functions, and more. In our page, we place the code inside `<script>` tags.

```
VB.NET
```

```
<script runat="server" language="VB" src="mycodefile.vb">
```

```
VB.NET
```

```
<script runat="server">  
Sub mySub()  
  ' Code here  
End Sub  
</script>
```



Mycodefile.aspx
contains only
subroutines without
any script or html
tags

Code within a code declaration block is only executed when it is called or triggered by user or page interactions.

ASP.Net Page Structure

- **Code Render Blocks**

- ▣ You can use code render blocks to define inline code or inline expressions that execute when a page is rendered.
- ▣ Code within a code render block is executed immediately as it is encountered, usually when the page is loaded or rendered for the first time, and every time the page is loaded subsequently

Inline code: <%	%>
Inline expression: <%=	%>

Inline code in asp.net cannot contain any function

ASP.Net Page Structure

- **Server-Side Comments**

`<%--and --%>`

- **Server-Side Include Directives**

Server-side include directives enable developers to insert the contents of an external file anywhere within an ASP.NET page.

```
<!-- #INCLUDE file="myinclude.aspx" -->
```

Asp.Net Page life cycle

<https://msdn.microsoft.com/en-us/library/ms178472.aspx>

Controls

- Controls are small building blocks of the graphical user interface, which include text boxes, buttons, check boxes, list boxes, labels, and numerous other tools. Using these tools, the users can enter data, make selections and indicate their preferences.
- Controls are also used for structural jobs, like validation, data access, security, creating master pages, and data manipulation.
- **Web Server Controls (ASP.Net Server Controls)**
- Custom Controls

ASP.Net Server Controls

- ASP.NET server controls are the primary controls used in ASP.NET.

Syntax: **<asp:control_name id="some_id" runat="server" />**

- ASP.NET server controls with a visual aspect are derived from the WebControl class and inherit all the properties, events, and methods of this class.
- The WebControl class itself and some other server controls that are not visually rendered are derived from the System.Web.UI.Control class.
- ASP.Net server controls inherit all properties, events, and methods of the WebControl and System.Web.UI.Control class.
- For the full list of common properties and methods visit

https://www.tutorialspoint.com/asp.net/asp.net_server_controls.htm

ASP.Net Server Controls

- Standard Controls

- Validation

- Data

- Navigation

- Login

- Webparts

- .

- .

Menu
SiteMapPath
TreeView

Compare validator
Custom validator
Range validator
Regular expression validator
Required field
Validation summary

Gridview
Datalist
Listview
Repeater

.
.

Label
Button
Checkbox
Checkboxlist
Dropdownlist
Hyperlink
Image
Imagebutton
Listbox
Linkbutton
Imagebutton
Radiobutton
Radiobuttonlist

.
.
.

Standard Controls

- TextBox
- Button
- Hyperlink
- Checkbox
- Radiobutton

TextBox

- This is an input control which is used to take user input.

```
<asp:TextBox ID="TextBox1" runat="server" ></asp:TextBox>
```

Properties:

ID: specifies the identity of the control

Text: It is used to set text to be shown for the control.

Visible: To set visibility of control on the form. (True/False)

BorderColor: It is used to set border color of the control.

MaxLength:It is used to set maximum number of characters that can be entered.

ReadOnly:It is used to make control readonly. (True/False)

TextMode: SigleLine/Multiline/password/Date

AutoPostBack: If set to TRUE, will cause the page to postback if the text inside textbox is changed

EnableViewState: It is true by default. If set to false, textbox will not retain the contents after postback

-
-
-

Event: onTextChanged : will be invoked once the text inside the textbox is changed

Button

- ASP.NET provides three types of button control:
 - **Button** : It displays text within a rectangular area.
 - **Link Button** : It displays text that looks like a hyperlink.
 - **Image Button** : It displays an image.

Syntax: **<asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="Click" / >**

Properties:

Text: The text displayed on the button. This is for button and link button controls only.

ImageUrl: For image button control only. The image to be displayed for the button.

PostBackUrl: The URL of the page that is requested when the user clicks the button.

CommandName: A string value that is passed to the command event when a user clicks the button.

CommandArgument: A string value that is passed to the command event when a user clicks the button.

.
. .
.

Event : **onclick**

Hyperlink

- Control to create hyperlink

```
<asp:HyperLink ID="HyperLink1 "  
  runat="server">HyperLink</asp:HyperLink>
```

Properties:

Text: hyperlink text

NavigateUrl: url of the destination webpage

Target: _blank/_parent/_top/...

Suitable for query strings

List Controls

- CheckBoxList / ListBox
- DropDownList / RadioButtonList

DropDownList

- The DropDownList is a ASP.Net server control which is used to create an HTML Select component. It allows us to select an option from the dropdown list. It can contain any number of items

```
<asp:DropDownList ID="DropDownList1" runat="server" >
```

```
    <asp:ListItem>Korea </asp:ListItem>
```

```
    <asp:ListItem>Japan</asp:ListItem>
```

```
    <asp:ListItem>Palestine</asp:ListItem>
```

```
    <asp:ListItem>France</asp:ListItem>
```

```
</asp:DropDownList>
```



Korea	▼
Korea	
Japan	
Palestine	
France	

Properties:

Items: collection of items. DropDownList control can be populated at design time using items collection

Autopostback: if set to true, page will be posted back once the item is selected

EnableViewState: True by default. Retains the selected option after postback.

DataSource: DropDownList control can be bound to some datasource. This property is used to specify the datasource

Datamember: Can be the name of table

DateTextField: Specified the name of field from the table. Data items in the selected column will appear as options in the DropDownList control

DataValueField:

DropDownlist-Example

DropDownExample.aspx

```
.  
.br/>Selected Country:<asp:Label ID="Label1"  
  runat="server" Text=""></asp:Label>
```

```
Country: <asp:DropDownList ID="DropDownList1"  
  runat="server">
```

```
<asp:ListItem>Korea </asp:ListItem>
```

```
<asp:ListItem>Japan</asp:ListItem>
```

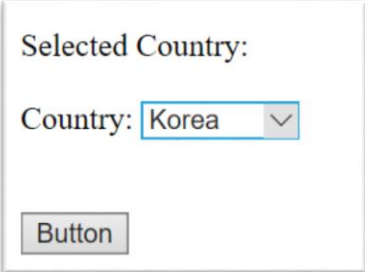
```
<asp:ListItem>Palestine</asp:ListItem>
```

```
<asp:ListItem>France</asp:ListItem>
```

```
</asp:DropDownList>
```

```
<asp:Button ID="Button2" runat="server"  
  Text="Button" />
```

```
.  
.
```

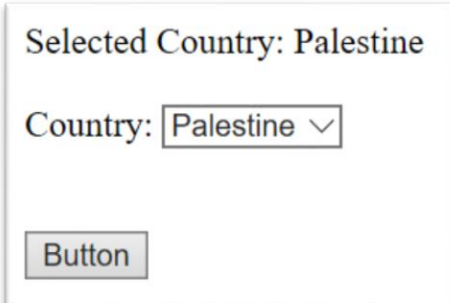


Selected Country:
Country: Korea ▾
Button

Code Behind

DropDownExample.vb

```
Label1.Text =  
  DropDownList1.SelectedItem.ToString
```



Selected Country: Palestine
Country: Palestine ▾
Button

CheckBoxList

- CheckBoxlist control allows a user to select multiple options.

```
<asp:CheckBoxList ID="CheckBoxList1" runat="server">  
  <asp:ListItem>Painting</asp:ListItem>  
  <asp:ListItem>Reading</asp:ListItem>  
  <asp:ListItem>Gardning</asp:ListItem>  
  <asp:ListItem>surfing</asp:ListItem>  
</asp:CheckBoxList>
```

```
<asp:Button ID="Button3" runat="server" Text="Button" />
```

Selected Hobbies are:

Painting
 Reading
 Gardning
 surfing

Button

Codebehind

Selected Hobbies are: Reading surfing

- Painting
 Reading
 Gardning
 surfing

Button

```
Dim strHobby As String = ""  
For Each li As ListItem In CheckBoxList1.Items  
  If li.Selected Then  
    strHobby = strHobby + " " + li.Text  
  End If  
Next  
Label3.Text = strHobby
```


Exercise

- Ex1: Consider a webform *SelectCountry.aspx* with code behind file enabled. add two DropDownList controls (DDLC1, DDLC2) to the *SelectCountry.aspx*. Populate the DDLC1 control with the list of Countries (Australia, Bangladesh, India, Pakistan). Write a program for the OnSelectedIndexChanged event of DDLC1 on codebehind file which populates DDLC2 control with the name of the cities depending upon the selected country in DDLC1.
- Ex2: Consider a webform *Movemyhobbies.aspx* with code behind file enabled. With the help of code snippet show how can two ListBox controls (LB1, LB2) and a submit button be added to the *Movemyhobbies.aspx*. Populate the LB1 control with the list of hobbies (Reading, Swimming, Gardening, Net surfing, Music). write a program for the onClick event of a button on codebehind file which moves the selected items from LB1 to LB2 and removes those items from LB1

Data Controls

- GridView

- Repeater

-

-

We will discuss these controls
In ADO.Net

Validation Controls

- Required Field Validator Control
- Regular Expression Validator Control
- Compare Validator Control
- Range Validator
- Custom ValidatorControl
- Validation Summary

Validation Controls

- Validation Controls automatically adds both clientside and server side code. If browser is capable of supporting javascript, client side validation scripts are automatically sent to the browser. If the browser is incapable of supporting javascript, the validation routines are automatically implemented in server side code.
- To disable client side form validation
 - `<%@ page clientTarget = "down level" %>`
 - Or
 - Set "EnableClientScript" property of validation control to false
- We can disable validation, both client and server validation, when certain buttons are pushed (e.g cancel button)
 - `<asp:Button id="btnCancel" text="cancel" CausesValidation=False RunAt=Server/>`

Required Field Validation Control

- Used to make sure that control is not submitted blank
- Properties
 - **ControlToValidate**: Specifies the ID of control we want to validate
 - **EnableClientScript**: Enables or Disables client side validation (True by default)
 - **Enabled**: Enables or disables both client and server side validation
 - **Error message**: Message to be displayed in a validation summary when validation is invalid
 - **Text**: Message to be displayed when validation is invalid
 - **IsValid**: Has the value true when the validation check succeeds and false otherwise
 - **Display**: how the validator is rendered to your page (Static/ Dynamic/ None) {whether space for displaying error message is reserved in advanced on the page or will be created on the Go}
- Methods
 - **Validate**: Performs validation and updates IsValid Property

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
```

```
<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" ControlToValidate="TextBox1"  
Text="cannot be blank"> </asp:RequiredFieldValidator>
```

Compare Validator Control

- Used to compare the contents
 - E.g., password and confirm password
- Properties
 - ▣ **ControlToValidate**: Specifies the ID of control we want to validate
 - ▣ **ControlToCompare**: Specifies the ID of control with which comparison is made
 - ▣ **Text**: Error message to be displayed
 - ▣ **Operator**: EQUAL/NOTEQUAL/GREATERTHAN/... (default is EQUAL)
 - ▣ **Type**: datatype to use when comparing values (String/Integer/Double/Date/Currency) (default is String)
 - ▣ **Validation Group**:

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
```

```
<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
```

```
<asp:CompareValidator ID="CompareValidator1" runat="server"  
    ControlToCompare="TextBox2" ControlToValidate="TextBox1"  
    Text="comparison fail"></asp:CompareValidator>
```

Regular Expression Validator Control

- Used to match the value entered into a form field to a regular expression. For example we can match email address, mobile number etc.

- Properties

- ControlToValidate
- ValidationExpression

- Validating email address

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
```

```
<asp:RegularExpressionValidator ID="RegularExpressionValidator1"
  runat="server" ControlToValidate="TextBox1" Text="Wrong email Format"
  ValidationExpression="\w+([-+.']\w+)*@\w+([-.\w+)*\.\w+([-
  .]\w+)*"></asp:RegularExpressionValidator>
```

Custom Validator Control

- Dot Net enables us to handle wide range of validation tasks. However., we cannot perform certain types of validation with the included controls. To enable any type of validation that is not covered by the standard validation controls, we need to use the custom validation control.
- Properties
 - **ControlToValidate:**
 - **ClientValidationFunction:** Specifies the name of the clientside validation function
 - **ValidateEmptyText:** Whether the validator validates the control when the text of the control is empty
- Method
 - **Validate:**
 - **OnServerValidate:** Raises ServerValidate event
- Event
 - **ServerValidate:** Represents the function for performing server side validation

Custom Validator Control

- Example: write a custom validation function to make sure that the string in the textbox contains "CUKmr".

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
```

```
<asp:CustomValidator ID="CustomValidator1"
  ControlToValidate="TextBox1" ClientValidationFunction="myclientcustomfunction"
  OnServerValidate="CustomValidator1_ServerValidate"
  ValidateEmptyText="True"
  Text="Does not contain valid data"
  runat="server"
  </asp:CustomValidator>
```

```
<asp:Button ID="Button1" runat="server" Text="Button" />
```

CustomValidator1_ServerValidate

Subroutine is implemented on code behind file and validation is done server side

```
Protected Sub CustomValidator1_ServerValidate(source As Object, args As ServerValidateEventArgs) Handles CustomValidator1.ServerValidate
```

```
    Dim str1 As String
```

```
    str1 = args.Value
```

```
    If str1.IndexOf("CUKmr") > -1 Then
```

```
        args.IsValid = True
```

```
    Else
```

```
        args.IsValid = False
```

```
    End If
```

```
End Sub
```

```
Protected Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
```

```
    If IsValid Then
```

```
        Label1.Text = "valid data"
```

```
    Else
```

```
        Label1.Text = "try again"
```

```
    End If
```

myclientcustomfunction

Function is written in head section of .aspx file and validation is done on clientside

```
<script type="text/javascript">
    function myclientcustomfunction(oSrc, args)
    {

        str1=args.Value
            if (str1.indexOf("CUKmr") > -1)
                args.IsValid = true;
            else
                args.IsValid = false;

    }
</script>
```

Validation Summary

- Displays a report of all validation errors occurred in a Web page (All error messages together)
- Properties
 - DisplayMode— Enables you to specify how the error messages are formatted. Possible values are BulletList, List, and SingleParagraph.
 - HeaderText— Enables you to display header text above the validation summary.
 - ShowMessageBox— Enables you to display a popup alert box (By Default False)
 - ShowSummary— Enables you to hide the validation summary in the page (By default True)

Validation Summery - Example

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
```

```
<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"  
  ControlToValidate="TextBox1" Display="None" ErrorMessage="Cannot be  
  empty"> </asp:RequiredFieldValidator>
```

```
<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
```

```
<asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>
```

```
<asp:CompareValidator ID="CompareValidator1" runat="server"  
  ControlToCompare="TextBox2" ControlToValidate="TextBox3"  
  Display="None" ErrorMessage="Password miss match">  
</asp:CompareValidator>
```

```
<asp:ValidationSummary ID="ValidationSummary1" runat="server"  
  HeaderText="All Errors together" ShowMessageBox="True"  
  ShowSummary="False" />
```

Exercise

- ❑ Validate a form when submit button is clicked and no validation should be performed when cancel button is pressed.
- ❑ Using Regular expression validator, validate username which starts with an alphabet and should not contain “ . ” at the end.
- ❑ Working with “validationGroup” property of validation controls

Navigation in ASP.Net

- Navigation Controls
 - Hyperlink Control
 - Menu
 - Treeview

In addition to above controls there are other options which are used to move the control from one webpage to another:

- HTML Anchor Tag
- Response.redirect
- Server.Transfer
- Crosspage Postback

- <https://msdn.microsoft.com/en-us/library/ms973231.aspx>
- <http://vb.net-informations.com/framework/assembly.htm>
- <https://www.c-sharpcorner.com/article/difference-between-net-framework-and-net-core/>