

Unit 4: Essentials of ASP.NET using VB.NET

1. Working with Web Forms

ASP.NET Web Forms is a framework for developing dynamic web applications. It provides server controls, events, validation, navigation, and state management.

A Web Form generally consists of:

- .aspx file → User interface/design
- .aspx.vb file → VB.NET code-behind
- Web.config → Application configuration

Example

Default.aspx

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
Inherits="_Default" %>
```

```
<form id="form1" runat="server">
```

```
    <asp:Label ID="lblMessage" runat="server" Text="Enter your
name:"></asp:Label>
```

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

```
    <asp:Button ID="btnSubmit" runat="server"
```

```
        Text="Submit" />
```

```
</form>
```

Default.aspx.vb

```
Partial Class _Default
```

```
    Inherits System.Web.UI.Page
```

```
    Protected Sub btnSubmit_Click(sender As Object, e As EventArgs) _
```

```
        Handles btnSubmit.Click
```

```
        lblMessage.Text = "Hello " & txtName.Text
```

```
    End Sub
```

```
End Class
```

2. Directory Structure in ASP.NET

An ASP.NET application contains several important files and folders.

File/Folder	Purpose
.aspx	Web Form/user interface
.aspx.vb	VB.NET code-behind file
Web.config	Configuration settings
Global.asax	Application-level events
App_Code	Shared VB.NET classes
App_Data	Database/XML/data files
Bin	Compiled DLL files
App_Themes	Themes and skins
Scripts	JavaScript files

Important folders

App_Code:

Contains reusable classes and business logic.

App_Data:

Stores application data such as databases and XML files.

Bin:

Contains compiled assemblies (.dll files).

App_Themes:

Contains themes and skins used to control the appearance of Web Forms.

3. ASP.NET Compilation Model

ASP.NET uses a compilation process to convert Web Form code into executable code.

Process

.aspx / .aspx.vb



ASP.NET Compilation



Intermediate/compiled code



.NET Assembly



CLR executes code



HTML response



Browser

Dynamic Compilation

ASP.NET can compile a Web Form when it is requested for the first time or when the source code changes.

Precompilation

An application can also be compiled before deployment.

Advantages

- Detects compilation errors
- Improves performance after compilation
- Supports VB.NET and other .NET languages
- Produces reusable assemblies

4. Code-Behind Model

The **Code-Behind Model** separates the presentation/design from the programming logic.

ASPX file

```
<asp:Button ID="btnClick" runat="server" Text="Click Me" />
```

VB.NET code-behind

```
Protected Sub btnClick_Click(sender As Object, e As EventArgs) _
```

```
Handles btnClick.Click
```

```
Response.Write("Button clicked!")
```

```
End Sub
```

Advantages

1. Separates UI and logic.
 2. Makes maintenance easier.
 3. Makes debugging easier.
 4. Improves code organization.
 5. Allows developers and designers to work separately.
-

5. Working with Web Form Controls

Web Form controls are server-side controls used to create interactive web pages.

They normally use:

```
runat="server"
```

Common controls

- Label
- TextBox
- Button
- CheckBox
- RadioButton
- DropDownList
- ListBox
- Image
- HyperLink

Example

```
<asp:Label ID="lblName" runat="server" Text="Name:"></asp:Label>
```

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

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

VB.NET event

```
Protected Sub btnSubmit_Click(sender As Object, e As EventArgs) _
```

```
    Handles btnSubmit.Click
```

```
    lblName.Text = "Welcome " & txtName.Text
```

```
End Sub
```

6. Navigation Controls

ASP.NET provides controls for navigating between different pages.

A. Menu Control

Used to create menus.

```
<asp:Menu ID="Menu1" runat="server">
```

```
    <Items>
```

```
        <asp:MenuItem Text="Home"           NavigateUrl="~/Home.aspx" />
```

```
        <asp:MenuItem Text="About"
```

```
            NavigateUrl="~/About.aspx" />
```

```
        <asp:MenuItem Text="Contact"
```

```
            NavigateUrl="~/Contact.aspx" />
```

```
    </Items>
```

```
</asp:Menu>
```

B. TreeView Control

Displays hierarchical navigation.

Example:

Home

├── Products

| ├── Mobile

| └── Laptop

└── Contact

C. SiteMapPath

Displays the user's current location in the website.

Example:

Home > Products > Laptop

It commonly uses a Web.sitemap file.

7. Validation Controls

Validation controls are used to check whether the data entered by the user is valid.

Main validation controls

Control	Purpose
RequiredFieldValidator	Checks that a value is entered
CompareValidator	Compares two values
RangeValidator	Checks a value within a range
RegularExpressionValidator	Checks a specific pattern
CustomValidator	Performs custom validation
ValidationSummary	Displays validation errors

RequiredFieldValidator

Checks whether the user has entered a value.

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

```
<asp:RequiredFieldValidator ID="rfvName" runat="server"
```

```
    ControlToValidate="txtName" ErrorMessage="Name is required">
```

```
</asp:RequiredFieldValidator>
```

CompareValidator

Used to compare two controls.

For example, confirming a password:

```
<asp:TextBox ID="txtPassword" runat="server"
```

```
TextMode="Password"></asp:TextBox>
<asp:TextBox ID="txtConfirm" runat="server"
TextMode="Password"></asp:TextBox>
<asp:CompareValidator ID="cvPassword" runat="server"
ControlToValidate="txtConfirm" ControlToCompare="txtPassword"
ErrorMessage="Passwords do not match">
</asp:CompareValidator>
```

RangeValidator

Checks whether a value is within a specified range.

```
<asp:TextBox ID="txtAge" runat="server"></asp:TextBox>
<asp:RangeValidator
ID="rvAge" runat="server" ControlToValidate="txtAge"
MinimumValue="18"
MaximumValue="60"
Type="Integer"
ErrorMessage="Age must be between 18 and 60">
</asp:RangeValidator>
```

RegularExpressionValidator

Checks whether data matches a specified pattern.

Example for an email:

```
<asp:RegularExpressionValidator
ID="revEmail"
runat="server"
ControlToValidate="txtEmail"
ValidationExpression="^[^@\s]+@[^@\s]+\.[^@\s]+$"
ErrorMessage="Enter a valid email">
</asp:RegularExpressionValidator>
```

CustomValidator

Used when built-in validators are not sufficient.

ASPX

```
<asp:CustomValidator
    ID="cvNumber"
    runat="server"
    ControlToValidate="txtNumber"
    OnServerValidate="cvNumber_ServerValidate"
    ErrorMessage="Enter an even number">
</asp:CustomValidator>
```

VB.NET

```
Protected Sub cvNumber_ServerValidate(
    source As Object,
    args As ServerValidateEventArgs)
    Dim number As Integer = Convert.ToInt32(args.Value)
    If number Mod 2 = 0 Then
        args.IsValid = True
    Else
        args.IsValid = False
    End If
End Sub
```

8. Validation Groups

ValidationGroup allows different controls on the same page to have separate validation.

For example, suppose a page has **Login** and **Registration** forms. Clicking Login should validate only Login fields.

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

```
<asp:RequiredFieldValidator
    ID="rfvUsername"
    runat="server"
    ControlToValidate="txtUsername"
    ValidationGroup="Login"
    ErrorMessage="Username is required">
</asp:RequiredFieldValidator>
<asp:Button ID="btnLogin" runat="server"
    Text="Login"
    ValidationGroup="Login" />
```

Only validators belonging to the Login group are triggered by the Login button.

9. Client-Side Validation

Client-side validation is performed in the user's browser, usually with JavaScript.

Advantages

- Fast response
- No server request is normally required
- Provides immediate feedback
- Reduces server workload

Example

If a required textbox is empty, the browser can display:

Name is required.

without sending the form to the server.

Limitation

Client-side validation can be bypassed or disabled, so it should **not be relied upon for security**.

10. Server-Side Validation

Server-side validation is performed on the web server.

In ASP.NET, we can check:

```
If Page.IsValid Then
```

```
    'Process the data
```

```
End If
```

Example:

```
Protected Sub btnSubmit_Click(sender As Object, e As EventArgs) _
```

```
    Handles btnSubmit.Click
```

```
    If Page.IsValid Then
```

```
        lblMessage.Text = "Data is valid."
```

```
    Else
```

```
        lblMessage.Text = "Please correct the errors."
```

```
    End If
```

```
End Sub
```

Advantages

- More reliable
- Cannot be bypassed simply by disabling JavaScript
- Important before storing or processing user data

Disadvantage

- Requires communication with the server
- Can be slower than client-side validation

11. Client-Side vs Server-Side Validation

Client-Side	Server-Side
Runs in browser	Runs on server
Faster	Relatively slower
Usually no server request	Requires server request
Gives immediate feedback	Feedback after server processing

Client-Side	Server-Side
Can be bypassed	More reliable
Usually uses JavaScript	Uses VB.NET/server code

Important exam point

Use client-side validation for quick feedback, but always perform server-side validation before trusting or processing important user input.
