Developing ASP.NET Web Form Applications

By

Narasimha Rao T

Microsoft.Net FSD Trainer

tnrao.trainer@gmail.com

1.1 What is ASP.NET?

- ASP.NET is a web application framework developed by Microsoft for building dynamic web pages, web applications, and web services.
- Runs on the .NET Framework (or .NET Core/ASP.NET Core for cross-platform).
- Allows developers to build server-side web applications using languages like C# or VB.NET.
- Integrates easily with HTML, CSS, JavaScript, and SQL Server.
- Offers security, scalability, and performance for enterprise-level applications.

1.2 ASP.NET Application Types:

- Web Forms
- MVC (Model-View-Controller)
- Web API
- Razor Pages
- Blazor (for interactive SPA)

1.3 What is ASP.NET Web Forms?

- **ASP.NET Web Forms** is a **web application development model** under the **ASP.NET framework**
- It allows developers to build **dynamic, interactive web applications** with a **visual, drag-and-drop interface** using **server controls**.
- It mimics the event-driven model of Windows desktop applications but runs over the web.

2. Characteristics of Web Forms

- **Page-centric**: Each `.aspx` page acts as a separate unit of functionality.
- **Event-driven model**: Similar to Windows Forms (WinForms), it uses events like `Page_Load`, `Button_Click`, etc.
- **Stateful behavior** using **ViewState**: Maintains control state across HTTP requests.
- **Server controls** abstract HTML elements and provide additional functionality.

3. Key Components of a Web Forms Application

Component	Description
`.aspx` file	Front-end markup (HTML + Web controls)
CodeBehind(`.aspx.cs`)	Backend logic (event handlers, data manipulation)
Server Controls	ASP.NET-specific controls like `<asp:TextBox>`, `<asp:GridView>`
ViewState	Hidden field that stores page and control state
Postback	Mechanism to send data back to the same page for processing
Master Pages	Templates to define common layout across pages
Web.config	Configuration settings (security, database, etc.)

4. Structure of a Web Form Page

Example: `Sample.aspx`

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

<html>
  <body>
    <form runat="server">
      <asp:TextBox ID="txtName" runat="server" />
      <asp:Button ID="btnSubmit" Text="Submit" runat="server" OnClick="btnSubmit_Click" />
      <asp:Label ID="lblGreeting" runat="server" />
    </form>
  </body> </html>
```

Example: `Sample.aspx.cs` (Code-Behind)

```
protected void btnSubmit_Click(object sender, EventArgs e) {
    lblGreeting.Text = "Hello, " + txtName.Text;
```

5. ASP.NET Page Lifecycle

Understanding the lifecycle is **crucial** for Web Form development. Following table provides Lifecycle Stages:

Stage	Description
1. Page Request	Client requests the page
2. Start	Page properties initialized
3. Initialization	Controls initialized (but not loaded with data yet)
4. Load	ViewState restored, control data loaded
5. Postback Event Handling	Button clicks and other events are processed
6. Rendering	Page is rendered into HTML and sent to browser

6. What is ViewState?

- A **mechanism to preserve the state** of server controls between postbacks.
- Stored in a **hidden field** (`__VIEWSTATE`) in the page.
- Not secure by default—can be **encrypted** for protection.
- May increase **page size** due to hidden field data.

7. Features of ASP.NET Web Forms

Server Controls:

- Abstract HTML to server-side components.
- e.g., `<asp:Button>`, `<asp:GridView>`, `<asp:DropDownList>`

Event-Driven Programming:

- Uses server-side events like `Click`, `Load`, `SelectedIndexChanged`.

Validation Controls:

- Built-in form validation
    - `RequiredFieldValidator`
    - `RangeValidator`
    - `CompareValidator`
    - `CustomValidator`
    - `ValidationSummary`

Data Binding:

- Controls like `GridView`, `Repeater`, `ListView` support binding to data sources.
- Can bind to:
  - SQL Server
  - XML
  - Objects
  - LINQ queries

Security:

- Windows, Forms, and Passport authentication supported.
- Authorization via roles or user identity.
- Membership and Roles APIs for user management.

8. State Management Techniques

Technique	Scope	Use Case
ViewState	Per Page	Maintain control values
Session	Per User	Store user-specific info (e.g. cart)
Application	Global	Share data across users (e.g. settings)
Cookies	Client	Small client-side storage
Query String	URL	Passing simple values via URL

9. Advantages of Web Forms

- Rapid Development using Visual Studio Designer.
- Event-driven, familiar to Windows developers.
- Built-in support for data access, validation, state management.
- Highly integrated with Microsoft ecosystem (SQL Server, IIS).

10. Limitations of Web Forms

- Tight coupling between UI and logic (harder to test).
- ViewState can bloat page size.
- Limited control over HTML output.
- Not ideal for modern SPAs (Single Page Applications).
- Page life cycle can be complex for new developers.

11. Web Form Application Execution Flow

1. User requests a `.aspx` page.
2. Server processes page and events.
3. ViewState and controls reconstructed.
4. Server executes event handlers (e.g. button click).
5. Page is rendered as HTML and sent to the browser.
6. On next interaction, page posts back with hidden data.

12. Summary

- ASP.NET Web Forms = page-centric, event-driven web model.
- Uses server controls, ViewState, and postbacks.
- Simple to develop using drag-and-drop and Visual Studio.
- Ideal for internal, rapid-development web applications.
- Not suitable for high-performance, large-scale public-facing SPAs.

Q & A