UEAMPUS
STUDY GUIDE

Level 5 C#.NET Programming

uecampus.com/

First published in 2026 by UE Campus

Unit specifications can be found on the UE Campus Portal: https://uecampus.com/

Contents

Using your Study Guide

Welcome to the study guide, designed to support you in completing your Level 5 Diploma in Information Technology.

This study guide follows the order of the syllabus, which is the basis for your studies. Each chapter starts by listing the syllabus learning outcomes covered and the assessment criteria.

Level 5 Units

Unit Reference	Mandatory Units	Level	TQT	Credit	GLH
F/617/6740	Technopreneurship	5	200	20	100
J/617/6741	Network Security	5	200	20	100
L/617/6742	C#.NET Programming	5	200	20	100
R/617/6743	System Administration	5	200	20	100
Unit Reference	**Optional Units**	**Level**	**TQT**	**Credit**	**GLH**
Y/617/6744	Network Routing and Switching	5	200	20	100
D/617/6745	Network Design and Administration	5	200	20	100
H/617/6746	Content Management Systems	5	200	20	100
M/617/6748	Web Design 2	5	200	20	100
T/617/6749	Business to Business (B2B) E-commerce	5	200	20	100
K/617/6750	Business to Consumer (B2C) E-commerce	5	200	20	100

The study guide includes a number of features to enhance your studies:

	'Over to you:' activities for you to apply what you have learned.
	'Industry Insights:' discover up-to-date trends, expert opinions, and real-world examples from the .NET development ecosystem.
	'Did you know?' highlights interesting facts or surprising information to deepen your understanding.
	'Case studies:' realistic scenarios to reinforce and test your understanding.
	'Revision on the go:' use your phone camera to capture key pieces of learning and save them as revision notes.
	'Need to know:' key pieces of information highlighted in the text.
	'Examples:' illustrating points made in the text to show how it works in practice.

Note: Website addresses current as of March 2026.

Level 5 C#.NET Programming

About this unit

This unit aims to provide you with the basic concepts and principles of ASP.NET programming using C#. ASP.NET is Microsoft's powerful, enterprise-grade framework for building dynamic web applications, web services, and APIs. Combined with the C# programming language, it provides a robust, type-safe, and highly productive development environment used by millions of developers worldwide.

You will learn how the .NET framework operates, understand the components and structure of ASP.NET, compare it with other web development models, and appreciate the role of server-side validation. You will then apply these concepts practically by designing web applications using styles, themes, and master pages for consistent layouts, connecting to relational databases using ADO.NET for dynamic data display, and implementing various navigation techniques including client-side redirects, cross-page posting, and server-side transfers.

By the end of this unit, you will be able to create complete, database-driven web applications using ASP.NET and C#, and understand how to deploy them on the internet.

Chapter One – Use of ASP.NET

Introduction

This chapter explores the theoretical foundations of ASP.NET. You will study the evolution of web development from static HTML pages to dynamic server-side frameworks, understand the architecture and components of ASP.NET within the .NET framework, evaluate how ASP.NET compares with competing web development models, and analyse the critical role of server-side validation in building secure and reliable web applications.

Understanding these concepts is essential before you begin building applications in Chapter Two, as they inform the architectural decisions and best practices that distinguish professional web development from amateur coding.

Learning Outcomes

On completing the chapter, you will be able to:

1. **Understand the use of ASP.NET.**

Assessment Criteria

1.1 Analyse the components / structure of ASP.NET.

1.2 Evaluate the advantages and disadvantages of using ASP.NET compared with other web development models.

1.3 Analyse the advantages of using validators.

1.1 The components and structure of ASP.NET

Over to you – Video Watch: Introduction to ASP.NET

Watch this YouTube video:

Title: ASP.NET Core Crash Course – Traversy Media

Duration: 1:05:43

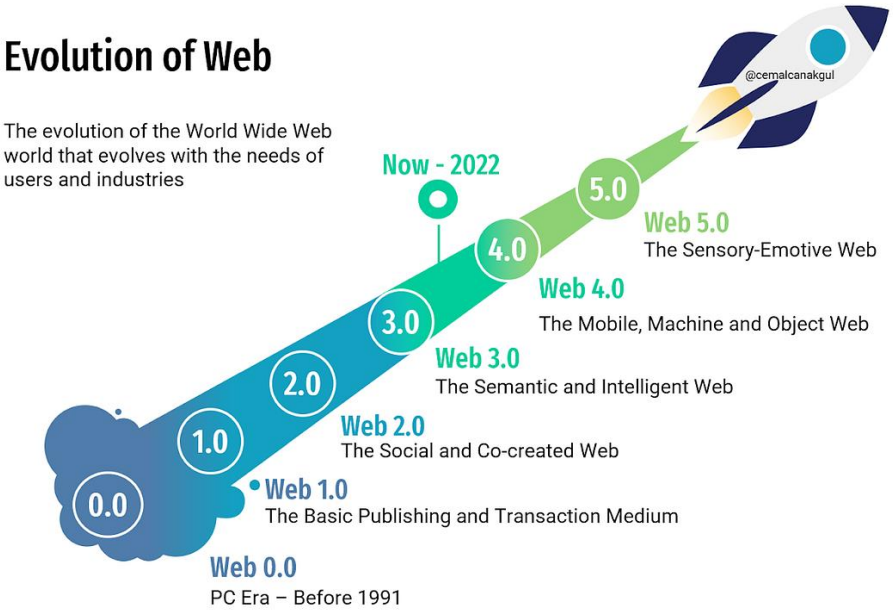
Link: https://www.youtube.com/watch?v=BfEjDD8mWYg

Watch the first 20 minutes covering the .NET ecosystem and project structure. Note the key differences between ASP.NET Web Forms and ASP.NET Core.

The Evolution of Web Development

Web development has evolved through several generations. The earliest websites (1990s) were static HTML pages – simple documents with no interactivity or dynamic content. As the web matured, technologies emerged to generate HTML dynamically on the server: Common Gateway Interface (CGI) scripts, PHP, Java Servlets, and classic ASP (Active Server Pages) from Microsoft. These first-generation server-side technologies were powerful but often produced code that mixed presentation (HTML) with logic (programming), making applications difficult to maintain.

ASP.NET, released by Microsoft in 2002 as part of the .NET Framework, represented a major leap forward. It introduced a component-based programming model, separation of code and markup, rich server-side controls, and tight integration with Microsoft's development tools. ASP.NET has since evolved through several generations: ASP.NET Web Forms (the original model, event-driven like Windows desktop apps), ASP.NET MVC (Model-View-Controller pattern for cleaner architecture), and ASP.NET Core (the modern, cross-platform, open-source successor).



The .NET Framework and .NET Platform

The .NET platform provides the runtime environment and libraries that ASP.NET applications run on:

- Common Language Runtime (CLR) – the execution engine that manages memory (garbage collection), enforces type safety, handles exceptions, and provides security. Code is compiled to an Intermediate Language (IL) that runs on the CLR, enabling language interoperability (C#, VB.NET, F# can all target the same runtime).
- Base Class Library (BCL) – a vast library of pre-built classes for common tasks: file I/O, string manipulation, data structures, networking, XML processing, and much more.
- .NET SDK – the Software Development Kit includes the compiler, build tools, project templates, and command-line tools (dotnet CLI) needed to develop .NET applications.
- NuGet – the package manager for .NET, providing access to thousands of third-party libraries (similar to npm for JavaScript or pip for Python).

Did you know?

The .NET platform has undergone a major transformation. The original .NET Framework (2002–2019) ran only on Windows. In 2016, Microsoft released .NET Core – a complete rewrite that is cross-platform (Windows, macOS, Linux), open-source, and significantly faster. From .NET 5 onwards, Microsoft unified the platform under a single name: '.NET' (without 'Framework' or 'Core'). The current version, .NET 8 (LTS), is the recommended platform for all new development. This transformation reflects Microsoft's shift to open-source and cross-platform development under CEO Satya Nadella.

The C# Language

C# (pronounced 'C sharp') is the primary language for .NET development. Created by Anders Hejlsberg at Microsoft and released in 2000, C# is a modern, object-oriented, type-safe language that combines the power of C++ with the productivity of Visual Basic. Key features include:

- Strong typing – variables must be declared with a specific type, catching errors at compile time rather than runtime.
- Object-oriented – supports classes, inheritance, polymorphism, encapsulation, and interfaces.
- Memory management – automatic garbage collection eliminates manual memory management.
- LINQ (Language Integrated Query) – a powerful query syntax built into the language for querying collections, databases, and XML.
- Async/await – built-in support for asynchronous programming, essential for responsive web applications.
- Pattern matching, records, and modern features – C# 12 (current) includes features that make code more concise and expressive.

ASP.NET Web Forms Structure

ASP.NET Web Forms uses a component-based, event-driven programming model that abstracts the stateless nature of HTTP:

- .aspx pages – the presentation layer, containing HTML markup mixed with ASP.NET server controls. Each page has a corresponding code-behind file (.aspx.cs) containing the C# logic.
- Server controls – reusable UI components that run on the server and generate HTML. Categories include: HTML server controls (standard HTML with runat="server"), Web server controls (asp:TextBox, asp:Button, asp:GridView – richer functionality), validation controls, and data controls.
- Page lifecycle – each page request follows a defined lifecycle: Init → Load → Event Handling → Rendering → Unload. Understanding this lifecycle is essential for writing correct Web Forms code.
- ViewState – a mechanism that maintains the state of server controls between page requests (postbacks) by storing encoded data in a hidden form field. This gives Web Forms a 'stateful' feel despite HTTP being stateless.
- Code-behind model – separates the presentation (.aspx) from the logic (.aspx.cs), improving maintainability. Event handlers (e.g. Button1_Click) in the code-behind respond to user interactions.
- Web.config – the XML configuration file for the application, controlling settings such as authentication mode, connection strings, custom error pages, and compilation options.

Visual Studio

Visual Studio is Microsoft's flagship IDE for .NET development. Key features for ASP.NET development include:

- IntelliSense – intelligent code completion that suggests classes, methods, and properties as you type.
- Designer view – a visual drag-and-drop editor for designing Web Forms pages.
- Debugger – set breakpoints, step through code, inspect variables, and evaluate expressions at runtime.
- Server Explorer – connect to databases, browse tables, and manage data connections directly from the IDE.
- NuGet Package Manager – install, update, and manage third-party libraries.
- Built-in web server (IIS Express) – test and debug web applications locally without deploying to a production server.
- Visual Studio Community Edition – a free, full-featured version available for students and individual developers.

Over to you – Setting Up Your Development Environment

Download and install Visual Studio Community Edition from https://visualstudio.microsoft.com/. During installation, select the 'ASP.NET and web development' workload. Create a new ASP.NET Web Forms project (or ASP.NET Core Web App if using .NET 8). Run the default template and explore the project structure. Take screenshots of: (1) the Solution Explorer showing the project files, (2) the .aspx page in both Source and Design views, and (3) the running application in the browser.

1.2 ASP.NET compared with other web development models

To evaluate ASP.NET effectively, you must understand how it compares with the other major web development frameworks and approaches available to developers.

ASP.NET vs PHP

PHP is the most widely used server-side language (powering approximately 77% of websites with a known server-side language, including WordPress). Compared with ASP.NET:

- Advantages of ASP.NET: strongly typed (C# catches more errors at compile time), better IDE support (Visual Studio), superior debugging and profiling tools, built-in security features (anti-forgery tokens, output encoding), high performance (especially ASP.NET Core), better suited to large enterprise applications, integrated with the entire Microsoft ecosystem (Azure, SQL Server, Active Directory).
- Disadvantages of ASP.NET: historically Windows-only (though ASP.NET Core is cross-platform), smaller hosting ecosystem (though Azure and Linux hosting are now widely available), steeper learning curve for beginners, licensing costs for Visual Studio Enterprise (though Community is free), smaller open-source community compared to PHP.
- Advantages of PHP: easy to learn, massive community, widely available cheap hosting, extensive CMS ecosystem (WordPress, Drupal, Joomla), cross-platform from the start.

ASP.NET vs PHP: An End-to-End Comparison

Parameter	ASP.NET	PHP
Foundation	The ASP.NET syntax is based on Visual Basic programming languages.	PHP syntax is based on the C programing language.
Support	Supported by Microsoft corporation.	It's supported by the PHP foundation, Zend technologies, and online communities.
Uses	ASP.NET is used as a web application framework for building enterprise-grade applications.	It's used for server-side coding and programming web solutions.
Execution	It follows compiled code method.	PHP follows interpreted code method.
Focus	ASP.NET focuses more on enhanced security and functionality.	It focuses on hosting a better interface and enhancing client-side performance.
Debugging	Debugging is complicated in ASP.NET.	PHP eases the bug-fixing process within the code.
Performance	ASP.NET provides slower performance for desktop apps.	PHP provides improved performance for desktop, server-side, and web-based applications.

ASP.NET vs Node.js (Express)

Node.js uses JavaScript on the server, allowing developers to use a single language for both client and server:

- Advantages of ASP.NET: strong typing (TypeScript can be used with Node.js but is optional), more mature enterprise features, better performance for CPU-intensive

tasks, richer built-in framework features (authentication, authorisation, caching, session management).

- Disadvantages of ASP.NET: cannot share code between client and server (unlike full-stack JavaScript), larger framework footprint, less suited to microservices compared to lightweight Node.js.
- Advantages of Node.js: JavaScript everywhere (full-stack), lightweight and fast for I/O-intensive applications, massive npm ecosystem, excellent for real-time applications (WebSockets, chat, streaming).

ASP.NET vs Python (Django/Flask)

- Advantages of ASP.NET: higher runtime performance, better tooling (Visual Studio), more suitable for large-scale enterprise applications, stronger type system.
- Disadvantages of ASP.NET: more verbose syntax, less popular for data science and AI integration, smaller presence in the startup ecosystem.
- Advantages of Python: simpler syntax (faster development for simple applications), dominates in data science and machine learning, Django provides 'batteries included' rapid development.

ASP.NET Web Forms vs ASP.NET MVC vs ASP.NET Core

Within the ASP.NET ecosystem itself, there are different models:

- Web Forms – event-driven, component-based model. Strengths: rapid development with drag-and-drop, familiar to Windows Forms developers. Weaknesses: heavy ViewState, less control over rendered HTML, difficult to unit test, considered legacy technology.
- MVC (Model-View-Controller) – separates the application into three concerns. Strengths: clean architecture, full control over HTML, testable, SEO-friendly URLs. Weaknesses: steeper learning curve, more boilerplate code.
- ASP.NET Core – the modern, unified platform. Strengths: cross-platform, high performance, minimal APIs, built-in dependency injection, modern middleware pipeline. This is the recommended platform for all new projects.

Industry Insight – .NET in the Enterprise

Despite the rise of JavaScript frameworks and Python, .NET remains one of the most important enterprise development platforms. According to the Stack Overflow 2024 Developer Survey, C# is used by approximately 30% of professional developers, and ASP.NET Core is the sixth most popular web framework globally. Major companies using .NET include Microsoft (Azure, Office 365), Stack Overflow (built entirely on ASP.NET), Dell, GE Healthcare, Siemens, and numerous financial institutions. The .NET ecosystem is particularly strong in enterprise applications, financial services, healthcare, and government systems where reliability, security, and long-term support are paramount.

Explore: https://dotnet.microsoft.com/en-us/platform/customers

Over to you – Framework Comparison

Create a comparison table evaluating ASP.NET Core, Node.js (Express), PHP (Laravel), and Python (Django) across the following criteria: performance, ease of learning, community size, enterprise suitability, hosting options, and job market demand. For each criterion, rate each framework as Strong, Moderate, or Limited, with a brief justification. Then write a 300-word recommendation for: (a) a startup building a SaaS product, and (b) a bank building an internal enterprise application. Which framework would you recommend for each and why?

1.3 The advantages of using validators

Validation is the process of ensuring that user input meets expected criteria before it is processed by the application. In web development, validation is critical for both data quality and security. ASP.NET provides a rich set of built-in validation controls that simplify this essential task.

Why Validation Matters

- Data integrity – ensures that only valid data enters the system. A database expecting a date should not receive random text; an email field should contain a properly formatted email address.
- Security – prevents injection attacks (SQL injection, XSS) by rejecting or sanitising malicious input before it reaches the database or is displayed on the page.
- User experience – provides immediate, helpful feedback when users make errors, reducing frustration and form abandonment.
- Business logic enforcement – ensures that business rules are respected (e.g. order quantity must be positive, date of birth must be in the past, password must meet complexity requirements).

Client-Side vs Server-Side Validation

Client-side validation (using JavaScript in the browser) provides instant feedback without a server round-trip, improving the user experience. However, client-side validation can be bypassed by disabling JavaScript or manipulating HTTP requests directly. Therefore, server-side validation is always required as the final line of defence. ASP.NET validation controls automatically generate both client-side JavaScript validation and server-side validation, providing the best of both worlds.

ASP.NET Validation Controls

RequiredFieldValidator

Ensures that a field is not left empty. The most commonly used validator. Example: <asp:RequiredFieldValidator ControlToValidate="txtName" ErrorMessage="Name is required" runat="server" />

RangeValidator

Checks that a value falls within a specified range. Supports numeric, date, and string comparisons. Example: ensuring an age field contains a value between 18 and 120.

CompareValidator

Compares the value of one control with another control or a fixed value. Common use: confirming that the ‘Confirm Password’ field matches the ‘Password’ field.

RegularExpressionValidator

Validates input against a regular expression pattern. Essential for complex format validation: email addresses, phone numbers, postcodes, and custom patterns. Example: validating a UK postcode format.

CustomValidator

Allows you to write your own validation logic in C# for scenarios not covered by the built-in validators. You define a server-side validation function (and optionally a client-side JavaScript function) that returns true or false.

ValidationSummary

Displays a consolidated list of all validation error messages on the page, typically at the top of the form. This provides users with a clear overview of all errors that need to be corrected.

Advantages of ASP.NET Validators

- Dual validation – automatically generates both client-side (JavaScript) and server-side validation from a single declarative control.
- Declarative syntax – validators are added in the markup (.aspx), reducing the amount of C# code needed.
- Consistent error display – validation messages are displayed consistently using a standardised approach.
- Validation groups – allow different sections of a page to be validated independently (e.g. a login form and a search form on the same page).
- Page.IsValid property – in the code-behind, checking if (Page.IsValid) ensures that all validators have passed before processing the form submission.
- Extensibility – the CustomValidator allows any validation logic you can imagine.

Example – Registration Form with Validators

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

<asp:RequiredFieldValidator ControlToValidate="txtEmail" ErrorMessage="Email is required" runat="server" />

<asp:RegularExpressionValidator ControlToValidate="txtEmail" ValidationExpression="\w+@\w+\.\w+" ErrorMessage="Invalid email format" runat="server" />


<asp:TextBox ID="txtPassword" TextMode="Password" runat="server" />

<asp:TextBox ID="txtConfirm" TextMode="Password" runat="server" />
```

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

<asp:ValidationSummary HeaderText="Please correct the following errors:" runat="server" />
```

Over to you – Video Watch: ASP.NET Validation Controls

Watch this YouTube video:

Title: ASP.NET Validation Controls Tutorial – kudvenkat

Duration: 25:15

Link: https://www.youtube.com/watch?v=QRYMz4hMbMc

After watching, create a registration form in Visual Studio with fields for name, email, password, confirm password, age, and a submit button. Apply at least four different validator types. Test both client-side and server-side validation.

Over to you – Validation Task

Design a web form for a job application that includes: name (required), email (required, valid format), phone (required, UK format), date of birth (required, must be at least 18 years ago), desired salary (required, numeric, range £20,000–£150,000), and a cover letter (required, minimum 50 characters). Specify which ASP.NET validator you would use for each field and write the markup for each. Explain how the ValidationSummary would present errors to the user.

Reading List

- Freeman, A. (2024). *Pro ASP.NET Core 8*. 9th edn. New York: Apress.
- Lock, A. (2024). *ASP.NET Core in Action*. 3rd edn. Shelter Island, NY: Manning.
- Murach, J. (2023). *Murach's ASP.NET Core Programming with C#*. 2nd edn. Fresno, CA: Mike Murach & Associates.
- Price, M.J. (2024). *C# 12 and .NET 8 – Modern Cross-Platform Development Fundamentals*. 8th edn. Birmingham: Packt Publishing.
- Troelsen, A. & Japikse, P. (2024). *Pro C# 12 with .NET 8*. 11th edn. New York: Apress.
- Walther, S. (2023). *ASP.NET Unleashed*. 3rd edn. Indianapolis, IN: Sams Publishing.

Summary

In this chapter, you have developed a thorough understanding of ASP.NET. You have traced the evolution of web development from static HTML to modern server-side frameworks and studied the components and structure of the .NET platform, including the CLR, BCL, C# language, and Visual Studio IDE. You have analysed the ASP.NET Web Forms architecture including pages, server controls, the page lifecycle, ViewState, and the code-behind model. You have evaluated ASP.NET against competing frameworks (PHP, Node.js, Django) and the different models within ASP.NET itself (Web Forms, MVC, Core). Finally, you have analysed the advantages of ASP.NET's validation controls, understanding how they provide dual client-side and server-side validation to ensure data integrity, security, and a good user experience.

Chapter Two – Designing Web Applications Using ASP.NET and ADO.NET

Introduction

This chapter moves from theory to practice. You will build web applications using ASP.NET's powerful features for creating attractive, consistent layouts (styles, themes, and master pages), connecting to relational databases to display dynamic data (ADO.NET and data binding), and implementing navigation between pages using multiple techniques. By the end of this chapter, you will be able to create a complete, data-driven web application.

Learning Outcomes

On completing the chapter, you will be able to:

1. **Design web applications using ASP.NET and ADO.NET.**

Assessment Criteria

2.1 Use styles, themes and master pages to create an attractive and easily navigable web application.

2.2 Display dynamic data from a relational database by using ADO.NET and data binding through different languages including C#.

2.3 Create a web page that uses client side navigation, client side browser redirect, cross page posting and server side transfer that meets the brief.

2.1 Styles, themes and master pages

Creating a professional web application requires consistent visual design across all pages. ASP.NET provides three complementary mechanisms for achieving this: CSS styles, ASP.NET themes, and master pages.

Over to you – Video Watch: Master Pages in ASP.NET

Watch this YouTube video:

Title: ASP.NET Master Pages Tutorial – kudvenkat

Duration: 18:42

Link: https://www.youtube.com/watch?v=UStmkWMGlsE

After watching, create a master page with a header (containing a logo and navigation menu), a content placeholder area, and a footer. Create three content pages that use this master page.

CSS Styles in ASP.NET

Cascading Style Sheets (CSS) control the visual presentation of web pages – colours, fonts, spacing, layout, and responsive design. In ASP.NET, CSS works the same way as in any web application:

- External stylesheets – CSS files (e.g. styles.css) linked in the page head. Best practice for site-wide styling.
- Inline styles – applied directly to individual elements using the style attribute. Should be used sparingly.
- CSS classes – ASP.NET server controls support the CssClass property: <asp:Button CssClass="btn-primary" runat="server" />.
- Responsive design – using CSS media queries and frameworks like Bootstrap (which integrates seamlessly with ASP.NET project templates) to create layouts that adapt to different screen sizes.

ASP.NET Themes and Skins

ASP.NET themes extend CSS by allowing you to define default visual properties for server controls across the entire application:

- Theme folder – themes are stored in the App_Themes folder. Each theme is a subfolder containing CSS files and .skin files.
- Skin files (.skin) – define default property values for server controls. For example, a skin file can set the default font, colour, and size for all TextBox controls throughout the application.
- Applying themes – themes can be applied at the page level (<%@ Page Theme="BlueSky" %>) or at the application level in Web.config (<pages theme="BlueSky" />).

- StyleSheetTheme vs Theme – StyleSheetTheme applies before control-level properties (can be overridden); Theme applies after (cannot be overridden). StyleSheetTheme is generally preferred for flexibility.

Master Pages

Master pages are one of ASP.NET's most powerful features for creating consistent layouts. A master page (.master) defines the common layout elements (header, navigation, footer, sidebar) that appear on every page, with ContentPlaceHolder controls marking the areas where individual pages can insert their unique content.

Key concepts:

- Master page (.master) – defines the overall HTML structure, including <html>, <head>, <body>, and shared elements. Contains one or more <asp:ContentPlaceHolder> controls.
- Content pages (.aspx) – reference the master page using the MasterPageFile attribute and provide content using <asp:Content> controls that correspond to the master page's placeholders.
- Nested master pages – a master page can itself use another master page, enabling hierarchical layouts (e.g. a site-level master with a section-level master).
- Programmatic access – content pages can access controls in the master page using Master.FindControl() for dynamic interactions.

Example – Master Page and Content Page

SiteMaster.master:

```
<%@ Master Language="C#" %>
<html><head><link href="styles.css" rel="stylesheet" /></head>
<body>
 <header><h1>My Web App</h1><nav>...</nav></header>
 <main><asp:ContentPlaceHolder ID="MainContent" runat="server" /></main>
 <footer>© 2026 My Company</footer>
</body></html>
```

Home.aspx:

```
<%@ Page MasterPageFile="~/SiteMaster.master" %>
<asp:Content ContentPlaceHolderID="MainContent" runat="server">
 <h2>Welcome to our website!</h2>
```

```
 <p>This content appears inside the master page layout.</p>
</asp:Content>
```

Website Navigation Using ASP.NET

ASP.NET provides built-in navigation controls that integrate with a site map:

- Web.sitemap – an XML file defining the hierarchical structure of the website (pages, titles, URLs).
- SiteMapDataSource – a data source control that reads the site map and provides it to navigation controls.
- Menu control – renders a hierarchical navigation menu (horizontal or vertical).
- TreeView control – renders a tree-style expandable/collapsible navigation structure.
- SiteMapPath (breadcrumb) – displays the user's current location in the site hierarchy (e.g. Home > Products > Laptops).

Over to you – Layout Design Project

Create a complete ASP.NET Web Forms website with: (1) a master page including a header with a navigation menu, a main content area, and a footer, (2) an external CSS stylesheet providing responsive styling, (3) at least four content pages (Home, About, Products, Contact), (4) a Web.sitemap file defining the site structure, (5) a Menu or TreeView control for navigation, and (6) a SiteMapPath breadcrumb. Test on different browser window sizes. Submit your project with screenshots.

2.2 Dynamic data with ADO.NET and data binding

ADO.NET (ActiveX Data Objects for .NET) is the data access technology in the .NET framework that allows your web application to connect to relational databases, execute queries, and display dynamic data. It provides a bridge between your C# code and the database.

Over to you – Video Watch: ADO.NET and Data Binding

Watch this YouTube video:

Title: ADO.NET Tutorial – C# Database Access – IAmTimCorey

Duration: 45:32

Link: https://www.youtube.com/watch?v=bIiEv__QNxw

Watch the first 25 minutes covering SqlConnection, SqlCommand, and SqlDataReader. Set up a SQL Server database (using LocalDB or SQL Server Express) with a sample Products table and write C# code to retrieve and display the data.

ADO.NET Architecture

ADO.NET provides two models for data access:

Connected Model (Data Reader)

Maintains an active connection to the database while reading data. Uses SqlConnection, SqlCommand, and SqlDataReader. The DataReader provides fast, forward-only, read-only access to query results. It is the most efficient approach when you need to read data once and display it immediately.

```
string connStr = ConfigurationManager.ConnectionStrings["MyDB"].ConnectionString;
using (SqlConnection conn = new SqlConnection(connStr))
{
  conn.Open();
  SqlCommand cmd = new SqlCommand("SELECT * FROM Products", conn);
  SqlDataReader reader = cmd.ExecuteReader();
  while (reader.Read())
  {
    string name = reader["ProductName"].ToString();
    decimal price = (decimal)reader["Price"];
  }
}
```

Disconnected Model (DataSet/DataAdapter)

Retrieves data into an in-memory DataSet and closes the connection. Uses SqlDataAdapter to fill the DataSet. The data can then be manipulated offline and changes sent back to the database. The disconnected model is suitable for scenarios where you need to work with data across multiple operations or cache results.

```
SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM Products", connStr);
DataSet ds = new DataSet();
adapter.Fill(ds, "Products");
// Data is now in memory; connection is closed
GridView1.DataSource = ds.Tables["Products"];
GridView1.DataBind();
```

Parameterised Queries (Security)

Never concatenate user input directly into SQL strings – this creates SQL injection vulnerabilities. Always use parameterised queries:

```
SqlCommand cmd = new SqlCommand("SELECT * FROM Products WHERE Category = @cat", conn);
cmd.Parameters.AddWithValue("@cat", categoryDropDown.SelectedValue);
```

Data Binding

Data binding connects data from a source (database, collection, XML) to ASP.NET controls for display. Key data-bound controls include:

- GridView – displays data in a tabular format with built-in support for sorting, paging, editing, and deleting rows. The most commonly used data control for displaying database records.
- DetailsView – displays a single record at a time in a vertical key-value layout. Supports insert, update, and delete operations.
- FormView – similar to DetailsView but uses templates for complete control over the layout of individual records.
- Repeater – the most flexible data-bound control. Provides no built-in formatting – you define the HTML template for each item. Ideal for custom layouts.
- ListView – combines the flexibility of Repeater with built-in support for paging, sorting, editing, and deleting.
- SqlDataSource – a declarative data source control that connects directly to a database from the markup, reducing the amount of C# code needed for simple data retrieval.

Example – GridView with SqlDataSource

```
<asp:SqlDataSource ID="dsProducts" runat="server"
 ConnectionString="<%$ ConnectionStrings:MyDB %>"
 SelectCommand="SELECT ProductID, ProductName, Price, Category FROM Products" />

<asp:GridView ID="gvProducts" runat="server"
 DataSourceID="dsProducts"
 AutoGenerateColumns="False"
 AllowSorting="True" AllowPaging="True" PageSize="10">
 <Columns>
  <asp:BoundField DataField="ProductName" HeaderText="Product"
SortExpression="ProductName" />
  <asp:BoundField DataField="Price" HeaderText="Price" DataFormatString="{0:C}" />
  <asp:BoundField DataField="Category" HeaderText="Category" />
 </Columns>
</asp:GridView>
```

Case Study – Employee Directory Application

A company needs a web-based employee directory. The SQL Server database contains an Employees table with columns: EmployeeID (int, PK), FirstName (nvarchar), LastName (nvarchar), Department (nvarchar), Email (nvarchar), HireDate (date), and Salary (decimal).

Task: Build the application in ASP.NET Web Forms with: (1) a master page with company branding and navigation, (2) a page displaying all employees in a GridView with sorting and paging, (3) a search feature allowing filtering by department, (4) a details page showing a single employee using DetailsView, (5) an add/edit form with appropriate validation controls, and (6) all database access using parameterised ADO.NET queries. Submit your complete project with screenshots.

2.3 Navigation, redirects and page transfers

Web applications require multiple ways to navigate between pages. ASP.NET supports several navigation techniques, each with different characteristics and appropriate use cases.

Client-Side Navigation (Hyperlinks)

The simplest form of navigation: standard HTML hyperlinks (<a href="page.aspx">) or ASP.NET HyperLink controls (<asp:HyperLink NavigateUrl="~/page.aspx" runat="server" />). The browser requests the target page directly. No server-side processing occurs before navigation. Suitable for: standard page-to-page navigation, external links, navigation menus.

Client-Side Browser Redirect (Response.Redirect)

Response.Redirect() sends an HTTP 302 redirect response to the browser, which then requests the new page. The browser's address bar updates to show the new URL:

```
Response.Redirect("Products.aspx?id=42");
```

Characteristics: the browser makes a new request (two round-trips); the URL changes in the browser; data can be passed via query string parameters; works for both internal and external URLs; the user can see and bookmark the new URL. Suitable for: navigating after a form submission (Post-Redirect-Get pattern), linking to external sites, passing simple data via query strings.

Cross-Page Posting

By default, ASP.NET Web Forms pages post back to themselves. Cross-page posting allows a button to submit form data to a different page:

```
<asp:Button Text="Submit" PostBackUrl="~/ProcessOrder.aspx" runat="server" />
```

On the target page, the previous page's controls can be accessed:

```
TextBox txtName = (TextBox)PreviousPage.FindControl("txtName");
string name = txtName.Text;
```

Characteristics: form data is sent directly to the target page; the target page can access controls and data from the source page via PreviousPage; the URL changes in the browser. Suitable for: multi-step forms, wizard-style workflows, passing complex form data to a processing page.

Server-Side Transfer (Server.Transfer)

Server.Transfer() transfers execution from the current page to another page on the server without the browser making a new request:

Server.Transfer("ConfirmationPage.aspx");

Characteristics: the URL in the browser does NOT change (the user still sees the original URL); only one round-trip; the target page can access the source page's data via PreviousPage; only works for pages within the same application (cannot transfer to external URLs); slightly faster than Response.Redirect because it avoids a round-trip. Suitable for: internal page transfers where you don't want the URL to change, processing pipelines where multiple pages handle a request.

Comparison Table

Feature	Response.Redirect	Cross-Page Post	Server.Transfer
URL changes	Yes	Yes	No
Round-trips	Two	One	One
External URLs	Yes	No	No
Pass data	Query string	PreviousPage	PreviousPage
Browser back	Works normally	Works normally	May cause issues
Bookmarkable	Yes	Yes	No (shows old URL)

Did you know?

The Post-Redirect-Get (PRG) pattern is a web development best practice where after processing a form submission (POST), the server responds with a redirect (302) to a results page (GET). This prevents the common problem of form resubmission when the user refreshes the page. In ASP.NET, this is implemented by calling Response.Redirect() after successfully processing form data. Most professional web applications use this pattern for any page that modifies data.

Over to you – Video Watch: ASP.NET Navigation Techniques

Watch this YouTube video:

Title: ASP.NET Navigation – Response.Redirect vs Server.Transfer vs Cross Page Posting – kudvenkat

Duration: 15:08

Link: https://www.youtube.com/watch?v=3m-qYFfszHs

After watching, create a two-page application demonstrating all three navigation techniques. Page 1 has a text field and three buttons (one for each technique). Page 2 receives and displays the entered text. Document the differences you observe.

Over to you – Complete Web Application

Build a complete ASP.NET Web Forms application for a simple online product catalogue with the following features: (1) master page with navigation, (2) home page with a welcome message, (3) product listing page with a GridView displaying products from a SQL Server database (with sorting and paging), (4) a product details page (accessed via Response.Redirect with a query string ID parameter), (5) an 'Add Product' form page with at least four validation controls that posts to a confirmation page using cross-page posting, and (6) the confirmation page uses Server.Transfer to redirect to the product listing after insertion. Apply a consistent theme throughout. Submit your project folder, database script, and screenshots demonstrating each feature.

Reading List

- Freeman, A. (2024). *Pro ASP.NET Core 8*. 9th edn. New York: Apress.
- Lock, A. (2024). *ASP.NET Core in Action*. 3rd edn. Shelter Island, NY: Manning.
- MacDonald, M. (2023). *Beginning ASP.NET in C#*. Updated edn. New York: Apress.
- Murach, J. (2023). *Murach's ASP.NET Core Programming with C#*. 2nd edn. Fresno, CA: Mike Murach & Associates.
- Price, M.J. (2024). *C# 12 and .NET 8*. 8th edn. Birmingham: Packt Publishing.
- Troelsen, A. & Japikse, P. (2024). *Pro C# 12 with .NET 8*. 11th edn. New York: Apress.

Summary

In this chapter, you have developed practical skills for building web applications using ASP.NET and ADO.NET. You have learned to create consistent, attractive layouts using CSS styles, ASP.NET themes and skins, and master pages with ContentPlaceHolder controls. You have studied ASP.NET's navigation controls (Menu, TreeView, SiteMapPath) and the Web.sitemap file for structured site navigation. You have connected to relational databases using ADO.NET's connected (DataReader) and disconnected (DataSet) models, always using parameterised queries for security. You have used data-bound controls (GridView, DetailsView, FormView, Repeater) to display and manipulate dynamic data. Finally, you have compared and implemented multiple navigation techniques – client-side links, Response.Redirect, cross-page posting, and Server.Transfer – understanding the appropriate use case for each.

Glossary

Word / Term	Explanation
ADO.NET	ActiveX Data Objects for .NET; the data access technology for connecting to databases.
ASP.NET	Microsoft's framework for building dynamic web applications on the .NET platform.
C#	A modern, object-oriented, type-safe programming language for .NET development.
CLR	Common Language Runtime; the execution engine that manages .NET application execution.
Code-Behind	A file containing C# logic separated from the presentation markup (.aspx).
ContentPlaceHolder	A control in a master page that marks where content pages insert their content.
Cross-Page Posting	A technique where a form submits data to a different page for processing.
CSS	Cascading Style Sheets; controls the visual presentation of web pages.
Data Binding	Connecting a data source to a UI control for automatic display of data.
DataReader	An ADO.NET object providing fast, forward-only, read-only access to query results.
DataSet	An in-memory cache of data retrieved from a database (disconnected model).
GridView	An ASP.NET control that displays data in a table with sorting, paging, and editing.
IntelliSense	Visual Studio's intelligent code completion feature.
LINQ	Language Integrated Query; a C# feature for querying data collections and databases.
Master Page	A page defining common layout elements shared across multiple content pages.
NuGet	The package manager for .NET, providing access to third-party libraries.
.NET	Microsoft's cross-platform development platform including runtime and libraries.
Postback	When a Web Forms page submits to itself for server-side processing.
Response.Redirect	Sends an HTTP 302 redirect to the browser, causing it to request a new page.
Server.Transfer	Transfers execution to another page on the server without a browser round-trip.
Server Control	A UI component that runs on the server and generates HTML output.

Skin File	Defines default visual properties for ASP.NET server controls within a theme.
SqlConnection	An ADO.NET class representing a connection to a SQL Server database.
SqlDataSource	A declarative data source control for database access directly from markup.
Theme	A collection of styles and skins applied to an ASP.NET application for visual consistency.
Validator	An ASP.NET control that validates user input on both client and server sides.
ViewState	A mechanism that maintains server control state between page postbacks.
Visual Studio	Microsoft's IDE for .NET development with code editing, debugging, and design tools.
Web.config	The XML configuration file for an ASP.NET application.
Web.sitemap	An XML file defining the hierarchical navigation structure of an ASP.NET site.

MCQs and True & False Questions (self-assessment)

True or False Questions

1. ASP.NET is a server-side web development framework.
2. C# is a dynamically typed language.
3. The CLR manages memory through garbage collection.
4. ViewState is stored in the database.
5. Master pages allow consistent layouts across multiple pages.
6. Client-side validation is sufficient without server-side validation.
7. ADO.NET provides both connected and disconnected data access models.
8. Response.Redirect changes the URL in the browser.
9. Server.Transfer makes a new HTTP request from the browser.
10. The GridView control supports sorting and paging.
11. SQL injection can be prevented by using parameterised queries.
12. ASP.NET Web Forms uses an event-driven programming model.
13. The .NET Framework is cross-platform.
14. NuGet is the package manager for .NET.
15. Cross-page posting allows a form to submit data to a different page.
16. A RequiredFieldValidator checks for valid email format.
17. The DataReader provides forward-only, read-only access to data.
18. Themes and skins only apply to HTML elements, not server controls.
19. Visual Studio Community Edition is free.
20. LINQ allows querying data directly in C# code.

Multiple Choice Questions

1. ASP.NET runs on which runtime?

A. JVM

B. CLR

C. Node.js

D. PHP Interpreter

2. Which file extension is used for ASP.NET Web Forms pages?

A. .html

B. .php

C. .aspx

D. .jsp

3. ViewState is stored in:

A. The database

B. A cookie

C. A hidden form field

D. Server memory

4. Which validator checks that input matches a pattern?

A. RequiredFieldValidator

B. RangeValidator

C. RegularExpressionValidator

D. CompareValidator

5. The code-behind file for Default.aspx is:

A. Default.aspx.vb

B. Default.aspx.cs

C. Default.cs

D. Default.code

6. Which ADO.NET object maintains an active database connection?

A. DataSet

B. DataAdapter

C. DataReader

D. DataTable

7. Master pages use which control to mark content areas?

A. PlaceHolder

B. Panel

C. ContentPlaceHolder

D. Container

8. Response.Redirect involves how many round-trips?

A. Zero

B. One

C. Two

D. Three

9. Which control displays data in a table with built-in paging?

A. Repeater

B. ListView

C. GridView

D. FormView

10. Parameterised queries prevent:

A. Slow performance

B. SQL injection

C. Cross-page posting

D. ViewState corruption

11. Server.Transfer differs from Response.Redirect because it:

A. Changes the URL

B. Works for external sites

C. Does not change the URL

D. Is slower

12. Which technology is NOT part of the .NET platform?

A. CLR

B. BCL

C. NuGet

D. XAMPP

13. CSS classes are applied to server controls using:

A. class attribute

B. CssClass property

C. Style property

D. Theme attribute

14. A skin file defines:

A. Database connections

B. Default visual properties for server controls

C. Navigation structure

D. Validation rules

15. SqlDataSource is a:

A. Server control

B. C# class

C. Declarative data source control

D. JavaScript library

16. The Page.IsValid property checks:

A. If the page has loaded

B. If all validators have passed

C. If the database is connected

D. If CSS is applied

17. Cross-page posting uses which button property?

A. NavigateUrl

B. CommandName

C. PostBackUrl

D. RedirectUrl

18. Which navigation control shows breadcrumbs?

A. Menu

B. TreeView

C. SiteMapPath

D. ListView

19. The disconnected model in ADO.NET uses:

A. SqlDataReader

B. SqlConnection only

C. DataSet and DataAdapter

D. SqlCommand only

20. ASP.NET Core is:

A. Windows-only

B. Cross-platform and open-source

C. Only for desktop apps

D. Deprecated

Answers to True/False Questions

1. *True.* ASP.NET processes code on the server and sends HTML to the browser.

2. *False.* C# is a statically (strongly) typed language; types are checked at compile time.

3. *True.* The CLR's garbage collector automatically reclaims unused memory.

4. *False.* ViewState is stored in a hidden form field on the page, not in the database.

5. *True.* Master pages define shared layout elements with ContentPlaceHolder controls.

6. *False.* Client-side validation can be bypassed; server-side validation is always required.

7. *True.* ADO.NET provides DataReader (connected) and DataSet/DataAdapter (disconnected).

8. *True.* Response.Redirect sends a 302 status code, causing the browser to request a new URL.

9. *False.* Server.Transfer executes on the server without a new browser request; the URL does not change.

10. *True.* GridView supports AllowSorting and AllowPaging properties for these features.

11. *True.* Parameterised queries separate SQL code from data, preventing injection attacks.

12. *True.* Web Forms uses events (Button_Click, Page_Load) similar to desktop applications.

13. *False.* The original .NET Framework was Windows-only; .NET Core/.NET 5+ is cross-platform.

14. *True.* NuGet provides access to thousands of .NET packages and libraries.

15. *True.* The PostBackUrl property on a Button directs the form submission to a different page.

16. *False.* RequiredFieldValidator checks for empty fields; RegularExpressionValidator checks format.

17. *True.* DataReader is a fast, forward-only cursor for reading query results sequentially.

18. *False.* Skins specifically define default properties for ASP.NET server controls.

19. *True.* Visual Studio Community is free for individual developers, students, and small teams.

20. *True.* LINQ enables SQL-like queries directly in C# for collections, databases, and XML.

Answers to Multiple Choice Questions

1. (B) CLR

2. (C) .aspx

3. (C) A hidden form field

4. (C) RegularExpressionValidator

5. (B) Default.aspx.cs

6. (C) DataReader

7. (C) ContentPlaceHolder

8. (C) Two

9. (C) GridView

10. (B) SQL injection

11. (C) Does not change the URL

12. (D) XAMPP

13. (B) CssClass property

14. (B) Default visual properties for server controls

15. (C) Declarative data source control

16. (B) If all validators have passed

17. (C) PostBackUrl

18. (C) SiteMapPath

19. (C) DataSet and DataAdapter

20. (B) Cross-platform and open-source