Practical 1 : To Study the ASP.NET Framework

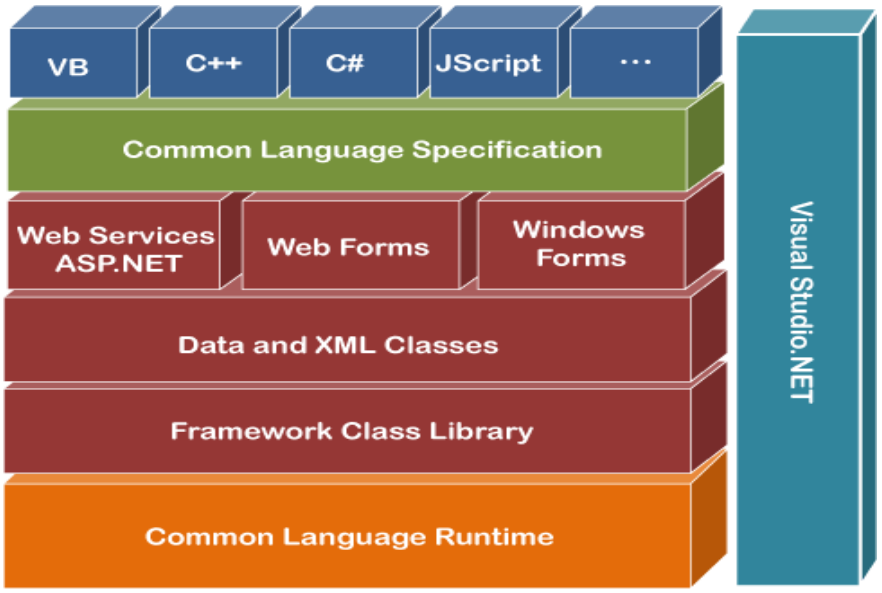
The .NET Framework is a software development platform that was introduced by Microsoft in the late 1990 under the NGWS. On 13 February 2002, Microsoft launched the first version of the .NET Framework, referred to as the .NET Framework 1.0.

What is .NET Framework

It is a virtual machine that provide a common platform to run an application that was built using the different language such as C#, VB.NET, Visual Basic, etc. It is also used to create a form based, console-based, mobile and web-based application or services that are available in Microsoft environment. Furthermore, the .NET framework is a pure object oriented, that similar to the Java language. But it is not a platform independent as the Java. So, its application runs only to the windows platform.

The main objective of this framework is to develop an application that can run on the windows platform. The current version of the .Net framework is 4.8.

Components of .NET Framework



- There are following components of .NET Framework:
- CLR (Common Language Runtime)
- CTS (Common Type System)
- BCL (Base Class Library)
- CLS (Common Language Specification)
- FCL (Framework Class Library)
- .NET Assemblies
- XML Web Services

CLR (common language runtime)

It is an important part of a .NET framework that works like a virtual component of the .NET Framework to executes the different languages program like c#, Visual Basic, etc. A CLR also helps to convert a source code into the byte code, and this byte code is known as CIL (Common Intermediate Language) or MSIL (Microsoft Intermediate Language). After converting into a byte code, a CLR uses a JIT compiler at run time that helps to convert a CIL or MSIL code into the machine or native code.

CTS (Common Type System)

It specifies a standard that represent what type of data and value can be defined and managed in computer memory at runtime. A CTS ensures that programming data defined in various languages should beinteract with each other to share information. For example, in C# we define data type as int, while in VB.NET we define integer as a data type.

BCL (Base Class Library)

The base class library has a rich collection of libraries features and functions that help to implement many programming languages in the .NET Framework, such as C #, F #, Visual C ++, and more. Furthermore, BCL divides into two parts:

- **User defined class library**

- **Assemblies -** It is the collection of small parts of deployment an application's part. It contains either the DLL (Dynamic Link Library) or exe (Executable) file. DLL and EXE files are types of binary files in Windows. An EXE file is an executable file that runs a program, while a DLL file contains code and data that can be used by multiple programs simultaneously
    - In DLL, it uses code reusability, whereas in exe it contains only output file/ or application.
    - DLL file can't be open, whereas exe file can be open.
    - DLL file can't be run individually, whereas in exe, it can run individually.
    - In DLL file, there is no main method because it is designed to be used by other applications rather than executed on its own, whereas exe file has main method.

- ## Predefined class library
    - **Namespace -** It is the collection of predefined class and method that present in .Net. In other languages such as, C we used header files, in java we used package similarly we used "using system" in .NET, where using is a keyword and system is a namespace.

CLS (Common language Specification)

It is a subset of common type system (CTS) that defines a set of rules and regulations which should be followed by every language that comes under the .net framework. In other words, a CLS language should be cross-language integration or interoperability. For example, in C# and VB.NET language, the C# language terminate each statement with semicolon, whereas in VB.NET it is not end with semicolon, and when these statements execute in .NET Framework, it provides a common platform to interact and share information with each other.

Microsoft .NET Assemblies

A .NET assembly is the main building block of the .NET Framework. It is a small unit of code that contains a logical compiled code in the Common Language infrastructure (CLI), which is used for deployment, security and versioning. It defines in two parts (process) DLL and library (exe) assemblies.

When the .NET program is compiled, it generates a metadata with Microsoft Intermediate Language, which is stored in a file called Assembly.

FCL (Framework Class Library)

It provides the various system functionality in the .NET Framework, that includes classes, interfaces and data types, etc. to create multiple functions and different types of application such as desktop, web, mobile application, etc. In other words, it can be defined as, it provides a base on which various applications, controls and components are built in .NET Framework.

2.To Study &Create Presentation Layer using HTML & CSS using visual studio 2019

Certainly! Visual Studio 2019 is a great tool for web development, and it provides a convenient environment for creating HTML and CSS files. Here's a step-by-step guide on how to create a presentation layer using HTML and CSS in Visual Studio 2019:

1.Install Visual Studio 2019: If you haven't already, download and install Visual Studio 2019 from the official Microsoft website.

1. **Create a New Project:** Open Visual Studio 2019 and create a new project by following these steps:
    - Go to "File" > "New" > "Project..."
    - In the "Create a new project" window, select "Web" from the left sidebar.
    - Choose "ASP.NET Web Application" as the project template.
    - Enter a name for your project and choose a location to save it.
    - Click "Create."
2. **Choose Web Application Template:** In the "Create a new ASP.NET Core Web Application" window:
    - Select "Empty" template. This will give you a clean slate to build your presentation layer.
3. **Add HTML and CSS Files:** Right-click on the "wwwroot" folder in the Solution Explorer and select "Add" > "New Folder." Name the folder "css" to store your CSS files.

    Right-click on the "css" folder and select "Add" > "New Item..." In the "Add New Item" dialog, select "Web" from the left sidebar and choose "Style Sheet (CSS)" from the available templates. Name the file "styles.css" and click "Add."
    Next, right-click on the "wwwroot" folder again and select "Add" > "New Item..." Choose "HTML Page" from the available templates. Name the file "index.html" and click "Add."
4. **Edit HTML and CSS:**

    Double-click on "index.html" to open it in the code editor.
    1.Define the Basic HTML Structure

a) Create the skeleton of the webpage with HTML.
b) Include sections like header, navigation, content, and footer.

2.Link CSS to HTML:

In the <**head**> section of your "index.html" file, link the "styles.css" file by adding the following line:<link rel=”stylesheet” href=”styles.css”>

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Presentation Layer</title>
<!-- Link to external CSS -->
<link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
<header class="header">
<h1>My Application</h1>
<nav>
<ul>
<li><a href=”home.html">Home</a></li>
<li><a href="about.html">About</a></li>
<li><a href="contact.html">Contact</a></li>
</ul>
</nav>
</header>
<main class="content">
<h2>Welcome to the Presentation Layer</h2>
<p>This is an example of a simple presentation layer.</p>
</main>
<footer class="footer">
<p>&copy; 2024 My Application. All rights reserved.</p>
</footer>
</body>
</html>
```

3. Style the HTML Using CSS
- Use CSS to add design elements like colors, fonts, spacing, and alignment.
- Make the layout responsive for different screen sizes.

Example (styles.css):

```
/* General styles */ body {
font-family: Arial, sans-serif; margin: 0;
padding: 0;
background-color: #f4f4f9;
}
```

```
/* Header styles */
.header {
background-color: #333; color: white;
padding: 20px;
text-align: center;
}

.header nav ul {
list-style-type: none; margin: 0;
padding: 0;
}

.header nav ul li { display: inline; margin-right: 15px;
}

.header nav ul li a { color: white;
text-decoration: none;
}

/* Content styles */
.content {
padding: 20px;
text-align: center;
}

.content h2 {
color: #333;
}

.content p {
color: #555;
}

/* Footer styles */
.footer {
background-color: #333; color: white;
text-align: center; padding: 10px; position: fixed; bottom: 0;
width: 100%;
}
```

4.Preview in Browser: Right-click on "index.html" in the Solution Explorer and select "View in Browser" to preview your presentation layer in a web browser.

Test and Iterate

- Save the HTML and CSS files.
- Open the HTML file in a web browser to view the output.
- Adjust the styles or layout as needed for a better user experience.

Output

1. Header: Contains the application name and navigation menu.

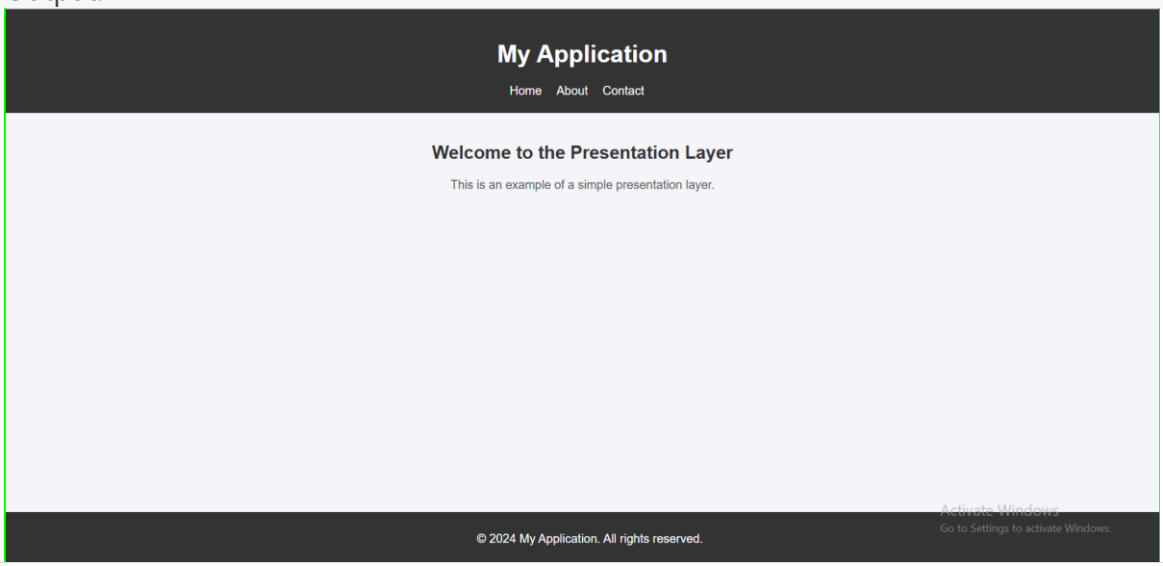
2. Content Area: Displays a welcome message and descriptive text.
3. Footer: Displays copyright information, fixed at the bottom. About Us page

5.Make Changes and Refine: Use Visual Studio's code editor to make changes to your HTML and CSS. You can preview the changes by refreshing the browser.

6. Responsive Design: To make your presentation layer responsive, you can add CSS media queries to your "styles.css" file. These queries will apply different styles based on the screen size.

Benefits of HTML and CSS in the Presentation Layer
1. Flexibility: HTML and CSS are lightweight and widely supported.
2. Customizability: Provides complete control over design and layout.
3. Responsiveness: CSS media queries enable dynamic adjustments for various screen sizes.

Output:-



Index.html

```html
<!DOCTYPE html>
<html>
<head>
    <title>Presentation Layer</title>
    <!-- Link to external CSS -->
    <link rel="stylesheet" type="text/css" href="StyleSheet1.css">
</head>
<body>
    <header class="header">
        <h1>My Application</h1>
        <nav>
            <ul>
                <li><a href="#home">Home</a></li>
                <li><a href="#about">About</a></li>
                <li><a href="#contact">Contact</a></li>
            </ul>
        </nav>
    </header>
    <main class="content">
        <h2>Welcome to the Presentation Layer</h2>
        <p>This is an example of a simple presentation layer.</p>
    </main>
    <footer class="footer">
        <p>&copy; 2024 My Application. All rights reserved.</p>
    </footer>
</body>
</html>
```

109 % No issues found

styles.css

```css
/* General styles */

body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f9;
}

/* Header styles */
.header {
    background-color: #333;
    color: white;
    padding: 20px;
    text-align: center;
}

    .header nav ul {
        list-style-type: none;
        margin: 0;
        padding: 0;
    }

        .header nav ul li {
            display: inline;
            margin-right: 15px;
        }

            .header nav ul li a {
                color: white;
                text-decoration: none;
            }

/* Content styles */
.content {
    padding: 20px;
    text-align: center;
}

    .content h2 {
        color: #333;
    }

    .content p {
        color: #555;
    }

/* Footer styles */
.footer {
    background-color: #333;
    color: white;
    text-align: center;
    padding: 10px;
    position: fixed;
    bottom: 0;
    width: 100%;
}
```

Practical 3 : To study and create Master page ,user control etc.

A Master Page in ASP.NET is a template that allows you to define a consistent layout for multiple pages within a web application. It provides a way to structure the common parts of your web pages (such as headers, footers, navigation bars, etc.) in a central location, and then apply that structure to many other pages, known as Content Pages.

Key Features of Master Pages:

1. Consistency: Ensures a uniform layout across multiple pages.

2. Reusability: A single Master Page can be applied to several content pages, saving development time and reducing redundancy.

3. Maintainability: Changes made to the Master Page (like altering the layout or design) automatically reflect on all pages that use the Master Page.

How It Works:

- A Master Page defines a layout template that contains placeholders for dynamic content. The common elements like a header, footer, or menu are placed in the Master Page.
- A Content Page defines only the unique content for that page and links to the Master Page for layout.
- The Master Page and Content Pages work together to render a complete web page with the common layout and the specific content for each page.

Steps to Create a Master Page in ASP.NET using Visual Studio 2019

Step 1: Open **Visual Studio 2019**.

Step 2: Create a new project.

Step 3: Go to
`File → New → Project → ASP.NET Web Application (C#).`

Step 4: Enter your project name in the **Project Name** text box (e.g., `demoMasterPage`) and click the **Create** button.

Step 5: In the next window, select **Empty** (Empty Project Template) to create a new ASP.NET Web Application, then click **Create**.

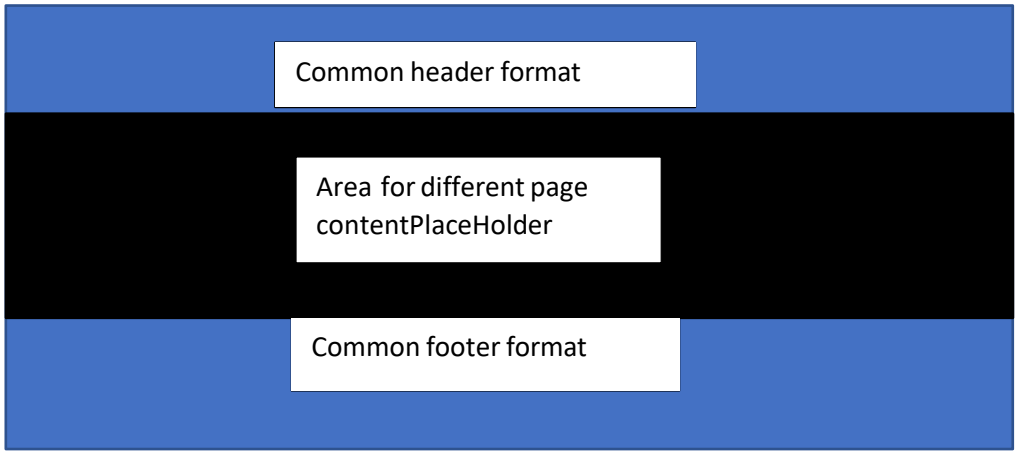
Step 6: In the **Solution Explorer**, right-click on the project name (`demoMasterPage`) → **Add** → **New Item** → choose **Web Form Master Page**.

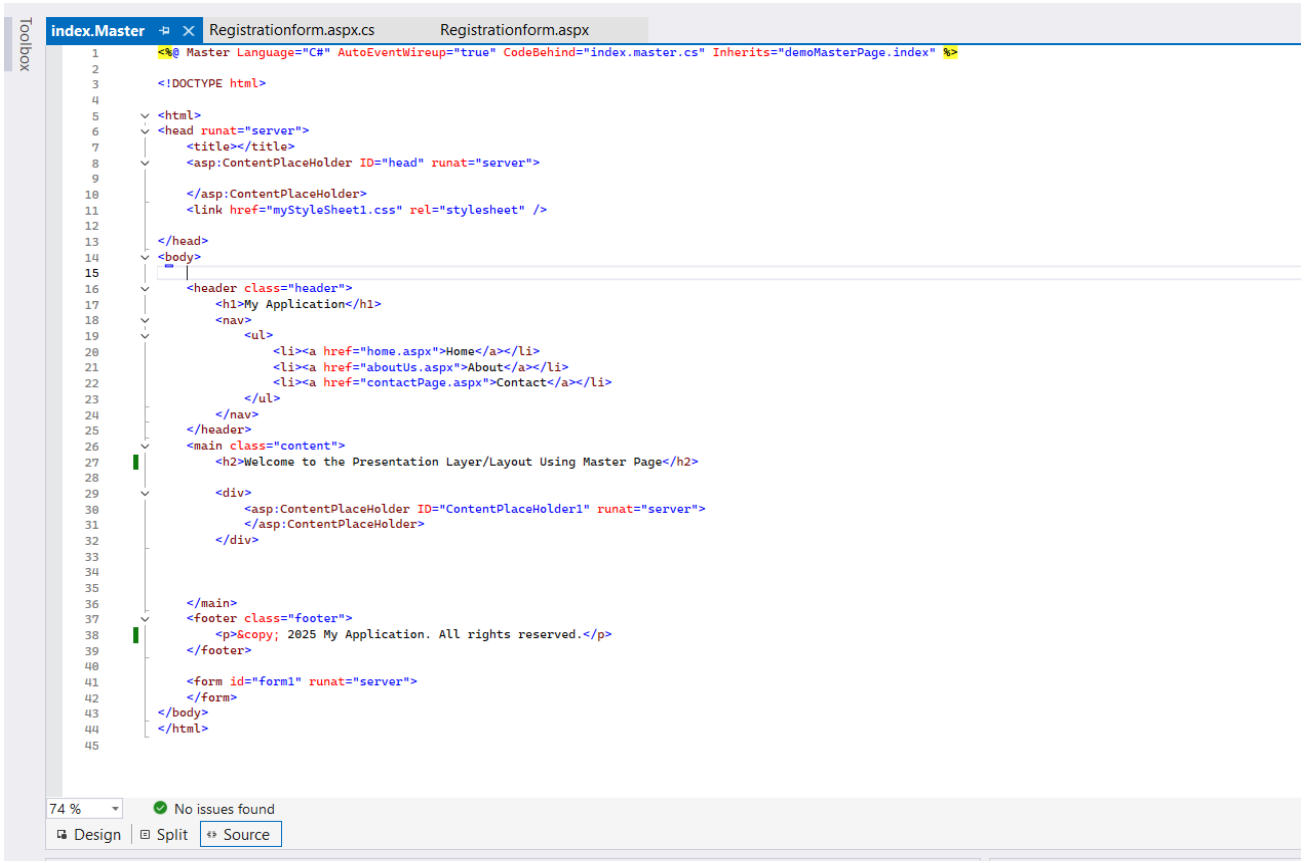
- Enter the name of the master page (e.g., `index.master`).
- Click **Add**.
- You will now see that the Master Page has been inserted into your project.

Step 7: Design the layout of the Master Page.

Step 8: After designing the **Header, Footer, and Navigation Bar** using CSS:

- Create a CSS file.
- Add a link to the CSS file inside the `<head>` tag of the Master Page's source code.



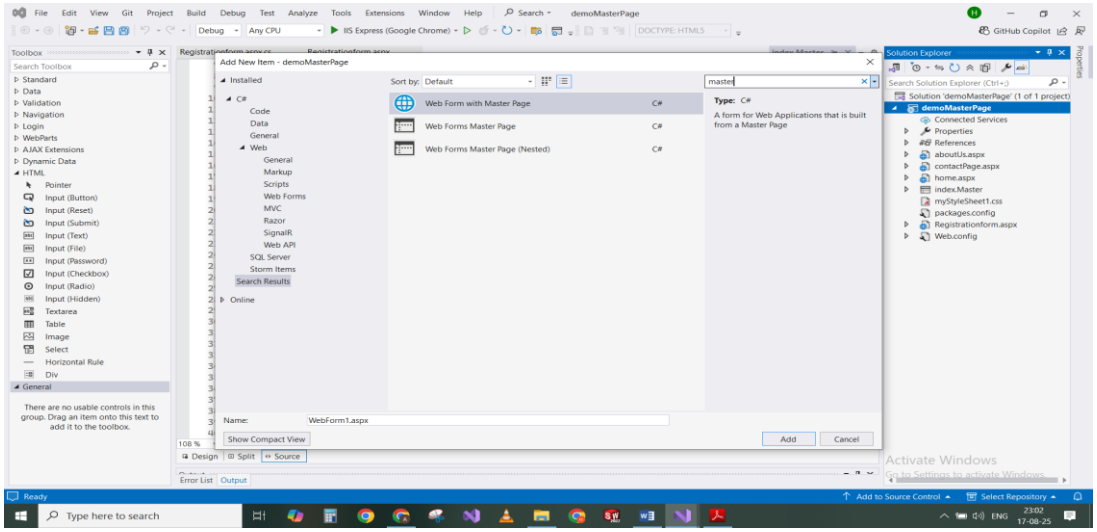


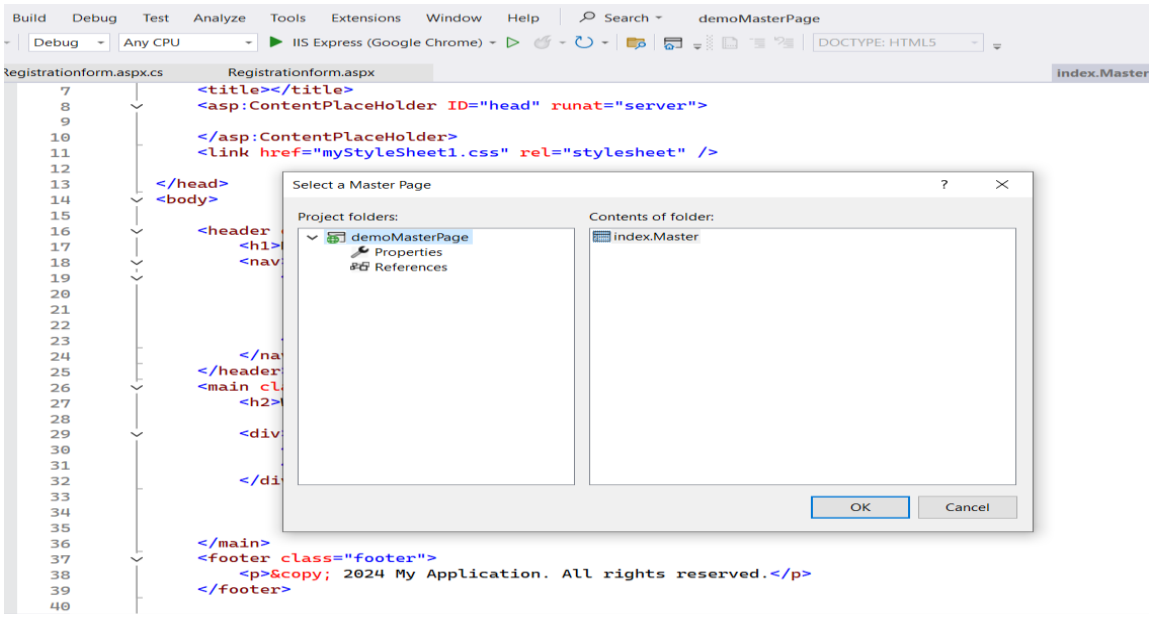
Step 9: Add a new Web Form with Master Page.

- Right-click on the project → **Add** → **New Item** → select **Web Form with Master Page**.
- Enter the name as **Home.aspx** → click **Add**.
- A pop-up window will appear → select **index.master** (the Master Page you created earlier) → click **OK**.

Step 10: Repeat **Step 9** to add additional pages:

- Create **About.aspx** and link it to **index.master**.
- Create **Contact.aspx** and link it to **index.master**.

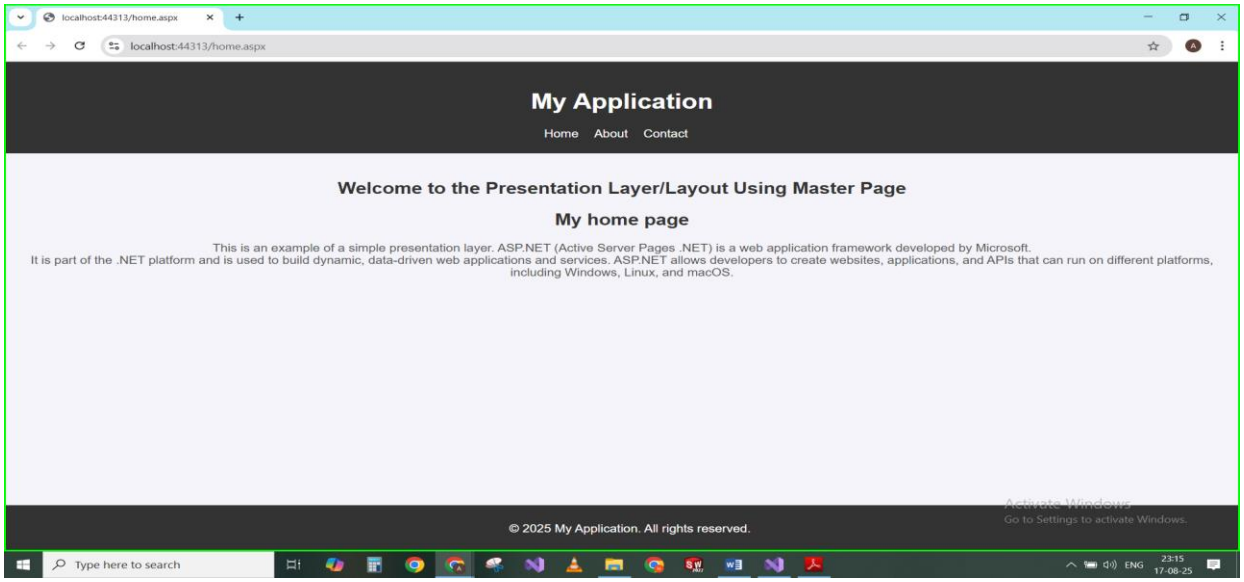




Step 11: add pages name in nav bar link pages.

```
<header class="header">
    <h1>My Application</h1>
    <nav>
        <ul>
            <li><a href="home.aspx">Home</a></li>
            <li><a href="aboutUs.aspx">About</a></li>
            <li><a href="contactPage.aspx">Contact</a></li>
        </ul>
    </nav>
</header>
```

This is my My web application Layout using Master page


localhost:44313/home.aspx
localhost:44313/home.aspx
My Application
Home About Contact
Welcome to the Presentation Layer/Layout Using Master Page
My home page
This is an example of a simple presentation layer. ASP.NET (Active Server Pages .NET) is a web application framework developed by Microsoft.
It is part of the .NET platform and is used to build dynamic, data-driven web applications and services. ASP.NET allows developers to create websites, applications, and APIs that can run on different platforms, including Windows, Linux, and macOS.
© 2025 My Application. All rights reserved.
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
23:15
17-08-25