Session 15: GUI/Desktop Apps with C#

Objectives

- Describe Web applications
- List the steps to create Web app using C# 9 and Visual Studio 2019
- Explain what are UWP apps
- List steps to create UWP apps
- Explain WPF and Windows Forms
- Describe creation of WPF app and Windows Forms app

Web Applications Using .NET 1-2

A Web application:

- Runs on a Web server
- Are accessed by the user through a Web browser
- Allows you to share and access information over the Internet
- Can help you perform online transactions

Architecture of a Web application depends on the system in which layers of the application are distributed and communicated to each other.

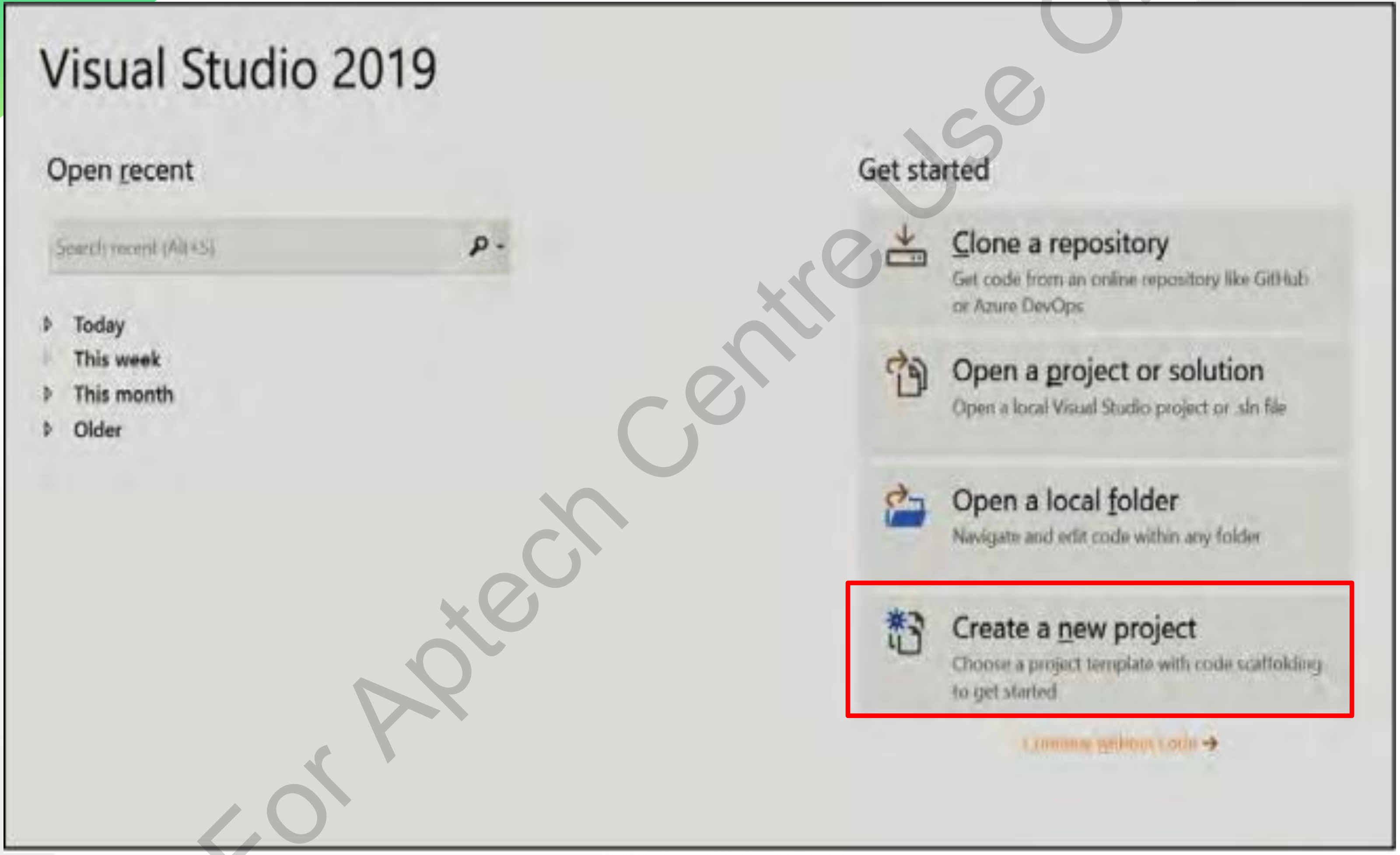
C# 9.0, .NET 5.0, and Visual Studio 2019 provides a complete platform to create a Web application.

Web Applications Using .NET 2-2

Microsoft ASP.NET

- Is a framework for developing dynamic Web applications.
- Is a server-side technology with simplicity, security, and scalability.
- Applications comprise .aspx Web pages that combine both client- and server-side scripts.
- Has ASP.NET Core, released in 2016, as a successor.
- Helps developers to build Web apps and services.

Creating a Web App Using C# 1-13



Visual Studio 2019

Open recent

Search recent (Alt+S)

- Today
- This week
- This month
- Older

Get started

Clone a repository
Get code from an online repository like GitHub or Azure DevOps

Open a project or solution
Open a local Visual Studio project or .sln file

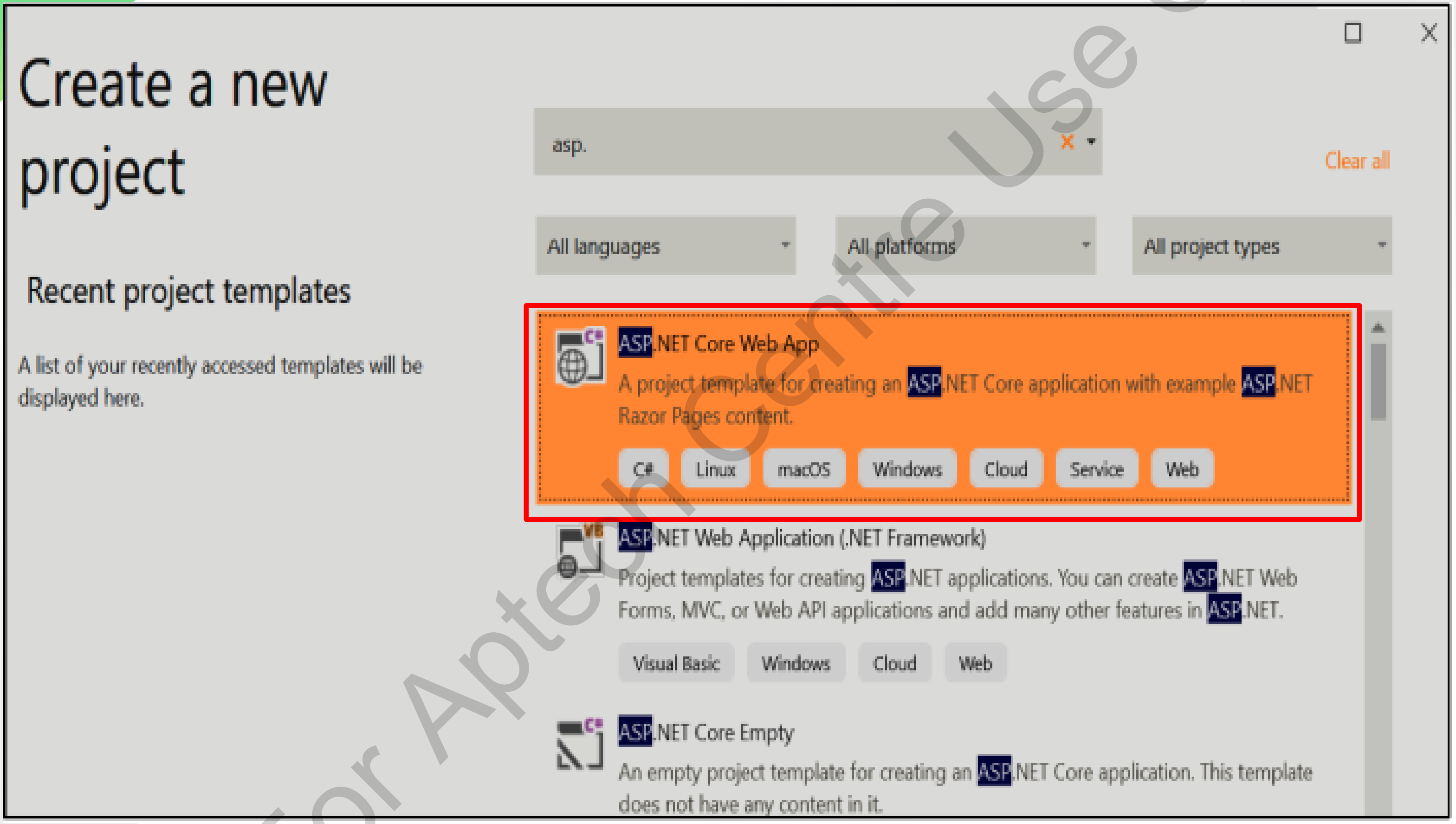
Open a local folder
Navigate and edit code within any folder

Create a new project
Choose a project template with code scaffolding to get started

Continue without code →

Start Window of Visual Studio 2019

Creating a Web App Using C# 2-13



A list of your recently accessed templates will be displayed here.

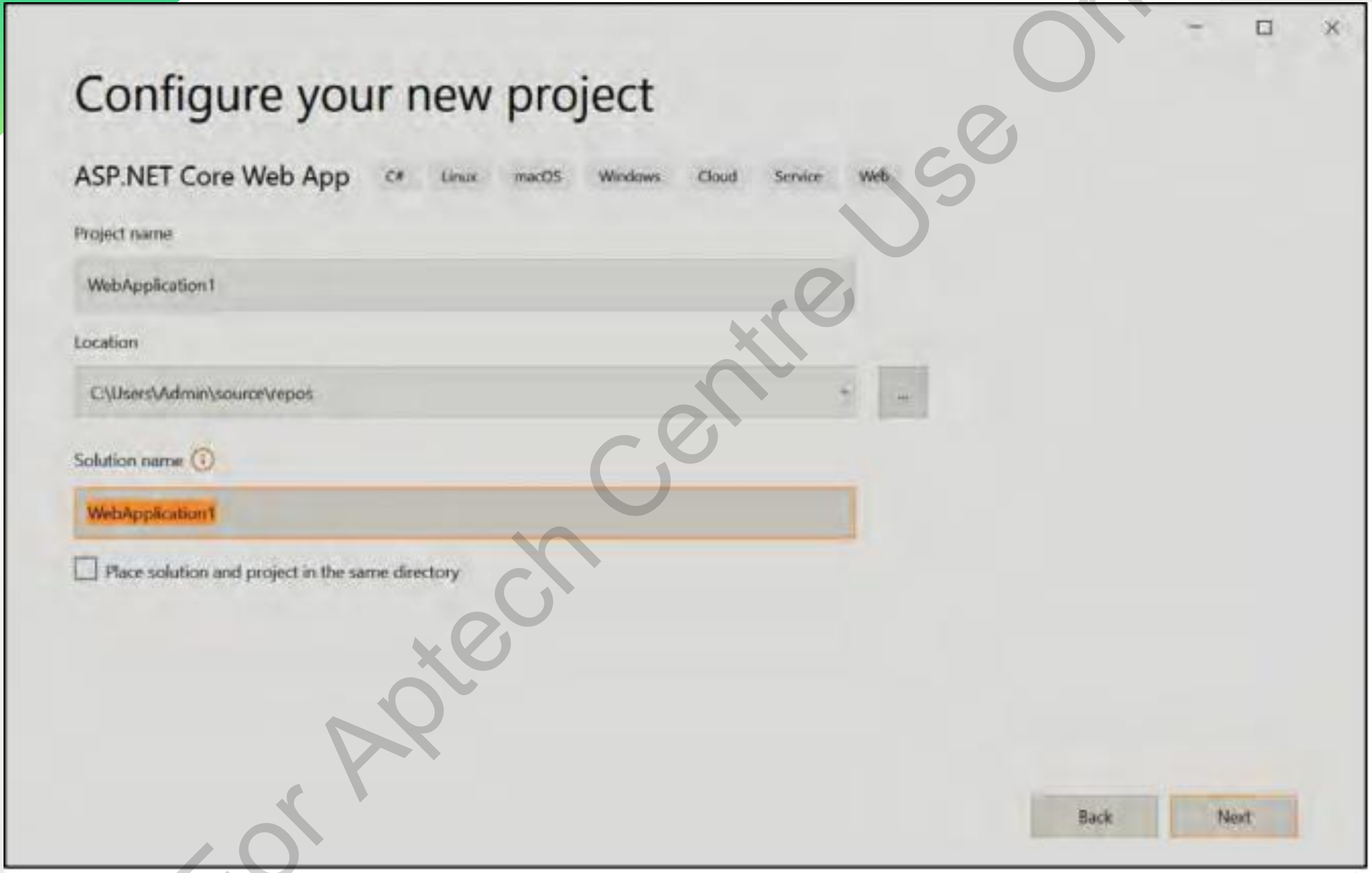
ASP.NET Core Web App
A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.
C# Linux macOS Windows Cloud Service Web

ASP.NET Web Application (.NET Framework)
Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.
Visual Basic Windows Cloud Web

ASP.NET Core Empty
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.

ASP.NET Core WebApp

Creating a Web App Using C# 3-13


Configure your new project

ASP.NET Core Web App

Project name: WebApplication1

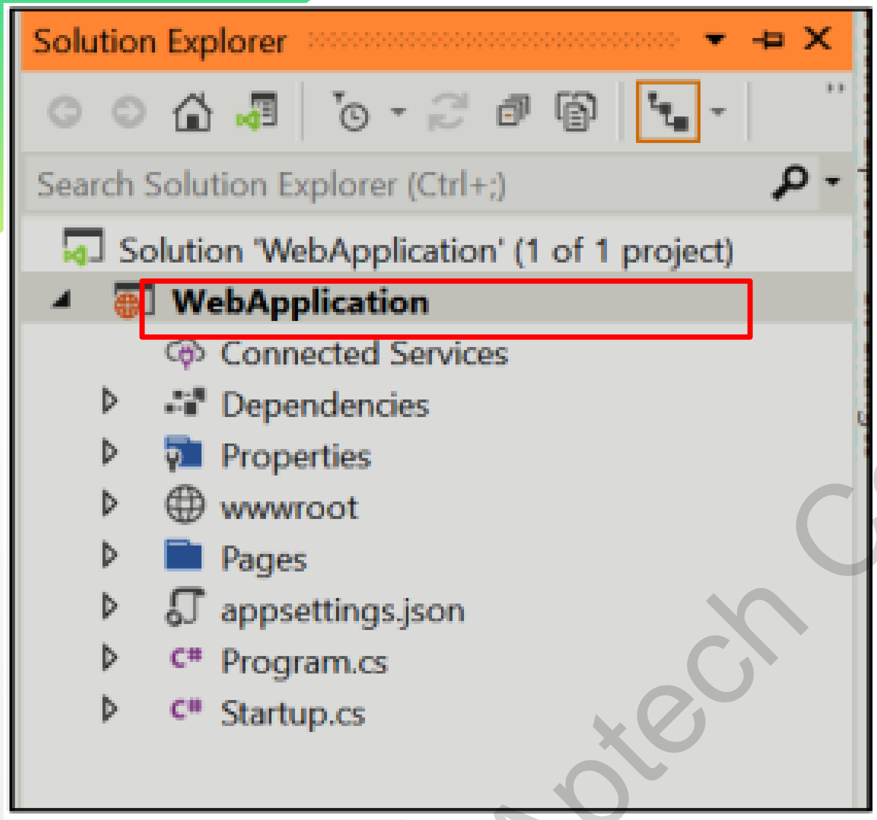
Location: C:\Users\Admin\source\repos

Solution name: WebApplication1

☐ Place solution and project in the same directory

Naming the Project and Specifying Location

Creating a Web App Using C# 4-13

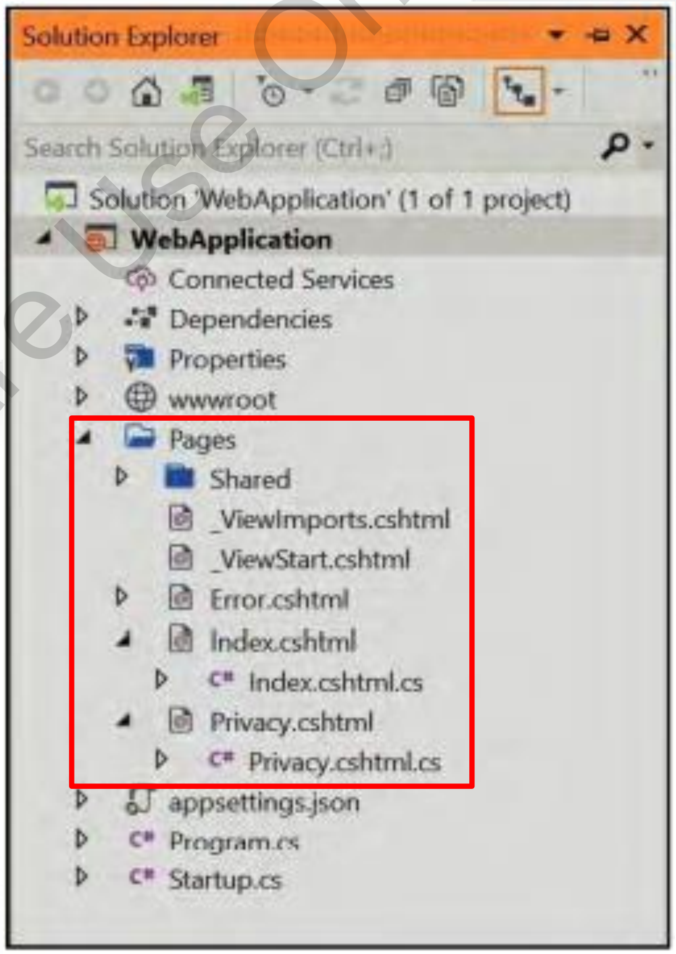


Solution Explorer

Search Solution Explorer (Ctrl+;)

- Solution 'WebApplication' (1 of 1 project)
  - WebApplication
    - Connected Services
    - Dependencies
    - Properties
    - wwwroot
    - Pages
    - appsettings.json
    - Program.cs
    - Startup.cs

Solution Explorer Showing ASP.NET Core Web Application



Solution Explorer

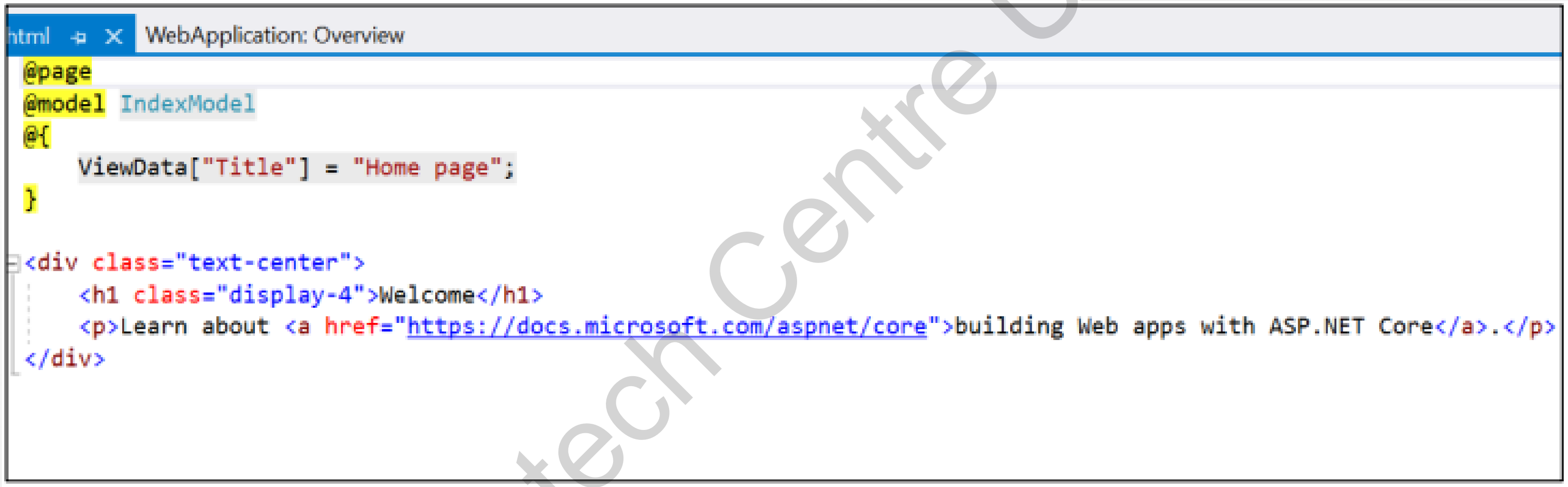
Search Solution Explorer (Ctrl+;)

- Solution 'WebApplication' (1 of 1 project)
  - WebApplication
    - Connected Services
    - Dependencies
    - Properties
    - wwwroot
    - Pages
      - Shared
      - _ViewImports.cshtml
      - _ViewStart.cshtml
      - Error.cshtml
      - Index.cshtml
        - Index.cshtml.cs
      - Privacy.cshtml
        - Privacy.cshtml.cs
    - appsettings.json
    - Program.cs
    - Startup.cs

Pages Folder

Creating a Web App Using C# 5-13

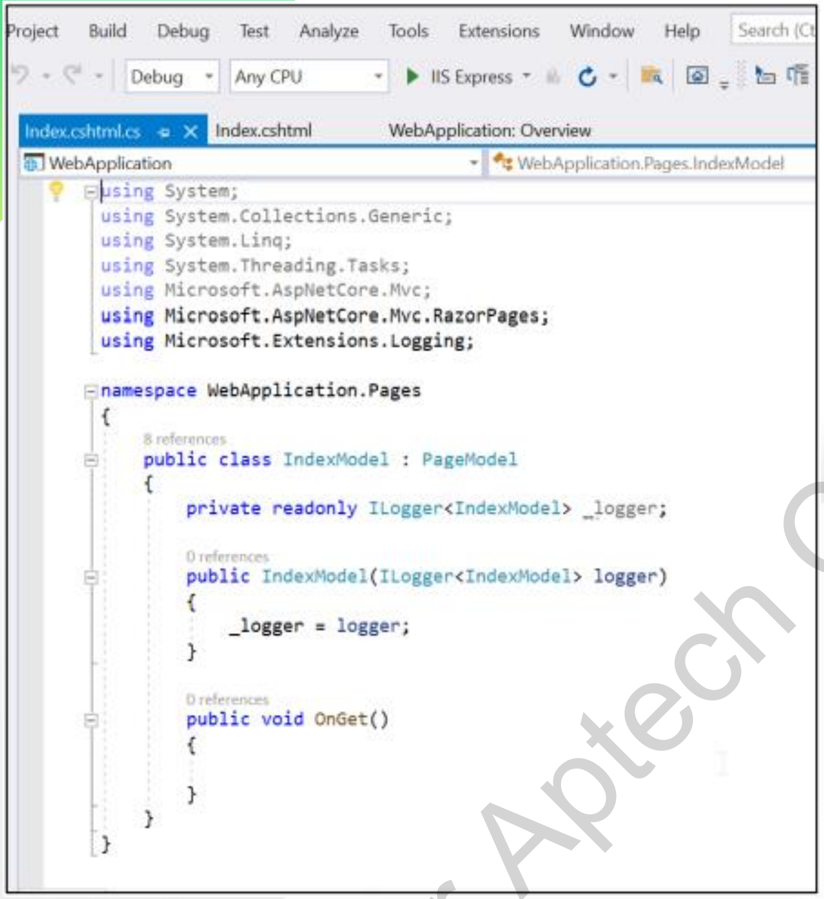
- Edit the boilerplate text and give a title to the HTML page, add links of CSS, and add Google fonts in the `Index.cshtml` file, if required.



```
@page
@model IndexModel
@{
    ViewData["Title"] = "Home page";
}

<div class="text-center">
    <h1 class="display-4">Welcome</h1>
    <p>Learn about <a href="https://docs.microsoft.com/aspnet/core">building Web apps with ASP.NET Core</a>.</p>
</div>
```

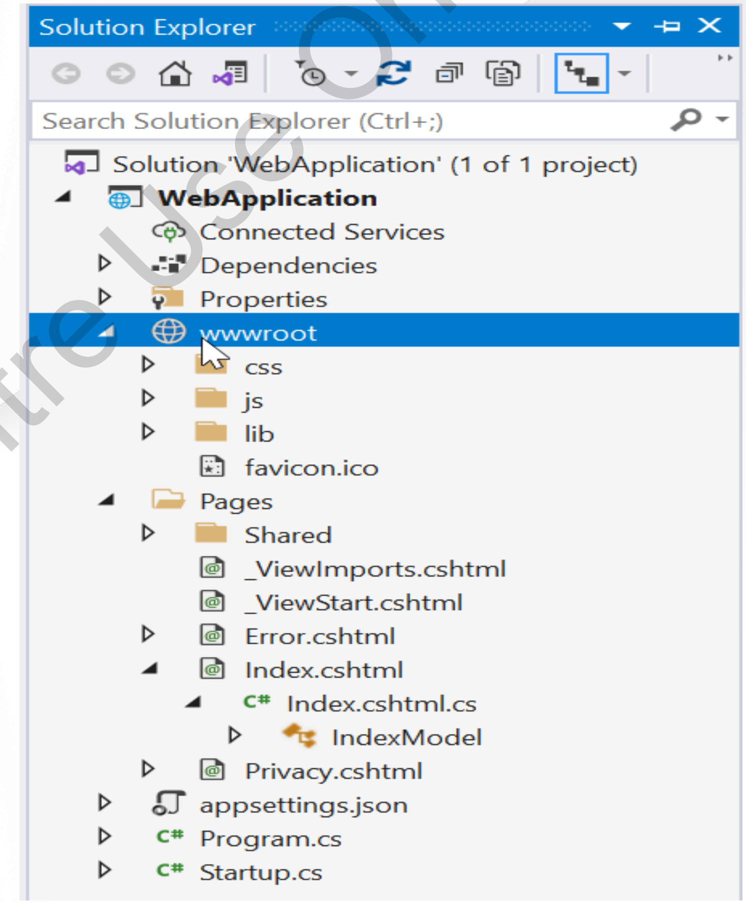
Default Autogenerated Code for Index File

Creating a Web App Using C# 6-13


Project Build Debug Test Analyze Tools Extensions Window Help
Debug
Any CPU
IIS Express
Index.cshtml.cs
Index.cshtml
WebApplication: Overview
WebApplication
WebApplication.Pages.IndexModel

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using Microsoft.Extensions.Logging;

namespace WebApplication.Pages
{
    8 references
    public class IndexModel : PageModel
    {
        private readonly ILogger<IndexModel> _logger;

        0 references
        public IndexModel(ILogger<IndexModel> logger)
        {
            _logger = logger;
        }

        0 references
        public void OnGet()
        {

        }
    }
}
```

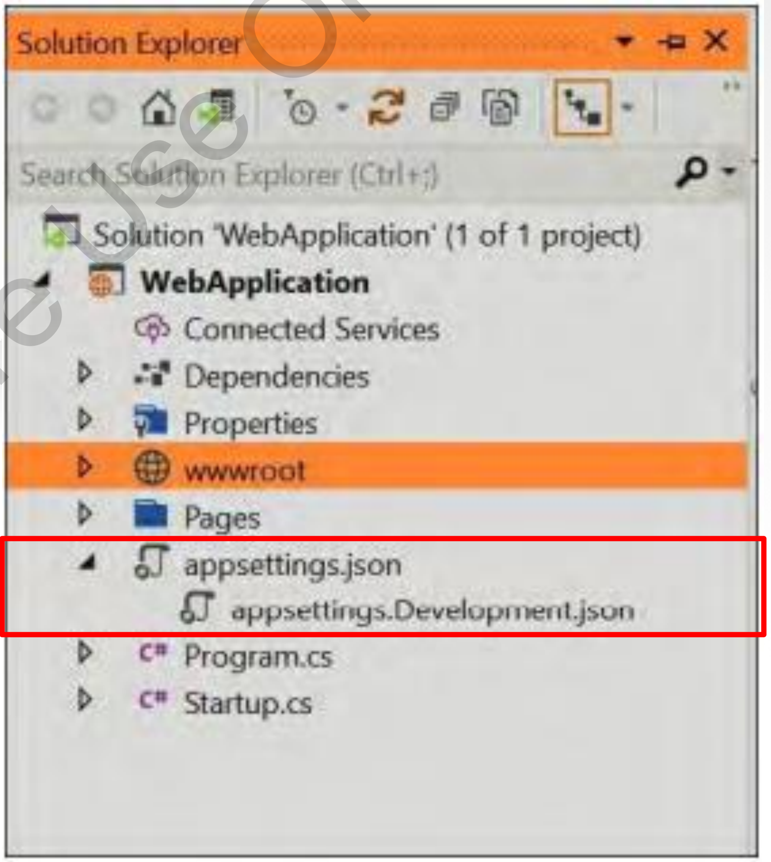
Code File in Editor



Root Folder

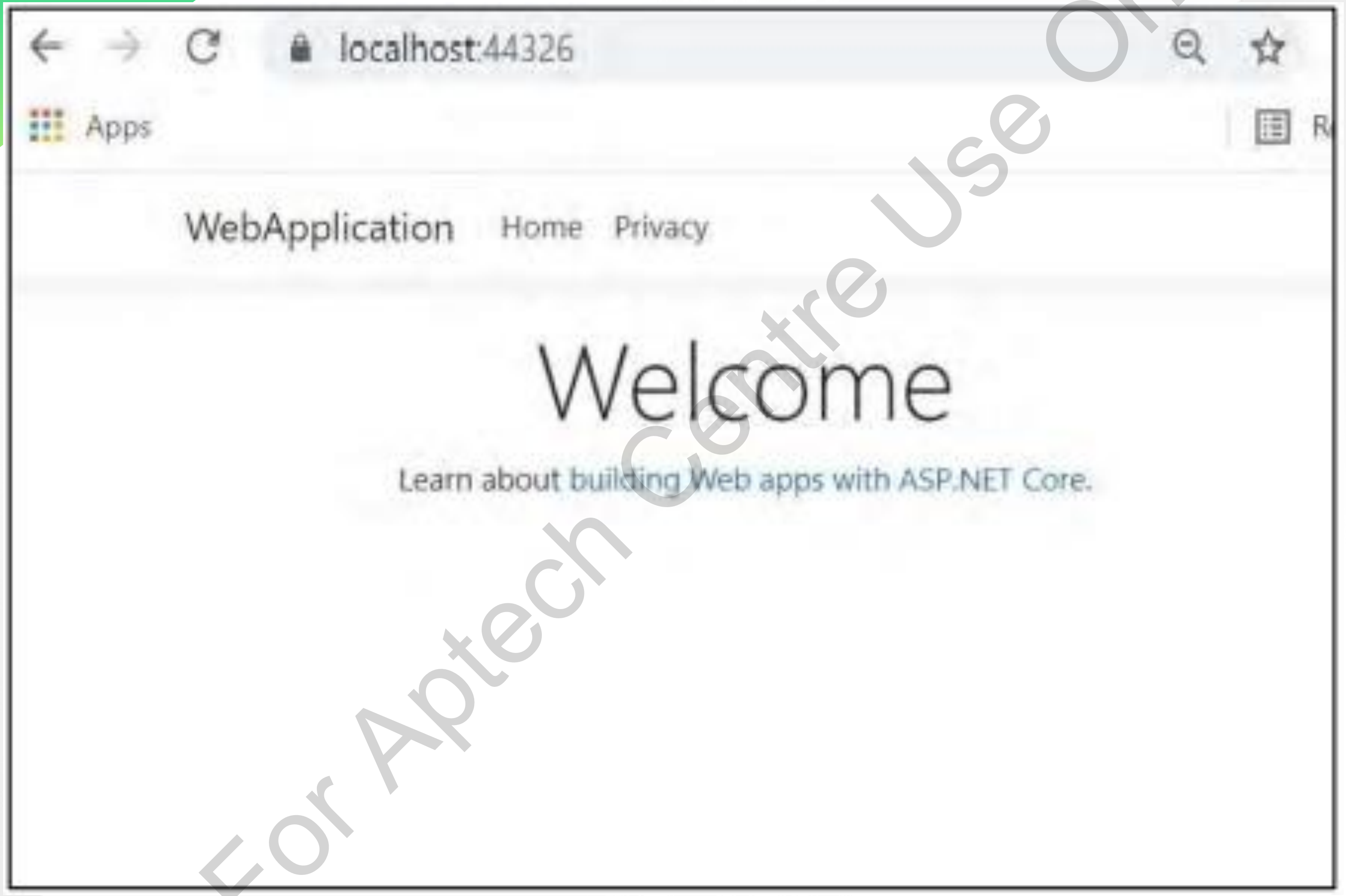
Creating a Web App Using C# 7-13

- To view the `appsettings.Development.json` file, expand `appsettings.json`



`appsettings.json`

Creating a Web App Using C# 8-13

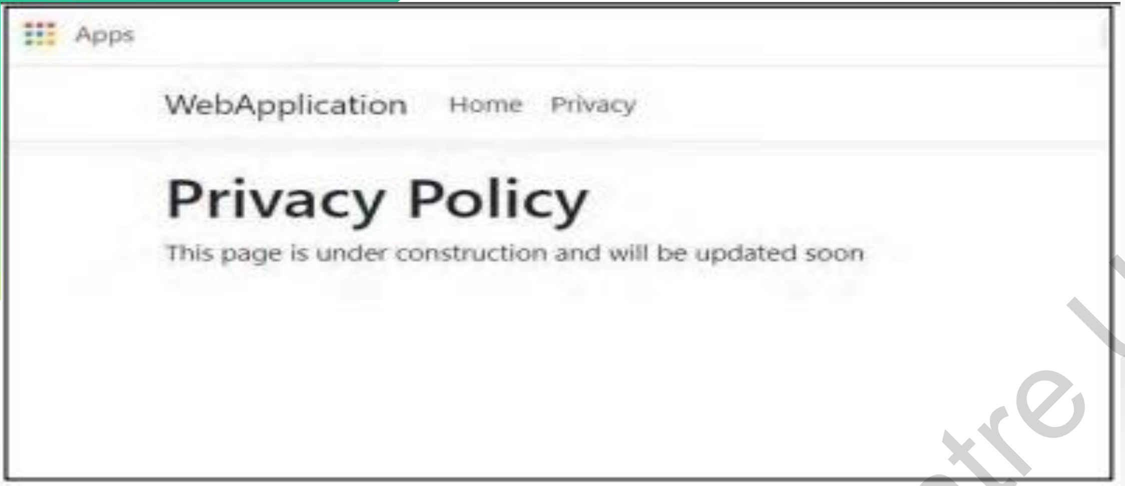


Welcome

Learn about building Web apps with ASP.NET Core.

Output of Web Application in Browser Window

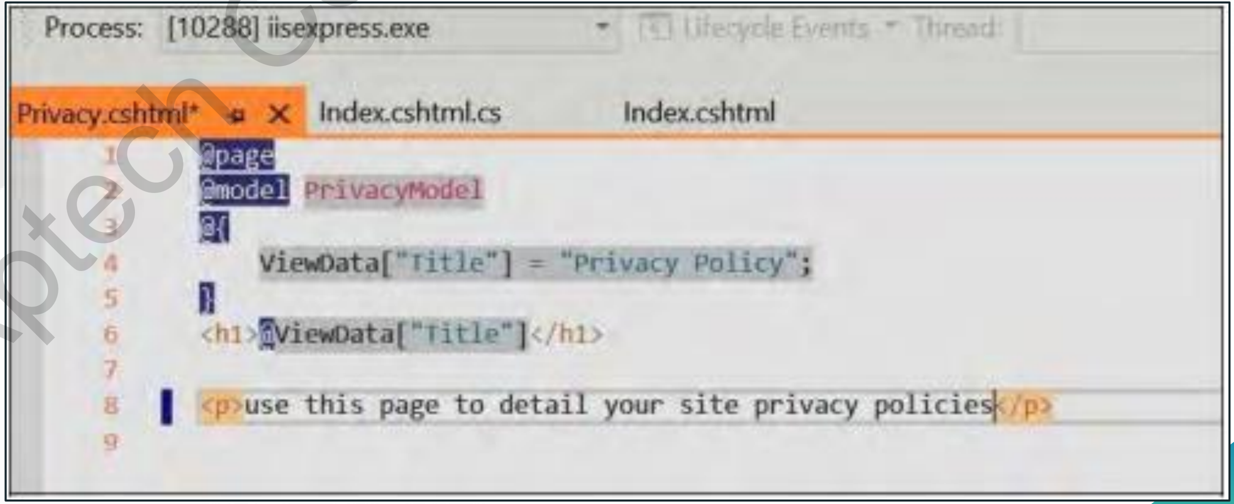
Creating a Web App Using C# 9-13



Privacy Policy

This page is under construction and will be updated soon

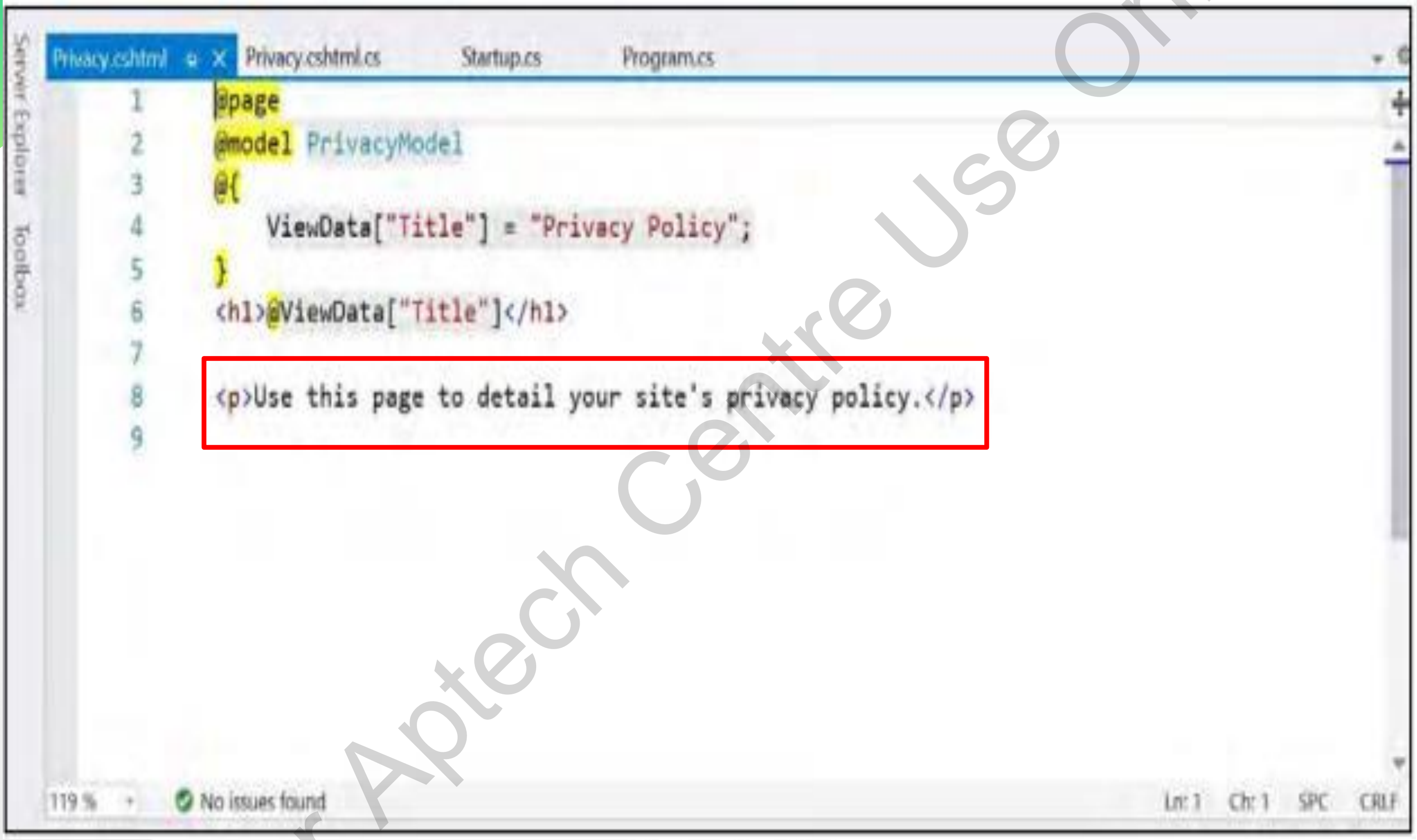
Privacy Tab



```
@page
@model PrivacyModel
@{
    ViewData["Title"] = "Privacy Policy";
}
<h1>@ViewData["Title"]</h1>

<p>use this page to detail your site privacy policies</p>
```

Privacy.cshtml

Creating a Web App Using C# 10-13

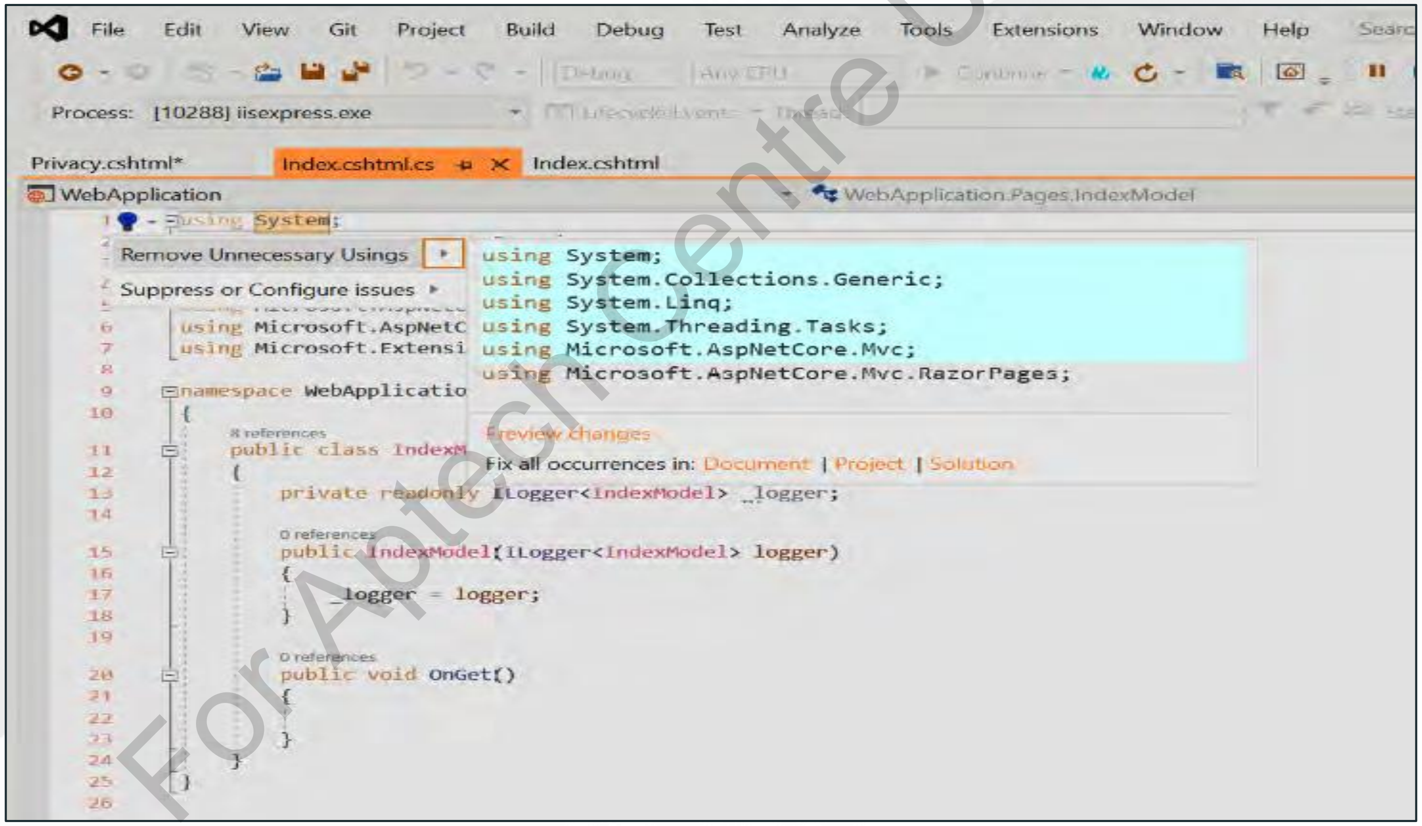


```
@page
@model PrivacyModel
@{
    ViewData["Title"] = "Privacy Policy";
}
<h1>@ViewData["Title"]</h1>

<p>Use this page to detail your site's privacy policy.</p>

```

Adding the Customized Content

Creating a Web App Using C# 11-13

- Select `Privacy.cshtml.cs` and remove/clean up the using directive which are in grey color.
- Right-click the grey colored code and select the light bulb icon. From the drop-down menu, select ‘Remove unnecessary using’ as shown:


File
Edit
View
Git
Project
Build
Debug
Test
Analyze
Tools
Extensions
Window
Help
Process: [10288] iisexpress.exe
Privacy.cshtml*
Index.cshtml.cs
Index.cshtml
WebApplication
WebApplication.Pages.IndexModel
Remove Unnecessary Usings
Suppress or Configure issues
Preview changes
Fix all occurrences in: Document | Project | Solution

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
```

```
using System;

using Microsoft.AspNetC
using Microsoft.Extensi

namespace WebApplicatio
{
    8 references
    public class IndexM
    {
        private readonly ILogger<IndexModel> _logger;

        0 references
        public IndexModel(ILogger<IndexModel> logger)
        {
            _logger = logger;
        }

        0 references
        public void OnGet()
        {

        }
    }
}
```

Removing Unnecessary `Using`

Creating a Web App Using C# 12-13

Code Snippet 1:

```
public void OnGet()
{
    string dateTime =
         DateTime.Now.ToShortDateString();
    ViewData["TimeStamp"] = dateTime;
}
```

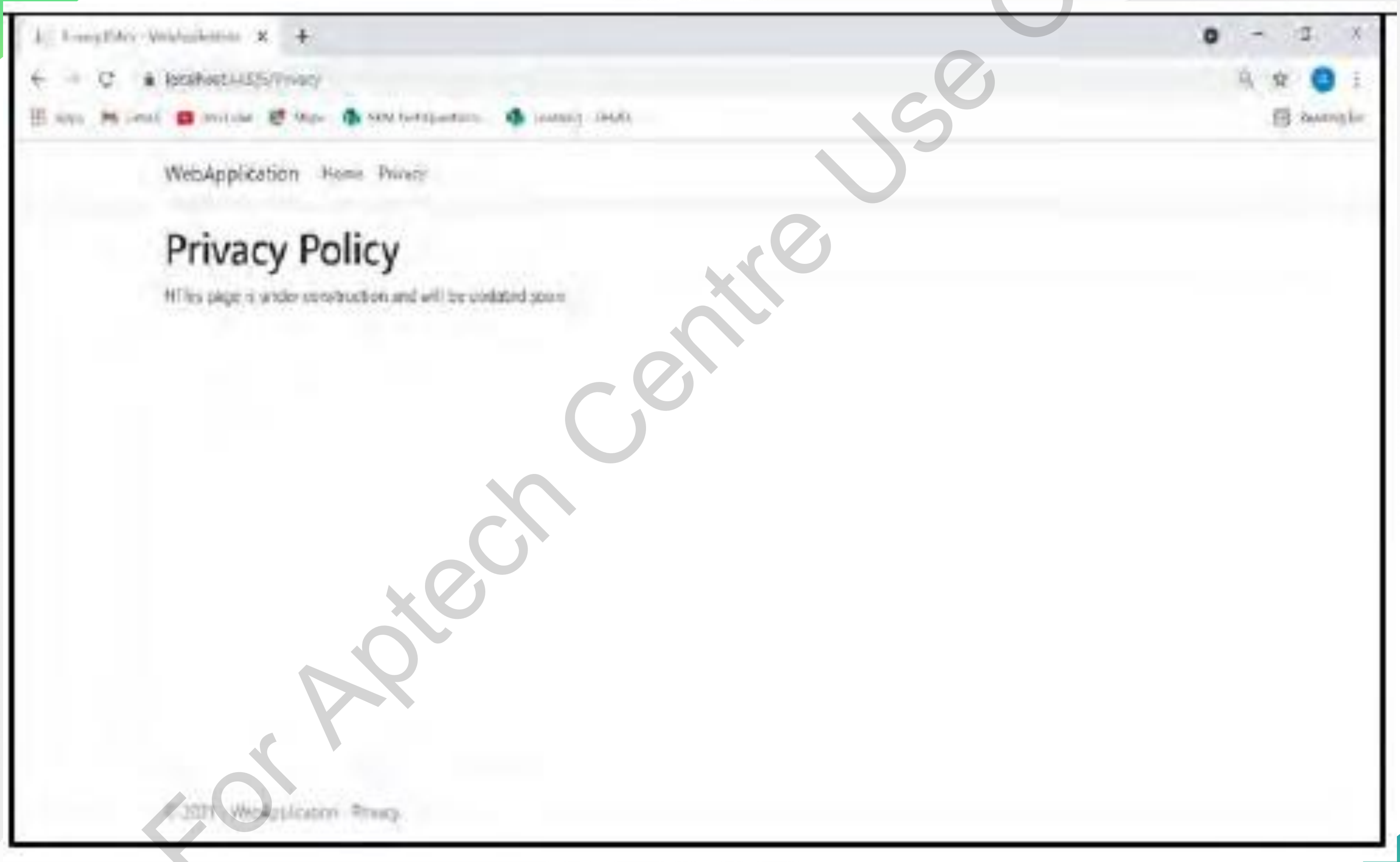
- An error will be displayed under `DateTime`. The error is displayed because `DateTime` data type is not in the scope. Following error message is displayed:

  CS0103 The name 'Date Time' does not exist in the current context.

- To fix the error, right-click the error and select Quick Action. A drop-down menu will appear. From the drop-down menu, select `using System` to add directives.

Creating a Web App Using C# 13-13

Output:


WebApplication Home Privacy

Privacy Policy

Hi! This page is under construction and will be updated soon

© 2021 - WebApplication - Privacy

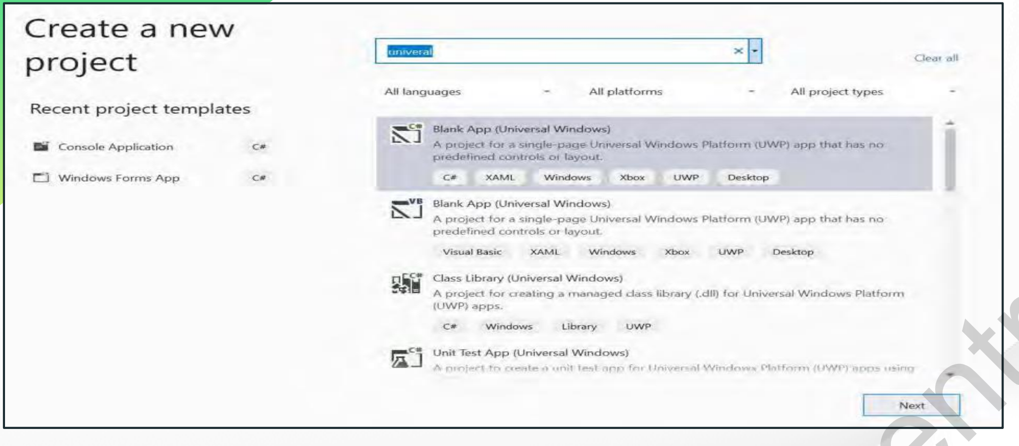
Creating Universal Windows Platform Apps

- Universal Windows Platform (UWP) is used to create client applications for Windows. Visual Studio 2019 along with .NET 5.0 supports creation of a UWP app. UWP apps use WinRT APIs to provide powerful UI and advanced asynchronous features.

- A UWP app is secure and uses a common API on all devices that run Windows 10. It is also programmable in C#, C++, Visual Basic, and JavaScript.

- UWP makes use of Extensible Application Markup Language (XAML) for creating the UI, which provides a declarative model for application programming. It also makes use of WinUI, HTML, and optionally, DirectX.

Create the Project 1-2



Create a new project

Recent project templates

Console Application C#

Windows Forms App C#

univeral

Clear all

All languages · All platforms · All project types

Blank App (Universal Windows)
A project for a single-page Universal Windows Platform (UWP) app that has no predefined controls or layout.
C# XAML Windows Xbox UWP Desktop

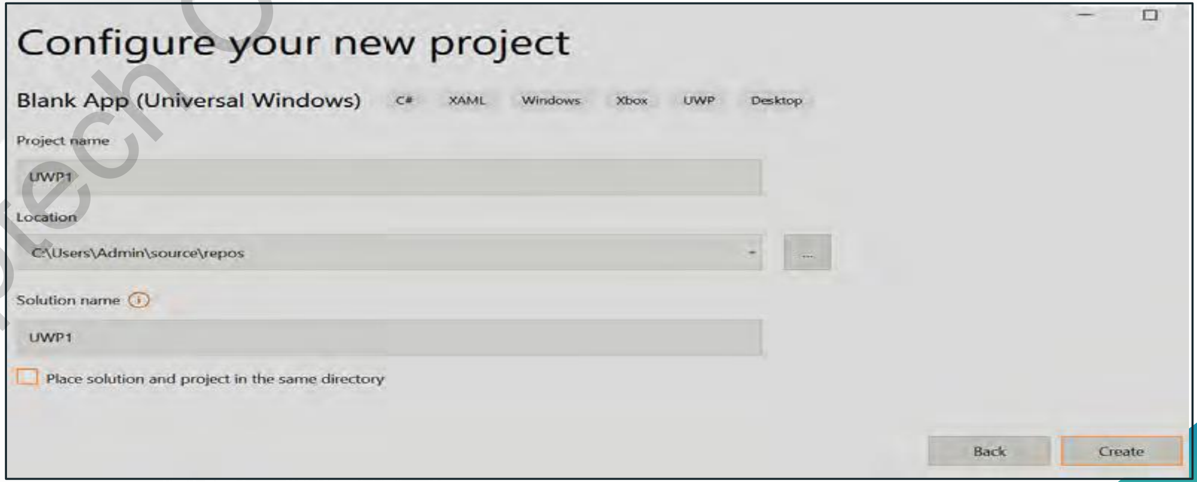
Blank App (Universal Windows)
A project for a single-page Universal Windows Platform (UWP) app that has no predefined controls or layout.
Visual Basic XAML Windows Xbox UWP Desktop

Class Library (Universal Windows)
A project for creating a managed class library (.dll) for Universal Windows Platform (UWP) apps.
C# Windows Library UWP

Unit Test App (Universal Windows)
A project to create a unit test app for Universal Windows Platform (UWP) apps using

Next

Create a New Project Window



Configure your new project

Blank App (Universal Windows) C# XAML Windows Xbox UWP Desktop

Project name

UWP1

Location

C:\Users\Admin\source\repos

Solution name

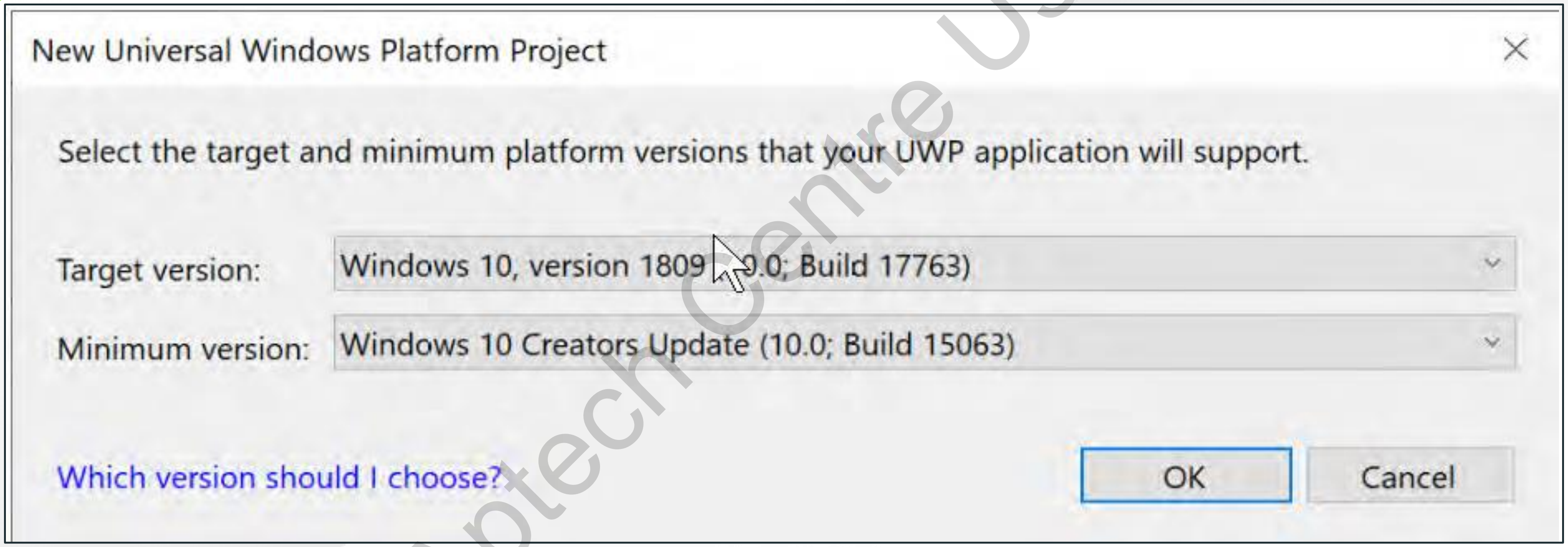
UWP1

☐ Place solution and project in the same directory

Back Create

Adding the Name of the Project

Create the Project 2-2



New Universal Windows Platform Project

Select the target and minimum platform versions that your UWP application will support.

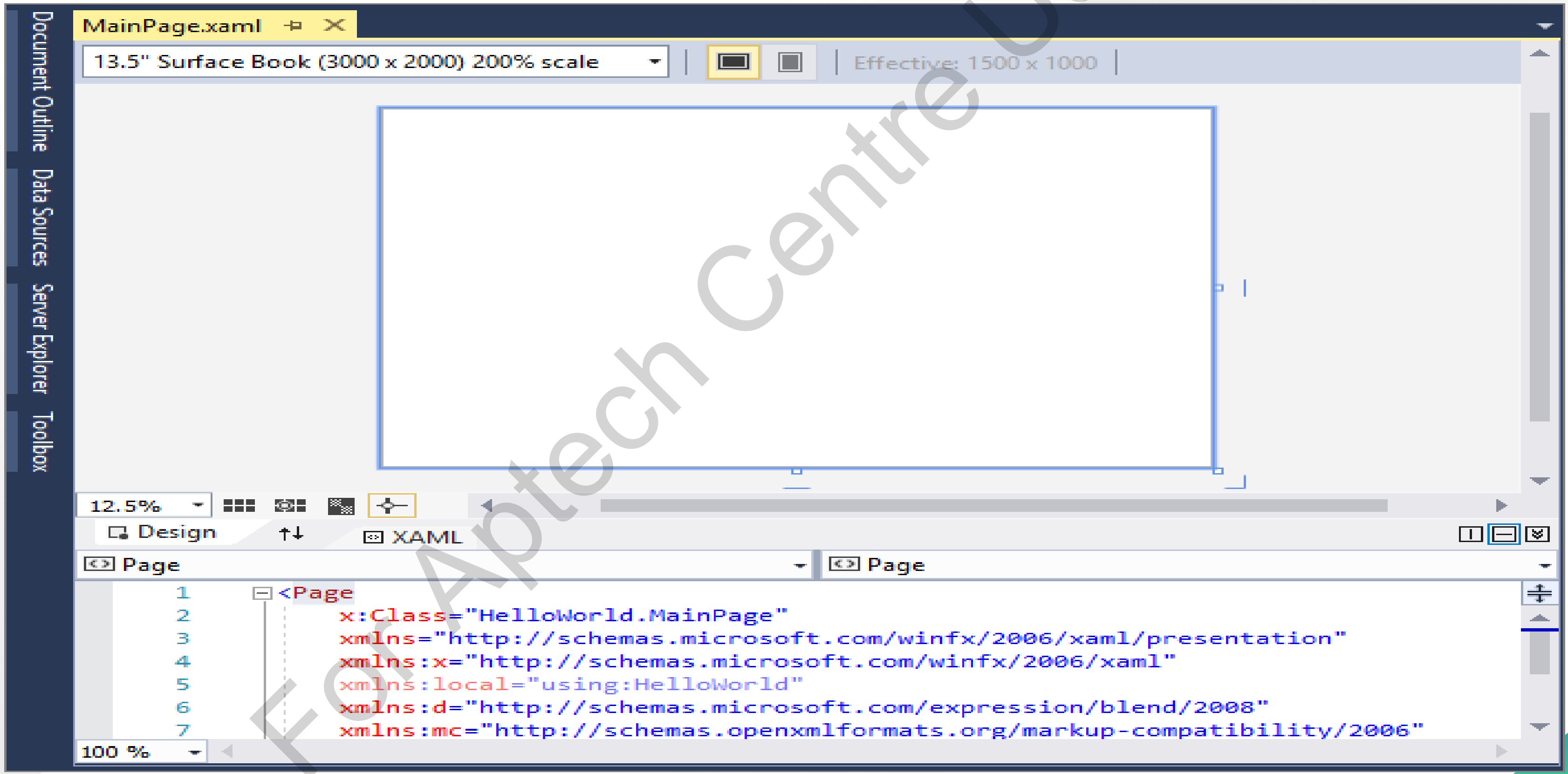
Target version: Windows 10, version 1809 (10.0; Build 17763)

Minimum version: Windows 10 Creators Update (10.0; Build 15063)

Default Target Version Window

Create the Application 1-4

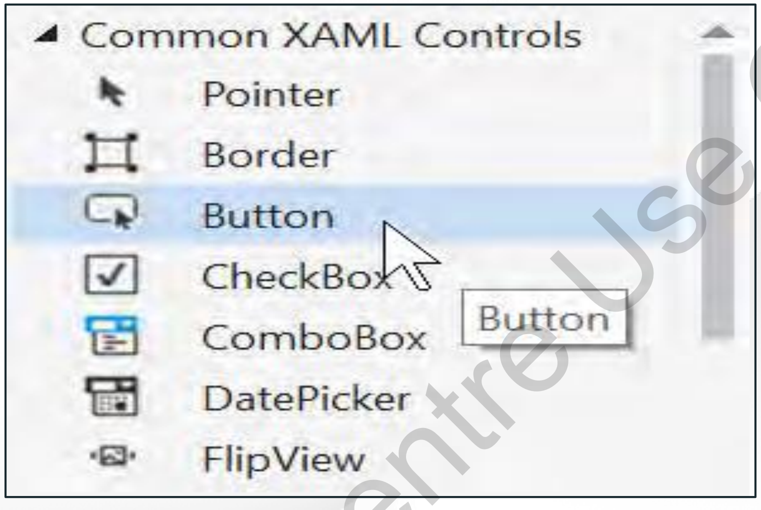
- A Button control and action for the button will be added. The Toolbox displays a wide variety of XAML controls, one of which is the Button control.
- There are several types of Buttons such as Button, HyperlinkButton, ToggleButton, and so on.



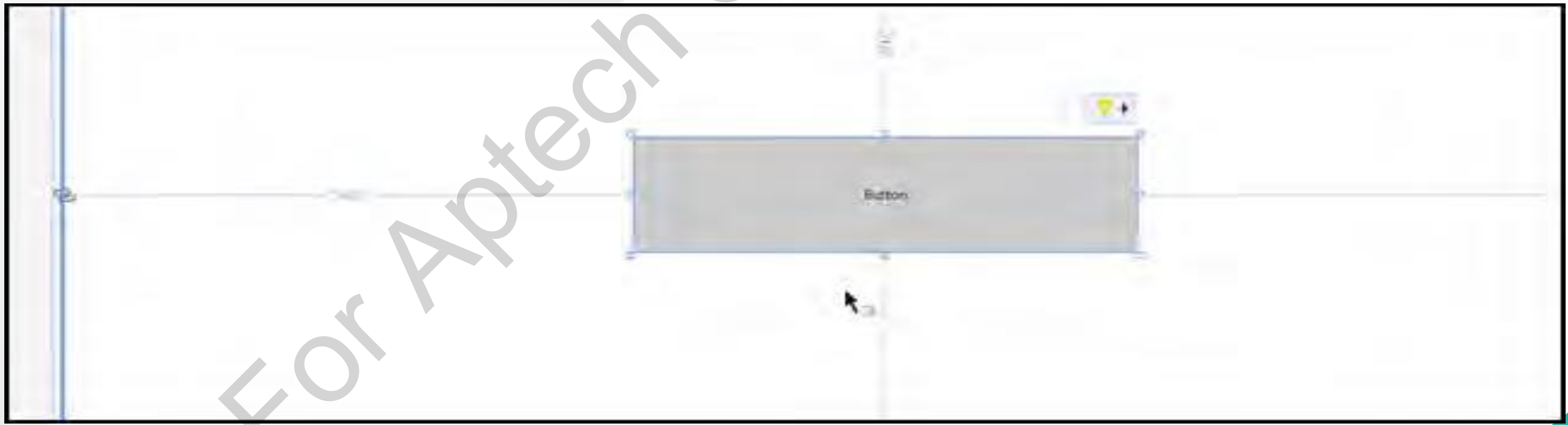
```
<Page
    x:Class="HelloWorld.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:HelloWorld"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
```

XAML Editor

Create the Application 2-4

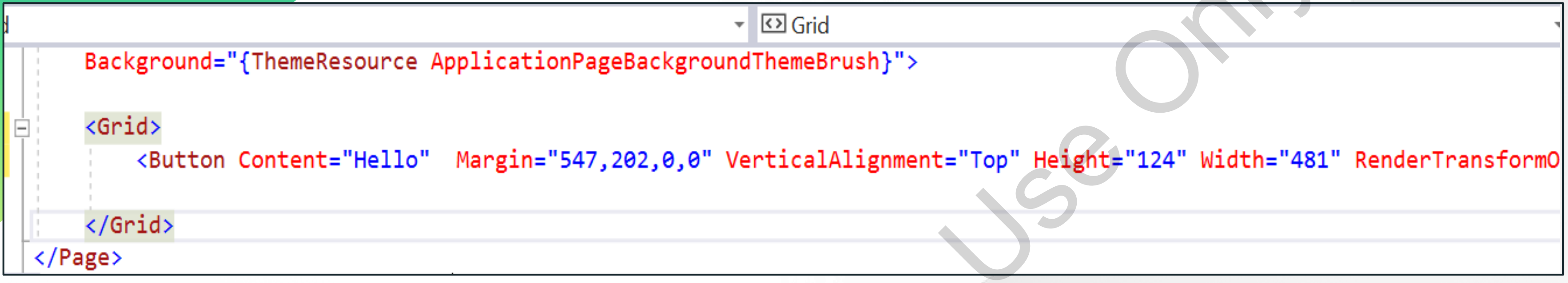


Common XAML Controls Drop-down



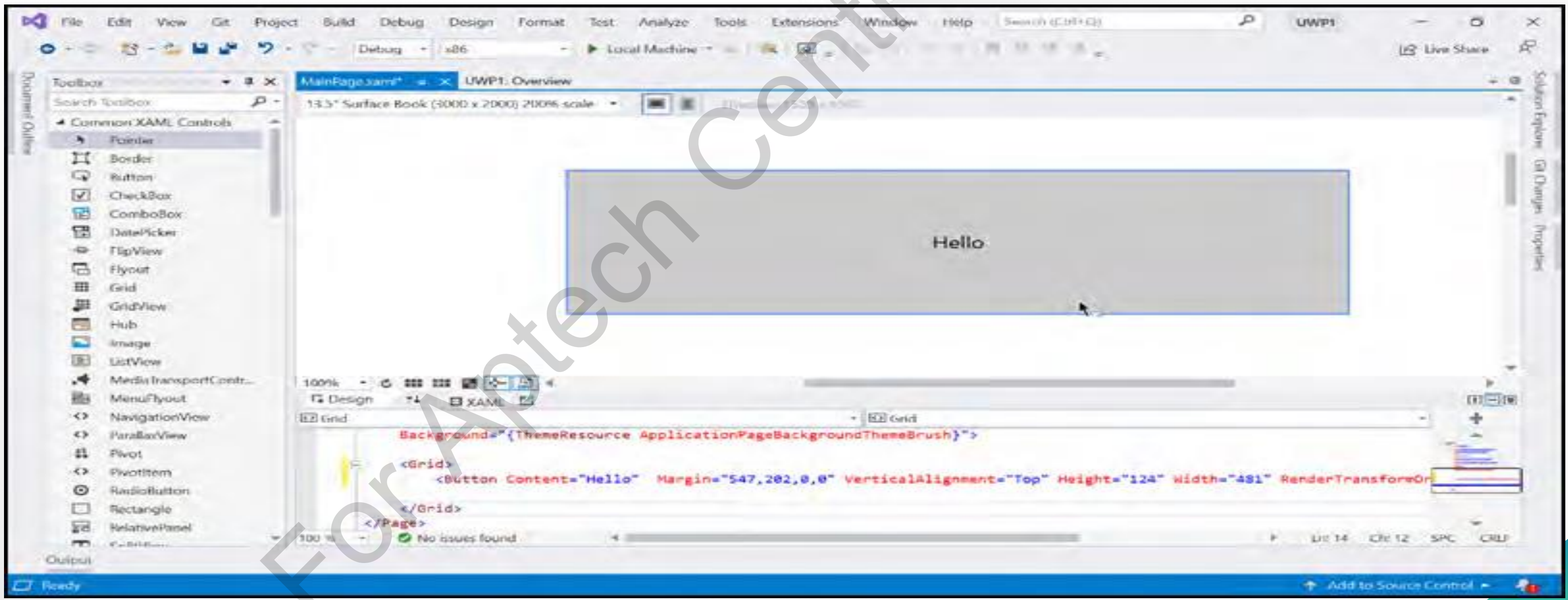
Button Icon Added on Designer Window

Create the Application 3-4



```
Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <Grid>
        <Button Content="Hello"  Margin="547,202,0,0" VerticalAlignment="Top" Height="124" Width="481" RenderTransformO

    </Grid>
</Page>
```

Code for Changing the Label of Button in XAML Editor



```
Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
    <Grid>
        <Button Content="Hello"  Margin="547,202,0,0" VerticalAlignment="Top" Height="124" Width="481" RenderTransformOr
    </Grid>
</Page>
```

Button Label Changed to Hello

Create the Application 4-4

Code Snippet:

```
private void button_Click(object sender, RoutedEventArgs
e)
 {
       button.Background = new
SolidColorBrush(Windows.UI.Colors.Blue);
 }
```

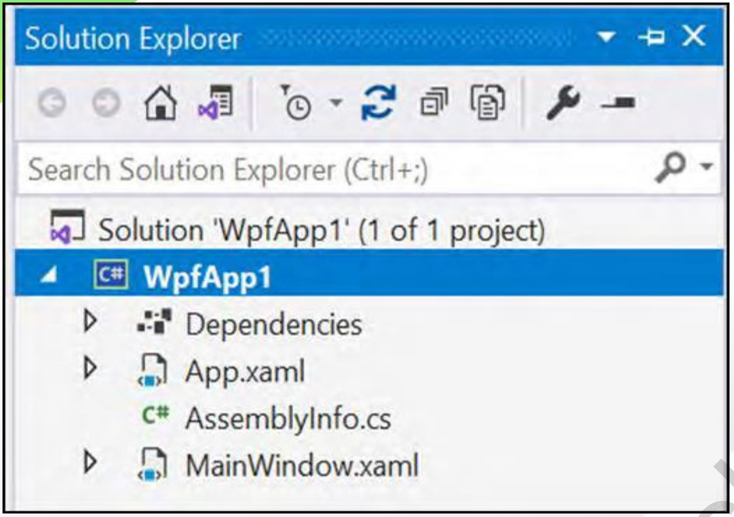
- Build and run the application. Upon click of the button, the background color of the button will change to blue.

Creating Windows Presentation Foundation Apps 1-9

- Windows Presentation Foundation (WPF) is a UI framework that is used to create Windows or desktop client applications.
- The WPF development platform supports a broad set of application development features, including an application model, resources, controls, graphics, layout, data binding, documents, and security. .NET 5.0 along with Visual Studio 2019 provides a complete platform to design and develop WPF apps.
- WPF also makes use of XAML. One can use Visual Studio 2019 and .NET framework or .NET.

Creating Windows Presentation Foundation Apps 2-9

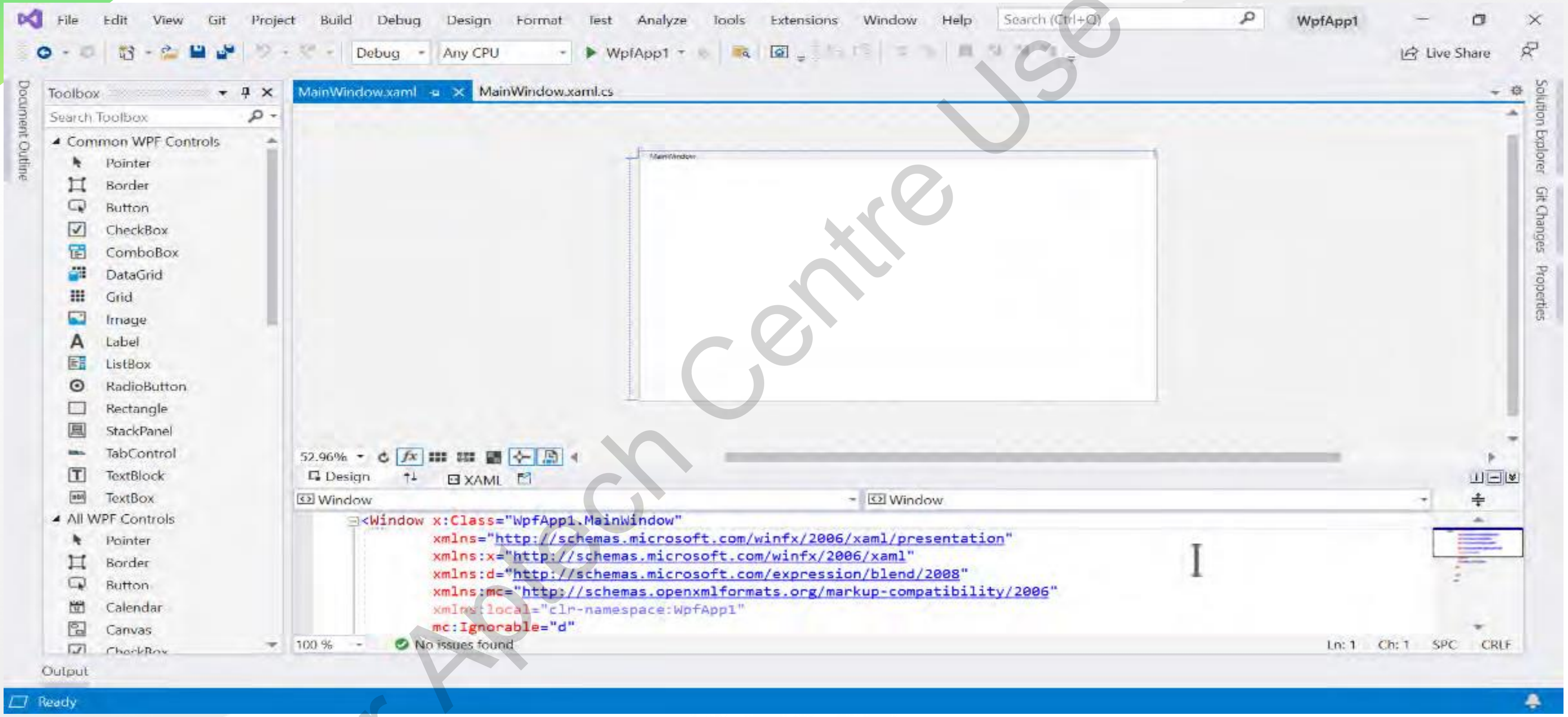
Create a Project:



WpfApp1 as New Project

- To create a project, open Visual Studio 2019, and in the start window, select Create a new project.
- Type WPF, select WPF Application framework and click Next. The Configure your new project screen will appear. Write the project name such as WPFApp1 and click Next.
- In the drop-down menu that is displayed, select the target framework as .NET 5.0 from the options available.
- Click Create to create the new project.

Creating Windows Presentation Foundation Apps 3-9



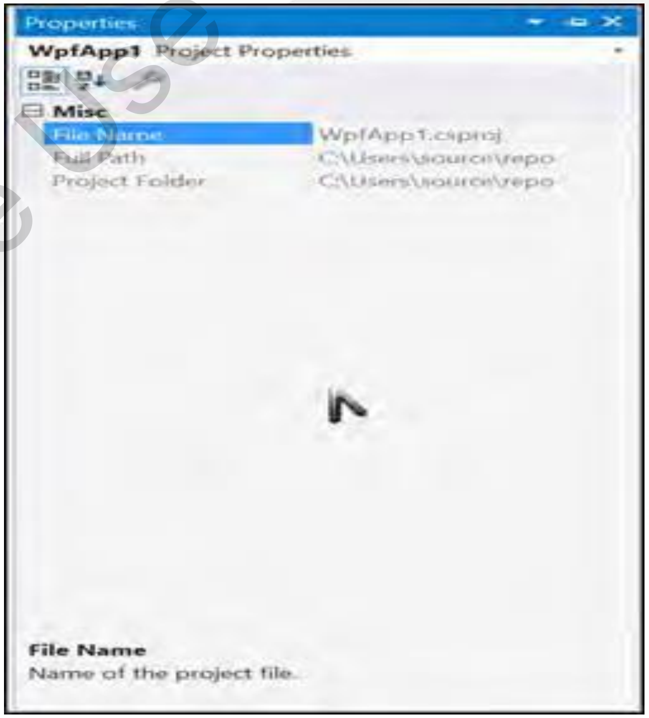
```
<Window x:Class="WpfApp1.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:WpfApp1"
        mc:Ignorable="d"
```

Views Available in WpfApp1

Creating Windows Presentation Foundation Apps 4-9

- To customize the project, you can use Properties window.

- Allows to change the options for project items, controls, and other items in the application. As of now, no controls are added yet to the project.

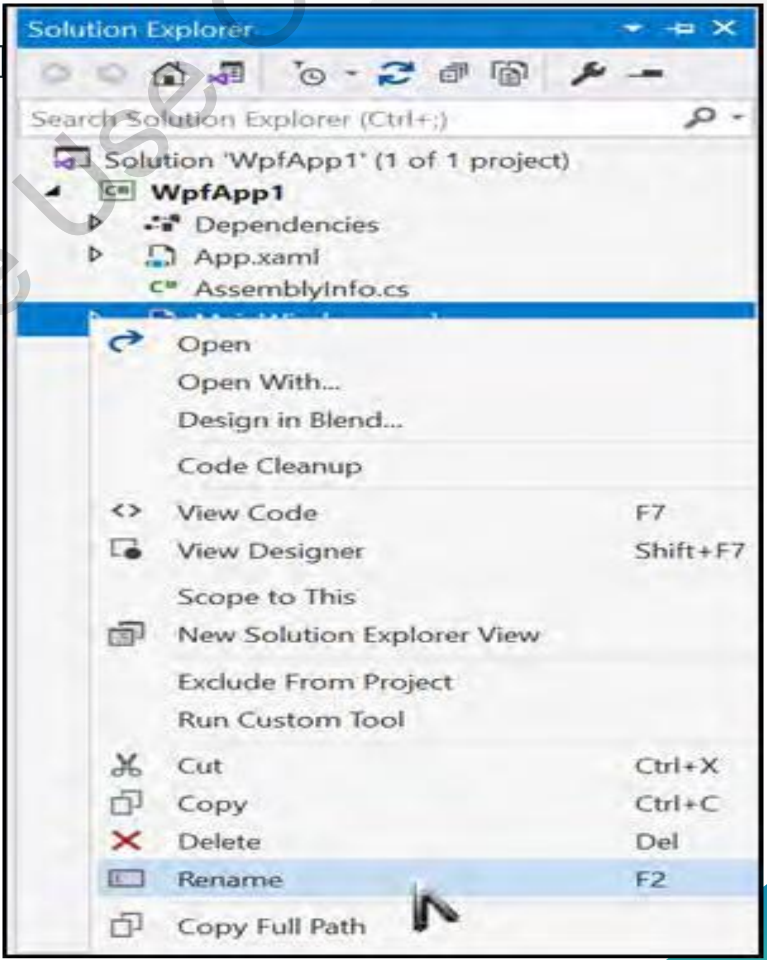

Properties
WpfApp1 Project Properties
Misc
File Name WpfApp1.csproj
Full Path C:\Users\source\repo
Project Folder C:\Users\source\repo
File Name
Name of the project file.

Properties Window

Creating Windows Presentation Foundation Apps 5-9

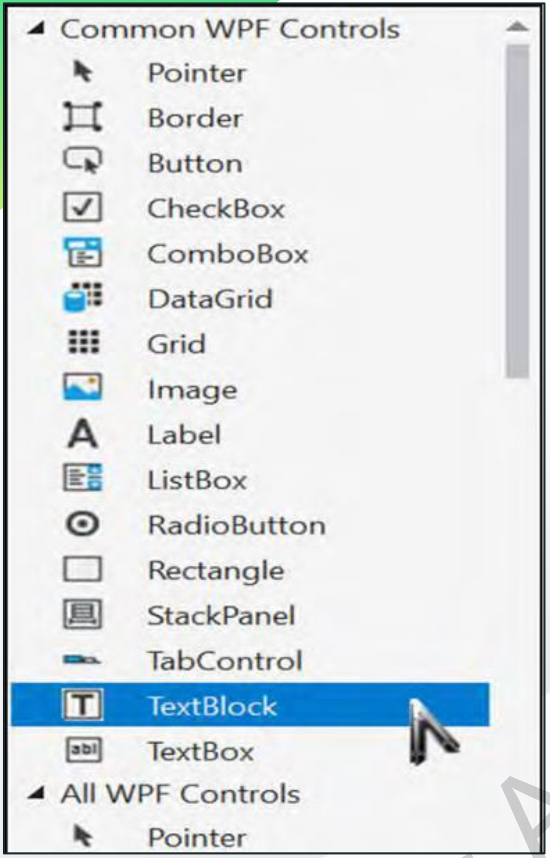
Change the name of `MainWindow.xaml`

- To rename `MainWindow.xaml`, go to Solution Explorer, right-click the file name, and select the option Rename.
- Add the name as per the requirement, such as `Greet.xaml`

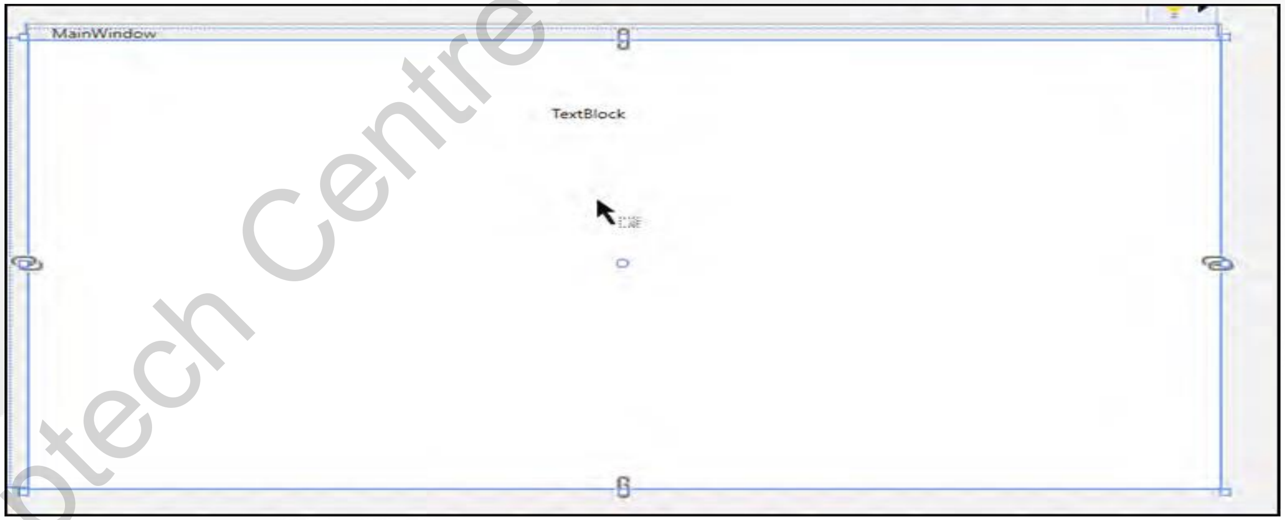


Renaming
`MainWindow.xaml`

Creating Windows Presentation Foundation Apps 6-9



Adding TextBlock Control

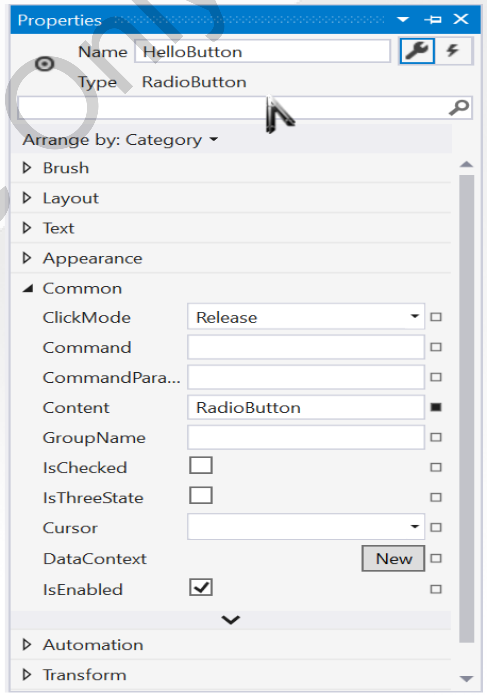


TextBlock Placed at Top of Window

Creating Windows Presentation Foundation Apps 7-9

- In this step, user will add two RadioButton controls, one for greeting with Hello and another for Goodbye. The procedure for adding the radio button is same as TextBlock control.

- Drag and drop the RadioButton controls to the surface window. In the left radio button, write HelloButton and on the right radio button, write GoodbyeButton.



Name	HelloButton
Type	RadioButton

Arrange by: Category

- Brush
- Layout
- Text
- Appearance
- Common

ClickMode	Release
Command	
CommandPara...	
Content	RadioButton
GroupName	
IsChecked	☐
IsThreeState	☐
Cursor	
DataContext	New
IsEnabled	✓

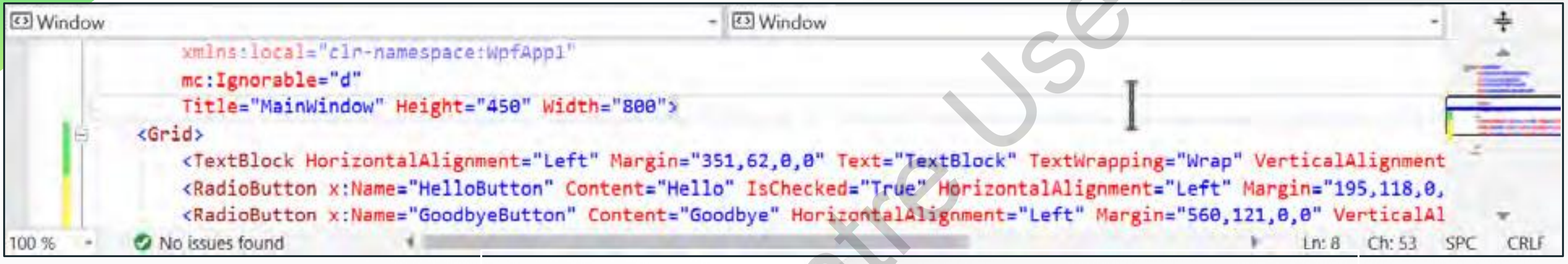
- Automation
- Transform

Adding Radio Buttons

Code Snippet 3:

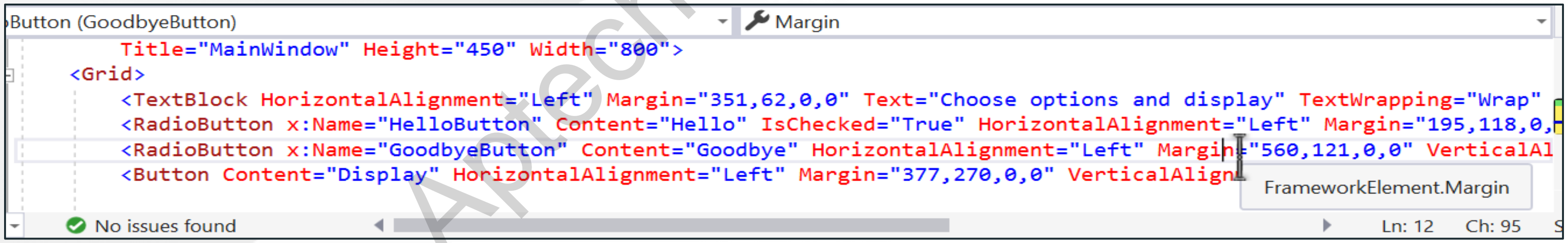
```
<Grid>
     <TextBlock HorizontalAlignment="Left" Margin="252,47,0,0" TextWrapping="Wrap"
Text="Select a message option and then choose the Display button."
VerticalAlignment="Top"/>
     <RadioButton x:Name="HelloButton" Content="Hello" HorizontalAlignment="Left"
Margin="297,161,0,0" VerticalAlignment="Top"/>
     <RadioButton x:Name="GoodbyeButton" Content="Goodbye" HorizontalAlignment="Left"
Margin="488,161,0,0" VerticalAlignment="Top"/>
</Grid>
```

Creating Windows Presentation Foundation Apps 8-9



```
Window                                        Window
        xmlns:local="clr-namespace:WpfApp1"
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="800">
    <Grid>
        <TextBlock HorizontalAlignment="Left" Margin="351,62,0,0" Text="TextBlock" TextWrapping="Wrap" VerticalAlignment
        <RadioButton x:Name="HelloButton" Content="Hello" IsChecked="True" HorizontalAlignment="Left" Margin="195,118,0,
        <RadioButton x:Name="GoodbyeButton" Content="Goodbye" HorizontalAlignment="Left" Margin="560,121,0,0" VerticalAl
100 %   No issues found                                                                    Ln: 8   Ch: 53   SPC   CRLF
```

Adding `IsChecked="True"` in Code

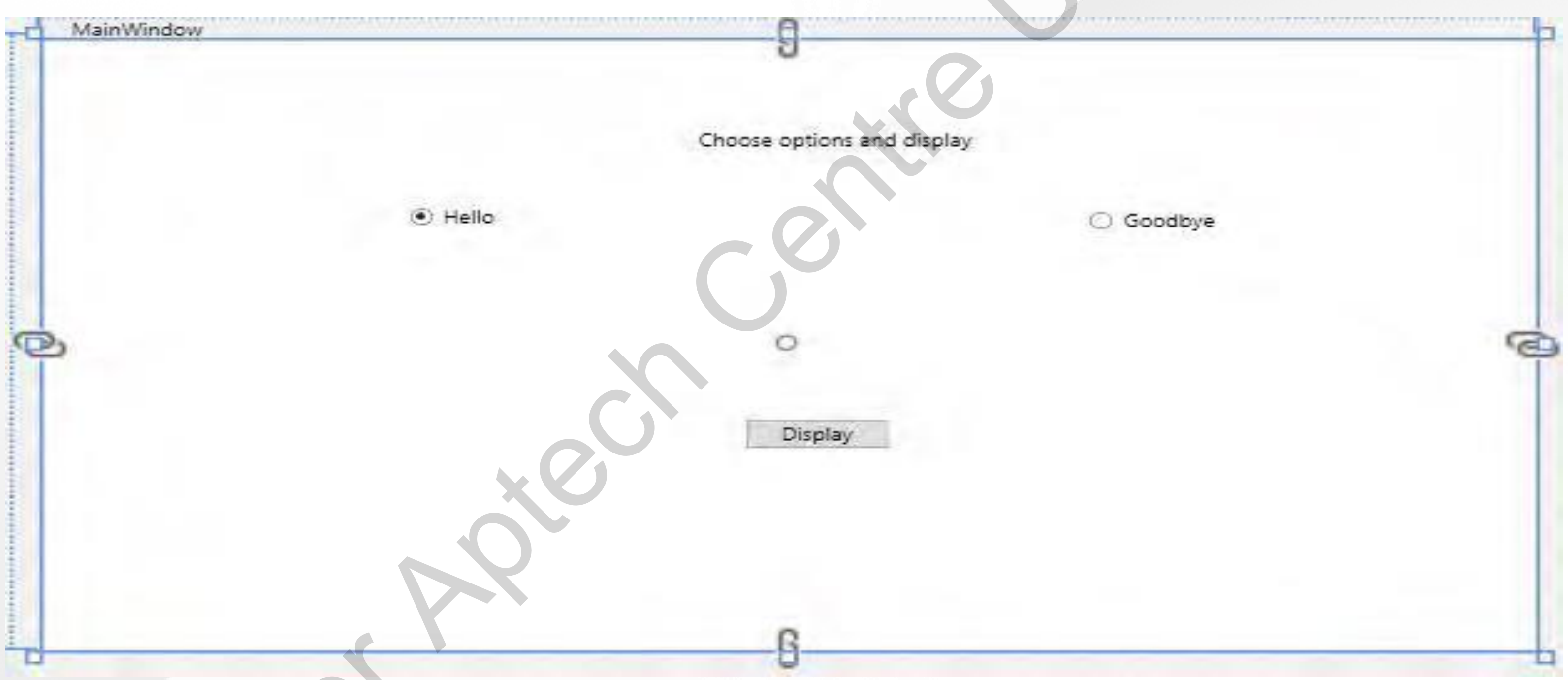


```
Button (GoodbyeButton)                              Margin
        Title="MainWindow" Height="450" Width="800">
    <Grid>
        <TextBlock HorizontalAlignment="Left" Margin="351,62,0,0" Text="Choose options and display" TextWrapping="Wrap"
        <RadioButton x:Name="HelloButton" Content="Hello" IsChecked="True" HorizontalAlignment="Left" Margin="195,118,0,
        <RadioButton x:Name="GoodbyeButton" Content="Goodbye" HorizontalAlignment="Left" Margin="560,121,0,0" VerticalAl
        <Button Content="Display" HorizontalAlignment="Left" Margin="377,270,0,0" VerticalAlign
                                                                       FrameworkElement.Margin
    No issues found                                                                    Ln: 12   Ch: 95
```

Editing the TextBlock

Creating Windows Presentation Foundation Apps 9-9

Output:

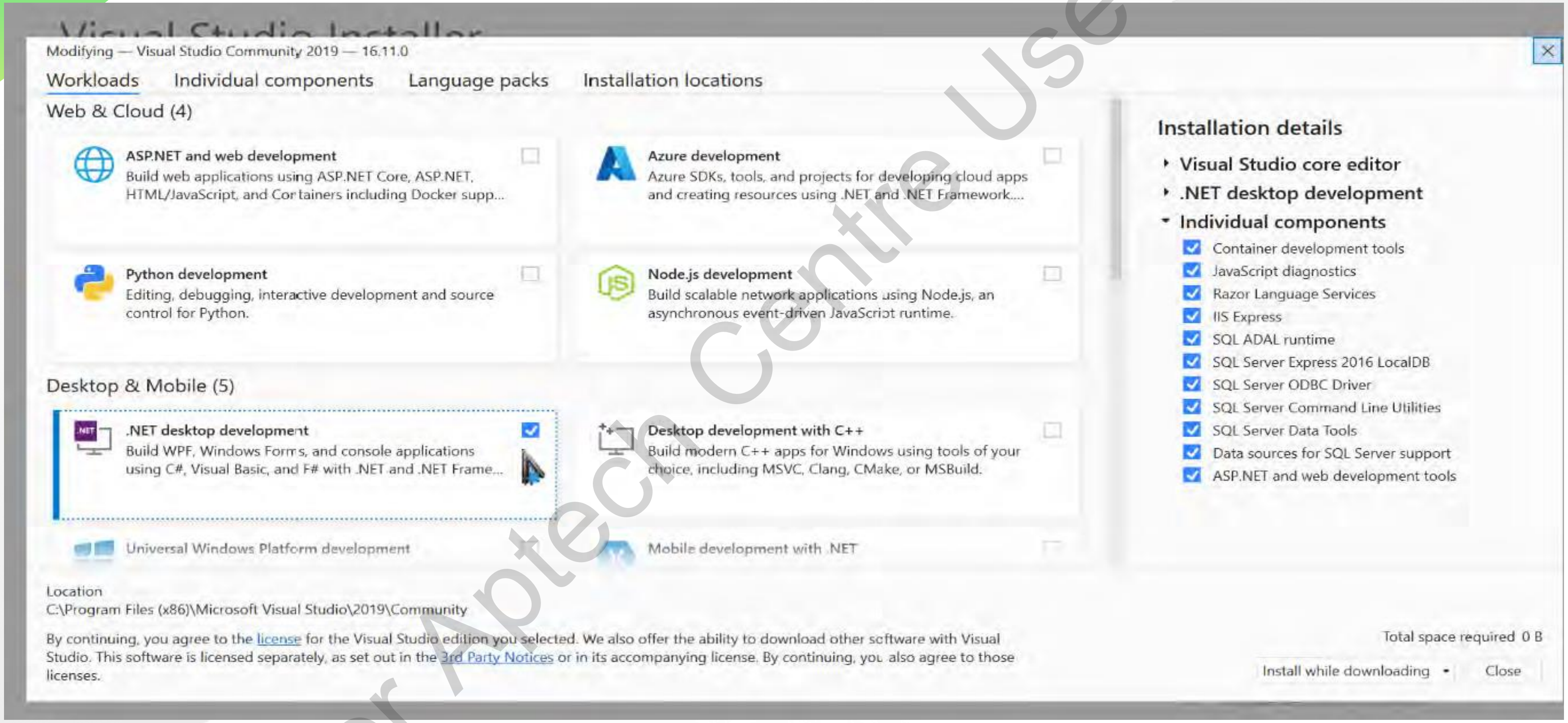


Creating Windows Forms Apps 1-6

- Windows Forms or WinForms is the first UI framework that was created for building Windows desktop apps. It was introduced together with .NET 1.0 in 2002.
- WinForms offers better than WPF or UWP, which is ease of development.
- WinForms is a great choice for making a quick prototype of an application.
- The learning curve is less steep than it is for WPF or UWP. Users will not have to struggle with complex syntax of XAML – there is more of drag-and-drop development here as compared to WPF or UWP.
- Microsoft Visual Studio 2019 along with .NET 5.0 framework supports WinForms development

Creating Windows Forms Apps 2-6

Creating a Project


Modifying — Visual Studio Community 2019 — 16.11.0

Workloads | Individual components | Language packs | Installation locations

Web & Cloud (4)

ASP.NET and web development ☐
Build web applications using ASP.NET Core, ASP.NET, HTML/JavaScript, and Containers including Docker supp...

Azure development ☐
Azure SDKs, tools, and projects for developing cloud apps and creating resources using .NET and .NET Framework....

Python development ☐
Editing, debugging, interactive development and source control for Python.

Node.js development ☐
Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Desktop & Mobile (5)

.NET desktop development ✓
Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET and .NET Frame...

Desktop development with C++ ☐
Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.

Universal Windows Platform development

Mobile development with .NET

Installation details
- Visual Studio core editor
- .NET desktop development
- Individual components
  - ✓ Container development tools
  - ✓ JavaScript diagnostics
  - ✓ Razor Language Services
  - ✓ IIS Express
  - ✓ SQL ADAL runtime
  - ✓ SQL Server Express 2016 LocalDB
  - ✓ SQL Server ODBC Driver
  - ✓ SQL Server Command Line Utilities
  - ✓ SQL Server Data Tools
  - ✓ Data sources for SQL Server support
  - ✓ ASP.NET and web development tools

Location
C:\Program Files (x86)\Microsoft Visual Studio\2019\Community

By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

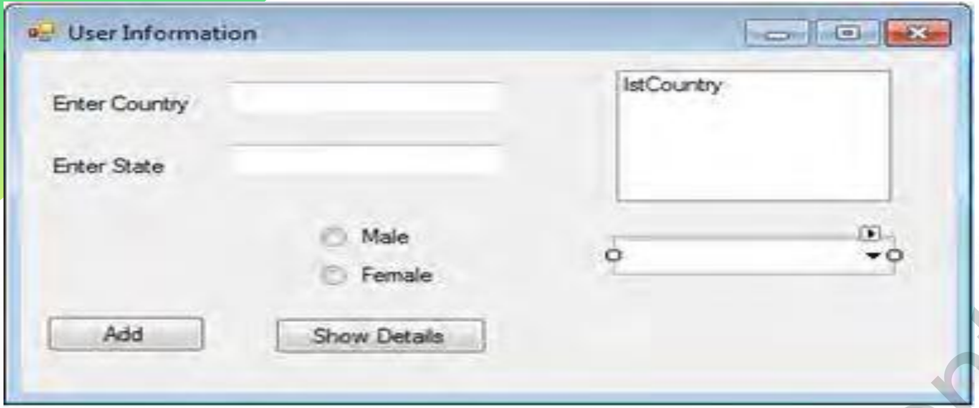
Total space required 0 B

Install while downloading | Close

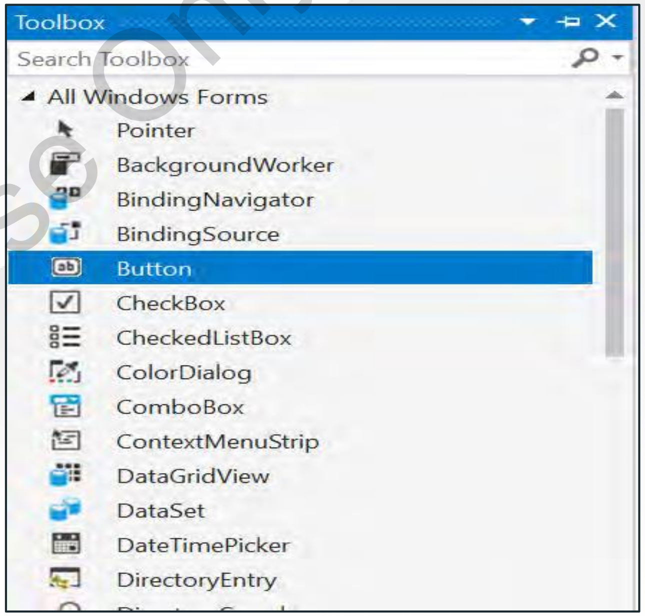
Visual Studio Installer

Creating Windows Forms Apps 3-6

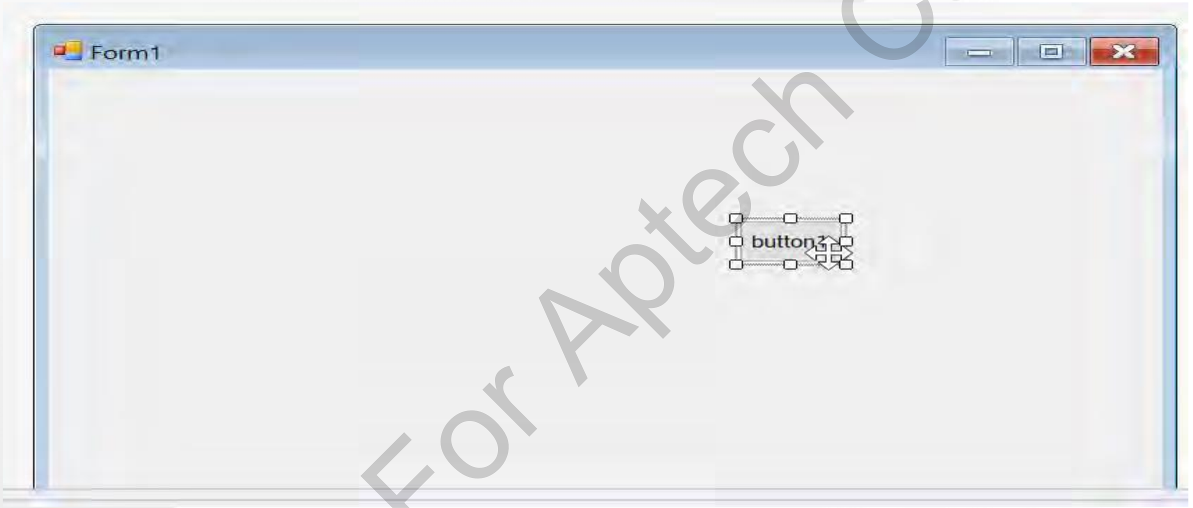
Adding Controls



Sample Form



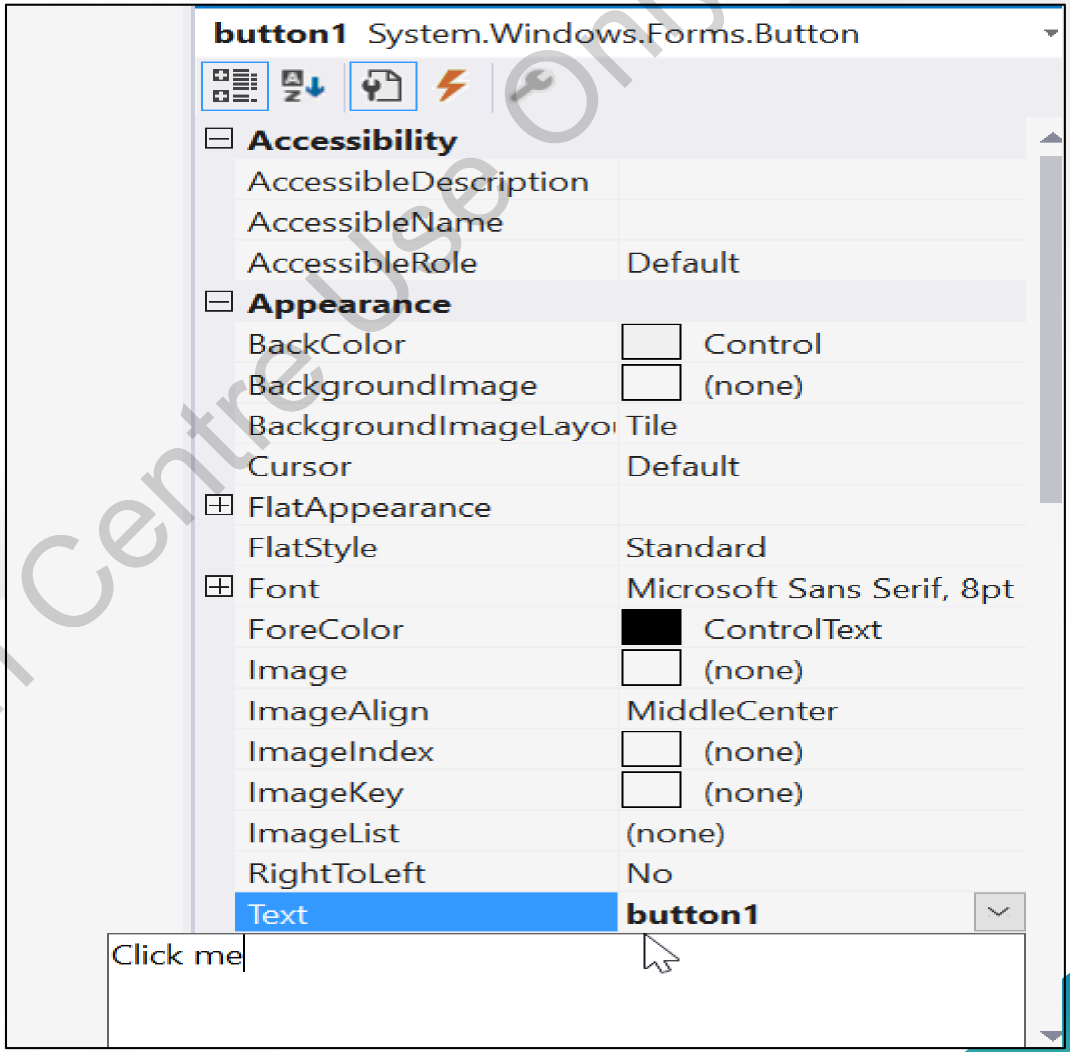
Selecting Button from Toolbox



Button Added in Form Window

Creating Windows Forms Apps 4-6

- To modify the text of the Button, go to the Properties window, locate Text option, and change the text from button1 to the desired name `Click Me.`



button1 System.Windows.Forms.Button

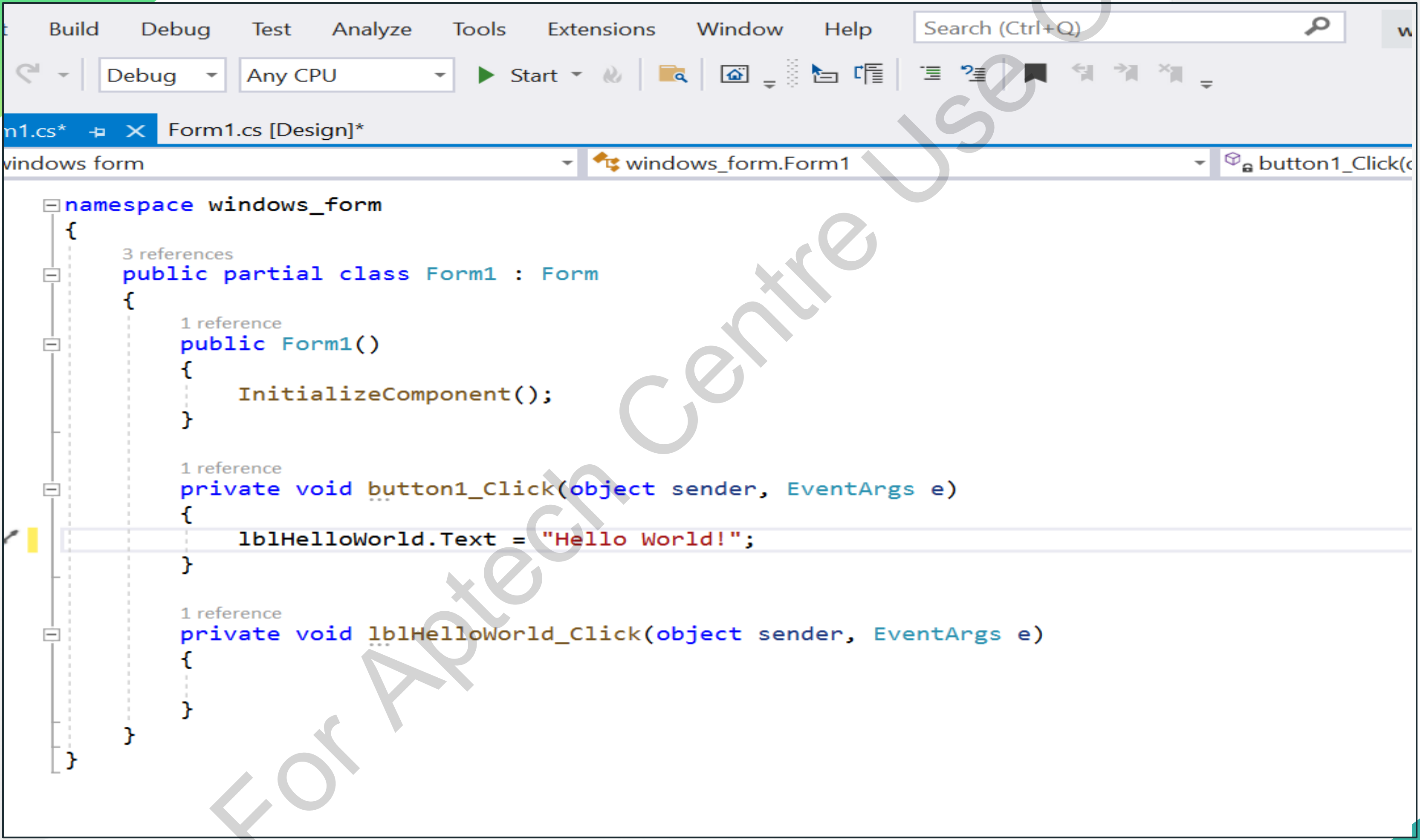
Accessibility	
AccessibleDescription	
AccessibleName	
AccessibleRole	Default
Appearance	
BackColor	Control
BackgroundImage	(none)
BackgroundImageLayout	Tile
Cursor	Default
FlatAppearance	
FlatStyle	Standard
Font	Microsoft Sans Serif, 8pt
ForeColor	ControlText
Image	(none)
ImageAlign	MiddleCenter
ImageIndex	(none)
ImageKey	(none)
ImageList	(none)
RightToLeft	No
Text	**button1**

Click me

Modifying the Button Text

Creating Windows Forms Apps 5-6

Adding Code to the Form

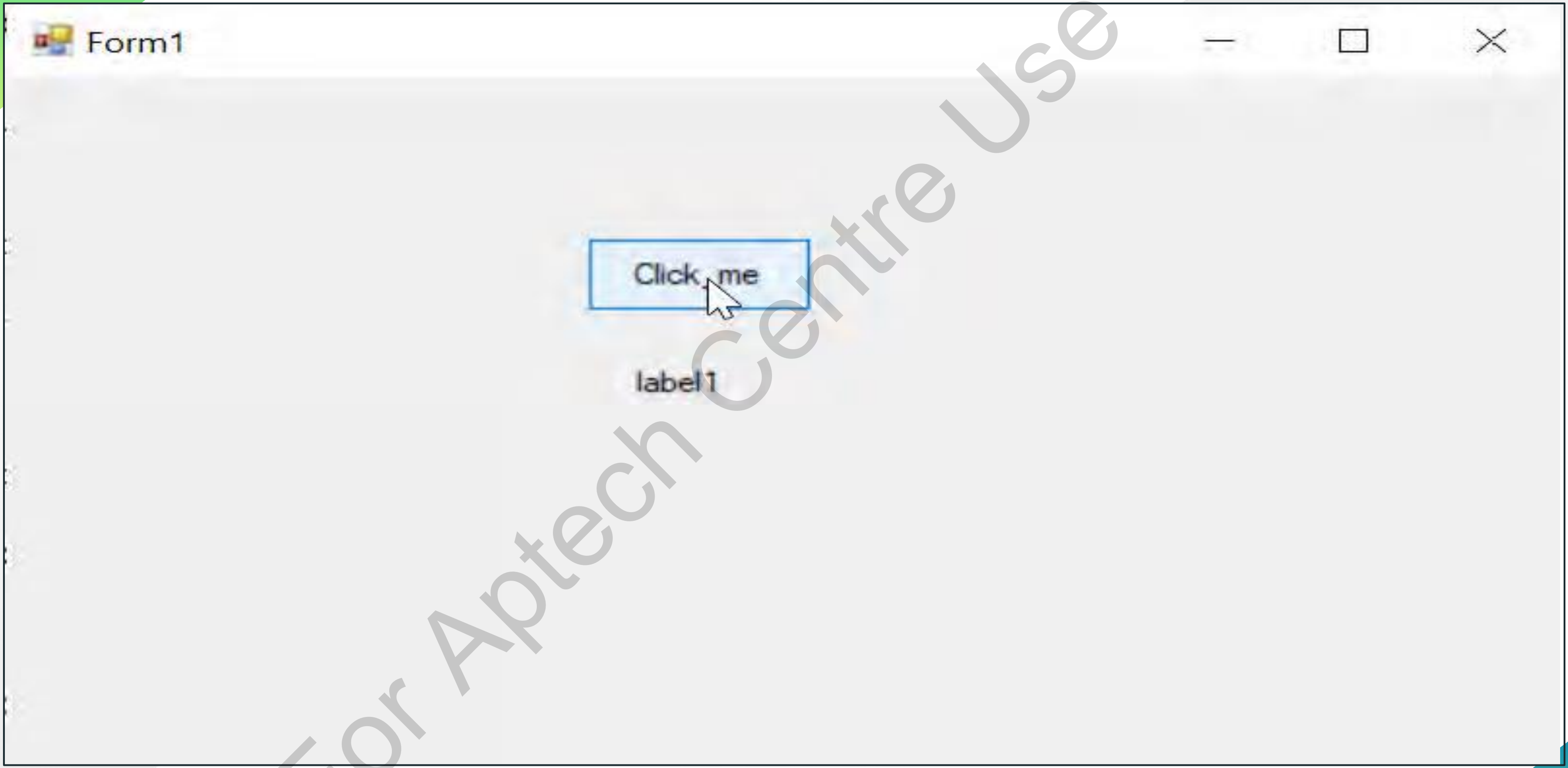


```
namespace windows_form
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            lblHelloWorld.Text = "Hello World!";
        }

        private void lblHelloWorld_Click(object sender, EventArgs e)
        {

        }
    }
}
```

Adding Code to the Form

Creating Windows Forms Apps 6-6

Running the Form



Summary

- C# 9.0 along with .NET 5.0 can be used to create Web apps and desktop apps.
- ASP.NET and ASP.NET Core are Web technologies offered by Microsoft for Web development with C#.
- Desktop apps can be created using UI frameworks such as WPF, UWP, and WinForms.
- UWP apps make use of WinUI, XAML, HTML, or DirectX for designing UI.
- WPF designer has two views, namely, designer view and XAML view. The XAML view is used to write and edit XAML markup.
- A Form in a Windows Forms application represents a window or dialog box that is part of makes up an application's UI and is used by the user to add buttons, labels, check box, and so on.