Programming Languages

C#, WinForms and WPF

Objective

Today's primary objective:

- You are introduced to C#
- You have created a C# Winform project
- You see the files generated in C# projects, and how C# divides the files between interface and code

Java & C# Similarities

- Both are “pure” OO languages
- Both use garbage collection
- All objects are references in both
- Both are type-safe
- Both only allow single inheritance but many interfaces
- Both have built-in thread and synchronization support
- Both have formal exception handling
- Both have built-in Unicode support

https://msdn.microsoft.com/en-us/library/ms836794.aspx

Java & C# Differences

Java will run on any operating system

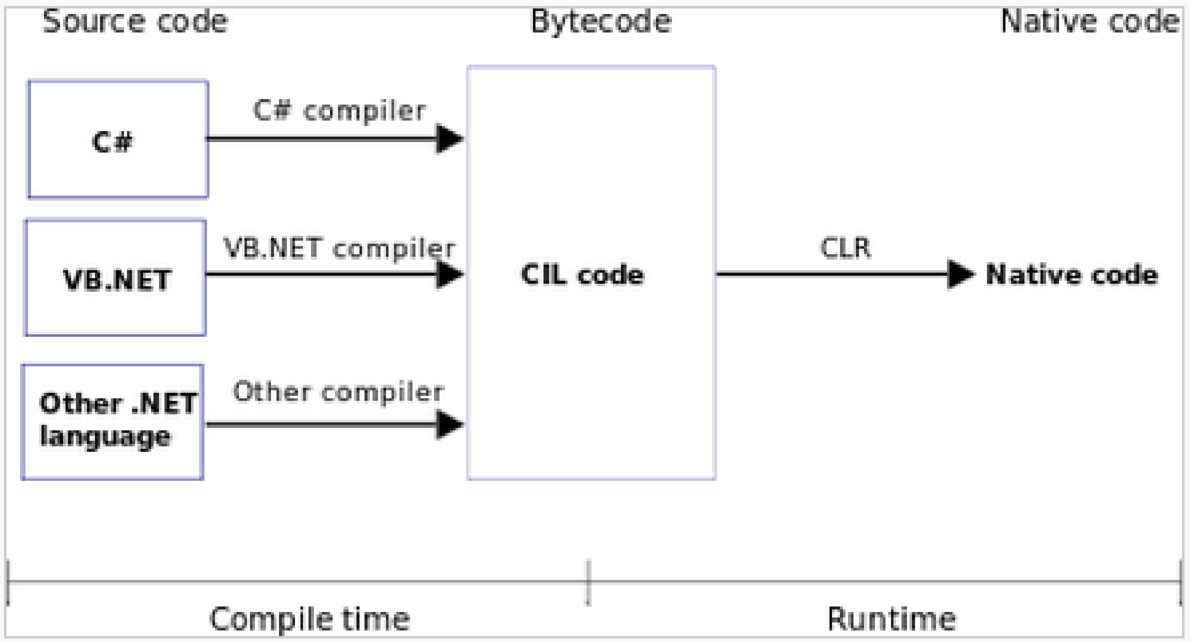
C# is also compiled to an intermediate language but their Intermediate Language (MSIL) is only supported by a few operating systems

Dynamic and Just-In-Time Compilation (Chapter 1)

- Java -------> byte code   Send over Internet to run on any
  complied   platform
                                    (Could use just-in-time compiler
                                    byte code ---> machine code)

- C# ---------> CIL   Send over Internet to run on certain
  compiled   platform
                                    (Uses just-in-time compiler
                                    CIL ----> machine code)

Microsoft's Common Language Runtime (CLR)



The CLR converts CIL (Common Intermediate Language) to native code.

Java & C# Differences

C# is more complex than Java

- Java built to keep a developer from shooting himself
- C# built to give the developer a gun but with the safety latch on

- C++ you've got the gun, be careful
- C# is safe, but to be compatible with C++, has an unsafe mode

Java & C# Differences

Keyword Comparison

Java Keyword	C# Keyword
const	readonly
boolean	bool
package	namespace
private	internal
instanceof	is
import	using

Java & C# Differences

Keyword Comparison

Java Keyword	C# Keyword
super	base
synchronized	lock
final	sealed
extends	Use: public class : baseClass
implements	Use: Public class : Interface<class>

Building Applications in C#

Two ways to build GUI applications in C#:

- WinForms
- WPF

WinForms versus WPF

WinForms	WPF
Graphical API that provides access to native Microsoft Windows interface elements	Graphical subsystem that renders UIs in Windows based applications
Event driven application supported by the Microsoft .NET Framework	Markup language defines UI elements and relationships with other UI elements

Advantages of WPF

- Newer and thereby more in tune with current standards
- More flexible, so you can do more things without having to write or buy new controls
- XAML makes it easy to create and edit your GUI, and allows the work to be split between a designer (XAML) and a programmer (C#, VB.NET etc.)
- Databinding, which allows you to get a more clean separation of data and layout
- Uses hardware acceleration for drawing the GUI, for better performance
- It allows you to make user interfaces for both Windows applications and web application (Silverlight/XBAP)

Advantages of WinForms

- Older and thereby more tried and tested
- More online resources, developer communities, examples ,etc. are available
- More 3rd party controls available
- The designer in Visual Studio is still better for WinForms than for WPF, where you will have to do more of the work yourself with WPF
- WPF will not run on windows 2000 or lower
- WPF requires .NET Framework 3.0

WinForms

Create Windows App using WinForms

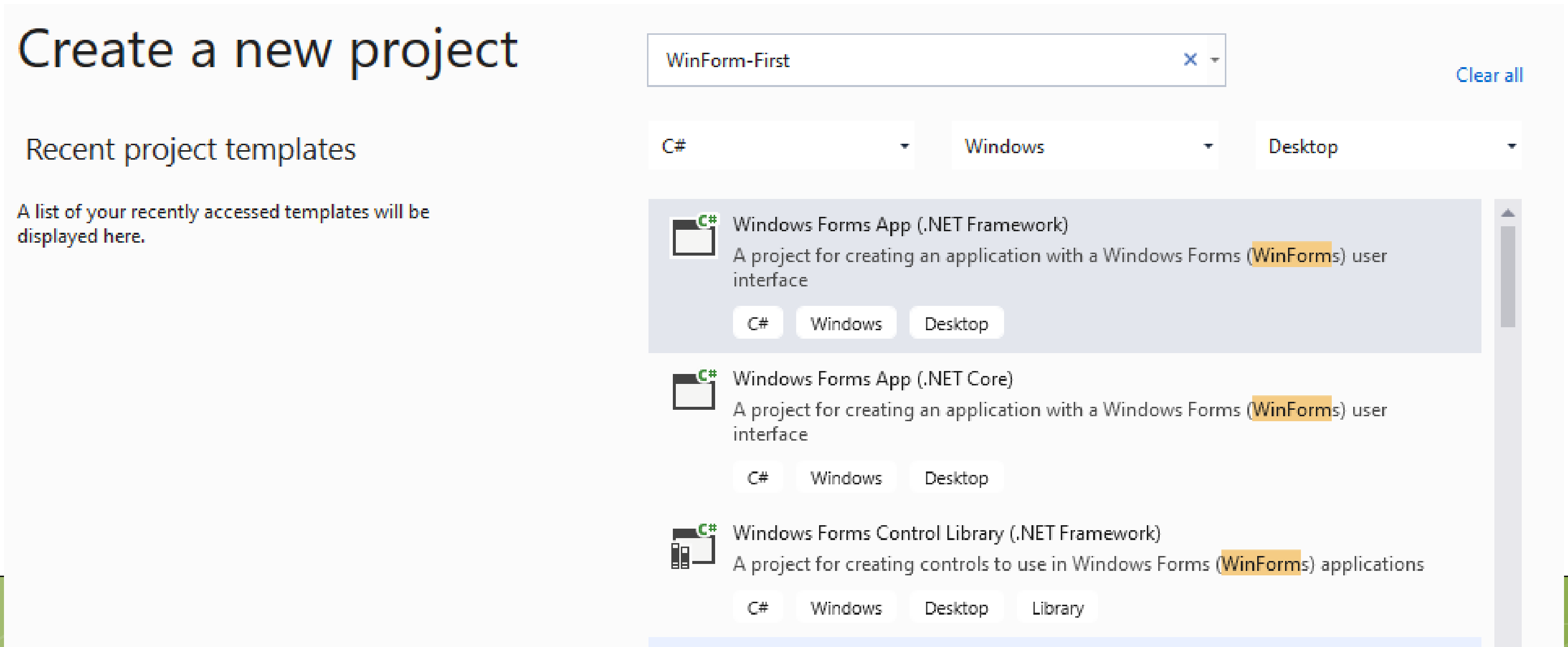
Create a Windows Forms Application:

- Open Visual Studio
- Create a new "Windows Form App (.Net Framework)

    Language – C#

    Platform – Windows

    Type - Desktop



Create a new project

WinForm-First

Clear all

Recent project templates

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

C# | Windows | Desktop

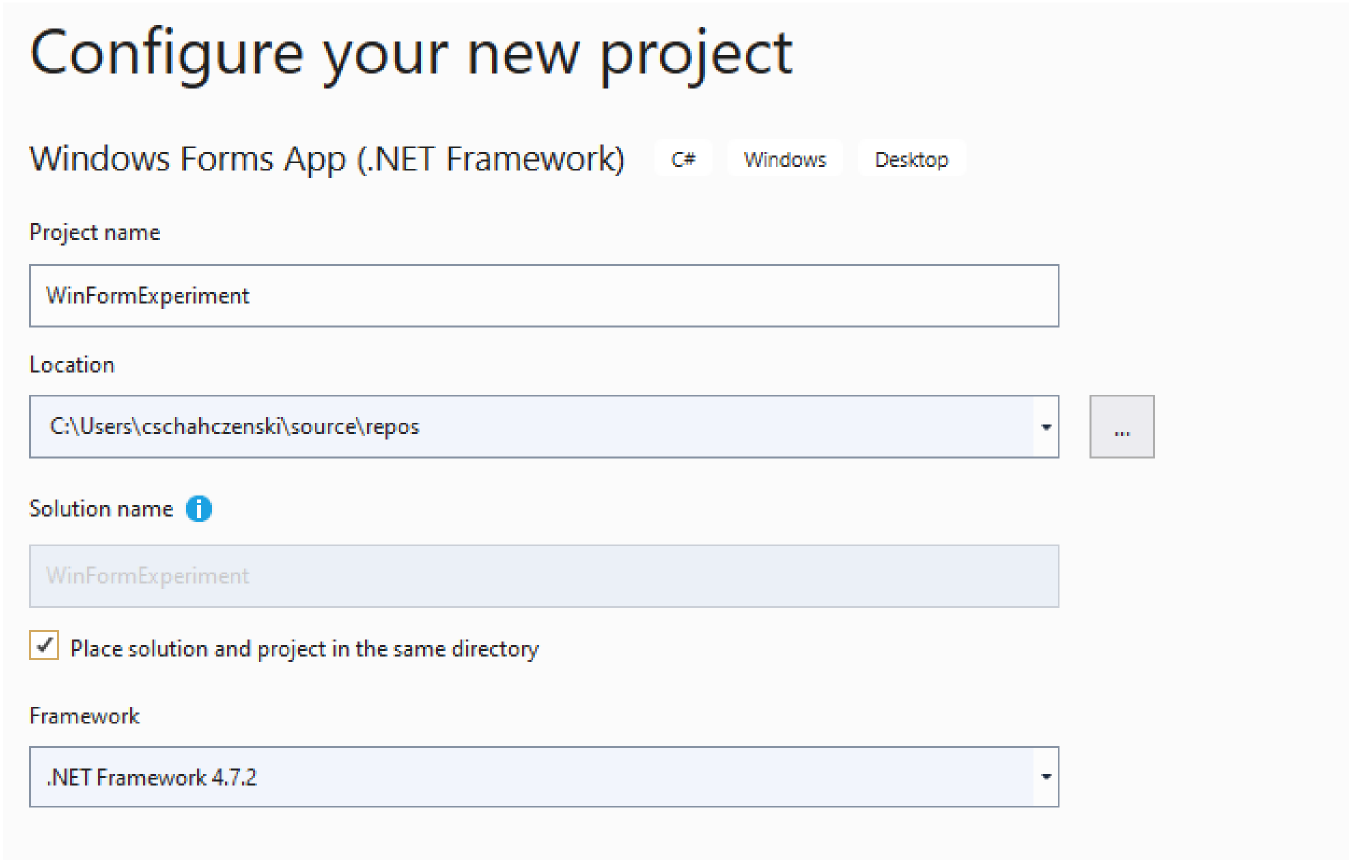
Windows Forms App (.NET Framework)
A project for creating an application with a Windows Forms (WinForms) user interface
C# Windows Desktop

Windows Forms App (.NET Core)
A project for creating an application with a Windows Forms (WinForms) user interface
C# Windows Desktop

Windows Forms Control Library (.NET Framework)
A project for creating controls to use in Windows Forms (WinForms) applications
C# Windows Desktop Library

Winforms

Give it a name and location . Check the box to place the solution and project in the same directory.



Configure your new project

Windows Forms App (.NET Framework) C# Windows Desktop

Project name

WinFormExperiment

Location

C:\Users\cschahczenski\source\repos

Solution name

WinFormExperiment

[✓] Place solution and project in the same directory

Framework

.NET Framework 4.7.2

Winforms

Locate the directory and see what was created.

Directory Structure

The directory structure will be:

```
SolutionName
        SolutionName
                bin
                        debug
                                SolutionName.vshost.exe
                                SolutionName.vshost.exe.manifest
                                …
                obj
                        debug   …
                properties
                                …
                        form1.cs
                        form1.Designer.cs
                        SolutionName.csproj
                        Program.cs
        SolutionName.sln – what is clicked to open the project
```

Solution Explorer

Solution Explorer (typically a tab in the right panel of the Visual Studio display) enables manipulating files.

Change the name of the file for the form. To FirstWindow. (Let this change permeate throughout the project. The title on the form will not change.)

Go back to the files and see what has changed.

Properties

Properties (typically below Solution Explorer, see drop-down arrow to customize display) to change characteristics of display

(View/Properties Window to open)

Select form and see it's properties

Change Text property to change the name on the form.

Toolbox

Toolbox Properties (typically tab on left) to add items to display

Drag Label and two buttons onto form .

label text “Which direction?”

button – “Up”

button – “Down”

Partial Classes

Class is split into what code to change, and what code not to change

FirstWindow.Designer.cs

- don’t modify directly

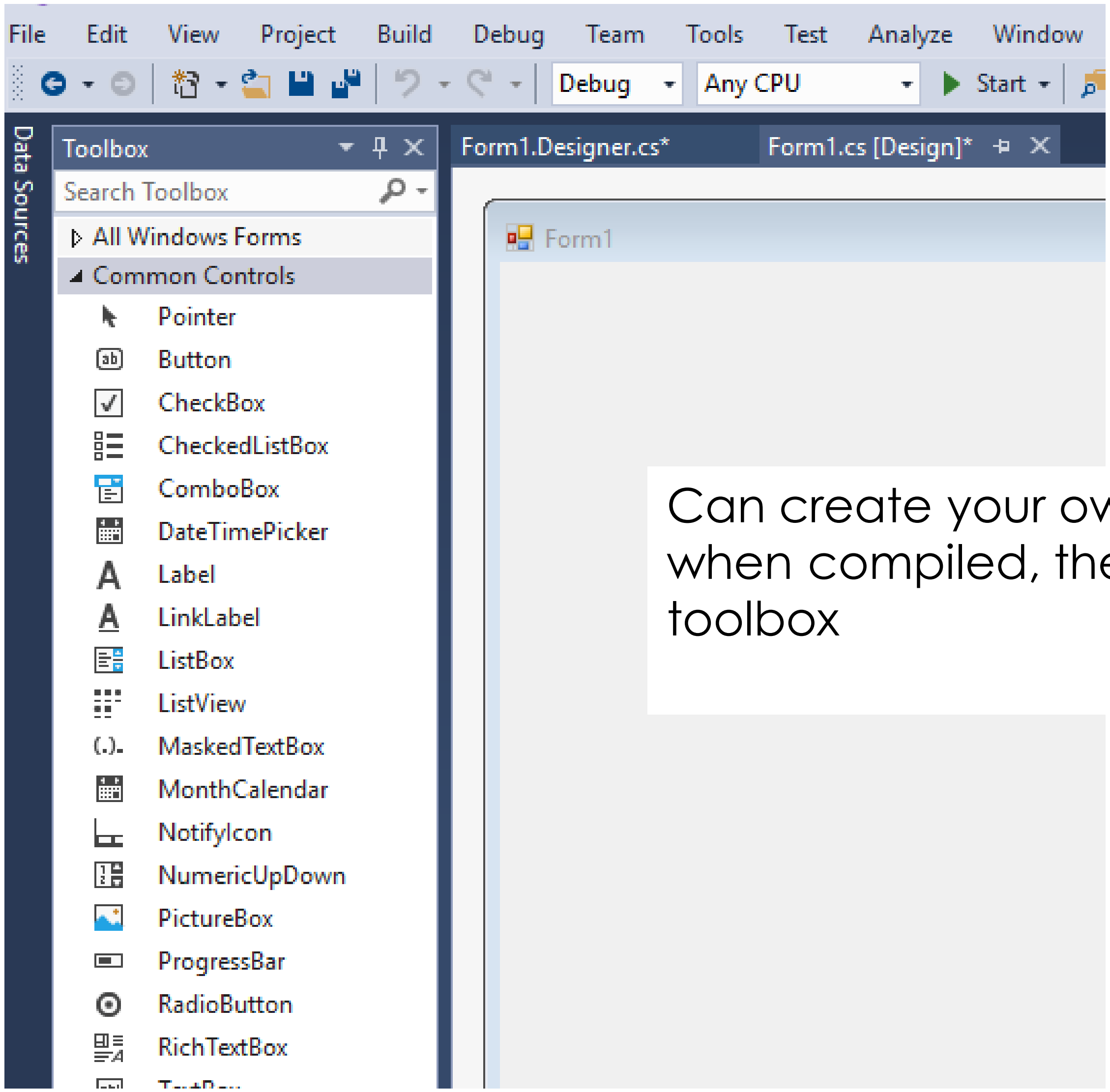
FirstWindow.cs

- shows in the designer view, edit properties here

FirstWindows.cs

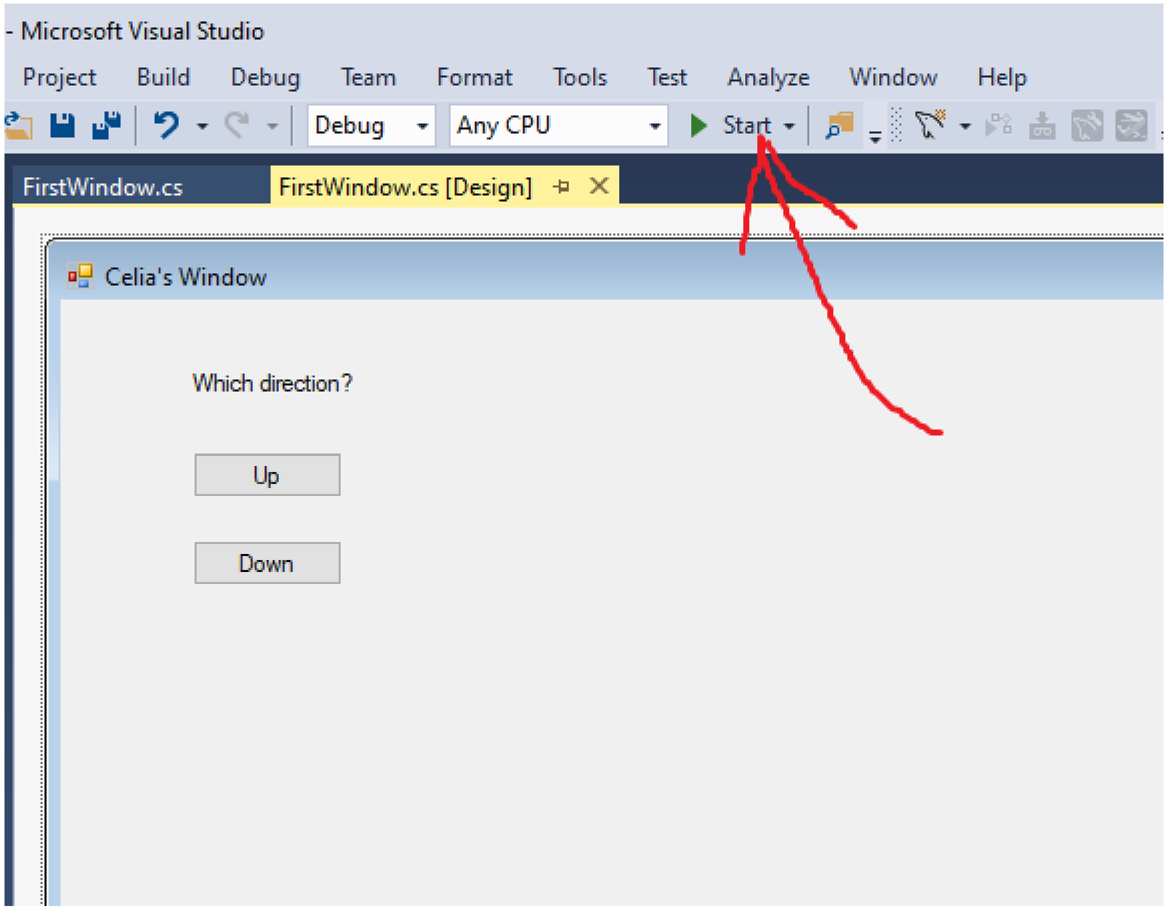
- code view, edit code here

WinForms



Can create your own widgets and when compiled, they appear in toolbox

Execute Code



WPF

Windows Presentation Foundation

Windows Presentation Foundation, WPF

- Separates the UI from procedural code by introducing data bindings
- Resembles similar XML-oriented object models
- Previously known as “Avalon”
- Initially released with .NET Framework 3.0
- Microsoft has not stated anything about WPF replacing WinForms

Extensible Application Markup Language, XAML

- Declarative XML based language (Microsoft)
- Enables a declarative definition of UI rather than procedural code
- Used to define UI elements and their data bindings
- Files can be created and editing with visual design tools or with a text editor
- No new objects - anything created or implemented in XAML can be expressed using more traditional .NET language

Create Windows App using WPF

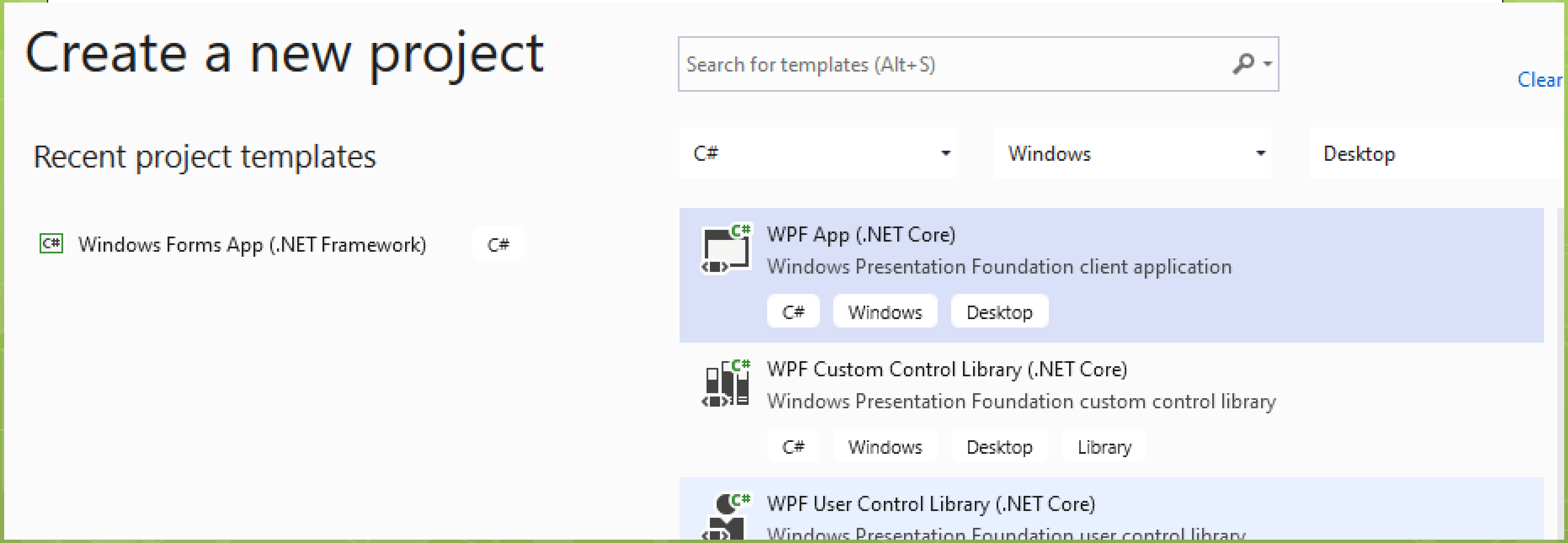
Create a Windows Forms Application:

- Open Visual Studio
- Create a new "WPF App (.Net Core)"   (Framework is older)

  Language – C#

  Platform – Windows

  Type - Desktop



Create a new project

Search for templates (Alt+S)

Clear

Recent project templates

Windows Forms App (.NET Framework) C#

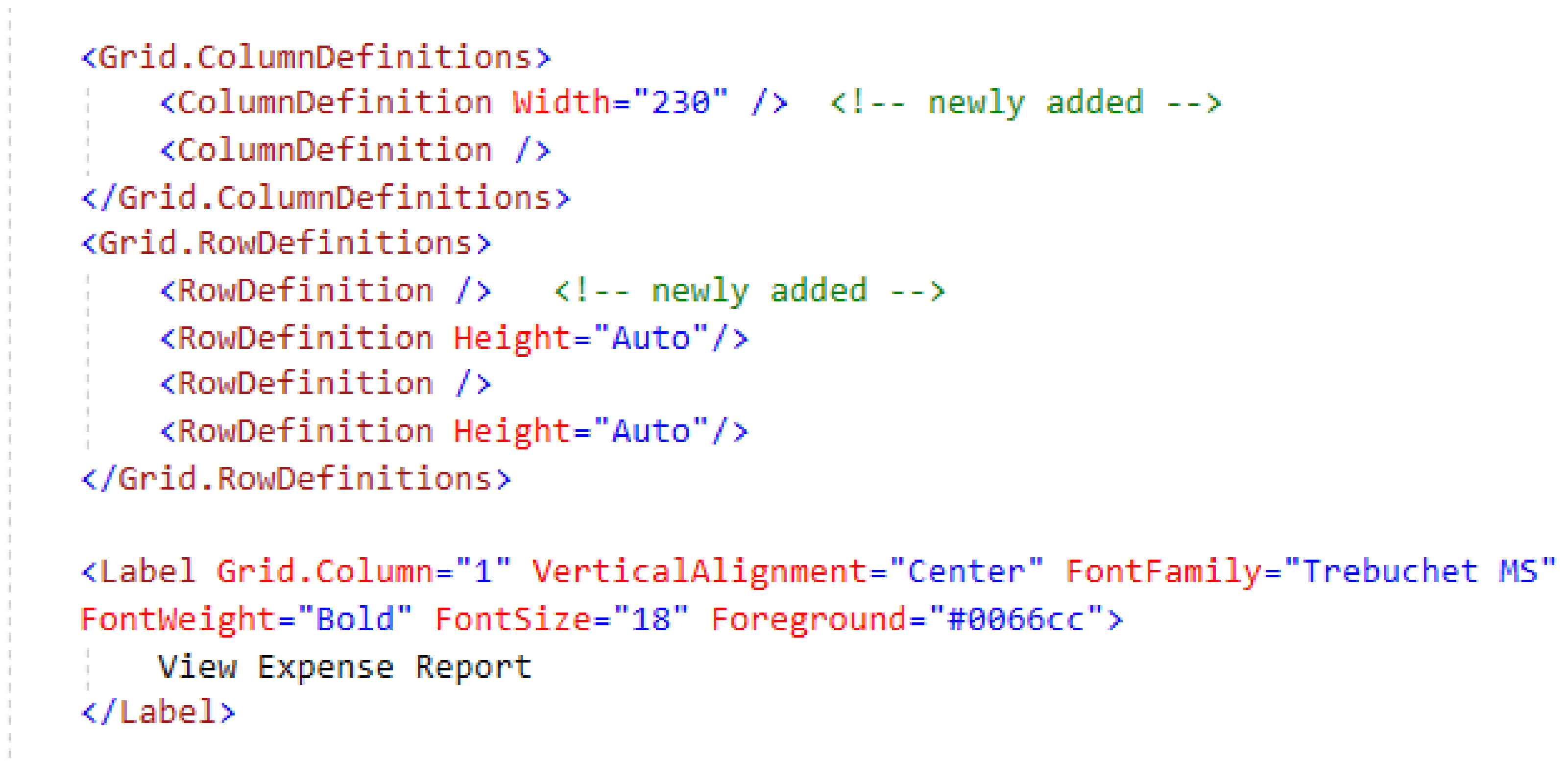
C# | Windows | Desktop

WPF App (.NET Core)
Windows Presentation Foundation client application
C# Windows Desktop

WPF Custom Control Library (.NET Core)
Windows Presentation Foundation custom control library
C# Windows Desktop Library

WPF User Control Library (.NET Core)
Windows Presentation Foundation user control library

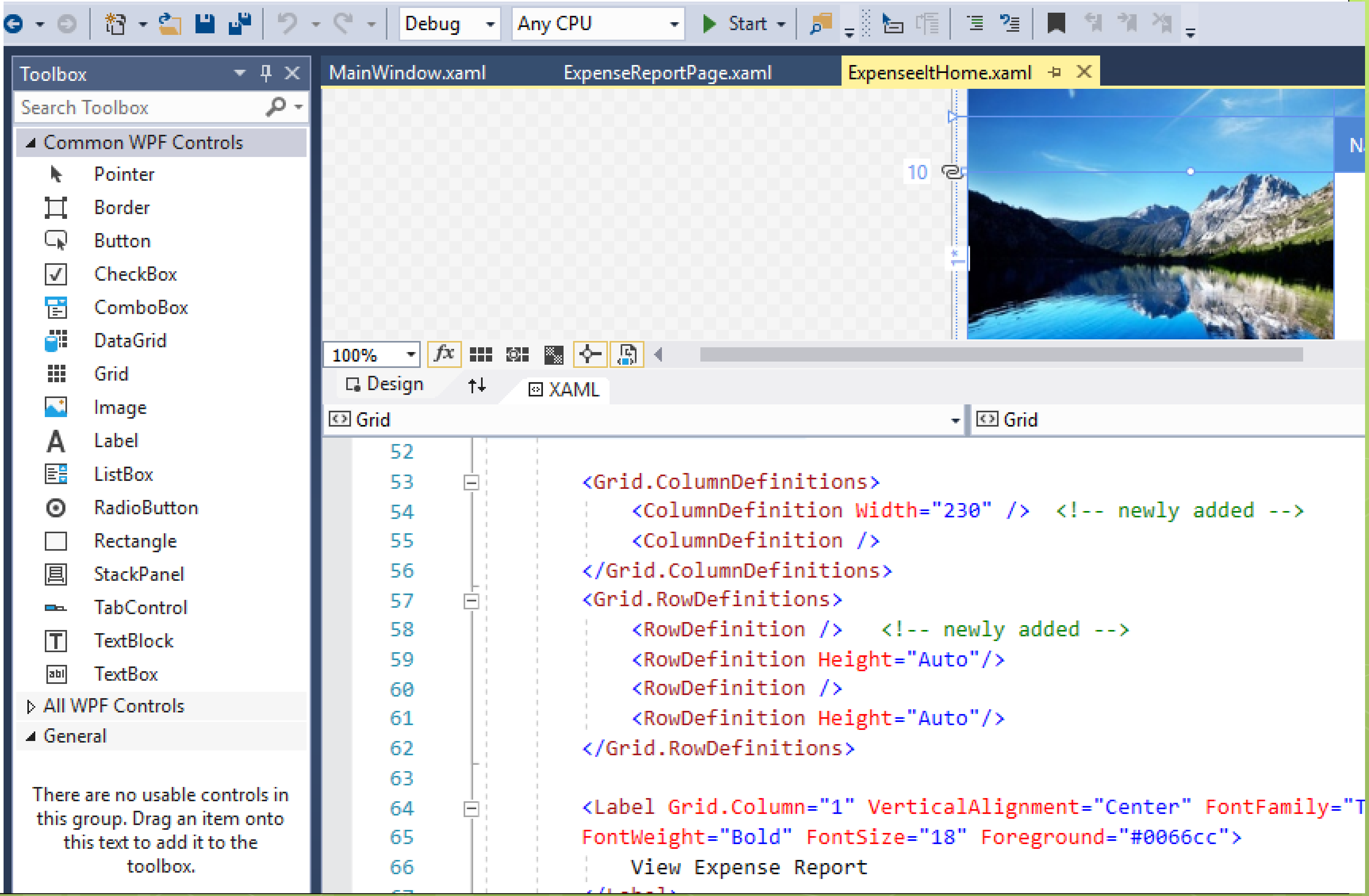
WPF Example



```
<Grid.ColumnDefinitions>
    <ColumnDefinition Width="230" />  <!-- newly added -->
    <ColumnDefinition />
</Grid.ColumnDefinitions>
<Grid.RowDefinitions>
    <RowDefinition />   <!-- newly added -->
    <RowDefinition Height="Auto"/>
    <RowDefinition />
    <RowDefinition Height="Auto"/>
</Grid.RowDefinitions>

<Label Grid.Column="1" VerticalAlignment="Center" FontFamily="Trebuchet MS"
FontWeight="Bold" FontSize="18" Foreground="#0066cc">
    View Expense Report
</Label>
```

WPF Toolbox



There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.

```

        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="230" />  <!-- newly added -->
            <ColumnDefinition />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition />    <!-- newly added -->
            <RowDefinition Height="Auto"/>
            <RowDefinition />
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <Label Grid.Column="1" VerticalAlignment="Center" FontFamily="T
FontWeight="Bold" FontSize="18" Foreground="#0066cc">
            View Expense Report
```

WPF Example

```
<TextBox Text="{Binding fname}"
        Margin="10" Grid.Column="1">
</TextBox>

<TextBox Text="{Binding lname}"
        Margin="10" Grid.Column="1"  Grid.Row="1">
</TextBox>
```