GUIs and WinForms/Avalonia

Cpt S 321

Washington State University

GUIs

- A Graphical User Interface, or GUI (pronounced gooey) is present in most software that you're used to using
- It is fairly likely that the average industry programmer these days will be making interfaces with HTML and CSS, but there are a variety of other options based on the desired type of application
- If we are targeting the Windows OS, we can create **WinForms** (Windows Forms) and **WPF** (Windows Presentation Foundation) applications which provide GUI design capabilities. It's much easier to do this in C# than managed C++.

WinForms

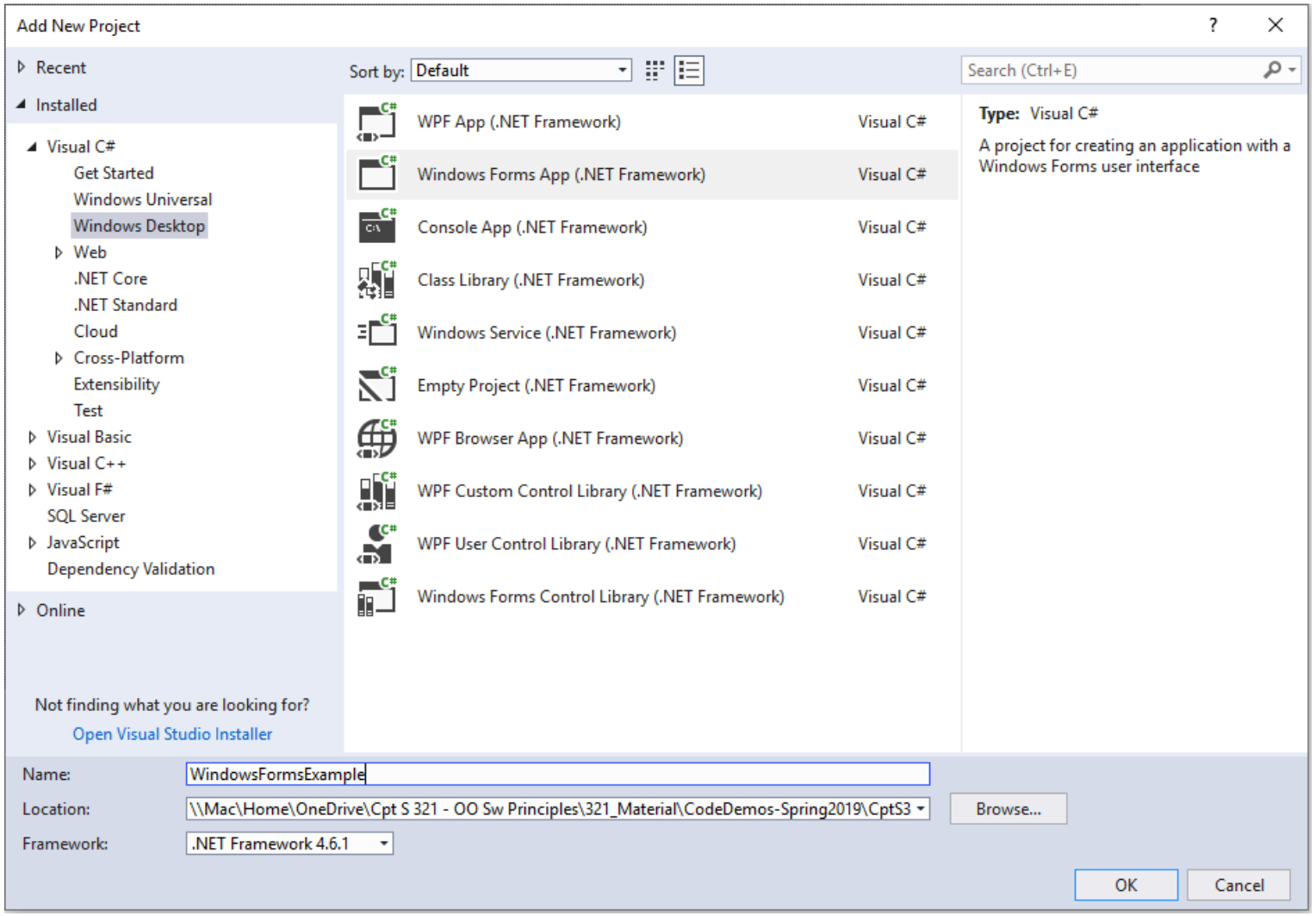
- Allows us to create applications that have typical UI elements such as buttons, list boxes, text boxes, progress bars, and so on.

- Drag-and-drop interface building within Visual Studio.

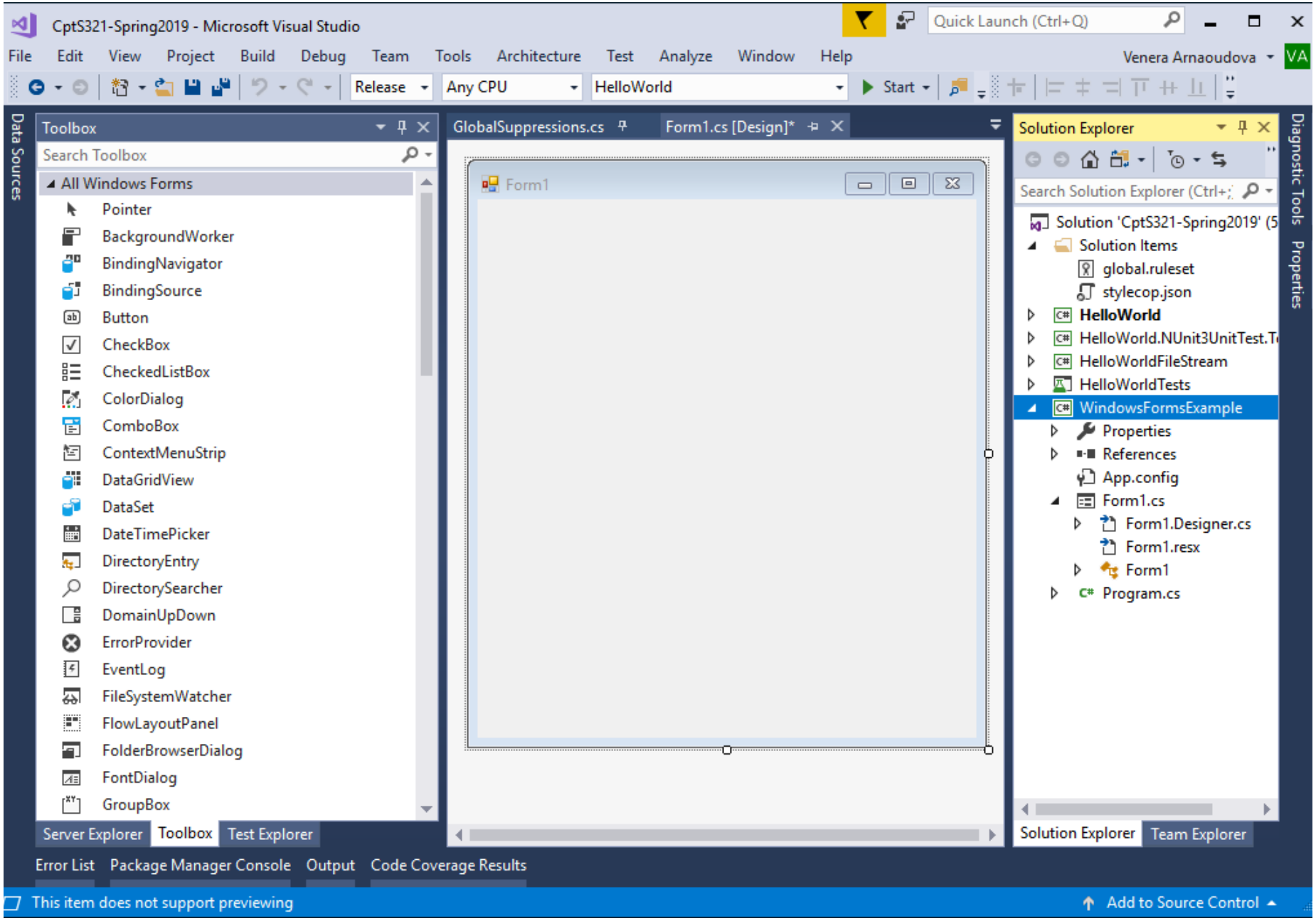
- This is what some of you are using for this class if you are building a Windows Desktop Application
  - Simpler layout system than WPF
  - Older than WPF => more extensive documentation

Windows Presentation Foundation (WPF)

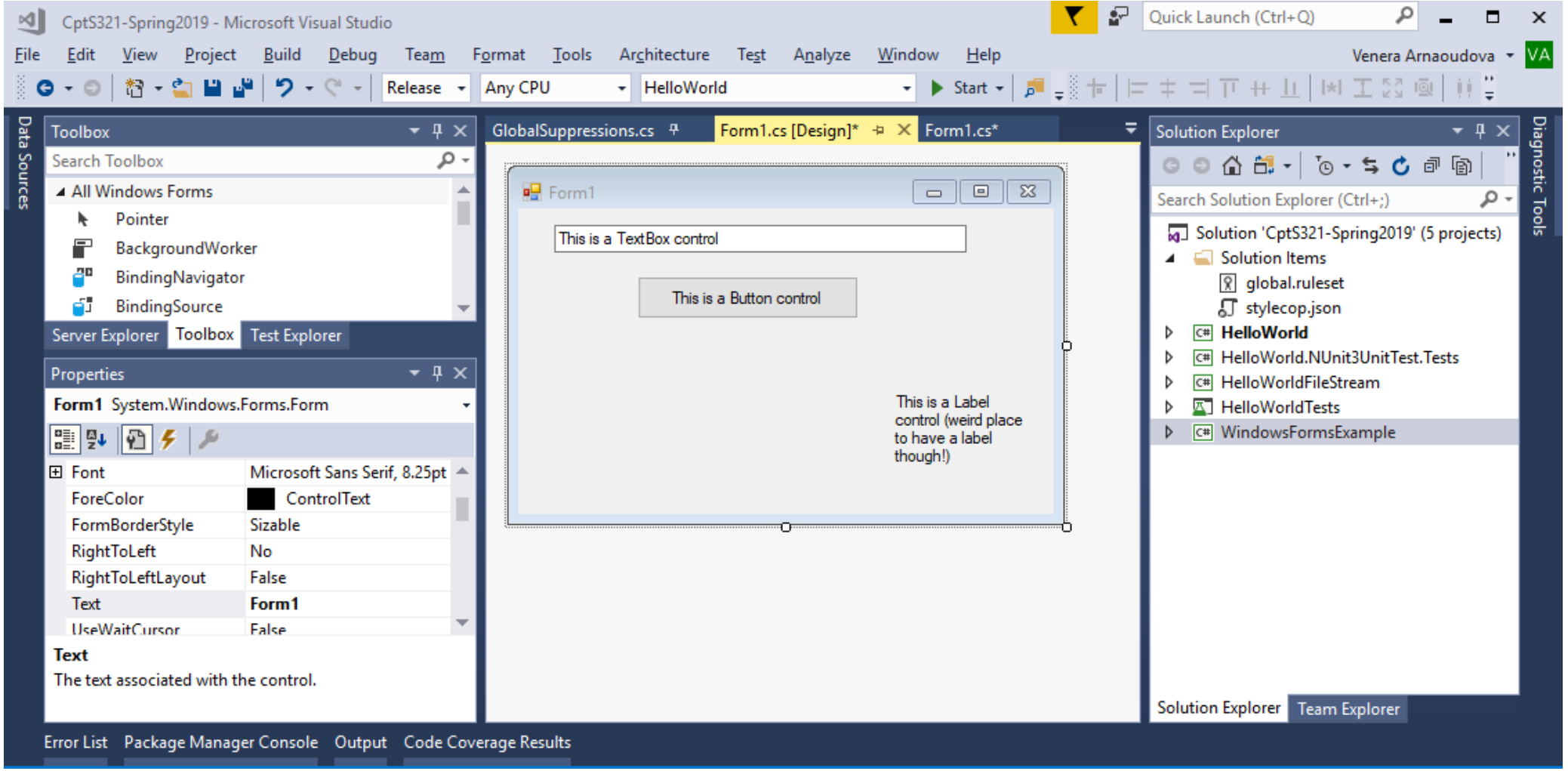
- Another GUI design option for C# applications that target the Windows OS.
- Newer, but not meant to completely replace WinForms
- XAML for the user interface declarations. Still has a drag-and-drop designer, but this produces the XAML code (a declarative XML-based language developed by Microsoft).
- Elements in the UI have a location and size that can be dependent upon what other UI element they're contained within. In contrast, pretty much everything in WinForms has a X,Y position relative to the upper-left corner of the parent UI element.



- In VS create a new project (Right click on the Solution -> Add -> New Project)
- Make sure you have Visual C# -> Windows Desktop selected in the left pane.
- Select “Windows Forms App (.NET Framework)”



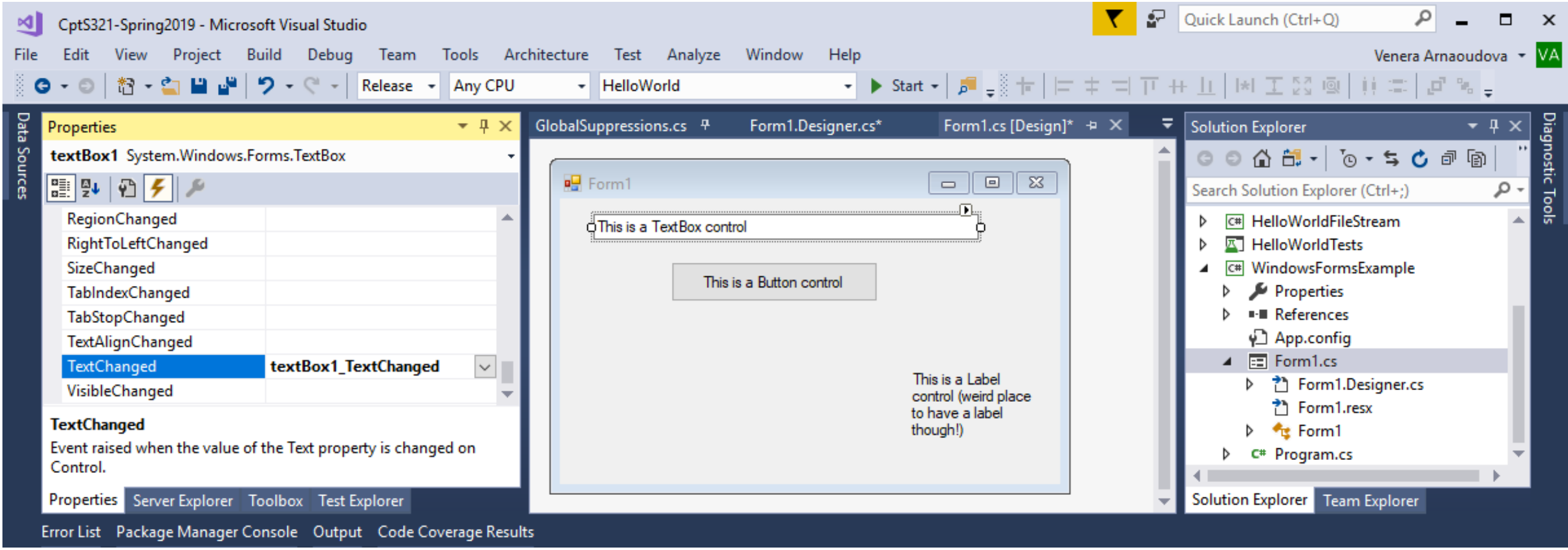
- You'll get the Windows Forms designer



- In the Toolbox window (View -> Toolbox if you can't see it) are controls. Controls are User Interface (UI) elements that you can drag and drop onto the form to design your application's interface.
- The Properties window allows us to set properties and **event handlers** for the control.

What’s an Event?

- Events are delivered by the operating system and can include:
  - Mouse movements
  - Mouse button clicks
  - Keyboard input
  - Other peripheral device input
  - Timer events
  - Operating system requests for the application to take some action, like re-render the interface or terminate.



- Different controls can have different types of events available. A button has a click event, a text box has a text-changed event, and so on.

Event-Driven Applications

- You're used to:

```
public void main()
{
  // Do …, code executes line by line
}
```

- GUI applications are event driven.
- An event-driven application still has a main function, but you generally don't implement it (in the old days you had to).

- The logic of an event-driven application's main function is:

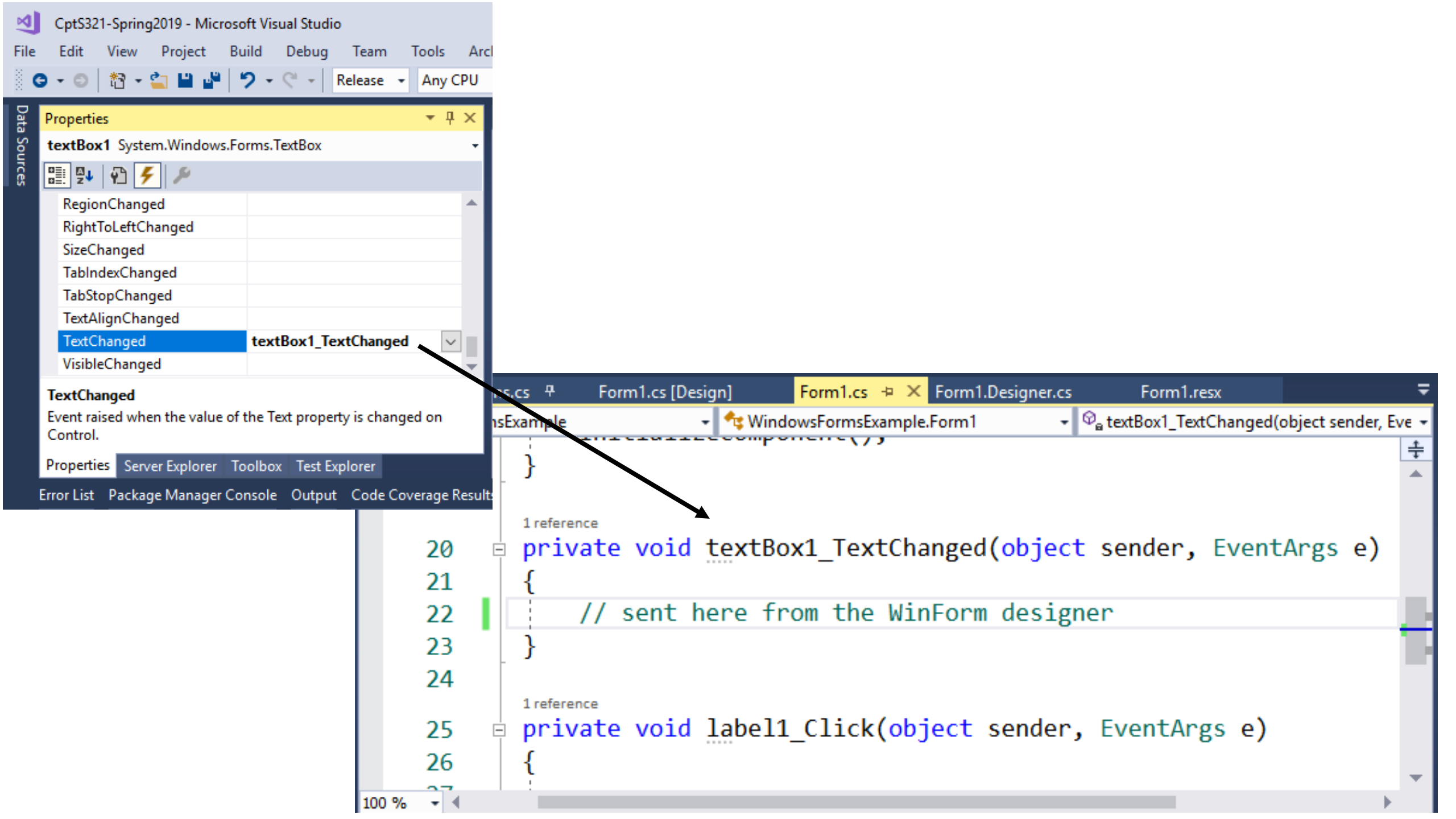
```
public void main()
{
  while (1 or more events in queue)
  {
    ProcessAllEventsInEventQueue();
    WaitFor1OrMoreEventsToBePutInQueue();
  }
}
```

Event-Driven Applications

- If we don’t write the main method then what do we do?

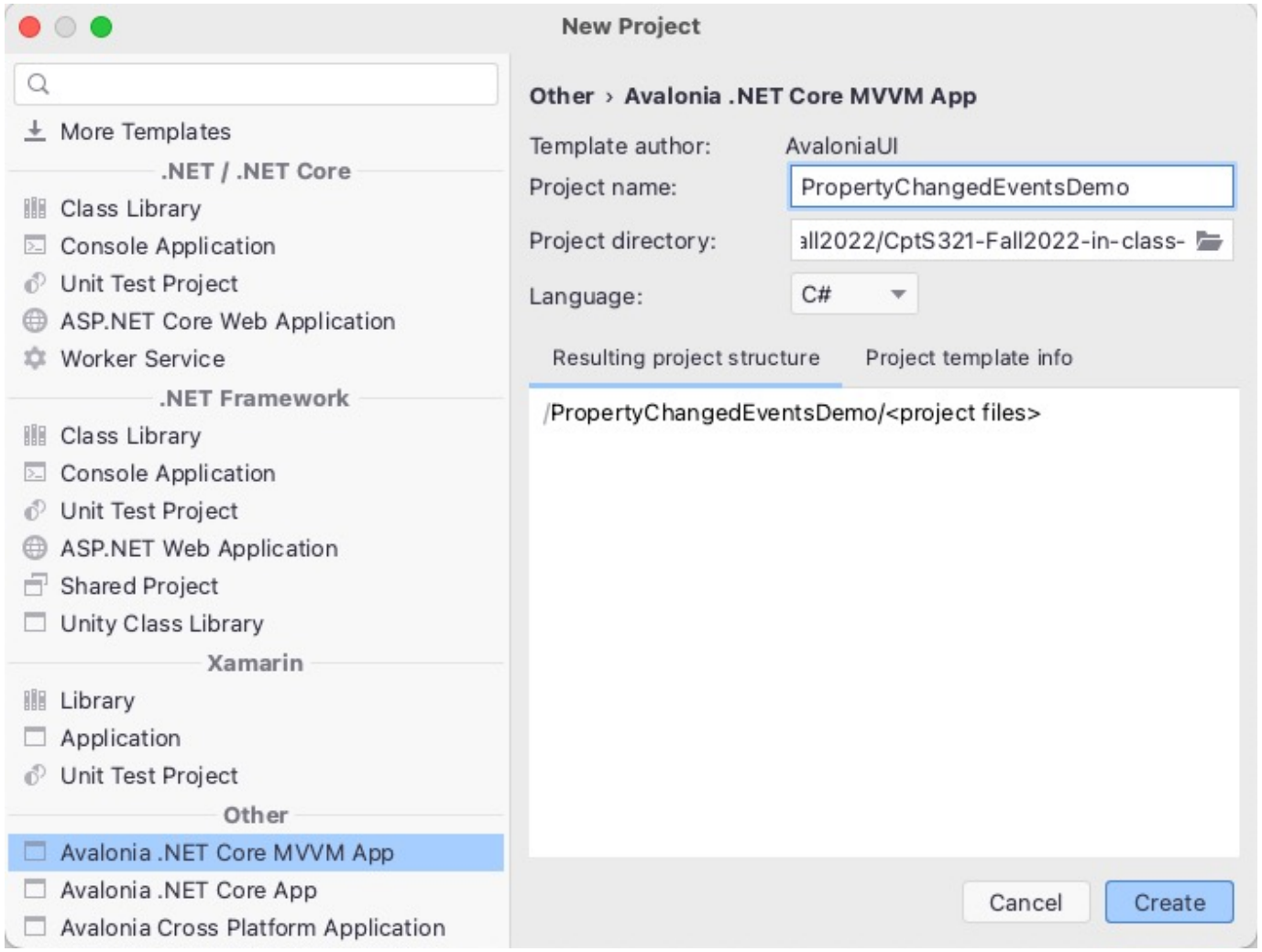
- We design our event-driven application by setting up (or linking) events for controls in the interface.

- **Events handlers** are simply methods in our code.


CptS321-Spring2019 - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools
Release
Any CPU
Data Sources
Properties
textBox1 System.Windows.Forms.TextBox
RegionChanged
RightToLeftChanged
SizeChanged
TabIndexChanged
TabStopChanged
TextAlignChanged
TextChanged
textBox1_TextChanged
VisibleChanged
TextChanged
Event raised when the value of the Text property is changed on Control.
Properties
Server Explorer
Toolbox
Test Explorer
Error List
Package Manager Console
Output
Code Coverage Results
Form1.cs [Design]
Form1.cs
Form1.Designer.cs
Form1.resx
WindowsFormsExample.Form1
textBox1_TextChanged(object sender, Eve
}
1 reference
20 private void textBox1_TextChanged(object sender, EventArgs e)
21 {
22 // sent here from the WinForm designer
23 }
24
1 reference
25 private void label1_Click(object sender, EventArgs e)
26 {
100 %

Avalonia UI - Cross Platform C# Desktop Application Development

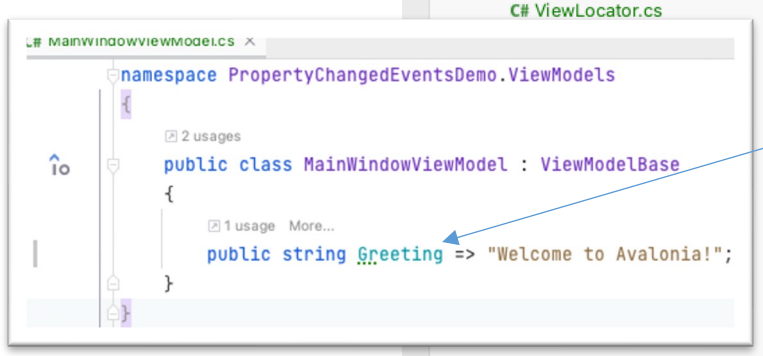
- Open-source UI framework for .NET (Windows, macOS, Linux, iOS, Android, Web Assembly and Raspberry Pi)
- Avalonia's Instant Viewer allows us to visualize (preview) the application
- No Graphical designer
- In addition to the C# code, we write XAML code for the UI definition (very similar to WPF)

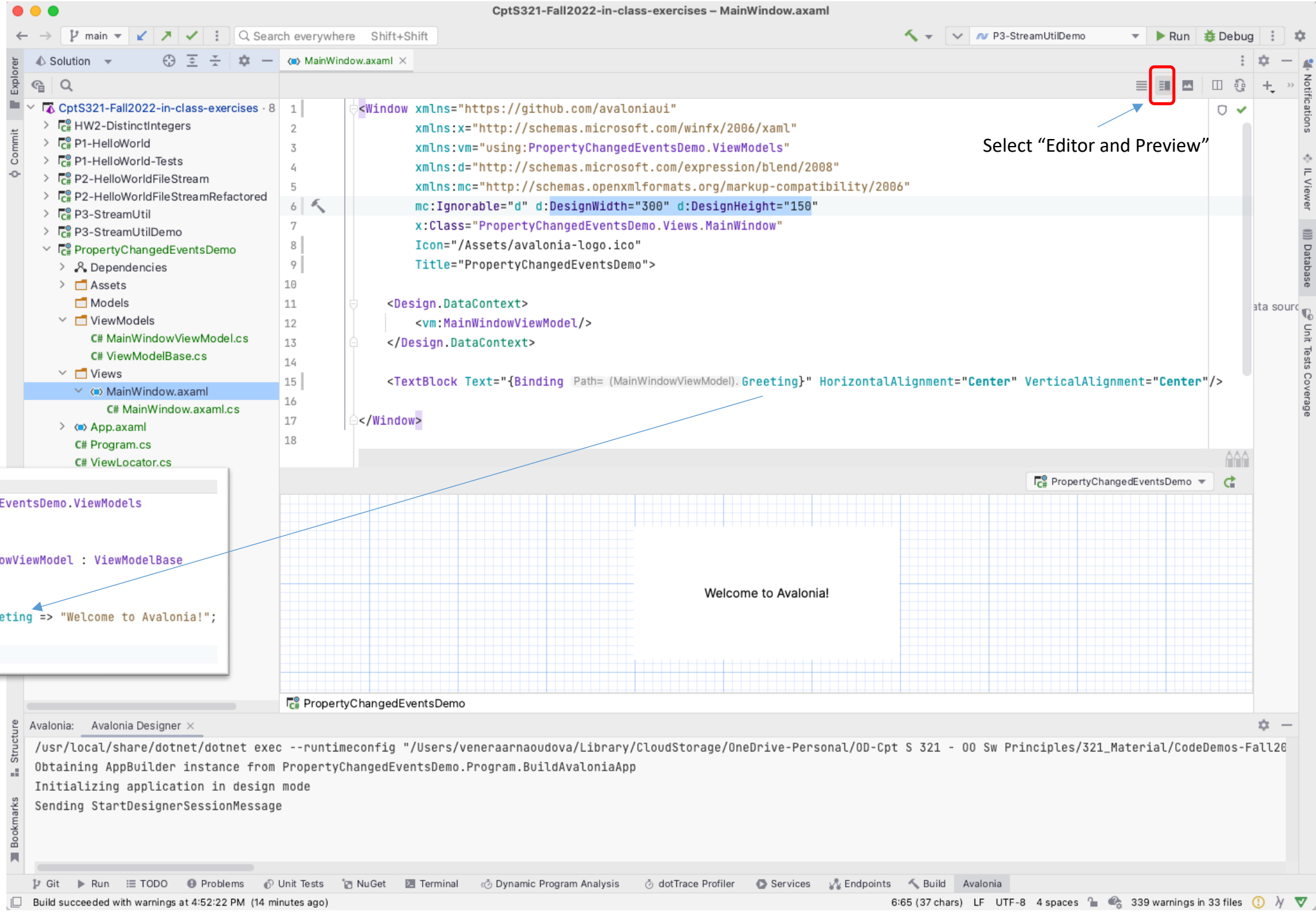


Using Rider, create a new Avalonia application:

- In Rider create a new project (Right click on the Solution -> Add -> New Project)
- Select “Avalonia .NET Core MVVM App”

- Your Avalonia Application



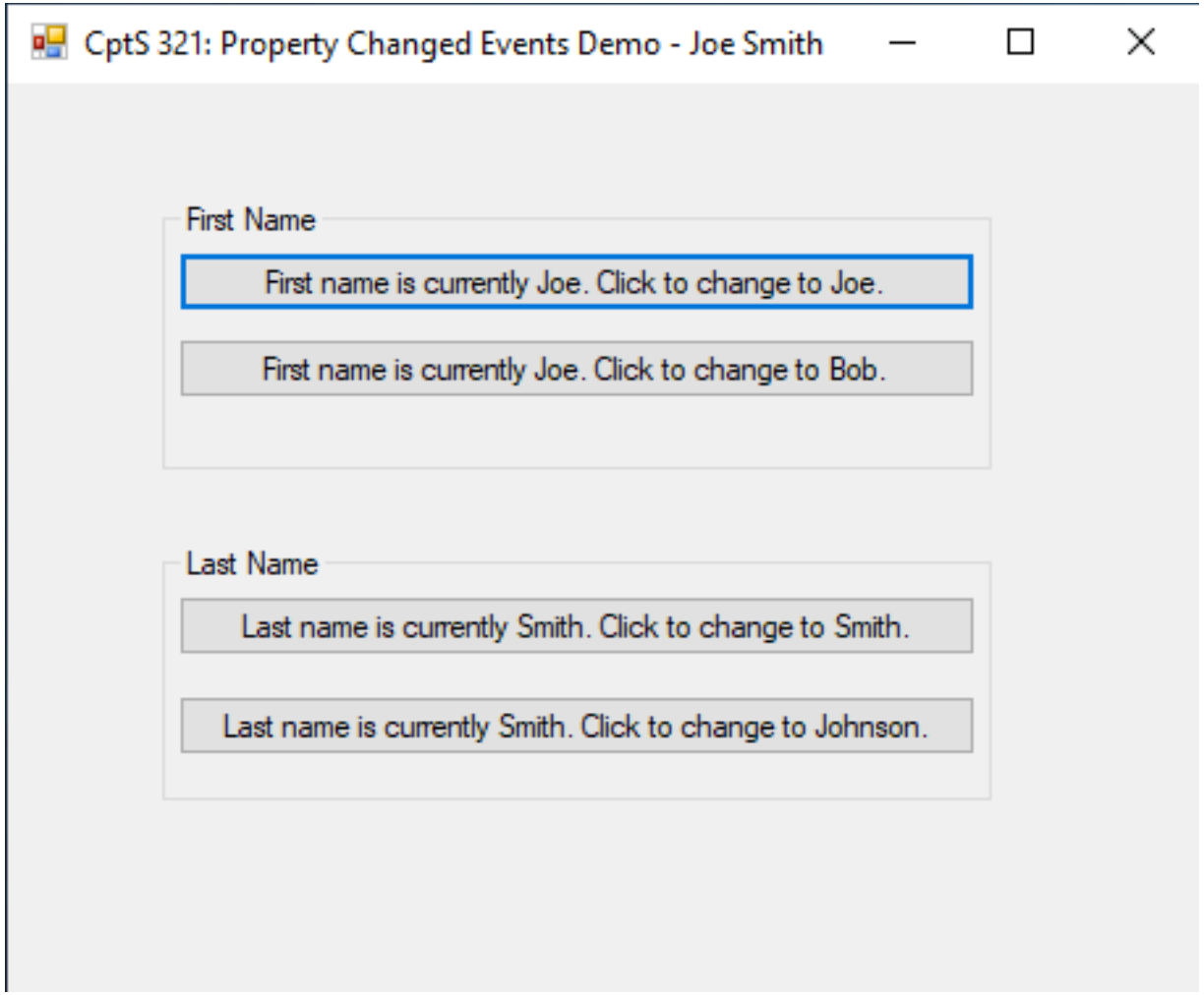


Code Demo

Tasks for today:

1. Create a project "**PropertyChangedEventsDemo**"
2. Build a GUI that looks like the one shown here
3. Create a class Person with
   - Fields firstName and lastName
   - Properties FirstName and LastName

Do not worry about linking the UI to the class Person for now.



Additional notes

WinForm

- Check the GroupBox control
- Explore the rest of the controls

Avalonia

```
<StackPanel
    <StackPanel ... ><!--First Name panel-->
      <TextBlock ...
      <StackPanel ...
        <Button ...
        <Button ...
    <StackPanel ... ><!--Last Name panel-->
      <TextBlock ...
      <StackPanel ...
        <Button ...
        <Button ...
```