Advanced Programming 2

DR. ELIAHU KHALASTCHI

2016

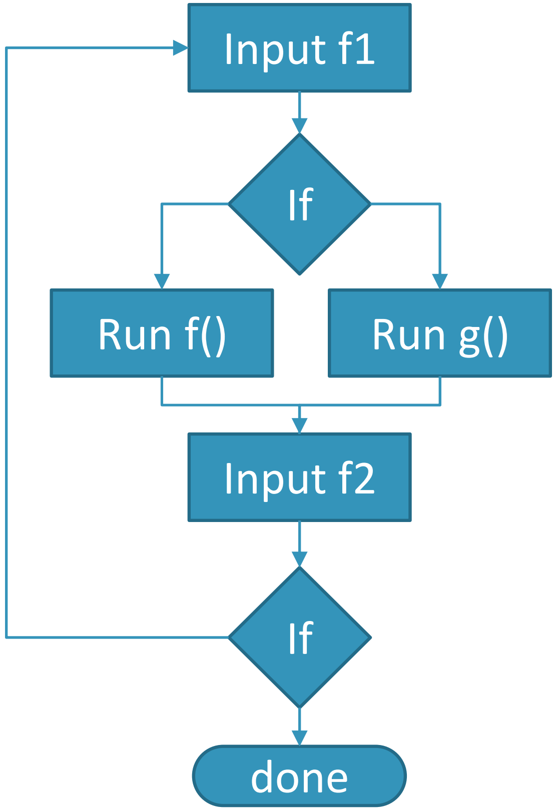
Agenda…

- Event Driven Programming
- The Multiplatform GUI problem (JVM)
- Visual editor to XML (and not code)
- WPF (.Net)
  - WPF tutorial
- XAML
- Custom WPF Control

Event Driven Programming

Procedural Programming

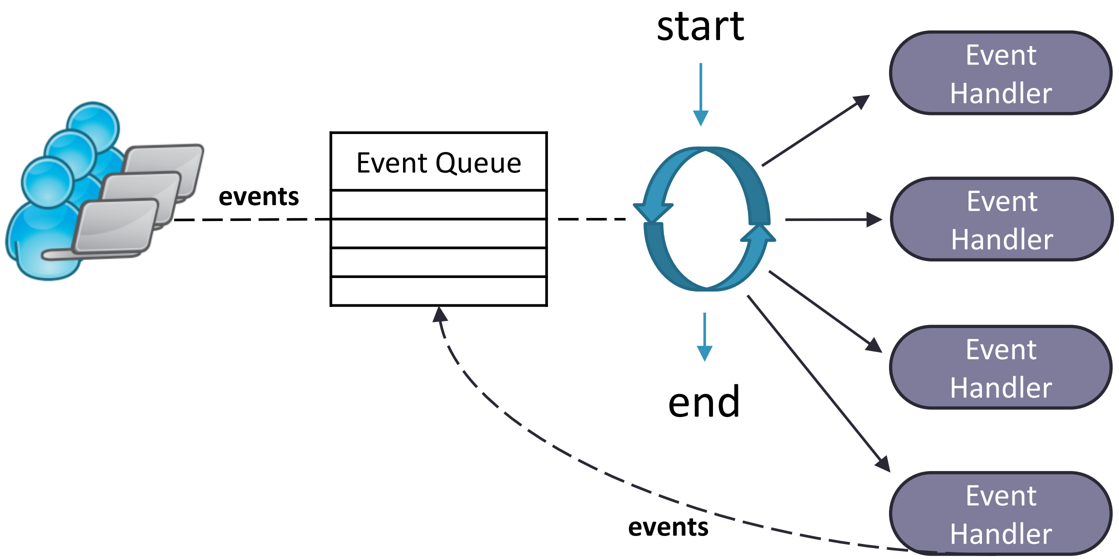
- A procedural program **dictates** when events (or inputs) are acquired
- Events / inputs are acquired in a procedural manner
  - Get an input
  - Handle the input
  - Get the next input, and so on…
- Suitable for console applications
- NOT suitable for
  - GUI based applications – where the user dictates when events occur
    - e.g., the user decides what button to push and when to push it
  - Server side – where clients dictate when events occur
    - i.e., the server does not know when clients will connect or what will they request
  - etc.



Event Driven Programming

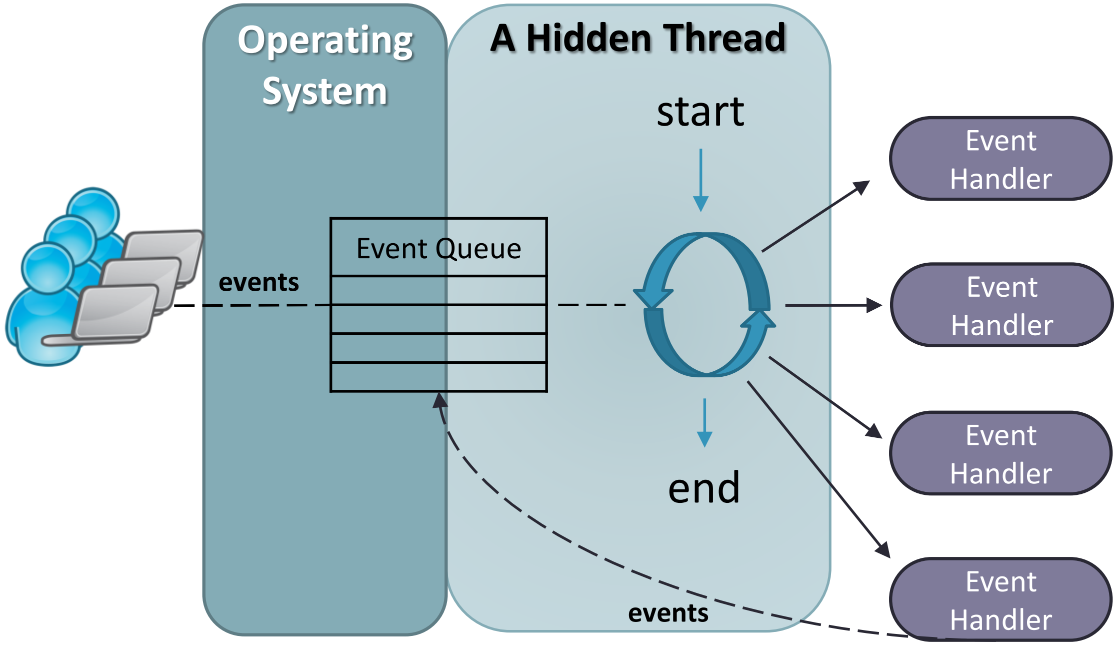
- The program continuously **listens** to defined **events** that **may occur** at **any given time**
- Upon the occurrence of an event, the program “fires” the appropriate **event handler**
  - This is the desired reaction for the event defined by the programmer
  - The event handler code **may trigger new events** as well

- Event driven programming includes:
  - The defined **events**
  - The **event queue** of created runtime events
  - The **event handlers** for the defined events
  - The **main event loop** that extracts events from the queue and triggers the event-handler’s code



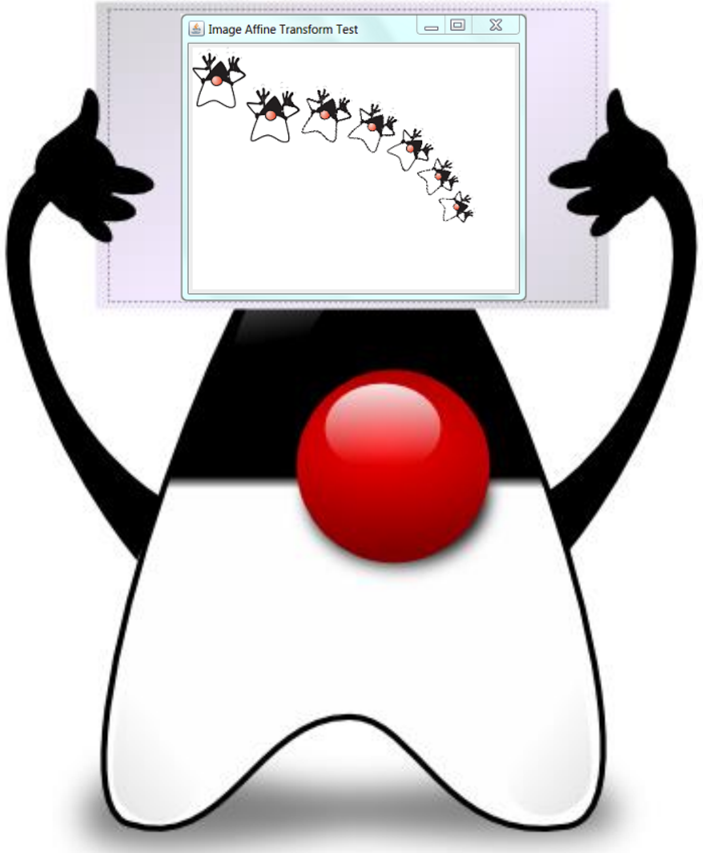
Event Driven Programming

- The program continuously **listens** to defined **events** that **may occur** at **any given time**
- Upon the occurrence of an event, the program “fires” the appropriate **event handler**
  - This is the desired reaction for the event defined by the programmer
  - The event handler code **may trigger new events** as well
- Event driven programming includes:
  - The defined **events**
  - The **event queue** of created runtime events
  - The **event handlers** for the defined events
  - The **main event loop** that extracts events from the queue and triggers the event-handler’s code



GUI in Java

THE MULTIPLATFORM PROBLEM



The multiplatform problem

The same Java code

↓

The JVM

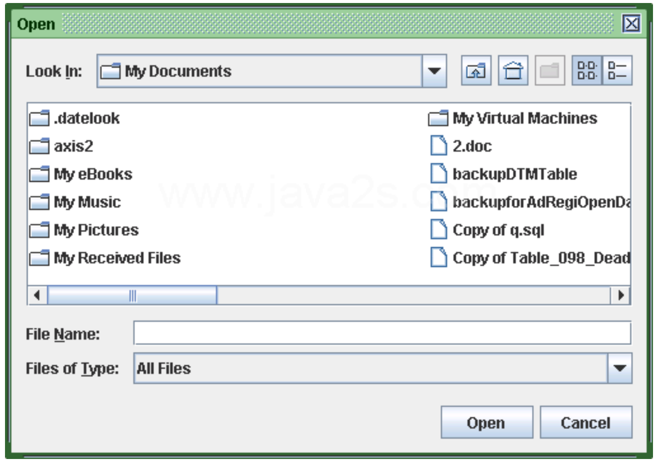
↓

But a window, a button, or text box are native to the OS

We need to use the ***look & feel*** of the OS

GUI Technologies

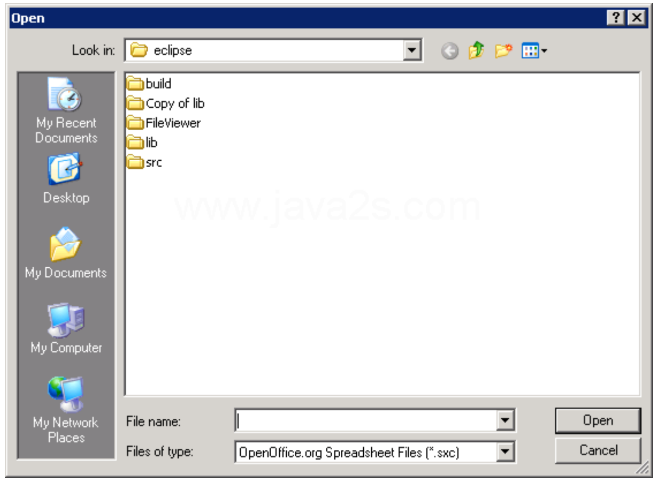
- AWT – Abstract Windowing Toolkit (1995)
  - Weighty – calls OS's components
  - Lowest common denominator problem
- Swing / JFC (1998)
  - The official GUI for Java (SUN, later ORACLE)
  - Every component is written in Java (light weight)
  - Tires to mimic OS's look
- SWT - Standard Widget Toolkit (IBM, 2001)
  - Uses the OS's components when available
  - Uses Java implementation when they are not
  - OS's look & feel



Look In: My Documents

File Name:

Files of Type: All Files

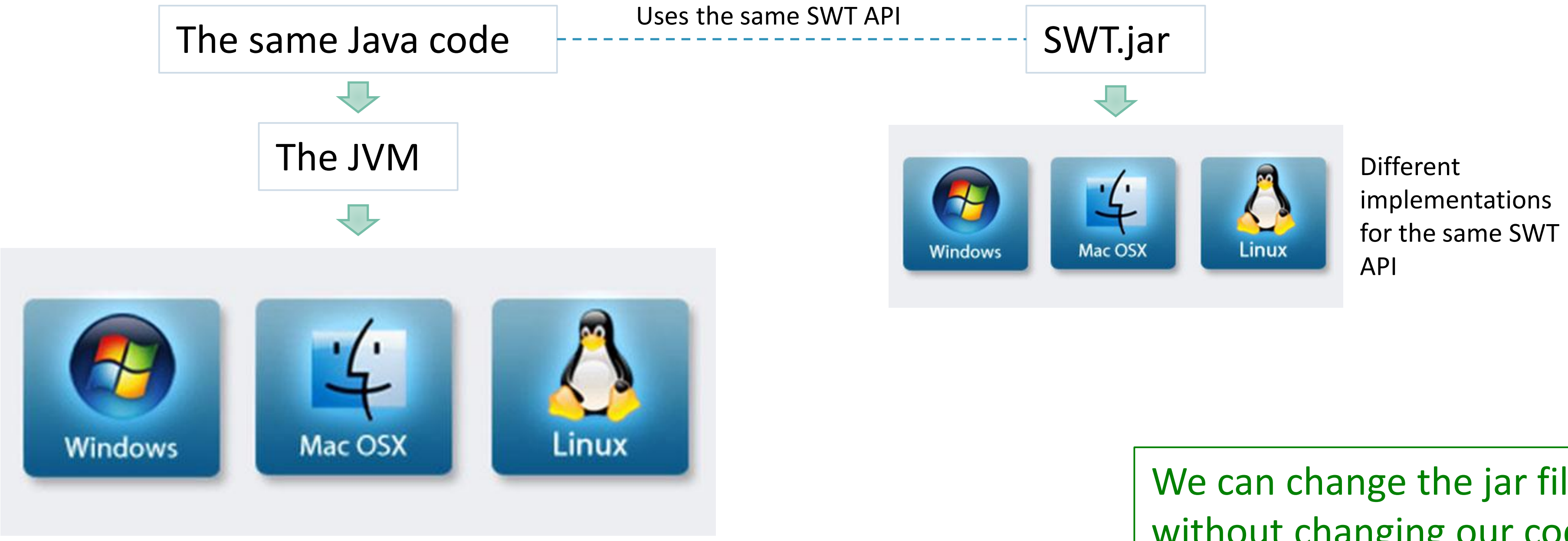


Look in: eclipse

File name:

Files of type: OpenOffice.org Spreadsheet Files (*.sxc)

SWT technology uses dependency injection



The same Java code

Uses the same SWT API

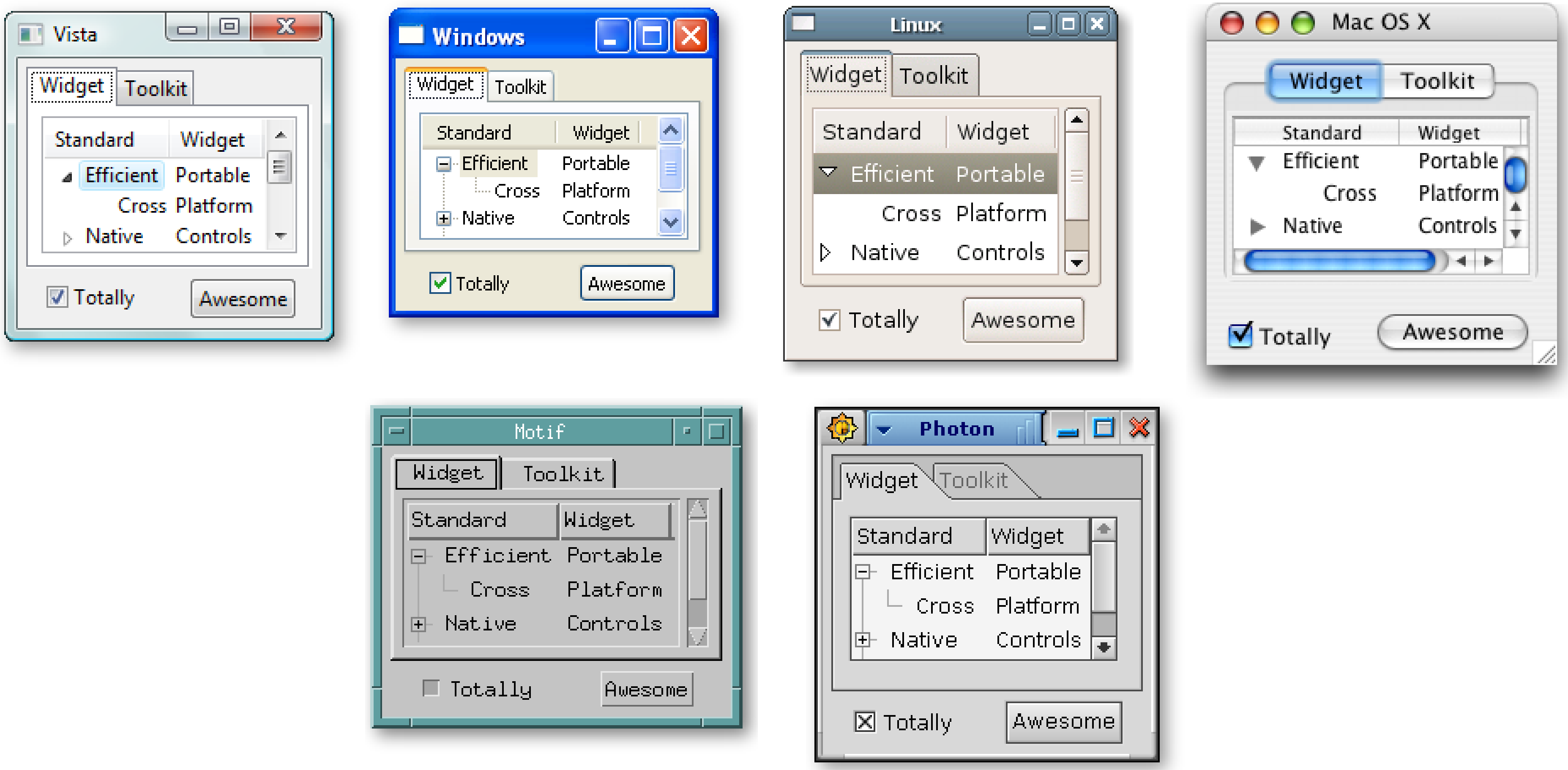
SWT.jar

The JVM

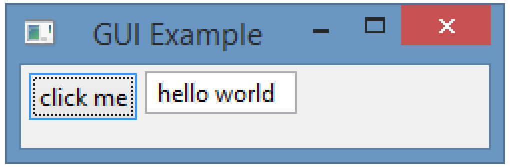
Different implementations for the same SWT API

We can change the jar file,
without changing our code,
and get the look & feel of the OS. ☺

Same java code, different OS GUI



Java GUI Example – SWT



```
private void initComponents(){
  display = new Display();
  shell = new Shell(display);
  shell.setSize(250,80);
  shell.setText("GUI Example");
  shell.setLayout(new RowLayout());

  Button b=new Button(shell,SWT.PUSH);
  b.setText("click me");

  final Text t =new Text(shell, SWT.BORDER);

  // add an event handler for pushing the button
  b.addSelectionListener(new SelectionListener() {
    @Override
    public void widgetSelected(SelectionEvent e) {
      t.setText("hello world");
    }
    @Override
    public void widgetDefaultSelected(SelectionEvent e){}
   }
  );
  shell.open();
}
```

Window Creation

A button

A text box

```
// runs in a different thread
public void run(){
 initComponents();
 // main event loop
 while(!shell.isDisposed()){ // window isn't closed
  if(!display.readAndDispatch()){
   display.sleep();
  }
 }
 display.dispose();
}
```

The main event loop (runs as a thread)

The event handler assignment

The event object

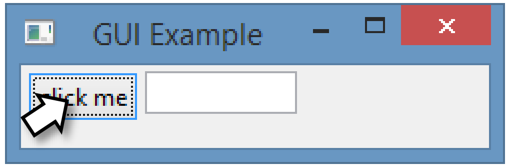
The event handler assigned to the button selection event changes the text in the text box to “hello world”

Java GUI Example – SWT

```
private void initComponents(){
  display = new Display();
  shell = new Shell(display);
  shell.setSize(250,80);
  shell.setText("GUI Example");
  shell.setLayout(new RowLayout());

  Button b=new Button(shell,SWT.PUSH);
  b.setText("click me");

  final Text t =new Text(shell, SWT.BORDER);

  // add an event handler for pushing the button
  b.addSelectionListener(new SelectionListener() {
    @Override
    public void widgetSelected(SelectionEvent e) {
      t.setText("hello world");
    }
    @Override
    public void widgetDefaultSelected(SelectionEvent e){}
   }
  );
  shell.open();
}
```

```
// runs in a different thread
public void run(){
 initComponents();
 // main event loop
 while(!shell.isDisposed()){ // window isn't closed
  if(!display.readAndDispatch()){
   display.sleep();
  }
 }
 display.dispose();
}
```

The main event loop (runs as a thread)



The event handler assigned to the button selection event changes the text in the text box to “hello world”

Shared Concepts

PRESENTATION & PRESENTATION LOGIC

Shared Concepts

PRESENTATION

Setting a layout

- Where each component should be
- E.g., a border pan, a grid view

Adding components, for instance:

- TextBox userName = new TextBox();
- Button reset = new Button();

Each technology has a different API yet, the concepts are the same.

PRESENTATION LOGIC

Adding logic

- typically with strategy pattern

For instance:

```
reset.addListener(
 new SelectionListener(){
          void selected(Event e){
                    userName.setText(“”);
                    notifyObservers();
          }
 }
);
```

Button examples

```
// SWING
JButton b= new JButton(“click me");
b.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    //…
  }
});
```

```
// JavaFX
Button b= new Button(“click me");
b.setOnAction(new EventHandler<ActionEvent>(){
   public void handleEvent(ActionEvent e){
      //…
   }
});
```

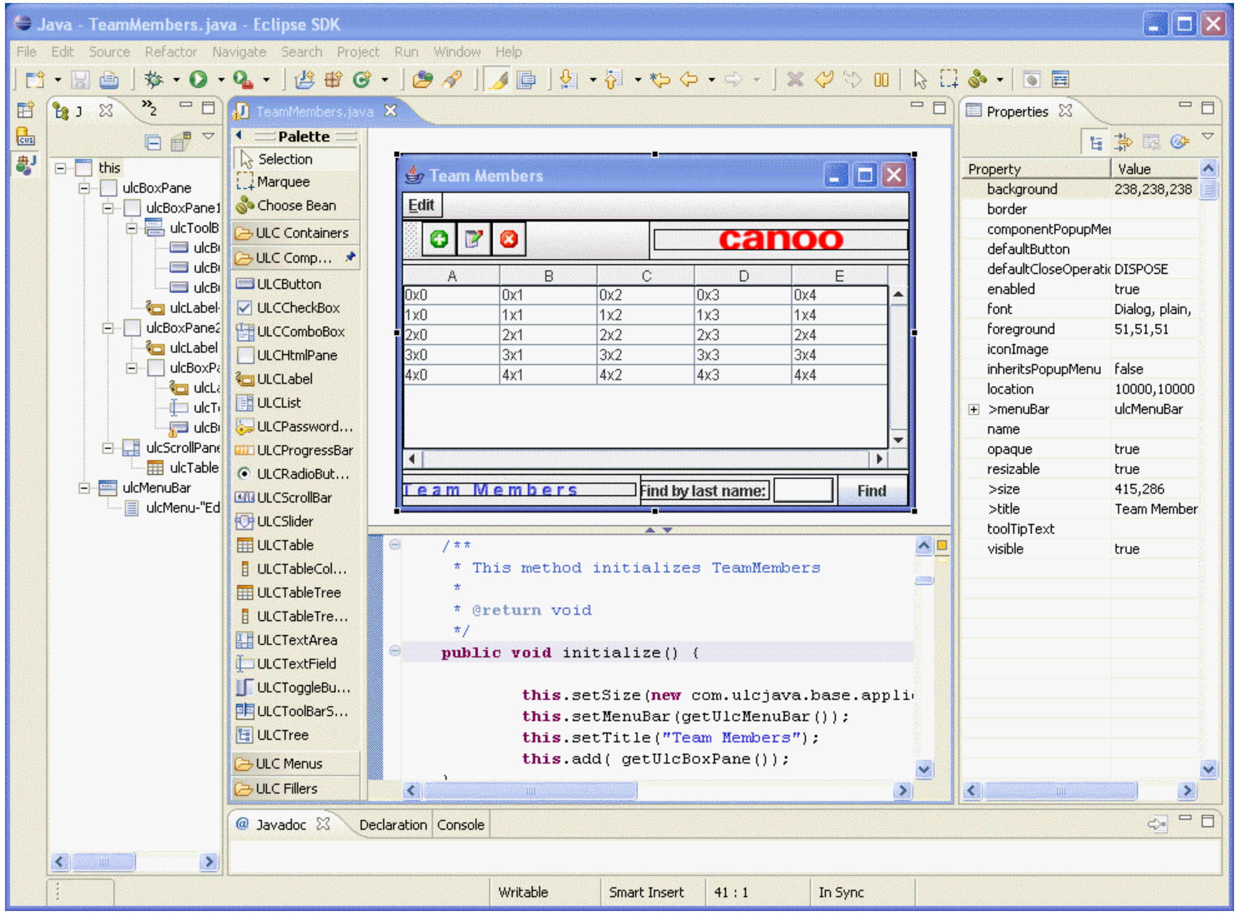
```
// SWT
Button b=new Button(shell,SWT.PUSH);
b.setText("click me");
b.addSelectionListener(new SelectionListener() {
  public void widgetSelected(SelectionEvent e) {
    //…
  }
  public void widgetDefaultSelected(SelectionEvent arg0) {}
});
```

Visual Editing

TO CODE / TO XML

Visual Editor → code behind

- The code is written in a **specific language**
  - Hard to migrate the visuals to a project written in another language
- The code is messy
- Hard to maintain
- We want as little code-behind as possible



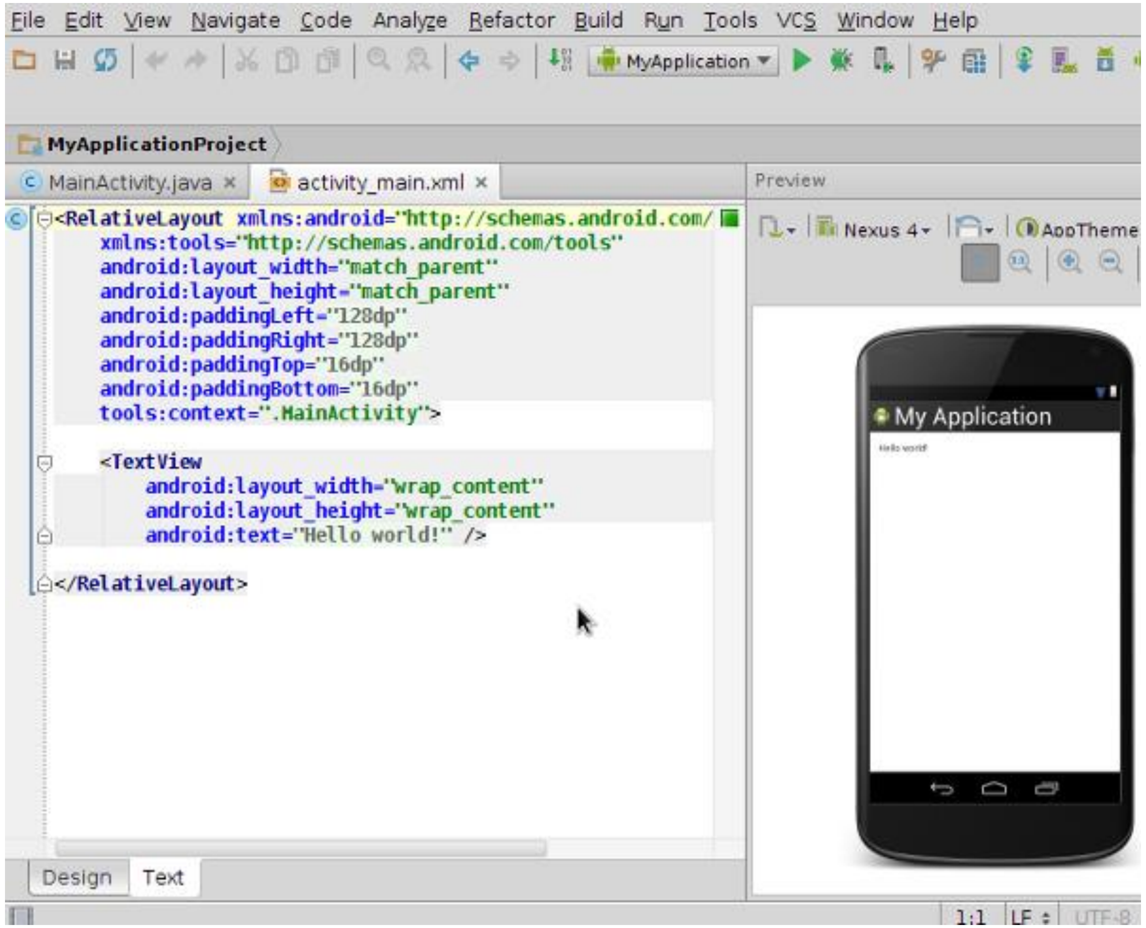
A	B	C	D	E
0x0	0x1	0x2	0x3	0x4
1x0	1x1	1x2	1x3	1x4
2x0	2x1	2x2	2x3	2x4
3x0	3x1	3x2	3x3	3x4
4x0	4x1	4x2	4x3	4x4

```
/**
 * This method initializes TeamMembers
 *
 * @return void
 */
public void initialize() {

        this.setSize(new com.ulcjava.base.appli
        this.setMenuBar(getUlcMenuBar());
        this.setTitle("Team Members");
        this.add( getUlcBoxPane());
```

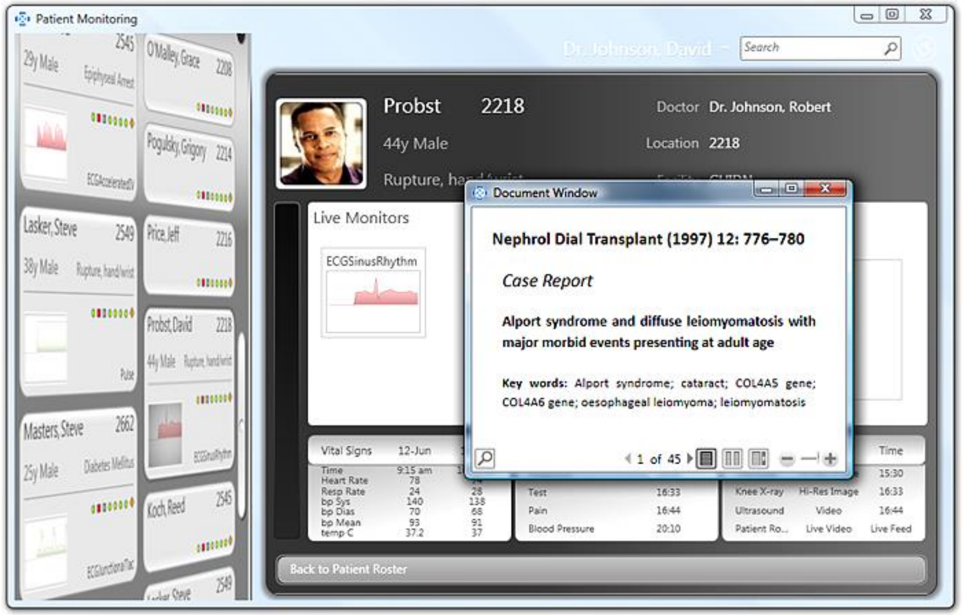
Property	Value
background	238,238,238
border	
componentPopupMe	
defaultButton	
defaultCloseOperati	DISPOSE
enabled	true
font	Dialog, plain,
foreground	51,51,51
iconImage	
inheritsPopupMenu	false
location	10000,10000
>menuBar	ulcMenuBar
name	
opaque	true
resizable	true
>size	415,286
>title	Team Member
toolTipText	
visible	true

Visual Editor ➔ XML code

- The visuals are not specific to any programming language
  - Easy to migrate
- Typically, the functionality is still done in a code-behind
- The WPF technology gives us just that



```xml
<RelativeLayout xmlns:android="http://schemas.android.com/
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="128dp"
    android:paddingRight="128dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello world!" />

</RelativeLayout>
```



Probst 2218

44y Male

Rupture, hand/wrist

Doctor Dr. Johnson, Robert

Location 2218

Document Window

Nephrol Dial Transplant (1997) 12: 776–780

Case Report

Alport syndrome and diffuse leiomyomatosis with major morbid events presenting at adult age

Key words: Alport syndrome; cataract; COL4A5 gene; COL4A6 gene; oesophageal leiomyoma; leiomyomatosis

1 of 45

Vital Signs	12-Jun	
Time	9:15 am	
Heart Rate	78	
Resp Rate	24	28
bp Sys	140	138
bp Dias	70	68
bp Mean	93	91
temp C	37.2	37

Test	16:33
Pain	16:44
Blood Pressure	20:10

		Time
		15:30
Knee X-ray	Hi-Res Image	16:33
Ultrasound	Video	16:44
Patient Ro...	Live Video	Live Feed

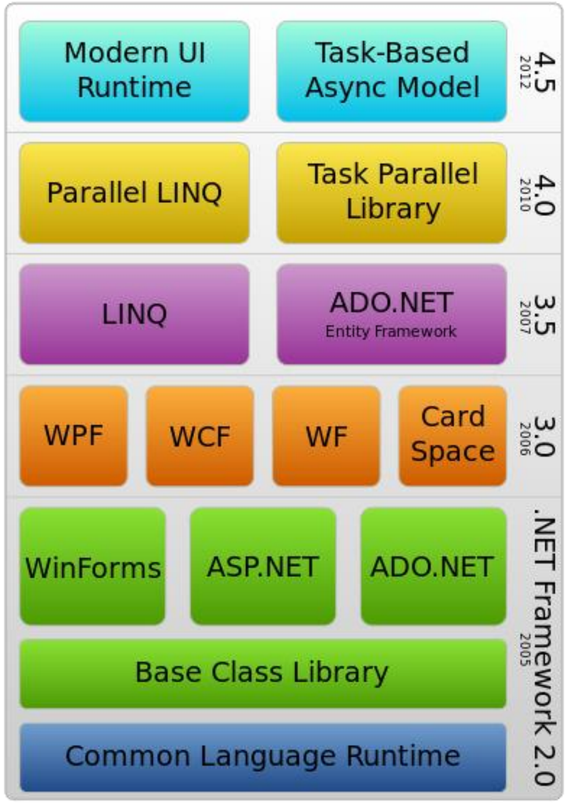
Introduction to WPF

EVENT DRIVEN PROGRAMMING IN WINDOWS PRESENTATION FOUNDATION

WPF – Windows Presentation Foundation

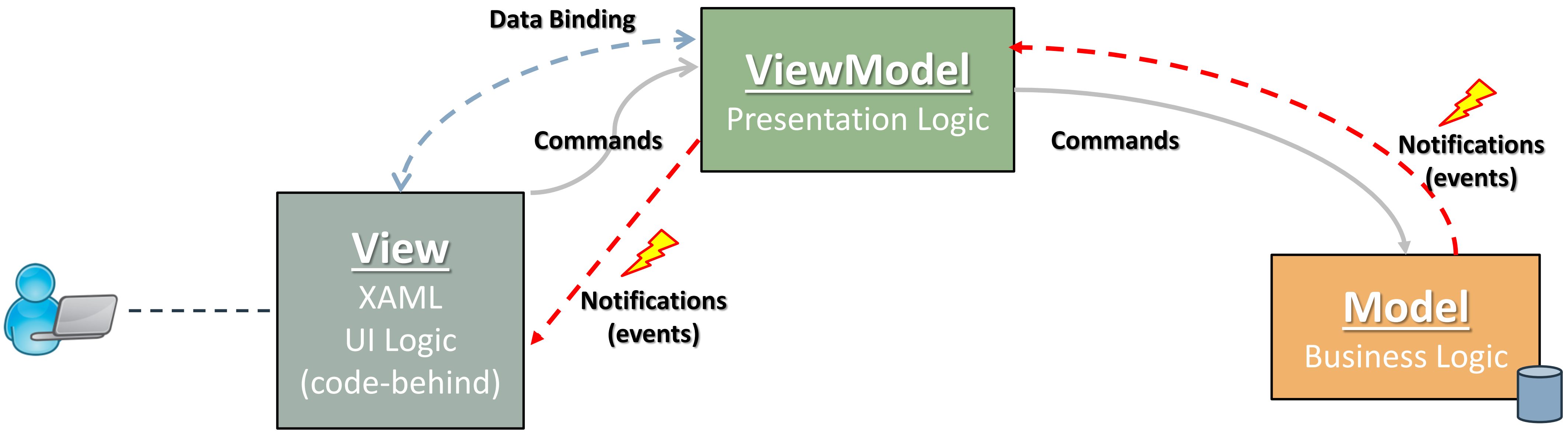
- Presented at 2006
- A platform for building rich user experiences on Windows
- Unified platform for modern user interfaces
- A WPF interface can combine images, text, 2D and 3D graphics,
- and more…

Why use WPF when we already have winforms??



The .NET Framework Stack

WPF technology allows an MVVM architecture



Data Binding

Commands

Notifications (events)

ViewModel
Presentation Logic

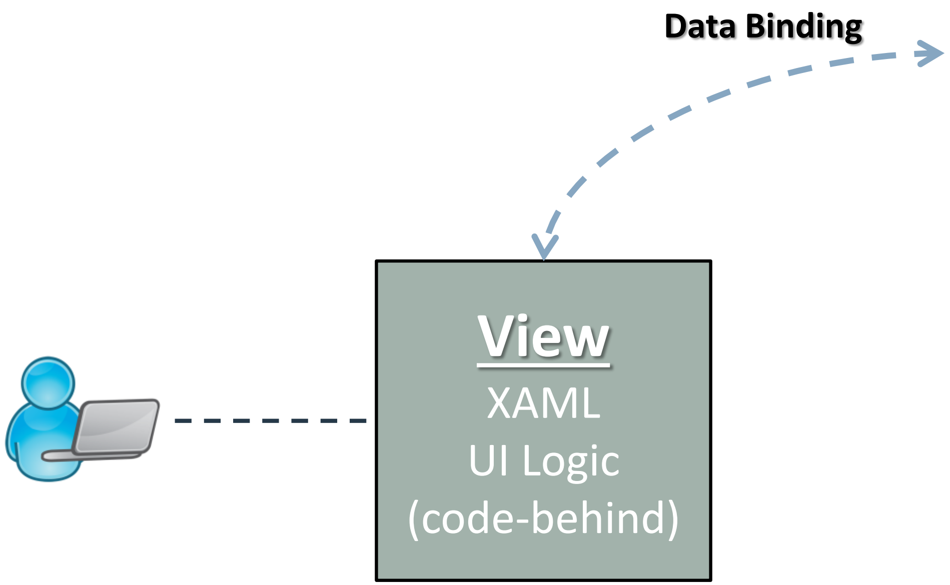
Commands

Notifications (events)

View
XAML
UI Logic
(code-behind)

Model
Business Logic

WPF technology allows an MVVM architecture

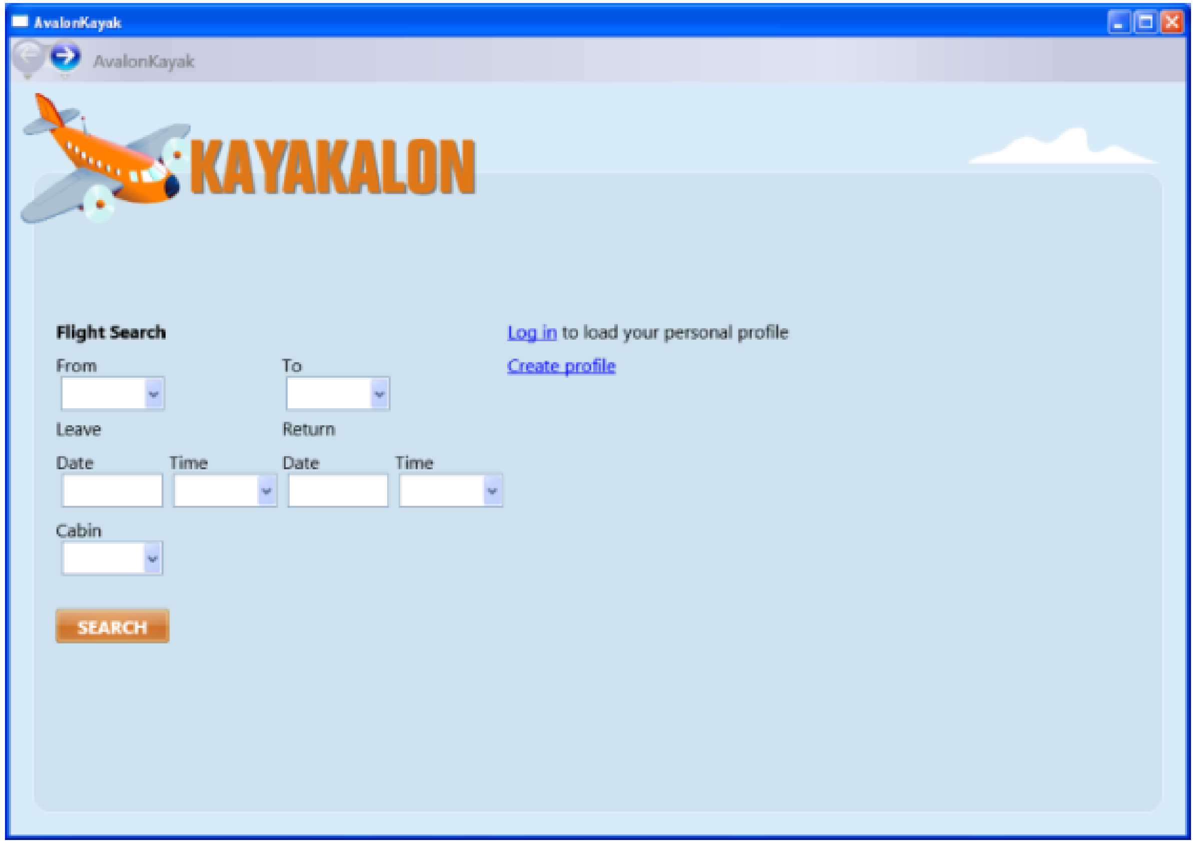


- Data Binding support
  - Display events automatically change the data
  - Data events are automatically displayed
  - We don’t have to command it, just define a binding
- The UI Logic is written in XAML (XML file)
  - The visual designer doesn’t have to be a programmer
  - The visual design is independent from the project
- Very small code-behind is needed

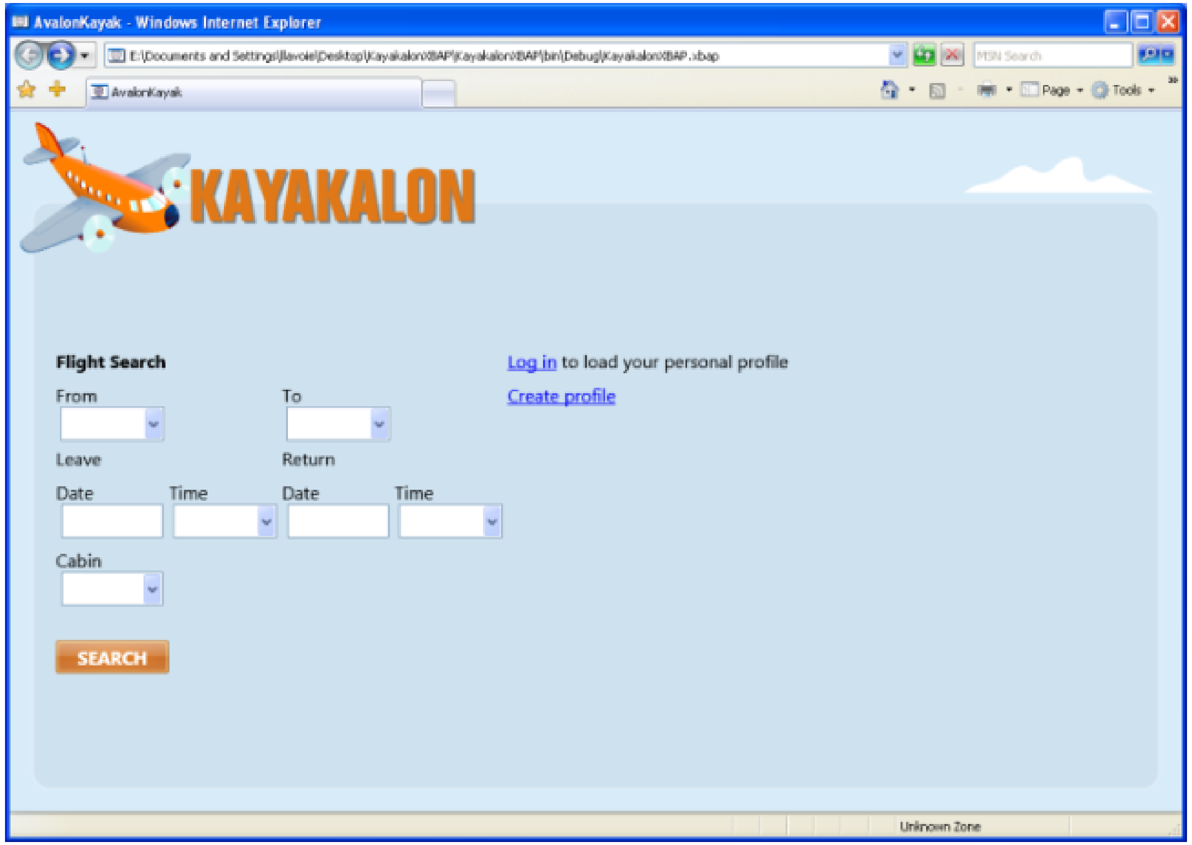
In addition…

- UI customization & graphics
  - We can change the looks and the behaviors as we wish
  - The graphics support is way better than WinForms: 3D objects, Animations, and Media
- “There is no control for that” no longer applies
  - Easy to write your own
  - There are literally hundreds of third-party WPF controls available
- Ability to run in a browser
- Microsoft firmly switched its focus on WPF
- Strong ties to Silverlight technology
  - If you know WPF, it is easy to use Silverlight

Same visuals, different platforms…



Log in to load your personal profile

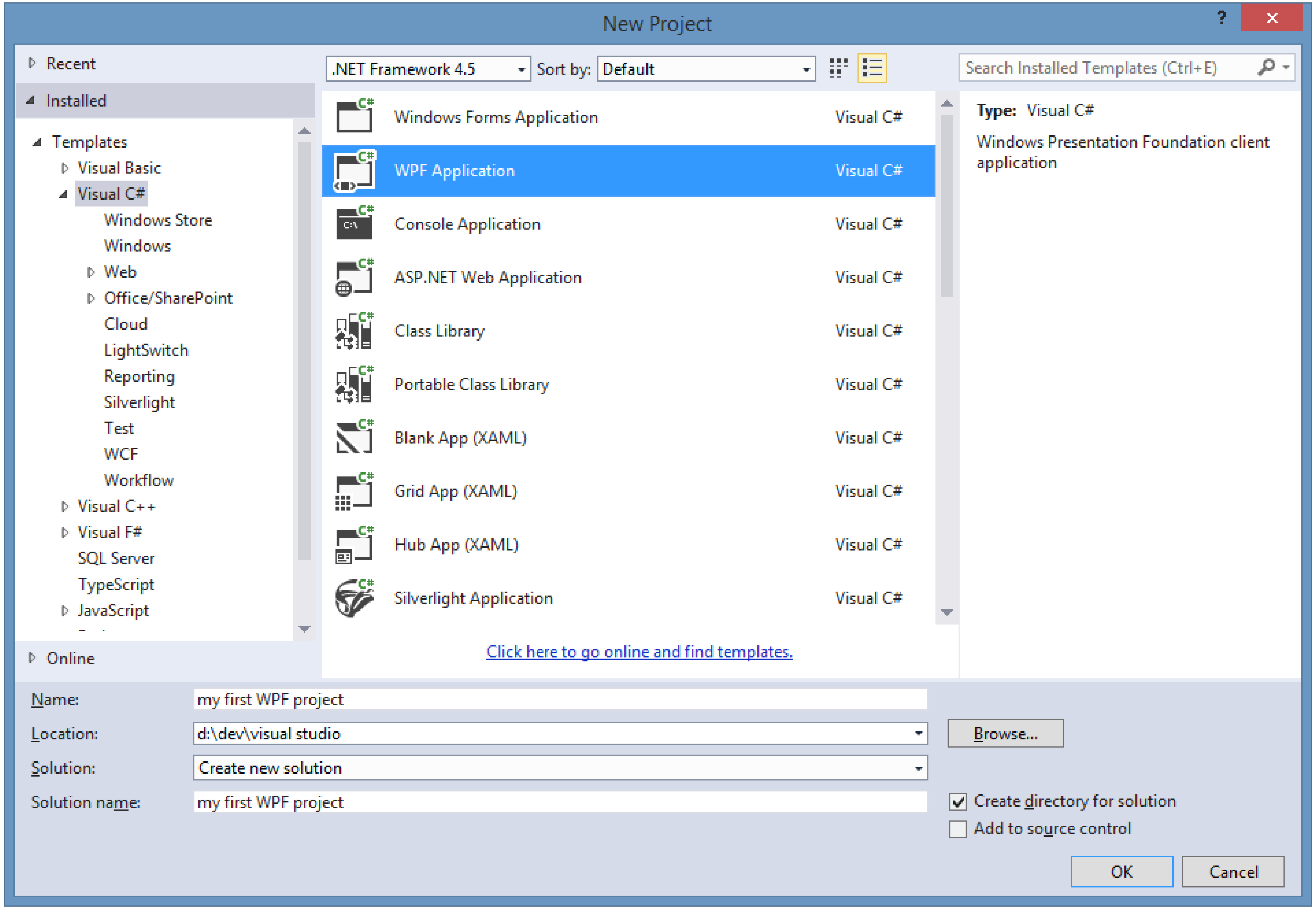

AvalonKayak - Windows Internet Explorer
E:\Documents and Settings\llavoie\Desktop\KayakalonXBAP\KayakalonXBAP\bin\Debug\KayakalonXBAP.xbap
MSN Search
AvalonKayak
Page
Tools
KAYAKALON
Flight Search
From
To
Leave
Return
Date
Time
Date
Time
Cabin
SEARCH
Create profile
Unknown Zone

Log in to load your personal profile

WPF tutorial

MAKING A BUTTON DO SOMETHING…

How to create WPF applications



New Project
Recent
Installed
Templates
Visual Basic
Visual C#
Windows Store
Windows
Web
Office/SharePoint
Cloud
LightSwitch
Reporting
Silverlight
Test
WCF
Workflow
Visual C++
Visual F#
SQL Server
TypeScript
JavaScript
Online
Sort by:
Search Installed Templates (Ctrl+E)
Windows Forms Application Visual C#
WPF Application Visual C#
Console Application Visual C#
ASP.NET Web Application Visual C#
Class Library Visual C#
Portable Class Library Visual C#
Blank App (XAML) Visual C#
Grid App (XAML) Visual C#
Hub App (XAML) Visual C#
Silverlight Application Visual C#
Click here to go online and find templates.
Browse...
OK
Cancel

.NET Framework 4.5

Sort by: Default

Type: Visual C#

Windows Presentation Foundation client application

Name: my first WPF project

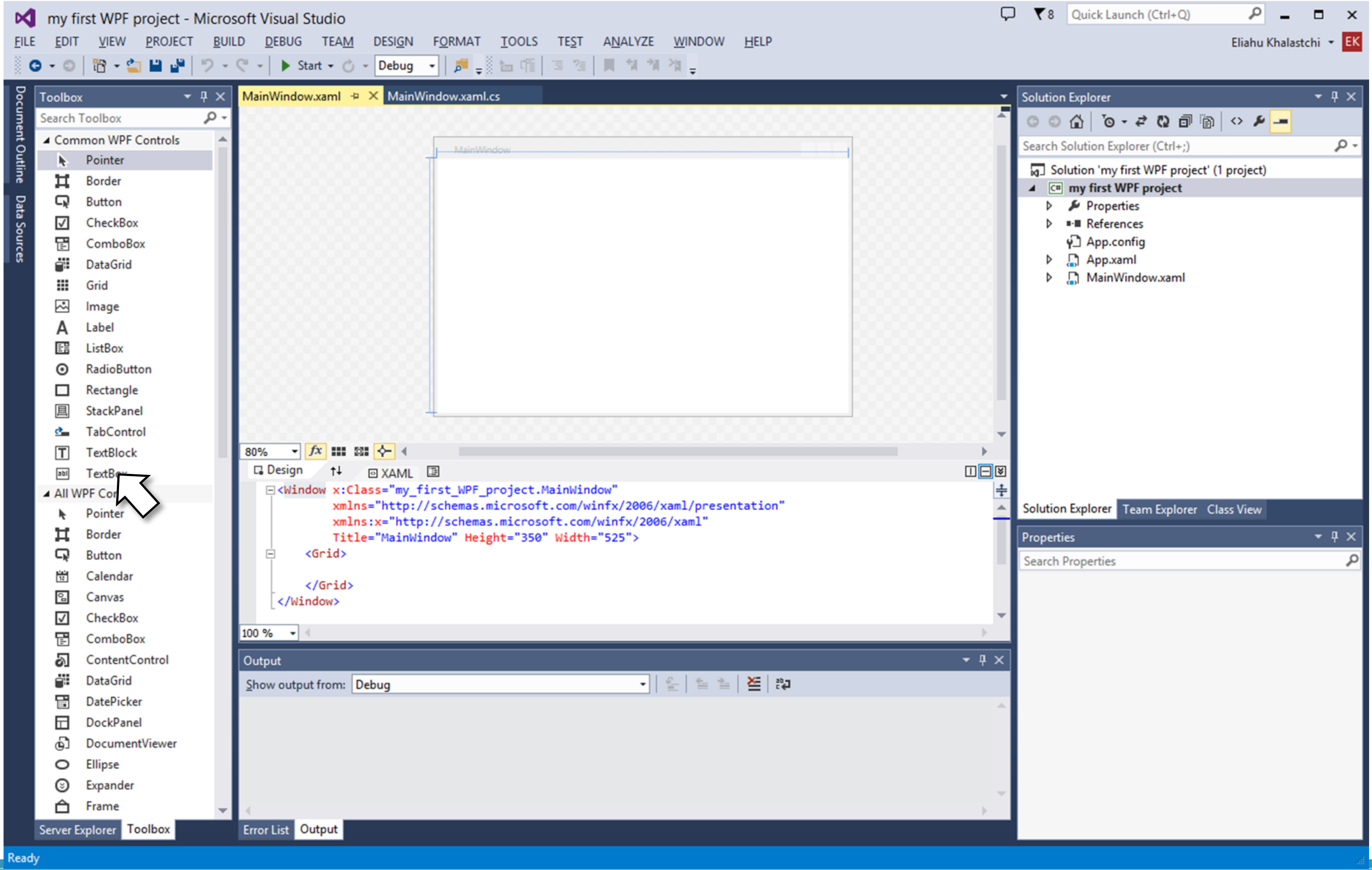
Location: d:\dev\visual studio

Solution: Create new solution

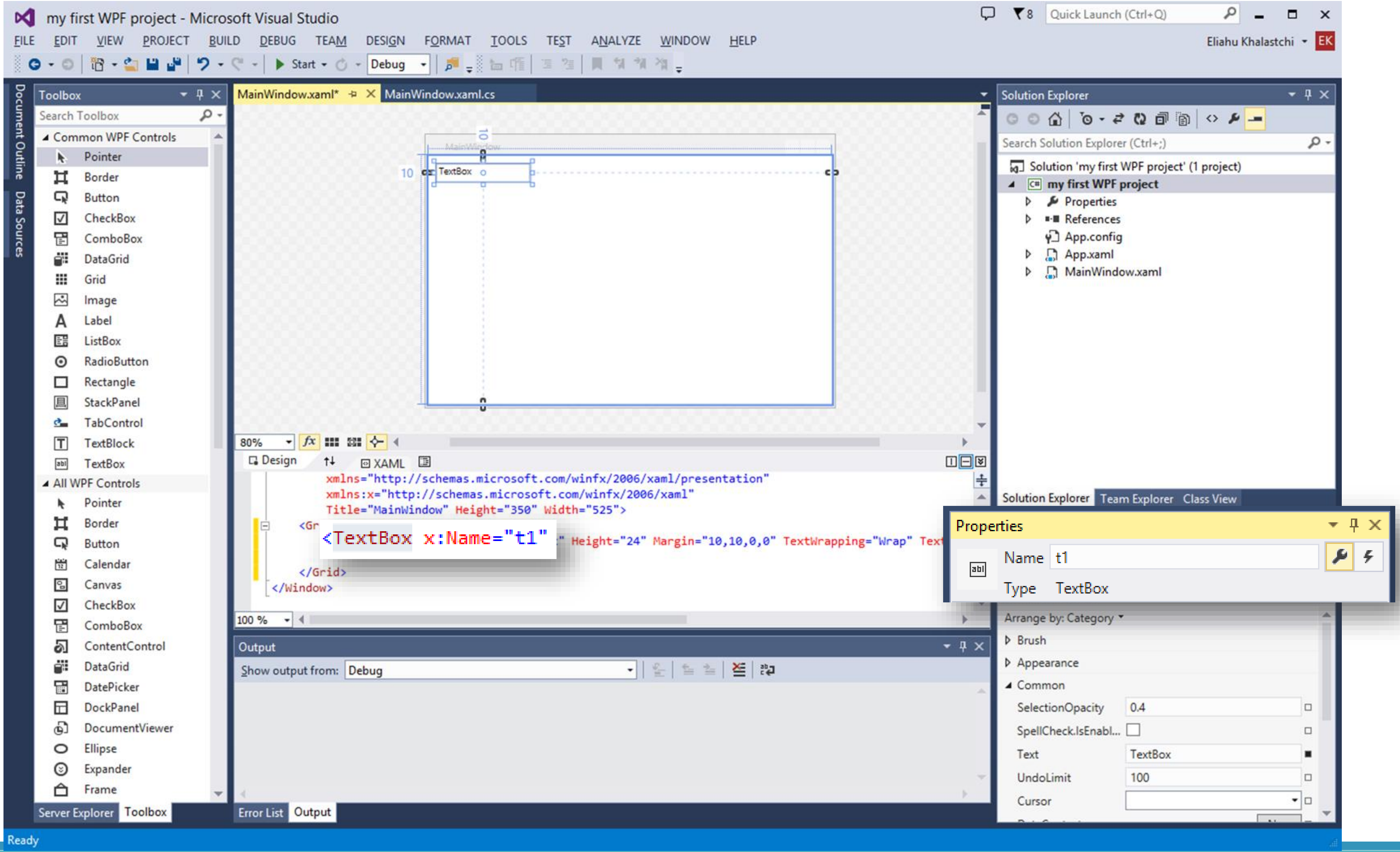
Solution name: my first WPF project

☑ Create directory for solution

□ Add to source control



```
<Window x:Class="my_first_WPF_project.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">
    <Grid>

    </Grid>
</Window>
```



```
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Title="MainWindow" Height="350" Width="525">
<Gr <TextBox x:Name="t1" " Height="24" Margin="10,10,0,0" TextWrapping="Wrap" Tex
    </Grid>
</Window>
```

Name t1

Type TextBox

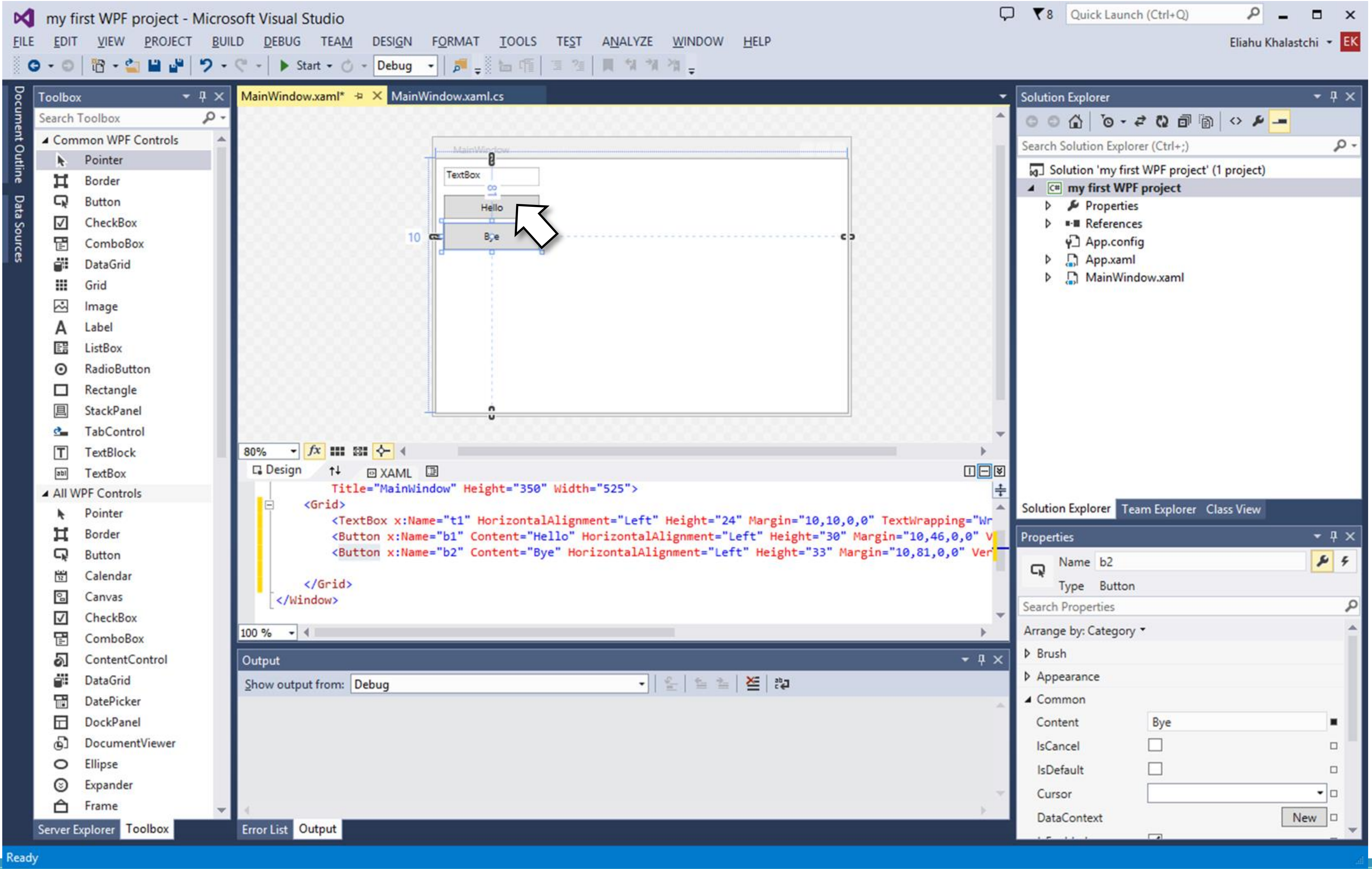
SelectionOpacity 0.4

SpellCheck.IsEnabl... □

Text TextBox

UndoLimit 100

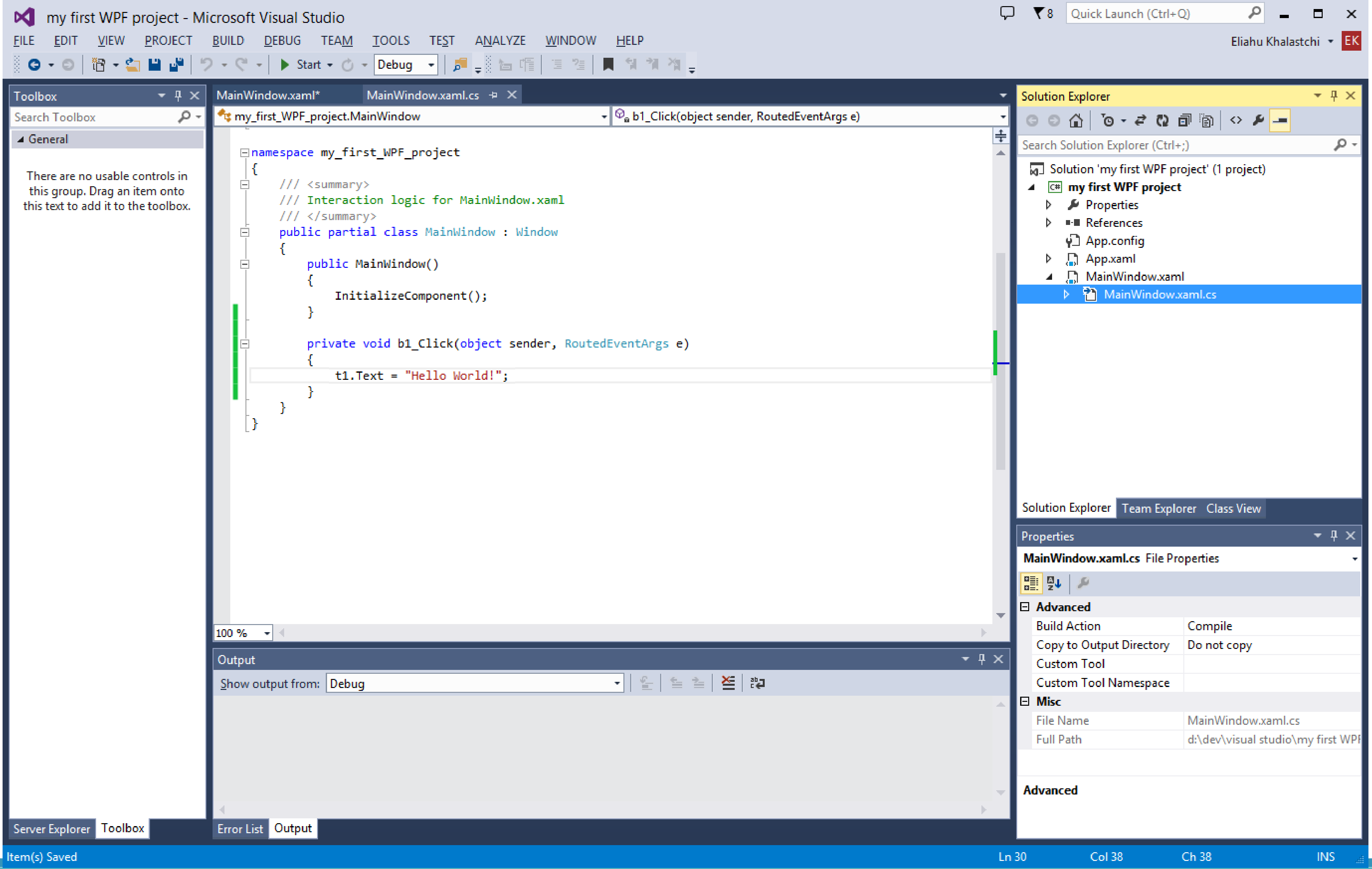
Cursor



```
        Title="MainWindow" Height="350" Width="525">
    <Grid>
        <TextBox x:Name="t1" HorizontalAlignment="Left" Height="24" Margin="10,10,0,0" TextWrapping="Wr
        <Button x:Name="b1" Content="Hello" HorizontalAlignment="Left" Height="30" Margin="10,46,0,0" V
        <Button x:Name="b2" Content="Bye" HorizontalAlignment="Left" Height="33" Margin="10,81,0,0" Ver

    </Grid>
</Window>
```

Name b2
Type Button

Content Bye
IsCancel □
IsDefault □
Cursor
DataContext



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

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

        private void b1_Click(object sender, RoutedEventArgs e)
        {
            t1.Text = "Hello World!";
        }
    }
}
```

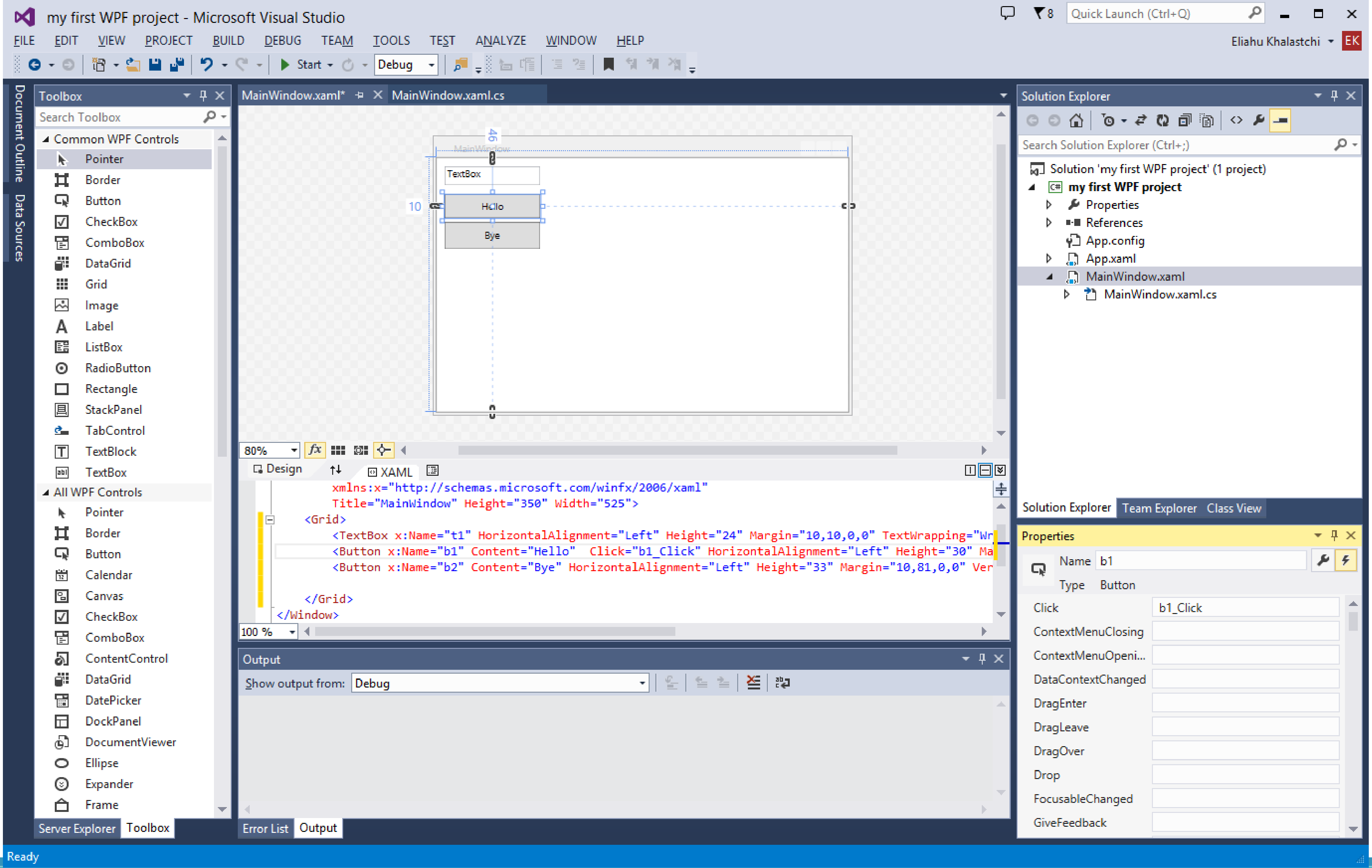
Show output from: Debug

MainWindow.xaml.cs File Properties

Advanced	
Build Action	Compile
Copy to Output Directory	Do not copy
Custom Tool	
Custom Tool Namespace	
Misc	
File Name	MainWindow.xaml.cs
Full Path	d:\dev\visual studio\my first WPF

Advanced

Item(s) Saved   Ln 30   Col 38   Ch 38   INS



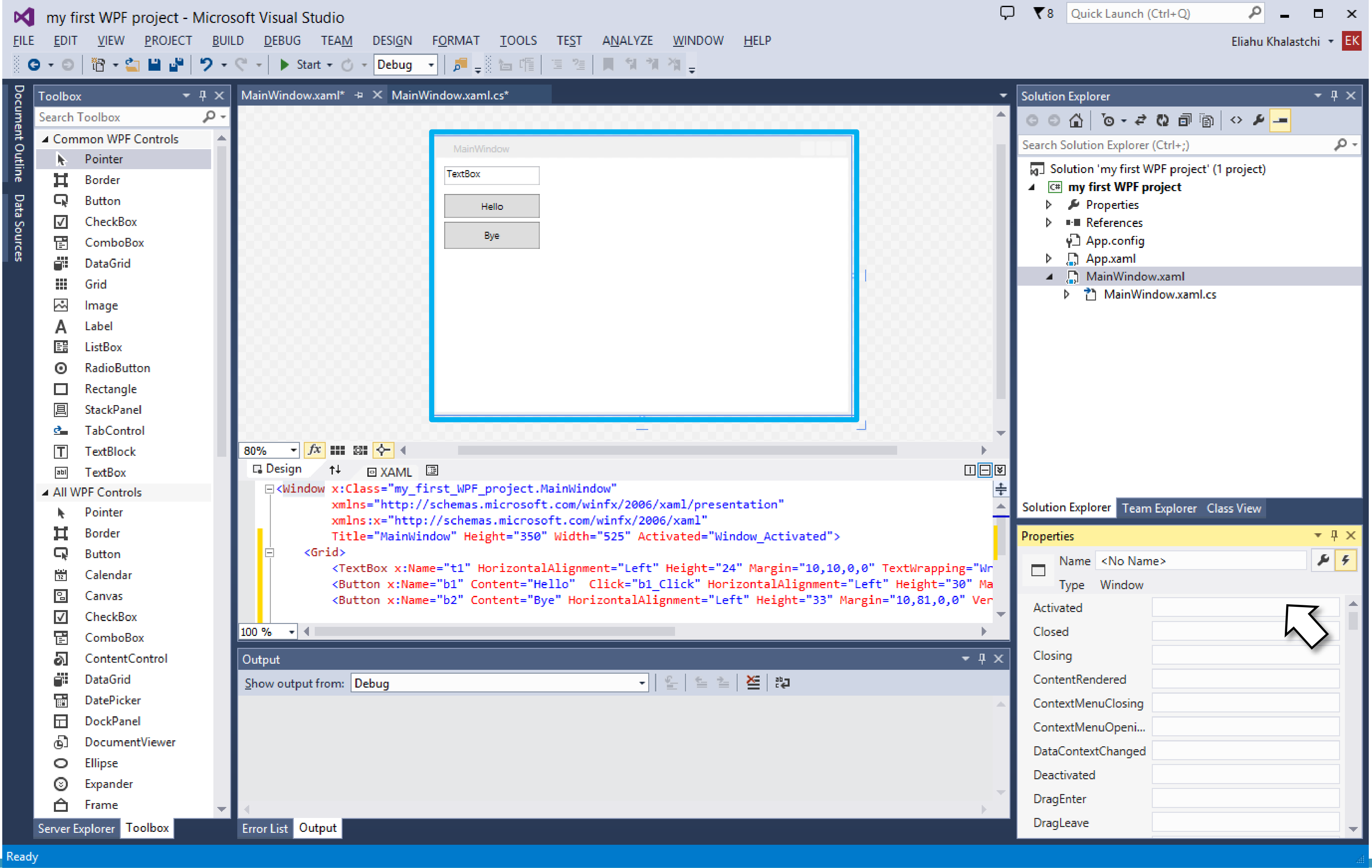
```
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
    <Grid>
        <TextBox x:Name="t1" HorizontalAlignment="Left" Height="24" Margin="10,10,0,0" TextWrapping="Wr
        <Button x:Name="b1" Content="Hello"  Click="b1_Click" HorizontalAlignment="Left" Height="30" Ma
        <Button x:Name="b2" Content="Bye" HorizontalAlignment="Left" Height="33" Margin="10,81,0,0" Ver

    </Grid>
</Window>
```

Show output from: Debug

Name b1

Type Button

Click	b1_Click
ContextMenuClosing	
ContextMenuOpeni...	
DataContextChanged	
DragEnter	
DragLeave	
DragOver	
Drop	
FocusableChanged	
GiveFeedback	



my first WPF project - Microsoft Visual Studio

```
<Window x:Class="my_first_WPF_project.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" Activated="Window_Activated">
    <Grid>
        <TextBox x:Name="t1" HorizontalAlignment="Left" Height="24" Margin="10,10,0,0" TextWrapping="Wr
        <Button x:Name="b1" Content="Hello"  Click="b1_Click" HorizontalAlignment="Left" Height="30" Ma
        <Button x:Name="b2" Content="Bye" HorizontalAlignment="Left" Height="33" Margin="10,81,0,0" Ver
```

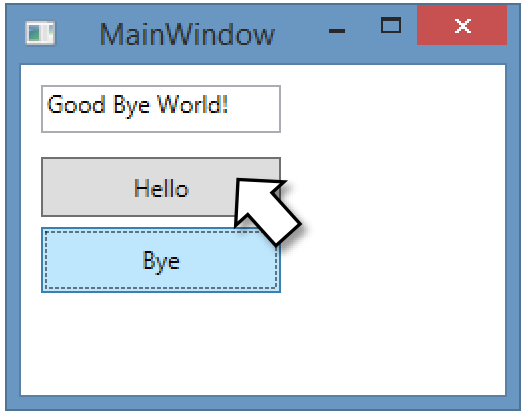
Name <No Name>

Type Window

Show output from: Debug

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

        private void b1_Click(object sender, RoutedEventArgs e)
        {
            t1.Text = "Hello World!";
        }

        private void Window_Activated(object sender, EventArgs e)
        {
            b2.Click += b2_Click;
        }

        void b2_Click(object sender, RoutedEventArgs e)
        {
            t1.Text = "Good Bye World!";
        }
    }
```

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

        private void b1_Click(object sender, RoutedEventArgs e)
        {
            t1.Text = "Hello World!";
        }

        private void Window_Activated(object sender, EventArgs e)
        {
            b2.Click += b2_Click;
        }

        void b2_Click(object sender, RoutedEventArgs e)
        {
            t1.Text = "Good Bye World!";
        }
    }
```



Good Bye World!

```
<TextBox x:Name="t1" ... />
<Button x:Name="b1" Content="Hello“
              Click="b1_Click" .../>
<Button x:Name="b2" Content="Bye".../>
```

Partial Classes

- It is possible to split a class, a struct, an interface, or a method over two or more source files
- In large projects, multiple programmers can work in the same time on the same class
- Automatically generated source + your code of the same class, without a problem…
- Our MainWindow class is partially edited by the visual studio

```
public partial class Employee {
  public void DoWork() { }
}

//... In another source file

public partial class Employee {
  public void GoToLunch() { }
}
```

XAML

What is XAML?

- Extensible Application Markup Language
- XAML is a declarative markup language
- XAML simplifies creating a UI for a .NET Framework application
- Separates the UI definition from the run-time logic by using code-behind files
- XAML enables a workflow where
  - separate parties can work on the UI and the logic of an application
  - using potentially different tools
- XAML are XML files with .xaml extenssion

XAML – Extensible Application Markup Language

- XAML can create objects, set Properties, and connect to events
- XAML cannot call methods,
  - for this we have the code-behind – to handle events and change items dynamically

```
<Window x:Class="my_first_WPF_project.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" Activated="Window_Activated">
    <Grid>
        <TextBox x:Name="t1" HorizontalAlignment="Left" Height="24" Margin="10,10,0,0" TextWrapping="Wrap"
              Text="TextBox" VerticalAlignment="Top" Width="120"/>
        <Button x:Name="b1" Content="Hello"  Click="b1_Click" HorizontalAlignment="Left" Height="30"
              Margin="10,46,0,0" VerticalAlignment="Top" Width="120"/>
        <Button x:Name="b2" Content="Bye" HorizontalAlignment="Left" Height="33" Margin="10,81,0,0"
              VerticalAlignment="Top" Width="120"/>
    </Grid>
</Window>
```

XAML Syntax - Object Elements & Properties

- XAML object elements declares an instance of a type
- Use the *Attribute Syntax* to set the properties of an object
- When *Attribute Syntax* is not possible – use *Property Element Syntax*

```
<StackPanel>
  <Button Content="Click Me"/>
</StackPanel>
```

2 xml elements instantiates a **StackPanel** and a **Button**

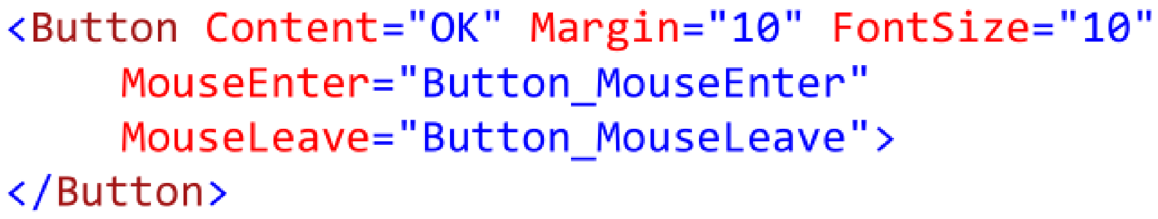
```
<Button Background="Blue" Foreground="Red" Content="This is a button"/>
```

```
<Button>
  <Button.Background>
    <SolidColorBrush Color="Blue"/>
  </Button.Background>
  <Button.Foreground>
    <SolidColorBrush Color="Red"/>
  </Button.Foreground>
  <Button.Content>
    This is a button
  </Button.Content>
</Button>
```

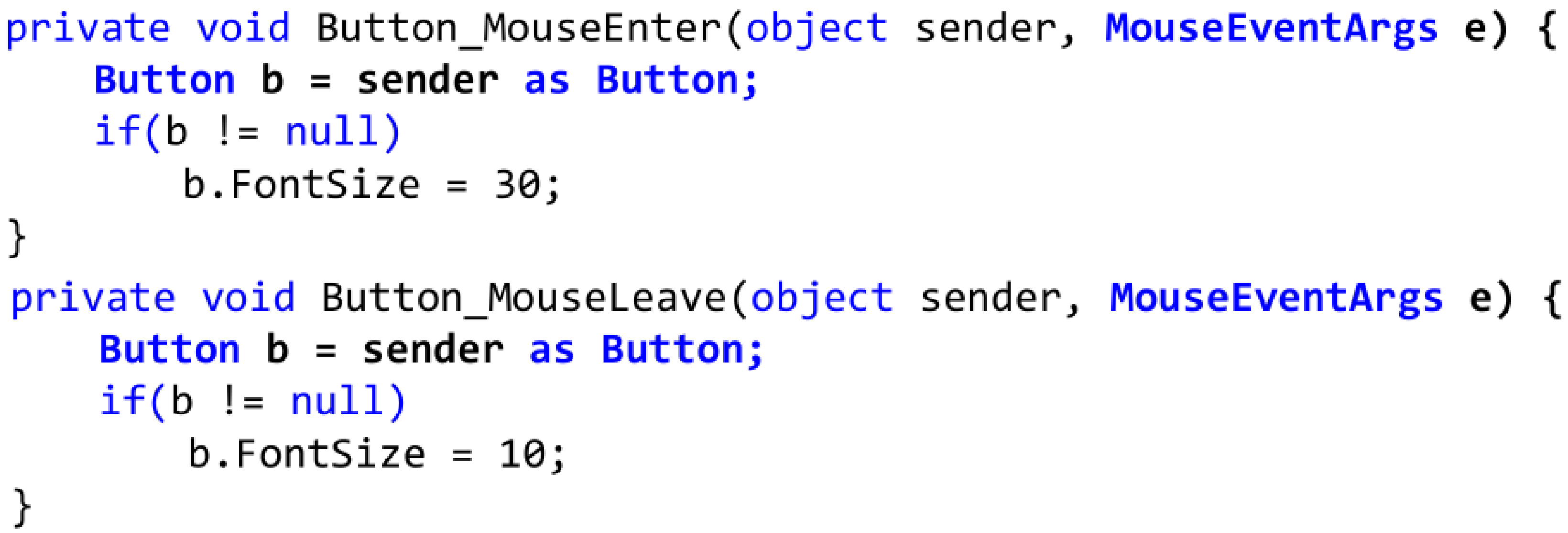
Attribute syntax can also be used for members that are **events** rather than properties:

```
<Button Click="Button_Click" />
```

Replacing code behind with Property Triggers



```xml
<Button Content="OK" Margin="10" FontSize="10"
     MouseEnter="Button_MouseEnter"
     MouseLeave="Button_MouseLeave">
</Button>
```



```csharp
private void Button_MouseEnter(object sender, MouseEventArgs e) {
    Button b = sender as Button;
    if(b != null)
        b.FontSize = 30;
}
private void Button_MouseLeave(object sender, MouseEventArgs e) {
    Button b = sender as Button;
    if(b != null)
        b.FontSize = 10;
}
```

Replacing code behind with Property Triggers

Must be wrapped in a style

No need to revert the property back

```
<Button Content="OK" Margin="10">
    <Button.Style>
        <Style TargetType="Button">
            <Style.Triggers>
                <Trigger Property="IsMouseOver" Value="True">
                    <Setter Property="FontSize" Value="30" />
                </Trigger>
            </Style.Triggers>
        </Style>
    </Button.Style>
</Button>
```

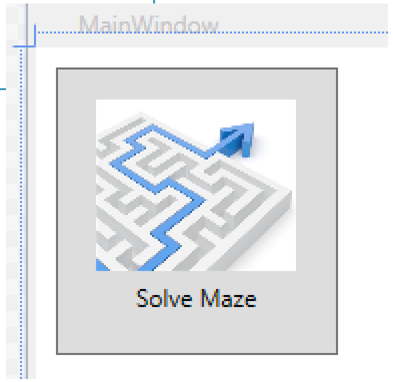
Custom WPF Controls

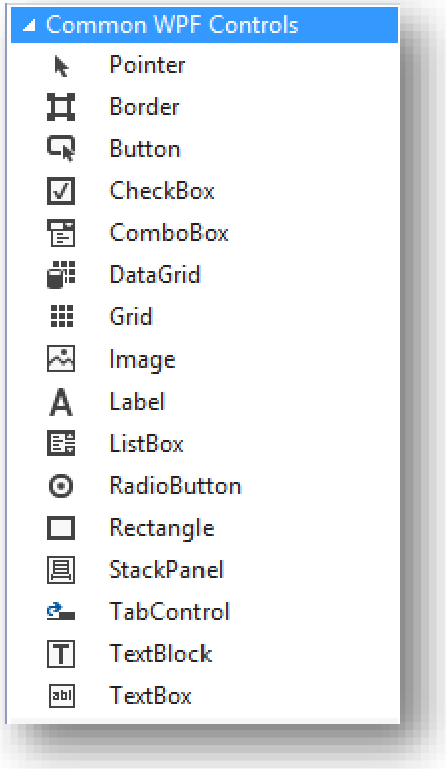
A TUTORIAL

What are controls?

- Controls: elements capable of receiving focus and handling input
- Many controls are available “out of the box”
- Custom controls can be created
  - User controls that **wrap** one or more controls and expose higher level properties
  - Custom controls that **derive** from an existing control and extend its functionality

```
<Button x:Name="solveMaze" Margin="10,10,366,165">
     <StackPanel Orientation="Vertical">
          <Image Source="resources\Maze.jpg" Width="100" />
          <Label Content="Solve Maze"
                   HorizontalContentAlignment="Center"/>
     </StackPanel>
</Button>
```





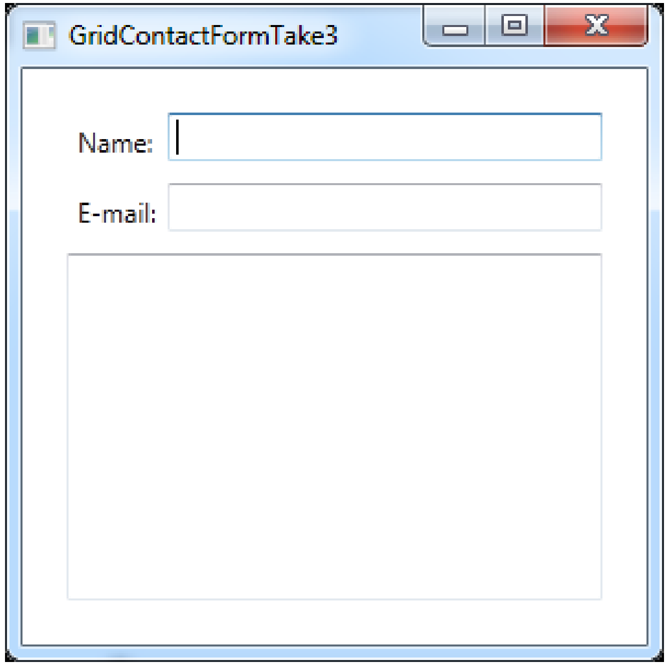
Layouts

- Panel elements control the rendering of their child elements
  - their size and dimensions, their position, and the arrangement

Panel name	Description
Canvas	Defines an area within which you can explicitly position child elements by coordinates relative to the Canvas area.
DockPanel	Defines an area within which you can arrange child elements either horizontally or vertically, relative to each other.
Grid	Defines a flexible grid area that consists of columns and rows.
StackPanel	Arranges child elements into a single line that can be oriented horizontally or vertically.
WrapPanel	Positions child elements in sequential position from left to right, breaking content to the next line at the edge of the containing box. Subsequent ordering occurs sequentially from top to bottom or right to left, depending on the value of the Orientation property.

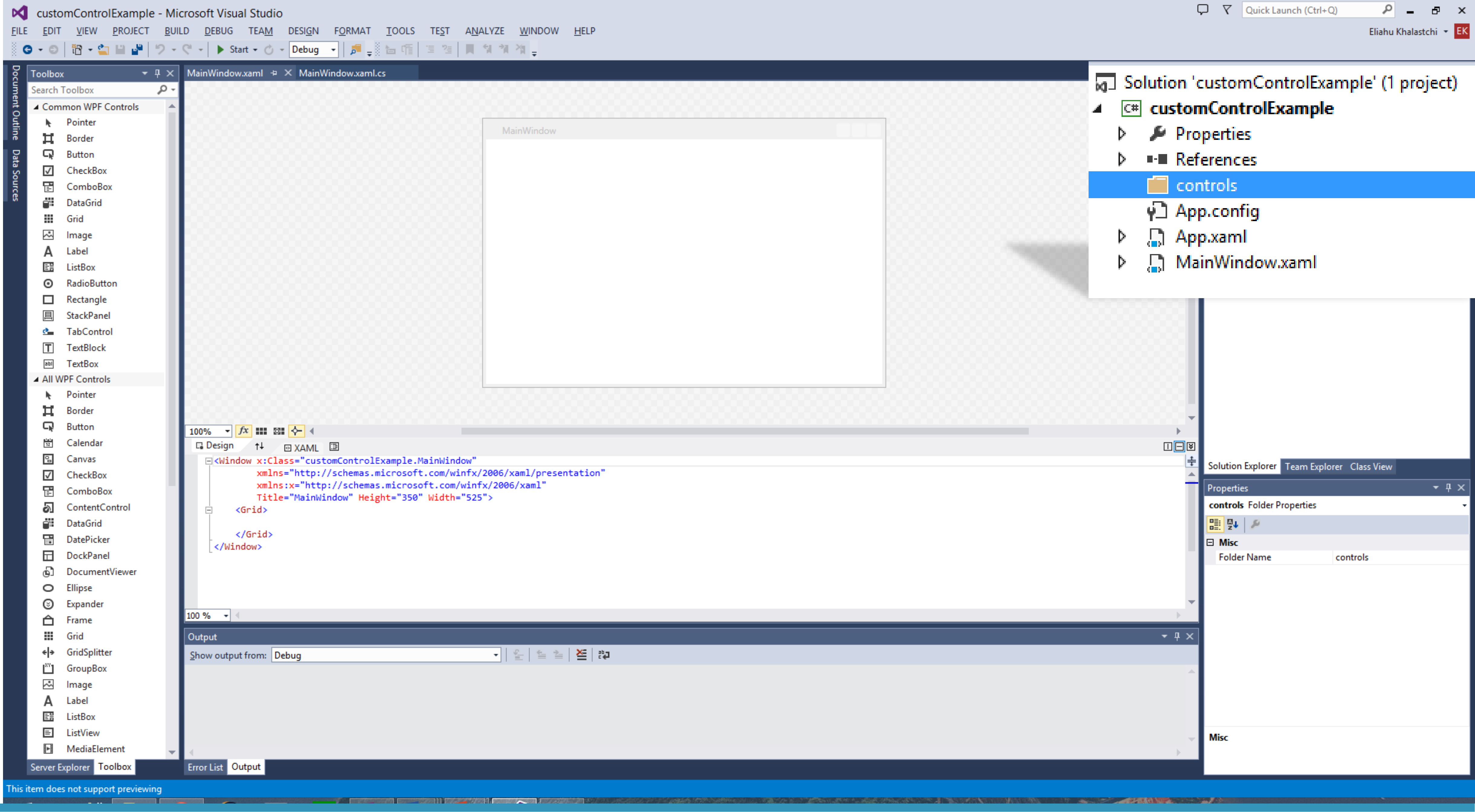
Grid

```
<Grid Margin="10">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto" />
        <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto" />
        <RowDefinition Height="Auto" />
        <RowDefinition Height="*" />
    </Grid.RowDefinitions>

    <Label>Name:</Label>
    <TextBox Grid.Column="1" Margin="0,0,0,10" />
    <Label Grid.Row="1">E-mail:</Label>
    <TextBox Grid.Row="1" Grid.Column="1" Margin="0,0,0,10" />
    <TextBox Grid.ColumnSpan="2" Grid.Row="2" AcceptsReturn="True" />
</Grid>
```



Example

TILE PUZZLE



Solution 'customControlExample' (1 project)
- customControlExample
  - Properties
  - References
  - controls
  - App.config
  - App.xaml
  - MainWindow.xaml

```
<Window x:Class="customControlExample.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">
    <Grid>

    </Grid>
</Window>
```

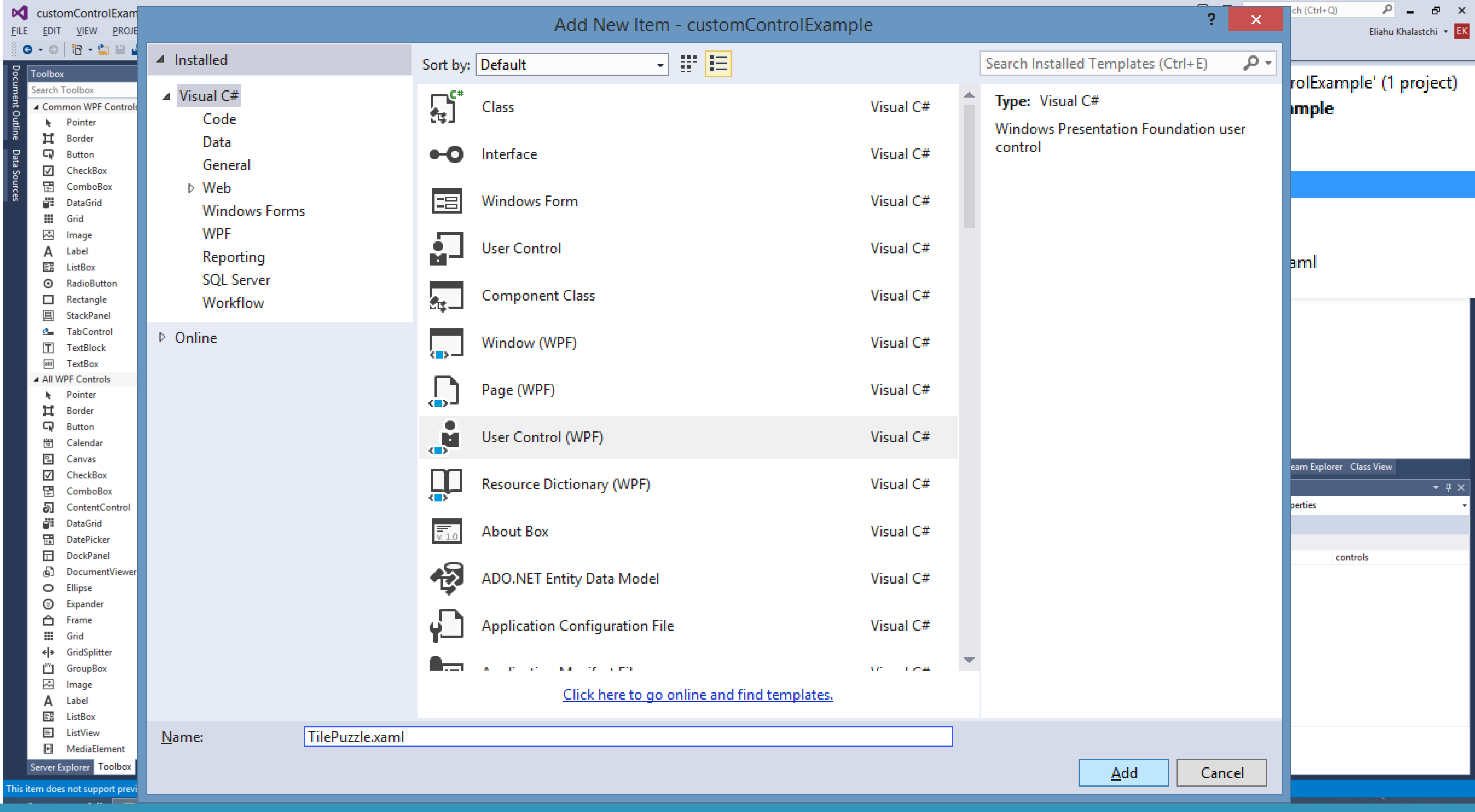
100 %

Show output from: Debug

controls Folder Properties

Folder Name	controls

This item does not support previewing



Add New Item - customControlExample

Sort by: Default

Search Installed Templates (Ctrl+E)

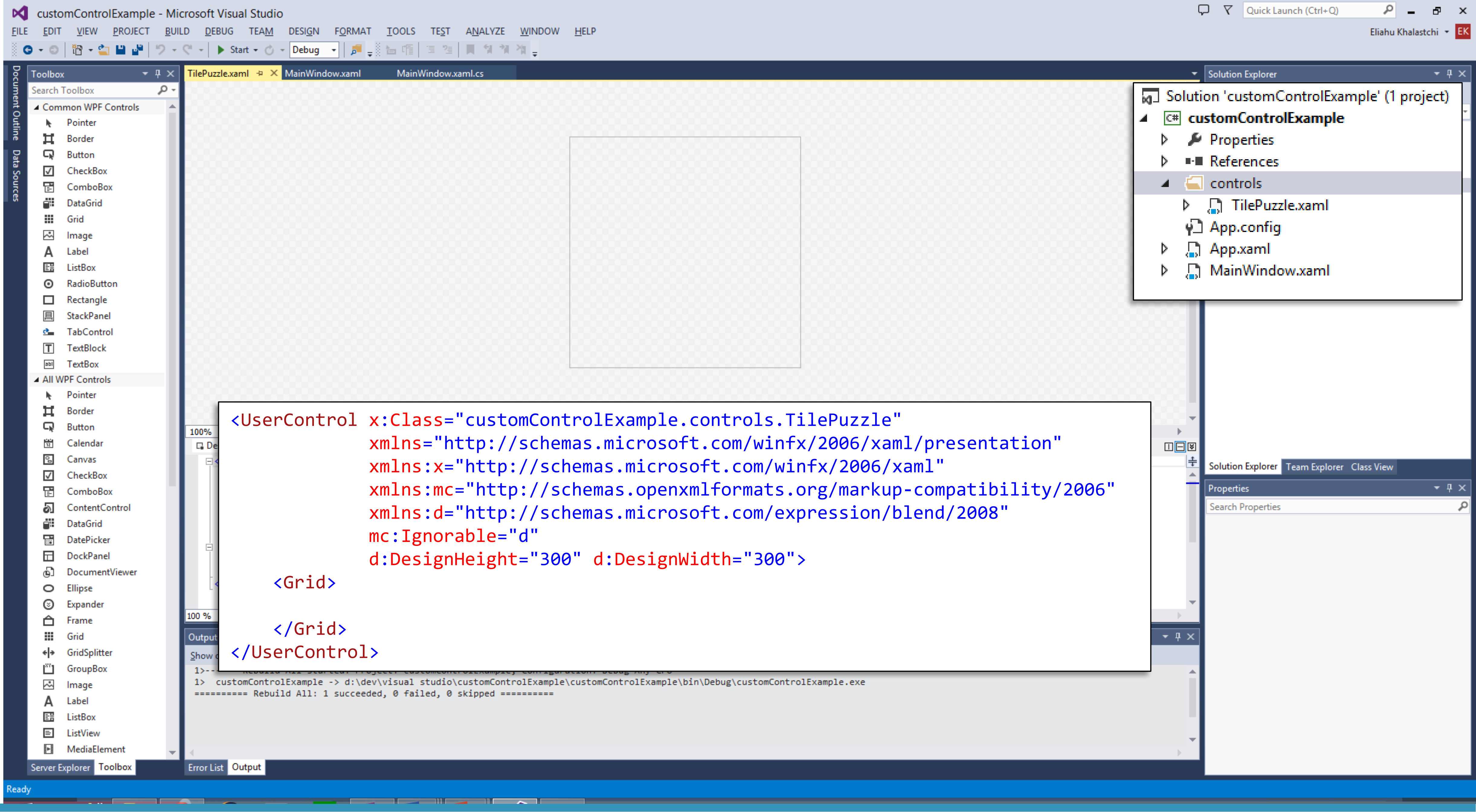
Class	Visual C#
Interface	Visual C#
Windows Form	Visual C#
User Control	Visual C#
Component Class	Visual C#
Window (WPF)	Visual C#
Page (WPF)	Visual C#
User Control (WPF)	Visual C#
Resource Dictionary (WPF)	Visual C#
About Box	Visual C#
ADO.NET Entity Data Model	Visual C#
Application Configuration File	Visual C#

Type: Visual C#

Windows Presentation Foundation user control

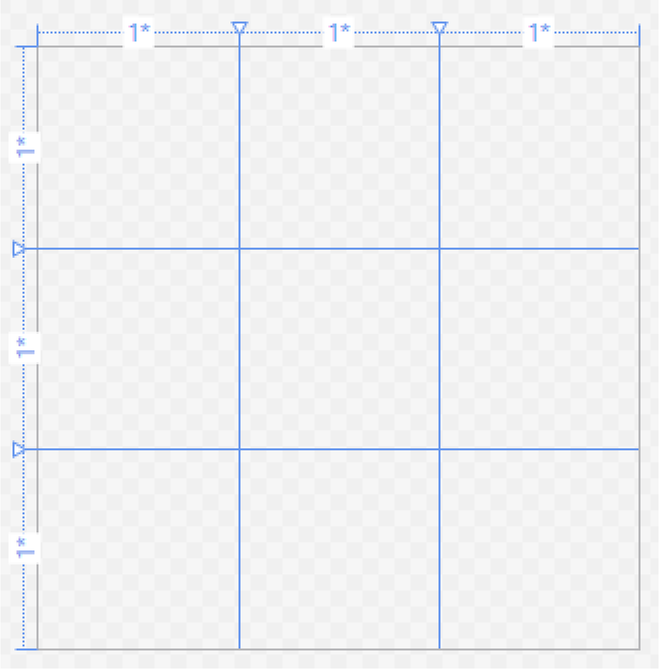
Click here to go online and find templates.

Name: TilePuzzle.xaml

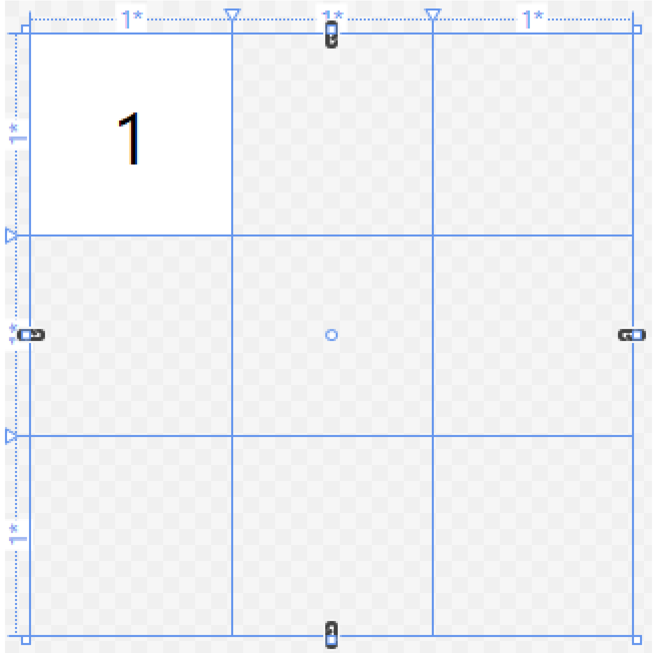


```
<UserControl x:Class="customControlExample.controls.TilePuzzle"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             d:DesignHeight="300"  d:DesignWidth="300">
    <Grid>

    </Grid>
</UserControl>
```

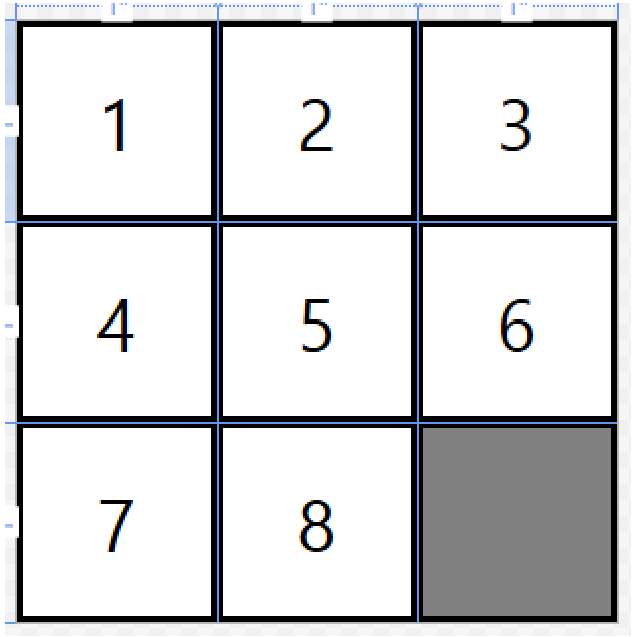
```
1>  customControlExample -> d:\dev\visual studio\customControlExample\customControlExample\bin\Debug\customControlExample.exe
========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========
 <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>

</Grid>
```


1*
1*
1*
1*
1*
1*

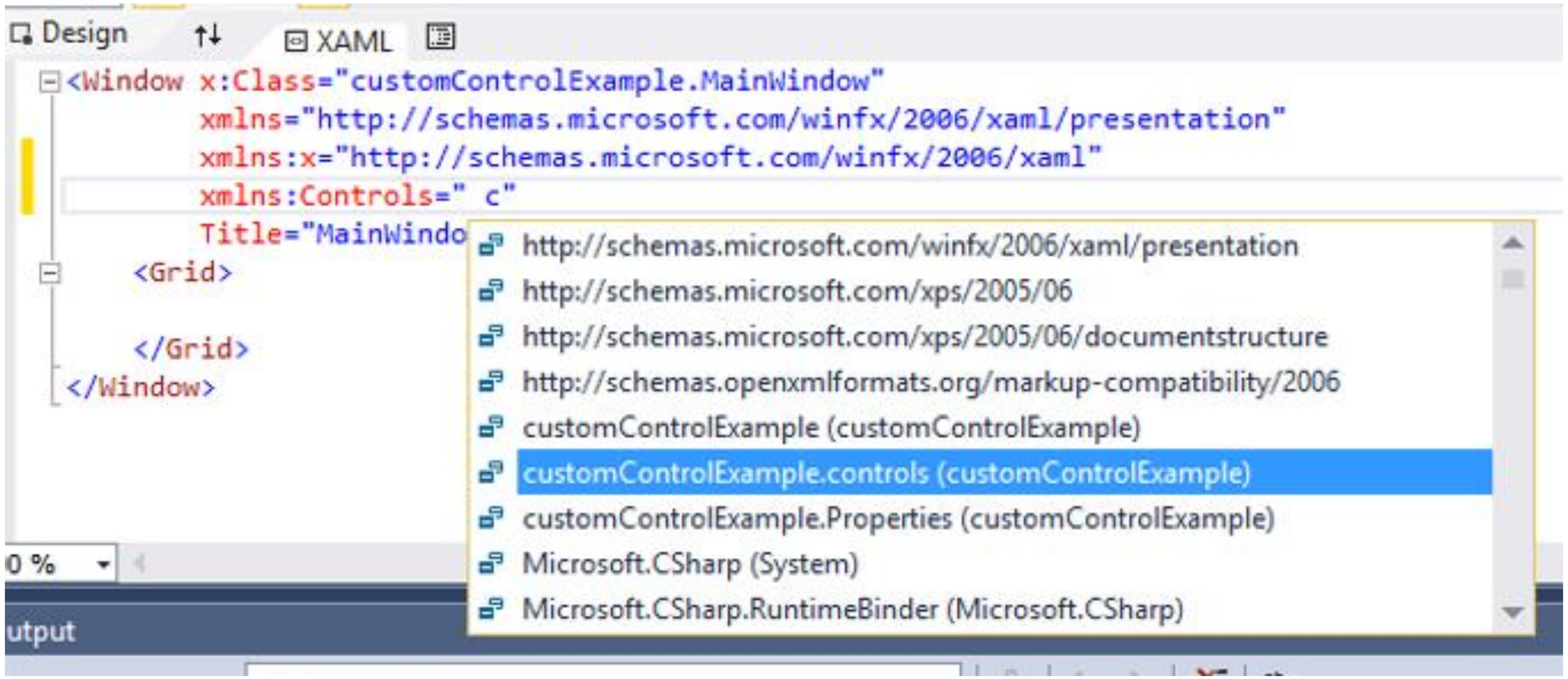
```
 <Grid>
       <Grid.RowDefinitions>
           <RowDefinition Height="*"/>
           <RowDefinition Height="*"/>
           <RowDefinition Height="*"/>
       </Grid.RowDefinitions>

       <Grid.ColumnDefinitions>
           <ColumnDefinition Width="*"/>
           <ColumnDefinition Width="*"/>
           <ColumnDefinition Width="*"/>
       </Grid.ColumnDefinitions>

    <Label Content="1" Background="White"
        BorderBrush="Black" BorderThickness="3" FontSize="36"
        HorizontalContentAlignment="Center"
        VerticalContentAlignment="Center"/>
</Grid>
```



```
 <Grid>
        <Grid.RowDefinitions>
                ...
         </Grid.RowDefinitions>

        <Grid.ColumnDefinitions>
                ...
         </Grid.ColumnDefinitions>


<Label x:Name="l1" Content="1" Grid.Row="0" Grid.Column="0" .../>
<Label x:Name="l2" Content="2" Grid.Row="0" Grid.Column="1" .../>
<Label x:Name="l3" Content="3" Grid.Row="0" Grid.Column="2" .../>
<Label x:Name="l4" Content="4" Grid.Row="1" Grid.Column="0" .../>
<Label x:Name="l5" Content="5" Grid.Row="1" Grid.Column="1" .../>
<Label x:Name="l6" Content="6" Grid.Row="1" Grid.Column="2" .../>
<Label x:Name="l7" Content="7" Grid.Row="2" Grid.Column="0" .../>
<Label x:Name="l8" Content="8" Grid.Row="2" Grid.Column="1" .../>
<Label x:Name="l9" Content=" " Grid.Row="2" Grid.Column="2" Background="Gray" .../>

</Grid>
```

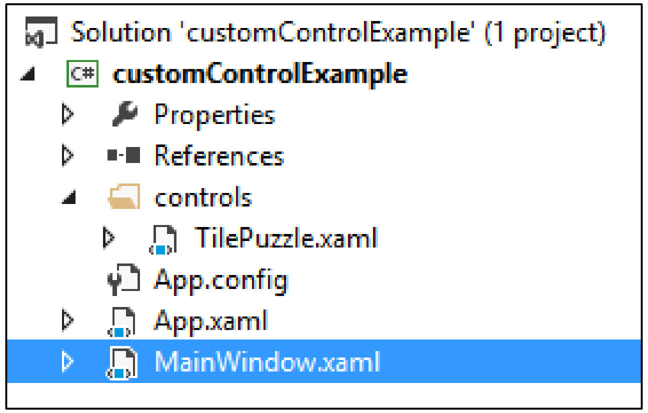


Now, let’s add this control to our window…



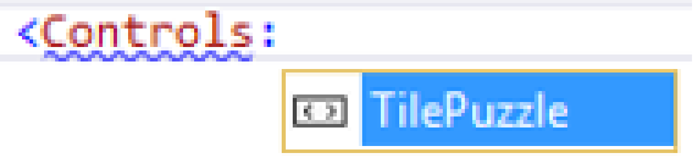
```
<Window x:Class="customControlExample.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:Controls=" c"
        Title="MainWindo
    <Grid>

    </Grid>
</Window>
```

```
http://schemas.microsoft.com/winfx/2006/xaml/presentation
http://schemas.microsoft.com/xps/2005/06
http://schemas.microsoft.com/xps/2005/06/documentstructure
http://schemas.openxmlformats.org/markup-compatibility/2006
customControlExample (customControlExample)
customControlExample.controls (customControlExample)
customControlExample.Properties (customControlExample)
Microsoft.CSharp (System)
Microsoft.CSharp.RuntimeBinder (Microsoft.CSharp)
```



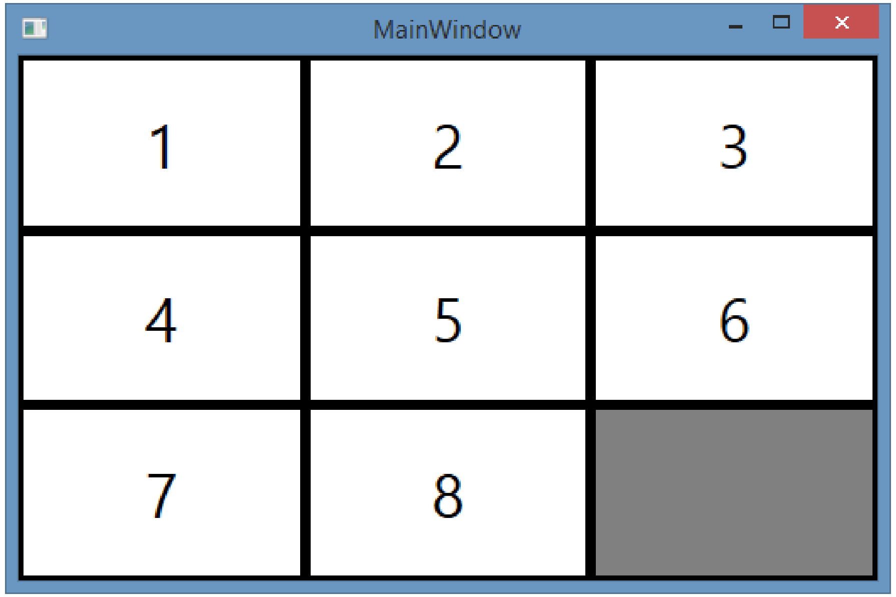
```
Solution 'customControlExample' (1 project)
  customControlExample
    Properties
    References
    controls
      TilePuzzle.xaml
    App.config
    App.xaml
    MainWindow.xaml
```

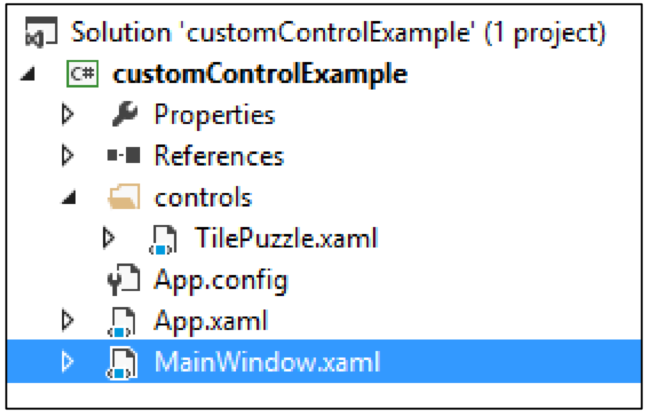
```
<Window x:Class="customControlExample.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:Controls="clr-namespace:customControlExample.controls"
        Title="MainWindow" Height="350" Width="525">
    <Grid>
```



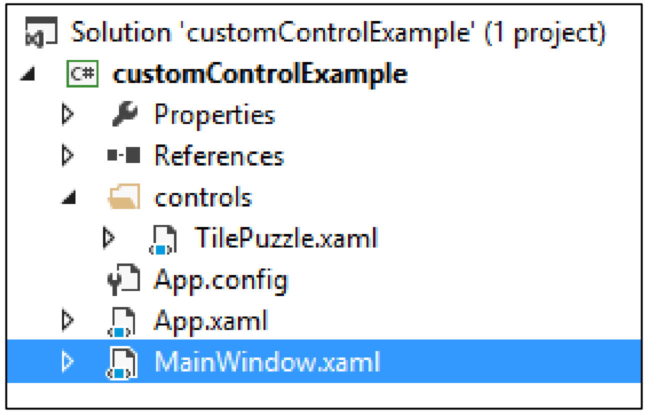
```
<Controls:
TilePuzzle
```

```
    </Grid>
</Window>
```



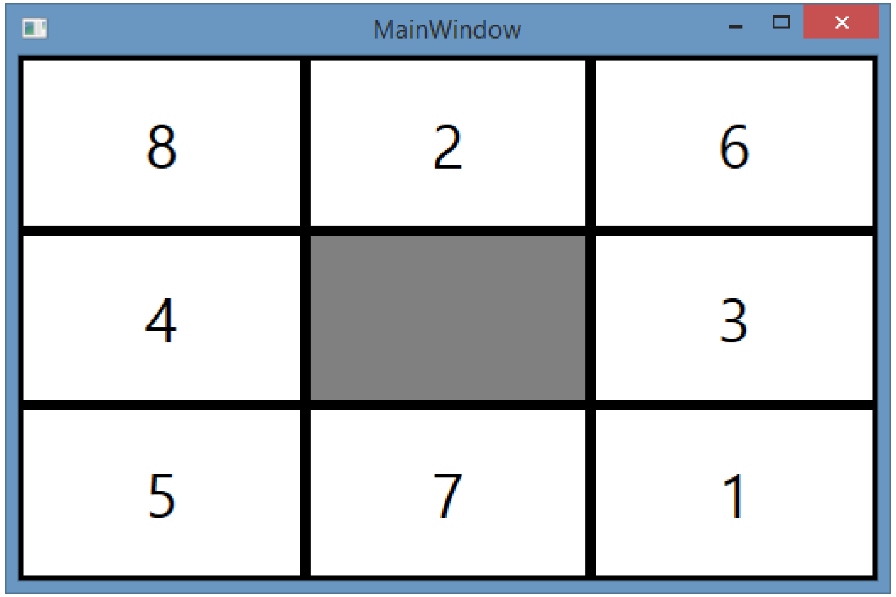


```
<Window x:Class="customControlExample.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:Controls="clr-namespace:customControlExample.controls"
        Title="MainWindow" Height="350" Width="525">
    <Grid>
        <Controls:TilePuzzle />
    </Grid>
</Window>
```



```
Solution 'customControlExample' (1 project)
◢ customControlExample
    ▷ Properties
    ▷ References
    ◢ controls
        ▷ TilePuzzle.xaml
      App.config
    ▷ App.xaml
    ▷ MainWindow.xaml
```

We want to be able to do this:

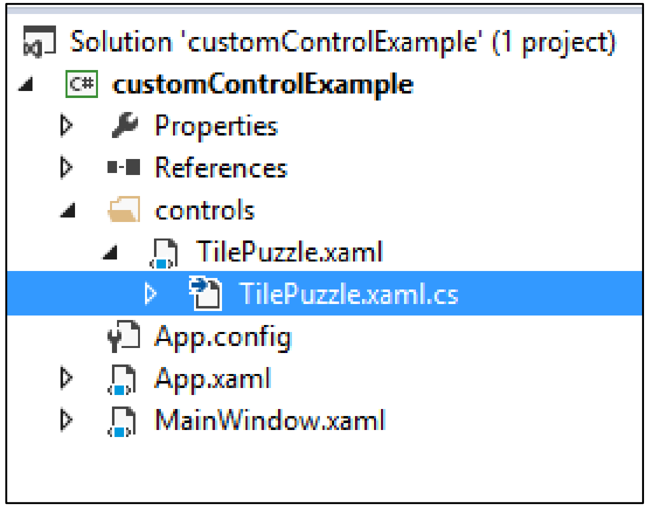


MainWindow

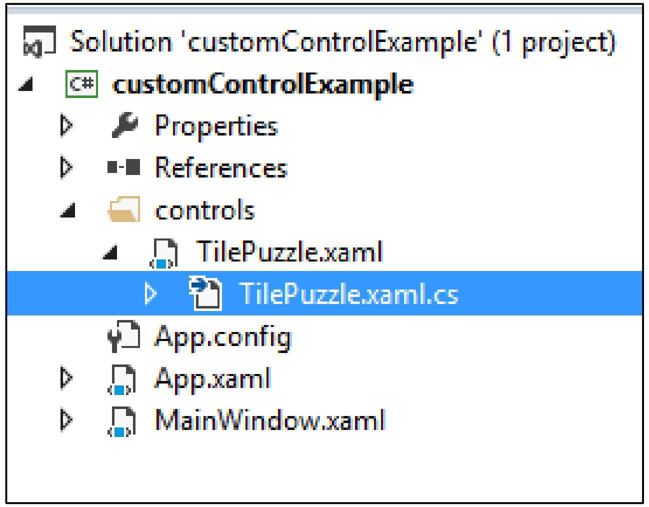
8	2	6
4		3
5	7	1

```
<Window x:Class="customControlExample.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:Controls="clr-namespace:customControlExample.controls"
        Title="MainWindow" Height="350" Width="525">
    <Grid>
        <Controls:TilePuzzle Order="8264 3571"/>
    </Grid>
</Window>
```

```csharp
namespace customControlExample.controls
{
    /// <summary>
    /// Interaction logic for TilePuzzle.xaml
    /// </summary>
    public partial class TilePuzzle : UserControl
    {
        public TilePuzzle()
        {
            InitializeComponent();
        }

    }
}
```



Solution 'customControlExample' (1 project)
- customControlExample
  - Properties
  - References
  - controls
    - TilePuzzle.xaml
      - TilePuzzle.xaml.cs
  - App.config
  - App.xaml
  - MainWindow.xaml

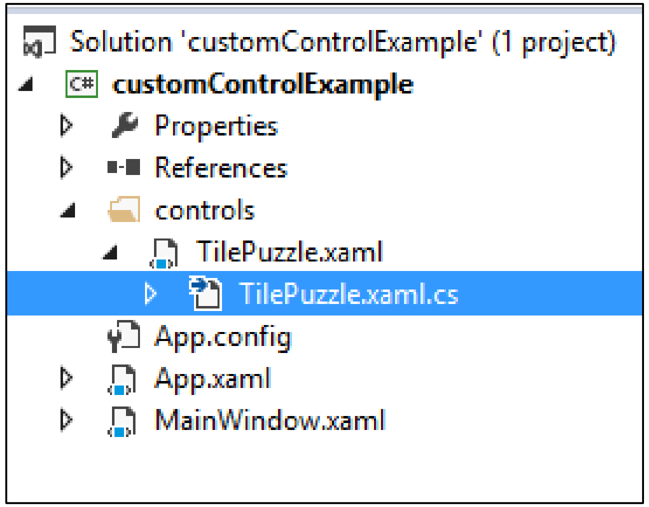


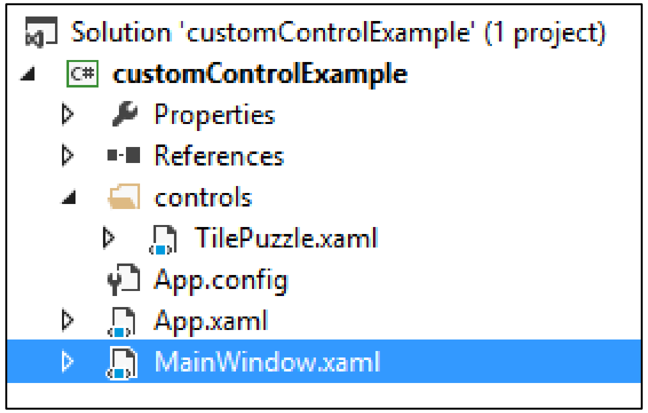
- Solution 'customControlExample' (1 project)
  - customControlExample
    - Properties
    - References
    - controls
      - TilePuzzle.xaml
        - TilePuzzle.xaml.cs
    - App.config
    - App.xaml
    - MainWindow.xaml

```
public partial class TilePuzzle : UserControl
{
        Label[] tiles;
        public TilePuzzle()
        {
            InitializeComponent();
            tiles = new Label[9];
            tiles[0] = l1;
            tiles[1] = l2;
            tiles[2] = l3;
            tiles[3] = l4;
            tiles[4] = l5;
            tiles[5] = l6;
            tiles[6] = l7;
            tiles[7] = l8;
            tiles[8] = l9;
        }
```

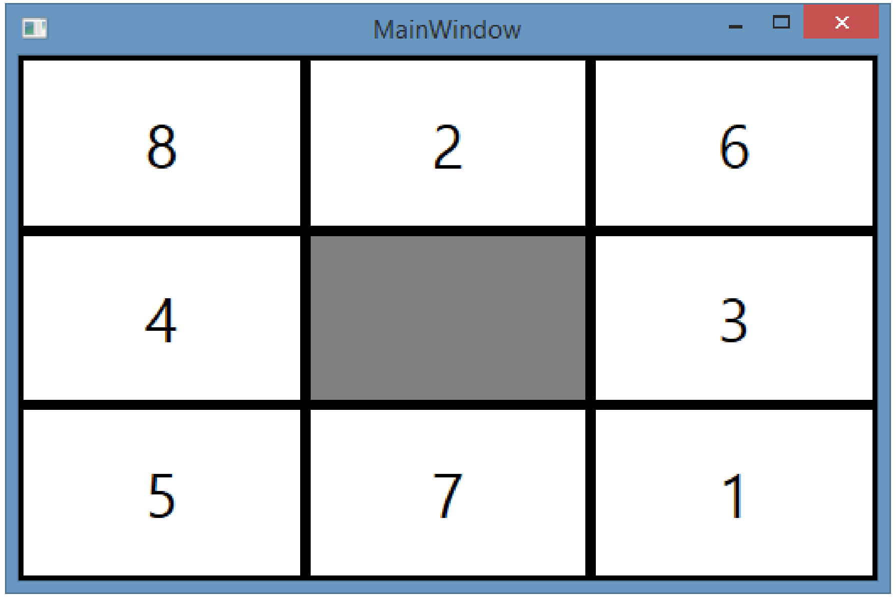
Let’s add a property named Order

```
public string Order
{
    get {
        string s="";
        foreach(Label l in tiles)
            s+=l.Content.ToString();
        return s;
    }
    set {
        string s = value;
        for (int i = 0; i < 9; i++) {
            tiles[i].Content = s[i];
            if (s[i] == ' ')
                tiles[i].Background =Brushes.Gray;
            else
                tiles[i].Background = Brushes.White;
        }
    }
}
```



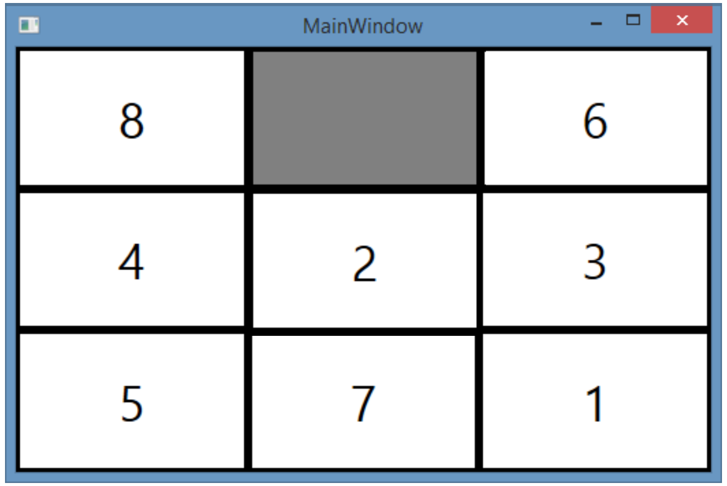


We achieved this!

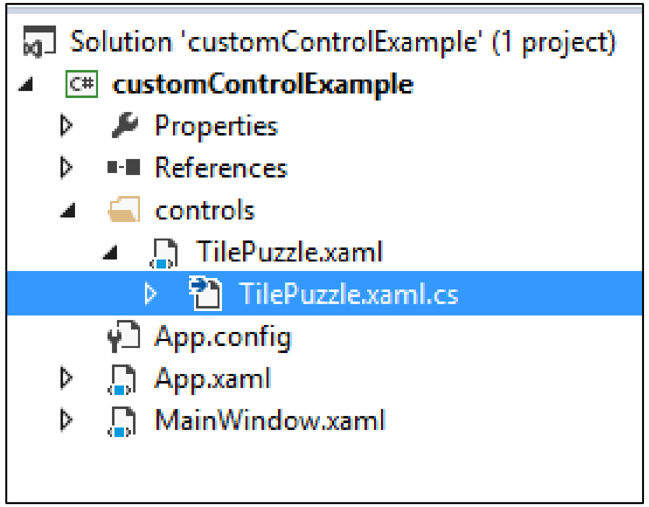


8	2	6
4		3
5	7	1

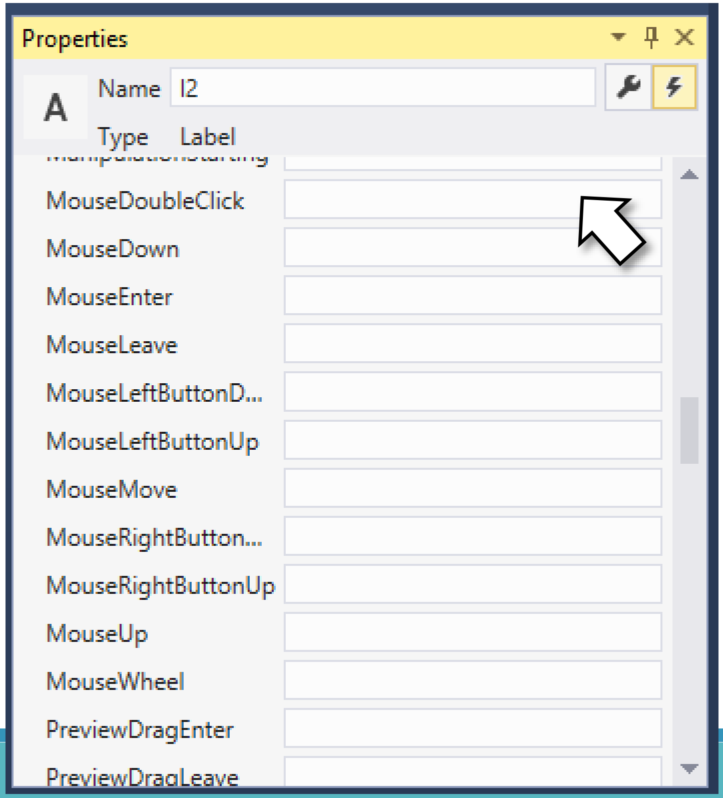
```
<Window x:Class="customControlExample.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:Controls="clr-namespace:customControlExample.controls"
        Title="MainWindow" Height="350" Width="525">
    <Grid>
        <Controls:TilePuzzle Order="8264 3571"/>
    </Grid>
</Window>
```



8		6
4	2	3
5	7	1


Solution 'customControlExample' (1 project)
- customControlExample
  - Properties
  - References
  - controls
    - TilePuzzle.xaml
      - TilePuzzle.xaml.cs
  - App.config
  - App.xaml
  - MainWindow.xaml

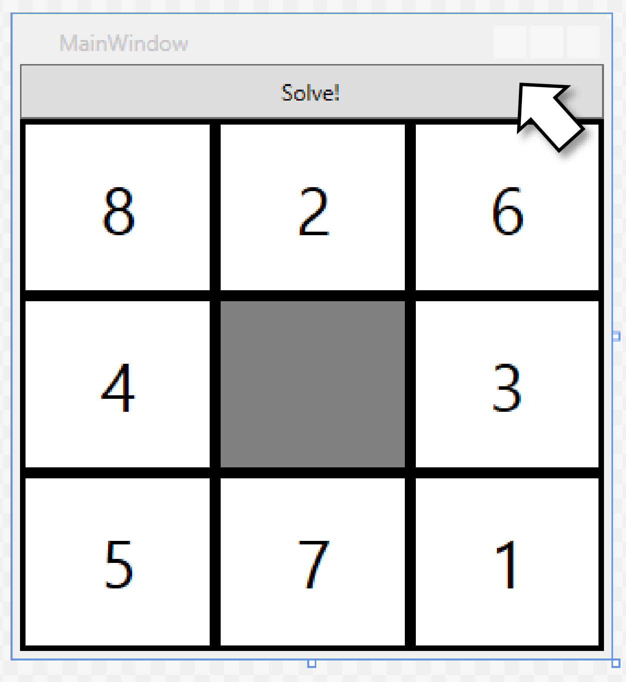
```
private void l2_MouseDoubleClick(object sender, MouseButtonEventArgs e)
{
    char[] s = Order.ToCharArray() ;
    if      (s[0] == ' ') s[0] = s[1];
    else if (s[2] == ' ') s[2] = s[1];
    else if (s[4] == ' ') s[4] = s[1];

    s[1] = ' ';
    Order = new String(s);
}

// Order="8264 3571" → Order="8 6423571"
```

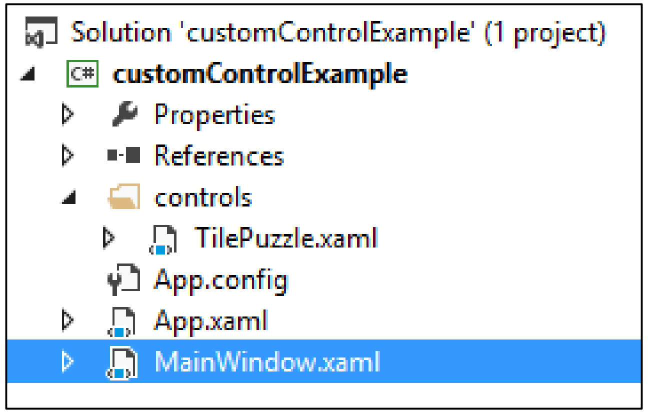

Properties

Name l2

Type Label

ManipulationStarting	
MouseDoubleClick	
MouseDown	
MouseEnter	
MouseLeave	
MouseLeftButtonD...	
MouseLeftButtonUp	
MouseMove	
MouseRightButton...	
MouseRightButtonUp	
MouseUp	
MouseWheel	
PreviewDragEnter	
PreviewDragLeave	



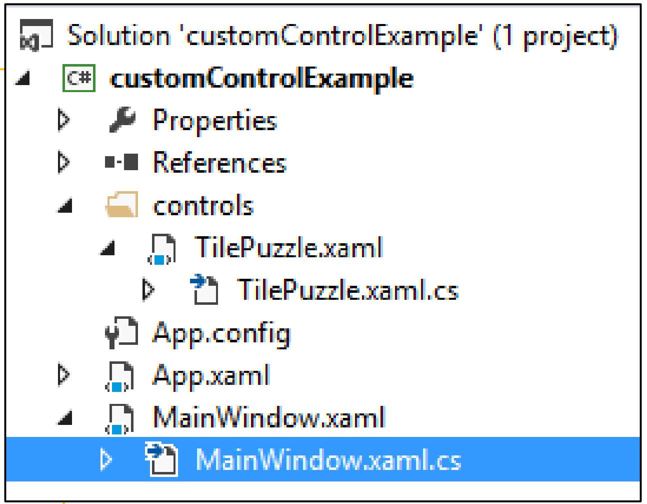


- Solution 'customControlExample' (1 project)
  - customControlExample
    - Properties
    - References
    - controls
      - TilePuzzle.xaml
    - App.config
    - App.xaml
    - MainWindow.xaml

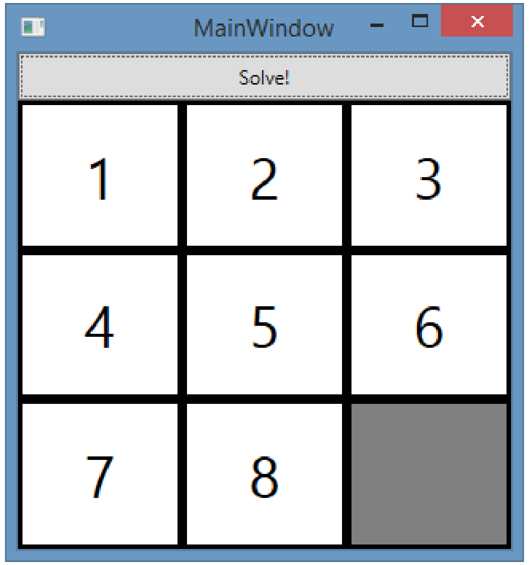
Let’s add a “solve!” Button
in a dock panel

```
<Window x:Class="customControlExample.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:Controls="clr-namespace:customControlExample.controls"
        Title="MainWindow" Height="350" Width="325">
    <DockPanel>
        <Button DockPanel.Dock="Top" Height="30">Solve!</Button>
        <Controls:TilePuzzle x:Name="puzzle" Order="8264 3571"/>
    </DockPanel>
</Window>
```

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

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            // this should lead the Model to solve the puzzle...
            // for now let's just do this:
            puzzle.Order = "12345678 ";
        }
    }
}
```



Solution 'customControlExample' (1 project)
- customControlExample
  - Properties
  - References
  - controls
    - TilePuzzle.xaml
      - TilePuzzle.xaml.cs
  - App.config
  - App.xaml
  - MainWindow.xaml
    - MainWindow.xaml.cs



1	2	3
4	5	6
7	8	