

Properties

abelski

Introduction

- ❖ Properties are values that belong to objects. Properties determine objects' state, behavior and appearance.
- ❖ XAML allows us to set properties values in a relatively simple way.

```
<Label Content="Shalom" Background="Red"/>
```

- ❖ Behind this simple code WPF uses type converters. The type converter is responsible for setting the property value.

Introduction

- ❖ If required, the type converter will create objects and set them as the properties values. In most cases, ready-to-use objects are already available through static fields.

```
<Label Content="Shalom" Background="Red"/>
```



```
ob.Background = Brushes.Red;
```

Property Element Syntax

- ❖ This syntax allows us to write XAML code that represent a property as a separated element that has childs.

```
<Grid>
  <Grid.Background>
    <RadialGradientBrush>
      <GradientStop Color="#FFFFFF" Offset="0.00"/>
      <GradientStop Color="#FFFF0000" Offset="0.20"/>
      <GradientStop Color="#FFFFFF00" Offset="0.40"/>
      <GradientStop Color="#FF00FF00" Offset="0.60"/>
      <GradientStop Color="#FF00FFFF" Offset="0.80"/>
      <GradientStop Color="#FF0000FF" Offset="1.00"/>
    </RadialGradientBrush>
  </Grid.Background>
</Grid>
```

six RadialGradientBrush objects are created in order to describe gradient stops

Property Element Syntax

- ❖ The property element's name includes the control's type (`Grid`), followed by a dot (`.`), followed by the property's name (`Background`).

Property Element Syntax

- ❖ We can alternatively hard code the background property assignment. We will get the same result.

```
RadialGradientBrush bg = new RadialGradientBrush();  
bg.GradientStops.Add(new GradientStop(Colors.White, 0.00));  
bg.GradientStops.Add(new GradientStop(Colors.Red, 0.20));  
.  
.  
.  
.  
ob.Background = bg;
```

Common Properties

- ❖ The WPF control classes inherit from the Control class.
- ❖ The properties inherited from this class are common to all of the Control classes.
- ❖ Most of these common properties control common characteristics such as the size, the color, the position and the font.

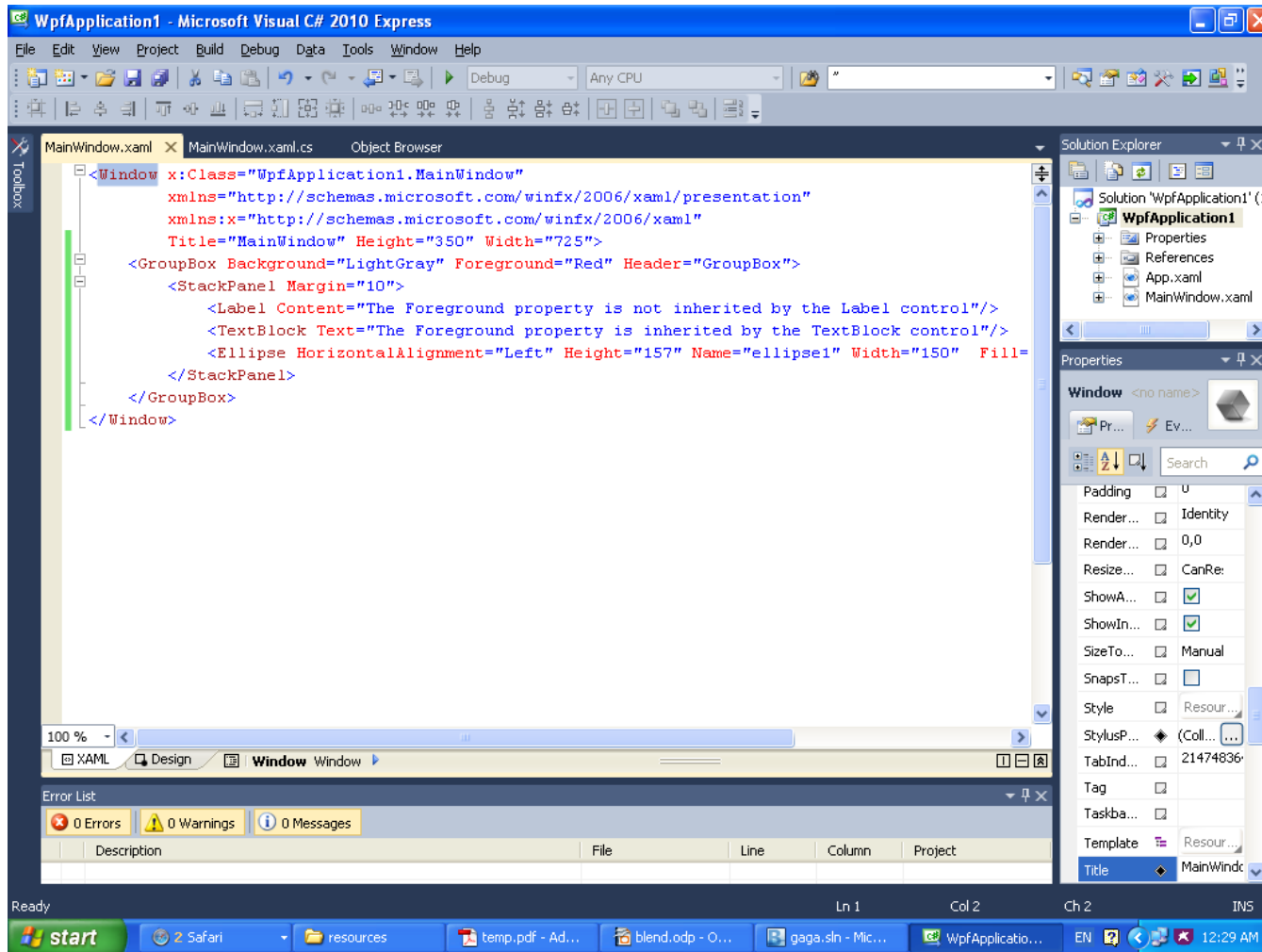
Properties Inheritance

- ❖ Similarly to OOP, WPF supports the concept of properties inheritance from one control to another.
- ❖ In addition, WPF supports the inheritance of properties' values from one control to another when the other is contained within the first. This sort of inheritance doesn't always apply. There are many exceptions.

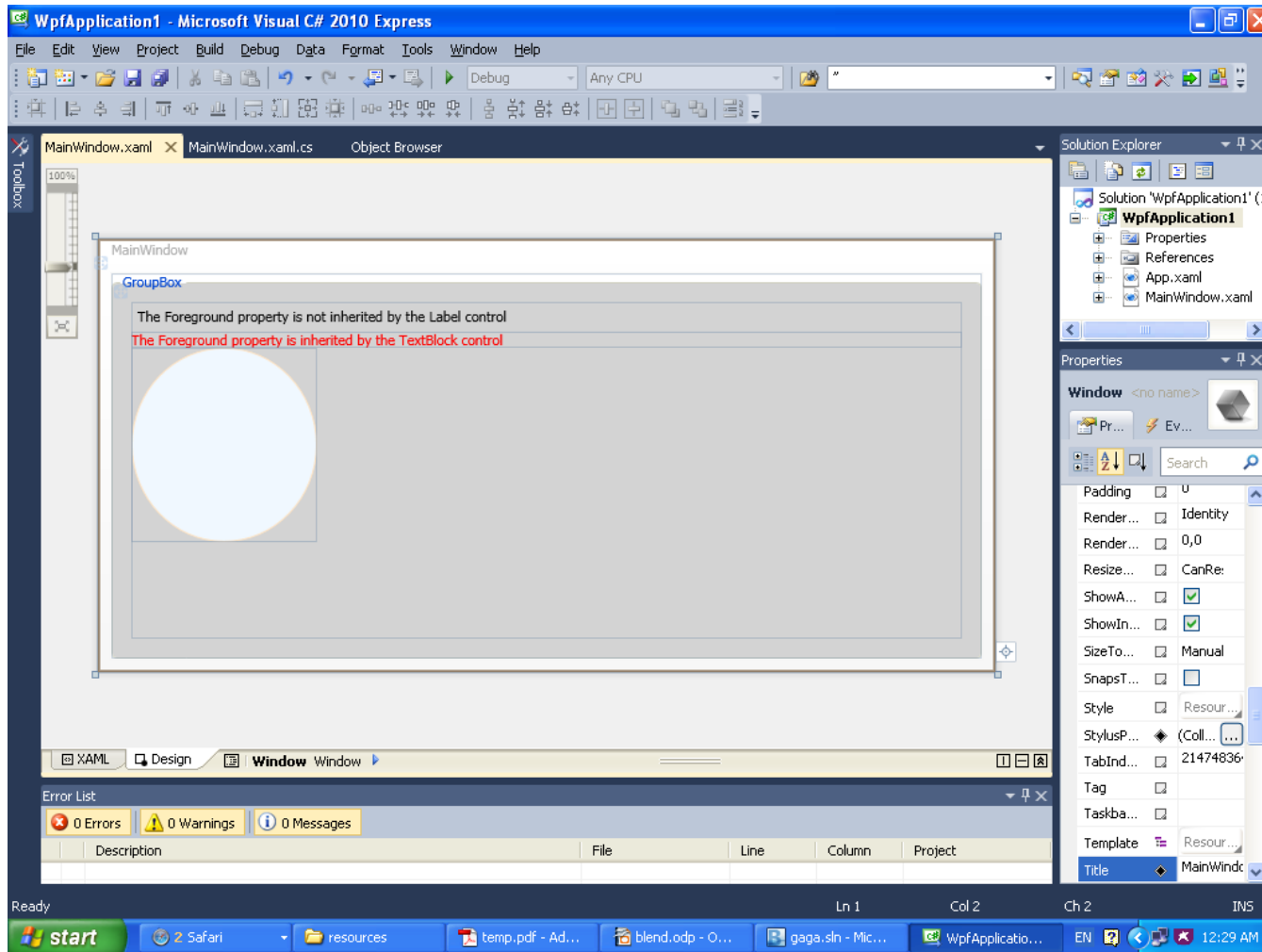
Color

- ❖ When dealing with a drawing control the `Stroke` property determines the brush in use to draw the shape's outline and the `Fill` property determines how the shape will be filled.
- ❖ When dealing with non-drawing controls the `Background` property determines how the control's interior is filled and the `Foreground` property determines the color in use when drawing objects on top of that control.

Color



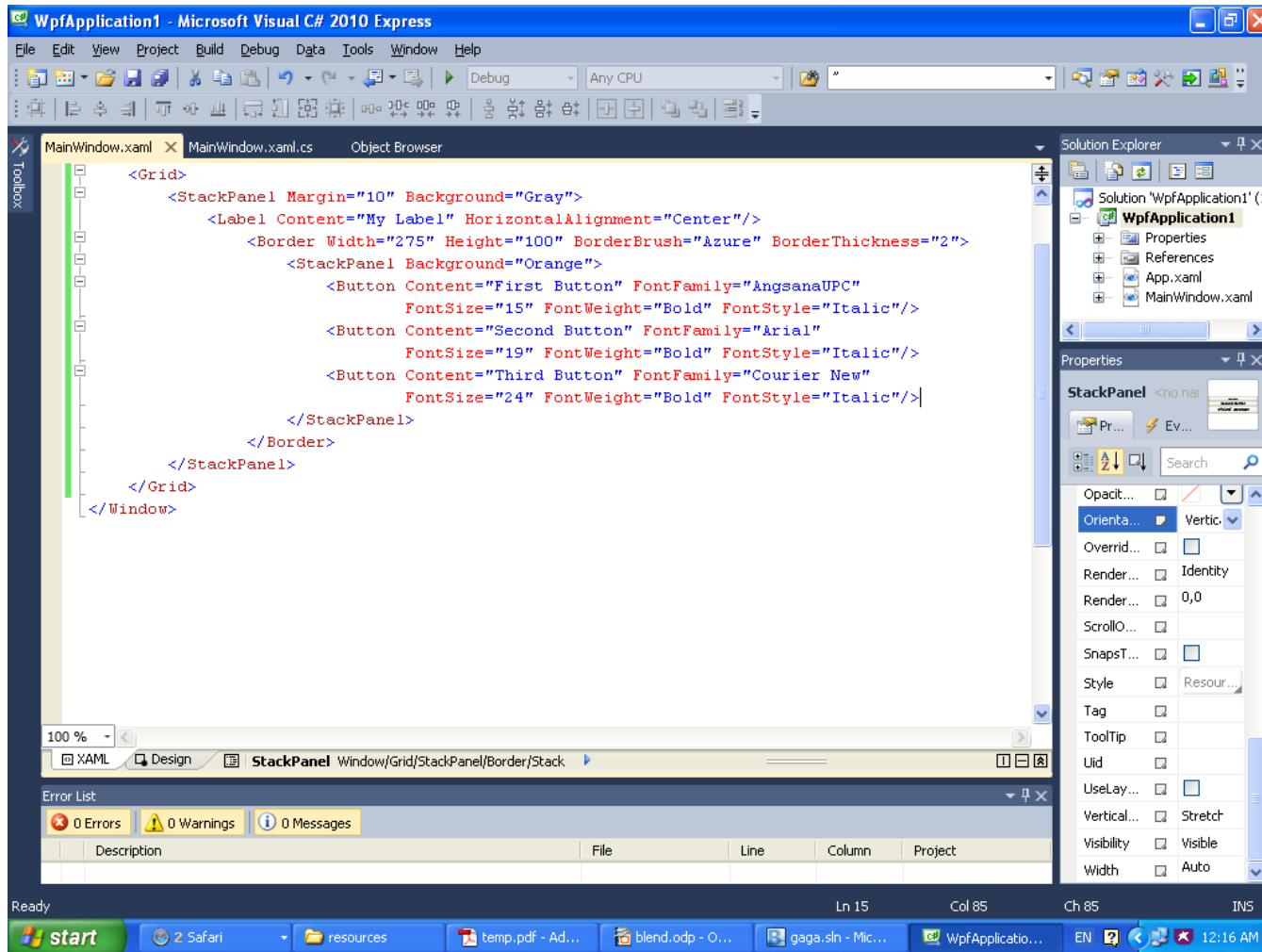
Color



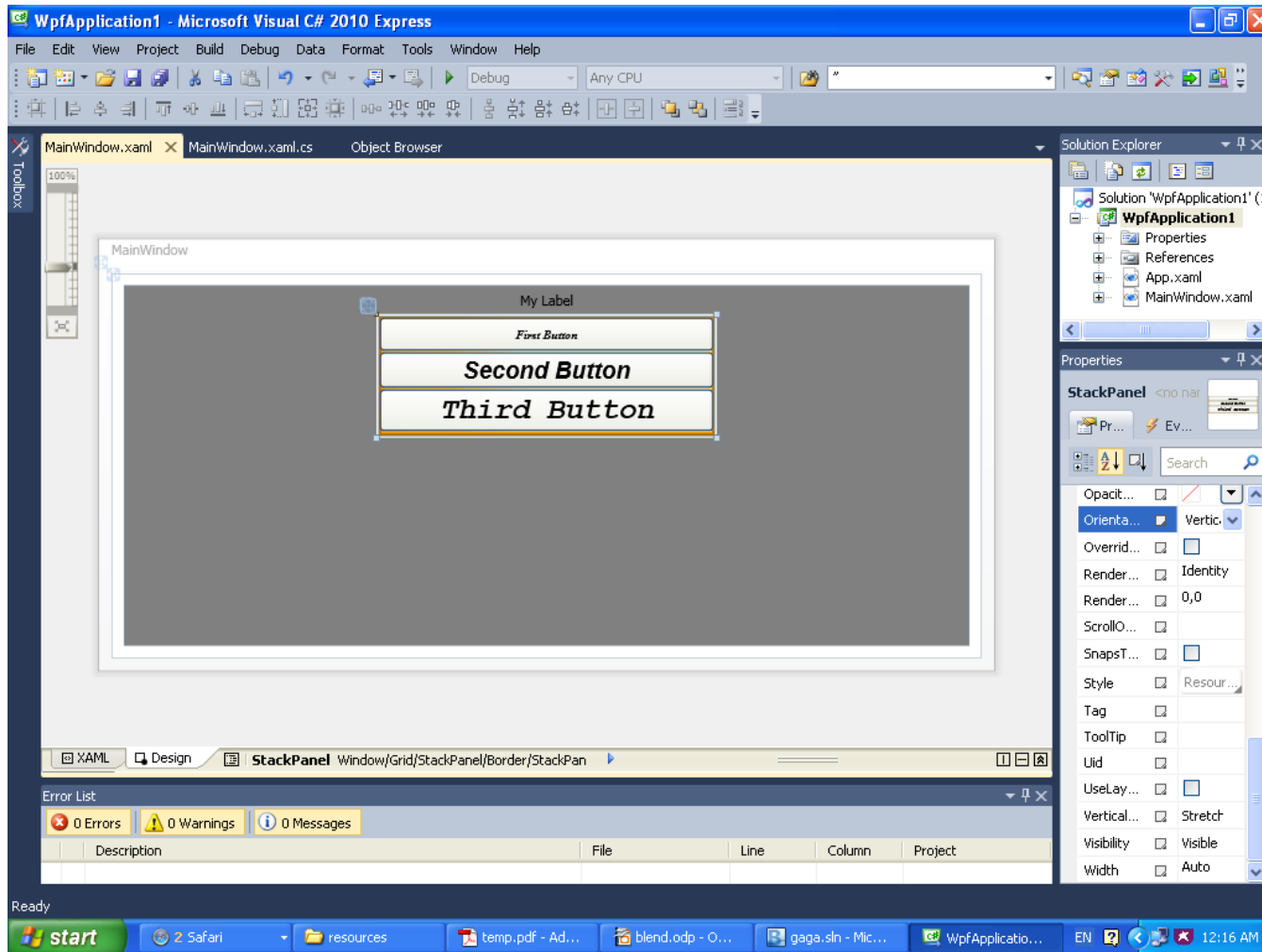
Font

- ❖ The most important font related properties include the following: `FontFamily`, `FontSize`, `FontStyle` and `FontWeight`,

Font



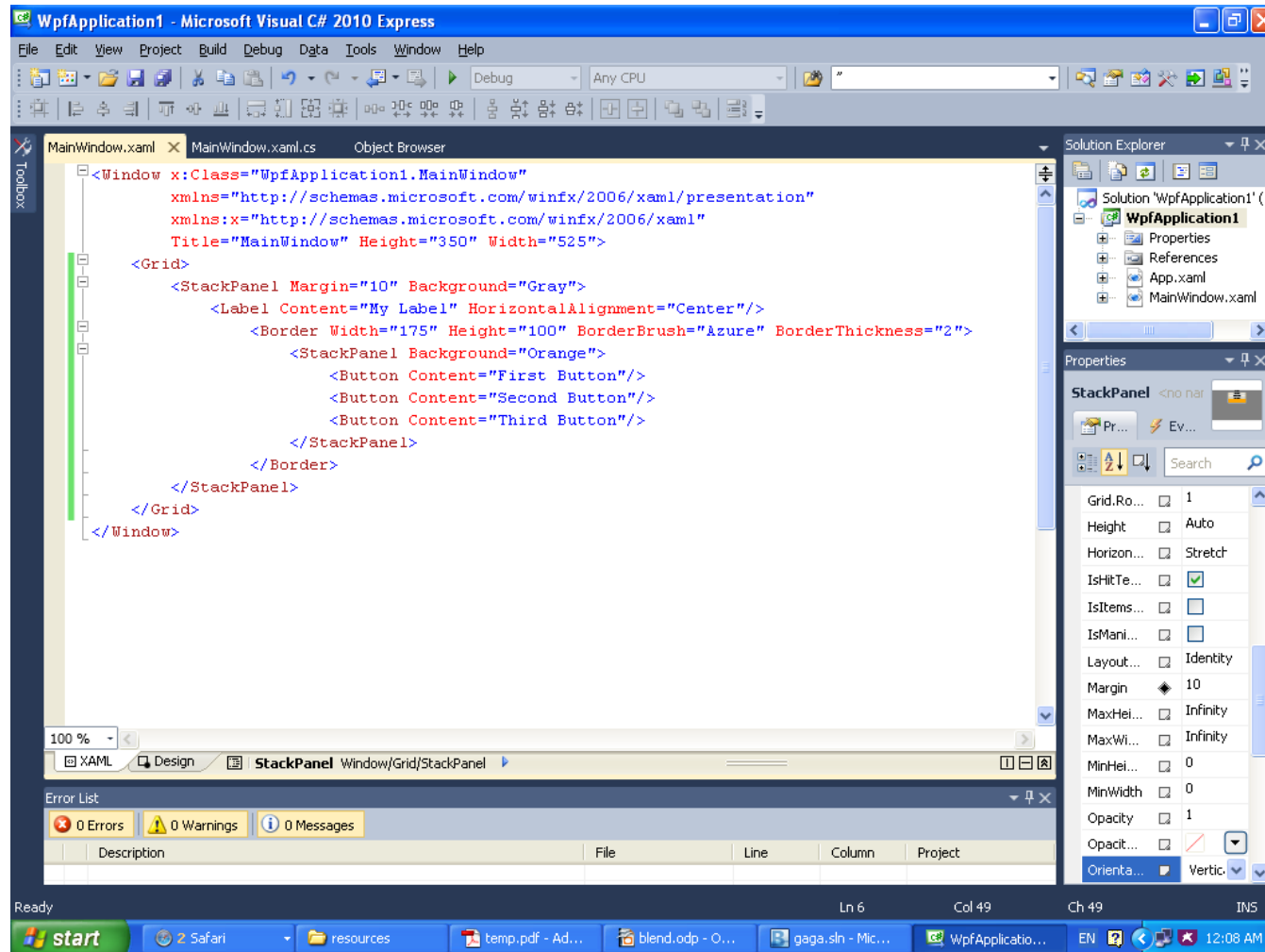
Font



Control Size & Position

- ❖ The most important properties that control the size and the position are `width`, `height`, `HorizontalAlignment`, `VerticalAlignment` and `margin`.
- ❖ The exact meaning of the each value we assign the properties might depend on the specific container we use.

Control Size & Position



Control Size & Position

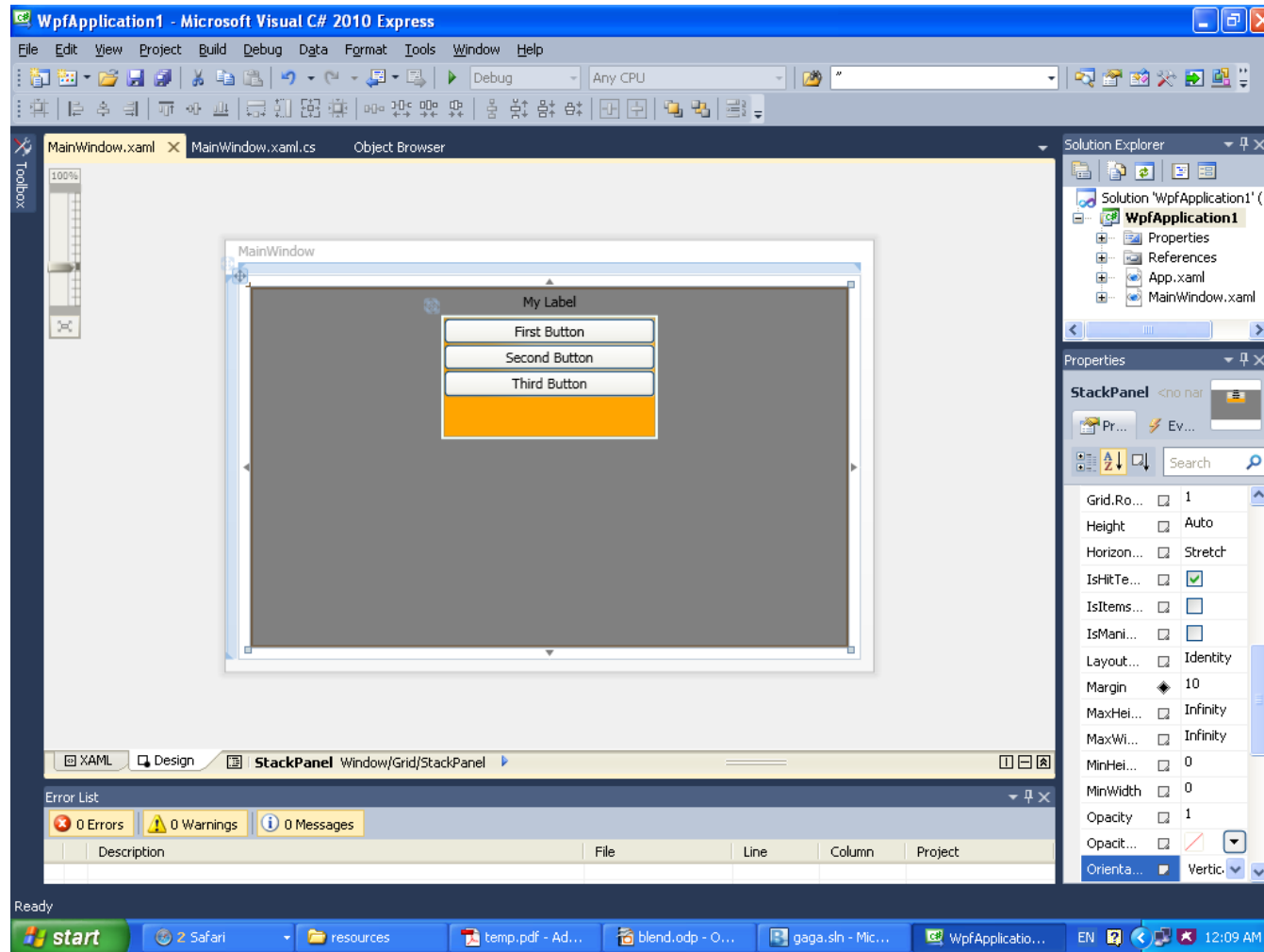


Image Shape

- ❖ In most cases the Shape is displayed in a rectangular area. Changing the value of the `OpacityMask` property we can change that. The pixel's alpha component is the relevant one.

Image Shape

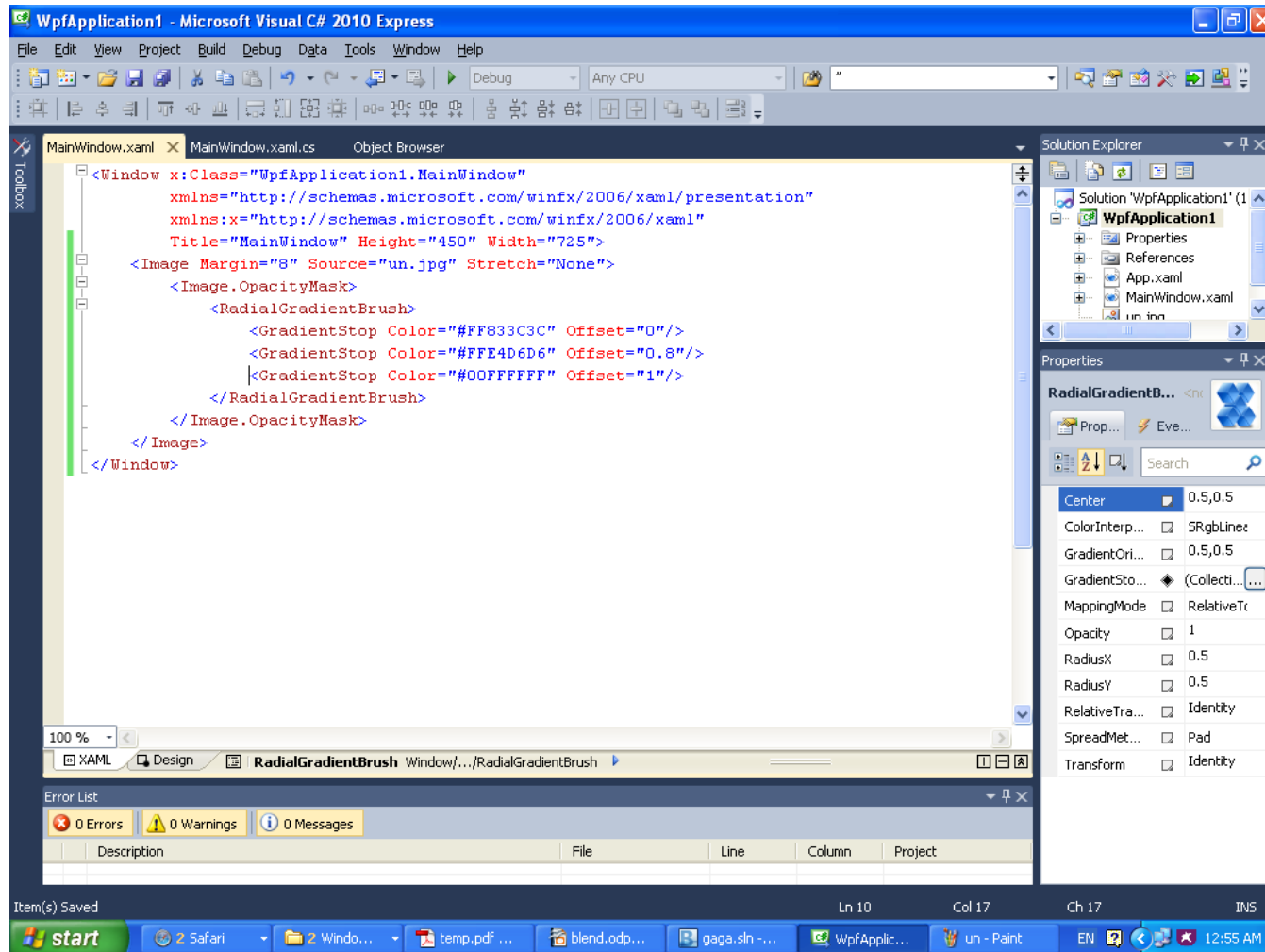


Image Shape

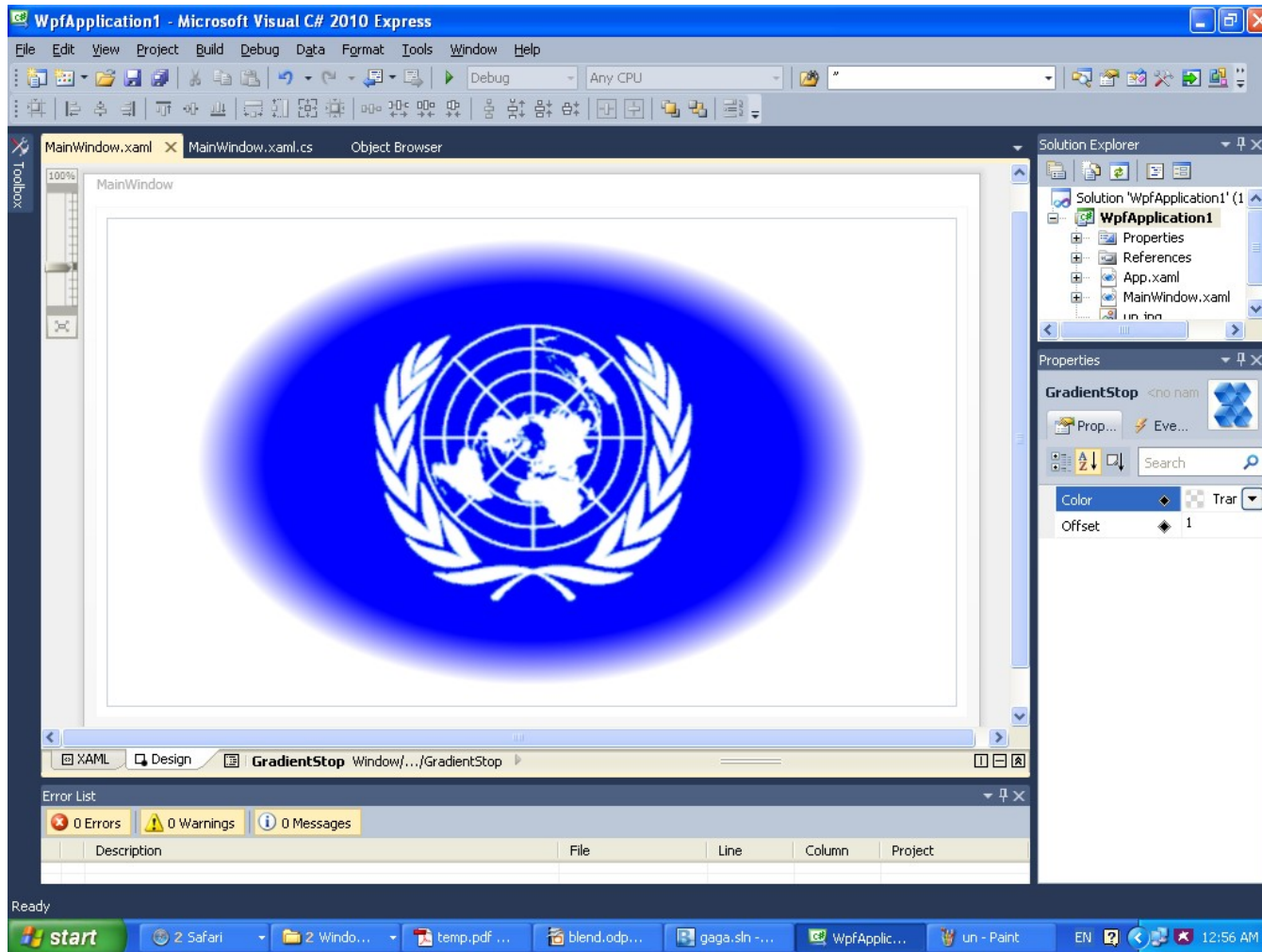


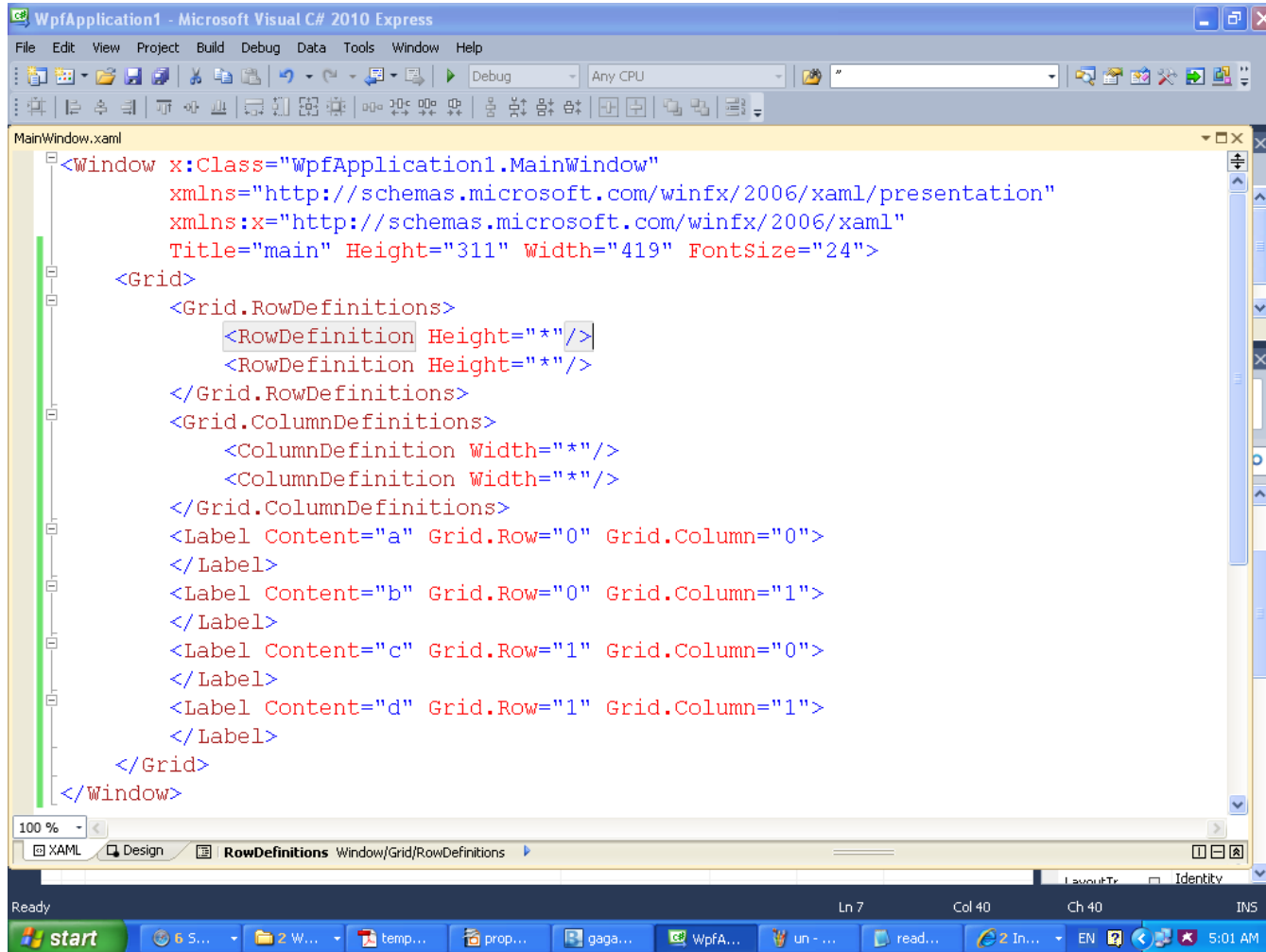
Image Shape

- ❖ The `Image.OpacityMask` element contains the `RadialGradientBrush` element, that contains the `GradientStops` elements.
- ❖ The `GradientStops` elements define the colors the brush uses. The `Offset` attribute defines the ratio of the distance from the center where the color changes.

Attached Properties

- ❖ The attached property is a property available for other controls.
- ❖ The syntax includes the name of the property provider, followed by a dot, followed by the name of the property.

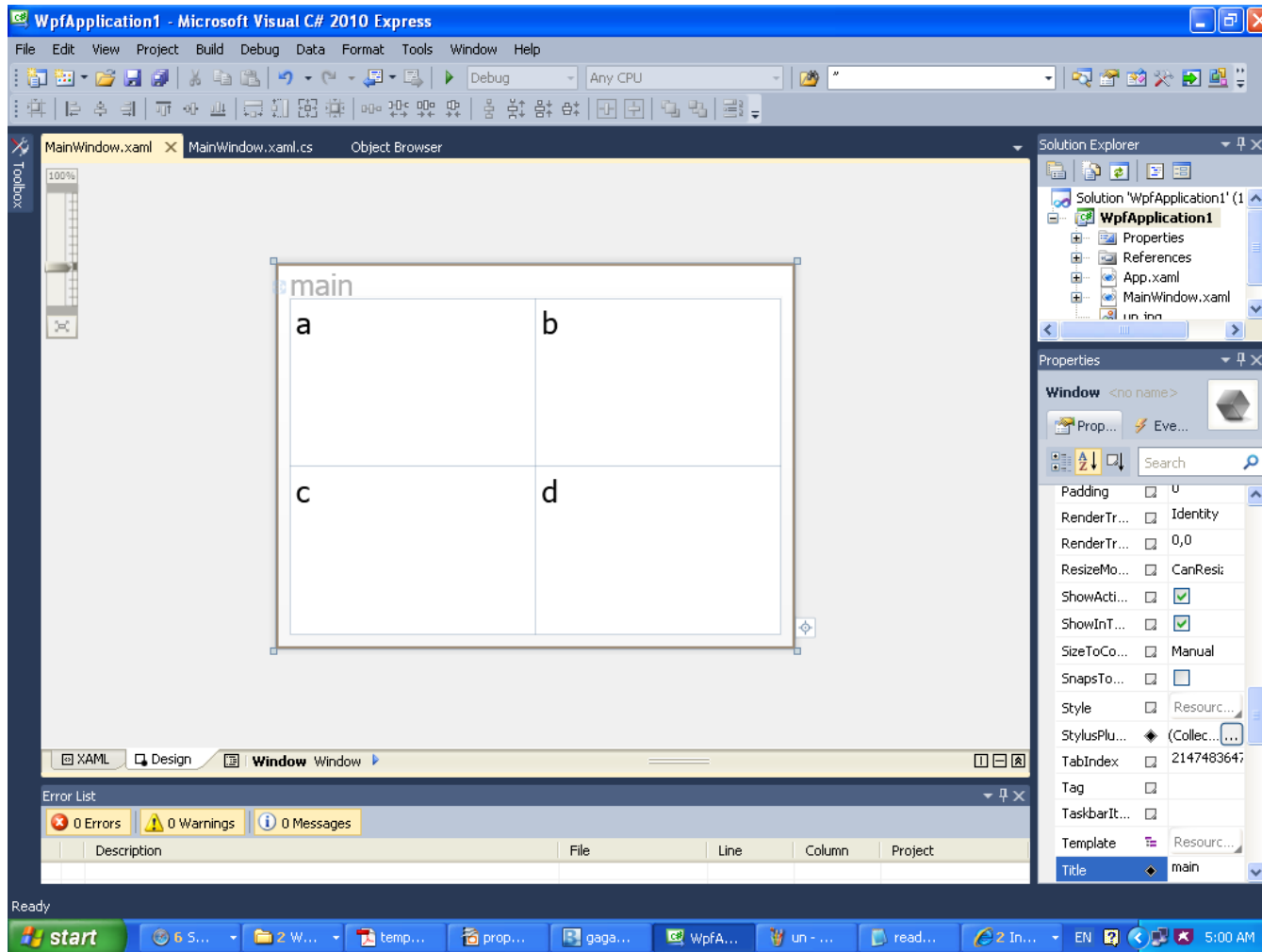
Attached Properties



The screenshot displays the Microsoft Visual C# 2010 Express IDE. The main window shows the XAML code for a WPF application named 'WpfApplication1'. The code defines a 'MainWindow' class and a 'Grid' containing four labels arranged in a 2x2 grid. The labels are labeled 'a', 'b', 'c', and 'd'.

```
<Window x:Class="WpfApplication1.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="main" Height="311" Width="419" FontSize="24">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="*" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>
        <Label Content="a" Grid.Row="0" Grid.Column="0" />
        </Label>
        <Label Content="b" Grid.Row="0" Grid.Column="1" />
        </Label>
        <Label Content="c" Grid.Row="1" Grid.Column="0" />
        </Label>
        <Label Content="d" Grid.Row="1" Grid.Column="1" />
        </Label>
    </Grid>
</Window>
```

Attached Properties



Properties

abelski

Introduction

- ❖ Properties are values that belong to objects. Properties determine objects' state, behavior and appearance.
- ❖ XAML allows us to set properties values in a relatively simple way.

```
<Label Content="Shalom" Background="Red"/>
```
- ❖ Behind this simple code WPF uses type converters. The type converter is responsible for setting the property value.

Introduction

- ❖ If required, the type converter will create objects and set them as the properties values. In most cases, ready-to-use objects are already available through static fields.

```
<Label Content="Shalom" Background="Red"/>
```



```
ob.Background = Brushes.Red;
```

Property Element Syntax

- ❖ This syntax allows us to write XAML code that represent a property as a separated element that has childs.

```
<Grid>
  <Grid.Background>
    <RadialGradientBrush>
      <GradientStop Color="#FFFFFF" Offset="0.00"/>
      <GradientStop Color="#FFF000" Offset="0.20"/>
      <GradientStop Color="#FFF000" Offset="0.40"/>
      <GradientStop Color="#FF00FF" Offset="0.60"/>
      <GradientStop Color="#FF00FF" Offset="0.80"/>
      <GradientStop Color="#FF0000" Offset="1.00"/>
    </RadialGradientBrush>
  </Grid.Background>
</Grid>
```

six RadialGradientBrush objects are created in order to describe gradient stops

Property Element Syntax

- ❖ The property element's name includes the control's type (`Grid`), followed by a dot (`.`), followed by the property's name (`Background`).

Property Element Syntax

- ❖ We can alternatively hard code the background property assignment. We will get the same result.

```
RadialGradientBrush bg = new RadialGradientBrush();  
bg.GradientStops.Add(new GradientStop(Colors.White, 0.00));  
bg.GradientStops.Add(new GradientStop(Colors.Red, 0.20));  
. . .  
ob.Background = bg;
```

Common Properties

- ❖ The WPF control classes inherit from the Control class.
- ❖ The properties inherited from this class are common to all of the Control classes.
- ❖ Most of these common properties control common characteristics such as the size, the color, the position and the font.

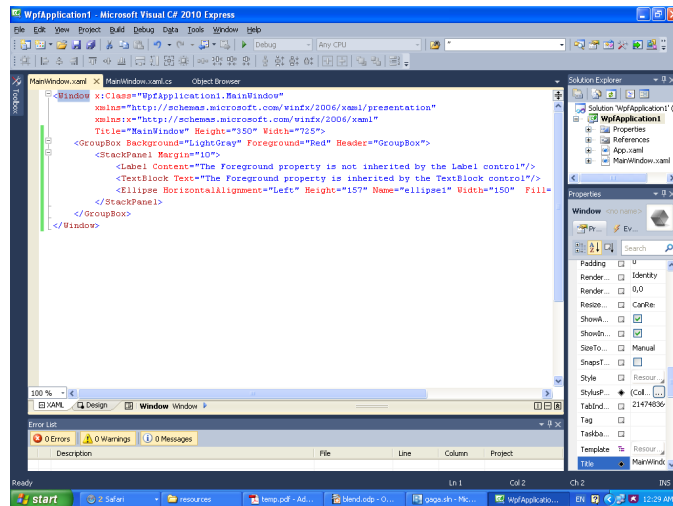
Properties Inheritance

- ❖ Similarly to OOP, WPF supports the concept of properties inheritance from one control to another.
- ❖ In addition, WPF supports the inheritance of properties' values from one control to another when the other is contained within the first. This sort of inheritance doesn't always apply. There are many exceptions.

Color

- ❖ When dealing with a drawing control the `Stroke` property determines the brush in use to draw the shape's outline and the `Fill` property determines how the shape will be filled.
- ❖ When dealing with non-drawing controls the `Background` property determines how the control's interior is filled and the `Foreground` property determines the color in use when drawing objects on top of that control.

Color

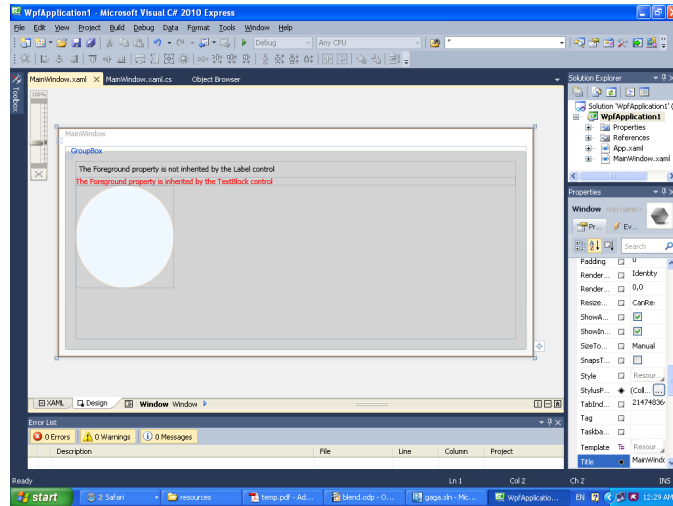


04/25/10

© 2008 Haim Michael

10

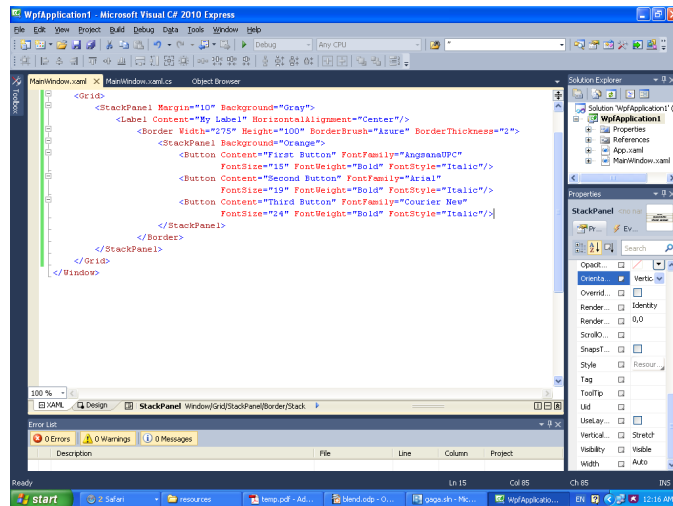
Color



Font

- ❖ The most important font related properties include the following: `FontFamily`, `FontSize`, `FontStyle` and `FontWeight`,

Font

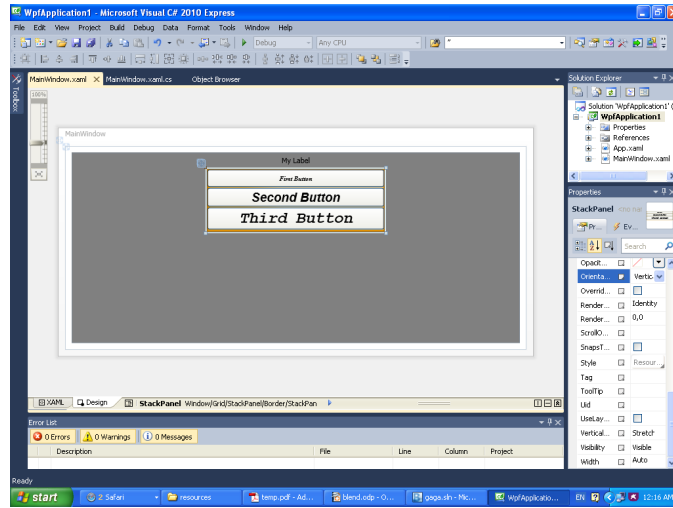


04/25/10

© 2008 Haim Michael

13

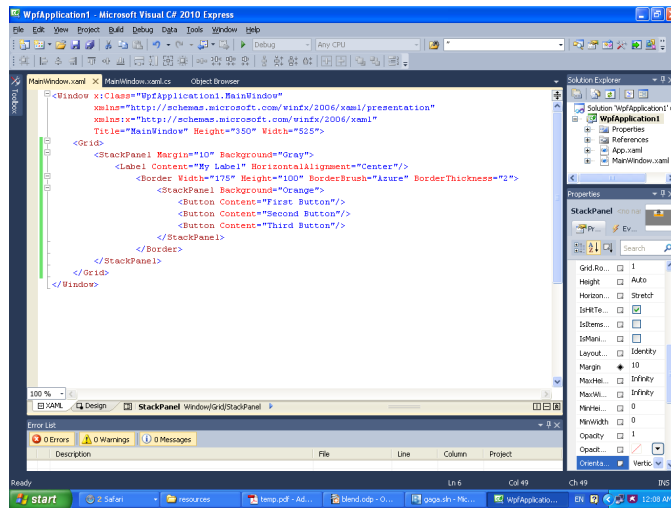
Font



Control Size & Position

- ❖ The most important properties that control the size and the position are `width`, `height`, `HorizontalAlignment`, `VerticalAlignment` and `margin`.
- ❖ The exact meaning of the each value we assign the properties might depend on the specific container we use.

Control Size & Position

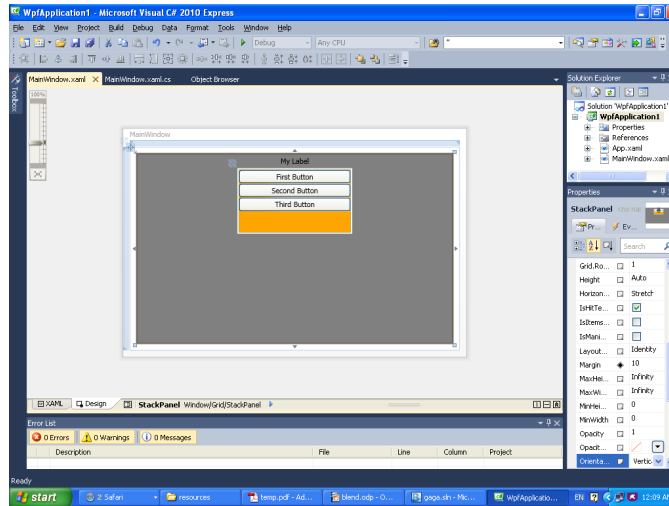


04/25/10

© 2008 Haim Michael

16

Control Size & Position



04/25/10

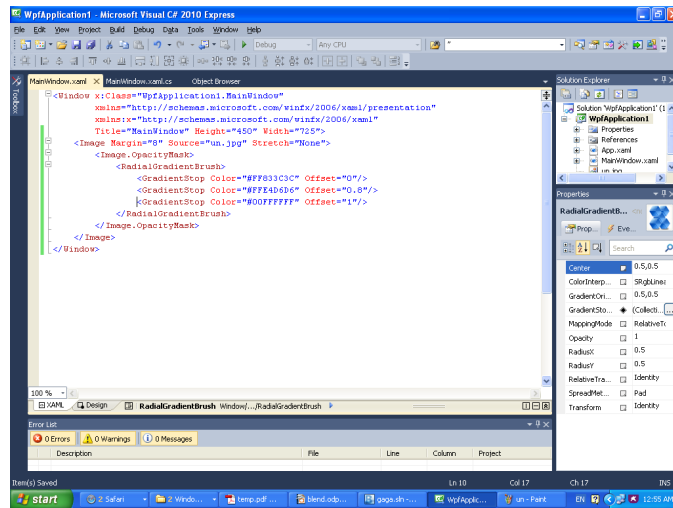
© 2008 Haim Michael

17

Image Shape

- ❖ In most cases the Shape is displayed in a rectangular area. Changing the value of the `OpacityMask` property we can change that. The pixel's alpha component is the relevant one.

Image Shape



04/25/10

© 2008 Haim Michael

19

Image Shape



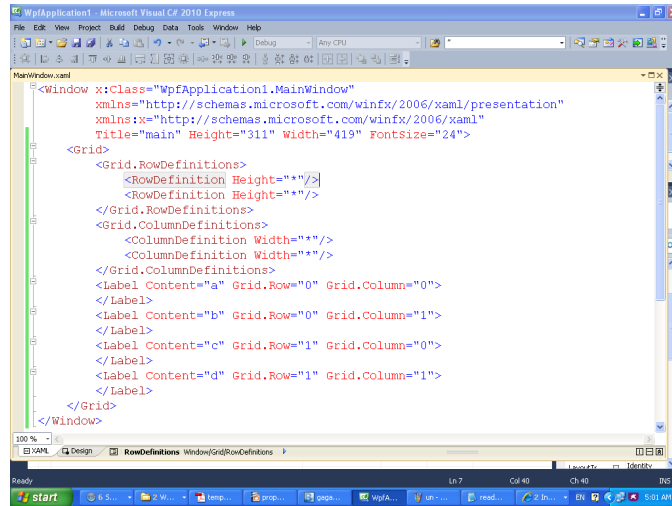
Image Shape

- ❖ The `Image.OpacityMask` element contains the `RadialGradientBrush` element, that contains the `GradientStops` elements.
- ❖ The `GradientStops` elements define the colors the brush uses. The `Offset` attribute defines the ratio of the distance from the center where the color changes.

Attached Properties

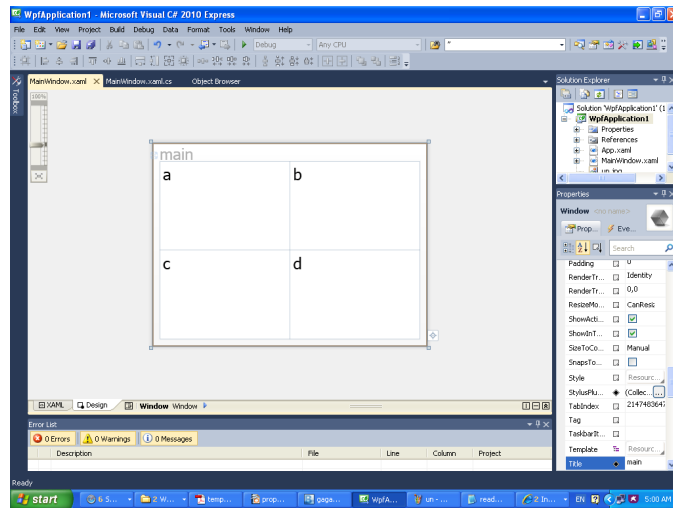
- ❖ The attached property is a property available for other controls.
- ❖ The syntax includes the name of the property provider, followed by a dot, followed by the name of the property.

Attached Properties



```
<?xml version="1.0" encoding="utf-8" ?>
<Window x:Class="WpfApplication1.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="main" Height="311" Width="419" FontSize="24">
  <Grid>
    <Grid.RowDefinitions>
      <RowDefinition Height="*" />
      <RowDefinition Height="*" />
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
      <ColumnDefinition Width="*" />
      <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>
    <Label Content="a" Grid.Row="0" Grid.Column="0">
    </Label>
    <Label Content="b" Grid.Row="0" Grid.Column="1">
    </Label>
    <Label Content="c" Grid.Row="1" Grid.Column="0">
    </Label>
    <Label Content="d" Grid.Row="1" Grid.Column="1">
    </Label>
  </Grid>
</Window>
```

Attached Properties



04/25/10

© 2008 Haim Michael

24