**Events**

You can subscribe to these events, which means that your application will be notified when they occur and you may react to that.

Nos podemos subscriber al evento indicándole el event handler en la vista:



**O incluso, indicándole el event handler en c#: (Code behind)**

using System;

using System.Windows;

using System.Windows.Input;

namespace WpfTutorialSamples.XAML

{

public partial class EventsSample : Window

{

public EventsSample()

{

InitializeComponent();

pnlMainGrid.MouseUp += new MouseButtonEventHandler(pnlMainGrid\_MouseUp);

}

private void pnlMainGrid\_MouseUp(object sender, MouseButtonEventArgs e)

{

MessageBox.Show("You clicked me at " + e.GetPosition(this).ToString());

}

}

App.xaml structure

ne of the most commonly used features of the App.xaml file is to define global resources that may be used and accessed from all over an application, for instance global styles.

The main thing to notice here is the StartupUri property. This is actually the part that instructs which Window or Page to start up when the application is launched

Sustituyendo startup\_uri por evento …Este evento saltará al iniciar la aplicación…y podemos por ejemplo mostrar splash screen, etc…

<Application x:Class="WpfTutorialSamples.App"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Startup="Application\_Startup">

<Application.Resources></Application.Resources>

</Application>

public partial class App : Application

{

private void Application\_Startup(object sender, StartupEventArgs e)

{

// Create the startup window

MainWindow wnd = new MainWindow();

// Do stuff here, e.g. to the window

wnd.Title = "Something else";

// Show the window

wnd.Show();

}

}

Por ejemplo, aquí podemos iniciar la aplicación desde consola, o desde el IDE, CON PARÁMETROS…y dichos parámetros los leeríamos en el evento de start\_up:

namespace WpfTutorialSamples

{

public partial class App : Application

{

private void Application\_Startup(object sender, StartupEventArgs e)

{

MainWindow wnd = new MainWindow();

if(e.Args.Length == 1)

MessageBox.Show("Now opening file: \n\n" + e.Args[0]);

wnd.Show();

}

}

}

Resources—

Ya hemos visto como podemos declarar en XAML static y dinamyc resources a nivel de elemento, window y app. Veamos cómo, una vez declarados estos resources, podríamos acceder a ellos desde c#

public partial class ResourcesFromCodeBehindSample : Window

{

public ResourcesFromCodeBehindSample()

{

InitializeComponent();

}

private void btnClickMe\_Click(object sender, RoutedEventArgs e)

{

lbResult.Items.Add(pnlMain.FindResource("strPanel").ToString());

lbResult.Items.Add(this.FindResource("strWindow").ToString());

lbResult.Items.Add(Application.Current.FindResource("strApp").ToString());

}

}

Manejando unhandled exceptions:

Obviamente lo mejor es manejarlas de manera local con try, catch…pero, siempre se nos puede escapar algo…Esto lo podemos manejar a nivel de aplicación, en el app.css

using System;

using System.Windows;

namespace WpfTutorialSamples

{

public partial class App : Application

{

private void Application\_DispatcherUnhandledException(object sender, System.Windows.Threading.DispatcherUnhandledExceptionEventArgs e)

{

MessageBox.Show("An unhandled exception just occurred: " + e.Exception.Message, "Exception Sample", MessageBoxButton.OK, MessageBoxImage.Warning);

e.Handled = true;

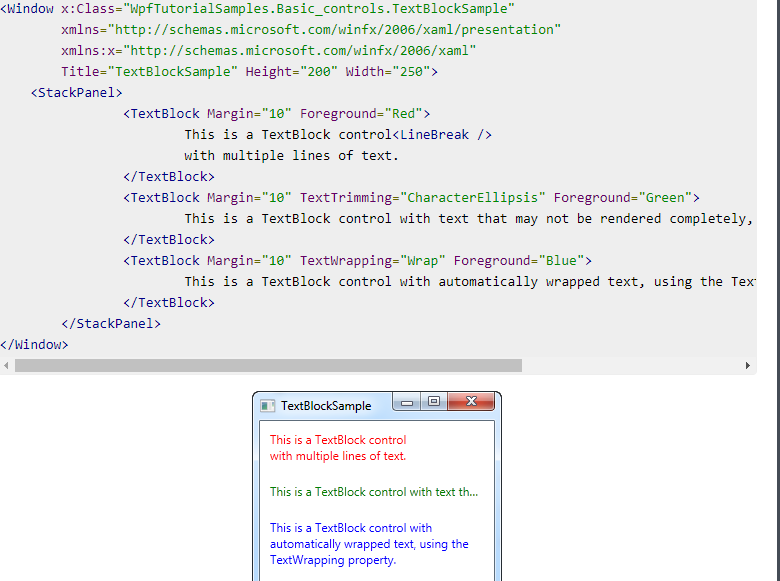
}

}

}

TEXTBOX:

Managing long texts:



Labels and Access Keys

