[C# WPF Tutorial #10 - OpenFileDialog (File Picker)](https://www.youtube.com/watch?v=Ks9bzPSx7Vs&list=PLih2KERbY1HHOOJ2C6FOrVXIwg4AZ-hk1&index=10)

Window will look like this.

A screenshot of a computer

Description automatically generated

Completed code will look like this. Please note that this is the Multiselect version.

MainWindow.xaml

<Window x:Class="\_10\_WPF\_Tutorial\_v2.MainWindow"

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

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

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:\_10\_WPF\_Tutorial\_v2"

mc:Ignorable="d"

Title="MainWindow" Height="250" Width="800">

<Grid>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Button Name="btnFire"

Content="Fire"

Width="100"

Height="40"

Click="btnFire\_Click"

/>

<TextBlock Grid.Row="1"

Name="tbInfo"

FontSize="16"

VerticalAlignment="Center"

HorizontalAlignment="Center"

/>

</Grid>

</Window>

MainWindow.xaml.cs

using Microsoft.Win32;

using System.Windows;

namespace \_10\_WPF\_Tutorial\_v2

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

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

{

OpenFileDialog fileDialog = new OpenFileDialog();

fileDialog.Filter = "C# Source Files | \*.cs";

fileDialog.InitialDirectory = "C:\\temp";

fileDialog.Title = "Please select C# Source File(s)...";

fileDialog.Multiselect = true;

bool? success = fileDialog.ShowDialog();

if (success == true)

{

// Do something

string[] paths = fileDialog.FileNames;

string[] fileNames = fileDialog.SafeFileNames;

//tbInfo.Text = fileName;

}

else

{

// They didn't pick anything

}

}

}

}

# Start new WPF solution in Visual Studio.

A black screen with white text

Description automatically generated

A black rectangular object with a black background

Description automatically generated

# Name it:

A screenshot of a computer

Description automatically generated

# Starting point. Code looks like this initially.

MainWindow.xaml

<Window x:Class="\_10\_WPF\_Tutorial\_v1.MainWindow"

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

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

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:\_10\_WPF\_Tutorial\_v1"

mc:Ignorable="d"

Title="MainWindow" Height="450" Width="800">

<Grid>

</Grid>

</Window>

# Add the following code to the .xaml file to create the GUI.

A screenshot of a computer

Description automatically generated

MainWindow.xaml

<Window x:Class="\_10\_WPF\_Tutorial\_v1.MainWindow"

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

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

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:\_10\_WPF\_Tutorial\_v1"

mc:Ignorable="d"

Title="MainWindow" Height="250" Width="800">

<Grid>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Button Name="btnFire"

Content="Fire"

Width="100"

Height="40"

/>

<TextBlock Grid.Row="1"

Name="tbInfo"

FontSize="16"

VerticalAlignment="Center"

HorizontalAlignment="Center"

/>

</Grid>

</Window>

# The MainWindow.xaml.cs code behind page will initially look like this:

MainWindow.xaml.cs

using System.Windows;

namespace \_10\_WPF\_Tutorial\_v1

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

}

}

# Add a Click event handler to the button

MainWindow.xaml

<Window x:Class="\_10\_WPF\_Tutorial\_v1.MainWindow"

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

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

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:\_10\_WPF\_Tutorial\_v1"

mc:Ignorable="d"

Title="MainWindow" Height="250" Width="800">

<Grid>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Button Name="btnFire"

Content="Fire"

Width="100"

Height="40"

Click="btnFire\_Click"

/>

<TextBlock Grid.Row="1"

Name="tbInfo"

FontSize="16"

VerticalAlignment="Center"

HorizontalAlignment="Center"

/>

</Grid>

</Window>

# Add non-static object OpenFileDialog to click event handler

Using Microsoft.Win32 will be automatically added by Visual Studio.

At this point the Open Dialog window opens but does nothing else.

Note it is Modal.

Note the return type is a nullable Boolean.

MainWindow.xaml.cs

using Microsoft.Win32;

using System.Windows;

namespace \_10\_WPF\_Tutorial\_v2

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

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

{

OpenFileDialog fileDialog = new OpenFileDialog();

fileDialog.ShowDialog();

}

}

}

# Working with the Nullable Boolean

Assign the Boolean return value to a variable ‘success’ and test if it was True or False.

Note: Because Boolean is nullable it can be three states, True, False, NULL therefore the if statement must have double equals sign (==) and not just the ‘if(success) part.

MainWindow.xaml.cs

using Microsoft.Win32;

using System.Windows;

namespace \_10\_WPF\_Tutorial\_v2

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

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

{

OpenFileDialog fileDialog = new OpenFileDialog();

bool? success = fileDialog.ShowDialog();

if(success == true)

{

// Do something

}

else

{

// They didn't pick anything

}

}

}

}

# Assign the path and file name to a variable ‘path’

Put a breakpoint on the new line to watch the variable ‘path’ have the full address assigned.

MainWindow.xaml.cs

using Microsoft.Win32;

using System.Windows;

namespace \_10\_WPF\_Tutorial\_v2

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

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

{

OpenFileDialog fileDialog = new OpenFileDialog();

bool? success = fileDialog.ShowDialog();

if(success == true)

{

// Do something

string path = fileDialog.FileName;

tbInfo.Text = path;

}

else

{

// They didn't pick anything

}

}

}

}

# Return File Name only

If you only want the file name use this line instead.

Put a breakpoint on the new line to watch the variable ‘fileName’ have the File Name assigned.

MainWindow.xaml.cs

using Microsoft.Win32;

using System.Windows;

namespace \_10\_WPF\_Tutorial\_v2

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

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

{

OpenFileDialog fileDialog = new OpenFileDialog();

bool? success = fileDialog.ShowDialog();

if(success == true)

{

// Do something

string path = fileDialog.FileName;

string fileName = fileDialog.SafeFileName;

tbInfo.Text = fileName;

}

else

{

// They didn't pick anything

}

}

}

}

# Adding Filters for what can and cannot be accessed

Add Description | file suffix e.g. \*.cs

MainWindow.xaml.cs

using Microsoft.Win32;

using System.Windows;

namespace \_10\_WPF\_Tutorial\_v2

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

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

{

OpenFileDialog fileDialog = new OpenFileDialog();

fileDialog.Filter = "C# Source Files | \*.cs";

bool? success = fileDialog.ShowDialog();

if (success == true)

{

// Do something

string path = fileDialog.FileName;

string fileName = fileDialog.SafeFileName;

tbInfo.Text = fileName;

}

else

{

// They didn't pick anything

}

}

}

}

# Setting initial directory

OpenDialog normally opens the last directory that you opened but it you want to set the initial open location add the following code.

Note: If the location you coded doesn’t exist it will default to last opened directory.

MainWindow.xaml.cs

using Microsoft.Win32;

using System.Windows;

namespace \_10\_WPF\_Tutorial\_v2

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

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

{

OpenFileDialog fileDialog = new OpenFileDialog();

fileDialog.Filter = "C# Source Files | \*.cs";

fileDialog.InitialDirectory = "C:\\temp";

bool? success = fileDialog.ShowDialog();

if (success == true)

{

// Do something

string path = fileDialog.FileName;

string fileName = fileDialog.SafeFileName;

tbInfo.Text = fileName;

}

else

{

// They didn't pick anything

}

}

}

}

# Add a title to the OpenDialog window

MainWindow.xaml.cs

using Microsoft.Win32;

using System.Windows;

namespace \_10\_WPF\_Tutorial\_v2

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

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

{

OpenFileDialog fileDialog = new OpenFileDialog();

fileDialog.Filter = "C# Source Files | \*.cs";

fileDialog.InitialDirectory = "C:\\temp";

fileDialog.Title = "Please choose a C# Source File...";

bool? success = fileDialog.ShowDialog();

if (success == true)

{

// Do something

string path = fileDialog.FileName;

string fileName = fileDialog.SafeFileName;

tbInfo.Text = fileName;

}

else

{

// They didn't pick anything

}

}

}

}

# Enabling MultiSelect

To see this in action put a breakpoint at about line 28 and watch the variables.

MainWindow.xaml.cs

using Microsoft.Win32;

using System.Windows;

namespace \_10\_WPF\_Tutorial\_v2

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

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

{

OpenFileDialog fileDialog = new OpenFileDialog();

fileDialog.Filter = "C# Source Files | \*.cs";

fileDialog.InitialDirectory = "C:\\temp";

fileDialog.Title = "Please select C# Source File(s)...";

fileDialog.Multiselect = true;

bool? success = fileDialog.ShowDialog();

if (success == true)

{

// Do something

string[] paths = fileDialog.FileNames;

string[] fileNames = fileDialog.SafeFileNames;

//tbInfo.Text = fileName;

}

else

{

// They didn't pick anything

}

}

}

}