**Data Binding**

There are two types of data bindings.

1. One Way Data Binding
2. Two Way Data Binding

Date : 9/11/2022

**Input in WPF:**

1. **WPFMouseInput**.
   * MouseEnter
   * MouseLeave
   * MouseMove
   * MouseDown
2. **WPFKeyboardInput**
   * KeyDown
   * KeyUp
   * TextInput etc.
3. **WPFCommandsInput**.
   * Open
   * Save
   * New

**Command Line Argument:**

Steps:

1. Create MainWindow.xaml window
2. Now subscribe the Startup event in App.xaml file as shown below.
3. Given below is the implementation of the app\_Startup event in App.xaml.cs which will get the command line arguments.
4. Now, in the MainWindow class, the program will open the txt file and write all the text on textbox.
5. If there is some error found, then the program will display an error message on textbox.
6. When the above code is compiled and executed, it will produce a blank window with a textbox because this program needs a command line argument. So Visual Studio provides an easy way to execute your application with command line parameters.
7. Right click on your WPF project in the solution explorer and select properties, it will display the following window.
8. Select Debug option and write the file path in the Command line argument.
9. Create a txt file with Test.txt and write some text in that file and save it on any location. In this case, the txt file is saved on “**D:\**” hard drive.
10. Save the changes in your project and compile and execute your application now. You will see the text in TextBox which the program reads from the Text.txt file.
11. Now let’s try and change the file name on your machine from **Test.txt** to **Test1.txt** and execute your program again, then you will see that error message in the text box.

**Resources**

Resources can be of two types −

* StaticResource
* DynamicResource

**Template**

There are two types of templates which are most commonly used −

* Control Template
* Data Template

**Style**

Styles can be defined on the following levels –

* control level
* Layout level
* Windows level
* Application level

**Triggers**

There are three types of triggers –

* Property Triggers
* Data Triggers
* Event Triggers (used for animation)

**Debugging in C#**

**10/11/2022**

**Custom Controls**

**Steps**

1. Let’s take an example to understand how custom controls work. Create a new WPF project and then right-click on your solution and select Add > New Item...
2. It will open the following window. Now select **Custom Control (WPF)** and name it **MyCustomControl**.
3. Click the Add button and you will see that two new files (Themes/Generic.xaml and MyCustomControl.cs) will be added in your solution.
4. Here is the XAML code in which style is set for the custom control in Generic.xaml file
5. Here is the C# code for MyCustomControl class which is inherited from the button class and in constructor it overrides the metadata.
6. Here is the custom control click event implementation in C# which updates the text of the text block.
7. Here is implementation in MainWindow.xaml to add the custom control and a TextBlock.
8. When you compile and execute the above code, it will produce the following window with a custom control which is a customized button.
9. Upon clicking the customized button, you will see that the text inside text block is updated.

**Exceptions:**

The following table lists the standard exceptions provided by the runtime and the conditions under which you should create a derived class.

|  |  |  |
| --- | --- | --- |
| **Exception type** | **Base type** | **Description** |
| **Exception** | Object | Base class for all exceptions. |
| **SystemException** | Exception | Base class for all runtime-generated  errors. |
| **IndexOutOfRangeException** | SystemException | Thrown by the runtime only when an  array is indexed improperly. |
| **NullReferenceException** | SystemException | Thrown by the runtime only when a null  object is referenced. |
| **AccessViolationException** | SystemException | Thrown by the runtime only when invalid  memory is accessed. |
| **InvalidOperationException** | SystemException | Thrown by methods when in an invalid  state. |
| **ArgumentException** | SystemException | Base class for all argument exceptions. |
| **ArgumentNullException** | ArgumentException | Thrown by methods that do not allow an  argument to be null. |
| **ArgumentOutOfRangeException** | ArgumentException | Thrown by methods that verify that  arguments are in a given range. |
| **ExternalException** | SystemException | Base class for exceptions that occur or  are targeted at environments outside  the runtime. |
| **SEHException** | ExternalException | Exception encapsulating Win32  structured exception handling information. |

**Localization - Not Understand**

It is used in convert same application project in different language to expand business.

# **WPF - Interaction**

In WPF, an interaction shows how a view interacts with controls located in that view. The most commonly known interactions are of two types −

* Behaviors //not understand
* Drag and Drop