Chapter 18

Errors Handling

- Learning how to handle errors

18.1 Introduction

Error handling is an essential procedure in Visual Basic 2010 programming because it can help make the program error-free. An error-free program can run smoothly and efficiently, and the user does not have to face all sorts of problems such as program crash or system hang.

Errors often occur due to incorrect input from the user. For example, the user might make the mistake of attempting to enter a text (string) to a box that is designed to handle only numeric values such as the weight of a person, the computer will not be able to perform arithmetic calculation for text therefore will create an error. We call these errors synchronous errors.

Therefore, a good programmer should be more alert to the parts of program that could trigger errors and should write errors handling code to help the user in managing the errors. Writing errors handling code is a good practice for Visual Basic programmers, so do not try to finish a program fast by omitting the errors handling code. However, there should not be too many errors handling code in the program as it create problems for the programmer to maintain and troubleshoot the program later.

VB2010 has improved a lot in built-in errors handling compared to Visual Basic 6. For example, when the user attempts to divide a number by zero, Vb2010 will not return an error message but gives the 'infinity' as the answer (although this is mathematically incorrect, because it should be undefined)

18.2 Using On Error GoTo Syntax

Visual Basic 2010 still supports the VB6 errors handling syntax that is the *On Error GoTo* program_label structure. Although it has a more advanced error handling method,

we shall deal with that later. We shall now learn how to write errors handling code in VB2010. The syntax for errors handling is

```
On Error GoTo  program_label
```

Where **program_label** is the section of code that is designed by the programmer to handle the error committed by the user. Once the program detects an error, the program will jump to the program_label section for error handling.

Example 18.1: Division by Zero

In this example, we will deal with the error of entering non-numeric data into the textboxes that suppose to hold numeric values. The program_label here is error_hanldler. When the user enter a non-numeric values into the textboxes, the error message will display the text "One of the entries is not a number! Try again!" If no error occurs, it will display the correct answer. Try it out yourself.

The Code

```
Public Class Form1

Private Sub CmdCalculate_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CmdCalculate.Click

Lbl_ErrorMsg.Visible = False

Dim firstNum, secondNum As Double

On Error GoTo error_handler

firstNum = Txt_FirstNumber.Text

secondNum = Txt_SecondNumber.Text

Lbl_Answer.Text = firstNum / secondNum
```

```
Exit Sub 'To prevent error handling even the inputs are valid

error_handler:

Lbl_Answer.Text = "Error"

Lbl_ErrorMsg.Visible = True

Lbl_ErrorMsg.Text = " One of the entries is not a number! Try again!"

End Sub

End Class
```

The Output

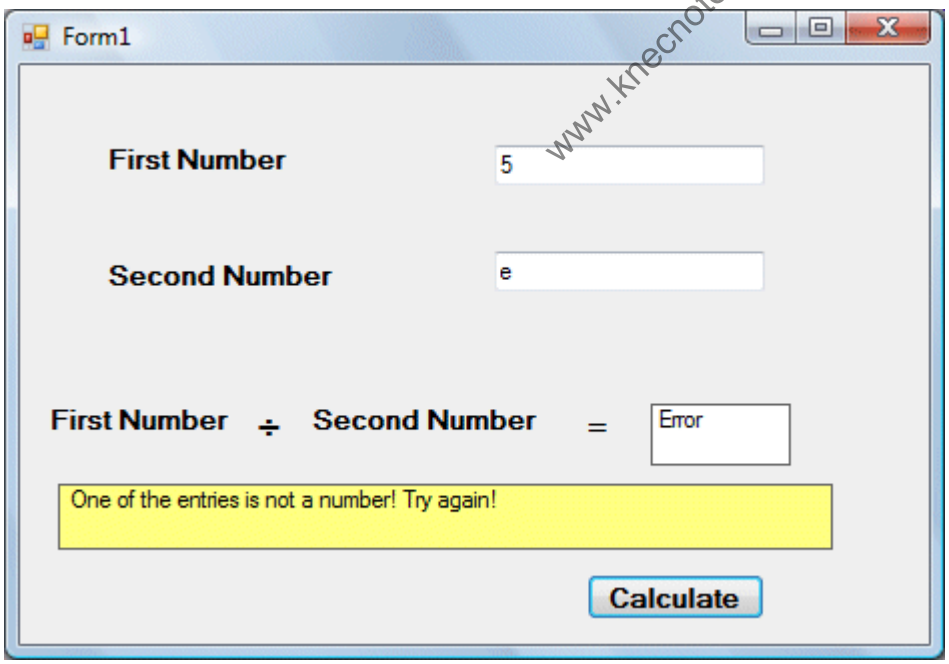

Figure 18.1

18.3 Errors Handling using Try...Catch...End Try Structure

VB2010 has adopted a new approach in handling errors, or rather exceptions handling. It is supposed to be more efficient than the old On Error Goto method, where it can handles various types of errors within the Try...Catch...End Try structure.

The structure looks like this

```
Try
statements
Catch exception_variable as Exception
statements to deal with exceptions
End Try
```


Example 18.2

This is a modification of Example 18.1. Instead of using On Error GoTo method, we use the Try...Catch...End Try method. In this example, the Catch statement will catch the exception when the user enters a non-numeric data and return the error message. If there is no exception, there will not any action from the Catch statement and the program returns the correct answer.

The code

```
Public Class Form1

Private Sub CmdCalculate_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CmdCalculate.Click

Lbl_ErrorMsg.Visible = False

Dim firstNum, secondNum, answer As Double

Try

firstNum = Txt_FirstNumber.Text
```

```
secondNum = Txt_SecondNumber.Text

answer = firstNum / secondNum

Lbl_Answer.Text = answer

Catch ex As Exception

Lbl_Answer.Text = "Error"

Lbl_ErrorMsg.Visible = True

Lbl_ErrorMsg.Text = " One of the entries is not a number! Try again!"

End Try

End Sub

End Class
```

The output is shown in Figure 18.2

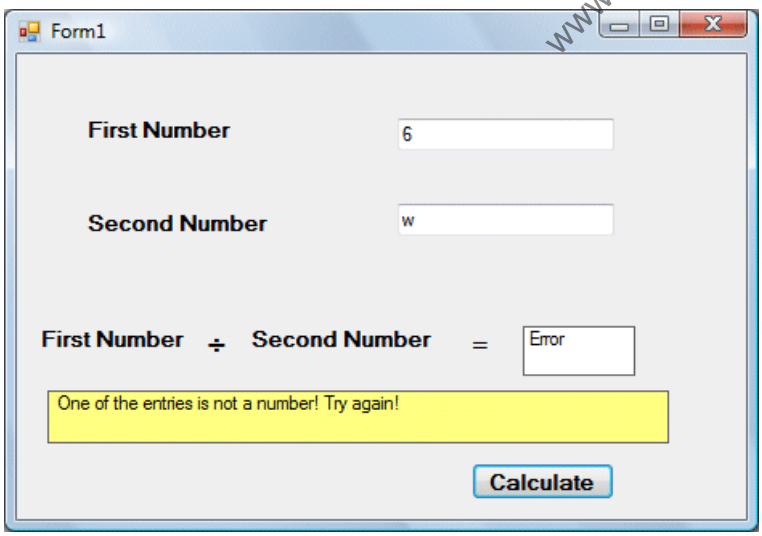

Figure 18.2

Summary

- In section 18.1, you learned the basic principle of handling errors.
- In section 18.2, you learned how to use On Error Goto Syntax.
- In section 18.3, you learned how to write code to handle errors using Try...Catch...End Try structure.