Visual Web Development

Terry Marris September 2008

3 *Exceptions*

We see how to enable, disable, hide and make visible, controls such as labels, text boxes and buttons. We also see how to create and make use of procedures and how to create our own exceptions. An exception is just a fancy word for an error situation.

3.1 Use Case

We shall write an implementation of the use case shown below. It is almost identical to the one shown previously in Text Input Output, except this time there is an explicit pre-condition and an Exceptions section.

Use Case	Exceptions
Goal	to display a personalised greeting
Pre-condition	a none-empty name is entered
Post-condition	a personalised greeting is displayed
Initiating Actor	the user

Main Success Scenario

1 system prompts user for their identity or name
2 user enters their id or name
3 system displays a greeting
4 exit success

Exceptions

2a user enters nothing
  1 system displays error message
  2 resume 2

The Use Case is named Exceptions.

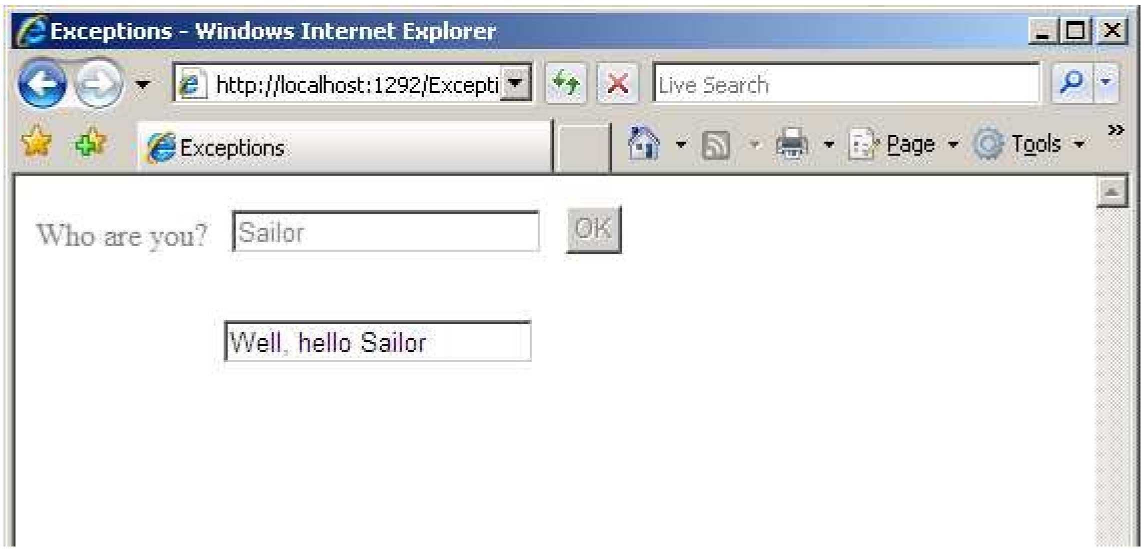
The pre-condition is that the user enters at least one character for their name.

Exceptions are what can go wrong; they are the error cases. There is just one error case here: the user enters nothing, in which case, an error message is displayed and control returns to step 2 of the main success scenario.

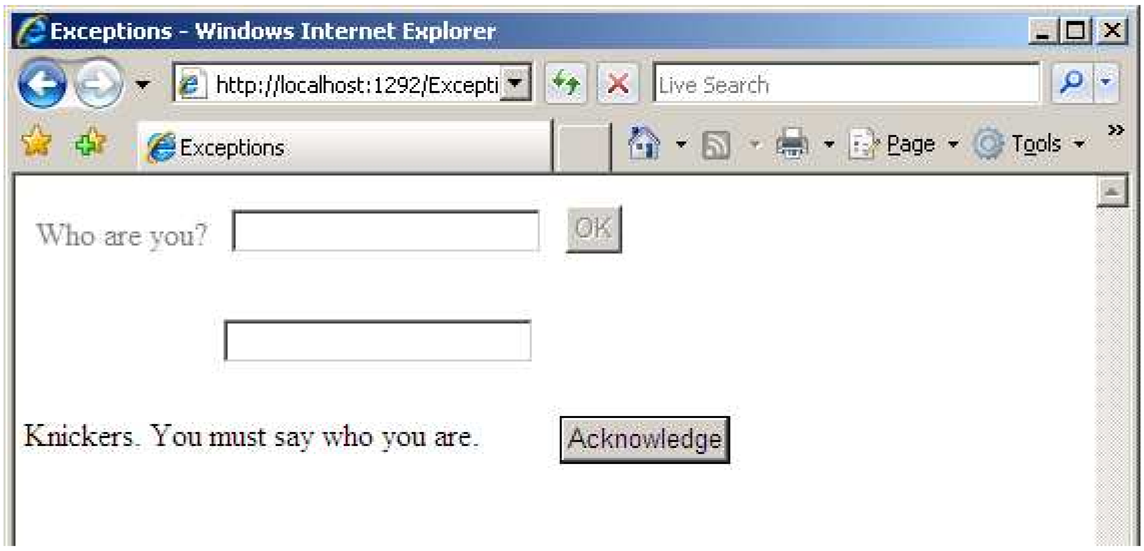
Us programmers are like gods: we want to remain in control at all times.

3.2 Examples of Program Runs

Here, the user has entered *Sailor* in response to the prompt *Who are you?* and clicked OK (or pressed Enter). The message Well, hello Sailor is shown and the prompt, text input and OK button are disabled.



Here the user has entered nothing and clicked OK (or pressed Enter). The error message is shown along with a button that is required to be pressed to acknowledge that the message has been seen.

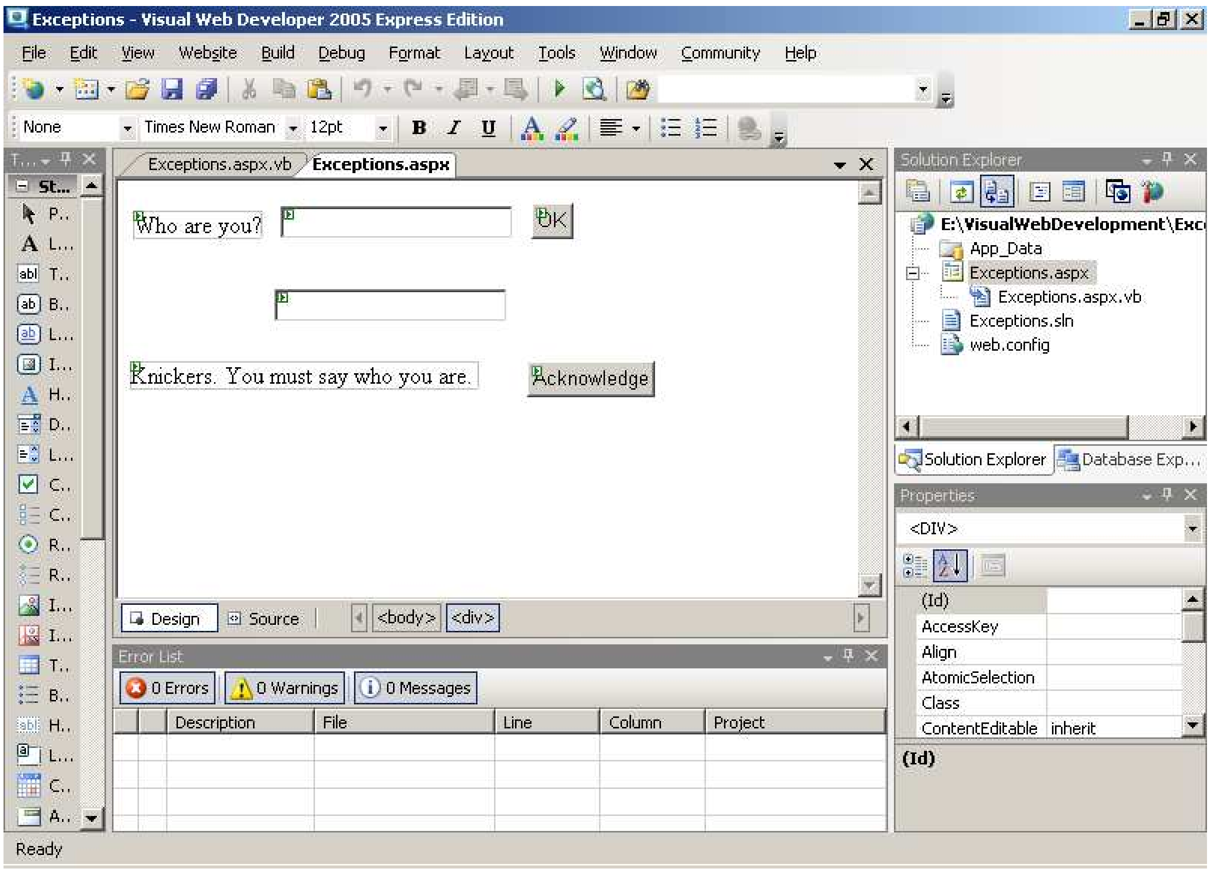

Exceptions - Windows Internet Explorer
http://localhost:1292/Excepti
Live Search
Exceptions
Page
Tools
Who are you?
OK
Knickers. You must say who you are.
Acknowledge

To those of us who want to use a MsgBox() - we cannot because we are dealing with Web Forms and not VB Forms.

3.3 Form Design

We drag and drop labels, text boxes and buttons onto the form as shown below. We set the following properties:

Control	Property	Value
Form	File Name Title	Exceptions.aspx Exceptions
Label	ID Text	lblPrompt Who are you?
Text Box	ID Text	txtName <blank>
Button	ID Text	btnOK OK
Text Box	ID ReadOnly Text	txtOutput True <blank>
Label	ID Text Width	lblError Knickers. You must say who you are. 232px (Could be set by dragging the control)
Button	ID Text Width	btnAck Acknowledge 85px


Exceptions - Visual Web Developer 2005 Express Edition
File Edit View Website Build Debug Format Layout Tools Window Community Help
None Times New Roman 12pt
Exceptions.aspx.vb Exceptions.aspx
Who are you?
OK
Knickers. You must say who you are.
Acknowledge
Design Source <body> <div>
Error List
0 Errors 0 Warnings 0 Messages
Description File Line Column Project
Solution Explorer
E:\VisualWebDevelopment\Exc
App_Data
Exceptions.aspx
Exceptions.aspx.vb
Exceptions.sln
web.config
Solution Explorer Database Exp...
Properties
<DIV>
(Id)
AccessKey
Align
AtomicSelection
Class
ContentEditable inherit
(Id)
Ready

3.4 Program Design

The entire program is shown below.

```
Partial Class _Default
    Inherits System.Web.UI.Page

    Sub EnableInput()
        lblPrompt.Enabled = True
        txtName.Enabled = True
        btnOK.Enabled = True
    End Sub

    Sub DisableInput()
        lblPrompt.Enabled = False
        txtName.Enabled = False
        btnOK.Enabled = False
    End Sub

    Sub HideError()
        lblError.Visible = False
        btnAck.Visible = False
    End Sub

    Sub ShowError()
        lblError.Visible = True
        btnAck.Visible = True
    End Sub
```

Procedures may be written by ourselves.

A procedure defines one small task. It starts with the VB word Sub and ends with the VB words End Sub.

A procedure is named by ourselves, and we always chose descriptive names, e.g. EnableInput(), and always include brackets.

Controls can be enabled or disabled, made visible or made invisible. A disabled control is shown greyed out, but you can still see it.

```
    Protected Sub btnOK_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnOK.Click
        Try
            If txtName.Text.Length() = 0 Then
                Throw New Application.Exception()
            End If
```

If the user enters nothing we say, "Hey, that's an error"

```
            txtOutput.Text = "Well, hello " + txtName.Text
            DisableInput()
```

We display a greeting

```
        Catch ex As ApplicationException When txtName.Text.Length() = 0
            DisableInput()
            ShowError()
            btnAck.Focus()
        End Try
    End Sub
```

But if the user has entered nothing we display a polite error message

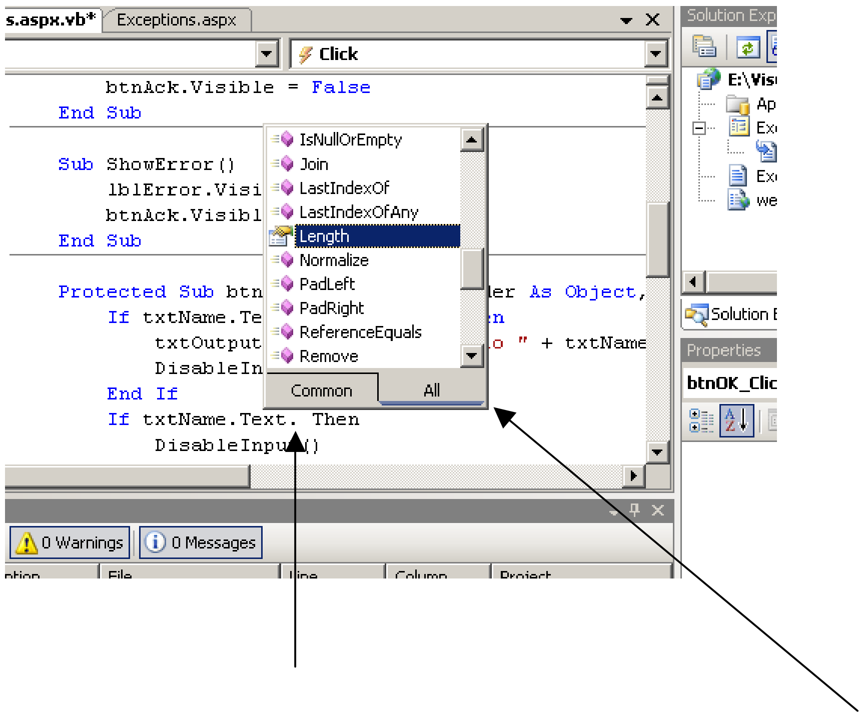
```
    Protected Sub btnAck_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnAck.Click
        HideError()
        EnableInput()
        txtName.Focus()
    End Sub

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        EnableInput()
        HideError()
        txtName.Focus()
    End Sub
End Class
```

Click on an empty part of the form to start the Page_Load procedure. On start up we want to enable the input, hide the error message and acknowledge button, and set the focus on the txtName text box.

3.5 The Intellisense System

The Intellisense system is the (sometimes) intelligent system that suggests what you can use. You can choose and accept what is suggested (by selecting it and pressing return), or you can ignore it and carry on typing, or press the Escape key.



```
        btnAck.Visible = False
    End Sub

    Sub ShowError()
        lblError.Visi
        btnAck.Visibl
    End Sub

    Protected Sub btn                        ler As Object,
        If txtName.Te                        n
            txtOutput                        o " + txtName
            DisableIn
        End If
        If txtName.Text. Then
            DisableInput()
```

Here the programmer has just entered the dot following Text and the (intelligent) suggestion list pops up.

3.6 The Exception

```
Try
    If txtName.Text.Length() = 0 Then
        Throw New System.Exception()
    End If

    txtOutput.Text = "Well, hello " + txtName.Text
    DisableInput()

Catch ex As Exception When txtName.Text.Length() = 0
    DisableInput()
    ShowError()
    btnAck.Focus()
End Try
```

If the user enters nothing we say, "Hey, that's an error"

If there is no error we display a greeting and disable the input

But if the user has entered nothing we disable the input, display a polite error message and enable the Acknowledge button

The entire coding is enclosed within Try ... End Try.

First, we deal with the error case - OK button clicked but nothing in the input text box. The error is that the length of the text in the input text box is zero. We throw a new exception - which is caught and dealt with later.

Then we deal with the non-error case: we display a greeting and then disable the input because we have finished.

Now we catch the error and deal with it. We disable the input, show a helpful error message and enable the Acknowledge button - which has to be pressed if the user is to continue.

Exeptions is perhaps the best way to deal with errors. We can either write our own, as we have done here, or use exceptions provided by the language, as we shall do in the next chapter, Numbers IO.

3.7 Exercise

1. Try out the Exceptions program shown above.