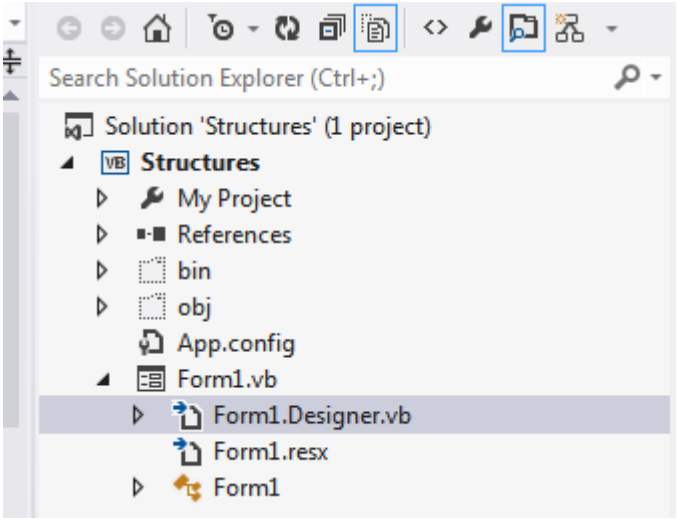
Week 1 – dynamic controls

Creating a pattern of buttons or check boxes on a screen is a tedious task. In fact, (almost) any routine task in the programming world can be automated. You can create dynamic controls which even respond to user events. These notes show you how.

This is also known as creating controls at run-time or programmatically creating controls.

IN fact, all the controls you place on the form at design time are created at run-time in code, the code is just hidden from you. To see the code which generates your form, use Solution Explorer:

1. Click the button at the top of the window that “Shows all Files”
2. Expand the triangle alongside your form and double click on the Designer.vb file. This is the code that is automatically generated as you drag and drop controls onto the form. Be careful not to change it, unless you know what you are doing.



Dynamic Controls

Dynamic controls are useful where a pattern of controls is required, such as representing seats in a booking system.

If you need to add a handler for your new control (so that it responds to user events), the code is included as well.

Example: The code to generate a set of coloured bars (Picture Boxes) is:

```
Dim x As Integer = 10

 'create 11 new picture boxes
 For n As Integer = 0 To 10

     'make a new picture box
     Dim temp As New PictureBox
     With temp
         'set the properties
         'position and size
         .Left = x
         .Width = 5
         .Top = 10
         .Height = 40
```

```
                'give it a colour
                .BackColor = Color.Blue
                'add it to the form Controls collection
                'so that it is automatically redrawn
                Me.Controls.Add(temp)
            End With
            'calculate the next position of the next picture box
            x = x + 7
        Next
```

The critical steps in the process are:

Add the new control to the form 'Controls' collection. This makes sure the control is re-drawn every time the form is repainted.

```
Me.Controls.Add(temp)
```

Calculate the new position of the next control. In this case, only the x-coordinate was changed. If you don't use a different location, every new control will be drawn over the top of the previous one, and it looks like nothing has happened.

```
x = x + 7
```

Adding a handler

You can add handlers to your dynamic controls, so they respond to events in the same way as other controls. The bad news is, you have to code each handler routine yourself. The steps are:

1. Create a handler – use any name, but the **parameters must match** the event you are handling.

```
Private Sub PicBoxClick(sender As Object, e As EventArgs)
    MsgBox(sender.name.tostring)
End Sub
```

2. Use the 'Add Handler' function to add this handler to the control

```
AddHandler temp.Click, AddressOf PicBoxClick
```

The first argument to AddHandler is the event of the control you've created (and it has to be something the control can actually do). The second argument is prefaced by "AddressOf" and is the name of your sub-routine.

Grids

You can use nested 'for' loops to create a grid of controls. This code creates a set of buttons, which could represent the seats in a theatre.

The code for the form (fSeats) is included. Note that to use the 'sender' in the handler will cause an error unless you turn off Option Strict – as in the line of code before `Public Class fSeats`.

```
Option Strict Off
Public Class fSeats

    Private Sub fSeats_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim rows As Integer = 5
        Dim cols As Integer = 6
        Dim width As Integer = 50
        Dim space As Integer = 3

        Dim r, c As Integer
        For r = 0 To rows
            For c = 0 To cols
                Dim temp As New Button
```

```
                With temp
                    .Width = width
                    .Height = width
                    .Left = c * (space + width)
                    .Top = r * (space + width)
                    .Text = r & "-" & c
                    .Name = "btn" & r & " " & c
                    AddHandler temp.Click, AddressOf myhandler
                End With
                Me.Controls.Add(temp)
            Next
        Next
    End Sub

    Private Sub myhandler(sender As Object, e As EventArgs)
        sender.backcolor = Color.Red
    End Sub
End Class
```