Review of the Data Entry Controls

There is a lot of content here. To cover the topic in one webpage it takes a lot of information and explanation. Don’t worry if you spend an afternoon (4 or 5 hours) with this document (making a webpage that experiments with the content) you will be fast-forwarding your understanding very quickly until you can use each of the webpage controls with ease and mastery.

Partial Class SimpleControls

 Inherits System.Web.UI.Page

 'This program demonstrates the common data entry controls. As the programs get bigger it helps to place the data into regions. You can open up any of the regions by clicking the + sign. To make a region type #Region "name of the region" and place the term #End Region at the end of the code for that secion

#Region "Textboxes"

 Protected Sub Clickme(sender As Object, e As EventArgs) Handles Button13.Click

 'a textbox is used to gather textual data from the human. Common data entered include name, address, etc. The textbox is also used to gather numbers from the human, but this is a bit problematic as the numbers typed into the textboxes are considered text so you have to convert them to decimal values such as using convert.todecimal.

 'Textboxes are also used for output (display) to screen. You can use a textbox with its selection mode set to multiline so you can display multiple lines of text. When you want to display a number in a textbox, you have to convert it back to text. You can use the built in .tostring or .tostring("C0") function as follows: txtOutput.text = dectotal.tostring("C0"). The ("C0") formats the number with commas where appropriate and a $, and here 0 decimal places. You can also use N for numeric (justs adds commas between the thousands, and you can use P for %.

 'Later in the training we use gridviews (aka gridviews) to easily show tables of numbers such as displaying the data from an array.

 Dim decNumber As Decimal

 If IsNumeric(TextBox8.Text) Then 'this is a test condition. Lines 18-21 will be run if a number was typed

 decNumber = Convert.ToDecimal(TextBox8.Text)

 'it is easy to increase the value stored in a decimal as follows

 decNumber += 1000

 txtOutput.Text = "The number was changed to " & decNumber.ToString("N0")

 Else

 'in case the program user typed a word then reverse the term for fun. Lines 18-21 will NOT be run if a number was typed

 txtOutput.Text = "Is the secret meaning of your words " & StrReverse(TextBox8.Text) & " ?"

 txtOutput.Text &= vbNewLine & vbNewLine & "a built-in function named StrReverse was used to reverse the entered data. This string reverse trick provided as a hint to the versatility of Visual Studio.NET"

 End If

 End Sub

 Protected Sub Button14\_Click(sender As Object, e As EventArgs) Handles Button14.Click

 'this is how you clear out any prior content from a textbox.

 txtOutput.Text = Nothing

 TextBox8.Text = Nothing

 End Sub

 Protected Sub Button2\_Click(sender As Object, e As EventArgs) Handles Button2.Click

 'the textbox has a selectionmode of date, or datetime, or datetimelocal. One or more of these selection modes should work (some web browsers do not render the control correctly). Rather than use .longdate you can also use .shortdate formatting, change it and see. BTW if you change around any of the code in this webpage, you'll have some fun!

 txtOutput.Text = "Date selected was " & FormatDateTime(TextBox2.Text, DateFormat.LongDate)

 End Sub

#End Region

#Region "Imagebutton"

 Protected Sub ImageButton1\_Click(sender As Object, e As ImageClickEventArgs) Handles ImageButton1.Click

 'The textmode property of the textbox is set to password. You can still capture and verify the password that was typed however. There is a control called an imagebutton, it works

 If txtPassword.Text = Nothing Then 'just don't do anything if no password was typed.

 Exit Sub

 End If

 'you can evaluate what was typed and generate different functionality depending on a condition. So the code below reads ...evaluate the value captured into the text property of the txtPassword textbox. IN CASE they typed swordfish, write a msg to the outputbox, IN CASE they type Go Cougs output a different message. In all other cases just output the password that was typed.

 Select Case txtPassword.Text 'this is the test condition

 Case "swordfish"

 txtOutput.Text = "locate the map under your seat to go to start the next adventure"

 Case "Go Cougs"

 txtOutput.Text = "The strength of the MIS network is the Coug and the strength of the Coug is the MIS network"

 Case Else 'on all other circumstances

 txtOutput.Text = txtPassword.Text

 End Select

 End Sub

#End Region

#Region "Linkbutton - also check the navigation section below"

 Protected Sub LinkButton1\_Click(sender As Object, e As EventArgs) Handles LinkButton1.Click

 'this is a linkbutton - it looks like a hyperlink, but actually is used to run a procedure

 If IsNumeric(TextBox3.Text) = False OrElse TextBox3.Text = Nothing OrElse Convert.ToInt16(TextBox3.Text) < 1 OrElse Convert.ToInt16(TextBox3.Text) > 10 Then

 txtOutput.Text = "check that data is numeric in the range 1-10"

 Exit Sub

 End If

 'this code will only run if all the test conditions were satisfied. You often have to spend time and due diligence with error checking!

 Select Case Convert.ToInt16(TextBox3.Text)

 Case 1 To 3

 txtOutput.Text = "simplicitiy in life can unlock its innate quality"

 Case 4 To 6

 txtOutput.Text = "moderation enables a long, peaceful life, free of disease"

 Case 7 To 10

 txtOutput.Text = "a busy life is a good life"

 End Select

 End Sub

#End Region

#Region "Checkboxes"

 Protected Sub Button4\_Click(sender As Object, e As EventArgs) Handles Button4.Click

 'The checkbox is a yes/no field useful for storing a boolean true or false value which is useful for program changes or for example saving a true/false value into a VIP column. You can also use the checkbox to alter program flow or processes as shown.

 Dim decPrice As Decimal = 4.5 'you can create a local variable and set its value in one line

 If CheckBox1.Checked = True Then 'VIP selected

 decPrice -= 2 'looks like a $2 discount

 txtOutput.Text = "VIP client, offer a free lollipop and charge " & decPrice.ToString("C2")

 Else 'only run this next if if teh checkbox was not selected.

 txtOutput.Text = "Charge the client " & decPrice.ToString

 End If

 End Sub

 Protected Sub CheckBox2\_CheckedChanged(sender As Object, e As EventArgs) Handles CheckBox2.CheckedChanged

 If CheckBox2.Checked = True Then 'the Oahu checkbox was selected

 Image1.Visible = True

 Else 'the checkbox was not selected

 Image1.Visible = False

 End If

 'probably not important both it is interesting that both of these object have true/fase properties (something about the control can be evaluated as either true or false). Here the image control will be set to visible when the checkbox is checked. The checkbox has a .checked property that is true when the control was selected and false when the checkbox was not selected.

 ' Image1.Visible = CheckBox2.Checked

 End Sub

#End Region

#Region "Checkboxlists with multi-select"

 Protected Sub Button5\_Click(sender As Object, e As EventArgs) Handles Button5.Click

 'This section is pretty advanced for the first week do not worry about fully understanding it yet. The main demonstration is that you have to sometimes used advanced techniques to enable advanced functionality.

 'This is a checkbox control (a type of list control that allows multiple selections. This is the preferred control to allow the human to select more than one item from the list. This content is not presented for quite a while - in the module for loops. It is interesting to see it work though.

 'More detail provided in a later module, but each list control has an items collection. A bunch of items which are the options for the control, that the program user can select.

 'we create a variable of datatype ListItem that allows us to exam each of the items in the list of items. All of the list controls (radiobuttonlist, checkboxlist, listbox) have an items collection (aka items list). You the developer either type in the items when designing the webpage, or you can download a list from a column in a database (shown later). So we examine each item and if it was selected (checked) then the text for the item (here flavors of ice cream) are added to a string variable and later displayed. Check the items collection of the control to see the options.

 MaintainScrollPositionOnPostBack = True

 Dim Flavor As ListItem

 Dim strListofSelectedFlavors As String 'this variable used to build the list of flavors

 'Since we need to examine each of the items on the list we need a way to examine every item on the list of items. A loop can be used to run the same code over and over. that looks at each item one at a time. This loop looks at each of the items in the list of items. If the item is selected it is added to a variable and displayed. Pretty cool?

 For Each Flavor In CheckBoxList1.Items 'for each item on the list of items for the checkbox. Run this loop once for each item on the list of flavors. So 4 times. Use a loop to run code 4 times. Much more on loops another day.

 If Flavor.Selected = True Then

 strListofSelectedFlavors &= Flavor.Text & vbNewLine

 End If

 Next

 'now display the list of flavors

 TextBox4.Text = "The list of flavors is: " & strListofSelectedFlavors

 End Sub

#End Region

#Region "Dropdownlist, Listbox, RadiobuttonList"

 Protected Sub Button6\_Click(sender As Object, e As EventArgs) Handles Button6.Click

 MaintainScrollPositionOnPostBack = True

 'Dropdown lists are used quite often as they simplify user entry and reduce the number of data entry errors. After all the human cannot type whatever they like, they are limited to make a selection from a list (and we can make sure they selected something on the list). The items on the list are numbered beginning at index 0. The item that is selected is captured in a property called the selectedindex. If nothing was selected on the list then the .selectedindex would be -1. We can use this test to see if a selection was made, and if not, stop the program.

 'In some list controls such as a dropdown list (not a radiobutton list or checkbox list) we use the 0 item to prompt the program user to make a selection. If the 0 numbered item "make a selection" item was selected or nothing was selected then we stop the program and prompt the program user. That is the meaning of the next line of code.

 If DropDownList1.SelectedIndex <= 0 Then 'if a valid selection was not made then just stop. 0 here is not a valid selection

 TextBox6.Text = "There are " & DropDownList1.Items.Count - 1 & " items on the list, please make a selection from the list "

 Exit Sub

 Else

 'Dropdownlists, radiobuttonlists and a listbox all work the same. They have a .selecteditem.text property that allows us to see and display the textual item. They also have a .selectedvalue property that holds a numeric value that was previously input. Pretty handy. This next line of code would replace about a dozen lines of select case code, back in the old days. One the .aspx page look at the properties for the dropdownlist and specifically look at the .items collection. You will see a .text and a .value entries for each item on the list. The program designer sets these when they design their program. The .text and .value properties can also be pulled from columns in a database table.

 TextBox6.Text = DropDownList1.SelectedItem.Text & " charge " & DropDownList1.SelectedValue

 End If

 End Sub

 Protected Sub Button7\_Click(sender As Object, e As EventArgs) Handles Button7.Click

 'This listbox can also be used as a multi-select, you would have to hold the control button down on your keyboard as you are clicking on the items. You do have to change the selectionmode of the control to allow multiple selections. You would also have to implement looping functionality similar to as shown above. We do not use the listbox in multi-select mode, but you should know it is possible.

 'a nice trick is shown here, you can add more lines of text to a textbox by using the concatenator symbol &=

 If ListBox1.SelectedIndex > -1 Then 'so 0 here is a valid selection

 TextBox5.Text = "Jingio Spa Experience" & vbNewLine

 TextBox5.Text &= "The index # of the item selected was " & ListBox1.SelectedIndex & vbNewLine

 TextBox5.Text &= ListBox1.SelectedItem.Text & " charge " & ListBox1.SelectedValue

 Else

 TextBox5.Text = "Make a selection from the list, before click the button, thanks. "

 Exit Sub

 End If

 End Sub

 Protected Sub Button10\_Click(sender As Object, e As EventArgs) Handles Button10.Click

 MaintainScrollPositionOnPostBack = True

 'being able to capture the values in the .selecteditem.text and .selectedvalue properties of the control makes coding much easier.

 'in the old days you had to use a select case. now below we have just a few lines of code for the same functionality

 'The radiobutton list is a control where only one of the items from the list can be selected (such as small, medium, large). The code is similar as the dropdownlist. This control has a repeatdirection property that lets you display the items vertically or horizontally.

 If RadioButtonList1.SelectedIndex > -1 Then

 TextBox5.Text = RadioButtonList1.SelectedItem.Text & " state charge " & RadioButtonList1.SelectedValue

 End If

 End Sub

#End Region

#Region "Labels"

 Protected Sub Button8\_Click(sender As Object, e As EventArgs) Handles Button8.Click

 MaintainScrollPositionOnPostBack = True

 'labels are just used to display content (text and numbers). You can also use a textbox for this functionality. Here when the label was added to the form, it's text property displayed the question. When this procedure runs, an answer to the question is concatenated (added to the label)

 Label1.Text &= " to learn how to love"

 End Sub

#End Region

#Region "Series of radiobuttons"

 Protected Sub RadioButton1\_CheckedChanged(sender As Object, e As EventArgs) Handles RadioButton1.CheckedChanged

 'a radiobuttonlist is a great productivity improvement. You do not need to look at the following code for more than a minute, it is a peek into the dark abyss of older programming that was needed to generate required functionlality. Visual Studio.NET has evolved and improved over the years (such as adding radiobuttonlists and checkboxlists).

 'Look at all this work it used to take to make a series (group) of checkboxes work together. At least you get to see a new event - the checkchanged event. This code runs when the radiobutton is selected or deselected. Notice these three procedures execute when a radiobutton is clicked on. you have to manually deselect the other two radiobuttons. The checkbox list is designed for multi-selection, radiobuttons are not.

 If RadioButton1.Checked = True Then

 RadioButton2.Checked = False

 RadioButton3.Checked = False

 TextBox9.Text = "Customer wants a " & RadioButton1.Text

 End If

 End Sub

 Protected Sub RadioButton2\_CheckedChanged(sender As Object, e As EventArgs) Handles RadioButton2.CheckedChanged

 If RadioButton2.Checked = True Then

 RadioButton1.Checked = False

 RadioButton3.Checked = False

 TextBox9.Text = "Customer wants a " & RadioButton2.Text

 End If

 End Sub

 Protected Sub RadioButton3\_CheckedChanged(sender As Object, e As EventArgs) Handles RadioButton3.CheckedChanged

 If RadioButton3.Checked = True Then

 RadioButton1.Checked = False

 RadioButton2.Checked = False

 TextBox9.Text = "Customer wants a " & RadioButton3.Text

 End If

 End Sub

 Protected Sub Button11\_Click(sender As Object, e As EventArgs) Handles Button11.Click

 'To clear out all selections made in the checkboxes

 RadioButton1.Checked = False

 RadioButton2.Checked = False

 RadioButton3.Checked = False

 TextBox9.Text = Nothing

 End Sub

#End Region

#Region "Calendars"

 Protected Sub Button3\_Click(sender As Object, e As EventArgs) Handles Button3.Click

 MaintainScrollPositionOnPostBack = True

 'calendars are very easy to code against, the problem is that the control is large. The .selectedDate property is used to capture the day(s) selected in the calendar. You can format the output in many different ways (ie. tolongdatestring, etc.). The control is older, and being replaced by the textbox's date .textmode.

 TextBox7.Text = Calendar1.SelectedDate.ToShortDateString 'so the formatted date is shorter

 End Sub

#End Region

#Region "Navigation"

 'depending on which linkbutton was clicked on a differet view (sortof like a tab) of the multiview is displayed.

 Private Sub LinkButton3\_Click(sender As Object, e As EventArgs) Handles LinkButton3.Click

 MultiView1.ActiveViewIndex = 2

 End Sub

 Private Sub LinkButton4\_Click(sender As Object, e As EventArgs) Handles LinkButton4.Click

 MultiView1.ActiveViewIndex = 0

 End Sub

 Private Sub LinkButton5\_Click(sender As Object, e As EventArgs) Handles LinkButton5.Click

 MultiView1.ActiveViewIndex = 1

 End Sub

 Private Sub SimpleControls\_Init(sender As Object, e As EventArgs) Handles Me.Init

 'run this code when th epage loads up for the first time

 MultiView1.ActiveViewIndex = -1

 End Sub

#End Region

End Class