Dealing with Multi-select controls

http://cb-ot-devst05.ad.wsu.edu/featherman/IntroDevClass/Mod2/radiobuttonlist.aspx

This program demonstrates how to use loops to handle multi-selection controls. The checkbox list and listbox are the multi-select controls.

A FOR EACH …NEXT loop is used to examine each and every item in the checkbox list. You can also use a FOR NEXT loop with some sort of counter but that does not seem natural, just check each of them right?

This started out to be a relatively simple program, but you the designer need to run the program in unexpected ways (that the program user will use) to see if the program crashes. For example the price of the toppings should only be calculated if a pizza size was selected, so we checked for that and other likely user errors. Featherman calls this making your code error-proof. It’s a good idea to give your program to a friend preferably a non-developer to see if they can crash your program. The designer knows how the program should be used, but must anticipate how it will be used in unpredictable ways.

This program uses a loop to solve a problem and is the best solution in this case. New developers often fall in love with loops such as writing a term into a textbox 100,000 times. Loops are very, very useful. Be sure to learn SQL programming though because when trying to perform calculations with loops, there is probably an easier way to do the aggregation process, such as with SQL GROUP BY() or PIVOT() queries.

'this program demonstrates working with multi-select controls. We demonstrate with a checkboxlist, however the same methodology could also be used with a listbox control. Each of the items in the checkbox list's items collection (ie items list) has both a .text and a .value property that the programmer sets when they create the program (or derived from a SQL query)

'This program shows how to build a string variable with all the names of the items selected and uses a decimal variable to accumulate the values of the items selected. A for Each loop is used to examine each and every item in the items collection of the multi-select checkbox list. If the item is checked, then the .text name is added to the string variable and the .value number is added to the decimal variable resulting in a list of items selected and the price for all the items selected.

'This started out to be a relatively simple program, but you the designer need to run the program in unexpected ways to see if the program malfunctions. For example the price of the toppings should only be calculated if a pizza size was selected, so we checked for that and other likely user errors. Featherman calls this making your code error-proof.

Partial Class IntroDevClass\_radiobuttonlist

 Inherits System.Web.UI.Page

 Protected Sub RadioButtonList1\_SelectedIndexChanged(sender As Object, e As EventArgs) Handles RadioButtonList1.SelectedIndexChanged

 'this code runs when an item in the radiobutton list is selected. When the program starts (or the control is cleared of previous selections) the index of the item changes from -1 (meaning nothing selected) to a larger number (the index number of the item selected). This change in the index number from -1 to another number fires this procedure and the rest of the code runs

 ' You need to make your decision though where to set the .items or .values of a list control a) in code in the page\_init procedure either typed in or from a SQL query or b) in html in the sourceview (this is actually the old standard), or c) in the items property of the control when you are creating the program. Recall that you are transitioning from a consumer of web pages to a designer of web pages. The designer has a lot to set-up.

 'This code will only run if an item was selected so there is no need for that error checking to see if an item was selected. Here we create the variable and assign it the selected value of the control all in one line of code. Anything to save an extra line of code right? The selected value in converted to decimal. The CDEC() funciton is the same as convert.todecimal

 Dim decTotal As Decimal = CDec(RadioButtonList1.SelectedValue)

 'the selectedItem.text is the word from the list, BE CAREFUL TO NOT FORGET THE FINAL .TEXT you will get a strange error message that is hard to decipher

 txtTicketPrice.Text = "Charge " & FormatCurrency(decTotal) & " to that " & RadioButtonList1.SelectedItem.Text

 End Sub

 Protected Sub btnClearRBL1\_Click(sender As Object, e As EventArgs) Handles btnClearRBL1.Click

 'Clear the selection from the radiobuttonlist and the textbox. The setting the index to -1 deselects any prior item selection

 RadioButtonList1.SelectedIndex = -1

 txtTicketPrice.Text = Nothing

 End Sub

 Protected Sub rblPizzas\_SelectedIndexChanged(sender As Object, e As EventArgs) Handles rblPizzas.SelectedIndexChanged

 'this procedure runs if an item was selected in the radiobutton list so no error checing is needed to see if a selection was made from the list. We clear the toppings selcted to ensure the program user selects them again and calculates their price. Once code is placed into a general procedure (not tied to any control being clicked then it can be called - envoked - from anywhere is the program that is useful).

 txtPizzaPrice.Text = "Price of " & rblPizzas.SelectedItem.Text & ": " & rblPizzas.SelectedValue

 Call ClearToppings()

 End Sub

 Protected Sub chkToppings\_SelectedIndexChanged(sender As Object, e As EventArgs) Handles chkToppings.SelectedIndexChanged

 'Here is the solution to the interface problem of multiple select. The code examines each and every item in the items collection of the checkboxlist. If the item was selected (a pizza topping) then the cost for that item needs to be added to the running total so you can figure out the cost of the pizza.

 'How does it do this? By using the for each item in the items collection LOOP. Don't fall in love with loops, but in this case they are the best solution. The code iterates (loops) the items collection list, runn the code between the FOR EACH IN and NEXT once for each item in the list. If the item is checked then the word is added to the variable in effect buildding a list, and the cost of that item added to a variable in effect calculating the cost of teh pizza toppings. Recall the items collection of a list control is actually a vertical stack of numerically indexed items each having a text and value field defined. Here we create a listitem object to examine each item in the items list of the checkboxlist (this same approach would work for any multi-select list control).

 'So the "For each in list" type of loop will examine each item in the full list to see if it was indeed checked. Another time we will use a FOR EACH data row in a datatable of rows.

 'We use an if statement inside the loop to see if teh item on the list was selected. If it was selected then we capture the text and numeric value for the item. If the item was selected the string variable is appended using concatenation, and the local decimal variable is accumulated. These first 2 variable are set to have no value. This is actually not necessary, but some developers are overly cautious. A local variable is declared anew when the event occurs (the event here is shown by the handles term - Handles chkToppings.SelectedIndexChanged) so there is no need to worry if it has a value...but...

 Dim strToppingsList As String = Nothing

 Dim decToppingsCost As Decimal = 0

 Dim decPizzaPrice As Decimal

 'We better check to see if any of the pizza toppings were selected from the list, some people prefer jsut cheese pizza with no toppings. Such a boring pizza if no anchovies no? Anyway copy the cost of the pizza into the variable, otherwise stop the procedure from proceding.

 If rblPizzas.SelectedIndex > -1 Then

 decPizzaPrice = rblPizzas.SelectedValue

 Else

 txtPizzaPrice.Text = "Please select pizza size first"

 Call ClearToppings()

 Exit Sub

 End If

 'now calculate the price of the toppings, here is where we loop each and every item in the items collection (list) of the checkbox list.

 If chkToppings.SelectedIndex > -1 Then

 'this item is used to iterate through the list, don't put it inside the loop

 Dim toppings As ListItem

 For Each toppings In chkToppings.Items

 'examine each item in the list, if it is checked then grab the word and value. Increment the price of toppings and overall price of the pizza. You could add up the price of the toppings and add it to the base price for the pizza OUTSIDE the loop if you prefer.

 If toppings.Selected = True Then

 strToppingsList &= toppings.Text & ", "

 decToppingsCost += toppings.Value

 decPizzaPrice += toppings.Value

 End If

 Next

 End If

 'ok show the output. Notice that you can provide currency formatting to the end of a parentheses encapsulated caclulation. Here we add in 6% tax.

 txtPizzaPrice.Text = "Base price for " & rblPizzas.SelectedItem.Text & vbNewLine & rblPizzas.SelectedValue & vbNewLine & vbNewLine & "Price for the " & strToppingsList & "toppings: " & decToppingsCost.ToString("N2") & vbNewLine & vbNewLine & "Charge " & (decPizzaPrice \* 1.06).ToString("C2") & " for pizza with tax"

 End Sub

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

 Call ClearToppings()

 rblPizzas.SelectedIndex = -1

 txtPizzaPrice.Text = Nothing

 End Sub

 Private Sub ClearToppings()

 'notice this procedure is not connected to any event occuring - there is no handles syntax. We check each item on the list and see if it was selected, if it is that de-select it.

 Dim toppings As ListItem

 For Each toppings In chkToppings.Items

 'examine each item in the list, if it is checked then add it to the list

 toppings.Selected = False

 Next

 End Sub

End Class