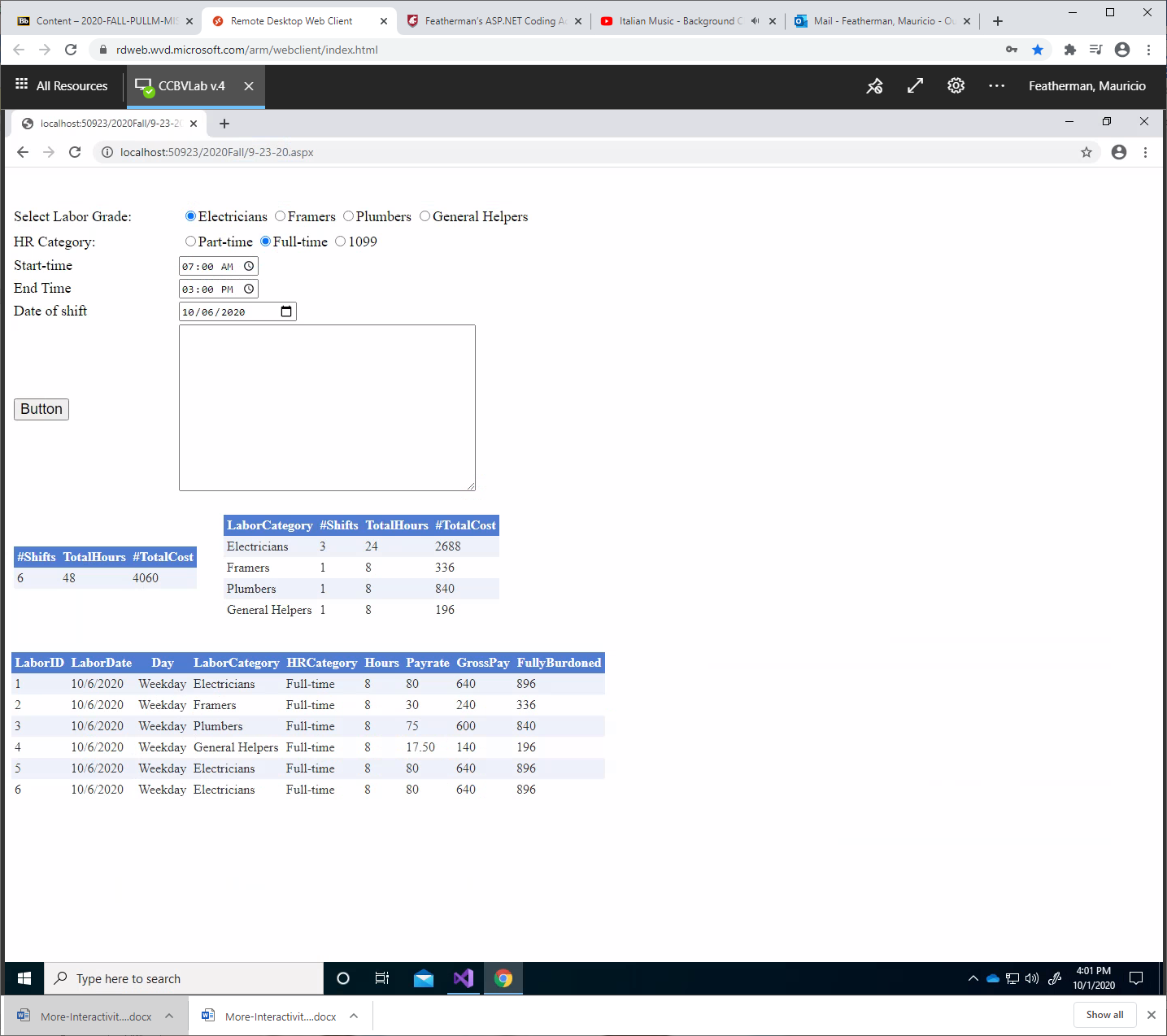
**HW Helper - Arrays**  
  
Use this document if you are having trouble   
  
a) loading the first column of a summary array, or  
  
b) updating the right row of an array (e.g., updating the Electricians total shifts, hours, cost after another shift was recorded for an electrician.

Other documentation shows how to do these tasks quickly and perhaps even ingeniously. This document shows an easier to implement method.

In the first scenario above an easy but verbose way of doing this is shown in yellow below.

In the second scenario a SELECT CASE() is used – this is easy to understand, but is also verbose. This code is in blue below.

Imports System.Data

Partial Class \_2020Fall\_9\_23\_20

Inherits System.Web.UI.Page

Public Shared gdtPayrollRecords, gdtLaborGradeTotals, gdtGrandTotals As New DataTable

#Region "Create Arrays"

Private Sub \_2020Fall\_9\_23\_20\_Init(sender As Object, e As EventArgs) Handles Me.Init

'we need to use an .init procedure NOT the page\_load procedure. The Init procedure runs only once

'here we add columns to the database table. We first check if the columns were already added, and exit if they were.

If gdtPayrollRecords.**Columns.Count > 0** OrElse gdtGrandTotals.**Columns.Count > 0** Then

Exit Sub 'already added the columns

End If

With gdtPayrollRecords '

.Columns.Add("LaborID", GetType(Integer)) 'first we add a column with the datatype integer

.Columns.Add("LaborDate", GetType(String))

.Columns.Add("Day", GetType(String))

*Use the integer datatype only for the first column such as vendorID, employeeID, transactionID.*

*For any numbers use decimal datatype. Also we are not doing any timespan calculations of the date so we store the labordate value using a string – so we can format it nicely.*

.Columns.Add("LaborCategory", GetType(String))

.Columns.Add("HRCategory", GetType(String))

.Columns.Add("Hours", GetType(Decimal))

.Columns.Add("Payrate", GetType(Decimal))

.Columns.Add("GrossPay", GetType(Decimal))

.Columns.Add("FullyBurdoned", GetType(Decimal))

End With

*Use the integer datatype only for the first column such as vendorID, employeeID, transactionID.*

*For any numbers use decimal datatype. Also we are not doing any timespan calculations of the date so we store the labordate value using a string – so we can format it nicely.*

'add constraints or autonumbering or initial values to any column

'gdtPayrollRecords.Columns("LaborID").AutoIncrement = True

*This is how you make the first column auto-number starting at 1, and incrementing 1 at a time*

'gdtPayrollRecords.Columns("LaborID").AutoIncrementSeed = 1

'gdtPayrollRecords.Columns("LaborID").AutoIncrementStep = 1

With gdtPayrollRecords.Columns("LaborID")  
 .AutoIncrement = True  
 .AutoIncrementSeed = 1  
 .AutoIncrementStep = 1  
 End With

*This is another way to rewrite the code above. Does it look easier to read? Many people say the WITH END WITH code makes code easier to read when it is repetitive.*

'this is the update table for grand totals

With gdtGrandTotals 'here we create the 3 columns for our grandtotals table, setting initial values to 0

.Columns.Add("#Shifts", GetType(Decimal))  
 .Columns.Add("TotalHours", GetType(Decimal))  
 .Columns.Add("#TotalCost", GetType(Decimal))

*IMPORTANT!!!  
When you want to use an array for running totals, you MUST set the values to zero. Here we set the .defaultvalue to 0*

.Columns("#Shifts").DefaultValue = 0  
 .Columns("TotalHours").DefaultValue = 0  
 .Columns("#TotalCost").DefaultValue = 0

End With

Dim dr As DataRow = gdtGrandTotals.NewRow 'this row will display the totals

gdtGrandTotals.Rows.Add(dr)

'now create the columns and rows for the laborgrade totals update table (summary)

'step 2 'here we create the 3 columns for our grandtotals table, setting initial values to 0

If gdtLaborGradeTotals.Columns.Count > 0 Then  
 Exit Sub  
 End If

With gdtLaborGradeTotals

*IMPORTANT!!!  
When you want to use an array for running totals, you MUST set the values to zero. Here we set the .defaultvalue to 0*

.Columns.Add("LaborCategory", GetType(String))  
 .Columns.Add("#Shifts", GetType(Decimal))  
 .Columns.Add("TotalHours", GetType(Decimal))  
 .Columns.Add("#TotalCost", GetType(Decimal))

.Columns("#Shifts").DefaultValue = 0  
 .Columns("TotalHours").DefaultValue = 0  
 .Columns("#TotalCost").DefaultValue = 0

End With

'step 3 add the rows to the array so that we can edit their values

Dim dr1 As DataRow = gdtLaborGradeTotals.NewRow  
 Dim dr2 As DataRow = gdtLaborGradeTotals.NewRow  
 Dim dr3 As DataRow = gdtLaborGradeTotals.NewRow  
 Dim dr4 As DataRow = gdtLaborGradeTotals.NewRow

'Step 4: here we populate the first column of the array

dr1.Item("LaborCategory") = "Electricians"  
 dr2.Item("LaborCategory") = "Framers"  
 dr3.Item("LaborCategory") = "Plumbers"  
 dr4.Item("LaborCategory") = "General Helpers"

'now add the rows to the datatable

gdtLaborGradeTotals.Rows.Add(dr1)  
 gdtLaborGradeTotals.Rows.Add(dr2)  
 gdtLaborGradeTotals.Rows.Add(dr3)  
 gdtLaborGradeTotals.Rows.Add(dr4)

'here we display the array

GridView3.DataSource = gdtLaborGradeTotals  
 GridView3.DataBind()

End Sub  
#End Region

Protected Sub Button1\_Click(sender As Object, e As EventArgs) Handles **Button1.Click**

'this procedure is used to

'a) add rows of data to a datatable

'b) then update the grand totals table

Page.MaintainScrollPositionOnPostBack = True

Dim ts As TimeSpan

Dim dr As DataRow = gdtPayrollRecords.NewRow

' here we add the values into the new row

dr("LaborDate") = DateTime.Parse(txtDate.Text).ToShortDateString

Select Case Weekday(DateTime.Parse(txtDate.Text))

Case 1

dr("Day") = "Sunday"

Case 2 To 6 'week

dr("Day") = "Weekday"

Case 7 'saturday

dr("Day") = "Saturday"

End Select

dr("LaborCategory") = rblLaborGrades.SelectedItem.Text

dr("HRCategory") = rblHRCategory.SelectedItem.Text

ts = DateTime.Parse(txtEnd.Text) - DateTime.Parse(txtStart.Text)

dr("Hours") = ts.TotalHours

dr("Payrate") = rblLaborGrades.SelectedValue

'here calculate the gross pay

dr("GrossPay") = ts.TotalHours \* rblLaborGrades.SelectedValue

Select Case rblHRCategory.SelectedIndex

Case 0 'pt

dr("FullyBurdoned") = ts.TotalHours \* rblLaborGrades.SelectedValue

Case 1 'ft

dr("FullyBurdoned") = (ts.TotalHours \* rblLaborGrades.SelectedValue) \* 1.4

Case 2 '1099

dr("FullyBurdoned") = (ts.TotalHours \* rblLaborGrades.SelectedValue) \* 1.15

End Select

'now that we have data in each column of the row, add the row tot he datatable (array)

gdtPayrollRecords.Rows.Add(dr)

'now display the data using the new gridview control

GridView1.DataSource = gdtPayrollRecords  
 GridView1.DataBind()

'now let's update the **grand totals table**

**gdtGrandTotals**.Rows(0).Item("#Shifts") += 1  
 **gdtGrandTotals**.Rows(0) .Item("TotalHours") += ts.TotalHours  
 **gdtGrandTotals**.Rows(0).Item("#TotalCost") += dr("FullyBurdoned")

'this prior line takes the value that was stored in the transaction table and uses that to increase the value in a column of the summary table. Now display the **grand totals table**

GridView2.DataSource = gdtGrandTotals  
 GridView2.DataBind

‘now update the running totals for one labor grade

Select Case rblLaborGrades.SelectedItem.Text

*This is an easy way to control which row of data gets updated. The row numbers are 0,1,2,3*

*Using SELECT CASE lets you control the processing. Notice the code within each CASE statement is near identical, except for which row gets updated.*

**Case "Electricians"**

With gdtLaborGradeTotals.Rows(**0)**  
 .Item("#Shifts") += 1  
 .Item("TotalHours") += ts.TotalHours  
 .Item("#TotalCost") += dr("FullyBurdoned")  
 End With

**Case "Framers"**

With gdtLaborGradeTotals.Rows(1)  
 .Item("#Shifts") += 1  
 .Item("TotalHours") += ts.TotalHours  
 .Item("#TotalCost") += dr("FullyBurdoned")

End With

**Case "Plumbers"**

With gdtLaborGradeTotals.Rows(2)  
 .Item("#Shifts") += 1  
 .Item("TotalHours") += ts.TotalHours  
 .Item("#TotalCost") += dr("FullyBurdoned")  
 End With

*This code has the same functionality but does not use an WITH END WITH*

**Case "General Helpers"**

gdtLaborGradeTotals.Rows(3).Item("#Shifts") += 1  
 gdtLaborGradeTotals.Rows(3).Item("TotalHours") += ts.TotalHours  
 gdtLaborGradeTotals.Rows(3).Item("#TotalCost") += dr("FullyBurdoned")

End Select

'display updated labor grade totals table

GridView3.DataSource = gdtLaborGradeTotals  
 GridView3.DataBind()

End Sub  
End Class