



School of Information Technology

Practical 02: Microsoft Visual Studio Web Application Development with C#.

OBJECTIVES:

By the end of this Practical students should be able to:

- Create an Empty ASP.NET Website.
- Add a Master Page.
- Add controls to a Web Form.
- Understand the relationship between the properties of a control in ASP code and the Properties window.
- Change the properties of controls in:
 - a. *Properties* window.
 - b. ASP code in *Source* view.
- Work with events and event handlers.
- Understand how ASP.NET binds event handlers to controls.
- Understand how the Properties window displays the event handlers bound to controls.

Introduction

Today, we will use Visual Studio to create a simple CPF Calculator Web Application.

An employee who works for a company will need to contribute to his CPF account which consists of the Ordinary, Special and Medisave Account. The company he works for will also need to contribute to his CPF account. The company's contribution is separate and is NOT taken from the employee's salary.

The contribution rates by the employee and his employer are shown in the table below.

Table 1

Employee Age (years)	Contribution By Employer (% of salary)	Contribution By Employee (% of salary)	Credited Into		
			Ordinary Account (Ratio of Con)	Special Account (Ratio of Con)	Medisave Account (Ratio of Con)
35 & below	14.5	20	0.6667	0.1449	0.1884
Above 35 to 45	14.5	20	0.6088	0.1739	0.2173
Above 45 to 50	14.5	20	0.5509	0.2028	0.2463
Above 50 to 55	10.5	18	0.4562	0.2456	0.2982
Above 55 to 60	7.5	12.5	0.575	0	0.425
Above 60 to 65	5	7.5	0.28	0	0.72
Above 65	5	5	0.1	0	0.9

Allocation of CPF Contributions

The CPF contributions are allocated to the Ordinary, Special and Medisave Accounts based on the ratio of contributions shown in

Table 1. Contributions are first allocated to the Medisave Account, followed by the Special Account. The balance is then allocated to the Ordinary Account.

Example:

If a employee's salary is \$4000 and his age is 35 and below,

CPF Contribution

Employer contribution = $4000 * 14.5\% = 580$

Employee contribution = $4000 * 20\% = 800$

Total CPF contribution = $800 + 580 = 1380$

Employee's Net Salary

Employee Net Salary = $4000 - 800 = 3420$

Amounts Credited to each Account

Special Account = $0.1449 * 1380 = 199.96$

Medisave Account = $0.1884 * 1380 = 259.99$

Ordinary Account = $1380 - 199.96 - 259.99 = 920.05$

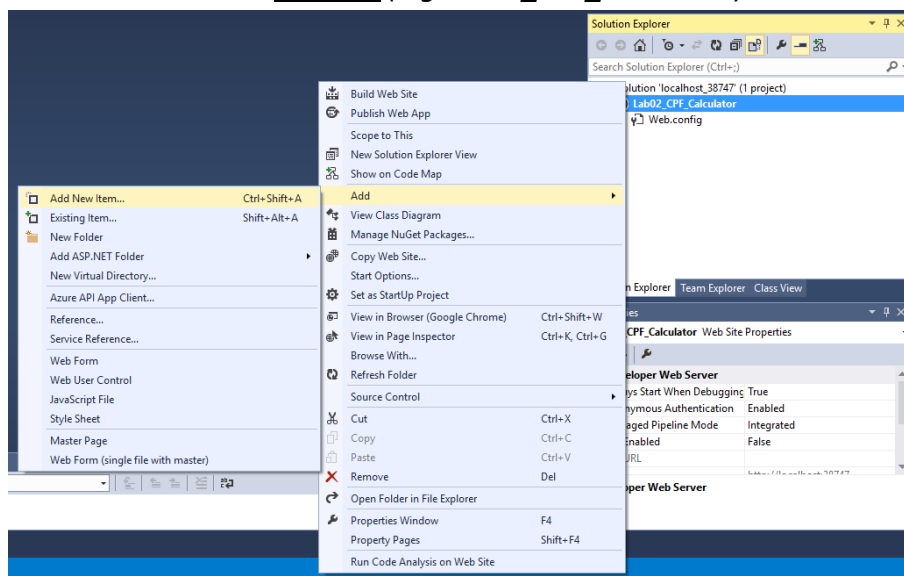
Important: We calculate the Ordinary Account by subtracting Special Account and Medisave Account from the CPF contribution to account for errors that may be introduced by rounding up. This is a very important for financial calculations and other calculations that need to accurately account for every unit.

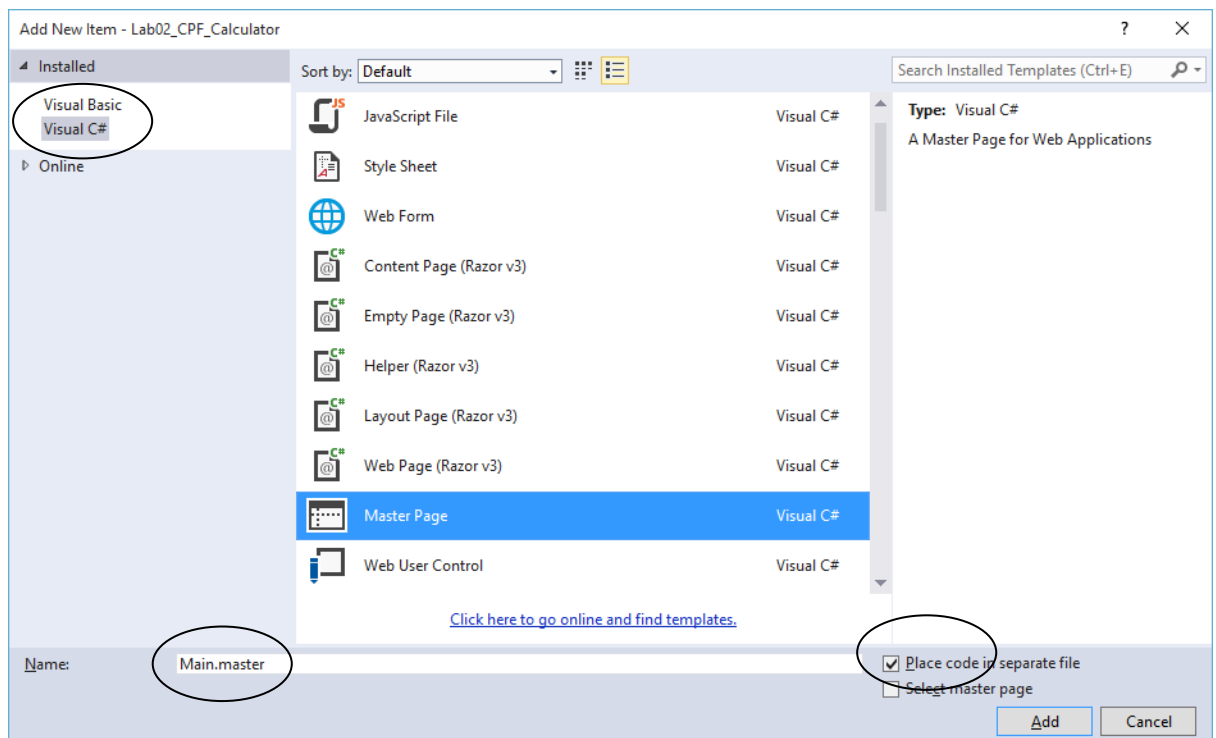
Exercise 1

We will first create a prototype of the application where the contribution is calculated using the rates for age 35 and below.

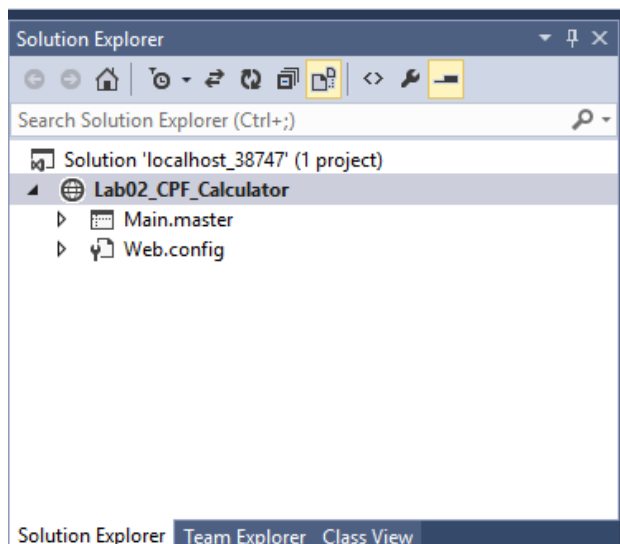
Creating the UI

1. Create a **new empty website** by:
 - a. Select **File** ➔ **New Web Site....**
 - b. In the *New Web Site* dialog:
 - i. Under *Installed Templates*, select *Visual C#*.
 - ii. Select *ASP.NET Empty Web Site*.
 - iii. For the Web Location, use *Lab02_CPF_Calculator* as the folder name.
2. Add a Master Page to the web site by:
 - a. RIGHT-CLICK Website (e.g *Lab02_CPF_Calculator*) ➔ **Add** ➔ **Add New Item....**



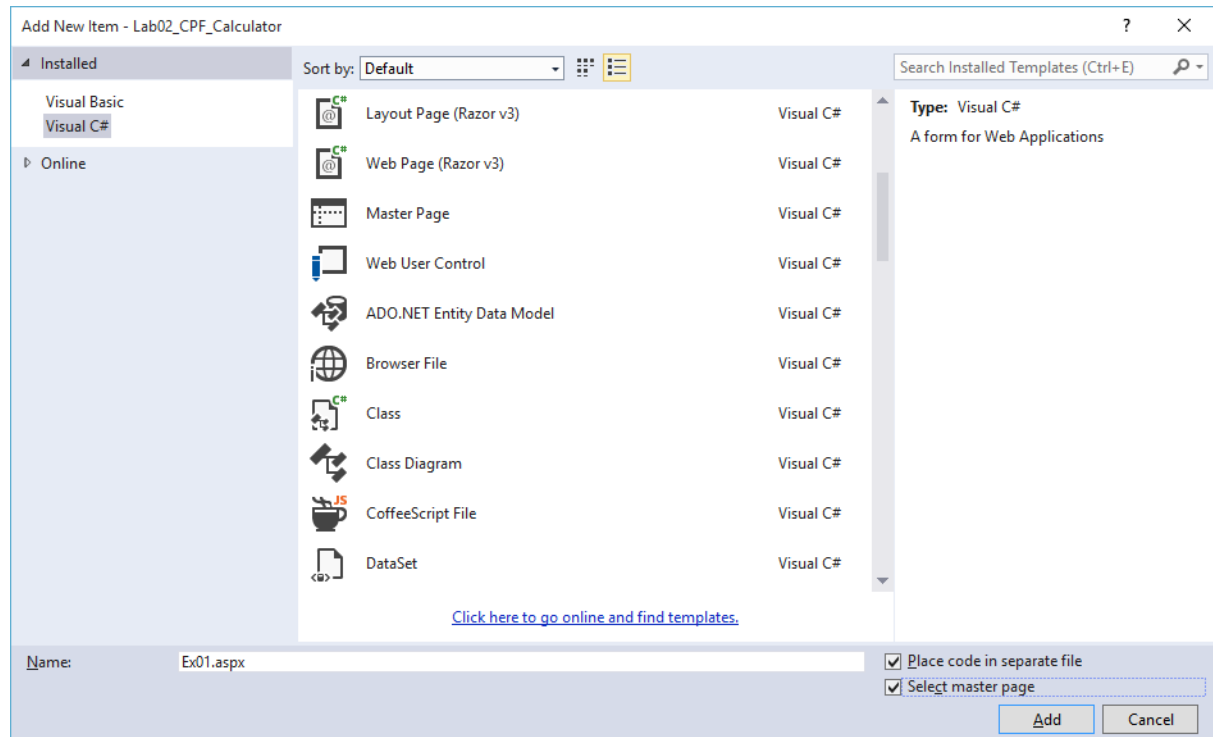


- Select *Visual C#*.
- Select *Master Page*.
- Change the file name to *Main.master*.
- Tick *Place code in separate file*.
- Un-tick *Select master page*.
- You should get an additional file *Main.master* in your solution explorer.

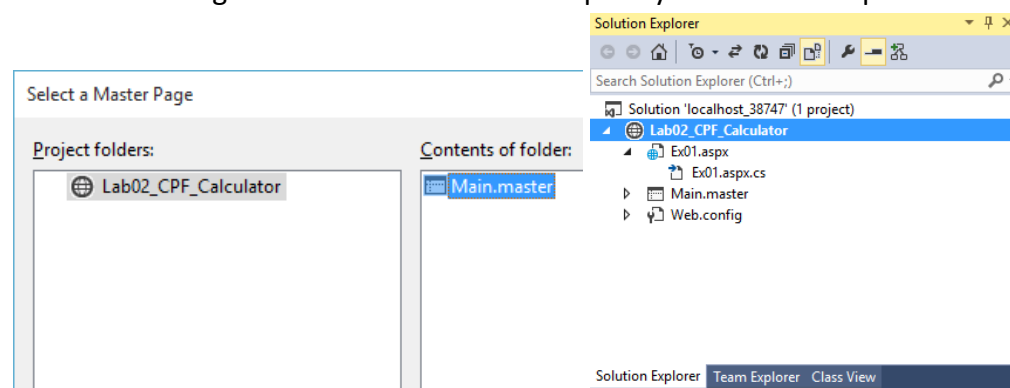


TIP: When you create any significant ASP.NET Web Application, it is a good idea to create a Master Page and then make sure all your ASPX web pages use the Master Page. If the Master Page is not needed eventually, you can just leave it blank. If it is needed, then you don't need to make major changes to the ASPX web pages just to add a Master Page.

3. Add a new ITEM into the website - C# **Web Form** named *Ex01.aspx* (the 3rd character is the number zero and NOT the letter 'O'):
 - a. RIGHT-CLICK Website (e.g *Lab02_CPF_Calculator*) → *Add New Item...*
 - b. Add a new “Web Form” named *Ex01.aspx*.
 - i. With code behind (hint: which box do you need to tick?), and
 - ii. Use *Main.master* as the master page (hint: which box do you need to tick?).

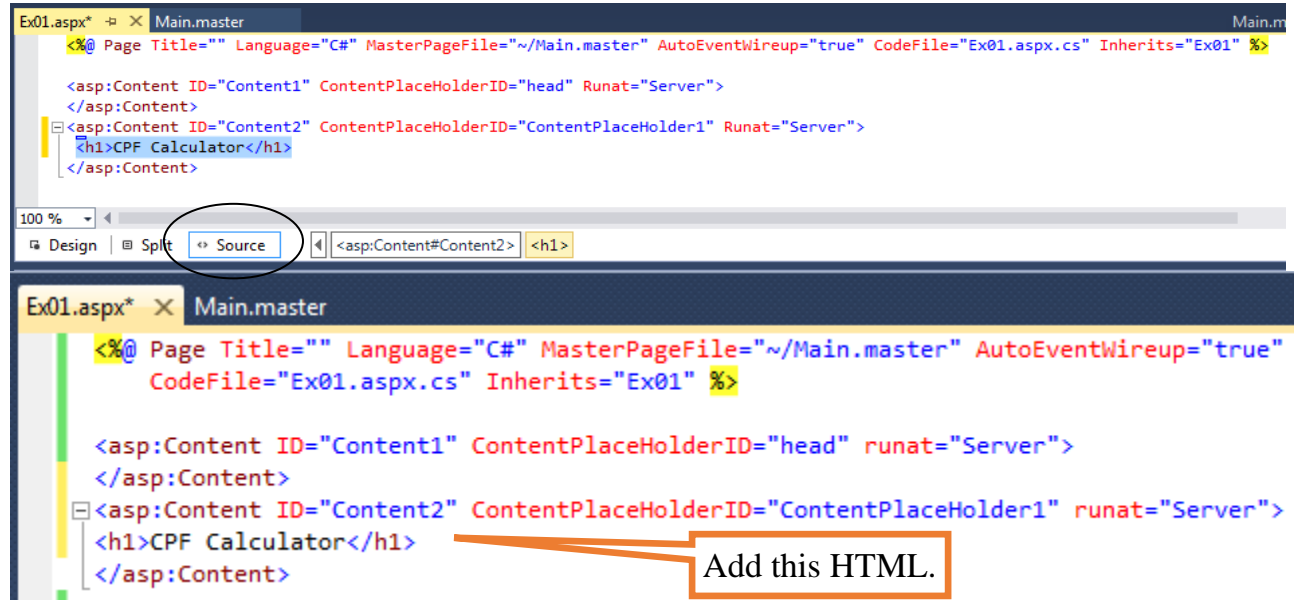


4. You should get an additional file *Ex01.aspx* in your solution explorer.

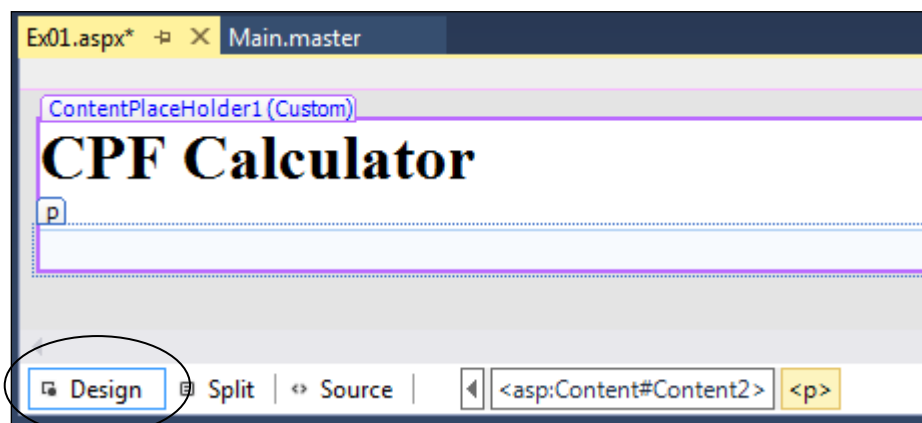


5. Double-clicked “Ex01.aspx”
6. While in “Source” view, add the HTML codes shown in Figure 1 to *Ex01.aspx*.

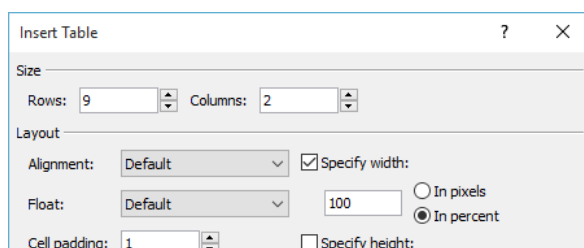
Figure 1

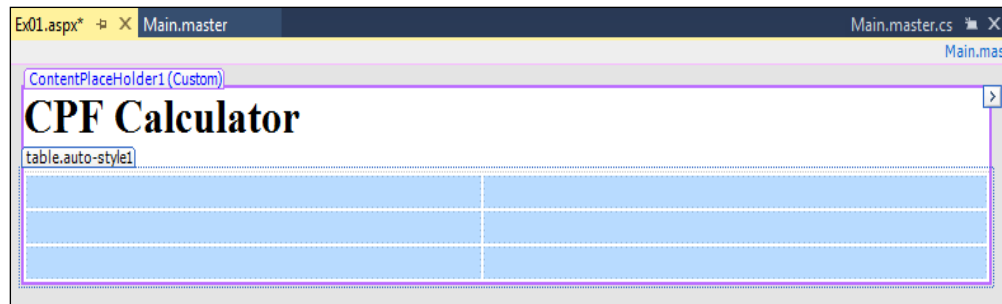


7. **TIP:** Press Ctrl + K + D on your keyboard. If there is nothing wrong with your code, VS will auto format your code neatly with proper indentation.
8. Switch to "Design" view of *Ex01.aspx* and insert a table with 3 rows and 2 columns by:

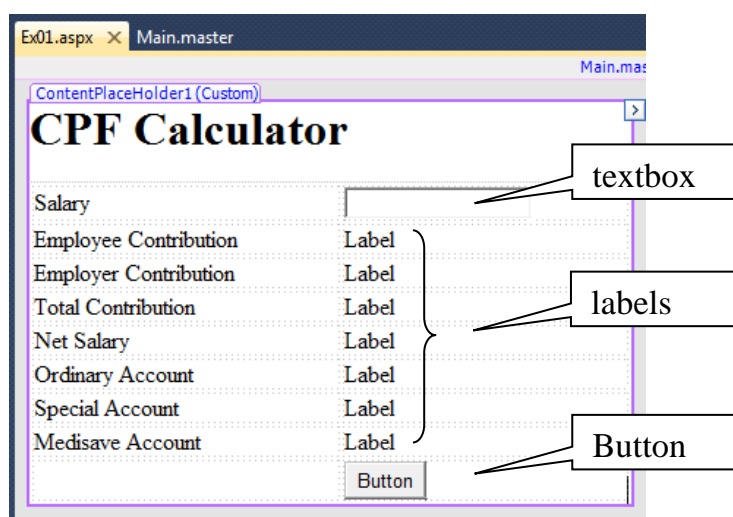


- a. Place the keyboard cursor after the words "CPF Calculator" and press the "Enter" key on your keyboard.
- b. From the menu bar, select *Table* → *Insert Table*.
- c. In the *Insert Table* dialog, set 9 rows and 2 columns.
- d. Click the OK button.

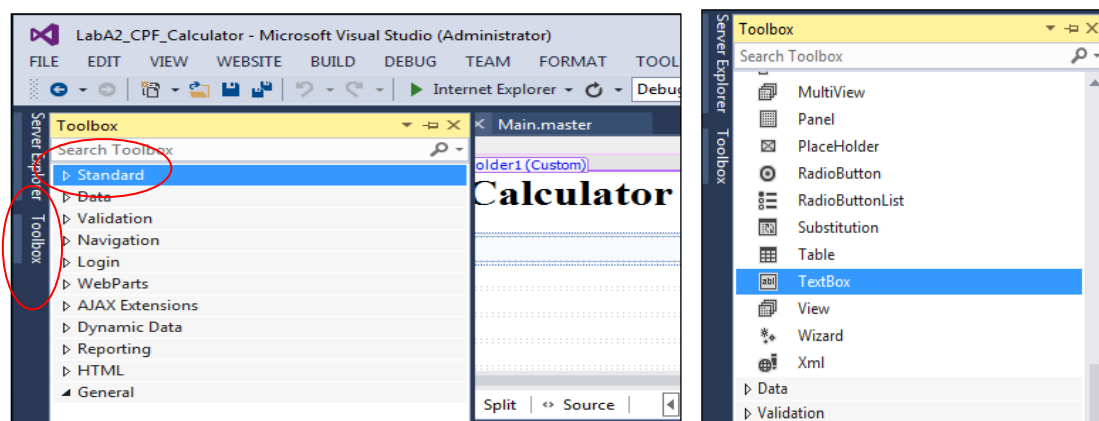




9. You should now have a table on your *Ex01.aspx* web form.
10. In the next few steps we will attempt to drag-drop some Toolbox controls onto the table.
11. The screenshot below will be out final outcome.

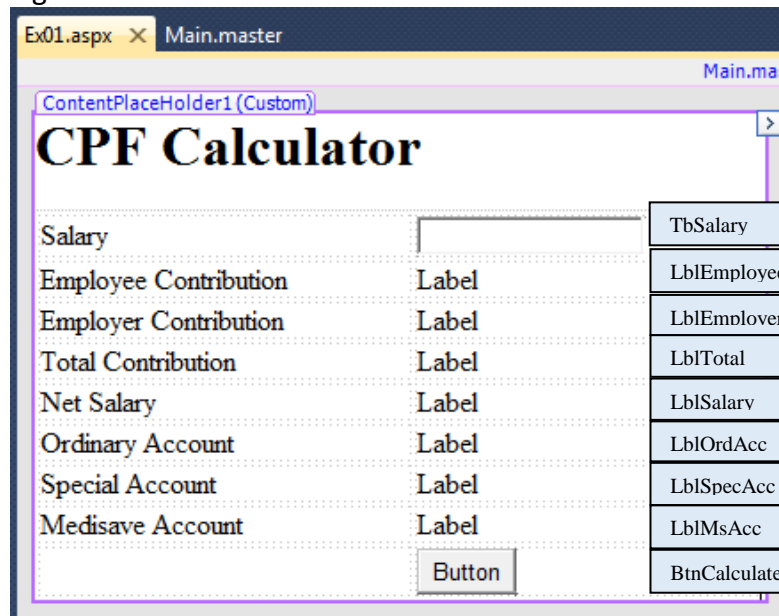


12. From the *Toolbox* panel, look inside the “Standard” toolbox for a *Textbox* control
If you do not see your toolbox, click on *View – Toolbox (Ctrl-Alt-X)* to display the toolbox.



13. Drag-drop the *Textbox* control into the cell at 1st row, 2nd column. (When you do that, the table might resize its columns. That is normal as we have not fixed the width of the columns.)

Figure 2



Important: From your previous programming modules, you would have learned the importance of Naming Conventions, to make our programming tasks easier. To make it easier to recall the purpose of each of the controls on our web form, we should adopt a Naming Convention to set the IDs of the controls to reflect their Type and Purpose. For this module, we will use a prefix to indicate the Type of the control and a suffix to indicate the Purpose of the control.

We will use **Tb**, **Lbl** and **Btn** to as the prefix of the *IDs* of **Textboxes**, **Labels** and **Buttons** respectively.

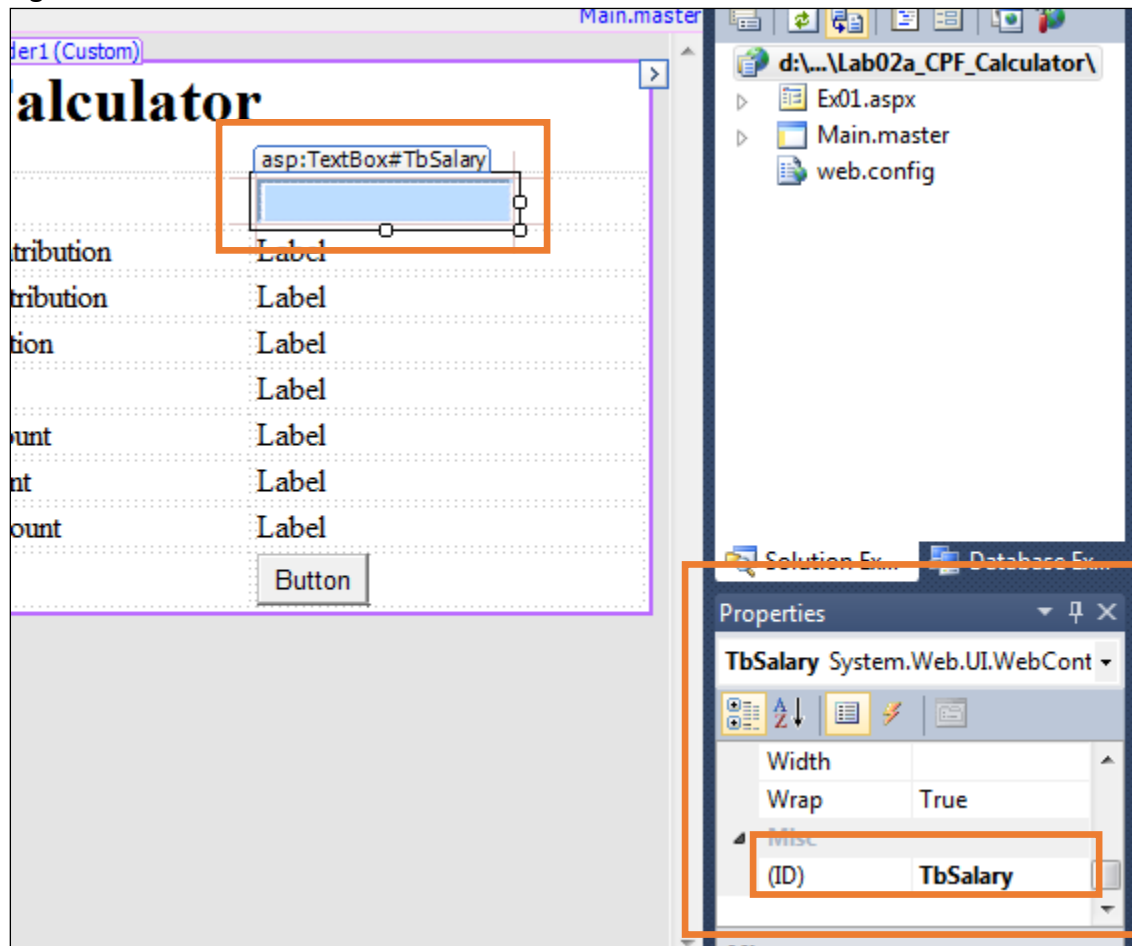
So, according to our naming convention, the *Textbox* in Ex01.aspx should be named *TbSalary*, as it is a **Textbox** for entering the employee's **salary**.

14. Change the *ID* property of the *TextBox* by (see Figure 3):

- Select the *Textbox* by clicking on it in *Ex01.aspx* web form (NOT the *Textbox* in the *Toolbox* panel).
- Look for (*ID*) in the *Properties* window.
- Change the value of (*ID*) to *TbSalary*.

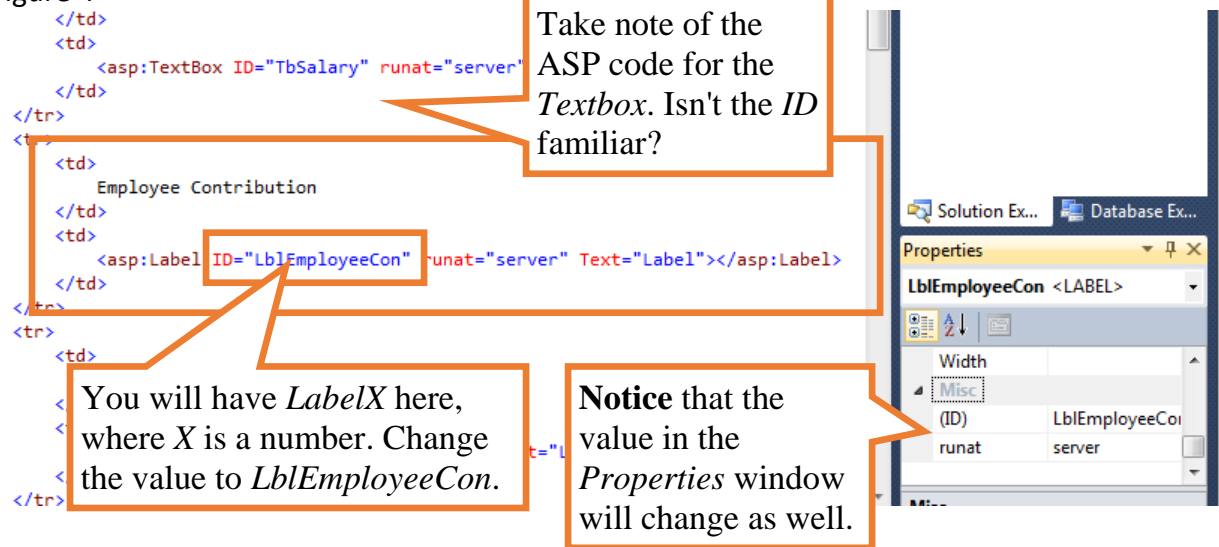
TIP: If you cannot find the *Properties* window, you can find it in the *View* menu. On the menu bar, click *View* → *Properties Window*. Or press F4 on your keyboard. Similarly, you can find and open the other windows and panels like the *Solution Explorer* by exploring the *View* menu.

Figure 3



15. Drag-drop the required Label controls into the table.
16. Now you know two ways of adding controls to a web form. Continue and create the web form as shown in Figure 2.
17. Now, let's learn a different method to change the *ID* of a control:
 - a. Assuming you have created all the required Label controls.
 - b. Switch *EX01.aspx* to *Source* view.
 - c. Look for **Employee Contribution** in the code.
 - d. Then, look for the *Label* control associated with it. (Hint: look for <asp:Label ...>.)
 - e. Change the *ID* property/attribute to *LblEmployeeCon* (the first 3 letters are capital L, small B and small L.)

Figure 4



18. From the above two methods of editing the IDs of the controls, what conclusion can you make about the relationship between the Properties window and the property/attributes of the ASP code in Source view?

Answer:

Important: What we call *attributes* in HTML code, are called *properties* in ASP code. From now on we will use the word **properties** instead of **attributes** for both HTML and ASP. Do remember that outside of this module and in the future, the two words may be used interchangeably by your classmates, lecturers/tutors and co-workers to refer to the same thing.

19. Now, change the IDs and Text properties of all the controls in Ex01.aspx as shown in Table 2. You may use the Properties window or edit the ASP code directly in Source view.

Table 2

Control	ID Property	Text Property	Remarks
Employer Contribution Label	LblEmployerCon		Delete the current value of all the <i>Labels</i> (which is <i>Label</i>) and leave it empty.
Total Contribution Label	LblTotalCon		
Net Salary Label	LblSalary		
Ordinary Account Label	LblOrdAcct		
Special Account Label	LblSpecAcct		
Medisave Account Label	LblIMsAcct		
Button	BtnCalculate	Calculate	

Implementing the Logic

We want the Web Application to calculate the CPF contribution and the distribution to the 3 different accounts accordingly when the user types in his/her salary and clicks the *Calculate*

button. To do that, we need to bind (or link) the click event on the *Calculate* button to an *Event Handler*.

An *Event Handler* is simply a C# method that is automatically called when the respective bound event happens. Most of the ASP.NET controls have more than one possible event, but they usually have a *Default Event Handler* for their frequently triggered event. For example, the *Default Event Handler* of the *Button* control is for the *Click* event. VS will automatically create the *Default Event Handler* for us when we double-click on a control.

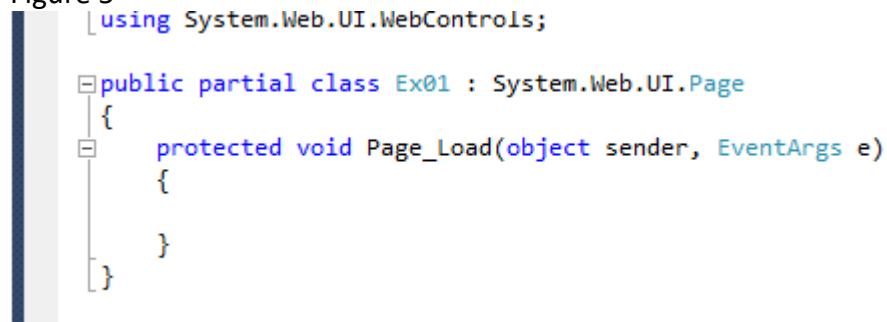
Another *Default Event Handler* that we have encountered in the previous labs is the *Page_Load* event for our Web Forms.

There are other methods of adding event handlers. We will explore those methods in a later lab. For now, we shall just create the event handler for the *Calculate* button and add the necessary C# code to perform the calculations.

Creating and Binding an Event Handler

1. Open *Ex01.aspx.cs* and examine it. Notice there is only **ONE** method in the class, the *Page_Load* event handler (see Figure 5).

Figure 5

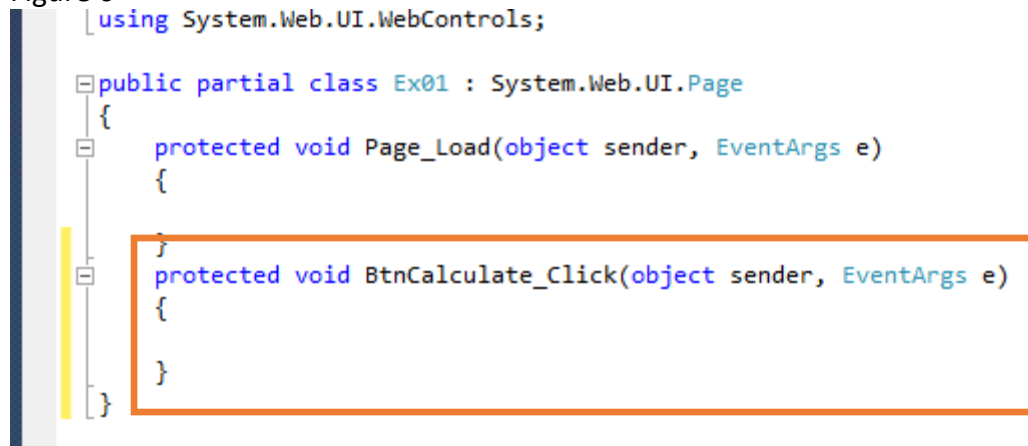


```
using System.Web.UI.WebControls;

public partial class Ex01 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
}
```

2. Switch to the *Design* view of *Ex01.aspx* and **double-click** the **Calculate button**. VS will automatically switch to *Ex01.aspx.cs* file and add the click event handler for you (see Figure 6). Notice also, that its name uses the ID of the *Calculate* button (*BtnCalculate*) and the event (*Click*).

Figure 6



```
using System.Web.UI.WebControls;

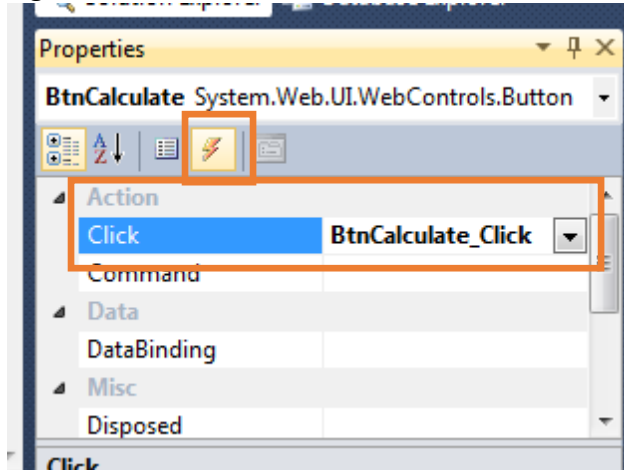
public partial class Ex01 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }

    protected void BtnCalculate_Click(object sender, EventArgs e)
    {
    }
}
```

3. Go back to the *Design* view of *Ex01.aspx*.

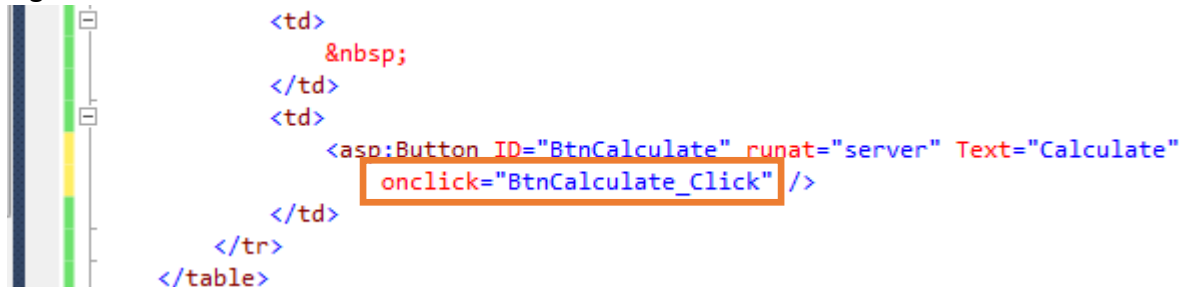
4. Select the *Calculate* button.
5. In the *Properties* window, click on the *Lightning Bolt* icon (see Figure 7).
6. Look for the *Click* action (see Figure 7). Examine its value and notice it is the name of the click event handler that was just created in *Ex01.aspx.cs*.

Figure 7



7. Switch to the *Source* view of *Ex01.aspx* and look for the *Calculate* button's ASP code (see Figure 8). Notice that VS has also added an *onclick* property to it, and the value is the name of the click event handler that was just created in *Ex01.aspx.cs*.

Figure 8



8. Can you see how ASP.NET binds the event handlers to the ASP.NET controls? And how that is represented in the *Properties* window?

Programming the Logic

In this step, we shall write the C# code to calculate the various values.

1. Switch to *Ex01.aspx*.
2. Add the C# code shown in Figure 9 to the *BtnCalculate_Click* event handler. Be sure to copy and read the comments as they explain some of the code.

Figure 9

```

protected void Page_Load(object sender, EventArgs e)
{
}

protected void BtnCalculate_Click(object sender, EventArgs e)
{
    // declare variables
    double grossSalary;
    double employerCon;
    double employeeCon;

    // retrieve user input.
    // use Convert class to convert the string input from the TextBox
    // into a number of type double.
    grossSalary = Convert.ToDouble(TbSalary.Text);

    // calculate values
    // Math.Round(v, n) rounds up v to n decimal digits
    // E.g. Math.Round(1.235, 2) will return a value of 1.24
    employerCon = Math.Round(grossSalary * 0.145, 2);
    employeeCon = Math.Round(grossSalary * 0.2, 2);

    // display values
    LblEmployerCon.Text = employerCon.ToString();
    LblEmployeeCon.Text = employeeCon.ToString();
}

```

Testing the Web Application

1. Test the web application (recall how we did that in the previous labs).
2. Enter 4000 in the *Textbox* and click the *Calculate* button. Your results should match Figure 9 (don't worry if the layout does not match exactly).

Figure 10

CPF Calculator	
Salary	4000
Employee Contribution	800
Employer Contribution	580
Total Contribution	
Net Salary	
Ordinary Account	
Special Account	
Medisave Account	
<input type="button" value="Calculate"/>	

Complete the Logic and Test

1. Write the rest of the code on your own to complete all the calculations assuming that the user is 35 years old or younger. We have not yet learnt how to validate the

user's input, so do not worry about that yet, and expect the Web Application to display an error if you enter non-numeric values.

2. As you write the code, test your web application to check your work.
3. Once you have completed, do a final test to make sure everything is working correctly. The final results for 4000 should be as shown in Figure 11.
4. Show your completed work to your tutor.

Figure 11

The image shows a web application titled "CPF Calculator". It features a list of labels on the left and their corresponding values on the right. The "Salary" field is an input box containing the number "4000". Below the list, there is a "Calculate" button. The results are as follows:

Salary	4000
Employee Contribution	800
Employer Contribution	580
Total Contribution	1380
Net Salary	3200
Ordinary Account	920.05
Special Account	199.96
Medisave Account	259.99

Calculate

Conclusion

In this practical, you have learnt:

- How to create an Empty Website.
- How to add a Master Page.
- How to add controls to a Web Form.
- How to change the properties of controls in:
 - a. *Properties* window.
 - b. ASP code in *Source* view.
- Work with events and event handlers.
- How ASP.NET binds event handlers to controls.
- How the Properties window displays the event handlers bound to controls.

~ End ~