

located at the top of this page. For instructions on how to use the Dropbox, read these [step-by-step instructions](#).

(See the Syllabus section "Due Dates for Assignments & Exams" for due dates.)

In this iLab, you will create an **Annual Salary Calculator ASP.NET Web application** using Visual Studio.NET.

For the iLabs, you will use the Microsoft Visual Studio software application. You have two options for using this application:

- You may use a copy of Visual Studio that is installed on your local PC. If you do not already have this software, there is a link in the Syllabus (in the Textbook and Required Materials section) that you can follow to obtain a copy at low or no cost through the DeVry Student Software Fulfillment Center.

or

- You may run Visual Studio over the Web from the DeVry iLab (Citrix) server. Instructions for using the iLab (Citrix) server are on the iLab page under Course Home.

Throughout this course, you will be creating a Web application for a fictitious company called ACIT (Academy of Computing and Information Technology). To get familiar with the development environment, follow the Lab Steps to create a simple **Annual Salary Calculator** ASP.NET Web application. You will be adding to this application each week.

Overview of Fictitious Company



The Academy of Computing and Information Technology is a mid-size indie (independent) film studio that is in need of a payroll system. We have outgrown our current system, a simple spreadsheet, to the extent that it takes three people over one week to pay everyone on a timely basis.

Overview of Our Company

We have over 2,000 full-time and part-time employees. We

produce comedy, fiction, and science fiction films with budgets of \$250K–\$20 million. We have produced over 50 films since we first began 20 years ago.

We are very profitable and have strong links to many of the industry's most powerful people.

Current Payroll System

Our current system consists of mostly manual processing using an MS Excel spreadsheet to control who gets paid.

The system consists of the three payroll staff members reviewing each of the full-time staff members' wages, calculating how many hours they worked based on their hourly rate, and paying them by issuing a check.

The check needs to be entered in another worksheet for each of the people who were paid so that we can tell when it went out, how much it was for, and if it was cashed or not. If a [Date_Cashed] is entered, we deduct that amount from our working payroll capital account to track how much money we have for payroll.

This process is then repeated for all part-time staff and independent contractors.

Our Needs

We need a more automated way to pay our staff. We need to use the same logical flow as a basis, yet improve on it by having a way to

1. easily calculate the projected annual salary based on rate and number of hours;
2. search, enter, update, and delete staff and independent employee information in one place;
3. search, enter, update, and delete payroll automation information for the employee to set up recurring payments or one-time payments;
4. produce reports to show the following: (a) summary of who got paid per week, (b) summary of all payments per week, (c) details of any employee to-date (allow entry of a specific employee ID);
5. allow transactions to be rolled back in case we pay someone too much;

6. make use of transaction processing in case the system unexpectedly shuts down;
7. ensure the system is secure (logins required);
8. allow only authorized payroll personnel to control all aspects of the system;
9. allow only authorized payroll personnel to view transactions and user activity;
10. allow only authorized payroll personnel to e-mail reports to users using the e-mail on file; and
11. incorporate error handling into all processes in the system and e-mail any errors to the technical support people.

Required Software

Microsoft Visual Studio.Net

Access the software at <https://lab.devry.edu>.

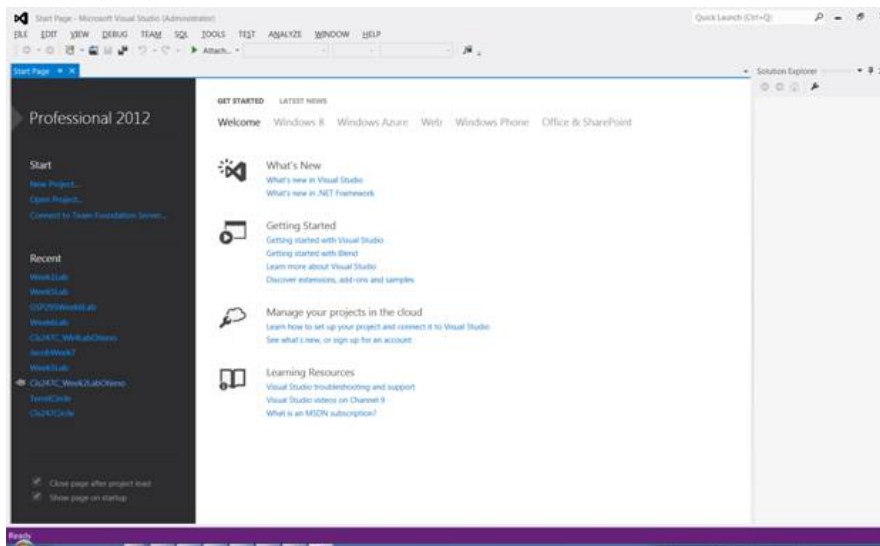
Steps: 1, 2, and 3

STEPS 1–3: Create Website and Home Page

Please watch this video for a similar example to the one we are doing. It introduces event handlers, getting data from textboxes, performing basic calculations, and formatting the output to be displayed:

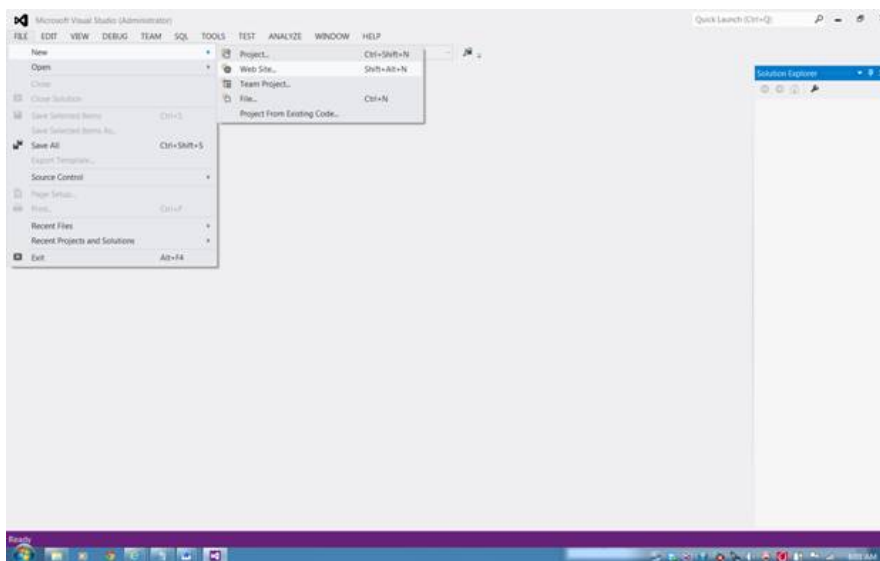
In this iLab, we will learn how to create a simple ASP.NET Web application using Microsoft Visual Studio.NET. The application will display the text "Hello, World" on the home page.

1. Open Microsoft Visual Studio.

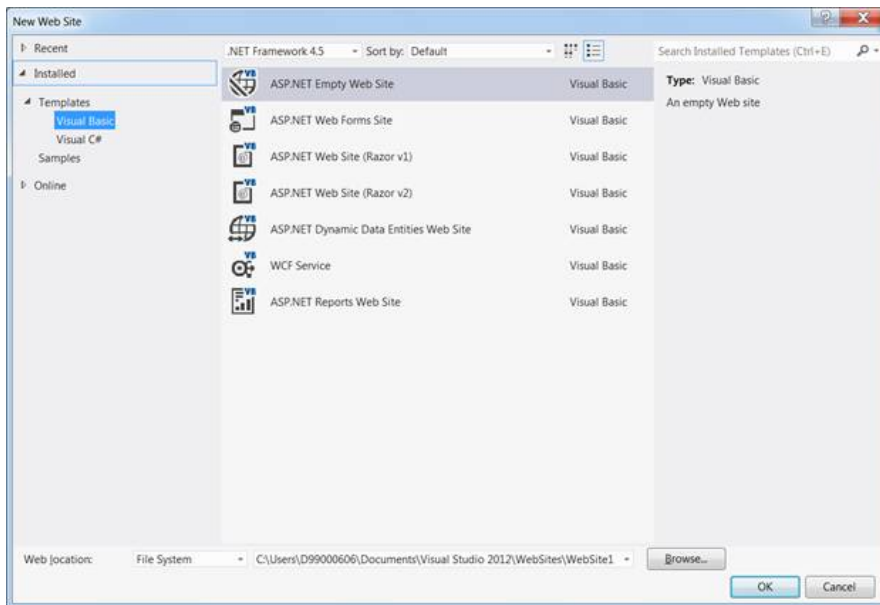


2. Create a new ASP.NET website called "PayrollSystem."

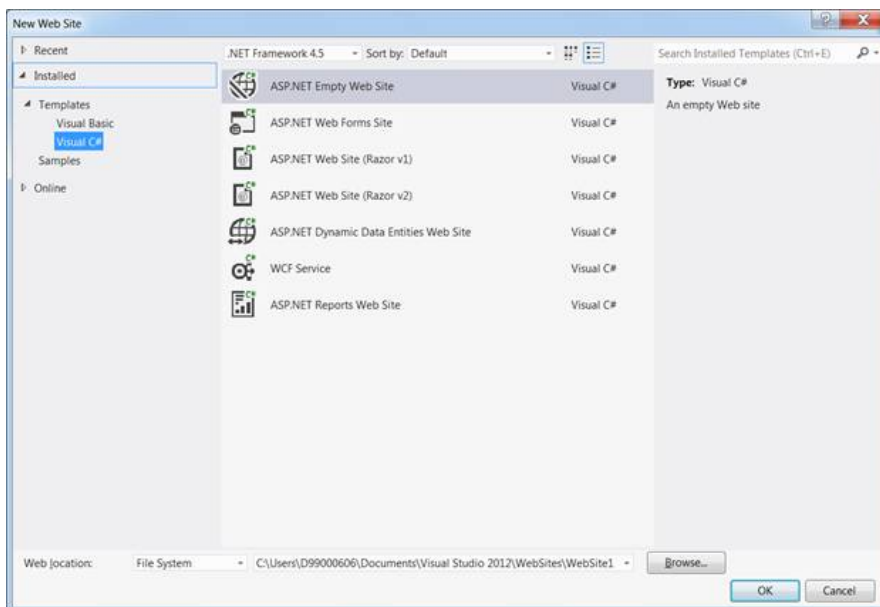
To do this, select **File -> New -> Web Site**



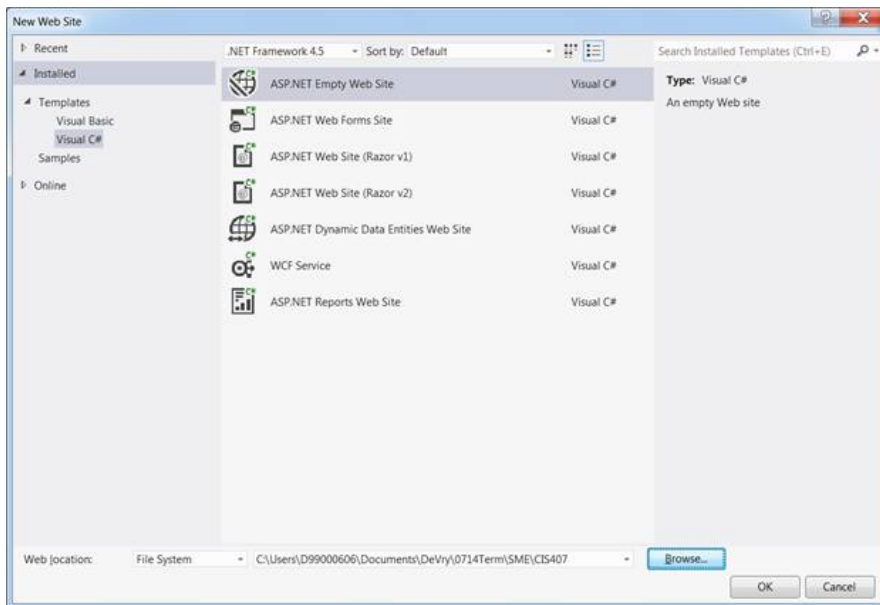
You will get the following screen which shows Visual Basic.



Select Visual C# template on the left of your screen if it is not already selected. Notice that your screen will now show Visual C# instead of Visual Basic.

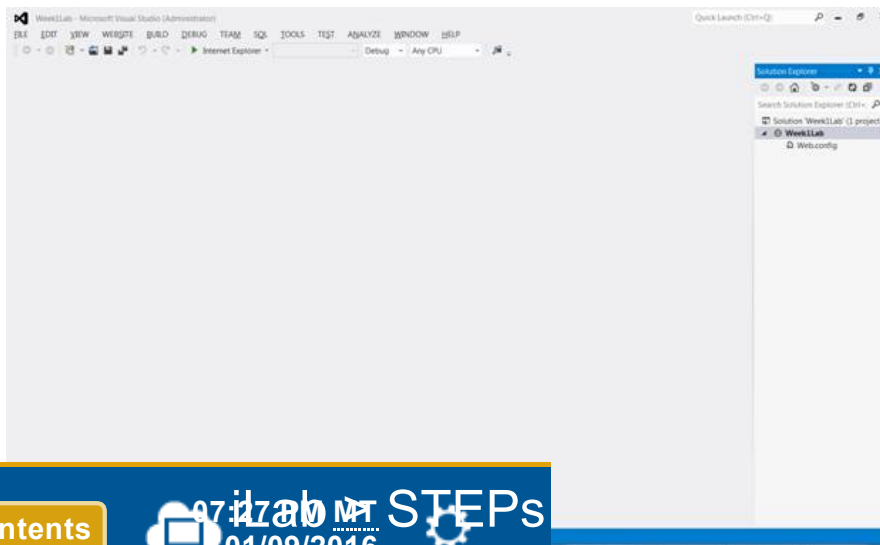


Click Browse at the bottom of the screen to change Web Location, if necessary to get the screen below.



Notice that my **Web Location** is now different.

Select ASP.NET Empty project and click OK to get the screen below.



Contents

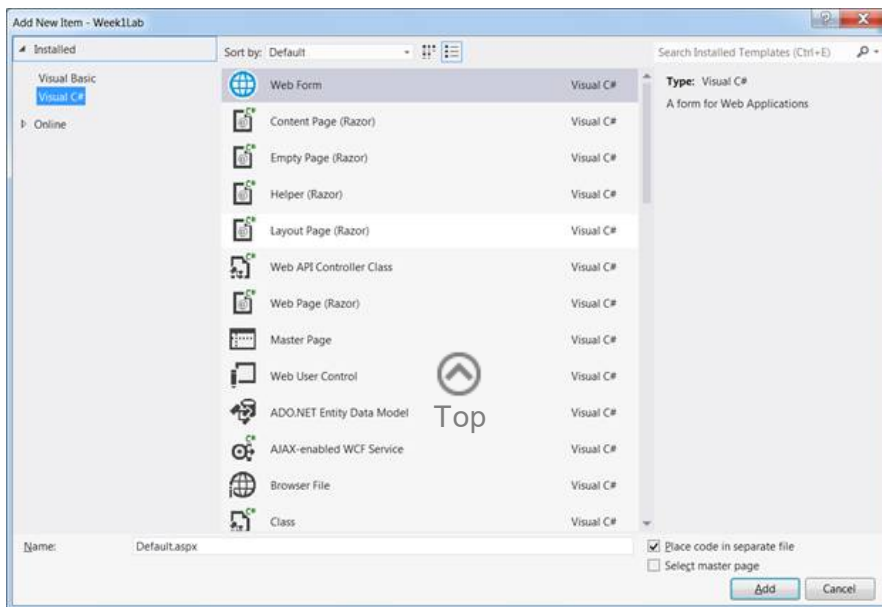
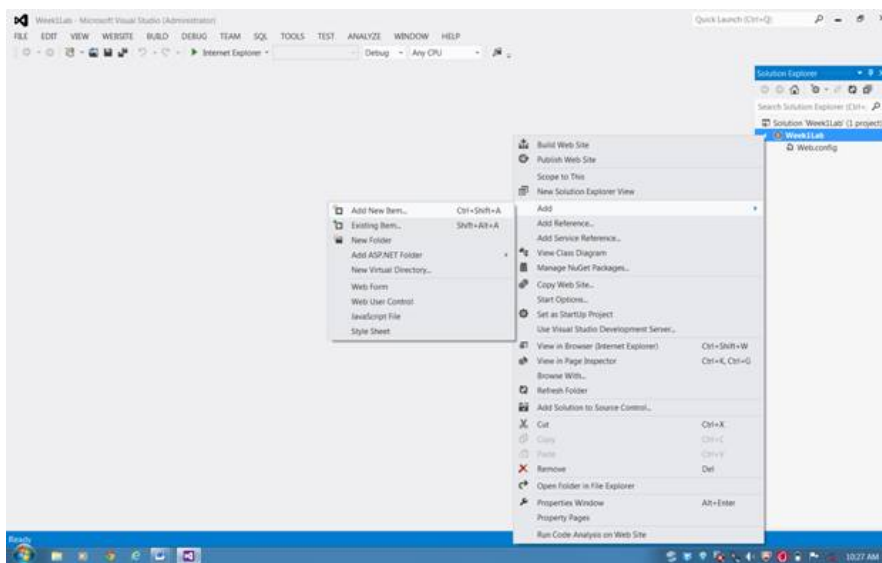


07:47 AM
01/09/2016

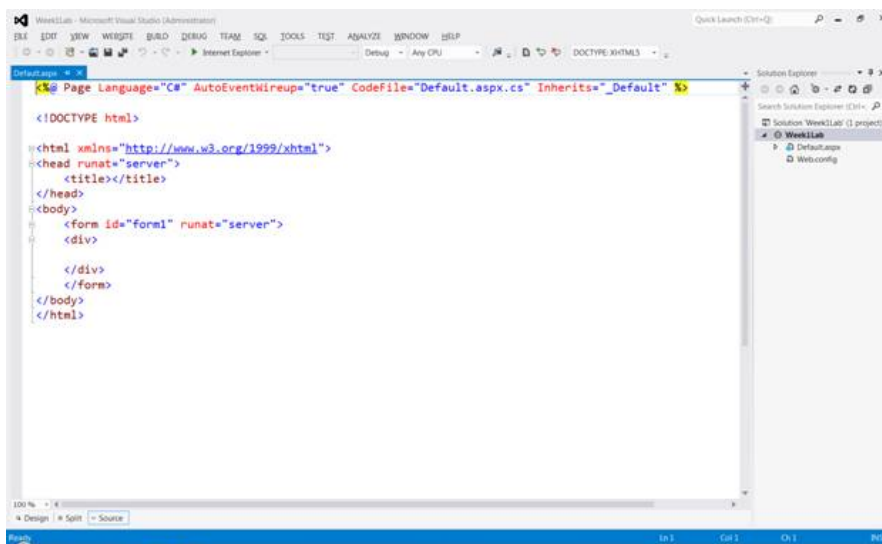


STEPS

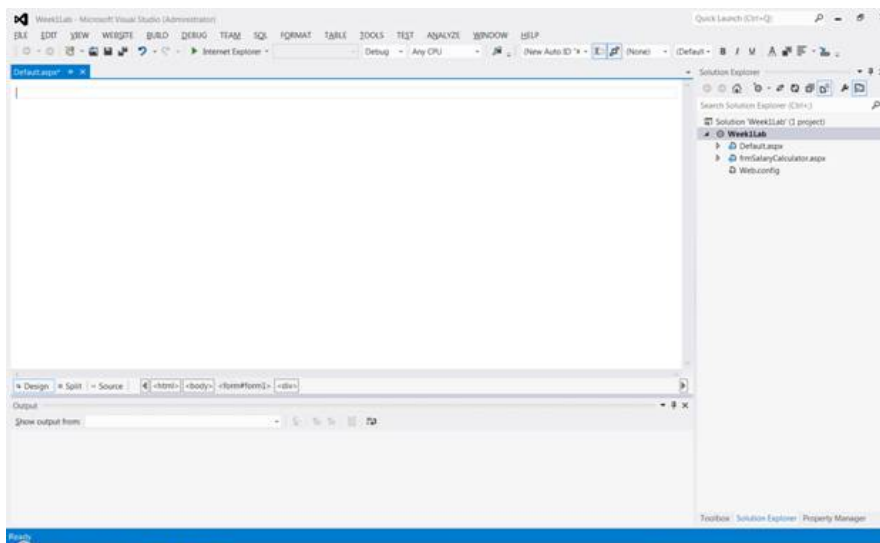
Right-click on the project name from Solution Explorer, then select **Add->Add New Item** to get the following screen.



Select **Web Form**, accept Default.aspx file, and click Add to get:

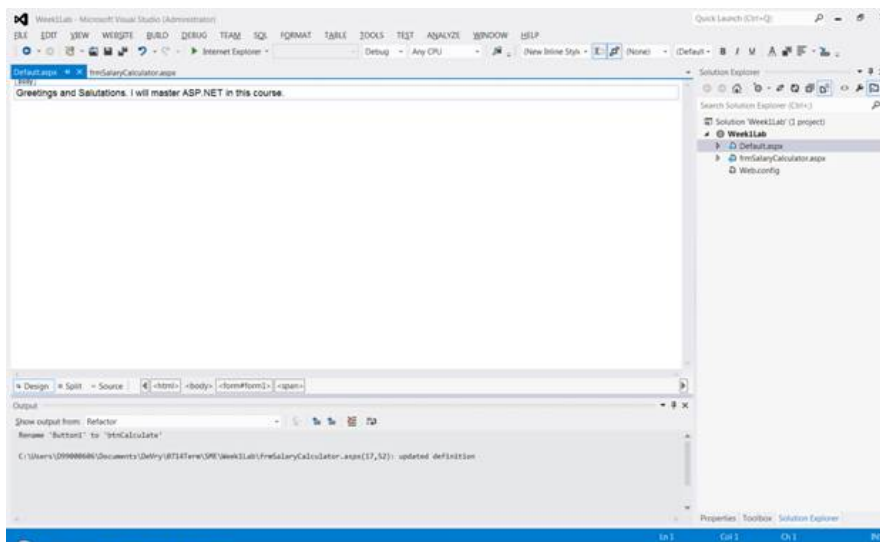


Click **Design** at the bottom left-hand of the window to show the following:



Edit the Default.aspx file (the home page for your site) to add the message "Greetings and Salutations. I will master ASP.NET in this course."

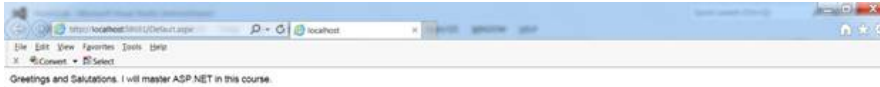
To do this, if necessary, click the Design button below the editing window to switch to Design view, then click in the editing window and type " Greetings and Salutations. I will master ASP.NET in this course." (without the quotes) to get the following screen.



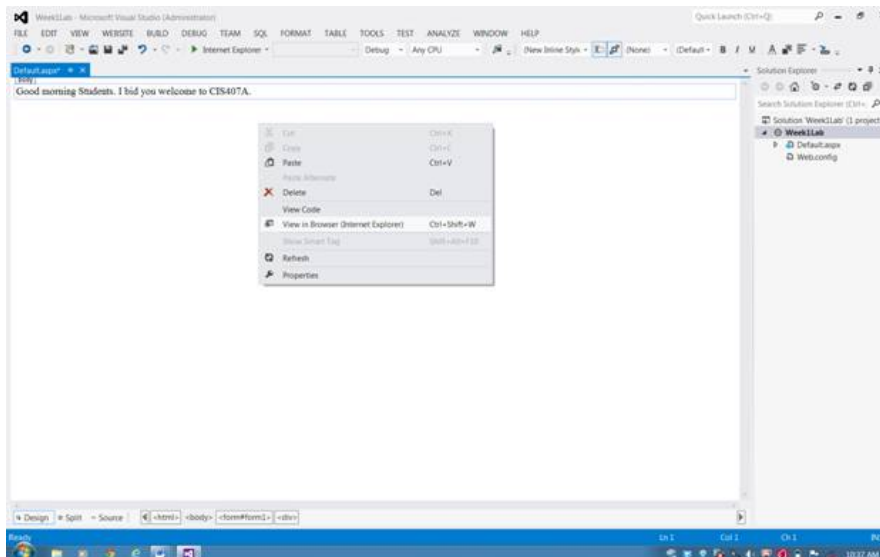
Click the Save button on the toolbar to save the changes to Default.aspx.

STEPS 4–5: Start Debugging; NOTE: Citrix users have different steps!

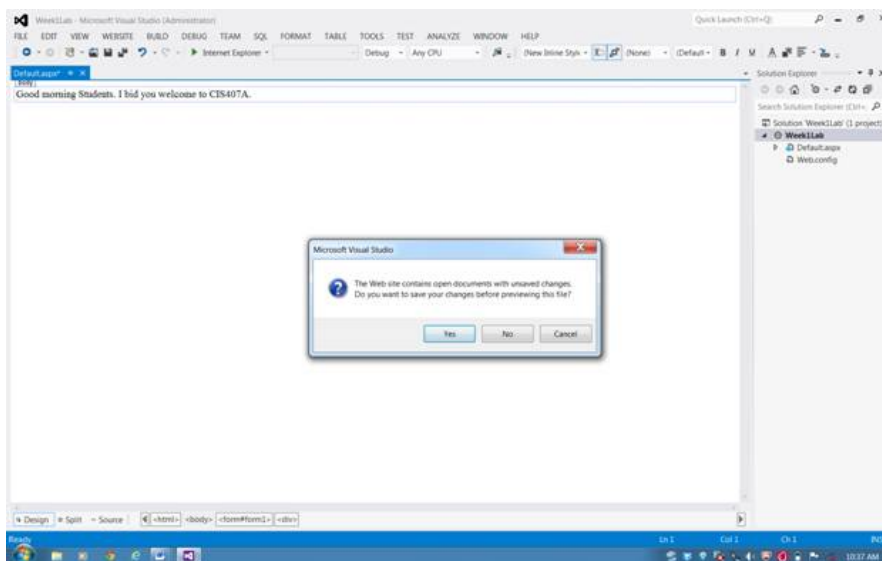
3. To ensure that everything is working properly, click the Start Debugging button on the toolbar, or press the F5 key on the keyboard, or pull down the Debug menu and select Start Debugging.



Save and run by **Right-Click-> Run** from Browser.

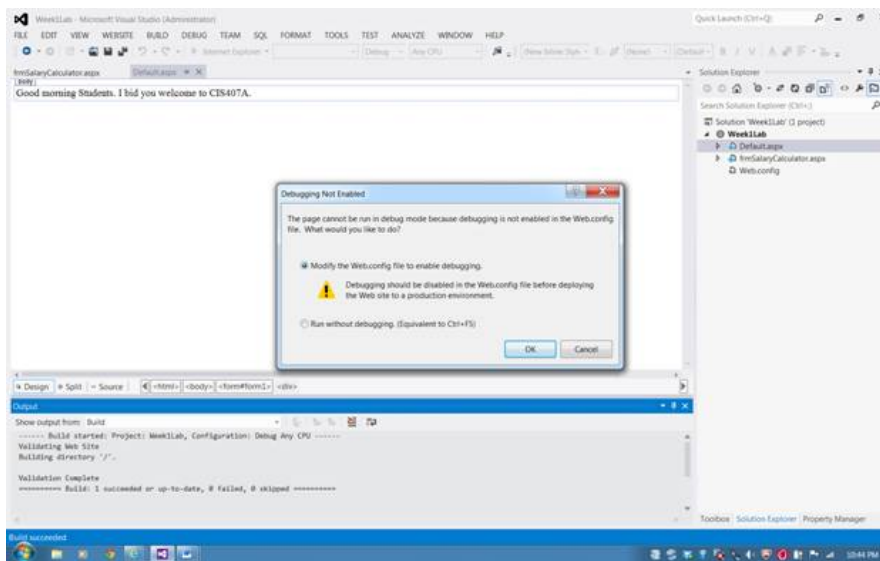
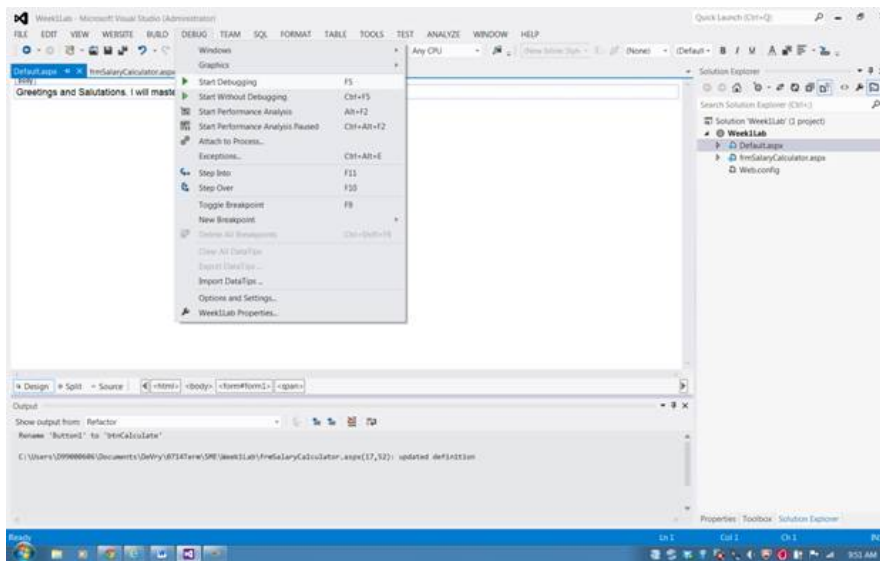


Press Yes.



Click the Start Debugging button on the toolbar, or press the F5 key on the keyboard, or pull down the Debug menu and select "Start Debugging."





Notice that this time the project build and website validation started.

If the Debugging Not Enabled dialog box appears, select the option to add or modify the Web.config file to enable debugging and click OK. You should only have to do this the first time you debug the site.

You will get a clean run just as you did previously. Your output screen looks like the screen below.

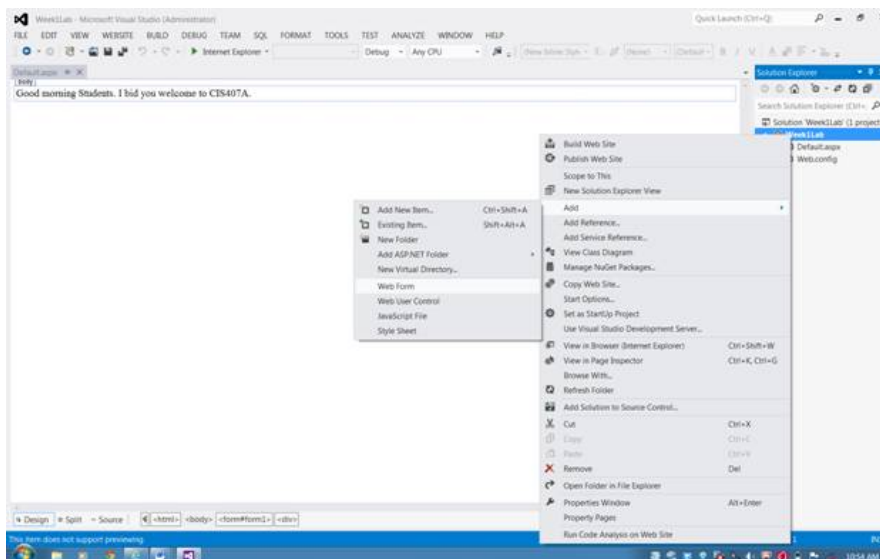


NOTE: To execute the application, you have these options:

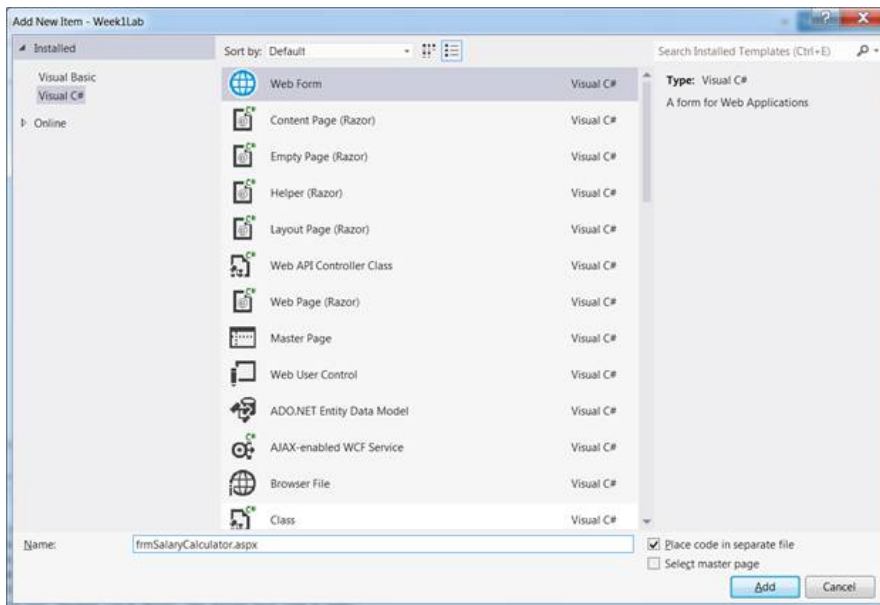
- If you are using Citrix, press CTRL + F5 to Start Without Debugging. You will not be deducted points for this part.
- If you are using a standalone version, press F5 to Start with Debugging, or you can press CTRL + F5 to Start Without Debugging.

Create the Salary Calculator Form

1. Right click on the Project name. Choose Add, then Select Web Form to get the screen below.



And you get the Add New Item screen, shown below.

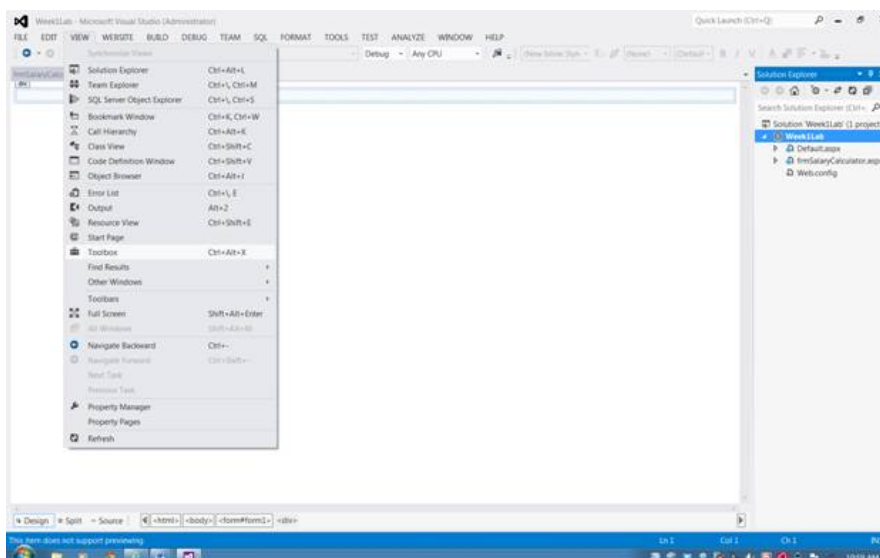


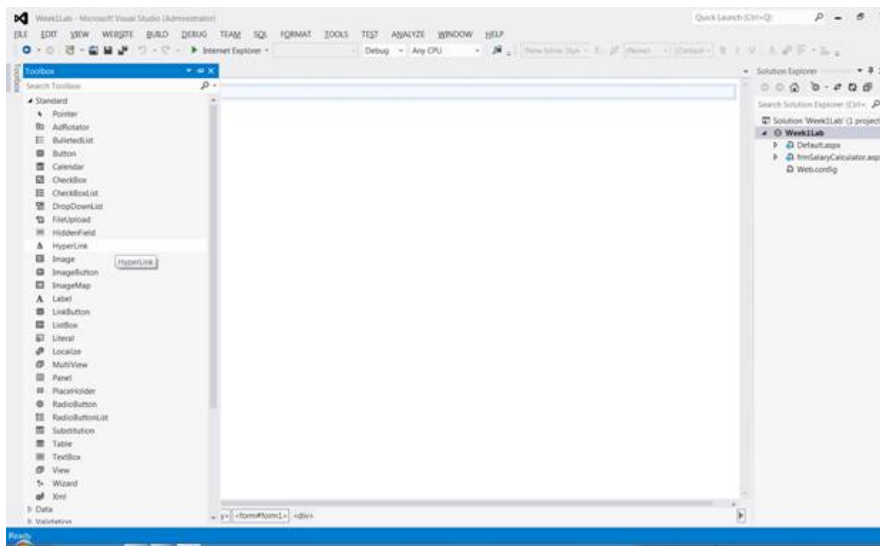
2. Select the name of the form you will add **frmSalaryCalculator.aspx**. Make sure "Place code in separate file" is checked and "Select master page" is unchecked.

You will create a Web-based salary calculator on this new page.

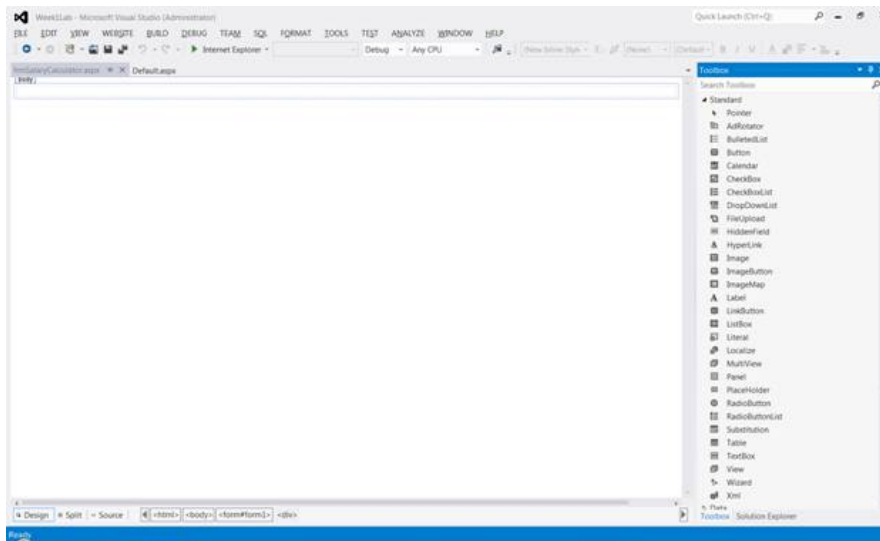
Click the **Design** view.

Add the Tools Window using **View-> Toolbox**.





3. You can choose to adjust the ToolBox to tab with Solution Explorer to look like the following screen.



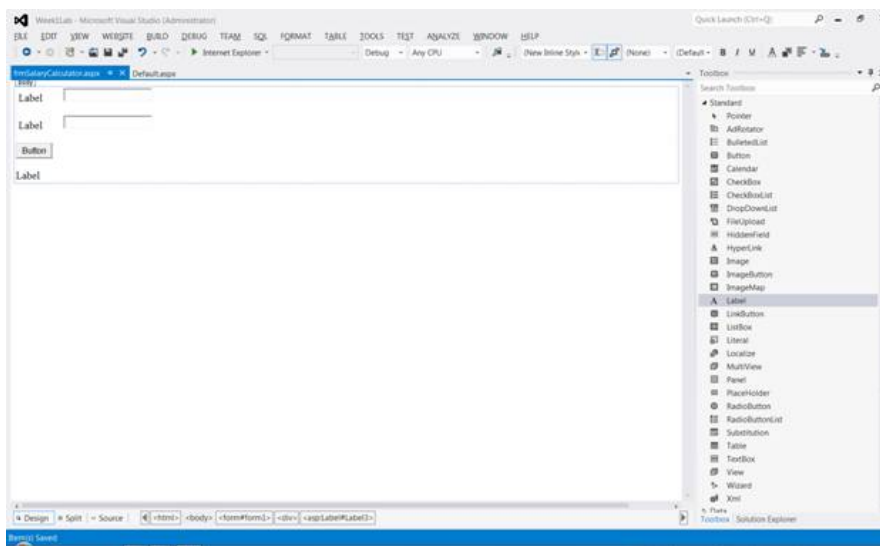
You will create a Web-based salary calculator on this new page.

To do this, open the **aspx** page in **Design** view and, from the Toolbox, drag a **label** into the form, click after the label and add about 5 spaces, then drag a **textbox** control after the label.

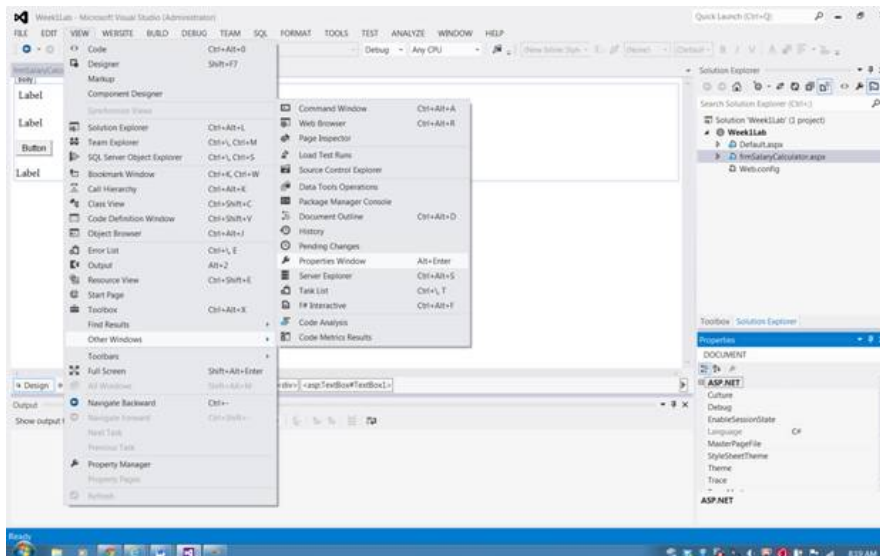
Press Enter after the textbox to put the cursor on the next line.

Add another **label** and **textbox** below the first set and press Enter.

Then add a **button** control. Finally, on the last line, add another **label**. Your form should look like the screen displayed below.



4. If necessary, add the Property Window as shown in the screen below, using View->Properties Window.

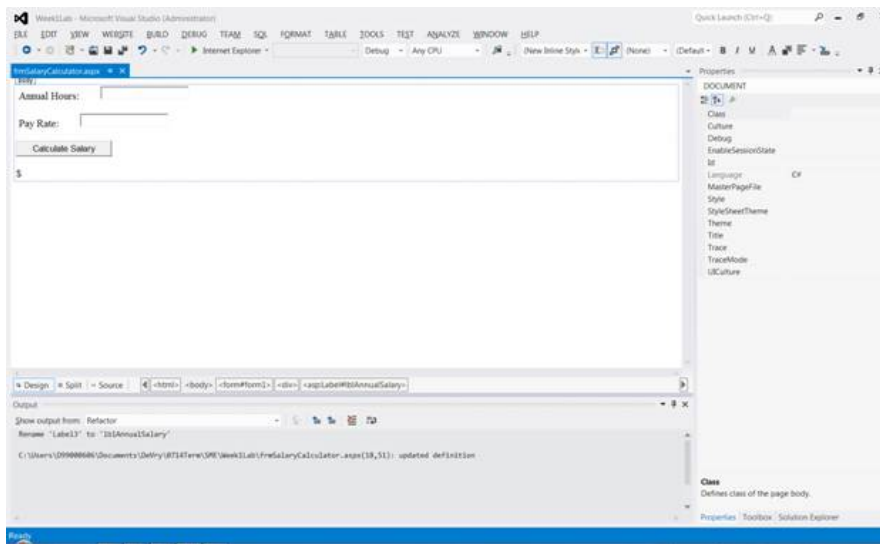


5. Now we will modify the page to add proper labels and give the textboxes names.

- Change the text displayed in each label so that the first label displays **Annual Hours**; the second label should display Rate; and the third label should display \$.
- To change the text displayed, click on the label control. This causes the property window to show all properties of the label, then change the Text property of the control in the Properties window.
- Set the ID property of the top textbox to **txtAnnualHours**.
- Set the ID property of the second textbox to **txtPayRate**. Set the ID of the bottom label (the one we set the text property

to "\$") to **lblAnnualSalary**. (Note: We set these IDs as we will be accessing the control values from the C# code. You can set the button ID and the other two labels' ID properties as well, but we won't be accessing them from our code.)

- Change the button text to display **Calculate Salary**. (Hint: To change the text displayed as the button label, change the Text property of the button.) The ID of the button should be **btnCalculateSalary**. Your form should now look like the screen displayed below.



This code will be called each time the user presses the button. It is important to remember that code in the code behind page executes on the **server**, not on the user's browser. This means that when the button is pressed, **the page is submitted back to the Web server and is processed by the ASP.Net application server on the Web server**. It is this code (between the { and } in this method) that will execute on the server. Once it is done executing, the page will be sent back to the browser. Any changes we make to the page or controls on the page will be shown to the user in the updated page.

- In this method, add code that will get the text in the txtAnnualHours text box, convert it to a Double, and store it in a double variable.
- Add code that will get the text from the txtPayRate text box, convert it to a Double, and store it in another variable.
- Create a third variable of type Double and set its value to the annual hours variable value multiplied by the rate double variable value.
- Take this resulting value and convert it to a string (text), and update the lblAnnualSalary Text property with this new string.

Let's look at a similar example. **After reviewing this example, write the code needed to calculate the annual salary.**

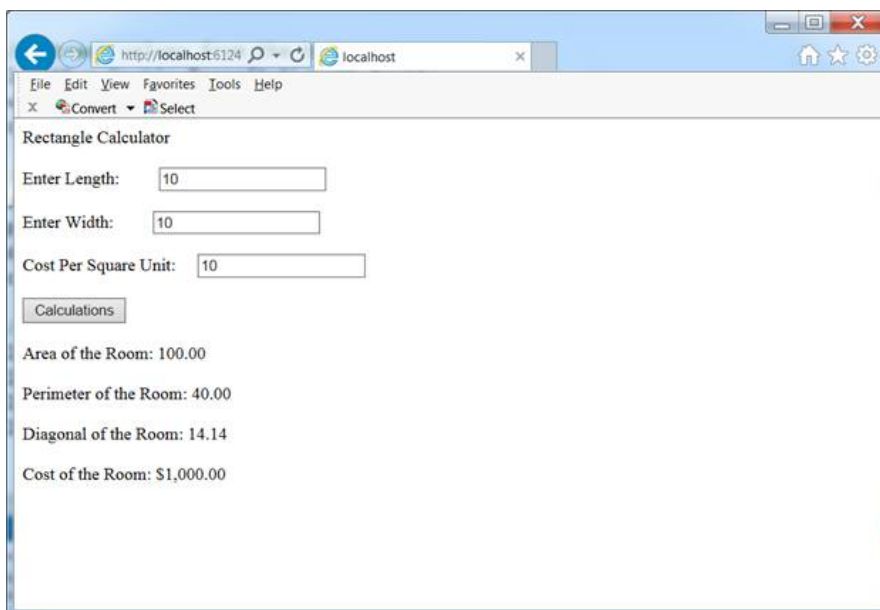
CostOfRoom Calculator is the alternate example, which demonstrates the skills you need to complete this assignment. See video at the top of this lab document and screenshots below.

What follows is an example of code-behind the Calculate button for the CostOfRoom Calculator:

Code-behind the calculate button

A control's property can be accessed by simply using the control ID followed by a . followed by the name of the property. For example, the value stored in the Text property of the txtAnnualHours control can be accessed by using this: txtAnnualHours.Text. Text properties on controls are of type string.

The output of the CostOfRoom calculator is shown in the screen below with Length, Width, Cost Per Square Unit labels and input boxes, and the Calculations button.



Use small values for length and width.
Use large values for length and width and see the formatting of the output.