

## Configuring Visual Studio – Solutions and Projects

Visual Studio will be an important tool in the development of our project. We have however only scratched the surface of its capabilities.

During the first year we have used it to assist us in creating basic interfaces and attaching code and database functionality.

If we are to work in a more real world way there are certain requirements we need to meet.

We need to design our system in such a way that code may be re-used across any and all systems we create now and in the future.

To achieve this we will create a class library. The library will allow us to share important code.

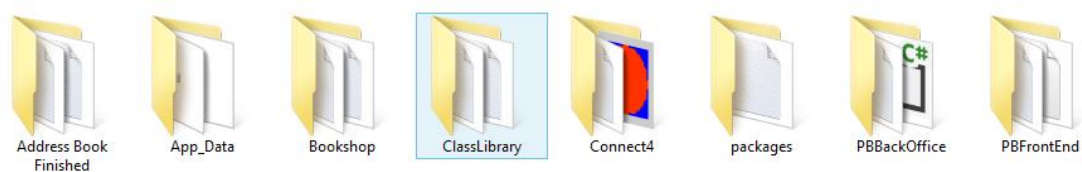
We also need to create code that is easy to share with other people.

In creating a system in the real world it is highly unlikely that we will be the only person needing to access it. We need to organise the code such that multiple programmers have access and there is a system for tracking who has changed what, i.e. version control. We also need to think about what happens when two programmers want to change the same section of code.

Another important aspect we need to think about is how we plan to test the system. Manual testing is a tedious time consuming exercise that is not terribly rigorous. It would make a great deal of sense if there is some way of automating our testing such that the system handles this for us.

### ***The Folder Structure***

For this module we are going to create a folder structure along the following lines...



App\_Data will contain our database file that may be shared across all applications we create.

ClassLibrary will contain any code that we want to share across multiple applications.

The other applications (Address Book Finished, Bookshop, Connect4, PBBackOffice, PBFrontEnd) are the applications we will look at which may or may not share the resources provided in the database / class library.

We will look at how to configure Visual Studio to allow us to do this.

## ***Solutions and Projects***

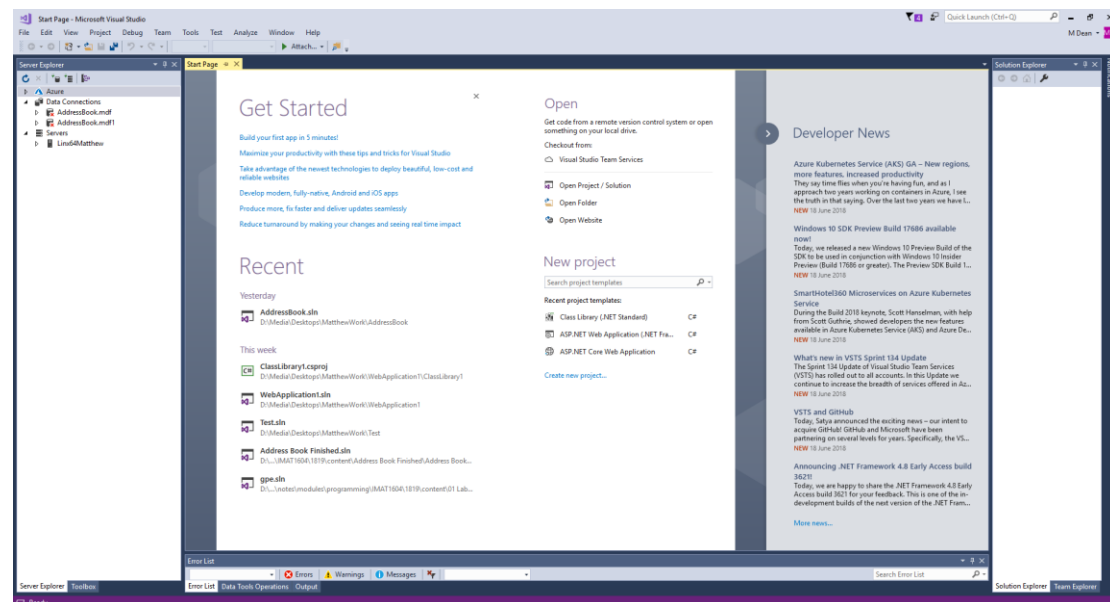
Visual Studio allows us to create both solutions and projects.

A project is typically an application of some sort.

E.g a desktop application, a web site or a class library (to name a few)

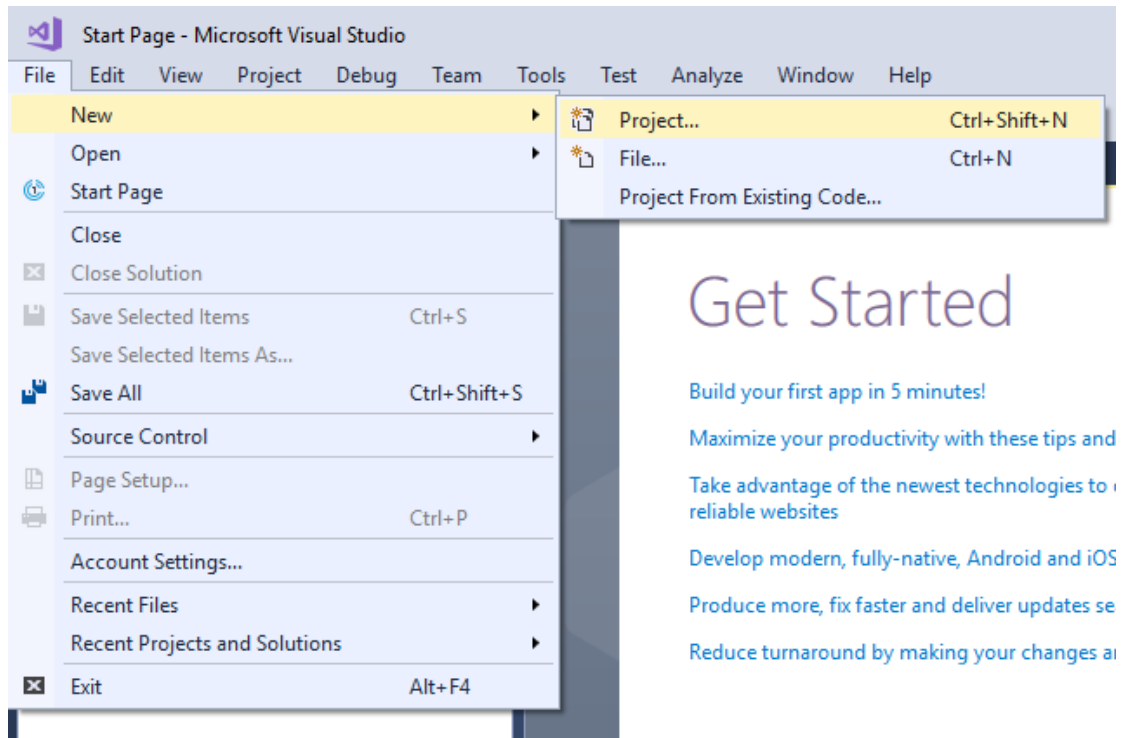
A solution is container that allows us to work on multiple projects at the same time.

Start Visual Studio and you should see the main screen for the program.



## Creating the Solution

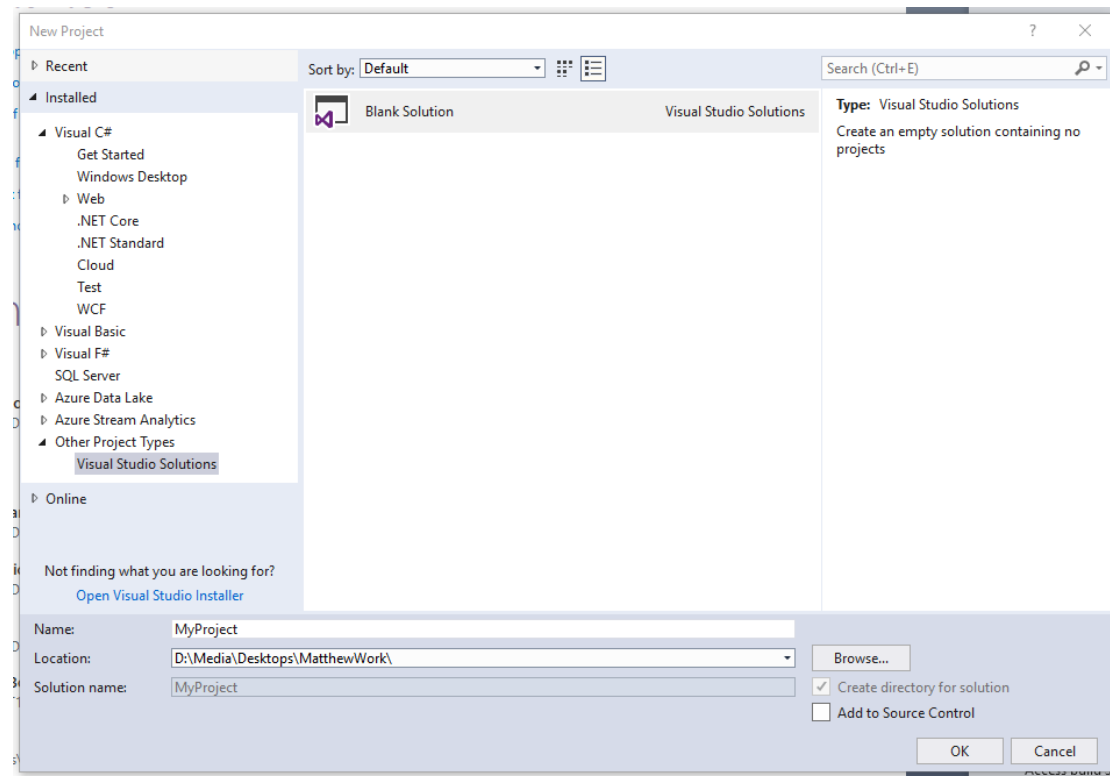
To create the solution that will allow us to manage the different projects we plan to create do the following.



Select File – New – Project

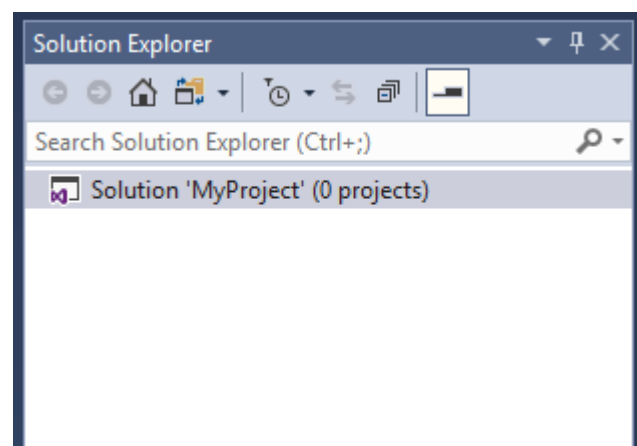
(A solution is technically a type of project in Visual Studio!)

Under “Other Project Types” select “Visual Studio Solutions”.



Give the solution a suitable name e.g. “MyProject” and decide on a suitable path for the solution.

Press OK and you should see something like this in the solution explorer...



Notice that we didn’t need to specify a language for the solution.

The nice thing about creating solutions in Visual Studio is that the projects we add to it may be in any language supported by Visual Studio.

This means we could create a class library in C# and share that code with a web application created in VB.NET.

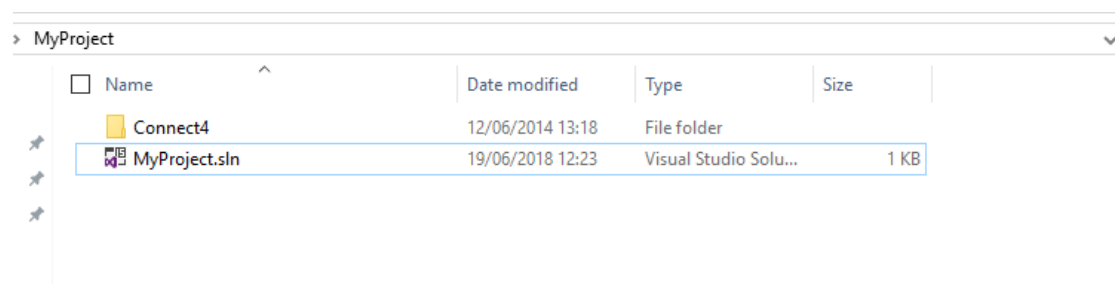
There are many different combinations allowed by this configuration.

## ***Adding an Existing Project***

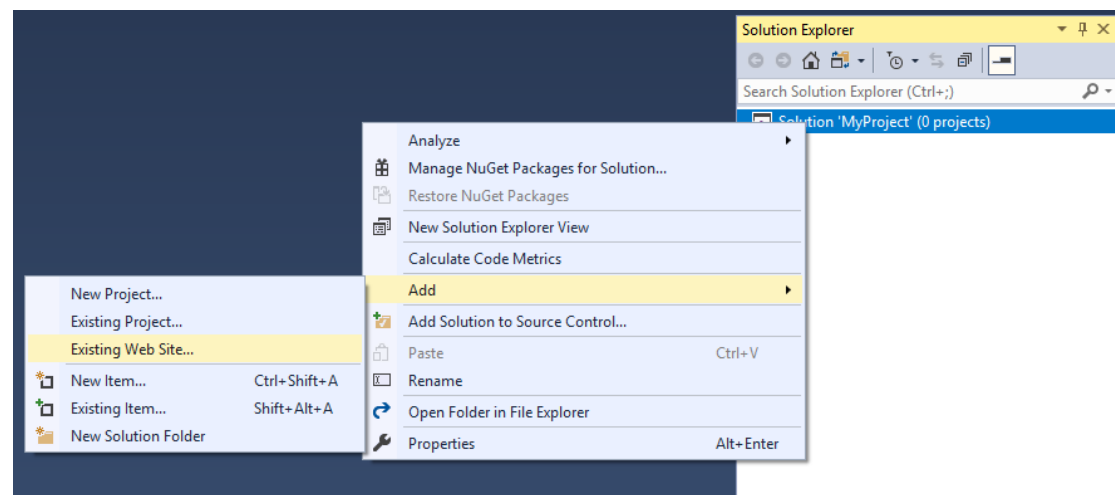
In order to set up the solution we shall start by adding an existing project to the solution (in this example a web site).

From the module web site download the zip file for Connect 4. Extract the contents and copy the folder structure into the folder you created for your solution.

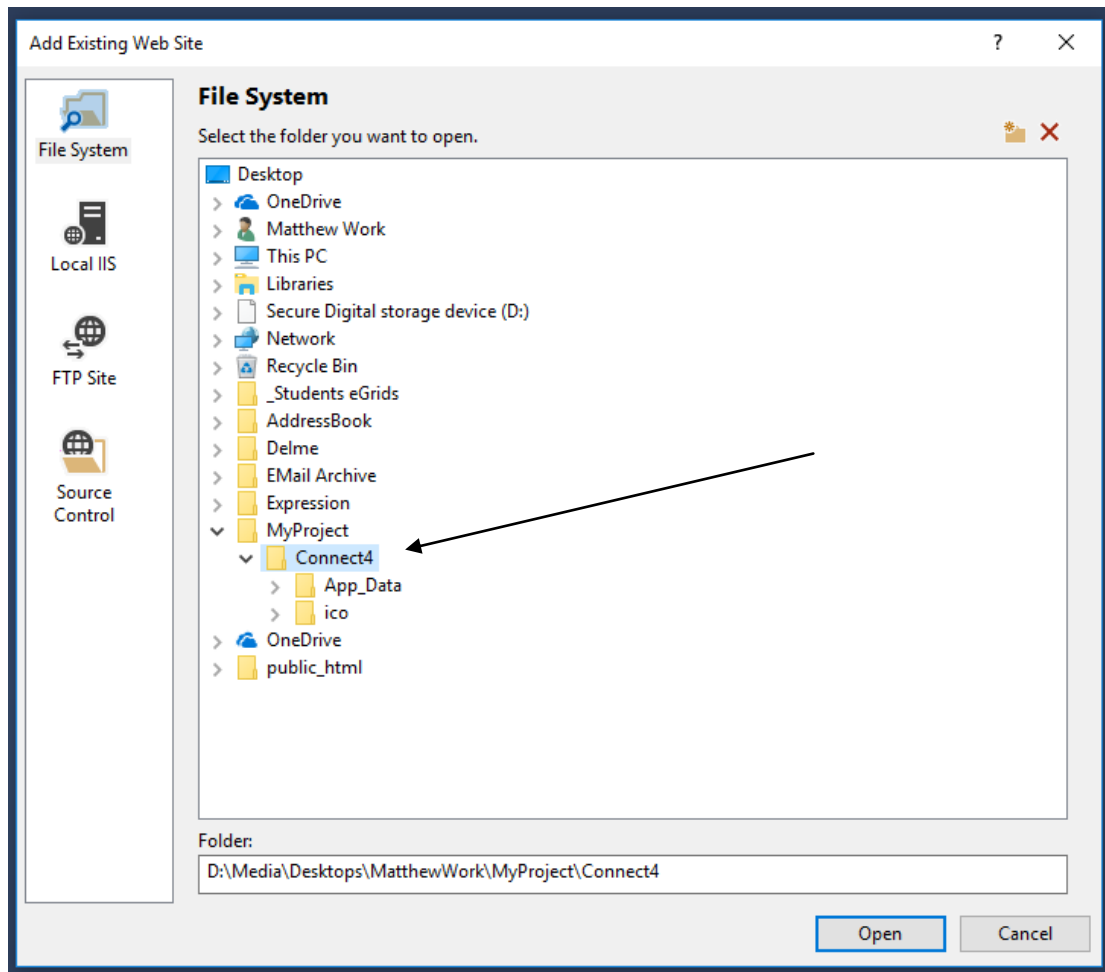
It should look something like this...



In Visual Studio right click on the solution and select, add – existing web site...

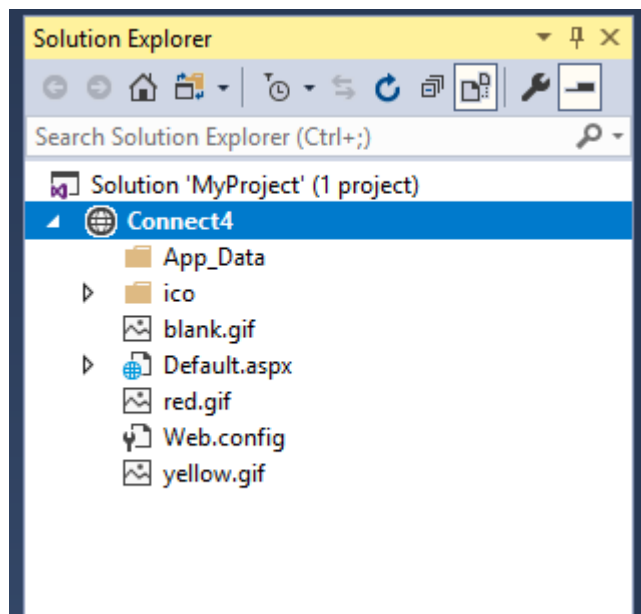


Navigate to the location for the Connect 4 site...



And then select open.

The solution explorer should look something like this...



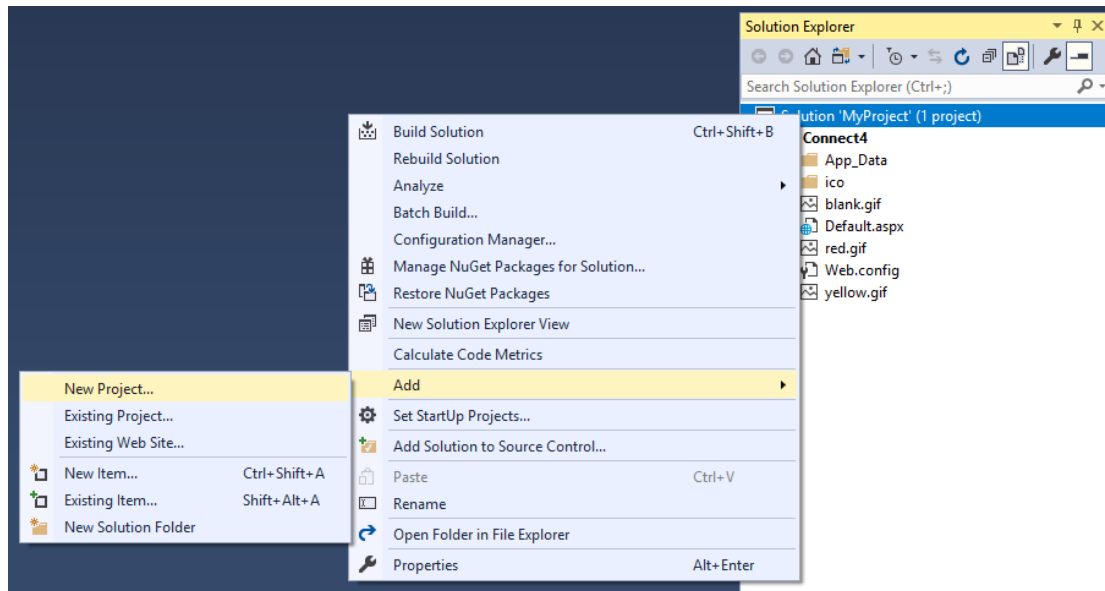
Run the program to see what happens. (You should be able to have a quick game of connect 4)

Ok nothing new so far. This doesn't look too different to what we did in the first year.

## ***Creating the Project Bank Back Office Application***

We will now add a second project to the solution to see how this all works.

Right click on the solution in Visual Studio and select Add – New Project...

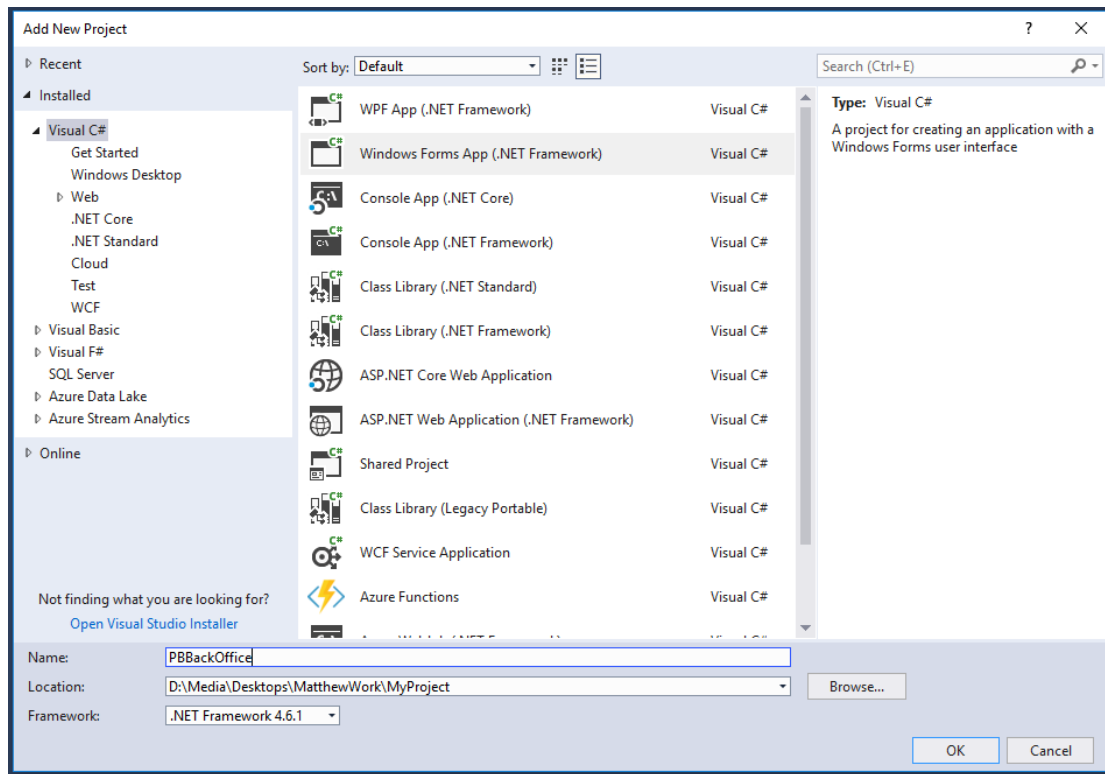


We will create the back office application for this program.

The back office is typically the part of a system which runs on a PC, i.e. not a web application.

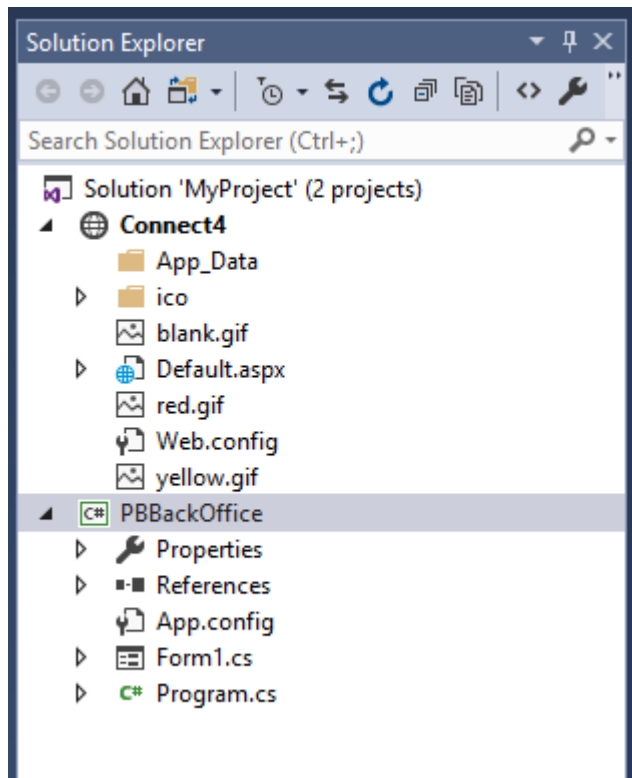


Select the following options for the new project...

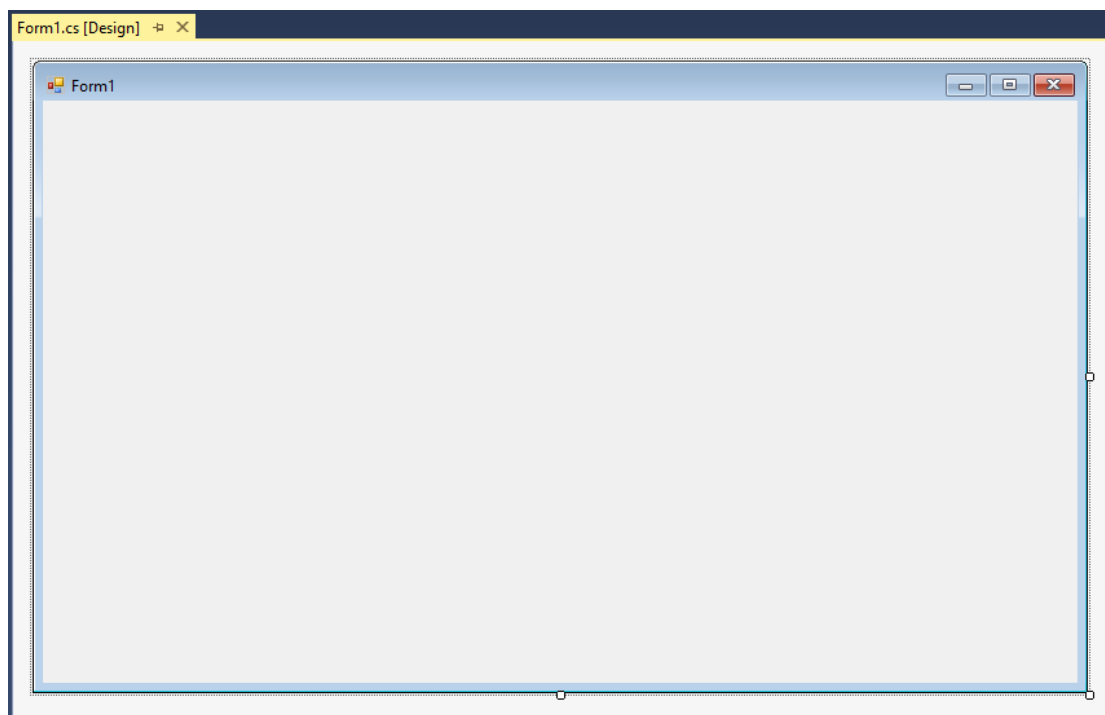


Make sure that the language is C#, the name of the project is PBBackOffice and the location points to the folder for your solution.

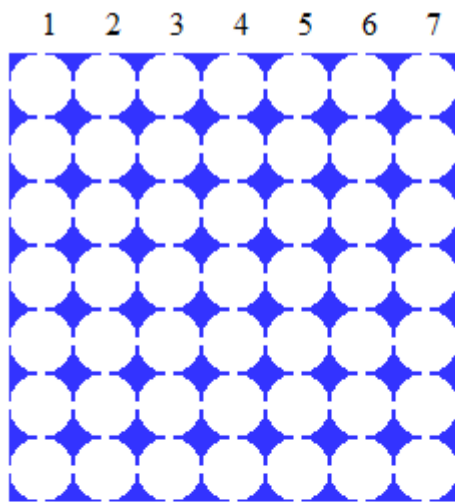
Press OK and you should see the following in the solution explorer...



Notice also that Visual Studio has created a default form.



Run the solution and see what happens. What should happen is that you end up with Connect 4 again...



Enter a number 1 - 7

Go

Enter a number 1 - 7

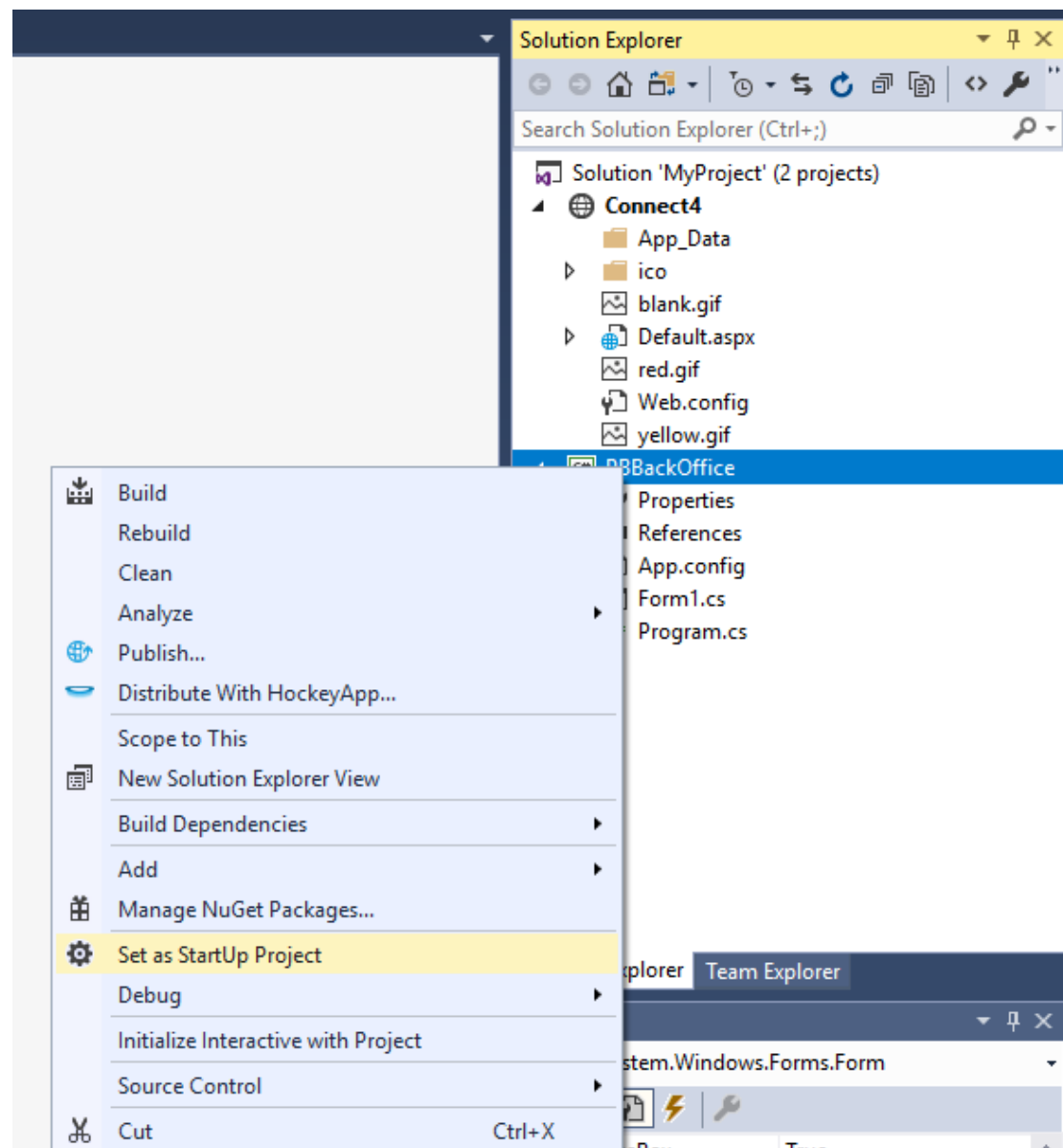
Go



Clear The Board

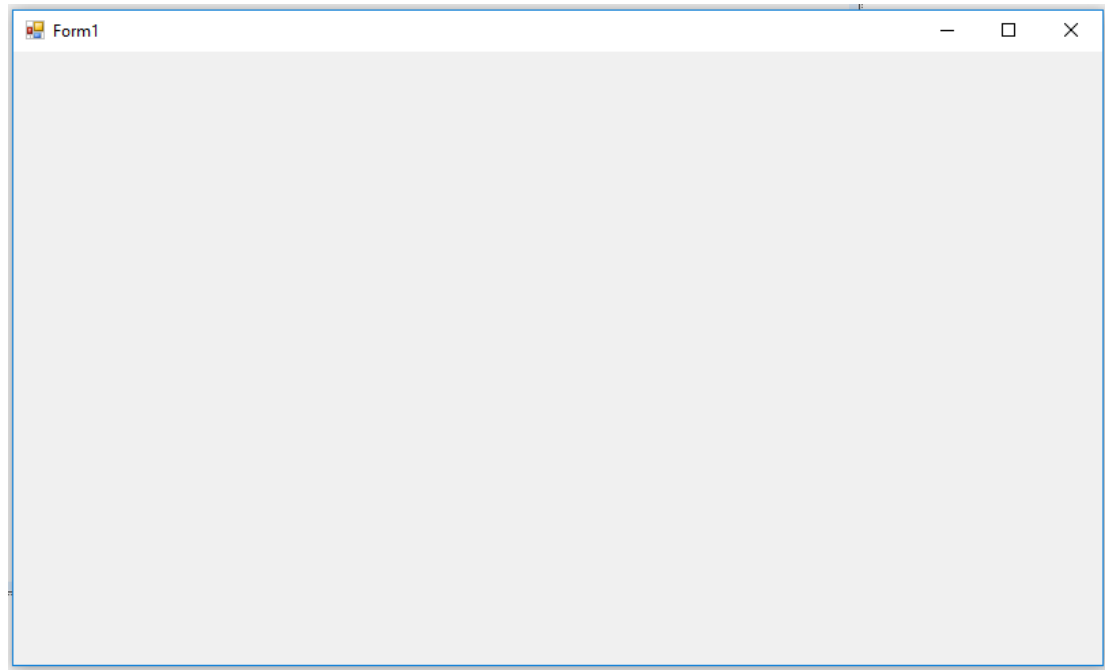
Stop the application and go back to Visual Studio.

To make sure we run the new project we need to right click on the project and set it as start-up project for this solution.



Once you have done this run the solution again to see what happens.

You should see a windows desktop app appear rather than the Connect 4 web application.



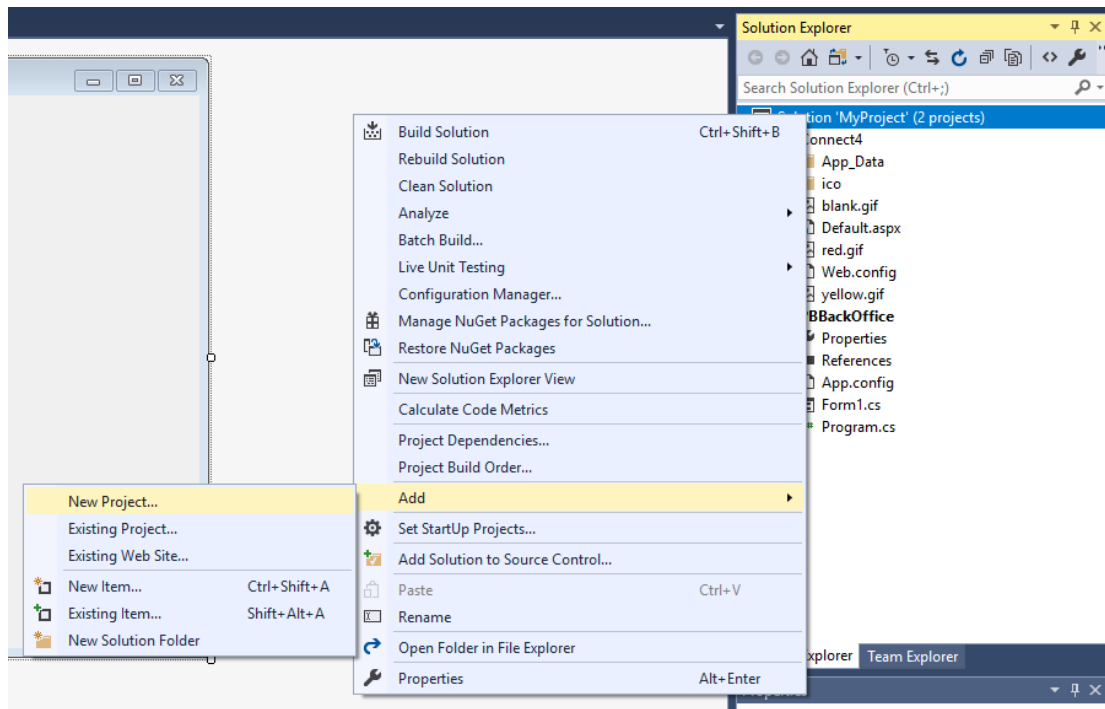
Close the program and go back to Visual Studio.

We now have two projects in the solution and a mechanism for switching between them.

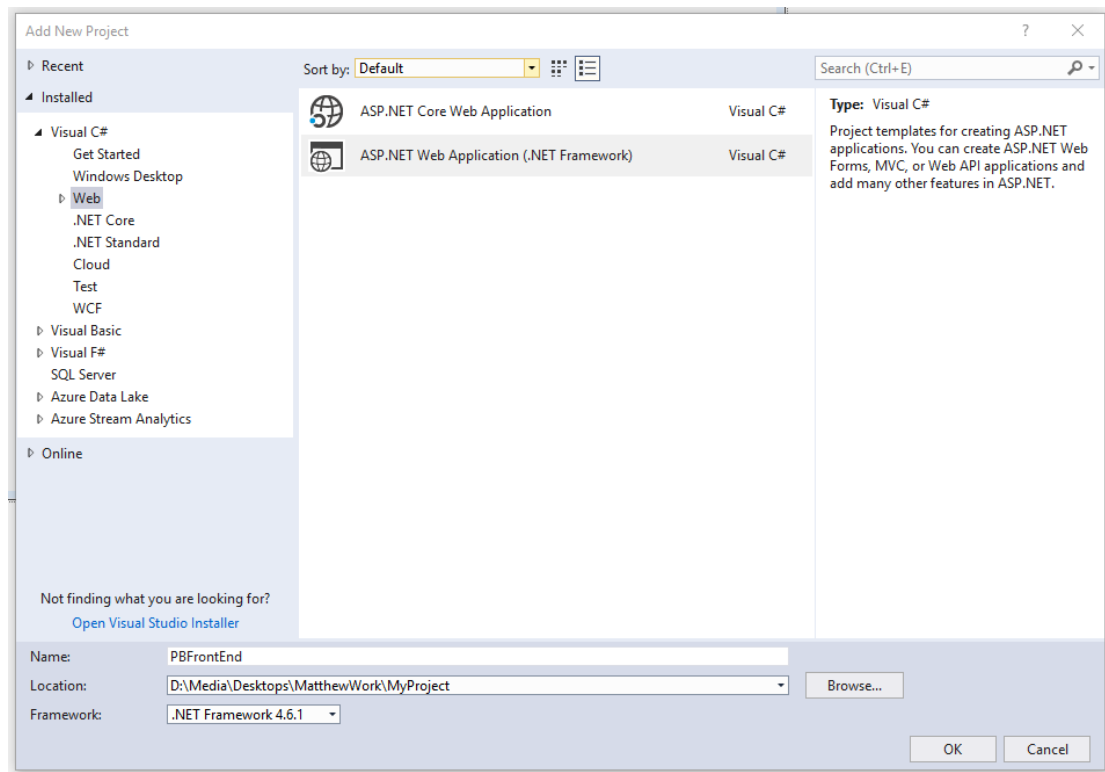
### ***Creating the Project Bank Web Application***

Having created the back office for the application we need to set up the public facing web site.

Again right click on the solution and select Add - New Project...

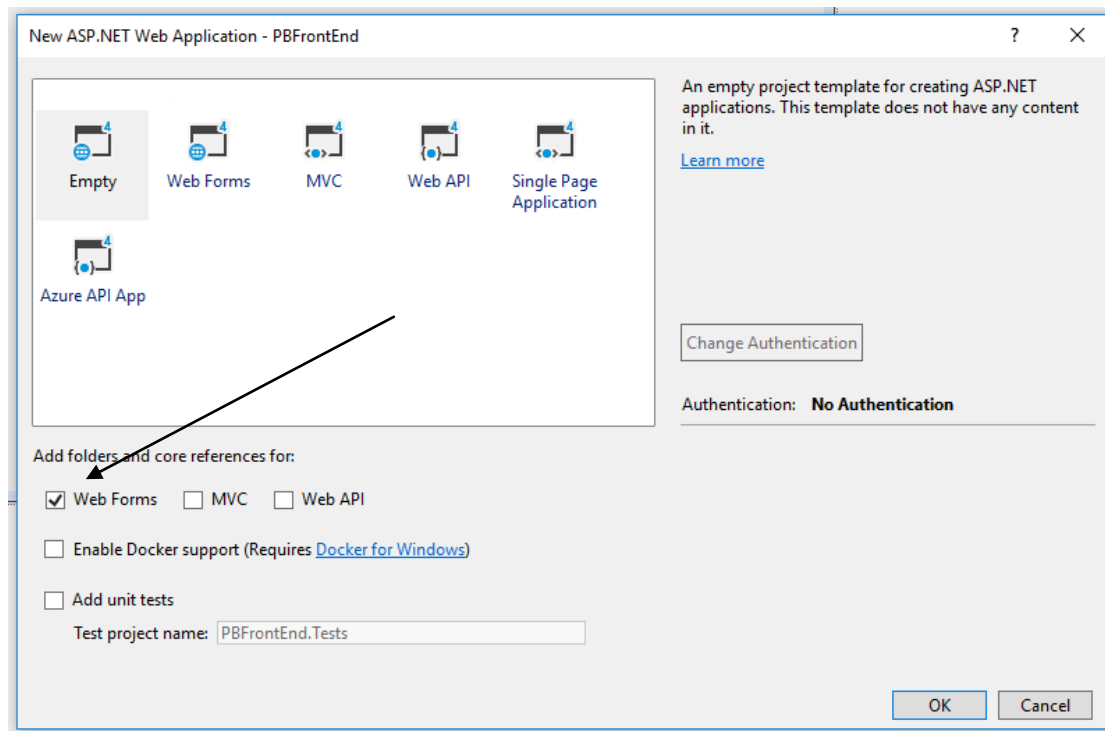


Set up the new web site like so...



Make sure that the language is C# and that you are creating the web site (PBFrontEnd) inside your solution folder.

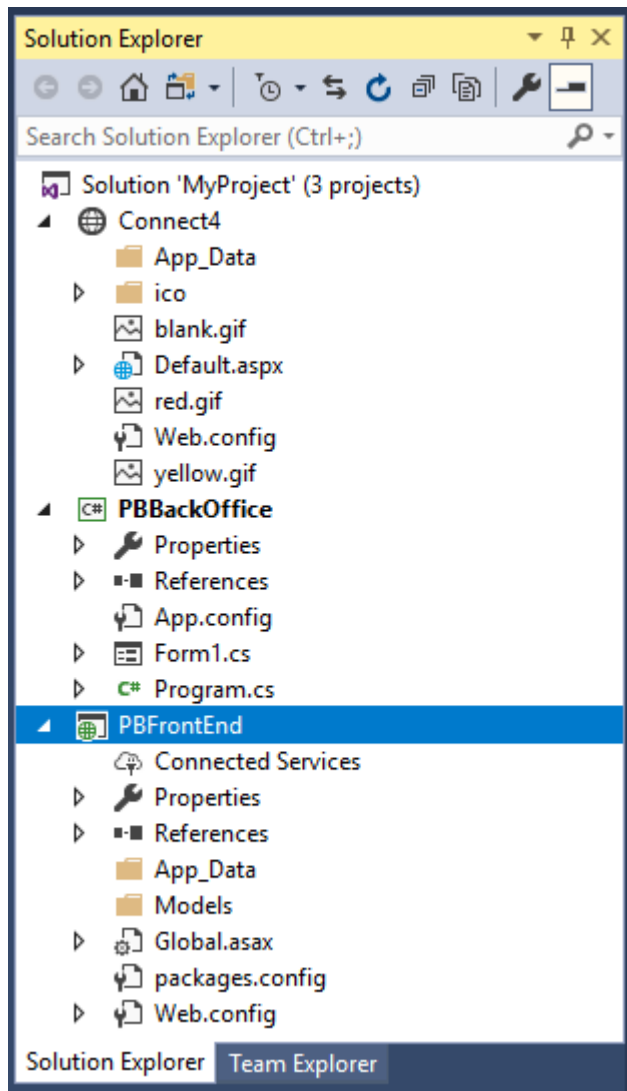
Press OK and you will be presented with the option to select which components to include in your web site...



Make sure that Web Forms is selected.

Press OK and the solution explorer should look like this...





## ***Creating the Presentation Layer in PBFrontEnd***

One important thing to consider as a developer is that there is no point in creating extra work for ourselves.

If there is an easy way to do things then let's do it!

In order to create the presentation layer we will set up three types of files.

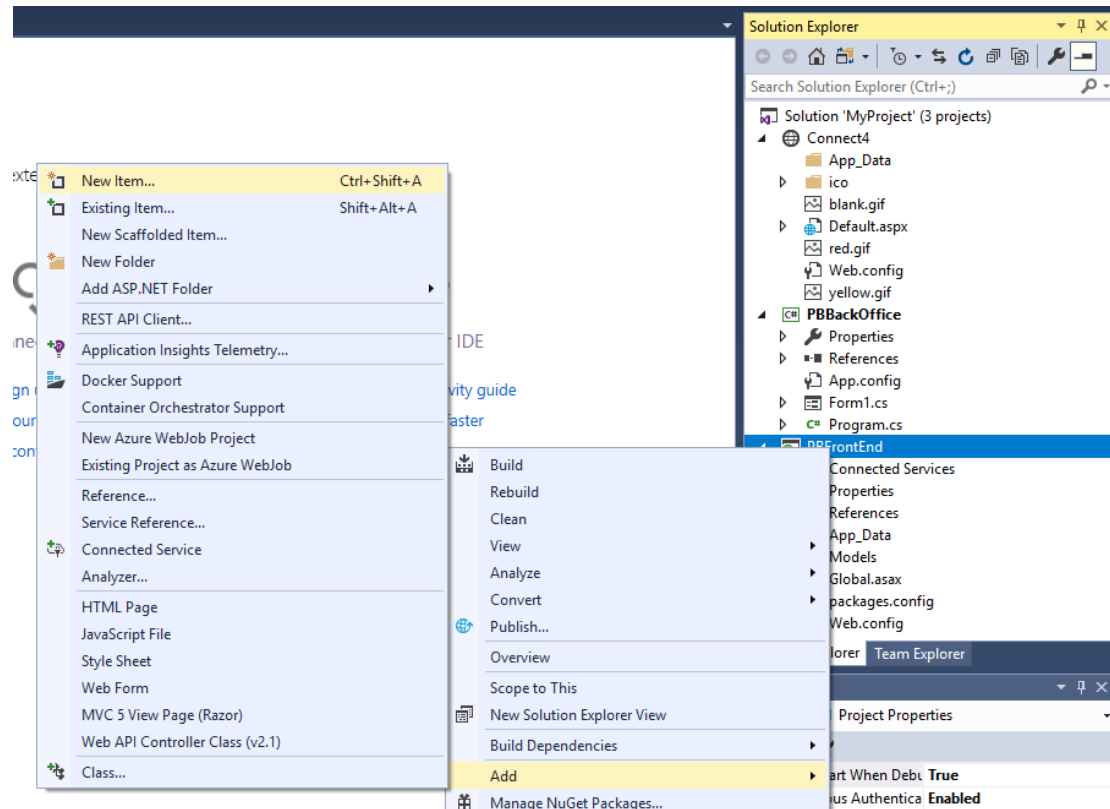
- 1 x External style sheet
- 1 x Master page
- N x Web Forms

Right click on the web form you created to test the project and delete it.

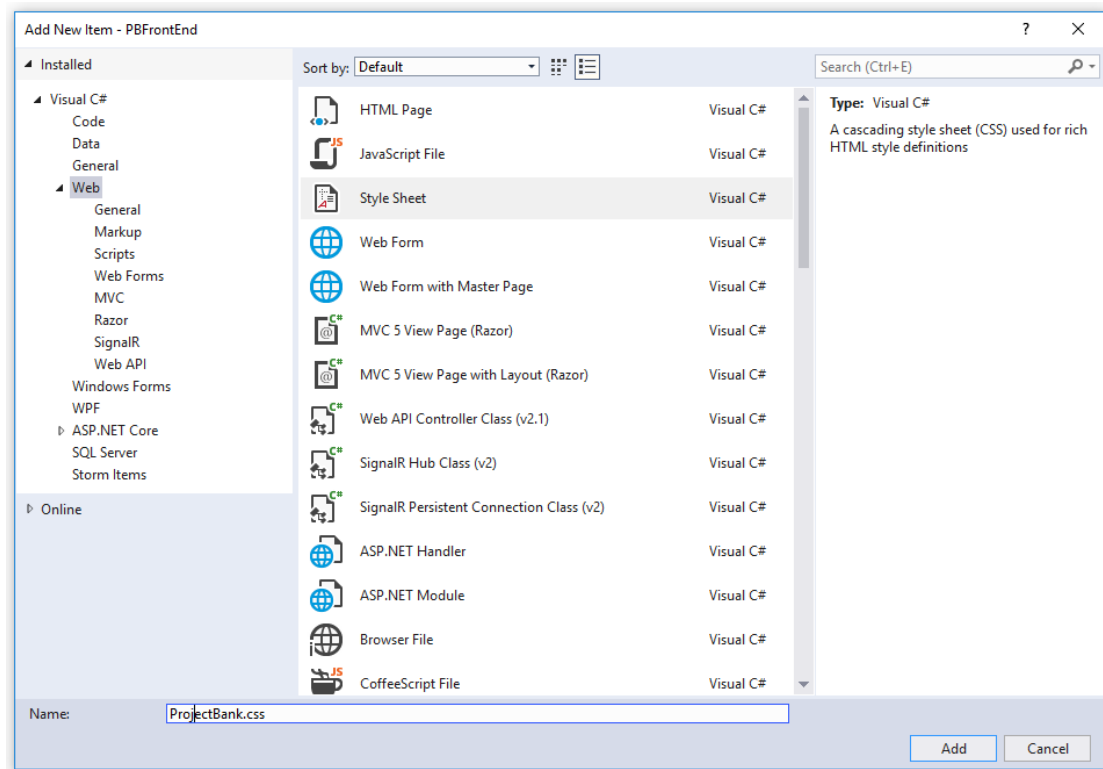
### ***The External Style Sheet***

The external style sheet or cascading style sheet (CSS) allows us to centralise the code that controls the look and feel of the pages we will create.

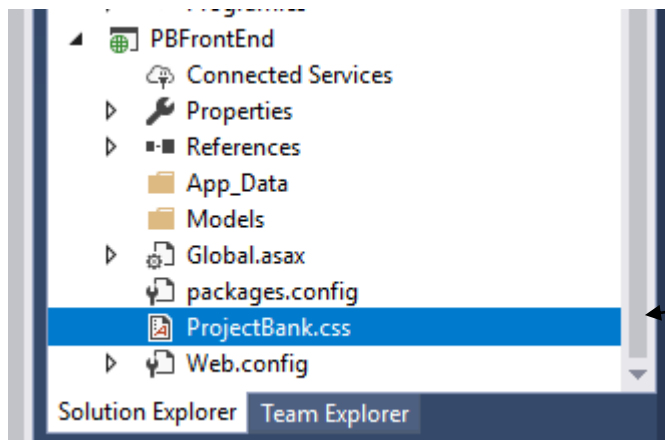
To create a new CSS right click on your web site in the solution explorer. Select Add New Item.



Find Style Sheet in the list of items and name it ProjectBank.css



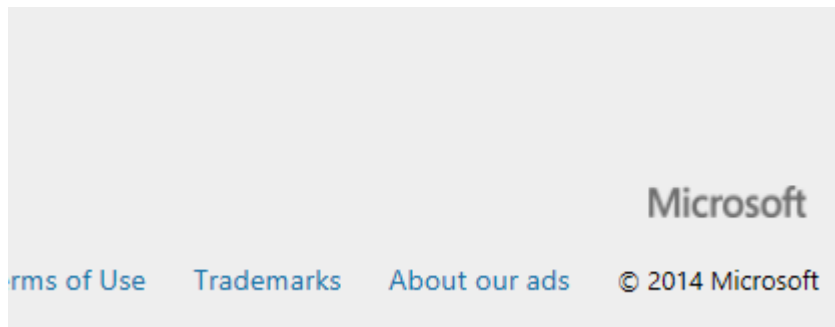
Press Add and the CSS will be created for you.



## Creating the Master Page

Master pages are a useful tool allowing you to manage the layout of multiple pages in a site.

Imaging that you have a web site of 100 pages and you want to change the copyright date on each page.



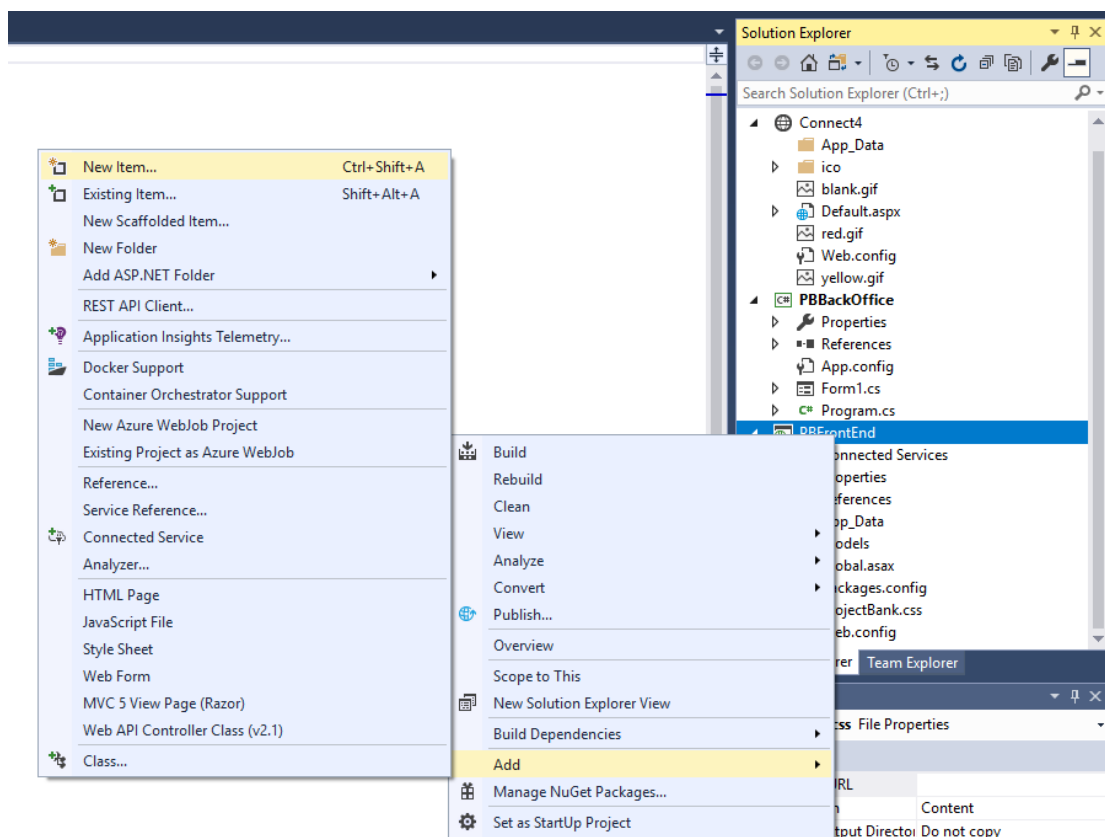
If this text exists on every single page in the site then you will need to make the same change 100 times.

The idea of a master page is that you create text that appears on every page in the master page.

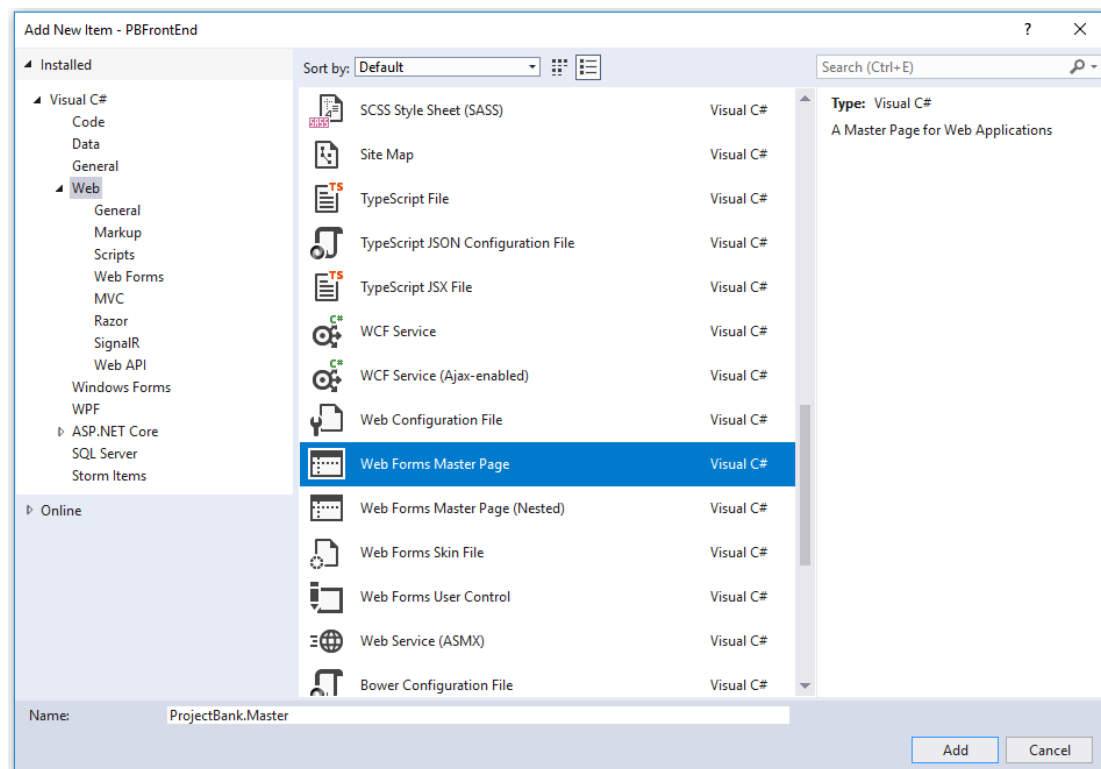
Within the master page you create a special area that will contain the text that is unique on each page across the site.

Creating the master page is a similar procedure to creating the CSS.

Right click on the web site and add a new item.



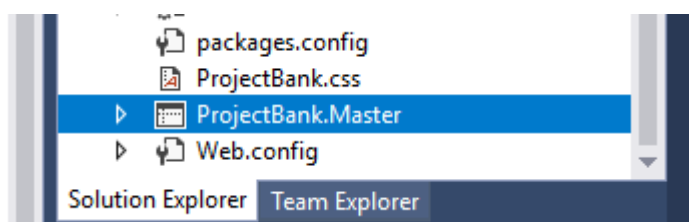
This time select Master Page...



Set the name of the master page as above.

Make sure that you set the language as C#.

This should create the master page within your web site.



## ***Linking the CSS and the Master Page***

Copy and paste the code below and replace all of the code in your master page. (If you have used different names for the CSS and or master Page you will need to edit accordingly!)

```
<%@ Master Language="C#" AutoEventWireup="true"
CodeBehind="ProjectBank.master.cs" Inherits="PBFrontEnd.ProjectBank" %>

<!DOCTYPE html>

<html>
  <head id="Head1" runat="server">
    <title>Sample Master Page</title>
    <link href="ProjectBank.css" rel="stylesheet" type="text/css" />
  </head>

  <body>
    <form id="form1" runat="server">

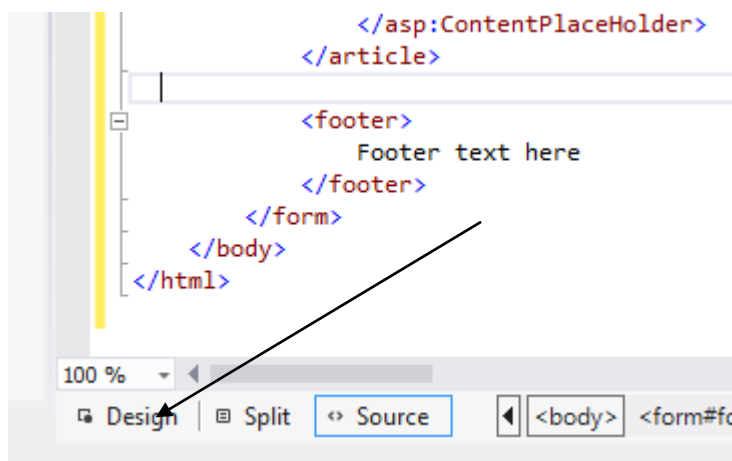
      <header>
        Header text here
      </header>

      <article>
        Your text here
        <asp:ContentPlaceholder id="ContentPlaceholder1"
runat="server">
          </asp:ContentPlaceholder>
        </article>

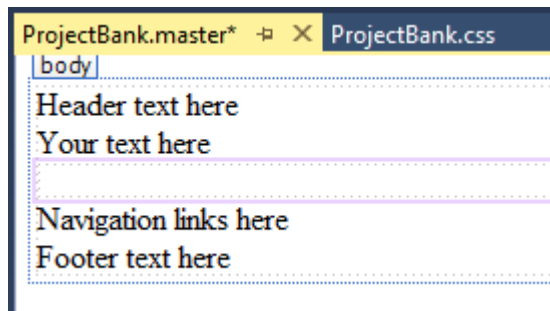
      <nav>
        Navigation links here
      </nav>

      <footer>
        Footer text here
      </footer>
    </form>
  </body>
</html>
```

To see what the master page looks like flip to design view by pressing the design button at the bottom of the screen.



Your master page should look something like this...



Impressive eh?

It may not look very pretty but we have done some important things here.

Flip back to the code view for the master page and we shall look at some sections of the HTML for the page.

The first code of interest is

```
<!DOCTYPE html>
```

This tells us that the page is constructed using HTML .

HTML 5 uses tags to identify each section of the document.

A very simple HTML 5 document has the following tags...

```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="AddressBook.master.cs"
Inherits="AddressBook" %>

<!DOCTYPE html>

<html>
  <head>
    <title>Page Title Here</title>
    <link href="ProjectBank.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    Page text goes here
  </body>
</html>
```

Each tag has an open tag e.g. <body> and a matching closing tag </body>

Any tags that are opened must have a matching closing tag.

Tags are case sensitive and must match.

One important bit of information in this page is the code that links the page to the CSS

```
<link href="ProjectBank.css" rel="stylesheet" type="text/css" />
```

This tells the master page to obtain its formatting from the external CSS.

One approach to applying formatting in HTML 5 is to mark up the page using special tags provided by the language.

In this case we are using the following tags...

```
<header>
  Header text here
</header>

<article>
  Your text here
  <asp:ContentPlaceHolder id="ContentPlaceholder1" runat="server">
  </asp:ContentPlaceHolder>
</article>

<nav>
  Navigation links here
</nav>

<footer>
  Footer text here
</footer>
```

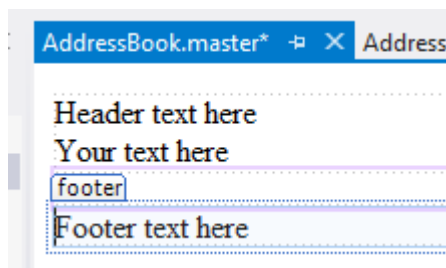
These tags identify the sections of the page for different purposes.

The final bit of mark-up of interest is the content placeholder.

```
<asp:ContentPlaceHolder id="ContentPlaceholder1" runat="server">
</asp:ContentPlaceHolder>
```

The content placeholder is used to create the area of the page that will be different on each page we create. We shall see how this works in a moment.

Since the page looks like this...



We need to apply some formatting to the master page to make it start to look like a web page.



Copy and paste the following code replacing the existing code in the external style sheet.

```
body
{
    background-color: #6699FF;
}

header
{
    background-color: #600;
    height: 10%;
    position: fixed;
    width: 100%;
    top: 0%;
    left: 0%;
}

footer
{
    background-color: #600;
    height: 10%;
    position: fixed;
    width: 100%;
    top: 90%;
    left: 0%;
}

nav
{
    background-color: #FF5050;
    position: fixed;
    width: 20%;
    height: 90%;
    top: 10%;
    left: 0%;
}

article
{
    background-color: #CC66FF;
    position: fixed;
    width: 80%;
    top: 10%;
    left: 20%;
}
```

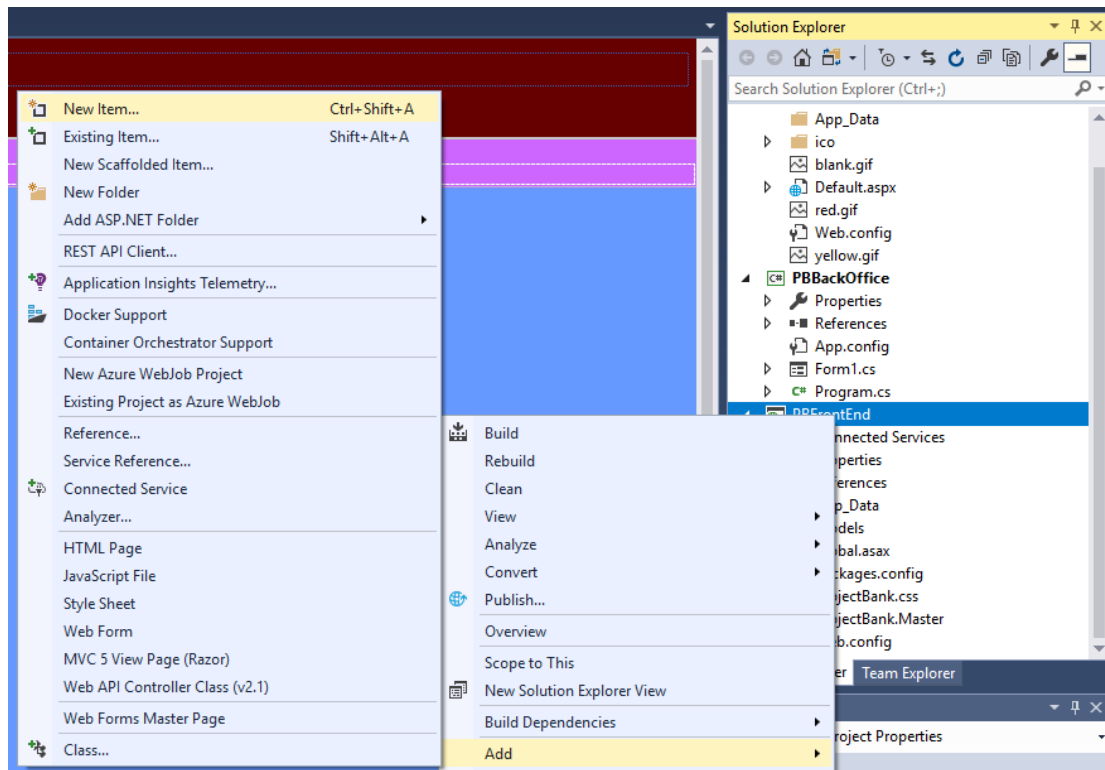
Close and save the external style sheet and now look at the master page in design view...



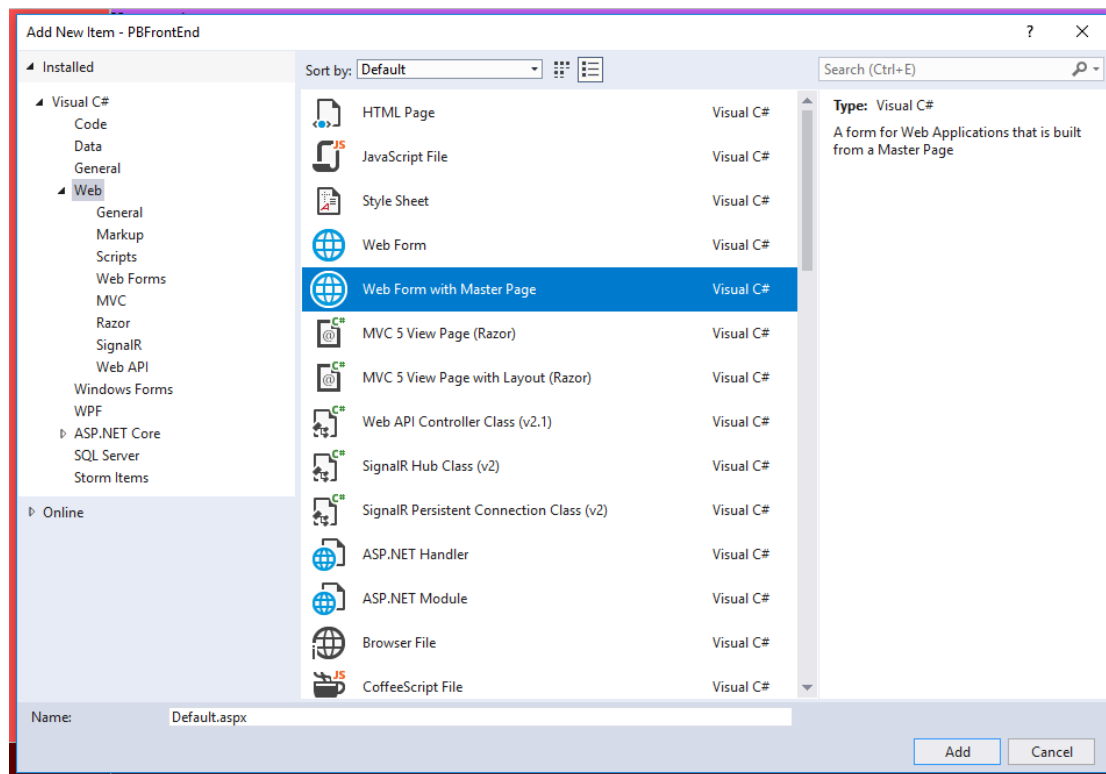
Not the prettiest web site in the world but it is starting to make a bit more sense.

## ***Creating the Web Form***

To create the Default.aspx web form, from the main menu select WEBSITE – ADD NEW ITEM



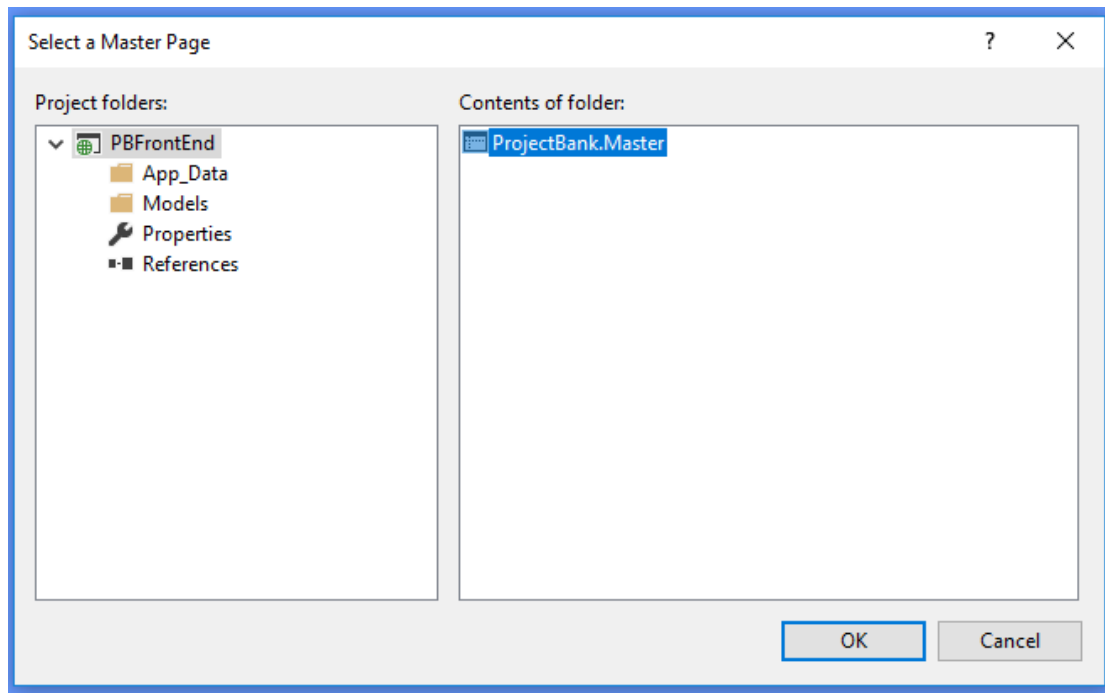
You should see the following page...



Notice that the name of the first page is called Default.aspx. This page name has special significance to the system. It tells the system that this is the main page for the entire site.

Also make sure that the language Visual C# is selected.

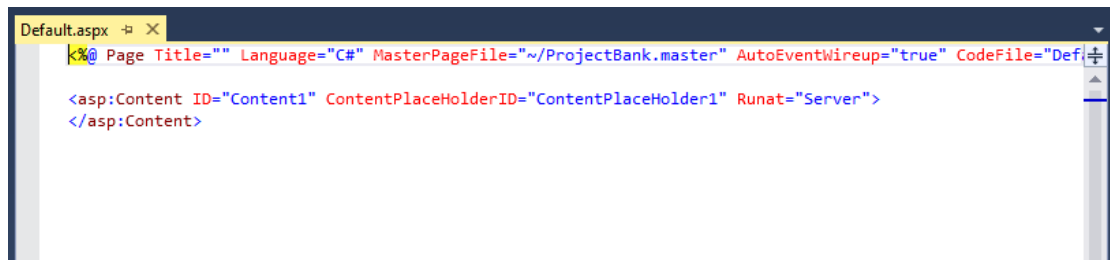
Click “add” and Visual Studio will ask you the name of the master page to use.



Select the master page and press OK.

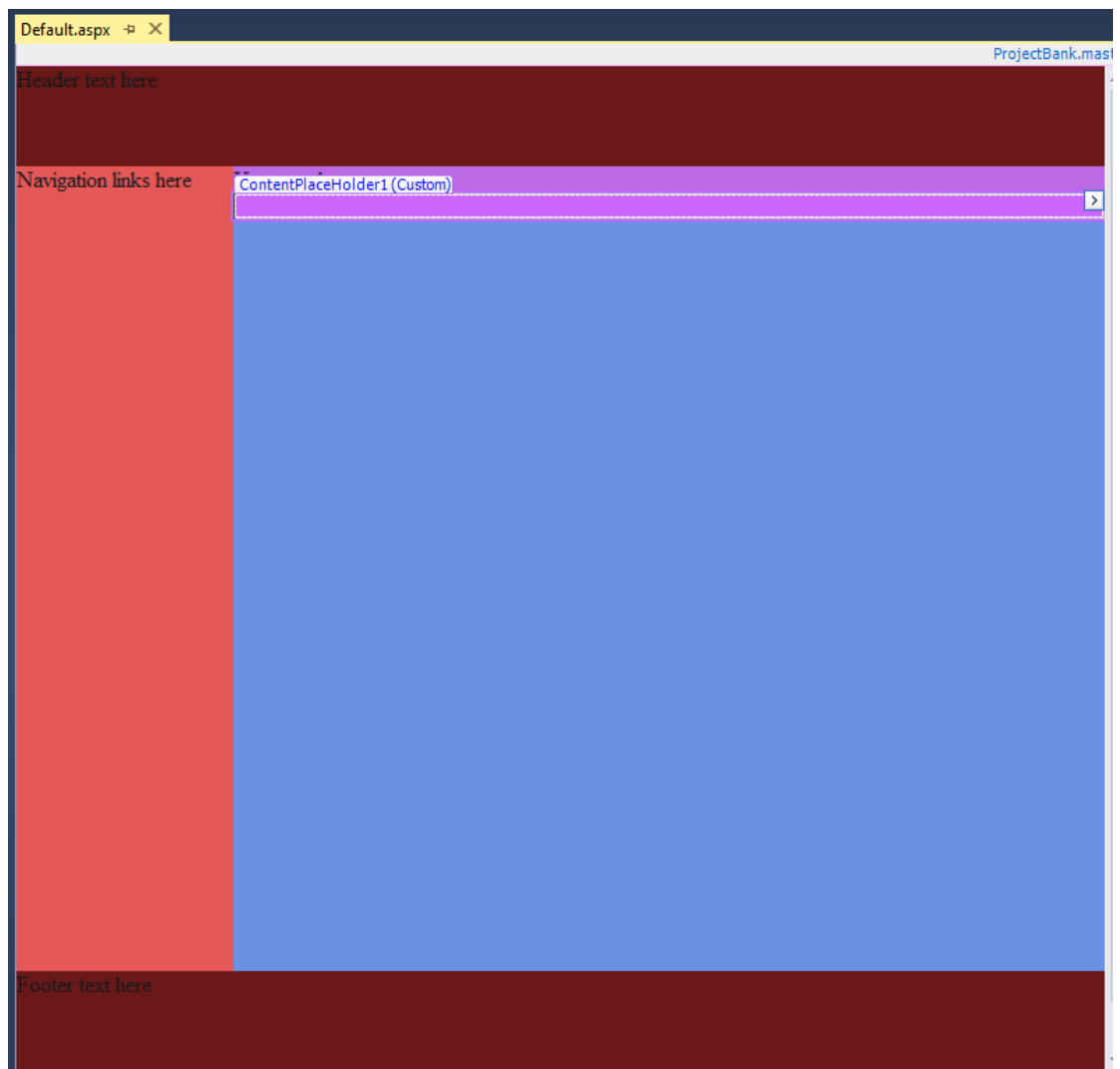
You should now have a new page added to your site.

You should see the following...



```
Default.aspx - [X]
k% Page Title="" Language="C#" MasterPageFile="~/ProjectBank.master" AutoEventWireup="true" CodeFile="Def
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
</asp:Content>
```

Switch the web form to design view to see what it looks like.



What is happening is that the new page is inheriting its formatting from the CSS via the master page.

## ***Modifying the Form***

The first thing to notice is what you can't modify.

Try changing the text “Footer text here” to something else.

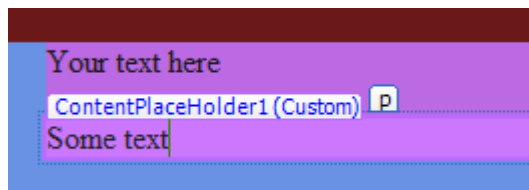
You will find that you can't as this text is controlled by the master page.

To change this text you need to do it from the master page.

Place your cursor in the section of the page marked so...



You should be able to type here



What you need to understand is what parts of the site belong to the master page and which parts belong to the individual pages.

If there is some text or formatting that needs to be on every single page in your site then it needs to go in the master page.

If there is text or formatting that only exists on one page then it needs to go into that page and NOT the master page.

To finish off the work so far modify the style sheet such that the colour scheme is something more attractive and professional.