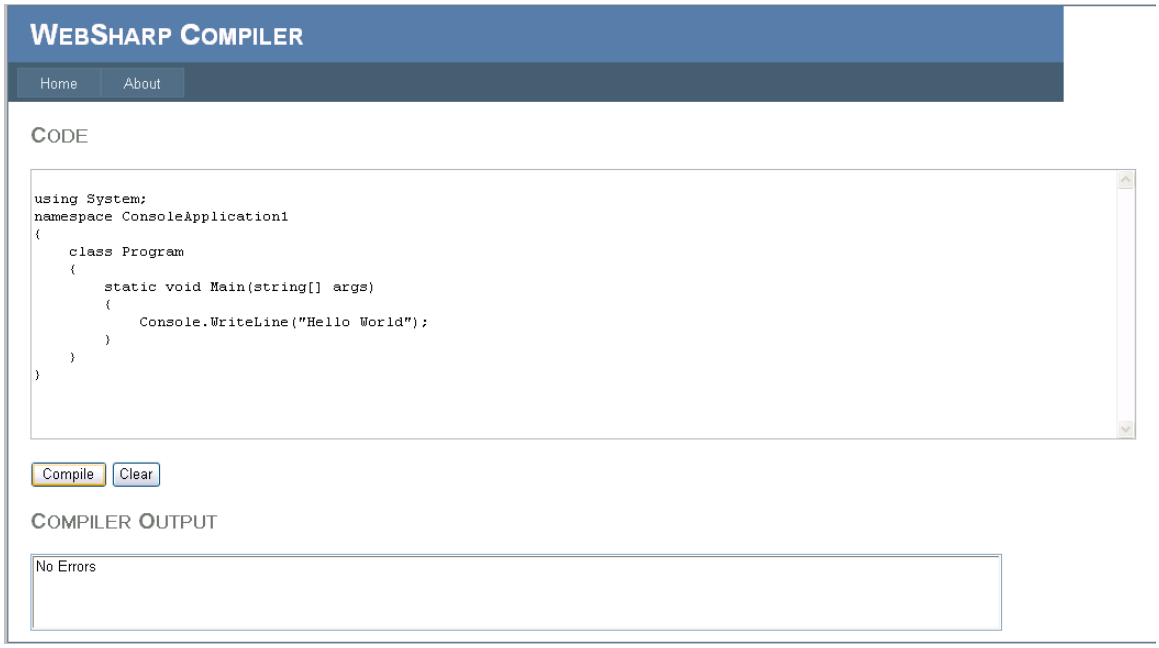
WebSharpCompiler

Building a web based C# compiler using ASP.NET and TDD

Author: Dominic Millar
Tech Review:Matt Rumble
Editor:Cathy Tippett
Feb 2011



http://domscode.com

Introduction

This tutorial is an introduction to Test Driven Development (TDD) in Visual Studio 2010 (VS2010) with C# 4 and ASP.NET 4. This will focus on as using the tool and coding as much as possible and not a great deal of words so enjoy the ride.

We are covering some material that was presented in my earlier tutorial http://domscode.com/2010/05/21/net-tutorial-c4-0-and-test-driven-development-cannonattack/ so if you haven't used VS2010 and TDD have a look.

The WebSharpCompiler Requirements/Specs:

The following is a combination of requirements and specifications that will give us an idea of what we are trying to build:

1. ASP.NET Application;
2. Textbox will contain code to compile
3. Empty Textbox is not allowed to be compiled
4. A Compile button will trigger compile
5. A Clear button will clear the textbox
6. When [Compile] is clicked, text from textbox will be compiled using a C# compiler and messages from compile displayed in a Listbox

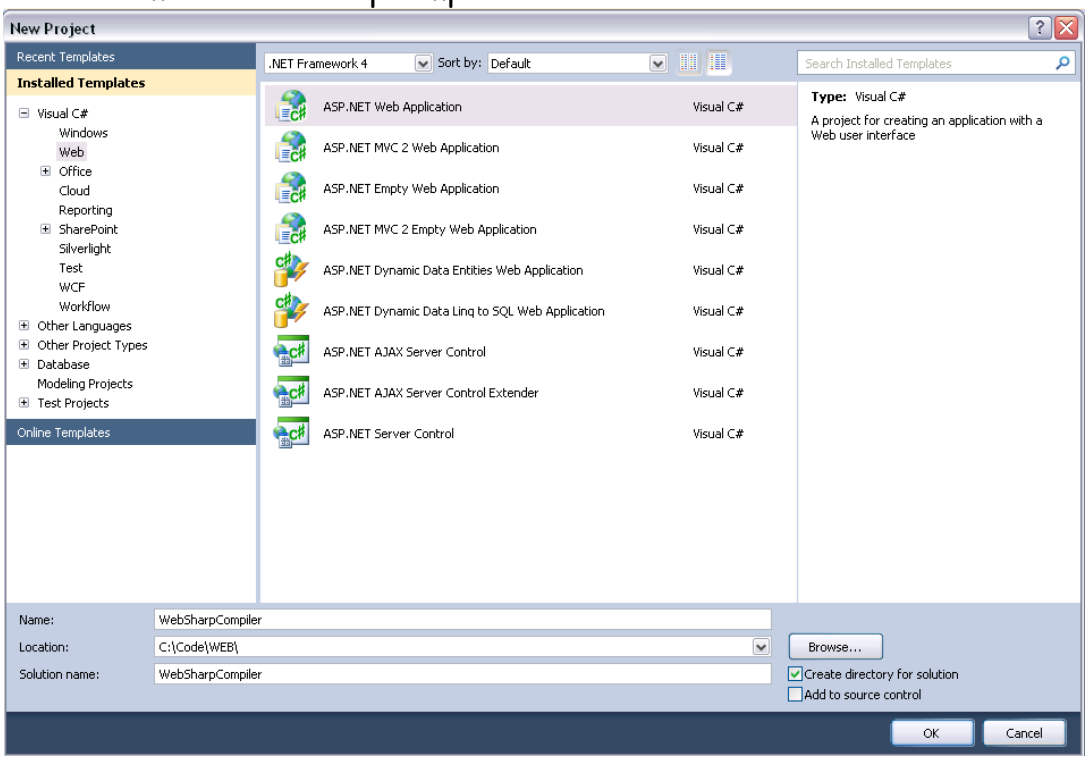
Iteration 1

We are going to setup the project and build our first test.

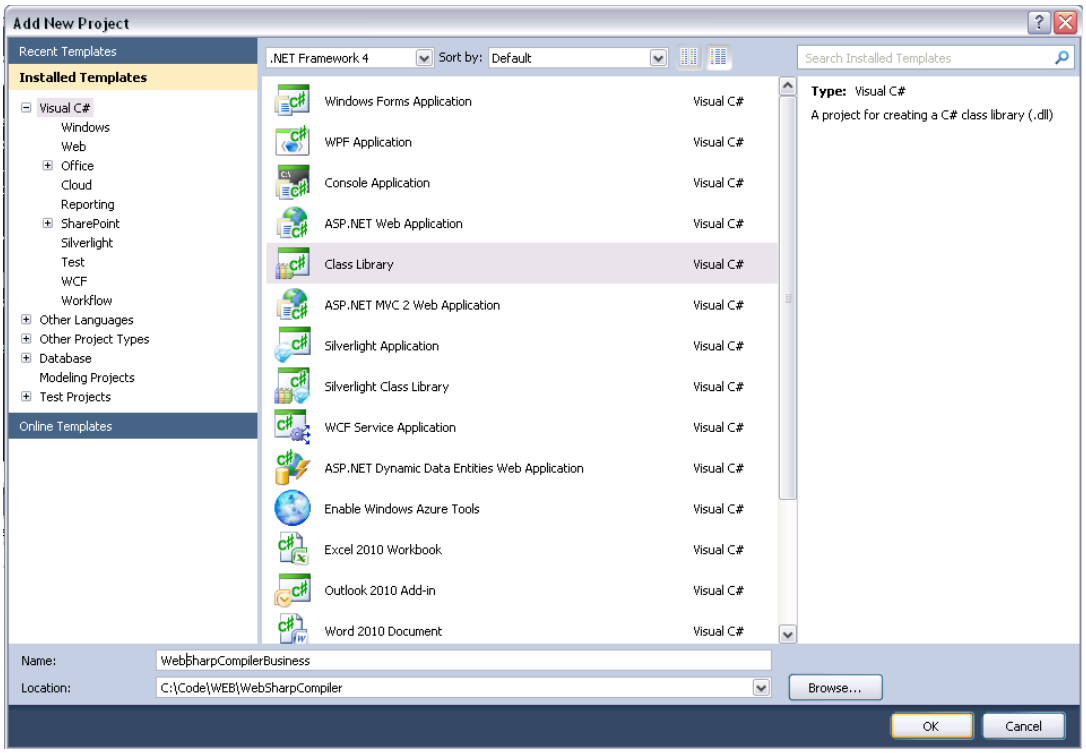
Iteration 1 – *Creating the Solution and our first test*

- Start Visual Studio

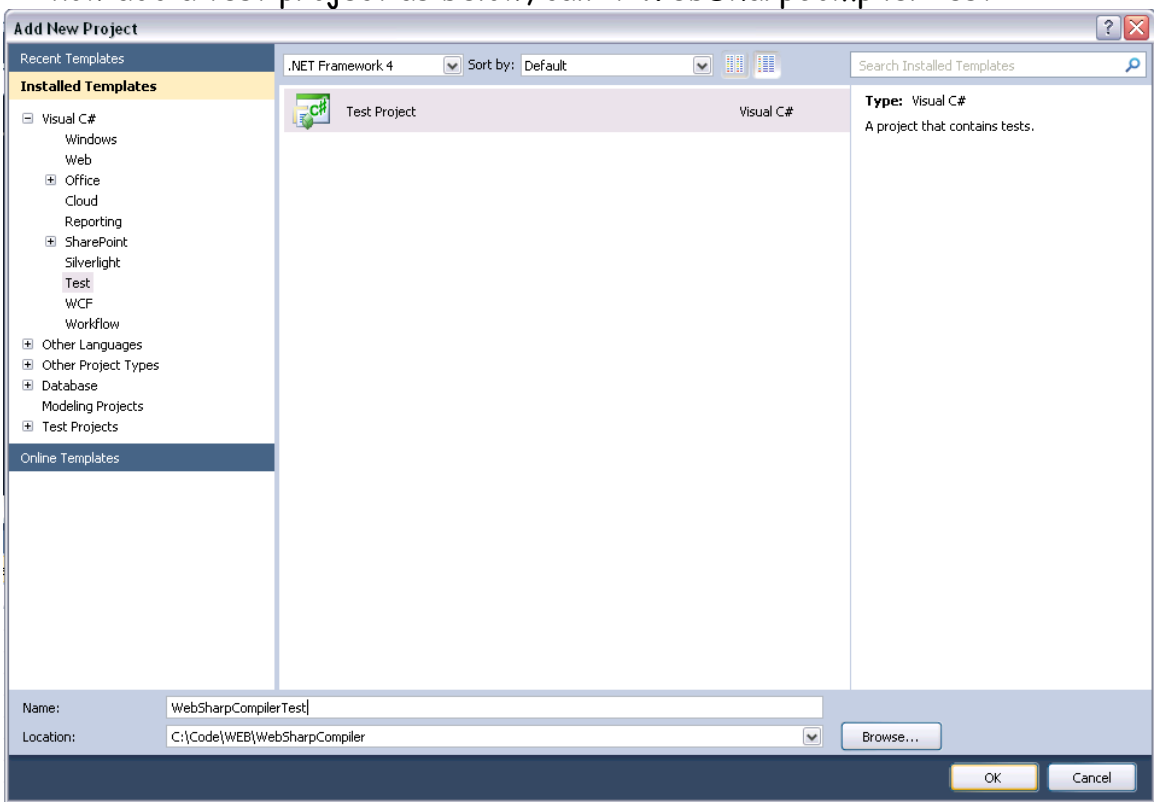
- Click the File menu
- Click New Project…
- Click Web
- Select ASP.NET Web Application
- Name it WebSharpCompiler



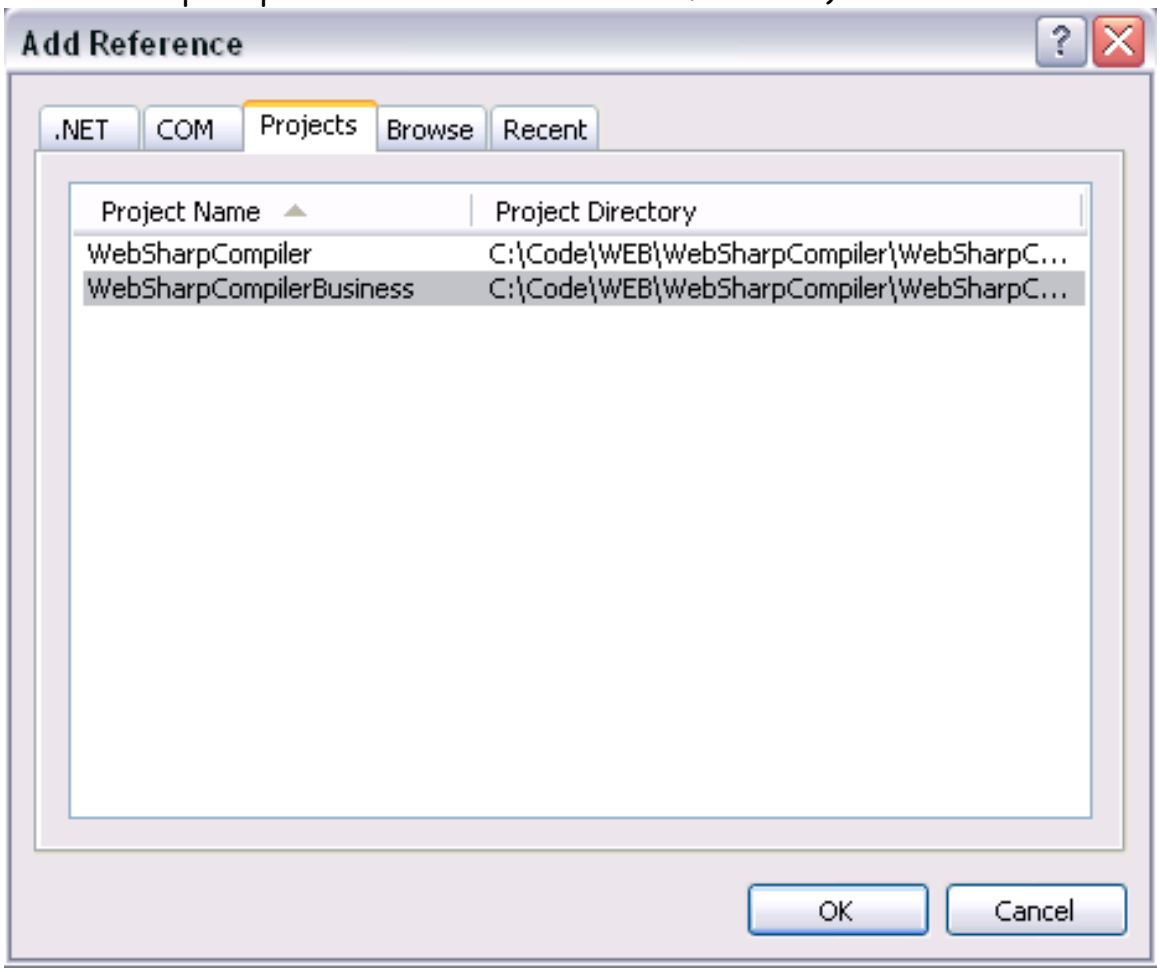
- now add a Class library project (right click on the solution and select ADD->New Project) as below, call it WebSharpCompilerBusiness:



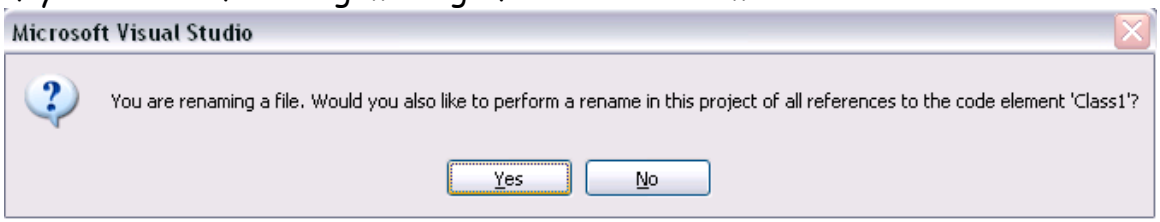
- now add a test project as below, call it WebSharpCompilerTest



- Add a reference to the WebSharpCompilerBusiness class library project from the WebSharpCompilerTest project (click on References in WebSharpCompilerTest and select ADD Reference…)

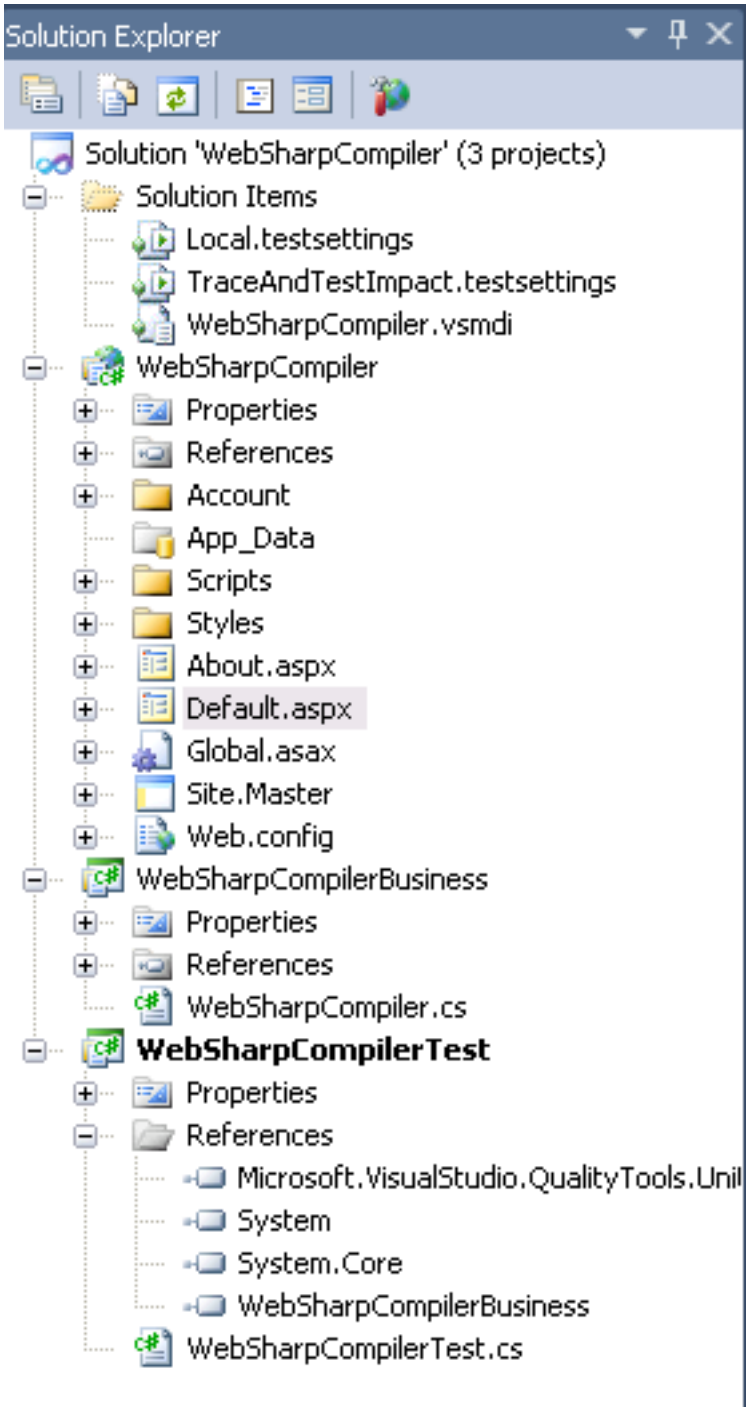


- Also add a reference from the WebSharpCompiler web project to the WebSharpCompilerBusiness(we need this when we create the web page later on).
- In WebSharpCompilerTest rename the file UnitTest1.cs to WebSharpCompilerTest.cs.
- rename Class1 in WebSharpCompilerBusiness to WebSharpCompiler – if you see the following message for either rename actions:



- click Yes

- Right click on the WebSharpCompilerTest and select Set as Startup Project.
- Solution should now look like



- We need to add our first test so double click on WebSharpCompilerTest.cs.

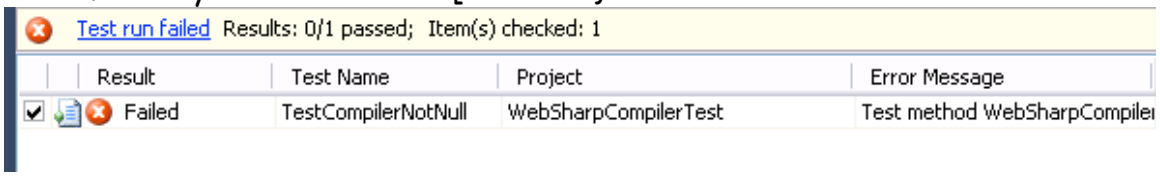
- Add the following at the top of the file :
  Using WebSharpCompilerBusiness ;
- replace existing TestMethod1 with the following code:

```
[TestMethod]
public void TestCompilerNotNull()
{
    WebSharpCompiler compiler = new WebSharpCompiler();
    Assert.IsNotNull(compiler.Compile(""));
}
```

- As you can see we call Compile but it doesn't exist so lets add a new method
- Right click on the Compile text and select Generate -> Method Stub
- If we look at the generated code in WebCompilerTest.cs we see:

```
public object Compile(string p)
{
    List<string> messages = new List<string>();

    if (string.IsNullOrEmpty(p))
    {
        messages.Add("program text cannot be null or empty");
    }

    return messages;
}
```

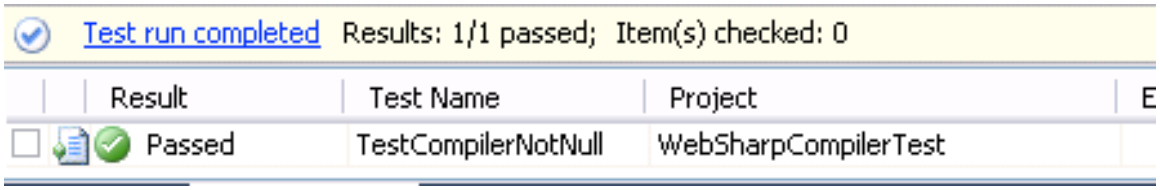
- Now if we try to run the test [CTRL-F5} we should see



- So that's RED in our good old Red Green Refactor process. Now its time to fix this method as follows:

```
public object Compile(string p)
{
    List<string> messages = new List<string>();

    if (string.IsNullOrEmpty(p))
    {
        messages.Add("program text cannot be null or empty");
    }

    return messages;
}
```

- If we run the test now we find…



- OK so that's the green we wanted, so now time to refactor. So we'll clean up the usings, rename the parameter and replace the return type with a more specific type as follows:

```
public List<string> Compile(string programText)
    {
        List<string> messages = new List<string>();
        if (String.IsNullOrEmpty(programText))
        {
            messages.Add("program text cannot be null or                empty");
        }
        CompilerResults compilerResults = ProcessCompilation(programText);
        foreach (CompilerError error in compilerResults.Errors)
        {
            messages.Add(String.Format("Line {0} Error No:{1} -
{2}", error.Line, error.ErrorNumber, error.ErrorText));
        }

        return messages;
    }
```

Just remember you can use some cool VS2010 features to make these changes including:

- Right click on the code window and click organize usings->Remove and Sort
- Right click on a variable and click refactor->Rename

Iteration 2

- Lets add another test

```
[TestMethod]
  public void TestCompilerSingleError()
  {
      WebSharpCompiler compiler = new WebSharpCompiler();

      string programText= @"
          using **** System;

          namespace HelloWorld
          {
              class HelloWorldClass
              {
                  static void Main(string[] args)
                  {
                      Console.ReadLine();
                  }
              }
          }";
```

```
        List<string> compilerErrors = compiler.Compile(programText);

        Assert.AreEqual(compilerErrors.Count, 1);
    }
```

- We run the tests and find it fails so lets add the code needed to make it pass:

```
namespace WebSharpCompilerBusiness
{
    public class WebSharpCompiler
    {
        public object Compile(string p)
        {
            List<string> messages = new List<string>();

            if (string.IsNullOrEmpty(p))
            {
                messages.Add("program text cannot be null or empty");
            }

            messages.Add("program contains 1 error");
            return messages;
        }
    }
}
```

- Now when we run the tests we see that they work
- Of course its time to refactor so lets do it properly as follows:

```
using System;
using System.CodeDom.Compiler;
using System.Collections.Generic;
namespace WebSharpCompilerBusiness
{
    public class WebSharpCompiler
    {
        public List<string> Compile(string programText)
        {
            List<string> messages = new List<string>();
            if (String.IsNullOrEmpty(programText))
            {
                messages.Add("program text cannot be null or empty");
            }
            CompilerResults compilerResults = ProcessCompilation(programText);
            foreach (CompilerError error in compilerResults.Errors)
            {
                messages.Add(String.Format("Line {0} Error No:{1} -
{2}", error.Line, error.ErrorNumber, error.ErrorText));
            }

            return messages;
        }

        public CompilerResults ProcessCompilation(string programText)
        {
            CodeDomProvider codeDomProvider = CodeDomProvider.CreateProvider("CSharp");
            System.CodeDom.Compiler.CompilerParameters parameters = new CompilerParameters();
            parameters.GenerateExecutable = false;
            System.Collections.Specialized.StringCollection assemblies = new System.Collection
s.Specialized.StringCollection();
            return codeDomProvider.CompileAssemblyFromSource(parameters, programText);
        }
    }
}
```

So that's almost the end of the 2nd iteration

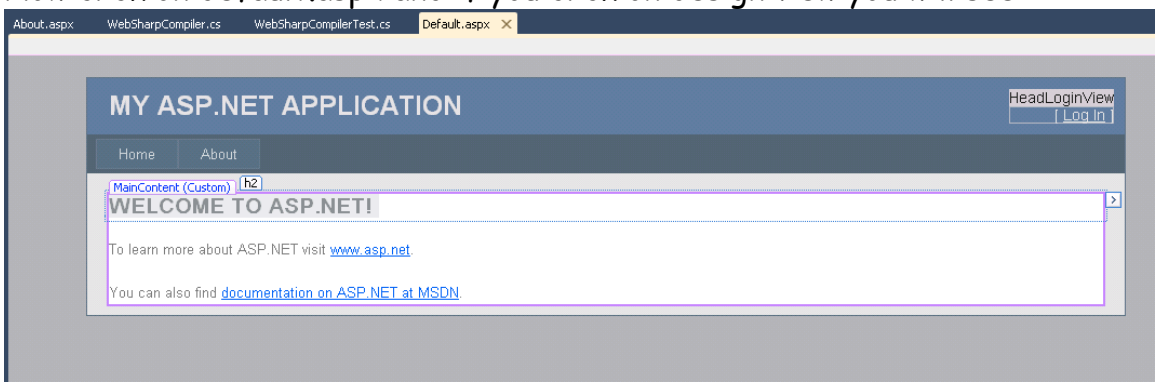
It is worthwhile adding some more tests at this time although I wont walk through the process anymore but I'll show you the tests we added:

```
[TestMethod]
public void TestCompilerFiveErrors()
{
    WebSharpCompiler compiler = new WebSharpCompiler();

    string programText = @"
        using **** System;

        namesp8ce HelloWorld
        {
            clas HelloWorldClass
            {
                static void Main(string[] args)
                {
                    Console.ReadLine();
                }
            }";

    List<string> compilerErrors = compiler.Compile(programText);

    Assert.AreEqual(compilerErrors.Count, 5);
}

[TestMethod]
public void TestCompilerSuccessfulCompilation()
{
    WebSharpCompiler compiler = new WebSharpCompiler();

    string programText = @"
        using System;

        namespace HelloWorld
        {
            class HelloWorldClass
            {
                static void Main(string[] args)
                {
                    Console.ReadLine();
                }
            }
        }";

    List<string> compilerErrors = compiler.Compile((programText));

    Assert.AreEqual(compilerErrors.Count, 0);
}
```

3rd Iteration

The third iteration of work is to complete the web site and given that the UI is very simple this won't take very long.

- Now creating the website is very simple, first right click on the Web Application project and select Set as startup project.
- Now click on default.aspx and if you click on design view you will see:



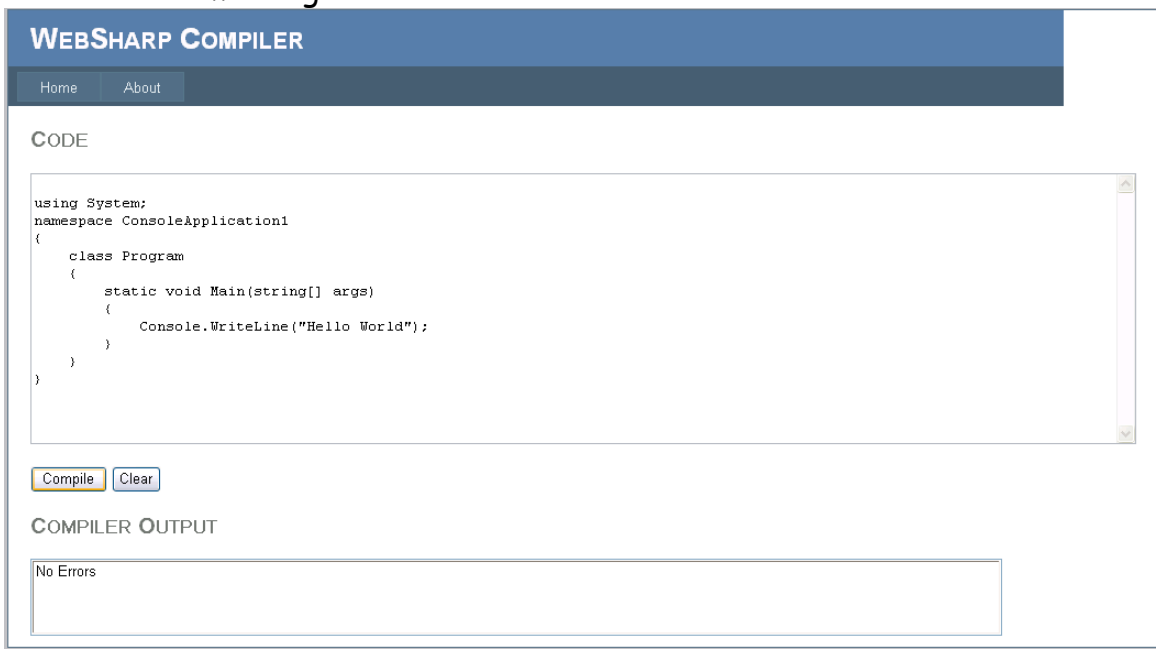
- Lets make all the changes to the markup we need and our markup of default.aspx should look like:

```
<%@ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.master" AutoEventWireup
="true"
    CodeBehind="Default.aspx.cs" Inherits="WebSharpCompiler._Default" %>
<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">
</asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
    <h2>Code</h2>
    <p>

        <asp:TextBox ID="txtCode" runat="server" Height="240px" Width="100%"
            TextMode="MultiLine">
using System;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine(&amp;quot;Hello World&amp;quot;);
        }
    }
}
</asp:TextBox>
    </p>
    <p>
        <asp:Button ID="btnCompile" runat="server" onclick="btnCompile_Click" Text="Compil
e" />
        <asp:Button ID="btnClear" runat="server"  onclick="btnClear_Click" Text="Clear" />
    </p>
    <h2>Compiler Output</h2>
    <p>
        <asp:ListBox ID="lstCompilerOutput" runat="server" Width="100%"></asp:ListBox>
    </p>
</asp:Content>
```

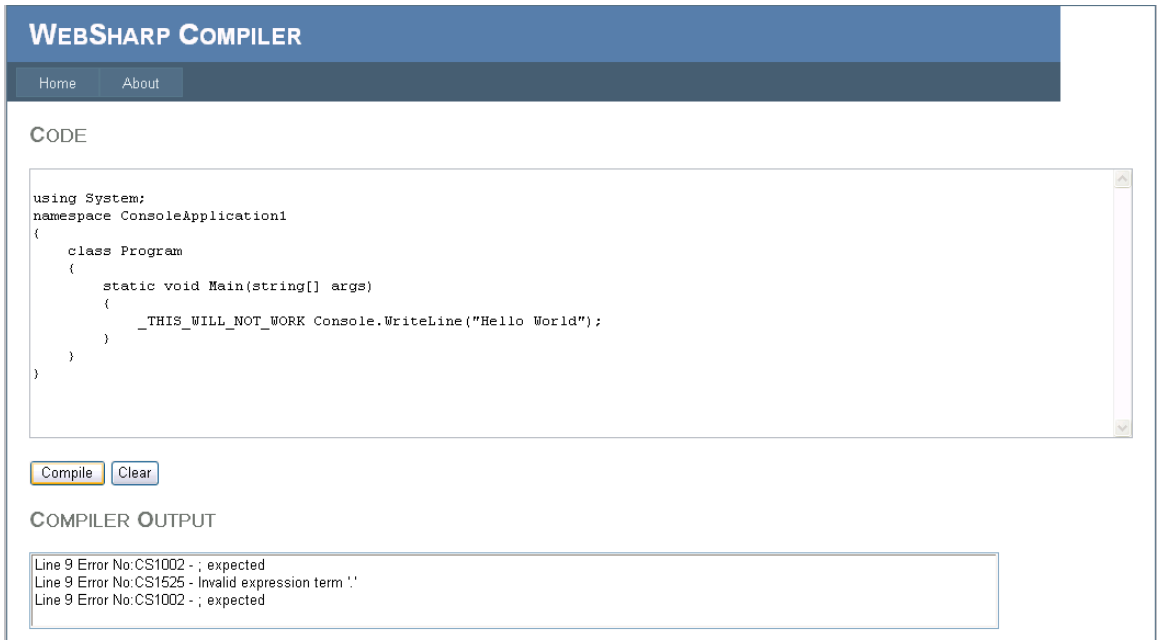
- And if we now go to the default.aspx.cs and replace with

```
using System;
using System.Collections.Generic;

namespace WebSharpCompiler
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void btnCompile_Click(object sender, EventArgs e)
        {
            lstCompilerOutput.Items.Clear();
            WebSharpCompilerBusiness.WebSharpCompiler compiler = new
             WebSharpCompilerBusiness.WebSharpCompiler();
            List<string> compilerErrors = compiler.Compile(txtCode.Text);

            if (compilerErrors.Count == 0)
            {
                lstCompilerOutput.Items.Add("No Errors");
            }

            foreach (string error in compilerErrors)
            {
                lstCompilerOutput.Items.Add(error);
            }

        }

        protected void btnClear_Click(object sender, EventArgs e)
        {
            txtCode.Text = string.Empty;
        }
    }
}
```

- You can also make some changes to the master file if you wish and you should see something like this:



or



In Conclusion

so there we have our C# compiler. Source code link is below and feel free to drop me a line at dommillar@gmail.com if you have any thoughts/advice on this.

All the best
Dom.

References

http://msdn.microsoft.com/en-us/library/system.codedom.compiler.codedomprovider.aspx

Source Code

http://websharpcompiler.codeplex.com