Jonathan Tweedle

2019/06/26

Assignment 01

Creating ASP.Net web pages

# Introduction

This assignment is a “hands on” activity to demonstrate my ability for creating ASP.Net web pages. The desired goal is to show how to create ASP web pages based on the C# .Net framework. The pages will also include some basic HTML and JavaScript.

# Getting Started

## Create an empty project

Open a new instance of Visual Studio 2019 and on the right hand side, click the button for creating a new project (Figure 1). Next select the project template from the list on the right for an ASP.NET Web Application (Figure 2), making sure to use template for the ‘C#’ language and not ‘Visual Basic’.

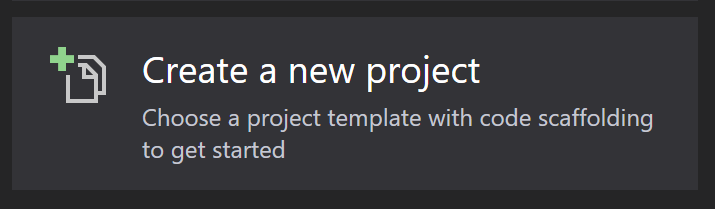


Figure 1: Create a new project

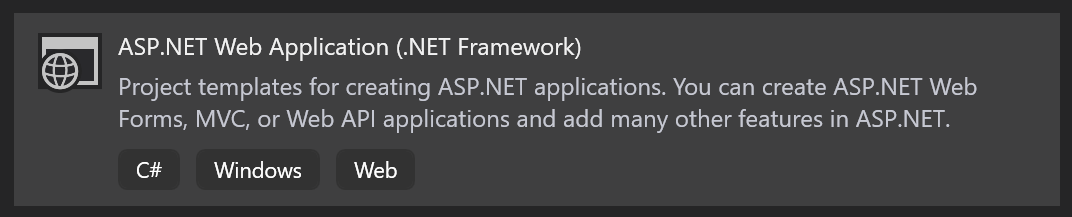


Figure 2: ASP.NET Web Application template

Configure the project details (Figure 3) to specify the project name, location where the project files will be created, and Solution name.

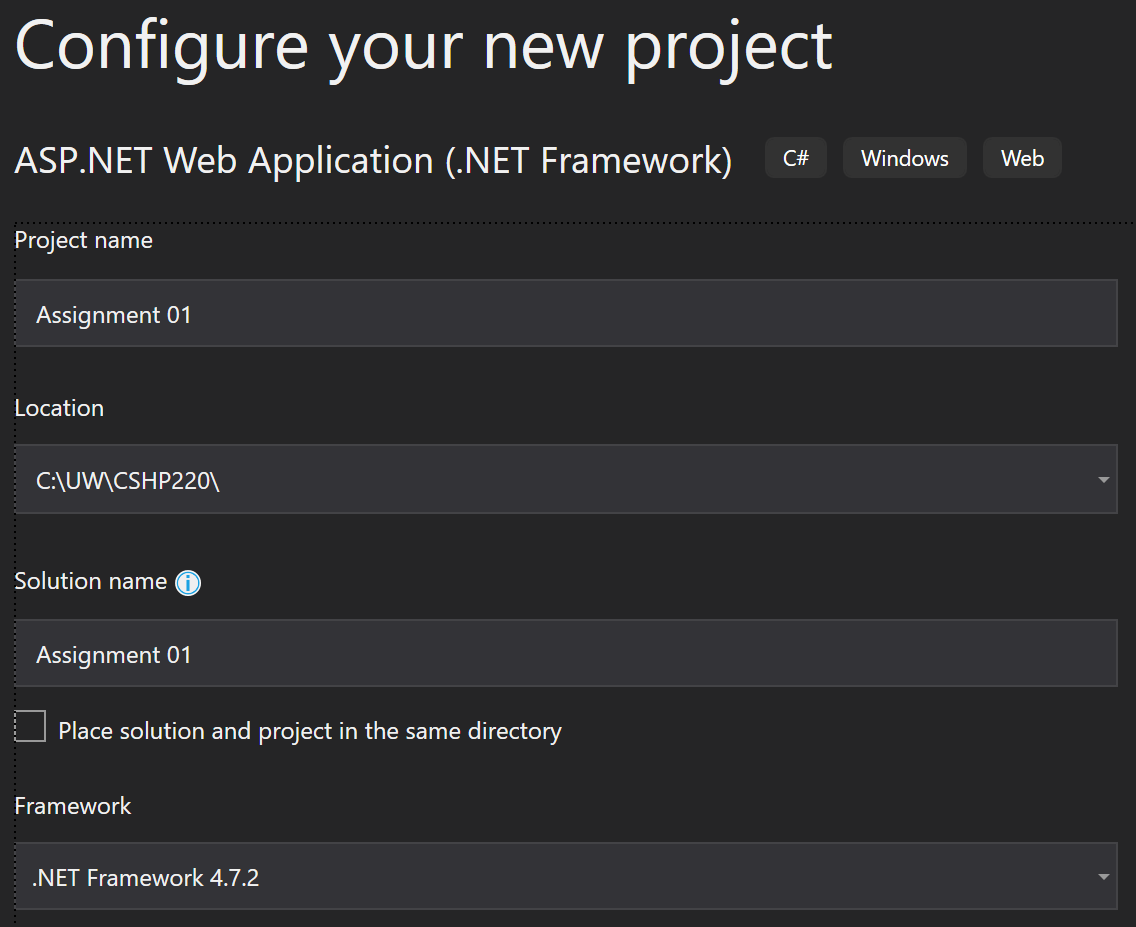


Figure 3: Configure your new project

When you click the “Create” button another screen (Figure 4) provides various ‘Boiler plate’ templates for setting up common project types. For this project, I will start with an ‘Empty’ project that does not include any folder structure or core references.

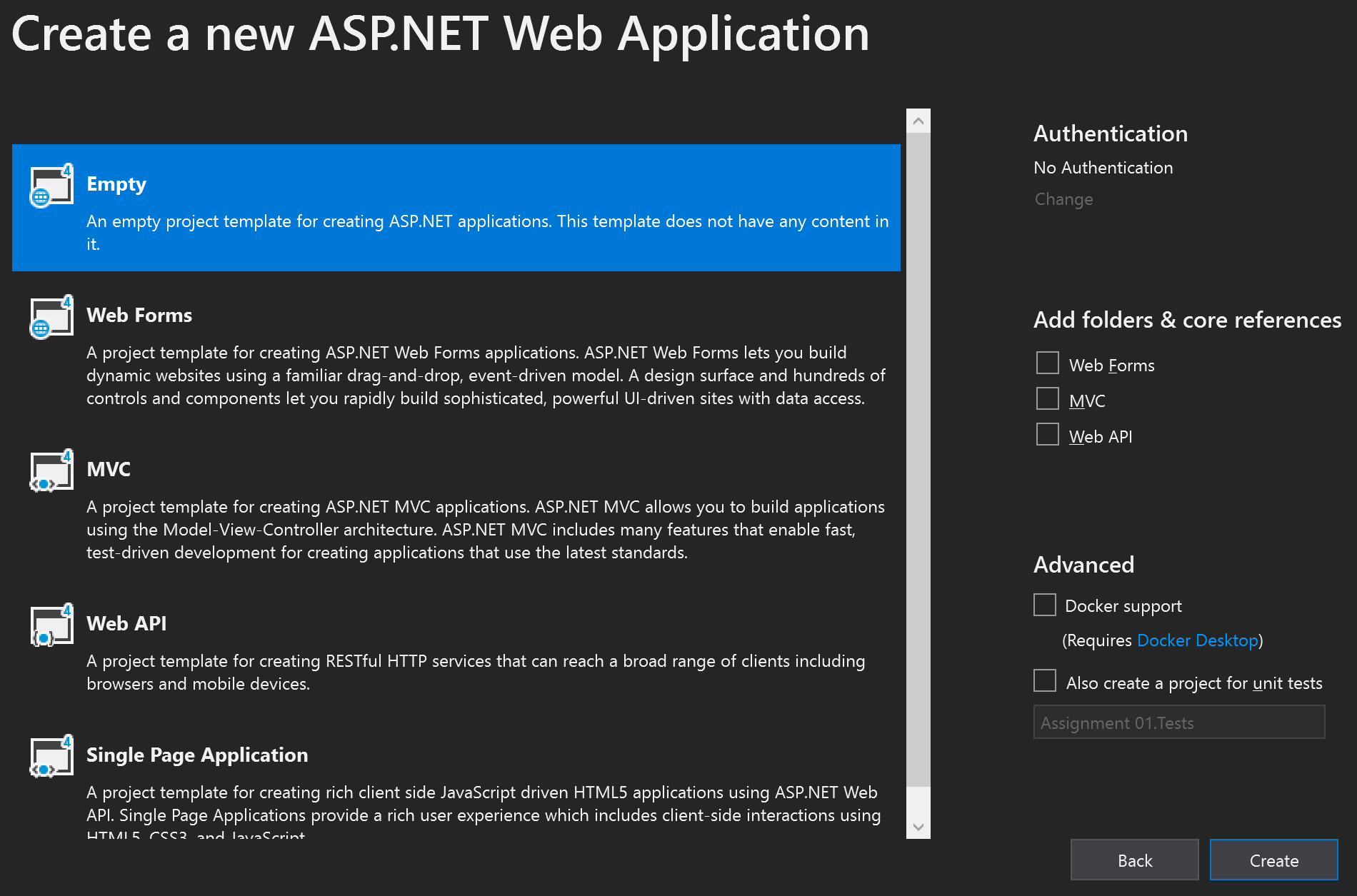


Figure 4: Boiler Plate selection for new web application projects

After clicking the Create button, the IDE will create the project and generate the files needed along with the base references needed. If you try to launch the project, you will receive a 403 error (Figure 5) suggesting you are forbidden. This is because there is no default web page, and the default behaviour is to list the directory which is not configured.

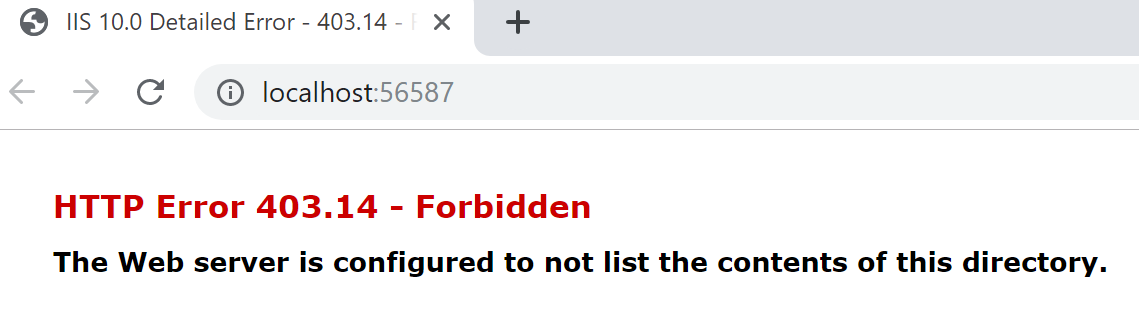


Figure 5: 403 Forbidden error message

## Creating a default web page

IIS supports various default documents which it will try loading when a page is not specified in the web request (or URL).

(MSDN 2019 <https://docs.microsoft.com/en-us/iis/configuration/system.webserver/defaultdocument/>) (External Site)

Based on the documentation, I added a new HTML page to the project (Figure 6) and named it “Default.htm” (Figure 7). The IDE helps by scaffolding the html file to include some of the various html elements. I updated it to include a Title for the page and some content to make sure it works.

1. <!DOCTYPE html>
2. <html>
3. <head>
4. <meta charset="utf-8" />
5. <title>Greeting</title>
6. </head>
7. <body>
8. <h1>Hello World</h1>
9. </body>
10. </html>

After saving, start debugging (F5) and the default browser should open displaying my very simple page (Figure 8).

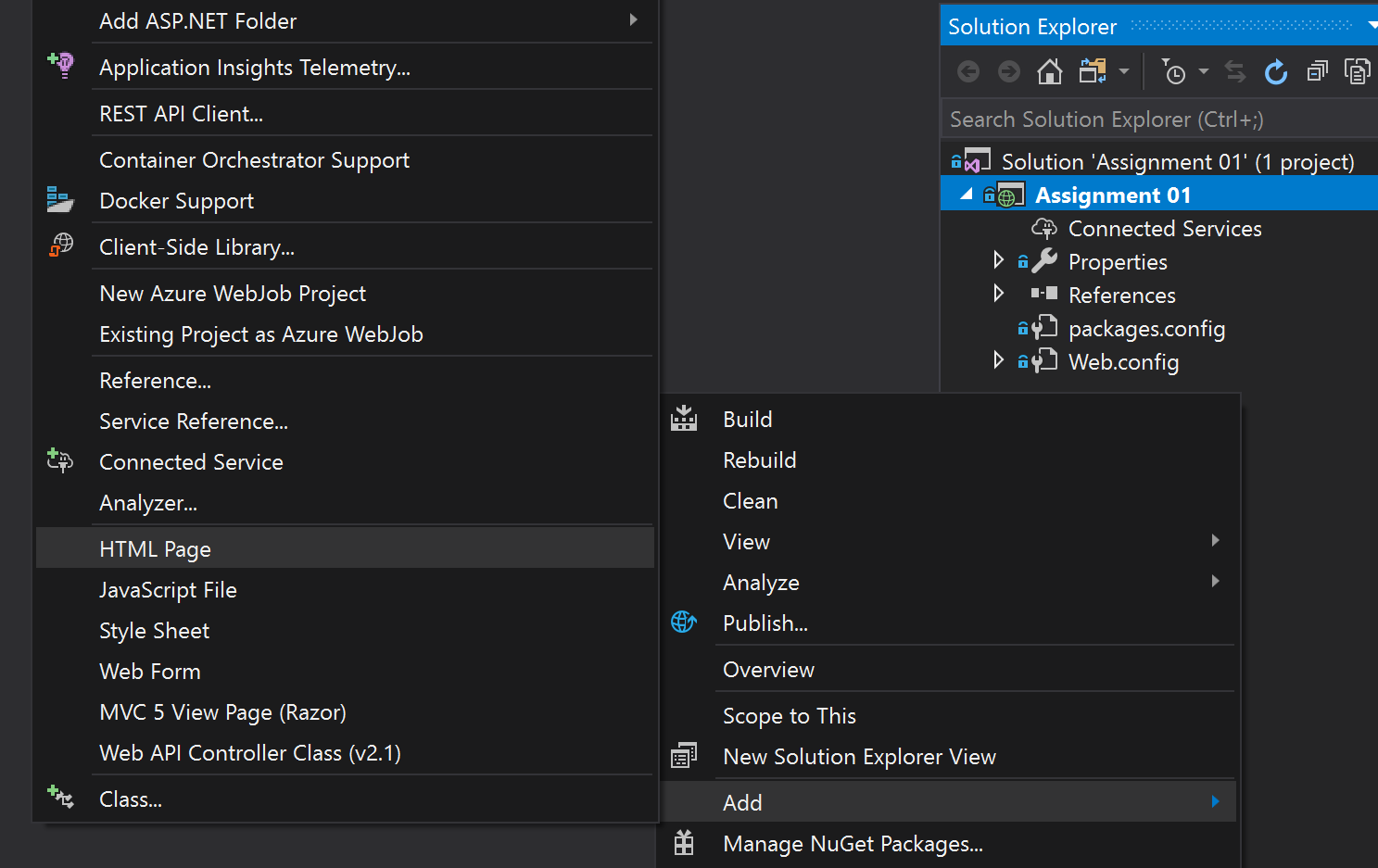


Figure 6: Adding new HTML page

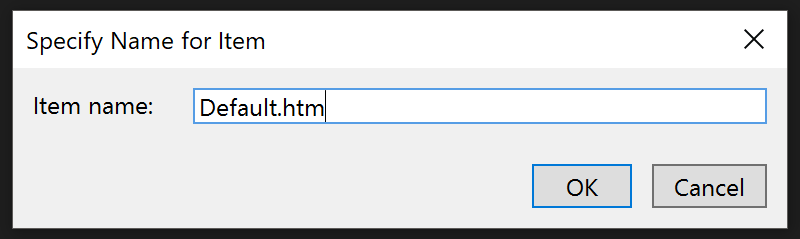


Figure 7: Default Item

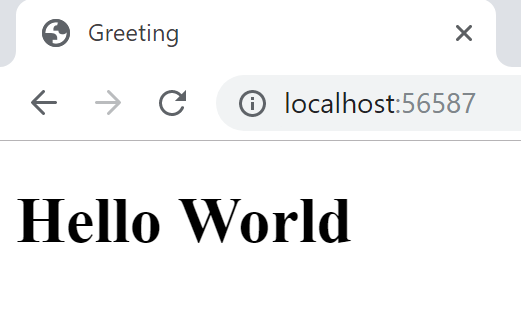


Figure 8: Default page in the browser

# Assignment demo page

## Change page extension

The default page has the extension “.htm” and is rendered client-side on the user browser. One of the goals with ASP.NET is to render content server side. To support the functionality, I have to rename the “Default.htm” file to “Default.aspx”.

## Display server Time

I want to have my page display the server time. This helps when users connect to servers located in different time zones.

To start, update the page to include a named element that will be updated by the server before sending the html to the client browser for rendering.

<body>

<h1>Hello World</h1>

<div>

<Span>Server Time: </span><asp:Label ID="serverTime" runat="server" />

</div>

</body>

Testing the change does not yet provide the server time. Some scripting is needed to run on the server to help update the page when loading the page. Adding the following in the <Head></Head> of the page will achieve this.

<title>Greeting</title>

<script runat="server">

void Page\_Load()

{

serverTime.Text = DateTime.Now.ToString();

}

</script>

Running the project now resulted in a compilation error (Figure 9). This is because the server needs to know which language the server side is using. So the following needs to be added at the top of the document.

<%@ Page Language="C#" %>

<!DOCTYPE html>

<html>

The page should render and provide the server time as expected (Figure 10).

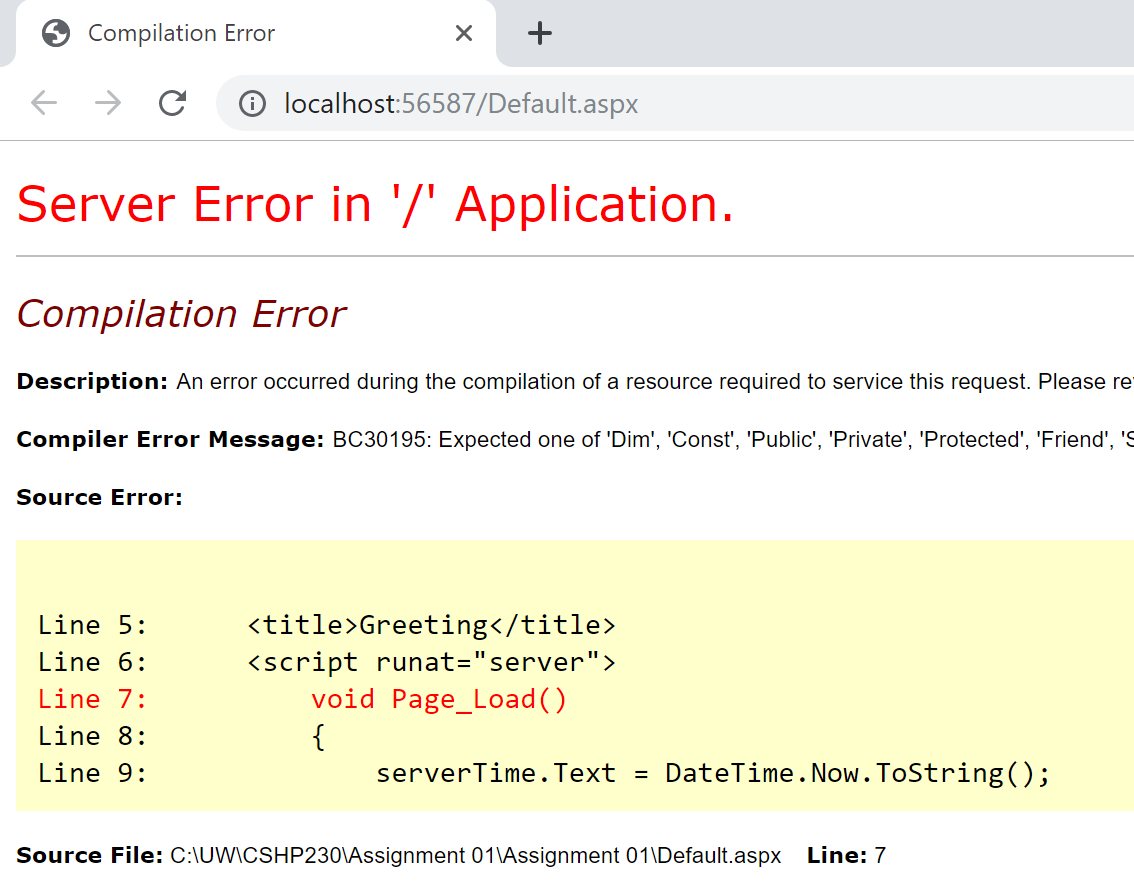


Figure 9: Compilation Error

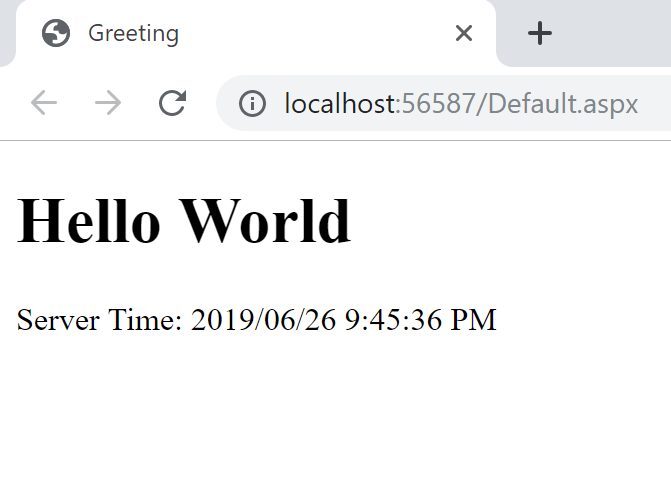


Figure 10: Scripted server time

## Client-Side Scripting

ASP.NET pages still support client side scripting through the use of JavaScript. I updated the body of the page to include the following below the div for the server time.

<br>

<div id="askName">

<form>

<span>What is your name? </span><input type="text" name="txtUserName" />

<input type="button" value="Submit" onClick="greet(this.form)" />

</form>

</div>

<div id="greetName" style="display: none">

<span>Hello, </span><span id="displayName"></span>

</div>

If I try debug the page and try to click the submit button the following error can be found in the debug tools of the browser window (Figure 11). I am missing the supporting JavaScript that will be called when clicking the submit button.

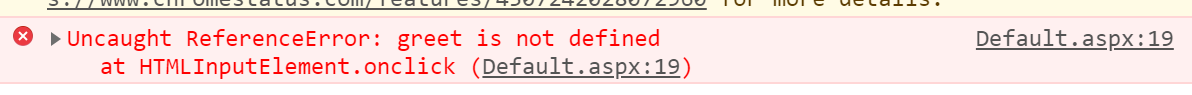


Figure 11: Undefined function

To make the simple form work, the following script is added below the other script in the head of the document. This script takes the input value and updates the DOM in a separate element while hiding the input form in the process.

<script type="text/javascript">

function greet(objForm)

{

var userName = objForm.txtUserName.value;

var target = document.getElementById("displayName");

target.innerText = userName;

document.getElementById("askName").style.display = "none";

document.getElementById("greetName").style.display = "block";

}

</script>

# Summary

The page is simple but demonstrates the basic concepts of using ASP.NET server pages with a mixture of server-side and client-side scripting to make a dynamic web page. This is achieved with some embedded C# code to handle the server-side scripting and JavaScript for the client-side scripting.