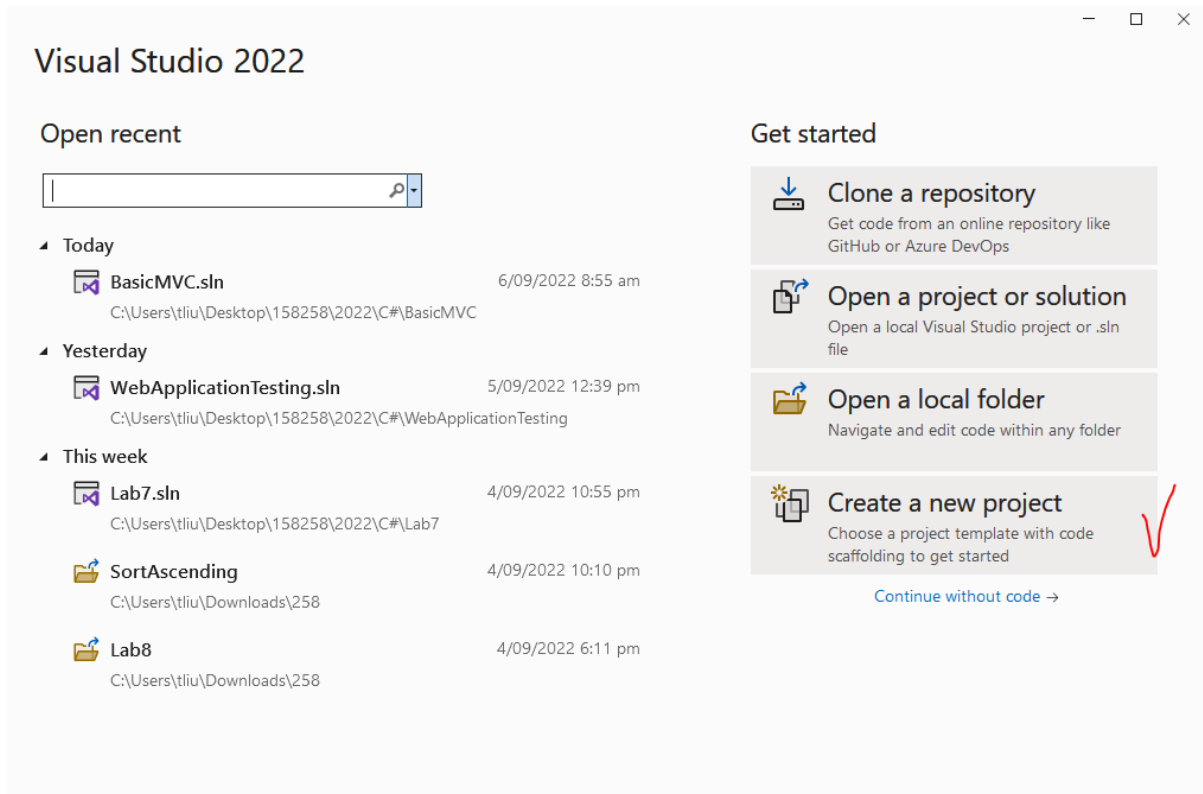
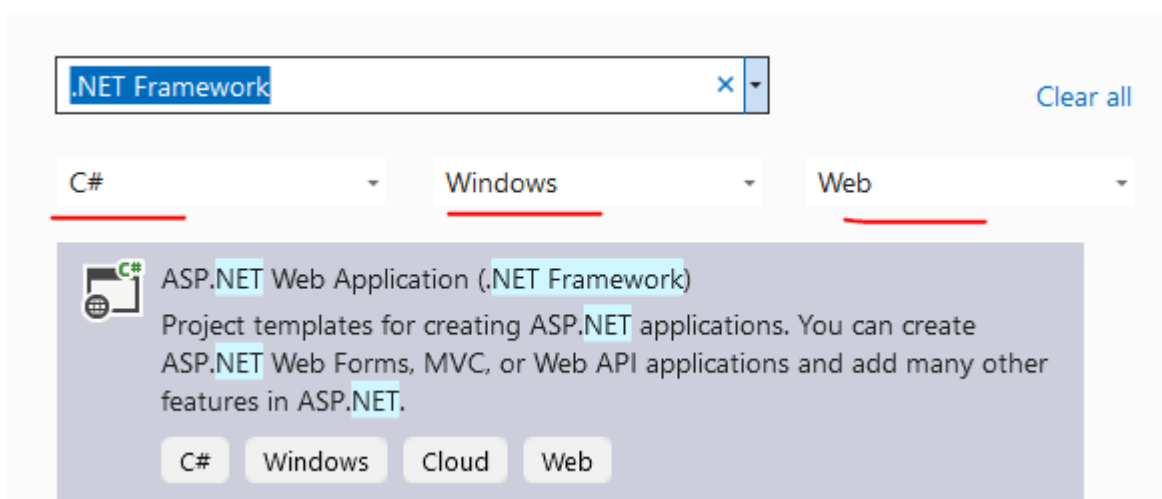


Creating a basic MVC Website

Start Visual Studio 2022, Select **Create a new project**



select your language, platform and project type. I selected **C#, Windows, and Web**, enter **.NET Framework**, select **ASP.NET Web Application (.NET Framework)**, Click **Next**



Note: If you cannot see **ASP.NET Web Application (.NETFramework)**, just scroll down and click **install more tools and features**, make sure that **ASP.NET** and **web development** and **.NET Framework project and item templates** are selected. Then click **Modify** to install the selected tools.

Name your Project and choose the location of your project and Click **Create** to continue.

Configure your new project

ASP.NET Web Application (.NET Framework) C# Windows Cloud Web

Project name
BasicMVCProject

Location
C:\Users\tliu\Desktop\158258\2022\C#

Solution name ⓘ
BasicMVCProject


☐ Place solution and project in the same directory


Framework
.NET Framework 4.7.2


Back Create


select **MVC**, Click **Create** to create the project.


Create a new ASP.NET Web Application

**Empty**
An empty project template for creating ASP.NET applications. This template does not have any content in it.

**Web Forms**
A project template for creating ASP.NET Web Forms applications. ASP.NET Web Forms lets you build dynamic websites using a familiar drag-and-drop, event-driven model. A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access.

**MVC**
A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.

**Web API**
A project template for creating RESTful HTTP services that can reach a broad range of clients including browsers and mobile devices.

**Single Page Application**
A project template for creating rich client side JavaScript driven HTML5 applications using ASP.NET Web API. Single Page Applications provide a rich user experience which includes client-side

Authentication

None

Add folders & core references

☐ Web Forms

☒ MVC

☐ Web API

Advanced

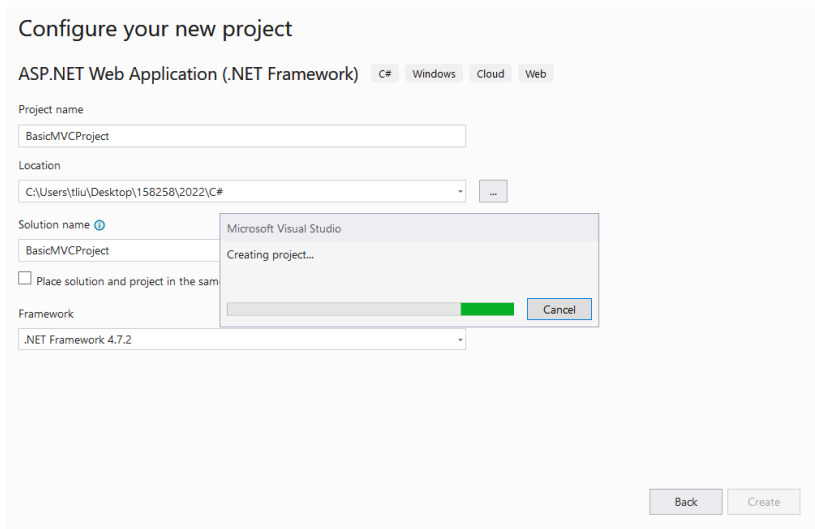
☒ Configure for HTTPS

☐ Docker support
(Requires [Docker Desktop](#))

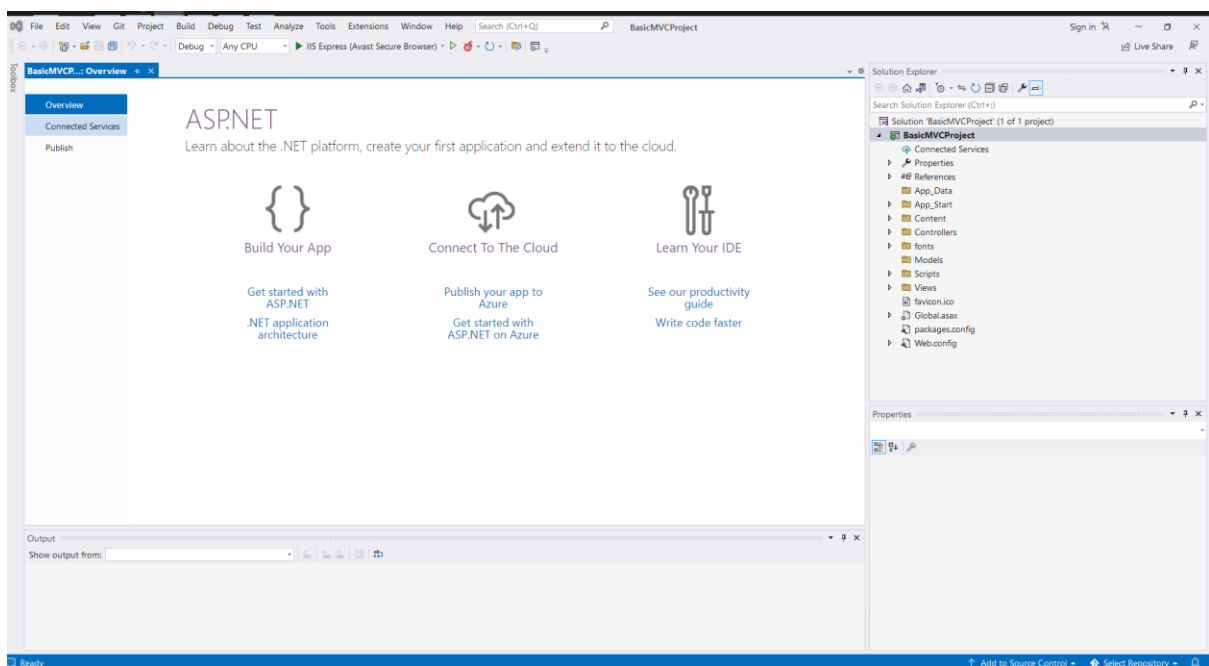
☐ Also create a project for unit tests

BasicMVCProject.Tests

Back Create



Once the project is created you should see a page like this:



Debug-> Start Without Debugging (Ctrl+F5) or click the green play arrow on the menu bar to test/view your website.

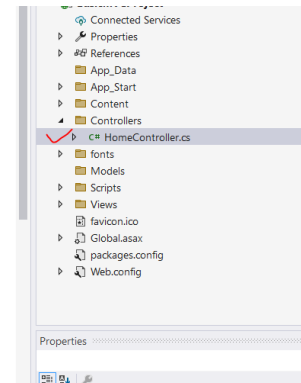
Alter the about method of the Controllers\HomeController.cs file:

```
using System.Web.Mvc;

namespace BasicMVCProject.Controllers
{
    0 references
    public class HomeController : Controller
    {
        0 references
        public ActionResult Index()
        {
            return View();
        }

        0 references
        public ActionResult About()
        {
            ViewBag.Message = "Your application description page.";
            return View();
        }

        0 references
        public ActionResult Contact()
        {
            ViewBag.Message = "Your contact page.";
            return View();
        }
    }
}
```



```
public ActionResult About(string id)

{

    ViewBag.Message = "Your application description page. You entered the ID " + id;

    return View();

}
```

```
using System.Web.Mvc;

namespace BasicMVCProject.Controllers
{
    0 references
    public class HomeController : Controller
    {
        0 references
        public ActionResult Index()
        {
            return View();
        }

        0 references
        public ActionResult About(string id)
        {
            ViewBag.Message = "Your application description page. You entered the ID " + id;
            return View();
        }

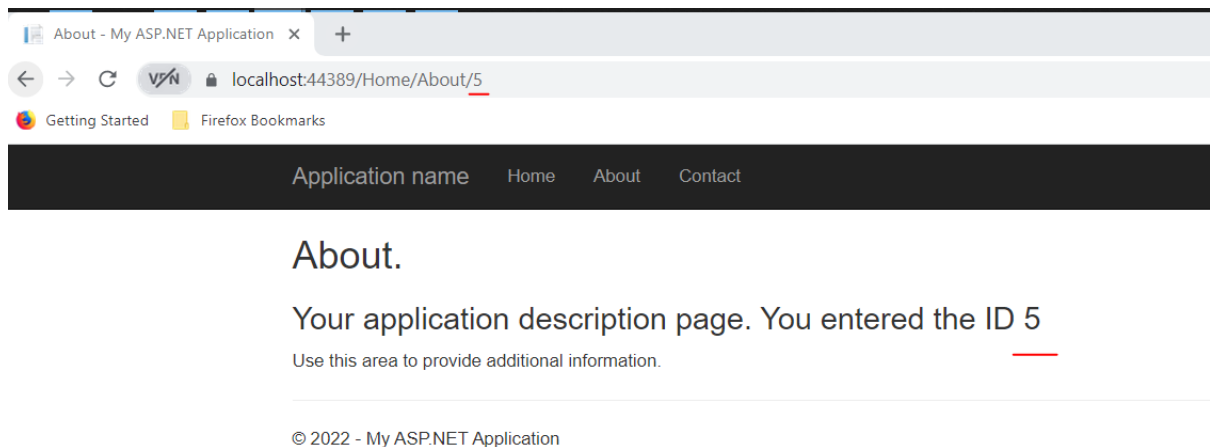
        0 references
        public ActionResult Contact()
        {
            ViewBag.Message = "Your contact page.";
            return View();
        }
    }
}
```

Here we are telling the method to process a parameter passed into it called id. This corresponds with the name of optional third parameter in the default route. The ASP.NET MVC model binding

system is used to automatically map the ID parameter from the URL into the id parameter in the About method.

Now start the web site without debugging and navigate to the About page. Now add an ID onto the end of the URL such as <https://localhost:44389/Home/About/5> (Note: For your localhost: 44389/ may be different). This will then process 5 as the value of the ID parameter and pass this to the About method, which then adds this to the ViewBag for display in the view.

The output would be something like:



ASP.NET MVC also automatically matches any parameters in the HTTP request to a method parameter if they have the same name. The matching is case insensitive. For example, entering <http://localhost:44389/Home/About?ID=5> will return the same result as previous, because the id parameter is automatically mapped to the id parameter in the About method.

Email link

Add Email link on the _Layout.cshtml.

```
<footer>
    <address>
        Written by <a href="mailto:webmaster@massey.ac.nz">Tong Liu</a>.<br>
        Visit us at: Massey.ac.nz<br>
        New Zealand<br>
    </address>
</footer>
```

JavaScript:

Add MyJavaScripts.js in the Scripts folder with the following code”

```
function msg(){
    alert("Hello World!");
}
```

Add MyJavaScripts.js in the BundleConfig.cs file:

```
bundles.Add(new ScriptBundle("~/bundles/bootstrap").Include(
    "~/Scripts/bootstrap.js", "~/Scripts/MyJavaScript.js"));
```

Add the following code in Home/Index.cshtml:

```
<div>
    <form>
        <input type="button" value="click" onclick="msg()" />
    </form>
</div>
```

CSS:

Add the following code in Site.css

```
.navbar {
    background-color: deepskyblue;
    border-radius: 5px;
    padding: 0px 5px;
}
```

Add Site.css in the BundleConfig.cs file

```
bundles.Add(new StyleBundle("~/Content/css").Include(
    "~/Content/bootstrap.css",
    "~/Content/Site.css"));
```

Image:

Create an Images folder in Content folder, load images.

Add the following code with your own image name in Home/Index.cshtml:

```
<div>
    <p></p>
</div>
```

Video:

Add the following code in Web.config file

```
<system.webServer>
    <staticContent>
        <mimeMap fileExtension=".mp4" mimeType="video/mp4" />
    </staticContent>
</system.webServer>
```

Create a video folder in Content folder, load your video.

Add the following code with your own video name in Home/Index.cshtml:

```
<div>
  <video width="320" height="240" controls="controls" autoplay="autoplay">
    <source src="~/Content/Video/Demo.mp4" type="video/mp4" />
    Your browser does not support the video tag.
  </video>
</div>
```