

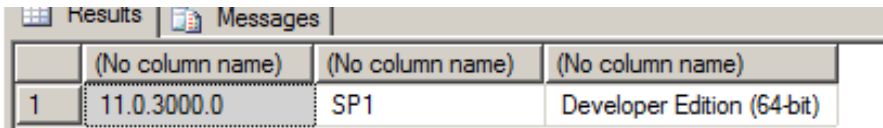
# INFO3067 Week 1 Class 1

## Introduction

- Welcome
- Course Outline
- Email - elauersen@fanshaweonline.ca
- Expectations
  - Attendance
  - Typical Daily 3 hr agenda
    - 1<sup>st</sup> hour will be a work hour with Q and A for previous work
    - 2<sup>nd</sup> hour will be the lecture and checkpoint requirements new work
    - 3<sup>rd</sup> hour will be a work hour and Q and A available any work
    - Class 1 work due Fridays @ 1:00 pm
    - Class 2 work due Mondays @ 1:00 pm

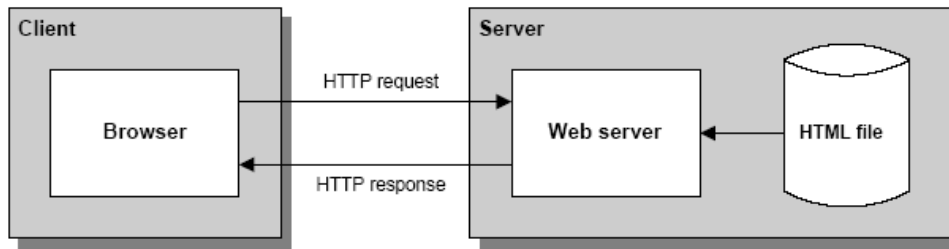
## Course Software

- Visual Studio 2013 + Update 1
  - C#.Net will be used for all server side development
  - JQuery (Javascript) will be used for all client side scripting
- The course was developed on SQL Server 2012 Developer Edition
  - Check version of SQL Server with this SQL:

```
SELECT SERVERPROPERTY('productversion'), SERVERPROPERTY ('productlevel'), SERVERPROPERTY ('edition')
```
  - 

	(No column name)	(No column name)	(No column name)
1	11.0.3000.0	SP1	Developer Edition (64-bit)
  -
- You will also need a html5 compliant browser, the course was developed and I am recommending to use **Chrome**:
  - <https://www.google.com/intl/en/chrome/browser/>

## What is a Web Application? – A Review



Classic web applications utilize something known as an http transaction. All HTTP transactions follow the same general format. Each client request and server response has three parts: the request or response line, a header section, and the entity body. Using the following code we can see the base structure of an html 5 form:

```
<!DOCTYPE HTML>
<html lang="en">
<head>
  <title>Simple Html Form</title>
</head>
<body>
  <form action="SimpleHtmlForm.htm" method="get">
    <b>Username:</b>
    <br />
    <input type="text" size="30" name="username" />
    <p>
      <b>Comments:</b>
      <br />
    </p>
    <textarea name="comments" rows="10" cols="60"></textarea>
    <p>
      <input type="submit" value="Add Comment" />
    </p>
  </form>
</body>
</html>
```

Open notepad or another editor and paste this code in, save the file as **SimpleHtmlForm.htm**. The important line in this code is the <form.. line it has two important attributes. The “**action**” attribute informs the browser which page to execute on the form’s submission. The “**method**” attribute informs the browser how to pass the state (properties) of the form. In this case we are using “**get**”, which is the traditional way to pass state using the **query string**. A query string is designated with a “?” in the address bar. Fill in the name and comments areas and then click the button. You’ll see the state is passed in the url now for instance my url reads:

<file:///C:/Evan.../SimpleHtmlForm.htm?username=Evan&comments=some+comment>

We could just as easily use **post**, except this time we would not see the state passed in the address bar, it would be sent as part of the http transmission in what's called the message body. W3 Consortium says:

*“use GET if processing of a form is idempotent (i.e. it has no lasting observable effect on the state of the world), eg. many database searches have no visible side-effects and make ideal applications of query forms. If the service associated with the processing of a form has side effects (for example, modification of a database or subscription to a service), the method should be **POST**.”*

## ASP.Net Web Forms vs ASP.Net MVC vs Web API

Microsoft has 3 ASP based frameworks for developing web apps. Unfortunately we do not have time to learn all of these strategies. This course traditionally used the **Web Forms** approach as the MVC/Web API approaches were not mature enough for writing complicated web apps in a timely fashion. Asp.Net web forms work a lot like the WPF forms we used in INFO3070. With VS2012/VS2013 Microsoft has done a lot of good things to beef up the MVC offering so subsequently the course has been re-written to **focus on the MVC** pattern. The 3<sup>rd</sup> offering called SPA – Single Page Applications using Web API, is also an option. We'll use a few of the technologies in the case study this summer, and you'll get a comprehensive look at these types of apps in the 6<sup>th</sup> semester client server course. All of the technologies can be found on Microsoft's [www.asp.net](http://www.asp.net) site.

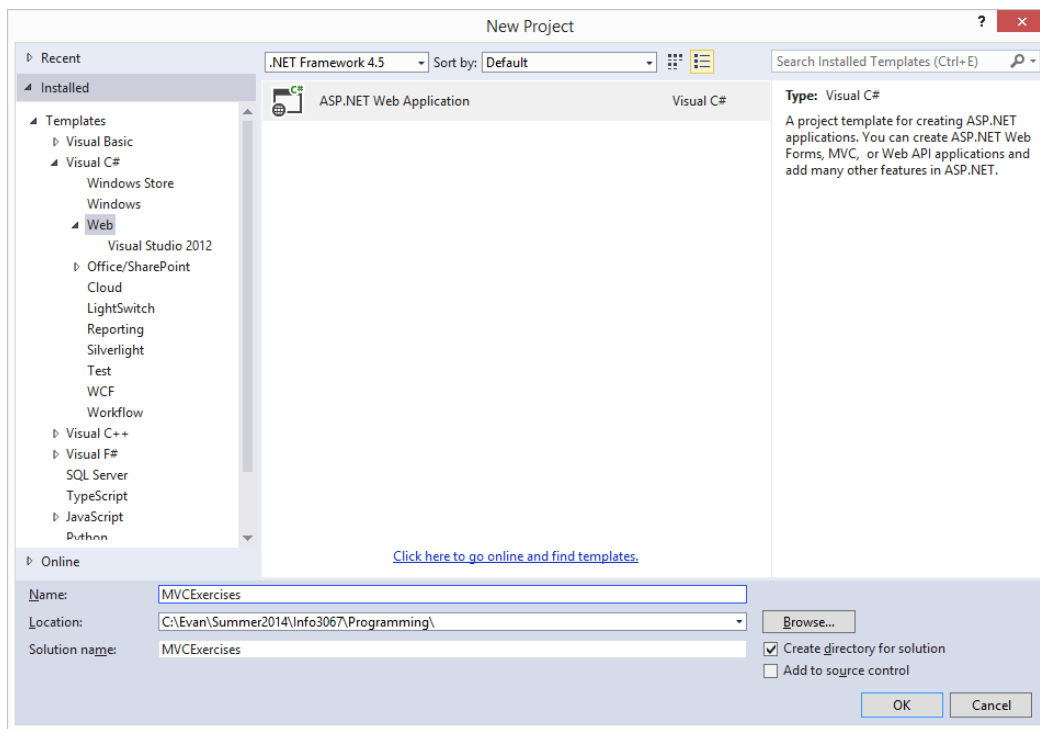
## Learning MVC Step by Step

Initially we'll be doing a temporary website to focus on one or two particular aspects of MVC, there is a larger learning curve with the MVC approach but it has a lot of attractive features. The knowledge we get from this temporary site will be used to build the case study. The format for the case study will take a familiar pattern, here's what the first one will look like:

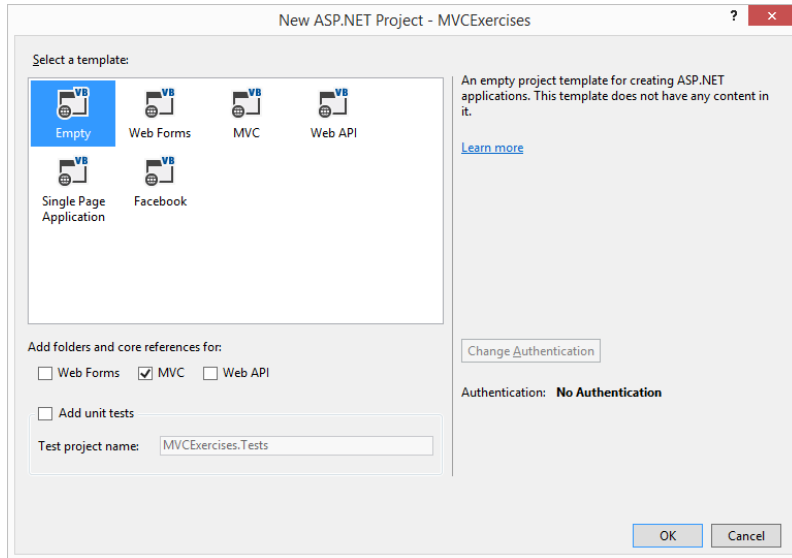
- **eStoreCase1** - solution
  - **eStoreWebsite**(ASP.Net 4.5 MVC website)
    - Controllers
    - Views
    - JQuery
  - **eStoreViewModels** (C# class library)
    - CustomerViewModel.cs
    - eStoreViewModelConfig.cs –
    - References eStoreModels C# class library project
  - **eStoreModels** (C# class library)
    - CustomerModel.cs
    - eStoreModelConfig.cs
    - Contains Entity Framework classes from the eStoreDb database

## MVCExercises – Web Site

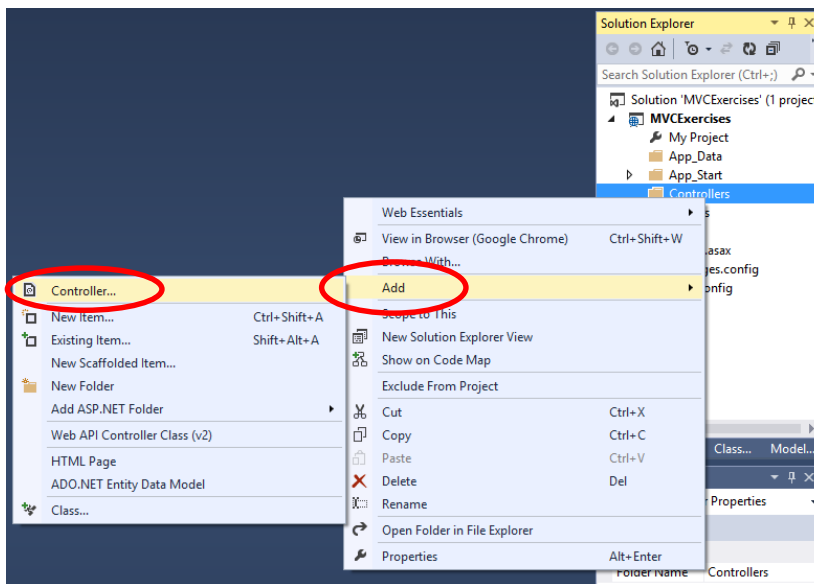
- Create a folder called MVCExercises
- Start VS 2013 and create a C# based ASP.Net Web application (MVCExercises)

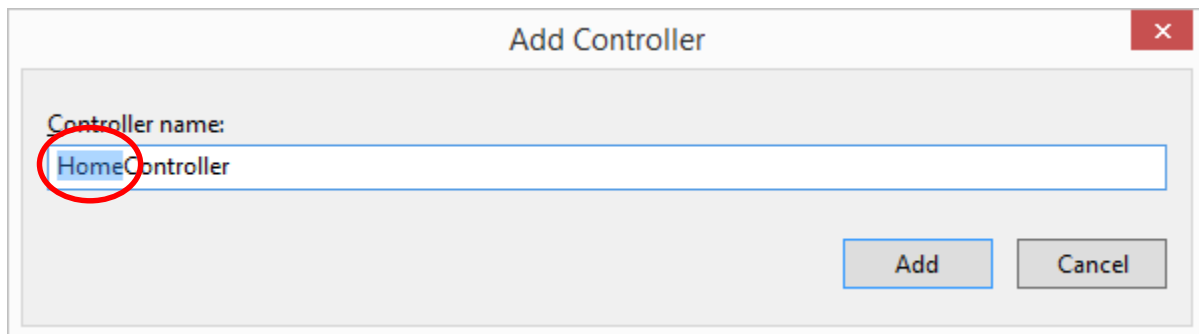
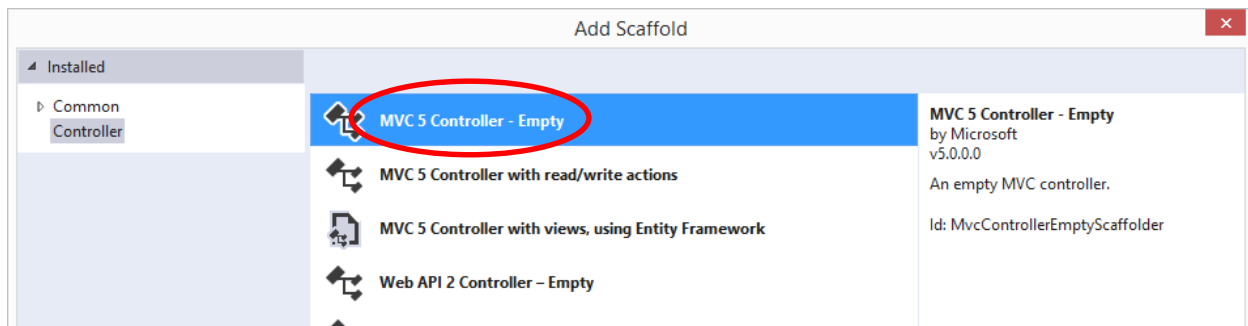


- Choose the “Empty” template, and check off the “MVC” folders and reference

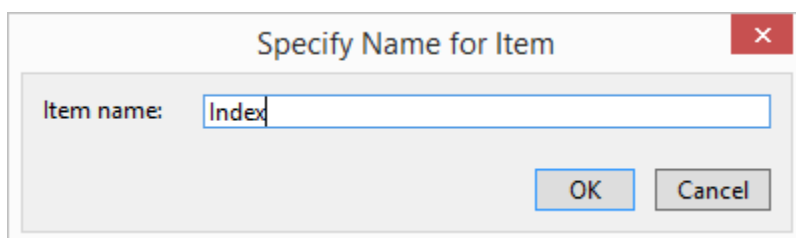
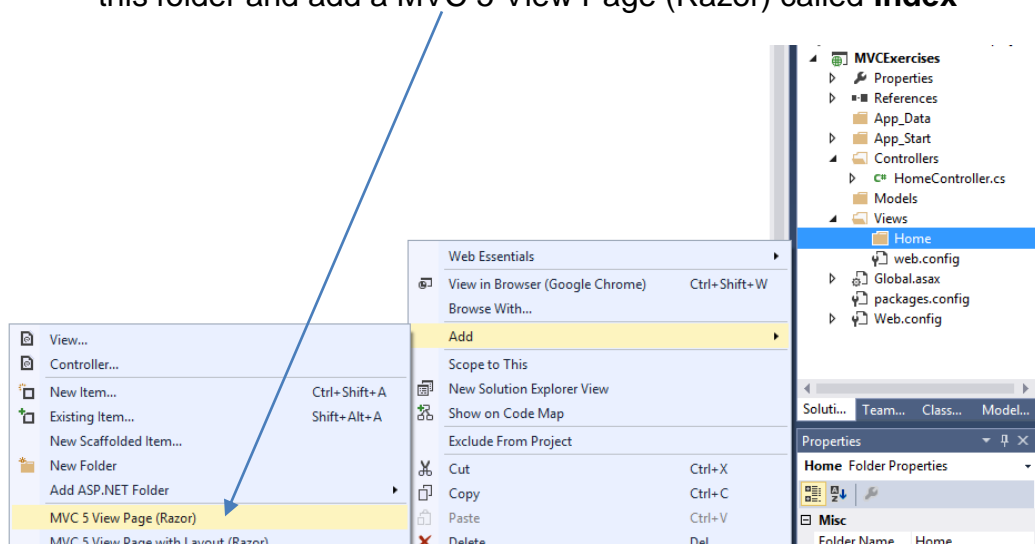


- This will give you basically an empty web site with a number of folders for you to lay out an MVC application. There are some nuances which we will start to figure out now.
- Add an empty “**MVC 5 Controller**” called **HomeController** to the Controller’s folder (right-click Controllers):





- This should create a folder called Home in the Views folder, locate and right click this folder and add a MVC 5 View Page (Razor) called **Index**



- Then add a line of code to the generated markup:

```

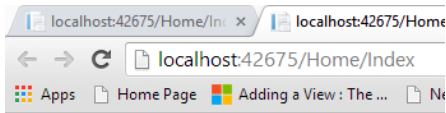
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title></title>
</head>
<body>
    <div>
        <h1>Welcome to INFO367!</h1>
    </div>
</body>
</html>

```

- Press F5 to run the application:



**Welcome to INFO367!**

- The MVC engine by default will look for a controller called **Home** to be its starting point and execute a method called **Index**. Notice the URL in the browser isn't exposing any name or page or anything. You can change this default starting point if you wanted by modifying a file called **Route.config** in the **App\_Start** folder:



- Controllers typically perform a block of logic called an **Action**. The action in turn, has an outcome that is returned in the form of what is called an **ActionResult**. The main type of ActionResult we'll be using is called a **View**:
  - Controller Method(logic) → ActionResult(View)

- Change the Index method by adding the following code to the Index method:

```
namespace MVCExercises.Controllers
{
    // References
    public class HomeController : Controller
    {
        // GET: /Home/
        // References
        public ActionResult Index()
        {
            ViewBag.Message = "Welcome Again to Info3067";
            return View();
        }
    }
}
```

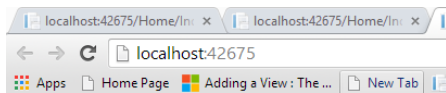
- Return to the Views\Home folder and edit the Index.cshtml file again, and change it to the following:

```
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title></title>
</head>
<body>
    <div>
        <h1>@ViewBag.Message</h1>
    </div>
</body>
</html>
```

- Executing (F5) this we get our output again:



## Welcome Again to Info3067

Some notes about this format:

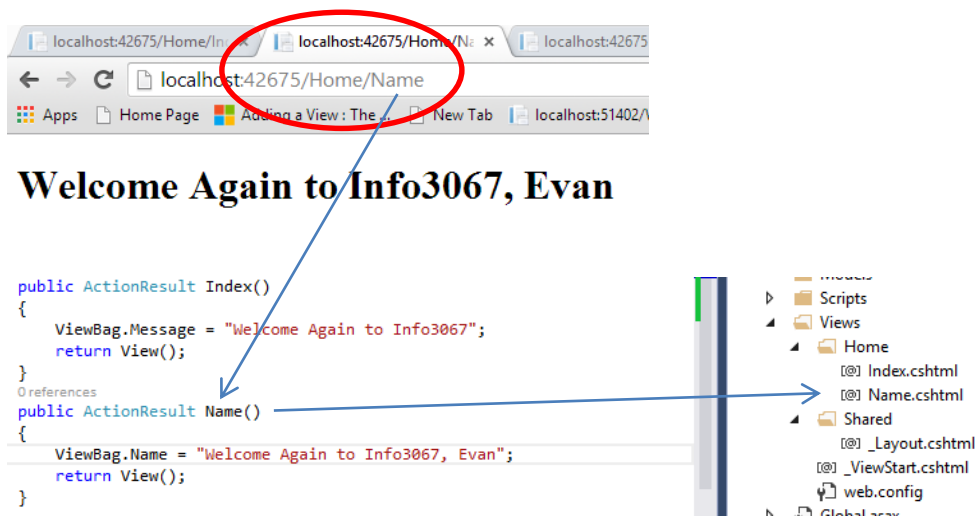
- We added 1 line of code to the Index method using an object called **ViewBag**. A ViewBag is an instance of a dictionary class called **ViewData**. It has the equivalent of doing this:  

```
ViewData["Message"] equivalent to ViewBag.Message
```
- As mentioned earlier, the return type for a controller is typically an ActionResult which displays a **View**. In our case we returned a View called **Index**. So, given a controller name (in our case **HomeController**) the MVC engine will automatically look in the Views folder for a folder with the same name (before the Controller



part), this is why we added the Home subfolder. To test your understanding. Try this:

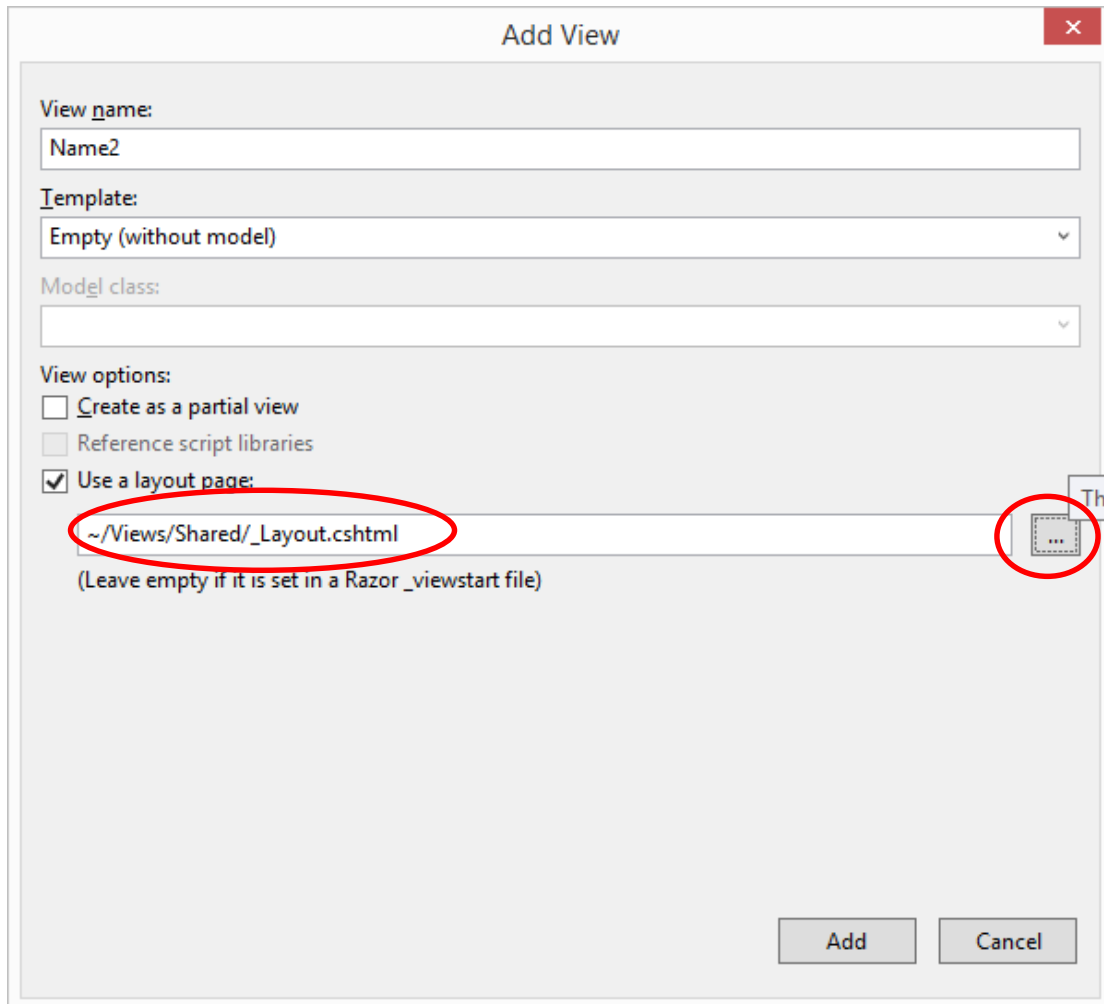
- Add a new action method to the HomeController called **Name** and add your own name to the ViewBag instance's **Name** property, that contains a string "Welcome Again to INFO3067, yourname"
- Create a new View called **Name** in the Home folder to display ViewBag.Name
- Press the Play button (F5) to execute the website in VS2013 and after the Home page appears, invoke the new Action/View with the following URL: localhost:####/Home/Name (#### is the port no. the IIS express Web Server is running on, the example below is 42675 but yours will be something else)



## Using a Master Style

- Every time we add a View we'll get an entire new page. Typically we'll want our Views to have the same look and style. So instead of repeating the entire html page we'll employ a master style. If you look in the Views folder there is a sub folder called **Shared**. Inside this folder is a master style page called **\_Layout.cshtml**.
- Let's use this master style by re-working the Name View. Add a new Action Method to your Home controller called **Name2** and copy the logic from the Name method into it, make a small change to the string so you know you're hitting this new method and create a new property for ViewBag called **Name2**

- In your Views/Home folder create a new View called **Name2**, this time however click the check box labeled **Use a layout or master page**.



View name:  
Name2

Template:  
Empty (without model)

Model class:

View options:

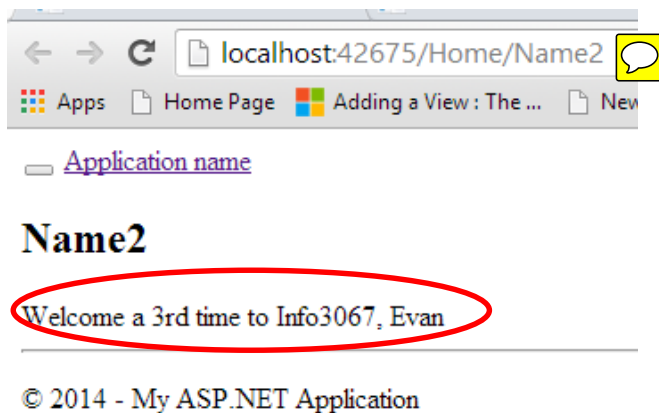
- ☐ Create as a partial view
- ☐ Reference script libraries
- ☒ Use a layout page:

~/Views/Shared/\_Layout.cshtml

(Leave empty if it is set in a Razor \_viewstart file)

Add Cancel

- Add your @ViewBag.Name2 line to this view:
- and test in the browser by pointing to ...Home/Name2



- If we look at the page source for this rendered page we see that we have a full blown HTML 5 page even though our view had none of the starting or ending tags:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="utf-8" />
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Name2 - My ASP.NET Application</title>
7     <link href="/Content/Site.css" rel="stylesheet" type="text/css" />
8     <link href="/Content/bootstrap.min.css" rel="stylesheet" type="text/css" />
9     <script src="/Scripts/modernizr-2.6.2.js"></script>
10 </head>
11 <body>
12     <div class="navbar navbar-inverse navbar-fixed-top">
13         <div class="container">
14             <div class="navbar-header">
15                 <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
16                     <span class="icon-bar"></span>
17                     <span class="icon-bar"></span>
18                     <span class="icon-bar"></span>
19                 </button>
20                 <a class="navbar-brand" href="/">Application name</a>
21             </div>
22             <div class="navbar-collapse collapse">
23                 <ul class="nav navbar-nav">
24                 </ul>
25             </div>
26         </div>
27     </div>
28
29     <div class="container body-content">
30
31
32
33 <h2>Name2</h2>
34 Welcome a 3rd time to Info3067, Evan
35
36
37     <hr />
38     <footer>
39         <p>&copy; 2014 - My ASP.NET Application</p>
40     </footer>
41 </div>
42
43 <script src="/Scripts/jquery-1.10.2.min.js"></script>
44 <script src="/Scripts/bootstrap.min.js"></script>
45
46 <!-- Visual Studio Browser Link -->
47 <script type="application/json" id="__browserLink_initializationData">
48     {"appName":"Chrome","requestId":"a54fcded018b40ceab49004ecdea615d"}
```

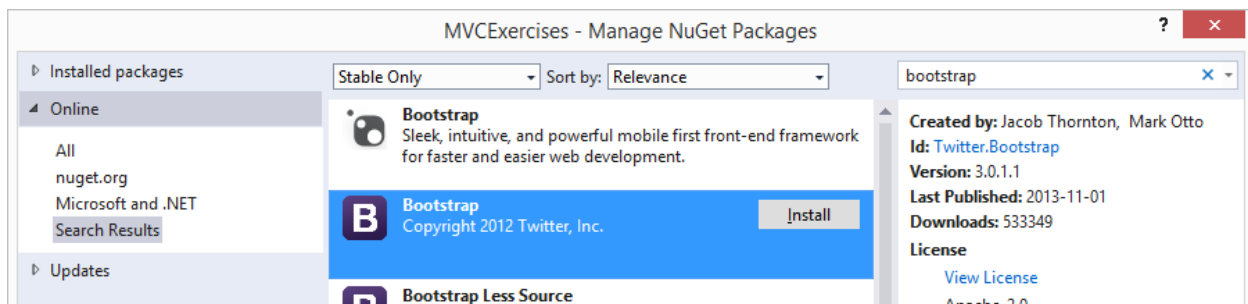
## Adding Twitter's Bootstrap to our Site

Instead of coming up with an entire new set of CSS we can use a popular template called **Bootstrap**. Bootstrap is provided to the developer community by Twitter. Wikipedia defines Bootstrap as:

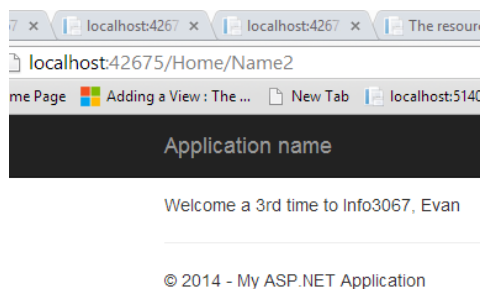
***Twitter Bootstrap** is a [free](#) collection of tools for creating websites and web applications. It contains [HTML](#) and [CSS](#)-based design templates for [typography](#), forms, buttons, charts, navigation and other interface components, as well as optional [JavaScript](#) extensions*

You don't even have to go out and get bootstrap, VS2013 has a utility called **Nuget** that will do it for you:

- Right click your project (stop executing first) and choose the **Manage Nuget Packages option...**
- Search online for Bootstrap and you'll get a number of different packages back:



- The package I'd like you to use today is the one by Twitter (ver 3.0.1.1 at the time of this writing). Just click the **Install** button.
- Now revisit your name 2 page and you should see it styled a little bit differently, as now the page is finding the bootstrap styling:



## LAB 1 – Home Page – 3 Partial Views using Bootstrap

- Add the following markup to the body of the master style page:

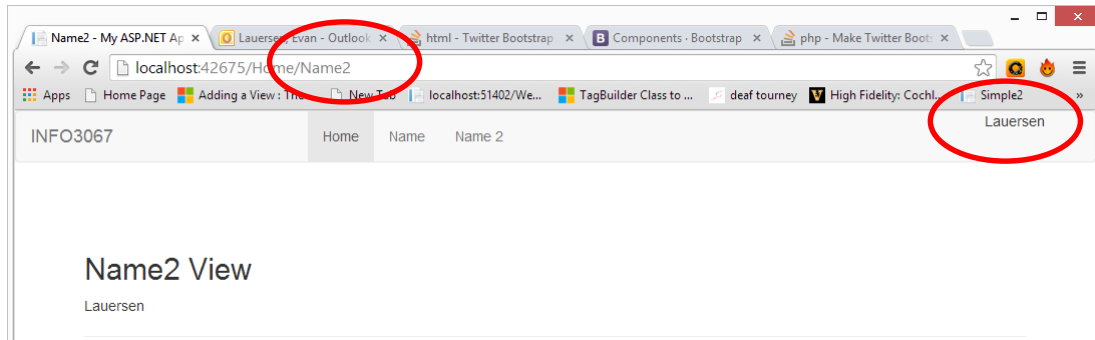
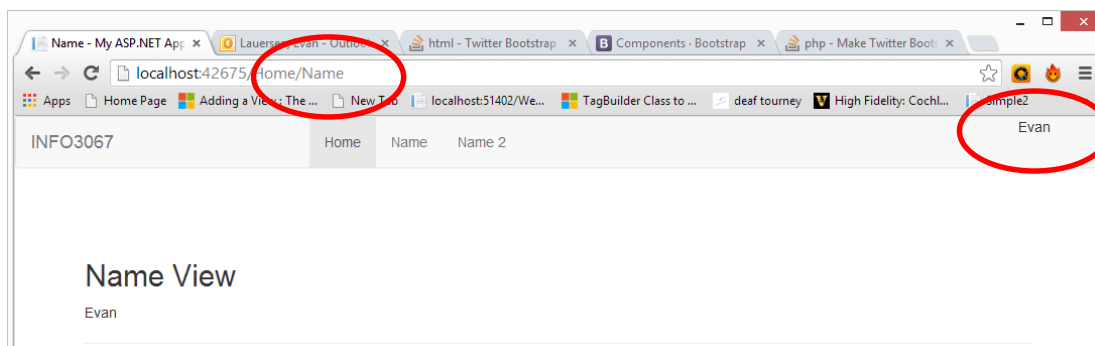
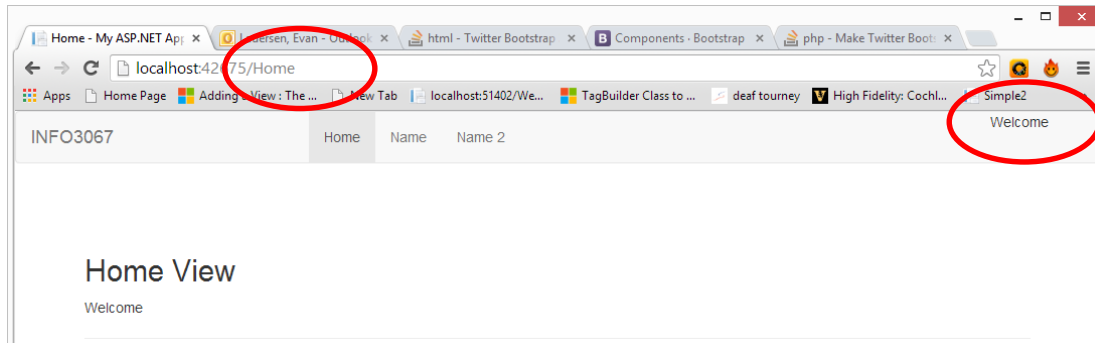
```
<body>
  <nav class="navbar navbar-default" role="navigation">
    <div class="navbar-header">
      <a class="navbar-brand" href="#">INFO3067</a>
    </div>
    <div class="col-md-2"></div>
    <ul class="nav navbar-nav">
      <li class="active"><a href="/Home">Home</a></li>
      <li><a href="/Home/Name">Name</a></li>
      <li><a href="/Home/Name2">Name 2</a></li>
    </ul>
    <ul class="nav pull-right" style="padding-right:50px;">
      <li>@ViewBag.Message</li>
    </ul>
  </nav>
  <div class="container body-content" style="padding-top:50px;">
    @RenderBody()
    <hr />
  </div>
  <script src="~/Scripts/jquery-1.10.2.min.js"></script>
  <script src="~/Scripts/bootstrap.min.js"></script>
</body>
```

- Change the Home Controller **Index** action to put “Welcome” in the ViewBag.Message property:
- Change the Home Controller **Name** action to put your first name in the ViewBag.Message property:
- Change the Home Controller **Name2** action to put your last name in the ViewBag.Message property
- Change all of your Views to utilize the master style page, for example here is the markup for the Home View:

```
@{
    ViewBag.Title = "Home";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
<h2>Home View</h2>

@ViewBag.Message
```

- Submit 3 screen shots, one for each menu option. Make sure I can see the entire URL:



## Summary of Key Terms

- Form Method – Get vs Post (idempotent criteria)
- Query String
- ASP.Net MVC vs Web Forms vs Webpages
- Home Controller
  - Index method
  - Return Type
    - ActionResult
    - String
  - ViewBag
    - ViewData
- App\_Start\Route.config
- View
  - Index method
- /Views/Shared/\_Layout.cshtml
- Bootstrap
- Nuget