**Transcript - English http://s.pluralsight.com/mn/img/sh/down-arrow-green-v1.gif**

* Introduction to ASP.NET MVC 4
  + [Introduction](javascript:void(0))

[Hi, this is Scott Allen and this is the first module in the course design](javascript:void(0)) [to demonstrate everything you need to build applications with ASP.NET MVC.](javascript:void(0)) [This course is primarily aimed at developers who are new to ASP.NET MVC,](javascript:void(0)) [but even if you've been working with the framework I hope to show you a few tricks.](javascript:void(0)) [We'll be starting at the very beginning in this first module.](javascript:void(0)) [I'll first show you how to install all the tools that you need.](javascript:void(0)) [We'll be using Visual Studio express 2012 for the web which is a free tool](javascript:void(0)) [for building web applications, but you can use other editions](javascript:void(0)) [of Visual Studio 2012 if you want to follow along.](javascript:void(0)) [By the end of this first module, you'll see how to create and run your first application](javascript:void(0)) [and also have an understanding of the design goals of the MVC framework.](javascript:void(0)) [So, let's get started.](javascript:void(0))

* + [Web Platform Installer](javascript:void(0))

[What we're looking at now is a machine with a fresh new copy of Microsoft Windows 8.](javascript:void(0)) [I don't even have any development tools installed yet.](javascript:void(0)) [So, I will go to a web browser and search for web platform installer.](javascript:void(0)) [This will work on Windows 7 also by the way.](javascript:void(0)) [If you haven't moved to Windows 8, but when I do that search I should see a link that I can click](javascript:void(0)) [on to go to a page where I can download Microsoft Web Platform Installer.](javascript:void(0)) [As this button implies this is a free download.](javascript:void(0)) [It's a tool that you can just tell it to run and it will go out and be able](javascript:void(0)) [to download software products and frameworks and libraries and install them on your computer.](javascript:void(0)) [It just takes a little bit of time to initialize, but what we're going to look](javascript:void(0)) [for is Visual Studio 2012 web edition.](javascript:void(0)) [When I install that it will also install a web server that I can use to host my web application](javascript:void(0)) [on this machine and a SQL server instance that we can use to save data for an application.](javascript:void(0)) [So, now that it is initialized, I can come into the search box that's in the top right here](javascript:void(0)) [and I will search for 2012 web and click enter.](javascript:void(0)) [The very first search result here is Visual Studio Express 2012 for Web](javascript:void(0)) [with Windows Azure SDK that is the product that I want.](javascript:void(0)) [So, I'll click add.](javascript:void(0)) [You click add.](javascript:void(0)) [You can go through and search for other products that you might want to install,](javascript:void(0)) [but you'll notice if you click on the hyperlink here.](javascript:void(0)) [This will show you everything it's going to install here](javascript:void(0)) [that includes Visual Studio 2012, some Windows Azure tools.](javascript:void(0)) [It also includes IIS 8 Express and Microsoft SQL Express LocalDB Edition.](javascript:void(0)) [We'll be talking about in using those two tools during this course and just selecting](javascript:void(0)) [that one item will give me everything that I need.](javascript:void(0)) [So, I will close this out and go ahead and tell the Web Platform Installer](javascript:void(0)) [that downloads everything I need and install this.](javascript:void(0)) [I'll need to accept some license agreements and chances are,](javascript:void(0)) [it might need to reboot the machine once or twice,](javascript:void(0)) [but we'll come back once installation is finished.](javascript:void(0))

* + [The Tools](javascript:void(0))

[Now, it's just a few minutes later and the Web Platform Installer has completed.](javascript:void(0)) [It didn't require any reboot on this machine which is always nice.](javascript:void(0)) [So, we're ready to use Visual Studio for the first time.](javascript:void(0)) [But before we do, let's just a take a step back and talk](javascript:void(0)) [about the tools we're going to use in this course.](javascript:void(0)) [What we've done so far is set up the Web Platform Installer](javascript:void(0)) [and you can launch this tool again at any time if you want to install more software.](javascript:void(0)) [But we've already installed the three essential tools that we need to move forward.](javascript:void(0)) [The first is Visual Studio.](javascript:void(0)) [Visual Studio is where we will write task and debug our code.](javascript:void(0)) [It includes editors, debuggers and shortcuts for all the technologies that we need](javascript:void(0)) [to bring together in a web application.](javascript:void(0)) [It can handle C sharp code.](javascript:void(0)) [It can also handle JavaScript HTML and CSS.](javascript:void(0)) [We've also installed SQL server.](javascript:void(0)) [Later in this course, we'll see how to persist](javascript:void(0)) [and query data that's stored in a SQL server database.](javascript:void(0)) [And finally we've installed IIS Express.](javascript:void(0)) [IIS Express is a light weight web server we can use during development](javascript:void(0)) [and it doesn't require any special privileges to run.](javascript:void(0)) [As a web server, this is the component that will respond to HTTP request that come](javascript:void(0)) [in from a web browser and it responds by executing the logic](javascript:void(0)) [that we build inside of our MVC application.](javascript:void(0)) [We won't interact with IIS Express on a regular basis.](javascript:void(0)) [It just pretty much sits behind the scenes and faithfully hosts and executes our application.](javascript:void(0)) [The first step in using all these tools is to just launch Visual Studio.](javascript:void(0))

* + [New Project](javascript:void(0))

[Once studio is installed, you should be able to find it by searching your Windows start menu.](javascript:void(0)) [And Windows 8 the start menu is an enormous screen full of squares and rectangles,](javascript:void(0)) [but I should be able to find VS Express for web.](javascript:void(0)) [I'm going to right click that and tell Windows to pin it to my task bar.](javascript:void(0)) [That makes it really easy to launch from a desktop view in the future](javascript:void(0)) [and then just click on the icon to get started.](javascript:void(0)) [Visual Studio Express doesn't want you to register and obtain a product key.](javascript:void(0)) [Again this is a free product, but you'll have to provide some basic contact information](javascript:void(0)) [to obtain the key and keep using VS Express.](javascript:void(0)) [But right now I'm going to cancel this and just move forward,](javascript:void(0)) [but you'll probably want to register right away.](javascript:void(0)) [The first time that you launch Visual Studio,](javascript:void(0)) [it might take a little extra time to perform some initialization.](javascript:void(0)) [But every start up after this will be considerably faster.](javascript:void(0)) [Now, we're ready to go.](javascript:void(0)) [Every great application that's built using Visual Studio starts by clicking](javascript:void(0)) [on the new project link or you can also go to file new project.](javascript:void(0)) [And inside of here you'll see templates create all sorts of different applications,](javascript:void(0)) [Silverlight Applications, class libraries, of course the one we're interested in,](javascript:void(0)) [is an ASP.NET MVC 4 Web application.](javascript:void(0)) [You can create this using Visual Basic or for this course we'll be using C sharp.](javascript:void(0)) [I'm going to give the project a name I'll call this OdeToFood because I want to build a website](javascript:void(0)) [where I can enter restaurant reviews and I can select any folder on my hard drive even](javascript:void(0)) [when it doesn't exist yet or I want to save this project and be able to work on it.](javascript:void(0)) [Now, I can click okay and Visual Studio will ask me what template I want](javascript:void(0)) [to use to create this project.](javascript:void(0)) [In the future, you might see additional templates in this dialog box and the idea is](javascript:void(0)) [that you'll pick the template that is the closest match to the type](javascript:void(0)) [of application you want to build.](javascript:void(0)) [The empty template is great if you know exactly what you want](javascript:void(0)) [and you want to start from scratch.](javascript:void(0)) [The internet application template is the one I'll be selecting.](javascript:void(0)) [We'll talk about some of these other templates later in the course,](javascript:void(0)) [but the internet application template comes with the default home page](javascript:void(0)) [and all of the user interface pieces and infrastructure needed for users to register](javascript:void(0)) [on the site and then log on and log off.](javascript:void(0)) [For the view engine, I'm going to leave the default value here of Razor.](javascript:void(0)) [A view engine in MVC is a component that works inside the application](javascript:void(0)) [to help you produce the HTML that you need to send back to the client's browser.](javascript:void(0)) [Razor is the preferred view engine for ASP.NET MVC, it's the one I strongly suggest you use.](javascript:void(0)) [I'll also let Visual Studio go ahead and create a unit test project for me.](javascript:void(0)) [The project will have the same name as my web project which is OdeToFood.](javascript:void(0)) [It will just have dot Tests appended at the end.](javascript:void(0)) [And with that I'll click okay and Visual Studio will go ahead](javascript:void(0)) [and start putting together my project.](javascript:void(0)) [One thing that's new with ASP.NET MVC 4 is all of the little pieces that go into this project.](javascript:void(0)) [All of the different libraries like jQuery and jQuery UI.](javascript:void(0)) [They are all added as NuGet packages.](javascript:void(0)) [We'll be talking about NuGet later in this course too.](javascript:void(0)) [But it makes all of these pieces very easy to update.](javascript:void(0)) [So, if there is an update to one of let's say the JavaScript Libraries in my project.](javascript:void(0)) [I can easily go out and install that update](javascript:void(0)) [without manually downloading and copying files around.](javascript:void(0)) [We'll see that, but for now, Visual Studio has created the project.](javascript:void(0)) [It's ready to go.](javascript:void(0)) [So, let me run the application, I can click the play button up here in the tool bar](javascript:void(0)) [to run this application inside of Internet Explorer](javascript:void(0)) [that will have Visual Studio compile everything that will launch IIS Express](javascript:void(0)) [and also Internet Explorer and there we can see the application as running.](javascript:void(0)) [That's a good sign.](javascript:void(0)) [You can see that the template we pick, the internet application template,](javascript:void(0)) [it give us a home page with some default colors and styles.](javascript:void(0)) [We have a menu with working links.](javascript:void(0)) [So, I can go to an about page.](javascript:void(0)) [I can also go to a contact page.](javascript:void(0)) [Notice the URL here is a nice friendly readable URL slash home slash contact.](javascript:void(0)) [There is no file extensions present.](javascript:void(0)) [And if we view the source code to this, we can see its nice clean looking HTML.](javascript:void(0)) [In fact, that's HTML file because we have an HTML 5 doc type and MVC 4 includes some](javascript:void(0)) [of the HTML 5 best practices, for instance, specifying the language from my mark up.](javascript:void(0)) [Also specifying the characters set encoding.](javascript:void(0)) [It turns out that it's important to have that meta tag](javascript:void(0)) [with the character set encoding specified.](javascript:void(0)) [If you want to avoid some strange and subtle cross site scripting vulnerabilities.](javascript:void(0)) [We also have a meta view port tag by default and this is important for mobile devices.](javascript:void(0)) [You can think of this as an instruction for how the page should render.](javascript:void(0)) [So, without this meta tag, a mobile browser will assume that your page needs](javascript:void(0)) [over 900 pixels to display properly.](javascript:void(0)) [So, it will zoom out to fit all of the content on the screen.](javascript:void(0)) [But because it's so zoomed out, the user cannot work with the site until they zoom in.](javascript:void(0)) [But with this tag in place, we are telling a mobile browser](javascript:void(0)) [that our site will adapt to the width of the device.](javascript:void(0)) [So, please don't assume anything about how many pixels we need.](javascript:void(0)) [This all works because of some instructions that are in the style sheet](javascript:void(0)) [that are included in this application by default.](javascript:void(0)) [I have a few other videos on Pluralsight that use MVC 4 with mobile devices and jQuery mobile,](javascript:void(0)) [if you're more interested in that topic.](javascript:void(0)) [We also have a link to Modernizer JavaScript library.](javascript:void(0)) [Modernizer will ensure our HTML 5 mark up works with older browsers that were](javascript:void(0)) [around before HTML 5 was given any thought.](javascript:void(0)) [So browsers like IE 6 which is over 10 years old now.](javascript:void(0)) [We can be sure that using the new HTML 5 elements like header and section and nav,](javascript:void(0)) [they will appear properly on that older browser.](javascript:void(0)) [Later in the course, we'll explore where all of this comes from and how to modify it.](javascript:void(0)) [For now, let's just be happy that MVC 4 is keeping up with the latest standards](javascript:void(0)) [and has given us a running web application to work with.](javascript:void(0)) [But right now, I want to close the browser and return to Visual Studio](javascript:void(0)) [to start poking around inside of this project.](javascript:void(0))

* + [Models, Views, Controllers](javascript:void(0))

[Here inside the Visual Studio you'll have a window with the title Solution Explorer.](javascript:void(0)) [It is the Solution Explorer window that allows you to get to all of your files, C sharp files,](javascript:void(0)) [JavaScript files, CSS files, icons, images and everything that goes into your application.](javascript:void(0)) [When we created this application using the internet project template,](javascript:void(0)) [Visual Studio went ahead and populated our application with all](javascript:void(0)) [of the folders and files that you see here.](javascript:void(0)) [That's why we already have a running application with contact in about links working.](javascript:void(0)) [Three of these folders have a special significance, the controllers,](javascript:void(0)) [the models and the views folders.](javascript:void(0)) [Let's talk about those names for a minute.](javascript:void(0)) [The ASP.NET MVC framework derives its name from the model view controller design pattern.](javascript:void(0)) [This design pattern has been present in software applications for several decades at this point](javascript:void(0)) [and it's a design pattern to follow when you want to separate the responsibilities](javascript:void(0)) [of the components in your user interface layer.](javascript:void(0)) [The C and MVC is for controller.](javascript:void(0)) [A controller is a software component that will be the target for some external stimulus.](javascript:void(0)) [In the case of a web application that external stimulus is usually an incoming HTTP request.](javascript:void(0)) [So, when someone launches a web browser and points it to the slash home slash](javascript:void(0)) [about location of my application.](javascript:void(0)) [That incoming request needs to go to a controller that is in my application.](javascript:void(0)) [When the controller receives the request, it's responsible for building a model that M in MVC.](javascript:void(0)) [It's the model that contains all the information that you need to present to the user](javascript:void(0)) [to satisfy that incoming request.](javascript:void(0)) [In the case, of slash home slash about, the model might just be some information](javascript:void(0)) [about the website or about the company or the people behind the website.](javascript:void(0)) [Another example would be a controller for handling request to view recipes.](javascript:void(0)) [You might have a recipe controller that builds a list of the most popular recipes.](javascript:void(0)) [That list would be the model.](javascript:void(0)) [The controller then selects a view to display the model.](javascript:void(0)) [Views in the MVC design pattern are very simple objects.](javascript:void(0)) [Think of them as templates.](javascript:void(0)) [They take pieces of data from the model and they place them into a proper location on the page.](javascript:void(0)) [If the model was a list of recipes, then the controller might select a view that will take](javascript:void(0)) [that list and display the recipes inside an HTML table.](javascript:void(0)) [That end result is that you isolate the behaviors in your UI into one](javascript:void(0)) [of these three categories, model, view or controller.](javascript:void(0)) [A view would never need to know how to call into the data access layer](javascript:void(0)) [because the model already contains all the data it needs.](javascript:void(0)) [Meanwhile, a controller would never need to know about where to place an error message](javascript:void(0)) [or how to color it because that's the responsibility of the view.](javascript:void(0)) [That isolation that you achieve with the MVC pattern makes it easy to maintain](javascript:void(0)) [and change your application moving forward because the code inside of each](javascript:void(0)) [of these pieces is very focused and easier to understand.](javascript:void(0)) [You can make a change in the controller about where to get your list of recipes](javascript:void(0)) [or how to calculate what the best recipes are and that shouldn't impact the view](javascript:void(0)) [which is only worried about displaying the recipes.](javascript:void(0)) [I do want to point out that the MVC design pattern doesn't dictate what type](javascript:void(0)) [of data access you use.](javascript:void(0)) [You can use web services, relational databases, file system, document databases,](javascript:void(0)) [any form of storage behind the scenes.](javascript:void(0)) [And it also doesn't dictate what your business objects or domain layer should look like.](javascript:void(0)) [In fact, the MVC design pattern doesn't care if your application has layers or not.](javascript:void(0)) [It's simply a design pattern for building a user interface and nothing more.](javascript:void(0)) [So the MVC framework is designed to help you follow this MVC design pattern](javascript:void(0)) [by giving you tools and classes to build models, views and controllers.](javascript:void(0)) [The MVC framework also has some additional goals.](javascript:void(0)) [One of these goals is to embrace the web and to be able to work very closely](javascript:void(0)) [with web technologies like JavaScript, HTML and CSS.](javascript:void(0)) [There is no large obstructions here that try to shield you from knowing things](javascript:void(0)) [like what HTTP verb is being used to place a request.](javascript:void(0)) [We'll learn about those things in this course.](javascript:void(0)) [There is also nothing that makes it harder for you to work with the designer](javascript:void(0)) [who can help you style and make your site look good using CSS.](javascript:void(0)) [Another goal for MVC is to run on top of the core ASP.NET run time.](javascript:void(0)) [The ASP.NET run time is been around for over ten years now.](javascript:void(0)) [And it's proven itself to be secure and stable and optimized for performance.](javascript:void(0)) [If you're already familiar with ASP.NET and you know how to work with HTTP modules and handlers](javascript:void(0)) [and caching and diagnostics, then you'll feel right at home in the MVC environment.](javascript:void(0)) [A third goal for the framework is to be extensible.](javascript:void(0)) [We'll look at many extensibility points as we work through this course.](javascript:void(0)) [If you do not like how the MVC framework behaves in some certain area, chances are,](javascript:void(0)) [all you need to do is plug in the right component](javascript:void(0)) [to make it behave a little bit differently.](javascript:void(0)) [And finally the last goal of the MVC framework is to be testable.](javascript:void(0)) [We will look at unit testing and tester and development techniques in this course](javascript:void(0)) [and we'll see that MVC framework went to a lot of trouble to try and obstruct the ways of some](javascript:void(0)) [of the thornier issues in testing user interface-oriented code.](javascript:void(0)) [But right now what I want to do is return the Visual Studio and make some](javascript:void(0)) [of these concepts a little more concrete.](javascript:void(0))

* + [Making Changes](javascript:void(0))

[Let's look at the details of what happens when we click on the about link in our application.](javascript:void(0)) [The browser is going to send off a request to slash home slash about](javascript:void(0)) [and that request will reach our development web server here on the local host](javascript:void(0)) [which is this machine and that request will be received by IIS Express.](javascript:void(0)) [IIS Express is running and it's in the background.](javascript:void(0)) [You can see it if you go to the system tray area of the task bar.](javascript:void(0)) [Sometimes it hides itself here because it's shy.](javascript:void(0)) [But if you click on the little arrow, you can see that the icon looks](javascript:void(0)) [like stacked blue pizza boxes and I can right click it to see more details](javascript:void(0)) [like show me all the applications that you're currently running.](javascript:void(0)) [I can see it's running my OdeToFood application and it's running it on port 56470.](javascript:void(0)) [That's just a temporary port used for development.](javascript:void(0)) [ISS express is going to take that request and deliver it to my MVC application.](javascript:void(0)) [Inside of every MVC application, there is a routing engine which takes request and tries](javascript:void(0)) [to deliver them to the proper component.](javascript:void(0)) [We'll look at the routing API in the next module, but for now you're just going to have](javascript:void(0)) [to trust me when I tell you that the request ultimately ends up inside](javascript:void(0)) [of a class that's inside of the controllers folder.](javascript:void(0)) [In fact, it ends up in a class called home controller because by default a request](javascript:void(0)) [for slash home something will come to the home controller and a request for slash home slash](javascript:void(0)) [about will end up inside of this about method inside of the home controller.](javascript:void(0)) [So you can see there is a naming convention in play here.](javascript:void(0)) [Where if you just name things correctly, request will find their way to the proper place](javascript:void(0)) [and this isn't the only scenario where naming conventions play a role in MVC.](javascript:void(0)) [The about method doesn't have much work to do](javascript:void(0)) [and by the way we may call this an action in ASP.NET MVC.](javascript:void(0)) [So the public method inside of a controller are actions.](javascript:void(0)) [This action doesn't have much work to do.](javascript:void(0)) [It's not even really building a model.](javascript:void(0)) [But let me put a break point here by clicking out to the left](javascript:void(0)) [of the editor window and getting that little red dot.](javascript:void(0)) [Now, we can run it with the debugger](javascript:void(0)) [and see execution actually pause here when I click on the about link.](javascript:void(0)) [The debuggers are a great way to step through your code](javascript:void(0)) [and see what it's doing and inspecting variable values.](javascript:void(0)) [Let me click on about and you can see we hit this breakpoint.](javascript:void(0)) [And now I can press F10 to step through code one line at a time.](javascript:void(0)) [I can hover over things to view their values and so forth.](javascript:void(0)) [The line of code we're on now is about to return a view.](javascript:void(0)) [A view is one type of action result that you can return from an action](javascript:void(0)) [that tells the MVC framework what to do next.](javascript:void(0)) [Maybe you want to redirect the user or maybe you want to return some data](javascript:void(0)) [in a format for JavaScript to consume it.](javascript:void(0)) [Returning a view is telling the MVC framework that the next thing you want](javascript:void(0)) [to do is render a view, but what view will the MVC framework render?](javascript:void(0)) [Well there is also a naming convention in play here too.](javascript:void(0)) [The MVC framework is going to look inside of the views folder of this application.](javascript:void(0)) [Notice there is a folder called home that matches the controller I'm in,](javascript:void(0)) [that home controller without the controller pat of it.](javascript:void(0)) [But these are the views for the home controller and if I expand that,](javascript:void(0)) [I can see there is a view called about, that matches the action that I'm inside of.](javascript:void(0)) [The about action of the home controller and if I open this, what we'll see is the information](javascript:void(0)) [that is pretty much what we'll see if I hit F5 to continue and let this view render.](javascript:void(0)) [This about view has a dot CSHTML extension that is just the default extension for Razor views.](javascript:void(0)) [Notice that that view that appears on the left hand side doesn't include everything](javascript:void(0)) [that you see in the browser.](javascript:void(0)) [It doesn't include the text, your logo here or the register link or the log](javascript:void(0)) [on link or home about and contact.](javascript:void(0)) [It really just includes the content that is specific to this about action.](javascript:void(0)) [All these other pieces including the copyright at the bottom, those are things that we need](javascript:void(0)) [to appear on every page in this application.](javascript:void(0)) [So, they have been factored out of the individual views like about dot CSHTML.](javascript:void(0)) [We'll see how to build what's known as a lay out view that contain all that common mark up.](javascript:void(0)) [For now, what I want to focus on is just how do I deliver additional information](javascript:void(0)) [from the controller to this view and then have it displayed on this page.](javascript:void(0)) [So, let me stop debugging and come back into the home controller](javascript:void(0)) [and we'll make some changes here inside of about.](javascript:void(0)) [There is a couple of different ways to get information to the view.](javascript:void(0)) [One approach is to use the approach that we see here in the about action and that's simply](javascript:void(0)) [to put information inside of the ViewBag.](javascript:void(0)) [ViewBag is a dynamically typed object in C sharp.](javascript:void(0)) [That means you can add any sort of property to the ViewBag and it will be available inside](javascript:void(0)) [of the view to pull out and retrieve and display.](javascript:void(0)) [That's currently what's happening with this message property.](javascript:void(0)) [So it could also say something like ViewBag.Location equals Maryland](javascript:void(0)) [and the United States of America and do a build](javascript:void(0)) [and that value will be available inside of ViewBag also.](javascript:void(0)) [If I want to display that, I could come into about dot CSHTML](javascript:void(0)) [and let's remove what is currently inside of here and come up.](javascript:void(0)) [We're already displaying ViewBag.Message inside of an h2 tag so let me just have a div](javascript:void(0)) [that says location is at ViewBag.Location.](javascript:void(0)) [The amp sign is a way to tell the razor view engine here is a C sharp expression.](javascript:void(0)) [I want you to evaluate it and take the result and write it into the response right here](javascript:void(0)) [where I'm placing this particular expression.](javascript:void(0)) [So by having that in place there and saving the view, I should be able to come back](javascript:void(0)) [out to the browser and refresh and see that we now have location mailed in the USA.](javascript:void(0)) [So this is one approach that a controller can use to pass along information to a view](javascript:void(0)) [but let's look at a different approach,](javascript:void(0)) [an approach using what we call a strongly typed model.](javascript:void(0)) [Every MVC project will start with a models folder inside.](javascript:void(0)) [You can use this folder when you create new models for your application.](javascript:void(0)) [Although in reality, models can live anywhere even in a different project](javascript:void(0)) [that you reference from this project.](javascript:void(0)) [But we use the models folder for now.](javascript:void(0)) [All I need to do is right click this.](javascript:void(0)) [I want to add a new class.](javascript:void(0)) [I'll call this the about model.](javascript:void(0)) [And we'll keep things very simple.](javascript:void(0)) [This is a class, I'm just going to give it two properties.](javascript:void(0)) [I'll add them using a code snippet in Visual Studio so I can just type P-R-O-P](javascript:void(0)) [and hit tab twice that will expand out to a property.](javascript:void(0)) [I'll give the first property a type of string.](javascript:void(0)) [We will call this property name and I'll give us another property.](javascript:void(0)) [This one also type string will include the location.](javascript:void(0)) [And let's say that this about model represents everything that we want to show](javascript:void(0)) [when the user visits the about page.](javascript:void(0)) [So what I'll need to do is come into the HomeController and instead of using the ViewBag,](javascript:void(0)) [I'll declare a new instance of our about model.](javascript:void(0)) [The about model isn't a different name space that I'm currently in.](javascript:void(0)) [So for this to be legal C sharp code, I need to include the name space](javascript:void(0)) [that this is defined in which is OdeToFood.Models.](javascript:void(0)) [An easy way to do that is just to use this drop down menu here that's sometimes hard](javascript:void(0)) [to get a hold of, and click on this to add the using for OdeToFood.Models.](javascript:void(0)) [A quick way to do this is when your cursor is on that unexpected identifier.](javascript:void(0)) [Just hold down the control key and hit period.](javascript:void(0)) [That will drop down the menu then select it using enter.](javascript:void(0)) [That adds the using statement for me and we now have legal C sharp code.](javascript:void(0)) [Now, usually when you build a model, you need to do some sort](javascript:void(0)) [of data access or some calculations.](javascript:void(0)) [But right now, we'll just counter up something.](javascript:void(0)) [We'll save the data access stuff for later.](javascript:void(0)) [I'll give this a name, you can use your name and I will say](javascript:void(0)) [that my location is still Maryland in the United States.](javascript:void(0)) [And now that the model is complete, I can tell the MVC framework](javascript:void(0)) [that when it renders this view, you use this model.](javascript:void(0)) [So pass this model along to the view.](javascript:void(0)) [The next step would then be to modify the view to use this new model.](javascript:void(0)) [So let me come into the About.cshtml view.](javascript:void(0)) [The first thing I'll do up here at the top is tell the view about this model.](javascript:void(0)) [So I'm going to use what's known as a directive which is at model and that's a lower case M,](javascript:void(0)) [very important that you use the lower case M here,](javascript:void(0)) [tell it that the model it should expect will be an instance of about model.](javascript:void(0)) [And again in order for the name space to work, I'll have to hit control period and drop down](javascript:void(0)) [and say that this isn't OdeToFood.Models.AboutModel,](javascript:void(0)) [in this case it doesn't add a using statement,](javascript:void(0)) [just adds the pooling name space qualified type name which is just fun.](javascript:void(0)) [And now instead of going to ViewBag which we never knew quite what is in ViewBag.](javascript:void(0)) [There could be a message property, there might not be a message property.](javascript:void(0)) [Instead of using ViewBag, I will use the model property of this view.](javascript:void(0)) [The model property knows what we're passing in as a model](javascript:void(0)) [and quite often you can get IntelliSense, let me do a quick build](javascript:void(0)) [by doing shift control B making sure the project builds,](javascript:void(0)) [and then maybe when I hit period, I can see the IntelliSense.](javascript:void(0)) [I can see there is a name, that's what we'll put out here.](javascript:void(0)) [And I can see there is a location.](javascript:void(0)) [So, let me switch the server to the model.location just to save everything,](javascript:void(0)) [do another build, amount and refresh.](javascript:void(0)) [And we're now building this view using a strongly typed model.](javascript:void(0)) [And this is a very common approach to MVC development.](javascript:void(0)) [So, let me just reiterate what we did here.](javascript:void(0)) [This model property with the capital M at Model that will represent the model object](javascript:void(0)) [that I passed in to this view from the controller action.](javascript:void(0)) [And in fact, I told Razor that it should be expecting a model of type about model](javascript:void(0)) [and now visual studio and Razor and everyone in the world knows](javascript:void(0)) [that the object it should be receiving is an object of that type](javascript:void(0)) [that it should have name and location properties.](javascript:void(0)) [And what we've seen so far is really the essence of MVC, I have a request arrive](javascript:void(0)) [for slash home slash about that will be delivered by the framework to this](javascript:void(0)) [about action on my home controller.](javascript:void(0)) [I build a model, pass it to the default view.](javascript:void(0)) [When that view renders, it picks apart that model and puts all the pieces](javascript:void(0)) [in the right location for-- to display correctly to the user.](javascript:void(0)) [So, I can see this working, but how can I be a little more assured](javascript:void(0)) [that this will always work for me?](javascript:void(0)) [That's where unit testing come into play.](javascript:void(0)) [So, let's look at unit testing with ASP.NET MVC next.](javascript:void(0))

* + [Unit Testing](javascript:void(0))

[As I mentioned previously, one of the goals of the MVC framework is to be testable.](javascript:void(0)) [Specifically, your controller should be testable.](javascript:void(0)) [I should be able to exercise the logic inside of the controller action from a unit test](javascript:void(0)) [and be able to write assert to make sure that the logic is behaving correctly](javascript:void(0)) [and I should be able to do that without involving a web server or HTTP messages,](javascript:void(0)) [or my network card or other things that make testing complicated.](javascript:void(0)) [Fortunately, when I created this application, we told Visual Studio](javascript:void(0)) [to create a unit test project for us.](javascript:void(0)) [And if I look in the solution explorer window,](javascript:void(0)) [I'll see there are actually two projects in my application solution.](javascript:void(0)) [The first one is OdeToFood that's the web application](javascript:void(0)) [and the second one is OdeToFood.test.](javascript:void(0)) [This is my testing project.](javascript:void(0)) [Inside of the testing project are some test the Visual Studio added](javascript:void(0)) [when I created this application, these aren't the most fantastic tests in the world,](javascript:void(0)) [but they do demonstrate how to implement unit test.](javascript:void(0)) [This is a new feature for Visual Studio 2012 by the way.](javascript:void(0)) [In previous versions of Visual Studio the free express edition did not include a unit testing](javascript:void(0)) [tool, but it 2012 we have the framework known as MS test.](javascript:void(0)) [So, let me open up the one set of test that it's inside of here the home controllers test.](javascript:void(0)) [And you can see that tests are identified by attributes.](javascript:void(0)) [With MS test you need an attribute on every class that contains a test,](javascript:void(0)) [that's the test class attribute that you see.](javascript:void(0)) [This attribute helps the test runner which we'll use in just a bit](javascript:void(0)) [to find all the test inside of this project.](javascript:void(0)) [And then every individual tests inside of a test class has a test method attribute.](javascript:void(0)) [What a test runner is going to do is once it finds your test,](javascript:void(0)) [it will instantiate every test class](javascript:void(0)) [and then invoke every method that's a test method inside of that class.](javascript:void(0)) [It will then check to see if each of those test passes or fails.](javascript:void(0)) [You can tell the test runner when something fails.](javascript:void(0)) [Inside of every unit test, you typically have three steps.](javascript:void(0)) [Each step can have one or more lines of code.](javascript:void(0)) [The first step in the arrange step.](javascript:void(0)) [This is where you create objects that you want to test.](javascript:void(0)) [In this index test, we're looking at the arrange act just consisting](javascript:void(0)) [of just instantiating the home controller.](javascript:void(0)) [That's the object that we want to test.](javascript:void(0)) [Thanks to the testability of ASP.NET that we see we can instantiate our controllers from anywhere](javascript:void(0)) [and not have them fail just because they aren't processing a real HTTP request.](javascript:void(0)) [After the arrange step comes the act step.](javascript:void(0)) [This is where you set things in motion.](javascript:void(0)) [You usually invoke a method or set a property, do something to the object that you want to test](javascript:void(0)) [and have it produce a result which you'll capture and examine](javascript:void(0)) [in the third step which is the assert step.](javascript:void(0)) [This is where you rate an assertion that should prove whether or not something work correctly.](javascript:void(0)) [In this test we're just asserting that the home controller produced a result](javascript:void(0)) [or the result .ViewBag.Message was equal to some specific string that's here.](javascript:void(0)) [The test is just making sure](javascript:void(0)) [that ViewBag.Message property is set the way it expects it.](javascript:void(0)) [As I said, this isn't the greatest in the world.](javascript:void(0)) [There is no real logic or need to test it that exact string as a match,](javascript:void(0)) [but it does demonstrate how to use the test framework and to test API](javascript:void(0)) [and you can see the arrange act assert steps that are common to almost every unit test.](javascript:void(0)) [The assert API by the way includes a number of different assertions you can write.](javascript:void(0)) [This assert is the R equal assert.](javascript:void(0)) [It's to make sure that two objects are equal.](javascript:void(0)) [But you can also assert that something is true or something is false or something is not null.](javascript:void(0)) [There are many different types of assert.](javascript:void(0)) [If the assertion fails, the test will fail.](javascript:void(0)) [To see what test are passing and what test are failing.](javascript:void(0)) [I just need to get the test runner going.](javascript:void(0)) [I can do that by going to the test menu and saying run all the tests,](javascript:void(0)) [the shortcut key there is control R followed by an A and when I do](javascript:void(0)) [that I'll see the test explorer appear, test that have a green icon next](javascript:void(0)) [to them are passing test that's good.](javascript:void(0)) [Test that have a red icon next to them are failing test that would be bad.](javascript:void(0)) [Let's see if we can get a failing test, right now we only have three tests](javascript:void(0)) [and they are all passing but I can break something easily.](javascript:void(0)) [I'll come over into the home controller and remove this bit of code](javascript:void(0)) [that is passing the model in to build the view result.](javascript:void(0)) [That means my view will not receive a model object and if I run the application when it's](javascript:void(0)) [in this state, and go to that about action, we'll get a runtime error because the view needs](javascript:void(0)) [that model when we don't pass a model](javascript:void(0)) [in to build the view result we'll get a null model property and when we try to--](javascript:void(0)) [dereference that by doing model.name we'll get the object reference not set](javascript:void(0)) [to an instance of an object exception.](javascript:void(0)) [I could avoid this with a unit test, a unit test that makes sure that the about action](javascript:void(0)) [of the home controller sets a model object and that the model object is of the right type](javascript:void(0)) [and having a unit test in place for that will mean I can see](javascript:void(0)) [that it's failing before it ever try to push this somewhere where someone will see it,](javascript:void(0)) [I'll catch this error before the application ever runs.](javascript:void(0)) [Let's try to do that.](javascript:void(0)) [If I come over into the home controller test, I can see there already is an about test here.](javascript:void(0)) [But the about test is just making sure that the about action is producing a result,](javascript:void(0)) [any result and that the result is not null.](javascript:void(0)) [And so, the change I made in the home controller will still not break this test.](javascript:void(0)) [Let's run all the test again.](javascript:void(0)) [All the test are still passing.](javascript:void(0)) [I need to do a little more in this test.](javascript:void(0)) [I need to really assert that result.model is not null.](javascript:void(0)) [That's the important part that the view needs.](javascript:void(0)) [We need a result but that result also needs a model associated with it.](javascript:void(0)) [So with that change in the test and we run all the test again, now we have a failing test.](javascript:void(0)) [If I click on the failing test, I'll get down here in the bottom of the test explorer window,](javascript:void(0)) [a StackTrace I can click on the top link here, it will take me right to the line of code](javascript:void(0)) [that failed, in this case it failed because result.model is null](javascript:void(0)) [and the assertion is making sure that it's not null and to fix that, I just need to go back](javascript:void(0)) [to the controller and put the code back in place so I actually pass a model along,](javascript:void(0)) [now I can hit control RA to run all the test again.](javascript:void(0)) [And we should come out with three passing tests.](javascript:void(0)) [So this is all the time we have to talk about testing right now but we'll return](javascript:void(0)) [to this topic later in the course.](javascript:void(0)) [This is just a quick demonstration that was intended to show you how easy it can be](javascript:void(0)) [to test the logic in your presentation layer, specifically the logic](javascript:void(0)) [that you have inside of your controllers.](javascript:void(0)) [Controllers are relatively plain C sharp classes.](javascript:void(0)) [We can instantiate them without having a web server](javascript:void(0)) [or a web browser or an HTTP request running.](javascript:void(0)) [And compared to the view, it's the controllers that are doing all the hard work.](javascript:void(0)) [Views are just simple in the MVC design pattern we use them just as templates](javascript:void(0)) [to control the placement of data to most of the unit testing effort in an MVC application,](javascript:void(0)) [it was against the controllers and the models.](javascript:void(0))

* + [JavaScript and CSS](javascript:void(0))

[So far, we've started to see how views, models and controllers work together](javascript:void(0)) [and we've also seen how you see this to write unit test for an MVC application.](javascript:void(0)) [But I also mentioned earlier that another goal of the MVC framework is to embrace the web](javascript:void(0)) [and that means being able to use standard web technologies like HTML 5, JavaScript, and CSS 3.](javascript:void(0)) [We've already seen a bit about how to work with HTML.](javascript:void(0)) [We know the razor views that are in our views folder emit this HTML and we've already seen](javascript:void(0)) [that the project does set up to use HTML 5.](javascript:void(0)) [But there's a couple of other folders we haven't looked at yet,](javascript:void(0)) [the scripts folder and the content folder.](javascript:void(0)) [Inside of the scripts folder you'll find JavaScript files.](javascript:void(0)) [The JavaScript files here include the popular jQuery library as well](javascript:void(0)) [as some other jQuery plug ins we'll be using later in this course,](javascript:void(0)) [plugins like jQuery validation and jQuery UI.](javascript:void(0)) [And the content you will find the CSS style sheet for this site.](javascript:void(0)) [This file site dot CSS is the file that's responsible for all the color](javascript:void(0)) [and font choices that you see in the application.](javascript:void(0)) [Let me make some basic changes to the body style here and see how that's reflected](javascript:void(0)) [in the application instead of a white background color, let's go with 3, 3, 3, and instead this](javascript:void(0)) [or that color let's change that over to white.](javascript:void(0)) [So essentially we're flipping the background and foreground colors.](javascript:void(0)) [And now once I save that file and refresh this page, you can see that influence is how](javascript:void(0)) [at least that top of the page displays.](javascript:void(0)) [But let me make a few more tweaks to that file and come right back.](javascript:void(0)) [Now I finished making a few tweaks to my CSS file.](javascript:void(0)) [And I've and added an image to the project.](javascript:void(0)) [And this is what the applications looks like now.](javascript:void(0)) [It's still not the prettiest application in the world but this is what we're going to be working](javascript:void(0)) [on throughout the rest of the course.](javascript:void(0)) [And hopefully we'll be adding features and making look it better.](javascript:void(0)) [If you are interested in looking at the source code for this project,](javascript:void(0)) [just go to odetofoodmvc4.codeplex.com.](javascript:void(0)) [From here you can come into the source code tab.](javascript:void(0)) [And from in the source code section you can download the latest source code.](javascript:void(0)) [You can also click this History link and go back and see what the code look](javascript:void(0)) [like at specific point in time like this point in time with the first video is nearly over.](javascript:void(0))

* + [Summary](javascript:void(0))

[In this first module I've demonstrated how to use the Web Platform Installer](javascript:void(0)) [to create a development environment or you can build ASP.NET MVC Web Applications.](javascript:void(0)) [With just a few clicks we were able to install Visual Studio, SQL Server and the MVC Framework.](javascript:void(0)) [We still how to create a new MVC application and we talked](javascript:void(0)) [about the Model, View, Controller design pattern.](javascript:void(0)) [In the rest of this course we will be again drilling](javascript:void(0)) [into specific areas of the MVC Framework.](javascript:void(0)) [And see how to build Controllers, how to build Models, send story information in SQL Server.](javascript:void(0)) [We'll take a very close look at Razor and also how to use jQuery plugins](javascript:void(0)) [to add Ajax features to our application.](javascript:void(0)) [We'll be writing more unit test and then](javascript:void(0)) [in the end we'll deploy our application live to the internet.](javascript:void(0)) [Thanks to Microsoft Windows Azure websites.](javascript:void(0)) [So don't go away, we're only getting started with this excitement.](javascript:void(0))

* Controllers in ASP.NET MVC 4
  + [Introduction](javascript:void(0))

[Hi, this is Scott Allen and in this module we're going to take some time](javascript:void(0)) [to understand how controllers work in the mvc framework.](javascript:void(0)) [Specifically we're going to look at the routing rules which deliver in the incoming requests](javascript:void(0)) [for from the web to our controllers and then look at controller actions.](javascript:void(0)) [You might remember from the introduction](javascript:void(0)) [that the controller actions are the public methods on our controllers.](javascript:void(0)) [They have the ability to respond to an incoming http request from the web.](javascript:void(0)) [We'll also be taking a look at action filters which can introduce pre and post processing](javascript:void(0)) [to an action and then see how we can input data to an action with action parameters](javascript:void(0)) [and output different types of results from an action using different action results.](javascript:void(0))

* + [Routes and Controllers](javascript:void(0))

[One of the questions that we left unanswered in the introduction was this,](javascript:void(0)) [how does the sp dot net know how to deliver a request like slash home,](javascript:void(0)) [slash about to our home controller.](javascript:void(0)) [And the answer to that question is in the routing engine.](javascript:void(0)) [The routing engine is a core part of asp dot net, it's not tied to the mvc framework,](javascript:void(0)) [you can use the routing engine to route requests for web forms, WCF services,](javascript:void(0)) [really any type of resource but in mvc we use this routing engine](javascript:void(0)) [to direct requests to our controllers.](javascript:void(0)) [To do this we give the routing engine a map to follow using a map route API.](javascript:void(0)) [A route map allows us to provide a friendly name for the route, a pattern for the route](javascript:void(0)) [and default parameters for the route, the most important pieces to look](javascript:void(0)) [at are the pattern and the default.](javascript:void(0)) [Think about the goal of the routing engine, its job is to examine a URL and figure](javascript:void(0)) [out where to send it for processing.](javascript:void(0)) [When it examines the URL it needs to pick put pieces of data from that URL,](javascript:void(0)) [little hints about where to send the request.](javascript:void(0)) [In an MVC application we want the URL to specify the controller and action to invoke](javascript:void(0)) [and perhaps some other data so we give the routing engine a pattern it can use](javascript:void(0)) [and apply to find parameters in the URL.](javascript:void(0)) [The words inside those curly braces will be the parameter names,](javascript:void(0)) [if the routing engine doesn't find the specific piece of data in the URL](javascript:void(0)) [like the controller name or the action name it can use one of the default values](javascript:void(0)) [that we specify in the parameter defaults.](javascript:void(0)) [Let switch into visual studio and experiment with this.](javascript:void(0)) [Inside a visual studio let me introduce you to the global dot asax dot cs file.](javascript:void(0)) [This is a bit of a magical file in asp dot net.](javascript:void(0)) [If I open it up you can see we have a class here derived from http application and this allows us](javascript:void(0)) [to hook into hook into some application level events like application start,](javascript:void(0)) [this method will be magically invoked by asp dot net before you process the first http request.](javascript:void(0)) [So when your application starts running the code here will execute one time before any](javascript:void(0)) [of your controllers start executing and so this is where we put in some configuration](javascript:void(0)) [that the application needs like the routing configuration.](javascript:void(0)) [The routing configuration is done by this line of code, route config dot register route,](javascript:void(0)) [what we pass in is the global routing table the table](javascript:void(0)) [that contains all the routs for the entire application.](javascript:void(0)) [It will be empty initially but when we call register routes](javascript:void(0)) [that will add entries into the routing table.](javascript:void(0)) [I can get into the definition of this method](javascript:void(0)) [if I just put my cursor tight here on the method and press F12.](javascript:void(0)) [Now we are inside of a class called route config, this class actually lives](javascript:void(0)) [in the app start folder, you can find it right here.](javascript:void(0)) [It's given to me by default by visual studio when I created this application](javascript:void(0)) [so all this code was already written for me and if you've been working](javascript:void(0)) [with mvc previously this is a little bit different in mvc 4 than it was](javascript:void(0)) [in previous versions because all of the start up code that used to be inside](javascript:void(0)) [of global dot asax dot cs has now been factored into classes that are in this app start folder.](javascript:void(0)) [All these classes are called](javascript:void(0)) [from the application start method in global dot asax dot cs.](javascript:void(0)) [Here's the code that you saw on that previous slide, we're walking up to the route collection](javascript:void(0)) [and saying let's add a new route, we'll call it default and here's the type of URL you're going](javascript:void(0)) [to look for so if you see something that looks like slash home slash index,](javascript:void(0)) [treat home as the controller name, index as the action name and you can also look for an ID](javascript:void(0)) [but if there's not one that's okay we've provided a default value for it.](javascript:void(0)) [And in fact if you don't see an action name just treat the default action name as index.](javascript:void(0)) [When the routing engine picks apart a URL into these pieces it builds up a data structure](javascript:void(0)) [that it passes along to the mvc framework and then the framework can look](javascript:void(0)) [at that data structure and say what did you find for a controller and what did you find](javascript:void(0)) [for an action and it uses that information](javascript:void(0)) [to direct the request tour controller like the home controller.](javascript:void(0)) [In fact that information is available to us, it's available any where throughout the request.](javascript:void(0)) [Let's swing over to the home controller and see if we can inspect these values.](javascript:void(0)) [Instead of building a message that says something about modify this template,](javascript:void(0)) [let me build a different sort of message.](javascript:void(0)) [What I want to do is see what kind of controller we're inside of and I can do that by going](javascript:void(0)) [to the route data data structure, this is the data structure that's built](javascript:void(0)) [by the routing engine, it'll have a collection of values inside and I can basically walk](javascript:void(0)) [up to the values and say what do you have for a controller value,](javascript:void(0)) [I could also ask it what it has for an action value and finally I can say what do you have](javascript:void(0)) [for an ID value if any route data dot value sub ID and I can put all these pieces together](javascript:void(0)) [into a message, let's use string dot format and I want parameter 0](javascript:void(0)) [which will be the controller name, parameter 1 which is the action name](javascript:void(0)) [and parameter 2 which will be the ID.](javascript:void(0)) [So controller action ID and this is now the message that I'll display on the home page](javascript:void(0)) [so instead of modify this template we'll put a message here and let me run the application.](javascript:void(0)) [And what we should be able to see that just by going to the root of the application,](javascript:void(0)) [just local host slash we get a value of whom for controller and a value of index for the action](javascript:void(0)) [and what happens if we go to slash home, same values, home and index.](javascript:void(0)) [So I can go to slash home slash index slash 5 1 2](javascript:void(0)) [and you can see you get the value 5 1 2 for an ID.](javascript:void(0)) [What we're going to see in just a little bit is that it is actually very easy to get](javascript:void(0)) [that ID parameter, we do not have to look inside of route data explicitly](javascript:void(0)) [but for just right now I want to demonstrate the whole goal of the routing engine is](javascript:void(0)) [to pick apart that URL and then build this data structure that is available not only](javascript:void(0)) [to the mvc framework to figure out where it's going to go but also to us,](javascript:void(0)) [it's available inside of controllers, route data is also available inside of a view.](javascript:void(0)) [Typically our code is not going to be looking into route values like this,](javascript:void(0)) [it's the mvc framework that does that to find a controller and you'll notice that the value](javascript:void(0)) [of the controller which is home tells the mvc framework to go out and append controller](javascript:void(0)) [to that word home controller and look for a class by that name.](javascript:void(0)) [It's a case and sensitive search.](javascript:void(0))

* + [Actions and Parameters](javascript:void(0))

[Now that we know a little bit about how routing works let's go into route config dot cs](javascript:void(0)) [and define a new route, this is only something you need to do if a default route doesn't work](javascript:void(0)) [for you but let me show you a scenario where a new route can help.](javascript:void(0)) [Let's imagine that the user expects to be able to come into the application and search](javascript:void(0)) [for a cuisine by its name, so French cuisine or Italian cuisine, in this case at second entry](javascript:void(0)) [in the URL the second segment of the path isn't really an action identifier it's more](javascript:void(0)) [of a parameter, it can be French or German or Italian.](javascript:void(0)) [So the default route really won't work very well for us because we don't want to add an action](javascript:void(0)) [to our controller for every type of cuisine that we have we just want](javascript:void(0)) [that path sent to the action as a parameter.](javascript:void(0)) [So I'll define a new route for this and I need to be careful where I define this route](javascript:void(0)) [because the order in which rotes are added to the global route collection is significant,](javascript:void(0)) [what the routing engine will do is evaluate each mapped route that we place](javascript:void(0)) [into the route collection the first one that matches the URL will win.](javascript:void(0)) [The default route entry that we have here is very greedy;](javascript:void(0)) [it matches nearly any URL that you want to throw at it.](javascript:void(0)) [It matches slash home slash index, it matches slash home, we've even seen it work](javascript:void(0)) [when you just go to the root of the application and it can do this](javascript:void(0)) [because it provides default parameter values to anything that's missing in the URL.](javascript:void(0)) [IF we want to define our own route we'd probably want to place it in front of that default route](javascript:void(0)) [and make that URL a little more restrictive.](javascript:void(0)) [So for example let me map a route and we'll give it the friendly name of cuisine](javascript:void(0)) [and we'll say the URL to match for this has to start with the word cuisine](javascript:void(0)) [and then it can have a parameter which we'll call name which will be the second segment](javascript:void(0)) [of the URL and you might say how can this work there's no control, there's no action.](javascript:void(0)) [Well, the mvc framework does need to know what the controller will be](javascript:void(0)) [and what the action will be so let me add those as defaults.](javascript:void(0)) [I'll create a new anonymously typed object and say controller equals the cuisine controller,](javascript:void(0)) [the default action, let's give it a default action of search](javascript:void(0)) [and name will just make it an empty name and that defines our new route.](javascript:void(0)) [Now if we have a request come in for slash cuisine the routing engine should see that,](javascript:void(0)) [that matches the cuisine route and send things off to a cuisine controller.](javascript:void(0)) [But if a request comes in for slash home,](javascript:void(0)) [slash index that doesn't match the cuisine route it doesn't start with the word cuisine](javascript:void(0)) [so we'll go to the next route entry which is the default route entry slash home slash index will](javascript:void(0)) [match that one.](javascript:void(0)) [As a side note let me briefly mention the purpose](javascript:void(0)) [of this routes dot ignore route method call, the routing engine will not try to process a request](javascript:void(0)) [that going to reach a real file on the file system.](javascript:void(0)) [That's why we can have a request for something like slash content, slash site dot css](javascript:void(0)) [and the routing engine won't interfere with that request.](javascript:void(0)) [The server can simply send back that css file in response, we don't need this request routed](javascript:void(0)) [to a controller, the file already exists we just need to pick it up off the disk and send it back](javascript:void(0)) [to the browser and that's what will happen because the routing engine will see that,](javascript:void(0)) [that's going to reach something on the disk](javascript:void(0)) [and it doesn't interfere and try and route things around.](javascript:void(0)) [But there are some files served up by asp dot net that end with a axt extension](javascript:void(0)) [and these files don't actually exist on the file system they're virtual files](javascript:void(0)) [but asp dot net knows how to serve them up.](javascript:void(0)) [If you've ever worked with asp dot net tracing you might be familiar with trace dot exd.](javascript:void(0)) [We'll take a look at a situation where we have an axd end point later in the course](javascript:void(0)) [but for now just know this is a way to tell a routing engine if you see something](javascript:void(0)) [like trace dot axd in the request just ignore it and don't try to route it,](javascript:void(0)) [someone else will take care of that even though the file doesn't actually exist on the disk.](javascript:void(0)) [Now that we have our new route to find let me run the application by pressing control F 5](javascript:void(0)) [and let's try to come into the cuisine controller and look for Swedish food.](javascript:void(0)) [The response I get back from my application is an acdp 404 error, this is the error code](javascript:void(0)) [that is recognized throughout the noon universe as meaning I cannot find what you're looking](javascript:void(0)) [for which is exactly what I would expect because I told the mcv framework to go looking](javascript:void(0)) [for a cuisine controller but I do not have a cuisine controller in the application.](javascript:void(0)) [So let's fix this problem, I'm going to right click on the controller's folder](javascript:void(0)) [and select add controller, this will bring up a dialogue box](javascript:void(0)) [where I can give my new controller a name, I'll call it the cuisine controller.](javascript:void(0)) [There's a number of other options inside of this dialogue which we'll talk about later](javascript:void(0)) [in the course, for right now I'll just leave the empty NBS controller template selected](javascript:void(0)) [and click add, this will add a controller to my controller's folder.](javascript:void(0)) [If it isn't quote empty it does have an index method inside,](javascript:void(0)) [it is derived from that base controller class and if I build this application](javascript:void(0)) [and run it again, refresh the browser we'll still have a 404 error message](javascript:void(0)) [because we also told the mvc framework inside of val config](javascript:void(0)) [that the action should be the search action and this controller does not have a search action.](javascript:void(0)) [There's also no way to specify the action in the URL we didn't provide](javascript:void(0)) [for that flexibility inside of this route.](javascript:void(0)) [So we really need to call the index action, the search action, we also don't have a view as yet](javascript:void(0)) [but that's okay I'm going to do the simplest possible thing inside](javascript:void(0)) [of an action which is just to return content.](javascript:void(0)) [When you return content you just send a string back down the browser,](javascript:void(0)) [there's no html involved unless you put html in the string, there's no view involved,](javascript:void(0)) [I could just say hello, just to prove that we're reaching the search action properly](javascript:void(0)) [and if I refresh again now we have a result, we have reached that method inside the controller.](javascript:void(0)) [Now let's see if we can pull out that name parameter out of the URL but before I do](javascript:void(0)) [that let me just oint out that actions are nothing more](javascript:void(0)) [than public methods inside of a controller class.](javascript:void(0)) [Anytime you add a public method to this class or any controller class you have to think about it](javascript:void(0)) [as something that will be URL addressable depending on your routing configuration.](javascript:void(0)) [If someone types the right thing into their browser address bar they might be able](javascript:void(0)) [to invoke a public method inside of a controller so don't add any methods inside of here](javascript:void(0)) [that you wouldn't expect to be called via a URL.](javascript:void(0)) [So just keep that in mind as you move forward.](javascript:void(0)) [Right now we're going to try to pull out the name value from the URL](javascript:void(0)) [and I've already demonstrated that we could get to that through route data,](javascript:void(0)) [I could ask for route data dot value sub name but it turns](javascript:void(0)) [out asp dot net mvc makes this even easier because if you add a parameter](javascript:void(0)) [to an action what the mvc framework will do is go out and try to find something that matches](javascript:void(0)) [that parameter name and then just give it to you.](javascript:void(0)) [It will do everything it can to populate that parameter,](javascript:void(0)) [it will look all around the request it will look in routing data,](javascript:void(0)) [so things that were picked out of the URL.](javascript:void(0)) [It will also look in the query string and it will look in posted form values.](javascript:void(0)) [In the case of coming to slash cuisine slash Swedish the mvc framework will see I need a](javascript:void(0)) [parameter called name and something called name was extracted from the URL](javascript:void(0)) [and it will automatically pass that into me.](javascript:void(0)) [What I want to do is just echo that fact when I return this content but before I do](javascript:void(0)) [that what I want to do is make sure I properly encode this value,](javascript:void(0)) [so I'll say message equals server dot html encode name server is a property that I inherit](javascript:void(0)) [on my controller that I can use to get to server type utilities](javascript:void(0)) [and variable including html encode which ill make sure if a user snuck through some sort](javascript:void(0)) [of malicious script tag or something like that it will render as text](javascript:void(0)) [and it will prevent a cross-site scripting attack.](javascript:void(0)) [The razor view engine will do this automatically for me](javascript:void(0)) [but since I'm using a content result I have](javascript:void(0)) [to be a little bit more careful about how I manage user input.](javascript:void(0)) [Let's just take that encoded message and return it as the content of our message,](javascript:void(0)) [now I will do a build, I will come back and refresh the browser](javascript:void(0)) [and you can see we get the name of the cuisine even though it is misspelled,](javascript:void(0)) [we get the name of the cuisine back out and we can show it in the browser.](javascript:void(0)) [Now currently the way our route is defined I can come to the cuisine controller and not pass](javascript:void(0)) [in a cuisine name and it renders just fine that's because in our routes we said](javascript:void(0)) [that if you do not find a name just use and empty string.](javascript:void(0)) [If I take this default out of here though something else will happen,](javascript:void(0)) [let me do a quick build and refresh now we're back to a 404 error message](javascript:void(0)) [because essentially what has happened is that we've told the mvc framework that an order](javascript:void(0)) [to match this route you have to have a name it's not optional, we didn't provide a default value](javascript:void(0)) [for it, therefore we didn't match the cuisine route we went to the default route,](javascript:void(0)) [we probably went to the index action of the cuisine controller which doesn't exist](javascript:void(0)) [and that's why we have this problem.](javascript:void(0)) [So let me put this back in and instead of specifying an empty string I'm going](javascript:void(0)) [to specify URL parameter dot optional, that's just telling the mvc framework it's okay](javascript:void(0)) [for this to not exist inside of the URL and now if I do a build and refresh we're getting](javascript:void(0)) [to the cuisine controller again there's just no value present for the name.](javascript:void(0)) [If I wanted to provide a default value I could say name equals French or name equals Swedish,](javascript:void(0)) [I could also do that using c sharp and saying that this has a default value of say French](javascript:void(0)) [and now if I do another build and refresh the french value comes](javascript:void(0)) [through even though I didn't specify that in the URL and just to show you](javascript:void(0)) [that the mvc framework is looking around in different places](javascript:void(0)) [for something called name let me enter a query string so I'll go to the cuisine controller](javascript:void(0)) [but say query string name equals a Italian and we get Italian comes](javascript:void(0)) [through so even though the routing engine didn't find a name](javascript:void(0)) [in the URL the mvc framework found something called name in the query string](javascript:void(0)) [and it just passed it into the action force automatically.](javascript:void(0)) [So you almost never inside of a controller action directly inspect route data](javascript:void(0)) [or directly inspect the query string which you could do by going](javascript:void(0)) [to request dot query string inside of a controller method,](javascript:void(0)) [all of those things are taken care of for us by the mvc framework,](javascript:void(0)) [it's just going to find things in the request and pass them into us.](javascript:void(0)) [And of course it should still work if I say slash cuisine slash Swedish](javascript:void(0)) [and actually spell it correctly this time that value of course was found in the URL.](javascript:void(0)) [As we move further along in this course we'll see more advanced scenarios involving multiple](javascript:void(0)) [action parameters and even complex types here with multiple properties](javascript:void(0)) [and see how the mvc framework behaves with those.](javascript:void(0))

* + [Action Results](javascript:void(0))

[At this point we've used routing data and query string values as inputs](javascript:void(0)) [to our cuisine controller search action and that will prove to be useful knowledge later](javascript:void(0)) [when we start doing data access and need to use those incoming values but for now I want](javascript:void(0)) [to talk about the output of an action.](javascript:void(0)) [We just used the content result to return a string literal from our action](javascript:void(0)) [and the producing method which is a factory type method that produces](javascript:void(0)) [and action result, its called content.](javascript:void(0)) [So that's simply the method that you need to invoke to create a content result.](javascript:void(0)) [There's many other types of action results available the name column represents the type](javascript:void(0)) [that is derived from action result, the type of the object that you'll create](javascript:void(0)) [and produce in return from an action.](javascript:void(0)) [So all of the class names that you see here they all derive from action result](javascript:void(0)) [and the producing method is the factory method you can invoke](javascript:void(0)) [to construct a particular result, it puts things together for you.](javascript:void(0)) [Previously we've also used the view method to produce a view result and we saw](javascript:void(0)) [in the introduction how this will render a view that is somewhere inside of our views folder,](javascript:void(0)) [there are also results we can use to return a file back to the client,](javascript:void(0)) [return data in java script object notation for java script to consume and also a result](javascript:void(0)) [to tell the browser to redirect to a different URL.](javascript:void(0)) [Let's see how some of these different action results behave](javascript:void(0)) [by switching back into visual studio.](javascript:void(0)) [We've seen how the content and view results work, let's try a different result,](javascript:void(0)) [let's look at one of the redirect results,](javascript:void(0)) [you can see just in the intelliSense window there's many types of redirects,](javascript:void(0)) [for instance there is a redirect permanent, this returns an http 3 or 2 status code](javascript:void(0)) [which would basically say if someone came to this cuisine controller it would say go](javascript:void(0)) [over the Microsoft dot come and never look back this is a permanent redirect,](javascript:void(0)) [you never have to come here again.](javascript:void(0)) [There's also redirects that you can use to go to specific actions in the same controller](javascript:void(0)) [or an action on a different controller that is in this application so I could say when you come](javascript:void(0)) [to the cuisine controller what I want to do is redirect you to the index action](javascript:void(0)) [of the home controller and I can even pass along values with this.](javascript:void(0)) [This is a temporary redirect and I'll show you later in the course why these types](javascript:void(0)) [of redirects are very common for some scenarios.](javascript:void(0)) [What's interesting about this redirect to action is that it actually coordinates](javascript:void(0)) [with the routing engine behind the scenes, it walks up toe the routing engine and says](javascript:void(0)) [if I want to go to a URL that would reach the index action of the home controller what would](javascript:void(0)) [that URL look like and based on the way we've configured the routing engine,](javascript:void(0)) [the routing engine will reply with the proper URL](javascript:void(0)) [and you can even pass along additional parameters, if the parameters are known](javascript:void(0)) [to the route they're actually in the URL, the routing engine will produce a URL](javascript:void(0)) [that has the value in the URL but I can also say let's go to the index action](javascript:void(0)) [of the home controller and pass along the name so the way you do this is](javascript:void(0)) [to pass an anonymously typed object in as the third parameter, mvc framework will pick](javascript:void(0)) [that apart and decide that okay you need to get a parameter called name over here.](javascript:void(0)) [So what would this look like, let me do a build, we're currently at cuisine slash Swedish](javascript:void(0)) [so the cuisine is Swedish but after I've built the application if I refresh this we should end](javascript:void(0)) [up back on the home page and notice the URL is just slash query string name equals Swedish](javascript:void(0)) [because there is no name in the URL when we go to that default route](javascript:void(0)) [to reach the home controller and so the framework decided to build this URL](javascript:void(0)) [with the name parameter in the query string.](javascript:void(0)) [We know now that it's very easy to pull that value out of the query string if we need it](javascript:void(0)) [and this was a redirect result so if you're not familiar with that the browser](javascript:void(0)) [of requested slash cuisine slash Swedish and the server responded with an http code](javascript:void(0)) [and a location basically saying no you don't want this URL you want to go over here](javascript:void(0)) [so the browser had to issue another get request and come back here](javascript:void(0)) [to the home controller, that's how we ended up here.](javascript:void(0)) [You can also if you need to redirect to a route by name so here I could say redirect to route](javascript:void(0)) [and say please redirect to the default route and now I could plug in values if I wanted to,](javascript:void(0)) [I could say please pass along so that we reach the home controller and the about action,](javascript:void(0)) [so with redirect of the route you don't pass the controller](javascript:void(0)) [and action name was parameters you have to pass them in an anonymously typed object](javascript:void(0)) [but those values will match up with however the route was configured.](javascript:void(0)) [Let's try a couple other different type of routes,](javascript:void(0)) [just for kicks let's return a file result, so I can return file and here I can pass](javascript:void(0)) [in the file contents if I already have them loaded into a byte array,](javascript:void(0)) [there's different overloads of this, just pass in a stream if I already have a stream open](javascript:void(0)) [or pass in a file name the mvc framework will figure out how to pick](javascript:void(0)) [that up off disk and pass it back.](javascript:void(0)) [I'm going to pick the overload here where I just need to pass a file name and a content type.](javascript:void(0)) [So this will be a little bit weird but let's just say when the user comes](javascript:void(0)) [to the search action of the cuisine controller we want to return our site style sheet,](javascript:void(0)) [site dot css it's in the content directory.](javascript:void(0)) [So I'll need to come up with the physical path to that file, I can do that...do](javascript:void(0)) [that server property here it is one more time, this time I'm going to use the map path method](javascript:void(0)) [on that and from here what I can do is pass in a virtual path,](javascript:void(0)) [the path that's in my web application slash content slash site dot css](javascript:void(0)) [and server dot map path will turn that into a physical path](javascript:void(0)) [that the operating system understands and I can say](javascript:void(0)) [that the content type of this is text slash css.](javascript:void(0)) [Let me do a build and let's see what happens when I come back out to the browser](javascript:void(0)) [and refresh the cuisine controller this time, now what I'm looking](javascript:void(0)) [at is essentially our style sheet, site dot css, we returned that as the action result.](javascript:void(0)) [Let's try one more, let's come back and return a Json method and then pass in an object.](javascript:void(0)) [Behind the scenes what will happen is the framework will run a java script serialize](javascript:void(0)) [over this object to turn it into java script object notation.](javascript:void(0)) [And this object can be pretty much anything, it can be one of your models,](javascript:void(0)) [it can be just an anonymously typed object so I could say give me an object with message equal](javascript:void(0)) [to that current message that we computed and I'll just throw in something extra just](javascript:void(0)) [for the fun of it so I don't have a name property called Scott and I need](javascript:void(0)) [to add one more parameter here to allow this to work when I get request,](javascript:void(0)) [I need to explicitly say allow a get request to retrieve this Json and now do a build.](javascript:void(0)) [Let's come back to the browser and refresh and there you can see Json appears](javascript:void(0)) [in the browser window, message is French name of Scott.](javascript:void(0)) [Later in the course we'll see just how easy it is to take that and consume it](javascript:void(0)) [from java script then turn it into anything we want, turn it into an html display.](javascript:void(0))

* + [Action Selectors](javascript:void(0))

[When the mvc framework is selecting one of your controllers public methods to invoke](javascript:void(0)) [as an action it will use any action selector attributes that might be present](javascript:void(0)) [to find the correct action to invoke.](javascript:void(0)) [One such selector is the action name selector, when you apply this attribute](javascript:void(0)) [to a controller action it specifies the action name for the method or the edit method](javascript:void(0)) [that we see here we can no longer reach this method as an action named edit we have](javascript:void(0)) [to reach it as an action named modify, action name is an attribute you can use when you went](javascript:void(0)) [to alias, the name of your action methods.](javascript:void(0)) [Accept verbs on the other hand will specify the http verb that is allowed](javascript:void(0)) [to reach a particular action, you can say that an action method is only reachable](javascript:void(0)) [with an http get request or an http post request or delete or put](javascript:void(0)) [or some combination of those verbs.](javascript:void(0)) [The accept attribute will become extremely important later when we start modifying data,](javascript:void(0)) [we'll be using both get and post then but right now we're just using get](javascript:void(0)) [but I'm introducing this topic now so you're ahead of the game.](javascript:void(0)) [Let's see how this works in visual studio.](javascript:void(0)) [Inside of our search action let's change it back to returning a simple content result](javascript:void(0)) [with a message that we've computed](javascript:void(0)) [but I will make one more change I will add an attribute here that says](javascript:void(0)) [that you can only invoke this during an http post message and now if I do a build](javascript:void(0)) [and refresh the browser on slash cuisine I'll get the 404 error message,](javascript:void(0)) [essentially the mvc framework could not find an action it could invoke](javascript:void(0)) [in response to this get request.](javascript:void(0)) [A get request is what a browser sends off by default if you just type some something](javascript:void(0)) [into the address bar and hit enter.](javascript:void(0)) [On the other hand if I had specified http get and now do a build](javascript:void(0)) [and a refresh everything should work just fine because now the mvc framework can find an action](javascript:void(0)) [to invoke and we see French again because that's the default value for this parameter.](javascript:void(0)) [Let me give you a preview of why this is extremely useful.](javascript:void(0)) [What happens if we had two actions both with the name of search, the first problem here is](javascript:void(0)) [that the c sharp compiler will be unhappy](javascript:void(0)) [because these methods have the exact same signature, the same name or the same number](javascript:void(0)) [of parameters we wouldn't be able to get this to build but let's take the parameter off](javascript:void(0)) [and instead of returning content with the message, let me just return content](javascript:void(0)) [that says search and now do a build and come out and refresh the browser.](javascript:void(0)) [We'll get a different sort of error, the mvc framework sees](javascript:void(0)) [that it should invoke a search action but it's not going to try to distinguish](javascript:void(0)) [between the two search actions that we have, one that takes a string,](javascript:void(0)) [one that doesn't have any parameters it's just going to throw](javascript:void(0)) [up its hands and throw an exception.](javascript:void(0)) [There's some scenarios later in this course where it's going to be very useful](javascript:void(0)) [to have two different versions of an action and have one respond to an http get request only](javascript:void(0)) [and have the other one respond only to an http post request](javascript:void(0)) [and now by placing these attributes here the mvc framework can clearly see](javascript:void(0)) [that if it's a get request for search it should go to this action and display research,](javascript:void(0)) [if it's a post request for search it will come over to this one.](javascript:void(0)) [So I just did a build and if I refresh now we should get rid](javascript:void(0)) [of the error and we get a search text.](javascript:void(0)) [Again that's because the mcv framework chose this one, it was an http get request,](javascript:void(0)) [it saw the search action...it saw](javascript:void(0)) [that this first one was only legal during a post so it selected the second one.](javascript:void(0)) [Just remember this discussion when we come back to modifying data later.](javascript:void(0))

* + [Action Filters](javascript:void(0))

[In addition to the action selector attributes we just looked at there are also a number](javascript:void(0)) [of action filter attributes you can use.](javascript:void(0)) [Action filters apply pre and post processing logic to a controller action and its result.](javascript:void(0)) [Action filters are the components that you want to use to apply cross-cutting logic that's logic](javascript:void(0)) [that has to execute across multiple controller actions but you don't want](javascript:void(0)) [to duplicate code inside of individual controllers.](javascript:void(0)) [One example is the output cash attribute this tells the run time that it's allowed](javascript:void(0)) [to cash the final output of some action and to use](javascript:void(0)) [that cashed result to service future requests.](javascript:void(0)) [When you apply this attribute in the right places you can dramatically increase the](javascript:void(0)) [through put and scalability of an application,](javascript:void(0)) [we'll take a look at cashing later in this course.](javascript:void(0)) [Another example is the authorize attribute, authorize allows you to ensure a user is logged](javascript:void(0)) [in and perhaps a in a specific role like the admin.](javascript:void(0)) [role before the action is allowed to process that user's request.](javascript:void(0)) [We'll be looking at most of these filters during the duration of this course](javascript:void(0)) [but right now I just want to show you how to apply a filter and also take a quick look](javascript:void(0)) [at building your own custom action filter.](javascript:void(0)) [Here inside of visual studio let's get rid of our second search action result and go back just](javascript:void(0)) [to a single search action result, we'll have it respond to both a get and a post](javascript:void(0)) [and I'm going to use the authorize attribute.](javascript:void(0)) [You can just use the authorize attribute without any parameters, you can also specify something](javascript:void(0)) [like roles equals admin., that means the user has to be logged in and they have to be](javascript:void(0)) [in the admin role before we can invoke this function, you can also specify user names inside](javascript:void(0)) [of here, I'm just going to use the authorize attribute without any parameters,](javascript:void(0)) [what that tells the mvc framework is the user has to be logged in in order](javascript:void(0)) [to be able to use the search action.](javascript:void(0)) [Now if I do a build and I come back to refresh the application](javascript:void(0)) [and what has happened is we've been redirected to the log on page because I've tried](javascript:void(0)) [to go somewhere that requires me to be authorized, it requires me to be authenticated](javascript:void(0)) [and the run time will see authorized attribute and it's smart enough to redirect me to the log](javascript:void(0)) [on page where if I were to register on this site I could create an account and the log in.](javascript:void(0)) [After I log in I'd be redirected back to the cuisine controller where I originally tried](javascript:void(0)) [to get to, back to the original URL that I tried.](javascript:void(0)) [We'll be looking at authentication and authorization and other security topics](javascript:void(0)) [in a later module but right now I just want to focus on action filters and I want](javascript:void(0)) [to demonstrate that you can place them not only on an individual action](javascript:void(0)) [but you can also place them on a controller and when they're at the controller level they apply](javascript:void(0)) [to all the actions inside of that controller so again if I tried to get](javascript:void(0)) [to the cuisine controller now that we've rebuilt I'll still be redirected to the log on page,](javascript:void(0)) [I need to log in before I can get past that authorized gate keeper.](javascript:void(0)) [In addition to applying action filters at the action level](javascript:void(0)) [and at the controller level you can also have what's known as global filters.](javascript:void(0)) [Global filters are typically registered during application start](javascript:void(0)) [up which means we register them during the application start event to let me return](javascript:void(0)) [to the magical global dot asax dot cx file and there's the application start event one](javascript:void(0)) [of the calls that you can see in here is a call to filter config dot registered global filters,](javascript:void(0)) [I'll put the cursor right on that method, press F12 and we'll jump over to that class](javascript:void(0)) [which by the way it's also inside of the app start folder.](javascript:void(0)) [Here you can see that we're registering a single global filter, it's the handle error attribute.](javascript:void(0)) [A global filter like handle error attribute will be in effect for every single request](javascript:void(0)) [that is processed by any controller inside of your application.](javascript:void(0)) [The purpose of the handle error attribute is to display a friendly error page](javascript:void(0)) [to users when something goes wrong.](javascript:void(0)) [Let's see how this works, let me return to the cuisine controller,](javascript:void(0)) [we've commented out the authorized attributes so we should be able to get inside of this action](javascript:void(0)) [without logging in and we'll pretend](javascript:void(0)) [that something terrible has happened during the execution of our program](javascript:void(0)) [and we'll throw an exception that's going to escape the action.](javascript:void(0)) [That means if I come back to the cuisine controller, let's go to cuisine,](javascript:void(0)) [what we'll see is affectionately known as the asp dot net yellow screen of death.](javascript:void(0)) [It's the screen that shows you there was an un-handled exception on the server](javascript:void(0)) [and it gives you a stack trace and some additional information that you](javascript:void(0)) [as a developer probably will find useful because it can help you track down a problem](javascript:void(0)) [but your users don't want to see this in fact it can be dangerous](javascript:void(0)) [because it can give away information that someone who is](javascript:void(0)) [up to no good can use to try to attack your server.](javascript:void(0)) [But in actuality your users will probably never see this page because by default this level](javascript:void(0)) [of detailed error information is only shown when the request originates from the same machine](javascript:void(0)) [that the application is on so on the local host.](javascript:void(0)) [If we want to see what the user is actually going to see we just have](javascript:void(0)) [to configure our application a little bit differently.](javascript:void(0)) [Configuration is another topic that we'll cover later in this course](javascript:void(0)) [but for now let me introduce you to the web dot config files that is inside of this project,](javascript:void(0)) [this controls configuration settings for the entire application](javascript:void(0)) [and one of the things I can configure in here is how to handle errors, I want custom errors](javascript:void(0)) [and I want the mode to be on by default it is that third option there,](javascript:void(0)) [remote only which means it only shows a pretty error page to remote users.](javascript:void(0)) [I'll just type in that xml element that's inside of the system dot web section,](javascript:void(0)) [save the web dot config file and now let me come back and refresh this page,](javascript:void(0)) [the application will restart any time that you twiddle with the web dot config file](javascript:void(0)) [and this is the pretty error page that the users will see.](javascript:void(0)) [Of course pretty is in the eye of the beholder but the reason we end up here is primarily](javascript:void(0)) [because of that handle error attribute, it is providing post processing logic on an action](javascript:void(0)) [and when it sees that an exception has escaped](javascript:void(0)) [from an action it will display an error view instead of the yellow screen of death.](javascript:void(0)) [The error view is actually in your application by default you can find it if you go](javascript:void(0)) [into the solution explorer window and look under views shared,](javascript:void(0)) [views that are in this folder are available to controllers throughout the application,](javascript:void(0)) [there's an error dot see html view here so the error view is just a razor view](javascript:void(0)) [like all the other views we are using and it's inside of here](javascript:void(0)) [where you can provide some additional information or additional messages](javascript:void(0)) [or give someone the number to the help desk](javascript:void(0)) [so that they can call them in the middle of the night.](javascript:void(0)) [Let me just change that view and if I refresh we'll be able to see](javascript:void(0)) [that change reflected in the browser.](javascript:void(0)) [Let me also demonstrate that if that if we were to go into our app start filter config](javascript:void(0)) [and comment on this line it places the handle error attribute](javascript:void(0)) [into the global filter collection then after I do a build](javascript:void(0)) [and refresh we get a slightly different error, well actually it's the same error it's just](javascript:void(0)) [that asp dot net is displaying a different page it's not giving out a level of detail](javascript:void(0)) [in this error because we have custom errors mode equals on we're going](javascript:void(0)) [to see exactly what the user sees here,](javascript:void(0)) [they're not going to see a stack trace they're just going to be aware](javascript:void(0)) [that something is wrong on the server.](javascript:void(0)) [What I should do is come back and put this filter into place](javascript:void(0)) [so that we have friendly errors and also go into web dot config and change this mode](javascript:void(0)) [to remote only, that means our end users will see custom error pages](javascript:void(0)) [but when we're developing here on local host we'll still get to see stack traces](javascript:void(0)) [which can be very useful for debugging.](javascript:void(0)) [Now before we finish this module I also want to demonstrate that it's very easy](javascript:void(0)) [to build your own custom action filters.](javascript:void(0)) [I'm going to go to the filters folder that's already in this application and click add class,](javascript:void(0)) [what we're going to build is a logging attribute so I will call this log attribute,](javascript:void(0)) [that will create a class, log attribute, I'm going to derive it from action filter attribute](javascript:void(0)) [that is in the system dot web dot mvc name space and once I derive](javascript:void(0)) [from the base filter attribute class what I can do is override one of four important methods.](javascript:void(0)) [On action executing is a method you can override to look](javascript:void(0)) [at the request before an action even executes where as](javascript:void(0)) [on action executed is a method they you can override and you get to poke around inside](javascript:void(0)) [of what happened at a point in time after the action executes.](javascript:void(0)) [You can also catch before the result is executed and after the result is executed](javascript:void(0)) [so for instance before a view vendors and a view vendors](javascript:void(0)) [and these action filters are very powerful, you can change the environment,](javascript:void(0)) [you can change results you can change parameters,](javascript:void(0)) [let me just go ahead and override these methods.](javascript:void(0)) [Let's override on action executing and override on action executed, override on result executing](javascript:void(0)) [and override on result executed and then I will come in and press F9 at the top of each](javascript:void(0)) [of these methods to set a break point and now let's apply this attribute](javascript:void(0)) [to our cuisine controller and see how it behaves.](javascript:void(0)) [All we need to do that is place it on an action or at the controller level,](javascript:void(0)) [this log attribute is in a different name space so I'll need to bring in the name space ode](javascript:void(0)) [to food dot filters, but at this point I should be able to run](javascript:void(0)) [with the debugger and see where we break.](javascript:void(0)) [Let me go to slash cuisine and you can see we get inside](javascript:void(0)) [of on action executing even before we get inside the action itself](javascript:void(0)) [and one of the things I could look at inside of here is the filter context.](javascript:void(0)) [From filter context you can find out all sorts of things,](javascript:void(0)) [you can find out what parameters are being passed to an action, you can add parameters,](javascript:void(0)) [you can modify parameters, if I continue on and press F5 you can see now we're in the action,](javascript:void(0)) [we're throwing and exception, something terrible has happened, press F5 to continue again](javascript:void(0)) [and now we're in one action executed, this also has a filter context,](javascript:void(0)) [this filter context can have more information on it because now the action has executed.](javascript:void(0)) [You might have a result that you want to inspect or you might have an exception that you want](javascript:void(0)) [to inspect so here we can see that an un-handled exception escaped from that.](javascript:void(0)) [If I was really writing a log attribute I would probably record that exception somewhere,](javascript:void(0)) [I'd probably also be recording what time the controller started executing](javascript:void(0)) [and what time it finished and I could do all sorts of interesting](javascript:void(0)) [and useful things with attributes.](javascript:void(0)) [I'd probably also install this log attribute as a global filter attribute so it would execute](javascript:void(0)) [for everything inside the application but for now I just want you to know](javascript:void(0)) [that these are available when you see attributes on actions and controllers you'll have an idea](javascript:void(0)) [of what they can do and you can see how easy it is if you ever need](javascript:void(0)) [to write your own attribute, how easy that is to do.](javascript:void(0))

* + [Summary](javascript:void(0))

[This module was a stepping stone to give us some of the foundational knowledge that we need](javascript:void(0)) [to know about how mvc works before we can move on and build out our application.](javascript:void(0)) [I demonstrated how to use the routing api's to route requests to controllers](javascript:void(0)) [and how to receive parameters in my action methods that arrive in the URL and query string,](javascript:void(0)) [we also looked at action filters and different types of action results,](javascript:void(0)) [these action results will come in useful in future modules.](javascript:void(0))

* Razor Views
  + [Introduction](javascript:void(0))

[Hi. This is Scott Allen and in this module we're going to build views](javascript:void(0)) [with the Razor view engine in ASP.NET MVC.](javascript:void(0)) [The topics we'll cover in this module include the Razor syntax and how to transition](javascript:void(0)) [between c sharp code and mark up inside of a Razor view.](javascript:void(0)) [We'll also see how to use HTML helpers to keep our view simple and how to work](javascript:void(0)) [with a layout view that defines the structure of our user interface.](javascript:void(0)) [No module on views would be complete without a security discussion so we'll also talk about how](javascript:void(0)) [to avoid cross-site scripting attacks and how to use HTML encoding.](javascript:void(0)) [Finally I'll also show you how to use partial views and when they can be helpful.](javascript:void(0))

* + [Razor Basics](javascript:void(0))

[The Razor view engine allows us to use Razor templates to produce HTML.](javascript:void(0)) [Razor views if you remember from previous modules,](javascript:void(0)) [those are the files with the CHTML extension.](javascript:void(0)) [Think of them as templates that consist of mark up which is your HTML](javascript:void(0)) [and C sharp code expressions that are evaluated to place data into the mark up.](javascript:void(0)) [So if I have a view and my model is a RestaurantReview I'll take properties](javascript:void(0)) [of that review like the restaurant name and address and place those property values](javascript:void(0)) [into specific locations in the HTML.](javascript:void(0)) [When it all comes together I render a full HTML page of the client's browser](javascript:void(0)) [that gives them all the information that they want to see.](javascript:void(0)) [Let's look at this in an application to make the concept all the more concrete.](javascript:void(0)) [Imagine that in the application we want to have a place where you can come in](javascript:void(0)) [and see the latest reviews in the data base, we'll call that slash reviews.](javascript:void(0)) [We won't be working with the data base just yet, we're going to focus on views](javascript:void(0)) [but we cannot even get to a view yet because we do not have a reviews controller and as we saw](javascript:void(0)) [in the last module, that leads to a 404 error message.](javascript:void(0)) [So let me right click on the controller's folder and add a new controller.](javascript:void(0)) [We will call this the reviews controller and instead of using the empty template I'm going](javascript:void(0)) [to pick the MVC controller with empty read write actions.](javascript:void(0)) [This will give me an index method the default action for a controller](javascript:void(0)) [but it will also give me other actions that will ultimately allow me to edit a review.](javascript:void(0)) [Create a review, delete a review, none of the code will be there as yet](javascript:void(0)) [but we can fill it in as we go along.](javascript:void(0)) [It's the index action where you typically would display a list of everything.](javascript:void(0)) [But we don't even have anything that represents a RestaurantReview yet, so let me also go](javascript:void(0)) [and add a class called RestaurantReview and add this to the project.](javascript:void(0)) [We'll give it just a few simple properties.](javascript:void(0)) [Ultimately one day we'll have a data base behind this and we'll need some sort of property](javascript:void(0)) [to hold a primary key so I will add an integer ID property to the RestaurantReview.](javascript:void(0)) [We'll also give this a place to store the name of the restaurant,](javascript:void(0)) [the city that the restaurant is in, the country, and finally some sort of rating.](javascript:void(0)) [And that's all I need for a RestaurantReview.](javascript:void(0)) [One day, this will be in the data base.](javascript:void(0)) [But for right now in the reviews controller, what I'm going to do is come](javascript:void(0)) [down to the very bottom and just give us some in memory data to work with,](javascript:void(0)) [so we don't focus on data access yet, we can focus on views.](javascript:void(0)) [So I'm going to paste in some code that is just a static private list of RestaurantReview.](javascript:void(0)) [I need to bring the name space in the scope using OdeToFood.Models](javascript:void(0)) [and now this is legal C sharp code and this is not something I suggest that you usually do](javascript:void(0)) [which is keep around a static list of in memory data in your controller that never works.](javascript:void(0)) [This is just so that we don't have to get a data base involved just yet.](javascript:void(0)) [What this will allow me to do is to come back into my index action and say that my model](javascript:void(0)) [for this action is going to be from R in reviews, we'll do a little link query here,](javascript:void(0)) [let's order by R dot country and select every review.](javascript:void(0)) [And that will be my model for the view that doesn't exist just yet.](javascript:void(0)) [But even though it doesn't exist it's still educational to build this application](javascript:void(0)) [and come back to a browser and refresh to see how it behaves.](javascript:void(0)) [And what we're looking at now is not the 404 error](javascript:void(0)) [that we have before but we still do have an error.](javascript:void(0)) [We've told the MVC runtime that we want to render a view and the runtime has gone looking](javascript:void(0)) [for what we call the conventional view.](javascript:void(0)) [Since we're inside the index action of the reviews controller the MVC runtime went](javascript:void(0)) [to the reviews folder and looked for something called index,](javascript:void(0)) [that has the same name as the action.](javascript:void(0)) [And it looks for a couple of different types of index files](javascript:void(0)) [because there's actually two view engines that are registered by default in the MVC runtime.](javascript:void(0)) [One view engine is the web forms view engine and web form views have the extension ASPX or ASCX.](javascript:void(0)) [The other view engine is the Razor view engine with the CSHTML extension,](javascript:void(0)) [and since both of these view engines are around by default, the runtime is looking](javascript:void(0)) [for all these different types of files.](javascript:void(0)) [And it looks for these files in the Views slash reviews folder](javascript:void(0)) [since we are in the reviews controller.](javascript:void(0)) [The convention is to have a views folder with the same name as your controller,](javascript:void(0)) [but the runtime also went looking in the Shared folder because views on the Shared folder are](javascript:void(0)) [as you might have guessed from the name, shared among all of the controllers.](javascript:void(0)) [But the runtime didn't find that view anywhere, we need to create that view.](javascript:void(0)) [An easy way to add a view is just to right click inside of a controller action](javascript:void(0)) [like this Index action and select the Add View.](javascript:void(0)) [You can also hit control M control V. This will launch a dialog box](javascript:void(0)) [and the dialog is rather smart, it knows on inside the index action.](javascript:void(0)) [So it wants to create an index view which is perfect.](javascript:void(0)) [I do want to use the Razor view engine and I want to create a strongly-typed view.](javascript:void(0)) [Remember in the introduction, we talked about strongly-typed views.](javascript:void(0)) [This is where the view knows the exact type of model object that you are passing into it.](javascript:void(0)) [I'm going to tell it that my model is a RestaurantReview.](javascript:void(0)) [In fact, I'm going to have a list of RestaurantReviews.](javascript:void(0)) [And I'm also going to pick a scaffolding template of list.](javascript:void(0)) [What scaffolding in Visual Studio will do is reflect upon your model object,](javascript:void(0)) [figure out what properties are interesting and then build out a basic UI for you.](javascript:void(0)) [It will create a view and it's something that you can go in and modify and customize.](javascript:void(0)) [But it gives you a starting point and the list scaffold template assumes](javascript:void(0)) [that you have multiple model objects that you need to display.](javascript:void(0)) [There's also a scaffolding to create, and edit, and delete,](javascript:void(0)) [we'll be looking at those in this course too.](javascript:void(0)) [I'm going to leave all the other settings in place and just click Add,](javascript:void(0)) [and you'll see what happens as we now have a Reviews folder in the views folder.](javascript:void(0)) [That reviews folder has an index view.](javascript:void(0)) [The index view itself is strongly-typed against the IEnumerable of RestaurantReview](javascript:void(0)) [so it expects a collection of RestaurantReviews.](javascript:void(0)) [And it's going to build a table to display those multiple reviews.](javascript:void(0)) [The table will have header cells with the name the City, the Country, the Rating,](javascript:void(0)) [and then for each item in the model, we'll be writing out a table row with cells](javascript:void(0)) [that show the name, the City, the Country, the Rating and then a whole bunch of links.](javascript:void(0)) [We're going to come back and talk about the syntax and what this HTML thing is.](javascript:void(0)) [For now, let me just save everything, do a build to make sure everything is up to date](javascript:void(0)) [and I should be able to refresh my view.](javascript:void(0)) [And now, we have a list of reviews on the screen.](javascript:void(0))

* + [Code Expressions](javascript:void(0))

[Inside of a Razor view like this index view for RestaurantReviews our responsibility is](javascript:void(0)) [to take the model object given to this by the controller](javascript:void(0)) [and present the model in a user interface.](javascript:void(0)) [And the user interface in the web application is generally built using HTML.](javascript:void(0)) [As we talked about earlier, you can think of Razor as a template,](javascript:void(0)) [where you combine literal text with pieces of data from your model](javascript:void(0)) [that you pull out using sharp C sharp code.](javascript:void(0)) [The literal text is like the h2 and Index text too in this view.](javascript:void(0)) [If I want a view to display a more descriptive header, I can change this text.](javascript:void(0)) [Let's say that this will display, the Latest Reviews.](javascript:void(0)) [I can just save a view, refresh the browser and I can see that literal text appears here.](javascript:void(0)) [Razor just output that without changing anything, the text gets sent to the browser](javascript:void(0)) [where the h2 element becomes a level two header, so it gives this a bit of a title effect here](javascript:void(0)) [to announce that these are the Latest Reviews.](javascript:void(0)) [When you see an at sign in Razor, chances are then you are not dealing with literal text](javascript:void(0)) [because an at sign can introduce C sharp code](javascript:void(0)) [and the C sharp code can contain an expression for Razor to evaluate.](javascript:void(0)) [So I can use C sharp code to reach into my model and pull out data.](javascript:void(0)) [I could have something like we are showing the latest at Model.Count reviews.](javascript:void(0)) [Save this, put parenthesis because count is a link operator,](javascript:void(0)) [it needs to be invoked and now refresh the view.](javascript:void(0)) [And we get-- we are showing the latest 3 reviews.](javascript:void(0)) [So we didn't output at Model.Count, instead Razor saw that that was C sharp code,](javascript:void(0)) [it evaluated it, it produced the number 3, so it sent that number 3 down to the browser.](javascript:void(0)) [And technically, you don't need to only access the model in the view.](javascript:void(0)) [You could also access ViewBag.](javascript:void(0)) [In fact, you could access anything.](javascript:void(0)) [Let me just put out a property that is inherited by a view.](javascript:void(0)) [The views VirtualPath and save this.](javascript:void(0)) [And when I do that I can see](javascript:void(0)) [that the VirtualPath property contains the path of the view in this application.](javascript:void(0)) [So I don't need to stick to just the model property.](javascript:void(0)) [I can go anywhere to pick up data but remember there is no data access or calculation logic](javascript:void(0)) [in the view at least not heavy calculation logic.](javascript:void(0)) [Because data access in calculation logic, that's what controllers and models do.](javascript:void(0)) [And while I'm looking at it,](javascript:void(0)) [a tilde is something we've seen before but I'm not sure I explained it.](javascript:void(0)) [A tilde represents the root of the application so tilde slash Views would mean go](javascript:void(0)) [to the view folder from the root of this application.](javascript:void(0)) [It's common to use a tilde everywhere you specify a path](javascript:void(0)) [because without the tilde you need to be careful of your app](javascript:void(0)) [that gets deployed into a subdirectory.](javascript:void(0)) [You have to know the name of the directory where the app is deployed](javascript:void(0)) [in order to hand out correct links.](javascript:void(0)) [And that gets very messy.](javascript:void(0)) [So anytime you need to point to something in the application whether it's a JavaScript file](javascript:void(0)) [or an image or another view, using the tilde is a safe and easy approach.](javascript:void(0)) [Let's go back to the code we have before.](javascript:void(0)) [I'll hit Control Z a couple times and we're back to at Model.Count.](javascript:void(0)) [You can think of the at sign here a bit like response dot right if you've ever done](javascript:void(0)) [that in ASP.NET or classic ASP, the result of the C Sharp expression Model.Count is sent](javascript:void(0)) [down to the client essentially with a response dot right but there is a significant difference](javascript:void(0)) [between this code and response dot right](javascript:void(0)) [because Razor will automatically HTML encode any output sent through the at sign](javascript:void(0)) [to help prevent cross-site scripting attacks.](javascript:void(0)) [Let me demonstrate this.](javascript:void(0)) [If we switch over into the ReviewsController and come down to our data that we're displaying,](javascript:void(0)) [what we could do is change the city for the Marrakesh restaurant from Washington D.C.](javascript:void(0)) [to something like script alert XSS which stands for cross-site scripting attack slash script.](javascript:void(0)) [So what this represents is what would happen if somehow, someway a malicious user was able](javascript:void(0)) [to get something into your database that was a malicious script.](javascript:void(0)) [And if I build the application and refresh the browser, what we'll see is that Marrakesh,](javascript:void(0)) [the city is now script with an alert box.](javascript:void(0)) [Now an alert box is just an annoyance.](javascript:void(0)) [But if someone actually did get a malicious script into the database](javascript:void(0)) [and Razor was not HTML encoding this,](javascript:void(0)) [the browser would actually see a script tag and execute that JavaScript.](javascript:void(0)) [Cross-site scripting attacks can do terrible things.](javascript:void(0)) [They can steal cookies from users,](javascript:void(0)) [they can display fake password dialogs and steal user's passwords.](javascript:void(0)) [Cross-site scripting attacks are very dangerous](javascript:void(0)) [and unfortunately they are the number one vulnerability on the web.](javascript:void(0)) [So you have to be very careful when displaying data that the user has given to you](javascript:void(0)) [and make sure you HTML encode that data which again fortunately Razor does that by default.](javascript:void(0)) [We can see that if we do a View Source here and I do a quick search for XSS,](javascript:void(0)) [instead of actually outputting a less than character which would allow the browser to see](javascript:void(0)) [that this is a script tag and start executing JavaScript, instead Razor converted that less](javascript:void(0)) [than character into ampersand LT colon which is a way to tell the browser I want](javascript:void(0)) [to display a less than character.](javascript:void(0)) [And so now what we have is a script that's showing up in the city which is a lot better](javascript:void(0)) [than actually having executable script.](javascript:void(0)) [And just to show you what happens if you really want to display something](javascript:void(0)) [without HTML encoding, you can do that in Razor.](javascript:void(0)) [What we'll do is we'll come down to this line of code that is displaying the city and we'll talk](javascript:void(0)) [about what HTML.Display for does in just a bit.](javascript:void(0)) [But the way to do this in Razor is to use HTML.Raw,](javascript:void(0)) [that's a way of saying take this expression, take the string, do not HTML encode it.](javascript:void(0)) [So if I just put item.City in here, save everything and refresh the browser.](javascript:void(0)) [Now we have actual executable script on the page which in this case is an annoyance](javascript:void(0)) [if that really was a malicious script, terrible things could happen.](javascript:void(0)) [So let me quickly get rid of that script.](javascript:void(0)) [I'm into the ReviewsController, just set Control Z to get back to Washington D.C.](javascript:void(0)) [And now we can move forward again.](javascript:void(0)) [And now, let's look at item.Rating.](javascript:void(0)) [I'm going to comment out this line of code because we're not quite ready](javascript:void(0)) [to talk about what display for does.](javascript:void(0)) [I just want to show you that I can also output the item rating juts by saying at item.Rating.](javascript:void(0)) [And if I save everything and rebuild, we've gotten rid of our script](javascript:void(0)) [but we should still be showing the rating as a 10 which is correct.](javascript:void(0)) [And the at item.Rating is what we will call an implicit code expression in Razor.](javascript:void(0)) [We'd let Razor figure out what a C Sharp code and what is mark up.](javascript:void(0)) [It's smart enough to know that at item.Rating is a C Sharp expression whereas less](javascript:void(0)) [than slash TD are going to back into mark up.](javascript:void(0)) [But what if we have some text displaying right after this rating?](javascript:void(0)) [So the rating is on the scale of 1 to 10 but if we wanted to let someone know](javascript:void(0)) [that by saying 10 out of 10 or 3 out of 10.](javascript:void(0)) [I just put a slash 10 here.](javascript:void(0)) [There's two ways to interpret this.](javascript:void(0)) [You could say take the rating and divide it by 10 because this is all a C Sharp expression](javascript:void(0)) [or show slash 10 as literal text.](javascript:void(0)) [And by default what Razor will do is figure that that is text,](javascript:void(0)) [not part of the C Sharp expression, so we get 10 out of 10.](javascript:void(0)) [But what if we really did want to take the rating and divide it by 10?](javascript:void(0)) [Then we would need what's known as an explicit code expression where you use parenthesis](javascript:void(0)) [and the at sign to explicitly denote what your C Sharp code is.](javascript:void(0)) [And if I'll save this version of the view, all the ratings turn into 1](javascript:void(0)) [because we're taking 10 and dividing it by 10.](javascript:void(0)) [But for the most part, Razor is really good at figuring](javascript:void(0)) [out what is C Sharp code and what is not C Sharp code.](javascript:void(0)) [If I turn this back into an implicit expression, let me just show you a couple other edge cases.](javascript:void(0)) [Imagine I wanted to prefix the rating with an R. So if I put the letter R here and we want](javascript:void(0)) [to have an R10 or an R3 and I save this and refresh,](javascript:void(0)) [what we see is at item.Rating is now output as literal text.](javascript:void(0)) [That's because Razor makes a special case for e-mail addresses and when it sees something](javascript:void(0)) [at something dot something, it assumes you might want to display an e-mail address](javascript:void(0)) [so it doesn't evaluate this as C Sharp code.](javascript:void(0)) [The nice thing about this is if you actually have an e-mail address](javascript:void(0)) [and you put it there something like scottallen@Pluralsight.com,](javascript:void(0)) [you don't need to worry about escaping anything.](javascript:void(0)) [But in this scenario where I really do want to say something like R1 or R10,](javascript:void(0)) [I need to go back to using an explicit code expression](javascript:void(0)) [and surrounding my C Sharp there with parenthesis.](javascript:void(0)) [Now that will be treated as C Sharp code and I get the R10 out.](javascript:void(0)) [So Razor is smart enough to figure out e-mail addresses](javascript:void(0)) [because of the presence of the at sign.](javascript:void(0)) [Well what happens if I want to put out something that has an at sign in it like a Twitter handle?](javascript:void(0)) [Now this Razor handle this.](javascript:void(0)) [Well, it thinks at OdeToCode should be a C Sharp expression.](javascript:void(0)) [It goes off looking for something called OdeToCode but it doesn't find one.](javascript:void(0)) [This would a case where I just need to escape the at sign by using a double at sign.](javascript:void(0)) [And now if I save the view, now we're outputting at OdeToCode literally](javascript:void(0)) [which is what we intended to do for a Twitter handle.](javascript:void(0))

* + [Code Blocks](javascript:void(0))

[In addition to code expressions, both implicit and explicit, Razor also supports code blocks.](javascript:void(0)) [We can see an example of this here at the top of the view.](javascript:void(0)) [You can see a block of code that starts with the at sign](javascript:void(0)) [and has an opening curly brace and a closing curly brace.](javascript:void(0)) [In between those curly braces, we can place as much C Sharp code as we want inside of a view.](javascript:void(0)) [But remember, keep the view simple.](javascript:void(0)) [So a lot of code is usually a bad sign.](javascript:void(0)) [The C Sharp code that is inside of this block is going to execute.](javascript:void(0)) [It's not going to have its expressions evaluated and written to the client.](javascript:void(0)) [It simply executes or produces side effects.](javascript:void(0)) [We can set things on the ViewBag like you can see us setting the ViewBag.Title property.](javascript:void(0)) [We'll see the significance of that line of code in just a little bit](javascript:void(0)) [when we talk about layout views with Razor.](javascript:void(0)) [I can even declare variables and use them inside of the code block](javascript:void(0)) [so I could say let's create a variable called firstReview that is equal to Model.First.](javascript:void(0)) [So it literally gave me the first item in the collection that is my model.](javascript:void(0)) [And inside of the code block, you have](javascript:void(0)) [to observe the correct usage of semicolons for C Sharp codes.](javascript:void(0)) [We'll terminate that with a semicolon.](javascript:void(0)) [Now we can use firstReview anywhere else inside of this view.](javascript:void(0)) [So at the top of the view, I could now say firstReview.Name,](javascript:void(0)) [we'll up the name of the restaurant.](javascript:void(0)) [And you can see we get the name of the restaurant, the first restaurant](javascript:void(0)) [and all of the reviews here at the top.](javascript:void(0)) [Another example of a code block is the foreach statement here in the middle of the view.](javascript:void(0)) [This foreach statement allows us to iterate](javascript:void(0)) [through the model pulling out one item at a time.](javascript:void(0)) [And you can see that Razor allows us to transition](javascript:void(0)) [between C Sharp code before each statement and back into HTML without making any explicit marks](javascript:void(0)) [or using any sort of delimiters to tell it when we're transitioning from C Sharp code to HTML.](javascript:void(0)) [We simply have the opening curly brace and then we have a TR to start the table row.](javascript:void(0)) [It works in the other direction too.](javascript:void(0)) [So at the bottom of the foreach, we have a closing curly brace.](javascript:void(0)) [Razor is smart enough to figure out that the slash TR at the bottom is HTML](javascript:void(0)) [but that closing curly brace is part of my C Sharp code](javascript:void(0)) [that terminates the foreach statement.](javascript:void(0)) [Let me demonstrate a few other features of code blocks by removing this table](javascript:void(0)) [and we'll rewrite this a different way.](javascript:void(0)) [I'll remove the table.](javascript:void(0)) [I'm also going to remove the bit about a firstReview to clean up this view a little bit.](javascript:void(0)) [And now here above the create link, let's write a new each statement.](javascript:void(0)) [So for each review that we have in the model, let's-- instead of writing out a table,](javascript:void(0)) [let's write out a div for each review.](javascript:void(0)) [And in the div we'll have an h4 that has the name of the restaurant.](javascript:void(0)) [We'll include a span that has the rating associated with that restaurant.](javascript:void(0)) [And then perhaps just a paragraph tag that has the city--](javascript:void(0)) [And the country.](javascript:void(0)) [And then, for one last touch, let me go in to the div and give it a class of review](javascript:void(0)) [and I'll come in to our style sheet for the Site.css.](javascript:void(0)) [And down here at the bottom, I'm just going to paste](javascript:void(0)) [in some code that will pretty up our review.](javascript:void(0)) [Give it a bottom border, add some color to the h4 tag that we've added just](javascript:void(0)) [to make it appear a little bit better.](javascript:void(0)) [I'll save the style sheet.](javascript:void(0)) [If I'll go save everything and build and let's refresh,](javascript:void(0)) [we're still displaying our reviews now, we're just not using a table.](javascript:void(0)) [We're using a series of divs.](javascript:void(0)) [And now I'd like to show you a couple things about code blocks.](javascript:void(0)) [For the most part, when I wrote that code block, I didn't have to worry](javascript:void(0)) [about when I was writing C Sharp code and when I was writing HTML.](javascript:void(0)) [I know I have to use an at sign when I want to write a C Sharp expression and I just do that.](javascript:void(0)) [But you'll notice that a comma became part of my literal text that's outputted in the view.](javascript:void(0)) [Again, no explicit delimiters there.](javascript:void(0)) [But again, there's a couple special edge cases here.](javascript:void(0)) [For instance, what if I wanted the text review to appear before this div](javascript:void(0)) [and not inside of its own HTML element?](javascript:void(0)) [So if I just put review here and save the view and refresh, Razor gets a little bit unhappy](javascript:void(0)) [and essentially that's because it didn't make the transition](javascript:void(0)) [from C Sharp code back into literal text.](javascript:void(0)) [It thinks that review is part of my C Sharp code.](javascript:void(0)) [The easiest way to fix this, if you really don't want review inside of some sort of HTML element,](javascript:void(0)) [is just to use at colon that tells Razor that this is literal text.](javascript:void(0)) [And now if I refresh, we'll have the word review in front of every div.](javascript:void(0)) [The same thing could be used if we, for some obscure reason,](javascript:void(0)) [wanted to have a closing curly brace underneath of every div and have](javascript:void(0)) [that actually display in the view.](javascript:void(0)) [Fairly, if I refresh that doesn't display.](javascript:void(0)) [Razor thinks this is part of C Sharp code](javascript:void(0)) [and it thinks this second closing curly brace is literal text](javascript:void(0)) [so that appears at the bottom of the view.](javascript:void(0)) [I wanted this curly brace to be part of my text.](javascript:void(0)) [Again just use at colon and now we have the closing curly brace appearing where we wanted.](javascript:void(0)) [But those are edge cases you probably won't run into this very often, for the most part,](javascript:void(0)) [just write what you feel in Razor, it will figure out the proper thing to do.](javascript:void(0))

* + [Layout Views](javascript:void(0))

[Next, let's turn our attention to what was not in our index view or restaurant reviews.](javascript:void(0)) [What was not in our index view was the navigational menu that you see](javascript:void(0)) [in the application, at home, about and contact links or the Ode to Food logo](javascript:void(0)) [or the copyright at the bottom of the page.](javascript:void(0)) [All the code and mark up in our index view was there only](javascript:void(0)) [to output the review information into the middle of the page.](javascript:void(0)) [As I alluded to in the introduction, there is another type of view at work here](javascript:void(0)) [to handle these common things that appear everywhere.](javascript:void(0)) [Every application has some common UI structure.](javascript:void(0)) [You need navigation links and headers on every page and maybe a footer.](javascript:void(0)) [To provide that structure in Razor, we use a layout view.](javascript:void(0)) [If you're familiar with ASP.NET web forms, then you have probably used master pages.](javascript:void(0)) [Razor layout views are similar to master pages but really, they're much,](javascript:void(0)) [much better and easier to understand.](javascript:void(0)) [In this layout views, that's typically where you have your doc type, your head tag](javascript:void(0)) [and all the common markup that you need.](javascript:void(0)) [And then a layout view has two special methods that I can call render body and render section.](javascript:void(0)) [These methods allow the content views like the index view that we wrote to plug in their pieces](javascript:void(0)) [of the UI into specific locations of the layout.](javascript:void(0)) [Let's look at an example.](javascript:void(0)) [Inside of the index view that we've been working on, let me remove some of the experiments](javascript:void(0)) [that we've done with code blocks and save the view and refresh.](javascript:void(0)) [And just point out again that the only thing we're really rendering here is the middle](javascript:void(0)) [section, the latest reviews.](javascript:void(0)) [All the other pieces like the logo and the menu, those are handled by the layout view.](javascript:void(0)) [You can find the layout view if you go into the Solution Explorer, go into view Shared folder](javascript:void(0)) [and here is underscore layout.csHTML.](javascript:void(0)) [There's nothing special about that underscore.](javascript:void(0)) [Many developers use underscore just as a convention to identify views](javascript:void(0)) [that are not primary content views like the index view that we wrote.](javascript:void(0)) [It's not a view you would typically return as a view result from a controller.](javascript:void(0)) [But the layout view itself is really nothing special.](javascript:void(0)) [For the most part, it's just another Razor view.](javascript:void(0)) [It has literal text that can have C Sharp code expressions, it can have code blocks.](javascript:void(0)) [It's where you typically have your doc type.](javascript:void(0)) [It's where you want to have your head tag](javascript:void(0)) [and you typically have a definition for the body element.](javascript:void(0)) [And any changes that I make here will be reflected across the entire application](javascript:void(0)) [because in this application, this is the only layout view that we're using.](javascript:void(0)) [You can have more than one but we may now have only one.](javascript:void(0)) [I can demonstrate that if I collapse this header section](javascript:void(0)) [and then let me just use Control K, Control C to comment that out.](javascript:void(0)) [I'll save the view and refresh.](javascript:void(0)) [And you can see not only did our restaurant review is changed but also if I was able](javascript:void(0)) [to navigate to the homepage, it's also missing that section now too.](javascript:void(0)) [To undo commenting that out and scroll down here a little bit,](javascript:void(0)) [and here is where you can see the call to a RenderBody.](javascript:void(0)) [When a layout view calls RenderBody, that is on the content view,](javascript:void(0)) [our Index.csHTML that is one it will render and output it to HTML](javascript:void(0)) [and that HTML will be inserted right here where RenderBody is.](javascript:void(0)) [One obvious question at this point might be,](javascript:void(0)) [how does the MVC runtime know to use this layout view?](javascript:void(0)) [We're not saying anything inside of the layout view that would be particular](javascript:void(0)) [to telling everyone that this is the layout view for the application.](javascript:void(0)) [We're certainly not doing anything inside of Index.csHTML that indicates](javascript:void(0)) [that that's the layout view we're using.](javascript:void(0)) [And the answer to this question actually comes](javascript:void(0)) [from a small magical file called underscore ViewStart.csHTML.](javascript:void(0)) [You'll find it in the root of your Views folder.](javascript:void(0)) [When I open that file up you can see there's a code block inside of here](javascript:void(0)) [that sets a layout property equal to tilde slash Views slash Shared slash](javascript:void(0)) [underscore Layout.csHTML.](javascript:void(0)) [So the secret here is that underscore ViewStart.csHTML has the ability](javascript:void(0)) [to execute this code before my view starts rendering and it sets this property.](javascript:void(0)) [And this is just a convention with the Razor view engine.](javascript:void(0)) [If you have a file called underscore ViewStart.csHTML,](javascript:void(0)) [anything you put in that code block at the top will be able to execute before the view does.](javascript:void(0)) [And this works in a hierarchy.](javascript:void(0)) [So if ViewStart.csHTML is in the root of the Views folder,](javascript:void(0)) [it applies to all the views that are inside of here.](javascript:void(0)) [But if I make a copy of this file and put it inside of Reviews,](javascript:void(0)) [let's open up this copy underscore ViewStart.csHTML](javascript:void(0)) [and change this over to Layout2.](javascript:void(0)) [Now I'll save everything and refresh the home page.](javascript:void(0)) [We have our header section back and this works normally.](javascript:void(0)) [But if I go to reviews, we'll get a runtime error because we've specified that the Views](javascript:void(0)) [and the Reviews folder should use a different layout file,](javascript:void(0)) [underscore Layout.csHTML and that doesn't exist.](javascript:void(0)) [So in order for this to work I don't have](javascript:void(0)) [to have underscore Layout.csHTML in the shared folder.](javascript:void(0)) [So let me delete this file.](javascript:void(0)) [First, let me make a copy of this.](javascript:void(0)) [And also I'll show you that you can overwrite this on a per view basis.](javascript:void(0)) [So if I open up our index view that's showing the latest reviews,](javascript:void(0)) [I can also specify the layout file here and let's just change it](javascript:void(0)) [to Layout3.csHTML so we can see a difference.](javascript:void(0)) [Save everything and refresh.](javascript:void(0)) [Now it's looking for Layout3.](javascript:void(0)) [I can also turn off a layout view here if I wanted to.](javascript:void(0)) [I could just say layout.](javascript:void(0)) [I don't want to use a layout page.](javascript:void(0)) [I'll set that equal to null.](javascript:void(0)) [And now it's working but we're also not using a layout page](javascript:void(0)) [so we do not have any style sheets applied and the view looks very plain and simple.](javascript:void(0)) [Let's go back to using the default layout view.](javascript:void(0)) [Now inside of a layout view, in addition to RenderBody](javascript:void(0)) [which is a required call somewhere inside of a layout view you must call RenderBody.](javascript:void(0)) [You can also call another method called RenderSection.](javascript:void(0)) [This is optional.](javascript:void(0)) [You can have one or more sections.](javascript:void(0)) [And what RenderSection will do is provide a content view like Index.csHTML,](javascript:void(0)) [the ability to plug in content into other sections of the page.](javascript:void(0)) [So this RenderSection is rendering a section called featured.](javascript:void(0)) [There's also another RenderSection down here at the bottom of the layout view](javascript:void(0)) [that will allow a content page to inject scripts at the bottom of the page.](javascript:void(0)) [We'll come back to that later when we talk about Ajax.](javascript:void(0)) [But right now let's talk about this RenderSection](javascript:void(0)) [which is currently not a required section](javascript:void(0)) [that means a content view can have this section that cannot.](javascript:void(0)) [If I switch this over to be a required section,](javascript:void(0)) [what will happen when we render this Reviews view is](javascript:void(0)) [that we get a runtime error saying you did not have this section called featured.](javascript:void(0)) [I want to change this back to being an optional section but we're also going to implement](javascript:void(0)) [that section inside of Index.csHTML.](javascript:void(0)) [Let's just take this bit of markup that we already have and put it inside of a section.](javascript:void(0)) [We can define these sections anywhere.](javascript:void(0)) [You just have to give them the name.](javascript:void(0)) [That name has to match what you specified in RenderSection.](javascript:void(0)) [And let's put that we are showing the latest X number of reviews inside](javascript:void(0)) [of the featured section and refresh.](javascript:void(0)) [And you can see that appears at the top of the view.](javascript:void(0)) [But inside of layout, if for some reason I wanted to show a featured content at the bottom](javascript:void(0)) [of the page, let's say in the footer, I could also work.](javascript:void(0)) [So let me move that section back up to the top.](javascript:void(0)) [And now that I'm looking at it, what would be really nice is that when I'm on the home page](javascript:void(0)) [of the application, I'd have a link here to go](javascript:void(0)) [to that ReviewsController instead of typing reviews into the URL.](javascript:void(0)) [That's easy enough.](javascript:void(0)) [This is the navigational menu that the layout page specifies.](javascript:void(0)) [Let me make a copy of that last action link and create a new action link.](javascript:void(0)) [We've looked at this before.](javascript:void(0)) [This is the text I want to display, Reviews.](javascript:void(0)) [This is the action that I want to reach from some controller, the Index action.](javascript:void(0)) [And this is the name of the controller that I want to reach, the ReviewsController.](javascript:void(0)) [And now if I save everything and refresh, we'll have a Reviews link](javascript:void(0)) [that will appear throughout the application.](javascript:void(0)) [I can click on it and we end up on the ReviewsController.](javascript:void(0))

* + [HTML Helpers](javascript:void(0))

[The action link helper that we just used is one](javascript:void(0)) [of many HTML helpers available inside of the view.](javascript:void(0)) [The purpose of an HTML helper is to make it easy to create small blocks of HTML.](javascript:void(0)) [There are helpers to create links like action link but there's also helpers to create inputs,](javascript:void(0)) [validation messages, labels and more.](javascript:void(0)) [Many of these helpers are quite intelligent.](javascript:void(0)) [For example, in these codes on the screen, we're calling HTML.EditorFor and passing](javascript:void(0)) [in an expression that references a property on the model, the FirstName property.](javascript:void(0)) [EditorFor can walk up to that property and examine its type,](javascript:void(0)) [determined that the FirstName property is a string and therefore what it needs](javascript:void(0)) [to do is emit an input type equals text.](javascript:void(0)) [If instead we were using EditorFor against the Boolean property,](javascript:void(0)) [then EditorFor could produce a checkbox input.](javascript:void(0)) [There's actually a number of other things that EditorFor will do.](javascript:void(0)) [We'll explore this as we move throughout the rest of the course.](javascript:void(0)) [But all of these methods like EditorFor, LabelFor, ValidationSummary,](javascript:void(0)) [they're available from an HTML property that a view inherits.](javascript:void(0)) [That's another reason we call them HTML helpers](javascript:void(0)) [and it's also the reason we access them using HTML dot.](javascript:void(0)) [Let's use a few of these in our application.](javascript:void(0)) [Inside of our Index view, lets' give ourselves the ability](javascript:void(0)) [to edit any particular restaurant review because the edit scenario is going](javascript:void(0)) [to involve a lot more HTML helpers than what we've seen so far.](javascript:void(0)) [So let me add a span inside of a div for each review,](javascript:void(0)) [a span that's going to be aligned to the right.](javascript:void(0)) [And inside of here we will use HTML.ActionLink to display a link that displays the text edit](javascript:void(0)) [and we'll go to the Edit action on this current controller.](javascript:void(0)) [I also need to pass along some information so that the Edit action knows](javascript:void(0)) [which review I'm trying to edit and I can do that by passing along the ID of the review.](javascript:void(0)) [So I just need to say item.Id.](javascript:void(0)) [The ID property then is wrapped up in an anonymous object.](javascript:void(0)) [And you can put as many additional values as you want inside that anonymous object,](javascript:void(0)) [that get passed along to the routing engine](javascript:void(0)) [and essentially say here's some additional information that needs to go in the URL.](javascript:void(0)) [The routing engine is going to look at ID and say, oh, I can see where I can put ID actually](javascript:void(0)) [into the URL, so we should come out with something like slash reviews, slash edit,](javascript:void(0)) [slash three to edit the review with an ID of three.](javascript:void(0)) [And let's test it out, I'll save the view, refresh the browser.](javascript:void(0)) [There's our Edit link.](javascript:void(0)) [It looks correct.](javascript:void(0)) [Slash review, slash edit, slash three.](javascript:void(0)) [And when I click on that, we already have an edit controller action](javascript:void(0)) [because when we scaffold it out this controller, we said that we wanted a controller](javascript:void(0)) [with the ability to update reviews.](javascript:void(0)) [But what we don't have yet is a view itself.](javascript:void(0)) [But that's okay because we also need to add a little bit more logic here into the edit action.](javascript:void(0)) [This is the edit action that responds to that Get request.](javascript:void(0)) [The ID that's in the URL will automatically be extracted](javascript:void(0)) [by the runtime and passed into our action.](javascript:void(0)) [All I need to do is go out to our data source and find that review that we want to edit.](javascript:void(0)) [So I can say that the review, it goes reviews give me the single review.](javascript:void(0)) [Or I'm given a review R dot ID equals that incoming ID.](javascript:void(0)) [So just a little link query to say give me single object out of this collection](javascript:void(0)) [that matches this criteria, its ID property has to match this incoming ID parameter](javascript:void(0)) [and that is the model for our view.](javascript:void(0)) [Now we actually need an edit views but we don't have this error screen over here.](javascript:void(0)) [I can right click inside the edit action say please add a view for me.](javascript:void(0)) [We do want it to be called the edit view, we do want to use Razor.](javascript:void(0)) [It's going to be strongly-typed against the RestaurantReview.](javascript:void(0)) [All of this is correct.](javascript:void(0)) [But I do want to scaffold out this time an edit view.](javascript:void(0)) [Now when I click Add, I'll have a brand new edit view.](javascript:void(0)) [And what I should be able to do now is just do a quick build](javascript:void(0)) [and refresh slash review slash edit slash three.](javascript:void(0)) [And now I have a form that will let me edit all these values.](javascript:void(0)) [So let's take a step back and see what some of these HTML helpers are doing.](javascript:void(0)) [And one easy way to do that is to go in and view the source for this](javascript:void(0)) [and compare it to what is inside of our view.](javascript:void(0)) [Let me scroll down a bit.](javascript:void(0)) [And now we can start to match some things up.](javascript:void(0)) [So HTML.BeginForm writes out an opening form tag.](javascript:void(0)) [Without passing any additional parameters into begin form, what it will do is emit](javascript:void(0)) [that form tag, it will set the action of that form tag to go to the same URL that we came](javascript:void(0)) [from slash Review slash Edit slash 3 but to do a post back when the user clicks on the button](javascript:void(0)) [to submit this form and save data.](javascript:void(0)) [Begin form is useful because you can pass in additional parameters](javascript:void(0)) [to change that URL, change the method.](javascript:void(0)) [It will take care of emitting the correct HTML.](javascript:void(0)) [A little further down we have a HTML.HiddenFor.](javascript:void(0)) [What this helper will do is it will emit an input type equals hidden into the form](javascript:void(0)) [so we're storing the ID value in the form.](javascript:void(0)) [We're just not giving the user the ability to edit that value, it's a hidden input.](javascript:void(0)) [And down a little bit further we have HTML.LabelFor.](javascript:void(0)) [Labels are great for accessibility.](javascript:void(0)) [And this is simply building a label attribute where the](javascript:void(0)) [for attribute is equal to the name of the property.](javascript:void(0)) [So the name is name-- or the name of the city, in that case will build label for city.](javascript:void(0)) [And then we have editor for HTML.EditorFor is literally saying I want an editor](javascript:void(0)) [for this property.](javascript:void(0)) [Now you go out and figure out what the best type of editor for.](javascript:void(0)) [Maybe that editor will be an input type equals text](javascript:void(0)) [or an input type equals number or a checkbox or radio button.](javascript:void(0)) [In fact if we scroll down a little bit further, we can see that the input](javascript:void(0)) [for a rating is type equals number.](javascript:void(0)) [That's a new input field for HTML 5 on browsers that are up-to-date with the standards,](javascript:void(0)) [they'll know that the user is supposed to input a number here not just some random string](javascript:void(0)) [like A B C. There's also a lot of data dash attributes in here](javascript:void(0)) [like data dash val, data dash val dash number.](javascript:void(0)) [Those are providing data for client side validation.](javascript:void(0)) [We'll talk about that later in this course.](javascript:void(0)) [But for now, I just want to point out that this is what HTML helpers are all about.](javascript:void(0)) [We could certainly for the rating come in and all by ourselves say](javascript:void(0)) [that we just want an input type equals number and set the value to at Model.Rating](javascript:void(0)) [but EditorFor will do that for us.](javascript:void(0)) [And also populate the ID and the name attributes so that the editor](javascript:void(0)) [of the input matches that property name of our model.](javascript:void(0)) [And if you remember back in our discussion in the last module, we talked about the way](javascript:void(0)) [that the MVC has a way of matching up things that we need by name.](javascript:void(0)) [So if we see that we need an ID parameter as a parameter to our action method, it will go out](javascript:void(0)) [and look for something called ID and it can look](javascript:void(0)) [in the routing data, it can look in the query string.](javascript:void(0)) [It can also look in these posted form values.](javascript:void(0)) [So if I need a rating for my model something that the user has typed into the screen](javascript:void(0)) [and then click save, the MVC framework can automatically find that rating and move it](javascript:void(0)) [into a property for me just based on a naming convention.](javascript:void(0)) [We'll see that here in just a second.](javascript:void(0)) [But let me close the source code view and implement what should happen](javascript:void(0)) [when the user clicks on the save button.](javascript:void(0)) [In that case that's going to be an HTTP post message.](javascript:void(0)) [This is also a topic we talked about in the last module.](javascript:void(0)) [That will end up not at this edit action because this responds to a get request.](javascript:void(0)) [It will end up at this edit action that explicitly advertises itself](javascript:void(0)) [as basically only replying to a post.](javascript:void(0)) [I'm going to change around the code that was given to us automatically just a little bit.](javascript:void(0)) [And the idea here is that I want to take values that the users typed in.](javascript:void(0)) [I want them-- and I want to move them into a review so I can save that.](javascript:void(0)) [First, let's grab the review that we need to edit.](javascript:void(0)) [And then to move the values in, I'm going to call a method called TryUpdateModel.](javascript:void(0)) [What TryUpdateModel will do is go through a process known as model binding.](javascript:void(0)) [In fact, model binding happens anytime you even have a parameter in an action method.](javascript:void(0)) [It's what ASP.NET MVC does when it goes out and it looks around in the request try](javascript:void(0)) [to find things to move into an object for you.](javascript:void(0)) [So when I have a parameter called ID on the edit action,](javascript:void(0)) [the model binder in ASP.NET MVC will find that ID, move it into that for me.](javascript:void(0)) [When I say TryUpdateModel on review, the model binder will go out and look at review,](javascript:void(0)) [see that it has a rating property, and then go out and try to find something called rating.](javascript:void(0)) [But fortunately, there should be a posted form input named rating.](javascript:void(0)) [The MVC runtime will find that and just move it into my review.](javascript:void(0)) [If anything fails, if any validation errors occur, TryUpdateModel will return false](javascript:void(0)) [and I don't want to save that review.](javascript:void(0)) [We'll talk more about validation later.](javascript:void(0)) [And if that happens, I can return the view with that same review to try](javascript:void(0)) [to let the user fix whatever problem they have.](javascript:void(0)) [But if TryUpdateModel works, this is the point where I would save that into the database.](javascript:void(0)) [We're not working with the database yet.](javascript:void(0)) [And for right now, what I will do is return a redirect action.](javascript:void(0)) [Let's go back to the index action.](javascript:void(0)) [I want to redirect here instead of just letting the users sit](javascript:void(0)) [on the saved values on that posted form field.](javascript:void(0)) [It's very common that after an HTTP post](javascript:void(0)) [where the user has saved something you redirect them back to a page](javascript:void(0)) [where they can view the changed results.](javascript:void(0)) [That way, they don't hit refresh on the result of this post operation](javascript:void(0)) [and accidentally submit something twice.](javascript:void(0)) [But with this logic in place, let's try to do a build and let me come out and do a get](javascript:void(0)) [on this page slash review slash edit slash 3 so we can start everything over.](javascript:void(0)) [So you can come up again with the house of Elliot let's bump the rating down to a 9](javascript:void(0)) [and click save and you can see we made that change.](javascript:void(0)) [Let's edit this again change this back to a 10, perhaps using alternate spelling for Gent](javascript:void(0)) [and click save, those change is also reflected here.](javascript:void(0)) [And one of the reasons this was so easy to build is because we're using TryUpdateModel](javascript:void(0)) [which just relies on the names of properties to match up data](javascript:void(0)) [with what needs to get pushed into the model.](javascript:void(0)) [And all of that matches up because our edit view is using things like HTML.EditorFor](javascript:void(0)) [which will automatically build the correct inputs with the right names and the IDs](javascript:void(0)) [and the validation and everything else associated with it.](javascript:void(0)) [There are many other HTML helpers that are available than what you see here,](javascript:void(0)) [you can see them in the IntelliSense window.](javascript:void(0)) [There's ways to build check boxes.](javascript:void(0)) [There's ways to display things, there's ways to generate names,](javascript:void(0)) [route links and validation messages.](javascript:void(0)) [You can also build your own custom HTML helpers.](javascript:void(0)) [And I would suggest that if you find yourself typing a lot of C sharp code into a Razor view](javascript:void(0)) [to build an HTML element, I would suggest you go out and look at how to build a custom helper.](javascript:void(0)) [They're very easy to build and then they're very easy to use and a custom helper can really help](javascript:void(0)) [to get logic out of your view and keep the view as simple as possible.](javascript:void(0)) [That's really the ultimate goal of the HTML helpers, keeps views simple.](javascript:void(0)) [( Pause )](javascript:void(0))

* + [Partial Views](javascript:void(0))

[Another type of view that you'll find useful](javascript:void(0)) [when building MVC applications is the partial view.](javascript:void(0)) [A partial view allows you to allow you to put HTML on C sharp code into a file](javascript:void(0)) [that you can reuse across multiple other views.](javascript:void(0)) [We'll see some used cases for partial views in this course but for now, let's just imagine](javascript:void(0)) [that the div we have here to display a review is something that we need](javascript:void(0)) [in several different places in the application.](javascript:void(0)) [We show the reviews in a couple different places and we'd like the display to be consistent](javascript:void(0)) [but we do not want to duplicate this code every time we display a review.](javascript:void(0)) [We want to reuse it that's why I'm going to select the code and cut it out of this view.](javascript:void(0)) [Then I'll come over to our views folder, the reviews folder in the views folder](javascript:void(0)) [and say add a new view and I'll call this underscore review.](javascript:void(0)) [Again the underscore is just a convention that means this is a special type of view](javascript:void(0)) [in this case it's a partial view.](javascript:void(0)) [It's still a Razor view.](javascript:void(0)) [It can still be strongly typed against the restaurant review](javascript:void(0)) [but this time I'll leave the Scaffold template empty](javascript:void(0)) [because I already have the mark up that I want to put in here.](javascript:void(0)) [And I will select the check box that says create as a partial view.](javascript:void(0)) [What that really means is that when Visual Studio creates this,](javascript:void(0)) [it won't automatically add a code block at the top of the view to set the page title](javascript:void(0)) [because we don't want to set the page title](javascript:void(0)) [from a partial view that's the responsibility of the content view.](javascript:void(0)) [This is just rendering a portion of the page.](javascript:void(0)) [And now I can click add and I will paste in the code that I cut out from the index view](javascript:void(0)) [and hit control K, control D to do some formatting.](javascript:void(0)) [In this case, it just bumps it up against the left margin and aligns some things.](javascript:void(0)) [But in the mean time you have some HTML or C sharp code](javascript:void(0)) [that doesn't quite fit right, it's not formatted properly.](javascript:void(0)) [If you hit control K, control D that will do some formatting for you.](javascript:void(0)) [There is one other change I have to make in this view.](javascript:void(0)) [It used to be inside of a foreach loop that had a local variable called item.](javascript:void(0)) [Now, the model for this view is actually going to be a single restaurant review.](javascript:void(0)) [So I need to change item to model.](javascript:void(0)) [And one quick and easy way to do that is with quick replace.](javascript:void(0)) [We'll just change item to model, replace all and now our view should be ready to go.](javascript:void(0)) [And you can see that the partial view is just another Razor view.](javascript:void(0)) [It has the same csHTML extension that can still be strongly typed.](javascript:void(0)) [It still has a model that it can work against.](javascript:void(0)) [It's just the intention is that we're going to reuse this from other views](javascript:void(0)) [in the way you render a partial view, is to use another HTML helper HTML.Partial.](javascript:void(0)) [When you call HTML partial, you give it the name or the view that you want](javascript:void(0)) [to render in this case, Review.csHTML.](javascript:void(0)) [And you don't need to specify the file extension when you ask the MVC runtime to render a partial](javascript:void(0)) [but I can pass in the model that it needs to render.](javascript:void(0)) [The model in this case is each individual review each item.](javascript:void(0)) [And with all this in place, I can just save everything and refresh the application.](javascript:void(0)) [The user won't see any visual difference in the page.](javascript:void(0)) [It looks exactly the same as it did before.](javascript:void(0)) [We've just made a structural change in how the page displays.](javascript:void(0)) [We've created this partial view and we might have done that because we want](javascript:void(0)) [to reuse that across several other views.](javascript:void(0)) [Another good reason to use a partial view is just to simplify a view.](javascript:void(0)) [Now the foreach statement is a lot easier to read because I can clearly see when it starts](javascript:void(0)) [and when it ends and I can see that all we really do](javascript:void(0)) [in here is render each review using a partial view.](javascript:void(0)) [I also want to point out that since we put underscore review in the reviews folder,](javascript:void(0)) [it's only available from other views that are in the same folder.](javascript:void(0)) [If instead, we had placed this view in the shared folder, it would be available anywhere](javascript:void(0)) [in the application from any view to say render partial this review.](javascript:void(0)) [I also want to show you another scenario where partial views can be useful.](javascript:void(0)) [So one way to render a partial is to use this HTML.PartialHelper.](javascript:void(0)) [Typically, this is the approach you take when you want to reuse a piece of the HTML](javascript:void(0)) [or you want to simplify a view because you have a really complex model.](javascript:void(0)) [But regardless of why you're using HTML.Partial, when you do use it the only data](javascript:void(0)) [or model information that you can pass to HTML.Partial is model information](javascript:void(0)) [that you already have available inside of your view.](javascript:void(0)) [So you can pass the model item or you can pass a subset of the model item](javascript:void(0)) [like here, we just passed a single review.](javascript:void(0)) [But you can also come across a situation where you want](javascript:void(0)) [to render something that is not part of your model.](javascript:void(0)) [This frequently comes up with the layout view.](javascript:void(0)) [Because the layout view is executing for every page in the application,](javascript:void(0)) [and so it's very difficult to tie it to any particular model, there's ways to do that](javascript:void(0)) [but it involves a lot of messy inheritance.](javascript:void(0)) [So the scenario I'm talking about is imagine in the layout view if we want the best review](javascript:void(0)) [in the database to always appear somewhere on the page.](javascript:void(0)) [If I went to do that there's no way for me to go into the layout view](javascript:void(0)) [and specify a model directive here that says this view will always have a restaurant review](javascript:void(0)) [as a model because this layout view is used again throughout the entire application.](javascript:void(0)) [It's used on pages that display restaurant reviews but it will also be used on pages](javascript:void(0)) [that display restaurants or allow a user to register and log in.](javascript:void(0)) [And all of those other scenarios, they do not involve restaurant reviews.](javascript:void(0)) [So I cannot strongly type this layout view to a particular model.](javascript:void(0)) [But there is another helper available that can allow me to solve that situation.](javascript:void(0)) [And it involves partial views quite frequently.](javascript:void(0)) [This other helper is called HTML.Action.](javascript:void(0)) [So imagine we do want to show the best review on every page in the application.](javascript:void(0)) [We'll show it down here just above the footer.](javascript:void(0)) [What I can do is invoke HTML.Action and what HTML.Action allows you to do is set](javascript:void(0)) [up a sub request inside of this primary MVC request.](javascript:void(0)) [So it's a subrequest that can go out and call another controller action](javascript:void(0)) [that builds its own model and dependent of what the main controller is doing](javascript:void(0)) [and render its own view, a partial view that will be inserted right here](javascript:void(0)) [where the call to HTML.Action is.](javascript:void(0)) [So let's say that we need to build an action called BestReview.](javascript:void(0)) [And this is always going to be on the Reviews controller.](javascript:void(0)) [Just like HTML.Action link, you can also pass in additional parameters here](javascript:void(0)) [that will get passed along to this other controller action,](javascript:void(0)) [but we don't need to pass any parameters.](javascript:void(0)) [And this is basically saying go out and call this action on the reviews controller](javascript:void(0)) [and whatever it omits, whatever outputs whatever the result is,](javascript:void(0)) [place it here where I have HTML.Action.](javascript:void(0)) [And I do want to point out that this is not a separate HTTP request.](javascript:void(0)) [This doesn't require the browser to come back to the server and call this other controller action](javascript:void(0)) [into controller action that's called from server code it's like a subrequest.](javascript:void(0)) [So let's go over to the ReviewsController we need to implement BestReview](javascript:void(0)) [and I'll just put it as the top action here.](javascript:void(0)) [This can still be an action that returns an action result.](javascript:void(0)) [Actually it's going to be returning a PartialViewResult and it's called BestReview.](javascript:void(0)) [We'll need to find the best review and I realized that all](javascript:void(0)) [of our reviews right now have a rating of 10, so we'd need to find a way](javascript:void(0)) [to reconcile a tie or let's say from R in reviews.](javascript:void(0)) [Now we orderby R dot rating in descending order.](javascript:void(0)) [Now, we have our reviews in sorted order and now we're going](javascript:void(0)) [to return not a view this time, but a PartialView.](javascript:void(0)) [And we could create a brand new PartialView for this action to render.](javascript:void(0)) [It could be called BestReview.](javascript:void(0)) [We could also render that other PartialView that we just created,](javascript:void(0)) [the one that is called underscore Review.csHTML.](javascript:void(0)) [Again, no extension needed here and we will pass in the first review that we picked up based](javascript:void(0)) [on putting the reviews in the sorted order.](javascript:void(0)) [This is by the way how you would overwrite what view and action is going to render,](javascript:void(0)) [you can always explicitly specify a view by name here without the file extension.](javascript:void(0)) [But I can do that for the IndexView too.](javascript:void(0)) [If I didn't want it to return the IndexView](javascript:void(0)) [for some reason I could always specify a view name by string here.](javascript:void(0)) [But now I should be able to do a bill and let's come out to the home page of the application.](javascript:void(0)) [And what we should see here at the bottom of the home page is there's our best review.](javascript:void(0)) [We should probably put a better title around it so we know why](javascript:void(0)) [that information is appearing here.](javascript:void(0)) [But that should now be on every page of the application, because it's in the lay out view.](javascript:void(0)) [And we did that by calling an action that got to build its own model, render its own view](javascript:void(0)) [so it's completely independent from everything else,](javascript:void(0)) [it didn't really complicate our layout view to do those.](javascript:void(0)) [But one thing to keep in mind when you do build an action like this BestReview,](javascript:void(0)) [as I said before, your public methods are available to the browser.](javascript:void(0)) [So if I come in to the browser and I go](javascript:void(0)) [to slash Reviews slash BestReview I will be able to get that result.](javascript:void(0)) [Because like all public methods on the controller, if the user enters the right URL](javascript:void(0)) [into the browser, they're going to be able to invoke the controller action.](javascript:void(0)) [But in this case, if you do not want that to happen, there's an attribute](javascript:void(0)) [that you can apply to this action.](javascript:void(0)) [It's called ChildActionOnly.](javascript:void(0)) [And if I do that and do a build it's no longer legal to call](javascript:void(0)) [that directly using an HTTP request.](javascript:void(0)) [You can only access this by calling HTML.Action as a child request,](javascript:void(0)) [so it would still work in the layout view.](javascript:void(0)) [I just can't call it directly.](javascript:void(0)) [And now you know a couple of different ways to use partial views.](javascript:void(0)) [There's HTML.Partial which allows you to simplify a view and reuse HTML.](javascript:void(0)) [And there's also HTML.Action which allows you to set up a completely independent subrequest](javascript:void(0)) [that builds its own model and renders its own PartialView.](javascript:void(0)) [( Pause )](javascript:void(0))

* + [Summary](javascript:void(0))

[In this module, we covered a number of different topics related](javascript:void(0)) [to building views in an MVC application.](javascript:void(0)) [We look at the Razor syntax, and learned about implicit](javascript:void(0)) [and explicit code expressions as well as code blocks.](javascript:void(0)) [We talked about HTML Helpers and I showed you how helpers](javascript:void(0)) [like EditorFor will help you build form inputs.](javascript:void(0)) [But I haven't told you the whole story there yet.](javascript:void(0)) [We'll see more when we start validating incoming data.](javascript:void(0)) [We did look at layout views and how they can define the UI structure for our application](javascript:void(0)) [and I demonstrated how Razor does HTML encoding for us](javascript:void(0)) [to help avoid cross-site scripting attacks.](javascript:void(0)) [Finally, I showed you a couple of scenarios where PartialViews can be your friend.](javascript:void(0)) [You can use partials to render a portion of your model in a reasonable way](javascript:void(0)) [and you can also use partials and child requests with HTML.Action](javascript:void(0)) [to delegate work to another controller.](javascript:void(0))

* Working With Data (Part I)
  + [Introduction](javascript:void(0))

[Hi! This is Scott Allen and this module of the course is devoted to Data.](javascript:void(0)) [In this first Data module, we want to start using a real database with our application.](javascript:void(0)) [So the focus of the first module is to bring SQL Server into the picture.](javascript:void(0)) [In the next Data module, we'll look at other data-related features](javascript:void(0)) [in the MVC framework, features like model validation.](javascript:void(0)) [What I'll demonstrate in this first module is how to use a technology known](javascript:void(0)) [as the Entity Framework and combine the Entity Framework](javascript:void(0)) [with the plain simple C Sharp class definitions we've been working with to build an application](javascript:void(0)) [that uses SQL Server to persist data.](javascript:void(0)) [We will see how to query data using link in the Entity Framework,](javascript:void(0)) [and also use the Entity Framework Migrations](javascript:void(0)) [to manage our database schema from C Sharp and Visual Studio.](javascript:void(0))

* + [The Entity Framework](javascript:void(0))

[The first stop we'll take in this module is to get up and running with the Entity Framework.](javascript:void(0)) [The Entity Framework is a part of the .NET framework and you'll already have a reference](javascript:void(0)) [to the Entity Framework in the new MVC internet application.](javascript:void(0)) [The Entity Framework or EF as I sometimes call it for short,](javascript:void(0)) [allows me to access a relational database using strongly typed C Sharp code,](javascript:void(0)) [or Visual Basic code, or really any .NET language.](javascript:void(0)) [When I'm working with classes to talk to the Entity Framework, I don't have to worry](javascript:void(0)) [about SQL connections, SQL commands, SQL parameters, SQL data readers,](javascript:void(0)) [none of the low level abstractions you might have programmed with in the past with .NET.](javascript:void(0)) [And if you don't know them, that's good for you because you won't have to learn them.](javascript:void(0)) [With the Entity Framework, I work with my C Sharp code and I can issue queries](javascript:void(0)) [to the database using a language feature called Language Integrated Query also known as LINQ](javascript:void(0)) [and this is a feature that's in both the C Sharp and Visual Basic languages.](javascript:void(0)) [There's a few different ways to get started with the Entity Framework.](javascript:void(0)) [One approach is called the Schema First approach.](javascript:void(0)) [In this approach, you open up a graphical designer in Visual Studio,](javascript:void(0)) [you point it to an existing database, and it can import the database schema](javascript:void(0)) [and generate all the classes you need to query and update that database.](javascript:void(0)) [Another approach is a Model First approach, this is where I use the same graphical designer](javascript:void(0)) [in Visual Studio to draw a conceptual model for my applications, so what classes do I want.](javascript:void(0)) [And then I have the Entity Framework generate both my class definitions](javascript:void(0)) [and my database schema.](javascript:void(0)) [Finally, there's a Code First approach you can take with EF.](javascript:void(0)) [In this approach, I just write C Sharp classes and the Entity Framework](javascript:void(0)) [and use those class definitions to create a database for me.](javascript:void(0)) [And it will do that either using conventions like naming conventions,](javascript:void(0)) [very similar to the naming conventions we've seen with the MVC framework and that is](javascript:void(0)) [where things just magically work if they had the right name.](javascript:void(0)) [I can also provide explicit mappings or change the mappings if I don't like any](javascript:void(0)) [of these conventions or defaults that the Entity Framework uses.](javascript:void(0)) [In this module, we're going to use a Code First approach,](javascript:void(0)) [so we won't see a visual design surface or any XML configuration files.](javascript:void(0)) [I hope that makes you happy.](javascript:void(0)) [And when I say that the Entity Framework supports relational databases,](javascript:void(0)) [then the Entity Framework supports quite a few.](javascript:void(0)) [You can point the Entity Framework towards a full-blown SQL Server installation](javascript:void(0)) [on a remote server that has hundreds of gigabytes of RAM and network storage.](javascript:void(0)) [Or you can point it to SQL Server Compact edition which is running in process](javascript:void(0)) [and accessing a database file on the local file system.](javascript:void(0)) [You can also use it with usher in the clown (phonetic),](javascript:void(0)) [you can use it with Oracle, you can use it with DB2.](javascript:void(0)) [But my code really doesn't have to know what's on the back end.](javascript:void(0)) [The Entity Framework knows how to work with all these different databases.](javascript:void(0)) [I can build an application without worrying so much about the database details.](javascript:void(0)) [In this module, we'll be working with SQL Server,](javascript:void(0)) [specifically the local DB version of SQL Server.](javascript:void(0))

* + [Building Entities](javascript:void(0))

[So far in the application, we've been using some in-memory data and this Restaurant Review class](javascript:void(0)) [to get some things done and experiment with razor and controllers.](javascript:void(0)) [But this restaurant review really combines a restaurant, the restaurant name and a review](javascript:void(0)) [which is the rating, I really need to separate those out](javascript:void(0)) [and form a relationship if we're going to move forward.](javascript:void(0)) [So there should be a restaurant class which I can create right now.](javascript:void(0)) [And that restaurant class will have multiple reviews associated with it.](javascript:void(0)) [We'll give a couple of properties to the restaurant,](javascript:void(0)) [there's going to be a primary key fields so we'll give it an ID property.](javascript:void(0)) [We'll give it a name.](javascript:void(0)) [We'll also store the city that the restaurant is in and the country.](javascript:void(0)) [And then every restaurant can have some associated reviews.](javascript:void(0)) [We'll put that in ICollection, a collection of restaurant reviews.](javascript:void(0)) [And this seems like a good definition for a restaurant.](javascript:void(0)) [Let's go over into the review class and change some things around.](javascript:void(0)) [A restaurant review will still have an ID, the name, city and country,](javascript:void(0)) [that's really restaurant information.](javascript:void(0)) [We don't need that.](javascript:void(0)) [We'll have the rating.](javascript:void(0)) [Perhaps we want the body of the review, so if a person wants to type out some text,](javascript:void(0)) [what they really thought about the restaurant, they can put that in here.](javascript:void(0)) [And although it's not entirely necessary,](javascript:void(0)) [it does make some scenarios easier if I add a restaurant ID field.](javascript:void(0)) [And this restaurant ID will point back to the restaurant that this review is associated with.](javascript:void(0)) [And now restaurant review and restaurant are my two entities, these are going to be objects](javascript:void(0)) [that I expect to instantiate and save in a database and retrieve from a database.](javascript:void(0)) [But right now, I'm leaving their definition in the models folder.](javascript:void(0)) [But if you continue down this path on larger applications,](javascript:void(0)) [the model's folder becomes way too big.](javascript:void(0)) [There's too many classes inside of it.](javascript:void(0)) [You might consider having a separate folder or even a separate project](javascript:void(0)) [where you define entities like this.](javascript:void(0)) [But this is good enough for our purposes to get started with.](javascript:void(0)) [Now let me also add a class that will use the Entity Framework](javascript:void(0)) [to persist and retrieve this data.](javascript:void(0)) [We'll call it OdeToFoodDb.](javascript:void(0)) [OdeToFoodDb needs to derive from an Entity Framework class known as the DbContext.](javascript:void(0)) [To use DbContext, I'll need to bring the System.Data.Entity name space](javascript:void(0)) [and the scope, so I'll add that using.](javascript:void(0)) [And then a DbContext, you can have properties of type DbSet that represent the entities](javascript:void(0)) [that you want to query and persist.](javascript:void(0)) [So we'll have a DbSet of restaurant called Restaurants.](javascript:void(0)) [And walking up to that DbSet and asking](javascript:void(0)) [for all the restaurants would essentially do a select star](javascript:void(0)) [from a restaurant table in SQL Server.](javascript:void(0)) [We'll have Restaurants and we'll have a DbSet of RestaurantReview.](javascript:void(0)) [RestaurantReviews is not something that I absolutely must have,](javascript:void(0)) [I can always get to reviews through a restaurant but I'm going to add it as a DbSet here.](javascript:void(0)) [Again, it makes some things easier.](javascript:void(0)) [To see how easy this is to use, let's say that we want the home page of the application now](javascript:void(0)) [to display a list of restaurants that we have in the database.](javascript:void(0)) [So let me swing over into the home controller and we're going to change](javascript:void(0)) [around the index action and also the view for this controller.](javascript:void(0)) [We don't really want to show the controller and the action and the ID anymore,](javascript:void(0)) [instead, we want to show restaurants.](javascript:void(0)) [So here I can instantiate OdeToFoodDb and I'll also point](javascript:void(0)) [out the sense this is a disposable resource, I should override dispose.](javascript:void(0)) [And if this is not null, tell it to dispose itself.](javascript:void(0)) [To dispose API and .NET is just a way to clean up resources that might be open.](javascript:void(0)) [Anything that implements the IDisposable interface or has a dispose method,](javascript:void(0)) [you should call it as soon as possible](javascript:void(0)) [to make sure everything is cleaned up as early as possible.](javascript:void(0)) [But with this bit of code in place, let's come into the home index action](javascript:void(0)) [and say that our model is equal to db.Restaurants.ToList.](javascript:void(0)) [What the Entity Framework will do when I execute that line of code is go into SQL Server,](javascript:void(0)) [find where it stored all the restaurants, retrieve all of them and put them into a list.](javascript:void(0)) [Eventually, if this application grows, we will need to provide some search facilities,](javascript:void(0)) [we'll need to provide some paging facilities but this should be enough to build a screen](javascript:void(0)) [that has a list of all our restaurants.](javascript:void(0)) [And now let's swing over into the index action for the home controller.](javascript:void(0)) [And let's clean some things up.](javascript:void(0)) [Let's get rid of the feature section and also the information about what we suggest](javascript:void(0)) [because we really just want to show restaurants on this page.](javascript:void(0)) [And because this view is already created, I'll add a model directive here manually to say](javascript:void(0)) [that this is strongly typed against IEnumerable of restaurant.](javascript:void(0)) [And I need to fully name space qualify that as OdeToFood.Models.Restaurant.](javascript:void(0)) [Again, just control period will give you that drop down menu](javascript:void(0)) [and you can have Visual Studio add that for you.](javascript:void(0)) [And now let's write those restaurants out.](javascript:void(0)) [So for each item in my model or each item as a restaurant, let's write out a div.](javascript:void(0)) [It has an h4 inside with the restaurant name.](javascript:void(0)) [And we can also write out the city and the country.](javascript:void(0)) [And then perhaps just put in a little horizontal line](javascript:void(0)) [to distinguish one restaurant from the other.](javascript:void(0)) [Now at this point, we don't have any data in a database because we don't even have a database](javascript:void(0)) [but we're very close to doing that.](javascript:void(0)) [Just by having a DbContext derived class, if I instantiate this class and ask it for data,](javascript:void(0)) [it will go out and see if a database exist.](javascript:void(0)) [And if that database doesn't exist, it will be created for me.](javascript:void(0)) [So I should be able to do a build.](javascript:void(0)) [And, of course, we'll have some compiler areas because we already had some code inside](javascript:void(0)) [of the reviews controller that was depending on a different definition of restaurant review.](javascript:void(0)) [So for right now, I'm going to hit Control A to highlight all the code inside](javascript:void(0)) [of our reviews controller and then hit Control K, control C to just comment it out.](javascript:void(0)) [We'll come back to the reviews controller later and rebuild it](javascript:void(0)) [so that it actually displays reviews that are associated with our restaurant instead](javascript:void(0)) [of using this in-memory data that we had here before.](javascript:void(0)) [There's another change that I want to make because that's commented out.](javascript:void(0)) [We're also going to have to change the layout view because if you remember,](javascript:void(0)) [caught into the reviews controller to list the best review but we don't really need that.](javascript:void(0)) [I'm going to delete that bit of code from the layout view.](javascript:void(0)) [And since the definition of our restaurant review has changed so dramatically,](javascript:void(0)) [I'm going to go ahead into the Reviews' views folder](javascript:void(0)) [and just delete our existing views from inside of here.](javascript:void(0)) [And this is all okay.](javascript:void(0)) [We're just taking in an iterative approach here to building an application](javascript:void(0)) [and learning MVC at the same time.](javascript:void(0)) [But I should have done enough now to be able to do a build and run the application.](javascript:void(0)) [And on the home page, it will try to list our restaurants](javascript:void(0)) [but of course we have no restaurants.](javascript:void(0)) [But the question here is, we didn't get an error so the Entity Framework talked](javascript:void(0)) [to some database, where is that database?](javascript:void(0)) [And to answer that question, let's go to view Database Explorer and add a connection.](javascript:void(0)) [This is going to be a SQL Server connection.](javascript:void(0)) [And I'm going to specify LocalDB v11.0 as the server I want to connect to.](javascript:void(0)) [This is a special connection string for SQL Server 2012 Express, the LocalDB edition.](javascript:void(0)) [So if you haven't heard of SQL Server LocalDB, it's a new edition](javascript:void(0)) [of SQL Server that's primarily targeted towards developers.](javascript:void(0)) [The big difference between this and SQL Server Express which you might have used](javascript:void(0)) [in the past during development is that LocalDB runs](javascript:void(0)) [with your identity and it doesn't run as a service.](javascript:void(0)) [It's launched on-demand when you need to get to it.](javascript:void(0)) [Other than that, it's very much like SQL Server.](javascript:void(0)) [And now let's pull down this drop down list to see what databases are available there.](javascript:void(0)) [And I can see that one of the databases is OdeToFood.Models.OdeToFoodDb.](javascript:void(0)) [I know this is the database that the Entity Framework just created because the name](javascript:void(0)) [of that database exactly matches the name of the OdeToFoodDb class that we created.](javascript:void(0)) [So by default, when you instantiate one of these and try to query data,](javascript:void(0)) [if the Entity Framework doesn't find a database and you have no other configuration in place](javascript:void(0)) [that tells the Entity Framework where to go, it will just look at LocalDB and create a database](javascript:void(0)) [that has the same name as your DBContext class.](javascript:void(0)) [And if we come in here, indeed we can see Restaurants.](javascript:void(0)) [We can see the RestaurantReviews.](javascript:void(0)) [We can see that RestaurantReviews have a primary key.](javascript:void(0)) [That's the ID column.](javascript:void(0)) [We can see that Restaurants also have an ID column.](javascript:void(0)) [And just to prove that this is working, if I go in and I say, show table data,](javascript:void(0)) [what I should be able to do here is add some new restaurant into the database manually.](javascript:void(0)) [Save that in the database.](javascript:void(0)) [And if I come up to the home page of the application again and refresh,](javascript:void(0)) [that restaurant is displaying on the home page.](javascript:void(0)) [So we are querying SQL Server and we've created our first database.](javascript:void(0)) [All this was taken care of by the Entity Framework just using the definitions](javascript:void(0)) [of our C Sharp classes, Restaurant and RestaurantReview.](javascript:void(0))

* + [Database Migrations](javascript:void(0))

[You might be wondering if it's possible to influence](javascript:void(0)) [where the Entity Framework creates database and also](javascript:void(0)) [if you can influence how the Entity Framework creates a database.](javascript:void(0)) [Can you change the names of tables and add indexes and things like that?](javascript:void(0)) [Let's look at the first question, the where question.](javascript:void(0)) [You can configure your data context with a specific database connection string.](javascript:void(0)) [The way you do this is to call into the base class constructor.](javascript:void(0)) [And the syntax for that in C Sharp is to use the keyword base.](javascript:void(0)) [And right here, I could pass a connection string.](javascript:void(0)) [Explicitly specify the server, the database, and the credentials.](javascript:void(0)) [But we generally don't hard code connection strings into an application because when we want](javascript:void(0)) [to point the application to a different database that means we have](javascript:void(0)) [to change the code and recompile.](javascript:void(0)) [And since we usually write applications that used different databases, at least,](javascript:void(0)) [there's one database during development and one for deployment, and perhaps one for testing.](javascript:void(0)) [Well, it's a good idea not to hard code connection stings.](javascript:void(0)) [Instead, with ASP.NET, we can store connection strings in our web.config file.](javascript:void(0)) [I'm going to open up the web.config file that's in the root of this application.](javascript:void(0)) [And we can see there already is a connection string here in place](javascript:void(0)) [for us, it's called default connection.](javascript:void(0)) [The project template put this in place when we created this new application.](javascript:void(0)) [And we can change this connection string to point anywhere.](javascript:void(0)) [I'm just going to tweak it a little bit.](javascript:void(0)) [I still want it to point to LocalDB but I want the initial catalog not](javascript:void(0)) [to have the word ASP.NET and all this other stuff in it.](javascript:void(0)) [I just want it to be OdeToFoodDb Integrated Security equals SSPI.](javascript:void(0)) [That basically means connect with the process credentials, which is fine.](javascript:void(0)) [We do want it to attach to a file that's in the DataDirectory.](javascript:void(0)) [And again, I'm going to call this OdeToFoodDb.mdf](javascript:void(0)) [so it's very easy to identify there.](javascript:void(0)) [And now, if I want the Entity Framework to use this connection string, what I do when I call](javascript:void(0)) [into the base class constructor is say, name equals, and the name of my connection string.](javascript:void(0)) [And that tells it to go out into the web.config and use whatever that connection string value is](javascript:void(0)) [when it connects to the database.](javascript:void(0)) [Now, I never need to recompile the application when I want to point somewhere else,](javascript:void(0)) [I just need to go into the config file and I should be able to run the application again.](javascript:void(0)) [And we should be back to a blank page, there it is,](javascript:void(0)) [because we're now pointing to a new database.](javascript:void(0)) [It doesn't have that restaurant in there that I added manually.](javascript:void(0)) [But I should still be able to use the Database Explorer to poke around in that database.](javascript:void(0)) [In fact, default connection is already here in my data connections.](javascript:void(0)) [It just detected that when I loaded this project.](javascript:void(0)) [And there's Restaurants and RestaurantReviews.](javascript:void(0)) [I could add data in there manually again.](javascript:void(0)) [And also, I should be able to see this database now if I come](javascript:void(0)) [into the Solution Explorer and look in the App Data folder.](javascript:void(0)) [What I'll have to do is allow Solution Explorer to show all files with this toolbar button.](javascript:void(0)) [And there I can see OdeToFoodDb.mdf.](javascript:void(0)) [There are some other databases here that I should clean out](javascript:void(0)) [but we'll worry about that later.](javascript:void(0)) [Let's go back to our second question at the beginning which is,](javascript:void(0)) [how can I influence how the Entity Framework creates my schema?](javascript:void(0)) [And can I have it initially populates some data for me into that database?](javascript:void(0)) [And the answer to both of these questions is yes.](javascript:void(0)) [All we need to do is use Entity Framework Migrations.](javascript:void(0)) [Migrations are a feature of the Entity Framework that allow you to configure database schemas](javascript:void(0)) [with C Sharp code, seed your databases with C Sharp code and Migrations can then track changes](javascript:void(0)) [that you're making in your entity classes.](javascript:void(0)) [Those classes like Restaurant and RestaurantReview.](javascript:void(0)) [And it can keep the database schema in sync with the changes that you make in your C Sharp code.](javascript:void(0)) [The way to get started with Migrations is to open up the Package Manager Console.](javascript:void(0)) [There's two ways to do this.](javascript:void(0)) [You can go to the View menu and go to Other Windows and find the Package Manager Console](javascript:void(0)) [or you can use this Quick Launch toolbox that's up here in the upper right of Visual Studio](javascript:void(0)) [and just type, Package or Package Manager Console, it'll find all the commands](javascript:void(0)) [in the menus everywhere throughout Visual Studio really quickly](javascript:void(0)) [and then you can just click on it.](javascript:void(0)) [The Package Manager Console will open up here inside of Visual Studio as another window.](javascript:void(0)) [It's essentially a PowerShell command line.](javascript:void(0)) [If you know how to write scripts in PowerShell then you can use any PowerShell command in here](javascript:void(0)) [that you want and list directories and manage services.](javascript:void(0)) [Inside of here if you want to get what directory you're in, you can type GET-D and then hit tab.](javascript:void(0)) [You can see it has some tab completion.](javascript:void(0)) [Let's try and Get-Date instead, that executes, that's a PowerShell command.](javascript:void(0)) [But we're here actually for Migrations.](javascript:void(0)) [And what I want to do is enable Migration.](javascript:void(0)) [So, enable dash with an M, it knows I want Enable Migrations.](javascript:void(0)) [And I'm going to have to specify a ContextTypeName.](javascript:void(0)) [This is the name of my DBContext that I want to enable Migrations for.](javascript:void(0)) [In this case, just typing in OdeToFoodDb is sufficient.](javascript:void(0)) [And now when I press enter, this is going to kick off some logic that is part](javascript:void(0)) [of the Code First Migrations of the Entity Framework that goes out and pokes](javascript:void(0)) [around the environment and looks like at my classes, looks at any existing database,](javascript:void(0)) [and it will add a new folder with a new file into my solution.](javascript:void(0)) [The new folder is called Migrations.](javascript:void(0)) [And inside of Migrations, right now there's two files, Configuration.cs](javascript:void(0)) [and basically a schema change script that's written in C Sharp.](javascript:void(0)) [Let's talk about configuration first.](javascript:void(0)) [The configuration class is really about controlling Code First Migrations.](javascript:void(0)) [How do you want it to perform?](javascript:void(0)) [When should it run?](javascript:void(0)) [The one setting in here that is most notable is the Automatic Migrations Enabled setting.](javascript:void(0)) [By default, it will be false.](javascript:void(0)) [And that means, basically, that the Entity Framework won't make any changes](javascript:void(0)) [in your database unless you explicitly tell the Entity Framework](javascript:void(0)) [that you want it to make changes.](javascript:void(0)) [I'm going to set this to true because quite often,](javascript:void(0)) [when you are initially starting a project, you just want to be able to make changes](javascript:void(0)) [in your C Sharp code wherever those classes are and just have the database be ready](javascript:void(0)) [for you in the application to run.](javascript:void(0)) [If you're working in a more controlled environment](javascript:void(0)) [and the product is a little more mature, you might want to set this to false](javascript:void(0)) [and be very explicit about when changes in the database happen.](javascript:void(0)) [But I'll show the impact of setting this to true a little bit later.](javascript:void(0)) [The second nice part of the configuration class is the seed method.](javascript:void(0)) [This is basically where you can tell the Entity Framework](javascript:void(0)) [to populate the database with some initial data.](javascript:void(0)) [And now, every time that it goes to update the database, it's going to invoke the seed method.](javascript:void(0)) [And this gives you the opportunity to populate those tables](javascript:void(0)) [that always need data, things like lookup data.](javascript:void(0)) [You need a list of countries.](javascript:void(0)) [You need a list of postal codes, all of those types of things can go](javascript:void(0)) [into seed, and that's usually static data.](javascript:void(0)) [But I'm going to do something a little bit different because we do not have a screen](javascript:void(0)) [yet that will allow us to create a restaurant or a review, I'm going to paste in some code](javascript:void(0)) [that will create restaurants for us.](javascript:void(0)) [I'm pasting it in so you don't have to watch my typing skills, I know it's painful.](javascript:void(0)) [I do need to bring in some name spaces, OdeTofood.models and system.collection.generic.](javascript:void(0)) [But now that this code is legal, let me explain a little about what it does.](javascript:void(0)) [This is walking up to the RestaurantsDb set on our OdeToFoodDb context and it's saying add](javascript:void(0)) [or update the following restaurants.](javascript:void(0)) [So literally go into the database and see if any of these restaurants exist](javascript:void(0)) [by name, it's looking them up by name.](javascript:void(0)) [So what we'll see if there's a Sabatino's in the Restaurants table.](javascript:void(0)) [And if there is, it will update that record with the information I have here.](javascript:void(0)) [If that name doesn't exist in the Restaurants table,](javascript:void(0)) [it will add this new restaurant as a row into that table.](javascript:void(0)) [It will do that for all three of these, in fact,](javascript:void(0)) [it's even going to add a review for this last restaurant here.](javascript:void(0)) [And now, every time I update the database, the seed method will run.](javascript:void(0)) [And when I'm talking about updates, I'm talking about Migration updates.](javascript:void(0)) [You typically run a database update when you want to migrate the schema.](javascript:void(0)) [And that's why it's important that we're not just doing an add](javascript:void(0)) [because we might update this database multiple times.](javascript:void(0)) [As we're making schema changes, we're doing an add or update](javascript:void(0)) [so this information isn't duplicated inside of here.](javascript:void(0)) [So how do you update the database?](javascript:void(0)) [There's a couple of different ways to do it.](javascript:void(0)) [You can configure the application to automatically apply updates,](javascript:void(0)) [that's going to be very useful much later in the course when we deploy this](javascript:void(0)) [and it will just create a database for us in the Cloud.](javascript:void(0)) [We can also do it explicitly through the Package Manager Console.](javascript:void(0)) [It's a command update-database which I'm going to run](javascript:void(0)) [with the flag-verbose, so update database-verbose.](javascript:void(0)) [Verbose isn't going to add much information to this update](javascript:void(0)) [because we don't have any other changes to do.](javascript:void(0)) [And the two important pieces of output here are number one,](javascript:void(0)) [there were no pending code-based Migrations which is good.](javascript:void(0)) [That means essentially nothing in our C Sharp class has changed](javascript:void(0)) [so it didn't have to change the database.](javascript:void(0)) [But it did run the seed method that would that put that information into the database.](javascript:void(0)) [I'd be able to see that if I went into the Database Explorer.](javascript:void(0)) [I should also be able to see it if I just refresh the application.](javascript:void(0)) [And there you can see, I pulled those records out of the database.](javascript:void(0)) [So they are always there.](javascript:void(0)) [And every time I do the update database, it will check to make sure that they're there](javascript:void(0)) [and update them if they already are.](javascript:void(0)) [The other piece of output about no pending code-based Migrations, again,](javascript:void(0)) [that's because the Entity Framework didn't detect any changes.](javascript:void(0)) [But the other thing I can do when I run update database is have the database synchronize](javascript:void(0)) [with my model.](javascript:void(0)) [So for instance, if at this point,](javascript:void(0)) [I realize that a review should really also include the name of the person](javascript:void(0)) [that was reviewing this, I would probably come into this class and give this a new property,](javascript:void(0)) [let's call it ReviewerName and do a Build.](javascript:void(0)) [And at this point, if I come back to the Package Manager Console, I could do one of two things.](javascript:void(0)) [I could tell the Entity Framework that I explicitly need a Migration script to move](javascript:void(0)) [from the database schema as it currently exists into a new database schema](javascript:void(0)) [that can store ReviewerName or I can just tell it to update the database and let it figure](javascript:void(0)) [that out without being very explicit about it.](javascript:void(0)) [I'd only be able to do that because Automatic Migrations are enabled.](javascript:void(0)) [So what I will want to do is just to update the database.](javascript:void(0)) [But before I do that, let me show you what a Migration script would look like.](javascript:void(0)) [Because when we enabled Migrations in this project,](javascript:void(0)) [the Entity Framework created both the configuration.csfile](javascript:void(0)) [but also this, and this will create .CS file.](javascript:void(0)) [And if we look in here, it's essentially a schema change in C Sharp code.](javascript:void(0)) [In order to initially create the database,](javascript:void(0)) [what we need to do is create a table called Restaurants, give it the following columns, ID,](javascript:void(0)) [name, city, and country, and set the ID to be primary key.](javascript:void(0)) [By the way, ID is also an identity column.](javascript:void(0)) [That means in SQL Server, you'll have](javascript:void(0)) [that value automatically populated when you insert a new row.](javascript:void(0)) [We also need to create a table called RestaurantReviews with the following columns.](javascript:void(0)) [It has a primary key.](javascript:void(0)) [It also has foreign key that references another table.](javascript:void(0)) [And we want to apply an index to RestaurantId because it will probably be important](javascript:void(0)) [to query the reviews given a specific Restaurant ID](javascript:void(0)) [so we can find all the reviews for a given Restaurant.](javascript:void(0)) [I'll just quickly point out that there's many other commands that you can run in here.](javascript:void(0)) [For instance, if you just need it to execute some row of SQL during a Migration,](javascript:void(0)) [maybe to update some data, you could put an update statement in here,](javascript:void(0)) [execute it through that SQL method.](javascript:void(0)) [And when you update the database,](javascript:void(0)) [when the Entity Framework applies the script, that statement will execute.](javascript:void(0)) [So these Migrations, I keep calling them scripts even though they're not really SQL scripts,](javascript:void(0)) [they're C Sharp code.](javascript:void(0)) [But you can generate SQL scripts from them.](javascript:void(0)) [The Entity Framework keeps track of which ones have been applied to a database,](javascript:void(0)) [which new ones need to be applied, and then what order they need to be applied.](javascript:void(0)) [It takes care of all that for you to a table that's hidden inside](javascript:void(0)) [of database called underscore Migration History.](javascript:void(0)) [It will be a system table in your database.](javascript:void(0)) [But at this point, what has happened is that we have changed the definition of review.](javascript:void(0)) [We really need to update the database to get that in sync.](javascript:void(0)) [So let me run Update-Database-Verbose again and you'll see in the output](javascript:void(0)) [that the Entity Framework automatically applied an alter table RestaurantReviews](javascript:void(0)) [to add ReviewerName to the table.](javascript:void(0)) [So that's what I could've generated as an explicit Migration](javascript:void(0)) [that would come out in the C Sharp code style.](javascript:void(0)) [But since I have the Automatic Migrations enabled,](javascript:void(0)) [I just let the Entity Framework take care of that.](javascript:void(0)) [And if I wanted to actually have a name in there for this particular review,](javascript:void(0)) [I could always say ReviewerName equal Scott.](javascript:void(0)) [Run update database again and it will run that seed method,](javascript:void(0)) [that value Scott should be put into the database.](javascript:void(0)) [And now, one more thing to know about, you might have noticed that when we added ReviewerName,](javascript:void(0)) [it was added as in nvarchar(max) field.](javascript:void(0)) [That may not be what you want.](javascript:void(0)) [You might want to restrict that field to a specific lane.](javascript:void(0)) [But I'm going to defer that discussion till the next module when we start talking](javascript:void(0)) [about data annotations because we can change that through a data annotation.](javascript:void(0)) [In fact, we can change nearly everything that the Entity Framework is doing to the database](javascript:void(0)) [in some fashion either through attributes or through some explicit code.](javascript:void(0)) [But for our application, we're just going to move ahead.](javascript:void(0)) [We've seen just enough to know how to use Entity Framework,](javascript:void(0)) [how to use Entity Framework Migrations.](javascript:void(0)) [And we have an application that's up and running that's displaying restaurants](javascript:void(0)) [that are coming from a table in SQL Server.](javascript:void(0))

* + [Using LINQ](javascript:void(0))

[We now have a working database in our project and we will be using LINQ statements](javascript:void(0)) [to query the data in that database.](javascript:void(0)) [LINQ stands for Language Integrated Query.](javascript:void(0)) [And I want you to know about two different styles of queries that you can write with LINQ.](javascript:void(0)) [The first style looks almost like a query you would write](javascript:void(0)) [with structured query language against SQL Server.](javascript:void(0)) [One difference you'll notice that might jump out immediately though is that select appears](javascript:void(0)) [at the end of the statement, unlike in Structured Query Language](javascript:void(0)) [where you begin the query with a select.](javascript:void(0)) [These LINQ queries are going to be strongly typed.](javascript:void(0)) [That is the Compiler and Visual Studio, they know about the types](javascript:void(0)) [of objects you're querying against.](javascript:void(0)) [So if you change something in a class, you can end up with compiler areas in your queries](javascript:void(0)) [because the compiler will know exactly what is inside of those objects.](javascript:void(0)) [This particular query is what we call the Comprehension Query Syntax.](javascript:void(0)) [It starts with a from keyword to specify where the data is coming from](javascript:void(0)) [and it introduces a range variable.](javascript:void(0)) [In this case, the name of the range variable is r. You can use that throughout the rest](javascript:void(0)) [of the query when you want to do filtering, grouping, joining, and projecting.](javascript:void(0)) [There are keywords in C Sharp to let you do the filtering, grouping, joining, and projecting.](javascript:void(0)) [Keywords like where, and orderby, and select.](javascript:void(0)) [That's just a subset of the available keywords.](javascript:void(0)) [In this example, we're taking our restaurants,](javascript:void(0)) [filtering out so we only have the restaurants that have a country of USA.](javascript:void(0)) [We're going to order them by their names and that's an ascending order by default](javascript:void(0)) [and then select the entire Restaurant.](javascript:void(0)) [We're not doing a projection into some smaller set of data.](javascript:void(0)) [The job of the Entity Framework then is to take this query and translate it](javascript:void(0)) [into an efficient query for the database, then execute that query](javascript:void(0)) [and bring you back objects, actual restaurant objects.](javascript:void(0)) [But you can also express this query a second way.](javascript:void(0)) [This query is using extension method and lambda expressions.](javascript:void(0)) [The first part of this query would produce the exact same results as the query above.](javascript:void(0)) [We want restaurants whose country is equal to a specific value,](javascript:void(0)) [and we want them ordered by name.](javascript:void(0)) [But when you use this syntax, there are some additional operators that you can use.](javascript:void(0)) [For instance, Skip and Take.](javascript:void(0)) [They're quite commonly used when you want to do paging operations.](javascript:void(0)) [I want to skip the first ten results because they're on the first page.](javascript:void(0)) [I want to take the next ten results because I'm going](javascript:void(0)) [to display the second page of results to users.](javascript:void(0)) [There's no C Sharp keywords to express those operators](javascript:void(0)) [so you'll always see those invoked as extension methods.](javascript:void(0)) [There are dozens of LINQ operators available.](javascript:void(0)) [If you want to explore the syntax and the possibilities,](javascript:void(0)) [then you can look for 101 LINQ Samples on MSDN.](javascript:void(0)) [These samples are available on both C Sharp and Visual Basic.](javascript:void(0)) [You can also download a free utility called LINQPad.](javascript:void(0)) [It comes with hundreds of samples and you can write LINQ queries](javascript:void(0)) [in an interactive manner and try them out immediately.](javascript:void(0)) [Also, if you're a Pluralsight subscriber, I have a few courses devoted to LINQ.](javascript:void(0)) [Well let's try out a few different queries inside of our application.](javascript:void(0)) [Inside the application, we're displaying a list of restaurants now on the home page](javascript:void(0)) [but they're obviously not in alphabetical order.](javascript:void(0)) [So let's try to fix that by changing the query](javascript:void(0)) [around in the index action of the HomeController.](javascript:void(0)) [I'm going to switch over to using the comprehension query syntax for a bit](javascript:void(0)) [and say from r, where r is my range variable, it represents a restaurant](javascript:void(0)) [because I'm going to say from r in db.restaurants.](javascript:void(0)) [Let's OrderBy r. Name.](javascript:void(0)) [By default, this will be ascending, you can explicitly specify that using the keyword.](javascript:void(0)) [You could also specify descending but we'll stick with ascending for right now](javascript:void(0)) [and just select the entire restaurant.](javascript:void(0)) [Just by putting that change in the code, if I refresh, we should now be getting Great Lakes](javascript:void(0)) [on top of the list which we do, so that's good.](javascript:void(0)) [But we could do some more fancier ordering.](javascript:void(0)) [Let me try to arrange these windows so we can see everything.](javascript:void(0)) [And now let's say that we want to display the restaurants but perhaps we want](javascript:void(0)) [to display the most popular restaurants at the top.](javascript:void(0)) [And perhaps one way to measure the popularity of a restaurant is](javascript:void(0)) [to see how many times it's been reviewed.](javascript:void(0)) [So we'll order by r.Reviews.Count descending, so the one with the most review should be here](javascript:void(0)) [at the top and that's going to be our restaurant in Sweden, if I refresh.](javascript:void(0)) [And now that appears to work too.](javascript:void(0)) [And behind the scenes, the Entity Framework is taking this query and translating it](javascript:void(0)) [into a query for SQL Server then it executes that query and brings back a result set](javascript:void(0)) [from SQL Server but transforms that result set into real objects,](javascript:void(0)) [into real restaurants that I can use in C Sharp code.](javascript:void(0)) [And you can see the query we have now is a bit of a mix.](javascript:void(0)) [It is in the comprehension query syntax but there's a little bit of extension method](javascript:void(0)) [in here too because we need to count the reviews.](javascript:void(0)) [And there's no keyword in C Sharp to say count.](javascript:void(0)) [Now maybe someone discovers that the restaurants coming to the top are restaurants](javascript:void(0)) [that have a lot of reviews but they're all negative reviews.](javascript:void(0)) [So what we'd really like to do is order by the average review and find](javascript:void(0)) [out what the best restaurants are.](javascript:void(0)) [So there I could say r.Reviews, please give me an average, compute an average.](javascript:void(0)) [And for the average extension method to work, I need to pass in a lambda expression](javascript:void(0)) [that tells the Entity Framework which property I want to average.](javascript:void(0)) [So given a review, please average Review.rating,](javascript:void(0)) [and we'll still specify this in descending order.](javascript:void(0)) [And if I do a build and refresh, you can see now we have a relatively complicated query going on,](javascript:void(0)) [still going to produce the same result though](javascript:void(0)) [because we only have one restaurant with reviews in the database.](javascript:void(0)) [And actually this query is quite mild in complexity compared to some](javascript:void(0)) [of the LINQ queries I've worked with in the past.](javascript:void(0)) [But let me show you one more thing.](javascript:void(0)) [Suppose that in addition to displaying the restaurant name, city and country,](javascript:void(0)) [we also want to display how many reviews that we have for that restaurant.](javascript:void(0)) [There's a couple different ways to do that but I want to show you one of the most performant way](javascript:void(0)) [and also a popular approach to this.](javascript:void(0)) [And that is if the view needs to know all this information including the number of reviews](javascript:void(0)) [and perhaps the model that we build for should include the number of reviews.](javascript:void(0)) [So I'm going to do a projection and not just hand back a list of restaurants to view,](javascript:void(0)) [it could use that list of restaurants to compute the number of reviews for each restaurant.](javascript:void(0)) [But I'm going to do a transformation.](javascript:void(0)) [Let's say give me a new object that includes r.Id, r.Name, r.City, r.Country,](javascript:void(0)) [and also a new property NumberOfReviews equals r.Reviews.Count](javascript:void(0)) [and just put my missing comma in here.](javascript:void(0)) [This is creating a new anonymous type that has properties named ID, Name, City and Country.](javascript:void(0)) [The C Sharp compiler will just give it those property names](javascript:void(0)) [if you don't specify something equals.](javascript:void(0)) [And we're also adding in this NumberOfReviews equals r.Reviews.Count.](javascript:void(0)) [I could take this and still pass it to my View.](javascript:void(0)) [But being an anonymously typed object, we have the problem](javascript:void(0)) [that we don't know its name, it's anonymous.](javascript:void(0)) [What would I put as a strongly typed model directive here?](javascript:void(0)) [Well, what I could do instead is create a new model.](javascript:void(0)) [I'll call it a View Model because it's dedicated to this View.](javascript:void(0)) [So, let's create a class called RestaurantListViewModel and I'm going](javascript:void(0)) [to give it those properties that we need.](javascript:void(0)) [I'll just paste in that code to give it the properties that I need: ID,](javascript:void(0)) [Name, City, Country, CountOfReviews.](javascript:void(0)) [And now, I'll use this class when I'm building a model.](javascript:void(0)) [Select new RestaurantListViewModel, ID equals r.Id, Name equals r.Name, City equals r.City,](javascript:void(0)) [Country equals r.Country and CountOfReviews will be equal to r.Reviews.Count.](javascript:void(0)) [So this is an entity that I've created, something that I'll add as a DbSet](javascript:void(0)) [to my DbContext to save in the database.](javascript:void(0)) [It's what I call a View Model, it's something that is going to carry the information along](javascript:void(0)) [that the View needs to display and respond to this request.](javascript:void(0)) [So let me take this class and we will open up the Index View for the HomeController](javascript:void(0)) [and we'll change the model from being Restaurant to being a RestaurantListViewModel.](javascript:void(0)) [And now one of the pieces of information that we can add here would include item.CountOfReviews.](javascript:void(0)) [Let me save everything and do a Build and refresh our home page,](javascript:void(0)) [and now we're getting a little more insight into each restaurant,](javascript:void(0)) [a little more information out here.](javascript:void(0)) [View models, by the way, are a topic that we will return to talk about again.](javascript:void(0)) [They're quite useful because, very often, a page that you need](javascript:void(0)) [to display requires more information that one of your entities can have.](javascript:void(0)) [So you need to aggregate some information from a couple different places,](javascript:void(0)) [a couple different sources, wrap it all up into one object to hand over to the view](javascript:void(0)) [so it can do its job very efficiently.](javascript:void(0)) [And it's very easy to build a View Model.](javascript:void(0)) [You can do a projection with the select operator.](javascript:void(0)) [You can use a framework like AutoMapper.](javascript:void(0)) [AutoMapper is very good at copying data from one object into another object.](javascript:void(0)) [But what I want to focus on is just to show you that in addition](javascript:void(0)) [to using this pretty comprehension query syntax,](javascript:void(0)) [it's entirely legal and we can see it works well.](javascript:void(0)) [We could rate the same query using extension methods.](javascript:void(0)) [So I could say model equals db.Restaurants.](javascript:void(0)) [and let's do the OrderBy.](javascript:void(0)) [Now I need to pass a lambda expression to specify how to OrderBy.](javascript:void(0)) [So I'll say given a restaurant r, r goes to r.Reviews.average,](javascript:void(0)) [I need another lambda expression here that says given a review called review,](javascript:void(0)) [make you sure you average review.rating.](javascript:void(0)) [And in this case, to make this descending, what I should really invoke is not OrderBy,](javascript:void(0)) [I should invoke OrderByDescending into different extension method,](javascript:void(0)) [different LINQ operator that's available.](javascript:void(0)) [And now we just need to select.](javascript:void(0)) [I could say select, given a restaurant r, r goes to and let me try to save myself some typing,](javascript:void(0)) [I'm going to take this bit of code, new RestaurantListViewModel and try to paste it](javascript:void(0)) [into my Select statement here and highlight this, Control K, Control U to uncomment](javascript:void(0)) [and then close things off with a parenthesis and semicolon.](javascript:void(0)) [Control K, Control D to do some formatting which didn't really work in this case.](javascript:void(0)) [Sometimes it's not quite smart enough to know how you wanted something formatted.](javascript:void(0)) [I'll just select everything, tab this over.](javascript:void(0)) [But that is the equivalent query in the comprehension syntax.](javascript:void(0)) [There is no reason to choose these based on performance.](javascript:void(0)) [It's really which syntax do you like better and what operators do you need](javascript:void(0)) [because again some operators like Take and Skip,](javascript:void(0)) [you can only invoke them using this extension method syntax.](javascript:void(0)) [So I could say just give me the first ten restaurants that are in the database.](javascript:void(0)) [And if I do a Build right now and refresh, we should get the same result](javascript:void(0)) [because we have the exact same query that we had before.](javascript:void(0)) [And now, the one scenario we haven't looked at quite yet is filtering.](javascript:void(0)) [Let's look at that next.](javascript:void(0))

* + [A Search Filter](javascript:void(0))

[Let's adds some filtering to our query with a where clause.](javascript:void(0)) [But instead of using a where operator, in a hard coded value like I did in the slide](javascript:void(0)) [that introduced you to LINQ, let me add a parameter because a user might come](javascript:void(0)) [to the home page searching for a specific restaurant.](javascript:void(0)) [So we will allow them to enter a searchTerm and I'll provide a default value of this for null,](javascript:void(0)) [which is slightly redundant because if the MVC runtime doesn't find a searchTerm](javascript:void(0)) [in the request, it will pass along a null for the string parameter.](javascript:void(0)) [But there is a unit testing project in this solution that is invoking the index action](javascript:void(0)) [and the unit test and it's not passing any parameters.](javascript:void(0)) [So I'm just going to provide a default value here](javascript:void(0)) [so that the compiler will be happy with that unit test.](javascript:void(0)) [And now we can use that with a Where operator.](javascript:void(0)) [I can say Where, given a restaurant r, r goes to and then we can check a searchTerm equals null.](javascript:void(0)) [If searchTerm equals null, everything will be true, we want all the restaurants,](javascript:void(0)) [we're not searching for any specific restaurant or r.Name starts](javascript:void(0)) [with whatever searchTerm was entered.](javascript:void(0)) [And now if I do a Build and refresh the Home application, we'll have the same result so far](javascript:void(0)) [because we didn't enter a searchTerm but I just want to verify](javascript:void(0)) [that it still works and it looks like it does.](javascript:void(0)) [And now I can come up here and say searchTerm equals S, and now we just get the restaurants](javascript:void(0)) [that starts with an S. So that looks good but of course the user might not know](javascript:void(0)) [to enter something in the query string.](javascript:void(0)) [How can we provide a bit of a user interface that allows them to do some filtering?](javascript:void(0)) [That will require a modification to the index view.](javascript:void(0)) [So lets' come in here and up above the list of cities, let's add a form.](javascript:void(0)) [Now, I could add this form using HTML.begin form but there's no real special parameters](javascript:void(0)) [that I need this form to have, so I'm just going to type in a form tag.](javascript:void(0)) [I do want to change one thing about the default though, the default method.](javascript:void(0)) [I want that to issue a get request but I don't need to specify a parameter for action.](javascript:void(0)) [If there's no action there, it's just going to go back to the URL it came from which is perfect](javascript:void(0)) [because we can show the index view, the user can type something in, hit the submit button](javascript:void(0)) [and will go right back to where we came from which is the index action of the HomeController.](javascript:void(0)) [I'll explain why I used the get method in just a second.](javascript:void(0)) [But let's have an input type equals submit, value equals Search By Name.](javascript:void(0)) [That will be the button that the user clicks when they want to do the search](javascript:void(0)) [and then we'll have an input type equals search and that is](javascript:void(0)) [where they can type in to perform the search.](javascript:void(0)) [And, of course, we'll have to give this a name](javascript:void(0)) [and remember ASP.NET MVC matches things up in the request by name.](javascript:void(0)) [Because we have a searchTerm parameter on the index action, I would want the name of this](javascript:void(0)) [to match, so the name of this should be searchTerm.](javascript:void(0)) [So let's save everything.](javascript:void(0)) [Do a build just to be safe and refresh the application to see what this looks like.](javascript:void(0)) [And it's not the optimum user interface but it should be functional.](javascript:void(0)) [If I go in and type a G and click Search](javascript:void(0)) [By Name then you'll notice we get the right result, we get Great Lake.](javascript:void(0)) [You can see in the query string, we have searchTerm equals G. And the reason that is](javascript:void(0)) [in the query string is because our form method equals get.](javascript:void(0)) [When you click on a submit button for a form that does a get request,](javascript:void(0)) [it has to wrap up all these input values and put them in the query string.](javascript:void(0)) [If instead we had used post and done a submit, all these values would be inside](javascript:void(0)) [that it should be message body, you wouldn't see them in the query string.](javascript:void(0)) [It's nice to use a get request here because by putting these things in the query string,](javascript:void(0)) [someone can copy this and paste it into an e-mail or they can bookmark it,](javascript:void(0)) [and they can always come back to this page and see all of the restaurants that start with a G](javascript:void(0)) [or an S or whatever they're looking for.](javascript:void(0)) [You do have to be careful about when you use get because you wouldn't want to use a get request](javascript:void(0)) [for a form that is submitting information to the server to save in the database,](javascript:void(0)) [so something that is processing a credit card or creating a restaurant](javascript:void(0)) [or editing a review, those are all post operations.](javascript:void(0)) [Post operations are write operations.](javascript:void(0)) [A search is a read operation.](javascript:void(0)) [It's perfectly safe to issue a get request or a search request.](javascript:void(0))

* + [Summary](javascript:void(0))

[In this module, we took our first steps with the Entity Framework.](javascript:void(0)) [We let it take our C Sharp classes and not only save objects in the database](javascript:void(0)) [but we also let it create the database for us.](javascript:void(0)) [This was all done using the DbSet and DbContext classes of the Entity Framework.](javascript:void(0)) [We also used EF Migrations to seed the database](javascript:void(0)) [with some initial restaurants and manage the schema.](javascript:void(0)) [Finally, I showed you a few different LINQ queries to give you an idea of what is possible](javascript:void(0)) [when querying a database with the Entity Framework.](javascript:void(0)) [In the next module, we're going to build on what we learned here to look at editing](javascript:void(0)) [and inserting data along with all the features you need in those scenarios,](javascript:void(0)) [features like validation and from the error messages.](javascript:void(0))

* Working With Data (Part II)
  + [Introduction](javascript:void(0))

[Hi. This is Scott Allen and this is the second module in this course focusing on working](javascript:void(0)) [with data and models in ASP.NET MVC.](javascript:void(0)) [In this module we'll continue working with the data base and the Entity Framework](javascript:void(0)) [and we'll build some views to edit and create data.](javascript:void(0)) [We'll also see how to validate incoming information and uncover more details](javascript:void(0)) [about the model binding and html helpers in ASP.NET MVC.](javascript:void(0))

* + [A Restaurant Controller](javascript:void(0))

[Thus far in the application we've been accessing the data base and using different queries](javascript:void(0)) [to show and search for restaurants on the home page of the application.](javascript:void(0)) [I think the next step would be to add the ability to create update,](javascript:void(0)) [edit and delete restaurants but I don't think we want this functionality on the home page,](javascript:void(0)) [so let's create a new section of the application to do this.](javascript:void(0)) [We need a restaurant area that will let us manage restaurants and what I'll do is come](javascript:void(0)) [into the layout view of the application](javascript:void(0)) [and let's replace the link that's currently pointing towards reviews and instead is going](javascript:void(0)) [to say Restaurant is going to go to the index action of the restaurant controller](javascript:void(0)) [which means we need a restaurant controller, we don't have a restaurant controller as yet.](javascript:void(0)) [So I'm going to right click the controllers folder,](javascript:void(0)) [say add a new controller this will be the RestaurantController and I'm going](javascript:void(0)) [to select a new template, I'm going to select the template MVC controller](javascript:void(0)) [with read/write actions and views using Entity Framework.](javascript:void(0)) [Essentially, the visual studio can create your controller and all the views that you need](javascript:void(0)) [if you tell-- first of all the model class which in this case will be restaurant,](javascript:void(0)) [I can just type the first few words to this and find it in the drop down list](javascript:void(0)) [and the data context class which in our case is going to be OdeToFoodDb](javascript:void(0)) [and now I'll let visual studio work its magic by just clicking ad and it will go](javascript:void(0)) [out it will create that controller, populate all the actions with all the necessary code](javascript:void(0)) [to save things and query things from the data base,](javascript:void(0)) [also give me all of the views that I need in my views folder.](javascript:void(0)) [And when it's done it will open up all these new files in the editor.](javascript:void(0)) [So now I can see that I have a restaurant controller.](javascript:void(0)) [Let's build the application and try it out.](javascript:void(0)) [The build was successful now if I refresh the home page we should have a restaurant link](javascript:void(0)) [and if I click that I'll go to the restaurant controller.](javascript:void(0)) [And there we can still see all of our restaurants now we're back](javascript:void(0)) [in the pre-generated view that shows the restaurants in a table but I should now be able](javascript:void(0)) [to read a new restaurant just by typing in a name and I'll give this restaurant a City](javascript:void(0)) [and click create, and we now have a new restaurant in the data base.](javascript:void(0)) [Let's see if we can edit this because it's actually Yellow Brick Bank Restaurant](javascript:void(0)) [and that seems to work.](javascript:void(0)) [Can we delete the restaurant?](javascript:void(0)) [Yes we can delete the restaurant too.](javascript:void(0)) [This was all thanks to the scaffolding that I selected when I created this controller,](javascript:void(0)) [it gave me a controller that instantiated an OdeToFoodDb, give me an index action](javascript:void(0)) [that will show all the restaurants.](javascript:void(0)) [If we go down here to the bottom it also provided the dispose method](javascript:void(0)) [to make sure things were cleaned up.](javascript:void(0)) [And in between were a number of actions like delete and edit that will save](javascript:void(0)) [and create and delete restaurants.](javascript:void(0)) [The idea again behind scaffolding is that it gives you a good starting point.](javascript:void(0)) [You cannot necessarily write an entire application just by scaffolding everything out.](javascript:void(0)) [There's very few applications that are that simple but it does give you a code base](javascript:void(0)) [to start working with and shows you some of the basic MVC patterns that we usually follow.](javascript:void(0)) [You can certainly go in and change the controller code, you can go in](javascript:void(0)) [and change the view code but I'm not going to drill into each of these methods just now](javascript:void(0)) [because what's ultimately going to happen is we're going to impellent many](javascript:void(0)) [of the same methods in another controller, the reviews controller](javascript:void(0)) [so that we can create and edit reviews.](javascript:void(0)) [But I will take a minute just to walk you](javascript:void(0)) [through one controller action the details action.](javascript:void(0)) [The details action is what is invoked when I click on the details link,](javascript:void(0)) [it just shows all the details oft the restaurant, not the prettiest UI](javascript:void(0)) [so certainly it's something that you'll look at changing but the way the controller works is](javascript:void(0)) [to accept an incoming ID parameter that's the restaurant that we have to look up](javascript:void(0)) [and then using the restaurants DbSet](javascript:void(0)) [on my OdeToFoodDb it will invoke the find method passing in that ID.](javascript:void(0)) [Find is one of those convenient methods that the Entity Framework provides when it's EPI,](javascript:void(0)) [it's basically saying I'll go out and find something by it's primary key value all you have](javascript:void(0)) [to do is give me the primary key value, if it does find that object,](javascript:void(0)) [great, and return to a restaurant.](javascript:void(0)) [No link queries or link operators or lambda expressions here required.](javascript:void(0)) [If it doesn't find that restaurant it will return null,](javascript:void(0)) [so one of the things we could do here is check for null, if we didn't find the null](javascript:void(0)) [or return HttpNotFound which is another method that will produce a result,](javascript:void(0)) [that essentially return to 404 to the browser.](javascript:void(0)) [If we were to come in to this application in look for slash restaurant slash 22](javascript:void(0)) [where there shouldn't be a restaurant with a 22 in here, so we get the 404 error message](javascript:void(0)) [which may or may not be what you want.](javascript:void(0)) [If it is uncommon that a user should click a link](javascript:void(0)) [for a restaurant it doesn't exist anymore then you might just go ahead and return a 404](javascript:void(0)) [but for some reason there's a lot of links out there and you delete a lot of restaurants,](javascript:void(0)) [you might consider returning some sort of custom view.](javascript:void(0)) [So I could write a view called NotFound.chhtml put into the reviews folder](javascript:void(0)) [and basically put a little message there that says, "Sorry, we couldn't find that restaurant"](javascript:void(0)) [and maybe provide a link that you can click on to go back and search for other restaurants,](javascript:void(0)) [but for now just for turning a NotFound the area is fine, if we do find the restaurant](javascript:void(0)) [and of course we'll pass that into the view and it will display](javascript:void(0)) [as much restaurant information as possible.](javascript:void(0)) [But one thing the details are missing is a list of any reviews that the restaurant might have.](javascript:void(0)) [That would be easy enough to add to this details view, I could certainly come down here somewhere](javascript:void(0)) [and have a for each statement and look through Model.Reviews and print out some information](javascript:void(0)) [about each review or list how many reviews it has or its average rating.](javascript:void(0)) [But I thought we could do something that would be a little bit of the beat and path.](javascript:void(0)) [I'd like to be able to click on a link here that says reviews and go to the reviews](javascript:void(0)) [for the restaurant and then have the ability to add reviews to that restaurant](javascript:void(0)) [or edit the existing reviews for that restaurant.](javascript:void(0)) [This will be a little bit off the beat and path because the built in scaffolding cannot create](javascript:void(0)) [that for me, it can be close but what we'll do is write most of it from scratch](javascript:void(0)) [so that we can actually see how some](javascript:void(0)) [of these data modification actions need to be implemented.](javascript:void(0)) [And along the way we'll be talking about model binding, model validation and learn some more](javascript:void(0)) [about how to work with the MVC framework.](javascript:void(0))

* + [Listing Reviews](javascript:void(0))

[Let's implement a listing of reviews for each restaurant.](javascript:void(0)) [Since we're going to not use this details view anymore I'm going to delete](javascript:void(0)) [that from my views folder and I'll also delete the details action for my restaurant controller.](javascript:void(0)) [And since we don't have a details action I need](javascript:void(0)) [to change the index view that's displaying all the restaurant to not have a link to details,](javascript:void(0)) [instead we will say here's where you can go to see the reviews for this restaurant](javascript:void(0)) [and that will go to the index action on the reviews controller.](javascript:void(0)) [And now there's a little set of bug here in the way that I've invoked to action link](javascript:void(0)) [because I've picked the wrong overload.](javascript:void(0)) [I expect that ID equals item.ID will go as the route values parameter.](javascript:void(0)) [Because it did before I just added one more thing here, the name of the controller to call,](javascript:void(0)) [why wouldn't it go to route values.](javascript:void(0)) [And just because of the way that the action link overloads](javascript:void(0)) [or design you'll notice I'm not passing this as route values,](javascript:void(0)) [I'm passing this as htmlAttributes.](javascript:void(0)) [And the string reviews is being passed as route values and I see this happen all the time I want](javascript:void(0)) [to be able to show you how to spot this particular bug.](javascript:void(0)) [If I save this view and refresh the browser and I hover over one of the reviews links that's](javascript:void(0)) [about to appear I'll see in the little chrome pop window that previous the url](javascript:void(0)) [that I do not have an ID in the url, instead I have restaurant query string length equal seven.](javascript:void(0)) [Anytime you see length equal seven you probably picked the wrong over load](javascript:void(0)) [of action link unless you really did mean to pass length in the query string.](javascript:void(0)) [The only thing I need to do to fix this is to pass an additional parameter](javascript:void(0)) [so that reviews becomes the controller name, ID equals item.ID goes to route values](javascript:void(0)) [and this last parameter goes to htmlAttributes.](javascript:void(0)) [htmlAttributes by the way, that's a way to add additional attributes](javascript:void(0)) [to that anchor tag that is being produced.](javascript:void(0)) [So if you wanted to give that anchor tag a special class](javascript:void(0)) [or title attribute you could specify another anonymously typed object here something](javascript:void(0)) [like new title equals foo and that will be emitted into the html.](javascript:void(0)) [The passing null here will suffice if I don't want to set any htmlAttributes](javascript:void(0)) [that should also fix the link if I come back out here and refresh the view.](javascript:void(0)) [Now, I can see that this looks like what I need, slash review slash index slash one or slash two,](javascript:void(0)) [so the ID is now in the url, very good.](javascript:void(0)) [But if I click that link we'll have an exception because we comment it](javascript:void(0)) [out our reviews controller in an earlier module.](javascript:void(0)) [What I want to do now is select everything with control A uncomment with control KU](javascript:void(0)) [and delete code that we don't need anymore, for instance we don't need this in memory data.](javascript:void(0)) [Really, the only thing we need is we'll just start back with the index view that's right here](javascript:void(0)) [at the very top of the controller.](javascript:void(0)) [And in fact we don't need this best review anymore.](javascript:void(0)) [We deleted that from the layout action in the previous module.](javascript:void(0)) [What we do need is a data source.](javascript:void(0)) [So let me go ahead and add an OdeToFoodDb private variable and instantiate](javascript:void(0)) [up here when the controller is created.](javascript:void(0)) [Now, we'll also go ahead and override dispose.](javascript:void(0)) [You can just type override, have this lists pop up.](javascript:void(0)) [Pick the method that you want to override and then type code into it.](javascript:void(0)) [This looks good.](javascript:void(0)) [And now, what we'll do is change the index action around.](javascript:void(0)) [So, the index action will receive an Id parameter.](javascript:void(0)) [And what can be a little bit confusing here is](javascript:void(0)) [that this Id parameter is not the Id for overview.](javascript:void(0)) [Typically, when you're working with an Id inside of the controller it's an Id for the entity](javascript:void(0)) [that that controller is centered around.](javascript:void(0)) [Instead this is an ID that represents a restaurant Id.](javascript:void(0)) [That's a little bit confusing.](javascript:void(0)) [I could in the index view say instead](javascript:void(0)) [of Id equals item.Id I could say restaurant Id equals item.Id](javascript:void(0)) [and pass restaurant Id along in the query string.](javascript:void(0)) [Remember the routing engine doesn't know anything about restaurant Id in the URL](javascript:void(0)) [so we'd have to go in the query string.](javascript:void(0)) [I could add a new route if I wanted a restaurant Id to appear in the URL.](javascript:void(0)) [But I want to show you another option.](javascript:void(0)) [I'm just trying to change the name of this from Id to restaurant Id and instead](javascript:void(0)) [of changing the routes or changing where this parameter is passed,](javascript:void(0)) [I'm putting it into the query string, I'm just going to alias.](javascript:void(0)) [I can do that with the bind attribute.](javascript:void(0)) [But bind allows you to do is tell the MVC Model Binder that when it goes to look](javascript:void(0)) [for the restaurant Id parameter value look for something called Id](javascript:void(0)) [which is not very obvious syntax I know,](javascript:void(0)) [but that's essentially what this attribute is going to do.](javascript:void(0)) [When it looks for restaurant Id it will look for with the name of Id.](javascript:void(0)) [And now I can paste on some code that will go out and find](javascript:void(0)) [that restaurant by its restaurant Id.](javascript:void(0)) [If it's not equal to null we will render a view with that restaurant.](javascript:void(0)) [Otherwise we'll return a four for error.](javascript:void(0)) [And at this point we don't have any views for the reviews controller.](javascript:void(0)) [So, we're going to need to add a view.](javascript:void(0)) [I'll add an index view.](javascript:void(0)) [It's going to use razor.](javascript:void(0)) [I wanted to be strongly typed against a restaurant review.](javascript:void(0)) [Even though we're passing in a restaurant I'm going to strongly type it to restaurant review](javascript:void(0)) [because I want the scaffolding to create as much of the HTML forming as possible.](javascript:void(0)) [We're going to need to go in and fix this up because we're actually passing](javascript:void(0)) [in a restaurant not a restaurant review.](javascript:void(0)) [But that's okay.](javascript:void(0)) [This is going to help me.](javascript:void(0)) [When I click add I'm going to do a couple things.](javascript:void(0)) [First of all let me copy this model directive because it's going to come in useful here just](javascript:void(0)) [in a minute and change the model to say, no, you don't need a restaurant review.](javascript:void(0)) [What you'll be receiving is a restaurant.](javascript:void(0)) [And that means I could change this H2 tag to say something like reviews for at Model.Name](javascript:void(0)) [and that will give me the restaurant name.](javascript:void(0)) [And now let me take that model directive actually and paste it right here](javascript:void(0)) [and take all the rest of those code out because this code](javascript:void(0)) [that the scaffolding generated is here towards iterating](javascript:void(0)) [through a collection of reviews and writing them out.](javascript:void(0)) [But the model for this view is a restaurant.](javascript:void(0)) [What I want to do is create a partial view that will take care of rendering this table for me](javascript:void(0)) [and the model for that partial view will be an IEnumerable of restaurant review.](javascript:void(0)) [Inside of the reviews folder I'll create a new partial view.](javascript:void(0)) [Let's call it Reviews.](javascript:void(0)) [Using a Razor I strongly type to restaurant review](javascript:void(0)) [and we'll create it as a partial view of course.](javascript:void(0)) [I'll click Add.](javascript:void(0)) [Actually, delete the model directive that's here and paste this in.](javascript:void(0)) [This gives me everything that I need.](javascript:void(0)) [Our model is going to be IEnumerable of restaurant review.](javascript:void(0)) [I wanted to create a table of these reviews.](javascript:void(0)) [I don't necessarily want it to display the restaurant Id because that's meaningless](javascript:void(0)) [to an end user so let's delete that column.](javascript:void(0)) [I'll also delete that down here where it's displaying the value](javascript:void(0)) [and I think we'll just go with an edit link.](javascript:void(0)) [So, delete the details and the delete link and save everything.](javascript:void(0)) [Let me do a build and see if this works.](javascript:void(0)) [Oh, it won't work yet because back in the index view we need to render that partial.](javascript:void(0)) [I'll do that ahead of the create link.](javascript:void(0)) [So, html.partial, please render the reviews view for me and pass in Model.Reviews.](javascript:void(0)) [I'll take the reviews that are associated with this restaurant and pass it in to](javascript:void(0)) [that partial view as the strongly type model.](javascript:void(0)) [It will take care or rendering the table for me.](javascript:void(0)) [And now, let me do another build just to make sure everything is out there.](javascript:void(0)) [And let's refresh this page.](javascript:void(0)) [And let me go to the reviews for this first restaurant that I know there is a review](javascript:void(0)) [in there because we see that the database with a review.](javascript:void(0)) [But when I click that link I'll get a very strange air.](javascript:void(0)) [This is something that you could spend a long time debugging](javascript:void(0)) [if you don't know what's going on.](javascript:void(0)) [Because the error is a little bit misleading it's telling you that you passed](javascript:void(0)) [in a restaurant to this partial view.](javascript:void(0)) [But the partial view needs an IEnumerable of Restaurant Review.](javascript:void(0)) [And then you'll go back and look at your code, and you'll say,](javascript:void(0)) [but I'm not passing a restaurant to this partial view.](javascript:void(0)) [I'm passing in at Model.Reviews which should be a collection of restaurant review.](javascript:void(0)) [What is debug?](javascript:void(0)) [What's broken about the runtime here?](javascript:void(0)) [But the problem is not that you're passing the wrong thing.](javascript:void(0)) [The problem is that these reviews property is going to be null by default.](javascript:void(0)) [And this is something that you run into with the entity framework.](javascript:void(0)) [There's several different ways to solve this problem.](javascript:void(0)) [So, just to make it clear let me open up the Restaurant class and tell you](javascript:void(0)) [that in the controller when we do Restaurants.Find](javascript:void(0)) [in passing an Id what the entity framework will do is happily load](javascript:void(0)) [up these properties, Id, Name, City and Country.](javascript:void(0)) [It won't touch reviews because that's information that's in a different table.](javascript:void(0)) [There's a one to many relationship between restaurant and reviews.](javascript:void(0)) [And the entity framework just doesn't load up this associated child collections automatically.](javascript:void(0)) [At least not in this scenario the way I have things to find here.](javascript:void(0)) [There's at least six different ways to solve this problem.](javascript:void(0)) [I'll show you the easiest one which is just to add the virtual keyword here.](javascript:void(0)) [Now, what the entity framework will do when you spell virtual correctly is that it will](javascript:void(0)) [at runtime create a little wrapper for this restaurant class](javascript:void(0)) [and intercept recalls to the reviews property.](javascript:void(0)) [So that when you go to get to the reviews](javascript:void(0)) [for a restaurant it will make sure that it loads them up for you.](javascript:void(0)) [And that requires a second query to the database.](javascript:void(0)) [In other words Db.Restaurants.Find will just pull in the restaurant information.](javascript:void(0)) [Later in the view when we say Model.Reviews behind the scenes the entity framework will](javascript:void(0)) [issue a second query to pull in those reviews.](javascript:void(0)) [And you might say, but that's data access from a view](javascript:void(0)) [and I'll say that's fine because it's implicit.](javascript:void(0)) [I'm not writing explicit code data access code to load something.](javascript:void(0)) [It's just happening behind the scenes for me.](javascript:void(0)) [But some people will worry that there are two queries instead](javascript:void(0)) [of one to get all this information.](javascript:void(0)) [And for now, I'll just point you to some documentation that if you read](javascript:void(0)) [through it will explain how to eagerly load related information at the time you do a query.](javascript:void(0)) [I can get you down to just a single query to pull in a restaurant and a bunch of reviews.](javascript:void(0)) [As I said there's a number of different ways to solve this.](javascript:void(0)) [But just having the virtual keyword there is one of the easiest solutions](javascript:void(0)) [if you're not too worried about performance.](javascript:void(0)) [So, with the virtual keyword in place on a Restaurant Reviews, now I've done a build.](javascript:void(0)) [Let me go in and refresh this.](javascript:void(0)) [And this area should go away.](javascript:void(0)) [So a very non intuitive error what is going now because we have some reviews here](javascript:void(0)) [and it should also work when there are on reviews](javascript:void(0)) [because behind the scenes the entity framework is making sure it gives you some collection even](javascript:void(0)) [if it's an empty collection.](javascript:void(0)) [Now, again, our display isn't the prettiest display in the world.](javascript:void(0)) [It would be nice not to have ReviewerName, all this one word.](javascript:void(0)) [We'll come back and fix that later when we start looking](javascript:void(0)) [at some annotations that you can add to a model.](javascript:void(0)) [For right now I want to focus on creating a new review.](javascript:void(0))

* + [Creating a Review](javascript:void(0))

[Let's turn our attention to creating a review.](javascript:void(0)) [When I click on this Created New link to create a review I'm going to need](javascript:void(0)) [to know the restaurant to associate the review with.](javascript:void(0)) [So, in the index action I will add a parameter here](javascript:void(0)) [that will tell the routing engine, please pass along.](javascript:void(0)) [A parameter called RestaurantId set it to the Restaurants Id property.](javascript:void(0)) [And I didn't pick the correct overload of action link here.](javascript:void(0)) [I can see that's going into route values.](javascript:void(0)) [So the routing engine will say, restaurant Id, I don't see that in the URL.](javascript:void(0)) [So, I'll place it in the query string.](javascript:void(0)) [This then should give us the proper link.](javascript:void(0)) [Now, what I need in the reviews controller is an action](javascript:void(0)) [that will respond to a Get request for create.](javascript:void(0)) [This is a very simple action.](javascript:void(0)) [It just needs to return a view that contains everything the user needs](javascript:void(0)) [to type in to create a review.](javascript:void(0)) [An early response to a get request and even though the restaurant Id will be](javascript:void(0)) [in the query string and I can take it](javascript:void(0)) [as a parameter here I don't actually use it in this create action.](javascript:void(0)) [But I'm going to leave the parameter there just so we know that it's available](javascript:void(0)) [in the query string and in the future if we create perhaps some sort of view model](javascript:void(0)) [that will provide some defaults for some of the RestaurantReview fields.](javascript:void(0)) [We can always populate it with that restaurant ID.](javascript:void(0)) [But really, this is all we need in the Create Action for this simple scenario.](javascript:void(0)) [I do need to create view so I will add a view called Create using Razor,](javascript:void(0)) [strongly typed against the RestaurantReview](javascript:void(0)) [and this time I will select the Create Scaffold template and we'll get a view](javascript:void(0)) [that is strongly typed to RestaurantReview.](javascript:void(0)) [It's using the BeginForm helper that will post back this form when the user clicks Save.](javascript:void(0)) [It'll post back to the same URL it came from.](javascript:void(0)) [So if we are on /Reviews/Create?restaurantId=3, we'll post back to that same location.](javascript:void(0)) [We'll still have a restaurant ID in the query string](javascript:void(0)) [and here's a little trick about BeginForm.](javascript:void(0)) [What Html.BeginForm will do is return an object that implements iDisposable and that means](javascript:void(0)) [that we can use it with a C# using statement.](javascript:void(0)) [Typically, iDisposable objects are objects that hold on to precious resources](javascript:void(0)) [like database connections and you want to dispose them, and an easy way to do that is](javascript:void(0)) [to wrap something that's iDisposable with a using statement and create a code block.](javascript:void(0)) [At the bottom of the code block, the C# compiler will emit the proper code](javascript:void(0)) [to automatically call Dispose on that object.](javascript:void(0)) [In this case, BeginForm isn't really hanging on to any precious connections](javascript:void(0)) [but when you call Dispose on that object, that's when it will emit the closing form tag.](javascript:void(0)) [So that's just some information on how BeginForm works](javascript:void(0)) [and why we're placing it inside of a using statement.](javascript:void(0)) [Inside of the form, we'll have some validation messages.](javascript:void(0)) [We'll see those in this module.](javascript:void(0)) [We also have some literal text that we could make a little bit prettier instead](javascript:void(0)) [of RestaurantReview all one word, I could give this a legend of New Review.](javascript:void(0)) [Then we have labels and editors for everything the user needs to fill in, the rating, the body,](javascript:void(0)) [the Reviewer Name and oops, here's restaurant ID.](javascript:void(0)) [So sometimes the scaffolding is a little eager and emits things](javascript:void(0)) [that you don't necessarily want these or to edit.](javascript:void(0)) [We're going to delete that.](javascript:void(0)) [Sometimes it is not so eager and it doesn't emit things that you want it to,](javascript:void(0)) [but in this case we just had to delete something.](javascript:void(0)) [That should be enough so that if I do a build and make sure everything is saved,](javascript:void(0)) [if I refresh this page, to get the new Create New link that has a restaurant ID in it.](javascript:void(0)) [I should be able to click this and here we're on a page where I can enter a rating,](javascript:void(0)) [I can enter a body, and I could enter a Reviewer Name and click Create.](javascript:void(0)) [That'll do a post back.](javascript:void(0)) [So we need a controller action on the Reviews controller](javascript:void(0)) [that will respond to that post request.](javascript:void(0)) [That action will look like this.](javascript:void(0)) [We respond to a post request and we take a RestaurantReview as a parameter.](javascript:void(0)) [That's where the model binder in MVC will jump in and say, "Oh, you need a RestaurantReview."](javascript:void(0)) [I'll instantiate an instance of that type and then look through all the properties to see](javascript:void(0)) [if I can match them up with something in the request and it will find a rating,](javascript:void(0)) [it will find a body, it will push those values into it.](javascript:void(0)) [It will also find the restaurant ID so it will populate the restaurant ID property of a Review.](javascript:void(0)) [So it's associated with the proper restaurant and then inside of that Create Action,](javascript:void(0)) [the first thing you typically do is check to see if ModelState is valid](javascript:void(0)) [and if this flag is valid, returns false,](javascript:void(0)) [that means something went wrong with the model binding.](javascript:void(0)) [Something that was required isn't there.](javascript:void(0)) [Something is too long or something was too short.](javascript:void(0)) [It runs through all the validation rules associated with an entity](javascript:void(0)) [and tells you if everything worked or not.](javascript:void(0)) [We'll add some additional validation rules in this module but for now,](javascript:void(0)) [just know that if that returns false, what we want to do is not save the thing](javascript:void(0)) [in the database, something was invalid, instead, we'll return the Create View again](javascript:void(0)) [and let the user try to fix any problems that might have occurred.](javascript:void(0)) [Those Html.validation message for helpers will show validation messages to the user.](javascript:void(0)) [We'll see that here in just a second.](javascript:void(0)) [But if ModelState is valid, that means everything was proper.](javascript:void(0)) [So we will tell the Entity Framework to please add this review to its reviews collection](javascript:void(0)) [and then the Entity Framework doesn't actually save anything](javascript:void(0)) [to the database until you call SaveChanges.](javascript:void(0)) [At that point, it will issue Insert or Update or Delete Statements or all three,](javascript:void(0)) [depending on how many objects you changed to reconcile the changes that you made](javascript:void(0)) [in memory with what's in the database.](javascript:void(0)) [In this case, it should just issue a single Insert statement to insert a new review](javascript:void(0)) [into the Reviews Table and if that works, we will redirect the user back to the index action.](javascript:void(0)) [We don't want to let them sit there on the Create View that was the result of a post](javascript:void(0)) [because they might hit Refresh and create another review accidentally](javascript:void(0)) [or redirect them back to another page where they can see that review.](javascript:void(0)) [And now I should have everything that I need if I do a Build where I can come to this page](javascript:void(0)) [and click Create and there I can see my new review in the list.](javascript:void(0)) [Let's try it with a different restaurant.](javascript:void(0)) [Let's see if we can add a review for a restaurant that doesn't have any reviews yet.](javascript:void(0)) [And that seems to work.](javascript:void(0)) [And now before we start talking about validation and how to add additional validation rules,](javascript:void(0)) [let's finish off the edit scenario.](javascript:void(0))

* + [Editing a Review](javascript:void(0))

[For editing a review, we already have a link specified here that will take us](javascript:void(0)) [to an edit action and pass along the ID of the review to edit.](javascript:void(0)) [All we need to do to respond politely to that request is](javascript:void(0)) [to implement an Edit Action that responds to an HttpGet.](javascript:void(0)) [All we need to do here is take that ID parameter, lookup a review using](javascript:void(0)) [that primary key and then return a view that will allow the user to edit that.](javascript:void(0)) [We need the view so I will right click and add a view called Edit using Razor,](javascript:void(0)) [strongly type to a RestaurantReview.](javascript:void(0)) [This time the Scaffold template will be Edit.](javascript:void(0)) [This is going to look very similar to the Create View.](javascript:void(0)) [It still has a BeginForm.](javascript:void(0)) [It still has validation summaries.](javascript:void(0)) [It still has a legend that we don't quite like so we can say Edit Review.](javascript:void(0)) [It does have something a little bit different here.](javascript:void(0)) [It has a hidden input for the model.](javascript:void(0)) [That hidden input will allow us to track the ID of the review that's being edited](javascript:void(0)) [and it's hidden because the user doesn't need to see it.](javascript:void(0)) [We just need to preserve that piece of state between the requests,](javascript:void(0)) [and then we have the labels and the editors for Rating, Body, Reviewer Name](javascript:void(0)) [and ah, here again is restaurant ID.](javascript:void(0)) [This time I'm going to delete it again but we also need to keep track of this restaurant ID](javascript:void(0)) [because when we update this review, we're going to need](javascript:void(0)) [to properly populate the restaurant ID field.](javascript:void(0)) [What I can do is just add another Html.HiddenFor that says given a model called Model,](javascript:void(0)) [please give me a hidden input for restaurant ID.](javascript:void(0)) [There are some other approaches that we could use for this but this makes it easy](javascript:void(0)) [and now when the user clicks on the Submit button,](javascript:void(0)) [we will have to respond to that post request.](javascript:void(0)) [So back in the Reviews Controller, I'll implement an edit action that responds](javascript:void(0)) [to a post that's very similar to create but it has one little difference.](javascript:void(0)) [It's still going to take a RestaurantReview.](javascript:void(0)) [The MVC model binder is still going to populate that object](javascript:void(0)) [with things that it finds in the request.](javascript:void(0)) [So again, we don't have to look in the query strings or post of form values ourselves.](javascript:void(0)) [The model binder will take care of all that for us.](javascript:void(0)) [We'll still check if ModelState is valid.](javascript:void(0)) [Return to Edit View if ModelState is not valid because that will allow the user](javascript:void(0)) [to fix any problems that they have but if it is valid, we're going to come in](javascript:void(0)) [and use a slightly different API and I do need to bring in a namespace here,](javascript:void(0)) [System.Data, for this to be legal C# code.](javascript:void(0)) [But now that it's legal, I can tell you about this entry API on the db context.](javascript:void(0)) [The entry API is a way to tell the Entity Framework, "Here is a review that I want you](javascript:void(0)) [to start tracking so you can make changes in the database](javascript:void(0)) [for this review but it's not a new review.](javascript:void(0)) [I don't want it inserted into the database.](javascript:void(0)) [Instead, this is a review that's already in the database.](javascript:void(0)) [I just want you to take ownership of this object and treat it as if it had some new data inside."](javascript:void(0)) [That's the purpose of EntityState.Modified.](javascript:void(0)) [Now, when I call SavesChanges, you will update that record](javascript:void(0)) [in the database, the record that has the same ID.](javascript:void(0)) [In the Create Action, we add a review and EF will insert the record.](javascript:void(0)) [In the Edit Action, we attach an existing review with that entry, set the state to modified](javascript:void(0)) [and EF will update the record, and once that is successful, again, we'll redirect the user back](javascript:void(0)) [to the index action where they should be able to see the effect of their edit.](javascript:void(0)) [So after a build, let me come in and edit this review and because I'm really excited](javascript:void(0)) [about this restaurant, I will add extra exclamation points and click Save](javascript:void(0)) [and I can see that the edit worked.](javascript:void(0)) [And we're not going to go into validation just yet.](javascript:void(0)) [Before we do that, I want to tell you about some serious security implications of model binding,](javascript:void(0)) [some things that you need to know about.](javascript:void(0))

* + [Mass Assignment](javascript:void(0))

[You do need to be careful with model binding in MVC.](javascript:void(0)) [The model binder has a brute force algorithm that will try to move anything and everything](javascript:void(0)) [into your model that it can possibly move.](javascript:void(0)) [This means you might values you didn't expect in a model which can lead](javascript:void(0)) [to security related problems you might not have thought about.](javascript:void(0)) [Let's pretend that once a review is created, no one should be allowed](javascript:void(0)) [to edit the Reviewer Name of a review.](javascript:void(0)) [If you didn't watch this video, you would think that would be very easy.](javascript:void(0)) [All I need to do is take Reviewer Name out of my form.](javascript:void(0)) [If there's no more editor for Reviewer Name and I don't see it on the form,](javascript:void(0)) [how can I possibly change the Reviewer Name?](javascript:void(0)) [Well unfortunately, there's all sorts of tools out there that you can use to modify data](javascript:void(0)) [in a request or just create a request with arbitrary data and send it off to a server.](javascript:void(0)) [People use these tools all the time to break into servers](javascript:void(0)) [and send people information they didn't expect.](javascript:void(0)) [In fact, we don't really need a fancy tool here, all I need to do to get something](javascript:void(0)) [into reviewer name is to add a parameter to the query string.](javascript:void(0)) [I'll add reviewer name equals hack.](javascript:void(0)) [We usually get request for this form and now has that value in the query string and I don't need](javascript:void(0)) [to save anything here, all I want to do is get that reviewer name in there.](javascript:void(0)) [If I click save, we now have a new reviewer name.](javascript:void(0)) [And the problem here is that the model binder didn't know about my forum, about my view code,](javascript:void(0)) [it didn't know that I didn't want a reviewer name to be edited.](javascript:void(0)) [All it knows is that the restaurant review has a reviewer name property](javascript:void(0)) [and that the request had a reviewer name value so it moved the value into the property.](javascript:void(0)) [This problem is often referred to by one of two names.](javascript:void(0)) [Some people call it over posting that is we created more values than you expected](javascript:void(0)) [in the request but it also goes by the name mass assignment](javascript:void(0)) [because the model binder just assigns all the values it can](javascript:void(0)) [without regard to how you wrote your view code.](javascript:void(0)) [If someone knows the right syntax, they can move anything into your model.](javascript:void(0)) [There's a number of different solutions to this problem.](javascript:void(0)) [I'll show you one quick and easy solution probably not the best solution but if you want](javascript:void(0)) [to see more alternatives, you can read the post on my blog](javascript:void(0)) [where I cover this topic in more detail.](javascript:void(0)) [For now, let me show you one quick easy solution which is to use that bind attribute again.](javascript:void(0)) [We use this before when we aliased a parameter name.](javascript:void(0)) [This time, I'll show you some of the other things that you can pass in the bind,](javascript:void(0)) [they include a list of things to exclude and a list of things to include.](javascript:void(0)) [You typically pick one or the other.](javascript:void(0)) [Exclude is a black list.](javascript:void(0)) [I can say please exclude the following fields, reviewer name.](javascript:void(0)) [And if I had additional field here, I could just separate them by commas](javascript:void(0)) [but that's the one thing I want to be excluded when you do the model binding.](javascript:void(0)) [You can also use include and that's a white list,](javascript:void(0)) [I just list all the things that you do want to be bound.](javascript:void(0)) [But exclude works for me well here because I only need to exclude one thing.](javascript:void(0)) [Another common approach to this problem is to define a view model.](javascript:void(0)) [This time, an input view model and the view model would only have the properties you expect](javascript:void(0)) [to receive from the user, nothing else.](javascript:void(0)) [You'll use the view model as the parameter to your action and then you'll need](javascript:void(0)) [to copy the values into a real entity to update the database.](javascript:void(0)) [But by only having the values you expect from the user in the view model,](javascript:void(0)) [you can be sure you don't pick up anything unexpected from one of your shadier customers.](javascript:void(0)) [Also, related to security, make sure to watch the security module and learn](javascript:void(0)) [about cross site request forgeries.](javascript:void(0)) [Cross site request forgeries are a dangerous settle attack you should worry](javascript:void(0)) [about when posting form data and we'll look at the attack](javascript:void(0)) [and how to avoid a cross site request forgery in the security module later in the course.](javascript:void(0))

* + [Validation Annotations](javascript:void(0))

[Currently, when we edit a review, we could run into a couple problems.](javascript:void(0)) [There's nothing that constrains the rating to any sort of scale so if someone wanted](javascript:void(0)) [to save a rating of 1000 into the database, we would let them.](javascript:void(0)) [There's also nothing that requires them to enter a body or restricts them from entering a body](javascript:void(0)) [with 10,000 exclamation points in it.](javascript:void(0)) [These are all types of validations that we can do very easily with the MVC framework.](javascript:void(0)) [Before we look at them, let me just add a couple of things back into the project.](javascript:void(0)) [Let's remove the bind attribute for right now, I actually want people](javascript:void(0)) [to be able to edit the reviewer name.](javascript:void(0)) [I'll hit control Z in the edit view to bring back the editor for the reviewer name.](javascript:void(0)) [And now, let's go over and look at restaurant review and apply some data annotations.](javascript:void(0)) [So, one annotation that you can apply is the range annotation.](javascript:void(0)) [I can say that this rating has to be between one and ten and now, when we do model binding](javascript:void(0)) [in ASP.NET MVC, the model binder after it moves property values into this properties,](javascript:void(0)) [it also exercises any validation logic that is applied through this attributes.](javascript:void(0)) [So it defines that the rating is 100, it will tell you that the model state is not valid](javascript:void(0)) [and then you can let the user try this again.](javascript:void(0)) [There's also a required attribute although putting this](javascript:void(0)) [on an integer property is a little bit redundant because to the model binder,](javascript:void(0)) [an integer property is required by default because integers and C sharp are value types.](javascript:void(0)) [That means they cannot be null.](javascript:void(0)) [And when the model binder comes across a value type like int or date time, long and decimal,](javascript:void(0)) [it automatically requires a value to be present in the request for that.](javascript:void(0)) [We can also come in and tell the body-- tell the model binder that the body should be required](javascript:void(0)) [and we can also give it a maximum string length, let's say 1,024 characters.](javascript:void(0)) [In addition to this attributes, there's also attributes](javascript:void(0)) [to apply a regular expression validation, there's attributes to compare two properties](javascript:void(0)) [and there's an attribute to do a remote validation.](javascript:void(0)) [Actually, call back to the server and check your property value](javascript:void(0)) [as the user is typing into a forum.](javascript:void(0)) [You can read all about these validation annotations on MSDN and MVC documentation.](javascript:void(0)) [So just having this attributes in place and doing a build,](javascript:void(0)) [we'll make something different happen in my application](javascript:void(0)) [and that something different might not be something that you expect right away](javascript:void(0)) [but that something different is this.](javascript:void(0)) [The entity framework is always checking the model that is in effect against the model](javascript:void(0)) [that you use to create the database including any migrations](javascript:void(0)) [that have been put into the database.](javascript:void(0)) [And at this point in time, it is detected that there is something different about the model](javascript:void(0)) [and if you think about it, some](javascript:void(0)) [of these attributes really do influence the database schema.](javascript:void(0)) [For instance, saying that Body is required is like saying I need a not null column.](javascript:void(0)) [Previously, this was a null column, you can insert null values into it.](javascript:void(0)) [Specifying a string length gives this a definite string length.](javascript:void(0)) [Previously, it was nvarchar max, now it has a maximum length of 1,024 characters.](javascript:void(0)) [If I want it to get pass this error, there's a couple different ways to do it.](javascript:void(0)) [There is a way to just tell the entity framework trust me, I know what I'm doing,](javascript:void(0)) [just work with me and move ahead.](javascript:void(0)) [Or, you can actually apply the migrations to get the database schema and sync with what is](javascript:void(0)) [in your C sharp code and that's what we'll do.](javascript:void(0)) [Remember, we have automatic migrations enabled so what I am able to do is just come in](javascript:void(0)) [and say let's update the database and I'll throw in the Verbose flag.](javascript:void(0)) [If I do this, it's going to fail the first time because migrations is smart enough to realize](javascript:void(0)) [that we're taking one column, the body column or choose to be nvarchar max](javascript:void(0)) [and we're making it smaller that could result in data loss.](javascript:void(0)) [But here, I can say basically, I know what I'm doing and I want to force this through and now,](javascript:void(0)) [we can see things like alter table RestaurantReviews, alter the column body](javascript:void(0)) [to make it nvarchar 1024, not null.](javascript:void(0)) [And then we rerun the seed method, that mens all my data should still be in here.](javascript:void(0)) [We didn't loose anything.](javascript:void(0)) [And now, after I do a build one more time just to be safe,](javascript:void(0)) [I should be able to refresh the home page and we'll have a working application again.](javascript:void(0)) [So let's go into restaurants and look at one that has reviews](javascript:void(0)) [and see if we can edit this review and put the value of 1,000 in for the rating.](javascript:void(0)) [No, as soon as I tab off of that, it will give me a message](javascript:void(0)) [that the field rating must be between one and ten.](javascript:void(0)) [Notice, I didn't even click the Save button and that's because one of the nice things](javascript:void(0)) [that the editor for HTML helper will do for you is emit some Metadata into the input tag](javascript:void(0)) [that client-side JavaScript can pick up](javascript:void(0)) [and exercise the validation logic for you on the client-side.](javascript:void(0)) [So before the user even tries to submit these to the server,](javascript:void(0)) [we can tell them that they've done something wrong.](javascript:void(0)) [Same thing should happen here with Great, if I blank that out and try to click Save this time,](javascript:void(0)) [now, it can come back and tell me the body field is required.](javascript:void(0)) [That request actually didn't go to the server.](javascript:void(0)) [That was triggered on the client side by JavaScript.](javascript:void(0)) [Some of these validations require a little more](javascript:void(0)) [than just filling something in and tabbing out of the input.](javascript:void(0)) [But as soon as I start typing in here again, the validation message goes away.](javascript:void(0)) [And as a user, I feel happy and I can also fix the rating here.](javascript:void(0)) [And just to be clear, these validations are run on a client side,](javascript:void(0)) [they're also run on the server side so even if the user has JavaScript disabled for some reason](javascript:void(0)) [and a bad value comes into your server, the model binder will run the same validation rules](javascript:void(0)) [after it's done its model binding for this restaurant review.](javascript:void(0)) [And if any one of those rules failed, it will set model state is valid equals False.](javascript:void(0)) [And then when you re-render the view, things like the validation message four will pick](javascript:void(0)) [up those error messages and display them in the view.](javascript:void(0)) [In addition to data annotations for validation,](javascript:void(0)) [there's also a data annotations that can influence the display.](javascript:void(0)) [So if you remember, we had HTML helpers like display for and display name](javascript:void(0)) [for that we would use to write out things like the reviewer name here.](javascript:void(0)) [And the problem is that reviewer name as all one word isn't very pretty.](javascript:void(0)) [That's being written out by LabelFor ReviewerName.](javascript:void(0)) [So let me introduce you to just a couple of the display annotations here with the reviewer name.](javascript:void(0)) [One of the things we could use here is the display annotation](javascript:void(0)) [and one of the parameters we can pass for the display annotation is the name](javascript:void(0)) [that should be used when this is shown in the UI.](javascript:void(0)) [So I could just say this is perhaps user name as two words.](javascript:void(0)) [There's also a display format attribute.](javascript:void(0)) [With this attribute, you can specify things like the data format string](javascript:void(0)) [that should be applied that's quite useful when you want](javascript:void(0)) [to format numbers and dates for display.](javascript:void(0)) [For this one, what I could do is say that the NullDisplayText perhaps is anonymous.](javascript:void(0)) [We don't know who the user is and it always helps when I spell that correctly.](javascript:void(0)) [And now, let me do a build and come back out and refresh this page.](javascript:void(0)) [And we can see that Reviewer Name has changed to User Name.](javascript:void(0)) [And if I blank out the User Name to get rid of this ugly value that some malicious user put](javascript:void(0)) [in my database, I can see that User Name displays as anonymous when it has no value.](javascript:void(0)) [When I go into Edit it, I see the real value that's there which is empty,](javascript:void(0)) [it's just when we use the display for helper,](javascript:void(0)) [it's going to use that null value display text in place of a null string.](javascript:void(0)) [Again, there's many more Display attributes they can use, if you just want to poke around inside](javascript:void(0)) [of the System.ComponentModel.DataAnnotations name space,](javascript:void(0)) [I think you'll find a number of useful attributes in there.](javascript:void(0))

* + [Custom Validations](javascript:void(0))

[I'm sure you can think of many validation scenarios that are not covered](javascript:void(0)) [by the validation attributes provided with the framework.](javascript:void(0)) [This is when you can implement custom validation logic.](javascript:void(0)) [And there's a couple different approaches you can take](javascript:void(0)) [that will plug right into the framework.](javascript:void(0)) [One popular approach is to write a custom validation attribute.](javascript:void(0)) [Use this approach when you have validation logic that you want to apply to multiple models.](javascript:void(0)) [An example might look like this.](javascript:void(0)) [I'm going to temporarily paste some code inside of here,](javascript:void(0)) [that's not something we're going to keep in the project.](javascript:void(0)) [So I'm just going to keep it in the same file as a Restaurant Review](javascript:void(0)) [and we'll remove it after I explain what it is.](javascript:void(0)) [This is a maxWords attribute.](javascript:void(0)) [What I can do with this is restrict the number of words](javascript:void(0)) [that a user can enter into a string property.](javascript:void(0)) [It does this just by using a simple check and looking at the number of space characters.](javascript:void(0)) [So it's not an advanced validation.](javascript:void(0)) [I'm just really showing you how to go about implementing this approach.](javascript:void(0)) [And before I explain what it does, let me just show you how you could use it.](javascript:void(0)) [Let's say that we wanted to restrict Reviewer Name to be a single word.](javascript:void(0)) [And I could say, maxWords and you need to pass in a parameter](javascript:void(0)) [which is the maximum number of words.](javascript:void(0)) [So let's say just a single word for Reviewer Name.](javascript:void(0)) [And now, if I do a build and come out into the application,](javascript:void(0)) [I'm going to try to edit this User Name and put in Scott Allen](javascript:void(0)) [and click Save, User Name has too many words.](javascript:void(0)) [But if I go with just Scott, everything is fine.](javascript:void(0)) [So my validation attribute is working.](javascript:void(0)) [Let me remove it from here and I'll show you what it does.](javascript:void(0)) [First you derive from a base class ValidationAttribute.](javascript:void(0)) [The MVC runtime and the model binder, they respect ValidationAttibute.](javascript:void(0)) [So that automatically plugs you into the framework.](javascript:void(0)) [And when you derive from ValidationAttribute, you can override a method is valid.](javascript:void(0)) [When the model binders performing validation on your model, it will pass in the value](javascript:void(0)) [of the property where you place this attribute.](javascript:void(0)) [So here, it would pass in the Reviewer Name when it's validating a Restaurant Review.](javascript:void(0)) [And then it's up to me to inspect that value and see if it meets the proper criteria.](javascript:void(0)) [We're just doing a String.Split to create an array and it's going to split](javascript:void(0)) [up that string wherever it sees a space character then look at the Length](javascript:void(0)) [and see if that is created with maxWords.](javascript:void(0)) [So again, a very simple brute force validation.](javascript:void(0)) [I'm sure you can think of all sorts of edge cases where this would fail](javascript:void(0)) [but this is really more just about showing you how to implement custom validation logic.](javascript:void(0)) [You could have anything inside of here.](javascript:void(0)) [If the model passes validation, you can just return ValidationResult.Success.](javascript:void(0)) [But if there is a problem, you want to return a new ValidationResult](javascript:void(0)) [and associate an errorMessage with it.](javascript:void(0)) [And the error message you can create using another method that you inherit](javascript:void(0)) [from the base class FormatErrorMessage, we actually set the error to display string to use](javascript:void(0)) [for this attribute and the constructor.](javascript:void(0)) [And then back in is valid.](javascript:void(0)) [We'll build that string using Display Name.](javascript:void(0)) [And that's what ultimately produce the string, User Name has too many words.](javascript:void(0)) [It also allows other people to set a custom error message if they want.](javascript:void(0)) [You can do this actually on any attribute even the built in attributes.](javascript:void(0)) [Anyone of these attributes, I could walk up to them and say, errorMessage equals](javascript:void(0)) [and have a custom error message here that will display when the user is outside of that range.](javascript:void(0)) [And I could do that for maxWords too just because we're using FormatErrorString](javascript:void(0)) [and deriving from a validation attribute base class.](javascript:void(0)) [Now, this type of validation only happens on the server-side.](javascript:void(0)) [You can implement custom JavaScript code to plug in to the client-side validation framework.](javascript:void(0)) [You'll find many examples of this on the web, just remember you don't need](javascript:void(0)) [to duplicate all the validation logic on the client.](javascript:void(0)) [Some validations are difficult on the client.](javascript:void(0)) [If you at least have the required and length attributes and simple text like that,](javascript:void(0)) [you can give quick feedback for common errors that's going to help your users out quite a bit.](javascript:void(0)) [In addition to this validation attribute approach,](javascript:void(0)) [you can also have a model object implement and IValidatable Object Interface.](javascript:void(0)) [This is a good approach to use if you want to do a deep inspection of a model](javascript:void(0)) [because what I can do with this interface is use control period to tell Visual Studio that I want](javascript:void(0)) [to implement this interface and it will give me that validate method.](javascript:void(0)) [That's the method that you implement to provide the custom validation logic.](javascript:void(0)) [And it's inside of here where you have access to the entire model.](javascript:void(0)) [When you write a validation attribute, you only have easy access to a single property,](javascript:void(0)) [that's the property value where the attribute I was used and that's the property value](javascript:void(0)) [that the model binder will pass in to you.](javascript:void(0)) [You can get to other properties on the model but it's a bit messy.](javascript:void(0)) [With Validate, I'm inside my model object so it would be really easy for me](javascript:void(0)) [to exercise validation logic that looks at multiple properties, for instance,](javascript:void(0)) [I could say something like, if Rating is less than two and ReviewerName ToLower,](javascript:void(0)) [startsWith Scott and maybe we just have a trouble maker](javascript:void(0)) [on the site with the User Name of Scott.](javascript:void(0)) [And it gives everyone a bad and low rating.](javascript:void(0)) [We just want to prevent him from entering messages.](javascript:void(0)) [And what I could do here is use yield return](javascript:void(0)) [and create a new ValidationResult with an error message.](javascript:void(0)) [And otherwise, if everything succeeds, I don't need to yield return anything here,](javascript:void(0)) [that'll essentially produce an empty IEnumerable](javascript:void(0)) [and the MVC runtime will know there's no problems.](javascript:void(0)) [And now, if I build the application again and we come into edit a review and I have a User Name](javascript:void(0)) [of Scott and I tried to put in a Rating of one.](javascript:void(0)) [I'll save the error message.](javascript:void(0)) [Notice that this error message appears above any properties that's](javascript:void(0)) [because we didn't associate this validation error](javascript:void(0)) [with a specific property like User Name or Rating.](javascript:void(0)) [And in that case, it just gets associated with the model](javascript:void(0)) [and it will display in the Validation Summary.](javascript:void(0)) [If I open up the Edit view for a Review real quick, that error message is appearing here](javascript:void(0)) [in the Validation Summary which by default only shows errors that aren't associated](javascript:void(0)) [with a specific field in the model.](javascript:void(0)) [That's what the true parameter is telling the Validation Summary to do.](javascript:void(0)) [If you change that value over a false value, the Validation Summary will display all errors](javascript:void(0)) [on the models for all properties and also the model of all errors.](javascript:void(0))

* + [Summary](javascript:void(0))

[In this module, we added the ability to create, delete and edit restaurants](javascript:void(0)) [and also added the ability to create and edit reviews.](javascript:void(0)) [We looked at Model Binding, some Entity Framework APIs for data updates and saw how](javascript:void(0)) [to use data annotations to influence the display of the model information](javascript:void(0)) [and also to validate model information.](javascript:void(0)) [We saw two different approaches to custom validation that work with the Model Binder](javascript:void(0)) [and the MVC framework to automatically set model state and populate our error messages.](javascript:void(0)) [In the next module, we'll be turning out attention to JavaScript](javascript:void(0)) [and adding some Ajax features to the application.](javascript:void(0))

* AJAX and ASP.NET MVC
  + [Introduction](javascript:void(0))

[Hi, this is Scott Allen.](javascript:void(0)) [And in this module, we will look at adding AJAX features to an MVC application.](javascript:void(0)) [Specifically, we'll look at the AJAX helpers offered by the MVC framework.](javascript:void(0)) [We'll also spend time using jQuery and jQuery UI to work directly in script code.](javascript:void(0)) [I'll show you how client validation works, how to minify and bundle your scripts,](javascript:void(0)) [how to produce JSON from your application,](javascript:void(0)) [and what to do with JSON once you have it on the client.](javascript:void(0)) [I will do all these things while adding features to the application.](javascript:void(0))

* + [The Scripts](javascript:void(0))

[The letter J in AJAX stands for JavaScript which means we'll need](javascript:void(0)) [to send JavaScript to the client.](javascript:void(0)) [And I think it will be good to look at what Script Libraries are already available](javascript:void(0)) [in the project and how to manage those scripts to get them to the client](javascript:void(0)) [which will demonstrate some new features in the framework.](javascript:void(0)) [So inside of the Scripts folder in Solution Explorer,](javascript:void(0)) [you can see the JavaScript files are available when you create a new project.](javascript:void(0)) [We'll start at the top with \_references.js.](javascript:void(0)) [This file works with Visual Studio 2012 to give Studio a list of the files you commonly use](javascript:void(0)) [so that you can get better IntelliSense when writing JavaScript.](javascript:void(0)) [If we open up the file, we'll see it contains references inside of JavaScript comments.](javascript:void(0)) [These references look suspiciously like XML.](javascript:void(0)) [But these references are telling Visual Studio what files you commonly use.](javascript:void(0)) [We can see the list includes jquery, jquery-ui and more.](javascript:void(0)) [And now, Visual Studio will go out and find those files and parse them](javascript:void(0)) [and try to provide better IntelliSense for you.](javascript:void(0)) [It should know about method names and attributes of the objects](javascript:void(0)) [that you work with from these libraries.](javascript:void(0)) [If you were commonly working with additional libraries in the application,](javascript:void(0)) [you probably want to list those files here too.](javascript:void(0)) [Underneath of references is the jQuery library.](javascript:void(0)) [There's three versions of the core jQuery library file.](javascript:void(0)) [The first one has IntelliSense in the name.](javascript:void(0)) [You'll never use this file.](javascript:void(0)) [You'll never send it to the client.](javascript:void(0)) [This is just another aid for Visual Studio to generate intelligent prompts with IntelliSense.](javascript:void(0)) [When the references file says to reference jQuery,](javascript:void(0)) [Visual Studio will automatically find this file which has some additional comments inside](javascript:void(0)) [to make Visual Studio look smarter.](javascript:void(0)) [The jquery-1.7.1.js file, that's the core jQuery library that you can send to the client.](javascript:void(0)) [If you aren't familiar with the jQuery, then it is a JavaScript library](javascript:void(0)) [that gives us an API we can use on the client to select and manipulate DOM elements,](javascript:void(0)) [make asynchronous call to the server, and lots more.](javascript:void(0)) [jQuery supports all the modern browsers including Internet Explorer,](javascript:void(0)) [Chromes, Safari, Firefox, Opera.](javascript:void(0)) [And if you're new to jQuery, you'll see a few tips on this module.](javascript:void(0)) [This version of the jQuery file is very readable.](javascript:void(0)) [If you open it up, you can find lots of white space, well-formatted code, and comments.](javascript:void(0)) [The jQuery file with .min in the name is minified.](javascript:void(0)) [Minification is the process of going through a JavaScript file and making it small as possible](javascript:void(0)) [by removing unneeded white space, unneeded comments, and even shortening the names](javascript:void(0)) [of symbols and variables that are exposed to the outside world.](javascript:void(0)) [If you open that up, you'll see it's not very readable.](javascript:void(0)) [But it is smaller, and by making the file smaller, the browser will need less time](javascript:void(0)) [to download the file, which means, your page can load faster and users like fast pages.](javascript:void(0)) [However, we don't really need to touch this file because there's feature in ASP.NET MVC 4](javascript:void(0)) [that will automatically minify files for us](javascript:void(0)) [or pick up minified files as we'll see in just a bit.](javascript:void(0)) [In the Scripts folder, we also have jquery-ui which is a plug-in or extension for jQuery](javascript:void(0)) [that gives you UI widgets like dialog and accordion that you can use.](javascript:void(0)) [There's a minified and unminified version.](javascript:void(0)) [We also have jquery.validate, another plug-in for jQuery.](javascript:void(0)) [This one provides client side validation features.](javascript:void(0)) [The MVC framework actually relies on this library.](javascript:void(0)) [We also have knockout.js which you can use on the client to apply a model view--](javascript:void(0)) [view model design pattern in client script.](javascript:void(0)) [It provides features like declarative data by mean.](javascript:void(0)) [And then, there is modernizr, which as I mentioned early in the course,](javascript:void(0)) [is a library to detect and enable HTML5 features on a browser.](javascript:void(0)) [The files I skipped over all have unobtrusive in the mean, and these files are authored](javascript:void(0)) [by the MVC team and they serve as a bridge between ASP.NET MVC and jQuery.](javascript:void(0)) [In order for some MVC features to work like client side validation,](javascript:void(0)) [you need both the jQuery validation library and jQuery,](javascript:void(0)) [and the jquery.validate.unobtrusive script file because it takes metadata that's omitted](javascript:void(0)) [by HTML helpers like Editor 4 and feeds that data into jQuery validations](javascript:void(0)) [so it can apply the validation rules on the client.](javascript:void(0)) [We'll see that metadata in this module.](javascript:void(0)) [And now that we know what's here, let's look at how we send these script files](javascript:void(0)) [to the client in an efficient manner.](javascript:void(0))

* + [Managing Scripts](javascript:void(0))

[Libraries like jQuery and Modernizr are libraries that you typically use everywhere](javascript:void(0)) [in an application, which means you need a script reference on every page.](javascript:void(0)) [And as we've learned, it is the Layout views that are the place to go](javascript:void(0)) [when you have some common markup that you want on every page.](javascript:void(0)) [This application only has one Layout view.](javascript:void(0)) [It's in the Shared folder.](javascript:void(0)) [It's called \_Layout.](javascript:void(0)) [And at the very top of the Layout view, you can see a call to Script.Render.](javascript:void(0)) [And this somehow involves Modernizr.](javascript:void(0)) [Before we talk about the details of Scripts.Render, let me just assure you,](javascript:void(0)) [this will put a script tag referencing the Modernizr JavaScript library into the top](javascript:void(0)) [of the page here, inside of the head tag.](javascript:void(0)) [It is important to include some JavaScript libraries at the top of a page.](javascript:void(0)) [This is true for Modernizr.](javascript:void(0)) [Modernizr needs to start executing before the rest of your HTML goes to the client.](javascript:void(0)) [As we talked about in the first module of the course, Modernizr will make sure older browser](javascript:void(0)) [like IE6 are prepared to work with a new HTML5 elements like nav](javascript:void(0)) [and section which appear in this page.](javascript:void(0)) [And to do this, Modernizr has to start executing before those elements appear.](javascript:void(0)) [Other libraries and scripts don't need to appear at the top of every page.](javascript:void(0)) [You can instead include them at the bottom of a page.](javascript:void(0)) [And if we come down to the bottom of the Layout view, we can see another call to Script.Render.](javascript:void(0)) [This one somehow involves jQuery.](javascript:void(0)) [When possible, you should add scripts to the bottom of the page](javascript:void(0)) [because it can make a page perform better, or at least,](javascript:void(0)) [it looks like the page is performing better because the browser doesn't have to block](javascript:void(0)) [as it downloads the script and executes the code inside.](javascript:void(0)) [Web browsers tend to bring everything else to a halt when they encounter a script tag.](javascript:void(0)) [But by placing scripts at the bottom, we get more content to the client](javascript:void(0)) [as the HTML images will start to display and then finally,](javascript:void(0)) [the browsers sees the scripts and loads them at the end.](javascript:void(0)) [And now, let's go to the question of what is Scripts.Render doing?](javascript:void(0)) [And for that matter, what is Styles.Render doing?](javascript:void(0)) [Because it turns out, these two calls are closely related.](javascript:void(0)) [They both rely on new features and ASP.NET that can bundle and minify files for you.](javascript:void(0)) [Scripts.Render can give you a minified job script bundle.](javascript:void(0)) [Style.Render can give you a minified bundle of style sheets.](javascript:void(0)) [What's a bundle, you might ask.](javascript:void(0)) [Well, another way to boost the performance of a webpage is to combine files together,](javascript:void(0)) [so a browser doesn't need to download as many resources.](javascript:void(0)) [For example, if you have three style sheets to include in a page,](javascript:void(0)) [you might want to bundle them together into a single style sheet](javascript:void(0)) [so the browser only downloads one file instead of three.](javascript:void(0)) [But, during development, it's often nice to have three separate files.](javascript:void(0)) [They're probably creative for different purposes and it's easy to find](javascript:void(0)) [and maintain the code when it's nicely separated.](javascript:void(0)) [That provides a bit of tension because we went three files](javascript:void(0)) [for development, but a single file for download.](javascript:void(0)) [And this is where the bundling feature in ASP.NET steps](javascript:void(0)) [in because it can dynamically bundle files together at runtime.](javascript:void(0)) [To see this, let's open up the Global.asax.cs file, look in our friend,](javascript:void(0)) [the application start event where we can see a call in here to BundleConfig.RegisterBundles.](javascript:void(0)) [Up at the cursor there, and strike the F12 key to open that file up.](javascript:void(0)) [And here we arrive inside of a class](javascript:void(0)) [that is building various style bundles and script bundles.](javascript:void(0)) [There are two parts to a bundle.](javascript:void(0)) [First, you give the bundle a virtual path to reach the bundle.](javascript:void(0)) [In other words, if I open up a browser and come to this URL,](javascript:void(0)) [I'll receive all the files that are included in the bundle.](javascript:void(0)) [This is also the path or the identifier you use when you reference a bundle and want](javascript:void(0)) [to render it using Scripts.Render or Styles.Render.](javascript:void(0)) [The files that are included in the bundle are listed with calls to include](javascript:void(0)) [and you might want to include one file.](javascript:void(0)) [You can certainly have a one file bundle.](javascript:void(0)) [But you can also include multiple files.](javascript:void(0)) [You can specify a full name here.](javascript:void(0)) [You can also specify a place holder for a version number and you can use wild cards.](javascript:void(0)) [For instance, the first script bundle here that is basically including jQuery is going](javascript:void(0)) [to have a placeholder for the version.](javascript:void(0)) [That means if I upgrade jQuery from -1.7.1 to -1.8.0,](javascript:void(0)) [I don't need to come in and change my C# code.](javascript:void(0)) [The script bundle will automatically pick up the latest version from the file system,](javascript:void(0)) [whatever is in my Scripts folder, and it's smart enough to distinguish](javascript:void(0)) [between IntelliSense files, minified files, and unminified files.](javascript:void(0)) [So even though we have three versions of that core jQuery library, the IntelliSense,](javascript:void(0)) [the minified and the regular file, it's only going to pick the one that it needs](javascript:void(0)) [and it won't include jQuery more than once.](javascript:void(0)) [You can also use wild cards.](javascript:void(0)) [So down here in the jQuery validation bundle, you can see a call to jquery.unobtrusive,](javascript:void(0)) [that means go out and find all the files that start with jquery.unobtrusive.](javascript:void(0)) [But again, it's going to be smart enough not to include both the minified](javascript:void(0)) [and the unminified file, it can distinguish between those two.](javascript:void(0)) [So the first bundle, the jQuery bundle, that's a bundle of one file.](javascript:void(0)) [And that's okay because in bundling will add some other features to this,](javascript:void(0)) [it can minify files that aren't minified.](javascript:void(0)) [In the jQuery validation bundle, that's going to include at least two files,](javascript:void(0)) [jquery.unobtrusive and jquery.validate.](javascript:void(0)) [Once you've created a bundle which includes all of the files that you need,](javascript:void(0)) [you render it in markup using Scripts.Render or Styles.Render as we saw on the Layout view.](javascript:void(0)) [And you can see, it is using that virtual path,](javascript:void(0)) [that identifier that you selected for the bundle.](javascript:void(0)) [And I call it a virtual path because there's not really a file at that location.](javascript:void(0)) [But the MVC runtime is smart enough to intercept that request and route it through the bundling](javascript:void(0)) [and minification pictures here which will served up the files you need.](javascript:void(0)) [So once you've created a bundle, you render Scripts.Render or Styles.Render.](javascript:void(0)) [And these methods will behave differently depending on your application configuration.](javascript:void(0)) [So for example, back in Bundle.Config, there was a StyleBundle to find](javascript:void(0)) [for all of the jQuery UI style sheets.](javascript:void(0)) [You can see those quite a few of them because they factor them out based on features.](javascript:void(0)) [If you wand to use just a dialog, then all you need](javascript:void(0)) [to do is include the core style sheet and the dialog style sheet.](javascript:void(0)) [But if you use everything, you'd want to include all these style sheets.](javascript:void(0)) [I'm going to take this bundle and render it from the Layout view](javascript:void(0)) [because eventually we will be using jQuery UI.](javascript:void(0)) [So I just need to add a call to Styles.Render and pass that path in.](javascript:void(0)) [And now, let's run the application.](javascript:void(0)) [And I'll do a View Source.](javascript:void(0)) [And you can see, with my single call to Styles.Render,](javascript:void(0)) [I omitted a link for each style sheet into the page, almost 12 links or exactly 12 links.](javascript:void(0)) [So it added a style sheet link for every file in the bundle.](javascript:void(0)) [You might think that's not much of a bundling feature if we have individual links,](javascript:void(0)) [or certainly not combining things into a single download.](javascript:void(0)) [But this is because we are running the application in debug mode.](javascript:void(0)) [And in debug mode, you'll find it easier to debug scripts and styles](javascript:void(0)) [when they are not bundled up into a single download.](javascript:void(0)) [However, if I come over into the Web.config file, we can change how we're running.](javascript:void(0)) [I just need to find the debug setting which is right here at the top.](javascript:void(0)) [We'll change it from true to false so that we're running in release mode.](javascript:void(0)) [And I'll run the application again, and do a View Source.](javascript:void(0)) [And now, you can see there is a single link tag admitted since we're in a release mode.](javascript:void(0)) [And if I were to take this URL and place it into the browser to download this directly,](javascript:void(0)) [it will all get back as a css file, let's open it in Notepad.](javascript:void(0)) [And that's all of the css from all 12 of those files combined into a single file.](javascript:void(0)) [It's also been minified, so it's been made as small as possible.](javascript:void(0)) [There's no white space or comments to make this a larger download.](javascript:void(0)) [This bundling and minification features then can really decrease the amount of time needed](javascript:void(0)) [to load a page, and they're easy to use.](javascript:void(0)) [We'll be using them in this application as we send more scripts to the client just](javascript:void(0)) [by defining the new bundles and using Scripts.Render](javascript:void(0)) [to put those scripts in the client.](javascript:void(0)) [( Pause )](javascript:void(0))

* + [AJAX Helpers](javascript:void(0))

[Now that we know how to get scripts down to the client,](javascript:void(0)) [let's add some AJAXy features to the application.](javascript:void(0)) [We'll start with a homepage.](javascript:void(0)) [We currently have a search here which is working just fine,](javascript:void(0)) [but let's make things a little bit harder.](javascript:void(0)) [Instead of three restaurants, let's use 1,000 restaurants.](javascript:void(0)) [And the easiest way to do this is to see the database with a thousand restaurants.](javascript:void(0)) [I'll swing over into our Migrations folder and open up Configuration.cs.](javascript:void(0)) [Inside of here, you remember, is our Seed method.](javascript:void(0)) [And what I can do inside of here is just add a loop from 0 to less than 1,000.](javascript:void(0)) [And for each iteration of the loop, we'll just add another restaurant](javascript:void(0)) [that just has a name like 1, 2, 3, and 4.](javascript:void(0)) [All these restaurants are in no where USA.](javascript:void(0)) [Just having that code in place, if I open up the Package Manager Console,](javascript:void(0)) [I should be able to update the database which will execute that Seed method](javascript:void(0)) [and will have a lot more restaurant in the database.](javascript:void(0)) [Now, the Seed method is complete, so let's swing back into the application.](javascript:void(0)) [And now that we have all these restaurants, we're currently only taking the top ten.](javascript:void(0)) [It becomes much more important to be able to search.](javascript:void(0)) [We'll also need some paging, but we'll worry about that later.](javascript:void(0)) [Let's say I scroll halfway down the page and I search for restaurants.](javascript:void(0)) [Let's start with 77.](javascript:void(0)) [And the search still works, I can find restaurant 77 and 770.](javascript:void(0)) [But you'll notice what happens when I do the search, when I scrolled down into the page,](javascript:void(0)) [is I completely lose my scroll position.](javascript:void(0)) [And that's because we're doing a full request back](javascript:void(0)) [to the server withdrawing the entire page and it starts fresh.](javascript:void(0)) [It would be nice if I could search for restaurants and not lose my scroll position.](javascript:void(0)) [In other words, I just type in a search term, click Search,](javascript:void(0)) [and just this portion of the page refreshes.](javascript:void(0)) [You could think of this is a partial page update very similar](javascript:void(0)) [to what update panel could do in ASP.NET.](javascript:void(0)) [Let's add that feature, but first, I want to fix somethings up the way](javascript:void(0)) [that scripts are managed in this application right now.](javascript:void(0)) [What I want to do to simplify things is have a single script bundle that's rendered](javascript:void(0)) [at the bottom of every page from the Layout view.](javascript:void(0)) [And in order to do that, I'm going to need to fix up a few things that I've been scaffolded](javascript:void(0)) [into the data modification views like the view to edit a restaurant.](javascript:void(0)) [When we've been scaffolding up these views, we've been allowing the scaffolder](javascript:void(0)) [to include Scripts.Render at the bottom of this views or inside of the scripts section.](javascript:void(0)) [And it's rendering the scripts for everything that's needed for jQuery validation](javascript:void(0)) [and the clients side validation to work.](javascript:void(0)) [Basically, I want to take those scripts out of here.](javascript:void(0)) [If I just do a Search and Find in Files for section Scripts,](javascript:void(0)) [I'll find all the files where this is in place.](javascript:void(0)) [And what I'm going to do is coming to each one, and I'll leave the script section in place,](javascript:void(0)) [but I want to delete that Scripts.Render from each one.](javascript:void(0)) [And now that I've deleted eight instances of that script bundle,](javascript:void(0)) [let's close out all the documents that I opened, and come in to out App\_Start folder.](javascript:void(0)) [It's inside of here where we will define a new bundle in BundleConfig.cs](javascript:void(0)) [that will take care of all our script needs.](javascript:void(0)) [I'll paste the code in here at the top of the class.](javascript:void(0)) [We're essentially going to create bundles under ~/bundles/otf or OdeToFood.](javascript:void(0)) [And we'll include jquery, jquery-ui and all of the scripts we need](javascript:void(0)) [for unobtrusive JavaScript and jQuery validation.](javascript:void(0)) [And now, I can come in to the Layout view.](javascript:void(0)) [And at the bottom of the page, instead of rendering just jQuery,](javascript:void(0)) [we will render that script bundle.](javascript:void(0)) [And now, the application should behave the same.](javascript:void(0)) [We're just going to have more scripts inside of every page.](javascript:void(0)) [It's going to simplify things for us in this application.](javascript:void(0)) [And now, we can work on this feature in the Index view for the HomeController.](javascript:void(0)) [Inside of here, let me comment out the simple form that we have in place](javascript:void(0)) [and replace it with the call to Ajax.BeginForm.](javascript:void(0)) [Ajax.BeginForm is very similar to Html.BeginForm.](javascript:void(0)) [It writes out to form tag and you can tell what controller and action and route values to use.](javascript:void(0)) [But Ajax.BeginForm will make an asynchronous request to the server.](javascript:void(0)) [Html.BeginForm makes a synchronous request to the server that withdraws the entire screen.](javascript:void(0)) [Ajax.BeginForm makes an async request and it can withdraw just a portion of the screen.](javascript:void(0)) [All you need to do is tell the helper what to update on the page, and you could do this](javascript:void(0)) [with the AjaxOptions object that we pass it.](javascript:void(0)) [We tell the helper to make a get request and whatever comes back should replace the content](javascript:void(0)) [in the page that's currently identified by this UpdateTargetId.](javascript:void(0)) [We don't have an HTML element by that name, so I need to add one.](javascript:void(0)) [This element will delimit essentially the area that we want updated.](javascript:void(0)) [So I want to update all of the restaurant information and all of that is now inside](javascript:void(0)) [of a div with an id of restaurantList.](javascript:void(0)) [And with just this code in place, let me press F5 to run this application with the debugger.](javascript:void(0)) [That will prompt me to change by Web.config back.](javascript:void(0)) [If you remember, we went into Web.config and set debug equals false.](javascript:void(0)) [This is going to set that back to true which makes things a little bit easier](javascript:void(0)) [to debug if something goes wrong.](javascript:void(0)) [I'll just click Okay and let it make that change.](javascript:void(0)) [And now, in the application, let's search for 23.](javascript:void(0)) [And you can see this kind of works.](javascript:void(0)) [Asynchronous form made a request back to the server, got back a response,](javascript:void(0)) [and then we're after that into the page where the restaurant list used to be.](javascript:void(0)) [So we now we have a page inside of a page that has a restaurant list](javascript:void(0)) [and I could continue doing that infinitely.](javascript:void(0)) [So the problem in here is that the client side stuff all worked, we need this server](javascript:void(0)) [to respond differently when we do a search.](javascript:void(0)) [We need it to return just updated restaurants that should be displayed on the page.](javascript:void(0)) [And that means this restaurantList needs to display in two ways.](javascript:void(0)) [First of all, it needs to display on the homepage](javascript:void(0)) [when the user first comes to the homepage.](javascript:void(0)) [We need the restaurant list there.](javascript:void(0)) [And then, when we do a search, we need the restaurant list](javascript:void(0)) [to display and just the restaurant list.](javascript:void(0)) [And since we're going to need to this in two places,](javascript:void(0)) [that means we'll be using a partial view.](javascript:void(0)) [Let me cut the restaurantList out of the Index view and come into the views](javascript:void(0)) [for the HomeController and add a view called \_Restaurants.](javascript:void(0)) [This will be a Razor view.](javascript:void(0)) [It's going to be a partial view.](javascript:void(0)) [And I'll just click Add and paste in our HTML.](javascript:void(0)) [We can also make this strongly typed.](javascript:void(0)) [It's going to have the same model is the Index view, so I'll just copy the model directive](javascript:void(0)) [out of the Index view, paste it into \_Restaurants, that's our partial view.](javascript:void(0)) [That's part of what we have to do.](javascript:void(0)) [Now, we can render this when someone renders the Index view, so that we get a full page](javascript:void(0)) [with the restaurants and the header and the login and everything.](javascript:void(0)) [So we'll render Restaurants and we'll send the model along to be rendered.](javascript:void(0)) [And now, it's really going to be up to the controller to decide how to render a response](javascript:void(0)) [to the indexed action because this is all happening through the index action,](javascript:void(0)) [the homepage, the search results.](javascript:void(0)) [Let's open up the HomeController and I'll show you how easy it is](javascript:void(0)) [to make this decision inside of the HomeController.](javascript:void(0)) [What I'll do is stop debugging, shift F5.](javascript:void(0)) [And then paste in a little bit of code that can determine if a request arriving](javascript:void(0)) [to the server is in an asynchronous request.](javascript:void(0)) [It does this behind the scenes by looking into the HTTP headers.](javascript:void(0)) [There'll be a special flag there that you can find](javascript:void(0)) [out easily just by asking Request.IsAjaxRequest.](javascript:void(0)) [So if it is an asynchronous request, we'll return just that partial view](javascript:void(0)) [with the restaurant information, otherwise, we'll return the full view.](javascript:void(0)) [Use the same model object for both views.](javascript:void(0)) [But now, if I run the application again, we should have the result we're looking for.](javascript:void(0)) [I'll scroll down the page just a little bit and search for 24.](javascript:void(0)) [And you can see that bottom portion of the screen just updates instantly.](javascript:void(0)) [We don't lose our scroll position.](javascript:void(0)) [In fact, it's sometimes hard to tell that the screen changed.](javascript:void(0)) [We might need to add a little animation or fade-in, fade-out effect,](javascript:void(0)) [just to draw the user's attention to the fact that the screen has in fact updated.](javascript:void(0))

* + [An Async Search](javascript:void(0))

[We've just seen how Ajax.BeginForm works.](javascript:void(0)) [Although, we haven't really seen how it works behind the scenes.](javascript:void(0)) [I think, it's interesting to take a look at what is happening there](javascript:void(0)) [because we'll see some interesting techniques we can apply to our own JavaScript code.](javascript:void(0)) [There are basically three features MVC provides out of the box](javascript:void(0)) [that we will consider Ajax related.](javascript:void(0)) [There's Ajax.BeginForm, which we just used, and there's also an Ajax.ActionLink.](javascript:void(0)) [Just like Ajax.BeginForm, Ajax.ActionLink creates a link to make an asynchronous request](javascript:void(0)) [and update the screen instead of navigating the browser to a new page.](javascript:void(0)) [The third MVC feature is Client Side Validation.](javascript:void(0)) [All of these features use an approach to JavaScript programming](javascript:void(0)) [that is known as Unobtrusive JavaScript.](javascript:void(0)) [Unobtrusive JavaScript means, we don't have JavaScript littered throughout the view](javascript:void(0)) [on click events.](javascript:void(0)) [That's obtrusive to other developers who try to read the code.](javascript:void(0)) [And it's also obtrusive to users](javascript:void(0)) [because it usually means the content is only available if they have JavaScript enabled.](javascript:void(0)) [If I were to remove the scripts from this page or turn off JavaScript in IE,](javascript:void(0)) [the homepage would continue to work and the search would continue to work,](javascript:void(0)) [it just wouldn't be nice and asynchronous.](javascript:void(0)) [But it would continue to work and this is](javascript:void(0)) [because of the way the scripts interact with the page.](javascript:void(0)) [If we look at the source code, we'll get an idea of how BeginForm works.](javascript:void(0)) [The AjaxBeginForm helper admits data-attributes into the form tag.](javascript:void(0)) [Data dash attributes are part of the HTML5 specification and you are allowed to invent](javascript:void(0)) [as many different data dash attributes as you want.](javascript:void(0)) [They're basically private data for the application](javascript:void(0)) [to consume, the browser ignores them.](javascript:void(0)) [Then in scripts that you load on the page, there will be some JavaScript that will go](javascript:void(0)) [and interpret those data dash attributes and start attaching events](javascript:void(0)) [or doing whatever is necessary to add AJAX features,](javascript:void(0)) [whatever the data dash AJAX instructions tell them to do.](javascript:void(0)) [Client side validation works the same way.](javascript:void(0)) [Let's close this view and go out to a Review Edit page and look at the source code to this.](javascript:void(0)) [And here, you can see data dash attributes that specify all the validation rules for rating.](javascript:void(0)) [They're on the input for the rating value.](javascript:void(0)) [They include things like the minimum value, the maximum value, and the error message](javascript:void(0)) [to show it something is out of range.](javascript:void(0)) [So there's JavaScript code coming through, finding these data dash attributes,](javascript:void(0)) [interpreting the values inside and then adding behavior to the page.](javascript:void(0)) [It also works if JavaScript disabled, we won't have client validation, but nothing will break.](javascript:void(0)) [The data dash attributes will go unused, but the form will still post](javascript:void(0)) [and we'll have server side validation.](javascript:void(0)) [Let me show you how easy it would be to come in to the Index controller.](javascript:void(0)) [And instead of using Ajax.BeginForm, we'll go back to our simpler form](javascript:void(0)) [and implement this using just our own JavaScript code with some jQuery.](javascript:void(0)) [I will add some additional information to this opening form tag.](javascript:void(0)) [So it's still method equals get.](javascript:void(0)) [Now, I'm going to explicitly specify an action to make sure this form has an action attribute.](javascript:void(0)) [I'm using the URL helper to generate a URL to a controller action.](javascript:void(0)) [In this case, the HomeController index action, and then two data dash attributes.](javascript:void(0)) [One to identify that this form should be AJAX, (inaudible) to behave asynchronously,](javascript:void(0)) [and one to ID the DOM element that needs to be updated](javascript:void(0)) [when data comes back from the form submission.](javascript:void(0)) [Very similar to what we had in the AjaxOptions that we have to pass into begin form helper.](javascript:void(0)) [And I'm using a bit of a prefix in here, otf, just so my data dash attributes don't conflict](javascript:void(0)) [with any data dash attributes defined by the MVC framework.](javascript:void(0)) [And now, we will need a script to interpret these data dash attributes.](javascript:void(0)) [Let me come in to the Scripts folder and add a new item.](javascript:void(0)) [We'll search the installed templates for script,](javascript:void(0)) [find JavaScript File, and let's just call it otf.js.](javascript:void(0)) [This will be a JavaScript file that we use throughout the application,](javascript:void(0)) [so not just on the homepage.](javascript:void(0)) [And I can have these asynchronous forms anywhere.](javascript:void(0)) [For this to work everywhere, we will have to add it to our bundle](javascript:void(0)) [that is rendered on the Layout page.](javascript:void(0)) [So I'll just add this new script down here at the bottom.](javascript:void(0)) [It's going to depend on jQuery, so it has to come after jQuery.](javascript:void(0)) [But right there should be fine.](javascript:void(0)) [It will now be everywhere in the application.](javascript:void(0)) [And now, let's add some code to the script.](javascript:void(0)) [The first thing we'll do is use jQuery to hook up to the DOM ready event.](javascript:void(0)) [If you haven't used jQuery before, this is when you just invoke the jQuery function](javascript:void(0)) [which is the dollar sign, and then pass in a function that tells jQuery that you want](javascript:void(0)) [to execute some code when the DOM is ready.](javascript:void(0)) [That's the point when all the HTML has been received, has been parsed by the browser,](javascript:void(0)) [it's been put into memory in the document object model or the DOM.](javascript:void(0)) [So all the elements are in memory, they're already to be scripted.](javascript:void(0)) [Then we can use a jQuery selector to go out and find all the forms](javascript:void(0)) [that have this data attribute present and set to the value true.](javascript:void(0)) [Another great feature of jQuery is that it lets use css selector syntax](javascript:void(0)) [like this to select elements on a page.](javascript:void(0)) [This would select one form element or zero form elements or multiple form elements just as--](javascript:void(0)) [however, many have the data-otf.ajax attributes set the true inside them.](javascript:void(0)) [And then the third feature of jQuery is that it's very easy to wire up events.](javascript:void(0)) [So once I have selected zero or more form elements, I want to wire up to submit events,](javascript:void(0)) [so that when the user clicks a Save button or a Submit button, instead of that posting back](javascript:void(0)) [to the server, instead of that going back to the server, it will come into my JavaScript code.](javascript:void(0)) [In this case, call a function ajaxFormSubmit.](javascript:void(0)) [We haven't written that function yet, but it's just a normal function](javascript:void(0)) [that you can write in JavaScript.](javascript:void(0)) [Inside of this function, it's going to be our responsibilities to handle this form submission.](javascript:void(0)) [So we'll have to collect all the parameters, send them off to the server, get the result back](javascript:void(0)) [from the server, and then graph it into the page somewhere.](javascript:void(0)) [So the first thing I'll do is grab a reference to the form that is being submitted.](javascript:void(0)) [You can do that because it will be the this reference inside of the event handler.](javascript:void(0)) [And then, I'm just going to wrap it inside of jQuery](javascript:void(0)) [so that I can use jQuery functions on that element.](javascript:void(0)) [I want to use those functions to build an options object](javascript:void(0)) [that will contain the URL that we're going to go to.](javascript:void(0)) [I can get that by looking at the action attribute that is on that form.](javascript:void(0)) [The type of request to make, that would be a get or a post and we can get](javascript:void(0)) [that from the method attribute on the form, and finally the data to send along to the server.](javascript:void(0)) [Whatever inputs are in that form, we need to collect them all up](javascript:void(0)) [and to name value pairs and post them.](javascript:void(0)) [In the case of a search, that will just contained the search term,](javascript:void(0)) [but there could be additional data in there too.](javascript:void(0)) [And then once we have the options together, it's time to make the asynchronous call.](javascript:void(0)) [There's a number of different ways](javascript:void(0)) [to make asynchronous calls back to the server, the jQuery.](javascript:void(0)) [One is with $.ajax.](javascript:void(0)) [That's the one that gives you the most flexibility in the most options.](javascript:void(0)) [Here, I can just pass the options object in that will tell jQuery where to call the URL,](javascript:void(0)) [and also the data to pass along, and whether to do or get or a post.](javascript:void(0)) [And then when it is done, this is a call back function](javascript:void(0)) [when that request is complete and successful.](javascript:void(0)) [This function will be invoked and the response from the server will be in this data object.](javascript:void(0)) [What I need to do inside of this call back is go out and find the target](javascript:void(0)) [that is what is the DOM element on the page that I wanted to update with this data,](javascript:void(0)) [so we will go out and find that by digging the identifier out of the data dash attribute.](javascript:void(0)) [And then using the jQuery HTML method or rather what we could do is use the replace method,](javascript:void(0)) [we could say replace that target with this hunk of HTML that we got back from the server.](javascript:void(0)) [That will update the page.](javascript:void(0)) [And then, one last touch inside of ajaxFormSubmit, we need to prevent the browser](javascript:void(0)) [from doing its default action which is navigating away and going to the server](javascript:void(0)) [by itself and redrawing the page.](javascript:void(0)) [I can do that just by returning false down here.](javascript:void(0)) [And with all these code in place, I should be able to do a build and run the application.](javascript:void(0)) [Now, we'll be performing the same action as Ajax.BeginForm,](javascript:void(0)) [but we'll be doing it with our code.](javascript:void(0)) [So it's a little more open to customization and flexibility.](javascript:void(0)) [I should be able to search and I get a JavaScript error which happens occasionally.](javascript:void(0)) [I picked the wrong method to update screen with.](javascript:void(0)) [Let's try replaceWith to replace this element with what is inside of this data.](javascript:void(0)) [Save everything again, let's refresh the page.](javascript:void(0)) [And scroll down a little bit and try the search again.](javascript:void(0)) [And now, we're working just fine.](javascript:void(0)) [And now, we have some code to build on that we can customize](javascript:void(0)) [for our application using data dash attributes and we could this code,](javascript:void(0)) [make a loading message visible or log errors into all sorts](javascript:void(0)) [of interesting things that the application needs.](javascript:void(0)) [( Pause )](javascript:void(0))

* + [Autocompletion](javascript:void(0))

[Now that we have this nice asynchronous search feature,](javascript:void(0)) [it would be nice if we could also help our users find a restaurant they are looking](javascript:void(0)) [for by providing some suggestions when they start typing in the search box.](javascript:void(0)) [In other words, if I type the letter S, I'd like to see a list of possible restaurants](javascript:void(0)) [that start with the letter S as I'm typing.](javascript:void(0)) [Fortunately, we have jQuery UI in the project](javascript:void(0)) [and jQuery UI includes an autocomplete widget that will help us do this.](javascript:void(0)) [The way autocomplete works is that when the user starts typing, we can configure autocomplete](javascript:void(0)) [to make a request to the server and find possible matches.](javascript:void(0)) [Of course, we'll be calling a controller action](javascript:void(0)) [and the controller action will need to return JSON.](javascript:void(0)) [That's one of the formats that autocomplete can work with.](javascript:void(0)) [And then, once it has the list of possible restaurants,](javascript:void(0)) [it will take care of the UI part and put that list on the screen.](javascript:void(0)) [Since we're going to need some JSON data, let's start with a controller.](javascript:void(0)) [And since we are on the Index view of the HomeController, I'll put this action](javascript:void(0)) [that autocomplete will call in the HomeController.](javascript:void(0)) [This action is called autocomplete, but you could call it quick search or restaurant search](javascript:void(0)) [or anything that you want because ultimately, we'll tell autocomplete what URL to call](javascript:void(0)) [and we'll have control over what action it we call.](javascript:void(0)) [But I do expect this action to receive a parameter called term.](javascript:void(0)) [How do I know that, because if you look on jQueryUI.com,](javascript:void(0)) [they have lots of wonderful examples and demonstrations of how these widgets work.](javascript:void(0)) [And the documentation for autocomplete says when the user is typing, it will send a request](javascript:void(0)) [to the server and include a parameter in the request called term](javascript:void(0)) [and that will represent what the user has typed so far.](javascript:void(0)) [So I know if it sends something named term, I can get that as an action parameter.](javascript:void(0)) [I just need to take that term and query the database with it.](javascript:void(0)) [We'll say please give us the restaurants where the name of the restaurant starts](javascript:void(0)) [with whatever term that autocomplete has given me.](javascript:void(0)) [We will limit the result set to the first ten restaurants that we find](javascript:void(0)) [and then we'll do a projection with the select operator.](javascript:void(0)) [We'll turn every restaurant into an object that has a label property](javascript:void(0)) [and that label property will be equal to the restaurant name.](javascript:void(0)) [Why am I creating something with a label property?](javascript:void(0)) [Because, again, the jQuery UI documentation for autocomplete says the JSON you return,](javascript:void(0)) [the object should have a label property, or a value property, or a label and a value](javascript:void(0)) [that will use any combination of that.](javascript:void(0)) [Just giving it a label will be enough to have it put something on the screen](javascript:void(0)) [and when the user selects it, it'll put it into the input box for us.](javascript:void(0)) [All I need to do is to put that into JSON format which as we saw early](javascript:void(0)) [in this course is very easy to do, just invoke the JSON method,](javascript:void(0)) [it will serialize my model into JSON format.](javascript:void(0)) [We're going to allow this to happen during a get request.](javascript:void(0)) [And I should be able to test up this controller action from the browser.](javascript:void(0)) [If I just do a build, I can come and go to that controller action from the browser](javascript:void(0)) [and see the JSON that it's going to produce.](javascript:void(0)) [We'll go to home, autocomplete and let's pass a search term of 1 along.](javascript:void(0)) [And I can see the browser is telling me that some JSON came back the server.](javascript:void(0)) [We'll open that up.](javascript:void(0)) [And there I can see my JSON array.](javascript:void(0)) [It looks like all the objects have label properties.](javascript:void(0)) [Looks like all the objects have the label property set correctly.](javascript:void(0)) [So I think this is going to work with autocomplete.](javascript:void(0)) [Now, we just need to implement the client side of this.](javascript:void(0)) [The way autocomplete works from the client site is you have to wire it up against the input](javascript:void(0)) [that needs the autocomplete behavior.](javascript:void(0)) [And the way I'll identify the inputs that need an autocomplete behavior is with, guess what,](javascript:void(0)) [a data dash attribute, data-otf-autocomplete.](javascript:void(0)) [And the value for this attribute will be the URL that autocomplete needs to talk to.](javascript:void(0)) [So again, I'm using the URL helper to generate a URL that will point](javascript:void(0)) [to the HomeController's autocomplete action with this](javascript:void(0)) [and that value will be embedded in the data dash attribute.](javascript:void(0)) [That's the only change I need to make in the view, everything else will be JavaScript.](javascript:void(0)) [What I need to do using jQuery is go out and find those inputs](javascript:void(0)) [that have that data dash attribute.](javascript:void(0)) [So again, I'll use a selector here.](javascript:void(0)) [Find me all the inputs that have data-otf-autocomplete.](javascript:void(0)) [And for each one, call this function createAutocomplete.](javascript:void(0)) [In this case, we're not really wiring up an event like we did with the form](javascript:void(0)) [where we wired up the submitted end.](javascript:void(0)) [In this case, we need to go up to each object and give it the autocomplete widget.](javascript:void(0)) [So createAutocomplete will be a function that we need to write.](javascript:void(0)) [And when jQuery invokes this function, it will pass along the input as the this parameter.](javascript:void(0)) [So for each input that it finds with that data dash attribute, it will invoke this function,](javascript:void(0)) [pass along that single input as the this reference.](javascript:void(0)) [This code is just wrapping that input in jQuery using the dollar sign function](javascript:void(0)) [so I can use the jQuery API against it.](javascript:void(0)) [And very similar to last time, I'll also create an options object.](javascript:void(0)) [Now, there's all sorts of options that you can specify for the autocomplete.](javascript:void(0)) [You can tell it the minimum number of characters that the user has](javascript:void(0)) [to type before it will send a request to the server.](javascript:void(0)) [You can tell it how many milliseconds it has to wait](javascript:void(0)) [until after the user has stopped typing before it sends off that request.](javascript:void(0)) [But the one option that you must specify is the source option](javascript:void(0)) [that tells the autocomplete widget where to get the data.](javascript:void(0)) [So in this case, we'll just take the URL that was embedded in that data dash attribute.](javascript:void(0)) [We'll pull it out using the attribute method of jQuery and put it](javascript:void(0)) [in the source property on the options object.](javascript:void(0)) [And then, all I need to do to wire up autocomplete is to walk up to the input,](javascript:void(0)) [invoke the autocomplete method and pass in that options object.](javascript:void(0)) [This is how a lot of jQuery plug-ins work.](javascript:void(0)) [They extend jQuery by adding additional methods](javascript:void(0)) [on the jQuery objects so you can just invoke them.](javascript:void(0)) [jQuery UI adds methods to jQuery like autocomplete and dialog and button and tabs](javascript:void(0)) [and accordion, all sorts of user interface type things.](javascript:void(0)) [And at this point, I think I have everything correct where if I run the application,](javascript:void(0)) [I should be able to see autocomplete work.](javascript:void(0)) [So let's type in the letter S and there we can see the two restaurants](javascript:void(0)) [that start with S. So this looks promising.](javascript:void(0)) [Let's type in the number 11.](javascript:void(0)) [There, I can see all sorts of restaurants that start with 11.](javascript:void(0)) [And after I select one from the list, if I click Search, then it appears.](javascript:void(0)) [So what I need to do is first type something, select it, it'll be placed into the input box](javascript:void(0)) [for me by jQuery autocomplete and then I need to do a Search.](javascript:void(0)) [So a lot of people look at that and say can't make it so that one someone selects one](javascript:void(0)) [of these items, it just automatically submits that search](javascript:void(0)) [for them so that the results are updated.](javascript:void(0)) [And the answer is of course we can.](javascript:void(0)) [Over here in the options that we passed to autocomplete, when it creates itself,](javascript:void(0)) [we can pass in an additional option which is the function to invoke when the select even occurs.](javascript:void(0)) [So when the user selects something, please call a function called submit autocomplete form.](javascript:void(0)) [That is another function that we'll have to write.](javascript:void(0)) [And once again, the this reference will be set up to point](javascript:void(0)) [to the DOM element that we're interacting with.](javascript:void(0)) [In this case, it's the input again.](javascript:void(0)) [And I'll wrap that DOM element with jQuery so that we can do some interesting things with it.](javascript:void(0)) [For instance, one of the things that we'll have to do is set its value](javascript:void(0)) [because even though jQuery autocomplete will automatically populate the input with the item](javascript:void(0)) [that the user has selected, sometimes](javascript:void(0)) [that doesn't happen before this select event is raised.](javascript:void(0)) [So you can get into submit autocomplete form and still have an old input value there,](javascript:void(0)) [not the new one that the user selected.](javascript:void(0)) [We'll circumvent that problem just by setting the input value ourselves.](javascript:void(0)) [So this is the jQuery API to set the value of an input.](javascript:void(0)) [We're going to set it to you ui.item.label.](javascript:void(0)) [UI is a parameter that autocomplete passes in.](javascript:void(0)) [This is something also that you'll find in the documentation.](javascript:void(0)) [That UI object will have all sorts o interesting information about the state of autocomplete](javascript:void(0)) [and the parameters of autocomplete including the label of the item that the user clicked on.](javascript:void(0)) [So once the input is set, we just need to submit the form.](javascript:void(0)) [But we have to find the form first.](javascript:void(0)) [Fortunately, that's pretty easy with jQuery too.](javascript:void(0)) [What I can do is say, dear input, go and look through the parents above you,](javascript:void(0)) [all the DOM elements above you, and find a form.](javascript:void(0)) [In fact, find the first form.](javascript:void(0)) [That's just in case you're inside of a form that's inside of a form.](javascript:void(0)) [That would seem a little bit weird, but we'll just take the first form that you find](javascript:void(0)) [as you look up through your ancestors.](javascript:void(0)) [And once I find the form, there's a jQuery API where I can tell the form to submit itself.](javascript:void(0)) [That will raise the submit event and that will ultimately come back](javascript:void(0)) [into our AJAX form submit function that we wrote earlier.](javascript:void(0)) [And now, if I save this script, the only change I made was in script, so I should just be able](javascript:void(0)) [to save the file, refresh the browser and see this new behavior.](javascript:void(0)) [Let's try looking for 11 and selecting something.](javascript:void(0)) [And there, I can see the form was automatically submitted for me.](javascript:void(0)) [Try it again with the letter S. That seems to work too, and 11 too, all this seem to work.](javascript:void(0)) [And the one thing that still bothers me is that it's hard to tell](javascript:void(0)) [when the bottom half of the screen is updated.](javascript:void(0)) [This is automatically updating for me now.](javascript:void(0)) [I almost don't notice it.](javascript:void(0)) [It would be nice if I could draw the user's attention](javascript:void(0)) [to the bottom half of the screen somehow.](javascript:void(0)) [This is another place where jQuery UI can help.](javascript:void(0)) [It includes a number of effects that you can apply to DOM elements.](javascript:void(0)) [Effects that make things pulsate or bounce or grow or shrink.](javascript:void(0)) [They're very easy to use.](javascript:void(0)) [Let me show you an example.](javascript:void(0)) [Up here, when we take some HTML back from the server and we replace what's in the page](javascript:void(0)) [for the restaurant list with that new data, we can make it highlight itself.](javascript:void(0)) [I'll place some code in here that will take the HTML that's coming from the server and wrap it](javascript:void(0)) [in jQuery so we can manipulate it with the jQuery UI.](javascript:void(0)) [And then, after we have replaced the restaurant list with this new HTML,](javascript:void(0)) [I'll call the jQuery UI Effect method and tell it to run the highlight effect.](javascript:void(0)) [This is where you can pass the parameter to tell it if it should explode or pulsate or highlight.](javascript:void(0)) [We'll choose highlight.](javascript:void(0)) [Just, again, save the script.](javascript:void(0)) [Let's refresh the page to get that new script in the browser.](javascript:void(0)) [And then I'll search for 11.](javascript:void(0)) [And there you can see, it's just a very brief flash.](javascript:void(0)) [Hopefully, you can see it in the video.](javascript:void(0)) [But it should be just enough to draw my attention to that area and reassure me that,](javascript:void(0)) [yes, the results here actually have changed for me automatically.](javascript:void(0)) [( Pause )](javascript:void(0))

* + [Paging Results](javascript:void(0))

[Finally, there's one more feature I'd like to add.](javascript:void(0)) [I'd like the ability to pageList lists of restaurants.](javascript:void(0)) [The last search we did was to search for restaurants that start with a 20.](javascript:void(0)) [And I can see 20 and 201, but it only goes to 208](javascript:void(0)) [and there is a restaurant 209 but I cannot see it here.](javascript:void(0)) [There's no indication that there's more results.](javascript:void(0)) [That's because we only take the first ten records from the database that we get.](javascript:void(0)) [So let's have paging.](javascript:void(0)) [To do the paging, I'm going to rely on a third party component and one of the best ways](javascript:void(0)) [to install a third party component is by right clicking on the References folder](javascript:void(0)) [and selecting Manage NuGet Packages.](javascript:void(0)) [NuGet is a package manager for .NET.](javascript:void(0)) [You can download software packages which might be JavaScript libraries or .NET assemblies](javascript:void(0)) [or something that has both of those and more.](javascript:void(0)) [Now, you don't have to find the project and download the zip](javascript:void(0)) [and extract the files and add them to your project.](javascript:void(0)) [All you need to do is come to that References and open up the NuGet Package manager.](javascript:void(0)) [From here, I can see a list of all the packages that are installed in my application.](javascript:void(0)) [That includes things like the EntityFramework and jQuery UI.](javascript:void(0)) [I can also go online and I want to search for paged list.](javascript:void(0)) [I'll search from the text box up here in the top right of the screen](javascript:void(0)) [and that will find me PagedList.Mvc and PagedList.](javascript:void(0)) [Two great projects by a fellow named Troy Goode.](javascript:void(0)) [I'm going to tell NuGet to install PagedList.Mvc.](javascript:void(0)) [NuGet understands package dependencies too and it can see that this package depends](javascript:void(0)) [on PagedList, so it downloads them both and adds them to my project.](javascript:void(0)) [I can click Close and come in to the assembly references and see PagedList and PagedList.Mvc.](javascript:void(0)) [Now, I can start making changes to the application.](javascript:void(0)) [We'll start with the HomeController.](javascript:void(0)) [The index action of the HomeController will no longer be able to just return just ten records.](javascript:void(0)) [Instead, we need to return the proper page of records.](javascript:void(0)) [And what the PagedList package will give me is, first of all, an extension method that I can use](javascript:void(0)) [in the query to get the proper page of results and also some HTML helpers that I can use](javascript:void(0)) [in a view to display a pager widget, something with more and next links and page numbers in it](javascript:void(0)) [that a user can click on to navigate around through a large result.](javascript:void(0)) [The first thing we need to use though is the Extension method in a query.](javascript:void(0)) [So let me add a using statement for PagedList.](javascript:void(0)) [And now, I can come down to the index action and instead of having that take operator in there,](javascript:void(0)) [we will end this query using a call ToPagedList.](javascript:void(0)) [ToPagedList, I need to pass it the page number that I want.](javascript:void(0)) [Where do I get that from?](javascript:void(0)) [We'll take it as a parameter, because when the user interacts with the pager that we're going](javascript:void(0)) [to produce, we'll make sure there's something in the request called page.](javascript:void(0)) [It might not always be there though, so we'll have a default of 1.](javascript:void(0)) [That way, if you come to this page and you don't specify what page number you want,](javascript:void(0)) [you get the first page of results.](javascript:void(0)) [And so, ToPagedList will get whatever page has been passed in.](javascript:void(0)) [And also, we need to specify the page size.](javascript:void(0)) [Let's just say ten restaurants per page is good.](javascript:void(0)) [And these are all the changes that we need to make to the controller.](javascript:void(0)) [Everything else is going to be in the UI.](javascript:void(0)) [The first change we'll make in the UI is to tell it about the new model type](javascript:void(0)) [because this is now a model that is of type IPagedList.](javascript:void(0)) [That's going to allow me to use the pager helpers to build a paging control.](javascript:void(0)) [Let's come in to the index action and instead of being a model that is IEnumerable](javascript:void(0)) [of RestaurantListViewModel, it's now going to be a model of IPagedLisr of Restaurant View Model.](javascript:void(0)) [Right now, I have a red squiggly here because razor doesn't understand what an IPagedLIst is.](javascript:void(0)) [I could add a using statement in here to tell it what namespace that is in,](javascript:void(0)) [or I can just add that as a default namespace for my entire application.](javascript:void(0)) [In fact, I could add OdeToFood.Models also as a default namespace.](javascript:void(0)) [This just cleans up the code and makes it a little easier to work with.](javascript:void(0)) [So the way that you do this is by opening up the Web.config file that is in the Views folder.](javascript:void(0)) [Not the Web.config file that is in the root of the application, but the Web.config file that's](javascript:void(0)) [in the Views folder because it is this web config that controls the razor configuration.](javascript:void(0)) [This is the list of default namespaces that razor uses.](javascript:void(0)) [I want to add three additional namespaces, PagedList,](javascript:void(0)) [PagedList.Mvc, and also my OdeToFood.Models.](javascript:void(0)) [Now, quite often, when you change this section of Web.config, Visual Studio](javascript:void(0)) [and that razor parser, they do not pickup the changes here until you restart Visual Studio.](javascript:void(0)) [And because this always happens to me, I'm just going to go ahead and I've maid these changes](javascript:void(0)) [and save everything and close Visual Studio.](javascript:void(0)) [We will reopen it right away.](javascript:void(0)) [And hopefully, all the red squigglies are gone from Index.cshtml.](javascript:void(0)) [And they are, so we're in good shape.](javascript:void(0)) [Now, from the Index view, we're also rendering a partial view with the same model.](javascript:void(0)) [I need to come into this partial view and also change the model directive there to let me open](javascript:void(0)) [up \_restaurants and we'll change this from in IEnumerable to also be an IPagedList.](javascript:void(0)) [And now, we can get rid of the OdeToFood.Models namespace qualifier too.](javascript:void(0)) [It is inside of here, in fact, inside of our restaurant list too and I will place a pager,](javascript:void(0)) [a call to Html.PagedListPager that's provided by PagedList.Mvc.](javascript:void(0)) [This will build the pager for me.](javascript:void(0)) [All I need to do is give it the model.](javascript:void(0)) [It will figure out how many total things are in there and also a lambda expression](javascript:void(0)) [that given a page will be able to return a URL to go to that page.](javascript:void(0)) [The PagedListPager will pass me the page.](javascript:void(0)) [I'll use Url.Action to generate the URL](javascript:void(0)) [to the HomeController Index action passing page along in the request.](javascript:void(0)) [And we know now, that will get into the action and into that to PagedList call.](javascript:void(0)) [There's all sorts of options that you can specify for the PagedList pager.](javascript:void(0)) [I'm going to select MinimalWithItemCountText.](javascript:void(0)) [That will just give me previous and next links and the total number of items that are](javascript:void(0)) [in my model that you can play around with here with the different options.](javascript:void(0)) [You can have individual page numbers.](javascript:void(0)) [You can do all sorts of crazy things with the PagedList pager](javascript:void(0)) [and make it work the way you want it to work.](javascript:void(0)) [But this should be enough that if I run the application, we should have a pager](javascript:void(0)) [on the screen now that tells me how many restaurants are in the database.](javascript:void(0)) [And indeed, there's the pager with the next link and it tells me I'm seeing ten restaurants](javascript:void(0)) [out a possible 1,003, that's the rate number.](javascript:void(0)) [So this seems to be working.](javascript:void(0)) [And I can click on Next to go to the next list of results.](javascript:void(0)) [So this is all very good, but there are two problems.](javascript:void(0)) [One is that the pager, it's not aesthetically pleasing at the moment.](javascript:void(0)) [That's an easy fix.](javascript:void(0)) [And also, you'll notice that we're redrawing the entire page when I click on the Next link.](javascript:void(0)) [We're going off and just redrawing the whole page.](javascript:void(0)) [I'd like that to be able to redraw just this portion of the page just like we do](javascript:void(0)) [when we do a search on this page.](javascript:void(0)) [So let's fix the easy one first and that is to improve the user interface here.](javascript:void(0)) [One of the files that the PagedList package will install](javascript:void(0)) [into my application is a PagedList.css file.](javascript:void(0)) [So a style sheet to make that pager look a little bit better.](javascript:void(0)) [I just need to make sure those styles are included with my styles, or put them in a bundle](javascript:void(0)) [to download with my style sheet bundles and that's the path I'll take.](javascript:void(0)) [This is the bundle that includes my site.css.](javascript:void(0)) [I'll also just include PagedList.css along with that.](javascript:void(0)) [And just doing a build and refreshing the application,](javascript:void(0)) [we should see a better implementation of the pager.](javascript:void(0)) [And indeed, that has helped quite a bit.](javascript:void(0)) [Now, I have the Previous and Next links, and those still work.](javascript:void(0)) [Now, let's try to make this asynchronous which will be a little bit more challenging.](javascript:void(0)) [What I need to do, just like how previously we intercepted this form submission,](javascript:void(0)) [when I click on this, we catch that event and send off the request ourselves](javascript:void(0)) [and redraw the screen, now, I have to intercept click events on these anchor tags that are here.](javascript:void(0)) [Intercept those requests, go off and get the page data myself and redraw the screen.](javascript:void(0)) [To intercept those requests, of course, we'll still be using jQuery.](javascript:void(0)) [So let's go back to our script file, but we'll do this a little bit differently.](javascript:void(0)) [What I'm going to do is not go out and look for those anchor tags and wear out Dot Click event.](javascript:void(0)) [That would be easy enough to do with jQuery.](javascript:void(0)) [I'm not going to do that because we destroy those anchor tags every time we repaint](javascript:void(0)) [the screen.](javascript:void(0)) [We go off and get a fresh batch of HTML and then we put that HTML in the page](javascript:void(0)) [and destroy the anchor tags that were there and replace them with new ones.](javascript:void(0)) [I'd have to rewire the event every time we redrew the screen like that.](javascript:void(0)) [Instead, what I'm going to do is walk up to something called main-content.](javascript:void(0)) [main-content is in the Layout view.](javascript:void(0)) [You can see it right here.](javascript:void(0)) [A section with a class of main-content.](javascript:void(0)) [I'm going to wire up a click even on this section because it contains all](javascript:void(0)) [of the HTML that our view renders.](javascript:void(0)) [And this section isn't destroyed.](javascript:void(0)) [It's always going to be on the page.](javascript:void(0)) [So this syntax is saying go out and find the element or elements with the class](javascript:void(0)) [of main-content and wire up the Click even.](javascript:void(0)) [This is using the On method with jQuery.](javascript:void(0)) [What this allows you to do is wire up an event on a DOM element that's higher up in the tree](javascript:void(0)) [and then specify how to filter those events.](javascript:void(0)) [This is saying, yes, I want click events, but I only want click events that originated](javascript:void(0)) [from an anchor tag that was inside of an element with a class of pagedList.](javascript:void(0)) [This way, I don't pickup just any anchor tag that might be on the page.](javascript:void(0)) [I'm picking up just the anchor tags that are inside of this div with a class of pagedList.](javascript:void(0)) [Now, we're intercepting those events.](javascript:void(0)) [We can call a method called getPage.](javascript:void(0)) [getPage is a function that we'll write.](javascript:void(0)) [And inside of getPage, this reference will point to the anchor tag that the user clicked on.](javascript:void(0)) [We'll get a reference to that and wrap it with jQuery](javascript:void(0)) [so that we can extract some values from it.](javascript:void(0)) [We will extract values like the Href attribute that's on that anchor tag](javascript:void(0)) [where it's pointing to, because that contains the information that we need to use](javascript:void(0)) [to get to the correct page number.](javascript:void(0)) [It will be something like /home/index query string page equals 5.](javascript:void(0)) [That's the Href value that we need.](javascript:void(0)) [We'll take that value.](javascript:void(0)) [Put it into a URL property on this options object.](javascript:void(0)) [Just like we did earlier with a form, we're going to build an options object](javascript:void(0)) [that tells jQuery how to contact the server.](javascript:void(0)) [This is the URL we want to go to.](javascript:void(0)) [This is the type of request that we want to make and get request.](javascript:void(0)) [And the rest of the code would look like this.](javascript:void(0)) [Given that options object that we just built, go out and do $.ajax](javascript:void(0)) [and make a request to get request to that URL.](javascript:void(0)) [And when it's done and you have the new data, call this function.](javascript:void(0)) [We'll go out and find the target, the thing that we need to update which will be restaurant list,](javascript:void(0)) [but I'm trying to rate this code in a generic fashion so it could be used anywhere](javascript:void(0)) [with any page list in the application.](javascript:void(0)) [All you need to specify after the user has clicked on one of the pager links,](javascript:void(0)) [what DOM element that you want to update.](javascript:void(0)) [I put that here in a data dash attribute inside of the div with a class of pagedList.](javascript:void(0)) [So we're just looking for this value to extract restaurant list.](javascript:void(0)) [And once we have extracted restaurant list, we'll go out and select that element with jQuery](javascript:void(0)) [and replace it with the fresh data from the server.](javascript:void(0)) [Let's cross our fingers and run the application and see if we can get this to work.](javascript:void(0)) [I'll scroll down a bit.](javascript:void(0)) [That's always a good test to see if you're doing a partial page refresh,](javascript:void(0)) [or refreshing the whole screen.](javascript:void(0)) [And it looks like this is not working.](javascript:void(0)) [I'm able to go to every page just by clicking Next,](javascript:void(0)) [and we're not refreshing the entire screen.](javascript:void(0)) [Does it work with the search?](javascript:void(0)) [Let's search for 20 again.](javascript:void(0)) [And there's my 20 results.](javascript:void(0)) [If I go to Next, that seems to be broken and that's because when we go off](javascript:void(0)) [to fetch the next page of result in here, we're not taking into account the search term](javascript:void(0)) [that user has entered on the page.](javascript:void(0)) [All we're passing along is the page number that we got out of the Href for the anchor tag.](javascript:void(0)) [But this is actually a very easy problem to solve.](javascript:void(0)) [What I can do inside of our AjaxOptions is say please add some additional data to this request.](javascript:void(0)) [Essentially, go out and get the form that's on this page and serialize it](javascript:void(0)) [and send those values along in the request too.](javascript:void(0)) [And now, if I save the script, and let's do a hard refresh in Internet Explore.](javascript:void(0)) [Now, let me search for 20 and page through the results,](javascript:void(0)) [and there's only two pages of results and that looks good.](javascript:void(0)) [Now, I may have a list of restaurants on the homepage that can search through](javascript:void(0)) [and page through, and it all happens asynchronously](javascript:void(0)) [to provide a nice slick user interface.](javascript:void(0)) [( Pause )](javascript:void(0))

* + [Summary](javascript:void(0))

[In this module, we added some AJAX features to our application using jQuery,](javascript:void(0)) [jQuery UI, and our own custom JavaScript.](javascript:void(0)) [We brought controller actions to return JSON and implemented paging and searching](javascript:void(0)) [with async behavior to provide an updated page without disrupting the user experience.](javascript:void(0)) [Finally, we saw quite a few data dash attributes in our HTML.](javascript:void(0)) [We saw how we could work with those attributes from our script](javascript:void(0)) [and also have the MVC framework uses those attributes to add AJAX functionality](javascript:void(0)) [through AJAX helpers and also to support client validation.](javascript:void(0))

* Security and ASP.NET MVC 4
  + [Introduction](javascript:void(0))

[Hi. This is Scott Allen.](javascript:void(0)) [In this module is we'll look at the security features of ASP.NET MVC.](javascript:void(0)) [In this module I want to show you how to work with users and roles and also how to use OpenID](javascript:void(0)) [and OAuth so your customers can sign in to your website using a third party](javascript:void(0)) [like Microsoft, Google, or Facebook.](javascript:void(0)) [I also want to show you some of the attacks that you'll face](javascript:void(0)) [if you deploy an application on a web server.](javascript:void(0)) [So we'll look at cross-site request forgeries](javascript:void(0)) [and how to make your application resist such an attack.](javascript:void(0)) [We'll start by looking at authentication and see how we can use Windows Integrated Authentication](javascript:void(0)) [for internet style apps and forms-based authentication for web applications.](javascript:void(0)) [( Pause )](javascript:void(0))

* + [Authentication](javascript:void(0))

[When you authenticate a user, you are verifying the identity of the user.](javascript:void(0)) [You might need to know a user's identity because you're building an application](javascript:void(0)) [that only specific users should access, like a payroll system.](javascript:void(0)) [You cannot let just anyone poke around in the salary information.](javascript:void(0)) [Or maybe you're building a website where users post their picture collections.](javascript:void(0)) [You might want to know the user's identity just so you can track their photographs](javascript:void(0)) [and only allow users to delete their own photographs and not the photographs of others.](javascript:void(0)) [So the first step would be identifying the user and making sure you know who they are.](javascript:void(0)) [There are three ways to do this with ASP.NET.](javascript:void(0)) [The first mechanism is called Forms Authentication.](javascript:void(0)) [With forms-based authentication, the website is responsible for providing a page](javascript:void(0)) [with an input form where the user can enter their user name and enter their password.](javascript:void(0)) [And when they click a log on button, something inside our application is responsible](javascript:void(0)) [for making sure they entered the right password.](javascript:void(0)) [We'll see how MVC provides a lot of the infrastructure support for Forms Authentication](javascript:void(0)) [so it's not as hard as it sounds.](javascript:void(0)) [Another way to authenticate a user is to use OpenID or OAuth.](javascript:void(0)) [These are open standards for authentication and authorization respectively and using OpenID](javascript:void(0)) [or OAuth means your users don't need to create or share a password with your site](javascript:void(0)) [and you don't have to store or validate a user's password.](javascript:void(0)) [Instead, you rely on a third party like Twitter or Microsoft](javascript:void(0)) [to authenticate the user and then tell you who they are.](javascript:void(0)) [We'll see this in a bit.](javascript:void(0)) [The third approach is Windows Authentication.](javascript:void(0)) [Windows Authentication is typically used for internet applications because it uses components](javascript:void(0)) [and services provided by the Windows Operating System.](javascript:void(0)) [Microsoft also refers to Windows Auth as "Integrated Auth", because once a user is logged](javascript:void(0)) [in to a domain, Windows can automatically authenticate them to My Application](javascript:void(0)) [if My Application uses Windows Auth.](javascript:void(0)) [Although Windows Authentication does work across a variety of browsers](javascript:void(0)) [and not just Internet Explorer, it's still best to use for an internet application](javascript:void(0)) [that runs inside your company's Firewall and where all of your users are logged](javascript:void(0)) [into a Windows domain and you have an active directory server somewhere.](javascript:void(0)) [Your users can have a single sign on experience, because once they logged on to their desktops,](javascript:void(0)) [they use that identity for your application as well as network shares](javascript:void(0)) [and network printers and other internet apps.](javascript:void(0)) [If you're building a public website then you probably want to avoid Windows Auth](javascript:void(0)) [because users might not be on the same domain.](javascript:void(0)) [In fact, they might not even be running on a device that supports Windows Authentication.](javascript:void(0)) [And that's where Forms Authentication comes in, because any web browser can let a user sign](javascript:void(0)) [into a site using Forms Authentication.](javascript:void(0)) [It's highly customizable because you build the log on page](javascript:void(0)) [and you control the log on experience.](javascript:void(0)) [You can customize how the form looks and how strong of a password that you require.](javascript:void(0)) [Forms Authentication relies on cookies by default.](javascript:void(0)) [So once a user is signed in to an application, the runtime will store a cookie](javascript:void(0)) [in the client's browser so they don't need to sign in again during the same browsing session.](javascript:void(0)) [One word of warning with Forms Authentication, is that SSL is required](javascript:void(0)) [to make Forms Authentication secure.](javascript:void(0)) [Unless I'm logging into a site using an HTTPS address instead of regular HTTP,](javascript:void(0)) [then I'm passing in my user name and password across the network in plain text and someone](javascript:void(0)) [who is watching network traffic can see my password and steal it.](javascript:void(0)) [OpenID and OAuth are also designed to work with internet applications.](javascript:void(0)) [When you need to validate a user's identity, you redirect their browser to a third party website,](javascript:void(0)) [like a website run by Google or Microsoft,](javascript:void(0)) [and the user proves their identity to this third party.](javascript:void(0)) [Then the third party will redirect the user back to you with a message](javascript:void(0)) [that proves they successfully authenticated themselves.](javascript:void(0)) [This all works securely, thanks to the magic of cryptography.](javascript:void(0)) [We'll see how to work with these three different approaches to authentication in this module.](javascript:void(0)) [And we'll get started by looking at Windows Authentication.](javascript:void(0)) [( Pause )](javascript:void(0))

* + [Windows Authentication](javascript:void(0))

[The OdeToFood application we've been working on was created](javascript:void(0)) [with the internet application project template.](javascript:void(0)) [You might remember that from the first module.](javascript:void(0)) [Internet applications use Forms-Based Authentication by default,](javascript:void(0)) [but I want to demonstrate Windows Authentication, and changing this application](javascript:void(0)) [over to use Windows Authentication is a bit messy.](javascript:void(0)) [It's possible but it's messy.](javascript:void(0)) [So instead of getting messy I'll launch a second instance of Visual Studio](javascript:void(0)) [and we'll create a new temporary project just to look at how Windows Authentication works.](javascript:void(0)) [So a new project using MVC 4, I don't really care about the name](javascript:void(0)) [because we won't be spending anytime with this.](javascript:void(0)) [So we don't need a unit test project either.](javascript:void(0)) [I just want to make sure instead of creating an internet application,](javascript:void(0)) [I create an intranet application and then I'll click okay and be back](javascript:void(0)) [in a second once Visual Studio has the setup.](javascript:void(0)) [Now the project is ready to go and Visual Studio will open up a readme.text file by default](javascript:void(0)) [when you create an intranet application.](javascript:void(0)) [It'll list a couple manual steps that you have to go through in order for this to work](javascript:void(0)) [because many web servers today ship with features turned off by default.](javascript:void(0)) [And one of the features that is off by default is the ability to do Windows Authentication.](javascript:void(0)) [To fix that, I need to come into to the MvcApplication and open up the property window.](javascript:void(0)) [And I do that by going to View, Properties Window.](javascript:void(0)) [In this case I cannot right-click on the Project and go to Properties.](javascript:void(0)) [That takes me to the wrong place.](javascript:void(0)) [I need to open up the Properties Window by clicking this or pressing F4.](javascript:void(0)) [And that allows me to set some properties on the development server.](javascript:void(0)) [Remember the development server is IIS Express.](javascript:void(0)) [That's a little application running down here in the system tray.](javascript:void(0)) [We've hardly looked at it at all.](javascript:void(0)) [It just sits there and runs in the background to process our request.](javascript:void(0)) [We need to come in for IIS Express and tell it to enable Windows Authentication.](javascript:void(0)) [And that's all the configuration we really need right now.](javascript:void(0)) [Everything in an intranet application requires authentication by default.](javascript:void(0)) [So it will not let a user in until it knows who they are.](javascript:void(0)) [And that means if I run the application,](javascript:void(0)) [the first thing Internet Explorer will do is pop up a log in box.](javascript:void(0)) [And you might wonder, well, what's so integrated about Windows Authentication](javascript:void(0)) [if Internet Explorer is always going to ask me to log in?](javascript:void(0)) [But this behavior is peculiar to Internet Explorer](javascript:void(0)) [because Internet Explorer will only automatically log you into a site if it thinks](javascript:void(0)) [that site is on your local network.](javascript:void(0)) [And by default, Internet Explorer doesn't consider local host to be on the local network.](javascript:void(0)) [If I want to change that experience I would have to go into the Tools menu in IE,](javascript:void(0)) [so Alt T will open up the menu, and then I can go to Internet Options.](javascript:void(0)) [Go to the Security tab.](javascript:void(0)) [Click on Local Intranet and then go to sites.](javascript:void(0)) [Go to Advanced, and finally I reach a dialog where I can tell Internet Explorer](javascript:void(0)) [that local host should be considered an intranet site.](javascript:void(0)) [And the reason this is important is because if we look at the custom settings](javascript:void(0)) [for a local internet, essentially what are all the settings that are applied in this soon,](javascript:void(0)) [and I explore all the way down, you'll see right here](javascript:void(0)) [that Internet Explorer will only automatically log](javascript:void(0)) [on for a website that's in the intranet zone.](javascript:void(0)) [And it doesn't matter if you've created an internet site or an internet site with MVC,](javascript:void(0)) [this is completely based on IE's heuristics](javascript:void(0)) [on determining what's an internet site and what's not an internet site.](javascript:void(0)) [But now, local host should be considered in the internet zone](javascript:void(0)) [and if I close all these dialog boxes, and close the browser and press Ctrl F5 to run this again,](javascript:void(0)) [Internet Explorer will automatically log me in using my current credentials for this machine](javascript:void(0)) [or my domain if I have a domain controller.](javascript:void(0)) [The machine is WIN8VPC, my user name is Scott and that can automatically display it](javascript:void(0)) [because I've been authenticated, even here on the Homepage.](javascript:void(0)) [And no log in box was required this time.](javascript:void(0)) [Where does this text come from?](javascript:void(0)) [That's in the layout view for this application.](javascript:void(0)) [If we go to Views, Shared, Layout, we'll find there's a section](javascript:void(0)) [in here using User.Identity.Name.](javascript:void(0)) [So the user property and User.Identity, they're both available for Windows Authentication](javascript:void(0)) [and also Forms-based Authentication.](javascript:void(0)) [You can do things like find out the user's name.](javascript:void(0)) [You can also use identity to find out if the user has been authenticated](javascript:void(0)) [or not and how they were authenticated.](javascript:void(0)) [And we'll see User.Identity again when we come back to Forms-Based Authentication.](javascript:void(0)) [But for now, this is how Windows Authentication works.](javascript:void(0)) [And the one other primary difference between this application and the OdeToFood application,](javascript:void(0)) [if I come into web.config, inside of here there will be a section called "Authentication",](javascript:void(0)) [and you can see the mode is equal to Windows.](javascript:void(0)) [Inside the OdeToFood application, we have that same section,](javascript:void(0)) [authentication section, but the mode is equal to forms.](javascript:void(0)) [And that's essentially what's telling ASP.NET](javascript:void(0)) [if it's using Windows or Forms-Based Authentication.](javascript:void(0)) [( Pause )](javascript:void(0))

* + [Forms Authentication](javascript:void(0))

[Now we can look at Forms Authentication in the application we've been building, OdeToFood.](javascript:void(0)) [But before we get into the code, I feel we should take a step back and look](javascript:void(0)) [at the big picture of how Forms Authentication works in general](javascript:void(0)) [and how ASP.NET implements Forms Authentication in MVC 4.](javascript:void(0)) [First, imagine the user tries to go to a members only area of the site,](javascript:void(0)) [an area that requires user authentication.](javascript:void(0)) [We'll see how to enforce this requirement.](javascript:void(0)) [But MVC will know when you require authentication.](javascript:void(0)) [And when it sees an anonymous user trying to get to such a place,](javascript:void(0)) [it will redirect the user's browser to a log in page.](javascript:void(0)) [You can configure where this log in page lives](javascript:void(0)) [and in an MVC application this will not be a page of course, it will be a controller action.](javascript:void(0)) [But to the user it's a page and it's a page where they enter the user name and password.](javascript:void(0)) [When ASP.NET does this redirect, it will also save the URL,](javascript:void(0)) [the user originally wanted to access.](javascript:void(0)) [So if they successfully authenticate,](javascript:void(0)) [ASP.NET will automatically redirect them back to where they wanted to go.](javascript:void(0)) [It does this by storing the original URL in the query string](javascript:void(0)) [with the name ReturnUrl and it will use that value.](javascript:void(0)) [So once a user logs in, they don't have to click around the site trying](javascript:void(0)) [to find that members only area again.](javascript:void(0)) [They'll be redirected right back to it.](javascript:void(0)) [If the user cannot get logged in, they'll never get to the members area.](javascript:void(0)) [But if they do give the user name and password, ASP.NET will redirect them back and they'll get](javascript:void(0)) [to see the content that you're protecting.](javascript:void(0)) [ASP.NET also gives the browser a cookie for the application.](javascript:void(0)) [A cookie is like a piece of data that the browser will hold on to.](javascript:void(0)) [And once a browser holds a cookie for a site, it will continue to send that cookie along](javascript:void(0)) [in every request that it makes to the site or at least as long as the browsing session lasts,](javascript:void(0)) [possibly longer if you and the user allow it.](javascript:void(0)) [Inside of that cookie is some encrypted information letting ASP.NET know](javascript:void(0)) [that the user has already authenticated themselves and this means,](javascript:void(0)) [as the user is browsing through that members only area or other sections of the application](javascript:void(0)) [that require authentication, ASP.NET will see that cookie,](javascript:void(0)) [know the user has been authenticated, it's not an anonymous user,](javascript:void(0)) [and ASP.NET will allow them to succeed with that request.](javascript:void(0)) [It won't force them to reauthenticate or sign in again.](javascript:void(0)) [What I'm describing about Forms Authentication isn't specific to ASP.NET,](javascript:void(0)) [many web frameworks take a similar approach to Forms Authentication.](javascript:void(0)) [They use log in pages, they use cookies, just the names of the pages and the names](javascript:void(0)) [of the cookies and query string values change.](javascript:void(0)) [Let's talk about how this works in ASP.NET MVC 4 specifically.](javascript:void(0)) [First, when you create a new application using the internet project template,](javascript:void(0)) [the project template will include everything you need to register a user, allow them to log in,](javascript:void(0)) [allow them to change their password, all the basics that you need for Forms Authentication.](javascript:void(0)) [All these basics are provided by an AccountController,](javascript:void(0)) [you'll find that in your Controllers folder as well as a set of views](javascript:void(0)) [in the Views Account folder and there're also some models.](javascript:void(0)) [You'll find them in the Models folder.](javascript:void(0)) [If you peek at the code inside the AccountController,](javascript:void(0)) [you'll see it's using a class named WebSecurity.](javascript:void(0)) [And WebSecurity is from a Microsoft library named WebMatrix.](javascript:void(0)) [Perhaps, you've heard of WebMatrix.](javascript:void(0)) [It's another tool from Microsoft, you can use to create websites](javascript:void(0)) [and ASP.NET MVC is borrowing some of that technology from WebMatrix](javascript:void(0)) [to implement Forms Authentication.](javascript:void(0)) [It's the WebSecurity class you can talk to from your code in the AccountController](javascript:void(0)) [to tell it to validate a user's password.](javascript:void(0)) [WebSecurity in turn talks to a component known as this SimpleMembershipProvider.](javascript:void(0)) [And although there are more layers that what I'm showing here.](javascript:void(0)) [Ultimately, you can think of the membership provider as doing the hard work](javascript:void(0)) [of storing information in the database.](javascript:void(0)) [Earlier I mentioned that Forms Authentication is customizable.](javascript:void(0)) [And you can certainly go into the AccountController](javascript:void(0)) [and the AccountController views and make as many changes as you'd](javascript:void(0)) [like to have the log in form look different.](javascript:void(0)) [You can also customize the application by taking control over the underlying SQL Storage.](javascript:void(0)) [You can customize what you want to store about a user.](javascript:void(0)) [For instance, you could customize the Register screen by asking the user](javascript:void(0)) [to name their favorite restaurant, and then store that in a table in SQLServer along](javascript:void(0)) [with the rest of the user information that you need.](javascript:void(0)) [I'll show you how all this works together and some steps you want](javascript:void(0)) [to make to customize the application.](javascript:void(0)) [( Pause )](javascript:void(0))

* + [Taking Control of Membership](javascript:void(0))

[Before we get in to registering and logging in and seeing how to require authentication](javascript:void(0)) [in the application, let me show you a few steps you can take to have more control](javascript:void(0)) [over what's happening behind the scenes, particularly at the database layer](javascript:void(0)) [and particularly if you want to customize the information that's being stored about a user.](javascript:void(0)) [The first step I would recommend is coming into the project and finding the Filters folder.](javascript:void(0)) [Inside of here you'll find the file, Initialize SimpleMembershipAttribute.](javascript:void(0)) [If we open this up, there's a lot of complex code in here](javascript:void(0)) [because when you start a new MVC 4 application with this project template that we used,](javascript:void(0)) [the internet project template, it's not 100 percent sure](javascript:void(0)) [that you want to use Forms Authentication.](javascript:void(0)) [So there's a lot of complex code here to make sure](javascript:void(0)) [that Forms Authentication is initialized in a lazy manner.](javascript:void(0)) [Perhaps you're not even going to use a database in this website.](javascript:void(0)) [And in that case, none of this code would have to execute.](javascript:void(0)) [But if you've already decided that you are going to use Forms Authentication,](javascript:void(0)) [there's no reason to have all of this code.](javascript:void(0)) [What you really want to get to is the essence inside of here](javascript:void(0)) [which is WebSecurity dot InitializeDatabaseConnection.](javascript:void(0)) [This is a call to initialize all the membership infrastructure and tell it,](javascript:void(0)) [a name of the connection string that you're going to use to connect to the database,](javascript:void(0)) [which is DefaultConnection, the same database we're using](javascript:void(0)) [to store a restaurant information is also specifies the name of the table](javascript:void(0)) [that will contain user information, its UserProfile,](javascript:void(0)) [a column that contains the primary key value for a user, so it can look a user](javascript:void(0)) [up given their ID value and also the column that contains the UserName.](javascript:void(0)) [You can have as many other columns in this table as you want when you store user profiles,](javascript:void(0)) [but you at least need to give it a primary key and the user name.](javascript:void(0)) [If we know we're using Forms Authentication, we might as well take this line of code and cut it](javascript:void(0)) [out of here, Ctrl X, and run it during application startup.](javascript:void(0)) [So I will come into Global.asax.cs](javascript:void(0)) [and inside the application start method I will paste that code inside of here.](javascript:void(0)) [We'll need to bring in a namespace, so control period and select that first entry,](javascript:void(0)) [this WebSecurity class is in the WebMatrix.WebDataNameSpace.](javascript:void(0)) [Now that namespace is added and we have legal C# code.](javascript:void(0)) [If you want to be consistent, you could wrap up this call into some static method on a class](javascript:void(0)) [so everything looks like something .Register.](javascript:void(0)) [But right now we'll keep things simple](javascript:void(0)) [and just leave initialize database connection right here in the application start.](javascript:void(0)) [Now that that is here, there's no use to have this filter.](javascript:void(0)) [So we'll come into this filter and delete it.](javascript:void(0)) [And yes, the next time you build you'll get compiler errors.](javascript:void(0)) [But we're going to fix this.](javascript:void(0)) [Before I fix this, there's one more step I want to take.](javascript:void(0)) [Inside of the Models folder, you will find an AccountModel.cs file.](javascript:void(0)) [Inside of here, there's a class that derives from DbContext, you recognize that now](javascript:void(0)) [as the entity framework DbContext.](javascript:void(0)) [This is giving the AccountController access to that user's table](javascript:void(0)) [through a DbContext derived class.](javascript:void(0)) [But we want to control this.](javascript:void(0)) [We want to have this as part of our DbContext.](javascript:void(0)) [So I'm just going to delete this DbContext.](javascript:void(0)) [We don't need it.](javascript:void(0)) [We do need something to represent a user profile.](javascript:void(0)) [So I might as well start with this class.](javascript:void(0)) [It has a UserID and a UserName.](javascript:void(0)) [I could add additional properties here.](javascript:void(0)) [For instance if wanted to know a user's favorite restaurant then I can include](javascript:void(0)) [that as another property in the user profile.](javascript:void(0)) [But just so I know that this is my code, I want to own this code,](javascript:void(0)) [it's not part of the generated code that came from this template really anymore.](javascript:void(0)) [I've customize it.](javascript:void(0)) [I'm going to cut it out of here and add it to its own class file.](javascript:void(0)) [So let's add a class called UserProfile and I will delete what comes out of here by default](javascript:void(0)) [and just paste in that UserProfile that we just had.](javascript:void(0)) [I'll need to include a couple of namespaces here, too.](javascript:void(0)) [So Ctrl period and bring in System.Component Model.DataAnnotations.Schema.](javascript:void(0)) [This attribute is one way to tell the entity framework that when you need](javascript:void(0)) [to store user profile information, when you need to store objects of this type,](javascript:void(0)) [put it in a table with this name, UserProfile.](javascript:void(0)) [We also need to bring in a namespace for the key attribute that's](javascript:void(0)) [in the System.Component Model.DataAnnotations.](javascript:void(0)) [We've used that before when we added some model validations.](javascript:void(0)) [This particular attribute is a way to tell the entity framework](javascript:void(0)) [that this is the primary key value for a user.](javascript:void(0)) [And by the way, it's also an identity column, meaning the database will manage the values](javascript:void(0)) [that get placed into here when you insert a record.](javascript:void(0)) [It'll automatically be generated by SQL Server.](javascript:void(0)) [And now that we have our own user profile class that can store our favorite restaurant,](javascript:void(0)) [I need to define a way to get to that in my own DbContext class.](javascript:void(0)) [So I will open this one up, OdeToFoodDb and let's add a property of type DbSet](javascript:void(0)) [of UserProfile and we could call this UserProfiles.](javascript:void(0)) [And now if you've been watching the previous modules, you know we've made a change](javascript:void(0)) [to what we're storing in the database.](javascript:void(0)) [We're going to need to do a schema migration.](javascript:void(0)) [But before we can do that we have to get the project](javascript:void(0)) [to build now that we've deleted some classes.](javascript:void(0)) [Let me do a quick build and that would give me a list to work from.](javascript:void(0)) [So first of all, if I double-click one of these errors,](javascript:void(0)) [this Initialize SimpleMembershipAttribute doesn't exist any longer.](javascript:void(0)) [That's okay.](javascript:void(0)) [We're initializing things explicitly during application start, so I can just delete that.](javascript:void(0)) [Then I'll double click another error.](javascript:void(0)) [This one is complaining about UsersContext, we deleted that.](javascript:void(0)) [That was what the AccountController was using to get to that UserProfile table.](javascript:void(0)) [Now I can get to it through OdeToFoodDb.](javascript:void(0)) [So let me just change this to use the var keyword and instead](javascript:void(0)) [of a new UsersContext we'll instantiate a new OdeToFoodDb.](javascript:void(0)) [Let me try another build, Shift Ctrl B. Now we've successfully built.](javascript:void(0)) [Let's quickly migrate the database so we can store our custom information.](javascript:void(0)) [I'll open up the Package Manager Console.](javascript:void(0)) [We have implicit migrations enabled, so I should just be able to say, update database](javascript:void(0)) [and I'll throw in the Verbose flag just so we can see exactly what it's going to do.](javascript:void(0)) [And it finished the migrations.](javascript:void(0)) [It's running the Seed method.](javascript:void(0)) [But you can see where it went out and created a table, UserProfile, that has UserID,](javascript:void(0)) [UserName, and FavoriteRestaurant.](javascript:void(0)) [We can probably tweak the annotations on UserProfile](javascript:void(0)) [so that User Name was an nvarchar max.](javascript:void(0)) [It would be much better for performance if that had a reasonable size](javascript:void(0)) [like nvarchar 255 or nvarchar 80.](javascript:void(0)) [It will be simple enough to do that just using a string length attribute on UserName,](javascript:void(0)) [and the same for FavoriteRestaurant.](javascript:void(0)) [But at this point, we're good.](javascript:void(0)) [We have more control over what's going to be happening in the database.](javascript:void(0)) [And now, we can turn our attention to looking at how to register,](javascript:void(0)) [how to log in, how to require authentication.](javascript:void(0)) [( Pause )](javascript:void(0))

* + [Forms Authentication In Action](javascript:void(0))

[Let's see how Forms Authentication works inside of the application now.](javascript:void(0)) [As a user, I can come in and click the Register link.](javascript:void(0)) [And this will give me a page where I can fill out the bare amount of information](javascript:void(0)) [to create an account, just a User name, a Password, and then confirm my password.](javascript:void(0)) [I can click Register and I have an account and I'm automatically logged into the site.](javascript:void(0)) [My name appears at the top of the page and that's done using User.Identity.Name,](javascript:void(0)) [just like we saw with Windows Authentication in the Layout view.](javascript:void(0)) [At this point, I could click on my name to manage my account and change my password.](javascript:void(0)) [I could also log off of the site and then come in and log in again.](javascript:void(0)) [And all of this functionality is provided out of the box](javascript:void(0)) [when we create a new internet application.](javascript:void(0)) [The register page for example is given to us by adding a view](javascript:void(0)) [to the account folder called Register.cshtml.](javascript:void(0)) [And it's inside of here where you can see textboxes to input the user name](javascript:void(0)) [and password boxes to input the password and confirm the password.](javascript:void(0)) [This is one of the places you'd have to come in and modify if you wanted to force a user](javascript:void(0)) [to provide you additional information when they register like the name of a favorite restaurant.](javascript:void(0)) [And this view is rendered from an AccountController which we were also given.](javascript:void(0)) [Of course that's in the Controllers folder.](javascript:void(0)) [And it includes methods like Register.](javascript:void(0)) [And based on what you know about the MVC framework so far, you can probably figure](javascript:void(0)) [out what most of the code inside of here is doing.](javascript:void(0)) [We have a register action that responds to an HTTP GET request](javascript:void(0)) [that gives the user a form to fill out their information.](javascript:void(0)) [And we have a register action that response to an HTTP POST request and takes information](javascript:void(0)) [from the user to create a record in the database.](javascript:void(0)) [We also have ModelState and we have a RegisterModel which is](javascript:void(0)) [like a view model containing only the information about a user that we need](javascript:void(0)) [in the view to get them registered.](javascript:void(0)) [These are all concepts we've looked at so far.](javascript:void(0)) [So let's focus on what we haven't seen which includes the call to WebSecurity.](javascript:void(0)) [WebSecurity is basically a wrapper around membership functionality.](javascript:void(0)) [It takes care of data access, cryptography, and all the other code](javascript:void(0)) [that you need for Forms-Based Authentication.](javascript:void(0)) [Here we can see when I successfully register, there's a call to CreateUserAndAccount,](javascript:void(0)) [that's what actually puts me into the database.](javascript:void(0)) [So what database do we go to?](javascript:void(0)) [Well, if you remember, we placed a call](javascript:void(0)) [to WebSecurity.InitializeDatabase in the application start event.](javascript:void(0)) [And we passed the parameter specifying the database connection name](javascript:void(0)) [to use which was DefaultConnection.](javascript:void(0)) [The same connection string that we used for storing restaurant information and that means](javascript:void(0)) [if I go to the Database Explorer, and look at our DefaultConnection and open this](javascript:void(0)) [up to see the tables, I will see not only Restaurants and RestaurantReviews](javascript:void(0)) [which we had before, but also the UserProfile table which we've just added in this module,](javascript:void(0)) [that's a table that we're in control of.](javascript:void(0)) [And then, there's a number of tables that are used](javascript:void(0)) [by WebSecurity, they all have a webpages\_prefix.](javascript:void(0)) [And if we open up some of these tables to poke around,](javascript:void(0)) [you can see the UserProfile table has an ID, UserName, and FavoriteRestaurant columns.](javascript:void(0)) [That's exactly what would we expect based on our definition for UserProfile.](javascript:void(0)) [And if I right-click on this and select Show Data, we'll see there's exactly one record](javascript:void(0)) [in here, the sallen user that I just created has a UserID but no FavoriteRestaurant](javascript:void(0)) [because we didn't forced anyone to fill that information out.](javascript:void(0)) [We don't have a place yet to fill that information out.](javascript:void(0)) [And now if I look into webpages\_Membership, what I'll find is it there is a corresponding record](javascript:void(0)) [in here that has the same UserID so we can join these two tables together](javascript:void(0)) [to get the whole picture for a user.](javascript:void(0)) [And this one includes information like a hash password for sallen.](javascript:void(0)) [WebSecurity will automatically hash password which is good](javascript:void(0)) [because then we're not storing the user's password in plain text in the database.](javascript:void(0)) [That's something you almost never, ever, ever want to do,](javascript:void(0)) [store plain text password in your database.](javascript:void(0)) [And there are some additional columns in here, too.](javascript:void(0)) [But overall, the database is now a combination of tables that we used to manage restaurants](javascript:void(0)) [and reviews and user profiles, and tables used by WebSecurity](javascript:void(0)) [and its friends to manage membership and roles.](javascript:void(0)) [And this database was created through a combination of WebSecurity](javascript:void(0)) [and our entity framework migrations.](javascript:void(0)) [And now that we know where the data is stored, let's go back to the AccountController](javascript:void(0)) [and take a look at the second WebSecurity call, the call to log in.](javascript:void(0)) [You'll also see this in the log in action which is where we come](javascript:void(0)) [after the user has clicked the log on link and they filled in the user name and their password,](javascript:void(0)) [they click Log in, we'll end up in this action that simply needs to take that user name](javascript:void(0)) [and password and pass it to WebSecurity for validation.](javascript:void(0)) [If the model says it is valid and WebSecurity says that we can log in,](javascript:void(0)) [what we'll do is redirect back to the ReturnUrl.](javascript:void(0)) [We'll see how that works in just a second.](javascript:void(0)) [You can think about what WebSecurity.Login has to do.](javascript:void(0)) [It has to, first of all, compare the hashed version of this password against the hash that's](javascript:void(0)) [in the database and see if they match.](javascript:void(0)) [And if they do, they can issue that cookie that we talked about.](javascript:void(0)) [Let's actually take a look and see if we can see that.](javascript:void(0)) [Here in Internet Explorer, on the log in page, let me press F12 to open the Developer Tools](javascript:void(0)) [and let's go to the Network tab and click Start Capturing.](javascript:void(0)) [This will capture network request.](javascript:void(0)) [And I'll now try to log in with my user name and password, I can click Remember Me](javascript:void(0)) [if I want the cookie to stick around between browsing sessions and click log in.](javascript:void(0)) [( Pause )](javascript:void(0)) [And here we can see what happened.](javascript:void(0)) [I posted my user name and password to /Account/Login.](javascript:void(0)) [It determine that I logged in successfully so it returned a 302 result which is a redirect,](javascript:void(0)) [which says, "Dear web browser, please go somewhere else."](javascript:void(0)) [So it redirected me back to that root of the application, the homepage as a logged in user.](javascript:void(0)) [And if we open up this response from the web server and go to the Response headers,](javascript:void(0)) [then what I want you to pay attention to is the Set-Cookie header.](javascript:void(0)) [This is how you tell a browser to accept a cookie.](javascript:void(0)) [The name of this cookie is .ASPXAUTH.](javascript:void(0)) [When decrypted will tell ASP.NET MVC that this user has successfully authenticated](javascript:void(0)) [to the application, there's no need to make them sign in again and now the browser is going](javascript:void(0)) [to send that cookie along on every subsequent request.](javascript:void(0)) [We can see that if we go back to the summary view and look at this request](javascript:void(0)) [that comes back into the homepage.](javascript:void(0)) [If I double-click on that, and now we look at the Request headers, so what the browser sends](javascript:void(0)) [to the server when it's requesting this page.](javascript:void(0)) [Down here in the Cookie section, if I double-click that,](javascript:void(0)) [you can see that we send along the ASPXAUTH cookie.](javascript:void(0)) [And that will actually go out on every subsequent request now until I close the browser](javascript:void(0)) [or actually it should stick around longer because I clicked on the Remember Me check box.](javascript:void(0)) [But the other thing I want to show you here if we go back to the summary view and look](javascript:void(0)) [at Account Logon, double-click on that to go into the details and look at the request body.](javascript:void(0)) [If I scroll over here, you can also see my UserName and my Password.](javascript:void(0)) [And so can anyone who might be sniffing network traffic between me and the web server.](javascript:void(0)) [So this is obviously not secure and this is why pretty much any reputable website that needs](javascript:void(0)) [to force its users to log in is going to use SSL.](javascript:void(0)) [They'll acquire an SSL certificate and install it on the web server](javascript:void(0)) [and then force the log in process to go through HTTPS.](javascript:void(0)) [If you go down that path, that's always a good idea to come into methods like this log](javascript:void(0)) [in method and add an attribute RequireHttps.](javascript:void(0)) [This is an action filter.](javascript:void(0)) [We've talked about action filters in this course but this is an action filter that will make sure](javascript:void(0)) [that the request that is arriving is coming over a secure encrypted connection.](javascript:void(0)) [And that would make sure that we're not passing any passwords in clear text.](javascript:void(0)) [No one should be able to see those traveling over the network.](javascript:void(0)) [And now that we know a little bit about how Forms Authentication works,](javascript:void(0)) [let's put on our developer hat and see how we can apply it to force users](javascript:void(0)) [to authenticate before they reach specific sections of the application.](javascript:void(0)) [( Pause )](javascript:void(0))

* + [Authorize](javascript:void(0))

[Now let's look at forcing a user to authenticate.](javascript:void(0)) [First, I'm going to remove the RequireHttps attribute that we put on the log on action](javascript:void(0)) [because I do not have an SSL certificate set up.](javascript:void(0)) [And instead, we'll talk about a different attribute, the Authorize attribute.](javascript:void(0)) [Let's imagine that for some reason, you have some secret members only information inside](javascript:void(0)) [of the About page for the application.](javascript:void(0)) [In that case, what I would want to do is use an Authorize attribute here.](javascript:void(0)) [Authorize sounds like it's more about authorization than authentication.](javascript:void(0)) [But when you use the Authorize attribute like this with no other parameters, essentially,](javascript:void(0)) [you're specifying the type of authorization that says, only authenticated users are authorize](javascript:void(0)) [to invoke this controller action.](javascript:void(0)) [So before we get into the About action, a user has to be logged in.](javascript:void(0)) [And just by having that attribute there, if I do a build and we come back out to the application,](javascript:void(0)) [let me go to the homepage and close the Developer Tools.](javascript:void(0)) [And from the homepage I'm currently logged in.](javascript:void(0)) [So I'll also log off and now try to go to the About page.](javascript:void(0)) [And ASP.NET MVC has detected that Authorize attribute.](javascript:void(0)) [It knows I have to be at least logged in into this application before it can view that.](javascript:void(0)) [So it redirects me to the log in page where I can log in.](javascript:void(0)) [And notice that the ReturnUrl includes /Home/About,](javascript:void(0)) [it's been URL encoded with percent 2f.](javascript:void(0)) [But it's /Home/About is the ReturnUrl, that way once I log in MVC will be able](javascript:void(0)) [to redirect me back to where I was trying to get to which is the About page.](javascript:void(0)) [Now if you're wondering how MVC knew to send me to /Account/Login when it detected that I needed](javascript:void(0)) [to authenticate, that is something that you can configure in an application.](javascript:void(0)) [If you look at the root web.config in the project, at the authentication section,](javascript:void(0)) [the mode is equal to forms, and we have a loginUrl specified](javascript:void(0)) [in the forms element underneath that authentication element.](javascript:void(0)) [So the loginUrl tells the MVC framework where to go to authenticate a user.](javascript:void(0)) [The timeout value here specifies how long the user will be authenticated after logging in.](javascript:void(0)) [And this number is specified in minutes, so 2,880 minutes.](javascript:void(0)) [But coming back to our Authorize attribute, like many of these attributes,](javascript:void(0)) [you can apply them at the action level.](javascript:void(0)) [You can also take something like the Authorize attribute and apply it at the controller level.](javascript:void(0)) [So I'll remove it from the About action](javascript:void(0)) [and let's just apply it to the entire HomeController.](javascript:void(0)) [Now the user would need to be authenticated to do anything with the HomeController.](javascript:void(0)) [If there were just a couple of things that you wanted a user to be able to do,](javascript:void(0)) [an anonymous user to be able to do, there is an AllowAnonymous attribute.](javascript:void(0)) [I can place that-- let's say on the index action.](javascript:void(0)) [That will at least allow a user to get to the homepage of the application without logging in.](javascript:void(0)) [So let's try this real quick.](javascript:void(0)) [Authorize is at the controller level.](javascript:void(0)) [AllowAnonymous is on the index action.](javascript:void(0)) [So if I log off then I come back out here to the homepage](javascript:void(0)) [and it's viewing the homepage just fine.](javascript:void(0)) [And if I try to go to About or Contact, now I need to log in.](javascript:void(0)) [But before I log in, let me show you a couple of other things that you can do with Authorize.](javascript:void(0)) [You can be a little more specific about who is allowed into the HomeController.](javascript:void(0)) [You can specify users.](javascript:void(0)) [So I could say Users equals sallen, plall.](javascript:void(0)) [So you can have a comma separated list of users.](javascript:void(0)) [And having that in place, if I login as sallen, I should be able to get somewhere.](javascript:void(0)) [So sallen and my password, and now I can get to the Contact page.](javascript:void(0)) [But it's not very frequently that you can specify specific users except](javascript:void(0)) [for perhaps an admin user or a super user.](javascript:void(0)) [It's more common often to include roles instead of specific users.](javascript:void(0)) [And then you could say roles equal something like administrators and sales.](javascript:void(0)) [These users and roles parameters, by the way,](javascript:void(0)) [I should point out that they also work with Windows Authentication.](javascript:void(0)) [When you specify users, you'd be specifying users in active directory.](javascript:void(0)) [When you specify roles, you'd be specifying groups in Windows.](javascript:void(0)) [But now that I've added roles to my Authorize attribute, I've hit a bit of a stumbling block](javascript:void(0)) [because I do not have any roles defined.](javascript:void(0)) [We saw in the Database Explorer that there is a table where you can store roles,](javascript:void(0)) [but I don't have any user interface available for me to create roles.](javascript:void(0)) [We'll look at a way to populate some of these membership tables next.](javascript:void(0)) [( Pause )](javascript:void(0))

* + [Seeding Membership](javascript:void(0))

[What I'd like to do now is show you how](javascript:void(0)) [to automatically populate your database with some users and roles.](javascript:void(0)) [Having a database with users and roles predefined is nice to have during development](javascript:void(0)) [because if you're building an application that restricts access to specific pages based](javascript:void(0)) [on the user's role membership then needing to set up those roles when you start](javascript:void(0)) [with a fresh project only slows you down.](javascript:void(0)) [I'm a big believer in making it easy to get started working on a project.](javascript:void(0)) [There're several techniques that you could use to populate the database.](javascript:void(0)) [You could, just write SQL statements that insert data](javascript:void(0)) [into the membership tables we looked at earlier.](javascript:void(0)) [But I want to show you how to do this with the membership API that is given](javascript:void(0)) [to us by the SimpleMembershipProvider.](javascript:void(0)) [Remember that's one of the components that working behind the layers](javascript:void(0)) [of other components in our MVC 4 application.](javascript:void(0)) [And since this provider owns the schema, I think it would be more robust](javascript:void(0)) [to populate the database using this provider APIs.](javascript:void(0)) [The question then is when and where to populate these roles that we need?](javascript:void(0)) [We could during the application start up, make sure that the roles exist in the database.](javascript:void(0)) [But since we're using entity framework migrations, we can also use the Seed method](javascript:void(0)) [of our entity framework migration class.](javascript:void(0)) [Remember, that's in the Migrations folder.](javascript:void(0)) [It's called Configuration.cs.](javascript:void(0)) [This is where we added a thousand restaurants into the database](javascript:void(0)) [so we could do some testing in our AJAX module.](javascript:void(0)) [So inside of the Seed method, I'll add a call to a new method](javascript:void(0)) [that we're going to write called SeedMembership.](javascript:void(0)) [I can use Visual Studio to create this method for me if I just hit Ctrl period](javascript:void(0)) [and select that option in the drop down.](javascript:void(0)) [That will generate a method stub for me down here that I can now fill in.](javascript:void(0)) [The first thing we'll do here is use a call](javascript:void(0)) [to WebSecurity dot InitializeDatabaseConnection just to make sure that everything is set up](javascript:void(0)) [and the schema is in place for the SimpleMembershipProvider.](javascript:void(0)) [I do need to bring in a namespace for this, it's WebMatrix.WebData.](javascript:void(0)) [We're using the same parameters that we're using during application start](javascript:void(0)) [up when we call this method.](javascript:void(0)) [It probably would be a good idea to take this two lines of code and put them](javascript:void(0)) [in a static method in a class somewhere so that I can call them from both SeedMembership](javascript:void(0)) [and also the Global.asax.cs application start event](javascript:void(0)) [and not have this hard coded string (inaudible) duplicated everywhere.](javascript:void(0)) [But I'll leave that as an exercise for the viewer.](javascript:void(0)) [So once the database is set up, I can go out and get access to the current role provider](javascript:void(0)) [and the current membership provider by walking up to a property called Provider](javascript:void(0)) [on the roles class and the membership class.](javascript:void(0)) [These are static properties.](javascript:void(0)) [They give you the current provider that's in effect.](javascript:void(0)) [I do need to bring a namespace for this to work, System.Web.Security.](javascript:void(0)) [But now I have access to two objects that I can use to create users or see if a user exists,](javascript:void(0)) [see what roles the user is in, all the good things that you need to do with membership.](javascript:void(0)) [So there's not just a single membership provider that does both users and roles.](javascript:void(0)) [Those are actually a little more fine-grained than that.](javascript:void(0)) [There's a role provider and there's a membership provider.](javascript:void(0)) [I just think of them both really as the SimpleMembershipProvider.](javascript:void(0)) [And here is the API that we can use with them.](javascript:void(0)) [We can walk up to the role provider and say,](javascript:void(0)) ["Is there a role called admin., if not create the role."](javascript:void(0)) [And then, "Is there a user with user name sallen,](javascript:void(0)) [if not create that user and here's the password."](javascript:void(0)) [And finally, check to see if sallen is in the admin role, and if not,](javascript:void(0)) [add that sallen user to the admin role.](javascript:void(0)) [We always want to check if things exist first before we just add them](javascript:void(0)) [because remember the Seed method runs every time](javascript:void(0)) [that we execute update database in the Package Manager Console.](javascript:void(0)) [We'll be doing that many times as we change the schema and change our models](javascript:void(0)) [through development, and we wouldn't want this Seed membership method to try](javascript:void(0)) [to insert duplicate data that will actually generate an exception.](javascript:void(0)) [But with that code in placed, let me come in to the Package Manager Console](javascript:void(0)) [and run update database, that will execute the Seed method](javascript:void(0)) [and it will have a disappointing result.](javascript:void(0)) [And the disappointing result is an exception that has been thrown saying](javascript:void(0)) [that the role manager feature has not been enabled.](javascript:void(0)) [It took me a bit of time and debugging to figure this out.](javascript:void(0)) [But I finally realized that I needed to add some explicit configuration](javascript:void(0)) [to the web.config file for this to work.](javascript:void(0)) [So the SimpleMembership stuff just works inside of your MVC application.](javascript:void(0)) [That's a real web application.](javascript:void(0)) [But here when we're running update database, part of the entity framework migrations.](javascript:void(0)) [This isn't a web application and it seems to require a little bit](javascript:void(0)) [of configuration to get this to work.](javascript:void(0)) [So the configuration goes into web.config](javascript:void(0)) [because this is the configuration that migrations will use.](javascript:void(0)) [It has all the settings that are in place when that migrations process executes.](javascript:void(0)) [And the bit of configuration I have to put in here is just](javascript:void(0)) [to explicitly configure a role manager which is the simple role provider](javascript:void(0)) [and set it to enabled equals true.](javascript:void(0)) [And also explicitly configure the SimpleMembershipProvider](javascript:void(0)) [as the membership provider.](javascript:void(0)) [And just having that bit of config in placed and saving the file,](javascript:void(0)) [I should be able to update database again.](javascript:void(0)) [And now, we'll have users and roles in the database for sure.](javascript:void(0)) [So that worked.](javascript:void(0)) [And don't take my word for it that it just works behind the scenes.](javascript:void(0)) [Let's actually test it out.](javascript:void(0)) [I'm going to remove the Authorize attribute from the HomeController and we'll place it somewhere](javascript:void(0)) [where we might actually want some authorization.](javascript:void(0)) [So I'll remove that attribute and while I'm at it,](javascript:void(0)) [I'll remove this AllowAnonymous attribute from the index method.](javascript:void(0)) [It makes no sense if nothing in here is-- needs authorization.](javascript:void(0)) [And instead, let's pretend that we only want to allow administrators](javascript:void(0)) [to be able to create restaurants.](javascript:void(0)) [That means making a change in the restaurant controller.](javascript:void(0)) [Specifically, we'll add an Authorize attribute to the create action](javascript:void(0)) [and say that this can only be done by users in the admin role.](javascript:void(0)) [And putting this on the create action that responds to a post,](javascript:void(0)) [we'll make sure that no one can create a restaurant unless they're an admin.](javascript:void(0)) [But I will also put it on the create action that responds to a get request,](javascript:void(0)) [just to make sure that someone who is not authorized doesn't come in](javascript:void(0)) [and try to create restaurant and fill out all the information](javascript:void(0)) [and they don't have the proper permissions that would be a disappointment.](javascript:void(0)) [Speaking of which, it might also make sense to go into the Index view for restaurant and here](javascript:void(0)) [with the Create New restaurant link appears, we might want to make sure](javascript:void(0)) [that we only display this link if the user is an admin.](javascript:void(0)) [Again, that mean standard users won't click that and get confused when they have to login.](javascript:void(0)) [So there's an API for this, just User.IsInRole, the same user property](javascript:void(0)) [where we can see User.Identity.Name.](javascript:void(0)) [There's also an IsInRole check, you just pass in the name of the role and that will return false](javascript:void(0)) [if the user is not an admin and we won't show that link.](javascript:void(0)) [Let's try it out.](javascript:void(0)) [I'll run the application and we'll come in to look at the list of restaurant.](javascript:void(0)) [And since I'm not logged in, I do not see it create new link.](javascript:void(0)) [So let me log in and now my database has an admin role.](javascript:void(0)) [We've seeded that into the database.](javascript:void(0)) [And I put sallen in that role.](javascript:void(0)) [So once I logged in and come in to the list of restaurants.](javascript:void(0)) [Now, I can see the Create New button.](javascript:void(0)) [And if I were to go directly to this Create action as a regular user, so I'll log off](javascript:void(0)) [and paste that URL in here, the MVC framework would decide that it needs to authenticate me](javascript:void(0)) [in order to see what role I'm in and that's where I could log in.](javascript:void(0)) [And I can now get to that Create page.](javascript:void(0)) [And I feel like I can sleep well at night now,](javascript:void(0)) [knowing that only administrators can create restaurants in the application.](javascript:void(0)) [My application is error tight and a malicious user should never be able to get](javascript:void(0)) [through this authorization checks.](javascript:void(0)) [Or can they?](javascript:void(0)) [( Pause )](javascript:void(0))

* + [A cross site request forgery](javascript:void(0))

[One type of attack, you will potentially face](javascript:void(0)) [in a web application is a Cross-Site Request Forgery.](javascript:void(0)) [I want to make you aware of this attack because it's subtle](javascript:void(0)) [and potentially dangerous and it's also easy to prevent.](javascript:void(0)) [A Cross-Site Request Forgery or CSRF is an attack a malicious user can execute using](javascript:void(0)) [unsuspecting users who are logged in to your site.](javascript:void(0)) [A CSRF attack might do something a user will find annoying like log them](javascript:void(0)) [out of the site or change the profile name.](javascript:void(0)) [But this attack can also be destructive and steal or destroy data.](javascript:void(0)) [Let me give you an example.](javascript:void(0)) [Let's imagine I'm a bad person who wants to create a restaurant on OdeToFood,](javascript:void(0)) [but I don't have administrator access.](javascript:void(0)) [So what I will do is go out and create my own web application.](javascript:void(0)) [I'll start a new wen application here with Visual Studio, again,](javascript:void(0)) [it'll just be a throw away application.](javascript:void(0)) [I'll create an MVC 4 application but it doesn't need to be an MVC application.](javascript:void(0)) [It could just be an HTML page that I upload somewhere on my website.](javascript:void(0)) [I'll use an empty project template.](javascript:void(0)) [And now that that's ready, all I'm really trying to do here is build an application](javascript:void(0)) [that I can run in IIS Express and point out that it's a different application](javascript:void(0)) [than OdeToFood, so it's someone else's website.](javascript:void(0)) [And then inside of here, I will add a new item, just a regular HTML page that can be served](javascript:void(0)) [up from my site, let me call it, funnypictures.html.](javascript:void(0)) [And my goal here is to execute a CSRF attack against someone who is logged in to OdeToFood.](javascript:void(0)) [In order to do it, I usually need to know a little bit about what HTML is on the site](javascript:void(0)) [that I want to attack, but that's usually not too hard to get,](javascript:void(0)) [and the HTML that I'm particularly interested in is the form](javascript:void(0)) [that is used to create a new restaurant.](javascript:void(0)) [So let me copy it out of the source view of this page and paste that into my HTML page.](javascript:void(0)) [And then just clean it up a bit because we really only need a form tag and the inputs](javascript:void(0)) [for name, the city, and the country, and hit control K,](javascript:void(0)) [control D to do some formatting, that looks better.](javascript:void(0)) [And now I'll add the values that I want to get into OdeToFood through an unsuspecting user,](javascript:void(0)) [my values and this could represent trying to transfer money or change someone's profile,](javascript:void(0)) [anything on a site that only an authenticated user can get to, that's what I want to get to](javascript:void(0)) [and place my own values into the application.](javascript:void(0)) [And I'll do just a couple of other things to dress this up.](javascript:void(0)) [Let's give the form an ID.](javascript:void(0)) [I'll just call it the form.](javascript:void(0)) [Let me add in inline style.](javascript:void(0)) [So nothing inside this form actually displays to the user.](javascript:void(0)) [The goal is not to have the user fill this out, the goal is for me to trick the user](javascript:void(0)) [into taking this HTML and submitting it to OdeToFood application behind their backs.](javascript:void(0)) [And that means the action, we're not going to be posting back to my website.](javascript:void(0)) [My website doesn't have anything to do with restaurants.](javascript:void(0)) [It's just full of malicious pages.](javascript:void(0)) [I'm going to change that to postback to the real website which is over here](javascript:void(0)) [on a different port number on this machine.](javascript:void(0)) [And I don't even need the user to click a button because I'm going to add some script here](javascript:void(0)) [that will just automatically submit this form after a 2,000 millisecond delay.](javascript:void(0)) [So what I've done essentially is craft a form to look just like the form that OdeToFood expects.](javascript:void(0)) [And now if I run this application or just view this in a browser, as me the bad person,](javascript:void(0)) [the malicious user that form gets submitted to OdeToFood which redirects me to the login page](javascript:void(0)) [because I need to login but unfortunately I'm not an administrator.](javascript:void(0)) [However, if I take this URL and somehow get it to another user who is an administrator,](javascript:void(0)) [maybe I'll send a link in the e-mail, use some social engineering,](javascript:void(0)) [get them to click on something else that's on another page, thinking they're going](javascript:void(0)) [to download a free song or something.](javascript:void(0)) [If I get that person to do it, someone who is logged in to OdeToFood all they need](javascript:void(0)) [to do is click a link and come to this page and then I get a value into the database.](javascript:void(0)) [It will be down here at the bottom.](javascript:void(0)) [And of course there's all sorts of things I could do to make this less obvious to the user](javascript:void(0)) [that they just posted something to OdeToFood.](javascript:void(0)) [I could use jQuery to do the submission in the background](javascript:void(0)) [so they never even see the OdetoFood website.](javascript:void(0)) [All they see is that page of line where I put a funny picture on there.](javascript:void(0)) [Behind the scene something is happening on OdeToFood that they don't know about.](javascript:void(0)) [And me the bad person has used someone who is an administrator on OdeToFood as a proxy](javascript:void(0)) [and channeled my malicious data through them to get it into the database.](javascript:void(0)) [That's a Cross Site Request Forgery.](javascript:void(0)) [It works because that other user is logged in to the application.](javascript:void(0)) [And because they're logged in to the site, their browser has a cookie.](javascript:void(0)) [And because their browser has a cookie, the browser sends that cookie along on every request](javascript:void(0)) [that it makes to OdeToFood even if the user didn't intent for that request to go](javascript:void(0)) [to OdeToFood, even if me, the malicious person tricked them into sending a request to OdeToFood](javascript:void(0)) [that carried along something dangerous.](javascript:void(0)) [So when you are building an application that uses forms authentication and you have areas](javascript:void(0)) [with form posts that you want to protect with authorization rules, then it's not enough](javascript:void(0)) [to just authenticate a user, and authorize them based on their role.](javascript:void(0)) [You have to make sure that the form they're submitting to you is a form that you gave them](javascript:void(0)) [and not a form that a malicious user tricked them into sending.](javascript:void(0)) [It should be your restaurant create form not someone else's form from another website.](javascript:void(0)) [And fortunately, this is easy to do with ASP.NET MVC.](javascript:void(0)) [It has the concept of an anti-forgery token to ensure that the form post that is coming](javascript:void(0)) [in when you create a restaurant is something that you gave the user.](javascript:void(0)) [The way we use an anti-forgery token is to go to the restaurant controller](javascript:void(0)) [and on this create action, the destination for a post put an attribute](javascript:void(0)) [to say validate anti-forgery token.](javascript:void(0)) [This is part of the solution.](javascript:void(0)) [And just having that attribute in place, if I build the application come back out as a user](javascript:void(0)) [who is logged in to the site and try to create something new with any name, any city,](javascript:void(0)) [any country, I'll get an exemption](javascript:void(0)) [that there was no request verification token that I submitted.](javascript:void(0)) [So the way this anti-forgery token works is that we need to go into the form](javascript:void(0)) [that will post this controller action and use an HTML helper](javascript:void(0)) [that will add a verification token into the form.](javascript:void(0)) [The verification token is a hidden input that holds a cryptographically significant value.](javascript:void(0)) [So that is basically, added to every form that the user is going](javascript:void(0)) [to submit into an authorized area.](javascript:void(0)) [That cryptographically significant value is also placed into a cookie in the user's browser.](javascript:void(0)) [So when the user submits the form, the form value has](javascript:void(0)) [to match the cookie value before MVC will allow that request through.](javascript:void(0)) [The idea is that even if a malicious user gives one of your members some evil HTML,](javascript:void(0)) [they will not be able to set that cookie in your user's browser](javascript:void(0)) [because browsers don't allow one site to set cookies for a different site](javascript:void(0)) [and that is how an anti-forgery token will prevent a CSRF attack.](javascript:void(0)) [So the evil application that I was building over here,](javascript:void(0)) [it wouldn't be enough just to have these inputs anymore.](javascript:void(0)) [I would also need to figure out what the user's anti-forgery token is and set a cookie](javascript:void(0)) [in their browser, that's just impossible for me to do, so I would give up.](javascript:void(0)) [But we need to get this working in the real application](javascript:void(0)) [and I have validate anti-forgery token that attribute is on my create action.](javascript:void(0)) [I also need to go into the create view for a restaurant and for this to work inside of Form,](javascript:void(0)) [I'll just use Html.AntiForgeryToken.](javascript:void(0)) [That is all that it's needed and now if I come into the browser let me refresh this page.](javascript:void(0)) [And I should be able to create a restaurant with a name of two and a city,](javascript:void(0)) [and a country of two just to see if this works.](javascript:void(0)) [And down here at the bottom on my list I can see now, I can successfully add that.](javascript:void(0)) [So one more quick summary if you're building something that uses forms authentication,](javascript:void(0)) [you have form post coming in to authorized areas add an anti-forgery token to the form](javascript:void(0)) [and validate that anti-forgery token in the action that is a destination for the post.](javascript:void(0)) [Taking these two steps will help to prevent a Cross Site Request Forgery.](javascript:void(0))

* + [OpenID and OAuth](javascript:void(0))
  + [Summary](javascript:void(0))
* ASP.NET MVC 4 Infrastructure
  + [Introduction](javascript:void(0))

[Hi, this is Scott Allen and in this module I'll show you how to use some](javascript:void(0)) [of the infrastructure features of ASP.net inside an MVC application.](javascript:void(0)) [In this module, I'll show you how to use the underlying ASP.net caching engine](javascript:void(0)) [to improve the performance of an application.](javascript:void(0)) [We'll also see how to work with ASP.net resource files to globalize](javascript:void(0)) [and localize some application to support multiple cultures.](javascript:void(0)) [Finally I'll show you some techniques you can use to enable diagnostics](javascript:void(0)) [and logging inside of an MVC application.](javascript:void(0)) [( Silence )](javascript:void(0))

* + [Caching](javascript:void(0))

[Output caching allows you to store the output of a particular controller action in memory.](javascript:void(0)) [ASP.net can then respond to future requests for the same action just](javascript:void(0)) [by giving back the cached result and that avoids executing any of the code inside the action.](javascript:void(0)) [Because you're executing less code and doing less work,](javascript:void(0)) [caching can lead to some pretty dramatic performance gains but you do have to be able](javascript:void(0)) [to careful when using the cache to avoid erroneous responses.](javascript:void(0)) [I'll show you such a scenario in our demo.](javascript:void(0)) [Caching itself is easy to enable using the output cache action filter.](javascript:void(0)) [You can cache actions that produce few results and jayson (phonetic) results](javascript:void(0)) [or really any type of data oriented result.](javascript:void(0)) [You'd primarily want to cache the actions that get called the most or the ones](javascript:void(0)) [that are expensive because they execute a slow database query.](javascript:void(0)) [Those are some of those places where you would get your biggest win](javascript:void(0)) [with the output cache attribute.](javascript:void(0)) [The attribute will also work on child (phonetic) actions as we'll see and there's a variety](javascript:void(0)) [of options you can specify on the attribute itself like the duration](javascript:void(0)) [that the cache items should leave in memory, that's specified in seconds](javascript:void(0)) [and what parameters should bury the cache.](javascript:void(0)) [This is best understood through an example.](javascript:void(0)) [( Silence )](javascript:void(0)) [Inside the home controller for the application, let's add an output cache attribute here.](javascript:void(0)) [We'll add it to the index action.](javascript:void(0)) [The index action represents the home page of the application](javascript:void(0)) [and chances are it might get the most traffic of the entire site.](javascript:void(0)) [And I say might because I don't really know.](javascript:void(0)) [One of the tips to implementing a successful caching strategy is you really do have](javascript:void(0)) [to know where your traffic is going.](javascript:void(0)) [You have to know what are the most expensive operations in your software.](javascript:void(0)) [You have to take some measurements and do some logging in order for caching](javascript:void(0)) [to really work effectively but this is just a demonstration](javascript:void(0)) [of how the output cache attribute works.](javascript:void(0)) [So I'll place the attribute there.](javascript:void(0)) [Set a break point at the beginning of this action and then run with the debugger.](javascript:void(0)) [( Silence )](javascript:void(0)) [And what we should see here is on this first request, we will hit that break point](javascript:void(0)) [so we're inside of the index action and you'll notice the one parameter I have to specify](javascript:void(0)) [with the output cache attribute is the duration.](javascript:void(0)) [How long do I cache the response?](javascript:void(0)) [It's specified in seconds and this is 60 seconds.](javascript:void(0)) [So now if I press F5 to continue with the debugger, I can now see the home page](javascript:void(0)) [and now I can refresh this as many times as I want and the caching logic inside](javascript:void(0)) [of ASP.net is looking at the request, seeing that there's a cached response for this request.](javascript:void(0)) [So it doesn't even go into my code at all.](javascript:void(0)) [It just returns that cached response.](javascript:void(0)) [We don't execute code inside of the index action.](javascript:void(0)) [We don't render any views.](javascript:void(0)) [Everything is just a cached response that's being served back from memory](javascript:void(0)) [and not executing code is the best optimization of all.](javascript:void(0)) [It's the best way to increase performance and scalability.](javascript:void(0)) [Now this also works with child actions.](javascript:void(0)) [So let's stop debugging and let me add another action here.](javascript:void(0)) [It's going to be a child action.](javascript:void(0)) [It's called say hello.](javascript:void(0)) [It's also going to be cached and we'll say for 60 seconds.](javascript:void(0)) [And I went to change the duration of the output cache for the index action to just 5 seconds](javascript:void(0)) [and now inside of the view for this index action,](javascript:void(0)) [we will need to call and render this child action.](javascript:void(0)) [So let me go into index.cshtml and do a quick html.action.](javascript:void(0)) [Please render say hello.](javascript:void(0)) [Call into that controller action.](javascript:void(0)) [Save everything and let's run with the debugger again](javascript:void(0)) [after I set a break point here, inside of say hello.](javascript:void(0)) [So F5 and on this first request, we should see that we come into the index action which we do](javascript:void(0)) [and we come into the say hello child action.](javascript:void(0)) [So I'll continue debugging and now if I refresh, more than 5 seconds elapsed.](javascript:void(0)) [So we're back inside the index action and we need to re-execute all of that code](javascript:void(0)) [but we don't need to execute say hello again.](javascript:void(0)) [So the MVC framework is able to cache that small part of the response independently](javascript:void(0)) [and when I refresh the page, yes it has to rebuild everything else](javascript:void(0)) [and execute the database inquiry inside the index action](javascript:void(0)) [but it doesn't to call back in to say hello.](javascript:void(0)) [If I had something that was particularly expensive to produce](javascript:void(0)) [and it was relatively static so I could aggressively cache it, that's a good place](javascript:void(0)) [to use html.action just to cache the result of something expensive.](javascript:void(0))

* + [Cache Settings](javascript:void(0))

[( Silence )](javascript:void(0)) [In addition to duration, the output cache attribute supports a number](javascript:void(0)) [of different settings.](javascript:void(0)) [These settings are all available from the underlying ASP.net cache engine.](javascript:void(0)) [It's the same engine that web forms will use](javascript:void(0)) [which is why I call caching part of the ASP.net infrastructure.](javascript:void(0)) [It's not something that's specific to just the MVC framework](javascript:void(0)) [but one of these settings is the vary by (phonetic) parameter.](javascript:void(0)) [The default setting for this is star or asterisk which means vary by every parameter possible](javascript:void(0)) [and that is usually the setting that you want.](javascript:void(0)) [That's the default because you normally do not want to return the same cached response](javascript:void(0)) [for different parameters and by parameters think of things like query string parameters.](javascript:void(0)) [A query string parameter can contain a search term and you wouldn't want](javascript:void(0)) [to return the same cached response when someone is searching for a](javascript:void(0)) [and b. Those would be two different results for two different search strings but if I do want](javascript:void(0)) [to take control over this, I could say vary by param equals none](javascript:void(0)) [which means don't vary by any parameter.](javascript:void(0)) [Always return the same response regardless of what you see in the query string](javascript:void(0)) [or I can say vary by and specify the name of a parameter that I want to vary by.](javascript:void(0)) [And you can have multiple names in there separated by a semicolon.](javascript:void(0)) [So if I were to say vary by a param search term and someone searches](javascript:void(0)) [and passes along the parameter value of a. That would be one cached response that ASP.net saves.](javascript:void(0)) [A little bit someone comes along and passes a search term of z.](javascript:void(0)) [That would be a second cached result that the ASP.net saves and it would only use](javascript:void(0)) [that in response to another request that came](javascript:void(0)) [in that says search term equals z. We'll see that in demo in just a bit.](javascript:void(0)) [You can also set the cache location.](javascript:void(0)) [The default value here is anywhere meaning it will cache on the server](javascript:void(0)) [and the client can also cache the result.](javascript:void(0)) [You can be very specific and say that the result should only be cached on the server](javascript:void(0)) [or on the client or on proxy servers in between.](javascript:void(0)) [There's also a vary by header setting.](javascript:void(0)) [That allows you to vary the cache based](javascript:void(0)) [on a specific HTTB (phonetic) header like except language.](javascript:void(0)) [We wouldn't want to return a response](javascript:void(0)) [that included English text when someone needs German text.](javascript:void(0)) [We'll see how except language works in this module.](javascript:void(0)) [And if all fails, you can always do vary by custom.](javascript:void(0)) [When you do vary by custom you'll need](javascript:void(0)) [to override a method inside the global.asax.cs file.](javascript:void(0)) [Inside of that method, you can build your own custom caching string telling ASP.net what](javascript:void(0)) [to cache and how to categorize it.](javascript:void(0)) [You could look at anything in the request including headers or quickie values.](javascript:void(0)) [There's some examples on MSDN that will show you how to do that.](javascript:void(0)) [There's also examples that will show you how to setup a sequel dependency.](javascript:void(0)) [A sequel dependency will cache a response until data inside of a sequel server table changes.](javascript:void(0)) [It sounds great but it's not widely used just because there are a lot of restrictions](javascript:void(0)) [on the type of sequel query that can be used.](javascript:void(0)) [( Silence )](javascript:void(0)) [In Visual Studio, now that we know how the output cache attribute works](javascript:void(0)) [with a child action, I'm going to remove this child action](javascript:void(0)) [because it's not really providing any value.](javascript:void(0)) [It was just here for demonstration purposes.](javascript:void(0)) [(Silence) And I'll also need to remove the call to html.action](javascript:void(0)) [that was rendering that child action.](javascript:void(0)) [And now that we've made that change, let's bump up the duration on the output cache attribute](javascript:void(0)) [for our index action and run this with the debugger to see how it behaves with paging.](javascript:void(0)) [And as expected that first request comes into index action, I press F5.](javascript:void(0)) [We now have a cached response for the home page and I can come up and pound](javascript:void(0)) [on the refresh button as much as I like.](javascript:void(0)) [The ASP.net run time is always returning a cached response.](javascript:void(0)) [Now let me try to go to the next page.](javascript:void(0)) [We'll come back into the index action because the output cache directive will vary](javascript:void(0)) [by every parameter and previously, we didn't have a page parameter.](javascript:void(0)) [Now we do have a page parameter.](javascript:void(0)) [The page number is equal to 2.](javascript:void(0)) [So it needs to pick up a new response and cache that.](javascript:void(0)) [Now we have cached response for page 2.](javascript:void(0)) [I go to the next page, same thing.](javascript:void(0)) [We come into the index action, produce that response for page 3,](javascript:void(0)) [and now I have cached responses for pages 2 and 3.](javascript:void(0)) [I can toggle between the two.](javascript:void(0)) [It's not going to go back into the index action.](javascript:void(0)) [So does this also work for search?](javascript:void(0)) [Let's search for a 20.](javascript:void(0)) [We come into the index action.](javascript:void(0)) [That changed the search term parameter.](javascript:void(0)) [We get the response and it appears on the screen correctly.](javascript:void(0)) [What if I go to 21?](javascript:void(0)) [Again into the index action, come back to 20.](javascript:void(0)) [That just serves up a cached response.](javascript:void(0)) [Let me show you a problem.](javascript:void(0)) [What if I come directly to the application through a bookmark](javascript:void(0)) [and I'm looking for search term equals 20?](javascript:void(0)) [Well now the browser issued a get request not asynchronous request but a get request](javascript:void(0)) [and it's displaying a cached response that was only a portion of the page.](javascript:void(0)) [So we're missing the layout view.](javascript:void(0)) [We're missing our style sheets.](javascript:void(0)) [We're missing our scripts and here's the problem.](javascript:void(0)) [The only difference between a full request which is what I just did](javascript:void(0)) [that needs the entire page full of html and the AJAX request that we were using previously,](javascript:void(0)) [the only difference between those two is an HTTB header known as x requested with.](javascript:void(0)) [So the ASP.net caching layer, there's no change in the parameters.](javascript:void(0)) [So it will happily return the partial view that it cached to fulfill a previous request](javascript:void(0)) [but that's not what we want in this scenario and searching by going directly to the page](javascript:void(0)) [with the value in the query string, if that's a supported scenario we need to fix this.](javascript:void(0)) [Actually there's two things that we need to do.](javascript:void(0)) [First, I'll stop debugging.](javascript:void(0)) [( Silence )](javascript:void(0)) [And we'll come into our output cache attribute and also tell it vary by a header.](javascript:void(0)) [The x requested with header because this will be present on an AJAX request](javascript:void(0)) [but not on a whole page request and now ASP.net will be able to tell the difference](javascript:void(0)) [between the two and cached different responses.](javascript:void(0)) [One with a full page of html, one with just the portion of the page that needs to update.](javascript:void(0)) [Unfortunately this often isn't enough.](javascript:void(0)) [We're also allowing the browser to cache the response with this output cache attribute](javascript:void(0)) [and some browsers won't be smart enough to detect the difference](javascript:void(0)) [between those two requests either.](javascript:void(0)) [So I am also going to add a location parameter.](javascript:void(0)) [To do this I need to bring in a name space for this numerated value, system.web.ui](javascript:void(0)) [but now we're only going to allow caching on the server.](javascript:void(0)) [And ASP.net will send instructions to the browser](javascript:void(0)) [and the response so the response isn't cached.](javascript:void(0)) [The browser always has to come back to the server and check.](javascript:void(0)) [On the server, ASP.net can just serve up the response](javascript:void(0)) [from memory if it has a cached response.](javascript:void(0)) [Of course another solution to this problem would be to call a different action for AJAX requests](javascript:void(0)) [and that would make a lot of sense.](javascript:void(0)) [I combine the two into a single action here because it made it easy for a demo](javascript:void(0)) [but it would be easy to change the url for an AJAX request in our JAVA script](javascript:void(0)) [and have it call a different action.](javascript:void(0)) [And then we wouldn't have this check for request.AJAXrequest.](javascript:void(0)) [We would only return a full response.](javascript:void(0)) [That is AJAX request method, by the way also looks at the (inaudible) requested with header](javascript:void(0)) [to determine if a request is an asynch request.](javascript:void(0)) [Now we'll run the application again and just to avoid any strange things.](javascript:void(0)) [I'm going to go into the developer tools by pressing F12 and just telling Internet Explorer,](javascript:void(0)) [clear out anything that you might have previously cached.](javascript:void(0)) [This way we'll make sure that we're starting with a fresh slate when I test](javascript:void(0)) [out this functionality with the new parameter values in place.](javascript:void(0)) [So let's try the search for 20 and we hit the index action.](javascript:void(0)) [That was good.](javascript:void(0)) [Try a search for 21.](javascript:void(0)) [Again, hitting the index action.](javascript:void(0)) [Try an asynchronous search for 20.](javascript:void(0)) [That was a cached response which was good and now let's go to the query string](javascript:void(0)) [and enter a full request for search term equals 20 and we do, indeed,](javascript:void(0)) [come into the index action and the page renders normally.](javascript:void(0)) [And that's all thanks to doing a vary by header and only caching on the server.](javascript:void(0)) [If we wanted the client browser to be able to cache too, we really would need](javascript:void(0)) [to split this index action into two pieces.](javascript:void(0)) [One that only responds to an AJAX request and one that responds for a full page.](javascript:void(0))

* + [Cache Profiles](javascript:void(0))

[( Silence )](javascript:void(0)) [In reality, it is almost impossible to pick the correct cache duration setting](javascript:void(0)) [in a development environment.](javascript:void(0)) [You need to put the application into a production like environment.](javascript:void(0)) [Put it under load, take some measurements, and then make some adjustments.](javascript:void(0)) [You might found out that you're using too much memory and caching requests](javascript:void(0)) [that just aren't being used or you might find that you're not caching enough.](javascript:void(0)) [For these reasons, ASP.net allows you to specify a cache profile in the output cache attribute.](javascript:void(0)) [A cache profile then is something that's stored](javascript:void(0)) [in your web.config file and can specify a duration.](javascript:void(0)) [It's underneath a section named caching and it's inside of here](javascript:void(0)) [where you can put duration values.](javascript:void(0)) [As this example demonstrates, you can have multiple cash profiles](javascript:void(0)) [and you reference the cache profile by name in the output cache attribute.](javascript:void(0)) [I generally recommend using cache profiles instead of hard code and cache settings](javascript:void(0)) [because first of all, it avoids repetition in the cache attributes.](javascript:void(0)) [If you want to double the duration inside of all of your cache attributes,](javascript:void(0)) [you don't have to do a search and replace throughout the code.](javascript:void(0)) [You just go to one location in your web.config file.](javascript:void(0)) [It's also easier to change the cache settings once the application is deployed](javascript:void(0)) [because now all you do is edit the config file and then you avoid changing the settings](javascript:void(0)) [in code, recompiling, and redeploying.](javascript:void(0)) [And typically when you're doing performance tuning on an application,](javascript:void(0)) [you need to run it while taking some measurements.](javascript:void(0)) [Then you tweak the cache settings and then you run it again](javascript:void(0)) [to make sure you did really get better results and it's not getting worse.](javascript:void(0)) [And you'll do this over and over again until you reach some goal like reduced memory usage](javascript:void(0)) [or reduced CPU or an increase in how many requests per second your application handles.](javascript:void(0)) [( Silence )](javascript:void(0)) [Inside our application if I wanted to use a cache profile instead of this hard code](javascript:void(0)) [of duration, all I would need to do is come into the root level web.config and somewhere inside](javascript:void(0)) [of system.web, create a caching element.](javascript:void(0)) [We're going to create output cache settings.](javascript:void(0)) [Notice you get intel sense (phonetic) the entire way through and we're going](javascript:void(0)) [to create multiple output cache profiles.](javascript:void(0)) [So let's add one called long that has a duration of 300 seconds and add another one](javascript:void(0)) [with a duration of 3 seconds called short.](javascript:void(0)) [And of course, you could have as many of these as you want.](javascript:void(0)) [Now I just need to reference one of these from the home controller.](javascript:void(0)) [So instead of having a duration, have cache profile equals long.](javascript:void(0)) [And the application should behave just like it did before.](javascript:void(0)) [Just now I have my numbers inside of a config file instead](javascript:void(0)) [of hard coded inside of the controller.](javascript:void(0)) [( Silence )](javascript:void(0))

* + [Localization](javascript:void(0))

[The next topic we want to talk about it is how to localize an application.](javascript:void(0)) [There are two important settings for every thread of execution](javascript:void(0)) [that determine how an application behaves under different cultures.](javascript:void(0)) [The first setting is exposed by the static current culture property of the thread class.](javascript:void(0)) [This property tells the run time how to format strings.](javascript:void(0)) [For instance, if we have a dot 2 (phonetic) string on a date time or a dot 2 on a currency,](javascript:void(0)) [then how should the date time be formatted and what currency symbol should be used?](javascript:void(0)) [Is it going to be a euro, a dollar, a rupee or a yen?](javascript:void(0)) [Another important setting is exposed from the static current UI culture property.](javascript:void(0)) [This setting impacts how the run time performs resource loading which we're about to look at.](javascript:void(0)) [You can set these properties in your own code.](javascript:void(0)) [For example, if you have one of those sites](javascript:void(0)) [where you let the user select what language they want to see and that's in a drop down list](javascript:void(0)) [or they click on a flag icon, that would be a scenario](javascript:void(0)) [where you would set these properties manually.](javascript:void(0)) [Or you can let ASP.net set these properties for you.](javascript:void(0)) [ASP.net will set the properties based](javascript:void(0)) [on the accept language HTTP header that the client's browser sends.](javascript:void(0)) [In order for ASP.net to do this automatically though,](javascript:void(0)) [you do have to add a globalization section to your web.doc config file and set the culture](javascript:void(0)) [and UI culture attributes to auto.](javascript:void(0)) [Let's see how these settings make our application behave.](javascript:void(0)) [( Silence )](javascript:void(0)) [Inside of the index view for the home controller,](javascript:void(0)) [let's do a little experiment with the globalization settings.](javascript:void(0)) [So first I'm going to add a code block](javascript:void(0)) [that defines two variables, amount which is a decimal.](javascript:void(0)) [That's what the m suffix is and sum date](javascript:void(0)) [which is a date time representing July 9th of the year 2002.](javascript:void(0)) [And now I will add some code to output these two variables by calling two string.](javascript:void(0)) [Amount will call two string and pass in a C parameter which means please format this](javascript:void(0)) [as a currency and with sum date, we'll just call two short date string.](javascript:void(0)) [And before we run the application, I will come into the web config and inside of system.web,](javascript:void(0)) [I will add that globalization section that we talked about.](javascript:void(0)) [We need to set culture equals auto and we need to set UI culture equals auto.](javascript:void(0)) [And now let's see how this behaves.](javascript:void(0)) [I will run the application with Internet Explorer and since my default culture](javascript:void(0)) [on this machine is United States English, I see the currency value is formatted](javascript:void(0)) [with a dollar sign and the date is formatted as month, day, year.](javascript:void(0)) [But now let me come into the Internet Explorer options.](javascript:void(0)) [So ALT T will open the tools menu.](javascript:void(0)) [Come into internet options.](javascript:void(0)) [Under appearance, go to languages and here I can set language preferences.](javascript:void(0)) [It's from here where I can add a language and in this list, let me select French](javascript:void(0)) [and not the Canadian French but the real Francois from France and let me take](javascript:void(0)) [that language and move it up in the list.](javascript:void(0)) [So this is now essentially my preferred language and I will close this out and refresh the page.](javascript:void(0)) [( Silence )](javascript:void(0)) [And now my currency is formatted with a comma and a euro sign](javascript:void(0)) [and the date format has also changed.](javascript:void(0)) [It is now day month year.](javascript:void(0)) [So what we've done so far is to see how globalization works](javascript:void(0)) [and this is just a starting point for localization.](javascript:void(0)) [We've changed how data amounts were formatted but we would also need to go](javascript:void(0)) [through the application and take care of string literals like search by name if we really want](javascript:void(0)) [to localize those to a specific culture](javascript:void(0)) [because all the literal texts in here are still in English.](javascript:void(0)) [Let's see how we can change that.](javascript:void(0))

* + [Resources](javascript:void(0))

[( Silence )](javascript:void(0)) [To localize the text, I went to display in this application.](javascript:void(0)) [I'm going to rely on resource files.](javascript:void(0)) [A resource file is an XML with a resx extension,](javascript:void(0)) [r e s x and it stores localized strings for a culture.](javascript:void(0)) [It can also contain binary assets like images](javascript:void(0)) [but we'll just be using text in this application.](javascript:void(0)) [To support different languages, I'll need at least one resource file](javascript:void(0)) [for each language that I explicitly support.](javascript:void(0)) [The first resource file I create will be the base resource file](javascript:void(0)) [and this might have a name like strings.resx.](javascript:void(0)) [If I want to produce a version of strings.resx that works with Spanish,](javascript:void(0)) [I'd also create a strings.es.resx.](javascript:void(0)) [So you can see the language the resx supports, it's embedded in the file name](javascript:void(0)) [and it follows a naming convention to describe to the dot.net runtime what language](javascript:void(0)) [and what culture a particularly resx file supports.](javascript:void(0)) [You can also be very specific about the culture.](javascript:void(0)) [So es dash mx would be Spanish Mexico.](javascript:void(0)) [In dot.net there's a resource manager that will load the appropriate resource file](javascript:void(0)) [for the current UI culture that's set but we generally do not have to interact](javascript:void(0)) [with this resource manager directly](javascript:void(0)) [because Visual Studio will automatically compile these resources into our project](javascript:void(0)) [and generate code to access the resources with a strongly typed class.](javascript:void(0)) [If you go to the properties window for a resx file,](javascript:void(0)) [you'll see Visual Studio has the build action set to embedded resource.](javascript:void(0)) [We'll talk about some of the other settings here when we get to the demonstration](javascript:void(0)) [but to use a resource is very simple.](javascript:void(0)) [In a view, I can get to a resource with a greeting property inside by going](javascript:void(0)) [to the generated class and asking for the greeting property.](javascript:void(0)) [This might return the string hello for English but it could also return hola](javascript:void(0)) [if the current UI culture is set to Spanish and a Spanish resource exists.](javascript:void(0)) [If a resource doesn't exist for the current UI settings,](javascript:void(0)) [the resource manager will fall back to the base resource.](javascript:void(0)) [Let's go into Visual Studio and see how all of this works.](javascript:void(0)) [Inside the index view for the home controller, let me add a greeting to this view just](javascript:void(0)) [above the other culturally sensitive information that we're displaying.](javascript:void(0)) [And now what I want to do is localize this so it's not a hard coded string in English.](javascript:void(0)) [We could display English or French or any other language that we create a resource file for.](javascript:void(0)) [So the first step is to create that resource file.](javascript:void(0)) [And one of the decisions you'll have to make when creating a resource file is where to create](javascript:void(0)) [that because you can have multiple resource files and these resource files could live](javascript:void(0)) [in the web project, in this ode to food (phonetic) project.](javascript:void(0)) [You could also put them in a separate class library if that was more maintainable for you](javascript:void(0)) [and you also need to select what folder these resource files are going to go into.](javascript:void(0)) [To keep things simple, I think I might need a resource file just](javascript:void(0)) [for the views for the home controller.](javascript:void(0)) [And so what I might do in that situation is come into the home views folder and add a new item.](javascript:void(0)) [I'll search for res and that should bring up resources file.](javascript:void(0)) [I will call this just resources.resx and this will be my base resource file.](javascript:void(0)) [It opens up in a Visual Studio resx file editor.](javascript:void(0)) [One of the changes that I have to make sure I do inside](javascript:void(0)) [of here is to change this access modifier.](javascript:void(0)) [It's currently internal.](javascript:void(0)) [I'm going to change that over to be public.](javascript:void(0)) [This is just required because of the compilation model.](javascript:void(0)) [It turns out that razor views are compiled into a different assembly, a different dll file](javascript:void(0)) [than our ode to food assembly and without having these resource files be generated](javascript:void(0)) [as public classes, we wouldn't be able to see them from our razor views but just changing](javascript:void(0)) [that access modifier will set that for you.](javascript:void(0)) [Behind the scenes what's happening is there is a class being generated for you.](javascript:void(0)) [You can see it if you expand a resources.resx, there's a generated .cs file behind here](javascript:void(0)) [and we'll take a look at that in just a second.](javascript:void(0)) [First, let me come in and add a name.](javascript:void(0)) [So this is how I want to reach a specific resource.](javascript:void(0)) [I want to have the greeting resource and this is where I would give it a value like hello.](javascript:void(0)) [And now if I save this resources file and open up the generated code,](javascript:void(0)) [you'll see there's a class that's generated in the odetofood.views.home name space.](javascript:void(0)) [It's called resources and all the way down here at the bottom --](javascript:void(0)) [( Silence )](javascript:void(0)) [-- there will be a greeting property that I can get to and all the other code that is inside](javascript:void(0)) [of here is just code that is required](javascript:void(0)) [to have the .net resource manager load the appropriate resource file.](javascript:void(0)) [And if I wanted to use this code from an index view,](javascript:void(0)) [all I need to do now is replace my hard coded text with the name spaces, the class name,](javascript:void(0)) [and the name of the resource that I want to display.](javascript:void(0)) [In this case, greeting and at this point,](javascript:void(0)) [if anyone comes to the site specifying any language, we're always going to display hello](javascript:void(0)) [because this our base resource file.](javascript:void(0)) [We don't have a resource file specified for any other languages so everyone gets English.](javascript:void(0)) [If I wanted to customize this for say, French, that's when I could come](javascript:void(0)) [in and add a second resource file.](javascript:void(0)) [So let's add a new item --](javascript:void(0)) [( Silence )](javascript:void(0)) [-- with the same name, resources.fr.resx and inside](javascript:void(0)) [of here I would duplicate the name value pairs.](javascript:void(0)) [You can obviously have as many name value pairs as you want but we just have one.](javascript:void(0)) [The name is always greeting but this would be the French translation, bonjour.](javascript:void(0)) [Notice there is no code generation for this resource file.](javascript:void(0)) [Code generation only needs to happen on the base resource file, resources.resx.](javascript:void(0)) [All of these files get embedded into my assembly by Visual Studio when it builds.](javascript:void(0)) [That's why the build action is set to embedded resource.](javascript:void(0)) [So you can think of all this information getting compiled right alongside with your C sharp code](javascript:void(0)) [that goes with your application wherever you copy the dll file to.](javascript:void(0)) [Well, let's run this and see how it behaves.](javascript:void(0)) [( Silence )](javascript:void(0)) [On the home page of the application, I'm still configured to prefer the French language.](javascript:void(0)) [So I see bonjour.](javascript:void(0)) [I see Euros.](javascript:void(0)) [Now let me go into tools, internet options, languages, set language preferences,](javascript:void(0)) [and move English back up to the top.](javascript:void(0)) [Then close all these windows and refresh Internet Explorer](javascript:void(0)) [and refresh Internet Explorer and nothing happens.](javascript:void(0)) [I'm still seeing bonjour.](javascript:void(0)) [Is this a problem with the way I created my resource files?](javascript:void(0)) [Is this a problem with Internet Explorer?](javascript:void(0)) [Well no, none of the above.](javascript:void(0)) [It turns out, once again that we have to be very careful with something that we added earlier](javascript:void(0)) [which is the output cache attribute.](javascript:void(0)) [In this case, what is probably happening now is I'm making this request](javascript:void(0)) [but all the MVC run time caching cares about is that all the parameters are the same,](javascript:void(0)) [the x requested with hasn't changed.](javascript:void(0)) [It's going to return a cached response that was built for the French language.](javascript:void(0)) [I need to tell ASP.net that there's another header that it's need to vary the cache with](javascript:void(0)) [and that is the accept language header.](javascript:void(0)) [And we actually don't separate these with commas.](javascript:void(0)) [We separate these with semicolons.](javascript:void(0)) [And now if I build the application, let's try this again.](javascript:void(0)) [I'll refresh this page and now we're displaying hello.](javascript:void(0)) [That seems good.](javascript:void(0)) [And let me go into tools, internet options, languages, set language preferences,](javascript:void(0)) [move French back to the top, close one window.](javascript:void(0)) [Close two windows.](javascript:void(0)) [Close three windows and refresh and now we display bonjour.](javascript:void(0)) [So caching isn't interfering with our localization anymore.](javascript:void(0)) [And while we're here at the home controller there's something else you should know](javascript:void(0)) [about resource files.](javascript:void(0)) [There's really nothing that restricts you to using them only from a view.](javascript:void(0)) [So I could have a greeting string defined here that is set](javascript:void(0)) [to odetofood.views.home.resources.greeting and that would work just as well.](javascript:void(0)) [The correct string will be loaded on the UI culture setting and in fact,](javascript:void(0)) [you can also use these resources if you want](javascript:void(0)) [to build custom error messages for your data annotations.](javascript:void(0)) [So let's look at the restaurant review model that we built and we had things like required.](javascript:void(0)) [And if the user doesn't fill something out here,](javascript:void(0)) [they'll get an error message like the body field is required.](javascript:void(0)) [Let's say you wanted to customize that and also customize it for different languages.](javascript:void(0)) [In that case, you could come in and you wouldn't use error message.](javascript:void(0)) [Error message would just be hard coded text on what you wanted the error message to be.](javascript:void(0)) [We want this to be localized so I would specify error message resource type and I would set this](javascript:void(0)) [to the type of my resource class, the generated class that was given to me](javascript:void(0)) [and I would also specify the error message resource name.](javascript:void(0)) [That is what to look up inside of that resource file.](javascript:void(0)) [This is where I would provide one of the names that's inside of my resource file.](javascript:void(0)) [Greeting, obviously, wouldn't make any sense as an error message but it's the only one I have](javascript:void(0)) [in there now so I'm adding it here.](javascript:void(0)) [You'd obviously have to create some error messages.](javascript:void(0)) [Maybe they would be in a different resource file somewhere in your project](javascript:void(0)) [but you could just specify the type of the resource and the name to pull out of](javascript:void(0)) [that resource file and then you can have localized error messages too.](javascript:void(0)) [( Silence )](javascript:void(0))

* + [Diagnostics](javascript:void(0))

[Our last infrastructure topic is diagnostics.](javascript:void(0)) [Diagnostics help us answer the question just what is our application doing?](javascript:void(0)) [When did it start?](javascript:void(0)) [When it restart?](javascript:void(0)) [Are there any unhandled exceptions?](javascript:void(0)) [Are there any authorization failures?](javascript:void(0)) [To get this type of information, there are a number of options available.](javascript:void(0)) [First, ASP.net does have a health monitoring system you can configure to report on everything](javascript:void(0)) [from application start to application end and all the events in between,](javascript:void(0)) [exceptions, failures, compilation events.](javascript:void(0)) [There are also many third party options for logging and monitoring.](javascript:void(0)) [Logfornet is a popular open source library you can reference](javascript:void(0)) [to build your own logging infrastructure.](javascript:void(0)) [And the patterns and practices team at Microsoft also has an application logging block.](javascript:void(0)) [One really popular open source library is ELMAH, e l m a h. That stands](javascript:void(0)) [for Error Logging Handles and Modules.](javascript:void(0)) [ELMAH can capture events about your application and store them in a variety of destinations,](javascript:void(0)) [xml files, text files, databases.](javascript:void(0)) [It could even post them to Twitter and it has some prebuilt UI](javascript:void(0)) [to show you the events it has recorded.](javascript:void(0)) [Let's take a look.](javascript:void(0))

* + [Health Monitoring and ELMAH](javascript:void(0))

[( Silence )](javascript:void(0)) [The default configuration for ASP.net health monitoring lives in a web.config file](javascript:void(0)) [but it doesn't live in the web.config file that's in the root of your project.](javascript:void(0)) [Instead it lives in a machine level web.config file that applies configuration](javascript:void(0)) [to every ASP.net application that runs on this computer.](javascript:void(0)) [You can find that if you go into your Windows directory.](javascript:void(0)) [Go to Microsfoft.net.](javascript:void(0)) [Go to framework and then pick the right version number.](javascript:void(0)) [We're using Visual Studio 2012 with .net 4 point 5.](javascript:void(0)) [So of course we go to the version 4 directory](javascript:void(0)) [because version 4 point 5 was an in place update to version 4.](javascript:void(0)) [Dot.net version numbers have always been confusing](javascript:void(0)) [and dot.net 4 point 5 continues that historical tradition.](javascript:void(0)) [We go into that directory.](javascript:void(0)) [There'll be a config directory.](javascript:void(0)) [It's inside of here where you will find a machine level web.config](javascript:void(0)) [that applies configuration everywhere.](javascript:void(0)) [I can drag that into Visual Studio and we'll do a search,](javascript:void(0)) [control I in here to find health monitoring.](javascript:void(0)) [And the two most important sections in here are -- well actually there's 3 important sections.](javascript:void(0)) [One is providers.](javascript:void(0)) [Providers include destinations for where you can publish events.](javascript:void(0)) [You might want to save them in the event log.](javascript:void(0)) [In that case, you'd use the event log provider or maybe you want to store them in sequel server](javascript:void(0)) [or maybe you want to use WOMI (phonetic) to publish these events.](javascript:void(0)) [So those are the built-in providers.](javascript:void(0)) [You can always add more providers if you write one.](javascript:void(0)) [Another important section is event mappings.](javascript:void(0)) [This categorizes all the possible events into buckets with friendly names.](javascript:void(0)) [So we have all events which is every event that could happen in the health monitoring system,](javascript:void(0)) [lifetime events, request processing events, and all errors.](javascript:void(0)) [It is the rules section that will define where to send something like all errors.](javascript:void(0)) [What provider do we send these to?](javascript:void(0)) [So by default because this is in this machine level web config,](javascript:void(0)) [we will send all errors to the event log.](javascript:void(0)) [Any unhandled exception that occurs](javascript:void(0)) [in the ASP.net application should appear in the Windows event log.](javascript:void(0)) [Also, any failure audits should appear in the event log.](javascript:void(0)) [You can add additional rules.](javascript:void(0)) [The best thing to do would be to take this health monitoring section and mimic](javascript:void(0)) [in your own web.config file if you want to reconfigure any of these rules](javascript:void(0)) [but the one thing you need to be careful about is not to do something](javascript:void(0)) [like send all request processing events to the Windows error log.](javascript:void(0)) [It's not designed to handle a high volume of events being pumped into it.](javascript:void(0)) [That would be something that you could do with sequel server though.](javascript:void(0)) [But because we have these rules in effect, all errors go to the event log,](javascript:void(0)) [that means if I have an unhandled exception, I should be able to see](javascript:void(0)) [that in the Windows event log and in fact, we still do have a controller,](javascript:void(0)) [the cuisine controller, that we wrote a long time ago that has an unhandled exception.](javascript:void(0)) [It intentionally has an unhandled exception.](javascript:void(0)) [So let's run the application and go there once.](javascript:void(0)) [And now something terrible has happened.](javascript:void(0)) [Let's see if I can find this in the Windows event log by going](javascript:void(0)) [to the gigantic start menu and typing --](javascript:void(0)) [( Silence )](javascript:void(0)) [-- event and under settings, there's view event logs.](javascript:void(0)) [( Silence )](javascript:void(0)) [ASP.net errors would be recorded under Windows logs, under application and here](javascript:void(0)) [at the very top we can see there's been an ASP.net problem.](javascript:void(0)) [It's listed as a warning but it's inside of here where you will find the event message which is](javascript:void(0)) [that there was an unhandled exception.](javascript:void(0)) [You can find the path to the application where this occurred, the machine name.](javascript:void(0)) [You can find out who the web server was running as.](javascript:void(0)) [Since we're using IS Express, it's running under my account and there's a full stack trace](javascript:void(0)) [that we could use to try to track down exactly where this error came from.](javascript:void(0)) [So all of that information's recorded in the event log for you automatically](javascript:void(0)) [but sometimes the event log can be a little cumbersome to get to.](javascript:void(0)) [Sometimes you want to store things in a way that you can get to them through the web interface](javascript:void(0)) [or perhaps, you want to be able to email errors out to the people](javascript:void(0)) [and that's when you can turn to ELMAH.](javascript:void(0)) [Remember this stands for Error Logging Modules and Handlers.](javascript:void(0)) [The easy way to install it is through newget (phonetic).](javascript:void(0)) [We'll just do a search for ELMAH and the package I want to install is ELMAH.mvc](javascript:void(0)) [because this will allow me to reach the ELMAH error logs](javascript:void(0)) [in a nice MVC friendly way just by going to slash ELMAH.](javascript:void(0)) [And now if I close this, it's been added to the project.](javascript:void(0)) [I'm just going to do a build to make sure all the right files get copied to the right places.](javascript:void(0)) [Otherwise, there's no code changes I need to make.](javascript:void(0)) [We'll go back to the cuisine controller.](javascript:void(0)) [I have a few exceptions pumped in here by pounding on the refresh key](javascript:void(0)) [and now let me go to slash ELMAH.](javascript:void(0)) [And I can see all the exceptions that it just recorded.](javascript:void(0)) [You can drill into the details and see the stack trace.](javascript:void(0)) [You can also configure ELMAH to send these via email or store them in different places.](javascript:void(0)) [All that configuration can take place in your web.config file.](javascript:void(0)) [In fact, ELMAH did add some configuration to my web.config file](javascript:void(0)) [and it added some of these app settings.](javascript:void(0)) [For instance, who is allowed to see the event log?](javascript:void(0)) [Since we now have rules in the application, I could say well only admins.](javascript:void(0)) [You can also configure the route that you use to get to see those error messages.](javascript:void(0)) [So by default it's ELMAH and just go to slash ELMAH to see the errors, that's very simple.](javascript:void(0)) [And one more thing, since we are going to require rules here.](javascript:void(0)) [I'm going to explicitly say this requires authentication before you can view it.](javascript:void(0)) [You can read more details about ELMAH and ELMAH configuration on the ELMAH website.](javascript:void(0)) [( Silence )](javascript:void(0))

* + [Summary](javascript:void(0))

[In this module, we spent some time with the ASP.net output cache attribute.](javascript:void(0)) [We saw how to use the attribute and vary by location and vary by header](javascript:void(0)) [to hopefully increase the performance of the application.](javascript:void(0)) [I say hopefully because I never took any performance measurements before](javascript:void(0)) [and after caching and the only way to know for sure that you've helped is](javascript:void(0)) [to start taking hard measurements.](javascript:void(0)) [So I encourage you to always take measurements before you start applying cache attributes.](javascript:void(0)) [We also looked at the localization features of ASP.net and saw how we can let resource files](javascript:void(0)) [and ASP.net take care of most of the hard work.](javascript:void(0)) [We saw, for example, how the globalization settings](javascript:void(0)) [in web.config can automatically give us different formatting behaviors](javascript:void(0)) [for different languages depending on the client's accept language header.](javascript:void(0)) [Finally, we took a look at diagnostics and we now have ELMAH added to the application](javascript:void(0)) [and watching for errors and giving us an easy way to see the unhandled exceptions that occur.](javascript:void(0)) [This was just a quick look at all the functionality that ELMAH can provide.](javascript:void(0)) [( Silence )](javascript:void(0))

* Unit Testing with ASP.NET MVC 4
  + [Introduction](javascript:void(0))

[Hi, this is Scott Allen and this module looks at test-driven development](javascript:void(0)) [and unit testing with the "ASP.NET MVC Framework".](javascript:void(0)) [In this module, I want to show you how to use test-driven development](javascript:void(0)) [to drive the design of a feature.](javascript:void(0)) [We'll look at the TDD cycle, which is red, green, re-factor.](javascript:void(0)) [First, you write a failing test which gives you a red result in the "Test Runner".](javascript:void(0)) [Then you make the test pass, which gives you a green result.](javascript:void(0)) [And then you re-factor the code to improve the design.](javascript:void(0)) [I'm also going to demonstrate how to unit test controllers, and how to build test doubles](javascript:void(0)) [to control the environment the controller executes inside of when we test them.](javascript:void(0)) [( Silence )](javascript:void(0))

* + [Test Driven Development](javascript:void(0))

[To give you some idea of the test-driven development flow, we're going to step away](javascript:void(0)) [from the web project for a bit and focus on the test project.](javascript:void(0)) [If you remember, in the first module of this course when we created the application,](javascript:void(0)) [we also created a unit testing project.](javascript:void(0)) [We can run tests inside of this project using the "Test Menu"](javascript:void(0)) [or by using the "Control-R A" shortcut key.](javascript:void(0)) ["Visual Studio" will find all of the test classes inside this project](javascript:void(0)) [and run the test methods.](javascript:void(0)) [We currently have a failing test, and if you look at the problem here by clicking](javascript:void(0)) [on the test, and down here at the bottom of the test explorer,](javascript:void(0)) [it will show you the exception that happened.](javascript:void(0)) [You'll see there was an exception because the "Home Controller" tries to talk to the database,](javascript:void(0)) [but it doesn't find the connection string that it's looking for,](javascript:void(0)) [and this test is envoking the "Home Controller Index Action".](javascript:void(0)) [We have that connection string defined in the web application,](javascript:void(0)) [so the web application works just fine when we view the app with a browser,](javascript:void(0)) [but the tests are failing because we haven't configured this properly yet.](javascript:void(0)) [We're going to defer this problem and come back later.](javascript:void(0)) [And I'll show you a couple approaches you can use for testing controllers that need data.](javascript:void(0)) [For right now, I just want to focus on what it would look like to try test-driven development.](javascript:void(0)) [I encourage you to try test-driven development, and write unit tests](javascript:void(0)) [because I personally have found them](javascript:void(0)) [to be the most beneficial practice I have ever followed in my career.](javascript:void(0)) [And the one secret to this is that tests are just as much about design](javascript:void(0)) [as they are about insuring quality.](javascript:void(0)) [Let me show you what I mean.](javascript:void(0)) [To get ready to implement this feature,](javascript:void(0)) [I've already added a "Features" folder into my "Unit Tests" project.](javascript:void(0)) [All I need to do is right-click this and say "Add a Unit Test".](javascript:void(0)) [It's called "Unit Test One.cs".](javascript:void(0)) [We can change the name later.](javascript:void(0)) [For now, I'm going to paste in some comments that I jotted down as I was talking](javascript:void(0)) [to the business people about this feature.](javascript:void(0)) [It turns out that they want to try at least two different approaches](javascript:void(0)) [and see which one works the best.](javascript:void(0)) [One approach to calculating the overall rating for a restaurant would be](javascript:void(0)) [to just take the simple average of the rating property for the last N number of reviews](javascript:void(0)) [where N is something an administrator can configure,](javascript:void(0)) [so it might be the last 10 reviews or the last 100 reviews.](javascript:void(0)) [Another approach would be to compute a weighted mean](javascript:void(0)) [where the most recent reviews are more heavily weighted.](javascript:void(0)) [That would benefit a restaurant that is improving](javascript:void(0)) [after a bad start and getting better reviews.](javascript:void(0)) [It might also benefit a customer who is thinking about going to a restaurant that used](javascript:void(0)) [to have great quality, but is now on the decline.](javascript:void(0)) [I like starting with some comments inside of my test file,](javascript:void(0)) [just so I can refer to them and think about them.](javascript:void(0)) [They usually describe the business scenario that I'm trying to solve.](javascript:void(0)) [I don't use the TDD approach for all of the code I write, but this represents the type](javascript:void(0)) [of scenario where I probably would use TDD.](javascript:void(0)) [I'm not sure how to design this feature, and test-driven development is a great design tool.](javascript:void(0)) [I can delete the comments when I'm done, but for right now it provides me something concrete](javascript:void(0)) [that I can look at and remember the goal, and I can start thinking about a solution and thinking](javascript:void(0)) [about how I'd implement this feature.](javascript:void(0)) ["Visual Studio" has already given me a test class.](javascript:void(0)) [It has a method inside of it, "Test Method One".](javascript:void(0)) [The first thing you might try to do is give this class and method a more descriptive name,](javascript:void(0)) [but don't get too caught up on naming right away.](javascript:void(0)) [My idea is that I want to use "Visual Studio" like I used](javascript:void(0)) [to use a whiteboard before I started test-driven development.](javascript:void(0)) [Someone would give me a software requirement, and I'd go to the whiteboard and start trying](javascript:void(0)) [to diagram out some classes and methods that I think I would need.](javascript:void(0)) [I'm going to do that in "Visual Studio" instead because "Visual Studio" gives me a compiler,](javascript:void(0)) [and I can use tests to validate that the code works.](javascript:void(0)) [And since I'm not sure what direction I'm going just yet,](javascript:void(0)) [there's no need to spend time thinking of names.](javascript:void(0)) [Once I've figured that out, we can rename things later.](javascript:void(0)) [I do know I'm going to be working with some data, some restaurant data,](javascript:void(0)) [and we already have that restaurant class defined.](javascript:void(0)) [I just need to pull in the name space "Ode to Food.models",](javascript:void(0)) [and my restaurant will have some reviews associated with it, and I need to bring](javascript:void(0)) [in the name space for "List of T", which is "system.collections.generic".](javascript:void(0)) [And once I have that, I'm ready to add some reviews to this restaurant.](javascript:void(0)) [I'll just add a single review, and I think the only thing that we're going to care](javascript:void(0)) [about for the purposes of these tests is that a review has a rating.](javascript:void(0)) [In this case, we'll just have a single review, keep things very simple; the rating is a four.](javascript:void(0)) [That's the data I want to evaluate.](javascript:void(0)) [And now I can start thinking about the API that I want to use to compute the restaurant rating.](javascript:void(0)) [This would be the point where I could experiment with different class names and method names](javascript:void(0)) [until I found something that looks like it might work.](javascript:void(0)) [And let's say I reach the point where I've decided](javascript:void(0)) [that I think a restaurant rater might work,](javascript:void(0)) [so I will create a restaurant rater and pass the data into it.](javascript:void(0)) [And, of course, this code doesn't compile, but I don't expect it to yet.](javascript:void(0)) [Remember, this is just a whiteboard.](javascript:void(0)) [In fact, one of the first steps that you'll take many times](javascript:void(0)) [with test-driven development is writing some code that doesn't even compile yet.](javascript:void(0)) [The goal is to go and do the simplest possible thing to get it](javascript:void(0)) [to compile once you're relatively happy with the names and the design.](javascript:void(0)) [Next, I might say that this rater -- I can tell it to compute a rating given some number](javascript:void(0)) [that it will use to determine how many reviews to use, so let's say 10 reviews,](javascript:void(0)) [and it will give me back some sort of result.](javascript:void(0)) [Again, this is the time where I am exploring how I want to invoke these API's,](javascript:void(0)) [and what these API's should look like.](javascript:void(0)) [What parameters do I want to pass?](javascript:void(0)) [What do I want to name things?](javascript:void(0)) [And forging ahead with what I have so far, I'm thinking that if I tell this](javascript:void(0)) [to compute the result for a restaurant with a single review,](javascript:void(0)) [and we're doing just the simple averaging for right now, I should be able to write an assert](javascript:void(0)) [at this point, and assert that four would be the "result.rating".](javascript:void(0)) [At this point, if I tried to do a build or run the test, it's going to fail miserably](javascript:void(0)) [because we don't compile, but I can use "Visual Studio" to help me create the classes.](javascript:void(0)) [Just put the cursor on "Restaurant Rater", open the drop-down list with "Control-Period",](javascript:void(0)) [and let's generate a class from "Restaurant Rater"](javascript:void(0)) [that will create a new ".cs" file in this project.](javascript:void(0)) [Ultimately, it's going to need to be in the "Ode to Food" project because it's going](javascript:void(0)) [to be business logic that we need to exercise from the web application,](javascript:void(0)) [but leaving it in here is fine for right now.](javascript:void(0)) [I haven't solidified anything yet; things might change.](javascript:void(0)) [At some point, the design will start to firm up and I can move these ".cs" files](javascript:void(0)) [into the right project, and then adjust the name spaces accordingly.](javascript:void(0)) [My next compiler error is that the "Restaurant Rater" needs a constructor](javascript:void(0)) [that takes some restaurant information.](javascript:void(0)) [Again, I can generate this from "Visual Studio", just "Control-Period",](javascript:void(0)) [and select the entry "Generate Constructor Stub".](javascript:void(0)) [That should be there now.](javascript:void(0)) [It's going to need a method called "Compute Rating".](javascript:void(0)) [I will also generate that.](javascript:void(0)) [And let's look at what we've developed so far in side of "Restaurant Rater".](javascript:void(0)) [We have a "Public Constructor" that takes a restaurant that seems good.](javascript:void(0)) [Let's get rid of the spurious name space here by deleting that and bringing in a using statement](javascript:void(0)) [for "Ode to Food.models", and then I can delete that here also,](javascript:void(0)) [just trying to clean up the code already.](javascript:void(0)) [I should really wait until I have a passing test, and then come in and do changes like this.](javascript:void(0)) [That would be the re-factoring step.](javascript:void(0)) [This "Compute Rating" method, I believe, will ultimately return some sort of rating result.](javascript:void(0)) [Let's create that, generate a class for "Rating Result".](javascript:void(0)) [So it's not going to return "Object"; it's going to return a rating result,](javascript:void(0)) [and we don't need this exception here.](javascript:void(0)) [What I could do, the simplest possible thing to make this work,](javascript:void(0)) [would be to just return a new rating result, not even do any calculations just yet.](javascript:void(0)) [And are we ready to build?](javascript:void(0)) [Not quite yet.](javascript:void(0)) [It looks like my "Rating Result" needs a rating property,](javascript:void(0)) [and my "Compute Rating" method needs to be public.](javascript:void(0)) [So let's come in and change this.](javascript:void(0)) [This is a public method, and "Rating Result" will ultimately be a public class,](javascript:void(0)) [and it's going to have a "Property of Type Integer" that is the rating.](javascript:void(0)) [I know this is an integer because the business specifically asked me](javascript:void(0)) [to make ratings an integer value so we don't have ratings like seven point six six six](javascript:void(0)) [that scare users; that would just be a seven or an eight.](javascript:void(0)) [And at this point, I think, I can do a build.](javascript:void(0)) [Build succeeded.](javascript:void(0)) [I'll run all the tests with "Control-R A", and of course, our new test method fails.](javascript:void(0)) [We told it that it should be expecting a four result; the actual result it got back](javascript:void(0)) [from "Result.rating" was a zero because we haven't done any calculations.](javascript:void(0)) [And this is the first step in TDD.](javascript:void(0)) [I've designed some classes, designed some methods.](javascript:void(0)) [I now have a failing test.](javascript:void(0)) [My next goal is to do the simplest possible thing that will make that test pass,](javascript:void(0)) [and the absolute simplest possible thing would be to say --](javascript:void(0)) [let's say that the result equals a new rating result, "'Result.rating' equals four".](javascript:void(0)) ["Return Result", run the tests, now "Test Method One" passes.](javascript:void(0)) [And you might look at that and say, "It's completely ridiculous.](javascript:void(0)) [You're hard-coding computations just to make the test pass."](javascript:void(0)) [However, stick with me on this.](javascript:void(0)) [One of the tenets of doing test-driven development is to do the simplest possible thing](javascript:void(0)) [to make a test pass, and then you just keep adding tests that will test more conditions.](javascript:void(0)) [You're going to have to come in and change this "Compute Rating" code](javascript:void(0)) [so it actually behaves correctly.](javascript:void(0)) [But along the way, as you're making these changes, you're going to be writing tests](javascript:void(0)) [that validate scenarios that are going to prove very valuable later on.](javascript:void(0)) [They're going to make sure that you're making the right changes, that as you're adding things](javascript:void(0)) [and adding features and reconstructing the code,](javascript:void(0)) [that you're not breaking anything that used to work.](javascript:void(0)) [That's one of the real values of having tests.](javascript:void(0)) [And once you have a passing test, one of the first things you'll want to do is come in](javascript:void(0)) [and re-factor, clean code up, make sure things are named properly.](javascript:void(0)) [I don't really like the "To-Do" statement here.](javascript:void(0)) [In fact, I don't really like having a data field in here when this is really a restaurant.](javascript:void(0)) [And I can use "Visual Studio", "Control-Period", to say,](javascript:void(0)) ["Rename data to restaurant everywhere," And that fixes things up for me.](javascript:void(0)) [I also want this to be called "Restaurant".](javascript:void(0)) [Same trick here, "Control-Period", it will rename the local use](javascript:void(0)) [of that restaurant variable, also, this "P parameter", no idea what that means,](javascript:void(0)) [so let's just change it right away to "Number of Reviews".](javascript:void(0)) [That's what it's supposed to be, ultimately.](javascript:void(0)) [And now I'll hit "Control-R", "Control-A" again.](javascript:void(0)) [Test method is still passing.](javascript:void(0)) [My changes haven't broken anything.](javascript:void(0)) [Those are very simple changes, but you can imagine, as things build up in complexity,](javascript:void(0)) [you change things in one place it breaks something somewhere else.](javascript:void(0)) [That's a sign that the tests are helping you, and also that you might have a design issue.](javascript:void(0)) [Maybe something is too coupled to another class.](javascript:void(0)) [But now that we've seen a basic test in action, I'm going to accelerate things a bit,](javascript:void(0)) [and work through the rest of the design and implementation of this feature.](javascript:void(0)) [( Silence )](javascript:void(0))

* + [Test Driven Design](javascript:void(0))

[I've added a second test to the "Test Fixture".](javascript:void(0)) [This one passes in two "Restaurant Reviews" to the "Restaurant Rater".](javascript:void(0)) [Since the code is starting to take some form, I've also given the tests some names,](javascript:void(0)) ["Computes Result for One Review", and "Computes Results for Two Reviews".](javascript:void(0)) [I've also changed the implementation inside of the "Restaurant Rater".](javascript:void(0)) [The simplest possible thing I've found to work is a very simple link statement.](javascript:void(0)) [It averages the reviews.](javascript:void(0)) [The good news is both of the tests now pass and I'm in a re-factoring stage.](javascript:void(0)) [I think the "Restaurant Rater" is in good shape so far.](javascript:void(0)) [It's actually the tests that are bothering me.](javascript:void(0)) [One thing you'll learn about unit testing in general is that the tests are just](javascript:void(0)) [as important as the application code.](javascript:void(0)) [And it bothers me that I have so much setup work inside of each test to make a restaurant](javascript:void(0)) [and make the reviews for that restaurant, so in this re-factoring phase,](javascript:void(0)) [I'm going to re-factor the tests to make the setup a little bit easier.](javascript:void(0)) [( Silence )](javascript:void(0)) [What I've done to re-factor the tests is to add a method, a private method,](javascript:void(0)) [named "Build Restaurants and Reviews".](javascript:void(0)) [This is not a test method that the "Test Runner" will invoke to see if something passes or fails.](javascript:void(0)) [It doesn't have a "Test Method Attribute".](javascript:void(0)) [This is a helper method I can use for my other tests, to keep them simple.](javascript:void(0)) [When you look at a test, it's a little easier to see that "Arrange, Act,](javascript:void(0)) [Assert" pattern that we talked about in the introduction to this course.](javascript:void(0)) [This new method takes a variable number of integer parameters, and uses some link operators](javascript:void(0)) [to transform the integers into a list of reviews.](javascript:void(0)) [Those reviews are attached to the restaurant.](javascript:void(0)) [We just return the "Restaurant" then.](javascript:void(0)) [I can use this method in my tests by either creating an array of integers and passing them](javascript:void(0)) [into this method, as we did here with the values four and eight, or I can use the params ability](javascript:void(0)) [of "C Sharp" to just pass them in one by one.](javascript:void(0)) [And now I'm feeling a bit better about these tests.](javascript:void(0)) [They look a little bit cleaner, and both the tests are still passing, which is good.](javascript:void(0)) [For the next step, I might want to start to test boundary conditions.](javascript:void(0)) [What happens when we rate a restaurant with zero reviews,](javascript:void(0)) [or if we rate a restaurant that has a negative review?](javascript:void(0)) [Should we throw exceptions?](javascript:void(0)) [How do we deal with odd numbers?](javascript:void(0)) [Those would all be good tests to write, but I'm sure I can come up with an implementation](javascript:void(0)) [to satisfy all those odd scenarios.](javascript:void(0)) [They don't feel like they'd be high risk or difficult to implement.](javascript:void(0)) [And at this early point, I'm really trying to focus on design issues first because, remember,](javascript:void(0)) [TDD is primarily about design, and if I look at these requirements again,](javascript:void(0)) [they say we have to support at least two different types of averaging.](javascript:void(0)) [There's the simple average and the weighted average, and the business people also hinted](javascript:void(0)) [that there might be more in the future.](javascript:void(0)) [They want to try, and experiment, and find out what will work the best.](javascript:void(0)) [I want to see what happens if I introduce the second form of averaging,](javascript:void(0)) [and see how it impacts my design.](javascript:void(0)) [( Silence )](javascript:void(0)) [I have now added a new test, the "Test the Weighted Average of Two Reviews".](javascript:void(0)) [Remember, the requirements also said I need to weight the most recent reviews twice](javascript:void(0)) [as heavily as the oldest reviews.](javascript:void(0)) [And we haven't worked with review dates.](javascript:void(0)) [We haven't sorted anything yet, but in the future, again, I know I can write some tests](javascript:void(0)) [to make sure that the reviews are properly sorted before I compute the rating.](javascript:void(0)) [I can do that later.](javascript:void(0)) [Sorting is another implementation detail I know I can figure out,](javascript:void(0)) [and I'm not finished writing tests yet but I am still focused on design.](javascript:void(0)) [In this weighting test, the rating for a restaurant with review ratings of three](javascript:void(0)) [and nine is five because the value three is more heavily weighted than the value nine.](javascript:void(0)) [It comes before the value nine in order.](javascript:void(0)) [I've put together a quick and dirty implementation that will pass the test, and,](javascript:void(0)) [of course, I did this after I watched the test fail.](javascript:void(0)) [And this is what the implementation looks like.](javascript:void(0)) [I'm not in the re-factoring phase again.](javascript:void(0)) [I just want to show you some of the things](javascript:void(0)) [that tests will make you think about, for instance, rounding.](javascript:void(0)) [Right now, I'm truncating the average rating,](javascript:void(0)) [so the link statement computes the average rating and casts the result to an int.](javascript:void(0)) [The weighted average also performs division](javascript:void(0)) [between two integer values, which will result in truncation.](javascript:void(0)) [So the question is, is truncating the right behavior?](javascript:void(0)) [This is one of those scenarios that tests will make you think about, and many times you'll need](javascript:void(0)) [to go back to the business people and ask for clarification.](javascript:void(0)) [All I know so far is that they told me that the rating should be a whole number like six](javascript:void(0)) [or seven so I need to clarify the exact behavior.](javascript:void(0)) [Do I need to round?](javascript:void(0)) [Do I need to truncate?](javascript:void(0)) [Once I have an answer, I can write the tests that make sure that the rounding](javascript:void(0)) [or the truncating happens correctly.](javascript:void(0)) [But for right now, I still want to think about high level design issues,](javascript:void(0)) [and come back to handling simple averages versus weighted averages.](javascript:void(0)) [One thing that bothers me about this "Restaurant Rater" right now is how it is responsible](javascript:void(0)) [for computing a simple rating, which I would probably rename if it was going](javascript:void(0)) [to stay here, and computing a weighted rate.](javascript:void(0)) [But I know, based on my notes, that this is an area where I can anticipate change.](javascript:void(0)) [The business is going to want to add new algorithms, and change algorithms all the time](javascript:void(0)) [in search for something that works the best.](javascript:void(0)) [If we had to go into the "Restaurant Rater" and add a new method every time the business came](javascript:void(0)) [up with a new algorithm, I think that would make things difficult to change.](javascript:void(0)) [I want to make things easy to change, make them as easy as possible.](javascript:void(0)) [Let me do a bit more re-factoring, and I'll see if I can make it easy](javascript:void(0)) [to change algorithms while still having my tests pass.](javascript:void(0)) [It's wonderful having these passing tests now because I can rip code apart,](javascript:void(0)) [rebuild everything, and my tests will tell me if I did anything wrong.](javascript:void(0)) [( Silence )](javascript:void(0)) [I've done a bit more re-factoring, and I've decided that the](javascript:void(0)) ["Restaurant Rater" should not be responsible for computing the actual result.](javascript:void(0)) [You'll notice the class is much smaller now.](javascript:void(0)) [Instead of computing the result directly, the rater is going to rely on an algorithm](javascript:void(0)) [that is passed in and abstracted behind an interface definition.](javascript:void(0)) [The interface definition is "I-Rating algorithm", and it looks like this.](javascript:void(0)) [Any object that implements this interface you can call](javascript:void(0)) ["Compute on" and pass in a list of reviews.](javascript:void(0)) [It will then return the proper new rating result.](javascript:void(0)) [I've made two concrete implementations of this interface.](javascript:void(0)) [They're both in the same file.](javascript:void(0)) [That's something that I'd want to change later on, move these classes](javascript:void(0)) [to their own files perhaps, but we can worry about that later.](javascript:void(0)) [The first implementation performs a "Simple Averaging of Reviews".](javascript:void(0)) [The second implementation performs a "Weighted Averaging of the Reviews".](javascript:void(0)) [You might have noticed that the code inside the algorithms is the same code I had before inside](javascript:void(0)) [of the "Restaurant Rater" itself, so I just moved code around.](javascript:void(0)) [I took code out of the "Restaurant Rater" and pasted it into these algorithm classes.](javascript:void(0)) [Is that good?](javascript:void(0)) [I think so, because it's assigning specific responsibilities to different classes now.](javascript:void(0)) [I now have algorithms that are dedicated to and focused on computing a rating result,](javascript:void(0)) [and I have a "Restaurant Rater" that is now going to be focused on coordinating all](javascript:void(0)) [of the pieces together needed to produce that result.](javascript:void(0)) [And now when a client needs to determine a rating, like if we looked inside](javascript:void(0)) [of our tests now, we don't need to figure out which method to call on the "Restaurant Rater".](javascript:void(0)) [We always call "Compute Result", but we pass](javascript:void(0)) [in the algorithm that's required to perform the computation.](javascript:void(0)) [This is the Strategy Design pattern for you pattern fans out there.](javascript:void(0)) [What's important about this pattern is how I can now introduce new algorithms](javascript:void(0)) [without changing any of the code inside the "Restaurant Rater" or inside](javascript:void(0)) [of any of the existing algorithms.](javascript:void(0)) [I've made the system much easier to extend, and we should be able to keep](javascript:void(0)) [up with all the changes that the business wants to introduce in this area as they experiment](javascript:void(0)) [with different ways to rate restaurants.](javascript:void(0)) [All I need to do is write a class that implements "I-Rating algorithm".](javascript:void(0)) [The good news is, after I've ripped all this code apart, all of the tests still pass,](javascript:void(0)) [so I know I haven't broken anything.](javascript:void(0)) [And you might be wondering, if all I did was move code around,](javascript:void(0)) [then why couldn't I just call this algorithm directly instead](javascript:void(0)) [of going through this rater class?](javascript:void(0)) [If I'm a client, like our unit tests are clients for this code,](javascript:void(0)) [why do I instantiate a "Restaurant Rater" and call "Compute Rate"?](javascript:void(0)) [Why don't I just instantiate an algorithm and call "Compute Rate"?](javascript:void(0)) [And the answer is that we haven't given the "Restaurant Rater" all](javascript:void(0)) [of the responsibilities it needs to take care of just yet.](javascript:void(0)) [Let me write another test and come right back.](javascript:void(0)) [I've written a new test, this one to make sure that the](javascript:void(0)) ["Restaurant Rater" only uses the top N number of reviews in the calculation.](javascript:void(0)) [To test this scenario, I've built a restaurant with six reviews, one, one, one followed by 10,](javascript:void(0)) [10, 10, and I expect if I tell the rater to only use the first three reviews,](javascript:void(0)) [it should produce an average of one.](javascript:void(0)) [If I run the tests with this new test in place, I'll discover that this test fails](javascript:void(0)) [because I haven't implemented the logic for this to work yet.](javascript:void(0)) [Let me go into the "Restaurant Rater", and get a list of filtered reviews.](javascript:void(0)) [And I think what I can do is just take "Restaurant.reviews", and take the first number](javascript:void(0)) [of reviews to use, and that should satisfy the logic.](javascript:void(0)) [This should make the test pass, but let me do a "Filtered Reviews.to List", run the test again,](javascript:void(0)) [and I can see that that test now passes.](javascript:void(0)) [And now you can see that the "Restaurant Rater" is going to have a purpose in life.](javascript:void(0)) [It's going to massage the data, and set everything](javascript:void(0)) [up properly so the algorithm can work.](javascript:void(0)) [It's still the algorithm that does the computations and leaves us open](javascript:void(0)) [to extending the software with new algorithms.](javascript:void(0)) [And at this point, I'm sure we could think of many more tests to write.](javascript:void(0)) [We could write tests that check for rounding errors and truncating errors,](javascript:void(0)) [tests for restaurants with no reviews,](javascript:void(0)) [tests to make sure the reviews are sorted properly, and the list goes on.](javascript:void(0)) [I'm not going to write those tests for you in this video.](javascript:void(0)) [I'll leave that as an exercise for you.](javascript:void(0)) [Instead, we're going to move on and look at more MVC-specific issues around unit testing.](javascript:void(0)) [But the goal of this was that I hope I've shown you some of the benefits](javascript:void(0)) [of test-first development and how to apply TDD.](javascript:void(0)) [It has a real learning curve.](javascript:void(0)) [You need to keep trying, and then learning, and then trying again.](javascript:void(0)) [You might think this example was too simplistic, but I assure you that after many years](javascript:void(0)) [of doing this if you can make your code easy to test, like in this example,](javascript:void(0)) [and there's always a way to make code easier to test, then you're going to have success,](javascript:void(0)) [not only with test-driven development but also](javascript:void(0)) [in writing maintainable code and building software.](javascript:void(0)) [( Silence )](javascript:void(0))

* + [Home Controller Tests](javascript:void(0))

[When you are unit testing an "ASP.NET MVC" application,](javascript:void(0)) [you'll probably want to unit test your controllers.](javascript:void(0)) [And one of the first decisions you'll need to make is if you want to isolate your controllers](javascript:void(0)) [from infrastructure-related services that they use, like web services or mail servers, or,](javascript:void(0)) [in the case of our application, a database.](javascript:void(0)) [There are two ways we could go.](javascript:void(0)) [We could write unit tests that hit the controllers,](javascript:void(0)) [and the controllers go against the database.](javascript:void(0)) [This is perfectly reasonable for some people.](javascript:void(0)) [Other people think it is an atrocity.](javascript:void(0)) [I just want to let you know there is some debate on this topic.](javascript:void(0)) [The problem with writing unit tests against your controllers when your controllers need](javascript:void(0)) [to access a database is that the unit tests can run a little bit slower,](javascript:void(0)) [and in unit testing you generally want the test to run as fast as possible.](javascript:void(0)) [But there's actually a bigger problem, and that's usually setup.](javascript:void(0)) [Because you are writing tests, you want the controllers to behave predictably,](javascript:void(0)) [and that means you need predictable data.](javascript:void(0)) [So you need to make sure the database is set up, it has the right data in it,](javascript:void(0)) [there's nothing new there that could break a test.](javascript:void(0)) [In the long run, this can actually end up being a lot of work.](javascript:void(0)) [I would suggest if you go down this path,](javascript:void(0)) [you might look at an embedded database like "Sequel Server Compact".](javascript:void(0)) [With the "Entity Framework", you can point the connection string to a Sequel compact database,](javascript:void(0)) [and it can be a little bit easier to work with compared to a real Sequel server.](javascript:void(0)) [But "Sequel Compact" doesn't support all the queries, and features that regular](javascript:void(0)) ["Sequel Server" supports, so you can still run into a little trouble,](javascript:void(0)) [in which case you might stick with a real Sequel server, but your tests are going to be slower.](javascript:void(0)) [I want to show you how to isolate your controllers](javascript:void(0)) [from the database using a simple approach, the simplest approach I can come up with.](javascript:void(0)) [Right now, a test for the home controller is failing](javascript:void(0)) [because the entity framework is running inside,](javascript:void(0)) [and it's looking for a database connection string.](javascript:void(0)) [It's the index test, and the exception that's being thrown is](javascript:void(0)) ["no connection string named default connection can be found in the application config file."](javascript:void(0)) [If I want this test to pass, I can set up a real database for my unit test to use,](javascript:void(0)) [or I can remove the database from the application when it's under test.](javascript:void(0)) [And removing the database usually involves some sort of interface definition.](javascript:void(0)) [Let me show you how this works.](javascript:void(0)) [First, we'll have an interface "IO Defu DB" (phonetic).](javascript:void(0)) [This will represent all the operations that I want to do against a real database.](javascript:void(0)) [Not everything is there yet.](javascript:void(0)) [All we have so far is just a "Query of T" method.](javascript:void(0)) [I want to be able to query different objects, restaurants and reviews.](javascript:void(0)) [The "Ode to Food DB" class that we've been using all along now implements "IO Defu DB".](javascript:void(0)) [It implements that query method explicitly, meaning you can only get to this query method](javascript:void(0)) [through an "IO Defu DB" reference.](javascript:void(0)) [All this method needs to do is turn around and call into the DB context that it derives from.](javascript:void(0)) [There's a set method on that DB context that would essentially be asking](javascript:void(0)) [for the queryable set of entities.](javascript:void(0)) [So if someone wants a query of "Restaurant", all we need to do is turn around](javascript:void(0)) [and return a set of "Restaurant".](javascript:void(0)) [Then inside the "Home Controller", I can have it work against an](javascript:void(0)) ["IO Defu DB" reference instead of a real "Ode to Food DB".](javascript:void(0)) [At runtime, when it's actually running on a web server, we'll give it a real "Ode to Food DB",](javascript:void(0)) [so when it says "DB.query Restaurants",](javascript:void(0)) [that'll actually be something that goes to "Sequel Server".](javascript:void(0)) [But when it's under test, the beautiful thing about an interface is the](javascript:void(0)) ["Home Controller" really doesn't know what it's talking to, and under test we can have](javascript:void(0)) ["IO Defu DB" point to something that just contains some in-memory data.](javascript:void(0)) [We do just have to make a couple changes to the "Home Controller".](javascript:void(0)) [One is that we can no longer say, "DB.restaurants".](javascript:void(0)) [Now we have to say, "I want to query restaurant," and we'll need to do](javascript:void(0)) [that in two places, once in the "Auto-Complete" and once in the "Index".](javascript:void(0)) [The other change that we'll have to make is that when the controller is constructed,](javascript:void(0)) [we'll need to initialize this to point to something.](javascript:void(0)) [When we're in a test, we want it to point to something fake that the test passes in.](javascript:void(0)) [When we're not in a test, we want it to use a real "Ode to Food DB".](javascript:void(0)) [For right now, let me just define two constructors, one default constructor](javascript:void(0)) [that will initialize this to the "Ode to Food DB" that we've always been using,](javascript:void(0)) [and another constructor where you will pass in an "IO Defu DB", and we'll just assign it.](javascript:void(0)) [That'll give a unit test a chance to pass in something for the "Home Controller" to use](javascript:void(0)) [that looks like a real database but it isn't.](javascript:void(0)) [And with that in place, what I can do inside of the test now](javascript:void(0)) [when we're arranging this controller, I can set up a fake database.](javascript:void(0)) [So I could say, "DB equals New, Fake Ode Defu DB".](javascript:void(0)) [This is a class that I've already written.](javascript:void(0)) [What it does is simply implement the "IO Defu DB" interface.](javascript:void(0)) [And behind the scenes there's just in-memory data.](javascript:void(0)) [It's not the prettiest code in the world with all the generics,](javascript:void(0)) [but essentially behind the scenes there's a dictionary of type.](javascript:void(0)) [So when you want to query "Restaurant", we'll pull out a bunch of "Restaurant" out of](javascript:void(0)) [that dictionary, something that's I-queryable, and hand it over to you.](javascript:void(0)) [And there's a method here called "Add Set", which allows you to program the restaurants](javascript:void(0)) [that you want inside of this "Fake Ode Defu" database.](javascript:void(0)) [In fact, I've already defined a class, "Test Data" that has some restaurants](javascript:void(0)) [that it just conjures up in-memory.](javascript:void(0)) [So 100 restaurants, every restaurant has a review with a rating of four.](javascript:void(0)) [And inside the "Home Controller" tests, what I can do is initialize the database with that,](javascript:void(0)) [to add a set from "Test Data.restaurants", and I can use that "Test Data.restaurant" over](javascript:void(0)) [and over again for all the controllers that need to query restaurant information.](javascript:void(0)) [And now when I construct the "Home Controller", I don't want it to use the real database.](javascript:void(0)) [I want it to use that fake in-memory database.](javascript:void(0)) [Just with those changes, let's run the test again and see what happens.](javascript:void(0)) [And now the index method is still failing, but it's a different exception now.](javascript:void(0)) [The exception now is because of our call to "Is Ajacks (phonetic) Request".](javascript:void(0)) [We've run into trouble with this before, but I've been leaving it in the](javascript:void(0)) ["Home Controller" just to make a point, and that is there's nowhere else in the application](javascript:void(0)) [where we have actually gone up to an "HCDP Context Object" or messed with the query string](javascript:void(0)) [or cookies or anything like that.](javascript:void(0)) [And for the most part, with "ASP.NET MVC", you don't need to do any of those things.](javascript:void(0)) [But when you do, you can almost be certain](javascript:void(0)) [that it will make the tests a little bit harder to write.](javascript:void(0)) [To this "Call to Request.is Ajacks Request",](javascript:void(0)) [what's happening behind the scenes is it's looking at the HCDP request.](javascript:void(0)) [Well, there isn't one; it sees a NULL reference somewhere, and an exception gets thrown.](javascript:void(0)) [Fortunately, with "ASP.net" MVC, there is a way to work around this.](javascript:void(0)) [And I have another class defined, if we come over here into the test project.](javascript:void(0)) [It is a "Fake Controller Context".](javascript:void(0)) [A "Fake Controller Context" is something we'll assign to the controller](javascript:void(0)) [and give it a "Fake HCDP Context".](javascript:void(0)) [A "Fake HCTP Context", then, is going to have a "Fake Request Object",](javascript:void(0)) [and this "Fake Request Object", again, really doesn't do anything,](javascript:void(0)) [just returns nulls, just returns empty collections.](javascript:void(0)) [All of this is just here so that we don't get that exception.](javascript:void(0)) [Very simple implementation, but it's a lot of code just to have that one call inside](javascript:void(0)) [of the index method to check if it's an "Ajacks Request".](javascript:void(0)) [And using these classes, what I can do is come back into the test and ALSO TELL the controller](javascript:void(0)) [that its controller context is going to be a "Fake Controller Context".](javascript:void(0)) [And now let's run this test one more time.](javascript:void(0)) [It's still going to fail, but at least we've gotten to the "Assert" statement.](javascript:void(0)) [The "Assert" is failing because we no longer put a message in the VIEW BAG that says,](javascript:void(0)) ["Modify this template to jump-start your MVC application."](javascript:void(0)) [We've changed it and gotten away from that starter template,](javascript:void(0)) [so this "Assert" really isn't valid anymore.](javascript:void(0)) [But now what could we assert.](javascript:void(0)) [We just fed the "Home Controller" a fake data source that has 100 restaurants in it.](javascript:void(0)) [We only want 10 on the home page, so maybe what I should do is look at the model that's in this](javascript:void(0)) ["View Result" and assert that we have 10 restaurants inside of it.](javascript:void(0)) [Remember, our "Home Controller" should be returning a sequence](javascript:void(0)) [of "Restaurant List View Model".](javascript:void(0)) [So what I could say here is that I expect to have an I-numerable](javascript:void(0)) [of "Restaurant List View Model", that's my model, and it's equal to "Resolve.model",](javascript:void(0)) [which is typed as "Object" because "View Result" really doesn't know what type](javascript:void(0)) [of model you're going to have, so if I just do some type coercion here,](javascript:void(0)) [then my code should be happy and, now I could write an assert like,](javascript:void(0)) ["Assert that R equal 10 is the number of those items in the model."](javascript:void(0)) [And finally, run the test one more time, and finally,](javascript:void(0)) [we have a passing test that's actually testing something that we want.](javascript:void(0)) [And this is the kind of path that you'll go down if you want to isolate your controllers](javascript:void(0)) [from actual infrastructure things, program to interfaces, have fake or stub](javascript:void(0)) [or mock implementations of those interfaces for your tests.](javascript:void(0)) [And we can build a whole course around the best testing strategies, and fakes versus mocks.](javascript:void(0)) [In fact, you'll find those courses on "Pluralsight"](javascript:void(0)) [if you're a "Pluralsight" subscriber.](javascript:void(0)) [I'm just trying to give you a sample of what this would look like.](javascript:void(0)) [( Silence )](javascript:void(0))

* + [Create Action Tests](javascript:void(0))

[Let's see what it would look like to test the "Create" action of the "Restaurant Controller".](javascript:void(0)) [Here are two tests with less than 10 lines of code for each test.](javascript:void(0)) [The first test wants to make sure a controller saves a restaurant when the restaurant is valid.](javascript:void(0)) [The second test wants to make sure we do not save a restaurant](javascript:void(0)) [when the model state is invalid.](javascript:void(0)) [Both of these tests use our fake database to make it easy to test.](javascript:void(0)) [In the scenario where there is an error, we set up "Model State" to look like validation failed.](javascript:void(0)) [I can do that by adding a model error to "Model State".](javascript:void(0)) [You can call "Add Model Error" to simulate an error condition.](javascript:void(0)) [You can also call "Add Model Error" inside of "Controller"](javascript:void(0)) [if you wanted a real error condition that would display to the user.](javascript:void(0)) [But I will point out that you need to be careful.](javascript:void(0)) [If you're calling "Add Model Error" a lot in your controllers,](javascript:void(0)) [it might mean that your controllers have too many validation responsibilities,](javascript:void(0)) [and validation is generally a responsibility you can assign elsewhere.](javascript:void(0)) [In this application, we're using validation attributes,](javascript:void(0)) [but you could also have a model validate itself.](javascript:void(0)) [Once we call the "Create Action", we'll make sure nothing was added to the](javascript:void(0)) ["Added" collection of our fake database.](javascript:void(0)) [And this demonstrates how fakes have a different purpose than real objects.](javascript:void(0)) [Fakes should make things easy to test, perhaps even adding additional properties and methods](javascript:void(0)) [to figure out how the object was used by the code under test.](javascript:void(0)) [In the "Create Scenario", where we want the "Create" to work,](javascript:void(0)) [we can make sure our restaurant was added to the database by asserting that "Added.count is equal](javascript:void(0)) [to one", and also check the saved property of the fake database](javascript:void(0)) [to ensure the controller called "Save Changes", which will save everything.](javascript:void(0)) [Obviously, the fake database has some additional features now for the "Create" scenario](javascript:void(0)) [to be testable like this, so let's see what they are.](javascript:void(0)) [We'll start with the interface definition.](javascript:void(0)) [This interface definition now contains everything](javascript:void(0)) [that we currently need to do against the database.](javascript:void(0)) [We can query objects, add an object, update an object, remove an object,](javascript:void(0)) [and call "Save Changes" to flush everything to the database.](javascript:void(0)) [For the real "Ode to Food DB" that implements this interface,](javascript:void(0)) [we just need to forward these calls to the "Entity Framework" DB Context API,](javascript:void(0)) [calling methods like "Add" or "Remove" on a DB set,](javascript:void(0)) [or in the case of "Update" we call the "Entry" API.](javascript:void(0)) [Remember, "Entry" was a way of attaching an existing object to the context](javascript:void(0)) [so the entity framework would track the object, and it will issue an update statement](javascript:void(0)) [because we tell the framework the object has been modified.](javascript:void(0)) [The fake database takes a different approach.](javascript:void(0)) [When you call "Add", it keeps the object in memory in a collection](javascript:void(0)) [that remembers all the added objects.](javascript:void(0)) [So during a test, it would be easy to see what's been added,](javascript:void(0)) [what's been updated, what's been removed.](javascript:void(0)) [We also have a flag telling us when someone calls "Save Changes", so we can write asserts](javascript:void(0)) [and make sure that changes were committed to the database.](javascript:void(0)) [And the end result is that we can write relatively simple tests,](javascript:void(0)) [tests that will execute very fast because all the data is in memory.](javascript:void(0)) [We can also control the data in this fake database, and we don't have to set up a schema](javascript:void(0)) [or database file or make sure we're pointed to the right database server.](javascript:void(0)) [And this represents, as I said, just one approach to testing controllers.](javascript:void(0)) [Some people will test against the database; some people will test against in-memory fakes.](javascript:void(0)) [You have to find what you are comfortable with,](javascript:void(0)) [and what works for your team and for your application.](javascript:void(0)) [And the tests here might be tests you want to write,](javascript:void(0)) [or you can certainly have different tests or additional tests.](javascript:void(0)) [You might want to test that the controller returns the proper "Action Result",](javascript:void(0)) [like a "Redirect" result when it successfully saves or updates and object.](javascript:void(0)) [And that would also be a valid and easy test to write.](javascript:void(0)) [( Silence )](javascript:void(0))

* + [Summary](javascript:void(0))

[In this module, we looked at test-driven development](javascript:void(0)) [and unit testing with "ASP dot" and "MVC".](javascript:void(0)) [I demonstrated the test-first development cycle](javascript:void(0)) [of red-green re-factor to implement a new feature.](javascript:void(0)) [We saw how TDD is more about design than quality control, but a wonderful side effect](javascript:void(0)) [of TDD is how we have unit tests in place when we're finished with the implementation,](javascript:void(0)) [and that allows us to make changes in the future to the code.](javascript:void(0)) [We also looked at unit testing controllers.](javascript:void(0)) ["MVC" makes controllers easy to test.](javascript:void(0)) [For the most part, you just need to instantiate them and pass](javascript:void(0)) [in parameters, and assert on the result.](javascript:void(0)) [We just have to make sure the code we write inside the controller is testable.](javascript:void(0)) [And we looked at using fake testables to give us complete control over the execution environment](javascript:void(0)) [for a controller, and avoid using a database in our unit tests.](javascript:void(0)) [If you're new to unit testing, I really hope this module has given you some inspiration](javascript:void(0)) [to try it out.](javascript:void(0))

* Deployment and Configuration
  + [Introduction](javascript:void(0))

[Hi, this is Scott Allan.](javascript:void(0)) [And in this module I'll be focusing on configuration](javascript:void(0)) [and deployment of an ASP.NET MVC application.](javascript:void(0)) [I'll talk about the various configuration files that you'll find on the machine,](javascript:void(0)) [and what those configuration files mean to your application.](javascript:void(0)) [I'll also be transforming my development machine into a webserver](javascript:void(0)) [by installing internet information services -- or IIS --](javascript:void(0)) [and building a web deployment package to install the application into IIS.](javascript:void(0)) [Then finally, we'll take a look at deploying](javascript:void(0)) [to the cloud using Microsoft's Windows Azure websites.](javascript:void(0)) [( Silence )](javascript:void(0))

* + [Configuration Files](javascript:void(0))

[Configuration files and .NET are XML files that control everything](javascript:void(0)) [about the environment your code will execute inside of.](javascript:void(0)) [For web applications, this includes the authentication settings as we saw](javascript:void(0)) [in the security module of this course, but config files also control the compilation](javascript:void(0)) [of use, database connection strings, cryptography settings,](javascript:void(0)) [custom errors settings, and much, much more.](javascript:void(0)) [The configuration system in .NET is also extensible,](javascript:void(0)) [so if you want to build your own custom configuration section,](javascript:void(0)) [all you need to do is write some classes to support that.](javascript:void(0)) [I'm not going to show you that specific scenario in this module,](javascript:void(0)) [but I will demonstrate an easy way for you to store custom settings in a configuration file.](javascript:void(0)) [Inside of Visual Studio I'm going to open up the web.config file that's in the root](javascript:void(0)) [of this project; the one we've been using repeatedly throughout this course.](javascript:void(0)) [This is where we specify the connection string for the database that we want to use.](javascript:void(0)) [We've also looked at the globalization settings in here, we've created some cash profiles,](javascript:void(0)) [we've configured custom errors, and we've looked at authentication.](javascript:void(0)) [So there are various components of ASP.NET and ASP.NET MVC, and also of the webserver itself](javascript:void(0)) [that look in this configuration file for settings.](javascript:void(0)) [And so, the obvious question might be what if I want](javascript:void(0)) [to keep my own settings inside of web.config?](javascript:void(0)) [One of the easiest things to do if you just have a simple value that you want to store is](javascript:void(0)) [to add it to the app settings section.](javascript:void(0)) [Right here I can add a new setting with a key of mail server and give it a value.](javascript:void(0)) [And now, I want to be able to retrieve that value inside of my MVC application; let's say,](javascript:void(0)) [somewhere inside of the home controller.](javascript:void(0)) [Let's just read that value out, and put it into a view.](javascript:void(0)) [The easiest way to do that would be to just put it into viewbag.](javascript:void(0)) [And the way I can read that setting is to use configuration manager that's](javascript:void(0)) [in a namespace system.configuration.](javascript:void(0)) [Configuration manager will give me access to connection strings, to app settings;](javascript:void(0)) [really to anything that's in a config file.](javascript:void(0)) [But here I'm explicitly asking for the mail server app setting.](javascript:void(0)) [We'll grab that value.](javascript:void(0)) [We'll put it in viewbag, and in the index view that that action renders.](javascript:void(0)) [Let's delete some of the culture globalization stuff that we were fiddling with](javascript:void(0)) [and instead write out viewbag.mailserver to see if we've retrieved this correctly.](javascript:void(0)) [I'll do a build and refresh the homepage of the application,](javascript:void(0)) [and there you can see we get out mail.server.com.](javascript:void(0)) [So that worked.](javascript:void(0)) [And configuration manager makes it very easy to grab app settings,](javascript:void(0)) [and really any setting that's in a web.config file.](javascript:void(0)) [And this is the place where you want to put things](javascript:void(0)) [that you don't necessarily want to hardcode.](javascript:void(0)) [So connection strings, server names, file share names; all of those types](javascript:void(0)) [of things can go into your configuration file.](javascript:void(0)) [That way, you can change them just by changing the config file and you don't have](javascript:void(0)) [to recompile and redeploy all your binaries.](javascript:void(0)) [( Silence )](javascript:void(0))

* + [Configuration Hierarchy](javascript:void(0))

[Configuration files for .NET are hierarchical, meaning the configuration](javascript:void(0)) [for your application is controlled not only by the web.config in your project,](javascript:void(0)) [but also by configuration files at a higher level.](javascript:void(0)) [When you're running the MVC application, the configuration starts](javascript:void(0)) [with a machine level configuration file.](javascript:void(0)) [That file controls basic settings for all .NET applications that run on the same computer;](javascript:void(0)) [desktop applications, windows services, and even web applications -- everything.](javascript:void(0)) [We'll take a look at where this file lives and what's inside in just a minute.](javascript:void(0)) [There's also a machine level web.config file.](javascript:void(0)) [We looked at that when we were poking around in the health monitoring settings.](javascript:void(0)) [It puts in place all the default settings for every ASP.NET application on the machine.](javascript:void(0)) [And before we get to the web.config that's in our application,](javascript:void(0)) [we might also have a parent web.config file that we inherit settings from, and that would happen](javascript:void(0)) [if our application is deployed underneath another ASP.NET web application.](javascript:void(0)) [These configuration settings then are inherited downwards in the configuration in effect](javascript:void(0)) [for your application is a combination of all the configuration files.](javascript:void(0)) [Inside my web.config file I can generally override any settings that might be in place](javascript:void(0)) [from the machine level config or the machine level web.config file,](javascript:void(0)) [although administrators do have the option of locking down settings](javascript:void(0)) [and preventing me from changing a value.](javascript:void(0)) [Let's take a look at what these configuration files look like.](javascript:void(0)) [Using Windows Explorer, let's find the machine level configuration file.](javascript:void(0)) [I need to look on the C drive under windows, under Microsoft.net, framework.](javascript:void(0)) [And as I mentioned in an earlier module, even though we're running .NET 4.5 we need](javascript:void(0)) [to look in the version 4 config folder.](javascript:void(0)) [It's inside of here where we will find the machine level web.config.](javascript:void(0)) [I can open that up.](javascript:void(0)) [It's about the most boring configuration file you'll ever see.](javascript:void(0)) [Most of it is just sections describing the types to use](javascript:void(0)) [for other configuration sections when they're loaded.](javascript:void(0)) [But there are some things in here, and there are some things](javascript:void(0)) [in here that actually apply to ASP.NET.](javascript:void(0)) [For instance, all the way at the bottom,](javascript:void(0)) [here's the default membership provider that's configured for all ASP.NET applications.](javascript:void(0)) [It's the ASPNET sql membership provider, but we're not using this in our application.](javascript:void(0)) [We have changed the setting in our web.config -- overridden it.](javascript:void(0)) [Down here at the bottom -- remember we added our own membership section](javascript:void(0)) [that is using the simple membership provider.](javascript:void(0)) [So that's an example of changing the defaults that are specified at a higher level.](javascript:void(0)) [We can also find the root level web.config file here.](javascript:void(0)) [This is the file that would apply to all ASP.NET web applications](javascript:void(0)) [that are running version 4 or version 4.5 of .NET.](javascript:void(0)) [It's inside of here where you can find things like the default authorization rule;](javascript:void(0)) [will allow all users, even anonymous users into every site unless you tell us otherwise.](javascript:void(0)) [It's inside of this file where you can also find the default health monitoring rules --](javascript:void(0)) [we looked at those previously -- the default HTTP handlers;](javascript:void(0)) [that is like a file with and ASHX extension.](javascript:void(0)) [What is the handler for that?](javascript:void(0)) [What class does the .NET runtime need to instantiate to handle that request?](javascript:void(0)) [And we can also see inside of here some default modules.](javascript:void(0)) [Modules provide services like implementing our forms authentication,](javascript:void(0)) [doing checks against the cookie that arrives to see if the user has been authenticated.](javascript:void(0)) [Also, the routing engine is an HTTP module.](javascript:void(0)) [It runs in the ASP.NET pipeline and gets to look at every request.](javascript:void(0)) [That's what an HTTP module can do.](javascript:void(0)) [So many settings in this root level web.config file, but eventually the settings inside](javascript:void(0)) [of our own web.config file get to take effect.](javascript:void(0)) [And these can override the settings like for membership.](javascript:void(0)) [And we can even inside of our own application have web.config files](javascript:void(0)) [that override settings for a specific directory.](javascript:void(0)) [For instance, there's a web.config file in the views folder.](javascript:void(0)) [This is the web.config file that controls razor configuration.](javascript:void(0)) [These are the namespaces that are in effect when a razor template is parts and compiled.](javascript:void(0)) [We added a few custom namespaces here during the course.](javascript:void(0)) [And there's other interesting things in here too.](javascript:void(0)) [Like if I scroll down here a little bit,](javascript:void(0)) [there is an HTTP handler registered essentially saying, if a request arrives here looking](javascript:void(0)) [for any type of file using any type of HTTP verb -- doesn't matter if it's a get or a post --](javascript:void(0)) [the component that will handle that is the HTTP not found handler.](javascript:void(0)) [The HTTP not found handler essentially returns a 404 error to the client.](javascript:void(0)) [Why are we doing that inside of the views folder?](javascript:void(0)) [We're doing that because if someone launches a browser and request and file that is](javascript:void(0)) [in this views folder directly -- like if they typed](javascript:void(0)) [in localhost/views/home/index.cshtml, we want to return a 404 error.](javascript:void(0)) [A user shouldn't be able to go directly to a view using a browser.](javascript:void(0)) [They should go through a controller.](javascript:void(0)) [It's the controller that gets to select a view.](javascript:void(0)) [Therefore, we'll just return a 404 if anyone requests a file inside of here directly.](javascript:void(0)) [And this file is here by default when you start a new MVC application.](javascript:void(0)) [( Silence )](javascript:void(0))

* + [Hosting ASP.NET MVC](javascript:void(0))

[Now that we understand a little bit about ASP.NET configuration,](javascript:void(0)) [let's see how to deploy an application and how to change configurations on a deployed website.](javascript:void(0)) [First, understand that when you build an MVC application, you're producing a .DLL file](javascript:void(0)) [with the application logic inside.](javascript:void(0)) [That's going to sit in the bin directory of your project.](javascript:void(0)) [DLL's don't execute by themselves.](javascript:void(0)) [They need some sort of host process to load them into memory.](javascript:void(0)) [And for websites that host processes, also then responsible](javascript:void(0)) [for delivering HTTP requests to the logic inside the DLL.](javascript:void(0)) [We've been using IIS Express as the host for our application so far.](javascript:void(0)) [IIS Express makes things very easy for development because it runs with our identity,](javascript:void(0)) [and we can start and stop the webserver whenever we need to.](javascript:void(0)) [But if you're deploying an application for real on the internet or intranet,](javascript:void(0)) [chances are you'll be using the full version of internet information services.](javascript:void(0)) [You can install IIS on nearly any version of Windows, but it is off by default](javascript:void(0)) [on nearly all versions of Windows.](javascript:void(0)) [So you might need to go into Windows features](javascript:void(0)) [or use the web platform installer to get IIS onto a machine.](javascript:void(0)) [Let me show you how that works.](javascript:void(0)) [What I want to do now is transform my development environment](javascript:void(0)) [into a server type environment by installing IIS.](javascript:void(0)) [And one of the easiest ways to do that is to use the web platform installer.](javascript:void(0)) [You might remember that we used this in the introduction](javascript:void(0)) [to this course to install Visual Studio.](javascript:void(0)) [Now, I'm going to launch it, and install IIS as well as a few other tools.](javascript:void(0)) [So up here in the top right where I can search, I'm going to search for IIS and ASP.NET.](javascript:void(0)) [That should find me this entry; IIS with ASP.NET 4.5.](javascript:void(0)) [Let's add that.](javascript:void(0)) [I also want to be able to manage IIS.](javascript:void(0)) [So let me search for IIS, and select the IIS management console.](javascript:void(0)) [And I'm also going to install sql server --](javascript:void(0)) [a sql server express edition because even though we already have local DB installed,](javascript:void(0)) [it turns out that using local DB from IIS is a bit of a pain.](javascript:void(0)) [It can be done, but it requires a lot of configuration and it's configuration](javascript:void(0)) [that you'd never have to do, because when you really do deploy to production,](javascript:void(0)) [chances are you'll have a real instance of sql server and a real license,](javascript:void(0)) [and everything will be set up for you.](javascript:void(0)) [I'm just going to install sql server express just to get this running under IIS,](javascript:void(0)) [and let you see what it would look like.](javascript:void(0)) [I'm also going to install sql management tools because chances are we're going to have to log](javascript:void(0)) [into that database and tweak some permissions.](javascript:void(0)) [So I will install the sql server management studio.](javascript:void(0)) [Four components to install just click the install button.](javascript:void(0)) [I will need to fill out some information about the SA password.](javascript:void(0)) [SA stands for system administrator.](javascript:void(0)) [That's the all-powerful account inside of sql server.](javascript:void(0)) [I'm just giving it an initial password.](javascript:void(0)) [I can click continue and accept, and we'll come back after all](javascript:void(0)) [of this has finished downloading and installing.](javascript:void(0)) [Now the web platform installer has finished.](javascript:void(0)) [I should be able to exit out.](javascript:void(0)) [And I should be able to open up internet explorer and go to local host.](javascript:void(0)) [And I can see that IIS 8 is running on this machine.](javascript:void(0)) [That's a good sign.](javascript:void(0)) [This is exactly where we want to deploy our application.](javascript:void(0)) [So if I come to local host I won't see the IIS banner page.](javascript:void(0)) [I'll instead see Ode To Food.](javascript:void(0)) [Let's check to make sure the other components installed too.](javascript:void(0)) [I'm going to go and search for sql management studio, and once this has initialized I will try](javascript:void(0)) [to connect to the sql express instance.](javascript:void(0)) [There's two ways to get to a local sql express instance.](javascript:void(0)) [You can specify a server name/sqlexpress.](javascript:void(0)) [You can also use a period here to specify that you want to go](javascript:void(0)) [to the local sql express instance.](javascript:void(0)) [I should be able to click connect.](javascript:void(0)) [And that looks good, there's no databases there yet.](javascript:void(0)) [That's where we're going to put our Ode To Food database.](javascript:void(0)) [You can also use this management tool by the way, to connect to local DB.](javascript:void(0)) [A connection string is just like you see in your web.config file.](javascript:void(0)) [It's local DB inside a parenthesis, slash V11.0.](javascript:void(0)) [And when I connect to that I can see our Ode To Food DB as well](javascript:void(0)) [as some other temporary databases that Visual Studio has created for things](javascript:void(0)) [like storing the unit test results.](javascript:void(0)) [And finally, there was one more tool that we installed.](javascript:void(0)) [That was the IIS management console.](javascript:void(0)) [It wants me to launch the web platform installer.](javascript:void(0)) [We don't need to do that right now.](javascript:void(0)) [I just want to see that IIS is running on this machine.](javascript:void(0)) [Inside of this manager you can go through and modify any of the settings for this server.](javascript:void(0)) [A lot of this just maps down to the same XML files](javascript:void(0)) [that we've been looking at -- configuration files.](javascript:void(0)) [Inside of a server I can see two nodes: there's application pools.](javascript:void(0)) [An application pool is a process that a web application will run inside of.](javascript:void(0)) [We'll be using the default application pool.](javascript:void(0)) [I can already see there's one application running there.](javascript:void(0)) [That's the website that we went to that just has the IIS banner.](javascript:void(0)) [If you go into the task manager for Windows --](javascript:void(0)) [and in Windows 8 the task manager is a little bit fancier](javascript:void(0)) [than the previous versions of Windows.](javascript:void(0)) [So let me go to more details, and go to the details tab.](javascript:void(0)) [Inside of here I should be able to find a W3WP.EXE.](javascript:void(0)) [That's a world wide web worker process .EXE.](javascript:void(0)) [That will be the process that will host our MVC application default app pool.](javascript:void(0)) [And also, inside the manager if I drill into sites I can see there's one site in here now.](javascript:void(0)) [It's the default website.](javascript:void(0)) [That's where we will deploy our application.](javascript:void(0)) [You can have additional websites, and you can have web applications listed under that website.](javascript:void(0)) [We want to deploy our application to the root of this website.](javascript:void(0)) [So that's what we'll get ready to do next.](javascript:void(0)) [( Silence )](javascript:void(0))

* + [Preparing For Deployment](javascript:void(0))

[There's just a couple housekeeping tasks that I want to take care](javascript:void(0)) [of to get my application ready for deployment.](javascript:void(0)) [These mostly revolve around the database.](javascript:void(0)) [The first thing I'd like to do is re-do our database migrations.](javascript:void(0)) [We've been a little loose with our migrations.](javascript:void(0)) [We've been running with automatic migrations enabled, which means,](javascript:void(0)) [we don't get explicit migration scripts; or migration files in the migrations folder.](javascript:void(0)) [Instead when we go to the package manager and go to update database,](javascript:void(0)) [it just makes whatever changes in the database that it sees fit to do.](javascript:void(0)) [Whatever is needed to sink up the schema with our models.](javascript:void(0)) [But for my first deployment, I'd like to baseline the database and get us into a state](javascript:void(0)) [where we will not be using automatic migrations anymore.](javascript:void(0)) [We'll be a little more careful because after we deploy, we'll get live data in the database.](javascript:void(0)) [We need to start taking care of things, and making sure we're managing the schema properly.](javascript:void(0)) [At least that's how I feel.](javascript:void(0)) [I essentially want to restart the migrations.](javascript:void(0)) [There's nothing in the database that I need right now.](javascript:void(0)) [All the data comes from our seed method down here, which while I'm in here,](javascript:void(0)) [I'm also going to remove the 1000 restaurants that we create just to test paging and sorting](javascript:void(0)) [because I don't want to deploy those either.](javascript:void(0)) [And I will also delete our initial database migration script.](javascript:void(0)) [I'm just going to recreate this from scratch using a brand new database.](javascript:void(0)) [And the definition of the schema will be based on what is in our model classes right now;](javascript:void(0)) [what's a restaurant, what's a review.](javascript:void(0)) [And the best way to do this with entity framework migrations is to come in](javascript:void(0)) [and also delete your database, so it doesn't see anything there and knows](javascript:void(0)) [that you want to start from scratch.](javascript:void(0)) [And I can do this through the sql server management studio just by highlighting Ode](javascript:void(0)) [To Food and right-clicking and saying delete; or hitting the delete key.](javascript:void(0)) [I'm going to tell sql management studio to close any existing connections](javascript:void(0)) [to make sure we can drop this database.](javascript:void(0)) [And it looks like it's gone.](javascript:void(0)) [And now reopen the package manager console, and let's create our initial migration.](javascript:void(0)) [Remember, we're not using automatic migrations anymore.](javascript:void(0)) [We won't be able to just say update database and have it create a database for us.](javascript:void(0)) [We need to have a migration script there first.](javascript:void(0)) [So add migration, initial create.](javascript:void(0)) [And I need to make sure that this executes us under the right project;](javascript:void(0)) [not OdeToFood.test, but OdeToFood.](javascript:void(0)) [That's where my entity framework DB context class is.](javascript:void(0)) [And now we will have an initial create that includes everything that we've been using,](javascript:void(0)) [the user profile class, restaurants, restaurant reviews.](javascript:void(0)) [And this is the point where I would also go through this file](javascript:void(0)) [and make sure all the column sizes are reasonable.](javascript:void(0)) [Make sure indexes are applied.](javascript:void(0)) [But for this application we can forge ahead.](javascript:void(0)) [Now, the other thing I'm going to want to do with this application is that it when deploys,](javascript:void(0)) [I want it to run the migrations automatically.](javascript:void(0)) [There's a way to do that through configuration, but we're actually going to do it through code.](javascript:void(0)) [And the other thing I want to do before I forget is make sure](javascript:void(0)) [that web security does not get initialized more than once](javascript:void(0)) [because if you call initialized database more than once,](javascript:void(0)) [this line of code will throw an exception and stop everything.](javascript:void(0)) [It's very easy to check if web security is initialized.](javascript:void(0)) [If web security initialized -- and put a not in front of here --](javascript:void(0)) [so if it's not initialized, we will initialize the database.](javascript:void(0)) [And the reason I'm putting that check there is because now we're going](javascript:void(0)) [to be running migrations when the web application runs.](javascript:void(0)) [So this initialized database connection can run when the web application is running.](javascript:void(0)) [And we have another call to initialize the database connection here during](javascript:void(0)) [application start.](javascript:void(0)) [I'm also going to put a guard around this.](javascript:void(0)) [Although I'm fairly certain it's always going to run before we reach this point because I want](javascript:void(0)) [to run the migrations before we get into this point in application start.](javascript:void(0)) [I'm still going to put that check there just to make sure it doesn't throw and exception.](javascript:void(0)) [And it is during the application start event when I want to run the migrations,](javascript:void(0)) [and I can do that using a class called DB migrator.](javascript:void(0)) [It is in the system.data.entity.migrations namespace.](javascript:void(0)) [So I'm just going to bring that in.](javascript:void(0)) [And we give it our configuration class that is in our project in a migrations folder.](javascript:void(0)) [So also bring that namespace in.](javascript:void(0)) [And then just tell it to update.](javascript:void(0)) [That will run any schema changes that need applied.](javascript:void(0)) [That will run the seed method to make sure the database is populated.](javascript:void(0)) [And the reason I'm doing this in code instead of in my web.config file, which is possible is](javascript:void(0)) [because I want to make very sure about when these migrations run.](javascript:void(0)) [I want them to run at this point before web security tries to initialize the database.](javascript:void(0)) [If you're not using forms authentication or if you are using forms authentication,](javascript:void(0)) [but not using web security, you wouldn't have to use this approach.](javascript:void(0)) [You could just put some configuration in place that would run the migrations automatically.](javascript:void(0)) [And the tools as we'll see in just a bit, will automatically add that configuration for us.](javascript:void(0)) [I'll point that out when we get to it.](javascript:void(0)) [But for now, I think we've reached the point where we are ready to deploy.](javascript:void(0))

* + [Deploying to IIS](javascript:void(0))

[When you're ready to deploy an application, you can right click](javascript:void(0)) [on your web project and select publish.](javascript:void(0)) [Visual Studio will figure out all of the files that need to be deployed to a web server,](javascript:void(0)) [collect them all together, and push them there.](javascript:void(0)) [All you need to do is tell it how to do that.](javascript:void(0)) [The first thing you do is select a publishing profile.](javascript:void(0)) [I'll create a new profile.](javascript:void(0)) [I'll give it the name release because this will be my release mode build that I want to publish.](javascript:void(0)) [And on the next screen, I can select a publish method.](javascript:void(0)) [There's several different methods here.](javascript:void(0)) [If you need to FTP the files to the server, Visual Studio can do that.](javascript:void(0)) [If you need to just push something onto a file system, you already know the directory](javascript:void(0)) [where the webserver will serve this application from; you can also do that.](javascript:void(0)) [You can select web deploy, and just enter in a server name and the site that you want](javascript:void(0)) [and it will contact the server and push your files over.](javascript:void(0)) [You can do this with shared hosting providers on the internet.](javascript:void(0)) [You can also do it on intranet servers if you're deploying with inside your company.](javascript:void(0)) [The one catch to web deploy here is I would want to push to local host.](javascript:void(0)) [And the problem is that in order to do this, I need to be an administrator.](javascript:void(0)) [So I could re-launch Visual Studio using renams](javascript:void(0)) [( assumed spelling )](javascript:void(0)) [administrator and then right click and publish and be successful,](javascript:void(0)) [but I'm going to show you a different approach.](javascript:void(0)) [We're going to build a web deploy package.](javascript:void(0)) [A web deploy package is ultimately just a zip file, and you can take the zip file and put it](javascript:void(0)) [on a USB key or email it to a administrator at your company, and they will be able](javascript:void(0)) [to take the web deploy package and copy it to a server, and install it for you.](javascript:void(0)) [In this case, we'll be doing the installing.](javascript:void(0)) [Let me create a package -- let me create a directory under dev.](javascript:void(0)) [We'll call it Ode To Food release.](javascript:void(0)) [And I need to include .zip in the name so release.zip.](javascript:void(0)) [And actually, let me put the zip file inside of a folder called release.](javascript:void(0)) [And then I pick the site where I want this to be deployed.](javascript:void(0)) [So it could be something like default website/OdeToFood, but I actually want this](javascript:void(0)) [in the root directory of the website.](javascript:void(0)) [So if someone just goes to local host, they will see this application running.](javascript:void(0)) [So the site I wanted to deploy it to on this IIS server is default website.](javascript:void(0)) [And that's actually just a parameter that you can change when you deploy it.](javascript:void(0)) [But now, if I click next, the next screen will go through what sort](javascript:void(0)) [of configuration do I want it to build in; release or to bug.](javascript:void(0)) [We will deploy in release mode and I can also pick a connection string](javascript:void(0)) [to use for default connection.](javascript:void(0)) [Remember, right now default connection is the connection string that's](javascript:void(0)) [in the web.config that's pointing to local DB.](javascript:void(0)) [When I deploy this to IIS, I want it to point to that sql express instance that I just installed.](javascript:void(0)) [So I will say that the default connection should actually point here.](javascript:void(0)) [The server name will be dot/sqlexpress.](javascript:void(0)) [I will use Windows authentication to connect.](javascript:void(0)) [We'll see how that creates a slight hiccup, but we'll work around that,](javascript:void(0)) [and the database that we will use; I will call it OTF production just](javascript:void(0)) [to make this dramatically different.](javascript:void(0)) [And select okay.](javascript:void(0)) [We can say yes, go ahead create this database for me.](javascript:void(0)) [And when this application is deployed, that connection string will be substituted](javascript:void(0)) [into the default connection, and replace whatever we are using there for development.](javascript:void(0)) [And then this checkbox here, execute first code migrations.](javascript:void(0)) [That's where I was telling you if you select this checkbox, Visual Studio will also add a bit](javascript:void(0)) [of configuration to your web.config file that will automatically run](javascript:void(0)) [into the framework code first migrations.](javascript:void(0)) [But we're actually going to explicitly run the migrations in our code,](javascript:void(0)) [so I'm going to leave that unselected.](javascript:void(0)) [Then there's a preview screen, which will tell you, you are about to create release.zip.](javascript:void(0)) [And let me just click publish.](javascript:void(0)) [All the files will be packaged up together.](javascript:void(0)) [And what I can do now is open up a command prompt; make sure I right click it and say run](javascript:void(0)) [as administrator because I will need administrator privileges](javascript:void(0)) [to actually get this installed into IIS.](javascript:void(0)) [And then come into that directory where I placed the zip file,](javascript:void(0)) [and inside of here I'll find the zip file.](javascript:void(0)) [I'll also find a command script that I can run that will do the installation for me.](javascript:void(0)) [And if I run it without any parameters, it will pop up a little help and notepad](javascript:void(0)) [that will tell me there's two ways to run this.](javascript:void(0)) [You can pass the /T to test the deployment and simulate what would happen,](javascript:void(0)) [and it will show you any errors that might occur.](javascript:void(0)) [Or, you can do a /Y to just go ahead and do the full deployment.](javascript:void(0)) [You can also point it to a remote computer, pass in a username and password;](javascript:void(0)) [all sorts of additional flags you could use.](javascript:void(0)) [Let's see what happens if we run this with a /T.](javascript:void(0)) [I don't see any error messages, so this just might go through if I run it with a /Y.](javascript:void(0)) [And now, if I come back into internet explorer, where we used to see an IIS 8 banner page,](javascript:void(0)) [if I refresh, we'll see our application trying to make a connection](javascript:void(0)) [to the database, but it can't quite get there.](javascript:void(0)) [It doesn't have permission to do it.](javascript:void(0)) [And this is because the IIS worker process that we looked at earlier W3WP.EXE runs](javascript:void(0)) [under an identity known as IIS app pool default app pool.](javascript:void(0)) [And this user account doesn't have any permissions](javascript:void(0)) [in the database or on the server at all.](javascript:void(0)) [When we're using local DB for development, we're administrators for local DB](javascript:void(0)) [and IIS express is running under our identity, so we can do anything with local DB.](javascript:void(0)) [But to get this application running with a production database,](javascript:void(0)) [I need to give some permissions to that default app pool account.](javascript:void(0)) [So I'll open up sql server management studio, and under security there will be logins.](javascript:void(0)) [What I want to do is create a new login for that account.](javascript:void(0)) [So the account is IIS app pools/default app pool.](javascript:void(0)) [And I'll come over into user mapping, and say please give this login the ability to get](javascript:void(0)) [into OTF production as a user, and give this login the ability to read data,](javascript:void(0)) [to write data, and to execute DDL.](javascript:void(0)) [Essentially execute things like create table.](javascript:void(0)) [The entity framework migrations will need to modify the schema.](javascript:void(0)) [We'll need to be DDL admin here.](javascript:void(0)) [And now click okay.](javascript:void(0)) [And now come back and refresh the browser and our application is now running under IIS 8.](javascript:void(0)) [We have our database in place; we have our C data in the database.](javascript:void(0)) [I should be able to log in, create restaurants, create reviews, and use all the features](javascript:void(0)) [that we've built into this application.](javascript:void(0))

* + [A Second Deployment](javascript:void(0))

[Now once you've completed all this work and you're done with the single deployment,](javascript:void(0)) [subsequent deployments are much smoother.](javascript:void(0)) [For instance, I'm looking at the home page.](javascript:void(0)) [I'm seeing it's displaying mail.server.com on the home page.](javascript:void(0)) [I don't necessarily want that there.](javascript:void(0)) [That was just an experiment we were doing to make sure](javascript:void(0)) [that we could read the configuration file that was in the home index view.](javascript:void(0)) [Let me open up and remove that, save the view, come back out to the project, and say publish.](javascript:void(0)) [It has already selected by release package that we created earlier.](javascript:void(0)) [So let me go ahead and click publish.](javascript:void(0)) [That will rebuild the zip file.](javascript:void(0)) [I can come out here to the command line, re-execute to the deploy.](javascript:void(0)) [Notice that it's smart enough to understand what's](javascript:void(0)) [on the server; what's new, what's changed.](javascript:void(0)) [It only copied over the files that needed change.](javascript:void(0)) [And now if I refresh the application, then that text is gone and we just made a small tweak](javascript:void(0)) [to the deployed version of the application.](javascript:void(0)) [And let's just see what's happening behind the scenes.](javascript:void(0)) [If I open up the IIS manager, I'll be able to come into the default website,](javascript:void(0)) [and switch over into content view.](javascript:void(0)) [That actually shows me the files that are sitting out here.](javascript:void(0)) [It does need to be refreshed.](javascript:void(0)) [And now I can see my views folder, my scripts folder,](javascript:void(0)) [my content folder with site.css inside of it.](javascript:void(0)) [You might be wondering what happened to the controllers folder.](javascript:void(0)) [This all looks like exactly like it did in my solution except there's some things missing.](javascript:void(0)) [Well the controllers folder is all C sharp code.](javascript:void(0)) [Visual Studio compiles all that code into a DLL.](javascript:void(0)) [Remember I said that when you build a web application you produce a DLL,](javascript:void(0)) [and that goes into this bin folder.](javascript:void(0)) [Here we can see OdeToFood.DLL that will have all the controller code compiled inside](javascript:void(0)) [of that assembly.](javascript:void(0)) [We also have a lot of other assemblies.](javascript:void(0)) [These are all things that we reference and that we need to run.](javascript:void(0)) [Assemblies like DotNetOpenAuth, which as I described](javascript:void(0)) [in the security module is what helps users log in using OpenAuth and OpenId.](javascript:void(0)) [And when a request arise or the default website -- aka local host --](javascript:void(0)) [IIS will spin up that W3WP.EXE process, load our assemblies into it,](javascript:void(0)) [and start sending us requests to process.](javascript:void(0)) [That's when MVC takes over, calls into our controller, which renders a view.](javascript:void(0)) [That's how the application all works.](javascript:void(0))

* + [Deploying to Windows Azure](javascript:void(0))

[Now that we've deployed to IIS on a local server,](javascript:void(0)) [let's deploy to IIS running in the cloud with Windows Azure.](javascript:void(0)) [To get started, I'll to go the Windows Azure website; WindowsAzure.com.](javascript:void(0)) [And if you haven't heard of Azure, it is Microsoft's Cloud platform](javascript:void(0)) [where you can host websites, databases and virtual machines.](javascript:void(0)) [You can run Windows or Linux, and you can start small and scale up really big just](javascript:void(0)) [by adding more resources for Microsoft's global data centers.](javascript:void(0)) [If I want to put a new website into Azure, I would first log into the Azure portal.](javascript:void(0)) [This is currently in preview, but by the time you watch this video it might be live.](javascript:void(0)) [I'll first need to sign in with my live ID.](javascript:void(0)) [( Silence )](javascript:void(0)) [And the first screen that is about to come up will be a screen](javascript:void(0)) [where I can see everything I have deployed in Azure under this account.](javascript:void(0)) [It's my portal where I can add new databases, new websites,](javascript:void(0)) [new virtual machines, new media services.](javascript:void(0)) [It's also where I can manage my existing services, and where I can create new services,](javascript:void(0)) [which is what I'll want to do when this is finished loading.](javascript:void(0)) [We want to add a new website and give the website a database for storage.](javascript:void(0)) [And you can see I have a few things out there already; couple websites, couple databases,](javascript:void(0)) [a virtual machine, a storage account.](javascript:void(0)) [But I want to create a new website with a database to host Ode To Food.](javascript:void(0)) [First I'll need to give my website a URL.](javascript:void(0)) [What will happen is that my website will be OdeToFood.AzureWebsites.net.](javascript:void(0)) [I can also get a DNS and point this to my website in Azure,](javascript:void(0)) [so it could just be OdeToFood.com; but we'll leave it as AzureWebsites.net for right now.](javascript:void(0)) [I get to pick a region for the data center.](javascript:void(0)) [I'll pick the eastern United States.](javascript:void(0)) [I want to create a new sequel database.](javascript:void(0)) [My connection string name -- let's call it OdeToFoodDB.](javascript:void(0)) [And in the next step I'll need to specify my database settings.](javascript:void(0)) [This is the name that I want to use.](javascript:void(0)) [The server will be a sql server that I already have in Azure; I just need to log in properly.](javascript:void(0)) [And I can show you what the advanced database settings look like.](javascript:void(0)) [This is where I can select my addition wherever business The maximum database size one gigabyte](javascript:void(0)) [will be good enough to start, and my collation.](javascript:void(0)) [And with all that in place I can select okay,](javascript:void(0)) [and Azure will go off and start creating this website.](javascript:void(0)) [As soon as it's done creating that I'll have a website on the internet that I can go to.](javascript:void(0)) [It'll have just the blank Azure template.](javascript:void(0)) [And what we'll need to do is publish our website now, and set it to local IIS to this website.](javascript:void(0)) [There are a number of publishing options available with Azure websites.](javascript:void(0)) [You can see them in the list here.](javascript:void(0)) [You can use GIT, you can use TFS, you can use Visual Studio or WebMatrix.](javascript:void(0)) [We're going to be using Visual Studio; it's very simple.](javascript:void(0)) [First I'll go to this website.](javascript:void(0)) [This brings up the dashboard where I can look at things like how much CPU has been used](javascript:void(0)) [by the website over the last 24 hours or over the last week.](javascript:void(0)) [But it's also where I can download a publishing profile.](javascript:void(0)) [Remember when we were publishing to IIS we created a publishing profile.](javascript:void(0)) [This is going to download one for me that I could just plug into Visual Studio,](javascript:void(0)) [and it will have all the settings that it needs to upload my website here.](javascript:void(0)) [And we just downloaded it into my downloads folder.](javascript:void(0)) [And now, let me go into Visual Studio, and let's publish this website again.](javascript:void(0)) [I'm going to go to the profile tab, because I want to import this new profile.](javascript:void(0)) [And there it is; the publish settings file.](javascript:void(0)) [If you open that up, it's just a big scary amount of XML inside, but it has everything](javascript:void(0)) [that I need to get this to the right server.](javascript:void(0)) [So it already includes the URL, the site name, the user name, passwords.](javascript:void(0)) [All you need to do is -- I can validate the connection; make sure this is going to work.](javascript:void(0)) [And that looks good.](javascript:void(0)) [So let's go to the next phase where you can see we're going](javascript:void(0)) [to again deploy a release mode build.](javascript:void(0)) [It is already understood that my default connections string will probably want](javascript:void(0)) [to point to that sql server in the cloud.](javascript:void(0)) [It will have populated the server name and the user name](javascript:void(0)) [and the password, and everything I need.](javascript:void(0)) [We're not going to run the entity code first migrations.](javascript:void(0)) [We're taking explicit control of our migrations in the code.](javascript:void(0)) [I'll just go to next.](javascript:void(0)) [This allows me to preview what it is going to push out.](javascript:void(0)) [But I'm feeling lucky, so I'm just going to click publish.](javascript:void(0)) [Since this is going over the internet, it might take a little bit of time.](javascript:void(0)) [The first time you need to publish it needs to push out a lot of assemblies,](javascript:void(0)) [a lot of data going over the network.](javascript:void(0)) [We'll come back in a second when it's finished pushing everything up into the cloud.](javascript:void(0)) [And the publishing operation has just succeeded.](javascript:void(0)) [I should now be able to go back into the browser.](javascript:void(0)) [Let's refresh this website.](javascript:void(0)) [And I'm now running on AzureWebsites.net, very simple to deploy.](javascript:void(0)) [All my functionality should be there.](javascript:void(0)) [The database is working.](javascript:void(0)) [We see the database.](javascript:void(0)) [That means I should be able to log into this application.](javascript:void(0)) [( Silence )](javascript:void(0)) [That was successful.](javascript:void(0)) [I should also be able to go and create new restaurant at this point.](javascript:void(0)) [And since I just launched, I don't expect a lot of traffic.](javascript:void(0)) [The wonderful thing about ager is that this is now in someone else's data center.](javascript:void(0)) [They make sure the electricity stays on.](javascript:void(0)) [They're making sure they take backups at all the right time.](javascript:void(0)) [And as this grows in popularity, I can come in and scale this thing out.](javascript:void(0)) [I can reserve my own machine, crank up the number of instances,](javascript:void(0)) [so I have more instances available to serve up all the requests that are coming in.](javascript:void(0)) [And of course, you'll pay for the resources that you use, but hopefully you'll be making money](javascript:void(0)) [from all of the customers that come to the website,](javascript:void(0)) [and that will pay for everything that you need in Azure.](javascript:void(0)) [( Silence )](javascript:void(0))

* + [Good Bye, and Good Luck](javascript:void(0))

[This concludes this module on configuration and deployment where we looked](javascript:void(0)) [at all the configuration files that are in effect when running a web application,](javascript:void(0)) [and we also deployed to IIS both locally and in Windows Azure.](javascript:void(0)) [This also concludes my ASP.NET MVC 4 course.](javascript:void(0)) [We've seen how to work with routing, controllers, and razor views.](javascript:void(0)) [We added javascript and used jQuery and jQuery UI to add Ajax features to the application.](javascript:void(0)) [And we also saw how to work with foreign authentication and the membership provider](javascript:void(0)) [that comes with our MVC application.](javascript:void(0)) [I hope you enjoyed this course and were able to use the videos](javascript:void(0)) [to either jumpstart your MVC development or solidify your existing knowledge](javascript:void(0)) [of the framework, just a reminder that all the code](javascript:void(0)) [that we've been working on is published on codeplex.](javascript:void(0)) [If you look at the history of the check-ins you'll see I did one check-in for each module.](javascript:void(0)) [So if you want to go back in time to a specific point and work along](javascript:void(0)) [with the video, you should be able to do that.](javascript:void(0)) [And now that we are finished I'd just like to say thanks for listening,](javascript:void(0)) [and good luck with your software development endeavors.](javascript:void(0))