Models, Views, Controllers 1

Here inside the Visual Studio you'll have a window with the title Solution Explorer. It is the Solution Explorer window that allows you to get to all of your files, C sharp files, JavaScript files, CSS files, icons, images and everything that goes into your application. When we created this application using the internet project template, Visual Studio went ahead and populated our application with all of the folders and files that you see here. That's why we already have a running application with contact in about links working. Three of these folders have a special significance, the controllers, the models and the views folders. Let's talk about those names for a minute. The ASP.NET MVC framework derives its name from the model view controller design pattern. This design pattern has been present in software applications for several decades at this point and it's a design pattern to follow when you want to separate the responsibilities of the components in your user interface layer. The C and MVC is for controller. A controller is a software component that will be the target for some external stimulus. In the case of a web application that external stimulus is usually an incoming HTTP request. So, when someone launches a web browser and points it to the slash home slash about location of my application. That incoming request needs to go to a controller that is in my application. When the controller receives the request, it's responsible for building a model that M in MVC. It's the model that contains all the information that you need to present to the user to satisfy that incoming request. In the case, of slash home slash about, the model might just be some information about the website or about the company or the people behind the website

Models, Views, Controllers 2

Another example would be a controller for handling request to view recipes. You might have a recipe controller that builds a list of the most popular recipes. That list would be the model. The controller then selects a view to display the model. Views in the MVC design pattern are very simple objects. Think of them as templates. They take pieces of data from the model and they place them into a proper location on the page. If the model was a list of recipes, then the controller might select a view that will take that list and display the recipes inside an HTML table. That end result is that you isolate the behaviors in your UI into one of these three categories, model, view or controller. A view would never need to know how to call into the data access layer because the model already contains all the data it needs. Meanwhile, a controller would never need to know about where to place an error message or how to color it because that's the responsibility of the view. That isolation that you achieve with the MVC pattern makes it easy to maintain and change your application moving forward because the code inside of each of these pieces is very focused and easier to understand. You can make a change in the controller about where to get your list of recipes or how to calculate what the best recipes are and that shouldn't impact the view which is only worried about displaying the recipes. I do want to point out that the MVC design pattern doesn't dictate what type of data access you use. You can use web services, relational databases, file system, document databases, any form of storage behind the scenes. And it also doesn't dictate what your business objects or domain layer should look like. In fact, the MVC design pattern doesn't care if your application has layers or not. It's simply a design pattern for building a user interface and nothing more. So the MVC framework is designed to help you follow this MVC design pattern by giving you tools and classes to build models, views and controllers. The MVC framework also has some additional goals. One of these goals is to embrace the web and to be able to work very closely with web technologies like JavaScript, HTML and CSS. There is no large obstructions here that try to shield you from knowing things like what HTTP verb is being used to place a request.

Action Filters 1

In addition to the action selector attributes we just looked at there are also a number of action filter attributes you can use. Action filters apply pre and post processing logic to a controller action and its result. Action filters are the components that you want to use to apply cross-cutting logic that's logic that has to execute across multiple controller actions but you don't want to duplicate code inside of individual controllers. One example is the output cash attribute this tells the run time that it's allowed to cash the final output of some action and to use that cashed result to service future requests. When you apply this attribute in the right places you can dramatically increase the through put and scalability of an application, we'll take a look at cashing later in this course. Another example is the authorize attribute, authorize allows you to ensure a user is logged in and perhaps a in a specific role like the admin. role before the action is allowed to process that user's request. We'll be looking at most of these filters during the duration of this course but right now I just want to show you how to apply a filter and also take a quick look at building your own custom action filter. Here inside of visual studio let's get rid of our second search action result and go back just to a single search action result, we'll have it respond to both a get and a post and I'm going to use the authorize attribute. You can just use the authorize attribute without any parameters, you can also specify something like roles equals admin., that means the user has to be logged in and they have to be in the admin role before we can invoke this function, you can also specify user names inside of here, I'm just going to use the authorize attribute without any parameters, what that tells the mvc framework is the user has to be logged in in order to be able to use the search action. Now if I do a build and I come back to refresh the application and what has happened is we've been redirected to the log on page because I've tried to go somewhere that requires me to be authorized, it requires me to be authenticated and the run time will see authorized attribute and it's smart enough to redirect me to the log on page where if I were to register on this site I could create an account and the log in.