ASP.NET Core Razor Pages Vs MVC: Which Will Create Better Web Apps in 2018?

Zealous System Follow
Jul 20, 2018 · 6 min read

Brace yourself because this is going to be a lengthy but informative article!

With the release of new ASP.NET Core 2 framework, Microsoft and its community has provided us with a brand new alternative for the MVC (Model-View-Controller) approach. Microsoft has named it Razor Pages, and while it's a little bit different approach, but it's still similar to MVC in some ways.

In this article, we are going to cover following important points of ASP.NET Razor Pages.

- Razor Pages—what is it exactly?
- Drawbacks of Using ASP.NET MVC

- Advantages of Using Razor Pages
- A Quick Comparison of How Requests Are Handled in Both

Razor Pages—What is It Exactly?

A Razor Page is very similar toASP.NET MVC's view component. It has basically same syntax and functionality as MVC.

The key difference between Razor pages and MVC is that the model and controller code is also included within the Razor Page itself.

In simple terms, it is much like am MVVM (Model-View-View-Model) framework. It provides two-way data binding and a simpler development experience with isolated concerns.

Though MVC works fine with web apps that have large amount of dynamic server views, single page apps, REST APIs, and AJAX calls, but Razor Pages are perfect for simple pages that are read-only or do basic data input.

Now, the ASP.NET MVC has been extremely popular for web applications development, and it definitely has its benefits. In fact, the ASP.NET WebForms was specifically designed as an MVVM solution in MVC.

But, the new ASP.NET Core Razor Pages is the next evolution of ASP.NET WebForms.

Drawbacks of ASP.NET MVC

As most of you probably know, MVC stands for Model-View-Controller. It is an architectural pattern used in software development for implementing UI (user interfaces).

While MVC is one of the most popular framework and is being used by millions of web developers worldwide, but it still has its drawbacks. Let's look at the two most important of them.

#1—Complexity

In ASP.NET MVC, there are piles of concepts such as TempData, RouteCollection, ViewData, Linq to SQL, Controller Action, Lambda Expression, Custom Route, and HTML Helpers, all of which tie the Model, View, and Controller.

Now, you cannot build a web application using ASP.NET MVC until you learn all these basic concepts. Plus, even if you've learned them, you will still face complexity issues at times, especially when you're building large-scale applications.

#2—Cost of Frequent Updates

In ASP.NET MVC, web developers cannot completely ignore the view of the model even when both are separated. The reason is because when the model is changed frequently, the views of your application could be flooded with update requests.

Views are basically graphical displays which takes some time to render depending on the complexity of your application. And if your application is complex and the model has been changed a lot, then the view may fall behind update requests. So, the developers then need to spend extra time fixing this situation, resulting into higher costs.

Advantages of Using Razor Pages

We've been providing ASP.NET MVC development services for about 10 years now. In fact, we're certified Microsoft Gold Partner. So, based on our knowledge, experience, and expertise, there are two main benefits of using ASP.NET Core razor pages instead of MVC.

#1—Razor Pages is Better Organized

If you've ever used MVC for any kind of web development, then you probably know that how much time it takes to code an entire app. Creating dynamic routes, naming things properly, and hundred other stuff consumes a lot of time.

Razor Pages, on the other hand, is more organized compared to MVC.

In Razor Pages, the files are basically more organized. You have a Razor View and the entire code behind a file, same way the old ASP.NET WebForms did.

#2—Single Responsibility

Again, if you have ever used an MVC framework before, you have probably seen that there are some huge controller classes which are generally filled with a lot of various actions. These classes are like a virus which grows bigger and bigger as new things are added.

But, in Razor Pages, each app page is self-contained with its own view and code organized together, which as a results, is less complex than the MVC.

Overall, ASP.NET Core is a Modular Web Framework.

In MVC, if you add new things, then the .NET framework will force you to release a new version.

For example, Microsoft released *routing* in MVC 4, and later, they released *Attribute routing* for which they again had to release another new framework MVC 5.

In ASP.NET Core, on the other hand, everything is managed using the NuGet package, which means it is easier than MVC to upgrade existing framework without releasing new .net framework version every time new things are added.

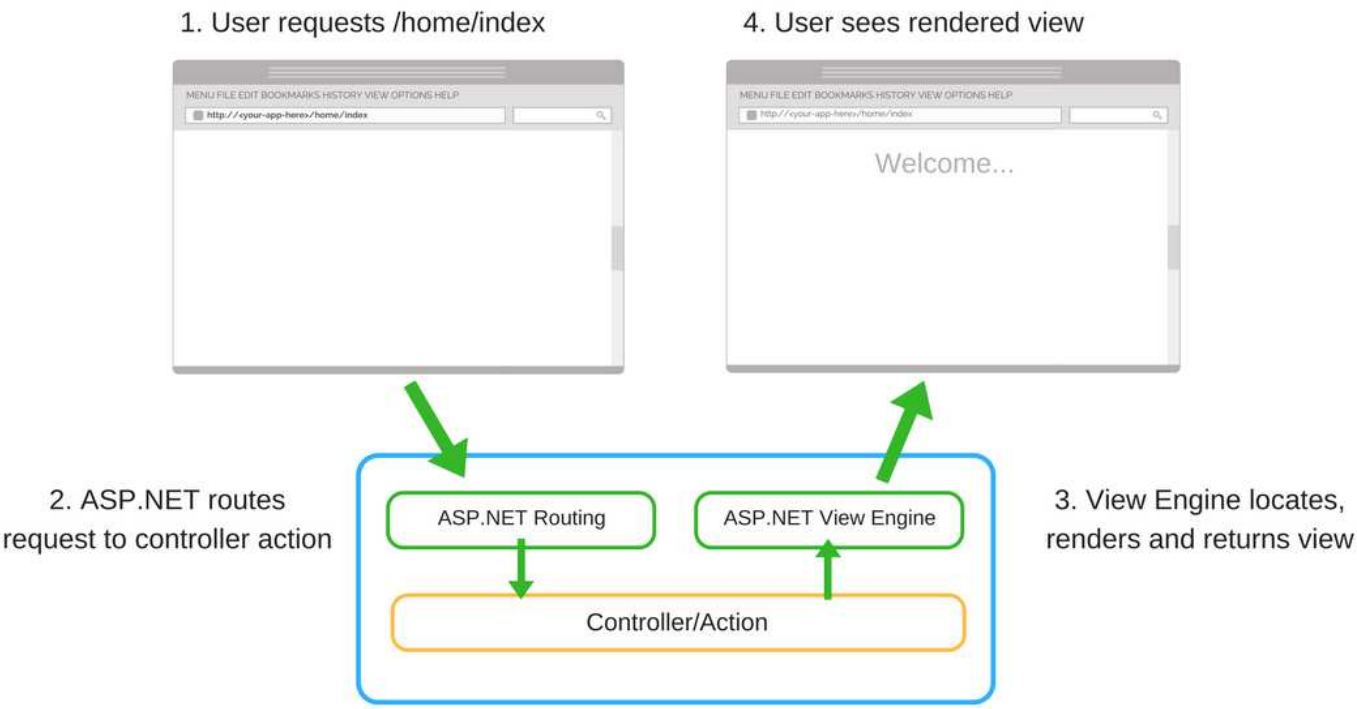
Additionally, In .NET Core, any community can release an update for new NuGet package version, and you can receive the latest changes by just updating your NuGet packages.

A Quick Comparison of How Requests Are Handled in Both

We explained in above points that building a web application using ASP.NET Core Razor Pages is less complex than the MVC. Here, we will demonstrate that with action.

Let's start with MVC

Here's a quick overview of how MVC handles the requests.



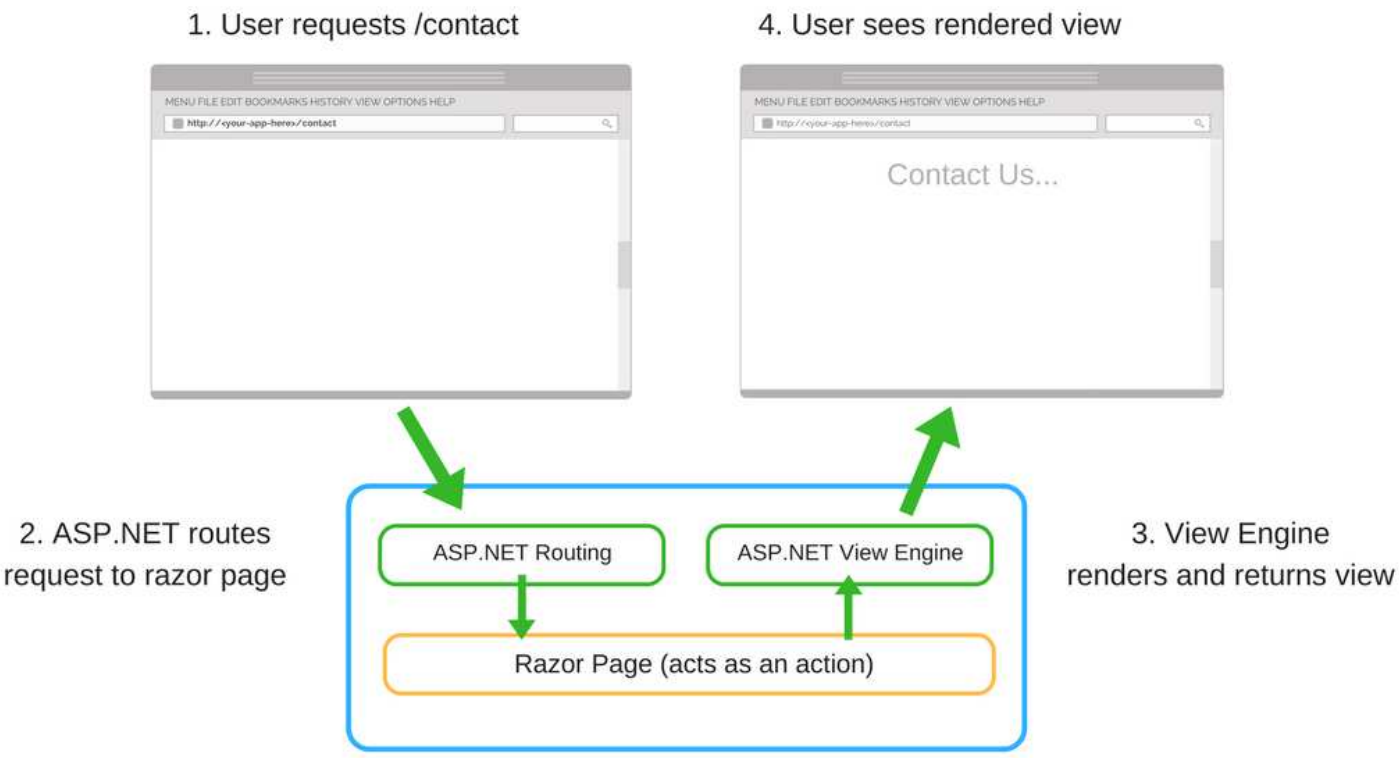
As you can see, routing is the key to how MVC decides to handle requests. The default configuration for routing is the combination of action and controller names.

So if you request for /staff/index, then it will route you the action named Index on the StaffController class.

But, it can be customized or configured to route any request to any controller with a block of code.

Now Compare the Same with Razor Pages

Here's a quick overview of how Razor Pages handle the requests.



The difference between the two is that in Razor Pages, when you make a request, the default routing configuration will find a Razor Page for that specific request in the Pages folder.

Suppose you make a request for /contact/, then ASP.NET Core will look for a page having same name that you used in request and will route you directly to it.

That means, a request to /contact/ will route you to Contact.cshtml

And for any .cshtml file to be considered as a Razor Page, it must be placed in the Pages folder and also contain @Page in its markup.

This way, the Razor Page will then act as a controller action. Now comparing this to MVC, the configuring the custom route will be less complex as there will no extra coding involved.

In Conclusion

Razor Pages seem like a promising start for modern web app development with less complexity in 2018. And as the comparison has shown that it provides the benefit of containing everything related to a particular request at one place, where in MVC requires spreading pieces

all around your app like a giant puzzle which you will then have to put back together with extra efforts of coding.

. . .

If you've any question, we welcome them all in the comments, and if you've like this article, show some love by clapping for it.