MICROSOFT ARCHITECT
By Joydip Kanjilal, Columnist, InfoWorld
APR 16, 2018 3:00 AM PDT

How to use response caching middleware in ASP.Net Core

Take advantage of response caching in ASP.Net Core to improve your application's performance by reducing the load on the web server

Microsoft's ASP.Net Core has already become a popular way to build high-performance, modern web applications that can run on Windows, Linux, or MacOS. One way it supports high performance of course is caching. Although ASP.Net Core doesn't have an in-built Cache object, it provides support for several different types of caching including in-memory caching, distributed caching, and response caching.

In previous articles I discussed how to use in-memory caching in ASP.Net Core and how to implement a distributed cache in ASP.Net Core. In this post, I will explain response caching and its benefits and then examine how we can work with the response caching middleware provided in ASP.Net Core.

[Get started with Visual Studio Code, Microsoft's lightweight editor for Windows, MacOS, and Linux. • Learn what's new in the latest version of Visual Studio Code. | Keep up with hot topics in programming with InfoWorld's App Dev Report newsletter.]

Response caching explained

Response caching refers to the ability to cache web server responses using cache-related headers in the HTTP response objects. Such headers are used to specify how to cache the responses for either all requests or a few selected requests. Note that unlike output caching, response caching in ASP.Net Core doesn't cache the responses in the memory of the web server.

Response caching in ASP.Net Core is actually a better and extensible form of output caching. It is used to inform the web browser to cache content by specifying cache-related headers on HTTP responses. This can significantly reduce the number of requests a client makes to the web server, and significantly reduce latency, as subsequent requests can be served from the client's cache. It should be noted that response caching uses memory to cache the data by default, but you can even configure custom storage providers if need be.

In the section that follows, we will learn to use and configure the response caching middleware in ASP.Net Core.

Create an ASP.Net Core application in Visual Studio

Assuming that .Net Core 2.0 is installed in your system, follow these steps to create an ASP.Net Core application in Visual Studio 2017.

1. Open Visual Studio and click on File > New > Project.
2. Select "ASP.Net Core Web Application (.Net Core)" from the list of the templates displayed
3. Specify a name for the project
4. Click OK to save
5. In the "New .Net Core Web Application..." window, select "Web API"
6. Ensure that "Enable Docker Support" is unchecked

7. Select “No Authentication” as we won’t be using authentication in this example
8. Click OK

This will create a new ASP.Net Core project with an example Controller to build and execute RESTful HTTP services.

The next step is to include the necessary package to use response caching middleware. To do this, select the project you just created in the Solution Explorer Window and add a reference to the Microsoft.AspNetCore.ResponseCaching package via the NuGet Package Manager.

Configure response caching middleware in ASP.Net Core

Next, add the middleware to the IServicesCollection in the ConfigureServices method as shown in the code snippet below.

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddResponseCaching();
    services.AddMvc();
}
```

To add the middleware to the request processing pipeline, you can take advantage of the UseResponseCaching extension method as shown in the following code snippet.

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory logge
{
    //Other code
    app.UseResponseCaching();
}
```

Response caching options in ASP.Net Core

The response caching middleware in ASP.Net Core provides three options to control how response caching works. These include the following:

1. SizeLimit — Used to specify the maximum size of the response cache. The default value is 100 MB.
2. UseCaseSensitivePaths — Used to specify if the responses are cached on case-sensitive paths.
3. MaximumBodySize — Used to specify the largest cacheable size of the response body. The default value is 64 MB.

The following code snippet shows how response caching can be controlled using one or more of the above options in the ConfigureServices method.

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddResponseCaching(options =>
     {
          options.UseCaseSensitivePaths = true;
          options.MaximumBodySize = 1024;
     });
    services.AddMvc();
}
```

Using the ResponseCache attribute in controller methods in ASP.Net Core

Now that the middleware is configured, you can use the [ResponseCache] attribute on your controller methods to set the appropriate HTTP headers to cache responses. You can use the following parameters with this attribute.

- Duration — Used to specify the maximum duration (in seconds) for which the response should be cached.
- Location — Used to specify the location where the response should be cached. This can have any of these values: Any, Client, or None. The default location value is Any.
- NoStore — Used to specify whether the data should be stored or not.
- CacheProfileName — Used to specify the name of the cache profile.
- VaryByHeader — Used to specify the Vary response header.
- VaryByQueryKeys — Used to specify the query string parameters based on which responses should be cached.

The following code snippet illustrates how you can use the ResponseCache attribute in your controller's action method.

```
[ResponseCache(Duration = 30)]
public IActionResult HomePage()
{
    ViewData["Message"] = "The current time is:" + DateTime.Now.ToString();
    return View();
}
```

When you run the application, you should be able to see the current DateTime of your system displayed. The response will be cached for 30 seconds as specified in the ResponseCache attribute.

It should be noted that the response caching middleware only caches the server responses that have a HTTP Status code 200. Other responses including errors and error pages are simply ignored.

Take a look at this MSDN article for more information on response caching middleware in ASP.Net Core.

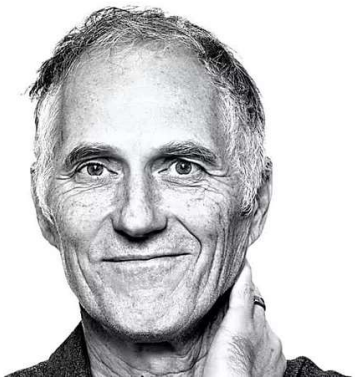
Joydip Kanjilal is a Microsoft MVP in ASP.Net, as well as a speaker and author of several books and articles. He has more than 20 years of experience in IT including more than 16 years in Microsoft .Net and related technologies.

Follow

- Stay up to date with InfoWorld's newsletters for software developers, analysts, database programmers, and data scientists.
- Get expert insights from our member-only Insider articles.

YOU MAY ALSO LIKE

Are industry clouds an opportunity or a distraction?



Tim O'Reilly: the golden age of the programmer is over

JDK 17: The new features in Java 17

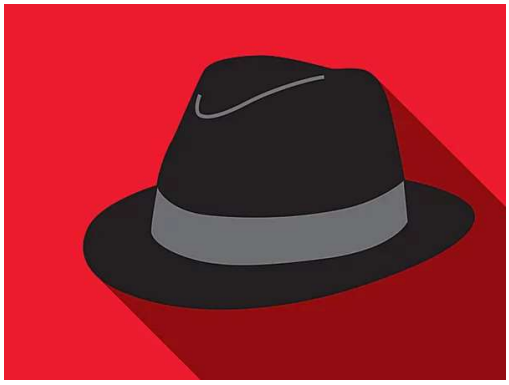
When to use Task.WaitAll vs. Task.WhenAll in .NET

LLVM 12 arrives with x86, AArch optimizations

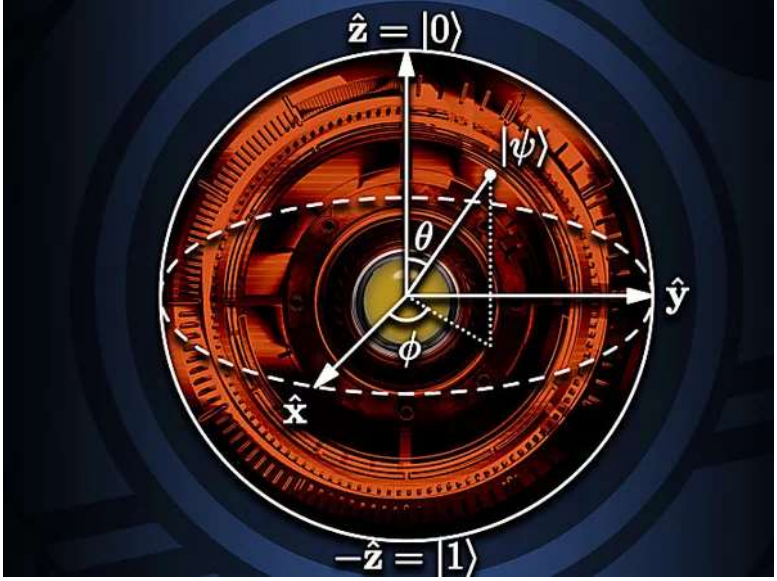
JetBrains takes TeamCity CI/CD to the cloud

Edge computing archetypes are emerging, and they are

6 reasons to switch to managed Kubernetes



Red Hat OpenShift ramps up security and manageability



Amazon Braket: Get started with quantum computing

6 ways Alibaba Cloud challenges AWS, Azure, and GCP