MICROSOFT ARCHITECT

By Joydip Kanjilal, Columnist, InfoWorld

FEB 22, 2021 3:00 AM PST

How to use LazyCache in ASP.NET Core MVC 5

Take advantage of LazyCache to improve the performance and scalability of your ASP.NET 5 Core applications in heavy load scenarios.

Microsoft's ASP.NET Core has become a popular way to build high-performance, modern web applications that can run on Windows, Linux, or MacOS. An important way to ensure high performance and reliability in applications that experience high volumes of requests is caching frequently used data.

LazyCache is a simple in-memory caching service that is both easy to use and thread safe. "Lazy" refers to the fact that LazyCache will never execute your cacheable delegates more than once for each "cache miss," i.e., whenever the data requested is not found in the cache. In other words, laziness reduces computational overhead.

[Also on InfoWorld: The most valuable software developer skills to get hired now]

This article talks about how we can work with LazyCache in ASP.NET Core 5.0. To work with the code examples illustrated in this article, you should have Visual Studio 2019 installed in your system. If you don't already have a copy, you can download Visual Studio 2019 here.

Create an ASP.NET Core MVC 5 project in Visual Studio 2019

First off, let's create an ASP.NET Core 5 project in Visual Studio 2019. Following these steps should create a new ASP.NET Core MVC 5 project in Visual Studio 2019.

1. Launch the Visual Studio IDE.
2. Click on "Create new project."
3. In the "Create new project" window, select "ASP.NET Core Web App (Model-View-Controller)" from the list of templates displayed.
4. Click Next.
5. In the "Configure your new project" window, specify the name and location for the new project.
6. Optionally check the "Place solution and project in the same directory" check box, depending on your preferences.
7. Click Next.
8. In the "Additional Information" window shown next, select .NET 5.0 as the target framework from the drop-down list at the top. Leave the "Authentication Type" set as None (default).
9. Ensure that the check boxes "Enable Docker," "Configure for HTTPS," and "Enable Razor runtime compilation" are unchecked as we won't be using any of those features here.
10. Click Create.

Following the above steps will create a new ASP.NET Core MVC 5 project. We'll use this project in the subsequent sections in this article.

Install LazyCache in ASP.NET Core MVC 5

To work with LazyCache in ASP.NET Core MVC 5.0, you should install the following two packages into your project:

- LazyCache
- LazyCache.AspNetCore

Both LazyCache and LazyCache.AspNetCore libraries are available as NuGet packages. You can install these packages either from the NuGet Package Manager or by using the following commands at the NuGet Package Manager Console window.


```
PM> Install-Package LazyCache
PM> Install-Package LazyCache.AspNetCore
```

What is caching? Why is caching needed?

Caching is a state management strategy that is often used in web applications to store relatively stale data in the memory for later reuse. Caching improves the application's performance by enabling the application to read the data from the memory instead of from disk — accessing memory is orders of magnitude faster than accessing the disk.

Although ASP.NET Core lacks a built-in Cache object, it provides support for several different types of caching including in-memory caching, distributed caching, and response caching.

What is LazyCache? Why should we use it?

LazyCache is an open-source, simple, thread-safe, extensible caching service with a developer-friendly API. Under the hood, LazyCache takes advantage of MemoryCache pertaining to the Microsoft.Extensions.Caching namespace and uses lazy locking to ensure the delegate only gets executed once.

LazyCache is a good choice for caching database calls, complex object graphs, and web service calls. Although you can store items in the cache for a shorter or longer duration, the default duration supported is 20 minutes.

Here is a list of the benefits of LazyCache at a quick glance:

- Extensible
- Open source
- Developer-friendly API
- Support for built-in lazy locking
- Uses MemoryCache under the hood

Configure dependency injection for LazyCache in ASP.NET Core MVC 5

You should call the AddLazyCache() method on the IServiceCollection instance in your ConfigureServices method as shown in the code snippet given below.

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddLazyCache();
    services.AddControllersWithViews();
}
```

This will make sure that you access LazyCache throughout your application.

Access to LazyCache is provided by the IAppCache interface. IAppCache represents the LazyCache service and provides a GetOrAddAsync method that accepts the following:

- A cache key that uniquely identifies the cache entry
- A factory that can be used to retrieve the data that is to be cached as Func<ICacheEntry, Task> addItemFactory
- A duration that specifies the amount of time the data should persist in the cache

Use dependency injection to inject the IAppCache instance in ASP.NET Core MVC 5

You should take advantage of constructor injection to inject the IAppCache instance as shown in the code snippet given below.

```
public class HomeController : Controller
{
  private readonly ILogger<HomeController> _logger;
  private readonly IAppCache _lazyCache = new CachingService();
  public HomeController(ILogger<HomeController> logger, IAppCache cache)
   {
     _logger = logger;
     _lazyCache = cache;
   }
}
```

You can now use the LazyCache instance to add and retrieve data to and from the cache.

Add or retrieve data to or from LazyCache in ASP.NET Core MVC 5

Consider the following method that returns a list of strings.

```
private async Task<List<string>> GetData()
{
    return new List<string>()
    {
       "Joydip Kanjilal",
       "Steve Smith",
       "Rick Smith"
    };
}
```

You can use the following code to retrieve data from the cache or add data to the cache if it isn't present.

```
var data = await _lazyCache.GetOrAddAsync("Authors", GetData, DateTimeOffset.Now.AddMinu
```

The GetOrAddAsync() extension method pertaining to the LazyCache library provides an easy and elegant way to implement caching in your applications. It takes advantage of a factory delegate and generics to add cached method calls to your code. Using this method you can get cached data when requested by the application or store data to the cache if the piece of data is not available in the cache.

If you would like to store more data in memory and you want a more advanced caching service, you could take advantage of Redis for distributed caching. The best part is that because we're using IAppCache in our application, you can change the underlying caching provider easily.

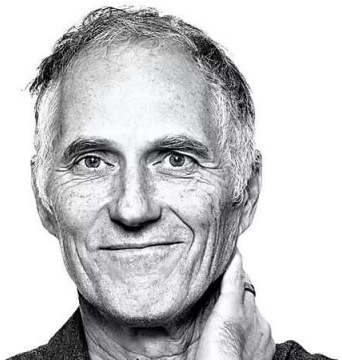
How to do more in ASP.NET Core 5:

- How to create PDF documents in ASP.NET Core 5
- How to use immutable objects in ASP.NET Core MVC 5
- Dependency injection best practices for ASP.NET Core MVC 5
- How to use security headers in ASP.NET Core MVC 5
- How to handle unknown actions in ASP.NET Core MVC 5
- How to overload action methods in ASP.NET Core MVC 5

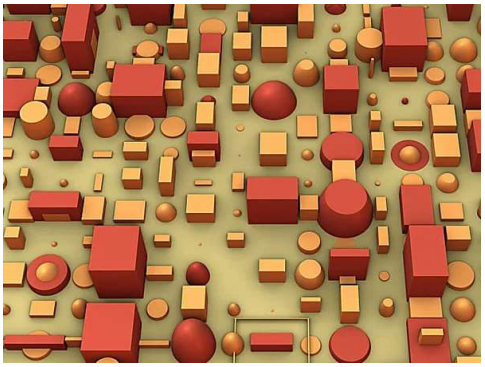
Joydip Kanjilal is a Microsoft MVP in ASP.Net, as well as a speaker and author of several books and articles. He has more than 20 years of experience in IT including more than 16 years in Microsoft .Net and related technologies.

Follow

YOU MAY ALSO LIKE



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



Running microservices on Google Cloud Platform

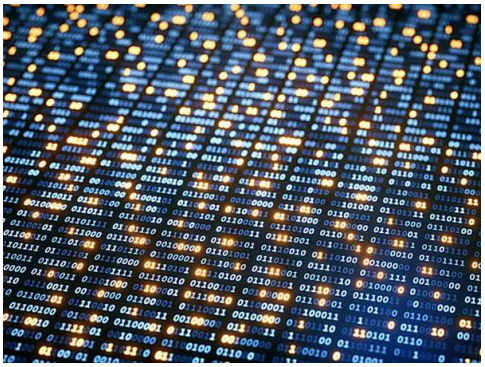
Are industry clouds an opportunity or a distraction?

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

What's new in Kubernetes 1.20

JDK 17: The new features in Java 17

The decline of Heroku



What is a computational storage drive? Much-

Red Hat Enterprise Linux takes aim at edge

How IT priorities are shifting during the COVID-19 crisis

The shifting market for PostgreSQL