Implement HTTP call retries with exponential backoff with IHttpClientFactory and Polly policies

Article • 03/01/2023

Tip

This content is an excerpt from the eBook, .NET Microservices Architecture for Containerized .NET Applications, available on .NET Docs or as a free downloadable PDF that can be read offline.

Download PDF

Microsoft
.NET Microservices: Architecture for Containerized .NET Applications

The recommended approach for retries with exponential backoff is to take advantage of more advanced .NET libraries like the open-source Polly library .

Polly is a .NET library that provides resilience and transient-fault handling capabilities. You can implement those capabilities by applying Polly policies such as Retry, Circuit Breaker, Bulkhead Isolation, Timeout, and Fallback. Polly targets .NET Framework 4.x and .NET Standard 1.0, 1.1, and 2.0 (which supports .NET Core and later).

The following steps show how you can use Http retries with Polly integrated into `IHttpClientFactory`, which is explained in the previous section.

Reference the .NET 7 packages

`IHttpClientFactory` is available since .NET Core 2.1, however, we recommend you use the latest .NET 7 packages from NuGet in your project. You typically also need to reference the extension package `Microsoft.Extensions.Http.Polly`.

Configure a client with Polly's Retry policy, in app startup

As shown in previous sections, you need to define a named or typed client HttpClient configuration in your standard *Program.cs* app configuration. Now you add incremental code specifying the policy for the Http retries with exponential backoff, as below:

C#

```
// Program.cs
builder.Services.AddHttpClient<IBasketService, BasketService>()
        .SetHandlerLifetime(TimeSpan.FromMinutes(5))  //Set lifetime to five
minutes
        .AddPolicyHandler(GetRetryPolicy());
```

The **AddPolicyHandler()** method is what adds policies to the `HttpClient` objects you'll use. In this case, it's adding a Polly's policy for Http Retries with exponential backoff.

To have a more modular approach, the Http Retry Policy can be defined in a separate method within the *Program.cs* file, as shown in the following code:

C#

```
static IAsyncPolicy<HttpResponseMessage> GetRetryPolicy()
{
    return HttpPolicyExtensions
        .HandleTransientHttpError()
        .OrResult(msg => msg.StatusCode ==
System.Net.HttpStatusCode.NotFound)
        .WaitAndRetryAsync(6, retryAttempt =>
TimeSpan.FromSeconds(Math.Pow(2,

retryAttempt)));
}
```

With Polly, you can define a Retry policy with the number of retries, the exponential backoff configuration, and the actions to take when there's an HTTP exception, such as logging the error. In this case, the policy is configured to try six times with an exponential retry, starting at two seconds.

Add a jitter strategy to the retry policy

A regular Retry policy can affect your system in cases of high concurrency and scalability and under high contention. To overcome peaks of similar retries coming from many clients in partial outages, a good workaround is to add a jitter strategy to the retry algorithm/policy. This strategy can improve the overall performance of the end-to-end system. As recommended in Polly: Retry with Jitter , a good jitter strategy can be implemented by smooth and evenly distributed retry intervals applied with a well-controlled median initial retry delay on an exponential backoff. This approach helps to spread out the spikes when the issue arises. The principle is illustrated by the following example:

C#

```csharp
var delay = Backoff.DecorrelatedJitterBackoffV2(medianFirstRetryDelay:
TimeSpan.FromSeconds(1), retryCount: 5);

var retryPolicy = Policy
    .Handle<FooException>()
    .WaitAndRetryAsync(delay);
```

Additional resources

- **Retry pattern**
  https://learn.microsoft.com/azure/architecture/patterns/retry

- **Polly and IHttpClientFactory**
  https://github.com/App-vNext/Polly/wiki/Polly-and-HttpClientFactory

- **Polly (.NET resilience and transient-fault-handling library)**
  https://github.com/App-vNext/Polly

- **Polly: Retry with Jitter**
  https://github.com/App-vNext/Polly/wiki/Retry-with-jitter

- **Marc Brooker. Jitter: Making Things Better With Randomness**
  https://brooker.co.za/blog/2015/03/21/backoff.html

Previous Next