

第24期-可复原性与瞬态故障处理机制

2020年4月22日 15:41

可复原性，容错能力，是指系统能够在发生故障后自我恢复正常的能力。基于云，基于微服务或物联网的应用程序通常依赖于不可靠的网络与其他系统进行通信。

由于诸如网络问题和超时之类的瞬时故障，或者子系统处于脱机状态，负载状态或其他情况下无响应，因此此类系统可能不可用或无法访问。

Transient fault handling and proactive resilience engineering

复原能力与容错性设计

重试机制。 这种技术有助于通过执行特定次数的调用重试避免短时间的间歇性故障，以防仅短时间内服务不可用。此问题可能是因为间歇性网络问题或微服务/容器移动到群集中的不同节点造成的。但是，如果这些重试没有针对断路器恰当设计，可能加剧连锁反应，最终导致拒绝服务 (DoS)，常用的重试算法采用使用指数回退算法和抖动算法。

超时机制。 通常情况下，客户端应设计为不会无限期阻止，且应在等待响应时使用超时。使用超时可确保永远不会无限期地占用资源。

断路器模式。 在此方法中，客户端进程跟踪失败请求数。如果错误率超出配置的限制，“断路器”跳变，使进一步尝试立即失败。（如果大量请求失败，则表明服务不可用，发送请求无意义。）超时期限过后，客户端应重试，如果新的请求成功，关闭断路器。

提供回退。 在此方法中，客户端进程在请求失败时执行回退逻辑，如返回缓存数据或默认值。这是适用于查询的一种方法，对于更新或命令较复杂。

限制请求数。 客户端还应对客户端微服务可发送到特定服务的未完成请求的数量设置上限。如果已达到限制，发出其他请求可能无意义，这些尝试应立即失败。对于实现，Polly Bulkhead 隔离策略可以用于满足此要求。这种方法实质上是将 SemaphoreSlim 作为实现的平行化限制。它也允许 bulkhead 外的“队列”。可在执行前主动舍弃额外的负载（例如，由于容量被视为已满）。这样一来，其对某些失败情景的响应速度比断路器要快，因为断路器会等待失败。Polly 中的 BulkheadPolicy 对象公开 bulkhead 和队列的已满程度，且提供有关溢出的事件，因此也可用于驱动自动水平缩放。

限流限次与计费机制。 ASP.NET CoreRateLimit是一个ASP.NET Core速率限制解决方案，旨在控制客户端根据IP地址或客户端ID向Web API或MVC应用发出的请求的速率。该ASP.NET CoreRateLimit包包含一个IpRateLimitMiddleware和ClientRateLimitMiddleware，每个中间件可以设定不同的场景多像限制允许IP或客户拨打电话的最大数目的时间间隔像每秒，15分钟等，您可以定义这些限制可满足对API的所有请求，也可以将限制范围限制在每个

API URL或HTTP动词和路径上。

[ASP.NET Core rate limiting middleware](#)

使用 Polly 开源库

Polly 是一个 .NET 弹性和瞬态故障处理库，允许开发人员以 Fluent 和线程安全的方式来实现重试、断路、超时、隔离、缓存和回退策略。

<https://github.com/App-vNext/Polly>

<https://github.com/App-vNext/Polly/wiki>

<https://github.com/App-vNext/Polly-Samples>

<https://github.com/dotnet/extensions/tree/master/src/HttpClientFactory/Polly>

<https://github.com/App-vNext/Polly/wiki/Polly-and-HttpClientFactory>

<https://github.com/App-vNext/Polly.Extensions.Http>

eShop 中的瞬态故障处理应用

使用 Polly 的重试策略和断路策略在 Startup 中配置 HttpClient 客户端

```
//ConfigureServices() - Startup.cs
services.AddHttpClient<IBasketService, BasketService>()
    .SetHandlerLifetime(TimeSpan.FromMinutes(5)) //Sample. Default
lifetime is 2 minutes
    .AddHttpMessageHandler<HttpClientAuthorizationDelegatingHandler>()
    .AddPolicyHandler(GetRetryPolicy())
    .AddPolicyHandler(GetCircuitBreakerPolicy());

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

static IAsyncPolicy<HttpResponseMessage> GetCircuitBreakerPolicy()
{

```

```

        return HttpPolicyExtensions
            .HandleTransientHttpError()
            .CircuitBreakerAsync(5, TimeSpan.FromSeconds(30));
    }

```

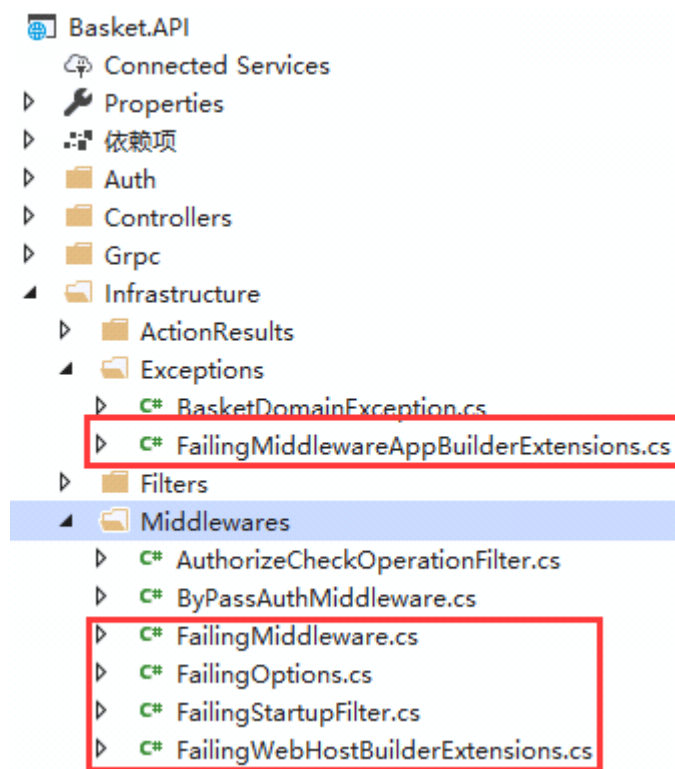
将抖动策略添加到重试策略

```

Random jitterer = new Random();
var retryWithJitterPolicy = HttpPolicyExtensions
    .HandleTransientHttpError()
    .OrResult(msg => msg.StatusCode == System.Net.HttpStatusCode.NotFound)
    .WaitAndRetryAsync(6, // exponential back-off plus some jitter
        retryAttempt => TimeSpan.FromSeconds(Math.Pow(2, retryAttempt))
            +
            TimeSpan.FromMilliseconds(jitterer.Next(0, 100))
    );

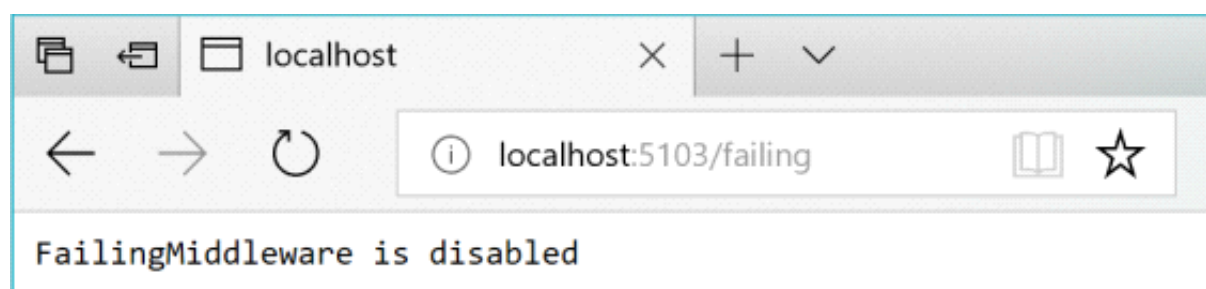
```

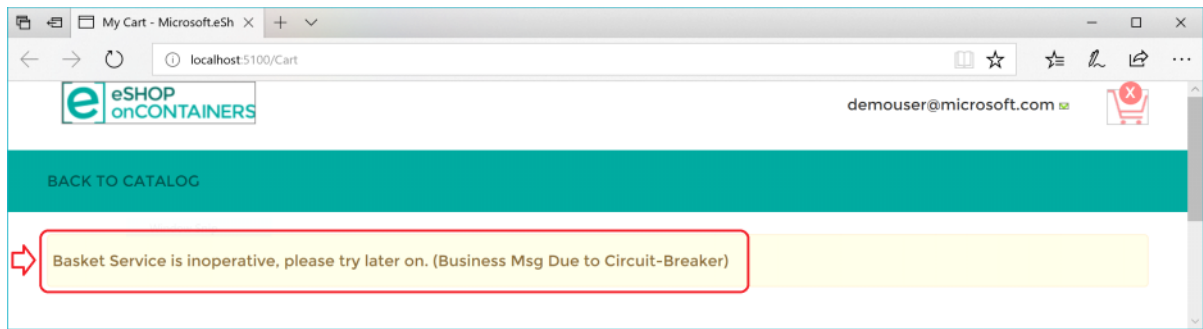
在 eShopOnContainers 模拟测试断路器



<http://localhost:5103/failing?enable>

<http://localhost:5103/failing?disable>





参考资料

[Liam Wang .NET 开源项目 Polly 介绍](#)

[腾飞: 开源服务容错处理库Polly使用文档](#)

[微软: 微服务中的复原和高可用性](#)

[微软: 实现可复原的应用程序](#)

[Azure. 复原模式](#)

[Azure: 复原能力与容错设计](#)