Bulletproof Transient Error Handling with Polly

Carl Franklin
App vNext


AppvNext
create. combine. connect.

Transient Errors

Network outages

Service outages

Denial of Service attacks

IO locks

Connected device failures

Polly to the rescue!

.NET 3.5 / 4.0 / 4.5 / PCL library

Fluently express transient exception handling policies:

Retry, Retry Forever, Wait and Retry, Circuit Breaker

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

Nuget: Install-Package Polly

We've made 10 new feature releases in 4 months!

Polly solves two different problems …

Retry … ‘Maybe it’s just a blip’

Circuit Breaker … ‘That system is down / struggling’

Use both nested, for double protection!

History of Polly

2013 Michael Wolfenden invents Polly

2014 Polly v2.0. Targets multi .NET versions including PCL.

2015 Scott Hanselman (Microsoft) recommends Polly!
Thoughtworks (Martin Fowler et al) recommend Polly!

Nov 2015 App-vNext take stewardship

Dec 2015 Polly v3.0. Full async support

2016 Polly v4.0. Circuit-breaker health reporting. Advanced circuit breaker.

Retry Patterns

Retry if app can continue (async writes, for example). Specify number of retries.

Wait and Retry Retry with a timeout in between each try. Change the timeout between each retry, eg exponential back-off.

Retry Forever if app cannot continue until call is made successfully.

Retry addresses ... ‘It’s probably a blip. Give it another go - it might succeed.’

Circuit Breaker

Circuit Breaker Breaks the circuit for a configured period if too many errors occur. Breaking the circuit prevents calls; breaker fails fast instead.

Circuit Breaker addresses … ‘Whoa, that system is struggling / down. Give it a break. And don’t hang around waiting for an answer that’s unlikely - fail fast!’

Step 1: Define Policy

```
var retryPolicy = Policy
    .Handle<EndpointNotFoundException>()
    .RetryForeverAsync();
```

Step 2: Execute with Policy

```
var response =
    await retryPolicy.ExecuteAsync(() => DoSomething());
```

Policy Basics

Define how transient exceptions should be handled

Fluent and concise

Reusable

Thread-safe

Decorate any `Action` or `Func`

Can be chained together

Sync and async

Demos

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

Further features

Handle multiple exception types in one policy; filter exceptions handled:

```
var policy = Policy.Handle<SqlException>(ex => ex.Number == 1205)
    .Or<TimeoutException>();
```

Register delegates (onRetry, onBreak) to capture policy events, eg for logging.

Future roadmap?

.NET core compatible (99% there)

Bulkhead isolation pattern (isolates problems; prevents catastrophic failure)

Configure fallbacks for failing actions

Respond to return values, not just exceptions

Emit health stats

App vNext Polly team

Carl Franklin .NET Rocks, Music to Code By

Joel Hulen NASA Jet Propulsion Laboratory (JPL)

Dylan Reisenberger coder and travel photographer. Working in the .NET /microservices, special interest in fault-handling.

… and all you folks out there who want to make open-source contributions…

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

Polly Wiki

Extended documentation

Patterns

Git Workflow

Future Roadmap

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

THANK YOU!

Carl Franklin
carl@appvnext.com