Polly: Eine "Resilience" Bibliothek

(Kategorie: nuetzliche NuGet Pakete)

Patrick Drechsler
Redheads Ltd.

2020-06-22

Resilience deutsch

All | News | Images | Videos | Shopping | More | Settings | Tools

About 34.700.000 results (0,31 seconds)

English – detected ⇄ German

Resilience

Elastizität

Translations of **resilience**

noun

die Elastizität
elasticity, resilience, resiliency, flexibility, suppleness, springiness

die Spannkraft
resilience, tone, vigor, vigour, tonus, tension

die Unverwüstlichkeit
resilience, resiliency, robustness

RESILIENCE

Fehlerhaftes Verhalten von anderen/externen Diensten als "normal" betrachten

> *Polly is a .NET **resilience** and transient-fault-handling library that **allows developers to express policies such as Retry, Circuit Breaker, Timeout**, Bulkhead Isolation, and Fallback **in a fluent and thread-safe manner**. Polly targets .NET 4.0, .NET 4.5 and .NET Standard 1.1.*

POLLY PROJECT

- stabiles, gepflegtes Repo
- grosse Community
- https://github.com/App-vNext/Polly
- http://www.thepollyproject.org

WAS SIND POLICIES?

Policies beschreiben das Verhalten, wenn was schief geht

BEISPIEL "POLICIES"

- **Timeout**
  - "Antwort dauert länger als 10sec"
- **Retry**
  - "keine Antwort -> nochmal probieren"
- **Circuit Breaker**
  - "wenn... -> verhindere weiter Anfragen, damit sich der Dienst erholen kann"
- ...

BEISPIEL 1: RETRY

```
public class BusinessLogic
{
  private readonly IFlakyService _srv;

  public BusinessLogic(IFlakyService service) ⇒ _srv = service;

  public int CallFlakyMethod()
  {
    return _srv.SlowMethod(); // ← SLOW!!
  }
}
```

```
public int CallFlakyMethod()
{
  return _srv.SlowMethod();
}
```

Policy.Execute(...)

```
public int CallFlakyMethod()
{
  return _policy.Execute(() ⇒ _srv.SlowMethod());
}
```

Policy.ExecuteAndContain(...)

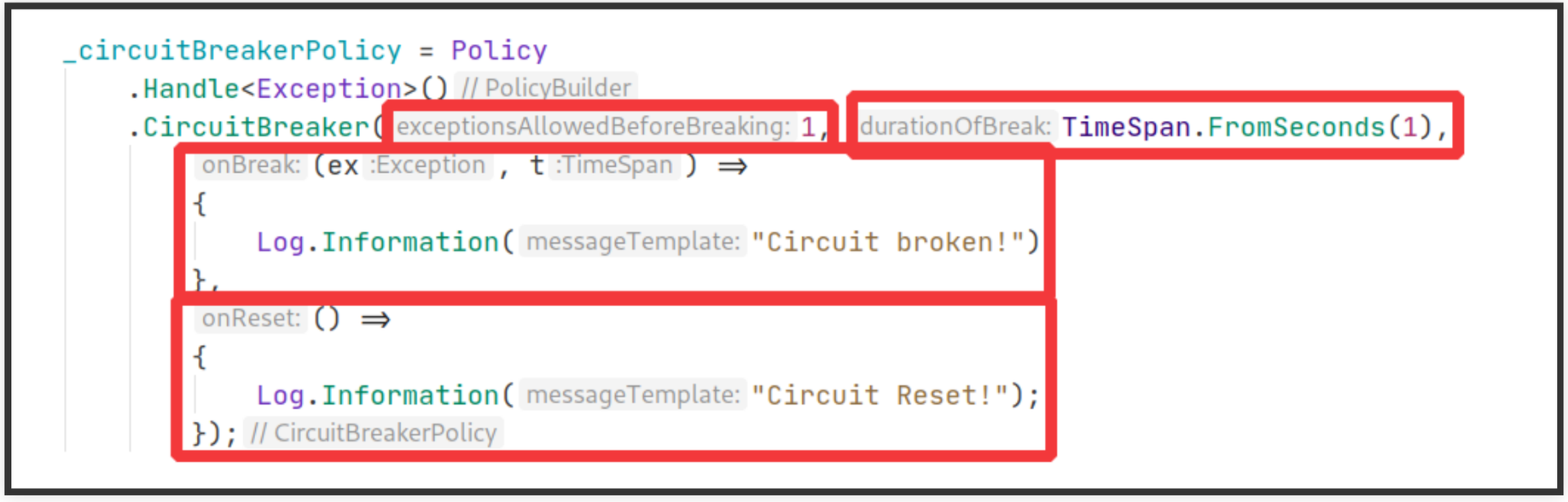
```
public int CallFlakyMethod()
{
  var policyResult =
    _policy.ExecuteAndContain(() ⇒ _srv.SlowMethod());

  // TODO
}
```

Policies koennen sehr feingranular definiert werden, z.B.:

- erst: 3x Retry
- dann: 2x Retry mit logarithmischen Abstaenden (in 2min, in 20min, etc)
- dann: Circuit Braker
- dann: Failover

BEISPIEL 2: CIRCUIT BREAKER

```
var circuitBreakerPolicy = Policy
  .Handle<Exception>()
  .CircuitBreaker(1, TimeSpan.FromSeconds(1),
      (ex, t) ⇒
      {
          Log.Information("Circuit broken!");
      },
      () ⇒
      {
          Log.Information("Circuit Reset!");
      });
```


_circuitBreakerPolicy = Policy
.Handle<Exception>() // PolicyBuilder
.CircuitBreaker(exceptionsAllowedBeforeBreaking: 1, durationOfBreak: TimeSpan.FromSeconds(1),
onBreak: (ex :Exception , t :TimeSpan) ⇒
{
Log.Information(messageTemplate: "Circuit broken!")
},
onReset: () ⇒
{
Log.Information(messageTemplate: "Circuit Reset!");
}); // CircuitBreakerPolicy