Spring Cloud Circuit Breaker 3.0.1

OVERVIEW LEARN SUPPORT SAMPLES

Introduction

Spring Cloud Circuit breaker provides an abstraction across different circuit breaker implementations. It provides a consistent API to use in your applications allowing you the developer to choose the circuit breaker implementation that best fits your needs for your app.

Supported Implementations

- Resilience4J
- Spring Retry

Core Concepts

To create a circuit breaker in your code you can use the `CircuitBreakerFactory` API. When you include a Spring Cloud Circuit Breaker starter on your classpath a bean implementing this API will automatically be created for you. A very simple example of using this API is given below

COPY

```
@Service
public static class DemoControllerService {
        private RestTemplate rest;
        private CircuitBreakerFactory cbFactory;

        public DemoControllerService(RestTemplate rest, CircuitBreakerFactory cbF
                this.rest = rest;
                this.cbFactory = cbFactory;
        }

        public String slow() {
                return cbFactory.create("slow").run(() -> rest.getForObject("/slo
        }

}
```

The `CircuitBreakerFactory.create` API will create an instance of a class called `CircuitBreaker`. The `run` method takes a `Supplier` and a `Function`. The `Supplier` is the code that you are going to wrap in a circuit breaker. The `Function` is the fallback that will be executed if the circuit breaker is tripped. The function will be passed the `Throwable` that caused the fallback to be triggered. You can optionally exclude the fallback if you do not want to provide one.

Circuit Breakers In Reactive Code

If Project Reactor is on the class path then you can also use `ReactiveCircuitBreakerFactory` for your reactive code.

COPY

```
@Service
public static class DemoControllerService {
        private ReactiveCircuitBreakerFactory cbFactory;
        private WebClient webClient;


        public DemoControllerService(WebClient webClient, ReactiveCircuitBreakerI
                this.webClient = webClient;
                this.cbFactory = cbFactory;
        }

        public Mono<String> slow() {
                return webClient.get().uri("/slow").retrieve().bodyToMono(String
                it -> cbFactory.create("slow").run(it, throwable -> return Mono.
        }
}
```

The `ReactiveCircuitBreakerFactory.create` API will create an instance of a class called `ReactiveCircuitBreaker`. The `run` method takes with a `Mono` or `Flux` and wraps it in a circuit breaker. You can optionally profile a fallback `Function` which will be called if the circuit breaker is tripped and will be passed the `Throwable` that caused the failure.

Spring Boot Config

The following starters are available with the Spring Cloud BOM

- Resilience4J - `org.springframework.cloud:spring-cloud-starter-circuitbreaker-resilience4j`
- Reactive Resilience4J - `org.springframework.cloud:spring-cloud-starter-circuitbreaker-reactor-resilience4j`
- Spring Retry - `org.springframework.cloud:spring-cloud-starter-circuitbreaker-spring-retry`

Quickstart Your Project

Bootstrap your application with Spring Initializr.

Get ahead

VMware offers training and certification to turbo-charge your progress.

Learn more

Get support

Spring Runtime offers support and binaries for OpenJDK™, Spring, and Apache Tomcat® in one simple subscription.

Learn more

Upcoming events

Check out all the upcoming events in the Spring community.

View all