Spring Cloud Netflix Hystrix

服务短路（CircuitBreaker）

QPS: Query Per Second

TPS: Transaction Per Second

QPS: 经过全链路压测，计算单机极限QPS，集群 QPS = 单机 QPS * 集群机器数量 * **可靠性比率**

全链路压测 除了压 极限QPS，还有错误数量

全链路：一个完整的业务流程操作

JMeter：可调整型比较灵活

Spring Cloud Hystrix Client

官网:https://github.com/Netflix/Hystrix

> Reactive Java 框架：
>
> - Java 9 Flow API
> - Reactor
> - RxJava（Reactive X）

激活 Hystrix

通过 `@EnableHystrix` 激活

> Hystrix 配置信息wiki：https://github.com/Netflix/Hystrix/wiki/Configuration

Hystrix

注解方式（Annotation）

```
@RestController
public class HystrixDemoController {

    private final static Random random = new Random();

    /**
     * 当{@link #helloWorld} 方法调用超时或失败时，
     * fallback 方法{@link #errorContent()}作为替代返回
     *
     * @return
     */
    @GetMapping("hello-world")
    @HystrixCommand(
            fallbackMethod = "errorContent",
            commandProperties = {
                    @HystrixProperty(name =
"execution.isolation.thread.timeoutInMilliseconds",
                            value = "100")
            }

    )
    public String helloWorld() throws Exception {

        // 如果随机时间 大于 100 ，那么触发容错
        int value = random.nextInt(200);

        System.out.println("helloWorld() costs "+value+" ms.");

        Thread.sleep(value);

        return "Hello,World";
    }

    public String errorContent() {
        return "Fault";
    }

}
```

编程方式

```
    /**
     * 当{@link #helloWorld} 方法调用超时或失败时,
     * fallback 方法{@link #errorContent()}作为替代返回
     *
     * @return
     */
    @GetMapping("hello-world-2")
    public String helloWorld2() {
        return new HelloWorldCommand().execute();
    }

    /**
     * 编程方式
     */
    private class HelloWorldCommand extends
com.netflix.hystrix.HystrixCommand<String> {

        protected HelloWorldCommand() {
            super(HystrixCommandGroupKey.Factory.asKey("HelloWorld"),
                    100);


        }

        @Override
        protected String run() throws Exception {
            // 如果随机时间 大于 100 , 那么触发容错
            int value = random.nextInt(200);

            System.out.println("helloWorld() costs " + value + " ms.");

            Thread.sleep(value);

            return "Hello,World";
        }

        //容错执行
        protected String getFallback() {
            return HystrixDemoController.this.errorContent();
        }
    }
```

对比 其他 Java 执行方式：

Future

```
public class FutureDemo {

    public static void main(String[] args) {

        Random random = new Random();

        ExecutorService service = Executors.newFixedThreadPool(1);

        Future<String> future = service.submit(() -> { // 正常流程
            // 如果随机时间 大于 100 ，那么触发容错
            int value = random.nextInt(200);

            System.out.println("helloWorld() costs " + value + " ms.");

            Thread.sleep(value);

            return "Hello,World";
        });

        try {
            future.get(100, TimeUnit.MILLISECONDS);
        } catch (Exception e) {
            // 超时流程
            System.out.println("超时保护！");
        }

        service.shutdown();
    }
}
```

RxJava

```
public class RxJavaDemo {

    public static void main(String[] args) {

        Random random = new Random();

        Single.just("Hello,World") // just 发布数据
                .subscribeOn(Schedulers.immediate()) // 订阅的线程池 immediate = Thread.currentThread()
                .subscribe(new Observer<String>() {
```

```
                    @Override
                    public void onCompleted() { // 正常结束流程
                        System.out.println("执行结束！");
                    }
                    @Override
                    public void onError(Throwable e) { // 异常流程（结束）
                        System.out.println("熔断保护！");
                    }

                    @Override
                    public void onNext(String s) { // 数据消费 s = "Hello,World"
                        // 如果随机时间 大于 100 ，那么触发容错
                        int value = random.nextInt(200);

                        if (value > 100) {
                            throw new RuntimeException("Timeout!");
                        }

                        System.out.println("helloWorld() costs " + value + "
ms.");

                    }
                });

    }
}
```

Health Endpoint(`/health`)

```
{
  status: "UP",
  diskSpace: {
    status: "UP",
    total: 500096983040,
    free: 304113217536,
    threshold: 10485760
  },
  refreshScope: {
    status: "UP"
  },
  hystrix: {
    status: "UP"
  }
}
```

激活熔断保护

`@EnableCircuitBreaker` 激活： `@EnableHystrix` + Spring Cloud 功能

`@EnableHystrix` 激活，没有一些 Spring Cloud 功能，如 `/hystrix.stream` 端点

Hystrix Endpoint(`/hystrix.stream`)


```
data: {
    "type": "HystrixThreadPool",
    "name": "HystrixDemoController",
    "currentTime": 1509545957972,
    "currentActiveCount": 0,
    "currentCompletedTaskCount": 14,
    "currentCorePoolSize": 10,
    "currentLargestPoolSize": 10,
    "currentMaximumPoolSize": 10,
    "currentPoolSize": 10,
    "currentQueueSize": 0,
    "currentTaskCount": 14,
    "rollingCountThreadsExecuted": 5,
    "rollingMaxActiveThreads": 1,
    "rollingCountCommandRejections": 0,
    "propertyValue_queueSizeRejectionThreshold": 5,
    "propertyValue_metricsRollingStatisticalWindowInMilliseconds": 10000,
    "reportingHosts": 1
}
```


Spring Cloud Hystrix Dashboard

激活

`@EnableHystrixDashboard`

```
@SpringBootApplication
@EnableHystrixDashboard
public class SpringCloudHystrixDashboardDemoApplication {

  public static void main(String[] args) {
    SpringApplication.run(SpringCloudHystrixDashboardDemoApplication.class,
args);
  }
}
```

整合 Netflix Turbine

问答部分