

Hystrix使用入门手册（中文）

star24 (/u/0d0c633a5494) [+ 关注](#)

1.5 2016.12.27 19:56* 字数 2462 阅读 58451 评论 39 喜欢 94 赞赏 1

(/u/0d0c633a5494)

导语：网上资料（尤其中文文档）对hystrix基础功能的解释比较笼统，看了往往一头雾水。为此，本文将通过若干demo，加入对官网How-it-Works (<https://links.jianshu.com/go?to=https%3A%2F%2Fgithub.com%2FNetflix%2FHystrix%2Fwiki%2FHow-it-Works>)的理解和翻译，力求更清晰解释hystrix的基础功能。所用demo均对官网How-To-Use (<https://links.jianshu.com/go?to=https%3A%2F%2Fgithub.com%2FNetflix%2FHystrix%2Fwiki%2FHow-To-Use>)进行了二次修改，见<https://github.com/star2478/java-hystrix> (<https://links.jianshu.com/go?to=https%3A%2F%2Fgithub.com%2Fstar2478%2Fjava-hystrix>)

Hystrix是Netflix开源的一款容错系统，能帮助使用者码出具备强大的容错能力和鲁棒性的程序。如果某程序或class要使用Hystrix，只需简单继承

HystrixCommand/HystrixObservableCommand 并重写 run()/construct()，然后调用程序实例化此class并执行 execute()/queue()/observe()/toObservable()。

```
// HelloWorldHystrixCommand要使用Hystrix功能
public class HelloWorldHystrixCommand extends HystrixCommand {
    private final String name;
    public HelloWorldHystrixCommand(String name) {
        super(HystrixCommandGroupKey.Factory.asKey("ExampleGroup"));
        this.name = name;
    }
    // 如果继承的是HystrixObservableCommand，要重写Observable construct()
    @Override
    protected String run() {
        return "Hello " + name;
    }
}
```

```
/* 调用程序对HelloWorldHystrixCommand实例化，执行execute()即触发HelloWorldHystrixCommand
String result = new HelloWorldHystrixCommand("HLX").execute();
System.out.println(result); // 打印出Hello HLX
```

pom.xml加上以下依赖。spring cloud也集成了hystrix，不过本文只介绍原生hystrix。

```
<dependency>
<groupId>com.netflix.hystrix</groupId>
<artifactId>hystrix-core</artifactId>
<version>1.5.8</version>
</dependency>
```



本文重点介绍的是Hystrix各项基础能力的用法及其效果，不从零介绍hystrix，要了解基础知识推荐官网wiki (<https://links.jianshu.com/go?to=https%3A%2F%2Fgithub.com%2FNetflix%2FHystrix%2Fwiki>)或民间blog (<https://links.jianshu.com/go?to=http%3A%2F%2Fhot66hot.iteye.com%2Fblog%2F2155036>)

(/apps/redi
utm_sourc
banner-clic

1、HystrixCommand vs HystrixObservableCommand

要想使用hystrix，只需要继承 `HystrixCommand` 或 `HystrixObservableCommand`，简单用法见上面例子。两者主要区别是：

- 前者的命令逻辑写在 `run()`；后者的命令逻辑写在 `construct()`
- 前者的 `run()` 是由新创建的线程执行；后者的 `construct()` 是由调用程序线程执行
- 前者一个实例只能向调用程序发送（emit）单条数据，比如上面例子中 `run()` 只能返回一个String结果；后者一个实例可以顺序发送多条数据，比如demo (<https://links.jianshu.com/go?to=https%3A%2F%2Fgithub.com%2Fstar2478%2Fjava-hystrix%2Fblob%2Fmaster%2Fsrc%2Fmain%2Fjava%2Fcom%2Fpingan%2Ftest%2Fspringbootdemo%2FHystrix%2FHelloWorldHystrixObservableCommand.java>)中顺序调用多个 `onNext()`，便实现了向调用程序发送多条数据，甚至还能发送一个范围的数据集 (<https://links.jianshu.com/go?to=https%3A%2F%2Fgithub.com%2Fstar2478%2Fjava-hystrix%2Fblob%2Fmaster%2Fsrc%2Ftest%2Fjava%2Fcom%2Fpingan%2Ftest%2Fspringbootdemo%2FHystrixObservableCommand4FailsStubbedTest.java>)

2、4个命令执行方法

`execute()`、`queue()`、`observe()`、`toObservable()` 这4个方法用来触发执行 `run()/construct()`，一个实例只能执行一次这4个方法，特别说明的是 `HystrixObservableCommand` 没有 `execute()` 和 `queue()`。

4个方法的主要区别是：

- `execute()`：以同步堵塞方式执行 `run()`。以demo (<https://links.jianshu.com/go?to=https%3A%2F%2Fgithub.com%2Fstar2478%2Fjava-hystrix%2Fblob%2Fmaster%2Fsrc%2Ftest%2Fjava%2Fcom%2Fpingan%2Ftest%2Fspringbootdemo%2FHystrixCommand4ExecuteTest.java>)为例，调用 `execute()` 后，hystrix先创建一个新线程运行 `run()`，接着调用程序要在 `execute()` 调用处一直堵塞着，直到 `run()` 运行完成
- `queue()`：以异步非堵塞方式执行 `run()`。以demo (<https://links.jianshu.com/go?to=https%3A%2F%2Fgithub.com%2Fstar2478%2Fjava-hystrix%2Fblob%2Fmaster%2Fsrc%2Ftest%2Fjava%2Fcom%2Fpingan%2Ftest%2Fspringbootdemo%2FHystrixObservableCommand4FailsStubbedTest.java>)为例，调用 `queue()` 后，hystrix先创建一个新线程运行 `run()`，接着调用程序可以在 `queue()` 调用处继续执行其他操作，直到 `run()` 运行完成



Fspringbootdemo%2FHystrixCommand4QueueTest.java)为例，一调用 `queue()` 就直接返回一个Future对象，同时hystrix创建一个新线程运行 `run()`，调用程序通过 `Future.get()` 拿到 `run()` 的返回结果，而 `Future.get()` 是堵塞执行的

(/apps/redi
utm_sourc
banner-clc

- `observe()`：事件注册前执行 `run()/construct()`。以demo (<https://links.jianshu.com/go?to=https%3A%2F%2Fgithub.com%2Fstar2478%2Fjava-hystrix%2Fblob%2Fmaster%2Fsrc%2Ftest%2Fjava%2Fcom%2Fpingan%2Ftest%2Fspringbootdemo%2FHystrixCommand4ObservableTest.java>)为例，第一步是事件注册前，先调用 `observe()` 自动触发执行 `run()/construct()`（如果继承的是 `HystrixCommand`，hystrix将创建新线程非堵塞执行 `run()`；如果继承的是 `HystrixObservableCommand`，将以调用程序线程堵塞执行 `construct()`），第二步是从 `observe()` 返回后调用程序调用 `subscribe()` 完成事件注册，如果 `run()/construct()` 执行成功则触发 `onNext()` 和 `onCompleted()`，如果执行异常则触发 `onError()`
- `toObservable()`：事件注册后执行 `run()/construct()`。以demo (<https://links.jianshu.com/go?to=https%3A%2F%2Fgithub.com%2Fstar2478%2Fjava-hystrix%2Fblob%2Fmaster%2Fsrc%2Ftest%2Fjava%2Fcom%2Fpingan%2Ftest%2Fspringbootdemo%2FHystrixCommand4ObservableTest.java>)为例，第一步是事件注册前，一调用 `toObservable()` 就直接返回一个 `Observable<String>` 对象，第二步调用 `subscribe()` 完成事件注册后自动触发执行 `run()/construct()`（如果继承的是 `HystrixCommand`，hystrix将创建新线程非堵塞执行 `run()`，调用程序不必等待 `run()`；如果继承的是 `HystrixObservableCommand`，将以调用程序线程堵塞执行 `construct()`，调用程序等待 `construct()` 执行完才能继续往下走），如果 `run()/construct()` 执行成功则触发 `onNext()` 和 `onCompleted()`，如果执行异常则触发 `onError()`

3、fallback（降级）

使用fallback机制很简单，继承 `HystrixCommand` 只需重写 `getFallback()`，继承 `HystrixObservableCommand` 只需重写 `resumeWithFallback()`，比如 `HelloWorldHystrixCommand` 加下面代码片段：

```
@Override
protected String getFallback() {
    return "fallback: " + name;
}
```

fallback实际流程是当 `run()/construct()` 被触发执行时或执行中发生错误时，将转向执行 `getFallback()/resumeWithFallback()`。结合下图，4种错误情况将触发fallback：

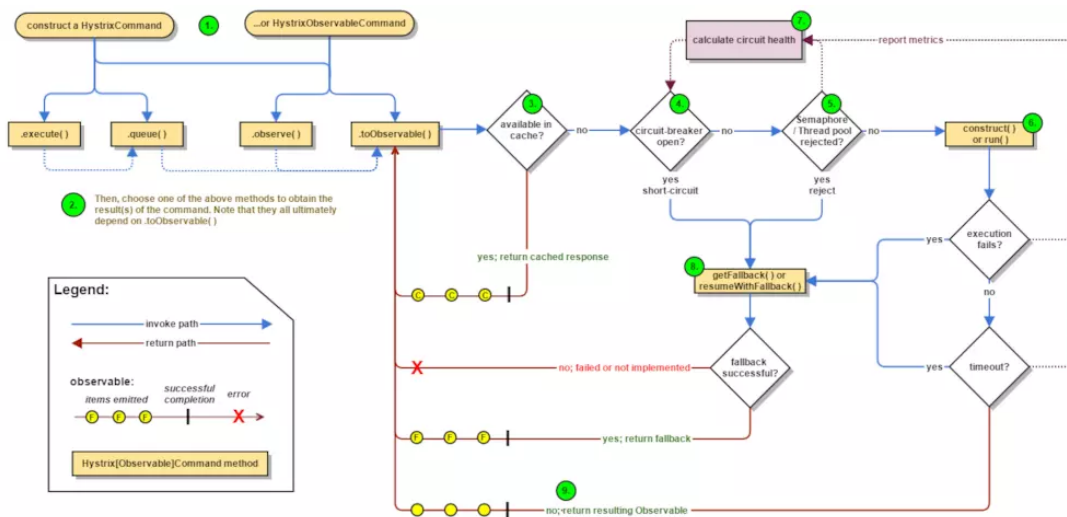
- 非 `HystrixBadRequestException` 异常：以demo (<https://links.jianshu.com/go?to=https%3A%2F%2Fgithub.com%2Fstar2478%2Fjava-hystrix%2Fblob%2Fmaster%2Fsrc%2Ftest%2Fjava%2Fcom%2Fpingan%2Ftest%2Fspringbootdemo%2FHystrixCommand4QueueTest.java>)为例，第一步是事件注册前，先调用 `observe()` 自动触发执行 `run()/construct()`（如果继承的是 `HystrixCommand`，hystrix将创建新线程非堵塞执行 `run()`；如果继承的是 `HystrixObservableCommand`，将以调用程序线程堵塞执行 `construct()`），第二步是从 `observe()` 返回后调用程序调用 `subscribe()` 完成事件注册，如果 `run()/construct()` 执行成功则触发 `onNext()` 和 `onCompleted()`，如果执行异常则触发 `onError()`



Fspringbootdemo%2FHystrixFallback4ExceptionTest.java)为例，当抛出 HystrixBadRequestException时，调用程序可以捕获异常，没有触发 getFallback()，而其他异常则会触发 getFallback()，调用程序将获得 getFallback() 的返回

(/apps/redi
utm_sourc
banner-lic

- run()/construct() 运行超时：以demo (<https://links.jianshu.com/go?to=https%3A%2F%2Fgithub.com%2Fstar2478%2Fjava-hystrix%2Fblob%2Fmaster%2Fsrc%2Ftest%2Fjava%2Fcom%2Fpingan%2Ftest%2Fspringbootdemo%2FHystrixFallback4ExceptionTest.java>)为例，使用无限while循环或sleep模拟超时，触发了 getFallback()
- 熔断器启动：以demo (<https://links.jianshu.com/go?to=https%3A%2F%2Fgithub.com%2Fstar2478%2Fjava-hystrix%2Fblob%2Fmaster%2Fsrc%2Ftest%2Fjava%2Fcom%2Fpingan%2Ftest%2Fspringbootdemo%2FHystrixCommand4CircuitBreakerTest.java>)为例，我们配置10s内请求数大于3个时就启动熔断器，请求错误率大于80%时就熔断，然后for循环发起请求，当请求符合熔断条件时将触发 getFallback()。更多熔断策略见下文
- 线程池/信号量已满：以demo (<https://links.jianshu.com/go?to=https%3A%2F%2Fgithub.com%2Fstar2478%2Fjava-hystrix%2Fblob%2Fmaster%2Fsrc%2Ftest%2Fjava%2Fcom%2Fpingan%2Ftest%2Fspringbootdemo%2FHystrixCommand4ThreadPoolTest.java>)为例，我们配置线程池数目为3，然后先用一个for循环执行 queue()，触发的 run() sleep 2s，然后再用第2个for循环执行 execute()，发现所有 execute() 都触发了fallback，这是因为第1个for的线程还在sleep，占用着线程池所有线程，导致第2个for的所有命令都无法获取到线程



来自hystrix github wiki

调用程序可以通过 isResponseFromFallback() 查询结果是由 run()/construct() 还是 getFallback()/resumeWithFallback() 返回的

4、隔离策略



hystrix提供了两种隔离策略：线程池隔离和信号量隔离。hystrix默认采用线程池隔离。

- 线程池隔离：不同服务通过使用不同线程池，彼此间将不受影响，达到隔离效果。以demo (<https://links.jianshu.com/go?to=https%3A%2F%2Fgithub.com%2Fstar2478%2Fjava-hystrix%2Fblob%2Fmaster%2Fsrc%2Ftest%2Fjava%2Fcom%2Fpingan%2Ftest%2Fspringbootdemo%2FHystrixCommand4ThreadPoolTest.java>)为例，我们通过 `andThreadPoolKey` 配置使用命名为 `ThreadPoolTest` 的线程池，实现与其他命名的线程池天然隔离，如果不配置 `andThreadPoolKey` 则使用 `withGroupKey` 配置来命名线程池
- 信号量隔离：线程隔离会带来线程开销，有些场景（比如无网络请求场景）可能会因为用开销换隔离得不偿失，为此hystrix提供了信号量隔离，当服务的并发数大于信号量阈值时将进入fallback。以demo (<https://links.jianshu.com/go?to=https%3A%2F%2Fgithub.com%2Fstar2478%2Fjava-hystrix%2Fblob%2Fmaster%2Fsrc%2Ftest%2Fjava%2Fcom%2Fpingan%2Ftest%2Fspringbootdemo%2FHystrixCommand4SemaphoreTest.java>)为例，通过 `withExecutionIsolationStrategy(ExecutionIsolationStrategy.SEMAPHORE)` 配置为信号量隔离，通过 `withExecutionIsolationSemaphoreMaxConcurrentRequests` 配置执行并发数不能大于3，由于信号量隔离下无论调用哪种命令执行方法，hystrix都不会创建新线程执行 `run()/construct()`，所以调用程序需要自己创建多个线程来模拟并发调用 `execute()`，最后看到一旦并发线程>3，后续请求都进入fallback

(/apps/redi
utm_sourc
banner-clc

5、熔断机制

熔断机制相当于电路的跳闸功能，举个栗子，我们可以配置熔断策略为当请求错误比例在10s内>50%时，该服务将进入熔断状态，后续请求都会进入fallback。

以demo (<https://links.jianshu.com/go?to=https%3A%2F%2Fgithub.com%2Fstar2478%2Fjava-hystrix%2Fblob%2Fmaster%2Fsrc%2Ftest%2Fjava%2Fcom%2Fpingan%2Ftest%2Fspringbootdemo%2FHystrixCommand4CircuitBreakerTest.java>)为例，我们通过 `withCircuitBreakerRequestVolumeThreshold` 配置10s内请求数超过3个时熔断器开始生效，通过 `withCircuitBreakerErrorThresholdPercentage` 配置错误比例>80%时开始熔断，然后for循环执行 `execute()` 触发 `run()`，在 `run()` 里，如果 `name` 是小于10的偶数则正常返回，否则超时，通过多次循环后，超时请求占所有请求的比例将大于80%，就会看到后续请求都不进入 `run()` 而是进入 `getFallback()`，因为不再打印 "running run():" + `name` 了。

除此之外，hystrix还支持多长时间从熔断状态自动恢复等功能，见下文附录。

6、结果cache

hystrix支持将一个请求结果缓存起来，下一个具有相同key的请求将直接从缓存中取出结果，减少请求开销。要使用hystrix cache功能，第一个要求是重写 `getCacheKey()`，用来构造cache key；第二个要求是构建context，如果请求B要用到请求A的结果缓存，A和B必须同处一个context。通过 `HystrixRequestContext.initializeContext()` 和 `context.shutdown()` 可以构建一个context，这两条语句间的所有请求都处于同一个context。



以demo (<https://links.jianshu.com/go?to=https%3A%2F%2Fgithub.com%2Fstar2478%2Fjava-hystrix%2Fblob%2Fmaster%2Fsrc%2Ftest%2Fjava%2Fcom%2Fpingan%2Ftest%2Fspringbootdemo%2FHystrixCommand4RequestCacheTest.java>)的 `testWithCacheHits()` 为例, `command2a`、`command2b`、`command2c`同处一个context, 前两者的cache key都是 2HLX (见 `getCacheKey()`), 所以`command2a`执行完后把结果缓存, `command2b`执行时就不走 `run()` 而是直接从缓存中取结果了, 而`command2c`的cache key是 2HLX1 , 无法从缓存中取结果。此外, 通过 `isResponseFromCache()` 可检查返回结果是否来自缓存。

(/apps/redi
utm_sourc
banner-clic

7、合并请求collapsing

hystrix支持N个请求自动合并为一个请求, 这个功能在有网络交互的场景下尤其有用, 比如每个请求都要网络访问远程资源, 如果把请求合并为一个, 将使多次网络交互变成一次, 极大节省开销。重要一点, 两个请求能自动合并的前提是两者足够“近”, 即两者启动执行的间隔时长要足够小, 默认为10ms, 即超过10ms将不自动合并。

以demo (<https://links.jianshu.com/go?to=https%3A%2F%2Fgithub.com%2Fstar2478%2Fjava-hystrix%2Fblob%2Fmaster%2Fsrc%2Ftest%2Fjava%2Fcom%2Fpingan%2Ftest%2Fspringbootdemo%2FHystrixCommand4RequestCollapsingTest.java>)为例, 我们连续发起多个queue请求, 依次返回f1~f6共6个Future对象, 根据打印结果可知f1~f5同处一个线程, 说明这5个请求被合并了, 而f6由另一个线程执行, 这是因为f5和f6中间隔了一个sleep, 超过了合并要求的最大间隔时长。

附录：各种策略配置

根据<http://hot66hot.iteye.com/blog/2155036> (<https://links.jianshu.com/go?to=http%3A%2F%2Fhot66hot.iteye.com%2Fblog%2F2155036>) 整理而得。

- `HystrixCommandProperties`



```

/* -----统计相关-----*/
// 统计滚动的时间窗口,默认:5000毫秒(取自circuitBreakerSleepWindowInMilliseconds)
private final HystrixProperty metricsRollingStatisticalWindowInMilliseconds;
// 统计窗口的Buckets的数量,默认:10个,每秒一个Buckets统计
private final HystrixProperty metricsRollingStatisticalWindowBuckets; // number of bu
// 是否开启监控统计功能,默认:true
private final HystrixProperty metricsRollingPercentileEnabled;
/* -----熔断器相关-----*/
// 熔断器在整个统计时间内是否开启的阈值,默认20。也就是在metricsRollingStatisticalWindowIn
private final HystrixProperty circuitBreakerRequestVolumeThreshold;
// 熔断时间窗口,默认:5秒.熔断器中断请求5秒后会进入半打开状态,放下一个请求进来重试, 如果该请
private final HystrixProperty circuitBreakerSleepWindowInMilliseconds;
//是否启用熔断器,默认true. 启动
private final HystrixProperty circuitBreakerEnabled;
//默认:50%。当出错率超过50%后熔断器启动
private final HystrixProperty circuitBreakerErrorThresholdPercentage;
//是否强制开启熔断器阻断所有请求,默认:false,不开启。置为true时,所有请求都将被拒绝,直接到f
private final HystrixProperty circuitBreakerForceOpen;
//是否允许熔断器忽略错误,默认false, 不开启
private final HystrixProperty circuitBreakerForceClosed;
/* -----信号量相关-----*/
//使用信号量隔离时, 命令调用最大的并发数,默认:10
private final HystrixProperty executionIsolationSemaphoreMaxConcurrentRequests;
//使用信号量隔离时, 命令fallback(降级)调用最大的并发数,默认:10
private final HystrixProperty fallbackIsolationSemaphoreMaxConcurrentRequests;
/* -----其他-----*/
//使用命令调用隔离方式,默认:采用线程隔离,ExecutionIsolationStrategy.THREAD
private final HystrixProperty executionIsolationStrategy;
//使用线程隔离时, 调用超时时间, 默认:1秒
private final HystrixProperty executionIsolationThreadTimeoutInMilliseconds;
//线程池的key,用于决定命令在哪个线程池执行
private final HystrixProperty executionIsolationThreadPoolKeyOverride;
//是否开启fallback降级策略 默认:true
private final HystrixProperty fallbackEnabled;
// 使用线程隔离时, 是否对命令执行超时的线程调用中断(Thread.interrupt())操作.默认:true
private final HystrixProperty executionIsolationThreadInterruptOnTimeout;
// 是否开启请求日志,默认:true
private final HystrixProperty requestLogEnabled;
//是否开启请求缓存,默认:true
private final HystrixProperty requestCacheEnabled; // Whether request caching is enab

```

(/apps/redi
utm_sourc
banner-lic

• HystrixCollapserProperties

```

//请求合并是允许的最大请求数,默认: Integer.MAX_VALUE
private final HystrixProperty maxRequestsInBatch;
//批处理过程中每个命令延迟的时间,默认:10毫秒
private final HystrixProperty timerDelayInMilliseconds;
//批处理过程中是否开启请求缓存,默认:开启
private final HystrixProperty requestCacheEnabled;

```

• HystrixThreadPoolProperties

```

/* 配置线程池大小,默认值10个 */
private final HystrixProperty corePoolSize;
/* 配置线程值等待队列长度,默认值:-1 建议值:-1表示不等待直接拒绝,测试表明线程池使用直接决绝策
private final HystrixProperty maxQueueSize;

```

参考文献

<https://github.com/Netflix/Hystrix> ([https://links.jianshu.com/go?](https://links.jianshu.com/go?to=https%3A%2F%2Fgithub.com%2FNetflix%2FHystrix)

[to=https%3A%2F%2Fgithub.com%2FNetflix%2FHystrix](https://github.com/Netflix/Hystrix/wiki/How-To-Use))

<https://github.com/Netflix/Hystrix/wiki/How-To-Use> (<https://links.jianshu.com/go?>



to=https%3A%2F%2Fgithub.com%2FNetflix%2FHystrix%2Fwiki%2FHow-To-Use)
http://hot66hot.iteye.com/blog/2155036 (https://links.jianshu.com/go?
to=http%3A%2F%2Fhot66hot.iteye.com%2Fblog%2F2155036)

(/apps/redi
utm_sourc
banner-clic

小礼物走一走，来简书关注我

赞赏支持



(/u/695ea1dca892)

📄 java原创 (/nb/4547515)

举报文章 © 著作权归作者所有



star24 (/u/0d0c633a5494)

写了 20124 字，被 189 人关注，获得了 195 个喜欢
(/u/0d0c633a5494)

+ 关注

我已委托“维权骑士”（rightknights.com）为我的文章进行维权行动。 https://github.com/star2478

喜欢 | 94



更多分享



登录 (/sign) 发表评论 source=desktop&utm_medium=not-signed-in-comr

39条评论

只看作者

按时间倒序 按时间正序



安以北往南 (/u/7219ede55429)

19楼 · 2019.06.17 16:47

(/u/7219ede55429)
学习了 谢谢

赞 回复



安静的Boy_f31d (/u/ecd59de8815f)

18楼 · 2019.05.31 18:55

(/u/ecd59de8815f)
corePoolSize的解释：

配置线程池大小,默认值10个. 建议值:请求高峰时99.5%的平均响应时间 + 向上预留一些
即可

不是太明白，比如我的web应用平均相应时间是500ms,那这个值是应该设置成
0.995*0.5=497个线程吗

赞 回复



star24 (/u/0d0c633a5494)：这里所写的建议值有问题（这部分已去掉），多谢提醒！建议值可以按这个公式来：线程数=一个任务平均执行总时间*cpu核数/一个任务平均cpu处理时间，其中，执行总时间=等待IO时间+cpu处理时间。

2019.06.01 17:00 回复

添加新评论

(/apps/redi
utm_sourc
banner-clc



Panda2018 (/u/fda80a57863c)

17楼 · 2019.02.22 11:28

(/u/fda80a57863c)
大感谢

赞 回复



KingsonWu (/u/f1b42d822d36)

16楼 · 2018.10.29 23:32

(/u/f1b42d822d36)
真的很不错，谢谢

赞 回复



一飞_0269 (/u/aeef75e43fc2)

15楼 · 2018.03.28 11:55

(/u/aeef75e43fc2)
5、熔断机制

这里面的10S是固定的吗？demo里没有10s配置

赞 回复

star24 (/u/0d0c633a5494)：@一飞_0269 (/users/aeef75e43fc2) 可以配置，默认是10s

2018.03.29 22:50 回复

添加新评论



秋林格瓦斯 (/u/ef99634ac8b8)

14楼 · 2018.03.10 13:58

(/u/ef99634ac8b8)
写的不错,加油

赞 回复



素描的天空chen (/u/d1532ed8152e)

13楼 · 2017.11.07 23:31

(/u/d1532ed8152e)

想咨询一下，用了这个HystrixCommand，fallback方法中我怎样能拿到具体的异常信息？比如因为超时，或者并发数过大等等异常

赞 回复

kacey_zhang (/u/5890386aefb4)：我也想问这个问题，run方法中是业务逻辑，业务逻辑可能会抛出业务异常。

2017.11.09 17:21 回复

素描的天空chen (/u/d1532ed8152e)：@kacey_zhang (/users/5890386aefb4) 在fallback里可以通过this.ExecuteException...拿到，你可以试试



2017.11.13 21:56 回复

star24 (/u/0d0c633a5494) : @素描的天空chen (/users/d1532ed8152e) 补充一下，在fallback里可以通过this.getExecutionException()获取执行run过程中抛出的异常，除了HystrixBadRequestException外。当run抛出HystrixBadRequestException，不会触发执行fallback，而且如果应用主程序不catch该异常，就会异常退出

2017.11.22 17:22 回复

添加新评论 | 还有3条评论，展开查看



边磊x (/u/ad32582be739)

12楼 · 2017.10.05 17:57

(/u/ad32582be739)

《Hystrix使用入门手册(中文) - 简书》写的不错不错，收藏了。

推荐下，分布式作业中间件 Elastic-Job 源码解析 16 篇：<http://tinyurl.com/y93r9wfg>
(<http://tinyurl.com/y93r9wfg>)

渔

13人赞 回复

逢尾 (/u/8afe04d66ee7) : 恩恩

还不错那

2017.10.06 15:34 回复

添加新评论



star24 (/u/0d0c633a5494) 作者

10楼 · 2017.09.21 14:04

(/u/0d0c633a5494)

@卖艺的大龄青年 (/users/2fd2f11e64e6) 对于超时的那次请求，业务不会被中断。当超时引发熔断器启动后，后续一段时间内的请求都将进入fallback，不会再调用业务

赞 回复

田大侠ly (/u/9c8b1cfd4040) : 请问Hystrix有中断线程的机制吗?能否设置超时后释放资源

2019.07.15 16:27 回复

添加新评论



千里浪打浪 (/u/2fd2f11e64e6)

9楼 · 2017.09.20 19:46

(/u/2fd2f11e64e6)

我设置了超时时间 比如我业务的执行时间是5s 我在hystrix中设置的timeout是2s，虽然执行了fallback方法 但是业务操作并没有发生中断，还是在继续执行，并打印出了后续的日志，请问这是为什么，我的策略用的也是Thread

赞 回复



唐植超 (/u/d8355c97ac16)

8楼 · 2017.08.30 15:42

(/u/d8355c97ac16)

(/apps/redi
utm_sourc
banner-clc



熔断了他自动恢复吗，还是需要手动恢复啊

赞 回复

star24 (/u/0d0c633a5494) : @唐植超 (/users/d8355c97ac16) 自动恢复
2017.08.30 18:42 回复

添加新评论

(/apps/redi
utm_sourc
banner-clic



风雨诗轩 (/u/5d98f2f7090c)

7楼 · 2017.08.07 13:19

(/u/5d98f2f7090c)

在真实的线上web项目中，如何保证所有请求处于同一个context中？我试过将HystrixRequestContext.initializeContext()放到filter中，但是每请求一次，就初始化一次，也不是在同一个context中，这样如何实现缓存？

赞 回复

star24 (/u/0d0c633a5494) : hystrix context主要用在collapse和cache，主要解决的都是同一个请求链路中的请求合并或数据共享问题，尤其请求链路中涉及到团队不同人的代码，大家可以按照hystrix context语义来协同开发，减少沟通。如果要实现不同请求共享cache，还不如换方案，比如redis，也能直接升级成分布式cache

2017.08.08 19:11 回复

李庚 (/u/59bcb222d3a8) : @star24 (/users/0d0c633a5494) 同一个请求链路中是指同一个url吗，但是同一个url每次也会执行初始化的，多次访问同一个url依然不能共享

2017.09.24 00:05 回复

star24 (/u/0d0c633a5494) : @李庚 (/users/59bcb222d3a8) 可以理解是一个url的一次访问，不是多次。一次访问可能会涉及到多个模块，不同模块可能由不同人开发，这些模块共享数据

2017.09.24 00:14 回复

添加新评论 | 还有2条评论， 展开查看



留存的情缘 (/u/14e4848a2caa)

6楼 · 2017.07.07 17:59

(/u/14e4848a2caa)

withCircuitBreakerSleepWindowInMilliseconds(3000)//熔断器打开到关闭的时间窗长度
这个是不是应该是到半关闭状态的时间，我改了代码，为啥我接下来输出：

```
====CircuitBreaker fallback: 5
====CircuitBreaker fallback: 6
====CircuitBreaker fallback: 7
====CircuitBreaker fallback: 8
====CircuitBreaker fallback: 9
====CircuitBreaker fallback: 10
====CircuitBreaker fallback: 11
running run():12
====CircuitBreaker fallback: 12
====CircuitBreaker fallback: 13
running run():14
====CircuitBreaker fallback: 14
====CircuitBreaker fallback: 15
running run():16
====CircuitBreaker fallback: 16
====CircuitBreaker fallback: 17
```



running run():18

====CircuitBreaker fallback: 18，感觉下面不该是轮询的，应该尝试一个连接，没连上，然后继续失败，等待3000ms呀

赞 回复

(/apps/redi
utm_sourc
banner-clic

star24 (/u/0d0c633a5494)：这是用哪个demo？详细代码贴来看看

2017.07.09 23:46 回复

留存的情缘 (/u/14e4848a2caa)：熔断器默认工作时间,默认:5秒.熔断器中断请求5秒后会进入半打开状态,放部分流量过去重试，这个放部分流量过去有什么机制吗？放过去的如果失败会怎样？

2017.07.28 10:57 回复

star24 (/u/0d0c633a5494)：@留存的情缘 (/users/14e4848a2caa) 先放一个请求进来，成功的话就关闭熔断，失败的话就再等一段时间（由circuitBreakerSleepWindowInMilliseconds设置）。熔断器工作流程详见：<https://github.com/Netflix/Hystrix/wiki/How-it-Works#CircuitBreaker> (<https://github.com/Netflix/Hystrix/wiki/How-it-Works#CircuitBreaker>)

2017.08.08 17:29 回复

添加新评论 | 还有1条评论，展开查看



star24 (/u/0d0c633a5494) 作者

5楼 · 2017.06.26 18:29

(/u/0d0c633a5494)

@程凯_3225 (/users/ae0b3958fda1) 我这边实际mvn test执行了几次

HystrixCommand4CircuitBreakerTest都会被熔断。demo中，要被熔断（直接进入fallback而不进入run）需要同时满足两个条件：一是10s内至少请求3次，二是异常占比超过80%。所以需要循环大概20次左右才会进入熔断，比如循环到25次时，正常请求有5次，超时异常累计到20次，此时满足两个条件

赞 回复



程凯_3225 (/u/ae0b3958fda1)

4楼 · 2017.06.21 16:17

(/u/ae0b3958fda1)

信号量隔离的demo中，改成这样就可以了。

```
public class SemaphoreCircuitBreakerCommandTest {
```

```
    public static void main(String[] args) throws IOException {
```

```
        for (int i = 0; i < 10; i++) {
```

```
            final SemaphoreCircuitBreakerCommand command = new
```

```
                SemaphoreCircuitBreakerCommand(String.valueOf(i));
```

```
            Thread th = new Thread(new Runnable() {
```

```
                @Override (/users/d55323762b08)
```

```
                public void run() {
```

```
                    command.execute();
```

```
                }
```

```
            });
```

```
            th.start();
```

```
        }
```

```
        System.in.read();
```



}

}

1人赞 回复

(/apps/redi
utm_sourc
banner-clic

1

2

下一页

被以下专题收入，发现更多相似内容

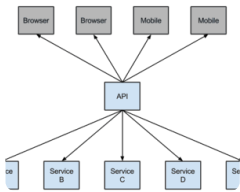
-  经典文章 (/c/fb15e641db83?utm_source=desktop&utm_medium=notes-included-collection)
-  赞 (/c/3bde1441e2a4?utm_source=desktop&utm_medium=notes-included-collection)
-  Spring ... (/c/bc163669cee8?utm_source=desktop&utm_medium=notes-included-collection)
-  Spring ... (/c/31ba3fdef06b?utm_source=desktop&utm_medium=notes-included-collection)
-  计算机 (/c/9aa45f8a9fe4?utm_source=desktop&utm_medium=notes-included-collection)
-  Hystrix (/c/b38c43ac3aa1?utm_source=desktop&utm_medium=notes-included-collection)
-  新技术 (/c/576cf7a3dae9?utm_source=desktop&utm_medium=notes-included-collection)

展开更多

推荐阅读


更多精彩内容 > (/)

(/p/46fd0faecac1?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendatio
Spring Cloud (/p/46fd0faecac1?utm_campaign=maleskine&utm_conte...

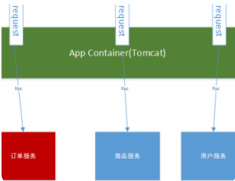
Spring Cloud为开发人员提供了快速构建分布式系统中一些常见模式的工具（例如配置管理，服务发现，断路器，智能路由，微代理，控制总线）。分布式系统的协调导致了样板模式, 使用Spring Cloud开发人员可...

 卡卡罗2017 (/u/d90908cb0d85?

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio



(/p/3e11ac385c73?



(/apps/redi
utm_sourc
recommendatio
banner-clip

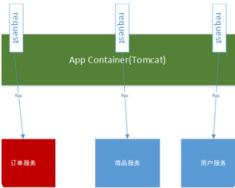
utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendatio
Hystrix技术解析 (/p/3e11ac385c73?utm_campaign=maleskine&utm_co...

一、认识Hystrix Hystrix是Netflix开源的一款容错框架，包含常用的容错方法：线程池隔离、信号量隔离、熔断、降级回退。在高并发访问下，系统所依赖的服务的稳定性对系统的影响非常大，依赖有很多不可控的...

新栋BOOK (/u/f2fa1bce6780?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

(/p/b6e8d91b2a96?



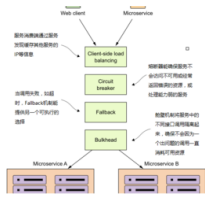
utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendatio
Hystrix技术解析(图片版) (/p/b6e8d91b2a96?utm_campaign=maleskine...

一、认识Hystrix Hystrix是Netflix开源的一款容错框架，包含常用的容错方法：线程池隔离、信号量隔离、熔断、降级回退。在高并发访问下，系统所依赖的服务的稳定性对系统的影响非常大，依赖有很多不可控的...

新栋BOOK (/u/f2fa1bce6780?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

(/p/6c574abe50c1?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendatio
服务容错保护——Spring Cloud Hystrix (/p/6c574abe50c1?utm_campa...

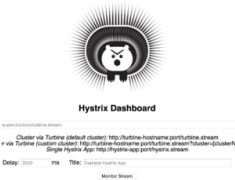
(git上的源码：https://gitee.com/rain7564/spring_microservices_study/tree/master/forth-spring-cloud-hystrix)

几乎每一个系统，特别是分布式系统，都会有调用失败的情况，最有效的办法...

sprinkle_liz (/u/c13f3c6ada04?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

(/p/efb049107572?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendatio
springcloud入门系列(6) - Hystrix详解 (/p/efb049107572?utm_campaign...

本篇会介绍下Hystrix的使用，会从最简单的入门实例开始，然后会讲述下Hystrix的隔离和熔断的原理和流程，最后讲解下Hystrix的各类参数来指导大家更快上手。 1. Hystrix实例 1) Pom增加依赖 2) Application...


monkey01 (/u/808054b533e9?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio



上不了首页的也是好写手 (/p/49d6ffdeb864?utm_campaign=maleskine&...


我的文章经常上首页，怎么觉得吧，上不上首页真的不重要，重要的是我的写作能够帮到人，我把写作当成是一种快乐，那种就是只为热爱生活，不为上首页，分享就是一种快乐，做最真实的自己，现在的我已经...

 简明估 (/u/b2d7c4804072?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation
utm_source=app_banner
utm_source=banner-click

爱，请从非暴力沟通开始 (/p/be5b30e677b8?utm_campaign=maleskine&...


本周与弗兰克和虐友们一起读完《非暴力沟通》，受益颇多。如今的我已过三十而立，对于家庭，上有老，下有小；对于工作，上有领导，下有小员。总感觉，自己就像汉堡包里的肉一样，两边受气。忍.....忍.....

 瑶瑶_9bbd (/u/4d05abbbc4f7?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation

用Python做一个在线学习网站 (/p/00bf258c61bc?utm_campaign=malesk...

*新手友好 Tip: 结合项目代码看比较好 几点说明 整体上 app 内包含主要的项目文件 tests 内包含两个测试文件 manager.py 是启动项目的文件 各部分以 flask蓝图 的形式存在 为了便于测试，准备了几个简陋的HTML...

 胡写八写 (/u/e3a3280fe5e6?)


utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation

(/p/7ed40300e764?)

utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation

08-03【晨读感悟】没有行动，都是空谈 (/p/7ed40300...

俗话说“千里之行始于足下”，任何想抵达的地方，都要从脚下出发；任何想实现的目标，都要从当下开始。没有行动，都是空谈；反之，成功的第一步“做，...

 朱朱的餐具 (/u/05aa394dafec?)



utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation

