Hystrix 的github地址，介绍了很多hystrix配置参数

<https://github.com/Netflix/Hystrix/wiki/Configuration#execution.isolation.strategy>

github上的wiki文档很清楚

Hystrix使用

要是用Hystrix自由的注解，需要在启动类上面使用@EnableHystrix

参考文档： <https://blog.csdn.net/qq_41543180/article/details/103701121>

feign client超时时间，大于Hystrix配置的超时时间。

feign使用Hystrix熔断配置

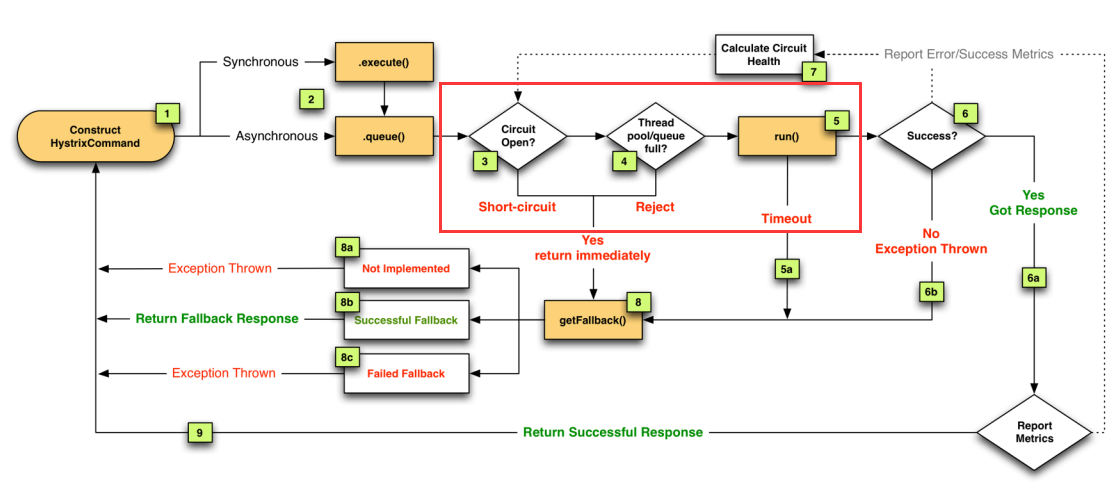
<https://www.cnblogs.com/lori/p/11112680.html>

<https://www.jianshu.com/p/52a3c60b6343>

hystrix原理

<https://www.cnblogs.com/amazement/p/8445294.html>

**熔断器具体执行逻辑**



1. 将远程服务调用逻辑封装进一个HystrixCommand。
2. 对于每次服务调用可以使用同步或异步机制，对应执行execute()或queue()。
3. 判断熔断器(circuit-breaker)是否打开或者半打开状态，如果打开跳到步骤8，进行回退策略，如果关闭进入步骤4。
4. 判断线程池/队列/信号量（使用了舱壁隔离模式）是否跑满，如果跑满进入回退步骤8，否则继续后续步骤5。
5. run方法中执行了实际的服务调用。

a. 服务调用发生超时时，进入步骤8。

1. 判断run方法中的代码是否执行成功。

a. 执行成功返回结果。

b. 执行中出现错误则进入步骤8。

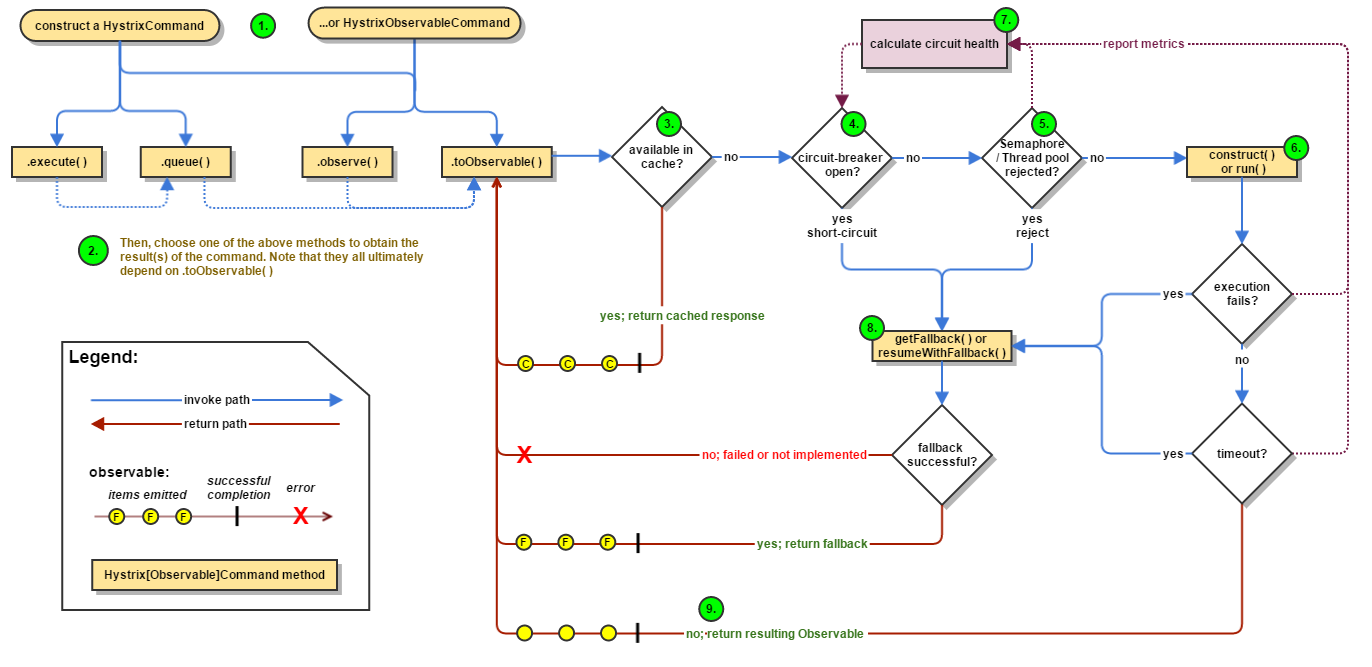
1. 所有的运行状态(成功，失败，拒绝，超时)上报给熔断器，用于统计从而影响熔断器状态。
2. 进入getFallback()回退逻辑。

a. 没有实现getFallback()回退逻辑的调用将直接抛出异常。

b. 回退逻辑调用成功直接返回。

c. 回退逻辑调用失败抛出异常。

1. 返回执行成功结果。



Hystrix will execute this fallback for all types of failure such as run() failure, timeout, thread pool or semaphore rejection, and circuit-breaker short-circuiting.

HystrixBadRequestException  not count as failures and trigger getFallback() and circuit-breaker logic.

# [SpringCloud系列-整合Hystrix的两种方式](https://www.cnblogs.com/haha12/p/11690489.html)

<https://www.cnblogs.com/haha12/p/11690489.html>

feign使用Hystrix

<https://blog.csdn.net/weixin_30407613/article/details/101084213>

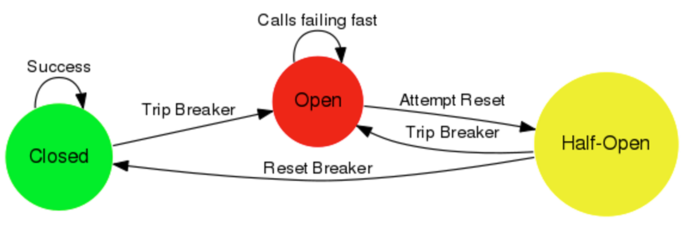
熔断器，3种状态：1、断开，2正常，3、半熔断

熔断打开 就是trip，表示跳闸了

正常状态下，电路处于关闭状态(Closed)，如果调用持续出错或者超时，电路被打开进入熔断状态(Open)，后续一段时间内的所有调用都会被拒绝(Fail Fast)，

这个拒绝时间withCircuitBreakerSleepWindowInMilliseconds控制默认是5s

一段时间以后，保护器会尝试进入半熔断状态(Half-Open)，允许少量请求进来尝试，如果调用仍然失败，则回到熔断状态，如果调用成功，则回到电路闭合状态;



参考文档

<https://www.cnblogs.com/amazement/p/8445294.html>

Hystrix使用是通过编辑java类继承 HystrixCommand抽象类，重写run(),getFallback()方法。上层代码使用这个command，使远程调用具有熔断、降级功能。

feign-hysrix有些差异。默认情况下，所有Hystrix功能都只能是用default配置，不够灵活，不能灵活控制每个方法。

@HystrixCommand注解是单独使用的，不能和feign一起使用。

hystrix还有dashboard，用于监控熔断降级情况。

熔断降级，不适合处理强依赖服务。只是和若依赖服务。

服务之间的依赖有强弱之分。

理论性的东西：

<https://blog.51cto.com/snowtiger/2057092>

这个文档很好。

Hystrix能做什么？

* 在通过第三方客户端访问（通常是通过网络）依赖服务出现高延迟或者失败时，为系统提供保护和控制
* 在分布式系统中防止级联失败
* 快速失败（Fail fast）同时能快速恢复
* 提供失败回退（Fallback）和优雅的服务降级机制
* 提供近实时的监控、报警和运维控制手段

Hystrix设计原则？

* 防止单个依赖耗尽容器（例如 Tomcat）内所有用户线程
* 降低系统负载，对无法及时处理的请求快速失败（fail fast）而不是排队
* 提供失败回退，以在必要时让失效对用户透明化
* 使用隔离机制（例如『舱壁』/『泳道』模式，熔断器模式等）降低依赖服务对整个系统的影响
* 针对系统服务的度量、监控和报警，提供优化以满足近实时性的要求
* 在 Hystrix 绝大部分需要动态调整配置并快速部署到所有应用方面，提供优化以满足快速恢复的要求
* 能保护应用不受依赖服务的整个执行过程中失败的影响，而不仅仅是网络请求

很好的文章

<https://www.jianshu.com/p/f6d7f832961d>