Feign整合Hystrix

Feign是以接口形式工作的，要如何整合Hystrix了？又是如何实现降级了？

事实上，SpringCloud默认已为Feign整合了Hystrix，只要Hystrix在项目的classpath中，Feign默认就会用断路器包裹所有方法。

(注意：从Spring Cloud Dalston开始，Feign默认是不开启Hystrix的。因此，如使用Dalston及以上版本请务必额外设置属性：feign.hystrix.enabled=true，否则断路器不会生效)

见示例：07-ms-consumer-order-feign-hystrix-fallback

Feign整合Hystrix的写法见<UserFeignClient>类，只需使用@FeignClient注解的fallback属性就可以为指定名称Feign客户端添加降级方法

```
/**
 * Feign的fallback测试
 * 使用@FeignClient的fallback属性指定回退类
 */
@FeignClient(name = "microservice-provider-user", fallback = FeignClientFallback.class)
public interface UserFeignClient {
  @RequestMapping(value = "/{id}", method = RequestMethod.GET)
  public User findById(@PathVariable("id") Long id);

}

/**
 * 回退类FeignClientFallback需实现Feign Client接口
 * FeignClientFallback也可以是public class，没有区别
 */
@Component
class FeignClientFallback implements UserFeignClient {
  @Override
  public User findById(Long id) {
    User user = new User();
    user.setId(-1L);
    user.setUsername("默认用户");
    return user;
  }
}
```

Feign禁用Hystrix

SpringCloud为Feign默认整合了Hystrix，也就是说只要Hystrix在项目的classpath中，Feign就会使用断路器包裹Feign客户端的所有方法(Dalston及以上版本默认Feign不开启Hystrix)。这样虽然方便，但有的场景并不需要该功能，如何为Feign禁用Hystrix呢?

见示例：07-ms-consumer-order-feign-hystrix-fallback

全局禁用Hystrix

只需在application.yml中配置feign.hystrix.enabled=false即可

为指定Feign客户端禁用Hystrix：

增加<FeignDisableHystrixConfiguration>类

```java
/**
 * 为Feign禁用Hystrix功能
 *
 */
@Configuration
public class FeignDisableHystrixConfiguration {

    @Bean
    @Scope("prototype")
    public Feign.Builder feignBuilder() {
        return Feign.builder();
    }
}
```

在FeignClient注解里加上configuration的属性配置，见下图：

```java
/**
 * Feign的fallback测试
 * 使用@FeignClient的fallback属性指定回退类
 */
@FeignClient(name = "microservice-provider-user"/*, configuration = FeignDisableHystrixConfiguration.class 禁用hystrix*/,
public interface UserFeignClient {
  @RequestMapping(value = "/{id}", method = RequestMethod.GET)
  public User findById(@PathVariable("id") Long id);

}

/**
 * 回退类FeignClientFallback需实现Feign Client接口
 * FeignClientFallback也可以是public class，没有区别
 */
@Component
class FeignClientFallback implements UserFeignClient {
  @Override
  public User findById(Long id) {
    User user = new User();
    user.setId(-1L);
    user.setUsername("默认用户");
    return user;
  }
}
```

Hystrix监控

除实现容错外， Hystrix还提供了近乎实时的监控。 HystrixCommand在执行时，会生成执行结果和运行指标，比如每秒执行的请求数、成功数等，这些监控数据对分析应用系统的状态很有用。

使用 Hystrix的模块 hystrix-metrics-event-stream，就可将这些监控的指标信息以text/event-stream的格式暴露给外部系统。 spring-cloud-starter-netflix-hystrix包含该模块，在此基础上，只须为项目添加 spring-boot-starter-actuator依赖，就可使用/hystrix.stream端点获得Hystrix的监控信息了。

Feign项目的Hystrix监控

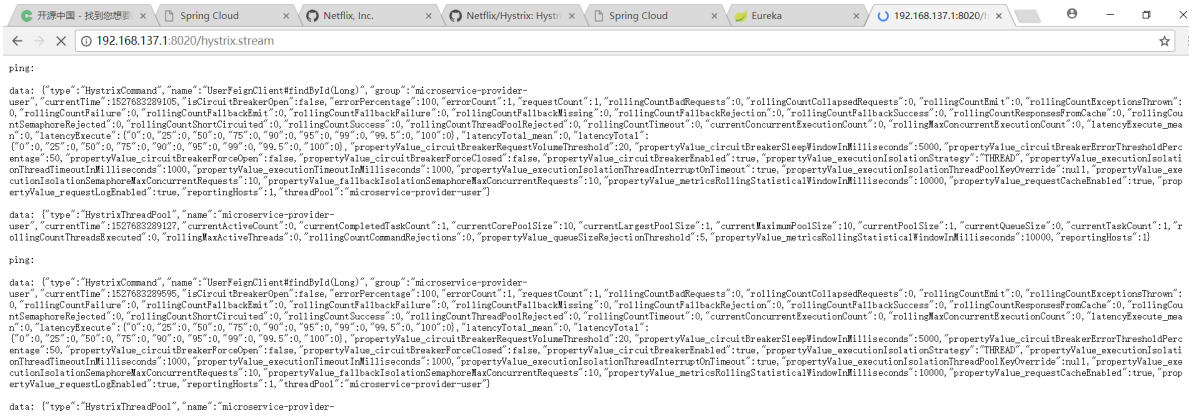
见示例：07-ms-consumer-order-feign-hystrix-fallback-stream

为项目增加依赖

```xml
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-netflix-hystrix</artifactId>
</dependency>
```

在启动类上增加@EnableCircuitBreaker，这样就可以使用/hystrix.stream端点监控Hystrix了

运行项目后访问地址：http://192.168.137.1:8020/hystrix.stream，效果如下图



使用Hystrix Dashboard可视化监控数据

前面讨论了Hystrix的监控，但访问/hystrix.stream端点获得的数据是以文字形式展示的。很难通过这些数据，一眼看出系统当前的运行状态。

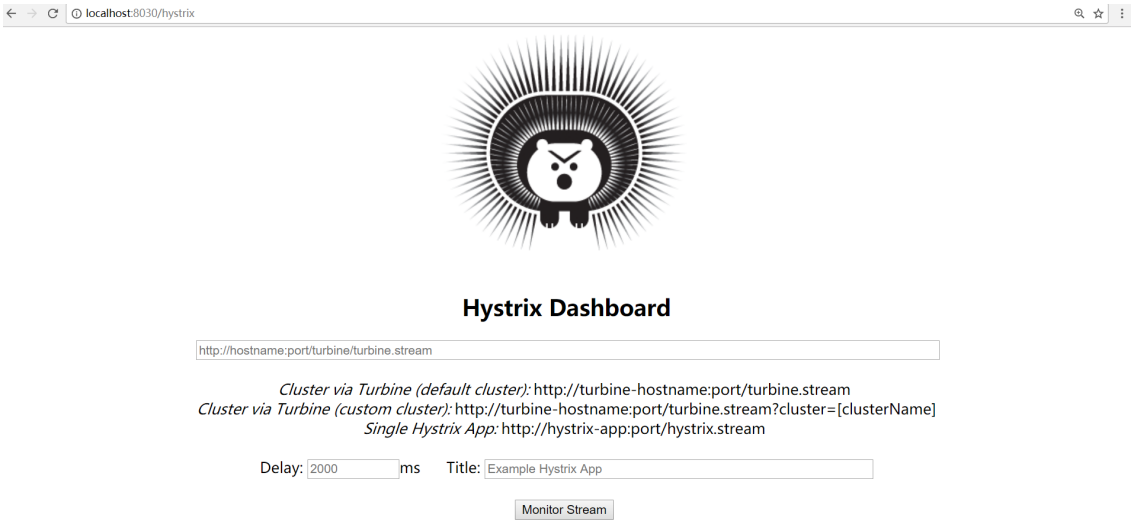
可使用 Hystrix Dashboard，让监控数据图形化、可视化

见示例：07-ms-hystrix-dashboard

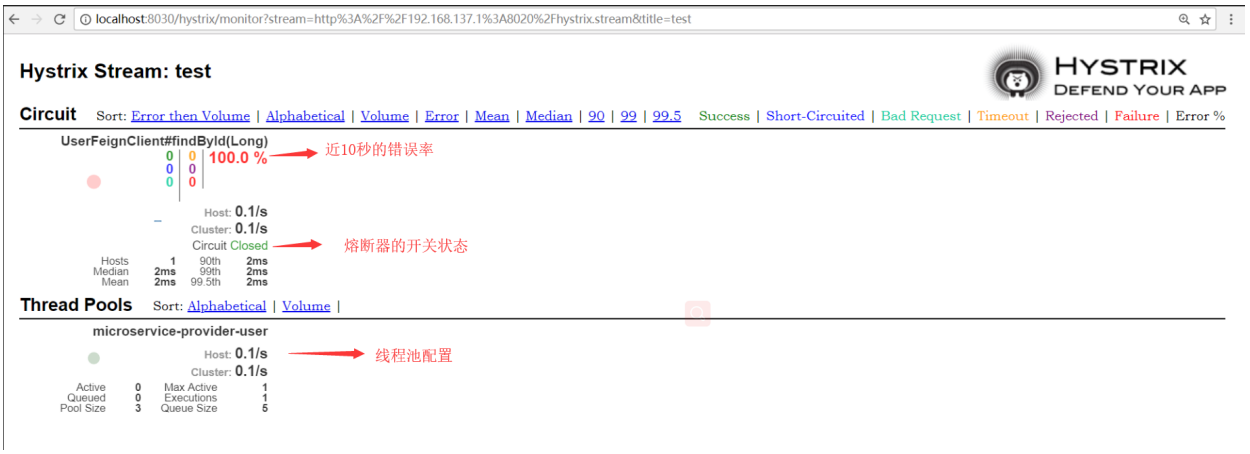
为项目添加依赖，并在启动类上增加注解@EnableHystrixDashboard

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-netflix-hystrix-dashboard</artifactId>
</dependency>
```

运行项目访问：http://localhost:8030/hystrix，效果如下图



将上一个项目的查看监控数据地址输入，在title里任意输入一个监控名称test，点击Monitor Stream按钮，效果如下图(须先运行至少一次订单服务的用户查询接口才能显示数据，否则无监控数据)



在监控的界面有两个重要的图形信息：一个实心圆和一条曲线。

实心圆：1、通过颜色的变化代表了实例的健康程度，健康程度从绿色、黄色、橙色、红色递减。2、通过大小表示请求流量发生变化，流量越大该实心圆就越大。所以可以在大量的实例中快速发现故障实例和高压实例。

曲线：用来记录2分钟内流浪的相对变化，可以通过它来观察流量的上升和下降趋势。

使用Turbine聚合监控数据

Turbine是一个聚合 Hystrix监控数据的工具，它可将所有相关/hystrix.stream端点的数据聚合到一个组合的/turbine.stream中，从而让集群的监控更加方便。

见示例：07-ms-hystrix-turbine

添加依赖，并在启动类上增加注解@EnableTurbine

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-netflix-turbine</artifactId>
</dependency>
```

编写配置文件application.yml

```
server:
  port: 8031
spring:
  application:
    name: microservice-hystrix-turbine
eureka:
  client:
    service-url:
      defaultZone: http://localhost:8761/eureka/
  instance:
    prefer-ip-address: true
turbine:
  appConfig: microservice-consumer-order,microservice-consumer-order-feign-hystrix-fallback-stream
  clusterNameExpression: "'default'"
```

使用以上配置，Turbine会在Eureka Server中找到microservice-consumer-order和microservice-consumer-order-feign-hystrix-fallback-stream这两个微服务，并聚合两个微服务的监控数据。

运行如下所有项目，详细步骤参看视频

1、运行07-ms-eureka-server项目

2、运行07-ms-provider-user项目

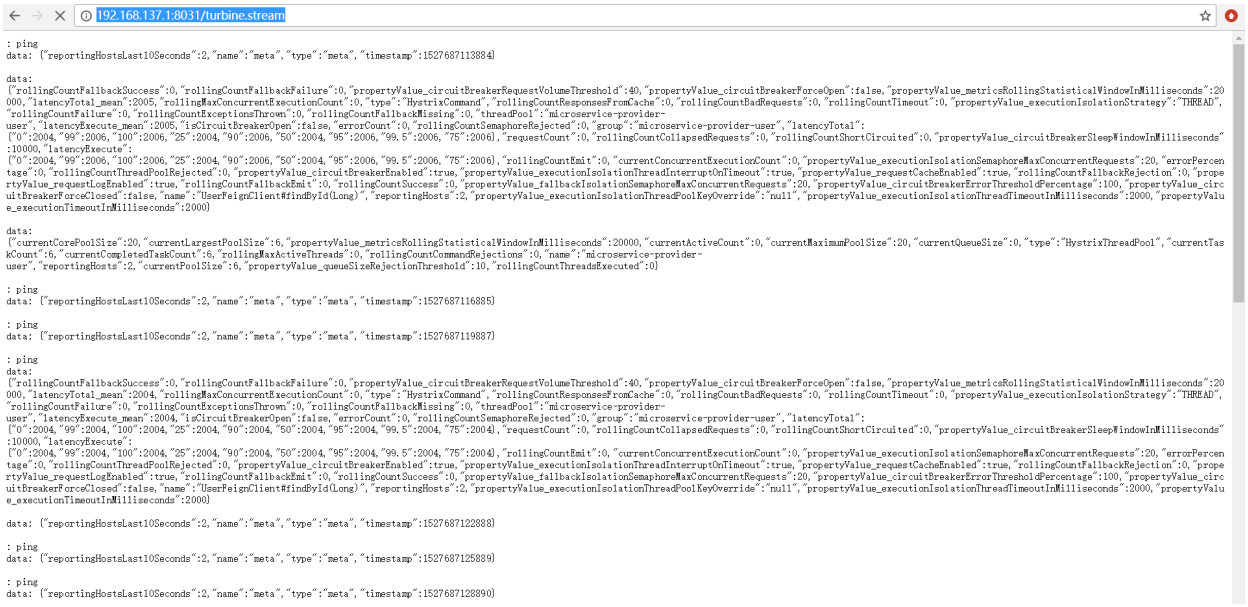
3、运行06-ms-consumer-order-ribbon-hystrix-fallback项目

4、运行07-ms-consumer-order-feign-hystrix-fallback-stream项目

5、运行07-ms-hystrix-turbine项目

6、运行07-ms-hystrix-dashboard项目

访问turbine监控地址：http://192.168.137.1:8031/turbine.stream，可以看到turbine的聚合监控数据，效果如下



访问dashboard地址：http://localhost:8030/hystrix，将该turbine的监控地址输入dashboard，效果如下图：