SpringCloud Alibaba 微服务实战二十 - 集成 Feign 的降级熔断

作者：jianzh5
原文链接：https://ld246.com/article/1603846717886
来源网站：链滴

在之前的项目中我们已经实现了使用Feign调用远程接口，本章内容主要是借助sentinel实现Feign接熔断器功能。

概述

首先我们看看不使用熔断器的情况下调用一个没有启动的服务会出现什么效果，然后再来看看使用sentnel熔断器后的效果。

如上，我们使用order-service中 FeignController调用account-service中的接口，在没启用熔断器情况下，接口会抛出500异常。

实现

使用sentinel实现熔断器很简单，简单几步即可。

1. 定义fallback类，当熔断时返回默认数据

```
package com.javadaily.feign.fallback;
@Slf4j
public class AccountFeignFallback implements AccountFeign {
    @Setter
    private Throwable cause;

    @Override
    public ResultData<String> insert(AccountDTO accountDTO) {
        return ResultData.fail("接口熔断");
    }

    @Override
    public ResultData<String> delete(String accountCode) {
        return ResultData.fail("接口熔断");
    }

    @Override
    public ResultData<String> update(AccountDTO accountDTO) {
        return ResultData.fail("接口熔断");
    }

    @Override
    public ResultData<AccountDTO> getByCode(String accountCode) {
        log.error("查询失败,接口异常" ,cause);
        AccountDTO account = new AccountDTO();
        account.setAccountCode("000");
```

```
        account.setAccountName("测试Feign");
        return ResultData.success(account);
    }

    @Override
    public ResultData<String> reduce(String accountCode, BigDecimal amount) {
        return ResultData.fail("接口熔断");
    }
}
```

2. 编写FallbackFactory

```
@Component
public class AccountFeignFallbackFactory implements FallbackFactory<AccountFeign> {
    @Override
    public AccountFeign create(Throwable throwable) {
        AccountFeignFallback accountFeignFallback = new AccountFeignFallback();
        accountFeignFallback.setCause(throwable);
        return accountFeignFallback;
    }
}
```

3. 给feign接口指定熔断工厂

```
@FeignClient(name = "account-service",fallbackFactory = AccountFeignFallbackFactory.class)
public interface AccountFeign {
    ...
}
```

4. 在消费中配置文件中开启熔断

```
feign:
  sentinel:
    enabled: true
```

经过以上四步我们就可以实现了接口的熔断，接下来重新启动order-service验证结果。系统居然无法常启动！！！

堆栈信息如下：

```
2020-10-23 15:22:14,286 ERROR SpringApplication:826 - Application run failed
org.springframework.beans.factory.UnsatisfiedDependencyException: Error creating bean with name 'feignController': Unsatisfied dependency expressed through field 'accountFeign'; neste exception is org.springframework.beans.factory.BeanCreationException: Error creating bean wth name 'com.javadaily.feign.account.AccountFeign': FactoryBean threw exception on object creation; nested exception is java.lang.IllegalStateException: No fallbackFactory instance of type class com.javadaily.feign.factory.AccountFeignFallbackFactory found for feign client account-srvice
    at org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor AutowiredFieldElement.inject(AutowiredAnnotationBeanPostProcessor.java:643) ~[spring-beas-5.2.4.RELEASE.jar:5.2.4.RELEASE]
    at org.springframework.beans.factory.annotation.InjectionMetadata.inject(InjectionMetada a.java:130) ~[spring-beans-5.2.4.RELEASE.jar:5.2.4.RELEASE]
```

问题解决

俗话说**出现问题不可怕，可怕的是我们害怕出现问题**，看上面的启动日志应该是无法找到 AccountFeinFallbackFactory，接下来我们看一下FeignClient的源码：

```
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface FeignClient {
    ...省略部分代码，保留关键代码...

    /**
     * Fallback class for the specified Feign client interface. The fallback class must
     * implement the interface annotated by this annotation and be a valid spring bean.
     * @return fallback class for the specified Feign client interface
     */
    Class<?> fallback() default void.class;
    ...省略部分代码，保留关键代码...
}
```

注意看注释部分，说的是Feign的降级类必须实现该Feign接口并且必须是一个Spring Bean。

出现错误的原因应该是**降级类没有被注册成Spring Bean。**

大家都知道，在SpringBoot启动的时候会默认扫描主启动类所在的包以及子包进行Bean实例化。

项目中的order-service的主启动类位于 com.javadaily.order，那么order-service项目启动时只会扫到 com.javadaily.order以及 com.javadaily.order的子包，而AccountFeignClient的降级类 AccounteignFallback的包路径为 com.javadaily.feign.fallback ，这样就无法被Spring实例化，最终导致项启动失败。

既然定位到了问题原因那就很好解决了，只要在启动类上配置Feign降级类的包路径即可。

```
@SpringBootApplication(scanBasePackages = {"com.javadaily.feign"})
```

加上这个配置后系统能正常启动，但是调用接口又返回如下的错误：

```
{
```

```
  "timestamp": "2020-10-27T03:30:44.664+0000",
  "status": 401,
  "error": "Unauthorized",
  "message": "Unauthorized",
  "path": "/order/getAccount/jianzh5"
}
```

出现这个问题的原因是我们通过scanBasePackages配置Feign降级类的路径后自身的Bean无法实例，所以我们还需要配置上我们自己项目的扫描路径，即：

```
@SpringBootApplication(scanBasePackages = {"com.javadaily.feign","com.javadaily.order"})
```

至此所有问题都解决，我们再次访问接口，当系统故障时返回我们默认结果。

小结

本章内容我们通过给Feign的接口加上熔断器，当实例故障时系统会返回默认数据。这样就不会出现某一个服务不可用时导致他的消费者长时间等待，线程池耗尽，进而影响到其他服务的线程调用，这是常说的 "雪崩效应"。

当然了实现过程中出现了一点小挫折，总结下来就是**如果各位的Feign客户端是由消费者自己编写，于消费者自己模块不会出现这个问题。如果是由生产者编写并提供则需要注意Spring Bean实例化的描路径，如果无法扫描实例化熔断类，只需要在启动类上通过 scanBasePackages扫描到对应的路即可。**