feign中使用Hystrix灵活控制接口超时（降级）

"FeignCTestFF#person1()":  
 execution:  
 isolation:  
 thread:  
 timeoutInMilliseconds: 2000  
 timeout:  
 enabled: true

下面是原理：

[**feignclient设置hystrix参数**](https://www.cnblogs.com/tiancai/p/9585005.html)

**序**

feign默认集成了hystrix，那么问题来了，如何像hystrix command那样设置每个方法的hystrix属性呢。

**实例**

@FeignClient("product")

public interface RemoteProductService {

@RequestMapping(method = RequestMethod.GET,value = "/product/{productId}")

public Product getProduct(@PathVariable(value = "productId") int productId);

}

**FeignClientsConfiguration**

spring-cloud-netflix-core-1.2.6.RELEASE-sources.jar!/org/springframework/cloud/netflix/feign/FeignClientsConfiguration.java

@Configuration

@ConditionalOnClass({ HystrixCommand.class, HystrixFeign.class })

protected static class HystrixFeignConfiguration {

@Bean

@Scope("prototype")

@ConditionalOnMissingBean

@ConditionalOnProperty(name = "feign.hystrix.enabled", matchIfMissing = true)

public Feign.Builder feignHystrixBuilder() {

return HystrixFeign.builder();

}

}

**HystrixFeign**

feign-hystrix-9.3.1-sources.jar!/feign/hystrix/HystrixFeign.java

private SetterFactory setterFactory = new SetterFactory.Default();

**SetterFactory**

feign-hystrix-9.3.1-sources.jar!/feign/hystrix/SetterFactory.java

public interface SetterFactory {

/\*\*

\* Returns a hystrix setter appropriate for the given target and method

\*/

HystrixCommand.Setter create(Target<?> target, Method method);

/\*\*

\* Default behavior is to derive the group key from {@link Target#name()} and the command key from

\* {@link Feign#configKey(Class, Method)}.

\*/

final class Default implements SetterFactory {

@Override

public HystrixCommand.Setter create(Target<?> target, Method method) {

String groupKey = target.name();

String commandKey = Feign.configKey(target.type(), method);

return HystrixCommand.Setter

.withGroupKey(HystrixCommandGroupKey.Factory.asKey(groupKey))

.andCommandKey(HystrixCommandKey.Factory.asKey(commandKey));

}

}

}

groupKey，这里为product，即@FeignClient("product")中的值。

**Feign.configKey**

feign-core-9.3.1-sources.jar!/feign/Feign.java

public static String configKey(Class targetType, Method method) {

StringBuilder builder = new StringBuilder();

builder.append(targetType.getSimpleName());

builder.append('#').append(method.getName()).append('(');

for (Type param : method.getGenericParameterTypes()) {

param = Types.resolve(targetType, targetType, param);

builder.append(Types.getRawType(param).getSimpleName()).append(',');

}

if (method.getParameterTypes().length > 0) {

builder.deleteCharAt(builder.length() - 1);

}

return builder.append(')').toString();

}

commandKey的构造，这里组装了类、方法名、参数，比如本文的实例，commandKey=RemoteProductService#getProduct(int)

**配置文件指定**

hystrix:

command:

"RemoteProductService#getProduct(int)":

execution:

isolation:

thread:

timeoutInMilliseconds: 500

**Java中指定**

@Bean

public Feign.Builder feignHystrixBuilder() {

return HystrixFeign.builder().setterFactory(new SetterFactory() {

@Override

public HystrixCommand.Setter create(Target<?> target, Method method) {

return HystrixCommand.Setter

.withGroupKey(HystrixCommandGroupKey.Factory.asKey(RemoteProductService.class.getSimpleName()))// 控制 RemoteProductService 下,所有方法的Hystrix Configuration

.andCommandPropertiesDefaults(

HystrixCommandProperties.Setter().withExecutionTimeoutInMilliseconds(10000) // 超时配置

);

}

});

}

**小结**

论灵活程度，还是配置文件灵活一点，唯一的工作量就是根据规则构造commandKey，然后就可以进行相关配置了。

import com.netflix.hystrix.HystrixCommand;

HystrixCommand.Setter

自定义commandkey

**Feign-灵活的使用Hystrix熔断（自定义CommandKey）**

Feign可以直接去集成Hystrix熔断。具体配置：[Hystrix熔断&&Feign熔断](https://www.jianshu.com/p/2b7bb15d6a71)

但是配置时，却不是很灵活，只是支持default和类名#方法名()的配置。这就不能对类或者一组方法进行统一的配置。

<https://www.jianshu.com/p/fcdff4eb05b0>

**2.1 Hystrix配置的优先级**

**优先级从低到高的配置：**

1. 内置全局属性默认值：写死在代码中的值；
2. 动态全局默认属性：通过配置文件配置全局的值；
3. 内置实例默认值：写死在代码中的实例的值；
4. 动态配置实例属性：通过配置文件配置特定实例的值；

例子在gitee上面：

git@gitee.com:astorage/consul-discovery.git

*# 用例包括  
## 1、consul服务发现  
## 2、使用loadBalancerClient和discoveryClient调用服务  
## 3、使用feign调用远程服务  
## 4、feign使用hystrix来熔断、降级  
## 5、直接使用hystrix来熔断、降级（用起来麻烦些，不能负载均衡，不好用在微服务中），可以借助第二调中的来实现负载均衡，不过有太麻烦。  
## 6、hystrix降级打印错误信息，不同commandkey更精细的控制降级策略。而不是用default来控制所有的。*