乐字节教育高级架构课程

正所谓“授人以鱼不如授人以渔”，你们想要的 Java 学习资料来啦！

不管你是学生，还是已经步入职场的同行，希望你们都要珍惜眼前的学习机会，奋斗没有终点，知识永不过时。

扫描下方二维码即可领取

乐字节晓啡

乐字节官方交流群

网关熔断

在 Edgware 版本之前，Zuul 提供了接口 `ZuulFallbackProvider` 用于实现 fallback 处理。从 Edgware 版本开始，Zuul 提供了接口 `FallbackProvider` 来提供 fallback 处理。

Zuul 的 fallback 容错处理逻辑，只针对 timeout 异常处理，当请求被 Zuul 路由后，**只要服务有返回（包括异常），都不会触发 Zuul 的 fallback 容错逻辑。**

> 因为对于Zuul网关来说，做请求路由分发的时候，结果由远程服务运算。远程服务反馈了异常信息，Zuul 网关不会处理异常，因为无法确定这个错误是否是应用程序真实想要反馈给客户端的。

代码示例

ProductProviderFallback.java

```
package com.example.fallback;

import org.springframework.cloud.netflix.zuul.filters.route.FallbackProvider;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.client.ClientHttpResponse;
import org.springframework.stereotype.Component;

import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.nio.charset.Charset;

/**
 * 对商品服务做服务容错处理
 */
@Component
public class ProductProviderFallback implements FallbackProvider {

    /**
     * return - 返回 fallback 处理哪一个服务。返回的是服务的名称。
     * 推荐 - 为指定的服务定义特性化的 fallback 逻辑。
     * 推荐 - 提供一个处理所有服务的 fallback 逻辑。
     * 好处 - 某个服务发生超时，那么指定的 fallback 逻辑执行。如果有新服务上线，未提供 fallback 逻辑，有一个通用的。
     */
    @Override
    public String getRoute() {
        return "product-service";
    }
```

```
    /**
     * 对商品服务做服务容错处理
     *
     * @param route 容错服务名称
     * @param cause 服务异常信息
     * @return
     */
    @Override
    public ClientHttpResponse fallbackResponse(String route, Throwable cause) {
        return new ClientHttpResponse() {
            /**
             * 设置响应的头信息
             * @return
             */
            @Override
            public HttpHeaders getHeaders() {
                HttpHeaders header = new HttpHeaders();
                header.setContentType(new MediaType("application", "json",
Charset.forName("utf-8")));
                return header;
            }

            /**
             * 设置响应体
             * Zuul 会将本方法返回的输入流数据读取，并通过 HttpServletResponse 的输出流输出到客户端。
             * @return
             */
            @Override
            public InputStream getBody() throws IOException {
                return new ByteArrayInputStream("{\"message\":\"商品服务不可用，请稍后再试。
\"}".getBytes());
            }

            /**
             * ClientHttpResponse 的 fallback 的状态码 返回 HttpStatus
             * @return
             */
            @Override
            public HttpStatus getStatusCode() throws IOException {
                return HttpStatus.INTERNAL_SERVER_ERROR;
            }

            /**
             * ClientHttpResponse 的 fallback 的状态码 返回 int
             * @return
             */
            @Override
            public int getRawStatusCode() throws IOException {
                return this.getStatusCode().value();
            }

            /**
```

```
             * ClientHttpResponse 的 fallback 的状态码 返回 String
             * @return
             */
            @Override
            public String getStatusText() throws IOException {
                return this.getStatusCode().getReasonPhrase();
            }

            /**
             * 回收资源方法
             * 用于回收当前 fallback 逻辑开启的资源对象。
             */
            @Override
            public void close() {
            }
        };
    }

}
```

访问

关闭商品服务，访问：http://localhost:9000/product-service/product/1?token=abc123 结果如下：