[乱码](https://so.csdn.net/so/search?q=%E4%B9%B1%E7%A0%81&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/qq_38380025/article/details/_blank)问题其实就是返回数据gateway 进行了截断所产生的，所以我们需要把返回的组进行join上就不会产生乱码的，如果还是乱码，请根据自己系统的实际编码进行转换即可，完美解决，

这里使用的是 DefaultDataBufferFactory 的join方法去合并多个dataBuffers ↓↓↓↓↓↓↓↓↓↓↓↓↓↓ 最大的区别

DataBufferFactory dataBufferFactory = new DefaultDataBufferFactory();

DataBuffer join = dataBufferFactory.join(dataBuffers);

byte[] content = new byte[join.readableByteCount()];

join.read(content);

DataBufferUtils.release(join);

String responseData = new String(content, StandardCharsets.UTF\_8);

详细代码

import com.google.common.base.Charsets;

import lombok.extern.slf4j.Slf4j;

import org.reactivestreams.Publisher;

import org.springframework.cloud.gateway.filter.GatewayFilterChain;

import org.springframework.cloud.gateway.filter.GlobalFilter;

import org.springframework.core.Ordered;

import org.springframework.core.io.buffer.DataBuffer;

import org.springframework.core.io.buffer.DataBufferFactory;

import org.springframework.core.io.buffer.DataBufferUtils;

import org.springframework.core.io.buffer.DefaultDataBufferFactory;

import org.springframework.http.HttpStatus;

import org.springframework.http.server.reactive.ServerHttpResponse;

import org.springframework.http.server.reactive.ServerHttpResponseDecorator;

import org.springframework.stereotype.Component;

import org.springframework.web.server.ServerWebExchange;

import reactor.core.publisher.Flux;

import reactor.core.publisher.Mono;

@Component

@Slf4j

public class DangDeleDangDangYiDangDangFilter implements GlobalFilter, Ordered {

@Override

public int getOrder() {

//-1 is response write filter, must be called before that

return -2;

}

@Override

public Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain) {

try {

//获取response的 返回数据

ServerHttpResponse originalResponse = exchange.getResponse();

DataBufferFactory bufferFactory = originalResponse.bufferFactory();

HttpStatus statusCode = originalResponse.getStatusCode();

if (statusCode == HttpStatus.OK) {

ServerHttpResponseDecorator decoratedResponse = new ServerHttpResponseDecorator(originalResponse) {

@Override

public Mono<Void> writeWith(Publisher<? extends DataBuffer> body) {

// 判断服务返回的数据类型进行拦截，根据自己的业务进行修改

if (APPLICATION\_JSON.isCompatibleWith(getDelegate().getHeaders().getContentType())) {

Flux<? extends DataBuffer> fluxBody = Flux.from(body);

return super.writeWith(fluxBody.buffer().map(dataBuffers -> {

DataBufferFactory dataBufferFactory = new DefaultDataBufferFactory();

DataBuffer join = dataBufferFactory.join(dataBuffers);

byte[] content = new byte[join.readableByteCount()];

join.read(content);

DataBufferUtils.release(join);

String responseData = new String(content, Charsets.UTF\_8);

if (log.isDebugEnabled()) {

log.debug("响应转前:{}", responseData);

}

responseData = responseData.replaceAll(":null", ":\"\"");

if (log.isDebugEnabled()) {

log.debug("响应转后:{}", responseData);

}

byte[] uppedContent = responseData.getBytes(Charsets.UTF\_8);

return bufferFactory.wrap(uppedContent);

}));

} else {

return chain.filter(exchange);

}

}

};

return chain.filter(exchange.mutate().response(decoratedResponse).build());

}

return chain.filter(exchange);

} catch (Exception e) {

log.error(" ReplaceNullFilter 异常", e);

return chain.filter(exchange);

}

}

}