链滴

okHttp 重试

作者：289306290
原文链接：https://ld246.com/article/1565775068679
来源网站：链滴

```
package club.wujingjian.util;

import com.fasterxml.jackson.databind.DeserializationFeature;
import com.fasterxml.jackson.databind.ObjectMapper;
import lombok.extern.slf4j.Slf4j;
import okhttp3.ConnectionPool;

import java.nio.charset.Charset;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.concurrent.TimeUnit;
import okhttp3.OkHttpClient.Builder;
import org.springframework.http.*;
import org.springframework.http.client.OkHttp3ClientHttpRequestFactory;
import org.springframework.http.converter.HttpMessageConverter;
import org.springframework.http.converter.StringHttpMessageConverter;
import org.springframework.http.converter.json.Jackson2ObjectMapperBuilder;
import org.springframework.http.converter.json.MappingJackson2HttpMessageConverter;
import org.springframework.util.ClassUtils;
import org.springframework.web.client.RestTemplate;

@Slf4j
public class OKHttpUtil {

    private static final boolean jackson2Present = ClassUtils.isPresent("com.fasterxml.jackson.d
tabind.ObjectMapper",
            RestTemplate.class.getClassLoader())
            && ClassUtils.isPresent("com.fasterxml.jackson.core.JsonGenerator", RestTemplate.cla
s.getClassLoader());

    private static final boolean springwebPresent = ClassUtils.isPresent(
            "org.springframework.http.converter.json.Jackson2ObjectMapperBuilder", RestTempla
e.class.getClassLoader());


    private int readTimeout = 2000;
    private int connectTimeout=1000;
    private boolean isRetry = true;
    private int retryCount = 3;
    private long retryDelay = 100;


    public okhttp3.OkHttpClient okHttpClient() {
        Builder builder = new Builder();
        ConnectionPool pool = new ConnectionPool(1000, 50, TimeUnit.MINUTES);
        builder.connectionPool(pool);
        builder.retryOnConnectionFailure(isRetry);
//        RetryInterceptor retryIntercepter = new RetryInterceptor(retryCount, retryDelay,new Ar
ayList<>());
        RetryInterceptor retryIntercepter = new RetryInterceptor(retryCount, retryDelay);
        builder.addInterceptor(retryIntercepter);
        builder.followRedirects(false);
        return builder.build();
```

```
    }

    public RestTemplate restClientOnlyTemplateSimple() {

        OkHttp3ClientHttpRequestFactory okHttp3ClientHttpRequestFactory = new OkHttp3Clie
tHttpRequestFactory(
                okHttpClient());
        okHttp3ClientHttpRequestFactory.setReadTimeout(readTimeout);
        okHttp3ClientHttpRequestFactory.setConnectTimeout(connectTimeout);
        RestTemplate restTemplate = new RestTemplate(okHttp3ClientHttpRequestFactory);

        // 使用 utf-8 编码集的 convert 替换默认的 convert（默认的 string conver 的编码集为
        // "ISO-8859-1")
        List<HttpMessageConverter<?>> messageConverters = restTemplate.getMessageConve
ters();
        Iterator<HttpMessageConverter<?>> iterator = messageConverters.iterator();
        while (iterator.hasNext()) {
            HttpMessageConverter<?> converter = iterator.next();
            if (converter instanceof StringHttpMessageConverter) {
                iterator.remove();
            }
            if (converter instanceof MappingJackson2HttpMessageConverter) {
                iterator.remove();
            }
        }

        StringHttpMessageConverter stringHttpMessageConverter = new StringHttpMessageCo
verter(
                Charset.forName("UTF-8"));
        stringHttpMessageConverter.setWriteAcceptCharset(false);
        List<MediaType> mediaTypeList = new ArrayList<>();
        mediaTypeList.add(MediaType.APPLICATION_JSON_UTF8);
        mediaTypeList.add(MediaType.APPLICATION_FORM_URLENCODED);
        mediaTypeList.add(MediaType.APPLICATION_JSON);
        mediaTypeList.add(MediaType.TEXT_PLAIN);
        mediaTypeList.add(MediaType.TEXT_HTML);
        mediaTypeList.add(new MediaType("text", "json"));
        mediaTypeList.add(new MediaType("text", "javascript"));
        stringHttpMessageConverter.setSupportedMediaTypes(mediaTypeList);
        messageConverters.add(0, stringHttpMessageConverter);

        // 兼容JSON与实体字段不对应
        if (jackson2Present && springwebPresent) {
            ObjectMapper objectMapper = Jackson2ObjectMapperBuilder.json().build();
            objectMapper.configure(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES, fal
e);
            MappingJackson2HttpMessageConverter jsonConverter = new MappingJackson2Htt
MessageConverter(objectMapper);
            messageConverters.add(jsonConverter);
        }

        return restTemplate;
    }
```

```
    public static void main(String[] args) {
        OKHttpUtil okHttpUtil = new OKHttpUtil();
        RestTemplate restTemplate = okHttpUtil.restClientOnlyTemplateSimple();
        String url ="http://mobileself-xxx.ffff.beta/api/v2/staffinfo/exist/xxx";
        HttpHeaders headers = new HttpHeaders();
        MediaType type = MediaType.parseMediaType("application/json;charset=UTF-8");
        headers.setContentType(type);
        headers.set("Accept", MediaType.APPLICATION_JSON_VALUE);
        headers.set("xiaofanAuthorization","aaa#aaa");
        HttpEntity<?> entity = new HttpEntity<>(headers);
        try {
            ResponseEntity<String> response = restTemplate.exchange(url, HttpMethod.POST, en
ity, String.class);
            if (null != response) {
                System.out.println("响应数据:" + response.getBody());
            }
        } catch (Exception e) {
            log.error("请求小凡接口异常:"+e.getMessage(),e);
        }

    }
}


package club.wujingjian.util;

import java.io.IOException;
import java.io.InterruptedIOException;
import java.net.ConnectException;
import java.net.UnknownHostException;
import java.nio.charset.Charset;
import java.util.Arrays;
import java.util.Collection;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;
import javax.net.ssl.SSLException;
import okhttp3.Interceptor;
import okhttp3.Request;
import okhttp3.Response;
import org.apache.commons.collections.CollectionUtils;
import org.apache.http.conn.HttpHostConnectException;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public final class RetryInterceptor implements Interceptor {
    private static Logger logger = LoggerFactory.getLogger(RetryInterceptor.class);
    private final int retryCount;
    private final long retryDelay;
    private final Set<Class<? extends IOException>> nonRetriableClasses;//不会重试的类型
    private static final Charset UTF8 = Charset.forName("UTF-8");

    protected RetryInterceptor(int retryCount, long retryDelay, Collection<Class<? extends IOE
```

```
ception>> clazzes) {
        this.retryCount = retryCount;
        this.retryDelay = retryDelay;
        this.nonRetriableClasses = new HashSet();
        if (CollectionUtils.isNotEmpty(clazzes)) {
            Iterator noRetryIterator = clazzes.iterator();

            while(noRetryIterator.hasNext()) {
                Class<? extends IOException> clazz = (Class)noRetryIterator.next();
                this.nonRetriableClasses.add(clazz);
            }
        }

    }

    public RetryInterceptor(int retryCount, long retryDelay) {
        this(retryCount, retryDelay, Arrays.asList(InterruptedIOException.class, UnknownHostExc
ption.class, SSLException.class, HttpHostConnectException.class,ConnectException.class));
    }

    public RetryInterceptor() {
        this(3, 50L);
    }

    public Response intercept(Chain chain) throws IOException {
        Response response = null;
        try {
            Request request = chain.request();
            for (int execCount = 0; execCount <= retryCount; execCount++) {
                if (execCount >= 1) {
                    logger.info("重试第{}次,url:{}", execCount, request.url().url());
                }
                try {
                    Exception exception;
                    Iterator varException;
                    try {
                        response = chain.proceed(request);
                        break;
                    } catch (Exception ex) {
                        exception = ex;
                        if (execCount >= this.retryCount) {
                            throw ex;
                        }
                        if (this.nonRetriableClasses.contains(ex.getClass())) {
                            throw ex;
                        }
                        varException = this.nonRetriableClasses.iterator();
                    }

                    while (varException.hasNext()) {
                        Class<? extends IOException> rejectException = (Class) varException.next();
                        if (rejectException.isInstance(exception)) {
                            throw exception;
                        }
```

```
                }

                try {
                    Thread.sleep(this.retryDelay);
                } catch (InterruptedException interruptedException) {
                    logger.error("重试错误:" + interruptedException.getMessage(), interruptedExce
tion);
                }
            } finally {
                if (execCount >= 1) {
                    logger.info("重试{}次结束,url:{}", execCount, request.url().url());
                }

            }
        }

    } catch (Exception e) {
        logger.error("请求错误了:"+e.getMessage(),e);
    }
    return response;
}


}
```

原文链接：okHttp 重试