链滴

基于 Okhttp 网络请求框架的封装

作者：pencilso

原文链接：https://ld246.com/article/1539157637798

来源网站：链滴

引入Okhttp依赖

```
<dependency>
    <groupId>com.squareup.okhttp3</groupId>
    <artifactId>okhttp</artifactId>
    <version>3.9.1</version>
    <scope>compile</scope>
</dependency>
```

网络请求

普通类型的网络请求

```
String url = "xxx.com/api/queryUserInfo";

Map<Object,Object> params = new HashMap<>();
Map<Object, Object> headers = new HashMap<>();
//添加请求体参数
params.put("userid","123456789");
//添加请求头参数
headers.put("token","41f05413-381c-4060-b8a1-acc6c5997164");
//GET 请求
String respone = okHttpPlugin.http(url, OkHttpPlugin.HttpMethod.GET, params, headers);
//POST 请求
String respone = okHttpPlugin.http(url, OkHttpPlugin.HttpMethod.POST, params, headers);
```

POST请求 提交JSON数据

```
String url = "xxx.com/api/queryUserInfo";

Map<Object,Object> body = new HashMap<>();
body.put("userid","123456789");

//序列化为JSON字符串
String bodyJson = GsonPlugin.toJson(body);

//提交请求
String respone = okHttpPlugin.http(url, bodyJson, OkHttpPlugin.MEDIA_TYPE_JSON);
```

下载文件 | 获取原始的Respone

```
String url = "xxx.com/api/queryUserInfo";

Map<Object,Object> params = new HashMap<>();
Map<Object, Object> headers = new HashMap<>();
//添加请求体参数
params.put("userid","123456789");
//添加请求头参数
headers.put("token","41f05413-381c-4060-b8a1-acc6c5997164");
```

```
    //GET 请求
    Response respone = okHttpPlugin.httpResponeBody(url, OkHttpPlugin.HttpMethod.GET, p
rams, headers);
    //POST 请求
    Response respone = okHttpPlugin.httpResponeBody(url, OkHttpPlugin.HttpMethod.POST,
arams, headers);

    //获取响应头
    Headers responeHeaders = respone.headers();

    //获取字符串 （如果响应格式是字符串的话 可以直接获取）
    String string = respone.body().string();

    //获取输入流
    InputStream inputStream = respone.body().byteStream();


    FileOutputStream outputStream = null;
    try {
      //下载文件
       outputStream = new FileOutputStream(new File("D://123456.mp4"));
      byte by[] = new byte[1024];
      int len = -1;
      while ((len = inputStream.read(by))!=-1){
          outputStream.write(by,0,len);
      }
   }catch (Exception e){
      e.printStackTrace();
   }finally {
      if (respone!=null) {
          respone.close();
      }
      if (outputStream!=null) {
          try {
              outputStream.close();
          } catch (IOException e) {
              e.printStackTrace();
          }
      }
   }
```

OkHttp工具类

```
import com.yscall.kulaidian.constant.ConstantConfig;
import com.yscall.kulaidian.utils.EmptyUtils;
import okhttp3.*;
import org.springframework.stereotype.Component;

import java.util.Map;
import java.util.Set;
import java.util.concurrent.TimeUnit;

/**
```

```
 * Created by Pencilso on 2018/10/10/012.
 * 网络请求框架
 *
 * @author Pencilso
 */
@Component
public class OkHttpPlugin {
    public static final MediaType MEDIA_TYPE_XML = MediaType.parse("application/xml");
    public static final MediaType MEDIA_TYPE_JSON = MediaType.parse("application/json");

    private static OkHttpClient okHttpClient = new OkHttpClient.Builder()
            .connectTimeout(ConstantConfig.Reptilian.REPTILIAN_VIDEO_TIMEOUT, TimeUnit.MILL
SECONDS)
            .readTimeout(ConstantConfig.Reptilian.REPTILIAN_VIDEO_TIMEOUT, TimeUnit.MILLIS
CONDS)
            .build();

    public enum HttpMethod {
        POST, GET
    }

    public Response httpResponeBody(String url, HttpMethod httpMethod) {
        return httpResponeBody(url, httpMethod, null);
    }

    public Response httpResponeBody(String url, HttpMethod httpMethod, Map<Object, Obje
t> paramster) {
        return httpResponeBody(url, httpMethod, paramster, null);
    }

    public Response httpResponeBody(String url, HttpMethod httpMethod, Map<Object, Obje
t> paramster, Map<Object, Object> header) {
        try {
            Request request = builderRequest(url, httpMethod, paramster, header);
            Response execute = okHttpClient.newCall(request).execute();
            return execute;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }

    public String http(String url, HttpMethod httpMethod) {
        return http(url, httpMethod, null, null);
    }

    public String http(String url, HttpMethod httpMethod, Map<Object, Object> paramster) {
        return http(url, httpMethod, paramster, null);
    }

    public String http(String url, HttpMethod httpMethod, Map<Object, Object> paramster, M
p<Object, Object> header) {
        Response response = httpResponeBody(url, httpMethod, paramster, header);
        try {
```

```
            String string = response.body().string();
            return string;
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (response != null) {
                response.close();
            }
        }
        return null;
    }


    public String http(String url, String context, MediaType mediaType, Map<Object, Object>
eader) {
        try {
            RequestBody body = RequestBody.create(mediaType, context);
            Request.Builder post = new Request.Builder()
                    .url(url)
                    .post(body);
            if (header != null && header.size() != 0) {
                Set<Map.Entry<Object, Object>> entries = header.entrySet();
                for (Map.Entry<Object, Object> entry : entries) {
                    String key = (String) entry.getKey();
                    String value = (String) entry.getValue();
                    if (EmptyUtils.isNull(key, value)) continue;
                    post.addHeader(key, value);
                }
            }
            Request build = post.build();
            Response execute = okHttpClient.newCall(build).execute();
            return execute.body().string();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }

    public String http(String url, String context, MediaType mediaType) {
        return http(url, context, mediaType, null);
    }

    private Request builderRequest(String url, HttpMethod httpMethod, Map<Object, Object>
paramster, Map<Object, Object> header) {
        Request.Builder builder = new Request.Builder();
        if (httpMethod == HttpMethod.POST) {
            //POST 请求
            FormBody.Builder formBody = new FormBody.Builder();
            forEachMap(paramster, ((key, value) -> formBody.add(key, value)));
            builder.post(formBody.build()).url(url);
        } else if (httpMethod == HttpMethod.GET) {
            //GET 请求
            if (paramster != null && paramster.size() != 0) {
                StringBuffer sf = new StringBuffer("?");
```

```
                forEachMap(paramster, ((key, value) -> sf.append(key).append("=").append(value).a
pend("&")));
                url += sf.toString();
            }
            builder.url(url);
        }
        //统一添加请求头
        forEachMap(header, (key, value) -> builder.addHeader(key, value));
        return builder.build();
    }

    private void forEachMap(Map<Object, Object> map, MapForInterface mapForInterface) {
        if (map != null && map.size() != 0) {
            Set<Map.Entry<Object, Object>> entries = map.entrySet();
            for (Map.Entry<Object, Object> entry : entries) {
                mapForInterface.result(String.valueOf(entry.getKey()), String.valueOf(entry.getValue(
));
            }
        }
    }


    interface MapForInterface {
        void result(String key, String value);
    }
}
```