API网关Java-SDK使用指南

1 SDK简介

欢迎使用API网关开发者工具套件(SDK)。API网关SDK是根据您自定义的所有API接口，自动生成的Java调用代码，让您无需复杂编程即可访问阿里云的API网关服务。这里向您介绍如何使用API网关SDK。

需要注意的一点是，所有API和文档都会按照RegionId、Group分组。下文中所有出现的{{group}}都是指API所属Group的名称，{{regionId}}都是指Group所在的地域(可用区)。

代码文件的层级结构如下：

- SDK文件夹
  - sdk/{{regionId}} JavaSDK文件夹，包含每个Group的所有API的接口调用代码
    - HttpApiClient_{{group}}.java 包含对应Group所有HTTP通道的API方法
    - HttpsApiClient_{{group}}.java 包含对应Group所有HTTPS通道API方法
    - WebSocketApiClient_{{group}}.java 包含对应Group所有WebSocket通道的API法
    - Demo_{{group}}.java 包含对应Group所有API调用示例
  - doc/{{regionId}}
    - ApiDocument_{{group}}.md 对应Group的API接口文档
  - lib
    - sdk-core-java-1.1.0.jar sdk的core包，为本sdk的依赖包
    - sdk-core-java-1.1.0-sources.jar 上述依赖包的源码
  - Readme.md 本SDK使用指南
  - LICENSE 版权许可

2 SDK使用

2.1 环境准备

1. 阿里云API网关Java SDK适用于JDK 1.6及以上版本
2. 您需要准备一对授权密钥供SDK生成鉴权和签名信息，即 AppKey和AppSecret

重要提示：AppKey和AppSecret是网关认证用户请求的密钥，这两个配置如果保存在客户端，请妥善加密。 3. 在pom.xml中添加如下依赖：

```
<dependency>
    <groupId>com.aliyun.api.gateway</groupId>
    <artifactId>sdk-core-java</artifactId>
    <version>1.1.0</version>
</dependency>
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.2.52</version>
</dependency>
```

2.2 引入SDK的API接口调用类

1. 把sdk文件夹中所有Group的通道类HttpApiClient_*.java、HttpsApiClient_*.java和WebSocketApiClient_*.java文件复制到您的项目文件夹中；
2. 修正这些类文件的package；

2.3 初始化通信通道类

要提交请求至阿里云API网关，您首选要将所有的通道类初始化。您可以参考Demo_*.java中的示例代码，使用对应的ClientBuilderParams类来初始化所有通道：

2.3.1 HTTP通道类的初始化

```
HttpClientBuilderParams httpParam = new HttpClientBuilderParams();
httpParam.setAppKey("");
httpParam.setAppSecret("");
HttpApiClient_chuxiang.getInstance().init(httpParam);
```

注意

- HttpClientBuilderParams 里面有很多构造参数，主要是对ApacheHttpClient开源组件的包装，具体参数含义可以自行搜索这个开源组件的参数说明（官方文档：https://hc.apache.org/httpcomponents-client-4.5.x/tutorial/html/index.html）。

2.3.2 HTTPS通道类的初始化

```
HttpClientBuilderParams httpsParam = new HttpClientBuilderParams();
httpsParam.setAppKey("");
httpsParam.setAppSecret("");
//忽略证书代码
//httpsParam.setRegistry(getNoVerifyRegistry());
HttpsClient_UnitTest.getInstance().init(httpsParam);
```

注意

- 如果您的SSL证书是自行生成的或者无效的，不被认可的证书，您需要使用忽略证书代码去初始化HTTPS通道才能正常调用；
- 忽略证书验证的方法是不安全的方法，证书有可能被篡改导致通信安全性降低，建议您去正规渠道购买SSL证书并配置到API网关上；
- getNoVerifyRegistry()的实现请参考带有HTTPS通道初始化的Demo中的代码，这段代码建议只在Demo测试的时候用，不建议用于生产。

2.3.3 WebSocket通道类的初始化

```
WebSocketClientBuilderParams clientParam = new WebSocketClientBuilderParams();
clientParam.setAppKey("");
clientParam.setAppSecret("");
clientParam.setApiWebSocketListner(new ApiWebSocketListner() {
    @Override
    public void onNotify(String message) {
        System.out.println("receive notice :" + message);
        notifyReturnMessage = message;
        notifyCountDown.countDown();
    }

    @Override
    public void onFailure(Throwable t, ApiResponse response) {
        t.printStackTrace();
        notifyCountDown.countDown();
    }
});
WebSocketClient_UnitTest.getInstance().init(clientParam);
```

注意

- ApiWebSocketListner用于监听双向通信接收服务器推下来的Message，如果不使用双向通信功能可以不初始化这个方法；

2.4 调用API接口

SDK是根据您在API网关自定义的参数进行生成的，每个API都被封装成了method，您可以参照Demo中的示例代码进行调用。另外，SDK为您封装了单例模式的调用方法，您可以使用`HttpClient_{{group}}.getInstance()`方法来获得通道类对象。 每个API同时提供同步和异步的调用方法，下面是调用示例：

```
//异步调用示例
public void test06_HttpCommon() throws Exception {
    HttpClient_UnitTest.getInstance().getUser(userId , new ApiCallback() {
        @Override
        public void onFailure(ApiRequest request, Exception e) {
            e.printStackTrace();
        }

        @Override
        public void onResponse(ApiRequest request, ApiResponse response) {
            try {
                System.out.println(response.getCode());
                System.out.println(response.getMessage());
                System.out.println(response.getFirstHeaderValue("x-ca-request-id"));
            }catch (Exception ex){
                ex.printStackTrace();
            }
        }
    });
}

//同步调用示例
public void test06_HttpGetUser(int userId) throws Exception {
    ApiResponse response = HttpClient_UnitTest.getInstance().getUser_syncMode(userId);
    System.out.println(response.getCode());
    System.out.println(response.getMessage());
    System.out.println(response.getFirstHeaderValue("x-ca-request-id"));
}
```

注意

- 您必须先`init()`初始化后，才可以调用`getInstance()`方法，否则会抛出异常。
- 建议使用异步调用，在整个等待应答期间，主线程不会被hang住。

3. 人工帮助

如果在使用中遇到棘手的问题，请加入我们官方用户支持群来找我们

- 钉钉群号 11747055