《OpenAI 代码自动评审组件》- 第4节-学习记录

来自： 码农会锁

爱海贼的无处不在

2024年07月30日 12:14

学习系列文章目录：

1、《OpenAI 代码自动评审组件》- 第1节-学习记录：https://articles.zsxq.com/id_bpapuj05ur03.html

2、《OpenAI 代码自动评审组件》- 第2节-学习记录：https://articles.zsxq.com/id_uokpg7mvt1uc.html

2、《OpenAI 代码自动评审组件》- 第3节-学习记录：https://articles.zsxq.com/id_bi51vtdhpp36.html

一、概述

本节开始作者针对这次的内容分别讲解了智普AI对接过程中的API文档说明、如何基于Token模式的对接访问、如何使用Java原生对象编写HTTP请求发送、发送消息体的结构化定义、响应对象的结构定义，最后通过代码提交代码测试。

二、正文

1、首先作者告诉了我们可以去智普AI网站上注册用户，这个我已经提前注册好了：

2、注册后我们会进入到主界面的控制台，然后我们可以查看资源包的情况，可以看到系统赠送了2500w的token信息，不过在仔细看，可以知道这个是一个月的有效期，所以当前项目的小伙伴一旦注册了，最好在1个月内使用哦，否则就浪费了这个资源了：

3、接下来作者带领大家开始写代码实现智普AI的对接，本节作者没有直接采用官方提供的Java SDK，而是自己基于其提供的Java API与原生的Java网络请求API实现了一个非常轻量级的组件对接，这也是作者在本次项目中提倡的，当前组件定位是一个轻量级的SDK组件，引入过多的SDK和依赖反而会显得太重了。这一点值得我们学习，如果未来我们是在server端做代码评审的的功能，可以用上一些SDK，来节省开发时间效率。

记得在上一节我在使用智普sdk实现了基本的代码评审后，稍不注意在Github上就会出现缺失依赖的问题。

本节作者分享了如何基于原始的Java API发送网络请求，那么对于相关秘钥信息，可以借助System的getEnv方法或者System.getProperty方法来处理：

```java
public class JavaCodeReviewHttpUtil {

    public static String codeReview(String diffCode) throws Exception {
        String zhipuKey = System.getProperty("ZHIPU_KEY");
        if (StringUtils.isEmpty(zhipuKey)) {
            return zhipuKey;
        }
        String apiKeySecret = zhipuKey;
        String token = BearerTokenUtils.getToken(apiKeySecret);
        URL url = new URL("https://open.bigmodel.cn/api/paas/v4/chat/completions");
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        connection.setRequestMethod("POST");
        connection.setRequestProperty("Authorization", "Bearer " + token);
        connection.setRequestProperty("Content-Type", "application/json");
        connection.setRequestProperty("User-Agent", "Mozilla/4.0 (compatible; MSIE 5.0; Windows NT; DigExt)");
        connection.setDoOutput(true);
        ChatCompletionRequest chatCompletionRequest = new ChatCompletionRequest();
        chatCompletionRequest.setModel(Model.GLM_4_FLASH.getCode());
        chatCompletionRequest.setMessages(new ArrayList<ChatCompletionRequest.Prompt>() {
            private static final long serialVersionUID = -7988151926241837899L; {
                add(new ChatCompletionRequest.Prompt("user",
                        "你是一个高级编程架构师，精通各类场景方案、架构设计和编程语言请，请您根据git diff记录，对代码做出评审。代码如下:"));
                add(new ChatCompletionRequest.Prompt("user", diffCode));
            }});
        try (OutputStream os = connection.getOutputStream()) {
            byte[] input = JSON.toJSONString(chatCompletionRequest).getBytes(StandardCharsets.UTF_8);
            os.write(input);
        }
        int responseCode = connection.getResponseCode();
        System.out.println(responseCode);
        BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream()));
        String inputLine;
        StringBuilder content = new StringBuilder();
        while ((inputLine = in.readLine()) != null) {
            content.append(inputLine);
        }
        in.close();
        connection.disconnect();
        System.out.println("评审结果: " + content.toString());
        ChatCompletionSyncResponse response = JSON.parseObject(content.toString(), ChatCompletionSyncResponse.class);
        return response.getChoices().get(0).getMessage().getContent();
    }
}
```


4、然后作者对于对接的方式，采用了Token方式，而没有直接采用令牌的方式，我想这个应该是从安全角度来考虑吧。官方支持的鉴权方式如下，我想大多数人会选择简单的使用API Key的方式来处理：

- 在调用模型接口时，支持两种鉴权方式：
- 传 API Key 进行认证
- 传鉴权 token 进行认证

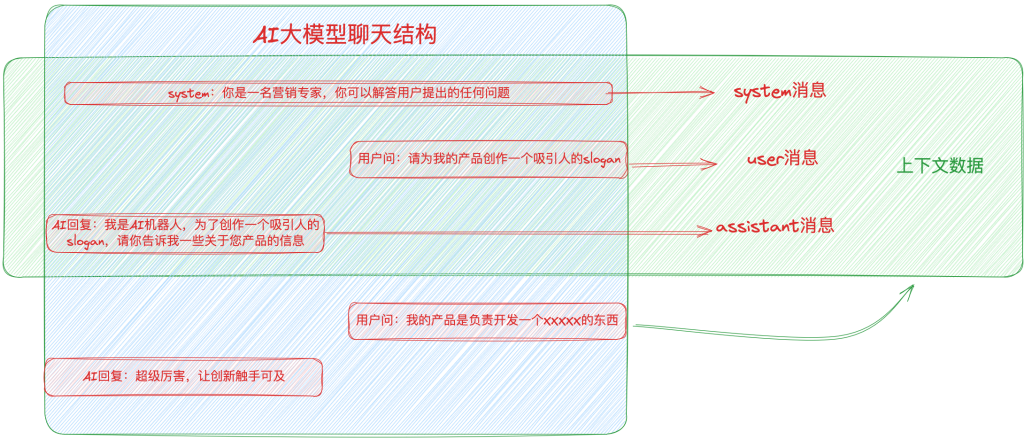
代码如下：

```java
public class BearerTokenUtils {


    // 过期时间；默认30分钟
    private static final long expireMillis = 30 * 60 * 1000L;

    // 缓存服务
    public static Cache<String, String> cache = CacheBuilder.newBuilder()
            .expireAfterWrite(expireMillis - (60 * 1000L), TimeUnit.MILLISECONDS)
            .build();

    public static String getToken(String apiKeySecret) {
        String[] split = apiKeySecret.split("\\.");
        return getToken(split[0], split[1]);          解析秘钥格式：AAAA.BBBBB
    }

    /**
     * 对 ApiKey 进行签名
     *
     * @param apiKey    登录创建 ApiKey <a href="https://open.bigmodel.cn/usercenter/apikeys">apikeys</a>
     * @param apiSecret apiKey的后半部分 828902ec516c45307619708d3e780ae1.w5eKiLvhnLP8MtIf 取 w5eKiLvhnLP8MtIf 使用
     * @return Token
     */
    public static String getToken(String apiKey, String apiSecret) {
        // 缓存Token
        String token = cache.getIfPresent(apiKey);
        if (null != token) return token;
        // 创建Token
        Algorithm algorithm = Algorithm.HMAC256(apiSecret.getBytes(StandardCharsets.UTF_8));
        Map<String, Object> payload = new HashMap<>();
        payload.put("api_key", apiKey);
        payload.put("exp", System.currentTimeMillis() + expireMillis);
        payload.put("timestamp", Calendar.getInstance().getTimeInMillis());
        Map<String, Object> headerClaims = new HashMap<>();
        headerClaims.put("alg", "HS256");
        headerClaims.put("sign_type", "SIGN");                    JWT封装为Token
        token = JWT.create().withPayload(payload).withHeader(headerClaims).sign(algorithm);
        cache.put(apiKey, token);
        return token;
    }

}
```

5、然后接下来作者定义了公共的请求参数和返回的响应结果，这里的这个接口非常通用和标准的AI大模型对接的基本通用的结构，解析到响应结果后，提交代码到Github仓库就实现了。这里面我们需要关注的是AI大模型的基本的请求结构，需要告诉AI大模型，当前用的模型是哪个，聊天会话信息时什么，提示词是什么，这块也基本上目前业界公共的了，作者把多余的东西都去掉了，为我们留下了核心的请求参数，这个是非常不错的，非常简单：

```java
public class ChatCompletionRequest {

    private String model = Model.GLM_4_FLASH.getCode();
    private List<Prompt> messages;

    public static class Prompt {
        private String role;
        private String content;

        public Prompt() {
        }

        public Prompt(String role, String content) {
            this.role = role;
            this.content = content;
        }

        public String getRole() {
            return role;
        }

        public void setRole(String role) {
            this.role = role;
        }

        public String getContent() {
            return content;
        }

        public void setContent(String content) {
            this.content = content;
        }

    }

    public String getModel() {
        return model;
    }
```

那么在这里我们需要关注这个提示词的数据结构的基础知识，提示词是AI大模型的第一阶的技术，也是每个AI从业者需要关注的内容，在通往AGI之路的飞书云文档中有过详细介绍，也是需要每个开发人员阅读的。那么从程序角度来说，我们需要理解整个AI的对话模型，可以用一张图来表示：

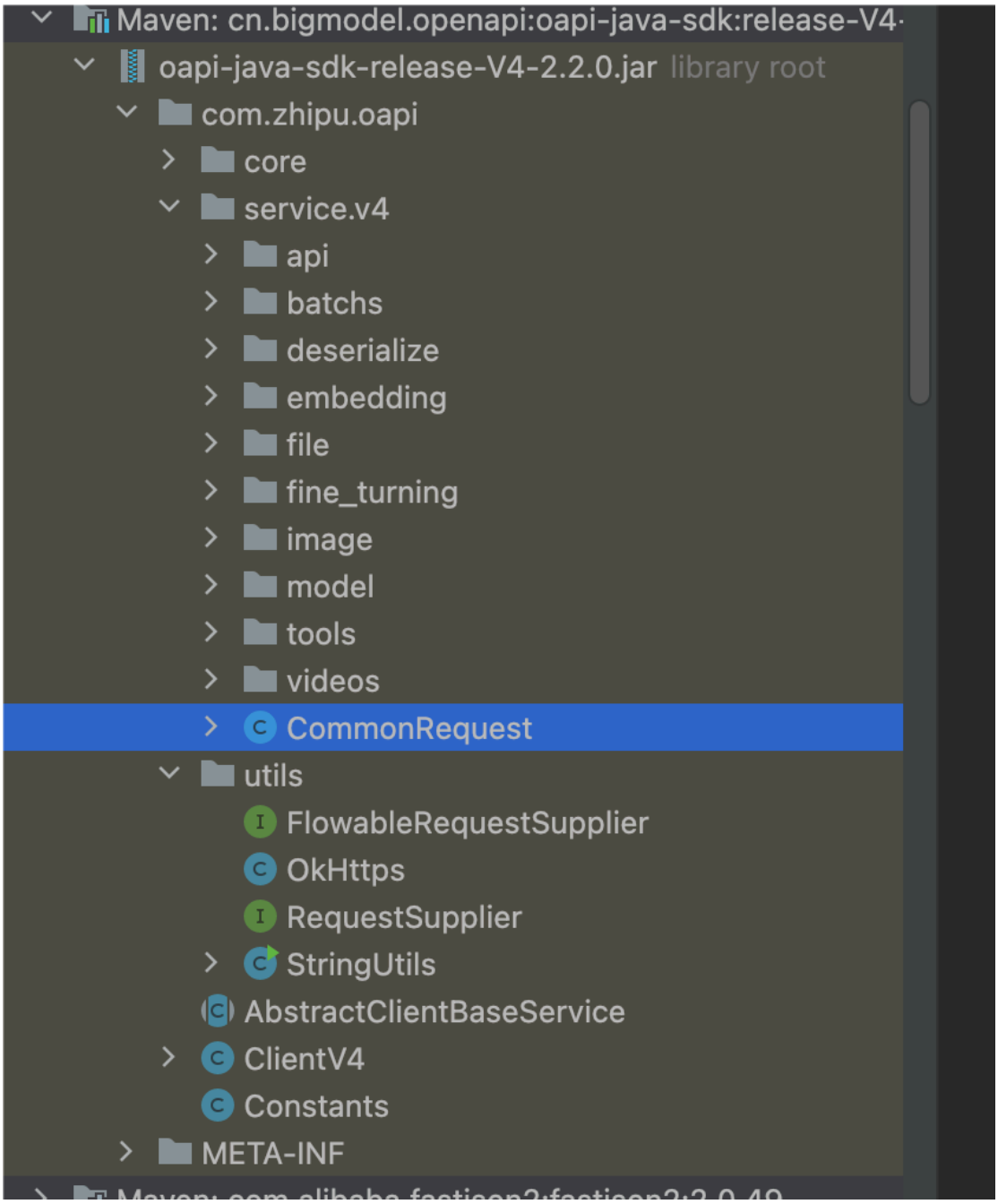


6、当我们熟悉了这个聊天结构后，后续我们优化提示词也是有一定的帮助的，从程序上开发角度，本质就是在构造聊天消息数组。有了这个结构后，作者也封装了返回结果，返回结果也是保留了核心的参数，对于AI大模型解析来说，基本上都是在解析choices[0].message.content这个内容，如果玩过JSONPath技术的小伙伴，就知道，未来可以用jsonpath来解析返回结果，这样就可以用通用的代码配置，来处理不同模型的我们的关注的返回结果。

当我们改完后，就可以提交代码到Github Actions上了。

这样，在作者分享下，我们的第三个AI代码评审迭代就完成了，第3个AI代码评审的架构图设计如下：

7、在学习完本节内容后，我们可以学习下智普AI的Java SDK设计思想，将里面设计好的地方拿来自己使用。比如说自己下载源码构建并学习下，结构内容也只值得学习，假设未来有一天需要自己设计个Java SDK，那么自己也是有能力设计了，程序员的抽象能力和设计能力的成长，往往来源这种日常小的组件的积累，在智普这个SDK中，我们就可以看到和官方文档匹配的抽象模型设计，也是完全可以把核心的内容放到项目中：



三、总结

本文作者分享了如何使用Java网络API给大模型发送请求，也演示了使用CURL发送网络请求，无论用什么，本质对接大模型就是给他发送一个请求，然后结合业务发送参数而已。虽然东西看着没啥，但是如果自己能够花时间深入学习，也是能学习到的知识点是非常多的。而不仅仅停留在CRUD的层面，背后的业务知识和技术也是需要自己处理的。