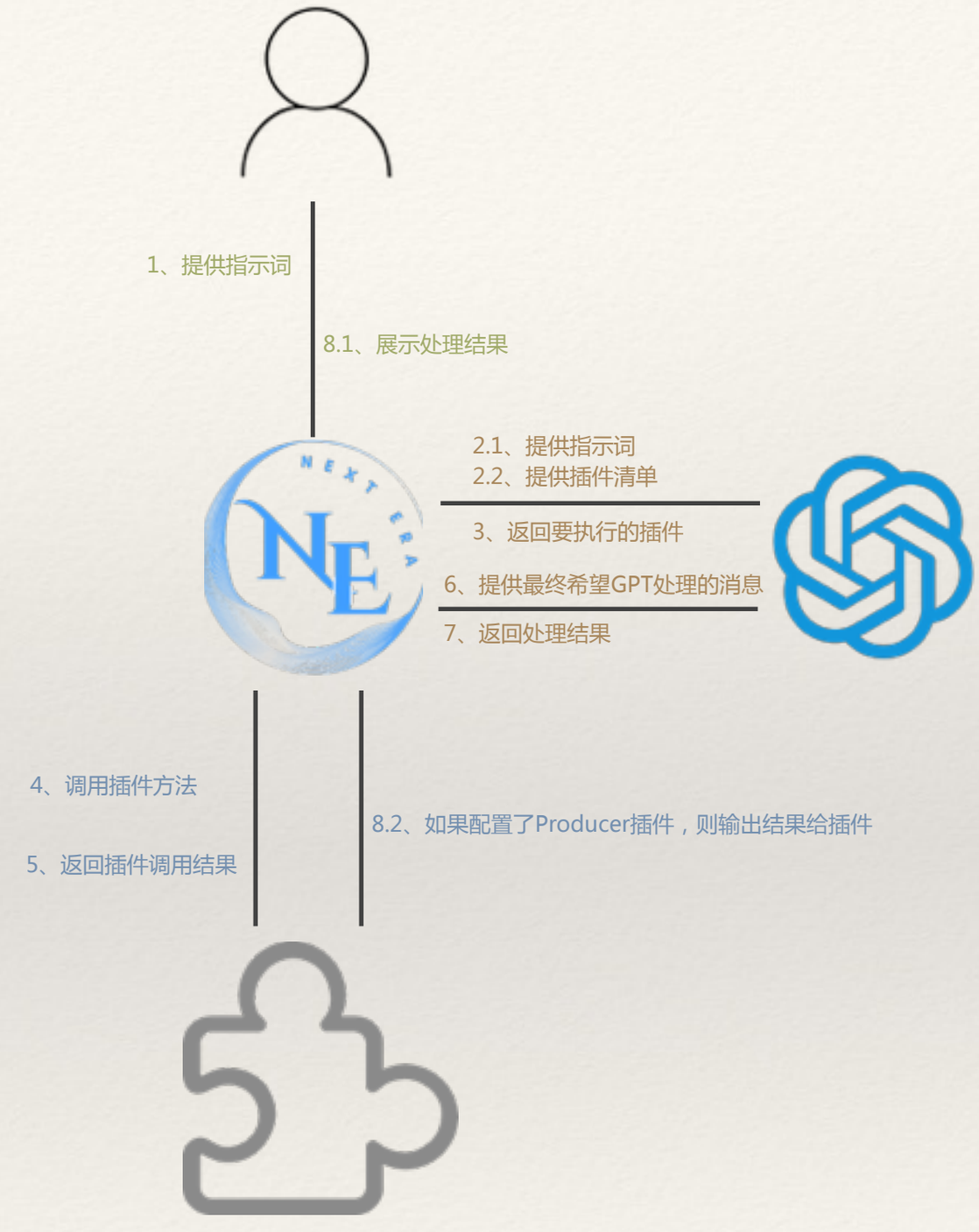
NextERA插件原理

NextERA的插件是基于OpenAI的Function Call机制实现的，按照官方要求会存在二次请求：第一次请求是将输入和所有Consumer类型的插件提交给官方平台，由它来判断是否触发FunctionCall，如果触发的话由官方平台解析输入的内容，返回Solt的内容，NextERA根据此调用插件系统，而第二次请求则是在NextERA调用Consumer类型插件得到结果之后，组装请求消息再次调用官方接口，官方第二次收到请求后，会根据预设的提示词来对让插件返回的内容进行处理。为了达到更好的效果，建议通过训练里的预设指令，告诉OpenAI应该如何处理消息。



插件开发的步骤

第一步，实现插件系统，并部署到公网能访问的服务器上

使用您所熟悉的任何一种编程语言，编写您想实现的任何插件功能，比如说Office文档的解析、内部知识图谱的检索、第三方聚合API接口的调用等等。NextERA对所暴露的插件能力要求就二个：返回指定格式的响应数据，响应数据里包含安全凭证。安全凭证是通过AES算法生成的，具体生成逻辑可以参看开源源代码里的lib包。

如果您也使用Java，可以移步Gitee，下载NextERA开源的插件，地址是：https://gitee.com/nextera/gpt-plugins。

响应数据的格式

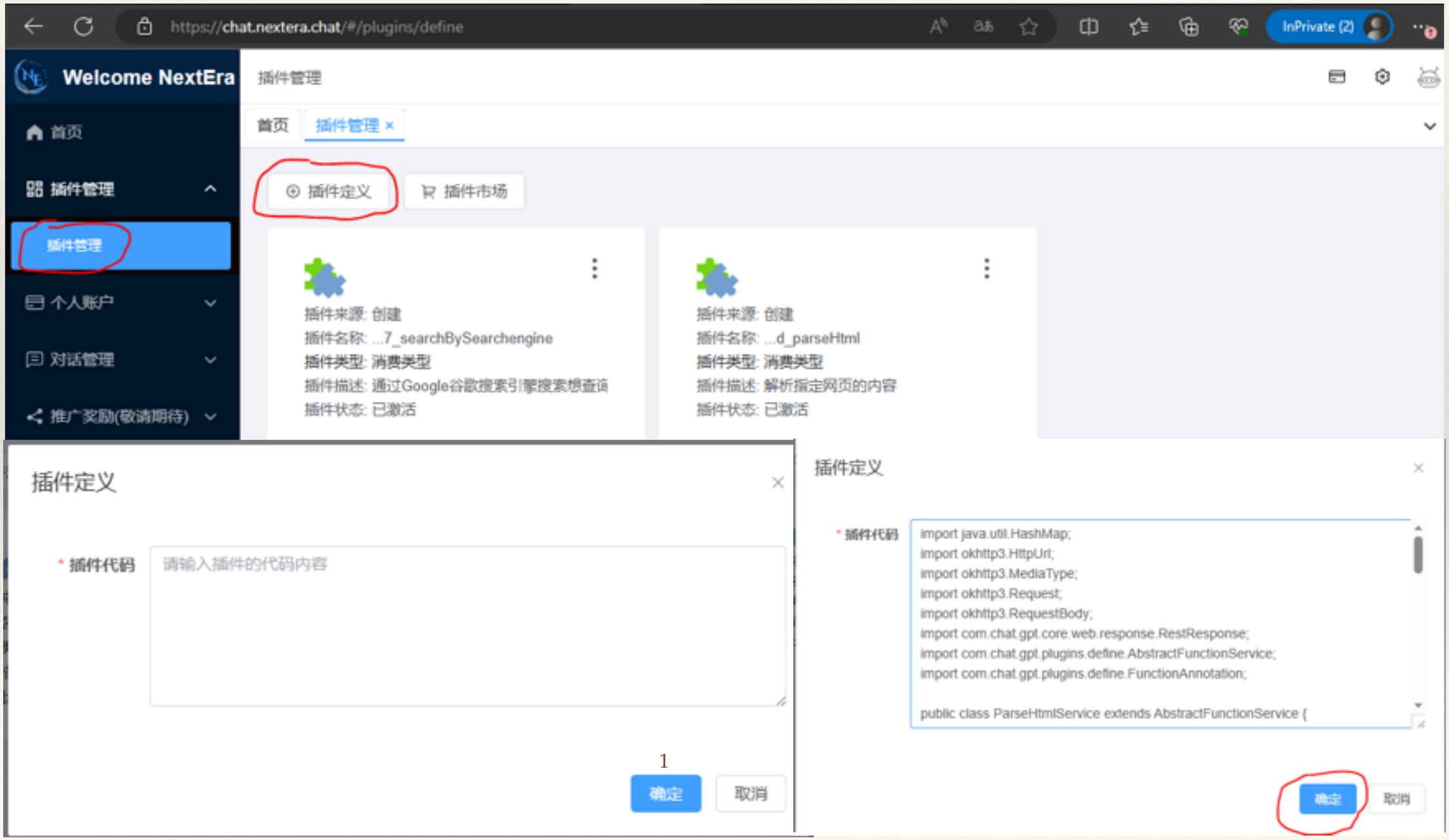

```
{
  "code": 20000,
  "message": "请求处理成功",
  "exceptionCode": -1,
  "exceptionTrace": "",
  "timestamp": 20230819130101695,
  "result": "{\"title\":\"六大主流开源协议总结 - touryung - 博客园\",\"content\":\"其中 Apache 许可证需要在每一个修改过的文件都放置版权声明，因此它要相对严格一点。\\nMIT 许可证是最宽松的，它允许衍生出来的软件使用它的名字促销。\\n举个例子：electron 和 vue 都是以 MIT 协议开源的，因此 electron-vue 就相当于是基于它们的衍生软件，因此就可以使用 electron 和 vue 来进行促销（假设商用的情况下）。\\n民间流传着关于 MIT 协议的一个很接地气的描述：“代码你随便用，但是出了事别找我”。\\n左边这三个许可证在修改代码之后必须开源，但又有一些小区别：\\n其余两种不限制开源协议，但是 Mozilla 许可证需要对代码修改之处提供修改说明文档，而 LGPL 许可证则不需要。\\n因此，六大许可证的宽松程度由严到宽分别是：GPL > Mozilla > LGPL > Apache > BSD > MIT。\\n\"}",
  "credential": "19909342-416B518DB0E6C2647C8E675D7EBEF34194219CBD40A1848C990150A72AF58553"
}
```


需注意，result属性必须是字符串类型。如果插件处理出现了异常，code返回一个非20000的结果，并且将异常内容记录在message里。

插件开发的步骤

第二步，在NextERA登记插件执行的代码

在NextERA登记插件主要有以下三个步骤：录入、测试、激活。先登录：https://chat.nextera.chat/



插件开发的步骤

录入时所需的插件执行代码，需用Java编写，具体样例可参看开源项目的README.RD

```
import java.util.HashMap;
import okhttp3.HttpUrl;
import okhttp3.MediaType;
import okhttp3.Request;
import okhttp3.RequestBody;
import com.chat.gpt.core.web.response.RestResponse;
import com.chat.gpt.plugins.define.AbstractFunctionService;
import com.chat.gpt.plugins.define.FunctionAnnotation;
public class ParseHtmlService extends AbstractFunctionService {
    private static final String pluginsUrl = "http://plugins.nextera.chat/api/plugins/parsehtml";
    @FunctionAnnotation(name = "parseHtml", type = "consumer", describe = "解析指定网页的内容")
    public String parseHtml(@FunctionAnnotation(name = "url", describe = "网址", required = true) String url) throws Exception {
        HashMap<String, String> params = new HashMap<>();
        params.put("url", url);
        params.put("encoding", "utf-8");
        try {
            HttpUrl httpUrl = getHttpUrl(pluginsUrl);
            RequestBody jsonBody = RequestBody.create(MediaType.parse("application/json"), beanToJson(params));
            Request httpRequest = new Request.Builder().url(httpUrl.newBuilder().build()).post(jsonBody).build();
            RestResponse<String> response = executeHttp(httpUrl, httpRequest);
            if (response.getCode() == 20000) {
                return response.getResult();
            } else {
                throw new Exception(response.getMessage());
            }
        } catch (Exception ex) {
            throw ex;
        }
    }
    @Override
    public Object[] testData() {
        return new Object[]{"https://hanyu.baidu.com/shici/detail?pid=247de09646ac9b265ae504216168e3f0&from=kg0"};
    }
    @Override
    public String revealAllTheDetails() {
        return "不能正确解析网址";
    }
}
```

1、 pluginsUrl这个常量是要调用插件的实际完整地址，必须公网能访问；

2、 FunctionAnnotation这个注解，需要在方法和变量名分别注解，用来生成函数和函数的参数；

2.1 name, 用来标记函数名或参数名，需注意函数名和定义的方法名必须一致;

2.2 type，可选consumer和producer，定义在函数上。以插件服务作为主体，consumer类型的插件表示会消费服务返回的信息，提供给OpenAI，让它对信息做处理；而producer类型的插件则是将OpenAI返回的结果提供给插件服务，此时它是信息提供方;

2.3 describe, 函数或参数的描述，务必描述正确，以方便OpenAI判断要不要触发插件，应该调用哪个插件；

2.4 required，标记参数是否必须，定义在参数上；

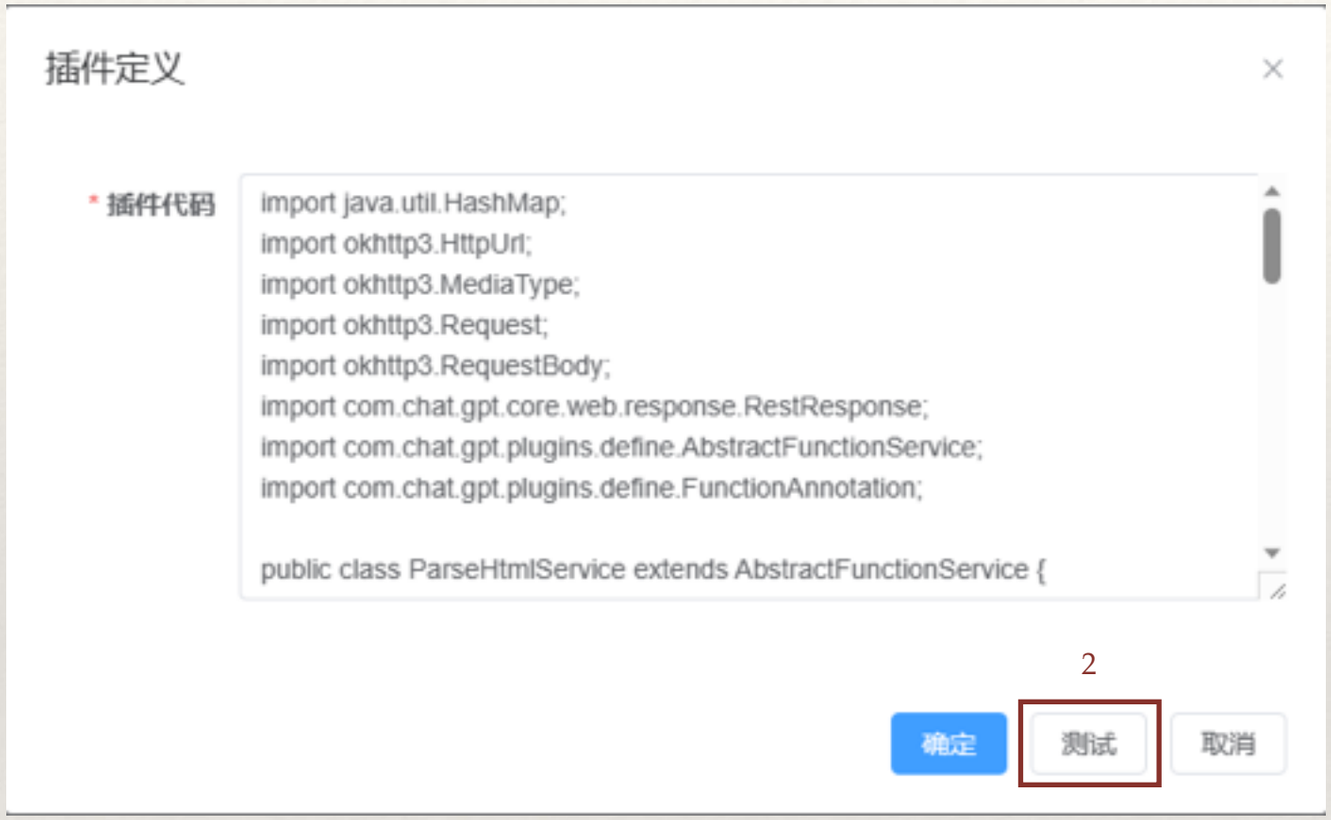
3、 params，是调用插件的参数，具体参数内容由OpenAI返回，参数的格式则由插件服务决定;

4、 对插件函数定义来讲，如果HTTP Method是POST，并且Content-Type是application/json，后续复制粘贴即可，如果是其它方法，用OKHTTP3发起请求即可。

5、 testData，必须实现，用于测试插件所需的参数，和插件函数里所定义的参数保持一致(包括次序、类型)；

插件开发的步骤

在NextERA登记插件主要有以下三个步骤：录入、测试、激活。



测试通过之后，会在测试结果显示插件执行后返回的内容，实际执行对话时这些内容将发送给OpenAI。

最后点击激活按钮，插件一经激活，就可以在模型定义里将模型和插件做关联。

插件开发的所有工作都已完成。如果想将插件放入市场让NextERA的所有用户都可以使用的话，可以使用发布功能发布插件。

为模型配置插件

对话模型关联插件，只需一个步骤：关联

当然，如果您希望OpenAI返回更符合您预期结果的话，可以通过训练功能先将预置指令登记进去。

模型插件里会显示所有已激活的插件以及从插件市场里成功订阅的插件，默认显示的是Consumer类型的插件，如果想切换到Producer类型的插件点击下方的生产类型插件进行切换即可。

这里有个不是很友好的问题：插件列表空间有限，只显示2条，需要拖到表格右边的滚动条才能显示其它插件，后续再想办法优化吧。

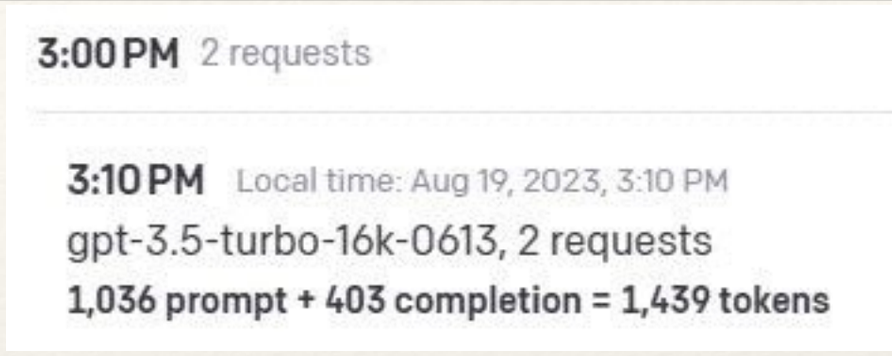
为模型配置插件

登记预置指令

使用训练功能时，通常只需要在训练指令里输入预置指示词即可。但如果OpenAI的回答不是特别符合你的要求，可以通过模拟问答，把您所问的内容和您希望得到的回答，或不希望得到的回答先模拟一下，OpenAI会根据您的指令和模拟问答，返回更符合您期望的结果。

指令触发不需要输入，这个是在定义Mr. Ranedeer时的一个特殊场景。

样例以及Token情况



提示词差10Token，完成差9Token；（费用0.0483元）
如果不用插件，提示词完全一致，完成差7Token；

标识	账单时间	提示词Token数	回答Token数	Token数小计	接口费用	平台费用	费用小计
1c1b5f57e2184...	2023-08-19 23:47:01	845	463	1,308	339	102	441
1c1b5f57e2184...	2023-08-19 23:46:01	322	47	369	90	28	118
1c1b57516535c...	2023-08-19 23:13:01	670	363	1,033	267	81	348
1c1b57516535c...	2023-08-19 23:12:02	376	49	425	103	32	135

提示词差11Token，完成差10Token；（费用0.0559元）
跟踪一段时间看看结果，尽量控制提示词差3以内，完成差8左右；
试用账号：guest@nextera.chat已配置插件，可体验。密码：(asdqwe123)