# 概念

HttpClient 是Apache Jakarta Common 下的子项目，可以用来提供高效的、最新的、功能丰富的支持 HTTP 协议的客户端编程工具包，并且它支持 HTTP 协议最新的版本和建议。

# 功能

（1）实现了所有HTTP的方法（GET POST PUT DELETE等），可以调用远程接口。

（2）支持自动转向

（3）支持HTTPS协议

（4）支持代理服务器等

# 导入的依赖

<dependency>

<groupId>org.apache.httpcomponents</groupId>

<artifactId>httpclient</artifactId>

<version>4.5.1</version>

</dependency>

# 使用方法

## 基本操作步骤

使用HttpClient发送请求、接收响应很简单，一般需要如下几步：

### 创建HttpClient对象

### 创建请求方法的实例，并指定请求的URL

如果需要发送GET请求，创建HttpGet对象，如果需要发送POST请求，创建HttpPost对象。

### 设置请求参数

如果需要发送请求参数，可以调用HttpGet、HttpPost共同的setParams(HttpParams params)方法来添加请求参数，对于HttpPost对象而言，也可调用setEntity(HttpEntity entity)方法来设置请求参数。

### 发送请求

调用HttpClient对象的execute(HttpUriRequest request)发送请求，该方法返回一个HttpResponse对象。

### 获取响应内容

调用HttpResponse对象的getAllHeaders()、getHeaders(String name)等方法可获取服务器的响应头，调用HttpResponse的getEntity()方法可获取HttpEntity对象，该对象包装了服务器的响应内容，程序可通过该对象获取服务器的响应内容。

### 释放连接

释放连接。无论执行方法是否成功，都必须释放连接。

## 代码示例



# GET请求

**public** **class** GetWS {

**public** **static** **void** **main**(String[] args) {

*get*();

}

**public** **static** **void** **get**() {

CloseableHttpClient **httpCilent** = HttpClients.*createDefault*();

HttpGet **httpGet** = **new** HttpGet("https://www.cnblogs.com/c9999/p/6636415.html");

**try** {

CloseableHttpResponse **response** = httpCilent.execute(httpGet);

System.***out***.println(response.getStatusLine());

HttpEntity **entity** = response.getEntity();

String **result** = EntityUtils.*toString*(entity);

System.***out***.println(result);

} **catch** (IOException **e**) {

e.printStackTrace();

}**finally** {

**try** {

httpCilent.close();//释放资源

} **catch** (IOException **e**) {

e.printStackTrace();

}

}

}

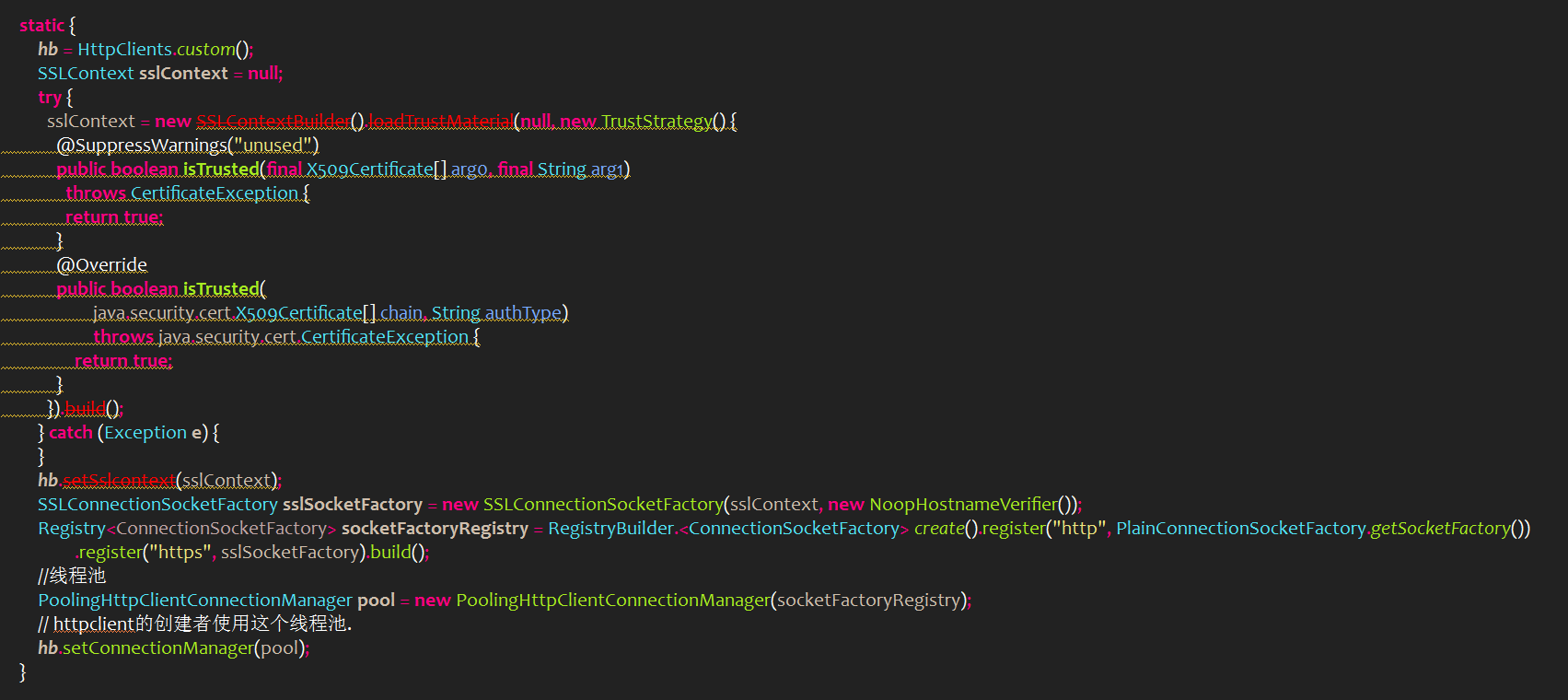
}

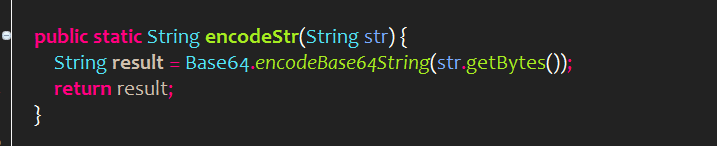
# 使用POST方式读取Basic Auth的接口

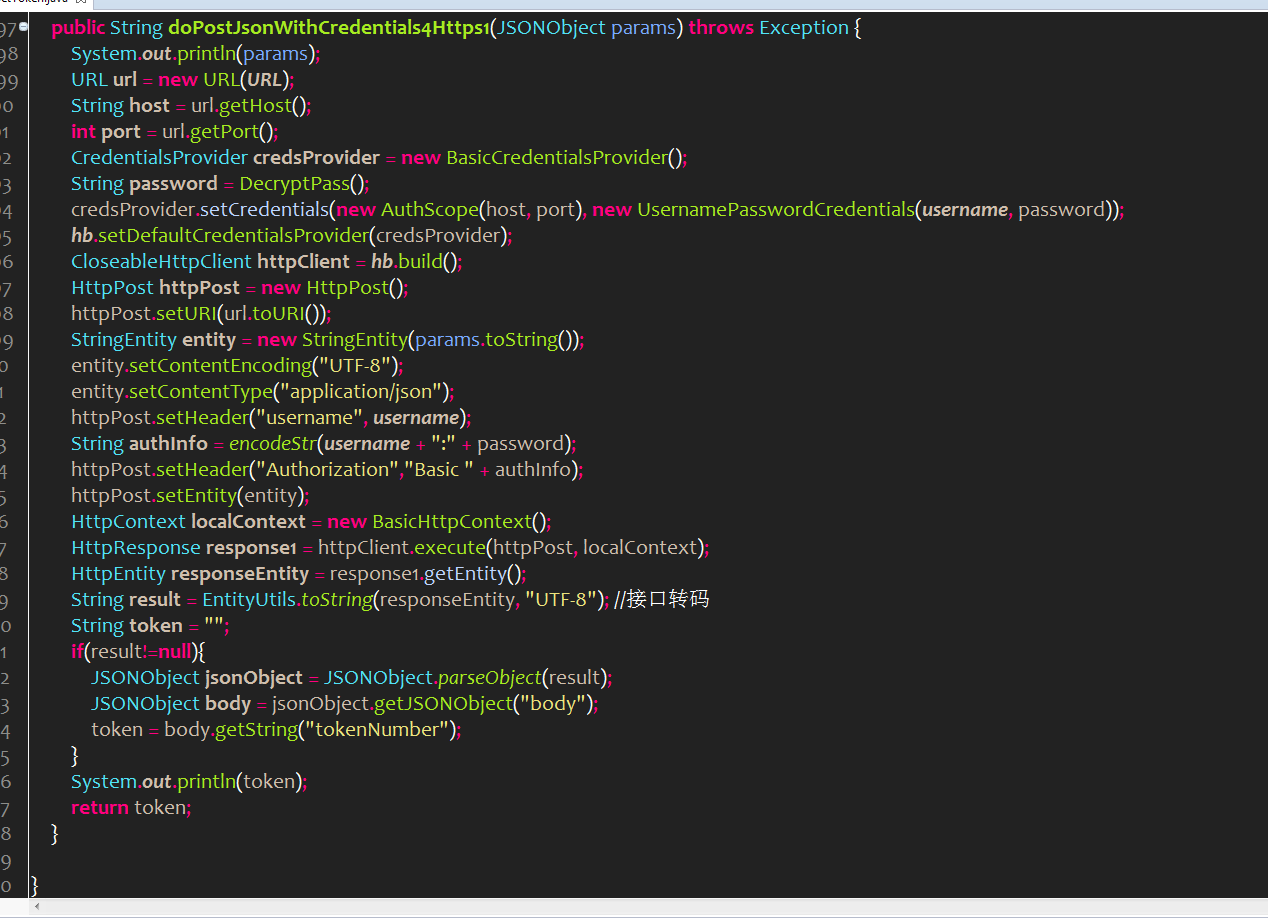
## 示例代码

我创建了GetToken.java类













# 使用POST方式提交数据（httpClient3）

httpclient使用了单独的一个HttpMethod子类来处理文件的上传，这个类就是MultipartPostMethod，该类已经封装了文件上传的细节，我们要做的仅仅是告诉它我们要上传文件的全路径即可，下面这里将给出关于两种模拟上传方式的代码

## （1）模拟上传url文件（该方式也适合做普通的post请求）

/\*\*

\* 上传url文件到指定URL

\* @param fileUrl 上传图片url

\* @param postUrl 上传路径及参数,注意有些中文参数需要使用预先编码 eg : URLEncoder.encode(appName, "UTF-8")

\* @return

\* @throws IOException

\*/

public static String doUploadFile(String postUrl) throws IOException {

if(StringUtils.isEmpty(postUrl))

return null;

String response = "";

PostMethod postMethod = new PostMethod(postUrl);

try {

HttpClient client = new HttpClient();

client.getHttpConnectionManager().getParams()

.setConnectionTimeout(50000);// 设置连接时间

int status = client.executeMethod(postMethod);

if (status == HttpStatus.SC\_OK) {

InputStream inputStream = postMethod.getResponseBodyAsStream();

BufferedReader br = new BufferedReader(new InputStreamReader(

inputStream));

StringBuffer stringBuffer = new StringBuffer();

String str = "";

while ((str = br.readLine()) != null) {

stringBuffer.append(str);

}

response = stringBuffer.toString();

} else {

response = "fail";

}

} catch (Exception e) {

e.printStackTrace();

} finally {

// 释放连接

postMethod.releaseConnection();

}

return response;

}

## （2）模拟文件上传到指定位置

/\*\*

\* 上传文件到指定URL

\* @param file

\* @param url

\* @return

\* @throws IOException

\*/

public static String doUploadFile(File file, String url) throws IOException {

String response = "";

if (!file.exists()) {

return "file not exists";

}

PostMethod postMethod = new PostMethod(url);

try {

//----------------------------------------------

// FilePart：用来上传文件的类,file即要上传的文件

FilePart fp = new FilePart("file", file);

Part[] parts = { fp };

// 对于MIME类型的请求，httpclient建议全用MulitPartRequestEntity进行包装

MultipartRequestEntity mre = new MultipartRequestEntity(parts,

postMethod.getParams());

postMethod.setRequestEntity(mre);

//---------------------------------------------

HttpClient client = new HttpClient();

client.getHttpConnectionManager().getParams()

.setConnectionTimeout(50000);// 由于要上传的文件可能比较大 , 因此在此设置最大的连接超时时间

int status = client.executeMethod(postMethod);

if (status == HttpStatus.SC\_OK) {

InputStream inputStream = postMethod.getResponseBodyAsStream();

BufferedReader br = new BufferedReader(new InputStreamReader(

inputStream));

StringBuffer stringBuffer = new StringBuffer();

String str = "";

while ((str = br.readLine()) != null) {

stringBuffer.append(str);

}

response = stringBuffer.toString();

} else {

response = "fail";

}

} catch (Exception e) {

e.printStackTrace();

} finally {

// 释放连接

postMethod.releaseConnection();

}

return response;

}

# 常用的接口

## HttpClient

## CredentialsProvider

# 常用的类

## SSLContext

此类的实例表示安全套接字协议实现，它充当安全套接字工厂或SSLEngines的工厂。 该类使用一组可选的密钥和信任管理器以及安全随机字节源进行初始化。

## HttpClientBuilder

## HttpClients

CloseableHttpClient实例的工厂方法。

### 方法

#### public static CloseableHttpClient createDefault()

使用默认配置创建CloseableHttpClient实例。

## HttpGet

## HttpPost

POST方法用于请求源服务器接受请求中包含的实体作为Request-URI中Request-URI标识的资源的新下级。 POST旨在允许统一的方法来涵盖以下功能：

（1）现有资源的注解

（2）将消息发布到公告板，新闻组，邮件列表或类似的文章组

（3）向数据处理过程提供数据块，例如提交表单的结果

（4）通过追加操作扩展数据库

### 构造函数

#### public HttpPost(final String uri)

## BasicCredentialsProvider

## RequestBuilder

这个类是HttpUriRequest实例的创建者

## StringEntity

一个自包含的、可重复的实体，从字符串中获取其内容。

## CloseableHttpClient

## EntityUtils

## StringEntity

# 遇到的问题

## 1、SSL异常javax.net.ssl.SSLHandshakeException: Received fatal alert: handshake\_failure

解决方法：参考如下网址：

https://blog.csdn.net/fsafs168/article/details/82020959

https://blog.csdn.net/sun5769675/article/details/81322324