### **HttpURLConnection介绍：**

**http://www.cnblogs.com/whoislcj/p/5520384.html**

HttpURLConnection是一种多用途、轻量极的HTTP客户端，使用它来进行HTTP操作可以适用于大多数的应用程序。对于之前为何一直使用HttpClient而不使用HttpURLConnection也是有原因的。具体分析如下

* HttpClient是apache的开源框架，封装了访问http的请求头，参数，内容体，响应等等，使用起来比较方便，而HttpURLConnection是java的标准类，什么都没封装，用起来太原始，不方便，比如重访问的自定义，以及一些高级功能等。
* 从稳定性方面来说的话，HttpClient很稳定，功能强，BUG少，容易控制细节，而之前的HttpURLConnection一直存在着版本兼容的问题，不过在后续的版本中已经相继修复掉了。

从上面可以看出之前一直使用HttClient是由于HttpURLConnection不稳定导致，那么现在谷歌虽然修复了HttpURLConnection之前存在的一些问题之后，相比HttpClient有什么优势呢？为何要废除HttpClient呢？

* HttpUrlConnection是Android SDK的标准实现，而HttpClient是apache的开源实现；
* HttpUrlConnection直接支持GZIP压缩；HttpClient也支持，但要自己写代码处理；
* HttpUrlConnection直接支持系统级连接池，即打开的连接不会直接关闭，在一段时间内所有程序可共用；HttpClient当然也能做到，但毕竟不如官方直接系统底层支持好；
* HttpUrlConnection直接在系统层面做了缓存策略处理，加快重复请求的速度。

### **HttpURLConnection使用：**

#### **Get请求实现：**

private void requestGet(HashMap<String, String> paramsMap) {

try {

String baseUrl = "https://xxx.com/getUsers?";

StringBuilder tempParams = new StringBuilder();

int pos = 0;

for (String key : paramsMap.keySet()) {

if (pos > 0) {

tempParams.append("&");

}

tempParams.append(String.format("%s=%s", key, URLEncoder.encode(paramsMap.get(key),"utf-8")));

pos++;

}

String requestUrl = baseUrl + tempParams.toString();

// 新建一个URL对象

URL url = new URL(requestUrl);

// 打开一个HttpURLConnection连接

HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();

// 设置连接主机超时时间

urlConn.setConnectTimeout(5 \* 1000);

//设置从主机读取数据超时

urlConn.setReadTimeout(5 \* 1000);

// 设置是否使用缓存 默认是true

urlConn.setUseCaches(true);

// 设置为Post请求

urlConn.setRequestMethod("GET");

//urlConn设置请求头信息

//设置请求中的媒体类型信息。

urlConn.setRequestProperty("Content-Type", "application/json");

//设置客户端与服务连接类型

urlConn.addRequestProperty("Connection", "Keep-Alive");

// 开始连接 urlConn.connect();

// 判断请求是否成功

if (urlConn.getResponseCode() == 200) {

// 获取返回的数据

String result = streamToString(urlConn.getInputStream());

Log.e(TAG, "Get方式请求成功，result--->" + result);

} else {

Log.e(TAG, "Get方式请求失败");

}

// 关闭连接 urlConn.disconnect();

} catch (Exception e) {

Log.e(TAG, e.toString());

}

}

#### **POST请求实现：**

private void requestPost(HashMap<String, String> paramsMap) {

try {

String baseUrl = "https://xxx.com/getUsers";

//合成参数

StringBuilder tempParams = new StringBuilder();

int pos = 0;

for (String key : paramsMap.keySet()) {

if (pos > 0) {

tempParams.append("&");

}

tempParams.append(String.format("%s=%s", key, URLEncoder.encode(paramsMap.get(key),"utf-8")));

pos++;

}

String params =tempParams.toString();

// 请求的参数转换为byte数组

byte[] postData = params.getBytes();

// 新建一个URL对象

URL url = new URL(baseUrl);

// 打开一个HttpURLConnection连接

HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();

// 设置连接超时时间

urlConn.setConnectTimeout(5 \* 1000);

//设置从主机读取数据超时

urlConn.setReadTimeout(5 \* 1000);

// Post请求必须设置允许输出 默认false

urlConn.setDoOutput(true);

//设置请求允许输入 默认是true

urlConn.setDoInput(true);

// Post请求不能使用缓存

urlConn.setUseCaches(false);

// 设置为Post请求

urlConn.setRequestMethod("POST");

//设置本次连接是否自动处理重定向

urlConn.setInstanceFollowRedirects(true);

// 配置请求Content-Type

urlConn.setRequestProperty("Content-Type", "application/json");

// 开始连接 urlConn.connect();

// 发送请求参数

DataOutputStream dos = new DataOutputStream(urlConn.getOutputStream());

dos.write(postData);

dos.flush();

dos.close();

// 判断请求是否成功

if (urlConn.getResponseCode() == 200) {

// 获取返回的数据

String result = streamToString(urlConn.getInputStream());

Log.e(TAG, "Post方式请求成功，result--->" + result);

} else {

Log.e(TAG, "Post方式请求失败");

}

// 关闭连接 urlConn.disconnect();

} catch (Exception e) {

Log.e(TAG, e.toString());

}

}

#### **处理网络流：将输入流转换成字符串**

/\*\*

\* 将输入流转换成字符串

\*

\* @param is 从网络获取的输入流

\* @return

\*/

public String streamToString(InputStream is) {

try {

ByteArrayOutputStream baos = new ByteArrayOutputStream();

byte[] buffer = new byte[1024];

int len = 0;

while ((len = is.read(buffer)) != -1) {

baos.write(buffer, 0, len);

}

baos.close();

is.close();

byte[] byteArray = baos.toByteArray();

return new String(byteArray);

} catch (Exception e) {

Log.e(TAG, e.toString());

return null;

}

}

以上就是HttpConnection的get、post的简单实现，如何实现文件的下载和上传呢？

#### **文件下载：**

private void downloadFile(String fileUrl){

try {

// 新建一个URL对象

URL url = new URL(fileUrl);

// 打开一个HttpURLConnection连接

HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();

// 设置连接主机超时时间

urlConn.setConnectTimeout(5 \* 1000);

//设置从主机读取数据超时

urlConn.setReadTimeout(5 \* 1000);

// 设置是否使用缓存 默认是true

urlConn.setUseCaches(true);

// 设置为Post请求

urlConn.setRequestMethod("GET");

//urlConn设置请求头信息

//设置请求中的媒体类型信息。

urlConn.setRequestProperty("Content-Type", "application/json");

//设置客户端与服务连接类型

urlConn.addRequestProperty("Connection", "Keep-Alive");

// 开始连接 urlConn.connect();

// 判断请求是否成功

if (urlConn.getResponseCode() == 200) {

String filePath="";

File descFile = new File(filePath);

FileOutputStream fos = new FileOutputStream(descFile);;

byte[] buffer = new byte[1024];

int len;

InputStream inputStream = urlConn.getInputStream();

while ((len = inputStream.read(buffer)) != -1) {

// 写到本地

fos.write(buffer, 0, len);

}

} else {

Log.e(TAG, "文件下载失败");

}

// 关闭连接 urlConn.disconnect();

} catch (Exception e) {

Log.e(TAG, e.toString());

}

}

#### **文件上传：**

private void upLoadFile(String filePath, HashMap<String, String> paramsMap) {

try {

String baseUrl = "https://xxx.com/uploadFile";

File file = new File(filePath);

//新建url对象

URL url = new URL(baseUrl);

//通过HttpURLConnection对象,向网络地址发送请求

HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();

//设置该连接允许读取

urlConn.setDoOutput(true);

//设置该连接允许写入

urlConn.setDoInput(true);

//设置不能适用缓存

urlConn.setUseCaches(false);

//设置连接超时时间

urlConn.setConnectTimeout(5 \* 1000); //设置连接超时时间

//设置读取超时时间

urlConn.setReadTimeout(5 \* 1000); //读取超时

//设置连接方法post

urlConn.setRequestMethod("POST");

//设置维持长连接

urlConn.setRequestProperty("connection", "Keep-Alive");

//设置文件字符集

urlConn.setRequestProperty("Accept-Charset", "UTF-8");

//设置文件类型

urlConn.setRequestProperty("Content-Type", "multipart/form-data; boundary=" + "\*\*\*\*\*");

String name = file.getName();

DataOutputStream requestStream = new DataOutputStream(urlConn.getOutputStream());

requestStream.writeBytes("--" + "\*\*\*\*\*" + "\r\n");

//发送文件参数信息

StringBuilder tempParams = new StringBuilder();

tempParams.append("Content-Disposition: form-data; name=\"" + name + "\"; filename=\"" + name + "\"; ");

int pos = 0;

int size=paramsMap.size();

for (String key : paramsMap.keySet()) {

tempParams.append( String.format("%s=\"%s\"", key, paramsMap.get(key), "utf-8"));

if (pos < size-1) {

tempParams.append("; ");

}

pos++;

}

tempParams.append("\r\n");

tempParams.append("Content-Type: application/octet-stream\r\n");

tempParams.append("\r\n");

String params = tempParams.toString();

requestStream.writeBytes(params);

//发送文件数据

FileInputStream fileInput = new FileInputStream(file);

int bytesRead;

byte[] buffer = new byte[1024];

DataInputStream in = new DataInputStream(new FileInputStream(file));

while ((bytesRead = in.read(buffer)) != -1) {

requestStream.write(buffer, 0, bytesRead);

}

requestStream.writeBytes("\r\n");

requestStream.flush();

requestStream.writeBytes("--" + "\*\*\*\*\*" + "--" + "\r\n");

requestStream.flush();

fileInput.close();

int statusCode = urlConn.getResponseCode();

if (statusCode == 200) {

// 获取返回的数据

String result = streamToString(urlConn.getInputStream());

Log.e(TAG, "上传成功，result--->" + result);

} else {

Log.e(TAG, "上传失败");

}

} catch (IOException e) {

Log.e(TAG, e.toString());

}

}