**.概念**

      HTTP 协议可能是现在 Internet 上使用得最多、最重要的协议了，越来越多的 Java 应用程序需要直接通过 HTTP 协议来访问网络资源。在 JDK 的 java.net 包中已经提供了访问 HTTP 协议的基本功能：HttpURLConnection。但是对于大部分应用程序来说，JDK 库本身提供的功能还不够丰富和灵活。

      除此之外，在Android中，androidSDK中集成了Apache的HttpClient模块，用来提供高效的、最新的、功能丰富的支持 HTTP 协议工具包，并且它支持 HTTP 协议最新的版本和建议。使用HttpClient可以快速开发出功能强大的Http程序。

**2.区别**

HttpClient是个很不错的开源框架，封装了访问http的请求头，参数，内容体，响应等等，

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。

**3.案例**

**HttpURLConnection**

String urlAddress = "http://192.168.1.102:8080/AndroidServer/login.do";  
URL url;  
HttpURLConnection uRLConnection;  
publicUrlConnectionToServer(){  
  
}//向服务器发送get请求  
publicString doGet(String username,String password){  
String getUrl = urlAddress + "?username="+username+"&password="+password;  
try{  
url =newURL(getUrl);  
uRLConnection = (HttpURLConnection)url.openConnection();  
InputStream is = uRLConnection.getInputStream();  
BufferedReader br =newBufferedReader(newInputStreamReader(is));  
String response = "";  
String readLine =null;  
while((readLine =br.readLine()) !=null){  
//response = br.readLine();  
response = response + readLine;  
}  
is.close();  
br.close();  
uRLConnection.disconnect();  
returnresponse;  
}catch(MalformedURLException e) {  
e.printStackTrace();  
returnnull;  
}catch(IOException e) {  
e.printStackTrace();  
returnnull;  
}  
}  
 //向服务器发送post请求  
publicString doPost(String username,String password){  
try{  
url =newURL(urlAddress);  
uRLConnection = (HttpURLConnection)url.openConnection();  
uRLConnection.setDoInput(true);  
uRLConnection.setDoOutput(true);  
uRLConnection.setRequestMethod("POST");  
uRLConnection.setUseCaches(false);  
uRLConnection.setInstanceFollowRedirects(false);  
uRLConnection.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");  
uRLConnection.connect();  
  
DataOutputStream out =newDataOutputStream(uRLConnection.getOutputStream());  
String content = "username="+username+"&password="+password;  
out.writeBytes(content);  
out.flush();  
out.close();  
  
InputStream is = uRLConnection.getInputStream();  
BufferedReader br =newBufferedReader(newInputStreamReader(is));  
String response = "";  
String readLine =null;  
while((readLine =br.readLine()) !=null){  
//response = br.readLine();  
response = response + readLine;  
}  
is.close();  
br.close();  
uRLConnection.disconnect();  
returnresponse;  
}catch(MalformedURLException e) {  
e.printStackTrace();  
returnnull;  
}catch(IOException e) {  
e.printStackTrace();  
returnnull;  
}  
}

**HTTPClient**

String urlAddress = "http://192.168.1.102:8080/qualityserver/login.do";  
publicHttpClientServer(){  
  
}  
  
publicString doGet(String username,String password){  
String getUrl = urlAddress + "?username="+username+"&password="+password;  
HttpGet httpGet =newHttpGet(getUrl);  
HttpParams hp = httpGet.getParams();  
hp.getParameter("true");  
//hp.  
//httpGet.setp  
HttpClient hc =newDefaultHttpClient();  
try{  
HttpResponse ht = hc.execute(httpGet);  
if(ht.getStatusLine().getStatusCode() == HttpStatus.SC\_OK){  
HttpEntity he = ht.getEntity();  
InputStream is = he.getContent();  
BufferedReader br =newBufferedReader(newInputStreamReader(is));  
String response = "";  
String readLine =null;  
while((readLine =br.readLine()) !=null){  
//response = br.readLine();  
response = response + readLine;  
}  
is.close();  
br.close();  
  
//String str = EntityUtils.toString(he);  
System.out.println("========="+response);  
returnresponse;  
}else{  
return"error";  
}  
}catch(ClientProtocolException e) {  
//TODO Auto-generated catch block  
e.printStackTrace();  
return"exception";  
}catch(IOException e) {  
//TODO Auto-generated catch block  
e.printStackTrace();  
return"exception";  
}  
}  
  
publicString doPost(String username,String password){  
//String getUrl = urlAddress + "?username="+username+"&password="+password;  
HttpPost httpPost =newHttpPost(urlAddress);  
List params =newArrayList();  
NameValuePair pair1 =newBasicNameValuePair("username", username);  
NameValuePair pair2 =newBasicNameValuePair("password", password);  
params.add(pair1);  
params.add(pair2);  
  
HttpEntity he;  
try{  
he =newUrlEncodedFormEntity(params, "gbk");  
httpPost.setEntity(he);  
  
}catch(UnsupportedEncodingException e1) {  
//TODO Auto-generated catch block  
e1.printStackTrace();  
}  
  
HttpClient hc =newDefaultHttpClient();  
try{  
HttpResponse ht = hc.execute(httpPost);  
//连接成功  
if(ht.getStatusLine().getStatusCode() == HttpStatus.SC\_OK){  
HttpEntity het = ht.getEntity();  
InputStream is = het.getContent();  
BufferedReader br =newBufferedReader(newInputStreamReader(is));  
String response = "";  
String readLine =null;  
while((readLine =br.readLine()) !=null){  
//response = br.readLine();  
response = response + readLine;  
}  
is.close();  
br.close();  
  
//String str = EntityUtils.toString(he);  
System.out.println("=========&&"+response);  
returnresponse;  
}else{  
return"error";  
}  
}catch(ClientProtocolException e) {  
//TODO Auto-generated catch block  
e.printStackTrace();  
return"exception";  
}catch(IOException e) {  
//TODO Auto-generated catch block  
e.printStackTrace();  
return"exception";  
}  
}

**servlet端json转化：**

resp.setContentType("text/json");  
resp.setCharacterEncoding("UTF-8");  
toDo =newToDo();  
List<UserBean> list =newArrayList<UserBean>();  
list = toDo.queryUsers(mySession);  
String body;  
  
//设定JSON  
JSONArray array =newJSONArray();  
for(UserBean bean : list)  
{  
JSONObject obj =newJSONObject();  
try  
{  
obj.put("username", bean.getUserName());  
obj.put("password", bean.getPassWord());  
}catch(Exception e){}  
array.add(obj);  
}  
pw.write(array.toString());  
System.out.println(array.toString());

**android端接收：**

String urlAddress = "http://192.168.1.102:8080/qualityserver/result.do";  
String body =  
getContent(urlAddress);  
JSONArray array =newJSONArray(body);  
for(inti=0;i<array.length();i++)  
{  
obj = array.getJSONObject(i);  
sb.append("用户名:").append(obj.getString("username")).append("\t");  
sb.append("密码:").append(obj.getString("password")).append("\n");  
  
HashMap<String, Object> map =newHashMap<String, Object>();  
try{  
userName = obj.getString("username");  
passWord = obj.getString("password");  
}catch(JSONException e) {  
e.printStackTrace();  
}  
map.put("username", userName);  
map.put("password", passWord);  
listItem.add(map);  
  
}  
  
}catch(Exception e) {  
//TODO Auto-generated catch block  
e.printStackTrace();  
}  
  
if(sb!=null)  
{  
showResult.setText("用户名和密码信息：");  
showResult.setTextSize(20);  
}else  
extracted();  
  
//设置adapter  
SimpleAdapter simple =newSimpleAdapter(this,listItem,  
android.R.layout.simple\_list\_item\_2,  
newString[]{"username","password"},  
newint[]{android.R.id.text1,android.R.id.text2});  
listResult.setAdapter(simple);  
  
listResult.setOnItemClickListener(newOnItemClickListener() {  
@Override  
publicvoidonItemClick(AdapterView<?> parent, View view,  
intposition,longid) {  
intpositionId = (int) (id+1);  
Toast.makeText(MainActivity.this, "ID："+positionId, Toast.LENGTH\_LONG).show();  
  
}  
});  
}  
privatevoidextracted() {  
showResult.setText("没有有效的数据！");  
}  
//和服务器连接  
privateString getContent(String url)throwsException{  
StringBuilder sb =newStringBuilder();  
HttpClient client =newDefaultHttpClient();  
HttpParams httpParams =client.getParams();  
  
HttpConnectionParams.setConnectionTimeout(httpParams, 3000);  
HttpConnectionParams.setSoTimeout(httpParams, 5000);  
HttpResponse response = client.execute(newHttpGet(url));  
HttpEntity entity =response.getEntity();  
  
if(entity !=null){  
BufferedReader reader =newBufferedReader(newInputStreamReader  
(entity.getContent(),"UTF-8"),8192);  
String line =null;  
while((line= reader.readLine())!=null){  
sb.append(line +"\n");  
}  
reader.close();  
}  
returnsb.toString();  
}

URLConnection

HTTPClient

Proxies and SOCKS

Full support in Netscape browser, appletviewer, and applications (SOCKS: Version 4 only); no additional limitations from security policies.

Full support (SOCKS: Version 4 and 5); limited in applets however by security policies; in Netscape can't pick up the settings from the browser.

Authorization

Full support for Basic Authorization in Netscape (can use info given by the user for normal accesses outside of the applet); no support in appletviewer or applications.

Full support everywhere; however cannot access previously given info from Netscape, thereby possibly requesting the user to enter info (s)he has already given for a previous access. Also, you can add/implement additional authentication mechanisms yourself.

Methods

Only has GET and POST.

Has HEAD, GET, POST, PUT, DELETE, TRACE and OPTIONS, plus any arbitrary method.

Headers

Currently you can only set any request headers if you are doing a POST under Netscape; for GETs and the JDK you can't set any headers.   
Under Netscape 3.0 you can read headers only if the resource was returned with a Content-length header; if no Content-length header was returned, or under previous versions of Netscape, or using the JDK no headers can be read.

Allows any arbitrary headers to be sent and received.

Automatic Redirection Handling

Yes.

Yes (as allowed by the HTTP/1.1 spec).

Persistent Connections

No support currently in JDK; under Netscape uses HTTP/1.0 Keep-Alive's.

Supports HTTP/1.0 Keep-Alive's and HTTP/1.1 persistence.

Pipelining of Requests

No.

Yes.

Can handle protocols other than HTTP

Theoretically; however only http is currently implemented.

No.

Can do HTTP over SSL (https)

Under Netscape, yes. Using Appletviewer or in an application, no.

No (not yet).

Source code available

No.

Yes.