

第 5 天 Android 基础

第五章 网络编程（下）	2
1.1 前言	2
1.1.1 需求说明	2
1.1.2 服务器搭建	2
1.1.3 编写布局	5
1.1.4 添加权限	6
1.2 使用 HttpURLConnection 提交数据	6
1.2.1 代码	6
1.2.2 总结	11
1.3 使用 HttpClient 提交数据	11
1.3.1 get 方式提交	11
1.3.2 post 方式提交	13
1.3.3 总结	14
1.4 使用 AsyncHttpClient 框架提交数据	16
1.4.1 get 方式提交	17
1.4.2 post 方式提交	18
1.4.3 总结	19
1.5 JavaSE 实现多线程下载	21
1.5.1 多线程下载原理	21
1.5.2 代码实现	21
1.6 Android 实现多线程下载	25
1.6.1 ProgressBar 的使用	26
1.6.2 编写布局	26
1.6.3 编写代码	28
1.6.4 添加权限	33
1.7 xUtils 实现多线程下载	33
1.7.1 xUtils 简介	33
1.7.2 xUtils 之 HttpUtils 的使用	34

第五章 网络编程（下）

- ◆ 使用 `URLConnection` 提交数据
- ◆ 使用 `HttpClient` 提交数据
- ◆ 使用 `AsyncHttpClient` 框架提交数据
- ◆ Android 实现多线程下载
- ◆ 使用 `xUtils` 框架实现多线程下载

1.1 前言

移动互联网时代哪个 app 不需要跟服务器进行交互呢？Android 给服务器提交数据的方式都有哪些呢？这正是本文前 3 节讨论的话题，每一节介绍一种提交数据的方式，但是 Android 提交数据的方式绝非仅仅这三种，这里给出的只是最基础的 3 中方式。将这些基础的方式学会了，其他再高级的方式对我们来说也不过是小菜一碟了。

本文的 1.1、1.2、1.3 三节中使用的需求和布局是一模一样的，甚至 1.2 和 1.3 节的工程就是直接从 1.1 节中的工程拷贝过来的，唯一不同的就是使用提交数据的框架（类）不同。因此这里一次性将需求给出。

1.1.1 需求说明

如图 1-1 所示，界面整体采用垂直的线性布局，前两行为两个 `EditText`，分别代表用户名和密码。第三、四两行为两个 `Button`，前者点击后采用 `get` 方式提交数据，后者点击后采用 `post` 方式提交数据。数据提交成功后，服务器会有返回值，并将返回值用 `Toast` 显示出来。

1.1.2 服务器搭建

服务端采用 `Servlet` 编写，名为 `LoginServlet`，并使用 `Tomcat` 作为其服务器。`LoginServlet.java` 源码见【文件 1-1】，其中黄色高亮部分为核心代码。该 `Servlet` 在 `web.xml` 中的配置见【文件 1-2】。因为服务器不是本文的重点，因此这里只简单介绍。

【文件 1-1】 `LoginServlet.java`

```
1. package com.itheima.servlet;  
2.  
3. import java.io.IOException;  
4. import javax.servlet.ServletException;
```

```
5. import javax.servlet.http.HttpServlet;
6. import javax.servlet.http.HttpServletRequest;
7. import javax.servlet.http.HttpServletResponse;
8.
9. public class LoginServlet extends HttpServlet {
10.
11. /**
12.  * Constructor of the object.
13.  */
14. public LoginServlet() {
15.     super();
16. }
17.
18. /**
19.  * Destruction of the servlet. <br>
20.  */
21. public void destroy() {
22.     super.destroy(); // Just puts "destroy" string in log
23.     // Put your code here
24. }
25.
26. /**
27.  * The doGet method of the servlet. <br>
28.  *
29.  * This method is called when a form has its tag value method equals to get.
30.  *
31.  * @param request the request send by the client to the server
32.  * @param response the response send by the server to the client
33.  * @throws ServletException if an error occurred
34.  * @throws IOException if an error occurred
35.  */
36. public void doGet(HttpServletRequest request, HttpServletResponse response)
37. throws ServletException, IOException {
38.     request.setCharacterEncoding("utf-8");
39.     String username = request.getParameter("username");
40.     String password = request.getParameter("password");
41.     if ("GET".equals(request.getMethod().toUpperCase())) {
42.         byte[] bytes = username.getBytes("iso-8859-1");
43.         username = new String(bytes, "utf-8");
44.     }
45.
46.     System.out.println("usernmae===="+username);
47.     System.out.println("password=== "+password);
48.     response.setCharacterEncoding("utf-8");
49.     response.getWriter().write("成功收到信息"+username+"/"+password);
```

```
50. }
51.
52. /**
53.  * The doPost method of the servlet. <br>
54.  *
55.  * This method is called when a form has its tag value method equals to post.
56.  *
57.  * @param request the request send by the client to the server
58.  * @param response the response send by the server to the client
59.  * @throws ServletException if an error occurred
60.  * @throws IOException if an error occurred
61.  */
62. public void doPost(HttpServletRequest request, HttpServletResponse response)
63. throws ServletException, IOException {
64.
65.     doGet(request, response);
66. }
67.
68. /**
69.  * Initialization of the servlet. <br>
70.  *
71.  * @throws ServletException if an error occurs
72.  */
73. public void init() throws ServletException {
74.     // Put your code here
75. }
76. }
```

【文件 1-2】 web.xml

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <web-app version="2.5"
3.     xmlns="http://java.sun.com/xml/ns/javaee"
4.     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5.     xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
6.         http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
7.     <servlet>
8.         <servlet-name>LoginServlet</servlet-name>
9.         <servlet-class>com.itheima.servlet.LoginServlet</servlet-class>
10.     </servlet>
11.     <servlet>
12.         <servlet-name>FileuploadServlet</servlet-name>
13.         <servlet-class>com.itheima.servlet.FileuploadServlet</servlet-class>
14.     </servlet>
15.     <servlet-mapping>
```

```
16.     <servlet-name>LoginServlet</servlet-name>
17.     <url-pattern>/servlet/LoginServlet</url-pattern>
18. </servlet-mapping>
19. <servlet-mapping>
20.     <servlet-name>FileuploadServlet</servlet-name>
21.     <url-pattern>/servlet/FileuploadServlet</url-pattern>
22. </servlet-mapping>
23. <welcome-file-list>
24.     <welcome-file>index.jsp</welcome-file>
25. </welcome-file-list>
26. </web-app>
27.
```



图 1-1 多种方式实现用户登录（数据提交）

1.1.3 编写布局

考虑到 1.2、1.3、1.4 节使用的工程布局是一模一样的，因此在这里先将布局给出。

【文件 1-3】 activity_main.java

```
1. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2.     xmlns:tools="http://schemas.android.com/tools"
3.     android:layout_width="match_parent"
4.     android:layout_height="match_parent"
5.     android:orientation="vertical">
6.
7.     <EditText
```

```
8.         android:id="@+id/et_username"
9.         android:layout_width="match_parent"
10.        android:layout_height="wrap_content"
11.        android:hint="请输入用户名" />
12.    <EditText
13.        android:id="@+id/et_password"
14.        android:inputType="textPassword"
15.        android:layout_width="match_parent"
16.        android:layout_height="wrap_content"
17.        android:hint="请输入密码" />
18.    <Button
19.        android:onClick="login"
20.        android:layout_width="match_parent"
21.        android:layout_height="wrap_content"
22.        android:layout_gravity="right"
23.        android:text="get 方式登录" />
24.    <Button
25.        android:onClick="login2"
26.        android:layout_width="match_parent"
27.        android:layout_height="wrap_content"
28.        android:layout_gravity="right"
29.        android:text="post 方式登录" />
30.
31.</LinearLayout>
```

1.1.4 添加权限

凡是网络访问的操作，必须添加如下权限。

```
<uses-permission android:name="android.permission.INTERNET"/>
```

1.2 使用 HttpURLConnection 提交数据

1.2.1 代码

MainActivity.java 代码清单见【文件 1-4】。

【文件 1-4】 MainActivity.java

```
1. package com.itheima.userlogin;
2.
3. import java.io.InputStream;
4. import java.io.OutputStream;
5. import java.net.HttpURLConnection;
```

```
6. import java.net.URL;
7. import java.net.URLEncoder;
8. import com.itheima.userlogin1.R;
9. import android.os.Bundle;
10. import android.os.Handler;
11. import android.app.Activity;
12. import android.text.TextUtils;
13. import android.view.View;
14. import android.widget.EditText;
15. import android.widget.Toast;
16. /**
17.  * 使用 HttpURLConnection 提交数据
18.  * 在该案例中分别使用了 get 和 post 两种
19.  * 方式提交数据。
20.  * 大家可以对比这两种方式使用上的不同。
21.  *
22.  * @author wzy 2015-11-5
23.  *
24.  */
25. public class MainActivity extends Activity {
26.
27.     protected static final int TIME_OUT = 5000;
28.     private EditText et_password;
29.     private EditText et_username;
30.     private Handler handler = new Handler(){
31.         @Override
32.         public void handleMessage(android.os.Message msg) {
33.             switch (msg.what) {
34.                 case RESULT_OK:
35.                     Toast.makeText(MainActivity.this, "成功:"
36. +msg.obj.toString(), Toast.LENGTH_LONG).show();
37.                     break;
38.                 case RESULT_CANCELED:
39.                     Toast.makeText(MainActivity.this, "失败:"
40. +msg.obj.toString(), Toast.LENGTH_LONG).show();
41.                     break;
42.
43.                 default:
44.                     break;
45.             }
46.         };
47.     };
48.
49.     @Override
50.     protected void onCreate(Bundle savedInstanceState) {
```

```
51.     super.onCreate(savedInstanceState);
52.     setContentView(R.layout.activity_main);
53.     //获取控件
54.     et_username = (EditText) findViewById(R.id.et_username);
55.     et_password = (EditText) findViewById(R.id.et_password);
56. }
57. /**
58.  * 使用 get 方式完成用户的登录
59.  * @param view
60.  */
61. public void login(View view){
62.     //获取用户数据
63.     final String username = et_username.getText().toString().trim();
64.     final String password = et_password.getText().toString().trim();
65.     //校验数据
66.     if (TextUtils.isEmpty(password) || TextUtils.isEmpty(username)) {
67.         Toast.makeText(this, "用户名或密码不能为空!", Toast.LENGTH_SHORT).show();
68.         return;
69.     }
70.     //开启子线程
71.     new Thread(new Runnable() {
72.
73.         @Override
74.         public void run() {
75.             /*
76.              * 因为是 get 方式提交参数，因此对提交的参数必须使用
77.              * URLEncoder.encode(String) 方法进行编码，
78.              * 该编码可以将中文、空格、特殊字符等转义为 16 进制的数字，
79.              * 这样可以兼容不同平台的差异性，而 Tomcat 内部会自动将这些
80.              * 16 进制数给重新变为普通文本。
81.              */
82.             String path =
83. "http://10.0.2.2:8080/userlogin/servlet/LoginServlet?username="
84. +URLEncoder.encode(username)+"&password="+password;
85.             try {
86.                 URL url = new URL(path);
87.                 HttpURLConnection connection =
88. (HttpURLConnection) url.openConnection();
89.                 //配置参数
90.                 connection.setRequestMethod("GET");
91.                 connection.setConnectTimeout(TIME_OUT);
92.                 connection.setReadTimeout(TIME_OUT);
93.                 //打开链接
94.                 connection.connect();
```



```
95.          //获取状态码
96.          int responseCode = connection.getResponseCode();
97.          /*
98.          * 判断返回的状态码是否等于 200,
99.          * 如果返回 200 则代表请求成功
100.         */
101.         if (200==responseCode) {
102.             //获取返回值
103.             InputStream inputStream = connection.getInputStream();
104.             //将字节输入流转化为字符串
105.             String data = StreamUtils.inputStream2String(inputStream);
106.             //将数据通过 handler 发送出去
107.             handler.obtainMessage(RESULT_OK, data).sendToTarget();
108.         }else {
109.             //如果返回状态码不等于 200 则代码请求失败
110.             //将失败消息也发送出去
111.             handler.obtainMessage(RESULT_CANCELED, responseCode).sendToTarget();
112.         }
113.
114.         } catch (Exception e) {
115.             e.printStackTrace();
116.             //将异常消息发送出去
117.             handler.obtainMessage(RESULT_CANCELED, e).sendToTarget();
118.         }
119.     }
120.     }).start();
121. }
122. /**
123.  * 使用 post 方式完成用户的登录
124.  * @param view
125.  */
126. public void login2(View view){
127.     //获取用户数据
128.     final String username = et_username.getText().toString().trim();
129.     final String password = et_password.getText().toString().trim();
130.     //校验数据
131.     if (TextUtils.isEmpty(password)||TextUtils.isEmpty(username)) {
132.         Toast.makeText(this, "用户名或密码不能为空!", Toast.LENGTH_SHORT).show();
133.         return;
134.     }
135.     //开启子线程
136.     new Thread(new Runnable() {
137.
138.         @Override
139.         public void run() {
```

```
140.         String path =
141.         "http://10.0.2.2:8080/userlogin/servlet/LoginServlet";
142.         try {
143.             URL url = new URL(path);
144.             HttpURLConnection connection =
145. (HttpURLConnection) url.openConnection();
146.             //配置参数
147.             connection.setRequestMethod("POST");
148.             /*
149.              * 设置该参数,才能以流的形式提交数据
150.              * 需要将要提交的数据转换为字节输出流
151.              */
152.             connection.setDoOutput(true);
153.             connection.setConnectTimeout(TIME_OUT);
154.             connection.setReadTimeout(TIME_OUT);
155.             //将提交的参数进行 URL 编码
156.             String param =
157. "username="+URLEncoder.encode(username)+"&password="+password;
158.             /*
159.              * 设置请求属性,相当于封装 http 的请求头参数
160.              */
161.             //设置请求体的长度
162.             connection.setRequestProperty("Content-Length", param.length()+"");
163.             //设置请求体的类型
164.             connection.setRequestProperty(
165. "Content-Type", application/x-www-form-urlencoded");
166.             //打开链接
167.             connection.connect();
168.             //获取输出流
169.             OutputStream os = connection.getOutputStream();
170.             //通过输出流将要提交的数据提交出去
171.             os.write(param.getBytes());
172.             //关闭输出流
173.             os.close();
174.             //判断状态码
175.             int responseCode = connection.getResponseCode();
176.             if (200==responseCode) {
177.                 //获取返回值
178.                 InputStream inputStream = connection.getInputStream();
179.                 //将字节流转换为字符串
180.                 String data = StreamUtils.inputStream2String(inputStream);
181.                 handler.obtainMessage(RESULT_OK, data).sendToTarget();
182.             }else {
183.                 handler.obtainMessage(RESULT_CANCELED, responseCode).sendToTarget();
```

```
184.         }
185.
186.         } catch (Exception e) {
187.             e.printStackTrace();
188.             handler.obtainMessage(RESULT_CANCELED, e).sendToTarget();
189.         }
190.     }
191. }).start();
192. }
193.
194. }
```

1.2.2 总结

在 http 协议中，get 请求协议没有请求体，只有请求头和请求行，因此如果想使用 get 方式提交数据，只能通过在 url 后面加上要提交的参数。这也意味着 get 方式只能提交比较小的参数。

如果 get 方式提交的参数有特殊字符或者中文字符那么必须对其进行 URL 编码，不然会导致服务器接收到的数据乱码。

对于 post 请求，提交的数据位于请求体中，因此需要设置 `connection.setDoOutput(true)` 参数，该参数设置后才允许将提交的数据通过输出流的形式提交。不过需要说明的是虽然采用 post 方式提交，依然需要将参数进行 URL 编码。

其实当我们使用浏览器进行 form 表单提交的时候，浏览器内部已经帮我们实现了 URL 编码。因为我们这里是使用代码模拟浏览器的提交数据过程，因此需要使用代码特殊处理。

1.3 使用 HttpClient 提交数据

HttpClient 是 Apache Jakarta Common 下的子项目，提供了高效的、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。

HttpClient 被内置到 Android SDK 中，因此可以在不添加任何额外 jar 包情况下，直接使用。

1.3.1 get 方式提交

在 1.2 节工程的基础上，只需要修改部分代码即可。因此这里只给出核心代码。

【文件 1-5】 get 方式提交数据代码片段

```
1. /**
2.  * HttpClient 使用 get 方式完成用户的登录
3.  *
4.  * @param view
5.  */
6. public void login3(View view) {
7.     // 获取用户数据
8.     final String username = et_username.getText().toString().trim();
```

```
9.      final String password = et_password.getText().toString().trim();
10.     // 校验数据
11.     if (TextUtils.isEmpty(password) || TextUtils.isEmpty(username)) {
12.         Toast.makeText(this, "用户名或密码不能为空!", Toast.LENGTH_SHORT).show();
13.         return;
14.     }
15.     // 开启子线程
16.     new Thread(new Runnable() {
17.
18.         @Override
19.         public void run() {
20.             String path =
21. "http://10.0.2.2:8080/userlogin/servlet/LoginServlet?username="
22. + URLEncoder.encode(username) + "&password=" + password;
23.             try {
24.                 // 创建一个 httpClient 对象
25.                 HttpClient client = new DefaultHttpClient();
26.                 // 创建一个请求方式
27.                 HttpGet request = new HttpGet(path);
28.                 // 执行操作
29.                 HttpResponse response = client.execute(request);
30.                 // 获取放回状态对象
31.                 StatusLine statusLine = response.getStatusLine();
32.                 // 获取状态码
33.                 int statusCode = statusLine.getStatusCode();
34.                 if (200 == statusCode) {
35.                     // 获取服务器返回的对象
36.                     HttpEntity entity = response.getEntity();
37.                     // 获取输入流
38.                     InputStream inputStream = entity.getContent();
39.                     // 将输入流转化为字符串
40.                     String data = StreamUtils.inputStream2String(inputStream);
41.                     handler.obtainMessage(RESULT_OK, data).sendToTarget();
42.                 } else {
43.                     handler.obtainMessage(RESULT_CANCELED, statusCode).sendToTarget();
44.                 }
45.
46.             } catch (Exception e) {
47.                 e.printStackTrace();
48.                 handler.obtainMessage(RESULT_CANCELED, e).sendToTarget();
49.             }
50.         }
51.     }).start();
52. }
```

1.3.2 post 方式提交

【文件 1-6】 post 方式提交数据代码片段

```
1. /**
2.  * HttpClient 使用 post 方式完成用户的登录
3.  *
4.  * @param view
5.  */
6. public void login4(View view) {
7.     // 获取用户数据
8.     final String username = et_username.getText().toString().trim();
9.     final String password = et_password.getText().toString().trim();
10.    // 校验数据
11.    if (TextUtils.isEmpty(password) || TextUtils.isEmpty(username)) {
12.        Toast.makeText(this, "用户名或密码不能为空!", Toast.LENGTH_SHORT).show();
13.        return;
14.    }
15.    // 开启子线程
16.    new Thread(new Runnable() {
17.
18.        @Override
19.        public void run() {
20.            String path = "http://10.0.2.2:8080/userlogin/servlet/LoginServlet";
21.            try {
22.                // 创建一个 httpClient 对象
23.                HttpClient client = new DefaultHttpClient();
24.                // 创建一个 post 请求方式
25.                HttpPost request = new HttpPost(path);
26.                // 创建集合用于添加请求的数据
27.                List<BasicNameValuePair> parameters =
28.                new ArrayList<BasicNameValuePair>();
29.                // 将请求数据添加到集合中
30.                parameters.add(new BasicNameValuePair("username", username));
31.                parameters.add(new BasicNameValuePair("password", password));
32.                /*
33.                 * 创建请求实体对象
34.                 * UriEncodedFormEntity 是 HttpEntity 的子类，
35.                 该类专门用于封装字符串类型的参数
36.                 * 指定数据的编码格式
37.                 */
38.                HttpEntity requestEntity =
39.                new UriEncodedFormEntity(parameters, "utf-8");
40.                // 设置请求体
41.                request.setEntity(requestEntity);
```

```
42.          // 执行操作
43.          HttpResponse response = client.execute(request);
44.          // 获取放回状态对象
45.          StatusLine statusLine = response.getStatusLine();
46.          // 获取状态码
47.          int statusCode = statusLine.getStatusCode();
48.          if (200 == statusCode) {
49.              // 获取服务器返回的对象
50.              HttpEntity entity = response.getEntity();
51.              // 获取输入流
52.              InputStream inputStream = entity.getContent();
53.              // 将输入流转化为字符串
54.              String data = StreamUtils.inputStream2String(inputStream);
55.              handler.obtainMessage(RESULT_OK, data).sendToTarget();
56.          } else {
57.              handler.obtainMessage(RESULT_CANCELED, statusCode).sendToTarget();
58.          }
59.
60.      } catch (Exception e) {
61.          e.printStackTrace();
62.          handler.obtainMessage(RESULT_CANCELED, e).sendToTarget();
63.      }
64.  }
65.  }).start();
66. }
```

1.3.3 总结

一、get 请求方式步骤

1、创建一个 httpClient 对象

```
HttpClient client = new DefaultHttpClient();
```

2、创建一个请求方式

```
String path =
"http://10.0.2.2:8080/userlogin/servlet/LoginServlet?username="
+ URLEncoder.encode(username) + "&password=" + password;
//因为是 get 方式，因此需要在 path 中添加请求参数
HttpGet request = new HttpGet(path);
```

3、开始访问网络，并接收返回值

```
HttpResponse response = client.execute(request);
```

4、获取返回状态码

```
//获取返回值的状态行
StatusLine statusLine = response.getStatusLine();
```

```
// 获取状态码
int statusCode = statusLine.getStatusCode();
```

5、判断状态码

```
//如果状态码为 200 则代表请求成功
if (200 == statusCode) {//TODO}
```

6、获取返回数据

```
// 获取服务器返回的对象
HttpEntity entity = response.getEntity();
// 获取输入流
InputStream inputStream = entity.getContent();
```

二、post 请求方式步骤

1、创建一个 httpClient 对象

```
HttpClient client = new DefaultHttpClient();
```

2、 创建一个 post 请求方式

```
String path = "http://10.0.2.2:8080/userlogin/servlet/LoginServlet";
HttpPost request = new HttpPost(path);
```

3、 设置请求体

```
//创建集合用于添加请求的数据
List<BasicNameValuePair> parameters =
new ArrayList<BasicNameValuePair>();

//将请求数据添加到集合中
parameters.add(new BasicNameValuePair("username", username));
parameters.add(new BasicNameValuePair("password", password));
/*
 * 创建请求实体对象
 * UriEncodedFormEntity 是 HttpEntity 的子类，该类专门用于封装字符串类型的参数
 * 指定数据的编码格式
 */
HttpEntity requestEntity =
new UriEncodedFormEntity(parameters, "utf-8");
//设置请求体
request.setEntity(requestEntity);
```

4、 开始访问网络，并接收返回值

```
// 执行操作
HttpResponse response = client.execute(request);
```

5、获取返回状态码

```
//获取返回值的状态行
StatusLine statusLine = response.getStatusLine();
// 获取状态码
int statusCode = statusLine.getStatusCode();
```

6、判断状态码

```
//如果状态码为 200 则代表请求成功
if (200 == statusCode) {//TODO}
```

7、获取返回数据

```
// 获取服务器返回的对象
HttpEntity entity = response.getEntity();
// 获取输入流
InputStream inputStream = entity.getContent();
```

1.4 使用 AsyncHttpClient 框架提交数据

AsyncHttpClient 是开源免费的基于 HttpClient 的网络请求框架，其源码托管在 github 上。下载地址：<https://github.com/loopj/android-async-http>。

如图 1-2 所示为 AsyncHttpClient 在 github 的网页，大家可以点击右下角的按钮，将源码下载下来，然后将下载好的压缩包解压，把 src 目录中源码拷贝到自己的工程的 src 目录下，比如图 1-3 所示。

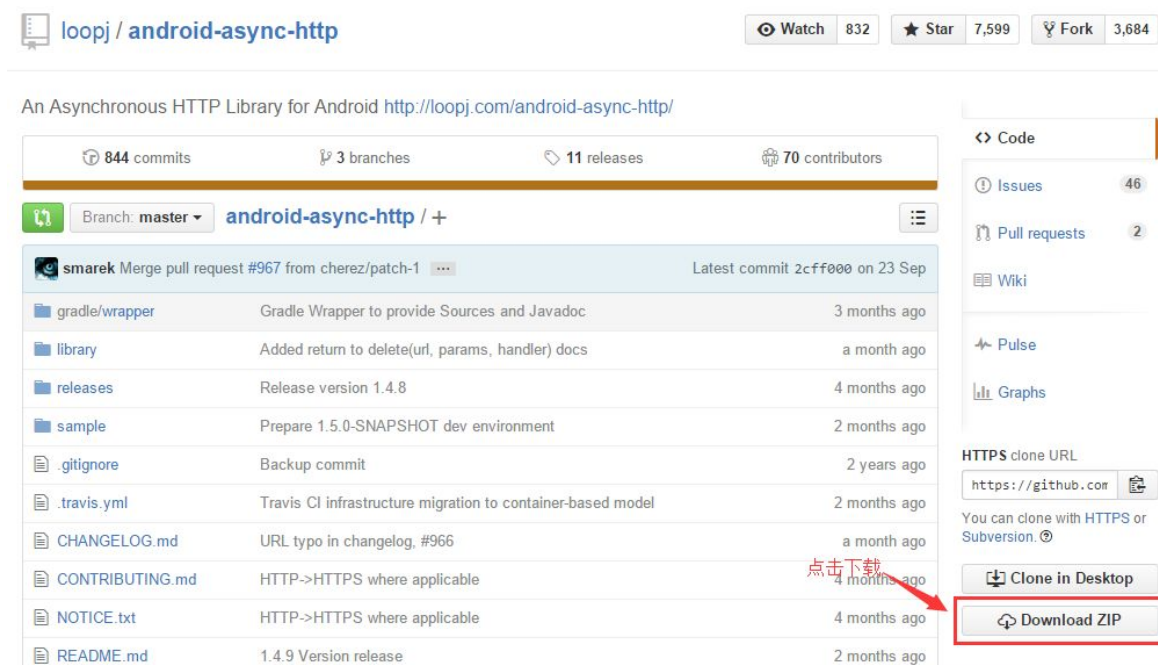


图 1-2 AsyncHttpClient 网页

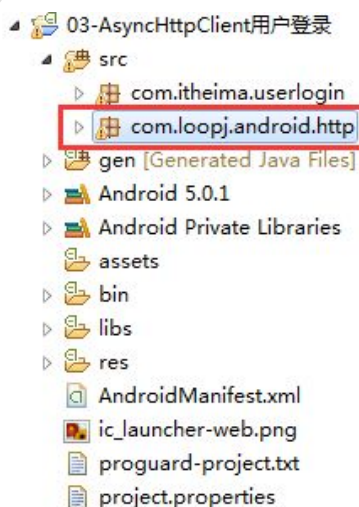


图 1-3 将 AsyncHttpClient 添加到工程中

1.4.1 get 方式提交

在 1.2 节工程的基础上，只需要修改部分代码即可。因此这里只给出核心代码。

【文件 1-7】 get 方式提交数据代码片段

```
1. /**
2.  * 使用 AsyncHttpClient 的 get 方式提交数据
3.  *
4.  * @param view
5.  */
6. public void login5(View view) {
7.     // 获取用户数据
8.     final String username = et_username.getText().toString().trim();
9.     final String password = et_password.getText().toString().trim();
10.    // 校验数据
11.    if (TextUtils.isEmpty(password) || TextUtils.isEmpty(username)) {
12.        Toast.makeText(this, "用户名或密码不能为空!", Toast.LENGTH_SHORT).show();
13.        return;
14.    }
15.    String path = "http://10.0.2.2:8080/userlogin/servlet/LoginServlet?username="
16. + URLEncoder.encode(username) + "&password=" + password;
17.    // 创建一个 AsyncHttpClient
18.    AsyncHttpClient client = new AsyncHttpClient();
19.    // 执行 get 方式请求
20.    client.get(path, new DataAsyncHttpResponseHandler() {
21.        /**
22.         * 请求网络是在子线程中进行的，当请求成功后回调 onSuccess 方法，
23.         * 该方法是在主线程中被调用了，其内部是通过 Handler 实现的
24.         * 当请求成功后回调该方法
25.         * 参数 1: 返回的状态码
26.         * 参数 2: 响应头
27.         * 参数 3: 返回的数据
28.         */
29.        @Override
30.        public void onSuccess(int statusCode, Header[] headers, byte[] responseBody) {
31.            Log.d("tag", Thread.currentThread().getName());
32.            // 将 responseBody 字节数组转化为字符串
33.            Toast.makeText(MainActivity.this, new String(responseBody),
34.            Toast.LENGTH_LONG).show();
35.        }
36.        /**
37.         * 当请求失败后回调该方法，该方法依然在主线程中被执行
38.         */
```

```
39.         @Override
40.         public void onFailure(int statusCode, Header[] headers, byte[] responseBody,
41.             Throwable error) {
42.             Toast.makeText(MainActivity.this, new String(responseBody),
43.                 Toast.LENGTH_LONG).show();
44.         }
45.     });
46. }
```

1.4.2 post 方式提交

【文件 1-8】 post 方式提交数据代码片段

```
1. /**
2.  * 使用 AsyncHttpClient 的 post 方式提交数据
3.  *
4.  * @param view
5.  */
6. public void login6(View view) {
7.     // 获取用户数据
8.     final String username = et_username.getText().toString().trim();
9.     final String password = et_password.getText().toString().trim();
10.    // 校验数据
11.    if (TextUtils.isEmpty(password) || TextUtils.isEmpty(username)) {
12.        Toast.makeText(this, "用户名或密码不能为空!", Toast.LENGTH_SHORT).show();
13.        return;
14.    }
15.    String path = "http://10.0.2.2:8080/userlogin/servlet/LoginServlet";
16.    // 创建一个 AsyncHttpClient
17.    AsyncHttpClient client = new AsyncHttpClient();
18.    //封装请求参数
19.    RequestParams params = new RequestParams();
20.    params.put("username", username);
21.    params.put("password", password);
22.    /**
23.     * 执行 post 请求
24.     * 参数 1: url 地址
25.     * 参数 2: 请求参数
26.     * 参数 3: 回调接口，在该接口中实现成功和失败方法，该过程是异步的。
27.     */
28.    client.post(path, params, new DataAsyncHttpResponseHandler() {
29.
30.        @Override
31.        public void onSuccess(int statusCode, Header[] headers, byte[] responseBody)
```

```
{
    32.         Toast.makeText(MainActivity.this, new String(responseBody),
    33.         Toast.LENGTH_LONG).show();
    34.     }
    35.
    36.     @Override
    37.     public void onFailure(int statusCode, Header[] headers, byte[] responseBody,
    38.     Throwable error) {
    39.         Toast.makeText(MainActivity.this, new String(responseBody),
    40.         Toast.LENGTH_LONG).show();
    41.     }
    42. });
    43. }
```

1.4.3 总结

一、get 请求方式步骤

1、创建一个 AsyncHttpClient 对象

```
AsyncHttpClient client = new AsyncHttpClient();
```

2、执行 get 方法

```
/*
 * 执行 get 方式请求
 * 参数 1: 请求的 url 地址
 * 参数 2: 回调接口。在该接口中实现相应成功和失败方法，该过程是异步的。
 */
client.get(path, new DataAsyncHttpResponseHandler() {
    /*
     * 请求网络是在子线程中进行的，当请求成功后回调 onSuccess 方法，
     * 该方法是在主线程中被调用了，其内部是通过 Handler 实现的
     * 当请求成功后回调该方法
     * 参数 1: 返回的状态码
     * 参数 2: 响应头
     * 参数 3: 返回的数据
     */
    @Override
    public void onSuccess(int statusCode, Header[] headers, byte[] responseBody) {
        Log.d("tag", Thread.currentThread().getName());
        //将 responseBody 字节数组转化为字符串
        Toast.makeText(MainActivity.this, new String(responseBody),
        Toast.LENGTH_LONG).show();
    }
    /*
     * 当请求失败后回调该方法，该方法依然在主线程中被执行
```

```
        */
        @Override
        public void onFailure(int statusCode, Header[] headers, byte[] responseBody,
        Throwable error) {
            Toast.makeText(MainActivity.this, new String(responseBody),
            Toast.LENGTH_LONG).show();
        }
    });
}
```

在执行 `get` 方法的时候需要传递一个 `ResponseHandlerInterface` 接口的子类，在上面使用了 `DataAsyncHttpResponseHandler` 抽象类，除此之外其实 `ResponseHandlerInterface` 还有很多子类，比如用于接收 `json` 字符串的 `TextHttpResponseHandler`，用于接收文件下载的 `TextHttpResponseHandler`。因为我们只需要接收字节数组，并将字节数组转化为字符串即可，因此我们使用 `DataAsyncHttpResponseHandler` 抽象类。

二、post 方式请求步骤

1、创建一个 `AsyncHttpClient` 对象

```
AsyncHttpClient client = new AsyncHttpClient();
```

2、封装请求参数

```
//封装请求参数
RequestParams params = new RequestParams();
params.put("username", username);
params.put("password", password);
```

3、执行 `post` 方法

```
/*
 * 执行 post 请求
 * 参数 1: url 地址
 * 参数 2: 请求参数
 * 参数 3: 回调接口，在该接口中实现成功和失败方法，该过程是异步的。
 */
client.post(path, params, new DataAsyncHttpResponseHandler() {

    @Override
    public void onSuccess(int statusCode, Header[] headers, byte[] responseBody) {
        Toast.makeText(MainActivity.this, new String(responseBody),
        Toast.LENGTH_LONG).show();
    }

    @Override
    public void onFailure(int statusCode, Header[] headers, byte[] responseBody,
    Throwable error) {
        Toast.makeText(MainActivity.this, new String(responseBody),
        Toast.LENGTH_LONG).show();
    }
});
```

`post` 方法跟 `get` 方式不同的就是多了一个 `RequestParams` 参数。

1.5 JavaSE 实现多线程下载

1.5.1 多线程下载原理

多线程下载就是将同一个网络上的原始文件根据线程个数分成均等份，然后每个单独的线程下载对应的一部分，然后再将下载好的文件按照原始文件的顺序“拼接”起来就构成了完整的文件了。这样就大大提高了文件的下载效率。

多线程下载大致可分为以下几个步骤：

一、获取服务器上的目标文件的大小

显然这一步是需要先访问一下网络，只需要获取到目标文件的总大小即可。目的是为了计算每个线程应该分配的下载任务。

二、计算每个线程下载的起始位置和结束位置

我们可以把原始文件当成一个字节数组，每个线程只下载该“数组”的指定起始位置和指定结束位置之间的部分。在第一步中我们已经知道了“数组”的总长度。因此只要再知道总共开启的线程的个数就好计算每个线程要下载的范围了。

每个线程需要下载的字节个数（blockSize）=总字节数（totalSize）/线程数（threadCount）。

假设给线程按照 0,1,2,3...n 的方式依次进行编号，那么第 n 个线程下载文件的范围为：

起始脚标 $\text{startIndex} = n * \text{blockSize}$ 。

结束脚标 $\text{endIndex} = (n-1) * \text{blockSize} - 1$ 。

考虑到 $\text{totalSize} / \text{threadCount}$ 不一定能整除，因此对已最后一个线程应该特殊处理，最后一个线程的起始脚标计算公式不变，但是结束脚标 $\text{endIndex} = \text{totalSize} - 1$ ；即可。

三、在本地创建一个跟原始文件同样大小的文件

在本地可以通过 `RandomAccessFile` 创建一个跟目标文件同样大小的文件，该 api 支持文件任意位置的读写操作。这样就给多线程下载提供了方便，每个线程只需在指定起始和结束脚标范围内写数据即可。

四、开启多个子线程开始下载

五、记录下载进度

为每一个单独的线程创建一个临时文件，用于记录该线程下载的进度。对于单独的一个线程，每下载一部分数据就在本地文件中记录下当前下载的字节数。这样子如果下载任务异常终止了，那么下次重新开始下载时就可以接着上次的进度下载。

六、删除临时文件

当多个线程都下载完成之后，最后一个下载完的线程将所有的临时文件删除。

1.5.2 代码实现

【文件 1-9】多线程下载 JavaSE 代码

```
1. package com.itheima.se.download;
2.
3. import java.io.BufferedReader;
4. import java.io.BufferedWriter;
5. import java.io.File;
6. import java.io.FileReader;
```

```
7. import java.io.FileWriter;
8. import java.io.InputStream;
9. import java.io.RandomAccessFile;
10. import java.net.HttpURLConnection;
11. import java.net.URL;
12. /*
13.  * 实现多线程断点续下载
14.  */
15. public class MultiDownlodTest {
16.
17.     /**
18.      * @param args
19.      */
20.     public static void main(String[] args) {
21.         String sourcePath = "http://localhost:8080/FeiQ.exe";
22.         String targetPath = "d://FeiQ.exe";
23.         new MultiDownloader(sourcePath,3,targetPath).download();
24.
25.     }
26.
27.     static class MultiDownloader {
28.         private String sourcePath;
29.         private int threadCount;
30.         private String targetPath;
31.         //未完成任务的线程
32.         private int threadRunning;
33.         public MultiDownloader(String sourcePath,int threadCount,String targetPath){
34.             this.sourcePath = sourcePath;
35.             this.threadCount = threadCount;
36.             this.targetPath = targetPath;
37.         }
38.         public void download(){
39.             try {
40.                 URL url = new URL(sourcePath);
41.                 HttpURLConnection connection =
42. (HttpURLConnection) url.openConnection();
43.                 //配置连接参数
44.                 connection.setRequestMethod("GET");
45.                 connection.setConnectTimeout(5000);
46.                 //打开连接
47.                 connection.connect();
48.                 int responseCode = connection.getResponseCode();
49.                 if (responseCode==200) {
50.                     int totalSize = connection.getContentLength();
```

```
51.          //计算每个线程下载的平均字节数
52.          int avgSize = totalSize/threadCount;
53.          for(int i=0;i<threadCount;i++){
54.              final int startIndex = i*avgSize;
55.              int endIndex = 0;
56.              if (i==threadCount-1) {
57.                  endIndex = totalSize-1;
58.              }else {
59.                  endIndex = (i+1)*avgSize-1;
60.              }
61.              threadRunning++;
62.              //开启子线程,实现下载
63.              new MyDownloadThread(i,
64. startIndex, endIndex,targetPath,sourcePath).start();
65.          }
66.
67.          }else {
68.              System.out.println("返回码为:"+responseCode+" 请求文件长度失败!");
69.              return;
70.          }
71.
72.      } catch (Exception e) {
73.          e.printStackTrace();
74.      }
75.  }
76.  /**
77.   *
78.   * @author wzy 2015-10-26
79.   *
80.   */
81.  class MyDownloadThread extends Thread{
82.      private int id;
83.      private int startIndex;
84.      private int endIndex;
85.      private String targetPath;
86.      private String sourcePath;
87.      public MyDownloadThread(int id,int startIndex,
88. int endIndex,String targetPath,String sourcePath){
89.          this.id = id;
90.          this.startIndex = startIndex;
91.          this.endIndex = endIndex;
92.          this.targetPath = targetPath;
93.          this.sourcePath = sourcePath;
94.      }
95.      @Override
```

```
96.         public void run() {
97.             try {
98.                 URL url = new URL(sourcePath);
99.                 HttpURLConnection connection =
100. (HttpURLConnection) url.openConnection();
101.                 //设置断点下载参数
102.                 connection.setRequestMethod("GET");
103.                 File file = new File("d://" + id + ".tmp");
104.                 /*
105.                  * 该属性设置后,返回的状态码就不再是 200,而是 206
106.                  */
107.                 int currentIndex = -1;
108.                 //读进度
109.                 if (file.exists()) {
110.                     BufferedReader reader =
111. new BufferedReader(new FileReader(file));
112.                     String readLine = reader.readLine();
113.                     currentIndex = Integer.valueOf(readLine);
114.                     reader.close();
115.                 }else {
116.                     currentIndex = startIndex;
117.                 }
118.                 ////只有设置了该属性才能下载指定范围的文件
119.                 connection.setRequestProperty("Range", "bytes=" + currentIndex + "-" + endIndex);
120.                 connection.setConnectTimeout(5000);
121.                 connection.setReadTimeout(5000);
122.                 connection.connect();
123.                 int responseCode = connection.getResponseCode();
124.                 //因为请求的是一个文件的部分，因此返回的状态码是 206
125.                 if (responseCode == 206) {
126.                     InputStream inputStream = connection.getInputStream();
127.                     //支持随机读写的文件类
128.                     RandomAccessFile raf =
129. new RandomAccessFile(targetPath, "rws");
130.                     //将文件指针定位到要写的位置
131.                     raf.seek(currentIndex);
132.                     byte[] buffer = new byte[1024 * 16];
133.                     int len = -1;
134.                     int totalDownloaded = 0;
135.                     while ((len = inputStream.read(buffer)) != -1) {
136.                         raf.write(buffer, 0, len);
137.                         totalDownloaded += len;
138.                     } //写进度
139.                     BufferedWriter writer =
```



```
140.     new BufferedWriter(new FileWriter(file));
141.                                     writer.write(currentIndex+totalDownloaded+"");
142.                                     writer.close();
143.                                     System.out.println(id+"下载了:"+totalDownloaded+", "+
144. (totalDownloaded+currentIndex-startIndex)*100/(endIndex-startIndex)+"%");
145.                                     }
146.                                     raf.close();
147.                                     inputStream.close();
148.                                     connection.disconnect();
149.                                     //线程执行完了
150.                                     threadRunning--;
151.                                     if (threadRunning==0) {
152.                                         for(int i=0;i<threadCount;i++){
153.                                             File tmpFile = new File("d://" + i + ".tmp");
154.                                             tmpFile.delete();
155.                                         }
156.                                     }
157.                                     }else {
158.                                         System.out.println("返回的状态码为:"+responseCode+
159. " 下载失败!");
160.                                     }
161.                                     } catch (Exception e) {
162.                                         e.printStackTrace();
163.                                     }
164.                                     }
165.                                     }
166.
167.     }
168.
169. }
170.
```

新技能：

1、多线程下载文件的请求属性

如果我们想请求服务器上某文件的一部分，而不是将整个文件都下载下来，那么就必须设置如下属性：`connection.setRequestProperty("Range", "bytes="+currentIndex+"-"+endIndex);`

2、请求部分文件返回成功的状态码是 206

当我们请求部分文件成功后，服务器返回的状态码不是 200，而是 206。

1.6 Android 实现多线程下载



将上面 JavaSE 实现多线程下载的代码经过一定的改造，便可移植到 Android 上。有所不同的是 Android 有界面可以跟用户进行良好的交互，在界面（如图 1-4）上让用户输入原文件地址、线程个数，然后点击

确定开始下载。为了让用户可以清晰的看到每个线程下载的进度根据线程个数动态的生成等量的进度条（ProgressBar）。



图 1-4 Android 实现多线程下载

1.6.1 ProgressBar 的使用

ProgressBar 是一个进度条控件，用于显示一项任务的完成进度。其有两种样式，一种是圆形的 ，该种样式是系统默认的，由于无法显示具体的进度值，适合用于不确定要等待多久的情形下；另一种是长条形的，，此类进度条有两种颜色，高亮颜色代表任务完成的总进度。对于我们下载任务来说，由于总任务（要下载的字节数）是明确的，当前已经完成的任务（已经下载的字节数）也是明确的，因此特别适合使用后者。

由于在我们的需求里 ProgressBar 是需要根据线程的个数动态添加的，而且要求是长条形的。因此可以事先在布局文件中编写好 ProgressBar 的样式。当需要用到的时候再将该布局（见【文件 1-11】）填充起来。

ProgressBar 的 max 属性代表其最大刻度值，progress 属性代表当前进度值。使用方法如下：

ProgressBar.setMax(int max);设置最大刻度值。

ProgressBar.setProgress(int progress);设置当前进度值。

给 ProgressBar 设置最大刻度值和修改进度值可以在子线程中操作的，其内部已经特殊处理过了，因此不需要再通过发送 Message 让主线程修改进度。

1.6.2 编写布局

这里给出两个布局，【文件 1-10】是 MainActivity 的布局，其显示效果如图 1-4。【文件 1-11】是 ProgressBar 布局。

【文件 1-10】 activity_main.java

```
1. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
2.     xmlns:tools="http://schemas.android.com/tools"
3.     android:layout_width="match_parent"
4.     android:layout_height="match_parent"
5.     android:orientation="vertical">
6.
7.     <EditText
8.         android:id="@+id/et_url"
9.         android:layout_width="match_parent"
10.        android:layout_height="wrap_content"
11.        android:text="http://10.0.2.2:8080/FeiQ.exe" />
12.    <LinearLayout
13.        android:layout_width="match_parent"
14.        android:layout_height="wrap_content"
15.        android:orientation="horizontal"
16.    >
17.        <EditText
18.            android:id="@+id/et_threadCount"
19.            android:layout_width="0dp"
20.            android:layout_weight="1"
21.            android:layout_height="wrap_content"
22.            android:text="3"
23.        />
24.        <Button
25.            android:onClick="download"
26.            android:layout_width="wrap_content"
27.            android:layout_height="wrap_content"
28.            android:text="下载"
29.        />
30.    </LinearLayout>
31.    <LinearLayout
32.        android:id="@+id/ll_pbs"
33.        android:layout_width="match_parent"
34.        android:layout_height="wrap_content"
35.        android:orientation="vertical"
36.    >
37.
38.    </LinearLayout>
39.
40. </LinearLayout>
41.
```

【文件 1-11】 progress_bar.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <ProgressBar xmlns:android="http://schemas.android.com/apk/res/android"
3.     style="?android:attr/progressBarStyleHorizontal"
4.     android:layout_width="match_parent"
```

```
5.     android:layout_height="wrap_content" >
6. </ProgressBar>
```

1.6.3 编写代码

【文件 1-12】 MainActivity.java

```
1. package com.itheima.android.downloader;
2.
3. import java.io.BufferedReader;
4. import java.io.BufferedWriter;
5. import java.io.File;
6. import java.io.FileReader;
7. import java.io.FileWriter;
8. import java.io.InputStream;
9. import java.io.RandomAccessFile;
10. import java.net.HttpURLConnection;
11. import java.net.URL;
12. import android.os.Bundle;
13. import android.support.v4.app.FragmentActivity;
14. import android.text.TextUtils;
15. import android.view.View;
16. import android.widget.EditText;
17. import android.widget.LinearLayout;
18. import android.widget.ProgressBar;
19. import android.widget.Toast;
20. /**
21.  * Android 实现多线程断点续传
22.  *
23.  * @author wzy 2015-11-6
24.  *
25.  */
26. public class MainActivity extends FragmentActivity {
27.
28.     private EditText et_url;
29.     private EditText et_threadCount;
30.     //用于添加 ProgressBar 的容器
31.     private LinearLayout ll_pbs;
32.
33.     @Override
34.     protected void onCreate(Bundle savedInstanceState) {
35.         super.onCreate(savedInstanceState);
36.         setContentView(R.layout.activity_main);
37.         //初始化控件
```

```
38.     et_url = (EditText) findViewById(R.id.et_url);
39.     et_threadCount = (EditText) findViewById(R.id.et_threadCount);
40.     ll_pbs = (LinearLayout) findViewById(R.id.ll_pbs);
41. }
42.
43. /**
44.  * 点击 Button 绑定的方法
45.  * 开始下载任务
46.  *
47.  * @param view
48.  */
49. public void download(View view) {
50.     // 获取用户输入的初始值
51.     final String sourcePath = et_url.getText().toString().trim();
52.     final int threadCount =
53. Integer.valueOf(et_threadCount.getText().toString().trim());
54.     // 校验数据
55.     if (TextUtils.isEmpty(sourcePath) || threadCount < 1) {
56.         Toast.makeText(this, "输入的内容不合法!", Toast.LENGTH_SHORT).show();
57.         return;
58.     }
59.     //存储到 Android 本地的位置
60.     final String targetPath = "/mnt/sdcard/FeiQ.exe";
61.     //初始化进度条
62.     //填充一个 ProgressBar
63.     for(int i=0;i<threadCount;i++){
64.         //将进度条布局填充为 ProgressBar
65.         ProgressBar pb = (ProgressBar) View.inflate(MainActivity.this,
66. R.layout.progress_bar, null);
67.         //将 ProgressBar 添加到 LinearLayout 中
68.         ll_pbs.addView(pb);
69.     }
70.     //开启子线程开始下载任务
71.     new Thread(new Runnable() {
72.
73.         @Override
74.         public void run() {
75.             //开启多线程下载器
76.             new MultiDownloader(sourcePath,threadCount,targetPath).download();
77.         }
78.     }).start();
79. }
80. class MultiDownloader {
81.     //服务器原始文件地址
82.     private String sourcePath;
```

```
83.    //线程个数
84.    private int threadCount;
85.    //本地目标存储文件路径
86.    private String targetPath;
87.    //未完成任务的线程
88.    private int threadRunning;
89.    public MultiDownloader(String sourcePath,int threadCount,String targetPath){
90.        this.sourcePath = sourcePath;
91.        this.threadCount = threadCount;
92.        this.targetPath = targetPath;
93.    }
94.    public void download(){
95.        try {
96.            URL url = new URL(sourcePath);
97.            HttpURLConnection connection = (HttpURLConnection) url.openConnection();
98.            //配置连接参数
99.            connection.setRequestMethod("GET");
100.           connection.setConnectTimeout(5000);
101.           //打开连接
102.           connection.connect();
103.           int responseCode = connection.getResponseCode();
104.           if (responseCode==200) {
105.               //获取总字节数
106.               int totalSize = connection.getContentLength();
107.               //计算每个线程下载的平均字节数
108.               int avgSize = totalSize/threadCount;
109.               //计算每个线程下载的范围
110.               for(int i=0;i<threadCount;i++){
111.                   final int startIndex = i*avgSize;
112.                   int endIndex = 0;
113.                   if (i==threadCount-1) {
114.                       endIndex = totalSize-1;
115.                   }else {
116.                       endIndex = (i+1)*avgSize-1;
117.                   }
118.                   threadRunning++;
119.                   //开启子线程,实现下载
120.                   new MyDownloadThread(i, startIndex,
121. endIndex,targetPath,sourcePath).start();
122.               }
123.
124.
125.           }else {
126.               System.out.println("返回码为:"+responseCode+" 请求文件长度失败!");
```

```
127.         return;
128.     }
129.
130.     } catch (Exception e) {
131.         e.printStackTrace();
132.     }
133. }
134. /**
135.  * 下载线程
136.  *
137.  * @author wzy 2015-10-26
138.  *
139.  */
140. class MyDownloadThread extends Thread{
141.     private int id;
142.     private int startIndex;
143.     private int endIndex;
144.     private String targetPath;
145.     private String sourcePath;
146.     public MyDownloadThread(int id,int startIndex,
147. int endIndex,String targetPath,String sourcePath){
148.         this.id = id;
149.         this.startIndex = startIndex;
150.         this.endIndex = endIndex;
151.         this.targetPath = targetPath;
152.         this.sourcePath = sourcePath;
153.     }
154.     @Override
155.     public void run() {
156.         try {
157.             URL url = new URL(sourcePath);
158.             HttpURLConnection connection =
159. (HttpURLConnection) url.openConnection();
160.             //设置断点下载参数
161.             connection.setRequestMethod("GET");
162.             //根据线程的 id 创建临时文件，用于记录下载的进度
163.             File file = new File("/mnt/sdcard/"+id+".tmp");
164.             /*
165.              * 该属性设置后,返回的状态码就不再是 200,而是 206
166.              */
167.             int currentIndex = -1;
168.             //读进度
169.             if (file.exists()) {
170.                 BufferedReader reader =
171. new BufferedReader(new FileReader(file));
```

```
172.                String readLine = reader.readLine();
173.                //读取历史进度
174.                currentIndex = Integer.valueOf(readLine);
175.                reader.close();
176.            }else {
177.                currentIndex = startIndex;
178.            }
179.            //设置多线程下载属性
180.            connection.setRequestProperty("Range",
"bytes="+currentIndex+"-"+endIndex);
181.            connection.setConnectTimeout(5000);
182.            connection.setReadTimeout(5000);
183.            connection.connect();
184.            int responseCode = connection.getResponseCode();
185.            if (responseCode==206) {
186.                InputStream inputStream = connection.getInputStream();
187.                //支持随机读写的文件类
188.                RandomAccessFile raf =
189. new RandomAccessFile(targetPath, "rws");
190.                //将文件指针定位到要写的位置
191.                raf.seek(currentIndex);
192.                byte[] buffer = new byte[1024*16];
193.                int len = -1;
194.                int totalDownloaded = 0;
195.                //获取线性布局的子控件(ProgressBar)
196.                ProgressBar pb = (ProgressBar) ll_pbs.getChildAt(id);
197.                //设置对打的进度值
198.                pb.setMax(endIndex-startIndex);
199.                while((len=inputStream.read(buffer))!=-1){
200.                    raf.write(buffer, 0, len);
201.                    totalDownloaded+=len;
202.                    //写进度
203.                    BufferedWriter writer =
204. new BufferedWriter(new FileWriter(file));
205.                    writer.write(currentIndex+totalDownloaded+"");
206.                    writer.close();
207.
208.                    //修改进度条
209.
210. pb.setProgress(currentIndex+totalDownloaded-startIndex);
211.                System.out.println(id+"下载了:"+totalDownloaded+", "+
212. (totalDownloaded+currentIndex-startIndex)*100/(endIndex-startIndex)+"%");
213.                }
214.                raf.close();
```



```
214.         inputStream.close();
215.         connection.disconnect();
216.         //线程执行完了
217.         threadRunning--;
218.         //如果所有的线程都下载完了，则删除所有的临时文件
219.         if (threadRunning==0) {
220.             for(int i=0;i<threadCount;i++){
221.                 File tmpFile = new File("/mnt/sdcard/"+i+".tmp");
222.                 tmpFile.delete();
223.             }
224.         }
225.         }else {
226.             System.out.println("返回的状态码为:"+responseCode+
227. " 下载失败!");
228.         }
229.     } catch (Exception e) {
230.         e.printStackTrace();
231.     }
232. }
233. }
234.
235. }
236. }
237.
```

1.6.4 添加权限

在该工程中不仅用到了网络访问还用到了 sdcard 存储，因此需要添加两个权限。

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

1.7 xUtils 实现多线程下载

1.7.1 xUtils 简介

xUtils 是开源免费的 Android 工具包，代码托管在 github 上。目前 xUtils 主要有四大模块：

- DbUtils 模块
操作数据库的框架。
- ViewUtils 模块
通过注解的方式可以对 UI，资源和事件绑定进行管理。
- HttpUtils 模块
提供了方便的网络访问，断点续传等功能。

- BitmapUtils 模块

提供了强大的图片处理工具。

我们在这里只简单实用 xUtils 工具中的 HttpUtils 工具。xUtils 的下载地址：

<https://github.com/wyoulf/xUtils>

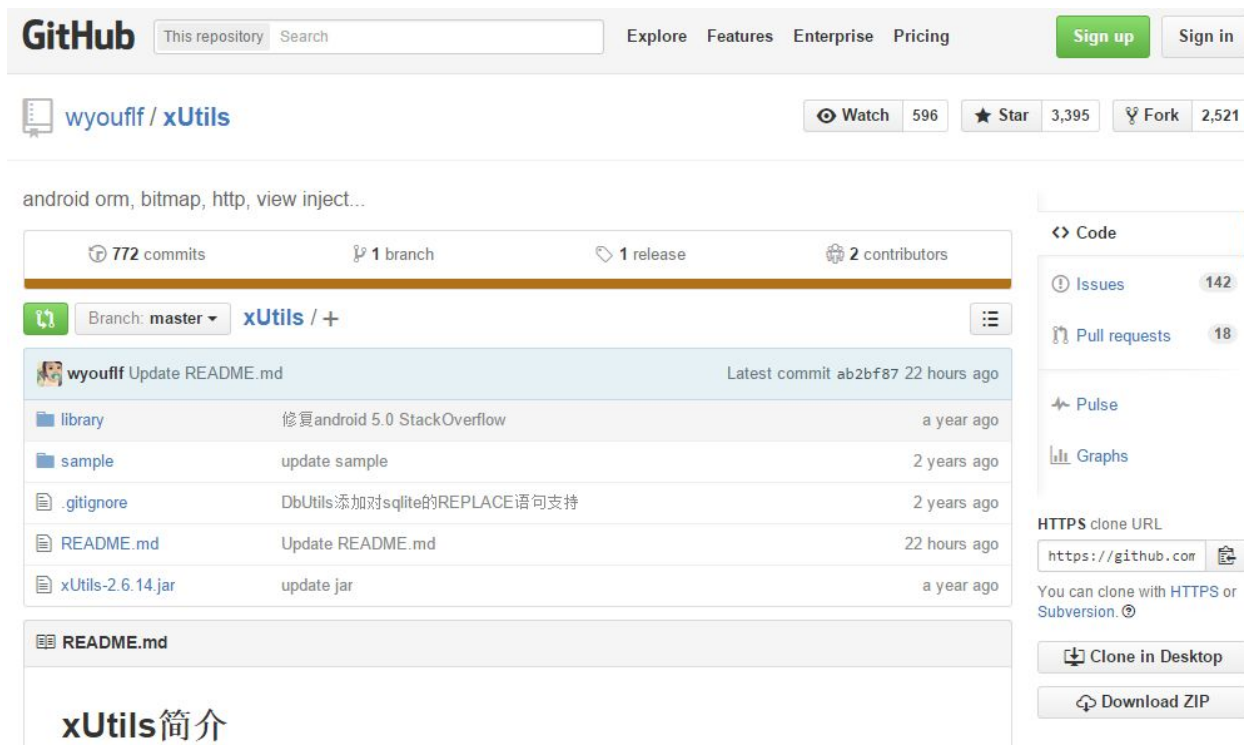


图 1-5 xUtils 在 Github 网页截图

1.7.2 xUtils 之 HttpUtils 的使用

一、下载 xUtils 工具如图 1-5 所示，右下角“Download ZIP”

二、将下载好的 zip 包解压，然后将 src 下的源代码拷贝在自己工程中如图 1-6。

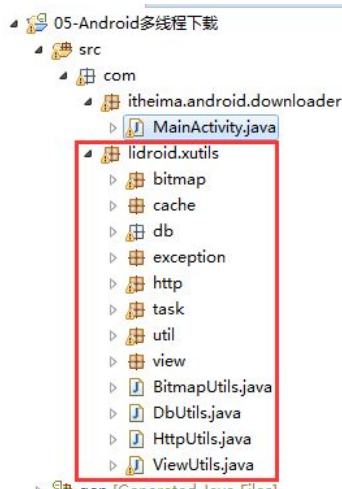


图 1-6 xUtils 拷贝到 src 目录中

三、使用 HttpUtils

HttpUtils 的使用非常简单，因此这里直接给出核心代码。

【文件 1-13】 使用 HttpUtils 完成文件下载

```
1.  /*
2.   * 使用 xUtils 中的 HttpUtils 模块进行下载
3.   */
4.  public void downloadHttpUtils(View view){
5.      HttpUtils httpUtils = new HttpUtils();
6.      String url="http://10.0.2.2:8080/FeiQ.exe";
7.      String target = "/mnt/sdcard/FeiQ2.exe";
8.      /*
9.       * 参数 1: 原文件网络地址
10.      * 参数 2: 本地保存的地址
11.      * 参数 3: 是否支持断点续传,true: 支持, false:不支持
12.      * 参数 4: 回调接口, 该接口中的方法都是在主线程中被调用的,
13.      * 也就是该接口中的方法都可以修改 UI
14.      */
15.      httpUtils.download(url, target, true, new RequestCallBack<File>() {
16.          //当下载任务开始时被调用
17.          @Override
18.          public void onStart() {
19.              Log.d("tag", "onStart"+Thread.currentThread().getName());
20.          }
21.          //每下载一部分就被调用一次, 通过该方法可以知道当前下载进度
22.          /*
23.           * 参数 1: 原文件总字节数
24.           * 参数 2: 当前已经下载好的字节数
25.           * 参数 3: 是否在上传, 对于下载, 该值为 false
26.           */
27.          @Override
28.          public void onLoading(long total, long current, boolean isUploading) {
29.              Log.d("tag", "onLoading"+Thread.currentThread().getName()+"//"+
30. "current"+current+"//"+"total="+total);
31.          }
32.          //下载成功后调用一次
33.          @Override
34.          public void onSuccess(ResponseInfo<File> responseInfo) {
35.              Toast.makeText(MainActivity.this,
36. "下载成功", Toast.LENGTH_SHORT).show();
37.          }
38.          //失败后调用一次
39.          @Override
40.          public void onFailure(HttpException error, String msg) {
41.              Toast.makeText(MainActivity.this,
42. "下载失败: "+msg, Toast.LENGTH_LONG).show();
```

```
43.      }  
44.      });  
45. }
```