# 使用HttpClient框架做GET提交

## 需要写一个web项目，做服务器

## 开启一个子线程做访问网络的操作

## 确定网址

* 数据还得作为参数拼接在网址后面
* 中文需要URL编码，英文和数字可以不用。因为编码后还是不变的

## 创建客户端对象，HttpClient

* 它是接口，所以new它的一个子类DefaultHttpClient

## 创建请求对象，httpGet

* 构造方法的参数就是要访问的网址，把网址传递过去

## 使用HttpCLient对象发送get请求，并且得到服务器的响应对象HttpResponse

* HttpResponse hr = HttpClient.execute(HttpRequest)

## 使用响应对象，得到响应数据中的状态行对象StatusLine

* StatusLine sl = HttpResponse.getStatusLine();

## 通过状态行对象拿到响应码，判断是否请求成功

* Int code = StatusLine.getStatusCode()

## 通过响应对象，拿到响应实体的对象HttpEntity

* HttpEntity he = HttpResponse.getEntity()

## 通过实体对象，拿到服务器响应的内容，就是服务器返回的输入流

* InputStream is = HttpEntity.getContent();

## 拿到输入流，进行处理，得到想要的数据

## 访问网络的权限需要添加

# 使用HttpClient框架做POST提交

## 确定网址

## 创建客户端H ttpClient对象

## 创建POST请求对象HttpPost，构造方法的参数就是请求的网址

## Post的请求数据要封装在实体对象HttpEntity中

* 使用HttpEntity的子类UrlEncodedFormEntity
* 构造方法需要两个参数：
  + 数据：把数据封装在List中，并且list里只能装NameVluePair
  + 编码

## 封装数据到List中，list在放进实体对象

* 这里数据不需要进行URL编码，因为BasicNameValuePair会帮我们完成
* 数据类型必须是NameVluePair，所以我们使用它的一个子类BasicNameValuePair
* 构造参数需要两个：
  + Key: 参数名
  + Value： 参数值
* 构建出多个BasicNameValuePair，放入List中
* 再把List放入实体对象

## 把实体对象设置进Post请求对象中

* HttpPost#setEntity(HttpEntity)

## 使用客户端对象发送post请求，并且得到响应对象HttpResponse

* HttpResponse hr = HttpClient.execute(HttpRequest)

## 使用响应对象得到状态行对象StatusLine

## 使用状态行对象得到响应码，做判断

## 使用响应对象得到服务器响应过来的实体对象

## 通过实体对象的到输入流，就是服务器发送过来的数据

## 操作输入流，得到数据

# 使用异步HttpClient框架做GET提交，android-async-http-master

* 这是第三方的框架，需要导入源码到项目
* android-async-http-master\library\src\main\java\com，这个目录里的都导入

## 创建异步HttpClient对象

|  |
| --- |
| //创建异步httpclient  AsyncHttpClient ahc = new AsyncHttpClient(); |

## 发送get请求提交数据

* AsyncHttpClient#get(url, responseHandler)
* 需要两个参数：
  + 请求的网址，因为是get请求，所以需要把参数拼接在网址中，并且进行URL编码
  + 响应结果处理器

## 响应结果处理器

* 自己创建一个，继承AsyncHttpResponseHandler
* 实现两个方法
  + onSuccess：请求成功调用（就是状态码=200），会传过来状态码和服务器返回的数据。数据以字节数组方式传过来
  + onFailure：请求失败调用（状态码 != 200），传递状态码。和错误页面的源文件，以字节数组形式传递

# 使用异步HttpClient框架做POST提交

## 创建异步HttpClient对象

## 发送post请求提交数据

* ahc.post(url, params, ResponseHandler);
* url：请求的网址
* params：要提交的数据，使用RequestParams
* responseHandler：响应结果处理器

## 要提交的数据使用RequestParams封装

* 数据不用URL编码，RequestParams会帮我们完成

|  |
| --- |
| RequestParams params = new RequestParams();  params.add("name", “zs”);  params.add("pass", “123”); |

# 多线程下载

* 原理：服务器CPU分配给每条线程的时间片相同，服务器带宽平均分配给每条线程，所以客户端开启的线程越多，就能抢占到更多的服务器资源
* 因为都是java代码，所以创建一个java项目就可以了，比较方便

## 实现过程

1. 要从服务器下载一个文件，这个文件的大小是10KB
2. 客户端的下载线程有三条，就要计算出每个线程要负责下载这个文件的哪一部分
3. 为三个线程分配线程id。0、 1、 2
4. 计算：
   1. Length：文件总字节的长度
   2. Size（每个线程下载数）：文件总字节 / 线程数， 取整数。因为会有余数，所以最后一个线程负责把剩余的字节也下载了
   3. 开始位置：线程ID \* 每个线程下载数， 这个位置是文件字节的下标
   4. 结束位置： (线程ID + 1) \* 每个线程下载数 – 1， 这个位置是文件字节的下标
   5. 最后一个线程的结束位置：文件最后一个字节的下标， Length – 1
5. 设置每个线程下载数据的区间（设置本次http请求所请求的数据的区间）：HttpURLConnection# setRequestProperty(“Range”, “bytes=” + 开始位置 + “-” + 结束位置)
6. 这时服务器返回的响应码就是206了， 206表示部分内容
7. 创建一个下载后存放的文件
   1. 存放文件需要先占好下载文件所需要的空间，所以创建文件时，就应该先为文件定义好大小
   2. 所以使用这个类：RandomAccessFile，传入路径/File，传入模式：rwd，可读可写，并且把写入该文件的数据实时同步到底层存储设备。而不使用缓冲来存储了。这样保证了文件的完整性，不会因为意外导致文件不完整
   3. 因为有三个线程在下载，所以每个线程不数据写入文件的开始位置也不相同。
   4. 是哪个线程在下载，那么这个线程从服务器获取数据的开始位置是多少，那么写入文件的开始位置就是多少
   5. 设置文件写入的位置：RandomAccessFile# seek(开始位置)
   6. 设置文件的大小：RandomAccessFile# setLength（文件总大小）
   7. 用完之后要关闭随机读写的这个文件

# 断点续传

* 原理：记录每条线程每次下载到的位置，下次继续从该位置开始下载即可

## 实现过程

1. While循环读取流中数据的时候，用一个变量，把每次读取到的字节数，加到这个变量中。并且把本次数据写入到文件后，同步把此时记录到的字节数写入临时文件。这样，当循环结束后，一个线程也就完成了下载任务，那个变量就记录了这个线程下载到的总字节数。并且记录在了临时文件中。
2. 有多少个线程，就生成几个临时文件，记录每条线程的下载到的字节数
3. 如果while循环中途停止了，也没关系，因为那个变量记录了上次while循环时，读取到的总字节数是多少，并且写入到了临时文件中
4. 在再次开始下载的时候，拿到对应线程生成的临时文件，取出文件中的数字。把它加上本次下载的开始位置，就得到了新的开始位置，也就是上次停止下载的位置。
5. 在这次下载时，使用新的下载位置，开始下载
6. 用一个变量记录每个线程是否下载完毕，如果一个线程的while循环完毕，就下载完毕。变量加1
7. 当这个变量等于总线程数的时候，就是文件下载完毕的时候，就把那几个记录下载位置的临时文件删除
8. 删除要注意线程同步的问题，只需要一条线程去删除就行了

# 移植到Android项目中

# 显示进度条

## 使用ProgressBar组件

* 显示进度条也是刷新ui，但是它可以在子线程中进行。只有在子线程中，才能显示进度条

|  |
| --- |
| <ProgressBar  android:id=*"@+id/pb"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"*  style=*"@android:style/Widget.ProgressBar.Horizontal"*  android:max=*"200"*  android:progress=*"100"*/> |
|  |

## 实现过程

1. 得到ProgressBar组件，设置它的最大值为：下载文件的总字节大小，ProgressBar#setMax(leng);
2. 设置当前值：因为有三条线程。所以定义个全局变量（记录下载进度），在while中，把每次读取到的字节数加到变量中，把变量设置为进度条的当前值

ProgressBar#setProgress(leng)

1. 需要支持断点续传。在取出上次一下载的总字节（记录在临时文件中的），把它累加在全局变量中

# 显示文本进度，百分比

* 需要在主线程中操作
* 计算公式：当前下载进度 \* 100/总字节数：(long)progressBar.getProgress() \* 100 / progressBar.getMax()
* 需要转换为long类型，因为计算的数字如果过大，int会显示不了，导致百分比显示不正常
* 在while循环设置了进度条显示的进度后，通知主线程显示文本进度
* 断点续传，设置了上次下载的进度条后，通知主线程显示文本进度
* 在下载完成之后，会出现文本进度停在99%的情况。实际上文件已经下载完成了。强制显示成100%即可

# xUtils下载

* 可以导入jar包：导到libs
* 也可以导入源码
* 需要写入sd卡权限，和网络权限