# 下载图片

* 用的是http协议
* 服务器的图片是以流的形式返回给浏览器的
* 请求超时：请求的网址，超过指定的时间没有连接上
* 读取超市：请求的网址连接上了，但是在指定的时间内没有返回数据到我们的客户端

## 4.0以前才能生效的做法

* 是因为请求网络连接是个耗时操作，写在了主线程中。4.0以前是可以这么写的，但是4.0以后就不可以了

### 确定请求的网址，注意ip不能写成localhost

|  |
| --- |
| String path = "http://192.168.1.107:8080/Android/wo.jpg"; |

### 通过网址，构造成一个java.net.URL对象

|  |
| --- |
| URL url = **new** URL(path); |

### 通过url，得到客户端与服务器的连接对象，此时还没有建立连接

|  |
| --- |
| HttpURLConnection conn = (HttpURLConnection) url.openConnection(); |

### 给连接对象设置初始化参数

|  |
| --- |
| conn.setRequestMethod("GET");//设置请求方式，必须是大写  conn.setConnectTimeout(5000);//设置连接超时时间，5秒  conn.setReadTimeout(5000);//设置读取超时时间，5秒 |

### 建立连接，这是一个阻塞式的代码

|  |
| --- |
| conn.connect(); |

### 获取响应/状态码，可以根据服务器返回的状态码，进行不同的操作

|  |
| --- |
| **if**(conn.getResponseCode() != 200) { //如果请求码不是200，就是请求没有成功  Toast.*makeText*(**this**, "请求失败！", 0).show();  } |

### 如果响应/状态码是200

|  |
| --- |
| //拿到服务器返回的输入流，流里的数据就是客户端请求的数据  InputStream is = conn.getInputStream();  //读取出理你的数据，并构造成位图对象。位图其实就是图片  Bitmap bm = BitmapFactory.*decodeStream*(is); |

### 把位图对象显示到ImageView中

|  |
| --- |
| //把位图对象显示到ImageView中，其实就是把图片显示在ImageView中  ImageView iv = (ImageView) findViewById(R.id.*iv*);  iv.setImageBitmap(bm); |

### 添加访问网络权限

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

## 主线程阻塞

* 在Android中，主线程被阻塞会导致应用不能刷新ui界面，不能响应用户操作，用户体验将非常差
* 主线程阻塞时间过长，系统会抛出ANR异常
* ANR：Application Not Response；应用无响应
* 任何耗时操作都不可以写在主线程
* 因为网络交互属于耗时操作，如果网速很慢，代码会阻塞，所以网络交互的代码不能运行在主线程

## 只有主线程能刷新ui

* 刷新ui的代码只能运行在主线程，运行在子线程是没有任何效果的
* 刷新UI有：弹吐司，界面显示图片… 凡是界面发生更改就是刷新UI，哪怕只在增加一行文字
* 如果需要在子线程中刷新ui，使用消息队列机制

## 消息队列

* MessageQueue：消息队列对象，主线程创建时就会被创建
* Looper：消息轮询器，主线程创建时就会被创建
* Handler：消息处理器，自己创建，最好创建静态的
* Message：消息对象，自己创建，不建议new，用Handler#obtainMessage()来创建
  + Obj属性：是一个object类型的属性。用来携带消息，如果有多条消息，使用集合或数组存放。然后存放在obj属性中即可
  + What属性：是一个int类型的属性。可以用来判断消息的类型。例如：传个1表示成功，0表示失败。那么就可以在Handerl#handleMessage(Message mas)方法中，对what属性进行判断
  + 如果消息不用携带数据，可以直接使用Hnadler#sendEmptyMessage(what)，来直接发送消息
* 轮询器的作用，就是不停的检测消息队列中是否有消息，消息队列一旦有消息，轮询器会把消息对象传给消息处理器，

处理器会调用handleMessage方法来处理这条消息，handleMessage方法运行在主线程中，所以可以刷新ui

* 总结：只要消息队列有消息，handleMessage方法就会调用
* 子线程如果需要刷新ui，只需要往消息队列中发一条消息，触发handleMessage方法即可
* 子线程使用处理器对象的sendMessage方法发送消息

## 对下载图片的代码进行更改

* 主线程不能有耗时操作，那就在开一个线程，来请求网络连接
* 子线程不能刷新ui，那么就使用消息队列。把弹吐司，还有要显示的图片，封装进Message的obj属性中，

使用Message的what属性对消息的类型作区分。把消息对象使用Handler传递到消息队列中

* 添加访问网络权限

## 再次更改，加入缓存图片的功能

* + 如果第一次访问哪个图片，就请求服务器获取图片。然后存到内部存储空间中。在显示到界面
  + 如果已经存在要访问的图片，直接在内部存储空间中取出，显示在界面上
  + 添加访问网络的权限

## 使用第三方的API，来下载图片

* android-smart-image-view-master.zip
* 里面是源码，直接复制到项目中即可
* 里面都是高度封装好的，原理跟上面我们写的一样。
* 不能使用ImageIView了，而是使用他自定义的一个控件：com.loopj.android.image.SmartImageView
* 自定义控件要写全名
* 缓存的路径也是在内部存储空间中
* 要自己添加访问网络的权限

# 下载文本文件

* 跟下载图片是一样的，只不过处理的数据类型不同
* 注意获取联网权限
* Android默认使用的UTF-8
* 如果有乱码，就更改服务器里原文件的编码
* 文本过多，使用ScrollView控件，使屏幕可滑动

# 新闻客户端

* 把新闻信息放在xml文件中，然后存在服务器端
* Xml文件有六个节点：
  + <newslist>：根节点
  + <news>：新闻节点，客户端读取到这个节点后，就把这个节点封装成javabean
  + <title>：新闻标题
  + <detail>：新闻正文
  + <comment>：评论数
  + <image>：新闻图片的网络地址

## 页面布局

* 需要一个ListView，显示新闻条目
* 需要一个布局文件，填充成view，作为条目显示在ListView中
* android:singleLine="true"：表示只显示单行文本，如果超出显示范围，超出的部分以“…”显示
* android:lines="2"：表示显示两行文本，如果超出显示范围，超出的部分以“…”显示

## 从服务器端获取news.xml并解析封装成javabean集合

## 把集合中的数据，显示在Listview中

* 其中，图片的显示可以直接使用第三方的图片查看器来操作。那么布局文件中的显示图片的组件，就要使用它自定义的那个了
* ListView的缓存优化，和查询组件的优化
* 因为获取服务器数据实在子线程中执行，所以要保证javabean集合已经被赋值完毕。把为listView设置Adpter的操作，放在子线程的javabean集合赋值完毕之后即可。
* 因为为ListView设置adpter也属于刷新UI，所以需要使用消息队列来完成

# 使用GET方式，向服务器提交数据

## 需要写一个web项目，来做服务器端

* 校验用户名密码是否正确，向客户端发送成功信息或失败信息

## 发送请求

* 在请求的网址后面，添加请求参数即可，其他操作跟读取客户端数据是一样的
* 中文的需要URL编码，而英文和数字可以不用，因为编码后还是不变的

## 获取客户端发送过来的信息，使用吐司弹出

# 使用POST方式，向服务器提交数据

* 基本跟get方式相同，只需修改几个地方就可以了

## 请求方式设置为POST，增加两个请求头

* 使用连接对象的setRequestProperty(name, value)
* Content-Type: application/x-www-form-urlencoded
* Content-Length: 请求体长度

## 不用再地址中带参数。而是使用输出流来带参数

* 连接对象#getOutputStream()：拿到输出流
* 使用输出流之前需要打开输出流：连接对象#setDoOutput(true)
* 这时还没有向客户端提交数据。而是在发送请求时，才会把数据提交