Android 子线程访问网络

之所以 Google 在 Android4.0 之后，禁止主线程访问网络，是为了更好的用户体验。也就是主线程是为了界面的显示。如果主线程访问网络，就会造成“卡顿”。也就是对于网络状况的不可预见性，很有可能在网络访问的时候造成阻塞，那么这样一来我们的主线程 UI 线程就会出现假死的现象，产生很不好的用户体验。所以，默认的情况下如果直接在主线程中访问就报出了这个异常，名字是 NetworkOnMainThreadException。主线程操作 5s 的相应时间，就会关闭该线程，所以将耗时很大的操作放在子线程。

解决该问题的办法

1. 独立线程
2. 异步线程 AsyncTask
3. StrictMode 修改默认的策略

（1） 独立线程的办法

启动一个新线程的代码：

```
new Thread() {
    @Override
    public void run() {
    Dosomething();
    handler.sendEmptyMessage(0);
    }
    }.start();
```

此处我们重写了线程类的 run 方法，执行 Dosomething. 在里面还有个 handler 对象，这又涉及到了跨线程修改 UI 元素内容的问题。在 java 中是不允许跨线程修改 UI 元素的，如我们在新启动的线程中想去修改 UI 主线程中 TextView 的文本时，会报错误的。如果想做这样的操作，我们就得借助 Handler 这个类来实现。 关于这个 handler 类的用法，我们单独的再来写一篇博客进行介绍。

（2） 异步调用的方法 AsyncTask

这里关于 AsyncTask 介绍的文章不错，详细情况看作者的介绍吧

http://www.cnblogs.com/dawei/archive/2011/04/18/2019903.html#2824345

接下来也将会有一篇博客专门介绍 关于更新主线程 UI 线程的所有办法

（3） StrictMode 修改默认的策略

在我们的 Activity 类的 onCreate 方法中，设置如下规则：

```
StrictMode.ThreadPolicy policy=new StrictMode.ThreadPolicy.Builder().p
ermitAll().build();
StrictMode.setThreadPolicy(policy);
```