在第一节中，我们使用Handler、Thread/Runnable 、URL、HttpURLConnection等等来进行异步下载网络图片。然后第二节中换了AsyncTask方式。那么这一节我们再应用其他方式，这个方式与第一节有点雷同，感觉走了一圈回到原地，只是风景更加美丽。人生很多时候也是如此，从NULL中来再回到NULL中。

那么它是谁呢？java.util.concurrent

java.util.concurrent 是在并发编程中很常用的实用工具类。

ExecutorService类：具有服务生命周期的Executors。

Executors 类：执行器，将为你管理Thread 对象。

我们知道这些是用来处理并发任务的，当然我们Demo只是请求一张图片而已，并不能体现并发，但是假设我们有一个ListView，里面每一项都需要一张网络图片显示呢？那么并发性就可以体现出来了:多个线程并发从网络下载图片。当然这个版本不会使用listView显示多个项图片，以后做个版本吧！研究下。

思路是这样的：

1：动态的创建N个线程，防在线程池中。

2：系统从线程池中取出一个线程投入执行，线程池中若没有线程可用，其他任务只有先等待了，直到有新线程释放，才调用。

如下有几个方法可以动态的指定多少个线程。

newFixedThreadPool(int nThreads) 指定线程个数

newCachedThreadPool() 系统为每个任务都建立一个线程

下面可以实现并发下载数据

case R.id.btnThress:

progress.setVisibility(View.VISIBLE);

final Handler newhandler=new Handler();

executorService.submit(new Runnable(){

@Override

public void run() {

try {

URL newurl = new URL(params);

HttpURLConnection conn = (HttpURLConnection)newurl.openConnection();

conn.setDoInput(true);

conn.connect();

InputStream inputStream=conn.getInputStream();

bitmap = BitmapFactory.decodeStream(inputStream);

newhandler.post(new Runnable(){

@Override

public void run() {

ImageView view=(ImageView)frameLayout.findViewById(R.id.image);

view.setImageBitmap(bitmap);

}

});

} catch (MalformedURLException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

});

break;