在上一节中，我们使用Handler、Thread/Runnable 、URL、HttpURLConnection等等来进行异步下载网络图片。

但是采用这种方式有一些缺点，如下：

线程的开销较大，如果每个任务都要创建一个线程，那么程序的效率要低很多。

线程无法管理，匿名线程创建并启动后就不受程序的控制了，如果有很多个请求发送，那么就会启动非常多的线程，系统将不堪重负。

另外，前面已经看到，在新线程中更新UI还必须要引入handler，这让代码看上去非常臃肿。

那么有没有比较更好好的实现方式呢?这个可以有！它就是AsyncTask

AsyncTask的特点是任务在主UI线程之外运行，而回调方法是在主UI线程中，这就有效地避免了使用Handler带来的麻烦。

AsyncTask定义了三种泛型类型 Params，Progress和Result。

•Params 启动任务执行的输入参数。

•Progress 后台任务执行的百分比。

•Result 后台执行任务返回的结果。

当然，使用它还必须覆盖它的一些抽象方法方法

doInBackground(Params...) 执行任务

onPostExecute(Result) 返回任务执行的结果，通常更新UI

onProgressUpdate (Progress... values) 进度更新

注意：红色的是必须实现的。

第一步：设计好UI，与上节一样

第二步：也与上节一样。

第三步：主要是实例化AsyncTask，并执行execute(Params)

我们必须继承AsyncTask，并覆盖它的一些方法，我们这里主要是要获取网络图片，并保存为Bitmap，以便UI根据Bitmap来更新的。

那么需要为AsyncTask设置返回的类型参数为String,Integer,Bitmap 类定义如下：

public class MyASyncTask extends AsyncTask<String, Integer, Bitmap> {

在doInBackground(Params...) 方法中 ，接受String ....params,返回我们需要的Bitmap.当然我们这里是获取图片Bitmap所以要返回Bitmap

如果你返回的需要是String或者其他复杂类型时候，需要修改类的定义参数类型为你需要返回的类型，当然接受参数也是根据你的请求需要改变。

@Override

protected Bitmap doInBackground(String... params) {

Bitmap bitmap=null;

try {

URL url = new URL(params[0]);

HttpURLConnection con=(HttpURLConnection) url.openConnection();

con.setDoInput(true);

con.connect();

InputStream inputStream=con.getInputStream();

bitmap=BitmapFactory.decodeStream(inputStream);

inputStream.close();

}

catch (MalformedURLException e) {

e.printStackTrace();

}catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return bitmap;

}

在 onPostExecute(Result) 中是请求获得结果后更新UI部分。你会看到他的参数就是我们类中的类型参数。代码如下：

//执行获得图片数据后，更新UI:显示图片，隐藏进度条

@Override

protected void onPostExecute(Bitmap Result){

ImageView imgView=(ImageView)this.viewGroup.getChildAt(0);

imgView.setImageBitmap(Result);

ProgressBar bar=(ProgressBar)this.viewGroup.getChildAt(1);

bar.setVisibility(View.GONE);

}

然后怎么用呢？在UI线程中执行吧：

﻿MyASyncTask yncTask=new MyASyncTask(this,frameLayout);

yncTask.execute(params);