

3.7 AsyncTask异步任务

分类 [Android 基础入门教程](#)

本节引言：

本节给大家带来的是Android给我们提供的一个轻量级的用于处理异步任务的类:AsyncTask，我们一般是 继承AsyncTask，然后在类中实现异步操作，然后将异步执行的进度，反馈给UI主线程~ 好吧，可能有些概念大家不懂，觉得还是有必要讲解下多线程的概念，那就先解释下一些概念性的东西吧！

1.相关概念

1) 什么是多线程：

答：先要了解这几个名称：应用程序，进程，线程，多线程！！

应用程序(Application)：为了完成特定任务，用某种语言编写的一组指令集合(一组静态代码)

进程(Process) **运行中的程序**，系统调度与资源分配的一个**独立单位**，操作系统会为每个进程分配 一段内存空间，程序的依次动态执行，经理代码加载 -> 执行 -> 执行完毕的完整过程！

线程(Thread)：比进程更小的执行单元，每个进程可能有多条线程，**线程需要放在一个进程中才能执行！** 线程是由程序负责管理的！！！而进程则是由系统进行调度的！！！！

多线程概念(Multithreading)：并行地执行多条指令，将CPU的**时间片**按照调度算法，分配给各个线程，实际上是**分时执行的**，只是这个切换的时间很短，用户感觉是同时而已！

举个简单的例子：你挂着QQ，突然想去听歌，你需要把QQ关掉，然后再去启动XX播放器吗？答案是否定的，我们直接打开播放器 放歌就好，QQ还在运行着，是吧！这就是简单的多线程~在实际开发中，也有这样的例子，比如应用正在运行，发现新版本了，想后台更新，这个时候一般我们会开辟出条后台线程，用于下载新版本的apk，但是这个时候 我们还可以使用应用的其他功能！这就是多线程的使用例子~

2) 同步与异步的概念：

答: **同步**：当我们执行某个功能时，在没有得到结果之前，这个调用就不能返回！简单点就是说必须 等前一件事做完才能做下一件事；举个

Android 基础入门教程(Q群号：153836263)

- 1.0 Android基础入门教程
 - 1.0.1 2015年最新Android基...
 - 1.1 背景相关与系统架构分析
 - 1.2 开发环境搭建
 - 1.2.1 使用Eclipse + ADT + S...
 - 1.2.2 使用Android Studio开...
 - 1.3 SDK更新不了问题解决
 - 1.4 Genymotion模拟器安装
 - 1.5.1 Git使用教程之本地仓...
 - 1.5.2 Git之使用GitHub搭建...
 - 1.6 .9(九妹)图片怎么玩
 - 1.7 界面原型设计
 - 1.8 工程相关解析(各种文件...
 - 1.9 Android程序签名打包
 - 1.11 反编译APK获取代码&...
- 2.1 View与ViewGroup的概念
 - 2.2.1 LinearLayout(线性布局)
 - 2.2.2 RelativeLayout(相对布...
 - 2.2.3 TableLayout(表格布局)
 - 2.2.4 FrameLayout(帧布局)
 - 2.2.5 GridLayout(网格布局)
 - 2.2.6 AbsoluteLayout(绝对...
- 2.3.1 TextView(文本框)详解
- 2.3.2 EditText(输入框)详解
- 2.3.3 Button(按钮)与ImageB...
- 2.3.4 ImageView(图像视图)
- 2.3.5.RadioButton(单选按钮...
- 2.3.6 开关按钮ToggleButton...
- 2.3.7 ProgressBar(进度条)
- 2.3.8 SeekBar(拖动条)
- 2.3.9 RatingBar(星级评分条)
- 2.4.1 ScrollView(滚动条)



反馈

简单的例子：比如你啪啪啪，为了避免弄出人命，肯定要戴好套套，然后再啪啪啪是吧~套套戴好 -> 然后啪啪啪，比如你没套套，那啪啪啪的操作就要等待了，直到你把套套买回来带上，这个时候就可以开始啪啪啪了~一个形象地例子，♫(^▽^*) **异步**：和同步则是相对的，当我们执行某个功能后，我们并不需要立即得到结果，我们额可以正常地做其他操作，这个功能可以在完成后通知或者回调来告诉我们；还是上面那个后台下载的例子，后台下载，我们执行下载功能后，我们就无需去关心它的下载过程，当下载完毕后通知我们就可以了~

3) Android为很么要引入异步任务

答：因为Android程序刚启动时，会同时启动一个对应的主线程(Main Thread)，这个主线程主要负责处理与UI相关的事件！有时我们也把他称作UI线程！而在Android App时我们必须遵守这个单线程模型的规则：**Android UI操作并不是线程安全的并且这些操作都需要在UI线程中执行！**假如我们在非UI线程中，比如在主线程中new Thread()另外开辟一个线程，然后直接在里面修改UI控件的值；此时会抛出下述异常：

android.view.ViewRoot\$CalledFromWrongThreadException: Only the original thread that created a view hierarchy can touch its views 另外，还有一点，如果我们把耗时的操作都放在UI线程中的话，如果UI线程超过5s没有响应用于请求，那么这个时候会引发ANR(Application Not Responding)异常，就是应用无响应~最后还有一点就是：Android 4.0后禁止在UI线程中执行网络操作~不然会报: **android.os.NetworkOnMainThreadException**

以上的种种原因都说明了Android引入异步任务的意义，当然实现异步也可以不用到我们本节讲解的AsyncTask，我们可以自己开辟一个线程，完成相关操作后，通过下述两种方法进行UI更新：

1. 前面我们学的Handler，我们在Handler里写好UI更新，然后通过sendMessage()等方法通知UI更新，另外别忘了Handler写在线程和子线程中的区别哦~
2. 利用Activity.runOnUiThread(Runnable)把更新ui的代码创建在Runnable中，更新UI时，把Runnable对象传进来即可~

2.AsyncTask全解析：

1) 为什么要用AsyncTask？

答:我们可以用上述两种方法来完成我们的异步操作，加入要我们写的异步操作比较多，或者较为繁琐，难道我们new Thread()然后用上述方法通知UI更新么？程序员都是比较喜欢偷懒的，既然官方给我们提供了AsyncTask这个封装好的轻量级异步类，为什么不用呢？我们通过几十行的代码就可以完成我们的异步操作，而且进度可控；相比起Handler，AsyncTask显得更加简单，快捷~当然，这适合简单的异步操作，另外，实际异步用的最多的地

2.4.2 Date & Time组件(上)

2.4.3 Date & Time组件(下)

2.4.4 Adapter基础讲解

2.4.5 ListView简单实用

2.4.6 BaseAdapter优化

2.4.7 ListView的焦点问题

2.4.8 ListView之checkbox错...

2.4.9 ListView的数据更新问题

2.5.0 构建一个可复用的自定...

2.5.1 ListView Item多布局的...

2.5.2 GridView(网格视图)的...

2.5.3 Spinner(列表选项框)...

2.5.4 AutoCompleteTextVie...

2.5.5 ExpandableListView(...

2.5.6 ViewPager(翻转视图)...

2.5.7 Toast(吐司)的基本使用

2.5.8 Notification(状态栏通...

2.5.9 AlertDialog(对话框)详解

2.6.0 其他几种常用对话框基...

2.6.1 PopupWindow(悬浮框...

2.6.2 菜单(Menu)

2.6.3 ViewPager的简单使用

2.6.4 DrawerLayout(官方侧...

3.1.1 基于监听的事件处理机制

3.2 基于回调的事件处理机制

3.3 Handler消息传递机制浅析

3.4 TouchListener PK OnTo...

3.5 监听EditText的内容变化

3.6 响应系统设置的事件(Co...

3.7 AsyncTask异步任务

3.8 Gestures(手势)

4.1.1 Activity初学乍练

4.1.2 Activity初窥门径

4.1.3 Activity登堂入室

4.2.1 Service初涉

4.2.2 Service进阶

4.2.3 Service精通

4.3.1 BroadcastReceiver牛...

4.3.2 BroadcastReceiver庖...

4.4.1 ContentProvider初探

方就是网络操作，图片加载，数据传输等，AsyncTask 暂时可以满足初学者的需求，谢谢小应用，但是到了公司真正做项目以后，我们更多的使用第三方的框架，比如Volley,OkHttp,android-async-http,XUtils等很多，后面进阶教程我们会选1-2个框架进行学习，当然你可以自己找资料学习学习，但是掌握AsyncTask还是很有必要的！

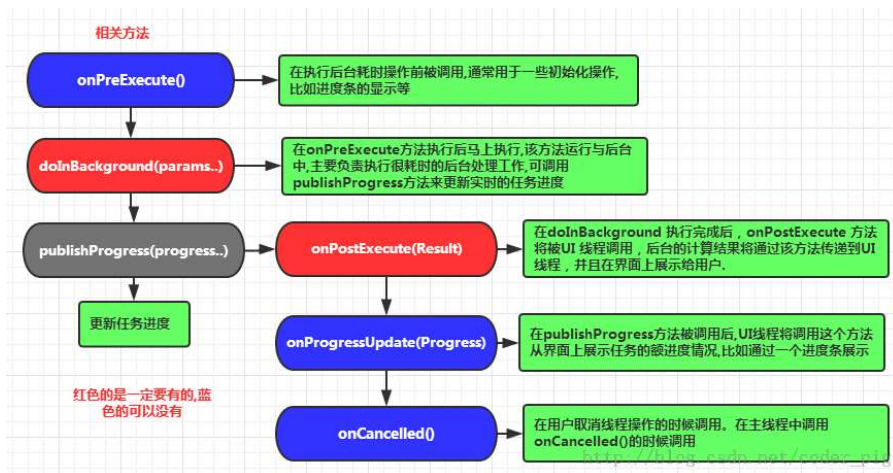
2) AsyncTask的基本结构：

AsyncTask是一个抽象类，一般我们都会定义一个类继承AsyncTask然后重写相关方法~ 官方API:[AsyncTask](#)

构建AsyncTask子类的参数：



相关方法与执行流程：



注意事项：



4.4.2 ContentProvider再探...

4.5.1 Intent的基本使用

4.5.2 Intent之复杂数据的传递

5.1 Fragment基本概念

5.2.1 Fragment实例精讲一...

5.2.2 Fragment实例精讲一...

5.2.3 Fragment实例精讲一...

5.2.4 Fragment实例精讲一...

5.2.5 Fragment实例精讲一...

6.1 数据存储与访问之——文...

6.2 数据存储与访问之——S...

6.3.1 数据存储与访问之——...

6.3.2 数据存储与访问之——...

7.1.1 Android网络编程要学...

7.1.2 Android Http请求头与...

7.1.3 Android HTTP请求方...

7.1.4 Android HTTP请求方...

7.2.1 Android XML数据解析

7.2.2 Android JSON数据解析

7.3.1 Android 文件上传

7.3.2 Android 文件下载 (1)

7.3.3 Android 文件下载 (2)

7.4 Android 调用 Webservice

7.5.1 WebView(网页视图)基...

7.5.2 WebView和JavaScip...

7.5.3 Android 4.4后WebVie...

7.5.4 WebView文件下载

7.5.5 WebView缓存问题

7.5.6 WebView处理网页返...

7.6.1 Socket学习网络基础准备

7.6.2 基于TCP协议的Socket...

7.6.3 基于TCP协议的Socket...

7.6.4 基于UDP协议的Socke...

8.1.1 Android中的13种Draw...

8.1.2 Android中的13种Draw...

8.1.3 Android中的13种Draw...

8.2.1 Bitmap(位图)全解析 P...

8.2.2 Bitmap引起的OOM问题

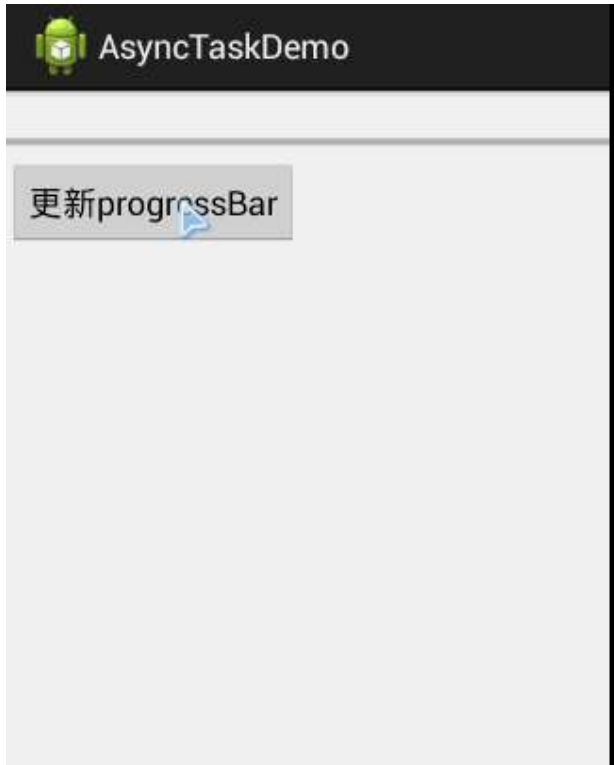
8.3.1 三个绘图工具类详解

8.3.2 绘图类实战示例

3.AsyncTask使用示例：

因为我们还没学到Android网络那块，这里照顾下各位初学者，这里用延时 线程来模拟文件下载的过程~后面讲到网络那里再给大家写几个例子~

实现效果图：



布局文件:activity.xml：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res
/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MyActivity">
    <TextView
        android:id="@+id/txttitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <!--设置一个进度条,并且设置为水平方向-->
    <ProgressBar
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/pgbar"
        style="?android:attr/progressBarStyleHorizontal"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/btnupdate"
        android:text="更新progressBar"/>
```

- 8.3.3 Paint API之—— Mask...
- 8.3.4 Paint API之—— Xferm...
- 8.3.5 Paint API之—— Xferm...
- 8.3.6 Paint API之—— Xferm...
- 8.3.7 Paint API之—— Xferm...
- 8.3.8 Paint API之—— Xferm...
- 8.3.9 Paint API之—— Color...
- 8.3.10 Paint API之—— Colo...
- 8.3.11 Paint API之—— Colo...
- 8.3.12 Paint API之—— Path...
- 8.3.13 Paint API之—— Sha...
- 8.3.14 Paint几个枚举/常量值...
- 8.3.15 Paint API之——Type...
- 8.3.16 Canvas API详解(Part 1)
- 8.3.17 Canvas API详解(Part...
- 8.3.18 Canvas API详解(Part...
- 8.4.1 Android动画合集之帧...
- 8.4.2 Android动画合集之补...
- 8.4.3 Android动画合集之属...
- 8.4.4 Android动画合集之属...
- 9.1 使用SoundPool播放音...
- 9.2 MediaPlayer播放音频与...
- 9.3 使用Camera拍照
- 9.4 使用MediaRecord录音
- 10.1 TelephonyManager(电...
- 10.2 SmsManager(短信管理...
- 10.3 AudioManager(音频管...
- 10.4 Vibrator(振动器)
- 10.5 AlarmManager(闹钟服务)
- 10.6 PowerManager(电源服...
- 10.7 WindowManager(窗口...
- 10.8 LayoutInflater(布局服务)
- 10.9 WallpaperManager(壁...
- 10.10 传感器专题(1)——相...
- 10.11 传感器专题(2)——方...
- 10.12 传感器专题(3)——加...
- 10.12 传感器专题(4)——其...
- 10.14 Android GPS初涉
- 11.0 《2015最新Android基...

```
</LinearLayout>
```

定义一个延时操作，用于模拟下载：

```
public class DelayOperator {
    //延时操作,用来模拟下载
    public void delay()
    {
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

自定义AsyncTask:

```
public class MyAsyncTask extends AsyncTask<Integer, Integer, String>
{
    private TextView txt;
    private ProgressBar pgbar;

    public MyAsyncTask(TextView txt, ProgressBar pgbar)
    {
        super();
        this.txt = txt;
        this.pgbar = pgbar;
    }

    //该方法不运行在UI线程中,主要用于异步操作,通过调用publishProgress
    ()方法
    //触发onProgressUpdate对UI进行操作
    @Override
    protected String doInBackground(Integer... params) {
        DelayOperator dop = new DelayOperator();
        int i = 0;
        for (i = 10; i <= 100; i+=10)
        {
            dop.delay();
            publishProgress(i);
        }
        return i + params[0].intValue() + "";
    }

    //该方法运行在UI线程中,可对UI控件进行设置
    @Override
    protected void onPreExecute() {
        txt.setText("开始执行异步线程~");
    }
}
```


//在doBackground方法中,每次调用publishProgress方法都会触发该方法
 //运行在UI线程中,可对UI控件进行操作

```
@Override
protected void onProgressUpdate(Integer... values) {
    int value = values[0];
    pgbar.setProgress(value);
}
}
```

MainActivity.java :

```
public class MyActivity extends ActionBarActivity {

    private TextView txttitle;
    private ProgressBar pgbar;
    private Button btnupdate;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        txttitle = (TextView) findViewById(R.id.txttitle);
        pgbar = (ProgressBar) findViewById(R.id.pgbar);
        btnupdate = (Button) findViewById(R.id.btnupdate);
        btnupdate.setOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View v) {
                MyAsyncTask myTask = new MyAsyncTask(txttitle,pgbar);
                myTask.execute(1000);
            }
        });
    }
}
```

本节小结：

好的，本节一开始给大家普及了下应用程序，进程，线程，多线程，异步，同步的概念；接着又讲解了下Android中为何要引入异步操作，然后介绍了下AsyncTask的用法，当然上面也说了，异步操作在网络操作作用的较多，后面在讲解网络操作时会用到这个AsyncTask，敬请期待~本节就到这里，谢谢~



在线实例

- HTML 实例
- CSS 实例
- JavaScript 实例
- Ajax 实例
- jQuery 实例
- XML 实例
- Java 实例

字符集&工具

- HTML 字符集设置
- HTML ASCII 字符集
- HTML ISO-8859-1
- HTML 实体符号
- HTML 拾色器
- JSON 格式化工具

最新更新

- Swift 正式开源
- PHP 7 正式发布
- Shell 编程快速入门
- Shell 文件包含
- Shell 输入/输出...
- Shell printf 命令
- Shell 基本运算符

站点信息

- 意见反馈
- 免责声明
- 关于我们
- 文章归档

关注微信



Copyright © 2013-2015 菜鸟教程 **runoob.com** All Rights Reserved. 备案号：闽ICP备15012807号-1