关于AsyncTask的使用

1. 简介

AsyncTask 是Android 开发一个常用的多线程异步任务组件。

为什么要用多线程异步任务的组件呢？因为安卓应用的主线程（也叫UI线程）不允许有耗时的操作在此线程中执行。比如加载大型图片，访问网络等等。如果占用主线程时间超过了5秒就会Application Not Response（ANR）的异常，也就是很多年前安卓设备的配置不太好时进行出现的程序没有响应，是否需要等待的情况。

虽然现在除了下载时显示下载进度等的需求以外，很少被使用。但是作为安卓框架的组件，大家有必要对AsyncTask有所了解。

2. 说明

2.1 参数说明

我们需要写一个继承自AsyncTask的class。会发现AsyncTask<Params,Progress,Result> 有三个数据类型, 这三个数据类型时可以自定义的。那这三个数据类型分别代表什么呢。

1. 第一个数据类型 Params：
   用于指定doInBackground中传入参数的数据类型。
   用于指定MyTask.execute(Params)中execute的数据类型。

2. 第二个数据类型 Progress：
   用于指定onProgressUpdate(Integer values)的数据类型。

3. 第三个数据类型 Result：
   用于指定doInBackground中返回值的数据类型。
   用于指定onPostExecute中传入参数的数据类型。

2.2 方法说明

讲完参数，该讲AsyncTask中的方法了。

1. onPreExecute()：
   在主线程执行。
   此方法是AsyncTask最先执行的方法，可以用于初始化，doInBackground方法的调用准备。

2. doInBackground(Params... params)：
   在子线程中执行。
   此方法AsyncTask的重点，在这里需要实现你的所有耗时操作。当此方法结束以后，会把返回值返回给主线程

3. publishProgress(Progress... values)：
   在子线程中执行。
   此方法是用于调用，不是用于重写。
   此方法用于通知更新进度信息。
   如果你想每5%更新当前的进度，则可以每到5%时调用此方法。

4. onProgressUpdate(Progress... values)：
   在主线程中执行。
   此方法用于更新View中的进度信息，所以此方法是在主线程中调用。

5. onPostExecute(Result result)：
   在主线程中执行。
   此方法是用于更新最后关于doInBackground返回值的View的更新。

在上述方法中，只有doInBackground 方法是必须要重写的抽象方法，如果你只是想要进行耗时但不需要进度的Task时，你可以只重写doInBackground 了。

2.3 执行方法说明

然后到了最后执行AsyncTask了。把上述示例代码作为例子，MyTask().execute("Hello world") 则方法会休息5秒后返回Hello world PEACE! 的结果。

如果你想要获取返回的值，你可以val reuslt = MyTask().execute("Hello world").get() 来获取。

需要注意的一点是，如果AsyncTask引用了View组件，如TextView组件，则必须要在Activity销毁时取消Task来防止内存泄漏。方法是myTask.cancel(true) 。

3. 示例代码

```
class MyTask: AsyncTask<String, Int, String>() {

    // 必须重写
    override fun onPreExecute() {
        super.onPreExecute()
        Log.d("MyTask", "onPreExecute")
        binding.textView.text = "MyTask onPreExecute"
    }

    // 必须重写
    override fun doInBackground(vararg params: String?): String {
        Log.d("MyTask", "doInBackground")

        for (i in 0..100) {
            Thread.sleep(50)
            publishProgress(i)
        }

        var result = "PEACE!"
        params[0]?.also {
            result = "$it $result"
        }

        return result
    }

    // 不必须重写
    override fun onProgressUpdate(vararg values: Int?) {
        super.onProgressUpdate(*values)

        binding.textView.text = "${values[0]} %"
    }

    // 不必须重写
    override fun onPostExecute(result: String?) {
        super.onPostExecute(result)
```

```
            Log.d("MyTask", "onPostExecute")
            binding.textView.text = "MyTask onPostExecute"
        }

    }
```

4. 简单原理说明

对源码分析就不详细展开了，估计大家也不想看。
简单说明一下AsyncTask的实现原理。因为AsyncTask要在主线程和子线程之间切换进行，内部是通过Handler实现的。
线程管理用的是java的线程池ThreadPoolExecutor 和java的并发阻塞队列LinkedBlockingQueue 来进行线程控制管理。

github： https://github.com/HyejeanMOON/AsyncTaskDemo