# 简介

AsyncTask可以让编程人员更加方便地使用UI线程，它能够让你在不操作Thread和Handler的条件下执行后台任务，并将结果反馈到UI线程中。一般用它来完成一些小型操作（若干秒），如果需要线程长时间运行，应当使用java.util.concurrent包提供的Executor、ThreadPoolExecutor、FutureTask等API。

# 规定

为了更好地使用该类，以下是一些必须遵守的规定：

（1）AsyncTask类必须在UI线程中加载（在4.1及之后已经能自动保证）；

（2）AsyncTask实例必须在UI线程中创建；

（3）execute(Params...)必须在UI线程中调用；

（4）不要手工调用onPreExecute(), onPostExecute(Result), doInBackground(Params...), onProgressUpdate(Progress...)这几个方法；

（5）每个实例只能执行一次，第二次会抛出异常。

（6）从3.1开始，所有AsyncTask都在单一线程中执行，如果确实需要并发，且能够保证不出错，则可以使用executeOnExecutor，并传入AsyncTask的static域THREAD\_POOL\_EXECUTOR。

（7）AsyncTask能够保证所有回调方法是同步好的，只要采取以下做法：

1、在构造器或是onPreExecute()中设置域，并在doInBackground(Params...)方法中引用。

2、在doInBackground(Params...)中设置域，并在onProgressUpdate(Progress...)和onPostExecute(Result)中引用。

# 参数

实际开发中一般通过扩展AsyncTask类，并覆盖它的几个关键方法来使用它。一般可以这么声明：private class MyTask extends AsyncTask<Params, Progress, Result> { ... }，其中三个泛型参数的意义如下：

（1）Params：doInBackground方法的参数类型；

（2）Progress：AsyncTask所执行的后台任务的进度类型；

（3）Result：后台任务的返回结果类型。

如果不需要的话，直接用Void类型代替即可。

# 覆盖方法

以下是可能会用到的，需要覆盖的方法：

（1）onPreExecute()：在任务开始之前调用，工作在UI线程；

（2）doInBackground(Params...)：在onPreExecute()完成后调用，用于处理请求，工作在后台线程。方法内部可以调用publicProgress(Progress…)方法，Progress值会被用于onProgressUpdate(Progress...)方法中。

（3）onProgressUpdate(Progress...)：在调用publishProgress(Progress...)后调用，用于显示进度，工作在UI线程。

（4）onPostExecute(Result)：doInBackground()完成后调用，doInBackground()的返回值会被传入该方法，工作在UI线程。

（5）onCancelled()：调用cancel()取消任务时调用。工作在UI线程。

其中doInBackground(Params...)是必须要覆盖的，onPostExecute(Result)一般也需要覆盖，其他视实际需求而定。

扩展完成并实例化后，使用execute (Params... params)启动任务，该方法返回自己的一个引用（this）。也可使用该类的static方法execute(Runnable)启动。

若需要获取结果，可以使用get()方法，该方法在后台任务未完成的情况下会进入等待状态。

# 取消任务

可以利用cancel(boolean)取消任务，调用该方法后，onCancelled()会在doInBackground(Params...)完成后调用。若想更快地结束任务，可以在doInBackground(Params...)方法内部检查isCancelled()的值。