Evan
Only let oneself become strong enough, good enough, can afford the life that you want to.

android AsyncTask介绍 AsyncTask和Handler对比

标签： android asynctask 异步 线程 handler

2015-12-30 13:07 182人阅读 评论(0) 收藏 举报

分类： 移动开发（38）

1 ） AsyncTask实现的原理,和适用的优缺点

AsyncTask,是**Android**提供的轻量级的异步类,可以直接继承AsyncTask,在类中实现异步操作,**并**提供接口反馈当前**异步执行的程度**(可以通过接口实现UI进度更新),最后反馈执行的结果给UI主线程.

使用的优点:

l 简单,快捷

l 过程可控

使用的缺点:

l 在使用多个异步操作和并需要进行Ui变更时,就变得复杂起来.

2 ）Handler异步实现的原理和适用的优缺点

在Handler 异步实现时,涉及到 Handler, Looper, Message,Thread四个对象，实现异步的流程是主线程启动Thread（子线程）àthread(子线程)运行并生成Message-àLooper获取Message并传递给HandleràHandler逐个获取Looper中的Message，并进行UI变更。

使用的优点：

l 结构清晰，功能定义明确

l 对于多个后台任务时，简单，清晰

使用的缺点：

l 在单个后台异步处理时，显得代码过多，结构过于复杂（相对性）

AsyncTask介绍

Android的AsyncTask比Handler更轻量级一些，适用于简单的异步处理。

首先明确Android之所以有Handler和AsyncTask，都是为了不阻塞主线程（UI线程），且UI的更新只能在主线程中完成，因此异步处理是不可避免的。

android为了降低这个开发难度，提供了AsyncTask。AsyncTask就是一个封装过的后台任务类，顾名思义就是异步任务。

AsyncTask直接继承于Object类，位置为android.os.AsyncTask。要使用AsyncTask工作我们要提供三个泛型参数，并重载几个方法(至少重载一个)。

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，比如HTTP请求的URL。
- Progress 后台任务执行的百分比。
- Result 后台执行任务最终返回的结果，比如String。

使用过AsyncTask 的同学都知道一个异步加载数据最少要重写以下这两个方法：

- doInBackground(Params...) 后台执行，比较耗时的操作都可以放在这里。注意这里不能直接操作UI。此方法在后台线程执行，[illegible]是工作，通常需要较长的时间。在执行过程中可以调用publicProgress(Progress...)来更新任务的进度。
- onPostExecute(Result) 相当于Handler 处理UI的方式，在这里面可以使用在doInBackground 得到的结果处理操作UI。 此方法在主线程执行，任务执行的结果作为此方法的参数返回

有必要的话你还得重写以下这三个方法，但不是必须的：

- onProgressUpdate(Progress...) 可以使用进度条增加用户体验度。 此方法在主线程执行，用于显示任务执行的进度。
- onPreExecute() 这里是最终用户调用Excute时的接口，当任务执行之前开始调用此方法，可以在这里显示进度对话框。
- onCancelled() 用户调用取消时，要做的操作

使用AsyncTask类，以下是几条必须遵守的准则：

- Task的实例必须在UI thread中创建；
- execute方法必须在UI thread中调用；
- 不要手动的调用onPreExecute(), onPostExecute(Result)，doInBackground(Params...), onProgressUpdate(Progress...)这几个方法；
- 该task只能被执行一次，否则多次调用时将会出现异常；

一个超简单的理解 AsyncTask 的例子：

main.xml:

```
[html]
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
    android:id="@+id/textView01"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
   <ProgressBar
   android:id="@+id/progressBar02"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    style="?android:attr/progressBarStyleHorizontal"
    />
    <Button
    android:id="@+id/button03"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="更新progressbar"
    />
</LinearLayout>
```

MainActivity.java

```java
package vic.wong.main;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ProgressBar;
import android.widget.TextView;

public class MainActivity extends Activity {
    private Button button;
    private ProgressBar progressBar;
    private TextView textView;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        button = (Button)findViewById(R.id.button03);
        progressBar = (ProgressBar)findViewById(R.id.progressBar02);
        textView = (TextView)findViewById(R.id.textView01);

        button.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                ProgressBarAsyncTask asyncTask = new ProgressBarAsyncTask(textView, progressBar);
                asyncTask.execute(1000);
            }
        });
    }
}
```

NetOperator.java

```java
package vic.wong.main;


//模拟网络环境
public class NetOperator {

    public void operator(){
        try {
            //休眠1秒
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

}
```

ProgressBarAsyncTask .java

```java
package vic.wong.main;
import android.os.AsyncTask;
import android.widget.ProgressBar;
import android.widget.TextView;

/**
 * 生成该类的对象，并调用execute方法之后
 * 首先执行的是onProExecute方法
 * 其次执行doInBackgroup方法
 *
 */
public class ProgressBarAsyncTask extends AsyncTask<Integer, Integer, String> {

```

```
    private TextView textView;
    private ProgressBar progressBar;


    public ProgressBarAsyncTask(TextView textView, ProgressBar progressBar) {
        super();
        this.textView = textView;
        this.progressBar = progressBar;
    }


    /**
     * 这里的Integer参数对应AsyncTask中的第一个参数
     * 这里的String返回值对应AsyncTask的第三个参数
     * 该方法并不运行在UI线程当中，主要用于异步操作，所有在该方法中不能对UI当中的空间进行设置和修改
     * 但是可以调用publishProgress方法触发onProgressUpdate对UI进行操作
     */
    @Override
    protected String doInBackground(Integer... params) {
        NetOperator netOperator = new NetOperator();
        int i = 0;
        for (i = 10; i <= 100; i+=10) {
            netOperator.operator();
            publishProgress(i);
        }
        return i + params[0].intValue() + "";
    }


    /**
     * 这里的String参数对应AsyncTask中的第三个参数（也就是接收doInBackground的返回值）
     * 在doInBackground方法执行结束之后在运行，并且运行在UI线程当中 可以对UI空间进行设置
     */
    @Override
    protected void onPostExecute(String result) {
        textView.setText("异步操作执行结束" + result);
    }


    //该方法运行在UI线程当中,并且运行在UI线程当中 可以对UI空间进行设置
    @Override
    protected void onPreExecute() {
        textView.setText("开始执行异步线程");
    }


    /**
     * 这里的Intege参数对应AsyncTask中的第二个参数
     * 在doInBackground方法当中，，每次调用publishProgress方法都会触发onProgressUpdate执行
     * onProgressUpdate是在UI线程中执行，所有可以对UI空间进行操作
     */
    @Override
    protected void onProgressUpdate(Integer... values) {
        int vlaue = values[0];
        progressBar.setProgress(vlaue);
    }

}
```


- 上一篇 使用Jsoup 抓取页面的数据
- 下一篇 Android 中Webview 自适应屏幕