Android 多任务

对于 Android 程序中，使用多线程的技术是必不可少的，就拿之前最简单的例子来说明，对于 Android4.0+的应用而言，访问网络必须另起线程才可以访问。

当应用程序启动时，系统会创建一个执行线程在这个应用程序的的进程中，一般被称为“主线程”。这个线程是非常重要的，因为它负责把事件分发给响应的用户组件，包括绘制事件等，因此主线程又被称为 UI 线程。系统并不会为每个组件创建一个单独的线程，而是在 UI 线程中，完成这些组件的初始化的，因此系统回调方法是运行在 UI 线程中，如 click 事件。

Android 的 UI ToolKit 包下的所有组件都不是线程安全的，所以，不能在一个单独的工作线程中操作这些 UI 组件，必须在 UI 线程中操作。因此，对于单线程模型，Android 有两个规则：

1. 不能阻塞 UI 线程
2. 不能在工作线程中访问 Android UI ToolKit 包下的组件。

MyWorkThread

Step1：Android Studio 中新建一个空工程。

Step2：在 app/manifests/AndroidManifest.xml 中添加

<uses-permission android:name="android.permission.INTERNET" />

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myworkthread">

    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Step3：在 app/res/layout/activity_main.xml 中设计 UI，添加三个 button。

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.myworkthread.MainActivity">

    <Button
        android:id="@+id/btnError"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Error" />


    <Button
        android:id="@+id/btnRunOnUiThread"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="RunOnUiThread"
        android:layout_below="@+id/btnError" />

    <Button
        android:id="@+id/btnPost"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Post"
        android:layout_below="@+id/btnRunOnUiThread" />

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/btnPost" />

</RelativeLayout>
```

Step4：程序中需要用到 Apache http 的包，而在最新版的 Android Studio 中已经不支持该包，因此需要在 Gradle Scripts/build.gradle 下添加

useLibrary ‘org.apach.http.legacy’

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 24
    buildToolsVersion "24.0.2"
    defaultConfig {
        applicationId "com.example.myworkthread"
        minSdkVersion 15
        targetSdkVersion 24
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner
"android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles
getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }

    useLibrary 'org.apache.http.legacy'
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])

androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.
2', {
        exclude group: 'com.android.support', module:
'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:24.2.1'
    testCompile 'junit:junit:4.12'
}
```

Step5：在 app/java/com.examle.myworkthread/MainActivity 下添加三个 button 的实现代码。

```
public class MainActivity extends AppCompatActivity {
    private Button btnError, btnRunOnUiThread, btnPost;
    private ImageView imageView;
    private static final String PATH_URL =
"http://images.cnblogs.com/cnblogs_com/plokmju/550907/o_hand.jpg";
    private static final String PATH_URL2 =
"http://ww1.sinaimg.cn/bmiddle/88ff29e8jw1e7pjnpfxbrj20dp0a90tb.jpg";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        imageView = (ImageView) findViewById(R.id.imageView);
        btnRunOnUiThread = (Button) findViewById(R.id.btnRunOnUiThread);
        btnError = (Button) findViewById(R.id.btnError);
        btnPost = (Button) findViewById(R.id.btnPost);

        btnError.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // 增加一个线程访问网络
                new Thread(new Runnable() {
                    @Override
                    public void run() {
                        // 获取网络路径下的图片
                        Bitmap btm = loadImageFromNetwork(PATH_URL);
                        imageView.setImageBitmap(btm);
                    }
                }).start();
            }
        });

        btnRunOnUiThread.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                new Thread(new Runnable() {
                    @Override
                    public void run() {
                        // 获取网络路径下的图片
                        final Bitmap btm = loadImageFromNetwork(PATH_URL);
                        // 在 UI 线程上操作 Bitmap 组件
```

```
                        MainActivity.this.runOnUiThread(new Runnable() {
                            @Override
                            public void run() {
                                imageView.setImageBitmap(btm);
                                Toast.makeText(getApplicationContext(),
"RunOnUiThread", Toast.LENGTH_SHORT).show();
                            }
                        });
                    }
                }).start();
            }
        });

        btnPost.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                new Thread(new Runnable() {
                    @Override
                    public void run() {
                        final Bitmap btm =
loadImageFromNetwork(PATH_URL2);
                        // 将操作 Bitmap 的方式发布到 UI 线程上
                        imageView.post(new Runnable() {
                            @Override
                            public void run() {
                                imageView.setImageBitmap(btm);
                                Toast.makeText(getApplicationContext(),
"Post", Toast.LENGTH_SHORT).show();
                            }
                        });
                    }
                }).start();
            }
        });
    }

    private Bitmap loadImageFromNetwork(String uri) {
        // 根据 URI 地址，获取器地址的图片资源
        Bitmap bitmap = null;
        HttpClient httpClient = new DefaultHttpClient();
        HttpGet httpGet = new HttpGet(uri);
        HttpResponse httpResponse = null;
        try {
            httpResponse = httpClient.execute(httpGet);
```

```
            if (200 == httpResponse.getStatusLine().getStatusCode()) {
                byte[] data =
EntityUtils.toByteArray(httpResponse.getEntity());
                bitmap = BitmapFactory.decodeByteArray(data, 0,
data.length);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

        return bitmap;
    }
}
```

ERROR 按钮新建一个线程下载图片并显示在 ImageView 中。这似乎是合理的，启动了一个新线程来访问网络，但是它违反了规则二，不能在 Android UI 主线程之外修改 UI 组件，而在 click 中 new Thread 的是一个工作线程，在工作线程中无法操作 UI 组件，所以会报错。

有两种方式修正以上错误。

•Activity.runOnUiThread(Runnable)：运行在指定的 UI 线程上，如果当前线程是 UI 线程，那么立即执行，如果当前线程不是 UI 线程，则发布到 UI 线程的事件队列中。

• View.post(Runnable)：将事件发布到 UI 线程中，立即被执行。

程序运行结果：

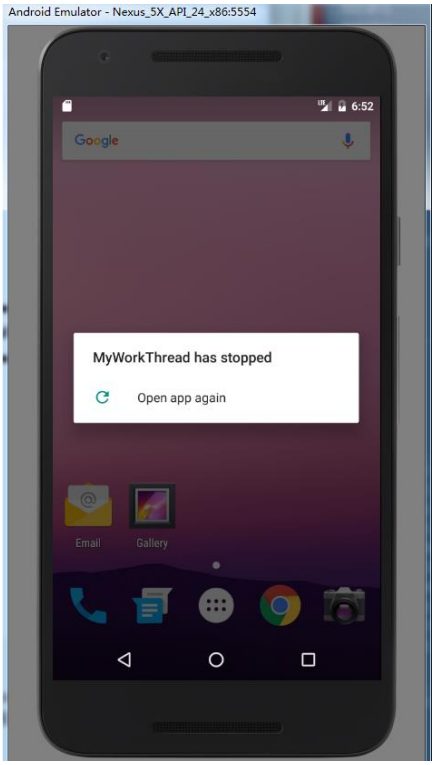

图 1. ERROR

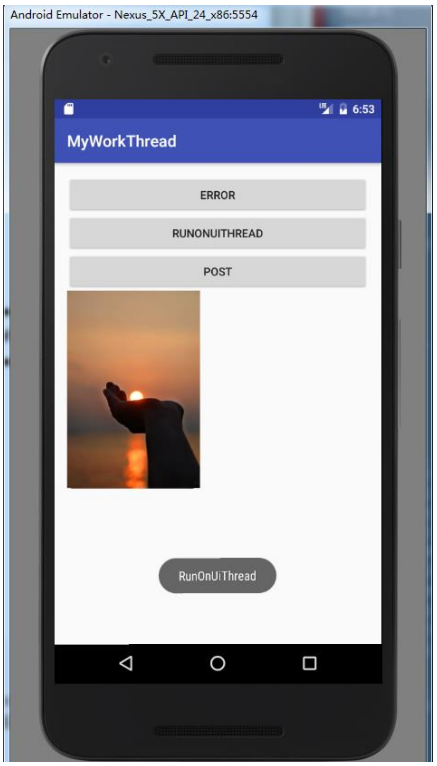

图 2. RUNONUITHREAD

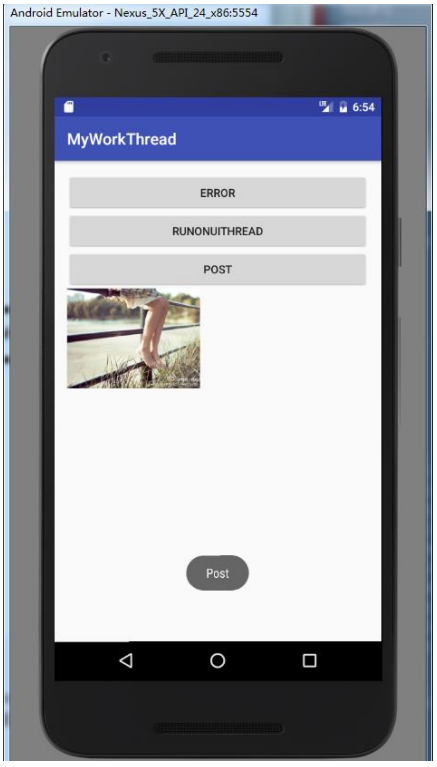

图 3. POST

AsyncTask

AsyncTask，异步任务，可以简单进行异步操作，并把执行结果发布到 UI 主线程。AsyncTask 是一个抽象类，它的内部其实也是结合了 Thread 和 Handler 来实现异步线程操作，但是它形成了一个通用线程框架，更清晰简单。AsyncTask 应该被用于比较简短的操作（最多几秒钟）。

AsyncTask 被定义为一个操作，运行在一个后台线程中，其结果被发布在 UI 线程上。它的异步工作的参数与返回值被泛型的三个参数指定：Params、Progress、Result。AsyncTask 将经历 4 个步骤：onPreExecute、doInBackground、onProgressUpdate、onPostExecute。

三个泛型参数：

- Params：被发送到执行任务的参数类型。
- Progress：进度的类型，发送后台的计算进度到 UI 线程类型。
- Result：异步任务的返回结果类型。

四个阶段：

- OnPreExecute()：执行在 UI 线程上调用执行任务之前，一般用于设置任务。
- doInBackground(Params...)：主要是用来执行异步任务的耗时操作，可以在这个方法中通过 publishProgress()方法发布进度信息，并在执行完成之后，返回执行结果。
- onProgreddUpdate(Progress...)：在 UI 线程上接受 doInBackground()传递过来的进度信息，并在 UI 线程上展示进度信息，它执行的时机是不确定的。
- onPostExecute(Result)：在 UI 线程上操作 doInBackground()执行的返回值。

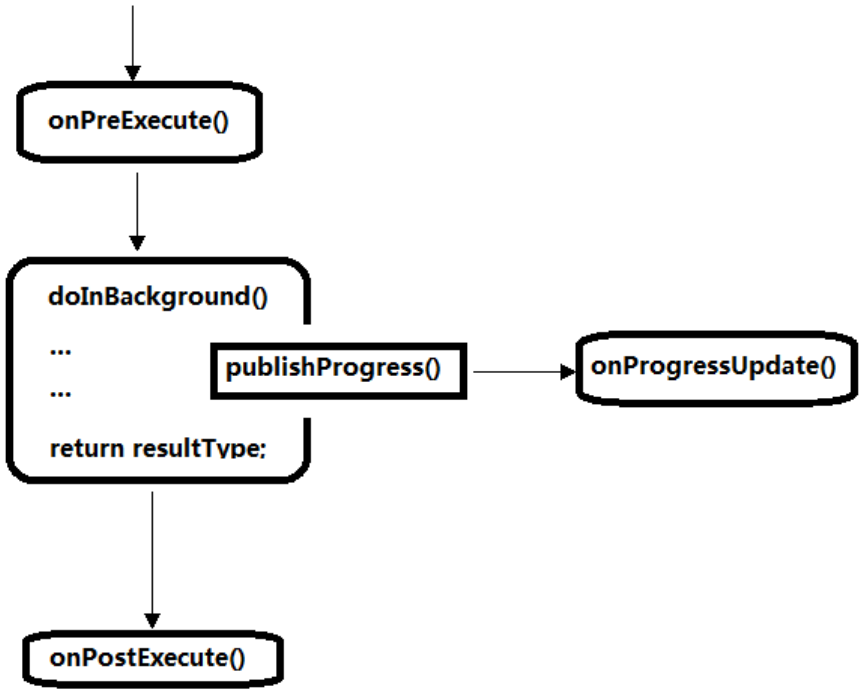

图 4. AsyncTask 四个步骤示意图

AsyncTask 的使用规则：

- AsyncTask 必须声明在 UI 线程上。
- AsyncTask 必须在 UI 线程上实例化。
- 必须通过 execute()方法执行任务。

• 不可以直接调用 onPreExecute()、onPostExecute(Resut)、doInBackground(Params...)、onProgressUpdate(Progress...)方法。

• 可以设置任务只执行一次，如果企图再次执行会报错。

示例代码：

```
public class AsyncTaskActivity1 extends Activity {
    private Button btnDown;
    private ImageView ivImage;
    private static String image_path =
"http://ww4.sinaimg.cn/bmiddle/786013a5jw1e7akotp4bcj20c80i3aao.jpg";
    private ProgressDialog dialog;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.asynctask_activity);

        btnDown = (Button) findViewById(R.id.btnDown);
        ivImage = (ImageView) findViewById(R.id.ivSinaImage);

        // 声明一个等待框以提示用户等待
        dialog=new ProgressDialog(this);
        dialog.setTitle("提示信息");
        dialog.setMessage("正在下载，请稍后...");

        btnDown.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                // 执行一个异步任务，并把图片地址以参数的形式传递进去
                new MyTask().execute(image_path);
            }
        });
    }

    // 以 String 类型的参数，Void 表示没有进度信息，Bitmap 表示异步任务返回
一个位图
    public class MyTask extends AsyncTask<String, Void, Bitmap> {
        // 表示任务执行之前的操作
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            //显示等待框
            dialog.show();
```

```
        }

        //主要是完成耗时操作
        @Override
        protected Bitmap doInBackground(String... params) {
            HttpClient httpClient=new DefaultHttpClient();
            HttpGet httpGet=new HttpGet(params[0]);
            Bitmap bitmap=null;
            try {
                //从网络上下载图片
                HttpResponse httpResponse =httpClient.execute(httpGet);
                if(httpResponse.getStatusLine().getStatusCode()==200){
                    HttpEntity httpEntity = httpResponse.getEntity();
                    byte[] data=EntityUtils.toByteArray(httpEntity);
                    bitmap=BitmapFactory.decodeByteArray(data, 0,
data.length);
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
            return bitmap;
        }

        //完成更新 UI 操作
        @Override
        protected void onPostExecute(Bitmap result) {
            super.onPostExecute(result);
            //设置 ImageView 的显示图片
            ivImage.setImageBitmap(result);
            // 销毁等待框
            dialog.dismiss();
        }
    }
}
```

程序运行结果：

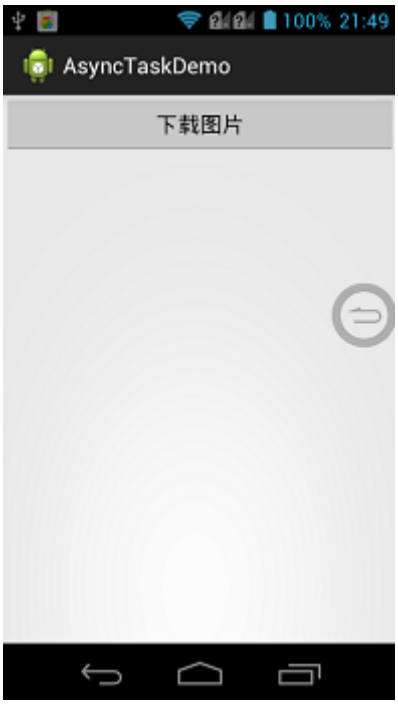

图 5. UI 界面

图 6. 图片下载中

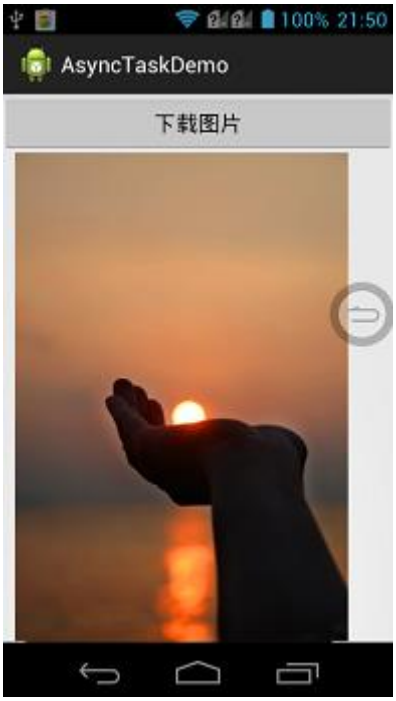

图 7. 图片下载完毕