

Android UI线程和非UI线程

Android UI线程和非UI线程

UI线程及Android的单线程模型原则

当应用启动，系统会创建一个**主线程 (main thread)**。

这个主线程负责向UI组件分发事件（包括绘制事件），也是在这个主线程里，你的应用和Android的**UI组件 (components from the Android UI toolkit (components from the `android.widget` and `android.view` packages))** 发生交互。

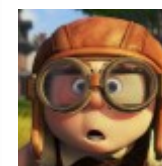
所以**main thread**也叫**UI thread**也即**UI线程**。

系统不会为每个组件单独创建线程，在同一个进程里的UI组件都会在UI线程里实例化，**系统对每一个组件的调用都从UI线程分发出去**。

结果就是，响应系统回调的方法（比如响应用户动作的**`onKeyDown()`**和各种生命周期回调）永远都是在UI线程里运行。

公告

欢迎来到圣骑士
Wind的博客！



Dandan Meng

[My Github](#)

友情链接

[我的github pages](#)

[WeYoung](#)

[yanni4night](#)

微信公众号: 圣骑士Wind: 分享
最新Android开发资讯



当App做一些比较重（intensive）的工作的时候，除非你合理地实现，否则单线程模型的performance会很poor。

特别的是，如果所有的工作都在UI线程，做一些比较耗时的工作比如访问网络或者数据库查询，都会**阻塞UI线程，导致事件停止分发**（包括绘制事件）。对于用户来说，应用看起来像是卡住了，更坏的情况是，如果UI线程blocked的时间太长（大约超过5秒），用户就会看到**ANR**（[application not responding](#)）的对话框。


另外，**Andoid UI toolkit并不是线程安全的，所以你不能从非UI线程来操纵UI组件**。你必须把所有的UI操作放在UI线程里，所以**Android的单线程模型有两条原则**：

- 1.不要阻塞UI线程。
- 2.不要在UI线程之外访问Android UI toolkit（主要是这两个包中的组件：[android.widget](#) and [android.view](#)）。

使用Worker线程

根据单线程模型的两条原则，首先，要保证应用的响应性，不能阻塞UI线程，所以当你的操作不是即时的那种（not instantaneous），你应该把他们放进单另的线程中（叫做**background**或者叫**worker线程**）。

比如点击按钮后，下载一个图片然后在ImageView中展示：



```
public void onClick(View v) {
    new Thread(new Runnable() {
        public void run() {
            Bitmap b = loadImageFromNetwork("http://example.com/image.png");
            mImageView.setImageBitmap(b);
        }
    })
```



Links :

[My Github](#)
[Android API Guides](#)
[GrepCode Android](#)
[AndroidXRef](#)
昵称：[圣骑士wind](#)
园龄：[6年6个月](#)
粉丝：[1359](#)
关注：[68](#)
[+加关注](#)

<		2018年3月			
日	一	二	三	四	
25	26	27	28	1	
4	5	6	7	8	
11	12	13	14	15	
18	19	20	21	22	
25	26	27	28	29	
1	2	3	4	5	

```
}).start();  
}
```

这段代码用新的线程来处理网络操作，但是它违反了第二条原则：

Do not access the Android UI toolkit from outside the UI thread.

从非UI线程访问UI组件会**导致未定义和不能预料的行为**。

为了解决这个问题，Android提供了一些方法，从其他线程访问UI线程：

- [Activity.runOnUiThread\(Runnable\)](#)
- [View.post\(Runnable\)](#)
- [View.postDelayed\(Runnable, long\)](#)

比如，上面这段代码可以这么改：

```
public void onClick(View v) {  
    new Thread(new Runnable() {  
        public void run() {  
            final Bitmap bitmap = loadImageFromNetwork("http://example.com/image.png");  
            mImageView.post(new Runnable() {  
                public void run() {  
                    mImageView.setImageBitmap(bitmap);  
                }  
            });  
        }  
    });  
}
```

搜索


常用链接

[我的随笔](#)
[我的评论](#)
[我的参与](#)
[最新评论](#)
[我的标签](#)

最新随笔

1. [Dagger2进阶必备技能](#)
2. [20+个很棒的Android开源项目](#)
3. [网络使用和电池消耗 原因和改进](#)
4. [Android Realm数据库使用指南](#)
5. [Say Hello to ConstraintLayout](#)
6. [Android Weekly Notes Issue #237](#)
7. [Android Weekly Notes Issue #236](#)
8. [Android Weekly Notes Issue #235](#)
9. [Android Weekly Notes Issue #234](#)

```
}).start();  
}
```



这么改之后就是线程安全的了。

但是，当操作变得复杂的时候，这种代码会变得非常复杂，为了处理非UI线程和UI线程之间更加复杂的交互，可以考虑在worker线程中使用一个[Handler](#)，来处理UI线程中传来的消息。

也可以继承这个类[AsyncTask](#)。

Communicating with the UI Thread

只有在UI线程中的对象才能操作UI线程中的对象，为了将非UI线程中的数据传送到UI线程，可以使用一个 [Handler](#)运行在UI线程中。

[Handler](#)是Android framework中管理线程的部分，一个[Handler](#)对象负责接收消息然后处理消息。

你可以为一个新的线程创建一个[Handler](#)，也可以创建一个[Handler](#)然后将它和已有线程连接。

如果你将一个[Handler](#)和你的UI线程连接，处理消息的代码就将会在UI线程中执行。

可以在你创建线程池的类的构造方法中实例化Handler的对象，然后用全局变量存储这个对象。

要和UI线程连接，实例化Handler的时候应该使用[Handler\(Looper\)](#) 这个构造方法。

这个构造方法使用了一个 [Looper](#) 对象，这是Android系统中线程管理的framework的另一个部分。

10. Android Weekly Notes Issue #233

我的标签

[Android](#)(178)

[Java](#)(86)

[Android Weekly](#)(55)

[Web](#)(38)

[设计模式](#)(29)

[JavaScript](#)(29)

[Kotlin](#)(29)

[RxJava](#)(27)

[Animation](#)(19)

[数据结构和算法](#)(17)

[更多](#)

随笔分类(521)

[3D基础](#)(3)

[Android](#)(107)

[Android Animation](#)(8)

[Android Testing](#)(3)

[Android Weekly](#)(57)

[Android 进阶](#)(8)

[Android开发相关类库](#)(3)

[C#](#)(3)

[C++](#)(1)

[CMake](#)(2)

[DirectX](#)(3)

[Git](#)(7)

[GPU](#)(1)

当你用一个特定的 **Looper** 实例来创建一个 **Handler** 时，这个 **Handler** 就运行在这个 **Looper** 的线程中。

在Handler中，要覆写 `handleMessage()` 方法。Android系统**会在Handler管理的相应线程收到新消息时调用这个方法**。

一个特定线程的所有Handler对象都会收到同样的方法。（这是一个“一对多”的关系）。

参考资料

官方Training: 与UI线程通信：

<http://developer.android.com/training/multiple-threads/communicate-ui.html>

Guides: Processes and Threads

<http://developer.android.com/guide/components/processes-and-threads.html>

类参考：

<http://developer.android.com/reference/android/os/Looper.html>

<http://developer.android.com/reference/android/os/Handler.html>

<http://developer.android.com/reference/android/os/HandlerThread.html>

博客：

Java(83)
JavaScript(24)
JNI(1)
jQuery(10)
JUnit(13)
JVM(7)
Linux(6)
MySQL(3)
OpenCV(2)
OpenGL(1)
OpenGL ES(1)
Python(2)
Testing(1)
Tools & IDE(2)
UX(5)
Web(28)
Windows(2)
XML(12)
XNA(1)
笔试面试(2)
编程调试(4)
服务器(3)
科研什么的(12)
每天一个算法题(10)
设计模式(29)
数据结构和算法(19)
数据库(7)
图形图像(2)
图形学(10)
网络(13)

随笔档案(444)

Android的线程使用来更新UI----Thread、Handler、Looper、TimerTask等：

<http://www.cnblogs.com/playing/archive/2011/03/24/1993583.html>

分类: [Android](#)

标签: [Android](#), [多线程](#)

好文要顶

关注我

收藏该文



圣骑士wind

关注 - 68

粉丝 - 1359



8

0

+加关注

« 上一篇：[Android中的Touch事件](#)

» 下一篇：[Android 尺寸单位转换和屏幕适配相关](#)

posted @ 2013-11-12 00:38 [圣骑士wind](#) 阅读(48908) 评论(4) 编辑 收藏

评论列表

#1楼 2014-02-11 10:43 [无敌小龙](#)

非常感谢楼主!受教了!

支持(0) 反对(0)

[2017年12月](#) (5)

[2017年11月](#) (4)

[2017年10月](#) (4)

[2017年9月](#) (2)

[2017年6月](#) (1)

[2017年5月](#) (4)

[2017年4月](#) (6)

[2017年3月](#) (5)

[2017年2月](#) (6)

[2017年1月](#) (4)

[2016年12月](#) (5)

[2016年11月](#) (6)

[2016年10月](#) (8)

[2016年9月](#) (5)

[2016年8月](#) (4)

[2016年7月](#) (1)

[2016年6月](#) (5)

[2016年5月](#) (1)

[2016年3月](#) (1)

[2015年10月](#) (1)

[2015年6月](#) (1)

[2015年5月](#) (3)

[2015年4月](#) (5)

[2015年3月](#) (1)

[2015年2月](#) (1)

[2015年1月](#) (6)

[2014年12月](#) (4)

[2014年9月](#) (2)

[2014年7月](#) (3)

[2014年6月](#) (1)

[2014年5月](#) (4)

[2014年4月](#) (17)

#2楼 2014-07-10 22:17 生火

这知识点对我现在来说太好，要是早点看到，就不用走弯路了。谢谢楼主，我爱你分享。

支持(0) 反对(0)

#3楼 2016-09-11 08:44 niky

楼主总结的不错，但是呢，案例使用代码再解释。应该更清晰！

支持(0) 反对(0)

#4楼 2017-09-14 08:25 澎湃的代码激情

“可以考虑在worker线程中使用一个Handler，来处理UI线程中传来的消息”应该是：可以考虑在UI线程中使用一个Handler，来处理worker线程中传来的消息。

支持(0) 反对(0)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问](#)网站首页。

【推荐】超50万VC++源码: 大型组态工控、电力仿真CAD与GIS源码库！

【活动】2050 大会日程，5.25-5.27相聚杭州·云栖

【抢购】新注册用户域名抢购1元起

【推荐】云+校园计划邀请好友拼团有礼，奖励多多

2014年3月 (10)
2014年2月 (11)
2014年1月 (3)
2013年12月 (4)
2013年11月 (7)
2013年10月 (2)
2013年9月 (11)
2013年8月 (14)
2013年7月 (10)
2013年6月 (34)
2013年5月 (26)
2013年4月 (21)
2013年3月 (38)
2013年2月 (25)
2013年1月 (38)
2012年12月 (18)
2012年11月 (20)
2012年10月 (4)
2012年9月 (3)
2012年8月 (13)
2012年7月 (2)
2012年5月 (1)
2012年4月 (2)
2011年8月 (1)

文章档案(8)

2015年4月 (1)
2015年1月 (1)
2013年12月 (1)
2013年8月 (3)
2013年3月 (1)
2012年11月 (1)



最新IT新闻:

- [Chrome OS新功能：全新壁纸筛选器、密码导出等](#)
- [谷歌72位量子计算机面世，比特币还安全吗？](#)
- [美团来打车、滴滴点外卖，但价格战可能打不起来了](#)
- [深度|“粮草”困局下 共享单车觅暖春](#)
- [百度新篇：陆奇之治](#)
- » [更多新闻...](#)



最新知识库文章:

- [写给自学者的入门指南](#)
- [和程序员谈恋爱](#)
- [学会学习](#)
- [优秀技术人的管理陷阱](#)
- [作为一个程序员，数学对你到底有多重要](#)

2018年11月 (+)

Android学习

[Android源码查看](#)

[Android源码项目git](#)

[老罗的Android之旅](#)

官方文档

[Android 官网](#)

[Git Reference](#)

[Python](#)

[Python官网](#)

开源社区

[GitHub](#)

[Google Code](#)

学习链接

[A Byte of Python](#)

[某G家程序员](#)

积分与排名

积分 - 856027

排名 - 101

最新评论

[» 更多知识库文章...](#)

1. Re:OpenCV实现人脸检测

解决无数人对安装opencv环境变量疑惑的终极答案:CentOS6.9
64位+opencv 2.4.13.5+Eclipse
CDT开发环境搭建+用Hog进行
行人检测:...

--hillpig

2. Re:Java IO File类

很好!

--亡灵序曲哦

3. Re:Java XML解析工具 dom4j 介绍及使用实例

感谢博主给分享的知识，感觉十分
的受用 对于java方面的资料我
曾经也看到过一个资料很全的地
方，“[java视频资料获取地址](#)”可
以免费领取

--开黑

4. Re:Java 静态导入

很棒，言简意赅

--东北小狐狸

5. Re:Java 多线程（三）线程 的生命周期及优先级

不错还可以再深入

--郑文强

阅读排行榜

1. Android PopupWindow的使用
和分析(221083)

2. Java 数组基础(143429)

3. Git常用命令总结(133996)

[4. Android Service的生命周期\(132052\)](#)

[5. Android Fragment和Activity\(123290\)](#)

评论排行榜

[1. Google Maps Android API V2的使用及问题解决\(151\)](#)

[2. OpenCV实现人脸检测\(15\)](#)

[3. Java 多线程（七）线程间的通信\(14\)](#)

[4. Android HTTP实例 使用GET方法和POST方法发送请求\(13\)](#)

[5. Android 抽屉效果的导航菜单实现\(13\)](#)

推荐排行榜

[1. Java 多线程（六）synchronized关键字详解\(19\)](#)

[2. Android Fragment 基本介绍\(16\)](#)

[3. Android PopupWindow的使用和分析\(13\)](#)

[4. Java 异常基础 Exception\(12\)](#)

[5. Android 抽屉效果的导航菜单实现\(12\)](#)

