**App冷启动，你还要我怎样？**

**什么叫冷启动**

拿起你的手机，把后台的任务都清理掉，然后再点击你想要打开的app图标，这个时候，就是所谓的app冷启动了。有人肯定会说，有冷启动那不是还有热启动？对的，当你退出app的时候，没把后台的任务清理，并且系统没有把这个app的进程服务给干掉，然后你点击这个app的图标再次进入的时候，这就是热启动了。

**背景**

有时候，你在打开一个app的时候会发现，你点击了这个app的图标，打开了之后会出现一段时间的白屏或者图标卡顿了一会再进入app界面，这个时候你就会吐槽了，为什么会那么卡的？难道是我的手机永久了变卡的原因吗？并不是的，这是因为app的冷启动时间过长的原因造成app进入假死状态了。这也是我们本篇文章所要优化的。

**解决**

我们先来解决打开app的时候闪白屏的这个问题。

首先我们得知道为什么会出现这个白屏的，在AndroidManifest.xml配置文件里面，你会看到你的LAUNCHER activity的配置

<activity

android:name=".StartPageActivity"

android:screenOrientation="portrait">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

如何你的LAUNCHER activity的配置是这样的话，那么恭喜你，打开app的时候就会出现白屏了。那么，怎么配置才可以解决这个问题呢？

我们先添加一个Android Theme

<style name="AppBlackTheme" parent="Theme.AppCompat.Light.NoActionBar">

<item name="android:windowIsTranslucent">true</item>

<item name="android:windowBackground">@android:color/transparent</item>

</style>

这个主题很简单，就是设置为透明的，然后我们再把这个主题添加到刚刚的LAUNCHER activity上。

<activity

android:name=".StartPageActivity"

android:theme="@style/AppBlackTheme"

android:screenOrientation="portrait">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

这个时候你再打开你的app，就会发现不会再闪白屏了，是不是非常的简单呢？这里效果图我后期再进行添加了，先把代码和原理在这里说清楚了。

随着app的功能不断完善，业务的不断发展，接入的第三方服务越来越多，你的app将变得越来越庞大，这个时候你就会发现，当你想进入app的时候，点击了图标，就会发现进入app的时候变得更长了，等待的时候也变得更长了，而且点击了图标之后没有立刻做出反应，而是进入到假死的状态了，当你等待超过了5秒再操作手机的时候，又会出现另一个问题了：ANR（应用程序无响应），这个就是我们这篇文章所要解决的最重要的问题了。

问题所在

后台无app残留的进程服务，重新打开app的时候，从初始化application，结束后再到欢迎界面，很多情况下，就是因为初始化application的时间过长导致了app假死的状态，然后再导致ANR的，所以我们就要优化application的初始化时间，如何去缩短初始化的时间，让app运行得更加流畅。

方法一：IntentService

简单说，IntentService是继承于Service并处理异步请求的一个类，在IntentService内有一个工作线程来处理耗时操作，启动IntentService的方式和启动传统Service一样，同时，当任务执行完后，IntentService会自动停止，而不需要我们去手动控制。另外，可以启动IntentService多次，而每一个耗时操作会以工作队列的方式在IntentService的onHandleIntent回调方法中执行，并且，每次只会执行一个工作线程，执行完第一个再执行第二个，以此类推。而且，所有请求都在一个单线程中，不会阻塞应用程序的主线程（UI Thread），同一时间只处理一个请求。

看到IntentService的介绍，我们就想，可以不可以把耗时的初始化都丢到这里来呢？

public class InitializeService extends IntentService {

private static final String ACTION\_INIT\_WHEN\_APP\_CREATE = "com.youcheyihou.iyoursuv.service.action.INIT";

public InitializeService() {

super("InitializeService");

}

public static void start(Context context) {

Intent intent = new Intent(context, InitializeService.class);

intent.setAction(ACTION\_INIT\_WHEN\_APP\_CREATE);

context.startService(intent);

}

@Override

protected void onHandleIntent(Intent intent) {

if (intent != null) {

final String action = intent.getAction();

if (ACTION\_INIT\_WHEN\_APP\_CREATE.equals(action)) {

//初始化

}

}

}

}

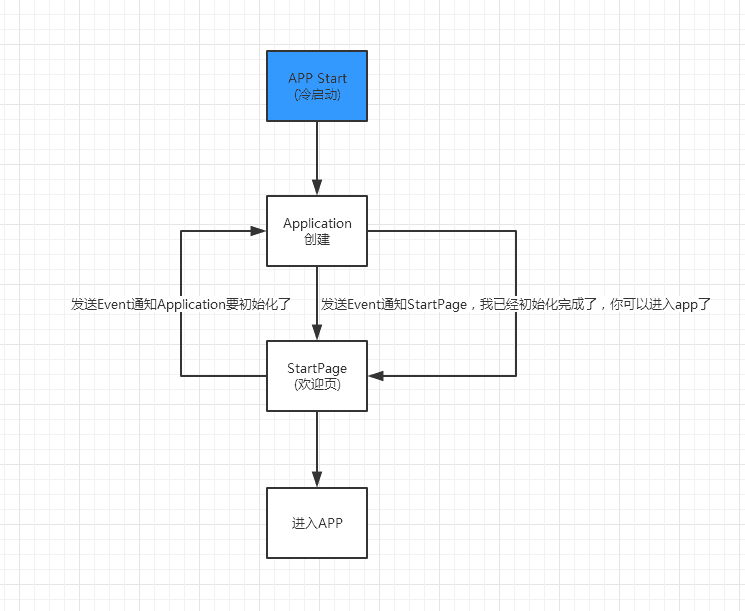
我们看到InitializeService.java的代码中，有一个onHandleIntent的方法，这个就是根据不同action来进行操作的，所以我们可以通过在onHandleIntent中来初始化一些耗时的操作，然后在application中onCreate中执行就行了，是不是很简单？下来你们可以试一下哦。

那么问题来了，有人会说，我怎么知道IntentService什么时候初始化完啊？而且这个是异步的，如何还没初始化完app就进入了主界面，就会出现第三方库或者其他异常了啊？

是的，我们无法确定什么时候确定什么时候能初始化完，所以这些问题我们始终都会遇到，比如我在项目中使用的时候就出现过，有一些第三方库就是必须在application创建的时候进行初始化，进入到app才可以使用的情况（ps：其实这样的第三方库真的是脑残，为什么非要在application中初始化呢？你的库初始化好了，我的app的用户体验就下降了）。那么，还有其他方法去解决初始化时间的问题吗？那肯定有啦，刚刚不是说了是方法一吗？现在就开始介绍方法二了。

方法二：简单粗暴法

何为简单粗暴法呢？且听我慢慢讲解。既然大部分第三方库和资源要在application中初始化，那就让它初始化嘛，但是这个前提是要先进入我们的欢迎界面StartPageActivity，然后再初始化。啊？这里是不是搞得头晕脑胀了，刚刚不是说了先创建application再跳转到StartPageActivity的吗？怎么又反过来了，别急，我们慢慢来说，不过首先你的项目要集成了EventBus，而且你要会用EventBus哦。



上图是方法二app冷启动的打开流程图，这个流程我就不说了，自己看图吧，这里代码也没什么还写，无非就是写EventBus的发送事件和处理事件，这个也不用上代码了吧？可以实践一下，这个也很简单的，有人会说，这个能优化的时间有多少啊？我大概测试了一下，优化的空间还是比较大的，可以缩短50%的时间左右，也就是说如果你的app一开始打开的时候需要三四秒，使用了这个简单粗暴法之后只需要一秒多就能打开了，是不是很爽？

总结

其实，我一直都这样认为的，代码是死的，但是人是活的，所以我们可以使用很多简单粗暴的方法去解决一下比较顽固的问题，不要一味说什么不符合逻辑不符合代码规范(不过一些最基本的代码规范还是要遵循的，因为代码写出来不只是给自己看的)，发散思维，总会有一些方法会被你找到去解决遇到的问题的。共勉！