

个人资料



bluedog

访问：93634次
积分：1068
等级：BLOG 4
排名：千里之外

原创：12篇 转载：0篇
译文：0篇 评论：16条

文章搜索

文章分类

[Android基础框架 \(4\)](#)
[Android应用 \(8\)](#)

文章存档

[2011年05月 \(4\)](#)
[2011年03月 \(8\)](#)

阅读排行

View的刷新机制

(20740)

Activity的启动流程

(19372)

Android的CursorAdapter

(12336)

AsyncQueryHandler异步

(7849)

Activity的View构建过程

(6815)

Android性能优化总结

(5184)

Android ListView快速查找

(4943)

Android数据库随笔

(3784)

Android数据库接口性能

(3078)

Activity的启动流程 (二)

(2781)

评论排行

Activity的启动流程

(7)

View的刷新机制

(3)

Activity的View构建过程

(2)

赠书 | 异步2周年,技术图书免费选 程序员8月书讯 项目管理+代码托管+文档协作,开发更流畅

Activity的启动流程

标签：token null thread application string object

2011-03-09 18:56 19373人阅读 评论(7) 收藏 举报

分类：Android基础框架 (3)

版权声明：本文为博主原创文章，未经博主允许不得转载。

以前看了很多，时间长了都忘了，所以还是勤快点，把看到的都记下来，算是给自己点积累。

Activity启动分为很多种情况，这里说的是打开新的应用程序第一个Activity的流程。

1. ActivityManager产生新进程，新进程从Android.app.ActivityThread.main开始运行。这里就是一般意义上的程序入口点，类似于C的main函数。

ActivityManagerService.Java

```
private final void startProcessLocked(ProcessRecord app, String hostingType, String hostingNameStr) {  
  
    // Process里面通知Zygote服务，Zygote真正产生新的进程（准确说是复制一个）  
  
    int pid = Process.start("android.app.ActivityThread",  
        mSimpleProcessManagement ? app.processName : null, uid, uid,  
  
    }  

```

2. ActivityThread的main函数中将Looper准备起来，prepareMainLooper标志着这是应用程序主线程的Looper对象。

ActivityThread.java

```
public static final void main(String[] args) {  
  
    Looper.prepareMainLooper();  
  
  
    ActivityThread thread = new ActivityThread();  
    thread.attach(false);  
    // 这里闭合消息循环  
    Looper.loop();  
  
    }  

```

3. 接下来调用attach，参数为false，表明这不是系统进程，是给普通应用

ActivityThread.java

关闭



免费云主机试用一年



http://blog.csdn.net/dragondog/article/details/6234972

1/6

- [Android ListView快速查找](#) (2)
- [Android性能优化总结](#) (2)
- [Activity的启动流程（二）](#) (1)
- [Android的CursorAdapter](#) (0)
- [Android数据库接口性能](#) (0)
- [HandlerThread](#) (0)
- [Android数据库随笔](#) (0)

推荐文章

- * CSDN日报20170828——《4个方法快速打造你的阅读清单》
- * CSDN博客模板调查问卷
- * 秒杀系统的一点思考
- * TCP网络通讯如何解决分包问题
- * 技术与技术人员价值
- * GitChat: 人工智能 | 除了深度学习，机器翻译还需要啥？

最新评论

- [Activity的启动流程](#)
Boolean布尔: 博主，我要给你生孩子
- [Activity的启动流程](#)
qluojie: 怎么这么厉害，你是怎么做到的，这个知识能告诉我是什么样领悟到的呀，看源码吗！
- [Android性能优化总结](#)
JiXianWanMei999: 3. 数据交换与网络和数据库的数据交换，不要为了方便，把类型都定义为String。其实我也不赞同这样子...
- [Android ListView快速查找](#)
起个名字好难: 同求例子
- [View的刷新机制](#)
xiaoxia19920920: android源码是最好的代码
- [Activity的启动流程](#)
放风筝的骚年: 很好 很感谢
- [Activity的启动流程（二）](#)
放风筝的骚年: 比较合口味 很好
- [Android性能优化总结](#)
kuangjiu: 说实话 就第一条有点参考意义，嘿嘿希望写点 组件优化之类的，，比如android四大组件的调用，销毁...
- [Activity的View构建过程](#)
kuangjiu: 感谢楼主无私分享
- [Activity的View构建过程](#)
kuangjiu: 但是这里需要注意的是，setContentView创建这些布局结构后，并没有立即跟系统的事件系统接上...



app开发报价单



```
private final void attach(boolean system) {

    ViewRoot.addFirstDrawHandler(new Runnable() {
        public void run() {
            ensureJitEnabled();
        }
    });

    RuntimeInit.setApplicationObject(mAppThread.asBinder());
    IActivityManager mgr = ActivityManagerNative.getDefault();
    mgr.attachApplication(mAppThread);
}

mAppThread是ApplicationThread对象，是提供给ActivityManagerService控制ActivityThread的

private final class ApplicationThread extends ApplicationThreadNative {

    public final void schedulePauseActivity(IBinder token, boolean finished,boolean userLeaving, int
configChanges){

        queueOrSendMessage(finished ? H.PAUSE_ACTIVITY_FINISHING : H.PAUSE_ACTIVITY,
            token, (userLeaving ? 1 : 0), configChanges);
    }

    public final void scheduleStopActivity(IBinder token, boolean showWindow, int configChanges) {
        queueOrSendMessage(showWindow ? H.STOP_ACTIVITY_SHOW : H.STOP_ACTIVITY_HIDE,
            token, 0, configChanges);
    }

    .....
}

ActivityManagerService要Pause当前Activity，就会调用schedulePauseActivity想本地的消息循环中加入一个
H.PAUSE_ACTIVITY的消息，然后立即返回以避免ActivityManagerService的阻塞。
```

4. 现在又回到了ActivityManagerService中

ActivityManagerService.java

```
// ActivityManagerService中XXXLocked函数才是真正干活的地方，XXX只是个套

public final void attachApplication(IApplicationThread thread) {
    int callingPid = Binder.getCallingPid();
    attachApplicationLocked(thread, callingPid);
}

private final boolean attachApplicationLocked(IApplicationThread thr

// 这里取出对应该Pid的ProcessRecord对象，如果取不出来，你就

app = mPidsSelfLocked.get(pid)

// 为ProcessRecord对象补充信息

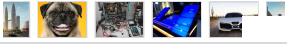
app.thread = thread;
app.curAdj = app.setAdj = -100;
```

关闭



免费云主机试用一年

硬盘数据恢复



```

app.curSchedGroup = Process.THREAD_GROUP_DEFAULT;
app.setSchedGroup = Process.THREAD_GROUP_BG_NONINTERACTIVE;
app.forcingToForeground = null;
app.foregroundServices = false;
app.debugging = false;
// 清除timeout监测
mHandler.removeMessages(PROC_START_TIMEOUT_MSG, app);

// 回调之前的ActivityThread，让它记住自己在ActivityManagerService中的相关信息，传这么大坨东西，真给力

thread.bindApplication(processName, app.instrumentationInfo != null
    ? app.instrumentationInfo : app.info, providers,
    app.instrumentationClass, app.instrumentationProfileFile,
    app.instrumentationArguments, app.instrumentationWatcher, testMode,
    isRestrictedBackupMode || !normalMode,
    mConfiguration, getCommonServicesLocked());

// topRunningActivityLocked的意思没看太明白

HistoryRecord hr = topRunningActivityLocked(null);

// 启动Activity

realStartActivityLocked(hr, app, true, true);
}

private final boolean realStartActivityLocked(HistoryRecord r,
    ProcessRecord app, boolean andResume, boolean checkConfig){

// 这里又跑到ActivityThread中去了

app.thread.scheduleLaunchActivity(new Intent(r.intent), r, System.identityHashCode(r),
    r.info, r.icicle, results, newIntents, !andResume, isNextTransitionForward());

}

```

5.ActivityThread开始调度用户的Activity启动了

ActivityThread.java

```

private final void handleLaunchActivity(ActivityRecord r, Intent customIntent) {
    Activity a = performLaunchActivity(r, customIntent);

    .....
}

```

```

private final void performLaunchActivity(ActivityRecord r, Intent customIntent) {

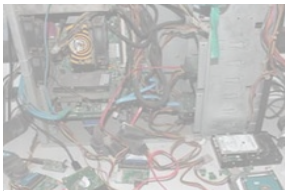
    java.lang.ClassLoader cl = r.packageInfo.getClassLoader();
    // 产生Activity
    activity = mInstrumentation.newActivity(cl, component.getClassName());

    // 将新生的Activity与当前应用关联

```

关闭

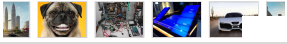
app开发报价单



免费云主机试用一年



硬盘数据恢复



```
activity.attach(appContext, this, getInstrumentation(), r.token,
    r.ident, app, r.intent, r.activityInfo, title, r.parent,
    r.embeddedID, r.lastNonConfigurationInstance,
    r.lastNonConfigurationChildInstances, config);

.....

}
```

6.Activity与当前App关联，直到这里，平时应用所见的Activity才真正被构建

Activity.java

```
final void attach(Context context, ActivityThread aThread, Instrumentation instr, IBinder token, int
    Application application, Intent intent, ActivityInfo info, CharSequence title, Activity parent, Strin
    Object lastNonConfigurationInstance, HashMap<String, Object> lastNonConfigurationChildInsta
    Configuration config) {
```

// 记录传入ContextImpl实例，这就是平时所见的Activity Context

```
attachBaseContext(context);
```

// 创建与Activity关联的Window，可以简单理解为显示窗口

```
mWindow = PolicyManager.makeNewWindow(this);
```

```
mUiThread = Thread.currentThread();
```

```
mMainThread = aThread;
```

```
mInstrumentation = instr;
```

```
mToken = token;
```

```
mIdent = ident;
```

```
mApplication = application;
```

```
mIntent = intent;
```

```
mComponent = intent.getComponent();
```

```
mActivityInfo = info;
```

```
mTitle = title;
```

```
mParent = parent;
```

```
mEmbeddedID = id;
```

// 构建WindowManager的代理LocalWindowManager

```
mWindow.setWindowManager(null, mToken, mComponent.flattenToString());
```

```
if (mParent != null) {
```

// 难道说整个Activity栈中的Activity都有同一个Container

```
mWindow.setContainer(mParent.getWindow());
```

```
}
```

```
mWindowManager = mWindow.getWindowManager();
```

```
}
```

7.ActivityThread

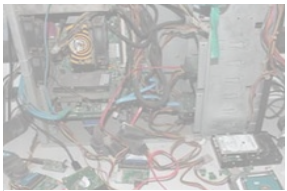
ActivityThread.java

```
private final void performLaunchActivity(ActivityRecord r, Intent customI
```

```
.....
```

// 回调Activity::onCreate

app开发报价单



免费云主机试用一年



关闭

硬盘数据恢复



```
mInstrumentation.callActivityOnCreate(activity, r.state);

// 记录新产生的Activity
mActivities.put(r.token, r);

}

private final void handleLaunchActivity(ActivityRecord r, Intent customIntent) {

    .....

    handleResumeActivity(r.token, false, r.isForward);

}

final void handleResumeActivity(IBinder token, boolean clearHide, boolean isForward) {
    ActivityRecord r = performResumeActivity(token, clearHide);

}

public final ActivityRecord performResumeActivity(IBinder token, boolean clearHide) {

    if (r.pendingIntents != null) {
        deliverNewIntents(r, r.pendingIntents);
        r.pendingIntents = null;
    }
    if (r.pendingResults != null) {
        deliverResults(r, r.pendingResults);
        r.pendingResults = null;
    }
    //回调Activity::onResume
    r.activity.performResume();

}

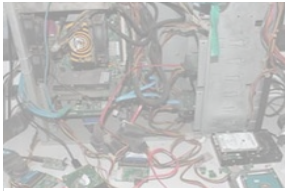
}
```

到这里，Activity从产生到初始化的过程就全部结束了。之后就是等待用户界面的消息，根据消息进行相应的处理。整个过程中，在ActivityManagerServer、ActivityThread的互相协作下构建出Activity，并在相应的时机回调Activity的相应接口，完成Activity的初始化。

顶 0 踩 1

关闭

app开发报价单



- 上一篇 Android的CursorAdapter与CursorFilter机制
- 下一篇 Android性能优化总结

相关文章推荐

- Activity启动流程
- Android A



免费云主机试用一年



硬盘数据恢复



- 【直播】大中型UGC信息网站SEO分享--乔向阳
- 想到做到：Android开发关键技术与精彩案例 (詹...)
- 【直播】打通Linux脉络 进程、线程和调度--宋宝华
- Activity的启动流程
- 【直播】Java最佳学习路线指导--肖海鹏
- 1：android5.1跨应用开启服务2：广播必须在activ...
- 【套餐】C++音视频实战技术套餐--夏曹俊
- 【套餐】0基础拿下HTML5和CSS3--李仁密
- 拦截Activity的项目
- 【套餐】机器学习之数学基础系列--AI100
- Activity启动流程番外篇
- activiti 工作流实例-实现请假流程
- Android源码基础解析之Activity启动流程
- 深入浅出Android



拓展魔鬼训练



猎头公司收费



美国波士顿房价



真实的达内



查看评论

6楼 Boolean布尔 2016-07-28 09:11发表



博主，我要给你生孩子

5楼 qluojieq 2014-10-23 10:19发表



怎么这么厉害，你是怎么做到的，这个知识能告诉我是怎么样领悟到的呀，看源码吗！

4楼 放风筝的骚年 2012-05-09 14:58发表



很好 很感谢

3楼 virus026 2011-08-19 10:25发表



I love you man!

2楼 quechao1234 2011-08-10 10:36发表



您好，感谢楼主的分享

我有一个问题：Activity启动时是如何读取AndroidManifest.xml中的相关信息的？如screenOrientation属性，是在哪个函数中完成的。

Re: bluedog 2011-08-26 09:35发表



回复quechao1234：AndroidManifest.xml 在apk安装时，PackageManagerService解析出所有AndroidManifest中列出的Activity以及信息，其中就包括screenOrientation属性。
解析的过程在scanPackageLI()中，得到的Activity信息保存在ActivityInfo中。
Activity启动时，ActivityManagerService会从PackageManagerService读取该信息，

1楼 forever_crying 2011-07-03 02:13发表



很感谢能作者把这篇文章分享出来

您还没有登录,请[登录](#)或[注册](#)

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

公司签约 | 诚邀合作 | 广告服务 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

客服 webmaster@csdn.net 400-660-0108 | 北京创新乐知信息技术有限公司 版权所有 | 江苏知之为计算机有限公司 |

© 1999-2017, CSDN.NET, All Rights Reserved



app开发报价单



免费云主机试用一年

