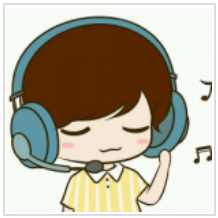
个人资料



Go稀

访问： 55134次
积分： 1279
等级： BLOG 4
排名： 第19344名

原创： 45篇
转载： 23篇
译文： 0篇
评论： 140条

Android Activity 、 Window 、 View之间的关系

2015-10-28 13:28 549人阅读 评论(0) 收藏 举报

分类： android 面试（7）

本想分析一下触摸事件的分发响应机制，但是发现分发事件的方法在Activity、View以及ViewGroup中各自存在，如图1表所示

方法名	View	ViewGroup	Activity
dispatchTouchEvent(MotionEvent ev)	有	有	有
onInterceptTouchEvent(MotionEvent ev)	没有	有	没有
onTouchEvent(MotionEvent ev)	有	有	有

图一

这样的话又牵扯到了三者之间的关系，那索性先理清楚Activity与另外两者的关系，在去分析触摸事件比较好。

什么是Activity 、View 、 Window?

Activity：是Android 四大组件之一， 是存放View对象的容器，也是我们界面的载体，可以用来展示一个界面。它有一个SetContentView（）方法，可以将我们定义的布局设置到界面上。

View：就是一个个视图的对象，实现了KeyEvent.Callback和Drawable.Callback。

Window：是一个抽象类，是一个顶层的窗口，它的唯一实例是PhoneWindow它提供标准的用户界面策略，如背景、标题、区域，默认按键处理等。

分析下三者之间的关系吧

View包含很多，TextView 、Imageview 、Listview 、 Button..就是一个一个展示不同图形的对象。我们可以把view通过xml布局，或者通过new View(),然后通过addview方法或动态或静态添加到Activity的布局上。我们都知道我们定义了layout布局，通过SetContentView就可以设置到Activity上，而Activity中的SetContentView()方法,又调用了Window的SetContentView方法，也就是View通过Activity最终添加到了Window上面。

那我们今天就看一下这个方法到底如何把layout布局加载进去，到底加载到哪里去了?

```java
/**
  * Set the activity content from a layout resource.  The resource will be
  * inflated, adding all top-level views to the activity.
  *
  * @param layoutResID Resource ID to be inflated.
  *
  * @see #setContentView(android.view.View)
  * @see #setContentView(android.view.View, android.view.ViewGroup.LayoutParams)
  */
 public void setContentView(@LayoutRes int layoutResID) {
     getWindow().setContentView(layoutResID);
     initWindowDecorActionBar();
 }
```

上面注释写的很清楚，通过一个layout资源给Activity设置内容，资源将被添加到Activity最顶层的View上也就是调用了方法体中的getWindow().set方法，首先这里getWindow() 拿到的是一个Window的子类，PhoneWindow的实例，那么这个Window对象是在哪里 赋值的呢，我们在Activity中找到attach方法如下所示:

```java
final void attach(Context context, ActivityThread aThread,
            Instrumentation instr, IBinder token, int ident,
            Application application, Intent intent, ActivityInfo info,
            CharSequence title, Activity parent, String id,
            NonConfigurationInstances lastNonConfigurationInstances,
            Configuration config, String referrer, IVoiceInteractor voiceInteractor) {
        attachBaseContext(context);

        mFragments.attachHost(null /*parent*/);

        mWindow = new PhoneWindow(this);
        mWindow.setCallback(this);
        mWindow.setOnWindowDismissedCallback(this);
        mWindow.getLayoutInflater().setPrivateFactory(this);
        if (info.softInputMode != WindowManager.LayoutParams.SOFT_INPUT_STATE_UNSPECIFIED)
            mWindow.setSoftInputMode(info.softInputMode);
        }
        if (info.uiOptions != 0) {
            mWindow.setUiOptions(info.uiOptions);
        }
        mUiThread = Thread.currentThread();

        mMainThread = aThread;
        mInstrumentation = instr;
        mToken = token;
        mIdent = ident;
        mApplication = application;
        mIntent = intent;
        mReferrer = referrer;
        mComponent = intent.getComponent();
        mActivityInfo = info;
        mTitle = title;
        mParent = parent;
        mEmbeddedID = id;
        mLastNonConfigurationInstances = lastNonConfigurationInstances;
        if (voiceInteractor != null) {
            if (lastNonConfigurationInstances != null) {
                mVoiceInteractor = lastNonConfigurationInstances.voiceInteractor;
            } else {
                mVoiceInteractor = new VoiceInteractor(voiceInteractor, this, this,
                        Looper.myLooper());
            }
        }

        mWindow.setWindowManager(
                (WindowManager)context.getSystemService(Context.WINDOW_SERVICE),
                mToken, mComponent.flattenToString(),
                (info.flags & ActivityInfo.FLAG_HARDWARE_ACCELERATED) != 0);
        if (mParent != null) {
```

```
50.             mWindow.setContainer(mParent.getWindow());
51.         }
52.         mWindowManager = mWindow.getWindowManager();
53.         mCurrentConfig = config;
54.     }
```

在第11行初始化mWindow对象，这个对象是window 接口的实现类 PhoneWindow 的实例。

那我们看一下PhoneWindow方法中的SetContentView方法代码如下所示：

```
[java]
01. @Override
02. public void setContentView(int layoutResID) {
03.         // Note: FEATURE_CONTENT_TRANSITIONS may be set in the process of installing the wi
04.        // decor, when theme attributes and the like are crystalized. Do not check the featu
05.        // before this happens.
06.         if (mContentParent == null) {
07.             installDecor();
08.        } else if (!hasFeature(FEATURE_CONTENT_TRANSITIONS)) {
09.            mContentParent.removeAllViews();
10.         }
11.
12.         if (hasFeature(FEATURE_CONTENT_TRANSITIONS)) {
13.             final Scene newScene = Scene.getSceneForLayout(mContentParent, layoutResID,
14.                     getContext());
15.             transitionTo(newScene);
16.        } else {
17.            mLayoutInflater.inflate(layoutResID, mContentParent);
18.        }
19.       final Callback cb = getCallback();
20.         if (cb != null && !isDestroyed()) {
21.             cb.onContentChanged();
22.         }
23.     }
```

这里我们只看下第6和第11行，首先判断mContentParent是不是 null，我们先搞明白mContentParent是什么东西?

```
[java]
01. // This is the top-level view of the window, containing the window decor.
02.    private DecorView mDecor;
03.
04.     // This is the view in which the window contents are placed. It is either
05.     // mDecor itself, or a child of mDecor where the contents go.
06.    private ViewGroup mContentParent;
```

通过查看源代码 我们这里知道mContentParent是一个ViewGroup对象 ，上面的注释我们可以明白这个Window中的内容就放置在mContentParent上面， 这个mContentParent或者是一个DecorView对象或者是一个DecorView的子类。

OK，搞明白了mContentParent是一个ViewGroup对象 ，那我们继续往下看

如果是installDecor()不用想我们也知道这个方法肯定是初始化了mContentParent，一起看下是不是我们想的那样吧。

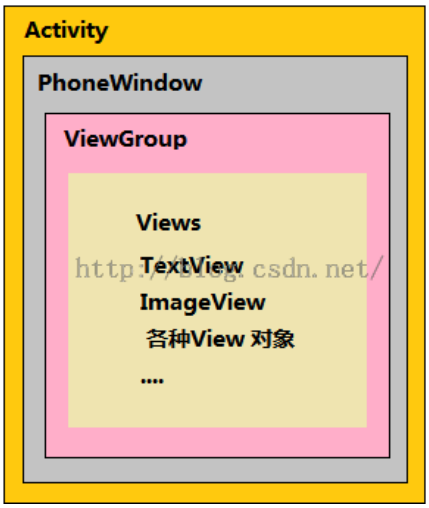
```
[java]
01. private void installDecor() {
02.         if (mDecor == null) {
03.             mDecor = generateDecor();
04.             mDecor.setDescendantFocusability(ViewGroup.FOCUS_AFTER_DESCENDANTS);
05.             mDecor.setIsRootNamespace(true);
06.             if (!mInvalidatePanelMenuPosted && mInvalidatePanelMenuFeatures != 0) {
07.                 mDecor.postOnAnimation(mInvalidatePanelMenuRunnable);
```

```
08.                 }
09.             }
10.             if (mContentParent == null) {
11.                 mContentParent = generateLayout(mDecor);
12.
13.                 // Set up decor part of UI to ignore fitsSystemWindows if appropriate.
14.             //.....
15.             //...... 省略若干代码
16.             }
17.         }
```

这里先判断 mDecor是不是null，如果是，初始化mDecor，然后判断mContentParent是不是null，如果是，通过mDecor去初始化 mContentParent对象。 对吧，跟我们想的一样是去初始化。

OK，这里创建出了mContentParent对象，我们接着看PhoneWindow的SetContentView方法的第11行，这里先进行了判断，具体判断 我们先不关心，我们继续往下执行在看第12行或者17行，我们就清楚了我们在Activity中设置的layoutid 在这里加载到了mContentParent 上面。也就是所有的所有的View 对象都是加载到了mContentParent对象上面，而我们前面知道mContentParent是根据DecorView而来的，这样我们就清楚了Activity与Window以及View的关系，这里用图2 表示一下他们的关系。

看图识关系

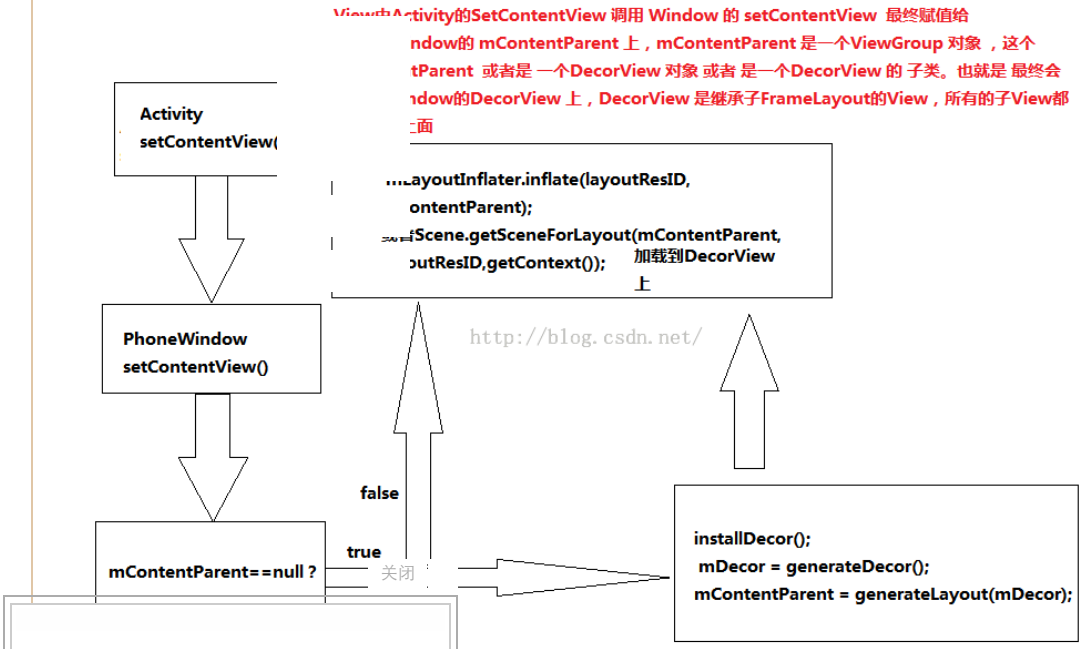


图二

对着这张图，打个比喻来帮助理解。

Activity就像是一扇贴着窗花的窗口，Window就想上窗口上面的玻璃，而View对象就像一个个贴在玻璃上的窗花。

最后的Activity与Window View的关联在画一个图3：



ndow的setContentView()将layout布局添加到DecorView上，而此
然后通过LayoutInflater.infalte()方法加载布局生成View对象并通过
一层的叠加到Window上）所以，Activity其实不是显示视图，

一个Window(PhoneWindow)，另外这个PhoneWindow有一个View
容器 mContentParent，这个
View容器是一个ViewGroup，是最初始的跟视图，然后通过addView方法将View一个个层叠
到mContentParent上，这些层叠的View最终放在Window这个载体上面。

OK 理清了Window Activity与View 的关系，接下来我们分析触摸事件的分发也会容易理解一些。

对于本文表达不到位，或者理解不到位的地方，欢迎朋友指正，感兴趣的朋友可以继续看下一篇Touch事件的分发响应： http://blog.csdn.net/u011733020/article/details/49452109。

----------欢迎爱学习的小伙伴加群
----------android交流群：230274309
----------一起分享，一起进步！需要你们
----------期待各位爱学习的小伙伴们的到来

上一篇 *Android开发中使用到的知识
下一篇 No resource found that matches the given name 'Theme.AppCompat.Light 的完美解决方案

我的同类文章

android 面试（7）

- *易到用车面试总结（android）
- Google发布Android 性能优化典范
- 看完这篇文章，你就了解了Android Handler的一切
- Android Touch事件的分发响应机制
- *《史上最详细 最基础的 android 面试 知识点总结》
- *Android面试实战总结2
- *Android面试实战总结

主题推荐 android

猜你在找

Android杂谈--ActivityWindowView的关系
转Android杂谈--ActivityWindowView的关系
Android杂谈--ActivityWindowView的关系
Android杂谈--ActivityWindowView的关系
Android 里面ActivityWindowView的关系

查看评论

暂无评论

发表评论

用 户 名: scboyhj__

评论内容:

提交

* 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

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenStack
VPN Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery
BI HTML5 Spring Apache .NET API HTML SDK IIS Fedora XML LBS Unity
Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra CloudStack
FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App SpringSide Maemo
Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP HBase Pure Solr
Angular Cloud Foundry Redis Scala Django Bootstrap