[Android中Activity的setContentView方法分析](https://segmentfault.com/a/1190000040838662)

[[](https://segmentfault.com/u/xiaoer_5f117772efe32)**公众号小二玩编程**](https://segmentfault.com/u/xiaoer_5f117772efe32)发布于 2021-10-20

PS：本文系转载文章，阅读原文可读性会更好，文章末尾有原文链接

ps：源码是基于 android api 27 来分析的

这一篇我们分析 Activity 的 setContentView 方法到底做了什么事情，有的读者可能心存疑虑，文章有的地方看不懂怎么办，之前我看文章的时候也是有这样的疑虑，目前我采取的办法有2种：（1）看不懂的地方可以先跳过，看完本篇文章后再 google 一下看不懂的地方；（2）看不懂的地方先 google 一下，再继续把这篇文章往下看。好了，言归正传，我们来看 Activity 的 setContentView 方法；

public void setContentView(@LayoutRes int layoutResID) {

getWindow().setContentView(layoutResID);

initWindowDecorActionBar();

}

这里的 getWindow 方法拿到的是 Window 对象，Window 是显示顶层窗口的外观，对一些 findViewById、事件分发等基础的行为进行封装，每一个 Window 都会被添加到 WindowManager 里面，Window 唯一的实现类是 PhoneWindow，所以 getWindow 方法拿到的是 Window 对象本质是 PhoneWindow 对象；如何知道 Window 唯一的实现类是 PhoneWindow 呢？可以从Android中View事件的分发第一篇这里找到答案，我们来看 PhoneWindow 的 setContentView 方法；

|  |
| --- |
| @Override  public void setContentView(int layoutResID) {  // Note: FEATURE\_CONTENT\_TRANSITIONS may be set in the process of installing the window  // decor, when theme attributes and the like are crystalized. Do not check the feature  // before this happens.  if (mContentParent == null) {  //1、  installDecor();  } else if (!hasFeature(FEATURE\_CONTENT\_TRANSITIONS)) {  mContentParent.removeAllViews();  }  if (hasFeature(FEATURE\_CONTENT\_TRANSITIONS)) {  final Scene newScene = Scene.getSceneForLayout(mContentParent, layoutResID,  getContext());  transitionTo(newScene);  } else {  //2、  mLayoutInflater.inflate(layoutResID, mContentParent);  }  mContentParent.requestApplyInsets();  final Window.Callback cb = getCallback();  if (cb != null && !isDestroyed()) {  //3、  cb.onContentChanged();  }  //4、  mContentParentExplicitlySet = true;  } |

注释3 表示回调 Activity 的 onContentChanged 方法；注释4 表示已经设置过布局，mContentParentExplicitlySet = false 的时候已经调用过 PhoneWindow 的 requestFeature 方法了，mContentParentExplicitlySet = true 的时候不可以再调用 PhoneWindow 的 requestFeature 方法，我们看 PhoneWindow 的 requestFeature 方法；

|  |
| --- |
| @Override  public boolean requestFeature(int featureId) {  if (mContentParentExplicitlySet) {  throw new AndroidRuntimeException("requestFeature() must be called before adding content");  }  ......  } |

真的像上面所说的那样，注释1 表示实例化 DecorView 并安装，我们看一下 installDecor 方法具体实现；

private void installDecor() {

mForceDecorInstall = false;

if (mDecor == null) {

//5、

mDecor = generateDecor(-1);

......

} else {

mDecor.setWindow(this);

}

if (mContentParent == null) {

//6、

mContentParent = generateLayout(mDecor);

......

}

}

注释5 中的 mDecor 是一个 DecorView 对象，也是我们 Activity 最顶层的视图，它的父类是 FrameLayout，我们来看看 PhoneWindow 的 generateDecor 方法是如何创建 DecorView 对象的；

protected DecorView generateDecor(int featureId) {

......

return new DecorView(context, featureId, this, getAttributes());

}

直接 new 一个 DecorView 并将它返回，我们回到 installDecor 方法注释6 的代码，也就是 PhoneWindow 的 generateLayout 方法；

protected ViewGroup generateLayout(DecorView decor) {

//7、

TypedArray a = getWindowStyle();

......

//8、

mIsFloating = a.getBoolean(R.styleable.Window\_windowIsFloating, false);

int flagsToUpdate = (FLAG\_LAYOUT\_IN\_SCREEN|FLAG\_LAYOUT\_INSET\_DECOR)

& (~getForcedWindowFlags());

if (mIsFloating) {

setLayout(WRAP\_CONTENT, WRAP\_CONTENT);

setFlags(0, flagsToUpdate);

} else {

setFlags(FLAG\_LAYOUT\_IN\_SCREEN|FLAG\_LAYOUT\_INSET\_DECOR, flagsToUpdate);

}

//9、

if (a.getBoolean(R.styleable.Window\_windowNoTitle, false)) {

requestFeature(FEATURE\_NO\_TITLE);

} else if (a.getBoolean(R.styleable.Window\_windowActionBar, false)) {

// Don't allow an action bar if there is no title.

requestFeature(FEATURE\_ACTION\_BAR);

}

......

//10、

mIsTranslucent = a.getBoolean(R.styleable.Window\_windowIsTranslucent, false);

......

//11、

int layoutResource;

//12、

int features = getLocalFeatures();

// System.out.println("Features: 0x" + Integer.toHexString(features));

if ((features & (1 << FEATURE\_SWIPE\_TO\_DISMISS)) != 0) {

......

} else if ((features & ((1 << FEATURE\_LEFT\_ICON) | (1 << FEATURE\_RIGHT\_ICON))) != 0) {

......

} else if ((features & ((1 << FEATURE\_PROGRESS) | (1 << FEATURE\_INDETERMINATE\_PROGRESS))) != 0

......

} else if ((features & (1 << FEATURE\_CUSTOM\_TITLE)) != 0) {

......

} else if ((features & (1 << FEATURE\_NO\_TITLE)) == 0) {

......

} else if ((features & (1 << FEATURE\_ACTION\_MODE\_OVERLAY)) != 0) {

layoutResource = R.layout.screen\_simple\_overlay\_action\_mode;

} else {

// Embedded, so no decoration is needed.

//13、

layoutResource = R.layout.screen\_simple;

// System.out.println("Simple!");

}

mDecor.startChanging();

//14、

mDecor.onResourcesLoaded(mLayoutInflater, layoutResource);

ViewGroup contentParent = (ViewGroup)findViewById(ID\_ANDROID\_CONTENT);

......

return contentParent;

}

注释7 表示获取当前 Window 的 Style 属性；注释8 表示是否是悬浮类型的 Window；注释9 表示是否需要标题栏，后面省略了很多 a.getBoolean 的 if 语句，而 if 里的 requestFeature 方法调用其实是对 Window 的一些状态进行设置，requestFeature 方法必须在 Activity 的内容布局加载出来之前先被调用；注释10 表示 Window 是否透明的，默认情况下不是透明的；注释11 表示 DecorView 要加载 xml 文件的 id；注释12 表示获取 Window 的 Feature 属性，后面有很多 features 的 if 判断，其实是根据 features 得到 DecorView 要加载相应的 xml 文件的 id；注释13 只是 DecorView 要加载 xml 文件时的其中一种，我们来看看 screen\_simple.xml 的布局；

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:fitsSystemWindows="true"

android:orientation="vertical">

<ViewStub android:id="@+id/action\_mode\_bar\_stub"

android:inflatedId="@+id/action\_mode\_bar"

android:layout="@layout/action\_mode\_bar"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:theme="?attr/actionBarTheme" />

<FrameLayout

android:id="@android:id/content"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:foregroundInsidePadding="false"

android:foregroundGravity="fill\_horizontal|top"

android:foreground="?android:attr/windowContentOverlay" />

</LinearLayout>

我们来看一下注释14 的代码，也就是 DecorView 的 onResourcesLoaded 方法；

|  |
| --- |
| void onResourcesLoaded(LayoutInflater inflater, int layoutResource) {  ......  mDecorCaptionView = createDecorCaptionView(inflater);  final View root = inflater.inflate(layoutResource, null);  //15、  if (mDecorCaptionView != null) {  if (mDecorCaptionView.getParent() == null) {  addView(mDecorCaptionView,  new ViewGroup.LayoutParams(MATCH\_PARENT, MATCH\_PARENT));  }  mDecorCaptionView.addView(root,  new ViewGroup.MarginLayoutParams(MATCH\_PARENT, MATCH\_PARENT));  //16、  } else {  // Put it below the color views.  addView(root, 0, new ViewGroup.LayoutParams(MATCH\_PARENT, MATCH\_PARENT));  }  ......  } |

注释15 的代码表示如果有标题，然后再判断 mDecorCaptionView.getParent() 是否为空，如果为空就把 mDecorCaptionView 添加到 DecorView 里，如果不为空，那么 mDecorCaptionView 已经添加过到 DecorView 里，最后直接将 root 添加到 mDecorCaptionView 里；注释 16 表示无标题，就将 root 直接添加到 DecorView 里。如何判断注释15 的蓝色描述的文字是否是正确的呢？我们可以看 创建 mDecorCaptionView 的 DecorView.createDecorCaptionView 方法；

|  |
| --- |
| private DecorCaptionView createDecorCaptionView(LayoutInflater inflater) {  DecorCaptionView decorCaptionView = null;  for (int i = getChildCount() - 1; i >= 0 && decorCaptionView == null; i--) {  View view = getChildAt(i);  if (view instanceof DecorCaptionView) {  // The decor was most likely saved from a relaunch - so reuse it.  //17、  decorCaptionView = (DecorCaptionView) view;  removeViewAt(i);  }  }  ......  if (!mWindow.isFloating() && isApplication && StackId.hasWindowDecor(mStackId)) {  // Dependent on the brightness of the used title we either use the  // dark or the light button frame.  if (decorCaptionView == null) {    //18、  decorCaptionView = inflateDecorCaptionView(inflater);  }  decorCaptionView.setPhoneWindow(mWindow, true /\*showDecor\*/);  } else {  decorCaptionView = null;  }  ......  return decorCaptionView;  } |

注释17 中将 View 强制为 DecorCaptionView，此时的 View 正是 DecorView 的子 View，证明上面所说的 mDecorCaptionView 已经添加过到 DecorView 里；看注释18 的代码，也就是 DecorView 的 inflateDecorCaptionView 方法；

private DecorCaptionView inflateDecorCaptionView(LayoutInflater inflater) {

final Context context = getContext();

// We make a copy of the inflater, so it has the right context associated with it.

inflater = inflater.from(context);

final DecorCaptionView view = (DecorCaptionView) inflater.inflate(R.layout.decor\_caption,

null);

setDecorCaptionShade(context, view);

return view;

}

看 inflateDecorCaptionView 方法中的 DecorCaptionView 创建，传入一个参数 null，这里创建的 DecorCaptionView 对象的 getParent() 就为空了。

假设 DecorView 要加载的 xml 文件是 screen\_simple.xml，那么我们回过头来看看 DecorView 的 installDecor 方法中的注释6 代码，发现 mContentParent 就是 screen\_simple.xml 中 id 为 content 的 FrameLayout；我们再回看 PhoneWindow 的 setContentView 方法中的注释2 的代码，发现我们 Activity 设置的内容布局文件是添加到了 screen\_simple.xml 中 id 为 content 的 FrameLayout；