# **从源码角度分析，为什么会发生Fragment重叠？**

该文分析的support包版本为23.3.0，在24.0.0及以上官方已修复文章中所说的Fragment重叠BUG。

我们在使用Fragment的过程中，有时会发现一直表现正常的Fragment，突然重叠了！

## 什么情况下会发生Fragment重叠？

一般满足下面2个条件才可能会发生重叠：

1、发生了页面重启（旋转屏幕、内存不足等情况被强杀重启）。  
2、重复replace｜add Fragment 或者 使用show , hide控制Fragment；

## 为什么会发生Fragment重叠？

##### 从源码角度分析，为什么发生页面重启后会导致重叠？（在以add方式加载Fragment的时候）

我们知道Activity中有个onSaveInstanceState()方法，该方法在app进入后台、屏幕旋转前、跳转下一个Activity等情况下会被调用，此时系统帮我们保存一个Bundle类型的数据，我们可以根据自己的需求，手动保存一些例如播放进度等数据，而后如果发生了页面重启，我们可以在onRestoreInstanceState()或onCreate()里get该数据，从而恢复播放进度等状态。

而产生Fragment重叠的原因就与这个保存状态的机制有关，**大致原因就是系统在页面重启前，帮我们保存了Fragment的状态，但是在重启后恢复时，视图的可见状态没帮我们保存，而Fragment默认的是show状态，所以产生了Fragment重叠现象。**

**分析：**  
我们先看FragmentActivity的相关源码：

public class FragmentActivity extends ... {

final FragmentController mFragments = FragmentController.createController(new HostCallbacks());

protected void onCreate(@Nullable Bundle savedInstanceState) {

...省略

if (savedInstanceState != null) {

Parcelable p = savedInstanceState.getParcelable(FRAGMENTS\_TAG);

mFragments.restoreAllState(p, nc != null ? nc.fragments : null);

}

}

@Override

protected void onSaveInstanceState(Bundle outState) {

super.onSaveInstanceState(outState);

Parcelable p = mFragments.saveAllState();

...省略

}

}

从上面源码可以看出，FragmentActivity确实是帮我们保存了Fragment的状态，并且在页面重启后会帮我们恢复！

其中的mFragments是FragmentController，它是一个Controller，内部通过FragmentHostCallback间接控制FragmentManagerImpl。  
相关代码如下：

public class FragmentController {

private final FragmentHostCallback<?> mHost;

public Parcelable saveAllState() {

return mHost.mFragmentManager.saveAllState();

}

public void restoreAllState(Parcelable state, List<Fragment> nonConfigList) {

mHost.mFragmentManager.restoreAllState(state, nonConfigList);

}

}

public abstract class FragmentHostCallback<E> extends FragmentContainer {

final FragmentManagerImpl mFragmentManager = new FragmentManagerImpl();

}

通过上面代码可以看出FragmentController通过FragmentHostCallback里的FragmentManagerImpl对象来控制恢复工作。

我们接着看FragmentManagerImpl到底做了什么：

final class FragmentManagerImpl extends FragmentManager {

Parcelable saveAllState() {

...省略 详细保存过程

FragmentManagerState fms = new FragmentManagerState();

fms.mActive = active;

fms.mAdded = added;

fms.mBackStack = backStack;

return fms;

}

void restoreAllState(Parcelable state, List<Fragment> nonConfig) {

// 恢复核心代码

FragmentManagerState fms = (FragmentManagerState)state;

FragmentState fs = fms.mActive[i];

if (fs != null) {

Fragment f = fs.instantiate(mHost, mParent);

｝

}

}

我们通过saveAllState()看到了关键的保存代码，原来是是通过FragmentManagerState来保存Fragment的状态、所处Fragment栈下标、回退栈状态等。

而在restoreAllState()恢复时，通过FragmentManagerState里的FragmentState的instantiate()方法恢复了Fragment（见下面的分析就明白啦）

我们看下FragmentManagerState：

final class FragmentManagerState implements Parcelable {

FragmentState[] mActive; // Fragment状态

int[] mAdded; // 所处Fragment栈下标

BackStackState[] mBackStack; // 回退栈状态

...

}

我们只看FragmentState，它也实现了Parcelable，保存了Fragment的类名、下标、id、Tag、ContainerId以及Arguments等数据：

final class FragmentState implements Parcelable {

final String mClassName;

final int mIndex;

final boolean mFromLayout;

final int mFragmentId;

final int mContainerId;

final String mTag;

final boolean mRetainInstance;

final boolean mDetached;

final Bundle mArguments;

...

// 在FragmentManagerImpl的restoreAllState()里被调用

public Fragment instantiate(FragmentHostCallback host, Fragment parent) {

...省略

mInstance = Fragment.instantiate(context, mClassName, mArguments);

}

}

至此，我们就明白了系统帮我们保存的Fragment其实最终是以FragmentState形式存在的。

此时我们再思考下**为什么在页面重启后会发生Fragment的重叠？ 其实答案已经很明显了，根据上面的源码分析，我们会发现FragmentState里没有Hidden状态的字段！**

而Hidden状态对应Fragment中的mHidden，该值默认false...

public class Fragment ... {

boolean mHidden;

}

我想你应该明白了，在以add方式加载Fragment的场景下，系统在恢复Fragment时，mHidden＝false，即show状态，这样在页面重启后，Activity内的Fragment都是以show状态显示的，而如果你不进行处理，那么就会发生Fragment重叠现象！

##### 为什么重复replace|add Fragment 或者 使用show , hide控制Fragment会导致重叠？

**重复**replace**|**add **Fragment**   
我们知道加载Fragment有2种方式：replace()和add()。  
不管哪种方式，重复加载Fragment都会导致重叠，这个很好理解，你加载同一个Fragment2次当然会重叠了；问题是我们在哪里重复加载了Fragment？  
一般情况下，我们会在Activity的onCreate()里或者Fragment的onCreateView()里加载根Fragment，如果在这里没有进行页面重启的判断的话，就可能导致重复加载Fragment引起重叠，正确的写法应该是：

@Override

protected void onCreate(@Nullable Bundle savedInstanceState) {

...

// 判空， Fragment同理

if(saveInstanceState == null){

// 这里replace或add 根Fragment

}

}

这里一定要在saveInstanceState==null时才加载Fragment，因为经过上面的分析，在页面重启时，Fragment的状态会被保存恢复，而此时再加载Fragment会重复加载，就导致栈已经有该Fragment的情况下，再加载一该Fragment，从而导致重叠！

**使用**show **,** hide**控制Fragment**  
我们使用show(),hide()时，都是使用add的方式加载Fragment的，add配合hide使Fragment的视图改变为GONE状态；而replace是销毁Fragment 的视图。  
页面重启时，add的Fragment会全部走生命周期，创建视图；而replace的非栈顶Fragment不会走生命周期，只有Back时，才会逐一走栈顶Fragment生命周期，创建视图。

结合上面的源码分析，在使用replace加载Fragment时，页面重启后，Fragment视图都还没创建，所以mHidden没有意义，不会发生重叠现象；  
而在使用add加载时，视图是存在的并且叠加在一起，页面重启后 mHidden=false，所有的Fragment都会是show状态显示出来（即VISIBLE），从而造成了Fragment重叠！