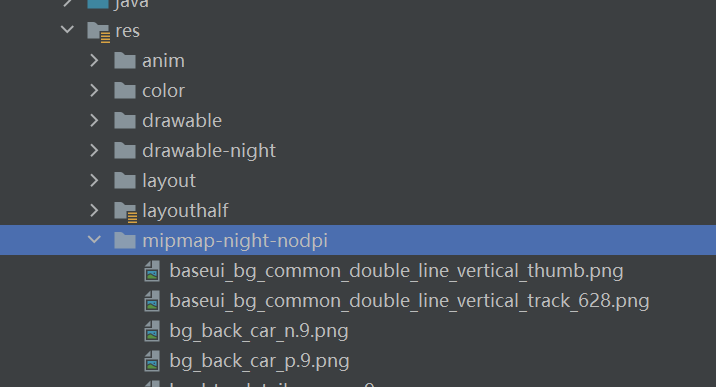
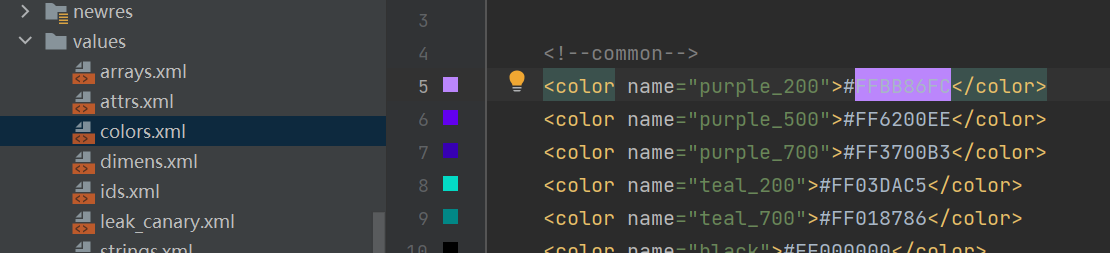
Android换肤主要包括图片切换和颜色切换

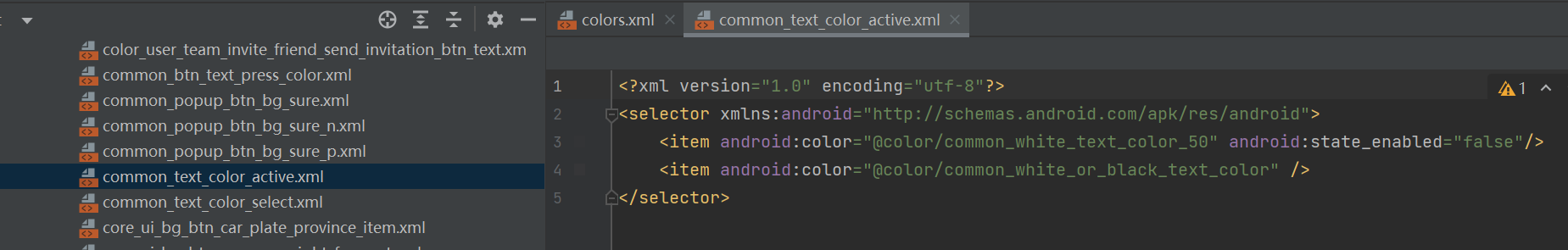
图片资源drawable（一般放置在drawable、mipmap等目录）



颜色资源color（一般放置在values目录下的xml中）



带状态的颜色资源（一般位于drawable）

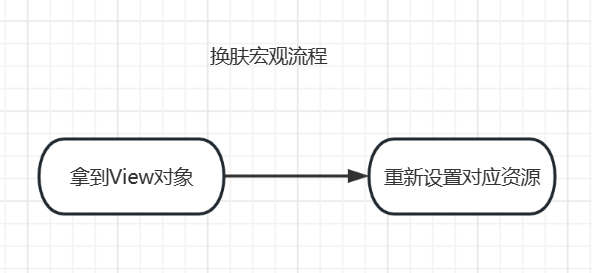


切换皮肤需要达到的效果：图片改变，文字颜色改变

要想达到这个效果，只能通过调用控件的API来更改（如setBackground、setTextColor）

所以、换肤的流程就是：在要需要换肤的时候，调用需要换肤的View的API，重新设置图片和颜色。

流程中的重点：



1. 在换肤时能拿到View
2. 拿到View了后知道该设置哪个图片哪个颜色

如何获取View

View的生成方式：

1. 直接在代码中new一个View

TextView tv = new AppCompatTextView(context);

1. 解析布局文件生成View

View view = LayoutInflater.*from*(context).inflate(layoutId, parent, true);

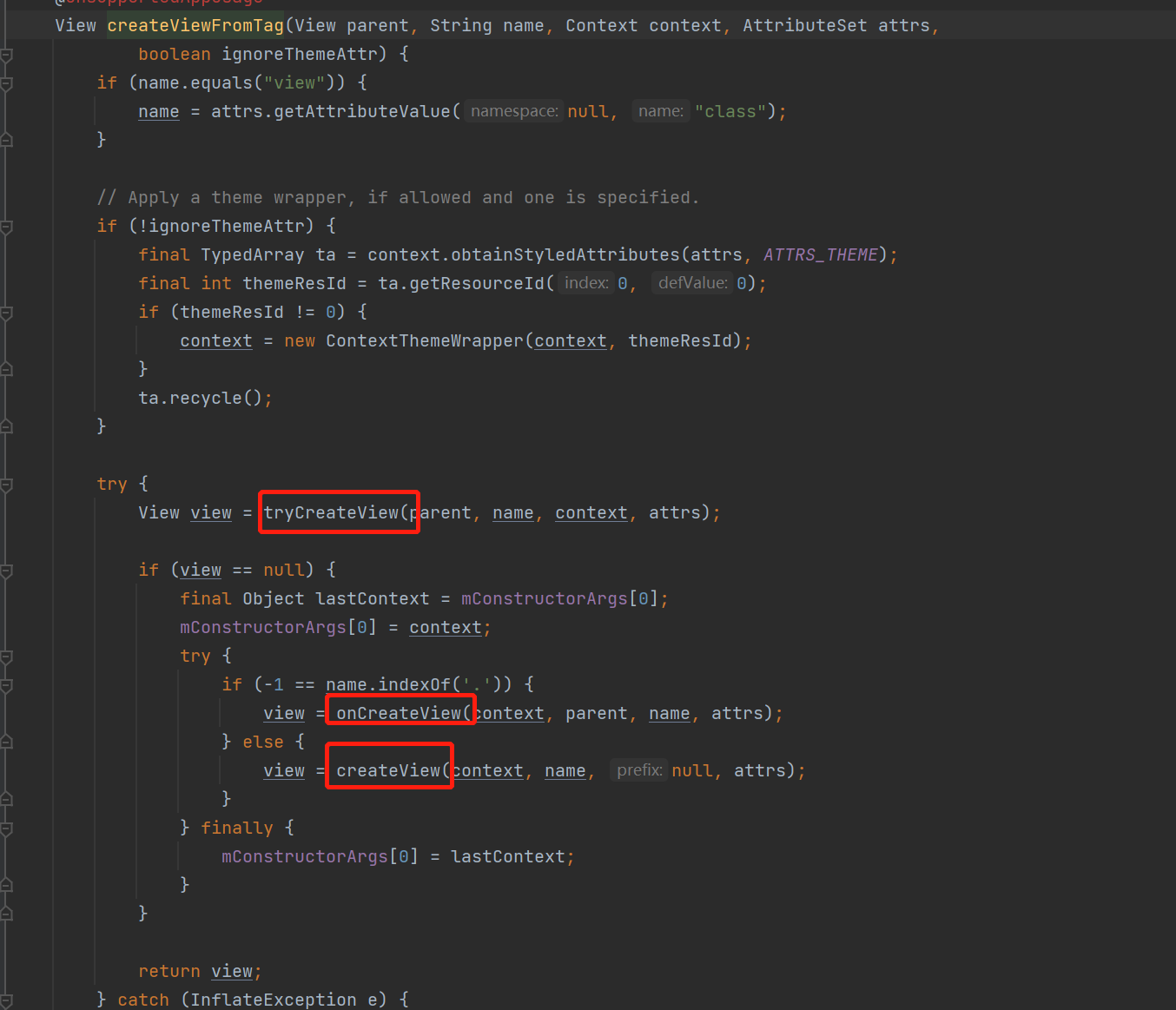
如何拿到LayoutInflater生成的View

LayoutInflater生成View的流程：



先判断释是否有预编译生成的View，没有则走普通解析xml的方式。

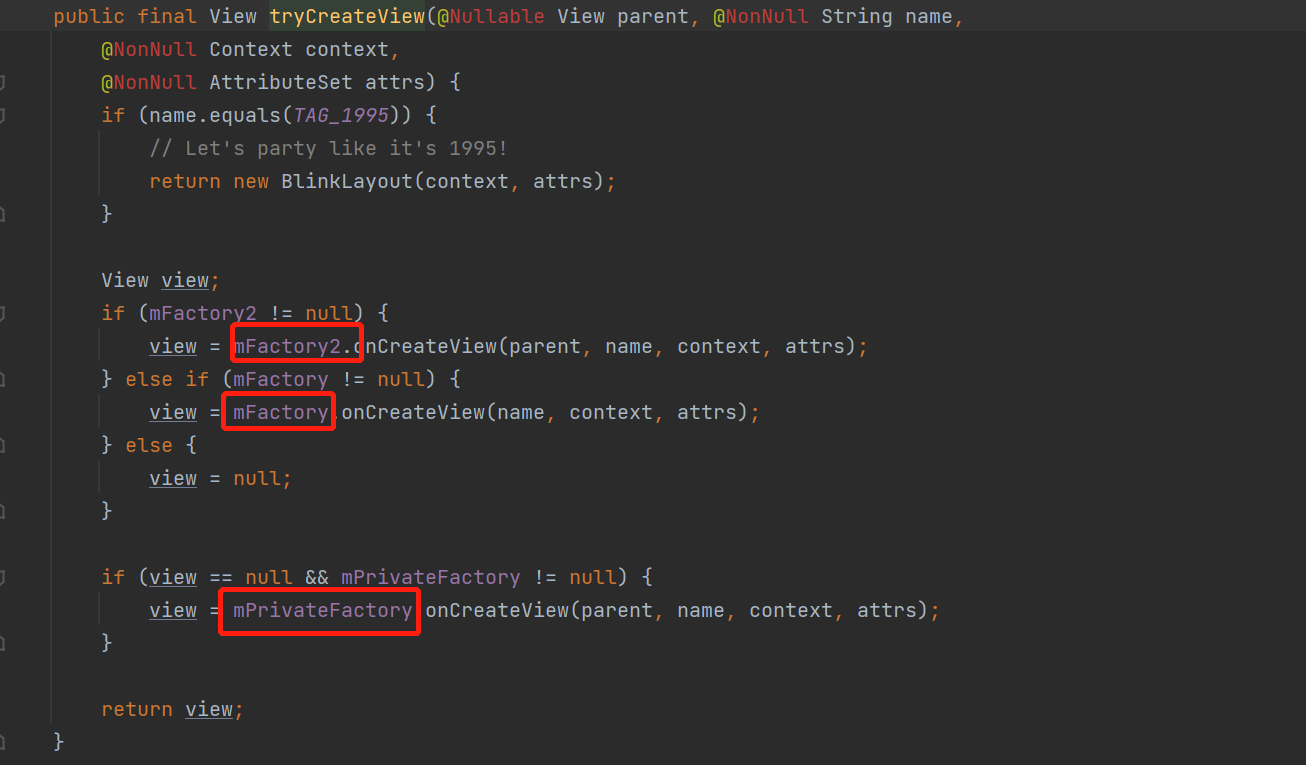
Inflate方法最终还是会走到createViewFromTag



其中关注3个方法：

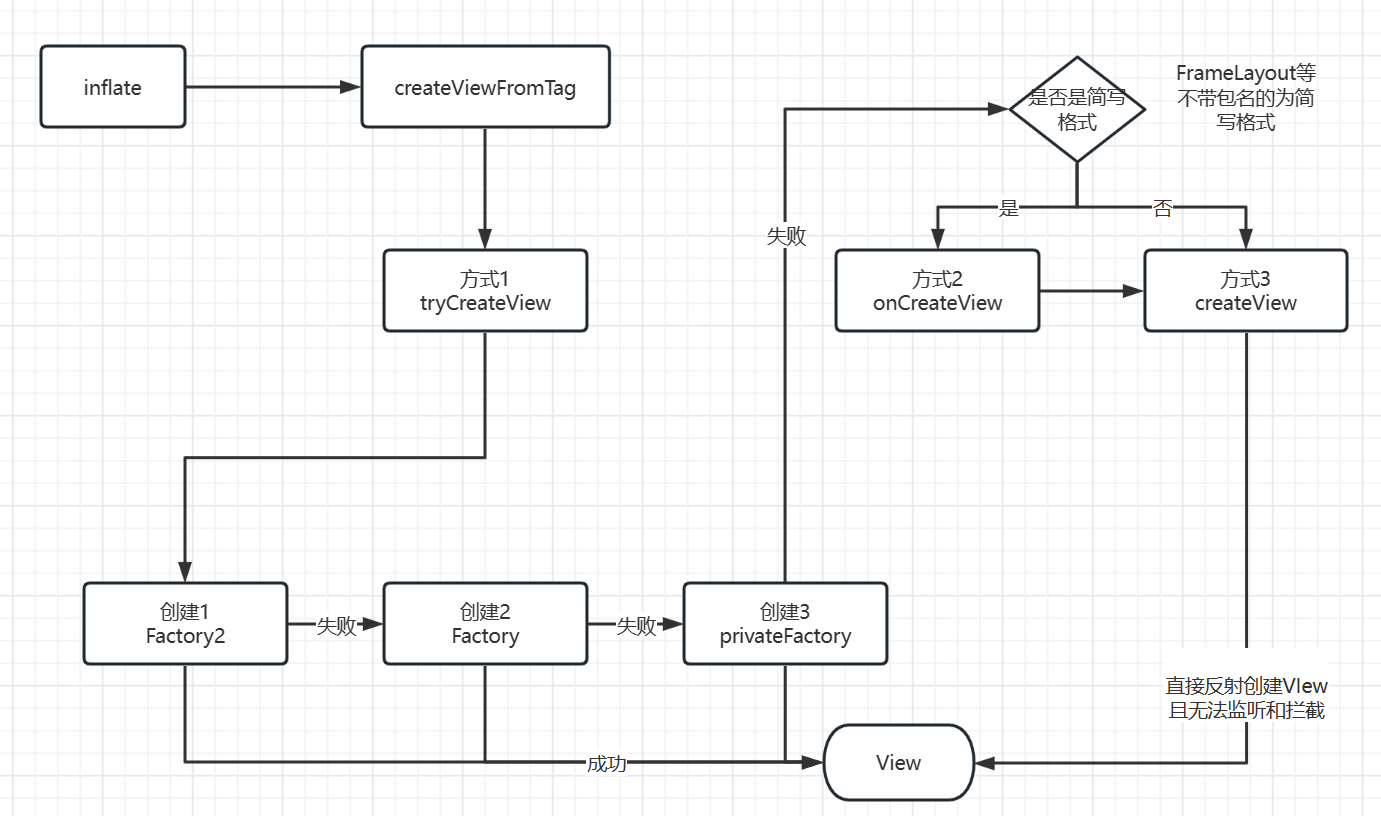
tryCreateView、onCreateView、createView

tryCreateView



其中是又通过Factory2、Factory以及privateFactory来创建视图对象，同时这些Factory都能在外部进行设置，理论上设置了自己的Factory就能监听所有的View的创建。

Inflate创建View总结：

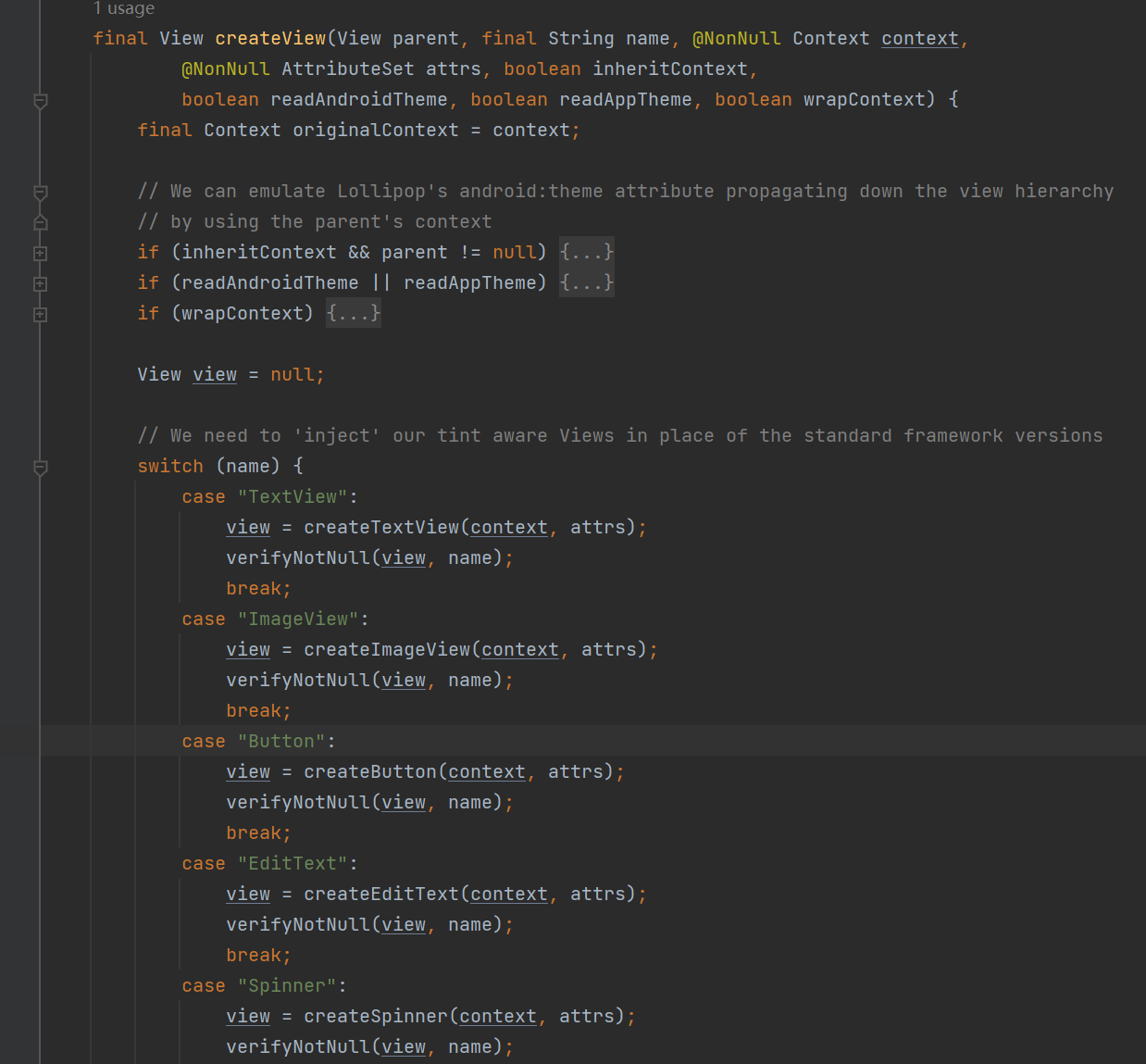


开发中两个比较重要的Factory2

AppCompatDelegateImpl，内部实现系统控件转换

系统中老版本控件转换为AppCompatXXX就是通过设置Factory2来实现的（在super.onCreate阶段设置）

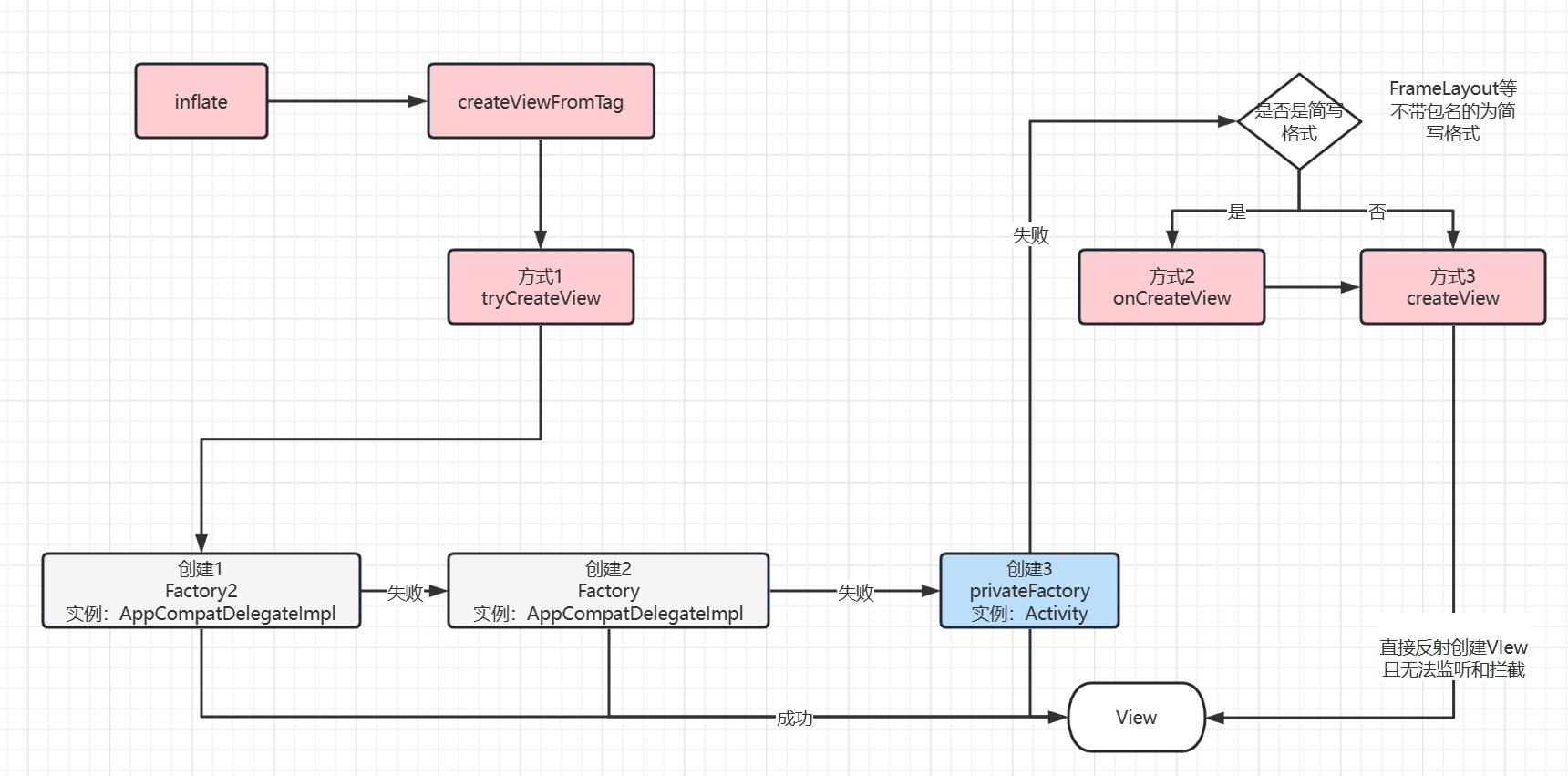
androidx.appcompat.app.AppCompatViewInflater.java



FragmentLayoutInflaterFactory

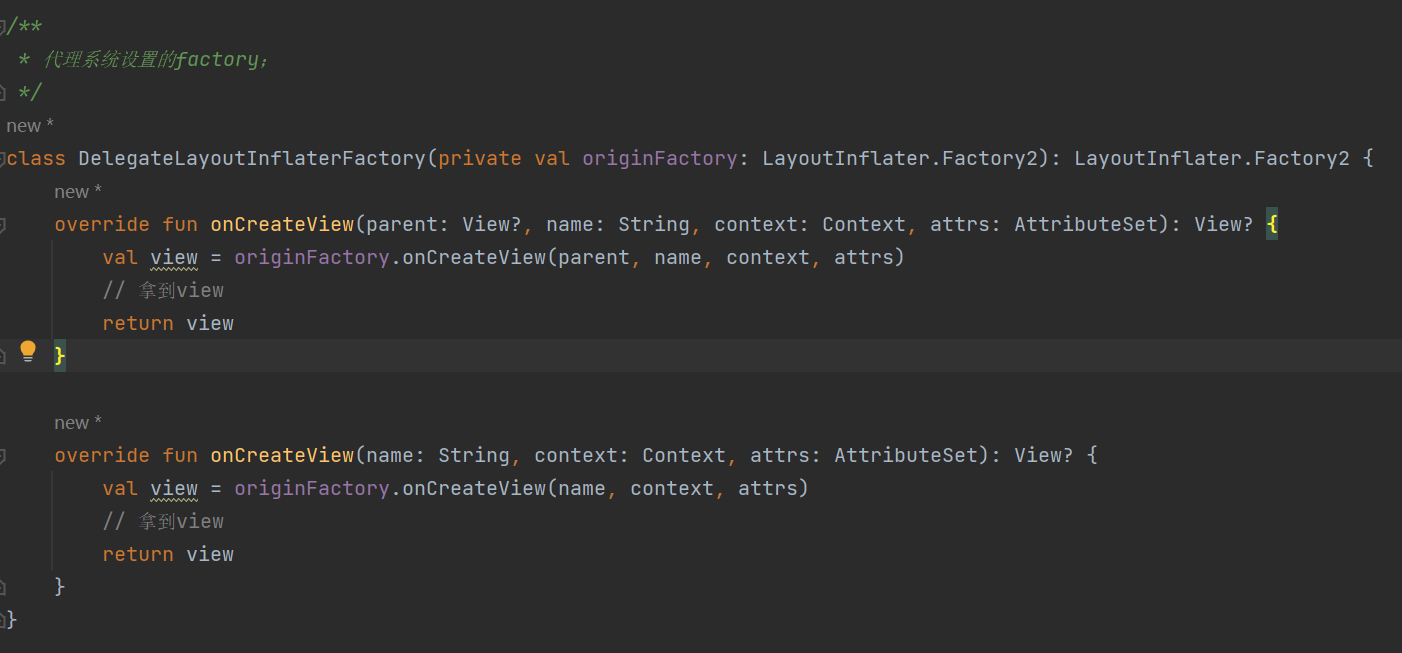
只负责解析<fragment>和<androidx.fragment.app.FragmentContainerView>标签，在FragmentActivity.onCreateView中间接被调用。

Activity，Activity也实现了Factory2接口，Activity实例会赋值给LayoutInflater.mPrivateFactory, 在Activity.attach中设置的。



如何拿到View

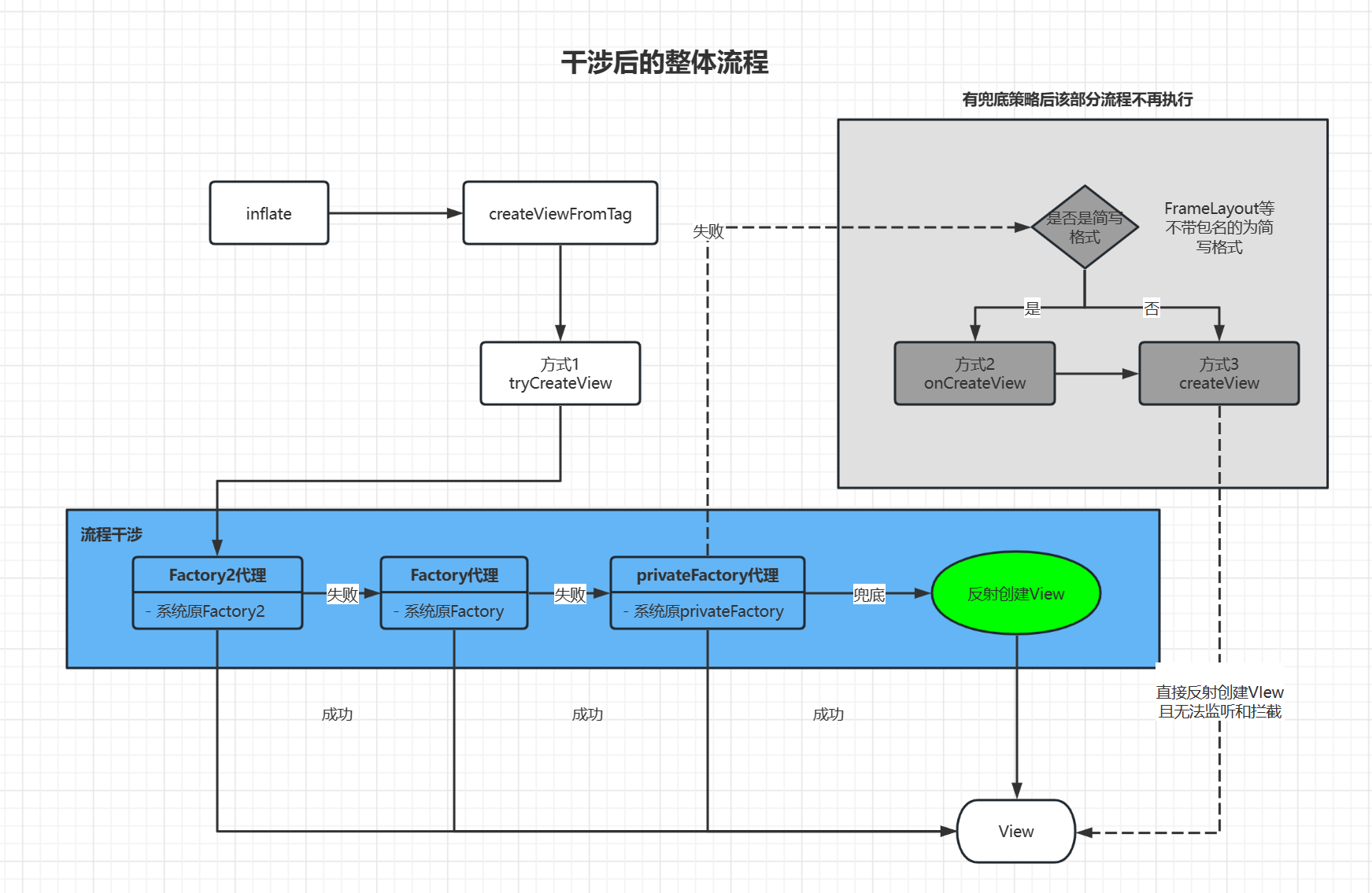
为了不破坏系统自带的Factory和Factory2的功能，可以通过设置代理Factory的方式来拦截



通过反射拿到系统设置的Factory，然后设置为自己的Factory

当代理的3种Factory都无法创建对象时，需要通过反射创建View，此时不能走系统反射创建View，因为后续流程无法拦截，无法得到系统反射创建的View。

所以需要进行兜底，我们自己来完成反射创建View的流程

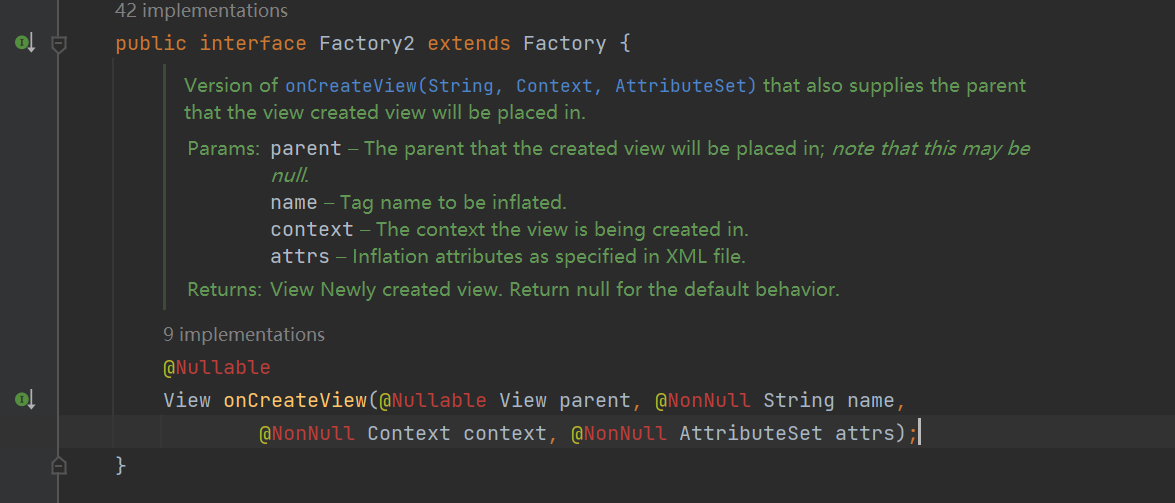


生成资源映射

通过干涉布局解析，拿到了所有的View，下一步就是要调用View的API设置背景颜色、背景图片、文本颜色等，但是具体设置什么颜色、什么图片呢？

干涉布局解析时记录View的各种属性

在Factory2或者Factory中onCreateView有一个AttributeSet参数，里面记录了view的各个属性



从AttributeSet获取所需属性

此处添加获取属性的方式和代码

完成了View属性解析和记录后，换肤的宏观流程就结束了。

皮肤包资源生成和读取

加载皮肤包方式：

要加载皮肤包，那么皮肤包中得有androidManifest.xml、resources.arsc、以及各种资源

皮肤包的本质是一个apk文件，可以删除apk文件中的二进制代码文件，不影响加载。

生成Resource对象

实际开发中的遇到的问题

换肤控件需要提供的额外功能

1. 支持动态添加和移除View的换肤功能
2. 方便开发