转载请注明链接：https://blog.csdn.net/feather_wch/article/details/79900081

本文详细分析从XML创建View的原理

通过XML创建View的原理

版本：2018/8/27-1

原理

本质原理：

1. Activity是通过 `Factory` 进行View的创建
2. 自定义 `Factory` 就能拦截创建过程，创建自己的 `View`

OnCreate流程

1、AppCompatActivity的OnCreate流程

```
//AppCompatActivity.java
protected void onCreate(@Nullable Bundle savedInstanceState) {
    final AppCompatDelegate delegate = getDelegate();
    //1. 初始化LayoutInflater，并且设置过Factory(没有设置过就新建)
    delegate.installViewFactory();
    //2. 执行正常的onCreate流程
    delegate.onCreate(savedInstanceState);
    //xxx
    super.onCreate(savedInstanceState);
}
//AppCompatDelegateImpV9.java
public void installViewFactory() {
    LayoutInflater layoutInflater = LayoutInflater.from(mContext);
    //1. 没有Factory,系统会创建一个Factory去进行XML到View的转换
    if (layoutInflater.getFactory() == null) {
        LayoutInflaterCompat.setFactory2(layoutInflater, this);
    } else {
        if (!(layoutInflater.getFactory2() instanceof AppCompatDelegateImplV9)) {
            Log.i(TAG, "The Activity's LayoutInflater already has a Factory installed" + " so w
        }
    }
}
//LayoutInflaterCompat.java
public static void setFactory2(@NonNull LayoutInflater inflater, @NonNull LayoutInflater.Factor
    //1. 能将Factory接口绑定到创建View的LayoutInflater(IMPL类型为LayoutInflaterCompatBaseImpl)
    IMPL.setFactory2(inflater, factory);
}
//LayoutInflaterCompat.java内部类LayoutInflaterCompatBaseImpl:
static class LayoutInflaterCompatBaseImpl {
    //xxx
    public void setFactory2(LayoutInflater inflater, LayoutInflater.Factory2 factory) {
        inflater.setFactory2(factory);
        //xxx
    }
}
//LayoutInflater.java-完成Factory的创建
public void setFactory2(Factory2 factory) {
    //xxx
    if (mFactory == null) {
        mFactory = mFactory2 = factory;
    } else {
        mFactory = mFactory2 = new FactoryMerger(factory, factory, mFactory, mFactory2);
    }
}
```

setContentView流程

2、AppCompatActivity的OnCreate中setContentView()的流程

```
//AppCompatDelegateImplV9.java
    public void setContentView(int resId) {
        //xxx
        //1. 获取到父容器Content
        ViewGroup contentParent = (ViewGroup) mSubDecor.findViewById(android.R.id.content);
        contentParent.removeAllViews();
        //2. 通过LayoutInflater加载布局文件
        LayoutInflater.from(mContext).inflate(resId, contentParent);
        mOriginalWindowCallback.onContentChanged();
    }

//LayoutInlfater.java
    public View inflate(@LayoutRes int resource, @Nullable ViewGroup root) {
        return inflate(resource, root, root != null);
    }

//LayoutInlfater.java
    public View inflate(@LayoutRes int resource, @Nullable ViewGroup root, boolean attachToRoot
        final Resources res = getContext().getResources();
        //xxx
        final XmlResourceParser parser = res.getLayout(resource);
        //1. 重点
        return inflate(parser, root, attachToRoot);
    }
//LayoutInlfater.java
    public View inflate(XmlPullParser parser, @Nullable ViewGroup root, boolean attachToRoot) 
        ...
        final String name = parser.getName();//控件名
        //1. 将XmlPullParser转换为View的属性AttributeSet,给其他方法使用
        final AttributeSet attrs = Xml.asAttributeSet(parser);
        //2. Temp是XML文件中的根布局(name为"LinearLayout"等等)
        final View temp = createViewFromTag(root, name, inflaterContext, attrs);
        //3. 将XML根布局中temp下面所有的子View都进行加载
        rInflateChildren(parser, temp, attrs, true);
        //4. 将根布局tmp中找到的所有View贴到root中(content view)
        if (root != null && attachToRoot) {
            root.addView(temp, params);
        }
        ...
    }

/**=======================================
* 通过提供的属性AttributeSet attrs, 创建View
* // LayoutInlfater.java
*=========================================*/
    View createViewFromTag(View parent, String name, Context context, AttributeSet attrs, boole
        //1. 彩蛋?<blink>标签会进行闪烁
        if (name.equals(TAG_1995)) {
            // Let's party like it's 1995!
            return new BlinkLayout(context, attrs);
        }
        //2. 通过Factory创建View
        View view;
        view = mFactory2.onCreateView(parent, name, context, attrs);
        //xxx
        return view;
    }

//AppCompatDelegateImplV9.java
    public final View onCreateView(View parent, String name, Context context, AttributeSet attr
       ...
        //创建View
        return createView(parent, name, context, attrs);
    }
//AppCompatDelegateImplV9.java
    public View createView(View parent, final String name, @NonNull Context context, @NonNull A
        ...
        return mAppCompatViewInflater.createView(parent, name, context, attrs, inheritContext,
                IS_PRE_LOLLIPOP, /* Only read android:theme pre-L (L+ handles this anyway) */
                true, /* Read read app:theme as a fallback at all times for legacy reasons */
                VectorEnabledTintResources.shouldBeUsed() /* Only tint wrap the context if enab
        );
    }
//AppCompatViewInflater.java-最终完成从XML到View的转变
    public final View createView(View parent, final String name, Context context, AttributeSet

        View view = null;
        switch (name) {
            case "TextView":
                view = new AppCompatTextView(context, attrs);
                break;
            case "ImageView":
                view = new AppCompatImageView(context, attrs);
                break;
            case "Button":
                view = new AppCompatButton(context, attrs);
                break;
            case "EditText":
                view = new AppCompatEditText(context, attrs);
                break;
            case "Spinner":
                view = new AppCompatSpinner(context, attrs);
                break;
            case "ImageButton":
                view = new AppCompatImageButton(context, attrs);
                break;
            case "CheckBox":
                view = new AppCompatCheckBox(context, attrs);
                break;
            case "RadioButton":
                view = new AppCompatRadioButton(context, attrs);
                break;
            case "CheckedTextView":
                view = new AppCompatCheckedTextView(context, attrs);
                break;
            case "AutoCompleteTextView":
                view = new AppCompatAutoCompleteTextView(context, attrs);
                break;
            case "MultiAutoCompleteTextView":
                view = new AppCompatMultiAutoCompleteTextView(context, attrs);
                break;
            case "RatingBar":
                view = new AppCompatRatingBar(context, attrs);
                break;
            case "SeekBar":
                view = new AppCompatSeekBar(context, attrs);
                break;
        }
        ...
        return view;
    }
```

换肤中的应用

3、自定义Activity中通过Factory对控件的创建进行拦截，实现“换肤”效果：

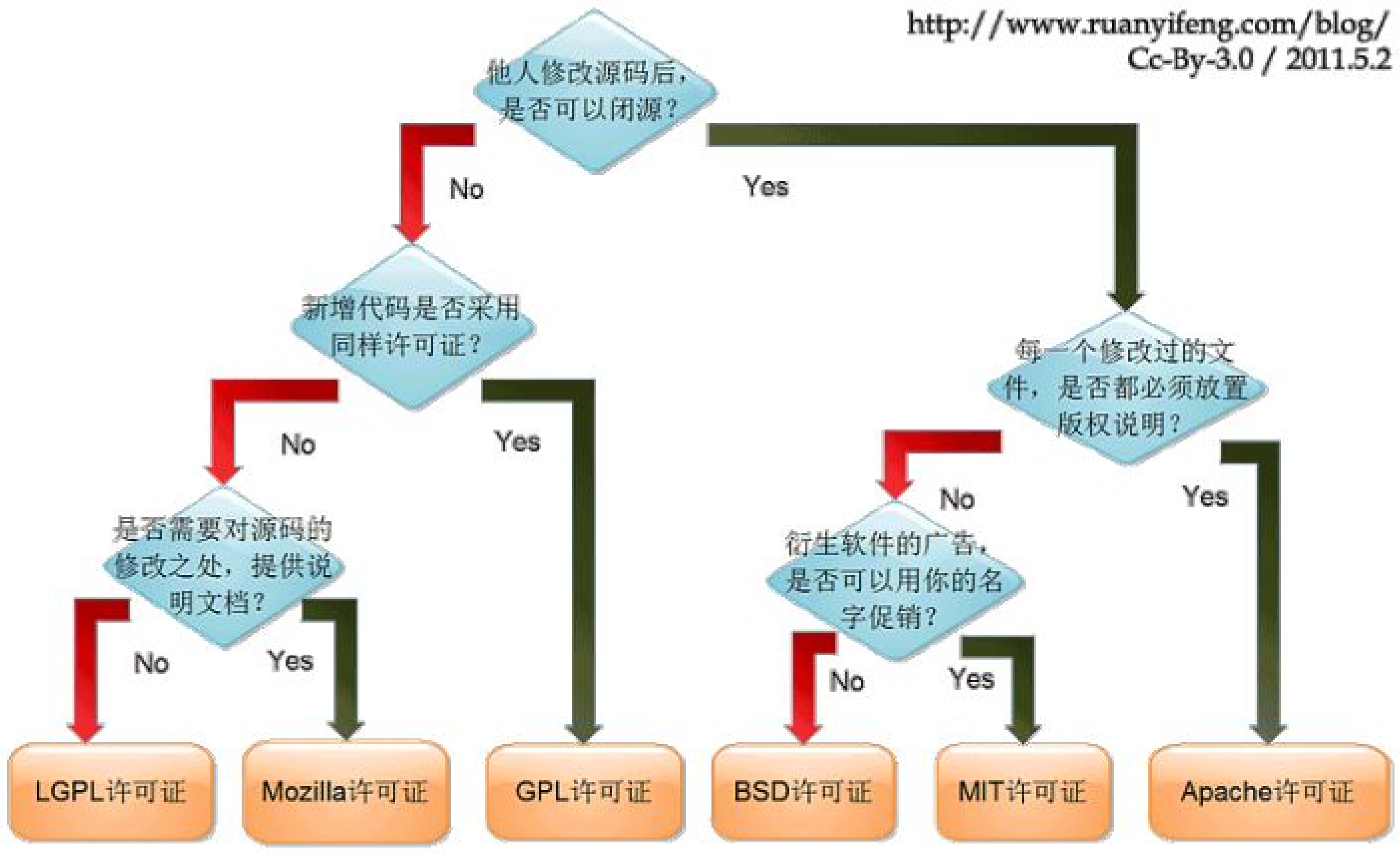
```java
public class SkinActivity extends AppCompatActivity{
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        LayoutInflaterCompat.setFactory2(LayoutInflater.from(this), new LayoutInflater.Factory2
            @Override
            public View onCreateView(View parent, String name, Context context, AttributeSet at
                AppCompatDelegate delegate = getDelegate();
                View view = delegate.createView(parent, name, context, attrs);
                return view;
            }

            @Override
            public View onCreateView(String name, Context context, AttributeSet attrs) {
                View view = null;
                switch (name) {
                    case "TextView":
                        view = new AppCompatTextView(context, attrs);
                        break;
                    case "ImageView":
                        view = new AppCompatImageView(context, attrs);
                        break;
                    case "Button":
                        view = new AppCompatButton(context, attrs);
                        break;
                    case "EditText":
                        view = new AppCompatEditText(context, attrs);
                        break;
                    //...
                }
                return view;
            }
        });
    }
}
```

知识储备

1、开源协议有哪些?



http://www.ruanyifeng.com/blog/
Cc-By-3.0 / 2011.5.2
他人修改源码后，是否可以闭源？
No
Yes
新增代码是否采用同样许可证？
No
Yes
每一个修改过的文件，是否都必须放置版权说明？
No
Yes
是否需要对源码的修改之处，提供说明文档？
No
Yes
衍生软件的广告，是否可以用你的名字促销？
No
Yes
LGPL许可证
Mozilla许可证
GPL许可证
BSD许可证
MIT许可证
Apache许可证

学习和参考资料

1. Android 探究 LayoutInflater setFactory
2. Github:Android-Skin(换肤库)