Fragment总结

https://blog.csdn.net/m0_37591251/article/details/84719444

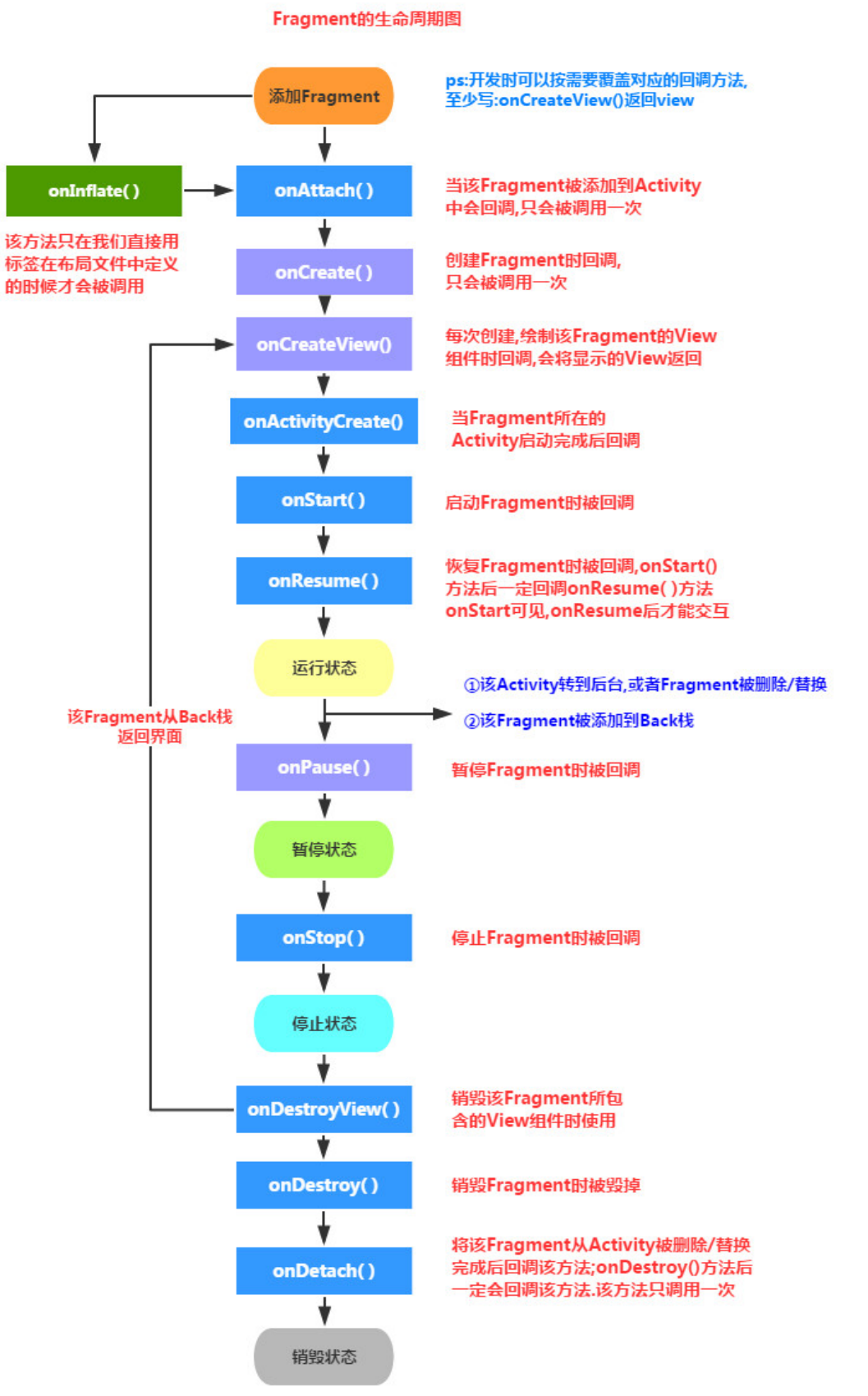
特性

1. Fragment是依赖于Activity的，不能独立存在的。
2. 一个Activity里可以有多个Fragment。
3. 一个Fragment可以被多个Activity重用。
4. Fragment有自己的生命周期，并能接收输入事件。
5. 我们能在Activity运行时动态地添加或删除Fragment。

有了这些特性，再想一想手机和我们之前使用的应用，我们发现它对手机开发还是有益的。比如说：

- **模块化（Modularity）**： 我们不必把所有代码全部写在Activity中，而是把代码写在各自的Fragment中。
- **可重用（Reusability)**： 多个Activity可以重用一个Fragment。
- **可适配（Adaptability）**： 根据硬件的屏幕尺寸、屏幕方向，能够方便地实现不同的布局，这样用户体验更好。

生命周期


Fragment的生命周期图
添加Fragment
ps:开发时可以按需要覆盖对应的回调方法,
至少写:onCreateView()返回view
onInflate()
该方法只在我们直接用
标签在布局文件中定义
的时候才会被调用
onAttach()
当该Fragment被添加到Activity
中会回调,只会被调用一次
onCreate()
创建Fragment时回调,
只会被调用一次
onCreateView()
每次创建,绘制该Fragment的View
组件时回调,会将显示的View返回
onActivityCreate()
当Fragment所在的
Activity启动完成后回调
onStart()
启动Fragment时被回调
onResume()
恢复Fragment时被回调,onStart()
方法后一定回调onResume()方法
onStart可见,onResume后才能交互
运行状态
①该Activity转到后台,或者Fragment被删除/替换
②该Fragment被添加到Back栈
该Fragment从Back栈
返回界面
onPause()
暂停Fragment时被回调
暂停状态
onStop()
停止Fragment时被回调
停止状态
onDestroyView()
销毁该Fragment所包
含的View组件时使用
onDestroy()
销毁Fragment时被毁掉
onDetach()
将该Fragment从Activity被删除/替换
完成后回调该方法;onDestroy()方法后
一定会回调该方法.该方法只调用一次
销毁状态
https://blog.csdn.net/final__static

走一遍生命周期：

①Activity加载Fragment的时候,依次调用下面的方法: onAttach -> onCreate -> onCreateView ->onActivityCreated -> onStart ->onResume
②当我们弄出一个悬浮的对话框风格的Activity,或者其他,就是让Fragment所在的Activity可见,但不获得焦点 onPause
③当对话框关闭,Activity又获得了焦点: onResume
④当我们替换Fragment,并调用addToBackStack()将他添加到Back栈中 onPause -> onStop -> onDestoryView ！！注意,此时的Fragment还没有被销毁哦!!!
⑤当我们按下键盘的回退键，Fragment会再次显示出来: onCreateView -> onActivityCreated -> onStart -> onResume
⑥如果我们替换后,在事务commit之前没有调用addToBackStack()方法将 Fragment添加到back栈中的话;又或者退出了Activity的话,那么Fragment将会被完全结束, Fragment会进入销毁状态 onPause -> onStop -> onDestoryView -> onDestory -> onDetach

接下来就是创建一个fragment，创建fragment有两个方式。

1. 静态添加：通过xml的方式添加，缺点是一旦添加就不能在运行时删除。
2. 动态添加：运行时添加，这种方式比较灵活，因此建议使用这种方式。

1. 静态加载

示例代码:

Step 1:定义Fragment的布局，就是fragment显示内容的
Step 2:自定义一个Fragment类,需要继承Fragment或者他的子类,重写onCreateView()方法 在该方法中调用:inflater.inflate()方法加载Fragment的布局文件,接着返回加载的view对象

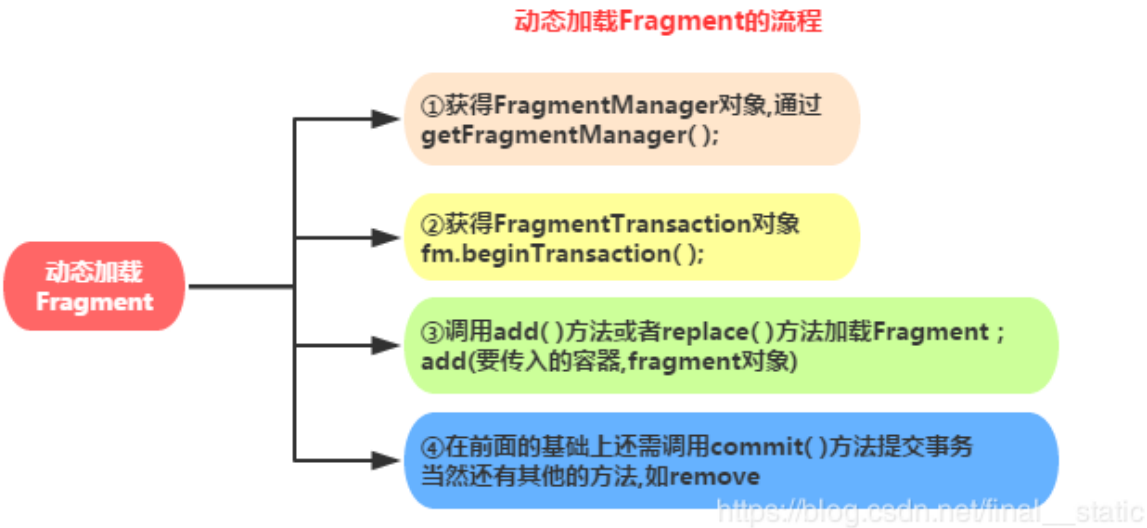
```
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment1, container,false);
        return view;
    }
}
```

Step 3:在需要加载Fragment的Activity对应的布局文件中添加fragment的标签， 记住，name属性是全限定类名哦，就是要包含Fragment的包名，如:

```
    android:id="@+id/fragment1"
    android:name="com.jay.example.fragmentdemo.Fragmentone"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1" />
```

Step 4: Activity在onCreate( )方法中调用setContentView()加载布局文件即可!

2. 动态加载Fragment

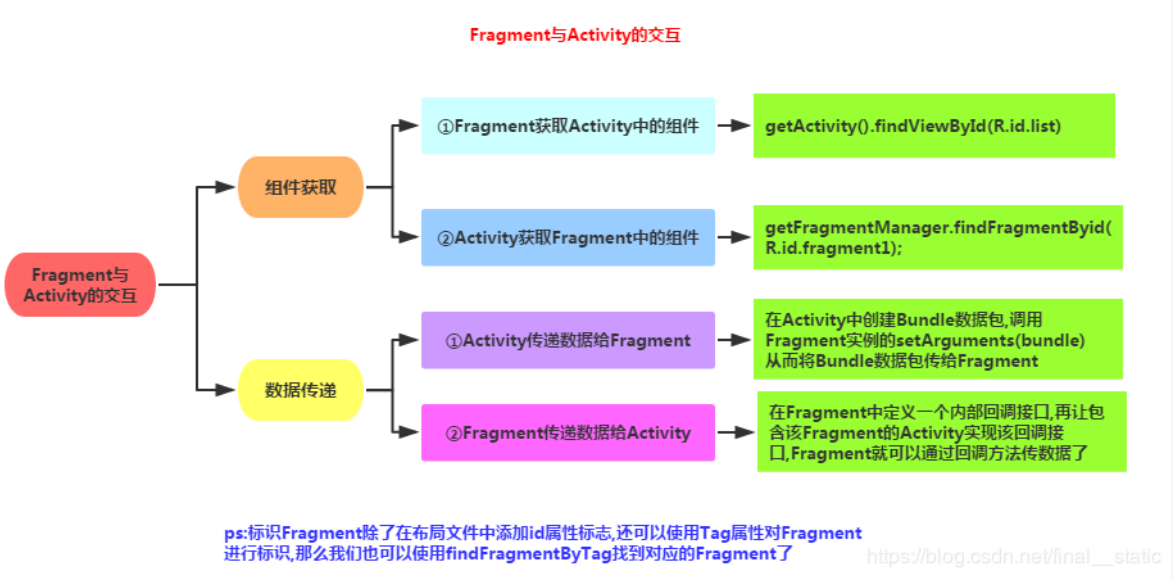



```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Display dis = getWindowManager().getDefaultDisplay();
        if(dis.getWidth() > dis.getHeight())
        {
            Fragment1 f1 = new Fragment1();
            getFragmentManager().beginTransaction().replace(R.id.LinearLayout1,
f1).commit();
        }

        else
        {
            Fragment2 f2 = new Fragment2();
            getFragmentManager().beginTransaction().replace(R.id.LinearLayout1,
f2).commit();
        }
    }
}
```

如何使用fragment:



重点讲fragment->Activity

Step 1:定义一个回调接口:(Fragment中)

```
/*接口*/
public interface CallBack{
    /*定义一个获取信息的方法*/
    public void getResult(String result);
}
```

Step 2：接口回调（Fragment中）

```
/*接口回调*/
public void getData(CallBack callBack){
    /*获取文本框的信息,当然你也可以传其他类型的参数,看需求咯*/
    String msg = editText.getText().toString();
    callBack.getResult(msg);
}
```

Step 3:使用接口回调方法读数据(Activity中)

```
/* 使用接口回调的方法获取数据 */
leftFragment.getData(new CallBack() {
 @Override
       public void getResult(String result) {              /*打印信息*/
            Toast.makeText(MainActivity.this, "-->>" + result, 1).show();
            }
        });
```

```
Activity->Fragment  Fragment的getArguments()拿不到数据为什么？

解决方法：确保Activity中实例化的fragment和replace等操作的是同一个实例；
```

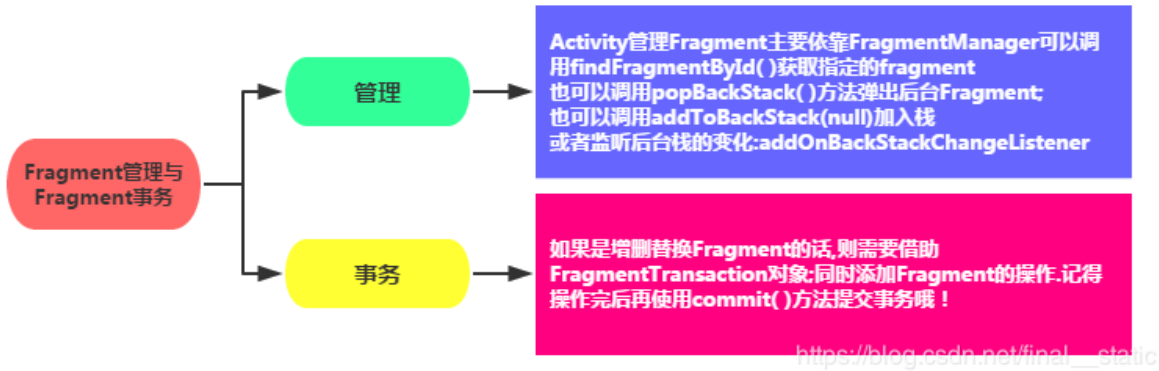
fragment之间怎么传递 数据

其实这很简单,找到要接受数据的fragment对象,直接调用setArguments传数据进去就可以了 通常的话是replace时,即fragment跳转的时候传数据的,那么只需要在初始化要跳转的Fragment 后调用他的setArguments方法传入数据即可!
如果是两个Fragment需要即时传数据,而非跳转的话,就需要先在Activity获得f1传过来的数据, 再传到f2了,就是以Activity为媒介~

代码示例:

```
FragmentManager fManager = getSupportFragmentManager( );
FragmentTransaction fTransaction = fManager.beginTransaction();
Fragmentthree t1 = new Fragmentthree();
Fragmenttwo t2 = new Fragmenttwo();
Bundle bundle = new Bundle();
bundle.putString("key",id);
t2.setArguments(bundle);
fTransaction.add(R.id.fragmentRoot, t2, "~~~");
fTransaction.addToBackStack(t1);
fTransaction.commit();
```

管理:



小结

经过再次学习fragment，清楚了fragment的两种初始化的方式，一般用动态创建的方式，因为比较灵活，还有fragment在后面可以结合viewpager一起使用，具体看viewpager笔记；