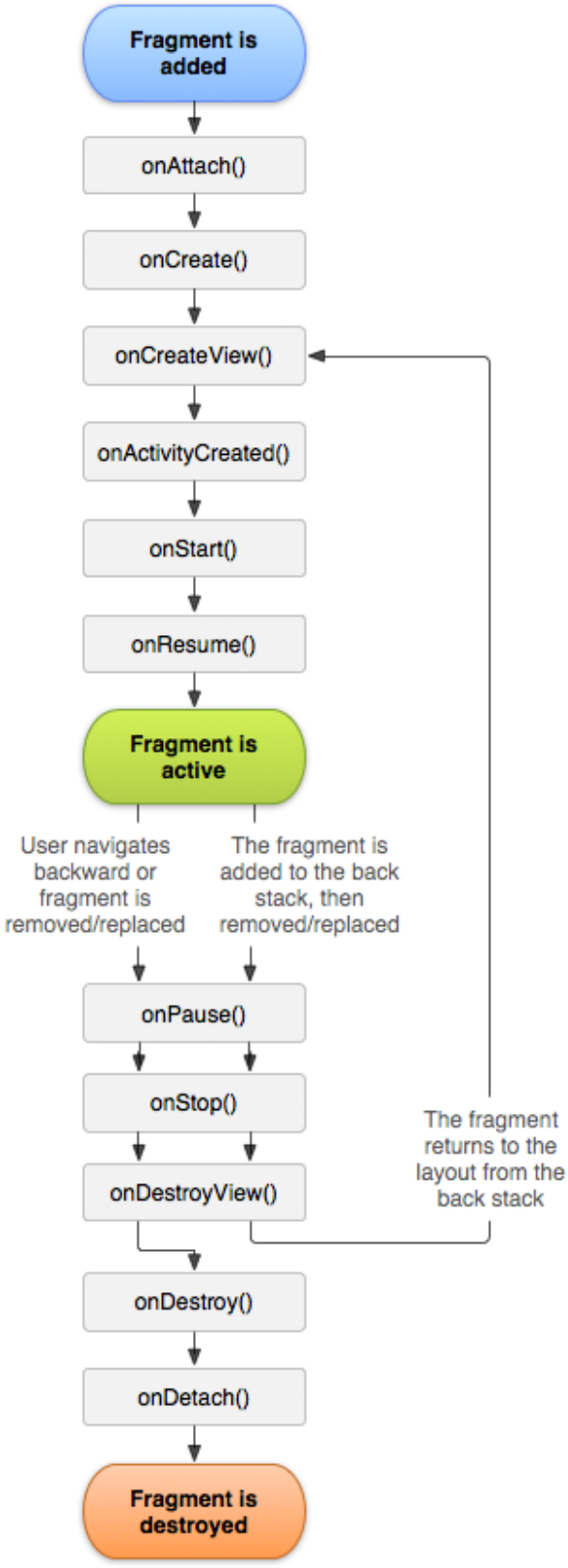
Fragment

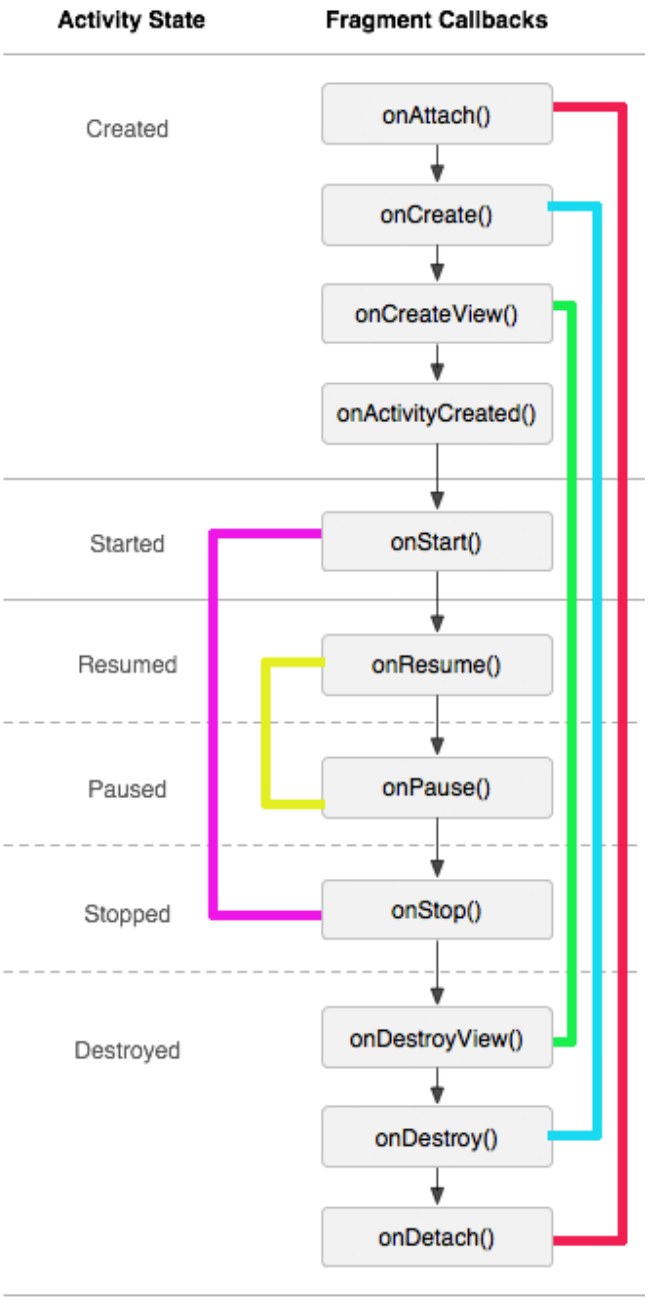
- Fragment概要
- Fragment生命周期
- 加载Fragment方法
  - 静态加载
  - 动态加载
- Fragment与Activity之间通信

1. Fragment概要

学习Fragment的时候可以联系之前学习过的Activity，因为它们有很多相似点：都可包含布局，有自己的生命周期，Fragment可看似迷你活动。正如Fragment的名字--碎片，它的出现是为了解决Android碎片化，它可作为Activity界面的组成部分，可在Activity运行中实现动态地加入、移除和交换。一个Activity中可同时出现多个Fragment，一个Fragment也可在多个Activity中使用。活动和碎片像极了夫妻， 虽然紧密联系但是又有独立空间，在一起让彼此变得更好。

2. Fragment生命周期





是不是能发现Fragment和Activity的生命周期太相似了，现在只需要再介绍几个Activity中没讲过的新方法：

> **onAttach()**：当Fragment和Activity建立关联时调用
> **onCreateView()**：当Fragment创建视图时调用
> **onActivityCreated()**：当与Fragment相关联的Activity完成onCreate()之后调用
> **onDestroyView()**：在Fragment中的布局被移除时调用
> **onDetach()**：当Fragment和Activity解除关联时调用

Fragment从创建到销毁整个生命周期中涉及到的方法依次为：

> **1.打开界面**
> onCreate()-> onCreateView()->onActivityCreated()->onStart()->onResume()
> **2.按下主屏幕键**
> onPause()->onStop()
> **3.重新打开界面**
> onStart()->onResume()
> **4.按下后退键**
> onPause()->onStop()->onDestroyView()->onDestroy()->onDetach()

在Activity中加入Fragment,对应的生命周期

> **1.打开**
> Fragment onCreate() -> Fragment onCreateView() ->
> `Activity` onCreate() ->
> Fragment onActivityCreated() ->
> `Activity` onStart()
> Fragment onStart() ->
> `Activity` onResume() ->
> Fragment onResume() ->
> **2.按下主屏幕键/锁屏**
> Fragment onPause() ->
> `Activity` onPause() ->
> Fragment onStop() ->
> `Activity` onStop() ->
> **3.再次打开**
> `Activity` onRestart() -> `Activity` onStart() ->
> Fragment onStart() ->
> `Activity` onResume() ->
> Fragment onResume()
> **4.按下后退键**
> Fragment onPause()->
> `Activity` onPause() ->
> Fragment onStop() ->
> `Activity` onStop() ->
> Fragment onDestroyView() -> Fragment onDestroy() -> Fragment onDetach() ->
> `Activity` onDestroy()

Fragment生命周期与Activity生命周期的一个关键区别就在于，Fragment的生命周期方法是由托管Activity而不是操作系统调用的。Activity中生命周期方法都是protected，而Fragment都是public，也能印证了这一点，因为Activity需要调用Fragment那些方法并管理它。

Activity和Fragment的异同?

> - 技术点：Fragment作用
> - 思路：分别解释“异”“同”
> - 参考回答： Activity和Fragment的相似点在于，它们都可包含布局、可有自己的生命周期，Fragment可看似迷

> 你活动。 不同点是，由于Fragment是依附在Activity上的，多了些和宿主Activity相关的生命周期方法，如onAttach()、onActivityCreated()、onDetach()；另外，Fragment的生命周期方法是由宿主Activity而不是操作系统调用的，Activity中生命周期方法都是protected，而Fragment都是public，也能印证了这一点，因为Activity需要调用Fragment那些方法并管理它。

Activity和Fragment的关系?

> - 技术点：Fragment作用
> - 思路：可从Fragment出现的目的、两者数量关系、调用关系展开
> - 参考回答：
>   - 正如Fragment的名字“碎片”，它的出现是为了解决Android碎片化 ，它可作为Activity界面的组成部分，可在Activity运行中实现动态地加入、移除和交换。
>   - 一个Activity中可同时出现多个Fragment，一个Fragment也可在多个Activity中使用。
>   - 另外，Activity的FragmentManager负责调用队列中Fragment的生命周期方法，只要Fragment的状态与Activity的状态保持了同步，宿主Activity的FragmentManager便会继续调用其他生命周期方法以继续保持Fragment与Activity的状态一致。

何时会考虑使用Fragment?

> - 技术点：Fragment作用
> - 思路：列举更适合使用Fragment的情况
> - 参考回答：非常经典的例子，即用两个Fragment封装两个界面模块，这样只使一套代码就能适配两种设备，达到两种界面效果；单一场景切换时使用Fragment更轻量化，如ViewPager和Fragment搭配使用

3. 加载Fragment方法

- 静态加载

  第一步：新建一个TestFragment类并继承Fragment，这里引用android.app包下的就可以，另一个包下主要用于兼容低版本的安卓系统。然后重写onCreateView()方法，这个方法里通过LayoutInflater的inflate()方法布局加载进来并得到一个View，再return这个View。

```
public class TestFragment extends Fragment {
 @Override
 public View onCreateView(LayoutInflater inflater, ViewGroup container,
                          Bundle savedInstanceState) {
    return inflater.inflate(R.layout.fragment_test, container, false);
 }
 ...
}
```

  第二步：新建fragment_test.xml，为Fragment指定一个布局，这里简单的放一个TextView和一个Button。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
tools:context=".TestFragment">
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello_blank_fragment" />
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="按钮"/>
</LinearLayout>
```

第三步：在Activity的布局中使用< fragment>标签添加碎片，并且一定要有android：name属性且值为被加载的Fragment类的包名.类名完整形式。

```
<fragment
    android:id="@+id/fragment"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:name="com.wuyazhou.fragment.TestFragment"/>
```

第四步：在MainActivity中加载main布局。现在TestFragment就完成了静态加载到MainActivity中，这时碎片里的控件自然也是活动的一个部分，可直接在活动中获取到Button的实例，来注册点击事件了。

```
    Button button = findViewById(R.id.button);
    button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Toast.makeText(mContext,"测试",Toast.LENGTH_LONG).show();
        }
    });
```

- 动态加载

第一步：在Activity的布局里面添加一个LinerLayout的控件并且给定Id

```
 <RelativeLayout
    android:id="@+id/layout"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:background="@color/colorAccent"/>
```

第二步：在MainActivity中加入TestFrgment

```
    TestFragment testFragment = new TestFragment();
    FragmentManager fragmentManager = getSupportFragmentManager();
    FragmentTransaction beginTransaction = fragmentManager.beginTransaction();

    beginTransaction.add(R.id.layout, testFragment);
    beginTransaction.addToBackStack(null);
    beginTransaction.commit();
```

动态加载进来的Fragment里的控件并不能直接在活动中findViewById得到

4. Fragment与Activity之间通信

A.可以使用F**ragmentManager**的**findFragmentById()**方法来得到相应碎片的实例，继而可以调用碎片里的方法

- 静态加载
  `TestFragment fragment = (TestFragment) fragmentManager.findFragmentById(R.id.fragment);`
- 动态加载
  `TestFragment fragment = (TestFragment) fragmentManager.findFragmentById(R.id.layout);`

B.碎片中也可以通过调用getActivity()方法来得到和当前碎片相关联的活动实例，这样调用活动里的方法就变得轻而易举了