使用 FragmentManager 和 FragmentTransaction

知识解析

FragmentManager

它是用于管理 Activity 中的 Fragment 的类

一般通过 Activity 的 getFragmentManager()方法来获得其对

它的主要方法：

- FragmentTransaction beginTransaction()：使用 Fragment 时，可以通过用户交互来执行一些动作，比如增加、移除、替换等。所有这些改变构成一个集合，这个集合被叫做一个 transaction。可以调用 FragmentTransaction 中的方法来处理这个 transaction，并且可以将 transaction 存进由 activity 管理的 back stack 中，这样用户就可以进行 fragment 变化的回退操作。这个方法可以开始一个 transaction，并且返回一个 FragmentTransaction 对象。
- Fragment findFragmentById(int id)：在 activity 中根据 id 寻找一个存在的 fragment。
- Fragment findFragmentByTag(String tag)：在 activity 中根据 tag 寻找一个存在的 fragment。
- popBackStack()：从 fragment 堆栈中弹出栈顶的 fragment。

FragmentTransaction

FragmentTransaction 用于处理一系列的 Fragment 操作，如新增、删除、替换 Fragment 等。

主要方法：

- add(int containerViewId,Fragment fragment)：将 fragment 加入到容器 containerViewId 中
- add(Fragment fragment,String tag)
- add(int containerViewId,Fragment fragment,String tag)
- addToBackStack(String tag)：将当前的 fragment 放入到 fragment 堆栈中，后续可

以使用 FragmentManager 的 popBackStack()取出

- attach(Fragment fragment)：将通过 detach()从 activity 中 detach 的 fragment 重新附加(attach)到 activity。
- detach(Fragment fragment)：解除 fragment
- remove(Fragment fragment)：移除 fragment
- replace(int containerViewId, Fragment fragment, String tag)：将 containerViewId 对应的容器中的 fragment 替换成 fragment，并设置 tag。
- replace(int containerViewId, Fragment fragment)
- show(Fragment fragment)：显示 fragment
- hide(Fragment fragment)：隐藏 fragment
- commit()：这个方法在前面的这些方法调用后必须调用，让以上的改变真正生效。

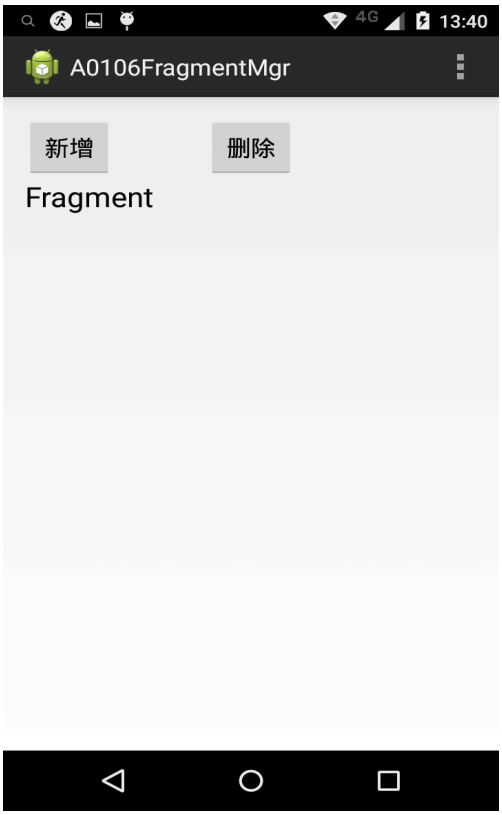
动态实例化 Fragment 的时机

如果 activity 发生配置变化（比如旋转屏幕）时，它将重新创建 activity，并且会自动实例化已经存在的 fragments

为了避免重复创建 fragment，如果在 Activity.onCreate()中动态创建 fragment，那么可以判断是新创建 activity 还是因为配置发生变化而重新创建 activity，可以根据 onCreate()中的 Bundle 参数是否为空来判断：

```
if(savedInstanceState == null){
    //创建 fragment
}
```

功能演示



实战操作

```
public class MainActivity extends Activity {
    FragmentTransaction trans;
    FragmentManager mgr;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mgr = this.getFragmentManager();
```

```
    }
    public void add(View view) {
        trans = mgr.beginTransaction();
        MyFragment fragment = new MyFragment();
        trans.add(R.id.linearLayout, fragment, "fragment");
        trans.commit();
    }

    public void del(View view) {
        Fragment fragment = mgr.findFragmentByTag("fragment");
        System.out.println();
        if (fragment != null) {
            trans = mgr.beginTransaction();
            trans.remove(fragment);
            trans.commit();
        }
    }
}
```

职业素质

FragementManager 可以完成如下几方面功能

- 使用 findFragementById()或者 findFragmentBuyTag()方法获取制定 Fragment
- 使用 popBackStack 方法将 Fragement 从后台栈中弹出
- 调用 addOnBackStackChangeListener（）注册一个监听器，由于监听后台栈变化。

为更好更方便的使用 Fragment 就必须熟练掌握 FragmentManager 和 FragmentTransaction 的使用，工欲善其事，必先利其器。