Demo16-Fragment最常用的基类BaseFragment

import android.app.Activity;

import android.os.Bundle;

import android.support.annotation.Nullable;

import android.support.v4.app.Fragment;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

/\*\*

\* 面向对象的本质：构造对象，调用象方

\* 因为有抽象方法强制子类一定要重写，所以变成抽象类

\*/

public abstract class BaseFragment extends Fragment {

//上下文声明，Fragment习惯性把所在Activity的实例单独写出来

public Activity context;

/\*\*

\* 重写生命周期方法：初始化context

\* @param savedInstanceState

\*/

@Override

public void onCreate(@Nullable Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

context=getActivity();

}

/\*\*

\* 创建视图时回调此方法，所以里面负责写创建视图的代码，创建实体对象，设置属性，但是不填充数据

\* @param inflater

\* @param container

\* @param savedInstanceState

\* @return

\*/

@Nullable

@Override

public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {

return initView();

}

//抽样方法，强制给子类自己写内容创建视图，而且是在onCreateView中执行

public abstract View initView();

/\*\*

\* Activity创建时回调此方法，而且填充实体对象的数据

\* @param savedInstanceState

\*/

@Override

public void onActivityCreated(@Nullable Bundle savedInstanceState) {

super.onActivityCreated(savedInstanceState);

initData(); //先有View,才能填充数据，所以不要把数据写到View里面去，可能会拖慢

}

public void initData() {

}

}