Demo17-最常用的使用FrameLayout加载Fragment

1. left\_fragment.xml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="vertical" android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<Button

android:id="@+id/btn\_1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:text="Button\_Left"

/>

</LinearLayout>

1. right\_fragment.xml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="vertical"

android:background="#3e3e3e"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<Button

android:id="@+id/btn\_2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:text="Button\_Right"

/>

</LinearLayout>

1. BaseFragment.java

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() {

}

}

1. LeftFragment.java

import android.os.Bundle;

import android.support.annotation.NonNull;

import android.support.annotation.Nullable;

import android.support.v4.app.Fragment;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.Button;

import android.widget.Toast;

public class LeftFragment extends BaseFragment {

private Button mButton;

@Override

public View initView() {

View view=View.inflate(context, R.layout.left\_fragment,null);

mButton=(Button)view.findViewById(R.id.btn\_1);

return view;

}

@Override

public void initData() {

super.initData();

mButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Toast.makeText(getContext(),"Left",Toast.LENGTH\_SHORT).show();

}

});

}

}

1. RightFragment.java

import android.os.Bundle;

import android.support.annotation.NonNull;

import android.support.annotation.Nullable;

import android.support.v4.app.Fragment;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.Button;

import android.widget.Toast;

public class RightFragment extends BaseFragment {

private Button mButton;

@Override

public View initView() {

View view=View.inflate(context, R.layout.right\_fragment,null);

mButton=(Button)view.findViewById(R.id.btn\_2);

return view;

}

@Override

public void initData() {

super.initData();

mButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Toast.makeText(getContext(),"Right",Toast.LENGTH\_SHORT).show();

}

});

}

}

1. 在MainActivity中使用FragmentManager开启事务来以Fragment对象实例填充activity\_main.xml的FrameLayout，就完成了。

import android.support.v4.app.FragmentManager;

import android.support.v4.app.FragmentTransaction;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

private static final String LEFT\_FRAGMENT\_TAG ="left\_fragment\_tag";

private static final String RIGHT\_FRAGMENT\_TAG ="right\_fragment\_tag" ;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

initFragment();

}

//初始化Fragment的view

private void initFragment() {

//1.构造FragmentManager对象

FragmentManager fm=getSupportFragmentManager();

//2、得到FragmentTransaction对象开启事务

FragmentTransaction ft=fm.beginTransaction();

//3、事务对象调用象方来使ContentFragment的view来覆盖fl\_main\_content

//LeftMenuFragment的view来覆盖fl\_left\_menu，tag是后面在其他地方可以找到这两个Fragment

ft.replace(R.id.left\_frame, new LeftFragment(), LEFT\_FRAGMENT\_TAG);

ft.replace(R.id.right\_frame, new RightFragment(), RIGHT\_FRAGMENT\_TAG);

//4、事务对象调用象方来提交

ft.commit();

}

/\*\*

\* 得到左侧碎片的实例

\* @return

\*/

public LeftFragment getLeftMenuFragment() {

//构造FragmentManager对象

FragmentManager fm=getSupportFragmentManager();

//构造LeftMenuFragment对象： 通过构造FragmentManager对象调用象方

LeftFragment leftFragment= (LeftFragment) fm.findFragmentByTag(LEFT\_FRAGMENT\_TAG);

return leftFragment;

}

/\*\*

\* 得到右侧碎片的实例

\* @return

\*/

public RightFragment getContentFragment() {

return (RightFragment) getSupportFragmentManager().findFragmentByTag(RIGHT\_FRAGMENT\_TAG);

}

}