FragmentManager API

FragmentManager用来管理Activity中的Fragment

方法：

findFragmentById() 通过id查找Activity中的Fragment

findFragmentByTag() 通过字符串查找Activity中的Fragment

popBackStack() 模拟用户的点击Back键的操作

addOnBackStackChangedListener() 添加后台栈改变监听器

Activity android.app.activity

FragmentActivity **import** android.support.v4.app.FragmentActivity;

所需要用到的类都需要用同个包的，必须相互兼容。

FragmentTransaction API

add() 向容器中添加一个Fragment

remove() 删除一个Fragment

replace() 将容器中的Fragment替换成新的Fragment

hide() 隐藏已存在的Fragment

show() 显示此前隐藏的Fragment

addToBackStack() 将事务添加到后台栈

commit() 提交事务，将改变应用到Activity

只有动态添加Fragment到宿主Activity中，并且对Fragment的每一个操作，Fragment都单独运行，才有独立的生命周期。此时Fragment的生命周期才与宿主Activity无关。

系统提供的Fragment

系统提供了几个其他的Fragmen类开发者可以继承

DialogFragment 显示一个悬浮对话框。

ListFragment 显示一个由adapter管理的项目列表

fragment\_demo.xml

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"**>  
  
 <**TextView  
 android:id="@+id/text1"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Hello，黄继升！"  
 android:textAppearance="?android:attr/textAppearanceLarge"  
 android:textColor="#FF0000"**/>  
</**LinearLayout**>

MyFragment.java

**package** com.amaker.hjs.fragmentmanagerdemo.fragment;  
  
**import** android.annotation.SuppressLint;  
**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.TextView;  
  
@SuppressLint(**"NewApi"**)  
**public class** MyFragment **extends** Fragment {  
 **private** TextView **text**;  
 **private int count**;  
  
 *// 注意：API规定：Fragment本身不允许有带参构造来创建并初始化Fragment对象  
 // 即new出的Fragment实例，其成员变量没有初始化  
 // 初始化Fragment对象的成员变量的操作，是必须让onCreate方法来完成的，相当于set方法  
 // 因此对于Fragment的成员初始化，是使用静态方法来创建对象，然后在onCreate方法中对成员赋值，  
 // 因此我们可以在创建Fragment对象的同时，在内部设置一个传值的Bundle键值对，用来暂存数据对象* **public static** MyFragment newInstace(**int** num){  
 MyFragment f = **new** MyFragment();  
  
 *//用bundle键对象暂存初始化数据* Bundle bundle = **new** Bundle();  
 bundle.putInt(**"count"**,num);  
  
 *//在Fragment中暂存数据对象，并返回对象* f.setArguments(bundle);  
 **return** f;  
 }  
  
 @Override  
 **public void** onCreate(@Nullable Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 Bundle bundle = getArguments();  
 **this**.**count** = bundle.getInt(**"count"**);*//成员变量赋初值* }  
  
 @Override  
 **public** View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  
 View view = inflater.inflate(R.layout.***fragment\_demo***,**null**);  
 **return** view;  
 }  
  
  
 */\*\*  
 \* 在拿到onCreateView方法解析执行后返回的view后，调用这个方法来拿到fragment的各种控件，并设置控件参数  
 \** ***@param view*** *onCreateView方法解析执行后返回的view  
 \** ***@param savedInstanceState*** *\*/* @Override  
 **public void** onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {  
 **super**.onViewCreated(view,savedInstanceState);  
 **text**=(TextView) view.findViewById(R.id.***text1***);  
 **text**.setText(**"Fragment #"**+**count**);  
 }  
  
  
  
  
 */\*\*封装  
 \* 只公开内部私有成员的Text只写属性  
 \** ***@param text*** *\*/* **public void** setGreeting(String text){  
 **this**.**text**.setText(text);  
 }  
}

activity\_main.xml

*<?***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"  
 tools:context=".MainActivity"  
 android:orientation="vertical"**>  
  
 *<!--把创建好的Fragment引进来，静态方式-->* <**fragment  
 android:id="@+id/myfragment1"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:name="com.amaker.hjs.fragmentmanagerdemo.fragment.MyFragment"**/>  
  
 <**Button  
 android:id="@+id/button"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="点这里"  
 android:onClick="changeGreeting"**/>  
  
 <**fragment  
 android:tag="testFragment"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:name="com.amaker.hjs.fragmentmanagerdemo.fragment.MyFragment"**/>/>  
  
  
</**LinearLayout**>

MainActivity.java

**package** com.amaker.hjs.fragmentmanagerdemo.fragment;  
  
**import** android.support.v4.app.FragmentActivity;  
**import** android.support.v4.app.FragmentManager;  
**import** android.os.Bundle;  
**import** android.view.View;  
  
**public class** MainActivity **extends** FragmentActivity {  
 **private** FragmentManager **mFragmentManager**;  
 **private** MyFragment **mMyFragment** ;  
 **private** MyFragment **mMyFragment2**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
  
 *//注意Fragment不能直接用findVIewById去得到  
 //（1）获得FragmentManager对象* **mFragmentManager** = getSupportFragmentManager();  
 *//（2）查找MyFragment，由于返回的是安卓父类的Fragment，需要做强制类型转换* **mMyFragment** = (MyFragment) **mFragmentManager**.findFragmentById(R.id.***myfragment1***);  
 **mMyFragment2** = (MyFragment) **mFragmentManager**.findFragmentByTag(**"testFragment"**);  
  
 }  
  
 **public void** changeGreeting(View view){  
 **mMyFragment2**.setGreeting(**"你竟然点击了我，黄继升！"**);  
 }  
}

activity\_demo.xml

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"**>  
  
 <**FrameLayout  
 android:id="@+id/layout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="200dp"**>  
  
 </**FrameLayout**>  
  
 <**Button  
 android:id="@+id/button3"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="动态添加Fragment"  
 android:onClick="dynamicAddFragment"**/>  
  
 <**Button  
 android:id="@+id/button4"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="动态移除Fragment"  
 android:onClick="dynamicRemoveFragment"**/>  
  
 <**Button  
 android:id="@+id/button5"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="显示和隐藏Fragment"  
 android:onClick="hideAndShowFragment"**/>  
  
</**LinearLayout**>

DemoActivity.java

**package** com.amaker.hjs.fragmentmanagerdemo.fragment;  
  
**import** android.os.Bundle;  
**import** android.support.annotation.Nullable;  
**import** android.support.v4.app.FragmentActivity;  
**import** android.support.v4.app.FragmentManager;  
**import** android.support.v4.app.FragmentTransaction;  
**import** android.view.View;  
  
**public class** DemoActivity **extends** FragmentActivity {  
 @Override  
 **protected void** onCreate(@Nullable Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_demo***);  
 }  
  
 **public void** dynamicAddFragment(View view){  
  
 *//1.获得FragmentManager* FragmentManager fm = getSupportFragmentManager();  
 *//2.通过FragmentManager得到FragmentTransaction对象* FragmentTransaction ft = fm.beginTransaction();  
 *//3.先创建Fragment对象，再将Fragment添加到容器中* MyFragment fragment = **new** MyFragment();  
 ft.add(R.id.***layout*** , fragment , **"test"**);*// 参数1：容器id 参数2：Fragment对象 参数3：Fragment的tag  
 //4.提交事务，表示完成* ft.commit();  
  
 }  
  
  
 **public void** dynamicRemoveFragment(View view){  
 *//1.获得FragmentManager* FragmentManager fm = getSupportFragmentManager();  
 *//2.通过FragmentManager得到FragmentTransaction对象* FragmentTransaction ft = fm.beginTransaction();  
 *//3.先查找，再移除* MyFragment fragment = (MyFragment) fm.findFragmentByTag(**"test"**);  
 ft.remove(fragment);  
 *//4.提交事务，表示完成* ft.commit();  
  
 }  
  
 **public void** hideAndShowFragment(View view){  
 *//1.获得FragmentManager* FragmentManager fm = getSupportFragmentManager();  
 *//2.通过FragmentManager得到FragmentTransaction对象* FragmentTransaction ft = fm.beginTransaction();  
 *//3.先查找，再显示或隐藏* MyFragment fragment = (MyFragment) fm.findFragmentByTag(**"test"**);  
 **if** (fragment.isHidden()){  
 ft.show(fragment);  
 }**else**{  
 ft.hide(fragment);  
 }  
 *//4.提交* ft.commit();  
 }  
  
}

activity\_demo2.xml

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:id="@+id/containerLayout"**>  
  
 <**Button  
 android:id="@+id/button2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="AddOrReplaceFragment"  
 android:onClick="replaceFragment"**/>  
  
 <**Button  
 android:id="@+id/button6"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="showDialog"  
 android:onClick="showDialog"**/>  
  
 <**Button  
 android:id="@+id/button7"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="showListFragment"  
 android:onClick="showListFragment"**/>  
</**LinearLayout**>

Demo2Activity.java

**package** com.amaker.hjs.fragmentmanagerdemo.fragment;  
  
  
**import** android.app.Dialog;  
**import** android.content.Context;  
**import** android.content.DialogInterface;  
**import** android.os.Bundle;  
**import** android.support.annotation.Nullable;  
**import** android.support.v4.app.DialogFragment;  
**import** android.support.v4.app.FragmentActivity;  
**import** android.support.v4.app.ListFragment;  
**import** android.support.v7.app.AlertDialog;  
**import** android.view.View;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.ListView;  
**import** android.widget.Toast;  
  
**public class** Demo2Activity **extends** FragmentActivity {  
 **private int num**=0;  
  
 @Override  
 **protected void** onCreate(@Nullable Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_demo2***);  
 }  
  
 **public void** replaceFragment(View view){  
 **num**++;  
 *//使用静态方法产生对象* MyFragment fragment = MyFragment.*newInstace*(**num**);  
 *//替换* getSupportFragmentManager().beginTransaction()  
 .replace(R.id.***containerLayout***,fragment)  
 .addToBackStack(**null**) *//模拟Activity回退栈，每次将被替换的Fragment进行压栈。参数是这个回退栈的名字* .commit();  
 }  
  
 **public void** showDialog(View view){  
 MyDialog dlg = MyDialog.*newInstance*(**"This is Hjs's text"**);  
 dlg.show(getSupportFragmentManager(),**"dialog"**);*//提供给它一个标签tag便于查找* }  
  
 **public void** ok(){  
 Toast.*makeText*(**this**,**"点击确定"**,Toast.***LENGTH\_LONG***).show();  
 }  
  
 **public void** showListFragment(View view){  
 String[] data = {**"黄继升"**,**"黄佳佳"**,**"黄楚华"**,**"黄春年"**};  
 MyListFragment list = MyListFragment.*newInstance*(data);  
 getSupportFragmentManager().beginTransaction()  
 .add(R.id.***containerLayout***,list)  
 .commit();*//add到Activity上去了，才会自动调用onAttach方法来和Activity做关联，并在onAttach方法内部完成适配器数据的绑定* }  
  
  
 **public static class** MyDialog **extends** DialogFragment{  
 **private** String **title**;  
  
 **public static** MyDialog newInstance(String title){  
 MyDialog dlg = **new** MyDialog();  
 Bundle args = **new** Bundle();  
 args.putString(**"title"**,title);  
 dlg.setArguments(args);  
 **return** dlg;  
 }  
  
 @Override  
 **public** Dialog onCreateDialog(Bundle savedInstanceState) {  
 **title**=getArguments().getString(**"title"**);  
 Dialog dlg = **new** AlertDialog.Builder(getActivity())  
 .setTitle(**title**)  
 .setIcon(R.drawable.***ic\_launcher\_background***)  
 .setPositiveButton(**"确定"**, **new** DialogInterface.OnClickListener() {  
 @Override  
 **public void** onClick(DialogInterface dialog, **int** which) {  
  
 ((Demo2Activity)getActivity()).ok();  
 *//或者 内部类DialogInterface.OnClickListener()调用外部类MyDialog的成员方法 getActivity()  
 //即：((Demo2Activity)MyDialog.this.getActivity()).ok(); 外部类.this.成员方法()* }  
 })  
 .setNegativeButton(**"取消"**,**null**)  
 .create();  
 **return** dlg;  
 }  
 }  
  
  
 **public static class** MyListFragment **extends** ListFragment{  
 **private** String[] **data**;  
 **public static** MyListFragment newInstance(String[] data){  
 MyListFragment list = **new** MyListFragment();  
 Bundle args = **new** Bundle();  
 args.putStringArray(**"data"**,data);  
 list.setArguments(args);  
 **return** list;  
 }  
  
 *//在关联Activity的方法时，顺便初始化数据，设置适配器来显示列表数据* @Override  
 **public void** onAttach(Context context) {  
 **super**.onAttach(context);  
 **data** = getArguments().getStringArray(**"data"**);  
 setListAdapter(**new** ArrayAdapter<String>(  
 getActivity(),  
 android.R.layout.***simple\_list\_item\_1***,  
 android.R.id.***text1***,  
 **data**));  
 }  
  
 *//也可以在onCreate方法里做数据的初始化  
/\* @Override  
 public void onCreate(@Nullable Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 data = getArguments().getStringArray("data");  
 setListAdapter(new ArrayAdapter<String>(  
 getActivity(),  
 android.R.layout.simple\_list\_item\_1,  
 android.R.id.text1,  
 data));  
 }\*/  
  
  
 //当点击ListView中的某一项时重写这一事件成员方法即可* @Override  
 **public void** onListItemClick(ListView l, View v, **int** position, **long** id) {  
 Toast.*makeText*(getActivity(),**data**[position]+**"，你好！"**,Toast.***LENGTH\_LONG***).show();  
 **super**.onListItemClick(l, v, position, id);  
 }  
 }  
  
}