第五章 UI进阶

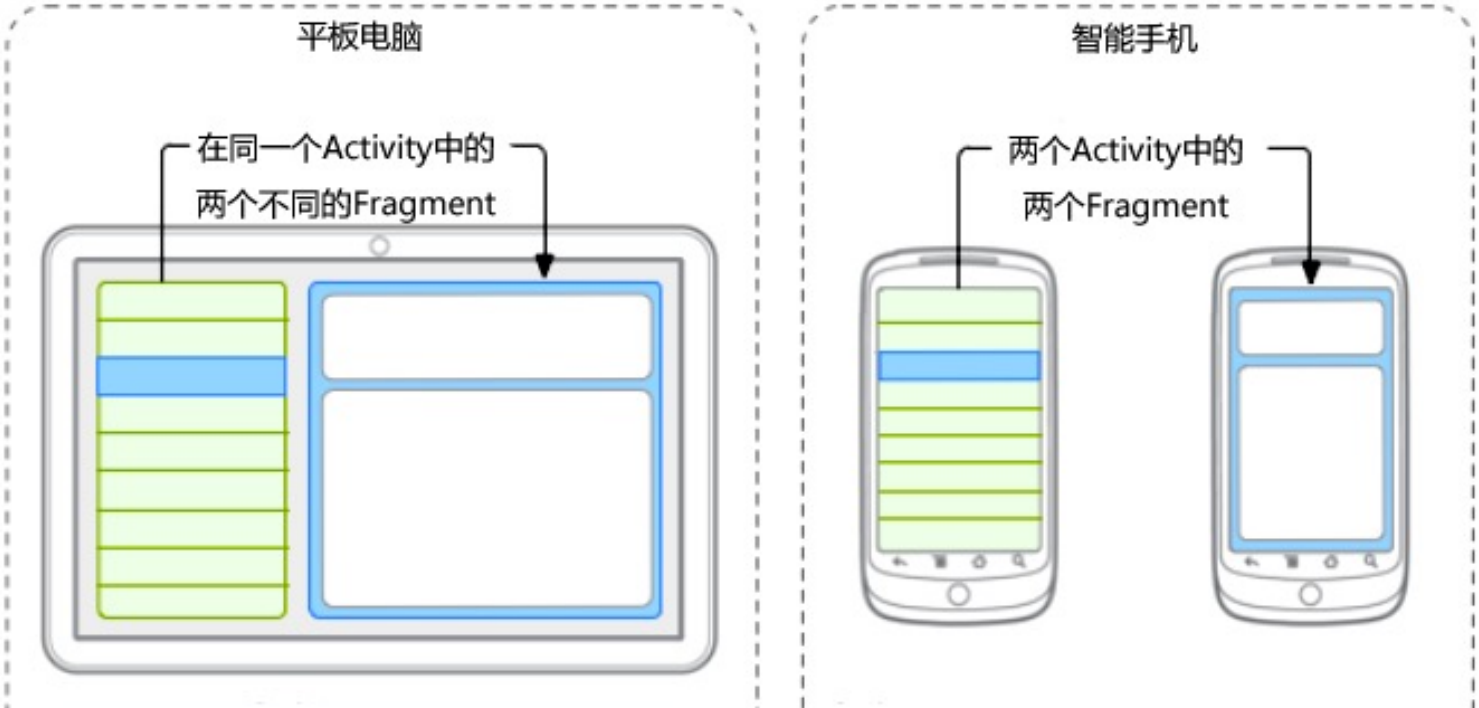
北京邮电大学 计算机学院
刘伟
w.liu@foxmail.com

本章重点

- 熟练使用Fragment
- 掌握Menu
- 掌握AdapterView组件
- 掌握Adapter的使用
- 精通ListView组件

5.1 Fragment

- Android从3.0开始引入Fragment（碎片）
- 允许将Activity拆分成多个完全独立封装的可重用的组件
- 每个组件拥有自己的生命周期和UI布局
- 为不同型号、尺寸、分辨率的设备提供统一的UI设计方案



5.1 Fragment

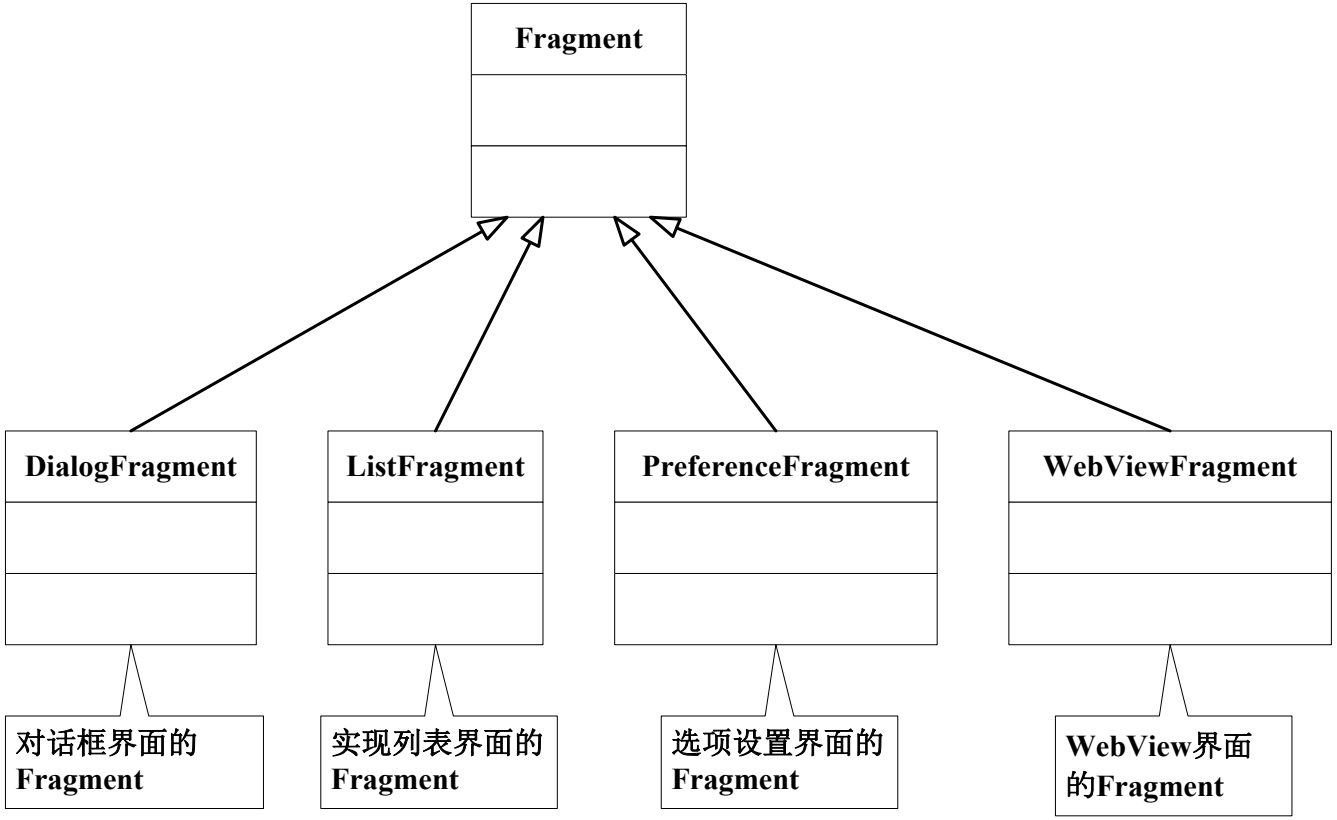
- Fragment是依赖于Activity的，不能独立存在的
- 一个Activity里可以有多个Fragment。
- 一个Fragment可以被多个Activity重用。
- Fragment有自己的生命周期，并能接收输入事件。
- 我们能在Activity运行时动态地添加或删除Fragment。

5.1 Fragment

- 通常情况下，创建Fragment需要继承Fragment的基类，并至少应实现onCreate()、onCreateView()和onPause()三个生命周期的回调函数
  - onCreate()函数是在Fragment创建时被调用，用来初始化Fragment中的必要组件
  - onCreateView()函数是Fragment在用户界面上第一次绘制时被调用，并返回Fragment的根布局视图
  - onPause()函数是在用户离开Fragment时被调用，用来保存Fragment中用户输入或修改的内容
- 如果仅通过Fragment显示元素，而不进行任何的数据保存和界面事件处理，则可仅实现onCreateView()函数

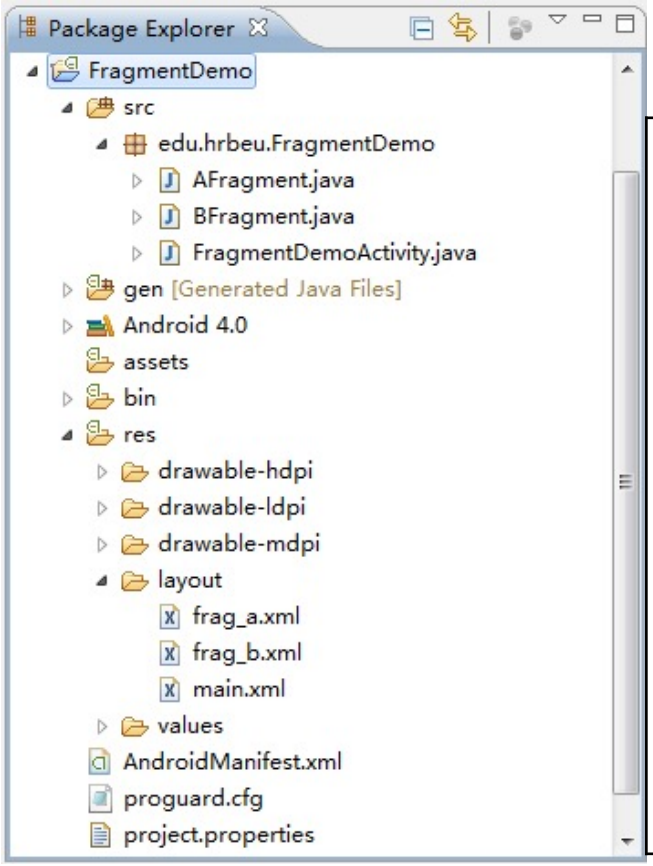
5.1.1 使用Fragment

- 自定义的Fragment必须继承Fragment类或其子类



5.1.1 使用Fragment-创建Fragment

- 将Fragment加载到Activity中主要有两种方式：
  - 把Fragment添加到Activity的布局文件中
  - 在Activity的代码中动态添加Fragment



- main.xml文件是Activity的布局文件，两个Fragment在界面上的位置关系就在这个文件中进行的定义

```
1.  <LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
2.      android:orientation="horizontal"
3.      android:layout_width="match_parent"
4.      android:layout_height="match_parent">
5.      <fragment android:name ="edu.bupt.FragmentDemo.AFragment"
6.              android:id="@+id/fragment_a"
7.              android:layout_weight="1"
8.              android:layout_width="0px"
9.              android:layout_height="match_parent" />
10.      <fragment android:name ="edu.bupt.FragmentDemo.BFragment"
11.             android:id="@+id/fragment_b"
12.             android:layout_weight="1"
13.             android:layout_width="0px"
14.             android:layout_height="match_parent" />
15. </LinearLayout>
```

5.1.1 使用Fragment-创建Fragment

- FragmentDemoActivity是该示例主界面的Activity，加载了main.xml文件声明的界面布局
- FragmentDemoActivity.java文件的完整代码如下：

```
1.  public class FragmentDemoActivity extends Activity {
2.      @Override
3.      public void onCreate(Bundle savedInstanceState) {
4.          super.onCreate(savedInstanceState);
5.          setContentView(R.layout.main);
6.      }
7.  }
```

- Android系统会根据代码第5行的内容加载界面布局文件main.xml，然后通过main.xml文件中对Fragment所在的“包+类”的描述，找到Fragment的实现类，并调用类中的onCreateView()函数绘制界面元素

5.1.1 使用Fragment-创建Fragment

- AFragment.java文件的核心代码如下：

```
1.  public class AFragment extends Fragment{
2.  @Override
3.      public View onCreateView(LayoutInflater inflater, ViewGroup container,
4.                               Bundle savedInstanceState) {
5.          return inflater.inflate(R.layout.frag_a, container, false);
6.      }
7.  }
```

- AFragment中只实现了onCreateView()函数(代码第3行)，返回值是AFragment的视图
- 代码第5行使用inflate()函数，通过指定资源文件R.layout.frag_a，获取到AFragment的视图

5.1.1 使用Fragment-创建Fragment

- 将Fragment加载到Activity中主要有两种方式：
  - 把Fragment添加到Activity的布局文件中
  - 在Activity的代码中动态添加Fragment

5.1.1 使用Fragment-管理Fragment

- 通过FragmentManager实现管理Fragment对象的管理
- 通过getFragmentManager()获取FragmentManager对象
- FragmentManager能够完成以下三方面的操作：
  - 通过findFragmentById()或findFragmentByTag()方法，来获取Activity中已存在的Fragment对象
  - 通过popBackStack()方法将Fragment从Activity的后退栈中弹出
  - 通过addOnBackStackChangedListerner()方法来注册一个侦听器以监视后退栈的变化

5.1.1 使用Fragment-管理Fragment

- 获取FragmentTransaction对象

  注意：FragmentTransaction被称作Fragment事务，与数据库事务类似，Fragment事务代表了Activity对Fragment执行的多个改变操作。

- 使用FragmentTransaction

```
//创建一个新的Fragment对象
Fragment newFragment=new ExampleFragment();
//通过FragmentManager获取Fragment事务对象
FragmentTransaction transaction=getFragmentManager().beginTransaction();
//通过replace()方法把fragment_container替换成新的Fragment对象
transaction.replace(R.id.fragment_container,newFragment);
//添加到回退栈
transaction.addToBackStack(null);
//提交事务
transaction.commit();
```

5.1.1 使用Fragment-管理Fragment

- 事务中动作的执行顺序可以随意，但需注意以下三点：
  - 程序的最后必须调用commit()方法
  - 程序中添加了多个Fragment对象，显示的顺序跟添加顺序一致
  - 当删除Fragment对象时，在没有调用addToBackStack()方法情况下，Fragment对象会被销毁

注意：调用commit()后，事务并不会马上提交，而是会在Activity的UI线程中等待直到线程能执行的时候才执行。

5.1.1 使用Fragment-与Activity通讯

- Fragment获取其所在的Activity中的组件

```
View listView=getActivity().findViewById(R.id.list);
```

- Activity获取指定Frament实例

```
ExampleFragment fragment = (ExampleFragment)getFragmentManager()
                .findFragmentById(R.id.example_fragment)
```

- 在Fragment中定义回调接口

```
public static class FragmentA extends ListFragment {
......省略
    //Activity必须实现下面的接口
    public interface OnNewsSelectedListener{
    //传递当前被选中的标题的id
    public void onNewsSelected(long id);
    }
......省略
}
```

5.1.1 使用Fragment-与Activity通讯

- 使用onAttach()方法检查Activity是否实现回调接口

```
public static class FragmentA extends ListFragment {
    OnNewsSelectedListener mListener;
    ......省略
    @Override
    public void onAttach(Activity activity){
        super.onAttach(activity);
        try{
            mListener =(OnNewsSelectedListener)activity;
            }catch(ClassCastException e){
                throw new ClassCastException(activity.toString()
                        +"必须继承接口 OnNewsSelectedListener");
        }
    }
    ......省略
}
```

5.1.1 使用Fragment-与Activity通讯

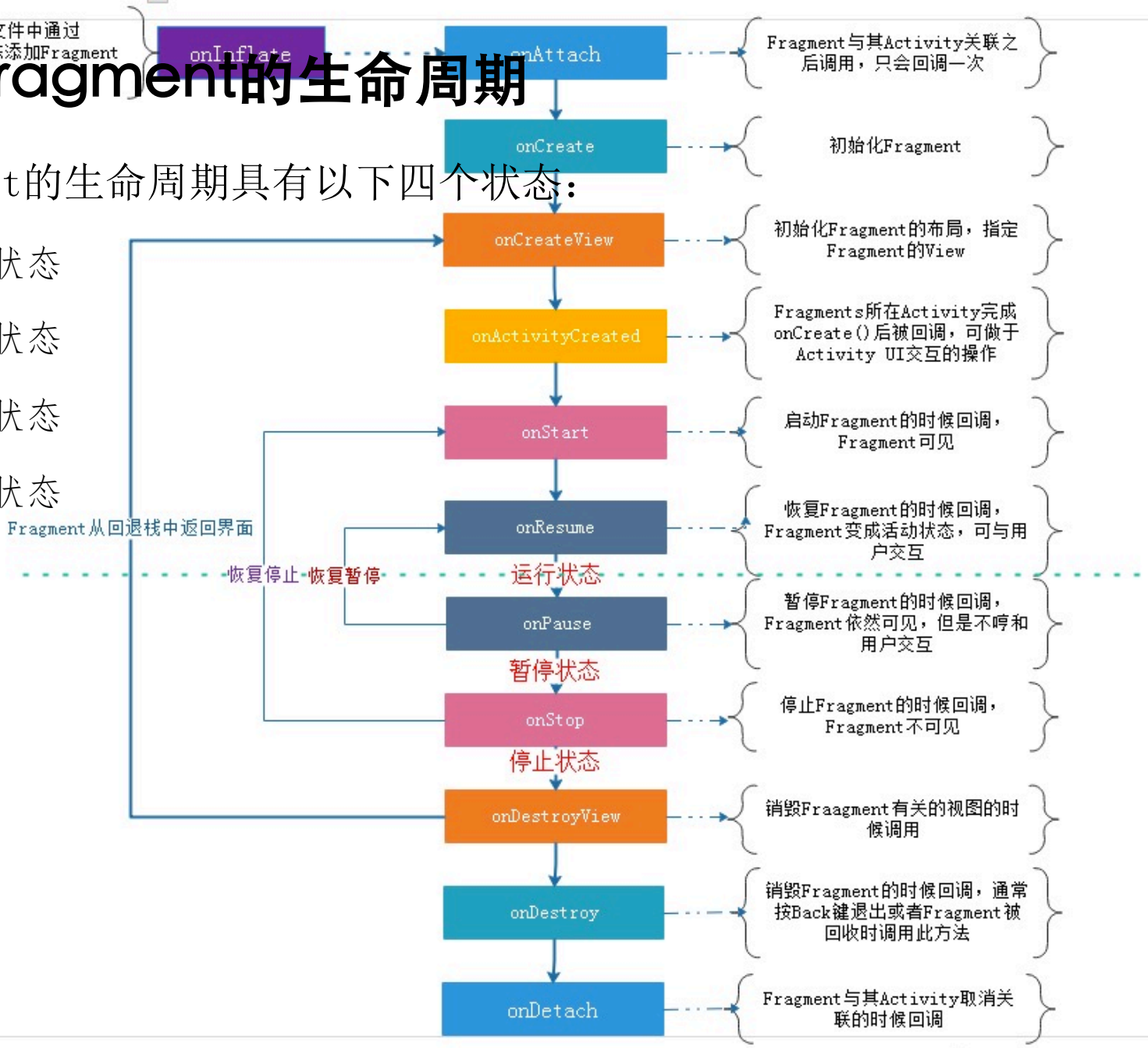
- Fragment与Activity共享事件

```
public static class FragmentA extends ListFragment {
    OnNewsSelectedListener mListener;
    ......省略
    @Override
    public void onListItemClick(ListView l,View v,int position,long id){
        mListener.onNewsSelected(id);
    }
    ......省略
}
```

注意：在数据传递时，也可以直接把数据从FragmentA传递给FragmentB，不过该方式降低了Fragment的可重用的能力。现在的处理方式只需要把发生的事件告诉宿主，由宿主决定如何处置，以便Fragment的重用性更好。

5.1.2 Fragment的生命周期

- Fragment的生命周期具有以下四个状态：
  - 活动状态
  - 暂停状态
  - 停止状态
  - 销毁状态



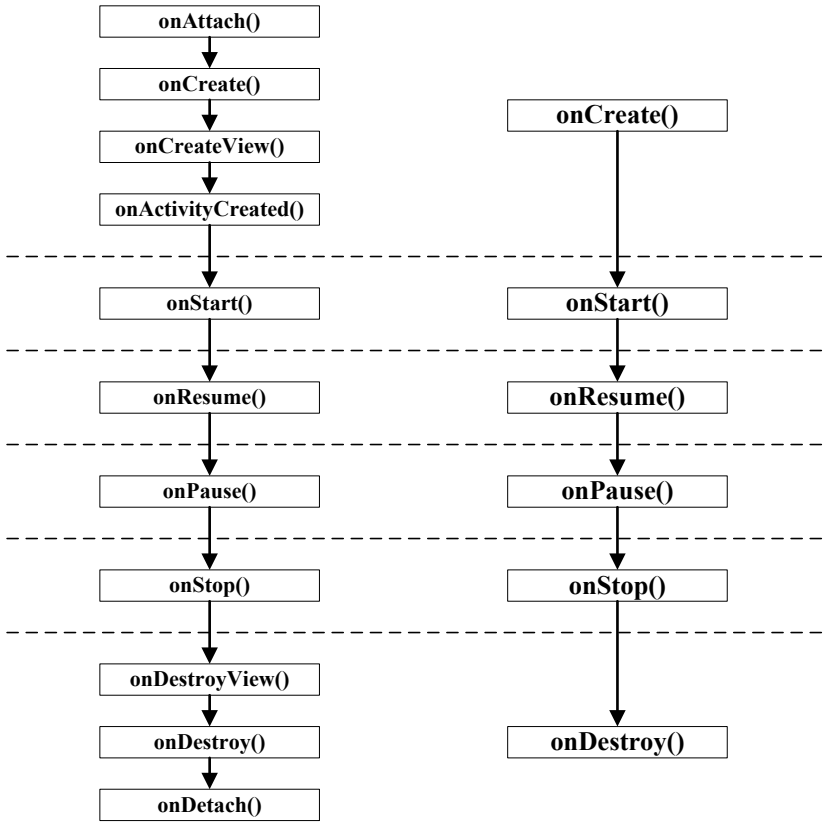
5.1.2 Fragment的生命周期

- Fragment生命周期中的方法

方法	功能描述
onAttach()	当一个Fragment对象关联到一个Activity时被调用。如果，需要使用Activity的引用或者使用Activity作为其他操作的上下文，将在此回调方法中实现
onCreate()	初始化创建Fragment对象时被调用
onCreateView()	当Activity获得Fragment的布局时调用此方法。当第一次绘制Fragment的UI时系统调用这个方法，该方法将返回一个View，如果Fragment不提供UI也可以返回null。注意，如果继承自ListFragment，onCreateView()默认的实现会返回一个ListView，所以不用自己实现。这个函数的Bundle参数和onCretate()函数的Bundle其实是同一个。
onActivityCreated()	当Activity对象完成自己的onCreate()方法时调用。可以在这个函数里面做和Activity UI交互的操作（因为Activity的onCreate()函数之后Activity的UI已经准备好了，可以UI交互）。
onStart()	Fragment对象在UI界面可见时调用
onResume()	Fragment对象的UI可以与用户交互时调用
onPause()	由Activity对象转为onPause状态时调用
onStop()	有组件完全遮挡，或者宿主Activity对象转为onStop状态时调用
onDestroyView()	Fragment对象清理View资源时调用，即移除Fragment中的视图
onDestroy()	Fragment对象完成对象清理View资源时调用
onDetach()	当Fragment被从Activity中删掉时被调用

5.1.2 Fragment的生命周期

- Fragment和Activity两者之间生命周期的关系
  - Activity直接影响其所包含的Fragment的生命周期
  - Fragment的回调方法要比Activity多，多出的方法主要用于与Activity的交互
  - 当Activity进入运行状态时（即running状态），才允许添加或删除Fragment
  - 有当Activity处于resumed状态时，Fragment的生命周期才能独立运转
  - 其他阶段依赖于Activity的生命周期



5.1.2 Fragment的生命周期-静态方式

- Fragment的生命周期调用过程：
  - 当首次展示布局页面时，其生命周期方法调用的顺序是：
    - onAttach()
    - onCreate()
    - onCreateView()
    - onActivityCreated()
    - onStart()
    - onResume()
  - 当关闭手机屏幕或者手机屏幕变暗时，其生命周期方法调用的顺序是：
    - onPause()
    - onStop()

5.1.2 Fragment的生命周期-静态方式

- Fragment的生命周期调用过程：
  - 当对手机屏幕解锁或者手机屏幕变亮时，其生命周期方法调用的顺序是：
    - onStart()
    - onResume()
  - 当对Fragment所在屏幕按返回键时，其生命周期方法调用的顺序是：
    - onPause()
    - onStop()
    - onDestroyView()
    - onDestroy()
    - onDetach()

5.1.2 Fragment的生命周期-动态方式

- 通过重写Fragment生命周期的方法
- 在Activity代码中动态使用Fragment

演示讲解

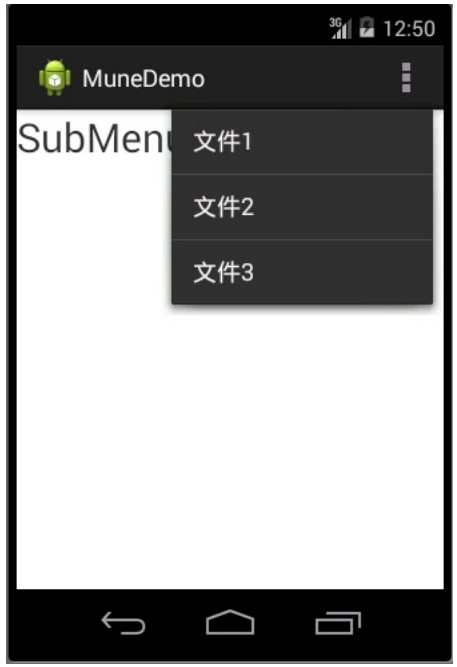
【代码5- 5】FragmentA.java

【代码5- 6】FragmentB.java

【代码5- 7】FragmentLifecircleActivity.java

5.2.1 Menu菜单

选项菜单

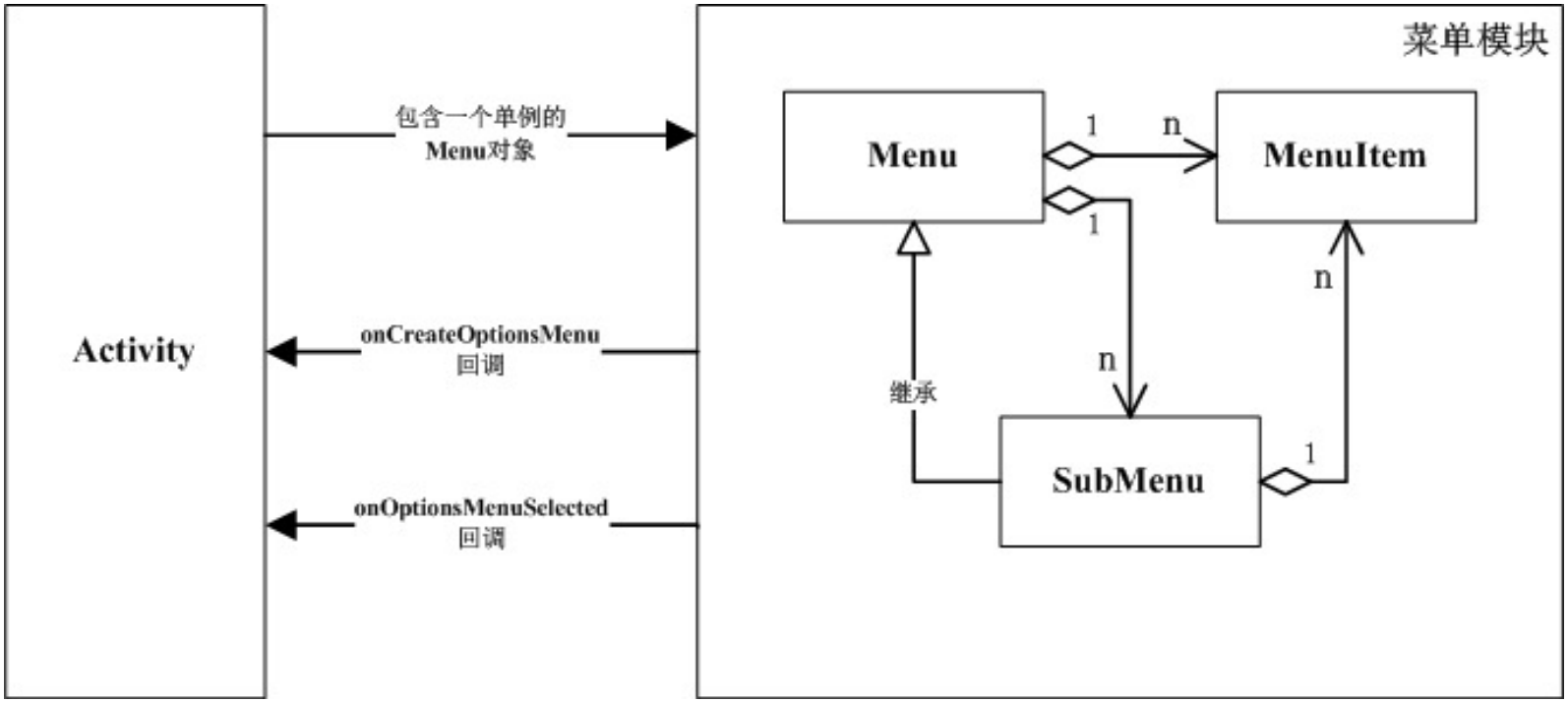


子菜单

上下文菜单

5.2.1 Menu菜单

- 系统创建菜单的方法主要有以下两种：
  - onCreateOptionsMenu()：创建选项菜单
  - onCreateContextMenu()：创建上下文菜单上下文菜单



5.2.1 Menu菜单-Options Menu选项菜单

- 在xml文件中定义布局文件
- 重写onCreateOptions(或Context)Menu，创建目录
- 重写onOptions(或Context)ItemSelected，响应目录的点击事件

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
        <item android:id="@+id/menu_id"
        android:title="menu_title" />
        ......
</menu>
```

- onCreateOptionsMenu(Menu menu)

  创建OptionsMenu
- onOptionsItemSelected(MenuItem Item)

  监听OptionsMenu的点击事件

  - 覆写onCreateContextMenu或onCreateOptionsMenu时，一定要写getMenuInflater.inflate(R.Menu.你需要的xml文件 ， menu）
  - 在覆写Selected方法时，常使用getItemId和switch方法
  - 在onStart中使用registerForContextMenu
  - 在onStop中使用unRegisterForContextMenu

5.2.1 Menu菜单-Options Menu选项菜单

```
@Override public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu,menu);
        //R.menu.menu是自己创建的目录xml文件 return true;
}
@Override public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        switch ( id ){
                case R.id.menu_1 :
                        Toast.makeText(MainActivity.this,"you click menu_1" ,Toast.LENGTH_LONG).show();
                        break;
                case R.id.menu_2 :
                        Toast.makeText(MainActivity.this,"you click menu_2" ,Toast.LENGTH_LONG).show();
                         break;
                default:
                        break;
        }
        return true;
}
```

5.2.1 Menu菜单-Options Menu选项菜单

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
        <item
                android:id="@+id/menu_1"
                android:title="111" />
        <item
                android:id="@+id/menu_2"
                android:title="222" />
</menu>
```

演示讲解

【代码5- 8】MenuDemoActivity.java

【代码5- 9】MenuDemoActivity.java

5.2.1 Menu菜单-响应菜单项

- Android为菜单提供了两种响应方式：
  - onOptionsItemSelected()方法
  - onMenuItemSelected()方法
- onMenuItemSelected与onOptionsItemSelected区别：
  - onMenuItemSelected()：当选择选项菜单或上下文菜单都会触发该事件处理方法
  - onOptionsItemSelected()：该方法只在选项菜单被选中时才会被触发

注意：如果Activity中同时重写onMenuItemSelected()和onOptionsItemSelected()方法时，当点击同一个菜单项时，将先调用onMenuItemSelected()方法，然后调用onOptionsItemSelected()方法性。

5.2.1 Menu菜单-SubMenu子菜单

- 创建子菜单的步骤：
  - ① 重写Activity类的onCreateOptionsMenu()方法
  - ② 调用Menu的addSubMenu()方法添加子菜单
  - ③ 调用SubMenu的add()方法为子菜单添加菜单项
  - ④ 重写Activity类的onOptionsItemSelected ()方法

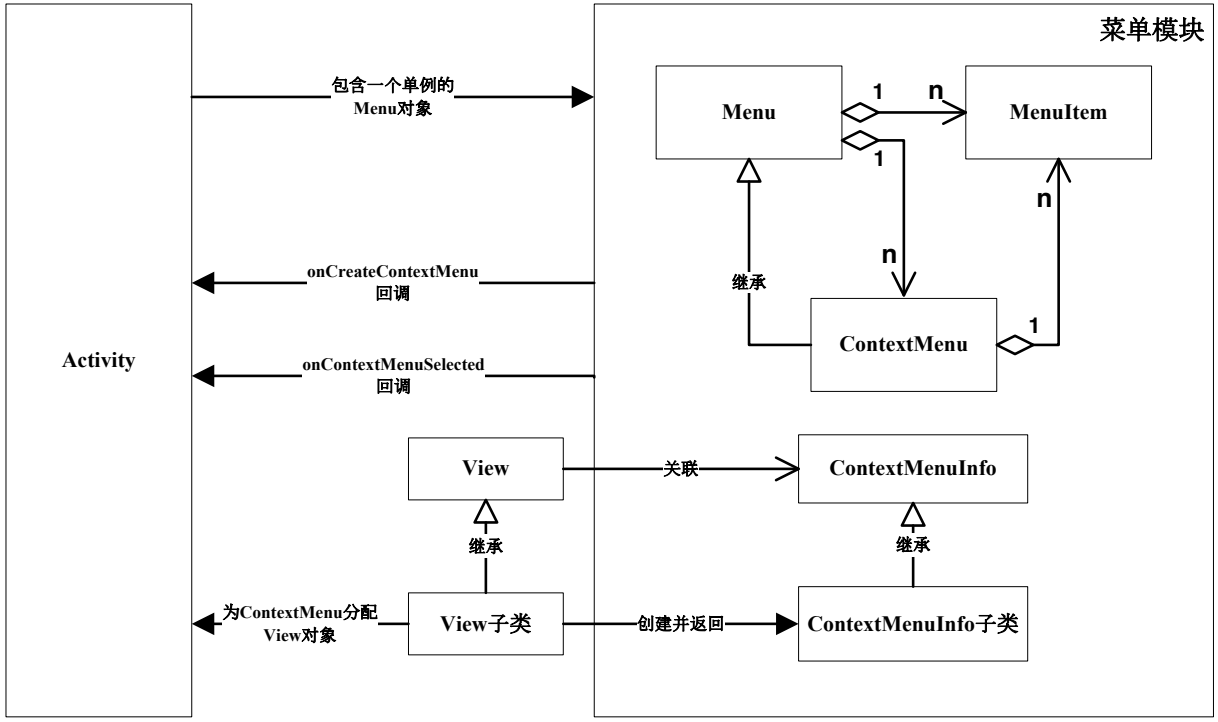
演示讲解

【代码5- 11】SubMenuDemoActivity.java

注意：当调用setIcon()方法设置图标时，图标无法显示是因为在MenuBuilder的optionalIconsVisible属性默认为false。当要显示图标时，需要通过反射机制调用MenuBuilder对象的setOptionalIconsVisible()方法，将其设置为true即可。

5.2.1 Menu菜单-ContextMenu上下文菜单

- 上下文菜单是通过调用ContextMenu接口中的方法来实现



5.2.1 Menu菜单-ContextMenu上下文菜单

- onCreateContextMenu()方法来生成ContextMenu对象

```
onCreateContextMenu(ContextMenu menu, View v,ContextMenu.ContextMenuInfo menuInfo)
```

注意：当需要传递额外信息时，需要重写getContextMenuInfo()方法，并返回一个带有数据的ContextMenuInfo实现类对象。

- 创建上下文菜单的步骤：

① 通过registerForContextMenu()方法为ContextMenu分配一个View对象

② 通过onCreateContextMenu()创建一个上下文对象

讲师演示讲解

【代码5- 12】ContextMenuDemoActivity.java

5.2.1 Menu菜单 - 使用XML资源生成菜单

- 使用XML资源生成菜单项的步骤：

① 在res目录中创建menu子目录

② 在menu子目录中创建一个Menu Resource file（XML文件）

③ 使用XML文件的资源ID，在Activity中将XML文件中所定义的菜单元素添加到menu对象中

④ 使用XML文件的资源ID，在Activity中将XML文件中所定义的菜单元素添加到menu对象中

讲师演示讲解

【代码5- 13】context_menu.xml

5.2.1 Menu菜单 - 使用XML资源生成菜单

- 在XML资源文件中完成对菜单项或分组等操作：
  - 资源文件实现子菜单

```
<item android:title="系统设置">
    <menu>
        <item android:id="@+id/mi_display_setting"android:title="显示设置"/>
        <item android:id="@+id/mi_network_setting"android:title="网络设置"/>
        <!—其他菜单项 -->
        </menu>
</item>
```

  - 为菜单项添加图标

```
<item android:id="@+id/mi_exit" android:title="退出"
        android:icon="@drawable/exit"/>
```

  - 设置菜单项的可选策略

```
<group android:id="..." android:checkableBehavior="all">
        <!-- 菜单项 -->
</group>
```

5.2.1 Menu菜单 - 使用XML资源生成菜单

- 在XML资源文件中完成对菜单项或分组等操作：
  - 使用android:checked设置特定菜单项

```
<item android:id="..." android:title="sometitle" android:checked="true"/>
```

  - 为菜单项添加图标

```
<item android:id="..." android:title="sometitle" android:enabled="false"/>
```

  - 设置菜单项可见/不可见

```
<item android:id="..." android:title="sometitle" android:visible="false"/>
```

5.3.1 ListView列表视图

- ListView通常具有两个职责：
  - 将数据填充到布局，以列表的方式来显示数据
  - 处理用户的选择、点击等操作SimpleCursorAdapter类
- 通常创建ListView有以下两种方式：
  - 直接使用ListView进行创建
  - 使用Activity继承ListActivity，实现ListView对象的获取

XML属性	功能描述
android:divider	设置列表的分隔条
android:dividerHeight	用来指定分隔条的高度
android:entries	指定一个数组资源
android:footerDividersEnabled	各个footer之间绘制分隔条
android:headerDividersEnabled	各个header之间绘制分隔条

5.3.1 ListView列表视图

- ListView从AbsListView中继承的属性

XML属性	功能描述
android:cacheColorHint	用于设置该列表的背景始终以单一、固定的颜色绘制
android:choiceMode	为视图指定选择的行为
android:drawSelectorOnTop	如果为true，选中的列表项将会显示在上面
android:fastScrollEnabled	用于设置是否允许使用快速滚动滑块
android:listSelector	设置选中项显示的可绘制对象
android:scrollingCache	设置在滚动时是否使用绘制缓存，默认为true。
android:smoothScrollbar	列表会使用更精确的基于条目在屏幕上的可见像素高度的计算方法
android:stackFromBottom	设置是否将列表项从底部开始显示
android:textFilterEnabled	设置是否对列表项进行过滤
android:transcriptMode	设置该组件的滚动模式

5.3.1 ListView列表视图

- 使用ListView步骤：

① 准备ListView所要显示的数据

② 使用数组或List集合存储数据

③ 创建适配器，作为列表项数据源

④ 将适配器对象添加到ListView，并进行展示

5.3.1 ListView列表视图

- ListView组件本身默认的id为@id/android:list
- 直接调用getListView()

演示讲解

【代码5- 19】ListViewSimpleDemoActivity.java

- 在ListActivity中显示其他组件的步骤：

① 先定义Activity的布局文件，在布局UI界面时先增加其他组件，再添加一个ListView组件用于展示数据

② 在Activity中通过setContentView()方法来添加布局对象

5.3.2 ListView列表视图

演示讲解

【代码4- 20】listview_demo.xml

【代码4- 21】ListViewDemoActivity.java

注意：通过继承ListActivity来实现ListView时，当用户也定义了一个id为@id/android:list的ListView，与ListActivity中的默认ListView组件id一致，则使用setContentView()方法可以指定用户定义的ListView作为ListActivity的布局，否则会使用系统提供的ListView作为ListActivity的布局。

5.3.3 AdapterView与Adapter

- AdapterView实现过程类似于MVC架构
- AdapterView实现过程：

① 控制层： Adapter适配器承担了控制层的角色

② 视图层： AdapterView用于将前端显示和后端数据分离

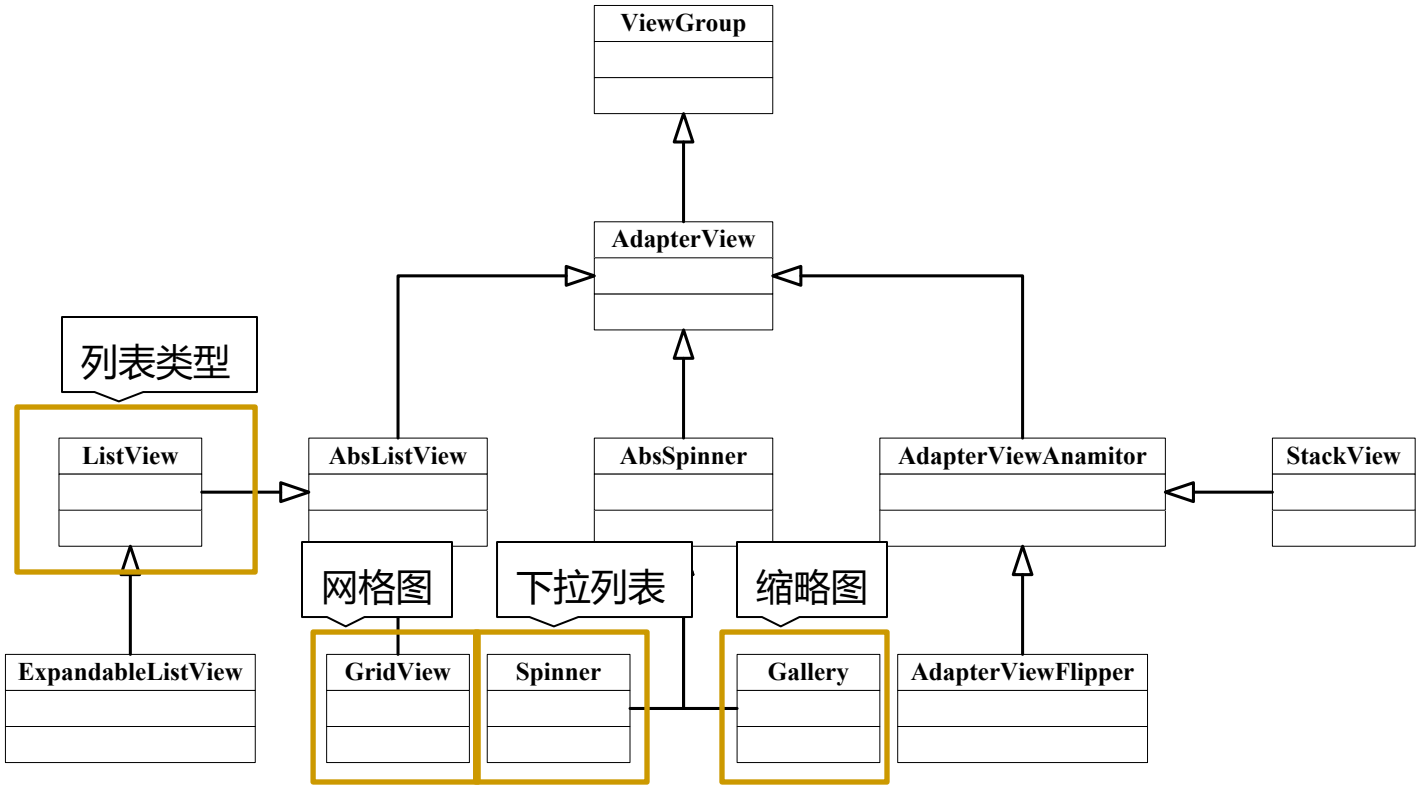
③ 模型层：数组、XML文件等形式的数据

5.3.3 AdapterView与Adapter - AdapterView组件

- AdapterView具有以下特征：
  - AdapterView继承了ViewGroup，其本质上是容器
  - AdapterView可以包括多个“列表项”，并将“列表项”以合适的形式显示出来
  - AdapterView所显示的“列表项”是由Adapter提供，通过AdapterView的setAdapter()方法来设置Adapter适配器。

5.3.3 AdapterView与Adapter - AdapterView组件

- AdapterView及其子类的继承关系：



注意：通常将ListView、GridView、Spinner和Gallery等AdapterView子类作为容器，然后使用Adapter为容器提供“列表项”，AdapterView负责采用合适的方式显示这些列表项。

5.3.3 AdapterView与Adapter - AdapterView组件

- Adapter的常用子接口：
  - ListAdapter接口
  - BaseAdapter抽象类
  - SimpleCursorAdapter类
  - ArrayAdapter类
  - SimpleAdapter类

注意：Adapter对象扮演着桥梁的角色，通过桥梁连接着AdapterView和所要显示的数据。Adapter提供了一个连通数据项的途径，将数据集呈现到View中。

本章总结

- Fragment允许将Activity拆分成多个完全独立封装的可重用的组件，每个组件拥有自己的生命周期和UI布局
- 创建Fragment需要实现三个方法：onCreate()、onCreateView()和onPause()
- Fragment的生命周期与Activity的生命周期相似，具有以下状态：活动状态、暂停状态、停止状态和销毁状态
- Android中提供的菜单有如下几种：选项菜单、子菜单、上下文菜单和图标菜单等
- 在Android中提供了一种高级控件，其实现过程就类似于MVC架构，该控件就是AdapterView
- ListView（列表视图）是手机应用中使用非常广泛的组件，以垂直列表的形式显示所有列表项