第四章

手机平板要兼顾--探究碎片

主讲：王海

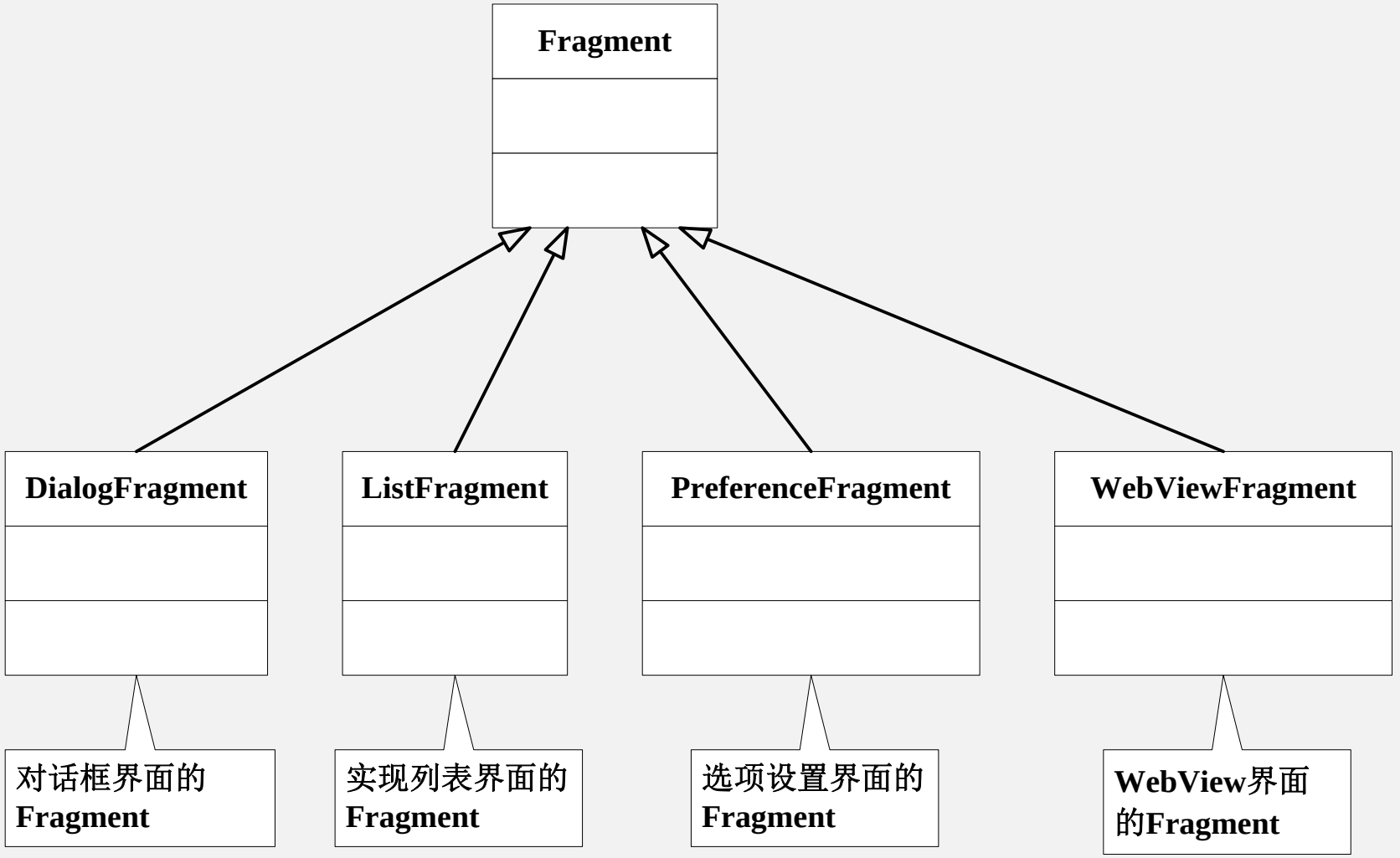
本章目标

- 能够熟练使用Fragment动态设计UI界面
- 能够熟练使用Menu和Toolbar组件
- 能够熟练使用AdapterView、ListView和GridView
- 掌握TabHost组件的使用

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

Fragment类及子类



创建Fragment

- 通常在创建Fragment时，需要实现三个方法：
  - onCreate()
  - onCreateView()
  - onPause()
- 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

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

管理Fragment

```
FragmentManager fragmentManager=getFragmentManager();
FragmentTransaction
fragmentTransaction=fragmentManager.beginTransaction();
```

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();
```

Fragment事务注意事项

- 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

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。

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
}
```

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
}
```

与Activity通讯

- Fragment与Activity共享事件

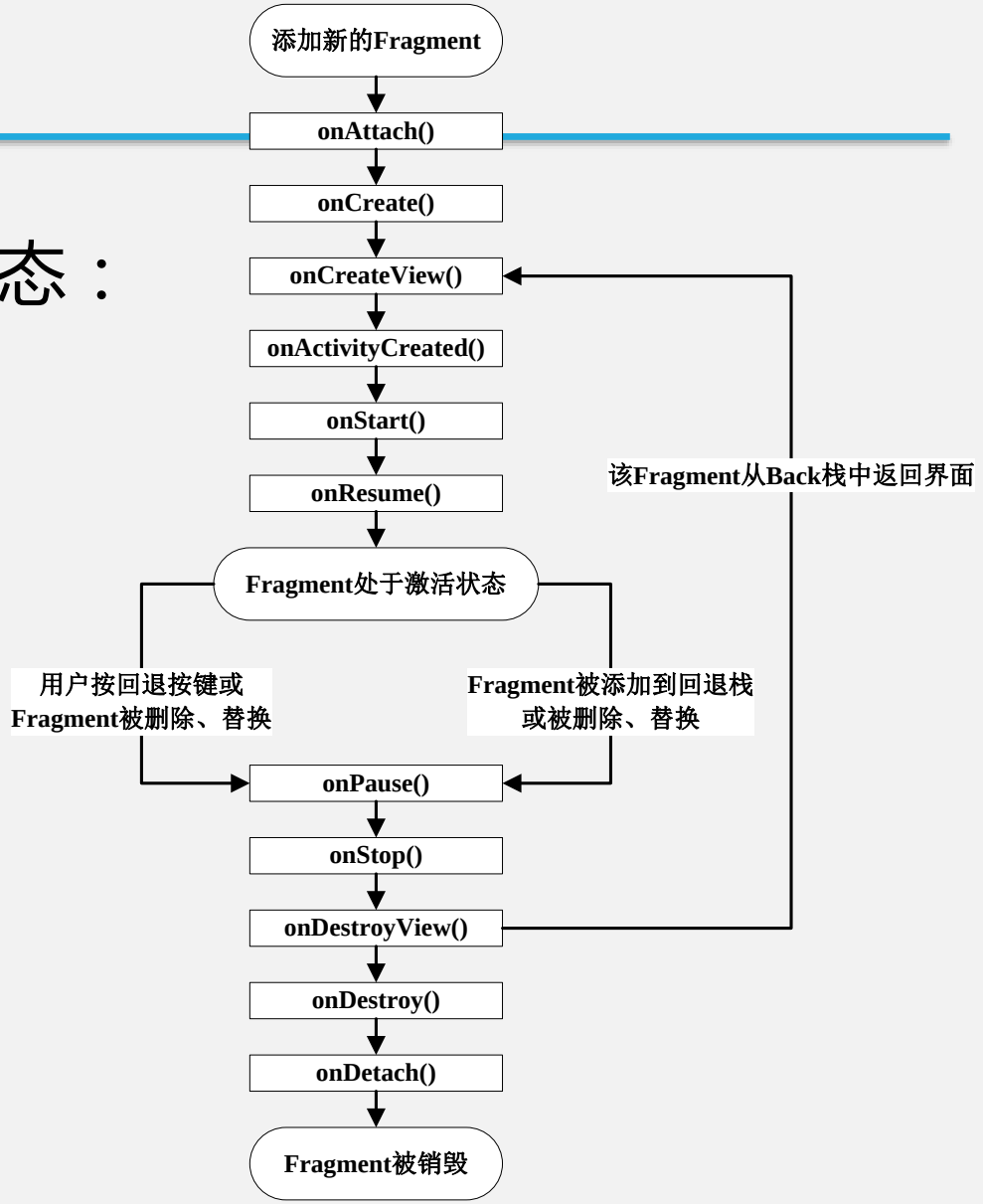
```
public static class FragmentA extends ListFragment {
     OnNewsSelectedListener mListener;
     ......省略
     @Override
     public void onListItemClick(ListView l,View v,int position,long
id){
    mListener.onNewsSelected(id);
     }
     ......省略
}
```

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。

Fragment的生命周期

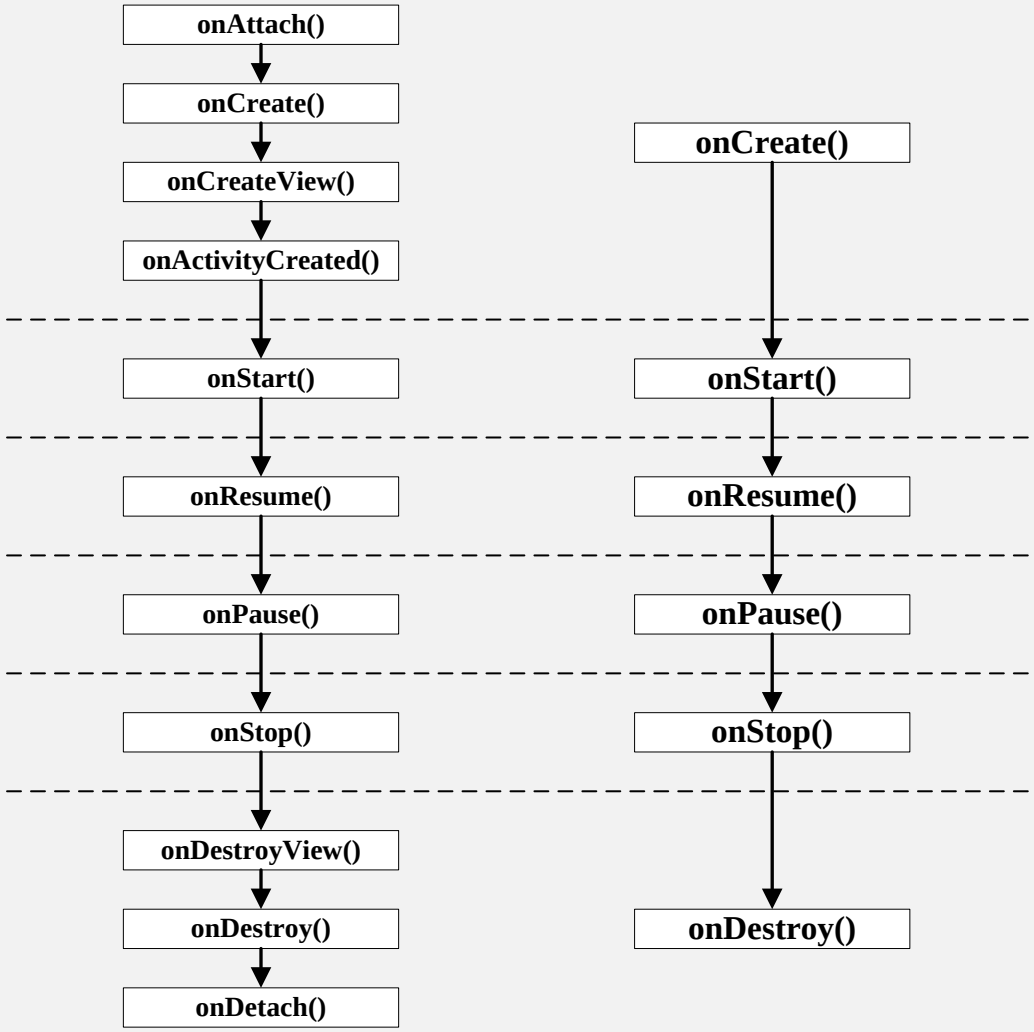
- Fragment的生命周期具有以下四个状态：
  - 活动状态
  - 暂停状态
  - 停止状态
  - 销毁状态



Fragment生命周期中的方法

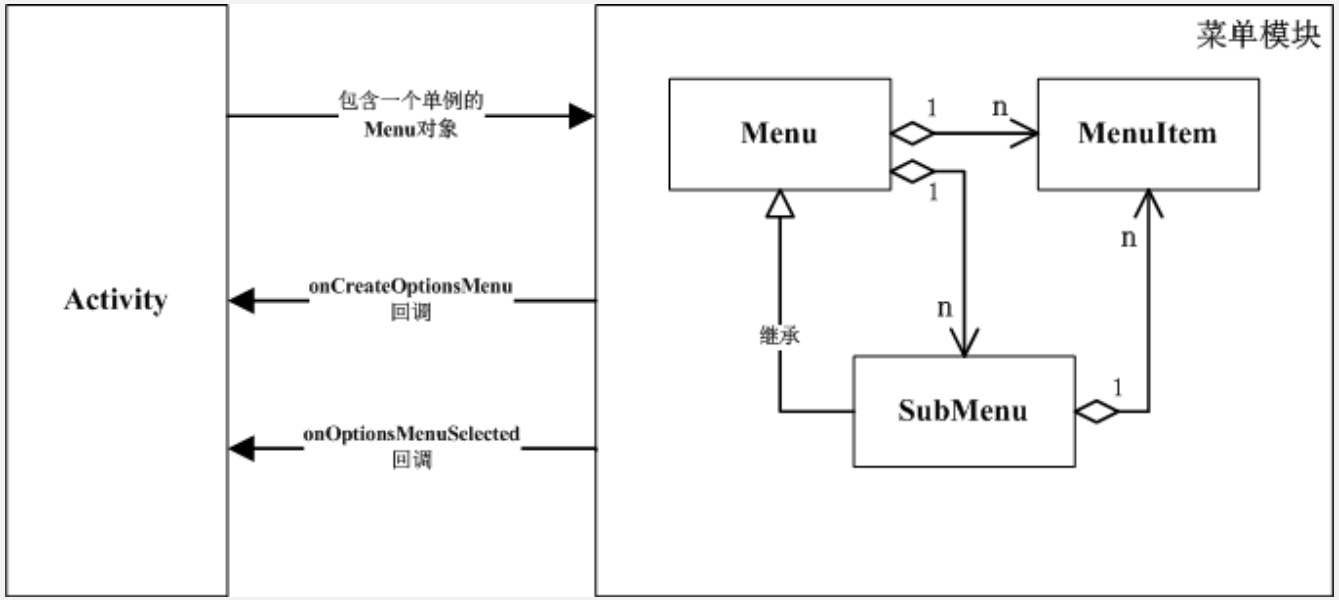
方法	功能描述
onAttach()	当一个Fragment对象关联到一个Activity时被调用
onCreate()	初始化创建Fragment对象时被调用
onCreateView()	当Activity获得Fragment的布局时调用此方法
onActivityCreated()	当Activity对象完成自己的onCreate()方法时调用
onStart()	Fragment对象在UI界面可见时调用
onResume()	Fragment对象的UI可以与用户交互时调用
onPause()	由Activity对象转为onPause状态时调用
onStop()	有组件完全遮挡，或者宿主Activity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

Fragment和Activity两者之间生命周期的关系



Menu菜单

- Android中提供的菜单有如下几种：
  - 选项菜单
  - 子菜单
  - 上下文菜单
  - 图标菜单
  - 扩展菜单



- 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

Options Menu选项菜单

- 使用onCreateOptionsMenu()回调方法对菜单进行初始化
- 使用onPrepareOptionsMenu()方法动态改变选项菜单的内容

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

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。

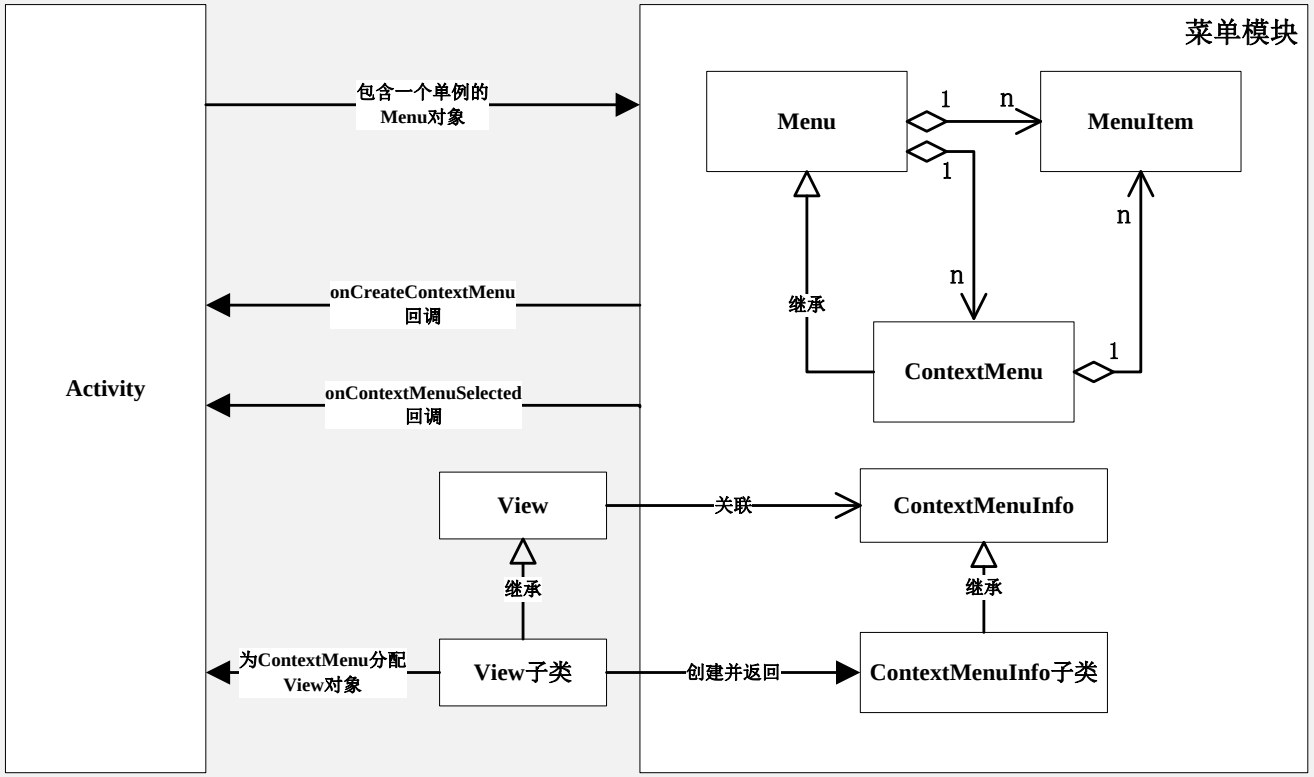
SubMenu子菜单

- 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

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。

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
- Menu、ContextMenu关系示意图



ContextMenu上下文菜单

- onCreateContextMenu()方法来生成ContextMenu对象

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

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

使用XML资源生成菜单

- 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

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>
```

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"/>
```

Toolbar操作栏

- Material Design风格的导航组件
- 取代Actionbar

方法	功能描述
setTitle(int resId)	设置标题
setSubtitle(int resId)	设置子标题
setTitleTextColor(int color)	设置标题字体颜色
setSubtitleTextColor(int color)	设置子标题字体颜色
setNavigationIcon(Drawable icon)	设置导航栏的图标
setLogo(Drawable drawable)	设置Toolbar的Logo图标

Toolbar的使用步骤

① 在build.gradle文件中添加对v7 appcompat库的支持

```
compile 'com.android.support:appcompat-v7:24.1.1'
```

② 在styles.xml文件中对原主题进行修改

```
<style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
```

③ 添加Toolbar组件

```
<android.support.v7.widget.Toolbar
    android:id="@+id/my_toolbar"
    android:layout_width="match_parent"
    android:layout_height="?attr/actionBarSize"
    android:background="?attr/colorPrimary" >
```

Toolbar的使用步骤

④ 显示Toolbar组件

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_my);
    Toolbar toolbar = (Toolbar)
findViewById(R.id.my_toolbar);
    setSupportActionBar(toolbar);
}
```

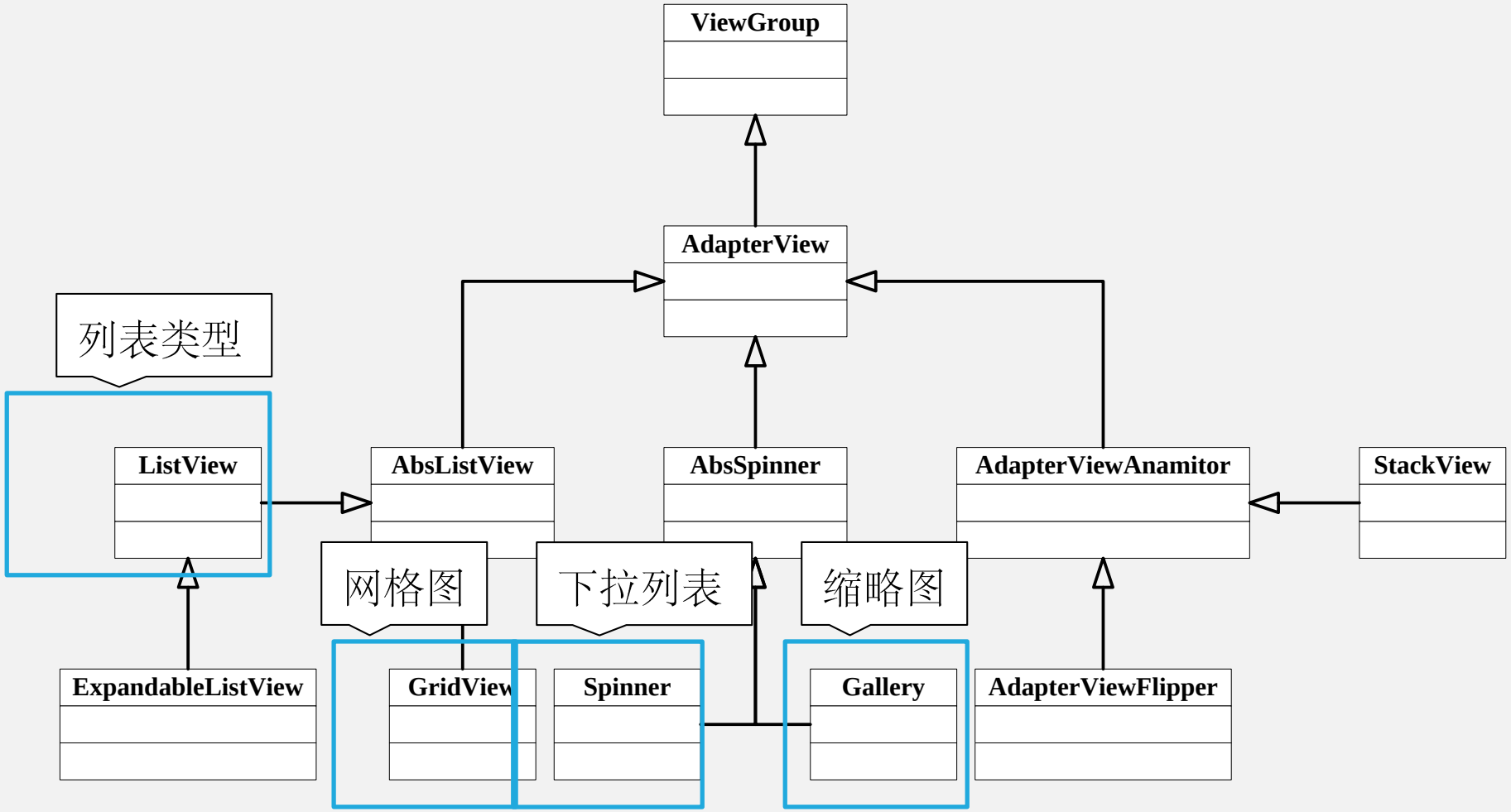
AdapterView与Adapter

- 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

AdapterView特征

- AdapterView继承了ViewGroup，其本质上是容器
- AdapterView可以包括多个“列表项
- 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。

AdapterView及其子类



AdapterView及其子类

Adapter的常用子接口：

- ListAdapter接口
- BaseAdapter抽象类
- SimpleCursorAdapter类
- ArrayAdapter类
- SimpleAdapter类

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。

ListView列表视图

- 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

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

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

GridView网格视图

- 用于按行和列的分布方式来显示多个组件
- 通过Adapter来提供显示数据

XML属性	功能描述
android:numColumns	设置列数
android:columnWidth	设置每一列的宽度
android:stretchMode	设置拉伸模式：
android:verticalSpacing	设置各个元素之间的垂直边距
android:horizontalSpacing	设置各个元素之间的水平边距

创建GridView的步骤

- 在布局文件中使用<GridView>元素来定义GridView组件
- 自定义一个Adapter，并重写其中的关键方法
- 注册列表选项的单击事件
- 创建Activity并加载对应的布局文件

TabHost

- 实现在窗口中放置多个标签页
- 通常需要与TabWidget、TabSpec组件结合使用
  - TabWidget组件用于显示TabHost标签页中上部和下部的按钮，点击按钮时切换选项卡
  - TabSpec代表选项卡界面，通过将TabSpec添加到TabHost中实现选项卡的添加
- TabHost创建、添加选项卡的方法：
  - newTabSpec(String tag)方法用于创建选项卡
  - addTab(tabSpec)方法用于添加选项卡

继承TabActivity时使用TabHost的步骤

① 定义布局：在XML文件中使用TabHost组件，并在其中定义一个FrameLayout选项卡内容

② 创建TabActivity：用于显示选项卡组件的Activity，需要继承TabActivity

③ 获取组件：通过getTabHost()方法获取TabHost对象

④ 创建选项卡：通过TabHost来创建一个选项卡

不继承TabActivity时使用TabHost的步骤

① 定义布局——在XML文件中使用TabHost组件

② 创建TabActivity——用于显示选项卡组件的Activity，需要继承TabActivity

③ 获取组件——通过findViewById()方法获取TabHost对象

④ 创建选项卡——通过TabHost来创建一个选项卡

TabActivity在Android 3.0以后已过时，推荐使用“不继承TabActivity的方式”使用TabHost。

WebView

- 用于在App中显示网页或开发用户自己的浏览器
- 使用Web检查器来调试HTML、CSS、Javascript等代码
- 可以对URL请求、页面加载、渲染以及页面的交互进行处理
- WebView具有以下几点功能：
  - WebChromeClient：辅助WebView实现与浏览器的交互动作
  - WebViewClient：帮助WebView处理各种通知、请求事件等
  - WebSettings：对WebView进行配置和管理
  - addJavascriptInterface()：将Java对象绑定到WebView中，以便JavaScript从页面中控制Java对象，实现WebView与HTML页面的交互

注意

在Android 4.3及以前版本WebView内部采用Webkit渲染引擎，在Android 4.4以上版本采用chromium渲染引擎来渲染View的内容。

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
- GridView用于在界面上按行、列分布的方式显示多个组件。