# Quanta基础知识-Activity详解、Intent使用、Fragment详解

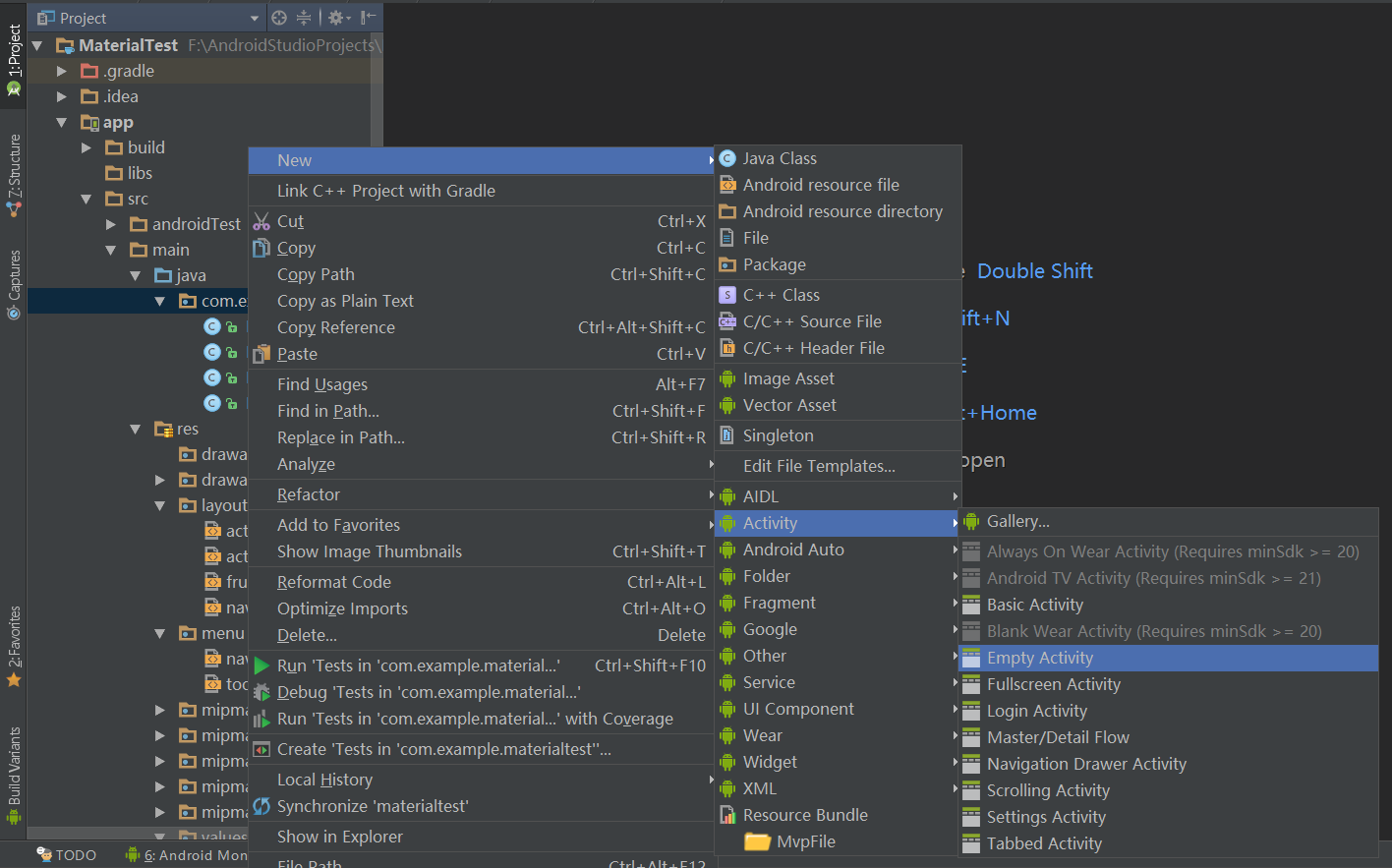
**Part 1：Activity详解**

1. 什么是活动

活动是一个可以包含用户界面的组件，主要用于和用户进行交互。一个应用程序可以包含多个活动。

1. 创建活动方法

①Android Studio自动创建（New→Activity→Empty Activity）



②新建Java Class继承AppCompatActivity/Activity，并在AndroidManifest文件中注册（Android四大组件用这种方法或者在eclipse上都要记得注册组件）



1. 在活动中使用Menu

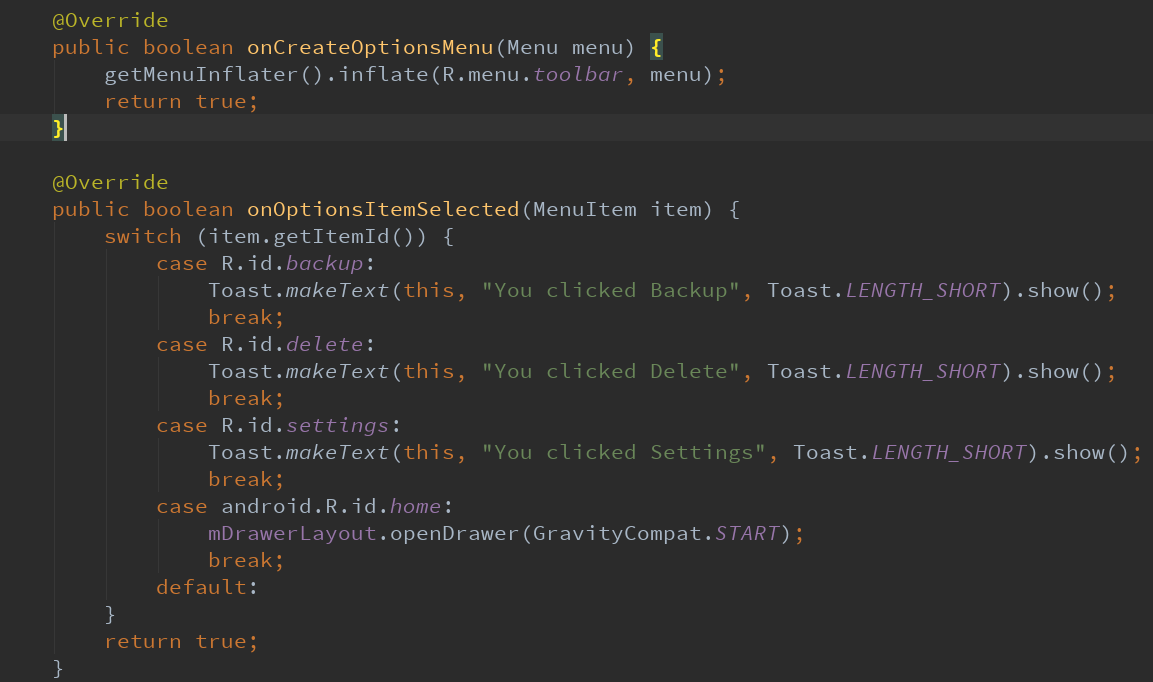
创建： res目录→New→Directory，输入文件名menu；在menu下新建New→Menu resource file

示例代码：



接着在Activity中来重写onCreateOptionMenu（）方法创建菜单、用onOptionItemSelected（）方法定义菜单的响应事件。

示例代码：



1. 销毁活动

finish（）方法

1. Activity的生命周期

onCreate()：在活动第一次被创建的时候调用

onStart()：在活动由不可见变为可见的时候调用

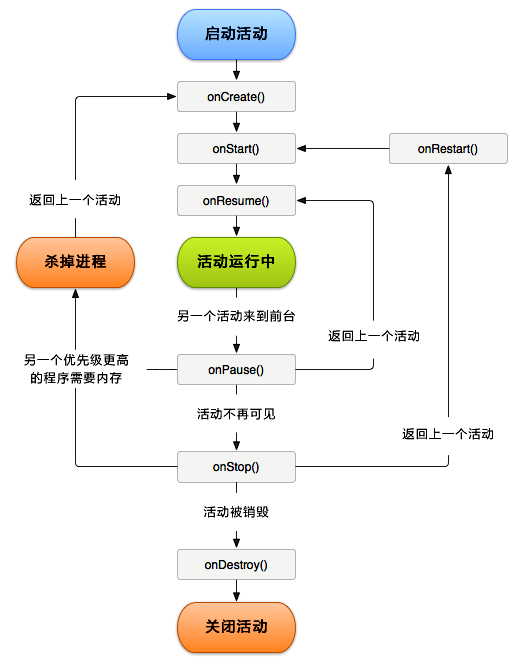
onResume()：在活动准备好和用户进行交互的时候调用

onPause()：暂停当前活动，在系统准备去启动或者恢复 另一个活动的时候调用

onStop()：在活动完全不可见的时候调用

onDestroy()：在活动被销毁之前调用

onRestart()：在活动由停止状态变为运行状态之前调用



1. Activity的启动模式

活动的启动模式一共有4中，分笔试standard、singleTop、singleTask、singleInstance，可以在AndroidManifest.xml中通过给<activity>标签指定android：launchMode属性来选择启动模式。

standard：（默认情况下）系统不会在乎这个活动是否在返回栈中存在，每次启动都会创建改活动的一个新的实例。

singleTop：在启动活动时如果返现返回栈的栈顶已经是该活动，则认为可以直接使用它，不会在创建新的活动实例。

singleTask：每次启动该活动是系统都会首先在返回栈中检查是否存在该活动的实例，如果发现已经存在则直接使用该实例，并且把在这个活动之上的所有活动统统出栈，如果没有发现就会创建一个新的活动实例。

singleInstance：指定为singleInstance模式的活动会启用一个新的返回栈来管理这个活动，实现其他程序和我们的程序可以共享这个活动的实例。

**Part 2：Intent的使用**

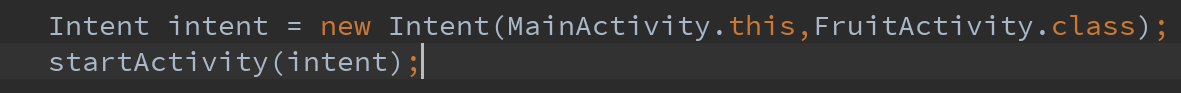
1. 什么是Intent

作用：从一个Activity跳转到另外一个Activity，也可以通过intent在不同组件之间传递数据。

1. 显式Intent

Intent有多个构造函数的重构。构造出一个合适的Intent，再把它放入对应函数中。

例：显式启动活动



1. 隐式Intent

隐式Intent通过在<activity>标签下配置<intent-filter>的内容，可以指定一系列更为抽象的action和category等信息，然后交由系统去分析这个Intent并帮我们找出合适的活动去启动。

例：指定首先启动的活动



只有<action>和<category>中的内容能够匹配上Intent中指定的action和category是，这个活动才能响应该Intent（不多不少原则）。每个Intent中只能指定一个action但却能指定多个category。需要注意的是，隐式Intent必须指定action。

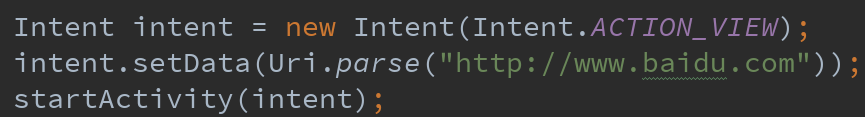
隐式启动MainActivity（"android.intent.category.DEFAULT"是一种默认的category，在调用startActivity（）方法时会自动将这个category添加到Intent中）





1. 更多Intent的应用

通过默认浏览器打开百度（Uri.parse（）将网址解析成一个Uri对象）



除http协议为，我们还可以指定很多其他协议，通过指定Intent的data属性，可以做到打开浏览器、相册、拨号界面等操作。

对应的，我们可以在<intent-filter>标签中在配置一个<data>标签，用于跟精确地指定当前活动能够响应审类型的数据。

android:scheme 用于指定数据的协议部分

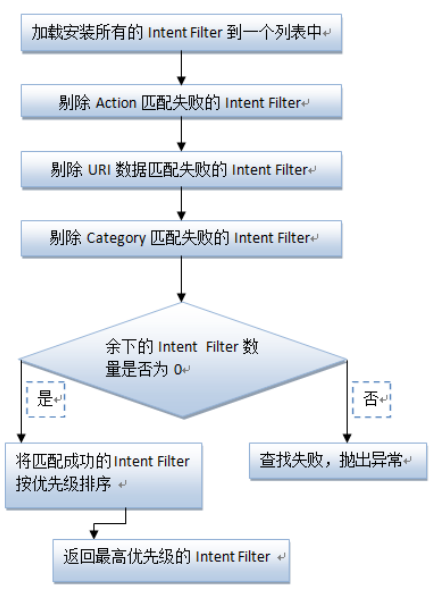
android:host 用于指定数据的主机名部分

android:port 用于指定数据的端口部分，一般紧随在主机名之后

android:path 用于指定主机名和端口名之后的部分

android:mimeType 用于指定可以处理的数据类型，允许使用通配符的方式进行指定

1. Intent的匹配规则



1. Intent与Bundle结合

Bundle提供了一系列类似Intent的方法用于保存数据，Intent还可以结合Bundle一起用于传递数据，首先可以把需要传递的数据都保存在Bundle对象中，然后在将Bundle对象方在Intent里，到了目标活动之后先从Intent中取出Bundle，在从Bundle中一一取出数据。

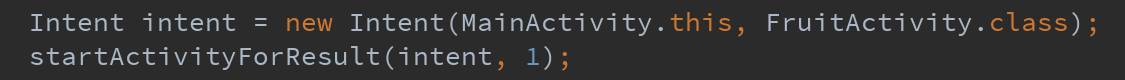
1. 向下一个活动传递数据

Intent的putExtra（键，值）和Intent的getExtra（键）

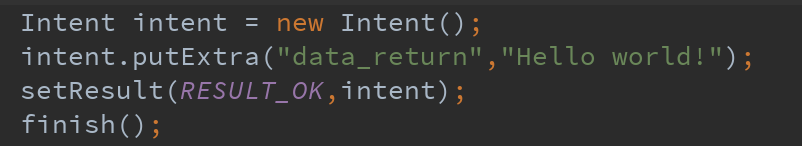
得到Intent实例的方法getIntent（）

1. 返回数据给上一个活动

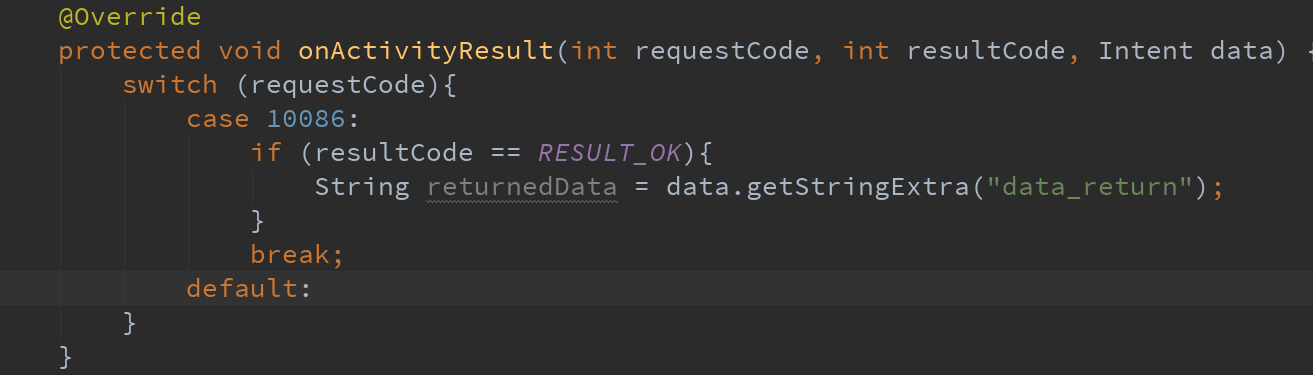
启动：startActivityForResult（intent，唯一请求码）



传输数据：setResult（RESULT\_OK/RESULT\_CANCELED,带有数据的Intent）



接受：onActivityResult（）方法，通过传进来的requestCode（启动时指定的唯一请求码）判断，进行逻辑处理



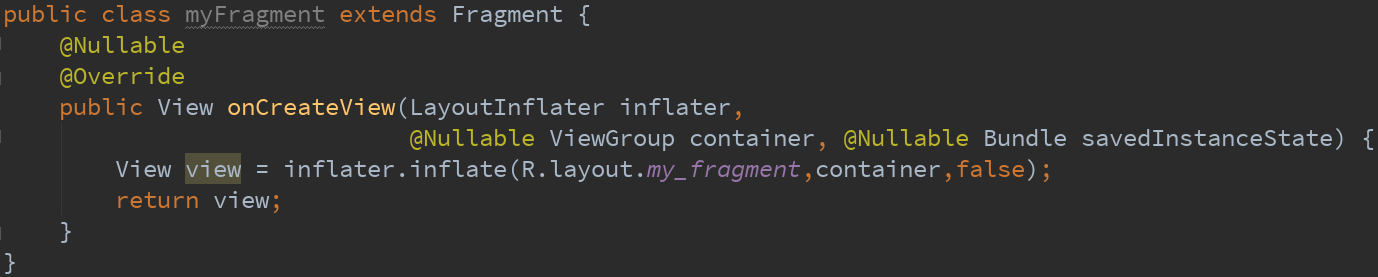
**Part 3：Fragment详解**

15.什么是碎片

碎片（Fragment）是一种可以嵌入到活动当中的UI片段，它和活动一样，同样能包含布局，同样都有自己的生命周期。（建议使用support-v4库中的Fragment）

1. 碎片的简单用法

新建碎片布局.xml，在代码中extends Fragment，重写onCreateView（）方法，将刚才定义的布局动态加载进来。



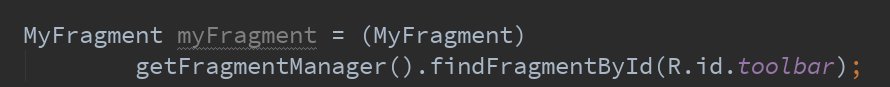
接下来在修改活动的布局.xml中的代码，使用<fragment>标签在布局中添加碎片，其中通过android:name属性来显式指明要添加的碎片类名，注意一定要将类的包名也加上。

1. 动态添加碎片

动态添加碎片主要分为5步：

1. 创建带添加碎片的实例
2. 获取FragmentManager，在活动中可以直接通过调用getSupportFragmentManager（）方法得到
3. 开启一个事务，通过调用beginTransaction（）方法开启
4. 向容器内添加或替换碎片，一般使用replace（）方法实现，需要传入容器的id和待添加的碎片实例
5. 提交事务，调用commit（）方法来完成
6. 如果想模拟类似返回栈的小郭，按下back键可以回到上一个碎片，使用addToBackStack（）方法即可
7. 碎片和活动之间进行通信

为方便碎片和活动之间进行通信，FragmentManager提供了一个类似于findViewById（）的方法，专门用于从布局文件中获取碎片的实例，代码如下：



每个碎片中都可以通过调用getActivity（）方法来得到和当前碎片相关联的活动实例，代码如下：



1. 碎片的生命周期

（1）碎片的状态：

运行状态：当一个碎片是可见的，并且它所关联的活动正处于运行状态时，该碎片也处于运行状态。

暂停状态：当一个活动进入暂停状态时（由于另一个未占满屏幕的活动被添加到了栈顶），与它相关联的可见碎片就会进入到暂停状态。

停止状态：当一个活动进入停止状态时，与它相关联的碎片就会进入到停止状态。或者通过调用 FragmentTransaction 的 remove()、replace()方法将碎片从活动中移除，但有在事务提交之前调用addToBackStack()方法，这时的碎片也会进入到停止状态。总的来说，进入停止状态的碎片对用户来说是完全不可见的，有可能会被系统回收。

销毁状态：碎片总是依附于活动而存在的，因此当活动被销毁时，与它相关联的碎片就会进入到销毁状态。或者通过调用FragmentTransaction 的 remove()、replace()方法将碎片从活动中移除，但在事务提交之前并没有调用 addToBackStack()方法，这时的碎片也会进入到销毁状态。

（2）碎片的回调方法：

Fragment类提供了一系列的回调方法，以覆盖碎片的每个环节，主要的回调方法有：

onAttach()：当碎片和活动建立关联的时候调用。

onCreateView()：为碎片创建视图（加载布局）时调用。

onActivityCreated()：确保与碎片相关联的活动一定已经创建完毕的时候调用。

onDestroyView()：当与碎片关联的视图被移除的时候调用。

onDetach()：当碎片和活动解除关联的时候调用。

（3）碎片完整的生命周期，如下图所示，图片来自android官网：

