**第二章**

1. 隐式Intent，可以通过指定action和category等信息，由系统判断来启动配置了对应action和category的Activity，每个Intent只能指定一个action，但可以指定多个category。<data>标签可以用来更精确地指定当前活动相应什么类型的数据，如配置<data android:scheme=”http” />可以响应打开网页的Intent。
2. Activity的启动模式：

standard：默认启动模式，每次启动都会创建该Activity的一个新实例；

singleTop：启动时如果发现回退栈的栈顶是该Activity，则直接使用，不重新创建，但如果该Activity不在栈顶，则会再创建一个该Activity的新实例；

singleTask：如果回退栈中包含有该Activity的实例，则把该实例上的所有活动统统出栈后使用该实例，没有则创建一个新的实例；

singleInstance：用一个单独的回退栈来管理这个Activity，不管从哪里启动该Activity，都公用这个回退栈。

1. 最佳实践：

在BaseActivity的onCreate中调用getClass.getSimpleName()打印子类的类名，手Q的BaseActivity中在打印日志时也有这么使用。

随时随地退出程序：用一个List来管理所有Activity，每个Activity在onCreate时将该Activity add进该List，onDestory时从该List中remove掉，退出应用时finish掉List中的所有Activity。由于在onDestory中将该Activity从List中remove掉了，因此，不会发生内存泄漏。

启动Activity时尽量统一在目标Activity中加一个static方法，传入context和intent要携带的参数。

**第三章**

1. LinearLayout中的布局，如果指定了layout\_weight属性，对应方向的layout\_width或layout\_height指定为0是一种比较规范的写法，因为此时该属性已经不起作用。
2. 同一个LinearLayout中的一个控件如果在对应的布局方向指定了wrap\_content，而另外的一些控件指定了对应的layout\_weight，则前面的控件按wrap\_content来计算，后面的所有控件占满LinearLayout的剩余空间。
3. 聊天Demo：见代码ChatDemo。

**第四章**

1. Activity通过调用FragmentManager.findFragmentById()来方法来获取对应的Fragment实例，从而调用该实例中的方法。
2. Fragment通过调用getActivity()获取到与当前Fragment关联的Activity，从而调用Activity中的方法。
3. 如果调用了FragmentTransaction的addToBackStack()方法，将一个Fragment事务添加到返回栈中，那么通过物理键back时，会回到上一个Fragment中，而不会直接退出当前程序。

**第五章**

1. 广播分为标准广播和有序广播，标准广播发出后，所有配置了对应action的的广播接收器都可以接收到这条广播，没有具体的先后顺序；而有序广播是有接收顺序的，有序广播的接收顺序根据广播接收器的优先级决定，优先级高的接收器可以阻断该广播，从而后面优先级低的接收器将无法接受到该广播。
2. 广播接收器的注册有两种，动态和静态，其中静态注册跟Activity的注册类似,静态注册广播接收器可实现开机自启动，这两类广播都是全局的。
3. 本地广播发出的广播只能在应用程序内部进行传递和接收，本地广播无法通过静态注册的方式来接收，
4. 强制下线Demo：见代码ForceOfflineDemo。

**第六章**

1. 数据持久化的三种方式：文件存储、SharedPreference、数据库。
2. 文件存储默认在/data/data/<packagename>/files目录下，SharedPreference实际上是将数据保存在一个xml文件中，在data/data/<packagename>/shared\_prefs目录下。
3. 数据库升级时，注意在onUpgrade中switch case中break的使用，为了保证跨版本升级，尽量不要用break。