Activity

- 生命周期全解析
- 四种启动模式
- IntentFilter匹配规则

生命周期

Activity的生命周期?

1. `onCreate()`

> 状态：Activity 正在创建
> 任务：做初始化工作，如setContentView界面资源、初始化数据
> 注意：此方法的传参Bundle为该Activity上次被异常情况销毁时保存的状态信息

2. `onSreat()`

> 状态：Activity 正在启动，这时Activity 可见但不在前台，无法和用户交互

3. `onResume()`

> 状态：Activity 获得焦点，此时Activity 可见且在前台并开始活动

4. `onPause()`

> 状态： Activity 正在停止
> 任务：可做数据存储、停止动画等操作
> 注意：Activity切换时，旧Activity的onPause会先执行，然后才会启动新的Activity

5. `onStop()`

> 状态：Activity 即将停止
> 任务：可做稍微重量级回收工作，如取消网络连接、注销广播接收器等
> 注意：新Activity是透明主题时，旧Activity都不会走onStop

6. `onDestroy()`

> 状态：Activity 即将销毁
> 任务：做回收工作、资源释放

7. `onResart()`

> 状态：Activity 重新启动，Activity由后台切换到前台，由不可见到可见

onStart()和onResume()、onPause()和onStop()的区别: onStart与onStop是从Activity是否可见这个角度调用的，onResume和onPause是从Activity是否显示在前台这个角度来回调的，在实际使用没其他明显区别

Activity生命周期的切换过程?

1.启动一个Activity:

> onCreate()-->onStart()-->onResume()

2.打开一个新Activity:

> 旧Activity的onPause() -->新Activity的onCreate()-->onStart()-->onResume()-->旧Activity的onStop()

3.返回到旧Activity:

> 新Activity的onPause（）-->旧Activity的onRestart()-->onStart()-->onResume()-->新Activity的onStop()-->onDestory();

4.Activity1上弹出对话框Activity2:

> Activity1的onPause()-->Activity2的onCreate()-->onStart()-->onResume()

5.关闭屏幕/按Home键:

> Activity2的onPause()-->onStop()-->Activity1的onStop()

6.点亮屏幕/回到前台:

> Activity2的onRestart()-->onStart()-->Activity1的onRestart()-->onStart()-->Activity2的onResume()

7.关闭对话框Activity2:

> Activity2的onPause()-->Activity1的onResume()-->Activity2的onStop()-->onDestroy()

8.销毁Activity1:

> onPause()-->onStop()-->onDestroy()

生命周期的各阶段?

1.完整生命周期

> Activity在onCreate()和onDestroy()之间所经历的。 在onCreate()中完成各初始化操作，在onDestroy()中释放资源

2.可见生命周期

> Activity在onStart()和onStop()之间所经历的。 活动对于用户是可见的，但仍无法与用户进行交互

3.前台生命周期

> Activity在onResume()和onPause()之间所经历的。 活动可见，且可交互

4.onSaveInstanceState和onRestoreInstanceState

> a.出现时机：异常 情况下Activity 重建，非用户主动去销毁
> b.系统异常终止时，调用onSavaInstanceState来保存状态。该方法调用在onStop之前，但和onPause没有时序关系。

onSaveInstanceState与onPause的区别：前者适用于对临时性状态的保存，而后者适用于对数据的持久化保存

> c.Activity被重新创建时，调用onRestoreInstanceState（该方法在onStart之后），并将onSavaInstanceState保存的Bundle对象作为参数传到onRestoreInstanceState与onCreate方法。

可通过onRestoreInstanceState(Bundle savedInstanceState)和onCreate((Bundle savedInstanceState)来判断Activity是否被重建，并取出数据进行恢复。但需要注意的是，在onCreate取出数据时一定要先判断savedInstanceState是否为空。另外，谷歌更推荐使用onRestoreInstanceState进行数据恢复。

Activity异常情况下生命周期分析？

1.由于资源相关配置发生改变，导致Activity被杀死和重新创建

例如屏幕发生旋转：当竖屏切换到横屏时，会先调用onSaveInstanceState来保存切换时的数据，接着销毁当前的Activity，然后重新创建一个Activity，再调用onRestoreInstanceState恢复数据。

> onSaveInstanceState-->onPause（不定）-->onStop--> onDestroy-->onCreate-->onStart-->onRestoreInstanceState-->onResume

为了避免由于配置改变导致Activity重建，可在AndroidManifest.xml中对应的Activity中设置android:configChanges="orientationIscreenSize"。此时再次旋转屏幕时，该Activity不会被系统杀死和重建，只会调用onConfigurationChanged。因此，当配置程序需要响应配置改变，指定configChanges属性，重写onConfigurationChanged方法即可。

2.由于系统资源不足，导致优先级低的Activity被回收

> a.Activity优先级排序

前台可见Activity>前台可见不可交互Activity（前台Activity弹出Dialog)>后台Activity（用户按下Home键、切换到其他应用）

> b.当系统内存不足时，会按照Activity优先级从低到高去杀死目标Activity所在的进程
> c.若一个进程没有四大组件在执行，那么这个进程将很快被系统杀死

Activity四种启动模式

设置Activity启动模式的方法?

> a.在AndroidManifest.xml中给对应的Activity设定属性
> android:launchMode="standard|singleInstance|single Task|singleTop"
> b.通过标记位设定，方法是intent.addFlags(Intent.xxx)。

Activity的四种LaunchMode?

1. `standard` ： 标准模式、默认模式

> 含义：每次启动一个Activity就会创建一个新的实例。 注意：使用ApplicationContext去启动standard模式Activity就会报错。因为standard模式的Activity会默认进入启动它所属的任务栈，但是由于非Activity的Context没有所谓的任务栈。

2. `singleTop` ： 栈顶复用模式

> 含义：如果新Activity已经位于任务栈的栈顶，就不会重新创建，并回调**onNewIntent(intent)**方法。

3. `singleTask` ： 栈内复用模式

> 含义：只要该Activity在一个任务栈中存在，都不会重新创建，并回调onNewIntent(intent)方法。如果不存在，系统会先寻找是否存在需要的栈，如果不存在该栈，就创建一个任务栈，并把该Activity放进去；如果存在，就会创建到已经存在的栈中。

4. `singleInstance` ： 单实例模式

> 含义： 具有此模式的Activity只能单独位于一个任务栈中，且此任务栈中只有唯一一个实例

常用的可设定Activity启动模式的标记位?

> - FLAG_ACTIVITY_SINGLE_TOP:对应singleTop启动模式
> - FLAG_ACTIVITY_NEW_TASK： 对应singleTask模式

IntentFilter匹配规则

原则:

a.一个intent只有同时匹配某个Activity的intent-filter中的action、category、data才算完全匹配，才能启动该Activity。
b.一个Activity可以有多个 intent-filter，一个 intent只要成功匹配任意一组 intent-filter，就可以启动该Activity。

1.action匹配规则:

> - 要求intent中的action 存在且必须和intent-filter中的其中一个 action相同
> - 区分大小写

2.category匹配规则：

> - intent中的category可以不存在，如果此时系统给该Activity 默认加上了< category android:name="android.intent.category.DEAFAULT" />属性值
> - 除上述情况外，有其他category，则要求intent中的category和intent-filter中的所有category相同

3.data匹配规则：

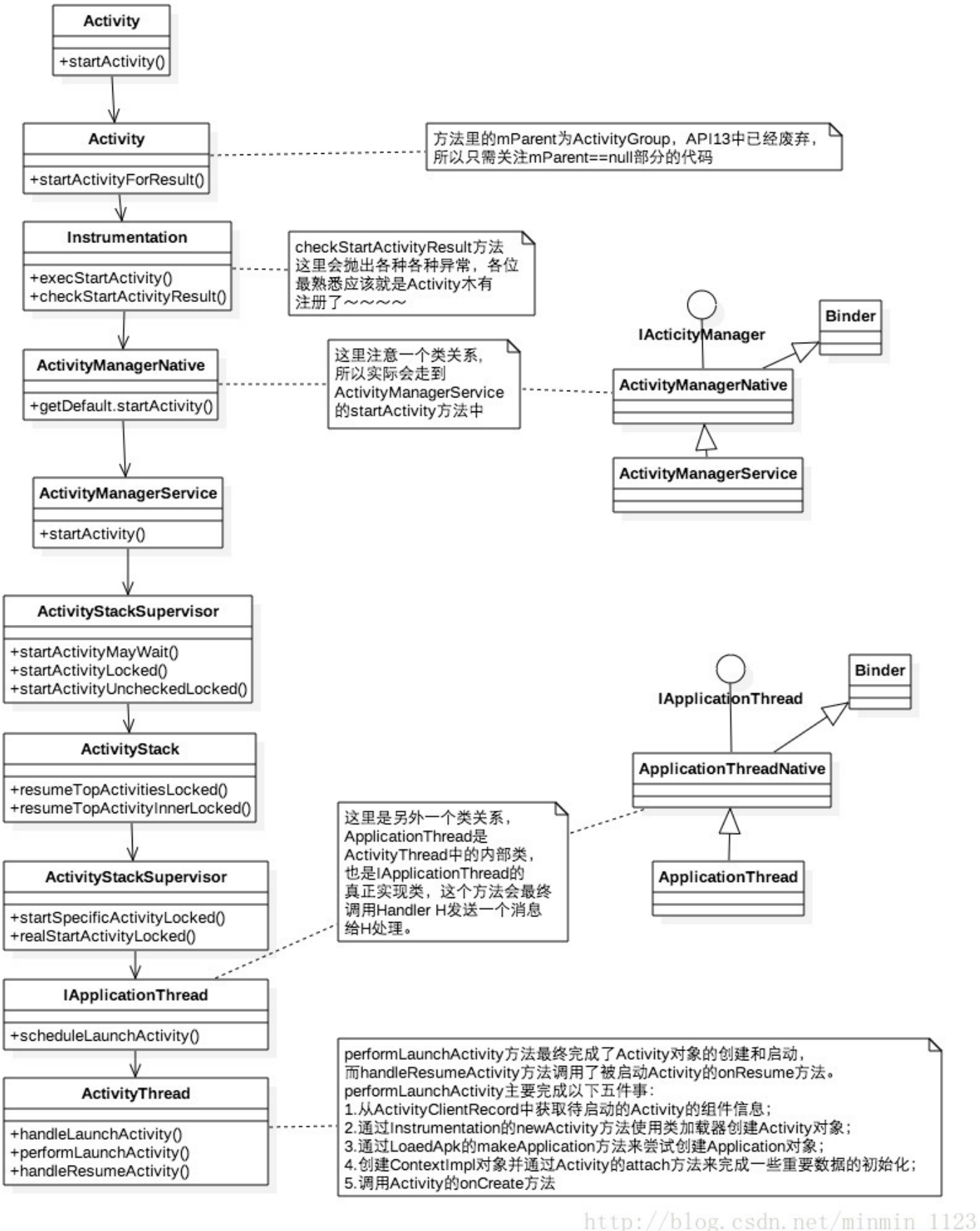
> - 如果intent-filter中有定义data，那么Intent中也必须也要定义date
> - data主要由mimeType(媒体类型)和URI组成。在匹配时通过intent.setDataAndType(Uri data, String type)方法对date进行设置

采用隐式方式启动Activity时，可以用PackageManager的resolveActivity方法或者Intent的resolveActivity方法判断是否有Activity匹配该隐式Intent。

Activity的启动过程

Activity的启动过程?

> **技术点：**Activity启动、ActivityManagerServie、ApplicationThread 思路：可大致介绍Activity启动过程涉及到的类，尤其是ActivityManagerServie、ApplicationThread从中发挥的作用。
> **参考回答：**调用startActivity()后经过重重方法会转移到ActivityManagerService的startActivity()，并通过一个IPC回到ActivityThread的内部类ApplicationThread中，并调用其scheduleLaunchActivity()将启动Activity的消息发送并交由Handler H处理。Handler H对消息的处理会调用handleLaunchActivity()->performLaunchActivity()得以完成Activity对象的创建和启动。
> **引申：**由于ActivityManagerService是一个Binder对象，可引申谈谈Binder机制



ActivityManagerService、ApplicationThread都是Binder。

- Application的创建也是通过Instrumentation来完成的，这个过程和Activity对象一样，都是通过类加载器来实现的。

- Activity的启动过程最终回到ApplicationThread中，通过ApplicationThread.scheduleLaunchActivity() 将启动Activity的消息发送并交由Handler H处理。
- Handler H对消息的处理会调用handleLaunchActivity()->performLaunchActivity()得以最终完成Activity的创建和启动。

源码分析：Activity的工作过程