四大组件包括：Activity、Service、BroadcastReceiver、ContentProvider

# Intent:

## 1.1 显示Intent&隐式Intent

显式Intent：通过提供目标应用的软件包名称或完全限定的组件类名来指定可处理 Intent 的应用。（显示Intent会始终会传递给其目标，无论组件声明的 Intent 过滤器如何均是如此。）

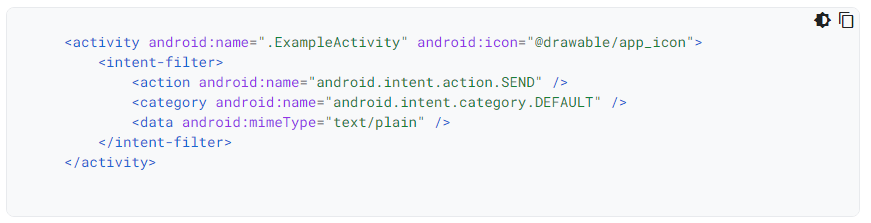
隐式Intent: 不会指定特定的组件，而是声明要执行的常规操作，从而允许其他应用中的组件来处理。(没有指定组件名称，即为隐式)

接收隐式Intent：在应用清单文件中使用 <intent-filter> 元素为每个应用组件声明一个或多个 Intent 过滤器。

## 1.2 Intent Filter

Intent Filter: 应用清单文件中的一个表达式，用于指定该组件要接收的 Intent 类型。通过为 Activity 声明 Intent 过滤器，您可以使其他应用能够直接使用某一特定类型的 Intent 启动 Activity。通过为 Activity 声明 Intent 过滤器，您可以使其他应用能够直接使用某一特定类型的 Intent 启动 Activity。

AndroidManifest.xml中可以在Activity中声明<intent-filter>来指定 Activity 能够响应的 intent 类型。



Intent 三要素：action、category、data

<action> 元素指定该 Activity 会发送数据。、

每个<intent-filter>至少拥有一个<action>，否则Activity不响应任何Intent

Action包括：

Intent.ACTION\_VIEW: 向用户显示数据()

Intent.ACTION\_VIEW\_LOCUS: 显示与唯一LocusId关联的Activity。（？）

Intent.ACTION\_VIEW\_PERMISSION\_USAGE: 显示有关给定权限组使用情况的信息。（使用该action之前，需要先在应用中创建处理该action的Activity）

Intent.ACTION\_VIEW\_PERMISSION\_USAGE\_FOR\_PERIOD: 启动 UI 以显示有关给定时间段内给定权限组的使用情况的信息。（使用该action之前，需要先在应用中创建处理该action的Activity）

Intent.ACTION\_VOICE\_COMMAND: 打开语音助手

Intent.ACTION\_WEB\_SEARCH: 网络搜索，如果是url会打开网站，如果是文本会进行google搜索。

Intent.ACTION\_EDIT: 修改资源，例如图片等

<category> Intent 组件类型的附加信息

DEFAULT： 可使 Activity 能够接收启动请求。

CATEGORY\_BROWSABLE：目标 Activity 允许本身通过网络浏览器启动，以显示链接引用的数据，如图像或电子邮件。

<data> 元素指定此 Activity 可以发送的数据类型。

可以通过setType，指定数据类型（其 MIME 类型）。

通过setData，指定数据uri。

通过setDataAndType()，指定数据和类型。

# Activity

### 跳转时的生命周期变化：

Activity A 启动另一个Activity B，回调如下:

Activity A 的onPause() → Activity B的onCreate() → onStart() → onResume() → Activity A的onStop()；

如果B是透明主题又或则是个DialogActivity，则不会回调A的onStop；

### 参数传递

方法一：使用intent进行参数传递（适合简单数据、序列化对象）

其中序列化对象分为 Serializable、 Parcelable。 前者是java提供的，后者是andorid提供的。

### 参数返回

方法一：使用startActivityForResult启动activity， 后一个activity使用setResult(code, intent);返回结果。前一个activity使用onActivityResult接收结果。 （显示被弃用，使用 registerForActivityResult代替。）

方法二：registerForActivityResult，在Activity super.onCreate()前注册跳转launch，其中需要声明输入输出、以及收到返回结果后的回调。

### 启动模式

Standard：默认模式，会创建新的实例

SingleTask：单任务模式，一个栈中如果已存在该实例，则将其上所有的Activity出栈，并显示该Activity

SingleTop：如果当前栈顶是该元素，则直接使用该实例显示。

SingleInstance：创建一个新的任务栈，并在该栈中创建目标实例。

**Intent Flag:**

FLAG\_ACTIVITY\_CLEAR\_TOP: 销毁栈中目标及以上的activity，并重新创建activity

FLAG\_ACTIVITY\_SINGLE\_TOP: 这个FLAG就相当于Activity启动模式中的singleTop

FLAG\_ACTIVITY\_CLEAR\_TASK | FLAG\_ACTIVITY\_NEW\_TASK: 清楚当前栈中所有activity，并且启动一个新栈，在栈中创建新的activity实例。

FLAG\_ACTIVITY\_NEW\_TASK：在新栈中创建目标activity实例。

# Service

### 开启服务的方式

方法一：startService，独立于Activity运行，自行结束或被叫停。通过stopService、stopSelf结束服务

方法二：bindService，绑定于Activity运行，Activity结束时，会被叫停。通过unbindService解绑服务。

注意：只有两种模式都不再使用Service时，Service才可以销毁。



### 前台服务

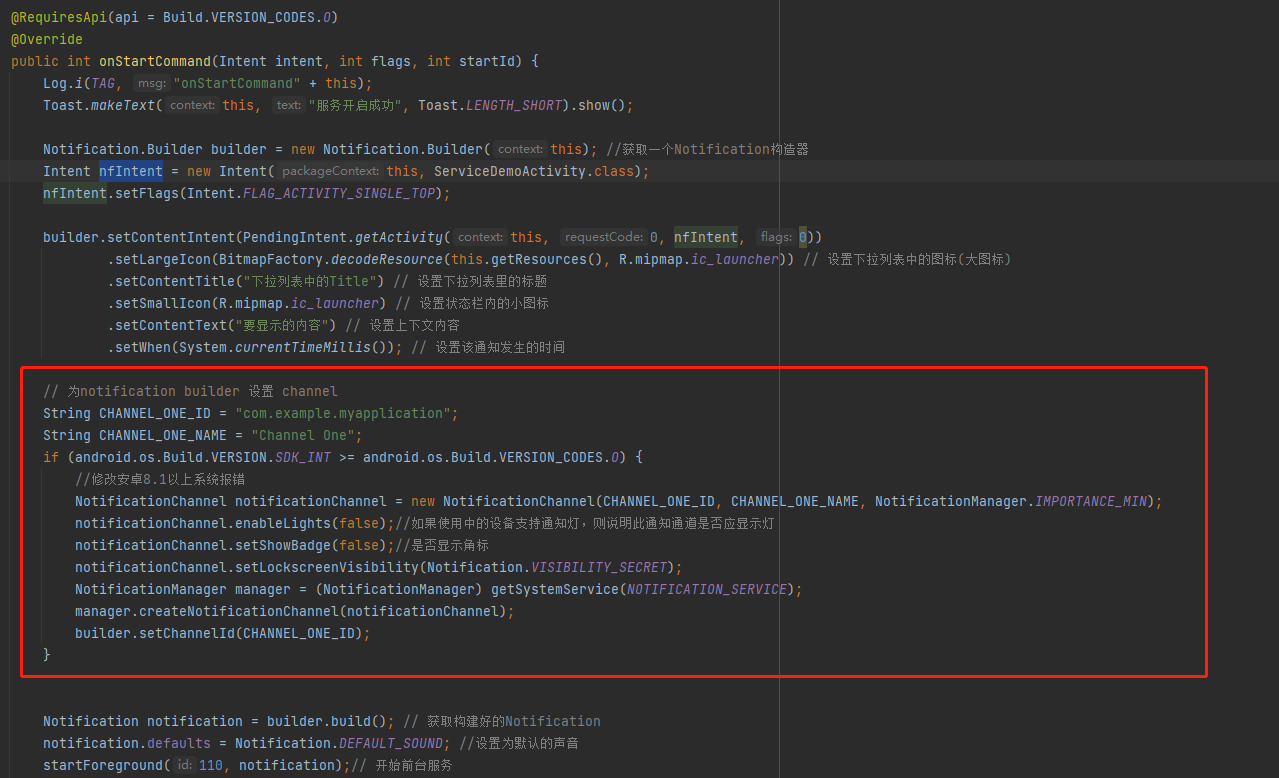
为了避免服务被后台杀死，可以把服务调整到前台。下图是创建前台服务的方法（与通知相结合）

首先需要以下权限，权限是用户权限，只需在注册文件中声明即可。

<uses-permission android:name="android.permission.FOREGROUND\_SERVICE"/>



在Android8以后需要为Notification.Builder 添加一个Channel。



### IntentService

IntentService是一个简化的Service，必须实现onHandleIntent方法。

对同一个IntentService提交的Intent，会在同一线程串行执行。

**IntentService 类会执行以下操作：**

1. 创建默认的工作线程，用于在应用的主线程外执行传递给 onStartCommand() 的所有 Intent。
2. 创建工作队列，用于将 Intent 逐一传递给 onHandleIntent() 实现，这样您就永远不必担心多线程问题。
3. 在处理完所有启动请求后停止服务，因此您永远不必调用 stopSelf()。
4. 提供 onBind() 的默认实现（返回 null）。
5. 提供 onStartCommand() 的默认实现，可将 Intent 依次发送到工作队列和 onHandleIntent() 实现。

### 自启动

### ANR机制

在ANR机制中，Service的响应时长不能超过20s

# BroadcastReceiver

## 4.1 注意事项

广播可作为跨应用和普通用户流之外的消息传递系统，但是如果滥用会降低系统性能。

## 4.2 广播接收器

需要实现onReceive方法，该方法用作接收到广播后的处理逻辑。注意，该方法是执行在主线程的。

### （1）静态广播接收器

需要在清单文件中注册的广播接收器。作用是静态注册的接收器会常驻在系统中（不管应用是否在运行）

### 动态广播接收器

在代码中注册的广播接收器，一般在onResume中注册，在onPause中卸载

## 4.3 广播发送

### 普通广播

普通广播在逻辑上是异步发给所有的接收者的。实际上是一个一个的发，只是顺序随即而已额。

1. 通过sendBroadcast发送。
2. Intent通过setAction设置广播标识，该标识需要全局唯一。（可以任意，但是推荐是包名+广播名）。
3. 可以通过putExtra添加附加信息。



### （2）有序广播

广播串行的在各个接收器中传递，每个接收器可以继续向下传递，也可以中止传递。

接收器接收的顺序由intent-filter中的优先级决定。

1. 发送有序广播



receiverPermission的作用是指定接收方必须拥有的权限字符串，一般可以赋值为null。

1. 创建自定义接收器。
2. 设置接收器优先级

使用Intent-Filter设置优先级。

1. 注册接收器。

registerReceiver(receiver, orderFilter) 注册

### ANR

BroadcastReceiver的ANR时间是10s

### 本地广播

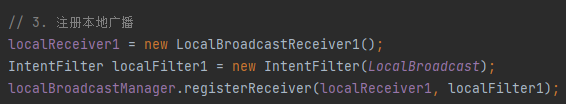
**优点：**

1. 发送的广播只会在自己的App内传播，不会泄漏给其他App，保障了数据的安全性。
2. 无法接受到其他App的广播，也就省去各种麻烦事。
3. 相较于全局广播效率更高。

**使用步骤：**

1. 获取LocalBroadcastManager
2. 通过localBroadcastManager.sendBroadcast发送本地广播
3. 通过localBroadcastManager.registerReceiver注册本地接收器
4. 通过localBroadcastManager.unregisterReceiver(); 卸载本地接收器。





### 清单声明广播接收器

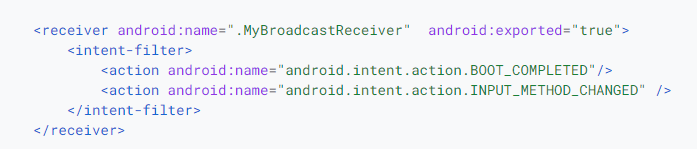
系统软件包管理器会**在应用安装时注册接收器**。然后，该接收器会成为应用的一个独立入口点，这意味着**如果应用当前未运行，系统可以启动应用并发送广播**。

**注意：**在Android8.0， sdk26 以后不能使用清单声明广播接收器。但是有的隐式广播可以使用清单声明广播接收器。

（https://developer.android.com/guide/components/broadcast-exceptions）

**使用方法：**

1. 在清单文件中声明<receiver>, 使用<intent-filter><action>来说明该接收器接收的广播。



1. 创建对应的Receiver自定义子类。

# ContentProvider

作用：让其他应用访问本应用数据。是内容共享的统一接口。

### ContentResolve

作用：获取其他应用的数据。通过Context.getContentResolve() 来获取

**Insert**： android.net.Uri insert(android.net.Uri url, android.content.ContentValues values)

**Query**： 具体参数如下图所示。函数返回值是Cursor，可以使用列名从中获取具体数据。



**Update：**更新记录，需要传入contentvalue （更新字段信息）；where,args（记录筛选条件）。

**Delete：**删除记录，需要传入where, args 来筛选删除条件。

### ContentProvider

ContentProvider具体数据的操作需要使用SQLite、Room等数据库工具完成数据的实际操作。

大多数情况，ContentProvider是单例的。

**系统预设ContentProvider:**

通讯录 com.android.contacts

媒体 media

日历 com.android.calendar

用户词典 user\_dictionary

修改日历需要权限 （需要动态申请）：

android.permission.READ\_CALENDAR

android.permission.WRITE\_CALENDAR

**UriMatcher**：主要作用是根据 Uri 匹配对应的数据表。

**Uri：**

authority 用作区分不同的程序

例如：com.example.app 程序的authority=com.example.app.provider

path 用作对同一程序的不同表作区分，如/table1、/table2

总的来说，uri分两种情况：table结尾的和id结尾的。可以使用uriMatcher来区分这两种情况

Com.example.app.provider/table1

Com.example.app.provider/table1/# (#是通配符，表示任意长度的数字)

**增删改查：**

在通过uri、以及Provider提供的insert、update、delete、query方法参数，我们可以使用SQLite或者Room来对数据进行实际的操作。

### MIME 类型

作用：用于指定某种扩展名的文件与应用程序的对应关系。一个 MIME 类型分为「主类型」+「子类型」。

主类型：

|  |  |  |
| --- | --- | --- |
|  | 主类型 | 子类型 |
| 声音 | Audio |  |
| 视频 | video |  |
| 图像 | image | Png(.png)  Jepg(.jepg) |
| 文本 | text | Html(.html) plain(.txt) |

# SQLite

### （1）协定类（\*\*\*Contract）

作用：存放Android本地SQLite数据库表的信息，如表名、列名、contentProvider Uri等。

### 自定义SQLiteOpenHelper

作用：提供了SQLite数据库相关的一些有用API。

需要重写onCreate、onUpgrade方法。

### 获取SQLiteDatabase

通过SQLiteOpenHelper.getWritableDatabase或者SQLiteOpenHelper.getReadableDatabase来获取。

SQLiteDatabase, 通过 insert、update、delete、query来对数据库进行操作。

# RecyclerView

### 监听事件

1. OnScrollListener

### （2）

# 参考

1. ContentProvider 筑基篇 <https://juejin.cn/post/6929518871921557512>
2. 安卓view曝光统计实现方案 <https://www.jianshu.com/p/588bc9060935>
3. GridLayoutManager https://juejin.cn/post/6844903930002948110