Intent和Intent过滤器

> 前言
> 看本文之前可以先了解Activity生命周期 和 Activity 状态更改

1.概览

`Intent` 是一个消息传递对象。可以用来从其它应用组件请求操作。
`Intent` 可以通过多种方式进行组件之间的通信，但其基本用处主要包括以下三个：

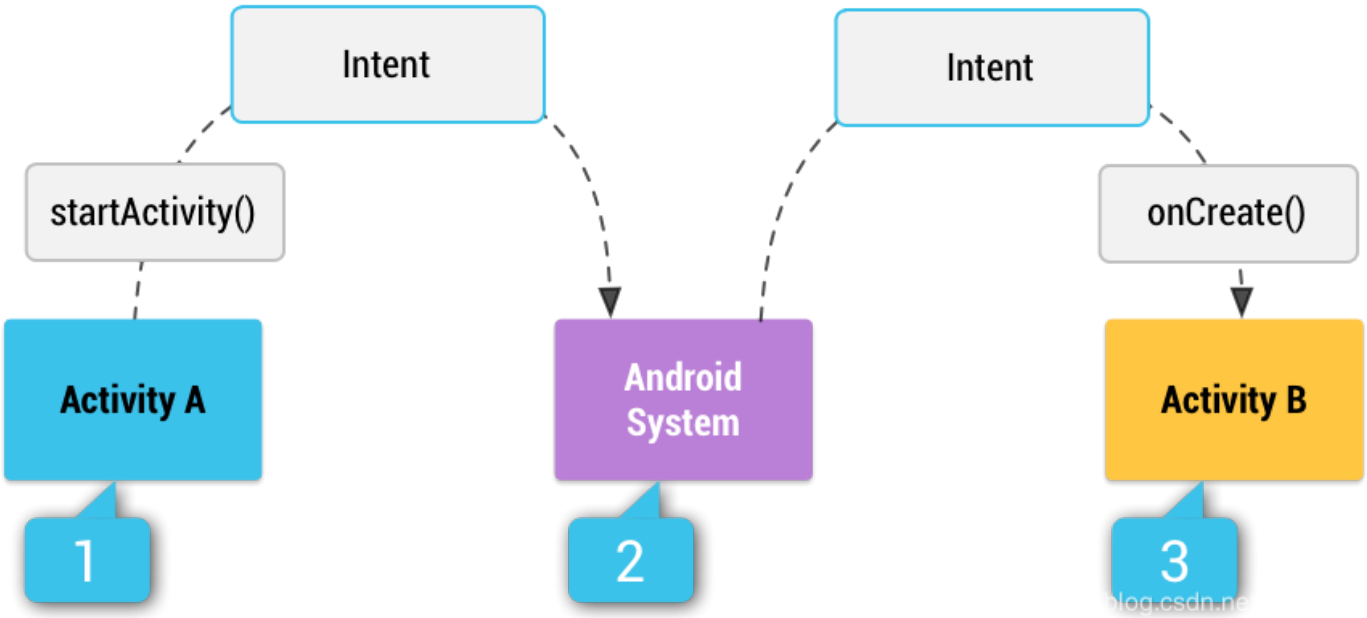
1. **启动 `Activity`**
   将 `Intent` 传递给 `startActivity()` ，可以启动新的 `Activity` 实例。 `Intent` 用于描述要启动的 `Activity` ，并携带任何必要的数据。
   如果希望在 `Activity` 完成后收到结果，可以调用 `startActivityForResult()` ,在 `Activity` 的 `onActivityReuslt()` 回调中接收。
2. **启动 `Service`**
   Service是一个不适用用户界面而在后台执行操作的组件。当API 大于21时，可以启动包含JobScheduler服务。对于API 21之前的版本，可以使用Service类的方法来启动Service。通过Intent传递给startService()。Intent用于描述要启动的Service，并携带任何必要信息。
3. **传递 `Broadcast`**
   `broadcast` 是一个任何应用都可以接收，系统会针对系统事件（例如：系统启动或设备开始充电时）传递各种广播。通过将 `Intent` 传递给 `sendBroadcast()` 或 `sendOrderedBroadcast()` ，可以将广播传递出。

2. `Intent` 类型

`Intent` 有两种类型：

- **显式 `Intent`** ：通过提供目标应用的软件包名或完全限定的组件类名来启动组件。通常，我们在自己的应用中使用显式 `Intent` 来启动组件（我们知道要启动的 `Activity` 或 `Service` 的类名）。
- **隐式 `Intent`** ：不会指定特定的组件，而是声明要执行的常规操作，从而允许其他中的组件来处理。例如，如需在地图上想用户显示位置，则可以使用隐式 `Intent` ,请求另一具有此功能的应用在地图上显示指定的位置。

显示如何在启动 Activity 时使用 Intent



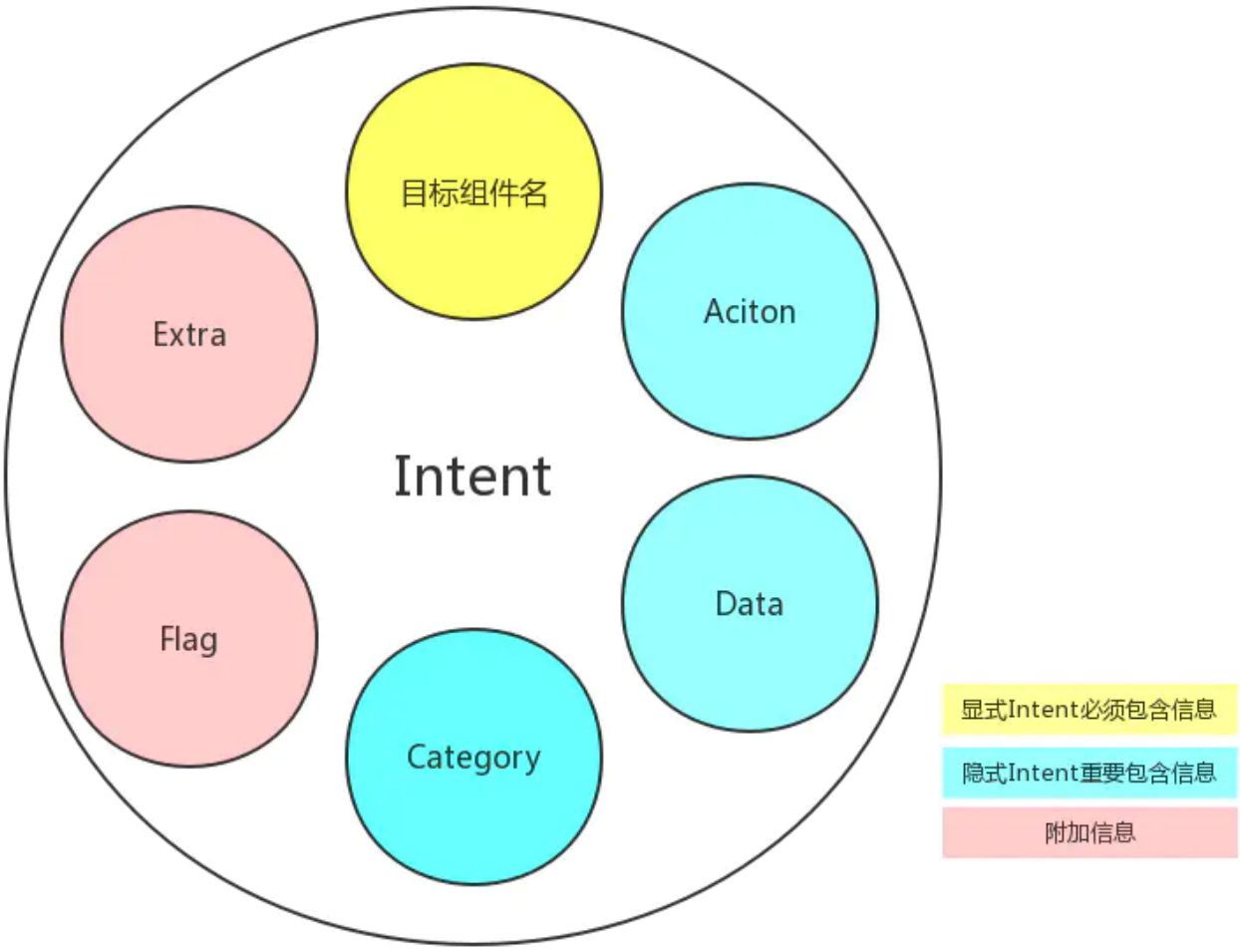
隐式 `Intent` 如何通过系统传递以启动其他 `Activity` ：

1. `Activity A` 创建包含操作描述的 `Intent` ，并将其传递给 `startActivity()` 。
2. `Android` 系统搜索所有应用中与 `Intent` 匹配的 `Intent` 过滤器。找到匹配项之后。
3. 该系统通过调用匹配 `Activity` （ `Activity B` ）的 `onCreate()` 方法并将其传递给 `Intent` ，以此启动匹配 `Activity` 。
   使用隐式 `Intent` 时， `Android` 系统通过将 `Intent` 的内容与在设上其他应用的清单文件中声明的 `intent-filters` 进行比较，从而找到要启动的相应组件。如果 `Intent` 与 `intent-filter` 兼容，则系统会显示一个对话框，支持用户选取要使用的应用。

> **注意**: 为了确保应用的安全性，**启动 `Service` 时，应该始终使用显式 `Intent`** ，且不要为 `Service` 声明 `intent-filters` 。使用隐式 `Intent` 启动服务存在安全隐患，因为无法确定哪些 `Service` 将响应 `Intent` ，且用户无法看到哪些服务已启动。从Android 5.0（API 21）开始，如果使用隐式 `Intent` 调用 `bindservice()` ，系统会抛出异常。

3. 构建Intent

Intent对象携带Android系统用来确定要启动哪个组件的信息（例如，准确的组件名称或应当接收该intent的组件类别），以及组件为了正确执行操作而使用的信息（例如，要采取得操作以及要处理的数据）。



- **组件名称（`ComponentName`）**
  要启动的组件名称。
  构建显式`Intent`的一项重要信息，这意味着`Intent`应当仅传递给由组件名称定义的应用组件。如果没有组件名称，则`Intent`为隐式，且系统将根据其他`Intent`信息（例如，`action`、`data`、`cateory`），如需在应用中启动特定的组件，则应指定该组件的名称。

```
public final class ComponentName implements Parcelable, Cloneable, Comparable<Comp
 private final String mPackage;
 private final String mClass;
 }
```

  可以使用`setComponent()`、`setClass()`、`setClassName()`，或`Intent`构造函数设置组件名称。

- 操作(`Action`)
  指`Intent`发向组件的主要动作，比如：图片应用中主要动作为查看图片的组件、地图应用中主要动作为查看地址的组件。另外，对于广播(`Broadcast`)组件而言，`Intent`的`action`则是指广播具体的值。当`Broadcast Receiver`接收到该值时代表了某事件已经发生。
  通常使用的主要是Android系统内置`action`，这些`action`实际上保存在`Intent`类中的静态常

量，系统的默认组件(如:默认浏览器、图片浏览器、拨号页面等)都可以响应的`action`。下面介绍常用的内置`action`：

- `ACTION_VIEW`
  向用户展示某些信息，比如使用浏览器打开网址，用图片应用显示图片等。
- `ACTION_SEND`
  用于发送数据，比如电子邮件应用或者一些社交应用。
- `ACTION_DIAL`
  显示带拨号盘的页面，让用户可以进行拨号动作。
  除了Android内置action之外，也可以自定义action,供`Intent`在自己的应用内使用（或者共其他应用在自己的应用中调用组件）。如果定义自己的操作，应该确保将应用的软件包名称作为前缀。我们可以使用`setAction()`或`Intent`构造函数为`Intent`指定操作。例如：

```
static final String ACTION_TIMETRAVEL = "com.example.action.TIMETRAVEL";
```

- 数据(`Data`)
  包含了URI对象和MIME 类型。提供的数据类型通常由Intent的action决定。例如，如果action是`ACTION_EDIT`,则数据应包含待编辑文档的URI。
  创建Intent时，除了指定URI外，指定MIME类型往往也是很重要。例如，能够显示图像的`Activity`可能无法播放音频文件。指定数据的MIME类型有助于Android系统找到接收`Intent`的最佳组件。 要仅设置数据URI,调用`setData()`。要仅设置MIME类型，调用`setType()`。如果两者都需要设置则调用`setDataAndType()`（**不可同时调用`setData()`、`setType()`达到此目的**）。

- 类别(`Category`)
  一个包含应处理Intent组件类型的附加信息的字符串，一个Intent可以添加多个Category。可以通过addCategory()指定类别。以下比较常见的Category:

  - `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，如图像或邮件。
  - `CATEGORY_LAUNCHER`
    该Activity是任务的初始化Activity,在系统的应用启动器中列出。

  以上列出的这些属性（`ComponentName`、`Action`、`Data`、`Category`）表示`Intent`的既定特征。通过读取这些属性，Android系统能够解析应当启动哪个组件。但是，`Intent`也有可能会携带一些不影响解析为组件的信息。`Intent`还可以提供一下信息：

- `Extra`
  携带完成请求操作所需的附加信息的键值对。正如某些操作使用特定类型的数据URI一样，有些操作也使用特定的extra。
  可以使用`putExtra()`方法添加extra数据，每种方法接受两个参数：`key`和`value`。还可以创建一个包含所有extra数据的Bundle对象，然后使用`putExtras()`将Bundle插入Intent中。

- 标志(`Flag`)
  标志在Intent类中定义。`Flag`可以指示Android系统如何启动Activity,以及启动之后如何处理。

4. 示例讲解

- 显示Intent示例
  是指用于启动某个特定应用组件（例如，应用中的某个特定Activity或服务）的Intent。要创建显示Intent,就要为Intent对象定义组件名称。
  例如,如果在应用中构建一个名为 `DownloadService` 从网页下载文件的服务，代码如下：

```
// Executed in an Activity, so 'this' is the Context
// The fileUrl is a string URL, such as "http://www.example.com/image.png"
Intent downloadIntent = new Intent(this, DownloadService.class);
downloadIntent.setData(Uri.parse(fileUrl));
startService(downloadIntent);
```

  `Intent(Context , Class)` 构造函数分别为应用分别为应用和组件提供 `Context` 和 `Class` 对象。因此，此Intent将显示启动该应用中的 `DownloadService` 类。

- 隐式Intent示例
  是指定能够在可以执行响应操作的设备上调用任何应用的操作。如果自己应用无法执行该操作而其他应用可以，并且希望用户选取要使用的应用，则使用隐式Intent。
  例如，如果希望用户与他人分享内容，可以使用 `ACTION_SEND` 操作创建Intent,并添加指定共享内容的extra。使用该Intent调用 `startActivity()` 时，用户可以选取共享内容所使用的应用。
  注意：用户可能没有任何应用处理发送到 `startActivity()` 的隐式Intent。或者，由于配置文件限制或管理员执行的设置，可能无法访问应用。如果发生这样的情况，调用失败，应用也会崩溃。要验证是否可跳转。如下：

```
 // Create the text message with a string
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");

// Verify that the intent will resolve to an activity
if (sendIntent.resolveActivity(getPackageManager()) != null) {
    startActivity(sendIntent);
}
```

- 强制使用应用选择器
  如果有多个应用响应隐式Intent,则用户可以选择要使用的应用，并将其设置为该操作的默认选项。如果用户可能希望每次使用相同的应用执行某项操作（例如，打开网页时，用户往往倾向于仅使用一种网络浏览器），则选择默认选项的功能十分有用。
  但是，如果多个应用可以响应Intent,且用户可能希望每次使用不同的应用，则应采用显示方式显示选择器对话框。选择器对话框会要求用户选择用于操作的应用（用户无法为操作选择默认应用）。要显示选择器，可以使用 `createChooser()` 创建Intent,并将其传递给 `startActivity()` ，如下例：

```
Intent sendIntent = new Intent(Intent.ACTION_SEND);
...

// Always use string resources for UI text.
// This says something like "Share this photo with"
String title = getResources().getString(R.string.chooser_title);
// Create intent to show the chooser dialog
Intent chooser = Intent.createChooser(sendIntent, title);

// Verify the original intent will resolve to at least one activity
if (sendIntent.resolveActivity(getPackageManager()) != null) {
    startActivity(chooser);
}
```

- 接收隐式Intent
  要公布应用可以接收哪些隐式Intent，可以在清单文件中使用`<intent-filter>`元素为每个应用组件声明一个或多个Intent过滤器。每个Intent过滤器均根据Intent的`Action`、`Data`和`Category`指定自身接受的Intent类型。仅当隐式Intent可以通过Intent过滤器匹配时，系统才会将该Intent传递给应用组件。
  应用组件应当为自身可执行的每个独特功能声明单独的过滤器。例如，图像库应用中的一个Activity可能会有两个过滤器，分别用于查看图像和编辑图像。当Activity启动时，将检查Intent并根据Intent中的信息决定具体的行为（例如，是否显示编辑器控件）。
  每个Intent过滤器均由应用清单文件中的元素定义，并嵌套在相应的应用组件（例如，元素）中。在内部，可以使用以下三个元素中的一个或多个指定要接收的Intent类型：
  - `<action>`：在name属性中，声明接受的Intent操作。该值必须是操作的文本字符串值，而不是常量值。
  - `<data>`：使用一个或多个指定数据URL(scheme、host、port、path)各个方面和MIME类型的属性，声明接受的数据类型。
  - `<category>`：在name属性中，声明接受的Intent类别。该值必须是操作的文本字符串值，而不是常量值。

  注意：要接收隐式Intent,必须将`CATEGORY_DEFAULT`类别包含在Intent过滤中。方法`startActivity()`和`startActivityForResult()`将按照其声明`CATEGORY_DEFAULT`类别的方式处理所有Intent。如果未在Intent过滤器中声明此类别，则隐式Intent不会解析该Actvity。
  例如，下例是一个使用包含Intent过滤器的Activity声明，当数据类型为文本时，系统将接收`ACTION_SEND` Intent：

```
 <activity android:name="ShareActivity">
    <intent-filter>
       <action android:name="android.intent.action.SEND"/>
       <category android:name="android.intent.category.DEFAULT"/>
       <data android:mimeType="text/plain"/>
    </intent-filter>
</activity
```

下例详细讲解：

```
 <activity android:name="MainActivity">
    <!-- This activity is the main entry, should appear in app launcher -->
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

<activity android:name="ShareActivity">
    <!-- This activity handles "SEND" actions with text data -->
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
    </intent-filter>
    <!-- This activity also handles "SEND" and "SEND_MULTIPLE" with media   data --
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <action android:name="android.intent.action.SEND_MULTIPLE"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="application/vnd.google.panorama360+jpg"/>
        <data android:mimeType="image/*"/>
        <data android:mimeType="video/*"/>
    </intent-filter>
</activity>
```

第一个Activity `MainActivity` 是应用的主要入口。当用户最初使用启动器图标启动应用时，该Activity将打开：

- `ACTION_MAIN` 操作指示这是主要入口点，且不要求输入任何Intent数据。
- `CATEGORY_LAUNCHER` 类别指示此Acitivity的图标应放入系统的应用启动器。如果 `<activity>` 元素未使用icon指定图标，则系统将使用 `<application>` 元素中的图标。

  这两个元素必须配对使用，Activity才会显示在应用启动器中。

第二个Acitivity `ShareActivity` 旨在便于共享文本和媒体内容。尽管用户可以通过从 `MainActivity` 导航进入此Activity，但也可以从发出隐式Intent（与两个Intent过滤器之一匹配）的另一应用中直接进入 `ShareActivity`。