安卓应用基础知识

Android 应用采用 Java 编程语言编写。Android SDK 工具将您的代码连同所有数据和资源文件编译到一个 Android 软件包中，即带有 .apk 后缀的存档文件中。一个 APK 文件包含 Android 应用的所有内容，它是基于 Android 系统的设备用来安装应用的文件。

安装到设备后，每个 Android 应用都运行在自己的安全沙箱内：

- Android 操作系统是一种多用户 Linux 系统，其中的每个应用都是一个不同的用户；
- 默认情况下，系统会为每个应用分配一个唯一的 Linux 用户 ID（该 ID 仅由系统使用，应用并不知晓）。系统为应用中的所有文件设置权限，使得只有分配给该应用的用户 ID 才能访问这些文件；
- 每个进程都具有自己的虚拟机 (VM)，因此应用代码是在与其他应用隔离的环境中运行；
- 默认情况下，每个应用都在其自己的 Linux 进程内运行。Android 会在需要执行任何应用组件时启动该进程，然后在不再需要该进程或系统必须为其他应用恢复内存时关闭该进程。

Android 系统可以通过这种方式实现最小权限原则。也就是说，默认情况下，每个应用都只能访问执行其工作所需的组件，而不能访问其他组件。 这样便营造出一个非常安全的环境，在这个环境中，应用无法访问系统中其未获得权限的部分。不过，应用仍然可以通过一些途径与其他应用共享数据以及访问系统服务：

- 可以安排两个应用共享同一 Linux 用户 ID，在这种情况下，它们能够相互访问彼此的文件。为了节省系统资源，可以安排具有相同用户 ID 的应用在同一 Linux 进程中运行，并共享同一 VM（应用还必须使用相同的证书签署）。
- 应用可以请求访问设备数据（如用户的联系人、短信、可装载存储装置 [SD 卡]、相机、蓝牙等）的权限。 用户必须明确授予这些权限。 如需了解详细信息，请参阅 使用系统权限。

以上内容阐述了有关 Android 应用在系统内存在方式的基础知识。本文的其余部分将向您介绍以下内容：

- 用于定义应用的核心框架组件
- 您用来声明组件和应用必需设备功能的清单文件
- 与应用代码分离并允许您的应用针对各种设备配置适当优化其行为的资源

应用组件

应用组件是 Android 应用的基本构建基块。每个组件都是一个不同的点，系统可以通过它进入您的应用。 并非所有组件都是用户的实际入口点，有些组件相互依赖，但每个组件都以独立实体形式存在，并发挥特定作用 — 每个组件都是唯一的构建基块，有助于定义应用的总体行为。共有四种不同的应用组件类型。每种类型都服务于不同的目的，并且具有定义组件的创建和销毁方式的不同生命周期。以下便是这四种应用组件类型：

Activity

Activity 表示具有用户界面的单一屏幕。例如，电子邮件应用可能具有一个显示新电子邮件列表的 Activity、一个用于撰写电子邮件的 Activity 以及一个用于阅读电子邮件的 Activity。 尽管这些 Activity 通过协作在电子邮件应用中形成了一种紧密结合的用户体验，但每一个 Activity 都独立于其他 Activity 而存在。 因此，其他应用可以启动其中任何一个 Activity（如果电子邮件应用允许）。 例如，相机应用可以启动电子邮件应用内用于撰写新电子邮件的 Activity，以便用户共享图片。

服务

服务是一种在后台运行的组件，用于执行长时间运行的操作或为远程进程执行作业。 服务不提供用户界面。 例如，当用户位于其他应用中时，服务可能在后台播放音乐或者通过网络获取数据，但不会阻断用户与 Activity 的交互。 诸如 Activity 等其他组件可以启动服务，让其运行或与其绑定以便与其进行交互。

服务作为 Service 的子类实现，您可以在服务开发者指南中了解有关它的更多详情。

内容提供程序

内容提供程序管理一组共享的应用数据。您可以将数据存储在文件系统、SQLite 数据库、网络上或您的应用可以访问的任何其他永久性存储位置。 其他应用可以通过内容提供程序查询数据，甚至修改数据（如果内容提供程序允许）。 例如，Android 系统可提供管理用户联系人信息的内容提供程序。 因此，任何具有适当权限的应用都可以查询内容提供程序的某一部分（如 ContactsContract.Data），以读取和写入有关特定人员的信息。内容提供程序也适用于读取和写入您的应用不共享的私有数据。 例如，记事本示例应用使用内容提供程序来保存笔记。

内容提供程序作为 ContentProvider 的子类实现，并且必须实现让其他应用能够执行事务的一组标准 API。 如需了解详细信息，请参阅内容提供程序开发者指南。

广播接收器

广播接收器是一种用于响应系统范围广播通知的组件。 许多广播都是由系统发起的 — 例如，通知屏幕已关闭、电池电量不足或已拍摄照片的广播。应用也可以发起广播 — 例如，通知其他应用某些数据已下载至设备，并且可供其使用。 尽管广播接收器不会显示用户界面，但它们可以创建状态栏通知，在发生广播事件时提醒用户。 但广播接收器更常见的用途只是作为通向其他组件的“通道”，设计用于执行极少量的工作。 例如，它可能会基于事件发起一项服务来执行某项工作。

广播接收器作为 BroadcastReceiver 的子类实现，并且每条广播都作为 Intent 对象进行传递。 如需了解详细信息，请参阅 BroadcastReceiver 类。

Android 系统设计的独特之处在于，任何应用都可以启动其他应用的组件。 例如，如果您想让用户使用设备的相机拍摄照片，很可能有另一个应用可以执行该操作，那么您的应用就可以利用该应用，而不是开发一个 Activity 来自行拍摄照片。 您不需要集成甚至链接到该相机应用的代码，而是只需启动拍摄照片的相机应用中的 Activity。完成拍摄时，系统甚至会将照片返回您的应用，以便您使用。对用户而言，就好像相机真正是您应用的组成部分。

当系统启动某个组件时，会启动该应用的进程（如果尚未运行），并实例化该组件所需的类。 例如，如果您的应用启动相机应用中拍摄照片的 Activity，则该 Activity 会在属于相机应用的进程，而不是您的应用的进程中运行。因此，与大多数其他系统上的应用不同，Android 应用并没有单一入口点（例如，没有 main() 函数）。

由于系统在单独的进程中运行每个应用，且其文件权限会限制对其他应用的访问，因此您的应用无法直接启动其他应用中的组件， 但 Android 系统却可以。因此，要想启动其他应用中的组件，您必须向系统传递一则消息，说明您想启动特定组件的 Intent。 系统随后便会为您启动该组件

启动组件

四种组件类型中的三种 — Activity、服务和广播接收器 — 通过名为 Intent 的异步消息进行启动。Intent 会在运行时将各个组件相互绑定（您可以将 Intent 视为从其他组件请求操作的信使），无论组件属于您的应用还是其他应用。

Intent 使用 Intent 对象创建，它定义的消息用于启动特定组件或特定类型的组件 — Intent 可以是显式的，也可以是隐式的。

对于 Activity 和服务， Intent 定义要执行的操作（例如，“查看”或“发送”某个内容），并且可以指定要执行操作的数据的 URI（以及正在启动的组件可能需要了解的信息）。例如，Intent 传达的请求可以是启动一个显示图像或打开网页的 Activity。 在某些情况下，您可以启动

Activity 来接收结果，在这种情况下，Activity 也会在 Intent 中返回结果（例如，您可以发出一个 Intent，让用户选取某位联系人并将其返回给您 — 返回 Intent 包括指向所选联系人的 URI）。

对于广播接收器， Intent 只会定义要广播的通知（例如，指示设备电池电量不足的广播只包括指示“电池电量不足”的已知操作字符串）。

Intent 不会启动另一个组件类型 – 内容提供程序，后者会在成为 ContentResolver 的请求目标时启动。 内容解析程序通过内容提供程序处理所有直接事务，使得通过提供程序执行事务的组件可以无需执行事务，而是改为在 ContentResolver 对象上调用方法。 这会在内容提供程序与请求信息的组件之间留出一个抽象层（以确保安全）。

每种类型的组件有不同的启动方法：

- 您可以通过将 Intent 传递到 startActivity() 或 startActivityForResult()（当您想让 Activity 返回结果时）来启动 Activity（或为其安排新任务）。
- 您可以通过将 Intent 传递到 startService() 来启动服务（或对执行中的服务下达新指令）。或者，您也可以通过将 Intent 传递到 bindService() 来绑定到该服务。
- 您可以通过将 Intent 传递到 sendBroadcast()、sendOrderedBroadcast() 或 sendStickyBroadcast() 等方法来发起广播；
- 您可以通过在 ContentResolver 上调用 query() 来对内容提供程序执行查询。
- 如需了解有关 Intent 用法的详细信息，请参阅 Intent 和 Intent 过滤器文档。 以下文档中还提供了有关启动特定组件的详细信息： Activity、服务、BroadcastReceiver 和内容提供程序。

清单文件

在 Android 系统启动应用组件之前，系统必须通过读取应用的 AndroidManifest.xml 文件（“清单”文件）确认组件存在。 您的应用必须在此文件中声明其所有组件，该文件必须位于应用项目目录的根目录中。

除了声明应用的组件外，清单文件还有许多其他作用，如：

- 确定应用需要的任何用户权限，如互联网访问权限或对用户联系人的读取权限
- 根据应用使用的 API，声明应用所需的最低 API 级别
- 声明应用使用或需要的硬件和软件功能，如相机、蓝牙服务或多点触摸屏幕
- 应用需要链接的 API 库（Android 框架 API 除外），如 Google 地图库

声明组件

清单文件的主要任务是告知系统有关应用组件的信息。例如，清单文件可以像下面这样声明 Activity：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest >
    <application android:icon="@drawable/app_icon.png" … >
        <activity android:name="com.example.project.ExampleActivity"
            android:label="@string/example_label" … >
        </activity>
        …
    </application>
</manifest>
```

在 <application> 元素中，android:icon 属性指向标识应用的图标所对应的资源。在 <activity> 元素中，android:name 属性指定 Activity 子类的完全限定类名，android:label 属性指定用作 Activity 的用户可见标签的字符串。

您必须通过以下方式声明所有应用组件：

- Activity 的 <activity> 元素
- 服务的 <service> 元素
- 广播接收器的 <receiver> 元素
- 内容提供程序的 <provider> 元素

您包括在源代码中，但未在清单文件中声明的 Activity、服务和内容提供程序对系统不可见，因此也永远不会运行。 不过，广播接收器可以在清单文件中声明或在代码中动态创建（如 BroadcastReceiver 对象）并通过调用 registerReceiver() 在系统中注册。

声明组件功能

如上文启动组件中所述，您可以使用 Intent 来启动 Activity、服务和广播接收器。 您可以通过在 Intent 中显式命名目标组件（使用组件类名）来执行此操作。 不过，Intent 的真正强大之处在于隐式 Intent 概念。 隐式 Intent 的作用无非是描述要执行的操作类型（还可选择描述您想执行的操作所针对的数据），让系统能够在设备上找到可执行该操作的组件，并启动该组件。 如果有多个组件可以执行 Intent 所描述的操作，则由用户选择使用哪一个组件。

系统通过将接收到的 Intent 与设备上的其他应用的清单文件中提供的 Intent 过滤器进行比较来确定可以响应 Intent 的组件。

当您在应用的清单文件中声明 Activity 时，可以选择性地加入声明 Activity 功能的 Intent 过滤器，以便响应来自其他应用的 Intent。 您可以通过将 <intent-filter> 元素作为组件声明元素的子项进行添加来为您的组件声明 Intent 过滤器。

例如，如果您开发的电子邮件应用包含一个用于撰写新电子邮件的 Activity，则可以像下面这样声明一个 Intent 过滤器来响应“send” Intent（以发送新电子邮件）：

```
<manifest ... >
...
<application ... >
    <activity android:name="com.example.project.ComposeEmailActivity">
        <intent-filter>
            <action android:name="android.intent.action.SEND" />
            <data android:type="*/*" />
            <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
    </activity>
</application>
</manifest>
```

然后，如果另一个应用创建了一个包含 ACTION_SEND 操作的 Intent，并将其传递到 startActivity()，则系统可能会启动您的 Activity，以便用户能够草拟并发送电子邮件。

如需了解有关创建 Intent 过滤器的详细信息，请参阅 Intent 和 Intent 过滤器文档。

声明应用要求

基于 Android 系统的设备多种多样，并非所有设备都提供相同的特性和功能。 为防止将您的应用安装在缺少应用所需特性的设备上，您必须通过在清单文件中声明设备和软件要求，为您的应用支持的设备类型明确定义一个配置文件。 其中的大多数声明只是为了提供信息，系统不会读取它们，但 Google Play 等外部服务会读取它们，以便当用户在其设备中搜索应用时为用户提供过滤功能。

例如，如果您的应用需要相机，并使用 Android 2.1（API 级别 7）中引入的 API，您应该像下面这样在清单文件中以要求形式声明这些信息：

```
<manifest ... >
<uses-feature android:name="android.hardware.camera.any"
```

```
        android:required="true" />
        <uses-sdk android:minSdkVersion="7" android:targetSdkVersion="19" />
    ...
</manifest>
```

现在，没有相机且 Android 版本低于 2.1 的设备将无法从 Google Play 安装您的应用。不过，您也可以声明您的应用使用相机，但并不要求必须使用。 在这种情况下，您的应用必须将 required 属性设置为 "false"，并在运行时检查设备是否具有相机，然后根据需要停用任何相机功能。

设备兼容性文档中提供了有关如何管理应用与不同设备兼容性的详细信息。

应用资源

Android 应用并非只包含代码 — 它还需要与源代码分离的资源，如图像、音频文件以及任何与应用的视觉呈现有关的内容。 例如，您应该通过 XML 文件定义 Activity 用户界面的动画、菜单、样式、颜色和布局。 使用应用资源能够在不修改代码的情况下轻松地更新应用的各种特性，并可通过提供备用资源集让您能够针对各种设备配置（如不同的语言和屏幕尺寸）优化您的应用。

对于您的 Android 项目中包括的每一项资源，SDK 构建工具都会定义一个唯一的整型 ID，您可以利用它来引用应用代码或 XML 中定义的其他资源中的资源。 例如，如果您的应用包含一个名为 logo.png 的图像文件（保存在 res/drawable/ 目录中），则 SDK 工具会生成一个名为 R.drawable.logo 的资源 ID，您可以利用它来引用该图像并将其插入您的用户界面。

提供与源代码分离的资源的其中一个最重要优点在于，您可以提供针对不同设备配置的备用资源。 例如，通过在 XML 中定义 UI 字符串，您可以将字符串翻译为其他语言，并将这些字符串保存在单独的文件中。 然后，Android 系统会根据向资源目录名称追加的语言限定符（如为法语字符串值追加 res/values-fr/）和用户的语言设置，对您的 UI 应用相应的语言字符串。

Android 支持许多不同的备用资源限定符。限定符是一种加入到资源目录名称中，用来定义这些资源适用的设备配置的简短字符串。 再举一例，您应该经常会根据设备的屏幕方向和尺寸为 Activity 创建不同的布局。 例如，当设备屏幕为纵向（长型）时，您可能想要一种垂直排列按钮的布局；但当屏幕为横向（宽型）时，应按水平方向排列按钮。 要想根据方向更改布局，您可以定义两种不同的布局，然后对每个布局的目录名称应用相应的限定符。 然后，系统会根据当前设备方向自动应用相应的布局。

如需了解有关可以在应用中包括的不同资源类型以及如何针对不同设备配置创建备用资源的详细信息，请阅读提供资源。