应用清单

本文内容

每个应用的根目录中都必须包含一个 `AndroidManifest.xml` 文件（且文件名精确无误）。 清单文件向 Android 系统提供应用的必要信息，系统必须具有这些信息方可运行应用的任何代码。

此外，清单文件还可执行以下操作：

- 为应用的 Java 软件包命名。软件包名称充当应用的唯一标识符。
- 描述应用的各个组件，包括构成应用的 Activity、服务、广播接收器和内容提供程序。它还为实现每个组件的类命名并发布其功能，例如它们可以处理的 `Intent` 消息。这些声明向 Android 系统告知有关组件以及可以启动这些组件的条件的信息。
- 确定托管应用组件的进程。
- 声明应用必须具备哪些权限才能访问 API 中受保护的部分并与其他应用交互。还声明其他应用与该应用组件交互所需具备的权限
- 列出 `Instrumentation` 类，这些类可在应用运行时提供分析和其他信息。这些声明只会在应用处于开发阶段时出现在清单中，在应用发布之前将移除。
- 声明应用所需的最低 Android API 级别
- 列出应用必须链接到的库

> **注**：准备要在 Chromebook 上运行的 Android 应用时，要考虑一些重要的硬件和软件功能限制。如需了解详细信息，请参阅 Chromebook 的应用清单兼容性文档。

清单文件结构

下面的代码段显示了清单文件的通用结构及其可包含的每个元素。每个元素及其所有属性全部记录在一个单独的文件中。

> **提示**：要查看本文档提及的任何元素的详细信息，只需点按元素名称。

下面是清单文件的示例：

```xml
<?xml version="1.0" encoding="utf-8"?>

<manifest>

    <uses-permission />
    <permission />
    <permission-tree />
    <permission-group />
    <instrumentation />
    <uses-sdk />
    <uses-configuration />
    <uses-feature />
    <supports-screens />
    <compatible-screens />
    <supports-gl-texture />

    <application>

        <activity>
            <intent-filter>
                <action />
                <category />
                <data />
            </intent-filter>
            <meta-data />
        </activity>

        <activity-alias>
            <intent-filter> . . . </intent-filter>
            <meta-data />
        </activity-alias>

        <service>
            <intent-filter> . . . </intent-filter>
            <meta-data/>
        </service>

        <receiver>
            <intent-filter> . . . </intent-filter>
            <meta-data />
        </receiver>

        <provider>
            <grant-uri-permission />
            <meta-data />
            <path-permission />
        </provider>

        <uses-library />

    </application>

</manifest>
```

以下列表包含可出现在清单文件中的所有元素，按字母顺序列出：

- <action>
- <activity>
- <activity-alias>
- <application>
- <category>
- <data>
- <grant-uri-permission>
- <instrumentation>

- <intent-filter>
- <manifest>
- <meta-data>
- <permission>
- <permission-group>
- <permission-tree>
- <provider>
- <receiver>
- <service>
- <supports-screens>
- <uses-configuration>
- <uses-feature>
- <uses-library>
- <uses-permission>
- <uses-sdk>

> **注**：这些是仅有的合法元素 – 您无法添加自己的元素或属性。

文件约定

本节描述普遍适用于清单文件中所有元素和属性的约定和规则。

元素

只有 <manifest> 和 <application> 元素是必需的，它们都必须存在并且只能出现一次。其他大部分元素可以出现多次或者根本不出现。但清单文件中必须至少存在其中某些元素才有用。

如果一个元素包含某些内容，也就包含其他元素。所有值均通过属性进行设置，而不是通过元素内的字符数据设置。

同一级别的元素通常不分先后顺序。例如，<activity>、<provider> 和 <service> 元素可以按任何顺序混合在一起。这条规则有两个主要例外：

- <activity-alias> 元素必须跟在别名所指的 <activity> 之后。
- <application> 元素必须是 <manifest> 元素内最后一个元素。换言之，</manifest> 结束标记必须紧接在 </application> 结束标记后。

属性

从某种意义上说，所有属性都是可选的。但是，必须指定某些属性，元素才可实现其目的。请使用本文档作为参考。对于真正可选的属性，它将指定默认值或声明缺乏规范时将执行何种操作。

除了根 <manifest> 元素的一些属性外，所有属性名称均以 android: 前缀开头。例如，android:alwaysRetainTaskState。由于该前缀是通用的，因此在按名称引用属性时，本文档通常会将其忽略。

声明类名

许多元素对应于 Java 对象，包括应用本身的元素（<application> 元素）及其主要组件：Activity (<activity>)、服务 (<service>)、广播接收器 (<receiver>) 以及内容提供程序 (<provider>)。

如果按照您针对组件类（Activity、Service 和 BroadcastReceiverContentProvider）几乎一直采用的方式来定义子类，则该子类需通过 name 属性来声明。该名称必须包含完整的软件包名称。例如，Service 子类可能会声明如下：

```
<manifest . . . >
    <application . . . >
        <service android:name="com.example.project.SecretService" . . . >
            . . .
        </service>
        . . .
    </application>
</manifest>
```

但是，如果字符串的第一个字符是句点，则应用的软件包名称（如 `<manifest>` 元素的 `package` 属性所指定）将附加到该字符串。以下赋值与上述方法相同：

```
<manifest package="com.example.project" . . . >
    <application . . . >
        <service android:name=".SecretService" . . . >
            . . .
        </service>
        . . .
    </application>
</manifest>
```

当启动组件时，Android 系统会创建已命名子类的实例。如果未指定子类，则会创建基类的实例。

多个值

如果可以指定多个值，则几乎总是在重复此元素，而不是列出单个元素内的多个值。例如，intent 过滤器可以列出多个操作：

```
<intent-filter . . . >
    <action android:name="android.intent.action.EDIT" />
    <action android:name="android.intent.action.INSERT" />
    <action android:name="android.intent.action.DELETE" />
    . . .
</intent-filter>
```

资源值

某些属性的值可以显示给用户，例如，Activity 的标签和图标。这些属性的值应该本地化，并通过资源或主题进行设置。资源值用以下格式表示：

```
@[<i>package</i>:]<i>type</i>/<i>name</i>
```

如果资源与应用在同一个软件包中，可以省略*软件包*名称。*类型*是资源类型，例如*字符串*或*可绘制对象*，*名称*是标识特定资源的名称。下面是示例：

```
<activity android:icon="@drawable/smallPic" . . . >
```

主题中的值用类似的方法表示，但是以 `?` 开头，而不是以 `@` 开头：

```
?[<i>package</i>:]<i>type</i>/<i>name</i>
```

字符串值

如果属性值为字符串，则必须使用双反斜杠 (`\\`) 转义字符，例如，使用 `\\n` 表示换行符或使用 `\\uxxxx` 表示 Unicode 字符。

文件功能

下文介绍在清单文件中体现某些 Android 特性的方式。

Intent 过滤器

应用的核心组件（例如其 Activity、服务和广播接收器）由 *intent* 激活。Intent 是一系列用于描述所需操作的信息（`Intent` 对象），其中包括要执行操作的数据、应执行操作的组件类别以及其他相关说明。Android 系统会查找合适的组件来响应 intent，根据需要启动组件的新实例，并将其传递到 `Intent` 对象。

组件将通过 *intent 过滤器*公布它们可响应的 intent 类型。由于Android 系统在启动某组件之前必须了解该组件可以处理的 intent，因此 intent 过滤器在清单中被指定为 `<intent-filter>` 元素。一个组件可有任意数量的过滤器，其中每个过滤器描述一种不同的功能。

显式命名目标组件的 intent 将激活该组件，因此过滤器不起作用。不按名称指定目标的 intent 只有在能够通过组件的一个过滤器时才可激活该组件。

如需了解有关如何根据 intent 过滤器测试 `Intent` 对象的信息，请参阅 Intent 和 Intent 过滤器文档。

图标和标签

对于可以显示给用户的小图标和文本标签，大量元素具有 `icon` 和 `label` 属性。此外，对于同样可以显示在屏幕上的较长说明文本，某些元素还具有 `description` 属性。例如，`<permission>` 元素具有所有这三个属性。因此，当系统询问用户是否授权给请求获得权限的应用时，权限图标、权限名称以及所需信息的说明均会呈现给用户。

无论何种情况下，在包含元素中设置的图标和标签都将成为所有容器子元素的默认 `icon` 和 `label` 设置。因此，在 `<application>` 元素中设置的图标和标签是每个应用组件的默认图标和标签。同样，为组件（例如 `<activity>` 元素）设置的图标和标签是组件每个 `<intent-filter>` 元素的默认设置。如果 `<application>` 元素设置标签，但是 Activity 及其 intent 过滤器不执行此操作，则应用标签将被视为 Activity 和 intent 过滤器的标签。

在实现过滤器公布的功能时，只要向用户呈现组件，系统便会使用为 intent 过滤器设置的图标和标签表示该组件。例如，具有 `android.intent.action.MAIN` 和 `android.intent.category.LAUNCHER` 设置的过滤器将 Activity 公布为可启动应用的功能，即，公布为应显示在应用启动器中的功能。在过滤器中设置的图标和标签显示在启动器中。

权限

*权限*是一种限制，用于限制对部分代码或设备上数据的访问。施加限制是为了保护可能被误用以致破坏或损害用户体验的关键数据和代码。

每种权限均由一个唯一的标签标识。标签通常指示受限制的操作。以下是 Android 定义的一些权限：

- `android.permission.CALL_EMERGENCY_NUMBERS`
- `android.permission.READ_OWNER_DATA`
- `android.permission.SET_WALLPAPER`
- `android.permission.DEVICE_POWER`

一个功能只能由一种权限保护。

如果应用需要访问受权限保护的功能，则必须在清单中使用 `<uses-permission>` 元素声明应用需要该权限。将应用安装到设备上之后，安装程序会通过检查签署应用证书的颁发机构并（在某些情况下）询问用户，确定是否授予请求的权限。如果授予权限，则应用能够使用受保护的功能。否则，其访问这些功能的尝试将会失败，并且不会向用户发送任何通知。

应用也可以使用权限保护自己的组件。它可以采用由 Android 定义（如 `android.Manifest.permission` 中所列）或由其他应用声明的任何权限。它也可以定义自己的权限。新权限用 `<permission>` 元素来声明。例如，Activity 可受到如下保护：

```
<manifest . . . >
    <permission android:name="com.example.project.DEBIT_ACCT" . . . />
    <uses-permission android:name="com.example.project.DEBIT_ACCT" />
    . . .
    <application . . .>
        <activity android:name="com.example.project.FreneticActivity"
                  android:permission="com.example.project.DEBIT_ACCT"
                  . . . >
            . . .
        </activity>
    </application>
</manifest>
```

请注意，在此示例中，DEBIT_ACCT 权限不仅是通过 <permission> 元素来声明，而且其使用也是通过 <uses-permission> 元素来请求。要让应用的其他组件也能够启动受保护的 Activity，您必须请求其使用权限，即便保护是由应用本身施加的亦如此。

同样还是在此示例中，如果将 permission 属性设置为在其他位置（例如，android.permission.CALL_EMERGENCY_NUMBERS）声明的权限，则无需使用 <permission> 元素再次声明。但是，仍有必要通过 <uses-permission> 请求其使用权限。

<permission-tree> 元素声明为代码中定义的一组权限声明命名空间，<permission-group> 为一组权限定义标签，包括在清单中使用 <permission> 元素声明的权限以及在其他位置声明的权限。这只影响如何对提供给用户的权限进行分组。<permission-group> 元素并不指定属于该组的权限，而只是为组提供名称。可通过向 <permission> 元素的 permissionGroup 属性分配组名，将权限放入组中。

库

每个应用均链接到默认的 Android 库，该库中包括用于开发应用（以及通用类，如 Activity、服务、intent、视图、按钮、应用、ContentProvider）的基本软件包。

但是，某些软件包驻留在自己的库中。如果应用使用来自其中任一软件包的代码，则必须明确要求其链接到这些软件包。清单必须包含单独的 <uses-library> 元素来命名其中每个库。库名称可在软件包的文档中找到。