# 可绘制对象资源

另请参阅

- 2D 图形
- Vector Asset Studio

可绘制对象资源是一般概念，是指可在屏幕上绘制的图形，以及可以使用 `getDrawable(int)` 等 API 检索或者应用到具有 `android:drawable` 和 `android:icon` 等属性的其他 XML 资源的图形。共有多种不同类型的可绘制对象：

### 位图文件

位图图形文件（`.png`、`.jpg` 或 `.gif`）。创建 `BitmapDrawable`。

### 九宫格文件

具有可拉伸区域的 PNG 文件，允许根据内容调整图像大小 (`.9.png`)。创建 `NinePatchDrawable`。

### 图层列表

管理其他可绘制对象阵列的可绘制对象。它们按阵列顺序绘制，因此索引最大的元素绘制在顶部。创建 `LayerDrawable`。

### 状态列表

此 XML 文件为不同状态引用不同位图图形（例如，按下按钮时使用不同的图像）。创建 `StateListDrawable`。

### 级别列表

此 XML 文件用于定义管理大量备选可绘制对象的可绘制对象，每个可绘制对象都分配有最大的备选数量。创建 `LevelListDrawable`。

### 转换可绘制对象

此 XML 文件用于定义可在两种可绘制对象资源之间交错淡出的可绘制对象。创建 `TransitionDrawable`。

### 插入可绘制对象

此 XML 文件用于定义以指定距离插入其他可绘制对象的可绘制对象。当视图需要小于视图实际边界的背景可绘制对象时，此类可绘制对象很有用。

### 裁剪可绘制对象

此 XML 文件用于定义对其他可绘制对象进行裁剪（根据其当前级别值）的可绘制对象。创建 `ClipDrawable`。

### 缩放可绘制对象

此 XML 文件用于定义更改其他可绘制对象大小（根据其当前级别值）的可绘制对象。创建 `ScaleDrawable`

### 形状可绘制对象

此 XML 文件用于定义几何形状（包括颜色和渐变）。创建 `ShapeDrawable`。

另请参阅动画资源文档，了解如何创建 `AnimationDrawable`。

> **注**：颜色资源也可用作 XML 中的可绘制对象。例如，在创建状态列表可绘制对象时，可以引用 `android:drawable` 属性的颜色资源 (`android:drawable="@color/green"`)。

# 位图

位图图像。Android 支持以下三种格式的位图文件：`.png`（首选）、`.jpg`（可接受）、`.gif`（不建议）。

您可以使用文件名作为资源 ID 直接引用位图文件，也可以在 XML 中创建别名资源 ID。

> **注**：在构建过程中，可通过 `aapt` 工具自动优化位图文件，对图像进行无损压缩。例如，不需要超过 256 色的真彩色 PNG 可通过调色板转换为 8 位 PNG。这样产生的图像质量相同，但所需内存更少。因此请注意，此目录中的图像二进制文件在构建时可能会发生变化。如果您计划将图像解读为比特流以将其转换为位图，请改为将图像放在 `res/raw/` 文件夹中，在那里它们不会进行优化。

## 位图文件

位图文件是 `.png`、`.jpg` 或 `.gif` 文件。当您将这些文件保存到 `res/drawable/` 目录中时，Android 将为它们创建 `Drawable` 资源。

文件位置：

> `res/drawable/*filename*.png`（`.png`、`.jpg` 或 `.gif`）
> 文件名用作资源 ID。

编译的资源数据类型：

> 指向 `BitmapDrawable` 的资源指针。

资源引用：

> 在 Java 中：`R.drawable.*filename*`
> 在 XML 中：`@[*package*:]drawable/*filename*`

示例：

> 当图像保存为 `res/drawable/myimage.png` 后，此布局 XML 会将图像应用到视图：
>
> ```
> <ImageView
>     android:layout_height="wrap_content"
>     android:layout_width="wrap_content"
>     android:src="@drawable/myimage" />
> ```
>
> 以下应用代码将图像作为 `Drawable` 检索：
>
> ```
> Resources res = getResources();
> Drawable drawable = res.getDrawable(R.drawable.myimage);
> ```

另请参阅：

- 2D 图形
- `BitmapDrawable`

## XML 位图

XML 位图是在 XML 中定义的资源，指向位图文件。实际上是原始位图文件的别名。XML 可以指定位图的其他属性，例如抖动和层叠。

> **注**：您可以将 `<bitmap>` 元素用作 `<item>` 元素的子项。例如，在创建状态列表或图层列表时，可以将 `android:drawable` 属性从 `<item>` 元素中排除，并在其中嵌套用于定义可绘制项的 `<bitmap>`。

文件位置：

`res/drawable/*filename*.xml`
文件名用作资源 ID。

编译的资源数据类型：

指向 `BitmapDrawable` 的资源指针。

资源引用：

在 Java 中：`R.drawable.*filename*`
在 XML 中：`@[*package*:]drawable/*filename*`

语法：

```
<?xml version="1.0" encoding="utf-8"?>
<bitmap
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:src="@[package:]drawable/drawable_resource"
    android:antialias=["true" | "false"]
    android:dither=["true" | "false"]
    android:filter=["true" | "false"]
    android:gravity=["top" | "bottom" | "left" | "right" | "center_vertical" |
                     "fill_vertical" | "center_horizontal" | "fill_horizontal" |
                     "center" | "fill" | "clip_vertical" | "clip_horizontal"]
    android:mipMap=["true" | "false"]
    android:tileMode=["disabled" | "clamp" | "repeat" | "mirror"] />
```

元素：

`<bitmap>`

定义位图来源及其属性。

属性：

`xmlns:android`

*字符串*。定义 XML 命名空间，其必须是 `"http://schemas.android.com/apk/res/android"`。这仅当 `<bitmap>` 是根元素时才需要，当 `<bitmap>` 嵌套在 `<item>` 内时不需要。

`android:src`

*可绘制对象资源*。**必备**。引用可绘制对象资源。

`android:antialias`

*布尔值*。启用或停用抗锯齿。

`android:dither`

*布尔值*。当位图的像素配置与屏幕不同时（例如：ARGB 8888 位图和 RGB 565 屏幕），启用或停用位图抖动。

`android:filter`

*布尔值*。启用或停用位图过滤。当位图收缩或拉伸以使其外观平滑时使用过滤。

`android:gravity`

*关键字*。定义位图的重力。重力指示当位图小于容器时，可绘制对象在其容器中放置的位置。

必须是以下一个或多个（用 '|' 分隔）常量值：

| | |
|---|---|

| 值 | 说明 |
| --- | --- |
| top | 将对象放在其容器顶部，不改变其大小。 |
| bottom | 将对象放在其容器底部，不改变其大小。 |
| left | 将对象放在其容器左边缘，不改变其大小。 |
| right | 将对象放在其容器右边缘，不改变其大小。 |
| center_vertical | 将对象放在其容器的垂直中心，不改变其大小。 |
| fill_vertical | 按需要扩展对象的垂直大小，使其完全适应其容器。 |
| center_horizontal | 将对象放在其容器的水平中心，不改变其大小。 |
| fill_horizontal | 按需要扩展对象的水平大小，使其完全适应其容器。 |
| center | 将对象放在其容器的水平和垂直轴中心，不改变其大小。 |
| fill | 按需要扩展对象的垂直大小，使其完全适应其容器。这是默认值。 |
| clip_vertical | 可设置为让子元素的上边缘和/或下边缘裁剪至其容器边界的附加选项。裁剪基于垂直重力：顶部重力裁剪上边缘，底部重力裁剪下边缘，任一重力不会同时裁剪两边。 |
| clip_horizontal | 可设置为让子元素的左边和/或右边裁剪至其容器边界的附加选项。裁剪基于水平重力：左边重力裁剪右边缘，右边重力裁剪左边缘，任一重力不会同时裁剪两边。 |

`android:mipMap`

*布尔值*。启用或停用 mipmap 提示。如需了解详细信息，请参阅 `setHasMipMap()`。默认值为 false。

`android:tileMode`

*关键字*。定义平铺模式。当平铺模式启用时，位图会重复。重力在平铺模式启用时将被忽略。

必须是以下常量值之一：

| 值 | 说明 |
| --- | --- |
| disabled | 不平铺位图。这是默认值。 |
| clamp | 当着色器绘制范围超出其原边界时复制边缘颜色 |
| repeat | 水平和垂直重复着色器的图像。 |
| mirror | 水平和垂直重复着色器的图像，交替镜像图像以使相邻图像始终相接。 |

示例：

```
<?xml version="1.0" encoding="utf-8"?>
<bitmap xmlns:android="http://schemas.android.com/apk/res/android"
    android:src="@drawable/icon"
    android:tileMode="repeat" />
```

另请参阅：

- `BitmapDrawable`
- 创建别名资源

# 九宫格

`NinePatch` 是一种 PNG 图像，在其中可定义当视图中的内容超出正常图像边界时 Android 缩放的可拉伸区域。此类图像通常指定为至少有一个尺寸设置为 `"wrap_content"` 的视图的背景，而且当视图扩展以适应内容时，九宫格图像也会扩展以匹配视图的大小。Android 的标准 `Button` 小部件使用的背景就是典型的九宫格图像，其必须拉伸以适应按钮内的文本（或图像）。

与普通位图一样，您可以直接引用九宫格文件，也可以从 XML 定义的资源引用。

如需有关如何创建包含可拉伸区域的九宫格文件的完整论述，请参阅 2D 图形文件。

# 九宫格文件

文件位置：

`res/drawable/filename.9.png`
文件名用作资源 ID。

编译的资源数据类型：

指向 `NinePatchDrawable` 的资源指针。

资源引用：

在 Java 中：`R.drawable.filename`
在 XML 中：`@[package:]drawable/filename`

示例：

当图像保存为 `res/drawable/myninepatch.9.png` 后，此布局 XML 会将九宫格应用到视图：

```
<Button
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:background="@drawable/myninepatch" />
```

另请参阅：

- 2D 图形
- `NinePatchDrawable`

# XML 九宫格

XML 九宫格是在 XML 中定义的资源，指向九宫格文件。XML 可以为图像指定抖动。

文件位置：

`res/drawable/filename.xml`
文件名用作资源 ID。

编译的资源数据类型：

指向 `NinePatchDrawable` 的资源指针。

资源引用：

在 Java 中：`R.drawable.filename`
在 XML 中：`@[package:]drawable/filename`

语法：

```
<?xml version="1.0" encoding="utf-8"?>
<nine-patch
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:src="@[package:]drawable/drawable_resource"
    android:dither=["true" | "false"] />
```

元素：

`<nine-patch>`

定义九宫格来源及其属性。

属性：

`xmlns:android`

*字符串*。**必备。**定义 XML 命名空间，其必须是 `"http://schemas.android.com/apk/res/android"`。

`android:src`

*可绘制对象资源*。**必备**。引用九宫格文件。

`android:dither`

*布尔值*。当位图的像素配置与屏幕不同时（例如：ARGB 8888 位图和 RGB 565 屏幕），启用或停用位图抖动。

示例：

```
<?xml version="1.0" encoding="utf-8"?>
<nine-patch xmlns:android="http://schemas.android.com/apk/res/android"
    android:src="@drawable/myninepatch"
    android:dither="false" />
```

# 图层列表

`LayerDrawable` 是管理其他可绘制对象阵列的可绘制对象。列表中的每个可绘制对象按照列表的顺序绘制，列表中的最后一个可绘制对象绘于顶部。

每个可绘制对象由单一 `<layer-list>` 元素内的 `<item>` 元素表示。

文件位置：

`res/drawable/filename.xml`
文件名用作资源 ID。

编译的资源数据类型：

指向 `LayerDrawable` 的资源指针。

资源引用：

在 Java 中：`R.drawable.filename`
在 XML 中：`@[package:]drawable/filename`

语法：

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list
    xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:drawable="@[package:]drawable/drawable_resource"
        android:id="@[+][package:]id/resource_name"
        android:top="dimension"
        android:right="dimension"
        android:bottom="dimension"
        android:left="dimension" />
</layer-list>
```

元素：

`<layer-list>`

**必备。**这必须是根元素。包含一个或多个 `<item>` 元素。

属性：

`xmlns:android`

*字符串*。**必备。**定义 XML 命名空间，其必须是 `"http://schemas.android.com/apk/res/android"`。

`<item>`

定义要放在图层可绘制对象中由其属性定义的位置的可绘制对象。必须是 `<selector>` 元素的子项。接受子 `<bitmap>` 元素。

属性：

`android:drawable`

*可绘制对象资源*。**必备**。引用可绘制对象资源。

`android:id`

*资源 ID*。此可绘制对象的唯一资源 ID。要为此项新建资源 ID，请使用以下形式：`"@+id/name"`。加号表示应创建为新 ID。可以使用此 ID 检索和修改具有 `View.findViewById()` 或 `Activity.findViewById()` 的可绘制对象。

`android:top`

*整型*。顶部偏移（像素）。

`android:right`

*整型*。右边偏移（像素）。

`android:bottom`

*整型*。底部偏移（像素）。

`android:left`

*整型*。左边偏移（像素）。

默认情况下，所有可绘制项都会缩放以适应包含视图的大小。因此，将图像放在图层列表中的不同位置可能会增大视图的大小，并且有些图像会相应地缩放。为避免缩放列表中的项目，请在 `<item>` 元素内使用 `<bitmap>` 元素指定可绘制对象，并且对某些不缩放的项目（例如 `"center"`）定义重力。例如，以下 `<item>` 定义缩放以适应其容器视图的项目：

```
<item android:drawable="@drawable/image" />
```

为避免缩放，以下示例使用重力居中的 `<bitmap>` 元素：

```
<item>
  <bitmap android:src="@drawable/image"
          android:gravity="center" />
</item>
```

示例：

XML 文件保存在 `res/drawable/layers.xml` 中：

```xml
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item>
      <bitmap android:src="@drawable/android_red"
        android:gravity="center" />
    </item>
    <item android:top="10dp" android:left="10dp">
      <bitmap android:src="@drawable/android_green"
        android:gravity="center" />
    </item>
    <item android:top="20dp" android:left="20dp">
      <bitmap android:src="@drawable/android_blue"
        android:gravity="center" />
    </item>
</layer-list>
```

请注意，此示例使用嵌套的 `<bitmap>` 元素为每个具有“中心”重力的项目定义可绘制对象资源。这可确保没有图像会为了适应容器的大小而缩放，因为偏移图像会造成大小调整。

此布局 XML 会将可绘制对象应用到视图：

```xml
<ImageView
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:src="@drawable/layers" />
```

结果导致一堆不断偏移的图像：

另请参阅：

- `LayerDrawable`

# 状态列表

`StateListDrawable` 是在 XML 中定义的可绘制对象，它根据对象的状态，使用多个不同的图像来表示同一个图形。例如，`Button` 小部件可以是多种不同状态（按下、聚焦或这两种状态都不是）中的其中一种，而且可以利用状态列表可绘制对象为每种状态提供不同的背景图片。

您可以在 XML 文件中描述状态列表。每个图形由单一 `<selector>` 元素内的 `<item>` 元素表示。每个 `<item>` 均使用各种属性来描述应用作可绘制对象的图形的状态。

在每个状态变更期间，将从上到下遍历状态列表，并使用第一个与当前状态匹配的项目 —此选择并非基于“最佳匹配”，而是选择符合状态最低条件的第一个项目。

文件位置：

`res/drawable/filename.xml`
文件名用作资源 ID。

编译的资源数据类型：

指向 `StateListDrawable` 的资源指针。

资源引用：

在 Java 中：`R.drawable.filename`
在 XML 中：`@[package:]drawable/filename`

语法：

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android"
    android:constantSize=["true" | "false"]
    android:dither=["true" | "false"]
    android:variablePadding=["true" | "false"] >
    <item
        android:drawable="@[package:]drawable/drawable_resource"
        android:state_pressed=["true" | "false"]
        android:state_focused=["true" | "false"]
        android:state_hovered=["true" | "false"]
        android:state_selected=["true" | "false"]
        android:state_checkable=["true" | "false"]
        android:state_checked=["true" | "false"]
        android:state_enabled=["true" | "false"]
        android:state_activated=["true" | "false"]
        android:state_window_focused=["true" | "false"] />
</selector>
```

元素：

`<selector>`

**必备。**这必须是根元素。包含一个或多个 `<item>` 元素。

属性：

`xmlns:android`

*字符串*。**必备。**定义 XML 命名空间，其必须是 `"http://schemas.android.com/apk/res/android"`。

`android:constantSize`

*布尔值*。如果可绘制对象报告的内部大小在状态变更时保持不变，则值为“true”（大小是所有状态的最大值）；如果大小根据当前状态而变化，则值为“false”。默认值为 false。

`android:dither`

*布尔值*。值为“true”时，将在位图的像素配置与屏幕不同时（例如：ARGB 8888 位图和 RGB 565 屏幕）启用位图的抖动；值为“false”时则停用抖动。默认值为 true。

`android:variablePadding`

*布尔值*。如果可绘制对象的内边距应根据选择的当前状态而变化，则值为“true”；如果内边距应保持不变（基于所有状态的最大内边距），则值为“false”。启用此功能要求您在状态变更时处理执行布局，这通常不受支持。默认值为 false。

`<item>`

定义要在某些状态期间使用的可绘制对象，如其属性所述。必须是 `<selector>` 元素的子项。

属性：

`android:drawable`

*可绘制对象资源*。**必备**。引用可绘制对象资源。

`android:state_pressed`

*布尔值*。如果在按下对象（例如触摸/点按某按钮）时应使用此项目，则值为“true”；如果在默认的未按下状态时应使用此项目，则值为“false”。

`android:state_focused`

*布尔值*。如果在对象具有输入焦点（例如当用户选择文本输入时）时应使用此项目，则值为“true”；如果在默认的非焦点状态时应使用此项目，则值为“false”。

`android:state_hovered`

*布尔值*。如果当光标悬停在对象上时应使用此项目，则值为“true”；如果在默认的非悬停状态时应使用此项目，则值为“false”。通常，这个可绘制对象可能与用于“聚焦”状态的可绘制对象相同。

此项为 API 级别 14 新引入的配置。

`android:state_selected`

*布尔值*。如果在使用定向控件浏览（例如使用方向键浏览列表）的情况下对象为当前用户选择时应使用此项目，则值为“true”；如果在未选择对象时应使用此项目，则值为“false”。

当焦点 (`android:state_focused`) 不充分（例如，列表视图有焦点但使用方向键选择其中的项目）时，使用所选状态。

`android:state_checkable`

*布尔值*。如果当对象可选中时应使用此项目，则值为“true”；如果当对象不可选中时应使用此项目，则值为“false”。（仅当对象可在可选中与不可选中小部件之间转换时才有用。）

`android:state_checked`

*布尔值*。如果在对象已选中时应使用此项目，则值为“true”；如果在对象未选中时应使用此项目，则值为“false”。

`android:state_enabled`

*布尔值*。如果在对象启用（能够接收触摸/点击事件）时应使用此项目，则值为“true”；如果在对象停用时应使用此项目，则值为“false”。

`android:state_activated`

*布尔值*。如果在对象激活作为持续选择（例如，在持续导航视图中“突出显示”之前选中的列表项）时应使用此项目，则值为“true”；如果在对象未激活时应使用此项目，则值为“false”。

此项为 API 级别 11 新引入的配置。

`android:state_window_focused`

*布尔值*。如果当应用窗口有焦点（应用在前台）时应使用此项目，则值为“true”；如果当应用窗口没有焦点（例如，通知栏下拉或对话框出现）时应使用此项目，则值为“false”。

> **注**：请记住，Android 将应用状态列表中第一个与对象当前状态匹配的项目。因此，如果列表中的第一个项目不含上述任何状态属性，则每次都会应用它，这就是默认值应始终放在最后的原因（如以下示例所示）。

示例：

XML 文件保存在 `res/drawable/button.xml` 中：

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="true"
          android:drawable="@drawable/button_pressed" /> <!-- pressed -->
    <item android:state_focused="true"
          android:drawable="@drawable/button_focused" /> <!-- focused -->
    <item android:state_hovered="true"
          android:drawable="@drawable/button_focused" /> <!-- hovered -->
    <item android:drawable="@drawable/button_normal" /> <!-- default -->
</selector>
```

此布局 XML 将状态列表可绘制对象应用到按钮：

```
<Button
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:background="@drawable/button" />
```

另请参阅：

- StateListDrawable

# 级别列表

管理大量备选可绘制对象的可绘制对象，每个可绘制对象都分配有最大的备选数量。使用 `setLevel()` 设置可绘制对象的级别值会加载级别列表中 `android:maxLevel` 值大于或等于传递到方法的值的可绘制对象资源。

文件位置：

`res/drawable/filename.xml`
文件名用作资源 ID。

编译的资源数据类型：

指向 `LevelListDrawable` 的资源指针。

资源引用：

在 Java 中：`R.drawable.filename`
在 XML 中：`@[package:]drawable/filename`

语法：

```
<?xml version="1.0" encoding="utf-8"?>
<level-list
    xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:drawable="@drawable/drawable_resource"
        android:maxLevel="integer"
        android:minLevel="integer" />
</level-list>
```

元素：

`<level-list>`

这必须是根元素。包含一个或多个 `<item>` 元素。

属性：

`xmlns:android`

*字符串*。**必备。**定义 XML 命名空间，其必须是 `"http://schemas.android.com/apk/res/android"`。

`<item>`

定义要在某特定级别使用的可绘制对象。

属性：

`android:drawable`

*可绘制对象资源*。**必备**。引用要插入的可绘制对象资源。

`android:maxLevel`

*整型*。此项目允许的最高级别。

`android:minLevel`

*整型*。此项目允许的最低级别。

示例：

```
<?xml version="1.0" encoding="utf-8"?>
<level-list xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:drawable="@drawable/status_off"
        android:maxLevel="0" />
    <item
        android:drawable="@drawable/status_on"
        android:maxLevel="1" />
</level-list>
```

在此项目应用到 `View` 后，可通过 `setLevel()` 或 `setImageLevel()` 更改级别。

另请参阅：

- `LevelListDrawable`

# 转换可绘制对象

`TransitionDrawable` 是可在两种可绘制对象资源之间交错淡出的可绘制对象。

每个可绘制对象由单一 `<transition>` 元素内的 `<item>` 元素表示。不支持超过两个项目。要向前转换，请调用 `startTransition()`。要向后转换，则调用 `reverseTransition()`。

文件位置：

`res/drawable/filename.xml`
文件名用作资源 ID。

编译的资源数据类型：

指向 `TransitionDrawable` 的资源指针。

资源引用：

在 Java 中：`R.drawable.filename`
在 XML 中：`@[package:]drawable/filename`

语法：

```
<?xml version="1.0" encoding="utf-8"?>
<transition
xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:drawable="@[package:]drawable/drawable_resource"
        android:id="@[+][package:]id/resource_name"
        android:top="dimension"
        android:right="dimension"
        android:bottom="dimension"
        android:left="dimension" />
</transition>
```

元素：

`<transition>`

**必备。**这必须是根元素。包含一个或多个 `<item>` 元素。

属性：

`xmlns:android`

*字符串*。**必备。**定义 XML 命名空间，其必须是 `"http://schemas.android.com/apk/res/android"`。

`<item>`

定义要用作可绘制对象转换一部分的可绘制对象。必须是 `<transition>` 元素的子项。接受子 `<bitmap>` 元素。

属性：

`android:drawable`

*可绘制对象资源*。**必备**。引用可绘制对象资源。

`android:id`

*资源 ID*。此可绘制对象的唯一资源 ID。要为此项新建资源 ID，请使用以下形式：`"@+id/name"`。加号表示应创建为新 ID。可以使用此 ID 检索和修改具有 `View.findViewById()` 或 `Activity.findViewById()` 的可绘制对象。

`android:top`

*整型*。顶部偏移（像素）。

`android:right`

*整型*。右边偏移（像素）。

`android:bottom`

*整型*。底部偏移（像素）。

`android:left`

*整型*。左边偏移（像素）。

示例：

XML 文件保存在 `res/drawable/transition.xml` 中：

```
<?xml version="1.0" encoding="utf-8"?>
<transition xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/on" />
    <item android:drawable="@drawable/off" />
</transition>
```

此布局 XML 会将可绘制对象应用到视图：

```
<ImageButton
    android:id="@+id/button"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:src="@drawable/transition" />
```

以下代码从第一个项目到第二个项目执行 500ms 的转换：

```
ImageButton button = (ImageButton) findViewById(R.id.button);
TransitionDrawable drawable = (TransitionDrawable) button.getDrawable();
drawable.startTransition(500);
```

另请参阅：

- TransitionDrawable

# 插入可绘制对象

在 XML 文件中定义的以指定距离插入其他可绘制对象的可绘制对象。当视图需要小于视图实际边界的背景时，此类可绘制对象很有用。

文件位置：

`res/drawable/filename.xml`
文件名用作资源 ID。

编译的资源数据类型：

指向 InsetDrawable 的资源指针。

资源引用：

在 Java 中：`R.drawable.filename`
在 XML 中：`@[package:]drawable/filename`

语法：

```
<?xml version="1.0" encoding="utf-8"?>
<inset
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/drawable_resource"
    android:insetTop="dimension"
    android:insetRight="dimension"
    android:insetBottom="dimension"
    android:insetLeft="dimension" />
```

元素：

`<inset>`

定义插入可绘制对象。这必须是根元素。

属性：

`xmlns:android`

*字符串*。**必备。**定义 XML 命名空间，其必须是 `"http://schemas.android.com/apk/res/android"`。

`android:drawable`

*可绘制对象资源*。**必备**。引用要插入的可绘制对象资源。

`android:insetTop`

*尺寸*。顶部插入，表示为尺寸值或尺寸资源

`android:insetRight`

*尺寸*。右边插入，表示为尺寸值或尺寸资源

android:insetBottom

*尺寸*。底部插入，表示为尺寸值或尺寸资源

android:insetLeft

*尺寸*。左边插入，表示为尺寸值或尺寸资源

示例：

```
<?xml version="1.0" encoding="utf-8"?>
<inset xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/background"
    android:insetTop="10dp"
    android:insetLeft="10dp" />
```

另请参阅：

- InsetDrawable

# 裁剪可绘制对象

在 XML 文件中定义的对其他可绘制对象进行裁剪（根据其当前级别）的可绘制对象。您可以根据级别以及用于控制其在整个容器中位置的重力，来控制子可绘制对象的裁剪宽度和高度。通常用于实现进度栏之类的项目。

文件位置：

res/drawable/*filename*.xml
文件名用作资源 ID。

编译的资源数据类型：

指向 ClipDrawable 的资源指针。

资源引用：

在 Java 中：R.drawable.*filename*
在 XML 中：@[*package*:]drawable/*filename*

语法：

```
<?xml version="1.0" encoding="utf-8"?>
<clip
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/drawable_resource"
    android:clipOrientation=["horizontal" | "vertical"]
    android:gravity=["top" | "bottom" | "left" | "right" | "center_vertical" |
                     "fill_vertical" | "center_horizontal" | "fill_horizontal" |
                     "center" | "fill" | "clip_vertical" | "clip_horizontal"] />
```

元素：

<clip>

定义裁剪可绘制对象。这必须是根元素。

属性：

xmlns:android

*字符串*。**必备**。定义 XML 命名空间，其必须是 `"http://schemas.android.com/apk/res/android"`。

`android:drawable`

*可绘制对象资源*。**必备**。引用要裁剪的可绘制对象资源。

`android:clipOrientation`

*关键字*。裁剪方向。

必须是以下常量值之一：

| 值 | 说明 |
| --- | --- |
| `horizontal` | 水平裁剪可绘制对象。 |
| `vertical` | 垂直裁剪可绘制对象。 |

`android:gravity`

*关键字*。指定可绘制对象中要裁剪的位置。

必须是以下一个或多个（用 '|' 分隔）常量值：

| 值 | 说明 |
| --- | --- |
| `top` | 将对象放在其容器顶部，不改变其大小。当 `clipOrientation` 是 `"vertical"` 时，在可绘制对象的底部裁剪。 |
| `bottom` | 将对象放在其容器底部，不改变其大小。当 `clipOrientation` 是 `"vertical"` 时，在可绘制对象的顶部裁剪。 |
| `left` | 将对象放在其容器左边缘，不改变其大小。这是默认值。当 `clipOrientation` 是 `"horizontal"` 时，在可绘制对象的右边裁剪。这是默认值。 |
| `right` | 将对象放在其容器右边缘，不改变其大小。当 `clipOrientation` 是 `"horizontal"` 时，在可绘制对象的左边裁剪。 |
| `center_vertical` | 将对象放在其容器的垂直中心，不改变其大小。裁剪行为与重力为 `"center"` 时相同。 |
| `fill_vertical` | 按需要扩展对象的垂直大小，使其完全适应其容器。当 `clipOrientation` 是 `"vertical"` 时，不会进行裁剪，因为可绘制对象会填充垂直空间（除非可绘制对象级别为 0，此时它不可见）。 |
| `center_horizontal` | 将对象放在其容器的水平中心，不改变其大小。裁剪行为与重力为 `"center"` 时相同。 |
| `fill_horizontal` | 按需要扩展对象的水平大小，使其完全适应其容器。当 `clipOrientation` 是 `"horizontal"` 时，不会进行裁剪，因为可绘制对象会填充水平空间（除非可绘制对象级别为 0，此时它不可见）。 |
| `center` | 将对象放在其容器的水平和垂直轴中心，不改变其大小。当 `clipOrientation` 是 `"horizontal"` 时，在左边和右边裁剪。当 `clipOrientation` 是 `"vertical"` 时，在顶部和底部裁剪。 |
| `fill` | 按需要扩展对象的垂直大小，使其完全适应其容器。不会进行裁剪，因为可绘制对象会填充水平和垂直空间（除非可绘制对象级别为 0，此时它不可见）。 |
| `clip_vertical` | 可设置为让子元素的上边缘和/或下边缘裁剪至其容器边界的附加选项。裁剪基于垂直重力：顶部重力裁剪上边缘，底部重力裁剪下边缘，任一重力不会同时裁剪两边。 |
| `clip_horizontal` | 可设置为让子元素的左边和/或右边裁剪至其容器边界的附加选项。裁剪基于水平重力：左边重力裁剪右边缘，右边重力裁剪左边缘，任一重力不会同时裁剪两边。 |

示例：

XML 文件保存在 `res/drawable/clip.xml` 中：

```xml
<?xml version="1.0" encoding="utf-8"?>
<clip xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/android"
    android:clipOrientation="horizontal"
    android:gravity="left" />
```

以下布局 XML 会将裁剪可绘制对象应用到视图：

```xml
<ImageView
    android:id="@+id/image"
    android:background="@drawable/clip"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content" />
```

以下代码用于获取可绘制对象，并增加裁剪量以便逐渐显示图像：

```
ImageView imageview = (ImageView) findViewById(R.id.image);
ClipDrawable drawable = (ClipDrawable) imageview.getDrawable();
drawable.setLevel(drawable.getLevel() + 1000);
```

增大级别可减少裁剪量并慢慢显示图像。此处的级别为 7000：

> **注**：默认级别为 0，即完全裁剪，使图像不可见。当级别为 10,000 时，图像不会裁剪，而是完全可见。

另请参阅：

- `ClipDrawable`

# 缩放可绘制对象

在 XML 文件中定义的更改其他可绘制对象大小（根据其当前级别）的可绘制对象。

文件位置：

`res/drawable/filename.xml`
文件名用作资源 ID。

编译的资源数据类型：

指向 `ScaleDrawable` 的资源指针。

资源引用：

在 Java 中：`R.drawable.filename`
在 XML 中：`@[package:]drawable/filename`

语法：

```
<?xml version="1.0" encoding="utf-8"?>
<scale
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/drawable_resource"
    android:scaleGravity=["top" | "bottom" | "left" | "right" | "center_vertical" |
                          "fill_vertical" | "center_horizontal" | "fill_horizontal" |
                          "center" | "fill" | "clip_vertical" | "clip_horizontal"]
    android:scaleHeight="percentage"
    android:scaleWidth="percentage" />
```

元素：

`<scale>`

定义缩放可绘制对象。这必须是根元素。

属性：

`xmlns:android`

*字符串*。**必备。**定义 XML 命名空间，其必须是 `"http://schemas.android.com/apk/res/android"`。

`android:drawable`

*可绘制对象资源*。**必备**。引用可绘制对象资源。

`android:scaleGravity`

*关键字*。指定缩放后的重力位置。

必须是以下一个或多个（用 '|' 分隔）常量值：

| 值 | 说明 |
| --- | --- |
| `top` | 将对象放在其容器顶部，不改变其大小。 |
| `bottom` | 将对象放在其容器底部，不改变其大小。 |
| `left` | 将对象放在其容器左边缘，不改变其大小。这是默认值。 |
| `right` | 将对象放在其容器右边缘，不改变其大小。 |
| `center_vertical` | 将对象放在其容器的垂直中心，不改变其大小。 |
| `fill_vertical` | 按需要扩展对象的垂直大小，使其完全适应其容器。 |
| `center_horizontal` | 将对象放在其容器的水平中心，不改变其大小。 |
| `fill_horizontal` | 按需要扩展对象的水平大小，使其完全适应其容器。 |
| `center` | 将对象放在其容器的水平和垂直轴中心，不改变其大小。 |
| `fill` | 按需要扩展对象的垂直大小，使其完全适应其容器。 |
| `clip_vertical` | 可设置为让子元素的上边缘和/或下边缘裁剪至其容器边界的附加选项。裁剪基于垂直重力：顶部重力裁剪上边缘，底部重力裁剪下边缘，任一重力不会同时裁剪两边。 |
| `clip_horizontal` | 可设置为让子元素的左边和/或右边裁剪至其容器边界的附加选项。裁剪基于水平重力：左边重力裁剪右边缘，右边重力裁剪左边缘，任一重力不会同时裁剪两边。 |

`android:scaleHeight`

*百分比*。缩放高度，表示为可绘制对象边界的百分比。值的格式为 XX%。例如：100%、12.5% 等。

`android:scaleWidth`

*百分比*。缩放宽度，表示为可绘制对象边界的百分比。值的格式为 XX%。例如：100%、12.5% 等。

示例：

```
<?xml version="1.0" encoding="utf-8"?>
<scale xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/logo"
    android:scaleGravity="center_vertical|center_horizontal"
    android:scaleHeight="80%"
    android:scaleWidth="80%" />
```

另请参阅：

- `ScaleDrawable`

# 形状可绘制对象

这是在 XML 中定义的一般形状。

文件位置：

`res/drawable/`*`filename`*`.xml`
文件名用作资源 ID。

编译的资源数据类型：

指向 `GradientDrawable` 的资源指针。

资源引用：

在 Java 中：`R.drawable.`*`filename`*
在 XML 中：`@[`*`package`*`:]drawable/`*`filename`*

语法：

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape=["rectangle" | "oval" | "line" | "ring"] >
    <corners
        android:radius="integer"
        android:topLeftRadius="integer"
        android:topRightRadius="integer"
        android:bottomLeftRadius="integer"
        android:bottomRightRadius="integer" />
    <gradient
        android:angle="integer"
        android:centerX="float"
        android:centerY="float"
        android:centerColor="integer"
        android:endColor="color"
        android:gradientRadius="integer"
        android:startColor="color"
        android:type=["linear" | "radial" | "sweep"]
        android:useLevel=["true" | "false"] />
    <padding
        android:left="integer"
        android:top="integer"
        android:right="integer"
        android:bottom="integer" />
    <size
        android:width="integer"
        android:height="integer" />
    <solid
        android:color="color" />
    <stroke
        android:width="integer"
        android:color="color"
        android:dashWidth="integer"
        android:dashGap="integer" />
</shape>
```

元素：

`<shape>`

形状可绘制对象。这必须是根元素。

属性：

`xmlns:android`

*字符串*。**必备。**定义 XML 命名空间，其必须是 `"http://schemas.android.com/apk/res/android"`。

`android:shape`

*关键字*。定义形状的类型。有效值为：

| 值 | 描述 |
| --- | --- |
| `"rectangle"` | 填充包含视图的矩形。这是默认形状。 |
| `"oval"` | 适应包含视图尺寸的椭圆形状。 |
| `"line"` | 跨越包含视图宽度的水平线。此形状需要 `<stroke>` 元素定义线宽。 |
| `"ring"` | 环形。 |

仅当 `android:shape="ring"` 如下时才使用以下属性：

`android:innerRadius`

*尺寸*。环内部（中间的孔）的半径，以尺寸值或尺寸资源表示。

`android:innerRadiusRatio`

*浮点型*。环内部的半径，以环宽度的比率表示。例如，如果 `android:innerRadiusRatio="5"`，则内半径等于环宽度除以 5。此值被 `android:innerRadius` 覆盖。默认值为 9。

`android:thickness`

*尺寸*。环的厚度，以尺寸值或尺寸资源表示。

`android:thicknessRatio`

*浮点型*。环的厚度，表示为环宽度的比率。例如，如果 `android:thicknessRatio="2"`，则厚度等于环宽度除以 2。此值被 `android:innerRadius` 覆盖。默认值为 3。

`android:useLevel`

*布尔值*。如果这用作 `LevelListDrawable`，则此值为“true”。这通常应为“false”，否则形状不会显示。

`<corners>`

为形状产生圆角。仅当形状为矩形时适用。

属性：

`android:radius`

*尺寸*。所有角的半径，以尺寸值或尺寸资源表示。对于每个角，这会被以下属性覆盖。

`android:topLeftRadius`

*尺寸*。左上角的半径，以尺寸值或尺寸资源表示。

`android:topRightRadius`

*尺寸*。右上角的半径，以尺寸值或尺寸资源表示。

`android:bottomLeftRadius`

*尺寸*。左下角的半径，以尺寸值或尺寸资源表示。

`android:bottomRightRadius`

*尺寸*。右下角的半径，以尺寸值或尺寸资源表示。

> **注**：（最初）必须为每个角提供大于 1 的角半径，否则无法产生圆角。如果希望特定角不要倒圆角，解决方法是使用 `android:radius` 设置大于 1 的默认角半径，然后使用实际所需的值替换每个角，为不希望倒圆角的角提供零（“0dp”）。

`<gradient>`

指定形状的渐变颜色。

属性：

`android:angle`

*整型*。渐变的角度（度）。0 为从左到右，90 为从上到上。必须是 45 的倍数。默认值为 0。

`android:centerX`

*浮点型*。渐变中心的相对 X 轴位置 (0 - 1.0)。

`android:centerY`

*浮点型*。渐变中心的相对 Y 轴位置 (0 - 1.0)。

`android:centerColor`

*颜色*。起始颜色与结束颜色之间的可选颜色，以十六进制值或颜色资源表示。

`android:endColor`

*颜色*。结束颜色，表示为十六进制值或颜色资源。

`android:gradientRadius`

*浮点型*。渐变的半径。仅在 `android:type="radial"` 时适用。

`android:startColor`

*颜色*。起始颜色，表示为十六进制值或颜色资源。

`android:type`

*关键字*。要应用的渐变图案的类型。有效值为：

| 值 | 说明 |
| --- | --- |
| `"linear"` | 线性渐变。这是默认值。 |
| `"radial"` | 径向渐变。起始颜色为中心颜色。 |
| `"sweep"` | 流线型渐变。 |

`android:useLevel`

*布尔值*。如果这用作 `LevelListDrawable`，则此值为“true”。

`<padding>`

要应用到包含视图元素的内边距（这会填充视图内容的位置，而非形状）。

属性：

`android:left`

*尺寸*。左内边距，表示为尺寸值或尺寸资源

`android:top`

*尺寸*。上内边距，表示为尺寸值或尺寸资源

`android:right`

*尺寸*。右内边距，表示为尺寸值或尺寸资源

`android:bottom`

*尺寸*。下内边距，表示为尺寸值或尺寸资源

`<size>`

形状的大小。

属性：

`android:height`

*尺寸*。形状的高度，表示为尺寸值或尺寸资源

`android:width`

*尺寸*。形状的宽度，表示为尺寸值或尺寸资源

> **注**：默认情况下，形状按照此处定义的尺寸按比例缩放至容器视图的大小。在 `ImageView` 中使用形状时，可通过将 `android:scaleType` 设置为 `"center"` 来限制缩放。

`<solid>`

用于填充形状的纯色。

属性：

`android:color`

*颜色*。应用于形状的颜色，以十六进制值或颜色资源表示。

`<stroke>`

形状的笔划中线。

属性：

`android:width`

*尺寸*。线宽，以尺寸值或尺寸资源表示。

`android:color`

*颜色*。线的颜色，表示为十六进制值或颜色资源。

`android:dashGap`

*尺寸*。短划线的间距，以尺寸值或尺寸资源表示。仅在设置了 `android:dashWidth` 时有效。

`android:dashWidth`

*尺寸*。每个短划线的大小，以尺寸值或尺寸资源表示。仅在设置了 `android:dashGap` 时有效。

示例：

XML 文件保存在 `res/drawable/gradient_box.xml` 中：

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <gradient
        android:startColor="#FFFF0000"
        android:endColor="#80FF00FF"
        android:angle="45"/>
    <padding android:left="7dp"
        android:top="7dp"
        android:right="7dp"
        android:bottom="7dp" />
    <corners android:radius="8dp" />
</shape>
```

此布局 XML 会将形状可绘制对象应用到视图：

```
<TextView
    android:background="@drawable/gradient_box"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content" />
```

此应用代码将获取形状可绘制对象，并将其应用到视图：

```
Resources res = getResources();
Drawable shape = res. getDrawable(R.drawable.gradient_box);

TextView tv = (TextView)findViewByID(R.id.textview);
tv.setBackground(shape);
```

另请参阅：

- ShapeDrawable