**[Android XML Drawables](http://megadarja.blogspot.ru/2011/03/android-xml-drawables.html)**

В Android довольно продуманная система организации ресурсов. И особого внимания заслуживают ресурсы-изображения.

О самих картинках сказать можно мало что. Основная фича тут такая: можно адаптировать картинки для разных разрешений и размеров экранов и складывать в специализированные папки. Для различных разрешений экрана используются drawable-hdpi, drawable-mdpi и пр.. Аналогичное деление для размеров экрана: drawable-normal, drawable-large и т.д.. Можно даже одно с другим совмещать и создавать папки типа drawable-normal-hdpi. Изображения, которые должны оставаться неизменными вне зависимости от разрешения экрана, следует размещать в папке drawable-nodpi.

Но тема данной статьи несколько другая. Изображение (drawable) в Android-приложении — это не обязательно картинка. Бывают еще изображения, заданные с помощью XML. Их-то мы и рассмотрим подробнее. Будет много XML и мелких картинок.

В [документации](http://developer.android.com/guide/topics/resources/drawable-resource.html) приведен список и краткое описания типов изображений, которые можно использовать в приложении. В статье же описываются только следующие:

[Shape Drawable](http://megadarja.blogspot.ru/2011/03/android-xml-drawables.html#shape) (ShapeDrawable)

Геометрическая форма.

[Layer List](http://megadarja.blogspot.ru/2011/03/android-xml-drawables.html#layer) (LayerDrawable)

Изображение, состоящее из нескольких других изображений.

[State List](http://megadarja.blogspot.ru/2011/03/android-xml-drawables.html#states) (StateListDrawable)

Правило, согласно которому задаются изображения для различных состояний компонента.

**Shape Drawable**

Как уже было сказано ранее, изображение — это не всегда картинка. Можно создавать и собственные изображения на основе геометрических фигур.

Вот простой пример:

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

<selector

xmlns:android="http://schemas.android.com/apk/res/android">

<item>

<shape android:shape="rectangle">

<solid android:color="#0377BE"/>

<stroke android:width="2dip" android:color="#084B72" />

</shape>

</item>

</selector>

Здесь мы задали прямоугольник, его цвет и абрис. Если выведем это изображение в ImageView, получим следующее:



Типов фигур, которые мы можем задавать, всего 4:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Изображение** | **Разметка** | **Примечание** |
| **rectangle(Прямоугольник)** | Solid rectangle | **shape\_rect.xml**  <selector  xmlns:android="http://schemas.android.com/apk/res/android">  <item>  <shape android:shape="rectangle">  <solid android:color="#0377BE"/>  </shape>  </item>  </selector> | Атрибут android:shape здесь необязателен: rectangle — это значение по умолчанию. |
| **oval (Эллипс)** | Oval | **shape\_oval.xml**  <?xml version="1.0" encoding="utf-8"?>  <selector  xmlns:android="http://schemas.android.com/apk/res/android">  <item>  <shape android:shape="oval">  <solid android:color="#FCD366"/>  </shape>  </item>  </selector> |  |
| **ring (Кольцо)** | Ring | **shape\_ring.xml**  <?xml version="1.0" encoding="utf-8"?>  <selector  xmlns:android="http://schemas.android.com/apk/res/android">  <item>  <shape android:shape="ring" android:innerRadiusRatio="3"  android:thicknessRatio="5.333">  <solid android:color="#7DBE15"/>  </shape>  </item>  </selector> | Для кольца имеются дополнительные атрибуты:  innerRadius  Внутренний радиус  innerRadiusRatio  Отношение между внешним и внутренним радиусами  thickness  Толщина кольца (т.е. разница между внешним и внутренним радиусами)  thicknessRatio  Отношение ширины кольца к его толщине |
| **line(Горизонтальная линия)** | Line | **shape\_ring.xml**  <?xml version="1.0" encoding="utf-8"?>  <selector xmlns:android="http://schemas.android.com/apk/res/android">  <item>  <shape android:shape="line">  <stroke android:width="1dip" android:color="#F20107" />  </shape>  </item>  </selector> | Линия может быть только горизонтальной |

Рассмотрим дополнительные возможности рисования.

**Скругление углов**

Актуально для формы rectangle. Имеется специальный тег corners. Можно скруглить все углы:

|  |  |
| --- | --- |
| **rectangle\_rounded\_all.xml**  <?xml version="1.0" encoding="utf-8"?>  <selector  xmlns:android="http://schemas.android.com/apk/res/android">  <item>  <shape>  <solid android:color="#B902B0"/>  <corners android:radius="10.0dip" />  </shape>  </item>  </selector> | Углы скруглены одинаково |

А можно задать радиусы для всех углов отдельно:

|  |  |
| --- | --- |
| **rectangle\_rounded\_some.xml**  <?xml version="1.0" encoding="utf-8"?>  <selector  xmlns:android="http://schemas.android.com/apk/res/android">  <item>  <shape>  <solid android:color="#EC6118"/>  <corners android:bottomRightRadius="0.1dp"  android:bottomLeftRadius="7dip"  android:topLeftRadius="7dip"  android:topRightRadius="0.1dp"/>  </shape>  </item>  </selector> | Углы скруглены по-разному |

**Градиентная заливка**

Все любят градиенты. Градиенты — это хорошо и полезно. В нашем мире без них просто нельзя обойтись. И конечно, очень здорово, когда есть средство простого создания градиентов без использования дополнительных изображений.

Вы наверно заметили, что во всех примерах выше для задания сплошной заливки мы использовали тег solid. Несложно догадаться, что для тег для задания градиентов называется gradient.

Доступны градиенты следующих видов:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Изображение** | **Разметка** | **Примечание** |
| **linear** | Линейный градиент | **gradient\_linear.xml**  <?xml version="1.0" encoding="utf-8"?>  <selector  xmlns:android="http://schemas.android.com/apk/res/android">  <item>  <shape>  <gradient android:type="linear" android:endColor="#3C0000" android:startColor="#FF0202" android:angle="45.0" />  <corners android:radius="10.0dip" />  </shape>  </item>  </selector> | * android:type="linear" можно опустить, оно так и есть по умолчанию. * Атрибут android:angle используется только линейным градиентом. А еще он должен быть кратен 45. |
| **radial** | Радиальный градиент | **gradient\_radial.xml**  <?xml version="1.0" encoding="utf-8"?>  <selector  xmlns:android="http://schemas.android.com/apk/res/android">  <item>  <shape>  <gradient android:type="radial" android:endColor="#3C0000" android:startColor="#FF0202" android:gradientRadius="50"/>  <corners android:radius="10.0dip" />  </shape>  </item>  </selector> | * android:gradientRadius является обязательным для радиального градиента, а у остальных игнорируется. * Имеются также атрибуты android:centerX и android:centerY, в которых можно задавать относительное (0.0 – 1.0) расположение центра градиента. |
| **sweep** | Конический градиент градиент | **gradient\_sweep.xml**  <?xml version="1.0" encoding="utf-8"?>  <selector  xmlns:android="http://schemas.android.com/apk/res/android">  <item>  <shape>  <gradient android:type="sweep" android:endColor="#3C0000" android:startColor="#FF0202" />  <corners android:radius="10.0dip" />  </shape>  </item>  </selector> | * android:centerX и android:centerY также можно использовать. |

В общем, с градиентами все довольно просто: type, startColor и endColor. Также иногда оказывается полезным атрибут centerColor.

**Другие полезные свойства форм**

**stroke**

Где-то выше мы уже задавали абрис для формы. Тут все просто: тег stroke, у которого есть атрибуты width (толщина) и color (цвет). Собственно, хочется обозначить еще два атрибута — dashWidth и dashGap, с помощью которых можно сделать абрис не простой линией, а штриховой.

**padding**

Внутреннее поле. Соответственно, атрибуты top, bottom, left, right.

**Layer List**

С помощью XML-разметки можем задавать не только простые формы, но и их комбинации. Для этого служит класс LayerDrawable. Объяснять тут особенно нечего, просто приведу пару примеров:

**Пример 1: Кнопка с бликом**

Кнопка с бликом

Такую кнопку можно сотворить с помощью такого XML-файла:

**layers\_button.xml**

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

<layer-list xmlns:android="http://schemas.android.com/apk/res/android">

<item>

<shape>

<gradient android:startColor="#339999" android:endColor="#006666" android:angle="-90.0"/>

<corners android:radius="10.0dip" />

</shape>

</item>

<item android:bottom="20dip">

<shape>

<solid android:color="#88339999"/>

<corners android:bottomRightRadius="0.1dip"

android:bottomLeftRadius="0.1dip"

android:topLeftRadius="10dip"

android:topRightRadius="10dip"/>

</shape>

</item>

</layer-list>

**Пример 2: Марка**



Строится из следующего XML-файла:

**layers\_postmark.xml**

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

<layer-list xmlns:android="http://schemas.android.com/apk/res/android">

<item>

<shape>

<solid android:color="#ffffff"/>

<stroke android:width="5dip" android:color="#ffffff" android:dashWidth="4dp" android:dashGap="4dp" />

<corners android:radius="4dip" />

<padding android:left="5dip" android:top="5dip" android:right="5dip" android:bottom="5dip"/>

</shape>

</item>

<item>

<shape>

<solid android:color="#FFFFE1"/>

<stroke android:width="1dip" android:color="#4A3321"/>

<padding android:left="10dip" android:top="10dip" android:right="10dip" android:bottom="10dip"/>

</shape>

</item>

<item>

<bitmap android:src="@drawable/image"/>

</item>

</layer-list>

**State List**

У некоторых элементов управления есть некоторые состояния. Вроде: нажато, выбрано, зачекано и т.д.. В Android есть возможность задавать изображения для любого из таких состояний. Рассмотрим примеры.

**Пример 1**

Вот стандартная кнопка, выглядящая довольно уныло:

Стандартная кнопка

И вот, разработчик хочет, чтобы она выглядела как-нибудь эдак:

Красивая кнопка

Да еще и чтобы подсвечивалась при нажатии.

Сделать это просто. Добавляем в ресурсы изображения нормального (button\_up.png) и нажатого (button\_down.png) состояний кнопки. И создаем в папке drawables XML-файл со следующим содержанием:

**states\_button.xml**

<?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\_down" />

<item android:drawable="@drawable/button\_up" />

</selector>

Важный момент: в списке состояний нужно сначала указывать специализированные состояния, а последним должно следовать состояние по умолчанию (без атрибутов state-\* вообще).

Кстати говоря, ранее в статье не приводилось примера использования XML-изображения в стилизации элементов управления. Тут ничего сложного: с точки зрения Андроида это то же самое, что обычная картинка. В разметке можно обращаться к ней по имени (@drawable/*resource-name*). Вот так:

**main.xml**

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

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent" android:layout\_height="match\_parent">

<Button android:layout\_width="wrap\_content" android:layout\_height="wrap\_content"

android:background="@drawable/states\_button"

android:text="Custom button"

android:textSize="18dip"

/>

</RelativeLayout>

В результате получим означенный эффект.

Доступны следующие состояния со вполне очевидными названиями:

* state\_focused
* state\_window\_focused
* state\_enabled
* state\_checkable
* state\_checked
* state\_selected
* state\_active
* state\_pressed

Для списков есть также:

* state\_single
* state\_first
* state\_middle
* state\_last

**Пример 2**

Можно использовать уже упомянутые нами геометрические изображения:

**states\_shapes.xml**

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

<selector xmlns:android="http://schemas.android.com/apk/res/android">

<item android:state\_pressed="true">

<shape>

<gradient android:type="linear" android:endColor="#6699CC" android:startColor="#99CCFF" android:angle="90.0" />

<corners android:radius="5.0dip" />

<padding android:left="20dip" android:right="20dip" android:top="7dip" android:bottom="7dip"/>

</shape>

</item>

<item>

<shape>

<gradient android:type="linear" android:endColor="#336699" android:startColor="#99CCFF" android:angle="90.0" />

<corners android:radius="5.0dip" />

<padding android:left="20dip" android:right="20dip" android:top="7dip" android:bottom="7dip"/>

</shape>

</item>

</selector>

Результат будет таким:

Shape button

**Пример 3**

Похожим образом можно задавать стили для RadioButtom, CheckBox и т.п.. Но тут мало того, что можно отдельным ресурсом задавать фон, так отдельным же ресурсом можно задавать саму пиктограмму (атрибут button).

Вот, например, изображение

**states\_compound.xml**

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

<selector xmlns:android="http://schemas.android.com/apk/res/android">

<item android:state\_checked="true">

<shape android:shape="oval">

<gradient android:startColor="#00FF00" android:endColor="#00000000" android:type="radial" android:gradientRadius="7"/>

<size android:width="10dip" android:height="10dip" />

</shape>

</item>

<item>

<shape android:shape="oval">

<gradient android:startColor="#FF0000" android:endColor="#00000000" android:type="radial" android:gradientRadius="7"/>

<size android:width="10dip" android:height="10dip" />

</shape>

</item>

</selector>

Так — разметка:

**main.xml**

<CheckBox android:layout\_width="wrap\_content" android:layout\_height="wrap\_content"

android:button="@drawable/states\_compound"

android:text="Item 1"

android:paddingLeft="20dip"

/>

<CheckBox android:layout\_width="wrap\_content" android:layout\_height="wrap\_content"

android:button="@drawable/states\_compound"

android:checked="true"

android:text="Item 2"

android:paddingLeft="20dip"

/>

<CheckBox android:layout\_width="wrap\_content" android:layout\_height="wrap\_content"

android:button="@drawable/states\_compound"

android:text="Item 3"

android:paddingLeft="20dip"

/>

А таким будет результат:



**Заключение**

Итак, мы рассмотрели некоторые случаи использования XML drawables. Это далеко не все, что можно творить с ресурсами-изображениями, но наиболее распространенные задачи.

[Исходники](http://narod.ru/disk/7290348001/DrawablesDemo.zip.html)

**Ссылки**

* [Android Drawable XML Documentation](http://idunnolol.com/android/drawables.html#selector) — подробное описание структуры XML drawables
* [Android Selectors](http://android-journey.blogspot.com/2009/12/android-selectors.html) — статья про State Lists
* [Tutorial: change look of CheckBox](http://www.anddev.org/tutorial_change_look_of_checkbox-t4553.html) — тоже про State Lists