客製化SWITCH

## Styles.xml ##

<style name="BrandedSwitch" parent="AppTheme">

<item name="colorAccent">@color/icmBlue</item>

# ON 時, Track的顏色 #

<item name="android:colorControlActivated">@color/icmBlue</item>

# OFF時, Track的顏色#

<item name="android:colorForeground">@color/white</item>

</style>

## layout ##

<Switch

android:id="@+id/valveSwitch"

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:layout\_gravity="center"

android:layout\_weight="1"

android:text=""

android:theme="@style/BrandedSwitch"

android:textOn="@string/NormOpen"

android:textOff="@string/NormClose" />

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

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

<item android:drawable="@drawable/on" android:state\_checked="true"/> //the drawable to be displayed when view is checked.

<item android:drawable="@drawable/off" android:state\_checked="false"/>//the drawable to be displayed when view is not checked.

</selector>

第五章-开关按钮ToggleButton和Switch

public abstract class

CompoundButton

extends Button

implements Checkable

java.lang.Object

↳ android.view.View

↳ android.widget.TextView

↳ android.widget.Button

↳ android.widget.CompoundButton

Known Direct Subclasses

CheckBox, RadioButton, Switch, ToggleButton

由继承结构可以看出Switch和ToggleButton都继承自CompoundButton这一抽象类。因此它们属性和方法都是类似的，首先看一下ToggleButton的用法。

ToggleButton

常用的属性很简单就两个：

* android:textOff：按钮关闭时显示的文本
* android:textOn：按钮开启时显示的文本

开关监听为：setOnCheckedChangeListener，覆写onCheckedChanged(CompoundButton buttonView, boolean isChecked)方法，其中isChecked表示是否选中。

用过小米手电筒APP的用户都会对它的UI有印象，也就是开灯时，手机屏幕里的灯泡也会亮，关灯时手机屏幕里的灯泡则不会亮，下面我们就结合ToggleButton实现这一功能。 主布局文件（activity\_main.xml）

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

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

android:layout\_width="match\_parent"

android:orientation="vertical"

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

<ImageView

android:id="@+id/iv"

android:layout\_margin="20dp"

android:layout\_gravity="center"

android:src="@drawable/off"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" />

<ToggleButton

android:id="@+id/tb"

android:layout\_gravity="center"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textOff="关灯"

android:disabledAlpha="@android:integer/config\_longAnimTime"

android:textSize="25sp"

android:layout\_margin="20dp"

android:textOn="开灯" />

</LinearLayout>

设置了一个ImageView用于显示灯泡的开关，设置了一个ToggleButton当做开关，并设置了textOff和textOn属性。

MainActivity.java代码（MainActivity.java）

public class MainActivity extends AppCompatActivity {

private ToggleButton toggleButton;

private ImageView imageView;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

toggleButton=(ToggleButton)findViewById(R.id.tb);

imageView=(ImageView)findViewById(R.id.iv);

toggleButton.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {

@Override

public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {

imageView.setImageResource(isChecked?R.drawable.on:R.drawable.off);

}

});

}

}

这里设置了setOnCheckedChangeListener选择变化的监听，并覆写了其onCheckedChanged方法，根据参数isChecked判断要显示哪张图片。 运行实例如下：





选择关灯时灯泡显示的也是暗的，选择开灯，灯泡相应变亮。 有同学会说这样的开关按钮不够直观、不够美观，下面介绍一下如何自定义ToggleButton的外观。

首先准备两张图片：

这里写图片描述

这里写图片描述

在drawable文件下创建xml文件，代码如下：

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

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

<item android:drawable="@drawable/on" android:state\_checked="true"/>

<item android:drawable="@drawable/off" android:state\_checked="false"/>

</selector>

设置了在android:state\_checked为true的时候显示on图片，在android:state\_checked为false时显示off图片。 在styles.xml文件中添加代码：

<style name="MySwitch" parent="@android:style/Widget.CompoundButton">

<item name="android:button">@drawable/switch\_button</item>

</style>

布局文件代码如下：

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

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

android:layout\_width="match\_parent"

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

<ToggleButton

android:id="@+id/sw"

style="@style/MySwitch"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:background="#00000000"

android:checked="true"

android:textOff="@null"

android:textOn="@null" />

</LinearLayout>

设置了style属性，引入我们设置的style样式，并设置textOff和textOn为@null这样就不会在图片上显示文字了。运行项目实例如下：



这时可以看到ToggleButton的外观改变了。

Switch

这里外观实现了Switch的自定义样式，功能实现了动态控制WIFI的开关。首先看一下滑块外观（switch\_bg.xml）：

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

<item android:state\_enabled="true" android:drawable="@drawable/enable" />

<item android:state\_pressed="true" android:drawable="@drawable/press" />

<item android:state\_checked="true" android:drawable="@drawable/check\_on" />

</selector>

这里定义了三种状态下的图片，一种是state\_enable（可用状态）为true时，一种是state\_pressed（按下状态时）为true，一种是state\_checked（选择状态）为true。每种状态对应不同的背景图片。 对于滑动轨道的背景定义了两种不同的图片（track\_bg.xml）：

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

<item android:drawable="@drawable/track\_disable" android:state\_enabled="false" />

<item android:drawable="@drawable/track\_default" />

</selector>

一种是可用状态即默认状态下显示track\_default图片，一种是不可用状态下，显示track\_default图片。 主布局文件如下：

<LinearLayout

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

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical"

android:gravity="center">

<Button

android:id="@+id/btn"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginTop="16dp"

android:text="禁用Switch按钮"

android:layout\_marginBottom="15dp"

android:onClick="button\_click"/>

<Switch

android:id="@+id/sw"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textOff="OFF"

android:textOn="ON"

android:layout\_alignParentRight="true"

android:layout\_marginRight="20dp"

android:thumb="@drawable/switch\_bg"

android:track="@drawable/track\_bg"

android:layout\_marginBottom="15dp" />

</LinearLayout>

定义了一个按钮Button，设置了其onClick事件用于动态控制Switch是否可用。定义了一个Switch按钮，设置了其thumb即滑块外观和track即滑动轨道外观分别为上面定义的两个图片。 MainActivity.java代码如下：

public class MainActivity extends Activity {

Button button;

Switch sw;

boolean is\_enable = true;

WifiManager wifiManager;

AudioManager audioManager;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

button = (Button)findViewById(R.id.btn);

sw = (Switch)findViewById(R.id.sw);

sw.setChecked(true);

audioManager = (AudioManager) getSystemService(Context.AUDIO\_SERVICE);

sw.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {

@Override

public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {

if (buttonView.isChecked()) {

wifiManager = (WifiManager) getSystemService(Context.WIFI\_SERVICE);

wifiManager.setWifiEnabled(true);

} else {

wifiManager = (WifiManager) getSystemService(Context.WIFI\_SERVICE);

wifiManager.setWifiEnabled(false);

}

}

});

}

public void button\_click(View view){

if(is\_enable == true)

{

is\_enable = false;

button.setText("启用Switch按钮");

}

else{

is\_enable = true;

button.setText("禁用Switch按钮");

}

sw.setEnabled(is\_enable);

}

}

设置了布尔型的标识位is\_enable用于判断Switch的使能状态，并动态控制Button按钮上文本的显示。实现了setOnCheckedChangeListener监听，判断Switch的选择与否，根据isChecked参数进行判断是否选中。 对于开启和关闭WIFI，这里首先通过getSystemService方法并传入Context.WIFI\_SERVICE常量获取WifiManager对象，调用这个对象的setWifiEnable传入true或者false就可以控制WIFI的开启或关闭了。注意这里要配置相应的权限，AndroidManifest.xml代码如下：

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

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

package="demo.androidwar.com.switchdemo">

<application

android:allowBackup="true"

android:icon="@mipmap/ic\_launcher"

android:label="@string/app\_name"

android:supportsRtl="true"

android:theme="@style/AppTheme">

<activity android:name=".MainActivity">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

</application>

<uses-permission android:name="android.permission.CHANGE\_WIFI\_STATE"/>

</manifest>

配置了CHANGE\_WIFI\_STATE的权限。 运行项目实例如下：



第一个图片展示开启WIFI的状态，我们可以在手机状态栏中看到WIFI的状态图标，第二个图片展示关闭WIFI的状态，手机状态栏中已经看不到WIFI状态图标了，第三个图片是禁用Switch按钮的状态。

# 第四章-玩转单选和多选按钮

public abstract class

CompoundButton

extends Button

implements Checkable

java.lang.Object

↳ android.view.View

↳ android.widget.TextView

↳ android.widget.Button

↳ android.widget.CompoundButton

Known Direct Subclasses

CheckBox, RadioButton, Switch, ToggleButton

可以看出CheckBox和RadioButton同样都是CompoundButton抽象类的子类。其中CheckBox一般用于多选的情形，RadioButton一般用于单选的情形。注意，若想要RadioButton实现单选功能，必须用一个RadionGroup标签包裹所有的RadioButton。 这两个控件的使用都是比较简单的，下面通过一个实例来对这两个控件的方法和属性进行全面的学习。 主布局文件（activity\_main.xml）

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<TextView

android:layout\_margin="10dp"

android:layout\_width="match\_parent"

android:text="请选择您的性别："

android:textSize="18sp"

android:layout\_height="wrap\_content" />

<RadioGroup

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content">

<RadioButton

android:id="@+id/boy"

android:text="男"

android:checked="true"

android:layout\_margin="10dp"

android:textSize="18sp"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" />

<RadioButton

android:id="@+id/girl"

android:text="女"

android:layout\_margin="10dp"

android:textSize="18sp"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" />

</RadioGroup>

<TextView

android:layout\_margin="10dp"

android:layout\_width="match\_parent"

android:text="请选择您的爱好："

android:textSize="18sp"

android:layout\_height="wrap\_content" />

<CheckBox

android:id="@+id/cb\_climb"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_margin="10dp"

android:textSize="19sp"

android:text="爬山"

android:button="@drawable/checkbox\_bg"

android:checked="true" />

<CheckBox

android:id="@+id/cb\_football"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_margin="10dp"

android:textSize="19sp"

android:text="足球"

android:button="@drawable/checkbox\_bg"/>

<CheckBox

android:id="@+id/cb\_swim"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_margin="10dp"

android:textSize="19sp"

android:text="游泳"

android:button="@drawable/checkbox\_bg"

android:checked="true" />

<Button

android:id="@+id/btn\_ok"

android:onClick="commit"

android:text="确定"

android:gravity="center"

android:textSize="18sp"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content" />

<TextView

android:id="@+id/tv\_result"

android:layout\_width="match\_parent"

android:padding="10dp"

android:text="选择结果"

android:textSize="18sp"

android:layout\_height="wrap\_content" />

</LinearLayout>

这里对CheckBox的外观进行了自定义，设置了其button属性为："@drawable/checkbox\_bg"。下面看一下checkbox\_bg.xml:

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

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

<item android:drawable="@drawable/checked" android:state\_checked="true" />

<item android:drawable="@drawable/unchecked" android:state\_checked="false" />

</selector>

Selector标签进行包裹，主要用来设置控件背景，其中每一个item代表一个状态。设置了属性state\_checked为true或false时分别对应不同的图片。 除了state\_checked属性还有如下常用属性：  android:state\_selected控件是否被选中  android:state\_focused控件是否获得焦点  android:state\_pressed控件是否被点击  android:state\_enabled控件是否可用

MainActivity.java代码（MainActivity.java）

public class MainActivity extends Activity implements CompoundButton.OnCheckedChangeListener {

private CheckBox checkBoxClimb, checkBoxFootBall, checkBoxSwim;

private RadioButton radioButtonBoy, radioButtonGirl;

private String string = new String();

private Set<String> set = new HashSet<String>();

private TextView textView;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

initViews();

initText();

}

//不做选择直接按确定按钮时

private void initText() {

if (radioButtonGirl.isChecked()) {

string ="性别：女、";

} else {

string="性别：男、";

}

if (checkBoxClimb.isChecked()) {

set.add("爬山");

if (checkBoxSwim.isChecked()) {

set.add("游泳");

if (checkBoxFootBall.isChecked()) {

set.add("足球");

}

}

}

}

//初始化控件，设置事件监听

private void initViews() {

textView = (TextView) findViewById(R.id.tv\_result);

checkBoxClimb = (CheckBox) findViewById(R.id.cb\_climb);

checkBoxFootBall = (CheckBox) findViewById(R.id.cb\_football);

checkBoxSwim = (CheckBox) findViewById(R.id.cb\_swim);

radioButtonBoy = (RadioButton) findViewById(R.id.boy);

radioButtonGirl = (RadioButton) findViewById(R.id.girl);

checkBoxClimb.setOnCheckedChangeListener(this);

checkBoxFootBall.setOnCheckedChangeListener(this);

checkBoxSwim.setOnCheckedChangeListener(this);

radioButtonBoy.setOnCheckedChangeListener(this);

radioButtonGirl.setOnCheckedChangeListener(this);

}

@Override

public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {

switch (buttonView.getId()) {

case R.id.boy:

if (isChecked) {

string="性别：男、";

}

break;

case R.id.girl:

if (isChecked) {

string="性别：女、";

}

break;

case R.id.cb\_climb:

if (isChecked) {

set.add("爬山");

}else{

set.remove("爬山");

}

break;

case R.id.cb\_football:

if (isChecked) {

set.add("足球");

}else{

set.remove("足球");

}

break;

case R.id.cb\_swim:

if (isChecked) {

set.add("游泳");

}else {

set.remove("游泳");

}

break;

}

}

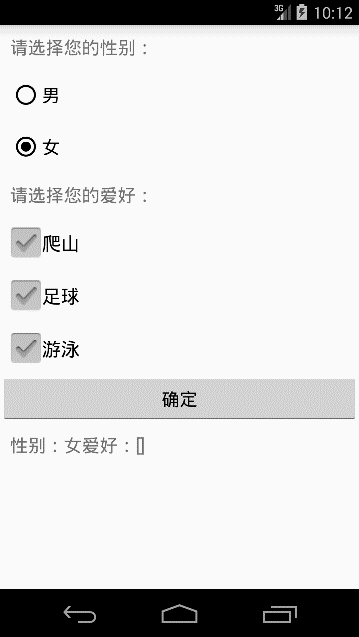
public void commit(View view) {

textView.setText(string.toString() + "爱好：" + set.toString());

}

}

Activity实现了CompoundButton.OnCheckedChangeListener接口，覆写了onCheckedChanged方法，有参数CompoundButton的对象buttonView结合其getId方法获取所有选择控件的ID，根据ID的不同区分不同的选项按钮。 在逻辑处理方面要注意，这里采用了Set集合介绍爱好中的字符串，Set集合不接收重复的元素，可以满足本实例的需求。其中initText方法用于初始化选择内容。 运行实例如下：





可以看出CheckBox的外观是我们添加的图片，选择完成后点击确定按钮即可输出选择的性别和爱好。

Android UI(Switch)详解

[switch](http://so.csdn.net/so/search/s.do?q=switch&t=blog) /

[android](http://so.csdn.net/so/search/s.do?q=android&t=blog) /

[UI布局](http://so.csdn.net/so/search/s.do?q=UI%E5%B8%83%E5%B1%80&t=blog)

18668

目录：  
    1.应用场景与概述  
    2.常用属性  
    3.简单使用  
    4.更改默认Switch的样式  
    5.自定义Switch  
      
1.应用场景与概述  
    Switch是在4.0以后推出的，所以要注意开发时的minsdk设置,google在API 21后也推出support v7 包下的SwitchCompa的Material Design   
开关控件，对低版本的有了更好的的支持。其实switch的应用场景和ToggleButton类似，多应用于两种状态的切换。  
      
2.常用属性    

[plain] [view plain](http://blog.csdn.net/qq_28057577/article/details/52261641) [copy](http://blog.csdn.net/qq_28057577/article/details/52261641)

android:typeface="normal":设置字体类型

android:track="":设置开关的轨迹图片

android:textOff="开"：设置开关checked的文字

android:textOn="关"：设置开关关闭时的文字

android:thumb="":设置开关的图片

android:switchMinWidth=""：开关最小宽度

android:switchPadding="":设置开关 与文字的空白距离

android:switchTextAppearance="":设置文本的风格

android:checked="":设置初始选中状态

android:splitTrack="true":是否设置一个间隙，让滑块与底部图片分隔（API 21及以上）

android:showText="true"：设置是否显示开关上的文字（API 21及以上）

3.简单使用  
    3.1)主布局

[html] [view plain](http://blog.csdn.net/qq_28057577/article/details/52261641) [copy](http://blog.csdn.net/qq_28057577/article/details/52261641)

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

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

    xmlns:tools="http://schemas.android.com/tools"

    android:layout\_width="match\_parent"

    android:layout\_height="match\_parent"

    xmlns:app="http://schemas.android.com/apk/res-auto"

    android:paddingBottom="@dimen/activity\_vertical\_margin"

    android:paddingLeft="@dimen/activity\_horizontal\_margin"

    android:paddingRight="@dimen/activity\_horizontal\_margin"

    android:paddingTop="@dimen/activity\_vertical\_margin"

    tools:context="com.example.aswitch.MainActivity">

    <!--

    android:typeface="normal":设置字体类型

    android:track="":设置开关的轨迹

    android:textOff="开"：设置开关checked的文字

    android:textOn="关"：设置开关关闭时的文字

    android:thumb="":设置开关的图片

    android:switchMinWidth=""：开关最小宽度

    android:switchPadding="":设置开关 与文字的空白距离

    android:switchTextAppearance="":设置文本的风格

    android:checked="":设置初始选中状态

    android:splitTrack="true":是否设置一个间隙，让滑块与底部图片分隔

    -->

    <TextView

        android:id="@+id/switch\_tv"

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content"

        android:text="switch:" />

    <Switch

        android:layout\_marginTop="10dp"

        android:layout\_below="@+id/switch\_tv"

        android:id="@+id/switch1"

        android:typeface="normal"

        android:textOff="开"

        android:textOn="关"

        android:switchMinWidth="40dp"

        android:switchPadding="10dp"

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content" />

    <TextView

        android:id="@+id/text"

        android:layout\_marginTop="10dp"

        android:layout\_below="@+id/switch1"

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content"

        android:text="Hello World!" />

    <TextView

        android:layout\_below="@+id/text"

        android:id="@+id/switch\_compat\_tv"

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content"

        android:text="switchCompat:" />

    <android.support.v7.widget.SwitchCompat

        android:layout\_marginTop="10dp"

        android:layout\_below="@+id/switch\_compat\_tv"

        android:id="@+id/switch\_compat"

        android:typeface="normal"

        android:switchMinWidth="40dp"

        android:switchPadding="10dp"

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content" />

    <TextView

        android:id="@+id/text1"

        android:layout\_marginTop="10dp"

        android:layout\_below="@+id/switch\_compat"

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content"

        android:text="Hello World!" />

</RelativeLayout>

    3.2)主布局java类

[java] [view plain](http://blog.csdn.net/qq_28057577/article/details/52261641) [copy](http://blog.csdn.net/qq_28057577/article/details/52261641)

package com.example.aswitch;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.support.v7.widget.SwitchCompat;

import android.widget.CompoundButton;

import android.widget.Switch;

import android.widget.TextView;

public class MainActivity extends AppCompatActivity implements CompoundButton.OnCheckedChangeListener{

    private Switch aSwitch;

    private SwitchCompat aSwitchCompat;

    private TextView text1,text2,switchText,switchCompatText;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_main);

        //实例化

        aSwitch = (Switch) findViewById(R.id.switch1);

        aSwitchCompat = (SwitchCompat) findViewById(R.id.switch\_compat);

        text1 = (TextView) findViewById(R.id.text);

        text2 = (TextView) findViewById(R.id.text1);

        //设置Switch事件监听

        aSwitch.setOnCheckedChangeListener(this);

        aSwitchCompat.setOnCheckedChangeListener(this);

    }

    /\*

    继承监听器的接口并实现onCheckedChanged方法

    \* \*/

    @Override

    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {

        switch (buttonView.getId()){

            case R.id.switch1:

                if(isChecked){

                    text1.setText("开");

                }else {

                    text1.setText("关");

                }

                break;

            case R.id.switch\_compat:

                if(isChecked){

                    text2.setText("开");

                }else {

                    text2.setText("关");

                }

                break;

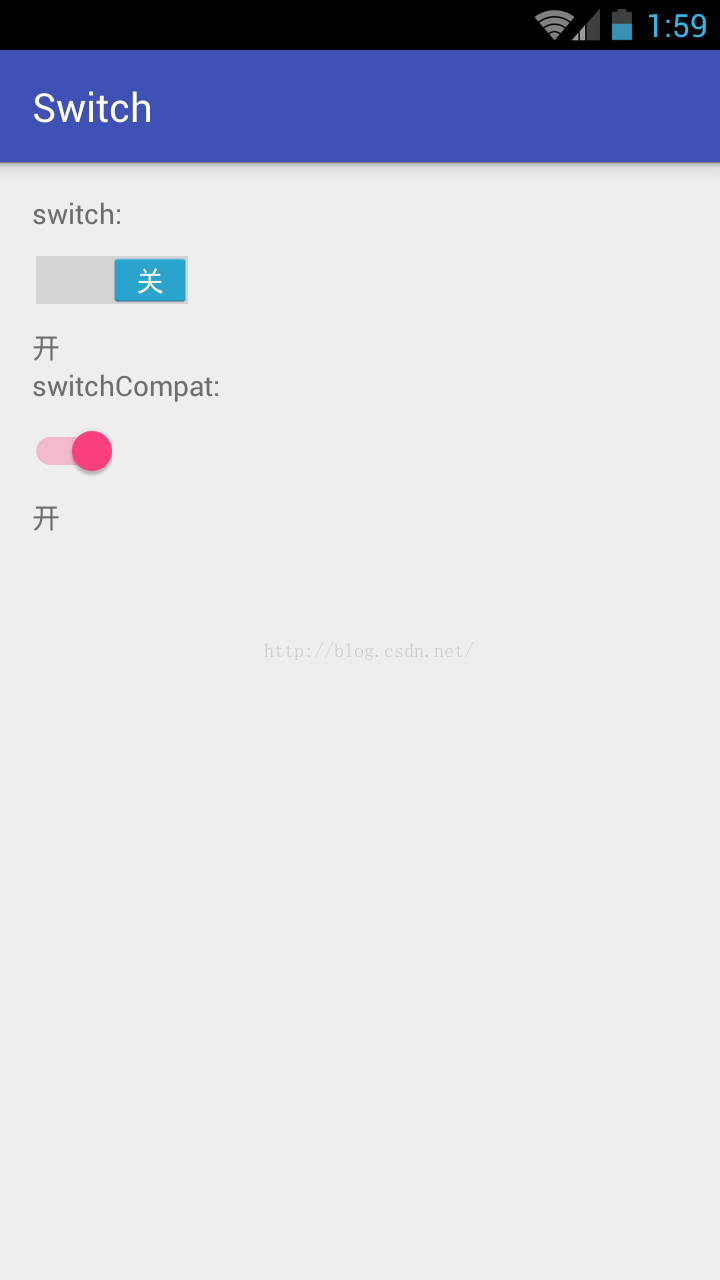
            default:

                break;

        }

    }

}

    3.3)截图效果  
      
4.更改默认Switch的样式  
    4.1)在styles.xml中自定义style

[html] [view plain](http://blog.csdn.net/qq_28057577/article/details/52261641) [copy](http://blog.csdn.net/qq_28057577/article/details/52261641)

<!--自定义switch的按钮和轨迹颜色theme-->

<style name="mySwitch" parent="Theme.AppCompat.Light">

    <!-- switch 打开时的按钮的颜色 轨迹颜色默认为30%（看效果就明白30%是怎么回事了）这个颜色 -->

    <item name="colorControlActivated">@android:color/holo\_green\_dark</item>

    <!--  switch关闭时的按钮的颜色 -->

    <item name="colorSwitchThumbNormal">@color/colorAccent</item>

    <!-- switch关闭时的轨迹的颜色 30%这个颜色 -->

    <item name="android:colorForeground">@color/colorPrimaryDark</item>

</style>

    4.1)在布局文件中通过android:theme="@style/mySwitch"设置

[html] [view plain](http://blog.csdn.net/qq_28057577/article/details/52261641) [copy](http://blog.csdn.net/qq_28057577/article/details/52261641)

<android.support.v7.widget.SwitchCompat

     android:layout\_marginTop="10dp"

     android:layout\_below="@+id/switch\_compat\_tv"

     android:id="@+id/switch\_compat"

     android:typeface="normal"

     android:theme="@style/mySwitch"

     android:switchMinWidth="40dp"

     android:switchPadding="10dp"

     android:layout\_width="wrap\_content"

     android:layout\_height="wrap\_content" />

5.自定义Switch      
  
    5.1)导入资源图片thumb.png ,thumb\_on.png ,track\_nomal.png ,track\_on.png ,track\_press.png  
      
    5.2)实现thumb\_selector.xml

[html] [view plain](http://blog.csdn.net/qq_28057577/article/details/52261641) [copy](http://blog.csdn.net/qq_28057577/article/details/52261641)

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

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

    <!--选中时的滑块图片-->

<item android:drawable="@drawable/thumb\_on" android:state\_checked="true"/>

    <!--正常情况滑块图片-->

<item android:drawable="@drawable/thumb"/>

</selector>

    5.3)实现track\_selector.xml

[html] [view plain](http://blog.csdn.net/qq_28057577/article/details/52261641) [copy](http://blog.csdn.net/qq_28057577/article/details/52261641)

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

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

    <!--打开时switch轨迹图片-->

    <item android:state\_pressed="true"  android:drawable="@drawable/track\_on" />

    <!--按压时switch轨迹图片-->

    <item android:state\_checked="true"  android:drawable="@drawable/track\_press" />

    <!--正常状态switch轨迹图片-->

    <item                               android:drawable="@drawable/track\_nomal" />

    </selector>

    5.4)主布局actiity\_second.xml

[html] [view plain](http://blog.csdn.net/qq_28057577/article/details/52261641) [copy](http://blog.csdn.net/qq_28057577/article/details/52261641)

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

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

    xmlns:tools="http://schemas.android.com/tools"

    android:layout\_width="match\_parent"

    android:layout\_height="match\_parent"

    xmlns:app="http://schemas.android.com/apk/res-auto"

    android:paddingBottom="@dimen/activity\_vertical\_margin"

    android:paddingLeft="@dimen/activity\_horizontal\_margin"

    android:paddingRight="@dimen/activity\_horizontal\_margin"

    android:paddingTop="@dimen/activity\_vertical\_margin"

    tools:context="com.example.aswitch.SecondActivity">

    <TextView

        android:id="@+id/CustomSwitchCompat\_tv"

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content"

        android:text="CustomSwitchCompat:" />

    <android.support.v7.widget.SwitchCompat

        android:layout\_marginTop="10dp"

        android:layout\_below="@+id/CustomSwitchCompat\_tv"

        android:id="@+id/CustomSwitchCompat"

        android:layout\_width="wrap\_content"

        android:minWidth="40dp"

        android:minHeight="20dp"

        android:layout\_height="wrap\_content" />

    <TextView

        android:id="@+id/custom\_result"

        android:layout\_marginTop="10dp"

        android:layout\_below="@+id/CustomSwitchCompat"

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content"

        android:text="Hello World!" />

</RelativeLayout>

    5.5)主布局java类SecondActivity.java

[java] [view plain](http://blog.csdn.net/qq_28057577/article/details/52261641) [copy](http://blog.csdn.net/qq_28057577/article/details/52261641)

package com.example.aswitch;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.support.v7.widget.SwitchCompat;

import android.widget.CompoundButton;

import android.widget.Switch;

import android.widget.TextView;

public class SecondActivity extends AppCompatActivity implements CompoundButton.OnCheckedChangeListener{

    private SwitchCompat customSwitchCompat;

    private TextView custom\_result,CustomSwitchCompat\_tv;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_second);

        //实例化

        customSwitchCompat = (SwitchCompat) findViewById(R.id.CustomSwitchCompat);

        custom\_result = (TextView) findViewById(R.id.custom\_result);

        //设置自定义的thumb和track

       customSwitchCompat.setThumbResource(R.drawable.thumb\_selector);

        customSwitchCompat.setTrackResource(R.drawable.track\_selector);

        //设置Switch事件监听

        customSwitchCompat.setOnCheckedChangeListener(this);

    }

    /\*

    继承监听器的接口并实现onCheckedChanged方法

    \* \*/

    @Override

    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {

                if(isChecked){

                    custom\_result.setText("开");

                }else {

                    custom\_result.setText("关");

                }

    }

}

    ps:其实自定义的途径还可以通过shape的绘制和java代码绘制，在这里就不详细说了  
      
    参考：<http://blog.csdn.net/zhyh1986/article/details/45406391>

自：<http://blog.csdn.net/u012585142/article/details/50756872>   
今天发现需要使用控件switch，但是需要自定义switch的样式，通过查阅资料，发现可以通过定义switch的thumb和track的图片来达到自定义switch样式的目的.现把相关的步骤记录下来，以备查阅。   
   
其中1为所需样式，2为系统自定义样式   
1、创建thumb（即来回滑动的滑动块）利用XML来定义一个drawable文件，命名为switch\_thumb

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

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

android:shape="rectangle">

<size

android:width="25dp"

android:height="25dp">

</size>

<!-- 填充的颜色：这里设置滑动块为白色 -->

<solid android:color="@color/white" />

<!--<solid android:color="@color/theme\_color" />-->

\*\*\*\*<!-- 边框的颜色 ：因为需要显示一点点背景色，所以设置了一个透明边框 -->\*\*

<stroke

android:width="2dp"

android:color="@android:color/transparent" />\*\*

<!--<corners android:radius="2dip" />-->

<corners android:radius="2dip" />

</shape>

2、track属性又该如何设置呢？   
问题在于打开switch与关闭switch，switch的下面的滑道应该是不同的颜色，我们想到了可以用drawable的selector标签，利用switch的不同状态，来加载不同的drawable文件。我们查看官方switch文档，发现其有   
void setChecked(boolean checked) Changes the checked state of this button.   
这样一个方法，所以我们可以利用其state\_checked状态，依据不同的状态来选择加载不同的drawable文件   
1）：首先，我们定义一个drawable文件：switch\_track\_on.xml

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

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

android:shape="rectangle">

<solid

android:color="@color/colorPrimary">

</solid>

<corners android:radius="2dip" />

</shape>

2）：再定义一个drawable文件：switch\_track\_off.xml

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

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

android:shape="rectangle">

<!--<size-->

<!--android:width="60dp"-->

<!--android:height="35dp">-->

<!--</size>-->

<solid

android:color="@android:color/darker\_gray">

</solid>

<corners android:radius="2dip" />

</shape>

3）：然后定义drawable文件：switch\_track.xml

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

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

<item

android:state\_checked="true"

android:drawable="@drawable/switch\_track\_on"></item>

<item

android:state\_checked="false"

android:drawable="@drawable/switch\_track\_off"></item>

</selector>

最后在布局文件

<Switch

android:id="@+id/switch\_autoupdate"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:thumb="@drawable/switch\_thumb"

android:track="@drawable/switch\_track"/>

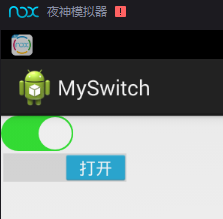
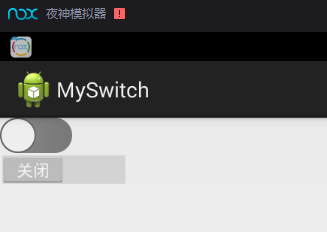
Android Switch控件修改样式

Android中自带的Switch控件在很多时候总觉得和整体系统风格不符，很多时候，自定义Switch是一种方法。

但其实不用这么麻烦，安卓自带的Switch通过修改一些属性，也可以达到和自定义Switch差不多的一个效果。

个人感觉，Switch的属性设置和其他控件还是有挺大区别的。因此，写下此文，方便有需要的同学参考。

先上效果图：



以上便是修改后效果 与 原生Switch的效果对比。代码在文章底部给出

实现方式：

**1.底部滑动条，在开关打开状态为绿色，开关关闭状态为灰色**

在 res/drawable 文件夹下面，写两个滑动条的底图 ，通过一个选择器selector进行控制。

gray\_track.xml ：非打开状态，灰色的底图

**[html]** [view plain](http://blog.csdn.net/qq_34763699/article/details/54954394) [copy](http://blog.csdn.net/qq_34763699/article/details/54954394)

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

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

    android:shape="rectangle" **>**

    <!-- 高度30   此处设置宽度无效-->

**<size** android:height="30dp"**/>**

    <!-- 圆角弧度 15 -->

**<corners** android:radius="15dp"**/>**

    <!-- 变化率  定义从左到右的颜色不变 -->

**<gradient**

        android:endColor="#888888"

        android:startColor="#888888" **/>**

**</shape>**

green\_track.xml：打开状态下，绿色的底图。

**[html]** [view plain](http://blog.csdn.net/qq_34763699/article/details/54954394) [copy](http://blog.csdn.net/qq_34763699/article/details/54954394)

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

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

    <!-- 高度40 -->

**<size** android:height="30dp"**/>**

    <!-- 圆角弧度 20 -->

**<corners** android:radius="15dp"**/>**

    <!-- 变化率 -->

**<gradient**

        android:endColor="#33da33"

        android:startColor="#33da33" **/>**

**</shape>**

选择器 track.xml   用于控制Switch不同状态下，滑动条的底图

**[html]** [view plain](http://blog.csdn.net/qq_34763699/article/details/54954394) [copy](http://blog.csdn.net/qq_34763699/article/details/54954394)

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

<!-- 底层下滑条的样式选择器，可控制Switch在不同状态下，底下下滑条的颜色 -->

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

**<item** android:state\_checked="true"  android:drawable="@drawable/green\_track" **/>**

**<item**                               android:drawable="@drawable/gray\_track" **/>**

**</selector>**

**2. 滑动按钮：底色我用的接近白色的淡灰色，打开时，边上的一圈线条为灰色，关闭时，边上的一圈线条为绿色**

实现方式和底部滑动一致

gray\_thumb.xml  :关闭状态，按钮边上一圈颜色为深灰色

**[html]** [view plain](http://blog.csdn.net/qq_34763699/article/details/54954394) [copy](http://blog.csdn.net/qq_34763699/article/details/54954394)

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

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

    android:shape="rectangle" **>**

    <!-- 高度40 -->

**<size** android:height="40dp" android:width="40dp"**/>**

    <!-- 圆角弧度 20 -->

**<corners** android:radius="20dp"**/>**

    <!-- 变化率 -->

**<gradient**

        android:endColor="#eeeeee"

        android:startColor="#eeeeee" **/>**

**<stroke** android:width="1dp"

        android:color="#666666"**/>**

**</shape>**

green\_thumb.xml : 打开状态，按钮边上一圈的颜色为绿色

**[html]** [view plain](http://blog.csdn.net/qq_34763699/article/details/54954394) [copy](http://blog.csdn.net/qq_34763699/article/details/54954394)

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

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

    android:shape="rectangle" **>**

    <!-- 高度40 -->

**<size** android:height="40dp" android:width="40dp"**/>**

    <!-- 圆角弧度 20 -->

**<corners** android:radius="20dp"**/>**

    <!-- 变化率 -->

**<gradient**

        android:endColor="#eeeeee"

        android:startColor="#eeeeee" **/>**

**<stroke** android:width="1dp"

        android:color="#33da33"**/>**

**</shape>**

选择器 thumb.xml   用于控制Switch不同状态下，按钮的显示状态

**[html]** [view plain](http://blog.csdn.net/qq_34763699/article/details/54954394) [copy](http://blog.csdn.net/qq_34763699/article/details/54954394)

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

<!-- 按钮的选择器，可以设置按钮在不同状态下的时候，按钮不同的颜色 -->

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

**<item** android:state\_checked="true"  android:drawable="@drawable/green\_thumb" **/>**

**<item**                               android:drawable="@drawable/gray\_thumb" **/>**

**</selector>**

**3. 将以上选择器设置给Switch，就好了**

界面  activity\_main.xml

**[html]** [view plain](http://blog.csdn.net/qq_34763699/article/details/54954394) [copy](http://blog.csdn.net/qq_34763699/article/details/54954394)

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

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

    android:layout\_width="match\_parent"

    android:layout\_height="match\_parent"

    android:orientation="vertical" **>**

**<Switch**

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content"

        android:switchMinWidth="20dp"

        android:textOn="  "

        android:textOff="  "

        android:thumb="@drawable/thumb"

        android:track="@drawable/track" **/>**

    <!-- 用于对比使用的不设置任何属性的Switch -->

**<Switch**

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content"

**/>**

**</LinearLayout>**

**4.高度，宽度的设置**

细心的同学会发现，修改  android:layout\_width  ， android:layout\_height  这两个属性，并不会实际修改Switch的大小

设置大了，边上会出现空白部分，设置小了，Switch显示不全。

实际设置高度方法：

上面定义滑动条和按钮底图的地方相信大家都注意到，   <size android:height="30dp"/>  这行代码，

修改  green\_track.xml,gray\_track.xml  中的高度，即可修改高度（修改green\_thumb.xml  gray\_thumb.xml  中的高度貌似无效）。

实际修改宽度的方法：

（1）修改滑动按钮的宽度：滑动按钮的宽度和按钮上的文字有关，

想要按钮变长，在按钮显示的文字上添加几个空字符串即可，想要按钮变短的话，减少按钮上显示的字即可（修改按钮上字体大小也可以试试）

Switch的属性

        android:textOn="  "  
        android:textOff="  "

（2）修改按钮  打开，关闭  两种状态之间滑动距离（貌似小到一定程度，再改小就无效了）

Switch的属性

android:switchMinWidth="20dp"

通过以上的设置，相信能满足大部分实际使用的需求了，希望对大家有帮助。

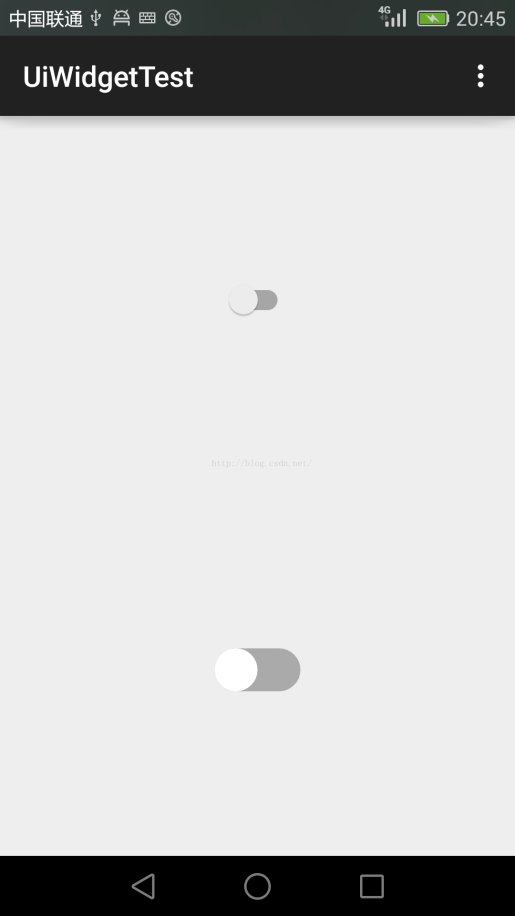
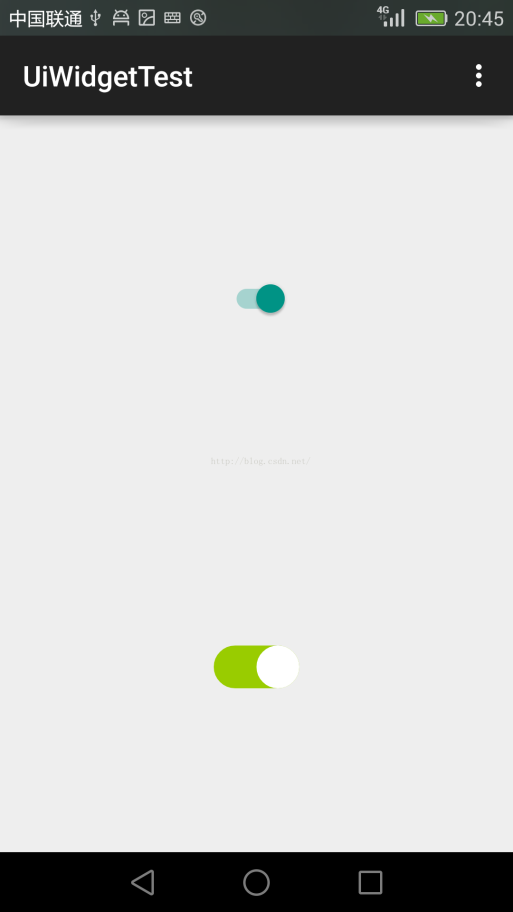
相信源代码才是大家最关注的吧，哈哈，我也是！  
**下载地址**：   http://download.csdn.net/detail/qq\_34763699/9751234

百度网盘：http://pan.baidu.com/s/1slp1CZV

Android UI之switch的thumb与track属性定制自己的switch

利用官方的switch控件的thumb与track属性，可以自定义自己的switch，实例如下：

上面的switch是官方默认的形状，下面的是通过设定自定义的thumb与track来实现的效果。



那如何来实现呢？

1.thumb比较好实现，利用XML来定义一个drawable文件，命名为switch\_thumb

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

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

android:shape="oval">

<size

android:width="30dp"

android:height="30dp">

</size>

<solid

android:color="@android:color/white">

</solid>

</shape>

thumb属性指的是：switch上面滑动的滑块，也就是上图中的白色圆形滑块

2、track属性又该如何设置呢？

问题在于打开switch与关闭switch，switch的下面的滑道应该是不同的颜色，我们想到了可以用drawable的selector标签，利用switch的不同状态，来加载不同的drawable文件。我们查看官方switch文档，发现其有

|  |  |
| --- | --- |
| void | setChecked(boolean checked)  Changes the checked state of this button. |

这样一个方法，所以我们可以利用其state\_checked状态，依据不同的状态来选择加载不同的drawable文件

1）：首先，我们定义一个drawable文件：switch\_track\_on.xml

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

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

android:shape="rectangle">

<solid

android:color="@android:color/holo\_green\_light">

</solid>

<corners

android:radius="32dp">

</corners>

</shape>

           2）：再定义一个drawable文件：switch\_track\_off.xml

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

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

android:shape="rectangle">

<solid

android:color="@android:color/darker\_gray">

</solid>

<corners

android:radius="30dp">

</corners>

</shape>

3）：然后定义drawable文件：switch\_track.xml

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

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

<item

android:state\_checked="true"

android:drawable="@drawable/switch\_track\_on"></item>

<item

android:state\_checked="false"

android:drawable="@drawable/switch\_track\_off"></item>

</selector>

4）：最后在布局文件中

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

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

tools:context=".MainActivity">

<Switch

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:layout\_weight="1"/>

<Switch

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:thumb="@drawable/switch\_thum"

android:track="@drawable/switch\_track"

android:layout\_weight="1"/>

</LinearLayout>

[Android中自定义switch控件样式](http://blog.csdn.net/csdnzouqi/article/details/75314703)

switch控件是有自带的开关样式的，需要使用自定义的话需要在xml文件中设置以下属性：

android:thumb="@null"

android:track="@null"

然后设置background属性为自己定义的selector，如下：

android:background="@drawable/switch\_plan\_notice\_selector"

selector资源文件如下：

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

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

<item android:state\_checked="true" android:drawable="@drawable/switch\_open"/>

<item android:state\_checked="false" android:drawable="@drawable/switch\_close"/>

</selector>

简单说，switch控件选中时候，设置一个选中的图片；未选中的时候，设置一个未选中的图片。

效果图展示如下：

选中状态



未选中状态

这是最简单的自定义样式，xml布局里设置控件的属性即可。

漂亮简单的Android 自定义 Switch 控件!

android原生的Switch控件可谓是极丑的~作为一个注重UI的android开发者~我很少会使用android自己的原生控件，想要做出一些比较好看的界面的话一般都会自己重写一些控件，比如说Switch控件，先上一个预览

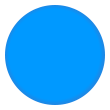
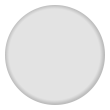
Switch

看起来比较酷炫的渐变和切换效果，实则只是对android动画的很少一部分的利用，下面我来介绍一下怎么实现这个控件。

思路

分解开来可以看做是在一个View的中间画出一条黑线，然后再用一个ImageView配上可以用动画形式改变图片的资源，自然而然的就想到了TransitionDrawable这个类，这个类主要对应以transition为根节点的drawable资源，然后再加上位置移动的动画，就可以很轻松的实现这个控件了~~

资源准备

首先准备两张图片，下面的图片可以直接复制回去   
   
分别取名为 switch\_button\_normal 和 switch\_button\_clicked   
接下来将这两个图片拷入drawable文件加下，并在文件夹下创建switch\_button\_drawable.xml文件，内容如下

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

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

<item android:drawable="@drawable/switch\_button\_normal" />

<item android:drawable="@drawable/switch\_button\_clicked" />

</transition>

第一个item对应的图片资源为默认显示的图片，当用TransitionDrawable类改变图片显示的时候，第二个item对应的图片会以透明度渐变的形式显示和隐藏

书写控件

建立MySwitchButton类继承自RelativeLayout   
首先定义一些需要用到的变量

private ImageView imgv\_handler = null; //用于显示Switch按钮

private Paint pat\_line = null; //用于画出Switch按钮底下的线

private boolean statue = false; //用于标示当前SwitchButton所处的状态，false为未激活状态，true为激活状态

private ObjectAnimator slideLeftAnimator = null; //按钮向左滑动的动画

private ObjectAnimator slideRightAnimator = null; //按钮向右滑动的动画

private TransitionDrawable transitionDrawable = null; //按钮切换图片资源的动画

接着在构造函数中初始化一部分变量

//初始化按钮

imgv\_handler = new ImageView(context);

imgv\_handler.setImageDrawable(getResources().getDrawable(R.drawable.switch\_button\_drawable));

this.addView(imgv\_handler);

//用八位色值将背景设为透明

this.setBackgroundColor(Color.parseColor("#00fafafa"));

pat\_line = new Paint();

pat\_line.setColor(Color.parseColor("#333333"));

pat\_line.setStyle(Style.STROKE);

//因为setStrokeWidth只能设置像素数，所以用这个方法将3dip转化为对应的像素数

pat\_line.setStrokeWidth((int) TypedValue.applyDimension(TypedValue.COMPLEX\_UNIT\_DIP, 3, getResources().getDisplayMetrics()));

//获取图片切换动画待后面使用

transitionDrawable= (TransitionDrawable) imgv\_handler.getDrawable();

因为下面初始化的变量需要用到组件自身高度宽度等参数，而在onMeasure函数执行时，系统已经给组件分配了对应的宽高，所以在onMeasure函数中初始化

@Override

protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {

//设置按钮图片对应的参数

LayoutParams params = new LayoutParams(getMeasuredHeight(), getMeasuredHeight());

params.addRule(ALIGN\_PARENT\_LEFT, TRUE);

imgv\_handler.setLayoutParams(params);

//设置右滑动动画

slideRightAnimator = ObjectAnimator.ofFloat(imgv\_handler, "translationX", 0, getMeasuredWidth() - getMeasuredHeight());

slideRightAnimator.setInterpolator(new DecelerateInterpolator()); //此处为动画插值器，用于设置动画曲线，不懂的同学百度一下~~~

slideRightAnimator.setDuration(300);

//设置做滑动动画

slideLeftAnimator = ObjectAnimator.ofFloat(imgv\_handler, "translationX", getMeasuredWidth() - getMeasuredHeight(), 0);

slideLeftAnimator.setInterpolator(new DecelerateInterpolator());

slideLeftAnimator.setDuration(300);

super.onMeasure(widthMeasureSpec, heightMeasureSpec);

}

画出SwitchButton的线

protected void onDraw(Canvas canvas) {

canvas.drawColor(Color.parseColor("#00fafafa"));

int startX = getMeasuredHeight() / 2;

int startY = getMeasuredHeight() / 2;

int endX = getMeasuredWidth() - getMeasuredHeight() / 2;

int endY = getMeasuredHeight() / 2;

canvas.drawLine(startX, startY, endX, endY, pat\_line);

super.onDraw(canvas);

}

到此，初始化工作变完成了，下面要做的就是对应的点击事件的操作了。

首先在构造函数中覆盖SwitchButton对应的点击事件，并让MySwitchButton类继承OnClickListener的接口

this.setOnClickListener(this);

接着在onClick函数中写相应的点击处理事件

@Override

public void onClick(View v) {

if (!statue) {

slideRightAnimator.start();

transitionDrawable.startTransition(300); //将资源从item1转化为item2

} else {

slideLeftAnimator.start();

transitionDrawable.reverseTransition(300); //将资源从item2转化为item1

}

statue = !statue;

}

到此为止这个组件已经可以实现其效果了，但是要是想要真正的投入使用，同学们可以自己加入一些自己需要的代码，比如getStatus(), 点击之后获取状态的接口等等，都很简单~这里就不一一介绍了~~

使用

直接在layout文件中写入下面的代码即可

<com.unique.nearfriend.MyView.MySwitchButton

android:layout\_width="60dp"

android:layout\_height="25dp"

android:layout\_alignParentRight="true"

android:layout\_centerVertical="true" />

很简单的一个组件~分享给大家~希望可以对你们有帮助