宝贵建议请发送至：wangzhenyang@itcast.cn

黑马程序员 itheima.com -编程，始于黑马

Android 课程同步笔记

Alpha 0.01 版

By 阳哥

Android 手机卫士-02

1.自定义对话框（★★★）

当我们第一次点击手机防盗的时候弹出一个对话框，提示我们输入密码并再次输入密码进行确定。密码设置好以后当我们再次点击手机防盗，那么弹出一个对话框，提示输入密码。我们将用户输入的密码保存到 SharedPreference 里面，这里我们需要做一个自定义对话框。对框框效果如下图所示。

该对话框的布局文件如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="300dp"
    android:gravity="center"
    android:layout_height="wrap_content"
    android:background="#ffffff"
    android:orientation="vertical" >

    <TextView
        android:gravity="center"
        android:layout_width="match_parent"
        android:layout_height="50dp"
```

```
        android:background="#66ff6600"
        android:textSize="20sp"
        android:text="设置密码" />

    <EditText
        android:id="@+id/et_password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textPassword"
        android:hint="请输入密码" />

    <EditText
        android:id="@+id/et_password_confirm"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textPassword"
        android:hint="请再次输入密码" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:orientation="horizontal" >

        <Button
            android:id="@+id/bn_cancel"
            android:layout_width="140dp"
            android:layout_height="wrap_content"
            android:text="取消" />

        <Button
            android:id="@+id/bn_ok"
            android:layout_width="140dp"
            android:layout_height="wrap_content"
            android:text="确定" />
    </LinearLayout>

</LinearLayout>
```

当点击手机防盗后判断密码是否设置的代码逻辑如下：

```
private void showPwdDialog() {
        //获取 SharedPreferences 对象
        sp = getSharedPreferences("config", MODE_PRIVATE);
        //获取密码
        String password = sp.getString("password", "");
        boolean isEmpty = TextUtils.isEmpty(password);
        //如果密码为空，进入设置密码对话框界面
        if (isEmpty) {
            showSetUpPwdDialog();
        } else {
            //如果不为空，进入输入密码对话框界面
            showInputPwdDialog();
        }
    }
```

第一次密码还未设置的情况下，进入 showSetUpPwdDialog 方法，方法清单如下：

```
private void showSetUpPwdDialog() {
        //创建一个 Builder 对象
        Builder builder = new Builder(this);
        //打气筒填充一个布局
        View view = View.inflate(this, R.layout.dialog_setup_password, null);
        //创建一个 AlertDialog 对象
        show = builder.create();
        //给对话框设置 View 对象
        show.setView(view, 0, 0, 0, 0);
        //显示对话框
        show.show();
        et_password = (EditText) view.findViewById(R.id.et_password);
        et_password_confirm = (EditText)
view.findViewById(R.id.et_password_confirm);
        bn_ok = (Button) view.findViewById(R.id.bn_ok);
        bn_cancel = (Button) view.findViewById(R.id.bn_cancel);
        bindCancel(bn_cancel);
        bn_ok.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                String password = et_password.getText().toString();
                String password_confirm = et_password_confirm.getText().toString();
                if (TextUtils.isEmpty(password)) {
                    Toast.makeText(HomeActivity.this, "密码不能为空",
```

```
                sp = getSharedPreferences("config", MODE_PRIVATE);
                //编辑对象
                Editor editor = sp.edit();
                //设置 password
                editor.putString("password", password);
                //提交
                editor.commit();
                //对话框消失
                show.dismiss();
                //进入手机防盗界面
                entryLostFind();
            } else {
                Toast.makeText(HomeActivity.this, "两次输入的密码不一致",
Toast.LENGTH_SHORT).show();
            }
        }
    });
}
```

2.MD5 加密（★★）

在上节中我们将用户的密码保存在 SharedPreferences 中，这样是很不安全的，如果通过 root 权限查看到了对应的 xml 文件，那么用户的密码也就泄露了。处于安全考虑，密码通常不会直接保存，通常做法是对密码进行 md5 加密，甚至是多次反复加密。

MD5 介绍：

MD5 的作用是对一段信息(message)生成信息摘要(message-digest)，该摘要对该信息具有 唯一性,可以作为数字签名。用于验证文件的有效性(是否有丢失或损坏的数据),对用户 密码的加密，在哈希函数中计算散列值。输入一个任意长度的字节串，生成一个 128 位的整数。由于算法的某些不可逆特征，在加密应用上有较好的安全性。并且，MD5 算法的使用不需要支付任何版权费用。

我们在工程中创建 com.itheima.mobileSafe.utils 包，该包用于存放各种工具类，我们新建一个 MD5Utils 类，在该类中实现 md5 的加密。代码清单如下：

```
//将字符串进行 md5 加密
    public static String encode(String password) {

        MessageDigest digest;
        try {
            //通过加密工具，并创建一个 md5 算法加密对象
            digest = MessageDigest.getInstance("md5");
            //将密码的字节码加密成字节数组
            byte[] result = digest.digest(password.getBytes());
            StringBuffer buffer = new StringBuffer();
            for (byte b : result) {
                //对每一个字节跟 255 进行和操作，转换为 int 类型
                int number = b & 0xff;
                //将 int 类型转换为 16 进制字符串类型
                String numberStr = Integer.toHexString(number);
                //如果 16 进制的长度只有一位（0~9）只有一位，在前面补 0
                if (numberStr.length() == 1) {
                    buffer.append("0");
                }
                //添加 16 进制字符串
                buffer.append(numberStr);
            }
            return buffer.toString();
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
            return "";
        }
    }
```

3.手机防盗设置向导的第一个界面（★★★）

在第一次输入密码后，进入向导页面。通过向导页面可以绑定 sim 卡、设置安全号码、开启防盗功能等。

向导页面效果图如下：

进入向导页先判断是否已经设置过，我们将设置过用一个 Boolean 值保存在 SharedPreferences 中。在向导设置完成以后我们提供一个按钮让用户可以重新进入设置向导。这些逻辑判断在 LostFindActivity 中实现。

```
public class LostFindActivity extends Activity {
    private SharedPreferences sp;
    private TextView tv_saveNumber;
    private TextView tv_isProtectting;
    private ImageView iv_isProtectting;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //获取 sp 对象
        sp = getSharedPreferences("config", MODE_PRIVATE);
        //是否已经设置
        boolean configed = sp.getBoolean("configed", false);
        if (configed) {//进入手机防盗主页
            setContentView(R.layout.lostfind_activity);
        }else {//进入向导页面
            Intent intent = new Intent(this, Setup1Activity.class);
            startActivity(intent);
            //关闭当前 Activity
            finish();
            return;
        }
```

```
        tv_saveNumber = (TextView) findViewById(R.id.tv_saveNumber);
        tv_isProtectting = (TextView) findViewById(R.id.tv_isProtectting);
        iv_isProtectting = (ImageView) findViewById(R.id.iv_isProtectting);
        boolean protectting = sp.getBoolean("protectting", false);
        if (protectting) {
            tv_isProtectting.setText("防盗功能已经开启");
            iv_isProtectting.setImageResource(R.drawable.lock);
        }else {
            tv_isProtectting.setText("防盗功能还未开启");
            iv_isProtectting.setImageResource(R.drawable.unlock);
        }
        String saveNumber = sp.getString("saveNumber", null);
        if(!TextUtils.isEmpty(saveNumber)){
            tv_saveNumber.setText(saveNumber);
        }else {
            tv_saveNumber.setText("");
        }
    }
    //重新设置向导
    public void reEntrySetting(View view){
        Intent intent = new Intent(this, Setup1Activity.class);
        startActivity(intent);
        finish();
    }
}
```

首先进入的是 Setup1 界面。activity_setup1.xml 是 Setup1Activity 的布局文件，清单如下：

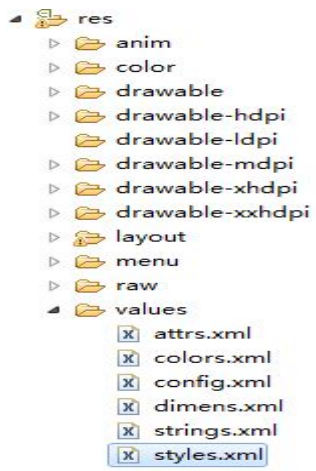
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="match_parent"
        android:layout_height="55dp"
        android:background="#8866ff00"
        android:gravity="center"
        android:text="1 欢迎使用手机向导"
        android:textColor="#000000"
        android:textSize="20sp" />
```

```
    <TextView
        style="@style/text_content_style"
        android:text="您的手机防盗卫士：" />

    <TextView
        style="@style/text_content_style"
        android:drawableLeft="@android:drawable/star_big_on"
        android:text="sim 卡变更报警" />

    <TextView
        style="@style/text_content_style"
        android:drawableLeft="@android:drawable/star_big_on"
        android:text="GPS 追踪" />

    <TextView
        style="@style/text_content_style"
        android:drawableLeft="@android:drawable/star_big_on"
        android:text="远程数据销毁" />

    <TextView
        style="@style/text_content_style"
        android:drawableLeft="@android:drawable/star_big_on"
        android:text="远程锁屏" />

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:orientation="horizontal" >

        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@android:drawable/presence_online" />

        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@android:drawable/presence_invisible" />

        <ImageView
            android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:src="@android:drawable/presence_invisible" />

        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@android:drawable/presence_invisible" />
    </LinearLayout>

    <RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" >

        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerInParent="true"
            android:src="@drawable/setup1" />

        <Button
            android:id="@+id/button1"
            style="@style/button_next" />
    </RelativeLayout>

</LinearLayout>
```

在该布局文件中用到了自定义样式知识点。

3.1 自定义样式



在工程中 res/values 目录下的 styles.xml 中添加如下样式，用于设置字体。

```
<style name="text_content_style" parent="AppBaseTheme">
        <item name="android:layout_width">wrap_content</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:textColor">#000000</item>
        <item name="android:textSize">18sp</item>
        <item name="android:gravity">center_vertical</item>
        <item name="android:layout_marginTop">8dp</item>
        <item name="android:layout_marginLeft">10dp</item>
    </style>
```

样式一样的可以通过 parent 属性实现“继承”。

4.自定义按钮状态背景（★★★）

在 Android 源码中，本人的地址为：

D:\Android_SDK_windows\adt-bundle-windows-x86-20131030\sdk\platforms\android-18\data\res\values

目录下，打开 styles.xml 文件，找到系统定义的 Button 按钮样式，清单如下：

```
<style name="Widget.Button">
        <item name="android:background">@android:drawable/btn_default</item>
        <item name="android:focusable">true</item>
        <item name="android:clickable">true</item>
        <item
name="android:textAppearance">?android:attr/textAppearanceSmallInverse</item>
        <item name="android:textColor">@android:color/primary_text_light</item>
        <item name="android:gravity">center_vertical|center_horizontal</item>
    </style>
```

我们可以参考上面的样式来自定义我们的按钮样式。Button 的自定义样式是通过

style="@style/button_next" 属性指定的。其中 button_next 是定义在 drawable 文件夹下的一个样式文件。

我们在 res 目录下新建一个 drawable 目录，如果已经存在可以直接使用，然后创建一个 xml 文件，button.xml。

Button.xml 布局清单如下：

在该布局文件中有三个 item 标签，android:state_pressed 代表点击后状态，android:state_focused 代

表获取焦掉时状态，最后一个 item 是默认状态。

```
<?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/function_greenbutton_pressed" /> <!-- pressed
-->
    <item android:state_focused="true"
          android:drawable="@drawable/function_greenbutton_pressed" /> <!-- focused
-->
    <item android:drawable="@drawable/function_greenbutton_normal" /> <!-- default
-->
</selector>
```

5.shape 形状资源（★★★）

通过 shape 形状资源，我们可以自定义控件的形状和颜色。如下图红色框住的控件就是我们自定义的按钮形状。

该控件在布局文件中其实就是 TextView 加上我们自定义的背景样式。该 TextView 属性如下：

```
<TextView
            android:background="@drawable/shape_selected"
            android:clickable="true"
            android:onClick="reEntrySetting"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_marginLeft="8dp"
            android:layout_marginRight="8dp"
            android:text="重新进入设置向导"
            android:textSize="16sp" />
```

在上面的属性中 android:background="@drawable/shape_selected"，其中 shape_selected 是我们在 drawable 目录下创建的 shape_selected.xml 文件名，该文件的代码清单如下：

```
<?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/gradient_box_press" /> <!-- pressed -->
    <item android:state_focused="true"
          android:drawable="@drawable/gradient_box_press" /> <!-- focused -->
    <item android:drawable="@drawable/gradient_box" /> <!-- default -->
</selector>
```

上面 xml 文件中的 gradient_box 属性值是我们在 drawable 目录下的 gradient_box.xml 文件，该文件的代码清单如下：该文件就是 shape 文件。

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" >
    <!-- 梯度，颜色的渐变效果 -->
    <gradient
        android:angle="45"
        android:endColor="#FF0000"
        android:startColor="#0000ff" />
    <!-- 间距 -->
    <padding
        android:bottom="7dp"
        android:left="7dp"
        android:right="7dp"
```

```
        android:top="7dp" />
    <!-- 圆角 -->
    <corners android:radius="8dp" />
</shape>
```

6.设置向导页面的切换动画（★★）

四个向导页面页面直接的切换动画默认使用的是系统的切换效果，当然我们也可以自定义切换效果。这里 Activity 之间的切换效果我们用 Activity 类的 overridePendingTransition 方法。

方法 overridePendingTransition(R.anim.*tran_in*, R.anim.*tran_out*);第一、第二个参数是 Activity 的进入和退出动画布局文件。此类布局文件在 res 目录下的 anim 目录中定义，如果没有 anim 目录那么首先创建 anim 目录。tran_in.xml 文件清单如下：

```
<?xml version="1.0" encoding="utf-8"?>
<translate xmlns:android="http://schemas.android.com/apk/res/android"
    android:fromYDelta="-100%p" android:toYDelta="0%p" android:duration="500">
</translate>
```

tran_out.xml 文件清单如下：

```
<?xml version="1.0" encoding="utf-8"?>
<translate xmlns:android="http://schemas.android.com/apk/res/android"
    android:fromYDelta="0%p" android:toYDelta="100%p" android:duration="500">
</translate>
```

android:fromYDelta="-100%p" android:toYDelta="0%p"两个属性代表当前 Activity 从屏幕上方（完全不可见）沿 Y 轴方向落满整个屏幕。android:fromYDelta="0%p" android:toYDelta="100%p"两个属性代表从当前屏幕往下完全滑出屏幕。

知识拓展：

有些手机上无法出现动画：是因为把播放动画的功能关闭了，为什么关闭呢，是为什么省点；

Android手机省电的几个技巧：

1.不要用动态的壁纸；

2.屏幕亮度调低一些；

3.3G网络关闭使用，2G网络，当然是不需要上网的情况下；

4.关闭手机执行动画；

5.经常杀一下后台的应用；

7.屏幕的滑动切换和抽取父类（★★★）

为了增强用户体验，我们可以让用户通过手势来切换 Activity。由于 4 个设置向导 Activity 都需要进行手势识别，因此 我们可以抽取出一个 BaseSetupActivity 类，在该类中实现手势的识别逻辑，然后凡是需要支持手势识别的 Activity 只需要继承该类即可。

```
public abstract class  BaseSetupActivity extends Activity {
    //声明一个手势识别器
    protected GestureDetector detector;
    protected SharedPreferences sp;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        sp = getSharedPreferences("config", MODE_PRIVATE);
        /**
         * 实例化一个手势识别器
         */
        detector = new GestureDetector(this, new
GestureDetector.SimpleOnGestureListener(){
            /**
             * 覆写 onFling 方法，代表滑动事件
             * velocityX 代表 X 轴方向的速度
             * velocityY 代表 Y 轴方向的速度
             * e1 代表滑动时第一次按下事件
```

```
            * e2 代表当前滑动时事件
            */
           @Override
           public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX,
float velocityY) {
               if (Math.abs(velocityY)<150) {
                   Toast.makeText(BaseSetupActivity.this, "大哥您的速度太慢啦，速度
="+Math.abs(velocityY), Toast.LENGTH_SHORT).show();
                   return true;
               }
               //如果在 Y 轴正方向的滑动距离大于 100pix 则调用 showNext 方法
               if (e2.getRawY()-e1.getRawY()>100) {
                   showNext();
                   return true;
               //如果在 Y 轴负方向的滑动距离大于 100pix 则调用 showPrevious 方法
               }else if (e2.getRawY()-e1.getRawY()<-100) {
                   showPrevious();
                   return true;
               }
               return false;
           }
       });
   }
   /**
    * 覆写父类的 onTouchEvent 方法，并把该 event 传递给 detector
    */
   @Override
   public boolean onTouchEvent(MotionEvent event) {
       detector.onTouchEvent(event);
       return super.onTouchEvent(event);
   }
   //声明抽象方法，让子类实现下一步业务逻辑
   public abstract void showNext();
   //声明抽象方法，让子类实现上一步业务逻辑
   public abstract void showPrevious();
}
```

8.绑定 sim 卡（★★★）

手机防盗最核心的就是绑定 sim 卡，之所以不绑定手机卡是因为有些 SIM 卡里没有写入手机号码，比如移动的卡

和联通的卡等等。绑定 SIM 卡的原理很简单，首先通过 getSystemService(*TELEPHONY_SERVICE*)，获取 TelephoneManager 对象，然后通过该对象的 getSimSerialNumber 方法获取 SIM 卡的序列号。获取 SIM 卡序列号后我们将该序列号保存在 SharedPreferences 中即可。Setup2Activity 主要实现了上述逻辑。代码清单如下：

```
public class Setup2Activity extends BaseSetupActivity {
    private SettingItemView bindSIM;
    private TelephonyManager tm;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_setup2);
        //获取 TelephonyManager 对象
        tm = (TelephonyManager) getSystemService(TELEPHONY_SERVICE);
        bindSIM = (SettingItemView) findViewById(R.id.bind_sim);
        //初始化界面时从 sp 对象中获取 sim 卡值
        String sim = sp.getString("sim", null);
        //如果 sim 卡已经存在，则说明已经绑定
        if (!TextUtils.isEmpty(sim)) {
            bindSIM.setChecked(true);
        }
        bindSIM.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                bindSIM.setChecked(!bindSIM.isChecked());
                Editor editor = sp.edit();
                if (bindSIM.isChecked()) {
                    //通过 tm 获取 sim 卡序列号
                    editor.putString("sim", tm.getSimSerialNumber());
                    editor.commit();
                    Toast.makeText(Setup2Activity.this, "SIM 卡已经绑定: " +
tm.getSimSerialNumber(), Toast.LENGTH_SHORT).show();
                } else {
                    editor.remove("sim");
                    editor.commit();
                    Toast.makeText(Setup2Activity.this, "SIM 卡解除绑定！",
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
```

```
    public void next(View view) {
        String sim = sp.getString("sim", null);
        if (TextUtils.isEmpty(sim)) {
            Toast.makeText(this, "SIM 卡信息还未绑定", Toast.LENGTH_SHORT).show();
            return;
        }
        Intent intent = new Intent(this, Setup3Activity.class);
        startActivity(intent);
        finish();
        overridePendingTransition(R.anim.tran_in, R.anim.tran_out);
    }

    public void previous(View view) {
        Intent intent = new Intent(this, Setup1Activity.class);
        startActivity(intent);
        finish();
        overridePendingTransition(R.anim.tran_pre_in, R.anim.tran_pre_out);
    }

    @Override
    public void showNext() {
        next(null);
    }

    @Override
    public void showPrevious() {
        previous(null);
    }
}
```

注意：因为在 Activity 中涉及到对 SIM 信息的获取，因此需要我们加上权限：

```
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
```

Setup2Activity 对应的布局文件 activity_setup2.xml 清单如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:itheima="http://schemas.android.com/apk/res/com.itheima.mobileSafe"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
    android:orientation="vertical" >
    <TextView
        android:layout_width="match_parent"
        android:layout_height="55dp"
        android:background="#8866ff00"
        android:gravity="center"
        android:text="2 手机卡绑定"
        android:textColor="#000000"
        android:textSize="20sp" />

    <TextView
        style="@style/text_content_style"
        android:text="通过绑定SIM卡：\n 下次重启手机如果发现SIM卡变化\n 就会发送报警短
信" />

    <com.itheima.mobileSafe.ui.SettingItemView
        android:id="@+id/bind_sim"
        android:layout_width="match_parent"
        android:layout_height="60dp"
        itheima:desc_off="sim卡没有绑定"
        itheima:desc_on="sim卡已经绑定"
        itheima:title="点击绑定sim卡" />

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:orientation="horizontal" >

        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@android:drawable/presence_invisible" />

        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@android:drawable/presence_online" />

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

```
        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@android:drawable/presence_invisible" />
    </LinearLayout>

    <RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" >

        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerInParent="true"
            android:src="@drawable/bind" />

        <Button
            android:id="@+id/button1"
            style="@style/button_next" />

        <Button
            android:id="@+id/button2"
            style="@style/button_previous" />
    </RelativeLayout>

</LinearLayout>
```

至此，本文档完！

2014 年 12 月 30 日 星期二 17:02:31
北京市海淀区中关村软件园国际软件大厦