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-01

1.Splash 界面（★★★）

1.1 Splash 界面的作用

1、用来展现产品的 Logo；

2、应用程序初始化的操作；

3、检查应用程序的版本；

4、检查当前应用程序是否合法注册；

1.2 Splash 界面布局

```
<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"
    android:id="@+id/rl_root"
    android:background="@drawable/luncher_bg" >

    <TextView
        android:id="@+id/tv_splash_version"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:shadowColor="#ff0000"
        android:shadowDx="1"
        android:shadowDy="1"
        android:shadowRadius="1"
        android:text="版本1.0"
        android:textColor="#000000"
        android:textSize="20sp" />
```

```
    <ProgressBar
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/tv_splash_version"
        android:layout_centerHorizontal="true" />

    <TextView
        android:text="下载进度"
        android:id="@+id/tv_download_progress"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/tv_splash_version"
        android:layout_centerHorizontal="true"
        />

</RelativeLayout>
```

1.3 动态获取应用程序版本

在 SplashActivity 中的 onCreate 方法中调用 getVersion 方法，用于动态显示当前应用的版本信息。

```
private String getVersion(){
      try {
         //通过 PackageManager 获取安装包信息
         PackageInfo packageInfo =
getPackageManager().getPackageInfo(getPackageName(),0);
         //返回版本信息
         return packageInfo.versionName;
      } catch (NameNotFoundException e) {
         return "";
      }
   }
```

1.4 链接服务器查看版本更新信息

在程序启动的时候会链接服务器获取最新版本信息，然后将自己的版本跟从服务器获取的版本信息进行对比，如果不一致（有最新版本）则提示用户更新。

注意：1、这里我们使用到了网络，因此要在清单文件中添加 android.permission.INTERNET 权限。2、因为访问网络是“耗时”过程，因此我们把检查版本更新业务要放在子线程中进行。

```
private void checkUpdate(){
       new Thread(new Runnable() {
          @Override
          public void run() {
             Message message = Message.obtain();
             try {
                URL url = new URL(getString(R.string.serverurl));
                HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
                connection.setConnectTimeout(2000);
                connection.setReadTimeout(2000);
                connection.setRequestMethod("GET");
                connection.connect();
                int code = connection.getResponseCode();
                if (code==200) {
                   InputStream inputStream = connection.getInputStream();
                   String string = StreamUtil.stream2String(inputStream);
                   System.out.println("请求结果: "+string);
                   JSONObject json = new JSONObject(string);
                   version = (String) json.get("version");
                   downloadurl = (String)json.get("downloadurl");
                   description = (String)json.get("description");
                   if (getVersion().equals(version)) {
                      message.what = ENTER_HOME;
                   }else{
                      message.what = SHOW_UPDATE;
                   }
                }else {
                   message.what = SERVER_ERROR;
                   message.obj = "获取更新失败: "+code;
                }
             } catch (Exception e) {
                System.out.println("获取更新失败: "+e);
                message.what = NET_ERROR;
                message.obj = "获取更新失败: "+e;
             }finally{
                handler.sendMessage(message);
             }
          }
```

```
        }).start();
    }
```

1.5 给 Splash 界面添加启动动画

在 onCreate 方法中，给 Splash 界面添加启动画面。这里添加的是渐变动画，透明度从 0 到 1 渐变，时长 500 毫秒。

```
AlphaAnimation alphaAnimation = new AlphaAnimation(0.f, 1.f);
RelativeLayout relativeLayout = (RelativeLayout) findViewById(R.id.rl_root);
alphaAnimation.setDuration(500);
relativeLayout.startAnimation(alphaAnimation);
```

1.6 升级应用程序

当检查到有最新版本的时候，我们会提示用户是否升级，如果用户选择了升级那么我们将在线进行升级。升级代码如下。在下载应用的时候我们使用了第三方工具 FinalHttp。

```
private void showUpdateDialog(){
      AlertDialog.Builder builder = new AlertDialog.Builder(this);
      builder.setTitle("发现新版本");
      builder.setMessage("已经发布新版本了，希望您能及时升级。");
      builder.setNeutralButton("升级", new OnClickListener() {

         @Override
         public void onClick(DialogInterface dialog, int which) {
            if
(Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)) {
               //升级
               FinalHttp http = new FinalHttp();
               http.configTimeout(1000);
               http.download(downloadurl,
Environment.getExternalStorageDirectory()+"/mobileSage2.0apk", new
AjaxCallBack<File>() {
```

```
@Override
public void onFailure(Throwable t, int errorNo, String strMsg) {
                        super.onFailure(t, errorNo, strMsg);
                        Toast.makeText(SplashActivity.this, "下载失败。",
Toast.LENGTH_SHORT).show();
                        enterHome();
                    }

                    @Override
                    public void onLoading(long count, long current) {
                        tv_download_progress.setVisibility(View.VISIBLE);
                        int progress = (int)(current*100/count);
                        tv_download_progress.setText("下载进度: "+progress+"%");
                    }

                    @Override
                    public void onStart() {
                        super.onStart();
                    }

                    @Override
                    public void onSuccess(File t) {
                        installAPK(t);
                    }
                });

            }else{
                Toast.makeText(SplashActivity.this, "sd卡不可用",
Toast.LENGTH_SHORT).show();
            }
        }
    });
    builder.setPositiveButton("暂不升级", new OnClickListener() {

        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.dismiss();
            enterHome();
        }
    });
    builder.setOnCancelListener(new OnCancelListener() {
```

```
            @Override
            public void onCancel(DialogInterface dialog) {
                enterHome();
            }
        });
        builder.show();
    }
    private void installAPK(File file) {
        Intent intent = new Intent();
        intent.setAction("android.intent.action.VIEW");
        intent.setDataAndType(Uri.fromFile(file),
"application/vnd.android.package-archive");
        startActivity(intent);
    }
```

注意：安装 APK 是通过一个隐式意图实现的。

1.7 两种上下文的区别

对话框是 Activity 的一部分，对话框是挂载在 Activity 上面的。如果 Activity 不存在，对话框就不能被创建。

Activity 实际上是应用程序 context 上下文的一个子集。

getApplicationContext();生命周期长，只要应用还存活它就存在；this 生命周期短，只要 Activity 不存在了，系统就会回收；其中：`getBaseContext(),getApplication(),`getApplicationContext(); 都不能放在 AlertDialog 做上下文；getApplicationContext（ ） 使用场景是比如频繁需要操作的数据库。

推荐用法:Activity.this。

2.应用程序主界面（★★★）

应用程序的主界面采用 9 宫格的形式，通过 GridView 控件来实现。

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="功能列表"
        android:textColor="#000000"
        android:textSize="20sp" />
    <com.itheima.mobileSafe.ui.FocusedTextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="欢迎使用黑马程序员私人订制版的手机安全卫士，制作人：王震阳，联系方式：wangzhenyang@itcast.cn"
       android:singleLine="true"
       android:ellipsize="marquee"
       android:marqueeRepeatLimit="100"
        />

    <GridView
        android:verticalSpacing="10dp"
        android:layout_marginTop="10dp"
        android:numColumns="3"
        android:id="@+id/list_home"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >
    </GridView>

</LinearLayout>
```

界面效果图如下所示：

2.1 自定义可以滚动的 TextView 控件

定义一个文字可以滚动的 Button 示例如下：

```
<Button
        android:focusableInTouchMode="true"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:singleLine="true"
        android:ellipsize="marquee"
        android:text="我是可以滚动的文字,啦啦啦啦啦啦啦啦啦啦啦啦啦啦啦啦啦啦啦啦啦啦啦
啦啦啦啦啦啦啦啦啦"
        />
```

定义一个文字可以滚动的 TextView，编写一个类 FocusedTextView 继承 TextView 类。覆写 isFocused 方法。

```
public class FocusedTextView extends TextView{
    public FocusedTextView(Context context) {
        super(context);
    }
    public FocusedTextView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }
    public FocusedTextView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }
    @Override
    public boolean isFocused() {
        return true;
    }
}
```

使用该控件时方法如下：

```
  <com.itheima.mobileSafe.ui.FocusedTextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="欢迎使用黑马程序员私人订制版的手机安全卫士，制作人：王震阳，联系方式：wangzhenyang@itcast.cn"
       android:singleLine="true"
       android:ellipsize="marquee"
       android:marqueeRepeatLimit="100"
        />
```

注意：这里的标签必须使用类全名。

3.设置中心（★★★）

3.1 自定义设置中心的组合控件

我们的设置中心如下实例图所示，设置中心的每一个 Item 都是自定义控件。我们这里首先制作“设置自动更新”功能。该设置控制软件启动时是否要检查更新。进入设置中心需要在主界面给“设置中心”图标添加一个点击事件。

当点击该“设置中心”的时候进入设置中心界面。

在主 HomeActivity 中给 GridView 设置点击事件。点击不同的图标进入不同的功能界面。

```
private GridView list_home;
protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.home_activity);
      list_home = (GridView) findViewById(R.id.list_home);
      list_home.setAdapter(new HomeAdapter());
      list_home.setOnItemClickListener(new HomeItemClickListener());
   }
```

```
private class HomeItemClickListener implements OnItemClickListener {

      @Override
      public void onItemClick(AdapterView<?> parent, View view, int position, long
id) {
          switch (position) {
          case 0:
             showPwdDialog();
             break;
          case 1:
             entryCallSMSSafe();
             break;
          case 2:
             entryAppManager();
             break;
          case 3:
             entryTaskManager();
             break;
          case 4:
             entryTraaficManager();
             break;
          case 5:
             entryAntiVirus();
             break;
          case 6:
             entryCleanCache();
             break;
          case 7:
             entryAtools();
             break;
```

```
            case 8:
                entrySetting();
                break;
            default:
                break;
            }
        }
    }
```

创建 SettingActivity 类，并创建对应的布局文件。

对于上图中的布局我们发现有很多功能相似的地方，比如每一个设置项的布局基本相同，都有 CheckBox 选项等，因

此我们会将上面的每一个条目抽取成自定义控件。如果不抽取出自定义的布局，那么原始布局文件如下：

```
<?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="60dp" >

    <TextView
        android:id="@+id/tv_tile"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_marginTop="8dp"
        android:text="设置自动更新"
        android:textColor="#000000"
        android:textSize="18sp" />

    <TextView
        android:id="@+id/tv_desc"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/tv_tile"
        android:layout_marginLeft="10dp"
        android:text="当前自动升级已经关闭"
        android:textColor="#99000000"
        android:textSize="16sp" />

    <CheckBox
        android:id="@+id/cb_checked"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_centerVertical="true"
        android:layout_marginRight="10dp" />

    <View
        android:layout_width="match_parent"
        android:layout_height="0.1dp"
        android:layout_alignParentBottom="true"
        android:layout_marginLeft="10dp"
        android:layout_marginRight="10dp"
        android:background="#66000000" />

</RelativeLayout>
```

我们针对上面的布局抽取出单独的自定义类 SettingItemView。我们把该类放在 com.itheima.mobileSafe.ui 包下面。

SettingItemView 类继承 RelativeLayout 类。代码清单如下：

```
public class SettingItemView extends RelativeLayout {
    private CheckBox cb_status;
    private TextView tv_desc;
    private TextView tv_title;
    private String title;
    private String desc_on;
    private String desc_off;
    private void initView(Context context) {
        RelativeLayout relativeLayout = (RelativeLayout) View.inflate(context,
R.layout.setting_item, SettingItemView.this);
        cb_status = (CheckBox) relativeLayout.findViewById(R.id.cb_checked);
        cb_status.setClickable(false);
        tv_desc = (TextView) relativeLayout.findViewById(R.id.tv_desc);
        tv_title = (TextView) relativeLayout.findViewById(R.id.tv_tile);
        tv_title.setText(title);
        if(cb_status.isChecked()){
            tv_desc.setText(desc_on);
        }else {
            tv_desc.setText(desc_off);
        }
    }
    public SettingItemView(Context context) {
        super(context);
        initView(context);
    }
    public SettingItemView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        initView(context);
    }
    /**
     * 该构造函数是在布局界面使用的
     * attrs.getAttributeValue 方法的第一个参数是命名空间，命名空间大的
http://schemas.android.com/apk/res/部分是参考的系统命名空间规则，后面跟上包名称
     * 第二个参数是属性名称
     */
    public SettingItemView(Context context, AttributeSet attrs) {
        super(context, attrs);
        //获取布局中的 title 属性值
title =
attrs.getAttributeValue("http://schemas.android.com/apk/res/com.itheima.mobileSa
fe", "title");
```

```
        //获取当 checkBox 打开时对应的描述信息
        desc_on =
attrs.getAttributeValue("http://schemas.android.com/apk/res/com.itheima.mobileSa
fe", "desc_on");
        //获取当 checkBox 关闭时对应的描述信息
        desc_off =
attrs.getAttributeValue("http://schemas.android.com/apk/res/com.itheima.mobileSa
fe", "desc_off");
        //初始化控件
        initView(context);
    }
    public void setChecked(boolean isChecked){
        cb_status.setChecked(isChecked);
        if (isChecked) {
            tv_desc.setText(desc_on);
        }else {
            tv_desc.setText(desc_off);
        }
    }
    public boolean isChecked(){
        return cb_status.isChecked();
    }
    public void setTitle(String title){
        tv_title.setText(title);
    }
    public String getTitle(){
        return tv_title.getText().toString();
    }
    public void setDesc(String desc){
        tv_desc.setText(desc);
    }
    public String getDesc(){
        return tv_desc.getText().toString();
    }

}
```

在 res 目录下的 values 目录下创建 attrs.xml 的文件 声明我们写的属性。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="com.itheima.mobileSafe.ui.SettingItemView">
       <attr name="title" format="string" />
       <attr name="desc_on" format="string" />
       <attr name="desc_off" format="string" />
   </declare-styleable>
</resources>
```

3.2 总结自定义组合控件的过程

1. 声明一个 View 对象，继承相对布局，或者线性布局或者其他的 ViewGroup。
2. 在自定义的 View 对象里面重写它的构造方法，在构造方法里面就把布局都初始化完毕。
3. 根据业务需求添加一些 api 方法，扩展自定义的组合控件；
4. 希望在布局文件里面 可以自定义一些属性。
5. 声明自定义属性的命名空间。

   xmlns:itheima="http://schemas.android.com/apk/res/com.itheima.mobilesafe"
6. 在 res 目录下的 values 目录下创建 attrs.xml 的文件 声明我们写的属性。
7. 在布局文件中写自定义的属性。
8. 使用这些定义的属性。自定义 View 对象的构造方法里面 有一个带两个参数的构造方法布局文件里面定义的属性都放在 AttributeSet attrs，获取那些定义的属性。

至此，本文档完！

2014 年 12 月 29 日 星期一 11:49:31
北京市海淀区中关村软件园国际软件大厦