android
developers

Android开发教程&笔记

android

Android 模拟器

模拟器参数

参数格式

```
emulator [option] [-qemu args]
```

option 选项

-sysdir <dir>	为模拟器在<dir>目录中搜索系统硬盘镜像
-system <file>	为模拟器从<file>文件中读取初始化系统镜像
-datadir <dir>	设置用户数据写入的目录
-kernel <file>	为模拟器设置使用指定的模拟器内核
-ramdisk <file>	设置内存 RAM 镜像文件 (默认为<system>/ramdisk.img)
-image <file>	废弃，使用 -system <file> 替代
-init-data <file>	设置初始化数据镜像 (默认为 <system>/userdata.img)
-initdata <file>	和 "-init-data <file>"使用方法一致
-data <file>	设置数据镜像 (默认为 <datadir>/userdata-qemu.img)
-partition-size <size>	system/data 分区容量大小 (MB)
-cache <file>	设置模拟器缓存分区镜像 (默认为 零时文件)
-no-cache	禁用缓存分区
-nocache	与"-no-cache"使用方法相同
-sdcard <file>	指定模拟器 SDCard 镜像文件 (默认为 <system>/sdcard.img)
-wipe-data	清除并重置用户数据镜像 (从 initdata 拷贝)
-avd <name>	指定模拟器使用 Android 虚拟设备
-skindir <dir>	设置模拟器皮肤 在<dir>目录中搜索皮肤 (默认为 <system>/skins 目录)
-skin <name>	选择使用给定的皮肤
-no-skin	不适用任何模拟器皮肤
-noskin	使用方法与"-no-skin"相同
-memory <size>	物理 RAM 内存大小(MB)
-netspeed <speed>	设置最大网络下载、上传速度
-netdelay <delay>	网络时延模拟
-netfast	禁用网络形态
-tarce <name>	代码配置可用
-show-kernel	显示内核信息
-shell	在当前终端中使用根 Shell 命令
-no-jni	Dalvik 运行时禁用 JNI 检测
-nojni	使用方法与"-no-jni"相同
-logcat <tag>	输出给定 tag 的 Logcat 信息

-no-audio	禁用音频支持
-noaudio	与"-no-audio"用法相同
-audio <backend>	使用指定的音频 backend
-audio-in <backend>	使用指定的输入音频 backend
-audoi-out <backend>	使用指定的输出音频 backend
-raw-keys	禁用 Unicode 键盘翻转图
-radio	重定向无线模式接口到个性化设备
-port <port>	设置控制台使用的 TCP 端口
-ports <consoleport>,<adbport>	设置控制台使用的 TCP 端口和 ADB 调试桥使用的 TCP 端口
-onion <image>	在屏幕上层使用覆盖 PNG 图片
-onion-alpha <%age>	指定上层皮肤半透明度
-onion-rotation 0|1|2|3	指定上层皮肤旋转
-scale <scale>	调节模拟器窗口尺寸(三种：1.0-3.0、dpi、auto)
-dpi-device <dpi>	设置设备的 resolution (dpi 单位) (默认 165)
-http-proxy <proxy>	通过一个 HTTP 或 HTTPS 代理来创建 TCP 连接
-timezone <timezone>	使用给定的时区，而不是主机默认的
-dns-server <server>	在模拟系统上使用给定的 DNS 服务
-cpu-delay <cpudelay>	调节 CUP 模拟
-no-boot-anim	禁用动画来快速启动
-no-window	禁用图形化窗口显示
-version	显示模拟器版本号
-report-console <socket>	向远程 socket 报告控制台端口
-gps <device>	重定向 GPS 导航到个性化设备
-keyset <name>	指定按键设置文件名
-shell-serial <device>	根 shell 的个性化设备
-old-system	支持旧版本(pre 1.4)系统镜像
-tcpdump <file>	把网络数据包捕获到文件中
-bootchart <timeout>	bootcharting 可用
-qemu args....	向 qemu 传递参数
-qemu -h	显示 qemu 帮助
-verbose	和"-debug-init"相同
-debug <tags>	可用、禁用调试信息
-debug-<tag>	使指定的调试信息可用
-debug-no-<tag>	禁用指定的调试信息
-help	打印出该帮助文档
-help-<option>	打印出指定 option 的帮助文档
-help-disk-images	关于硬盘镜像帮助
-help-keys	支持按钮捆绑(手机快捷键)
-help-debug-tags	显示出-debug <tag>命令中的 tag 可选值
-help-char-devices	个性化设备说明
-help-environment	环境变量
-help-keyset-file	指定按键绑定设置文件
-help-virtula-device	虚拟设备管理

-help-sdk-images	当使用 SDK 时关于硬盘镜像的信息
-help-build-images	当构建 Android 时，关于硬盘镜像的信息
-help-all	打印出所有帮助

进程：

在 Android 中，进程完全是应用程序的实现细节，不是用户一般想象的那样。

它们的用途很简单：

- 通过把不信任或是不稳定的代码放到其他进程中来提高稳定性或是安全性
- 通过在相同的进程中运行多个.apk 代码来减少消耗
- 通过把重量级代码放入一个分开的进程中来帮助系统管理资源。该分开的进程可以被应用程序的其他部分单独地杀死
- 如果两个没有共享相同的用户 ID 的.apk 试图在相同的进程中运行，这将不被允许，并且系统会为每一个 apk 程序创建不同的进程会

线程

- Android 让一个应用程序在单独的线程中，指导它创建自己的线程
- 应用程序组件（Activity、service、broadcast receiver）所有都在理想的主线程中实例化
- 没有一个组件应该执行长时间或是阻塞操作(例如网络呼叫或是计算循环)当被系统调用时，这将中断所有在该进程的其他组件
- 你可以创建一个新的线程来执行长期操作

Android 释放手机资源，进程释放优先级

当系统资源消耗，Android 将会杀死一些进程来释放资源。

进程优先级顺序：

① 前台进程：

包含一个前台 Activity、包含一个正在运行的广播接收器、正在运行的服务（当前用户所需的 Activity、正在屏幕顶层运行的 Activity）

② 可视进程：

包含一个可视化的 Activity（Activity 可视的，但是不是在前台的（onPause））、例如显示在一个前台对话框之后的以前的 Activity）

③ 服务进程：

包含一个被开启的服务(处理服务，不是直接可视，例如媒体播放器，网络上传、下载)

④ 后台进程：

包含一个不可视的 Activity(带有一个当前不可视的 Activity、可以在任意时刻杀死该进程来回收内存)

⑤ 空进程

没有持有任何应用程序组件

Android 应用开发 1

分析 Hello Android

打开 Hello Android 工程

src 文件夹 源文件	HelloAndroid.java 主程序文件	R.java 资源文件	Android Library Java 库	Assets 文件夹 静态文件 打包

res 文件夹		
drawable 文件夹	layout 文件夹	values 文件夹
程序图标 (ico.png)	布局 UI (main.xml)	程序用到的 String、颜色**(string.xml)

AndroidMainfest.xml		
描述应用程序、构成、组件、权限		
bin 文件夹		
classes.dex	HelloAndroid.apk	自定义的包文件夹
编译的 java 二进制码	Android 安装包(APK 包)	存放编译后的字节码文件

Main.xml

```
<LinearLayout></LinearLayout> 整体布局 表示线性布局
xmlns:android="http://schemas.android.com/apk/res/android"   名字空间
android:orientation="vertical" 控件布局 垂直往下布局
android:layout_width="fill_parent"
android:layout_height="fill_parent" 上层控件填充满

<TextView
    android:layout_width="fill_parent"     横向填充满
    android:layout_height="wrap_content"  纵向按实际高度填充
android:text="@string/hello"              要引用到的hello字符串
    />     图形空间 派生于 View
<Button
android:id="@+id/widget40_button_OK"      button控件ID
android:layout_width="wrap_content"
android:layout_height="wrap_content"    按实际宽度高度显示填充
android:text="OK"
></Button>
```

R.java

通过 res 文件夹下的 xml 文件定义自动生成的，main.xml ico.png string.xml 是配套的关联，进行修改后 R.java 自动重新生成

AndroidManifest.xml

有关版本，程序信息，java 包，程序图标，程序记录信息等。

Manifest.xml 文件轮廓

```
<manifest>
    <uses-permission>
    <permission>
    <application>
        <activity>
            <intent-filter>
                <action>
                <category>
                <data>
        <service>
        <receiver>
        <provider>
```

添加编辑框与按钮

```
package zyf.Study.AndroidSturdyByMyself;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
public class AndroidSturdyByMyself extends Activity {
    private EditText getNameEditText;
    private Button button_Login;
    private TextView show_Login_TextView;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    getNameEditText=(EditText)findViewById(R.id.widget29_getName_EditText);
    button_Login=(Button)findViewById(R.id.widget30_Login_Button);
    show_Login_TextView=(TextView)findViewById(R.id.widget31_showLogin_TextView);
    button_Login.setOnClickListener(new OnClickListener(){
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                show_Login_TextView.setText(getNameEditText.getText()+"欢迎您进入");
            }
    });
    }
}
```

使用 Intent 启动另一个 Activity

```
Intent showNextPage_Intent=new Intent();
showNextPage_Intent.setClass(UsingBundel.this,NextPageActivity.class);
startActivity(showNextPage_Intent);
```

在多个 Activity 之间切换时候，注意每个 Activity 都应在 AndroidManifest.xml 中有所声明定义（如下）

```
<application android:label="@string/app_name"
             android:icon="@drawable/chinazphone">
    <activity android:name=".UsingBundel"
              android:label="@string/app_name">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name=".NextPageActivity"
              android:label="@string/nextpage"></activity>
</application>
```

新的 Activity
在 AndroidManifest.xml 中
必须定义声明

在不同 Task 中启动 Activity

```
Intent.FLAG_ACTIVITY_NEW_TASK
```

Android 应用开发 2

Activity

何谓 Activity：

最简单的就是你可以把 Activity 看成一个 User Interface Program，原则上它会提供使用者一个交互式的接口功能，那一个 Activity 只有一个 UI 吗？非也，举例来说：一个 email 程序，就可能包含三个 Activity:

- 邮件列表的 Activity
- 显示邮件内容的 Activity
- 写新邮件或回复邮件的 Activity

◆ 所有的 Activity 在系统里由 Activity 堆栈所管理，当一个新的 Activity 被执行后，它将会被放置到堆栈的最顶部，并且变成 running Activity，而先前的 Activity 原则上还是会存在于堆栈中，但它此时不会是在前景的情况，除非刚刚那个新的 Activity 离开。

Intent 与 Intent filters

Intent：

Android 使用了一个很特别的类别 Intent，用来从一个画面跳另一个画面，Intent 是用来描述一个程序想要做些什么事情。在 Intent 的数据结构中有两个很重要的部分，一个是动作（action）及对数据产生反应（data to act upon）。Action 主要的内容有 MAIN（程序的入口点），VIEW，PICK，EDIT 等等。Data 则是用 URI 的形式来表示。比如：想要查看一个人的联络数据时，你需要建立一个 Intent，它包含了 VIEW 的动作（action）及指向该人数据的 URI 描述句。

Intent Filter ：

当 Intent 要求做某件事时，IntentFilter 被用来描述这个 Activity 能够做些什么事情。比如一个 Activity 要能够显示个人联络数据，你就必需要在 intentFilter 说明你要如何处理个人联络数据，并用 ACTION_VIEW 呈现出来。IntentFilter 都会在 AndroidManifest.xml 清单里面声明。

Broadcast Intent Receiver

◆ 当你想要写一个程序来对外部的事件做些处理时，可以实用 Broadcast Intent Receiver 。比如：当电话响铃时，有短信时。Broadcast Intent Receiver 它并不能够拿来显示 UI 画面，它必需利用 NotificationManager 来通知使用者它们感兴趣的事件发生了。

◆ Broadcast Intent Receiver 同样的可以在 AndroidManifest.xml 中声明，但你也可以用写 Context.registerReceiver（）程序的方式来注册你自己的 Broadcast Intent Receiver。你自己的程序并不会因为 Broadcast Receiver 被呼叫而被它执行起来，而是当 BroadcastReceiver 被触发时系统会依据需求来执行相应的程序。程序可以利用 Context.sendBroadcast()来发出它们自己的 IntentBroadcast 给其它的程序。

Intent 与 Activity

◆ 而画面的切来切去则是由 resolving Intent 来实现的。当你想产生新的画面时，现行的 Activity 就使用 startActivity(myIntent)。然后系统就会根据所有已安装的程序所定义的 Intent filter 来看哪个程序是最适合 myIntent。当 startActivity 被呼叫时，resolving Intents 的处理过程是伴随而来的。Resolving Intent 提供我们两个好处：

- 让 Activities 可以很容易的利用 Intent 的方式去使用别的程序的功能。
- Activities 可以很容易的在任何情况下由新的 Activity 所取代。

添加新的 Activity

```
package zyf.Android.Study;
import ……
import android.content.Intent;
import android.net.Uri;
import android.view.View.OnClickListener;
public class AndroidStudy_TWO extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final EditText inName = (EditText) findViewById(R.id.name);
        final TextView result = (TextView) findViewById(R.id.result);
        Button button_Start_Browser = (Button) findViewById(R.id.submit_toNET);
        Button button_Login=(Button) findViewById(R.id.show_Login);
        Button button_showLoginName=
                              (Button)findViewById(R.id.submit_toshowLoingName);
        button_Start_Browser.setOnClickListener(new OnClickListener(){
            @Override
            public void onClick(View v) {
            Uri myUri = Uri.parse("http://www.flashwing.net");
            Intent openBrowserIntent = new Intent(Intent.ACTION_VIEW,myUri);
            startActivity(openBrowserIntent);
            }
        });
        button_Login.setOnClickListener(new OnClickListener(){
            @Override
            public void onClick(View v) {
                Intent openWelcomeActivityIntent=new Intent();
                openWelcomeActivityIntent.setClass(AndroidStudy_TWO.this,
                                                        Welcome.class);
                startActivity(openWelcomeActivityIntent);
            }
        });
        button_showLoginName.setOnClickListener(new OnClickListener(){
            @Override
            public void onClick(View v) {
                result.setText(inName.getText()+"欢迎您进入");
            }
        });
    }
}
```

android
developers

android