

# 手机自动化测试简介

1. 手机自动化测试的现状
2. 手机实现自动化的主要方法和思路
3. 目前主要的手机自动化工具
4. **android**手机自动化实现的方法
5. **android**手机程序开发简介

1. 在手机设计公司中，由于手机软件平台设计的封闭性，原本的软件黑盒测试基本是通过手动测试来实现的，只有少数的研发实力很强的公司（西门子，诺基亚等），才会引入部分测试内容的自动化所以在这些公司，通常有着非常多的软件测试工程师岗位；
2. 最近几年来，智能机的出现让手动测试的难度越来越高；而产品设计周期不断缩短，设计成本的不断压缩，使得各家公司开始尝试用自动化测试来代替人工，以降低测试成本，提高产品质量。

## 主要的软件测试项：

1. 预测试：有些也叫**sanity test**；
2. 功能测试；
3. 冲突测试；
4. 压力测试；
5. **Stability (MTBF)** 稳定性测试；
6. **Monkey** 测试；
7. 多语种测试（本地化测试）；
8. 场测；

手机自动化是以性能测试优先覆盖，功能测试次之的方式；

手机自动化测试的实现主要有三种模式：

1. 开发程序，安装在手机上，直接进行程序测试；
2. 开发工具，安装在PC上，通过PC与手机通讯，驱动手机动作，模拟操作；
3. 以上两种兼用；

对手机进行PC 控制和模拟操作需要依赖于以下两个条件：

1. 用户操作的模拟；
2. PC和手机间的通讯实现；

## 1. Test Quest

**Test Quest**是一家专门从事手机自动化研究的公司，他们的平台可以对很多手机厂商的产品进行用户操作的模拟。（移动研究院）。

TQ的实现采用的是PC端控制手机通讯和按键模拟，同时在手机内针对手机平台不同植入agent库，负责PC消息（主要是按键消息）向手机底层消息的转换和转发。

2. 各个手机平台的自由的测试框架：**iOS**, **window**, **Android** 等
3. **QTP**插件（实现方式**TQ**，用的人很少，不知道怎么样）

## 1. Monkey Runner

**Monkey Runner**是Android官方提供的一套基于UI的测试工具，他主要用来通过adb通讯连接，从PC上模拟用户操作消息发送给手机，触发手机发生指定的动作；

该工具可以提供的用户事件模拟，包括按键和触屏。另外添加了截图的功能，使用者可以通过编写脚本自行设计测试方法，对手机进行测试。

```
def run():
    device = MonkeyRunner.waitForConnection()
    print 'Correct to get the test device .'

    strcmd = "am start -n com.sogou.kpi/.Sogou_Kpi_TestActivity"
    device.shell(strcmd)
    sleep(6)

    print "start to load test file..."
    device.shell("am broadcast -a ScriptMsg --es " + "File " + TEST_FILE)
    sleep(8)
    print "finish to load test file."

    print "start to read input file..."
    file_object = open('source.txt')
    print 'finish to load input file.'

    print "Test Start!"
    list_of_lines = file_object.readlines()
    for i in range(0, len(list_of_lines)):
        keychar = str(list_of_lines[i])

        #input a del to catch the start time
        device.touch(keyDel[0], keyDel[1], 'DOWN_AND_UP');

        for j in range(0, len(keychar)):
            device.touch(getKey(keychar[j])[0], getKey(keychar[j])[1],
            device.touch(keySpace[0], keySpace[1], 'DOWN_AND_UP');
            device.touch(keySpace[0], keySpace[1], 'DOWN_AND_UP');
            device.touch(keySpace[0], keySpace[1], 'DOWN_AND_UP');
            device.touch(keySpace[0], keySpace[1], 'DOWN_AND_UP');
```

## Monkey Runner

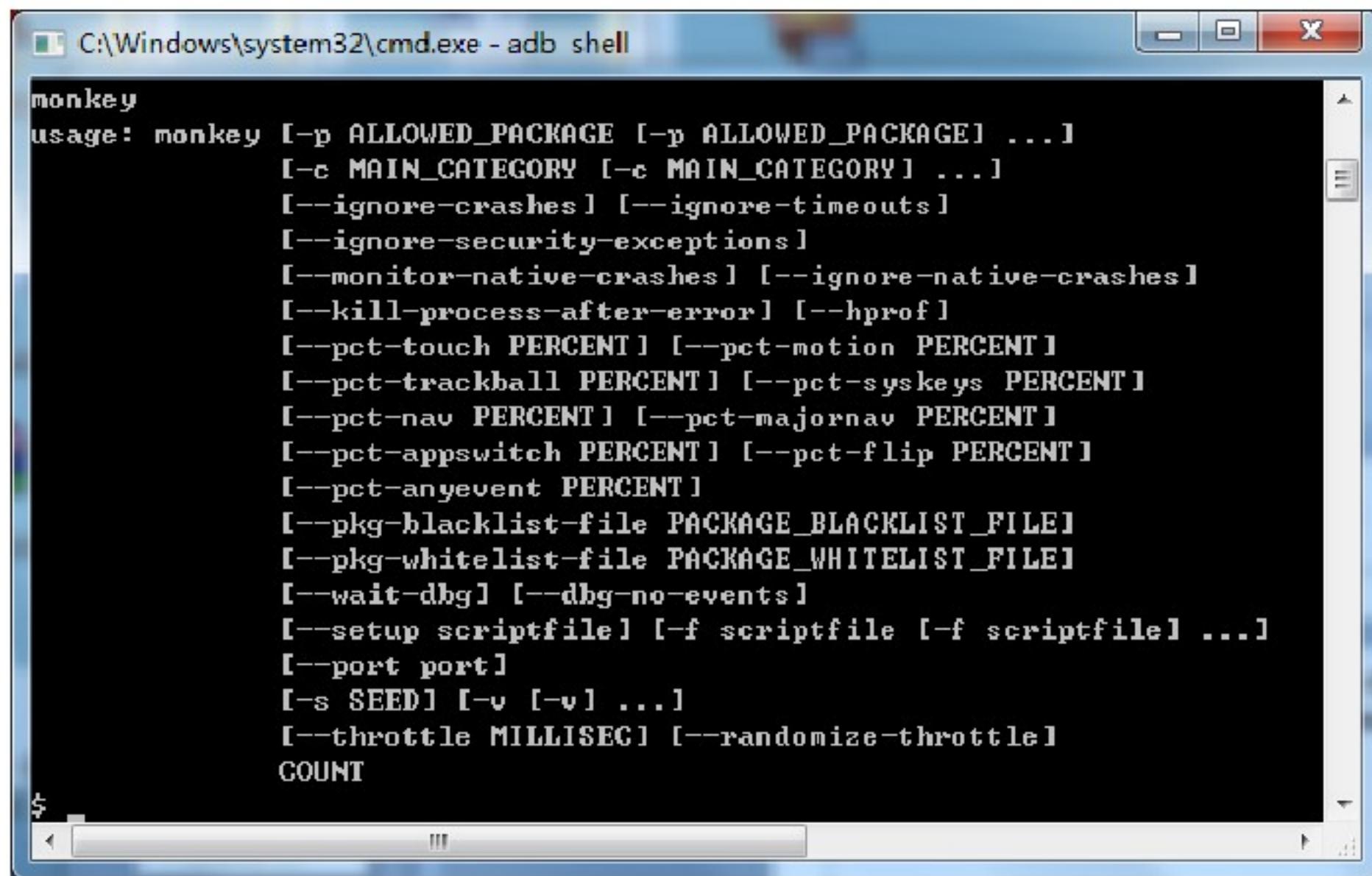
优点：提供了完整的手机驱动接口，测试者不需要了解驱动原理，只需要写脚本就可以自行进行程序测试；

缺点：同一时间只能驱动单一设备。

## 2. 自行设计手机按键模拟方式：

**monkey**: android提供了一个叫做**Monkey**的工具，这个工具用来向用户界面发送随机模拟事件，进行**monkey**测试。

**monkey**工具的实现，说明手机底层存在一套机制，可以接受按键消息，并产生事件模拟；



## 2. 自行设计手机按键模拟方式：

一个实验：

```
C:\Users>adb shell  
$ monkey --port 1083 1083  
monkey --port 1083 1083  
^C  
C:\Users>adb forward tcp:1083 tcp:1083  
C:\Users>telnet 127.0.0.1 1083
```

```
wake  
OK  
key down a  
OK  
key up a  
OK  
touch down 50 50  
OK  
touch up 50 50  
OK
```

利用左边这个原理，我们可以自行代码实现 **monkey runner** 的功能，并增加多设备多线程的控制，**log** 监控，结果反馈，报告生成等更多的支持。

## 3. 其他手机平台的自动化实现方式：

手机平台设计通常都有对外的测试接口，只是并不开放；

比如 iOS,有自己的测试框架；

Symbian也有自己的测试框架；

FeaturePhone的自动化测试控制大多数使用AT指令集及扩展来实现；

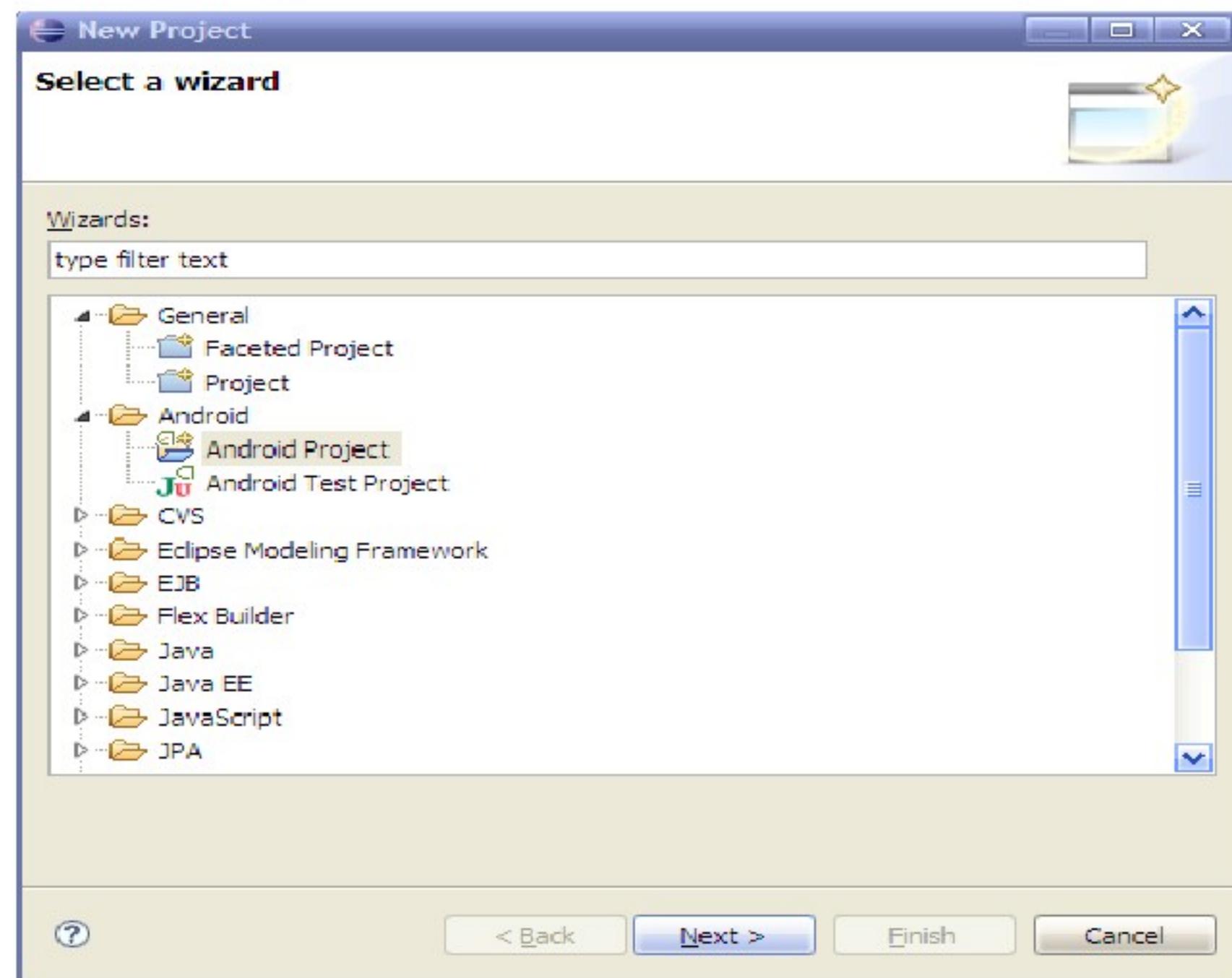
## 1. Android环境搭建

- 所需软件：
  - JDK： 1.6以上
  - Eclipse： 3.4以上
  - Android SDK : <http://developer.android.com/sdk>
  - ADT : <https://dl-ssl.google.com/android/eclipse>
- 1. 安装JDK、配置java环境
- 2. Eclipse安装
- 3. 安装SDK：下载解压后，运行“**SDK Setup.exe**”，选择要安装的API。
- 4. SDK配置：将SDK安装文件夹下的**tools**文件夹的路径加入环境变量“Path”中；
- 5. ADT：Android Development Tools Plug-in, 是Android在Eclipse上的开发工具
- 6. 安装ADT：启动eclipse ->Help->勾选Software Update 和 Available Software->Add Site->输入地址<https://dl-ssl.google.com/android/eclipse>, 关联SDK：菜单window-> Preferences->Android->Browse..., 选择Android SDK安装路径，->OK

## 2. Android-HelloAndroid

(1) 右键New——Project... ,

在“New Project”对话框中选择Android——Android Project



## 2. Android-HelloAndroid

(2) 点击“next”按钮，

进入“New Android Project”，

Project name中输入“HelloAndroid”，

Build Target中选择“Android 2.0”或其他

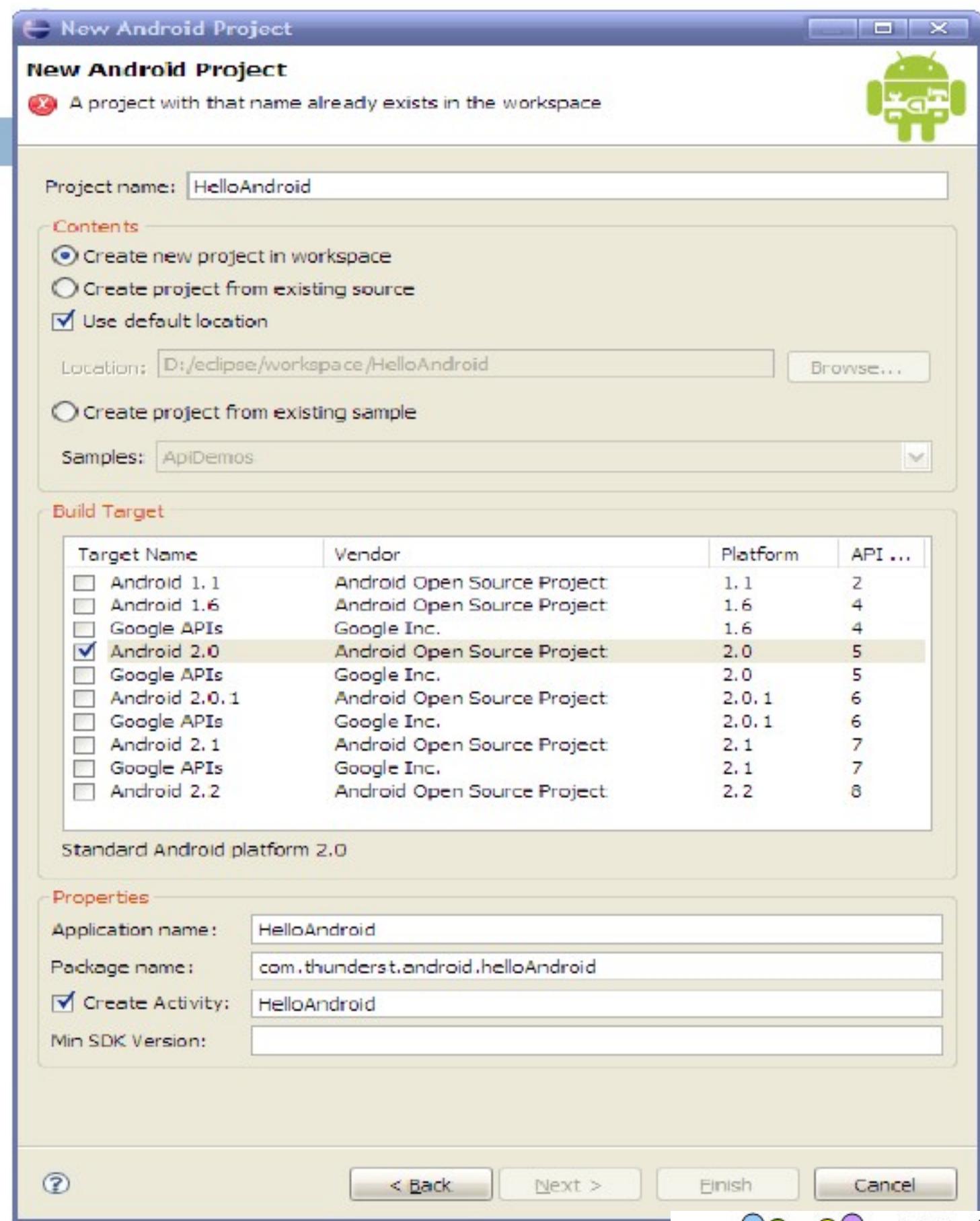
Application name中输入“HelloAndroid”

Package name中输入“

com.thunderst.android.helloAndroid”

Create Activity中输入“HelloAndroid”

点击“Finish”，HelloAndroid项目创建完成



## 2. Android-HelloAndroid

- **src/** java原代码存放目录
- **gen/** 自动生成目录，项目中所有资源的索引文件

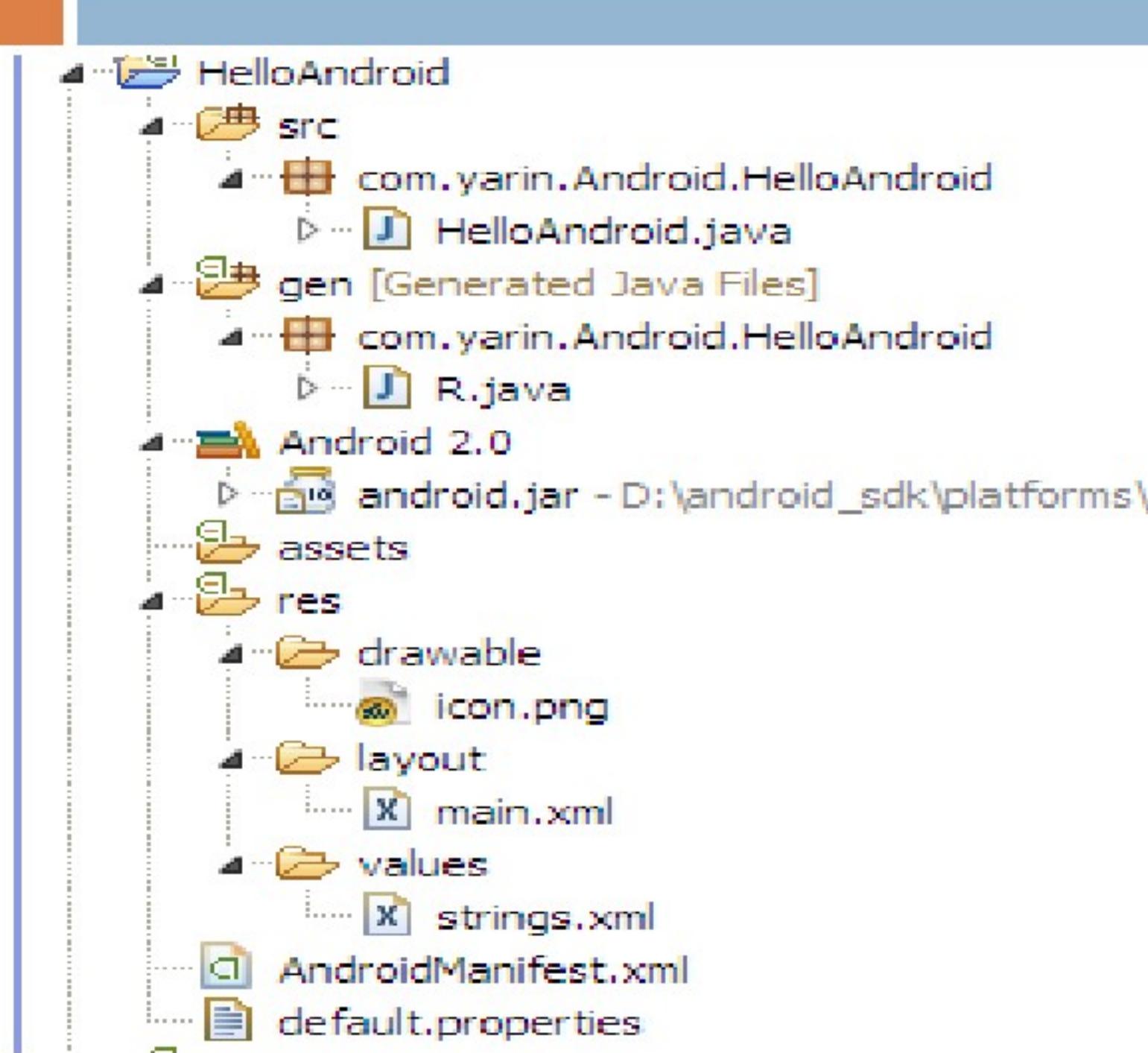
目录中存放所有由Android开发工具自动生成的文件。目录中最重要的就是R.java文件。这个文件由Android开发工具自动产生的。Android开发工具会自动根据你放入res目录的xml界面文件、图标与常量，同步更新修改R.java文件。正因为R.java文件是由开发工具自动生成的，所以我们应避免手工修改R.java。R.java在应用中起到了字典的作用，它包含了界面、图标、常量等各种资源的id，通过R.java，应用可以很方便地找到对应资源。另外编绎器也会检查R.java列表中的资源是否被使用到，没有被使用到的资源不会编绎进软件中，这样可以减少应用在手机占用的空间。

- **res/** 资源目录
- 在这个目录中我们可以存放应用使用到的各种资源，如xml界面文件，图片或数据。
- **AndroidManifest.xml** 功能清单文件

这个文件列出了应用程序所提供的功能，在这个文件中，你可以指定应用程序使用到的服务(如电话服务、互联网服务、短信服务、GPS服务等等)。另外当你新添加一个Activity的时候，也需要在这个文件中进行相应配置，只有配置好后，才能调用此Activity。

- <receiver android:name=".MyBroadcastReceiver">
- <intent-filter>
- <action android:name="ScriptMsg"/>
- </intent-filter>
- </receiver>
- **default.properties** 项目环境信息，一般是不需修改此文件

## 2. Android-HelloAndroid



## 2. Android-HelloAndroid

**HelloAndroid.java**分析：

1.此类必须继承**Activity**,

至少应该覆盖**onCreate()**方法

2.**setContentView(R.layout.main)**

方法设置了此**Activity**显示的UI

3.查看日志的方法：： **Log**类

**v:verbose**

**d:debug**

**i:info**

**e:error**

**w:warn**

```

1 package com.yarin.Android.HelloAndroid;
2
3 import android.app.Activity;
4 import android.os.Bundle;
5 import android.util.Log;
6
7 public class HelloAndroid extends Activity {
8
9     private static final String TAG = "HelloAndroid";
10
11    public void onCreate(Bundle savedInstanceState) {
12        super.onCreate(savedInstanceState);
13
14        Log.v(TAG, "VERBOSE");
15        Log.d(TAG, "DEBUG");
16        Log.i(TAG, "INFO");
17        Log.w(TAG, "WARN");
18        Log.e(TAG, "ERROR");
19
20        setContentView(R.layout.main);
21    }
22}
23

```

## 2. Android-HelloAndroid

Layout/main.xml分析：

1. UI界面的布局文件
2. <LinearLayout>:线性版面配置，所有组件由上到下排列  
    android:orientation 表示从上到下垂直排列  
    android:layout\_width 当前视图占屏幕的宽度  
    android:layout\_height 当前视图占屏幕的高度  
    android:text 填充的文字  
    fill\_parent 填充整个屏幕  
    wrap\_content 根据文字栏位的大小改变此视图的高或宽

```
1<?xml version="1.0" encoding="utf-8"?>
2<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3    android:orientation="vertical"
4    android:layout_width="fill_parent"
5    android:layout_height="fill_parent"
6    >
7<TextView
8    android:layout_width="fill_parent"
9    android:layout_height="wrap_content"
10   android:text="@string/hello"
11   />
12</LinearLayout>
13
```

## 2. Android-HelloAndroid

### R.java分析：

- 在建立项目自动生成，是只读文件，不能更改，是项目中所有资源的索引文件
- 定义了很多常量，这些常量的名字都与res文件夹中的文件名相同
- 在项目中加入新的资源时，只要刷新一下该项目，R.java文件便可以自动生成新的资源索引

```
1①/* AUTO-GENERATED FILE. DO NOT MODIFY.
2
3 package com.yarin.Android.HelloAndroid;
4
5
6 public final class R {
7     public static final class attr {
8
9     }
10    public static final class drawable {
11        public static final int icon=0x7f020000;
12    }
13    public static final class layout {
14        public static final int main=0x7f030000;
15    }
16    public static final class string {
17        public static final int app_name=0x7f040001;
18        public static final int hello=0x7f040000;
19    }
20
21
22 }
23
24 }
```

## 2. Android-HelloAndroid

AndroidManifest.xml分析：

**manifest**:根节点

**xmlns**:命名空间

**package**: 应用程序包

**application**: application级别组件的根节点

**application:icon** 应用程序图标

**application:label** 应用程序名称

**activity**: 与实际的Activity类对应

**Intent-filter**: 此activity支持的intent值

**action**: 组件支持的Intent action

**category**: 组件支持的Intent Category

**uses-sdk**: 此应用程序使用的SDK版本

```

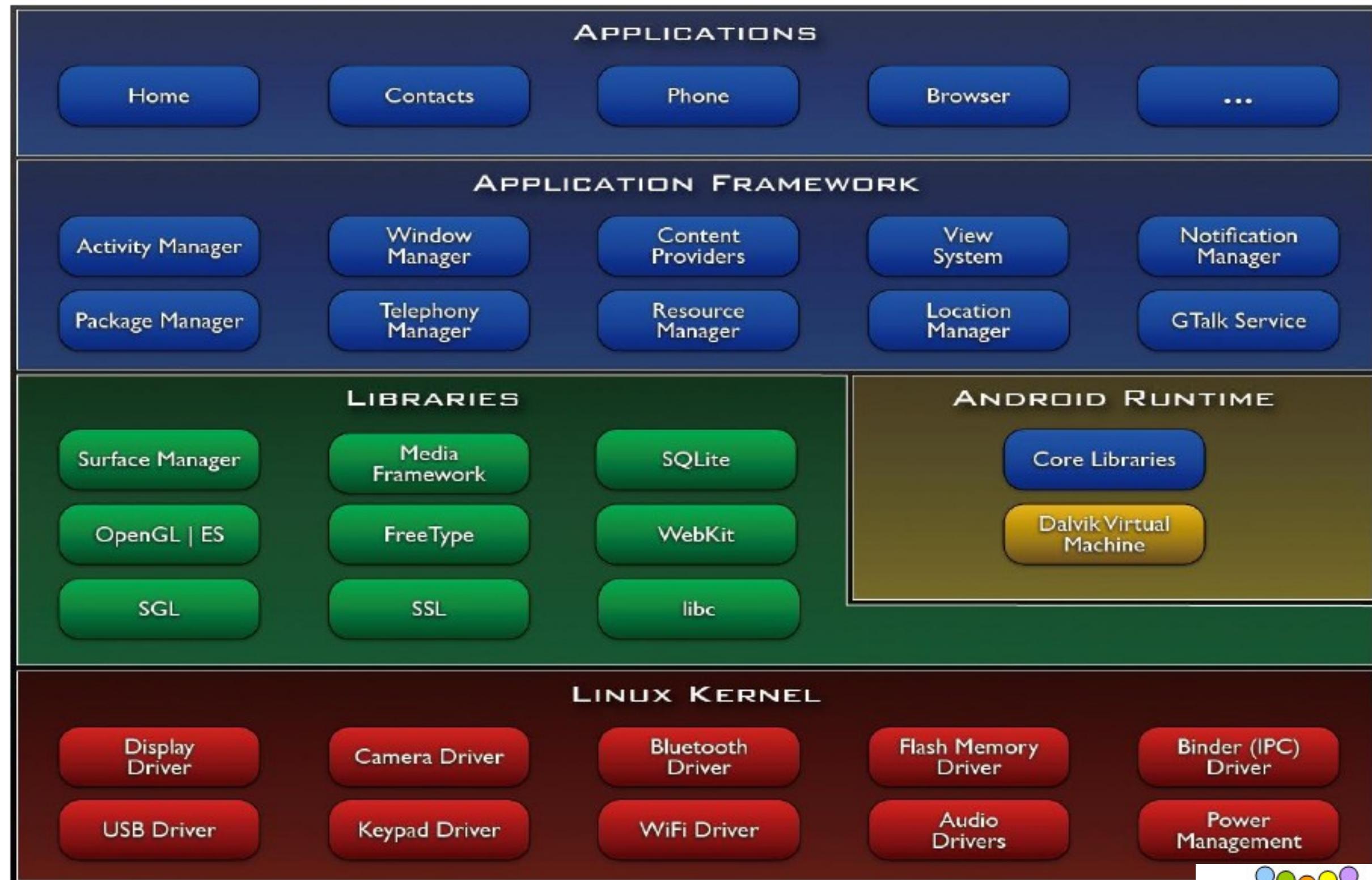
1<?xml version="1.0" encoding="utf-8"?>
2<manifest xmlns:android="http://schemas.android.com/apk/res/android"
3    package="com.yarin.Android.HelloAndroid"
4    android:versionCode="1"
5    android:versionName="1.0">
6    <application android:icon="@drawable/icon" android:label="@string/app_name">
7        <activity android:name=".HelloAndroid"
8            android:label="@string/app_name">
9            <intent-filter>
10                <action android:name="android.intent.action.MAIN" />
11                <category android:name="android.intent.category.LAUNCHER" />
12            </intent-filter>
13        </activity>
14    </application>
15    <uses-sdk android:minSdkVersion="5" />
16</manifest>
```

## 2. Android-HelloAndroid

Values/strings.java 分析：定义了字符串资源

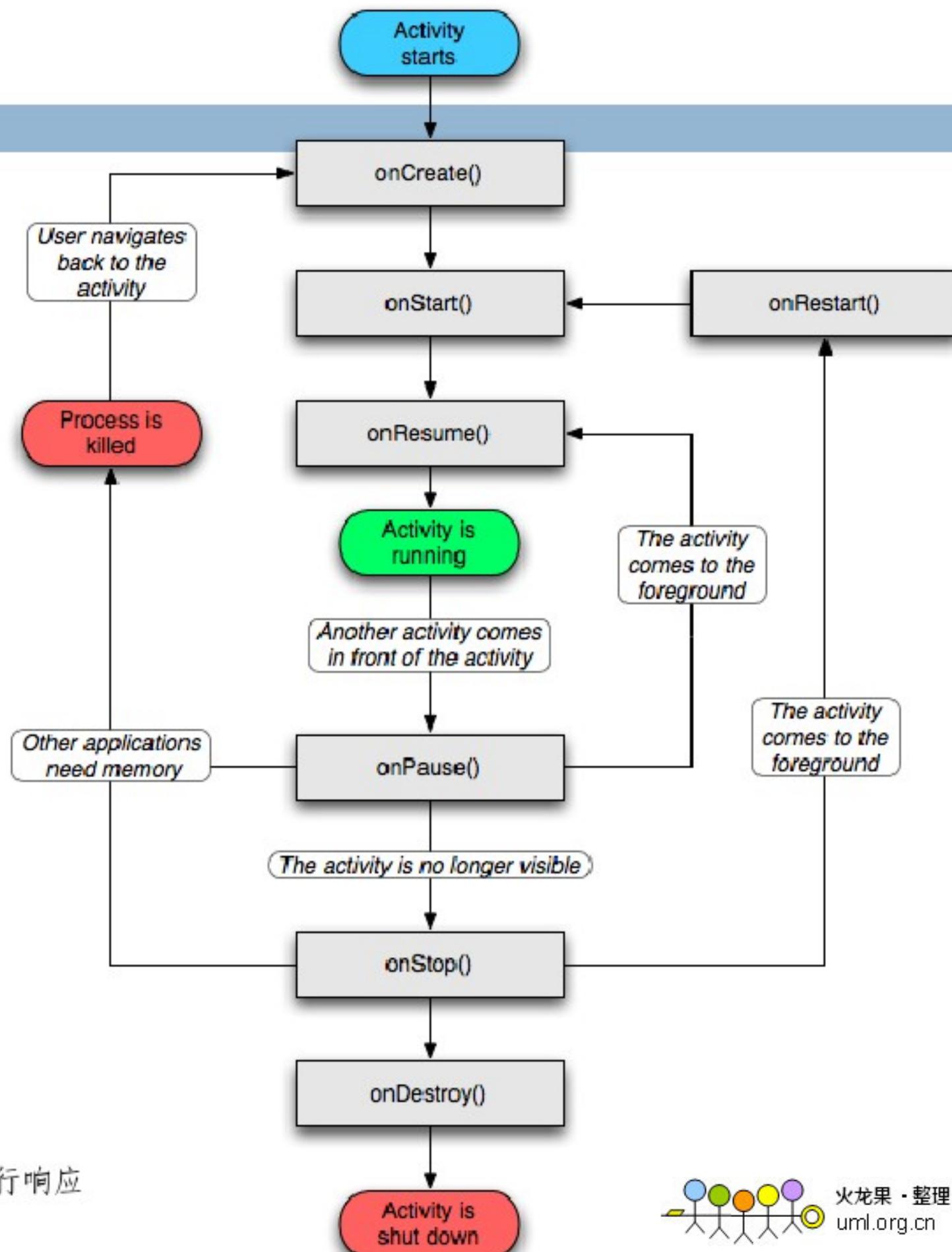
```
1<?xml version="1.0" encoding="utf-8"?>
2<resources>
3    <string name="hello">Hello World, HelloAndroid!</string>
4    <string name="app_name">HelloAndroid</string>
5</resources>
6
```

## 2. Android架构



### 3. 应用的生命周期

- 完全生命周期
  - 开始于`onCreate()`
  - 结束于`onDestroy()`。
- 可见生命周期
  - 开始于`onStart()`
  - 结束于`onStop()`
- 前台生命周期
  - 开始于`onResume()`
  - 结束于`onPause()`



#### Activity:

1. Android应用最基本的模块，称之为“活动”
2. 一个activity就是一个单独的屏幕
3. 每个activity都被实现为一个独立的类，都继承自`android.app.Activity`
4. 每个activity都会显示由视图UI组成的用户接口，对事件进行响应

## 4. 相关资料地址

1. [www.android.com](http://www.android.com)
2. <http://developer.android.com/index.html>
3. 其他