# Android应用和开发环境

## Android的简介

* Android是由Andy Ruby创立的一个手机操作系统，后被Google收购。
* 目标：Google希望能与各方共同建立一个标准化、开放式的移动电话软件平台，从而在移动产业内形成一个开放式的操作平台。

## Android9.x平台特性

* Android的底层建立在linux之上，主要由操作系统、中间件、用户界面和应用软件4层组成
  + 设计思路：采用一种被称为软件叠层(Software Stack)的方式进行构建。
  + 作用
    - 使得层与层之间互相分离，明确各层的分工
    - 保证了层与层的低耦合，当下层的层内或层下发生改变时，上层应用程序无需任何改变
  + 和网络以及java web的后端一样都使用了分层的思想去解决问题。
* Android系统主要由6部分组成。**序号从小到大，越来越底层。**
  1. 系统App层
     + Android系统将包含一系列核心App，如电话拨号，联系人。
     + 普通开发者开发的各种Android程序，都属于这一层
  2. Java API框架层
     + 提供了大量API供开发者使用
     + 调用原生C/C++库的服务。
  3. 原生C/C++库
     + Android包含一套被不同组件所使用的C/C++库的集合。一般来说，Android应用开发者不能直接调用这套C/C++库集，而是通过它上面的JavaAPI框架调用这些库
     + SQLlite：供所有应用程序使用的功能强大的轻量级关系数据库
     + WebKit
     + OpenMAX
     + Libc
     + Media Framework
     + SGL
     + OpenGL ES
  4. Android运行时
     + 由Android核心库和ART（Android runtime）组成
  5. 硬件抽象层（HAL）
     + 主要提供了对linux内核的封装，向上提供蓝牙、摄像头等设备的编程接口，向下隐藏底层的实现细节。
  6. Linux内核
     + 提供了安全性，内存管理、进程管理、网络协议栈和驱动模型等核心服务。

## Gradle自动化构建

* Gradle的运行需要Java\_Home环境变量
* 使用gradle 4.10.2

### Gradle文件结构

* bin：包含gradle的命令-gradle
* docs：包含用户手册、DSL参考指南、API文档
* lib：包含Gradle核心，以及它依赖的JAR包
* media：主要包含Gradle的一些图标
* sample：样例
* src：Gradle源代码

### Gradle项目结构

* Gradle构建文件的默认名字为build.gradle
* project:存放整个项目的全部资源
  + src:分类存放各种源文件、资源文件
    - main：主要存放与项目有关的源文件和资源
      * java：存放与项目相关的资源
      * resource：存放与项目相关的资源
    - test：主要存放与测试有关的源文件和资源
      * java：存放与测试相关的资源
      * resource：存放与测试相关的资源
  + build：存放编译后的class文件、资源的文件夹，该文件夹与src具有对应关系
  + libs：存放第三方JAR包的文件夹
  + build.gradle：gradle构建文件

#### 构建文件

* Gradle采用**领域对象模型**的概念来组织构建文件，在整个构建文件中涉及最核心的API
* Project：代表项目，通常一份构建文件代表一个项目。Project包含大量属性和方法
* TaskContainer：任务容器。每个Project都会维护一个TaskContainer类型的tasks属性。Project和TaskContainer由一一对应的关系
* Task：代表Gradle要执行的一个任务。Task允许指定它依赖的任务、该任务的类型，也可以通过configure（）方法配置任务。它还提供doFirst（）、doLast（）方法添加Action。
  + Action对象和Closure对象都可以代表Action。
  + Closure代表一个闭包，所以Action实际上就代表一个代码块

##### Task创建

* 调用Project的task（）方法创建Task
* 调用TaskContainer的create（）方法创建Task

###### Task指定如下3个常用属性

* dependsOn：指定该Task所依赖的其他Task
* type:指定该task的类型
* 代码块：通过传入的代码块参数配置Task
  + task hello{  
     println "hello world"  
    }  
    //使用  
    gradle hello

### 属性定义

1. 为已有的属性指定属性值
   * Project、Task都是Gradle提供的API，他们本身具有内置属性，因此可以在构建文件中为这些属性指定属性值
   * project的常用属性
     + name：项目的名字
     + path：项目的绝对路径
     + description：项目的描述信息
     + buildDir：项目的构建结果存放路径
     + version：项目的版本号
     + version = 1.0  
       description = "project的属性"  
       task hello{  
        description = "task的属性"  
        println "hello world"  
       }
2. 通过ext添加属性
   * Gradle中所有实现了ExtensionAware接口的API都可通过ext来添加属性
   * ext.prop1 = "111"  
     ext{  
      prop2 = "222"  
     }  
     task hello{  
      prop3 = "333"  
      println "hello world"  
     }
3. 通过-P选项添加属性
4. 通过JVM参数添加属性

### 增量式构建

* 背景：如果每次执行任务都没有造成任何改变，那么重复执行就没有意义
* 如何判断是否产生变化？
  + Gradle将每个任务当做一个黑盒子：只要该任务的输入部分和输出部分都没有改变，Gradle就认为该任务的执行没有造成任何改变
  + 输入：Gradle的task使用inputs属性来代表任务的输入
    - 是一个TaskInput的对象
    - 可以设置文件、文件集、目录、属性等
  + 输出：Gradle的task使用outputs属性来代表任务的输出
    - 该属性是一个TaskOutputs类型的对象
    - 可以设置文件、文件集、目录、属性等
  + task hello{
      
     //def 定义属性的关键字
      
     def sourceTxt = fileTree("source")
      
     def dest = file('dist.txt')
      
     inputs.dir sourceTxt
      
     outputs.file dest
      
     println "hello world"
      
    }