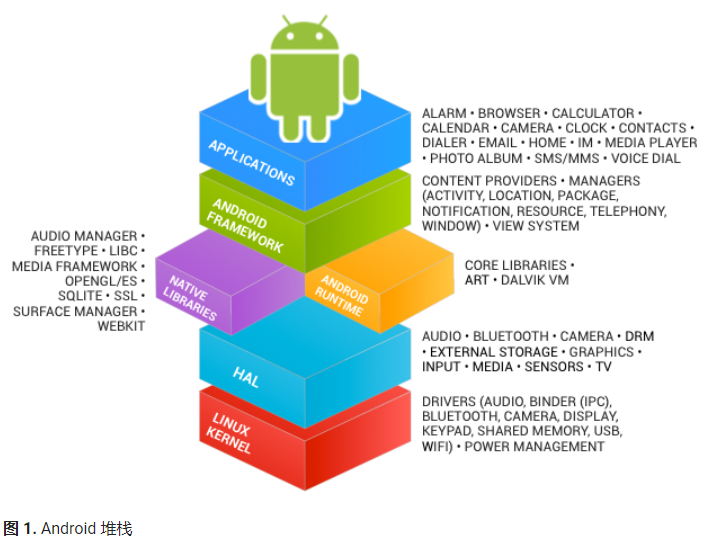
Framework层总体框架

# 概述

Android 是一个针对多种不同设备类型打造的开放源代码软件堆栈。（面试题）



Android系统大体分为四层：Linux内核层，核心库和HAL层，应用框架层，应用层。本文针对应用框架层，即Application Framework层进行分析。Application Framework层在Android系统中起到了关键的承上启下的作用。它使用下层模块实现了服务端组件功能、接口及Linux核心库和驱动相关封装；它对上层应用提供了应用程序开发所需要的API和客户端组件，帮助上层应用实现其应用需求。

本文对Android系统的Application Framework层进行整体架构分析，按照功能和代码分布对各个子模块划分，且对其主要功能、实现原理和相互关系进行分析说明。

# Framework 架构总体描述

Framework即应用框架层，该层在Android系统中起到了关键的承上启下的作用。

Framework为上层应用程序提供了实现程序功能所需要的API和功能组件。内部实现了核心模块的服务端和对上层应用的模块客户端用来提供核心功能，例如：Activity相关的服务端ActivityManagerService和客户端Activity、ActivityThread等、Window相关的服务端WindowManagerService和客户端PhoneWindow、Window、WindowManager等；还提供了Android运行时库和一些本地库，用来调用系统核心库和底层驱动功能为应用程序提供更多支持，例如: 封装了媒体库、SQLite、Binder机制等。

Android应用进程的创建和启动，是由Framework层ActivityManagerService发起，通过操作系统的域套接字（名字为：zygote或zygote\_secondary），将要创建的应用进程相关信息传递给zygote进程，由zygote进程fork创建新应用进程。因为zygote进程创建时，启动了虚拟机，初始化了Android系统运行时，注册了Android运行时库的入口函数，因此新应用进程也继承了zygote进程设置的环境。新应用进程继续初始化完成Binder进程间通信机制的基础设施，最后执行其入口函数，完成新应用进程的创建和启动。

在Framework中，可以通过jni机制调用Android运行时库和本地库，来提供和使用对应的核心库、系统调用和硬件功能。最具代表性的就是，所有由zygote进程启动的进程都会继承zygote进程注册的Android运行时库入口函数的环境设置，因此可以通过该库来使用一些如OpenGLES、InputDevice、Sqlite等库的功能。

综上所述，将Framework代码主要分为三个部分：API和功能组件部分、Android系统服务部分、本地接口层部分。Framework代码分层以及和外部系统的关系，如下图所示：



图表 2‑1

API和功能组件部分：为APP应用程序提供了实现其程序功能所需要的接口和功能组件。

Framework系统服务：实现Framework的核心服务功能的服务端，包括：ActivityManagerService模块、WindowManagerService模块等等。

本地接口层：通过jni机制，与本地库进行交互。

# Framework层模块的划分

对Framework依据其提供的功能进行模块划分，而功能模块内部实现的逻辑架构分层，则由具体子模块自行分析。

Framework根据功能，大致分为如下子模块。

Status Bar：实现StatusBar的显示和逻辑功能，并提供相应的接口给应用程序使用。

View System：实现UI显示逻辑的框架以及提供基础显示控件给应用程序，以便应用程序组织自己的显示UI。

Content Provider：提供应用程序间的数据共享的功能，应用程序通过其定义的标准接口实现提供数据和获得数据。

Resource Manager：对应用程序中非代码的资源做相关处理，并为应用程序提供使用对应资源的方法。

Activity Manager：用于创建和管理应用程序进程和组成应用程序的四大组件。

Window Manager：用于创建和管理系统的显示窗口。

Display Manager：用于创建和管理系统的显示类型和显示设备。

Keyguard Manager：实现系统锁屏的显示和逻辑功能。

Wallpaper Manager：实现系统壁纸的显示和逻辑功能。

Power Manager：用于控制设备的电源状态。

Alarm Manager：提供全局定时机制。

Sensor Manager：提供各种Sensor状态数据的查询和侦听。

Audio Manager：提供音频管理的功能，如可控制音量和钤声模式等。

Notification Manager：实现通知栏的显示和逻辑功能。

Telephony Manager：提供通信相关的功能。

Account Manager：提供账户管理相关功能。

Connectivity Manager：主要提供网络连接状态的监听和监听的操作。

Wifi Manager：提供WIFI相关的功能，包括Wifi station、个人热点、Wifi P2P等功能。

Network Management Service ：提供对物理网络接口的管理服务。

Camera Service：提供设备摄像头使用的功能。

USB Manager：提供管理USB设备的功能。

Battery Service：提供设备电池状态的查询和侦听。

Media Framework：提供对各种多媒体进行管理和支持的功能

Surface Manager：实现系统显示绘图图层处理的框架和功能。

由于工作主要在WiFi研发相关，对WifiManager相对比较熟悉，研究framework代码就从WiFi入手，下一篇文章：WiFi基础