Android 系統架構簡介

羅升陽

http://weibo.com/shengyangluo

http://blog.csdn.net/luoshengyang

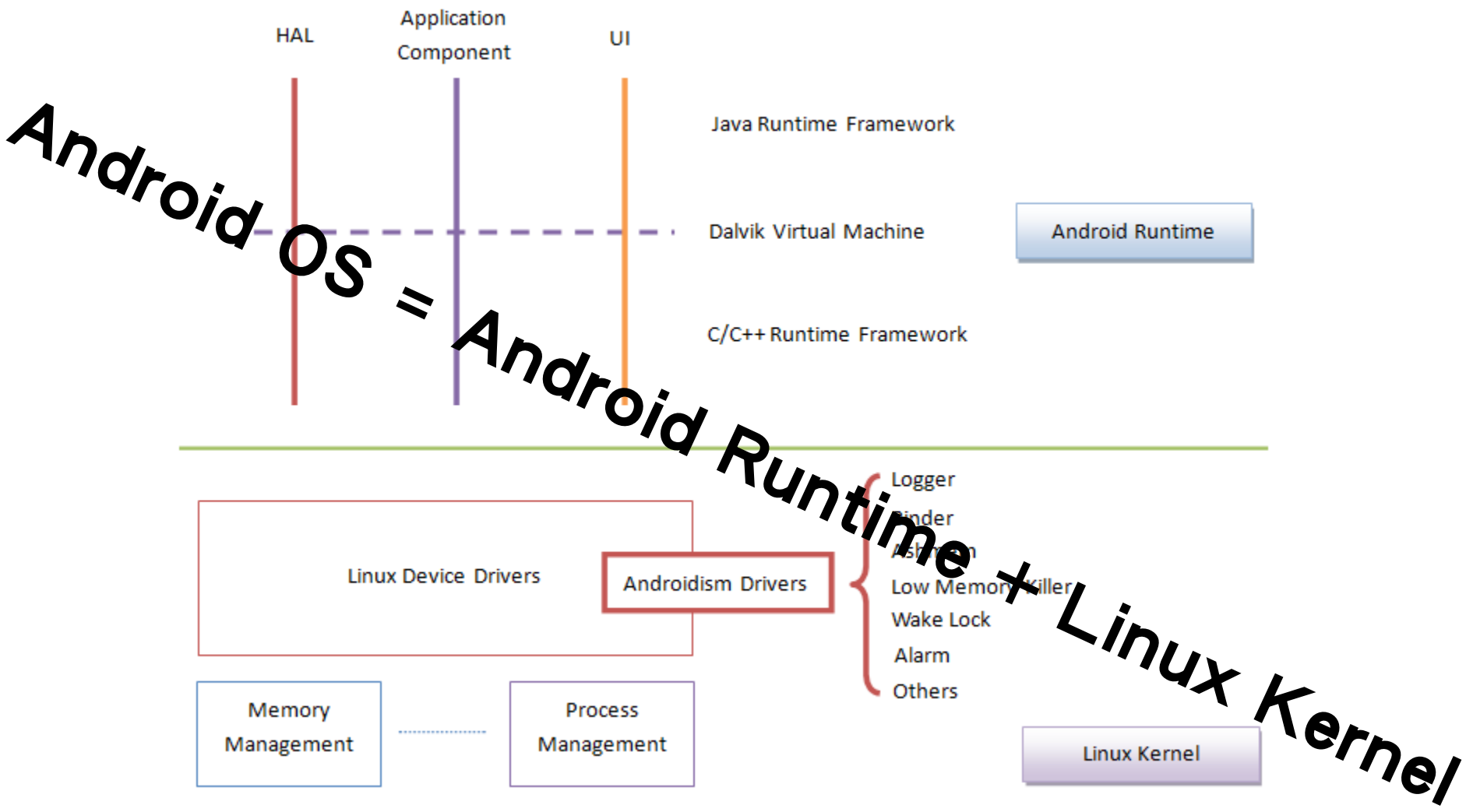
About Me

- 《老羅的 Android 之旅》博客作者
- 《 Android 系統來源碼情景解析》書籍作者
- 博客：http://blog.csdn.net/Luoshengyang
- 微博： http://weibo.com/shengyangluo

Agenda

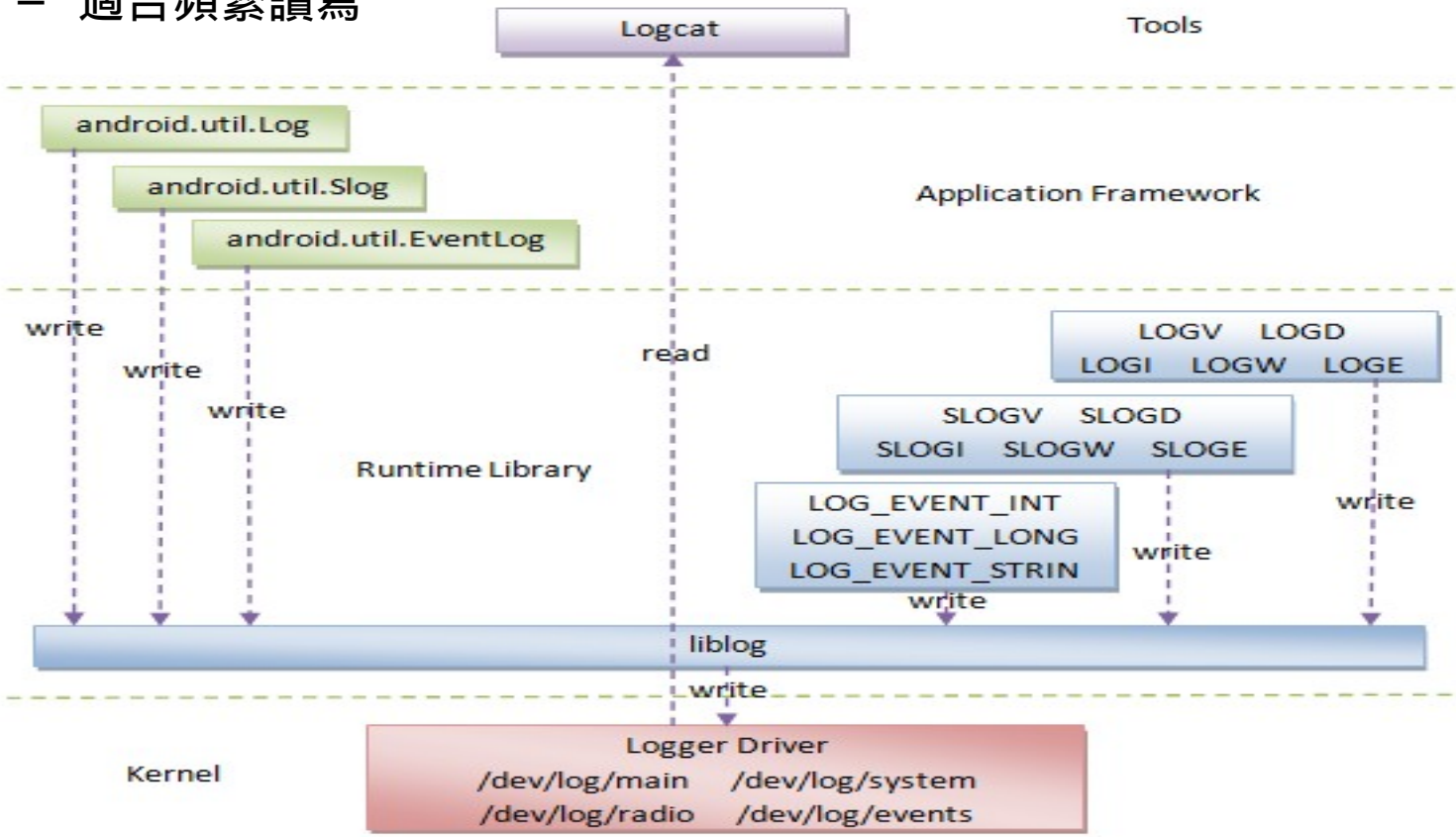
- Android 系統整體架構
- Android 專屬驅動
- Android 硬體抽象層
- Android 應用程式套件
- Android 應用程式框架
- Android 用戶介面架構
- Dalvik 虛擬機

Android 系統整體架構



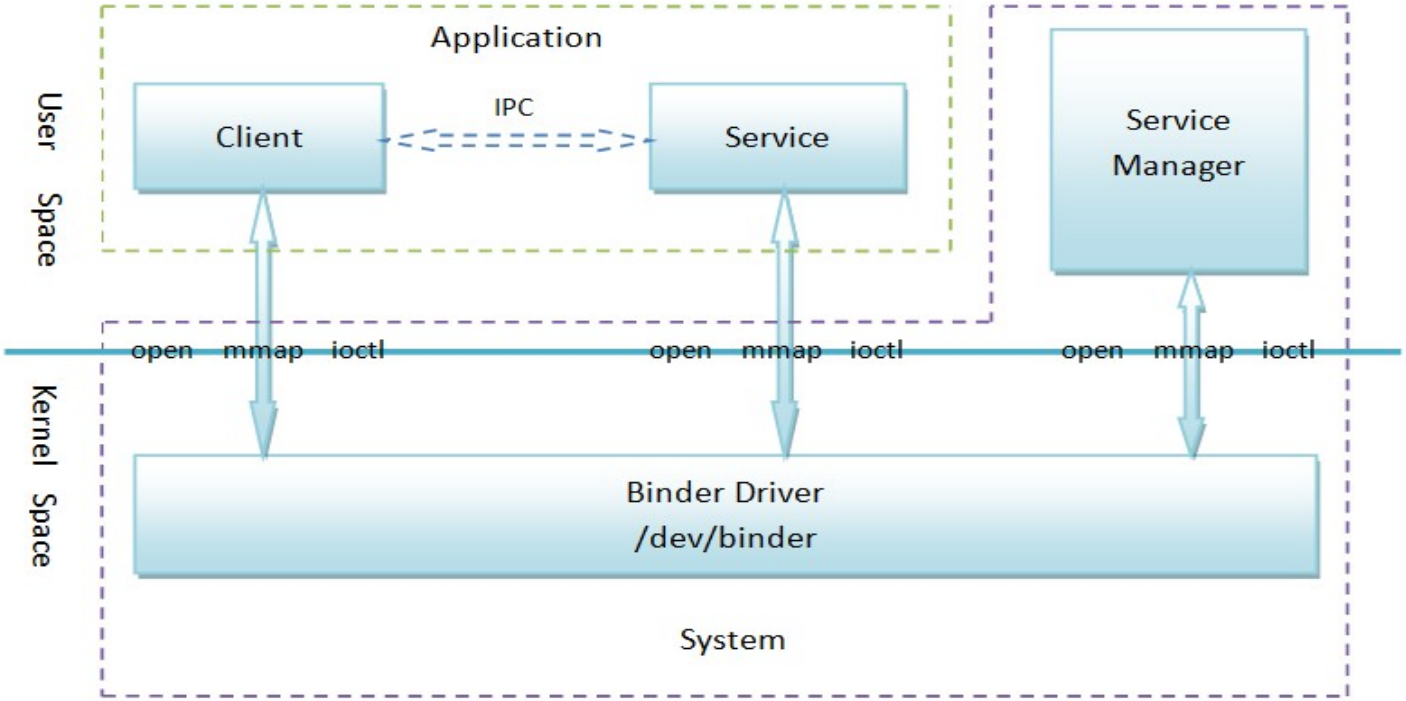
Android 專屬驅動

- Logger
  - 完全記憶體作業
  - 適合頻繁讀寫



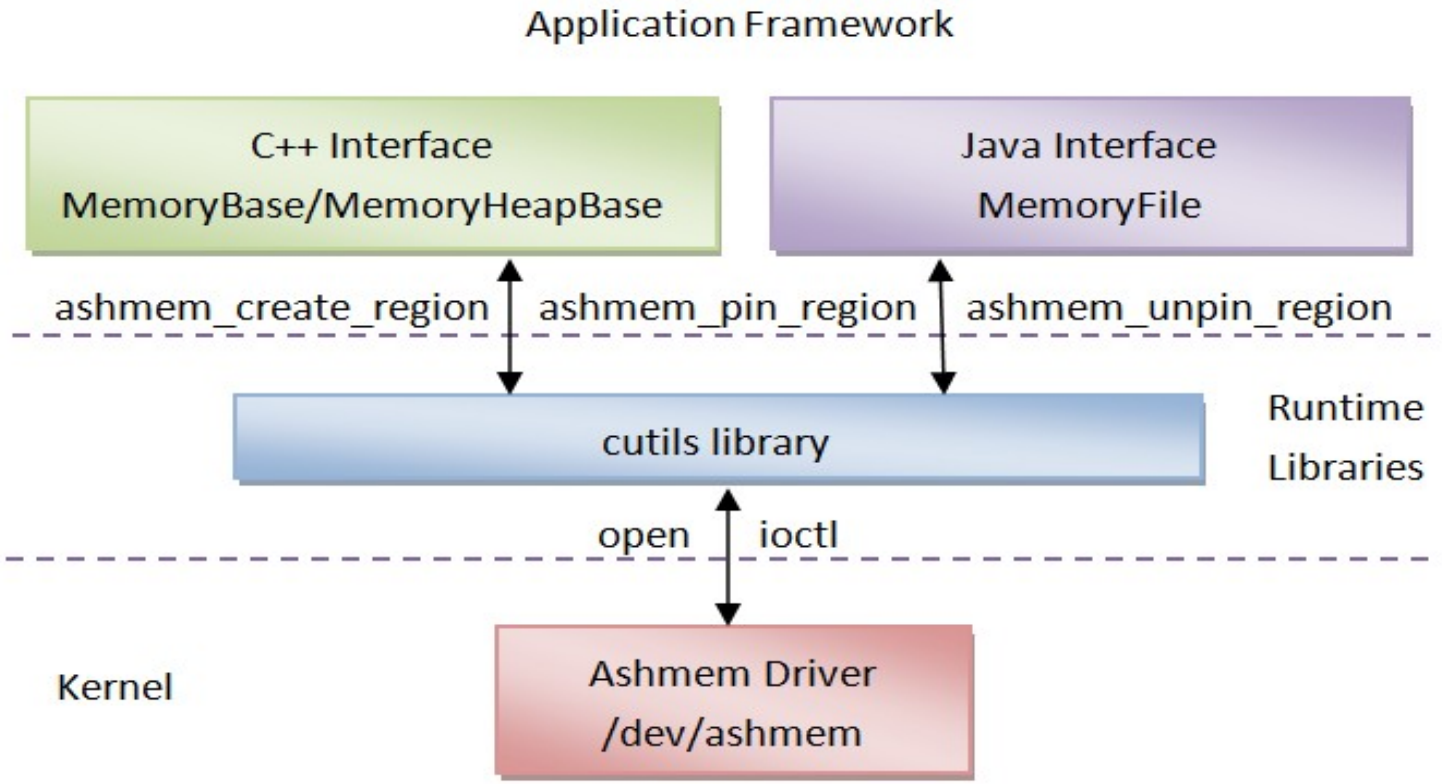
Android 專屬驅動 (續)

- Binder
  - Client/Server 模型
  - 程式間一次資料拷貝
  - 程式內直接呼叫



Android 專屬驅動 (續)

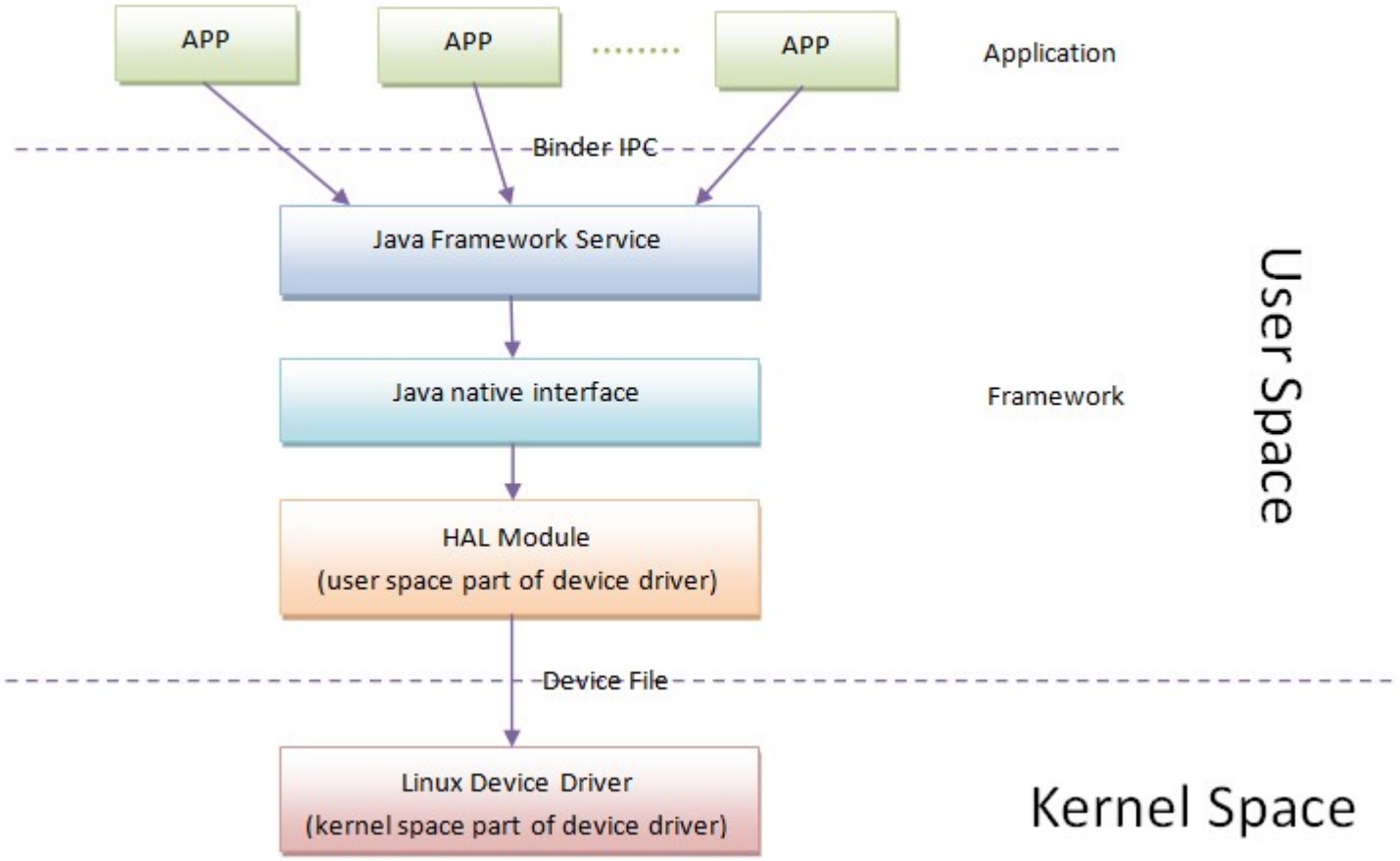
- Ashmem
  - 使用檔案說明符說明
  - 通過 Binder 在程式間傳遞



Android 硬體抽象層 HAL

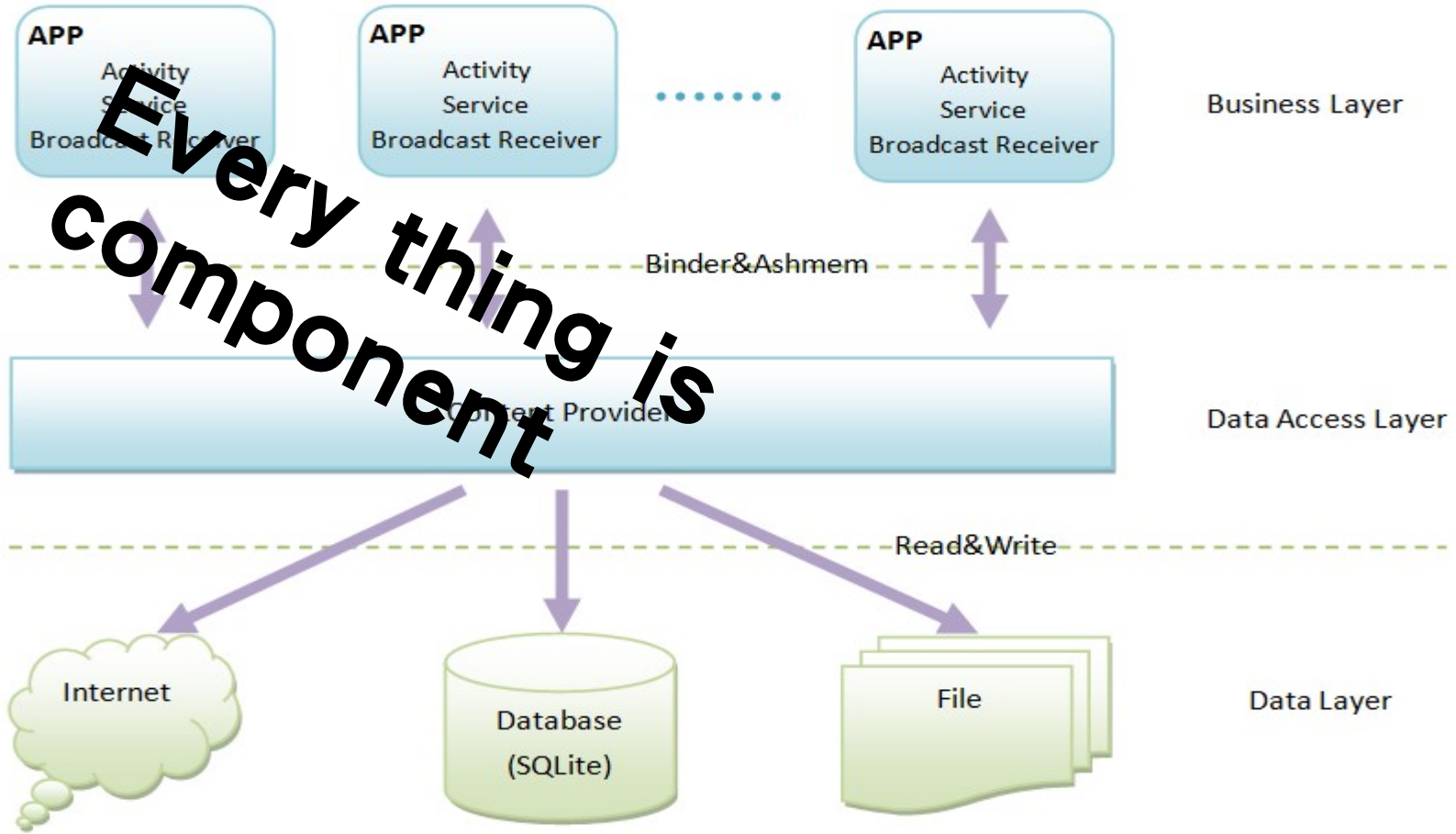
- 裝置驅動分為核心空間和用戶空間兩部分
  - 保護廠商利益（出發點）
  - 核心空間主要負責硬體存取邏輯（ GPL ）
  - 用戶空間主要負責參數和存取流程控制（ Apache License ）
- 用戶空間部分裝置驅動即為 HAL Module
  - HAL Module 通過裝置檔案存取核心空間部分裝置驅動
- 系統服務通過 HAL Module 對硬體進行管理
  - 系統服務通過 JNI 存取 HAL Module
- 應用程式通過系統服務對硬體進行存取
  - 應用程式通過 Binder IPC 存取系統服務

Android 硬體抽象層 HAL(續)



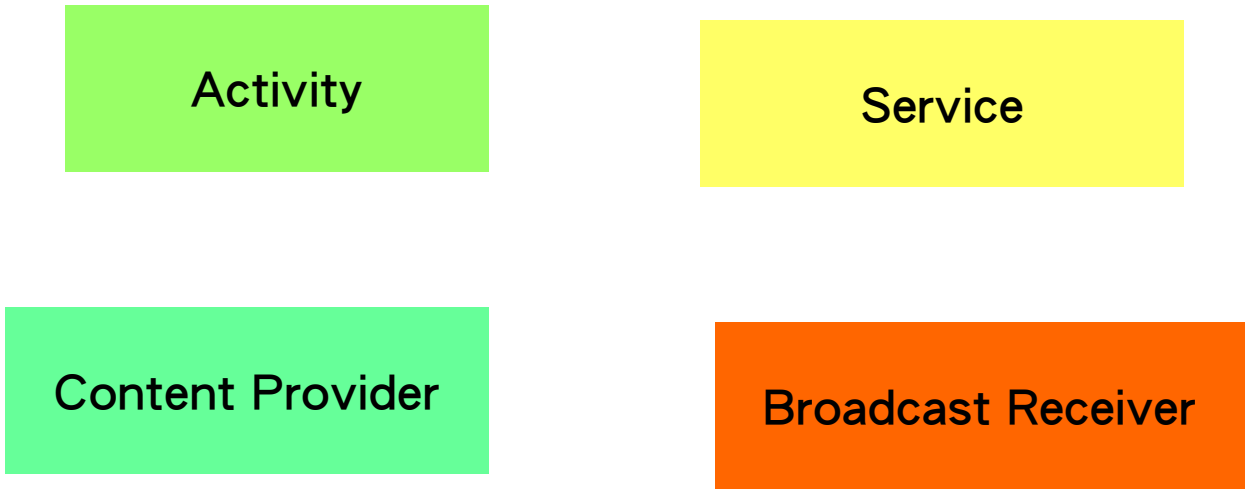
Android 應用程式套件

- Android 應用程式的一般架搆



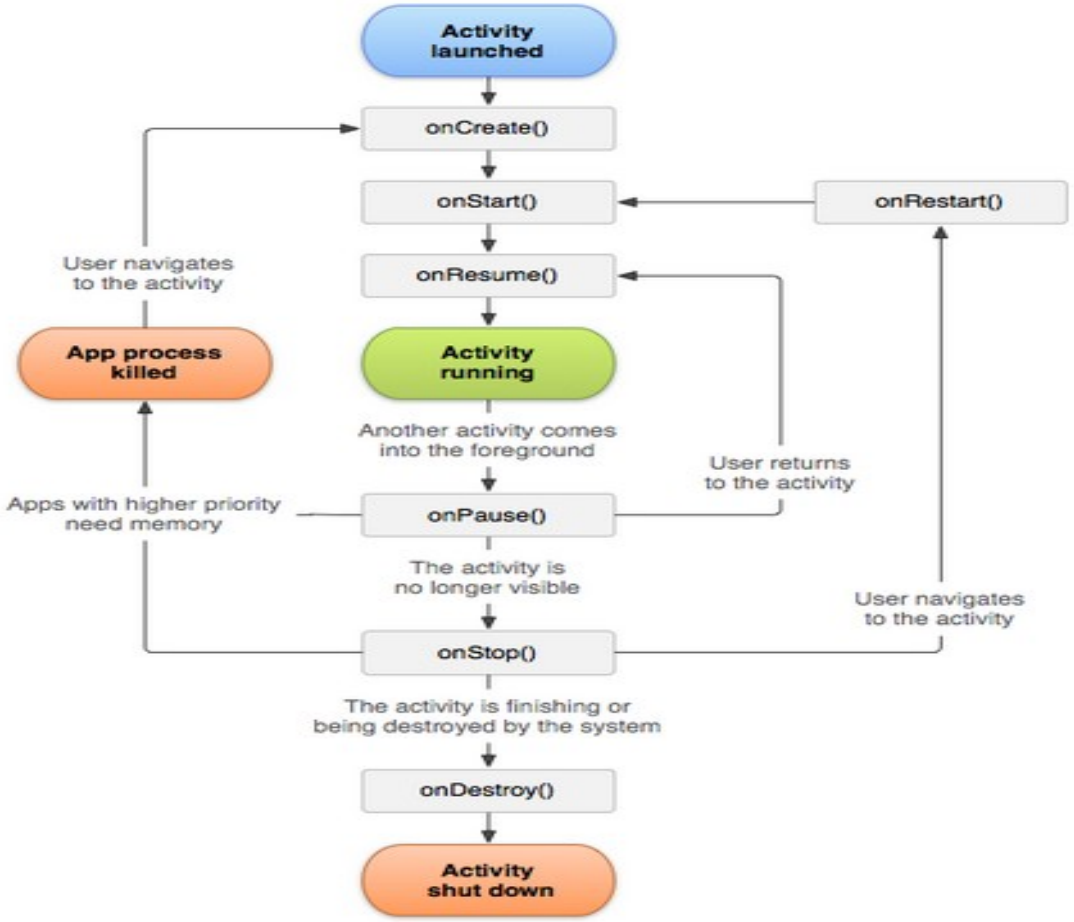
Android 應用程式套件 (續)

- 四大套件 (磚頭)
  - Activity -- UI 、互動
  - Service -- 背景計算
  - Broadcast Receiver -- 廣播
  - Content Provider -- 資料



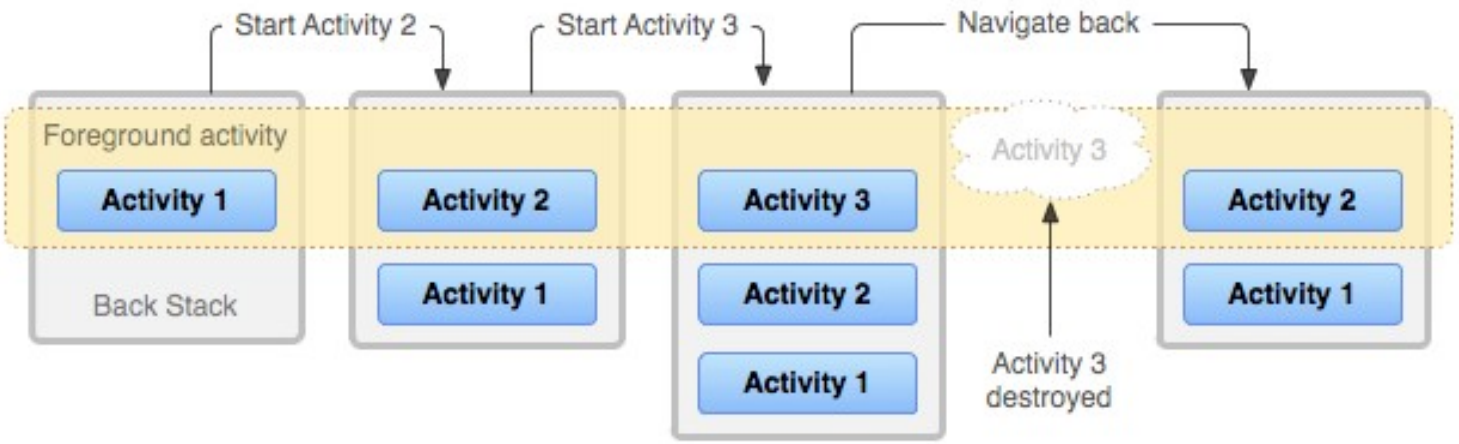
Android 應用程式套件 (續)

- Activity 生命周期
  - 由 ActivityManagerService 管理



Android 應用程式套件 (續)

- Activity 堆疊
  - 由 ActivityManagerService 維護

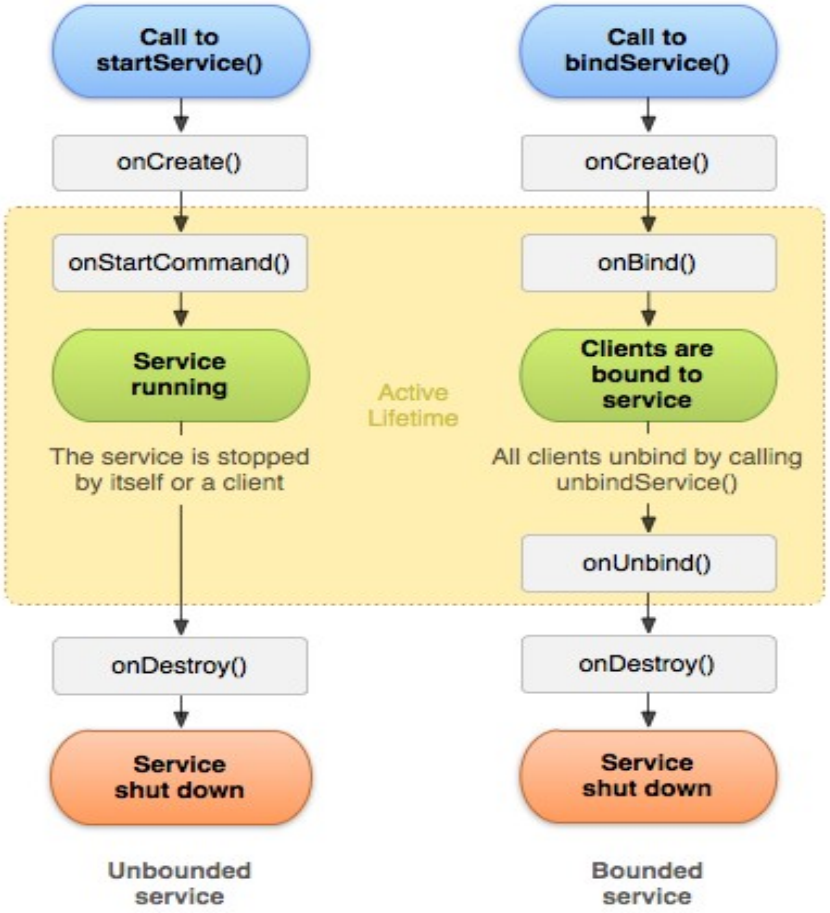


Android 應用程式套件 (續)

- Activity 在堆疊中以 Task 的形式聚集在一起
  - Task 由一家族相關的 Activity 組成，說明用戶完成某一個作業所需要的 Activity
  - 當我們從 Launcher 上點擊一個應用圖像的時候，就啟動一個 Task
  - Task 是用 Android 多工作的一種體現
  - http://developer.android.com/guide/components/tasks-and-back-stack.html

Android 應用程式套件 (續)

- Service
  - Unbounded service
  - Bounded service

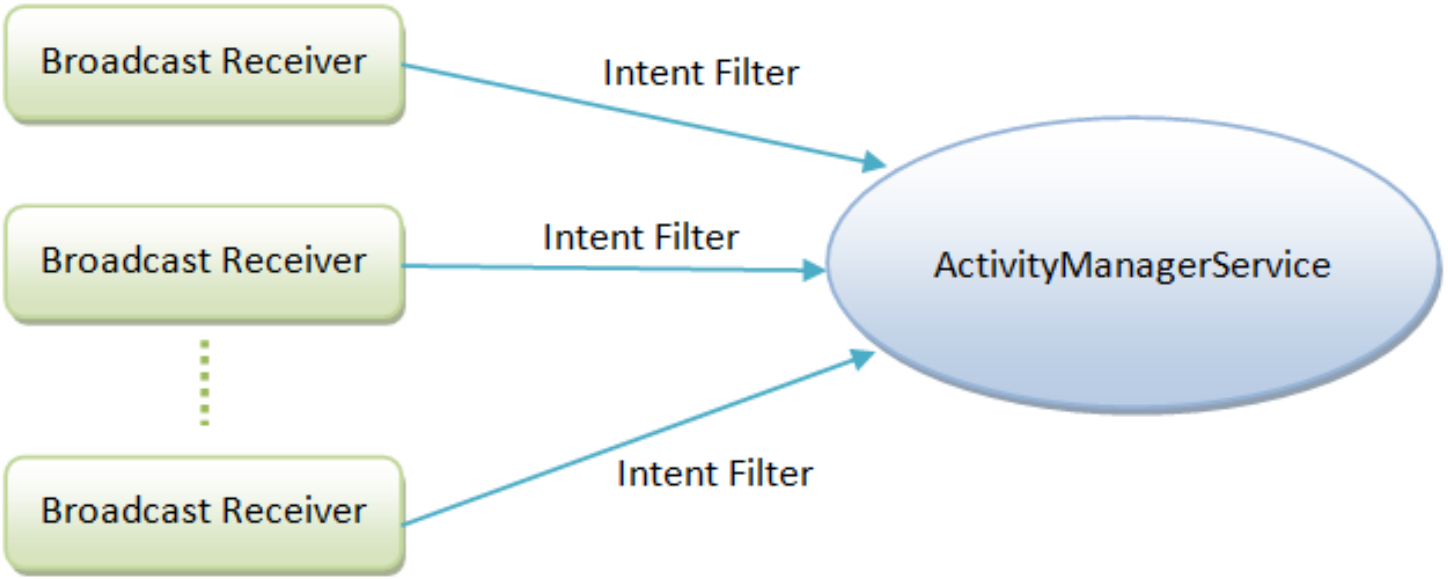


Android 應用程式套件 (續)

- Broadcast Receiver
  - 註冊
    - 靜態 -- AndroidManifest.xml
    - 動態 -- Context.registerReceiver
  - 廣播
    - 無序 -- Context.sendBroadcast
    - 有序 -- Context.sendOrderedBroadcast

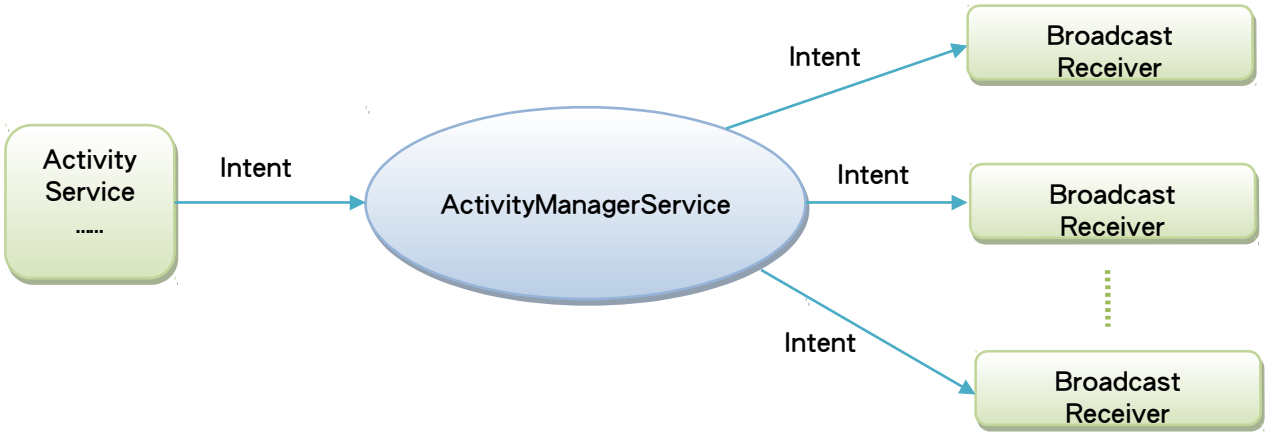
Android 應用程式套件 (續)

- 註冊廣播



Android 應用程式套件 (續)

- 傳送廣播

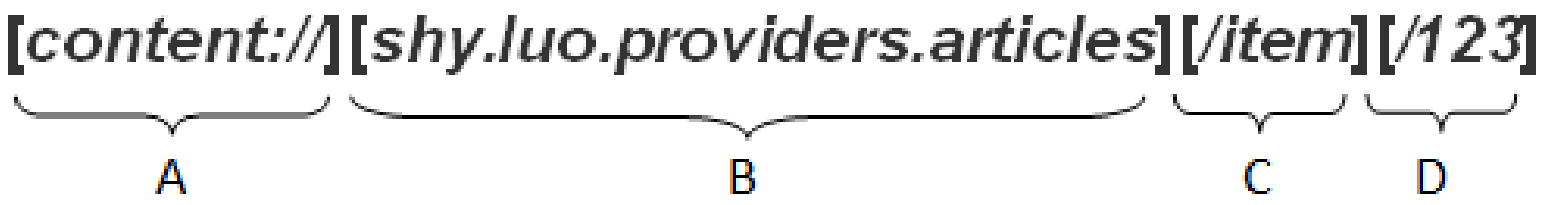


Android 應用程式套件 (續)

- Content Provider
  - 通過 URI 來說明
  - 資料存取介面
  - 資料更新機制

Android 應用程式套件 (續)

- Content Provider 的 URI 架構
  - A -- Scheme
  - B -- Authority
  - C -- Resource Path
  - D -- Resource ID



Android 應用程式套件 (續)

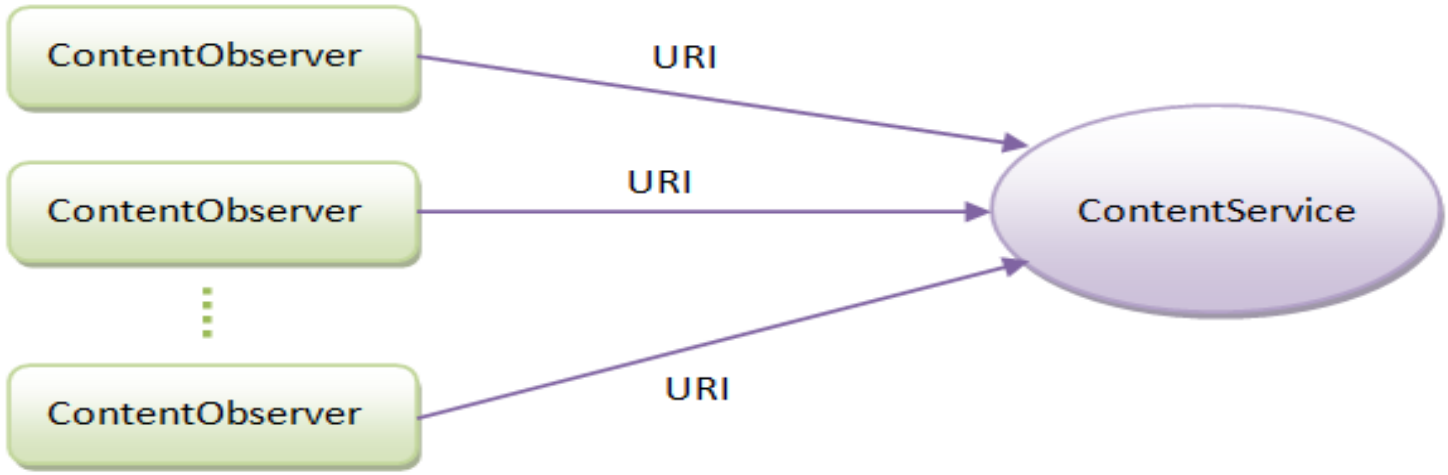
- Content Provider 資料存取介面
  - Insert
  - Update
  - Delete
  - Query
  - Call -- Hidden

Android 應用程式套件 (續)

- Content Provider 資料更新機制
  - 註冊內容觀察者 -- ContentResolver.ContentObserver
  - 傳送資料更新通知 -- ContentResolver.notifyChange

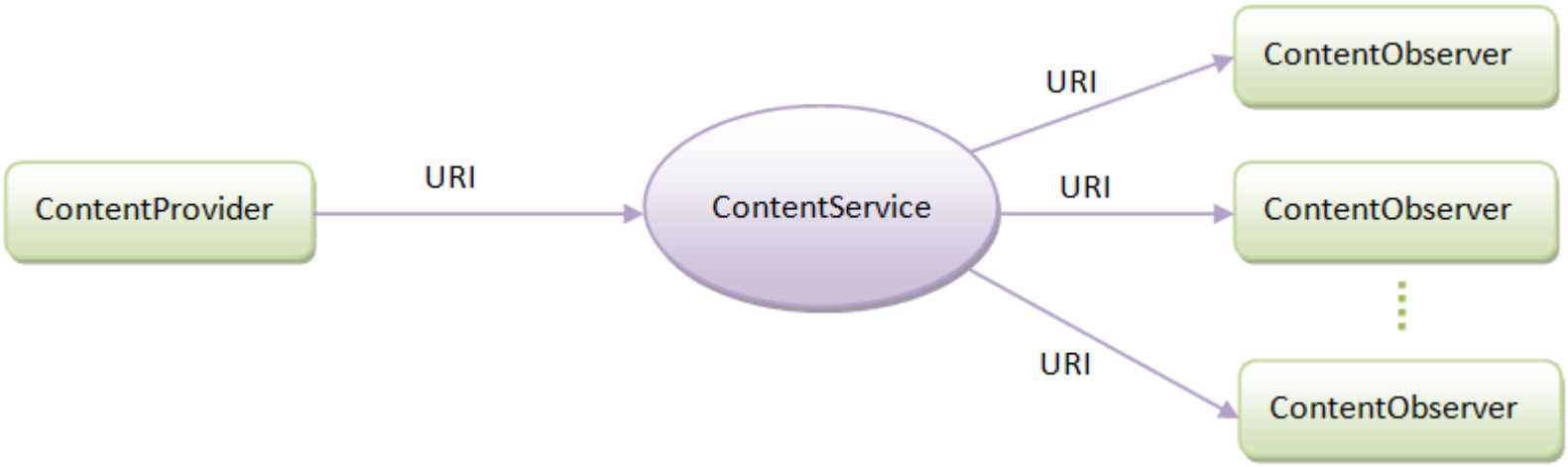
Android 應用程式套件 (續)

- 註冊 Content Provider 的內容觀察者



Android 應用程式套件 (續)

- 傳送 Content Provider 資料更新通知



Android 應用程式框架

- 管理硬體
- 提供服務
- 套件管理
- 程式管理

Android 應用程式框架 (續)

- 按服務類別划分
  - Hardware Service
    - CameraService
    - LocationManagerService
    - LightsService
    - ……
  - Software Service
    - PackageManagerService
    - ActivityManagerService
    - WindowManagerService
    - ……

Android 應用程式框架 (續)

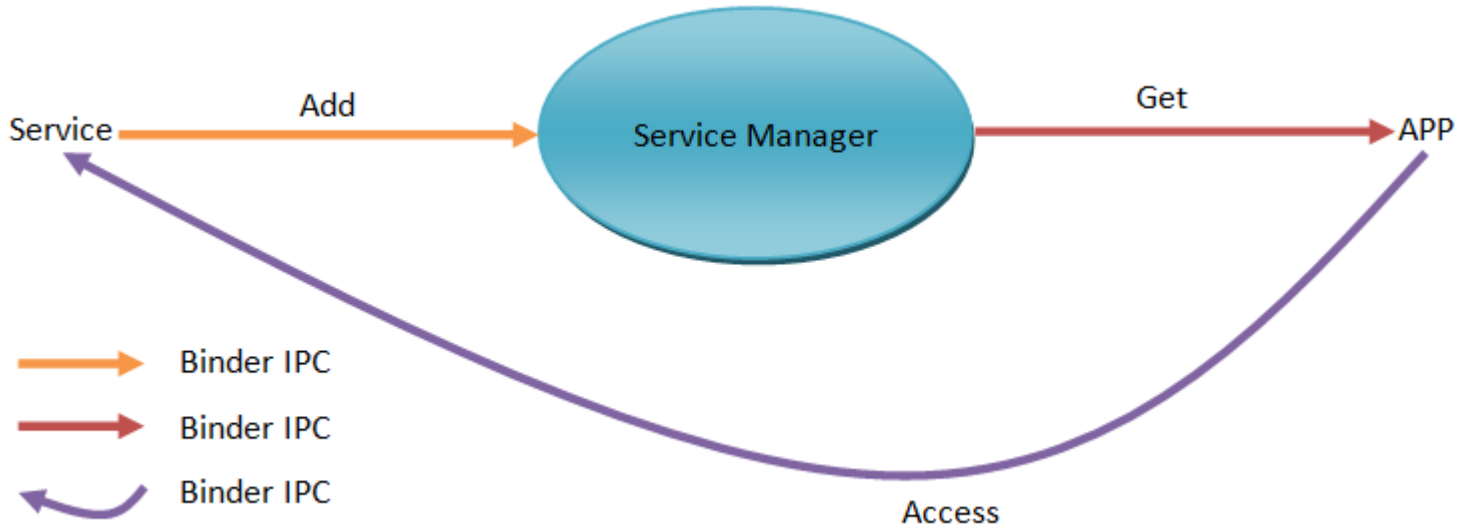
- 按開發語言划分
  - Java Runtime Framework
    - PackageManagerService
    - ActivityManagerService
    - WindowManagerService
    - ……
  - Native Runtime Framework
    - MediaPlayerService
    - SurfaceFlinger
    - AudioFlinger
    - ……

Android 應用程式框架 (續)

- 按程式划分
  - System Server Process
    - PackageManagerService
    - ActivityManagerService
    - WindowManagerService
    - ……
  - Independent Process
    - SurfaceFlinger
    - MediaPlayerService
    - ……

Android 應用程式框架 (續)

- 服務註冊、 抓取和存取過程

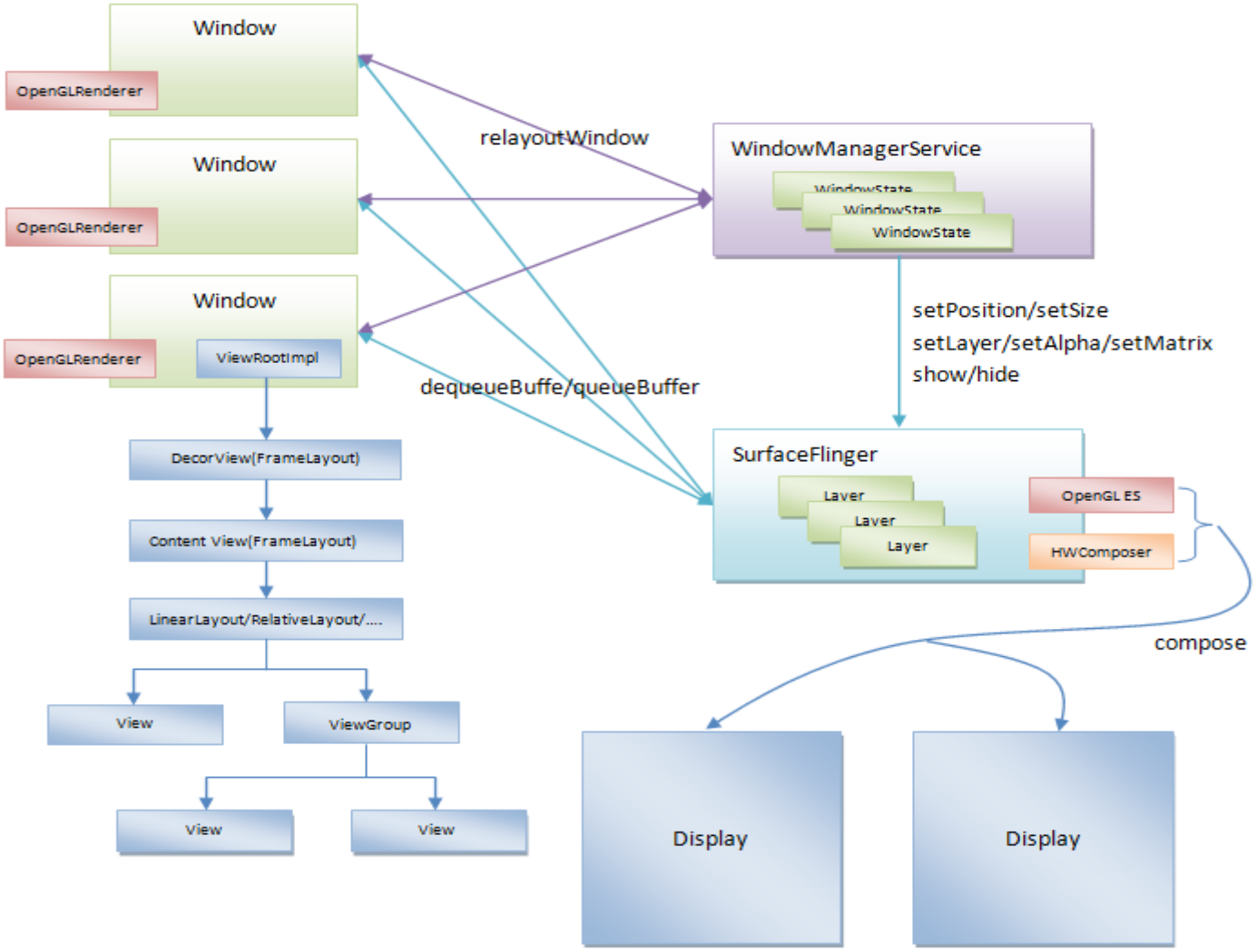


Android 用戶介面架構

- 視窗管理框架
  - Window
  - WindowManagerService
  - SurfaceFlinger
- 資源管理框架
  - AssetManager
  - Resources

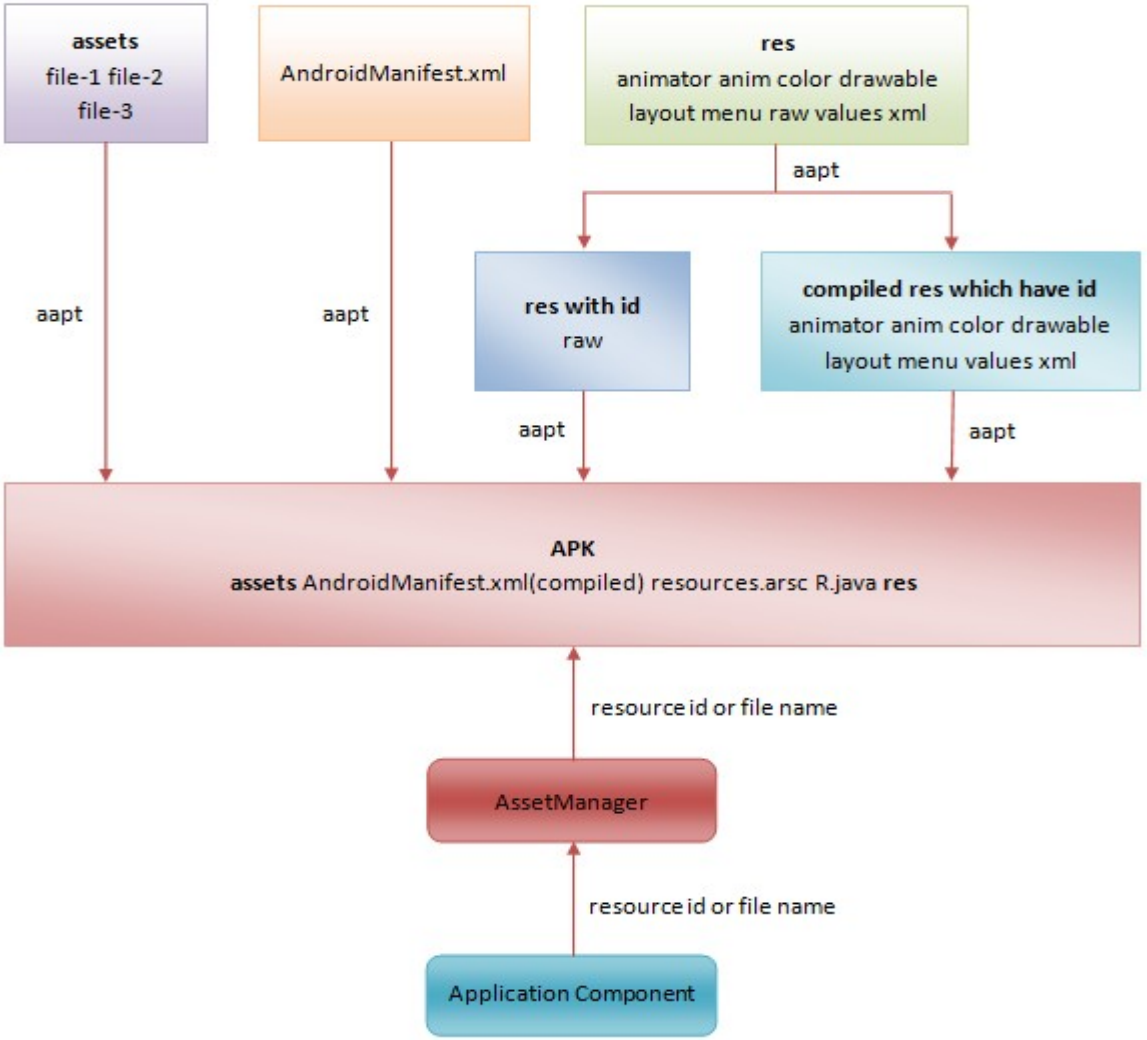
Android 用戶介面架構 (續)

- 視窗管理框架



Android 用戶介面架構 (續)

- 資源管理框架



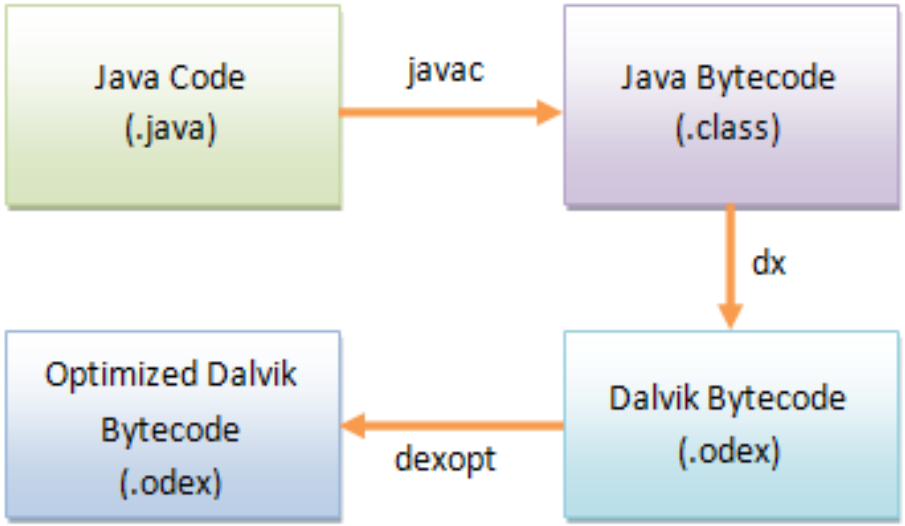
Dalvik 虛擬機

- Java 虛擬機與 Dalvik 虛擬機區別

	Java Virtual Machine	Dalvik Virtual Machine
Instruction Set	Java Bytecode (Stack Based)	Dalvik Bytecode (Register Based)
File Format	.class file (one file, one class)	.dex file (one file, many classes)

Dalvik 虛擬機 (續)

- Dex 檔案編譯和優化



Dalvik 虛擬機 (續)

- 記憶體管理
  - Java Object Heap
    - 大小受限， 16M/24M/32M/48M
  - Bitmap Memory(External Memroy) :
    - 大小計入 Java Object Heap
  - Native Heap
    - 大小不受限

Dalvik 虛擬機 (續)

- 垃圾收集 (GC)
  - Mark，使用 RootSet 記號物件參照
  - Sweep，回收沒有被參照的物件
- GingerBread 之前
  - Stop-the-word，也就是垃圾收集執行緒在執行的時候，其他的執行緒都停止
  - Full heap collection，也就是一次收集完全部的垃圾
  - 一次垃圾收集造成的程式中止時間通常都大於 100ms
- GingerBread 之后
  - Cocurrent，也就是大多數情況下，垃圾收集執行緒與其他執行緒是并發執行的
  - Partial collection，也就是一次可能只收集一部分垃圾
  - 一次垃圾收集造成的程式中止時間通常都小於 5ms

Dalvik 虛擬機 (續)

- 即時編譯 (JIT)
  - 從 2.2 開始支援 JIT，并且是可選的，編譯時通過 WITH_JIT 巨集進行控制
  - 基於執行路徑 (Executing Path) 對熱門的程式碼片斷進行優化 (Trace JIT)，傳統的 Java 虛擬機以 Method 為單位進行優化 (Method JIT)
  - 可以利用執行時資訊進行激進優化，獲得比靜態編譯語言更高的效能
  - 實現原理：http://blog.reverberate.org/2012/12/hello-jit-world-joy-of-simple-jits.html

Dalvik 虛擬機 (續)

- Java 區域呼叫 (JNI)
  - 實現 Java 與 C/C++ 程式碼互調
  - 大部分 Java 介面的都是通過 JNI 呼叫 C/C++ 介面實現的
  - 提供有 NDK 進行 JNI 開發
- 程式和執行緒管理
  - 與 Linux 程式和執行緒一一對應
  - 通過 fork 系統呼叫建立程式
  - 通過 pthread 程式庫介面建立執行緒

Q&A

Thank You