

Android development



Sviluppo di Mobile Apps
sul sistema operativo di Google

◆ Agenda

◆ Giorni:

- ◆ Gio 14/04/2011
- ◆ Ven 15/04/2011
- ◆ Gio 21/04/2011
- ◆ Ven 22/04/2011

◆ Suddivisione:

- ◆ Mattina: teoria
- ◆ Pomeriggio: pratica

◆ Chi siamo

◆ **Federico De Gioannini**

🤖 federico.degioannini@xenialab.it

◆ **Cristian Di Sclafani**

🤖 christian.disclafani@xenialab.it

Introduzione



Hello Android




◆ Storia recente del mobile

Quando?	Cosa?
nov-92	Uscita del primo SmartPhone: IBM Simon (concept product)
ott-96	Primo Nokia Communicator (Nokia 9000)
mar-97	Ericsson GS88 (concept product, primo device descritto come SmartPhone)
nov-00	Ericsson R380, primo SmartPhone con Symbian OS
gen-02	Microsoft annuncia i Windows SmartPhone
feb-02	RIM rilascia il primo BlackBerry
giu-07	Venduto il primo Apple iPhone
nov-07	La OHA annuncia la presentazione di Android
lug-08	Nasce l'AppStore di Apple
set-08	Esce la versione 1.0 di Android
ott-08	Esce il primo Android Phone: HTC Dream
mar-10	Esce l'iPad Apple
giu-10	Esce l'iPhone 4
set-10	Nokia N8 è il primo SmartPhone con Symbian^3
feb-11	Nokia annuncia l'utilizzo di Windows Phone 7 per gli SmartPhone di punta
mar-11	Esce l'iPad2 Apple

◆ Storia di Android

Versione	Data di uscita	Nome
1.0	23 Set 2008	
1.1	09 Feb 2009	
1.5	30 Apr 2009	Cupcake (tortino)
1.6	15 Set 2009	Donut (ciambellina)
2.0/2.1	26 Ott 2009	Éclair (bigne)
2.2	20 Mag 2010	Froyo (Frozen Yogurt)
2.3	06 Dic 2010	Gingerbread (pan di zenzero)
3.0	22 Feb 2011	Honeycomb (miele in favi)
3.1??		Ice Cream Sandwich



◆ Che cos'è Android

- ◆ Android è un sistema operativo Open Source per Mobile sviluppato da Google.
- ◆ Android partecipa alla Open Handset Alliance che è composta da circa 80 tra produttori e società telefoniche, ... (Acer, HTC, LG, Motorola, Samsung, Toshiba, Intel, Garmin, Sony Ericsson, Toshiba, Vodafone, Telecom Italia, T-Mobile ...)
- ◆ La base del sistema è Linux (Android 2.2 è basato su Linux 2.6.32).
- ◆ Il codice sorgente è:
 - ◆  C, C++
 - ◆  Java
 - ◆  XML

◆ Che cos'è un' Android App

- ◆ Ogni app Android è un processo interno a una istanza di VM (Dalvik)
 - 🤖 Dalvik è strutturato in modo tale che su uno stesso device possano coesistere diverse istanze in contemporanea
 - 🤖 Dalvik basso utilizzo della memoria (utilizzabile su macchine molto poco potenti)
 - 🤖 Dalvik utilizza classi Java compilate come .dex file
- ◆ L'ambiente di sviluppo “*consigliato*” è **Eclipse**, opportunamente munito di specifico SDK (contiene tools di debug e un valido simulatore)
- ◆ Per pubblicare app sul market :
 - 🤖 occorre pagare un contributo annuale di 25\$
 - 🤖 Processo molto semplice: upload dell'app, senza filtri da parte di Google (diametralmente opposto alle politiche di Apple e Microsoft)

◆ Dalvik VM

- ◆ Il kernel linux rappresenta un primo layer di astrazione fra l'HW e il SW. Al di sopra di questo layer si collocano le istanze Dalvik VM
- ◆ Diversamente dalle consuete Java VM, Dalvik utilizza una struttura a registri piuttosto che una struttura basata su stack
 - ◆  ↑ Utilizzo di un minor numero di istruzioni per caricare dati
 - ◆  ↓ Istruzioni più lunghe
- ◆ Dalvik presenta alcune caratteristiche peculiari per l'utilizzo su mobile:
- ◆ **Zygote:** le varie istanze di VM devono avere un avvio molto veloce. Android usa Zygote per consentire la condivisione di librerie e strutture in sola lettura per ottimizzare la condivisione di risorse attive

◆ Dalvik VM: nota di colore

◆ Dalvíkurbyggð

🤖 Il nome Dalvik deriva dal villaggio di pescatori **Dalvíkurbyggð** di cui la famiglia di Bornstein è originaria.

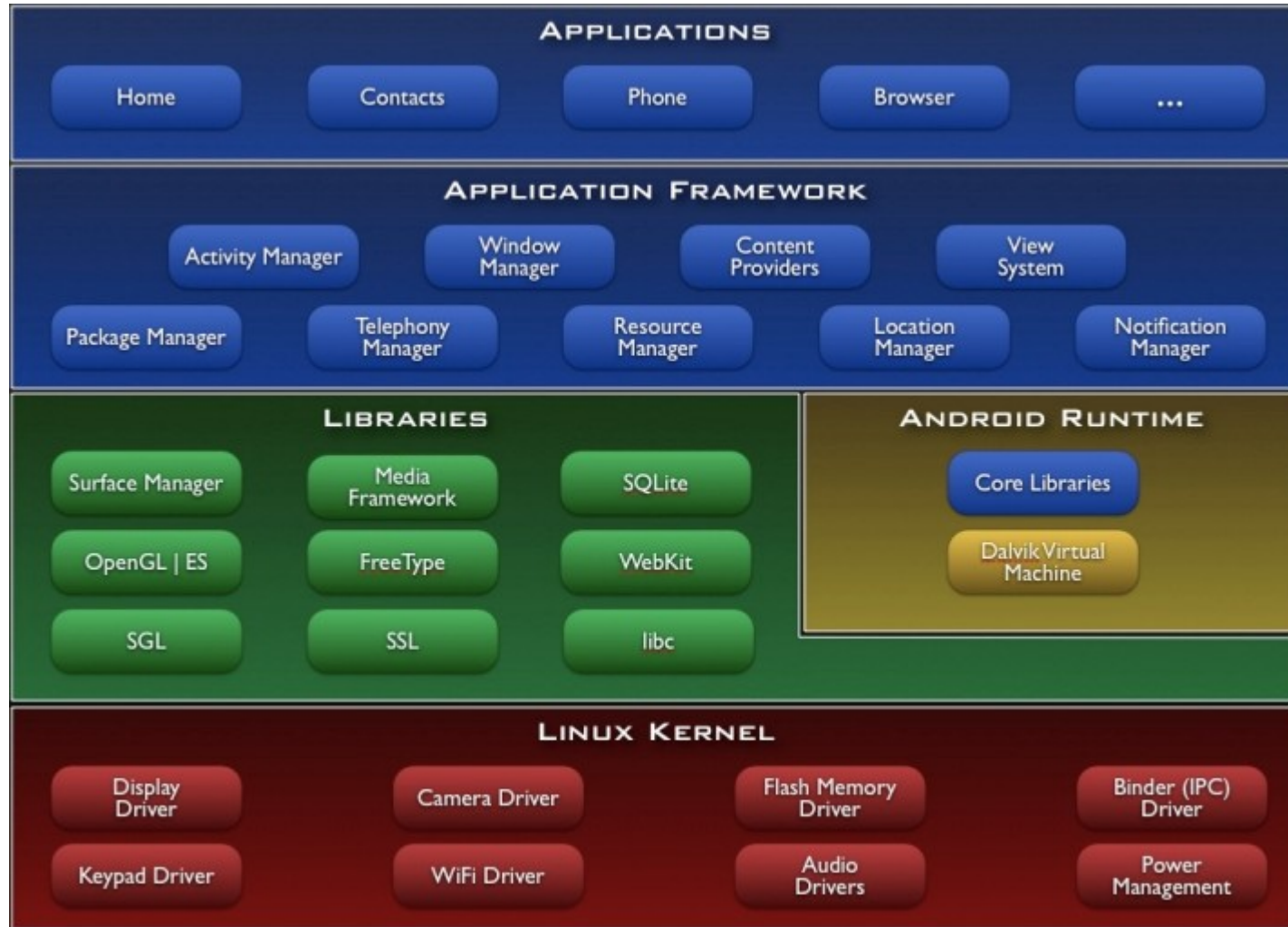


◆ Application Framework

- ◆ Architettura studiata per il riutilizzo dei componenti: **ogni applicazione può PUBBLICARE le proprie funzioni**. Queste funzioni possono essere utilizzate da altre App

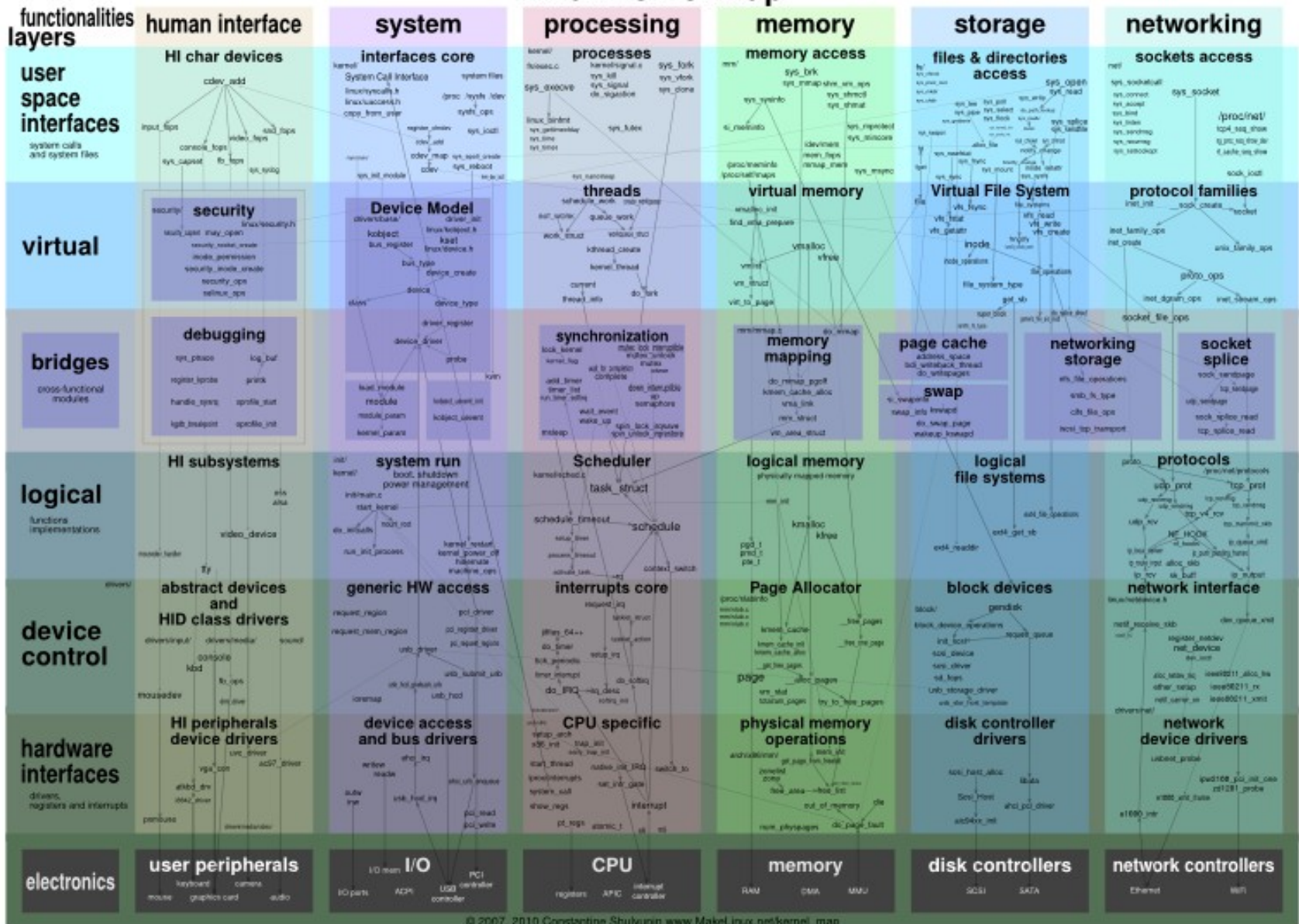
- ◆ Android mette a disposizione di tutte le App una serie di servizi di base, fra i quali:
 - 🤖 **View**: strutture grafiche di base utilizzate nell'interfaccia utente
 - 🤖 **Content Provider**: servizio che consente la condivisione di dati
 - 🤖 **Notification Manager**: servizio per la gestione delle notifiche
 - 🤖 **Activity manager**: gestione del ciclo di vita delle App

◆ Architettura Android (I)



◆ Architettura Android (II)

Linux kernel map



◆ Concetti di base delle App (I)

- ◆ Android è un OS Linux multiuser, nel quale **ogni App è un Utente**
- ◆ Android assegna ad ogni App un identificativo univoco. I permessi relativi ai files delle App sono settati in modo tale da essere accessibili solo all'App stessa
- ◆ Ogni App viene eseguita su una propria VM (isolamento dei processi)
- ◆ Ogni App esegue i propri processi (Linux). Ogni processo viene terminato quando non è più necessario o quando il sistema necessita memoria

Least privilege: ogni App accede solamente al proprio spazio.

◆ Concetti di base delle App (II)

Il principio del **Least privilege** però non è assoluto. Ci sono diverse strategie per condividere dati fra le applicazioni e per accedere ai servizi di sistema.

- ◆ App con lo stesso ID condividono dati, processi e VM
- ◆ Un'App può accedere alle risorse di sistema richiedendo il permesso in fase di installazione. In questo caso l'utente deve **SEMPRE** confermare

Sviluppare per Android



◆ L'ambiente di sviluppo

- ◆ Google mette a disposizione di tutti gli sviluppatori almeno due ambienti di sviluppo: **AppInventor** (ne parleremo in seguito) e **Android SDK**.

<http://developer.android.com/sdk/>

- ◆ In questa sezione discuteremo dell'installazione di **Android SDK**.

◆ Prerequisiti Android SDK

◆ Software:

OS:

Windows (XP, Vista, Windows 7)

Mac OSX 10.5.8 o successivi

Linux (testata con Ubuntu 8.04 o successive, con GNU C (glibc) 2.7 o successive, ...)

Eclipse IDE:

Eclipse 3.5 (Galileo) o successivo

Plugin Eclipse JDT

<http://www.eclipse.org/downloads/>

- ◆ Hardware: in generale è richiesto solamente di avere sufficiente spazio su disco per installare i vari pacchetti

◆ Installazione ambiente di sviluppo

- ◆ Scaricare e installare Android SDK

<http://developer.android.com/sdk/index.html>

 Dopo averlo scaricato, scompattarlo e collocarlo in un path noto

- ◆ Scaricare e installare il plugin di Eclipse Android Development Tools (ADT)

 Da Eclipse, selezionare **Help** → **Install New Software** e premere **Add**

 Nella ricerca inserire **ADT Plugin** come nome e <https://dl-ssl.google.com/android/eclipse/> come **URL**

 Proseguire e selezionare **Developer Tools**

 Proseguire fino alla fine del processo e riavviare Eclipse

◆ Configurare ADT

- ◆ Scaricare e installare Android SDK

<http://developer.android.com/sdk/index.html>

- ◆ Dopo aver riavviato Eclipse, per collegare l'ADT all'Android SDK:

- ◆ Entrare nelle preferenze (**Window → Preferences**) e selezionare **Android**

- ◆ Indicare la posizione dell'SDK (scaricato in precedenza) con **Browse....**

- ◆ Confermare e uscire

- ◆ Aggiornare l'ADT

- ◆ In Eclipse, **Help → Preferences** per controllare gli aggiornamenti

- ◆ Se ce ne sono, selezionare **Android DDMS, ADT, Android Hierarchy Viewer**

- ◆ Confermare e proseguire. Al termine riavviare Eclipse

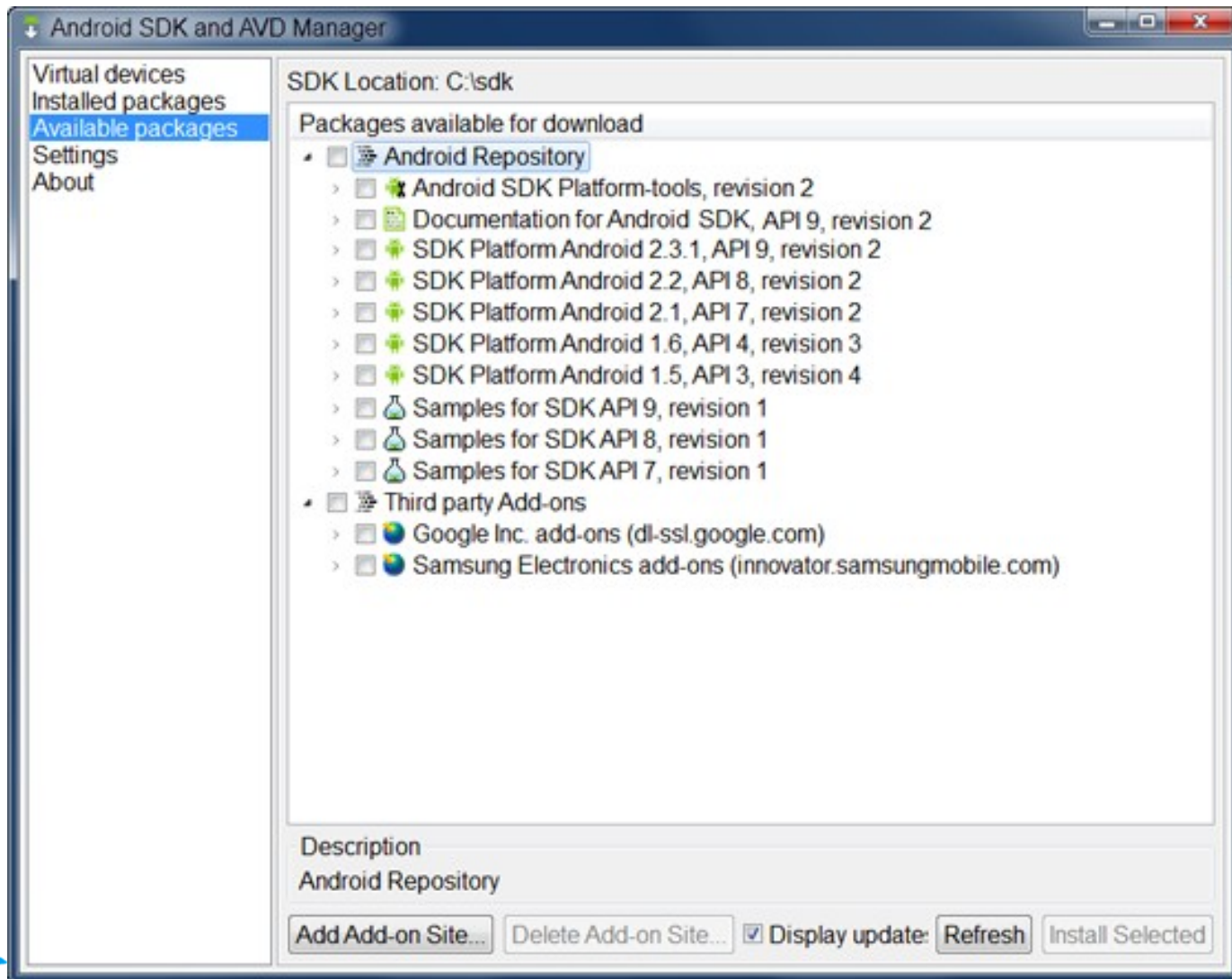
◆ Installazione ambiente di test (I)

Android è un sistema operativo Mobile con la capacità di essere installato su piattaforme con HW di diverso tipo. Per questo è necessario creare una serie di simulatori in grado di replicare il funzionamento dei diversi device.

A tal fine, sono disponibili diversi componenti di simulazione:

- ◆ da Eclipse, **Window → Android SDK e AVD manager**
- ◆ installare i componenti necessari alla simulazione degli ambienti di test che interessano (...che sono la maggior parte...)

◆ Installazione ambiente di test (II)



◆ ... e ora al lavoro!!

A questo punto siamo pronti per passare a lavorare su Eclipse per la prima App:

CorsoHello



◆ le Risorse di un'App

- ◆ Per Risorse si intende tutto quello che non è codice (file di varia natura) e che viene aggiunto dallo sviluppatore al progetto
- ◆ Per ogni risorsa presente nel progetto viene associato un ID per poterla identificare direttamente
- ◆ Le App possono caricare diverse risorse a seconda di diverse circostanze (diverse risorse per diverse risoluzioni o diverse lingue). Si utilizzano **Qualifiers**. I Qualifiers sono delle stringhe, inserite nel nome della cartella che contiene le diverse famiglie di risorse, che assumono valore diverso a seconda delle impostazioni del dispositivo. **A seconda delle impostazioni viene utilizzata una diversa famiglia di risorse.**

◆ I componenti di un'App (I)

Esistono quattro famiglie di componenti in un'App:

◆ **Activities:**

- 🤖 rappresenta la singola schermata dell'app con una propria interfaccia utente.
- 🤖 Sono indipendenti le une dalle altre.
- 🤖 Se pubbliche, possono essere richiamate da diverse App

◆ **Services:**

- 🤖 Indica un processo in background di durata medio-lunga (no interfaccia utente)

◆ I componenti di un'App (II)

◆ Content providers:

- 🤖 Gestisce i dati condivisi (dati in file system, SQLite, web...)
- 🤖 Attribuisce a diverse applicazioni diversi permessi sui dati

◆ Broadcast receivers:

- 🤖 Elemento che gestisce le notifiche di sistema
- 🤖 Non genera una interfaccia utente, ma può pubblicare le notifiche nella status bar o avviare un servizio in risposta alla notifica
- 🤖 Le notifiche di sistema possono essere generate dal sistema stesso o da App installate

◆ Utilizzo di processi esterni a un'App

- ◆ **Ogni applicazione può avviare un componente di un'altra applicazione** (aspetto presente – per ora – solo in Android)
- ◆ Anche se avviato da un'App, **un processo viene sempre eseguito nell'App a cui appartiene**. Non c'è un unico punto di accesso per un processo (no `main()`)
- ◆ L'accesso a processi esterni non è diretto (isolamento dei processi + permessi delle App). Per attivare un processo esterno, un'App invia comunicazione (**intent**) al sistema Android, il quale avvia il processo richiesto.

◆ Attivare un componente

- ◆ Tre dei quattro componenti di un'App (**Activities, Services e Broadcast Receivers**) vengono attivati da una chiamata di **Intent**
 - 🤖 per Activities e Services, Intent definisce un'azione da eseguire e opzionalmente un URI con i dati da utilizzare per l'azione (+ opzioni). Se è richiesta una risposta, anche questa viene fornita tramite Intent.
 - 🤖 Per Broadcast Receivers, l'Intent contiene solamente la notifica
- ◆ I **Content Providers** invece sono attivati da un **Content Resolver**.
 - 🤖 Il Content Resolver gestisce in toto la transazione fra l'App e il contenuto richiesto.
 - 🤖 Rappresenta un ulteriore livello di astrazione fra l'App e i contenuti, aumentando la sicurezza del sistema

◆ il Manifest

- ◆ L'**AndroidManifest.xml** è il file nel quale ogni App deve dichiarare i propri componenti.
- ◆ Oltre a questo il Manifest deve contenere:
 - 🤖 Dichiarazione dei permessi necessari all'App per accedere alle risorse richieste
 - 🤖 Dichiarazione della famiglia di API utilizzate
 - 🤖 Dichiarazione dei requisiti HW e SW
 - 🤖 Dichiarazione delle librerie richieste
 - 🤖 Altro....