Lezione 3

Programmazione Android

- Struttura di un'applicazione Android
  - In sviluppo
  - In deployment
  - In esecuzione
- Il sistema delle risorse
  - Architettura generale
  - Risorse alternative
  - Qualche esempio
- AndroidManifest.xml
- (Laboratorio?)

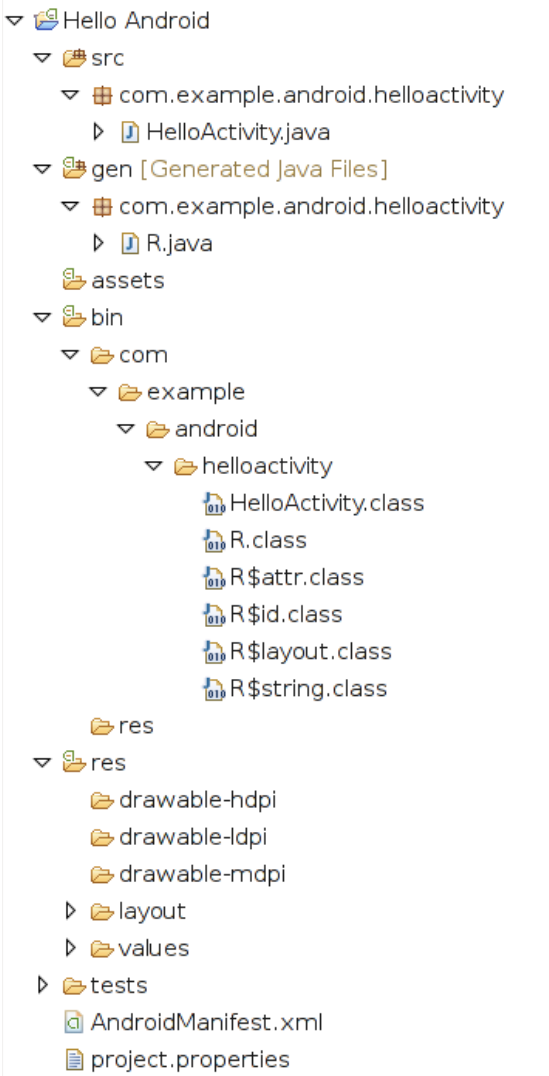
Struttura di un'applicazione Android

Struttura di un'applicazione

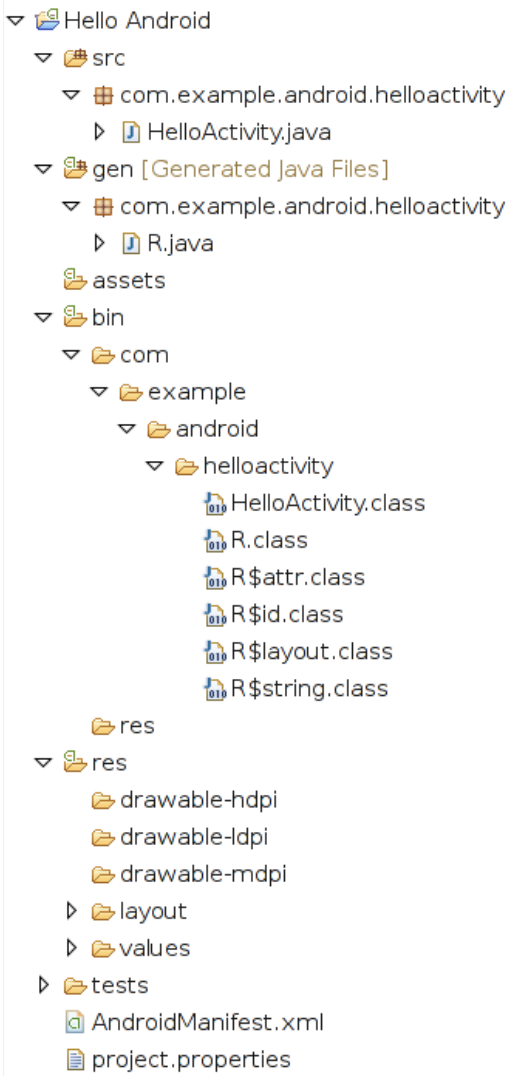
- Un'applicazione può assumere diverse forme nel corso della sua vita
  - In **sviluppo**: layout su disco (progetto)
  - In **deployment**: formato dei file .apk
  - In **esecuzione**: struttura in memoria
- Le varie forme sono legate da tre processi
  - **Build**: sorgente → .apk
  - **Deploy**: .apk (sul Market/ecc.) → .apk (sul device)
  - **Run**: .apk → processo in memoria

Struttura di un progetto

- Un'applicazione in sviluppo è chiamata **progetto**
- Il wizard di creazione di un nuovo progetto Android dell'ADT crea lo scheletro per noi
- Solo alcune directory sono di interesse per lo sviluppatore
  - Altre vengono generate automaticamente



Struttura di un progetto



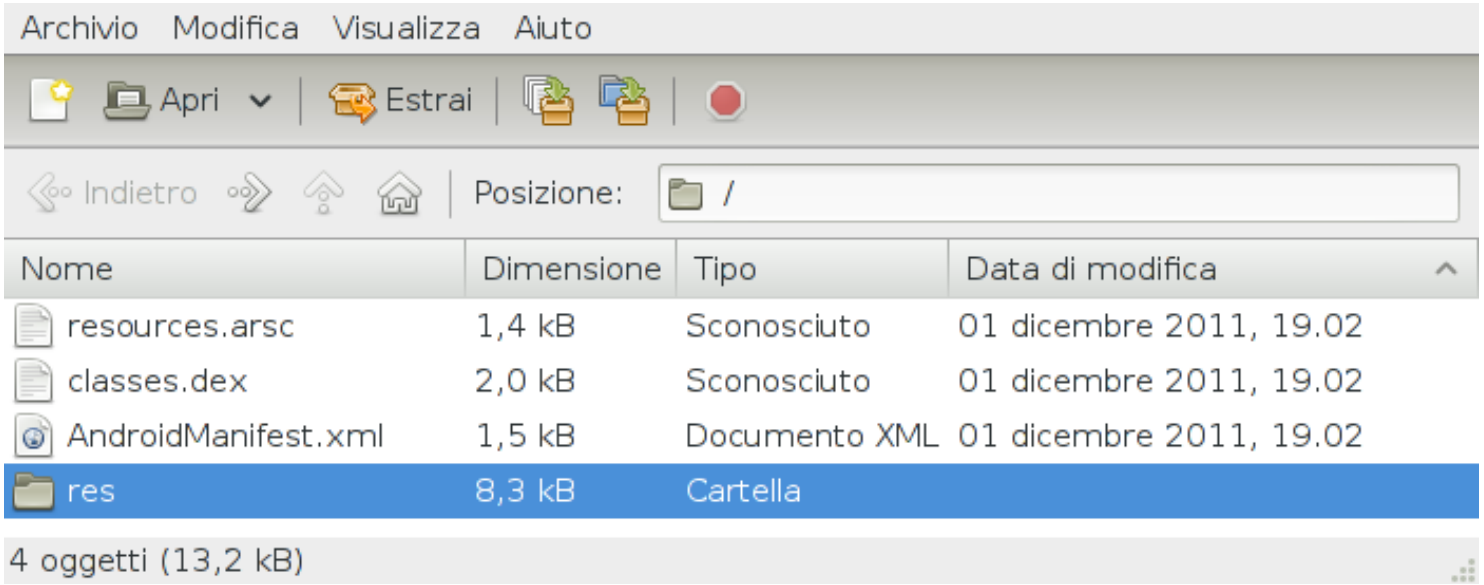
- **src** – sorgenti (.java, a volte .aidl o altri linguaggi)
- **gen** – sorgenti generati automaticamente (R.java)
- **assets** – file arbitrari, aggiunti al .apk
- **bin** – risultato della compilazione (risorse, .dex, .apk, …)
- **res** – risorse note al runtime Android
  - **anim** – animazioni
  - **color** – colori
  - **drawable** – immagini raster
  - **layout** – descrizioni del layout della UI
  - **menu** – menu usati dall'applicazione
  - **raw** – file arbitrari (alternativa ad assets)
  - **values** – costanti (stringa, interi, array, ecc.)
  - **xml** – file XML arbitrari (incluse configurazioni)
- **libs** – librerie native custom
- **AndroidManifest.xml** – metadati sull'applicazione
- Altri **.properties**, **.cfg**, **.xml** – configurazioni varie

Struttura di un progetto

- La struttura che abbiamo visto è quella tipica di un'**applicazione**
- Esistono altre due forme di progetto Android:
  - Una **libreria** contiene componenti destinati ad essere usati da altre applicazioni
    - In questo modo, i membri di una famiglia di applicazioni correlate possono condividere componenti
    - Ogni libreria può essere usata da varie applicazioni; ogni applicazione può usare varie librerie
  - Un **progetto test** contiene codice usato per il testing di un'altra applicazione
- Li vedremo a tempo debito!

Contenuti di un .apk

- Il **build** di un'applicazione Android produce un file in formato .apk
- Un .apk è una specializzazione di un .jar
  - Un .jar è una specializzazione di uno .zip



Contenuti di un .apk

- **resources.arsc** – file binario contenente la tabella che mappa ID a risorse
- **classes.dex** – tutti i .class dell'applicazione, convertiti in DEX, e unificati in un solo file
- **AndroidManifest.xml** – manifesto dell'applicazione
- **res/*** - file delle risorse
- **META-INF/*** - contiene i certificati pubblici (chiavi)
  - (solo per le applicazioni firmate)

Contenuti di un .apk

- **META-INF/MANIFEST.MF** contiene
  - Informazioni di versionamento
  - Un *digest* (checksum) dei contenuti di ciascun file
- **META-INF/CERT.*** – certificati RSA
- È necessario creare un proprio certificato per il deploy!
  - Uno di debug viene creato da ADT

```
Manifest-Version: 1.0
Created-By: 1.0 (Android)

Name: res/layout/main.xml
SHA1-Digest: 8syRBinqeui2pRMGibKHakq1Lew=

Name: AndroidManifest.xml
SHA1-Digest: q3nOaBAUplDlAalDCNaDnPQ0ELU=

...
```

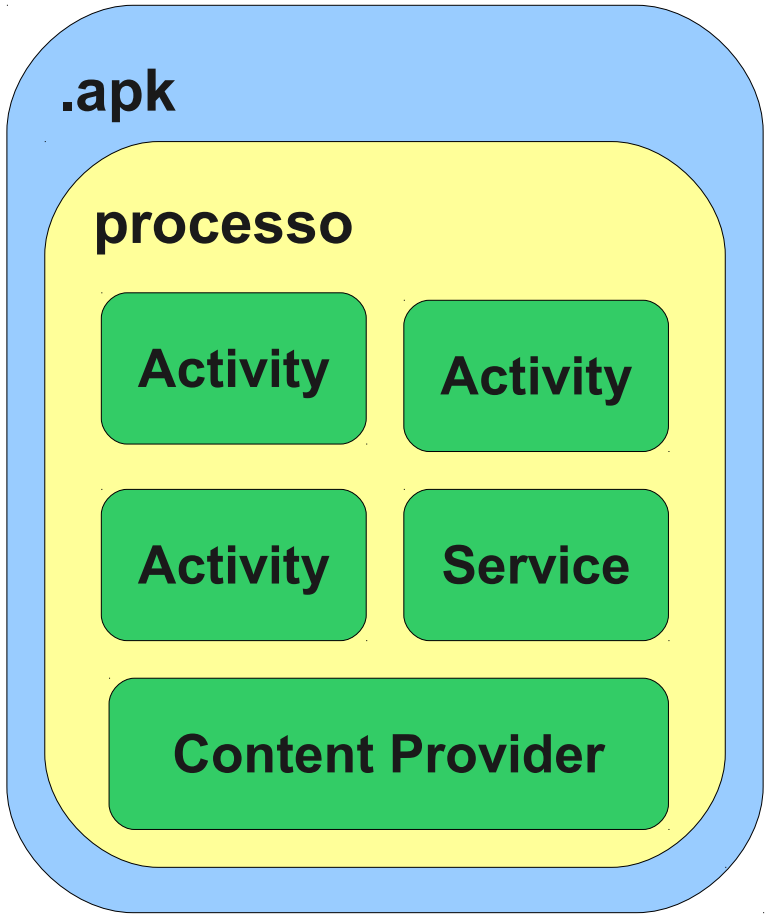
Contenuti di un .apk

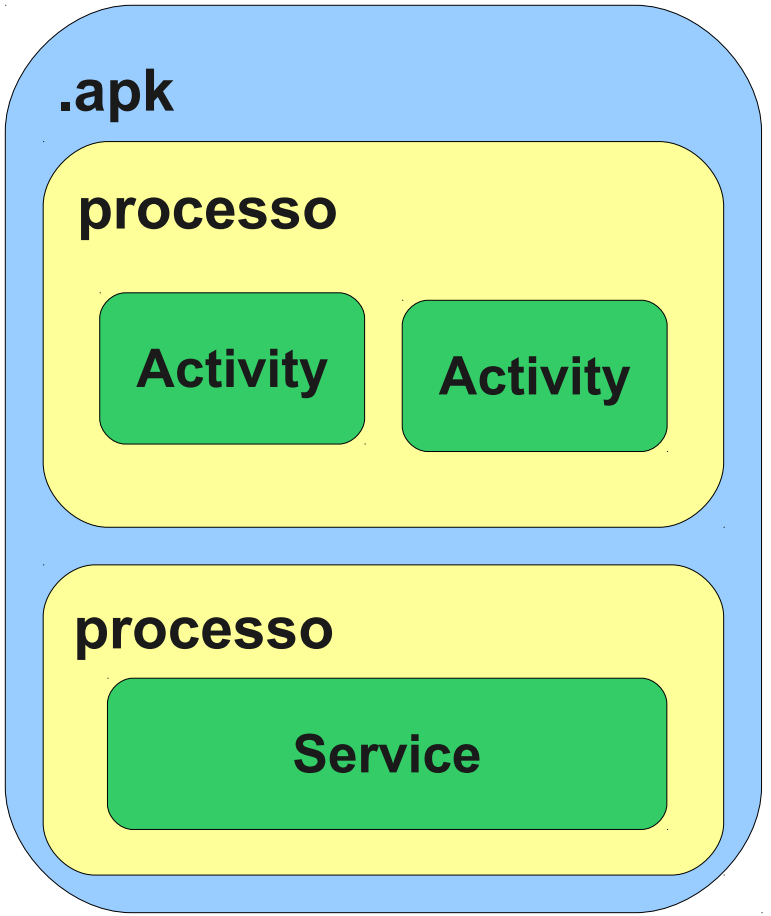
- Un .apk è dunque un archivio contenente tutti i componenti di un'applicazione
  - **Auto-descrittivo** (grazie ai manifesti)
  - **Compatto** (grazie alla compressione)
  - **Affidabile** (grazie alla firma digitale)
    - Non è possibile aprire un .apk, sostituire alcuni componenti e re-impacchettarlo – la firma non sarebbe valida!
  - Facilmente **distribuibile** (un file unico)
  - Facilmente **installabile** (niente “installer”)
    - In /sys/app (pre-installed) o /data/app (user-installed)

Struttura in memoria

- Una volta caricata in memoria, una applicazione è distinta in **componenti**
- Di norma, 1 app = 1 processo = 1 VM = 1 thread
  - Possibili variazioni: app multi-threaded, più app “amiche” in un solo processo, ecc.
- Il flusso di lavoro dell'utente (*task*) è fatto però spesso di componenti appartenenti ad applicazioni diverse, eseguiti da processi diversi
- Android **incoraggia** la condivisione (sicura) fra applicazioni

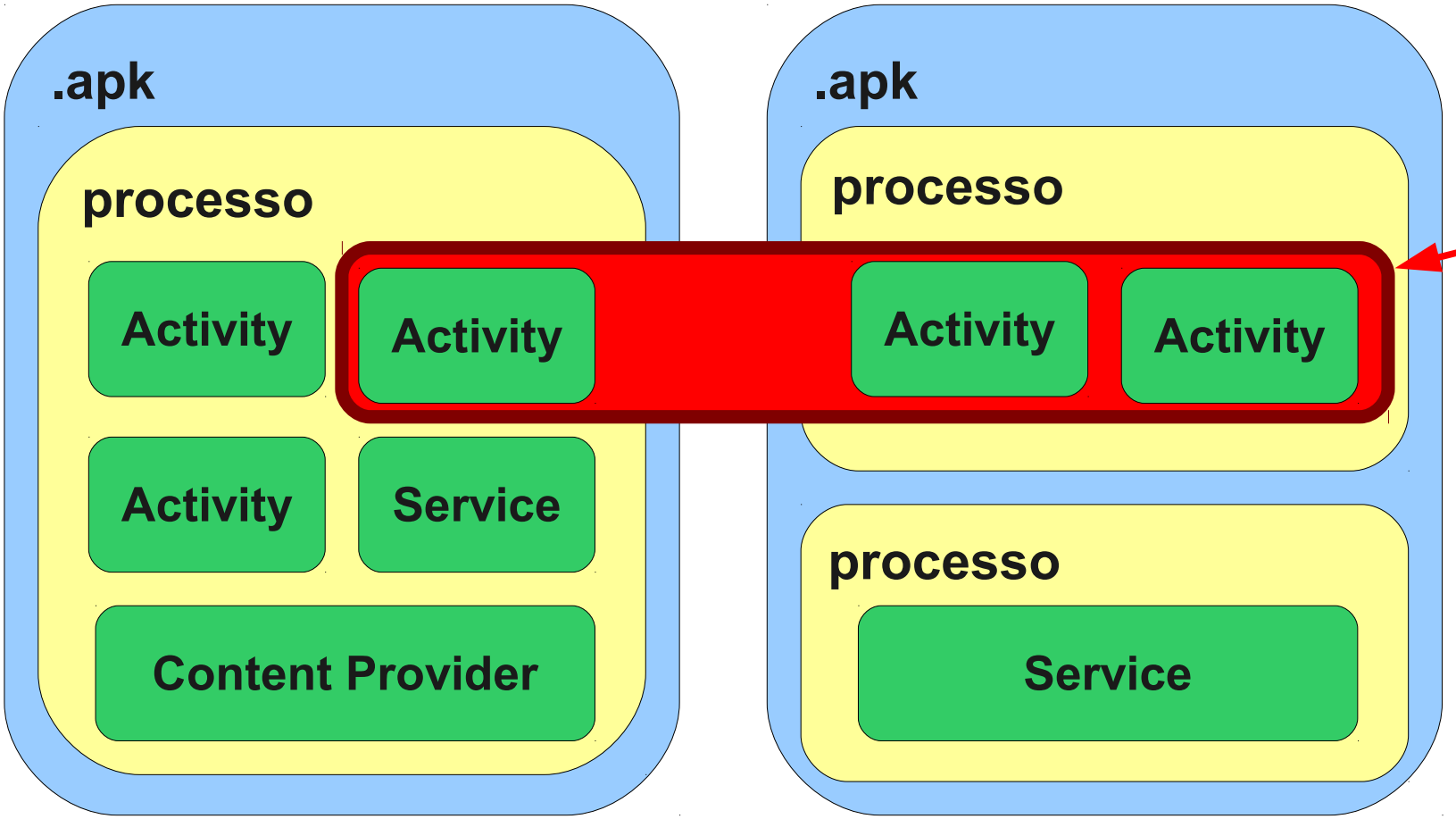
Struttura in memoria





Classi del framework

Struttura in memoria



Struttura in memoria

- Due visioni diverse
  - I **programmatori** percepiscono come “app” un .apk
  - Gli **utenti** percepiscono come “app” un task
- Per garantire un'esperienza utente gradevole, occorre che l'integrazione sia *seamless*
  - Assenza di cesure visuali / comportamentali
  - Uso di **stili**, **temi**, **stock icon**, style guide
  - Consistenza *fra* applicazioni più importante della consistenza *dentro* le applicazioni (o fra piattaforme)

Esempio (negativo)

Il sistema delle risorse

Due tipi di risorse

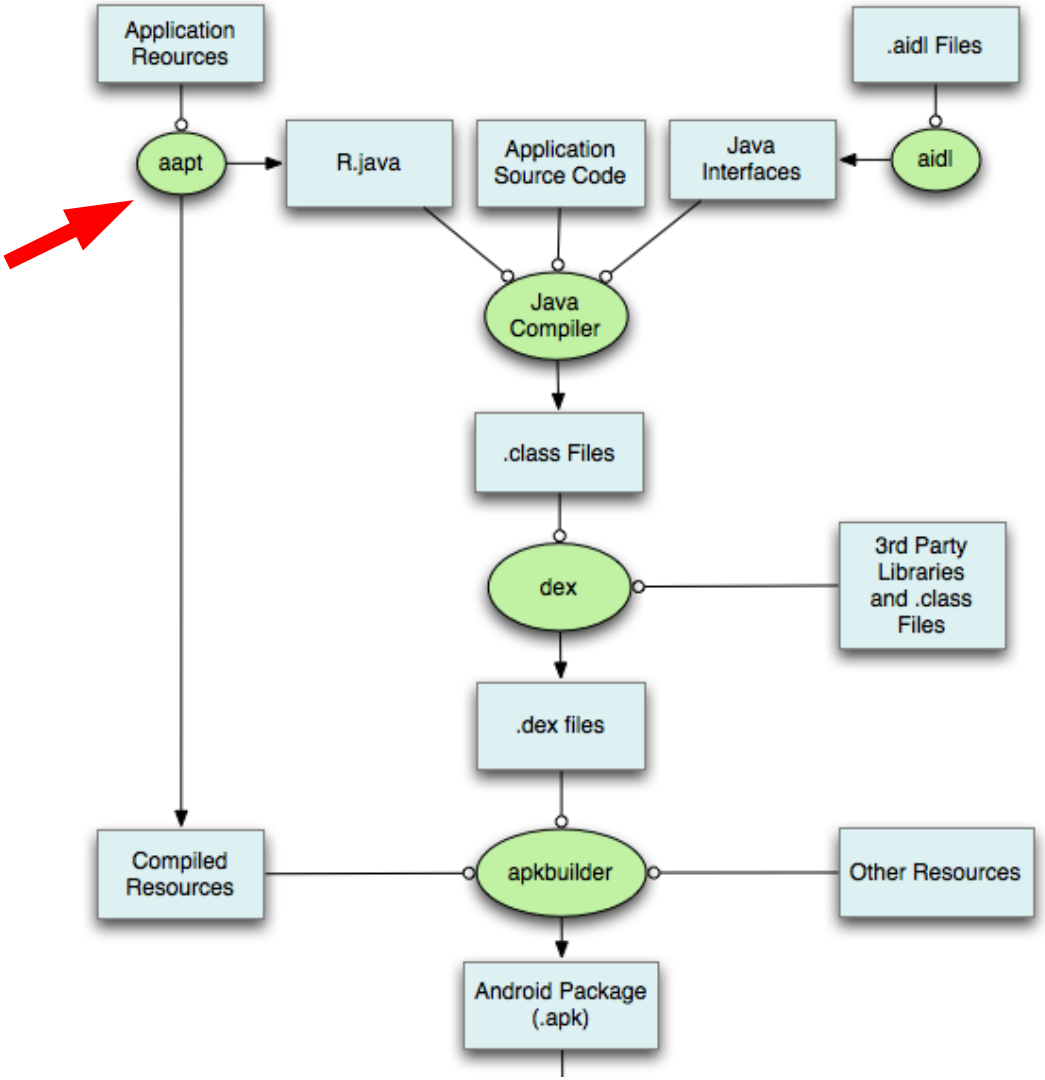
- Le risorse (in generale) sono dati usati dalle applicazioni
- Android distingue due casi:
  - Dati la cui struttura è nota al framework → **resources**
  - Altri dati → **assets**
- Gli asset vengono semplicemente aggiunti all'.apk, e acceduti come normali file
- Le risorse sono invece gestite dal **resource manager** in maniera articolata

Compilazione delle risorse

- Durante il processo di build, uno dei tool (**aapt**) processa le risorse come segue:
  - Converte i file XML in res/ in formato binario
  - Genera una tabella di corrispondenza fra ID numerici e offset nel formato binario
  - Genera una classe Java (in gen/R.java) contenente tanti campi (final static) quante sono le risorse
    - Nome del campo = nome della risorsa (public static final)
    - Inner class = tipo della risorsa (public static final)
    - Valore = ID numerico della risorsa
  - Compila la classe R (public final), che entra nello spazio dei nomi del progetto

Compilazione delle risorse

- Il processo di build di un'applicazione Android è piuttosto complesso
- Eclipse nasconde (quasi) tutto
- È sempre possibile intervenire manualmente se necessario
  - Toolchain da riga di comando



Default e alternative

- Per ogni risorsa, Android consente di definire un *default* e un numero (anche grande) di *alternative*
- Quando si accede a una risorsa a run-time, viene selezionata l'alternativa più specifica
  - In base all'ambiente o alle condizioni correnti
- Nel codice, si userà sempre un **identificativo di risorsa**; il binding alla risorsa vera è fatto dal framework
  - La famosa classe **R** generata automaticamente

Tipi di risorse

- Animation
- Color state list
- Drawable
- Menu
- String
- Style
- Bool
- Color
- Dimension
- ID
- Integer
- Integer array
- Typed array
- Raw*
- XML*

* Acceduti tramite API speciali per leggere i dati grezzi

Esempio: definizione

- In res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, HelloAndroidActivity!</string>
    <string name="app_name">Hello Android</string>
</resources>
```

- In gen/*package*/R.java

```
/* AUTO-GENERATED FILE.  DO NOT MODIFY. */
public final class R {
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

Esempio: uso

- In res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout … >
    <TextView    android:layout_width="fill_parent"
                 android:layout_height="wrap_content"
                 android:text="@string/hello" />
</LinearLayout>
```

- In HelloAndroidActivity.java

```
public class HelloAndroidActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Accesso alle risorse

- Più formalmente, si può accedere a una risorsa conoscendo:
  - Il suo **package** (opzionale per il “nostro” package)
  - Il tuo **tipo**
    - Per risorse complesse, dato dalla sottodirectory in res/
    - Per risorse semplici (contenute in un <resources>), dato dal loro tag
  - Il suo **nome**
    - Per risorse complesse, dato dal basename del file
    - Per risorse semplici (contenute in un <resources>), dato dall'attributo android:name di un nodo
- In un file XML: @[*package*:]*tipo*/*nome*
- In Java: [*package*.]R.*tipo*.*nome*

Accesso alle risorse

- ATTENZIONE!
- Il valore di R.*tipo*.*nome* è l'**id numerico** della risorsa, non il suo **valore**
- In particolare, gli ID sono di tipo **int** – non String, Image, etc.
- Molti metodi di classi dei vari framework richiedono un ID
  - es. setContentView()
- Context.getResources() restituisce l'oggetto **Resources** del nostro package

Accesso alle risorse
Metodi di Resources

Context è una superclasse di Activity, quindi spesso basta chiamare getResources()

- boolean getBoolean(int id)
- int getInt(int id)
- float getDimension(int id)
- int [] getIntArray(int id)
- Drawable getDrawable(int id)
- ecc.

- InputStream openRawResource(int id)
- XmlResourceParser getXml(int id)
- AssetManager getAssets()

Accesso alle risorse
Metodi di AssetManager

- String [] list(String path)
- AssetFileDescriptor openFd(String filename)
- InputStream open(String filename)
- XmlResourceParser openXmlResourceParser(String filename)
- ecc.

- Simile a un accesso a file, ma...
  - Le risorse sono compresse, vengono decompresse
  - Il sistema può mantenere una cache (accesso più rapido)

Risorse alternative

- In moltissimi casi, le risorse devono essere **adattate** all'ambiente
- Alcuni esempi
  - Stringhe per le varie lingue (localizzazione)
  - Colori adattati al paese (es.: bandiere)
  - Layout diversi in base all'*orientazione* del dispositivo :)
  - Icone diverse per carrier diversi (es.: logo)
  - Immagini a risoluzioni diverse
  - ecc.

Risorse alternative

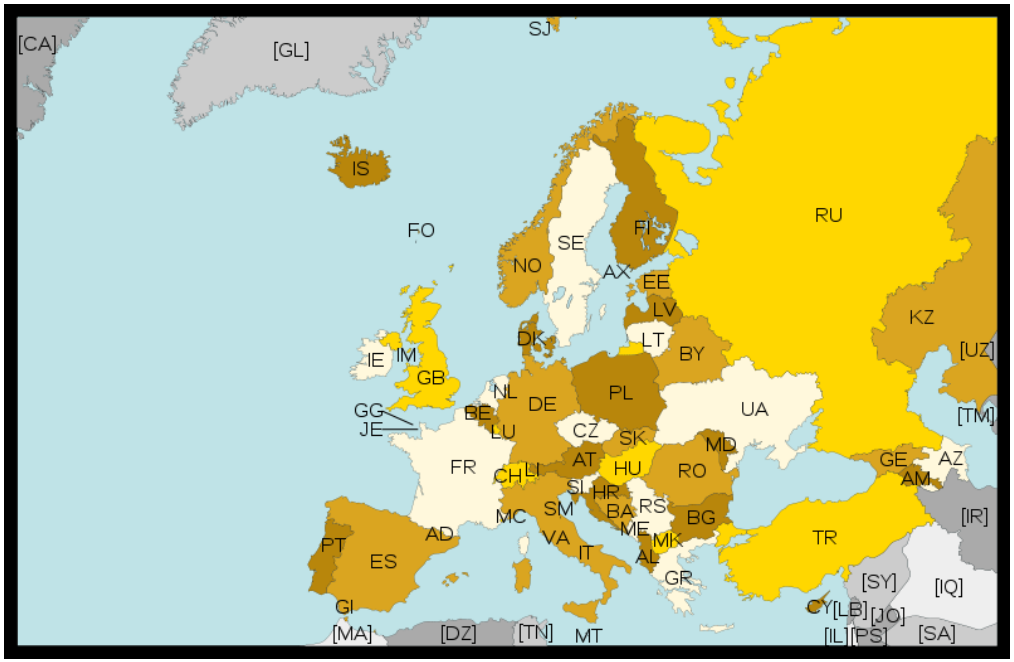
- Si possono definire risorse specializzate tramite l'uso di **qualificatori**
- Nella directory res/, si affiancano alle sotto-directory che già conosciamo (che sono i *default*) delle directory **qualificate**
- Forma: res/*tipo*-***qualificatori***/*file*
- Esempio:
  res/drawable/icon.png
  res/drawable-ldpi/icon.png
  res/drawable-hdpi/icon.png

Qualificatori – SIM e rete

- Mobile Country Code / Mobile Network Code
- Formato: -mcc*MCC,* -mnc*MNC*
- -mcc222 = Italia, -mcc222-mnc010 = Vodafone IT

MCC	MNC	Marca	Operatore	Note
222	1	TIM	Telecom Italia SpA	
222	2	Elsacom		Ritirato
222	10	Vodafone	Vodafone Omnitel N.V.	
222	30	RFI	Rete Ferroviaria Italiana	
222	77	IPSE 2000		Ritirato
222	88	Wind	Wind Telecomunicazioni SpA	
222	98	Blu		Ritirato
222	99	3 Italia	Hutchison 3G	

Qualificatori – lingua e regione



- Lingua e regione
- Formato: -*lingua*-r*regione*
- -it = italiano,
  -it-rCH = Canton Ticino,
  -en-rUS = Inglese, USA
  -es-rUS = Spagnolo, USA
- Soliti codici ISO-639 e ISO-3166:
  - Lingue: it, fr, en, es, …
  - Regioni: IT, FR, CA, GB, US, ES, …
  - Le Regioni coincidono (quasi) sempre con i TLD

Qualificatori - schermo

- Dimensione minima dello schermo
  - Lungo l'asse minore, quale che sia
- Formato: -sw*N*dp
- -sw300dp = minimo 300 pixel lungo l'asse minore

- Dimensione dello schermo in larghezza
- Formato: -w*N*dp
- -w720dp = minimo 720 pixel in larghezza

Qualificatori – schermo

- Dimensione dello schermo in altezza
- Formato: -h*N*dp
- -h600dp = minimo 600 pixel in altezza

- Dimensione generale dello schermo
- Formato: -*dimensione*
- -small, -normal, -large, -xlarge

Qualificatori – schermo

- Aspetto dello schermo (rapporto w:h, approx)
- Formato: -*aspetto*
- -long (stile 16:9), -notlong (stile 4:3)

- Orientamento dello schermo
- Formato: -*orientamento*
- -port (portrait), -land (landscape)

Qualificatori – ambiente

- Dock mode: se il dispositivo è in un dock
- Formato: -*dockmode*
- -car (installato in dock in auto), -desk (sul tavolo)

- Night mode: se è giorno o notte
- Formato -*nightmode*
- -night (è notte), -notnight (è non-notte :-)

Qualificatori – schermo (ancora!)

- Densità del display
- Formato: -*densità*dpi
- Valori ammessi:
  - -ldpi = ~120dpi, -mdpi = ~160dpi, -hdpi = ~240dpi
  - -xdpi = ~320dpi (Retina-style)
  - -tvdpi = ~213dpi (densità delle TV)
  - -nodpi = pixel assoluti, bitmap da non scalare

- Tipo di touch screen
- Formato: -*tipotouch*
- -notouch (eek!), -stylus (penna), -finger (dito)

Qualificatori – input

- Tastiera
- Formato: -keys*tipo*
- -keysexposed, -keyshidden, -keyssoft
- Formato: -*tipohw*
- -nokeys, -qwerty, -12key

- Tasti “cursore”
- Formato: -nav*tipo*
- -navexposed, -navhidden
- Formato: -*tiponav*
- -nonav, -dpad, -trackball, -wheel

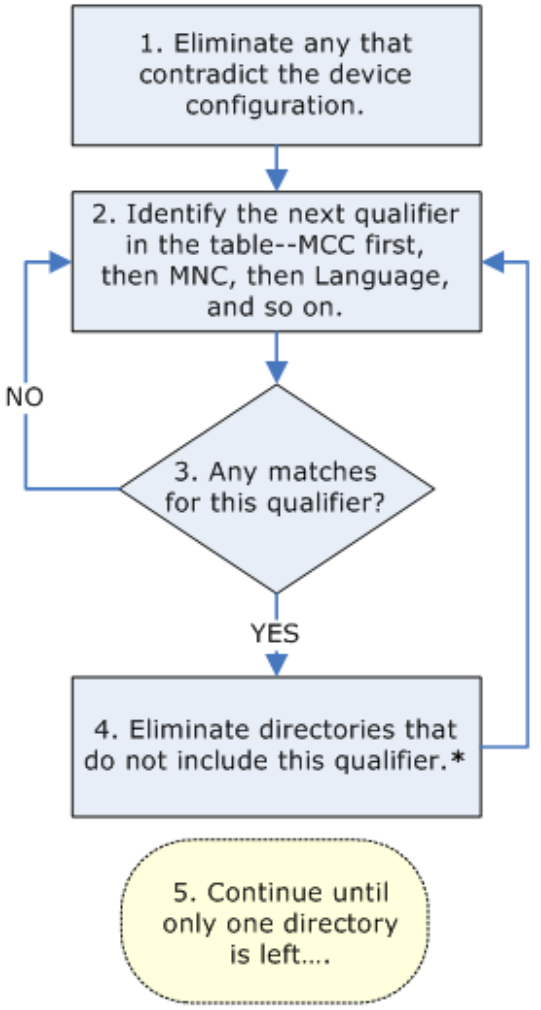
Qualificatori – versione OS

- Versione di Android (API level) corrente
- Formato: -v*versione*
- -v9 (Gingerbread), -v14 (Ice Cream Sandwich)

- Si possono indicare più qualificatori in sequenza
- Tutte le parti sono opzionali, ma **devono** essere nell'ordine in cui li abbiamo presentati!
- Es:
  - **res/drawable-de-rVA-night/button.png** → pulsante da usare in Vaticano la notte se la lingua dell'utente è il tedesco

Risorse alternative

- **A run-time**, il Resource Manager individua, per ogni risorsa, quale fra le tante alternative usare
- Usa l'algoritmo a lato per cercare il **match migliore**
- Ogni volta che la configurazione corrente cambia, il sistema **riavvia** l'Activity corrente con il nuovo insieme di risorse migliori
  - Salvo che l'Activity gestisca a mano...



Risorse alternative
Gestione dinamica

- In particolare, un'Activity può:
  - **Ignorare le modifiche**
    - Viene chiusa e riavviata (con le nuove risorse) dal sistema
  - **Salvare e ripristinare**
    - Si dichiara un oggetto che viene preparato dalla “vecchia” Activity e passato alla “nuova”, che lo usa per ripristinare uno stato transiente
  - **Gestire manualmente il cambiamento**
    - L'Activity **non** viene chiusa, ma viene chiamato un suo metodo passando i dettagli della nuova situazione