

Resources

Android è progettato per essere eseguito su un innumerevole insieme di dispositivi. In questo ambito ogni applicazione sviluppata dovrebbe attuare *feature* dinamiche, rendendo adattiva la *user interface* rispetto alle configurazioni dello schermo.

Come già accennato un'applicazione Android dovrebbe essere eseguita su una eterogenità di dispositivi, ognuno di essi con le proprie caratteristiche. Per riuscire nell'intento possono essere adeguati due approcci principali:

- **Soluzione tradizionale**, adottare tutte le molteplici alternative. Tuttavia questo si traduce in un abuso di costrutti condizionali, rendendo il codice di difficile lettura; inoltre occorre ricompilare il progetto ogni qualvolta dovesse presentarsi la necessità di variare il *Layout* oppure qualora aggiunti nuovi *package*
- **Soluzione Android**, separare il codice dalle *risorse*. Rispetto al paradigma descritto si usufruisce dell'approccio dichiarativo XML affinché possano essere definite le risorse

Prima di addentrarsi all'interno della composizione delle cartelle che caratterizzino un progetto è bene stabilire cosa sia una *risorsa*.

Definizione

Una *risorsa* è tutto ciò che non sia codice.

Pur di rispettare la definizione precedente è necessario che si susseguano tre condizioni:

- Separare gli aspetti visivi dagli aspetti gestionali. In tal senso si prosegue mediante il paradigma di "buon codice", ossia separare ciò che possa essere ritenuto associabile alla *business logic* da ciò che contraddistingue la *user interface*
- Provvedere a garantire un numero elevato di *features* che possano coprire il più ampio insieme di dispositivi
- Ricompilare solamente quando strettamente necessario

Solitamente è preferita la soluzione Android, data soprattutto dalla capacità del framework di rilevare tutte le caratteristiche configurative del dispositivo, associandone le risorse appropriate, e dalla facilità con cui sia possibile aggiungere nuove *resources* per supportare un'ulteriore categoria di device.

All'interno di un qualsiasi progetto le risorse sono destinate all'interno della directory **res**; si tratta di una gerarchia già predisposta per i molteplici *package* inseriti.

Resources Definition

Le risorse sono inizializzate mediante il metodo **dichiarativo** espresso dai file **XML**, in cui ognuna di esse possiede un proprio nome e identificativo.

```
<resources>
  <string name="hello">Hello World!</string>
  <string name="labelButton">Insert your name</string>
</resources>
```

Dichiarazione delle risorse mediante file XML

Tali risorse possono essere acquisite tramite la **classe R**, il cui funzionamento simula un *collante* tra il codice e il documento XML; si tratta di un file generato e gestito automaticamente, ricreato qualora dovessero essere riportate delle modifiche.

Applica il design pattern *Singleton*, pertanto l'intero progetto sviluppato fonda sulla singola istanza della classe. Quindi, ricapitolando, **R** contiene tutti gli identificativi delle resources poste all'interno della directory *res*.

```
public finale class R {
    public static final class string {
        public static final int hello=0x7040001;
        public static final int labelButton=0x7040005;
    }
}
```

Definizione della classe R

L'*ID* definito precedentemente è composto da due sezioni esplicative, definite come segue:

- Il **type**, ossia il tipo della risorsa, ad esempio string, menu, layout ...
- Il **name**, a sua volta contraddistinto dal *filename* e dal *value* posto all'interno del documento XML

Proseguendo, esistono due modalità di accesso al contenuto delle risorse:

- Accesso tramite XML, in cui è attuata una sintassi simile a

`@[<package_name>:]<resource_type>/<resource_name>`

Il `@[<package_name>:]`, oltre ad essere una sezione opzionale, definisce il nome del *package* in cui la risorsa è allocata. Il `<resource_type>` è il nome del *tipo* della risorsa. Il `<resource_name>` rappresenta il *filename* della risorsa.

Di seguito è proposto un titolo esemplificativo, in maniera tale che possa essere definito nel migliore dei modi il concetto espresso prima.

```
<LinearLayout>
    <TextView android:id="@+id/label1"
        android:text="@string/labelText"
        android:textcolor="@android:color/black"/>
    <Button android:id="@+id/button1" android:text="@string/labelButton"
        android:background="@color/my_red"/>
</LinearLayout>
```

Accesso al contenuto delle risorse mediante XML

Da esempio sono differenti i punti salienti in cui occorre soffermarsi. Nella dichiarazione della *TextView* si nota l'utilizzo della sintassi `<resource_type>/<resource_name>`, definita in `android:text="@string/labelText"`, mentre, sempre nella medesima sezione, si evidenzia la presenza di un value in cui è espresso il `@[<package_name>:]`, in `android:textcolor="@android:color/black"`. Contrariamente, all'interno del tag *Button* è visualizzata la denominazione di un ulteriore *package name*, in questo case ritrae un pacchetto ideato dallo sviluppatore e non dato built-in da Android, come da nomenclatura precedente.

- Accesso tramite *codice Java/Kotlin*, dove la sintassi adeguata promuove

[package_name]R.resource_type.resource_name

Nuovamente, il *package_name* definisce il *package* in cui la risorsa è allocata. Il *resource_type* è il nome del *tipo* della risorsa. Il *resource_name* è la denominazione del *filename*.

Values

METTERE SOLAMENETE ESEMPI, MAGARI COMMENTARLI, DA SLIDE 20 -> SLIDE