Introduzione alla piattaforma Android

IUM Prof.ssa Giuliana Vitiello

L'alba di una nuova era

- L' enorme diffusione di dispositivi quali Smartphones e Tablets ha sicuramente rivoluzionato il modo in cui un numero sempre maggiore di individui accede alle informazioni di interesse, comunica con amici e colleghi, fruisce di contenuti multimediali.
- Se da un lato questa rivoluzione offre l'opportunità di realizzare applicazioni innovative o la possibilità di studiare nuovi interessanti paradigmi di interazione con gli utenti, dall'altro ci troviamo di fronte ad una nuova serie di problematiche per progettisti e sviluppatori di applicazioni.

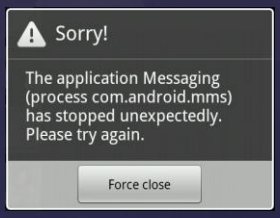
Le sfide dello sviluppo su piattaforme mobile

Rispetto ad un tradizionale Personal Computer, nei dispositivi mobile i progettisti di applicazioni devono quotidianamente confrontarsi con i seguenti limiti :

- Scarsa potenza di elaborazione
- Scarso quantitativo di memoria RAM
- Scarsa capacità di storage
- Schermi piccoli e poco risoluti
- Alti costi economici per il trasferimento di dati su reti mobili, basso transfer rate e alta latenza
- Connessione dati inaffidabile
- DURATA LIMITATA DELLA BATTERIA

Le sfide dello sviluppo su piattaforme mobile - Le aspettative degli utenti

- Performance
- Reattività
- Interfaccia grafica facile da usare
- Sicurezza dei dati



PARTE I : La piattaforma Android

Un po' di storia

- Inizialmente sviluppato dalla startup Android Inc.
- Nel 2005 la startup viene acquistata da Google
- Il 5 novembre 2007 l'Open Handset Alliance, un consorzio formato da 84 aziende il cui intento è promuovere l'utilizzo di standard aperti per i dispositivi mobile, presenta pubblicamente Android
- Il 12 novembre l'OHA rilascia il Software Development Kit.

Android : una piattaforma aperta per lo sviluppo mobile

The first truly open and comprehensive platform for mobile devices, all of the software to run a mobile phone but without the proprietaty obstacles that have hindered mobile innovation.

Andy Rubin

Android is a software stack for mobile devices that includes an operating system, middleware and key applications.

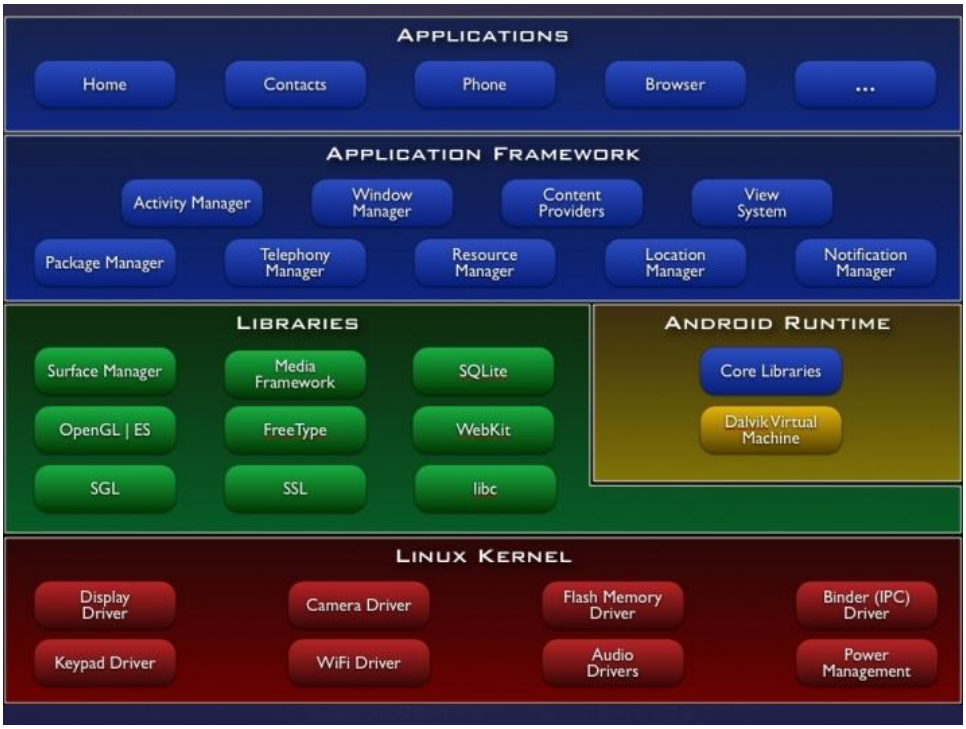
Android Developers Website

Le componenti del sistema Android

- Hardware Reference Design : descrive le caratteristiche richieste ad un dispositivo mobile per supportare lo stack software.
- Linux Kernel : fornisce una interfaccia a basso livello con l'hardware, gestione della memoria e controllo dei processi
- Librerie open source per lo sviluppo di applicazioni tra cui SQLite, WebKit, OpenGL
- Un sistema di run time usato per ospitare ed eseguire le appicazioni. Il run time include la virtual machine Dalvik e le librerie fondamentali che forniscono le funzionalità tipiche di Android.

Le componenti del sistema Android

- Un application framework che espone i servizi di sistema (es. servizi di telefonia o di localizzaione) all'application layer.
- Un framework per il lancio delle applicazioni
- Applicazioni preinstallate
- L' SDK utilizzato per creare le applicazioni



Dalvik, la virtual machine di Android

- Un elemento chiave del sistema Android
- Altamente ottimizzata per offrire buone prestazioni su dispositivi con poca memoria e potenza di calcolo limitata
- Utilizza i servizi sottostanti del Kernel per gestire le funzionalità di basso livello.
- Il bytecode su cui lavora non è bytecode java. Il tool dx, compreso nell'SDK, trasforma i tradizionali file .class di Java in un altro formato (.dex)
- Utilizzando la VM gli sviluppatori non devono mai preoccuparsi della implementazione hardware sottostante

PARTE II
Le applicazioni Android

Nozioni fondamentali

- Tipicamente scritte utilizzando il linguaggio Java
- Il codice compilato, insieme ad ogni altra possibile risorsa richiesta dall'applicazione, viene "pacchettizzato" in un Android package, un file con suffisso .apk
- Tutto il codice presente in un file APK rappresenta una applicazione Android
- Ogni applicazione Android viene eseguita in un processo separato, ciascuno con la propria istanza della Dalvik VM

Tassonomia delle applicazioni Android

La maggior parte delle applicazioni per Android ricade in una delle seguenti quattro categorie :

- *Foreground* : Una tipologia di applicazione utile solo quando si trova in primo piano es. un videogioco. Quando non è visibile questo tipo di applicazione viene ''sospesa'' dal sistema
- *Background* : Una tipologia di applicazione che effettua la maggior parte del suo lavoro in modo ''invisibile'' per l'utente. Esempio : Un autoresponder per gli SMS
- *Intermittent* : Una tipologia di applicazione che, dopo una prima interazione con l'utente, effettua la maggior parte del suo lavoro in background. Esempio : Un mediaplayer
- *Widget* : Una tipologia di applicazione utilizzabile soltanto sulla home screen del dispositivo. Esempio : Un indicatore della carica della batteria.

Le componenti fondamentali di una applicazione

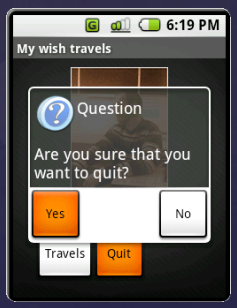
- Le applicazioni Android non hanno un metodo "main()" dal quale tutto ha inizio
- Posseggono (sono formate da) "componenti" che il sistema può instanziare ed eseguire secondo le necessità
- Ci sono quattro tipi di componenti :
  1. Activities
  2. Services
  3. Broadcast Receivers
  4. Content Providers

Activities

- Una Activity fornisce una interfaccia visuale per un compito che l'utente deve compiere
- Equivale ai "Form" delle applicazioni desktop
- Ogni Activity è indipendente dall'altra
- Piu "Activities" insieme formano una interfaccia utente.

Activities

- L'Activity corrente avvia l 'Activity successiva
- Ogni Activity possiede una finestra predefinita ove rappresentare il contenuto
- Una Activity può usare anche finestre addizionali es. un pop-up
- Il contenuto visuale effettivo di una ''finestra'' è fornito da una gerarchia di ''Views''



Activities

- Ogni View controlla un particolare rettangolo all'interno della finestra
- Le view sono quindi il luogo ove avviene l'effettiva interazione dell'utente con l'Activity corrente

Services

- Non possiedono una interfaccia grafica, ma sono eseguiti in background dal sistema per un periodo indefinito
- Esempio : un media player che esegue musica in sottofondo
- Sono eseguiti nel thread principale del processo
- Per evitare il blocco dell'applicaizione è possibile generare altri thread

Broadcast receivers

- Hanno il compito di ricevere e reagire ad un preciso messaggio generato all'interno del sistema
- Esempio : Livello di batteria basso, scatto di una foto, ricezione di un mesaggio di testo
- Una applicazione può avere un numero indefinito di B. Receivers
- Come con i Services anche i B. Receivers non dispongono di interfaccia utente
- Possono tuttavia istanziare una Activity come risposta ad un messaggio ricevuto

Content Providers

- Rendono possibile la condivisione delle informazioni
- Attraverso i C. Providers una applicazione può rendere disponibile i propri dati ad altre applicazioni
- I dati possono essere memorizzati in un qualsiasi formato : database Sqlite, file di testo ecc

Content providers

- Un C. Provider fornisce un insieme standard di metodi che permettono alle applicazioni di recuperare o memorizzare il particolare tipo di dati da esso gestito
- TUTTAVIA, i metodi non vengono invocati direttamente dall'applicazione interessata
- Viene invocato un ''Content Resolver'' che può interfacciarsi con qualunque C. Provider
- Un C. Resolver coopera con un C. Provider per servire la richiesta

E se l'applicazione che gestisce i dati che ci interessano non è in esecuzione?

- Ogni volta che siamo in presenza di una richiesta che dovrebbe essere soddisfatta da una particolare applicazione, Android si assicura che essa sia in esecuzione avviandola se necessario

Attivare un componente

- Un Content Provider viene attivato grazie a una richiesta da parte di un Content Resolver
- Come possiamo attivare una Activity, un Service o un Broadcast receiver?

Il collante del mondo Android : Gli Intents

- Una Activity, un Service o un Broadcast Receiver sono attivati mediante speciali messaggi asincroni
- Nel mondo di Android questi messaggi sono chiamati **Intents**

Gli Intents

- Un Intent è un particolare ''oggetto'' al cui interno è memorizzato un messaggio
- Per le activities e per i services un Intent memorizza l'azione richiesta es ''Visualizzare l'immagine appena scattata''
- Per i broadcast receiver un Intent memorizza ''l'evento'' accaduto

Approfondimento : il ciclo di vita di una Activity

- I vari componenti che formano una applicazione Android possiedono un ciclo di vita ben definito : hanno uno stato iniziale quando vengono istanziati per rispondere, ad esempio, ad un intent e uno stato finale quando vengono eliminati dal sistema
- Durante il loro ciclo di vita possono, tuttavia, assumere anche degli stati ''intermedi'' : possono essere attivi o inattivi oppure, nello specifico caso delle Activity possono essere o non essere visibili all'utente
- Capire il ciclo di vita di una activity è fondamentale per assicurare una buona esperienza utente e per gestire in modo efficiente le risorse

Approfondimento : il ciclo di vita di una Activity

- Una Activity ha essenzialmente TRE STATI
  - ***Active o Running*** : Quando ottiene il focus per interagire con l'utente
  - ***Paused*** : Quando ha perso il focus ma è ancora visibile all'utente
  - ***Stopped*** : Quando è completamente oscurata da un'altra Activity

Quando una Activity si trova negli stati Paused o Stopped il sistema Android può eliminarla dalla memoria sia invocando il metodo ***finish()*** sia, semplicemente, killando il suo processo

Approfondimento : il ciclo di vita di una Activity

- In presenza di un cambiamento di stato, abbiamo a disposizione sette metodi per ''informare'' la nostra Activity

1. onCreate ()
2. onStart ()
3. onRestart ()
4. onResume ()
5. onPause ()
6. onStop ()
7. onDestroy ()

Questi sette metodi definiscono l'intero ciclo di vita di una Activity

Approfondimento : il ciclo di vita di una Activity

- **onCreate** : Viene invocato nel momento in cui una Activity viene ''creata'' per la prima volta

  Rappresenta il posto ideale ove effettuare il setup delle nostre componenti

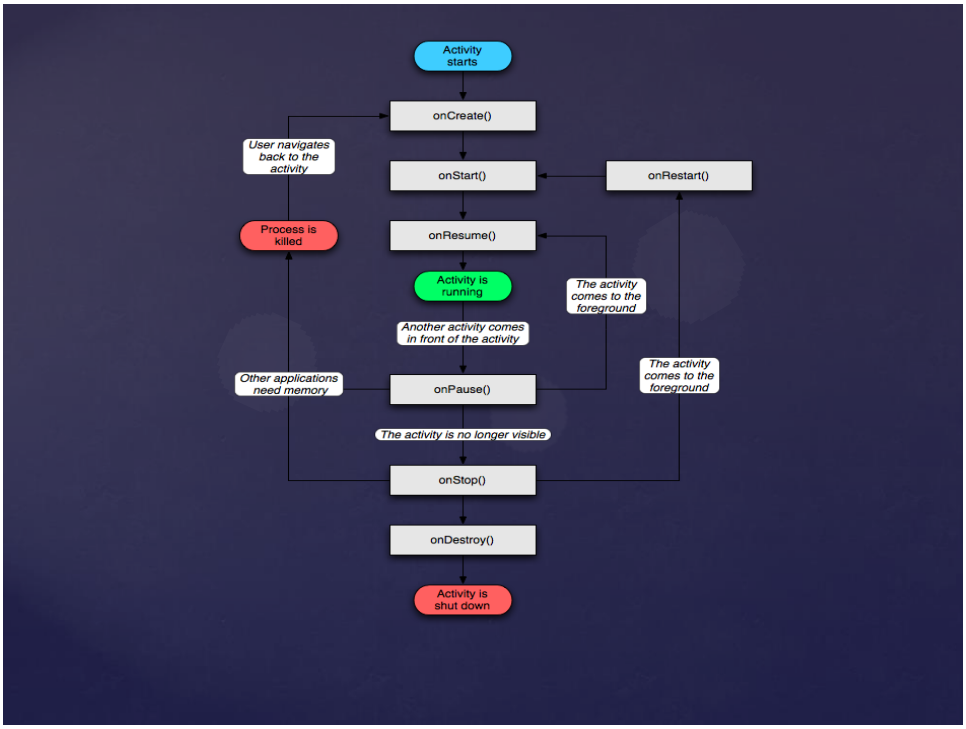
- **onRestart :** Viene invocato dopo che una Activity è stata stoppata ed appena prima che venga nuovamente avviata

Approfondimento : il ciclo di vita di una Activity

- **onStart** : Viene invocato prima che l'Activity venga resa visibile all'utente. E' seguito dal metodo onResume se l'Activity viene portata in primo piano o dal metodo onStop se l'Activity viene nascosta
- onResume : Viene invocato prima che l'Activity inizi la sua interazione con l'utente. E' sempre seguito dal metodo onPause

Approfondimento : il ciclo di vita di una Activity

- **onPause** : Viene invocato quando il sistema sta per effettuare il resume di un'altra Activity. E' seguito dal metodo onResume se l'Activity ritorna immediatamente in primo piano o dal metodo onStop se l'Activity diventa invisibile all'utente
- **onStop** : Viene invocato quando l'Activity non è più visibile all'utente. E' seguito dal metodo onRestart se l'Activity ritorna in primo piano o dal metodo onDestroy se l'Activity sta per essere cancellata definitivamente dal sistema
- **onDestroy** : Viene invocato quando l'Activity sta per essere distrutta



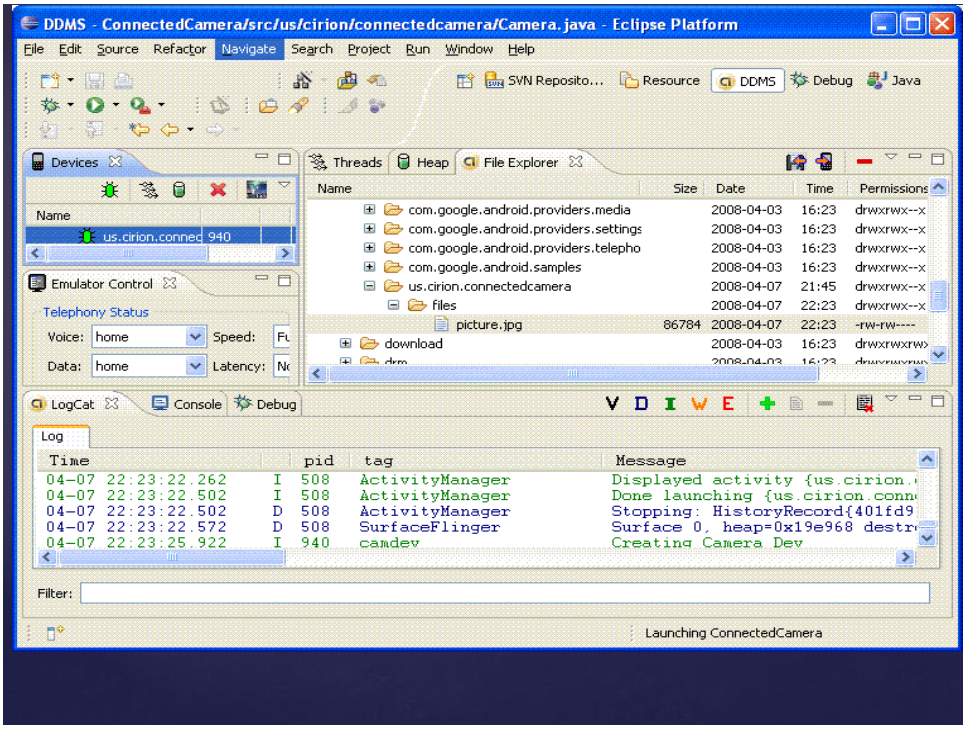
PARTE III – L'Android SDK e i tool di sviluppo

Le caratteristiche dell' SDK

- L' SDK di Android include tutti i tool e le API necessarie a sviluppare le nostre applicazioni
- Possiamo citare, tra le funzionalità a nostra disposizione :
  - API per utilizzare sensori hardware quali accellerometro, bussola digitale, GPS
  - Pieno supporto per applicazioni che integrano mappe cartografiche
  - Librerie per effettuare l'encoding e la riproduzione dei più comuni file multimediali
  - Un web browser integrato che supporta pienamente il nuovo HTML5
  - Librerie per l'utilizzo del Bluetooth
  - Ecc..

Un plugin per Eclipse

- Il modo migliore per sviluppare applicazioni per Android è installare l' ADT plugin
- Semplifica notevolmente le fasi di creazione, testing e debugging
- La guida completa per installare e configurare il plugin è disponibile al seguente indirizzo : http://developer.android.com/sdk/eclipse-adt.html#installing



Una applicazione di esempio

Il classico Hello World, dalla prossima lezione tratteremo qualcosa di più impegnativo…

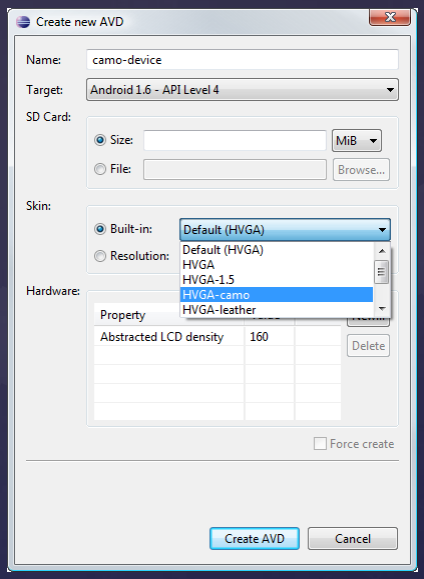
Una applicazione di esempio – Creazione di un AVD

- Testare ogni modifica alla nostra applicazione su un dispositivo fisico può alla lunga rivelarsi una soluzione scomoda
- L' SDK ci offre la possibilità di utilizzare un emulatore software di un dispositivo Android reale
- Un Android Virtual Device (AVD) definisce le caratteristiche (ad esempio la risoluzione dello schermo) del dispositivo fisico che vogliamo emulare
- Una volta creato possiamo riutilizzarlo per più progetti

Una applicazione di esempio – Creazione di un AVD

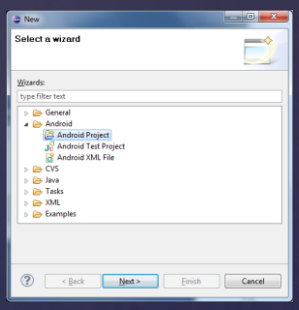
In Eclipse

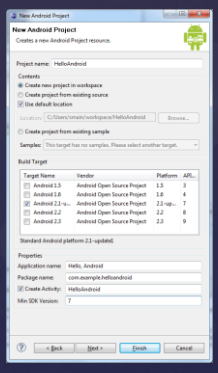
1. Selezionare Window -> Android SDK and ADV Manager
2. Selezionare Virtual Devices nel pannello a sinistra
3. Cliccare su New, comparirà la finestra di dialogo ''Create new AVD''
4. Immettere un nome es. AVD_Test
5. Selezionare il target (Il target è essenzialmente la versione di Android che vogliamo emulare ad esempio Android 1.6 o 2.3)
6. Cliccare su Create AVD



Una applicazione di esempio – Creazione di un nuovo progetto

In Eclipse
File -> New Project -> Android -> Android Project Clicchiamo infine su next





E' banale ☺ Soffermiamoci giusto un minuto sulla schermata di destra…

Una applicazione di esempio – Creazione di un nuovo progetto

- Build target : La versione dell' SDK che stiamo usando per creare la nostra applicazione. Ad esempio se scegliamo 2.3 potremo utilizzare tutte le API presenti nella versione 2.3. Il target che scegliamo al momento della creazione del progetto non deve essere necessariamente uguale alla versione che abbiamo scelto per il nostro emulatore. Deve però NECESSARIAMENTE essere minore o uguale
- Min SDK Version : Indica la minima versione di Android necessaria per eseguire la nostra applicazione. Ad esempio Android 2.1 corrisponde al valore 7 per Min SDK Version

Maggiori informazioni qui :
http://developer.android.com/guide/appendix/api-levels.html

Una applicazione di esempio – HelloAndroid.java

```
package com.example.helloandroid;

import android.app.Activity;
import android.os.Bundle;

public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Il metodo onCreate viene invocato quando la nostra Activity inizia la sua attività
Rappresenta la locazione ideale per effettuare il setup della nostra interfaccia grafica
Ricordiamo che solitamente una Activity possiede una interfaccia grafica, anche se non è obbligatorio…

Una applicazione di esempio – HelloAndroid.java

```
package com.example.helloandroid;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Hello, Android");
        setContentView(tv);
    }
}
```

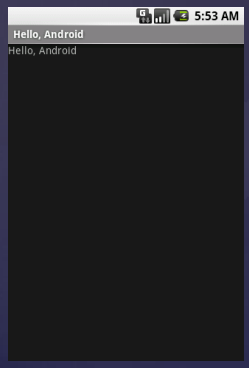
Ricordate cosa abbiamo detto sulle View poco fa?

Una applicazione di esempio – HelloAndroid.java

- Una view è un oggetto (come un pulsante, una immagine o , come nel nostro caso, una casella di testo) che contribuisce a formare il layout grafico della nostra applicazione
- Ciascuno di questi oggetti è una sottoclasse della classe View
- La sottoclasse che gestisce il testo è la TextView
- Avete notato il parametro del costruttore ?
  Maggiori dettagli in futuro….
- setContentView() visualizza la TextView sulla nostra Activity

Una applicazione di esempio - Esecuzione

- Run -> Run e selezioniamo Android Application



Hello, Android nella barra grigia rappresenta il nome della nostra applicazione.
Viene creata automaticamente dal plugin
Si tratta del valore memorizzato nel file strings.xml
e utilizzato nel file AndroidManifest.xml
Maggiori dettagli in seguito, per ora ci basta sapere che questi file sono presenti in ogni progetto di applicazione per Android

Hello,Android sotto la barra grigia rappresenta il testo effettivamente immesso nell'oggetto TextView

Una applicazione di esempio – Introduzione all' XML Layout

- La semplice interfaccia grafica che abbiamo costruito nell' esempio precedente è chiamata in gergo ''Programmatic UI Layout''
- Abbiamo definito e costruito la nostra interfaccia utente direttamente nel codice sorgente
- Non è effettivamente la scelta migliore…
- In progetti reali, con interfacce complesse, è facile superare tranquillamente anche le 1000 linee di codice
- Un piccolo cambiamento nel layout può causare comportamenti ''strani'' e costringerci a lunghe e noiose sessioni di debug…

Una applicazione di esempio – Introduzione all' XML Layout

- Android ci aiuta a superare questo problema offrendoci una alternativa per la definizione e costruzione delle nostre interfacce grafiche
- Definiamo il layout della nostra applicazione in file basati su XML
- L'interfaccia grafica risultate è totalmente uguale a quella definita all'interno del codice sorgente
- Vediamo un esempio….

Una applicazione di esempio – Introduzione all' XML Layout

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/textview"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:text="@string/hello"/>
```

La struttura è semplice :
Un albero di elementi XML dove ogni nodo rappresenta il nome di una View

I progettisti di Android si sono ispirati al mondo dello sviluppo WEB dove la separazione della logica di presentazione da quella di gestione dei dati è un concetto radicato da tempo…

Il file (main.xml) è contenuto nella sottodirectory res/layout di ogni progetto di applicazione per Android

Una applicazione di esempio – Introduzione all' XML Layout

- Ogni volta che avviamo un nuovo progetto, il plugin di Eclipse crea automaticamente per noi il file main.xml
- Modifichiamo quindi la nostra applicazione per utilizzare un layout basato su XML…

Una applicazione di esempio – Introduzione all' XML Layout

- Nel package explorer di Eclipse posizioniamoci nella sottodirectory /res/layout e modifichiamo il file main.xml ivi contenuto

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/textview"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:text="@string/hello"/>
```

Una applicazione di esempio – Introduzione all' XML Layout

- Posizioniamoci ora nella sottodirectory /res/values e modifichiamo il file strings.xml
- strings.xml contiene le stringhe di testo che utilizzeremo all'interno della nostra interfaccia utente

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello, Android! I am a string resource!</string>
    <string name="app_name">Hello, Android</string>
</resources>
```

Una applicazione di esempio – Introduzione all' XML Layout

- Modifichiamo la nostra classe…

```
import android.app.Activity;
import android.os.Bundle;

public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Cosa è cambiato ?

Al metodo setContentView al posto di un oggetto di tipo View abbiamo passato un reference al file con il nostro layout

Una applicazione di esempio – Introduzione all' XML Layout

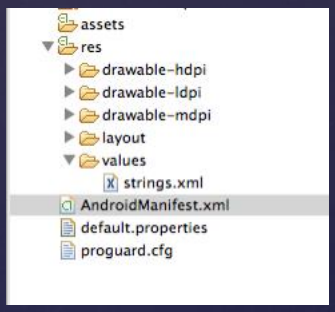
- ***R.layout.main*** è un ''oggetto'' che rappresenta il nostro file di layout main.xml
- ***R.java*** è un file che viene generato automaticamente e rappresenta un ''indice'' per tutte le risorse che abbiamo definito nel nostro file. Man mano che modifichiamo i nostri file di layout il file R.java viene automaticamente aggiornato

R.java

```
public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class id {
        public static final int textview=0x7f050000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

L'application Manifest

- Prima che Android possa eseguire un componente di una applicazione deve ''imparare'' che quel componente ''esiste''
- Ogni progetto di una applicazione per Android include un file speciale, AndroidManifest.xml
- Questo file permette di definire la struttura ed i metadati dell'applicazione, i suoi componenti e i suoi requisiti



L'application Manifest

- E' composto da vari nodi, uno per ciascuno dei componenti che contribuiscono a formare l'applicazione (Activities, Services, C. Providers, B. Receivers)
- Definisce come questi componenti comunicano fra loro o con le altre applicazioni installate (mediante gli Intents)
- Permette di specificare metadati specifici dell'applicazione (es. icone o temi)
- Permette di definire nodi speciali relativi ad esempio, a impostazioni di sicurezza, unit testing, requisiti hardware particolari ecc

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.example.test"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="7" />

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".TestActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>
</manifest>
```

Riferimenti

ANDROID DEVELOPERS
http://developer.android.com/index.html

Reto Meier
Professional Android 2 Application Development
Wrox Press