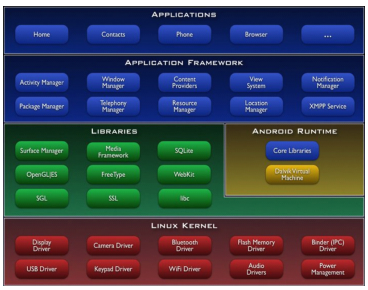
Android

A. Ferrari

Android

- Android è un sistema operativo per dispositivi mobili.
- Inizialmente sviluppato da Startup Android Inc. acquisita poi nel 2005 da Google Inc.
- Il cuore di Android è un kernel Linux.
- Direttamente nel kernel sono inseriti i driver per il controllo dell'hardware del dispositivo: driver per la tastiera, lo schermo, il touch screen, il Wi-Fi, il Bluetooth, il controllo dell'audio e così via.
- Sopra il kernel poggiano le librerie fondamentali, anche queste tutte mutuate dal mondo Open Source.
  - OpenGL, per la grafica,
  - SQLite, per la gestione dei dati,
  - WebKit, per la visualizzazione delle pagine Web.

Struttura del sistema



Programmazione

- La programmazione avviene in Java.
- Android dispone di una Java Virtual Machine non standard (Dalvik)
- I sorgenti java vengono compilati in formato dex (Dalvik Executable), una sorta di bytecode.
- Le applicazioni vengono distribuite in forma di pacchetto autoinstallante, un file con estensione .APK .
- Questo non è altro che un file compresso, contenente il software (file con estensione .dex) le sue risorse (immagini, suoni ecc...) e alcuni file XML.

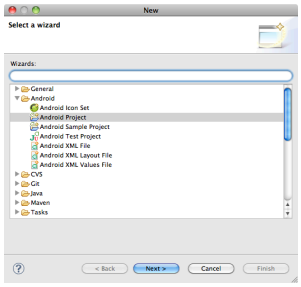
Sviluppo di applicazioni

- E' necessario installare Android SDK contenente emulatore, documentazione e librerie.
- http://developer.android.com/sdk/
- L'SDK Setup permette poi di scaricare i device virtuali e i componenti aggiuntivi.
- Con SDK è possibile creare device virtuali (Android Virtual Device AVD) per testare le applicazioni.

Applicazioni con Eclipse

- Per sviluppare applicazioni con Eclipse è necessario installare il plug-in Android Development Tool (ADT)
- https://dl-ssl.google.com/android/eclipse/
- Configurare Eclipse specificando il percorso dell'Android SDK

Nuovo progetto Android



Ciao Mondo

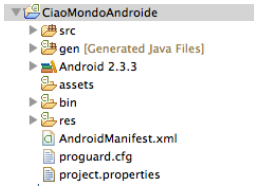
```
package it.pr.itis;

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

public class CiaoMondoAndroideActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        TextView tv = new TextView(this);
        tv.setText("Ciao Mondo");
        setContentView(tv);
    }
}
```

Le applicazioni Android

- Attività
  - Blocchi di un'applicazione che interagiscono con l'utente utilizzando lo schermo ed i dispositivi di input, normalmente fanno uso di android.widget.
  - Sono il modello più diffuso e si realizzano estendendo la classe android.app.Activity.
- Servizio
  - Gira in sottofondo e non interagisce direttamente con l'utente, si realizza estendendo la classe android.app.Service..
- Broadcast Receiver
  - Viene utilizzato quando si intende intercettare un particolare evento (esempio compiere un'azione quando si scatta una foto). La classe da estendere è android.content.BroadcastReceiver.
- Content Provider
  - Sono utilizzati per esporre dati ed informazioni. Costituiscono un canale di comunicazione tra le differenti applicazioni installate nel sistema. Si estende android.content.ContentProvider.

Struttura di una applicazione

- In un'applicazione Android troviamo una struttura abbastanza complessa di directory:
- *src* contiene i package e le classi
- *assets e res* ospitano le risorse esterne (immagini, file audio ecc). res ha una speciale struttura predefinita, formata dalle sotto-directory drawable, layout e values. Le cartelle del gruppo drawable servono per le immagini utilizzate dal software, mentre layout e values ospitano dei speciali file XML utili per definire in maniera dichiarativa l'aspetto dell'applicazione ed i valori utilizzati al suo interno
- *gen* contiene la speciale classe chiamata R. Invocando questa classe è possibile richiamare via codice le risorse memorizzate sotto la directory res.



Gestione dei valori

- I valori sono coppie chiave-valore dichiarate all'interno dei file XMLnella cartella res/values.
- Eclipse, per default, crea il file strings.xml, pensato per raccogliere le stringhe usate dall'applicazione che sarà sviluppata. E' possibile rinominare il file o aggiungerne altri. L'importante è che tutti i file presenti nella cartella values seguano il modello:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="scuola">ITIS</string>
</resources>
```

Richiamare le risorse da XML

- Uno dei problemi della programmazione è l'accoppiamento fra codice e dati. Non è raro vedere dei sorgenti in Java, in C o in qualsiasi altro linguaggio, con valori e messaggi digitati direttamente dentro il codice.
- E' sempre consigliabile separare i dati dal codice, perché in questo modo il software è più facile sia da realizzare sia da mantenere.
- Android favorisce la pratica del disaccoppiamento fra dati e codice.
- Un'applicazione Android è costituita da file dichiarativi XML e da classi Java. Sia in un caso sia nell'altro, ci sono scorciatoie per richiamare le risorse incluse in res.

Un esempio

- Il nome dell'applicazione è inserito in un file xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources> <string name="app_name">LaMiaApplicazione</string> </resources>
```

- e può essere richiamato nel descrittore dell'applicazione AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="mypackage" android:versionCode="1" android:versionName="1.0">
<application android:label="@string/app_name">
...
</application> </manifest>
```

Risorse da Java

- Valori e risorse possono essere richiamati da codice Java servendosi della classe android.content.res.Resources.
- All'interno di una attività è sufficiente richiamare il metodo getResources() per ottenere il punto d'accesso alle risorse dell'applicazione:

```
Resources res = getResources();
String nomeApplicazione = res.getString(R.string.app_name);
```

Ciao mondo (2)

```
package it.pr.itis;

import android.app.Activity;
import android.content.res.Resources;
import android.os.Bundle;
import android.widget.TextView;

public class CiaoMondoAndroideActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Resources res = getResources();
        String messaggio = res.getString(R.string.messaggio);
        TextView tv = new TextView(this);
        tv.setText(messaggio);
        setContentView(tv);
    }
}
```

Ciclo di vita di un'attività

I metodi di un'attività

- protected void onCreate(android.os.Bundle savedInstanceState)
  Richiamato non appena l'attività viene creata.
- protected void onRestart()
  Richiamato per segnalare che l'attività sta venendo riavviata dopo essere stata precedentemente arrestata.
- protected void onStart()
  Richiamato per segnalare che l'attività sta per diventare visibile sullo schermo.
- protected void onResume()
  Richiamato per segnalare che l'attività sta per iniziare l'interazione con l'utente.
- protected void onPause()
  Richiamato per segnalare che l'attività non sta più interagendo con l'utente.
- protected void onStop()
  Richiamato per segnalare che l'attività non è più visibile sullo schermo.
- protected void onDestroy()
  Richiamato per segnalare che l'applicazione sta per essere terminata.

Interfacce

- Gli oggetti fondamentali delle interfacce grafiche Android sono gli oggetti View e Viewgroup.
- Sono oggetti View:
  - bottoni
  - campi di testo
  - icone
  - gli altri oggetti di un'interfaccia grafica
- Sono oggetti ViewGroup:
  - i contenitori che possono mettere insieme più oggetti View.
  - I ViewGroup, inoltre, sono a loro volta degli oggetti View, e di conse- guenza un possono contenere altri ViewGroup.

Widget

- Con il termine widget (congegno) si indicano quei componenti di base per l'interazione con l'utente, come i bottoni, le check box, le liste, i campi di testo e così via.
- I widget predefiniti di Android estendono tutti (direttamente o indirettamente) la classe View, e sono nel package android.widget.

Layout

- Con il termine layout (disposizione, impaginazione), in Android, si identificano tutti quei ViewGroup utilizzabili per posizionare i widget sullo schermo. Android fornisce una serie di layout predefiniti.

I layout predefiniti

- FrameLayout
  - visualizza tutti i componenti figli nell'angolo in alto a sinistra. Se si aggiungono più figli, ogni nuovo figlio va a finire sopra il precedente nascondendo il precedente
- LinearLayout
  - allinea tutti i figli in una linea orrizontale o in verticale. Un Layout verticale ha una colonna di componeti, mentre un layout orrizontale ha una riga di componenti.
- RelativeLayout
  - permette di definire la posizione di ogni figlio relativamente agli altri e al bordo dello schermo.
- TableLayout
  - permette di disporre i componenti in una griglia di righe e colonne.
- Gallery
  - visualizza una singola riga di componenti in una lista scorrevole orizzontalmente.

XML e interfacce grafiche

- La moderna logica di programmazione suggerisce di separare il più possibile la definizione delle interfacce grafiche dalla logica di programmazione.
- L'ambiente di sviluppo Android propone come soluzione la gestione delle interfacce mediante XML
- Eclipse con il plug-in per Android forniscono un editor grafico che rende facile la definizione delle interfacce e memorizza la struttura in un fiel XML

editor grafico