Programmazione Android

Introduzione

Luca Vassena, Ph.D.
MoBe srl - www.mobe.it

mobile being

Scaletta

- Strumenti di sviluppo
- Prima applicazione
- Aspetti peculiari di Android
  - Activity e Intent
- Conclusioni

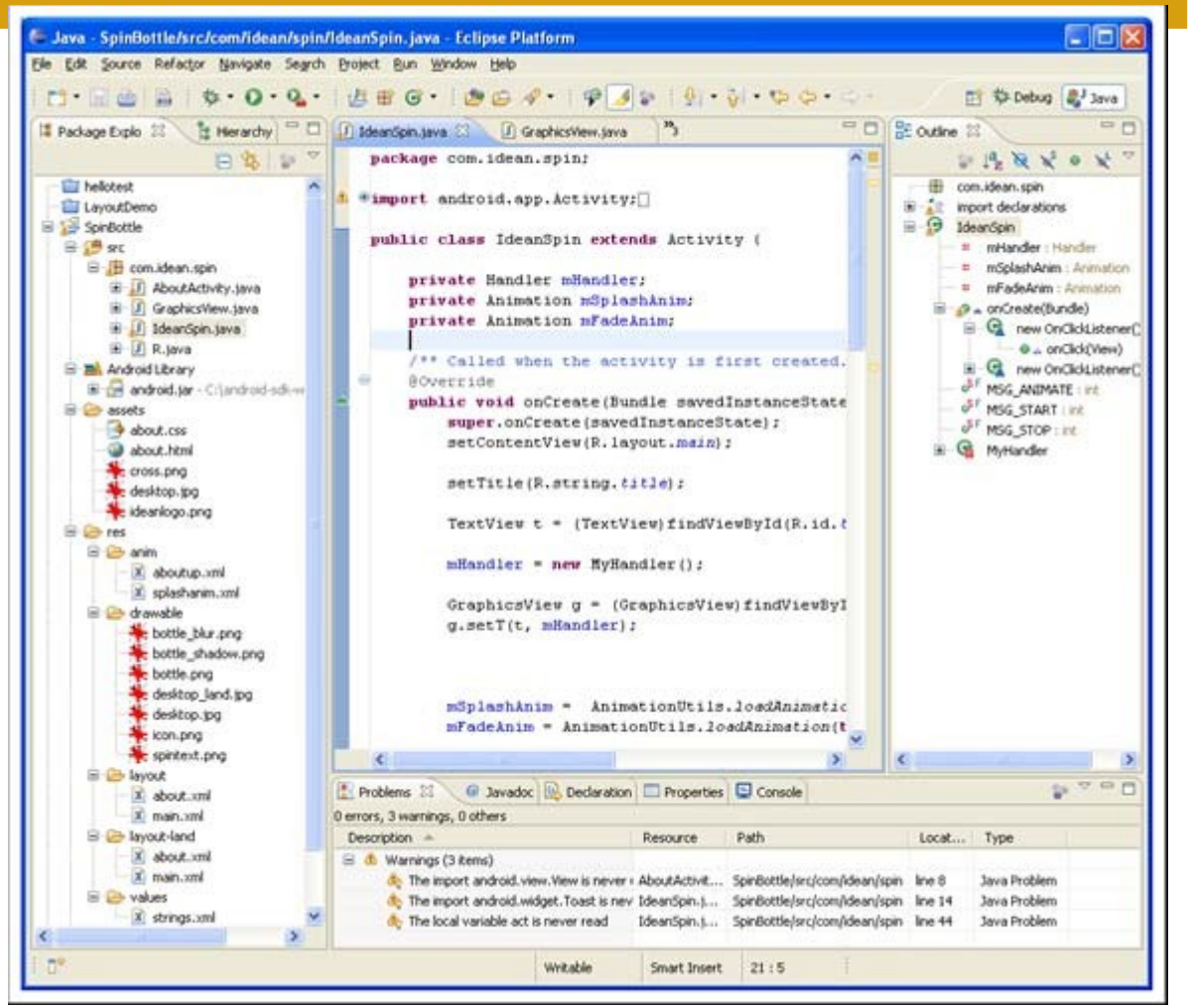
Strumenti

Strumenti di sviluppo

- Disponibili per Windows, Mac OS X (Intel) e Linux
  - A differenza di iPhone
- Strumenti
  - JDK (Java > 1.5)
  - Eclipse
  - Android SDK
    - SDK Starter package
    - Android Development Tools (ADT), plug-in Eclipse
  - Android NDK
- Ambiente integrato per lo sviluppo di applicazioni
  - Gestione automatica della "burocrazia"
  - Esecuzione e debugging: emulatore o dispositivo reale

Eclipse

- Editor più popolare e completo
  - Miriade di funzionalità
  - Refactoring
  - Debugging
  - Autocompletamento
  - Help integrato
  - ...



Android SDK: emulatore

- Emulatore
  - Configurabile attraverso AVD
- Android Virtual Device (AVD)
  - Dispositivo virtuali Android che permettono di modellare le caratteristiche di funzionamento dell'emulatore
  - Altamente configurabili
  - Emulazione di "quasi" tutte le caratteristiche HW e SW
    - RAM, schermo dimensione e risoluzione, ecc.
- Limitazioni
  - Bluetooth, USB, cattura audio/video

Android SDK: ADB

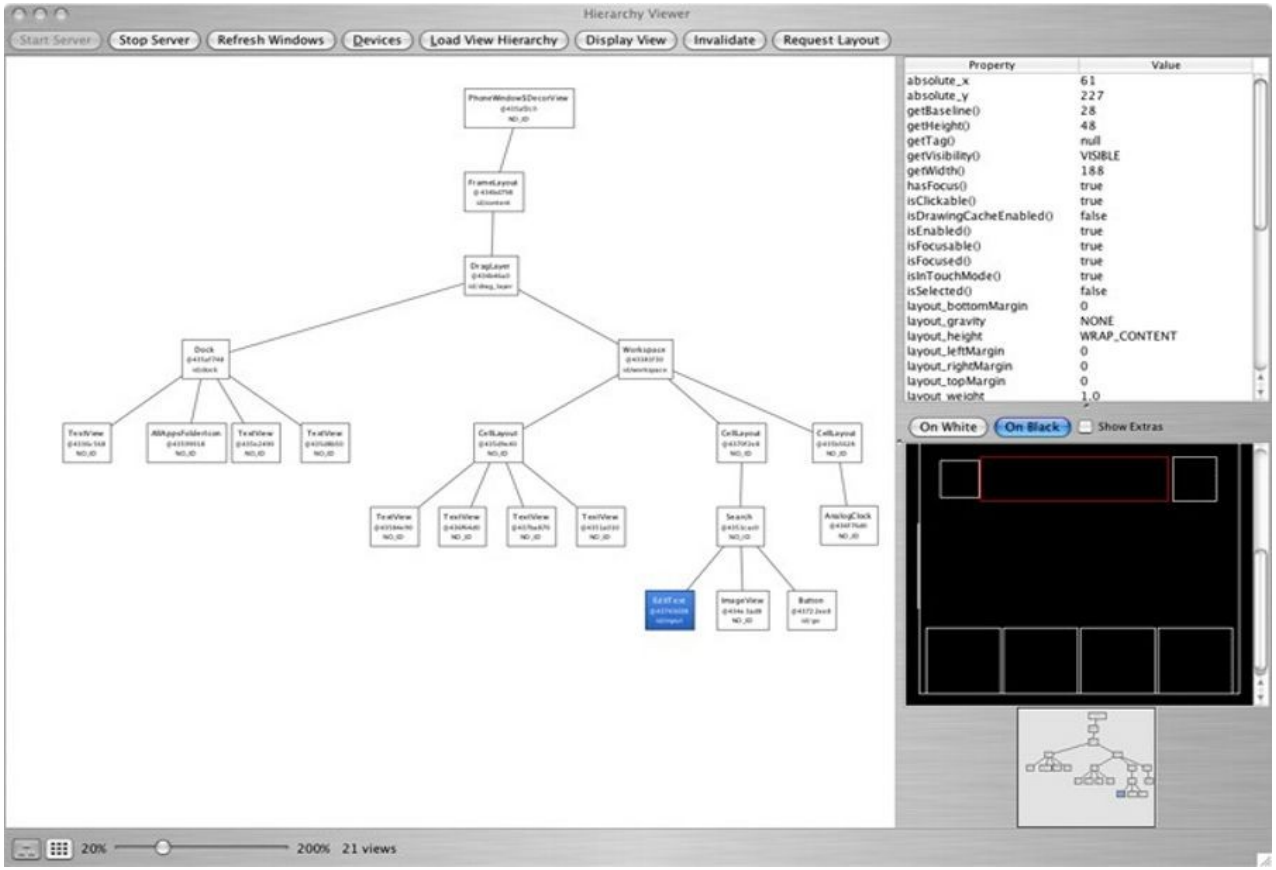
- Android Debug Bridge (ADB)
  - Strumento che permette di installare i pacchetti apk sull'emulatore o dispositivo vero e accedervi tramite linea di comando
- Funzionalità
  - Gestisce lo stato di emulatori e dispositivi reali
  - Installazione/rimozione di applicazioni
  - Copia file da/su dispositivi
  - Logcat
  - Shell remota

Android SDK: DDMS

- Dalvik Debug Monitor Service (DDMS)
  - Strumento per gestire processi sull'emulatore o un dispositivo vero
- Funzionalità
  - Debug
  - Informazioni su stato memoria e processi
  - Kill dei processi
  - Cattura screenshot
  - Esplorazione File System
  - Controlli avanzati dell'emulatore

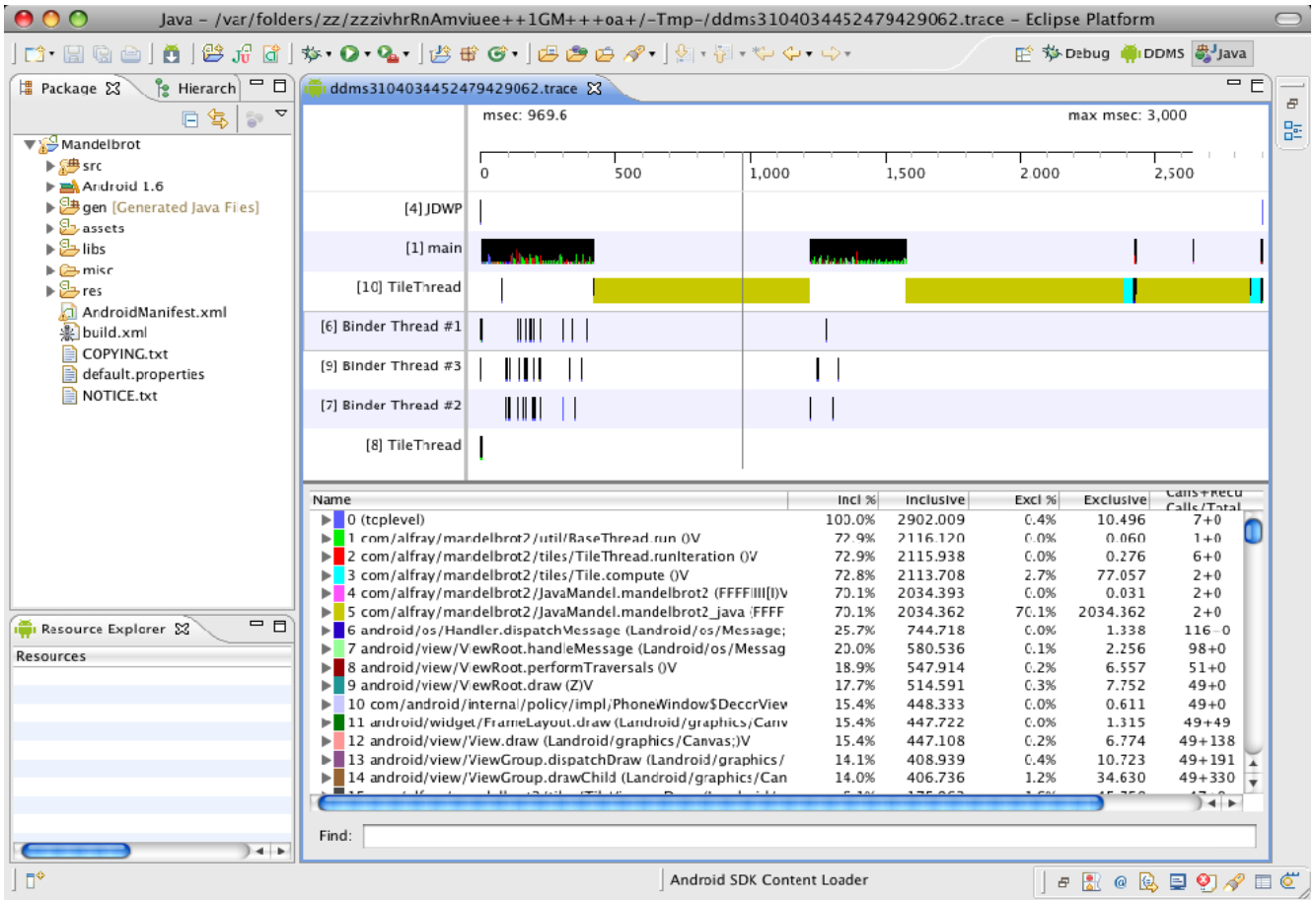
Android SDK: hierarchy viewer

- Visualizzazione della gerarchia della UI
- Pixel Perfect View



Android SDK: traceview

- Timeline Panel
  - Quando thread e metodi sono stati lanciati e fermati
- Profile Panel
  - Riassunto dell'esecuzione di un metodo



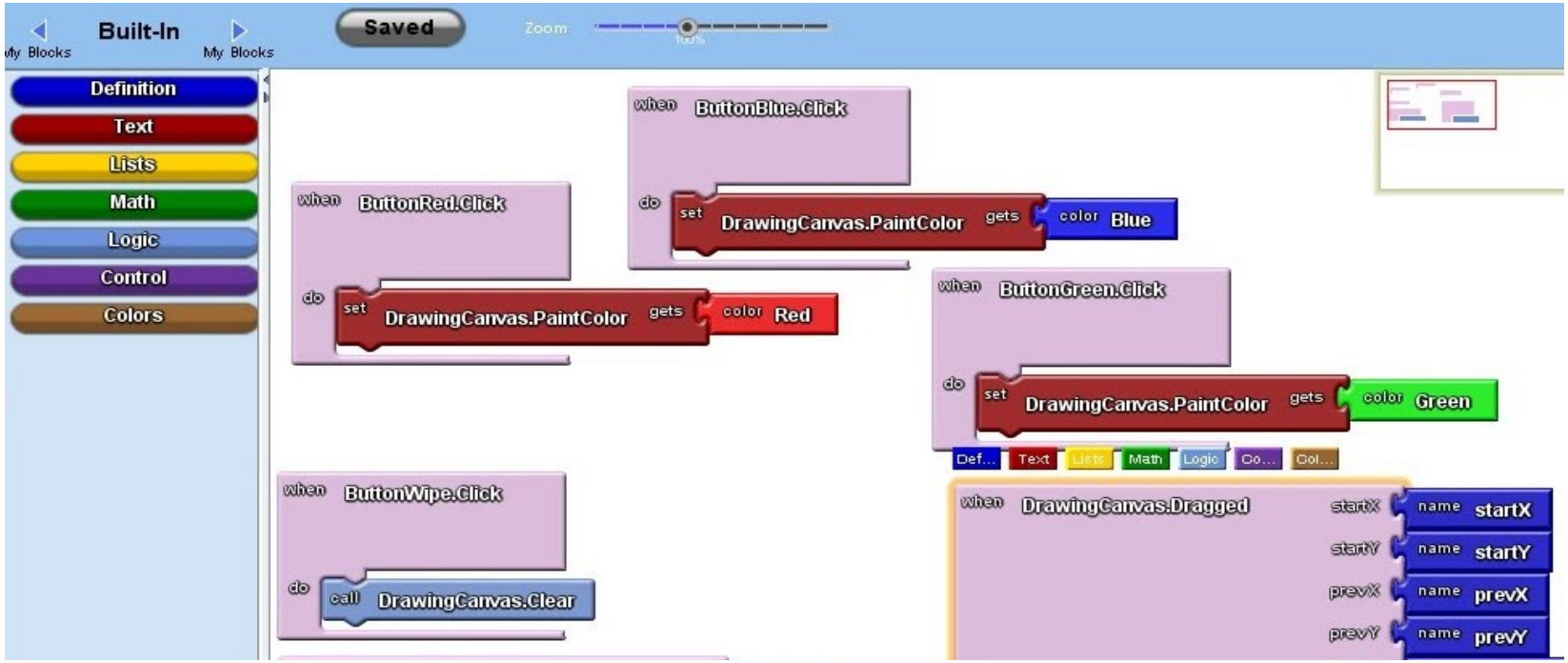
Android SDK: altro

- Android Asset Packaging Tool (AAPT)
  - Permette di creare pacchetti .apk per distribuire le applicazioni
- UI/Application Exerciser Monkey
  - Gira sull'emulatore o dispositivo reale e genera flussi pseudo-casuali (configurabili) di eventi tipo click, tocchi ed eventi a livello di sistema tipo batteria scarica
  - Serve a testare le applicazioni in condizioni di stress

Eclipse ADT plugin

- Rende Eclipse un ambiente di sviluppo integrato per applicazioni Android
  - Gestire diversi tool dell'SDK direttamente da Eclipse
  - Ad es. compilare ed eseguire direttamente l'applicazione nell'emulatore, prendere screenshot, gestire i breakpoint per il debugging tramite l'interfacciamento con il tool DDMS
- Dettagli
  - Wizard per creare nuovi progetti
  - Editor di interfaccia
  - Automatizzazione del processo di building
  - Gestione file XML (manifest file, resources, ecc.)
  - Gestione della costruzione dei pacchetti APK e delle firme rendendo le applicazioni pronte ad essere distribuite

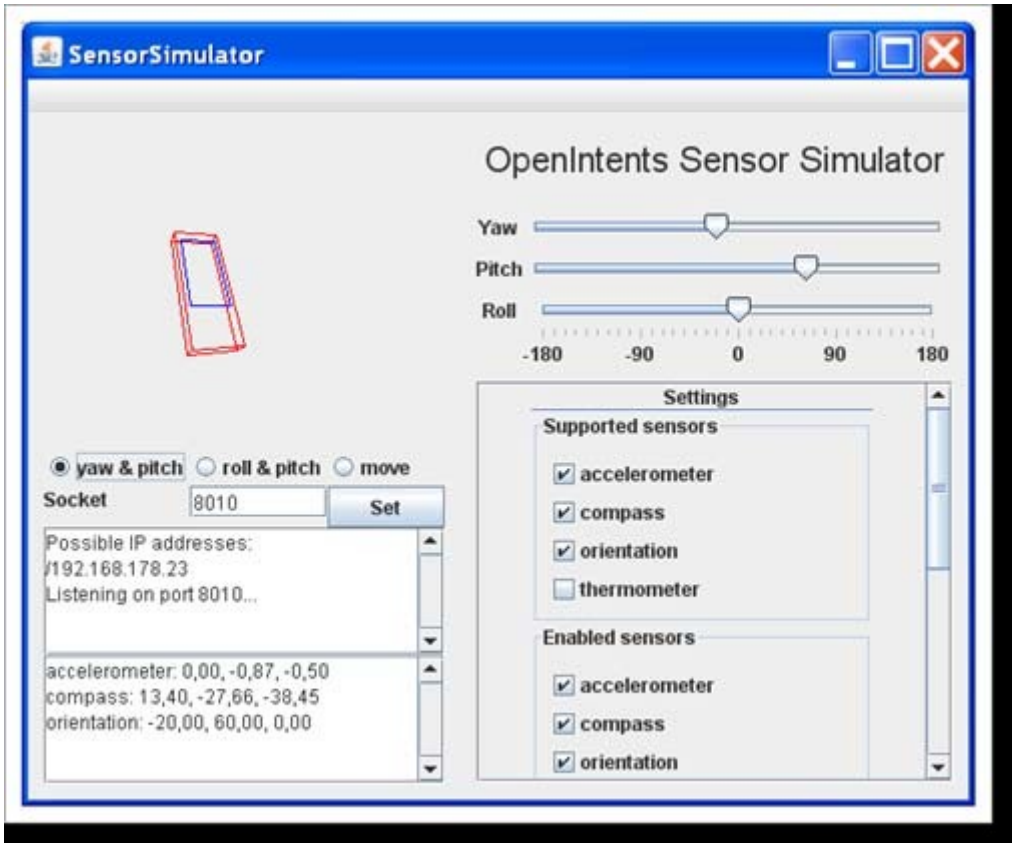
Google App Inventor



- Applicazione Web per la creazione di applicazioni
  - Obiettivi soprattutto educativi
  - Drag-and-drop di elementi grafici e logici

Tool di terze parti

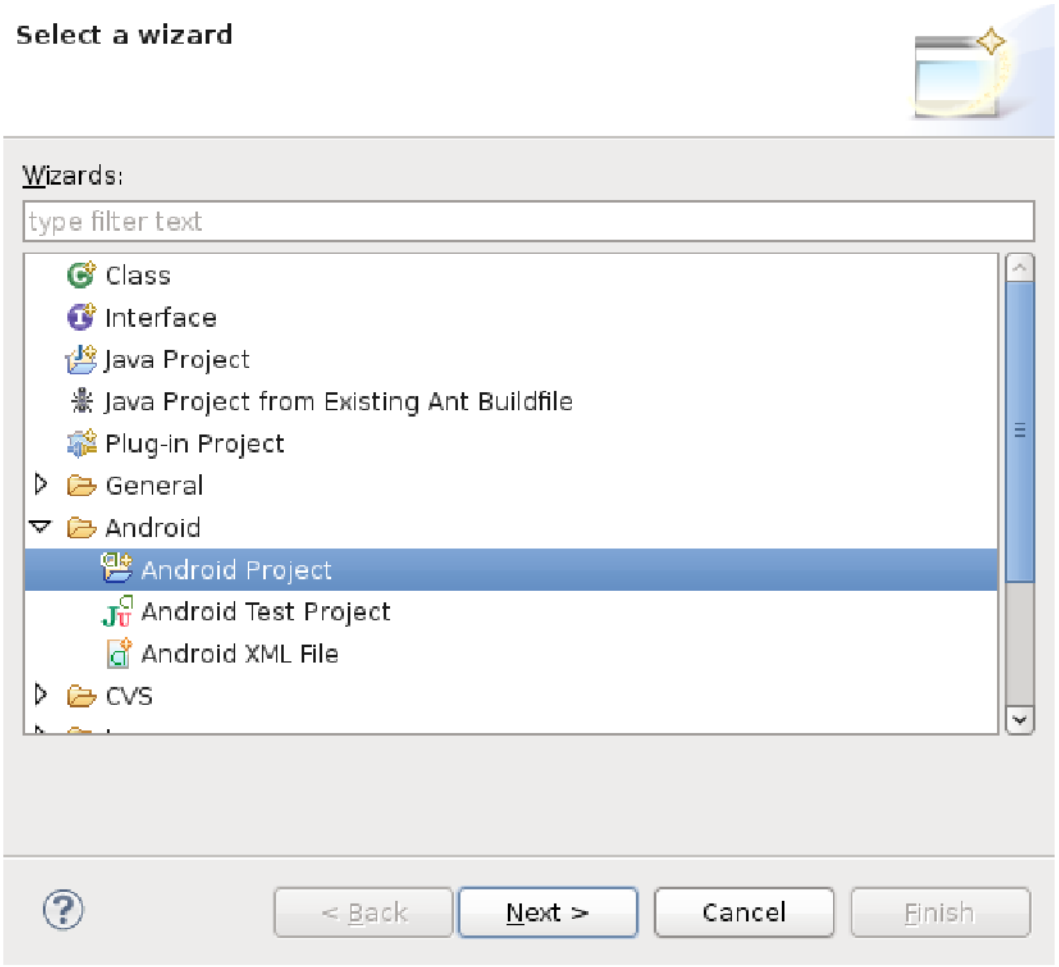
- Sensor simulator
  - http://code.google.com/p/openintents/wiki/SensorSimulator
  - Simula sull'emulatore dati relativi ai sensori



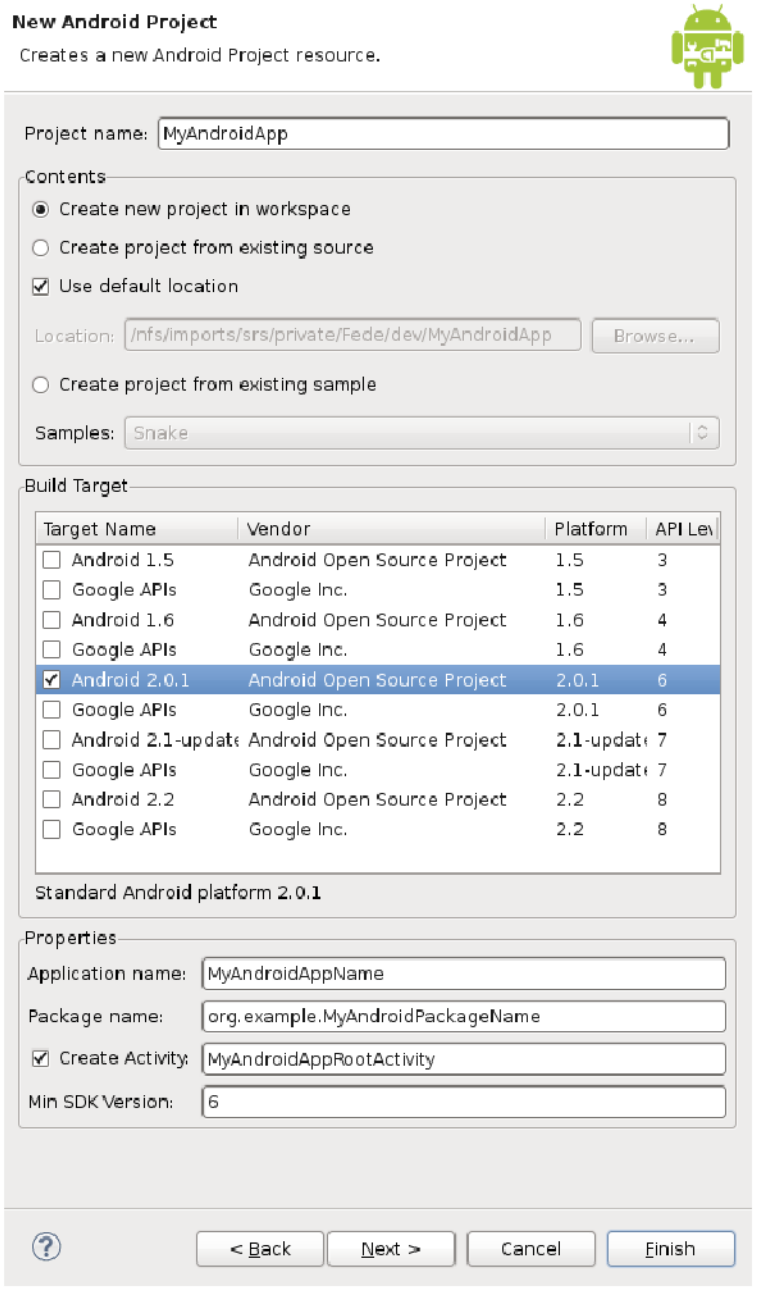
Hello Android

Nuovo progetto (1/2)

- Avviabile da Eclipse dopo aver installato il plugin ADT
- Procedura guidata che crea l'ambiente di lavoro (directory e file) necessari
- Viene creato un programma minimale già pronto per essere eseguito
- Permette di creare direttamente un'applicazione di test (JUnit, un framework per scrivere test ripetibili per applicazioni Java)



Nuovo progetto (2/2)

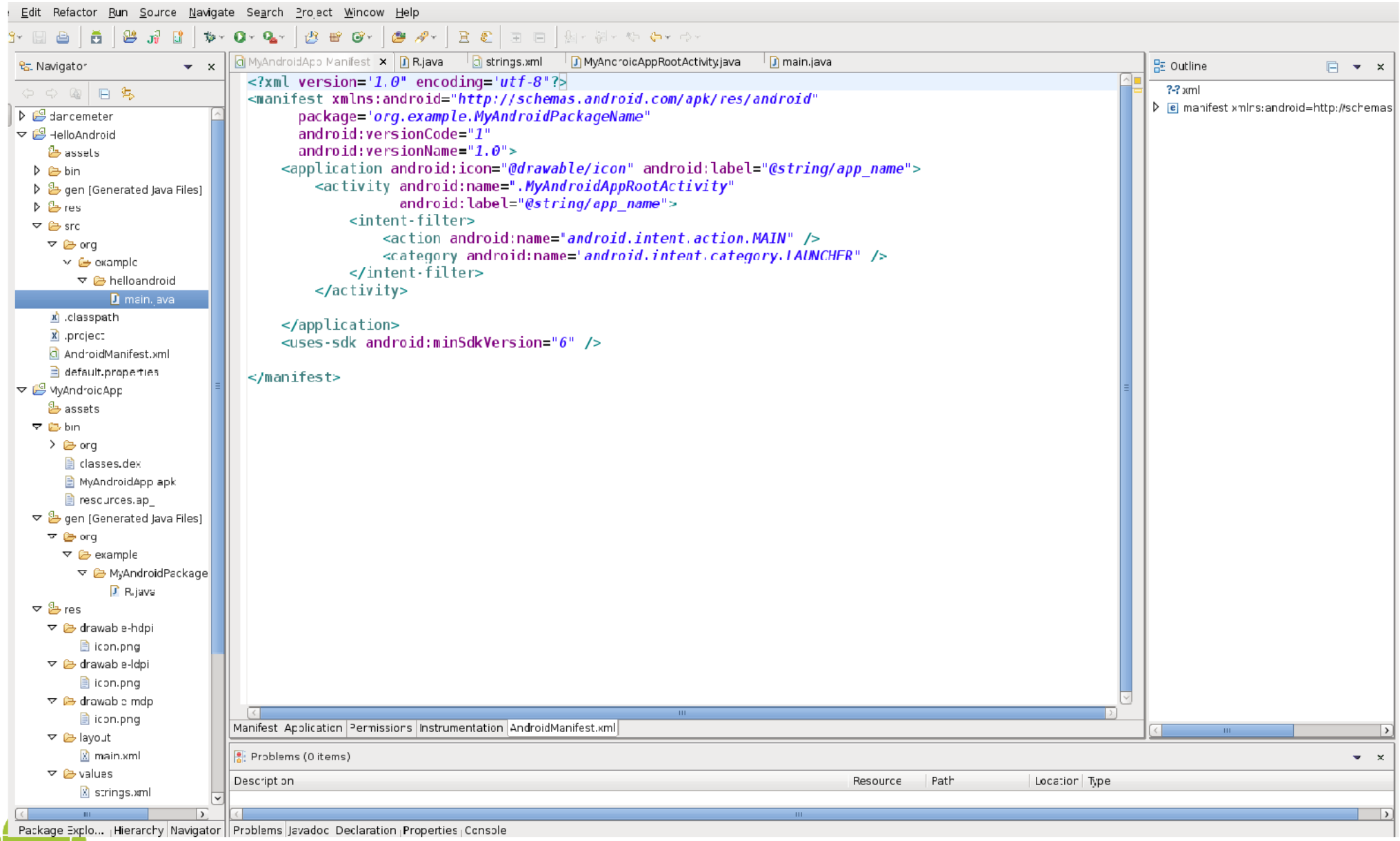


- Nome del progetto
- API level
- Nome applicazione
- Nome package
- Versione minima dello SDK
- ...

Ambiente (1/2)

- Alla fine del wizard si ha un progetto compilabile ed eseguibile
- Vengono creati:
  - Tutte le cartelle necessarie (di base)
  - Il file manifest.xml di base
  - string.xml: file per la gestione di localizzazione/internazionalizzazione delle stringhe dell'applicazione
  - File vari di progetto e risorse di esempio
- Eclipse fornisce già un IDE potente per lo sviluppo di codice Java (navigazione delle classi, autocompletamento, ...)
- Con l'ADT Plugin vengono aggiunte e integrate le funzionalità dell'Android SDK

Ambiente (2/2)


Edit Refactor Run Source Navigate Search Project Window Help
Navigator
MyAndroidApp Manifest
R.java
strings.xml
MyAndroicAppRootActivity.java
main.java
Outline
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="org.example.MyAndroidPackageName"
android:versionCode="1"
android:versionName="1.0">
<application android:icon="@drawable/icon" android:label="@string/app_name">
<activity android:name=".MyAndroidAppRootActivity"
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>
<uses-sdk android:minSdkVersion="6" />
</manifest>
Manifest Application Permissions Instrumentation AndroidManifest.xml
Problems (0 items)
Description Resource Path Location Type
Problems Javadoc Declaration Properties Console
Package Explo... Hierarchy Navigator

Codice iniziale (1/2)

```
package com.google.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 icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.main);
    }

}
```

Codice iniziale (2/2)

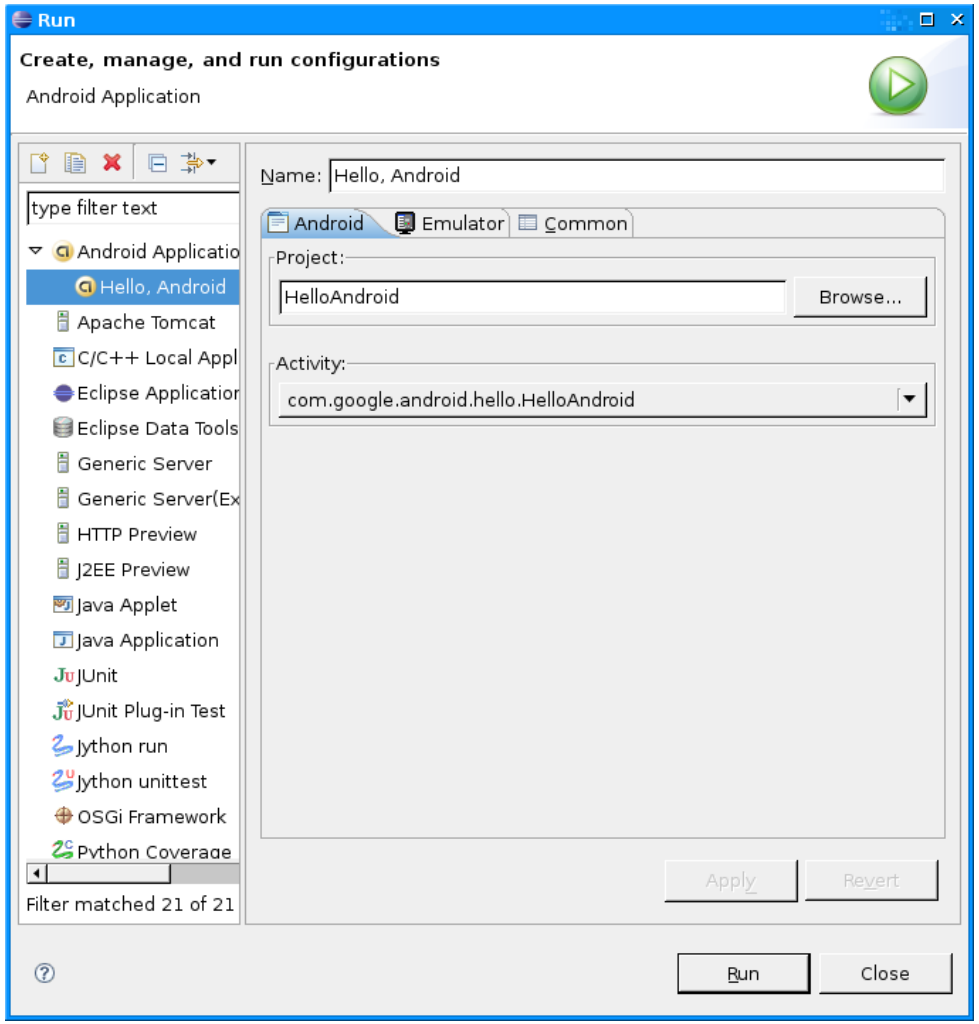
- Activity
  - Singola attività che l'utente può svolgere (pezzo di applicazione)
  - Generalmente coincide con una singola schermata
- OnCreate
  - Chiamato alla prima creazione dell'Activity
- Bundle
  - Riporta lo stato di eventuali precedenti Activity
- SetContenteView
  - Mostra GUI
  - Impostata nel file main.xml

Esecuzione (1/2)

- Grazie all'ADT Plugin è possibile direttamente da Eclipse
  - Gestire le configurazioni di API, versioni SDK, ...
  - Richiamare direttamente l'AVD Manager per creare e gestire le AVD (configurazioni per l'Android Emulator)
- Passi per l'esecuzione di un'applicazione
  - Compilazione
  - Creazione del pacchetto .apk
  - Avvio dell'Android Emulator
  - Installazione dell'applicazione
  - Esecuzione

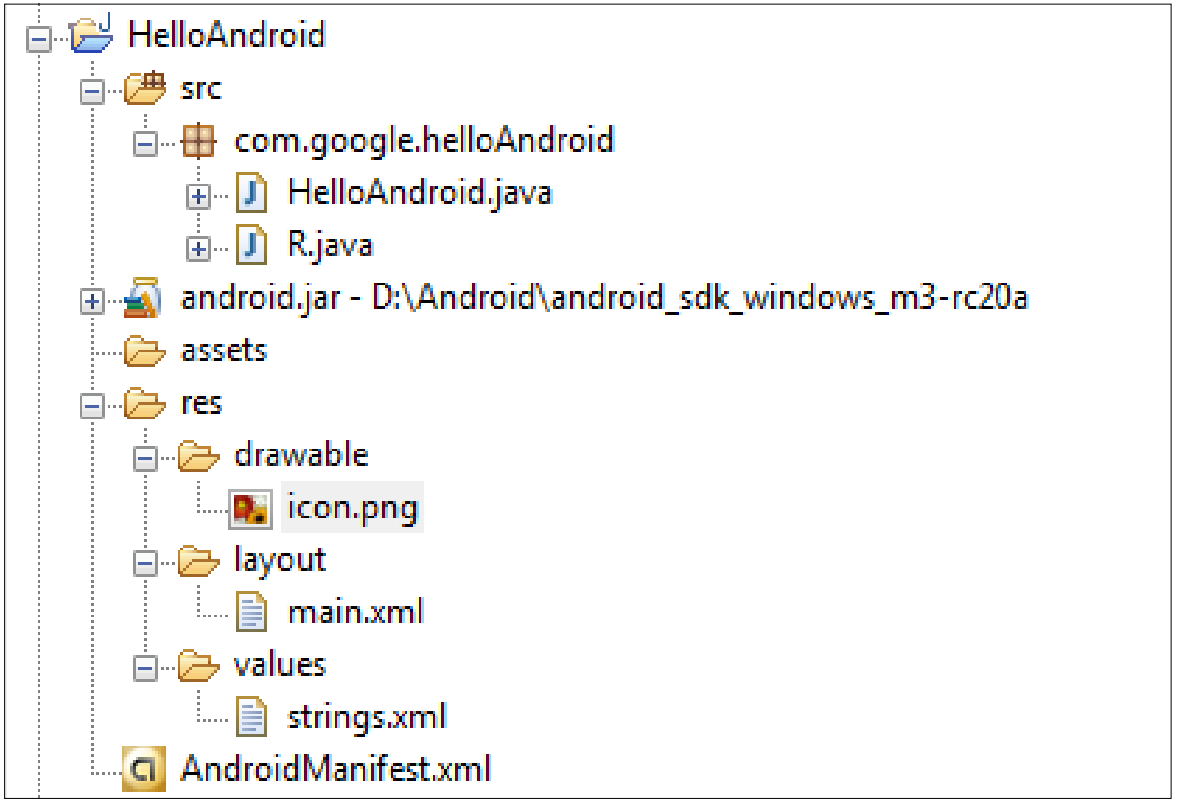
Esecuzione (2/2)

- Run → run



Elementi

- Src: directory sorgenti
  - Contiene packages e file .java
- Android.jar: tutto ciò che riguarda android
- Res: directory risorse esterne
  - Immagini
  - File di layout
  - Testo localizzato
- AndroidManifest.xml



AndroidManifest.xml

- File xml, necessario per ogni applicazione
  - Riporta informazioni globali sull'applicazione: componenti, Activity, dati gestibili, parametri sicurezza, ...

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="com.google.helloAndroid">

  <application android:icon="@drawable/icon">

    <activity class=".HelloAndroid" android:label="@string/app_name">
      <intent-filter>
         <action android:value="android.intent.action.MAIN" />
      </intent-filter>
    </activity>

  </application>

</manifest>
```

main.xml

- File xml di layout
  - Sistema alternativo per la generazione della GUI
  - Non programmatico, ma esterno al codice (come XUL, XAML, ecc.)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
  <TextView
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text=@String/app_name
      />
</LinearLayout>
```

strings.xml

- Stringhe di testo
- app_name richiamato nel manifest e nel main.xml

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

R.java

- Indice di tutte le risorse
  - Creato e gestito automaticamente da Eclipse
  - Verboten toccare

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

Aspetti particolari

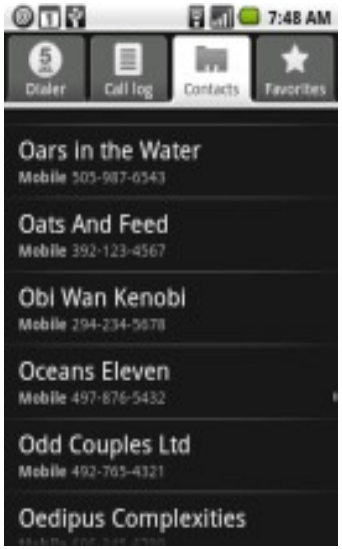
Activity e Intent

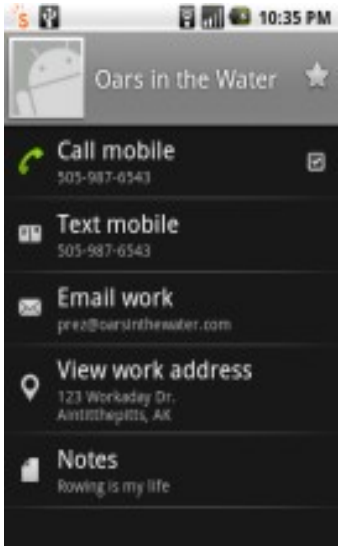
Activity (1/2)

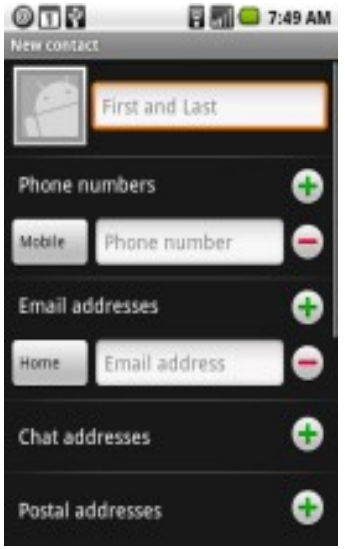
- Applicazione Android
  - Non è un'entità unica, ma una composizione modulare generata da componenti indipendenti detti **Activity**
- Activity
  - Assimilabile ad una schermata
  - Destinata ad eseguire un unico compito
  - Ha un'interfaccia visuale per l'interazione utente
- Un'applicazione che presenta "più schermate" verosimilmente avrà più Activity
  - Varie Activity
  - Passaggio di valori tra Activity
  - Da ogni Activity si può accedere e aprire altre Activity nella stessa o in **altre** applicazioni

Activity (2/2)

- Esempio
  - Varie Activity dell'applicazione per telefonare



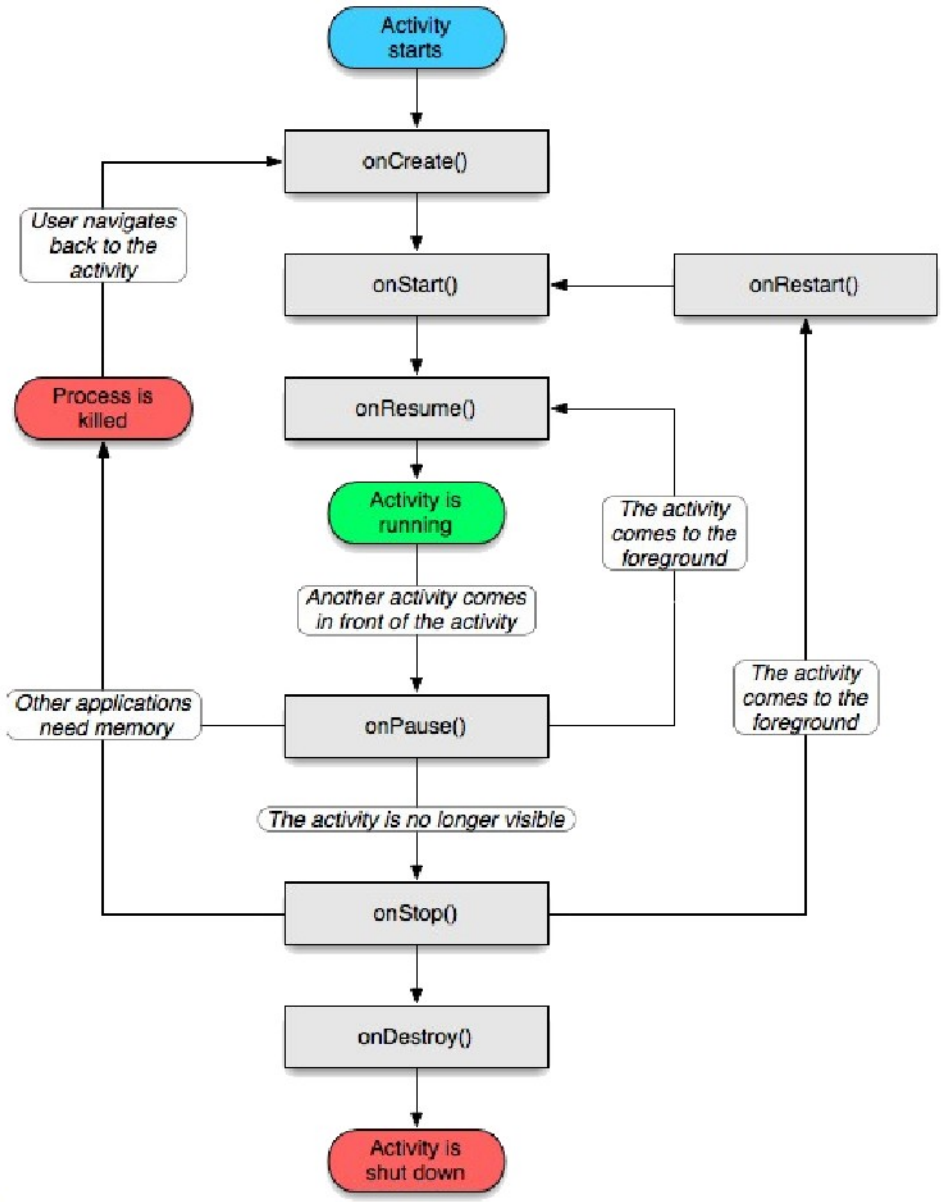




Ciclo di vita di un'Activity (1/4)

- Le Activity sono gestite con uno stack: quella corrente è in cima
- 4 stati
  - **Active/Running**: mostrata sullo schermo e ha focus
  - **Paused**: visibile, ma senza focus (Activity sotto Alert)
  - **Stopped**: completamente oscurata da un altra Activity. Mantiene memoria degli stati; può essere eliminata per liberare risorse
  - **Destroyed**: eliminata dalla memoria. Per rivisualizzarla deve essere nuovamente avviata e ripristinato lo stato

Ciclo di vita di un'Activity (2/4)



- Il ciclo di vita di ogni Activity
  - Inizia quando viene creata in risposta ad un Intent
  - Finisce quando viene distrutto (solitamente o perché ha finito la sua attività o per liberare memoria)
  - Fra nascita e distruzione del componente vi sono stati intermedi che vengono notificati all'Activity

Ciclo di vita di un'Activity (3/4)

- Ciclo di vita dei componenti dell'applicazione non è controllato dall'applicazione, ma **dal sistema**
- Android cerca di "allungare" la vita di ogni componente il più possibile (caching per aumentare la reattività)
- Sistema può chiudere autonomamente parti della applicazione (per es. singole Activity) per liberare risorse
- Se deve essere liberata memoria, avviene una scelta di quali componenti da sopprimere, basata su un ordine di importanza

Ciclo di vita di un'Activity (4/4)

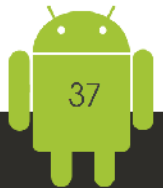
- Ordine di importanza
  - Processo in primo piano
    - Activity mostrata sullo schermo o IntentReceiver in esecuzione
  - Processo non in primo piano
    - Activity visibile, ma non in primo piano (onPause)
  - Servizio
    - Service chiamato con startService
  - Processo background
    - Activity non visibile (onStop)

Intent

- Messaggi scambiati tra i componenti di un'applicazione (es.: Activity) e tra la piattaforma e le applicazioni
  - Notificare eventi
  - Avviare le Activity
- Oggetto Intent: struttura dati passiva
  - Descrizione astratta delle operazioni da eseguire
    - Es.: applicazione vuole mostrare pagina web
    - Dichiara l'Intent di mostare una pagina web
    - Il sistema ricerca l'Activity in grado di eseguire l'operazione
  - Dichiarazione esplicita del componente da avviare
  - Notifica di qualche cosa che è avvenuto
    - Chiamata, batteria scarica, …
  - URI dei dati da elaborare

Navigazione tra Activity

- Navigazione tra schermate (Activity) equivale a risolvere Intent
  - Activity chiama startActivity(myIntent)
  - Il sistema risolve l'Intent e sceglie l'Activity opportuna
  - La nuova schermata (Activity) è informata dell'Intent con cui è avviata
- Risoluzione Intent
  - **Esplicita**: stabilisce univocamente il componente che eseguirà l'Intent. Di solito sono usati per messaggi interni all'applicazione visto che i nomi dei componenti di altre applicazioni non sono noti a priori
  - **Implicita**: non viene specificato il componente e quindi, in generale, si accetta che l'azione venga svolta dal componente designato dal sistema



Risoluzione Intent: esplicita

- Viene definita l'esatta Activity da mandare in esecuzione

```
//Creo Intent
Intent intent = new Intent(ActivityInizio.this, ActivityFine.class);

//Avvio
startActivity(intent);
```

Risoluzione Intent: implicita

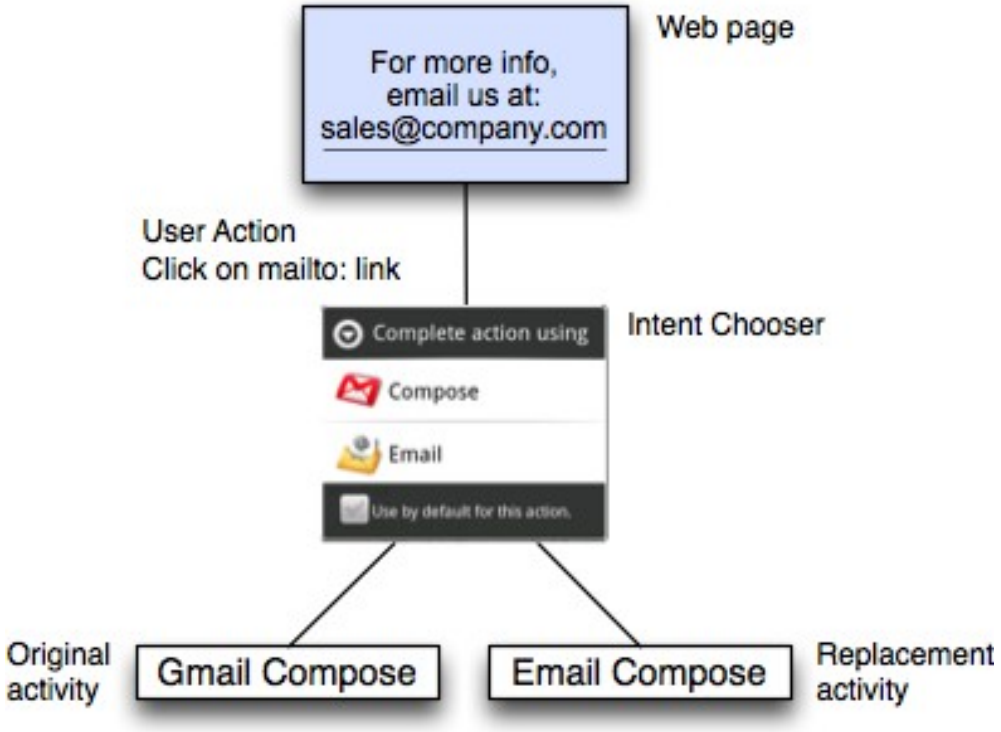
- Si specifica cosa debba essere fatto, il sistema cercherà l'Activity opportuna in base agli IntentFilters definiti

```
//Creo Intent
Intent i = new Intent("android.settings.LOCATION_SOURCE_SETTINGS");
//Avvio
startActivity(i);
```

action

```
<activity android:name="SecuritySettings" android:label="@string/name">
  <intent-filter>
    <action android:name="android.intent.action.MAIN"/>
    <action android:name="android.settings.SECURITY_SETTINGS"/>
    <action android:name="android.settings.LOCATION_SOURCE_SETTINGS"/>
    <category android:name="android.intent.category.DEFAULT" />
  </intent-filter>
</activity>
```

Risoluzione implicita: esempio



Considerazioni

- Vantaggi risoluzione implicita
  - Maggiore generalità
  - Separazione tra operazione da eseguire e componente che la esegue
  - Late binding
  - Possibilità di modificare a piacimento ogni componente della piattaforma Android

Conclusioni

Riassunto

- Cosa abbiamo visto
  - Strumenti di sviluppo
  - Prima applicazione
  - Aspetti peculiari di Android: Activity e Intent
- Abbiamo a malapena intaccato la superficie di Android
  - Molte altre cose interessanti

Riferimenti

- Siti
  - www.android.com
  - developer.android.com
  - code.google.com/android/index.html
- Presentazioni
  - Paolo Zuliani
  - Federico Zanco

Grazie per l'attenzione

Domande?

Luca Vassena, Ph.D.
MoBe srl - www.mobe.it

mobile being