Processamento audio e video su piattaforme Android

Ing. Simone SCARDAPANE

Un seminario per *Laboratorio per l'Elaborazione Multimediale*

intelligent signal processing and multimedia lab

SAPIENZA UNIVERSITÀ DI ROMA

Anno Accademico 2012/2013

Parte 1

1. (Brevissima) Introduzione ad Android
2. Fondamenti di OOP
3. Programmare su Android
4. Processamento Audio
5. Processamento Video

Cos'è Android?

Android è un sistema operativo open-source, disponibile per piattaforme mobile dotate di touch-screen. E' sviluppato da *Google* in collaborazione con il consorzio *Open Handset Alliance*.

Rilasciato per la prima volta nel 2007, è oggi il primo OS per smartphone:

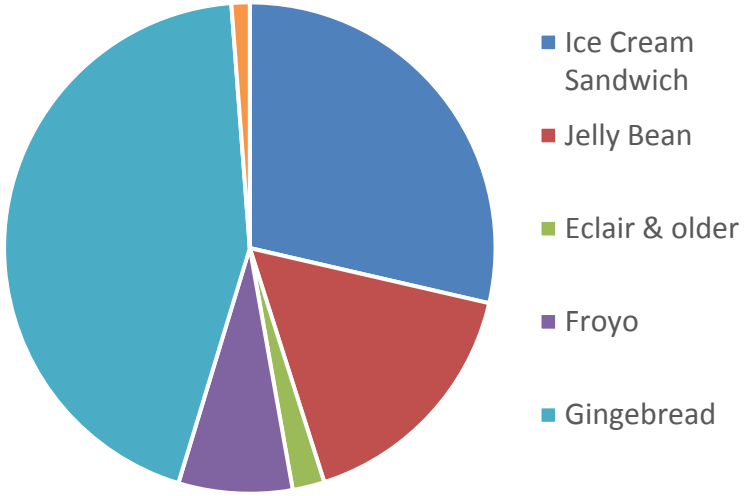
1. 75% dello share complessivo di mercato.
2. 1,3 milioni di attivazioni al giorno.
3. Oltre 700mila applicazioni già disponibili.

(Dati relativi al terzo trimestre del 2012)

Versioni

Una prima sfida alla programmazione su Android è data dalla presenza di numerose versioni in commercio:

Versione	Codename	%
2.2	Froyo	7,5%
2.3.3-2.3.7	Gingerbread	43,9%
4.1	Jelly Bean	14,9%
…		



http://developer.android.com/about/dashboards/index.html

Device Android

Altre problematiche da affrontare sono date dall'eterogeneità dei dispositivi che sfruttano Android:

1. Grandezza dello schermo (generalmente suddivisa in quattro formati).
2. Densità dello schermo.
3. Presenza di sensoristica varia (GPS, Bluetooth…)
4. Presenza di una/due fotocamere.
5. Capacità di calcolo estremamente varie.
6. Possibilità di disconnessione dalla rete.
7. …

Ulteriori Sfide

L'esempio della disconnessione dalla rete è solo uno degli innumerevoli eventi che possono «spezzare» il flusso di una app:

1. L'utente riceve una chiamata.
2. Viene premuto il tasto *Home*.
3. Un servizio viene chiuso da un task manager.
4. Cambia l'orientamento del device.
5. …

A differenza che nel più familiare caso di applicazione desktop, la maggior parte di questi eventi va gestita esplicitamente.

Il mercato di Android

Piattaforma di distribuzione per contenuti vari, tra cui (a seconda del paese):

1. Musica (Play Music),
2. Libri (Play Books),
3. Applicazioni (Play Store),
4. Magazine, Film, Device…

Sono richiesti 25$ all'anno per poter esporre app sullo Store. Ogni app viene prima soggetta ad approvazione delle linee guida.

Se l'app è a pagamento, si ottengono il 70% dei ricavi. E' possibile inserire pubblicità tramite il servizio Google AdMob, che vengono pagate a click effettuati.

Parte 2

1. (Brevissima) Introduzione ad Android
2. Fondamenti di OOP
3. Programmare su Android
4. Processamento Audio
5. Processamento Video

Classi ed Oggetti

Una *classe* è un costrutto di programmazione che contiene al suo interno (tra gli altri) una serie di *campi* e di *metodi*:

```
class Point{
   public int x;
   public int y;

   public double getDistance(){
      return Math.sqrt(this.x*this.x + this.y*this.y);
   }
}
```

Un *oggetto* è un'istanza di tale classe all'interno del codice:

```
Point myPoint;
...
int dist = myPoint.getDistance();
```

Ereditarietà

E’ possibile per una classe ereditare (ed estendere) campi e metodi da una classe base:

```
class Rectangle{
   ...
   double getPerimeter(){ ... };
}

class Square extends Rectangle { ... }
```

La classe Square possiede automaticamente i metodi definiti in Rectangle:

```
Square mySquare;
...
double p = mySquare.getPerimeter();
```

Interfacce

E' anche possibile definire un'interfaccia che una classe deve rispettare:

```
interface Listenable {
   void listen();
}
```

Qualsiasi classe che implementi l'interfaccia deve definire i metodi contenuti nell'interfaccia:

```
class Song implements Listenable{
   void listen() { ... }
}
```

Packages

Tutti questi concetti (e molti altri), se bene usati, permettono di sviluppare codice che sia allo stesso tempo (tra gli altri):

1. Facilmente comprensibile,
2. Estendibile in futuro,
3. Estremamente riusabile.

Ad esempio, possiamo riunire fra loro un insieme di classi ed interfacce *collegate* da utilizzare ovunque siano necessarie le funzionalità da esse svolte. Tali insiemi in Java vengono detti «packages» e possono essere importati all'interno del codice:

```
import MyPackage;
```

SDK

Ogni framework sufficientemente complesso mette a disposizione degli utenti un *kit di sviluppo software* (SDK), che contiene di solito:

1. Un insieme di package per interfacciarsi con le varie funzionalità del framework stesso.
2. Strumenti vari di sviluppo, ad esempio per la scrittura del codice o per il suo corretto debug.
3. Una serie di manuali e/o esempi di sviluppo.

La specifica dell'interfaccia da utilizzare per dialogare con il framework viene detta *application programming interface* (API).

Nel seguito, vedremo proprio cosa mette a disposizione Android e come è possibile sfruttarlo per sviluppare applicazioni.

Parte 3

1. (Brevissima) Introduzione ad Android
2. Fondamenti di OOP
3. Programmare su Android
4. Processamento Audio
5. Processamento Video

Android ADT Bundle

Sul sito ufficiale per gli sviluppatori Android è possibile scaricare l'ADT bundle:

http://developer.android.com/sdk/index.html

Questi contiene una serie di strumenti per lo sviluppo di applicazioni Android:

1. Una versione custom di Eclipse.
2. Android SDK e AVD manager.
3. Tools vari di sviluppo.

E' anche possibile sviluppare utilizzando un IDE di propria scelta scaricando separatamente i vari pacchetti.

Eclipse

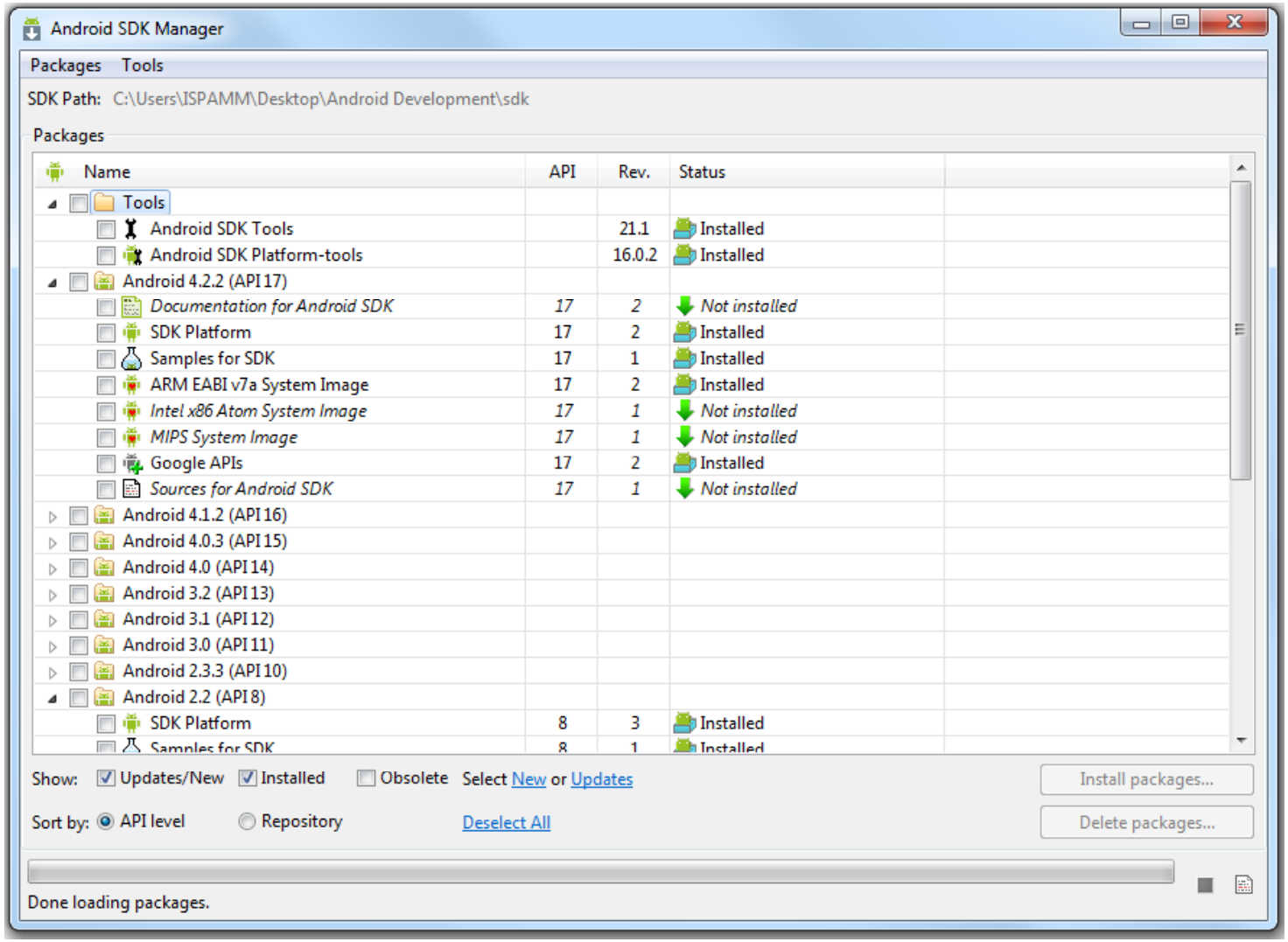
Eclipse è un ambiente di sviluppo multi-linguaggio, sviluppato in Java. E' rilasciato sotto licenza *Eclipse Public License*, libera ed open source.

Uno dei maggiori vantaggi di Eclipse è il suo sistema di plugins che lo rende estremamente estendibile.

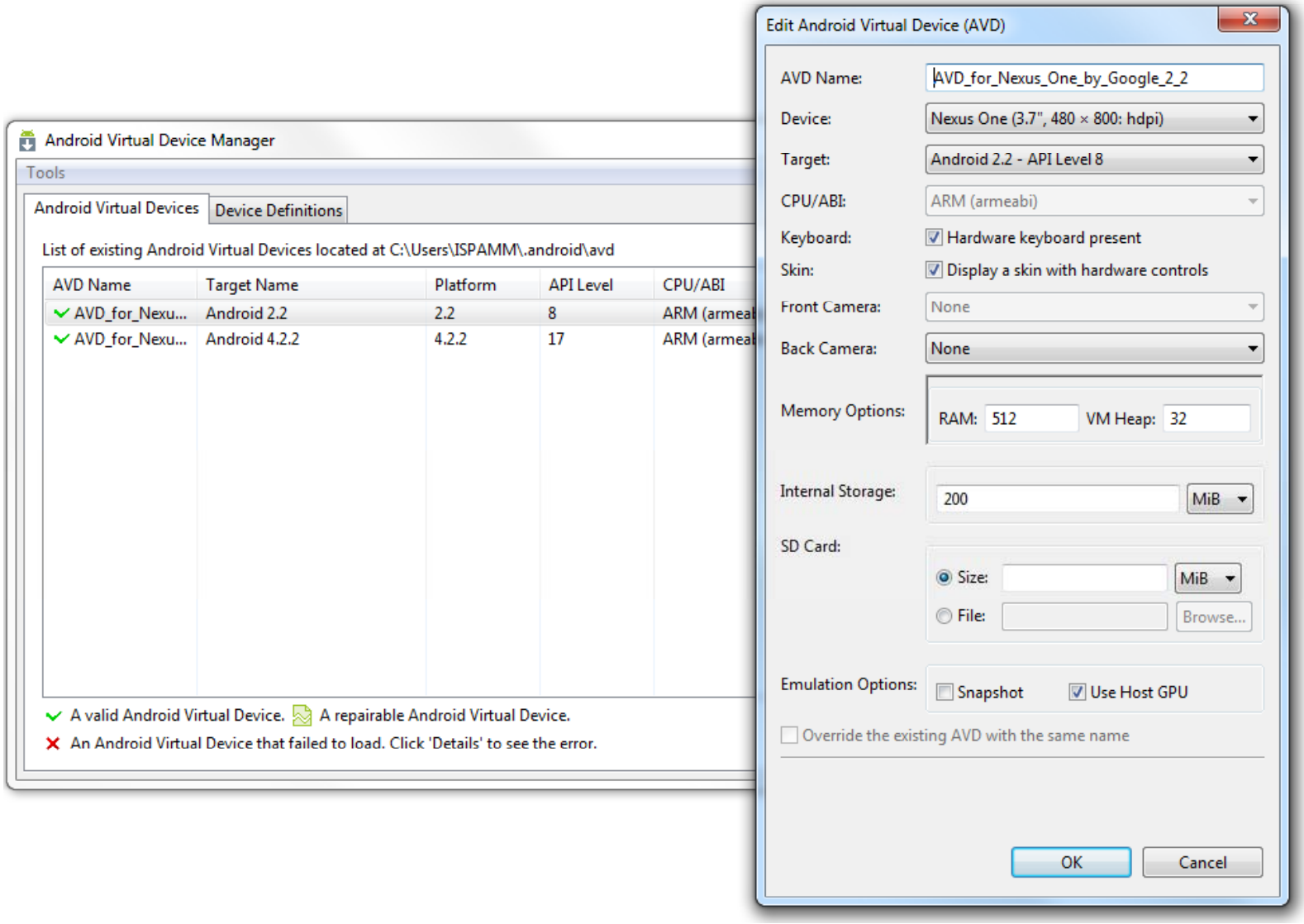
La versione di Eclipse presente nel bundle Android è aumentata dal plugin Google ADT (Android Development Tools), che permette:

1. La creazione rapida di progetti Android.
2. La propotipazione grafica dei propri progetti.
3. Una gestione efficace delle risorse progettuali.

Android SDK Manager



Android AVD Manager



Componenti Principali

Vediamo ora alcuni concetti fondamentali legati allo sviluppo Android, inclusi:

1. Attività, layout delle attività e loro lifecycle.
2. Intenti (impliciti ed espliciti).
3. Dichiarazione del manifesto.
4. Creazione di un progetto su Eclipse.

Toccheremo solo rapidamente molti punti. Per approfondire la maggior parte, è possibile seguire il training di Android:

http://developer.android.com/training/

O la guida più dettagliata:

http://developer.android.com/guide/

Attività

Un'*attività* è un componente dotato di interfaccia con cui l'utente può interagire. Un'applicazione si compone tipicamente di diverse attività che vengono richiamate a seconda delle azioni dell'utente.

La navigazione tra attività si può implementare in diversi modi:

- La pressione del tasto *Back* sul device riporta alla schermata padre.
- E' possibile rispondere a determinati gesti (*swypes*) sullo schermo.
- Si può impostare un layout a tabs.

XML associato all'Attività

Il layout secondo cui sono disposti i componenti è salvato all'interno di un file xml direttamente modificabile dall'utente:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:orientation="horizontal">

    <EditText android:id="@+id/edit_message"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message"
        android:layout_weight="1" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/send_message"
        android:onClick="sendMessage" />
</LinearLayout>
```

Risorse

Tutte le *risorse* dell'applicazione sono salvate in una serie di file xml, es. stringhe, numeri e così via.

A ciascuna di esse è associato un ID che permette di richiamarle all'interno del proprio codice:

```
R.<resource_type>.<resource_name>
```

O di un file xml:

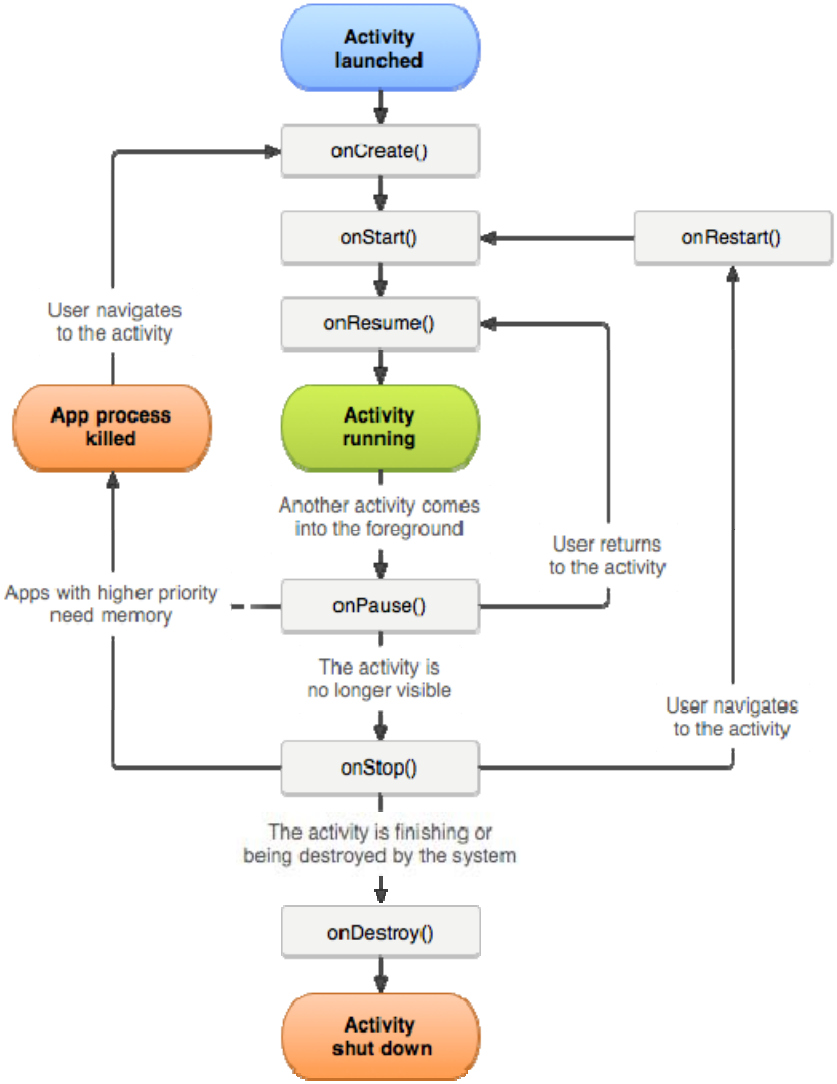
```
@[<package_name>:]<resource_type>/<resource_name>
```

Classe dell'attività

Ovviamente ogni classe ha associato una classe in cui viene dichiarato il suo comportamento:

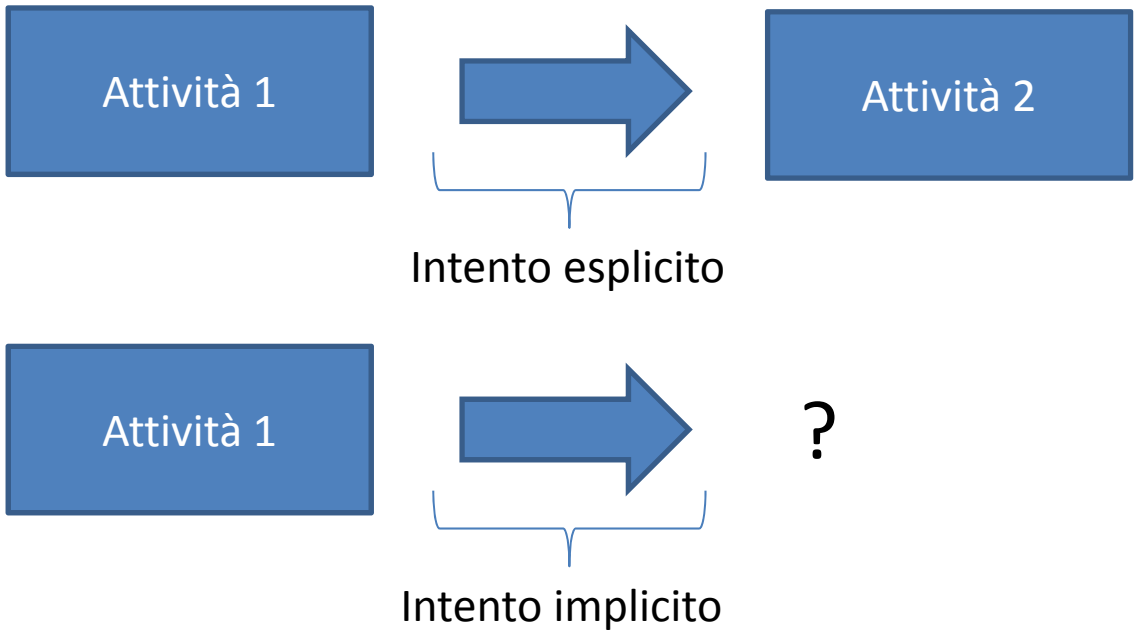
```
package it.uniroma1.ing.ispac.myfirstapp;

import android.app.Activity;

public class MainActivity extends Activity {

    public final static String EXTRA_MESSAGE = "it.uniroma1.ing.ispac.myfirstapp.MESSAGE";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }

    public void sendMessage(View view){
        Intent intent = new Intent(this, DisplayMessageActivity.class);
        EditText editText = (EditText) findViewById(R.id.edit_message);
        String message = editText.getText().toString();
        intent.putExtra(EXTRA_MESSAGE, message);
        startActivity(intent);
    }

}
```

Lifecycle


Activity launched
onCreate()
onStart()
onRestart()
onResume()
Activity running
App process killed
User navigates to the activity
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Intenti

Le attività (così come altri componenti che non vediamo, tra cui i servizi) vengono attivate all'arrivo di un oggetto di tipo *Intento*, che permette la comunicazione fra attività diverse.



Intenti /2

1) Dichiarazione di un intento esplicito:

```
public void sendMessage(View view){
    Intent intent = new Intent(this, DisplayMessageActivity.class);
    EditText editText = (EditText) findViewById(R.id.edit_message);
    String message = editText.getText().toString();
    intent.putExtra(EXTRA_MESSAGE, message);
    startActivity(intent);
}
```

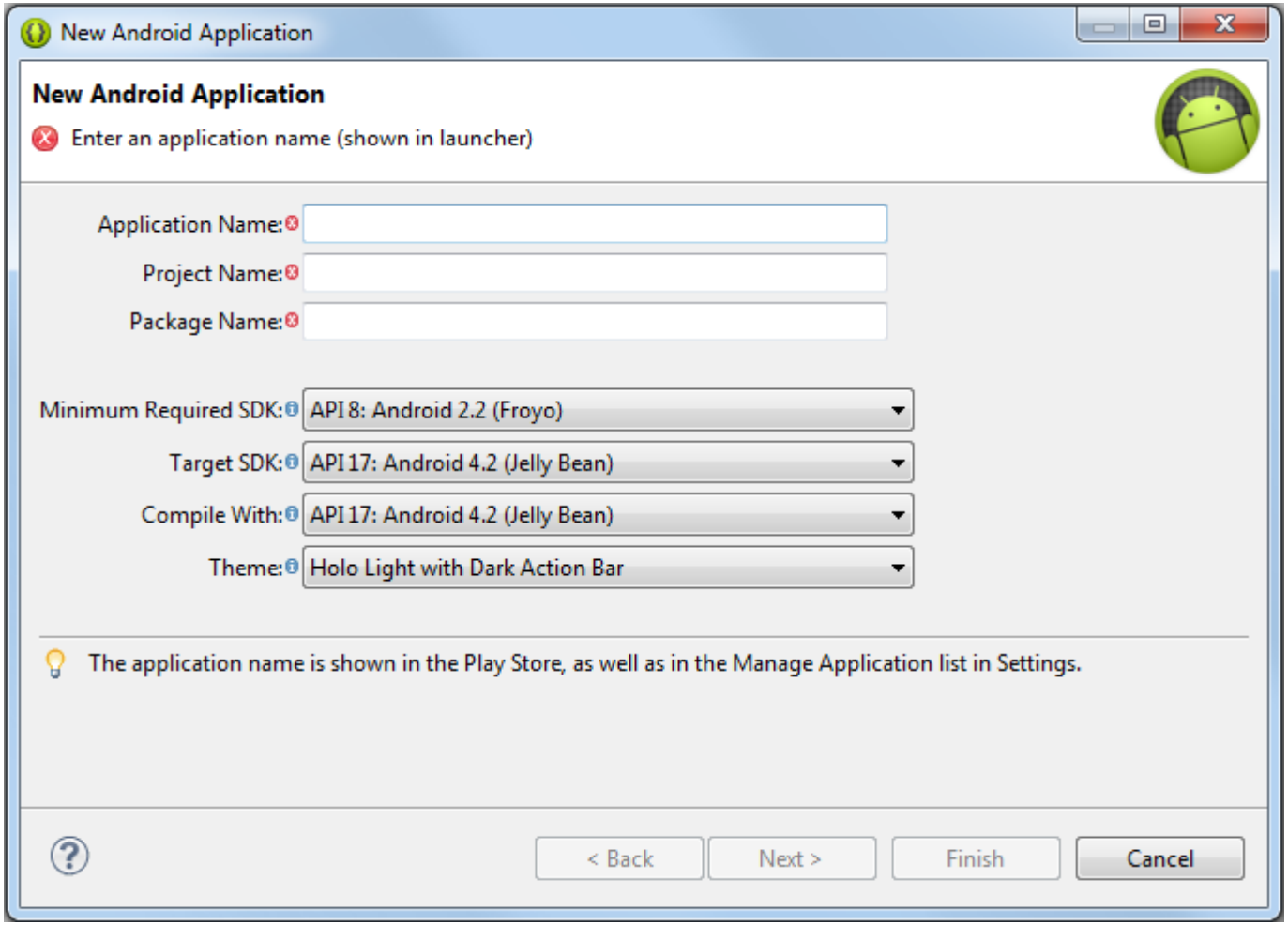
2) Ricezione ed utilizzo dell'intento:

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    Intent intent = getIntent();
    String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
```

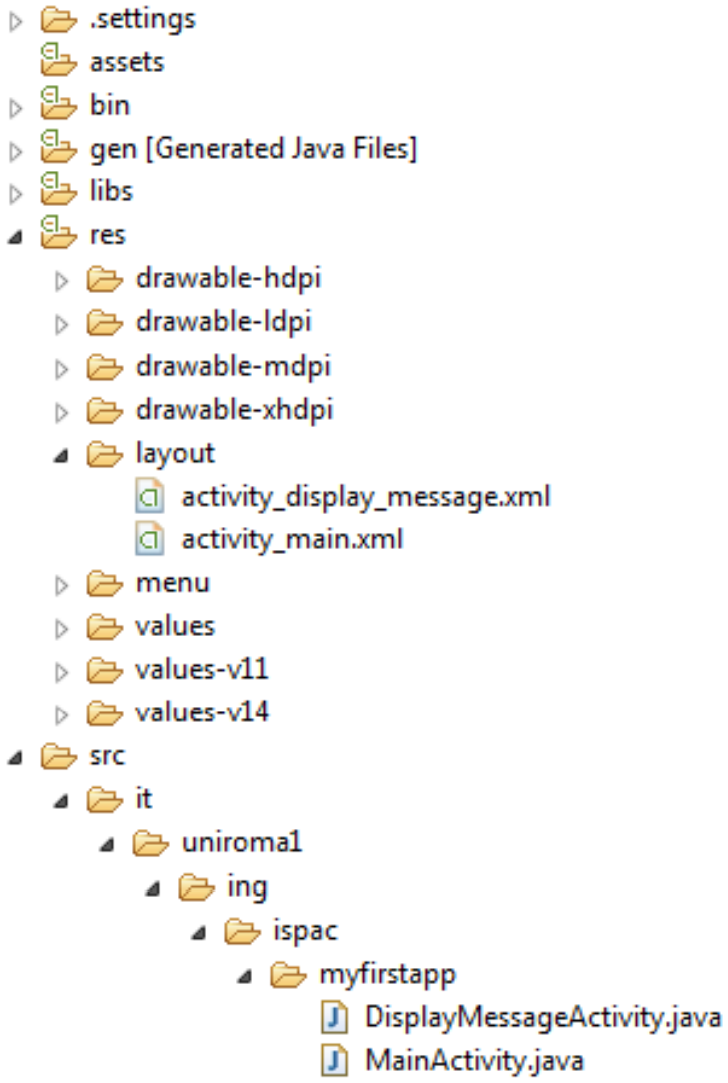
Manifesto

```
<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <activity
        android:name="it.uniroma1.ing.ispac.myfirstapp.MainActivity"
        android:label="@string/app_name" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity
        android:name="it.uniroma1.ing.ispac.myfirstapp.DisplayMessageActivity"
        android:label="@string/title_activity_display_message"
        android:parentActivityName="it.uniroma1.ing.ispac.myfirstapp.MainActivity" >
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="it.uniroma1.ing.ispac.myfirstapp.MainActivity" />
    </activity>
</application>
```

Nuovo progetto su Eclipse



La struttura del progetto



Parte 4

1. (Brevissima) Introduzione ad Android
2. Fondamenti di OOP
3. Programmare su Android
4. Processamento Audio
5. Processamento Video

Stream musicali e volume

- Android organizza i suoni in un insieme di *streams* audio, a ciascuno dei quali è associato un diverso volume.
- Quando l'utente alza o abbassa il volume, viene modificato il volume dello stream attivo.
- Una prima esigenza è quindi quella di identificare quale sia lo stream della vostra applicazione, e selezionarlo come attivo all'avvio:

```
setVolumeControlStream(AudioManager.STREAM_MUSIC);
```

Per approfondire:
http://developer.android.com/training/managing-audio/volume-playback.html

Audio focus

A ciascuno stream è associato un *focus*, che determina quale applicazione può utilizzarlo per riprodurre audio. Vediamo come ottenerlo.

1. La classe *AudioManager* ci permette di gestire i vari stream audio:

```
AudioManager am =
mContext.getSystemService(Context.AUDIO_SERVICE);
```

2. Richiediamo il focus:

```
int result =
am.requestAudioFocus(afChangeListener, AudioManager.STREA
M_MUSIC, AudioManager.AUDIOFOCUS_GAIN);
```

3. Verifichiamo di averlo ottenuto:

```
if (result == AudioManager.AUDIOFOCUS_REQUEST_GRANTED)
{ … }
```

Audio focus /2

Analizziamo i parametri del metodo *requestAudioFocus*:

```
am.requestAudioFocus(afChangeListener, AudioManager.STREAM_M
USIC, AudioManager.AUDIOFOCUS_GAIN);
```

1. Cominciamo dal secondo: si tratta dello stream di cui volete ottenere il focus.
2. Il terzo parametro è la modalità con cui volete il focus:

    1. `AUDIOFOCUS_GAIN` per un focus di durata sconosciuta.
    2. `AUDIOFOCUS_GAIN_TRANSIENT` per un focus di durata breve.
    3. `AUDIOFOCUS_GAIN_TRANSIENT_MAY_DUCK` per un focus di durata breve, nel quale l'app che ha correntemente il focus può limitarsi ad abbassare il volume.

http://developer.android.com/training/managing-audio/audio-focus.html

Audio focus /3

Il primo parametro è l'oggetto che deve essere avvisato al successivo cambio di focus.

Come funziona?

1. La vostra classe che riproduce i suoni implementa l'interfaccia **OnAudioFocusChangeListener**.
2. Questa definisce un metodo che l'AudioManager richiamerà e che la classe deve implementare: *onAudioFocusChange(int focusChange)*.

Questa è una tecnica molto generale, implementata in tutti i casi in cui una classe deve essere richiamata al verificarsi di un dato evento.

E' possibile definire un listener anche in una seconda maniera, detta *dichiarazione anonima* o *inline*. Non la vediamo in questo seminario.

MediaPlayer

Adesso che abbiamo gestito (parte) dei problemi di interazione con le altre applicazioni, vediamo come riprodurre un suono.

La classe *MediaPlayer* è la principale risorsa fornitaci da Android:

```
MediaPlayer mediaPlayer =
        MediaPlayer.create(context, R.raw.sound_file_1);
mediaPlayer.start();
```

Possiamo riprodurre anche contenuti dal web o in streaming.

Ovviamente terminata la riproduzione dobbiamo chiudere il media player e rilasciare il focus.

http://developer.android.com/guide/topics/media/mediaplayer.html

Considerazioni Varie

In realtà rimangono diverse questioni che vanno affrontate in un'implementazione realistica:

1. Se vi connettete in streaming, dovete dichiarare la necessità di utilizzare la connessione sul manifesto:

```
<uses-permission
android:name="android.permission.INTERNET" />
```

2. In generale, la preparazione dello stream dovrebbe essere fatta in maniera asincrona per non bloccare l'interfaccia grafica.

3. Assicurarci che il formato sia supportato:
   http://developer.android.com/guide/appendix/media-formats.html

4. E' possibile utilizzare MediaPlayer anche per aggiungere alcuni effetti sull'audio tramite il metodo **attachAuxEffect** (int effectId).

Registrazione Audio

Android possiede la classe *MediaRecorder* per gestire la registrazione di uno stream audio:

```
mRecorder = new MediaRecorder();
mRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
mRecorder.setOutputFormat
        (MediaRecorder.OutputFormat.THREE_GPP);
mRecorder.setOutputFile(mFileName);
mRecorder.setAudioEncoder
        (MediaRecorder.AudioEncoder.AMR_NB);
```

Maggiori dettagli:

http://developer.android.com/guide/topics/media/audio-capture.html

SoundPool

Se la vostra applicazione deve gestire numerosi suoni di breve durata, un'alternativa a gestirli manualmente con MediaPlayer è la classe *SoundPool*.

SoundPool è sviluppata per ottimizzare la latenza di riproduzione, il numero di suoni che possono essere riprodotti simultaneamente ed in generale la gestione della CPU.

Ad esempio, in un gioco è possibile sfruttarla per pre-caricare tutti i suoni in memoria durante il caricamento iniziale.

http://developer.android.com/reference/android/media/SoundPool.html

Concludendo

Utilizzare funzionalità native di Android vi garantisce (almeno ad un certo livello) di mantenere determinate prestazioni. Ad esempio, su Jelly Bean è stato fatto un grande sforzo a livello implementativo per abbassare la soglia di latenza di riproduzione.

Se avete particolari esigenze di processamento a basso livello, avrete bisogno di una libreria esterna a seconda dell'obiettivo, ad esempio:

http://www.surina.net/soundtouch/

Un'alternativo è l'utilizzo di Android NDK per sfruttare codice scritto in C/C++:

http://developer.android.com/tools/sdk/ndk/index.html

Parte 5

1. (Brevissima) Introduzione ad Android
2. Fondamenti di OOP
3. Programmare su Android
4. Processamento Audio
5. Processamento Video

VideoView

Teoricamente la classe MediaPlayer può essere usata per riprodurre video, ma in questo caso diventa molto più complessa da gestire.

Un'ulteriore astrazione è data invece dalla classe *VideoView* che ci nasconde tali complessità:

```
VideoView videoView =
(VideoView)findViewById(R.id.VideoView);
videoView.setVideoPath("/sdcard/something.3gp");
videoView.start();
```

Notiamo che l'oggetto videoView è un componente inserito all'interno di un'attività che richiamiamo nel codice tramite il suo ID.

VideoView /2

Possiamo registrare un Listener per effettuare determinate azioni al termine del video stesso:

```
videoView.setOnCompletionListener
        ((OnCompletionListener) this);
```

Ovviamente possiamo registrare un Listener per qualsiasi altro evento.

Possiamo anche inserire effetti base sui nostri video:

http://developer.android.com/reference/android/media/effect/package-summary.html

Registrazione di Video

Concludiamo vedendo come possiamo registrare un video.

Inviamo un intento implicito con l'azione da eseguire:

```
private void dispatchTakeVideoIntent() {
    Intent takeVideoIntent = new
        Intent(MediaStore.ACTION_VIDEO_CAPTURE);
    startActivityForResult
        (takeVideoIntent, ACTION_TAKE_VIDEO);
}
```

Riceviamo l'intento di ritorno e mostriamo il video:

```
private void handleCameraVideo(Intent intent) {
    mVideoUri = intent.getData();
    mVideoView.setVideoURI(mVideoUri);
}
```

http://developer.android.com/training/camera/videobasics.html

Grazie per l'attenzione. ☺

Domande?

(Buono sviluppo su Android!)