Prof. Roberto De Prisco

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno – Autunno 2026

# ANDROID Mobile Programming

1

1

## per favore…





…o almeno  …e NON RISPONDERE!!!!

Scrivere un'app che attiva la modalità silenziosa durante le lezioni!

2

## Info corso



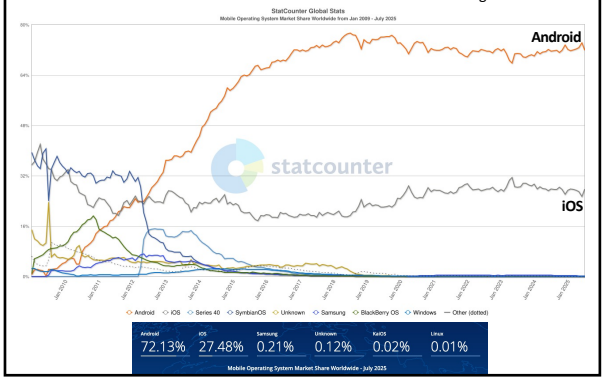
- Prof. Roberto De Prisco
  - studio: 4° piano, studio 58
    - numerazione Dip. di Informatica
  - robdep@unisa.it
- Orario lezioni
  - Martedì 15:30-17:30 PTE014
  - Mercoledì 09:00-11:00 PTE014
- Ricevimento
  - Lunedì 13:30-15:30
  - Giovedì 13:30-15:30

3

## Vendite per OS (worldwide)



Dati da: gs.statcounter.com




4

## Telefoni Android prodotti da

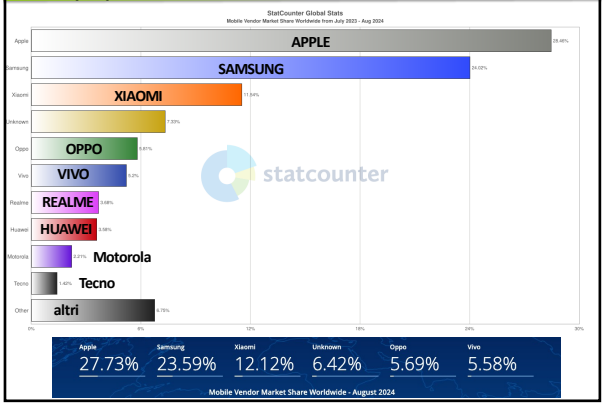


SAMSUNG, oppo, HUAWEI, xiaomi, honor, realme, LG, Google, acer, htc, SONY, ALCATEL, e altri …

5

## Vendite per marca






6

## Cosa insegna il corso

Come scrivere app per Android!!!




7

## Ambiente di sviluppo

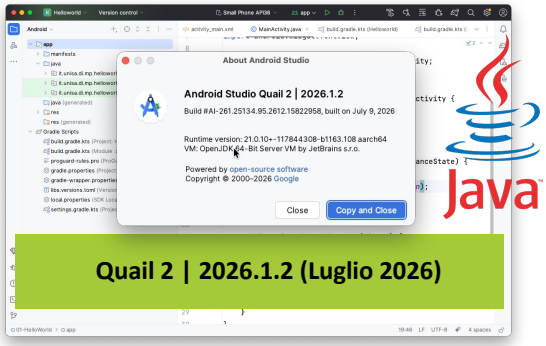
- Android Studio
  - facile da istallare
  - molti tools
  - Quail 2, 2026.1.2
    - Quail, quaglia




- Istallazioni delle app
  - facile
  - non richiede nessuna registrazione
- Moltissime risorse online

8

## Android Studio




Vedremo tutto ciò che serve per scrivere app …

9

## Linguaggio

Java

Kotlin

Corso Programmazione 2: **Java**

10

## Argomenti

- Piattaforma Android
- Android Studio
- Emulatore

| | | |
|---|---|---|
| • Layout | • Permessi | • Grafica |
| • Views | • Alarms | • Sensori |
| • Listener | • Frammenti | • Multimedia |
| • Intent | • Networking | • Data storage |

- Content Provider
- Broadcast
- Services

11

## Risorse didattiche

- **Lezioni!!!!**



- Google
  - http://developer.android.com
  - http://developer.android.com/guide
  - http://developer.android.com/training
- Books
  - BigNerd Ranch (in inglese)
    - Ultima edizione (5ª) usa Kotlin al posto di Java
  - http://www.bignerdranch.com/



- Cursera
  - ottimo video corso (in inglese)
  - https://www.coursera.org/course/android



- Googling!

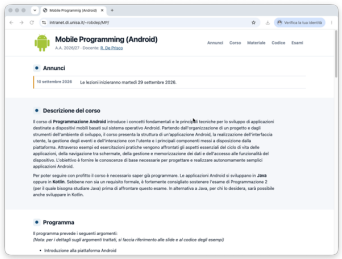
12

## Sito e piattaforma S3



- Sito
  - Informazioni
  - Annunci
  - Codice app
  - Esempi esame
- S3
  - Esami
  - Risultati

http://intranet.di.unisa.it/~robdep/MP/




13

## Contest



- La migliore app sviluppata DURANTE il corso verrà premiata con un dispositivo Android!
  - Per partecipare occorre
    - frequentare il corso
    - sviluppare in gruppi di 2 persone max
    - consegnare l'app una settimana prima del contest
    - app di una certa complessità
- Sponsorizzato da eTuitus
  - www.etuitus.it
  - commissione: docente, eTuitus



14

## Esame



- L'app sviluppata per il contest sostituisce la prova di laboratorio
  - ma non lo scritto
- Esame
  - Scritto
  - Laboratorio
    - vengono ammessi gli studenti che superano lo scritto
  - Orale a discrezione del docente

15

## Come contattare il docente



- Fine lezione (o nella pausa)
- Orario di ricevimento
- Email
  - Risposta non garantita, dipende dalla domanda!
  - Condizione necessaria: il messaggio deve contenere il nome del mittente
  - Appuntamento
- **NON** telefonare

16

## Domande?





17

## … si parte






18

ANDROID Mobile Programming

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno – Autunno 2026

# La piattaforma Android

19

19

## La piattaforma Android





- Sistema software per telefonini e tablet
  - OS kernel
  - Librerie di sistema
  - Framework per le applicazioni
  - Applicazioni di base
- SDK per lo sviluppo di nuove applicazioni
  - librerie
  - tool di sviluppo
  - documentazione
    - manuali
    - esempi

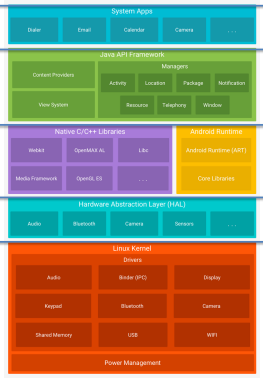
http://developer.android.com/training

20

## L'architettura Android








Applications

Java API

Librerie e Android Runtime

Hardware Abstraction Layer

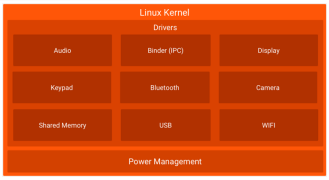
**Kernel Linux**

21

## Kernel Linux







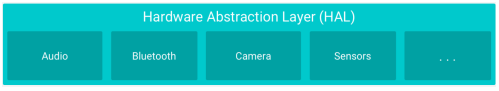

- Fornisce i servizi di base del sistema operativo
  - filesystem
  - gestione della memoria e dei processi
  - gestione dell'interfaccia di rete
  - driver per le periferiche
- Servizi specifici per Android
  - gestione della batteria
  - gestione della memoria condivisa
  - low memory killer
  - interprocess communication e altre

22

## Hardware Abstraction Layer








- HAL: Hardware Abstraction Layer
  - Interfacce standard per esporre le capacità hardware ai servizi di livello superiore
    - Audio
    - Bluetooth
    - Fotocamera
    - Sensori
    - …

23

## Java





- App Android sono scritte in Java e/o Kotlin
- Librerie Java forniscono classi pronte per l'uso:
  - classi di base: java.*, javax.*
  - classi per le app: android.*
  - Internet/web services: org.*
  - Unit testing: junit.*

24

## File dex e ART

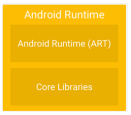


- App:
  - Scritte in Java (e/o Kotlin)
  - Compilate in file Java Bytecode
  - Un tool, DX, trasforma i file bytecode in un singolo file Dex Bytecode (classes.dex)
  - Il file classes.dex contiene anche tutti i file di dati necessari e viene istallato sulla target device
  - ART Virtual Machine esegue il file dex
- dex = Dalvik (Machine) EXecutable

25

## Android runtime: ART e Dalvik VM









- ART: Android Runtime è una VM specifica per sistemi Android
  - CPU meno veloci (rispetto a un PC)
  - Meno RAM
  - Batteria con durata limitata
- ART: da 5.0 API level 21, miglioramenti rispetto a Dalvik
- Dalvik: Virtual machine per API level < 21
- App che funzionano bene su ART dovrebbero funzionare bene anche su Dalvik
  - Il contrario no
- dex = Dalvik Executable (format)

26

## ART e Dalvik

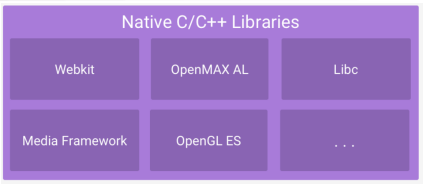


- Dalvik
  - Just-In-Time compilation
    - compila a run-time
  - Garbage collection
    - a volta cause pause nelle app in esecuzione
  - Richiede più memoria
  - Consuma più batteria
- ART
  - Ahead-Of-Time compilation
    - compila prima di eseguire
    - spesso all'istallazione
  - Garbage collection
    - più efficiente
  - Richiede meno memoria
  - Consuma meno batteria

27

## Librerie native



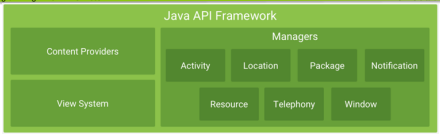



- Molte componenti Android necessitano di librerie native
  - Webkit
  - Libc
  - openGL ES
  - … (Surface manager, Media framework)

28

## Application framework



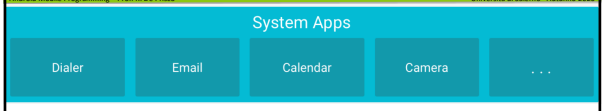



- Funzionalità del SO Android vengono esposte tramite un API
  - View System
    - fornisce gli elementi di base per le interfacce utente
      - icone, testo, bottoni, ecc.
  - Content Providers
    - Per accedere a dati di altre app, per es. ai contatti della rubrica
  - Package manager
    - gestisce l'istallazione delle app sul dispositivo mobile
  - Activity Manager
    - gestisce il ciclo di vita delle applicazioni
    - permette di passare da un'applicazione all'altra

29

## Applicazioni (app)






- Applicazioni già presenti nel sistema
  - Home: Main screen
  - Contatti
  - Telefono
  - Browser
  - Email client
  - Media player
  - … altre
- Ovviamente … si possono scrivere nuove app!

30

## Versioni



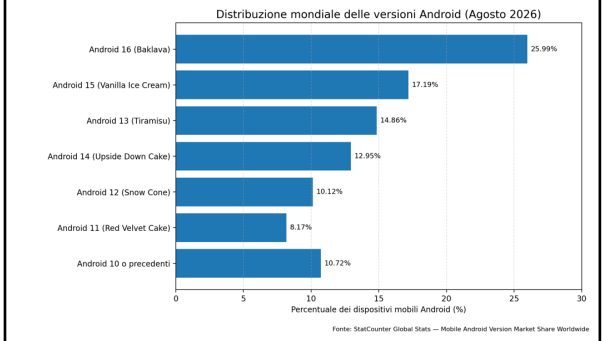
| | NOME | VERSIONE | KERNEL LINUX | DATA RILASCIO | LIVELLO API |
|---|---|---|---|---|---|
| A | Android 1.0 (Astro?) | 1.0 | 2.1 | 23/09/2008 | 1 |
| B | Android 1.1 (Bender?) | 1.1 | 2.6 | 09/02/2009 | 2 |
| C-D | Cupcake-Donut | 1.5/1.6 | 2.6.27/29 | 27/04-15/09/2009 | 3-4 |
| E-F-G | Eclair-Froyo-Gingerbread | 2.0 – 2.3.7 | 2.6.29-2.6.35 | 2009-2010 | 5-10 |
| H | Honeycomb | | | /2011 | 11-13 |
| I | Ice Cream Sandwich | | | /2011 | 14-15 |
| J-K | Jelly Bean-KitKat | | | 12-10/2013 | 16-20 |
| L | Lollipop | | | /2014 | 21-22 |
| M | Marshmallow | | | /2015 | 23 |
| N | Nougat | | | /2016 | 24 – 25 |
| O | Oreo | | | /2017 | 26 – 27 |
| P | Pie | | | /2018 | 28 |
| Q | Quince Tart | | | /2019 | 29 |
| R | Red Velvet Cake | | | /2020 | 30 |
| S | Snow cone | | | /2021 | 31-32 |
| T | Tiramisù | 13 | 5.4-5.10 | 15/08/2022 | 33 |
| U | Upside Down Cake | 14 | 5.10-5.15, 6.1 | 04/10/2023 | 34 |
| V | Vanilla Ice Cream | 15 | 6.6 | 03/9/2024 | 35 |
| B | Baklava | **16 (stabile)** | **6.12** | **10/6/2025** | **36** |
| C | Cinnamon Bun | **17 (beta)** | **6.18** | **16/6/2026** | **37** |




31

## Versioni

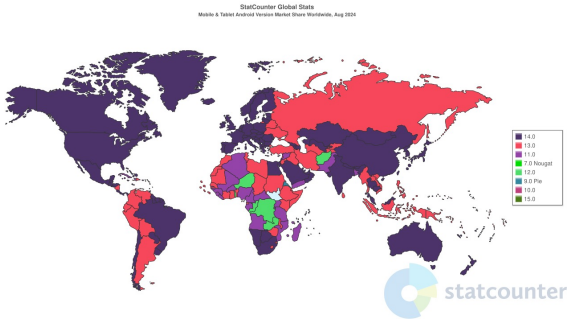





32

## Versioni

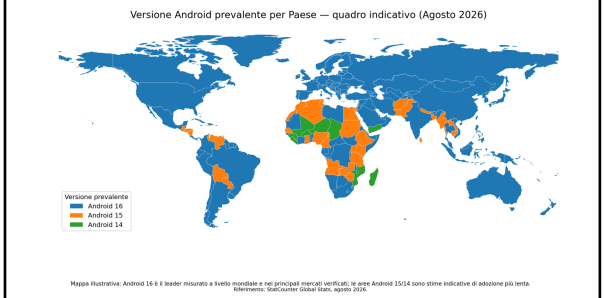





33

## Versioni






34



# ANDROID Mobile Programming



# Android Developer Tools



35

## Android Studio

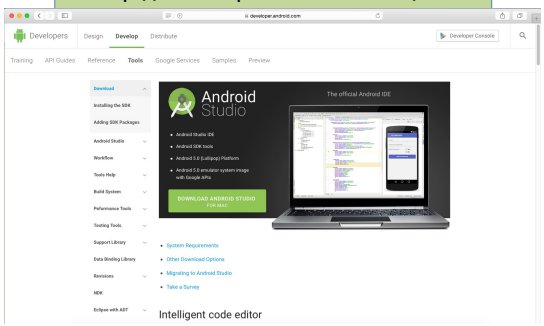


- Istallare l'ADT
- L'interfaccia
- L'emulatore Android
- Strumenti per il debug
- Altri strumenti

36

## Istallazione ADT

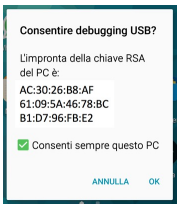
http://developer.android.com/sdk




37

## Modalità sviluppatore

- Modalità sviluppatore
  - Info dispositivo, Versione build
    - Click 7 volte
- Comparirà il menu Opzioni Sviluppatore
- Debug USB
  - Attivare
  - Dare il consenso per l'accesso




38

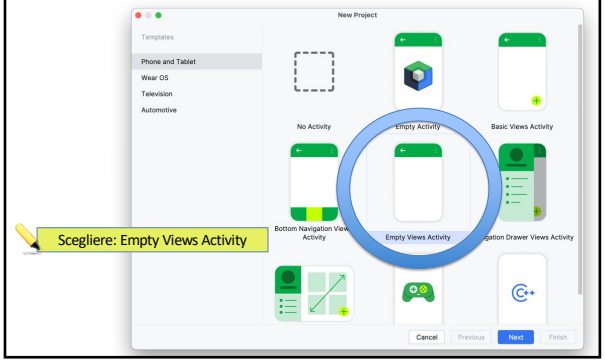
## Listeners

- Gli oggetti della classe View hanno dei metodi "listeners"
  - sono in "ascolto" per entrare in azione quando si verifica un evento specifico
- Ad esempio
  - un pulsante ha il metodo onClick che viene eseguito quando l'utente preme il pulsante

39

## New Project: Empty Views Activity

- File → New → New Project




40

## Prima app: CiaoMondo

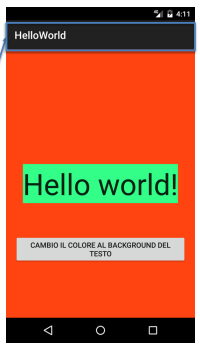
- Visualizza un saluto al mondo!



Icona: New→Image Asset Creator

01-CiaoMondo

- Vediamo
  - il codice
- Scriviamo l'app!
  - Tour di Android Studio

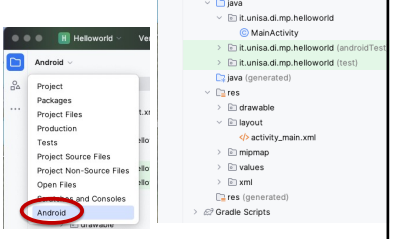



Action bar: compare solo se il Tema lo prevede

41

## CiaoMondo

- Manifesto
  - informazioni generali sull'app
    - permessi, attività, icona, …
- Java
  - file sorgenti
- res, risorse
  - drawable
  - layout
  - values
  - menu
  - mipmap

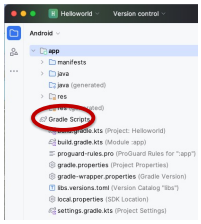



42

## CiaoMondo

- Gradle
  - build system
- Informazioni
  - dipendenze da librerie (libs.versions.toml)

```
[versions]
agp = "8.11.1"
junit = "4.13.2"
junitVersion = "1.2.1"
espressoCore = "3.6.1"
appcompat = "1.7.1"
material = "1.12.0"
activity = "1.10.1"
constraintlayout = "2.2.1"
```



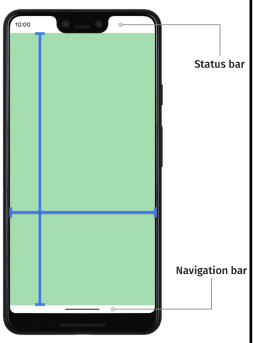
  - altre informazioni su come compilare

```
buildTypes {
    release {
        minifyEnabled false
        proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
    }
}
```

43

## Edge-to-edge e insets

- **da API 35**
  - view a tutto schermo
    - edge-to-edge
- Insets
  - Aree sovrapposte
    - status bar
    - navigation bar
    - menu laterali
    - tastiere soft




44

## Edge-to-edge e insets

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable( $this$enableEdgeToEdge: this);
        setContentView(R.layout.activity_main);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), ( View v,  WindowInsetsCompat insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });
    }
}
```

02-Insets

- setPadding
  - evita le sovrapposizioni con gli insets

45

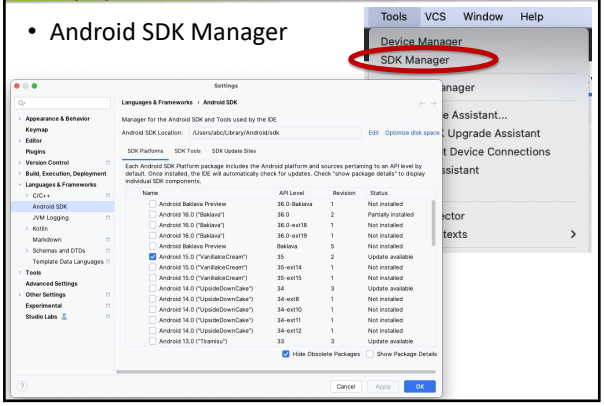
## Activity e AppCompatActivity

- Activity
  - Classe base parte del pacchetto android.app.
  - Fornisce il ciclo di vita standard (onCreate, onStart, onResume, ecc.)
  - e la gestione di base della UI.
  - Non include funzionalità di compatibilità o supporto ai temi/material design introdotti con le prime versioni di Android
- AppCompatActivity
  - Estende FragmentActivity (che a sua volta estende Activity)
  - definita in AndroidX: androidx.appcompat.app.AppCompatActivity.
  - Fa parte della Support Library / AndroidX AppCompat.
  - Offre retrocompatibilità: permette di usare componenti e funzionalità moderne (Toolbar, Material Design, ActionBar, temi chiari/scuri, ecc.) anche su versioni più vecchie di Android.
- In pratica:
  - Con Activity, rischi che alcune funzionalità moderne non funzionino su dispositivi vecchi.
  - Con AppCompatActivity, hai compatibilità estesa e accesso alle API moderne in modo uniforme.

46

## SDK

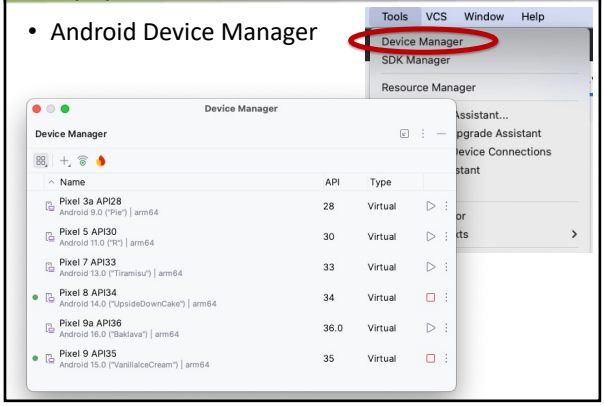
- Android SDK Manager




47

## Emulatori (Device virtuali)

- Android Device Manager




48

## Emulatore vs. real device

- Real device
  - veloce, facile gestire l'input (es. rotazioni display)
  - l'esecuzione è reale

Attivare modalità sviluppatore e debug USB!!!
7 click su Info dispositivo→Versione build
Poi attivare USB debug in opzioni sviluppatore

- Emulatore
  - lento (a volte molto), alcune operazioni sono difficoltose
  - è comunque un "simulatore"
  - possono esserci dei bug
  - Facile creare situazioni particolari:
    - batteria scarica
    - arrivo di un messaggio

49

## Emulatore

- telnet

```
~> telnet localhost 5554
Trying ::1...
telnet: connect to address ::1: Connection refused
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Android Console: Authentication required
Android Console: type 'auth <auth_token>' to authenticate
Android Console: you can find your <auth_token> in
'/Users/abc/.emulator_console_auth_token'
OK
auth fY82+IWngfr0xrSa
Android Console: type 'help' for a list of commands
OK
sms send 3331234567 "Ciao!"
OK
network speed edge
OK
help
Android console commands:
    help|h|?
    help-verbose
    ping
    event
    geo
    gsm
    cdma
    crash
    crash-on-exit
    kill
    network
    power
```



50

## Emulatore

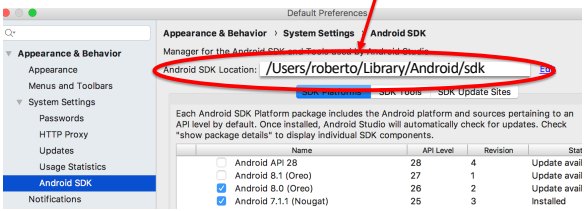
- Comunicazione fra due emulatori




51

## adb

- adb = Android Debug Bridge
  - tool istallato con l'SDK (Tools→SDK Manager)
  - nella directory "platform-tools"




52

## adb e porta telnet

```
platform-tools> adb devices
List of devices attached
emulator-5556 device
emulator-5554 device
```

https://developer.android.com/studio/command-line/adb

53

## Copiare un apk istallato

- Controllare (individuare) il nome del "package"

```
adb shell pm list packages
```

- Individuare il path di istallazione

```
adb shell pm path <package-name>
```

- Copiare l'apk

```
adb pull <full/path/of/the.apk>
```

54

ANDROID
Mobile Programming



# Layouts



55

## Layouts



- Layout
  - Definiscono l'aspetto grafico dell'interfaccia utente
- Si possono definire in due modi
  - Con un file XML
  - In modo programmatico*
- Non sono mutuamente esclusivi
  - si possono usare in sinergia




*programmaticamente, nel gergo Android, significa attraverso delle istruzioni nel programma (eseguite a runtime), quindi sono utili per gestire layout dinamici

56

## Layouts



- XML
  - vantaggi
    - facile da specificare
    - separa in modo netto la definizione dell'UI dal codice dell'applicazione (facile fare modifiche)
  - svantaggi
    - elementi statici
- Programmatico
  - vantaggi
    - dinamico, si può facilmente adattare
  - svantaggi
    - dobbiamo gestire il layout nel codice dell'applicazione

57

## Layout – elementi base (esempi)



- TextView

```
<TextView android:id="@+id/testo"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Ciao, sono un textView" />
```

Ciao, sono un TextView

- ImageView

```
<ImageView android:id="@+id/image"
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:src ="@drawable/dolphin256" />
```



58

## Layout – elementi base (esempi)



- Button

```
<Button android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Ciao, sono un pulsante"
    android:background="#00FF00" />
```

Ciao, sono un Pulsante

- Switch

```
<Switch android:id="@+id/mySwitch"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Seleziona on/off"
    android:checked="true" />
```

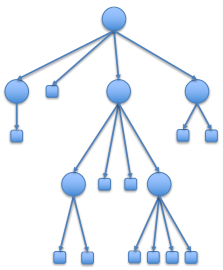
Seleziona on/off

59

## Layout - ViewGroup



- Gruppi di altri elementi
  - sia di base che altri gruppi
- Frame
- LinearLayout
  - orizzontali e verticali
- RelativeLayout
- ConstraintLayout
- TableLayout e GridView



60

## Layout - XML

Un solo elemento "radice" in ogni file XML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >
    <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button" />
</LinearLayout>
```




Elementi arbitrari contenuti nell'elemento radice

61

## Layout - XML - attributi

- Ogni elemento (View o ViewGroup) supporta degli attributi
  - specificano l'aspetto grafico
  - specificano dove visualizzare l'elemento
  - forniscono informazioni
- Es. TextView
  - textSize
- Alcuni attributi sono comuni a tutti gli elementi
  - altri sono specifici

62

## Layout - XML - attributi

- ID (creazione)
  android:id="@+id/text"
  @: il resto della stringa deve essere interpretato
  - android:layout_width="30px"
    - se non c'è @ il valore è quello letterale

  +: specifica che stiamo creando (aggiungendo) un nuovo identificatore (id) il cui nome è text
- ID (riferimento)
  android:id="@id/text"
  senza il + è un riferimento ad un ID esistente

63

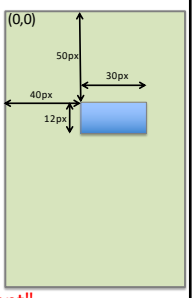
## Layout - XML - attributi

- Layout parameters
  - layout_*something*
- Ogni View ha dei parametri di layout
  - appropriati per il ViewGroup a cui la View appartiene
  - alcuni sono comuni a tutti i tipi di View
    - layout_width
    - layout_height
  - altri hanno significato solo per alcuni tipi
    - layout_alignParentTop

64

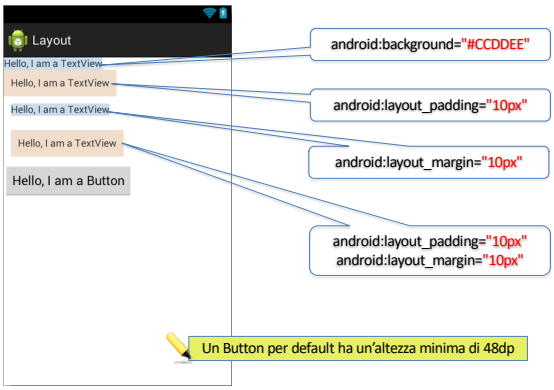
## Layout – posizione e grandezza

- Una view è un rettangolo
  - posizione: angolo in alto a sinistra
  - dimensione: larghezza ed altezza
- Posizione (relativa al parent)
  - determinata dal layout
- Dimensione
  - android:layout_width="30px"
  - android:layout_heigth="12px"
  - android:layout_width="match_parent"
  - android:layout_heigth="wrap_content"




65

## Layout – padding e margini



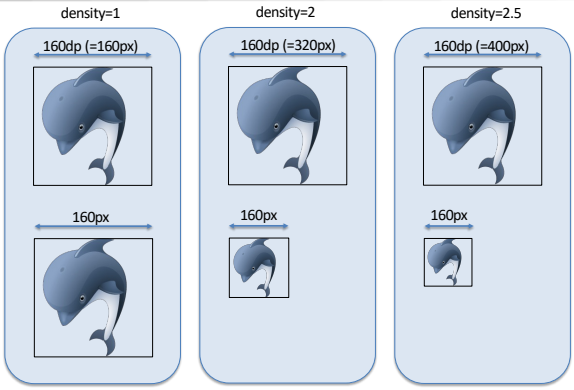

66

## Misure pixel: px vs. dp

- Screen size
  - grandezza reale del display (es. 4'')
- Screen density
  - Quanti pixel ci sono nell'unità di area
    - raggrupppati in: *low*, *medium*, *high* e *extra high*
    - es 240 dpi = 240 dot-per-inch
- px = pixel reali
  - es. 240 dpi x 4'' => 960 pixel
- dp (dip) = density independent pixels
  - dimensione calcolata su una densità di 160 dpi
    - un "dp" ha le dimensioni di un "px" a 160 dpi
  - la dimensione non dipenderà dalla densità reale

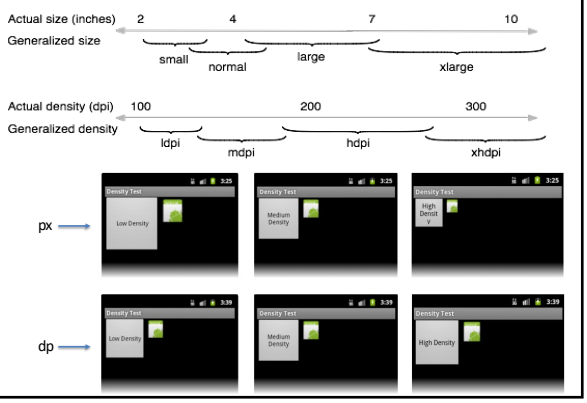
67

## Misure Pixel




68

## Density




69

## Alternative per i "drawable"

- Una buona app dovrebbe fornire alternative per gli oggetti da disegnare (drawable)
- Esempio: l'icona dell'applicazione dovrebbe essere fornita in 4 versioni:
  - 36x36 pixel per display con densità *low*
  - 48x48 pixel per display con densità *medium*
  - 72x72 pixel per display con densità *high*
  - 96x96 pixel per display con densità *extra high*
- Tutte le immagini in 4 versioni
  - Da Android 4.3, directory mipmap
    - MIP, Multum In Parvo (molto in poco)

70

## Unità di misura

- dp, density-independent pixels
- sp, scale-independent pixels
  - scalato in base alle preferenze dell'utente sulla grandezza del font (indipendente dalla densità)
    - quindi utile per testi
- mm, millimetri
- in, inches
- pt, points (1/72 di inch)
- px, real pixels

03-Density

71

## FrameLayout

```
<FrameLayout
  android:layout_width="200dp"
  android:layout_height="200dp">

  <ImageView
    android:src="@drawable/dolphin256"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>

  <TextView
    android:textColor="@color/red"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:layout_gravity="center"
        android:text="Delfino" />
</FrameLayout>
```




72

## Layouts – Linear Layout

Slide 73

- Posiziona gli elementi uno dopo l'altro (linearmente)
- Orientazione:
  - android:orientation="vertical"
  - android:orientation="horizontal"



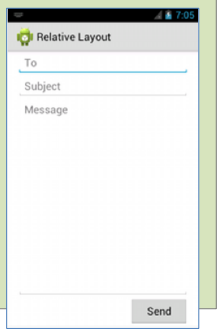
- In ogni figlio: android:layout_weight
  - peso che determina quanto spazio il singolo elemento prende nel Linear Layout

73

## Layouts – Linear Layout

Slide 74

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/to" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/subject" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/message" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/send" />
</LinearLayout>
```




74

## Layouts – Linear Layout

Slide 75

- LL in cui i figli si dividono equamente lo spazio:
  - verticalmente:
    android:layout_height="0dp"
    android:layout_weight="1"
  - orizzontalmente:
    android:layout_width="0dp"
    android:layout_weight="1"
- Non viene lasciato spazio vuoto
  - Se lo si vuole si devono inserire degli elementi fittizi
  - Es. Frame vuoti

75

## Layouts – Relative Layout

Slide 76

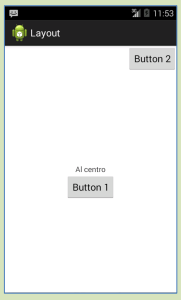
- La posizione degli elementi è "relativa"
  - al layout padre
  - agli altri elementi del layout
- Esempi
  - android:layout_alignParentTop="true"
  - android:layout_centerVertical="true"
  - android:layout_below="@id/other_object"
  - android:layout_toRightOf="@id/other_object"

76

## Layouts – Relative Layout

Slide 77

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="Al centro" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/textView1"
        android:layout_centerHorizontal="true"
        android:text="Button 1" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:text="Button 2" />
</RelativeLayout>
```




77

## Layout - ConstraintLayout

Slide 78

- simile al RelativeLayout
  - permette di specificare la posizione attraverso "vincoli"
  - utile per evitare una gerarchia di layouts innestati troppo profonda
    - che necessita di più tempo per essere disegnata
- Usata dall'editor grafico
- Si inseriscono dei "vincoli" che legano la posizione del nuovo oggetto rispetto a quelli esistenti

78

## Layouts – TableLayout

- Organizza gli elementi in righe e colonne

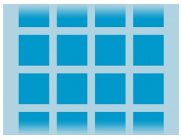
```
<TableLayout
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:stretchColumns="*"
   android:shrinkColumns="*">
   <!-- Riga 1 -->
   <TableRow>
      <TextView android:text="R1C1 " android:padding="8dp"/>
      <TextView android:text="R1C2 " android:padding="8dp"/>
      <TextView android:text="R1C3 " android:padding="8dp"/>
   </TableRow>
   <!-- Riga 2 -->
  <TableRow>
      <TextView android:text="R2C1 " android:padding="8dp"/>
      <TextView android:text="R2C2 " android:padding="8dp"/>
      <TextView android:text="R2C3 " android:padding="8dp"/>
  </TableRow>
</TableLayout>
```

stretch (allarga) e shrink (riduce) permettono di adattare la dimensione delle celle allo spazio disponibile

79

## Layouts – GridView

- Visualizza un insieme di elementi
  - numero totale variabile
  - visibile solo una parte
  - scroll



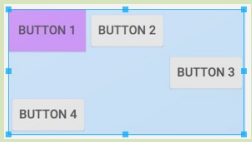
- Adapter
  - fornisce gli elementi da inserire nel GridView
    - lo vedremo in seguito

80

## Layouts – Grid Layout

```
<GridLayout
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:background="#abcdef"
   android:columnCount="3"
   android:rowCount="3" >

   <Button
      android:background="#aa55ee"
      android:id="@+id/button1"
      android:text="Button 1" />

   <Button
      android:id="@+id/button2"
      android:text="Button 2" />

   <Button
      android:id="@+id/button3"
      android:layout_column="2"
      android:layout_row="1"
      android:text="Button 3" />

   <Button
      android:id="@+id/button4"
      android:text="Button 4" />

</GridLayout>
```

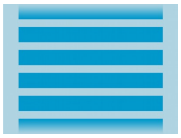
columnCount e rowCount previsti solo per Grid non per Table




81

## Layouts – List View

- Visualizza un insieme di elementi organizzati in una lista
  - numero totale variabile
  - visibile solo una parte
  - scroll

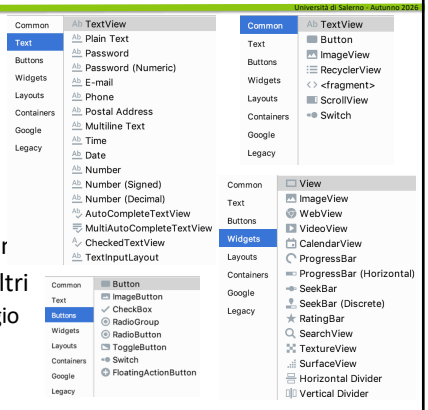


- Adapter
  - fornisce gli elementi da inserire nel List View
    - lo vedremo in seguito

82

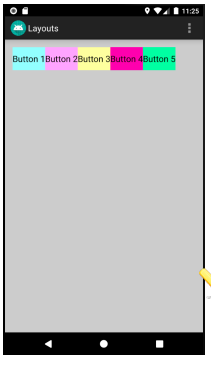
## Layouts - widget

- TextView
- Button
- TextEdit
- ImageView
- CheckBox
- RadioButtor
- … e molti altri
  - es. Orologio




83

## App Layouts




### 04-Layouts

- Cose che vediamo
  - menu
  - linear layout
    - grandezze, pesi
    - frame
  - grid layout
  - relative layout
  - margini e padding

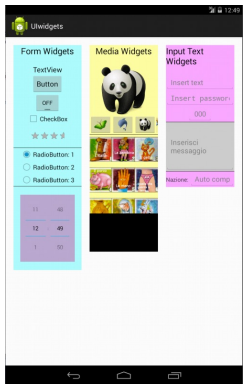
Usare un "Theme" che prevede l'action bar! Quello di default (Theme.Material3.DayNight.NoActionBar) non la prevede. Usare, ad esempio, Theme.AppCompat.DayNight.DarkActionBar

84

## Widgets








05-Widgets

- Textview
- Pulsanti
  - normali, toggle, radio
- Image view
- Input testo
  - vari tipi
- Widget avanzati
  - time picker, media player

85

## ANDROID Mobile Programming



**La nostra prima (vera) app**



86

## Prima app





06-Calcolatrice

- Una sola "Activity"
- Pulsanti
  - Listeners
- TextView




87

## Esercizio

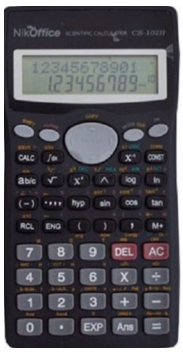




- Calcolatrice più realistica







88

## ANDROID Mobile Programming



**Android Studio Debugger**



89

## Copiare (rinominare)





- Fare una copia della directory
  - Rinominarla con *NuovoNome*
- Aprire NuovoNome in Android Studio
- Rinominare package e directories
  - Right-click sul package, Refactor>Rename
    - Rinomina directory
- Nel file build.gradle
  - Rinominare **ApplicationId** (vecchionome->nuovonome)
- Nel manifesto
  - Rinominare **package** (vecchionome->nuovonome)
- Nel file strings.xml
  - **<string name="app_name">NuovoNome</string>**
- Clean, Rebuild

90

## Android Studio Debugger

Slide 91



- Permette
  - Eseguire un'app in modalità "debug"
    - Sia con l'emulatore che con una device reale
  - Inserire dei "breakpoints"
    - Esaminare il valore delle variabili
  - Esecuzione "passo-passo"
- LLDB
  - Se c'è codice C/C++ viene usato anche il debugger LLDB
  - Useremo solo il debugger Java

91

## Android Studio Debugger

Slide 92



- Build variant
  - Deve essere "debuggable"
    - Build>Select Build Variant
- Per lanciare l'app in modalità debug
  - Run>Debug (CTRL-ALT-D)
  - icona
- Selezionare la device

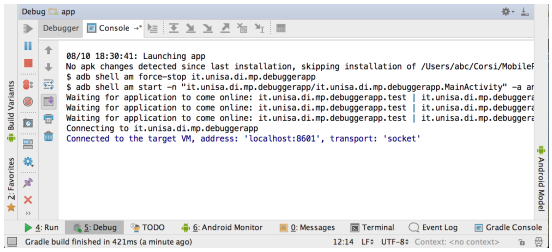
92

## Android Studio Debugger

Slide 93



- Si apre la finestra di debug
  - CMD-5




93

## Breakpoints

Slide 94



- Per inserire/eliminare breakpoints
  - Click a destra del numero di riga

```
25      public void somma(View v) {
26          int num1 = 0, num2 = 0;
27          String str1 = numero1.getText().toString();
28          if (str1.length() > 0) num1 = Integer.parseInt(str1);
29          String str2 = numero1.getText().toString();
30          if (str2.length() > 0) num2 = Integer.parseInt(str2);
31          int somma = num1 + num2;
32          risultato.setText(somma + "");
33      }
```

- L'esecuzione si fermerà ad ogni breakpoint
  - Possiamo controllare lo stato della memoria
  - Continuare l'esecuzione passo-passo, etc.

94

## Comandi debugger

Slide 95



- Valuta una espressone
- Esecuzione di una singola istruzione
- Entra all'interno di una funzione
- Esci dalla funzione
- Riprendi l'esecuzione fino al prossimo breakpoint

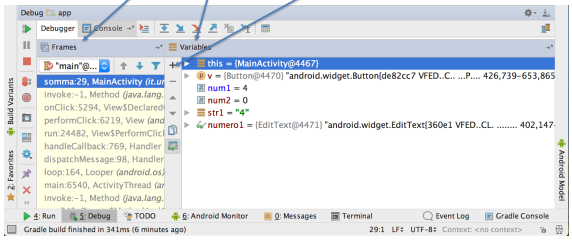
95

## Debugger

Slide 96



- Ispezione
  - Frames, stack delle chiamate
  - Variablili, valori memoria
  - Wacthes +




96

## Messaggi di log: Log.X



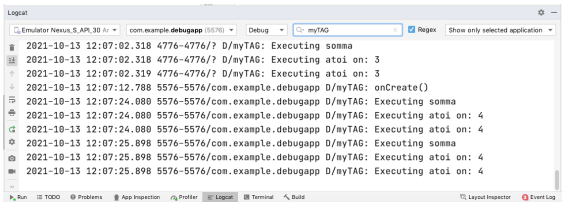
- private static final String TAG = "MyActivity";
- Log.**x**(TAG, "messaggio");
- **x** può essere
  - d: debug
  - e: errore
  - i: info
  - w: warning
  - v: verbose

97

## Logcat



- Logcat
  - Messaggi di log

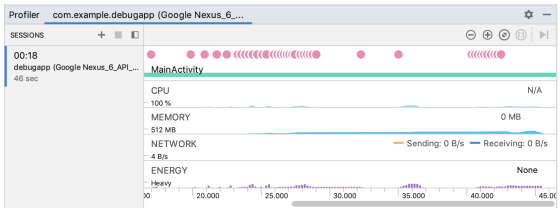



98

## Profiler



- Profiler
  - Uso memoria, CPU, rete e GPU

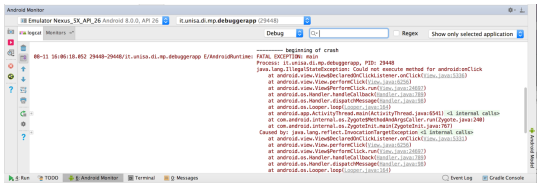



99

## Logcat



- Crash: stack trace
  - Fa vedere
    - la linea di codice che ha causato l'errore
    - lo stack delle chiamate



100

## DebugApp



07-Debug

- App con un errore
  - usiamo il debugger per trovarlo
- Memory leak
  - usiamo il profiler per vederlo

101

## ANDROID Mobile Programming



# ListView

102

## ListView



- Widget specifico per le liste
  - divide l'area disponibile in varie posizione
    - il numero dipende dall'area disponibile e dalla grandezza di ogni elemento
- Gli elementi vengono memorizzati in un array
  - solitamente sono di più rispetto alle posizioni disponibili nel widget
  - Si può "scorrere" la lista
- Adapter
  - fornisce gli elementi da visualizzare in base allo scorrimento effettuato dall'utente

103

## ListView



- Lista semplice

```
<ListView
   android:id="@+id/mylistview"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
</ListView>
```

- File xml per il singolo elemento della lista

```
<?xml version="1.0" encoding="utf-8" ?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent" android:layout_height="match_parent">
  <TextView
    android:id="@+id/textViewList"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="" android:padding="10dp"
    android:textSize="22dp"/>
</RelativeLayout>
```

104

## ListView



- Definire l'array con gli elementi

```
String [] array = {"Pasquale","Maria","Michele","Antonella", "Vincenzo",
    "Teresa", "Roberto", "Rossella", "Antonio", "Luca", "Liliana", "Stefania",
    "Francesca", "Andrea", "Marco", "Elisa", "Anna", "Lorenzo"};
```

- Definire un adapter

```
ArrayAdapter<String> arrayAdapter =
        new ArrayAdapter<String>(context, R.layout.list_element,
                                 R.id.textViewList, array);
```

- Individuare il widget listview

```
listView = findViewById(R.id.mylistview);
```

- Associare l'adapter al widget

```
listView.setAdapter(arrayAdapter);
```

105

## ListView



- Definire un listener per i click sugli elementi

```
listView.setOnItemClickListener(new OnItemClickListener() {

  @Override
  public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
    String  str  = listView.getItemAtPosition(position).toString();
    // Fai qualcosa con l'elemento
    …
    …
  }
});
```

106

## ListView



- ListView semplice
  - Ogni elemento è una stringa
- ListView personalizzato
  - Ogni elemento ha un proprio layout con dei sottoelementi
    - es. nome, cognome, telefono, foto
  - Il click è però su tutto l'elemento
- ListView personalizzato con click multiplo
  - si possono cliccare i singoli elementi

107

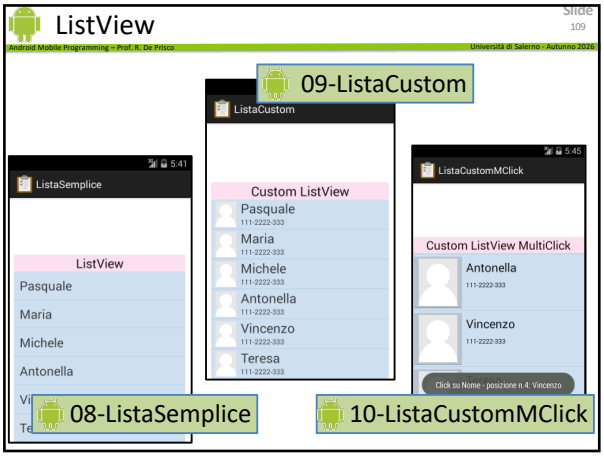
## ListView



- Per personalizzare gli elementi
  - il file di layout!
  - CustomAdapter
- Per il click multiplo
  - non possiamo più usare il listener del listview
    - funziona per tutto l'elemento
  - listeners ad-hoc per ogni sottoelemento
    - problema: non sappiamo più in quale posizione dell'array siamo!!!
    - trucchetto per risolverlo: setTag, getTag

108

## ListView




109

## ANDROID Mobile Programming

# Ciclo di vita

110

## Ciclo di vita delle attività

- Ogni "Activity" ha un ciclo di vita

**Attività non esiste**

1. onCreate()
2. onStart()
3. onResume()

**Attività in esecuzione**

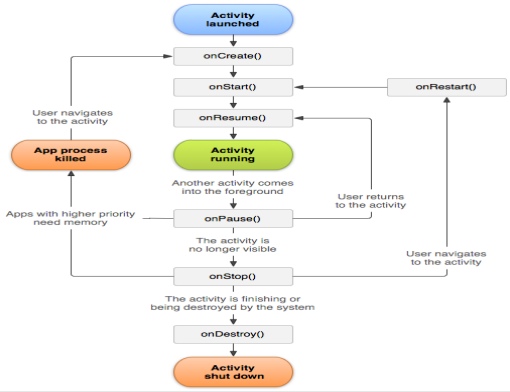
4. onPause()
5. onStop()
6. onDestroy()

**Attività non esiste**

- Eseguiti secondo un determinato schema

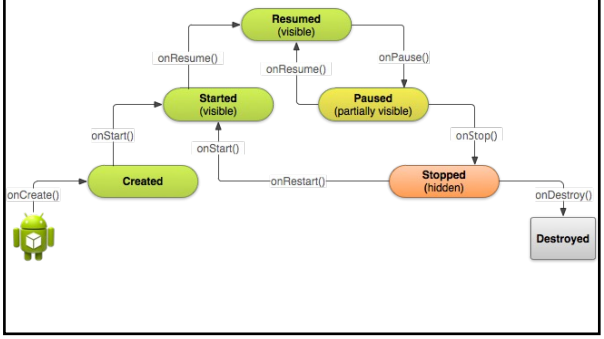
111

## Ciclo di vita delle attività




112

## Ciclo di vita delle attività




113

## Ciclo di vita delle attività

- Quando l'utente preme il pulsante "Home"
  - vengono chiamate
    - onPause()
    - onStop()
- Quando si ritorna all'attività
  - vengono chiamate
    - onRestart()
    - onStart()
    - onResume()

114

## Ciclo di vita delle attività



- Quando l'utente ruota il dispositivo
  - l'attività viene prima eliminata:
    - onPause()
    - onStop()
    - onDestroy()
  - e poi ricreata:
    - onCreate()
    - onStart()
    - onResume()
- onDestroy(): perdita dello stato!!!!

115

## onSaveInstanceState



- Si salva lo stato in onSaveInstanceState()

```
@Override
  public void onSaveInstanceState(Bundle savedInstanceState) {
      // Salvare lo stato dell'app
      savedInstanceState.putStringArrayList("LISTA_STRINGHE", array_di_stringhe);
      savedInstanceState.putInt("CONTATORE", counter);
      // Always call the superclass so it can save the view hierarchy state
      super.onSaveInstanceState(savedInstanceState);
  }
```

- Lo si recupera in onCreate()

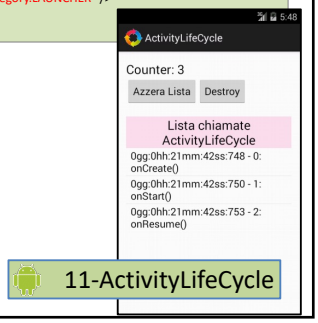
```
@Override
  protected void onCreate(Bundle savedInstanceState) {
      ...
      if (savedInstanceState != null) {
          array_di_stringhe = savedInstanceState.getStringArrayList("LISTA_STRINGHE");
          counter = savedInstanceState.getInt("CONTATORE");
      }
  }
```

116

## Main activity



```
<activity android:name=".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>
```




11-ActivityLifeCycle

117

## Ciclo di vita delle attività



- App Calcolatrice
  - Cosa succede se ruotiamo il dispositivo?

Correggere l'errore nell'app Calcolatrice

118

## ANDROID Mobile Programming



# Ciclo di vita e Cambi di configurazione



119

## Configurazione device



- Screen orientation (portrait, landscape)
- Layout direction: da sinistra a destra, da destra a sinistra
- Available width, height
- Screen size (small, normal, large, xlarge)
- Round screen (e.g. orologio)
- UI mode (car, desk, television, appliance,watch, vrheadset)
- keyboard availability (keysexposed, keyshidden, keyssoft)
- …. altre: configuration qualifier names (Table 2)

https://developer.android.com/guide/topics/resources/providing-resources

120

## Cambio di configurazione

Slide 121

- Il sistema operativo distrugge e ricrea l'attività in esecuzione
- Motivazione: permettere all'app di adattarsi al meglio alla nuova configurazione
- Per salvare lo stato
  - onSaveInstanceState()
  - ViewModel class
    - oggetti persistenti: dati che esistono nella vecchia activity che viene distrutta e rimangono a disposizione nella nuova istanza che viene ri-creata

121

## onConfigurationChanged()

Slide 122

- è possibile gestire in proprio il cambiamento
- Nel manifesto:

```
<activity android:name=".MainActivity"
   android:configChanges="orientation">
```

- Effetto:
  - l'activity NON viene distrutta; viene eseguito il metodo onConfigurationChanged()

122

## onConfigurationChanged()

Slide 123

```
@Override
public void onConfigurationChanged(Configuration newConfig) {
   super.onConfigurationChanged(newConfig);

   // Checks the orientation of the screen
   if (newConfig.orientation == Configuration.ORIENTATION_LANDSCAPE) {
      Toast.makeText(this, "landscape", Toast.LENGTH_SHORT).show();
   } else if (newConfig.orientation == Configuration.ORIENTATION_PORTRAIT){
      Toast.makeText(this, "portrait", Toast.LENGTH_SHORT).show();
   }
}
```

- Valori dell'oggetto Configuration
  - interi specificati nella classe Configuration

<https://developer.android.com/reference/android/content/res/Configuration

123

## Main activity

Slide 124

```
<activity android:name=".MainActivity" android:configChanges="orientation">
   <intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.LAUNCHER" />
   </intent-filter>
</activity>
```

- L'app non viene distrutta
  - non consigliato: *"It is not recommended to handle configuration changes yourself due to the hidden complexity of handling the configuration changes. However, if you are unable to preserve your UI state using the preferred options (onSaveInstanceState(), ViewModels, and persistent storage) you can instead prevent the system from restarting your activity during certain configuration changes. Your app will receive a callback when the configurations do change so that you can manually update your activity as necessary"*.

https://developer.android.com/guide/topics/resources/runtime-changes

11-OnConfigurationChanged




124

# ANDROID Mobile Programming

## Backstack

125

125

## Backstack

Slide 126

- Un'app normalmente è fatta di più activity
  - ogni activity ha uno specifico compito
    - modularità
- Es. un'app per la posta elettronica
  - un'attività per la scrittura del messaggio
  - un'attività per spedire il messaggio
  - un'attività per vedere una lista dei messaggi
  - un'attività per vedere il contenuto di un messaggio
  - ecc.

126

## Backstack

- Un'attività può lanciare un'altra attività
  - anche attività che appartengono ad altre app
- Class "Intent"
  - serve a lanciare una nuova attività e "passare" i dati all'attività che si lancia
  - la vedremo fra poco
- Task
  - è un insieme di attività con cui l'utente interagisce

127

## Backstack

- Più attività possono coesistere, vengono organizzate in un **backstack**
- Solitamente un task parte dall'*Home screen*
  - l'utente clicca un'icona e lancia un'attività
  - l'applicazione viene mostrata sulla schermo
    - gergo tecnico: viene portata in "*foreground*"
- Se vengono lanciate nuove attività
  - l'attività corrente viene messa nel backstack
  - l'utente ci può tornare con il pulsante Back

128

## Backstack

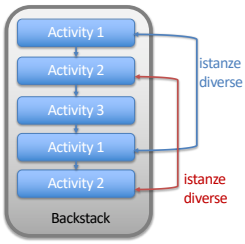



- Continuando a premere Back si ritorna all'Home screen

129

## Istanze multiple

- Se un'attività può essere lanciata da più di un'altra attività si possono avere istanze multiple



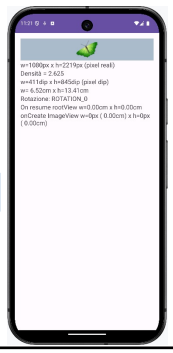

12-MultiActivity

http://developer.android.com/guide/components/tasks-and-back-stack.html

130

## Display

- Adattarsi alla caratteristiche del display



13-Display

131

## ANDROID Mobile Programming

# Intent

132

## Classe Intent



- Intent
  - è una descrizione (astratta) di un'operazione da svolgere
- Permette di
  - startActivity: lanciare una nuova attività
  - broadcastIntent: spedire l'intent in broadcast
    - verrà ricevuto dai BroadcastReceiver interessati
  - startService o bindService: comunicare con un servizio di background

133

## Intent



- Parti principale di un oggetto Intent
  - Action: l'azione da svolgere
    - es. ACTION_VIEW, ACTION_EDIT, ACTION_MAIN
  - Data: i dati su cui operare espressi come URI
    - Uniform Resourse Identifier: <schema>:<parte specifica>
      - "http://www.di.unisa.it/"
      - "mailto:robdep@unisa.it"
      - "geo:0,0?via+Posidonia+Salerno+Italy"
      - "tel:+391112223456"
      - "content://com.android.contacts/contacts"
- Esempi di coppie (azione,dati):
  - ACTION_VIEW, content://contacts/people/1
  - ACTION_DIAL, content://contacts/people/1
  - ACTION_DIAL, tel:1112233

134

## Intent



- Altre parti di un intent
  - Category
    - informazioni aggiuntive sull'azione da eseguire
      - es. CATEGORY_BROWSABLE significa che si può usare un browser come Component
  - Type
    - specifica in modo esplicito il tipo (MIME) dei dati. Normalmente il tipo viene dedotto automaticamente
  - Component
    - Specifica in modo esplicito l'attività da eseguire (che altrimenti verrebbe dedotta dalle altre informazioni)
  - Extras
    - un bundle di informazioni addizionali (dati specifici per l'attività).

135

## Intent



- Risoluzione esplicita
  - specifichiamo in modo esplicito l'attività (Component) che vogliamo lanciare
- Risoluzione implicita
  - Component non è specificata
  - Android sceglie un'attività appropriata, in base a
    - Action
    - Type
    - URI
    - Category
  - Le attività dichiarano le action che possono soddisfare nel manifesto

136

## Intent



```
Intent i;
i = new Intent(Intent.ACTION_PICK,ContactsContract.Contacts.CONTENT_URI);
startActivityForResult(i, REQUEST_CODE);
```

- Azione: ACTION_PICK
  - Chiede di selezionare un item
- Data:
  - *ContactsContract.Contacts.CONTENT_URI*
  - *"content://com.android.contacts/contacts"*
- startActivityForResult
  - lancia l'attività chiedendo un risultato
  - REQUEST_CODE serve ad identificare la richiesta

137

## Intent



```
@Override
protected void onActivityResult(int request, int result, Intent data) {
    if (request == REQUEST_CODE  && result == Activity.RESULT_OK) {
        …
```

- onActivityResult
  - viene chiamato quando si ritorna all'attività di partenza
  - permette di controllare il risultato restituito
    - controlliamo il REQUEST_CODE
    - in questo caso anche un flag di OK
    - gestiamo i dati restituiti

138

## Intent Extras



- Informazioni "extra"
  - coppie chiave-valore
    - putExtra
    - getExtra

```
intent.putExtra("CONTATORE",c);
intent.putExtra("STRINGA","Ciao");
intent.putExtras(bundle);  //Inserisce tutti i dati del Bundle bundle
```

```
c = intent.getExtra("CONTATORE");
stringa = intent.getExtra("STRINGA");
Bundle b = intent.getExtras();
```



## Intent Flags



- Informazione su come l'intent dovrebbe essere trattato
  - Esempi:
    - FLAG_ACTIVITY_NO_HISTORY
      - non memorizzare l'attività nello stack delle attività
    - FLAG_DEBUG_LOG_RESOLUTION
      - stampa informazioni addizionali quando l'intent viene eseguito
      - molto utile in fase di debug se l'intent che vogliamo far eseguire non viene eseguito



## Intent Component



- Permette di specificare l'attività "target"
  - da usare quando c'è una sola specifica attività (componente) che deve ricevere l'intent

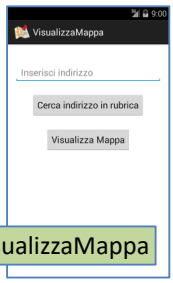
```
Intent intent = new Intent(Context context, Class<?> class);

//oppure
intent.setComponent(…);
intent.setClass(…);
intent.setClassName(…);
```



## VisualizzaMappa



- Un'app che usa attività di altre app
  - permette di inserire un indirizzo
- Indirizzo dalla rubrica
  - sfrutta un'attività della rubrica
  - **permesso per leggere la rubrica!!!**
- Visualizza la mappa
  - sfrutta un'attività dell'app GoogleMaps




14-VisualizzaMappa



## Intent



15-Quiz






## Quiz



- Cosa succede se ruotiamo lo schermo?
  - Un utente può barare sfruttando questo fatto

Correggere l'errore dovuto alle rotazioni nell'app Quiz

- L'app Quiz contiene (volutamente) alcuni errori e omissioni

Individuare e correggere gli errori e colmare le omissioni

ANDROID Mobile Programming

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno – Autunno 2026

# Permessi

145

145

## Permessi



Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2026

- Meccanismo di protezione
  - risorse e dati
  - per accedere c'è bisogno di un esplicito consenso da parte dell'utente
- Servono a limitare l'accesso a
  - informazioni dell'utente (e.g. i contatti della rubrica)
  - servizi con costi (e.g., invio SMS, chiamate tel., accesso a Internet)
  - Risorse di sistema (e.g., fotocamera, GPS)

146

## Permessi



Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2026

- Vengono rappresentati da stringhe
  - android.permission.CAMERA
  - android.permission.SEND_SMS
- Ogni app deve dichiarare nel manifesto i "permessi" che intende chiedere

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

147

## Permessi



Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2026

- I permessi sono divisi in due classi
  - Normali e "pericolosi"
- I permessi normali vengono ***concessi senza chiedere nulla all'utente***
- I permessi "pericolosi" devono essere approvati dall'utente
  - quando si istalla l'app (API < 23)
  - a runtime (API ≥ 23)

148

## Permessi



Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2026

- API < 23: Quando si istalla l'app
  - bisogna accettare i permessi
- API ≥23:Quando lanciamo l'app
  - alla seconda richiesta avremo la possibilità di rendere definitiva la risposta
- API ≥30
  - Possibilità di concedere il permesso «solo questa volta»



149

## Gestione permessi



Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2026

- Si può cambiare la decisione
- Menù impostazioni
  - App istallate
    - Autorizzazioni
- Quindi il programmatore non deve mai assumere che il permesso ci sia
  - nel codice occorre controllare se c'è
  - ContextCompat.checkSelfPermission(...)

150

## Permessi

```
if (ContextCompat.checkSelfPermission(
        CONTEXT, Manifest.permission.REQUESTED_PERMISSION) ==
        PackageManager.PERMISSION_GRANTED) {
    // You can use the API that requires the permission.
    performAction(...);
} else if (shouldShowRequestPermissionRationale(...)) {
    // In an educational UI, explain to the user why your app requires this
    // permission for a specific feature to behave as expected. In this UI,
    // include a "cancel" or "no thanks" button that allows the user to
    // continue using your app without granting the permission.
    showInContextUI(...);
} else {
    // You can directly ask for the permission.
    // The registered ActivityResultCallback gets the result of this request.
    requestPermissionLauncher.launch(
            Manifest.permission.REQUESTED_PERMISSION);
}
```

151

## Permessi normali (API 23, 6.0)

- ACCESS_LOCATION_EXTRA_COMMANDS
- ACCESS_NETWORK_STATE
- ACCESS_NOTIFICATION_POLICY
- ACCESS_WIFI_STATE
- BLUETOOTH
- BLUETOOTH_ADMIN
- BROADCAST_STICKY
- CHANGE_NETWORK_STATE
- CHANGE_WIFI_MULTICAST_STATE
- CHANGE_WIFI_STATE
- DISABLE_KEYGUARD
- EXPAND_STATUS_BAR
- FLASHLIGHT
- GET_PACKAGE_SIZE
- INTERNET
- KILL_BACKGROUND_PROCESSES
- MODIFY_AUDIO_SETTINGS
- NFC
- READ_SYNC_SETTINGS
- READ_SYNC_STATS
- RECEIVE_BOOT_COMPLETED
- REORDER_TASKS
- REQUEST_INSTALL_PACKAGES
- SET_TIME_ZONE
- SET_WALLPAPER
- SET_WALLPAPER_HINTS
- TRANSMIT_IR
- USE_FINGERPRINT
- VIBRATE
- WAKE_LOCK
- WRITE_SYNC_SETTINGS
- SET_ALARM
- INSTALL_SHORTCUT
- UNINSTALL_SHORTCUT

152

## Permessi pericolosi (API 23, 6.0)

| Gruppo | Permesso |
|---|---|
| CALENDAR | READ_CALENDAR |
| | WRITE_CALENDAR |
| CAMERA | CAMERA |
| CONTACTS | READ_CONTACTS |
| | WRITE_CONTACTS |
| | GET_ACCOUNTS |
| LOCATION | ACCESS_FINE_LOCATION |
| | ACCESS_COARSE_LOCATION |
| MICROPHONE | RECORD_AUDIO |
| PHONE | READ_PHONE_STATE |
| | CALL_PHONE |
| | READ_CALL_LOG |
| | WRITE_CALL_LOG |
| | ADD_VOICEMAIL |
| | USE_SIP |
| | PROCESS_OUTGOING_CALLS |

153

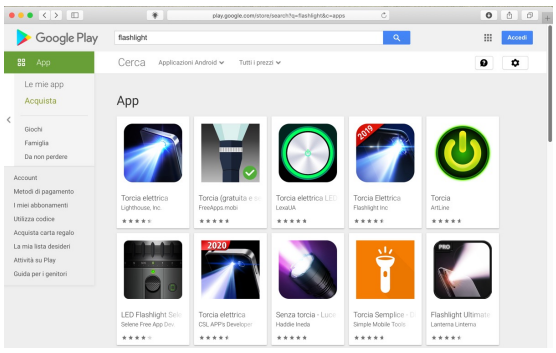
## Permessi pericolosi (API 23, 6.0)

| Gruppo | Permesso |
|---|---|
| SENSORS | BODY_SENSORS |
| SMS | SEND_SMS |
| | RECEIVE_SMS |
| | READ_SMS |
| | RECEIVE_WAP_PUSH |
| | RECEIVE_MMS |
| STORAGE | READ_EXTERNAL_STORAGE |
| | WRITE_EXTERNAL_STORAGE |

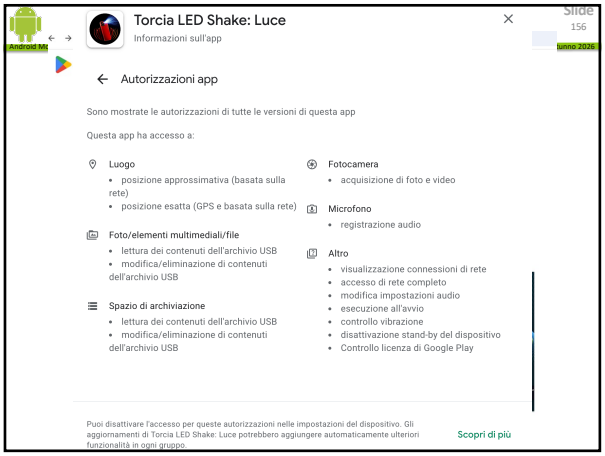
- Quando l'app richiede un permesso pericoloso
  - se ha già un permesso per lo stesso gruppo viene concesso automaticamente
  - altrimenti viene richiesto all'utente (dialog box) il permesso per il GRUPPO

154

## Permessi app "torcia"



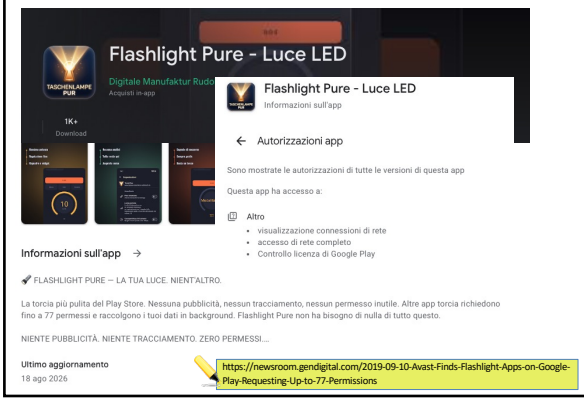

155




156

## Flashlight Pure






157

## ANDROID Mobile Programming



# Threads

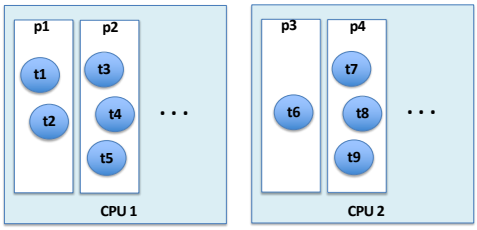


158

## Threads



- Computazione parallela all'interno di un processo
  - Ogni thread ha il proprio program counter ed il proprio stack (chiamate di funzioni)
  - condivide con gli altri thread del processo l'heap e la memoria statica (malloc)




159

## Java Threads



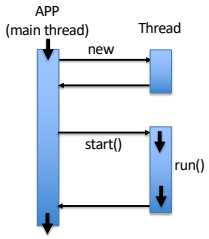
- Oggetti java.lang.thread
- Implementano l'interfaccia runnable
  - devono avere il metodo void run()
- Metodi che useremo
  - void start()
  - void sleep(long time)
  - void wait()
    - aspetta che un altro oggetto chiami notify() su questo oggetto
  - void notify()

160

## Threads



- Per usare un thread:
  - Creare un oggetto Thread
  - Chiamare il metodo start() del thread
    - che chiamerà il metodo run()




161

## Threads

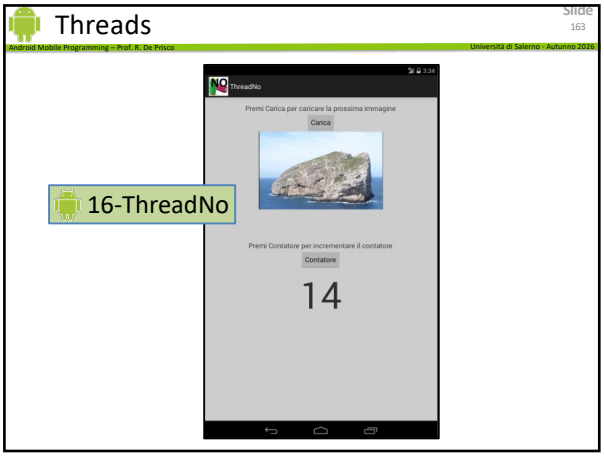


- Android non permette ai thread in background di interagire con l'interfaccia utente
- Solo il main thread può farlo
  - Es.: Non possiamo aggiornare l'immagine nel thread creato per caricare l'immagine
- Metodi
  - boolean View.post(Runnable action)
  - void Activity.runOnUiThread(Runnable action)

162

## Threads

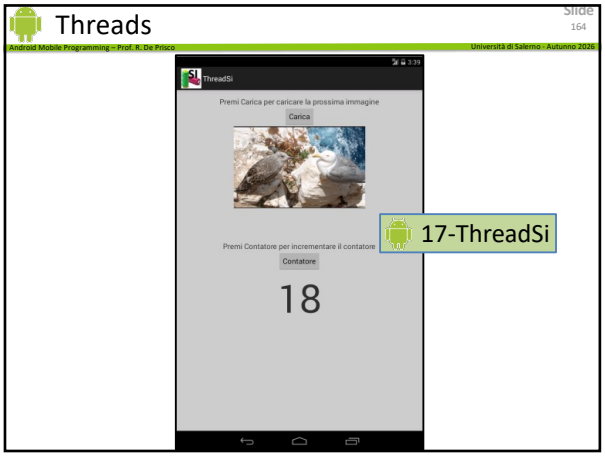





163

## Threads






164

## Async task



- Facilitano l'interazione fra background thread e main thread
- Background thread
  - esegue il task
  - notifica sullo stato di avanzamento
- Main (UI) thread
  - setup iniziale
  - display dello stato di avanzamento
  - usa i risultati (es. mostrandoli sul display)

165

## AsyncTask



- Classe Java generica

  class AsyncTask<Params, Progress, Result> {
  …
  }

- Parametri
  - Params: tipo (di dati) per il lavoro che deve svolgere il background thread
  - Progress: tipo (di dati) usato per lo stato di avanzamento
  - Result: tipo (di dati) per il risultato del task

166

## AsyncTask.execute()

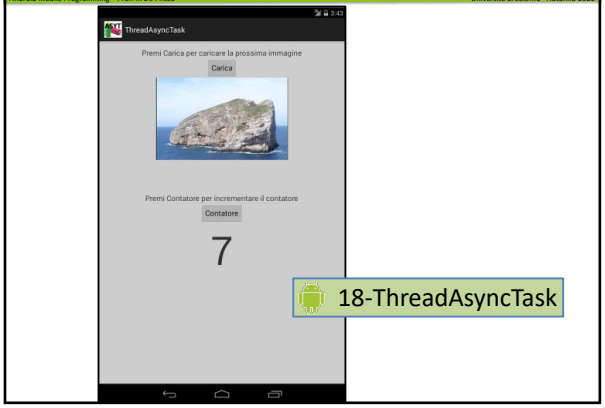


- void onPreExecute()
  - eseguito nel main thread prima di doInBackground()
- Result doInBackground(Params… params)
  - viene eseguito
    - lista variabile di parametri
    - restituisce un oggetto di tipo Result
  - può chiamare
    - void publishProgress(Progress… values)
- void onProgressUpdate(Progress… values)
  - nel main thread in risposta a publishProgress
- void onPostExecute(Result result)
  - nel main thread DOPO doInBackground() con il risultato di doInBackground come parametro

167

## AsyncTask






168

ANDROID Mobile Programming

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno – Autunno 2026

# Fragments

169

169

## Fragments

Slide 170

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2026

- Frammento
  - rappresenta una "porzione" dell'UI
- Un'activity può "ospitare" vari frammenti
  - I frammenti possono essere inseriti e rimossi durante l'esecuzione
- Si possono creare UI con molti frammenti
  - anche in funzione della grandezza dello schermo

170

## Fragments

Slide 171

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2026

- Un frammento è sempre "ospitato" da un'activity
- Un frammento è una sorta di sub-activity
  - ha il suo ciclo di vita
  - che è strettamente legato a quello dell'activity
    - es. se l'activity è in pausa (stato "paused" del ciclo di vita) lo sono anche tutti i suoi frammenti
    - se l'activity è in esecuzione (stato "resumed") allora i frammenti possono essere gestiti

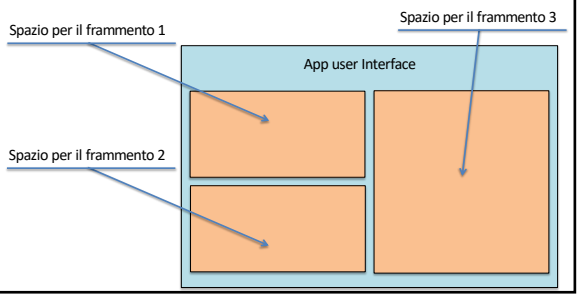
171

## Fragments

Slide 172

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2026

- La porzione di UI occupata dal frammento deve essere specificata nel layout
  - può essere definita dinamicamente




172

## Fragments

Slide 173

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2026

- Filosofia di progettazione
  - Interfacce utente dinamiche
  - in particolare per adattarsi sia a schermi grandi che a schermi piccoli
- Esempio tipico
  - App che gestisce un elenco di elementi
    - es. titoli di articoli di un giornale
  - Ogni elemento può essere cliccato per essere esaminato
    - es. visualizzazione dell'articolo

173

## Fragments

Slide 174

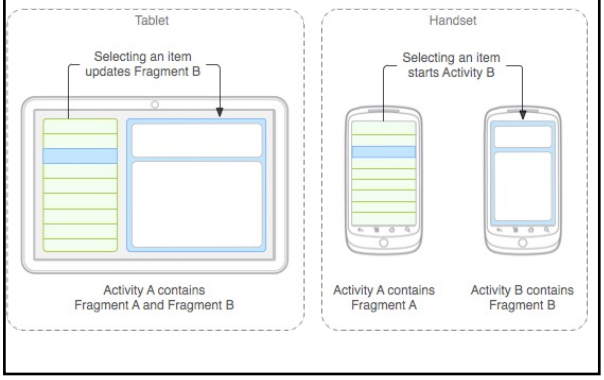
Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2026

- Si può usare
  - un frammento per l'elenco
  - un frammento per la visualizzazione
- Se lo schermo è piccolo
  - sarà visibile solo uno dei frammenti
    - cliccando un titolo si passerà dal frammento titoli al frammento visualizzazione
- Se lo schermo è grande
  - saranno visualizzati entrambi i frammenti

174

## Fragments



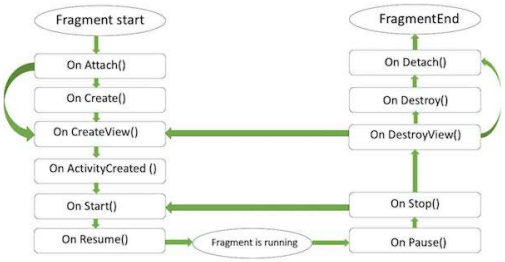



175

## Creare frammenti



- Istanziare un oggetto Fragment
  - la classe Fragment è simile alla classe Activity
  - proprio ciclo di vita




176

## Creare frammenti



- Normalmente dovremo implementare almeno
  - onCreate()
    - Inizializzazione come in un activity
    - NON definiamo il layout
  - onCreateView()
    - definiamo il layout. Il metodo deve restituire una View
    - facciamo l'inflate di un file di layout
  - onPause()
    - il primo metodo chiamato quando il frammento viene eliminato (si dovrebbero rendere permanenti eventuali cambiamenti altrimenti si perdono)

177

## Creare frammenti



```
public static class ExampleFragment extends Fragment {
  @Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container,
                           Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View v =inflater.inflate(R.layout.example_fragment, container, false);
    return  v;
  }
}
```

- è l'equivalente di setContentView nella activity host
  - container è un oggetto ViewGroup che serve a specificare i parametri di layout

178

## Fragments



public View **inflate** (int resource, ViewGroup root, boolean attachToRoot) Added in API level 1

Inflate a new view hierarchy from the specified xml resource. Throws `InflateException` if there is an error.

**Parameters**

| | |
|---|---|
| *resource* | ID for an XML layout resource to load (e.g., `R.layout.main_page`) |
| *root* | Optional view to be the parent of the generated hierarchy (if *attachToRoot* is true), or else simply an object that provides a set of LayoutParams values for root of the returned hierarchy (if *attachToRoot* is false.) |
| *attachToRoot* | Whether the inflated hierarchy should be attached to the root parameter? If false, root is only used to create the correct subclass of LayoutParams for the root view in the XML. |

**Returns**

The root View of the inflated hierarchy. If root was supplied and attachToRoot is true, this is root; otherwise it is the root of the inflated XML file.

179

## Creare frammenti



- Un frammento può essere inserito staticamente nel layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="horizontal"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <fragment android:name="com.example.news.ArticleListFragment"
      android:id="@+id/list"
      android:layout_weight="1"
      android:layout_width="0dp"
      android:layout_height="match_parent" />
  <fragment android:name="com.example.news.ArticleReaderFragment"
      android:id="@+id/viewer"
      android:layout_weight="2"
      android:layout_width="0dp"
      android:layout_height="match_parent" />
</LinearLayout>
```

180

## Creare frammenti



- Oppure dinamicamente a runtime

```
FragmentManager fm= getFragmentManager();
FragmentTransaction ft = fm.beginTransaction();
ExampleFragment fragment = new ExampleFragment();
ft.add(R.id.fragment_container, fragment);
ft.commit();
```

- R.id.fragment_container
  - è un ViewGroup nel layout dell'activity che individua la porzione dello schermo da dedicare a questo frammento

181

## Gestire i frammenti



- Usiamo il FragmentManager

```
FragmentManager fm = getFragmentManager();
```

- Iniziamo una transazione

```
FragmentTransaction ft = fm.beginTransaction();
```

- Effettuiamo le operazioni
  - inserire un frammento (già vista)
  - rimuovere un frammento
  - sostituire un frammento
- Commit

```
ft.commit();
```

182

## Gestire i frammenti

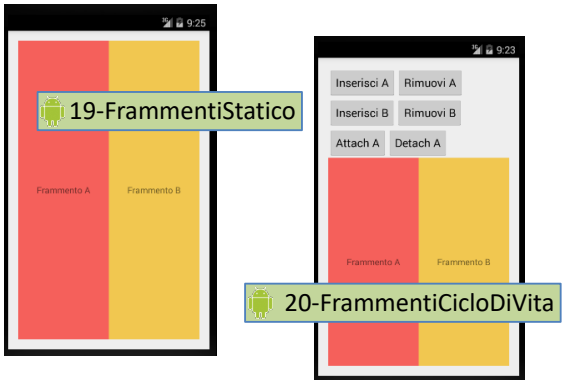


- addToBackStack()
  - per inserire i cambiamenti nel backstack
- Il backstack considera solo le activity
  - dobbiamo gestire manualmente i frammenti
- Se non chiamiamo addToBackStack
  - quando premiamo back "salteremo" i cambiamenti fatti con i frammenti
  - non è quello che l'utente si aspetta

183

## Esempi






184

## Comunicare con l'activity



- Può essere utile comunicare con l'activity
  - Creare dei metodi di callback
- Ad es. il frammento può definire un'interfaccia

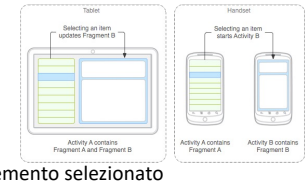
```
public static class MyFragment extends Fragment {
   ...
   // Container Activity must implement this interface
   public interface OnArticleSelectedListener {
      public void onArticleSelected(int index);
   }
   ...
}
```

185

## Comunicare con l'activity



- App
  - lista
  - dettagli elementi

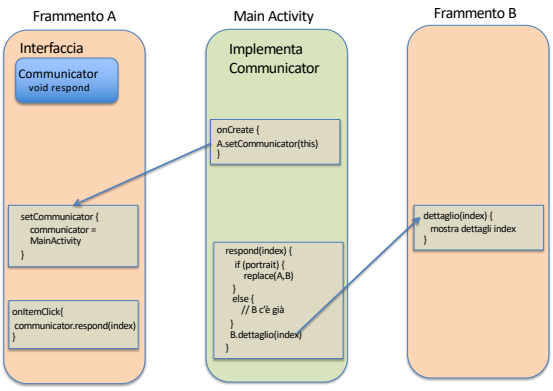



- Frammento lista
  - deve comunicare l'elemento selezionato
- Frammento dettagli
  - deve ricevere l'informazione
- Evitare la comunicazione diretta fra i frammenti
  - diminuisce la riusabilità

186

## Comunicare con l'activity






187

## Frammenti e backstack



- Nuova activity per la modalità portrait
  - facilita la gestione del backstack
  - per le activity è automatica
- Frammenti e backstack
  - i frammenti non vengono inseriti nel backstack
  - quando premiamo il pulsante back
    - si ritorna alla precedente activity
    - saltando eventuali cambiamenti dell'UI dovuti all'uso dei frammenti
  - se si vuole occorre gestire il backstack manualmente

188

## ListFragment

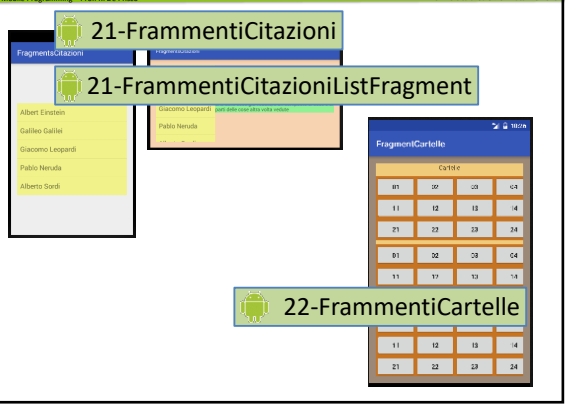


- **Fragment specializzato** che incorpora automaticamente un ListView.
- Evita di dover chiamare findViewById() per il ListView.
- Permette di collegare rapidamente un **adapter** (es. ArrayAdapter, CursorAdapter, SimpleAdapter…).
- Fornisce callback pronti come onListItemClick().

189

## Esempi






190

## Frammenti



- Esercizio (avanzato)
- L'app FrammentiCartelle utilizza un layout predefinito di 12 cartelle
  - quindi può gestire al massimo 12 cartelle
- Scrivere una nuova versione in cui il layout viene costruito dinamicamente
  - creare nuove view (LinearLayout, Frame, etc)
  - LayoutParameters
  - view.add()

191

## ANDROID Mobile Programming



# Networking

192

## Networking



- Comunicazione via rete
- Socket
  - Java.net
- HTTP
  - Classe httpURLConnection
- JSoup
  - HTML parsing

193

## Networking



- Classe InetAddress
  - permette di gestire gli indirizzi IP
- InetAddress.*getByName("www.server.com");*
- InetAddress.*getByName("11.22.33.44");*
- Restituisce l'indirizzo IP
  - stringa di 32 bit per IPv4
  - stringa di 128 bit per IPv6

194

## Networking



- classe Socket
  - crea il canale di comunicazione con il server
- Socket(InetAddress addr, int port)
  - socket = new Socket(serverAddr, port);
- Per leggere e scrivere
  - getInputStream(socket)
  - getOutputStream(socket)

195

## Networking



- Scrivere nel socket
- PrintWriter out =
  new PrintWriter(
  new BufferedWriter(
  new OutputStreamWriter(
  socket.getOutputStream())),
  **true**); //**Autoflush**
- out.println(strToSend);

196

## Networking



- Leggere dal socket
- BufferedReader in =
  new BufferedReader(
  new InputStreamReader(
  socket.getInputStream()));
- in.readLine(), in.read(), …

197

## Localhost

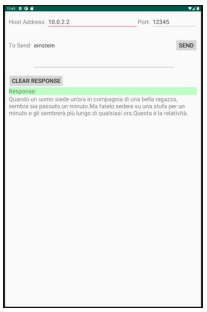


- Se non si dispone della connessione Internet e/o di un server
- Dall'emulatore
  - Indirizzo locale **10.0.2.2**
  - Corrisponde a "localhost"

198

## Networking

23-SocketRaw





24-SocketRawProgressBar

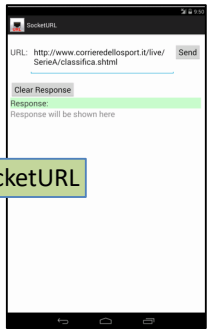
199

## URL

- Il trasferimento di pagine web è l'operazione più comune
  - esistono delle classi apposite
- HttpURLConnection
  - openConnection()
  - getInputStream()
- e poi si procede come prima leggendo i dati dallo stream

200

## Networking




25-SocketURL

201

## Documenti HTML

- Dati in documenti HTML
  - difficile estrarli
- Esistono delle librerie che implementano il parsing di documentl HTML
  - es. JSOUP
- Per utilizzare una libreria
  - procurarsi il file .jar (es. jsoup-1-1.15.3.jar)
  - memorizzarlo nella cartella libs del progetto
  - Aggiungere la dipendenza nel file build.gradle:
    - implementation files('libs/jsoup-1.15.3.jar')

202

## JSoup

- La classe Jsoup permette
  - parsing di documenti HTML
  - estrarre singoli parti del documento
- Esempi:
  - **Document** doc = **Jsoup**.connect("http://en.wikipedia.org/").get();
  - **Element** e = doc.getElementById("id");
  - **Elements** e = doc.select("[class=id");

203

## Errore: Cleartext not permitted

- API > 27:




26-ClassificaSerieA

```
D/DEBUG: doc is null
D/DEBUG: onPreExecute()
D/DEBUG: doInBackground: values[0]=URL=http://www.legaseriea.it/it/serie-a/classifica
W/System.err: java.io.IOException: Cleartext HTTP traffic to www.legaseriea.it not permitted
```

https://developer.android.com/training/articles/security-config

A partire da Android 9 (API 28), connessioni in chiaro (http) sono disabilitate per default. Occorre usare connessioni cifrate (https) o abilitare quelle in chiaro.

Pochi siti usano ancora connessioni in chiaro (http), quasi tutti usano connessioni cifrate (https)

204

ANDROID Mobile Programming

# Data Storage



205

## Data storage: memoria persistente





- Interna
  - Flash memory (o simile)
  - con varie partizioni
    - /system
    - /data
    - /cache
    - /boot
    - ....
- Esterna
  - microSD

206

## Data storage





- Interno vs esterno
  - memoria flash e SD card




- Privato vs pubblico
- Spazio privato all'app
  - ogni app ha uno spazio (directory dedicata) privato
    - nella memoria interna
    - anche nella memoria esterna (da API>=29)
  - Accessibile SOLO all'app a cui appartiene
- Spazio pubblico
  - file che possono essere condivisi con altre app

207

## Data storage: spazio privato (interno)





- spazio privato (interno)
  - getDataDir()

```
/data/user/0/com.example.myapp/      ← getDataDir()
├── cache/                           ← getCacheDir()
├── code_cache/                      ← cache for compiled/optimized code
├── databases/                       ← app's SQLite databases (.db)
├── files/                           ← getFilesDir()
│   ├── myfile.txt
│   └── ...
├── shared_prefs/                    ← SharedPreferences XML files
│   ├── settings.xml
│   └── ...
├── no_backup/                       ← data NOT backed up to cloud
└── app_*/                           ← private dirs for specific libraries
```

208

## Context and Environment





- getDataDir() (Context API24+)
  - `/data/user/0/<package_name>/`
- getDataDirectory() (Environment)
  - `/data/`
- Context
  - Superclasse di Activity
  - Metodi per le directories relative all'app
  - Sia spazio privato che directory condivise
- Environment
  - helper class con metodi statici per informazioni di sistema
  - Directories di sistema, condivise (metodi deprecati)

209

## Data storage: metodi del Context





- getDataDir() → `/data/user/0/<package_name>/`
- getFilesDir() → `/data/user/0/<package_name>/files/`
- getCacheDir() → `/data/user/0/<package_name>/cache/`
- getNoBackupFilesDir() → `/data/user/0/<package_name>/no_backup/`
  - prevista per files che non hanno copie di backup in Cloud, es. su Google Drive
- getDatabasePath(name) → `/data/user/0/<package_name>/databases/<name>`
- getDir(name, mode) → `/data/user/0/<package_name>/files/app_<name>`
  - se la directory non esiste viene creata (con «app_» prima di <name>)
  - mode = MODE_PRIVATE
  - Fino a API 24 prevedeva anche
    MODE_WORLD_READABLE
    MODE_WORLD_WRITABLE

Lo «0» in «/0/» fa riferimento all'ID utente. Se ci sono più utenti verranno identificati da «/n/». 0 è l'utente principale.

210

## Data storage: memoria privata esterna



- getExternalFilesDir()
  → `/storage/emulated/0/Android/data/<package_name>/files/`
- getExternalFilesDirs()
  - Restituisce un array
    - [0] → `/storage/emulated/0/Android/data/<package_name>/files/` (flash memory)
    - [1] → `/storage/1234-5678/Android/data/<package_name>/files/` (SD card)
- getExternalFilesDirs(Environment.DIRECTORY_PICTURES)
  - Restituisce un array
    - [0] → `/storage/emulated/0/Android/data/<package_name>/files/Pictures`
    - [1] → `/storage/1234-5678/Android/data/<package_name>/files/Pictures`

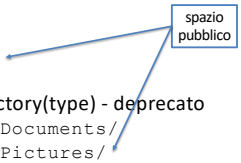
  ... (anche per altri tipi ovviamente)
- getExternalCacheDir()
  → `/storage/emulated/0/Android/data/<package_name>/cache/`

211

## Data storage: metodi dell'Environment



- getDataDirectory() → `/data/`
- getRootDirecotry() → `/system/`
- getDownloadCacheDirectory() → `/cache/`
- getExternalStorageDirectory()
  → `/storage/emulated/0/`
- getExternalStoragePublicDirectory(type) - deprecato
  → `/storage/emulated/0/Documents/`
  → `/storage/emulated/0/Pictures/`
  → `/storage/emulated/0/Movies/`
  ...




212

## External e SD card



- «External»
  - fa riferimento allo spazio privato dell'app, quindi esterno allo spazio privato
  - non necessariamente «memoria esterna»
  - per questo abbiamo usato «pubblico»
- Quindi «External» può essere
  - sulla memoria interna (non removibile)
    `/storage/emulated/0/`
  - sulla SD card (removibile)
    `/storage/XXXX-XXXX/`
    XXXX-XXXX è l'identificativo del volume dove è montata l'SD

213

## MediaStore e SAF



- getExternalStoragePublicDirectory(type)
  - permette l'accesso diretto ai file condivisi
  - deprecato da API29, introdotto MediaStore
- MediaStore è un Content Provider
  - pensato per gestire i dati condivisi
    - Immagini, Video, Audio, Downloads
- I dati sono memorizzati nelle directory pubbliche
- Il ContentProvider MediaStore mantiene indici
  - è un database SQL
  - vedremo i ContentProvider
- SAF (Storage Access Framework)
  - simile, per file arbitrari, non-media

214

## Permessi



- Android 9 e precedenti, API ≤ 28
  - READ_EXTERNAL_STORAGE
    - Accesso in lettura a tutti i file nella memoria esterna
      - /storage/emulated/0/... e SD cards.
  - WRITE_EXTERNAL_STORAGE
    - Accesso in scrittura **e lettura** alla memoria esterna
- da Android 10, API 29+
  - App sono «**sandboxed**» anche per la memoria esterna
  - Possono accedere solo al proprio spazio
- Nuovi permessi

215

## Permessi



- READ_EXTERNAL_STORAGE
- WRITE_EXTERNAL_STORAGE
  - Esistono per compatibilità ma l'accesso non viene dato
- API30+
  - WRITE_EXTERNAL_STORAGE ignorato
  - READ_EXTERNAL_STORAGE permette l'accesso solo ai media file
  - **MANAGE_EXTERNAL_STORAGE**
    - nuovo, permette un accesso senza restrizioni simile a WRITE
    - Richiede approvazione utente in Settings (→ "All files access").
    - Pensato per file managers, backup apps, antivirus, etc.
    - Apps devono giustificare la necessità per Google Play
- API33+
  - READ_MEDIA_IMAGES
  - READ_MEDIA_VIDEO
  - READ_MEDIA_AUDIO

216

## SD card



- Controllare se esistono SD card montate

```
Context context= getApplicationContext();
StorageManager sm = (StorageManager)
context.getSystemService(Context.STORAGE_SERVICE);
for (StorageVolume volume : sm.getStorageVolumes()) {
   Log.d("STORAGE", "Description: " + volume.getDescription(context));
   Log.d("STORAGE", "Directory: " + volume.getDirectory());
   Log.d("STORAGE", "Removable: " + volume.isRemovable());
}
```

- e poi getExternalFilesDirs()

→ `/storage/XXXX-XXXX/`

217

## SD card



- File[] dirs = getExternalFilesDirs(null);
  - restituisce un array
- dirs[0]: primary external (emulated internal)
  `/storage/emulated/0/`
- dirs[1]: se c'è, removable SD card path
  `/storage/XXXX-XXXX/Android/data/<package>/files/`
- API >= 29
  - l'SD è trattata come la memoria interna, ogni app ha il proprio spazio
  - sono accessibili solo le directory comuni (Documents, Pictures, Movies...)
  - Chiedendo i permessi
    - READ_EXTERNAL_STORAGE
    - WRITE_EXTERNAL_STORAGE
  - I permessi anche per API<29, ma si accede a tutto

218

## SharedPreferences



- Classe SharedPreferences
  - dati in coppie di chiave-valore
- 2 metodi della classe Activity
  - getSharedPreferences("filename")
    - quando si vogliono usare più file di "preferenze" (dati)
    - `/data/user/0/<package>/shared_prefs/<filename>.xml`
  - getDefaultSharedPreferences()
    - `/data/user/0/<package>/shared_prefs/<package>_preferences.xml`
  - restituiscono un oggetto SharedPreferences

Attenzione a non usare getPreferences (senza "Shared"), che serve per preferenze non condivise con altre activity dell'app. getPreferences restituisce /data/user/0/<package>/shared_prefs/MainActivity.xml

219

## SharedPreferences



- SharedPreferences obj;
- Leggere i dati: si usa "get"
  - Boolean v = obj.getBoolean("KEY");
- Scrivere i dati: serve un "editor"
  - lo si ottiene con il metodo edit
  - SharedPreferences.Editor editor = obj.edit();
- Con l'editor si può usare "put":
  - editor.putBoolean("KEY", bool_value);
  - editor.commit();

220

## File



- Per ogni app il sistema operativo prevede una directory privata
  - solo l'app può accedere a questa directory
  - se l'app viene disinstallata, la directory viene cancellata
    - anche quella sulla SD!
- Per creare e scrivere un file
  1. Chiamare openFileOutput(fileName, mode)
     - restituisce un FileOutputStream
  2. Scrivere nel file (write())
  3. Chiudere lo stream (close())

221

## File



```
String FILENAME = "hello_file";
String string = "hello world!";

FileOutputStream fos = openFileOutput(FILENAME, Context.MODE_PRIVATE);
fos.write(string.getBytes());
fos.close();
```

- La modalità può essere
  - MODE_PRIVATE (sarebbe stato meglio MODE_CREATE)
    - Private lo è di default se è nello spazio privato
    - Da API 29 gli spazi pubblici sono accessibili tramite MediaStore
  - MODE_APPEND
  - ~~MODE_WORLD_READABLE (leggibile da tutti)~~
  - ~~MODE_WORLD_WRITABLE (scrivibile da tutti)~~ Obsolete API 17

222

## File



- Per leggere un file
  1. Chiamare openFileInput(fileName)
     - restituisce un FileInputStream
  2. Leggere dal file (read())
  3. Chiudere lo stream (close())

È possibile usare un file "statico" mettendolo nella directory "res/raw" dell'applicazione. Lo si può leggere usando openRawResource passando come argomento l'identificatore R.raw.*<filename>*. Il metodo openRawResource restituisce un InputStream che può essere usato per leggere il file.

223

## File



- getFilesDir()
  - Restituisce la directory privata dell'app (dove vengono salvati i file)
- getDir()
  - Crea (o apre se esiste) una directory all'interno dello spazio privato dell'app
- deleteFile()
  - cancella il file nello spazio privato
- filelist()
  - Restituisce un array di file, quelli presenti nello spazio privato

224

## File temporanei



- Per i file temporanei si può usare una directory cache
  - Android cancellerà i file in questa directory SE necessario (quando manca spazio)
- getCacheDir()
  - restituisce la directory cache
  - è comunque responsabilità dell'app cancellare i file
  - non si dovrebbe usare la directory cache per file grandi (grandezza massima raccomandata 1MB)

225

## External Storage



```
/* Checks if external storage is available for read and write */
public boolean isExternalStorageWritable() {
   String state = Environment.getExternalStorageState();
   if (Environment.MEDIA_MOUNTED.equals(state)) {
      return true;
   }
   return false;
}
/* Checks if external storage is available to at least read */
public boolean isExternalStorageReadable() {
   String state = Environment.getExternalStorageState();
   if (Environment.MEDIA_MOUNTED.equals(state) ||
      Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
      return true;
   }
   return false;
}
```

226

## SQL – Quick tutorial



- Android fornisce supporto per database SQL
  - Structured Query Language
  - Il linguaggio standard per database relazionali
- Tavole
  - ogni riga è un elemento
  - ogni colonna rappresenta un campo

| ID | Nome | Cognome | Esame | Voto |
|---|---|---|---|---|
| 1356251 | Giuseppe | Verdi | MP | 18 |
| 1367288 | Attilio | Bianchi | Algoritmi | 25 |
| 5267712 | Valentino | Rossi | MotoGP | 30 |
| 7126714 | Giuseppe | Verdi | Musica | 30 |
| 1562689 | Marco | Arancione | MP | 23 |

227

## SQL – Quick tutorial



- Tavole
  - **CREATE**
    - crea una nuova tavola (o anche altro, es. view)
  - ALTER
    - Modifica una tavola (o altro)
  - DROP
    - Cancella una tavola
- Contenuto
  - **SELECT**
    - legge uno o più record (righe) di una tavola (o view)
  - **INSERT**
    - Inserisce un record
  - **UPDATE**
    - Modifica uno o più record
  - **DELETE**
    - Cancella uno o più record

228

## SQL – Quick tutorial



- Chiave principale
  - una colonna (o più colonne) che serve da identificatore univoco per ogni record
- Esempio creazione tavola:

```
SQL> CREATE TABLE CUSTOMERS (
   ID          INT             NOT NULL,
   NAME        VARCHAR (20)    NOT NULL,
   AGE         INT             NOT NULL,
   ADDRESS     CHAR (25) ,
   SALARY      DECIMAL (18, 2),
   PRIMARY KEY (ID)
);
```

229

## SQL – Quick tutorial



- Esempi di inserimento

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (1, 'Marco', 32, 'Napoli', 2000.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (2, 'Adele', 25, 'Milano', 1500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY) VALUES (3, 'Carla', 23, 'Palermo', 2000.00 );
```

```
INSERT INTO CUSTOMERS VALUES (4, 'Maria', 25, 'Roma', 6500.00 );

INSERT INTO CUSTOMERS VALUES (5, 'Pasquale', 27, 'Firenze', 8500.00 );

INSERT INTO CUSTOMERS VALUES (6, 'Renato', 22, 'Venezia', 4500.00 );
```

230

## SQL – Quick tutorial



- Tavola prodotta

| ID | NAME | AGE | ADDRESS | SALARY |
|---|---|---|---|---|
| 1 | Marco | 32 | Napoli | 2000 |
| 2 | Adele | 25 | Milano | 1500 |
| 3 | Carla | 23 | Palermo | 2000 |
| 4 | Maria | 25 | Roma | 6500 |
| 5 | Pasquale | 27 | Firenze | 8500 |
| 6 | Renato | 22 | Venezia | 4500 |

231

## SQL – Quick tutorial



- Esempi di select

```
SELECT ID, NAME, AGE, ADDRESS, SALARY FROM  CUSTOMERS;
```

```
+----+----------+-----+---------+---------+
| ID | NAME     | AGE | ADDRESS | SALARY  |
+----+----------+-----+---------+---------+
|  1 | Marco    |  32 | Napoli  | 2000.00 |
|  2 | Adele    |  25 | Milano  | 1500.00 |
|  3 | Carlo    |  23 | Palermo | 2000.00 |
|  4 | Maria    |  25 | Roma    | 6500.00 |
|  5 | Pasquale |  27 | Firenze | 8500.00 |
|  6 | Renato   |  22 | Veneia  | 4500.00 |
+----+----------+-----+---------+---------+
```

```
SELECT NAME, SALARY FROM CUSTOMERS;
SELECT NAME, SALARY FROM CUSTOMERS WHERE SALARY > 5000;
SELECT NAME, SALARY FROM CUSTOMERS WHERE SALARY > 5000 AND AGE < 26;
SELECT NAME, SALARY FROM CUSTOMERS WHERE NAME LIKE 'M%';
SELECT NAME FROM CUSTOMERS ORDER BY NAME ASC;
SELECT * FROM CUSTOMERS ORDER BY NAME, SALARY DESC;
```

232

## SQL – Quick tutorial



- Esempi di update

```
UPDATE CUSTOMERS SET ADDRESS = 'Salerno' WHERE ID = 5
```

| ID | NAME | AGE | ADDRESS | SALARY |
|---|---|---|---|---|
| 1 | Marco | 32 | Napoli | 2000 |
| 2 | Adele | 25 | Milano | 1500 |
| 3 | Carla | 23 | Palermo | 2000 |
| 4 | Maria | 25 | Roma | 6500 |
| 5 | Pasquale | 27 | Salerno | 8500 |
| 6 | Renato | 22 | Venezia | 4500 |

```
UPDATE CUSTOMERS SET SALARY = 1000.00;
```

| ID | NAME | AGE | ADDRESS | SALARY |
|---|---|---|---|---|
| 1 | Marco | 32 | Napoli | 1000 |
| 2 | Adele | 25 | Milano | 1000 |
| 3 | Carla | 23 | Palermo | 1000 |
| 4 | Maria | 25 | Roma | 1000 |
| 5 | Pasquale | 27 | Salerno | 1000 |
| 6 | Renato | 22 | Venezia | 1000 |

233

## Database SQL



- Android fornisce supporto per database SQL
  - solo all'interno dell'app
- Per usare un database
  - Creare una sottoclasse di SQLiteOpenHelper
  - sovrascrivere il metodo onCreate()
- Quindi si crea un nuovo Helper:
  - dbHelper = new DatabaseOpenHelper(this);
- Dal quale si ricava un database
  - SQLiteDatabase db = dbHelper.getWritableDatabase();

234

## Database SQL



- Sul database si possono applicare comandi standard SQL
- Il database (tavola) viene creato (comando CREATE) nel metodo onCreate() della sottoclasse DatabaseOpenHelper
- Metodi per inserire/cancellare/aggiornare:
  - db.insert()
  - db.delete()
  - db.update()

235

## Database SQL



- ContentValues
  - Classe «contenitore»
  - Scriviamo i valore di un record del database in un oggetto ContentValues
  - db.insert(nomeTavola, null, oggettoContentValues)
    - Se oggettoContentValues != nullo
      - il secondo parametro viene ignorato
    - Se oggettoContentValues == nullo
      - deve essere il nome di una colonna
        - » Se oggettoContentValues è nullo, l'SQL generato sarebbe «INSERT INTO table () VALUES ();» che è **invalido**. Inserendo esplicitamente la colonna si evita il problema

236

## Database SQL



- db.query
  - Esegue una query SQL sul database
- Cursor
  - Classe specifica per i risultati della query
  - Contiene tutti i record restituiti dalla query
  - Si può «iterare» per esaminare i singoli record
    - cursor.moveToFirst()
    - cursor.moveToNext()
    - cursor.getInt(columnIndex) ... Anche getString etc
    - cursor.getColumnIndex(columnName)

237

## Database



```
public class MyDBHelper extends SQLiteOpenHelper {
   private static final int DATABASE_VERSION = 1;
   private static final String TABLE_NAME = "MYTABLE";
   private static final String CREATE_CMD = ... // comando per la creazione della tavola
      //qualcosa del tipo: "CREATE TABLE MYTABLE
                           (_ID INTEGER PRIMARY KEY,
                           stringa TEXT NOT NULL,
                           numero INTEGER);"

   MyOpenHelper(Context context) {
      super(context, TABLE_NAME, null, DATABASE_VERSION);
   }
   @Override
   public void onCreate(SQLiteDatabase db) {
      db.execSQL(CREATE_CMD);
   }
   @Override
   public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
      // override necessario
   }
}
```

238

## Database



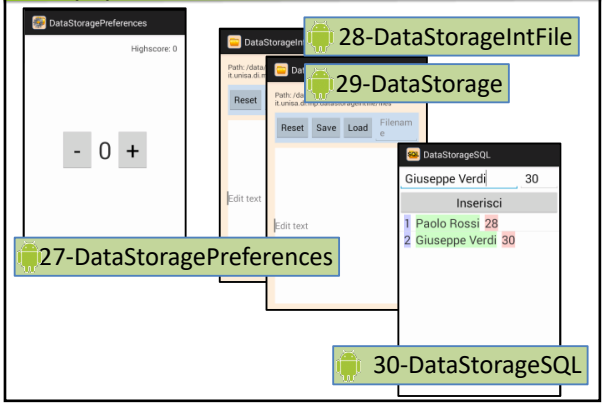
```
public class MainActivity extends AppCompatActivity {
   private SQLiteDatabase db = null;
   private MyDBHelper dbHelper;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);
       dbHelper = new MyDBHelper(this); // Create a new DatabaseHelper
       db = dbHelper.getWritableDatabase(); // Get the underlying database for writing
       ContentValues record = new ContentValues();
       record.put("STRINGA", "Roberto");
       record.put("NUMERO", 2);
       db.insert("MYTABLE", null, record);
       Cursor cursor = db.query(
               "MYTABLE",     // The table to query
               projection,    // The columns to return
               selection,     // The columns for the WHERE clause
               selectionArgs, // The values for the WHERE clause
               null,          // don't group the rows
               null,          // don't filter by row groups
               sortOrder      // The sort order
       );
       if (cursor.moveToFirst()) {
           do {
               Integer id = cursor.getInt(cursor.getColumnIndex("_ID"));
               String nome = cursor.getString(cursor.getColumnIndex("STRINGA"));
               String voto = cursor.getString(cursor.getColumnIndex("NUMERO"));
               Log.d("MYDEBUG","_ID="+id+" STRINGA="+nome+" NUMERO="+voto);
           } while (cursor.moveToNext());
       }
   }
}
```

```
SELECT COUNT(CustomerID), Country
FROM Customers
WHERE Name LIKE 'D%'
GROUP BY Country
HAVING COUNT(CustomerID) > 100
ORDER BY COUNT(CustomerID) DESC;
```

239

## Data Storage






240

ANDROID
Mobile Programming



# Grafica



241

## Grafica



- Un'immagine può essere disegnata in
  - un oggetto View
    - grafica semplice, senza necessità di cambiamenti
  - un oggetto Canvas
    - grafica complessa, aggiornamenti frequenti
- Classe Drawable
  - rappresenta un oggetto che può essere disegnato
    - un'immagine, ma anche un colore, una forma, etc
    - ShapeDrawable – una forma
    - BitmapDrawable – una matrice di pixels
    - ColorDrawable – un colore (uniforme)

242

## Grafica



- L'oggetto Drawable deve essere inserito nell'oggetto View
  - direttamente nel file XML
  - in modo programmatico
    - View.setImageDrawable()




31-GraficaSimpleImg

243

## Animazioni



- Classe Animation
  - permette di definire delle animazioni da applicare alle immagini
- Descritte con file XML
  - rotazione
  - traslazione
  - ridimensionamento
  - trasparenza
  - con controllo di vari parametri
    - es., punto di pivot, velocità, etc.

```
<rotate
  android:startOffset="0"
  android:duration="2000"
  android:repeatCount="3"
  android:fromDegrees="0"
  android:toDegrees="720"
  android:pivotX="50%"
  android:pivotY="50%" />
```

244

## ObjectAnimator



```
<set xmlns:android="http://schemas.android.com/apk/res/android"
android:ordering="sequentially">

<set android:ordering="together">
  <objectAnimator
     android:propertyName="scaleX"
     android:valueFrom="1"
     android:valueTo="1.5"
     android:duration="500" />
  <objectAnimator
     android:propertyName="scaleY"
     android:valueFrom="1"
     android:valueTo="1.5"
     android:duration="500" />
</set>

<objectAnimator
  android:propertyName="rotation"
  android:valueFrom="0"
  android:valueTo="180"
  android:duration="800" />
</set>
```

245

## Animation vs ObjectAnimator



| Attributo XML | Significato | Impostazione con ObjectAnimator |
|---|---|---|
| android:duration | Durata dell'animazione (ms) | anim.setDuration(long) |
| android:startOffset | Ritardo prima di iniziare | anim.setStartDelay(long) |
| android:repeatCount | Numero di ripetizioni | anim.setRepeatCount(int) |
| android:repeatMode | Modalità di ripetizione (restart/reverse) | anim.setRepeatMode(ValueAnimator.RESTART / REVERSE) |
| android:interpolator | Funzione di interpolazione | anim.setInterpolator(Interpolator) |
| android:fromAlpha / toAlpha | Valori iniziale e finale di trasparenza | ObjectAnimator.ofFloat(view, "alpha", 1f, 0f) |
| android:fromDegrees / toDegrees | Angoli di rotazione | ObjectAnimator.ofFloat(view, "rotation", 0f, 180f) |
| android:fromXDelta / toXDelta | Spostamento orizzontale | ObjectAnimator.ofFloat(view, "translationX", …) |
| android:fromYDelta / toYDelta | Spostamento verticale | ObjectAnimator.ofFloat(view, "translationY", …) |
| android:fromXScale / toXScale | Scala orizzontale | ObjectAnimator.ofFloat(view, "scaleX", …) |
| android:fromYScale / toYScale | Scala verticale | ObjectAnimator.ofFloat(view, "scaleY", …) |
| android:pivotX / pivotY | Punto di rotazione o scala | view.setPivotX(float), view.setPivotY(float) |
| android:fillAfter / fillBefore | Mantiene o ripristina stato finale | Non esiste: le proprietà restano modificate realmente (usare listener o reverse()) |
| android:zAdjustment / fillEnabled | Solo effetti visivi temporanei | Non applicabili a ObjectAnimator |

246

## Animation vs ObjectAnimator



- Animation
  - non modificano le proprietà della view
  - sono un effetto grafico
  - terminata l'animazione la view rimane esattamente come era prima
    - anche se esiste l'attributo fillAfter che mantiene lo stato grafico finale
      - ma è solo a livello di rendering
        - » es. se l'animazione sposta la view, getX e getY restituiscono quelle iniziali, anche con fillAfter
        - » Risultato: se il sistema ridisegna il tutto, la view torna nello stato originale
- ObjectAnimator
  - modifica realmente la view
  - gli effetti sono permanenti
  - può operare su oggetti arbitrari, non necessariamente view

247

## Animation vs ObjectAnimator



| Comportamento | `Animation` + `fillAfter` | `ObjectAnimator` |
|---|---|---|
| Modifica visiva | ✅ Sì | ✅ Sì |
| Modifica proprietà reali della View | ❌ No | ✅ Sì |
| Persistenza dopo redraw/layout | ❌ No (torna com'era) | ✅ Sì |
| Coordinate (`getX()`, `getY()`) aggiornate | ❌ No | ✅ Sì |
| Touch/click funzionano nella nuova posizione | ❌ No | ✅ Sì |
| Utilizzo di `fillAfter` necessario | ✅ | ❌ No (stato già reale) |

248

## ObjectAnimator



- può aggiornare qualsiasi campo (dell'oggetto)
  - ci deve essere un metodo «set<NomeCampo>»
  - opzionalmente il metodo «get<NomeCampo>»
- ObjectAnimator anim = ObjectAnimator.ofFloat(obj, "<NomeCampo>", 0f, 100f);
- E il tipo (ofFloat, ofInt, of Argb...) deve corrispondere

249

## ObjectAnimator



- ObjectAnimator anim = ObjectAnimator.ofFloat(obj, "x", 0f, 100f);

```
public class Punto {
   private float x;
   public void setX(float x) {
      this.x = x;
      System.out.println("x = " + x);
   }
   public float getX() {
      return x;
   }
}

Punto p = new Punto();
ObjectAnimator anim = ObjectAnimator.ofFloat(p, "x", 0f, 100f);
anim.setDuration(2000);
anim.start();
```

Durante l'animazione Android esegue:

```
p.setX(0f);
p.setX(5f);
p.setX(10f);
...
...
p.setX(100f);
```

250

## Esempio






251

## Custom Views

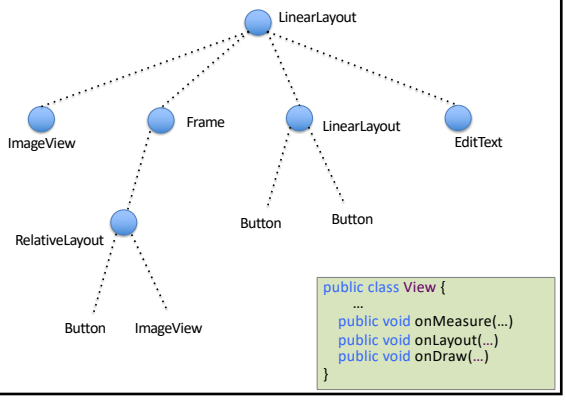


- Android ha molti widget
  - Pulsanti, Liste, ImageView, etc, etc.
- Per esigenze particolare possiamo definire dei widget personalizzati
- Permettono un maggiore controllo sulla grafica
  - ovviamente sono più complicati da usare

252

## Albero delle View





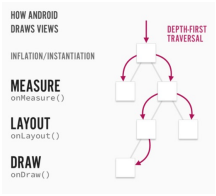

253

## Meccanismo di layout





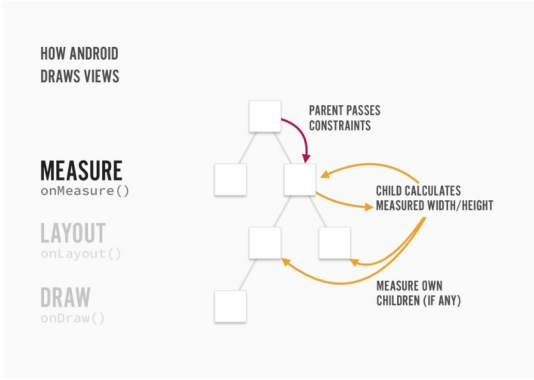

- 3 Fasi
  - Misura
  - Posizionamento
  - Disegno
- Per ognuna
  - Visita dell'albero delle view




254

## Fase di misurazione

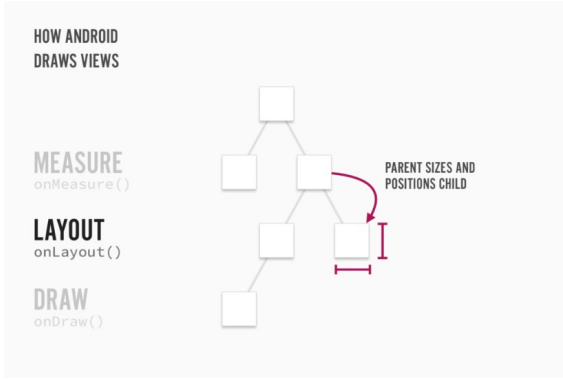





255

## Fase di layout

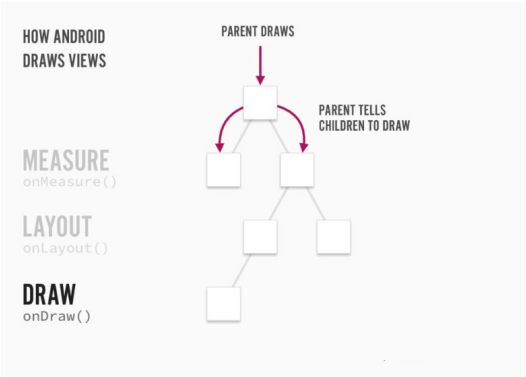





256

## Fase di disegno






257

## Fase di misurazione



- le informazioni viaggiano in due direzioni:
  - **Dal genitore al figlio:** i vincoli sulle dimensioni, tramite i MeasureSpec.
  - **Dal figlio al genitore:** le dimensioni calcolate, tramite measuredWidth e measuredHeight.
- Il genitore chiede a un figlio di misurarsi chiamando:
  - figlio.measure(widthMeasureSpec, heightMeasureSpec)

258

## Fase di misurazione



- "measured" size (width&height)
  - quanto grande la view vorrebbe essere
- size reale
  - quanto grande la view sarà in realtà
- Approccio top-down nell'albero
  - ogni view chiede ai suoi figli quanto vorrebbero essere grandi
  - Classe LayoutParams
    - I figli comunicano al parent dimensioni e posizione desiderate
  - Classe MeasureSpec
    - Il padre comunica ai figli le dimensioni effettive

259

## MeasureSpec



```
int widthMode = MeasureSpec.getMode(widthMeasureSpec);
int width = MeasureSpec.getSize(widthMeasureSpec);
int heightMode = MeasureSpec.getMode(heightMeasureSpec);
int height = MeasureSpec.getSize(heightMeasureSpec);}
```

- Usato per passare i vincoli sulla grandezza da un parent ai figli
  - widthMode e heigthMode
    - MeasureSpec.UNSPECIFIED
      - Usato per chiedere ai figli quanto spazio vorrebbero
    - MeasureSpec.EXACTLY
      - Per imporre una grandezza specifica
    - MeasureSpec.AT_MOST
      - Per imporre un limite alla grandezza
- width e heigth
  - in pixels

260

## Fase di misurazione (onMeasure)



- Ogni view può esprimere la propria preferenza usando la classe ViewGroup.LayoutParams
  - Un numero (di pixel)
  - MATCH_PARENT
  - WRAP_CONTENT
- onMeasure chiamata
  - Per richiedere quanto spazio il figlio vorrebbe
  - Per comunicare lo spazio assegnato
- Quando il processo di misurazione finisce ogni view deve aver definito
  - measuredWidth
  - measuredHeight

261

## Fase di layout (onLayout)



- Visita top-down dell'albero delle view
- View parent
  - decide la grandezza e la posizione (in accordo alle misure fatte nella fase precedente)
- onLayout(bool, int, int, int, int)
  - Vengono comunicate le coordinate dello spazio assegnato
  - Bool indica se c'è stato un cambiamento dalla precedente assegnazione

262

## Fase di layout (onLayout)



- Visita top-down dell'albero delle view
- View parent
  - decide la grandezza e la posizione (in accordo alle misure fatte nella fase precedente)
- onLayout(bool, int, int, int, int)
  - Vengono comunicate le coordinate dello spazio assegnato
  - Bool indica se c'è stato un cambiamento dalla precedente assegnazione

263

## Fase di disegno (onDraw)



- Se un CustomWidget
  - **non contiene figli**
    - nulla da fare in onLayout
    - Si può eliminare questo override.
- Per un **ViewGroup personalizzato**
  - implementare onLayout()
  - chiamare layout() su ciascun figlio
    - scegliendone la posizione.

264

## Meccanismo di layout



- "Container Views"
  - RelativeLayout, LinearLayout, …
- Il meccanismo di layout inizia quando viene chiamato il metodo requestLayout su una View dell'albero
  - solitamente un widget chiama requestLayout quando ha bisogno di altro spazio
- requestLayout mette un evento nella coda degli eventi UI
  - Quando l'evento viene processato, ogni container view ha la possibilità di interagire con i figli

265

## onMeasure()



```
public class MyView extends Views{
   MyView(Context context) {
     super(context);
   }
   …
   @Override
   public void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
     ….
   }
   ….
}
```

**Parameters**

| | |
|---|---|
| `widthMeasureSpec` | `int`: horizontal space requirements as imposed by the parent. The requirements are encoded with `View.MeasureSpec`. |
| `heightMeasureSpec` | `int`: vertical space requirements as imposed by the parent. The requirements are encoded with `View.MeasureSpec`. |

- gli "int" sono codificati con View.MeasureSpec
  - getSize e getMode (slide precedenti)

266

## onMeasure()



```
public class MyView extends Views{
   MyView(Context context) {
     super(context);
   }

   …

   @Override
   public void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        int width = resolveSize(
             getSuggestedMinimumWidth(),
             widthMeasureSpec);

        int height = resolveSize(
             getSuggestedMinimumHeight(),
             heightMeasureSpec);

        setMeasuredDimension(width, height);
   }

   ….
}
```

267

## Layout



- Nella fase di Layout le view container comunicano la posizione effettiva a ogni view figlio

```
public class MyView extends Views{
   …
   @Override
   public void onLayout (int x1, int y1, int x2, int y2) {
        Log.d("DEBUG","onLayout");
        Log.d("DEBUG","coordinate x1="+x1+" y1="+y1+"  x2="+x2+"  y2="+y2);
        int smw = getSuggestedMinimumWidth();
        int smh = getSuggestedMinimumHeight();
        Log.d("DEBUG","onLayout smw="+smw+"   smh="+smh);
   }
   ….
}
```

268

## Disegnare nel canvas



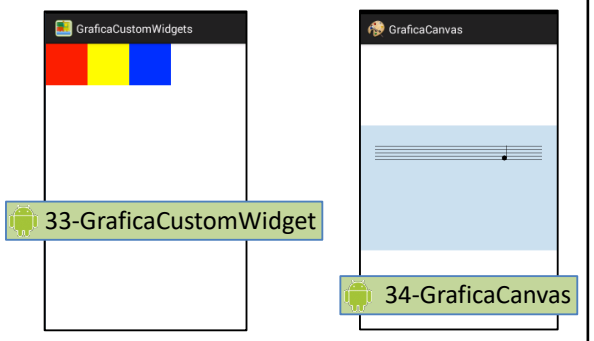
- Quando la view è stata posizionata verrà disegnata
  - metodo onDraw
- Oggetto Paint e metodi dell'oggetto Canvas

```
public class MyView extends Views{
   …
   @Override
   public void onDraw  (Canvas canvas) {
      //Codice per disegnare la view
   }
   ….
}
```

269

## Esempi






270

## Multitouch



- MotionEvent
  - rappresenta un movimento registrato da una periferica
    - penna, trackball, mouse
    - dita sul display
- Il movimento è rappresentato con
  - ACTION_CODE
    - cambiamento avvenuto
  - ACTION_VALUES
    - Posizione e proprietà del movimento
      - tempo, sorgente, pressione e altro

271

## Multitouch



- Focalizziamo l'attenzione sul Multitouch
- Multitouch display
  - Permettono il rilevamento di uno o più tocchi
- "Pointer"
  - il singolo evento (es. un dito che tocca lo schermo)
- Un MotionEvent rappresenta
  - un singolo pointer
  - a volte più di un pointer
    - in questo caso possiamo accedere ai singoli pointer usando un indice
- Ogni pointer ha un ID unico per tutto il tempo in cui esiste
  - L'indice di un MotionEvent multiplo NON è il pointer ID
    - il pointer ID è costante
    - l'indice può cambiare per eventi successivi

272

## Multitouch



- MotionEvents ACTION_CODES:
  - ACTION_DOWN
    - un dito tocca lo schermo ed è il primo
  - ACTION_POINTER_DOWN
    - un dito tocca lo schermo ma non è il primo
  - ACTION_MOVE
    - un dito che è sullo schermo si muove
  - ACTION_POINTER_UP
    - un dito che è sullo schermo non lo tocca più
  - ACTION_UP
    - l'ultimo dito sullo schermo viene alzato

273

## Multitouch



| Remark | Action | ID |
|---|---|---|
| Primo dito | ACTION_DOWN | 0 |
| | ACTION_MOVE | 0 |
| Secondo dito | ACTION_POINTER_DOWN | 1 |
| | ACTION_MOVE | 0,1 |
| Primo dito | ACTION_POINTER_UP | 0 |
| | ACTION_MOVE | 1 |
| Secondo dito | ACTION_UP | 1 |

274

## Multitouch



| Remark | Action | ID |
|---|---|---|
| Primo dito | ACTION_DOWN | 0 |
| | ACTION_MOVE | 0 |
| Secondo dito | ACTION_POINTER_DOWN | 1 |
| | ACTION_MOVE | 0,1 |
| Secondo dito | ACTION_POINTER_UP | 1 |
| | ACTION_MOVE | 0 |
| Primo dito | ACTION_UP | 0 |

275

## Multitouch



| Remark | Action | ID |
|---|---|---|
| Primo dito | ACTION_DOWN | 0 |
| Secondo dito | ACTION_POINTER_DOWN | 1 |
| Terzo dito | ACTION_POINTER_DOWN | 2 |
| | ACTION_MOVE | 0,1,2 |
| Secondo dito | ACTION_POINTER_UP | 1 |
| Primo dito | ACTION_POINTER_UP | 0 |
| Terzo dito | ACTION_UP | 2 |

276

## Multitouch



- Per gestire i MotionEvent:
  - getActionMasked()
    - Restituisce l'Action Code dell'evento
  - getPointerCount()
    - Numero di pointer coinvolti
  - getActionIndex()
    - indice di un pointer
  - getPointerID(int pointerIndex)
  - getX(int pointerIndex)
  - getY(int pointerIndex)
  - findPointerIndex(int pointerId)

277

## Multitouch



- Android notifica l'oggetto View
  - View.onTouchEvent(MotionEvent e)
- onTouchEvent()
  - deve restituire un Boolean
    - true, se l'evento è stato consumato
    - false, altrimenti
- Oggetti che vogliono ricevere la notifica
  - View.onTouchListener
  - View.setOnTouchListener()

278

## Multitouch



- onTouch
  - verrà invocata quando c'è un evento
    - finger down, up o movimento
- onTouch viene chiamata prima che la View venga notificata dell'evento
  - anche onTouch deve restituire un Boolean
    - true, se l'evento è stato consumato
    - false, altrimenti

Se restituiamo false le successive chiamate (ACTION_MOVE, ACTION_UP, …) per quell'evento non verranno fatte al nostro listener. In altre parole "non consumare" l'evento equivale a dire "non sono interessato a questo evento"

279

## Multitouch

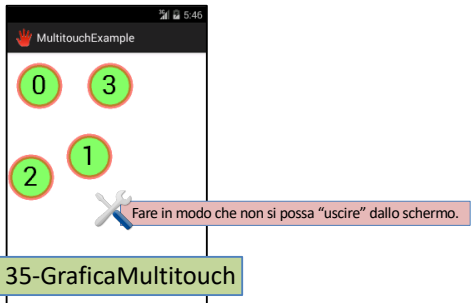


- Spesso si ha la necessità di gestire una combinazione di eventi
- Es. Il doppio "click" equivale a
  - ACTION_DOWN
  - ACTION_UP
  - ACTION_DOWN
  - ACTION_UP
  - in rapida successione

280

## Multitouch






Fare in modo che non si possa "uscire" dallo schermo.

Fare in modo che i cerchi non possano sovrapporsi, segnalando che due cerchi si sono toccati con un cambio di colore.

281

## GestureDetector



- Classe GestureDetector
  - permette di riconoscere dei gesti fatti sul display
- Gesti riconosciuti:
  - onDown() - primo contatto con lo schermo
  - onSingleTap() – come l'onClick
  - onDoubleTap() - doppio "click"
  - onFling() – scorrimento con rilascio
  - …e altri

ADVANCED TOPIC: è possibile anche definire dei gesti personalizzati attraverso un apposito tool di Android e poi riconoscerli tramite il GestureDetector

282

## GestureDetector



- Bisogna creare un GestureDetector
  - che implementa l'interfaccia
  - GestureDetector.OnGestureListener interface
- Riscrivere (override) il metodo onTouchEvent
  - che viene chiamato in risposta a un gesto
  - questo metodo delega il riconoscimento del gesto al metodo GestureDetector.OnGestureListener

283

## ViewAnimator e ViewFlipper



- View Animator
  - viewGrup speciale per passare fra più view
  - animazione cambiamento fra view
- ViewFlipper
  - sottoclasse di ViewAnimator
  - crea animazione fra 2 o più view del contenitore
  - Solo una view per volta viene visualizzata
  - Può anche cambiare views ad intervalli regolari
- metodi
  - showNext()
  - showPrevious()

284

## GestureDetector






36-GraficaGestureFlip

Implementare anche il fling per il decremento

285

## ANDROID Mobile Programming



# Media player



286

## MediaPlayer



- AudioManager
  - controlla le sorgenti audio e l'output
    - volume
- MediaPlayer
  - Play di audio e video
- Sorgente dati
  - Risorse locali
  - URI (interni)
  - URL

287

## MediaPlayer



- Play di un file in res/raw

```
MediaPlayer mediaPlayer = MediaPlayer.create(context, R.raw.sound_file);
mediaPlayer.start(); // no need to call prepare(); create() does that for you
```

- Play di un file da URL

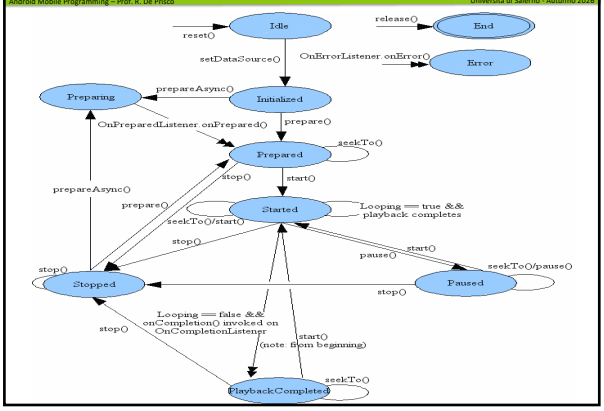
```
String url = "http://......"; // your URL here
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(url);
mediaPlayer.prepare(); // might take long! (for buffering, etc)
mediaPlayer.start();
```

setDataSource() richiede la gestione di IOException o di IllegalArgumentException

288

## MediaPlayer






289

## AudioAttributes per il MediaPlayer



- Per far capire ad Android che tipo di audio si riproduce

```
AudioAttributes attrs = new AudioAttributes.Builder()
    .setUsage(AudioAttributes.USAGE_MEDIA)
    .setContentType(AudioAttributes.CONTENT_TYPE_MUSIC)
    .build();

mediaPlayer.setAudioAttributes(attrs);
```

290

## MediaPlayer



- Rilasciare la risorsa

```
mediaPlayer.release();
mediaPlayer = null;
```

- mediaPlayer.start();
- mediaPlayer.pause();
- mediaPlayer.stop();

- Attenzione all'uso asincrono (prepareAsynch)
  - necessario per file audio forniti come URL esterni
  - Play da fare in/dopo onPrepareListener.onPrepared()

291

## MediaPlayer



- Audio focus
- Poiché c'è un solo canale di output
  - l'utilizzo da parte di più applicazioni può essere un problema
    - es. se stiamo ascoltando musica potremmo non sentire l'arrivo di un messaggio

- È possibile gestire l'accesso contemporaneo usando l'audio focus
  - un'app richiede l'audio focus per usare l'audio
  - se lo perde deve o smettere di suonare o abbassare il proprio volume

292

## AudioAttributes per l'AudioFocusRequest



- Servono ad Android per stabilire le priorità rispetto al canale audio fra le varie app

```
AudioFocusRequest focusRequest =
    new AudioFocusRequest.Builder(AudioManager.AUDIOFOCUS_GAIN)
        .setAudioAttributes(attrs)
        .setOnAudioFocusChangeListener(afListener)
        .setAcceptsDelayedFocusGain(true)   // opzionale
        .setWillPauseWhenDucked(false)     // per fare il ducking esplicito
        .build();
```

293

## Tipi di richieste AudioFocus



- AUDIOFOCUS_GAIN
  - Richiesta normale

- AUDIOFOCUS_GAIN_TRANSIENT
  - Richiesta per uso breve

- AUDIOFOCUS_GAIN_TRANSIENT_MAY_DUCK
  - Permette agli altri canali di continuare a basso volume

- AUDIOFOCUS_GAIN_TRANSIENT_EXCLUSIVE
  - Per uso breve con silenzio degli altri canali

294

## AudioFocus



- Tipi di audio
  - AudioAttributes.*CONTENT_TYPE_MUSIC*
  - AudioAttributes. *CONTENT_TYPE_MOVIE*
  - AudioAttributes.*CONTENT_TYPE_SPEECH*
  - AudioAttributes.*CONTENT_TYPE_SONIFICATION*
  - AudioAttributes.*CONTENT_TYPE_UNKNOWN*
- L'abbassamento del volume dipende (ducking) anche del tipo di audio
  - Es. SPEECH non viene abbassato, è meglio gestirlo nel listener così si può fare pausa e poi riprendere

295

## OnAudioFocusChangeListener



- Cambio (perdita/ripristino) AudioFocus
  - onAudioFocusChange(int focus)
    - AudioManager.AUDIOFOCUS_GAIN
    - AudioManager.AUDIOFOCUS_LOSS
    - AudioManager.AUDIOFOCUS_LOSS_TRANSIENT
    - AudioManager.AUDIOFOCUS_LOSS_TRANSIENT_CAN_DUCK
- L'app deve gestire opportunamente i vari casi
  - Es. fermando la riproduzione su LOSS
  - Riprendendola su GAIN
  - Abbasandola su LOSS_TRANSIENT_CAN_DUCK

296

## FocusRequest



- Occorre creare un oggetto AudioFocusRequest
  - Contiene informazioni
    - Tipo di audio
    - Comportamento rispetto a altri canali audio
    - Listener
- Listener = oggetto OnAudioFocusChangeListener
  - Implementa il metodo onAudioFocusChange

Approfondimenti su AudioFocus:
https://developer.android.com/reference/android/media/AudioFocusRequest

297

## MediaPlayer






37-MusicPlayer

Inserire altri brani nel MusicPlayer

Implementare la lista dei brani con un ListView

298

# ANDROID Mobile Programming



# Sensori

299

299

## Sensori



- Molti smartphones, tablet hanno sensori
  - di movimento
    - forze di accelerazione e di rotazione
      - accelerometri, bussola, giroscopio
  - di ambiente
    - temperatura, pressione, umidità
      - termometri, barometri
  - di posizione
    - posizione fisica
      - magnetometro, bussola, giroscopio
- Forniscono dati "grezzi"
  - accuratezza dipende dalla qualità

300

## Sensori



- SensorManager ci dice
  - sensori disponibili
  - caratteristiche del singolo sensore
    - range massimo
    - accuratezza
    - etc.
- ci permette di
  - leggere i dati grezzi del sensore
  - usare Listeners sui cambiamenti dei dati

301

## Accuratezza



- Molti sensori hanno un accuratezza FISSA
  - Accelerometro, Giroscopio, Sensore di prossimità, Sensore di luce, Barometro
  - Deriva dalla qualità dell'hardware
  - Android non invia mai onAccuracyChanged()
- Magnetometro (bussola)
  - Accuratezza dipende da disturbi magnetici esterni
- Sensori «fusion»
  - Calcolo derivante da vari sensori fra cui il magnetometro
    - Sensor.TYPE_ROTATION_VECTOR
    - Sensor.TYPE_GAME_ROTATION_VECTOR
    - Sensor.TYPE_GEOMAGNETIC_ROTATION_VECTOR

302

## Sensori



- Poche device hanno tutti i tipi di sensori
  - a volte più di un sensore dello stesso tipo
  - ecco un elenco parziale:

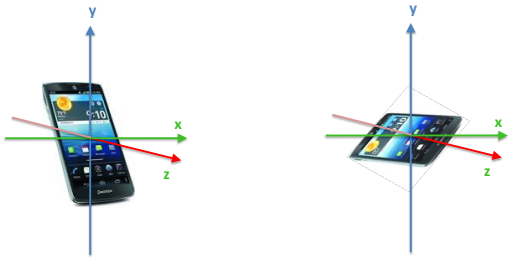
| Sensore | Tipo | Descrizione |
|---|---|---|
| TYPE_ACCELEROMETER | Hw | Misura le forze in m/s2 applicate alla device (inclusa la gravità) nelle 3 direzioni (x,y,z) |
| TYPE_AMBIENT_TEMPERATURE | Hw | Misura la temperatura dell'ambiente in gradi centigradi |
| TYPE_GRAVITY | Hw o Sw | Misura le forze di gravità sui 3 assi (x,y,z) |
| TYPE_GYROSCOPE | Hw | Misura la velocità di rotazione in rad/s nelle 3 direzioni (x,y,z) |
| TYPE_MAGNETIC_FIELD | Hw | Misura il campo magnetico sui tre assi |
| TYPE_ORIENTATION | Sw | Misura la rotazione riferita ai 3 assi |
| TYPE_RELATIVE_HUMIDITY | Hw | Misura la % di umidità dell'ambiente |
| TYPE_LIGHT | Hw | Misura la luminosità dell'ambiente |

303

## Sensori



- Coordinate fisse: se la device ruota
  - il sistema di riferimento rimane fermo
  - cambieranno i valori letti sugli assi




304

## Sensori



- L'attività deve implementare SensorEventListener

```
public class SensorActivity extends Activity implements SensorEventListener {
    ...
}
```

- Poi si deve controllare se il sensore esiste

```
private SensorManager mSensorManager;
...
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
if (mSensorManager.getDefaultSensor(Sensor.TYPE_LIGHT) != null){
        // Success! There's an ambient light sensor.
 }
else {
        // Failure! No light sensor
 }
```

305

## Sensori



```
@Override
protected void onResume() {
   super.onResume();
   mSensorManager.registerListener(this, mLight, SensorManager.SENSOR_DELAY_NORMAL);
}

@Override
protected void onPause() {
   super.onPause();
   mSensorManager.unregisterListener(this);
}
```

- Velocità di campionamento
  - SENSOR_DELAY_NORMAL (0,2sec)
  - SENSOR_DELAY_GAME (0,02sec)
  - SENSOR_DELAY_UI (0,06sec)
  - SENSOR_FASTEST (0sec )
- Registrazione e rilascio in onResume e onPause
  - per evitare di consumare la batteria

306

## Sensori

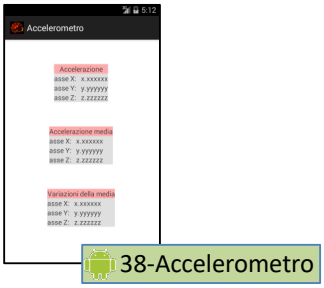


```
@Override
public final void onCreate(Bundle savedInstanceState) {
   ....
   mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
   mLight = mSensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
}

@Override
public final void onAccuracyChanged(Sensor sensor, int accuracy) {
   // Do something here if sensor accuracy changes.
}

@Override
public final void onSensorChanged(SensorEvent event) {
   // The light sensor returns a single value, many sensors return 3 values, one for each axis.
   float lux = event.values[0];
   ...
}
```

307

## Sensori






308

# ANDROID Mobile Programming



# Notifiche

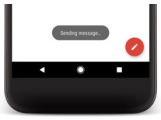


309

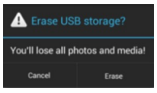
## Notifiche



- Notifiche: informazioni all'utente al di fuori dell'interfaccia grafica dell'app
- Toast
  - brevi messaggi temporanei
    - anche personalizzati




- Dialog
  - interazione con l'utente



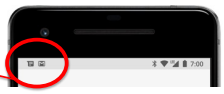

- Notifiche
  - status bar e cassetto delle notifiche

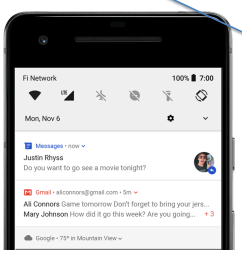
310

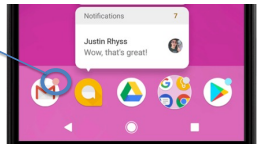
## Notifiche



- Messaggi
  - sulla barra di stato come icone
  - cerchietto sull'icona
  - nel "cassetto delle notifiche" con testo aggiuntivo






- Visibili fino a:
  - cancellati da utente
  - revocati dall'app

311

## Notifiche



- Da Android 5.0 (API 21)
  - notifiche a comparsa (heads-up notification) per notifiche che sono ritenute «urgenti»
- Messaggio in una finestra
  - Scompare dopo poco



- Condizioni che possono determinare un messaggio a scomparsa
  - utente in full screen
  - notifica con alta priorità (API ≤ 25)
  - canale di notifica con alta priorità (API ≥ 26)

312

## Notifiche



- Da Android 8.0 (API ≥ 26)
  - "canali" di notifica
  - prima ogni app aveva un singolo "canale"
- Ogni notifica deve essere "assegnata" a un canale
  - altrimenti non viene visualizzata
- I canali possono essere controllati dall'utente
  - disabilitare solo determinati canali
  - controllare il comportamento (suoni, visualizzazione)

313

## Notifiche - compatibilità



- Per API vecchie conviene usare le classi
  - NotificationCompat e NotificationManagerCompat
  - ottenere la migliore compatibilità possibile

```
dependencies {
    implementation "com.android.support:support-compat:28.0.0"
}
```

314

## Notifiche



- Canale
  - ch = new NotificationChannel(...)
  - Proprietà: ch.setLightColor(Color.*GREEN*);
    nome, importanza, suono, pattern vibrazione...
  - notificationManager.**createNotificationChannel**(ch);
- Notifica
  - nb = new NotificationCompat.Builder(...)
  - Proprietà: nb.setContentIntent(**pendingIntent**);
    nb.setSmallIcon(R.drawable.*ancora*)
    ...
  - notificationManager.notify(notification_id, nb.build());

A partire dall'API level 26 l'icona deve essere un'immagine con il fondo trasparente e un solo colore. Altrimenti compare un quadrato bianco



315

## Notifiche



- da API33
  - permesso POST_NOTIFICATIONS
  - nel manifesto e a run_time

```
//Da API 33 (Ansdroid 13) occorre richiedere il permesso a runtime!
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) { // Android 13+
    if (ContextCompat.checkSelfPermission( context: this,  permission: "android.permission.POST_NOTIFICATIONS")
            != PackageManager.PERMISSION_GRANTED) {
        // Request the notification permission at runtime
        ActivityCompat.requestPermissions( activity: this,
                new String[]{"android.permission.POST_NOTIFICATIONS"},
                requestCode: 1001); // Request code for POST_NOTIFICATIONS
        return;
    }
}
```

316

## Notifiche






https://developer.android.com/training/notify-user/build-notification

317

## ANDROID Mobile Programming



# Alarms

318

## Alarms



- Permettono di eseguire intent in funzione del tempo
  - Es. lanciare un intent una volta al giorno
- Un'applicazione che usa un alarm riesce ad eseguire porzioni di codice anche se l'applicazione è terminata
  - cioè gli alarm permettono di "attivare" un'app
- Un alarm è attivo anche se il telefono va in modalità di sleep
  - l'alarm può causare la ripresa dell'attività
  - oppure potrà essere gestito quando l'utente rimette il telefono in modalità normale

319

## Alarms



- Alarms «imprecisi» (inexact)
  - Per questo tipo di allarme il sistema non garantisce l'esecuzione al tempo richiesto
    - il SO operativo può modificare i triggerTime per minimizzare wakeups e l'uso della batteria
    - In particolare se ci sono restrizioni sull'uso della batteria
  - Da API 31, il sistema garantisce che vengano eseguiti entro un'ora dal tempo richiesto
- Esempi di utilizzo
  - backup periodici
  - aggiornamenti dati non critici
    - es. aggiornamenti app meteo ogni 30-60 minuti
  - Notifiche del tipo «ricontrolla fra qualche ora»

320

## Alarms



- Alarms «precisi» (exact)
  - Vengono eseguiti al tempo richiesto
  - Da API 31, è necessario un permesso
- Necessari
  - azioni che devono essere eseguite all'orario preciso
    - una sveglia
    - un timer
    - un reminder di un appuntamento

```
<manifest ...>
  <uses-permission android:name="android.permission.SCHEDULE_EXACT_ALARM"/>
  <uses-permission android:name="android.permission.USE_EXACT_ALARM"/>
  <application ...>
    ...
  </application>
</manifest>
```

321

## Alarms



- Gli alarms rimangono attivi fino a quando
  - vengono cancellati
  - la periferica viene spenta
- Esempi di alarms
  - (Sistema) app per gli MMS: usa alarm per controllare periodicamente i messaggi non spediti (retry scheduler)
  - (Sistema) Settings: usa un alarm per rendere la periferica non visibile via Bluetooth dopo un determinato tempo
  - Un'app che controlla le previsioni del tempo (su un sito esterno) ad intervalli regolari

322

## Alarms



- Per usare gli alarm in un'app
  - AlarmManager=getSystemService(Context.ALARM_SERVICE)
- Allarmi imprecisi
  - set(…)
  - setInexactRepeating(...)
  - setAndAllowWhileIdle(…)
- Allarmi precisi
  - garantiti se non ci sono restrizioni sulla batteria
    - void setExact(…),
    - setExactAndAllowWhileIdle(…)

    (nota: non c'è "Repeating" come per gli imprecisi)
  - Garantito sempre
    - setAlarmClock(…)

323

## Doze e standby – risparmio batteria



- Doze mode, si attiva
  - non in carica
  - schermo spento
  - device ferma (non in movimento fisico)
  - Nessuna attività in esecuzione (musica, navigazione)
- Comporta
  - Alarms e altre attività ritardate
  - Accesso rete non abilitato per le app in background

324

## App standby mode



- Serve a limitare attività di background per app non molto utilizzate
  - Standby riguarda la singola app
  - Doze riguarda l'intero sistema
- Si attiva
  - quando un'app non è in foreground
  - e non viene utilizzata per molto tempo
- Comporta
  - Attività di background sospesa
  - Alarms ritardati

325

## Alarms



| Feature | Exact Alarms | Inexact Alarms |
|---|---|---|
| Definition | Trigger at the **exact time** specified. | Trigger at a **flexible time window** to optimize system efficiency. |
| API Methods | `setExact()` / `setExactAndAllowWhileIdle()` / `setAlarmClock()` | `set()` / `setRepeating()` |
| Fires in Doze Mode? | Yes (with `setExactAndAllowWhileIdle()` or `setAlarmClock`). | No (may be deferred). |
| Precision | High — Fires exactly at the specified time. | Low — System may adjust timing for efficiency. |
| Battery Consumption | Higher due to precise scheduling. | Lower as alarms are grouped for efficiency. |
| Use Cases | Critical tasks like reminders, user-facing alarms. | Background tasks, non-critical reminders. |
| Permission Requirements | `SCHEDULE_EXACT_ALARM` (Android 12+ for | None. |

| Feature | setExact | setExactAndAllowWhileIdle | setAlarmClock |
|---|---|---|---|
| Fires in Doze Mode? | No | Yes | Yes |
| Fires in App Standby? | No | Yes | Yes |
| Wakes Device? | No (may defer if asleep) | Yes | Yes |
| User Interaction? | No | No | Yes (shows alarm icon and triggers UI intent) |
| Battery Impact | Low | Medium | High |
| Permission Requirements | None | `SCHEDULE_EXACT_ALARM` (Android 12+) | None |

326

## app da background a foreground

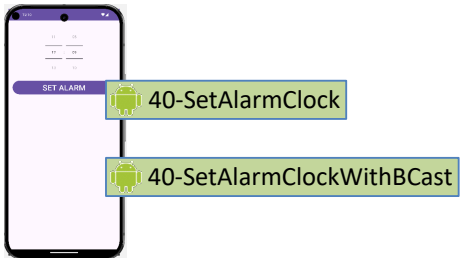


- Da API 29 ci sono restrizioni
- Un alarm che deve riportare in foreground un'app non può farlo direttamente
- Si deve creare un PendingIntent che genera un messaggio di broadcast
  - con PendingIntent.*getBroadcast*(...)
  - l'alarm lancerà il messaggio di brodcast
- Nel ricevitore del messaggio brodacast
  - Si crea una notifica
  - sarà l'utente a riportare l'app in foreground

327

## Alarm






328

## ANDROID Mobile Programming



# Content Providers, Broadcast, Services

329

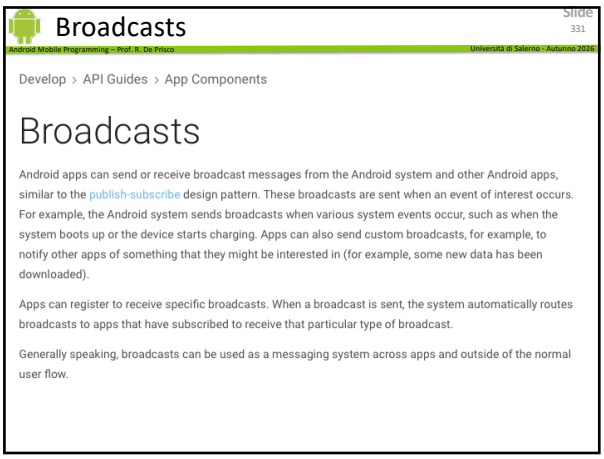
329

## Oltre le Activity

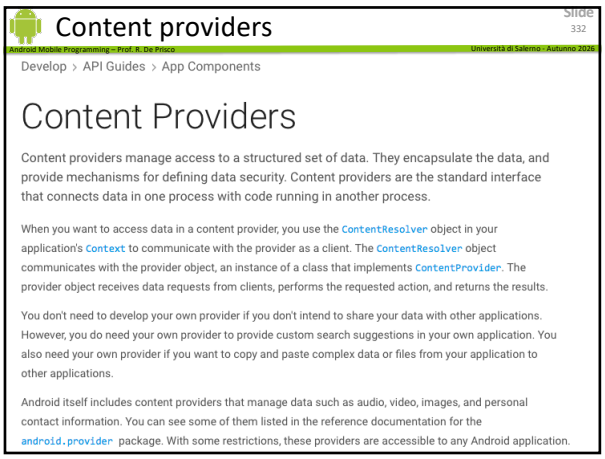


- 4 componenti fondamentali di Android
  - Activity
  - Broadcasts
  - Content Providers
  - Services
- Finora abbiamo parlato delle activity
  - servono per lo sviluppo delle app!
  - Le altre componenti sono di ausilio e servono in casi particolari, ma in alcuni casi sono cruciali

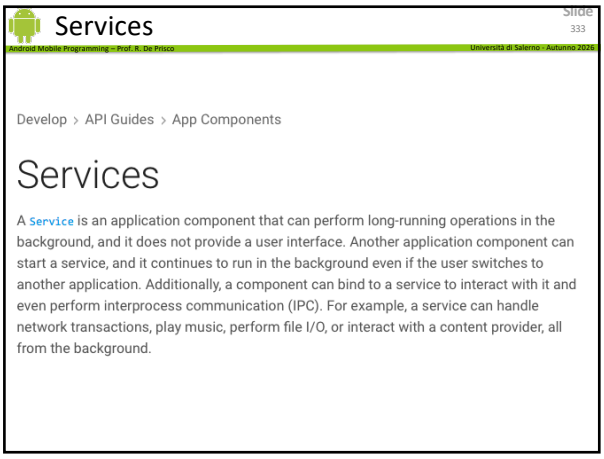
330



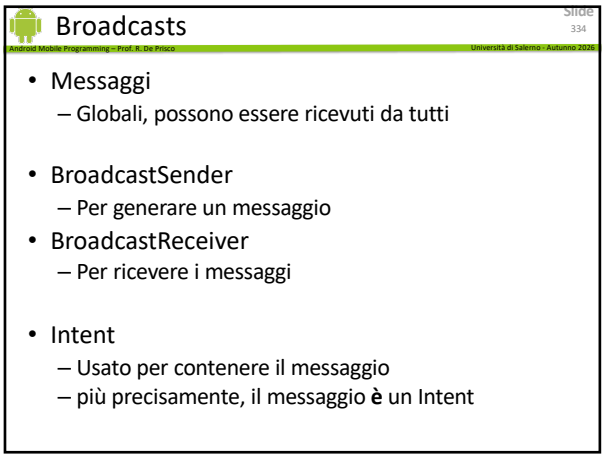

331




332



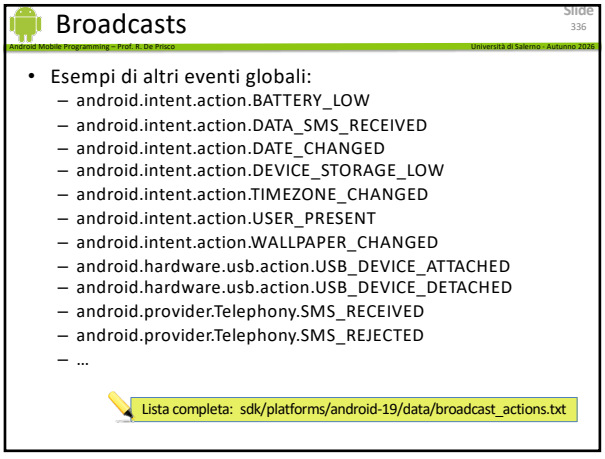

333




334




335




336

## BroadcastReceiver



- BroadcastReceiver
  - Devono essere dichiarati/registrati
- Dichiarati nel manifesto (statici)
  - Da API 26, solo alcuni broadcast receiver per eventi di sistema possono essere registrati staticamente, es.
    - android.intent.action.BOOT_COMPLETED
    - android.intent.action.ACTION_BATTERY_LOW
    - android.intent.action.ACTION_POWER_CONNECTED
- Registrati (dinamicamente)
  - registerReceiver()

337

## BroadcastReceiver



- Dichiarati nel manifesto (statici)

```
<receiver android:name=".MyBroadcastReceiver"  android:exported="true">
  <intent-filter>
    <action android:name="android.intent.action.BOOT_COMPLETED"/>
    <action android:name="android.intent.action.AIRPLANE_MODE" />
  </intent-filter>
</receiver>
```

  - il filtro specifica i messaggi di interesse
  - android:exported="true"
    - Rende visibile il receiver.
    - In generale permette a altre app di accedere all'activity, service, receiver che specifica il valore true. E' un meccanismo di sicurezza per impedire l'accesso dall'esterno a determinate componenti dell'app

338

## BroadcastReceiver



- android:exported
  - obbligatorio specificarlo

`android:exported`

This element sets whether the activity can be launched by components of other applications:

- If "`true`", the activity is accessible to any app, and is launchable by its exact class name.
- If "`false`", the activity can be launched only by components of the same application, applications with the same user ID, or privileged system components. This is the default value when there are no intent filters.

If an activity in your app includes intent filters, set this element to "`true`" to allow other apps to start it. For example, if the activity is the main activity of the app and includes the `category` "`android.intent.category.LAUNCHER`".

If this element is set to "`false`" and an app tries to start the activity, the system throws an `ActivityNotFoundException`.

339

## BroadcastReceiver



- Registrati dinamicamente
  - Sottoclasse di BroadcastReceiver
    - implementare il metodo onReceive

```
public class MyBroadcastReceiver extends BroadcastReceiver {
      @Override
      public void onReceive(Context context, Intent intent) {
            Log.d("DEBUG","Hello, another minute is gone!!!");
            counter++;
            String min = "minuti";
            if (counter == 1) min = "minuto";
            tv.setText(counter+" "+min);
      }
}

IntentFilter intentFilter = new IntentFilter(Intent.ACTION_TIME_TICK);
registerReceiver(receiver, intentFilter);
```

340

## Spedire Custom Broadcasts



- Custom broadcast
  - il "messaggio" è un Intent
    - implicito, è un'azione
      - i = Intent("it.unisa.di.mp.app.BROADCAST_PERSONALIZZATO")
      - i.putExtra("key", "value")
    - esplicito
      - i.component = ComponentName(" it.unisa.di.mp.app", "it.unisa.di.mp.app.MyBroadcastReceiver")
      - ricevuto solo dal receiver specificato
- Per spedire
  - sendBroadcast(i)

341

## Broadcast Locali



- Simili a quelli globali
  - comunicazione limitata all'interno di una singola app
    - fra le activity e i frammenti dell'app
- Spedire tramite il "Local" Bcast manager
  - LocalBroadcastManager.getInstance(context).sendBroadcast(intent)
- **LocalBroadcasts obsoleti!**
  - alternative più efficienti: ViewModel e LiveData, WorkManager

342

## Broadcasts



- Rimuovere i receivers se non servono più
- unregisterReceiver(bcastReceiver)
- Attenzione a usare un pattern consistente
  - es. registrato in onCreate, rimosso in onDestroy
  - es. registrato in onResume, rimosso in onPause
  - es. Non rimuovere in onSaveIstanceState
    - non viene chiamato se si usa il pulsante Back

343

## Riassumendo



- Il ricevitore riceve l'Intent tramite il metodo
  - onReceive(Context c, Intent i)

1. Il "ricevitore" si "registra" usando registerReceiver() (disponibile nel Context per quello globale)
2. L'evento viene creato (da qualche altra componente del sistema)
3. Android notifica il ricevitore chiamando onReceive()

344

## Riassumendo



- Se la registrazione è statica
  - il ricevitore viene registrato durante il Boot del sistema (oppure quando l'app viene istallata)

> **Note:** If your app targets API level 26 or higher, you cannot use the manifest to declare a receiver for *implicit* broadcasts (broadcasts that do not target your app specifically), except for a few implicit broadcasts that are exempted from that restriction. In most cases, you can use scheduled jobs instead.

- Se la registrazione è dinamica
  - il ricevitore viene registrato quando si chiama
    - Context.registerReceiver()
      - per i broadcast system-wide
    - ~~LocalBroadcastManager.registerReceiver()~~
      - ~~per i broadcast locali all'app~~
- è possibile anche revocare la registrazione usando unregisterReceiver()

345

## Spedizione broadcast



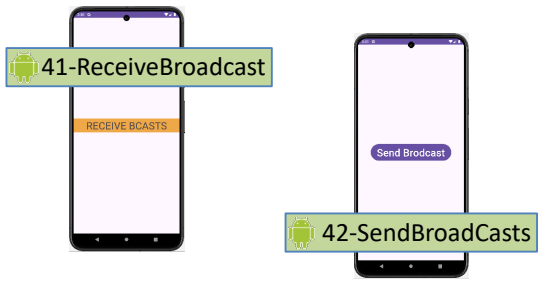
- Per spedire un messaggio si usa il metodo
  - Context.sendBroadcast(Intent i)
    - messaggi globali
  - ~~LocalBroadcastManager.sendBroadcast(Intent i)~~
    - ~~messaggi locali~~
- sendBroadcast(Intent i, String permission)
  - Se si specifica anche una stringa di permesso l'intent verrà consegnato solo ai ricevitori che hanno il permesso
  - il permesso lo deve avere l'app nel Manifesto

sendBroadcast(Intent i) è disponibile sia nel LocalBroadcastManager che nel Context. Si utilizza il primo per messaggi locali all'app ed il secondo per messaggi system-wide. sendBroadcast(Intent i, String permission) è disponibile solo nel Context.

346

## Esempi



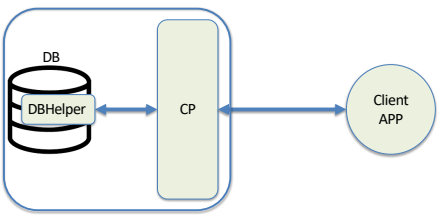



347

## ContentProvider



- Interfacce per accedere a un Database
  - Database = Contenuto (dati)
  - Provider = espone i dati del database
  - progettati per condividere i dati




348

## ContentProvider



- Esempi
  - Browser (info su bookmarks, history)
  - Call Log (info sulle chiamate)
  - Contact (info sui contatti presenti in rubrica)
  - Media (lista dei file multimediali utilizzabili)
  - UserDictionary (lista delle parola digitate)
  - … altri
- Accesso tramite un ContentResolver
  - interfaccia simile a quella di un database
  - comandi SQL-like
    - QUERY, INSERT, UPDATE, DELETE, etc
  - in più, notifiche su cambiamenti dei dati

349

## ContentProvider



- DB memorizza i dati in tabelle
- I client possono far riferimento ad uno specifico ContentProvider usando un URI
- URI: **content:**//*authority*/*path*/*id*
  - authority: specifica il content provider
  - path: specifica la tabella
  - id: specifica un particolare record
    - devono essere primary keys nel database
- Quindi un ContentProvider è in realtà una particolare "vista" del database
  - sullo stesso database possiamo aver più CP, con URI diversi

350

## ContentProvider



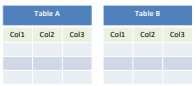
- Esempi di URI
  - content://com.android.contacts/contacts/
- Authority è com.android.contacts
- La tabella richiesta è "contacts"
- Non c'è nessun ID, quindi l'URI identifica l'intera tabella dei contatti

351

## CustomContentProvider



- Scrivere una classe che estende ContentProvider
- Definire un URI che specifica i dati da esporre
  - Un'intera tabella
  - Una parte
  - Dati provenienti da varie tabelle




- content://AUTHORITY/PATH
  - AUTHORITY = stringa univoca, definita dall'utente
  - PATH = stringa univoca, definita dall'utente

352

## CustomContentProvider



- AUTHORITY va dichiarata nel manifesto

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... package="it.unisa.mp.customcontentprovider">

  <application
    ....                                          AUTHORITY
    <activity>
        ...
    </activity>
    <provider
    android:authorities="it.unisa.mp.customcontentprovider"
    android:name="it.unisa.mp.customcontentprovider.CustomProvider"
    android:enabled="true"
    android:exported="true"/>
</application>
```

353

## CustomContentProvider



- Espone il metodo query

```
public Cursor query(Uri uri,
                    String[] projection \\ colonne
                    String selection    \\ SQL selection
                    String[] args       \\ SQL args
                    String sortOrder)   \\ ordinamento
```

- Restituisce un Cursor che ci permette di iterare sull'insieme di record restituiti dalla query

354

## Client

- Accede tramite il metodo query
  - e anche altri metodi eventualmente disponibili
    - insert, delete, etc.
- Il primo parametro è l'URI

```
Uri CONTENT_URI = Uri.parse(
"content://it.unisa.mp.customcontentprovider/ALL_STUDENTS"
);
```

- PATH="ALL_STUDENTS"
  - usato per individuare la tabella

355

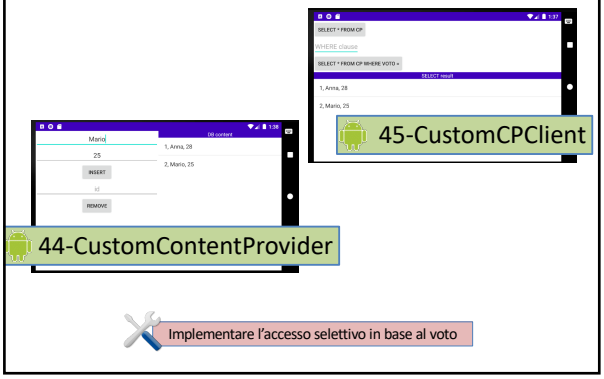
## Client

- A partire da API 30
  - occorre dichiarare nel manifesto i Content Provider a cui si vuole accedere
- <queries> … </queries>

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="it.unisa.mp.customcpclient">
    <queries>
        <provider android:authorities="it.unisa.mp.customcontentprovider" />
    </queries>
    <application>
        ...
    </application>
</manifest>
```

356

## CustomContentProvider

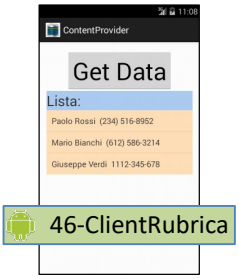



357

## Client Rubrica

- Serve il permesso:

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




358

## Services

- Servono a eseguire operazioni complesse che possono richiedere molto tempo e/o sono prolungate nel tempo, es.
  - scaricare un file da Internet
  - sincronizzare informazioni locali con un server
  - riconoscere determinate posizioni GPS
- classe Services
  - usare dei Services esistenti
  - definire dei nuovi Services

359

## Services

- Services non interagiscono con l'utente
  - non c'è una UI
- Servono per eseguire delle operazioni in background
- L'app interagisce con il servizio
  - interazione dipende da cosa offre il servizio
  - e da cosa vuole fare l'app

360

## Services



- Una volta partito, il Service può continuare la sua esecuzione fino a che la device è accesa
  - potrebbe anche essere interrotto se occorrono le risorse che esso usa
  - potrebbe anche terminare volontariamente
- Nell'utilizzo tipico un Service fatto partire da un'app termina la propria esecuzione dopo aver eseguito l'operazione richiesta
  - ma non sempre
- per default, il Service gira nel main thread dell'app che lo ha fatto partire
  - a meno che non sia esplicitamente specificato di usare un altro thread

361

## Services



- bound
  - permette (ad activity, frammenti o altri services) di "agganciarsi"
  - rimane attivo fino a quando ci sono componenti agganciate
- unbound
  - funziona indipendentemente dalla componente che lo ha lanciato
  - continua a funzionare anche se la componente che lo ha lanciato termina

362

## Services (unbound)



- StartService()
  - fa partire il service
- StopService()
  - lo ferma
- Altri metodi
  - definiti dal service

363

## Services (bound)



- bindService
  - una componente dell'app richiede di "agganciarsi" al service
    - diventa un client del service
- Client
  - comunicano con il service
- Un service bound
  - gira fino a che ci sono client agganciati
  - più client possono agganciarsi

364

## Services



- Un service può funzionare in entrambe le modalità
  - bound
  - unbound
- Dipende da come lo si fa partire
  - onStartCommand()
    - unbound, ma poi permette ai client di agganciarsi
  - onBind()

365

## Services



- Occorre dichiarare il Service nel Manifesto:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
...
    <application
    ...
        <service
            android:name="MyService"
        </service>
    </application>
</manifest>
```

366

## Esempi








47-PlayMusicService

367

## Services





- Nell'esempio visto il servizio è nella stessa app del client
- Per approfondire i Service:

http://developer.android.com/guide/components/bound-services.html

368

## ANDROID Mobile Programming



# Fine del corso!



370