



# ANDROID Mobile Programming



... o almeno ... e ... NON RISPONDERE!!!!



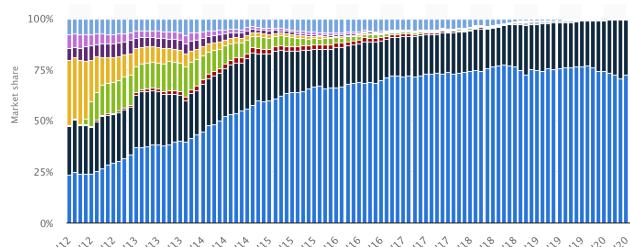
Scrivere un'app che attiva la modalità silenziosa durante le lezioni!

1



attivare il microfono solo quando serve intervenire

- Prof. Roberto De Prisco
  - studio: 4° piano, studio 58
    - numerazione Dip. di Informatica
  - robdep@unisa.it
- Orario lezioni
  - Martedì 11:00-13:00 Teams
  - Mercoledì 09:00-11:00 Teams
- Ricevimento
  - Mercoledì 11:00-13:00
  - Giovedì 16:00-17:00



● Android ● iOS ● KaiOS ● Windows Phone ● Series 40 (Nokia)\* ● Symbian OS\* ● Samsung  
● BlackBerry OS ● Unknown / Other



USA 3 M/E JUN 2020	
Android	56.9%
BlackBerry	0.0%
iOS	43.1%
Windows	0.0%
Other	0.0%
Compare	↔

JAPAN 3 M/E JUN 2020	
Android	50.2%
BlackBerry	0.0%
iOS	49.7%
Windows	0.0%
Other	0.1%
Compare	↔

ITALY 3 M/E JUN 2020	
Android	83.2%
BlackBerry	0.1%
iOS	16.3%
Windows	0.3%
Other	0.0%
Compare	↔

CHINA 3 M/E JUN 2020	
Android	83.1%
BlackBerry	0.0%
iOS	16.9%
Windows	0.1%
Other	0.0%
Compare	↔



## Argomenti

Android Mobile Programming – Prof. R. De Prisco

Slide

13

Università di Salerno - Autunno 2020

- Piattaforma Android
- Android Studio
- Emulatore

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

## Risorse didattiche

Android Mobile Programming – Prof. R. De Prisco

Slide

14

- Lezioni!!!!
- Google
  - <http://developer.android.com>
  - <http://developer.android.com/guide>
  - <http://developer.android.com/training>
- Books
  - BigNerd Ranch (in inglese)
    - Ultima edizione usa Kotlin al posto di Java
    - <http://www.bignerdranch.com/>
- Cursera
  - ottimo video corso (in inglese)
  - <https://www.coursera.org/course/android>
- Googling!



## Sito e piattaforma S3

- Sito
  - Informazioni
  - Annunci
  - Codice app

<http://www.di-srv.unisa.it/~robdep/MP/>

(Android) Mobile Programming  
alias Mobile Computing, ex Basi di Dati e Sistemi Informativi su Rete (BDSIR)  
Prof. Roberto De Prisco Dip. di Informatica, Università di Salerno - a.a. 2018-2019

Homepage      Homepage

Benvenuto nella homepage del corso di Mobile Computing per l'anno accademico 2018-2019.

Codice      Orario Lezioni

Riferimenti      • Lunedì 15:00-17:00 aula F8

Prerequisiti      • Giovedì 15:00-17:00 aula F8

Esami      Annunci

Risultati      • [08/09/2018] Il corso inizia lunedì 24/09/2017.

App contest

Corsa (Android) Mobile Programming - Prof. Roberto De Prisco, Dip. di Informatica, Università di Salerno - a.a. 2018-2019

- S3
  - Esami
  - Risultati

Slide

15

Università di Salerno - Autunno 2020

## Contest

Android Mobile Programming – Prof. R. De Prisco

Slide

16

Università di Salerno - Autunno 2020

- 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](http://www.etuitus.it)
- commissione: docente, eTuitus

## Contest – albo d'oro

Android Mobile Programming – Prof. R. De Prisco

Slide

17

Università di Salerno - Autunno 2020

- 2017: Andrea Sarto
  - Hidle
- 2016: Fabricio Madaio
  - Bizarre 3D Pong
- 2015: Raffaele D'Arco
  - ShootTheFruits
- 2014: Carmelo Orlando
  - AIR Orari Linee

Google Play

Hidle

Hidle Team - Social

Supervisore dei giochi

Aggiungi alla lista desideri

Install

Bizarre 3D Pong (Unreleased)

Media Router - Arcade

PED 3

Si tratta di un'app non ancora pubblicata, pertanto potrebbe essere instabile.

Non ha alcun dispositivo.

Aggiungi alla lista desideri

Install

ShootTheFruits

The Asynchronous Arcade

PED 2

Aggiungi alla lista desideri

Install

AIR Orari Linee

Carrello Orlando - Mappe e navigatori

PED 3

Aggiungi alla lista desideri

Install

## Esame

Android Mobile Programming – Prof. R. De Prisco

Slide

18

Università di Salerno - Autunno 2020

- L'app sviluppata per il contest non sostituisce la prova di laboratorio
  - ma verrà comunque valutata per l'esame

## • Esame

– Scritto

– Laboratorio

• vengono ammessi gli studenti che superano lo scritto

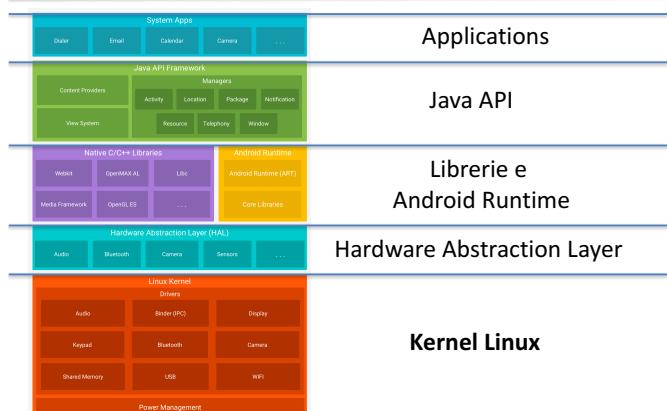
– Orale a discrezione del docente

- Fine lezione (o nella pausa)
- Orario di ricevimento studenti
  - Mercoledì 11:00-13:00
  - Giovedì 16:00-17:00
- Email
  - Risposta non garantita, dipende dalla domanda!
  - Condizione necessaria: il messaggio deve contenere il nome del mittente
  - Appuntamento
- **NON** telefonare



## 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>



## Kernel Linux

Android Mobile Programming – Prof. R. De Prisco

Slide  
25

Università di Salerno - Autunno 2020

- Fornisce i servizi di base del sistema operativo

- filesystem
  - gestione della memoria e dei processi
  - gestione dell’interfaccia di rete
  - drivers per le periferiche
- Servizi specifici per Android
  - gestione della batteria
  - gestione della memoria condivisa
  - low memory killer
  - interprocess communication e altre



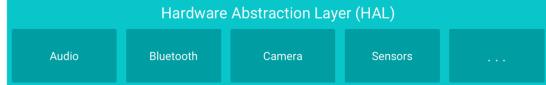
## Hardware Abstraction Layer

Android Mobile Programming – Prof. R. De Prisco

Slide  
26

Università di Salerno - Autunno 2020

### Hardware Abstraction Layer (HAL)



- HAL: Hardware Abstraction Layer

- Interfacce standard per esporre le capacità hardware ai servizi di livello superiore
  - Audio
  - Bluetooth
  - Fotocamera
  - Sensori
  - ...

## Java

Android Mobile Programming – Prof. R. De Prisco

Slide  
27

Università di Salerno - Autunno 2020

- App Android sono scritte in Java
- La libreria fornisce molte classi pronte per l’uso:
  - classi di base: `java.*`, `javax.*`
  - classi per le app: `android.*`
  - Internet/web services: `org.*`
  - Unit testing: `junit.*`

## File dex e ART

Android Mobile Programming – Prof. R. De Prisco

Slide  
28

Università di Salerno - Autunno 2020

- App:

- Scritte in Java
- 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 tutte i file di dati necessari e viene installato sulla target device
- ART Virtual Machine esegue il file Dex

## Android runtime: ART e Dalvik VM

Android Mobile Programming – Prof. R. De Prisco

Slide  
29

Università di Salerno - Autunno 2020



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

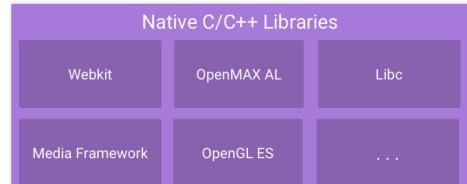
## Librerie native

Android Mobile Programming – Prof. R. De Prisco

Slide  
30

Università di Salerno - Autunno 2020

### Native C/C++ Libraries



- Molte componenti Android necessitano di librerie native

- Webkit
- Libc
- openGL ES
- ...

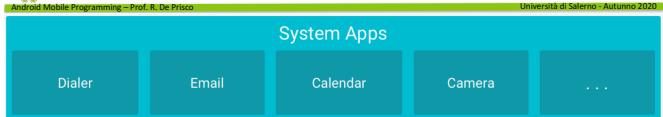
## Application framework



Slide  
31  
Università di Salerno - Autunno 2020

- Tutte le funzionalità del s.o. Android vengono esposte tramite un API
  - View System
    - Per accedere a dati di altre app, per es. ai contatti della rubrica
  - Content Providers
  - Package manager
    - gestisce l'installazione delle app sul dispositivo mobile
  - Activity Manager
    - gestisce il ciclo di vita delle applicazioni
    - permette di passare da un'applicazione all'altra

## Applicazioni (app)



Slide  
32  
Università di Salerno - Autunno 2020

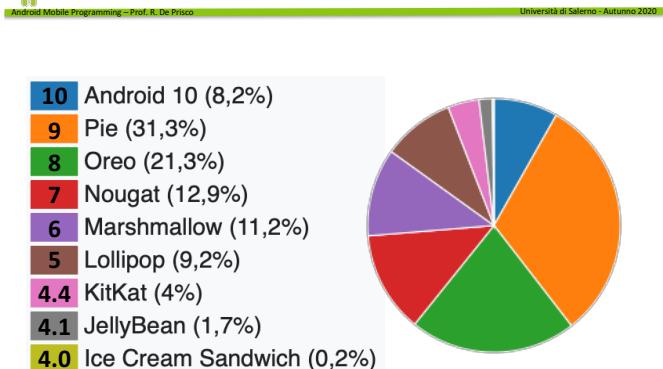
- Applicazioni già presenti nel sistema
  - Home: Main screen
  - Contatti
  - Telefono
  - Browser
  - Email client
  - Media player
  - ... altre
- Ovviamente ... si possono scrivere nuove app!

## Versioni

NAME	VERSIONE	KERNEL LINUX	DATA RILASIO	LIVELLO API
<a href="#">Android 1.0</a>	1.0	2.1	23/09/2008	1
<a href="#">Petit Four</a>	1.1	2.6	09/02/2009	2
<a href="#">Cupcake</a>	1.5	2.6.27	27/04/2009	3
<a href="#">Donut</a>	1.6	2.6.29	15/09/2009	4
<a href="#">Eclair</a>	2.0 – 2.1	2.6.29	26/10/2009	5-7
<a href="#">Froyo</a>	2.2 – 2.2.3	2.6.32	20/05/2010	8
<a href="#">Gingerbread</a>	2.3 – 2.3.7	2.6.35	06/12/2010	9-10
<a href="#">Honeycomb</a>	3.0 – 3.2.6	2.6.36	22/02/2011	11-13
<a href="#">Ice Cream Sandwich</a>	4.0 – 4.0.4	3.0.1	18/11/2011	14-15
<a href="#">Jelly Bean</a>	4.1 – 4.3.1	3.0.31 a 3.4.39	09/07/2012	16-18
<a href="#">KitKat</a>	4.4 – 4.4.4	3.10	31/10/2013	19-20
<a href="#">Lollipop</a>	5.0 – 5.1.1	3.16	12/11/2014	21-22
<a href="#">Marshmallow</a>	6.0 – 6.0.1	3.18	05/10/2015	23
<a href="#">Nougat</a>	7.0 – 7.1.2	4.4	22/08/2016	24 – 25
<a href="#">Oreo</a>	8.0 – 8.1	4.10	21/08/2017	26 – 27
<a href="#">Pie</a>	9	4.4.107, 4.9.84, 4.14.42	06/08/2018	28
<a href="#">Q</a>	10	4.15.0	03/09/2019	29
<a href="#">R</a>	11	5.0.3	?	30

Slide  
33  
Università di Salerno - Autunno 2020

## Versioni



Source: wikipedia Agosto 2020

# ANDROID Mobile Programming

Android Mobile Programming – Prof. R. De Prisco

Università di Salerno - Autunno 2020

## Android Developer Tools

35

## Android Studio

Android Mobile Programming – Prof. R. De Prisco

Università di Salerno - Autunno 2020

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

Slide  
36  
Università di Salerno - Autunno 2020

## Istallazione ADT

The screenshot shows the "Installing the SDK" section of the Android Developer website. It features a large "DOWNLOAD ANDROID STUDIO" button. Below it, there's a brief description: "è necessario avere Java 8 installato e, per l'emulatore, Intel HAXM".

Slide  
37

## Android Studio 3.5

Android Mobile Programming – Prof. R. De Prisco Università di Salerno - Autunno 2020

- Richiede:
  - Java 8
  - Intel Hardware Accelerated Execution Manager HAXM (per l'emulatore)
- Offre
  - Piattaforma Android
  - Android SDK Tools
    - sviluppo
    - debug
  - Gradle
  - Emulatore Android

Slide  
38

## Modalità sviluppatore

Android Mobile Programming – Prof. R. De Prisco Università di Salerno - Autunno 2020

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



Slide  
39

## Listeners

Android Mobile Programming – Prof. R. De Prisco Università di Salerno - Autunno 2020

- 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

Slide  
40

## Prima app: CiaoMondo

Android Mobile Programming – Prof. R. De Prisco Università di Salerno - Autunno 2020

- Visualizza un saluto al mondo!



Icona: New→Image Asset Creator



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

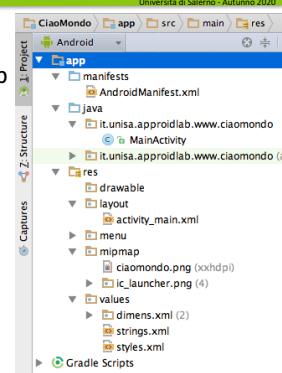
Per mostrare l'icona nell'ActionBar: `getActionBar().setDisplayHomeAsUpEnabled(true);`

Slide  
41

## CiaoMondo

Android Mobile Programming – Prof. R. De Prisco Università di Salerno - Autunno 2020

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



Slide  
42

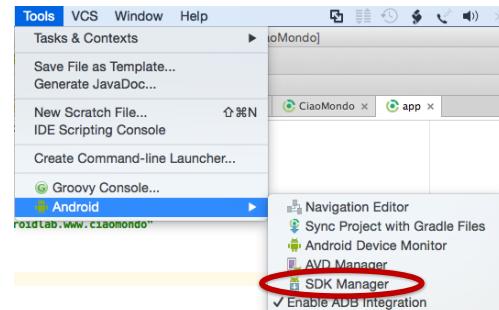
- Gradle
  - build system
- Informazioni
  - dipendenze da altro codice

```
dependencies {
    compile fileTree(dir: "libs", include: ["*.jar"])
    compile 'com.android.support:appcompat-v7:22.1.1'
}
```

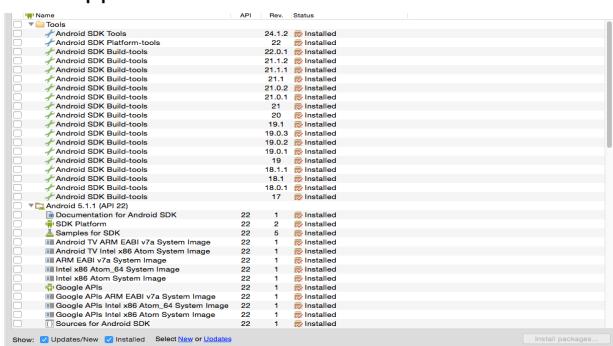
– altre informazioni su come compilare

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

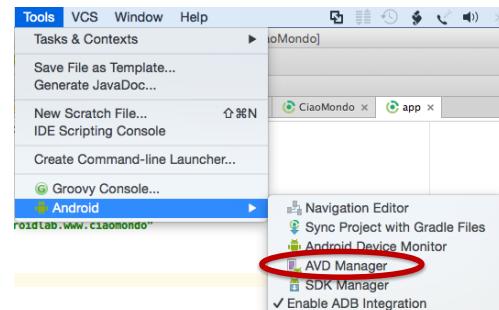
- Android SDK Manager



- Occorre installare le versioni per le quali si vuole sviluppare



- Android Virtual Device Manager



- 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

- telnet

```
abc ~ 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: 'auth <auth_token>' to authenticate
Android Console: you can find your <auth_token> in
'/Users/abc/.emulator_console_auth_token'
OK
auth f82+Iwngfr0xSg
Android Console: type 'help' for a list of commands
OK
sms send 3331234567 "Ciao!"
OK
network speed edge
OK
help
Android console commands:
  help[hi]?
  help-verbose
  ping
  event
  geo
  gsm
  camera
  crash
  crash-on-exit
  kill
  network
  power
```



## Emulatore

Android Mobile Programming – Prof. R. De Prisco

Slide  
49

Università di Salerno - Autunno 2020

- Comunicazione fra due emulatori

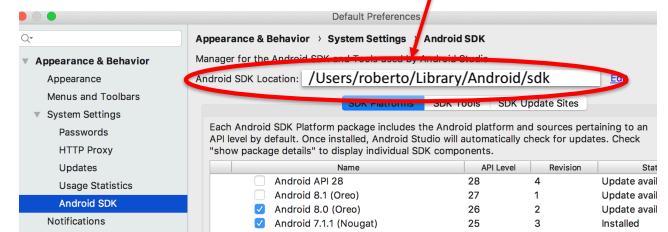


## adb

Android Mobile Programming – Prof. R. De Prisco

Slide  
50

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



## adb e porta telnet

Android Mobile Programming – Prof. R. De Prisco

Slide  
51

Università di Salerno - Autunno 2020

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

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

## Copiare un apk installato

Android Mobile Programming – Prof. R. De Prisco

Slide  
52

- Controllare (individuare) il nome del “package”  
`adb shell pm list packages`
- Individuare il path di installazione  
`adb shell pm path <package-name>`
- Copiare l’apk  
`adb pull <full/path/of/the.apk>`



# ANDROID Mobile Programming

Android Mobile Programming – Prof. R. De Prisco

Università di Salerno - Autunno 2020

## Layouts

53

## Layouts

Android Mobile Programming – Prof. R. De Prisco

Slide  
54

- 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

## Layouts

Android Mobile Programming – Prof. R. De Prisco

Slide  
55

Università di Salerno - Autunno 2020

- 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



## Layout – elementi base (esempi)

Android Mobile Programming – Prof. R. De Prisco

Slide  
56

Università di Salerno - Autunno 2020

- TextView

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

Ciao, sono un TextView

- 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

## Layout - ViewGroup

Android Mobile Programming – Prof. R. De Prisco

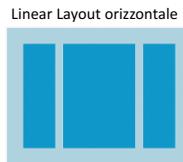
Slide  
57

Università di Salerno - Autunno 2020

- Gruppi di altri elementi
  - sia di base che altri gruppi

- LinearLayout

- orizzontali e verticali



- Relative Layout

- Grid Layout (griglia)

- Frame (contenitore)

## Layout - XML

Android Mobile Programming – Prof. R. De Prisco

Slide  
58

Università di Salerno - Autunno 2020

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="fill_parent"  
              android:layout_height="fill_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

## Layout - XML - attributi

Android Mobile Programming – Prof. R. De Prisco

Slide  
59

Università di Salerno - Autunno 2020

- 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

## Layout - XML - attributi

Android Mobile Programming – Prof. R. De Prisco

Slide  
60

Università di Salerno - Autunno 2020

- 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

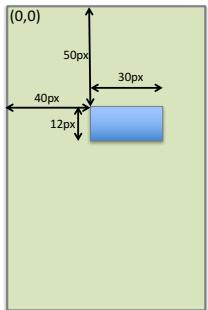
## Layout - XML - attributi

- 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`

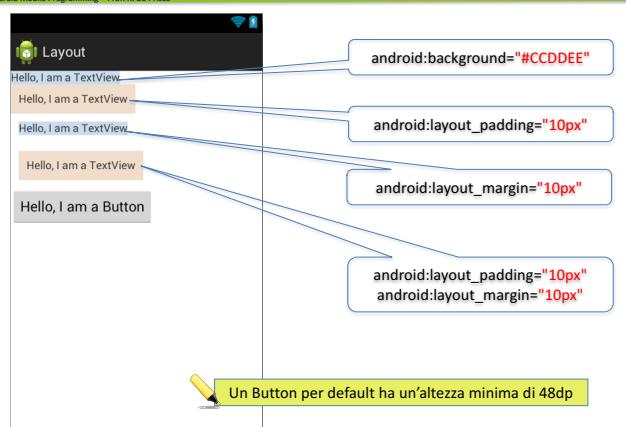
Slide  
61  
Università di Salerno - Autunno 2020

## 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_height="12px"`
  - `android:layout_width="match_parent"`
  - `android:layout_height="wrap_content"`



## Layout – padding e margini



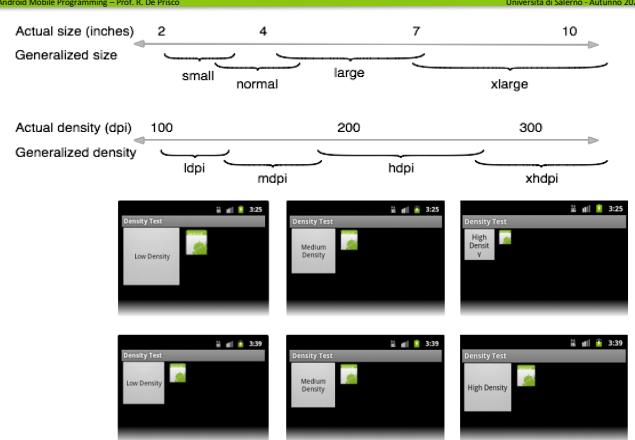
Slide  
63  
Università di Salerno - Autunno 2020

## Misure pixel: px vs. dp

- Screen size
  - grandezza reale del display (es. 4")
- Screen density
  - Quanti pixel ci sono nell'unità di area
    - raggruppatisi 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

Slide  
64  
Università di Salerno - Autunno 2020

## Misure Pixel



Slide  
65  
Università di Salerno - Autunno 2020

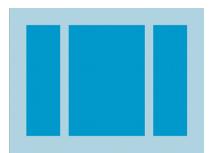
## Alternative per i "drawable"

- 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)

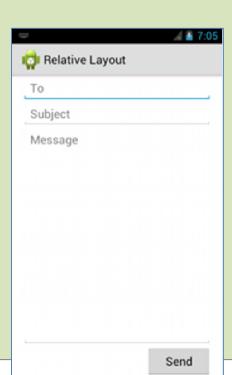
Slide  
66  
Università di Salerno - Autunno 2020

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

- 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



```
<?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="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/subject" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/send" />
</LinearLayout>
```



- 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

- 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"

```
<?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>
```



## Layout - ConstraintLayout

Android Mobile Programming – Prof. R. De Prisco

Slide  
73

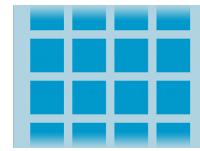
- simile al RelativeLayout
  - permette di specificare la posizione attraverso "vincoli"
  - utile per evitare una gerarchia di layouts innestati troppo profonda
    - che necessita di più tempo per essere disegnata
- Comoda quando si lavora con l'editor grafico
- Si inseriscono dei "vincoli" che legano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

## Layouts – Grid View

Android Mobile Programming – Prof. R. De Prisco

Slide  
74

- Visualizza un insieme di elementi
  - numero totale variabile
  - visibile solo una parte
  - scroll
- Adapter
  - fornisce gli elementi da inserire nel Grid View
    - lo vedremo in seguito

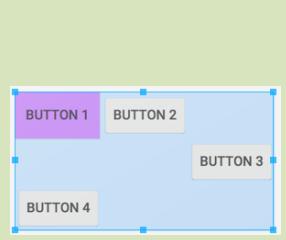


## Layouts – Grid Layout

Android Mobile Programming – Prof. R. De Prisco

Slide  
75

```
<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="2"  
    android:text="Button 3" />  
  
<Button  
    android:id="@+id/button4"  
    android:text="Button 4" />  
  
</GridLayout>
```

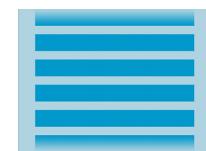


## Layouts – List View

Android Mobile Programming – Prof. R. De Prisco

Slide  
76

- 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

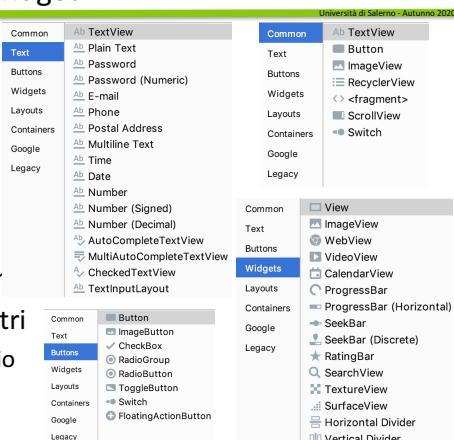


## Layouts - widget

Android Mobile Programming – Prof. R. De Prisco

Slide  
77

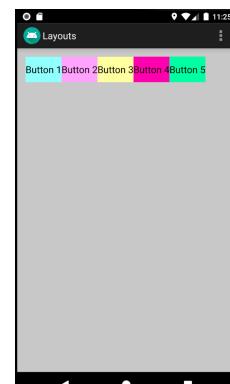
- TextView
- Button
- EditText
- ImageView
- CheckBox
- RadioButton
- ... e molti altri
  - es. Orologio



## App Layouts

Android Mobile Programming – Prof. R. De Prisco

Slide  
78



## 02-Layouts

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

## Widgets

Android Mobile Programming – Prof. R. De Prisco



### 03-Widgets

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

Slide  
79  
Università di Salerno - Autunno 2020



# ANDROID Mobile Programming

Android Mobile Programming – Prof. R. De Prisco

Università di Salerno – Autunno 2020

## La nostra prima (vera) app

80

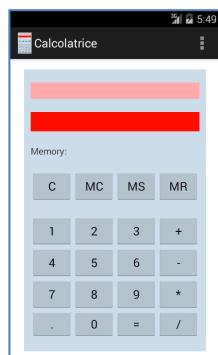
## Prima app

Android Mobile Programming – Prof. R. De Prisco

Slide  
81  
Università di Salerno - Autunno 2020

### 04-Calcolatrice

- Una sola “Activity”
- Pulsanti
  - Listeners
- TextView

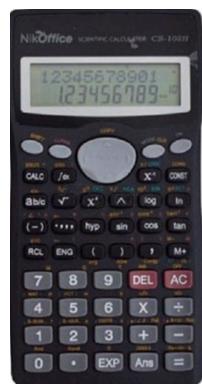


## Esercizio

Android Mobile Programming – Prof. R. De Prisco

Università di Salerno – Autunno 2020

- Calcolatrice più realistica



Slide  
82  
Università di Salerno - Autunno 2020



# ANDROID Mobile Programming

Android Mobile Programming – Prof. R. De Prisco

Università di Salerno - Autunno 2020

## Android Studio Debugger

83



## Copiare (rinominare)

Android Mobile Programming – Prof. R. De Prisco

Università di Salerno – Autunno 2020

- 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

Slide  
84  
Università di Salerno - Autunno 2020

## Android Studio Debugger

Android Mobile Programming – Prof. R. De Prisco

Slide  
85  
Università di Salerno - Autunno 2020

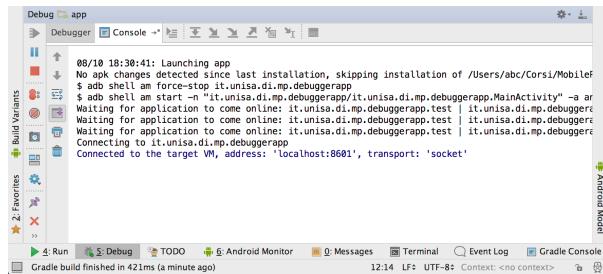
- Permette
  - Eseguire un'app in modalità “debug”
    - Sia con l'emulatore che con una device reale
  - Inserire dei “breakpoints”
    - Esaminare il valore delle variabili
  - Esecuzione “passo-passo”
- LLDB
  - Se c'è codice C/C++ viene usato anche il debugger LLDB
  - Useremo solo il debugger Java

## Android Studio Debugger

Android Mobile Programming – Prof. R. De Prisco

Slide  
87  
Università di Salerno - Autunno 2020

- Si apre la finestra di debug
  - CMD-5



## Android Studio Debugger

Android Mobile Programming – Prof. R. De Prisco

Slide  
86  
Università di Salerno - Autunno 2020

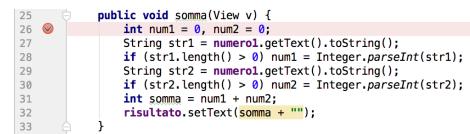
- Build variant
  - Deve essere “debuggable”
    - Build>Select Build Variant
- Per lanciare l'app in modalità debug
  - Run>Debug (CTRL-ALT-D)
  - icona
- Selezionare la device

## Breakpoints

Android Mobile Programming – Prof. R. De Prisco

Slide  
88  
Università di Salerno - Autunno 2020

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



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

## Comandi debugger

Android Mobile Programming – Prof. R. De Prisco

Slide  
89  
Università di Salerno - Autunno 2020

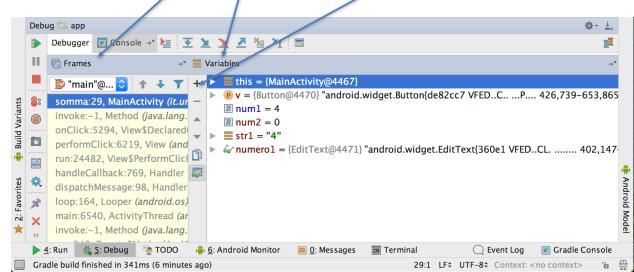
- Valuta una espressione
- Esecuzione di una singola istruzione
- Entra all'interno di una funzione
- Esci dalla funzione
- Riprendi l'esecuzione fino al prossimo breakpoint

## Debugger

Android Mobile Programming – Prof. R. De Prisco

Slide  
90  
Università di Salerno - Autunno 2020

- Ispezione
  - Frames, stack delle chiamate
  - Variables, valori memoria
  - Watches



## Messaggi di log: Log.X

Android Mobile Programming – Prof. R. De Prisco  
Università di Salerno – Autunno 2020

Slide  
91

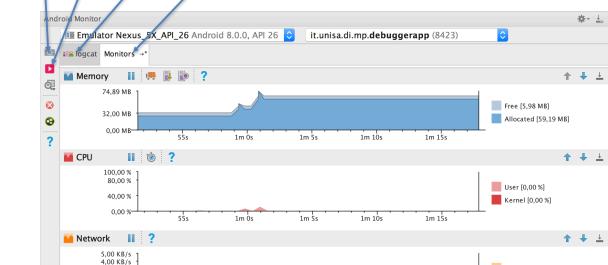
- private static final String TAG = "MyActivity";
- Log.**x**(TAG, "messaggio");
- **x** può essere
  - d: debug
  - e: errore
  - i: info
  - w: warning
  - v: verbose

## Monitor

Android Mobile Programming – Prof. R. De Prisco  
Università di Salerno – Autunno 2020

Slide  
92

- **Logcat**
  - Messaggi di log
- **Monitor**
  - Uso memoria, CPU, rete e GPU



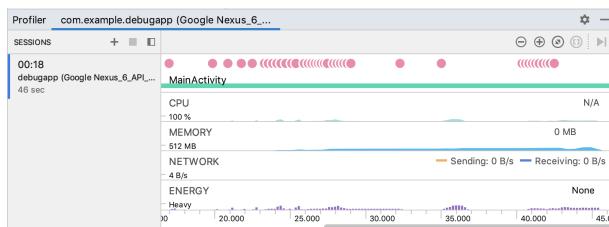
## Profiler

Android Mobile Programming – Prof. R. De Prisco

Slide  
93

Università di Salerno – Autunno 2020

- Android Studio 4
- Profiler



## Logcat

Android Mobile Programming – Prof. R. De Prisco

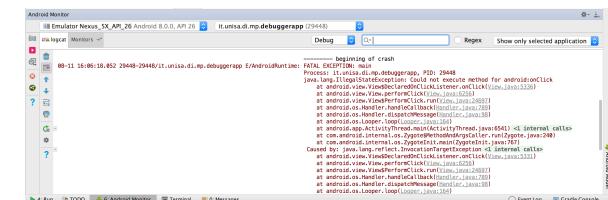
Slide  
94

Università di Salerno – Autunno 2020

- Crash: stack trace

– Fa vedere

- la linea di codice che ha causato l'errore
- lo stack delle chiamate



## DebugApp

Android Mobile Programming – Prof. R. De Prisco

Slide  
95

Università di Salerno – Autunno 2020

## 05-DebugApp

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



## ListView

## ListView

Android Mobile Programming – Prof. R. De Prisco

Slide  
97

Università di Salerno - Autunno 2020

- Widget specifico per le liste
  - divide l'area disponibile in varie posizioni
    - 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

## ListView

Android Mobile Programming – Prof. R. De Prisco

Slide  
98

Università di Salerno - Autunno 2020

- Lista semplice

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

- 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"/>  
</LinearLayout>
```

## ListView

Android Mobile Programming – Prof. R. De Prisco

Slide  
99

Università di Salerno - Autunno 2020

- 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 = (ListView) findViewById(R.id.mylistview);
- Associare l'adapter al widget
  - listView.setAdapter(arrayAdapter);

## ListView

Android Mobile Programming – Prof. R. De Prisco

Slide  
100

Università di Salerno - Autunno 2020

- 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  
        ...  
        ...  
    };
```

## ListView

Android Mobile Programming – Prof. R. De Prisco

Slide  
101

Università di Salerno - Autunno 2020

- 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

## ListView

Android Mobile Programming – Prof. R. De Prisco

Slide  
102

Università di Salerno - Autunno 2020

- 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 elemento
  - problema: non sappiamo più in quale posizione dell'array siamo!!!
  - trucchetto per risolverlo: setTag, getTag



## Ciclo di vita

### Ciclo di vita delle attività

- Ogni “Activity” ha un ciclo di vita

#### Attività non esiste

- onCreate()
- onStart()
- onResume()

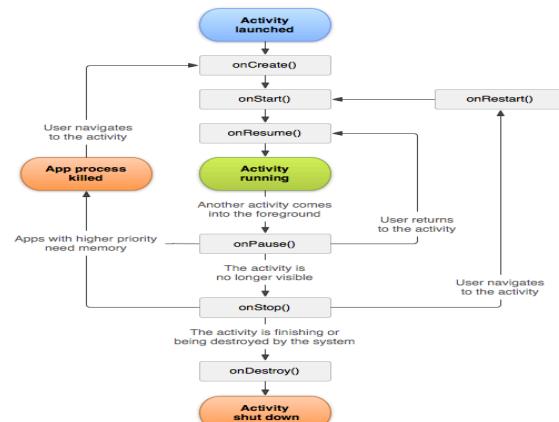
#### Attività in esecuzione

- onPause()
- onStop()
- onDestroy()

#### Attività non esiste

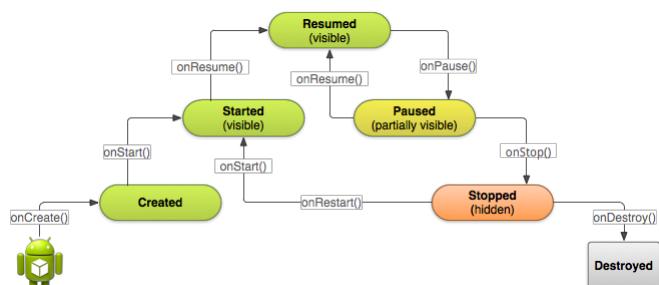
- Eseguiti secondo un determinato schema

### Ciclo di vita delle attività



### Ciclo di vita delle attività

### 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()

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

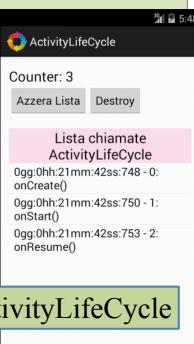
- Si salva lo stato in onSaveInstanceState()

```
@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");
    }
}
```

```
<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>
```



### • App Calcolatrice

- Cosa succede se ruotiamo il dispositivo?

 Correggere l'errore nell'app Calcolatrice



# Ciclo di vita e Cambi di configurazione

- 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, keysoft)
- .... altre: configuration qualifier names (Table 2)

## Cambio di configurazione

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

Slide  
115  
Università di Salerno - Autunno 2020

## onConfigurationChanged()

- 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()

Slide  
117  
Università di Salerno - Autunno 2020

## Main activity

Android Mobile Programming – Prof. R. De Prisco  
Università di Salerno - Autunno 2020

```
@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();  
    }  
}
```

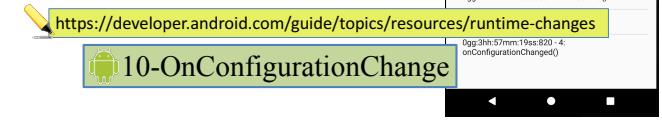
- Valori dell'oggetto Configuration
  - interi specificati nella classe Configuration

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

Android Mobile Programming – Prof. R. De Prisco  
Università di Salerno - Autunno 2020

```
<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."*



## 10-OnConfigurationChange

Android Mobile Programming – Prof. R. De Prisco  
Università di Salerno - Autunno 2020



# ANDROID Mobile Programming

## Backstack

## Backstack

Android Mobile Programming – Prof. R. De Prisco

Slide  
120  
Università di Salerno - Autunno 2020

- 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.

- 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

- 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

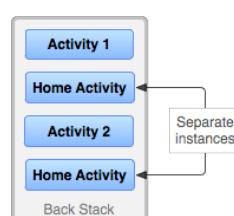


- Continuando a premere Back si ritorna all’Home screen

- Un task con le sue attività può essere spostato in “background”
  - quando l’utente inizia un nuovo task oppure preme il pulsante Home
  - Le attività vengono messe in stato di stop, ma il loro backstack rimane intatto



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



## Classe Intent

Android Mobile Programming – Prof. R. De Prisco

Slide  
127

Università di Salerno - Autunno 2020

- 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

## Intent

Android Mobile Programming – Prof. R. De Prisco

Slide  
128

Università di Salerno - Autunno 2020

- 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 Resource Identifier: <schema>:<parte specifica>
    - "http://www.di.unisa.it/"
    - "mailto:robdep@unisa.it"
    - "geo:0,0?via+Posidonia+Salerno+Italy"
    - "tel:+39112223456"
    - "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

## Intent

Android Mobile Programming – Prof. R. De Prisco

Slide  
129

Università di Salerno - Autunno 2020

- 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).

## Intent

Android Mobile Programming – Prof. R. De Prisco

Slide  
130

Università di Salerno - Autunno 2020

- 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

## Intent

Android Mobile Programming – Prof. R. De Prisco

Slide  
131

Università di Salerno - Autunno 2020

```
Intent i;  
i = newIntent(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

## Intent

Android Mobile Programming – Prof. R. De Prisco

Slide  
132

Università di Salerno - Autunno 2020

```
@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

## Intent Extras

Android Mobile Programming – Prof. R. De Prisco

Slide  
133

Università di Salerno - Autunno 2020

- 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.getStringExtra("CONTATORE");
stringa = intent.getStringExtra("STRINGA");
Bundle b = intent.getExtras();
```

## Intent Flags

Android Mobile Programming – Prof. R. De Prisco

Slide  
134

Università di Salerno - Autunno 2020

- 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

Android Mobile Programming – Prof. R. De Prisco

Slide  
135

Università di Salerno - Autunno 2020

- 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

Android Mobile Programming – Prof. R. De Prisco

Slide  
136

Università di Salerno - Autunno 2020

- 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



12-VisualizzaMappa

## Intent

Android Mobile Programming – Prof. R. De Prisco

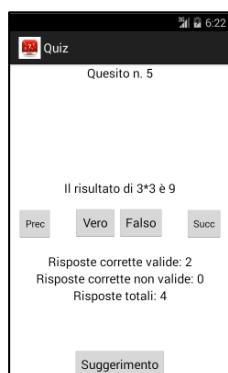
Slide  
137

Università di Salerno - Autunno 2020

Slide  
138

Università di Salerno - Autunno 2020

## 13-Quiz



## Quiz

Android Mobile Programming – Prof. R. De Prisco

Slide  
138

Università di Salerno - Autunno 2020

- 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 2020

Slide  
140

## Permessi

Android Mobile Programming – Prof. R. De Prisco

Università di Salerno – Autunno 2020

- Android protegge risorse e dati con un meccanismo di permessi di accesso
- 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)

139

## Permessi

Android Mobile Programming – Prof. R. De Prisco

Slide  
141

Università di Salerno – Autunno 2020

- Vengono rappresentati da stringhe
- Ogni app deve dichiarare nel manifesto i "permessi" che intende utilizzare

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

- L'utente deve accettare i permessi al momento dell'installazione

## Permessi

Android Mobile Programming – Prof. R. De Prisco

Slide  
142

Università di Salerno – Autunno 2020

- 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 installa l'app (API < 23)
  - a runtime (API >= 23)

## Permessi normali (API 23, 6.0)

Android Mobile Programming – Prof. R. De Prisco

Slide  
143

Università di Salerno – Autunno 2020

- [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](#)

## Permessi pericolosi (API 23, 6.0)

Android Mobile Programming – Prof. R. De Prisco

Slide  
144

Università di Salerno – Autunno 2020

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

## Permessi pericolosi (API 23, 6.0)

Android Mobile Programming – Prof. R. De Prisco

Slide  
145  
Università di Salerno - Autunno 2020

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

# ANDROID Mobile Programming

Android Mobile Programming – Prof. R. De Prisco

Università di Salerno - Autunno 2020

## Threads

- 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

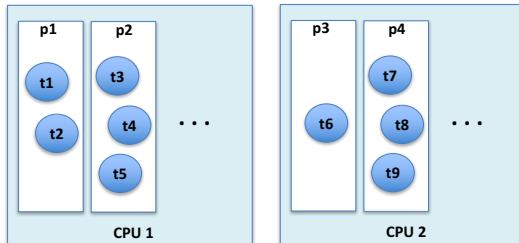
146

## Threads

Android Mobile Programming – Prof. R. De Prisco

Slide  
147  
Università di Salerno - Autunno 2020

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



## Java Threads

Android Mobile Programming – Prof. R. De Prisco

Slide  
148  
Università di Salerno - Autunno 2020

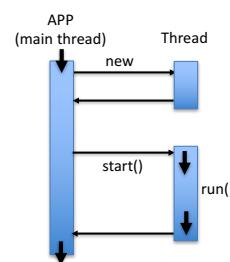
- Oggetti `java.lang.Thread`
- Implementano l'interfaccia `Runnable`
  - devono aver il metodo `void run()`
- Metodi che useremo
  - `void start()`
  - `void sleep(long time)`
  - `void wait()`
    - aspetta che un altro oggetto chiami `notify()` su questo oggetto
  - `void notify()`

## Threads

Android Mobile Programming – Prof. R. De Prisco

Slide  
149  
Università di Salerno - Autunno 2020

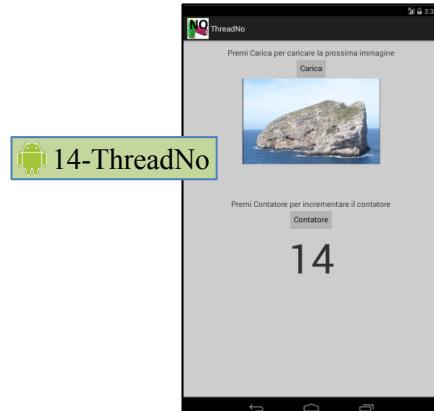
- Per usare un thread:
  - Creare un oggetto Thread
  - Chiamare il metodo `start()` del thread
    - che chiamerà il metodo `run()`



## Threads

Android Mobile Programming – Prof. R. De Prisco

Slide  
150  
Università di Salerno - Autunno 2020



## Threads

Android Mobile Programming – Prof. R. De Prisco



Slide  
151  
Università di Salerno - Autunno 2020

15-ThreadSi

## Threads

Android Mobile Programming – Prof. R. De Prisco

Slide  
152  
Università di Salerno - Autunno 2020

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

## Async task

Android Mobile Programming – Prof. R. De Prisco

- 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)

Slide  
153  
Università di Salerno - Autunno 2020

## AsyncTask

Android Mobile Programming – Prof. R. De Prisco

Slide  
154  
Università di Salerno - Autunno 2020

- 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

## AsyncTask.execute()

Android Mobile Programming – Prof. R. De Prisco

- 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

Slide  
155  
Università di Salerno - Autunno 2020

## AsyncTask

Android Mobile Programming – Prof. R. De Prisco

Slide  
156  
Università di Salerno - Autunno 2020



16-ThreadAsyncTask



# ANDROID Mobile Programming

Android Mobile Programming – Prof. R. De Prisco

Università di Salerno – Autunno 2020

Slide  
158

## Fragments

Android Mobile Programming – Prof. R. De Prisco

Università di Salerno – Autunno 2020

# Fragments

157

## Fragments

Android Mobile Programming – Prof. R. De Prisco

Slide  
159

Università di Salerno – Autunno 2020

Slide  
160

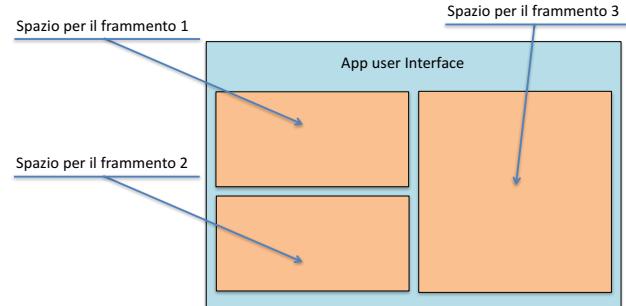
- Un frammento è sempre “ospitato” da un’activity
- Un frammento è una sorta di sub-activity
  - ha il suo ciclo di vita
  - che è strettamente legato a quello dell’activity
    - es. se l’activity è in pausa (stato “paused” del ciclo di vita) lo sono anche tutti i suoi frammenti
    - se l’activity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

## Fragments

Android Mobile Programming – Prof. R. De Prisco

Università di Salerno – Autunno 2020

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



## Fragments

Android Mobile Programming – Prof. R. De Prisco

Slide  
161

Università di Salerno – Autunno 2020

Slide  
162

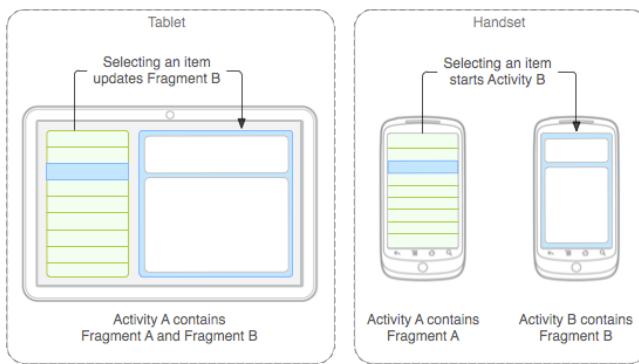
- Filosofia di progettazione
  - Interfacce utente dinamiche
  - in particolare per adattarsi sia a schermi grandi che a schermi piccoli
- Esempio tipico
  - App che gestisce un elenco di elementi
    - es. titoli di articoli di un giornale
  - Ogni elemento può essere cliccato per essere esaminato
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

## Fragments

Android Mobile Programming – Prof. R. De Prisco

Università di Salerno – Autunno 2020

- 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



## 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 perdonano)

## 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

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

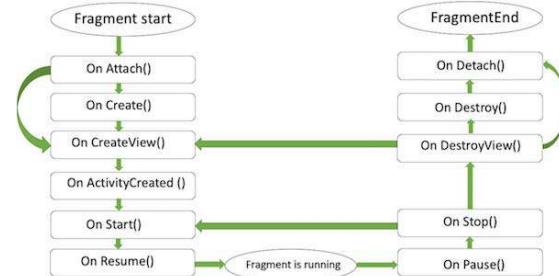
#### Returns

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.

## Creare frammenti

- Istanziare un oggetto Fragment

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



## Creare frammenti

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

```
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;
    }
}
```

## 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>
```

## Creare frammenti

Android Mobile Programming – Prof. R. De Prisco

Slide  
169

Università di Salerno - Autunno 2020

- Oppure dinamicamente a runtime

```
FragmentManager fm= getFragmentManager();
FragmentTransaction ft = fragmentManager.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

## Gestire i frammenti

Android Mobile Programming – Prof. R. De Prisco

Slide  
170

Università di Salerno - Autunno 2020

- Usiamo il FragmentManager

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

- Iniziamo una transazione

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

- Effettuiamo le operazioni

- inserire un frammento (già vista)
- rimuovere un frammento
- sostituire un frammento

- Commit

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

## Gestire i frammenti

Android Mobile Programming – Prof. R. De Prisco

Slide  
171

Università di Salerno - Autunno 2020

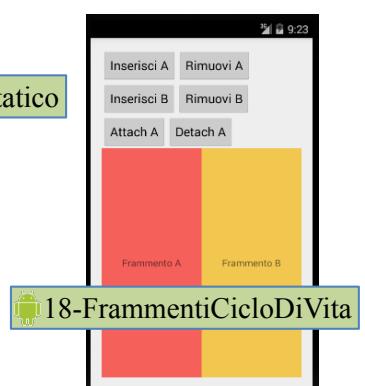
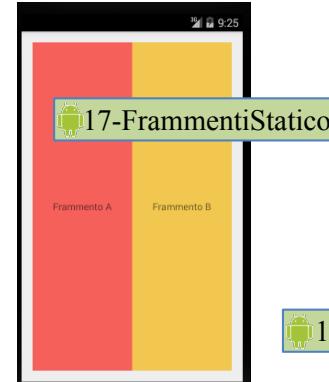
- addToBackStack()
- per inserire i cambiamenti nel backstack
- Il backstack considera solo le activity
  - dobbiamo gestire manualmente i frammenti
- Se non chiamiamo addToBackStack
  - quando premiamo back “salteremo” i cambiamenti fatti con i frammenti
  - non è quello che l'utente si aspetta

## Esempi

Android Mobile Programming – Prof. R. De Prisco

Slide  
172

Università di Salerno - Autunno 2020



## Comunicare con l'activity

Android Mobile Programming – Prof. R. De Prisco

Slide  
173

Università di Salerno - Autunno 2020

- 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);
    }
    ...
}
```

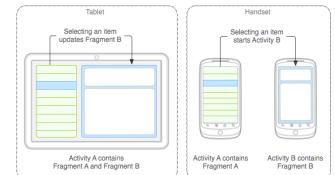
## Comunicare con l'activity

Android Mobile Programming – Prof. R. De Prisco

Slide  
174

Università di Salerno - Autunno 2020

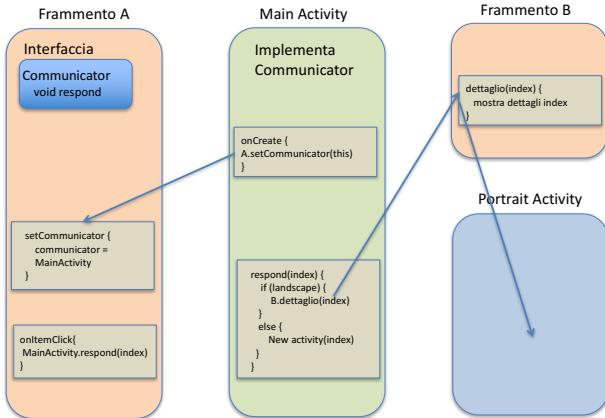
- App
  - lista
  - dettagli elementi
- Frammento lista
  - deve comunicare l'elemento selezionato
- Frammento dettagli
  - deve ricevere l'informazione
- Evitare la comunicazione diretta fra i frammenti
  - diminuisce la riusabilità



## Comunicare con l'activity

Android Mobile Programming – Prof. R. De Prisco  
Università di Salerno - Autunno 2020

Slide  
175



## Frammenti e backstack

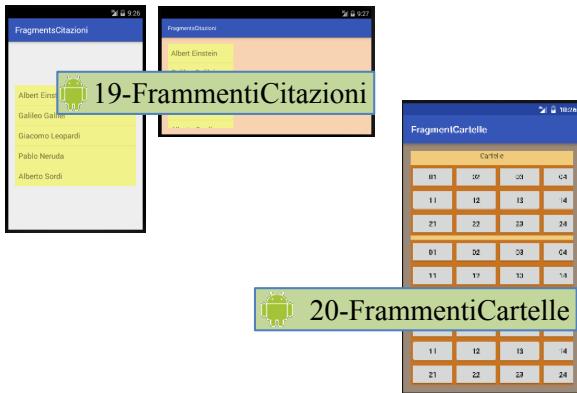
Android Mobile Programming – Prof. R. De Prisco  
Università di Salerno - Autunno 2020

- 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

## Esempi

Android Mobile Programming – Prof. R. De Prisco  
Università di Salerno - Autunno 2020

Slide  
177



## Frammenti

Android Mobile Programming – Prof. R. De Prisco  
Università di Salerno - Autunno 2020

- 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()

# ANDROID Mobile Programming

Android Mobile Programming – Prof. R. De Prisco  
Università di Salerno - Autunno 2020

## Networking

179

## Networking

Android Mobile Programming – Prof. R. De Prisco  
Università di Salerno - Autunno 2020

- Comunicazione via rete
- Socket
  - Java.net
- HTTP
  - Classe HttpURLConnection
- Data formats
  - JSON, XML

## Networking

Android Mobile Programming – Prof. R. De Prisco

Slide  
181

Università di Salerno - Autunno 2020

- 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

## Networking

Android Mobile Programming – Prof. R. De Prisco

Slide  
182

- 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)

## Networking

Android Mobile Programming – Prof. R. De Prisco

Slide  
183

Università di Salerno - Autunno 2020

- Scrivere nel socket

```
• PrintWriter out =  
    new PrintWriter(  
        new BufferedWriter(  
            new OutputStreamWriter(  
                socket.getOutputStream()),  
            true)); //Autoflush
```

```
• out.println(strToSend);
```

## Networking

Android Mobile Programming – Prof. R. De Prisco

Slide  
184

- Leggere dal socket

```
• BufferedReader in =  
    new BufferedReader(  
        new InputStreamReader(  
            socket.getInputStream()));
```

```
• in.readLine(), in.read(), ...
```

## Localhost

Android Mobile Programming – Prof. R. De Prisco

Slide  
185

Università di Salerno - Autunno 2020

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

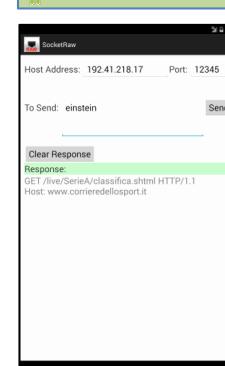
## Networking

Android Mobile Programming – Prof. R. De Prisco

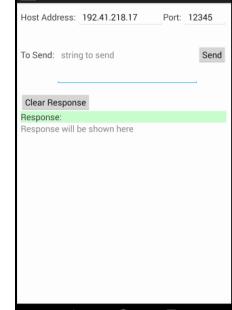
Slide  
186

Università di Salerno - Autunno 2020

### 21-SocketRaw



### 22-SocketRawProgressBar



### 22-SocketRawProgressBar



## URL

Android Mobile Programming – Prof. R. De Prisco

Slide  
187

Università di Salerno - Autunno 2020

- 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



## Networking

Android Mobile Programming – Prof. R. De Prisco

Slide  
188

Università di Salerno - Autunno 2020



23-SocketURL



## Documenti HTML

Android Mobile Programming – Prof. R. De Prisco

Slide  
189

Università di Salerno - Autunno 2020

- Dati in documenti HTML
  - difficile estrarli
- Esistono delle librerie che implementano il parsing di documenti HTML
  - es. JSOUP
- Per utilizzare una libreria
  - procurarsi il file .jar (es. jsoup-1-1.7.3.jar)
  - memorizzarlo nella cartella lib del progetto
  - Aggiungere il file jar nella lista delle librerie
    - Progetto -> Proprietà -> Java Build Path -> Librerie



## Jsoup

Android Mobile Programming – Prof. R. De Prisco

Slide  
190

Università di Salerno - Autunno 2020

- 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]");**



## Jsoup – classifica serie A

Android Mobile Programming – Prof. R. De Prisco

Slide  
191

Università di Salerno - Autunno 2020



24-SocketJSoup



## Errore: Cleartext not permitted

Android Mobile Programming – Prof. R. De Prisco

Slide  
192

Università di Salerno - Autunno 2020

- API > 27

```
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>

**Note:** The guidance in this section applies only to apps that target Android 8.1 (API level 27) or lower. Starting with Android 9 (API level 28), cleartext support is disabled by default.

## Data Storage

193

### Data Storage

Android Mobile Programming – Prof. R. De Prisco

Slide 194  
Università di Salerno – Autunno 2020

- Shared Preferences
  - dati privati, coppie chiave-valore
- File
  - File privati dell'app
  - File pubblici (accessibili da altre app)
- Database SQLite
  - Dati strutturati in database privati

### SharedPreferences

Android Mobile Programming – Prof. R. De Prisco

Slide 195  
Università di Salerno – Autunno 2020

- Classe SharedPreferences
  - permette di salvare e recuperare dati usando coppie di chiave-valore
- 2 metodi della classe Activity
  - getSharedPreferences("filename")
    - quando si vogliono usare più file di "preferenze" (dati)
  - getDefaultSharedPreferences()
    - quando basta un solo file
  - restituiscono un oggetto SharedPreferences



Attenzione a non usare getPreferences (senza "Shared"), che serve per preferenze non condivise con altre activity dell'app.

### SharedPreferences

Android Mobile Programming – Prof. R. De Prisco

Slide 196  
Università di Salerno – Autunno 2020

- 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();

### File

Android Mobile Programming – Prof. R. De Prisco

Slide 197  
Università di Salerno – Autunno 2020

- 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
- Per creare e scrivere un file
  1. Chiamare openFileOutput(fileName, mode)
    - restituisce un FileOutputStream
  2. Scrivere nel file (write())
  3. Chiudere lo stream (close())

### File

Android Mobile Programming – Prof. R. De Prisco

Slide 198  
Università di Salerno – Autunno 2020

```
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 (file accessibile solo all'app)
  - MODE\_APPEND
  - MODE\_WORLD\_READABLE (leggibile da tutti)
  - MODE\_WORLD\_WRITEABLE (scrivibile da tutti)



## File

Android Mobile Programming – Prof. R. De Prisco

Slide  
199

Università di Salerno - Autunno 2020

- Per leggere un file

1. Chiamare `openFileInput(fileName)`
  - restituisce un `InputStream`
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.



## File

Android Mobile Programming – Prof. R. De Prisco

Slide  
200

Università di Salerno - Autunno 2020

- `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



## File temporanei

Android Mobile Programming – Prof. R. De Prisco

Slide  
201

Università di Salerno - Autunno 2020

- 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)



## File su External Storage

Android Mobile Programming – Prof. R. De Prisco

Slide  
202

Università di Salerno - Autunno 2020

- Android permette l'utilizzo di una memoria esterna
  - tipicamente una SD card
- File nella memoria esterna sono pubblici (world-readable)
- Occorre richiedere il permesso di lettura/scrittura
- La memoria esterna può essere rimossa
  - quindi non si può assumere che i file siano sempre disponibili



## External Storage

Android Mobile Programming – Prof. R. De Prisco

Slide  
203

Università di Salerno - Autunno 2020

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

- Permesso Write include il permesso Read

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

A partire da Android 4.4, per lo spazio privato non c'è bisogno di permessi.



## External Storage

Android Mobile Programming – Prof. R. De Prisco

Slide  
204

Università di Salerno - Autunno 2020

```
/* 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;
}
```



## Condividere file con altre app

Android Mobile Programming – Prof. R. De Prisco

Slide  
205  
Università di Salerno - Autunno 2020

- `getExternalStoragePublicDirectory(type)`
  - type: DIRECTORY\_PICTURES, DIRECTORY\_MUSIC, DIRECTORY\_RINGTONES, ...
- Esempio: metodo che crea una nuova dir per delle foto nella dir pubblica delle immagini

```
public File getAlbumStorageDir(String albumName) {
    // Get the directory for the user's public pictures directory.
    File file = new File(Environment.getExternalStoragePublicDirectory(
        Environment.DIRECTORY_PICTURES), albumName);
    if (!file.mkdirs()) {
        Log.e(LOG_TAG, "Directory not created");
    }
    return file;
}
```



## SQL – Quick tutorial

Android Mobile Programming – Prof. R. De Prisco

Slide  
206  
Università di Salerno - Autunno 2020

- 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

## SQL – Quick tutorial

Android Mobile Programming – Prof. R. De Prisco

Slide  
207  
Università di Salerno - Autunno 2020

- 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

## SQL – Quick tutorial

Android Mobile Programming – Prof. R. De Prisco

Slide  
208  
Università di Salerno - Autunno 2020

- 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)
);
```

## SQL – Quick tutorial

Android Mobile Programming – Prof. R. De Prisco

Slide  
209  
Università di Salerno - Autunno 2020

- 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 );
```

## SQL – Quick tutorial

Android Mobile Programming – Prof. R. De Prisco

Slide  
210  
Università di Salerno - Autunno 2020

- 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

## SQL – Quick tutorial

Android Mobile Programming – Prof. R. De Prisco

Slide  
211

Università di Salerno - Autunno 2020

- Esempi di select

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

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	8500.00
6	Renato	22	Venezia	4500

```
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;
```

## SQL – Quick tutorial

Android Mobile Programming – Prof. R. De Prisco

Slide  
212

- 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

## Database

Android Mobile Programming – Prof. R. De Prisco

Slide  
213

Università di Salerno - Autunno 2020

- Android fornisce supporto per database SQL
  - solo all'interno dell'app
- Per usare un database
  - Creare una sottoclassificazione 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();

## Database

Android Mobile Programming – Prof. R. De Prisco

Slide  
214

Università di Salerno - Autunno 2020

- Sul database si possono applicare comandi standard SQL
- Il database (tavola) viene creato (comando CREATE) nel metodo onCreate() della sottoclassificazione DatabaseOpenHelper
- Nell'app vengono usati:
  - db.insert()
  - db.delete()
  - db.update()

## Database

Android Mobile Programming – Prof. R. De Prisco

Slide  
215

Università di Salerno - Autunno 2020

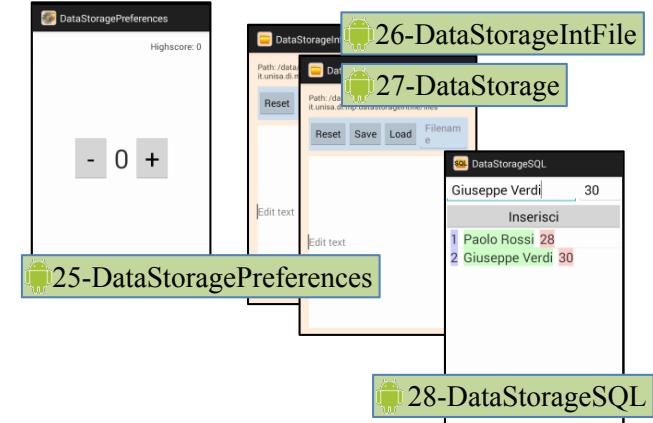
```
public class MyOpenHelper extends SQLiteOpenHelper {  
    private static final int DATABASE_VERSION = 1;  
    private static final String TABLE_NAME = "dictionary";  
    private static final String CREATE_CMD =  
        "CREATE TABLE " + DICTIONARY_TABLE_NAME + " (" +  
        KEY_WORD + " TEXT, " + KEY_DEFINITION + " TEXT);";  
  
    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  
    }  
}
```

## Data Storage

Android Mobile Programming – Prof. R. De Prisco

Slide  
216

Università di Salerno - Autunno 2020





# ANDROID Mobile Programming

Android Mobile Programming – Prof. R. De Prisco

Università di Salerno – Autunno 2020

Slide  
218

Università di Salerno – Autunno 2020

## Grafica

217

### Grafica

Android Mobile Programming – Prof. R. De Prisco

Slide

218

- 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)

### Grafica

Android Mobile Programming – Prof. R. De Prisco

Slide  
219

Università di Salerno – Autunno 2020

Slide  
219

Università di Salerno – Autunno 2020

- L'oggetto Drawable deve essere inserito nell'oggetto View

- direttamente nel file XML
- in modo programmatico
  - View.setImageResource()



29-GraficaSimpleImg

### Animazioni

Android Mobile Programming – Prof. R. De Prisco

Slide  
220

Università di Salerno – Autunno 2020

- Android permette di definire delle animazioni da applicare alle immagini

- Descritte con file XML

- rotazione
- traslazione
- scaling (dimensione)
- trasparenza
- con controllo di vari parametri
  - es., punto di pivot, velocità, etc.

### Animazioni

Android Mobile Programming – Prof. R. De Prisco

Slide  
221

Università di Salerno – Autunno 2020

Slide  
221

Università di Salerno – Autunno 2020

- Class Animation

- permette di
  - leggere le animazioni dai file XML
  - applicarle alle ImageView



30-GraficaAnimazioni

### Custom Views

Android Mobile Programming – Prof. R. De Prisco

Slide  
222

Università di Salerno – Autunno 2020

- Android ha molti widget

- Pulsanti, Liste, ImageView, etc, etc.

- Per esigenze particolare possiamo definire dei widget personalizzati

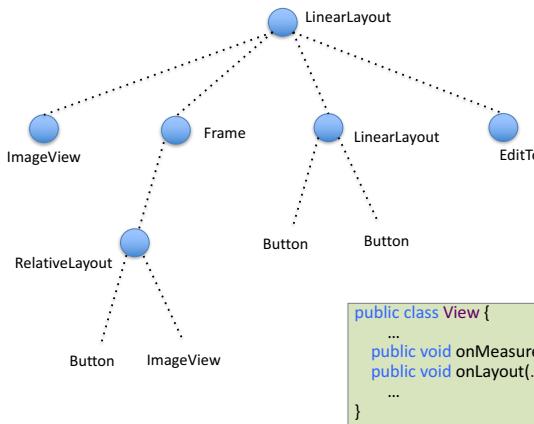
- Permettono un maggiore controllo sulla grafica
  - ovviamente sono più complicati da usare

## Albero delle View

Android Mobile Programming – Prof. R. De Prisco

Slide  
223

Università di Salerno - Autunno 2020



## Meccanismo di layout

Android Mobile Programming – Prof. R. De Prisco

Slide  
224

Università di Salerno - Autunno 2020



- 3 Fasi

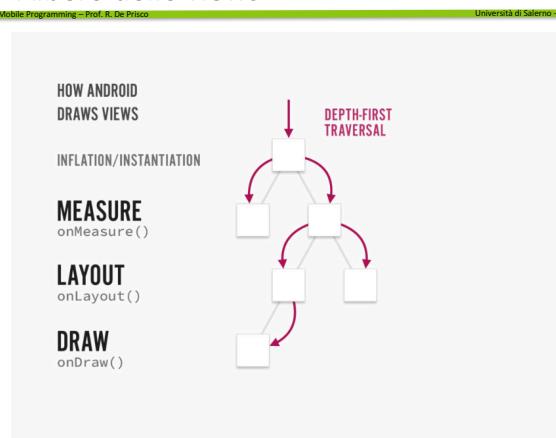
- Misura
- Posizionamento
- Disegno

## Albero delle views

Android Mobile Programming – Prof. R. De Prisco

Slide  
225

Università di Salerno - Autunno 2020



## Fase di misurazione

Android Mobile Programming – Prof. R. De Prisco

Slide  
226

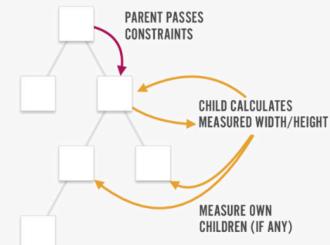
Università di Salerno - Autunno 2020

### HOW ANDROID DRAWS VIEWS

#### MEASURE onMeasure()

#### LAYOUT onLayout()

#### DRAW onDraw()

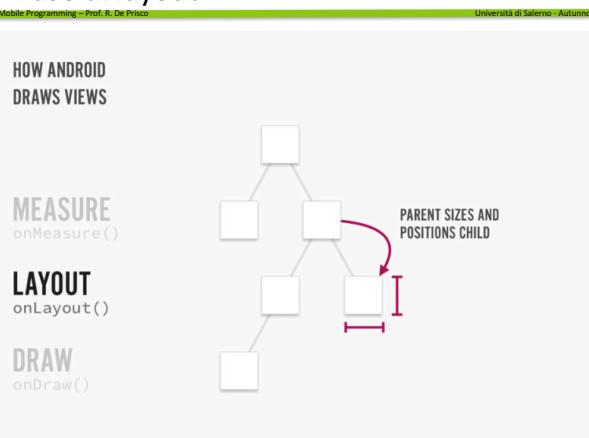


## Fase di layout

Android Mobile Programming – Prof. R. De Prisco

Slide  
227

Università di Salerno - Autunno 2020



## Fase di disegno

Android Mobile Programming – Prof. R. De Prisco

Slide  
228

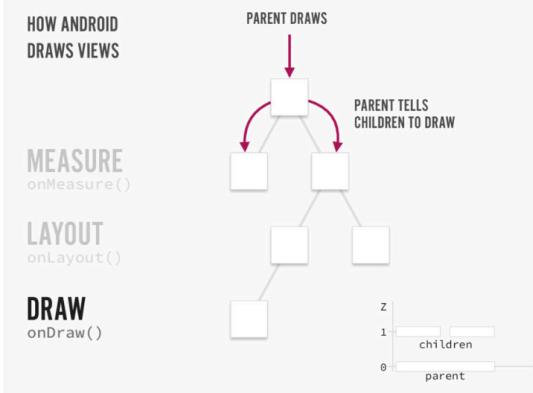
Università di Salerno - Autunno 2020

### HOW ANDROID DRAWS VIEWS

#### MEASURE onMeasure()

#### LAYOUT onLayout()

#### DRAW onDraw()



## Fase di misurazione

Android Mobile Programming – Prof. R. De Prisco

Slide  
229

Università di Salerno - Autunno 2020

- “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 MeasureSpec

## MeasureSpec

Android Mobile Programming – Prof. R. De Prisco

Slide  
230

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

- width e height
  - in pixels
- widthMode e heightMode
  - MeasureSpec.EXACTLY
  - MeasureSpec.AT\_MOST
  - MeasureSpec.UNSPECIFIED

## Fase di misurazione

Android Mobile Programming – Prof. R. De Prisco

Slide  
231

Università di Salerno - Autunno 2020

- Ogni view può esprimere la propria preferenza usando le opzioni della classe ViewGroup.LayoutParams
  - Un numero (di pixel)
  - MATCH\_PARENT
  - WRAP\_CONTENT
- La misurazione avviene nel metodo onMeasure

## Fase di misurazione

Android Mobile Programming – Prof. R. De Prisco

Slide  
232

- Quando il processo di misurazione finisce ogni view deve aver definito
  - measuredWidth
  - measuredHeight
- In alcuni casi c’è un processo di ‘negoziazione’ fra view parent e view figli
  - measure() chiamato più volte

## Fase di layout

Android Mobile Programming – Prof. R. De Prisco

Slide  
233

Università di Salerno - Autunno 2020

- Visita top-down dell’albero delle view
- View parent
  - decide la grandezza e la posizione (in accordo alle misure fatte nella fase precedente)
- onLayout()

## Fase di disegno

Android Mobile Programming – Prof. R. De Prisco

Slide  
234

- Dopo il posizionamento ogni view viene disegnata
  - onDraw()
- Quando c’è un cambiamento
  - invalidate()
    - chama onDraw sulla view
  - requestLayout
    - ripete l’intero processo su tutto l’albero.

## Meccanismo di layout

Android Mobile Programming – Prof. R. De Prisco

Slide  
235

Università di Salerno - Autunno 2020

- “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

## onMeasure()

Android Mobile Programming – Prof. R. De Prisco

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

Università di Salerno - Autunno 2020

- `onMeasure` potrebbe essere chiamata varie volte!
- gli “int” contengono anche dei bit addizionali

## Layout

Android Mobile Programming – Prof. R. De Prisco

Slide  
237

Università di Salerno - Autunno 2020

- Nella fase di Layout le view container comunicano la posizione effettiva ad 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+";");  
        setMeasuredDimension(smw,smh);  
    }  
    ...  
}
```

## Disegnare nel canvas

Android Mobile Programming – Prof. R. De Prisco

Slide  
238

Università di Salerno - Autunno 2020

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

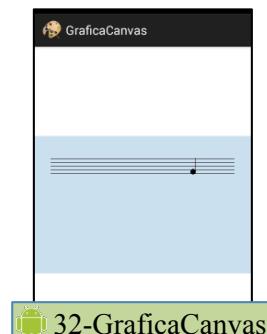
```
public class MyView extends Views{  
    ...  
    @Override  
    public void onDraw (Canvas canvas) {  
        //Codice per disegnare la view  
    }  
    ...  
}
```

## Esempi

Android Mobile Programming – Prof. R. De Prisco

Slide  
239

Università di Salerno - Autunno 2020



## Multitouch

Android Mobile Programming – Prof. R. De Prisco

Slide  
240

Università di Salerno - Autunno 2020

- `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

## Multitouch

Android Mobile Programming – Prof. R. De Prisco

Slide  
241

Università di Salerno - Autunno 2020

- 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

## Multitouch

Android Mobile Programming – Prof. R. De Prisco

Slide  
242

Università di Salerno - Autunno 2020

- 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

## Multitouch

Android Mobile Programming – Prof. R. De Prisco

Slide  
243

Università di Salerno - Autunno 2020

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

## Multitouch

Android Mobile Programming – Prof. R. De Prisco

Slide  
244

Università di Salerno - Autunno 2020

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

## Multitouch

Android Mobile Programming – Prof. R. De Prisco

Slide  
245

Università di Salerno - Autunno 2020

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

## Multitouch

Android Mobile Programming – Prof. R. De Prisco

Slide  
246

Università di Salerno - Autunno 2020

- 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)`

## Multitouch

Android Mobile Programming – Prof. R. De Prisco

Slide  
247

Università di Salerno - Autunno 2020

- 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()

## Multitouch

Android Mobile Programming – Prof. R. De Prisco

Slide  
248

Università di Salerno - Autunno 2020

- 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

## Multitouch

Android Mobile Programming – Prof. R. De Prisco

Slide  
249

Università di Salerno - Autunno 2020

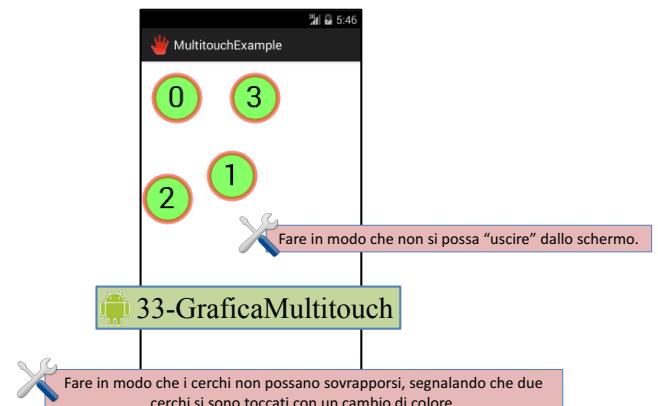
- Spesso si ha la necessità di gestire una combinazione di eventi
- Es. Il doppio "click" equivale a
  - ACTION\_DOWN
  - ACTION\_UP
  - ACTION\_DOWN
  - ACTION\_UP
  - in rapida successione

## Multitouch

Android Mobile Programming – Prof. R. De Prisco

Slide  
250

Università di Salerno - Autunno 2020



## GestureDetector

Android Mobile Programming – Prof. R. De Prisco

Slide  
251

Università di Salerno - Autunno 2020

- Classe GestureDetector
  - permette di riconoscere dei gesti fatti sul display
- Alcuni gesti riconosciuti:
  - pressione semplice
  - doppia pressione (doppio "click")
  - fling (scorrimento)

## GestureDetector

Android Mobile Programming – Prof. R. De Prisco

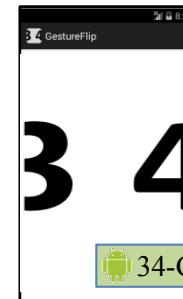
Slide  
252

Università di Salerno - Autunno 2020

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

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

- View Animator
  - classe per un contenitore di tipo FrameLayout
  - animazione cambiamento fra view
- Simple ViewAnimator
  - 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()



34-GraficaGestureFlip



Implementare anche il fling per il decremento

# ANDROID Mobile Programming

## Media player

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

## MediaPlayer

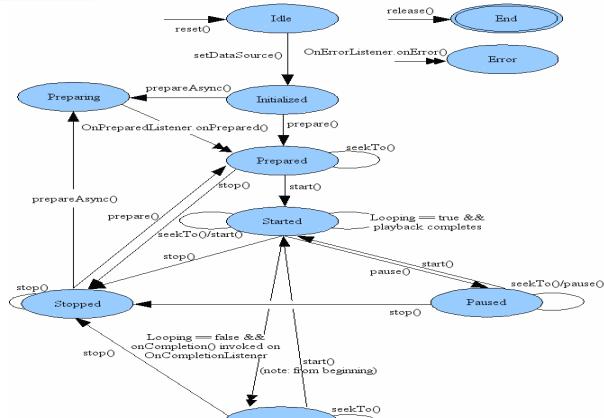
- Play di un file in res/raw

```
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`



- Rilasciare la risorsa

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

- mediaPlayer.start();
- mediaPlayer.pause();
- mediaPlayer.stop();
- Attenzione all'uso asincrono (prepareAsynch)
  - necessario per non appesantire l'app
  - richiede più attenzione
    - Play da fare in/dopo onPrepareListener.onPrepared()

- Audio focus
- Poichè 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



## Sensori

 Inserire altri brani nel MediaPlayer

 Implementare la lista dei brani con un ListView

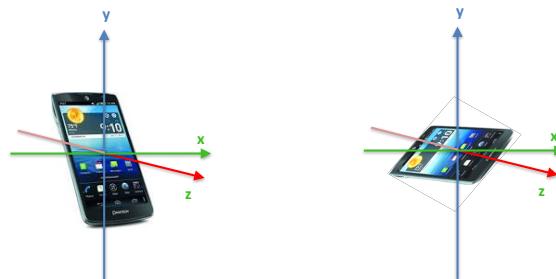
- Molti smartphones, tablet hanno sensori
  - di movimento
    - forze di accelerazione e di rotazione
      - accelerometri, bussola, giroscopio
  - di ambiente
    - temperatura, pressione, umidità
      - termometri, barometri
  - di posizione
    - posizione fisica
      - magnetometro, bussola,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

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

- Pochi 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/s <sup>2</sup> 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

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



- 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
}
```

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

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

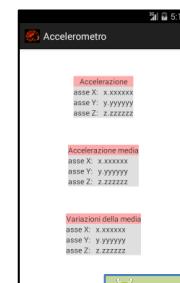
- Velocità di campionamento
  - SENSOR\_DELAY\_NORMAL (0,2sec)
  - SENSOR\_DELAY\_GAME (0,02sec)
  - SENSOR\_DELAY\_UI (0,06sec)
  - SENSOR\_FASTEST (0sec )

- Registrazione e rilascio in onResume e onPause
  - per evitare di consumare la batteria

```
@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];
}
...
```



## Notifiche

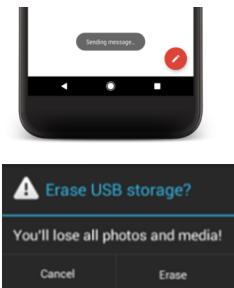
271

### Notifiche

Android Mobile Programming – Prof. R. De Prisco

Università di Salerno – Autunno 2020 | Slide 272

- Notifiche: informazioni all’utente al di fuori dell’interfaccia grafica dell’app
- Toast
  - brevi messaggi temporanei
- Dialog
  - interazione con l’utente
- Notifiche
  - status bar e cassetto delle notifiche



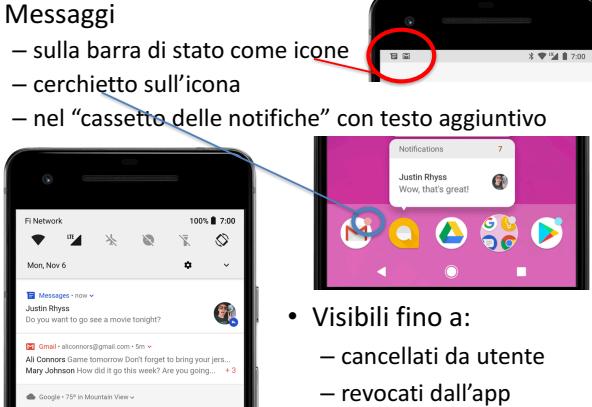
### Notifiche

Android Mobile Programming – Prof. R. De Prisco

Slide 273

Università di Salerno – Autunno 2020

- 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



### Notifiche

Android Mobile Programming – Prof. R. De Prisco

Slide 274

Università di Salerno – Autunno 2020

- Da Android 5.0 (API 21)
  - notifiche a comparsa (heads-up notification)
- Compare anche un messaggio temporaneo
  - simile a un toast
- Se
  - utente in full screen
  - notifica con alta priorità (API ≤ 25)
  - canale di notifica con alta priorità (API ≥ 26)
  - altre



### Notifiche

Android Mobile Programming – Prof. R. De Prisco

Slide 275

Università di Salerno – Autunno 2020

- Da Android 8.0 (API ≥ 26)
  - “canali” di notifica
  - prima in pratica 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)

### Notifiche - compatibilità

Android Mobile Programming – Prof. R. De Prisco

Slide 276

Università di Salerno – Autunno 2020

- Sistema di notifiche continua evoluzione
  - modifiche in
    - Android 4.1 API level 16
    - Android 4.4 API level 19,20
    - Android 5.0, API level 21
    - Android 7.0, API level 24
    - Android 8.0, API level 26
- Conviene usare le classi
  - NotificationCompat e NotificationManagerCompat
  - ottenere la migliore compatibilità possibile

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

- Canale
  - ch = `new NotificationChannel(...)`
  - Proprietà: ch.setLightColor(Color.GREEN);
  - ...
  - 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



37-Notifiche

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

## Alarms

- Gli alarms rimangono attivi fino a quando
  - vengono cancellati
  - la periferica viene spenta
- Esempi di alarms
  - app per gli MMS: usa alarm per controllare periodicamente i messaggi non spediti (retry scheduler)
  - Settings: usa un alarm per rendere la periferica non visibile via Bluetooth dopo un determinato tempo

- Permettono di eseguire intent in funzione di specifici eventi
  - intent lanciati in base al tempo
- 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

- Per usare gli alarm in un'app
  - AlarmManager
- Ottenere un riferimento all'AlarmManager:
  - `getSystemService(Context.ALARM_SERVICE)`
- Creare alarms
  - `void set(int type, long triggerAtTime, PendingIntent i)`
  - `void setRepeating(...), setInexactRepeating(...)`



A partire dall'API level 19 (KitKat) gli alarm non sono "esatti": il SO operativo può modificare i triggerTime per minimizzare wakeups e l'uso della batteria

- **ELAPSED\_REALTIME** (time since boot)
  - lancia il “pending” intent dopo un determinato tempo (ma non fa il wakeup della device)
- **ELAPSED\_REALTIME\_WAKEUP**
  - Se la device non è attiva fa il wakeup e lancia il “pending” intent
- **RTC (UTC time)**
  - lancia il pending event ad un determinato orario
- **RTC\_WAKEUP**
  - come prima ma fa il wakeup se serve



## Content Providers, Broadcast, Services

- 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 estremamente utili
- Nelle prossime slide ci sono dei cenni

## Broadcasts

Android apps can send or receive broadcast messages from the Android system and other Android apps, similar to the [publish-subscribe](#) design pattern. These broadcasts are sent when an event of interest occurs. For example, the Android system sends broadcasts when various system events occur, such as when the system boots up or the device starts charging. Apps can also send custom broadcasts, for example, to notify other apps of something that they might be interested in (for example, some new data has been downloaded).

Apps can register to receive specific broadcasts. When a broadcast is sent, the system automatically routes broadcasts to apps that have subscribed to receive that particular type of broadcast.

Generally speaking, broadcasts can be used as a messaging system across apps and outside of the normal user flow.

## Content Providers

Content providers manage access to a structured set of data. They encapsulate the data, and provide mechanisms for defining data security. Content providers are the standard interface that connects data in one process with code running in another process.

When you want to access data in a content provider, you use the `ContentResolver` object in your application's `Context` to communicate with the provider as a client. The `ContentResolver` object communicates with the provider object, an instance of a class that implements `ContentProvider`. The provider object receives data requests from clients, performs the requested action, and returns the results.

You don't need to develop your own provider if you don't intend to share your data with other applications. However, you do need your own provider to provide custom search suggestions in your own application. You also need your own provider if you want to copy and paste complex data or files from your application to other applications.

Android itself includes content providers that manage data such as audio, video, images, and personal contact information. You can see some of them listed in the reference documentation for the `android.provider` package. With some restrictions, these providers are accessible to any Android application.

## Services

A **Service** is an application component that can perform long-running operations in the background, and it does not provide a user interface. Another application component can start a service, and it continues to run in the background even if the user switches to another application. Additionally, a component can bind to a service to interact with it and even perform interprocess communication (IPC). For example, a service can handle network transactions, play music, perform file I/O, or interact with a content provider, all from the background.

- Parleremo

- della classe `BroadcastReceiver`
- di come un “ricevitore di broadcast” deve essere “registrato”
- dei modi in cui gli eventi possono essere inviati ai ricevitori di broadcast
- di come i ricevitori ricevono una notifica di un evento
- di come i ricevitori gestiscono la notifica

- serve per ricevere e “reagire” ad eventi
  - eventi sono rappresentati da Intent
- un “ricevitore” deve “registrarsi” dichiarando gli eventi ai quali è interessato
  - es. esiste un BroadcastReceiver che ha il compito di spedire messaggi MMS
- Quando un’altra componente crea un MMS invia un evento (Intent)
  - l’Intent viene mandato in broadcast al sistema
  - Il ricevitore lo intercetta e spedisce il messaggio

- Riassumendo

1. Il “ricevitore” si “registra” usando `registerReceiver()` (disponibile nel `LocalBroadcastManager` o nel `Context`)
2. L’evento viene creato (da qualche altra componente del sistema)
3. Android notifica il ricevitore chiamando `onReceive()`

- Lo si può fare
  - staticamente nel Manifesto dell’app
  - dinamicamente usando `registerReceiver()`

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
...
<application
...
    <receiver
        android:name=".My_Receiver"
        android:exported="false"
        <intent-filter>
            <action android:name="it.unisa.mp.MY_ACTION" />
        </intent-filter>
    </receiver>
</application>
</manifest>
```

- Se la registrazione è statica

- il ricevitore viene registrato durante il Boot del sistema (oppure quando l’app viene installata)

- Se la registrazione è dinamica

- il ricevitore viene registrato quando si chiama
  - `LocalBroadcastManager.registerReceiver()`
    - per i broadcast locali all’app
  - `Context.registerReceiver()`
    - per i broadcast system-wide

- è possibile anche revocare la registrazione usando `unregisterReceiver()`

## Spedizione broadcast

Android Mobile Programming – Prof. R. De Prisco

Slide  
295

Università di Salerno - Autunno 2020

- Per spedire un messaggio si usa il metodo
  - sendBroadcast(Intent i)
  - 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.

## BroadcastReceiver

Android Mobile Programming – Prof. R. De Prisco

Slide  
296

Università di Salerno - Autunno 2020

- Esempi di altri eventi globali.:
  - android.intent.action.AIRPLANE\_MODE
  - android.intent.action.BATTERY\_LOW
  - android.intent.action.DATA\_SMS\_RECEIVED
  - android.intent.action.DATE\_CHANGED
  - android.intent.action.DEVICE\_STORAGE\_LOW
  - android.intent.action.TIMEZONE\_CHANGED
  - android.intent.action.TIME\_TICK
  - android.intent.action.USER\_PRESENT
  - android.intent.action.WALLPAPER\_CHANGED
  - ...

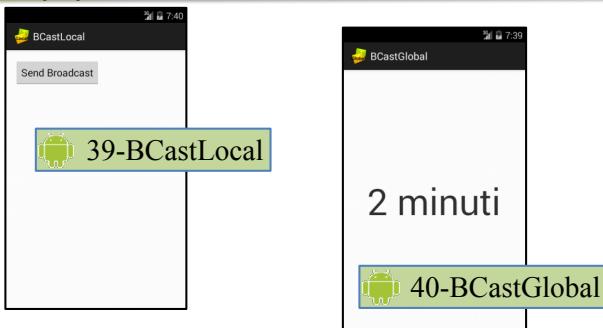
 Lista completa: sdk/platforms/android-19/data/broadcast\_actions.txt

## Esempi

Android Mobile Programming – Prof. R. De Prisco

Slide  
297

Università di Salerno - Autunno 2020



- ACTION\_TIME\_CLICK
  - Intent inviato ogni minuto



## ContentProvider

Android Mobile Programming – Prof. R. De Prisco

Slide  
298

Università di Salerno - Autunno 2020

- Rappresentano Contenitori Dati
  - progettati per condividere le informazioni fra le applicazioni
- Per accedere ad un ContentProvider si utilizza un ContentResolver
  - interfaccia simile a quella di un database
  - comandi SQL-like
    - QUERY, INSERT, UPDATE, DELETE, etc
  - in più, notifiche su cambiamenti dei dati

## ContentProvider

Android Mobile Programming – Prof. R. De Prisco

Slide  
299

Università di Salerno - Autunno 2020

- Per usare il resolver occorre recuperare un suo riferimento chiamando
  - Context.getContentResolver()
- ContentProviders standard
  - 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 parole digitate)
  - ... molti altri

## ContentProvider

Android Mobile Programming – Prof. R. De Prisco

Slide  
300

Università di Salerno - Autunno 2020

- I dati contenuti in un provider sono memorizzati in tabelle
- Gli utenti 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

## ContentProvider

Android Mobile Programming – Prof. R. De Prisco

Slide  
301

Università di Salerno - Autunno 2020

- 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

## ContentProvider

Android Mobile Programming – Prof. R. De Prisco

Slide  
302

Università di Salerno - Autunno 2020

- Per ottenere i dati usiamo una query ed un Cursor
- ContentResolver.query()

```
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

## ContentProvider

Android Mobile Programming – Prof. R. De Prisco

Slide  
303

Università di Salerno - Autunno 2020

- Esempio: leggere la rubrica
  - <uses-permission  
 android:name="android.permission.READ\_CONTACTS"
  - />



## Services

Android Mobile Programming – Prof. R. De Prisco

Slide  
304

Università di Salerno - Autunno 2020

- Servono a eseguire operazioni complesse che possono richiedere molto tempo
  - es. scaricare un file da Internet, sincronizzare informazioni locali con un server
- Vedremo
  - La classe Services
  - come usare dei Services esistenti
  - come definire dei nuovi Services

## Services

Android Mobile Programming – Prof. R. De Prisco

Slide  
305

Università di Salerno - Autunno 2020

- Services non interagiscono con l’utente
  - non c’è una UI
- Servono per eseguire delle operazioni in background
- L’app interagisce con il servizio
- La prima cosa da fare è far partire il Service
  - Context.startService(Intent i)

## Services

Android Mobile Programming – Prof. R. De Prisco

Slide  
306

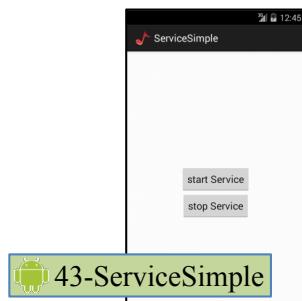
Università di Salerno - Autunno 2020

- 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
- per default, il Service gira nel main thread dell’app che lo ha fatto partire
  - in alcuni casi deve essere esplicitamente fatto girare su un thread separato

- Le componenti che vogliono interagire con un Service devono effettuare un “bind”
  - Context.bindService(Intent service, ServiceConnection conn, int flags);
- Il binding permette di
  - inviare richieste
  - ricevere riposte
- Se al momento delle richiesta di bind il Service non è ancora attivo
  - viene fatto partire
  - rimane attivo fino a quando c’è almeno un client connesso

- 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>
```



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

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



# Fine del corso!