Prof. Roberto De Prisco

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno – Autunno 2022

ANDROID
Mobile Programming

1

1

per favore …

Slide 2

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2022

… o almeno  … e … NON RISPONDERE!!!!

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

2

Info corso

Slide 3

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2022

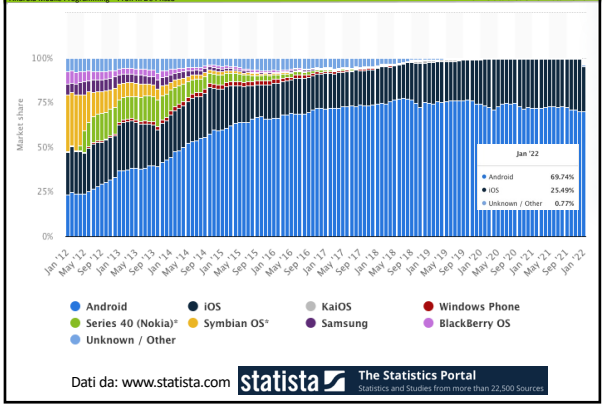
- Prof. Roberto De Prisco
  - studio: 4° piano, studio 58
    - numerazione Dip. di Informatica
  - robdep@unisa.it
- Orario lezioni
  - Martedì 14:00-16:00 F8
  - Giovedì 14:00-16:00 F8
- Ricevimento
  - Martedì 16:00-17:00
  - Giovedì 11:00-13:00

3

Vendite per OS (worldwide)

Slide 4

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2022



Dati da: www.statista.com statista The Statistics Portal Statistics and Studies from more than 22,500 Sources

4

Vendite per OS 2022

Slide 5

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2022

Dati da: https://www.kantarworldpanel.com/global/smartphone-os-market-share/

5

Telefoni Android prodotti da

Slide 6

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2022

SAMSUNG HUAWEI Google

xiaomi LG NOKIA

oppo ALCATEL honor

acer htc

SONY

e altri …

6

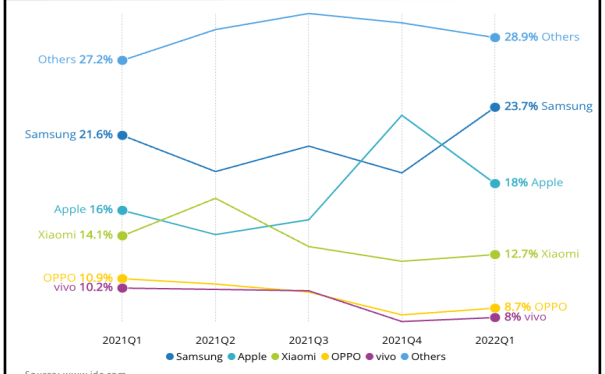
Vendite per marca

Company	2022Q1 Shipment Volumes	2022Q1 Market Share	2021Q1 Shipment Volumes	2021Q1 Market Share	Year-Over-Year Change
Samsung	74.5	23.7%	74.5	21.6%	0.0%
Apple	56.5	18.0%	55.3	16.0%	2.2%
Xiaomi	39.9	12.7%	48.6	14.1%	-17.9%
OPPO	27.4	8.7%	37.5	10.9%	-26.8%
vivo	25.3	8.0%	35.0	10.2%	-27.7%
Others	90.8	28.9%	93.9	27.2%	-3.3%
TOTAL	314.4	100.0%	344.7	100.0%	-8.8%

Source: www.idc.com

7

Vendite per marca

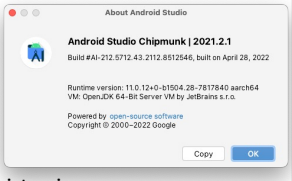


Source: www.idc.com

8

Altre motivazioni

- Ambiente di sviluppo (Android Studio)
  - facile da istallare
  - molti tools
  - Chipmunk, Aprile 2022
- Istallazioni delle app
  - facile
  - non richiede nessuna registrazione
- Moltissime risorse online



9

Cosa insegna il corso

Come scrivere app per Android!!!

10

Strumenti di sviluppo

Vedremo tutto ciò che serve per scrivere app …

11

Argomenti

- Piattaforma Android
- Android Studio
- Emulatore

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

12

Risorse didattiche

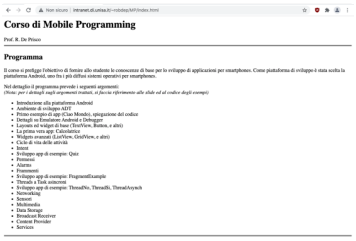
- **Lezioni!!!!**
- Google
  - http://developer.android.com
  - http://developer.android.com/guide
  - http://developer.android.com/training
- Books
  - BigNerd Ranch (in inglese)
    - Ultima edizione usa Kotlin al posto di Java
  - http://www.bignerdranch.com/
- Cursera
  - ottimo video corso (in inglese)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

13

Sito e piattaforma S3

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

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



14

Contest

- La migliore app sviluppata DURANTE il corso verrà premiata con un dispositivo Android!
  - Per partecipare occorre
    - frequentare il corso
    - sviluppare in gruppi di 2 persone max
    - consegnare l'app una settimana prima del contest
    - app di una certa complessità

- Sponsorizzato da eTuitus
  - www.etuitus.it
  - commissione: docente, eTuitus

15

Esame

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

16

Come contattare il docente

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

17

Domande?

18

… si parte

19

ANDROID Mobile Programming

La piattaforma Android

20

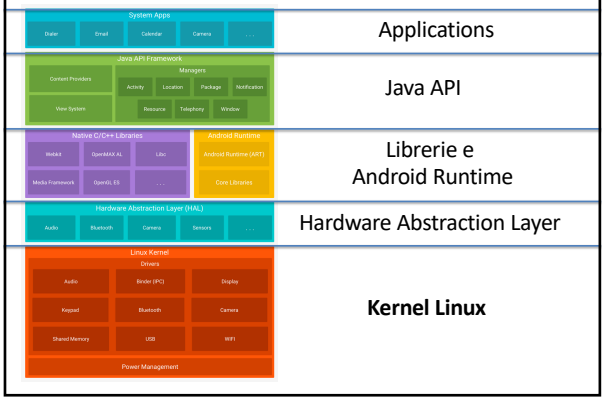
La piattaforma Android

- 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

21

L'architettura Android



Applications

Java API

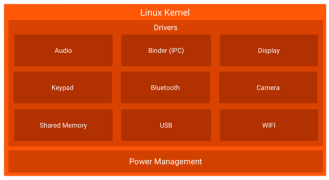
Librerie e
Android Runtime

Hardware Abstraction Layer

Kernel Linux

22

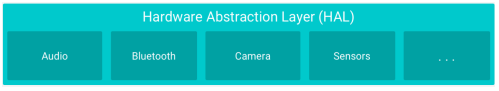
Kernel Linux



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

23

Hardware Abstraction Layer



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

24

Java

- 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.*

25

File dex e ART

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

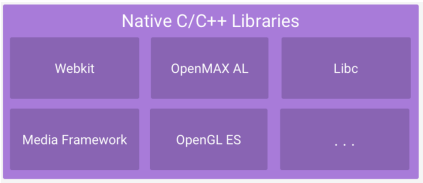
26

Android runtime: ART e Dalvik VM

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

27

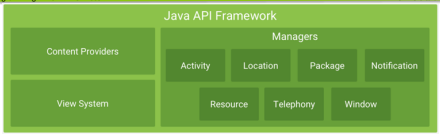
Librerie native



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

28

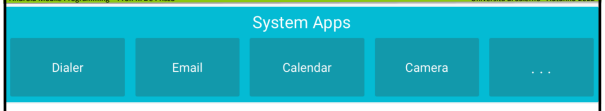
Application framework



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

29

Applicazioni (app)



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

30

Versioni

Slide 31

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2022

NOME	VERSIONE	KERNEL LINUX	DATA RILASCIO	LIVELLO API
Android 1.0	1.0	2.1	23/09/2008	1
Petit Four	1.1	2.6	09/02/2009	2
Cupcake-Donut	1.5/1.6	2.6.27/29	27/04-15/09/2009	3-4
Eclair	2.0 – 2.1	2.6.29	26/10/2009	5-7
Froyo	2.2 – 2.2.3	2.6.32	20/05/2010	8
Gingerbread	2.3 – 2.3.7	2.6.35	06/12/2010	9-10
Honeycomb	3.0 – 3.2.6	2.6.36	22/02/2011	11-13
Ice Cream Sandwich	4.0 – 4.0.4	3.0.1	18/11/2011	14-15
Jelly Bean	4.1 – 4.3.1	3.0.31 a 3.4.39	09/07/2012	16-18
KitKat	4.4 – 4.4.4	3.10	31/10/2013	19-20
Lollipop	5.0 – 5.1.1	3.16	12/11/2014	21-22
Marshmallow	6.0 – 6.0.1	3.18	05/10/2015	23
Nougat	7.0 – 7.1.2	4.4	22/08/2016	24 – 25
Oreo	8.0 – 8.1	4.10	21/08/2017	26 – 27
Pie	9	4.4.107, 4.9.84, 4.14.42	06/08/2018	28
Android 10 (Q)	10	4.15.0	03/09/2019	29
Android 11 (R)	11	5.0.3	08/09/2020	30
Snow Cone	12	?	04/10/2021	31-32
Android 13?	Beta	?	?	33

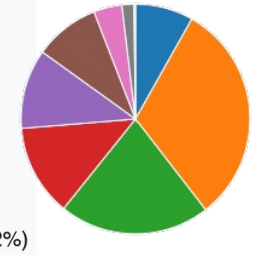
31

Versioni

Slide 32

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2022

- 10 Android 10 (8,2%)
- 9 Pie (31,3%)
- 8 Oreo (21,3%)
- 7 Nougat (12,9%)
- 6 Marshmallow (11,2%)
- 5 Lollipop (9,2%)
- 4.4 KitKat (4%)
- 4.1 JellyBean (1,7%)
- 4.0 Ice Cream Sandwich (0,2%)



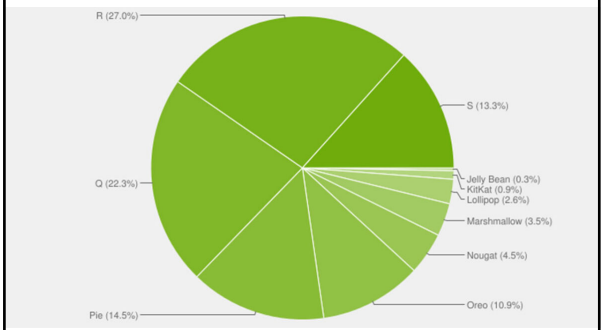
Source: wikipedia Agosto 2020

32

Versioni

Slide 33

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2022



Source: web, 2022

33

ANDROID Mobile Programming

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno – Autunno 2022

Android Developer Tools

34

Android Studio

Slide 35

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2022

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

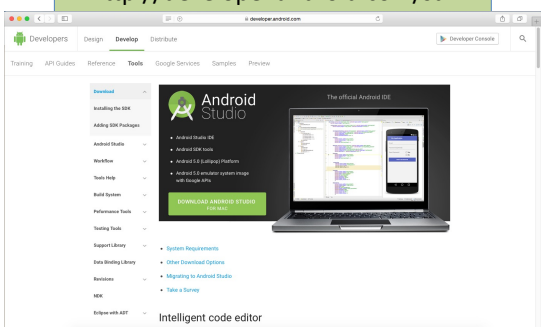
35

Istallazione ADT

Slide 36

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2022

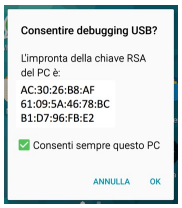
http://developer.android.com/sdk



36

Modalità sviluppatore

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



37

Listeners

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

38

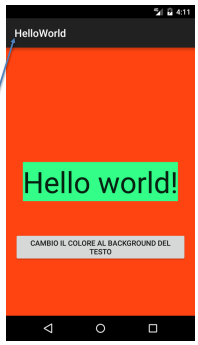
Prima app: CiaoMondo

- Visualizza un saluto al mondo!

Icona: New→Image Asset Creator

01-CiaoMondo

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

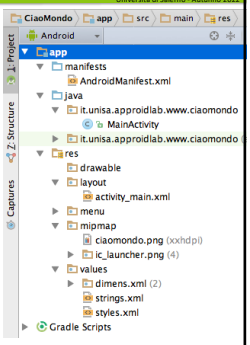


Per mostrare l'icona nell'ActionBar: getActionBar().setDisplayShowHomeEnabled(**true**);

39

CiaoMondo

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



40

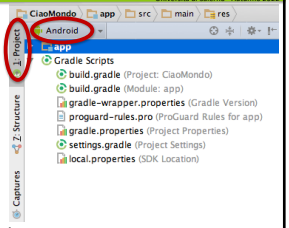
CiaoMondo

- 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

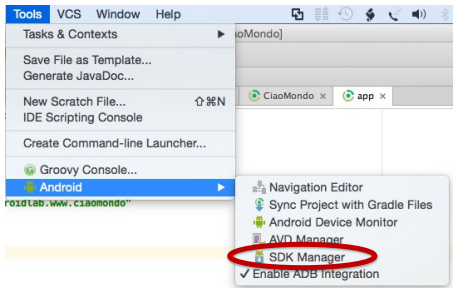
```
buildTypes {
    release {
        minifyEnabled false
        proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
    }
}
```



41

SDK

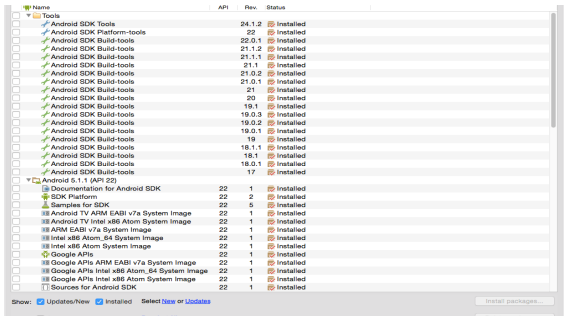
- Android SDK Manager



42

SDK

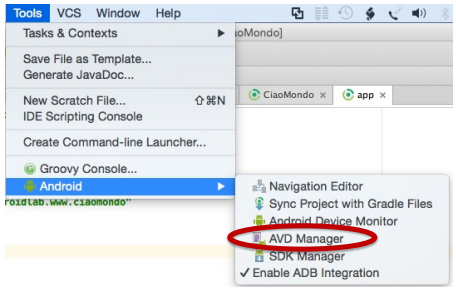
- Occorre istallare le versioni per le quali si vuole sviluppare



43

Emulatore Android

- Android Virtual Device Manager



44

Emulatore vs. real device

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

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

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

45

Emulatore

- telnet

46

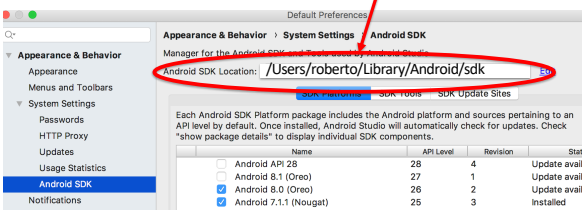
Emulatore

- Comunicazione fra due emulatori

47

adb

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



48

adb e porta telnet


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

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

49

Copiare un apk istallato

- Controllare (individuare) il nome del "package

```
adb shell pm list packages
```

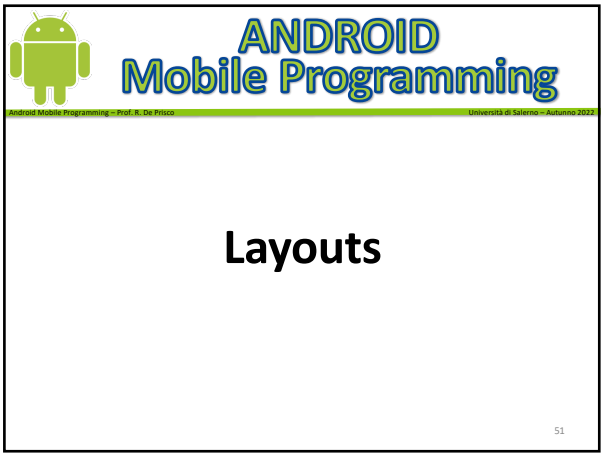
- Individuare il path di istallazione

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

- Copiare l'apk

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

50



51

Layouts

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

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

52

Layouts

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

53

Layout – elementi base (esempi)

- 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" />
```

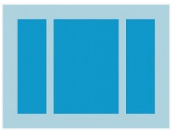
Ciao, sono un Pulsante

54

Layout - ViewGroup

- Gruppi di altri elementi
  - sia di base che altri gruppi
- Linear Layout
  - orizzontali e verticali
- Relative Layout
- Grid Layout (griglia)
- Frame (contenitore)

Linear Layout orizzontale



55

Layout - XML

Un solo elemento "radice" in ogni file XML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="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

56

Layout - XML - attributi

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

57

Layout - XML - attributi

- ID (creazione)

  android:id="@+id/text"

  @: il resto della stringa deve essere interpretato
  - android:layout_width="30px"
    - se non c'è @ il valore è quello letterale

  +: specifica che stiamo creando (aggiungendo) un nuovo identificatore (id) il cui nome è text

- ID (riferimento)

  android:id="@id/text"

  senza il + è un riferimento ad un ID esistente

58

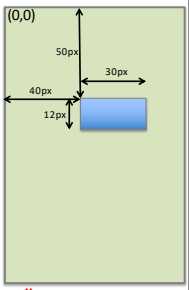
Layout - XML - attributi

- 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

59

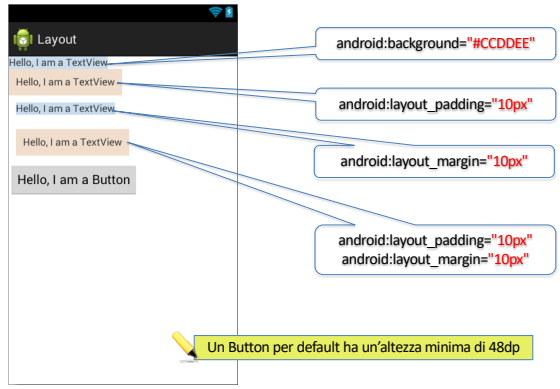
Layout – posizione e grandezza

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



60

Layout – padding e margini



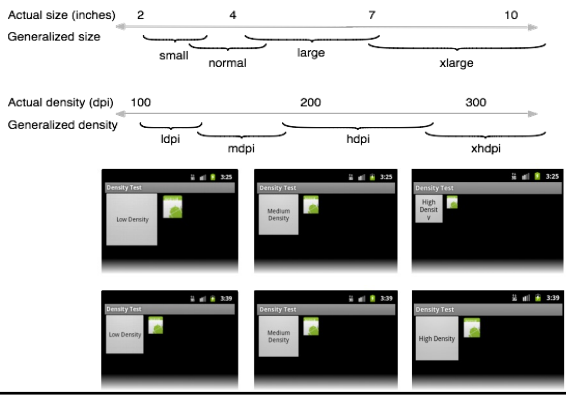
61

Misure pixel: px vs. dp

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

62

Misure Pixel



63

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)

64

Unità di misura

- 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

65

Layouts – Linear Layout

- 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

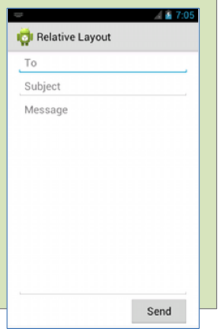
66

Layouts – Linear Layout

Slide 67

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2022

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



67

Layouts – Linear Layout

Slide 68

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2022

- 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

68

Layouts – Relative Layout

Slide 69

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2022

- 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"

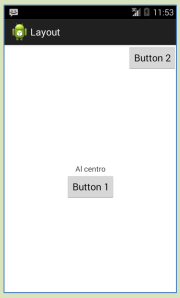
69

Layouts – Relative Layout

Slide 70

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2022

```
<?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>
```



70

Layout - ConstraintLayout

Slide 71

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2022

- 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

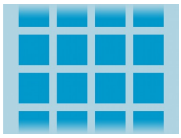
71

Layouts – Grid View

Slide 72

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2022

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

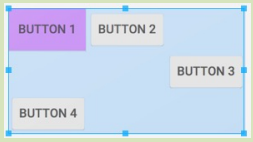


- Adapter
  - fornisce gli elementi da inserire nel Grid View
    - lo vedremo in seguito

72

Layouts – Grid Layout

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

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

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

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

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

</GridLayout>
```



73

Layouts – List View

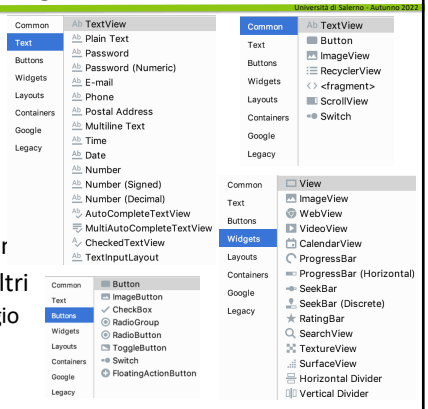
- Visualizza un insieme di elementi organizzati in una lista
  - numero totale variabile
  - visibile solo una parte
  - scroll

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

74

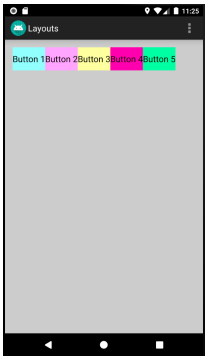
Layouts - widget

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



75

App Layouts

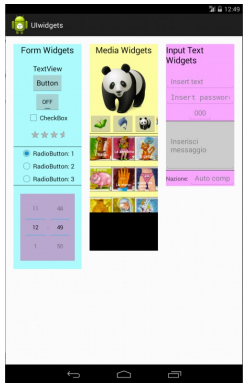


02-Layouts

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

76

Widgets



03-Widgets

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

77

ANDROID Mobile Programming

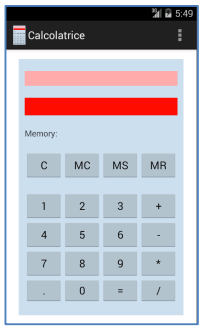
La nostra prima (vera) app

78

Prima app

04-Calcolatrice

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

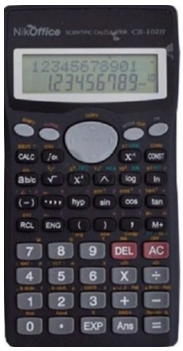


79

Esercizio

- Calcolatrice più realistica





80

ANDROID Mobile Programming

Android Studio Debugger

81

Copiare (rinominare)

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

82

Android Studio Debugger

- 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

83

Android Studio Debugger

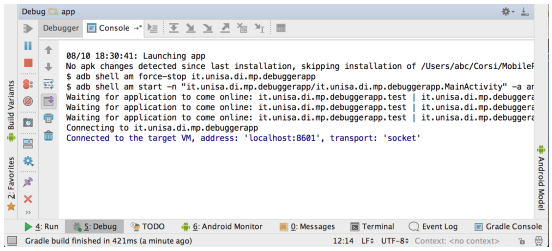
- Build variant
  - Deve essere "debuggable"
    - Build>Select Build Variant
- Per lanciare l'app in modalità debug
  - Run>Debug (CTRL-ALT-D)
  - icona
- Selezionare la device

84

Android Studio Debugger

Slide 85

- Si apre la finestra di debug
  - CMD-5



85

Breakpoints

Slide 86

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

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

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

86

Comandi debugger

Slide 87

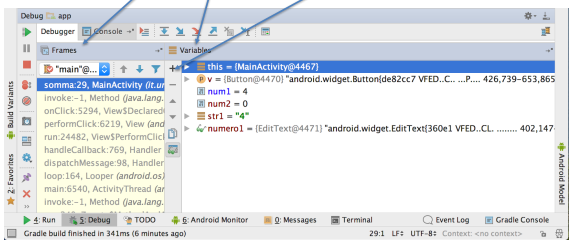
- Valuta una espressone
- Esecuzione di una singola istruzione
- Entra all'interno di una funzione
- Esci dalla funzione
- Riprendi l'esecuzione fino al prossimo breakpoint

87

Debugger

Slide 88

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



88

Messaggi di log: Log.X

Slide 89

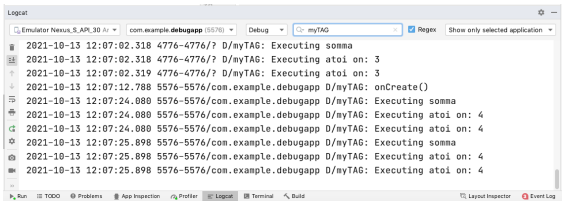
- private static final String TAG = "MyActivity";
- Log.**x**(TAG, "messaggio");
- **x** può essere
  - d: debug
  - e: errore
  - i: info
  - w: warning
  - v: verbose

89

Logcat

Slide 90

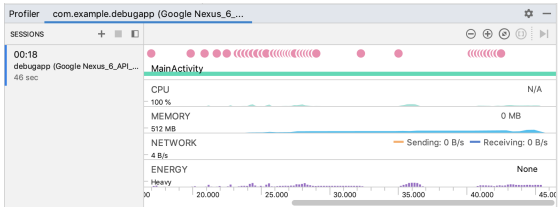
- Logcat
  - Messaggi di log



90

Profiler

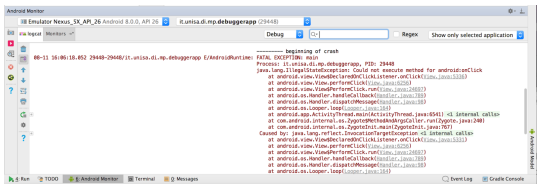
- Profiler
  - Uso memoria, CPU, rete e GPU



91

Logcat

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



92

DebugApp

05-DebugApp

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

93

ANDROID Mobile Programming

ListView

94

ListView

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

95

ListView

- Lista semplice

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

- File xml per il singolo elemento della lista

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

96

ListView

- Definire l'array con gli elementi
  - String [] array = {"Pasquale","Maria","Michele","Antonella", "Vincenzo", "Teresa", "Roberto", "Rossella", "Antonio", "Luca", "Liliana", "Stefania", "Francesca", "Andrea", "Marco", "Elisa", "Anna", "Lorenzo"};
- Definire un adapter
  - ArrayAdapter<String> arrayAdapter = new ArrayAdapter<String>(**context**, R.layout.***list_element***, R.id.***textViewList***, array);
- Individuare il widget listview
  - listView = findViewById(R.id.***mylistview***);
- Associare l'adapter al widget
  - **listView.setAdapter(arrayAdapter);**

97

ListView

- Definire un listener per i click sugli elementi

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

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

98

ListView

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

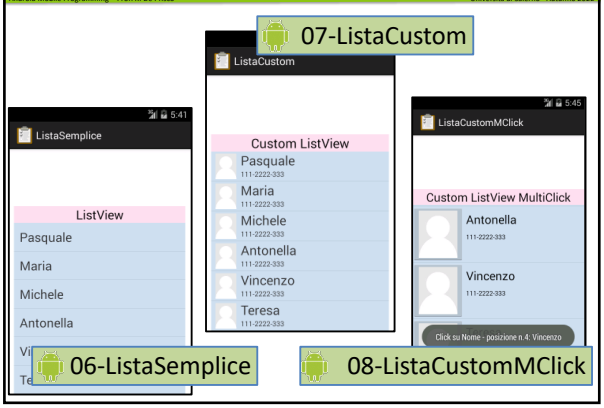
99

ListView

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

100

ListView



101

ANDROID Mobile Programming

Ciclo di vita

102

Ciclo di vita delle attività

- Ogni "Activity" ha un ciclo di vita

Attività non esiste

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

Attività in esecuzione

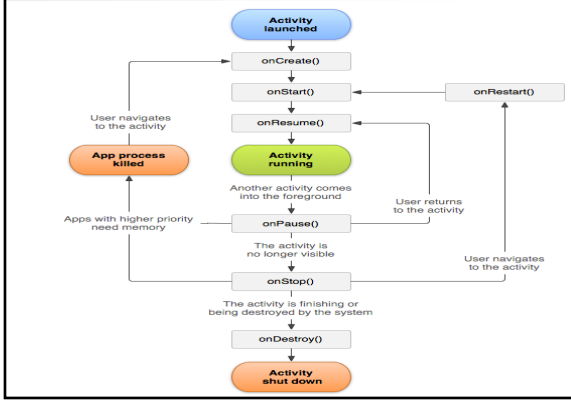
4. onPause()
5. onStop()
6. onDestroy()

Attività non esiste

- Eseguiti secondo un determinato schema

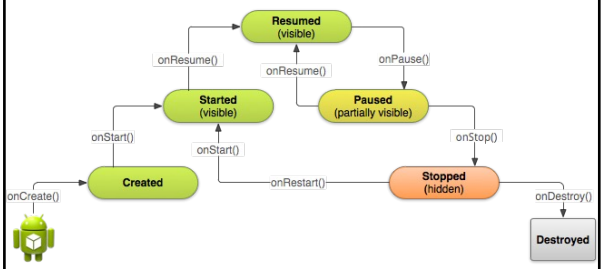
103

Ciclo di vita delle attività



104

Ciclo di vita delle attività



105

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()

106

Ciclo di vita delle attività

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

107

onSaveInstanceState

- Si salva lo stato in onSaveInstanceState()

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

- Lo si recupera in onCreate()

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

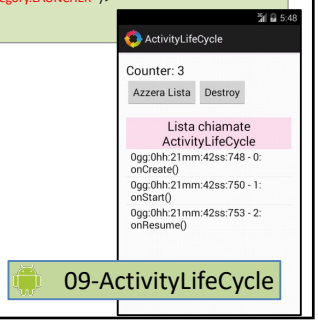
108

Main activity

Slide 109

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2022

```
<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>
```



09-ActivityLifeCycle

109

Ciclo di vita delle attività

Slide 110

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2022

- App Calcolatrice
  - Cosa succede se ruotiamo il dispositivo?

Correggere l'errore nell'app Calcolatrice

110

ANDROID Mobile Programming

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno – Autunno 2022

Ciclo di vita e Cambi di configurazione

111

111

Configurazione device

Slide 112

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2022

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

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

112

Cambio di configurazione

Slide 113

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2022

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

113

onConfigurationChanged()

Slide 114

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2022

- 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()

114

onConfigurationChanged()


```
@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

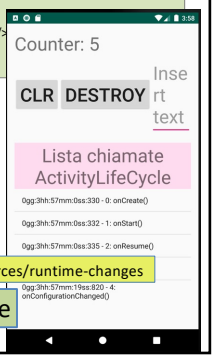
<https://developer.android.com/reference/android/content/res/Configuration

115

Main activity


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

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



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

10-OnConfigurationChange

116

ANDROID Mobile Programming

Backstack

117

Backstack

- 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.

118

Backstack

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

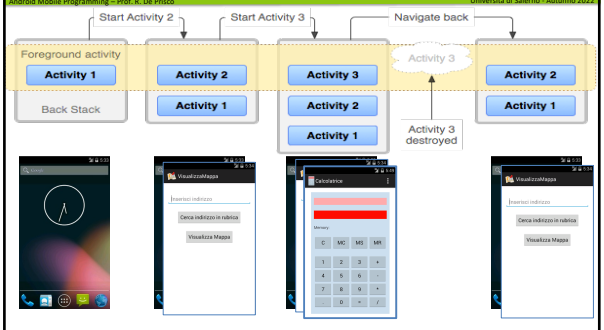
119

Backstack

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

120

Backstack

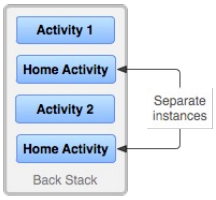


- Continuando a premere Back si ritorna all'Home screen

121

Istanze multiple

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



11-MultiActivity

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

122

ANDROID Mobile Programming

Intent

123

Classe Intent

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

124

Intent

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

125

Intent

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

126

Intent

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

127

Intent


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

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

128

Intent


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

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

129

Intent Extras

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

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

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

130

Intent Flags

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

131

Intent Component

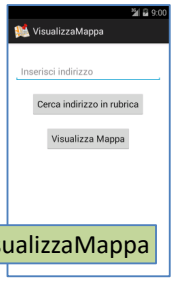
- Permette di specificare l'attività "target"
  - da usare quando c'è una sola specifica attività (componente) che deve ricevere l'intent

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

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

132

VisualizzaMappa

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



12-VisualizzaMappa

133

Intent

13-Quiz

134

Quiz

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

Correggere l'errore dovuto alle rotazioni nell'app Quiz

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

Individuare e correggere gli errori e colmare le omissioni

135

ANDROID Mobile Programming

Permessi

136

Permessi

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

137

Permessi

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

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

138

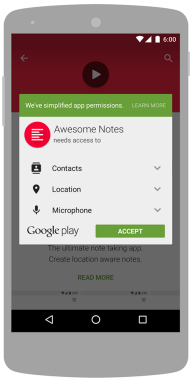
Permessi

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

139

API < 23

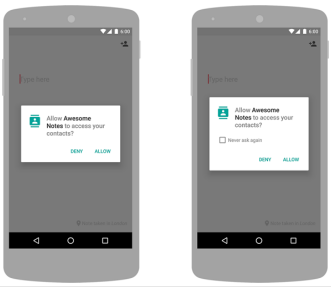
- Quando si istalla l'app
  - bisogna accettare i permessi



140

API ≥23

- Quando lanciamo l'app
  - alla seconda richiesta avremo la possibilità di rendere definitiva la risposta



141

Gestione permessi

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

142

Permessi


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

143

Permessi normali (API 23, 6.0)

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

144

Permessi pericolosi (API 23, 6.0)

Gruppo	Permesso
CALENDAR	READ_CALENDAR
	WRITE_CALENDAR
CAMERA	CAMERA
CONTACTS	READ_CONTACS
	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

145

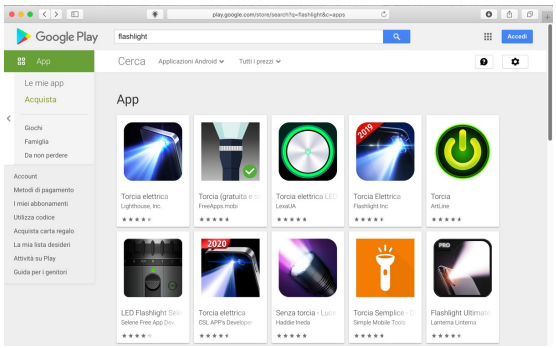
Permessi pericolosi (API 23, 6.0)

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

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

146

Permessi app "torcia"



147

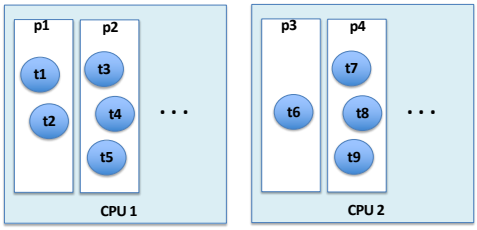
ANDROID Mobile Programming

Threads

148

Threads

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



149

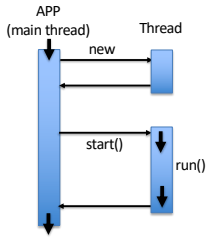
Java Threads

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

150

Threads

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



151

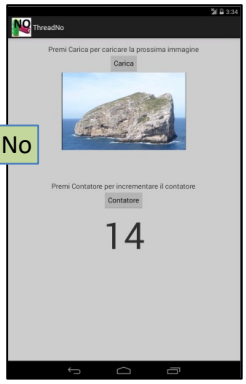
Threads

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

152

Threads

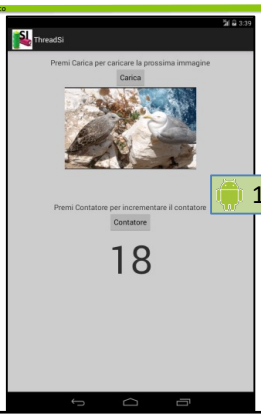
14-ThreadNo



153

Threads

15-ThreadSi



154

Async task

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

155

AsyncTask

- Classe Java generica

```
class AsyncTask<Params, Progress, Result> {
...
}
```

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

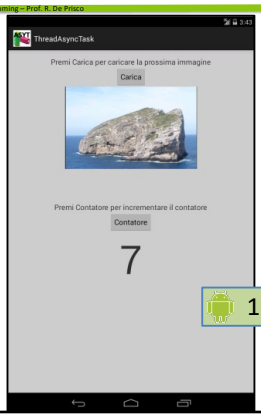
156

AsyncTask.execute()

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

157

AsyncTask



16-ThreadAsyncTask

158

ANDROID Mobile Programming

Fragments

159

159

Fragments

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

160

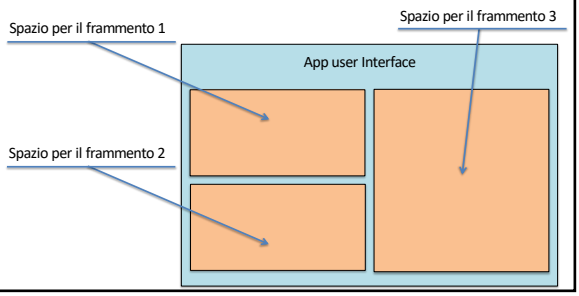
Fragments

- 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

161

Fragments

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



162

Fragments

- 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

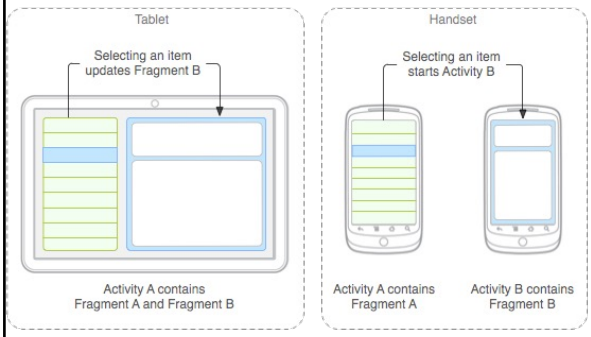
163

Fragments

- 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

164

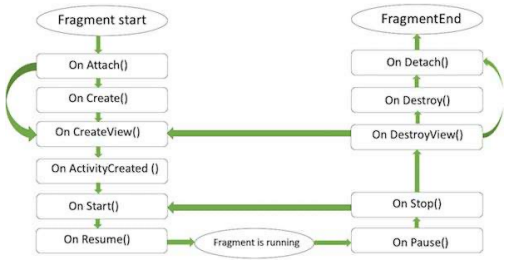
Fragments



165

Creare frammenti

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



166

Creare frammenti

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

167

Creare frammenti


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

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

168

Fragments

Slide 169

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2022

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

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

Parameters

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

Returns

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

169

Creare frammenti

Slide 170

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2022

- 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>
```

170

Creare frammenti

Slide 171

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2022

- Oppure dinamicamente a runtime

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

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

171

Gestire i frammenti

Slide 172

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2022

- Usiamo il FragmentManager

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

- Iniziamo una transazione

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

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

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

172

Gestire i frammenti

Slide 173

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2022

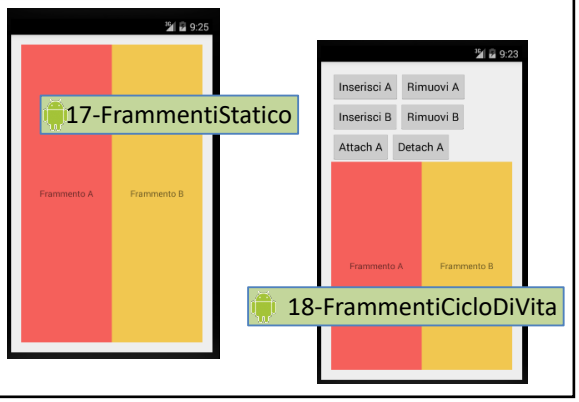
- addToBackStack()
  - per inserire i cambiamenti nel backstack
- Il backstack considera solo le activity
  - dobbiamo gestire manualmente i frammenti
- Se non chiamiamo addToBackStack
  - quando premiamo back "salteremo" i cambiamenti fatti con i frammenti
  - non è quello che l'utente si aspetta

173

Esempi

Slide 174

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno - Autunno 2022



174

Comunicare con l'activity

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

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

175

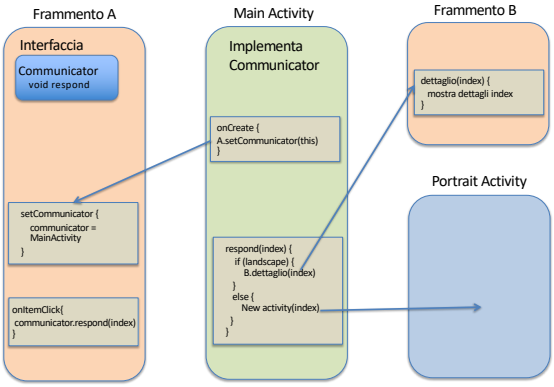
Comunicare con l'activity

- App
  - lista
  - dettagli elementi

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

176

Comunicare con l'activity



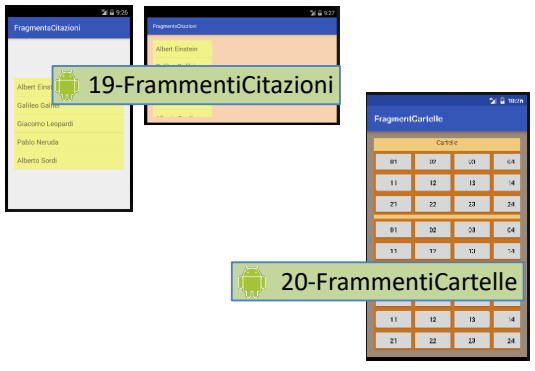
177

Frammenti e backstack

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

178

Esempi



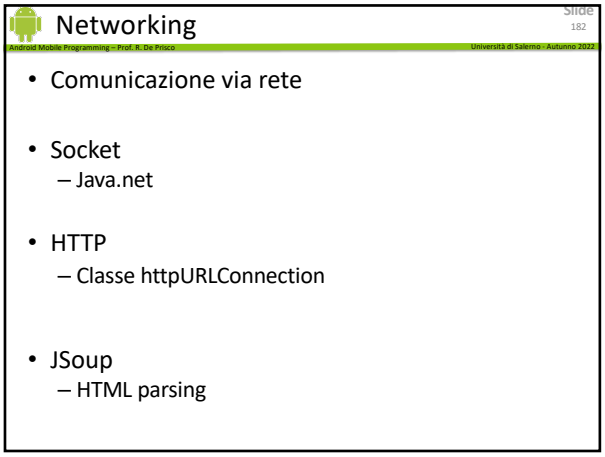
179

Frammenti

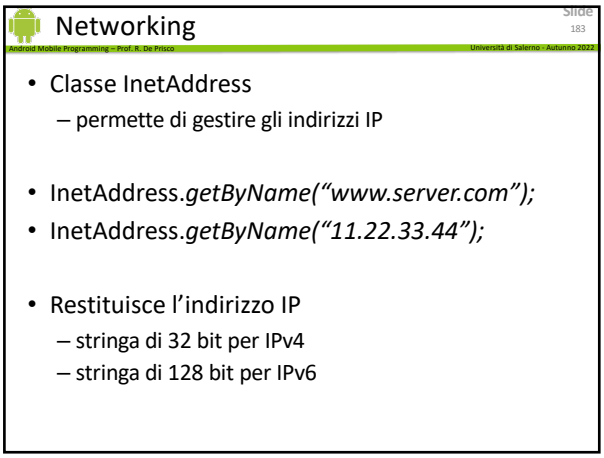
- Esercizio (avanzato)
- L'app FrammentiCartelle utilizza un layout predefinito di 12 cartelle
  - quindi può gestire al massimo 12 cartelle
- Scrivere una nuova versione in cui il layout viene costruito dinamicamente
  - creare nuove view (LinearLayout, Frame, etc)
  - LayoutParameters
  - view.add()

180

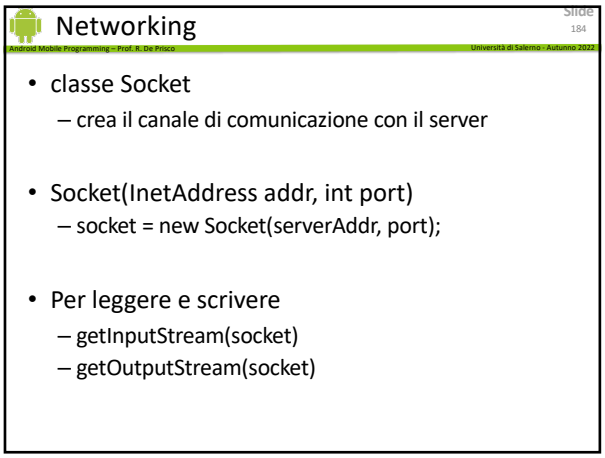
181



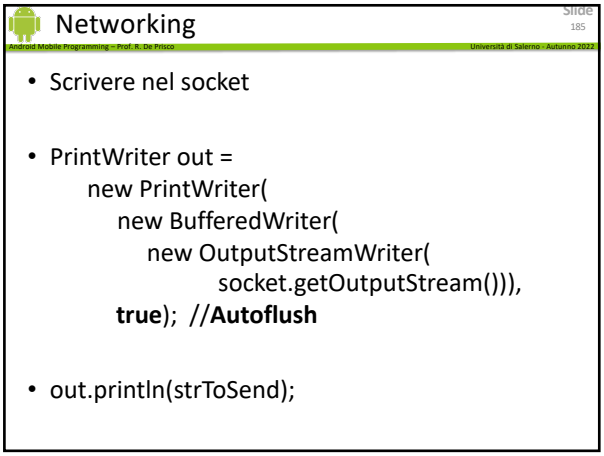
182



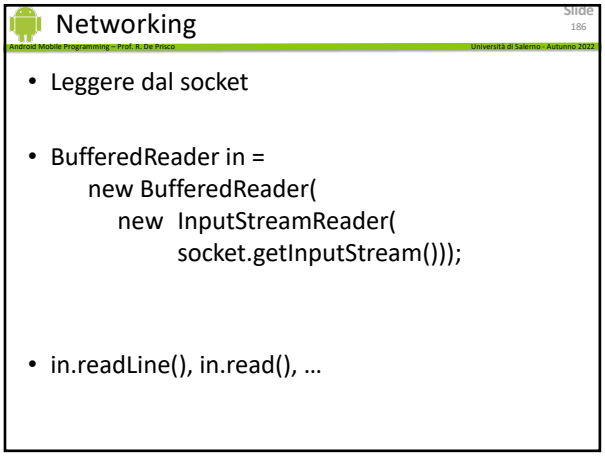
183



184



185



186

Localhost

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

187

Networking

21-SocketRaw

22-SocketRawProgressBar

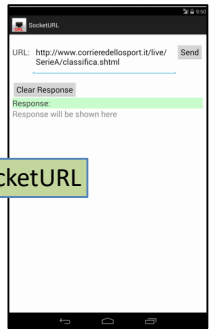
188

URL

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

189

Networking



23-SocketURL

190

Documenti HTML

- Dati in documenti HTML
  - difficile estrarli
- Esistono delle librerie che implementano il parsing di documentl HTML
  - es. JSOUP
- Per utilizzare una libreria
  - procurarci 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

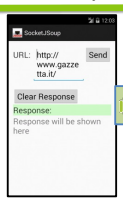
191

JSoup

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

192

Errore: Cleartext not permitted



24-SocketJSoup

- 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

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

193

ANDROID Mobile Programming

Data Storage

194

Data storage

- Relativo all'app
  - ogni app ha uno spazio privato per i file
    - nella memoria interna (app-specific)
    - possibile anche nella memoria esterna, in directory dedicate
- Shared storage
  - file che possono essere condivisi con altre app
- Preferences
  - dati chiave-valore privati
- Database
  - Database privati SQL

195

Data storage

	Type of content	Access method	Permissions needed	Can other apps access?	Files removed on app uninstall?
App-specific files	Files meant for your app's use only	From internal storage, `getFilesDir()` or `getCacheDir()` From external storage, `getExternalFilesDir()` or `getExternalCacheDir()`	Never needed for internal storage Not needed for external storage when your app is used on devices that run Android 4.4 (API level 19) or higher	No	Yes
Media	Shareable media files (images, audio files, videos)	`MediaStore` API	`READ_EXTERNAL_STORAGE` when accessing other apps' files on Android 11 (API level 30) or higher `READ_EXTERNAL_STORAGE` or `WRITE_EXTERNAL_STORAGE` when accessing other apps' files on Android 10 (API level 29) Permissions are required for all files on Android 9 (API	Yes, though the other app needs the `READ_EXTERNAL_STORAGE` permission	No

196

Data storage

	Type of content	Access method	Permissions needed	Can other apps access?	Files removed on app uninstall?
Documents and other files	Other types of shareable content, including downloaded files	Storage Access Framework	None	Yes, through the system file picker	No
App preferences	Key-value pairs	Jetpack Preferences library	None	No	Yes
Database	Structured data	Room persistence library	None	No	Yes

197

SharedPreferences

- 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.

198

SharedPreferences

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

199

File

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

200

File


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

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

- La modalità può essere
  - MODE_PRIVATE (file accessibile solo all'app)
  - MODE_APPEND
  - MODE_WORLD_READABLE (leggibile da tutti)
  - MODE_WORLD_WRITABLE (scrivibile da tutti)

Obsolete API 17

201

File

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

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

202

File

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

203

File temporanei

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

204

File su External Storage

- 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

205

External Storage


```
<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.

206

External Storage


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

207

Condividere file con altre app

- 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;
}
```

208

SQL – Quick tutorial

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

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

209

SQL – Quick tutorial

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

210

SQL – Quick tutorial

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

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

211

SQL – Quick tutorial

- Esempi di inserimento

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

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

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

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

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

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

212

SQL – Quick tutorial

- Tavola prodotta

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

213

SQL – Quick tutorial

- Esempi di select

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

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

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

214

SQL – Quick tutorial

- Esempi di update

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

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

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

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

215

Database SQL

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

216

Database SQL

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

217

Database SQL

- ContentValues
  - Classe «contenitore»
  - Scriviamo i valore di un record del database in un oggetto ContentValues
  - db.insert(nomeTavola, null, oggettoContentValues)
    - Il secondo argomento specifica cosa fare se l'oggetto ContentValues è nullo.
      - null = non inserire nessun record
      - nome di una colonna = inserisce un record con un valore nullo per quella colonna

218

Database SQL

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

219

Database


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

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

220

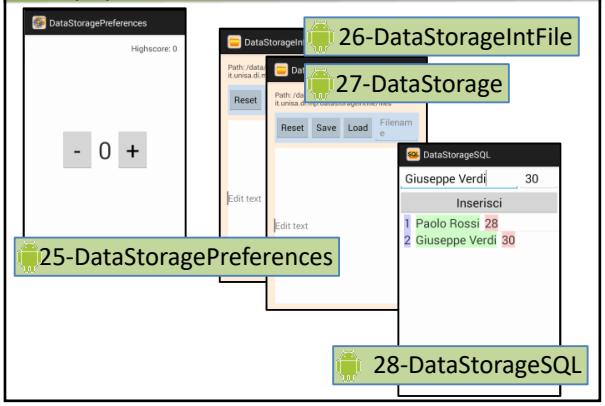
Database


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

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

221

Data Storage



222

ANDROID Mobile Programming

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno – Autunno 2022

Grafica

223

223

Grafica

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

224

Grafica

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

29-GraficaSimpleImg

225

Animazioni

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

226

Animazioni

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

30-GraficaAnimazioni

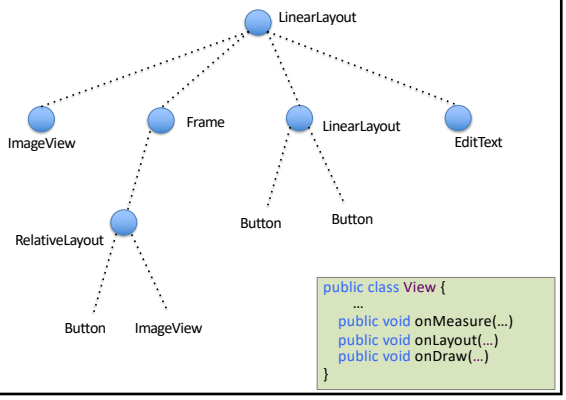
227

Custom Views

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

228

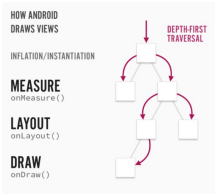
Albero delle View



229

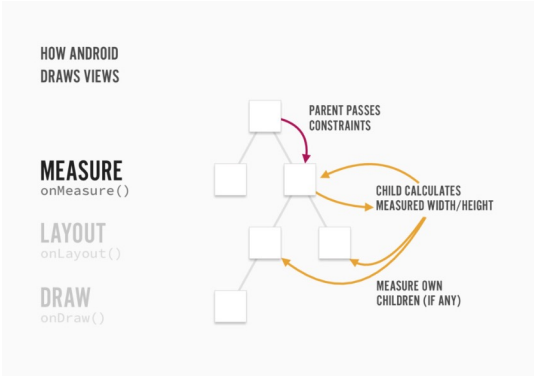
Meccanismo di layout

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



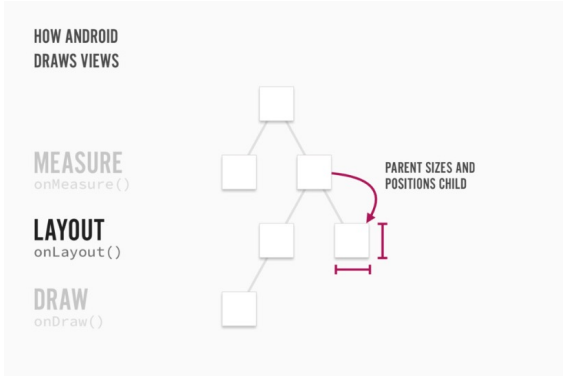
230

Fase di misurazione



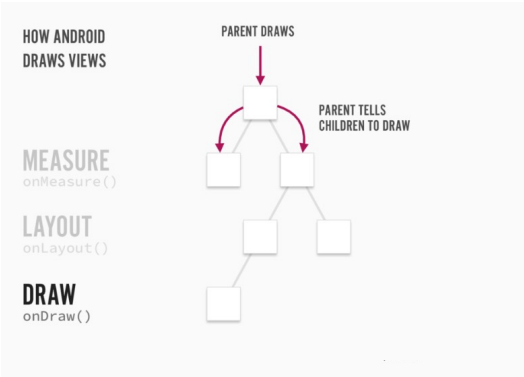
231

Fase di layout



232

Fase di disegno



233

Fase di misurazione

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

234

MeasureSpec


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

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

235

Fase di misurazione (onMeasure)

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

236

Fase di layout (onLayout)

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

237

Fase di disegno (onDraw)

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

238

Meccanismo di layout

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

239

onMeasure()


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

Parameters

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

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

240

Layout

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

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

241

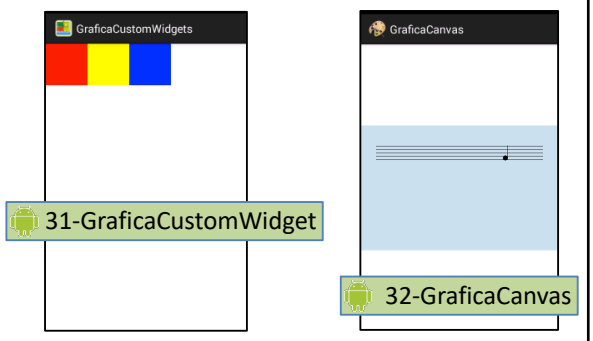
Disegnare nel canvas

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

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

242

Esempi



243

Multitouch

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

244

Multitouch

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

245

Multitouch

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

246

Multitouch

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

247

Multitouch

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

248

Multitouch

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

249

Multitouch

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

250

Multitouch

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

251

Multitouch

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

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

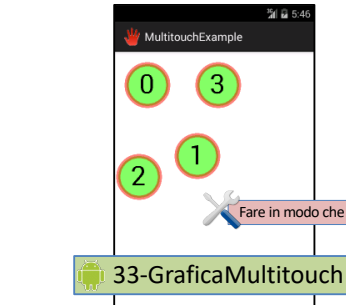
252

Multitouch

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

253

Multitouch



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

33-GraficaMultitouch

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

254

GestureDetector

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

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

255

GestureDetector

- 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

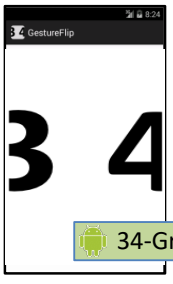
256

ViewAnimator e ViewFlipper

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

257

GestureDetector



34-GraficaGestureFlip

Implementare anche il fling per il decremento

258

ANDROID Mobile Programming

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno – Autunno 2022

Media player

259

259

MediaPlayer

Slide 260

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

260

MediaPlayer

Slide 261

- 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

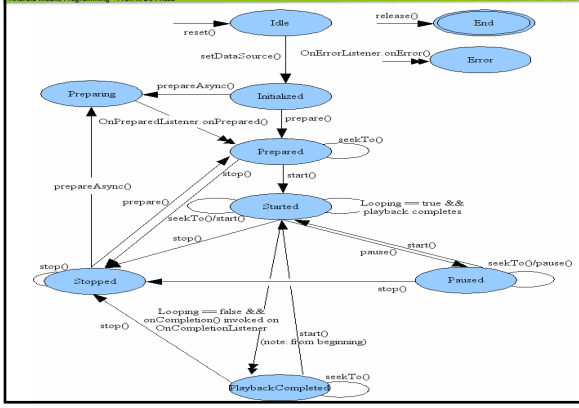
```
String url = "http://......"; // your URL here
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(url);
mediaPlayer.prepare(); // might take long! (for buffering, etc)
mediaPlayer.start();
```

setDataSource() richiede la gestione di IOException o di IllegalArgumentException

261

MediaPlayer

Slide 262



262

MediaPlayer

Slide 263

- Rilasciare la risorsa

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

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

263

MediaPlayer

Slide 264

- Audio focus
- Poiché c'è un solo canale di output
  - l'utilizzo da parte di più applicazioni può essere un problema
    - es. se stiamo ascoltando musica potremmo non sentire l'arrivo di un messaggio
- È possibile gestire l'accesso contemporaneo usando l'audio focus
  - un'app richiede l'audio focus per usare l'audio
  - se lo perde deve o smettere di suonare o abbassare il proprio volume

264

MediaPlayer

- Inserire altri brani nel MusicPlayer
- Implementare la lista dei brani con un ListView

265

ANDROID Mobile Programming

Sensori

266

Sensori

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

267

Sensori

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

268

Sensori

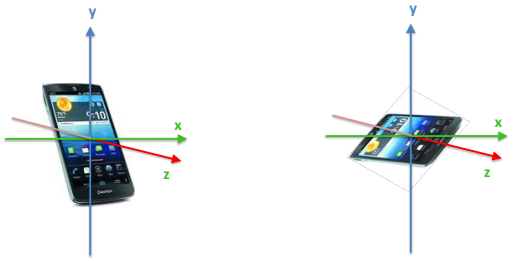
- Poche device hanno tutti i tipi di sensori
  - a volte più di un sensore dello stesso tipo
  - ecco un elenco parziale:

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

269

Sensori

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



270

Sensori

- L'attività deve implementare SensorEventListener

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

- Poi si deve controllare se il sensore esiste

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


Sensori

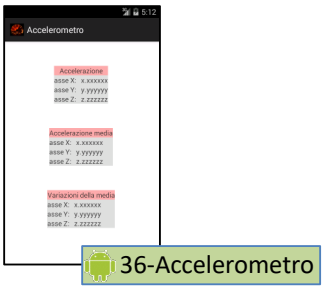

```
@Override
protected void onResume() {
   super.onResume();
   mSensorManager.registerListener(this, mLight, SensorManager.SENSOR_DELAY_NORMAL);
}

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

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

Sensori


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

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

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


Sensori

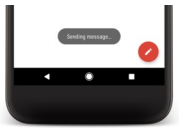


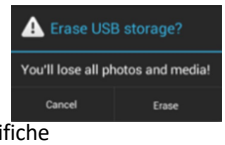
ANDROID
Mobile Programming

Notifiche

Notifiche

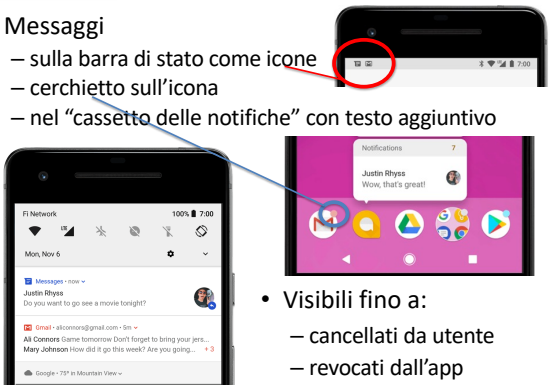
- Notifiche: informazioni all'utente al di fuori dell'interfaccia grafica dell'app
- Toast
  - brevi messaggi temporanei
- Dialog
  - interazione con l'utente
- Notifiche
  - status bar e cassetto delle notifiche





Notifiche

- 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

277

Notifiche

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

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

278

Notifiche

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

279

Notifiche - compatibilità

- Sistema di notifiche in 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"
}
```

280

Notifiche

- 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

281

Notifiche

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

282

ANDROID Mobile Programming

Alarms

283

Alarms

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

284

Alarms

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

285

Alarms

- Alarms «imprecisi» (inexact)
  - Per questo tipo di allarme il sistema non garantisce l'esecuzione al tempo richiesto
    - il SO operativo può modificare i triggerTime per minimizzare wakeups e l'uso della batteria
    - In particolare se ci sono restrizioni sull'uso della batteria
  - Da API 31, il sistema garantisce che vengano eseguito entro un'ora dal tempo richiesto
- Alarms «precisi» (exact)
  - Vengono eseguiti al tempo richiesto
  - Da API 31, è necessario un permesso

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

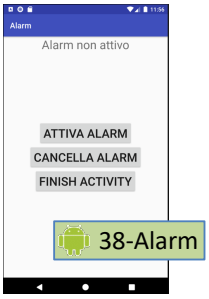
286

Alarms

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

287

Alarm



288

ANDROID Mobile Programming

Android Mobile Programming – Prof. R. De Prisco — Università di Salerno – Autunno 2022

Content Providers, Broadcast, Services

289

289

Oltre le Activity

- 4 componenti fondamentali di Android
  - Activity
  - Broadcasts
  - Content Providers
  - Services
- Finora abbiamo parlato delle activity
  - servono per lo sviluppo delle app!
  - Le altre componenti sono di ausilio e servono in casi particolari, ma in alcuni casi sono estremamente utili
- Nelle prossime slide ci sono dei cenni

290

Broadcasts

Develop › API Guides › App Components

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.

291

Content providers

Develop › API Guides › App Components

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.

292

Services

Develop › API Guides › App Components

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.

293

Broadcasts

- Messaggi
  - Globali, possono essere ricevuti da tutti
  - Locali, ricevuti solo all'interno dell'app
- BroadcastSender
  - Per generare un messaggio
- BroadcastReceiver
  - Per ricevere i messaggi
- Intent
  - Usato per contenere il messaggio
  - più precisamente, il messaggio **è** un Intent

294

Broadcasts

- Broadcast di sistema
  - spediti dal sistema operativo
- Eventi di interesse globale, esempi:
  - android.intent.action.AIRPLANE_MODE
  - android.intent.action.BOOT_COMPLETED
  - android.intent.action.ACTION_TIME_TICK
  - fino a API 24, esempi
    - android.intent.action.ACTION_NEW_PICTURE
    - android.intent.action.ACTION_TIME_VIDEO

295

Broadcasts

- BroadcastReceiver
  - Devono essere dichiarati/registrati
- Dichiarati nel manifesto (statici)
  - il filtro specifica i messaggi di interesse
  - android:exported="true"

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

296

BroadcastReceiver

- android:exported
  - obbligatorio specificarlo

`android:exported`

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

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

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

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

297

Broadcasts

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

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

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

298

Broadcasts

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

299

Broadcast

- LocalBroadcasts

This class is deprecated.
LocalBroadcastManager is an application-wide event bus and embraces layer violations in your app: any component may listen events from any other. You can replace usage of **`LocalBroadcastManager`** with other implementation of observable pattern, depending on your usecase suitable options may be **`LiveData`** or reactive streams.

300

Riassumendo

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

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

Riassumendo

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

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

- Se la registrazione è dinamica
  - il ricevitore viene registrato quando si chiama
    - 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

- Per spedire un messaggio si usa il metodo
  - LocalBroadcastManager.sendBroadcast(Intent i)
    - messaggi locali
  - Context.sendBroadcast(Intent i)
    - messaggi globali
- 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

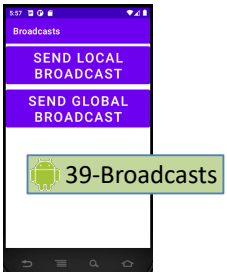
sendBroadcast(Intent i) è disponibile sia nel LocalBroadcastManager che nel Context. Si utilizza il primo per messaggi locali all'app ed il secondo per messaggi system-wide. sendBroadcast(Intent i, String permission) è disponibile solo nel Context.

BroadcastReceiver

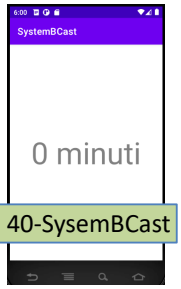
- Esempi di altri eventi globali.:
  - android.intent.action.AIRPLANE_MODE
  - android.intent.action.BATTERY_LOW
  - android.intent.action.DATA_SMS_RECEIVED
  - android.intent.action.DATE_CHANGED
  - android.intent.action.DEVICE_STORAGE_LOW
  - android.intent.action.TIMEZONE_CHANGED
  - android.intent.action.TIME_TICK
  - android.intent.action.USER_PRESENT
  - android.intent.action.WALLPAPER_CHANGED
  - …

Lista completa: sdk/platforms/android-19/data/broadcast_actions.txt

Esempi



39-Broadcasts



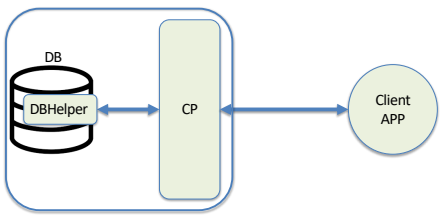
40-SysemBCast

41-AutoStart

- ACTION_TIME_CLICK
  - Intent inviato ogni minuto

ContentProvider

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



ContentProvider

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

307

ContentProvider

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

308

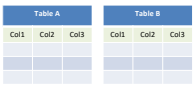
ContentProvider

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

309

CustomContentProvider

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



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

310

CustomContentProvider

- AUTHORITY va dichiarata nel manifesto

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

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

AUTHORITY

311

CustomContentProvider

- Espone il metodo query

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

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

312

Client

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

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

- PATH="ALL_STUDENTS"
  - usato internamente dal CP per individuare la tavola

313

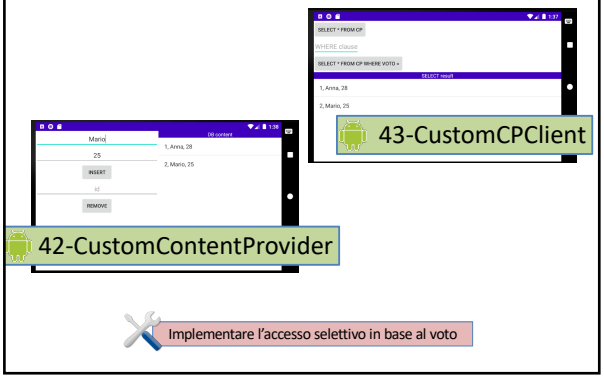
ContentProvider

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

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

314

CustomContentProvider



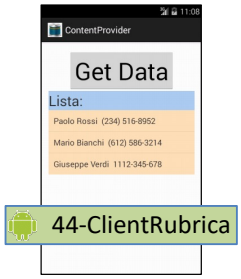
Implementare l'accesso selettivo in base al voto

315

Client Rubrica

- Serve il permesso:

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



316

Services

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

317

Services

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

318

Services

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

319

Services (unbound)

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

320

Services (bound)

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

321

Services

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

322

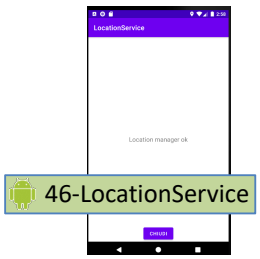
Services

- Occorre dichiarare il Service nel Manifesto:

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

323

Esempi



324

Services

- Nell'esempio visto il servizio è nella stessa app del client
- Per approfondire i Service:
  - http://developer.android.com/guide/components/bound-services.html

325

ANDROID Mobile Programming

Fine del corso!

328