

 Prof. Roberto De Prisco

Android Mobile Programming - Prof. R. De Prisco Università di Salerno - Autunno 2018

ANDROID Mobile Programming

(ex BDSIR)

1

 per favore ...

Android Mobile Programming - Prof. R. De Prisco Università di Salerno - Autunno 2018



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

 Scrivere un'app che mette il silenzioso dalle 15:00 alle 17:00 del lunedì e giovedì

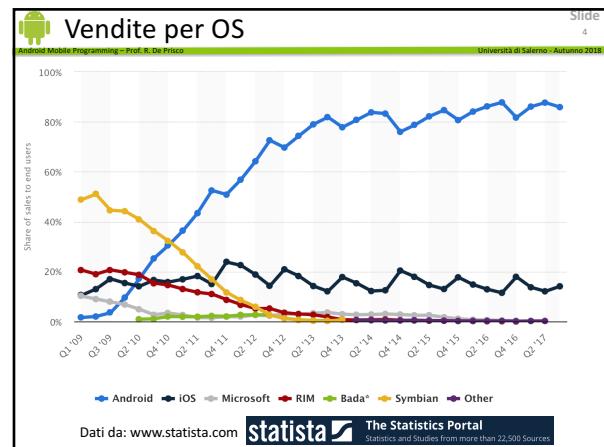
Slide 2

 Info corso

Android Mobile Programming - Prof. R. De Prisco Università di Salerno - Autunno 2018

- Prof. Roberto De Prisco
 - studio: 4° piano, studio 58
 - numerazione Dip. di Informatica
 - robdep@unisa.it
- Orario lezioni
 - Lunedì 15:00-17:00 Aula F8
 - Giovedì 15:00-17:00 Aula F8
- Ricevimento
 - Lunedì 17:00-18:00
 - Giovedì 11:00-13:00

Slide 3



 Telefoni android prodotti da

Android Mobile Programming - Prof. R. De Prisco Università di Salerno - Autunno 2018



ed altri ...

Slide 5

 Vendite per marca

Android Mobile Programming - Prof. R. De Prisco Università di Salerno - Autunno 2018

Period	Samsung	Apple	Huawei	OPPO	vivo	Others
2016Q1	23.8%	15.4%	8.4%	5.9%	4.4%	42.1%
2016Q2	22.7%	11.7%	9.3%	6.6%	4.8%	45.0%
2016Q3	20.9%	12.5%	9.3%	7.1%	5.9%	44.3%
2016Q4	18.0%	18.2%	10.5%	7.3%	5.7%	40.2%
2017Q1	23.3%	14.7%	10.0%	7.5%	5.5%	39.0%

Source: IDC, May 2017

Slide 6

Altre motivazioni

Slide 7
Android Mobile Programming - Prof. R. De Prisco
Università di Salerno - Autunno 2018

- Ambiente di sviluppo (Android Studio)
 - facile da installare
 - molti tools per lo sviluppo di app
- Installazioni delle app
 - facile
 - non richiede nessuna registrazione
- Moltissime risorse online

Cosa faremo nel corso

Slide 8
Android Mobile Programming - Prof. R. De Prisco
Università di Salerno - Autunno 2018

Impareremo a scrivere app per Android!!!

Cosa faremo nel corso

Slide 9
Android Mobile Programming - Prof. R. De Prisco
Università di Salerno - Autunno 2018

Android Studio v. 3.1.4

Vedremo tutto ciò che serve per scrivere app ...

Argomenti

Slide 10
Android Mobile Programming - Prof. R. De Prisco
Università di Salerno - Autunno 2018

- Piattaforma Android
- Android Studio
- Emulatore
- Layout
- Listener
- Intent
- Permessi
- Alarms
- Frammenti
- Networking
- Grafica
- Sensori
- Multimedia
- Data storage

Risorse didattiche

Slide 11
Android Mobile Programming - Prof. R. De Prisco
Università di Salerno - Autunno 2018

- Lezioni!!!!
- Google
 - <http://developer.android.com>
 - <http://developer.android.com/guide>
 - <http://developer.android.com/training>
- Books
 - BigNerd Ranch (in inglese)
 - <http://www.bignerdranch.com/>
- Coursera
 - ottimo video corso (in inglese)
 - <https://www.coursera.org/course/android>
- Googling!

Sito e piattaforma S3

Slide 12
Android Mobile Programming - Prof. R. De Prisco
Università di Salerno - Autunno 2018

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

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

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

Homepage	Homepage
Programma	Benvenuto nella homepage del corso di Mobile Computing per l'anno accademico 2018-2019.
Codice	
Slides	
Riferimenti	
Prerequisiti	• Lunedì 15:00-17:00 aula F8
Esami	• Giovedì 15:00-17:00 aula F8
Risultati	
App contest	Annunci

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

 **Contest**

Slide 13
Università di Salerno - Autunno 2018
Android Mobile Programming - Prof. R. De Prisco

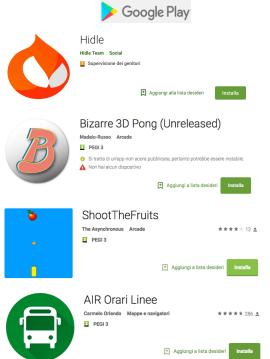
- La migliore app sviluppata DURANTE il corso verrà premiata con un dispositivo Android!
 - Per partecipare occorre
 - frequentare il corso
 - sviluppare in gruppi di 2 persone max
 - consegnare l'app una settimana prima del contest
 - app di una certa complessità
- Sponsorizzato da eTuitus
 - www.etuitus.it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



 **Contest – albo d'oro**

Slide 14
Università di Salerno - Autunno 2018
Android Mobile Programming - Prof. R. De Prisco

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



 **Esame**

Slide 15
Università di Salerno - Autunno 2018
Android Mobile Programming - Prof. R. De Prisco

- L'app sviluppata per il contest non sostituisce la prova di laboratorio
 - ma verrà comunque valutata per l'esame
- Esame
 - Scritto
 - Laboratorio
 - vengono ammessi gli studenti che superano lo scritto
 - Orale a discrezione del docente
 - svolto in caso di dubbi sulla autenticità delle prove precedenti

 **Come contattare il docente**

Slide 16
Università di Salerno - Autunno 2018
Android Mobile Programming - Prof. R. De Prisco

- Fine lezione (o nella pausa)
- Orario di ricevimento studenti
 - Lun 17:00-18:00
 - Gio 11:00-13:00
- Email
 - Risposta non garantita, dipende dalla domanda!
 - Condizione necessaria: il messaggio deve contenere il nome del mittente
 - Appuntamento
- NON telefonare**

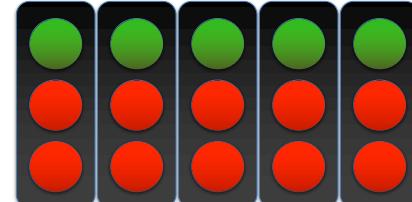
 **Domande?**

Slide 17
Università di Salerno - Autunno 2018
Android Mobile Programming - Prof. R. De Prisco



 **... si parte**

Slide 18
Università di Salerno - Autunno 2018
Android Mobile Programming - Prof. R. De Prisco

ANDROID Mobile Programming

Android Mobile Programming - Prof. R. De Prisco

Università di Salerno - Autunno 2018

La piattaforma Android

19

La piattaforma Android

Android Mobile Programming - Prof. R. De Prisco

Università di Salerno - Autunno 2018

- 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>

Slide 20

L'architettura Android

Android Mobile Programming - Prof. R. De Prisco

Università di Salerno - Autunno 2018

Si basa su un kernel Linux

Slide 21

Linux kernel Layer

Android Mobile Programming - Prof. R. De Prisco

Università di Salerno - Autunno 2018

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

Slide 22

Hardware Abstraction Layer

Android Mobile Programming - Prof. R. De Prisco

Università di Salerno - Autunno 2018

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

Slide 23

Java

Android Mobile Programming - Prof. R. De Prisco

Università di Salerno - Autunno 2018

- 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.*`

Slide 24

File dex e ART

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018

Slide 25

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

Android runtime: ART e Dalvik VM

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018

Slide 26

The diagram illustrates the Android Runtime architecture. It features a central box labeled "ANDROID RUNTIME" containing "Core Libraries". Below it are two other boxes: "Android Runtime (ART)" on the left and "Dalvik Virtual Machine" on the right.

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

Librerie native

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018

Slide 27

The diagram shows a grid of six components under the heading "Native C/C++ Libraries". The components are arranged in two rows of three: Webkit, OpenMAX AL, and Libc in the top row; Media Framework, OpenGL ES, and "..." in the bottom row.

- Molte componenti Android necessitano di librerie native
 - Webkit
 - Libc
 - openGL ES
 - ...

Application framework

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018

Slide 28

The diagram illustrates the Java API Framework. It is organized into several categories:

- Content Providers:** Activity, Location, Package, Notification
- View System:** Resource, Telephone, Window
- Managers:** Manager

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

Applicazioni (app)

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018

Slide 29

The diagram shows a row of five system application icons: Dialer, Email, Calendar, Camera, and "...".

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

Versioni

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018

Slide 30

API	Nome	Versione	%
10	Gingerbread	2.3	0.2%
15	Ice cream Sandwich	4.0	0.3%
16	Jelly Bean	4.1	1.2%
17		4.2	1.9%
18		4.3	0.5%
19	Kit Kat	4.4	9.1%
21	Lollipop	5.0	4.2%
22		5.1	16.2%
23	Marshmallow	6.0	23.5%
24	Nougat	7.0	21.2%
25		7.1	9.6%
26	Oreo	8.0	10.1%
27		8.1	2%
28	Pie	9.0	----

Dati aggiornati a luglio 2018

Data di uscita: 9 agosto 2018

 **Versioni**

Android Mobile Programming - Prof. R. De Friso

Slide 31

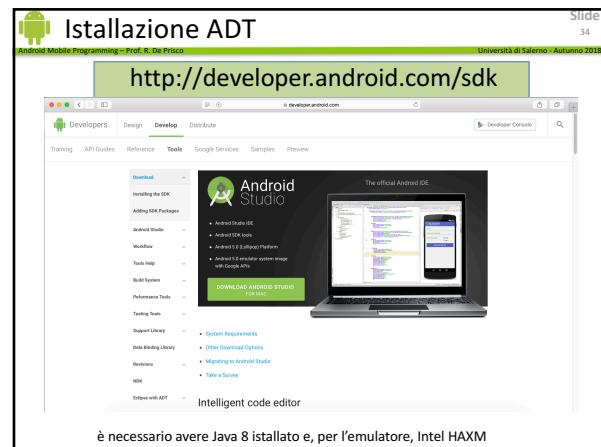
Università di Salerno - Autunno 2018

Android Name	Android Version	Usage Share
Nougat	7.0, 7.1	30.8%↓
Marshmallow	6.0	23.5%↓
Lollipop	5.0, 5.1	20.4%↓
Oreo	8.0, 8.1	12.1%↑
KitKat	4.4	9.1%↓
Jelly Bean	4.1.x, 4.2.x, 4.3.x	3.6%↓
Ice Cream Sandwich	4.0.3, 4.0.4	0.3%↓
Gingerbread	2.3.3 to 2.3.7	0.2%↓

Dati aggiornati a luglio 2018



-  **Android Studio**
- Android Mobile Programming - Prof. R. De Friso
- Slide 33
- Università di Salerno - Autunno 2018
- Installare l'ADT
 - L'interfaccia
 - L'emulatore Android
 - Strumenti per il debug
 - Altri strumenti



-  **Android Studio 2.3**
- Android Mobile Programming - Prof. R. De Friso
- Slide 35
- Università di Salerno - Autunno 2018
- Richiede:
 - Java 8
 - Intel Hardware Accelerated Execution Manager HAXM (per l'emulatore)
 - Offre
 - Piattaforma Android
 - Android SDK Tools
 - sviluppo
 - debug
 - Gradle
 - Emulatore Android

-  **Modalità sviluppatore**
- Android Mobile Programming - Prof. R. De Friso
- Slide 36
- Università di Salerno - Autunno 2018
- Modalità sviluppatore
 - Info dispositivo, Versione build
 - Click 7 volte
 - Comparirà il menu Opzioni Sviluppatore
 - Debug USB
 - Attivare
 - Dare il consenso per l'accesso
- Consentire debugging USB?

L'impronta della chiave RSA del PC è:

```
AC:30:26:88:AF
61:09:5A:46:78:BC
B1:D7:96:FB:E2
```

Consenti sempre questo PC

ANNULLA **OK**

Listeners

Slide 37

Android Mobile Programming - Prof. R. De Prisco

Università di Salerno - Autunno 2018

- 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

Prima app: CiaoMondo

Slide 38

Android Mobile Programming - Prof. R. De Prisco

Università di Salerno - Autunno 2018

- Visualizza un saluto al mondo!

CiaoMondo

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

CAMBIO IL COLORE AL BACKGROUND DEL TESTO

CiaoMondo

Slide 39

Android Mobile Programming - Prof. R. De Prisco

Università di Salerno - Autunno 2018

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

CiaoMondo

Slide 40

Android Mobile Programming - Prof. R. De Prisco

Università di Salerno - Autunno 2018

- 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'
    }
}
```

SDK

Slide 41

Android Mobile Programming - Prof. R. De Prisco

Università di Salerno - Autunno 2018

- Android SDK Manager

SDK

Slide 42

Android Mobile Programming - Prof. R. De Prisco

Università di Salerno - Autunno 2018

- Occorre installare le versioni per le quali si vuole sviluppare

Name	API	Rev.	Status
Android 24.1.2 API 24	24.1.2	2	Installed
Android SDK Tools	22	2	Installed
Android SDK Platform-tools	22	2	Installed
Android SDK Build-tools	22.0.1	1	Installed
Android NDK	22.0.1	1	Installed
Android SDK Build-tools	21.1.1	1	Installed
Android Support Repository	21.1	1	Installed
Android Support Library	21.0.2	1	Installed
Android Support R8	21.0.1	1	Installed
Android Support R8	20	1	Installed
Android Support R8	19.1.1	1	Installed
Android Support R8	19.0.3	1	Installed
Android Support R8	19.0.2	1	Installed
Android Support R8	19.0.1	1	Installed
Android Support R8	19	1	Installed
Android Support R8	18.1.1	1	Installed
Android Support R8	18.1	1	Installed
Android Support R8	18.0.1	1	Installed
Android Support R8	17	1	Installed
Android 5.1.1 (API 22)	22	1	Installed
SDK Platform	22	2	Installed
Android 5.0.1 (API 21)	22	1	Installed
Android TV ARM EABI v7a System Image	22	1	Installed
Android 5.0.1 (API 21)	22	1	Installed
ARM EABI v7a System Image	22	1	Installed
Intel x86 Atom_84 System Image	22	1	Installed
Google APIs	22	1	Installed
Google APIs Intel Atom System Image	22	1	Installed
Google APIs Intel x86 Atom_84 System Image	22	1	Installed
Google APIs Intel x86 Atom System Image	22	1	Installed

A screenshot of the Android Studio interface. The title bar says "Emulatore Android". The top navigation bar includes "Android Mobile Programming - Prof. R. De Prisco" and "Università di Salerno - Autunno 2018". The main content area has a bullet point "• Android Virtual Device Manager". Below this is a toolbar with icons for file operations like Open, Save, and Print. A "Tools" menu is open, showing options like "Tasks & Contexts", "Save File as Template...", "Generate JavaDoc...", "New Scratch File...", "IDE Scripting Console", "Create Command-line Launcher...", "Groovy Console...", and "Android". The "Android" option is highlighted with a red oval. A submenu for "Android" is displayed, containing "Navigation Editor", "Sync Project with Gradle Files", "Android Device Monitor", "AVD Manager" (which is also highlighted with a red oval), "SDK Manager", and "Enable ADB Integration".

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

slide 45

Università di Salerno - Autunno PV 2018/2019

- telnet

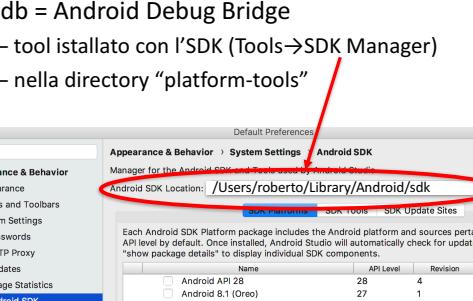
```
[root@Emulatorore ~]# telnet localhost 5554
-> telnet localhost 5554 -->n/a
Trying ::1
telnet: connect to address ::1: Connection refused
Trying 127.0.0.1...
Connected to localhost.
Administrator
Android Console: Authentication required
Android Console: type 'auth <auth_token>' to authenticate
Android Console: you can find your 'auth_token' in
'/Users/abc/.emulator_console_auth_token'
OK
auth F982-IWngfr0xSa
Android Console: type 'help' for a list of commands
OK
sms send 3331234567 "Ciao!"
OK
network speed edge
OK
help
Android console commands:
helpini?
help-verbose
ping
event
geo
gsm
cdma
crash
crash-on-exit
kill
network
power
```

- Comunicazione fra due emulatori

“numero di telefono”: 5554

“numero di telefono”: 5554

Non è possibile lanciare due istanze della stessa AVD: occorre creare due AVD diverse e lanciare un’istanza di ognuna.



• adb = Android Debug Bridge

- tool installato con l'SDK (Tools → SDK Manager)
- nella directory “platform-tools”

	Name	API Level	Revision	Status
<input type="checkbox"/>	Android 8.1 P	28	4	Update available
<input type="checkbox"/>	Android 8.0 (Oreo)	27	1	Update available
<input checked="" type="checkbox"/>	Android 8.0 (Oreo)	26	2	Update available
<input checked="" type="checkbox"/>	Android 7.1.1 (Nougat)	25	3	Installed



adb e porta telnet

Android Mobile Programming - Prof. R. De Prisco

Slide 48
Università di Salerno - Autunno 2018

ANDROID Mobile Programming



Layouts



49

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



50

Layouts

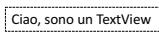
- XML
 - vantaggi
 - facile da specificare
 - separa in modo netto la definizione dell'UI dal codice dell'applicazione (facile fare modifiche)
 - svantaggi
 - elementi statici
- Programmatico
 - vantaggi
 - dinamico, si può facilmente adattare
 - svantaggi
 - dobbiamo gestire il layout nel codice dell'applicazione

51

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" />
```


- Button

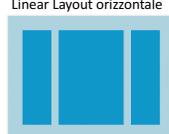

```
<Button android:id="@+id/button"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="Ciao, sono un Pulsante"  
        android:background="#00FF00" />
```



52

Layout - ViewGroup

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



53

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>
```



54

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

Layout - XML - attributi

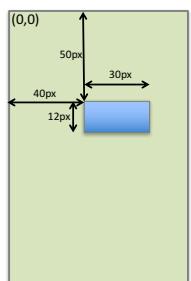
- ID (creazione)
`android:id="@+id/text"`
 - `@`: il resto della stringa deve essere interpretato
 - `android:layout_width="30px"`
 - se non c'è `@` il valore è quello letterale
- +: specifica che stiamo creando (aggiungendo) un nuovo identificatore (`id`) il cui nome è `text`
- ID (riferimento)
`android:id="@+id/text"`
 - senza il `+` è un riferimento ad un ID esistente

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`

Layout – posizione e grandezza

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



Layout – padding e margini



Annotations:

- `android:background="#CCDDEE"`
- `android:layout_padding="10px"`
- `android:layout_margin="10px"`
- `android:layout_padding="10px" android:layout_margin="10px"`
- Un Button per default ha un'altezza minima di 48dp

Misure pixel: px vs. dp

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

Misure Pixel

Android Mobile Programming - Prof. R. De Prisco

Università di Salerno - Autunno 2014

Slide 61

The diagram illustrates the relationship between screen size, density, and pixel dimensions:

- Actual size (inches):** 2, 4, 7, 10.
- Generalized size:** small, normal, large, xlarge.
- Actual density (dpi):** 100, 200, 300.
- Generalized density:** ldpi, mdpi, hdpi, xhdpi.

Below the diagram are six screenshots labeled "Density Test" showing the Android logo at different sizes across four density levels: Low Density (ldpi), Medium Density (mdpi), High Density (hdpi), and Extra High Density (xhdpi).

Alternative per i “drawable”

Android Mobile Programming - Prof. R. De Prisco

Università di Salerno - Autunno 2014

Slide 62

- 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)

Unità di misura

Android Mobile Programming - Prof. R. De Prisco

Università di Salerno - Autunno 2014

Slide 63

- 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

Layouts – Linear Layout

Android Mobile Programming - Prof. R. De Prisco

Università di Salerno - Autunno 2014

Slide 64

- 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

Layouts – Linear Layout

Android Mobile Programming - Prof. R. De Prisco

Università di Salerno - Autunno 2014

Slide 65

```
<?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>
```

Layouts – Linear Layout

Android Mobile Programming - Prof. R. De Prisco

Università di Salerno - Autunno 2014

Slide 66

- 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

Layouts – RelativeLayout

Slide 67

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

- 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"

Layouts – RelativeLayout

Slide 68

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

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

Layout - ConstraintLayout

Slide 69

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

- simile al RelativeLayout
 - permette di specificare la posizione attraverso “vincoli”
- Comoda quando si lavora con l’editor grafico
- Si inseriscono dei “vincoli” che legano la posizione del nuovo oggetto rispetto a quelli esistenti

Layouts – Grid View

Slide 70

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

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

Layouts – Grid Layout

Slide 71

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

```
<GridLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#abcdef"
    android:columnCount="3"
    android:rowCount="3" >
    <Button
        android:background="#aa55ee"
        android:id="@+id/button1"
        android:text="Button 1"/>
    <Button
        android:id="@+id/button2"
        android:text="Button 2"/>
    <Button
        android:id="@+id/button3"
        android:layout_column="2"
        android:layout_row="2"
        android:text="Button 3"/>
    <Button
        android:id="@+id/button4"
        android:text="Button 4"/>
</GridLayout>
```

Layouts – List View

Slide 72

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

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

Layouts - widget

Android Mobile Programming - Prof. R. De Friso
Slide 73
Università di Salerno - Autunno 2018

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

Layouts

ANDROID Mobile Programming

Android Mobile Programming - Prof. R. De Friso
Slide 74
Università di Salerno - Autunno 2018

La nostra prima (vera) app

Prima app

Android Mobile Programming - Prof. R. De Friso
Slide 75
Università di Salerno - Autunno 2018

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

Calcolatrice

ANDROID Mobile Programming

Android Mobile Programming - Prof. R. De Friso
Slide 76
Università di Salerno - Autunno 2018

Android Studio Debugger

Copiare un intero progetto

Android Mobile Programming - Prof. R. De Friso
Slide 77
Università di Salerno - Autunno 2018

- 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

Android Studio Debugger

Android Mobile Programming - Prof. R. De Friso
Slide 78
Università di Salerno - Autunno 2018

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

Android Studio Debugger

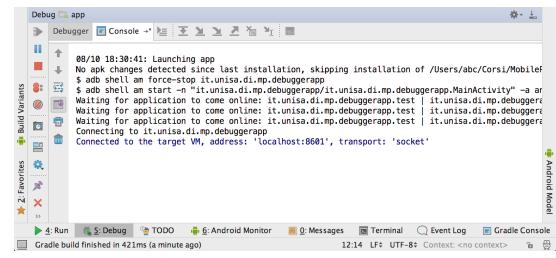
Slide 79

- 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

Android Studio Debugger

Slide 80

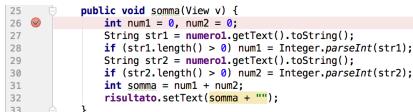
- Si apre la finestra di debug
 - CMD-5



Breakpoints

Slide 81

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



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

Comandi debugger

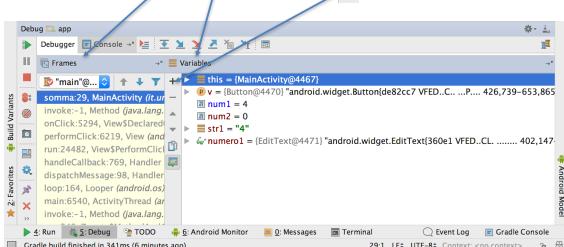
Slide 82

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

Debugger

Slide 83

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



Messaggi di log: Log.X

Slide 84

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

Monitor

Slide 85
Android Mobile Programming - Prof. R. De Prisco
Università di Salerno - Autunno 2018

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

Video capture
Screenshot

Logcat

Slide 86
Android Mobile Programming - Prof. R. De Prisco
Università di Salerno - Autunno 2018

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

ANDROID Mobile Programming

Slide 87
Android Mobile Programming - Prof. R. De Prisco
Università di Salerno - Autunno 2018

Widgets

Widgets

Slide 88
Android Mobile Programming - Prof. R. De Prisco
Università di Salerno - Autunno 2018

Widgets

ListView

Slide 89
Android Mobile Programming - Prof. R. De Prisco
Università di Salerno - Autunno 2018

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

ListView

Slide 90
Android Mobile Programming - Prof. R. De Prisco
Università di Salerno - Autunno 2018

- Lista semplice

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
    <ListView
        android:id="@+id/listView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="" android:padding="10dp"
        android:textSize="22dp"/>
</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/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="" android:padding="10dp"
        android:textSize="22dp"/>
</RelativeLayout>
```

ListView

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

Slide 91

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

ListView

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

Slide 92

- Definire un listener per i click sugli elementi

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

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

ListView

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

Slide 93

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

ListView

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

Slide 94

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

ListView

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

Slide 95

ANDROID Mobile Programming

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

Slide 96

Ciclo di vita

Ciclo di vita delle attività

Android Mobile Programming - Prof. K. De Friso
Università di Salerno - Autunno 2018

Slide 97

- Ogni “Activity” ha un ciclo di vita

Attività non esiste

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

Attività in esecuzione

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

Attività non esiste

- Eseguiti secondo un determinato schema



- Ciclo di vita delle attività**
- Android Mobile Programming - Prof. K. De Friso
Università di Salerno - Autunno 2018
- Slide 100
- Quando l’utente preme il pulsante “Home”
 - vengono chiamate
 - onPause()
 - onStop()
 - Quando si ritorna all’attività
 - vengono chiamate
 - onRestart()
 - onStart()
 - onResume()

- Ciclo di vita delle attività**
- Android Mobile Programming - Prof. K. De Friso
Università di Salerno - Autunno 2018
- Slide 101
- Quando l’utente ruota il dispositivo
 - l’attività viene prima eliminata:
 - onPause()
 - onStop()
 - onDestroy()
 - e poi ricreata:
 - onCreate()
 - onStart()
 - onResume()
 - onDestroy(): perdita dello stato!!!!

- onSaveInstanceState**
- Android Mobile Programming - Prof. K. De Friso
Università di Salerno - Autunno 2018
- Slide 102
- 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");
    }
}
      
```

Main activity

Slide 103

Android Mobile Programming - Prof. R. De Poli

Università di Salerno - Autunno 2016

```
<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>
```

ActivityLifecycle

Ciclo di vita delle attività

Slide 104

Android Mobile Programming - Prof. R. De Poli

Università di Salerno - Autunno 2016

- App Calcolatrice
- Cosa succede se ruotiamo il dispositivo?

Correggere l'errore nell'app Calcolatrice

BackStack

Slide 105

Android Mobile Programming - Prof. R. De Poli

Università di Salerno - Autunno 2016

- 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.

BackStack

Slide 106

Android Mobile Programming - Prof. R. De Poli

Università di Salerno - Autunno 2016

- 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

BackStack

Slide 107

Android Mobile Programming - Prof. R. De Poli

Università di Salerno - Autunno 2016

- 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 schermata
 - 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

BackStack

Slide 108

Android Mobile Programming - Prof. R. De Poli

Università di Salerno - Autunno 2016

- Continuando a premere Back si ritorna all'Home screen

Foreground e background

Android Mobile Programming - Prof. R. De Prisco
Università di Salerno - Autunno 2018
Slide 109

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

Istanze multiple

Android Mobile Programming - Prof. R. De Prisco
Università di Salerno - Autunno 2018
Slide 110

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

MultiActivity

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

VisualizzaMappa

Android Mobile Programming - Prof. R. De Prisco
Università di Salerno - Autunno 2018
Slide 111

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

ANDROID Mobile Programming

Android Mobile Programming - Prof. R. De Prisco
Università di Salerno - Autunno 2018
Slide 112

Intent

Classe Intent

Android Mobile Programming - Prof. R. De Prisco
Università di Salerno - Autunno 2018
Slide 113

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

Intent

Android Mobile Programming - Prof. R. De Prisco
Università di Salerno - Autunno 2018
Slide 114

- Parti principale di un oggetto Intent
 - Action: l'azione da svolgere
 - es. ACTION_VIEW, ACTION_EDIT, ACTION_MAIN
 - Data: i dati su cui operare espressi come URI
 - Uniform Resource Identifier: <schema>:<parte specifica>
 - "http://www.di.unisa.it/"
 - "mailto:robdep@unisa.it"
 - "geo:0,0?via+Posidonia+Salerno+Italy"
 - "tel:+39112223456"
 - "content://com.android.contacts/contacts"
- Esempi di coppie (azione,dati):
 - ACTION_VIEW, content://contacts/people/1
 - ACTION_DIAL, content://contacts/people/1
 - ACTION_DIAL, tel:1112233

 Intent

Android Mobile Programming - Prof. R. De Poli
Università di Salerno - Autunno 2016

Slide 115

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

 Intent

Android Mobile Programming - Prof. R. De Poli
Università di Salerno - Autunno 2016

Slide 116

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

 Intent

Android Mobile Programming - Prof. R. De Poli
Università di Salerno - Autunno 2016

Slide 117

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

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

 Intent

Android Mobile Programming - Prof. R. De Poli
Università di Salerno - Autunno 2016

Slide 118

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

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

 Intent Extras

Android Mobile Programming - Prof. R. De Poli
Università di Salerno - Autunno 2016

Slide 119

- Informazioni “extra”
 - coppie chiave-valore
 - putExtra
 - getExtra

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

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

 Intent Flags

Android Mobile Programming - Prof. R. De Poli
Università di Salerno - Autunno 2016

Slide 120

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

Intent Component

Slide 121

Android Mobile Programming - Prof. R. De Prisco

Università di Salerno - Autunno 2018

- 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(...);
```

Intent

Slide 122

Android Mobile Programming - Prof. R. De Prisco

Università di Salerno - Autunno 2018

Quiz

Slide 123

Android Mobile Programming - Prof. R. De Prisco

Università di Salerno - Autunno 2018

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

Correggere l'errore dovuto alle rotazioni nell'app Quiz

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

Individuare e correggere gli errori e colmare le omissioni

ANDROID Mobile Programming

Slide 124

Android Mobile Programming - Prof. R. De Prisco

Università di Salerno - Autunno 2018

Permessi

Permessi

Slide 125

Android Mobile Programming - Prof. R. De Prisco

Università di Salerno - Autunno 2018

- Android protegge risorse e dati con un meccanismo di permessi di accesso
- Servono a limitare l'accesso a
 - informazioni dell'utente (e.g. i contatti della rubrica)
 - servizi con costi (e.g., invio SMS, chiamate tel., accesso a Internet)
 - Risorse di sistema (e.g., fotocamera, GPS)

Permessi

Slide 126

Android Mobile Programming - Prof. R. De Prisco

Università di Salerno - Autunno 2018

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

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

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

Permessi

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018

Slide 127

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

Permessi normali (API 23, 6.0)

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018

Slide 128

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

Permessi pericolosi (API 23, 6.0)

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018

Slide 129

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

Permessi pericolosi (API 23, 6.0)

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018

Slide 130

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

Threads

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018

Slide 131

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

ANDROID Mobile Programming

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018

Slide 132

Threads

Java Threads

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2016

Slide 133

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

Threads

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2016

Slide 134

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

```

graph TD
    APP[APP (main thread)] -- new --> Thread[Thread]
    Thread -- start() --> Run[run()]
  
```

Threads

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2016

Slide 135

Threads

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2016

Slide 136

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

Threads

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2016

Slide 137

Async task

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2016

Slide 138

- 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)

 **AsyncTask**

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

Slide 139

- Classe Java generica


```
class AsyncTask<Params, Progress, Result> {  
    ...  
}
```
- Parametri
 - Params: tipo (di dati) per il lavoro che deve svolgere il background thread
 - Progress: tipo (di dati) usato per lo stato di avanzamento
 - Result: tipo (di dati) per il risultato del task

 **AsyncTask.execute()**

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

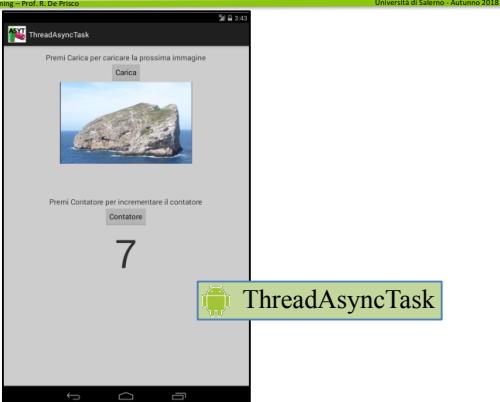
Slide 140

- 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

 **AsyncTask**

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

Slide 141



The screenshot shows a mobile application interface. It starts with a splash screen featuring a large image of a rocky island in the sea. Below the image is a button labeled "Carica". The next screen displays a counter with the number "7" and a button labeled "Contatore". At the bottom of the screen, there is a navigation bar with icons for back, forward, and home.

 **ThreadAsyncTask**

 **ANDROID Mobile Programming**

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

Slide 142

Fragments

 **Fragments**

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

Slide 143

- 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

 **Fragments**

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

Slide 144

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

Fragments

Android Mobile Programming - Prof. R. De Poli
Slide 145
Università di Salerno - Autunno 2016

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

Fragments

Android Mobile Programming - Prof. R. De Poli
Slide 146
Università di Salerno - Autunno 2016

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

Fragments

Android Mobile Programming - Prof. R. De Poli
Slide 147
Università di Salerno - Autunno 2016

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

Fragments

Android Mobile Programming - Prof. R. De Poli
Slide 148
Università di Salerno - Autunno 2016

Creare frammenti

Android Mobile Programming - Prof. R. De Poli
Slide 149
Università di Salerno - Autunno 2016

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

Creare frammenti

Android Mobile Programming - Prof. R. De Poli
Slide 150
Università di Salerno - Autunno 2016

- 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 linflate di un file di layout
 - onPause()**
 - il primo metodo chiamato quando il frammento viene eliminato (si dovrebbero rendere permanenti eventuali cambiamenti altrimenti si perdono)

 **Creare frammenti**

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018
Slide 151

```
public static class ExampleFragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup view,
                           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
 - `view` è un oggetto che serve a specificare i parametri di layout

 **Fragments**

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018
Slide 152

```
public View inflate (int resource, ViewGroup root, boolean attachToRoot)
```

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

Parameters

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

Returns

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

 **Creare frammenti**

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018
Slide 153

- Un frammento può essere inserito staticamente nel layout

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

 **Creare frammenti**

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018
Slide 154

- Oppure dinamicamente a runtime

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

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

 **Gestire i frammenti**

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018
Slide 155

- Usiamo il `FragmentManager`

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

- Iniziamo una transazione

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

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

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

 **Gestire i frammenti**

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018
Slide 156

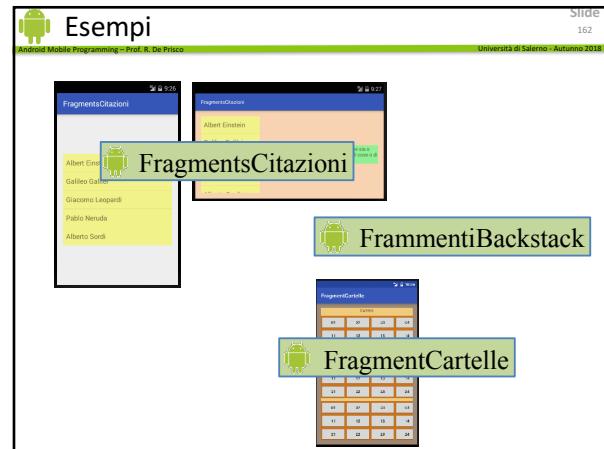
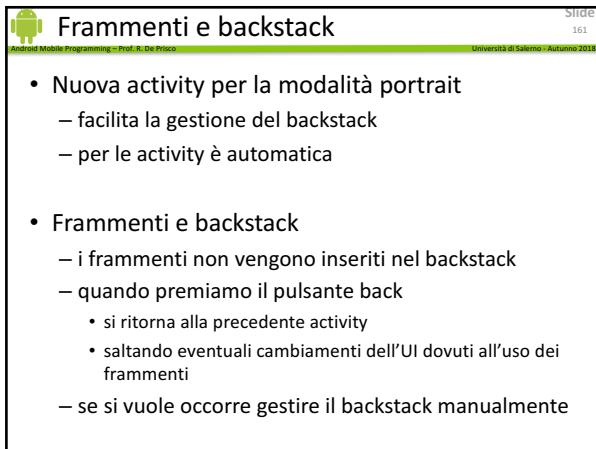
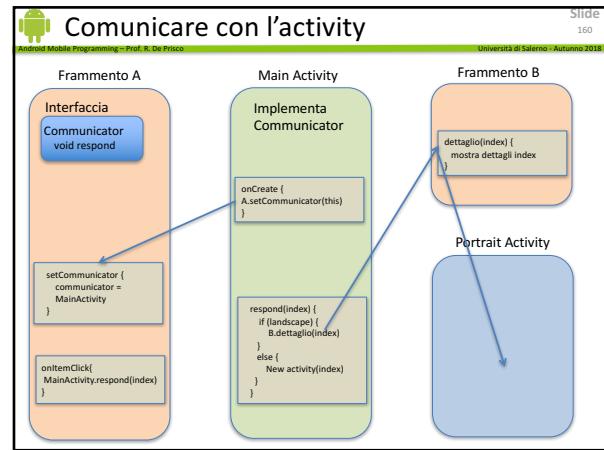
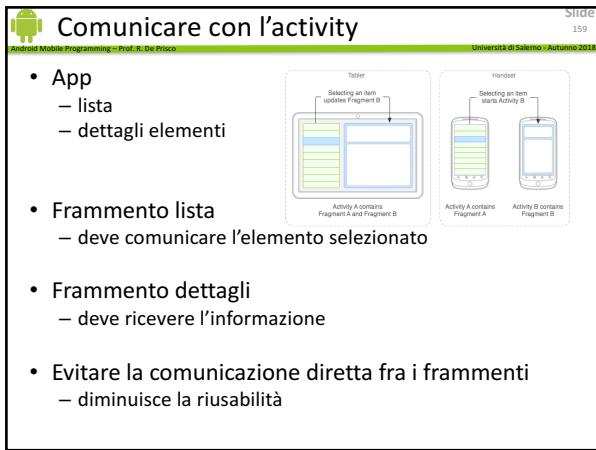
- `addToBackStack()`
 - per inserire i cambiamenti nel backstack
- Il backstack considera solo le activity
 - dobbiamo gestire manualmente i frammenti
- Se non chiamiamo `addToBackStack`
 - quando premiamo back “salteremo” i cambiamenti fatti con i frammenti
 - non è quello che l'utente si aspetta



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);
    }
    ...
}
```



 **Frammenti**

Slide 163
Università di Salerno - Autunno 2018

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

 **ANDROID Mobile Programming**

Slide 164
Università di Salerno - Autunno 2018

Networking

 **Networking**

Slide 165
Università di Salerno - Autunno 2018

- Socket
 - Java.net
- HTTP
 - org.apache
 - HttpRequest
 - Httpresponse
- Data formats
 - JSON, XML

 **Networking**

Slide 166
Università di Salerno - Autunno 2018

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

 **Networking**

Slide 167
Università di Salerno - Autunno 2018

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

 **Networking**

Slide 168
Università di Salerno - Autunno 2018

- Scrivere nel socket
- PrintWriter out =


```
new PrintWriter(
    new BufferedWriter(
        new OutputStreamWriter(
            socket.getOutputStream())),
    true); //Autoflush
```
- out.println(strToSend);

Networking

Android Mobile Programming - Prof. R. De Friso

Slide 169

Università di Salerno - Autunno 2018

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

Localhost

Android Mobile Programming - Prof. R. De Friso

Slide 170

Università di Salerno - Autunno 2018

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

Networking

Android Mobile Programming - Prof. R. De Friso

Slide 171

Università di Salerno - Autunno 2018

SocketRaw

Host Address: 192.41.218.17 Port: 12345

To Send: einstein

Send

Clear Response

Response:

HTTP/1.1 200 OK

GET /live/SerieA/classifica.shtml HTTP/1.1

Host: www.corrieredellosport.it

SocketRawProgressBar

URL e HTTP

Android Mobile Programming - Prof. R. De Friso

Slide 172

Università di Salerno - Autunno 2018

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

HTTP

Android Mobile Programming - Prof. R. De Friso

Slide 173

Università di Salerno - Autunno 2018

- classe AndroidHttpClient
- classe HttpGet
- classe responseHandler
- ci permettono facilmente di
 - stabilire una connessione HTTP
 - inviare una richiesta GET
 - leggere la risposta

Networking

Android Mobile Programming - Prof. R. De Friso

Slide 174

Università di Salerno - Autunno 2018

SocketURL

URL: http://www.corrieredellosport.it/live/SerieA/classifica.shtml

Send

Clear Response

Response:

HTTP/1.1 200 OK

SocketHTTP

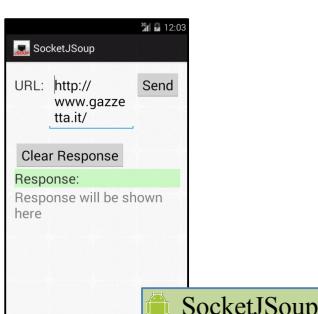
 Documenti HTML
Android Mobile Programming - Prof. R. De Friso
Slide 175
Università di Salerno - Autunno 2018

- Dati in documenti HTML
 - difficile estrarli
- Esistono delle librerie che implementano il parsing di documenti 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

 Jsoup
Android Mobile Programming - Prof. R. De Friso
Slide 176
Università di Salerno - Autunno 2018

- 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]");

 Networking
Android Mobile Programming - Prof. R. De Friso
Slide 177
Università di Salerno - Autunno 2018



 ANDROID
Mobile Programming
Android Mobile Programming - Prof. R. De Friso
Slide 178
Università di Salerno - Autunno 2018

Data Storage

 Data Storage
Android Mobile Programming - Prof. R. De Friso
Slide 179
Università di Salerno - Autunno 2018

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

 SharedPreferences
Android Mobile Programming - Prof. R. De Friso
Slide 180
Università di Salerno - Autunno 2018

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

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

SharedPreferences

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018

Slide 181

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

File

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018

Slide 182

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

File

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018

Slide 183

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

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

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

File

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018

Slide 184

- 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.

File

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018

Slide 185

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

File temporanei

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018

Slide 186

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

File su External Storage

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018
Slide 187

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

External Storage

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018
Slide 188

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

- Permesso Write include il permesso Read

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

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

External Storage

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018
Slide 189

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

Condividere file con altre app

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018
Slide 190

- `getExternalStoragePublicDirectory(type)`
 - type: DIRECTORY_PICTURES, DIRECTORY_MUSIC, DIRECTORY_RINGTONES, ...
- Esempio: metodo che crea una nuova dir per delle foto nella dir pubblica delle immagini

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

SQL – Quick tutorial

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018
Slide 191

- 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	30
1367288	Attilio	Bianchi	Algoritmi	25
5267712	Valentino	Rossi	MotoGP	30
7126714	Giuseppe	Verdi	Sicurezza	22
1562689	Marco	Arancione	MP	30

SQL – Quick tutorial

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018
Slide 192

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

 SQL – Quick tutorial

Slide 193
Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

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

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

 SQL – Quick tutorial

Slide 194
Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

- Esempi di inserimento

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

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

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

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

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

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

 SQL – Quick tutorial

Slide 195
Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

- Tavola prodotta

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

 SQL – Quick tutorial

Slide 196
Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

- 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	Carla	23	Palermo	2000.00
4	Maria	25	Roma	6500.00
5	Pasquale	27	Firenze	8500.00
6	Renato	22	Venezia	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;
```

 SQL – Quick tutorial

Slide 197
Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

- Esempi di update

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

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

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

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

 Database

Slide 198
Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

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

 Database

Slide 199
Università di Salerno - Autunno 2018

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

 Database

Slide 200
Università di Salerno - Autunno 2018

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

    MyOpenHelper(Context context) {
        super(context, TABLE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(CREATE_CMD);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // override necessario
    }
}
```

 Data Storage

Slide 201
Università di Salerno - Autunno 2018

 ANDROID Mobile Programming

Slide 202
Università di Salerno - Autunno 2018

Grafica

 Grafica

Slide 203
Università di Salerno - Autunno 2018

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

 Grafica

Slide 204
Università di Salerno - Autunno 2018

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

 GraficaSimpleImg

Animazioni

Slide 205
Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018

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

Animazioni

Slide 206
Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018

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



GraficaImgAnim

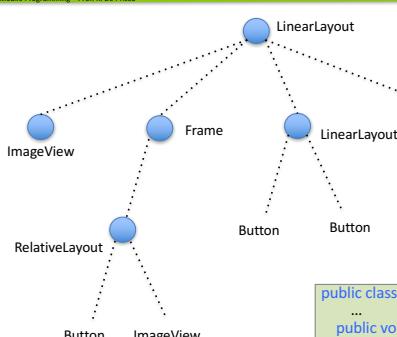
Custom Views

Slide 207
Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018

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

Albero delle View

Slide 208
Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018



```
public class View {
    ...
    public void onMeasure(...)
    public void onLayout(...)
}
...
```

Meccanismo di layout

Slide 209
Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018

Diagram illustrating the layout process:

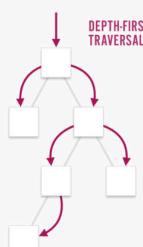
```
Measure --> Layout --> Draw
```

- 3 Fasi
 - Misura
 - Posizionamento
 - Disegno

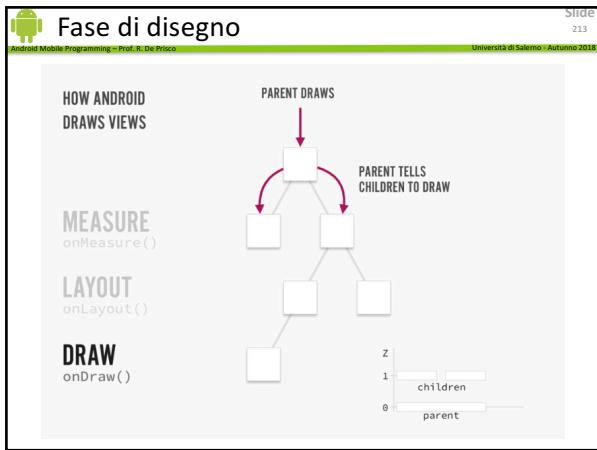
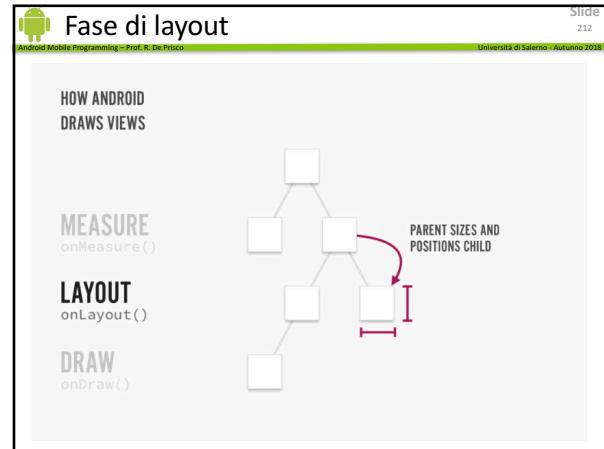
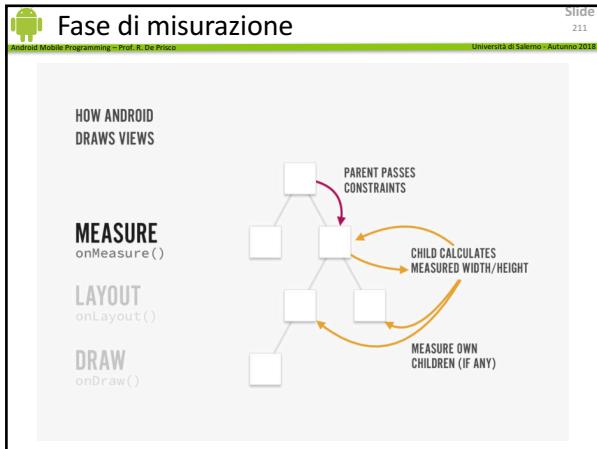
Albero delle views

Slide 210
Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018

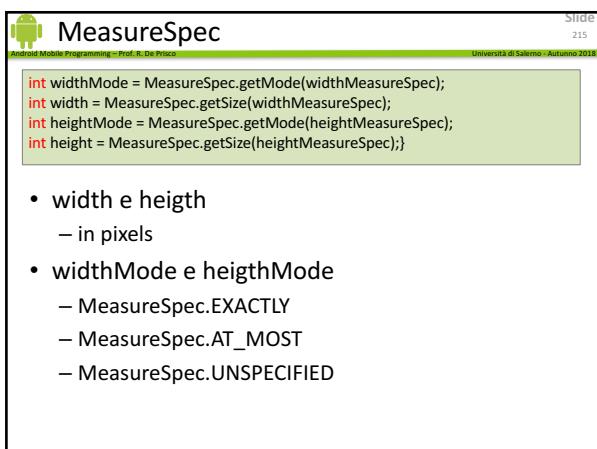
HOW ANDROID DRAWS VIEWS
INFLATION/INSTANTIATION
MEASURE
onMeasure()
LAYOUT
onLayout()
DRAW
onDraw()



DEPTH-FIRST TRAVERSAL



- Fase di misurazione**
- Android Mobile Programming - Prof. R. De Prisco
Università di Salerno - Autunno 2016
- Slide 214
- “measured” size (width&height)
 - quanto grande la view vorrebbe essere
 - size reale
 - quanto grande la view sarà in realtà
 - Approccio top-down nell’albero
 - ogni view chiede ai suoi figli quanto vorrebbero essere grandi
 - classe MeasureSpec



- Fase di misurazione**
- Android Mobile Programming - Prof. R. De Prisco
Università di Salerno - Autunno 2016
- Slide 216
- Ogni view può esprimere la propria preferenza usando le opzioni della classe ViewGroup.LayoutParams
 - Un numero (di pixel)
 - MATCH_PARENT
 - WRAP_CONTENT
 - La misurazione avviene nel metodo onMeasure

Fase di misurazione

Slide 217
Università di Salerno - Autunno 2018
Android Mobile Programming - Prof. R. De Frisco

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

Fase di layout

Slide 218
Università di Salerno - Autunno 2018
Android Mobile Programming - Prof. R. De Frisco

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

Fase di disegno

Slide 219
Università di Salerno - Autunno 2018
Android Mobile Programming - Prof. R. De Frisco

- 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.

Meccanismo di layout

Slide 220
Università di Salerno - Autunno 2018
Android Mobile Programming - Prof. R. De Frisco

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

onMeasure()

Slide 221
Università di Salerno - Autunno 2018
Android Mobile Programming - Prof. R. De Frisco

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

- onMeasure potrebbe essere chiamata varie volte!
- gli "int" contengono anche dei bit addizionali

Layout

Slide 222
Università di Salerno - Autunno 2018
Android Mobile Programming - Prof. R. De Frisco

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

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

Disegnare nel canvas

Slide 223
Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

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

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

Esempi

Slide 224
Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

GraficaCustomWidget

GraficaCanvas

Multitouch

Slide 225
Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

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

Multitouch

Slide 226
Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

- 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 singolo pointer usando un indice
- Ogni pointer ha un ID unico per tutto il tempo in cui esiste
 - L'indice di un MotionEvent multiplo NON è il pointer ID
 - il pointer ID è costante
 - l'indice può cambiare per eventi successivi

Multitouch

Slide 227
Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

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

Multitouch

Slide 228
Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

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

 Multitouch

Android Mobile Programming - Prof. R. De Frisco

Slide 229
Università di Salerno - Autunno 2018

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

 Multitouch

Android Mobile Programming - Prof. R. De Frisco

Slide 230
Università di Salerno - Autunno 2018

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

 Multitouch

Android Mobile Programming - Prof. R. De Frisco

Slide 231
Università di Salerno - Autunno 2018

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

 Multitouch

Android Mobile Programming - Prof. R. De Frisco

Slide 232
Università di Salerno - Autunno 2018

- 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.setOnTouchListener`
 - `View.setOnTouchListener()`

 Multitouch

Android Mobile Programming - Prof. R. De Frisco

Slide 233
Università di Salerno - Autunno 2018

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

 Multitouch

Android Mobile Programming - Prof. R. De Frisco

Slide 234
Università di Salerno - Autunno 2018

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

Multitouch

Android Mobile Programming - Prof. R. De Friso

Slide 235

Università di Salerno - Autunno 2018

MultitouchExample

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

GestureDetector

Android Mobile Programming - Prof. R. De Friso

Slide 236

Università di Salerno - Autunno 2018

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

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

GestureDetector

Android Mobile Programming - Prof. R. De Friso

Slide 237

Università di Salerno - Autunno 2018

- 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

ViewAnimator e ViewFlipper

Android Mobile Programming - Prof. R. De Friso

Slide 238

Università di Salerno - Autunno 2018

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

GestureDetector

Android Mobile Programming - Prof. R. De Friso

Slide 239

Università di Salerno - Autunno 2018

GestureFlip

Implementare anche il fling per il decremento

ANDROID Mobile Programming

Android Mobile Programming - Prof. R. De Friso

Slide 240

Università di Salerno - Autunno 2018

Media player

MediaPlayer

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

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

MediaPlayer

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

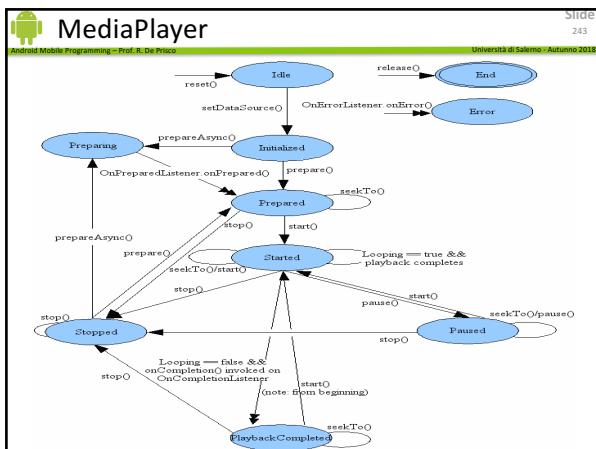
- Play di un file in res/raw

```
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



MediaPlayer

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

- Rilasciare la risorsa

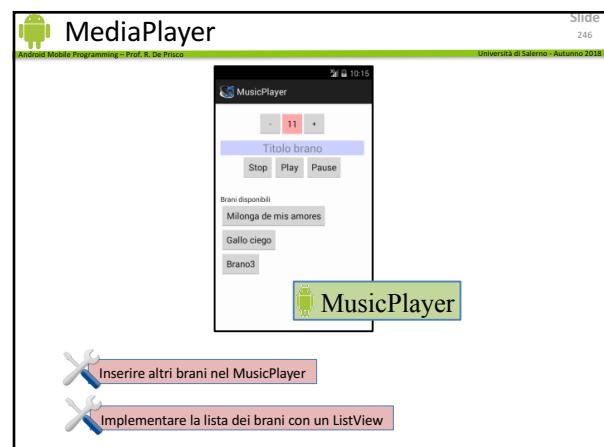
```
mediaPlayer.release();
mediaPlayer = null;
```

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

MediaPlayer

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018

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





ANDROID Mobile Programming

Android Mobile Programming - Prof. R. De Friso

Sensori

247



Sensori

Android Mobile Programming - Prof. R. De Friso

Università di Salerno - Autunno 2018

- Molti smartphone, 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"
 - accuracy dipende dalla qualità



Sensori

Android Mobile Programming - Prof. R. De Friso

Università di Salerno - Autunno 2018

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

249



Sensori

Android Mobile Programming - Prof. R. De Friso

Università di Salerno - Autunno 2018

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

Sensore	Tipo	Descrizione
TYPE_ACCELEROMETER	Hw	Misura le forze in m/s ² 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

250

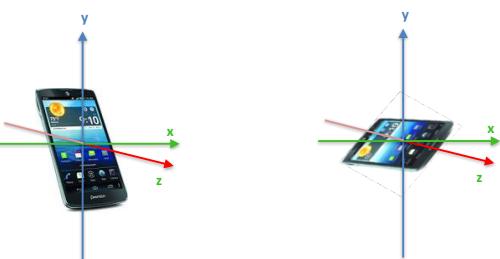


Sensori

Android Mobile Programming - Prof. R. De Friso

Università di Salerno - Autunno 2018

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



251



Sensori

Android Mobile Programming - Prof. R. De Friso

Università di Salerno - Autunno 2018

- 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  
}
```

252

Sensori

Android Mobile Programming - Prof. R. De Prisco

Slide 253

```

@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

Android Mobile Programming - Prof. R. De Prisco

Slide 254

```

@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 - Prof. R. De Prisco

Slide 255

Accelerometro

ANDROID Mobile Programming

Android Mobile Programming - Prof. R. De Prisco

Slide 256

Notifiche

Notifications

Android Mobile Programming - Prof. R. De Prisco

Slide 257

- Notifiche: informazioni all'utente al di fuori dell'interfaccia grafica dell'app
 - Toast
 - Dialog
 - Notification Area (Status Bar)

Notifications

ANDROID Mobile Programming

Android Mobile Programming - Prof. R. De Prisco

Slide 258

Alarms

 Alarms

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2016
Slide 259

- Permettono di eseguire intent in funzione di specifici eventi
 - intent lanciati in base al tempo
- Un'applicazione che usa un alarm riesce ad eseguire porzioni di codice anche se l'applicazione è terminata
- 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

 Alarms

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2016
Slide 260

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

 Alarms

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2016
Slide 261

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

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

 Tipi di Alarm

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2016
Slide 262

- ELAPSED_REALTIME
 - lancia il "pending" intent dopo un determinato tempo (ma non fa il wakeup della device)
- ELAPSED_REALTIME_WAKEUP
 - Se la device non è attiva fa il wakeup e lancia il "pending" intent
- RTC
 - lancia il pending event ad un determinato orario
- RTC_WAKEUP
 - come prima ma fa il wakeup se serve

 ANDROID
Mobile Programming

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2016
Slide 263

Content Providers, Broadcast, Services

 Oltre le Activity

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2016
Slide 264

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

 **Broadcasts**

Android Mobile Programming - Prof. R. De Frisco

Slide 265

Università di Salerno - Autunno 2018

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.

 **Content providers**

Android Mobile Programming - Prof. R. De Frisco

Slide 266

Università di Salerno - Autunno 2018

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.

 **Services**

Android Mobile Programming - Prof. R. De Frisco

Slide 267

Università di Salerno - Autunno 2018

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.

 **Broadcast ed eventi**

Android Mobile Programming - Prof. R. De Frisco

Slide 268

Università di Salerno - Autunno 2018

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

 **BroadcastReceiver**

Android Mobile Programming - Prof. R. De Frisco

Slide 269

Università di Salerno - Autunno 2018

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

 **BroadcastReceiver**

Android Mobile Programming - Prof. R. De Frisco

Slide 270

Università di Salerno - Autunno 2018

- Il ricevitore riceve l’Intent tramite il metodo
 - `onReceive(Context c, Intent i)`
- Riassumendo
 1. Il “ricevitore” si “registra” usando `registerReceiver()` (disponibile nel `LocalBroadcastManager` o nel `Context`)
 2. L’evento viene creato (da qualche altra componente del sistema)
 3. Android notifica il ricevitore chiamando `onReceive()`

Registrazione BroadcastReceiver

Slide 271

Android Mobile Programming - Prof. R. De Prisco
Università di Salerno - Autunno 2018

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

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

Registrazione BroadcastReceiver

Slide 272

Android Mobile Programming - Prof. R. De Prisco
Università di Salerno - Autunno 2018

- Se la registrazione è statica
 - il ricevitore viene registrato durante il Boot del sistema (oppure quando l'app viene installata)
- Se la registrazione è dinamica
 - il ricevitore viene registrato quando si chiama
 - LocalBroadcastManager.registerReceiver()
 - per i broadcast locali all'app
 - Context.registerReceiver()
 - per i broadcast system-wide
- è possibile anche revocare la registrazione usando unregisterReceiver()

Spedizione broadcast

Slide 273

Android Mobile Programming - Prof. R. De Prisco
Università di Salerno - Autunno 2018

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

sendBroadcast(Intent i) è disponibile sia nel LocalBroadcastManager che nel Context. Chiaramente 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.

Esempi

Slide 274

Android Mobile Programming - Prof. R. De Prisco
Università di Salerno - Autunno 2018

BCastLocal

BCastGlobal

- ACTION_TIME_CLICK
 - Intent inviato ogni minuto

BroadcastReceiver

Slide 275

Android Mobile Programming - Prof. R. De Prisco
Università di Salerno - Autunno 2018

- 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

ContentProvider

Slide 276

Android Mobile Programming - Prof. R. De Prisco
Università di Salerno - Autunno 2018

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

 ContentProvider

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018
Slide 277

- Per usare il resolver occorre recuperare un suo riferimento chiamando
 - Context.getContentResolver()
- ContentProviders standard
 - Browser (info su bookmarks, history)
 - Call Log (info sulle chiamate)
 - Contact (info sui contatti presenti in rubrica)
 - Media (lista dei file multimediali utilizzabili)
 - UserDictionary (lista delle parole digitate)
 - ... molti altri

 ContentProvider

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018
Slide 278

- I dati contenuti in un provider sono memorizzati in tabelle
- Gli utenti possono far riferimento ad uno specifico ContentProvider usando un URI
- URI: content://authority/path/id
 - authority: specifica il content provider
 - path: specifica la tabella
 - id: specifica un particolare record

 ContentProvider

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018
Slide 279

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

 ContentProvider

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018
Slide 280

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


```
Cursor query(Uri uri,
           String[] projection \\\ colonne
           String selection \\\ SQL selection
           String[] args \\\ SQL args
           String sortOrder) \\\ ordinamento
```
- Restituisce un Cursor che ci permette di iterare sull’insieme di record restituiti dalla query

 ContentProvider

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018
Slide 281

- Esempio: leggere la rubrica


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



 ContentProvider

 Services

Android Mobile Programming - Prof. R. De Friso
Università di Salerno - Autunno 2018
Slide 282

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

Services

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018
Slide 283

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

Services

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018
Slide 284

- Una volta partito, il Service può continuare la sua esecuzione fino a che la device è accesa
 - potrebbe anche essere interrotto se occorrono le risorse che esso usa
 - potrebbe anche terminare volontariamente
- Nell'utilizzo tipico un Service fatto partire da un'app termina la propria esecuzione dopo aver eseguito l'operazione richiesta
- per default, il Service gira nel main thread dell'app che lo ha fatto partire
 - in alcuni casi deve essere esplicitamente fatto girare su un thread separato

Services

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018
Slide 285

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

Services

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018
Slide 286

- 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>
```

Services

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018
Slide 287

Services

Android Mobile Programming - Prof. R. De Frisco
Università di Salerno - Autunno 2018
Slide 288

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



A screenshot of the Google Play Store listing for the app "Cartellone Tombola Zappetelle". The title "Buon Natale!" is at the top. The app icon shows two crossed sticks. The description says "Cartellone Tombola Zappetelle". It has a rating of 2 stars and 2 reviews. The developer is "ApproidLab". The app is free and has been published on "05/gen/2016". It includes features like "Annuncia numero", "Annuncia smorfia", "Risposta parabolico", "Animazione panariello", and "Sensore miscchia". Two screenshots are shown: one showing a bingo card with numbers 1-20 and another showing a face with a smiley expression. The bottom part of the screen shows other apps by the developer: "Cartelle Tombola Zappetelle" and "Cartellone Tombola Zappetelle".