Installare l'SDK

Introduzione

Per sviluppare una applicazione per Android è necessario scaricare l'SDK dal sito ufficiale.

SDK sta per Software Development Kit e contiene una serie di tool ed un emulatore per testare le applicazioni create direttamente sul proprio PC.

Prerequisiti 10/2012

- Android può essere installato su Windows (XP,Vista, 7, 8), Linux (Ubuntu dalla 8.04) e Mac OS X (dalla 10.5.8/x86).
- Bisogna avere installato il Java Development Kit (JDK) versione 6 (non è sufficiente il runtime-environment JRE)
- L'IDE consigliato è Eclipse (dalla 3.6.2 Helios o successivo) + JDT plugin
- Infine, è necessario installare l'ADT (Android Development Tools) plugin
- Ultimamente è possibile utilizzare Android Studio

Download

http://developer.android.com/sdk

Developers | Design | Develop | Distribute

Training | API Guides | Reference | Tools | Google Services | Samples

- Developer Tools
- Download
  - Setting Up the ADT Bundle
  - Setting Up an Existing IDE
  - Android Studio
  - Exploring the SDK
  - Download the NDK
- Workflow
- Support Library
- Tools Help
- Revisions
- Samples
- ADK

Get the Android SDK

The Android SDK provides you the API libraries and developer tools necessary to build, test, and debug apps for Android.

If you're a new Android developer, we recommend you download the ADT Bundle to quickly start developing apps. It includes the essential Android SDK components and a version of the Eclipse IDE with built-in **ADT (Android Developer Tools)** to streamline your Android app development.

Download the SDK
ADT Bundle for Windows

With a single download, the ADT Bundle includes everything you need to begin developing apps:

- Eclipse + ADT plugin
- Android SDK Tools
- Android Platform-tools
- The latest Android platform
- The latest Android system image for the emulator

Android Studio Early Access Preview

A new Android development environment called Android Studio, based on IntelliJ IDEA, is now available as an **early access preview**. For more information, see Getting Started with Android Studio.

If you prefer to use an existing version of Eclipse or another IDE, you can instead take a more customized approach to installing the Android SDK. See the following instructions:

ADT Bundle – SDK

- E' possibile installare l'SDK partendo da una versione di Eclipse precedentemente installata. In questo caso sono necessari alcuni passi di configurazione dell'ambiente
- Oppure, è possibile scaricare direttamente l'ADT Bundle che consente di inziare a lavorare senza alcuna configurazione (consigliato)

ADT Bundle

- Dopo aver scaricato il file zip dell'ADT Bundle è necessario scompattarlo e salvarlo in una cartella del tipo:

C:\development\

- Nessun altro passo e necessario e si può iniziare a creare il primo progetto Android

Installare l'SDK 1/2

- In realtà l'SDK non prevede una installazione vera e propria: è sufficiente scompattare il file ZIP in una cartella, il cui path sarà utilizzato in Eclipse. Per es.:
  - Windows: C:\apps\android-sdk-windows
  - Linux: /home/username/androidsdk_linux_86
  - Mac OS X: /Users/username/android_sdk_86
- Per Windows è disponibile un installer.
- A questo punto è possibile già sviluppare, debuggare, interagire con l'emulatore da riga di comando.

Installare l'SDK 2/2

- Se si vuole lavorare a riga di comando è necessario aggiungere alla variabile di ambiente PATH il percorso completo della sottodirectory ”tools” e ”platform-tools”

Installare l'SDK (MAC OS X) 1/2

- Il file in cui indicare il path è ~/.bash_profile
- Per verificare se esiste è sufficiente aprire un Terminale nella home e digitare ls -a
- Se esiste editarlo con il comando:

open -a TextEdit .bash_profile

- Aggiungendo le seguenti righe:

export PATH=$PATH:**<sdk>/**platform-tools

export PATH=$PATH:**<sdk>/**tools

Installare l'SDK (MAC OS X) 2/2

- Altrimenti, se non esiste, nella home digitare:

echo 'export PATH=<sdk>/plaftom-tools:$PATH' >> ~/.bash_profile

echo 'export PATH=<sdk>/tools:$PATH' >> ~/.bash_profile

Installare il plugin ADT

- Se invece si preferisce lavorare all'interno di un ambiente integrato di sviluppo è possibile installare il plugin ADT (Android Development Tools) all'interno dell'IDE Eclipse.

- Per fare ciò all'interno di Eclipse bisogna selezionare ”Help | Install New Software...” ed aggiungere il seguente sito:

https://dl-ssl.google.com/android/eclipse/

Configurare il plugin

- L'unica configurazione richiesta dal plugin ADT per funzionare correttamente all'interno di Eclipse consiste nel fornirgli il percorso completo all'interno del quale è stato scompattato l'SDK.

- Per fare ciò in Eclipse bisogna selezionare:
  - MAC: ”Eclipse > Preferences..”
  - Altri: ”Window > Preferences... ”

- E poi ”Browse...” e selezionare la cartella dell'SDK utilizzata al passo precedente per la scompattazione.

Aggiornare il plugin

Periodicamente potrebbero essere rilasciati aggiornamenti del plugin ADT per Eclipse.

Si suggerisce pertanto di controllare di tanto in tanto selezionando ”Help | Software Updates” e provando ad aggiornare il plugin di Android.

Aggiornare l'SDK

Mediante l'Android SDK Manager (che può essere richiamato sia dall'ambiente Eclipse, sia da riga di comando (Android) è possibile seguire gli aggiornamenti dell'SDK, nonchè installare esempi, documentazione e le importanti Google API.

Inoltre l'emulatore deve essere lanciato almeno una volta a riga di comando con l'opzione ”-wipe-data” in modo da cancellare i dati relativi alle esecuzioni precedenti.

Test installazione 1/3

Tramite riga di comando digitiamo il comando:
android

Sarà visualizzato l' Android SDK Manager.
Avremmo ottenuto lo stesso risultato utilizzando Eclipse mendiante la voce di menu:

Window>Android SDK Manager

oppure sulla relativa icona.

Nota: le Google API

Test installazione 2/3

Tramite riga di comando digitiamo il comando:
android avd

Sarà visualizzato l' Android Virtual Devices Manager (AVD Manager).
Avremmo ottenuto lo stesso risultato utilizzando Eclipse mendiante la voce di menu:

Window>Android AVD Manager

oppure sulla relativa icona.

Test installazione 3/3

- Nel caso in cui l'installazione sia stata effettuata tramite l'ADT Bundle, sarà sufficiente andare nella cartella

  adt-bundle-<os_platform>/eclipse

- Ed avviare eclipse

Conclusioni

In questa lezione abbiamo visto come installare, configurare ed aggiornare il Software Development Kit (SDK) di Android ed il relativo plugin per l'IDE Eclipse con e senza l'ADT Bundle.

Strumenti di sviluppo

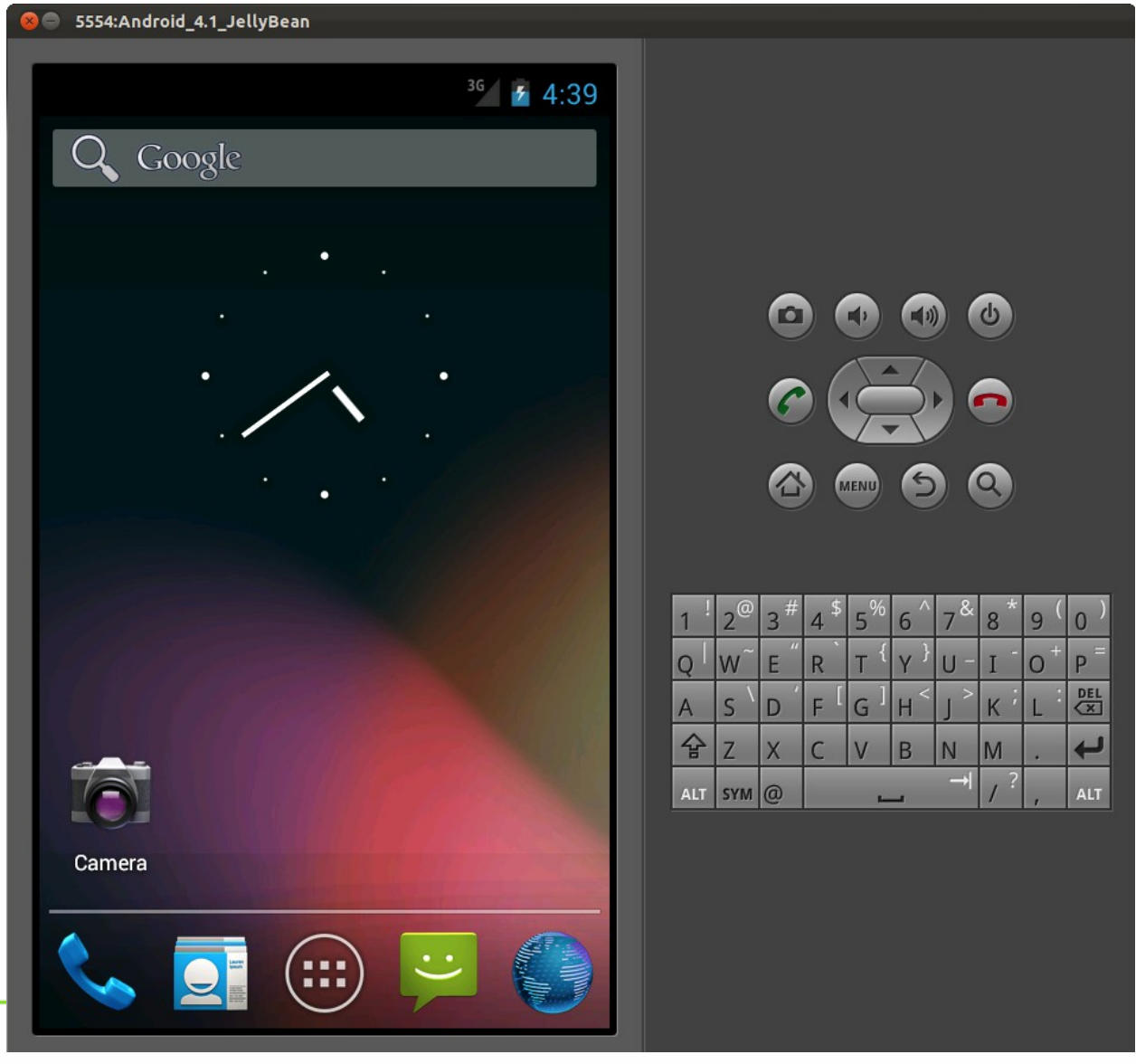
Introduzione

In questa lezione verranno elencati e descritti brevemente i principali strumenti di sviluppo forniti con l'SDK di Android, in modo da prendere dimestichezza con ognuno di essi.

Emulatore Android

- L'SDK include l'emulatore di un dispositivo mobile in grado di girare su un comune PC.

- L'emulatore permette di testare le applicazioni sviluppate senza disporre del dispositivo fisico.



Caratteristiche dell'emulatore

- L'emulatore emula il comportamento hardware e software di un tipico smartphone, con la sola eccezione di non essere in grado di effettuare vere telefonate.
- In compenso le sue capacità di debugging fanno sì che tramite una console testuale sia possibile leggere il log del kernel e simulare l'arrivo di interruzioni (come ad esempio la ricezione di un SMS).

Funzionamento interno

L'emulatore Android è basato su QEMU (un noto emulatore open-source) e fornisce l'emulazione di:

- una CPU ARMv5 e la corrispondente MMU (Memory Managment Unit)
- un display LCD con 64K colori
- una o più tastiere
- un chip audio con input e output
- l'accesso a partizioni sulla memoria Flash
- un modem GSM, inclusa una SIM simulata
- una video-camera, utilizzando la webcam del pc

http://developer.android.com/guide/developing/devices/emulator.html

Screen Configuration

	Low density (120), *ldpi*	Medium density (160), *mdpi*	High density (240), *hdpi*	Extra high density (320), *xhdpi*
Small screen	**QVGA (240x320)**		480x640	
Normal screen	**WQVGA400 (240x400)** **WQVGA432 (240x432)**	**HVGA (320x480)**	**WVGA800 (480x800)** **WVGA854 (480x854)** 600x1024	640x960
Large screen	**WVGA800 (480x800)** **WVGA854 (480x854)**	**WVGA800 (480x800)** **WVGA854 (480x854)** 600x1024		
Extra Large screen	1024x600	**WXGA (1280x800)** 1024x768 1280x768	1536x1152 1920x1152 1920x1200	2048x1536 2560x1536 2560x1600

Emulator Keyboard Key

Menu (left softkey)	F2 *or* Page-up button
Star (right softkey)	Shift-F2 *or* Page Down
Back	ESC
Call/dial button	F3
Hangup/end call button	F4
Search	F5
Power button	F7
Audio volume up button	KEYPAD_PLUS, Ctrl-5
Audio volume down button	KEYPAD_MINUS, Ctrl-F6
Camera button	Ctrl-KEYPAD_5, Ctrl-F3
Switch to previous layout orientation (for example, portrait, landscape)	KEYPAD_7, Ctrl-F11
Switch to next layout orientation (for example, portrait, landscape)	KEYPAD_9, Ctrl-F12
Toggle cell networking on/off	F8
Toggle code profiling	F9 (only with `-trace` startup option)
Toggle fullscreen mode	Alt-Enter
Toggle trackball mode	F6
Enter trackball mode temporarily (while key is pressed)	Delete
DPad left/up/right/down	KEYPAD_4/8/6/2
DPad center click	KEYPAD_5
Onion alpha increase/decrease	KEYPAD_MULTIPLY(*) / KEYPAD_DIVIDE(/)

Attività 1/2

- Creare un nuovo AVD (es. Android2.3.3)
- Avviarlo (da Eclipse e riga di comando)

  emulator @Android2.3.3
- Descrivere le principali caratteristiche
- Utilizzare le keyboard Key
- Chiusura
- Individuare l'immagine e i file di configurazione dell'AVD creato

Attività 2/2

- Interagire con l'AVD utilizzando ad es.:
  - Chiamata
  - SMS
  - Batteria
- Tramite DDMS
- Tramite riga di comando:

  telnet localhost 5554

  gsm call 12345678

  power status full

  sms send 12345678 Ciao!

L'Android Debug Bridge (o ADB)

- Permette inoltre di accedere a riga di comando al dispositivo virtuale e reale e di metterlo in collegamento con un debugger standard.
- Permette di installare i file .apk di una applicazione sull'emulatore o sul dispositivo.

  Nota: il file apk si trova all'interno della cartella \bin

```
adb install NomeApp.apk
```

Attività 1/2

- Visualizzare gli AVD o dispositivi reali connessi:

  adb devices

- Aprire una shell direttamente sull'emulatore

  adb shell

- adb push nome_file_da_mandare <percorso-in-emulatore>/nome_file_destinazione
- adb pull <percorso-in-emulatore>/nome_file nome_file_destinazione
- adb kill-server
- adb start-server

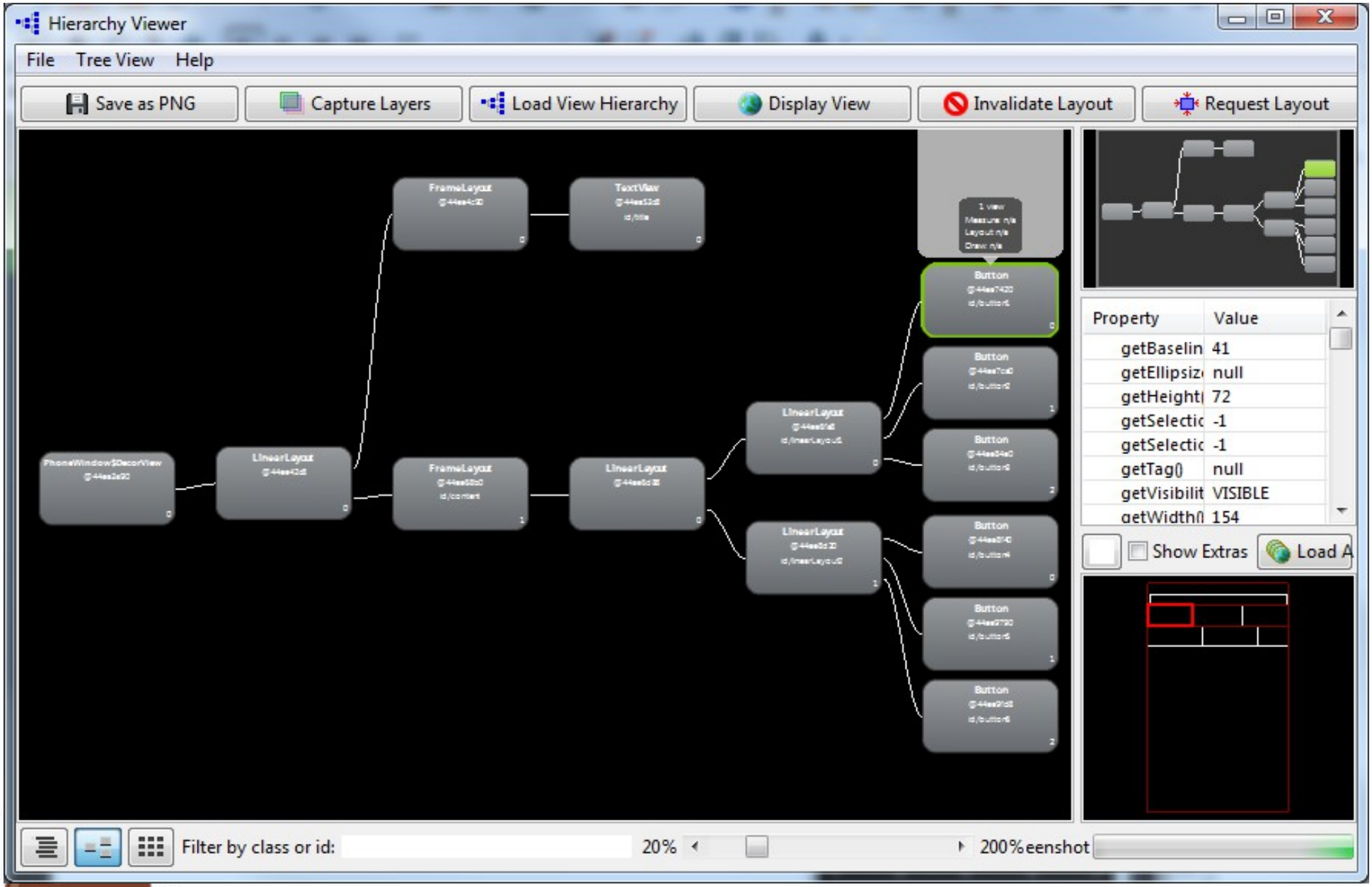
DDMS

- Il Dalvik Debug Monitor Service (o DDMS) permette di gestire i processi in esecuzione sull'emulatore o sul dispositivo reale.
- E' possibile killare i processi, selezionare uno specifico processo da debuggare, generare informazioni di trace, visualizzare le informazioni relative allo heap ed ai thread, catturare screenshot dell'emulatore o del dispositivo, etc.

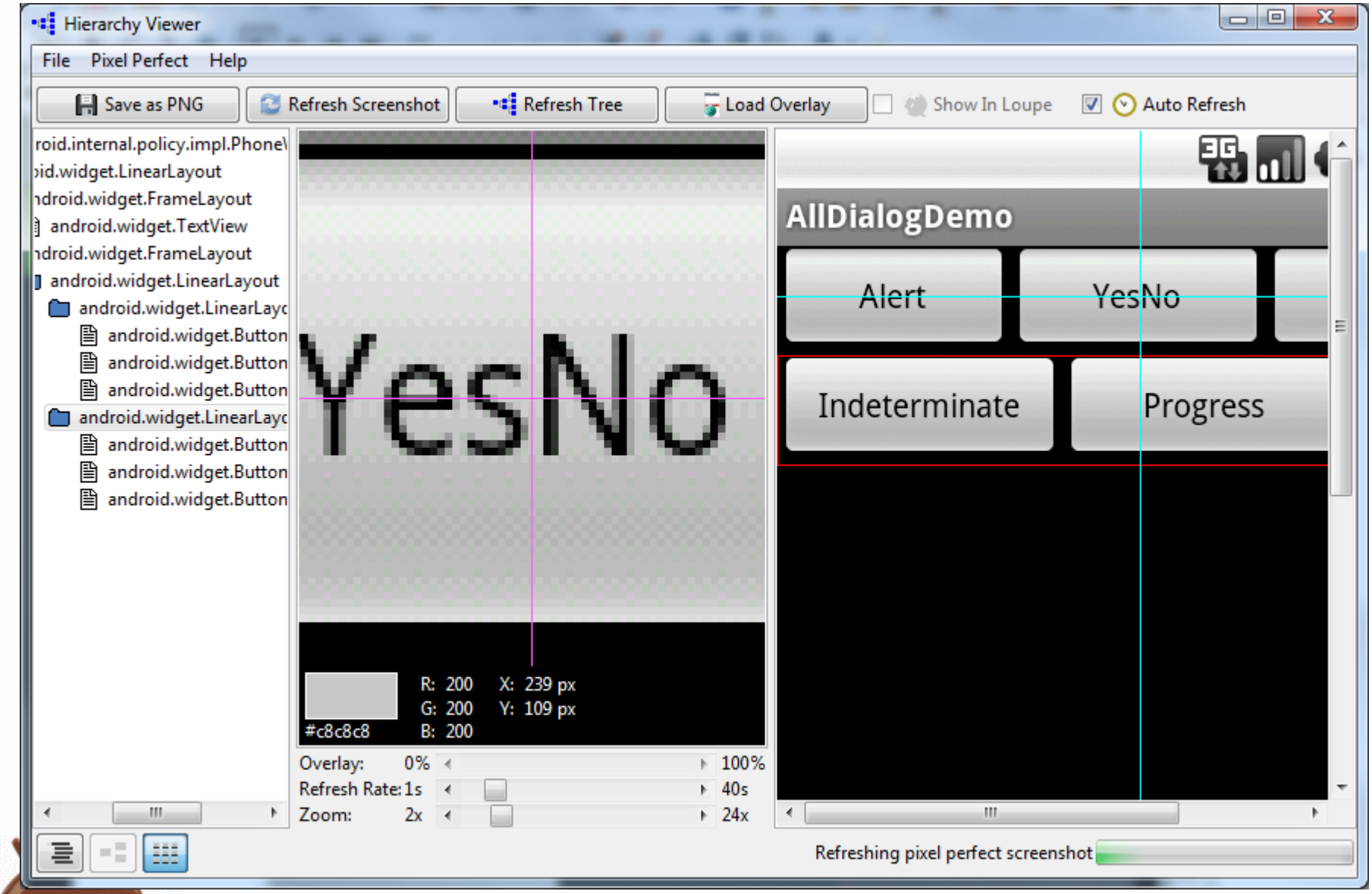
Hierarchy Viewer

- Lo Hierarchy Viewer permette di debuggare e ottimizzare l'interfaccia utente.
- La **Layout View** fornisce una rappresentazione visuale della gerarchia di View dell'applicazione.
- La **Pixel Perfect View** fornisce invece una vista ingrandita del display emulato.

Layout View

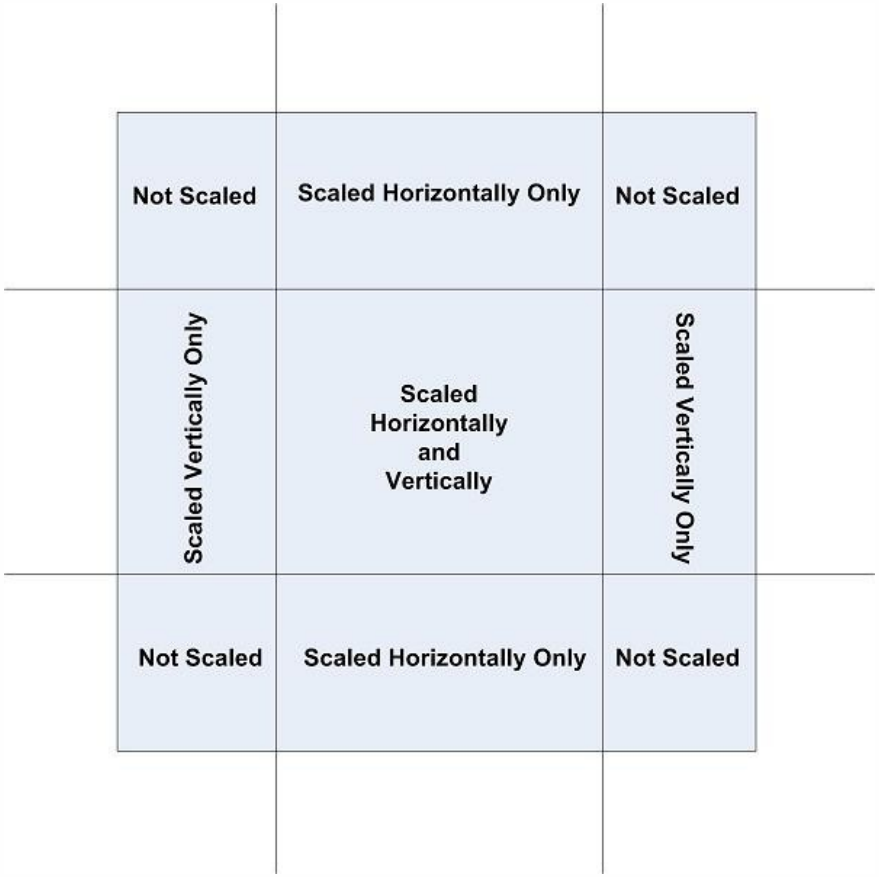

Hierarchy Viewer
File
Tree View
Help
Save as PNG
Capture Layers
Load View Hierarchy
Display View
Invalidate Layout
Request Layout
Property
Value
getBaselin
41
getEllipsiz
null
getHeight
72
getSelectic
-1
getSelectic
-1
getTag()
null
getVisibilit
VISIBLE
getWidth()
154
Show Extras
Load A
Filter by class or id:
20%
200%
eenshot

Pixel Perfect View

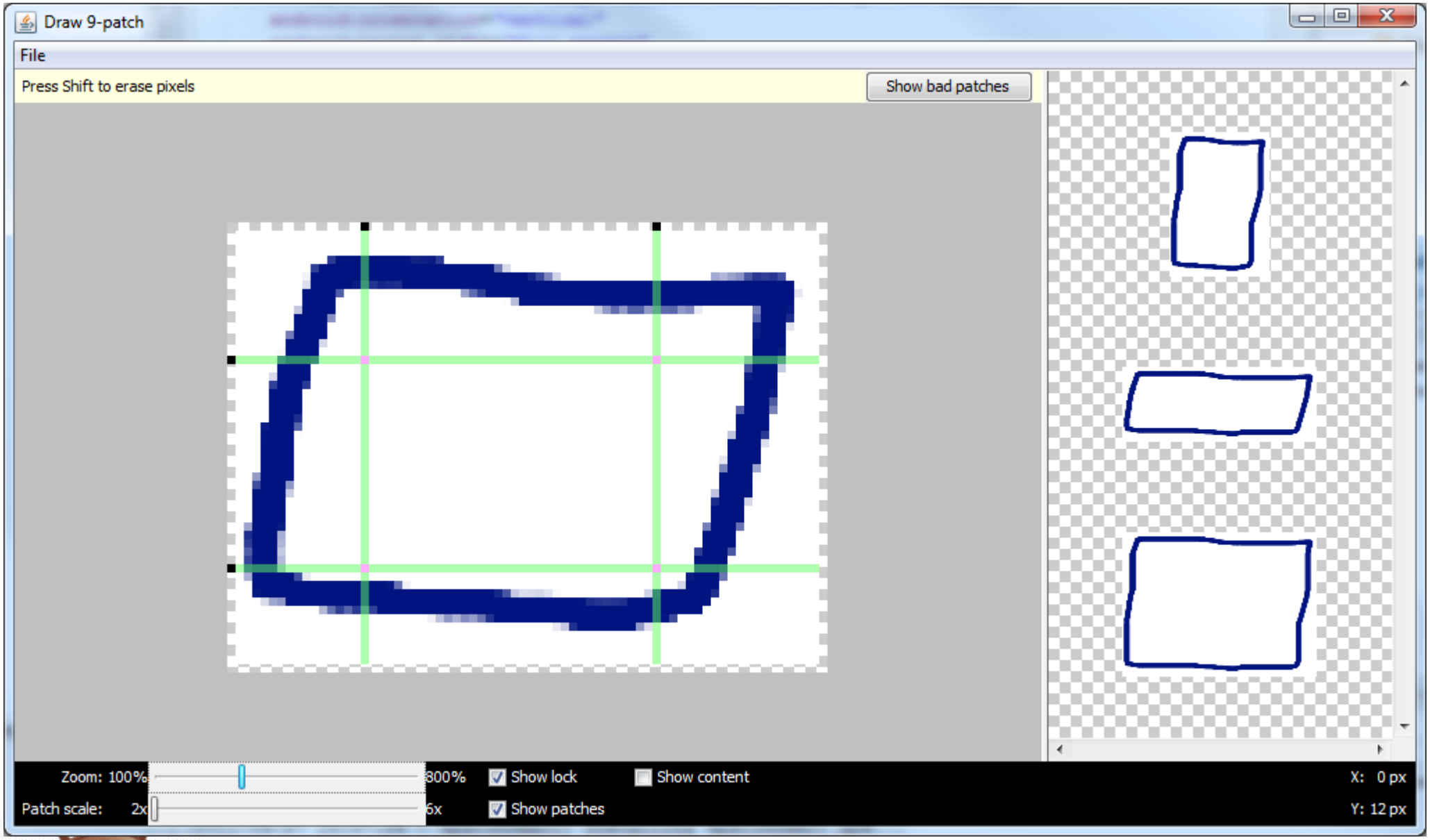


Draw 9-patch 1/3

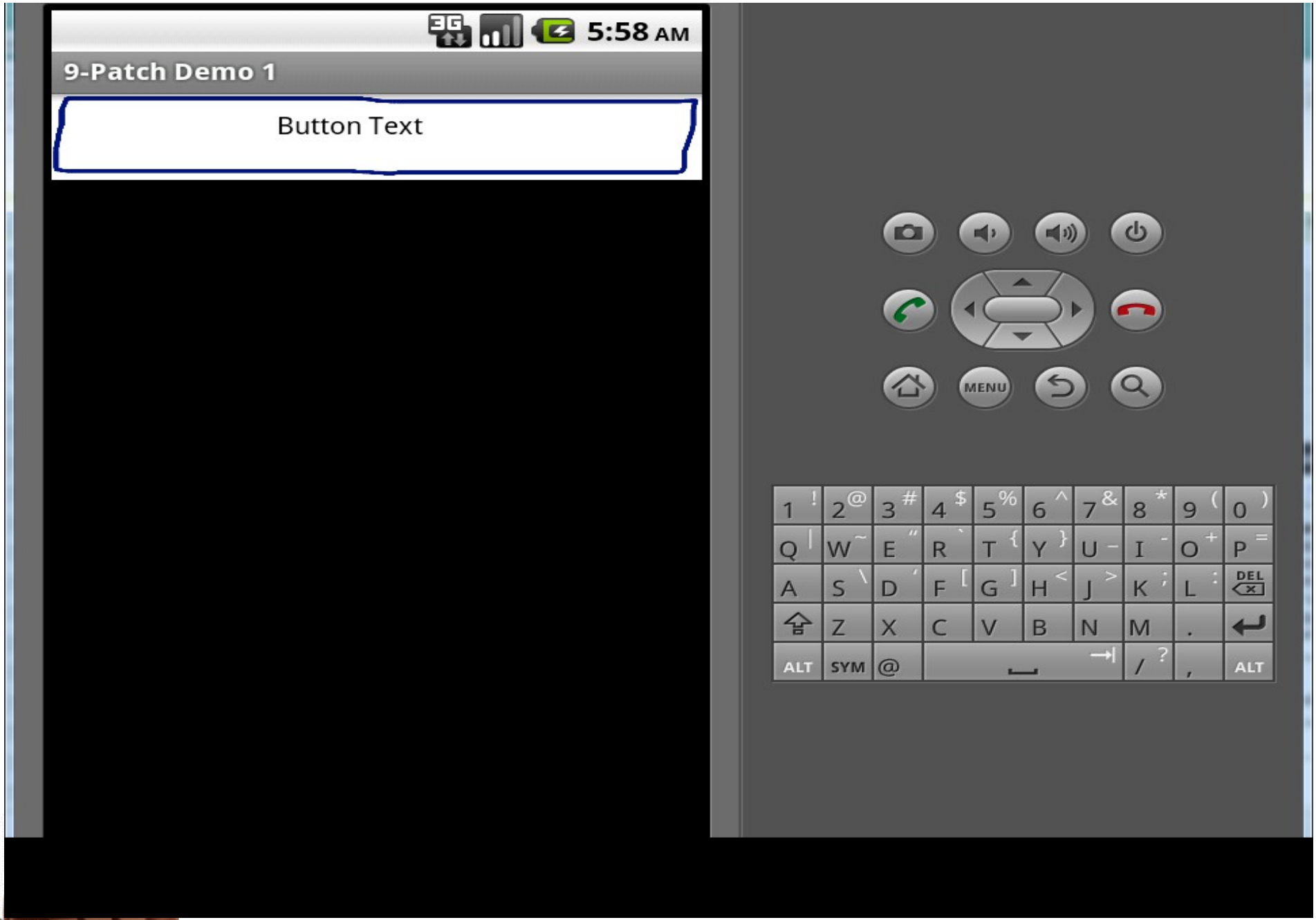
- E' uno strumento WYSIWYG che permette di creare cornici 9-patch.
- Si tratta di bitmap ridimensionabili divise in 9 sezioni: i 4 angoli non vengono mai scalati, i lati della cornice vengono scalati solo in una dimensione e il solo centro dell'immagine viene scalato in entrambe le dimensioni.



Draw 9-patch 2/3



Draw 9-patch 2/3



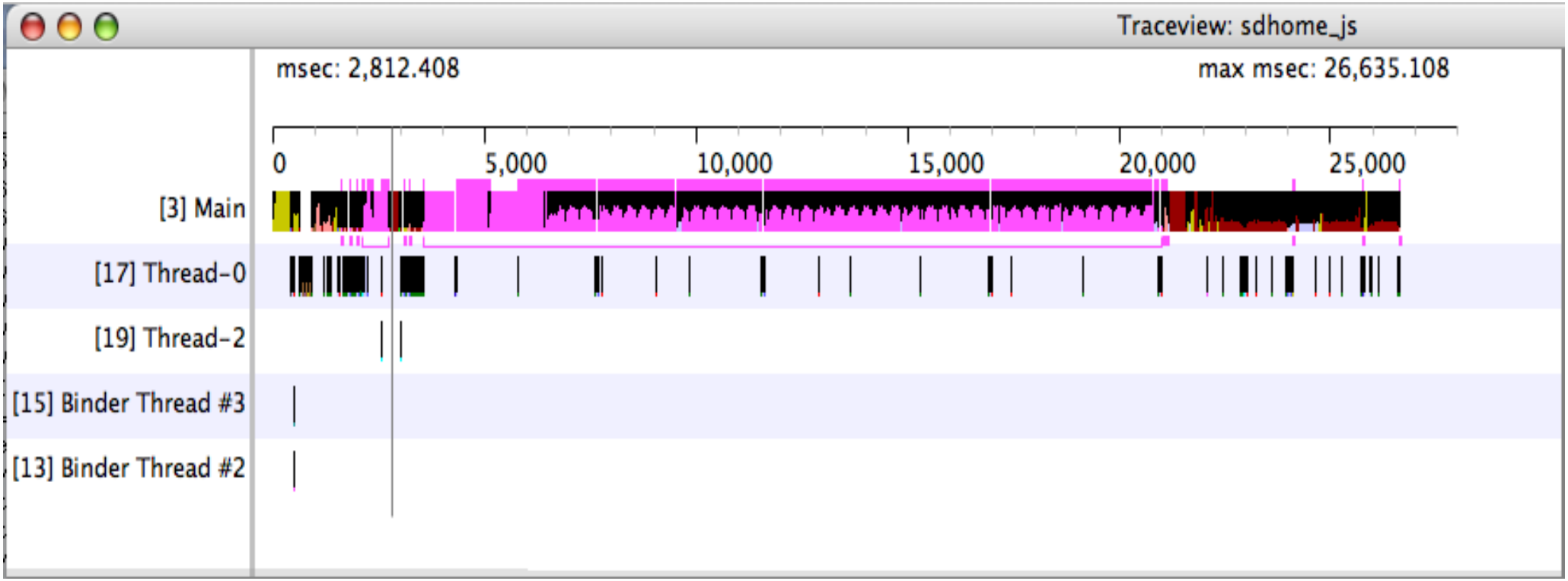
AAPT

- L'Android Asset Packaging Tool (o AAPT) è lo strumento che permette di creare i file .apk contenenti i binari e le risorse delle applicazioni Android.
- E' lo strumento che viene automaticamente chiamato da Eclipse per svolgere questo lavoro, ma può anche essere invocato a riga di comando.

Traceview

- Traceview fornisce una rappresentazione grafica dei log di esecuzione salvati dall'applicazione.
- Il **Timeline Panel** mostra in quale momento ogni thread o metodo è stato avviato e fermato.
- Il **Profile Panel** fornisce un riepilogo di cosa è avvenuto all'interno di un metodo.

Timeline Panel



Profile Panel

Name	Incl %	Inclusive	Excl %	Exclusive	Calls+Rec
▼ 4 android/webkit/LoadListener.nativeFinished ()V	66.6%	17734.382	53.2%	14161.950	14+0
3 android/webkit/LoadListener.tearDown ()V	100.0%	17734.382			14/14
6 android/view/View.invalidate (IIII)V	19.8%	3516.410			2413/2853
57 android/webkit/BrowserFrame.startLoadingResource (ILjava	0.3%	44.636			3/15
53 java/util/HashMap.put (Ljava/lang/Object;Ljava/lang/Objec	0.0%	6.223			6/326
20 android/webkit/JWebCoreJavaBridge.setSharedTimer (J)V	0.0%	2.593			2/730
378 android/view/ViewGroup.requestLayout ()V	0.0%	1.139			2/54
315 java/util/HashMap.<init> (I)V	0.0%	0.879			3/41
629 android/webkit/BrowserFrame.loadCompleted ()V	0.0%	0.285			1/1
598 android/webkit/WebView.didFirstLayout ()V	0.0%	0.231			1/2
703 android/webkit/BrowserFrame.windowObjectCleared (I)V	0.0%	0.036			1/2
▶ 5 android/webkit/JWebCoreJavaBridge$TimerHandler.handleMessa	16.3%	4342.697	0.5%	132.018	730+0
▶ 6 android/view/View.invalidate (IIII)V	15.6%	4161.341	1.2%	319.164	2853+0
▶ 7 android/webkit/JWebCoreJavaBridge.access$300 (Landroid/webk	15.1%	4025.658	0.1%	26.727	729+0
▶ 8 android/webkit/JWebCoreJavaBridge.sharedTimerFired ()V	15.0%	3998.931	8.5%	2256.801	729+0
▶ 9 android/view/View.invalidate (Landroid/graphics/Rect;)V	13.8%	3671.342	0.9%	246.190	2853+0
▶ 10 android/view/ViewGroup.invalidateChild (Landroid/view/View;La	12.4%	3298.987	6.3%	1687.629	876+1148
▶ 11 android/event/EventLoop.processPendingEvents ()V	6.3%	1674.317	0.6%	151.201	12+0
▶ 12 android/view/ViewRoot.handleMessage (Landroid/os/Message;)V	4.6%	1217.210	0.0%	1.992	35+0
▶ 13 android/view/ViewRoot.performTraversals ()V	4.5%	1209.815	0.0%	7.190	34+0
▶ 14 android/view/ViewRoot.draw (Z)V	4.1%	1096.832	0.0%	11.508	34+0
▶ 15 android/policy/PhoneWindow$DecorView.drawTraversal (Landrc	3.9%	1040.408	0.0%	2.218	34+0
▶ 16 android/widget/FrameLayout.drawTraversal (Landroid/graphics,	3.8%	1023.779	0.0%	3.129	34+48
▶ 17 android/view/View.drawTraversal (Landroid/graphics/Canvas;La	3.8%	1022.611	0.1%	19.213	34+154
▶ 18 android/view/ViewGroup.dispatchDrawTraversal (Landroid/grap	3.8%	1000.413	0.2%	42.609	34+130
▶ 19 android/view/ViewGroup.drawChild (Landroid/graphics/Canvas;	3.7%	983.346	0.2%	42.926	34+150
▶ 20 android/webkit/JWebCoreJavaBridge.setSharedTimer (J)V	3.5%	929.506	0.2%	57.241	730+0
▶ 21 android/webkit/WebView.nativeDrawRect (Landroid/graphics/Ca	3.5%	923.805	3.0%	807.952	15+0
▶ 22 android/net/http/QueuedRequest.start (Landroid/net/http/Queu	3.2%	847.172	0.0%	3.556	15+0
▶ 23 android/net/http/QueuedRequest$QREventHandler.endData ()V	3.1%	828.592	0.0%	1.619	15+0
▶ 24 android/net/http/QueuedRequest.setupRequest ()V	3.1%	819.888	0.0%	5.860	15+0
▶ 25 android/net/http/QueuedRequest.requestComplete ()V	3.1%	816.585	0.0%	1.506	15+0
▶ 26 android/webkit/CookieManager.getCookie (Landroid/content/Cc	2.7%	722.837	0.0%	8.081	15+0
▶ 27 android/webkit/LoadListener.commitLoad ()V	2.6%	688.168	0.1%	17.708	58+0
▶ 28 android/webkit/LoadListener.nativeAddData ([BI)V	2.3%	621.864	1.2%	306.817	57+0
▶ 29 android/graphics/Rect.offset (II)V	2.2%	573.985	2.2%	573.985	17210+0

Find:

mksdcard

- mksdcard è uno script che facilita il compito di creare una immagine di disco che possa essere usata nell'emulatore.
- Simula la presenza di una scheda di memoria esterna (ad esempio SD-Card).
- L'immagine di disco viene creata in formato FAT32.

The Monkey

- Il cosiddetto ”UI/Application Exerciser Monkey” (letteralmente ”la scimmia che esercita l'interfaccia grafica e l'applicazione”) è un programma che gira sull'emulatore o sul dispositivo reale sequenze pseudo-casuali di eventi (come i click e vari tocchi dello schermo) come pure vari tipi di eventi di sistema.
- The Monkey è usato per stressare l'applicazione che si sta sviluppando, in maniera casuale ma ripetibile.

Conclusioni

In questa lezione abbiamo descritto gli strumenti messi a disposizione dall'SDK che possono essere particolamente utili per risolvere specifici problemi.

Sviluppo di una applicazione

Introduzione

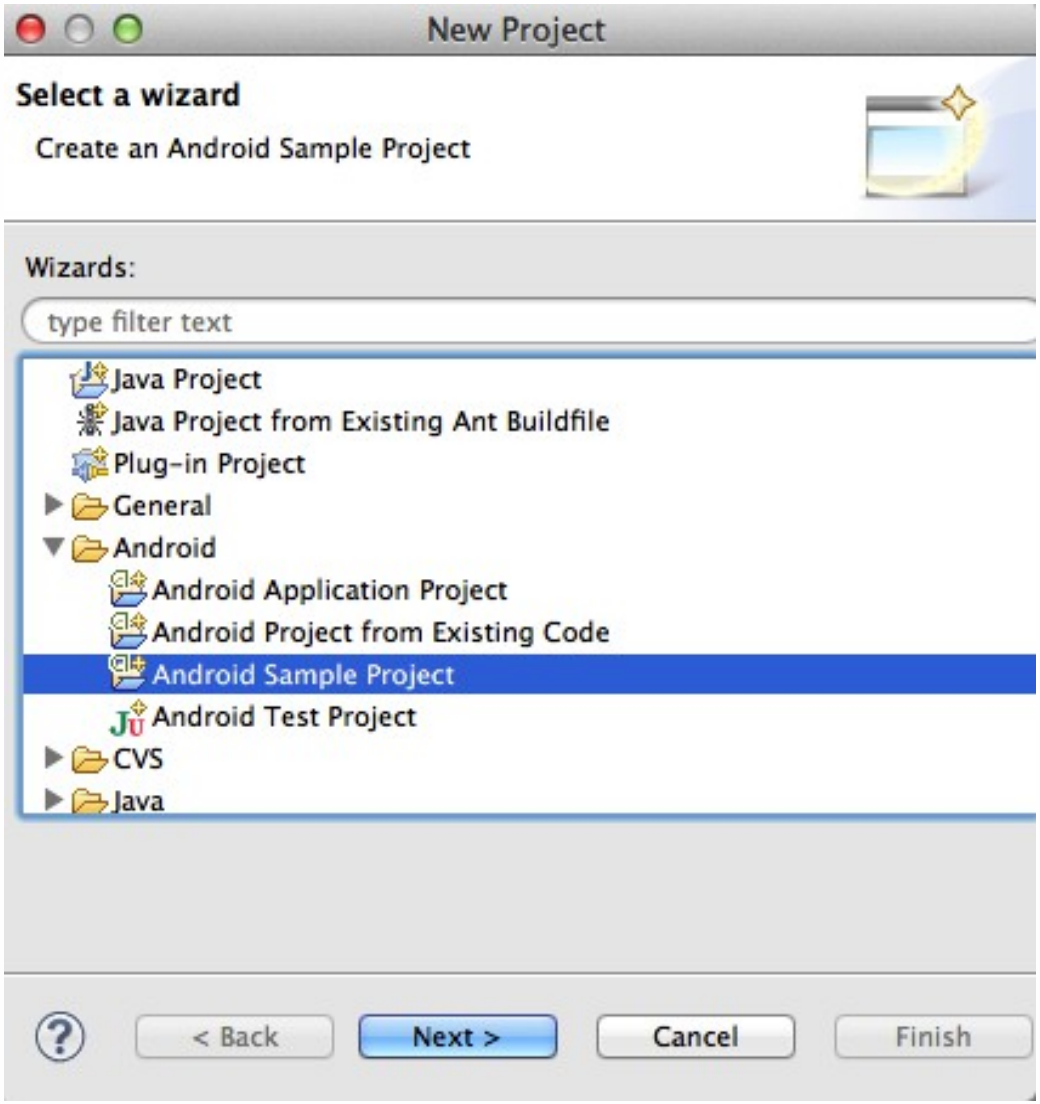
- Questa lezione mostra il ciclo di sviluppo tipico di una applicazione per Android.
- Verrà mostrato come creare e compilare un nuovo progetto, e come eseguirlo nell'emulatore.
- Verrà mostrato il procedimento da usare con Eclipse o su riga di comando.

Creare un progetto Eclipse passo 1

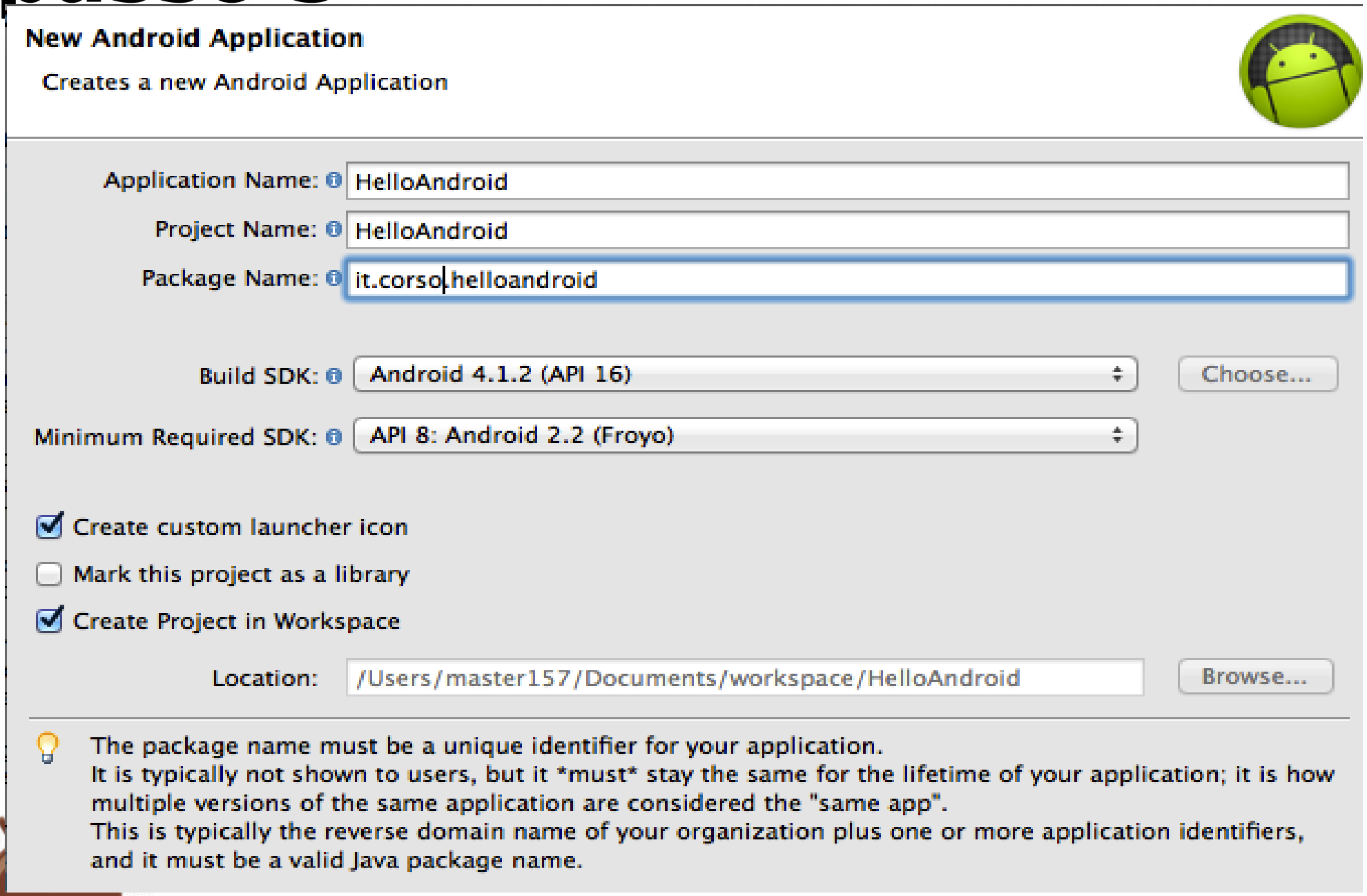
- Per iniziare lo sviluppo di una nuova applicazione per Android bisogna creare un progetto e una configurazione di lancio.
- Per creare il progetto bisogna ovviamente disporre di Eclipse e del plugin ADT installato correttamente.

Creare un progetto Eclipse passo 2

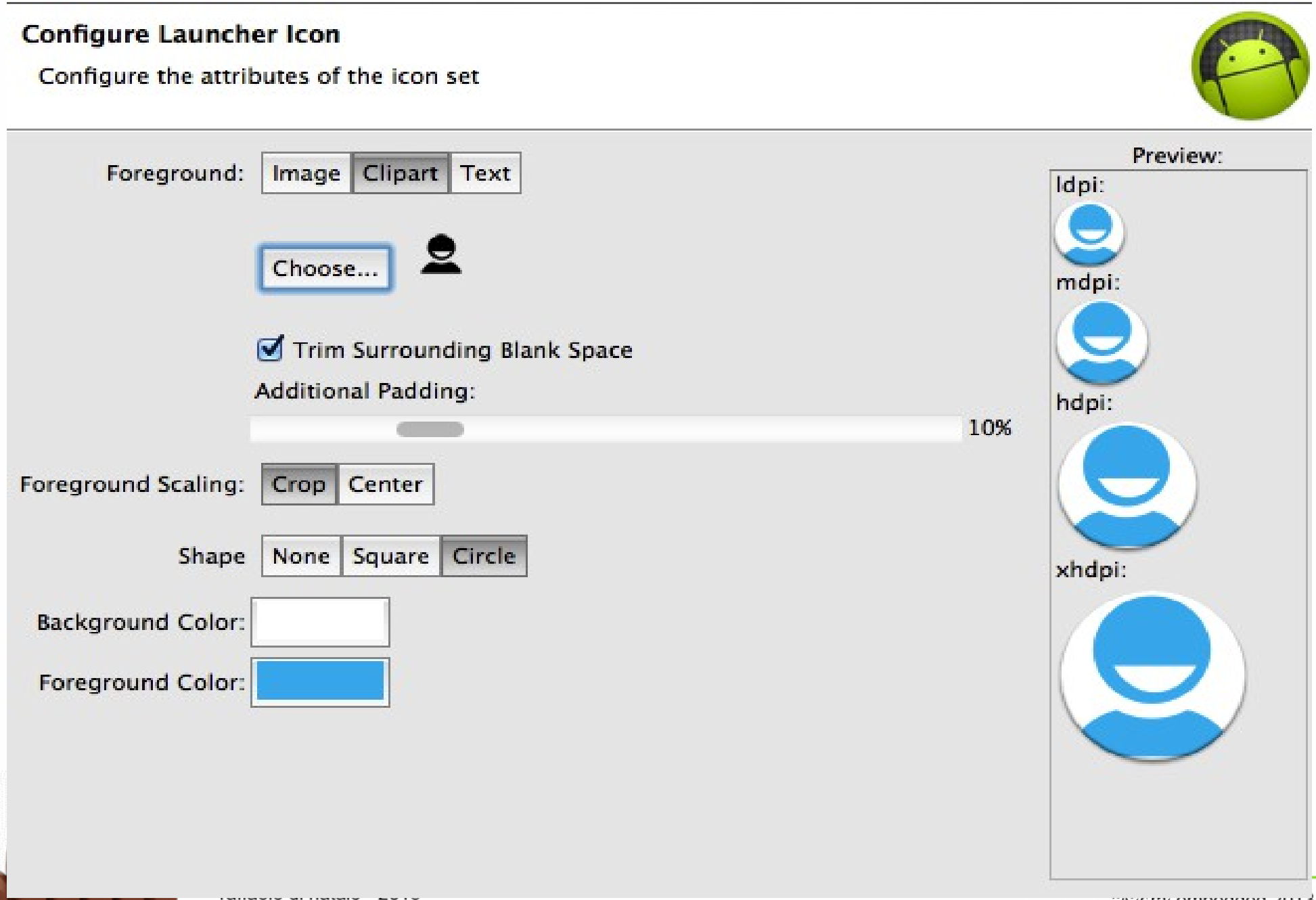
- Il progetto si crea selezionando nell'ordine ”File | New | Project | Android | Android Project”.
- Successivamente dovranno essere definiti il project name, il target, l'activity name, application name ed il package name



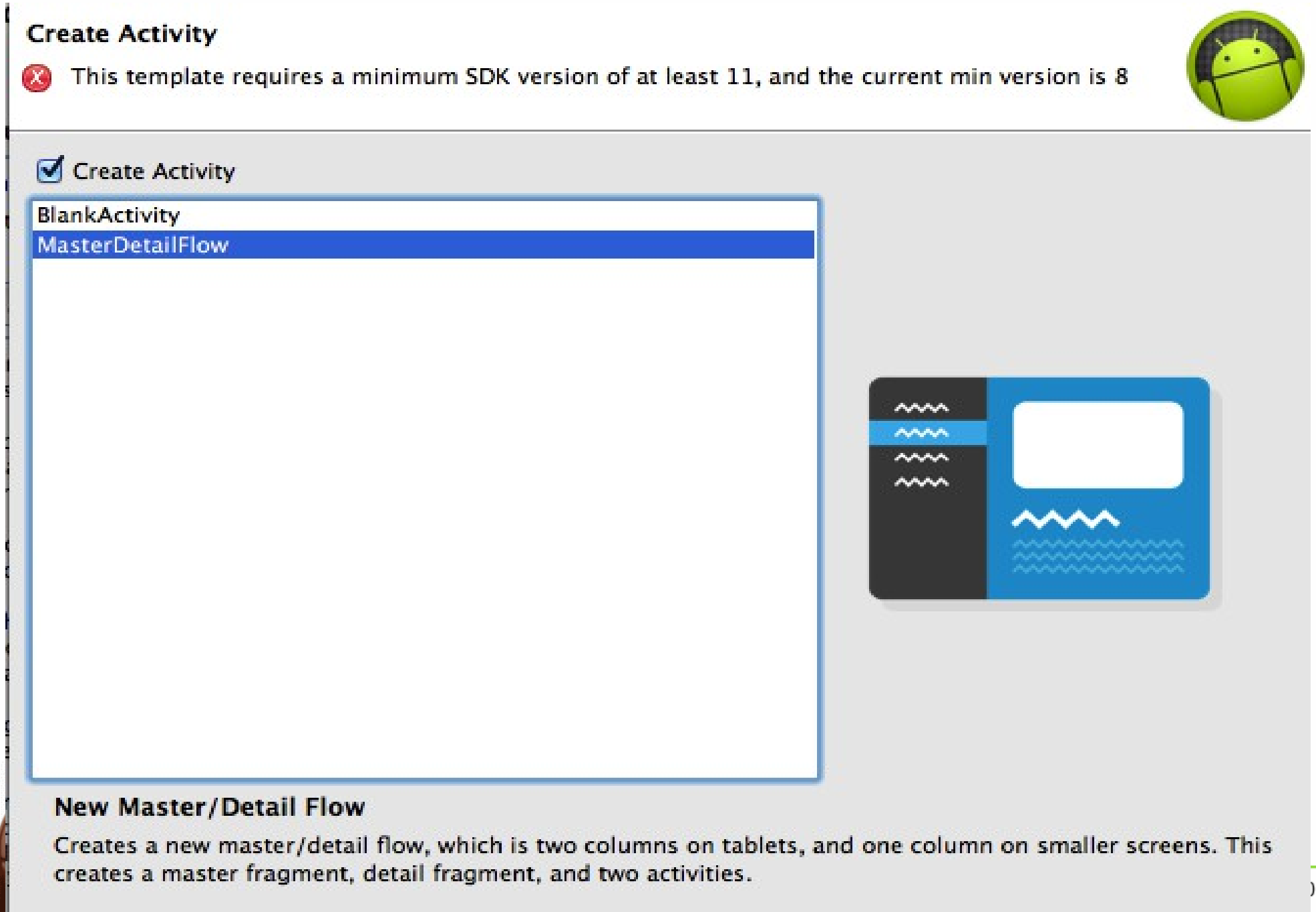
Creare un progetto Eclipse passo 3



Creare un progetto Eclipse passo 4

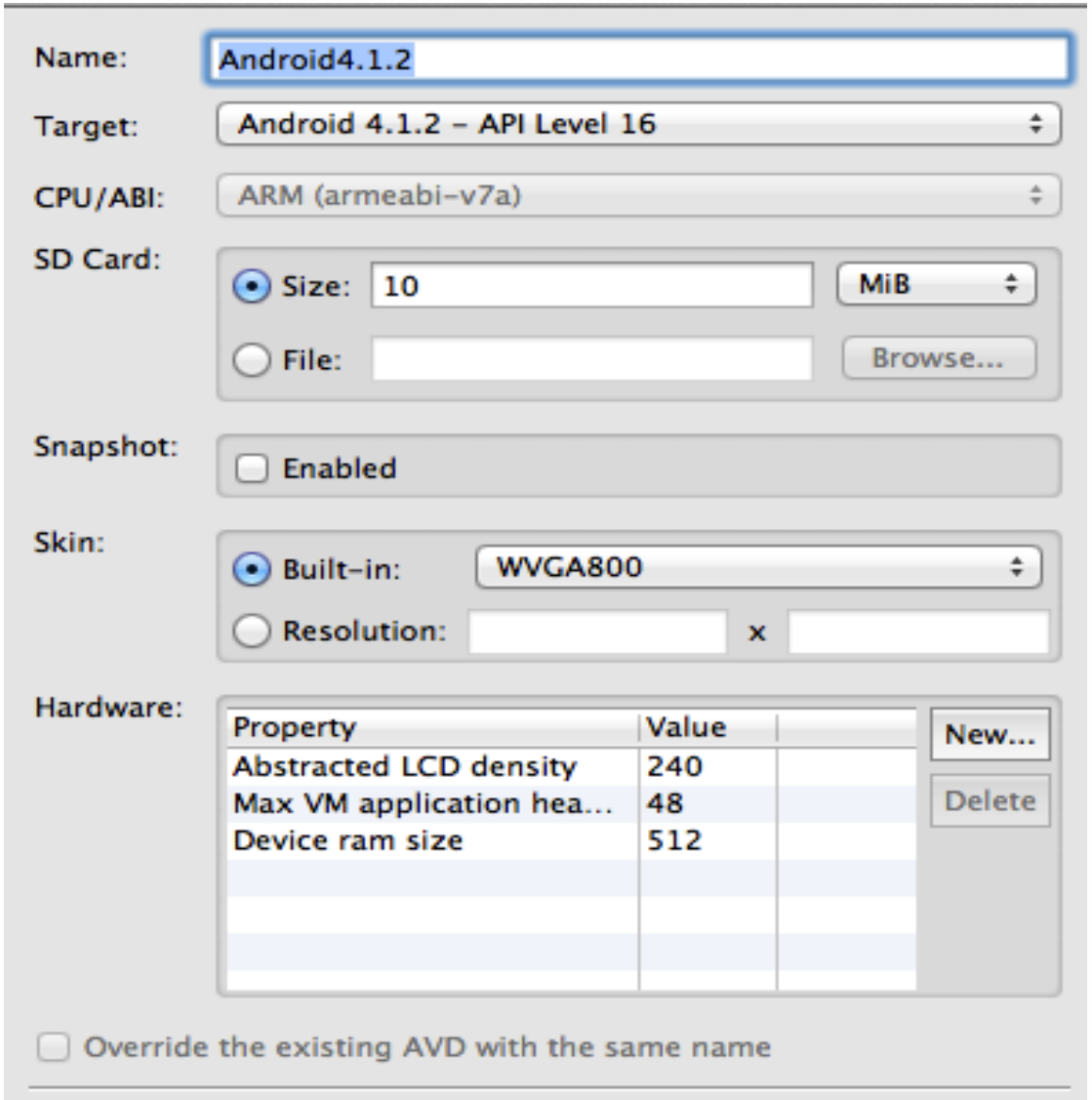


Creare un progetto Eclipse passo 5



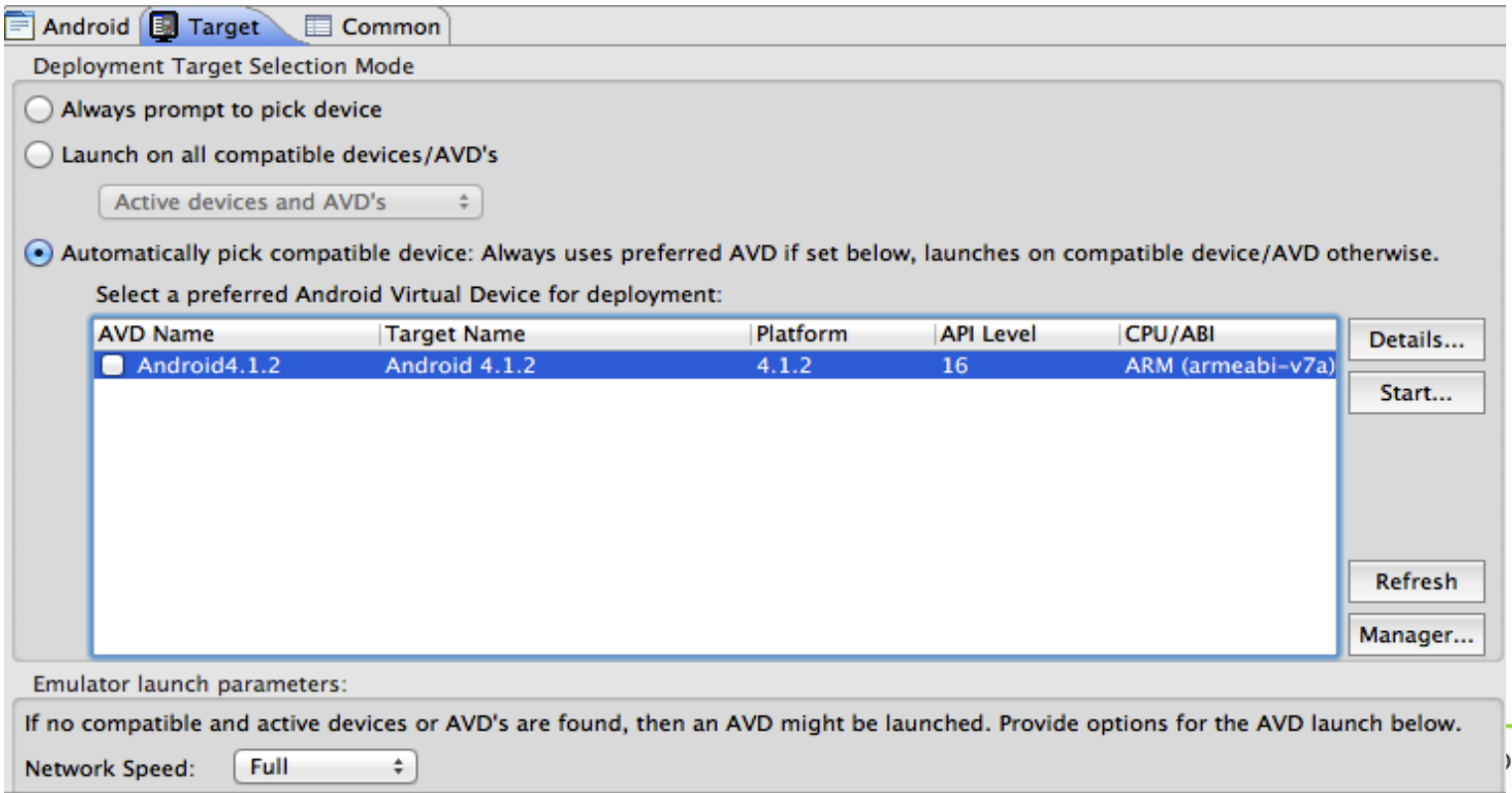
Creare un Android Virtual Device

- Per poter eseguire la nostra applicazione è necessario disporre di un Android Virtual Device (AVD)
- Per crearne uno tramite Eclipse andare su Window | Android Sdk and AVD Manager, oppure richiamando il tool Emulator senza argomenti da prompt dei comandi



Creare la configurazione di lancio

- Bisogna creare una configurazione di lancio in Eclipse selezionando il menu ”Run” e poi uno dei quattro tra ”Open Run Dialog”, ”Open Debug Dialog”, ”Run Configurations” oppure ”Debug Configurations”.
- Dopo avere assegnato un nome alla configurazione specificare il nome del progetto, l'Activity da lanciare ed eventuali parametri da passare all'emulatore.



Eseguire e debuggare l'applicazione

- Adesso è possibile avviare l'applicazione selezionando ”Run | Run” oppure ”Run | Debug” a seconda di quello che si intende fare.
- Questa azione farà riferimento alla configurazione di lancio usata più di recente.
- I passi che adesso verranno eseguiti saranno nell'ordine l'avvio dell'emulatore, la compilazione del progetto (solo in caso di modifiche) e l'installazione dell'applicazione nell'emulatore.

Sviluppo con altri IDE

- L'ambiente di sviluppo raccomandato per sviluppare una applicazione Android è Eclipse con il plugin ADT installa,oppure l'ADT Bundle ma è possibile utilizzare altri metodi.
- A partire dalla versione 9 l'IDE IntelliJ IDEA di JetBrains integra un supporto all'Android SDK.
- E' possibile utilizzare l'IDE Eclipse senza avere installato il plugin ADT.
- In tutti i casi i tool contenuti nell'SDK forniscono gli strumenti per configurare, compilare e debuggare una applicazione.

Creare un progetto da riga di comando

- Per creare un nuovo progetto da riga di comando si utilizza sempre il tool android:

```
android create project
-t <target_ID>
-n <your_project_name>
-p  path/to/your/project
-a <your_activity_name>
-k <your_package_namespace>
```

```
android create project
-t 8
-n prova
-p c:\prova
-a ActivityDemo
-k it.corso.activitydemo
```

- Assicurarsi di avere inserito le directory platform-tools/ e tools/ tra le directory della variabile d'ambiente PATH

File ottenuti 1/3

L'esecuzione del precedente comando produce in output i seguenti file:

- **AndroidManifest.xml** è il file di manifesto contenente le informazioni relative al progetto
- **build.xml** è un file Ant che può essere usato per compilare l'applicazione a riga di comando
- **src/it/corso/ActivityDemo.java** è il file sorgente che conterrà l'activity specificata.
- **proguard.cfg** è utilizzato dal tool proguard che offusca ed ottimizza il codice producendo un .apk di dimensioni ridotte.

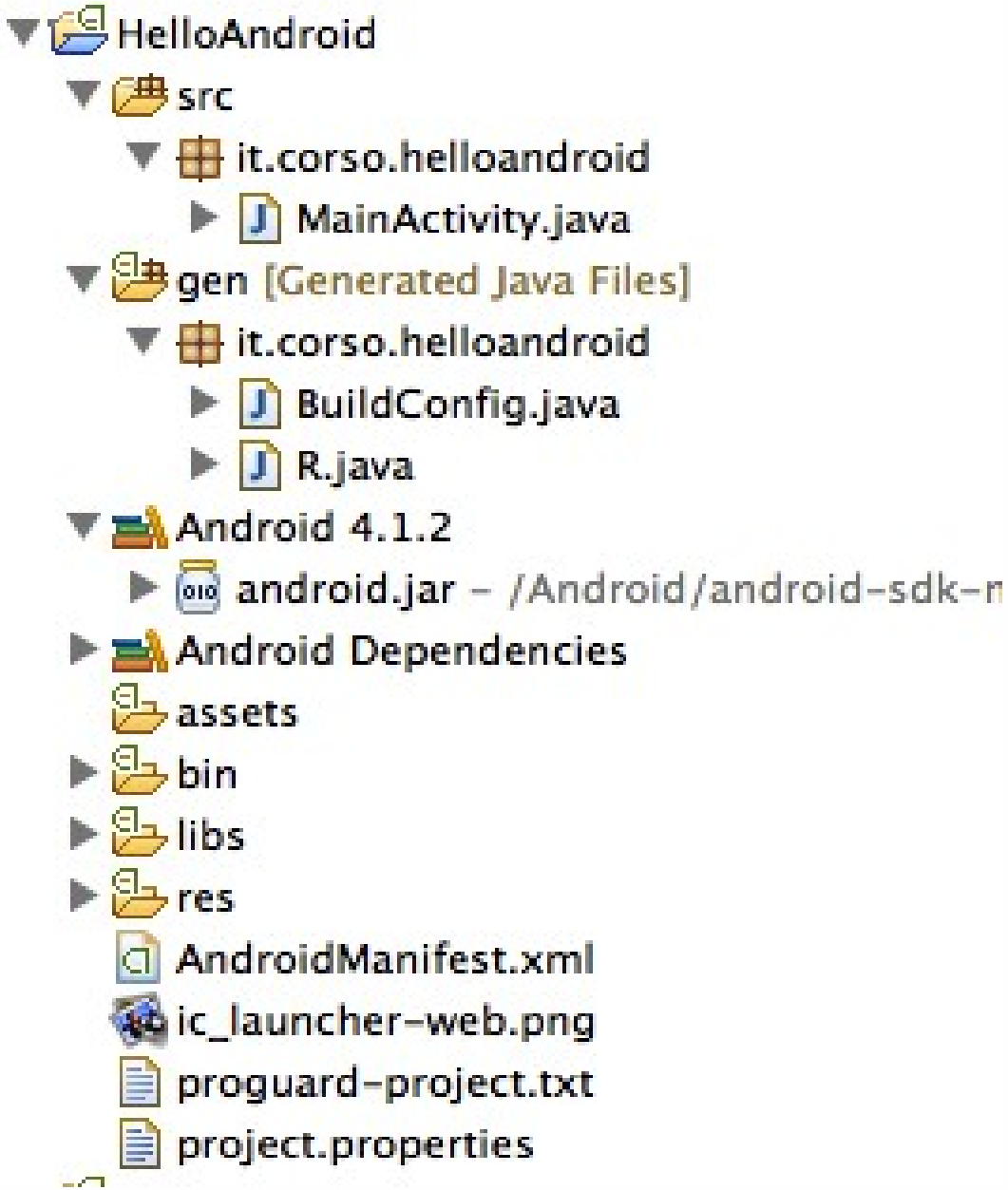
File ottenuti 2/3

- **Proguard-project.txt**

  Si riferisce al file di progetto del tool Proguard (ottimizzazione e offuscamento del codice)

- **project.properties**

  Viene generato automaticamente dagli Android Tool. Contiene sdk target.



File ottenuti 3/3

- **gen/** contiene i file .Java generati dall'ADT come per es. R.java.
- **res/** contiene le risorse dell'applicazione (immagini, stringhe, layout)
- **assets/** inizialmente vuota contiene altri file di risorse come texture o dati cui e non sono associati alle costanti della classe R.
- **libs/** contiene le library utilizzate dall'applicazione

Directory ottenute

- **src/** conterrà i sorgenti
- **res/** per le risorse esterne
- **bin/** contiene il file .apk e le altre risorse compilate
- **jni/** contiene il codice nativo sviluppato mediante l'Android NDK

Settare l'ambiente per la compilazione

- Per compilare il progetto a riga di comando è necessario avere installato Apache Ant 1.8 o successivo.
- Bisogna inoltre settare la variabile d'ambiente JAVA_HOME per puntare correttamente al percorso in cui è installato il JDK; sotto Windows tale percorso NON può contenere spazi, quindi ad esempio non è possibile installare il JDK nella directory predefinita ”Program Files”:

```
set JAVA_HOME=c:\Programmi\Java\
```

Compilare a riga di comando

- Dopo avere settato la variabile d'ambiente JAVA_HOME è sufficiente spostarsi nella directory che contiene il file build.xml ed eseguire semplicemente ”ant”.
- Ovviamente ogni volta che si cambia un file sorgente o una risorsa bisogna ricompilare tutto rieseguendo il comando ”ant”.

Avviare l'emulatore a riga di comando

- Per eseguire la propria applicazione bisogna innanzitutto caricarla all'interno del file-system del sistema emulato.
- Per prima cosa bisogna avviare l'emulatore con il comando:

```
emulator @avd_name
```

- Per conoscere gli AVD disponibili (prec. creati):

```
android  list
```

- Nell'interfaccia grafica dell'emulatore la vecchia versione della propria applicazione non deve essere in primo piano, quindi conviene spostarsi alla pagina principale premendo il pulsante ”Home”.

Attività 1/2

- Per installare la propria applicazione da riga di comando si può utilizzare il comando ADB (con un AVD avviato).
- La sintassi da usare è **`adb install bin/<nome_applicazione>.apk`**
- Questo comando ricopia il file APK all'interno della directory /data/app dell'emulatore.
- Accedendo all'emulatore la nostra applicazione apparirà tra quelle installate.

Attività 2/2

Rimuovere l'applicazione installata

- Tramite l'emulatore
- Utilizzando l'ADB

  adb uninstall it.dominio.nomepackage

  Es:

  adb uninstall it.corso.helloandroid
- Verificare l'avvenuta rimozione dell'app

Conclusioni

In questa lezione abbiamo visto come creare un nuovo progetto Eclipse e come eseguire la nostra applicazione all'interno dell'emulatore.

Abbiamo inoltre visto come sia possibile effettuare le stesse operazioni anche senza Eclipse, utilizzando solo gli strumenti a riga di comando forniti con l'SDK di Android.

Hello, Android!

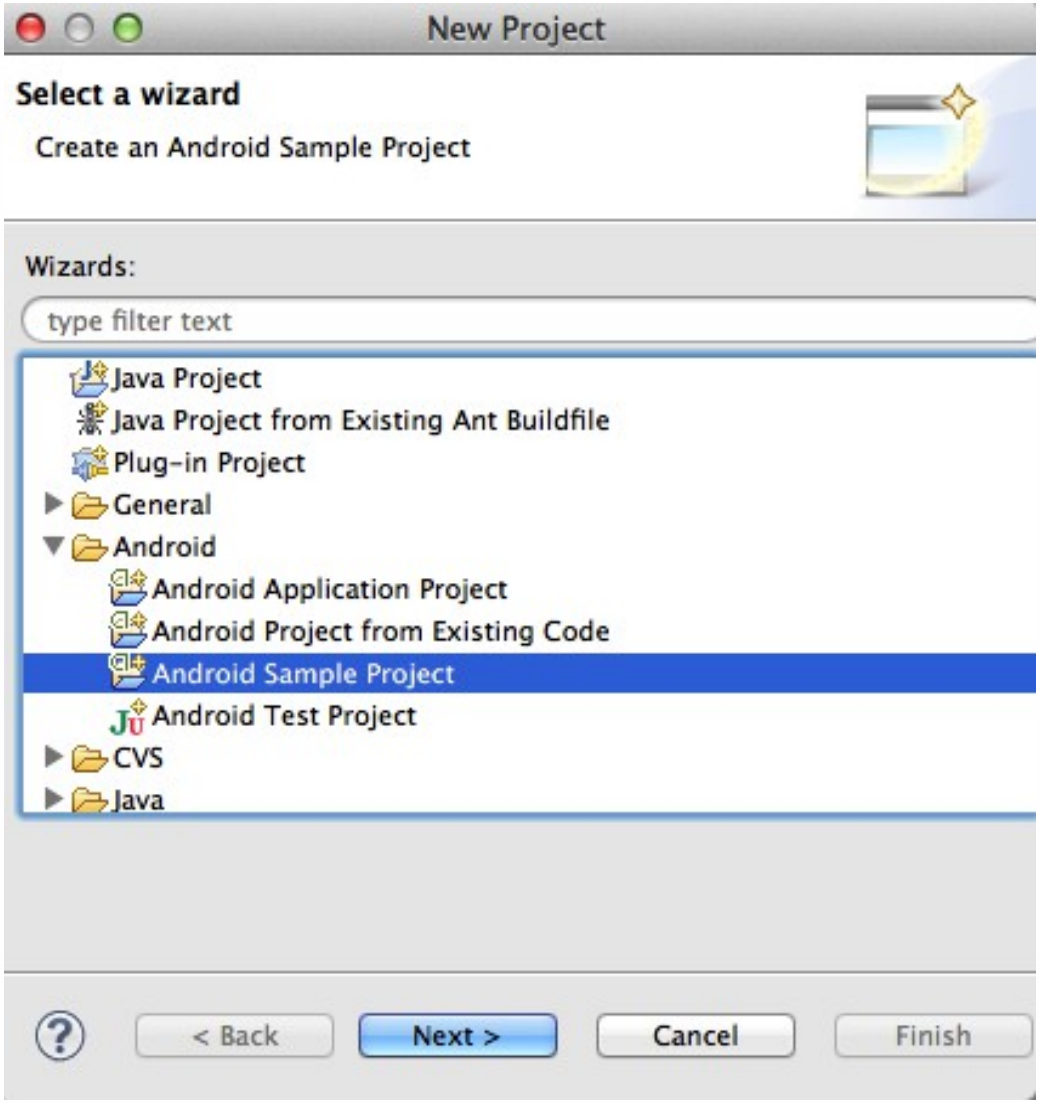
Introduzione

In questa lezione vedremo come applicare i concetti acquisiti nella lezione precedente e creeremo la nostra prima applicazione per Android.

Come tradizione la prima applicazione sviluppata in un nuovo linguaggio si chiama ”Hello World” (nel nostro caso ”Hello Android”) e si limita a visualizzare sullo schermo un messaggio di saluto.

Creare il progetto

- Per creare il progetto Eclipse bisogna selezionare ”File | New | Project | Android Project”:



Dettagli del progetto

New Android Application

Creates a new Android Application

Application Name: HelloAndroid

Project Name: HelloAndroid

Package Name: it.corso.helloandroid

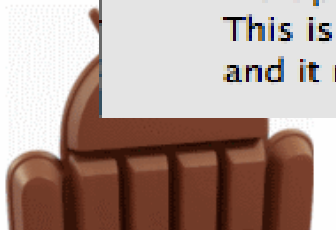
Build SDK: Android 4.1.2 (API 16) — Choose...

Minimum Required SDK: API 8: Android 2.2 (Froyo)

- [x] Create custom launcher icon
- [ ] Mark this project as a library
- [x] Create Project in Workspace

Location: /Users/master157/Documents/workspace/HelloAndroid — Browse...

The package name must be a unique identifier for your application.
It is typically not shown to users, but it *must* stay the same for the lifetime of your application; it is how multiple versions of the same application are considered the "same app".
This is typically the reverse domain name of your organization plus one or more application identifiers, and it must be a valid Java package name.



Nome progetto e Activity

- Il nome del progetto e il nome dell'Activity possono coincidere o meno: nell'esempio il progetto è ”HelloAndroid” e l'activity è ”HelloAndroidActivity”.
- La differenza è che il progetto dà il nome alla directory che lo conterrà, mentre il nome dell'Activity è il nome della classe che deriveremo dalla classe Activity di Android e che sarà la classe principale dell'applicazione.

Nome del package

Bisogna specificare il namespace sotto il quale si vuole che risieda il nostro codice sorgente, seguendo le stesse regole del linguaggio Java.

Poiché è importante che nel sistema il nome del package sia unico rispetto a tutti quelli installati nel sistema, si usa un nome di package in stile ”nome di dominio”. Nell'esempio è stato utilizzato ”it.corso”.

Nome dell'applicazione

•L'ultimo dettaglio da specificare al momento della creazione del progetto è l'Application Name: questo è il nome dell'applicazione che apparirà sia come descrizione nell'elenco di applicazioni installate che sulla barra del titolo quando la nostra applicazione sarà in esecuzione, quindi è importante che si tratti di un nome facilmente comprensibile: nel nostro caso ”HelloAndroid”.

Può contenere spazi e punteggiatura.

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="it.corso.helloandroid"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".HelloAndroidActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>
</manifest>
```

Codice autogenerato

Dopo avere cliccato su ”Finish” il file HelloAndroidActivity.Java conterrà il seguente codice:

```
package it.corso.helloandroid;

import android.app.Activity;
import android.os.Bundle;

public class HelloAndroidActivity extends Activity
{
 /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle
savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Codice modificato

```
package it.corso.helloandroid;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class HelloAndroidActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Hello, Android!");
        setContentView(tv);
    }
}
```

View e TextView

- In Android le interfacce utente sono composte da gerarchie di classi dette View.
- Una View è semplicemente un oggetto che può essere disegnato, come un radio button, una animazione o (come nel nostro caso) una etichetta di testo.
- Il nome della sottoclasse della classe View che permette di gestire il testo è semplicemente TextView.

Creazione della TextView

Per creare una TextView si procede come si fa di solito in Java invocando il costruttore:

```
TextView tv = new TextView(this);
```

L'argomento passato al costruttore deve essere una istanza del Context di Android, dove il Context è semplicemente un handle che ci permette di accedere al sistema (ad esempio per risolvere risorse, ottenere l'accesso a database e preferenze e così via).

Metodi di TextView

Poiché la classe HelloAndroid è derivata da Activity che a sua volta è derivata da Context, allora al costruttore di TextView possiamo passare ”this” ovvero la nostra istanza di HelloAndroid.

Possiamo poi invocare un metodo di TextView per dirgli quale deve essere il testo da visualizzare:

```
tv.setText("Hello, Android");
```

Collegamento al display

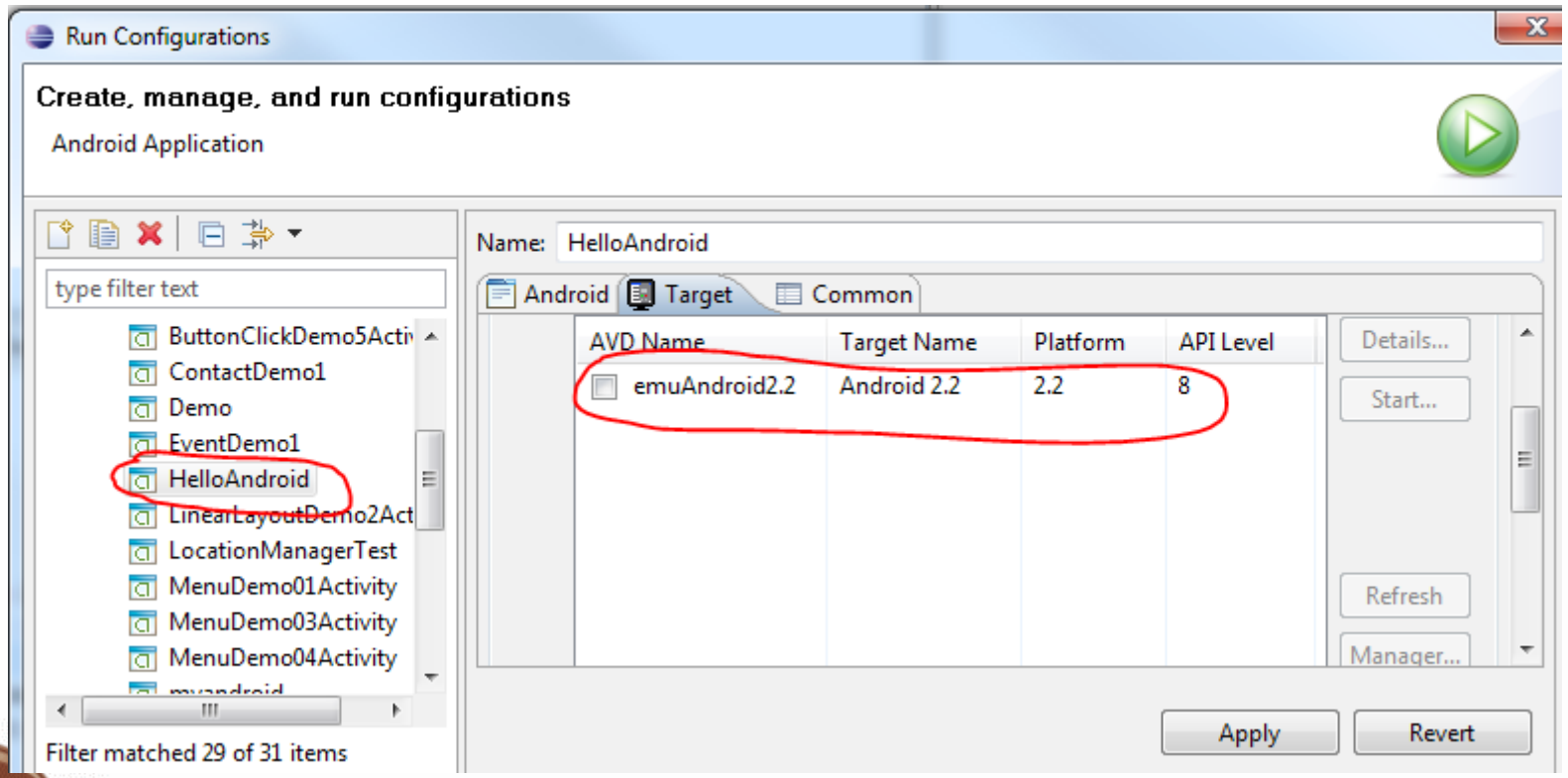
Dopo avere creato la TextView e averle detto quale testo mostrare, l'ultimo passo consiste nel collegare questa TextView con lo schermo:

```
setContentView(tv);
```

Questo metodo di Activity indica al sistema quale View deve essere associata all'interfaccia grafica dell'Activity stessa (in questo caso una TextView). Se non viene mai invocato l'Activity mostrerà solo uno schermo vuoto.

Eseguire Hello Android

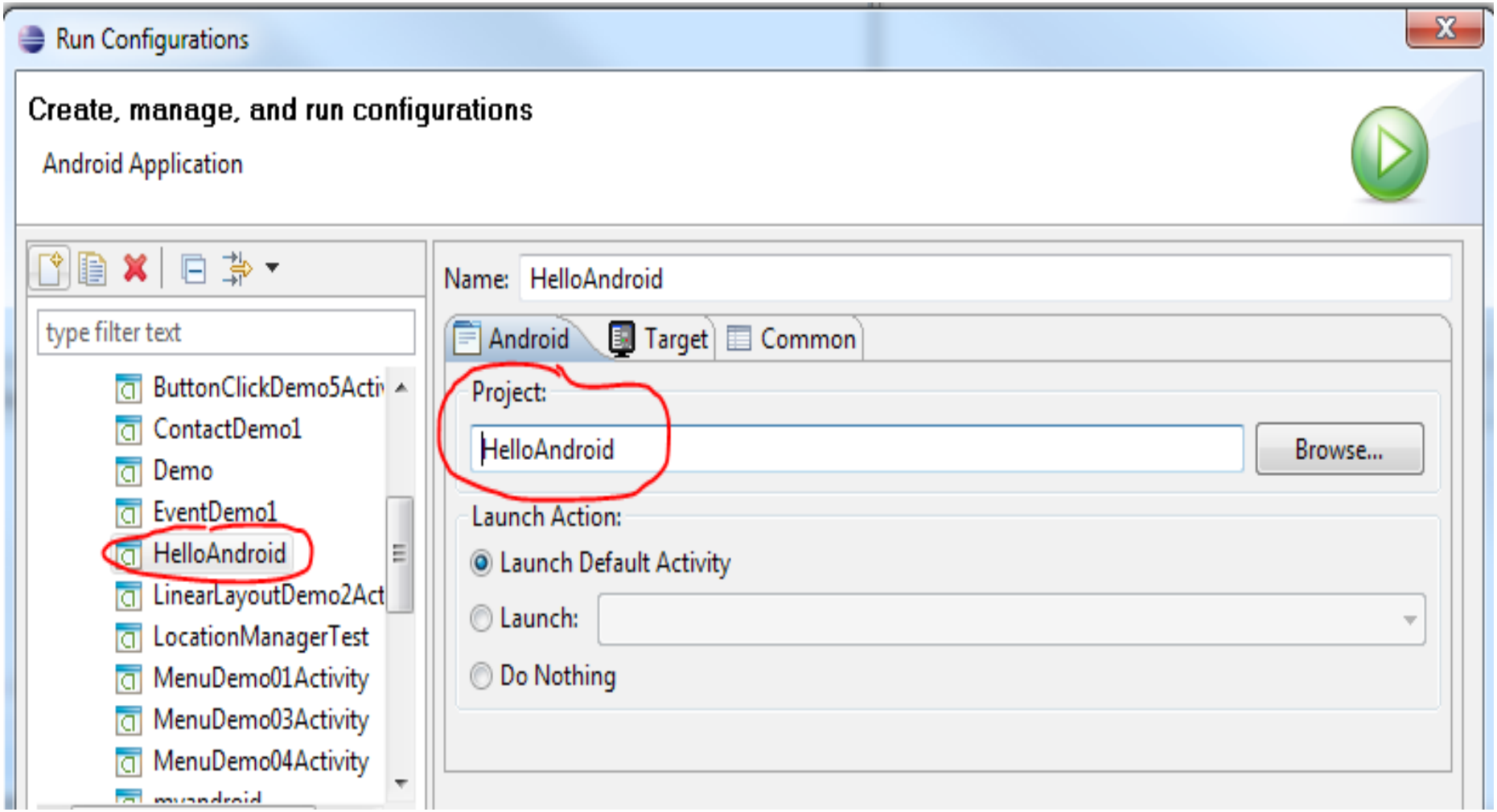
- Per eseguire l'applicazione a seconda della versione di Eclipse bisogna selezionare ”Run | Open Run Dialog” oppure ”Run | Run Configurations”:



Personalizzare la configurazione

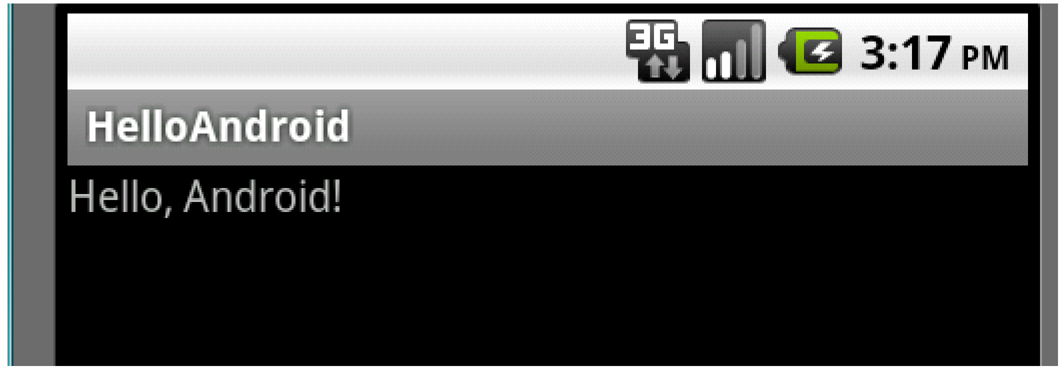
- Selezionare dal menu ”Android Application | New_configuration”.
- Assegnare un nome significativo alla nuova configurazione, per esempio ”Hello, Android”.
- Selezionare il progetto ”HelloAndroid”.
- Selezionare l'Activity principale del progetto, ovvero ”com.android.hello.HelloAndroid”.
- Il modulo compilato sarà:

Configurazione di lancio



Esecuzione

- Infine premendo il pulsante ”Run” si otterrà:



- Durante l'avvio, la finestra console dell'IDE ci comunica passo passo le operazioni in corso:

Console

Android

```
[2011-04-13 17:16:51 - HelloAndroid] ------------------------------
[2011-04-13 17:16:51 - HelloAndroid] Android Launch!
[2011-04-13 17:16:51 - HelloAndroid] adb is running normally.
[2011-04-13 17:16:51 - HelloAndroid] Performing it.corso.helloandroid.HelloAndroidActivity activity launch
[2011-04-13 17:16:51 - HelloAndroid] Automatic Target Mode: Preferred AVD 'emuAndroid2.2' is available on emulator 'emulator-5554'
[2011-04-13 17:16:51 - HelloAndroid] Uploading HelloAndroid.apk onto device 'emulator-5554'
[2011-04-13 17:16:51 - HelloAndroid] Installing HelloAndroid.apk...
[2011-04-13 17:17:08 - HelloAndroid] Success!
[2011-04-13 17:17:09 - HelloAndroid] Starting activity it.corso.helloandroid.HelloAndroidActivity on device emulator-5554
```

Commenti

- Come si vede nell'immagine precedente, la scritta ”Hello, Android” viene mostrata due volte.
- La scritta più in alto è il nome dell'applicazione mostrato sulla barra del titolo; è la stringa che abbiamo scelto come Application Name quando abbiamo creato il progetto.
- Invece la seconda scritta è il testo che abbiamo associato alla TextView.

Conclusioni

In questa lezione abbiamo visto come creare la nostra prima applicazione.

Abbiamo messo in pratica quanto appreso alla lezione precedente riguardo allo sviluppo e all'esecuzione di una nuova applicazione per Android.

Abbiamo inoltre iniziato a prendere dimestichezza con le classi Context, Activity, View e TextView.