Android
Corso Introduttivo

App programming

Andrea Tortorella - Developer

Obiettivi

Imparare l'architettura, le API, e i tool di sviluppo su Android.

Acquisire pattern di programmazione adatti a dispositivi resource constrained in java.

Realizzare un'applicazione completa.

Materiale

- Codice <https://github.com/eliantor/AndroidCourse>
- Groups <https://groups.google.com/forum/#!forum/androidcourses>
- Slide <http://eliantor.github.io/AndroidCourse>
- Riferimenti, libri, link contenuti video ecc

*il materiale verrà aggiornato lezione per lezione

Prima lezione

Android OS e introduzione al framework

Storia

- 2003 Nasce il progetto da Android Inc. fondata da Andy Rubin e Rich Miner
- 2005 Acquisizione Google
- 2007 Open Handset Alliance
- 2008 HTC Dream (T Mobile G1) rilasciato il 22 ottobre
- 2009 2.8% del market share
- 2011 +4.4% di crescita per settimana
- 2013 64% del mercato mobile
- 2013+ Android Everywhere: GoogleTV, Project Glass, Media players, Android@Home, Project Shield...

SPECCHI!!

...e fotocamere

Andy Rubin

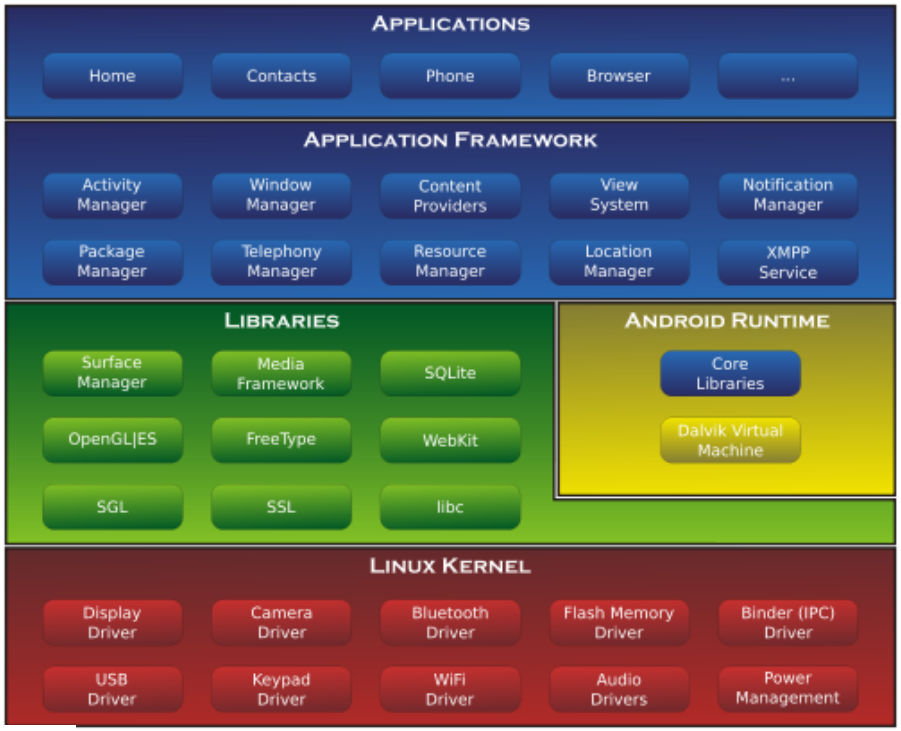
Mindset

Di cosa devo tener conto quando scrivo un'app android

- - Intero sistema operativo a disposizione (Api potenti)
  - Tutte le app sono create uguali
  - App come servizi integrati
  - Applicazioni event driven e context based (gps, sensori, touch,audio,camera)
  - Varietà di scelta dei dispositivi
- - Risorse limitate (memoria, batteria, banda)
  - Use cases complessi, le app non hanno un main
  - Callback hell
  - Frammentazione del mercato

Architettura

- Linux (Experimental branch start merge mainline 3.8)
- Binder IPC (driver al cuore del funzionamento di android)
- Dalvik (vm a registri che esegue dex bytecode)
- System Services (WindowManager, Audio, Networking, Telephony ...)
- SDK libs (le API a nostra disposizione)
- Java (versione 6)



Building blocks

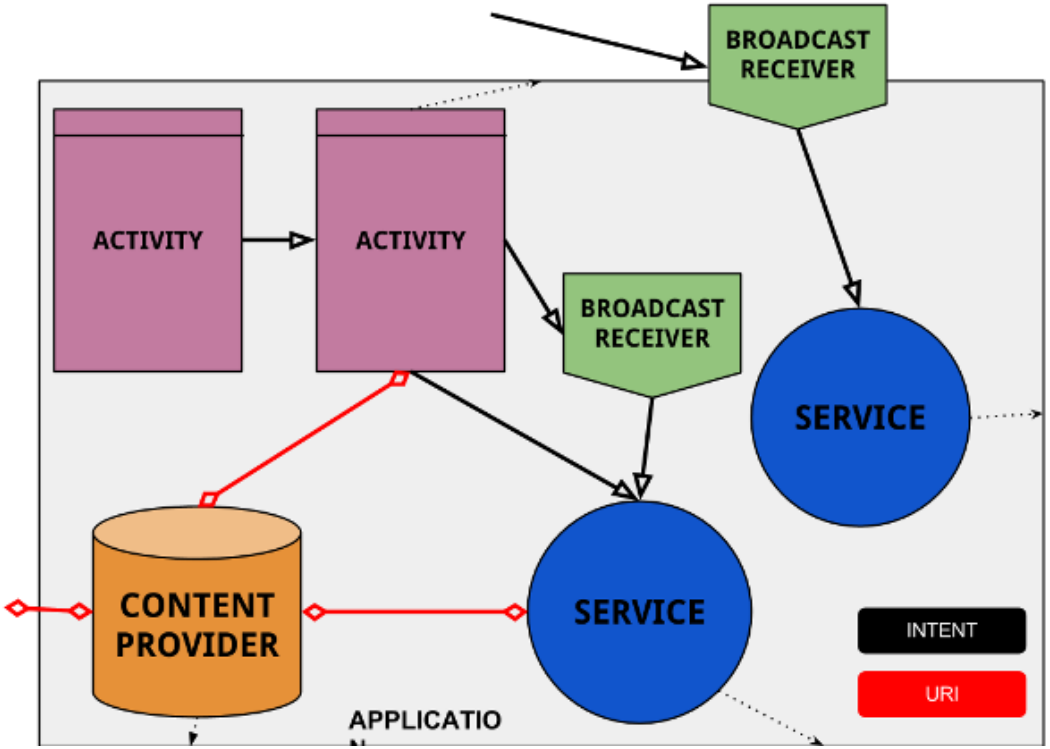
intro

Un'app android é costituita da componenti attivi (lousely coupled) e da un insieme di risorse.
Questi pezzi disaccoppiati sono uniti insieme a runtime in un'unica Application (Context) da un file di configurazione manifest.

Una caratteristica fondamentale di Android é che un app puó attivare direttamente un componente di un'altra applicazione, inviando messaggi intent o richiamando specifiche uri. Il manifest specifica quali messaggi un componente é in grado di ricevere tramite specifici filtri.

Building blocks

components



App basics

Organizzazione

- src/ (contiene il codice java)
- res/ (contiene le risorse dell'app)
- assets/ (file system interno readonly)
- gen/ (codice generato dai tool)
- Il manifest

XML

```xml
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="corso.sample"
  android:versionCode="1"
  android:versionName="1.0" >
  <uses-sdk
    android:minSdkVersion="9"
    android:targetSdkVersion="17"/>
  <application
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name">
  </application>
</manifest>
```

TOOLING

Activity

Activity

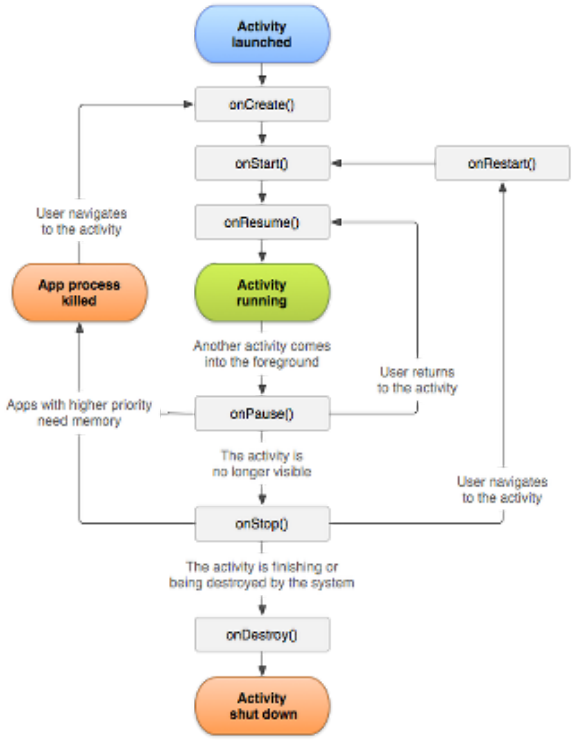
intro

Un'activity rappresenta una schermata (di solito copre l'intera finestra) con cui l'utente puó interagire per realizzare un'azione.

- Pattern MVC
- Attivata da un'intent
- Ha un layout (vista), tipicamente associato tramite una risorsa xml
- Puó far partire altre activity
- Piú activity costituiscono un task
- Ciclo di vita gestito dal sistema
- Puó contenere fragments e loaders

Activity

lifecycle



- Un'activity puó essere distrutta dal sistema per recuperare memoria
- L'intero processo dell'applicazione puó essere terminato dopo onPause()
- Un'activity puó attraversare diverse volte i metodi tra onCreate() e onDestroy() nel suo ciclo di vita
- Dobbiamo preoccuparci di rilasciare le risorse durante i metodi di terminazione

Activity

code

JAVA

```java
package com.corso.sample.activity;
import com.corso.sample.R;
import android.app.Activity;

public class ExampleActivity extends Activity {
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // The activity is being created.
    setContentView(R.layout.my_layout);
  }
}
```

XML

```xml
<application>
  <activity android:name=".activity.ExampleActivity"
      android:label="@string/app_name"
      android:icon="@drawable/ic_launcher">
      <!-- ..... -->
  </activity>
</application>
```

Activity

more code

XML

```xml
<?xml version="1.0" encoding="utf-8"?>
<!-- res/layout/mylayout.xml -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical">
  <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/hello"/>
</LinearLayout>
```

XML

```xml
<resources>
  <string name="app_name">Sample</string>
  <string name="hello">Hello world!/string>
</resources>
```

CODE

Wiews

Gestire l'interfaccia

Dobbiamo poter recuperare le viste nel codice

XML

```xml
<TextView
  android:id="@+id/tv_hello"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="@string/hello"/>
```

JAVA

```java
TextView mHello;
@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.my_layout);
  mHelloOut = (TextView)findViewById(R.id.tv_hello);
}
```

Wiews

Gestire l'interfaccia

Alcune viste hanno associato un comportamento

JAVA

```java
View mInteractive;

@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.my_layout);
  mInteractive = findViewById(R.id.btn);
  mInteractive.setOnClickListener(new OnClickListener{
    @Override
    public void onClick(View v){
      //do something
    }
  });
}
```

Wiews

Gestire l'interfaccia

Alcune viste contengono dati

JAVA

```java
//....
@Override
public void onCreate(Bundle savedInstanceState) {
  //....
  mInteractive = (EditText)findViewById(R.id.btn);
  mReplaceButton.setOnClickListener(this);
}
@Override
public void onClick(View v){
  if(v.getId()==mReplaceButton.getId()){
    mSavedContent = replaceContent(mSavedContent);
  }
}
//...

private String replaceContent(String content){
  Editable content =mInteractive.getText();
  mInteractive.setText(content);
  return content.toString();
}
```

CODE

Activity

Mantenere lo stato

Il sistema operativo puó interrompere il nostro processo

- Per recuperare memoria o durante un cambio di configurazione (Es. rotazione dello schermo)
- Dobbiamo salvare lo stato in modo persistente, (il processo viene deallocato)

JAVA

```java
private final static String SAVED_KEY="SAVED_KEY";
@Override
public void onCreate(Bundle savedInstanceState) {
  if(savedInstanceState!=null){
    //activity restarted
    mState =  savedInstanceState.getBoolean(SAVED_KEY);
  }else{
    mState = initializeState();
  }
}
@Override
protected void onRestoreInstanceState(Bundle savedInstanceState){/*or here after onStart()*/}
@Override
protected void onSaveInstanceState(Bundle outState){
  outState.putBoolean(SAVED_KEY,mState)
}
```

Activity

Navigazione

- Creare link tra activity
- Delegare l'esecuzione di un task ad un'altra activity e riceverne il risultato
- Ricordiamo che la comunicazione avviene sempre tramite intent

Activity

Navigazione 2

Creare link tra activity

```java
private void launch(boolean implicit){
  final Intent intent;
  if(implicit){
    intent = new Intent(this,AnotherActivityInMyPackage.class);
  }else{
    intent = new Intent("an.explicit.action");
  }
  intent.setData(Uri.parse("mydata://somedata/1000"));
  intent.putExtra("A_KEY",1000);
  this.startActivity(intent);
}
```

Delegare l'esecuzione di un task

```java
private final static int MY_REQUEST_CODE = 1;
//...
this.startActivityForResult(intent,MY_REQUEST_CODE);
```

Activity

Navigazione 3

Creare link tra activity

JAVA

```java
@Override
public void onCreate(Bundle savedInstanceState){
    handle(getIntent());
}
@Override
protected void onNewIntent(Intent intent){
    setIntent(intent); //Optionally
    handle(intent);
}
private void handle(Intent intent){}
```

Delegare l'esecuzione di un task

JAVA

```java
setResult(RESULT_OK,new Intent().putExtra("content",response));
//data is optional
//setResult(int x); RESULT_OK RESULT_CANCELED RESULT_FIRS_USER
finish();
```

Activity

Navigazione 4

Ricevere il risultato

JAVA

```java
private final static int MY_REQUEST_CODE = 1;
//...
@Override
protected void onActivityResult (int requestCode, int resultCode, Intent data){
  if(requestCode==MY_REQUEST_CODE){
    if(resultCode==RESULT_OK){
      //do something with data may be null
    }else{
      //do something when user refused to complete action
    }
  }else{
    // not my business another request
  }
}
```

CODE

JAVA

```
e;


tView,View theList){
.item_layout,theList,false);
```

JAVA

JAVA

CODE

Seconda lezione

Fragments

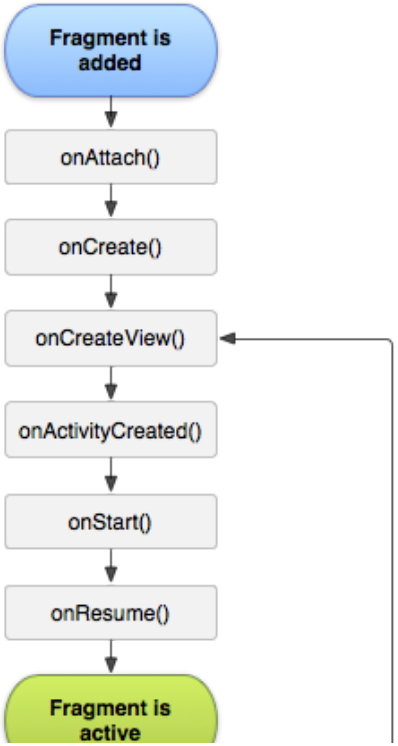
Fragments

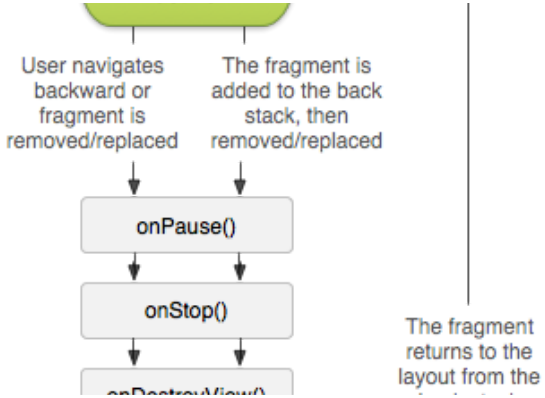
Introduzione

- Introdotti in android 3.0 per supportare i tablet
- Decompongono un'activity in sottocomponenti
- Permettono il riutilizzo del codice
- NON sono componenti di Android nel senso classico ma classi di supporto del framework
- Possono essere usati su versioni piu' vecchie tramite libreria statica

Fragments

Introduzione 2





Fragments

Code

JAVA

```java
public class MyFragment extends Fragment {
  @Override
  public View onCreateView(LayoutInflater inflater,ViewGroup container,Bundle savedInstanceState){
    View v = inflater.inflate(R.layout.my_fragment,container,false);
    // ... setup
    return v;
  }

  @Override
  public void onActivityCreated(Bundle savedInstanceState){
    // ...
  }
  // ...
}
```

XML

```xml
<!-- in the layout for the activity -->
<!-- .... -->
<fragment class="com.example.MyFragment"
          android:id="@+id/MyFragmentId"
```

```
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>
```


Fragments

Code

JAVA

```
public class MyActivity extends FragmentActivity{
    @Override
    public void onCreate(Bundle savedInstanceState){
        setContentView(R.layout.my_activity);
        //...
        FragmentManager manager = getSupportFragmentManager();
        MyFragment f = (MyFragment)manager.findFragmentById(R.id.MyFragmentId);

    }
}
```

CODE

Fragment Dinamici

Introduzione

- I fragment possono non essere dichiarati staticamente nei layout
- In questo caso possono non avere una vista associata

JAVA

```
// in the activity

private final static String TRANSACTION_TAG = "A TAG";

//...

MyFragment f = new MyFragment();

FragmentManager m = getSupportFragmentManager();
m.beginTransaction()
 .replace(R.id.viewgroup,f,TRANSACTION_TAG)
 .addToBackStack(null)
 .commit();

//..

m.findFragmentByTag(TRANSACTION_TAG);
```

Fragment

... non finisce qui

- I fragment si rivelano utili in molte situazioni, diverse da quelle per cui sono stati progettati.
- Possiamo separare il ciclo di vita da quello dell'activity

```java
setRetainInstance(true);
```

- Utile per salvare stato dell'applicazione complesso **Fragment Memory Card**
- Utile supporto per la concorrenza: fa da bridge con altri thread.

CODE

Terza lezione

Persistenza

Persistenza

Alternative

- File system
- Preferenze
- Database

Persistenza

File system

- Internal private storage

```java
String FILENAME = "myprivatefile";
FileInputStream in =context.openFileInput(FILENAME,Context.MODE_PRIVATE);
//...
FileOutputStream out = context.openFileOutput(FILENAME,Context.MODE_PRIVATE);
```

- Caching

```java
File f =context.getCacheDir(); // i file possono essere rimossi dal sistema
```

- External storage

```java
// controllare se abbiamo un sd
Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED);
// aprire la nostra root sull'sd esterna
File f =Environment.getExternalFilesDir(); // /Android/data/mio.package/files/
File f =Environment.getExternalStoragePublicDirectory(DIRECTORY_MUSIC); //shared
```

Persistenza

Preferences

Specifiche per activity o globali (con nome)

Salvate come file xml nello storage interno

JAVA

```java
SharedPreferences localPrefs = activity.getPreferences();

SharedPreferences prefs = context.getSharedPreferences("PREFS_FILE",MODE_PRIVATE);
boolean myOptionalPref = context.getBoolean("onOffPreference",/*default*/false);
String myOptionalText = context.getString("textPreference",/*default*/"fallback");
//...
SharedPreferences.Editor editor = prefs.edit();
editor.putBoolean("key",false)
      .putInt("intKey",69)
      .commit():
```

Terza lezione

Database

Database

SQLite

SQLite é un database embedded (nessuna connessione tramite jdbc) gira nel nostro stesso processo

Non ha tutte le funzionalitá di un database full fledged

Il db é un singolo file

Non é tipizzato: una colonna puó contenere qualsiasi tipo

Manca di alcune features importanti come gli outer join e i foreign constraints sono disabilitati di default

Pessima concorrenza

Nonostante tutto ottimo per applicazioni embedded

Database

SQLite

Creiamo un db tramite SQliteOpenHelper

JAVA

```
public class TodoOpenHelper extends SqliteOpenHelper{
   TodoOpenHelper(Context context){
        super(context,DATABASE_FILE,/*CursorFactory*/null,DATABASE_INT_VERSION);
   }

   public void onCreate(SQLiteDatabase db) {
     db.execSql(CREATE_TABLE_SQL);
   }

   public void onUpgrade(SQLiteDatabase db,int oldVersion,int newVersion) {
     db.execSql(ALTER_TABLE_SQL);
   }
    //....
}
```

Database

Interazione

JAVA

```java
TodoHelper helper = ...;
SQLiteDatabase db =helper.getWritableDatabase();

ContentValues values = new ContentValues(); // riga da inserire
long newid=db.insert("table_name",null,values);
int numUpdates=db.update("table_name",null,values,"_id = ?",new String[]{"1"});
int numDeletes=db.delete("table_name","_id = ?",new String[]{"2"});
Cursor cursor =db.query("table_name",
                        new String[]{"_id","text"},"_id = ?",new String[]{"3"},
                        /*groupby*/null,/*having*/null,/*orderby*/null,/*limit*/null);

int count = cursor.getCount();
if(cursor.moveToFirst()){
    String name = cursor.getString(cursor.getColumnIndexOrThrow("name"));
}

cursor.close();
```

Terza lezione

Content Providers

ContentProviders

Introduzione

developers.android.com dice "you don't need content providers if you don't want to share content with other apps but..."

- Vi serve se volete implemetare facilmente search e suggerimenti
- Vi serve se volete implementare il drag & drop con le altre app
- Vi serve se volete sfruttare al meglio gli altri componenti del framework
- Vi serve se volete sincronizzare i dati in modo facile con la rete
- Vi serve se volete strutturare bene un app

Content Providers

Elementi essenziali

I content provider implementano un'architettura client/server

Client: ContentResolver <-> Server: ContentProvider

Un CP risponde ad una o piú AUTHORITY

Si comunica con un CP tramite uri nella forma **content://authority/path**

I CP possono implementare schemi complessi di permessi

API ottimizzata per dati tabulari (SQL) o file, ma nessun altro enforcement per il backend

Possono auto sincronizzarsi con la rete

Consentono di creare ui reattive

Content Providers

Api client side

JAVA

```java
// lato client stessa api di sqlite con in piú l'uri
ContentResolver resolver =context.getContentResolver();
Cursor cursor =resolver.query(uri,projection,selection,selectionArgs,null,null,null);
// o per l'apertura di file
InputStream input =resolve.openInputStream(uri);
```

JAVA

```java
// il content provider
public class TodoProvider extends ContentProvider{
    public boolean onCreate(){}

    public Cursor query(Uri uri,String[] projection,String selection,...);
    public Uri insert(Uri uri,ContentValues values);
    public int delete(Uri,...);
    public int update(Uri uri,...);
    public String getType(Uri uri);
    public ParcelFileDescriptor openFile(Uri uri,String mode);
}
```

XML

```xml
<!--nel manifest-->
<provider android:authorities="com.jdk.todo.provider"
            android:readPermission="com.jdk.permissions.READ_TODO"
            android:writePermission="com.jdk.permissions.WRITE_TODO">>
</provider>
```


CODE

Terza lezione

Concorrenza

Concorrenza

Necessaria...

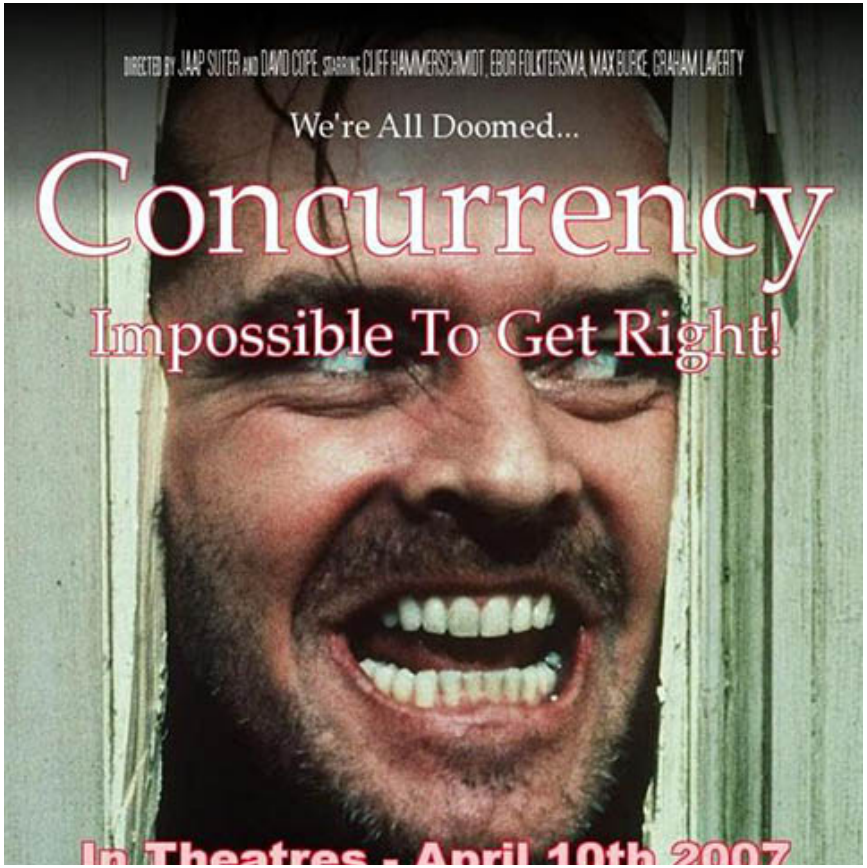
- L'event loop principale deve essere libero da operazioni costose
- Le operazioni di I/O bloccano il thread su cui vengono eseguite
- In particolare l'accesso alla rete
- Android su questo é molto severo:

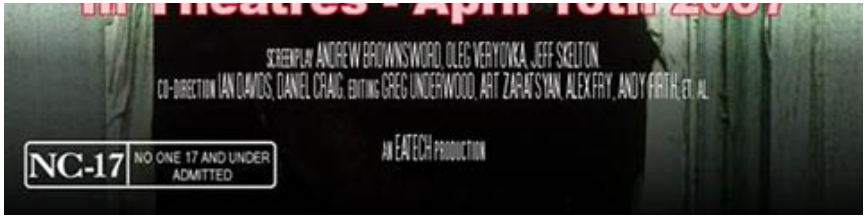
JAVA

```
05-14 23:52:47.258:
                    E/AndroidRuntime(21329): FATAL EXCEPTION: main
05-14 23:52:47.258: E/AndroidRuntime(21329): android.os.NetworkOnMainThreadException
05-14 23:52:47.258: E/AndroidRuntime(21329):    at
                    android.os.StrictMode$AndroidBlockGuardPolicy.onNetwork(StrictMode.java:1126)
05-14 23:52:47.258: E/AndroidRuntime(21329):    at
                    java.net.InetAddress.lookupHostByName(InetAddress.java:385)
05-14 23:52:47.258: E/AndroidRuntime(21329):    at
                    java.net.InetAddress.getAllByNameImpl(InetAddress.java:236)
05-14 23:52:47.258: E/AndroidRuntime(21329):    at
                    java.net.InetAddress.getAllByName(InetAddress.java:214)
05-14 23:52:47.258: E/AndroidRuntime(21329):    at
```

```
libcore.net.http.HttpConnection.<init>(HttpConnection.java:70)
....
```


Concorrenza





Concorrenza

Componenti essenziali

L'unitá di concorrenza in java é il Thread

Ogni thread esegue un path di esecuzione indipendente. In modo concorrente, (realmente concorrente se siamo su un multi core)

Un Thread é anche una radice per quanto concerne il garbage collector, la concorrenza é la causa dei memory leak

I thread hanno bisogno di comunicare tra loro e sincronizzarsi

Concorrenza

Threads

Uso diretto dei thread

- Completa gestione del multithreading
- Non gestiscono il ciclo di vita dei componenti android
- Non facilitano update della ui

Threads

Quando usarli

- Necessitiamo un controllo diretto e preciso del multithreading
- Conosciamo esattamente il comportamento della concorrenza nella nostra applicazione
- Sappiamo come sincronizzarli e metterli in comunicazione, (lock, semafori, mutex, atomic CAS)

...ad esempio...

Chiaro?

Concorrenza

... continua...

Threads e lock, anche se a volte necessari, sono strumenti di basso livello.

In particolare, non possiamo comunque bloccare il thread della ui, in attesa di un mutex

Ci servono altre primitive

atomic CAS

....ehr

Threads

Esempio

JAVA

```
private TextView mTv;

protected void onCreate(Bundle savedInstanceState) {
   //...
   mTv = (TextView)findViewById(...);
   new Thread(){
     public void run() {
        final String result = obtainStringFromBlockingSource();
        // mTv.setText(result); -> Exception ui update from non Main Thread
        runOnUiThread(new Runnable(){
            public void run(){
                mTv.setText(result);
            }
        })
```

```
        }
    }.start();
}
```


Threads

Problemi

Leak! Il thread (non statico) contiene un riferimento all'activity, il garbage collector non puó eliminarla. Rischio di OutOfMemory

mTv non é piú visibile dopo un cambio di configurazione

Strategia: bloccare il thread in onDestroy o onStop e riprendere il lavoro in onStart()

Minimizzare il rischio di thread non interruptible, ad esempio timeout, controllare InterruptedException

Minimizzare il rischio di memory leak, disaccoppiando l'activity dal thread: retained fragments

Ancora meglio:

Non usate i thread nelle activity a meno che non siate sicuri di quello che state facendo

sicuri

Soluzioni

AsyncTask

Facilita l'interazione con la ui

Sconsigliabile per operazioni di lunga durata.

Non gestisce il cilco di vita delle activity.

JAVA

```
public class Task extends AsyncTask<Params,Progress,Result>{
    protected void onPreExceute(){/* OPT sul thread della ui prima di essere avviato*/}

    protected Result doInBackground(Params... params){
        //REQ esegue il lavoro in background
        publishProgress(Progress...); // invia risultati parziali al thread della ui
    }

    protected void onProgressUpdate(Progress... progress){
        //OPT callback di publishProgress()
    }
    protected void onPostExecute(Result res){
```

```
            //OPT riceve il valore di ritorno da doInBackground sulla ui
        }
    }
```


...continua

JAVA

```
public AsyncTask execute(Params ... params); //avvia il task

public final boolean cancel(boolean interrupt); //aborts

public final boolean isCancelled(); // is aborted

public void onCancelled(){ } // callback di cancel
```

AsyncTask

Valutazione

- Meglio dei thread
- Leak ancora presenti
- Non usare inner classes non statiche
- Usare weak reference per aiutare il garbage collector
- Cancellare prima di onDestroy
- Disaccopiare dall'activity tramite retained Fragments

CODE

Concorrenza

L'engine di Async Task

Async Task sfrutta al suo interno tre componenti basilari di android

- Looper: l'astrazione di un ciclo while true associato ad un thread
- MessageQueue: coda di messaggi associata ad un looper
- Handler: gestori di messaggi associati al looper
- Sul main thread gira il looper principale
- Dei tre l'unico che si usa direttamente nel codice utente con una certa frequenza é l'handler

Concorrenza

Handler

JAVA

```
private final static int ARG = 1;
private final static int EVENT_1 = 1;

public MyHandler extends Handler{
    @Override
    public void handleMessage(Message msg) {
        switch(msg.what) {
            case EVENT_1:
                handleEvent(msg.obj,msg.arg1,msg.arg2);
                break;
            //...
        }
    }
}

MyHandler handler = new MyHandler();
//...

Message msg =handler.obtainMessage(EVENT_1,ARG,ARG,"ciao");
handler.sendMessage(msg);
```

```
handler.sendMessageDelayed(handler.obtainEmptyMessage(EVENT_1),1000);
handler.post(new Runnable(){/*code to execute*/})

handler.removeMessages(EVENT_1);
```


Concorrenza

Alternative

AsyncQueryHandler

Wrappa un contentResolver: le operazioni sono effettuate serialmente su un looper in background

JAVA

```
private final static int QUERY_ID = 1;

private static class ContentHandler extends AsyncQueryHandler{
    protected void onQueryComplete(int token,Object cookie,Cursor result){
        // callback
    }
}
ContentHandler h = new ContentHandler(getContentResolver());

h.startQuery(QUERY_ID,/*cookie*/null,uri,....);
```

Concorrenza

Alternative

HandlerThread un thread su cui gira un looper

JAVA

```java
public class MyLoopingThread extends HandlerThread{

    public MyLoopingThread(){
        super("name for debugging purposes");
    }

    @Override
    protected void onLooperPrepared() {
        // do some setup before start looping
    }

    public Handler createHandler(Handler.Callback callback){
```

```
            return new Handler(getLooper(),callback);
        }
    }
```


Concorrenza

Loaders

I loader permettono di caricare dati in background in modo sicuro

- Effettuano un task in background (come async task)
- Restituiscono il risultato sul main thread (come async task)
- Continuano ad eseguire il task durante un cambio di configurazione (come async task)
- Ricordano il loro stato tra una configurazione e l'altra
- Possono monitorare una sorgente dati (CP) e notificare l'activity dei cambiamenti.
- rileggete bene il punto precedente
- rileggetelo ancora

Concorrenza

Loaders Continua

- Presenti dall'api 11
- Usabili con la supprt library dall'api 4
- Non possono essere implementati come classi interne non statiche (policy enforced)
- due componenti
  - Loader: che si occupa di fare il lavoro
  - Loader.Callbacks: che vengono notificati dal loader quando l'operazione é conclusa
  - Gestiti tramite LoaderManager
- Ci interessa in particolare il CursorLoader

Concorrenza

Loaders Continua

JAVA

```
private final static int ID = 1;
//il loader manager avvia un loader identificato da un id se e solo se non
//ne esiste giá uno precedente (anche in una configurazione precedente)
getSupportLoaderManager().initLoader(ID,null,fCallbacks);
// fCallbacks viene notificato quando il risultato é realizzato

 //kill currently running loader notify if already completed
getSupportLoaderManager().destroyLoader(ID);

// praticamente la somma dei due precedenti
getSupportLoaderManager().restartLoader(ID,null fCallbacks
```

CODE