Lezione 12

Programmazione Android

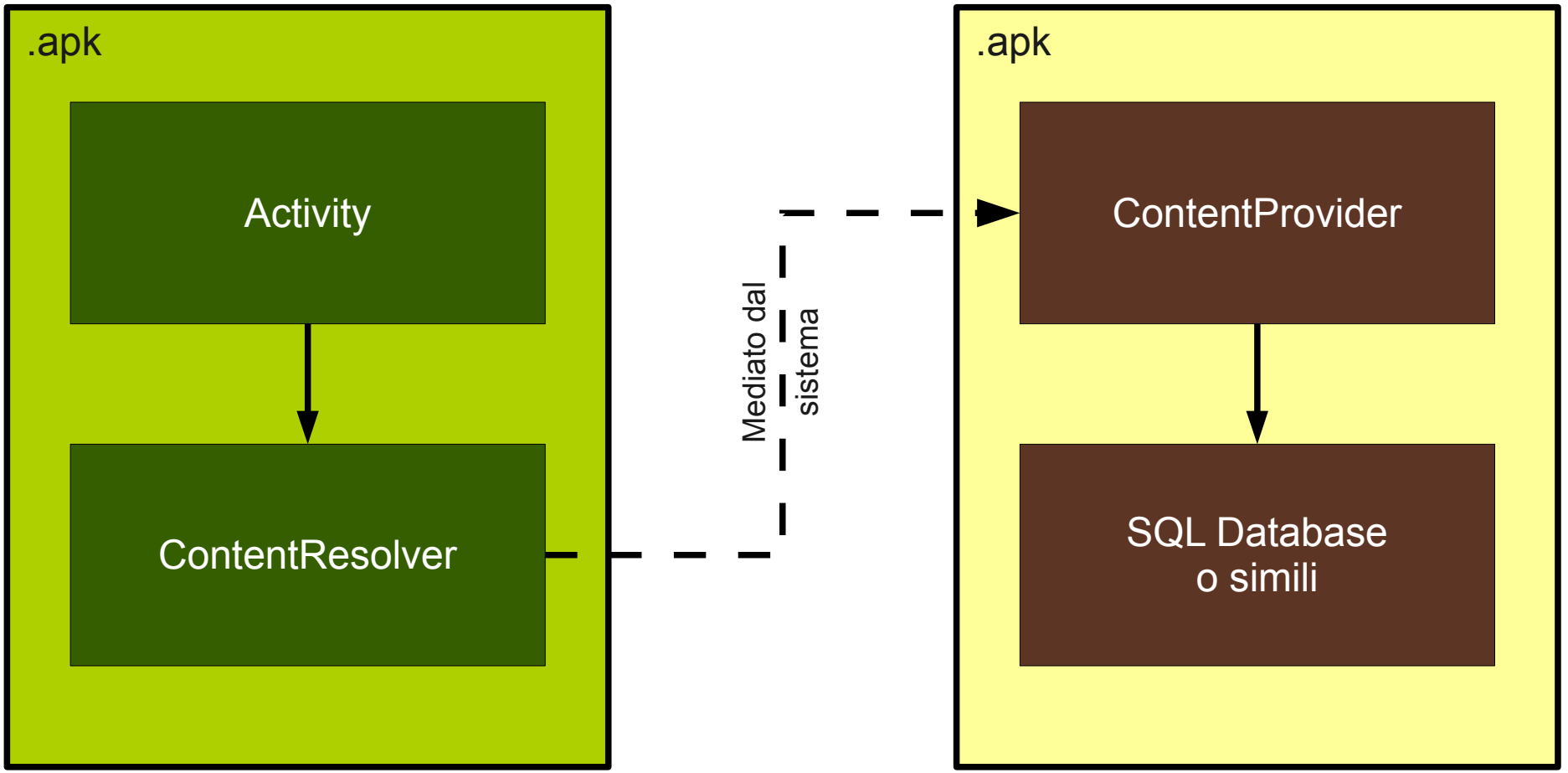
- Storage temporaneo
  - **Salvataggio temporaneo dello stato**
- Storage permanente
  - **Preferenze**
    - Shared & Private Preferences
    - PreferenceScreen e PreferenceActivity
  - **Accesso al File System**
  - **Accesso a Database**
- Condivisione di dati
  - **Content Provider**

Content Provider

Content Provider

- Abbiamo visto che le applicazioni Android possano utilizzare file e DB SQL per la memorizzazione
- In generale, si vuole poter **condividere** i dati fra più applicazioni indipendenti
  - In maniera universale ma controllata
- Due aspetti
  - Accedere ai dati resi disponibili da altri
  - Rendere i propri dati accessibili agli altri

Accesso a Content Provider



Leggere dati da un Content Provider

```
ContentResolver cr=getContentResolver();

Uri uri =MediaStore.Images.Media.
        EXTERNAL_CONTENT_URI;
        /* content://media/external/images */

Cursor c=cr.query(uri,
    null,     /* colonne */
    null,     /* selezione */
    null,     /* args */
    null      /* sort */
);

String [] cols = c.getColumnNames();
for (String col: cols) {
   Log.d("TCA",col);
}

Log.d("TCA","# rows = "+c.getCount());
```

```
02-16 18:47:13.327: D/TCA(21998): _id
02-16 18:47:13.327: D/TCA(21998): _data
02-16 18:47:13.327: D/TCA(21998): _size
02-16 18:47:13.327: D/TCA(21998): _display_name
02-16 18:47:13.327: D/TCA(21998): mime_type
02-16 18:47:13.327: D/TCA(21998): title
02-16 18:47:13.327: D/TCA(21998): date_added
02-16 18:47:13.327: D/TCA(21998): date_modified
02-16 18:47:13.327: D/TCA(21998): description
02-16 18:47:13.327: D/TCA(21998): picasa_id
02-16 18:47:13.327: D/TCA(21998): isprivate
02-16 18:47:13.327: D/TCA(21998): latitude
02-16 18:47:13.327: D/TCA(21998): longitude
02-16 18:47:13.327: D/TCA(21998): datetaken
02-16 18:47:13.327: D/TCA(21998): orientation
02-16 18:47:13.327: D/TCA(21998): mini_thumb_magic
02-16 18:47:13.327: D/TCA(21998): bucket_id
02-16 18:47:13.327: D/TCA(21998): bucket_display_name
02-16 18:47:13.327: D/TCA(21998): puid
02-16 18:47:13.327: D/TCA(21998): protect_status
02-16 18:47:13.327: D/TCA(21998): use_count
02-16 18:47:13.327: D/TCA(21998): date_used
02-16 18:47:13.327: D/TCA(21998): rating
02-16 18:47:13.327: D/TCA(21998): width
02-16 18:47:13.327: D/TCA(21998): height
02-16 18:47:13.327: D/TCA(21998): dlna_profile
02-16 18:47:13.327: D/TCA(21998): dlna_share
02-16 18:47:13.327: D/TCA(21998): maker
02-16 18:47:13.327: D/TCA(21998): # rows = 762
```

Leggere dati da un Content Provider

- Una volta ottenuto un Cursor, si procede come di consueto
  - while (!cur.isAfterLast()) { … = cur.getString(3); … }
- **Nota**:
  - Il metodo query() del ContentResolver può richiedere un tempo significativo
  - **MAI** eseguire operazioni lunghe nel thread della UI!
  - La classe di utilità CursorLoader viene in aiuto
    - Caricamento asincrono dei risultati di una query
    - La vedremo successivamente

Leggere dati da un Content Provider

- Per procurarsi l'URI, ci sono vari metodi
  - I Content Provider di sistema offrono spesso costanti predefinite
    - MediaStore.Images.Media.INTERNAL_CONTENT_URI
    - UserDictionary.Words.CONTENT_URI
    - Contacts.People.CONTENT_URI
    - MediaStore.Audio.Albums.EXTERNAL_CONTENT_URI
    - MediaStore.Video.Thumbnails.EXTERNAL_CONTENT_URI
    - VoicemailContract.Vocemails.CONTENT_URI
    - …
  - Per gli altri, ci si affida alla documentazione!

Il formato delle URI

- Le URI usate nei content provider hanno un formato noto
  - Un'intera tabella:
    - **content:**//media/internal/images/
  - Uno specifico record:
    - **content:**//media/internal/images/45

path: identifica la tabella

schema

authority: identifica il provider

id: identifica la riga

Il formato delle URI

- La classe **android.net.Uri** fornisce metodi di utilità per costruire e convertire URI
- In particolare,
  - Uri Uri.**parse**(String s)
    - data una stringa, costruisce l'Uri relativa
  - Uri Uri.**withAppendedPath**(Uri base, String path)
    - Data una Uri, vi aggiunge in fondo un componente (tipicamente, è il numero di riga)
- Altri metodi sono utili per confrontare o spezzare Uri nella parti componenti

Accesso in scrittura ai dati di un Content Provider

- Se si dispone dei giusti permessi, è possibile anche modificare righe esistenti, o aggiungerne di nuove
- Si tratta sempre di operazioni **richieste** al ContentProvider
  - Starà a lui decidere se e come implementarle
  - In nessun caso si può accedere ai dati sottostanti
- Tipicamente (ma non sempre), le operazioni vengono riflesse su una tabella SQL sottostante

Accesso in scrittura ai dati di un Content Provider

- Per inserire un nuovo record, si crea un oggetto ContentValues (mappa colonne-valori) e si invoca il metodo **insert()** del ContentResolver

```
ContentValues cv = new ContentValues();
cv.put("Name","Vincenzo");
cv.put("Surname","Gervasi");
cv.put("Age", 43);

Uri newrow = cr.insert(uri,cv);
                // campo _ID è aggiunto automaticamente
```

Uri della tabella

Uri del nuovo record

Accesso in scrittura ai dati di un ContentProvider

- Per modificare uno o più record esistenti, si usa il metodo **update()** del ContentResolver
  - Simile a UPDATE … WHERE … in SQL

```
ContentValues cv = new ContentValues();
cv.put("Age", 44);

int n = cr.update(uri, cv, "CF=?", args);
```

- Analogamente per cancellare uno o più record

```
int n = cr.delete(uri, "CF=?", args);
```

Il ruolo dei permessi

- Chi offre un Content Provider **può** richiedere dei permessi nel suo manifesto
  - Con <requires-permission>
- Chi accede a un ContentProvider **deve** usare i permessi nel suo manifesto (se richiesti)
  - Con <uses-permission>
- Lo scopo è quello di garantire che chi usa un ContentProvider “lo conosca” a fondo
  - Per esempio, deve sapere lo schema delle tabelle
- Il nome del permesso fa un po' da “password”

Definire un ContentProvider

- Qualunque applicazione può definire un ContentProvider per offrire accesso ai propri dati
- Un ContentProvider è un **componente top-level** dell'applicazione (come le Activity)
- Ha una sua sezione in AndroidManifest.xml

```
<provider android:name="ArcobalenoProvider"
    android:authorities="it.unipi.di.sam.arcobaleno"
    android:exported="true"
    android:enabled="true"
    android:description="@string/abdesc">
</provider>
```

Come al solito, usiamo il nome del package come prefisso → unicità

Definire un ContentProvider

- Si crea una sottoclasse di **ContentProvider**

```
public class ArcobalenoProvider extends ContentProvider {

@Override
public int delete(Uri uri, String selection, String[] selectionArgs) { /* ... */ }

@Override
public String getType(Uri uri) { /* ... */ }

@Override
public Uri insert(Uri uri, ContentValues values) { /* ... */ }

@Override
public boolean onCreate() { /* ... */ }

@Override
public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)
{ /* ... */ }

@Override
public int update(Uri uri, ContentValues values, String selection,String[] selectionArgs) { /* ... */ }

}
```

Definire un ContentProvider

- La parte della URI comprendente schema e authority è gestita dal sistema
- La parte comprendente il path è a discrezione del provider
  - Ma meglio adeguarsi agli usi standard!
- Data una URI “nostra”, getType() deve restituire il suo tipo MIME
  - vnd.it.unipi.di.sam.cursor.**dir**/x-rainbow
  - vnd.it.unipi.di.sam.cursor.**item**/x-rainbow-color

Definire un ContentProvider

- Il nostro ContentProvider avrà il suo ciclo di vita
  - Distinto da quello dell'Activity!
  - Gestito automaticamente dal sistema
    - Viene istanziato quando un ContentResolver deve gestire un'URI la cui authority corrisponde alla nostra
    - Viene chiuso quando non è più necessario
      - Al limite, verrà re-istanziato più avanti
- Il metodo onCreate() deve creare (se non esiste) o aprire (se esiste) il DB corrispondente
  - Solo nel caso (comune) in cui abbiamo un DB!
  - Ricordate **SQLiteOpenHelper**...

Content Provider
Riassunto

- Un'applicazione può definire un ContentProvider
  - Identificato da un'*authority* (parte di URI)
  - Dichiarato in AndroidManifest.xml (con eventuali permessi)
- Un'applicazione può usare un ContentProvider
  - Passa una URI al ContentResolver
  - Quest'ultimo controllo se, nel sistema, è installato un provider in grado di gestire quell'URI
  - Se si, lo istanzia (nel processo del ricevente!) e instrada le richieste query/insert/update/delete

Esempio di uso di Content Provider:

ContactManager

Esercizio

- Estendere l'esempio **AndroidGallery** illustrato nella lezione sugli Adapter
  - In quel caso, avevamo utilizzato un Adapter grezzo che recuperava per suo conto le immagini da rete
  - Modificarlo in modo che le immagini mostrate dalla Gallery provengano dalle foto dell'utente
    - Ottenute dal ContentProvider relativo!
  - Suggerimento: si ricordi che esistono degli adapter di sistema già pronti per lavorare con i Cursor...