**STORAGE**

**In che modo (o modi) varie activity che fanno parte della stessa app possono condividere dati? Si discuta dei vantaggi e svantaggi di ciascuno dei modi descritti.**

Varie activity facenti parte della stessa app possono condividere dati in tre modi: attraverso ***SharedPreferences***, files o basi di dati.

* ***SharedPreferences*** sono delle preferenze (localizzate in opportuni file di minime dimensioni) condivise tra più activity, che consentono la memorizzazione, tipicamente, di valori primitivi (interi, caratteri, booleani, …) attraverso setter e getter. Per ricavare le preferenze condivise, è possibile utilizzare il metodo ***getDefaultSharedPreferences(Context c)*** oppure ***getSharedPreferences(String filename).*** Per salvare informazioni, occorre un oggetto ***SharedPreferences.Editor***, ricavabile attraverso il metodo ***edit()*** sulle preferenze ricavate: su quest’oggetto è possibile invocare i vari setter per salvare dati primitivi; dopo aver inserito i vari dati, occorre effettuare il ***commit()*** dell’editor.
* ***File*** sono il meccanismo più semplice ed efficiente per salvare informazioni (anche dati non primitivi, cioè oggetti serializzabili). I file utilizzabili con Android possono risiedere in tre memorie differenti: interna, esterna e cache; chiaramente, le directory per utilizzare i file nelle tre categorie precedenti sono diverse, ed esistono metodi che consentono di ricavarle in modo semplice. Il meccanismo di lettura/scrittura su file è, ovviamente, quello classico di Java. Per scrivere su file sulla memoria esterna è necessario richiedere l’opportuno permesso. Ogni applicazione in Android ha la sua directory privata, a cui possono accedere le sole activity della stessa applicazione.
* ***Database*** sono delle strutture che consentono la memorizzazione persistente di informazioni. Android consente un utilizzo semplificato (seppur analogamente complesso da utilizzare) di basi di dati relazionali SQL attraverso ***SQLite***.

Vantaggi e svantaggi di ciascuno dei modi descritti dipendono principalmente da tre fattori: dati memorizzabili, semplicità ed efficienza. Le ***SharedPreferences*** possono memorizzare soltanto dati primitivi, sono molto semplici da utilizzare ed efficienti, in quanto trattano dati atomici facilmente ricavabili e memorizzabili in file opportuni di basse dimensioni. I file possono memorizzare qualsiasi oggetto al loro interno (nel caso in cui siano file puramente testuali o binari), sono semplici ed efficienti da utilizzare. I database sono strutture che consentono di memorizzare dati strutturati e relazionati in tabelle (quindi, si parla di dati complessi) persistentemente, sono (non molto) semplici da utilizzare, ma sono spesso inefficienti.

**Si descrivano le varie possibilità offerte da Android per la memorizzazione statica (file) dei dati.**

Per ogni app il sistema operativo prevede una directory privata, solo l’app può accedere a questa directory, all’interno della quale viene scritto un file per la memorizzazione dei dati.

Per creare e scrivere un ***file:***

1. Chiamare openFileOutput(fileName, mode), restituisce un FileOutputStream
2. Scrivere nel file (write())
3. Chiudere lo stream (close())

La modalità di accesso può essere:

* MODE\_PRIVATE (file accessibile solo all’app)
* MODE\_APPEND

Per leggere un file

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

Per i ***file temporanei*** si può usare una directory cache, Android cancellerà i file in questa directory SE necessario (manca spazio).

Android permette l’utilizzo di una ***memoria esterna***, tipicamente una SD card. La memoria esterna può essere rimossa, quindi non si può assumere che i file siano sempre disponibili.

**Si descrivano le SharedPreferences. Cosa sono? Come funzionano? Che differenza c’è fra** **SharedPreferences e Preferences?**

Le ***SharedPreferences*** sono uno dei tre meccanismi utilizzati per effettuare in modo persistente la memorizzazione di dati condivisi tra più activity della stessa applicazione in un file di piccole dimensioni. Le preferenze condivise permettono la memorizzazione di soli dati primitivi (interi, caratteri, floating points, booleani, …); tuttavia, sono molto efficienti e facili da usare.

Per ottenere le preferenze condivise, è possibile usare uno dei seguenti metodi:

* ***getDefaultSharedPreferences(),*** che ricava le preferenze condivise di default dell’applicazione;
* ***getSharedPreferences(String filename),*** che ricava le preferenze condivise nel file specificato.

Per inserire un valore nelle ***SharedPreferences***, occorre dapprima ricavare uno ***SharedPreferences.Editor*** attraverso il metodo ***edit()*** sull’oggetto ***SharedPreferences*** ricavato; in seguito, è possibile chiamare uno dei metodi ***putInt(String tag, int value), putChar(String tag, char value),*** …, sull’editor; dopo aver inserito i dati, occorre effettuare il ***commit()*** dell’editor.

Per ricavare un valore dalle ***SharedPreferences***, semplicemente s’invoca uno dei metodi seguenti sull’oggetto ***SharedPreferences*** ricavato: ***getInt(String tag),*** ***getChar(String tag),*** …

Bisogna fare attenzione quando si invoca il metodo ***getSharedPreferences().*** Esiste, infatti, un metodo ***getPreferences()*** che ricava le preferenze della singola activity, pertanto NON CONDIVISE tra tutte le activity dell’applicazione, a differenza di ***getSharedPreferences().***

**Si consideri il salvataggio di dati tramite le SharedPreferences. Si forniscano gli opportuni frammenti di codice per memorizzare una stringa (String), un intero (Int) ed un intero lungo (Long) e per recuperarli. Si indichi anche dove i frammenti di codice vanno inseriti.**

Per effettuare la memorizzazione permanente di dati primitivi condivisi da varie activity, utilizzare le ***SharedPreferences*** rappresenta una modalità molto semplice.

Solitamente, la memorizzazione avviene prima della distruzione dell’activity, per cui:

- in ***onDestroy(),*** per memorizzarli:

*SharedPreferences prefs = getDefaultSharedPreferences();*

*SharedPreferences.Editor editor = prefs.edit();*

*editor.putString(“stringa”, string); // si assume che string sia stata definita in precedenza*

*editor.putInt(“intero”, integer); // si assume che integer sia stato definito in precedenza*

*editor.putLongInt(“longint”, l\_int);*

*editor.commit()*

- in ***onCreate(),*** per recuperarli:

*SharedPreferences prefs = getDefaultSharedPreferences();*

*String string = prefs.getString(“stringa”);*

*int integer = prefs.getInt(“intero”);*

*long l\_int = prefs.getLongInt(“longint”);*

**A cosa servono le classi DatabaseOpenHelper?**

Per la gestione di tutte le operazioni relative al database, è stata data una classe di supporto che si chiama ***SQLiteOpenHelper***. Gestisce automaticamente la creazione e l'aggiornamento del database. La sua sintassi è riportata di seguito:

|  |
| --- |
| *public class DBHelper extends SQLiteOpenHelper {*  *public DBHelper(){*  *super(context,DATABASE\_NAME,null,1);*  *}*  *public void onCreate(SQLiteDatabase db) {}*  *public void onUpgrade(SQLiteDatabase database, int oldVersion, int newVersion) {}*  *}* |

**Si spieghi il ruolo della classe Cursor per l’utilizzo dei database SQL.**

Contiene il set di risultati di una query effettuata su un database in Android. La classe Cursor ha un'API che consente a un'app di leggere (in modo sicuro dal tipo) le colonne restituite dalla query e di scorrere le righe del set di risultati.

Una volta che un cursore è stato restituito da una query del database, un'app deve scorrere il set di risultati e leggere i dati della colonna dal cursore. Internamente, il cursore memorizza le righe di dati restituiti dalla query insieme a una posizione che punta alla riga di dati corrente nel set di risultati. Quando un cursore viene restituito da un metodo query () , la sua posizione punta allo spot prima della prima riga di dati. Ciò significa che prima di poter leggere qualsiasi riga di dati dal cursore, è necessario spostare la posizione in modo che punti a una riga di dati valida. Ci sono vari metodi per la manipolazione della posizione del cursore.

|  |
| --- |
| *Cursor resultSet = mydatbase.rawQuery("Select \* from TutorialsPoint",null);*  *resultSet.moveToFirst();*  *String username = resultSet.getString(0);*  *String password = resultSet.getString(1);* |

**Che cosa fa il seguente metodo?**

**private Cursor readSelectedEntries() {**

**String[] projection = {**

**SchemaDB.Tavola.\_ID,**

**SchemaDB.Tavola.COLUMN\_NAME,**

**SchemaDB.Tavola.COLUMN\_VOTO**

**};**

**String sortOrder = SchemaDB.Tavola.COLUMN\_VOTO + " ASC";**

**String selection = SchemaDB.Tavola.COLUMN\_NAME + " LIKE ? + " and " + SchemaDB.Tavola.COLUMN\_VOTO + " >? ";**

**String[] selectionArgs = {"Car%", "25"};**

**Cursor cursor = db.query(**

**SchemaDB.Tavola.TABLE\_NAME,**

**projection,**

**selection,**

**selectionArgs,**

**null,**

**null,**

**sortOrder**

**);**

**return cursor;**

**}**