



Développement mobile

PERSISTANCE DES DONNÉES

Stockage de données

- ▶ Android fournit plusieurs types de stockage
 - ▶ Données actives d'une activité (`Bundle`)
 - ▶ Fichier ressources `read-only` (répertoire `res`)
 - ▶ Préférences partageable (`SharedPreferences`)
 - ▶ Le presse papier (`ClipboardManager`)
 - ▶ Système de fichiers
 - ▶ Donnée interne de l'application (mémoire flash)
ou données en cache (résultat de calcul intermédiaire)
 - ▶ Donnée externe partagée (carte SD)
 - ▶ En ligne sur le cloud
 - ▶ Base de données (`SQLite`)

Stockage de données

- ▶ Android fournit plusieurs types de stockage
 - ▶ Données actives d'une activité (Bundle)
 - ▶ Fichier ressources read-only (répertoire res)
 - ▶ **Préférences partageable (SharedPreferences)**
 - ▶ Le presse papier (ClipboardManager)
 - ▶ **Système de fichiers**
 - ▶ Donnée interne de l'application (mémoire flash)
ou données en cache (résultat de calcul intermédiaire)
 - ▶ Donnée externe partagée (carte SD)
 - ▶ En ligne sur le cloud
 - ▶ **Base de données (SQLite3)**

Préférences Partageables

- ▶ `SharedPreferences`

- ▶ Simple
- ▶ Type limité à `boolean`, `float`, `int`, `long` et `String`
- ▶ Différents modes
- ▶ basé sur un/des fichier(s)
- ▶ Les données sont stockées sous forme de paires (clé, valeur)

Préférences Partageables

- ▶ Pour récupérer les préférences de l'application :
`SharedPreferences context.getSharedPreferences(mode)`
- ▶ Pour récupérer les préférences partagées
`context.getSharedPreferences(name, mode)`

`mode: MODE_[PRIVATE | WORD_READABLE |
WORD_WRITABLE]`

- ▶ Pour récupérer les valeurs :
`get[Boolean | Float | Int | Long | String](String key, X defaultValue)`

Préférences Partageables

- ▶ Pour écrire des couples clé (`String`)/valeur
Modification transactionnelle des données
 - ▶ On récupère un éditeur avec `SharedPreferences.Editor edit()`
 - ▶ On fait des `editor.putX(key, value)`
 - ▶ On valide atomiquement les changements `editor.commit()`
ATTENTION: `commit` peut renvoyer `false` !
- ▶ Listener sur les modifications
`registerOnSharedPreferenceChangeListener()`

Préférences Partageables - Exemple

```
package fr.enseirb.mysharedpreferences;

import ...

public class MainActivity extends AppCompatActivity {
    private transient Data data;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Data data = new Data();
        SharedPreferences preferences = getPreferences(MODE_PRIVATE);
        data.load(preferences);
        this.data = data;

        EditText email = (EditText)(findViewById(R.id.email));
        email.setText(data.emailAddress);

        EditText message = (EditText)(findViewById(R.id.message));
        message.setText(data.message);
    }

    static class Data{
        private String emailAddress;
        private String message;

        public void load(SharedPreferences preferences){
            emailAddress = preferences.getString("emailAddress", null);
            message = preferences.getString("message", null);
        }

        public void store(SharedPreferences preferences){
            SharedPreferences.Editor editor = preferences.edit();
            editor.putString("emailAddress", emailAddress);
            editor.putString("message", message);
            editor.commit();
        }
    }
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    tools:context="fr.enseirb.mysharedpreferences.MainActivity">

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/email"/>

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/message"/>

</LinearLayout>
```


Préférences Partageables - Exemple 2

```
private void saveMyColor(String myColor) {  
    SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(this);  
    SharedPreferences.Editor editor = preferences.edit();  
    editor.putString(MY_COLOR_NAME, myColor);  
    editor.commit();  
}
```

```
private String loadMyColor() {  
    SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(this);  
    return preferences.getString(MY_COLOR_NAME, DEFAULT_COLOR);  
}
```


Les fichiers

► Mémoire interne

```
private void writeFile() {
    String myFileName = "myFile";
    String fileContent = "Text in my file";

    FileOutputStream fos;
    try {
        fos = openFileOutput(myFileName, Context.MODE_PRIVATE);
        fos.write(fileContent.getBytes());
        fos.close();
    }
    catch (FileNotFoundException e) {
        displayFileNotFoundError();
    }
    catch (IOException e) {
        displayWritingError();
    }
}
```


Les fichiers

- ▶ Carte mémoire
 - ▶ Permission (READ_EXTERNAL_STORAGE, WRITE_EXTERNAL_STORAGE)
 - ▶ Présence d'un périphérique
 - ▶ Récupération du dossier racine

```
if (Environment.MEDIA_MOUNTED.equals(Environment.getExternalStorageState())) {  
    // Carte mémoire présente  
    File myDirectory = Environment.getExternalStorageDirectory();  
    myDirectory.getPath();  
} else {  
    // Pas de carte mémoire  
}
```


Bases de données - SQLite

- ▶ Moteur de base de données relationnel sur fichier sans support de concurrence
- ▶ Persistance de structure de donnée en tables
 - ▶ Habituellement, 1 ligne 1 objet, un champ par colonne
- ▶ Jointure, index de tri, etc.

Gestions des tables

- ▶ Création

```
CREATE TABLE IF NOT EXISTS gpstrace(  
    date INTEGER PRIMARY KEY,  
    latitude REAL NOT NULL,  
    longitude REAL NOT NULL,  
    altitude REAL) ;
```

- ▶ Création d'index

```
CREATE INDEX IF NOT EXISTS latitude ON gpstrace (latitude);
```

- ▶ Effacement de table

```
DROP TABLE gpstrace;
```


Requêtes

- ▶ Sélection

```
SELECT col1,col2,...,coln  
FROM table  
WHERE expr  
GROUP BY expr  
HAVING expr {UNION, INTERSECT, EXCEPT} SELECT ...
```

- ▶ Insertion

```
INSERT INTO table (col1,col2,...,coln) VALUES (val1,val2,...,valn) ;
```

- ▶ Mise à jour

```
UPDATE table SET col1=val1, col2=val2, ...,coln=valn WHERE expr ;
```

- ▶ Suppression

```
DELETE FROM table WHERE expr
```

- ▶ Expressions ;

- ▶ Opérateurs classiques: | | */%+-<>>& | <=>>===!=NOT

- ▶ Like et regex: colname LIKE x: (% pour .* et _ pour ?) ou colname REGEXP r

SQLiteOpenHelper

- ▶ Helper dont on hérite pour ouvrir une base
 - ▶ Un constructeur :
`SQLiteOpenHelper(context, dbName, null, version)`
 - ▶ Deux méthodes à redéfinir
 - ▶ exécuter le script de création de la base (table + index)
`onCreate(SQLiteDatabase)`
 - ▶ Mettre à jour une base de la version v1 à la version v2
`onUpgrade(SQLiteDatabase, int v1, int v2)`
 - ▶ Obtenir la base en lecture ou lecture/écriture
 - ▶ `get[Readable|Writable]Database()`

SQLiteDatabase

- ▶ Exécuter une requête

- ▶ `Cursor.rawQuery(String query, String[] selectionArgs)`
- ▶ `Cursor.query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy, String limit)`
 - ▶ Bien séparer `selection` et `selectionArgs` pour éviter l'injection de commande SQL

- ▶ Exemples :

- ▶ `Cursor c = db.rawQuery("SELECT date, latitude, longitude FROM gpstraces WHERE latitude > ? ORDER BY date DESC limit 100", new String[] {"45"}) ;`
- ▶ `Cursor cursor = db.query("gpstraces", new String[] {"date", "latitude", "longitude"}, "latitude > ?", new String[]{"45"}, null, null, "date DESC", 100) ;`

Cursor

- ▶ On parcourt les résultats d'une requête avec un `Cursor` (pas thread safe)
 - ▶ Méthodes usuelles
 - ▶ nombre de lignes
`int getCount()`
 - ▶ Valeur de la colonne à l'index de la ligne courante
`get[Short|Int|Long|Float|Double|String|Blob](index)`
 - ▶ Bouger à l'élément prochain ou renvoie false
`boolean moveToNext()`

Ecriture

- ▶ Insertion

- ▶ `db.insert(String table, String nullColumnHack, ContentValues values)`

- ▶ Mise à jour

- ▶ `db.update(String table, ContentValues values, String whereClause, String[] whereArgs)`

- ▶ Suppression

- ▶ `db.delete(String table, String whereClause, String[] whereArgs)`

avec `ContentValues` est une `Map` (column name -> value)