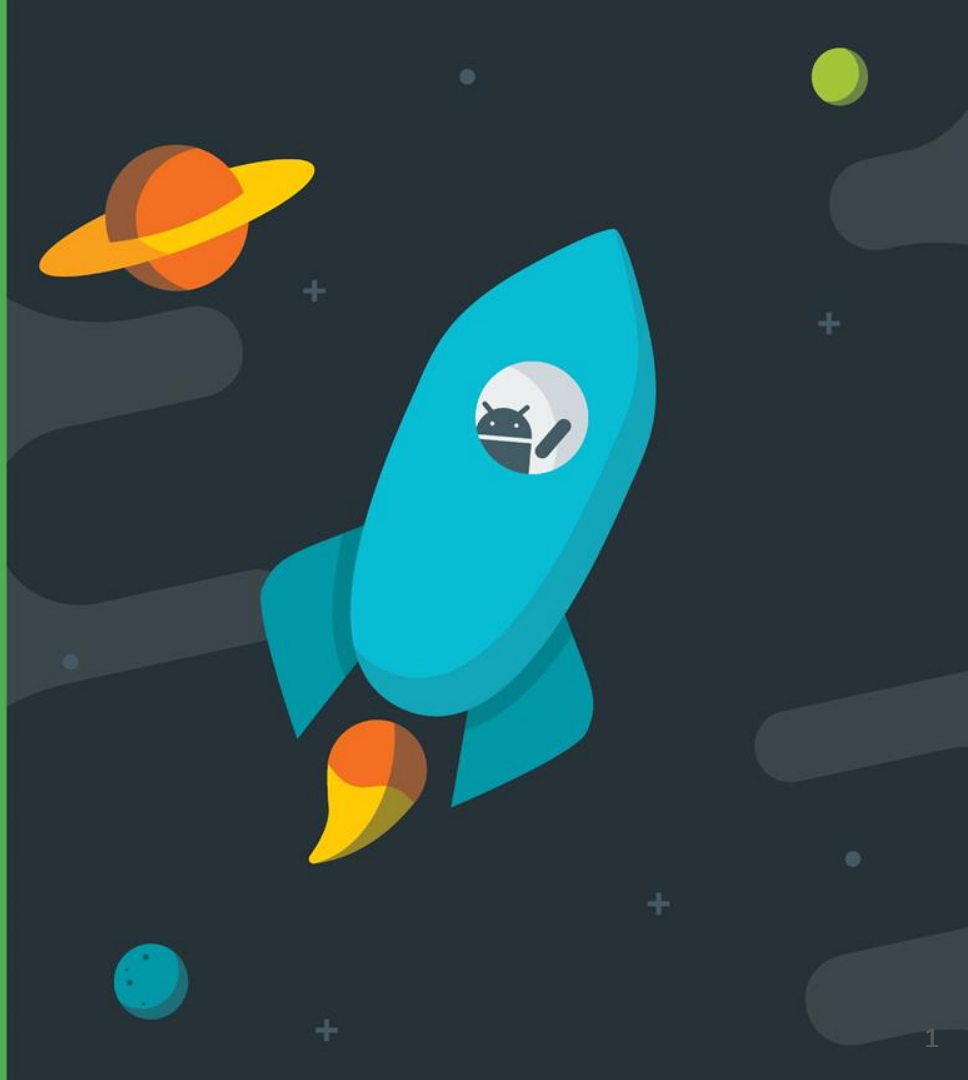
UNITÉ D ’ENSEIGNEMENT (UE) :

DÉVELOPPEMENT MOBILE

CHAPITRE IV:

Stockage



Options de Stockage

- Plusieurs options de stockage pour sauvegarder des données persistantes
- Le choix de la solution idéale dépend des besoins spécifiques:
  - Si les données doivent être privées ou accessibles par d'autres applications
  - Combien d'espace disponible est-il requis?
  - Les données sont-elles structurées, semi-structurées ou pas structurées?

Options de Stockage

Les options de stockage sont les suivantes:

- Shared Preferences
- Stockage Interne
- Stockage Externe
- Bases de Données SQLite
- Le stockage sur le réseau

SHARED PREFERENCES

- Classe qui fournit un framework général qui permet de sauvegarder et extraire des paires clef-valeur persistantes de types primitifs
- Il est possible d'utiliser les **SharedPreferences** pour sauvegarder des Booleans, Floats, Integers, Longs et Strings...
- Les *SharedPreferences* sont typiquement utilisées pour sauvegarder les préférences utilisateur , tel que : quelle sonnerie l'utilisateur a-t-il choisi?

  –Pour cela, utiliser une PreferenceActivity

- Mais pas uniquement...

Utilisation

- Pour lire un objet SharedPreferences, utiliser l'une de ces méthodes:

  -getSharedPreferences (String name, int mode): pour utiliser plusieurs fichiers de préférences identifiés par nom

  -getPreferences (int mode): pour utiliser un seul fichier de préférences, donc sans définir un nom de fichier

- Ses deux méthodes doivent être appelé depuis le Context de l'application : getApplicationContext()
- Le paramètre mode peut prendre plusieurs valeurs : MODE_PRIVATE, MODE_WORLD_READABLE, MODE_WORLD_WRITEABLE, MODE_MULTI_PROCESS. Par défaut, on donne 0 qui correspond à MODE_PRIVATE.

Utilisation

- Ses deux fonctions nous retournent un objet SharedPreferences. Les fonctions de cet objet :

-getAll(), retourne l'ensemble des valeurs de l'objet dans un objet type Map<String, "value">

-getBoolean(String key, boolean defValue), qui retourne le booléen stocké sous le nom donné

-getFloat(String key, float defValue), qui retourne le float stocké sous le nom donné

-getInt(String key, int defValue), qui retourne l'int stocké sous le nom donné

-getLong(String key, long defValue), qui retourne le long stocké sous le nom donné

Utilisation

-getString(String key, String defValue), qui retourne la chaine de caractère stockée sous le nom donné

-getStringSet(String key, Set<String> defValues), retourne un ensemble de chaine de caractère sous un objet de type Set<String>

-registerOnSharedPreferenceChangeListener(SharedPreferences.OnSharedPreferenceChangeListener listener), enregistre un callback appelé dés qu'une modification a lieu au préférence

-unregisterOnSharedPreferenceChangeListener(SharedPreferences.OnSharedPreferenceChangeListener listener), enlève le callback enregistré

-contains(String key), vérifie l'existence d'une clé

Edition

- Pour éditer les préférences, il faut appelé la méthode edit() de l'objet SharedPreferences. Cette fonction nous retourne un objet de type SharedPreferences.Editor. Les fonctions disponibles :
  - apply(), enregistre les valeurs modifiées sans informé si un échec a lieu
  - clear(), vide toutes les valeurs de préférence enregistrées
  - commit(), enregistre les valeurs modifiées mais retourne si la mise à jour est réussite ou non
  - putBoolean(String key, boolean defValue), enregistre un booléen sous le nom donné
  - putFloat(String key, float defValue), enregistre un float sous le nom donné

Edition

- putInt(String key, int defValue), enregistre un int sous le nom donné
  - putLong(String key, long defValue), enregistre un long sous le nom donné
  - putString(String key, String defValue), enregistre une chaine de caractère sous le nom donné
  - putStringSet(String key, Set<String> defValues), enregistre un ensemble de chaine de caractère sous un objet de type Set<String>
  - remove(String key), supprime un ensemble clé/valeur des préférences

Exemple

- Voir complément 1

Stockage Interne

- Il est possible d'utiliser le stockage interne de votre téléphone pour stocker des fichiers
- Par défaut, les fichiers sauvegardés dans le stockage interne sont privés à l'application (inaccessibles à partir d'autres applications)
- Quand l'utilisateur désinstalle l'application, les fichiers sont automatiquement supprimés

Ecriture

- Pour créer et écrire dans un fichier privé en stockage interne

  – Appeler openFileOutput avec le nom du fichier et le mode opératoire (retourne un FileOutputStream)

  – Utiliser write() pour écrire dans le fichier

  – Fermer le flux d'écriture avec close()

Ecriture

- Les modes opératoires sont:
  - MODE_PRIVATE : Le fichier n'est accessible que par l'application qui l'a créé.
  - MODE_WORLD_READABLE : Le fichier est accessible en lecture par les autres applications.
  - MODE_WORLD_WRITEABLE : Le fichier est accessible en écriture par les autres applications.
  - MODE_APPEND : Si le fichier existe déjà, les données seront ajoutées à la fin.

Lecture

- Pour lire le contenu d'un fichier interne:
  - Appeler openFileInput avec le nom du fichier (retourne un FileInputStream)
  - Utiliser read() pour lire des bytes à partir du fichier
  - Fermer le flux de lecture avec close()

Exemple

- Voir complément 2

Stockage externe

- Tous les appareils compatibles Android supportent un espace de stockage externe
  - Peut être une SD card, ou un espace interne non-amovible
- Les fichiers sauvegardés dans un espace de stockage externe sont accessibles à toutes les applications en lecture
- Ils peuvent être modifiés par l'utilisateur si le « USB mass storage » est activé pour transférer les fichiers sur un ordinateur

Utilisation

- Pour lire ou écrire des fichiers sur le stockage externe, l'application doit avoir les permissions (à déclarer dans votre fichier AndroidManifest.xml) READ_EXTERNAL_STORAGE ou WRITE_EXTERNAL_STORAGE

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

- Vérifier la disponibilité du support de stockage grâce à la méthode getExternalStorageState
- Sauvegarder vos fichiers

getExternalStorageState

```
private static boolean isExternalStorageReadOnly() {
   String extStorageState = Environment.getExternalStorageState();
   if (Environment.MEDIA_MOUNTED_READ_ONLY.equals(extStorageState)) {
      return true;
   }
   return false;} ///* Checks if external storage is available for read and write */

private static boolean isExternalStorageAvailable() {
   String extStorageState = Environment.getExternalStorageState();
   if (Environment.MEDIA_MOUNTED.equals(extStorageState)) {
      return true;
   }
   return false; } /* Checks if external storage is available to at least read */
```

Exemple

- Voir complément 3

BASE DE DONNÉES SQLITE

- Android fournit un support total du SGBD SQLite
- SQLite est une bibliothèque logicielle qui implémente un moteur de base de données SQL avec zéro-configuration, léger et sans dépendances externes
- Toutes les bases de données créées dans une application seront accessibles par nom à travers toute cette application, mais pas de l'extérieur
- La méthode recommandée pour la création d'une base de données SQLite d'utiliser une sous-classe de SQLiteOpenHelper

SQLITE

- SQLite est embarquée dans tous les appareils Android. L'utilisateur de SQLite ne requiert aucune action d'installation ou d'administration de la part du développeur.
- La base de données d'une application est stockée dans le répertoire DATA/data/APP_NAME/databases/FILENAME où DATA = Environment.getDataDirectory(), APP_NAME = nom de l'application et FILENAME = le nom donné à la base.
- Le package *android.database* contient toutes les classes nécessaires pour travailler avec les base de données.
- Le package *android.database.sqlite* contient quant à lui toutes les classes pour SQLite.

SQLiteOpenHelper

- Classe d'assistance qui aide l'utilisateur à créer et manipuler sa base de données de manière simple
- Étapes à suivre:
  - Créer une classe qui hérite de SQLiteOpenHelper
  - Créer la base de données et les tables nécessaires
  - Implémenter les méthodes suivantes
    - **Le constructeur**
    - **onCreate** : contient les opérations réalisées à la création de la base de données
    - **onUpgrade** : opérations réalisées quand la base fait un upgrade

SQLiteOpenHelper

- Les deux méthodes reçoivent en paramètre un objet SQLiteDatabase, qui est la représentation Java de la base. Avec la classe SQLiteOpenHelper et ses méthodes getReadableDatabase() et getWriteableDatabase() nous avons accès à un objet SQLiteDatabase.
- SQLiteDatabase est la classe pour travailler dans Android avec une base de données. Elle fournit des méthodes pour ouvrir, interroger, mettre à jour et fermer la base de données. SQLiteDatabase met à notre disposition les fonctions insert(), update() and delete().
- SQLiteDatabase fournit également la méthode execSQL(), qui permet d'exécuter directement du SQL.

Exemple

- Voir complément 4