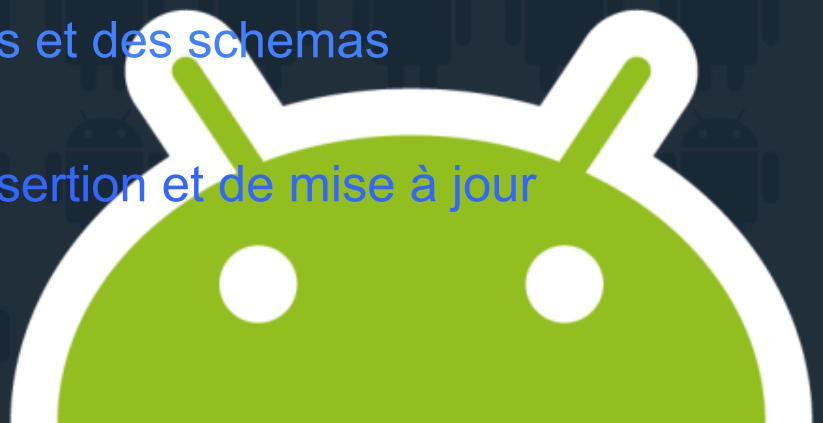
Android
API de Persistance

James Douglass Lefruit
Ingénieur de Recherche, INRIA Grenoble

Android

- Les Préférences
  - SharedPreferences
- Stockage SQLite sur Android
  - Introduction Générale à SQLite
  - Les API SQLite sur Android
    - SQLiteOpenHelper
      - Création de base de données et des schemas
    - Les Curseurs (Cursor)
      - Operations de sélection, d'insertion et de mise à jour



Préférences

- SharedPreferences
  - Mécanisme de stockage de données de configurations d'une application dans un fichier xml
  - Les données peuvent avoir plusieurs portées
    - MODE_PRIVATE, accessible en lecture/écriture seulement aux composants d'une meme application; vivement conseillé par Google en guise de sécurité
    - MODE_WORLD_READABLE, accessible seulement en lecture seule par toute autre application; Google déconseille leur usage n'étant pas trop fiable
    - MODE_WORLD_WRITEABLE, accessible seulement en lecture seule par toute autre application; Google déconseille leur usage n'étant pas trop fiable
- Localisation des fichiers préférences
  - /data/data/nom_package/shared_prefs/
- Exemple de fichier préférence
  - Syntaxe générale : type, clef/valeur

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="app_version" value="1.0" />
</map>
```

SharedPreferences

- Gestionnaire de préference via Context
  - Permet de créer ou charger un fichier préférence pour le lire ou l'editer
  - Ex:

```
SharedPreferences sharedPrefs =
this.getBaseContext().getSharedPreferences ( "config",
Context.MODE_PRIVATE);
//retourne app_version si elle existe ou la valeur par defaut :  unkown;
sharedPrefs.getString("app_version", "unknown");

SharedPreferences.Editor editor = sharedPrefs.edit();
editor.putString("app_version", "2.0");
//faut faire commit pour stocker la valeur
editor.commit();
//maintenant app_version retournera 2.0 car elle est creee
sharedPrefs.getString("app_version", "unknown");
```

SQLite

- Moteur de base de données relationnelles écrit en C
- Permet de manipuler des données avec le langage de requete SQL
- Les données sont stockées dans des fichiers binaires
  - accessibles par l'outil en ligne de commande ***sqlite3***
- Disponible sous Android par l'intermediaire de l'API du framework

API SQLite

- SQLiteOpenHelper
  - Classe abstraite qui encapsule les mécanisme de création/manipulation de base et objets SQLite
  - Crée par défaut la base de donné dans la partition de donnée du telephone
    - /data/data/nom_package/databases/
- Création de base de donnée
  - Il faut étendre la classe SQLiteOpenHelper
  - puis redéfinir ces deux méthodes
    - onCreate (), invoqué si la base n'a pas encore été créée
    - OnUpgrade() , si la version de la base change et mérite une mise à jour

API SQLite

```
public class TestSqliteHelper extends SQLiteOpenHelper {
private final static String TAG="TestSQliteHelper";
public static final String DATABASE_NAME="employe.db";
public static final String EMP_TABLE="contact";
 //texte en langage sql de creation d'une table
private static final String
 EMP_SCHEMA="CREATE TABLE employe (_id integer primary key autoincrement,"+
                                                    "nom VARCHAR(255) NULL DEFAULT NULL, " +
                                                    "prenom VARCHAR(255) NULL DEFAULT NULL, " +
                                                    "poste VARCHAR(255) NULL DEFAULT NULL, " +
                                                    ");";
//Par convention, un champ id est plutot nommé _id pour Android
public static final String _ID="_id";      private static final int DB_VERSION=1;
public TestSqliteHelper(Context context){
            super(context, DATABASE_NAME, null, DB_VERSION);
}
@Override
  public void onCreate(SQLiteDatabase db) {
            Log.v(TAG, "Creating INRIA database table");
            db.execSQL(EMP_SCHEMA);
   }
 @Override
  public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
            Log.v(TAG, "Upgrading database");
   }
}
```

API SQLite

- **Création de la base**
  - Instancier la classe fille de SQLiteOpenHelper en lui passant le contexte de l'application
  - Invoquer la méthode getWritableDatabase() de la classe fille
    - A cet instant, la méthode onCreate est exécutée pour créer la base

API SQLite

```
//Exemple de mise en œuvre de la base
public class TestSqlite extends Activity {
private SQLiteOpenHelper dbHelper;
private SQLiteDatabase database;

 public void onCreate(Bundle savedInstance){
                super.onCreate(savedInstance);
                setContentView(R.layout.main);
                this.dbHelper = new TestSqliteHelper(this.getBaseContext());
                //maintenant la base est crée avec l'appel de la methode suivante
                this.database = this.dbHelper.getWritableDatabase();
                //faut toujours fermer la base
                this.database.close();
  }
}
```

API SQLite

- **Requete SQL et Curseur(Cursor)**
  - Les requêtes SQL se font via l'objet Database retournée par getWritableDatabase() ou getReadableDatabase()
  - Les requêtes utilisent les clauses SQL sous forme d'argument fourni à la méthode query() de Database
    - Retourner la liste de tous les employes
      - en SQL brut : select * from employe;
      - API SQLite
        - » Database database=sqliteHelper.getWritableDatabase();
        - » Cursor cursor=database.query("employe", null, null, null, null, null, null)
  - L'objet Cursor permet de parcourir les enregistrements retournés par la requête en utilisant la methode moveToNext() dans une boucle
  - moveToNext() retourne true ou false pour indiquer la présence ou l'absence de données
  - L'API SQLite permet de lire la donnée d'une colonne via son numéro d'indice qui peut être retourné avec la méthode getColumIndex du curseur
    - cursor.getString(cursor.getColumnIndex("nom"));

API SQLite

- **Requêtes avec paramètre**
  - Permet de filtrer sa requête avec des valeurs contenu dans des variables
  - Les paramètres sont indiqués par des points d'interrogation ?
  - Retourner le prénom et nom des employés dont le poste est gestion
    - En SQL: select nom,prenom from employe where poste='gestion';
    - API SQLite:
      - database.query("employe",
        new String[]{"nom","prenom"},
        " poste = ? ", new String[]{"gestion"},
        null,
        null,
        null)

API SQLite

- **Insertion de données**
  - L'insertion de données se fait via la méthode insert()
  - Les données à inserer sont fournis via un objet de type ContentValues contenant le couple nom colonne et valeur
  - Inserer un nouvel employe
    - En SQL : insert into employe(nom,prenom,poste) values ('Claude','Bernard','informatique')
    - API SQLite
      - ContentValues data = New ContentValues();
      - data.put("nom", "Claude");
      - data.put("prenom", "Bernard");
      - data.put("poste", "Informatique");
      - this.database.insert("employe", null, data);