# 09. L□u tr□ d□ li□u

## Các gi□i pháp l□u tr□ d□ li□u

- Shared Preferences
- B□ nh□ trong
- B□ nh□ ngoài
- SQLite
- Network

## Các tiêu chí l□a ch□n gi□i pháp l□u tr□ d□ li□u

- Kh□ năng truy c□p d□ li□u t□:
  - Chính □ng d□ng đó
  - □ng d□ng khác/Ng□□i dùng
- Khi □ng d□ng:
  - Đã g□ cài đ□t ho□c ch□a
  - Đã k□t n□i m□ng ho□c ch□a
- Dung l□□ng l□u tr□

# Shared Preferences

## Shared Preferences

- L□u tr□ng thái ng□□i dùng: Tr□ng thái đăng nh□p (đã đăng nh□p hay ch□a) đ□ ng□□i dùng không ph□i đăng nh□p l□i m□i khi m□ □ng d□ng.
- L□u cài đ□t: Cài đ□t ngôn ng□ □u tiên c□a ng□□i dùng trong □ng d□ng.
- Ch□a d□ li□u private nguyên th□y theo d□ng key-value (booleans, floats, ints, longs, and strings).
- Shared preferences không ch□ đ□ l□u tr□ thi□t l□p c□a ng□□i dùng.

- PreferenceActivity đ□□c s□ d□ng đ□ t□o thi□t l□p ng□□i dùng.

- Shared preferences đ□□c l□u tr□ trong t□p XML trên thi□t b□ có đ□□ng d□n nh□ sau: `/data/data/<application's package name>/shared_prefs`
- S□ b□ xóa b□ khi g□ cài đ□t □ng d□ng.
- Shared preferences có th□ đ□□c liên k□t v□i □ng d□ng ho□c m□t Activity c□ th□.
- S□ d□ng m□t trong hai cách sau đ□ l□y đ□i t□□ng SharedPreferences
  - getSharedPreferences(String name, int mode) - S□ d□ng khi c□n nhi□u t□p preferences đ□□c xác đ□nh b□i tên.
  - getPreferences() - S□ d□ng khi ch□ c□n m□t t□p preferences cho hành đ□ng.
    - C□ ch□ bên trong: s□ d□ng tên l□p c□a hành đ□ng nh□ tên c□a preferences

## Shared Preferences - Đ□c

- Đ□c preferences s□ d□ng các ph□□ng th□c sau:
  - `contains(String key)` - Ki□m tra xem preferences có ch□a m□t preference c□ th□ không.
  - `getAll()` - L□y toàn b□ values t□ preferences.
  - `getBoolean(String key, boolean defValue)` - L□y m□t giá tr□ ki□u boolean t□ preferences.
  - `getFloat(String key, float defValue)` - L□y m□t giá tr□ ki□u float t□ preferences.
  - `getInt(String key, int defValue)` - L□y m□t giá tr□ ki□u int t□ preferences.
  - `getLong(String key, long defValue)` - L□y m□t giá tr□ ki□u long t□ preferences.
  - `getString(String key, String defValue)` - L□y m□t giá tr□ ki□u String t□ preferences.
  - `getStringSet(String key, Set<String> defValues)` - L□y m□t t□p giá tr□ ki□u String t□ preferences.

## Shared Preferences - Ghi

- Các b□□c ghi giá tr□:
  1. G□i hàm `edit()` đ□ nh□n SharedPreferences.Editor.
  2. Thêm giá tr□ b□ng các hàm `putBoolean()` và `putString()`.
  3. Ghi nh□n giá tr□ m□i v□i hàm `commit()`.

```
// We need an Editor object to make preference changes.
// All objects are from android.context.Context
SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
SharedPreferences.Editor editor = settings.edit();
editor.putBoolean("silentMode", mSilentMode);

// Commit the edits!
editor.commit();
```

# Android File System

## Android File System

- L□u d□ li□u t□m th□i: Các t□p hình □nh t□m th□i đ□ hi□n th□ trong □ng d□ng tr□□c khi t□i lên máy ch□.
- L□u tr□ d□ li□u: Các t□p JSON ho□c CSV ch□a d□ li□u □ng d□ng cho vi□c x□ lý nhanh và d□ dàng.
- Android s□ d□ng Linux kernel tuân th□ VFS (Virtual FileSystem) - m□t l□p tr□u t□□ng phía trên c□a các t□p tin h□ th□ng c□ th□.

## Android File System - C□u trúc th□ m□c thông th□□ng

- `etc`: M□t liên k□t t□□ng tr□ng t□i `/system/etc`.
- `proc`: Mount point cho t□p tin h□ th□ng procfs, cho phép truy c□p t□i c□u trúc d□ li□u kernel.
- `root`: Th□ m□c g□c cho tài kho□n root.
- `sdcard`: M□t liên k□t t□□ng tr□ng t□i m□t th□ m□c trong `/storage/*`.
- `sys`: Mount point cho t□p tin h□ th□ng sysfs pseudo, là s□ ánh x□ c□a c□u trúc đ□i t□□ng thi□t b□ c□a kernel.
- `system`: Mount point cho `/dev/block/mtdblock0`. Các th□ m□c trong đây th□□ng đ□□c nhìn th□y trong th□ m□c root c□a các b□n phân ph□i tiêu chu□n Linux bao g□m các th□ m□c bin, etc, lib, usr, và xbin.
- `vendor`: M□t liên k□t t□□ng tr□ng t□i `/system/vendor` ch□a các đo□n mã thu□c v□ nhà s□n xu□t.

# B nh trong

## B nh trong (Internal Storage)

- Lu tr c s d liu SQLite:
  - Lu tr thông tin ngi dùng, danh sách sn phm, lch s truy cp, vv.
- Lu tr tp nh:
  - Lu tr các tp nh đc to ra bi ngi dùng trong ng dng.

## Đc

- Gi openFileInput() và truyn vào tên ca tp tin cn đc. Hàm s tr li mt FileInputStream.
- Đc bytes t tp tin vi hàm read()
- Đóng lung vi hàm close()

```
FileInputStream fis = context.openFileInput("hello.txt");
InputStreamReader isr = new InputStreamReader(fis);
BufferedReader bufferedReader = new BufferedReader(isr);
StringBuilder sb = new StringBuilder();
String line;
while ((line = bufferedReader.readLine()) != null)
{
  sb.append(line);
}
fis.close();
```

## Ghi

```
String filename = "myfile";
String fileContents = "Hello world!";
FileOutputStream outputStream;

try {
  outputStream = openFileOutput(filename, Context.MODE_PRIVATE);
  outputStream.write(fileContents.getBytes());
  outputStream.close();
} catch (Exception e) {
```

```
  e.printStackTrace();
}
```

## T□p tin Cache

- S□ d□ng ph□□ng th□c getCacheDir() đ□ m□ m□t File đ□i di□n cho th□ m□c l□u tr□ t□m th□i
- Các t□p cache s□ t□ đ□ng đ□□c xóa b□i h□ th□ng n□u c□n ch□ tr□ng trên đĩa
  - H□ th□ng s□ luôn luôn xóa các t□p cũ tr□□c b□ng cách s□ d□ng ph□□ng th□c lastModified()
  - M□t cách có ki□m soát h□n là s□ d□ng *setCacheBehaviorGroup*(File, boolean) và *setCacheBehaviorTombstone*(File, boolean)
- Khuy□n khích gi□ dung l□□ng cache s□ d□ng d□□i đ□nh m□c quy đ□nh b□i *getCacheQuotaBytes(java.util.UUID)*. Đ□nh m□c thay đ□i theo th□i gian ph□ thu□c vào:
  - T□n su□t ng□□i dùng t□□ng tác v□i □ng d□ng
  - Dung l□□ng đĩa system-wide disk s□ d□ng

## Các ph□□ng th□c khác

- *getFilesDir*() - L□y đ□□ng d□n tuy□t đ□i t□i th□ m□c filesystem, n□i mà các t□p internal đ□□c l□u tr□.
- *getDir*() - T□o (ho□c m□ n□u đã t□n t□i) th□ m□c bên trong b□ nh□ trong
- *deleteFile*() - Xóa t□p tin trong b□ nh□ trong
- *fileList*() - Tr□ v□ m□ng t□p tin hi□n t□i đ□□c l□u tr□ b□i □ng d□ng.

# B□ nh□ ngoài

## B□ nh□ ngoài (External Storage)

- L□u tr□ d□ li□u c□a □ng d□ng trên b□ nh□ ngoài, th□□ng là SD card ho□c phân vùng khác c□a thi□t b□.
- C□n quy□n truy c□p b□ nh□ ngoài.
- N□u □ng d□ng b□ g□ cài đ□t, d□ li□u trên b□ nh□ ngoài không b□ xóa.
- Đ□c/ghi d□ li□u cũng nh□ b□ nh□ trong.

- Yêu c□u quy□n h□ th□ng READ_EXTERNAL_STORAGE ho□c WRITE_EXTERNAL_STORAGE đ□ đ□c và ghi các t□p tin.
  - Quy□n đ□c đ□□c ng□m yêu c□u b□i quy□n ghi.

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

## B□ nh□ ngoài (External Storage) - Ki□m tra tính s□n có

- Tr□□c khi vi□t vào b□ nh□ ngoài, hãy ki□m tra xem nó có s□n không.

```
/* Checks if external storage is available for read and write */
public boolean isExternalStorageWritable() {
  String state = Environment.getExternalStorageState();
  return Environment.MEDIA_MOUNTED.equals(state));
}

/* Checks if external storage is available to at least read */
public boolean isExternalStorageReadable() {
  String state = Environment.getExternalStorageState();
  return Environment.MEDIA_MOUNTED.equals(state) ||
Environment.MEDIA_MOUNTED_READ_ONLY.equals(state);
}
```

- G□i getExternalStoragePublicDirectory(String type) đ□ nh□n th□ m□c l□u tr□ các t□p tin công khai
- *Tham s□ type* thu□c vào các giá tr□ sau:  DIRECTORY_MUSIC, DIRECTORY_PODCASTS, DIRECTORY_RINGTONES, DIRECTORY_ALARMS, DIRECTORY_NOTIFICATIONS, DIRECTORY_PICTURES, DIRECTORY_MOVIES, DIRECTORY_DOWNLOADS, DIRECTORY_DCIM, ho□c DIRECTORY_DOCUMENTS

- Đ□□ng d□n tr□ v□ b□i ph□□ng th□c này có th□ không t□n t□i, nên s□ d□ng path.mkdirs() đ□ đ□m b□o đi□u này
- Path = “<external storage path>/” + DIRECTORY_*

  ○DIRECTORY_MUSIC  = "Music";

  ○DIRECTORY_PODCASTS  = "Podcasts";

  ○DIRECTORY_RINGTONES  = "Ringtones";

  ○DIRECTORY_ALARMS  = "Alarms";

  ○DIRECTORY_NOTIFICATIONS  = "Notifications";

  ○DIRECTORY_PICTURES  = "Pictures";

  ○DIRECTORY_MOVIES  = "Movies";

  ○DIRECTORY_DOWNLOADS  = "Download";

  ○DIRECTORY_DCIM  = "DCIM";

```
public File getAlbumStorageDir(String albumName) {
    // Get the directory for the user's public pictures directory.
    File file = new File(Environment.getExternalStoragePublicDirectory(
            Environment.DIRECTORY_PICTURES), albumName);
    if (!file.mkdirs()) {
        Log.e(LOG_TAG, "Directory not created");
    }
    return file;
}
```

## t□p tin private

- V□n có th□ truy c□p b□i ng□□i dùng và □ng d□ng khác, nh□ng không cung c□p giá tr□ cho ng□□i dùng bên ngoài □ng d□ng.
- Khi ng□□i dùng g□ b□ cài đ□t, h□ th□ng xóa toàn b□ t□p tin trong th□ m□c external private
- G□i ph□□ng th□c getExternalFilesDir(String type) đ□ nh□n th□ m□c l□u tr□ các t□p tin private.

```
void deleteExternalStoragePrivateFile() {
    // Get path for the file on external storage.  If external
    // storage is not currently mounted this will fail.
    File file = new File(getExternalFilesDir(null), "DemoFile.jpg");
    if (file != null) {
        file.delete();
```

```
    }
}
```

# SQLite

## SQLite

- H□ th□ng qu□n lý c□ s□ d□ li□u quan h□ nh□ (RDBMS) đ□□c tích h□p s□n trong Android.
- S□ d□ng ngôn ng□ SQL đ□ truy v□n, c□p nh□t c□ s□ d□ li□u.
- L□u tr□ d□ li□u trong các b□ng và quan h□ gi□a các b□ng.
- Thích h□p cho các □ng d□ng yêu c□u tìm ki□m, s□p x□p và truy v□n d□ li□u ph□c t□p.

### Đ□nh nghĩa l□□c đ□ và Contract

- L□p Contract ch□ đ□nh rõ khung c□a l□□c đ□ d□ li□u theo m□t cách có h□ th□ng và t□ tài li□u hóa.
  - ○Ch□a các h□ng đ□nh nghĩa tên cho URIs, b□ng, và c□t.
  - ○Cho phép s□ d□ng chung h□ng trong toàn b□ các l□p trong cùng package.
- Cách t□ ch□c l□p Contract đ□□c khuy□n khích:
  - ○Đ□t các đ□nh nghĩa có tính t□ng quát đ□i v□i c□ s□ d□ li□u vào m□c root c□a l□p.
  - ○Sau đó t□o inner class cho m□i b□ng li□t kê các c□t c□a nó

```
public final class FeedReaderContract {
    // To prevent someone from accidentally instantiating the contract class,
    // make the constructor private.
    private FeedReaderContract() {}

    /* Inner class that defines the table contents */
    public static class FeedEntry implements BaseColumns {
        public static final String TABLE_NAME = "entry";
        public static final String COLUMN_NAME_TITLE = "title";
        public static final String COLUMN_NAME_SUBTITLE = "subtitle";
    }
}
```

## Tạo cơ sở dữ liệu

```
public class FeedReaderDbHelper extends SQLiteOpenHelper {
    private static final String SQL_CREATE_ENTRIES =
        "CREATE TABLE " + FeedEntry.TABLE_NAME + " (" +
        FeedEntry._ID + " INTEGER PRIMARY KEY," +
        FeedEntry.COLUMN_NAME_TITLE + " TEXT," +
        FeedEntry.COLUMN_NAME_SUBTITLE + " TEXT)";

    // If you change the database schema, you must increment the database version.
    // ...
    // ...
    public FeedReaderDbHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(SQL_CREATE_ENTRIES);
    }
}
```

## đọc từ csdl

```
SQLiteDatabase db = mDbHelper.getReadableDatabase();

// Define a projection that specifies which columns from the database
// you will actually use after this query.
String[] projection = {FeedEntry._ID, FeedEntry.COLUMN_NAME_TITLE, FeedEntry.COLUMN_NAME_SUBTITLE};
// Filter results WHERE "title" = 'My Title'
String selection = FeedEntry.COLUMN_NAME_TITLE + " = ?";
String[] selectionArgs = { "My Title" };
// How you want the results sorted in the resulting Cursor
String sortOrder = FeedEntry.COLUMN_NAME_SUBTITLE + " DESC";

Cursor cursor = db.query(
    FeedEntry.TABLE_NAME,                     // The table to query
    projection,                               // The columns to return
    selection,                                // The columns for the WHERE clause
    selectionArgs,                            // The values for the WHERE clause
    null,                                     // don't group the rows
    null,                                     // don't filter by row groups
    sortOrder                                 // The sort order
    );
```

## Ghi vào csdl

```
// Gets the data repository in write mode
SQLiteDatabase db = mDbHelper.getWritableDatabase();

// Create a new map of values, where column names are the keys
```

```
ContentValues values = new ContentValues();
values.put(FeedEntry.COLUMN_NAME_TITLE, title);
values.put(FeedEntry.COLUMN_NAME_SUBTITLE, subtitle);

// Insert the new row, returning the primary key value of the new row
long newRowId = db.insert(FeedEntry.TABLE_NAME, null, values);
```

xoá từ csdl

```
// Define 'where' part of query.
String selection = FeedEntry.COLUMN_NAME_TITLE + " LIKE ?";
// Specify arguments in placeholder order.
String[] selectionArgs = { "MyTitle" };
// Issue SQL statement.
db.delete(FeedEntry.TABLE_NAME, selection, selectionArgs);
```

# Network

- Lưu trữ dữ liệu từ máy chủ từ xa:

  Sử dụng API của Firebase để lưu trữ và truy xuất dữ liệu từ máy chủ Firebase
- Lưu trữ dữ liệu realtime từ Internet:

  Sử dụng WebSocket để truyền và nhận dữ liệu realtime từ một máy chủ từ xa
- Lưu dữ liệu trên máy tính khác/đám mây, truy cập qua mạng
- Để thực hiện các thao tác liên quan tới mạng, sử dụng các gói sau:

  java.net.*

  android.net.*