



# LẬP TRÌNH DI ĐỘNG

---

Bài 6: các phương pháp lưu trữ trong android và làm việc với SQLite

# Nhắc lại bài trước

---

- Khái niệm intent
- Giao tiếp giữa 2 activity sử dụng Intent
- Intent, intent service & intent filter
- Intent tường minh & intent ngầm định
- Các thành phần của intent: component, action, category, data, type, extras
- Hai kiểu gọi activity:
  - **startActivity**: thực hiện, không cần kết quả trả về
  - **startActivityForResult**: muốn nhận kết quả trả về

# Nhắc lại bài trước

---

- Intent là cơ chế chuẩn của android để giao tiếp giữa các ứng dụng
  - Có thể giao tiếp theo nhiều cách khác, nhưng dùng intent là cách mà tất cả các ứng dụng đều áp dụng
- Gọi thực hiện một **nhiệm vụ** cụ thể, dùng được với activity của ứng dụng bất kỳ:

```
startActivity(new Intent(Intent.ACTION_DIAL,  
    Uri.parse("tel:0912102165")));
```

- Gọi thực hiện một **activity** cụ thể:

```
startActivity(new Intent(this,  
    Activity2.class));
```

# Nhắc lại bài trước

---

- A chuẩn bị dữ liệu và gọi B:

```
intent = new Intent(...);  
intent.putExtra(key, value);  
...  
startActivityForResult(intent, CODE-OF-B);
```

- B khởi chạy và lấy dữ liệu do A gửi:

```
intent = getIntent();  
V = intent.getStringExtra(key);  
...
```

- B trả về kết quả:

```
intent = new Intent();  
intent.putExtra(key, value);  
...  
setResult(RESULT_OK, intent);
```

# Nhắc lại bài trước

---

- A bắt kết quả trả về từ B trong `onActivityResult`:

```
protected void onActivityResult(int code, int result,
                                Intent data) {
    if (code == CODE-OF-B) {
        // xử lý trường hợp B trả về kết quả thành công
        if (result == RESULT_OK) {
            ...
        }
        // xử lý các kết quả khác của B
        ...
    }
    // xử lý các CODE do các activity khác trả về
    ...
    // gọi xử lý của activity cha
    super.onActivityResult(code, result, data);
}
```

# Nội dung

---

1. Tổng quan về lưu trữ trong android
2. Shared Preferences
3. Files
  1. File trên internal storage
  2. File tạm
  3. File trên external storage
  4. File nội bộ (trong file apk)
4. Làm quen với SQLite

Phần 1

# Tổng quan về lưu trữ trong android

# Tổng quan: các loại lưu trữ

---

- Android có nhiều phương pháp lưu trữ dữ liệu
  - Mỗi phương pháp có mục đích sử dụng khác nhau (vì vậy cần hiểu chính xác để sử dụng hợp lý nhất)
  - Cơ chế phân quyền và kiểm soát truy cập kiểu Linux
- Local storages:
  - Raw: **File services** (memory, cached, internal card, sdcard,...)
  - OS services: **Shared preferences, SQLite, Content providers**
  - Extra services: **Content providers**
- Remote storages: **Internet services**



# Tổng quan: quá trình cài ứng dụng

---

- Ứng dụng android ở dạng **.apk**
- Từ API 8, có thể đặt ứng dụng ở sdcard: thêm đoạn mã **android:installLocation="preferExternal"** vào file **AndroidManifest.xml**
- Quá trình ứng dụng được cài đặt vào hệ thống:
  - Kiểm tra sự toàn vẹn của file .apk dựa trên chữ kí số
  - Chép file .apk vào thư mục ứng dụng
  - Tạo thư mục riêng cho ứng dụng đó (theo tên package)
  - Thiết lập quyền phù hợp cho thư mục riêng
  - Cập nhật CSDL về các thành phần của ứng dụng

# Tổng quan: một số folder cơ bản

---

- Theo thiết lập chuẩn của Android OS:
  - Ứng dụng hệ thống: `/system/app`
  - Ứng dụng thường: `/data/app`
  - Ứng dụng ở sdcard: `/storage/sdcard0/.android_secure`
  - Dữ liệu của ứng dụng: `/data/data/<package_name>`
    - Folder “shared\_prefs”: chứa share preferences
    - Folder “cache”: chứa các file tạm
    - Folder “databases”: chứa các CSDL SQLite
  - Dữ liệu ở sdcard: `/Android/data/<package_name>/files/`
- Cần lấy các folder bằng API của hệ thống

Phần 2

# Shared Preferences

# Shared Preferences

---

- Shared Preferences cho phép lưu trữ dữ liệu theo cặp key/value với các kiểu dữ liệu cơ bản
- File lưu ở dạng XML, có thể chia sẻ với ứng dụng khác (mục tiêu cũng là để chia sẻ)
- Các kiểu dữ liệu hỗ trợ: String, float, int, long và boolean
- Cách làm việc:
  1. Lấy đối tượng SharedPreferences: dùng phương thức `getSharedPreferences(string, int)` hoặc `getPreferences(int mode)`
  2. Sử dụng các phương thức của class `SharedPreferences` để thao tác với dữ liệu bên trong

# getSharedPreferences

---

- Phương thức “public abstract SharedPreferences **getSharedPreferences**(String xml, int mode)”
  - Đây là phương thức của context
  - Phương thức lấy về đối tượng SharedPreferences để đọc ghi dữ liệu lên file xml với tên được chỉ định bằng tham số truyền vào
  - File XML phải nằm trong folder **shared\_prefs** của data
  - Tham số mode dùng để thiết lập quyền truy xuất đến file xml mà đối tượng SharedPreferences tham chiếu đến

# getSharedPreferences

---

- Có ba loại mode:
  - **MODE\_PRIVATE**: chỉ có thể được truy xuất bên trong ứng dụng tạo ra nó
  - **MODE\_WORLD\_READABLE**: có thể được đọc bởi các ứng dụng khác
  - **MODE\_WORLD\_WRITEABLE**: có thể được ghi bởi các ứng dụng khác
- Chú ý:
  - Có quyền root vẫn đọc được dữ liệu dù nó thiết lập chế độ MODE\_PRIVATE
  - Có quyền ghi thì đương nhiên có quyền đọc

# getPreferences (int mode)

---

- Phương thức “public SharedPreferences **getPreferences**(int mode)”
  - Phương thức của activity
  - Phương thức này sẽ gọi lại getSharedPreferences(...) với tham số String xml là tên của activity hiện tại
  - Tham số int mode trong phương thức tương tự như tham số mode slide trước
- Nhận xét: phương thức này giúp lập trình viên tạo SharedPreferences ứng với từng activity mà không cần quá quan tâm tới package name

# Ghi dữ liệu

---

- Gọi phương thức `SharedPreferences.edit()` để lấy về đối tượng `SharedPreferences.Editor` đối tượng này sử dụng để ghi dữ liệu xuống file xml
- Thêm dữ liệu vào file xml bằng cách gọi các phương thức putXXX:
  - `SharedPreferences.Editor.putBoolean()`
  - `SharedPreferences.Editor.putString()`
  - ...
- Gọi phương thức `SharedPreferences.commit()` để hoàn tất việc thay đổi nội dung và ghi dữ liệu



# Ví dụ

---

```
// Đọc dữ liệu
SharedPreferences settings =
getSharedPreferences(PREFS_NAME, 0);
    boolean silent = settings.getBoolean("silentMode",
false);
    setSilent(silent);
}

public static final String MYPREFS = "mySharedPreferences";
protected void savePreferences(){
// tạo đối tượng lưu trữ và truy xuất dữ liệu shared preference
int mode = Activity.MODE_PRIVATE;
SharedPreferences mySharedPreferences = getSharedPreferences
(MYPREFS,mode);
// nhận về một editor để sửa đổi shared preferences.
SharedPreferences.Editor editor = mySharedPreferences.edit();
// lưu trữ vào . shared preference
editor.putBoolean("isTrue", true);
editor.putFloat("lastFloat", 1f);
editor.putInt("wholeNumber", 2);
editor.putLong("aNumber", 31);
editor.putString("textEntryValue", "Not Empty");
// Commit the changes.
editor.commit();
}
```

# Preferences Activity

---

- Ứng dụng phổ biến nhất của Preference là dùng để tạo một trang settings
- Tham khảo bài đọc về PreferencesActivity (tài liệu)
- Ở một ứng dụng A, muốn mở shared preferences của ứng dụng khác (nếu được share), thực hiện như sau:

```
other = createPackageContext(package_name,  
Context.MODE_WORLD_WRITEABLE);  
share = other.getSharedPreferences(xml_name, 0);
```

Phần 3

# Files

# Files

---

- Android cung cấp khá nhiều cách để đọc và lưu trữ dữ liệu từ/xuống các tập tin
  - Dựa trên các API về file của Java
  - Dựa trên một số dạng đặc biệt chỉ có trong android (các tập tin tài nguyên chẳng hạn)
- Một số dạng tập tin phổ biến
  - File trên bộ nhớ trong (internal storage)
  - File đệm (cached)
  - File trên bộ nhớ ngoài (external storage)
  - File tài nguyên (resources, nằm trong APK)

Phần 3.1

# File trên bộ nhớ trong

# File trên bộ nhớ trong

---

Mặc định thì tập tin này sẽ thuộc về ứng dụng tạo ra nó và các ứng dụng khác không thể truy xuất đến nó

01. Gọi phương thức `FileOutputStream openFileOutput (String name, int mode)` để tạo một đối tượng `FileOutputStream` dùng để ghi dữ liệu lên tập tin.

Phương thức này nhận vào hai tham số:

- Tên tập tin.
- Chế độ ghi. Có ba chế độ:
  - `MODE_PRIVATE`: Tập chỉ có thể được truy xuất bên trong ứng dụng tạo ra nó.
  - `MODE_WORLD_READABLE`: Tập có thể được đọc bởi các ứng dụng khác.
  - `MODE_WORLD_WRITEABLE`: Tập có thể được đọc và ghi bởi các ứng dụng khác.

02. Gọi phương thức `FileOutputStream.write(...)` để ghi dữ liệu xuống file.

03. Gọi phương thức `FileOutputStream.close()` để đóng luồng ghi dữ liệu xuống file.

# Đọc dữ liệu từ tập tin

---

- Để đọc dữ liệu từ tập tin ta thực hiện các bước:
  - Gọi phương thức “`public abstract FileInputStream openFileInput(String name)`” tạo luồng đọc dữ liệu từ file.
    - Phương thức này nhận vào một tham số là tên file cần đọc
  - Gọi phương thức `FileInputStream.read(...)` để đọc dữ liệu từ file
  - Gọi phương thức `FileInputStream.close()` để đóng luồng đọc dữ liệu từ file
- Các phương thức làm việc đều tương tự như cách làm việc tiêu chuẩn với file của java

# Ví dụ: đọc dữ liệu từ file

---

```
String FILENAME = "hello_file";  
String string = "";
```

```
FileInputStream fis=null;  
try {
```

```
    fis = openFileInput(FILENAME);  
    byte[] buffer = new byte[1024];  
    int count = fis.read(buffer);  
    string = new String(buffer);
```

```
} catch (IOException e) {  
    e.printStackTrace();  
}
```

```
finally{  
    try {  
        fis.close();  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}
```

Trả về số byte  
thực đọc



Phần 3.2

# File tạm

# Sử dụng tập tin cache

---

- Sử dụng khi muốn lưu trữ tập tin vào thư mục cache thay vì lưu trữ vĩnh viễn
- Các tập tin này sẽ tự động bị xóa khi thiết bị thiếu bộ nhớ trong
- Sử dụng phương thức `getCacheDir()` để lấy về thư mục cache lưu trữ dữ liệu của ứng dụng (thường là `data/data/<package_name>/cache`)

# Ví dụ: ghi dữ liệu lên tập tin cache

---

```
File file = getCacheDir();
String cacheDir = file.getPath();
File downloadingMediaFile = new
File(cacheDir, "downloadingMedia.dat");
FileOutputStream out = null;
try {
    out = new FileOutputStream(downloadingMediaFile);
    out.write(str.getBytes());
} catch (Exception e) {
    e.printStackTrace();
}
finally{
    try {
        out.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

# Các thư mục chuẩn

---

Android SDK định nghĩa một số thư mục chuẩn thành hằng số trong class [android.os.Environment](#)

- DIRECTORY\_ALARMS
- DIRECTORY\_DCIM (picture + video ở chế độ device as camera)
- DIRECTORY\_DOCUMENTS
- DIRECTORY\_DOWNLOADS
- DIRECTORY\_MOVIES
- DIRECTORY\_MUSIC
- DIRECTORY\_NOTIFICATIONS
- DIRECTORY\_PICTURES
- DIRECTORY\_PODCASTS
- DIRECTORY\_RINGTONES

# Ví dụ: một số hàm hữu ích

---

```
public File getTempFile(Context context, String url) {
    File file;
    try {
        String fileName = Uri.parse(url).getLastPathSegment();
        file = File.createTempFile(fileName, null, context.getCacheDir());
    } catch (IOException e) {
        // Error while creating file
    }
    return file;
}

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

Phần 3.3

# File trên bộ nhớ ngoài

# Sử dụng bộ nhớ ngoài

---

Bộ nhớ ngoài (external memory) là thiết bị lưu trữ có thể tháo rời (thường là SDCARD) nên cần tiến hành kiểm tra trạng thái trước khi đọc và ghi dữ liệu

```
boolean mExternalStorageAvailable = false;
boolean mExternalStorageWriteable = false;
String state = Environment.getExternalStorageState();

if (Environment.MEDIA_MOUNTED.equals(state)) {
    // Có thể đọc và ghi dữ liệu
    mExternalStorageAvailable = mExternalStorageWriteable =
true;
} else if (Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
    // Chỉ có thể đọc
    mExternalStorageAvailable = true;
    mExternalStorageWriteable = false;
} else {
    mExternalStorageAvailable = mExternalStorageWriteable =
false;
}
```

# Sử dụng bộ nhớ ngoài

---

- Nếu muốn ghi trên SDCARD, cần cấp quyền `android.permission.WRITE_EXTERNAL_STORAGE`
- Truy cập file ở bộ lưu trữ ngoài
  - API Level  $\geq 8$ , sử dụng `getExternalFilesDir()` lấy về đối tượng file chứa đường dẫn tới thư mục gốc bộ nhớ ngoài
  - API Level  $\leq 7$ , sử dụng `getExternalStorageDirectory()`
- Thông thường dữ liệu được lưu trữ theo đường dẫn `/Android/data/<package_name>/files/`
- Dữ liệu trên sdcard có thể không được bảo vệ



Phần 3.4

# File nội bộ (trong file apk)

# Truy xuất các files trong Resources

---

- Nếu ứng dụng đòi hỏi nguồn tài nguyên từ tập tin bên ngoài, có thể gộp chúng vào thư mục res/raw trong dự án
- Sử dụng phương thức `openRawResource` lấy về luồng `InputStream`
- Không thể ghi vào resource

```
Resources myResources = getResources();  
InputStream myFile = myResources.  
    openRawResource(R.raw.<tên file>);
```

# Ví dụ: nạp font từ asset

---

```
fontPath = "fonts/batman.ttf";  
tf = Typeface.createFromAsset(getAssets(), fontPath);  
txtText.setTypeface(tf);
```

# Ví dụ: cài apk từ asset

---

```
String rarPath = "rar/sms.apk";
AssetManager assetManager = getAssets();

InputStream in = assetManager.open(rarPath);
OutputStream out = new
    FileOutputStream("/sdcard/myapk.apk");

byte[] buffer = new byte[1024];
int read;
while((read = in.read(buffer)) != -1)
    out.write(buffer, 0, read);
```

# Ví dụ: cài apk từ asset

---

```
in.close();  
out.flush();  
out.close();
```

```
Intent intent = new Intent(Intent.ACTION_VIEW);  
intent.setDataAndType(  
    Uri.fromFile(new File("/sdcard/myapk.apk")),  
    "application/vnd.android.package-archive"  
);
```

```
startActivity(intent);
```

Phần 4

# SQLite

# Giới thiệu

---

- SQLite là một CSDL nhỏ gọn, viết bằng C++ và được tích hợp trên rất nhiều hệ điều hành di động
- SQLite được tích hợp vào HĐH, vì thế mọi ứng dụng đều có thể làm việc được mà không cần thư viện hỗ trợ
- Mỗi CSDL SQLite thường là một file duy nhất, LTV mở file đó (giống như mở file thông thường) sau đó thực hiện các câu lệnh SQL để thao tác file
- Không nên lạm dụng SQLite vì khá chậm (làm việc trên text) và thiếu uyển chuyển

# SQLite API

---

- Gói `android.database.sqlite` chứa các class hỗ trợ làm việc với CSDL SQLite, 2 class quan trọng:
  - `SQLiteDatabase`: class giúp chúng ta làm việc trực tiếp với file CSDL, thực thi các thao tác CSDL bằng SQL hoặc bằng các phương thức hỗ trợ của class
  - `SQLiteOpenHelper`: class giúp lập trình viên quản lý việc, tạo và nâng cấp file CSDL
- Android SDK cung cấp công cụ `sqlite3` giúp tương tác với CSDL thông qua dòng lệnh, các LTV có thể dùng công cụ này để kiểm tra lại kết quả làm việc với file CSDL một cách nhanh chóng



# Các method của SQLiteDatabase

---

- Tạo/Mở CSDL: `openDatabase`
- Đóng CSDL: `close`
- Thực thi SQL: `execSQL`
- Chèn dữ liệu: `insert`
- Cập nhật dữ liệu: `update`
- Xóa dữ liệu: `delete`
- Thực hiện truy vấn SELECT: `rawQuery`

# Ví dụ đơn giản

---

- Xây dựng CSDL quản lý Sách
  - Bảng Books
    - BookID INT
    - BookName TEXT
    - Page INT
    - Price FLOAT
    - Description TEXT
  - Sau khi tạo xong bảng thì chèn 5 bản ghi vào bảng
  - Thực hiện các câu lệnh xóa có điều kiện
  - Cập nhập giá tiền, tên sách theo mã sách
  - Tìm kiếm sách lần lượt với các điều kiện như: mã, tên gần đúng (sử dụng like), giá tiền...

# Tạo database bằng code

---

```
String sqltext = "DROP TABLE IF EXISTS BOOKS;\n"  
+ "CREATE TABLE BOOKS(BookID integer PRIMARY KEY, BookName text,  
Page integer, Price Float, Description text);\n"  
+ "INSERT INTO BOOKS VALUES(1, 'Java', 100, 9.99, 'sách về  
java');\n"  
+ "INSERT INTO BOOKS VALUES(2, 'Android', 320, 19.00, 'Android cơ  
bản');\n"  
+ "INSERT INTO BOOKS VALUES(3, 'Học làm giàu', 120, 0.99, 'sách đọc  
cho vui');\n"  
+ "INSERT INTO BOOKS VALUES(4, 'Tủ điển Anh-Việt', 1000, 29.50, 'Tủ  
điển 100.000 từ');\n"  
+ "INSERT INTO BOOKS VALUES(5, 'CNXH', 1, 1, 'chuyện cổ tích');";
```

# Tạo database bằng code

---

// tạo DB và thực hiện một số câu SQL

```
SQLiteDatabase db = openOrCreateDatabase("books.db", MODE_PRIVATE,  
null);
```

```
for (String sql : sqltext.split("\n"))
```

```
    db.execSQL(sql);
```

```
db.close();
```

# Xem kết quả truy vấn SELECT

---

```
bPrev = (Button) findViewById(R.id.button1);  
bNext = (Button) findViewById(R.id.button2);  
bId = (TextView) findViewById(R.id.textView1);  
bName = (TextView) findViewById(R.id.textView2);  
bPage = (TextView) findViewById(R.id.textView3);  
bPrice = (TextView) findViewById(R.id.textView4);  
bDes = (TextView) findViewById(R.id.textView5);
```

# Xem kết quả truy vấn SELECT

---

```
try {  
    db = openOrCreateDatabase("books.db", MODE_PRIVATE, null);  
    cs = db.rawQuery("SELECT * FROM BOOKS", null);  
}  
catch (Exception e) {  
    finish();  
}  
cs.moveToNext();  
updateRecord();
```

# Xem kết quả truy vấn SELECT

---

```
void updateRecord() {  
    bId.setText(cs.getString(0));  
    bName.setText(cs.getString(1));  
    bPage.setText(cs.getString(2));  
    bPrice.setText(cs.getString(3));  
    bDes.setText(cs.getString(4));  
    bPrev.setEnabled(!cs.isFirst());  
    bNext.setEnabled(!cs.isLast());  
}
```

# Xem kết quả truy vấn SELECT

---

```
public void btnPrev(View v) {  
    cs.moveToPrevious();  
    updateRecord();  
}  
public void btnNext(View v) {  
    cs.moveToNext();  
    updateRecord();  
}  
public void onBackPressed(){  
    cs.close();  
    db.close();  
    super.onBackPressed();  
}
```