TRUNG TÂM TIN HỌC - ĐẠI HỌC KHOA HỌC TỰ NHIÊN
ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

CHƯƠNG TRÌNH ĐÀO TẠO
LẬP TRÌNH VIÊN CHUYÊN NGHIỆP TRÊN THIẾT BỊ DI ĐỘNG

TÀI LIỆU

LẬP TRÌNH THIẾT BỊ DI ĐỘNG TRÊN ANDROID

MODULE 03:
LẬP TRÌNH CƠ SỞ DỮ LIỆU, TIẾN TRÌNH VÀ CHUYỂN ĐỘNG TRONG ỨNG DỤNG

Tháng 06/2015

Bài 1
LƯU TRỮ, TRUY VẤN VÀ SẮP XẾP DỮ LIỆU VỚI SQLITE

1. KHÁI NIỆM CƠ SỞ DỮ LIỆU (DATABASE CONCEPTS)

- Dữ liệu là những sự kiện có thể ghi lại được và có ý nghĩa.
- Một cơ sở dữ liệu (CSDL) là một tập hợp dữ liệu có liên quan với nhau, được lưu trữ trên máy tính, có nhiều người sử dụng và được tổ chức theo một mô hình.
- Một CSDL biểu thị một khía cạnh nào đó của thế giới thực. Thông tin được đưa vào trong CSDL tạo thành một không gian CSDL hoặc một "thế giới nhỏ" (miniworld).
- Một CSDL là một tập hợp dữ liệu liên kết với nhau một cách logic và mang một ý nghĩa nào đó. Một CSDL được thiết kế và được phổ biến cho một mục đích riêng. Một hệ quản trị cơ sở dữ liệu là một tập chương trình giúp cho người sử dụng tạo ra, duy trì và khai thác CSDL. Người ta gọi *cơ sở dữ liệu* và *hệ quản trị cơ sở dữ liệu* bằng một thuật ngữ chung là *hệ cơ sở dữ liệu*.
- **Hệ quản trị cơ sở dữ liệu** (tiếng Anh: *Database Management System* - **DBMS**), là phần mềm hay hệ thống được thiết kế để quản trị một cơ sở dữ liệu. Cụ thể, các chương trình thuộc loại này hỗ trợ khả năng lưu trữ, sửa chữa, xóa và tìm kiếm thông tin trong một CSDL. Có rất nhiều loại hệ quản trị CSDL khác nhau: từ phần mềm nhỏ chạy trên máy tính cá nhân cho đến những hệ quản trị phức tạp chạy trên một hoặc nhiều siêu máy tính.
- Tuy nhiên, đa số hệ quản trị CSDL trên thị trường đều có một đặc điểm chung là sử dụng ngôn ngữ truy vấn theo cấu trúc mà tiếng Anh gọi là *Structured Query Language (SQL)*. Các hệ quản trị CSDL phổ biến được nhiều người biết đến là MySQL, Oracle, PostgreSQL, SQL Server, DB2, Infomix, v.v... Phần lớn các hệ quản trị CSDL kể trên hoạt động tốt trên nhiều hệ điều hành khác nhau như Linux, Unix và MacOS ngoại trừ SQL Server của Microsoft chỉ chạy trên hệ điều hành Windows.

2. GIỚI THIỆU SQLITE

- SQLite là cơ sở dữ liệu mở được viết dưới dạng thư viện tích hợp nhúng vào Android, hỗ trợ các đặc điểm về quan hệ chuẩn của cơ sở dữ liệu như cú pháp, transaction, các câu lệnh. SQLite được sử dụng rộng rãi trong các ứng dụng di động trên Android, iOS, … Mozilla Firefox cũng sử dụng SQLite để lưu trữ các dữ liệu về cấu hình.

- Với bộ thư viện được tích hợp sẵn SQLite sẽ giúp các lập trình viên có thể lưu trữ và phục hồi dữ liệu bất cứ lúc nào. Tuy chỉ là phiên bản rút gọn nhưng SQLite có thể đáp ứng được hầu hết các nhu cầu về xử lý dữ liệu. Tính linh động trong SQLite cũng giúp ta thể kiểm soát được các thông tin dữ liệu.
- Sử dụng SQLite không yêu cầu về thiết lập bất cứ cơ sở dữ liệu hoặc đòi hỏi quyền admin.
- SQLite hỗ trợ các kiểu dữ liệu : TEXT, INTEGER, REAL.
- Đường dẫn của cơ sở dữ liệu:

  DATA/data/<PACKAGE_NAME>/databases/FILENAME
- Thể hiện dữ liệu ở dạng bảng quan hệ:
  - Cột: thể hiện trường dữ liệu (hoặc thuộc tính dữ liệu).
  - Dòng: một thể hiện dữ liệu.

word	app id	frequency	Locale	_ID
mapreduce	user1	100	en_US	1
precompiler	user14	200	fr_FR	2
applet	user2	225	fr_CA	3
const	user1	255	pt_BR	4
Int	user5	100	en_UK	5

- Mặc định mỗi ứng dụng sẽ được cấp phát một thư mục cho việc lưu trữ cơ sở dữ liệu và nó chỉ có thể được dùng bởi ứng dụng đó. Nếu muốn chia sẻ dữ liệu dùng chung giữa các ứng dụng ta có thể sử dụng Content Provider.

3. XÂY DỰNG CƠ SỞ DỮ LIỆU VỚI SQLITE

3.1. Tạo Cơ sở dữ liệu

- Tạo cơ sở dữ liệu thông qua lớp SQLiteOpenHelper.
  - **SQLiteOpenHelper** là một lớp ảo, SQLiteOpenHelper giúp tạo các cơ sở dữ liệu dùng SQLite (vì SQLite không hỗ trợ các phương thức khởi tạo cơ sở dữ liệu). Vậy làm sao để sử dụng được SQLiteOpenHelper? Vì SQLiteOpenHelper là một lớp ảo nên ta cần khai báo một lớp khác kế thừa lớp này.
  - SQLiteOpenHelper thực hiện các phương thức cần thiết cho phép khởi tạo, nâng cấp cơ sở dữ liệu. Tạo đối tượng để truy cấp cơ sở dữ liệu (Read và Write).
  - SQLiteDatabase cung cấp phương thức *insert(), update(), delete(),* hoặc *execSQL()* cho phép thực hiện truy xuất dữ liệu.
- *Override* phương thức onCreate() để tạo cơ sở dữ liệu.
- *Override* phương thức onUpgrade() để nâng cấp cơ sở dữ liệu.

- Lớp SQLiteOpenHelper cung cấp 2 phương thức getReadableDatabase() và getWriteableDatabase() để trả về đối tượng *SQLiteDatabase*.

Ví dụ:

- Tạo một CSDL có tên: COUNTRY_DB
- Trong đó:
  - Tạo một Table có tên: COUNTRY.
  - Table COUNTRY có 3 cột là: _id, enName và viName với _id là khóa chính và có giá trị tự động tăng.

```
public class MySQLiteHelper extends SQLiteOpenHelper {
     public static final String DB_NAME = "COUNTRY_DB";
     public static final String TABLE_NAME = "COUNTRY   ";
     public static final String COL_ID = "_id";
     public static final String COL_EN_NAME = "enName";
     public static final String COL_VI_NAME = "viName";
     public static final String CREATE_TABLE = "CREATE TABLE "
            + TABLE_NAME + " ("
            + COL_ID + " integer primary key autoincrement, "
            + COL_EN_NAME + " text not null, "
            + COL_VI_NAME + " text not null);";.

     public MySQLiteHelper(Context context) {
               super(context, DB_NAME, null, 1);
     }
}
```

- *Lưu ý*: Hàm khởi tạo có các thông số sau:
  - **Context**: Biến ngữ cảnh.
  - **DB_NAME**: Tên của cơ sở dữ liệu.
  - **CursorFactory**: Đôi lúc chúng ta có thể extend lớp cursor để kế thừa một số phương thức và truy vấn. Trong trường hợp đó, ta dùng một instance của CursorFactory để tham chiếu đến lớp chúng ta tạo thay cho mặc định. Khi dùng mặc định thì ta để nó null.
  - **Version**: Version của cơ sở dữ liệu.
- Tiến hành tạo CSDL trong hàm onCreate():

```
@Override
public void onCreate(SQLiteDatabase db) {
    Log.d("SQLite", "On Create Database");
```

```
    //Tạo Table với câu lệnh execSQL
     db.execSQL(CREATE_TABLE);
 }

 @Override
 public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
 }
```

3.2. Mở Cơ sở dữ liệu

- Dùng cú pháp getReadableDatabase() và getWriteableDatabase() để trả về đối tượng SQLiteDatabase cần sử dụng.
- SQLiteDatabase là đối tượng để truy cập cơ sở dữ liệu (Read và Write).
- SQLiteDatabase cung cấp phương thức insert(), update(), delete(), hoặc execSQL() cho phép thực hiện truy xuất dữ liệu.

```
MySQLiteHelper myHelper = new MySQLiteHelper(context);
SQLiteDatabase database;
try{
     database = myHelper.getWritableDatabase();
}catch(SQLiteException ex){
    database = myHelper.getReadableDatabase();
}
```

- Nếu cơ sở dữ liệu đã tồn tại thì lớp SQLiteOpenHelper sẽ không tạo thêm mà chỉ mở kết nối đến cơ sở dữ liệu đó.

3.3. Đóng Cơ sở dữ liệu

```
public void close(){
  myHelper.close();
}
```

4. TRUY VẤN DỮ LIỆU

4.1. Truy vấn (Query)

4.1.1. Truy vấn với câu lệnh SQL

- Sử dụng phương thức rawQuery() của lớp SQLiteDatabase.
- Phương thức rawQuery() nhận vào một giá trị chuỗi là câu lệnh SQL dùng để truy vấn dữ liệu và trả ra một đối tượng Cursor.

```
String queryStr = “SELECT * FROM COUNTRY WHERE _id=? AND
                                                    enName=?”;
Cursor cursor = database.rawQuery(queryStr, new
                                          String[]{“1”,”Vietnam”});
```

- Cú pháp truy xuất dữ liệu trong câu lệnh SQL:
  - Phát biểu truy vấn SQL có dạng:

```
SELECT {Tên trường cần truy vấn, * nếu lấy tất cả các trường}
FROM {Tên Table}
WHERE {Biểu thức điều kiện}
```

- SELECT dùng để đọc thông tin từ cơ sở dữ liệu theo trường hợp quy định hay những biểu thức cho trường hợp đó.
- FROM chỉ ra tên một bảng hay những bảng có liên quan cần truy vấn thông tin.
- WHERE để tạo nên điều kiện cần lọc mẩu tin theo tiêu chuẩn được định nghĩa. Thông thường, WHERE dùng cột (trường) để so sánh với giá trị cột khác, hay biểu thức chứa cột (trường) bất kỳ có trong bảng (table).

Bảng "T"	Câu truy vấn	Kết quả
C1 C2 1 a 2 b	`SELECT * FROM T;`	C1 C2 1 a 2 b
C1 C2 1 a 2 b	`SELECT C1 FROM T;`	C1 1 2
C1 C2 1 a 2 b	`SELECT * FROM T WHERE C1 = 1;`	C1 C2 1 a

Hình 3 1. Ví dụ truy vấn với câu lệnh SELECT.

- Các phép toán so sánh trong câu lệnh SQL:

>	Lớn WHERE _id>10; < : nhỏ hơn WHERE _id<10;
>=	Lớn hơn hoặc bằng WHERE _id>=10;
<=	Nhỏ hơn hoặc bằng WHERE _id<=10;
=	Bằng WHERE _id=1;
!=	Khác (ko bằng) WHERE _id!=1;

- Các phép toán logic trong câu lệnh SQL:

AND	WHERE `_id`=1 AND `enName`='admin';
OR	WHERE `_id`=1 OR `enName`='admin';
NOT	WHERE enName is not NULL;
NOT IN	WHERE _id NOT IN('10','20');
BETWEEN	WHERE _id BETWEEN 10 And 20;
LIKE	WHERE enName LIKE '%Vietnam';
NOT LIKE	WHERE enName NOT LIKE '%Vietnam ;
IN	WHERE _id IN ('10','20','30');

4.1.2. Truy vấn với cú pháp hàm

- Sử dụng phương thức query() của lớp SQLiteDatabase.
- Hỗ trợ lập trình viên truy xuất dữ liệu từ CSDL mà không cần biết cú pháp các câu lệnh SQL.
- Phương thức query() cũng trả ra một đối tượng Cursor. Để lấy dữ liệu từ Cursor, trước hết cần chuyển đến vị trí xác định với các phương thức: *moveToFirst(), moveToNext(), moveToPosition(int position)…* như được giới thiệu ở Bài 2.
- Cursor:
    - Đối tượng dữ liệu được trả về khi thực hiện truy vấn dữ liệu trong cơ sở dữ liệu của SQLite hoặc Content Provider.
    - Thể hiện dữ liệu ở dạng bảng quan hệ :
        - Cột: thể hiện trường dữ liệu (hoặc thuộc tính dữ liệu).

- Dòng: một thể hiện dữ liệu.

```
Cursor cursor = database.query(String table,
                               String[] columns,
                               String selection,
                               String[] selectionArgs,
                               String groupBy,
                               String having,
                               String orderBy);
```

String table: tên của bảng cần truy vấn.
String[] columns: danh sách các cột sẽ trả về dữ liệu.

String selection: chứa các điều kiện truy vấn.

String[] selectionArgs: danh sách các tham số phụ cho câu điều kiện.

String[] groupBy: gom nhóm các cột kết quả.

String[] having: bộ lọc theo điều kiện.

String[] orderBy: sắp xếp theo mảng cột được chỉ định.

Ví dụ:

- Sử dụng phương thức query(), lấy ra tên tiếng Anh và tên tiếng Việt của dòng có _id là 1.

```
String selectedColumns = new String[]{ COL_EN_NAME, COL_VI_NAME};
Cursor c = database.query(TABLE_NAME, selectedColumns, “_id=?”,
new String[]{“1”}, null, null, null);
if (c.moveToFirst()) {
      while (!c.isAfterLast())
      {
           String enName =
               c.getString(c.getColumnIndex(selectedColumns[0]));
           String viName =
               c.getString(c.getColumnIndex(selectedColumns[1]));
          c.moveToNext();
      }
}
```

4.2. Quản lý

- Lớp SQLiteDatabase cung cấp các phương thức như: insert(), update(), delete() hoặc execSQL() cho phép người dùng có thể quản lý và truy xuất dữ liệu.

4.2.1. Insert

- Tạo một dòng mới để insert dữ liệu:

```
ContentValues newValues = new ContentValues();
```

- Gán giá trị vào mỗi cột trong dòng vừa tạo:

```
newValues.put(COLUMN_NAME,values);
```

- Ví dụ:

```
newValues.put(COL_EN_NAME, “Germany”);
newValues.put(COL_VI_NAME, “Đức”);
//Thêm dòng đó vào database
database.insert(TABLE_NAME, null, newValues);
```

4.2.2. Update

- Tạo một dòng mới để update dữ liệu:

```
ContentValues newValues = new ContentValues();
```

- Gán giá trị vào mỗi cột trong dòng vừa tạo:

```
newValues.put(COLUMN_NAME, values);
```

- Ví dụ:

```
newValues.put(COL_EN_NAME,”Germany2”);
```

  - Cập nhật lại dòng vừa tạo với đúng mệnh đề WHERE:

```
String where = “_id=” + rowId;
database.update(TABLE_NAME, newValues, where, null);
```

4.2.3. Delete

- Sử dụng câu truy vấn delete truyền vào tên bảng và chỉ số của hàng cần xóa trong mệnh đề WHERE.

- Nếu mệnh đề WHERE không có sẽ thực hiện xóa tất cả các dòng:

```
database.delete(TABLE_NAME, "_id=" + rowID, null);
```

5. SẮP XẾP DỮ LIỆU

5.1. Truy vấn sắp xếp

- Thông thường trong khi truy vấn mẩu tin từ bảng dữ liệu, kết quả hiển thị sắp xếp theo chiều tăng hay giảm dựa trên ký tự ALPHABET. Nhưng bạn cũng có thể sắp xếp theo một tiêu chuẩn bất kỳ .
- Cú pháp cho mệnh đề ORDER BY cùng với trạng thái tăng dần (ASCENDING) hoặc giảm dần (DESCENDING):

5.1.1. Tăng dần (ASCENDING)

```
ORDER BY columnName ASC
ORDER BY columnName1 + columnName2 ASC
```

- Ví dụ:
  - Truy vấn với câu lệnh SQL:

```
SELECT * FROM COUNTRY ORDER BY enName ASC
```

  - Truy vấn với cú pháp hàm:

```
String orderBy =  myHelper.COL_EN_NAME + " ASC";
database.query(TABLE_NAME, null, null, null, null, orderBy);
```

5.1.2. Giảm dần (DESCENDING)

```
ORDER BY columNname DESC
ORDER BY columNname1 + columnName2 DESC
```

- Ví dụ:
  - Truy vấn với câu lệnh SQL:

```
SELECT * FROM COUNTRY ORDER BY enName DESC
```

  - Truy vấn với cú pháp hàm:

```
String orderBy =  myHelper.COL_VI_NAME + " DESC";
```

```
database.query(TABLE_NAME, null, null, null, null, orderBy);
```

5.2. Truy vấn gom nhóm

- Ví dụ minh họa:
  - Ta có CSDL như sau:

_id	MaSo	TenBaiHat	LoiMoDau	TacGia
424	54630	1 2 3 CHIA ĐÔI LỐI ...	Nuốt nước mắt khẽ...	Bảo Chinh
425	55205	1 2 3 CHIA TAY	Một là yêu em, hai ...	Nhất Trung
426	53097	1 2 3 DZÔ	Uống cho say cùng...	Anh Khanh ; Minh ...
427	52591	1,2,3 NGÔI SAO	Một ngày anh nhớ,...	Sỹ Luân
428	54379	12 GIỜ	Mười hai giờ rung ...	Nguyễn Duy Hùng
429	54378	100% YÊU EM	Mãi mãi bên nhau ...	Bảo Thạch
430	53098	1000 LÝ DO ANH Đ...	Thôi nhắn tin làm c...	Vũ Quốc Bình ; Vũ ...
431	52858	2+1=0	Biết làm gì khi có h...	Nguyễn Hoài Anh
432	55206	24 GIỜ 7 NGÀY	Tiếng nói ấm áp đâ...	Huy Tuấn
433	53366	8 VẠN 6 NGÀN 400 ...	Ngày tháng trôi qu...	Nhất Trung
434	51279	9 CON SỐ 1 LINH ...	Em còn nhớ không...	Quốc Dũng
435	50931	999 ĐÓA HỒNG	Nhắc chi chuyện x...	Nhạc Hoa
436	54152	À ƠI HAI TIẾNG GI...	Từ ngày đưa mẹ tôi...	Huy Tập

Hình 1.1. CSDL mẫu.

  - Gom nhóm các bài hát có MaSo lớn hơn 55500 bằng cú pháp GROUP BY và mệnh đề HAVING với cú pháp:

```
SELECT column1, column2
FROM table1, table2
WHERE [ conditions ]
GROUP BY column1, column2
HAVING [ conditions ]
```

  - Có thể truy vấn với câu lệnh SQL:

```
SELECT * FROM BAIHATCUTHE GROUP BY TenBaiHat HAVING MaSo>55500
```

  - Hoặc truy vấn với cú pháp hàm:

```
String groupBy =  “BAIHATCUTHE”;
String having = “Maso>55500”
```

```
database.query(TABLE_NAME, null, null, groupBy, having, null);
```

- Kết quả:

_id	MaSo	TenBaiHat	LoiMoDau	TacGia
5872	55667	1, 2, 3 KHÓC	Màn đêm xuống dầ...	Khánh Đơn
5873	55668	AI KHỔ VÌ AI	Anh biết chăng an...	Thương Linh
6345	55950	ANH CẦN EM	Anh ngồi suy nghĩ ...	Khắc Việt
6346	55951	ANH KHÁC HAY E...	Có phải em hết yêu...	Khắc Việt
6347	55952	ANH KHÔNG LÀ GI...	Khi em quen anh b...	Phạm Hoàng Long
6108	55713	ANH LÀ ÁNH SAO ...	Đời như giọt sương...	Trung Lê
6349	55954	ANH MUỐN NÓI	Trong tình yêu ngư...	Minh Vy
6348	55953	ANH MƠ	Sáng nay anh được...	Anh Khang
6350	55955	ANH QUÊN BẢN T...	Ngày bên nhau an...	Hà Hải Đăng
6351	55956	ANH TRÔI VỀ EM	Mưa lại rơi ngẩn ng...	Huy Tuấn
6352	55957	ANH VẪN THỨ THA	Ngày tháng trống v...	Đỗ Đình Phúc
5878	55669	ANH VỀ ĐI	Có những lúc em t...	Hà Thái Hoàng
6357	55962	BA BÀ ĐI BÁN LƠN...	Ba bà đi bán lơn co...	Lê Cao Phan

Hình 1.2. Kết quả sau khi gom nhóm dữ liệu.

Bài 2
QUẢN LÝ DỮ LIỆU VỚI CONTENT PROVIDER

1. GIỚI THIỆU CONTENT PROVIDER

1.1. Content Provider

- Content Provider là một dạng lưu trữ cơ sở dữ liệu - nơi mà ta có thể tương tác với dữ liệu dạng SQLite và chia sẻ dữ liệu cho các ứng dụng khác. Trong các thiết bị Android cũng có sẵn một số Content Provider dùng chung như danh bạ, các tập tin đa phương tiện, …
- Nói cách khác, Content Provider là nơi đóng gói các dữ liệu cho phép các thành phần trong ứng dụng hoặc các ứng dụng khác truy cập và sử dụng.
- Thể hiện dữ liệu ở dạng bảng quan hệ:
    - Cột: thể hiện trường dữ liệu (hoặc thuộc tính dữ liệu).
    - Dòng: một thể hiện dữ liệu.
- ContentProvider được cung cấp sẵn trong hệ thống:
    - Hệ thống báo thức (Alarm).
    - Nội dung trình duyệt (Browser).
    - Danh bạ (Contacts).
    - Lịch (Calendar).
    - Tập tin tài liệu (Document).
    - Tập tin đa truyền thông (Media Store).
    - Tùy chỉnh hệ thống (Setting).
    - Điện thoại (Telephony).
    - Từ điển người dùng (UserDictionary).
    - Hộp thư thoại (VoiceMailContract).
- Cách thức hoạt động của Content Provider:
    - Mô hình dữ liệu theo định dạng cột và dòng giống như cơ sở dữ liệu. Với mỗi cột là trường dữ liệu và mỗi dòng là một record.
    - Cách thức hoạt động giống giao thức REST trong web.
    - Sử dụng chuỗi URI để thực hiện việc rút trích cũng như lưu dữ liệu trong content provider.

Ví dụ: content://com.android.book.BookProvider

content://com.android.book.BookProvider/7

Android cung cấp một số các gói Content Provider dùng cho việc lưu trữ chung như:

- Browser.
- CallLog.
- Contact (people, phones, photos, group).
- MediaStore (music, video, image).
- Setting.

Cần khai báo trong file AndroidManifest.xml cho biết ứng dụng sử dụng những Content Provider nào.

```
<provider
            android:name= “SomeProvider”
            android:authorities= “com.your-company.SomeProvider”>
 <provider
            android:name= “NotePadProvider”
            android:authorities= “com.your-company.NodePad”>
```

1.2. Content URI

- Đối tượng cho phép định nghĩa cách thức truy xuất dữ liệu trên Content Provider, bao gồm:
  - **Authority**: Tên đầy đủ của Provider.
  - **Path**: tên bảng tồn tại trong Provider
- Ví dụ về Content URI truy xuất dữ liệu danh bạ:
  - ContactsContract.Data.CONTENT_URI
  - content://com.android.contact/data
    - Trong đó:
      - content: scheme.
      - com.android.contact: authority.
      - data: tên bảng.

1.3. ContentResolver

- Đối tượng cho phép truy xuất dữ liệu và thao tác trên Content Provider, bao gồm:
  - Query.
  - Insert.
  - Update.
  - Delete.
- Truy xuất đối tượng ContentResolver thông qua phương thức:
  - **getContentResolver();**
- Phương thức Query:
  Cú pháp:
  - **query(Uri, projection, selection, selectionArgs, sortOrder)**

- Uri: đường dẫn truy xuất dữ liệu. – URI.
- Projection: mảng tên các trường dữ liệu (cột) cần truy xuất. – String[]
- Selection: câu điều kiện truy vấn. – String.
- SelectionArgs: mảng biến tham số cho câu điều kiện truy vấn. – String[]
- SortOrder: tiêu chí sắp xếp dữ liệu truy vấn (tên cột). - String

Kết quả trả về:

- Cursor: chứa bảng dữ liệu kết quả truy vấn.

Ví dụ truy xuất tất cả dữ liệu danh bạ trên thiết bị:

```
// Định nghĩa đường dẫn truy xuất
Uri uri = ContactsContract.Data.Content_URI;
// Định nghĩa tham số truy vấn
String projection = null;  // Lấy ra tất cả các cột
String selection = null; // Không có điều kiện truy vấn
String selectioArgs = null; // Không có tham số truy vấn
String sortOrder = null; // Sắp xếp mặc định
// Đối tượng truy vấn
ContentResolver cr = this.getContentResolver();
Cursor cursor = cr.query(Uri, projection, selection, selectionArgs,
sortOrder);
```

Ví dụ truy xuất dữ liệu danh bạ trên thiết bị có tên bắt đầu bằng “M” và sắp xếp theo theo ID.

```
Uri uri = ContactsContract.Data.Content_URI; //Định nghĩa đường dẫn
truy xuất
// Định nghĩa tham số truy vấn
String projection = null;  // Lấy ra tất cả các cột
String selection = Data.DISPLAY_NAME + “ like ?”; // điều kiện truy vấn
String selectioArgs = { “M%”}; // Tham số truy vấn
String sortOrder = Data._ID; // Sắp xếp theo ID
// Đối tượng truy vấn
ContentResolver cr = this.getContentResolver();
Cursor cursor = cr.query(Uri, projection, selection, selectionArgs,
sortOrder);
```

- Phương thức Insert, Update: được thực hiện thông qua tham số dữ liệu **ContentValues**:
    - ContentValues**:** cho phép định nghĩa dữ liệu khi thao tác, thông qua phương thức put(Key, Value).
    - Trong đó:
        - Key: tên trường dữ liệu (cột). - String
        - Value: dữ liệu tương ứng với trường dữ liệu – String, Float, Long…
- Phương thức Insert:

  Cú pháp:
    - **insert(Uri, ContentValues)**
        - Uri: đường dẫn truy xuất dữ liệu. – URI.
        - ContentValues: dữ liệu cần thêm vào. - ContentValues.

  Kết quả trả về:
    - URI: đường dẫn đến dữ liệu vừa thêm vào.
- Phương thức Update:

  Cú pháp:
    - **update(Uri, ContentValues, where, selectionArgs)**
        - Uri: đường dẫn truy xuất dữ liệu. – URI.
        - ContentValues: dữ liệu cần cập nhật. – ContentValues.
        - Where: điều kiện dữ liệu được cập nhật. – String.
        - SelectionArgs: mảng tham số cho câu điều kiện. – String[].

  Kết quả trả về:
    - int: số lượng dòng dữ liệu đã được cập nhật.
- Phương thức Delete:

  Cú pháp:
    - **delete(Uri, ContentValues, where, selectionArgs)**
        - Uri: đường dẫn truy xuất dữ liệu. – URI.
        - Where: điều kiện dữ liệu được xóa. – String.
        - SelectionArgs: mảng tham số cho câu điều kiện. – String[].

  Kết quả trả về:
    - int: số lượng dòng dữ liệu đã được xóa.

2. XÂY DỰNG CONTENT PROVIDER CHO ỨNG DỤNG

2.1. Vấn đề xây dựng ContentProvider

- Việc sử dụng ContentProvider trong ứng dụng khi cần xây dựng những tính năng sau:
    - Cung cấp các dữ liệu hoặc tập tin đặc trưng, phức tạp cho những ứng dụng khác.
    - Cho phép người dùng sao chép các dữ liệu vào các ứng dụng khác.
    - Sử dụng và tùy chỉnh tính năng tìm kiếm dữ liệu trong ứng dụng.
- Thực hiện xây dựng ContentProvider:
    - Xây dựng dữ liệu:
        - Dữ liệu tập tin (not recommended).

        - Dữ liệu có cấu trúc (SQLite).
    - Thiết kế các biến truy xuất dữ liệu (Contract class):
        - Authority – Content URI (required).
        - Biến đại diện cho các trường dữ liệu (tên cột).
    - Tạo lớp kế thừa và thực hiện các phương thức truy vấn dữ liệu
        - ContentProvider (required)
            - onCreate.
            - query - delete - insert – update.
- Khai báo lớp sử dụng trong Manifest thông qua thẻ <provider/>
    - Khai báo Permission.

2.2. Authority & UriMatcher

2.2.1. Authority:

- Định nghĩa đường dẫn truy xuất vào ContentProvider.
    - Định nghĩa đề xuất: **<pakage_name>.<app_name>.provider**

  Ví dụ:

  com.htsi.t3h.onlineshop.provider

- Định nghĩa đường dẫn truy xuất vào từng bảng riêng biệt trong ContentProvider.
    - Định nghĩa đề xuất: **<authority>/<table_name>**

  Ví dụ:

  com.htsi.t3h.onlineshop.provider/accessory

  com.htsi.t3h.onlineshop.provider/clothes

2.2.2. UriMatcher:

- Đối tượng cho phép kiểm tra các Content URI truy xuất vào Content Provider.
    - Sử dụng dấu đại diện:

      * : Cho phép truy xuất vào tất cả các dạng dữ liệu.

      # : Cho phép truy xuất vào một dòng trong một bảng.

  Ví dụ định nghĩa Content URI truy xuất dữ liệu:

- Truy xuất tất cả các bảng trong ContentProvider:
  content://com.htsi.t3h.onlineshop.provider/*
- Truy xuất tất cả các dữ liệu trong bảng accessory:
  content://com.htsi.t3h.onlineshop.provider/accessory/*
- Truy xuất dòng dữ liệu có id = # trong bảng accessory:
  content://com.htsi.t3h.onlineshop.provider/accessory/#

3. TRUY VẤN DỮ LIỆU HỆ THỐNG VỚI CONTENT PROVIDER

3.1. Các phương thức điều khiển Cursor

- Sử dụng Cursor để thực hiện các thao tác trên bảng dữ liệu của provider.

ID	ISBN	Tên sách	NXB
1	123456	Lập trình Android	T3H
2	987654	Lập trình iPhone	T3H
3	134679	Lập trình Java	T3H1
4	258456	Lập trình C#	T3H2

Sách của NXB T3H

Cursor →

ID	ISBN	Title	NXB
1	123456	Lập trình Android	T3H
2	987654	Lập trình iPhone	T3H

Hình 2.1. Error! No text of specified style in document - 1. Mô tả về Cursor.

- Một số các phương thức để điều khiển Cursor thao tác trên dữ liệu:
  - **movetoFisrt()**: trở về hàng đầu, false nếu cursors rỗng.
  - **movetoNext()**: chuyển đến hàng tiếp theo, false nếu đang ở hàng cuối cùng.
  - **movetoPrevious()**: chuyển đến hàng trước đó, false nếu đang ở đầu.
  - **movetoPosition(int position)**: chuyển đến vị trí xác định, false nếu vị trí không thể tới.
  - **getColumnName(int columnIndex)**: trả về tên cột nếu biết chỉ số cột.
  - **getColumnIndex(String columnName)**: trả về chỉ số cột nếu biết tên cột.
  - **moveToLast()**: chuyển đến vị trí cuối cùng, false nếu danh sách rỗng.
  - **getString(int columnIndex)**: lấy giá trị ở cột có chỉ số truyền vào.
  - Ngoài ra thì còn một số hàm kiểm tra khác như: **boolean isAfterLast(), isBeforeFirst(), isNull(columnIndex)** …

Ví dụ:

```
if(cursor.moveToFirst() == false){
      //không có dữ liệu
      return;
 }
 //Cursor đã ở dòng dữ liệu đầu tiên.
 //Thử truy xuất vào một cột nào đó để lấy dữ liệu
 int columnIndex = cursor.getColumnIndex(People.NAME);
```

```
            String name = cursor.getString(columnIndex);
            while(cursor.moveToNext()){
            //tiến hành truy cập vào các trường khác trong bảng
                 cursor.moveToNext();}
```

3.2. Truy vấn dữ liệu hệ thống

- Sử dụng Content Provider để truy xuất vào dữ liệu của SMS trên điện thoại:
  - Sử dụng Uri : content://sms/inbox.
  - Tạo mảng chứa tên cột cần truy xuất dữ liệu.
  - Duyệt Cursor qua bảng dữ liệu để lấy những thông tin cần thiết.

```
 public void getInboxMess(ContentResolver content) {
    String[] projection = { "address", "body" }
    Cursor cursor = content.query(Uri.parse("content://sms/inbox"),
                                  projection, null, null, null);
    if (cursor.moveToFirst()) {
       while (!cursor.isAfterLast()) {
          String address =
cursor.getString(c.getColumnIndex(projection[0]));
          String body =
cursor.getString(c.getColumnIndex(projection[1]));
          Log.d("SMS", "Adress: " + address + " - Body: " + body);
           cursor.moveToNext();
        }
     }
}
```

Bài 3
MENU - ACTION BAR – TOOLBAR

1. MENU

Menu là dạng Widget chứa các chức năng phụ hoặc các tùy chỉnh dành riêng cho từng ứng dụng. Bao gồm các dạng cơ bản:

1.1. Option Menu

- Là menu chính trong ứng dụng chứa các thao tác cơ bản cho một ứng dụng được gọi khi người dùng nhấn phím Menu.
- Từ phiên bản Android 2.3 trở xuống, thao tác gọi Menu được thực hiện bằng phím Menu trên thiết bị.

Hình 1.1. OptionMenu.

- Từ phiên bản Android 3.0 trở đi, Option Menu được tích hợp vào thanh Action Bar.
  - Tạo Menu từ trong tập tin XML:

```
<menu
xmlns:android="http://schemas.android.com/apk/res/android" >
   <item
         android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:showAsAction="ifRoom "
        android:title="@string/action_settings"/>
<item
         android:id="@+id/action_help"
        android:title="@string/action_help"/>
</menu>
```

- Sử dụng trong Java code qua hàm onCreateOptionMenu():

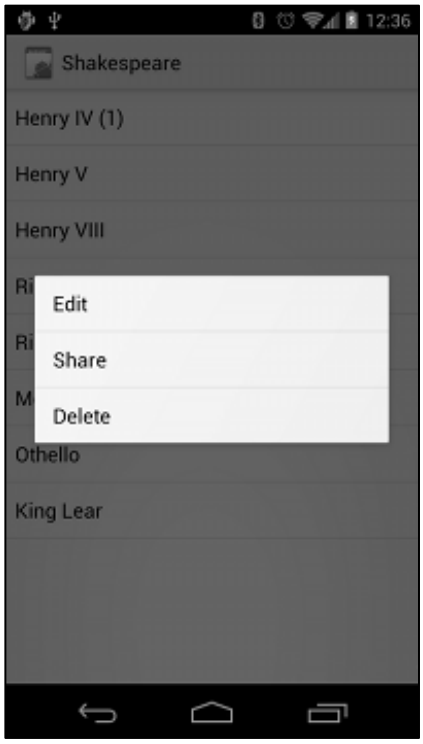
```
@Override
public void onCreateOptionMenu(Menu menu) {
        MenuInflater   inflate  = get MenuInflater;
         Inflate. inflate  (R.menu.game_menu, menu);
         return true;
}
```

- Xử lý sự kiện trong OptionMenu:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
           switch (item.getItemId()) {
  case R.id.action_setting:
    //show Setting
    break;
  case R.id.action_help:
    break;
           }
}
```

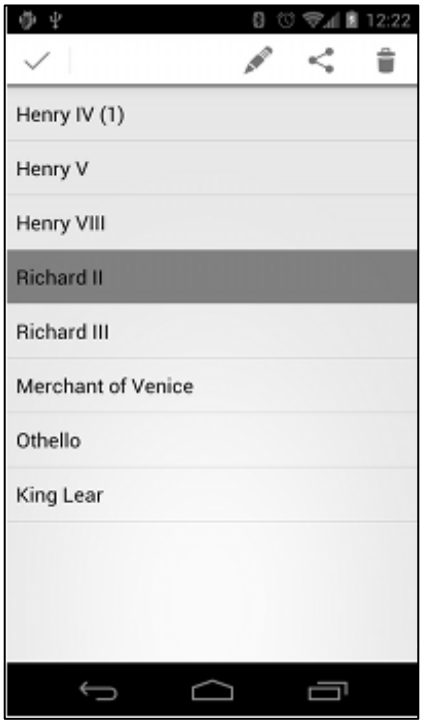
1.2. Context Menu

- Dạng menu xuất hiện khi người dùng tương tác với các Item trên ViewGroup, thường là ListView hoặc GridView.
- Bao gồm 2 dạng:
  - *Floating Context Menu:* dạng menu hiển thị khi người dùng nhấn và giữ một item trên ViewGroup (giống như Dialog).



Hình 1.2. Floating Context Menu.

  - *Contextual Action Mode (API level 11):* một thanh công cụ hiển thị phía trên ứng dụng, nó cho phép người thực hiện nhiều thao tác khác nhau trên Item được lựa chọn, hoặc thực hiện một thao tác trên nhiều Item nếu ứng dụng có hỗ trợ.



Hình 1.3. Contextual Action Mode.

- Tạo Floating Context Menu
  - Đăng ký đối tượng View sẽ sử dụng bằng phương thức **registerForcontextMenu(View)**
  - Thực hiện override phương thức onCreateContextMenu()

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
ContextMenuInfo menuInfo) {
        super.onCreateContextMenu(menu, v, menuInfo);
        getMenuInflater().inflate(R.menu.context_menu, menu);
}
```

  - Xử lý sự kiện trên Floating Context Menu bằng cách override phương thức oncontextItemSelected()

```
@Override
public boolean onContextItemSelected(MenuItem item) {
```

```
AdpterContextMenuInfo info = (AdapterContextMenuInfo)
item.getMenuInfo();
    switch (item.getItemId()) {
       case R.id.edit:
            editItem(info.id);
            break;
       case R.id.delete:
            deleteItem(info.id);
            break;
    }
}
```

- Khai báo và sử dụng Contextual Action Mode (API level 11)
  - Có 2 kiểu khai báo:
    - Khai báo dành cho đối tượng View: khai báo biến ActionMode call back để khởi tạo hàm onCreateActionMode()

```
private ActionMode.Callback mActionModeCallback = new
ActionMode.Callback() {
    @Override
    public boolean onCreateActionMode(ActionMode mode, Menu menu)
{

         MenuInflater inflater = mode.getMenuInflater();
         inflater.inflate(R.menu.context_menu, menu);
         return true;
    }
}
```

    - Thực hiện override hàm onActionItemClicked() để nhận biết các item nào trên menu được gọi thực hiên.

```
@Override
public boolean onActionItemClicked(ActionMode mode, MenuItem
item) {
```

```
        switch (item.getItemId()) {
            case R.id.menu_share:
       //Các hàm xử lý
                return true;
            default:
                return false;
        }
}
```

- Gọi thực hiện hàm setOnLongClickListener() nếu người dùng nhấn và giữ lâu trên đối tượng nào đó (someView).

```
someView.setOnLongClickListener(new View.OnLongClickListener() {
        // Called when the user long-clicks on someView
        public boolean onLongClick(View view) {
               if (mActionMode != null) {
                   return false;
               }

               mActionMode=
this.getActivity().startActionMode(mActionModeCallback);
               view.setSelected(true);
               return true;
         }
});
```

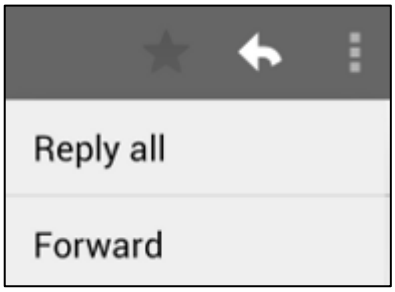
- Khai báo dành cho đối tượng ViewGroup

```
ListView listView = getListView();
listView.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE_MODAL);

listView.setMultiChoiceModeListener(new
MultiChoiceModeListener().
{
      @Override
      public    boolean    onActionItemClicked(ActionMode    mode,
MenuItem item) {
       }
```

```
      @Override
      public boolean onCreateActionMode(ActionMode mode, Menu
menu) {
              MenuInflater inflater = mode.getMenuInflater();
              inflater.inflate(R.menu.context, menu);
              return true;
      }
});
```

1.3. PopUp Menu

- Dạng menu hiển thị khi người dùng nhấn và giữ lâu trên một đối tượng.



Hình 1.4. PopupMenu.

- Ví dụ về tạo một Popup Menu:
  - Tạo đối tượng cần show Popup Menu

```
<ImageButton
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:src="@drawable/ic_overflow_holo_dark"
   android:contentDescription ="@string/descr_overflow_button"
  android:onClick="showPopup" />
```

  - Tạo xử lý hàm onClick cho đối tượng để hiện ra Popup Menu

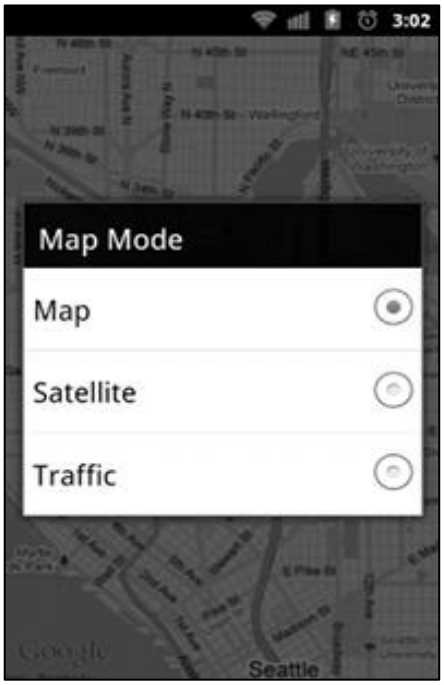
```
public void showPopup(View v) {
    PopupMenu popup = new PopupMenu(this, v);
    MenuInflater inflater = popup.getMenuInflater();
    inflater.inflate(R.menu.actions, popup.getMenu());
```

```
    popup.show(); }
```

- Thao tác xử lý trên từng Item của Menu:

```
  @Override public boolean onMenuItemClick(MenuItem item) {
     switch(item.getItemId()) {
          case R.id.archive:
               archive(item);
               return true;
          case R.id.delete:
               delete(item);
               return true;
          default:
               return false;
     }
  }
```

- Sử dụng Checkable Item Menu
  - Sử dụng các Radio Checkbox trên các item của Menu



Hình 1.5. Ví dụ về Checkable Item Menu.

- Cách khai báo:

```
<?xml version="1.0" encoding="utf-8"?>
<menu  xmlns:android="http://schemas.android.com/apk/res/android">
  <group android:checkableBehavior="single">
    <item android:id="@+id/red"
       android:title="@string/red" />
    <item
       android:id="@+id/blue"
       android:title="@string/blue" />
    </group>
</menu>
```

- Xử lý khi người dùng click chọn lên item:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
   switch (item.getItemId()) {
```

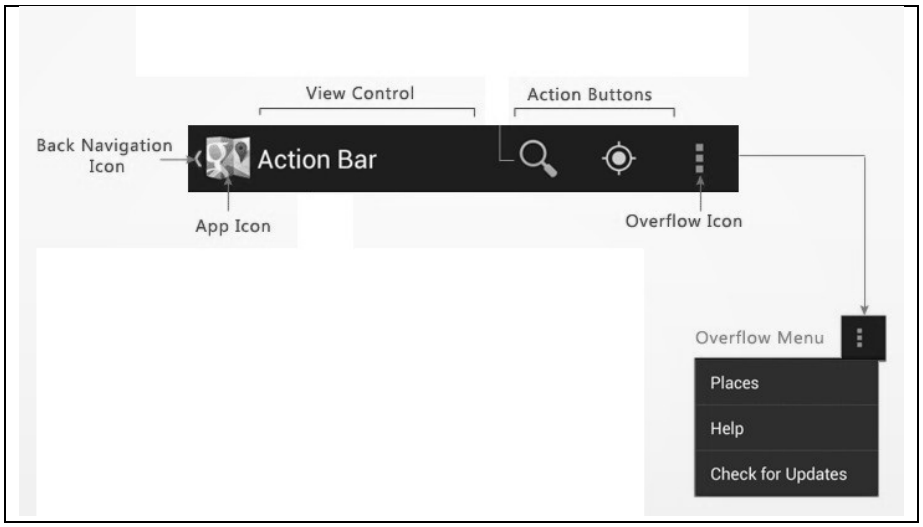
```
    case R.id.vibrate:
    case R.id.dont_vibrate:
        if (item.isChecked())
          item.setChecked(false);
        else item.setChecked(true);
          return true;
    default:
          return super.onOptionsItemSelected(item);
   }
}
```

2. ACTION BAR

2.1. Giới thiệu

Trong lần đầu giới thiệu chiếc Samsung Galaxy Nexus, Google đã loại bỏ nút Menu ra khỏi cụm phím điều hướng chính của hệ thống, chỉ còn lại nút Quay về (Back), nút trở về màn hình chính (Home) và nút liệt kê các ứng dụng đã chạy gần đây (Recents Apps). Nó biến thành một thứ gọi là "Action Bar", nơi mà người dùng có thể khởi chạy các hoạt động tùy thuộc vào ứng dụng.

- Action Bar có chức năng:
  - Điều hướng giao diện ứng dụng.
  - Hiển thị các thao tác trên toàn bộ hệ thống ứng dụng hoặc thao tác tại màn hình hiển thị (tìm kiếm, chỉnh sửa, xóa…).
  - Thao tác chuyển đến các giao diện khác nhau trong cùng một màn hình hiển thị (tabhost, list navigation…).
- Những tính năng nào không thể hiện hết thì sẽ nằm trong dấu ba chấm dạng dọc ở cuối Action Bar, gọi là Action Overflow.
- ActionBar có từ phiên bản Android 3.0 (API level 11).
- Các tổ hợp điều khiển trên ActionBar:
  - App Icon: thường là logo của ứng dụng hoặc bất kỳ Icon nào mà bạn muốn.
  - View Control: thường hiển thị Title cho nội dung màn hình hiện tại, có thể tùy chỉnh thành một Spinner.
  - Action Button: chứa một số Button để thực hiện chức năng quan trọng.
  - Action Overflow: Chứa một Menu Dropdown các chức năng hiển thị.



Hình 1.6. ActionBar.

2.2. Tạo ActionBar

- Tạo một tập tin menu cho ActionBar trong thư mục res/menu:

```
<?xml version="1.0" encoding="utf-8"?>
<menu   xmlns:android="http://schemas.android.com/apk/res/android"
>
   <item
        android:id="@+id/action_search"
        android:actionViewClass="android.widget.SearchView"
        android:icon="@drawable/ic_action_search"
        android:showAsAction="ifRoom"
        android:title="@string/action_search"/>
    <item
                android:id="@+id/action_location_found"
                 android:icon="@drawable/ic_action_location_found"
                android:showAsAction="ifRoom"
                  android:title="@string/action_location_found"/>
 <item
        android:id="@+id/action_refresh"
        android:icon="@drawable/ic_action_refresh"
        android:showAsAction="ifRoom"
        android:title="@string/action_refresh"/>
    <item
        android:id="@+id/action_help"
        android:icon="@drawable/ic_action_help"
        android:showAsAction="ifRoom"
        android:title="@string/action_help"/>
   <item
```

```
        android:id="@+id/action_check_updates"
        android:icon="@drawable/ic_action_refresh"
        android:showAsAction="ifRoom"
        android:title="@string/action_check_updates"/>
</menu>
```

- Đối với mỗi thẻ item sẽ có một số thuộc tính quan trọng sau:
  - android:id - id của Action Button.
  - android:icon - icon hiển thị cho chức năng.
  - android:title - tiêu đề cho chức năng.
  - android:showAsAction - Xác định việc hiển thị cho Action Button.

- Tiếp tục, mở MainActivity ra và thực hiện kéo tập tin Menu vừa tạo vào cho ứng dụng. Chúng ta sẽ thực hiện việc này trong phương thức onCreateOptionsMenu():

```
public booleanonCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar
if it is
  present.
        getMenuInflater().inflate(R.menu.action_bar_menu, menu);
        return true;
}
```

- Khai báo sử dụng ActionBar trong Java Code:

```
// android.support.v7.app.ActionBar
ActionBar actionBar = getSupportActionBar();
// android.app.ActionBar
ActionBar actionBar = getActionBar();
```

2.3. Thao tác với ActionBar

- Chúng ta sẽ kích hoạt các chức năng khi chạm vào các Action Button. Việc này sẽ được thực hiện trong onOptionsItemSelected():

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
    case R.id.action_search:
       return true;
```

```
    case R.id.action_location_found:
       LocationFound();
       return true;
    case R.id.action_help:
       return true;
    case R.id.action_refresh:
       return true;
    case R.id.action_check_updates:
       return true;
    default:
    return true;
    }
}
```

- Tiếp theo, chúng ta tìm hiểu việc hiển thị và ẩn thành Action Bar. Có 2 phương thức rất đơn giản, đó là hide() và show()

```
actionBar.show();
actionBar.hide();
```

- *Lưu ý:* ứng dụng sử dụng việc hiển thị - giấu ActionBar liên tục có thể thay thế bằng chế độ Overlay trên Android 4.3 và 4.4.
- Thay đổi App Icon cho Action Bar. Chúng ta sử dụng phương thức setIcon() để thay đổi App Icon trên Action Bar

```
actionBar.setIcon(R.drawable.ico_actionbar);
```

- **SplitActionBar**: cơ chế cho phép hiển thị ActionBar ở dạng chia đôi khi có nhiều item.
    - Sử dụng:
        - Khai báo thuộc tính uiOptions="splitActionBarWhenNarrow“ trong thẻ <activity>.
    - Ví dụ khai báo cho SplitActionBar cho các phiên bản cũ hơn: Cần khai báo thêm thẻ <meta-data>

```
<manifest ...>
    <activity uiOptions="splitActionBarWhenNarrow" ... >
```

```
        <meta-data android:name="android.support.UI_OPTIONS"
                   android:value="splitActionBarWhenNarrow" />
    </activity>
</manifest>
```

- **Navigation Up Icon**: chế độ hiển thị và tương tác với biểu tượng ActionBar cho phép người dùng có thể quay về màn hình trước đó ứng dụng (tùy thuộc ngữ cảnh).
    - Sử dụng: khai báo thông qua phương thức setDisplayHomeAsUpEnable()
    - Ví dụ:

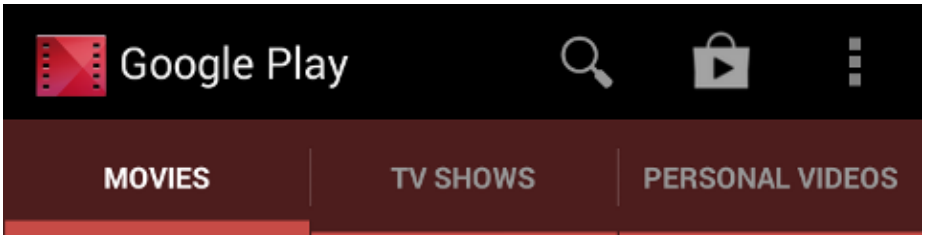
```
ActionBar actionBar = getSupportActionBar();
actionBar.setDisplayHomeAsUpEnabled(true);
```

    - Navigation Up Icon:
        - Khai báo sử dụng trong AndroidManifest.xml
        - Ví dụ:

```
<activity
        android:name="com.htsi.t3h.SecondActivity"
        android:label="@string/title_activity_display_message"

        android:parentActivityName="com.htsi.t3h.MainActivity">

        <!- Hỗ trợ khai báo cho các phiên bản cũ hơn-->
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="com.htsi.t3h.MainActivity" />

 </activity>
```

3. CHẾ ĐỘ ĐIỀU HƯỚNG

- NavigationMode: thực hiện khai báo chế độ điều hướng thông qua phương thức setNavigationMode.
- NavigationMode cho phép hiển thị dữ liệu màn hình ở hai chế độ:
    - Tab:
        - Dữ liệu màn hình bao gồm nhiều trang, mỗi trang được điều hướng hiển thị theo tab của ActionBar.



Hình 1.7. NavigationMode - Tab.

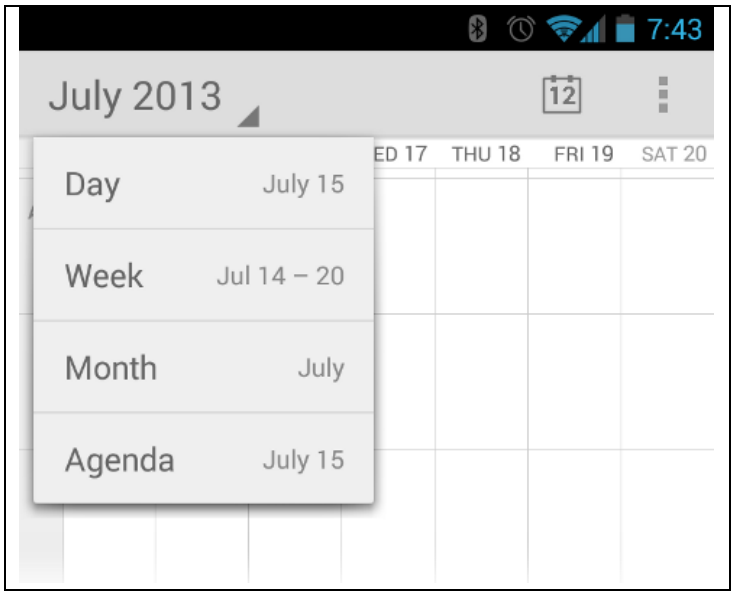
- Khai báo:

```
actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
```

- Xây dựng Tab:
  - Gọi phương thức newTab(): đối tượng trả về - Tab.
  - Thiết lập bộ lắng nghe – setTabListener.
  - Biểu tượng – setIcon..
  - Tiêu đề cho từng Tab – setText.
  - Gắn Tab vào ActionBar thông qua phương thức addTab, tham số truyền vào là một đối tượng Tab.
- Ví dụ khai báo sử dụng:

```
Tab tab = actionBar.newTab()
                        .setText(R.string.artist)
                        .setTabListener(new
TabListener<ArtistFragment>(
                                this,                       "artist",
ArtistFragment.class));
actionBar.addTab(tab);

tab = actionBar.newTab()
                    .setText(R.string.album)
                    .setTabListener(new
TabListener<AlbumFragment>(
                              this, "album", AlbumFragment.class));
actionBar.addTab(tab);
```

- List:
  - Dữ liệu màn hình có thể bao gồm nhiều trang hoặc chế độ hiển thị, mỗi chế độ được chọn lựa từ Spinner thiết lập trên ActionBar.



Hình 1.8. NavigationMode - List.

- Khai báo:

```
actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_LIST);
```

- Xây dựng List:
  - Tạo đối tượng SpinnerAdapter.
  - Khai báo bộ lắng nghe sự kiện trên List – OnNavigationListener.
  - Gọi phương thức setListNavigationCallbacks, tham số truyền vào là một adapter và bộ lắng nghe sự kiện.
- Ví dụ khai báo sử dụng :

```
// Khai báo Adapter
SpinnerAdapter mSpinnerAdapter = ArrayAdapter.createFromResource
              (this, R.array.action_list,


  android.R.layout.simple_spinner_dropdown_item);
// OnNavigationListner
OnNavigationListner            mNavigationCallback=            new
OnNavigationListener() {…}
// Thiết lập Adapter và lắng nghe sự kiện
actionBar.setListNavigationCallbacks(mSpinnerAdapter,
```

```
mNavigationCallback);
```

4. TOOLBAR

4.1. Giới thiệu

Từ phiên bản API 21 (Android 5.0 Lollipop), Android hỗ trợ Toolbar để thay thế ActionBar. Toolbar được xem là bản cải tiến của ActionBar và được kết thừa từ ViewGroup, vẫn có thể dùng một số thuộc tính hay phương thức của ActionBar như thêm Logo, Labels, Navigation items… và có bổ sung thêm các phương thức để giúp thay đổi màu, tùy chỉnh kích thước hay có thể thiết lập bất kỳ vị trí hiển thị trên giao diện… Có thể thiết lập hiển thị Toolbar cho Activity nào đó bằng phương thức "setActionBar()".

Toolbar cùng lúc có thể chứa các yếu tố sau:

- *A Navigation Button.* Nó có thể là Up arrow, navigation menu toggle, close, collapse, done hay một số hình ảnh mà bạn tự thiết kế.
- *A branded logo image.* Có thể thiết lập chiều cao cho logo, thiết lập khoảng cách tùy ý.
- *A title and subtitle.* Thiết lập tiêu đề cho Toolbar, nhưng có thể nó sẽ bị che lấp khi ta thiết lập logo quá lớn.
- *One or more custom views.* Có thể thiết lập một hoặc nhiều View do chúng ta tùy chình. Để thiết lập vị trí hiển thị chúng ta cần thông qua lớp "Toolbar.LayoutParams", chẳng hạn như canh giữa theo chiều ngang thì chúng ta dùng biến cờ CENTER_HORIZONTAL thông qua Gravity, vị trí được canh giữa trên không gian trống còn lại (tức là nếu chúng ta đã thiết lập logo cho Toolbar rồi thì không đối tượng View mà ta thêm vào thì sẽ nằm ở vị trí trống còn lại và được canh giữa theo chiều ngang).
- *An action menu.* Cũng giống như thanh ActionBar, "menu" sẽ mằm ở vị trí cuối của Toolbar.

4.2. Tạo ToolBar

- *Xóa ActionBar cũ.*
  Chúng ta cần xóa ActionBar cũ ra khỏi theme. Mở thư mục *res -> value -> styes.xml* và viết lại như bên dưới. Chúng ta chọn theme **Theme.AppCompat.Light.NoActionBar** để không hiển thị ActionBar.

```
<resources>
    <style name="AppTheme"
                    parent="Theme.AppCompat.Light.NoActionBar">
    </style>
</resources>
```

Đến file xml để thêm Toolbar cho giao diện. Chúng ta nên thêm vào giao diện khi chúng ta cần. Vì Toolbar là một ViewGroup nên có thể thêm ở các ViewGroup và có thể thiết lập các thuộc tính ID, color, width và height. Chúng ta có thể thêm thuộc tính margins (left và right) bằng cách dùng **contentInsetEnd** và **contentInsetStart**.

```
<Android.support.v7.widget.Toolbar

xmlns:Android="http://schemas.Android.com/apk/res/Android"
        xmlns:app="http://schemas.Android.com/apk/res-auto"
        Android:id="@+id/toolbar"
        Android:layout_width="match_parent"
        Android:layout_height="wrap_content"
        Android:background="#ff696969"
        app:contentInsetEnd="0dp"
        app:contentInsetStart="0dp"
        ></Android.support.v7.widget.Toolbar>
```

- *Thiết lập cho lớp Activity.*
  Chúng ta cần kế thừa lớp **ActionBarActivity** cho Activity. Và dùng thư viện **Theme.AppCompat.Light.NoActionBar** và **Android.support.v7.widget.Toolbar**.

```
import android.support.v7.widget.Toolbar;
import android.support.v7.app.ActionBarActivity;
public class MainActivity extends ActionBarActivity{
}
```

- *Truy xuất và sử dụng Toolbar.*
  Lấy Toolbar ở file xml đã thiết lập bằng phương thức **findViewById()**. Thiết lập Toolbar giống như một ActionBar bằng phương thức **setSupportActionBar()**.

```
Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
setSupportActionBar(toolbar);
```

4.3. Thao tác với Toolbar

- *Thiết lập menu cho Toolbar.*
  Tạo menu thông qua phương thức "inflateMenu()", ví dụ: toolbar.inflate(R.menu. toolbar_menu), ta sẽ thiết kế toolbar_menu để truyền vào phương thức này như sau:

```
    <menu
xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        tools:context="com.tqky.toolbar.MainActivity" >
```

```
        <item android:id="@+id/action_search"
            android:title="@string/action_search"
            android:icon="@drawable/ic_search"
            android:showAsAction="ifRoom" />

        <item android:id="@+id/action_share"
            android:title="@string/action_share"
            android:icon="@drawable/ic_share"
            android:showAsAction="ifRoom" />


        <item android:id="@+id/action_settings"
            android:title="@string/action_settings"
            android:orderInCategory="100"
            android:showAsAction="never" />
    </menu>
```

Bắt lại sự kiện khi người dùng tương tác lên các item của menu.

```
toolbar.setOnMenuItemClickListener(new
Toolbar.OnMenuItemClickListener() {
            @Override
            public boolean onMenuItemClick(MenuItem menuItem) {

                switch (menuItem.getItemId()){
                    case R.id.action_share:
                        Toast.makeText(MainActivity.this,
"Share",
Toast.LENGTH_SHORT).show();
                        return true;
                }

                return false;
            }
        });
```

- *Thiết lập Navigation Icon.*
  Thiết lập icon thông qua phương thức "setNavigationIcon()". Bắt sự kiện khi nhấn vào icon thông qua phương thức "setNavigationOnClickListener()".

```
toolbar.setNavigationIcon(R.drawable.ic_launcher);
        toolbar.setNavigationOnClickListener(new
```

```
View.OnClickListener() {
            @Override
            public void onClick(View view) {
Toast.makeText(MainActivity.this,"Navigation",Toast.LENGTH_SHORT)
.show();
            }
        });
```

Bài 4
ACTION PROVIDER VÀ ĐIỀU KHIỂN TÌM KIẾM

1. ACTION PROVIDER

- ActionProvider (API Level 14) là một dạng menu được đính kèm để chúng ta có thể thao tác nhanh ngay trên ActionBar. Trong trường hợp cụ thể, chúng ta sẽ nghiên cứu về ShareActionProvider – một dạng menu chia sẻ rất tiện lợi.
- ShareActionProvider có từ API level 14.

Hình 1.1. Chia sẻ hình với ShareActionProvider.

- Để tạo ra một ShareActionProvider chúng ta chỉ việc khai báo thuộc tính android:actionProviderClass trong cặp thẻ <item> của file tài nguyên menu.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
   <item
            android:id="@+id/menu_item_share"
            android:showAsAction="ifRoom"
            android:title="Share"
```

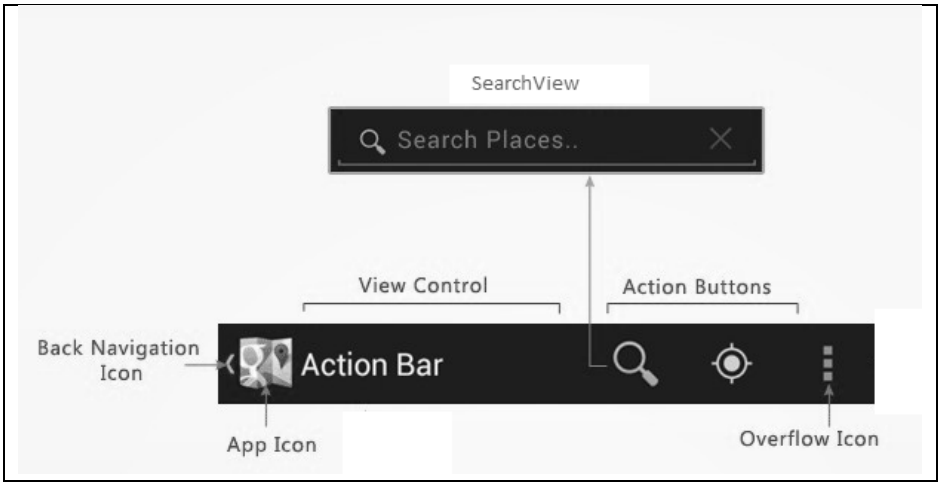
```
            android:actionProviderClass=
                "android.widget.ShareActionProvider" />
</menu>
```

- Xử lý sự kiện với ShareActionProvider:

```
private ShareActionProvider mShareActionProvider;
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate menu resource file.
        getMenuInflater().inflate(R.menu.share_menu, menu);

        // Locate MenuItem with ShareActionProvider
        MenuItem item = menu.findItem(R.id.menu_item_share);

        // Fetch and store ShareActionProvider
        mShareActionProvider =

(ShareActionProvider)item.getActionProvider();

         // Return true to display menu
         return true;
    }
    // Call to update the share intent
    private void setShareIntent(Intent shareIntent) {
    if (mShareActionProvider != null) {
            mShareActionProvider.setShareIntent(shareIntent);
     }
  }
```

2. ĐIỀU KHIỂN TÌM KIẾM

- ActionView là dạng điều khiển hiển thị trên ActionBar cho phép thực hiện các chuỗi thao tác trên cùng một điều khiển để thay đổi dữ liệu.
- Khai báo sử dụng thông qua hai thuộc tính:
  - actionLayout.
  - actionViewClass.
- SearchView: Trong ứng dụng, khi thực hiện thao tác với khối dữ liệu lớn, hoặc muốn truy xuất ngay tức thời một tập tin nào đó, người dùng thường hay bối rối và mất nhiều thời gian để tìm ra chúng. Ở bài này chúng ta tích hợp điều khiển tìm kiếm SearchView như một thành phần của ActionBar.



Hình 1.2. SearchView.

Ví dụ sau đây sẽ hướng dẫn chi tiết về ứng dụng của điều khiển tìm kiếm này.

- Thực hiện kéo ListView vào tập tin activity_main.xml:

```
<RelativeLayout
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"
    tools:context=".MainActivity" >
    <ListView
        android:id="@+id/listCountries"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true" >
```

```
    </ListView>
</RelativeLayout>
```

- Sau đó trong thư mục res/menu tạo tập tin search_view.xml thực hiện khai báo như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<menu   xmlns:android="http://schemas.android.com/apk/res/android"
>
    <item android:id="@+id/action_search"
          android:title="@string/search"
          android:icon="@android:drawable/ic_menu_search"
          android:showAsAction="always"
          android:actionViewClass="android.widget.SearchView" />
</menu>
```

- Cho lớp MainActivity thực thi giao diện onQueryTextListener và tiến hành khai báo các biến cần thiết.

```
public class MainActivity extends Activity implements
OnQueryTextListener{
  private SearchView mSearchView;
  private ListView mListView;
  String[]  COUNTRIES = {"Belgium","Canada","France",
                         "Germany","Italy","Vietnam",
                         "Croatia", "Japan", "Finland",
                          "Venezuela", "Greek", "India"};
  ArrayAdapter<String> adapter;
```

- Trong hàm onCreate thực hiện khởi tạo và tham chiếu giá trị cho các biến đã khai báo ở trên:

```
 @Override
    protected void onCreate(Bundle savedInstanceState) {
           super.onCreate(savedInstanceState);
           setContentView(R.layout.activity_main);
           adapter = new ArrayAdapter<String>(this,
           android.R.layout.simple_list_item_1, COUNTRIES);
            mListView = (ListView)
findViewById(R.id.listCountries);
           mListView.setAdapter(adapter);
}
```

- Override phương thức onCreateOptionMenu để thực hiện gọi SearchView từ thanh ActionBar và gán sự kiện.

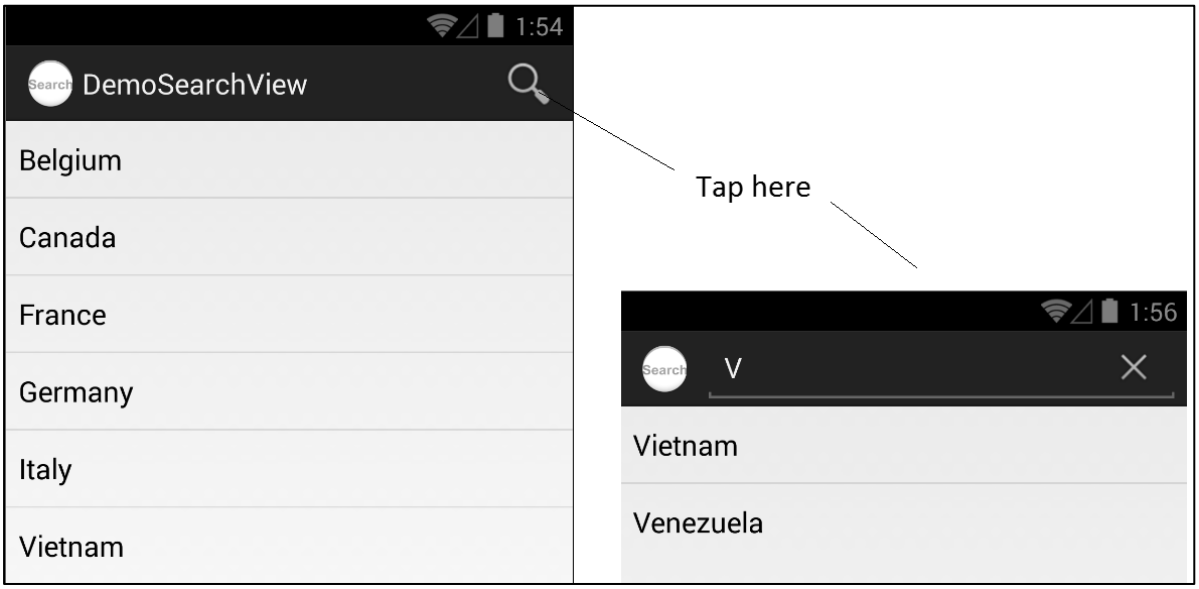
```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
  getMenuInflater().inflate(R.menu.search_view, menu);
        MenuItem itemSearch  =
menu.findItem(R.id.action_search);
        mSearchView = (SearchView) itemSearch.getActionView();
        mSearchView.setOnQueryTextListener(this);
      return true;
}
```

- Cuối cùng thực thi hai phương thức truy vấn dữ liệu khi người dùng nhập váo khung SeachView, ở đây đơn giản ta gọi phương thức filter để Adapter tự sắp xếp dữ liệu.

```
@Override
public boolean onQueryTextChange(String newText) {
  //Instant Search
  if (TextUtils.isEmpty(newText)) {
```

```
        adapter.getFilter().filter("");
    mListView.clearTextFilter();
  }else{
        adapter.getFilter().filter(newText.toString());
  }
  return true;
}
```

- Thực hiện chạy ứng dụng: bấm vào biểu tượng tìm kiếm, một EditText sẽ xuất hiện, tiến hành nhập liệu. Và ListView sẽ tự động tìm kiếm và sắp xếp dữ liệu cho chúng ta.

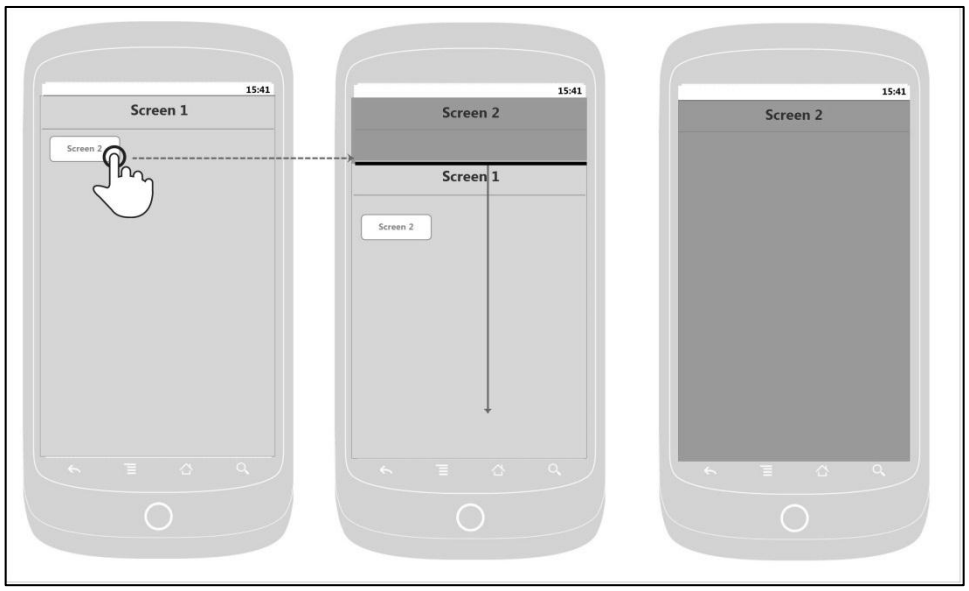


Hình 1.3. Hiển thị kết quả tìm kiếm với SearchView.

Bài 5
CÁC CHUYỂN HOẠT TRONG ỨNG DỤNG

1. PROPERTY ANIMATION

- Property Animation là dạng chuyển hoạt hỗ trợ chúng ta trong việc lập trình các chuyển hoạt cho các thuộc tính của một đối tượng nào đó bằng các thay đổi 2 giá trị 0 và 1. Ví dụ, khi ta có thể thay đổi màu sắc, độ trong suốt để làm hiệu ứng xuất hiện và biến mất của đối tượng, hoặc thay đổi các kích thước dòng chữ chẳng hạn.
- Được giới thiệu từ phiên bản Android 3.0 (API 11) có thể nói đây là bộ công cụ mạnh mẽ trong việc lập trình chuyển hoạt bởi vì chúng ta có thể ứng dụng nó vào bất kỳ thuộc tính nào chúng ta muốn.
- Mỗi chuyển hoạt sẽ được chứa trong một tập tin XML riêng biệt trong thư mục res/animator. Giống như Layout và Drawable, tên của tập tin sẽ được sử dụng làm định danh cho chuyển hoạt đó. Việc định nghĩa các chuyển hoạt chỉ đơn giản là gán một giá trị cho một thuộc tính nào đó, sau đó cấp phát cho đối tượng thực hiện chuyển hoạt đó.
-



Hình 1.1. Ví dụ mô tả Property Animation.

Ví dụ sau đây mô tả việc thay đổi thuộc tính độ trong suốt của một đối tượng bằng cách thiết lập thuộc tính alpha, tăng dần giá trị từ 0 đến 1 trong khoảng thời gian là 10 giây.

- Tạo file alpha_anim.xml trong thư mục res/animator:

```
<?xml version="1.0" encoding="utf-8"?>
<objectAnimator
xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="10000"
    android:propertyName="alpha"
    android:valueFrom="0.0"
    android:valueTo="1.0" >
</objectAnimator>
```

- Tham chiếu và sử dụng trong Java code:

```
TextView txtHello = (TextView) findViewById(R.id.txtHello);
ObjectAnimator animOb = (ObjectAnimator)
AnimatorInflater.loadAnimator(getApplicationContext(),

  R.animator.alpha_anim);
animOb.setTarget(txtHello);
animOb.start();
```

2. VIEW ANIMATION

2.1. Tween Animation

- Mỗi chuyển hoạt được chứa trong tập tin XML trong thư mục res/anim với các hiệu ứng sau: alpha (ẩn hiện mờ), scale (cân chỉnh tỉ lệ), translate (chuyển động) và rotate (quay góc). Mỗi hiệu ứng bao gồm các cặp thuộc tính và giá trị như bảng sau:

Hiệu ứng	Thuộc tính	Giá trị
Alpha	fromAlpha/toAlpha	Từ 0 đến 1 (số thực)
Scale	fromXScale/toXScale	Từ 0 đến 1 (số thực)
	fromYScale/toYScale	Từ 0 đến 1 (số thực)

	pivotX/pivotY	Tỉ lệ chiều ngang và chiều dọc từ 0% đến 100%
Translate	fromX/toX	Từ 0 đến 1 (số thực)
	fromY/toY	Từ 0 đến 1 (số thực)
Rotate	fromDegrees/toDegrees	Từ 0 đến 360 (số thực)
	pivotX/pivotY	Tỉ lệ chiều ngang và chiều dọc từ 0% đến 100%

- Chúng ta có thể sử dụng cặp thẻ set để thiết lập nhiều hiệu ứng với nhau. Khi đó, các hiệu ứng sẽ được bổ sung các một số thuộc tính để chỉnh sửa thời điểm và cách thức mỗi hiệu ứng sẽ thực thi. Danh sách các thuộc tính như sau:
  - *duration* – thời gian thực thi tất cả hiệu ứng, tính bằng mili giây.
  - *startOffset* – độ trễ trước khi hiệu ứng bắt đầu.
  - *fillBeforetrue* – cho phép biến đổi các hiệu ứng trước khi thực thi chuyển hoạt.
  - *fillAftertrue* – cho phép biến đổi các hiệu ứng sau khi kết thúc chuyển hoạt.
  - *interpolator* – thiết lập tốc độ của hiệu ứng thay đổi theo thời gian, để sử dụng thuộc tính này ta sẽ tham chiếu đến tài nguyên chuyển hoạt của hệ thống android:anim/interpolatorName.
- Nếu không thiết lập thuộc tính startOffset, tất cả hiệu ứng sẽ cùng thực thi đồng bộ với nhau.

Ví dụ sau sẽ mô tả chuyển hoạt quay đối tượng một góc 360 độ, rung và mờ dần:

- Tạo file shrink_anim.xml trong thư mục res/anim:

```
<?xml version="1.0" encoding="utf-8"?>
<set android:interpolator="@android:anim/accelerate_interpolator"
 xmlns:android="http://schemas.android.com/apk/res/android">

    <rotate
        android:fromDegrees="0"
        android:toDegrees="360"
        android:pivotX="50%"
        android:pivotY="0%"
        android:startOffset="500"
        android:duration="1000"/>

    <scale
        android:fromXScale="1.0"
        android:toXScale="0.0"
```

```
        android:fromYScale="1.0"
        android:toYScale="0.0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:startOffset="500"
        android:duration="500" />

    <alpha
        android:fromAlpha="1.0"
        android:toAlpha="0.0"
        android:startOffset="500"
        android:duration="500" />
</set>
```

- Tham chiếu và sử dụng trong Java Code:

```
AnimationSet anim_shrink = (AnimationSet)
AnimationUtils.loadAnimation(getApplicationContext(),
R.anim.shrink_anim);
txtHello.startAnimation(anim_shrink);
```

2.2. Frame Animation

- Frame animation là sự kiện diễn ra một chuỗi các chuyển hoạt được lồng ghép bởi nhiều hình ảnh khác nhau. Mỗi hình ảnh sẽ được thiết lập một thời gian để thực hiện. Do các chuyển hoạt liên quan đến tài nguyên dạng Drawable nên các tập tin Frame-by-Frame Animation sẽ được lưu trong thư mục tài nguyên res/drawable và tên tập tin sẽ được dùng như định dạng để tham chiếu sử dụng.
- Có thể tạo frame animation trong thư mục res/drawable vì frame animation cũng được xem như một dạng hình ảnh.

Ví dụ sau minh họa về chuyển hoạt đơn giản của các hình ảnh, mỗi hình ảnh sẽ được thể hiện trong một phần tư giây.

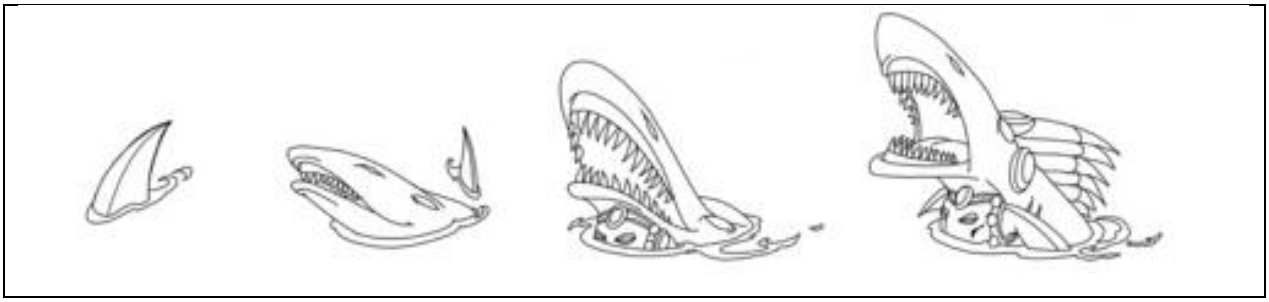
- Tạo file fizz_utl_animation.xml trong thư mục res/drawable:

```
<animation-list
xmlns:android="http://schemas.android.com/apk/res/android" >
  <item                          android:drawable="@drawable/shark1"
android:duration="250"></item>
```

```
 <item                          android:drawable="@drawable/shark2"
android:duration="250"></item>
 <item                          android:drawable="@drawable/shark3"
android:duration="250"></item>
 <item                          android:drawable="@drawable/shark4"
android:duration="250"></item>
 <item                          android:drawable="@drawable/shark5"
android:duration="250"></item>
 <item                          android:drawable="@drawable/shark6"
android:duration="250"></item>
 <item                          android:drawable="@drawable/shark7"
android:duration="250"></item>
 <item                          android:drawable="@drawable/shark8"
android:duration="250"></item>
 <item                          android:drawable="@drawable/shark9"
android:duration="250"></item>
</animation-list>
```

- Tham chiếu và sử dụng trong Java Code:

```
ImageView imgFizz = (ImageView) findViewById(R.id.imgFizz);
imgFizz.setBackgroundResource(R.drawable.fizz_utl_animation);
AnimationDrawable  animFizz         =         (AnimationDrawable)
imgFizz.getBackground();
animFizz.start();
```



Hình 1.2. Ví dụ mô tả Frame Animation.

3. DRAWABLE ANIMATION

- Thư mục tài nguyên: res/drawable/filename.xml.
    - Khai báo các thuộc tính trong XML:

AnimationDrawable - <animation-list>

```
<animation-list
xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot=["true" | "false"] >
    <item
android:drawable="@[package:]drawabledrawable_resource_name"
        android:duration="integer" />
</animation-list>
```

- Ví dụ khai báo Drawable Animation:

exam_drawable_anim.xml:

```
<animation-list
   xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="false”>
    <item android:drawable="@drawable/plane01”
         android:duration="200" />
    <item android:drawable="@drawable/plane02”
         android:duration="200" />
    <item android:drawable="@drawable/plane01”
         android:duration="200" />
</animation-list>
```

- Tham chiếu và sử dụng trong Java code:
    - Thực hiện truy xuất đối tượng AnimationDrawable từ đối tượng View đang hiển thị Drawable, bằng phương thức getBackgroundResource().
    - Gọi phương thức start() từ đối tượng AnimationDrawable.

Ví dụ tham chiếu AnimationDrawable:

```
ImageView planeView = (ImageView) findViewById(R.id.plane);
planeView.setBackgroundResource(R.drawable.exam_drawable_anim);
AnimationDrawable                          plane                          =
(AnimationDrawable)planeView.getBackground();
plane.start();
```

Bài 6
ASYNCTASK - THREAD – HANDLER

1. THREAD VÀ MULTITHREADING

1.1. Thread

Trong lập trình ứng dụng cho di động, một trong những vấn đề quan trọng cần giải quyết đó là tài nguyên hạn hẹp của các thiết bị động. Các thiết bị di động hầu hết có tài nguyên rất hạn chế, ngày nay tuy đã được cải thiện rất nhiều (nâng cấp CPU, GPU, RAM…), nhưng nhìn chung lại, không thế chạy ứng dụng trên các thiết bị di động như trên PC, laptop… Do đó, việc xử lí các thuật toán cần phải tinh giản và thực hiện một cách nhanh nhất có thể để có thể tiết kiệm được chi phí tài nguyên.

Thread là đơn vị nhỏ nhất của tiến trình được định thời bởi hệ điều hành và được bao hàm trong các tiến trình thực thi của máy tính. Mỗi một thread có một callstack cho các phương thức, đối số và biến cục bộ của thread đó.

Android chạy trên một hệ điều hành nhân Linux. Các ứng dụng của Android được viết bằng ngôn ngữ Java, và chúng chạy trong một máy ảo, đó không phải máy ảo Java(JVM) mà là máy ảo Dalvik (Dalvik Virtual Machine). Mỗi một máy ảo Android khi chạy đều có ít nhất một thread chính khi khởi động và có thể còn vài thread khác dùng để quản lý các tiến trình đang chạy. Ứng dụng có thể tự khởi động thêm các thread phụ vào các mục đích cụ thể. Các thread trong cùng một máy ảo tác động qua lại và được đồng bộ hóa bằng cách sử dụng các đối tượng chia sẻ và các monitor liên quan đến các đối tượng đó.

- Mỗi Thread có một độ ưu tiên (số nguyên từ 1 - 10) dùng để xác định lượng thời gian CPU (CPU time) mà thread có thể sử dụng dựa vào phương thức setPriority(int).
- Các phương thức của Thread như sleep(), wait() , notify() và notifyAll()…
- Để tạo ra một thread khác ngoài thread chính trên, chúng ta có 2 cách:
    - Cách 1:
        - Tạo 1 lớp mới kế thừa từ lớp java.lang.Thread và ghi đè lại phương thức run() của lớp này:

```
public class MyThread extends Thread{
  @Override
  public void run() {
     //do something }}
```

- Sử dụng Thread:

```
MyThread t = new MyThread();
t.start();
```

- Cách 2:
  - Tạo 1 lớp và cho lớp này implement Interface Runnable:

```
public class MyRunnable implements Runnable{
 @Override
 public void run() {
    //xử lý
 }
}
```

- Sử dụng thread:

```
MyRunnable b = new MyRunnable();
MyThread t = new MyThread(b);
t.start();
```

1.2. Multithreading

- Multithreading có nghĩa là đa luồng, nhiều thread. Với cơ chế đa luồng, các ứng dụng của bạn có thể thực thi đồng thời nhiều dòng lệnh cùng lúc. Có nghĩa là bạn có thể làm nhiều công việc đồng thời trong cùng một ứng dụng của bạn. Có thể hiểu một cách đơn giản: hệ điều hành với cơ chế đa nhiệm cho phép nhiều ứng dụng chạy cùng lúc thì ứng dụng với cơ chế đa luồng cho phép thực hiện nhiều công việc cùng lúc.
- Ưu điểm của việc sử dụng Multithreading:
  - Các thread chia sẻ tài nguyên của tiến trình nhưng vẫn có thể được thực thi một cách độc lập. Multi-thread sẽ rất hữu dụng khi dùng lập trình đa nhiệm.
  - Các ứng dụng Multi-thread chạy nhanh hơn trên các máy tính hỗ trợ xử lý đa luồng.
- Nhược điểm:
  - Khó khăn trong việc lập trình, việc sử dụng multithreading không hề dễ dàng, đòi hỏi lập trình viên có kinh nghiệm.
  - Khó khăn trong việc quản lí.
  - Có khả năng tiềm tàng xảy ra tình trang deadlock.

1.3. Handler

- Handler là đối tượng cho phép gửi, xử lí các thông điệp và các đối tương Runable có liên quan đến hàng đợi thông điệp. Mỗi Handler có thể liên kết với một thread và hàng đợi thông điệp của thread đó.
- Bạn có thể tạo ra một thread của riêng bạn và giao tiếp ngược với main thread của ứng dụng thông qua một Handler. Điều này được thực hiện bằng cách gọi sendMessage nhưng từ thread mới của bạn.
- Tạo một đối tượng Handler và ghi đè lại phương thức handleMessage để bắt và xử lý các thông điệp liên lạc.

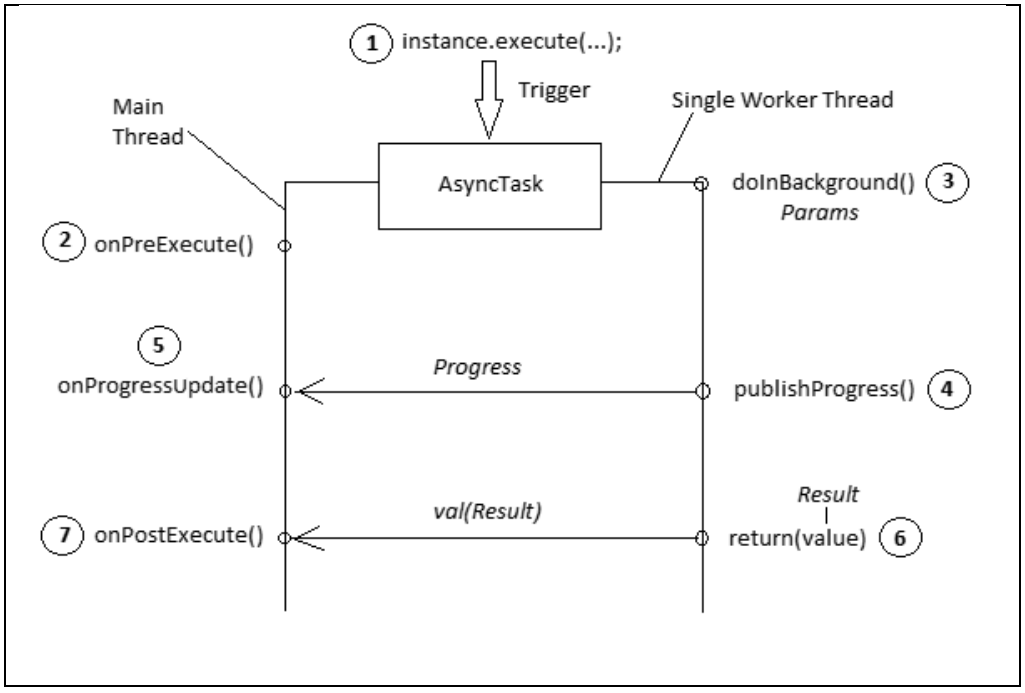
```
Handler handler = new Handler(new Handler.Callback() {
       @Override
       public boolean handleMessage(Message msg) {
    //Kiểm tra message
    if(msg != null){
       //Xử lý message nhận được từ Thread

       }
       return true;
    }
});
```

- Thread sẽ gửi tin nhắn lên Handler và Handler đóng vai trò kiểm tra tin nhắn là gì và làm những việc tương ứng.
- Ta chú ý các phương thức sendMessage sau:
    - sendMessage(): gửi tin nhắn lên Handler ngay lập tức.
    - sendMessageAtFrontOfQueue(): gửi tin nhắn lên Handler và tin nhắn đó sẽ ưu tiên giải quyết trước.
    - sendMessageAtTime(): gửi tin nhắn vào thời điểm chỉ định trước.
    - sendMessageDelayed(): gửi tin nhắn vào Handler sau một khoảng thời gian nhất định.

```
Thread t1 = new Thread(new Runnable() {
        @Override
        public void run() {
                Message mss = new Message();
                mss.arg1 = 1;
                handler.sendMessage(mss);
                 }
 });
 t1.start();
```

2. ASYNCTASK

- AsyncTask là đối tượng cho phép bạn thực hiện hành động xử lý phía background (Thread ngầm) và trả kết quả lên UI thread (Thread chính) mà không cần phải xử lý Thread hoặc Handler.
- Khi ứng dụng của bạn thực hiện một tác vụ dài, chiếm một khoảng thời gian nhất định. Ví dụ như khi click nút Login, ứng dụng phải gửi request lên server để xử lý và trả kết quả về. Nếu làm theo kiểu bình thường thì ứng dụng của bạn sẽ có cảm giác bị treo và chậm, gây ảnh hưởng đến trải nghiệm người dùng. Trong tình huống này, để giải quyết vấn đề chúng ta có thể sử dụng AsyncTask.
- Trong AsyncTask<Params, Progress, Result> có 3 đối số là các Generic Type:
    - Params: Là giá trị (biến) được truyền vào khi gọi thực thi tiến trình và nó sẽ được truyền vào doInBackground.
    - Progress: Là giá trị (biến) dùng để update giao diện diện lúc tiến trình thực thi, biến này sẽ được truyền vào hàm onProgressUpdate.
    - Result: Là biến dùng để lưu trữ kết quả trả về sau khi tiến trình thực hiện xong.
        - Những đối số nào không sử dụng trong quá trình thực thi tiến trình thì ta thay bằng kiểu Void.
- Thông thường trong một AsyncTask sẽ chứa 4 hàm, đó là :
    - onPreExecute(): Tự động được gọi đầu tiên khi tiến trình được kích hoạt.
    - doInBackground(): Được thực thi trong quá trình tiến trình chạy nền, thông qua hàm này để ta gọi hàm onProgressUpdate để cập nhật giao diện (gọi lệnh publishProgress). Ta không thể cập nhật giao diện trong hàm doInBackground().
    - onProgressUpdate(): Dùng để cập nhật giao diện lúc runtime.
    - onPostExecute(): Sau khi tiến trình kết thúc thì hàm này sẽ tự động xảy ra. Ta có thể lấy được kết quả trả về sau khi thực hiện tiến trình kết thúc ở đây.



Hình 1.1. Thứ tự hoạt động của các hàm trong AsyncTask.

Ví dụ:

```
public class MainActivity extends Activity {
  long startMillis;
  TextView txtHello;
           @Override
           protected void onCreate(Bundle savedInstanceState) {
               super.onCreate(savedInstanceState);
               setContentView(R.layout.activity_main);
                         txtHello          =          (TextView)
findViewById(R.id.txtHello);
                         MyAsyncTask     myAsyncTask     =     new
MyAsyncTask();
      myAsyncTask.execute();
            }
```

- Tạo lớp MyAsyncTask kế thừa từ lớp AsyncTask:

```
private class MyAsyncTask extends AsyncTask<String, Integer,
Void>{

        @Override
        protected void onPreExecute() {
    super.onPreExecute();
```

```
    startMillis = System.currentTimeMillis();
    Log.d("Test", "onPreExecute");
        }
         @Override
         protected void onProgressUpdate(Integer... values) {
         super.onProgressUpdate(values);
         Log.d("Test", "Công việc đang làm..." + values[0]);
          }
         @Override
         protected void onPostExecute(Void result) {
            super.onPostExecute(result);
            Log.d("Test", "Thời gian kết thúc: "
                  + (System.currentTimeMillis()-startMillis)/1000
+ " giây");
            Log.d("Test", "Công việc hoàn thành.");
         }
        @Override
        protected Void doInBackground(String... arg0) {
           Log.d("Test", "doInBackground");
           for (int i = 0; i < 3; i++) {
                try {
          Thread.sleep(2000);
          publishProgress(i);
       } catch (InterruptedException e) {
          e.printStackTrace();
       }
            }
            return null;
          }
}
```

- *Lưu ý:*
    - Đối tượng của AsyncTask chỉ được dùng trong UI thread (Thread chính của ứng dụng).
    - Các phương thúc onPostExecute, onPreExecute, onProgressUpdate chỉ là tùy chọn.

Bài 7
SERVICE – BROADCAST RECEIVER VÀ NOTIFICATION

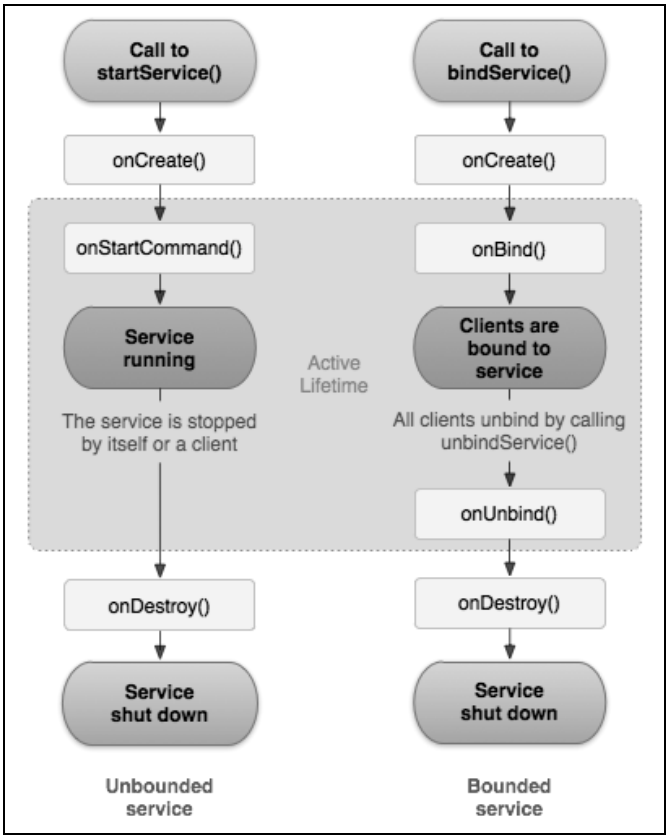
1. SERVICE

1.1. Giới thiệu

- Service được xem như một thành phần chạy ngầm phía bên dưới và hầu như không cần giao diện để tương tác với người dùng, có thể ví dụ một số tác vụ tiêu tốn nhiều thời gian như truy xuất, cập nhật dữ liệu, nhận và xử lý các thông báo người dùng, …
- Service giống như Activity thực hiện theo chu kỳ thời gian để quản lý sự thay đổi của trạng thái.
- Trong AndroidManifest.xml cần phải khai báo:

  <service android:name="com.your-company.MyService" />

  trong cặp thẻ ***<application>***
- Service chạy trên Main Thread nên có thể gây cản trở các tác vụ khác.
- Có 2 loại Service:
  - "**Started**" Service: Service được gọi khi thành phần của ứng dụng gọi nó bằng phương thức startService(). Và service sẽ chạy vô thời hạn. Loại service này thực hiện công việc đơn lẻ như download, tính toán và service sẽ dừng một khi công việc hòan thành.
  - "**Bound**" Service: Service được gọi khi thành phần của ứng dụng gọi nó bằng phương thức bindService(). Loại service đề nghị giao diện Client – Server cho phép thành phần ứng dụng tương tác với nó, gửi yêu cầu, lấy kết quả. Một khi tất cả thành phần ứng dụng ngừng liên kết, thì Service kết thúc .
- Vòng đời:



Hình 1.1. Mô tả vòng đời của Service.

1.2. Tạo Service

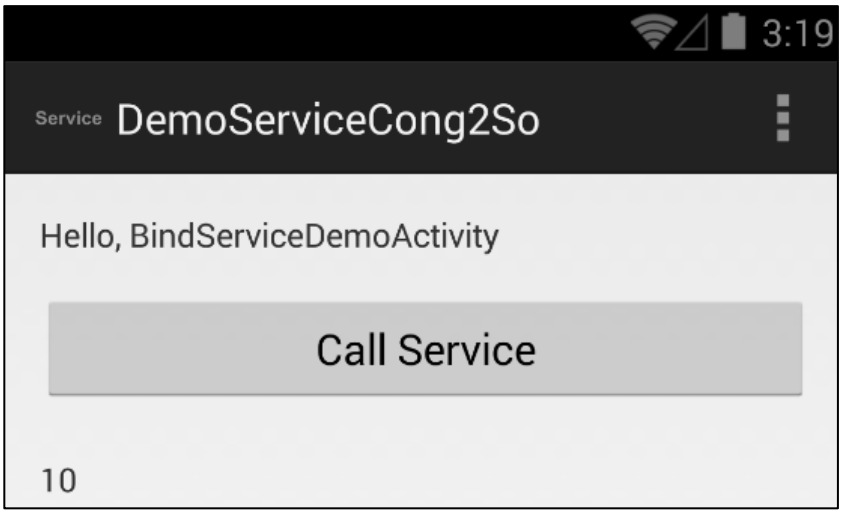
Ví dụ sau sẽ tạo một service có chức năng tính toán và trả về kết quả của phương thức cộng 2 số nguyên.

- Tạo một lớp có tên là MyService mở rộng từ lớp Service như sau:

```
public class MyService extends Service {
         // Binder cho cac client
        private final IBinder mBinder = new LocalBinder();
       @Override
       public IBinder onBind(Intent intent) {
                 return mBinder;
       }
       public class LocalBinder extends Binder {
       // Trả về đối tượng MyService để cho client có thể gọi
                public method MyService getService() {
                          return MyService.this;
```

```
            }
        }
        //Viết phương thức cho các client có thể gọi.
        public int cong2So(int a, int b){
                return a + b ;
        }
}
```

- Trong MainActivity khai báo biến, khởi tạo các đối tượng giao diện:



Hình 1.2. Màn hình ứng dụng gọi Service cộng 2 số.

```
public class MainActivity extends Activity implements
OnClickListener{
  TextView txtResult;
  Button btnCallService;
  MyService mService;
  boolean mBound;
  @Override
   protected void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
        txtResult = (TextView) findViewById(R.id.txtResult);
        btnCallService = (Button)
findViewById(R.id.btnCallService);
        btnCallService.setOnClickListener(this);
   }
}
```

- Tạo một đối tượng ServiceConnection chuyên dùng quản lý việc đóng mở kết nối đến Service. Overide 2 phương thức onServiceDisconnected() và onServiceConnected() sau nhằm bound đến MyService:

```
private ServiceConnection mConnection = new ServiceConnection() {
    @Override
    public void onServiceDisconnected(ComponentName name) {
                mBound = false;
     }
     @Override
     public void onServiceConnected(ComponentName name, IBinder
service) {
                LocalBinder binder = (LocalBinder) service;
                mService = binder.getService();
                mBound = true;
     }
 };
```

- Trong hàm OnStart(): chúng ta dùng phương thức bindService(parameter) để gọi đến MyService thông qua đối tượng Intent:

```
  @Override
  protected void onStart() {
```

```
        super.onStart();
        Intent i = new Intent(this, MyService.class);
         bindService(i, mConnection, Context.BIND_AUTO_CREATE);
  }
```

- Trong hàm onStop(): Ngắt kết nối đến MyService

```
protected void onStop() {
  super.onStop();
      if(mBound){
           mService.unbindService(mConnection);
           mBound = false ;
       }
}
```

- Cuối cùng trong phương thức onClick của button ta gọi đến hàm cong2so(int,int) và hiển thị kết quả trả về ra TextView:

```
public void onClick(View v) {
   switch (v.getId()) {
       case R.id.btnCallService:
                if(mBound){
                      //Cho service cộng 2 số 5 5
                      int result = mService.cong2So(5, 5);
           txtResult.setText(String.valueOf(result));}
        break;
        default: break;
     }
}
```

1.3. IntentService

- IntentService, đây là Service chạy độc lập với Main Thread của bạn, nó sẽ chạy Background và thực hiện nền những gì bạn cần mà không ảnh hưởng đến Main Thread.
- Ví dụ:

```
public class HelloIntentService extends IntentService{
          public HelloIntentService() {
                    super("HelloIntentService");
          }
         @Override
         protected void onHandleIntent(Intent intent) {
                    long  endTime  =  System.currentTimeMillis()  +
5*1000;
                    while (System.currentTimeMillis() < endTime) {
                     synchronized (this) {
                         try {
                                  wait(endTime                    -
System.currentTimeMillis());
                              } catch (Exception e) {}
        }
     }
   }
}
```

2. BROADCAST RECEIVER

2.1. Tạo BroadcastReceiver

- Broadcast Receiver là bộ tiếp nhận và xử lý các Intent dựa trên các chỉ định của chúng ta. Thành phần này sẽ cho phép mở các ứng dụng để xử lý các Intent tương ứng được gửi lên từ hệ thống.
- Trong ứng dụng Android, đôi khi bạn cần phải nhận được tín hiệu nếu như một sự kiện nào trong trên thiết bị được kích hoạt để xử lý cho các ứng dụng tốt hơn. Các sự kiện ví dụ như là pin sắp cạn, máy vừa được khởi động hoặc một ứng dụng vừa được cài đặt

trên máy ..., khi đó, sử dụng Broadcast Receiver sẽ giúp chúng ta bắt được các sử kiện này khi chúng bị kích hoạt.

- Để tạo một BroadcastReceiver, ta sẽ tạo một lớp kế thừa từ lớp BroadcastReceiver.

Ví dụ:

```
public class ConnectivityChangeReceiver extends BroadcastReceiver
{
  @Override
  public void onReceive(Context context, Intent intent) {
      Toast.makeText(context, "Connectivity changed",
Toast.LENGTH_SHORT).show();
       }
}
```

2.2. Đăng ký BroadcastReceiver

- Để đăng ký đối tượng của Broadcast receiver, sử dụng 2 cách:
  - Đăng ký động với phương thức: Context.registerReceiver()
    - Broadcast Receiver có thể được đăng ký trong Activity ở hàm onResume() như sau:

```
@Override
protected void onResume() {
     super.onResume();
     mReceiver = new ConnectivityChangeReceiver();
     registerReceiver(mReceiver,new
     IntentFilter(ConnectivityManager.CONNECTIVITY_ACTION);
}
```

    - Chúng ta cũng nên hủy đăng ký khi Activity trong hàm onPause():

```
@Override
protected void onPause() {
  super.onPause();
  unregisterReceiver(mReceiver);
}
```

    - Lưu ý: nếu tắt ứng dụng BroadcastReceive sẽ không còn lắng nghe.

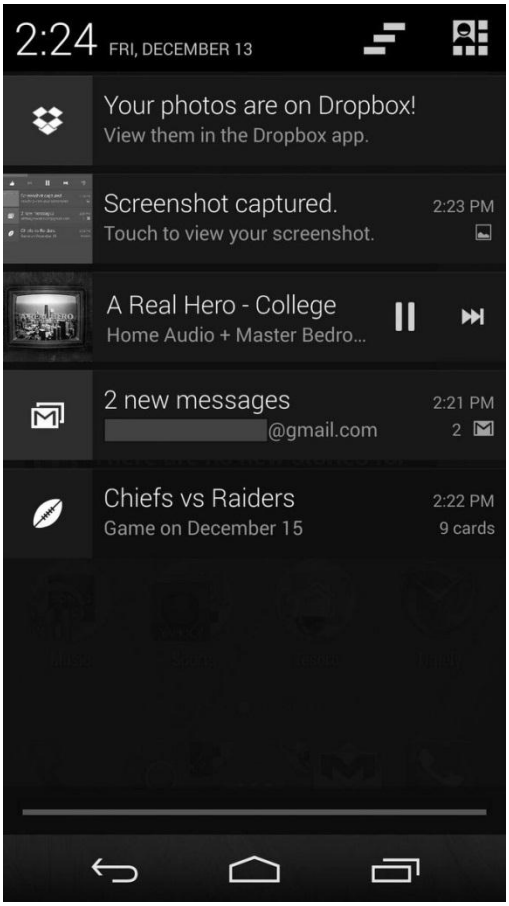
- Đăng ký trong Manifest.xml thông qua cặp thẻ <receiver/>

```
<receiver android:name="com.nghialt.ConnectivityChangeReceiver" >
 <intent-filter>
<action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
 </intent-filter>
 </receiver>
```

- Lưu ý: tự động lắng nghe kể cả khi đã đóng ứng dụng.

3. NOTIFICATION

- Một thông báo (Notification) là thành phần cho phép gửi các thông báo đến người dùng mà không chiếm quá nhiều việc điều khiển của người dùng trên thiết bị. Ví dụ, khi nhận được một tin nhắn hay một thư điện tử, thiết bị sẽ sử dụng Notification để thông báo người dùng thông qua nhạc chuông, đèn nền, thể hiện biểu tượng trên thanh tác vụ, …



Hình 1.3. Các dạng thông báo với Notification.

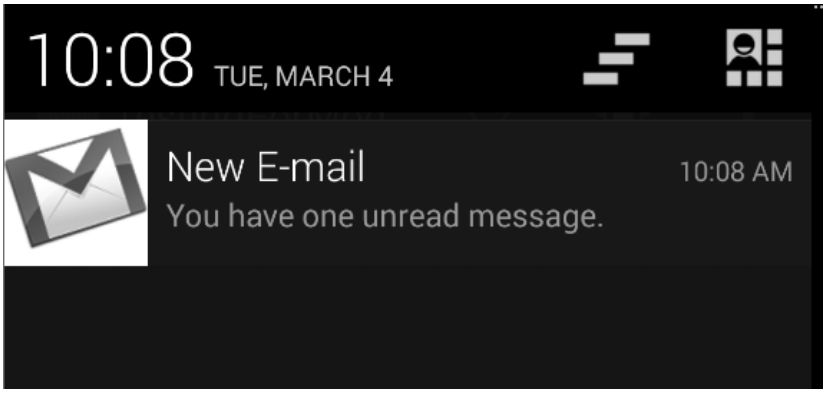
- Thông thường một thông báo được tự động kích hoạt nhằm thông báo tới người dùng là ứng dụng đó đã hoàn thành một công việc nào đó. Bạn có thể kích chọn trực tiếp vào thông báo đó để khởi chạy lại ứng dụng đó khi nó đang nằm trong trạng thái ngủ.
- Ví dụ sau đây sẽ tạo một thông báo với âm thanh và rung thiết bị.

```
public class MainActivity extends Activity {
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
final int NOTIF_ID = 1234;
  int icon = R.drawable.email;
  long when = System.currentTimeMillis();
  String title = "New E-mail";

NotificationManager  notificationManager  =  (NotificationManager)

getSystemService(Context.NOTIFICATION_SERVICE);
           Notification.Builder builder = new Builder(this);

           Intent    notificationIntent    =    new    Intent(this,
MainActivity.class);
           PendingIntent intent = PendingIntent.getActivity(this,

0,notificationIntent, 0);

           buildersetSmallIcon(icon).setContentText("You have one
unread
message.").setWhen(when)
                                  .setContentIntent(intent)
                                  .setContentTitle(title);
```

```
            Notification notification = builder.build();
            // Play default notification sound
           notification.defaults |= Notification.DEFAULT_SOUND;
           // Vibrate
           notification.defaults |= Notification.DEFAULT_VIBRATE;
           notificationManager.notify(NOTIF_ID, notification);
}
```

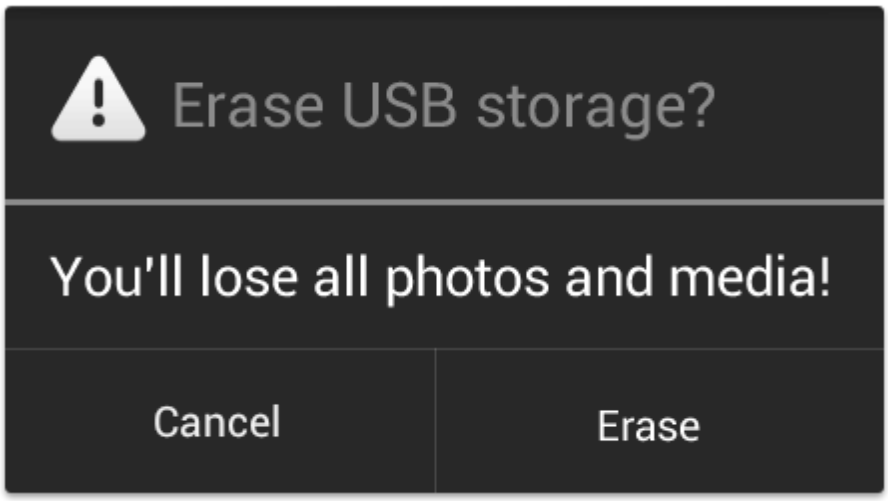
- Xin quyền nếu muốn rung thiết bị:

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



Hình 1.4. Kết quả hiển thị Notification vừa tạo.

- Một số dạng hiển thị thông báo khác:
  - Dialog:
    - Dialog là dạng hộp thoại, một thành phần giao diện khá phổ biến trên các phần mềm desktop, các trang web và cả những ứng dụng di động. Chúng thường được sử dụng để tạo ra các tình huống cần sự xác nhận của người dùng hoặc thể hiện các cảnh báo, lỗi khi người thao tác với ứng dụng.



Hình 1.5. Dialog.

- Có tất cả 3 kiểu hiển thị dialog:
  - Sử dụng trực tiếp lớp Dialog hoặc kế thừa: ngoài lớp AlertDialog để dùng chung Android còn xây dựng một số lớp khác để phục vụ cho từng mục đích riêng biệt. Dialog là lớp được xây dựng và điều khiển hoàn toàn trong Activily nên ta sẽ không cần khai báo trong Manifest khi cần sử dụng.
  - Activity sử dụng giao diện Dialog – chúng ta có thể áp đặt giao diện Dialog cho các Activity thông thường để nó có thể hiện thị như một Dialog chuẩn.
  - Để tạo Dialog chỉ cần tạo mới thực thể thuộc lớp Dialog và thiết lập tiêu đề cũng như giao diện cho Dialog bằng 2 phương thức tương ứng setTitle và setContentView. Sau khi thiết lập ta chỉ cần gọi phương thức show ở nơi mà chúng ta muốn hiện Dialog.

  Ví dụ:

```
Dialog dialog = new Dialog(MainActivity.this);
dialog.setTitle(“Demo Dialog”);
//Thiết lập giao diện cho Dialog bằng layout xml
dialog.setContentView(R.layout_dialog_layout);
dialog.show();
```

- Toast:

- Toast là một loại thông báo (message) ngắn. Hiển thị trong một khoảng thời gian nhất định và tự động mất dần. Bạn có thể sử dụng trong trường hợp hiển thông báo trong các mục thiết lập thông số cấu hình, hay đơn giản chỉ là hiển thị lên để xem thông tin tạm thời nào đó (giống như để kiểm tra một vấn đề xảy ra chẳng hạn).

- Toast có thể được tạo và hiển thị trong Activity hoặc trong Servive.
- Không cho phép người sử dụng tương tác.
- Khi hiển thị sau khoảng thời gian nào đó sẽ tự đóng lại.
- Có 2 giá trị mặc định (ta nên sử dụng 2 giá trị này, không nên gõ con số cụ thể vào): hằng số Toast.LENGTH_SHORT hiển thị trong 2 giây, Toast.LENGTH_LONG hiển thị trong 3.5 giây.

Hình dưới đây cho bạn biết một Toast đang hiển thị:

Hình 1.6. Toast.

- Cú pháp để tạo một thông báo Toast:

```
Toast toast = Toast.makeText(YourActivity.this, “Hiển thị gì thì ghi ở đây”,
    Toast.LENGTH_SHORT);
 toast.show();
```

Mục lục