|  |  |
| --- | --- |
|  | |
| BÀI TẬP | |
|  | |
|  | Lập trình viên trên thiết bị di động  Lập trình di động trên  Android  Module 03:  Lập trình cơ sở dữ liệu, tiến trình và chuyển động trong ứng dụng |

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

|  |  |
| --- | --- |
| **🖰** | Mục tiêu:  Biết cách tạo cơ sở dữ liệu SQLite.  Biết cách lưu trữ, truy vấn và sắp xếp dữ liệu với SQLite. |

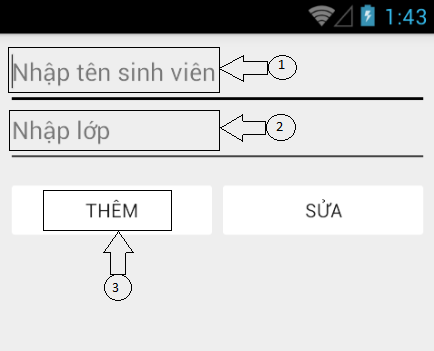
* 1. Tạo một cơ sở dữ liệu SQLite đơn giản

Đề bài:

Tạo một cơ sở dữ liệu SinhVien gồm: Mã sinh viên, Tên sinh viên, Lớp.

* Tạo cơ sở dữ liệu chứa bảng SinhVien.
* Thực hiện thêm, xóa, sửa.
* Lấy tất cả dữ liệu, lấy dữ liệu theo mã sinh viên, sắp xếp theo tên, theo lớp và xây dựng chức năng tìm kiếm sinh viên.

Với giao diện màn hình như sau:



*Hình 1.1*

* Sau khi nhấn thêm (3) thì thêm vào ListView (4) như sau:



*Hình 1.2*

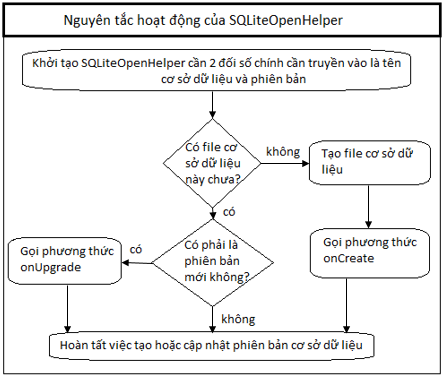
* Khi nhấn vào item (5) thì hiển thị tên và lớp của item này lên 2 EditText tương ứng. Sau đó, bạn có thể chỉnh sửa lại thông tin trên 2 EditText rồi ấn nút sửa (6) để cập nhật lại dữ liệu và hiển thị lên ListView (4).
* Khi nhấn vào ImageView (7) dòng nào thì xóa dòng đó trên ListView.

*Chú ý:* Mọi thay đổi trên ListView (4) thì dữ liệu trên SQLite đều thay đổi theo tương ứng.

Gợi ý thực hiện:

* Tạo lớp kế thừa lớp “SQLiteOpenHelper” để tạo cơ sở dữ liệu, bảng và viết các phương thức truy vấn (các bạn có thể tách các lớp ra để dễ nhìn và nếu có thêm nhiều bảng sẽ không bị rối).

Mô tả nguyên tắc hoạt động của SQLiteOpenHelper:



*Hình 1.3*

* Xin quyền ghi file.

Hướng dẫn chi tiết:

1. Tạo cở sở dữ liệu và bảng trên SQLite.
   1. Tạo lớp “DBHelper” kế thừa “SQLiteOpenHelper” sau đó Override lại các phương thức của lớp này. “SQLiteOpenHelper” là lớp dùng để tạo cơ sở dữ liệu và các bản có trong cơ sở dữ liệu.
   2. Khai báo các biến hằng (tên bảng, cột,...) dùng để tiện cho việc sử dụng khi truy vấn, thêm, xóa, sửa,...

// Tên cơ sở dữ liệu

**private** **static** **final** String *TEN\_DATABASE*= "QuanLySinhVien";

// Tên bảng

**public** **static** **final** String *TEN\_BANG\_SINHVIEN* = "SinhVien";

// Bảng gồm 3 cột \_id, \_ten và \_lop.

**public** **static** **final** String *COT\_ID* = "\_id";

**public** **static** **final** String *COT\_TEN* = "\_ten";

**public** **static** **final** String *COT\_LOP* = "\_lop";

* 1. Viết câu lệnh tạo bảng.

/\*

\* Câu lện tạo bảng.

\* Trong đó: “integer primary key autoincrement”

\* là khóa chính tự tăng và có kiểu dữ liệu là int;

\* “text not null” là không được để trống và kiểu

\* dữ liệu là String.

\*/

**private** **static** **final** String *TAO\_BANG\_SINHVIEN* = ""

+ "create table " + *TEN\_BANG\_SINHVIEN* + " ( "

+ *COT\_ID* + " integer primary key autoincrement ,"

+ *COT\_TEN* + " text not null, "

+ *COT\_LOP* + " text not null );";

* 1. Để tạo cơ sở dữ liệu phải thông qua hàm khởi tạo 4 tham số của lớp “SQLiteOpenHelper” mà chúng ta đã kế thừa. Chúng ta viết trong hàm khởi tạo của lớp “DBHelper” như sau:

**public** DBHelper(Context context) {

**super**(context, *TEN\_DATABASE*, **null**, 1);

}

*Giải thích:* các đối số tương ứng như trong hàm khởi tạo:

* + - context: dùng để mở hoặc tạo cơ sở dữ liệu
    - *TEN\_DATABASE*: tên file cơ sở dữ liệu.
    - **null**: dùng để tạo đối tượng Cursor, hoặc null thì mặc định.
    - 1: phiên bản của cơ sở dữ liệu, nếu là cơ sở dữ liệu cũ thì nó sẽ vào phương thức “onUpgrade” ngược lại thì vào “onDowngrade”.
  1. Dùng cơ sở dữ liệu vừa tạo xong truy xuất đến phương thức “execSQL” để tạo bảng cho cơ sở dữ liệu.

/\*

\* Tạo bảng cho cơ sở dữ liệu đã được tạo ở

\* hàm khởi tạo thông qua lớp SQLiteDatabase

\*/

@Override

**public** **void** onCreate(SQLiteDatabase db) {

db.execSQL(*TAO\_BANG\_SINHVIEN*);

}

­­­­­­­­

1. Tạo đối tượng SinhVien, gồm các thuộc tính như sau:

**private long** \_id;

**private** String \_ten;

**private** String \_lop;

1. Viết các phương thức truy vấn, thêm, xóa, sửa cơ sở dữ liệu.
   1. Tạo lớp “MyDatabase” dùng để hỗ trợ các phương thức tương tác với cơ sở dữ liệu, giảm bớt các đối tượng không cần thiết khi các lớp giao diện muốn truy xuất lấy dữ liệu.
   2. Khai báo 2 lớp SQLiteDatabase và DBHelper:

/\*

\* Lớp “SQLiteDatabase” dùng để hỗ trợ tương

\* tác với cơ sở dữ liệu.

\*/

SQLiteDatabase database;

/\*

\* Còn lớp “DBHelper” là lớp chúng

\* ta vừa tạo, dùng để tạo Cơ sở dữ liệu và bảng.

\*/

DBHelper helper;

* 1. Viết hàm khởi tạo cho lớp “MyDatabase”:

**public** MyDatabase(Context context) {

helper = **new** DBHelper(context);

/\*

\* Phương thức “getWritableDatabase()” dùng

\* để tạo hoặc mở cơ sở dữ liệu để đọc và

\* ghi vào cơ sở dữ liệu.

\*/

database = helper.getWritableDatabase();

}

* 1. Thực hiện lấy tất cả dữ liệu trong cơ sở dữ liệu:

**public** Cursor layTatCaDuLieu() {

// Biến cot là khai báo danh sách các cột cần lấy.

String[] cot = { DBHelper.*COT\_ID*,

DBHelper.*COT\_TEN*,

DBHelper.*COT\_LOP* };

/\*

\* Cursor như là 1 bảng cơ sở dữ liệu được trả ra

\* sau khi truy vấn trong cơ sở dữ liệu.

\*/

Cursor cursor = **null**;

/\*

\* Dùng lớp “SQLiteDatabase” truy xuất đến phương

\* thức “query” để lấy dữ liệu trong cơ sở dữ liệu ra.

\* Ở trong phương thức này là yêu cầu lấy tất cả nên

\* chỉ cần truyền vào các tham số như: tên bảng, các

\* cột cần lấy (cot) và sắp xếp nếu cần.

\*/

cursor = database.query(DBHelper.

*TEN\_BANG\_SINHVIEN*, cot, **null**, **null**, **null**, **null**,

DBHelper.*COT\_ID* + " DESC");

**return** cursor;

}

* 1. Thực hiện thêm vào cơ sở dữ liệu:

**public** **long** them(SinhVien sinhVien) {

/\*

\* ContentValues là đối tượng lưu trữ dữ liệu, và

\* SQLiteDatabase sẽ nhận dữ liệu thông qua đối tượng

\* này để thực hiện các câu lệnh truy vấn.

\*/

ContentValues values = **new** ContentValues();

values.put(DBHelper.*COT\_TEN*,

sinhVien.get\_ten());

values.put(DBHelper.*COT\_LOP*,

sinhVien.get\_lop());

/\*

\* Thêm vào cơ sở dữ liệu cần 2 đối số chính là

\* Tên Bảng và dữ liệu cần thêm.

\*/

**return** database.insert(DBHelper.

*TEN\_BANG\_SINHVIEN*, **null**, values);

}

* 1. Thực hiện xóa một dòng dữ liệu:

**public** **long** xoa(SinhVien sinhVien) {

/\*

\* Tương tự cho xóa một dòng dữ liệu cần

\* 3 đối số là tên bảng, câu điều kiện và

\* cuối cùng là đối số cho câu điều kiện

\* nhưng có thể gộp 2 đối số làm 1 như ở

\* dưới.

\*/

**return** database.delete(DBHelper

.*TEN\_BANG\_SINHVIEN*, DBHelper

.*COT\_TEN* + " = " + "'" +

sinhVien.get\_ten() + "'", **null**);

}

* 1. Thực hiện sửa một dòng dữ liệu:

**public** **long** sua(SinhVien sinhVien) {

ContentValues values = **new** ContentValues();

values.put(DBHelper.*COT\_TEN*,

sinhVien.get\_ten());

values.put(DBHelper.*COT\_LOP*,

sinhVien.get\_lop());

/\*

\* Sửa 1 dòng dữ liệu thì cần 3 đối số chính là

\* tên bảng, dữ liệu cần sửa và câu điều kiện

\*/

**return** database.update(DBHelper

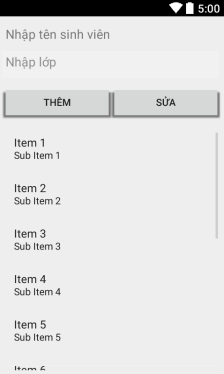
.*TEN\_BANG\_SINHVIEN*, values,

DBHelper.*COT\_ID* + " = "

+ sinhVien.get\_id(), **null**);

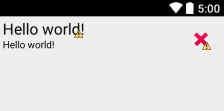
}

1. Thiết kế giao diện “activity\_main.xml” để tương tác với cơ sở dữ liệu:



*Hình 1.4*

1. Thiết kế item “design\_item\_list\_view.xml” cho ListView như sau:



*Hình 1.5*

1. Viết xử lý trong class “MainActivity.java”.
   1. Khai báo một số lớp:

/\*

\* listView dùng để hiển thị cơ sở dữ liệu

\* cho người dùng xem.

\*/

**public** **static** ListView *listView*;

// dữ liệu để hiển thị lên listView

**public** **static** ArrayList<SinhVien> *sinhViens*;

// Lớp để hỗ trợ truy vấn dữ liệu

**public** **static** MyDatabase *database*;

// Ô nhập liệu

**private** EditText editTextTen, editTextLop;

* 1. Khai báo một số biến dùng để lưu trữ giá trị nhập và tương tác từ người dùng:

**private** String ten, lop;

**private** **static** **long** *id* = -1;

* 1. Viết phương thức lấy dữ liệu gán vào ArrayList:

**public** **void** capNhatDuLieu() {

**if** (*sinhViens* == **null**) {

*sinhViens* = **new** ArrayList<SinhVien>();

} **else** {

*sinhViens*.removeAll(*sinhViens*);

}

// Lấy dữ liệu, dùng Cursor nhận lại

Cursor cursor = *database*.layTatCaDuLieu();

**if** (cursor != **null**) {

/\*

\* Di chuyển đến từng dòng dữ liệu

\* thông qua phương thức moveToNext

\*/

**while** (cursor.moveToNext()) {

SinhVien sinhVien = **new** SinhVien();

/\*

\* Mỗi dòng dữ liệu chúng ra sẽ lấy

\* theo cột và gán vào đối tượng

\* SinhVien

\*/

sinhVien.set\_id(Integer.*parseInt*

(cursor.getString(cursor

.getColumnIndex

(DBHelper.*COT\_ID*))));

sinhVien.set\_ten(cursor.getString

(cursor

.getColumnIndex

(DBHelper.*COT\_TEN*)));

sinhVien.set\_lop(cursor.getString

(cursor

.getColumnIndex

(DBHelper.*COT\_LOP*)));

// thêm vào danh sách SinhVien

*sinhViens*.add(sinhVien);

}

}

}

* 1. Thiết lập cho ListView:

// Hiển thị danh sách đã lấy được lên listView

**if** (*sinhViens* != **null**) {

*listView*.setAdapter

(**new** MyAdapter(getApplicationContext()));

}

// Bắt sự kiện khi click lên ListView

*listView*.setOnItemClickListener(**new**

OnItemClickListener() {

@Override

**public** **void** onItemClick(AdapterView<?> parent,

View view, **int** position, **long** id) {

editTextTen.setText

(*sinhViens*.get(position).get\_ten());

editTextLop.setText

(*sinhViens*.get(position).get\_lop());

MainActivity.*id* = id;

}

});

* 1. Viết phương thức lấy dữ liệu khi người dùng nhập:

/\*

\* Lấy dữ liệu từ người dùng nhập

\* gán vào đối tượng SinhVien

\*/

**public** SinhVien layDuLieuNguoiDung() {

ten = editTextTen.getText().toString();

lop = editTextLop.getText().toString();

**if** (ten.trim().length() == 0

|| lop.trim().length() == 0)

**return** **null**;

SinhVien sinhVien = **new** SinhVien();

sinhVien.set\_id(*id*);

sinhVien.set\_ten(ten);

sinhVien.set\_lop(lop);

**return** sinhVien;

}

* 1. Bắt sự kiện và xử lý khi người dùng nhấn yêu cầu thêm dữ liệu:

/\*

\* Thêm dữ liệu vào cơ sở dữ liệu

\*/

**public** **void** them(View view) {

SinhVien sinhVien1 = layDuLieuNguoiDung();

**if** (sinhVien1 != **null**) {

**if** (*database*.them(sinhVien1) != -1) {

*sinhViens*.add(sinhVien1);

capNhatDuLieu();

// Cập nhật lại danh sách

*listView*.invalidateViews();

editTextTen.setText(**null**);

editTextLop.setText(**null**);

*id* = -1;

}

}

}

* 1. Bắt sự kiện và xử lý khi người dùng yêu cầu sửa cơ sở dữ liệu:

/\*

\* Cập nhật cơ sở dữ liệu

\*/

**public** **void** sua(View view) {

SinhVien sinhVien1 = layDuLieuNguoiDung();

**if** (sinhVien1 != **null** && *id* != -1) {

*database*.sua(sinhVien1);

capNhatDuLieu();

// Cập nhật lại danh sách

*listView*.invalidateViews();

editTextTen.setText(**null**);

editTextLop.setText(**null**);

*id* = -1;

}

}

1. Viết adapter cho ListView. Đầu tiên kết thừa lại lớp “BaseAdapter” và Override lại các phương thức của lớp này và viết xử lý như sau:

**public** **class** MyAdapter **extends** BaseAdapter {

/\*

\* LayoutInflater là lớp hỗ trợ lấy tài nguyên layout

\* trong thư mục res

\*/

LayoutInflater inflater;

TextView textView;

Context context;

**public** MyAdapter(Context context) {

/\*

\* Khởi tạo lớp LayoutInflater

\* thông qua Context

\*/

inflater = LayoutInflater.*from*(context);

**this**.context = context;

}

@Override

**public** **int** getCount() {

// Tổng số lượng dòng dữ liệu

**return** MainActivity.*sinhViens*.size();

}

@Override

**public** Object getItem(**int** position) {

/\*

\* Lấy đối tượng của 1 dòng dữ liệu

\* theo vị trí (position)

\*/

**return** MainActivity.*sinhViens*.get(position);

}

@Override

**public** **long** getItemId(**int** position) {

/\*

\* Lấy id của 1 dòng dữ liệu

\* theo vị trí (position)

\*/

**return** MainActivity.*sinhViens*.get(position).get\_id();

}

@Override

**public** View getView(**final** **int** position, View convertView,

ViewGroup parent) {

// Lấy layout từ thư mục res

View view = inflater.inflate(R.layout

.*design\_item\_list\_view*, **null**);

// Thiết lập thông tin hiển thị

textView = (TextView) view.findViewById(R.id.*tv\_ten*);

textView.setText(MainActivity.*sinhViens*

.get(position).get\_ten());

textView = (TextView) view.findViewById(R.id.*tv\_time*);

textView.setText(MainActivity.*sinhViens*

.get(position).get\_lop());

// Bắt sự kiện click lên imageView.

((ImageView) view.findViewById(R.id.*imageView*))

.setOnClickListener(**new** OnClickListener() {

@Override

**public** **void** onClick(View v) {

MainActivity.*database*.xoa

(MainActivity.*sinhViens*.get(position));

MainActivity.*sinhViens*.remove(position);

// MainActivity.listView.invalidateViews();

notifyDataSetChanged();

}

});

**return** view;

}

}

* 1. Tạo cơ sở dữ liệu SQLite phức tạp hơn

Đề bài:

* Tạo một cơ sở dữ liệu SQLite gồm:
  + Bảng Nhân Viên: Mã NV, Họ Tên, Ngày Sinh, Giới Tính, Địa Chỉ, Điện Thoại, Ghi Chú.
  + Bảng Khách Hàng: Mã KH, Họ Tên, Ngày Sinh, Giới Tính, Email, Điện Thoại. Bảng Hóa Đơn: Mã HD, Ngày Bán, Mã NV, Mã KH, Tổng Tiền.
  + Bảng Chi Tiết Hóa Đơn: Mã HD, Mã Sản Phẩm, Giá Bán, Số Lượng, Tổng Tiền.
  + Bảng Sản Phẩm: Mã SP, Tên SP, Ngày SX, Giá Bán, Số Lượng.
* Tạo cơ sở dữ liệu SQLite có một bảng Nhân Viên. Thực hiện thêm, xóa, sửa, sắp xếp, lấy tất cả dữ liệu, lấy theo mã, lấy theo tên.
* Thêm 3 bảng còn lại vào cơ sở dữ liệu vừa tạo. Ràng buộc khóa chính, khóa ngoại. Xây dựng chức năng: tìm kiếm (Nhân Viên, Sản Phẩm, Khách Hàng), thực hiện mua/ bán hàng, thống kê số sản phẩm bán được, doanh thu theo thời gian nhập (ngày bắt đầu và ngày kết thúc), thống kê sản phẩm bán nhiều nhất trong tháng.

*Giao diện:* Tự thiết kế.

Gợi ý thực hiện:

* Tạo mô hình 3 – Layer để dễ quản lý.
* Tạo liên kết khóa ngoại như sau: “FOREIGN KEY (ten\_cot\_cua\_bang\_can\_lien\_ket) REFERENCES ten\_bang\_lien\_ket (khoa\_chinh) ”.

1. QUẢN LÝ DỮ LIỆU VỚI CONTENT PROVIDER

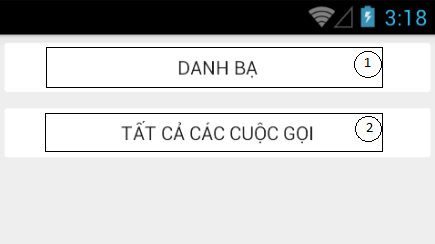
|  |  |
| --- | --- |
| **🖰** | Mục tiêu:  Có khả năng truy xuất tài nguyên dùng chung trên thiết bị.  Xây dựng ứng dụng với cơ chế chia sẻ tài nguyên.  Truy vấn, chỉnh sửa nội dung, thêm/xóa sửa dữ liệu thông qua ContentProvider. |

* 1. Đọc danh bạ và lịch sử các cuộc gọi điện thoại.

**Đề bài:**

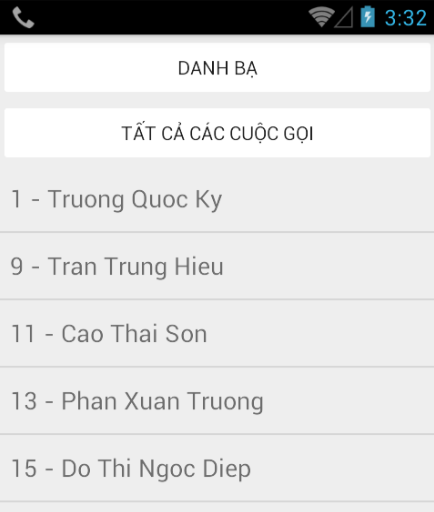
* Hãy viết chương trình để đọc danh bạ và lịch sử các cuộc gọi điện thoại.

*Giao diện:*



*Hình 2.1*

Nhấn vào Button “DANH BẠ” (1) hiển thị danh bạ điện thoại bên dưới:



*Hình 2.2*

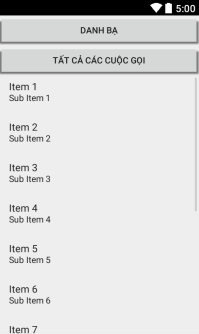
Tương tự, nhấn vào Button “TẤT CẢ CÁC CUỘC GỌI” (2) hiển thị các cuộc gọi bên dưới.

Gợi ý thực hiện:

* Tạo Uri trỏ đến bảng dữ liệu cần lấy. Ví dụ: Lấy bảng people là: content://contacts/people
* Lấy lớp “ContentResolver” truy xuất đến phương thức “query” để lấy dữ liệu, dữ liệu trả về là một Cursor. Duyệt Cursor ta lấy được dữ liệu.
* Xin quyền đọc danh bạ.

Hướng dẫn chi tiết:

1. Thiết kế giao diện trong file “activity\_main.xml” như sau:



*Hình 2.3*

1. Viết xử lý trong file “MainActivity.java”.
   1. Thực hiện lấy tất cả danh bạ khi nhấn vào Button “Danh Bạ”:

**public** **void** hienThiDanhBa(View v) {

ArrayList<String> list = **new** ArrayList<String>();

// Uri dùng để lưu trữ đường dẫn truy xuất

Uri uri = **null**;

// Đường dẫn trỏ đến bảng dữ liệu people

uri = ContactsContract.CommonDataKinds.Phone

.*CONTENT\_URI*;

// Lấy dữ liệu thông qua ContentResolver

Cursor c = getContentResolver().query(uri,

**null**, **null**, **null**, **null**);

// Lấy dữ liệu từ Cursor đổ vào list

c.moveToFirst();

**while** (!c.isAfterLast()) {

String s = "";

String idColumnName = ContactsContract

.Contacts.*\_ID*;

**int** idIndex = c.getColumnIndex(idColumnName);

s = c.getString(idIndex) + " - ";

String nameColumnName = ContactsContract

.Contacts.*DISPLAY\_ten*;

**int** nameIndex = c.getColumnIndex

(nameColumnName);

s += c.getString(nameIndex);

c.moveToNext();

list.add(s);

}

c.close();

// Hiển thị lên listView

ArrayAdapter<String> adapter = **new**

ArrayAdapter<String>(**this**,

android.R.layout.*simple\_list\_item\_1*, list);

listView.setAdapter(adapter);

}

* 1. Thực hiện lấy tất cả các cuộc gọi khi nhấn vào Button “Nhật ký cuộc gọi”:

**public** **void** nhatKyCuocGoi(View v) {

ArrayList<String> list = **new** ArrayList<String>();

Uri uri = CallLog.Calls.*CONTENT\_URI*;

String[] projection = **new** String[]

{Calls.*DATE*, Calls.*NUMBER*};

Cursor c = getContentResolver().query(uri,

projection, **null**, **null**, **null**);

c.moveToFirst();

**while** (!c.isAfterLast()) {

String s = "";

**for** (**int** i = 0; i < c.getColumnCount(); i++) {

s += c.getString(i) + "\n";

}

list.add(s);

c.moveToNext();

}

c.close();

ArrayAdapter<String> adapter = **new**

ArrayAdapter<String>(**this**,

android.R.layout.*simple\_list\_item\_1*, list);

listView.setAdapter(adapter);

}

1. Xin quyền trong file “AndroidManifest.xml”:

<uses-permission android:name=*"android.permission.READ\_CONTACTS"* />

<uses-permission android:name=*"android.permission.READ\_CALL\_LOG"* />

* 1. Tạo cơ sở dữ liệu dùng chung sử dụng Content Provider.

**Đề bài:**

* Tạo cơ sở dữ liệu bảng nhân viên gồm các thuộc tính: mã nhân viên, tên nhân viên và ngày sinh. Thực hiện truy vấn hiển thị lên giao diện.

*Chú ý:* Tạo Content Provider để thực hiện truy vấn, thêm, xóa… Và hiển thị danh sách truy vấn lên ListView.

*Giao diện:*



*Hình 2.4*

Khi nhấn vào Button “THÊM” thì lấy dữ liệu nhập từ 2 EditText (1) và (2) thêm vào cơ sở dữ liệu và hiển thị xuống ListView (5) bên dưới.



*Hình 2.5*

Tương tự, khi nhấn vào Button “HIỂN THỊ TẤT CẢ” thì lấy tất cả trong dữ liệu trong cơ sở dữ liệu hiển thị lên ListView (5) bên dưới.

Gợi ý thực hiện:

* Tạo lớp “NhanVienProvider” kế thừa lớp “ContentProvider”, sau đó tạo cơ sở dữ liệu. Viết các câu truy vấn dữ liệu trong các phương thức overrider từ ContentProvider cho cơ sở dữ liệu vừa tạo.
* Khai báo thẻ “provider” với thuộc tính “name” là: “NhanVienProvider” trong file “AndroidManifest.xml”.

Hướng dẫn chi tiết:

1. Tạo cơ sở dữ liệu như hướng dẫn bài tập tạo cơ sở dữ liệu đơn giản (bài tập 1.1), gồm có tên cơ sở dữ liệu, bảng và cột như sau:

**static** **final** String *TEN\_DATABASE*= "QuanLyNhanVien";

**public** **static** **final** String *TEN\_BANG\_NHANVIEN* = "NhanVien";

**public** **static** **final** String *\_ID* = "\_id";

**public** **static** **final** String *TEN* = "ten";

**public** **static** **final** String *NGAYSINH* = "ngaysinh";

1. Tạo ContentProvider.
   1. Tạo lớp “NhanVienProvider” kế thừa “ContenProvider” và Override lại các phương thức của lớp này. Lớp “NhanVienProvider” dùng để thực hiện việc truy vấn, thêm, xóa, sửa cơ sở dữ liệu theo Uri.
   2. Khai báo một số lớp và biến hỗ trợ cho việc tương tác với cơ sở dữ liệu:

// Tên package của ứng dụng, cũng là nơi để lưu dữ liệu dùng chung.

**static** **final** String *PROVIDER\_TEN* = "com.tqky.contentprovidercreate";

**static** **final** String *URI* = "content://" + *PROVIDER\_TEN* + "/NhanVien";

**static** **final** Uri *CONTENT\_URI* = Uri.*parse*(*URI*);

**private** SQLiteDatabase db;

**private** **static** HashMap<String, String> *NHANVIEN\_PROJECTION\_MAP*;

// Quy định truy xuất đến bảng dữ liệu

**static** **final** **int** *NHANVIEN* = 1;

// Quy định truy xuất đến dòng dữ liệu thông qua ID

**static** **final** **int** *NHANVIEN\_ID* = 2;

/\*

\* UriMatcher là cây Uri dùng để chứa các node là Uri.

\* Khi khởi tạo ta cần truyền cho nó mã gốc.

\*/

**static** **final** UriMatcher *uriMatcher*;

/\*

\* Khởi tạo UriMaticher chứa node Uri lấy tất cả dữ

\* liệu và lấy theo id

\*/

**static**{

*uriMatcher* = **new** UriMatcher(UriMatcher.*NO\_MATCH*);

*uriMatcher*.addURI(*PROVIDER\_TEN*, "NhanVien", *NHANVIEN*);

*uriMatcher*.addURI(*PROVIDER\_TEN*, "NhanVien/#", *NHANVIEN\_ID*);

}

*Chú ý:* Uri có cấu trúc định dạng như sau: <prefix>://<authority>/<data\_type>/<id>

* + - <prefix>: Nó luôn được thiết lập là content://
    - <authority>: Chỉ định tên cụ thể của Content Provider (Ví dụ: contacts, browser,…). Đối với một số Content Provider khác bạn sẽ phải chỉ định tên đầy đủ (Ví dụ: com.tqky.contentprovidercreate).
    - <data\_type>: Chỉ rõ bảng dữ liệu (Ví dụ: Để lấy tất cả các liên hệ trong Contacts Content Provider thì bảng dữ liệu là people và URI sẽ là: content://contacts/people.
    - <id>: Chỉ định rõ một id của một dòng trong bảng (Ví dụ: Nếu bạn muốn lấy id thứ 9 trong Contacts Content Provider thì URI sẽ là: content://contacts/people/9.
  1. Sau khi khởi tạo lớp “NhanVienProvider” thì phương thức chạy đầu tiên đó là “onCreate”. Chúng ta lợi dụng hàm này để khởi tạo đối tượng không cần phải tạo hàm khởi tạo cho lớp “NhanVienProvider”:

**public** **boolean** onCreate() {

Context context = getContext();

DBHelper dbHelper = **new** DBHelper(context);

// Truy xuất đọc, ghi file

db = dbHelper.getWritableDatabase();

**return** (db == **null**) ? **false**:**true**;

}

* 1. Tiếp theo viết xử lý cho các phương thức Override còn lại (delete, update, insert, ...). Viết xử lý cho phương thức “delete”:

**public** **int** delete(Uri uri, String selection,

String[] selectionArgs) {

**int** count = 0;

// Lấy dữ liệu theo đúng Uri truyền vào

**switch** (*uriMatcher*.match(uri)){

**case** *NHANVIEN*:

// Xóa dòng dữ liệu theo điều kiện

count = db.delete(DBHelper

.*TEN\_BANG\_NHANVIEN*,

selection, selectionArgs);

**break**;

**case** *NHANVIEN\_ID*:

// Xóa dòng dữ liệu theo id và điều kiện

count = db.delete(DBHelper

.*TEN\_BANG\_NHANVIEN*,

DBHelper.*\_ID* + "="

+ uri.getPathSegments().get(1)

+ (!TextUtils.*isEmpty*(selection) ? " AND ("

+ selection + ')': ""), selectionArgs);

**break**;

**default**:

**throw** **new** IllegalArgumentException("Unknown URI " + uri);

}

// Thông báo đã có 1 dòng dữ liệu đã thay đổi để đồng bộ

// dữ liệu.

getContext().getContentResolver().notifyChange(uri, **null**);

**return** count;

}

* 1. Lấy loại mà Android cung cấp dữ liệu bằng Uri truyền vào:

**public** String getType(Uri uri) {

**switch** (*uriMatcher*.match(uri)) {

**case** *NHANVIEN*:

/\*

\* Loại trả về là danh sách dữ liệu

\*/

**return** "vnd.android.cursor.dir/NhanVien";

**case** *NHANVIEN\_ID*:

/\*

\* Loại trả về là 1 dòng dữ liệu

\*/

**return** "vnd.android.cusor.item/NhanVien";

**default**:

**throw** **new** IllegalArgumentException("Unsupport URI " + uri);

}

}

* 1. Thêm một dòng vào cơ sở dữ liệu:

**public** Uri insert(Uri uri, ContentValues values) {

// Thêm cơ sơ dữ liệu

**long** rowID = db.insert(DBHelper.*TEN\_BANG\_NHANVIEN*,

"", values);

**if**(rowID > 0){

// Trả về Uri đã thêm thành công

Uri \_uri = ContentUris.*withAppendedId*(*CONTENT\_URI*, rowID);

// Thông báo đã có 1 dòng dữ liệu đã thay đổi để đồng bộ

// dữ liệu.

getContext().getContentResolver().notifyChange(\_uri, **null**);

**return** \_uri;

}

**throw** **new** SQLException("Failed to add a record into " + uri);

}

* 1. Thực hiện truy vấn dữ liệu:

**public** Cursor query(Uri uri, String[] projection, String selection,

String[] selectionArgs, String sortOrder) {

// Tạo đối tượng truy vấn dữ liệu

SQLiteQueryBuilder qb = **new** SQLiteQueryBuilder();

// Bảng cần thực hiện truy vấn

qb.setTables(DBHelper.*TEN\_BANG\_NHANVIEN*);

// Kiểm tra đối số truyền vào thuộc các lấy nào

**switch** (*uriMatcher*.match(uri)) {

**case** *NHANVIEN*:

qb.setProjectionMap(*NHANVIEN\_PROJECTION\_MAP*);

**break**;

**case** *NHANVIEN\_ID*:

qb.appendWhere(DBHelper.*\_ID* +

"=" + uri.getPathSegments().get(1) );

**break**;

**default**:

**throw** **new** IllegalArgumentException("Unknown URI " + uri);

}

// Thực hiện sắp xếp theo tên

**if**(sortOrder == **null** || sortOrder == ""){

sortOrder = DBHelper.*TEN*;

}

// Thực hiện truy vấn theo đối số truyền vào

Cursor c = qb.query(db, projection, selection, selectionArgs,

**null**, **null**, sortOrder);

// Đăng ký để xem có một Uri thay đổi

c.setNotificationUri(getContext().getContentResolver(), uri);

**return** c;

}

* 1. Thực hiện cập nhập dữ liệu:

**public** **int** update(Uri uri, ContentValues values, String selection,

String[] selectionArgs) {

**int** count = 0;

**switch** (*uriMatcher*.match(uri)){

**case** *NHANVIEN*:

// Cập nhập dữ liệu theo điều kiện

count = db.update(DBHelper.*TEN\_BANG\_NHANVIEN*,

values, selection,

selectionArgs);

**break**;

**case** *NHANVIEN\_ID*:

// Cập nhập dữ liệu theo điều kiện và id

count = db.update(DBHelper.*TEN\_BANG\_NHANVIEN*,

values, DBHelper.*\_ID* + "=" +

uri.getPathSegments().get(1) + (!TextUtils.*isEmpty*(selection) ?

" AND (" + selection + ')': ""), selectionArgs);

**break**;

**default**:

**throw** **new** IllegalArgumentException("Unknown URI " + uri);

}

// Thông báo đã có 1 dòng dữ liệu đã thay đổi để đồng bộ

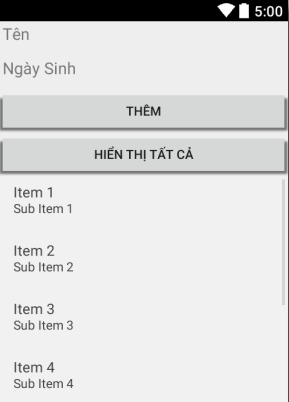
// dữ liệu.

getContext().getContentResolver().notifyChange(uri, **null**);

**return** count;

}

1. Thiết kế giao diện cho file “activity\_main.xml”:



*Hình 2.6*

1. Viết xử lý trong file “MainActivity.java”.
   1. Bắt sự kiện khi nhấn Button “THÊM” dữ liệu và xử lý:

**public** **void** onClickAdd(View view){

ContentValues values = **new** ContentValues();

values.put(DBHelper.*TEN*,

((EditText) findViewById(R.id.*editTextTen*))

.getText().toString());

values.put(DBHelper.*NGAYSINH*,

((EditText) findViewById(R.id.*editTextNgaySinh*))

.getText().toString());

getContentResolver().insert(NhanVienProvider.*CONTENT\_URI*,

values);

hienThiDuLieu();

}

* 1. Viết phương thức hiển thị tất cả dữ liệu khi người dùng ấn vào “HIỂN THỊ TẤT CẢ”:

**private** **void** hienThiDuLieu() {

String URI = "content://com.tqky.contentprovidercreate" +

"/NhanVien";

Uri NhanVien = Uri.*parse*(URI);

Cursor c = getContentResolver().query(NhanVien,

**null**, **null**, **null**, **null**);

**if** (c == **null**) {

**return**;

}

List<String> list = **new** ArrayList<>();

**if**(c.moveToFirst()){

**while** (!c.isAfterLast()) {

list.add(c.getString(c.getColumnIndex

(DBHelper.*\_ID*)) +

"\r\n " + c.getString(c.getColumnIndex

(DBHelper.*TEN*)) +

"\r\n " + c.getString(c.getColumnIndex

(DBHelper.*NGAYSINH*)));

c.moveToNext();

}

}

ArrayAdapter<String> adapter = **new** ArrayAdapter<>

(getApplicationContext(), R.layout.*simple\_list\_item*, list);

((ListView) findViewById(R.id.*listView*)).setAdapter(adapter);

}

* 1. Sử dụng CursorAdapter để truy vấn dữ liệu.

Đề bài:

* Làm tiếp bài tập 2.2, thực hiện tìm kiếm dữ liệu theo nội nhập và hiển thị dữ liệu đã được lọc theo nội dung nhập.

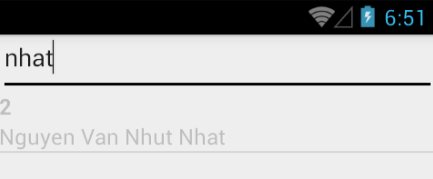
*Chú ý:* Sử dụng CursorAdapter để thực hiện truy vấn dữ liệu, lấy dữ liệu từ người dùng nhập bằng EditText và hiển thị dữ liệu lên giao diện bằng ListView.

*Giao diện:*



*Hình 2.7*

Khi nhập nội dung vào ô EditText (1) thì hiển thị danh sách dữ liệu đã được lọc lên ListView (2). Nếu để trống thì hiển thị tất cả dữ liệu.



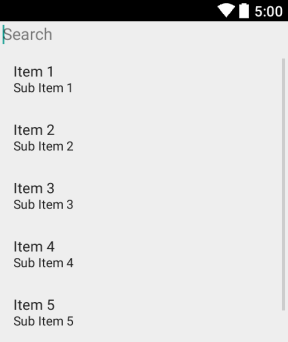
*Hình 2.8*

Gợi ý thực hiện:

* Trong lớp “MainActivity” kế thừa “LoaderCallbacks<Cursor>” và override lại các phương thức của Interface này.
* Tạo một “SimpleCursorAdapter” để setAdapter cho ListView. Sau khi setAdapter ta gọi LoaderManager lên truy xuất đến phương thức “initLoader” để khởi tạo bộ nạp.
* Viết câu truy vấn trong phương thức “onCreateLoader” và nhận giá trị trả về trong phương thức “onLoadFinished” .

Hướng dẫn chi tiết:

1. Thiết kế giao diện:



*Hình 2.9*

1. Viết xử lý trong MainActivity.
   1. Kế thừa FragmentActivity để gọi lớp “LoaderManager” của bộ thư việc v4 thông qua phương thức “getSupportLoaderManager” và triển khai interface “LoaderCallbacks<Cusor>” để thực hiện truy vấn và cập nhật nội dung truy vấn lên giao diện cho người dùng xem. Thực hiện Override lại các phương thức cảu interface.
   2. Khai báo một số biến cần để xử lý hiển thị:

// Lớp thiết lập hiển thị cho ListView

**private** SimpleCursorAdapter adapter;

**private** ListView listView;

// Các cột cần lấy

String[] from = {"\_id", "ten"};

// Lấy id của 2 textView

**int**[] to = {android.R.id.*text1*, android.R.id.*text2*};

**private** Uri NHANVIEN\_URI;

* 1. Viết xử lý khởi tạo một số biến và thiết lập hiển thị cho ListView:

NHANVIEN\_URI = Uri.*parse*

("content://com.tqky.contentprovidercreate" +

"/NhanVien");

listView = (ListView) findViewById(R.id.*listView*);

((EditText) findViewById(R.id.*editText*))

.addTextChangedListener(**this**);

adapter = **new** SimpleCursorAdapter

(getApplicationContext(),

android.R.layout.*two\_line\_list\_item*, **null**,

from, to, 2);

listView.setAdapter(adapter);

* 1. Viết xử lý cho phương thức “onCreateLoader” được Override của interface “LoaderCallbacks”:

**public** Loader<Cursor> onCreateLoader(**int** arg0,

Bundle bundle) {

String select = **null**;

**if** (bundle != **null**) {

String searchText = bundle.getString("SearchText");

select = "ten" + " like '%" + searchText + "%'";

}

// Lấy dữ liệu trả ra CursorLoader

CursorLoader cursorLoader = **new** CursorLoader(**this**,

NHANVIEN\_URI, **null**, select, **null**, **null**);

**return** cursorLoader;

}

* 1. Viết xử lý cập nhật lại giao diện sau khi truy vấn xong trong phương thức “onLoadFinished” được Override của interface “LoaderCallbacks”:

**public** **void** onLoadFinished(Loader<Cursor> arg0,

Cursor cursor) {

// Thực hiện cập nhật lại dữ liệu

adapter.swapCursor(cursor);

}

* 1. Khi thực hiện tìm kiếm ta viết xử lý như sau:

**public** **void** onTextChanged(CharSequence s, **int** start,

**int** before, **int** count) {

Bundle bundle = **new** Bundle();

bundle.putString("SearchText", s.toString());

// lấy dữ liệu tìm kiếm và thực hiện truy vấn

getSupportLoaderManager().restartLoader(0,

bundle, **this**);

}

1. MENU, ACTION BAR VÀ TOOLBAR

|  |  |
| --- | --- |
| **🖰** | Mục tiêu:  Hiểu và biết cách sử dụng Menu và ActionBar vào việc xây dựng ứng dụng. |

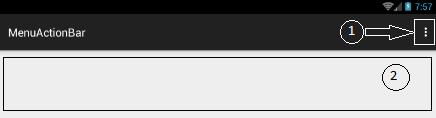
* 1. Tạo và sử dụng OptionMenu

**Đề bài:**

* Lọc những nhân viên có độ tuổi nhỏ hơn 25 và tất cả nhân viên.

*Chú ý:* Sử dụng OptionMenu.

*Giao diện:*



*Hình 3.1*

Khi nhấn vào OptionMenu (1) thì hiển thị các item:



*Hình 3.2*

Khi chọn vào item thì lọc dữ liệu hiển thị lên ListView (2).

Gợi ý thực hiện:

* Tạo file “main.xml” trong thư mục “res/menu/main.xml”.
* Trong hàm “MainActivity” override lại 2 phương thức “onCreateOptionsMenu” và “onOptionsItemSelected” để tạo và bắt sự kiện khi chọn item menu.

Hướng dẫn chi tiết:

1. Thiết kế file “main.xml” như sau:

<menu xmlns:android=*"http://schemas.android.com/apk/res/android"*

xmlns:app=*"http://schemas.android.com/apk/res-auto"*

xmlns:tools=*"http://schemas.android.com/tools"*

tools:context=*"com.tqky.menuactionbar.MainActivity"* >

<item

android:id=*"@+id/under\_25"*

android:orderInCategory=*"100"*

android:title=*"@string/under\_25"*

app:showAsAction=*"never"*/>

<item

android:id=*"@+id/all"*

android:orderInCategory=*"100"*

android:title=*"@string/all"*

app:showAsAction=*"never"*/>

</menu>

*Giải thích:*

* orderInCategory: có thể hiểu là quyền của item này trong 1 group nào đó.
* showAsAction: xác định item này hiện ở đâu trên ActionBar.

1. Viết xử lý trong lớp “MainActivity”.
   1. Tương tự các bước như bài 3.1, tóm tắt lại các bước như sau: đầu tiên triển khai interface “LoaderCallbacks” sau đó thiết lập hiển thị cho ListView và khai báo Uri để truy xuất đến bảng dữ liệu cần lấy. Thực hiện truy vấn và hiển thị thông qua 2 phương thức “onCreateLoader” và “onLoadFinished”.
   2. Thực hiện tạo “Option Menu” và viết xử lý để truy vấn.
      * Tạo “Option Menu” thông qua phương thức “onCreateOptionsMenu” được Override từ Activity:

**public** **boolean** onCreateOptionsMenu(Menu menu) {

getMenuInflater().inflate(R.menu.*main*, menu);

**return** **true**;

}

* + - Kiểm tra người dùng chọn item nào thông qua phương thức “onOptionsItemSelected” được Override từ Activity:

**public** **boolean** onOptionsItemSelected(MenuItem item) {

**int** id = item.getItemId();

Bundle bundle = **new** Bundle();

**if** (id == R.id.*under\_25*) {

bundle.putInt("age", 25);

} **else** **if** (id == R.id.*all*) {

bundle.putInt("age", 0);

}

getSupportLoaderManager().restartLoader(0,

bundle, **this**);

**return** **true**;

}

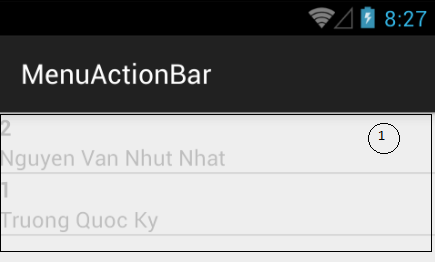
* 1. Tạo và dùng ContextMenu.

**Đề bài:**

* Sử dụng lại bài tập 3.1. Khi nhấn giữ vào nhân viên (trong danh sách hiển thị tất cả nhân viên) nào thì hiển thị thông tin chi tiết của nhân viên đó.

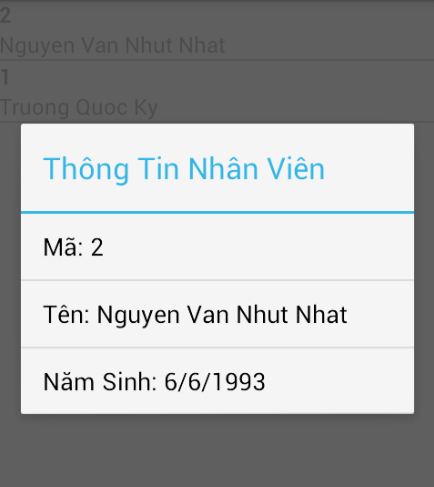
*Chú ý:* Danh sách nhân viên hiển thị trên ListView, nhấn giữ vào item của ListView thì dùng ContextMenu để hiển thị thông tin chi tiết của nhân viên.

*Giao diện:*



*Hình 2.4*

Khi nhấn giữ vào 1 item của ListView (1), thì hiển thị thông tin chi tiết nhân viên như hình:



*Hình 2.5*

Gợi ý thực hiện:

* Gọi phương thức “registerForContextMenu” và truyền ListView vào.
* Override phương thức “onCreateContextMenu” và viết xử lý để hiển thị các nội dung:

Hướng dẫn chi tiết:

1. Viết xử lý trong file “MainActivity.java”.
   1. Đăng ký “ContextMenu” cho ListView thông qua phương thức “registerForContextMenu(listView)”.
   2. Tạo “ContextMenu” và viết xử lý hiển thị thông qua phương thức “onCreateContextMenu” được Override từ Activity:

**public** **void** onCreateContextMenu(ContextMenu menu,

View v, ContextMenuInfo menuInfo) {

**super**.onCreateContextMenu(menu, v, menuInfo);

AdapterContextMenuInfo info =

(AdapterContextMenuInfo) menuInfo;

**long** id = adapter.getItemId(info.position);

Uri \_URI = Uri

.*parse*("content://com.tqky." +

"contentprovidercreate/NhanVien/"

+ id);

Cursor cursor = getContentResolver()

.query(\_URI, **null**, **null**, **null**, **null**);

cursor.moveToFirst();

menu.setHeaderTitle("Thông Tin Nhân Viên");

menu.add(0, v.getId(), 0, "Mã: "

+ cursor.getString(0));

menu.add(0, v.getId(), 0, "Tên: "

+ cursor.getString(1));

menu.add(0, v.getId(), 0, "Năm Sinh: "

+ cursor.getString(2));

cursor.close();

}

* 1. Bắt sự kiện khi người dùng click vào item của “ContextMenu” thông qua phương thức “onContextItemSelected” được Override từ Activity.
  2. Sử dụng ActionBar

**Đề bài:**

Sử dụng lại bài tập 3.2: tạo menu có 2 item (tuổi nhỏ hơn 25 và tất cả) và xử lý khi nhấn vào từng item và hiển thị danh sách lên cho người dùng. Khi nhấn giữ vào item thì dùng ContextMenu để hiển thị thông tin chi tiết của từng nhân viên. Tiếp tục bổ sung đề: khi nhấn item ListView thì ActionBar hiển thị giao diện (1 dòng text hiển thị tổng số item được chọn, 1 button để bỏ item đã chọn, 1 item xóa các item trên ListView đã chọn), về phần giao diện ListView chuyển sang trạng thái click item nào thì đổi màu để nhận biết item đó đang được chọn và click lần 2 thì bỏ chọn, khi các item được bỏ chọn hết hoặc ấn nút xóa trên ActionBar thì quay về trạng thái như bài 3.2.

Gợi ý thực hiện:

* Tạo file Menu mới bằng xml, có chứa các item
* Tạo ArrayList<Integer> dùng để chứa các id của các dòng item chọn.
* Tạo biến Menu dùng để lưu lại Menu mặc định của phương thức onCreateOptionMenu.
* Bắt sự kiện nhấn cho ListView, mỗi lần nhấn là lấy id thêm vào ArrayList. Và cài đặt màu cho item đó. Sau đó gọi hàm onCreateOptionMenu để cập nhật lại menu cho AcitionBar.

Hướng dẫn chi tiết: (học viên tự làm).

1. ACTION PROVIDER VÀ ĐIỀU KHIỂN TÌM KIẾM

|  |  |
| --- | --- |
| **🖰** | Mục tiêu:  Hiểu và biết cách sử dụng Action Provider và điều khiển tìm kiếm vào việc xây dựng ứng dụng. |

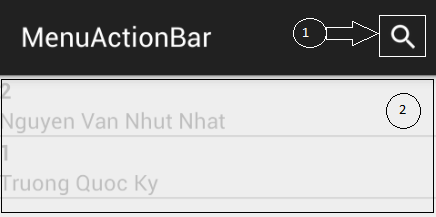
* 1. Sử dụng SearchView.

**Đề bài:**

* Sử dụng bài 3.3, thêm chức năng tìm kiếm nhân viên theo tên.

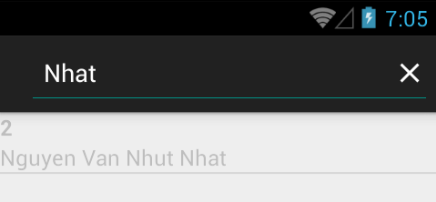
*Chú ý:* Sử dụng SearchView để lấy nội dung nhập từ người dùng.

*Giao diện:*



*Hình 4.1*

Khi nhấn vào SearchView (1) thì hiển thị ra ô nhập, người dùng nhập dữ liệu vào thì ListView (2) sẽ hiển thị dữ liệu đã được lọc theo nội dung nhập.



*Hình 4.2*

Gợi ý thực hiện:

* Vào thư mục “res” tạo thư mục “menu” (nếu chưa có). Tạo thêm file “main.xml” (nếu chưa có). Trong file “main.xml” ta thêm thẻ “item” trong thẻ “menu”, trong item thêm thuộc tính “actionViewClass” và truyền vào một cái “SearchView”.
* Vào “MainActivity.java” trong hàm “onCreateOptionsMenu” lấy SearchView và bắt sự kiện “setOnQueryTextListener” để lấy dữ liệu nhập từ bàn phím.

Hướng dẫn chi tiết:

1. Thiết kế file “main.xml” trong thư mục “res/menu” như sau:

<menu xmlns:android=*"http://schemas.android.com/apk/res/android"*

xmlns:app=*"http://schemas.android.com/apk/res-auto"*

xmlns:tools=*"http://schemas.android.com/tools"*

tools:context=*"com.tqky.menuactionbar.MainActivity"* >

<item

android:id=*"@+id/search"*

android:icon=*"@android:drawable/ic\_menu\_search"*

android:title=*""*

app:actionViewClass=*"android.support.v7.widget.SearchView"*

app:showAsAction=*"never"*>

</item>

</menu>

*Chú thích:* thuộc tính “actionViewClass” của thẻ item dùng để chứa đối tượng View được thiết kế sẵn trong Java code.

1. Viết xử lý trong file “MainActivity.java”.
   1. Lấy đối tượng SearchView vừa tạo trong file “main.xml” và thiết lập sự kiện trong phương thức “onCreateOptionsMenu” được Override của Activity:

setTitle(getResources().getString

(R.string.*app\_name*));

getMenuInflater().inflate(R.menu.*main*, menu);

MenuItem searchItem = menu.findItem

(R.id.*search*);

SearchView searchView = (SearchView)

MenuItemCompat.*getActionView*(searchItem);

searchView.setOnQueryTextListener(**this**);

*Chú thích:* Trong phương thức “onCreateOptionsMenu” có đối tượng MenuItem là một Interface dùng để truy xuất đến mục menu (main.xml) đã tạo trước đó. Tiếp theo ta dùng lớp “MenuItemCompat” để lấy ra đối tượng View mà ta đã cài đặt cho nó, và ép kiểu về lớp đối tượng View này.

* 1. Trong lớp MainActivity triển khi interface “OnQueryTextListener” để bắt lại dữ liệu nhập khi người dùng tương tác, sau đó Override các phương thức của interface này và viết sử

@Override

**public** **boolean** onQueryTextChange(String arg0) {

Bundle bundle = **new** Bundle();

bundle.putString("SearchText", arg0);

getSupportLoaderManager().restartLoader(0,

bundle, **this**);

**return** **true**;

}

@Override

**public** **boolean** onQueryTextSubmit(String arg0) {

**return** **false**;

}

1. CÁC CHUYỂN HOẠT TRONG ỨNG DỤNG

|  |  |
| --- | --- |
| **🖰** | Mục tiêu:  Hiểu và biết cách sử dụng các chuyển hoạt vào việc xây dựng ứng dụng (đặc biệt là các ứng dụng Game).  Tối ưu hoá cơ chế sắp xếp bố cục giao diện, chuyển hoạt và tuỳ chỉnh hiển thị điều khiển. |

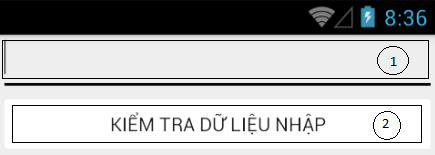
* 1. Tạo hiệu ứng lắc.

**Đề bài:**

* Tạo hiệu ứng lắc cho ô nhập liệu để cho biết người dùng nhập liệu sai ở vị trí nào trên giao diện.

*Chú ý:* Tạo hiệu ứng lắc cho EditText.

*Giao diện:*



*Hình 5.1*

Người dùng nhập dữ liệu vào EditText (1), nhấn nút “KIỂM TRA DỮ LIỆU NHẬP” (2): nếu để trống thì EditText sẽ có hiệu ứng lắc qua lại 7 lần liên tiếp.

Gợi ý thực hiện:

* Tạo thư mục “anim” trong “res”, sau đó tạo file “shake.xml” để viết hiệu ứng cho đối tượng View.
* Lấy hiệu ứng vừa viết xong trả ra lớp “Animation” thông qua lớp “AnimationUtils” để cài đặt cho đối tượng View.

Hướng dẫn chi tiết:

1. Tạo thư mục “anim” trong “res”, sau đó tạo file “shake.xml” có xử lý như sau:

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<translate xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:duration=*"1000"*

android:fromXDelta=*"0"*

android:interpolator=*"@anim/cycle\_7"*

android:toXDelta=*"10"* />

*Giải thích :*

* “duration” là thời gian mà hiệu ứng này chạy.
* “fromXDelta”, “toXDelta” là từ vị trí nào đến vị trí nào theo trục x.
* “interpolator” cho phép chúng ta thêm 1 hiệu ứng, hành động khác. Ở đây tạo 1 chu kỳ hoạt động cho hiệu ứng lắc này là 7 lần và viết trong file “res/anim/cycle\_7.xml” như sau:

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<cycleInterpolator xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:cycles=*"7"* />

1. Vào trong file “MainActivity.java” viết xử lý truy xuất đến file xml vừa thiết kế và dùng:

EditText editText;

@Override

**protected** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

editText = (EditText) findViewById(R.id.*editText1*);

((Button) findViewById(R.id.*button1*)).

setOnClickListener(**new** OnClickListener() {

@Override

**public** **void** onClick(View v) {

Animation shake = AnimationUtils

.*loadAnimation*(MainActivity.**this**,

R.anim.*shake*);

editText.startAnimation(shake);

}

});

}

*Giải thích :* lớp AnimationUtils cung cấp phương thức “loadAnimation” để lấy file hiệu ứng thiết kế trong thư mục “res/anim”. Để cài đặt hiệu ứng cho đối tượng View nào thì ta chỉ cần lấy đối tượng View đó truy xuất đến phương thức “startAnimation” và truyền vào một Animation thì nó sẽ hoạt động.

* 1. Tạo hiệu ứng ẩn hiện.

**Đề bài:**

* Tạo hiệu ứng cho danh sách dữ liệu, khi lướt đến đâu thì từng dòng dữ liệu từ không có màu sắc hiện rõ dần lên.

*Chú ý:* Dùng ListView để thể hiện yêu cầu đề bài. Tương tự có thể làm cho nhiều đối tượng View khác.

Gợi ý thực hiện:

* Tương tự ta tạo file xml trong thư mục anim và viết xử lý xử lý.

Hướng dẫn chi tiết:

1. Tạo và viết xử lý trong file “res/anim/fade\_in.xml”:

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<set xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:fillAfter=*"true"* >

<alpha

android:duration=*"1000"*

android:fromAlpha=*"0.0"*

android:interpolator=*"@android:anim/accelerate\_interpolator"*

android:toAlpha=*"1.0"* />

</set>

1. Tạo lớp Adapter cho ListView và kế thừa BaseAdapter. Sau khi Override các phương thức của lớp BaseAdapter ta viết xử lý cho phương thức “getView”:

**public** View getView(**int** position, View convertView, ViewGroup parent) {

LayoutInflater inflater = LayoutInflater.from(context);

View view = inflater.inflate(R.layout.*item*, **null**);

Animation shake = AnimationUtils.loadAnimation(context, R.anim.*fade\_in*);

view.startAnimation(shake);

((TextView) view.findViewById(R.id.*textView1*)).setText("Position = "

+ position);

**return** view;

}

1. Gọi và sử dụng trong file “MainActivity.java”.
   1. Tạo hiệu ứng quay 180 độ.

**Đề bài:**

* Hiệu ứng lật màn hình 180 độ (tức là màn hình trước quay được 90 độ thì mất, màn hình sau thì từ góc -90 độ bắt đầu quay đến 0 độ).

*Chú ý:* Làm trên Button, tương tự có thể làm trên các đối tượng View khác.

Gợi ý thực hiện:

* Tạo phương thức “flipit” có 2 tham số truyền vào dùng để tạo hiệu ứng giống như 2 đối tượng View úp lưng vào nhau, xoay một vòng thì thấy đối tượng View còn lại.
* Thiết lập vị trí xoay cho 2 đối tượng View. Một đối tượng thì xoay từ 0 độ đến 90 độ, còn một đối tượng thì từ -90 độ quay ngược về 0 độ.

Hướng dẫn chi tiết:

1. Tạo phương thức “flipit” và thiết lập các thuộc tính cho 2 đối tượng View cần xoay:

**private** Interpolator accelerator = **new**

AccelerateInterpolator();

**private** Interpolator decelerator = **new**

DecelerateInterpolator();

**public** **void** flipit(View a, View b) {

**final** View visibleList;

**final** View invisibleList;

**if** (a.getVisibility() == View.*GONE*) {

visibleList = b;

invisibleList = a;

} **else** {

invisibleList = b;

visibleList = a;

}

a.setVisibility(View.*GONE*);

ObjectAnimator visToInvis =

ObjectAnimator.*ofFloat*(visibleList,

"rotationY", 0f, 90f);

visToInvis.setDuration(500);

visToInvis.setInterpolator(accelerator);

**final** ObjectAnimator invisToVis =

ObjectAnimator.*ofFloat*(invisibleList,

"rotationY", -90f, 0f);

invisToVis.setDuration(500);

invisToVis.setInterpolator(decelerator);

visToInvis.addListener(**new**

AnimatorListenerAdapter() {

@Override

**public** **void** onAnimationEnd

(Animator anim) {

visibleList.setVisibility(View.*GONE*);

invisToVis.start();

invisibleList.setVisibility

(View.*VISIBLE*);

}

});

visToInvis.start();

a.setVisibility(View.*VISIBLE*);

}

* *Chú thích:* Ở đây thay vì chúng ta viết trên file xml thì chúng ta viết trên Java code, sử dụng lại một số thuộc tính mà đã biết. Trong đây có đối tượng ObjectAnimator để thiết lập vị trí di chuyển cho đối tượng View:
  + “rotationY” là xoay theo trục Y.
  + 0f” là vị trí bắt đầu.
  + 90f” là vị trí kết thúc.
* Còn 2 lớp “AccelerateInterpolator” và “DecelerateInterpolator” dùng để tăng giảm tốc độ xoay để tạo hiệu ứng thật hơn.

1. Gọi và sử dụng:

((Button) findViewById(R.id.*button2*)).setOnClickListener

(**new** OnClickListener() {

@Override

**public** **void** onClick(**final** View v) {

Button button = (Button) findViewById(R.id.*button3*);

button.setOnClickListener(**new** OnClickListener() {

@Override

**public** **void** onClick(View v1) {

flipit(v1, v);

}

});

flipit(v, button);

}

});

1. ASYNCTASK – THREAD – HANDLER

|  |  |
| --- | --- |
| **🖰** | Mục tiêu:  Tăng tốc ứng dụng với các luồng xử lý khác nhau.  Xây dựng chế độ làm việc ngầm cho ứng dụng với Asynctask. |

* 1. Cách sử dụng Asynctask

**Đề bài:**

* Viết chương trình đếm ngược thời gian.

*Chi tiết yêu cầu:* Có một EditText để nhập số nguyên dương, một Button khi nhấn vào thì bắt đầu đếm ngược từ số nguyên nhập về 0 và trong quá trình đếm ngược thì hiển thị số thời gian còn lại lên TextView cho người dùng xem (gồm số giây và tích tắc được lấy theo 2 số. Ví dụ: 9:99), khi nhấn vào Button thì Button và Editext đều không được tương tác cho đến khi thời gian được đếm xong (tức là về 0).

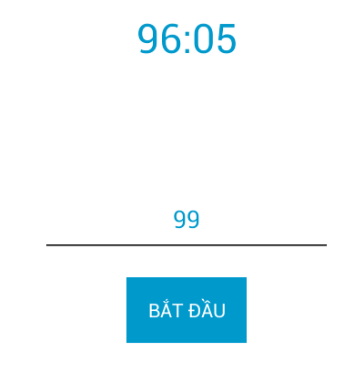
*Chú ý:* Dùng Asynctask.

*Giao diện:*



*Hình 6.1*

Nhập số và nhấn nút “BẮT ĐẦU” thì thời gian bắt đầu đếm ngược:



*Hình 6.2*

Gợi ý thực hiện:

* Tạo lớp kế thừa “AsyncTask”.
* Dùng “setEnabled” để thiết lập cho phép tương tác hoặc không cho đối tượng View

Hướng dẫn chi tiết:

1. Tạo lớp “Timer” kế thừa AsyncTask. Sau khi kế thừa cần bắt buộc Override lại phương thức “doInBackground”. Nhưng để giải quyết bài này cần Override thêm 3 phương thức “onPreExecute”, “onProgressUpdate” và “onPostExecute”. Sau khi khởi tạo lớp Timer thì phương thức “onPreExecute” sẽ chạy đầu tiên. Sau đó đến “doInBackground” và “onProgressUpdate” chạy song song, từ “doInBackground” có thể gửi dữ liệu qua “onProgressUpdate” thông qua phương thức “publishProgress”. Cuối cùng là phương thức “onPostExecute” và kết thúc.
   1. Tạo biến “num” có kiểu giá trị int dùng để lấy số nguyên dương mà người dùng nhập vào.
   2. Lấy dữ liệu nhập vào trong phương thức “onPreExecute”:

**protected** **void** onPreExecute() {

**super**.onPreExecute();

**try** {

// Lấy số nhập

num = Integer.*parseInt*

(editText.getText().toString()) \* 100;

/\*

\* Thiết lập không cho tương tác

\* với Button và EditText

\*/

button.setEnabled(**false**);

editText.setEnabled(**false**);

} **catch** (Exception e) {

}

}

* 1. Viết xử lý trong phương thức “doInBackground”:

**protected** Void doInBackground(Void... params) {

**while** (**true**) {

**try** {

**if** (num == 0) {

**return** **null**;

}

// Dừng trong 1/100 giây

Thread.*sleep*(10);

num = num - 1;

/\*

\* Cập nhật dữ liệu cho

\* onProgressUpdate

\*/

publishProgress(num);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}

* 1. Viết xử lý có “onProgressUpdate” để cập nhật giao diện trong quá trình xử lý dưới “doInBackground”:

**protected** **void** onProgressUpdate(Integer... values) {

**super**.onProgressUpdate(values);

// Lấy giá trị mà bên doInBackground gửi qua

// thông qua biến values

**int** millis = values[0] % 100;

textView.setText(values[0] / 100 + ":"

+ ((millis + "").length() != 2 ? "0"

+ millis : millis));

}

* 1. Viết xử lý khi kết thúc tiến trình trong phương thức “onPostExecute”:

**protected** **void** onPostExecute(Void result) {

**super**.onPostExecute(result);

/\*

\* Cho phép tương tác với Button

\* và EditText

\*/

button.setEnabled(**true**);

editText.setEnabled(**true**);

}

1. Viết gọi lớp vừa tạo cho chạy:

**public** **class** MainActivity **extends** Activity {

TextView textView;

EditText editText;

Button button;

@Override

**protected** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

textView = (TextView) findViewById(R.id.*textView*);

editText = (EditText) findViewById(R.id.*editText*);

button = (Button) findViewById(R.id.*button*);

}

**public** **void** onClick(View v) {

/\*

\* Khởi tạo Timer và cho thực hiện

\*/

**new** Timer().execute();

}

}

* 1. Cách sử dụng Thread.

**Đề bài:**

* Viết chương trình đếm ngược thời gian.

*Chi tiết yêu cầu:* Có một ô nhập số nguyên dương, một Button khi nhấn vào thì bắt đầu đếm ngược từ số nguyên nhập về 0 và trong quá trình đếm ngược thì hiển thị số thời gian còn lại lên TextView cho người dùng xem (gồm số giây và tích tắc được lấy theo 2 số. Ví dụ: 9:99), khi nhấn vào Button thì Button và Editext đều không được tương tác cho đến khi thời gian được đếm xong (tức là về 0).

*Chú ý:* Dùng Thread.

Gợi ý thực hiện:

* Khởi tạo Thread và truy xuất đến phương thức “start()”.
* Dùng “Handler” để gửi dữ liệu thiết lập hiển thị cho người dùng.
* Dùng “setEnabled” để thiết lập cho phép tương tác hoặc không cho đối tượng View.

Hướng dẫn chi tiết:

1. Tạo “Handler” để hỗ trợ tương tác với giao diện khi làm việc ở tiến trình.

Handler handler = **new** Handler() {

@Override

**public** **void** handleMessage(Message msg) {

**if** (msg.what == 0) {

button.setEnabled(**false**);

editText.setEnabled(**false**);

} **else** **if** (msg.what == 1) {

**int** millis = msg.arg1 % 100;

textView.setText(msg.arg1 / 100 + ":"

+ ((millis + "").length() != 2 ? "0"

+ millis : millis));

} **else** {

button.setEnabled(**true**);

editText.setEnabled(**true**);

}

}

};

*Chú ý:* Để nhận được dữ liệu thì ta phải thông qua một lớp “Message”. Chúng ta muốn gửi dữ liệu hoặc nhận dữ liệu đều phải thông qua lớp này, lớp “Message” cung cấp cho 4 thuộc tính: what (kiểu int), arg1 (kiểu int), arg2 (kiểu int) và obj (kiểu Object).

1. Tạo biến “numb” kiểu int dùng để lấy số nguyên dương từ người dùng nhập. Sau đó bắt sự kiện khi người dùng ấn Button và viết xử lý:

**public** **void** onClick(View v) {

numb = Integer.*parseInt*

(editText.getText().toString()) \* 100;

**new** Thread() {

**public** **void** run() {

// 0 là trạng bắt đầu

handler.obtainMessage(0).sendToTarget();

**while** (**true**) {

**try** {

**if** (numb == 0) {

// 2 là trạng thái kết thúc

handler.obtainMessage(2)

.sendToTarget();

**return**;

}

Thread.*sleep*(10);

numb = numb - 1;

// 1 là trạng thái đang chạy

handler.obtainMessage(1,

numb, 0).sendToTarget();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}

// Dùng start() để chạy ngầm, không làm đơ ứng dụng khi

// đang xử lý tiến trình bên trong

}.start();

}

1. SERVICES – BROADCAST RECEIVER VÀ NOTIFICATION

|  |  |
| --- | --- |
| **🖰** | Mục tiêu:  Hiểu được vai trò của Services, BroadcastReceiver, Notification.  Xây dựng chế độ làm việc ngầm cho ứng dụng. Thiết lập nhận và gửi thông báo cho ứng dụng. |

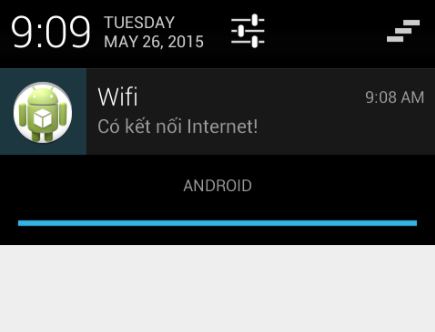
* 1. Cách dùng Service, Broadcast Receiver và Notification

**Đề bài:**

* Kiểm tra có kết nối Wifi.

*Chi tiết yêu cầu:* Gửi 1 Intent đi và bắt lại bằng Broadcast, sau đó hiển thị thông báo lên Notification. Thông báo khi có wifi và khi mất kết nối wifi trên Notification.

*Kết quả khi chạy chương trình:*



*Hình 7.1*

Gợi ý thực hiện:

* Tạo BroadcastReceiver để nhận kết quả trả về.
* Dùng IntentFilter để gửi yêu cầu nhận cho biết tình trạng kết nối của Wifi.
* Đăng ký BroadcastReceiver thông qua phương thức “registerReceiver”.

Hướng dẫn chi tiết:

1. Tạo BroadcastReceiver:

BroadcastReceiver receiver = **new** BroadcastReceiver() {

@Override

**public** **void** onReceive(Context context,

Intent intent) {

ConnectivityManager connectivityManager =

(ConnectivityManager)

context.getSystemService

(*CONNECTIVITY\_SERVICE*);

NetworkInfo networkInfo =

connectivityManager

.getActiveNetworkInfo();

**if** (networkInfo == **null**) {

displayNotification

("Không có kết nối Internet!");

} **else** {

displayNotification

("Có kết nối Internet!");

}

}

};

1. Tạo IntentFilter và đăng ký:

@Override

**protected** **void** onResume() {

**super**.onResume();

IntentFilter intentFilter = **new** IntentFilter(

ConnectivityManager.*CONNECTIVITY\_ACTION*);

registerReceiver(receiver, intentFilter);

}

1. Tạo NotificationManager để hiện thông báo trên thanh Notification:

**private** NotificationManager mNotificationManager;

**private** **int** notificationID = 100;

**protected** **void** displayNotification(String content) {

NotificationCompat.Builder mBuilder =

**new** NotificationCompat.Builder(**this**);

mBuilder.setContentTitle("Wifi");

mBuilder.setContentText(content);

mBuilder.setSmallIcon(R.drawable.*ic\_launcher*);

/\*

\* Thiết lập một hành động nào đó cho thanh

\* Notification. Có thể là hành động yêu cầu

\* đi thẳng đến Activity của ứng dụng để xử lý.

\* Để xử lý được hành động này thì cần thông qua

\* đối tượng “PendingIntent” có chứa Intent khởi

\* động Activity của ứng dụng. “PendingIntent”

\* sẽ giúp các bạn thực hiện hành động của các

\* bạn yêu cầu trên ứng dụng.

\*/

Intent resultIntent = **new** Intent(**this**,

NotificationView.**class**);

resultIntent.putExtra("Wifi", content);

TaskStackBuilder stackBuilder =

TaskStackBuilder.*create*(**this**);

stackBuilder.addParentStack

(NotificationView.**class**);

stackBuilder.addNextIntent(resultIntent);

PendingIntent resultPendingIntent =

stackBuilder.getPendingIntent(0,

PendingIntent.*FLAG\_UPDATE\_CURRENT*);

mBuilder.setContentIntent(resultPendingIntent);

mNotificationManager = (NotificationManager)

getSystemService(Context.*NOTIFICATION\_SERVICE*);

/\*

\* phương thức “NotificationManager.notify()”

\* dùng để gửi Notification vừa được thiết lập.

\*/

mNotificationManager.notify(notificationID,

mBuilder.build());

}

*Giải thích :*

* “setContentTitle” là cài đặt tiêu đề cho Notification, “setContentText” là cài đặt nội dung và “setSmallIcon” là cài đặt tấm hình trên dòng thông báo.

1. Activity hiển thị nội dung chi tiết khi chọn vào dòng thông báo:

**public** **class** NotificationView **extends** Activity{

@Override

**protected** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*notification*);

((TextView) findViewById(R.id.*textView*))

.setText(getIntent().getStringExtra("Wifi"));

}

}