BÀI 1. QUẢN LÝ DỮ LIỆU VỚI CONTENT PROVIDER

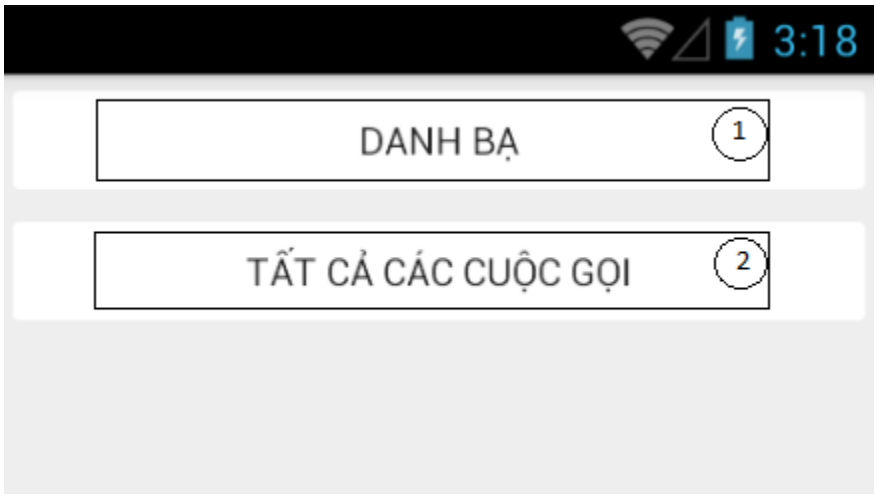
✓ **Mục tiêu:**

- *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.*

Bài tập 1.1. Bài 1: Đọc danh bạ và lịch sử các cuộc gọi điện thoại.

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

Giao diện:

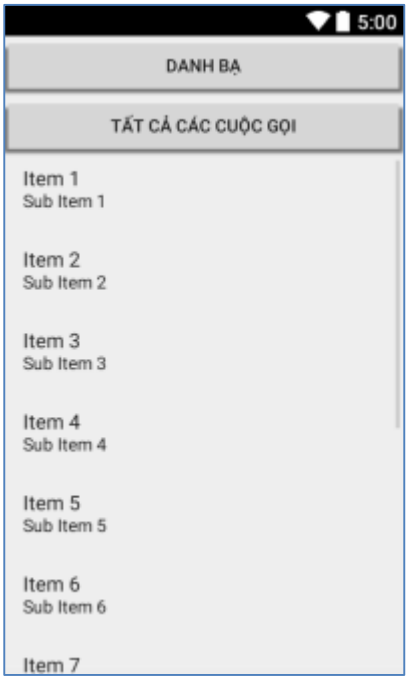


Nhấn vào Button "DANH BẠ" (1) hiển thị danh bạ điện thoại bên dưới:

+

Hướng dẫn chi tiết:

Bước 1: Thiết kế giao diện trong file "activity_main.xml" như sau:



Hình 1.1

Bước 2: Viết xử lý trong file "MainActivity.java".

Bước 2. 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);
  }
```

Bước 2. 2: 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);
  }
```

Bước 3: Xin quyền trong file "AndroidManifest.xml":

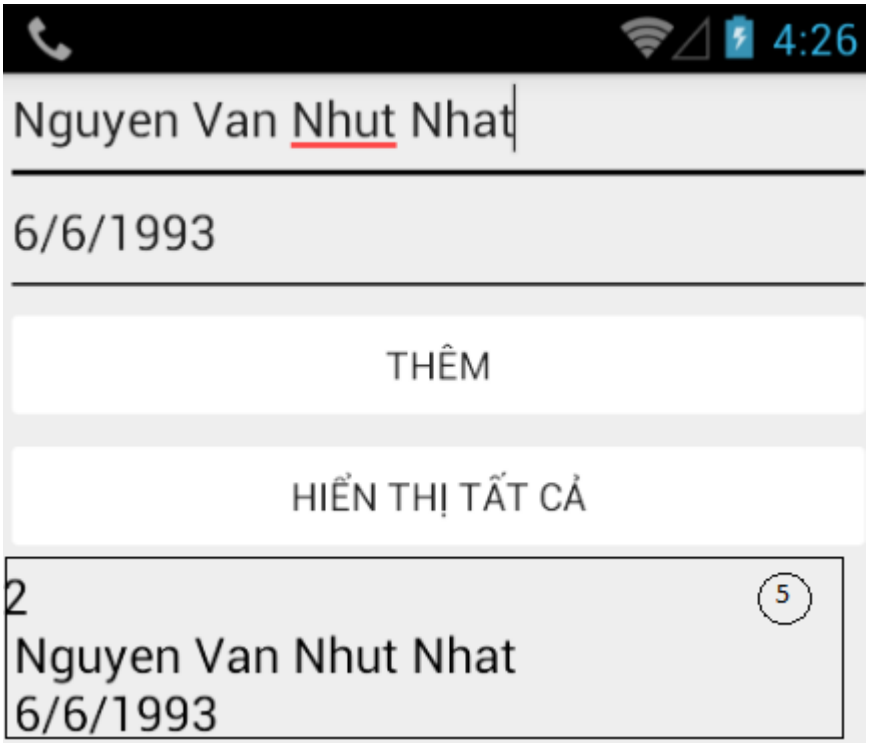
```
    <uses-permission android:name="android.permission.READ_CONTACTS" />
    <uses-permission android:name="android.permission.READ_CALL_LOG" />
```

Bài tập 1.2. 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:

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.



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.

Hướng dẫn chi tiết:

Bước 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";
```

Bước 2: Tạo ContentProvider.

Bước 2. 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.

Bước 2. 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.

Bước 2. 3: 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;
}
```

Bước 2. 4: 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;
}
```

Bước 2. 5: 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);
     }
}
```

Bước 2. 6: 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);
}
```

Bước 2. 7: 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;
    }
```

Bước 2. 8: 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;
```

```
}
```

Bước 3: Thiết kế giao diện cho file "activity_main.xml":

Hình 1.2

Bước 4: Viết xử lý trong file "MainActivity.java".

Bước 4. 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();
}
```

Bước 4. 2: 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);
}
```