



LẬP TRÌNH DI ĐỘNG

Bài 7: SQLite và Content Provider



Nhắc lại bài trước

- Nguyên tắc lưu trữ trong android: theo phân quyền của Linux, hỗ trợ nhiều loại lưu trữ với mục đích khác nhau
 - `MODE_PRIVATE`,
`MODE_WORLD_READABLE` và
`MODE_WORLD_WRITEABLE`
- Các vùng lưu trữ được cấp cho ứng dụng
- Shared preferences và PreferenceActivity
- Các loại lưu trữ: internal, external, cached và trong file apk của ứng dụng
- Làm việc với SQLite



Nội dung

1. Làm việc với SQLite API
 - Tạo/Mở/Đóng CSDL
 - Thực thi câu lệnh SQL
 - CRUD
 - Duyệt các kết quả trả về bởi SELECT
2. SQLiteOpenHelper
3. Kinh nghiệm làm việc với CSDL
4. Content Providers
 - Giới thiệu
 - (tự học) Sử dụng content providers
 - (tự học) Tự viết content provider



Phần 1

Làm việc với SQLite API



SQLiteDatabase – Tạo/Mở CSDL

```
public static SQLiteDatabase  
openDatabase(String path, CursorFactory  
factory, int flags)
```

```
SQLiteDatabase db =  
SQLiteDatabase.openDatabase("/data/data/<package>/DB.db",null,  
SQLiteDatabase.CREATE_IF_NECESSARY);
```

Flags: OPEN_READWRITE, OPEN_READONLY và
CREATE_IF_NECESSARY



SQLiteDatabase – Tạo/Mở CSDL

```
// filePath is a complete destination of the form
// "/data/data/<namespace>/<databaseName>"
// "/sdcard/<databasename>"
// "mnt/sdcard/<databasename>"
try {
    db = SQLiteDatabase.openDatabase(
        "/data/data/vn.mobipro.sql/myfriendsDB",
        null, SQLiteDatabase.CREATE_IF_NECESSARY);
    db.close();
}
catch (SQLException e) {
    Toast.makeText(this, e.getMessage(), 1).show();
}
```

An SD resident database requires the Manifest to include:

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



SQLiteDatabase – Đóng CSDL

```
public void close()
```

- Sử dụng khi đóng kết nối với CSDL
- CSDL tự đóng khi ứng dụng kết thúc
- Nên đóng CSDL ngay khi không dùng nữa

```
db = SQLiteDatabase.openDatabase(...);  
// thao tác với CSDL  
... db.close();
```

SQLiteDatabase – Thực thi SQL



- Sử dụng khi muốn thực thi một câu lệnh SQL và không quan tâm tới kết quả trả về
- Không thực hiện được “multiple statements” SQL
- Tài liệu của Google nói không thực hiện những câu SQL có trả về kết quả (SELECT, INSERT, UPDATE,...)

```
db.execSQL("CREATE TABLE      Book (BookID  
INTEGER PRIMARY KEY AUTOINCREMENT,  
BookName  TEXT);");  
db.execSQL("INSERT      INTO Book(BookName)  
VALUES ('Test      EXECSQL')");
```




SQLiteDatabase – Chèn



```
public long insert(String table, String  
nullColumnHack, ContentValues values)
```

- "table": tên bảng muốn insert dữ liệu
- "nullColumnHack": tên cột nào đó nhận giá trị NULL (*dùng trong trường hợp values = null*)
- "values": danh sách các cặp <cột>-<giá trị> sẽ chèn vào dòng mới
- Chú ý: hàm trả về giá trị row ID của dòng vừa chèn vào, nếu không thành công sẽ trả về -1



SQLiteDatabase – Chèn

```
ContentValues cv = new ContentValues();  
cv.put("BookName", "SQLite");  
cv.put("Price", "100");  
db.insert("Book", null, cv);
```

Tablename: Book		
Fieldname	Fieldtype	Fieldconstraint
 BookID	INTEGER	PRIMARY KEY AUTOINCREMENT
 BookName	TEXT	NULL



SQLiteDatabase – Cập nhật

```
public int update(String table,      ContentValues  
                  values, String    whereClause,  String[]  
                  whereArgs)
```

- table: tên bảng muốn update
- values: các cặp key/value – tên cột/giá trị mới muốn cập nhật
- whereClause: điều kiện để dòng được chọn
- whereArgs: mảng các giá trị ứng với whereClause
- Giá trị trả về: số bản ghi được cập nhật



SQLiteDatabase – Cập nhật

```
String [] whereArgs = {"2"};
ContentValues updValues = new ContentValues();
updValues.put("BookName", "ANDROID");
updValues.put("Price", "200");
int recAffected = db.update("Book", updValues,
"BookID=?", whereArgs);
```



SQLiteDatabase - Xóa

```
public int delete(String table, String  
whereClause, String[] whereArgs)
```

- table: tên bảng muốn xóa
- whereClause: điều kiện xóa
- whereArgs: mảng giá trị ứng với whereClause

```
String [] whereArgs = {"2"};
```

```
int recAffected = db.delete("Book",  
"BookID=?", whereArgs);
```

```
// db.delete("Book", "BookID=2", null);
```



SQLiteDatabase - SELECT

```
public Cursor rawQuery(String sql,  
String[] selectionArgs)
```

- sql: câu lệnh truy vấn
- selectionArgs: mảng giá trị các tham số trong câu lệnh sql (nếu có)
- Giá trị trả về: con trỏ đặc biệt hỗ trợ việc lấy dữ liệu và duyệt mảng các giá trị trả về



SQLiteDatabase - SELECT

// dạng đơn giản

```
Cursor c = db.rawQuery("SELECT * FROM  
Book", null);
```

// dạng có tham số

```
StringmySQL = "select count(*) as Total "+ "  
from tblAmigo "+ " whererecID > ?"+ " and  
name = ?";
```

```
String[] args = {"1", "BBB"};
```

```
Cursor c1 = db.rawQuery(mySQL, args);
```



SQLiteDatabase - Cursor

- Cursor trở tới 1 dòng trong kết quả trả về
- Dùng cursor để đọc giá trị trên các cột của dòng đó
- Khởi đầu cursor ở vị trí **before-first**
- Cursor có nhiều phương thức hỗ trợ:
 - **Kiểm tra vị trí hiện tại:** isFirst(), isLast(), isBeforeFirst(), isAfterLast()
 - **Dịch chuyển trong kết quả:** moveToFirst(), moveToLast(), moveToNext(), moveToPrevious(), move(n)
 - **Lấy dữ liệu:** getInt, getString, getFloat, getBlob, getDate,...
 - **Lấy cấu trúc bảng:** getCount, getColumnName, getColumnNames, getColumnIndex, getColumnCount,...



Mã chung khi sử dụng cursor

// thực hiện truy vấn bằng SELECT, kết quả luôn là 1 mảng

// lúc này cs trở tới trước dòng đầu tiên

```
Cursor cs = db.rawQuery("SELECT * FROM Book", null);
```

// dịch chuyển xuống dòng dưới

```
    while (cs.moveToNext()) {
```

// đọc dữ liệu ở cột đầu tiên

```
    int id = cs.getInt(0);
```

// đọc dữ liệu ở cột thứ hai

```
    String book = cs.getString(1);
```

```
    ...
```

```
}
```

// đóng kết quả truy vấn cs.close();

```
...
```



Phần 2

SQLiteOpenHelper



SQLiteOpenHelper

- SQLiteOpenHelper là phương pháp mà Google đề nghị để thống nhất việc quản lý, tạo, truy xuất và cập nhật cơ sở dữ liệu
- SQLiteOpenHelper cần 3 phương thức cơ bản sau để làm việc hiệu quả
 - **Constructor**: cung cấp các tham số cần thiết để làm việc với cơ sở dữ liệu
 - **onCreate()**: phương thức được tự động gọi khi lần đầu tạo file CSDL và tạo các bảng trong CSDL cũng như khởi tạo dữ liệu ban đầu
 - **onUpgrade()**: phương thức được tự động gọi khi nâng cấp CSDL từ các phiên bản cũ



SQLiteOpenHelper

- Các phương thức hữu ích của SQLiteOpenHelper
 - SQLiteDatabase `getReadableDatabase()`: lấy về CSDL ở dạng “chỉ đọc”
 - SQLiteDatabase `getWritableDatabase()`: lấy về CSDL ở dạng “đọc và ghi”
 - String `getDatabaseName()`: lấy tên CSDL
- Không dùng SQLiteOpenHelper cũng không có vấn đề gì, nhưng sử dụng SQLiteOpenHelper thì ta sẽ đỡ công sức viết code hơn, đặc biệt khi cập nhật phiên bản ứng dụng kèm theo nâng cấp CSDL



SQLiteOpenHelper

- Code: kế thừa `SQLiteOpenHelper` và viết lại 3 phương thức cơ bản
- Cách thức sử dụng `SQLiteOpenHelper`:
 - Thay vì mở CSDL trực tiếp, ta mở qua `SQLiteOpenHelper`
 - `SQLiteOpenHelper` kiểm tra xem CSDL đã có hay chưa, nếu chưa có thì sẽ tự động gọi `onCreate` để tạo
 - `SQLiteOpenHelper` kiểm tra phiên bản của CSDL hiện tại so với đề xuất có bằng nhau không, nếu không thì tự động gọi `onUpgrade` (hoặc `onDowngrade`, tùy tình huống)
 - LTV dùng `getReadableDatabase` / `getWritableDatabase` để lấy về CSDL đã sẵn sàng hoạt động



SQLiteOpenHelper - Sample

```
public class DatabaseHandler extends SQLiteOpenHelper {  
  
    // All static variables  
    // Database Version  
    private static final int DATABASE_VERSION = 1;  
  
    // Database Name  
    private static final String DATABASE_NAME = "WorklightVer1";  
  
    // Contacts table name  
    private static final String TABLE_FLAG = "activeFlag";  
  
    // Contacts Table Columns names  
    private static final String KEY_FLAG = "statusFlag";  
    //Hàm khởi tạo  
    public DatabaseHandler(Context context) {  
        super(context, DATABASE_NAME, null, DATABASE_VERSION);  
    }  
}
```



SQLiteOpenHelper - Sample

```
// Tạo Bảng
@Override
public void onCreate(SQLiteDatabase db) {
    String CREATE_CONTACTS_TABLE = "CREATE TABLE " + TABLE_FLAG +
    "("
        + KEY_FLAG + " TEXT DEFAULT \'unactive\'"
    + ")";
    db.execSQL(CREATE_CONTACTS_TABLE);
}

// Upgrading database
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
    // Drop older table if existed
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_FLAG);

    // Create tables again
    onCreate(db);
}
```



SQLiteOpenHelper - Sample

```
/**
 * All CRUD(Create, Read, Update, Delete) Operations
 */
// Adding new contact
void addContact(String str) {
    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put(KEY_FLAG, str); // Contact Name

    // Inserting Row
    db.insert(TABLE_FLAG, null, values);
    db.close(); // Closing database connection
}
```




SQLiteOpenHelper - Sample

```
public class Database extends SQLiteOpenHelper {
    public Database(@Nullable Context context, @Nullable String name, @Nullable
        super(context, name, factory, version);
    }
    //Truy vấn dữ liệu không trả kết quả: create, delete, update, insert
    public void queryData(String sql){
        SQLiteDatabase database=getWritableDatabase();
        database.execSQL(sql);
    }
    //Truy vấn dữ liệu trả kết quả,: select..
    public Cursor getData(String sql){
        SQLiteDatabase database=getReadableDatabase();
        Cursor cursor=database.rawQuery(sql, selectionArgs: null);
        return cursor;
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    }
}
```



SQLiteOpenHelper - Sample

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    String sql_cre_tab="create table if not exists book(id integer primary key autoincrement, ten va:  
    database.queryData(sql_cre_tab);  
    String sql_ins="insert into book(id, ten) values(null, 'Sách PHP')";  
    database.queryData(sql_ins);  
    lvSach=findViewById(R.id.lvSach);  
    //Lấy dữ liệu ra  
    Cursor cursor=database.getData( sql: "select * from book");  
    while (cursor.moveToNext()) {  
        int id=cursor.getInt( columnIndex: 0);  
        String ten=cursor.getString( columnIndex: 1);  
        Sach sach=new Sach(id, ten);  
        arrayList.add(sach);  
    }  
    adapter=new SachAdapter( context: MainActivity.this, R.layout.dong_sach,arrayList);  
    lvSach.setAdapter(adapter);  
}
```



Phần 3

Kinh nghiệm làm việc với CSDL

Kinh nghiệm làm việc với CSDL



- Viết class mô tả đối tượng cần thao tác (không liên quan gì tới database), tạm gọi là các model
- Dùng SQLiteOpenHelper xử lý ở cấp độ database
- Viết class bọc ngoài các table, cung cấp các thao tác làm việc giữa tables với model (data provider)
- Sử dụng các data provider để làm việc bất kể khi nào dùng tới database, không trực tiếp làm việc với database trong mọi trường hợp
- Các string SQL nên đặt ngoài resource hoặc có thể cập nhật online là tốt nhất



Thực Hành

- Xây dựng ứng dụng quản lý sách với CSDL Books trong bài học trước
- Ứng dụng gồm các chức năng chính:
 - Thêm sách mới
 - Hiển thị danh sách các sách ra ListView
 - Sửa sách
 - Xóa sách
 - Tìm kiếm sách theo các trường hoặc theo các tiêu chuẩn
 - Upgrade lên DB lên phiên bản 2: bổ sung trường tags gồm danh sách các từ khóa mô tả về sách
 - Cập nhật chức năng tìm kiếm theo tags



Phần 4

Content providers



Content Providers

- Là thành phần cơ bản của Android OS
- Chuẩn chia sẻ dữ liệu cho các ứng dụng khác trên Android
 - Content Providers (thường gọi tắt là providers) là cách thức để chia sẻ dữ liệu
 - Không phải là nơi chứa dữ liệu
 - Nhiều dữ liệu hệ thống cung cấp providers để có thể truy xuất đến chúng như: Calendar, Contact, CallLog, MediaStore,...
 - Ứng dụng của bên thứ 3 có thể tự viết provider để cung cấp dữ liệu cho các ứng dụng khác

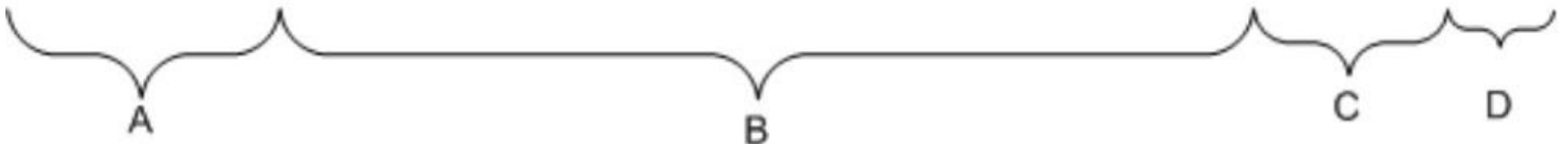


Content Providers

- Content Providers cung cấp một đối tượng con trỏ (cursor) có thể dễ dàng lấy được bất cứ dữ liệu lưu trữ nào thông qua URI chính xác dữ liệu đó
- URI: là quy tắc mô tả một đối tượng (bản thân các giao thức http, ftp, email, skype, torrent,... cũng dùng URI để làm việc)

URI

`content://com.example.transportationprovider/trains/122`





Cấu trúc URI của providers

- **Phần A:** chỉ ra URI được điều khiển bởi providers (luôn có dạng **content://**)
- **Phần B:** chỉ đến nơi lưu trữ dữ liệu, là tên package + class cung cấp content
- **Phần C:** chỉ ra loại dữ liệu. Chẳng hạn như dữ liệu contact, dữ liệu SMS,... thường cũng là tên của một table trong CSDL của provider (không nhất thiết)
- **Phần D:** tham số để thao tác dữ liệu, có thể coi phần này như là ID của row trong table hoặc một dữ liệu nào đó dùng để truy vấn



Sử dụng providers

- Phương thức `getContentResolver()` của context cho phép làm việc với với các provider
 - `getContentResolver()` trả về đối tượng `ContentResolver`
 - `getContentResolver().query(Uri uri)` trả về đối tượng `Cursor`
- Android cung cấp sẵn nhiều providers về hệ thống
 - Tập hợp này nằm trong package `android.provider`
 - Cần khai báo permission trong `AndroidManifest.xml` trước khi muốn truy xuất đến những tài nguyên
 - Sử dụng `CursorLoader` trong tình huống truy vấn kéo dài



Ví dụ

- Bài tập: viết ứng dụng đọc contacts trong thiết bị
- Kiến thức:
 - URI: “content://com.android.contacts/contacts/”
 - Hoặc hằng số: `ContactsContract.Contacts.CONTENT_URI`
 - Cách truy vấn thường:

```
Cursor c= getContentResolver().query(uri, null, null, null, null);
```
 - Nếu provider hoạt động quá lâu, truy vấn ở background:

```
CursorLoader loader = new CursorLoader(context, uri, null, null, null, null);  
Cursor c = loader.loadInBackground();
```



Đọc contacts – thiết lập quyền

Android Manifest Permissions

Permissions P U P P Az

U android.permission.READ_CONTACTS (Uses F

Add...

Remove...

Up

Down

Attributes for android.permission.READ_CONTACTS (Uses Permission)

U The [uses-permission](#) tag requests a "**permission**" that the containing package must be granted in order for it to operate correctly.

Name android.permission.READ_CONTACTS ▼

activity_main.xml MainActivity.java sr.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/tv" android:padding="10dp" android:textSize="16sp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

</TextView>
```



Đọc contacts – layout

activity_main.xml ✕ MainActivity.java

```
<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"
    android:gravity="fill_vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <ListView
        android:id="@+id/listView1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true" >

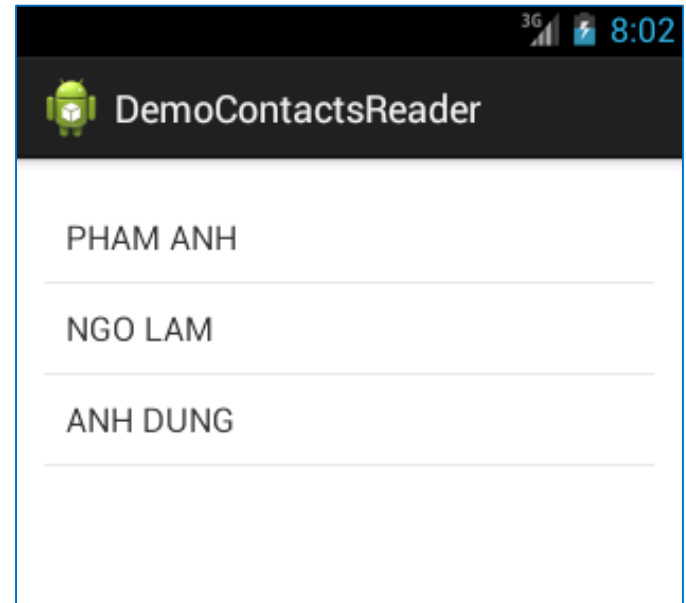
    </ListView>

</RelativeLayout>
```



Đọc contacts – layout

```
public class MainActivity extends Activity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        ListView lv = (ListView) findViewById(R.id.listView1);  
        ArrayList<String> al = new ArrayList<String>();  
        Cursor c = getContentResolver().query(android.provider.ContactsContract.Contacts.CONTENT_URI, null, null, null, null);  
        while(c.moveToNext()){  
            int nameID = c.getColumnIndex(android.provider.ContactsContract.Contacts.DISPLAY_NAME);  
            String name = c.getString(nameID);  
            al.add(name);  
        }  
        ArrayAdapter<String> aa = new ArrayAdapter<String>(this, R.layout.sr, al);  
        lv.setAdapter(aa);  
    }  
}
```





Xây dựng Content Providers

- Lý do: khi chúng ta muốn chia sẻ dữ liệu của ứng dụng cho các ứng dụng khác (đặc biệt với các ứng dụng từ cùng một nhà phát triển)
- Tạo class thừa kế lớp `ContentProvider`
- Viết lại các phương thức:
 - `onCreate()`
 - `query()`
 - Các phương thức `insert`, `delete`, `update` (nếu cần)
- Định nghĩa URI cho Content Provider trong `AndroidManifest.xml`

Viết lớp kế thừa ContentProvider



```
public class BookProvider extends ContentProvider {

    public static final String PROVIDER_NAME = "vn.mobipro.Books";
    public static final Uri CONTENT_URI = Uri.parse("content://" + PROVIDER_NAME + "/books");
    public static final String _ID = "_id";
    public static final String TITLE = "title";
    //Using SQLiteDatabase to store all content provider data
    private SQLiteDatabase bookDB;
    private static final String DATABASE_NAME = "Books";
    private static final String DATABASE_TABLE = "titles";
    private static final int DATABASE_VERSION = 1;
    private static final String DATABASE_CREATE =
        "create table " + DATABASE_TABLE +
        " (_id integer primary key autoincrement, "
        + "title text not null);";
}
```


Viết lớp kế thừa ContentProvider



```
@Override
```

```
public Uri insert(Uri uri, ContentValues values) {  
    long rowID = bookDB.insert(DATABASE_TABLE, "", values);  
    if(rowID > 0)  
    {  
        Uri mUri = ContentUris.withAppendedId(CONTENT_URI, rowID);  
        getContext().getContentResolver().notifyChange(mUri, null);  
        return mUri;  
    }  
    throw new SQLException("Failed to insert new row into " + uri);  
}
```

```
@Override
```

```
public boolean onCreate() {  
  
    Context context = getContext();  
    DatabaseHelper dbHelper = new DatabaseHelper(context);  
    bookDB = dbHelper.getWritableDatabase();  
    return (bookDB == null) ? false : true;  
}
```

Viết lớp kế thừa ContentProvider



```
@Override
```

```
public Cursor query(Uri uri, String[] projection, String selection,  
    String[] selectionArgs, String sortOrder) {  
    SQLiteQueryBuilder sqlBuilder = new SQLiteQueryBuilder();  
    sqlBuilder.setTables(DATABASE_TABLE);  
    Cursor c = sqlBuilder.query(bookDB, projection, selection, selectionArgs, null,  
        null, null);  
    c.setNotificationUri(getContext().getContentResolver(), uri);  
    return c;  
}
```

```
public int delete(Uri uri, String selection, String[] selectionArgs) {
```

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

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



Đăng kí ở AndroidManifest.xml

```
<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <activity
        android:name="vn.mobipro.democontactsreader.MainActivity"
        android:label="@string/app_name" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name="vn.mobipro.democontactsreader.ProviderActivity">|</activity>

    <provider android:name = "BookProvider"
        android:authorities="vn.mobipro.Books" />
</application>
```



Khai thác provider mới

```
public void addBook(String title)
{
    ContentValues values = new ContentValues();
    values.put(BookProvider.TITLE, title);
    Uri uriInsert = getContentResolver().insert(BookProvider.CONTENT_URI, values);
    if(uriInsert != null)
    {
        Toast.makeText(this, "Book's added", Toast.LENGTH_SHORT).show();
    }
    Log.d(getClass().getSimpleName(), uriInsert.toString());
}

public void getAllBooks()
{
    Uri uriGetListTitles = Uri.parse("content://vn.mobipro.Books/books");
    Cursor c = getContentResolver().query(uriGetListTitles, null, null, null, null);
    if(c.moveToFirst()){
        do{
            String bookRecord = "ID = " + c.getString(c.getColumnIndex(BookProvider._ID))
                                + " Title = " +
                                c.getString(c.getColumnIndex(BookProvider.TITLE));
            Toast.makeText(this, bookRecord, Toast.LENGTH_LONG).show();
        }while(c.moveToNext());
    }
}
```