Phát triển ứng dụng Smartphone

Tài liệu lưu hành nội bộ

Đây là tài liệu tham khảo sử dụng trong môn học Lập trình ứng dụng Smartphone – Android được tổng hợp, biên soạn từ nhiều nguồn bởi các thành viên của Nhóm nghiên cứu và ứng dụng công nghệ A106-Đại học Hoa Sen.

Phát triển ứng dụng Smartphone

Phần 03: Lưu trữ dữ liệu

Lê Đức Huy

Email: leduchuy89vn@gmail.com

Mục lục

1 Lưu trữ dạng file

Các đơn giản nhất để lưu trữ dữ liệu của ứng dụng là ghi xuống file. Tuy nhiên cách này hạn chế rất nhiều do các phương thức đọc, ghi dữ liệu rất thô sơ, đơn giản. Phần dưới đây thể hiện cách đọc ghi file đơn giản.

1.1 Sử dụng bộ nhớ trong

Bạn có thể lưu tệp tin trực tiếp bộ nhớ trong của ứng dụng. Mặc định thì tệp tin này sẽ thuộc về ứng dụng tạo ra nó và các ứng dụng khác không thể truy xuất đến nó.

1.1.1 Ghi dữ liệu xuống tệp tin

Để ghi dữ liệu xuống tệp tin ta thực hiện các bước:

01. Gọi phương thức FileOutputStream openFileOutput (String name, int mode) để tạo một đối tượng FileOutputStream dùng để ghi dữ liệu lên tệp tin.
    Phương thức này nhận vào ai tham số:
    - Tên tệp tin.
    - Chế độ ghi. Có ba chế độ:
        - MODE_PRIVATE: Tệp chỉ có thể được truy xuất bên trong ứng dụng tạo ra nó.
        - MODE_WORLD_READABLE: Tệp có thể được đọc bởi các ứng dụng khác.
        - MODE_WORLD_WRITEABLE: Tệp có thể được đọc và ghi bởi các ứng dụng khác.
02. Gọi phương thức FileOutputStream.write(…) để ghi dữ liệu xuống file.
03. Gọi phương thức FileOutputStream.close() để đóng luồng ghi dữ liệu xuống file.

VD minh họa:

```
String FILENAME = "hello_file";
String string = "hello world!";

FileOutputStream fos = openFileOutput(FILENAME,
Context.MODE_PRIVATE);
fos.write(string.getBytes());
fos.close();
```

1.1.2 Đọc dữ liệu từ tệp tin

Để đọc dữ liệu từ tệp tin ta thực hiện các bước:

01. Gọi phương thức public abstract **FileInputStream** openFileInput(String name) tạo luồng đọc dữ liệu từ file. Phương thức này nhận vào một tham số là tên file cần đọc.

02. Gọi phương thức FileInputStream.read(…) để đọc dữ liệu từ file.
03. Gọi phương thức FileInputStream.close() để đóng luồng đọc dữ liệu từ file.

VD minh họa:

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

FileInputStream fis=null;
try {

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

     string = new String(buffer);

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

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

1.2 Sử dụng tệp tin cache

Đôi lúc bạn cũng muốn lưu trữ tệp tin vào thư mục cache thay vì lưu trữ vĩnh viễn. Tuy nhiên nếu lưu trữ tệp tin trong thư mục cache thì khi thiết bị thiếu bộ nhớ trong thì hệ thống sẽ tự xóa các tệp tin này đi.

Việc lưu trữ tệp tin cũng tương tự như mục trên. Ta sẽ sử dụng phương thức public abstract **File** getCacheDir () để lấy về thư mục cache lưu trữ dữ liệu của ứng dụng. Thư mục này thường là: data/data/niit.android/cache trong đó niit.android là tên package của ứng dụng.

Ví dụ:

```
File file  = getCacheDir();

String cacheDir = file.getPath();
File downloadingMediaFile = new
File(cacheDir,"downloadingMedia.dat");
FileOutputStream out = null;
```

```
try {
     out = new FileOutputStream(downloadingMediaFile);
     out.write(str.getBytes());
} catch (Exception e) {
     e.printStackTrace();
}
finally{
     try {
          out.close();
     } catch (Exception e) {
          e.printStackTrace();
     }
}
```

Ví dụ minh họa trên sẽ gán File file bằng đối tượng trả về bởi phương thức getCacheDir(). Đây là đối tượng trỏ đến thư mục cache của ứng dụng.

Thực hiện lệnh String cacheDir = file.getPath(); để lấy về chuỗi đường dẫn tuyệt đối trỏ đến thư mục cache.

Lệnh File downloadingMediaFile = new File(cacheDir,"downloadingMedia.dat"); tạo đối tượng File trỏ đến tệp tin downloadingMedia.dat trong thư mục vừa nhắc đến ở trên.

Sau đó tiến hành tạo FileOutputStream như ví dụ ở mục trên để đọc và ghi tệp tin.

1.3 Sử dụng bộ nhớ ngoài

Ngoài việc lưu trữ tệp tin xuống bộ nhớ trong của thiết bị ta còn có thể lưu trữ xuống bộ nhớ ngoài(Thẻ SD). Bộ nhớ ngoài này là thiết bị lưu trữ có thể tháo rời nên ta cần tiến hành kiểm tra trạng thái của bộ nhớ ngoài(Đang cắm vào máy tính, đã tháo ra…) trước khi đọc và ghi dữ liệu lên nó. Ta tiến hành như sau:

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

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

Nếu bạn sử dụng API lever 8 trở lên thì bạn có thể gọi phương thức public abstract **File** getExternalFilesDir (String type) để lấy về đối tượng trỏ đến thư mục lưu trữ trên bộ nhớ ngoài (Thẻ nhớ SD) của thiết bị. Thông số String type là kiểu thư mục muốn truy xuất:

- DIRECTORY_ALARMS: Thư mục chứa các tệp tin âm thanh dùng làm báo thức.
- DIRECTORY_DCIM: Thư mục chứa các tệp tin hình ảnh và video được ghi lại bởi thiết bị.
- DIRECTORY_DOWNLOADS: Thư mục chứa các tệp tin được tải về.
- DIRECTORY_MOVIES: Thư mục chứa các tệp tin phim.
- DIRECTORY_MUSIC: Thư mục chứa các tệp tin âm nhạc.
- DIRECTORY_NOTIFICATIONS: Thư mục chứa các tệp tin âm thanh dùng làm âm báo notification.
- DIRECTORY_PICTURES: Thư mục chứa các tệp tin hình ảnh.
- DIRECTORY_PODCASTS: Thư mục chứa các tệp tin padcast.
- DIRECTORY_RINGTONES: Thư mục chứa các tệp tin dùng làm nhạc chuông.

Nếu bạn sử dụng API 7 trở xuống ta có thể sử dụng phương thức: public static **File** getExternalStorageDirectory () để tạo đối tượng File trỏ đến thư mục gốc của bộ nhớ ngoài. Ta sẽ cộng thêm vào đường dẫn này để chỉ đến thư mục cần lưu trữ dữ liệu:

- Alarms: Thư mục chứa các tệp tin âm thanh dùng làm báo thức.
- Download: Thư mục chứa các tệp tin được tải về.
- Movies: Thư mục chứa các tệp tin phim (Bao gồm cả các video quay bằng camera).
- Music: Thư mục chứa các tệp tin âm nhạc.
- Notifications: Thư mục chứa các tệp tin âm thanh dùng làm âm báo notification.
- Pictures: Thư mục chứa các tệp tin hình ảnh (Bao gồm cả hình ảnh chụp từ camera).
- Podcasts: Thư mục chứa các tệp tin padcast.
- Ringtones: Thư mục chứa các tệp tin dùng làm nhạc chuông.

Sau khi có được đối tượng File ta có thể tiếp tục làm việc như ở các ví dụ trên.

1.4 Một số phương thức hữu dụng:

public abstract **File** getFilesDir (): Lấy đường dẫn tuyệt đối đến thư mục hệ thống nơi tệp tin được lưu trữ.

public abstract **File** getDir (String name, int mode): Tạo mới (hoặc mở nếu nó đã tồn tại) một thư mục bên trong bộ nhớ trong của ứng dụng.

public abstract **boolean** deleteFile (String name): Xóa một tệp tin trong bộ nhớ trong.

public abstract **String**[] fileList (): Trả về danh sách các tệp tên đã được ghi xuống bộ nhớ trong của ứng dụng.

2 Android Preference

Khi làm việc với bất kì ứng dụng nào trên máy tính ta đã quen với khái niệm "Preferences". Một ví dụ điển hình là khi sử dụng trình duyệt web Firefox, một trong những thay đổi mà người dùng hay sử dụng nhất là thiết lập lại trang chủ. Những thông số thiết lập này được lưu trữ trong Preferences. Và nền tảng Android cũng hỗ trợ khái niệm này, tuy nhiên khái niệm Preferences trên Android có khác đôi chút so với khái niệm nguyên thủy của nó trên ứng dụng desktop. Preferences trên Android sẽ được lưu trữ trên tệp tin /data/data/[PACKAGE_NAME]/shared_prefs/[PREFERENCE_NAME].xml. Android hỗ trợ đối tượng SharedPreferences dùng để đọc và ghi dữ liệu dạng key-value lên file xml kể trên. Bạn có thể dùng SharedPreferences để lưu trữ và truy xuất mọi loại kiểu dữ liệu như: boolean, float, int, long và string. Để lấy một đối tượng SharedPreferences trên ứng dụng ta có thể sử dụng một trong hai phương thức sau:

- *public abstract **SharedPreferences** getSharedPreferences (String name, int mode):*
  Phương thức này sẽ lấy về đối tượng SharedPreferences để đọc ghi dữ liệu lên file xml với tên được chỉ định bằng tham số truyền vào. Tham số int mode trong phương thức trên dùng để thiết lập quyền truy xuất đến file xml mà đối tượng SharedPreferences tham chiếu đến. Có ba loại:
    - MODE_PRIVATE: Tệp xml mà đối tượng SharedPreferences chỉ đến chỉ có thể được truy xuất bên trong ứng dụng tạo ra nó.
    - MODE_WORLD_READABLE: Tệp xml mà đối tượng SharedPreferences có thể được đọc bởi các ứng dụng khác.
    - MODE_WORLD_WRITEABLE: Tệp xml mà đối tượng SharedPreferences có thể được đọc và ghi bởi các ứng dụng khác.

  Ở lần đầu tiên tạo ra đối tượng SharedPreference nếu file xml dùng để lưu trữ dữ liệu chưa tồn tại thì hệ thống sẽ tự tạo file xml với tên chỉ định ở tham số trên.

- *public **SharedPreferences** getPreferences (int mode):*
  Phương thức này sẽ lấy về đối tượng SharedPreferences để đọc ghi dữ liệu lên file xml dùng lưu trữ Preferences của Activity hiện tại. Phương thức này sẽ gọi lại phương thức *getSharedPreferences(...)* kể trên với tham số String name là tên của Activity hiện tại. Tham số int mode trong phương thức trên dùng để thiết lập quyền truy xuất đến file xml mà đối tượng SharedPreferences tham chiếu đến. Nó tương tự như tham số int mode ở mục

trên. Ở lần đầu tiên tạo ra đối tượng SharedPreference nếu file xml dùng để lưu trữ dữ liệu chưa tồn tại thì hệ thống sẽ tự tạo file xml với tên là tên của Activity.

2.1 Ghi dữ liệu

1. Gọi phương thức SharedPreferences.edit() để lấy về một đối tượng SharedPreferences.Editor. Ta sẽ sử dụng đối tượng này để ghi dữ liệu xuống file xml.
2. Thêm dữ liệu vào file xml bằng cách gọi các phương thức SharedPreferences.Editor.putBoolean(), SharedPreferences.Editor.putString().
3. Gọi lệnh SharedPreferences.comit() để hoàn tất việc thay đổi nội dung và ghi xuống file xml.

2.2 Đọc dữ liệu

Để đọc dữ liệu ta sử dụng phương thức SharedPreferences.getBoolean() hoặc SharedPreferences.getString().

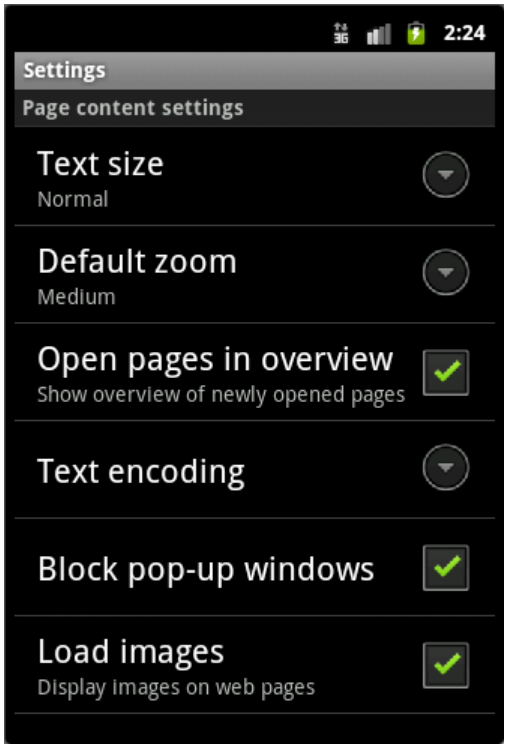
Dưới đây là một ví dụ minh họa cách đọc và ghi dữ liệu bằng Preferences:

```
public class PreferencesExample extends Activity {
	public static final String PREFS_NAME = "MyPrefsFile";

	@Override
	protected void onCreate(Bundle state){
		super.onCreate(state);
		. . .

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

	@Override
	protected void onStop(){
		super.onStop();
		SharedPreferences settings =
		getSharedPreferences(PREFS_NAME, 0);
		SharedPreferences.Editor editor = settings.edit();
		editor.putBoolean("silentMode", mSilentMode);

		// Commit
		editor.commit();
	}
```

```
}
```

2.3 Làm việc với PreferenceActivity

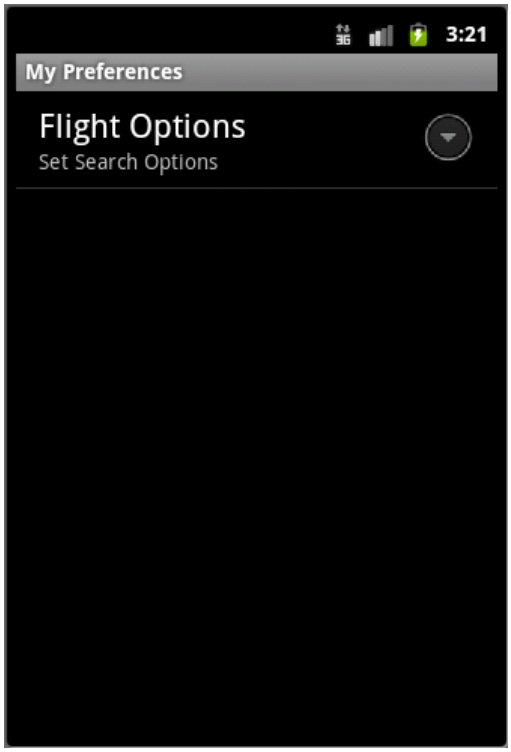
Ứng dụng phổ biến nhất của Preference hiển nhiên là dùng để tạo một trang Tùy chỉnh(Settings) cho ứng dụng như hình dưới đây:



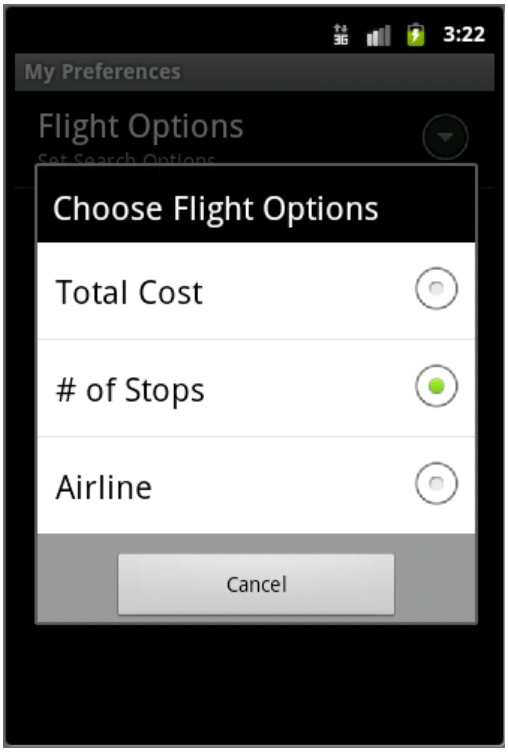
Theo những gì đã tìm hiểu ở phần trên thì ta đã có được một đối tượng SharedPreferences dùng để đọc ghi dữ liệu lên file xml. Tuy nhiên để hiện thực một trang Tùy chỉnh như trên ta cần phải dựng một Activity để hiển thị các tùy chọn và cho phép người dùng chỉnh sửa các tùy chọn đó. Tuy nhiên, trên Android ta không cần phải tự dựng một Activity để hiển thị các tùy chọn như ở trên, thay vào đó Android cung cấp sẵn cho ta một lớp PreferenceActivity dùng để hiển thị các tùy chọn cũng như cho phép người dùng thay đổi các tùy chọn đó. Sau khi thay đổi các giá trị của tùy chọn, giá trị này sẽ được lưu trữ vào Preferences. Sau đó ta có thể sử dụng SharedPreferences để đọc các giá trị này lên.

Để minh họa cách tạo một trang Tùy chỉnh ta tiến hành hiện thực ví dụ sau:

Giả sử xây dựng một ứng dụng cho phép xem thông tin về các chuyến bay. Thông tin này sẽ được liệt kê dưới dạng danh sách. Có nhiều cách sắp xếp thứ tự của các chuyến bay như: Sắp xếp theo giá vé, theo số trạm dừng ít nhất hay theo thứ tự các hãng hàng không. Trên ứng dụng có một trang "Tùy chỉnh" cho phép thay đổi cách sắp xếp các chuyến bay. Trang "Tùy chỉnh" có một thông số duy nhất là "Flight Options". Khi nhấn vô "Flight Options" ứng dụng mở lên một danh sách cho phép lựa một trong ba cách sắp xếp cách chuyến bay.



Hình 1 Giao diện trang tùy chỉnh



Hình 2 – Giao diện lựa chọn giá trị tùy chọn

Để tạo một trang giao diện như hình đầu ta tiến hành tạo mới một file flightoptions.xml với nội dung như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:key="flight_option_preference"
    android:title="@string/prefTitle"
    android:summary="@string/prefSummary">

     <ListPreference
          android:key="@string/selected_flight_sort_option"
          android:title="@string/listTitle"
          android:summary="@string/listSummary"
          android:entries="@array/flight_sort_options"
          android:entryValues="@array/flight_sort_options_values
     "
          android:dialogTitle="@string/dialogTitle"
          android:defaultValue="@string/flight_sort_option_defau
     lt_value" />
```

```
</PreferenceScreen>
```

Đây là file chứa các thẻ xml đánh dấu trang giao diện như ở hình trang tùy chọn ở trên. Khi trang "Tùy chọn" trên được mở ra. Hệ thống sẽ đọc lên file xml kể trên để tạo thành giao diện như hình. Trong đó thẻ gốc PreferenceScreen đại diện cho màn hình trang tùy chọn, trên đó sẽ có các tùy chọn. Trên trang tùy chọn trên chỉ có duy nhất một tùy chọn là "Flight Option". Để hiện thực tùy chọn này ta sử dụng thẻ ListPreference. Thẻ này cho phép định nghĩa một tùy chọn cho phép lựa chọn một giá trị từ một danh sách các giá trị cho trước. Ở đây tùy chọn "Flight Option" sẽ có giá trị là: Total cost, # of Stops hoặc Airline. Người dùng sẽ được phép chọn một trong ba cách sắp xếp các chuyến bay kể trên.

Thẻ ListPreference sẽ có một số thuộc tính như sau:

Thuộc tính	Mô tả
android:key	Thuộc tính khóa của tùy chỉnh. Thuộc tính này sẽ dùng để truy xuất giá trị của tùy chọn sau khi giá trị của nó được lưu trữ vào file xml.
android:title	Tiêu đề của tùy chỉnh.
android:summary	Mô tả tóm tắt của tùy chỉnh.
android:entries	Tập nhãn của các mục có thể được gán cho tùy chỉnh.
android:entryValues	Tập giá trị của các mục có thể được gán cho tùy chỉnh.
android:dialogTitle	Tiêu đề của hộp thoại hiển thị danh sách các giá trị của tùy chọn để người dùng lựa chọn.
android:defaultValue	Giá trị mặc định của tùy chọn.

Xem lại ví dụ kể trên ta thấy có hai thuộc tính đáng quan tâm nhất là android:entries và android:entryValues. Ở file flightoptions.xml giá trị của hai thuộc tính này lần lượt là:

android:entries="@array/flight_sort_options"
android:entryValues="@array/flight_sort_options_values"

Giá trị của hai thuộc tính trên là hai mảng được định nghĩa trong một file values. Để ví dụ mẫu trên có thể hoạt động ta tiến hành bổ xung file res/values/arrays.xml với nội dung như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<!-- This file is /res/values/arrays.xml -->
<resources>
     <string-array name="flight_sort_options">
          <item>Total Cost</item>
          <item># of Stops</item>
          <item>Airline</item>
     </string-array>
     <string-array name="flight_sort_options_values">
          <item>0</item>
```

```
            <item>1</item>
            <item>2</item>
        </string-array>
</resources>
```

File xml này khai báo hai mảng kiểu string là **flight_sort_options** và **flight_sort_options_values**. Mảng flight_sort_options tiêu đề của các giá trị có thể được gán cho tùy chỉnh và tập flight_sort_options_values chứa giá trị dùng gán cho tùy chỉnh. Khi trang "Tùy chỉnh" được mở thì sẽ được giao diện như hình "Giao diện trang tùy chỉnh". Khi người dùng nhấn vào "Flight Option" thì ứng dụng mở ra một hộp thoại hiển thị ba sự lựa chọn: Total cost, # of Stops và Airline. Ba tiêu đề này chính là mảng **flight_sort_options**. Khi người dùng chọn một giá trị trong ba giá trị kể trên thì một phần tử tương ứng trong mảng **flight_sort_options_values** sẽ được ghi xuống file xml.

Lưu ý: Bổ sung file res/values/strings.xml với nội dung như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<!-- This file is /res/values/strings.xml -->
<resources>
    <string name="app_name">Preferences Demo</string>
    <string name="prefTitle">My Preferences</string>
    <string name="prefSummary">Set Flight Option
Preferences</string>
    <string name="flight_sort_option_default_value">1</string>
    <string name="dialogTitle">Choose Flight Options</string>
    <string name="listSummary">Set Search Options</string>
    <string name="listTitle">Flight Options</string>
    <string
name="selected_flight_sort_option">selected_flight_sort_option</
string>
    <string name="menu_prefs_title">Settings</string>
    <string name="menu_quit_title">Quit</string>
</resources>
```

Sau khi soạn xong file xml ta tạo một file FlightPreferenceActivity.java có nội dung như sau:

```
package niit.android;

import android.os.Bundle;
import android.preference.PreferenceActivity;
import android.preference.PreferenceManager;

public class FlightPreferenceActivity extends PreferenceActivity
{

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.flightoptions);
```

```
        }

}
```

Lớp FlightPreferenceActivity sẽ tạo ra giao diện trang "Tùy chỉnh" như yêu cầu ta đã đặt ra ban đầu. Do ta không xác định tên của Preferences nên các giá trị tùy chọn kể trên sẽ được lưu vào file: /data/data/[PACKAGE_NAME]/shared_prefs/[PACKAGE_NAME]_preferences.xml. Đây là file chứa giá trị Preferences mặc định của toàn ứng dụng. Nếu muốn ta vẫn có thể chỉ định tên cho Preferences cũng như quyền truy xuất đến các file xml lưu trữ giá trị bằng cách bổ xung các câu lệnh sau:

```
PreferenceManager prefMgr = getPreferenceManager();
prefMgr.setSharedPreferencesName("main");
prefMgr.setSharedPreferencesMode(MODE_WORLD_WRITEABLE);
```

Lưu ý: Do FlightPreferenceActivity cũng là một Activity nên cần bổ xung khai báo cho Activity này trong file manifest.xml nhưng với các Activity khác.

Để sử dụng FlightPreferenceActivity kể trên ta có thể tạo một project có một Activity chính dùng để mở FlightPreferenceActivity. Bổ xung cách file sau vào project mới tạo ra:

```
<?xml version="1.0" encoding="utf-8"?>
<!-- This file is /res/menu/mainmenu.xml -->
<menu
xmlns:android="http://schemas.android.com/apk/res/android">
     <item android:id="@+id/menu_prefs"
     android:title="@string/menu_prefs_title"/>
     <item android:id="@+id/menu_quit"
     android:title="@string/menu_quit_title"/>
</menu>
```

```
<?xml version="1.0" encoding="utf-8"?>
<!-- This file is /res/layout/main.xml -->
<LinearLayout
     xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="vertical"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent">

     <TextView android:text="" android:id="@+id/text1"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     />

</LinearLayout>
```

```
package niit.android;
```

```
import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.widget.TextView;

public class main extends Activity {

	private TextView tv = null;

	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);

		tv = (TextView)findViewById(R.id.text1);

		setOptionText();
	}

	@Override
	public boolean onCreateOptionsMenu(Menu menu)
	{
		MenuInflater inflater = getMenuInflater();
		inflater.inflate(R.menu.mainmenu, menu);
		return true;
	}

	@Override
	public boolean onOptionsItemSelected (MenuItem item)
	{
		if (item.getItemId() == R.id.menu_prefs)
		{
			Intent intent = new Intent().setClass(this,
FlightPreferenceActivity.class);
			this.startActivityForResult(intent, 0);
		}
		else if (item.getItemId() == R.id.menu_quit)
		{
			finish();
		}
		return true;
	}
```

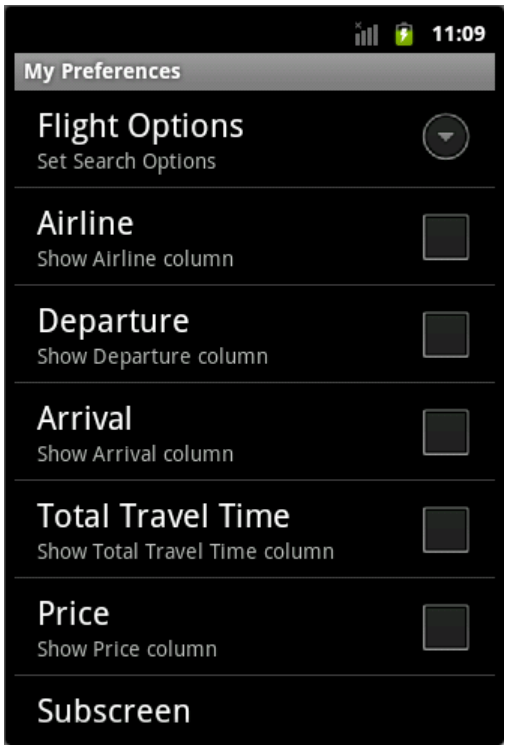
```
	@Override
	public void onActivityResult(int reqCode, int resCode,
Intent data)
	{
		super.onActivityResult(reqCode, resCode, data);
		setOptionText();
	}

	private void setOptionText()
	{
		SharedPreferences prefs =
getSharedPreferences("niit.android_preferences", 0);
		String option =
prefs.getString(this.getResources().getString(R.string.selected_
flight_sort_option),
	this.getResources().getString(R.string.flight_sort_option_d
efault_value));
		String[] optionText =
this.getResources().getStringArray(R.array.flight_sort_options);

		tv.setText("Option value is " + option + " (" +
optionText[Integer.parseInt(option)] + ")");
	}

}
```

Ngoài ListPreference ta còn có 03 loại Preference nữa là **CheckBoxPreference**, **EditTextPreference** và **RingtonePreference**. Tiến hành bổ sung file flightoptions.xml với nội dung như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:key="flight_option_preference"
    android:title="@string/prefTitle"
    android:summary="@string/prefSummary">
     <ListPreference
     android:key="@string/selected_flight_sort_option"
     android:title="@string/listTitle"
     android:summary="@string/listSummary"
     android:entries="@array/flight_sort_options"
     android:entryValues="@array/flight_sort_options_values"
     android:dialogTitle="@string/dialogTitle"
     android:defaultValue="@string/flight_sort_option_default_va
lue" />
     <CheckBoxPreference
             android:key="show_airline_column_pref"
             android:title="Airline"
```

```
            android:summary="Show Airline column" />
    <CheckBoxPreference
            android:key="show_departure_column_pref"
            android:title="Departure"
            android:summary="Show Departure column" />
    <CheckBoxPreference
            android:key="show_arrival_column_pref"
            android:title="Arrival"
            android:summary="Show Arrival column" />
     <CheckBoxPreference
            android:key="show_total_travel_time_column_pref"
            android:title="Total Travel Time"
            android:summary="Show Total Travel Time column" />
    <CheckBoxPreference
            android:key="show_price_column_pref"
            android:title="Price"
            android:summary="Show Price column" />
    <PreferenceScreen
    android:title="Subscreen"
    android:summary="Subscreen">
         <EditTextPreference
              android:key="package_name_preference"
              android:title="Set Package Name"
              android:summary="Set the package name for
generated code"
              android:dialogTitle="Package Name" />
         <RingtonePreference
              android:key="ring_tone_pref"
              android:title="Set Ringtone Preference"
              android:showSilent="true"
              android:ringtoneType="ringtone"
              android:summary="Set Ringtone" />
    </PreferenceScreen>
</PreferenceScreen>
```

Khi thực thi chương trình ta sẽ nhận được kết quả như sau:



CheckBoxPreference hiển thị CheckBox cho phép người dùng check chọn.

EditTextPreference sử dụng để hiển thị một hộp thoại cho phép người dùng nhập vào một đoạn văn bản. Giá trị của đoạn văn bản trên sẽ được lưu xuống file xml.

RingtonePreference sử dụng để hiển thị một hộp lựa chọn nhạc chuông. Đường dẫn chỉ đến file nhạc chuông sẽ được lưu trữ xuống file xml.

Ngoài ra ta có thể sử dụng thẻ **PreferenceScreen** để phân mục con cho trang "Tùy chỉnh" kể trên. Giá trị lưu xuống file xml như sau:

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <boolean name="show_departure_column_pref" value="false" />
    <boolean name="show_total_travel_time_column_pref"
    value="false" />
    <string
    name="ring_tone_pref">content://settings/system/ringtone</s
    tring>
    <string name="package_name_preference">huye</string>
    <string name="selected_flight_sort_option">1</string>
    <boolean name="show_airline_column_pref" value="false" />
    <boolean name="show_arrival_column_pref" value="false" />
    <boolean name="show_price_column_pref" value="false" />
</map>
```

3 Lưu trữ cơ sở dữ liệu với SQLite

Khi lập trình trên di động hay các thiết bị có dung lượng bộ nhớ hạn chế, người ta thường dùng SQLite. SQLite là một hệ quản trị cơ sở dữ liệu nhúng được hiện thực từ chuẩn SQL-92. Giống với cái tên của nó, SQLite chiếm dung lượng nhỏ (khoảng 275KB) nên việc truy xuất dữ liệu được nhanh chóng, không chiếm dụng quá nhiều tài nguyên hệ thống. Do SQLIte là phần mềm mã nguồn mở nên nó không bị giới hạn tác quyền. Vì lý do đó mà SQLite được nhiều hãng sử dụng (Adobe, Apple, Google, Sun, Symbian) và các dự án mã nguồn mở (Mozila, PHP, Python). Đặc biệt, đối với Android, SQLite rất thích hợp để tạo cơ sở dữ liệu cho các ứng dụng.

SQLite tuy nhẹ hơn so với các hệ cơ sở dữ liệu khác nhưng cũng không khác biệt nhiều. SQLite cũng sử dụng ngôn ngữ truy vấn SQL (SELECT, INSERT, DELETE…). SQLite có một hệ thống câu lệnh SQL đầy đủ với các triggers, transactions. Các câu truy vấn cũng như các hệ cơ sở dữ liệu khác. SQLite như một bản thu nhỏ của so với các hệ CSDL khác, vì vậy nó không thể có đầy đủ các chứ năng trên chiếc điện thoại di động của bạn.

SQLite là một lựa chọn thích hợp dành cho ứng dụng trên hệ điều hành Android. Ngoài dung lượng lưu trữ nhỏ gọn, SQLite còn cho phép sử dụng Unicode, kiểu dữ liệu không được cài đặt trong một số phiên bản Android.

3.1 Đặc trưng của SQLite

- SQLite được hiện thực từ tiêu chuẩn SQL-92 của một ngôn ngữ SQL nhưng vẫn còn chứa một số khiếm khuyết.
- Tuy SQLite hỗ trợ triggers nhưng bạn không thể viết trigger cho view. Hoặc SQLite không hỗ trợ lệnh ALTER TABLE, do đó, bạn không thể thực hiện chỉnh sửa hoặc xóa cột trong bảng.
- SQLite không hỗ trợ ràng buộc khóa ngoại , các transactions lồng nhau, phép kết RIGHT OUTER JOINT, FULL OUTER JOINT.
- SQLite sử dụng kiểu dữ liệu khác biệt so với hệ quản trị cơ sở dữ liệu tương ứng. Bạn có thể insert dữ liệu kiểu string vào cột kiểu integer mà không gặp phải bất kỳ lỗi nào.
- Vài tiến trình hoặc luồng có thể truy cập tới cùng một cơ sở dữ liệu. Việc đọc dữ liệu có thể chạy song song, còn việc ghi dữ liệu thì không được phép chạy đồng thời.
- Ngoài các khiếm khuyết trên thì Sqlite cung cấp cho người dùng gần như đầy đủ các chưng năng mà một hệ cơ sở dữ liệu cần có như tạo database; tạo bảng; thêm, xóa, sửa dữ liệu.

3.2 Thao tác với SQLite:

3.2.1 Khái niệm cơ bản

Để thao tác với cơ sở dữ liệu lưu trữ trên SQLite trên Android ta sẽ làm việc với hai loại đối tượng:

SQLiteOpenHelper: Đối tượng dùng để tạo, nâng cấp, đóng mở kết nối trên một cơ sở dữ liệu.

SQLiteDatabase: Đối tượng dùng để thực thi các câu lệnh SQL trên một cơ sở dữ liệu.

Bằng cách trên mỗi cơ sở dữ liệu trên một ứng dụng sẽ có một đối tượng SQLiteOpenHelper dùng để dùng để tạo mới, nâng cấp cũng như đóng mở cơ sở dữ liệu. Thông qua việc sử dụng đối tượng SQLiteOpenHelper ta sẽ lấy về một đối tượng kiểu SQLiteDatabase, đây chính là thể hiện của cơ sở dữ liệu ta cần thao tác. SQLiteOpenHelper hổ trợ hai phương thức để lấy về một đối tượng SQLiteDatabase là:

- getReadableDatabase(): Lấy về một đối tượng SQLiteDatabase ở dạng "chỉ đọc".
- getWriteableDatabase(): Lấy về một đối tượng SQLiteDatabase ở dạng "đọc và ghi".

Để tạo mới một cơ sở dữ liệu ta sẽ kế thừa lớp SQLiteOpenHelper. SQLiteOpenHelper hỗ trợ cho chúng ta 3 phương thức chính:

- Phương thức khởi tạo (constructor): Phương thức này cung cấp các tham số cần thiết để SQLiteOpenHelper có thể làm việc với cơ sở dữ liệu trên Android.
- Phương thức onCreate() được dùng để tạo file lưu trữ cơ sở dữ liệu, tạo các bảng có trong cơ sở dữ liệu cũng như nhập các dữ liệu ban đầu.
- Phương thức onUpgrade() được dùng để giúp bạn cập nhật lại các bảng (table) trong trong cơ sở dữ liệu.

Mỗi cơ sở dữ liệu trên một ứng dụng Android có nhiều phiên bản. Các phiên bản này có thể tương ứng với các phiên bản của ứng dụng.

VD: Ứng dụng quản lý thời khóa biểu sinh viên:

- Phiên bản 01: Chỉ có chức năng quản lý thời khóa biểu nên có các bảng:
  - Học kì.
  - Môn học.
  - Lớp môn học.
  - Bài tập.
- Phiên bản 02: Cập nhập thêm chức năng quản lý contact sinh viên, giảng viên nên có thêm bảng sau:
  - Contact

Tương ứng với mỗi phiên bản ứng dụng kể trên sẽ có hai phiên bản cơ sở dữ liệu khác nhau. Mỗi khi nâng cấp phiên bản ứng dụng ta cũng phải tiến hành nâng cấp phiên bản cơ sở dữ liệu. Điều này sẽ thực hiện ở phương thức onUpgrade() kể trên.

3.2.2 Xây dựng DroidReaderDBOpenHelper

Dưới đây minh họa việc xây dựng cơ sở dữ liệu cho ứng dụng DroidReader. Đây là một ứng dụng sử dụng để đọc tin rss. Ứng dụng có một cơ sở dữ liệu dùng để lưu lại cách kênh rs. Để có được một cơ sở như vậy ta tiến hành tạo một class DroidReaderDBOpenHelper như sau:

```
public class DroidReaderDBOpenHelper extends SQLiteOpenHelper{

     private static final String DATABASE_NAME = "droid_reader";

     private static final String DATABASE_CREATE_STATEMENTS = ""
+
     "CREATE TABLE \"channel\" (" +
     "\"c_id\" INTEGER PRIMARY KEY  AUTOINCREMENT  NOT NULL ," +
     "\"c_title\" VARCHAR," +
     "\"c_url\" VARCHAR," +
     "\"c_description\" VARCHAR," +
     "\"c_pub_date\" DATETIME DEFAULT CURRENT_DATE," +
     "\"c_last_build_date\" DATETIME DEFAULT CURRENT_DATE," +
     "\"c_copyright\" VARCHAR," +
     "\"c_generator\" VARCHAR" +
     ")";

     public DroidReaderDBOpenHelper(Context context,int
DATABASE_VERSION) {
          super(context, DATABASE_NAME, null, DATABASE_VERSION);
     }

     @Override
     public void onCreate(SQLiteDatabase db) {
          db.execSQL(DATABASE_CREATE_STATEMENTS);
     }

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

     }

     public    SQLiteDatabase  getDatabase(){
          return    this.getWritableDatabase();
     }
```

```
    public boolean closeAllDatabase(){
        try{
            this.close();
        }
        catch (Exception e) {
            return false;
        }
        return true;
    }

    public Cursor query(String table, String[] columns, String
selection, String[] selectionArgs, String groupBy, String
having, String orderBy){

        SQLiteDatabase database = this.getWritableDatabase();

        Cursor result = null;

        try {
            result = database.query(table, columns,
selection, selectionArgs, groupBy, having, orderBy);
        } catch (Exception e) {
            e.printStackTrace();
        }
        finally{
            database.close();
        }

        return result;
    }

    public long update(String tableName,ContentValues
values,String whereClause, String[] params){

        int count  = 0;

        SQLiteDatabase database = this.getWritableDatabase();

        try{
            count = database.update( tableName, values,
whereClause, params);
        } catch(Exception e){
            e.printStackTrace();
        } finally {
            database.close();
        }
```

```
            return count;
      }

      public long insert(String tableName,ContentValues values){

            SQLiteDatabase database = this.getWritableDatabase();

            long count = database.insert(tableName, null, values);

            return count;
     }

      public int deleteRow(String table,String clause,String[]
params){

            SQLiteDatabase database = getDatabase();

            int count = 0;

            try{
                  count = database.delete(table, clause, params);
            }
            catch(Exception e){
                  e.printStackTrace();
            }
            finally{
                  database.close();
            }

            return    count;

      }
}
```

Ta có thể thấy được ở class DroidReaderDBOpenHelper ở trên ta định nghĩa một phương thức khởi tạo với nội dung:

```
public DroidReaderDBOpenHelper(Context context,int
DATABASE_VERSION) {

            super(context, DATABASE_NAME, null, DATABASE_VERSION);

}
```

Phương thức khởi tạo trên đơn giản sẽ gọi phương thức khởi tạo của class SQLiteOpenHelper. Phương thức khởi tạo này sẽ tiến hành kiểm tra xem cơ sở dữ liệu trên đã tồn tại hay chưa. Nếu cơ sở dữ liệu chưa tồn tại thì nó sẽ gọi phương thức **onCreate()** để tạo mới cơ sở dữ liệu. Nếu cơ sở dữ liệu đã tồn tại thì nó sẽ tiến hành kiểm tra phiên bản cơ sở dữ liệu hiện tại, nếu phiên bản hiện tại cũ hơn

phiên bản khi khởi tạo đối tượng DroidReaderDBOpenHelper thì sẽ tiến hành gọi phương thức **onUpgrade()** để nâng cấp phiên bản cơ sở dữ liệu.

Trong DroidReaderDBOpenHelper ở trên ta thấy có phương thức sau:

- getDatabase():
  - Mô tả:
    - Dùng để lấy về một đối tượng kiểu SQLiteDatabase, đây chính là đối tượng cơ sở dữ liệu của ứng dụng.
- closeAllDatabase():
  - Mô tả:
    - Dùng để đóng toàn bộ các cơ sở dữ liệu đã được mở bởi DroidReaderDBOpenHelper. Phương thức này đơn giản sẽ gọi phương thức close() của class SQLiteOpenHelper.

Ngoài hai phương thức chính trên ta còn có những phương thức sau:

- query(): Dùng để truy vấn dữ liệu.
- insert(): Dùng để thêm dòng dữ liệu.
- update(): Dùng cập nhập dòng dữ liệu.
- delete(): Dùng xóa dòng dữ liệu.

Các phương thức này sẽ được trình bày rỏ hơn ở phần sau:

3.2.3 Phương thức query()

- Mô tả:
  - Dùng để thực thi một câu lệnh truy vấn.
  - Phương thức này sẽ tạo một đối tượng database kiểu SQLiteDatabase. Đây là đối tượng được trả về bởi phương thức getReadableDatabase() được định nghĩa trong class SQLiteOpenHelper. Phương thức này cho phép lấy về đối tượng SQLiteDatabase ở dạng "chỉ đọc". Ta sẽ sử dụng đối tượng database này để thực thi câu truy vấn.
- Input:
  - String table: Tên bảng cần truy vấn dữ liệu.
  - String[] colums: Mảng chứa tên các cột cần lấy dữ liệu.

- String selection: Mệnh đề điều kiện của câu truy vấn (Mệnh đề where). Mệnh đề này có dạng: "id=? AND name=?". Trong đó giá trị của id và name sẽ được chỉ định ở tham số String[] selectionArgs.
- String[] selectionArgs: Tham số của mệnh đề điều kiện ở trên.
- String groupBy: Điều kiện group by.
- String having: Điều kiện having.
- String orderBy: Điều kiện order by.

- Output:
  - Con trỏ Cursor chỉ đến dòng dữ liệu đầu tiên. Ta sẽ sử dụng con trỏ này để đọc các dòng dữ liệu trên tập dữ liệu được trả về bởi câu lệnh query.

Nội dung cài đặt của phương thức như sau:

```
public Cursor query(String table, String[] columns, String
selection, String[] selectionArgs, String groupBy, String
having, String orderBy){
     SQLiteDatabase database = this.getWritableDatabase();
     Cursor result = null;
     try {
          result = database.query(table, columns, selection,
selectionArgs, groupBy, having, orderBy);
     } catch (Exception e) {
          e.printStackTrace();
     }
     finally{
          database.close();
     }
     return result;
}
```

Phương thức query ở trên sẽ tạo một đối tượng database được trả về từ phương thức getWriteableDatabase() . Phương thức này cho phép lấy về đối tượng SQLiteDatabase ở dạng "đọc và ghi". Khi thực thi, phương thức database.query(…) sẽ trả về đối tượng Cursor. Đây là đối tượng con trỏ trỏ đến tập record được trả về bởi câu truy vấn. Ta sẽ sử dụng con trỏ này để đọc dữ liệu theo cách sau:

```
     DroidReaderDBOpenHelper myOpenHelper = new
DroidReaderDBOpenHelper(context);
     String[] colums = {"c_id","c_title","c_url","c_description"};

     Cursor result = myOpenHelper.query("channel", colums, null,
null, null, null, "c_title");

     ArrayList<Channel> channels = new ArrayList<Channel>();
```

```
// Di chuyển đến dòng đầu tiên
result.moveToFirst();
while (!result.isAfterLast()) {

      String c_id = result.getString(0);
      String c_title = result.getString(1);
      String c_url = result.getString(2);
      String c_description = result.getString(3);
      // Di chuyển đến dòng tiếp theo.
      result.moveToNext();
}
// Đóng con trỏ.
result.close();
// Đóng kết nối tất cả database đang mở.
myOpenHelper.close();
```

Với cách cài đặt phương thức DroidReaderDBOpenHelper.query() ở trên ta chỉ có thể truy vấn dữ liệu trên một bảng. Để truy vấn dữ liệu trên nhiều hơn một bảng ta có thể cài đặt lại phương thức này với một chút thay đổi. Thay vì gọi lệnh database.query() thì ta đổi thành database.rawQuery(String sql, String[] selectionArgs). Phương thức này nhận vào hai tham số. Trong đó tham số String sql là câu lệnh SQL dùng truy vấn dữ liệu. Câu lệnh này có dạng: "SELECT url, description FROM channel WHERE id=?". Trong đó giá trị của id sẽ được cung cấp trong mảng String[] selectionArgs. Bằng cách này ta có thể truy vấn dữ liệu từ nhiều hơn một bảng.

3.2.4 Phương thức insert():

- Mô tả:
  - Đây là phương thức dùng để thêm một dòng vào cơ sở dữ liệu.
  - Phương thức này sẽ tạo một đối tượng database kiểu SQLiteDatabase. Đây là đối tượng được trả về bởi phương thức getWriteableDatabase() được định nghĩa trong class SQLiteOpenHelper. Phương thức này cho phép lấy về đối tượng SQLiteDatabase ở dạng "đọc và ghi". Ta sẽ sử dụng đối tượng database này để thực thi câu truy vấn.
- Input:
  - String tableName: Tên bảng cần thêm dữ liệu.
  - ContentValues values: Đối tượng chứa dữ liệu của các cột cần thêm vào. Đối tượng này gần giống với kiểu dữ liệu từ điển. Trong đó key sẽ là tên cột cần thêm dữ liệu và value là dữ liệu cần thêm.
- Output:
  - Số dòng dữ liệu thêm thành công.

Nội dung cài đặt của phương thức như sau:

```
public long insert(String tableName,ContentValues values){
      SQLiteDatabase database = this.getWritableDatabase();
      long count = database.insert(tableName, null, values);
      return count;
}
```

Phương thức trên cũng tương tự như phương thức query ở trên. Tuy nhiên, ở đây ta sẽ gọi phương thức database.insert(…) để cập nhập dữ liệu với các tham số đã đề cập ở trên.

3.2.5 Phương thức update():

- Mô tả:
    - Đây là phương thức dùng để cập nhập một dòng trong cơ sở dữ liệu.
    - Phương thức này sẽ tạo một đối tượng database kiểu SQLiteDatabase. Đây là đối tượng được trả về bởi phương thức getWriteableDatabase() được định nghĩa trong class SQLiteOpenHelper. Phương thức này cho phép lấy về đối tượng SQLiteDatabase ở dạng "đọc và ghi". Ta sẽ sử dụng đối tượng database này để thực thi câu truy vấn.
- Input:
    - String tableName: Tên bảng cần cập nhật.
    - String whereClause: Mệnh đề điều kiện của dòng cần cập nhập dữ liệu (Mệnh đề where). Mệnh đề này có dạng: "id=? AND name=?". Trong đó giá trị của id và name sẽ được chỉ định ở tham số String[] parms.
    - String[] parms: Tham số của mệnh đề điều kiện ở trên.
    - ContentValues values: Đối tượng chứa dữ liệu của các cột cần cập nhập. Đối tượng này gần giống với kiểu dữ liệu từ điển. Trong đó key sẽ là tên cột và value là dữ liệu cần cập nhập.
- Output:
    - Số dòng cập nhập thành công.

Nội dung cài đặt của phương thức như sau:

```
public long update(String tableName,ContentValues values,String
whereClause, String[] params){
      int count  = 0;
      SQLiteDatabase database = this.getWritableDatabase();

      try{
```

```
            count = database.update( tableName, values,
whereClause, params);
      } catch(Exception e){
            e.printStackTrace();
      } finally {
            database.close();
      }
      return count;
}
```

3.2.6 Phương thức delete():

- Mô tả:
    - Đây là phương thức dùng xóa một dòng trong cơ sở dữ liệu.
    - Phương thức này sẽ tạo một đối tượng database kiểu SQLiteDatabase. Đây là đối tượng được trả về bởi phương thức getWriteableDatabase() được định nghĩa trong class SQLiteOpenHelper. Phương thức này cho phép lấy về đối tượng SQLiteDatabase ở dạng "đọc và ghi". Ta sẽ sử dụng đối tượng database này để thực thi câu truy vấn.
- Input:
    - String tableName: Tên bảng cần xóa dữ liệu.
    - String whereClause: Mệnh đề điều kiện của dòng cần xóa (Mệnh đề where). Mệnh đề này có dạng: "id=? AND name=?". Trong đó giá trị của id và name sẽ được chỉ định ở tham số String[] parms.
    - String[] parms: Tham số của mệnh đề điều kiện ở trên.
- Output:
    - Số dòng xóa thành công.

Nội dung cài đặt của phương thức như sau:

```
public int delete(String tableName,String whereClause,String[]
params){
      SQLiteDatabase database = getDatabase();
      int count = 0;
      try{
            count = database.delete(tableName, whereClause,
params);
      }
      catch(Exception e){
            e.printStackTrace();
      }
```

```
        finally{
            database.close();
        }
        return    count;
}
```