***Creating and Saving Preferences***

***Tạo và lưu các sở thích (Preferences).***

To create or modify a Shared Preference, call getSharedPreferences on the application Context, pass- ing in the name of the Shared Preferences to change. Shared Preferences are shared across an applica- tion’s components but aren’t available to other applications.

To modify a Shared Preference, use the SharedPreferences.Editor class. Get the Editor object by calling edit on the SharedPreferences object you want to change. To save edits, call commit on the Editor, as shown in the code snippet below.

Để tạo hoặc thay đổi một Shared Preference, gọi getSharedPreferences trên ứng dụng ngữ cảnh, chuyển sang tên của Shared Preference để thay đổi. Shared Preference được chia sẻ qua các thành phần của một ứng dụng nhưng không thể dùng được cho các ứng dụng khác.

Để thay đổi một Shared Preference, sử dụng lớp SharedPreferences.Editor. Lấy đối tượng Editor bằng cách gọi edit trên đối tượng SharedPreferences của đối tượng muốn thay đổi. Để lưu các edit, gọi commit trên Editor, giống như mã nguồn bên dưới.

public static final String MYPREFS = “mySharedPreferences”;

protected void savePreferences(){

// Create or retrieve the shared preference object.

// Tạo hoặc nhận đối tượng shared preference

int mode = Activity.MODE\_PRIVATE;

SharedPreferences mySharedPreferences = getSharedPreferences(MYPREFS, mode);

// Retrieve an editor to modify the shared preferences.

// Nhận một editor để thay đổi shared preferences

SharedPreferences.Editor editor = mySharedPreferences.edit();

// Store new primitive types in the shared preferences object.

// Lưu các kiểu nguyên thủy trong đối tượng shared preferences.

editor.putBoolean(“isTrue”, true);

editor.putFloat(“lastFloat”, 1f); editor.putInt(“wholeNumber”, 2); editor.putLong(“aNumber”, 3l); editor.putString(“textEntryValue”, “Not Empty”);

// Commit the changes.

// Đưa ra các thay đổi.

editor.commit();

}

***Retrieving Shared Preferences***

***Lấy Shared Preferences***

Accessing saved Shared Preferences is also done with the getSharedPreferences method. Pass in the name of the Shared Preference you want to access, and use the type-safe get<type> methods to extract saved values.

Each getter takes a key and a default value (used when no value is available for that key), as shown in the skeleton code below:

Truy cập Shared Preferences đã lưu cũng có thể được làm với phương thức getSharedPreferences. Qua tên của Shared Preference ta muốn truy cập, và sử dụng các loại phương pháp an toàn get <type> để trích xuất các giá trị lưu.

Mỗi getter lấy một khóa và một giá trị mặc định (dùng khi không có giá trị nào có thể dùng được cho khóa đó), như trong mã nguồn dưới đây:

public void loadPreferences(){

// Get the stored preferences

// Lấy các sở thích đã được lưu trữ.

int mode = Activity.MODE\_PRIVATE;

SharedPreferences mySharedPreferences = getSharedPreferences(MYPREFS, mode);

// Retrieve the saved values.

// Lấy các giá trị đã lưu

boolean isTrue = mySharedPreferences.getBoolean(“isTrue”, false); float lastFloat = mySharedPreferences.getFloat(“lastFloat”, 0f); int wholeNumber = mySharedPreferences.getInt(“wholeNumber”, 1); long aNumber = mySharedPreferences.getLong(“aNumber”, 0);

String stringPreference;

stringPreference = mySharedPreferences.getString(“textEntryValue”, “”);

}

***Saving the Activity State***

***Lưu trạng thái hoạt động***

If you want to save Activity information that doesn’t need to be shared with other components (e.g., class instance variables), you can call Activity.getPreferences() without specifying a preferences name. Access to the Shared Preferences map returned is restricted to the calling Activity; each Activity supports a single unnamed SharedPreferences object.

The following skeleton code shows how to use the Activity’s private Shared Preferences:

Nếu chúng ta muốn lưu thông tin hoạt động mà không cần phải chia sẻ với các thành phần khác (ví dụ: ví dụ lớp các biến), ta có thể gọi Activity.getPreferences() mà không cần đặc tả tên một preferences. Truy cập tới bản đồ Shared Preferences trả về bị hạn chế việc gọi hoạt động; mỗi hoạt động hỗ trợ một đối tượng SharedPreferences không tên riêng lẻ.

Mã nguồn dưới đây thể hiện làm cách nào để sử dụng Shared Preferences riêng của hoạt động.

protected void saveActivityPreferences(){

// Create or retrieve the activity preferences object.

// Tạo hay nhận đối tượng hoạt động preferences

SharedPreferences activityPreferences = getPreferences(Activity.MODE\_PRIVATE);

// Retrieve an editor to modify the shared preferences.

// Nhận một editor để thay đổi shared preferences

SharedPreferences.Editor editor = activityPreferences.edit();

// Retrieve the View

// Nhận View

TextView myTextView = (TextView)findViewById(R.id.myTextView);

// Store new primitive types in the shared preferences object.

// Lưu trữ các loại sơ khai trong đối tượng shared preferences

editor.putString(“currentTextValue”,

myTextView.getText().toString());

// Commit changes.

// Đưa ra các thay đổi.

editor.commit();

}

Saving and Restoring Instance State

Lưu và trả lại trường hợp trạng thái

To save Activity instance variables, Android offers a specialized alternative to Shared Preferences.

By overriding an Activity’s onSaveInstanceState event handler, you can use its Bundle parameter to save instance values. Store values using the same get and put methods as shown for Shared Pref- erences, before passing the modified Bundle into the superclass’s handler, as shown in the following code snippet:

Để lưu các biến trường hợp hoạt động, Android đưa ra một thay thế chuyên dụng cho Shared Preferences.

Bằng cách ghi đè xử lý sự kiện onSaveInstanceState của một hoạt động, ta có thể sử dụng tham số Bundle của nó để lưu các giá trị ví dụ. Lưu trữ các giá trị sử dụng cùng các phương thức get và put như đã trình bày cho Shared Preferences, trước khi đi qua các Bundle được chỉnh sửa vào xử lý của lớp cha, như thể hiện trong đoạn mã sau

private static final String TEXTVIEW\_STATE\_KEY = “TEXTVIEW\_STATE\_KEY”;

[@Override](mailto:@Override)

public void onSaveInstanceState(Bundle outState) {

// Retrieve the View

// Nhận View

TextView myTextView = (TextView)findViewById(R.id.myTextView);

// Save its state

// Lưu trữ trạng thái của nó

outState.putString(TEXTVIEW\_STATE\_KEY, myTextView.getText().toString());

super.onSaveInstanceState(outState);

}

This handler will be triggered whenever an Activity completes its Active life cycle, but only when it’s not being explicitly finished. As a result, it’s used to ensure a consistent Activity state between active life cycles of a single user session.

The saved Bundle is passed in to the onRestoreInstanceState and onCreate methods if the applica- tion is forced to restart during a session. The following snippet shows how to extract values from the Bundle and use them to update the Activity instance state:

Xử lý này sẽ được kích hoạt bất cứ khi nào một hoạt động hoàn thành vòng đời của hoạt động của nó, nhưng chỉ khi nó không được hoàn thành một cách rõ ràng. Kết quả là, nó được sử dụng để đảm bảo trạng thái Hoạt động nhất quán giữa các vòng đời hoạt động của một phiên làm việc duy nhất.

Bundle lưu được thông qua trong các phương pháp onRestoreInstanceState và onCreate nếu ứng dụng bắt buộc phải khởi động lại trong một phiên làm việc. Đoạn mã sau đây cho thấy làm thế nào để trích xuất các giá trị từ các Bundle và sử dụng chúng để cập nhật trạng thái Hoạt động:

[@Override](mailto:@Override)

public void onCreate(Bundle icicle) { super.onCreate(icicle); setContentView(R.layout.main);

TextView myTextView = (TextView)findViewById(R.id.myTextView); String text = “”;

if (icicle != null && icicle.containsKey(TEXTVIEW\_STATE\_KEY))

text = icicle.getString(TEXTVIEW\_STATE\_KEY);

myTextView.setText(text);

}

It’s important to remember that onSaveInstanceState is called only when an Activity becomes inactive, but not when it is being closed by a call to finish or by the user pressing the Back button.

Điều quan trọng là phải nhớ rằng onSaveInstanceState chỉ được gọi khi một hoạt động trở thành không hoạt động, nhưng không phải khi nó đang được đóng lại bằng việc gọi kết thúc hoặc do người sử dụng nhấn vào nút Back.

***Saving the To-Do List Activity State***

***Lưu trạng thái hoạt động To-Do List***

Currently, each time the To-Do List example application is restarted, all the to-do items are lost and any text entered into the text entry box is cleared. In this example, you’ll start to save the application state of the To-Do list application across sessions.

Hiện nay, mỗi khi ứng dụng mẫu To-Do List được khởi động lại, tất cả các mục phải làm bị mất và bất kỳ văn bản nhập vào ô nhập văn bản sẽ bị xóa. Trong ví dụ này, ta sẽ bắt đầu để lưu các trạng thái ứng dụng của ứng dụng To-Do list qua các phiên làm việc.

The instance state in the ToDoList Activity consists of three variables:

❑ Is a new item being added?

❑ What text exists in the new item entry textbox?

❑ What is the currently selected item?

Using the Activity’s default Shared Preference, you can store each of these values and update the UI when the Activity is restarted.

Trạng thái trong Hoạt động ToDoList bao gồm ba biến:

- Có phải một mục mới được thêm vào?

- Văn bản gì tồn tại trong textbox mới nhập hàng?

- Mục đang được chọn là gì?

Sử dụng Shared Preference mặc định của Hoạt động, ta có thể lưu trữ mỗi giá trị và cập nhật giao diện người dùng khi Hoạt động được khởi động lại.

Later in this chapter, you’ll learn how to use the SQLite database to persist the to-do items as well. This example is a first step that shows how to ensure a seamless experience by saving Activity instance details.

Sau đó trong chương này, ta cũng sẽ tìm hiểu làm thế nào để sử dụng cơ sở dữ liệu SQLite để tiếp tục các mục To-Do. Ví dụ này là bước đầu cho thấy làm thế nào để đảm bảo một trải nghiệm liền mạch bằng cách lưu những chi tiết mẫu của hoạt động.

1. Start by adding static String variables to use as preference keys.

Bắt đầu bằng việc thêm các biến tĩnh String để sử dụng như là các khóa preference.

private static final String TEXT\_ENTRY\_KEY = “TEXT\_ENTRY\_KEY”;

private static final String ADDING\_ITEM\_KEY = “ADDING\_ITEM\_KEY”;

private static final String SELECTED\_INDEX\_KEY = “SELECTED\_INDEX\_KEY”;

2. Next, override the onPause method. Get the Activity’s private Shared Preference object, and get its Editor object.

Tiếp theo, chép đè phương thức onPause. Nhận tin đối tượng Sở thích chung của hoạt động, và có được đối tượng biên tập viên của nó.

Using the keys you created in Step 1, store the instance values based on whether a new item is being added and any text in the “new item” Edit Box.

Sử dụng các phím mà bạn đã tạo ở bước một, lưu trữ các giá trị mẫu dựa trên việc một mục mới đang được bổ sung và bất kỳ văn bản trong "mục mới" Edit Box được hay không.

[@Override](mailto:@Override)

protected void onPause(){

super.onPause();

// Get the activity preferences object.

// Lấy đối tượng hoạt động preferences

SharedPreferences uiState = getPreferences(0);

// Get the preferences editor.

// Lấy editor preferences

SharedPreferences.Editor editor = uiState.edit();

// Add the UI state preference values.

// Thêm giá trị trạng thái preference giao diện người dùng

editor.putString(TEXT\_ENTRY\_KEY, myEditText.getText().toString());

editor.putBoolean(ADDING\_ITEM\_KEY, addingNew);

// Commit the preferences

// Đưa ra các preferences.

editor.commit();

}

3. Write a restoreUIState method that applies the instance values you recorded in Step 2 when the application restarts.

Modify the onCreate method to add a call to the restoreUIState method at the very end.

3. Viết một phương thức restoreUIState áp dụng các ví dụ giá trị mà bạn ghi lại ở bước 2 khi khởi động lại ứng dụng.

Sửa đổi các phương thức onCreate để thêm một cuộc gọi đến các phương pháp restoreUIState ở cuối cùng.

[@Override](mailto:@Override)

public void onCreate(Bundle icicle) {

[ ... existing onCreate logic ... ]

restoreUIState();

}

private void restoreUIState() {

// Get the activity preferences object.

// Lấy đối tượng hoạt động preferences

SharedPreferences settings = getPreferences(Activity.MODE\_PRIVATE);

// Read the UI state values, specifying default values.

// Đọc giá trị trạng thái giao diện người dùng, đặc tả các giá trị mặc định

String text = settings.getString(TEXT\_ENTRY\_KEY, “”);

Boolean adding = settings.getBoolean(ADDING\_ITEM\_KEY, false);

// Restore the UI to the previous state.

// Khôi phục giao diện người dùng về trạng thái trước đó

if (adding) {

addNewItem();

myEditText.setText(text);

}

}

4. Record the index of the selected item using the onSaveInstanceState / onRestore InstanceState mechanism. It’s then only saved and applied if the application is killed without the user’s explicit instruction.

4. Ghi chỉ số của các mục đã chọn sử dụng kỹ thuật onSaveInstanceState / onRestore InstanceState. Sau đó chỉ lưu và áp dụng nếu như ứng dụng bị hủy mà không có sự cho phép của người sử dụng.

[@Override](mailto:@Override)

public void onSaveInstanceState(Bundle outState) {

outState.putInt(SELECTED\_INDEX\_KEY, myListView.getSelectedItemPosition());

super.onSaveInstanceState(outState);

}

[@Override](mailto:@Override)

public void onRestoreInstanceState(Bundle savedInstanceState) {

int pos = -1;

if (savedInstanceState != null)

if (savedInstanceState.containsKey(SELECTED\_INDEX\_KEY))

pos = savedInstanceState.getInt(SELECTED\_INDEX\_KEY, -1);

myListView.setSelection(pos);

}

When you run the To-Do List application, you should now see the UI state persisted across sessions.

That said, it still won’t persist the to-do list items — you’ll add this essential piece of functionality later in the chapter.

Khi ta chạy các ứng dụng To-Do List, chúng ta sẽ thấy tình trạng giao diện người dùng liên tục theo phiên làm việc. Điều đó nói rằng, nó vẫn sẽ không tồn tại các mục trong danh sách công việc phải làm - ta sẽ thêm phần cần thiết của chức năng này ở phần sau của chương.

***Creating a Preferences Page for the Earthquake Viewer***

***Tạo một trang Preferences cho người quan sát động đất***

In Chapter 5, you created an earthquake monitor that showed a list of recent earthquakes based on an Internet feed.

Trong chương 5, ta tạo ra một màn hình động đất cho thấy một danh sách các trận động đất gần đây dựa trên nguồn dữ liệu từ Internet.

In the following example, you’ll create a Preferences page for this earthquake viewer that lets users configure application settings for a more personalized experience. You’ll provide the option to toggle automatic updates, control the frequency of updates, and filter the minimum earthquake magnitude displayed.

Trong ví dụ sau đây, ta sẽ tạo ra một trang Preferences cho người quan sát động đất, cho phép người dùng cấu hình cài đặt ứng dụng cho nhiều kinh nghiệm cá nhân hơn. Ta sẽ cung cấp tùy chọn để chuyển đổi tự động cập nhật, kiểm soát tần số cập nhật, và lọc các trận động đất cường độ tối thiểu hiển thị.

Later in this chapter, you’ll extend this example further by creating a Content Provider to save and share earthquake data with other applications.

Sau đó trong chương này, ta sẽ mở rộng ví dụ này hơn nữa bằng cách tạo ra một nhà cung cấp nội dung để lưu và chia sẻ dữ liệu động đất với các ứng dụng khác.

1. Open the Earthquake project you created in Chapter 5.

Add new String resources for the labels displayed in the “Preferences” screen. Also, add a String for the new Menu Item that will let users access the Preferences screen.

1. Mở dự án Earthquake đã tạo ra trong Chương 5.

Thêm nguồn String mới cho các nhãn hiển thị trong màn hình "Preferences". Ngoài ra, thêm một chuỗi cho các Menu Item mới sẽ cho phép người dùng truy cập vào màn hình Preferences.

<?xml version=”1.0” encoding=”utf-8”?>

<resources>

<string name=”app\_name”>Earthquake</string>

<string name=”quake\_feed”>

<http://earthquake.usgs.gov/eqcenter/catalogs/1day-M2.5.xml>

</string>

<string name=”menu\_update”>Refresh Earthquakes</string>

<string name=”auto\_update\_prompt”>Auto Update?</string>

<string name=”update\_freq\_prompt”>Update Frequency</string>

<string name=”min\_quake\_mag\_prompt”>Minimum Quake Magnitude</string>

<string name=”menu\_preferences”>Preferences</string>

</resources>

2. Create a new preferences.xml layout resource that lays out the UI for the Preferences Activ- ity. Include a checkbox for indicating the “automatic update” toggle, and spinners to select the update rate and magnitude filter.

2. Tạo một tài nguyên layout preferences.xml mới, đưa ra các giao diện người dùng cho các Hoạt động Preferences. Bao gồm một hộp kiểm cho thấy sự "cập nhật tự động" chuyển đổi, và quay để chọn tốc độ cập nhật và bộ lọc độ lớn.

<LinearLayout [xmlns:android=”http://schemas.android.com/apk/res/android”](http://schemas.android.com/apk/res/android) android:orientation=”vertical” android:layout\_width=”fill\_parent” android:layout\_height=”fill\_parent”>

<TextView android:layout\_width=”fill\_parent” android:layout\_height=”wrap\_content” [android:text=”@string/auto\_update\_prompt”](mailto:@string)

/>

<CheckBox [android:id=”@+id/checkbox\_auto\_update”](mailto:@+id) android:layout\_width=”fill\_parent” android:layout\_height=”wrap\_content”

/>

<TextView android:layout\_width=”fill\_parent” android:layout\_height=”wrap\_content”android:text=”@string/update\_freq\_prompt”

/>

<Spinner [android:id=”@+id/spinner\_update\_freq”](mailto:@+id) android:layout\_width=”fill\_parent” android:layout\_height=”wrap\_content” android:drawSelectorOnTop=”true”

/>

<TextView android:layout\_width=”fill\_parent” android:layout\_height=”wrap\_content” [android:text=”@string/min\_quake\_mag\_prompt”](mailto:@string)

/>

<Spinner [android:id=”@+id/spinner\_quake\_mag”](mailto:@+id) android:layout\_width=”fill\_parent” android:layout\_height=”wrap\_content” android:drawSelectorOnTop=”true”

/>

<LinearLayout android:orientation=”horizontal” android:layout\_width=”fill\_parent” android:layout\_height=”wrap\_content”>

<Button

[android:id=”@+id/okButton”](mailto:@+id) android:layout\_width=”wrap\_content” android:layout\_height=”wrap\_content” [android:text=”@android:string/ok”](mailto:@android:string)

/>

<Button [android:id=”@+id/cancelButton”](mailto:@+id) android:layout\_width=”wrap\_content” android:layout\_height=”wrap\_content” [android:text=”@android:string/cancel”](mailto:@android:string)

/>

</LinearLayout>

</LinearLayout>

3. Create four new array resources in a new res/values/arrays.xml file. They will provide the values to use for the update frequency and minimum magnitude filter.

3. Tạo bốn nguồn mảng mới trong tệp res/values/arrays.xml mới. Chúng sẽ cung cấp các giá trị sử dụng cho các bản cập nhật tần suất và cường độ lọc tối thiểu.

<?xml version=”1.0” encoding=”utf-8”?>

<resources>

<string-array name=”update\_freq\_options”>

<item>Every Minute</item>

<item>5 minutes</item>

<item>10 minutes</item>

<item>15 minutes</item>

<item>Every Hour</item>

</string-array>

<array name=”magnitude”>

<item>3</item>

<item>5</item>

<item>6</item>

<item>7</item>

<item>8</item>

</array>

<string-array name=”magnitude\_options”>

<item>3</item>

<item>5</item>

<item>6</item>

<item>7</item>

<item>8</item>

</string-array>

<array name=”update\_freq\_values”>

<item>1</item>

<item>5</item>

<item>10</item>

<item>15</item>

<item>60</item>

</array>

</resources>

4. Create the Preferences Activity. It will be used to display the application preferences.

Override onCreate to inflate the layout you created in Step 2, and get references to the

Checkbox and both Spinner controls. Then make a call to the populateSpinners stub.

4. Tạo các Hoạt động Preferences. Nó sẽ được sử dụng để hiển thị các ứng dụng preferences.

Ghi đè onCreate để đưa bố trí mà bạn đã tạo ở bước 2, và nhận được tài liệu tham khảo để các

Checkbox và cả hai điều khiển Spinner. Sau đó thực hiện gọi đến các populateSpinners còn sơ khai.

package com.paad.earthquake;

import android.app.Activity;

import android.content.SharedPreferences;

import android.content.SharedPreferences.Editor;

import android.os.Bundle;

import android.view.View;

import android.widget.ArrayAdapter;

import android.widget.Button; import android.widget.CheckBox; import android.widget.Spinner;

public class Preferences extends Activity { CheckBox autoUpdate;

Spinner updateFreqSpinner;

Spinner magnitudeSpinner;

[@Override](mailto:@Override)

public void onCreate(Bundle icicle) { super.onCreate(icicle); setContentView(R.layout.preferences);

updateFreqSpinner = (Spinner)findViewById(R.id.spinner\_update\_freq);

magnitudeSpinner = (Spinner)findViewById(R.id.spinner\_quake\_mag);

autoUpdate = (CheckBox)findViewById(R.id.checkbox\_auto\_update);

populateSpinners();

}

private void populateSpinners() {

}

}

5. Fill in the populateSpinners method, using Array Adapters to bind each Spinner to its corre- sponding array.

Điền vào các phương thức populateSpinners, sử dụng adapter mảng để ràng buộc mỗi Spinner đến mảng tương ứng của nó

private void populateSpinners() {

// Populate the update frequency spinner

// Gắn vào bản cập nhật tần số spinner

ArrayAdapter<CharSequence> fAdapter;

fAdapter = ArrayAdapter.createFromResource(this, R.array.update\_freq\_options, android.R.layout.simple\_spinner\_item);

fAdapter.setDropDownViewResource(

android.R.layout.simple\_spinner\_dropdown\_item);

updateFreqSpinner.setAdapter(fAdapter);

// Populate the minimum magnitude spinner

//Gắn vào spinner độ lớn nhỏ nhất

ArrayAdapter<CharSequence> mAdapter;

mAdapter = ArrayAdapter.createFromResource(this, R.array.magnitude\_options, android.R.layout.simple\_spinner\_item);

mAdapter.setDropDownViewResource(

android.R.layout.simple\_spinner\_dropdown\_item);

magnitudeSpinner.setAdapter(mAdapter);

}

6. Add public static String values to use to identify the named Shared Preference you’re going to create, and the keys it will use to store each preference value. Update the onCreate method to retrieve the named preference and call updateUIFromPreferences. The updateUIFromPreferences method uses the get<type> methods on the Shared Preference object to retrieve each preference value and apply it to the current UI.

Thêm giá trị toàn cục chuỗi tĩnh để sử dụng để xác định các Shared Preference ta sắp tạo ra, và các khóa sẽ sử dụng để lưu trữ giá trị của từng preference. Cập nhật các phương thức onCreate để lấy lại preference được đặt tên và gọi updateUIFromPreferences. Phương thức updateUIFromPreferences sử dụng các phương thức get <type> trên đối tượng Shared Preference để lấy giá trị của từng preference và áp dụng nó vào giao diện người dùng hiện tại.

public static final String USER\_PREFERENCE = “USER\_PREFERENCES”;

public static final String PREF\_AUTO\_UPDATE = “PREF\_AUTO\_UPDATE”;

public static final String PREF\_MIN\_MAG = “PREF\_MIN\_MAG”;

public static final String PREF\_UPDATE\_FREQ = “PREF\_UPDATE\_FREQ”; SharedPreferences prefs;

[@Override](mailto:@Override)

public void onCreate(Bundle icicle) { super.onCreate(icicle); setContentView(R.layout.preferences);

updateFreqSpinner = (Spinner)findViewById(R.id.spinner\_update\_freq); magnitudeSpinner = (Spinner)findViewById(R.id.spinner\_quake\_mag); autoUpdate = (CheckBox)findViewById(R.id.checkbox\_auto\_update);

populateSpinners();

prefs = getSharedPreferences(USER\_PREFERENCE, Activity.MODE\_PRIVATE);

updateUIFromPreferences();

}

private void updateUIFromPreferences() {

boolean autoUpChecked = prefs.getBoolean(PREF\_AUTO\_UPDATE, false);

int updateFreqIndex = prefs.getInt(PREF\_UPDATE\_FREQ, 2);

int minMagIndex = prefs.getInt(PREF\_MIN\_MAG, 0);

updateFreqSpinner.setSelection(updateFreqIndex);

magnitudeSpinner.setSelection(minMagIndex);

autoUpdate.setChecked(autoUpChecked);

}

7. Still in the onCreate method, add event handlers for the OK and Cancel buttons. Cancel should close the Activity, while OK should call savePreferences first.

7. Vẫn còn trong phương thức onCreate, thêm xử lý sự kiện cho nút OK và Cancel. Cancel nên đóng Hoạt động, trong khi OK nên gọi savePreferences trước.

[@Override](mailto:@Override)

public void onCreate(Bundle icicle) { super.onCreate(icicle); setContentView(R.layout.preferences);

updateFreqSpinner = (Spinner)findViewById(R.id.spinner\_update\_freq); magnitudeSpinner = (Spinner)findViewById(R.id.spinner\_quake\_mag); autoUpdate = (CheckBox)findViewById(R.id.checkbox\_auto\_update);

populateSpinners();

prefs = getSharedPreferences(USER\_PREFERENCE, Activity.MODE\_PRIVATE);

updateUIFromPreferences();

Button okButton = (Button) findViewById(R.id.okButton);

okButton.setOnClickListener(new View.OnClickListener() {

public void onClick(View view) { savePreferences(); Preferences.this.setResult(RESULT\_OK); finish();

}

});

Button cancelButton = (Button) findViewById(R.id.cancelButton);

cancelButton.setOnClickListener(new View.OnClickListener() {

public void onClick(View view) { Preferences.this.setResult(RESULT\_CANCELED); finish();

}

});

}

private void savePreferences() {

}

8. Fill in the savePreferences method to record the current preferences, based on the UI selections, to the Shared Preference object.

8. Điền vào các phương thức savePreferences để ghi lại các preferences hiện hành, dựa trên chọn lọc giao diện người dùng, đến đối tượng Shared Preference.

private void savePreferences() {

int updateIndex = updateFreqSpinner.getSelectedItemPosition(); int minMagIndex = magnitudeSpinner.getSelectedItemPosition(); boolean autoUpdateChecked = autoUpdate.isChecked();

Editor editor = prefs.edit();

editor.putBoolean(PREF\_AUTO\_UPDATE, autoUpdateChecked); editor.putInt(PREF\_UPDATE\_FREQ, updateIndex); editor.putInt(PREF\_MIN\_MAG, minMagIndex); editor.commit();

}

9. That completes the Preferences Activity. Make it accessible in the application by adding it to the application manifest.

9. Hoàn thành Hoạt động Preferences. Làm cho nó có thể truy cập vào các ứng dụng bằng cách thêm vào các biểu hiện ứng dụng.

<activity android:name=”.Preferences”

android:label=”Earthquake Preferences”>

</activity>

10. Now return to the Earthquake Activity, and add support for the new Shared Preferences file and a Menu Item to display the Preferences Activity.

Start by adding the new Menu Item. Extend the onCreateOptionsMenu method to include a new item that opens the Preferences Activity.

10. Bây giờ quay trở lại các hoạt động động đất, và thêm hỗ trợ cho các tập tin Shared Preferences mới và một mục trình đơn để hiển thị các Hoạt động Preferences.

Bắt đầu bằng cách thêm mới mục trình đơn. Mở rộng các phương pháp onCreateOptionsMenu để bao gồm một mục mới để mở Hoạt động Preferences.

static final private int MENU\_PREFERENCES = Menu.FIRST+1;

[@Override](mailto:@Override)

public boolean onCreateOptionsMenu(Menu menu) {

super.onCreateOptionsMenu(menu);

menu.add(0, MENU\_UPDATE, Menu.NONE, R.string.menu\_update);

menu.add(0, MENU\_PREFERENCES, Menu.NONE, R.string.menu\_preferences);

return true;

}

11. Modify the onOptionsItemSelected method to display the Preferences Activity when the new Menu Item is selected. Create an explicit Intent, and pass it in to the startActivityForResult method. This will launch the Preferences screen and alert the Earthquake class when the preferences are saved through the onActivityResult handler.

11. Sửa đổi các phương pháp onOptionsItemSelected để hiển thị các Hoạt động Preferences khi mục trình đơn mới được chọn. Tạo một ý định rõ ràng, và vượt qua nó để phương pháp startActivityForResult. Điều này sẽ khởi động màn hình Preferences và cảnh báo các lớp Earthquake khi đi những preferences được thông qua các xử lý onActivityResult.

private static final int SHOW\_PREFERENCES = 1;

public boolean onOptionsItemSelected(MenuItem item) {

super.onOptionsItemSelected(item);

switch (item.getItemId()) {

case (MENU\_UPDATE): { refreshEarthquakes(); return true;

}

case (MENU\_PREFERENCES): {

Intent i = new Intent(this, Preferences.class); startActivityForResult(i, SHOW\_PREFERENCES); return true;

}

}

return false;

}

12. Launch your application, and select Preferences from the Activity menu. The Preferences Activ- ity should be displayed as shown in Figure 6-1.

12. Chạy ứng dụng, và chọn Preference từ bảng chọn hoạt động. Hoạt động Preference được hiển thị giống như Hình 6.1

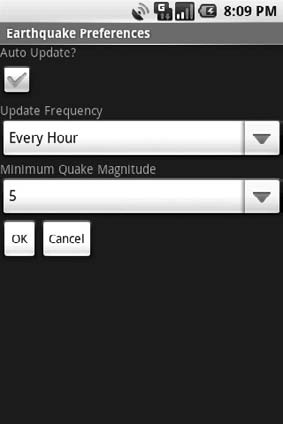


Figure 6-1

13. All that’s left is to apply the preferences to the Earthquake functionality.

Implementing the automatic updates will be left until Chapter 8, when you’ll learn how to use Services and background threads. For now, you can put the framework in place and apply the magnitude filter.

13. Tất cả những gì còn lại là áp dụng preferences cho các chức năng Earthquake.

Thực hiện cập nhật tự động sẽ được để lại cho đến Chương 8, lúc đó ta sẽ tìm hiểu làm thế nào để sử dụng dịch vụ và chủ đề nền. Bây giờ, ta có thể đặt khung tại chỗ và áp dụng bộ lọc độ lớn.

14. Start by creating a new updateFromPreferences method that reads the Shared Preference val- ues and creates instance variables for each of them.

14. Bắt đầu bằng cách tạo ra một phương pháp mới updateFromPreferences mà đọc các giá trị Shared Preference và tạo ra các biến ví dụ cho mỗi biến trong số chúng.

int minimumMagnitude = 0; boolean autoUpdate = false; int updateFreq = 0;

private void updateFromPreferences() { SharedPreferences prefs =

getSharedPreferences(Preferences.USER\_PREFERENCE, Activity.MODE\_PRIVATE);

int minMagIndex = prefs.getInt(Preferences.PREF\_MIN\_MAG, 0);

if (minMagIndex < 0)

minMagIndex = 0;

int freqIndex = prefs.getInt(Preferences.PREF\_UPDATE\_FREQ, 0);

if (freqIndex < 0)

freqIndex = 0;

autoUpdate = prefs.getBoolean(Preferences.PREF\_AUTO\_UPDATE, false); Resources r = getResources();

// Get the option values from the arrays.

// Lấy giá trị lựa chọn từ các mảng

int[] minMagValues = r.getIntArray(R.array.magnitude);

int[] freqValues = r.getIntArray(R.array.update\_freq\_values);

// Convert the values to ints.

// Chuyển các giá trị sang số nguyên

minimumMagnitude = minMagValues[minMagIndex];

updateFreq = freqValues[freqIndex];

}

15. Apply the magnitude filter by updating the addNewQuake method to check a new earthquake’s magnitude before adding it to the list.

15. Áp dụng các bộ lọc độ lớn bằng cách cập nhật các phương thức addNewQuake để kiểm tra độ mạnh của một trận động đất mới trước khi thêm nó vào danh sách.

private void addNewQuake(Quake \_quake) {

if (\_quake.getMagnitude() > minimumMagnitude) {

// Add the new quake to our list of earthquakes.

// Thêm rung động mới vào danh sách các trận động đất

earthquakes.add(\_quake);

// Notify the array adapter of a change.

// Thông báo mảng những sự thay đổi

aa.notifyDataSetChanged();

}

}

16. Override the onActivityResult handler to call updateFromPreferences and refresh the earthquakes whenever the Preferences Activity saves changes.

16. Ghi đè xử lý onActivityResult để gọi updateFromPreferences và làm mới các trận động đất bất cứ khi nào các Hoạt động Preferences lưu những thay đổi.

[@Override](mailto:@Override)

public void onActivityResult(int requestCode, int resultCode, Intent data) {

super.onActivityResult(requestCode, resultCode, data);

if (requestCode == SHOW\_PREFERENCES)

if (resultCode == Activity.RESULT\_OK) { updateFromPreferences(); refreshEarthquakes();

}

}

17. Finally, call updateFromPreferences in onCreate (before the call to refreshEarthquakes)

to ensure that the preferences are applied when the Activity first starts.

17. Cuối cùng, gọi updateFromPreferences trong onCreate (trước khi gọi đến refreshEarthquakes) để đảm bảo rằng các preferences được áp dụng khi các Hoạt động đầu tiên bắt đầu.

[@Override](mailto:@Override)

public void onCreate(Bundle icicle) { super.onCreate(icicle); setContentView(R.layout.main);

earthquakeListView = (ListView)this.findViewById(R.id.earthquakeListView);

earthquakeListView.setOnItemClickListener(new OnItemClickListener() {

public void onItemClick(AdapterView \_av, View \_v, int \_index, long arg3) {

selectedQuake = earthquakes.get(\_index);

showDialog(QUAKE\_DIALOG);

}

});

int layoutID = android.R.layout.simple\_list\_item\_1;

aa = new ArrayAdapter<Quake>(this, layoutID, earthquakes);

earthquakeListView.setAdapter(aa);

updateFromPreferences();

refreshEarthquakes();

}