INTENT TRONG LẬP TRÌNH ANDROID

Intent trong lập trình Android

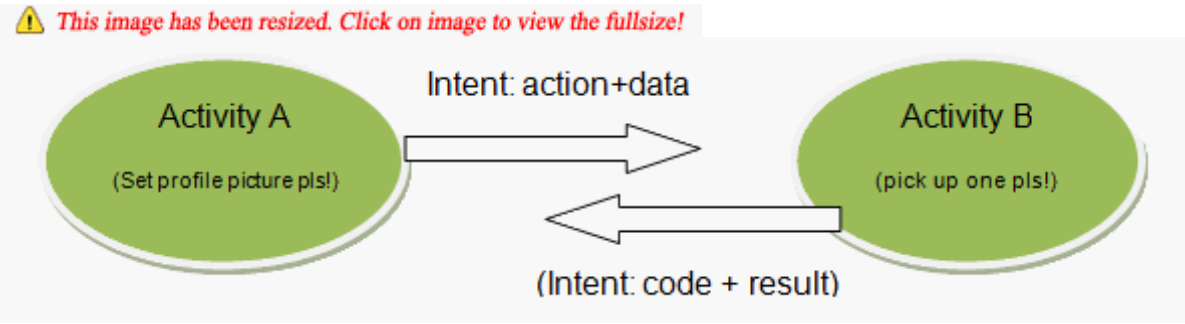
Part 1

Trong bài này mình sẽ trình bày tóm tắt những kiến thức cơ bản nhất về Intent trong lập trình Android, cách truyền nhận thông tin qua Intent và minh họa bằng một ví dụ đơn giản. Hy vọng sẽ giúp các bạn mới làm quen với Android nắm bắt được một cách nhanh chóng.

I- Intent là gì?

-Là một cấu trúc dữ liệu mô tả cách thức, đối tượng thực hiện của một Activity
-Là cầu nối giữa các Activity : ứng dụng Android thường bao gồm nhiều Activity, mỗi Activity hoạt động độc lập với nhau và thực hiện những công việc khác nhau.

Intent chính là người đưa thư, giúp chúng ta triệu gọi cũng như truyền các dữ liệu cần thiết để thực hiện một Activity từ một Activity khác. Điều này cũng giống như việc di chuyển qua lại giữa các Forms trong lập trình Windows Form



(Chú ý : trong hình vẽ trên Activity B chỉ trả về kết quả khi cần thiết. VD : giả sử Activity A nhắc người dùng chọn ảnh profile ; Activity B liệt kê các ảnh trong sdcard và cho phép người dùng chọn ảnh. Khi đó cặp "code+result" là cần thiết và có thể là "0:null" tức cancel hoặc "1:ảnh 20" tức chọn ảnh 20)

Để biết thêm về Activity xem tại đây.

-Intent là một khái niệm then chốt và đặc trưng của Android Platform. Có thể nói lập trình Android chính là lập trình intent-base.

II-Intent chứa những dữ liệu gì ?

-Intent về cơ bản là một cấu trúc dữ liệu, được mô tả trong lớp android.content.Intent

-Các thuộc tính của một đối tượng Intent :

Thuộc tính chính	Thuộc tính phụ
action -tên (string) của action mà Intent sẽ yêu cầu thực hiện -có thể là action được Android định nghĩa sẵn (built-in standard action) hoặc do người lập trình tự định nghĩa	**category** -thông tin về nhóm của action
	type -định dạng kiểu dữ liệu (chuẩn MIME) -thường được tự động xác định
data -dữ liệu mà Activity được gọi sẽ xử lý -định dạng Uri (thông qua hàm Uri.parse(data))	**component** -chỉ định cụ thể lớp sẽ thực thi Activity -khi được xác định, các thuộc tính khác trở thành không bắt buộc (optional)
	extras -chứa tất cả các cặp (key,value) do ứng dụng thêm vào để truyền qua Intent (cấu trúc Bundle)
http://developer.android.com/reference/android/content/Intent.html	

-Các action định nghĩa sẵn :

Built-in Standard Actions	
ACTION_MAIN	ACTION_ANSWER
ACTION_VIEW	ACTION_INSERT
ACTION_ATTACH_DATA	ACTION_DELETE
ACTION_EDIT	ACTION_RUN
ACTION_PICK	ACTION_SYNC
ACTION_CHOOSER	ACTION_PICK_ACTIVITY
ACTION_GET_CONTENT	ACTION_SEARCH
ACTION_DIAL	ACTION_WEB_SEARCH
ACTION_CALL	ACTION_FACTORY_TEST
ACTION_SEND	ACTION_SENDTO
Built-in Standard Broadcast Actions	
ACTION_TIME_TICK	ACTION_PACKAGE_RESTARTED
ACTION_TIME_CHANGED	ACTION_PACKAGE_DATA_CLEARED
ACTION_TIMEZONE_CHANGED	ACTION_UID_REMOVED
ACTION_BOOT_COMPLETED	ACTION_BATTERY_CHANGED
ACTION_PACKAGE_ADDED	ACTION_POWER_CONNECTED
ACTION_PACKAGE_CHANGED	ACTION_POWER_DISCONNECTED
ACTION_PACKAGE_REMOVED	ACTION_SHUTDOWN

Đây là những hằng String đã được định nghĩa sẵn trong lớp Intent. Đi kèm với nó là các Activity hay Application được xây dựng sẵn sẽ được triệu gọi mỗi khi Intent tương ứng được gửi (tất nhiên khi được cung cấp đúng data). VD:

+Dial một số phone:

PHP Code:

```
Intent dialIntent = new Intent(Intent.ACTION_DIAL, Uri.
parse("tel:123456"));
startActivity(dialIntent);
```

+Hiện danh bạ điện thoại:

PHP Code:

```
 Intent listContacts = new Intent(Intent.ACTION_VIEW,
                                                  Uri.pa
rse("content://contacts/people/");
startActivity(listContacts);
```

Đến đây chắc bạn sẽ tự hỏi những chuỗi data trong hàm Uri.parse(data) có nghĩa là gì? Đó là định dạng dữ liệu tương ứng với mỗi action (chuẩn RFC 3986). Một khi bạn đã sử dụng built-in action thì bạn phải cung cấp data cho nó theo định dạng này. Bảng dưới đây liệt kê một số định dạng và action tương ứng đã được định nghĩa sẵn:

Định dạng	Action	Mô tả
tel:phone_number	ACTION_VIEW	Mở Dial form (chưa gọi)
tel:phone_number	ACTION_CALL	Thực hiện gọi tới số phone
http://web_address https://web_address	ACTION_VIEW	Mở trình duyệt web với địa chỉ được cấp
"some_words" (string) http://web_address https://web_address	ACTION_WEB_SEARCH	Thực hiện search
sms://	ACTION_SENDTO	Gửi tin nhắn
geo:latitude,longitude geo:latitude,longitude?z=zoom geo:0,0?q=my+street+address geo:0,0?q=business+near+city	ACTION_VIEW	Mở ứng dụng Maps và chỉ tới vị trí được xác định

-Tự định nghĩa action

Về nguyên tắc bạn có thể đặt tên action của một intent là bất cứ thứ gì theo chuẩn đặt tên thông thường, hay thậm chí dùng luôn hằng action đã định nghĩa sẵn như ACTION_VIEW (hay "android.intent.action.VIEW"). Cái tên VIEW thực chất chỉ là một tên gợi tả, bạn có thể dùng nó với mục đích thực hiện một activity để … gửi mail! Tuy nhiên điều đó rõ ràng là rất "ngớ ngẩn". Thay vào đó ta hãy

dùng ACTION_SEND hay ACTION_SENDTO.

Việc đặt tên action cho intent đúng tên gợi tả còn có một ý nghĩa khác đó là app của bạn có thể được triệu gọi từ một app khác. Ví dụ bạn viết một app có activity đáp ứng intent ACTION_SEND và để chia sẻ một bức ảnh lên trang web của bạn (giống như ta làm với Facebook, Flickr etc.) Khi đó có thể app của bạn sẽ là một lựa chọn chia sẻ ảnh của người dùng điện thoại.

Intent trong lập trình Android

Part 2

III-Sử dụng Intent như thế nào?

-Các hàm thực thi Activity

Tên hàm	Mô tả
startActivity(intent) ←android.app.Activity	Thực thi activity như mô tả trong intent(không lấy kết quả trả về)
startActivityForResult(intent) ←android.app.Activity	Thực thi activity như mô tả trong intent(có lấy kết quả trả về)
sendBroadcast(intent) ←android.content.ContextWrapper	Phát tán intent tới bất kỳ thành phần BroadcastReceiver nào
startService(intent) ←android.content.ContextWrapper	Chạy một service
bindService(intent,ServiceConnection,int) ←android.content.ContextWrapper	Bind service

-Intent tường minh thực thi Activity

• Như đã trình bày ở phần II, intent có thể dùng thuộc tính phụ component để chỉ định đích danh tên lớp sẽ thực thi Activity. Để thực hiện điều này, lớp Intent cung cấp các hàm đó là setComponent(ComponentName) và setClass(Context, Class) và setClassName(Context, String) setClassName(String, String).

• Chỉ được dùng để gọi các Activities trong cùng một app

• VD:
PHP Code:

```
Intent intent = new Intent();
     intent.setClassName("ten_package", "ten_lop_ben_tr
ong_package");
     startActivity(intent);
```

-Intent không tường minh thực thi Activity

• Trong trường hợp này intent không chỉ định một lớp cụ thể mà thay vào đó dùng các dữ liệu khác (action, data, type, etc.) và để hệ thống tự quyết định xem lớp nào (app nào) sẽ thích hợp để đáp ứng intent đó.

• Thông tin action và category của activity trong một app đáp ứng intent đó phải được khai báo trong Manifest của app (AndroidManifest.xml) dưới dạng Intent-filter (tất nhiên nếu chúng ta muốn gọi một built-in action thì ta không cần quan tâm đến vấn đề này). VD:
PHP Code:

```
     <activity class=".NotesList" android:label="@strin
g/title_notes_list">
    <intent-filter>
         <action android:name="android.intent.action.GE
T_CONTENT" />
         <category android:name="android.intent.categor
y.DEFAULT" />
         <data android:mimeType="vnd.android.cursor.ite
m/vnd.google.note" />
       </intent-filter>
</activity>
```

IV-Truyền nhận thông tin giữa các Activity sử dụng intent

-Giả sử ta xây dựng một app có hai activites A và B như hình vẽ trên. Khi đó bên phái Activity A ta sẽ gọi hàm:
PHP Code:

```
     startActivity(intentA,request_code)
```

-Bên phía Activity B ta sẽ gọi hàm:

PHP Code:

```
setResult(return_code, intentB);
```

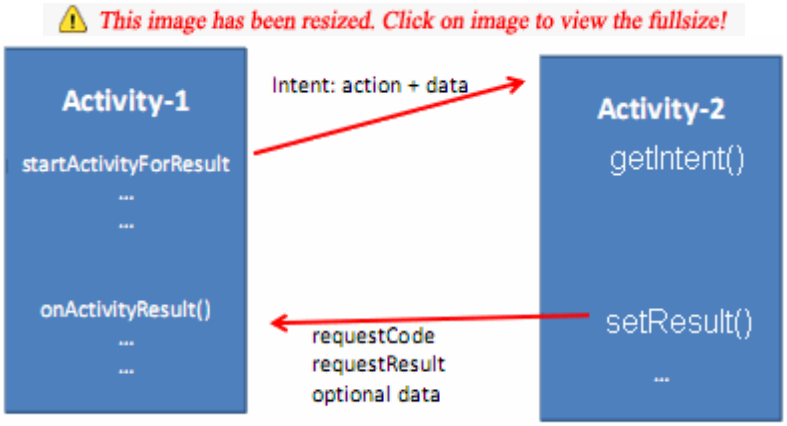
Trong phần 1, mình đã trình bày những kiến thức cơ bản về Intent. Tiếp theo mình sẽ hướng dẫn các bạn làm một Tutorial đơn giản để hiểu rõ hơn những vấn đề nêu trong lý thuyết.

-Giả sử bạn cần viết một app để tính các phần tử của một dãy số được cho theo quy luật:

PHP Code:

```
a0,b0 nhập từ bàn phím
a(n+1)=a(n)+b(n)
b(n+1)=a(n)*b(n)
```

Để thực hiện, chúng ta xây dựng hai Activities 1 và 2. Activity 1 sẽ làm nhiệm vụ lấy dữ liệu nhập vào sau đó gọi Activity 2 tính toán kết quả và lấy dữ liệu trả về. Người dùng sẽ quyết định tiếp tục tính toán hay reset lại từ đầu. Toàn bộ quá trình này được minh họa như hình vẽ dưới đây:

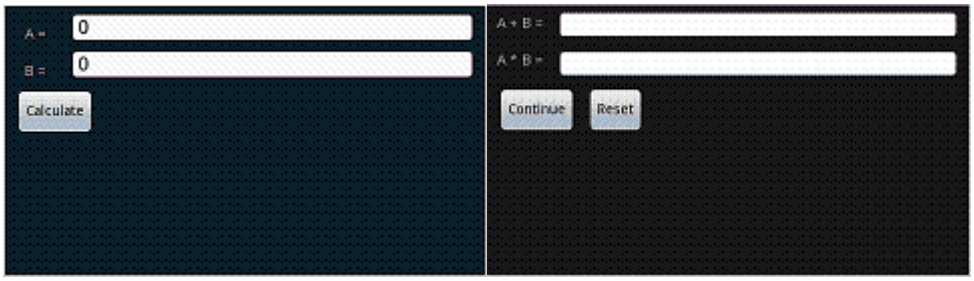


-Đầu tiên bạn tạo một New Project như sau:

PHP Code:

```
Project Name: IntentBasic
Target: bất kỳ (vd 1.5)
Package name: com.vietanddev.intent
Application name: Intent Basic
```

-Tiếp theo bạn tạo layout cho hai Activities như hình vẽ dưới



input.xml

PHP Code:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/RelativeLayout01"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
<TextView android:id="@+id/TextView01"
    android:layout_width="wrap_content"
    android:text="A = "
    android:layout_margin="20dip"
    android:layout_height="wrap_content"></TextView>

<EditText android:id="@+id/txtNum1"
    android:text="0"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_alignBottom="@id/TextView01"
    android:background="@android:drawable/editbox_background"
    android:layout_marginRight="10dip"
    android:layout_toRightOf="@id/TextView01"></EditText>
<TextView android:id="@+id/TextView02"
    android:layout_width="wrap_content"
    android:layout_below="@id/txtNum1"
    android:text="B = "
    android:layout_margin="20dip"
    android:layout_height="wrap_content"></TextView>

<EditText android:id="@+id/txtNum2"
    android:text="0"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
```

```
    android:background="@android:drawable/editbox_backg
round"
    android:layout_marginRight="10dip"
    android:layout_toRightOf="@id/TextView02"
    android:layout_alignBottom="@id/TextView02"></EditT
ext>
<Button android:id="@+id/btnGo"
    android:text="Calculate"
    android:layout_margin="10dip"
    android:layout_width="wrap_content"
    android:layout_below="@id/txtNum2"
    android:layout_height="wrap_content"></Button>
</RelativeLayout>
```

result.xml

PHP Code:

```
<RelativeLayout xmlns:android="http://schemas.android.c
om/apk/res/android"
    android:id="@+id/RelativeLayout01"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
<TextView android:id="@+id/TextView01"
    android:layout_width="wrap_content"
    android:text="A + B = "
    android:layout_margin="10dip"
    android:layout_height="wrap_content"></TextView>

<EditText android:id="@+id/txtSum"
    android:text=""
    android:layout_marginRight="10dip"
    android:layout_marginTop="5dip"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="@android:drawable/editbox_backg
round"
    android:layout_toRightOf="@id/TextView01"></EditTex
t>
<TextView android:id="@+id/TextView02"
    android:layout_width="wrap_content"
    android:layout_below="@id/TextView01"
```

```
    android:text="A * B = "
    android:layout_margin="10dip"
    android:layout_height="wrap_content"></TextView>

<EditText android:id="@+id/txtMul"
    android:text=""
    android:layout_marginRight="10dip"
    android:layout_marginTop="10dip"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/txtSum"
    android:background="@android:drawable/editbox_backg
round"
    android:layout_toRightOf="@id/TextView02"></EditTex
t>
<Button android:id="@+id/btnContinue"
    android:text="Continue"
    android:layout_margin="10dip"
    android:layout_width="wrap_content"
    android:layout_below="@id/txtMul"
    android:layout_height="wrap_content"></Button>
<Button android:id="@+id/btnReset"
    android:text="Reset"
    android:layout_marginTop="10dip"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/txtMul"
    android:layout_toRightOf="@id/btnContinue"></Button
>
</RelativeLayout>
```

LẬP TRINH TREN ANDROID

Part 3

Ví dụ 6. Trình xử lý SAX

```
import                                                  static
org.developerworks.android.BaseFeedParser.*;

public class RssHandler extends DefaultHandler{
    private List<Message> messages;
    private Message currentMessage;
    private StringBuilder builder;

    public List<Message> getMessages(){
        return this.messages;
    }
    @Override
    public  void  characters(char[]  ch,  int  start,  int
length)
            throws SAXException {
        super.characters(ch, start, length);
        builder.append(ch, start, length);
    }

    @Override
    public    void    endElement(String    uri,    String
localName, String name)
            throws SAXException {
        super.endElement(uri, localName, name);
        if (this.currentMessage != null){
            if (localName.equalsIgnoreCase(TITLE)){

currentMessage.setTitle(builder.toString());
            }                   else                    if
(localName.equalsIgnoreCase(LINK)){

currentMessage.setLink(builder.toString());
            }                   else                    if
```

```
(localName.equalsIgnoreCase(DESCRIPTION)){

currentMessage.setDescription(builder.toString());
            }                   else                    if
(localName.equalsIgnoreCase(PUB_DATE)){

currentMessage.setDate(builder.toString());
            }                   else                    if
(localName.equalsIgnoreCase(ITEM)){
                messages.add(currentMessage);
            }
            builder.setLength(0);
        }
    }

    @Override
    public void startDocument() throws SAXException {
        super.startDocument();
        messages = new ArrayList<Message>();
        builder = new StringBuilder();
    }

    @Override
    public   void   startElement(String   uri,   String
localName, String name,
            Attributes  attributes)  throws  SAXException
{
        super.startElement(uri,     localName,     name,
attributes);
        if (localName.equalsIgnoreCase(ITEM)){
            this.currentMessage = new Message();
        }
    }
}
```

Lớp `RssHandler` mở rộng lớp `org.xml.sax.helpers.DefaultHandler`. Lớp này cung cấp các thực thi mặc định, không thao tác cho tất cả các phương thức tương tự các sự kiện được tạo ra bởi trình phân tích SAX. Điều này cho phép các lớp con chỉ ghi chèn lên các

phương thức khi cần thiết. `RssHandler` có một API bổ sung, `getMessages`. Cái này trả về danh sách các đối tượng `Message` mà trình xử lý thu thập được khi nó nhận các sự kiện từ trình phân tích SAX. Nó có hai biến trong khác, một là `currentMessage` cho thể hiện `Message` đang được phân tích, và một là biến `StringBuilder` gọi là `builder` lưu trữ dữ liệu ký tự từ các nút văn bản. Các biến này đều được bắt đầu khi phương thức `startDocument` được dẫn ra khi trình phân tích gửi sự kiện tương ứng cho trình xử lý.

Hãy xem phương thức `startElement` trong Ví dụ 6. Phương thức này được gọi mỗi khi bắt gặp thẻ mở trong tài liệu XML. Bạn chỉ cần quan tâm khi nào thẻ đó là thẻ `ITEM`. Trong trường hợp đó, bạn tạo ra một `Message` mới. Bây giờ hãy nhìn vào phương thức `characters`. Phương thức này được gọi ra khi bắt gặp dữ liệu ký tự từ các nút văn bản. Dữ liệu dễ dàng được thêm vào biến `builder`. Cuối cùng hãy xem phương thức `endElement`. Phương thức này được gọi ra khi bắt gặp thẻ kết thúc. Đối với các thẻ tương ứng với các đặc tính của một `Message`, giống như `TITLE` và `LINK`, đặc tính thích hợp được thiết đặt trên `currentMessage` sử dụng dữ liệu từ biến `builder`. Nếu thẻ kết thúc là một `ITEM`, thì `currentMessage` thêm vào danh sách Messages. Đây là sự phân tích SAX rất điển hình; ở đây không có gì là duy nhất đối với Android. Vì thế nếu bạn biết cách viết một trình phân tích SAX Java, thì bạn biết cách viết một trình phân tích SAX Android. Tuy nhiên, Android SDK có bổ sung thêm một số tính năng thuận tiện vào SAX.

Phân tích SAX dễ dàng hơn

Android SDK có chứa một lớp tiện ích được gọi là `android.util.Xml`. Ví dụ 7 trình bày cách cài đặt một trình phân tích SAX với cùng lớp tiện ích như thế.

Ví dụ 7. Trình phân tích SAX Android

```
public class AndroidSaxFeedParser extends
BaseFeedParser {

    public AndroidSaxFeedParser(String feedUrl) {
        super(feedUrl);
    }
```

```
    public List<Message> parse() {
        RssHandler handler = new RssHandler();
        try {
            Xml.parse(this.getInputStream(),
Xml.Encoding.UTF_8, handler);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
        return handler.getMessages();
    }

}
```

Lưu ý là lớp này vẫn sử dụng trình xử lý SAX chuẩn, vì đơn giản bạn đã sử dụng lại `RssHandler` như trong Ví dụ 7 ở trên. Việc có thể sử dụng lại trình xử lý SAX rất tốt, nhưng nó vẫn có đôi chút phức tạp về mã trình. Bạn có tưởng tượng, nếu bạn phải phân tích một tài liệu XML phức tạp hơn rất nhiều, trình phân tích có thể trở thành mảnh đất màu mỡ cho các lỗi. Ví dụ, hãy xem lại phương thức `endElement` trong Ví dụ 6. Lưu ý cách phương thức này kiểm tra như thế nào nếu `currentMessage` có giá trị không trước khi nó cố cài đặt các thuộc tính? Bây giờ hãy nhìn vào XML mẫu trong Ví dụ 4. Lưu ý rằng có các thẻ `TITLE` và `LINK` nằm ngoài các thẻ `ITEM`. Đó là lý do tại sao kiểm tra giá trị không được đưa vào. Nếu không thì thẻ `TITLE` đầu tiên có thể gây ra một `NullPointerException`. Android bao gồm cả biến thể SAX API của chính nó (xem Ví dụ 8) loại bỏ yêu cầu bạn phải viết trình xử lý SAX của chính bạn.

Ví dụ 8. Trình phân tích SAX Android đơn giản

```
public class AndroidSaxFeedParser extends
BaseFeedParser {

    public AndroidSaxFeedParser(String feedUrl) {
        super(feedUrl);
    }

    public List<Message> parse() {
        final Message currentMessage = new Message();
```

```
        RootElement root = new RootElement("rss");
        final    List<Message>    messages    =    new
ArrayList<Message>();
        Element channel = root.getChild("channel");
        Element item = channel.getChild(ITEM);
        item.setEndElementListener(new
EndElementListener(){
            public void end() {
                messages.add(currentMessage.copy());
            }
        });

item.getChild(TITLE).setEndTextElementListener(new
EndTextElementListener(){
            public void end(String body) {
                currentMessage.setTitle(body);
            }
        });

item.getChild(LINK).setEndTextElementListener(new
EndTextElementListener(){
            public void end(String body) {
                currentMessage.setLink(body);
            }
        });

item.getChild(DESCRIPTION).setEndTextElementListener(ne
w
EndTextElementListener(){
            public void end(String body) {
                currentMessage.setDescription(body);
            }
        });

item.getChild(PUB_DATE).setEndTextElementListener(new
EndTextElementListener(){
            public void end(String body) {
                currentMessage.setDate(body);
            }
        });
```

```
            try {
                Xml.parse(this.getInputStream(),
Xml.Encoding.UTF_8,
root.getContentHandler());
            } catch (Exception e) {
                throw new RuntimeException(e);
            }
            return messages;
        }
}
```

Như đã hứa, mã phân tích SAX mới không sử dụng trình xử lý SAX. Thay vào đó nó sử dụng các lớp từ gói android.sax trong SDK. Các lớp này cho phép bạn mô hình hóa cấu trúc của tài liệu XML của bạn và thêm một trình nghe sự kiện nếu cần. Trong mã trình trên, bạn khai báo rằng tài liệu của bạn sẽ có một phần tử gốc có tên `rss` và rằng phần tử này sẽ có ba phần tử con là `channel`. Tiếp đến bạn nói rằng `channel` sẽ có ba phần tử con được gọi là `ITEM` và bạn bắt đầu gắn các trình nghe. Đối với mỗi trình nghe, bạn đã sử dụng một lớp bên trong vô danh đã thực hiện giao diện bạn quan tâm (hoặc `EndElementListner` hoặc `EndTextElementListener`). Chú ý không cần phải theo dõi dữ liệu ký tự. Việc này không chỉ đơn giản hơn mà thực sự còn hiệu quả hơn. Cuối cùng, khi bạn gọi dẫn phương thức tiện ích Xml.parse, bây giờ bạn đưa vào trình xử lý được tạo ra từ phần tử gốc.

Toàn bộ mã trình ở trên trong Ví dụ 8 thuộc loại tùy chọn. Nếu bạn thấy thoải mái với mã trình phân tích SAX chuẩn trong môi trường Java, thì bạn có thể tích vào đó. Nếu bạn muốn thử các trình bao bọc tiện lợi do Android SDK cung cấp, bạn cũng có thể sử dụng nó. Nếu bạn không muốn sử dụng SAX thì sao đây? Vẫn còn có một vài lựa chon khác. Lựa chọn đầu tiên bạn sẽ thấy đó là DOM.

LẬP TRINH TREN ANDROID

Part 4

Làm việc DOM

DOM phân tích trên Android được hỗ trợ hoàn toàn. Nó làm việc chính xác như khi nó làm việc trong mã trình Java mà bạn sẽ chạy trên máy tính để bàn hoặc trên một máy chủ. Ví dụ 9 trình bày một thực thi dựa trên DOM của giao diện trình phân tích.

Ví dụ 9. Thực thi dựa trên DOM của một trình phân tích điểm tin

```
public class DomFeedParser extends BaseFeedParser {

    protected DomFeedParser(String feedUrl) {
        super(feedUrl);
    }

    public List<Message> parse() {
        DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
        List<Message> messages = new
ArrayList<Message>();
        try {
            DocumentBuilder builder =
factory.newDocumentBuilder();
            Document dom =
builder.parse(this.getInputStream());
            Element root = dom.getDocumentElement();
            NodeList items =
root.getElementsByTagName(ITEM);
            for (int i=0;i<items.getLength();i++){
                Message message = new Message();
                Node item = items.item(i);
                NodeList properties =
item.getChildNodes();
                for (int
```

```
j=0;j<properties.getLength();j++){
                    Node property = properties.item(j);
                    String            name            =
property.getNodeName();
                    if (name.equalsIgnoreCase(TITLE)){

message.setTitle(property.getFirstChild().getNodeValue(
));
                    }              else              if
(name.equalsIgnoreCase(LINK)){

message.setLink(property.getFirstChild().getNodeValue()
);
                    }              else              if
(name.equalsIgnoreCase(DESCRIPTION)){
                        StringBuilder   text   =   new
StringBuilder();
                        NodeList         chars         =
property.getChildNodes();
                        for                          (int
k=0;k<chars.getLength();k++){

text.append(chars.item(k).getNodeValue());
                        }

message.setDescription(text.toString());
                    }              else              if
(name.equalsIgnoreCase(PUB_DATE)){

message.setDate(property.getFirstChild().getNodeValue()
);
                    }
                }
                messages.add(message);
            }
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
        return messages;
    }
```

```
}
```

Giống như ví dụ SAX đầu tiên, không có gì là cụ thể đối với Android về mã trình này. Trình phân tích DOM đọc tất cả các tài liệu XML vào bộ nhớ rồi sau đó cho phép bạn sử dụng các DOM API để chạy ngang qua cây XML, truy vấn dữ liệu mà bạn muốn. Đây là mã trình rất dễ làm, và, trong một số cách, còn đơn giản hơn cả các thực thi dựa trên SAX. Tuy nhiên, thông thường DOM tiêu tốn nhiều bộ nhớ hơn vì trước tiên mọi thứ đều được đọc vào bộ nhớ. Điều này có thể là một vấn đề trên thiết bị di động chạy Android, nhưng nó có thể đáp ứng được trong một vài trường hợp sử dụng nhất định mà dung lượng tài liệu XML sẽ không bao giờ quá lớn. Có thể điều này ngụ ý rằng các nhà phát triển Android đã đoán rằng trình phân tích SAX sẽ phổ biến hơn rất nhiều trên các ứng dụng Android, do đó các tiện ích bổ sung được cung cấp cho nó. Một loại trình phân tích XML khác cũng có trên Android, và đó là trình phân tích kéo.

Trình phân tích kéo XML

Như đã đề cập trong các phần trước, Android không cung cấp hỗ trợ cho StAX API của Java. Tuy nhiên Android lại đi kèm với một trình phân tích kéo làm việc tương tự như StAX. Nó cho phép mã ứng dụng của bạn kéo hoặc tìm kiếm các sự kiện từ trình phân tích, trái ngược với trình phân tích SAX tự động đẩy các sự kiện cho trình xử lý. Ví dụ 10 miêu tả một thực thi trình phân tích kéo của một giao diện trình phân tích điểm tin.

Ví dụ 10. Thực thi dựa trên trình phân tích kéo

```
public class XmlPullFeedParser extends BaseFeedParser {
    public XmlPullFeedParser(String feedUrl) {
        super(feedUrl);
    }
    public List<Message> parse() {
        List<Message> messages = null;
        XmlPullParser parser = Xml.newPullParser();
        try {
            // auto-detect the encoding from the stream
            parser.setInput(this.getInputStream(),
null);
```

```
            int eventType = parser.getEventType();
            Message currentMessage = null;
            boolean done = false;
            while             (eventType             !=
XmlPullParser.END_DOCUMENT && !done){
                String name = null;
                switch (eventType){
                    case XmlPullParser.START_DOCUMENT:
                        messages          =          new
ArrayList<Message>();
                        break;
                    case XmlPullParser.START_TAG:
                        name = parser.getName();
                        if
(name.equalsIgnoreCase(ITEM)){
                            currentMessage     =     new
Message();
                        }  else  if  (currentMessage  !=
null){
                            if
(name.equalsIgnoreCase(LINK)){

currentMessage.setLink(parser.nextText());
                            }          else          if
(name.equalsIgnoreCase(DESCRIPTION)){

currentMessage.setDescription(parser.nextText());
                            }          else          if
(name.equalsIgnoreCase(PUB_DATE)){

currentMessage.setDate(parser.nextText());
                            }          else          if
(name.equalsIgnoreCase(TITLE)){

currentMessage.setTitle(parser.nextText());
                            }
                        }
                        break;
                    case XmlPullParser.END_TAG:
                        name = parser.getName();
```

```
                        if  (name.equalsIgnoreCase(ITEM)
&&
currentMessage != null){

messages.add(currentMessage);
                        }             else             if
(name.equalsIgnoreCase(CHANNEL)){
                            done = true;
                        }
                        break;
                    }
                    eventType = parser.next();
                }
            } catch (Exception e) {
                throw new RuntimeException(e);
            }
            return messages;
        }
}
```

Trình phân tích kéo làm việc tương tự như trình phân tích SAX. Nó có các sự kiện tương tự (phần tử bắt đầu, phần tử kết thúc) nhưng bạn phải kéo từ chúng (`parser.next()`). Các sự kiện được gửi đi dưới dạng các mã số, vì thế bạn có thể sử dụng một case-switch đơn giản. Chú ý, thay vì nghe cho đến khi kết thúc các phần tử như trong phân tích SAX, với trình phân tích kéo, thật dễ dàng tiến hành hầu hết các xử lý ngay từ đầu. Trong mã trình trong Ví dụ 10, khi một phần tử bắt đầu, bạn có thể gọi dẫn `parser.nextText()` để kéo tất cả dữ liệu ký tự từ tài liệu XML. Điều này mang đến một sự đơn giản hóa tốt cho phân tích SAX. Cũng cần chú ý rằng bạn đặt một cờ (biến boolean `done`) để nhận biết khi nào bạn đến phần kết thúc nội dung mà bạn quan tâm. Điều này cho phép bạn sớm tạm dừng việc đọc tài liệu XML, vì bạn biết rằng mã trình sẽ không quan tâm đến phần còn lại của tài liệu. Điều này có thể rất hữu ích, đặc biệt nếu bạn chỉ cần một phần nhỏ tài liệu đang được truy cập. Bạn có thể giảm đáng kể thời gian phân tích bằng cách dừng việc phân tích càng sớm càng tốt. Hơn nữa, kiểu tối ưu hóa này đặc biệt quan trọng trên thiết bị di động nơi tốc độ kết nối có thể chậm. Trình phân tích kéo có một vài ưu điểm về hiệu năng cũng như ưu điểm sử dụng dễ dàng. Cũng có thể sử dụng nó để viết XML.

Tạo XML

Đến tận bây giờ, tôi vẫn đã và đang tập trung phân tích XML từ Internet. Tuy nhiên, thỉnh thoảng ứng dụng của bạn cần gửi XML tới một máy chủ ở xa. Hiển nhiên bạn có thể sử dụng một `StringBuilder` hoặc cái gì đó tương tự để tạo ra một chuỗi XML. Một thay thế khác nữa bắt nguồn từ trình phân tích kéo trong Ví dụ 11.

Ví dụ 11. Viết XML bằng trình phân tích kéo

```
private String writeXml(List<Message> messages){
    XmlSerializer serializer = Xml.newSerializer();
    StringWriter writer = new StringWriter();
    try {
        serializer.setOutput(writer);
        serializer.startDocument("UTF-8", true);
        serializer.startTag("", "messages");
        serializer.attribute("",              "number",
String.valueOf(messages.size()));
        for (Message msg: messages){
            serializer.startTag("", "message");
            serializer.attribute("",            "date",
msg.getDate());
            serializer.startTag("", "title");
            serializer.text(msg.getTitle());
            serializer.endTag("", "title");
            serializer.startTag("", "url");

serializer.text(msg.getLink().toExternalForm());
            serializer.endTag("", "url");
            serializer.startTag("", "body");
            serializer.text(msg.getDescription());
            serializer.endTag("", "body");
            serializer.endTag("", "message");
        }
        serializer.endTag("", "messages");
        serializer.endDocument();
        return writer.toString();
```

```
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
}
```

Lớp `XmlSerializer` là một phần trong gói giống như `XmlPullParser` được dùng trong phần trước. Thay vì kéo vào các sự kiện, nó đẩy chúng ra đến một luồng hoặc một bộ ghi. Trong trường hợp này, nó dễ dàng đẩy chúng sang một thể hiện `java.io.StringWriter`. Nó cung cấp một API đơn giản cùng với các phương thức để bắt đầu và kết thúc một tài liệu, xử lý các phần tử và thêm văn bản hoặc các thuộc tính. Đây có thể là một lựa chọn thay thế khá tốt cho việc sử dụng một `StringBuilder`, vì dễ dàng đảm bảo XML của bạn chuẩn xác.

Tổng kết

Loại ứng dụng nào bạn muốn xây dựng cho các thiết bị Android? Dù là loại nào đi nữa, nếu nó cần làm việc với dữ liệu từ Internet, thì có thể nó cần phải làm việc với XML. Trong bài viết này, bạn đã thấy rằng Android được tích hợp đi cùng với rất nhiều công cụ xử lý XML. Bạn có thể chọn lấy một trong các công cụ đó như là công-cụ-lựa-chọn của bạn, hoặc bạn có thể lựa chọn căn cứ vào trường hợp sử dụng. Thông thường sự lựa chọn an toàn là chọn cùng với SAX, và Android cung cấp cho bạn cả cách truyền thống để thực hiện SAX và một trình bao bọc tiện lợi khéo léo trên cả SAX. Nếu tài liệu của bạn nhỏ, thì có lẽ DOM là cách đơn giản hơn nên theo. Nếu tài liệu của bạn lớn, nhưng bạn chỉ cần một phần tài liệu, thì trình phân tích kéo XML có lẽ là cách hiệu quả hơn nên theo. Cuối cùng, để viết XML, gói trình phân tích kéo cũng cung cấp một cách thuận tiện để làm việc đó. Vì thế, cái mà XML của bạn cần có là gì đi nữa, thì Android SDK vẫn có cho bạn.