Làm việc với XML trên Android

Xây dựng các ứng dụng Java cho các thiết bị di động

Mức độ: Trung bình

Michael Galpin, Kiến trúc sư phần mềm, Ludi Labs

08 01 2010

Android là một hệ điều hành nguồn mở, hiện đại và là SDK cho các thiết bị di động. Với hệ điều hành này, bạn có thể tạo ra các ứng dụng di động rất mạnh. Điều này thậm chí còn trở nên hấp dẫn hơn nữa khi các ứng dụng của bạn có thể truy cập các dịch vụ Web, có nghĩa là bạn cần sử dụng ngôn ngữ của Web là: XML. Trong bài viết này, bạn sẽ thấy nhiều lựa chọn khác nhau để làm việc với XML trên Android và cách sử dụng chúng để xây dựng các ứng dụng Android của chính bạn.

Bắt đầu

Trong bài viết này, bạn học cách xây dựng các ứng dụng Android có thể làm việc với XML từ Internet. Các ứng dụng Android được viết bằng ngôn ngữ lập trình Java™, do vậy mà kinh nghiệm làm việc với công nghệ Java là điều cần phải có. Để phát triển cho Android, bạn sẽ cần đến Android SDK. Toàn bộ mã trình được trình bày trong bài viết này sẽ làm việc với bất kỳ phiên bản nào của Android SDK, nhưng phiên bản SDK 1.5_pre đã được sử dụng để phát triển mã trình. Bạn có thể phát triển các ứng dụng Android chỉ với SDK và một trình biên tập văn bản là đủ, nhưng sẽ dễ dàng hơn nhiều nếu sử dụng Android Developer Tools (ADT), là một trình bổ sung Eclipse. Đối với bài viết này, phiên bản 0.9 của ADT đã được dùng với Eclipse 3.4.2, một phiên bản Java. Xem Tài nguyên để lấy các liên kết dẫn đến tất cả các cộng cụ này.

XML trên Android

Nền tảng Android là một nền tảng phát triển di động mã nguồn mở. Nó giúp bạn truy cập vào tất cả các khía cạnh của thiết bị di động mà nó chạy trên đó, từ các đồ họa cấp thấp, đến phần cứng như là thiết bị camera trên điện thoại. Với rất nhiều thứ có thể sử dụng Android, có thể bạn sẽ tự hỏi tại sao bạn cần phiền đến XML. Đó không phải vì làm việc với XML rất thú vị; mà là nó đang làm việc với những thứ mà nó kích hoạt. XML thường được dùng như là một định dạng dữ liệu trên Internet. Nếu bạn muốn truy cập dữ liệu từ Internet, các khả năng có thể là dữ liệu sẽ ở dạng XML. Nếu bạn muốn gửi dữ liệu đến một dịch vụ Web, có thể bạn cũng cần gửi cả dữ liệu XML. Nói ngắn gọn là nếu ứng dụng Android của bạn thúc đẩy Internet, thì có thể bạn sẽ cần phải làm việc với XML. Thật may mắn là bạn có rất nhiều lựa chọn có sẵn để làm việc với XML trên Android.

Các trình phân tích XML

Một trong nhữn ưu điểm lớn nhất của nền tảng Android chính là việc nó thúc đẩy ngôn ngữ lập trình Java. Android SDK không hoàn toàn cung cấp sẵn mọi thứ cho Môi trường Thời gian chạy Java (JRE) chuẩn của bạn, nhưng nó lại hỗ trợ một phần rất đáng kể cho nó. Nền tảng Java đã và đang hỗ trợ rất nhiều cách khác nhau để làm việc với XML trong thời gian nhất định, và hầu hết các API có liên quan đến XML của Java đều được hỗ trợ đầy đủ trên Android. Ví dụ, Simple API của Java cho XML (SAX) và Document Object Model (DOM) hiện đều có sẵn trên Android. Nhiều năm qua, cả hai API này là một phần của công nghệ Java. Sản phẩm Streaming API mới đây cho XML (StAX) hiện chưa có trong Android. Tuy nhiên, Android lại cung cấp một thư viện tương đương về mặt chức năng. Điều

Các từ viết tắt thông dụng

- API: Application programming interface (Giao diện lập trình ứng dụng)
- RSS: Really Simple Syndication (Giao thức tập hợp thông tin đơn giản)
- SDK: Software Developers Kit (Bộ dụng cụ cho nhà phát triển phầm mềm)

- UI: User interface (Giao diện người dùng)
- URL: Universal Resource Locator (Địa chỉ tài nguyên)
- XML: Extensible Markup Language (Ngôn ngữ đánh dấu mở rộng)

cuối cùng là Java XML Binding API cũng không có sẵn trong Android. Chắc chắn có thể thực hiện API này trong Android. Tuy nhiên, nó lại có xu hướng là một API nặng ký, với rất nhiều thể hiện khác nhau thuộc các lớp khác nhau thường cần việc trình bày một tài liệu XML. Do vậy mà nó không lý tưởng lắm cho một môi trường bị ràng buộc chẳng hạn như thiết bị cầm tay mà Android được thiết kế để chạy trên đó. Trong các phần tiếp theo, bạn sẽ lấy một nguồn XML đơn giản có sẵn trên Internet, và xem cách phân tích nguồn đó như thế nào trong phạm vi một ứng dụng Android sử dụng các API khác nhau được nhắc đến ở trên.Trước tiên, hãy xem các phần cần thiết của ứng dụng đơn giản sẽ sử dụng XML từ Internet.

Trình đọc tin Android

Ứng dụng sẽ lấy điểm tin RSS từ trang nhà phát triển Android phổ biến Androidster và phân tách nó thành một danh sách các đối tượng Java đơn giản mà bạn có thể sử dụng để quay lại Android ListView (xem Tải về để lấy mã nguồn). Đây là hoạt động đa hình thái cổ điển — tức là các thực thi khác nhau (các thuật toán phân tích XML khác nhau) cung cấp hoạt động giống nhau. Ví dụ 1 cho bạn thấy bạn có thể mô hình hóa điều này dễ dàng như thế nào trong mã trình Java sử dụng một giao diện.

Ví dụ 1. giao diện trình phân tích điểm tin XML

```
package org.developerworks.android;
import java.util.List;

public interface FeedParser {
    List<Message> parse();
}
```

Trong Ví dụ 2, lớp `Message` là một POJO (Plain Old Java Object) cổ điển miêu tả một cấu trúc dữ liệu.

Ví dụ 2. `Message` POJO

```
public class Message implements Comparable<Message>{
    static SimpleDateFormat FORMATTER =
        new SimpleDateFormat("EEE, dd MMM yyyy HH:mm:ss Z");
    private String title;
    private URL link;
    private String description;
    private Date date;

      // getters and setters omitted for brevity
    public void setLink(String link) {
        try {
            this.link = new URL(link);
        } catch (MalformedURLException e) {
            throw new RuntimeException(e);
        }
    }

    public String getDate() {
        return FORMATTER.format(this.date);
    }

    public void setDate(String date) {
        // pad the date if necessary
        while (!date.endsWith("00")){
            date += "0";
        }
        try {
            this.date = FORMATTER.parse(date.trim());
        } catch (ParseException e) {
            throw new RuntimeException(e);
        }
    }

    @Override
    public String toString() {
             // omitted for brevity
```

```
    }

    @Override
    public int hashCode() {
            // omitted for brevity
    }

    @Override
    public boolean equals(Object obj) {
            // omitted for brevity
    }
      // sort by date
    public int compareTo(Message another) {
        if (another == null) return 1;
        // sort descending, most recent first
        return another.date.compareTo(date);
    }
}
```

Message, trong Ví dụ 2, thường rất dễ làm. Nó ẩn đi một vài trạng thái bên trong của mình bằng cách cho phép truy cập ngày tháng và các liên kết như các chuỗi đơn giản, trong khi thể hiện chúng như các đối tượng được sắp xếp một cách rõ ràng (một `java.util.Date` và một `java.net.URL`). Nó là một Value Object (Đối tượng Giá trị) cổ điển, do vậy nó thực thi `equals()` và `hashCode()` dựa trên trạng thái bên trong của nó. Nó cũng thực hiện giao diện `Comparable` vì thế bạn có thể sử dụng nó để sắp xếp (theo ngày tháng). Thực tế, dữ liệu được phân loại từ điểm tin, do vậy mà điều này không cần thiết.

Mỗi thực thi trình phân tích sẽ cần đưa một URL đến điểm tin Androidster và sử dụng cái này để mở một kết nối HTTP đến trang Androidster. Hoạt động phổ biến này được mô hình hóa một cách tự nhiên trong mã trình Java sử dụng lớp cơ sở trừu tượng như trong Ví dụ 3.

Ví dụ 3. Lớp trình phân tích điểm tin cơ bản

```
public abstract class BaseFeedParser implements FeedParser {

    // names of the XML tags
    static final String PUB_DATE = "pubDate";
    static final  String DESCRIPTION = "description";
    static final  String LINK = "link";
    static final  String TITLE = "title";
    static final  String ITEM = "item";

    final URL feedUrl;

    protected BaseFeedParser(String feedUrl){
        try {
            this.feedUrl = new URL(feedUrl);
        } catch (MalformedURLException e) {
            throw new RuntimeException(e);
        }
    }

    protected InputStream getInputStream() {
        try {
            return feedUrl.openConnection().getInputStream();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

Lớp cơ sở lưu trữ `feedUrl` và sử dụng nó để mở `java.io.InputStream`. Nếu có bất kỳ sai sót nào, đơn giản nó thả một `RuntimeException`, sao cho ứng dụng dừng hoạt động một cách nhanh chóng. Lớp cơ sở cũng xác định một vài hằng số đơn giản cho tên các thẻ. Ví dụ 4 trình bày một số nội dung mẫu từ điểm tin, qua đó bạn có thể thấy được ý nghĩa của các thẻ này.

Ví dụ 4. Điểm tin XML mẫu

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- generator="FeedCreator 1.7.2" -->

<rss version="2.0">
```

```
  <channel>

    <title>android_news</title>

    <description>android_news</description>

    <link>http://www.androidster.com/android_news.php</link>

    <lastBuildDate>Sun, 19 Apr 2009 19:43:45 +0100</lastBuildDate>

    <generator>FeedCreator 1.7.2</generator>

    <item>

      <title>Samsung S8000 to Run Android, Play DivX, Take Over the
World</title>

      <link>http://www.androidster.com/android_news/samsung-s8000-to-run-android-
play-divx-take-over-the-world</link>

      <description>More details have emerged on the first Samsung handset
to run Android. A yet-to-be announced phone called the S8000 is being
reported ...</description>

      <pubDate>Thu, 16 Apr 2009 07:18:51 +0100</pubDate>

    </item>

    <item>

      <title>Android Cupcake Update on the Horizon</title>

      <link>http://www.androidster.com/android_news/android-cupcake-update-
on-the-horizon</link>

      <description>After months of discovery and hearsay, the Android
build that we have all been waiting for is about to finally make it
out ...</description>

      <pubDate>Tue, 14 Apr 2009 04:13:21 +0100</pubDate>

    </item>

  </channel>

</rss>
```

Như bạn có thể thấy từ mẫu trong Ví dụ 4, một `ITEM` tương đương với một thể hiện `Message`. Các nút con của mục chọn (`TITLE`, `LINK` và v.v..) tương đương các đặc tính của thể hiện `Message`. Vì bạn biết điểm tin trông như thế nào rồi và có sẵn tất cả các phần phổ biến, hãy xem làm thế nào để phân tách điểm tin này sử dụng các công nghệ khác nhau có sẵn trên Android. Bạn sẽ bắt đầu với SAX.

Sử dụng SAX

Trong môi trường Java, bạn có thể thường xuyên sử dụng SAX API khi bạn muốn có một trình phân tích nhanh và muốn hạn chế tối đa việc sử dụng (footprint) bộ nhớ ứng dụng của bạn. Điều đó khiến cho cho nó rất phù hợp cho thiết bị di động chạy Android. Bạn có thể sử dụng SAX API như là từ môi trường Java, mà không cần đến những thay đổi đặc biệt cần thiết để chạy trên Android.Ví dụ 5 trình bày một thực thi SAX của giao diện `FeedParser`.

Ví dụ 5. Thực thi SAX

```
public class SaxFeedParser extends BaseFeedParser {

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

    public List<Message> parse() {
        SAXParserFactory factory = SAXParserFactory.newInstance();
        try {
            SAXParser parser = factory.newSAXParser();
            RssHandler handler = new RssHandler();
```

```
            parser.parse(this.getInputStream(), handler);
            return handler.getMessages();
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }
}
```

Nếu trước đây bạn đã sử dụng SAX, thì cái này trông cũng khá quen thuộc. Như với bất kỳ thực thi SAX nào, phần lớn các chi tiết đều nằm trong trình xử lý SAX. Trình xử lý nhận các sự kiện từ trình phân tích SAX khi nó chạy nhanh qua tài liệu XML. Trong trường hợp này, bạn vừa tạo ra một lớp mới gọi là `RssHandler` và đăng ký nó như là một trình xử lý cho trình phân tích, như trong Ví dụ 6.

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(new
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.

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.

Tải về

Tên	Kích thước	Phương thức tải
AndroidXml.zip	70KB	**HTTP**

→ Thông tin về phương thức tải

Tài nguyên

Học tập

- Develop Android applications with Eclipse (Frank Ableson,developerWorks, 02/2008): Cách dễ dàng nhất để phát triển các ứng dụng Android là sử dụng Eclipse. Hãy tìm hiểu tất cả điều này trong bài hướng dẫn này.
- Using integrated packages: Codehaus' Woodstox (Michael Galpin, developerWorks, 07/2007): Dành cho so sánh khác của SAX, DOM, và trình phân tích kéo, hãy đọc bài viết này.
- StAX'ing up XML, Part 2: Pull parsing and events (Peter Nehrer, developerWorks, December 2006): Cái nhìn sâu hơn về trình phân tích kéo XML.
- Understanding SAX (Nicholas Chase, developerWorks, 07/2003): Trở thành chuyên gia về phân tích SAX với bài hướng dẫn này.

- Understanding DOM (Nicholas Chase, developerWorks, 03/2007): Để hiểu thêm về trình phân tích, hãy đọc bài hướng dẫn này.
- Tài liệu Android SDK: Tìm hiểu bộ công cụ này để phát triển và sửa lỗi mã ứng dụng và thiết kế một UI ứng dụng.
- Open Handset Alliance: Tìm kiếm nhà tài trợ cho Android, một nhóm 47 công ty công nghệ và di động làm việc để đẩy nhanh tốc độ đổi mới trong công nghệ di động.
- Chứng chỉ XML của IBM: Tìm hiểu xem làm thế nào mà bạn có thể trở thành một Nhà phát triển có chứng chỉ IBM trong XML và các công nghệ liên quan.
- Thư viện kỹ thuật XML: Xem khu vực developerWorks XML nơi có rất nhiều các bài viết chuyên môn và các mẹo nhỏ, bài hướng dẫn, các tiêu chuẩn và Sách đỏ IBM.
- Các sự kiện kỹ thuật WebdeveloperWorks và web quảng bá: Làm quen với công nghệ trong các phiên này.
- developerWorks podcast: Nghe các bài phỏng vấn thú vị và các cuộc thảo luận dành cho các nhà phát triển phần mềm.

Lấy sản phẩm và công nghệ

- Android SDK: Tải về, truy cập tham chiếu API, và nhận tin tức mới nhất về Android từ trang web chính thức của các nhà phát triển Android.
- Dự án Mã nguồn Mở Android: Lấy mã nguồn mở cho Android.
- Eclipse IDE: Lấy phiên bản mới nhất và cho nó hoạt động.
- Các phiên bản đánh giá sản phẩm IBM: Tải về hoặc sử dụng các bản thử nghiệm trực tuyến trong IBM SOA Sandbox và bắt đầu sử dụng các công cụ phát triển ứng dụng và các sản phẩm phần mềm trung gian từ DB2®, Lotus®, Rational®, Tivoli®, và WebSphere®.

Thảo luận

- Tham gia diễn đàn thảo luận.
- Các diễn đàn thảo luận về XML: Tham gia một số cuộc thảo luận về XML.
- developerWorks blogs: Đọc các blog này và tham gia vào cộng đồng developerWorks.

Đôi nét về tác giả

Michael Galpin đã phát triển phần mềm Java một cách chuyên nghiệp từ năm 1998. Ông hiện đang làm việc cho eBay. Ông đã có bằng về toán học của Viện Công nghệ California.