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.