

## Android で XML データ処理 補足

### ●XML のメリットは？

天気予報、株価、商品情報など、今、様々なデータが XML 形式で（主にネットを通じて）配信されています。以下は、気象庁の天気予報情報を XML で配信しているサイト（<http://www.drk7.jp/weather/>）から入手したデータの一部です。

気象データ例)

```
<?xml version="1.0" encoding="UTF-8"?>
<weatherforecast>
  <title>weather forecast xml</title>
  <link>http://www.drk7.jp/weather/xml/14.xml
</link>
  <description>気象庁の天気予報情報を XML で配信。1 日 1 回 AM 6:00 ごろ更新。
</description>
  <pubDate>Sun, 6 Feb 2011 12:00:01 +0900</pubDate>
  <author>気象庁</author>
  <managingEditor>drk7.jp</managingEditor>
  <pref id="神奈川県">
    <area id="東部">
      <geo>
        <long>139.5479</long>
        <lat>35.5142</lat>
      </geo>
      <info date="2011/02/06">
        <weather>晴れのちくもり</weather>
        <weather_detail>東の風のち北の風晴れ昼過ぎからくもり所により夜雨
        </weather_detail>
        <wave>波 1 メートル</wave>
      </info>
    </area>
  </pref>
</weatherforecast>
```

---

例えば、地図データを XML 形式でインターネット上に公開することにより、その地図データを各ユーザが別々のプログラムで利用することが可能です。

同じ地図データを A さんは、マーケティングに使用し、B さんはカーナビの元データに使用するといったことが可能です。(XML データを扱うプログラムが各種用意されているため、開発も容易です。)

また、XML はオブジェクト指向と親和性の高い仕様となっているため、Java や C++ など現在主流のプログラムとの相性も良くなっています。

皆さんは Android 機器用に、どんなアプリケーションを作成してみたいでしょうか？

そのアプリケーションがどのようなものであれ、インターネット上のデータを扱うのであれば、おそらくそのアプリケーションは XML を扱う必要があります。Android には XML を扱うためのツールが豊富に用意されています。

## ●XML データを処理するクラスとメソッドの紹介

### ・ SAX について

org.xml.sax パッケージは、JDK に標準で入っている、XML を Java プログラムから処理する為のフレームワーク (クラス群) です。SAX(Simple API for Xml)では、XML 文書を順次読み込んでいく時にイベントを発生します。

XML 文書を順次読み込み、文書構造を解析しながら、必要なデータ処理を行う準備部分を Java コードで記述してみます。詳細は後で説明します。

```
//文書解析アプリの取得
SAXParserFactory factory = SAXParserFactory.newInstance();
SAXParser parser = factory.newSAXParser();
//文書解析開始
parser.parse( XML ファイル名 , イベントハンドラクラスのオブジェクト );
```

### ・ SAX によるシンプルな XML 文書の操作

SAX のプログラミングでは、XML 文書の構文解析を始める点は同じですが、構文解析を済

ませてから処理をするのではなく、構文解析を進めながら必要な情報を取得していきます。

どうやってそんなことを行うのか？という話ですが、原理は簡単です。構文解析が進むにつれ、パーサはタグやテキストデータを読み込んで認識していきます。これらのタグやテキストデータを認識できた時点でイベントを発生させる、というのが SAX パーサのポイントになります。

プログラマーはあらかじめ、コンテンツハンドラと呼ばれる特殊な形式のイベントリスナをパーサに登録しておきます。プログラマーはこのコンテンツハンドラを通じて、構文解析処理の過程で発生するイベント情報を受け取ることができるようになります。イベント情報には読み込んだタグの名前やテキストデータが含まれていますから、プログラマーはコンテンツハンドラを通じて構文解析結果を随時受け取ることができるわけです。

直感的に分かりにくいかもしれませんので、リスト 1 のサンプル文書を例に取り上げて説明してみましょう。

リスト 1 サンプル文書 (sample.xml)

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

<root>
  <data>あいうえお</data>
</root>
```

SAX パーサに対して、リスト 1 のサンプル文書を入力として与えたとしましょう。最初に、1 行目の `<root>` の開始タグが読み込まれたところで `startElement` イベントが発生します (図 1)。

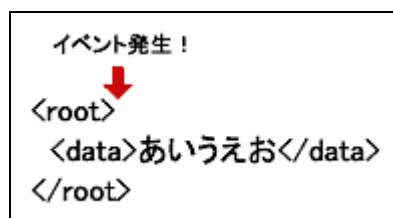


図 1 `<root>` まで読み込んだところ

次は、`<data>` の直前までのテキスト (改行を含むホワイトスペースです) を読み込んだ

ところで characters イベントが発生します (図 2)。

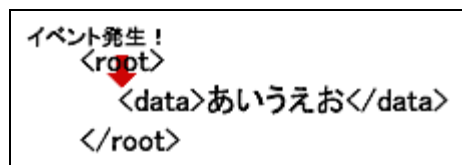


図 2 <data>の直前まで読み込んだところ

以下同様に、開始タグ<data>を読み込んだ時点、タグの内容「あいうえお」を読み込んだ時点、終了タグ</data>を読み込んだ時点……という順序でイベントが発生していきます (図 3)。

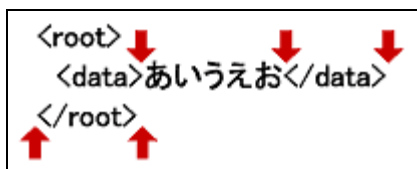


図 3 イベントが発生するポイント

XML 文書のどの部分 (タグ、テキストデータなど) を読み込んだかにより、発生するイベントは異なります。それぞれのイベントには、実際に読み込まれたタグの名前や属性の値、テキストデータの値に関する情報が付加されてきます。

これらのイベントを処理するのが、コンテンツハンドラというわけです。

## ・実際の SAX プログラミング

### [1] コンテンツハンドラの作成

SAX では、構文解析を行う過程で発生するイベントを処理するリスナ (コンテンツハンドラ) を作成する必要があります。これは、前節で挙げた `ContentHandler` のインターフェイスに定義されているメソッドを、あらかじめプログラマーがすべて実装するということになります。

ただし実際のところ、すべてのイベントを処理したいということは余りないでしょう。例えば、XML 文書内に記述されたテキストデータだけを取り出したいならば、開始タグや

終了タグに対応するイベントを処理する必要はありません。

このため SAX では、`org.xml.sax.helpers.DefaultHandler` という形で、コンテンツハンドラのデフォルト実装が提供されています。これを継承することで、必要なイベント処理のみをオーバーライドして記述することができるようになります(JAXPで生成される SAX パーサでは、そもそも `DefaultHandler` を継承したコンテンツハンドラしか利用できなくなっています)。

今回は、この `DefaultHandler` を継承し、`ContentHandler` の主なメソッドを定義したサンプルを作成してみます (リスト 2)。

---

```
import org.xml.sax.*;
import org.xml.sax.helpers.*;

public class SampleHandler extends DefaultHandler {
    // 文書開始時に呼ばれる
    public void startDocument() {
        System.out.println("XML 文書開始");
    }

    // 文書終了時に呼ばれる
    public void endDocument() {
        System.out.println("XML 文書終了");
    }

    // タグ開始時に呼ばれる
    public void startElement(String namespaceURI, String localName,
        String qName, Attributes atts) {

        System.out.println("開始タグ: " + qName);
    }

    // タグ終了時に呼ばれる
    public void endElement(String namespaceURI, String localName, String qName) {
        System.out.println("終了タグ: " + qName);
    }
}
```

```

    }

    // タグ内の値を取得すると呼ばれる
    public void characters(char[] ch, int start, int length) {
        System.out.print("タグ内の値: ");
        for (int i = 0; i < length; i++) {
            System.out.print(ch[start + i]);
        }
        System.out.println();
    }

}

} //class_end

```

---

リスト 2 サンプルのコンテンツハンドラ (SampleHandler.java)

各メソッドがどのようなイベントを処理するのかを以下にまとめておきました。いずれのメソッドでも、単純にイベントが発生したことのみに表示するだけの処理を記述しています。

#### ●startDocument, endDocument

サンプルでは省略しましたが、ドキュメントの先頭およびドキュメントの最後でもイベントは発生します。そのイベントを処理するためのメソッドです。

#### ●startElement, endElement

開始タグおよび終了タグを読み込んだ際に発生するイベントを処理するためのメソッドです。引数は、それぞれ以下の情報を表しています。

namespaceURI	ネームスペースを表す URI
localName	タグのローカル名
qName	タグの修飾名 (文字列)
Attributes	属性情報

タグの名称については、XML 名前空間の話が関連してくるので少し厄介です。本連載では XML 名前空間についての議論には触れていませんので、ここでは詳細は省きます。取りあえず、Xerces 1.2.3 の実装では、qName を参照すればタグの修飾名を取得することができますので、こちらを利用しておきます。

## ●characters

テキストデータを読み込んだ際に発生するイベントを処理するためのメソッドです。読み込んだテキストが引数で与えられます。

**ch**      読み込んだテキストデータが格納された配列  
**start**    配列内でテキストデータが配置されているオフセット  
**length**   読み込んだテキストデータの長さ

## [2] パーサファクトリおよびパーサの生成

コンテンツハンドラを準備しましたので、次は **SAX** パーサのインスタンスを生成します。

**SAX** パーサを生成するために、まずパーサファクトリを生成します。このパーサファクトリを用いて、**SAX** パーサのインスタンスを生成するのです（リスト 3）。

```
try {  
  
    SAXParserFactory factory = SAXParserFactory.newInstance();  
    SAXParser parser = factory.newSAXParser();  
  
} catch (SAXException e) {  
} catch (ParserConfigurationException e) {}
```

リスト 3   パーサファクトリおよびパーサの生成

このように 2 段階のステップを踏むようになっているのは、もちろん使用する **SAX** のライブラリによらず、統一したインターフェイスでパーサを生成するためです。

## [3] 構文解析処理の呼び出し

パーサを生成できたら、あとは **XML** 文書とコンテンツハンドラを引数に構文解析の処理 **parse** を呼び出すだけです（リスト 4）。

```
try {  
  
    String url = "..."; // XML 文書の実際の URL
```

```

DefaultHandler handler = new SampleHandler();

parser.parse(url, handler);

} catch (IOException e) {
} catch (SAXException e) {
}

```

リスト 4 パース処理の呼び出し

SAXParser の parse には、幾つかのバリエーションがあります。これらは構文解析を行う XML 文書をどのようにして与えるかという点、すなわち第 1 引数の型が異なります。

InputStream や File オブジェクトを引数として与えることができるようになっていますが、リスト 4 では XML 文書の URL を文字列で与えるものを使用しています。

リスト 1 のサンプル文書を、実際にこのハンドラの例で実行してみます。ここで示したコードは部分的なものなので、少し書き足す必要がありますが、ぜひ実際に書いてみてください。

参考までに、実行結果を以下に示しておきます。

#### 実行結果

---

XML 文書開始

開始タグ: root

タグ内のデータ:

開始タグ: data

タグ内のデータ: あいうえお

終了タグ: data

タグ内のデータ:

終了タグ: root

XML 文書

---



まず、最初と最後に `startDocument` と `startElement` が呼ばれていることを確認してください。

あとは順に `startElement`、`characters`、`startElement`…と続いていきます。この部分の呼び出しの順序は、図 1、図 2、図 3 で見たとおりですので、対応がとれているか確認してみてください。

## SAX って簡単？

今回は、SAX を用いて XML 文書を読むために必要な一通りの手順について見てみました。割とあっけなかったのではないのでしょうか？ このあたり、Simple API for XML の名前のとおりだといえなくもありません。

ただし、今回は XML 文書を最初から最後まで順に読んでみるという非常に簡単なサンプルを試してみただけです。

---

サンプルコード掲載)

### ●SampleHandler.java

```
package saxDemo;

import org.xml.sax.*;
import org.xml.sax.helpers.*;

public class SampleHandler extends DefaultHandler {
    // 文書開始時に呼ばれる
    public void startDocument() {
        System.out.println("XML 文書開始");
    }

    // 文書終了時に呼ばれる
    public void endDocument() {
        System.out.println("XML 文書終了");
    }
}
```

```

// タグ開始時に呼ばれる
public void startElement(String namespaceURI, String localName,
                        String qName, Attributes atts) {

    System.out.println("開始タグ: " + qName);
}

// タグ終了時に呼ばれる
public void endElement(String namespaceURI, String localName,
                      String qName) {

    System.out.println("終了タグ: " + qName);
}

// タグ内の値を取得すると呼ばれる
public void characters(char[] ch, int start, int length) {
    System.out.print("タグ内の値: ");
    for (int i = 0; i < length; i++) {
        System.out.print(ch[start + i]);
    }
    System.out.println();
}

}

} //class_end

```

---

#### ● 実行クラス SaxProcessMain.java

```

package saxDemo;

import java.io.IOException;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

```

```

public class SaxProcessMain {

    public static void main(String[] args) {
        try {
            //XML 文書解析アプリ (→パーサ) を取得
            SAXParserFactory factory = SAXParserFactory.newInstance();
            SAXParser parser = factory.newSAXParser();
            // XML 文書の URL パス名
            String url = "sample.xml";
            //コンテンツ イベントハンドルクラスの利用
            DefaultHandler handler = new SampleHandler();
            // 解析開始とともにコンソールに結果を表示
            parser.parse(url, handler);

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

    }

}

} //class_end

```

---

●XML データ例 sample.xml

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

```
<在庫リスト>
```

```
<商品>
```

```
<商品名>りんご</商品名>
```

```
<単価>520</単価>
```

```
</商品>
```

```
<商品>
```

```
<商品名>バナナ</商品名>
<単価>200</単価>
</商品>
<商品>
  <商品名>みかん</商品名>
  <単価>380</単価>
</商品>
</在庫リスト>
```

---

## [実行結果]

XML 文書開始

開始タグ: 在庫リスト

タグ内の値:

開始タグ: 商品

タグ内の値:

開始タグ: 商品名

タグ内の値: りんご

終了タグ: 商品名

タグ内の値:

開始タグ: 単価

タグ内の値: 520

終了タグ: 単価

タグ内の値:

終了タグ: 商品

タグ内の値:

.....

終了タグ: 在庫リスト

XML 文書終了

●twitter のタイムラインデータ例

1. <?xml version="1.0" encoding="UTF-8"?>
2. <statuses>
3. <status>
4.     <created\_at>Tue Apr 07 22:52:51 +0000 2009</created\_at>
5.     <id>1472669360</id>
6.     <text>At least I can get your humor through tweets. RT @abdur: I don't mean this in a bad way, but genetically speaking your a cul-de-sac.</text>
7.     <source><a href="http://www.tweetdeck.com/">TweetDeck</a></source>
8.     <truncated>>false</truncated>
9.     <in\_reply\_to\_status\_id></in\_reply\_to\_status\_id>
10.    <in\_reply\_to\_user\_id></in\_reply\_to\_user\_id>
11.    <favorited>>false</favorited>
12.    <in\_reply\_to\_screen\_name></in\_reply\_to\_screen\_name>
13.    <user>
14.        <id>1401881</id>
15.        <name>Doug Williams</name>
16.        <screen\_name>dougw</screen\_name>
17.        <location>San Francisco, CA</location>
18.        <description>Twitter API Support. Internet, greed, users, dougw and opportunities are my passions.</description>
19.        <profile\_image\_url>http://s3.amazonaws.com/twitter\_production/profile\_images/59648642/avatar\_normal.png</profile\_image\_url>
20.        <url>http://www.igudo.com</url>
21.        <protected>>false</protected>
22.        <followers\_count>1027</followers\_count>
23.        <profile\_background\_color>9ae4e8</profile\_background\_color>
24.        <profile\_text\_color>000000</profile\_text\_color>
25.        <profile\_link\_color>0000ff</profile\_link\_color>
26.        <profile\_sidebar\_fill\_color>e0ff92</profile\_sidebar\_fill\_color>
27.        <profile\_sidebar\_border\_color>87bc44</profile\_sidebar\_border\_color>
28.        <friends\_count>293</friends\_count>
29.        <created\_at>Sun Mar 18 06:42:26 +0000 2007</created\_at>
30.        <favourites\_count>0</favourites\_count>

```
31.    <utc_offset>-18000</utc_offset>
32.    <time_zone>Eastern Time (US & Canada)</time_zone>
33.    <profile_background_image_url>http://s3.amazonaws.com/twitter_produc
tion/profile_background_images/2752608/twitter_bg_grass.jpg</profile_backgro
und_image_url>
34.    <profile_background_tile>>false</profile_background_tile>
35.    <statuses_count>3390</statuses_count>
36.    <notifications>>false</notifications>
37.    <following>>false</following>
38.    <verified>>true</verified>
39.  </user>
40.  <geo/>
41. </status>
42. </statuses>
```

---

## ●楽天商品検索 API とは(version:2010-09-15)

参考: <http://webservice.rakuten.co.jp/api/itemsearch/>

楽天商品検索 API は、楽天市場の商品（共同購入商品・オークション商品・フリマ商品・楽天オークションの個人間オークション商品は除く。）の情報を取得することが可能な API です。デベロッパーはキーワードでの商品検索をはじめ、ショップ別・ジャンル別の絞り込み検索も可能となります。

## ■リクエスト URL

[http://api.rakuten.co.jp/rws/3.0/rest?\[/parameter\]=\[value\]...](http://api.rakuten.co.jp/rws/3.0/rest?[/parameter]=[value]...)

フィールド名 keyword, sort に対応する[value]は UTF-8 で URL エンコードされている必要があります。(URL エンコーは、URI 表記において使用禁止である値を使う際に行われる符号化のこと)

(リクエスト URL 全体をエンコードするのではなく、[value]部分を個別にエンコードしてください。)

たとえば、「福袋」という検索キーワードで検索し、結果を価格が安い順に並べたい (sort=+itemPrice) 場合のリクエスト URL は下記になります。

(実際には改行せずに 1 行につなげてリクエストしてください。)

```
http://api.rakuten.co.jp/rws/3.0/rest?
developerId=[YOUR_developerID]
&operation=ItemSearch
&version=2010-09-15
&keyword=%E7%A6%8F%E8%A2%8B
&sort=%2BitemPrice
```

---

※テキストのサンプルコード掲載

- main.xml
  - XML\_DataParserAndroid.java
  - twitterFeedHandle.java
  - RakutenFeedHandle.java
- 

実行結果例（ツイートボタンを押下すると、トータル件数、日付・テキスト内容が表示）



## ① main.xml

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

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <TextView android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="ツイート ID(検索キーワード)を入力してください : " />

    <EditText android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:id="@+id/txtUser" />

    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal">

        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/btnTWITTER "
            android:text="ツイート !" /></Button>

        <Button android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/btnRAKUTEN "
            android:text="楽天人気リスト" /></Button>

    </LinearLayout>

    <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
```



```

        android:id="@+id/ScrollView01"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">

        <TextView android:layout_width="fill_parent"
                android:layout_height="wrap_content" android:text="デフォ
                ルト内容"
                android:layout_gravity="center_horizontal"
                android:id="@+id/txtData" />

    </ScrollView>

</LinearLayout>

```

---

## ② XML\_DataParserAndroid.java

```

package xml.andorid;

import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLEncoder;

import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class XML_DataParserAndroid extends Activity {
    private TextView tvData;
    private EditText etUser;

```

```

private Button btnTWITTER;
private Button btnRAKUTEN;

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

    // httpClient を作成する
    tvData = (TextView) findViewById(R.id.txtData);
    etUser = (EditText) findViewById(R.id.txtUser);

    // XML で定義した UI インスタンスを読み込む
    btnTWITTER = (Button) findViewById(R.id. btnTWITTER);
    btnTWITTER.setOnClickListener(new Button.OnClickListener() {
        public void onClick(View v) {
            // ツイートの XML 内容を読み込むメソッドを呼ぶ
            parseTwitterXml();
        }
    });

    // XML で定義した UI インスタンスを読み込む
    btnRAKUTEN = (Button) findViewById(R.id. btnRAKUTEN);
    btnRAKUTEN.setOnClickListener(new Button.OnClickListener() {
        public void onClick(View v) {
            // 楽天サイトの XML 内容を読み込むメソッドを呼ぶ
            // parseRakutenXml();
        }
    });
}

// ツイートの XML 内容を読み込む
void parseTwitterXml() {
    try {
        URL feedUrl = null;
        try {

```

```

        // ★ポイント★
        // ツイーターAPI を使って、内容を取得する URL を初期化する
        feedUrl = new URL("http://twitter.com/statuses/user_timeline/"
                           + etUser.getText() + ".xml");
    } catch (MalformedURLException e1) {
        // エラー内容を表示する
        tvData.setText(e1.getMessage());
    }

    // SAX ツールを使う
    SAXParserFactory factory = SAXParserFactory.newInstance();

    // パーサーを生成する
    SAXParser parser = factory.newSAXParser();

    // ハンドルを生成する
    twitterFeedHandler tfh = new twitterFeedHandler();
    // サーバにアクセスし、XML データをダウンロードして、パーサーを実行する
    parser.parse(feedUrl.openConnection().getInputStream(), tfh);

    // 実行結果を表示する
    tvData.setText(tfh.getResults());
} catch (Exception e) {
    // エラー内容を表示する
    tvData.setText(e.getMessage());
}
} // parseTwitterXml0_end

// 楽天ウェブサービスの RESTXML 内容を読み込むメソッド定義
/*
void parseRakutenXml() {
    try {
        // URL インスタンスを初期化する
        URL feedUrl = null;
        try {
            // ★ポイント★

```

```

        // URL エンコードを使って、内容を取得する URL を指定する
String enString = URLEncoder.encode(txtData.getText().toString());
        feedUrl = new URL(
// 楽天ウェブサービス API 関数構成：HTTP + rakuten サイト名

"http://api.rakuten.co.jp/rws/3.0/rest?" +

        // developID（楽天ウェブサービスの API 関数構成の要素 1）

"developerId=903a68d611779d97c35749e355e8deed" +

        // オペレーション ID（楽天ウェブサービスの API 関数構成の要素 2）

"&operation=ItemSearch" +

        // API バージョン ID（楽天ウェブサービスの API 関数構成の要素 3）

"&version=2010-09-15" +

        // 検索キーワード（楽天ウェブサービスの API 関数構成の要素 4）
        "&keyword=" + enString +

        // ソート順（楽天ウェブサービスの API 関数構成の要素 5）

"&sort=%2BitemPrice" +

        // 検索件数（楽天ウェブサービスの API 関数構成の要素 6）
        "&hits=5");

    } catch (MalformedURLException e1) {
        // エラー内容を表示する
        tvData.setText(e1.getMessage());
    }

    // SAX ツールを使う
SAXParserFactory factory = SAXParserFactory.newInstance();

```



```

@Override
public void startElement(String namespaceURI, String localName,
                        String qName, Attributes atts) throws SAXException {

    try {
        //status タグの確認
        if (localName.equals("status")) {
            this.sb = new StringBuilder("");
            bStore = true;
        }
        if (localName.equals("user")) {
            bStore = false;
        }
        if (localName.equals("text")) {
            this.sb = new StringBuilder("");
        }
        if (localName.equals("created_at")) {
            this.sb = new StringBuilder("");
        }
    } catch (Exception ee) {
        Log.d("error in startElement", ee.getStackTrace().toString());
    }
}

```

```

@Override
public void characters(char ch[], int start, int length) {

    if (bStore) { //記録すべき文字列は蓄積する
        String theString = new String(ch, start, length);
        this.sb.append(theString);
    }
}

```

```

@Override
public void endElement(String namespaceURI, String localName,
    String qName)
    throws SAXException {

    if (bStore) {
        if (localName.equals("created_at")) {

            ret += "日付: " + sb.toString() + "¥n";
            sb = new StringBuilder("");
            return;
        }

        if (localName.equals("user")) {
            bStore = true;
        }

        if (localName.equals("text")) {
            ret += "内容: " + sb.toString() + "¥n¥n";
            sb = new StringBuilder("");
            return;
        }
    }
    if (localName.equals("status")) {
        howMany++;
        bStore = false;
    }
}

String getResults() {
    return "このツイートを取りました！¥n つぶやきが " +
        howMany + " 個あります¥n¥n" + ret;
}
}

```

---

#### ④ RakutenFeedHandle.java

```
package xml.andorid;

import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

import android.util.Log;

public class RakutenFeedHandle extends DefaultHandler {

    //一時保存用文字ビルダ
    StringBuilder sb = null;
    //出力文字列
    String ret = "";
    //記録対象か否か判定するフラグ
    boolean bOutput = false;
    //件数計数器
    int howMany = 0;

    @Override
    public void characters(char ch[], int start, int length) {

        if (bOutput) {
            String theString = new String(ch, start, length);
            this.sb.append(theString);
        }
    }

    @Override
    public void startElement(String namespaceURI, String localName,
        String qName, Attributes atts) throws SAXException {

        try {
            if (localName.equals("itemName")) {
```



```

        this.sb = new StringBuilder("");
        bOutput = true;
    }
    if (localName.equals("shopName")) {
        this.sb = new StringBuilder("");
        bOutput = true;
    }
} catch (Exception e) {

    Log.d("error in startElement", e.getStackTrace().toString());
}
}

@Override
public void endElement(String namespaceURI, String localName,
                        String qName)
                        throws SAXException {

    if (bOutput) {
        if (localName.equals("itemName")) {
            ret += "品名:" + sb.toString() + "¥n";
            sb = new StringBuilder("");
        }
        if (localName.equals("shopName")) {
            ret += "店名: " + sb.toString() + "¥n¥n";
            sb = new StringBuilder("");
        }
    }
    if (localName.equals("Item")) {
        howMany++;
        bOutput = false;
    }
}
}

```

```
//取得した文字列データを返す
public String getResults() {
    return "実行成功です.¥n 検索結果： " + howMany
        + " 件あります¥n¥n" + ret;
}

}
```

---

以上