About This Article

This article gives a introduction to XML parsing in Android. This article does not explain XML in details. Developers are requested to know XML before reading this article. This article describes about packages available in Android for various types of XML parsing. Sample code snippet demonstrating usage of KXML Parser is provided at the end of this article.

Scope:

This article is intended for Android developers wishing to develop mobile applications that use XML. It assumes good knowledge of Android, Java programming language and XML. This article focuses on XML parsing and therefore explaining the Android technology and XML is out of the scope in this documentation.

Introduction

XML is a way of representing text and data in a format that can be processed without much human or machine intelligence. Information formatted in XML can be exchanged across platforms, languages, and applications, and can be used with a wide range of development tools and utilities.

The design goals of XML emphasize simplicity, generality, and usability over the Internet. Android supports XML application programming interfaces (APIs) for processing of XML languages.

This article explains the various API's available in Android for XML processing. This article assumes that the developer is well equipped with the Eclipse IDE and has knowledge of Android, Java programming language and XML.

XML Parser

XML Parsing is of various types

1. Tree Model Parser

A. Document Object Model:
Document Object Model (DOM) parser is a tree model parser. DOM parser reads through the entire document, builds the entire XML document representation in memory and then hands the calling program the whole chunk of memory. DOM parser occupies extensive memory.

2. Stream Based Parser (Event based)

A. Push Parser:
A Push Parser reads through the document and as the parser encounters elements in an XML, it notifies the application through callback methods (listener objects). SAX parser is one such example of push parser.
B. Pull Parser:
A Pull Parser is opposite of push parser. Parser provides data only when the application requests it. The application drives the parser through the document by repeatedly requesting the next piece. Example of Pull Parser is StAX API

3. Data Binding XML Parser

Another form of XML processing API is XML data binding, where XML data is made available as a hierarchy of custom, strongly typed classes, in contrast to the generic objects created by a Document Object Model parser.

Java's Simple API for XML (SAX) and the Document Object Model (DOM) are both available on Android. Both of these APIs have been part of Java technology for many years. The newer Streaming

API for XML (StAX) is not available in Android. However, Android provides a functionally equivalent library. Finally, the Java XML Binding API is also not available in Android. This API could surely be implemented in Android.

Which parser to use in application depends on the characteristics of the application and XML documents?

SAX Parser

SAX is a lexical, event-driven interface in which a document is read serially and its contents are reported as callbacks to various methods on a handler object of the user's design. SAX is fast and efficient to implement, but difficult to use for extracting information at random from the XML, since it tends to burden the application author with keeping track of what part of the document is being processed.

SAX parser is available in Android **javax.xml.parsers** package. You can use the SAX API as-is from the Java environment, with no special modifications needed to run on.
Creating of SAX parser is done using the **SAXParser, SAXParserFactory** class present in **javax.xml.parsers**package and **XMLReader** class present in **org.xml.sax** package.

```
/* Create a URL we want to load some xml-data from. */
URL url = new URL("http://example.com/example.xml");

/* Get a SAXParser from the SAXPArserFactory. */
SAXParserFactory spf = SAXParserFactory.newInstance();
SAXParser sp = spf.newSAXParser();

/* Get the XMLReader of the SAXParser we created. */
XMLReader xr = sp.getXMLReader();
/* Create a new ContentHandler and apply it to the XML-Reader*/
MyDefaultHandler myDefaultHandler = new MyDefaultHandler();

/*set Content Handlet*/
xr.setContentHandler(myDefaultHandler );

/* Parse the xml-data from our URL. */
URL url = new URL("http://innovator.samsungmobile
                                 .com/servlet/rss/samsung_s60_symbian_best.xml?
platformId=1");
xr.parse(new InputSource(url.openStream()));
/* Parsing has finished. */

OR

/* Parse the xml-data from InputSource. */
String xmlStringContent = "<abc></abc>test";
InputSource is = new InputSource();
```

```
is.setCharacterStream(new StringReader(xmlStringContent));
xr.parse(is);
```

As with any SAX implementation, most of the details are in the SAX handler. The handler receives events from the SAX parser as it rips through the XML document. The handler in Android is **DefaultHandler** available in package**org.xml.sax.helpers**.

Developer needs to create handler that extends the **org.xml.sax.helpers.DefaultHandler** and override the necessary methods that correspond to the events raised by the SAX parser. The necessary methods that needs to be overridden are

```
public void startDocument() throws SAXException {}


public void endDocument() throws SAXException {}


public void startElement(String uri, String localName, String qName, Attributes attributes)
throws SAXException {}


public void  endElement(String  uri, String  localName, String  qName) throws SAXException
{}


public void  characters(char[] ch, int start, int length) throws SAXException {}
```

startDocument() and **endDocument()** methods are called at the start/end of each document. When the Parser reaches an opening tag, like , the startElement() method gets called. In this case, localName will be "platformId". The attributes variable will hold any associated attribute information: atts.getValue("name") will return "Android".

When the parser reaches a closing tag, like , the endElement() method gets called: In this case, localName will be "platformId".

In between an opening and closing tag, there can be a string, like <platformId name=“Android”>1</platformId>. The SAXParser reads in the string, one character at a time, but buffers method calls to the handler characters() method.

Android SAX Parser

Besides Java SAX API, Android provides a framework that makes it easy to write efficient and robust SAX handlers. Android SAX parser framework is present in android.sax package.

The android.sax package contains 5 interfaces namely ElementListener, EndElementListener, EndTextElementListener, StartElementListener, TextElementListener and 2 classes namely Element, RootElement.

Android SAX parser framework allows you to model the structure of your XML document and add an event listener as needed. To create a SAX Content Handler, you first need to describe the structure of the XML you are interested in, using the RootElement and Element classes.

For example parsing an xml file like shown below

```
<rss version="2.0
 <item>
   <title>test
 <pubDate>Sun, 15 Jun 2000 14:13:46 +0200
   <info
       url="http://xyz.com/abc.ogg"
       fileSize="37506213" type="application/ogg">
 </item>
</rss>
```

The above xml has **title, pubDate** and **info** as elements which are between the opening and closing tags of the item elements, which defines them as child elements of each **item** element. The item elements are likewise child elements of the **rss** element.
Declare your xml have a root element called rss

```
RootElement root = new RootElement("rss");
```

Then declare our interest in each of the item and info child elements using the **requireChild()** method of the RootElement class :

```
Element item = root.requireChild("item");
Element info = item.requireChild("info");
```

Next set some 'listeners' for each of the elements that you want to catch. For each listener, you used an anonymous inner class that implemented the interface that you were interested in (either **EndElementListner or EndTextElementListener**). Notice there was no need to keep track of character data. Not only is this simpler, but it is actually more efficient.
Finally when everything is in place, create the SAX Content Handler and pass it and the inputStream in to the**XML.parse()** method present in the **android.util** package

```
/* Parse the xml-data from our URL. */
URL url = new URL("http://xyz.com/.../");
InputStream inputStream = url.openStream();
Xml.parse(inputStream, Xml.Encoding.UTF-8, root.getContentHandler());
```

DOM Parser

The DOM parser reads the entire XML document into memory and then allows you to use the DOM APIs to transverse the XML tree, retrieving the data that you want.

DOM parsing on Android is provided in **javax.xml.parsers** package; same package for SAX parser. Creating DOM XML parser is done using the **javax.xml.parsers.DocumentBuilderFactory** class

```
DocumentBuilderFactory builderFactory = DocumentBuilderFactory.newInstance();
try {
    /* Parse the xml-data from our URL. */
    URL url = new URL("http://xyz.com/.../");
    InputStream inputStream = url.openStream();
    /*Get Document Builder*/
    DocumentBuilder builder = builderFactory.newDocumentBuilder();
    Document dom = builder.parse(inputStream);
} catch (Exception e) { }
```

Once the DOM object has been obtained, you can start traversing the elements and attributes present in the XML. Accessing the elements and attributes of the XML is done using the interfaces present in the **org.w3c.dom** package.
A DOM object contains a lot of different nodes connected in a tree-like structure. At the top is the Document object. The Document object has a single root element, which is returned by

calling **getDocumentElement()**

```
Element rootElement = dom.getDocumentElement();
```

The root element has children which can be elements, comments, processing instructions, characters etc. You get the children of an element like this:

```
NodeList items = root.getElementsByTagName(ITEM);
```

Returns a NodeList of all descendant Elements with a given tag name, in document order. Then subsequent nodes can be obtained like this

```
Node item = items.item(i);
```

The NodeList interface provides the abstraction of an ordered collection of nodes. The Nodes in the NodeList are accessible via an integral index, starting from 0.

```
for (int i=0;i<items.getLength();i++) {
  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();
  }
}
```

The **getChildNodes()** method is used to obtain a list of the children of that node. This is returned as a new NodeList object. One of methods present in NodeList class is the **getLength()** method, used to obtain the number of child nodes, in order to iterate through them.

Pull Parser

Android does not provide support for Java's StAX API. However, Android does come with a pull parser that works similarly to StAX.

The pull parser is present in **org.xmlpull.v1** package. It allows your application code to pull or seek events from the parser, as opposed to the SAX parser that automatically pushes events to the handler.
A pull parser is created using the XML class present in the **android.util** package.

```
XmlPullParser parser = Xml.newPullParser();
```

A pull parser works similarly to a SAX parser however you have to pull from them using **parser.next()** and**nextToken()** method. While next() provides access to high level parsing events, nextToken() allows access to lower level tokens.
The current event state of the parser can be determined by calling the **getEventType()** method. Initially, the parser is in the **START_DOCUMENT** state.
The method **next()** advances the parser to the next event. The int value returned from next determines the current parser state and is identical to the value returned from following calls to **getEventType()**.
The following event types are seen by **next()** method
START_TAG
An XML start tag was read.

TEXT
Text content was read; the text content can be retrieved using the getText() method. (when in validating mode next() will not report ignorable whitespace, use nextToken() instead)

END_TAG
An end tag was read

END_DOCUMENT
No more events are available

The **nextToken()** method provides more detailed access to the document structure, including components such as processing instructions, comments, entity references, and more.

```
XmlPullParser parser = Xml.newPullParser();
try {
    /* Parse the xml-data from our URL. */
    URL url = new URL("http://xyz.com/.../");
    InputStream inputStream = url.openStream();
    /*auto detect*/
    parser.setInput(this.getInputStream(), null);
    int eventType = parser.getEventType();
    boolean done = false;
    while (eventType != XmlPullParser.END_DOCUMENT && !done){
        switch (eventType){
             case XmlPullParser.START_DOCUMENT:
         //process
            break;
             case XmlPullParser.START_TAG:
                  String name = parser.getName();
                 if (name.equalsIgnoreCase(ITEM)){
                         //process
          }
            break;
            case XmlPullParser.END_TAG:
         //when done with the processing set done to true.
          done = true;
            break;
}
eventType = parser.next();
}
```

XmlResourceParser

Android has its own XML pull parser, XmlResourceParser, specifically tailored to efficiently parse its internal compiled XML format. XmlResourceParser is an interface present in android.content.res package.

XmlResourceParser is a standard XmlPullParser interface, as well as an extended AttributeSet interface and an additional close() method on this interface for the client to indicate when it is done reading the resource.

To use this parser, create xml folder in res folder. Create an instance and pass it to data file resource present in res/xml folder.

```
XmlResourceParser xrp = activity.getResources().getXml(R.xml.myxml);
```

Here R.xml.test represents the myxml.xml file present in res/xml folder. The returned XmlResourceParser is an instance of XmlPullParser, and it also implements java. util.AttributeSet.

XmlResourceParser is a pull parser which works pretty much the same way as pull parser.

```
XmlResourceParser xrp = activity.getResources().getXml(R.xml.myxml);
xrp.next();
int eventType = xrp.getEventType();
while (eventType != XmlPullParser.END_DOCUMENT)   {
    if(eventType == XmlPullParser.START_DOCUMENT)  {
    } else if(eventType == XmlPullParser.START_TAG)    {
    } else if(eventType == XmlPullParser.END_TAG)  {
    } else if(eventType == XmlPullParser.TEXT)   {
    }
    eventType = xrp.next();
 }
```

Sample Example

The sample example shows various tyepes of parser usage

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="samsung.parsers.xml"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="2" />
    <uses-permission android:name="android.permission.INTERNET">

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".XMLParser" android:label="XMLParser">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".MessageList" android:label="RSS-FEEDS">
                <intent-filter>
                <category android:name="android.intent.category.SAMPLE_CODE" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Files stored in res/layout folder

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">

    <Button android:text="SAX"
    android:id="@+id/buttonSAX"
    android:layout_height="wrap_content"
    android:layout_width="fill_parent"
    android:layout_gravity="center">
    </Button>

    <Button android:text="Android SAX"
    android:id="@+id/buttonAndroidSAX"
    android:layout_height="wrap_content"
    android:layout_width="fill_parent">
    </Button>

    <Button android:text="DOM"
    android:id="@+id/buttonDOM"
    android:layout_height="wrap_content"
    android:layout_width="match_parent">
        </Button>

    <Button android:text="XML Pull"
    android:id="@+id/buttonXMLPull"
    android:layout_height="wrap_content"
    android:layout_width="match_parent">
    </Button>

    <Button android:text="XMLResourceParser"
    android:id="@+id/buttonXMLResource"
    android:layout_height="wrap_content"
    android:layout_width="match_parent">
    </Button>

</LinearLayout>
```

row.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
```

```
        android:id="@+id/TextView01" android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
```

xmlPage.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="wrap_content"
    android:layout_height="wrap_content">

    <ListView android:id="@+id/android:list"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

Create a res/xml folder and place the myxml.xml in it.

myxml.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
    <channel>
        <title>android_news
        <description>android_news
        <link>http://www.xyz.com/android.php
        <lastBuildDate>Thu, 12 May 2011 11:28:03 +0100
        <item>
            <title>How To create Hello World
            <link>http://www.xyz.com/android/How To create Hello World
            <description>This document shows how to create
            <pubDate>Sun, 21 Oct 2010 02:15:41 +0200
        </item>
    </channel>
</rss>
```

Files present in the src folder.

XMLParser.java

```
package samsung.parsers.xml;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
```

```java
public class XMLParser extends Activity implements OnClickListener{

private Button btnSAX;
private Button btnAndroidSAX;
private Button btnDOM;
private Button btnXMLPull;
private Button btnXMLResource;

public static final String PARSER_TYPE = "ParserType";

public static final int CLASSIC_SAX = 1;
public static final int ANDROID_SAX = 2;
public static final int DOM = 3;
public static final int XML_PULL = 4;
public static final int XMLRESOURCEPARSER = 5;

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

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.main);
 btnSAX = (Button)findViewById(R.id.buttonSAX);
 btnSAX.setOnClickListener(this);
 btnAndroidSAX = (Button)findViewById(R.id.buttonAndroidSAX);
 btnAndroidSAX.setOnClickListener(this);
 btnDOM = (Button)findViewById(R.id.buttonDOM);
 btnDOM.setOnClickListener(this);
 btnXMLPull = (Button)findViewById(R.id.buttonXMLPull);
 btnXMLPull.setOnClickListener(this);
 btnXMLResource = (Button)findViewById(R.id.buttonXMLResource);
 btnXMLResource.setOnClickListener(this);
}

@Override
public void onClick(View v) {
 if(v == btnSAX) {
   Intent intent = new Intent(this.getApplication(),MessageList.class);
   intent.putExtra(PARSER_TYPE, CLASSIC_SAX);
   startActivity(intent);
  }else
  if(v == btnAndroidSAX) {
        Intent intent = new Intent(this.getApplication(),MessageList.class);
        intent.putExtra(PARSER_TYPE, ANDROID_SAX);
        startActivity(intent);
   }else
   if(v == btnDOM)      {
        Intent intent = new Intent(this.getApplication(),MessageList.class);
        intent.putExtra(PARSER_TYPE, DOM);
        startActivity(intent);
   }else
   if(v == btnXMLPull)  {
        Intent intent = new Intent(this.getApplication(),MessageList.class);
        intent.putExtra(PARSER_TYPE, XML_PULL);
        startActivity(intent);
   }else
   if(v == btnXMLResource) {
        Intent intent = new Intent(this.getApplication(),MessageList.class);
        intent.putExtra(PARSER_TYPE, XMLRESOURCEPARSER);
        startActivity(intent);
   }
  }
}
```

MessageList.java

```
package samsung.parsers.xml;

import java.io.InputStream;
import java.io.StringWriter;
import java.net.URL;
import java.net.URLConnection;
import java.util.ArrayList;
import java.util.List;

import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlSerializer;
import android.app.ListActivity;
import android.content.res.Resources;
import android.content.res.XmlResourceParser;
import android.os.Bundle;
import android.util.Log;
import android.util.Xml;
import android.widget.ArrayAdapter;

public class MessageList extends ListActivity{

private List<message></message> messages;
private URL feedURL;
private URLConnection feedURLConnection;
private InputStream is;

public static final String feedUrl = "http://www.androidster.com/android_news.rss";

@Override
public void onCreate(Bundle icicle) {
  super.onCreate(icicle);
  setContentView(R.layout.xmlpage);
  Bundle extras = getIntent().getExtras();
  int value = extras.getInt(XMLParser.PARSER_TYPE);

  if(value!= XMLParser.XMLRESOURCEPARSER) {
        try {
      feedURL = new URL(feedUrl);
          feedURLConnection = feedURL.openConnection();
          is = feedURLConnection.getInputStream();
         } catch (Exception e) {
           throw new RuntimeException(e);
         }
         if(value == XMLParser.CLASSIC_SAX)
                loadSAXFeed();
         else if(value == XMLParser.ANDROID_SAX)
             loadAndroidSAXFeed();
         else if(value == XMLParser.DOM)
                loadDOMFeed();
         else if(value == XMLParser.XML_PULL)
                loadXMLPullFeed();
   } else
   if(value == XMLParser.XMLRESOURCEPARSER)
                        loadXMLResourceParser();
}

private void loadXMLResourceParser(){
  try {
    Resources res = this.getResources();
    XmlResourceParser parser = res.getXml(R.xml.myxml);
    parser.next();
    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></message>();
           break;
       case XmlPullParser.START_TAG:
                 name = parser.getName();
                 if (name.equalsIgnoreCase(XMLParser.ITEM)){
                        currentMessage = new Message();
                 } else if (currentMessage != null){
                        if (name.equalsIgnoreCase(XMLParser.LINK)){
                          currentMessage.setLink(parser.nextText());
                         } else if (name.equalsIgnoreCase(XMLParser.DESCRIPTION)){
                          currentMessage.setDescription(parser.nextText());
                         } else if (name.equalsIgnoreCase(XMLParser.PUB_DATE)){
                          currentMessage.setDate(parser.nextText());
                         } else if (name.equalsIgnoreCase(XMLParser.TITLE)){
                          currentMessage.setTitle(parser.nextText());
                     }
                  }
                break;
                case XmlPullParser.END_TAG:
                        name = parser.getName();
                        if (name.equalsIgnoreCase(XMLParser.ITEM) && currentMessage !=
null){
                                messages.add(currentMessage);
                        } else if (name.equalsIgnoreCase(XMLParser.CHANNEL)){
                                done = true;
                        }
                break;
         }
         eventType = parser.next();
        }
   } catch (Exception e) {
        throw new RuntimeException(e);
   }
   String xml = writeXml();
   List<string></string> titles = new ArrayList<string></string>(messages.size());
   for (Message msg : messages){
         titles.add(msg.getTitle());
        }
        ArrayAdapter<string></string> adapter = new ArrayAdapter<string></string>(this,
R.layout.row,titles);
          this.setListAdapter(adapter);
    }

        private void loadAndroidSAXFeed(){
                AndroidSaxFeedParser parser = new AndroidSaxFeedParser();
                long start = System.currentTimeMillis();
        messages = parser.parse(is);
        long duration = System.currentTimeMillis() - start;
        String xml = writeXml();
        List<string></string> titles = new ArrayList<string></string>(messages.size());
        for (Message msg : messages){
                titles.add(msg.getTitle());
        }
        ArrayAdapter<string></string> adapter = new ArrayAdapter<string></string>(this,
R.layout.row,titles);
        this.setListAdapter(adapter);
        }

        private void loadDOMFeed(){
                DomFeedParser parser = new DomFeedParser();
                long start = System.currentTimeMillis();
        messages = parser.parse(is);
        long duration = System.currentTimeMillis() - start;
        String xml = writeXml();
        List<string></string> titles = new ArrayList<string></string>(messages.size());
        for (Message msg : messages){
                titles.add(msg.getTitle());
        }
        ArrayAdapter<string></string> adapter = new ArrayAdapter<string></string>(this,
R.layout.row,titles);
        this.setListAdapter(adapter);
        }

        private void loadXMLPullFeed(){
                XmlPullFeedParser parser = new XmlPullFeedParser();
```

```
                  long start = System.currentTimeMillis();
         messages = parser.parse(is);
         long duration = System.currentTimeMillis() - start;
         String xml = writeXml();
         List<string></string> titles = new ArrayList<string></string>(messages.size());
         for (Message msg : messages){
                  titles.add(msg.getTitle());
         }
         ArrayAdapter<string></string> adapter =
                  new ArrayAdapter<string></string>(this, R.layout.row,titles);
         this.setListAdapter(adapter);

         }

         private void loadSAXFeed(){
         try{
                  long start = System.currentTimeMillis();
                  SaxFeedParser parser = new SaxFeedParser();
                  messages = parser.parse(is);
                  long duration = System.currentTimeMillis() - start;
                  String xml = writeXml();
                  List<string></string> titles = new
ArrayList<string></string>(messages.size());
                  for (Message msg : messages){
                           titles.add(msg.getTitle());
                  }
                  ArrayAdapter<string></string> adapter = new
ArrayAdapter<string></string>(this, R.layout.row,titles);
                  this.setListAdapter(adapter);
         } catch (Throwable t){
         }
    }

         private String writeXml(){
                  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);
                  }
         }

}
```

Message.java

```
package samsung.parsers.xml;

import java.net.MalformedURLException;
import java.net.URL;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class 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;

         public String getTitle() {
                  return title;
         }
```

```
	public void setTitle(String title) {
		this.title = title.trim();
	}

	public URL getLink() {
		return link;
	}

	public void setLink(String link) {
		try {
			this.link = new URL(link);
		} catch (MalformedURLException e) {
			throw new RuntimeException(e);
		}
	}

	public String getDescription() {
		return description;
	}

	public void setDescription(String description) {
		this.description = description.trim();
	}

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

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

	public Message copy(){
		Message copy = new Message();
		copy.title = title;
		copy.link = link;
```

```
                  copy.description = description;
                  copy.date = date;
                  return copy;
         }

}
```

RssHandler.java

```
package samsung.parsers.xml;
import java.util.ArrayList;
import java.util.List;

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

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

         public List<message></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(XMLParser.TITLE)){
                                    currentMessage.setTitle(builder.toString());
                           } else if (localName.equalsIgnoreCase(XMLParser.LINK)){
                                    currentMessage.setLink(builder.toString());
                           } else if (localName.equalsIgnoreCase(XMLParser.DESCRIPTION)){
                                    currentMessage.setDescription(builder.toString());
                           } else if (localName.equalsIgnoreCase(XMLParser.PUB_DATE)){
                                    currentMessage.setDate(builder.toString());
                           } else if (localName.equalsIgnoreCase(XMLParser.ITEM)){
                                  messages.add(currentMessage);
                         }
                         builder.setLength(0);
                }
        }

        @Override
        public void startDocument() throws SAXException {
                super.startDocument();
                messages = new ArrayList<message></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(XMLParser.ITEM)){
                         this.currentMessage = new Message();
                }
        }
}
```

SaxFeedParser.java

```
package samsung.parsers.xml;

import java.io.InputStream;
import java.util.List;

import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

public class SaxFeedParser {

        public SaxFeedParser(){

        }

        public List<message></message> parse(InputStream is) {
                SAXParserFactory factory = SAXParserFactory.newInstance();
                try {
                         SAXParser parser = factory.newSAXParser();
                         RssHandler handler = new RssHandler();
                         parser.parse(is, handler);
```

```
                        return handler.getMessages();
                } catch (Exception e) {
                        throw new RuntimeException(e);
                }
        }
}
```
AndroidSaxFeedParser.java
```
package samsung.parsers.xml;

import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

import android.sax.Element;
import android.sax.EndElementListener;
import android.sax.EndTextElementListener;
import android.sax.RootElement;
import android.util.Xml;

public class AndroidSaxFeedParser{



        public AndroidSaxFeedParser() {

        }



        public List<message></message> parse(InputStream is) {
                final Message currentMessage = new Message();
                RootElement root = new RootElement(XMLParser.RSS);
                final List<message></message> messages = new
ArrayList<message></message>();
                Element channel = root.getChild(XMLParser.CHANNEL);
                Element item = channel.getChild(XMLParser.ITEM);
                item.setEndElementListener(new EndElementListener(){
                        @Override
                        public void end() {
                                messages.add(currentMessage.copy());
                        }
                });
                item.getChild(XMLParser.TITLE).setEndTextElementListener(new
EndTextElementListener(){
                        @Override
```

```
                           public void end(String body) {
                                    currentMessage.setTitle(body);
                           }
                  });
                  item.getChild(XMLParser.LINK).setEndTextElementListener(new
EndTextElementListener(){
                           @Override
                           public void end(String body) {
                                    currentMessage.setLink(body);
                           }
                  });
                  item.getChild(XMLParser.DESCRIPTION).setEndTextElementListener(new
EndTextElementListener(){
                           @Override
                           public void end(String body) {
                                    currentMessage.setDescription(body);
                           }
                  });
                  item.getChild(XMLParser.PUB_DATE).setEndTextElementListener(new
EndTextElementListener(){
                           @Override
                           public void end(String body) {
                                    currentMessage.setDate(body);
                           }
                  });
                  try {
                           Xml.parse(is, Xml.Encoding.UTF_8, root.getContentHandler());
                  } catch (Exception e) {
                           throw new RuntimeException(e);
                  }
                  return messages;
         }
}
```

DomFeedParser.java

```
package samsung.parsers.xml;

import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

public class DomFeedParser {

         public DomFeedParser() {

         }


         public List<message></message> parse(InputStream is) {
                  DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
                  List<message></message> messages = new ArrayList<message></message>();
                  try {
                           DocumentBuilder builder = factory.newDocumentBuilder();
                           Document dom = builder.parse(is);
                           Element root = dom.getDocumentElement();
                           NodeList items = root.getElementsByTagName(XMLParser.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(XMLParser.TITLE)){

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

message.setLink(property.getFirstChild().getNodeValue());
                                             } else if
(name.equalsIgnoreCase(XMLParser.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(XMLParser.PUB_DATE)){

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

XmlPullFeedParser.java

```
package samsung.parsers.xml;

import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

import org.xmlpull.v1.XmlPullParser;

import android.util.Log;
import android.util.Xml;

public class XmlPullFeedParser {

        public XmlPullFeedParser() {

        }



        public List<message></message> parse(InputStream is) {
                List<message></message> messages = null;
                XmlPullParser parser = Xml.newPullParser();
                try {
                        parser.setInput(is, 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></message>();
                                                      break;
                                             case XmlPullParser.START_TAG:
                                                      name = parser.getName();
                                                      if
(name.equalsIgnoreCase(XMLParser.ITEM)){
                                                               currentMessage = new
Message();
                                                      } else if (currentMessage != null){
                                                               if
(name.equalsIgnoreCase(XMLParser.LINK)){

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

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

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

currentMessage.setTitle(parser.nextText());
                                                               }
                                                      }
                                                      break;
                                             case XmlPullParser.END_TAG:
                                                      name = parser.getName();
                                                      if
(name.equalsIgnoreCase(XMLParser.ITEM) && currentMessage != null){
                                                               messages.add(currentMessage);
                                                      } else if
(name.equalsIgnoreCase(XMLParser.CHANNEL)){
                                                               done = true;
```

```
                        }
                        break;
                }
                eventType = parser.next();
            }
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
        return messages;
    }
}
```

ref:http://developer.samsung.com/android/technical-docs/Parsing-XML-in-Android