XML parsning i Java

Denne artikel beskriver hvordan man parser XML i Java.

Den beskriver W3C DOM, SAX og JDOM.

Den forudsætter kendskab til Java og XML.

Skrevet den **15. Feb 2010** af **arne_v** I kategorien **Programmering / Java** | ★★★☆☆

Historie:
V1.0 - 20/01/2004 - original
V1.1 - 31/01/2004 - forbedret formatering
V1.2 - 09/02/2004 - flere ændringer af formatering
V1.3 - 07/04/2004 - nævn at der er en anden artikel om skrivning af XML
V1.4 - 20/08/2005 - tilføj JDOM med SAXBuilder
V1.5 - 26/12/2008 - mindre ændringer og tilføjelse af links
V1.6 - 14/02/2010 - smårettelser

W3C DOM/SAX/JDOM

Der er 3 typer XML parsere til Java:
* W3C DOM parsere
* SAX parsere
* JDOM parser

En W3C DOM parser opbygger et såkaldt DOM træ af noder og er defineret af W3C (som står bag HTML, XML og en masse andre standarder).
En W3C DOM parser giver stor fleksibilitet idet man kan køre frem og tilbage i træet samt ændre i træet. Til gengæld kræver den også meget memory (3-5 x filens størrelse er ikke unormalt).

En SAX parser er event push drevet d.v.s. at den læser filen sekventielt og kalder metoder undervejs når den starter på et element, afslutter et element etc.. Man kan kun læse og kun sekventielt. Til gengæld er den hurtig og bruger næsten ingen memory.

JDOM parseren bygger sit træ udfra et W3C DOM træ, men har et API som er noget mere Java venligt. Det er ikke en egentlig standard. Jeg kan kun opfordre til at prøve den. Den er faktisk god.

W3C DOM og SAX parsere er indbygget i J2SE 1.4 og nyere. For ældre versioner kan man hente Apache Xerces her:
http://xml.apache.org/xerces2-j/

JDOM kan hentes her:
http://www.jdom.org/

Det er værd at bemærke at SUN Java 1.4 brugte Crimson XML parser mens Java 1.5 og nyere bruger Xerces (med ændrede package navne).

Hvis man vil bruge en anden XML parser end default kan man bruge:
  System.setProperty("javax.xml.parsers.DocumentBuilderFactory",
"org.apache.xerces.jaxp.DocumentBuilderFactoryImpl");
  System.setProperty("javax.xml.parsers.DocumentBuilderFactory",
"org.apache.crimson.jaxp.DocumentBuilderFactoryImpl");
  etc.

Eksempler

Lad os se nogle konkrete eksempler.

Vi vil parse følgende simple XML fil:

test.xml

```
<?xml version='1.0' standalone='yes'?>
<medlemmer>
    <medlem no="1">
        <navn>Niels Nielsen</navn>
        <adresse>Nellikevej 19</adresse>
    </medlem>
    <medlem no="2">
        <navn>Jens Jensen</navn>
        <adresse>Jagtvej 17</adresse>
    </medlem>
    <medlem no="3">
        <navn>Ole Olsen</navn>
        <adresse>Omfartsvejen 13</adresse>
    </medlem>
</medlemmer>
```

Output skal være:

```
no=1
navn=Niels Nielsen
adresse=Nellikevej 19
no=2
navn=Jens Jensen
adresse=Jagtvej 17
no=3
navn=Ole Olsen
adresse=Omfartsvejen 13
```

W3C DOM

ParseW3CDOM.java

```
import java.io.File;
```

```
import java.io.IOException;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.FactoryConfigurationError;
import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

public class ParseW3CDOM {
    private final static String XML_FILE = "C:\\test.xml";
    public static void main(String[] args) {
        try {
            // læs fra fil til DOM træ
            DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
            DocumentBuilder db = dbf.newDocumentBuilder();
            Document doc = db.parse(new File(XML_FILE));
            // lav liste over alle elementer af typen 'medlem'
            NodeList elements = doc.getElementsByTagName("medlem");
            for (int i = 0; i < elements.getLength(); i++) {
                Node element = (Element) elements.item(i);
                // find attributten no
                String no = ((Element) element).getAttribute("no");
                String name = "";
                String address = "";
                // lav liste over alle childs og find navn og adresse
                NodeList subelements = element.getChildNodes();
                for (int j = 0; j < subelements.getLength(); j++) {
                    String tag = subelements.item(j).getNodeName();
                    if (tag.equals("navn")) {
                        name =
subelements.item(j).getFirstChild().getNodeValue();
                    }
                    if (tag.equals("adresse")) {
                        address =
subelements.item(j).getFirstChild().getNodeValue();
                    }
                }
                System.out.println("no=" + no);
                System.out.println("navn=" + name);
                System.out.println("adresse=" + address);
            }
        } catch (FactoryConfigurationError e) {
            e.printStackTrace();
        } catch (ParserConfigurationException e) {
            e.printStackTrace();
        } catch (SAXException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
```

```
        return;
    }
}
```

W3C DOM kan godt virke lidt besværligt.

SAX

ParseSAX.java

```
import java.io.IOException;

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

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

public class ParseSAX {
    private final static String XML_FILE = "C:\\test.xml";
    public static void main(String[] args) {
        try {
            // process fil med MySaxParser som event handler
            SAXParserFactory spf = SAXParserFactory.newInstance();
            SAXParser sp = spf.newSAXParser();
            XMLReader xr = sp.getXMLReader();
            xr.setContentHandler(new MySaxParser());
            xr.parse(XML_FILE);
        } catch (FactoryConfigurationError e) {
            e.printStackTrace();
        } catch (ParserConfigurationException e) {
            e.printStackTrace();
        } catch (SAXException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

class MySaxParser extends DefaultHandler {
    private StringBuffer sb = new StringBuffer("");
    // akkumuler tekst (vigtigt: kan kaldes flere gang for en text node)
    public void characters(char buf[], int offset, int len) throws
SAXException {
        sb.append(new String(buf, offset, len));
        return;
```

```
    }
    // process start element, hvis 'medlem' find attribut no
    public void startElement(String namespaceURI, String localName, String
rawName, Attributes atts) throws SAXException {
        if (rawName.equals("medlem")) {
            String no = atts.getValue("no");
            System.out.println("no=" + no);
        }
        return;
    }
    // process end element, find navnog adresse
    public void endElement(String namespaceURI, String localName, String
rawName) throws SAXException {
        if (rawName.equals("navn")) {
            String name = sb.toString().trim();
            System.out.println("navn=" + name);
        }
        if (rawName.equals("adresse")) {
            String address = sb.toString().trim();
            System.out.println("adresse=" + address);
        }
        sb = new StringBuffer();
        return;
    }
}
```

SAX kan hurtigt blive til noget ustruktureret spagetti.

JDOM

ParseJDOM.java

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.List;

import org.jdom.Document;
import org.jdom.Element;
import org.jdom.JDOMException;
import org.jdom.adapters.DOMAdapter;
import org.jdom.adapters.XercesDOMAdapter;
import org.jdom.input.DOMBuilder;

public class ParseJDOM {
    private final static String XML_FILE = "C:\\test.xml";
    public static void main(String[] args) {
        try {
            DOMAdapter da = new XercesDOMAdapter();
            org.w3c.dom.Document w3cdoc = da.getDocument(new
FileInputStream(XML_FILE), false);
```

```
            DOMBuilder b = new DOMBuilder();
            Document doc = b.build(w3cdoc);
            List list = doc.getRootElement().getChildren();
            for (int i = 0; i < list.size(); i++) {
                Element elm = (Element) list.get(i);
                String no = elm.getAttributeValue("no");
                String name = elm.getChild("navn").getText();
                String address = elm.getChild("adresse").getText();
                System.out.println("no=" + no);
                System.out.println("navn=" + name);
                System.out.println("adresse=" + address);
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (JDOMException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Det fremgår klart at JDOM er nemmere at bruge end W3C DOM.

JDOM kan også opbygge sit træ med en en SAXBuilder.

```

import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.List;

import org.jdom.Document;
import org.jdom.Element;
import org.jdom.JDOMException;
import org.jdom.input.SAXBuilder;

public class ParseJDOMSAX {
    private final static String XML_FILE = "C:\\test.xml";
    public static void main(String[] args) {
        try {
            SAXBuilder b = new SAXBuilder();
            Document doc = b.build(new FileReader(XML_FILE));
            List list = doc.getRootElement().getChildren();
            for (int i = 0; i < list.size(); i++) {
                Element elm = (Element) list.get(i);
                String no = elm.getAttributeValue("no");
                String name = elm.getChild("navn").getText();
                String address = elm.getChild("adresse").getText();
                System.out.println("no=" + no);
```

```
                System.out.println("navn=" + name);
                System.out.println("adresse=" + address);
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (JDOMException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Jeg kender ikke nok til JDOM til at kunne udtale mig om forskellen i performance med DOMBuilder og SAXBuilder.

Videre

Ovenstående eksempler skulle gerne have givet lidt forståelse for de forskellige parsere, hvad de er gode til og hvad de ikke er gode til samt hvordan de kodes i praksis.

Artiklen http://www.eksperten.dk/artikler/245 "Mere XML i Java" beskriver:
- udskrift af W3C DOM og JDOM træer
- ændring af W3C DOM og JDOM træer
- W3C DOM Walker
- select med XPath i W3C DOM og JDOM

Artiklen http://www.eksperten.dk/artikler/1261 "Nye Java XML API'er" beskriver:
- StAX
- JAXB
- XStream

Kommentar af simonvalter d. 27. Jan 2004 | 1

Fine eksempler, jeg havde ikke nogen problemer med at bruge det efter at have læst dette.

Kommentar af ferrari_brian d. 15. Feb 2006 | 2

Kommentar af soaphovedet d. 04. Mar 2004 | 3

Fin artikel, jeg kunne snildt lave en parser efter at have læst dette.
Dog virker getFirstChild() i "subelements.item(j).getFirstChild().getNodeValue()" redundant i dette eksempel.
Artikelen kunne desuden godt bruge et eksempel på en xml writer.

Kommentar af abatabat d. 30. May 2006 | 4

jeg kunne let få det til at virke efter at havde læst artiklen, men kunne også godt bruge et eksempel på en

xml writer.