Denne guide er oprindeligt
udgivet på Eksperten.dk

COMPUTERWORLD

Mere XML i Java

Denne artikel beskriver brug af XML i Java udover parsning (som er beskrevet i artiklen "XML parsning i Java").

Den beskriver bl.a. udskrivning og ændring af XML.

Den forudsætter kendskab til Java og XML samt lidt kendskab til XML parsning i Java.

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

[vigtigt: artikel http://www.eksperten.dk/guide/245 er samme artikel som denne - der gik
koks i det i forbindelse med 2 store nedbrud på Eksperten
for mange år siden - først ville jeg ikke slette en af dem af hensyn til
dem som havde "betalt" for artiklen og nu beholder jeg duplikaterne af hensyn til afgivne kommentarer]

Historie:
V1.0 - 03/04/2004 - original
V1.1 - 07/04/2004 - understrege at teorien er beskrivet i den første artikel
V1.2 - 25/07/2004 - tilføje lidt flere forklaringer
V1.3 - 12/12/2004 - opdatere fra JDOM B8 til JDOM 1.0 (der er faktisk ændringer
i JDOM interfacet)
V1.4 - 20/08/2005 - tilføje brug af XPath
V1.5 - 26/12/2006 - tilføje standard løsninger som har erstattet Xerces specifikke
løsninger og tilføje links
V1.6 - 14/02/2010 - smårettelser

Indledning

For beskrivelse af W3C DOM og JDOM henvises til artiklen
http://www.eksperten.dk/artikler/100 "XML parsning i Java". Der er hele teorien. Det forudsættes
at teorien og parse teknikken er kendt. Her vil vi fokusere på brug
af W3C DOM og JDOM til andet end parsning.

SAX vil ikke blive omtalt da processingen af de indlæste data
sker løbende i forbindelse med parsningen og er 100%
applikations specifik.

Eksemplerne vil bruge følgende 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>
    <medlem no="4"/>
</medlemmer>
```

Udskrift W3C DOM træ

Når man nu har lært at læse XML filer ind er det jo naturligt at
ville kunne skrive dem ud igen.

Der er forskellige måder at udskrive et W3C DOM træ på:
- selv skrive noget kode
- bruge noget Xerces specifik kode
- bruge noget standard kode hvis ens JAXP version er tilstrækkelig ny
- bruge XSLT

Alle 3 metoder vil blive vist.

Først gør det selv koden som består af en enkelt metode writeXML som
man kalder med en node. Metoden udskriver så start tag med attributter,
kalder sig selv rekursivt for child elementer og udskriver slut tag.
Når den kaldes med document root, så udskriver den hele DOM træet.

WriteW3CDOMCustom.java

```
import java.io.IOException;
import java.io.PrintStream;

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.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

public class WriteW3CDOMCustom {
    public static void writeXML(PrintStream ps, String ind, Node parent) {
        switch (parent.getNodeType()) {
            case Node.ELEMENT_NODE :
                ps.print(ind + "<" + parent.getNodeName());
                NamedNodeMap atts = parent.getAttributes();
```

```
                for (int i = 0; i < atts.getLength(); i++) {
                    ps.print(" " + atts.item(i).getNodeName() +
                             "=\"" + atts.item(i).getNodeValue() + "\"");
                }
                NodeList childs = parent.getChildNodes();
                if (childs.getLength() == 0) {
                    ps.println("/>");
                } else {
                    ps.println(">");
                    for (int i = 0; i < childs.getLength(); i++) {
                        writeXML(ps, ind + "   ", (Node) childs.item(i));
                    }
                    ps.println(ind + "<" + parent.getNodeName() + "/>");
                }
                break;
            case Node.TEXT_NODE :
                if(!parent.getNodeValue().trim().equals("")) {
                    ps.println(ind + parent.getNodeValue().trim());
                }
                break;
            default :
                // nothing
        }
        return;
    }
    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("C:\\test.xml");
            // udskriv DOM træ
            writeXML(System.out, "", doc.getDocumentElement());
        } catch (FactoryConfigurationError e) {
            e.printStackTrace();
        } catch (ParserConfigurationException e) {
            e.printStackTrace();
        } catch (SAXException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return;
    }
}
```

Den nok mest brugte XML Java parser nemlig Xerces kommer med en en klasse XMLSerializer til at udskrive DOM træer med. Vigtigt: brug en OutputFormat'er med indenting true - ellers ser output ikke godt ud.

WriteW3CDOMXerces.java

```
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.xml.sax.SAXException;

import org.apache.xml.serialize.OutputFormat;
import org.apache.xml.serialize.XMLSerializer;

public class WriteW3CDOMXerces {
    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("C:\\test.xml");
            // udskriv DOM træ
            OutputFormat fmt = new OutputFormat();
            fmt.setIndenting(true);
            XMLSerializer ser = new XMLSerializer(System.out, fmt);
            ser.serialize(doc);
        } catch (FactoryConfigurationError e) {
            e.printStackTrace();
        } catch (ParserConfigurationException e) {
            e.printStackTrace();
        } catch (SAXException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return;
    }
}
```

Efter mange år uden en standard måde at udskrive DOM træer med kom der en metode.
Bemærk: denne metode kræver en nyere JAXP (enten nyere Java eller en nyere Xerces), fordi den kræver DOM 3.0 !

WriteW3CDOMStandard.java

```
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.DOMImplementation;
import org.w3c.dom.bootstrap.DOMImplementationRegistry;
import org.w3c.dom.ls.DOMImplementationLS;
import org.w3c.dom.ls.LSOutput;
import org.w3c.dom.ls.LSSerializer;
import org.xml.sax.SAXException;

public class WriteW3CDOMStandard {
    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("C:\\test.xml");
            // udskriv DOM træ
            DOMImplementation impl =
DOMImplementationRegistry.newInstance().getDOMImplementation("XML 3.0");
            DOMImplementationLS feature =
(DOMImplementationLS)impl.getFeature("LS","3.0");
            LSSerializer ser = feature.createLSSerializer();
            LSOutput output = feature.createLSOutput();
            output.setCharacterStream(System.console().writer());
            ser.write(doc, output);
        } catch (FactoryConfigurationError e) {
            e.printStackTrace();
        } catch (ParserConfigurationException e) {
            e.printStackTrace();
        } catch (SAXException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (InstantiationException e) {
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        }
        return;
    }
}
```

Mens man ventede på DOM 3.0 fandt snedige folk ud af at man kunne bruge XSLT med et tomt XSL til at udskrive med.

WriteW3CDOMXSLT.java

```
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 javax.xml.transform.Transformer;
import javax.xml.transform.TransformerConfigurationException;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.Document;
import org.xml.sax.SAXException;

public class WriteW3CDOMXSLT {
    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("C:\\test.xml");
            // udskriv DOM træ
            TransformerFactory tf = TransformerFactory.newInstance();
            Transformer t = tf.newTransformer();
            t.transform(new DOMSource(doc), new StreamResult(System.out));
        } catch (FactoryConfigurationError e) {
            e.printStackTrace();
        } catch (ParserConfigurationException e) {
            e.printStackTrace();
        } catch (SAXException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (TransformerConfigurationException e) {
            e.printStackTrace();
        } catch (TransformerException e) {
            e.printStackTrace();
        }
        return;
    }
}
```

Udskrift JDOM træ

JDOM har en indbygget måde at udskrive et træ på som iøvrigt ligner Xerces en del.

WriteJDOM.java

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

```
import org.jdom.Document;
import org.jdom.JDOMException;
import org.jdom.adapters.DOMAdapter;
import org.jdom.adapters.XercesDOMAdapter;
import org.jdom.input.DOMBuilder;
import org.jdom.output.Format;
import org.jdom.output.XMLOutputter;

public class WriteJDOM {
    public static void main(String[] args) {
        try {
            // læs fra fil til DOM træ
            DOMAdapter da = new XercesDOMAdapter();
            org.w3c.dom.Document w3cdoc = da.getDocument(new
FileInputStream("C:\\test.xml"), false);
            DOMBuilder b = new DOMBuilder();
            Document doc = b.build(w3cdoc);
            // udskriv DOM træ
            XMLOutputter fmt = new XMLOutputter(Format.getPrettyFormat());
            System.out.println(fmt.outputString(doc));
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (JDOMException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Ændring af og helt nyt W3C DOM træ

En anden naturlig ting er at ville er at ændre et indlæst træ.

Det er ret simpelt. Dokumentet har metoder til at oprette nye noder og alle noder har en metode til at tilføje de nyoprettede noder som børn.

ChangeW3CDOM.java

```
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;

import org.apache.xml.serialize.OutputFormat;
import org.apache.xml.serialize.XMLSerializer;

public class ChangeW3CDOM {
    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("C:\\test.xml");
            // fjern tomme medlemmer
            NodeList elements = doc.getElementsByTagName("medlem");
            for (int i = 0; i < elements.getLength(); i++) {
                Node element = (Element) elements.item(i);
                if(!element.hasChildNodes()) {
                    element.getParentNode().removeChild(element);
                }
            }
            // tilføj nyt medlem
            Element navn = doc.createElement("navn");
            navn.appendChild(doc.createTextNode("Lars Larsen"));
            Element adresse = doc.createElement("adresse");
            adresse.appendChild(doc.createTextNode("Ledvej 14"));
            Element medlem = doc.createElement("medlem");
            medlem.setAttribute("no", "4");
            medlem.appendChild(navn);
            medlem.appendChild(adresse);
            doc.getDocumentElement().appendChild(medlem);
            // udskriv DOM træ
            OutputFormat fmt = new OutputFormat();
            fmt.setIndenting(true);
            XMLSerializer ser = new XMLSerializer(System.out, fmt);
            ser.serialize(doc);
        } catch (FactoryConfigurationError e) {
            e.printStackTrace();
        } catch (ParserConfigurationException e) {
            e.printStackTrace();
        } catch (SAXException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return;
    }
}
```

Og at oprette et helt nyt træ er helt tilsvarende.

CreateW3CDOM.java

```
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.apache.xml.serialize.OutputFormat;
import org.apache.xml.serialize.XMLSerializer;

public class CreateW3CDOM {
    public static void main(String[] args) {
        try {
            // lav nyt DOM træ
            DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
            DocumentBuilder db = dbf.newDocumentBuilder();
            Document doc = db.newDocument();
            // tilføj elementer til DOM træ
            Element one1 = doc.createElement("one");
            one1.appendChild(doc.createTextNode("A"));
            Element one2 = doc.createElement("one");
            one2.appendChild(doc.createTextNode("BB"));
            Element one3 = doc.createElement("one");
            one3.appendChild(doc.createTextNode("CCC"));
            Element all = doc.createElement("all");
            all.appendChild(one1);
            all.appendChild(one2);
            all.appendChild(one3);
            doc.appendChild(all);
            // udskriv DOM træ
            OutputFormat fmt = new OutputFormat();
            fmt.setIndenting(true);
            XMLSerializer ser = new XMLSerializer(System.out, fmt);
            ser.serialize(doc);
        } catch (FactoryConfigurationError e) {
            e.printStackTrace();
        } catch (ParserConfigurationException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return;
    }
}
```

Ændring af og helt nyt JDOM træ

Med hensyn til at ændre og oprette træer ligner JDOM faktisk meget W3C DOM. Man kan bare oprette node med en almindelig constructor og interfacet er en anelse mere naturligt.

ChangeJDOM.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;
import org.jdom.output.Format;
import org.jdom.output.XMLOutputter;

public class ChangeJDOM {
    public static void main(String[] args) {
        try {
            // læs fra fil til DOM træ
            DOMAdapter da = new XercesDOMAdapter();
            org.w3c.dom.Document w3cdoc = da.getDocument(new
FileInputStream("C:\\test.xml"), false);
            DOMBuilder b = new DOMBuilder();
            Document doc = b.build(w3cdoc);
            // fjern tomme medlemmer
            List list = doc.getRootElement().getChildren();
            for (int i = 0; i < list.size(); i++) {
              Element elm = (Element) list.get(i);
              if(elm.getChildren().size() == 0) {
                    elm.getParent().removeContent(elm);
              }
            }
            // tilføj nyt medlem
            Element navn = new Element("navn");
            navn.setText("Lars Larsen");
            Element adresse = new Element("adresse");
            adresse.setText("Ledvej 14");
            Element medlem = new Element("medlem");
            medlem.addContent(navn);
            medlem.addContent(adresse);
            medlem.setAttribute("no", "4");
            doc.getRootElement().addContent(medlem);
            // udskriv DOM træ
            XMLOutputter fmt = new XMLOutputter(Format.getPrettyFormat());
            System.out.println(fmt.outputString(doc));
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
```

```
            e.printStackTrace();
        } catch (JDOMException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

CreateJDOM.java

```
import org.jdom.Document;
import org.jdom.Element;
import org.jdom.output.Format;
import org.jdom.output.XMLOutputter;

public class CreateJDOM {
    public static void main(String[] args) {
        // lav nyt DOM træ
        Document doc = new Document();
        // tilføj elementer til DOM træ
        Element one1 = new Element("one");
        one1.setText("A");
        Element one2 = new Element("one");
        one2.setText("BB");
        Element one3 = new Element("one");
        one3.setText("CCC");
        Element all = new Element("all");
        all.addContent(one1);
        all.addContent(one2);
        all.addContent(one3);
        doc.setRootElement(all);
        // udskriv DOM træ
        XMLOutputter fmt = new XMLOutputter(Format.getPrettyFormat());
        System.out.println(fmt.outputString(doc));
    }
}
```

Walker

W3C DOM har en smart måde at søge et træ igennem efter en bestemt slags noder.

Bemærk at denne feature er meget ny og ikke er understøttet i Java SDK 1.4 XML support. Man er nødt til at bruge 1.5 eller at hente f.eks. en nyere Xerces.

ScanWitHWalker.java

```
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.Node;
import org.w3c.dom.traversal.DocumentTraversal;
import org.w3c.dom.traversal.NodeFilter;
import org.w3c.dom.traversal.TreeWalker;
import org.xml.sax.SAXException;

public class ScanWithWalker {
    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("C:\\test.xml");
            // walk alle elementer af typen navn
            TreeWalker walk = ((DocumentTraversal)
doc).createTreeWalker(doc.getDocumentElement(),

NodeFilter.SHOW_ELEMENT, null, false);
            Node n;
            while ((n = walk.nextNode()) != null) {
                if (n.getNodeName().equals("navn")) {
                    System.out.println(n.getFirstChild().getNodeValue());
                }
            }
        } catch (FactoryConfigurationError e) {
            e.printStackTrace();
        } catch (ParserConfigurationException e) {
            e.printStackTrace();
        } catch (SAXException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Vi laver en walker som:
- starter i document root
- finder alle elementer
- ikke har et filter

Og så tester vi bare om elementet er et name. Den funktionalitet kunne også være lagt ind i et filter. Men det synes jeg ikke var umagen værd her.

Man kan selvfølgelig nemt selv lave noget kode som løber alle elementer igennem (f.eks. rekursivt lige som i writeXML metoden i første program). Men walkeren giver noget pænt og let læseligt kode.

XPath

Hvis man kun skal bruge nogle ganske bestemte noder kan man selecte dem med XPath som er et query sprog til XML.

Jeg vil ikke gå i detaljer med hensyn til XPath syntax det kan man og det er der skrevet bøger om.

Den ultra korte version er:
xxxx - finder elementer med navn xxxx
//xxxx/yyyy - finder elementer med navn yyyy under elementer med navn xxxx
xxxx[yyyy='abc'] - finder elementer med navn xxxx som har et under element med navn yyyy og en tekstværdi 'abc'
xxxx[@yyyy=123] - finder elementer med navn xxxx som har en attribut med navn yyyy og en talværdi 123

Man kan bruge XPath i både W3C DOM og JDOM.

XPath er ikke understøttet i Java SDK 1.4 XML support. Man er nødt til at bruge 1.5 eller at hente f.eks. en nyere Xerces.

For W3C DOM findes der både en Xerces specifik og en standard måde at gøre det på.

SelectWithXPathW3CDOMXerces.java:

```
import java.io.IOException;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.TransformerException;

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

import org.apache.xpath.XPathAPI;

public class SelectWithXPathW3CDOM {
    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("C:\\test.xml");
            // find element hvor no=2
            Element res1 =
```

```
(Element)XPathAPI.selectSingleNode(doc.getDocumentElement(),
"/medlemmer/medlem[@no=2]");
            Element name1 =
(Element)res1.getElementsByTagName("navn").item(0);
            System.out.println(name1.getFirstChild().getNodeValue());
            // find alle elementer hvor no >= 2
            NodeList res2 = XPathAPI.selectNodeList(doc.getDocumentElement(),
"/medlemmer/medlem[@no>=2]");
            for(int i = 0; i < res2.getLength(); i++) {
                Element name2 =
(Element)((Element)res2.item(i)).getElementsByTagName("navn").item(0);
                if(name2 != null) {
                    System.out.println(name2.getFirstChild().getNodeValue());
                }
            }
        } catch (ParserConfigurationException e) {
            e.printStackTrace();
        } catch (SAXException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (TransformerException e) {
            e.printStackTrace();
        }
    }
}
```

SelectWithXPathW3CDOMStandard.java:

```
import java.io.IOException;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.TransformerException;
import javax.xml.xpath.XPath;
import javax.xml.xpath.XPathConstants;
import javax.xml.xpath.XPathExpressionException;
import javax.xml.xpath.XPathFactory;

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

import org.apache.xpath.XPathAPI;

public class SelectWithXPathW3CDOMStandard {
    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("C:\\test.xml");
            XPath xpath = XPathFactory.newInstance().newXPath();
            // find element hvor no=2
            Element res1 = (Element)xpath.evaluate("/medlemmer/medlem[@no=2]",
doc.getDocumentElement(), XPathConstants.NODE);
            Element name1 =
(Element)res1.getElementsByTagName("navn").item(0);
            System.out.println(name1.getFirstChild().getNodeValue());
            // find alle elementer hvor no >= 2
            NodeList res2 =
(NodeList)xpath.evaluate("/medlemmer/medlem[@no>=2]",
doc.getDocumentElement(), XPathConstants.NODESET);
            for(int i = 0; i < res2.getLength(); i++) {
                Element name2 =
(Element)((Element)res2.item(i)).getElementsByTagName("navn").item(0);
                if(name2 != null) {
                    System.out.println(name2.getFirstChild().getNodeValue());
                }
            }
        } catch (ParserConfigurationException e) {
            e.printStackTrace();
        } catch (SAXException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (XPathExpressionException e) {
            e.printStackTrace();
        }
    }
}
```

SelectWithXPathJDOM.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;
import org.jdom.xpath.XPath;

public class SelectWithXPathJDOM {
    public static void main(String[] args) {
        try {
```

```
            // læs fra fil til DOM træ
            DOMAdapter da = new XercesDOMAdapter();
            org.w3c.dom.Document w3cdoc = da.getDocument(new
FileInputStream("C:\\test.xml"), false);
            DOMBuilder b = new DOMBuilder();
            Document doc = b.build(w3cdoc);
            // find element hvor no=2
            Element res1 = (Element)XPath.selectSingleNode(doc,
"/medlemmer/medlem[@no=2]");
            System.out.println(res1.getChild("navn").getText());
            // find alle elementer hvor no >= 2
            List res2 = XPath.selectNodes(doc, "/medlemmer/medlem[@no>=2]");
            for (int i = 0; i < res2.size(); i++) {
                Element name2 = ((Element)res2.get(i)).getChild("navn");
                if(name2 != null) {
                    System.out.println(name2.getText());
                }
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (JDOMException e) {
            e.printStackTrace();
        }
    }
}
```

Videre

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

Kommentar af simonvalter d. 03. May 2004 | 1

en masse gode eksempler som man kan tage udgangspunkt i og bygge videre på... sådan lærer jeg bedst.

Kommentar af thundergod d. 07. May 2004 | 2

Slet ikke nok forklarende tekst.