Nye Java XML API'er

Denne artikel beskriver nye XML features i Java 1.6 - StAX og JAXB, samt XStreams.

Den forudsætter kendskab til Java og brug af XML i Java svarende til tidligere artikler.

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

Historie:
V1.0 - 26/12/2008 - original
V1.1 - 14/02/2010 - smårettelser

Tidligere artikler

Hvis man ikke har læset dem, så kan det anbefales at starte med at læse dem:
http://www.eksperten.dk/artikler/100 "XML parsning i Java"
http://www.eksperten.dk/artikler/245 "Mere XML i Java"

StAX

StAX er ny i Java 1.6.

StAX tilbyder:
- at læse XML som stream (en event driven pull parser)
- at skrive XML som stream

StAX er meget tilsvarende .NET XmlTextReader og XmlTextWriter klasserne.

Lad os se et læse eksempel og et skrive eksempel.

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>
```

ParseStAX.java

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

import javax.xml.stream.XMLInputFactory;
import javax.xml.stream.XMLStreamException;
import javax.xml.stream.XMLStreamReader;

public class ParseStAX {
    public static void main(String[] args) {
        try {
            XMLInputFactory xif = XMLInputFactory.newInstance();
            XMLStreamReader xsr = xif.createXMLStreamReader(new
FileInputStream("C:\\test.xml"));
            StringBuilder sb = new StringBuilder("");
            while(xsr.hasNext()) {
                xsr.next();
                switch(xsr.getEventType()) {
                    case XMLStreamReader.CHARACTERS:
                        sb.append(xsr.getText());
                        break;
                    case XMLStreamReader.START_ELEMENT:
                        if (xsr.getLocalName().equals("medlem")) {
                            String no = xsr.getAttributeValue(null, "no");
                            System.out.println("no=" + no);
                        }
                        break;
                    case XMLStreamReader.END_ELEMENT:
                        if (xsr.getLocalName().equals("navn")) {
                            String name = sb.toString().trim();
                            System.out.println("navn=" + name);
                        }
                        if (xsr.getLocalName().equals("adresse")) {
                            String address = sb.toString().trim();
                            System.out.println("adresse=" + address);
                        }
                        sb = new StringBuilder();
                     break;
                }
            }
            xsr.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (XMLStreamException e) {
            e.printStackTrace();
        }
    }
}
```

CreateStAX.java

```
import javax.xml.stream.XMLOutputFactory;
import javax.xml.stream.XMLStreamException;
import javax.xml.stream.XMLStreamWriter;

public class CreateStAX {
    public static void main(String[] args) {
        try {
            XMLOutputFactory xof = XMLOutputFactory.newInstance();
            XMLStreamWriter xsw = xof.createXMLStreamWriter(System.out);
            xsw.writeStartDocument();
            xsw.writeStartElement("all");
            xsw.writeStartElement("one");
            xsw.writeCharacters("A");
            xsw.writeEndElement();
            xsw.writeStartElement("one");
            xsw.writeCharacters("BB");
            xsw.writeEndElement();
            xsw.writeStartElement("one");
            xsw.writeCharacters("CCC");
            xsw.writeEndElement();
            xsw.writeEndElement();
            xsw.writeEndDocument();
            xsw.close();
        } catch (XMLStreamException e) {
            e.printStackTrace();
        }
    }
}
```

Bemærk at den standard StAX som kommer med Java 1.6 ikke har mulighed for at bede om formatering/indentering med xof.setProperty, men det har andre implementeringer.

Vurdering:
- StAX løser samme opgave som SAX og er nemmere at bruge end SAX, så jeg vil anbefale StAX fremfor SAX som XML parser til store XML dokumenter
- StAX er nemt at bruge til at skrive XML dokumenter med, så det kan også anbefales

JAXB

JAXB er også ny i Java 1.6 (omend det har eksisteret i længere tid som del af diverse Java EE web service pakker).

JAXB kan konvertere mellem Java objekter og XML filer.

Inden vi kigger på JAXB så skal vi lige kigge på en feature som har eksisteret i Java side version 1.4 nemlig XMLEncoder og XMLDecoder. De kan nemlig også lave en sådan konvertering.

SaveXMLEnc.java

```
import java.beans.XMLEncoder;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;

public class SaveXMLEnc {
    public static void main(String[] args) {
        try {
            Data o = new Data(123, 123.456, "ABC");
            XMLEncoder xe = new XMLEncoder(new
FileOutputStream("C:\\xmlencdec.xml"));
            xe.writeObject(o);
            xe.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

LoadXMLDec.java

```
import java.beans.XMLDecoder;
import java.io.FileInputStream;
import java.io.FileNotFoundException;

public class LoadXMLDec {
    public static void main(String[] args) {
        try {
            XMLDecoder xd = new XMLDecoder(new
FileInputStream("C:\\xmlencdec.xml"));
            Data o = (Data)xd.readObject();
            xd.close();
            System.out.println(o.getIv() + " " + o.getXv() + " " + o.getSv());
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

xmlencdec.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.6.0" class="java.beans.XMLDecoder">
<object class="Data">
  <void property="iv">
```

```
   <int>123</int>
  </void>
  <void property="sv">
   <string>ABC</string>
  </void>
  <void property="xv">
   <double>123.456</double>
  </void>
</object>
</java>
```

Nu laver vi det samme med JAXB.

Data.java

```
import java.io.Serializable;

import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement
public class Data implements Serializable {
    public Data() {
        this(0, 0.0, "");
    }
    public Data(int iv, double xv, String sv) {
        this.iv = iv;
        this.xv = xv;
        this.sv = sv;
    }
    private int iv;
    private double xv;
    private String sv;
    public int getIv() {
        return iv;
    }
    public void setIv(int iv) {
        this.iv = iv;
    }
    public double getXv() {
        return xv;
    }
    public void setXv(double xv) {
        this.xv = xv;
    }
    public String getSv() {
        return sv;
    }
    public void setSv(String sv) {
        this.sv = sv;
    }
}
```

SaveJAXB.java

```
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;


public class SaveJAXB {
    public static void main(String[] args) {
        try {
            Data o = new Data(123, 123.456, "ABC");
            JAXBContext jxbctx = JAXBContext.newInstance(Data.class);
            OutputStream os = new FileOutputStream("C:\\jaxb.xml");
            Marshaller m = jxbctx.createMarshaller();
            m.marshal(o, os);
            os.close();
        } catch (JAXBException e) {
            e.printStackTrace();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

LoadJAXB.java

```
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;


public class SaveJAXB {
    public static void main(String[] args) {
        try {
            Data o = new Data(123, 123.456, "ABC");
```

```
            JAXBContext jxbctx = JAXBContext.newInstance(Data.class);
            OutputStream os = new FileOutputStream("C:\\jaxb.xml");
            Marshaller m = jxbctx.createMarshaller();
            m.marshal(o, os);
            os.close();
        } catch (JAXBException e) {
            e.printStackTrace();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

jaxb.xml

```
<?xml version="1.0" encoding="UTF-8"
standalone="yes"?><data><iv>123</iv><sv>ABC</sv><xv>123.456</xv></data>
```

Hvilket er betydeligt pænere XML end det der er genereret af XMLEncoder.

JAXB kan imidlertid også bruges omvendt, hvor man starter med et XML schema
og genererer Java klassen udfra det.

Data.xsd

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" version="1.0">
    <xsd:element name="data">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="iv" type="xsd:integer"/>
                <xsd:element name="xv" type="xsd:decimal"/>
                <xsd:element name="sv" type="xsd:string"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Kommando for at generere Data.obj:
xjc -d . -p "" Data.xsd

(i praksis vil man naturligvis altid angive et package navn til -p og ikke bruge
default package)

SaveJAXBrev.java

```
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.math.BigInteger;
import java.math.BigDecimal;

import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;

public class SaveJAXBrev {
    public static void main(String[] args) {
        try {
            Data o = new Data();
            o.setIv(new BigInteger("123"));
            o.setXv(new BigDecimal("123.456"));
            o.setSv("ABC");
            JAXBContext jxbctx = JAXBContext.newInstance(Data.class);
            OutputStream os = new FileOutputStream("C:\\jaxbrev.xml");
            Marshaller m = jxbctx.createMarshaller();
            m.marshal(o, os);
            os.close();
        } catch (JAXBException e) {
            e.printStackTrace();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

LoadJAXBrev.java

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

import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Unmarshaller;

public class LoadJAXBrev {
    public static void main(String[] args) {
        try {
            JAXBContext jxbctx = JAXBContext.newInstance(Data.class);
```

```
            InputStream is = new FileInputStream("C:\\jaxbrev.xml");
            Unmarshaller um = jxbctx.createUnmarshaller();
            Data o = (Data)um.unmarshal(is);
            is.close();
            System.out.println(o.getIv() + " " + o.getXv() + " " + o.getSv());
        } catch (JAXBException e) {
            e.printStackTrace();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

jaxbrev.xml

```
<?xml version="1.0" encoding="UTF-8"
standalone="yes"?><data><iv>123</iv><xv>123.456</xv><sv>ABC</sv></data
```

XStream

XStream er et XML serialiserings framework som ikke er en del af Java men som kan hentes hos [http://xstream.codehaus.org/.](http://xstream.codehaus.org/)

Eksempler med læsning og skrivning:

Data.java

```
import java.io.Serializable;

public class Data implements Serializable {
    public Data() {
        this(0, 0.0, "");
    }
    public Data(int iv, double xv, String sv) {
        this.iv = iv;
        this.xv = xv;
        this.sv = sv;
    }
    private int iv;
    private double xv;
    private String sv;
    public int getIv() {
        return iv;
    }
    public void setIv(int iv) {
        this.iv = iv;
```

```
    }
    public double getXv() {
        return xv;
    }
    public void setXv(double xv) {
        this.xv = xv;
    }
    public String getSv() {
        return sv;
    }
    public void setSv(String sv) {
        this.sv = sv;
    }
}
```

SaveXStream.java

```
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

import com.thoughtworks.xstream.XStream;
import com.thoughtworks.xstream.io.xml.DomDriver;

public class SaveXStream {
    public static void main(String[] args) {
        try {
            Data o = new Data(123, 123.456, "ABC");
            XStream xs = new XStream(new DomDriver());
            xs.alias("data", Data.class);
            OutputStream os = new FileOutputStream("C:\\xstream.xml");
            xs.toXML(o, os);
            os.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

LoadXStream.java

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

```
import com.thoughtworks.xstream.XStream;
import com.thoughtworks.xstream.io.xml.DomDriver;

public class LoadXStream {
    public static void main(String[] args) {
        try {
            XStream xs = new XStream(new DomDriver());
            xs.alias("data", Data.class);
            InputStream is = new FileInputStream("C:\\xstream.xml");
            Data o = (Data)xs.fromXML(is);
            is.close();
            System.out.println(o.getIv() + " " + o.getXv() + " " + o.getSv());
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

xstream.xml

```
<data>
  <iv>123</iv>
  <xv>123.456</xv>
  <sv>ABC</sv>
</data>
```

XStream er ret populært fordi det er nemt at bruge og fordi det genererer simpel XML.

Som et kuriosum kan det nævnes at XStreams også kan generere JSON !