Modélisation et interopérabilité : Semaine 48, cours 10

Benoît Valiron <benoit.valiron@monoidal.net>

http://inf356.monoidal.net/

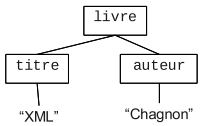
Représentation d'un document XML

Deux façons canoniques

- Traitement sous forme d'arbre :
  - Document Object Model
  - Problème : tout le document est chargé en mémoire
  - Avantage : accès aléatoire simple
- Traitement événementiel :
  - SAX : modèle Push
  - trSAX : modèle Pull
  - Problème : Accès aléatoire fastidieux
  - Avantage : Le document peut être (très) gros

Document Object Model


```
<livre lang="fr"><titre>XML</titre><auteur>Chagnon</auteur></livre>
```



DOM

- Interface abstraite : ensemble d'objets et de méthodes qui doivent être implémentée.
- Implémentation en java, javascript, C++, perl, php...
  - En java : la librairie Xerces
  - En javascript : l'implémentation de Mozilla
- Les interfaces génériques :
  - Node
  - NodeList
  - NamedNodeMap

Interface Node

- Quelques champs :

```
NamedNodeMap attributes;
NodeList childNodes;
Node firstChild;
Node lastChild;
DOMString localName;
DOMString namespaceURI;
Node nextSibling;
DOMString nodeName;
unsigned short nodeType;
DOMString nodeValue;
Node parentNode;
DOMString prefix;
Node previousSibling;
DOMString textContent;
```

En java : xxx → getXxx();

- Quelques méthodes :

```
Node appendChild(Node child);
Node cloneNode(boolean deep);
boolean hasAttributes();
boolean hasChildNodes();
Node insertBefore(Node newChild,
                  Node refChild);
Node removeChild(Node oldChild);
Node replaceChild(Node newChild,
                  Node oldChild);
```


Types de noeuds usuels

Valeur de nodeType	Interface DOM
`ATTRIBUTE_NODE`	`Attr` (nodeName : l'attribut)
`COMMENT_NODE`	`Comment` (nodeName : `#comment`)
`DOCUMENT_NODE`	`Document` (nodeName : `#document`)
`ELEMENT_NODE`	`Element` (nodeName : la balise)
`TEXT_NODE`	`Text` (nodeName : `#text`)

Interfaces spécifiques au type Node

- Ils sont sous-types de Node. On a Attr, Comment,
- Document :

```
Attr createAttribute(DOMString name);
Element createElement(DOMString tagName);
Text createTextNode(DOMString data);
Element getElementsById(DOMString id);
NodeList getElementsByTagName(DOMString tagName);
```

- Element :

```
DOMString getAttribute(DOMString name);
void setAttribute(DOMString name, DOMString value);
Element getElementsById(DOMString id);
NodeList getElementsByTagName(DOMString tagName);
```

- Text

```
boolean isContentWhiteSpace;
```


Autres interfaces génériques

- NamedNodeMap : table d'association DOMString ↔ Node

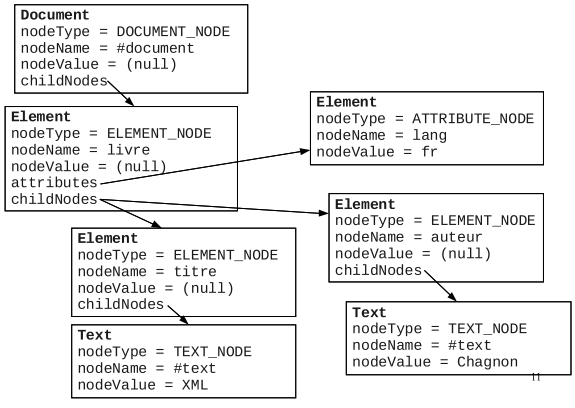
```
long length;
Node getNamedItem(DOMString name);
Node removeNamedItem(DOMString name);
Node setNamedItem(Node n);
```

- NodeList :

```
long length;
Node item(long i);
```



```
<livre lang="fr"><titre>XML</titre><auteur>Chagnon</auteur></livre>
```



Exemple

```
<?xml version="1.0" encoding="utf-8" ?>
<liste>
  <livre xml:lang="fr">
    <titre>XML</titre>
    <auteur>Harold</auteur>
    <auteur>Means</auteur>
  </livre>
  <livre>
    <titre>Quantum Computation</titre>
    <auteur>Nielsen</auteur>
  </livre>
</liste>
```

```
public class Simple {

    public static void main(String[] args) {
        DOMParser parser = new DOMParser();
        parser.parse("test.xml");
        Document doc = parser.getDocument();

        COMMANDE

    }

}
```

Exemple

```
<?xml version="1.0" encoding="utf-8" ?>
<liste>
  <livre xml:lang="fr">
    <titre>XML</titre>
    <auteur>Harold</auteur>
    <auteur>Means</auteur>
  </livre>
  <livre>
    <titre>Quantum Computation</titre>
    <auteur>Nielsen</auteur>
  </livre>
</liste>
```

```
public class Simple {

    public static void main(String[] args) {
        DOMParser parser = new DOMParser();
        parser.parse("test.xml");
        Document doc = parser.getDocument();

        doc.getNodeName();

    }

}
```

#document

Exemple

```
<?xml version="1.0" encoding="utf-8" ?>
<liste>
  <livre xml:lang="fr">
    <titre>XML</titre>
    <auteur>Harold</auteur>
    <auteur>Means</auteur>
  </livre>
  <livre>
    <titre>Quantum Computation</titre>
    <auteur>Nielsen</auteur>
  </livre>
</liste>
```

```
public class Simple {

    public static void main(String[] args) {
        DOMParser parser = new DOMParser();
        parser.parse("test.xml");
        Document doc = parser.getDocument();

        doc.getChildNodes().item(0);

    }

}
```

L'élément `liste`

Exemple

```
<?xml version="1.0" encoding="utf-8" ?>
<liste>
  <livre xml:lang="fr">
    <titre>XML</titre>
    <auteur>Harold</auteur>
    <auteur>Means</auteur>
  </livre>
  <livre>
    <titre>Quantum Computation</titre>
    <auteur>Nielsen</auteur>
  </livre>
</liste>
```

```
public class Simple {

    public static void main(String[] args) {
        DOMParser parser = new DOMParser();
        parser.parse("test.xml");
        Document doc = parser.getDocument();

        doc.getChildNodes().item(0).getNodeName();

    }

}
```

`liste`

Exemple

```
<?xml version="1.0" encoding="utf-8" ?>
<liste>
  <livre xml:lang="fr">
    <titre>XML</titre>
    <auteur>Harold</auteur>
    <auteur>Means</auteur>
  </livre>
  <livre>
    <titre>Quantum Computation</titre>
    <auteur>Nielsen</auteur>
  </livre>
</liste>
```

```
public class Simple {

    public static void main(String[] args) {
        DOMParser parser = new DOMParser();
        parser.parse("test.xml");
        Document doc = parser.getDocument();

        doc.getChildNodes().item(0).getChildNodes().item(1);

    }

}
```

`livre`

Exemple

```
<?xml version="1.0" encoding="utf-8" ?>
<liste>
  <livre xml:lang="fr">
    <titre>XML</titre>
    <auteur>Harold</auteur>
    <auteur>Means</auteur>
  </livre>
  <livre>
    <titre>Quantum Computation</titre>
    <auteur>Nielsen</auteur>
  </livre>
</liste>
```

```
public class Simple {

    public static void main(String[] args) {
        DOMParser parser = new DOMParser();
        parser.parse("test.xml");
        Document doc = parser.getDocument();

        doc.getChildNodes().item(0).getChildNodes().item(0);

    }

}
```

Du texte : un saut de ligne !

Exemple

```
<?xml version="1.0" encoding="utf-8" ?>
<liste>
  <livre xml:lang="fr">
    <titre>XML</titre>
    <auteur>Harold</auteur>
    <auteur>Means</auteur>
  </livre>
  <livre>
    <titre>Quantum Computation</titre>
    <auteur>Nielsen</auteur>
  </livre>
</liste>
```

```
public class Simple {

    public static void main(String[] args) {
        DOMParser parser = new DOMParser();
        parser.parse("test.xml");
        Document doc = parser.getDocument();

        doc.getChildNodes().item(0).getChildNodes().item(0).getNodeName();

    }

}
```

`#text`

Exemple

```
<?xml version="1.0" encoding="utf-8" ?>
<liste>
  <livre xml:lang="fr">
    <titre>XML</titre>
    <auteur>Harold</auteur>
    <auteur>Means</auteur>
  </livre>
  <livre>
    <titre>Quantum Computation</titre>
    <auteur>Nielsen</auteur>
  </livre>
</liste>
```

```
public class Simple {

    public static void main(String[] args) {
        DOMParser parser = new DOMParser();
        parser.parse("test.xml");
        Document doc = parser.getDocument();

        doc.getChildNodes().item(0).getChildNodes().
            item(1).getChildNodes().item(1).getChildNodes().
            item(0).getNodeName());

    }
}
```

```
#text
```


Exemple

```
<?xml version="1.0" encoding="utf-8" ?>
<liste>
  <livre xml:lang="fr">
    <titre>XML</titre>
    <auteur>Harold</auteur>
    <auteur>Means</auteur>
  </livre>
  <livre>
    <titre>Quantum Computation</titre>
    <auteur>Nielsen</auteur>
  </livre>
</liste>
```

```
public class Simple {

    public static void main(String[] args) {
        DOMParser parser = new DOMParser();
        parser.parse("test.xml");
        Document doc = parser.getDocument();

        doc.getChildNodes().item(0).getChildNodes().
            item(1).getChildNodes().item(1).getChildNodes().
            item(0).getNodeValue());

    }
}
```

```
XML
```


Exemple

```
<?xml version="1.0" encoding="utf-8" ?>
<liste>
  <livre xml:lang="fr">
    <titre>XML</titre>
    <auteur>Harold</auteur>
    <auteur>Means</auteur>
  </livre>
  <livre>
    <titre>Quantum Computation</titre>
    <auteur>Nielsen</auteur>
  </livre>
</liste>
```

```
public class Simple {

    public static void main(String[] args) {
        DOMParser parser = new DOMParser();
        parser.parse("test.xml");
        Document doc = parser.getDocument();

        doc.getElementsByTagName("titre").item(0).
            getChildNodes().item(0).getNodeValue();

    }
}
```

```
XML
```


Exemple

```
<?xml version="1.0" encoding="utf-8" ?>
<liste>
  <livre xml:lang="fr">
    <titre>XML</titre>
    <auteur>Harold</auteur>
    <auteur>Means</auteur>
  </livre>
  <livre>
    <titre>Quantum Computation</titre>
    <auteur>Nielsen</auteur>
  </livre>
</liste>
```

```
public class Simple {

    public static void main(String[] args) {
        DOMParser parser = new DOMParser();
        parser.parse("test.xml");
        Document doc = parser.getDocument();

        doc.getElementsByTagName("livre").item(0).hasAttributes();

    }
}
```

```
true
```


Exemple

```
<?xml version="1.0" encoding="utf-8" ?>
<liste>
  <livre xml:lang="fr">
    <titre>XML</titre>
    <auteur>Harold</auteur>
    <auteur>Means</auteur>
  </livre>
  <livre>
    <titre>Quantum Computation</titre>
    <auteur>Nielsen</auteur>
  </livre>
</liste>
```

```
public class Simple {

    public static void main(String[] args) {
        DOMParser parser = new DOMParser();
        parser.parse("test.xml");
        Document doc = parser.getDocument();

        doc.getElementsByTagName("livre").item(1).hasAttributes();

    }
}
```

```
false
```


En XSLT...

```
<xsl:stylesheet version='1.0'
                xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
  <xsl:output method="xml" />

  <xsl:template match="/|*">
    <xsl:value-of select="name()" />
    <xsl:text>
</xsl:text>
    <xsl:apply-templates select="*" />
  </xsl:template>

</xsl:stylesheet>
```

En java...

```
import org.w3c.dom.*;
import com.sun.org.apache.xerces.internal.parsers.DOMParser;

public class Test {

    public static void main(String[] args) {
        DOMParser parser = new DOMParser();
        parser.parse("test.xml");
        Document doc = parser.getDocument();
        traverse(doc);
    }

    private static void traverse(Node node) {
        int type = node.getNodeType();
        if( type == Node.ELEMENT_NODE ) {
            System.out.println(node.getNodeName());
        }
        NodeList children = node.getChildNodes();
        if ( children != null ){
            for (int i = 0 ; i < children.getLength() ; i++) {
                traverse(children.item(i));
            }
        }
    }
}
```


Exemple I/O XML

```
<?xml version="1.0" encoding="utf-8" ?>
<liste>
  <livre xml:lang="fr">
    <titre>XML en concentré</titre>
    <auteur>Harold</auteur>
    <auteur>Means</auteur>
  </livre>
  <livre>
    <titre>Quantum Computation</titre>
    <auteur>Nielsen</auteur>
  </livre>
</liste>
```

On veut supprimer les éléments titre

Exemple I/O XML en XSLT

```
<xsl:stylesheet version='1.0'
                xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
  <xsl:output method="xml" />

  <xsl:template match="titre">
  </xsl:template>

  <xsl:template match="*">
    <xsl:element name="{name()}">
      <xsl:apply-templates select="*|@*|text()" />
    </xsl:element>
  </xsl:template>

  <xsl:template match="@*">
    <xsl:attribute name="{name()}">
      <xsl:value-of select="." />
    </xsl:attribute>
  </xsl:template>

</xsl:stylesheet>
```


Exemple I/O XML en java

```
… some imports …
public class Test2 {

    private static void traverse(Node node) {
        NodeList children = node.getChildNodes();
        if (children != null)
            for (int i = 0; i < children.getLength(); i++) {
                if ((children.item(i).getNodeType() == Node.ELEMENT_NODE)
                        && (children.item(i).getNodeName().equals("titre"))) {
                    node.removeChild(children.item(i));
                } else
                    traverse(children.item(i));
            }
    }

    public static void main(String[] args) {
        DOMParser parser = new DOMParser();
        parser.parse("test.xml");
        Document doc = parser.getDocument();
        traverse(doc);

        TransformerFactory transfac = TransformerFactory.newInstance();
        Transformer trans = transfac.newTransformer();
        trans.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION, "yes");
        trans.setOutputProperty(OutputKeys.INDENT, "yes");

        StringWriter sw = new StringWriter();
        trans.transform(new DOMSource(doc), new StreamResult(sw));
    }
}
```


Lecture d'un fichier d'options:

opt.xml

```
<liste>
  <option nom="opt1" valeur="val1" />
  <option nom="opt2" valeur="val2" />
  <option nom="opt3" valeur="val3" />
</liste>
```


Lecture d'un fichier d'options

```
public class Options {
    public Hashtable<String, String> options;

    Options(String filename) {
        options = new Hashtable<String, String>();
        read(filename);
    }

    public void read(String filename) {
        DOMParser parser = new DOMParser();
        parser.parse(filename);
        Document racine = parser.getDocument();
        NodeList opts = racine.getChildNodes().item(0).getChildNodes();

        if ( opts != null ){
            for (int i = 0 ; i < opts.getLength() ; i++) {
                int type = opts.item(i).getNodeType();
                if( type == Node.ELEMENT_NODE ) {
                    String nom =
opts.item(i).getAttributes().getNamedItem("nom").getNodeValue();
                    String val =
opts.item(i).getAttributes().getNamedItem("valeur").getNodeValue();
                    options.put(nom, val);
                }
            }
        }
    }
```

Lecture d'un fichier d'options

```
import java.util.Iterator;
import java.util.Set;
import java.util.Map.Entry;

public class TestOpts {

    public static void main(String[] args) {
        Options opt = new Options("opt.xml");

        Set<Entry<String, String>> noms = opt.options.entrySet();
        Iterator<Entry<String, String>> i = noms.iterator();

        while ( i.hasNext() ) {
            Entry<String, String> pair = i.next();
            System.out.println(
                    "Option '" + pair.getKey() +
                    "' a pour valeur '" + pair.getValue() + "'"
            );
        }
    }
}
```


Lecture d'un fichier d'options (autrement)

```
public class Options {

    public Hashtable<String, String> options;

    Options(String filename) {
        options = new Hashtable<String, String>();
        read(filename);
    }

    public void read(String filename) {
        DOMParser parser = new DOMParser();
        parser.parse(filename);
        Document racine = parser.getDocument();
        NodeList opts = racine.getElementsByTagName("option");

        if ( opts != null ){
            for (int i = 0 ; i < opts.getLength() ; i++) {
                String nom = ((Element) opts.item(i)).getAttribute("nom");
                String val = ((Element) opts.item(i)).getAttribute("valeur");
                options.put(nom, val);
            }
        }
    }
}
```


Fichier de scores

hs.xml

```
<liste>
  <score nom="Ben" valeur="56" />
  <score nom="Bob" valeur="1" />
  <score nom="Bill" valeur="43" />
</liste>
```

On veut rajouter le score de Lea : elle a fait 1000 points.

Fichier de scores

```
public class HighScore {

    public Hashtable<String, String> scores;

    HighScore(String filename) {
        scores = new Hashtable<String, String>();
        read(filename);
    }

    public void write() {
        DocumentBuilderFactory dbfac = DocumentBuilderFactory.newInstance();
        DocumentBuilder docBuilder = dbfac.newDocumentBuilder();
        Document doc = docBuilder.newDocument();

        Element root = doc.createElement("scores");
        doc.appendChild(root);

        Set<Entry<String, String>> noms = scores.entrySet();
        Iterator<Entry<String, String>> i = noms.iterator();

        while ( i.hasNext() ) {
            Entry<String, String> pair = i.next();
            Element nom = doc.createElement("nom");
            nom.setAttribute("nom", pair.getKey());
            nom.setAttribute("valeur", pair.getValue());
            root.appendChild(nom);
        }
…
```


Fichier de scores

…

```
        TransformerFactory transfac = TransformerFactory.newInstance();
        Transformer trans = transfac.newTransformer();
        trans.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION, "yes");
        trans.setOutputProperty(OutputKeys.INDENT, "yes");

        StringWriter sw = new StringWriter();
        trans.transform(new DOMSource(doc), new StreamResult(sw));

        System.out.println(sw.toString());
    }

    public void read(String filename) {
        DOMParser parser = new DOMParser();
        parser.parse(filename);
        Document racine = parser.getDocument();

        NodeList opts = racine.getElementsByTagName("score");

        if ( opts != null ){
            for (int i = 0 ; i < opts.getLength() ; i++) {
                String nom = ((Element) opts.item(i)).getAttribute("nom");
                String val = ((Element) opts.item(i)).getAttribute("valeur");
                scores.put(nom, val);
            }
        }
    }
}
```


Fichier de scores

```
public class TestHS {

    public static void main(String[] args) {
        HighScore s = new HighScore("hs.xml");

        s.scores.put("Lea", "1000");

        s.write();
    }

}
```

Resultat :

```
<scores>
<nom nom="Bil" valeur="43"/>
<nom nom="Bob" valeur="1"/>
<nom nom="Ben" valeur="56"/>
<nom nom="Lea" valeur="1000"/>
</scores>
```