XML avec JAVA

eXtensible Markup Language
langage à balises extensibles

Langage de représentation de documents

1. XML
2. Parsing de documents XML avec SAX
3. L'arborescence DOM

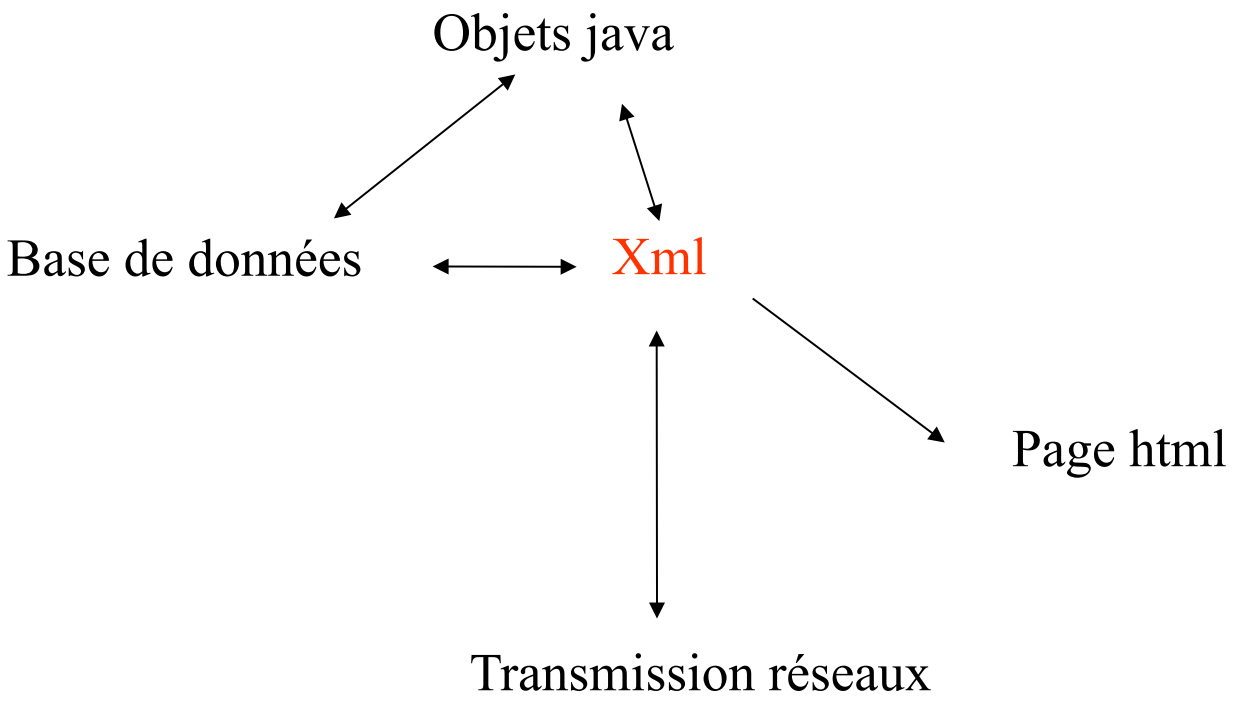
XML Exemple de document

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- le répertoire de Dupond -->
<repertoire creation="2014-12-01">
    <proprietaire>Louis Dupond</proprietaire>
    <entree>
        <nom>Durand</nom>
        <telephone>0323425342</telephone>
    </entree>
    <entree>
        <nom>Marie</nom>
        <telephone>0987654321</telephone>
    </entree>
    <entree>
        <nom>Jean</nom>
        <telephone>0123456789</telephone>
    </entree>
</repertoire>
```

Balises ouvrantes : <repertoire>, <proprietaire> , <entree> …
Balises fermantes : </repertoire>, </proprietaire>, </entree> …

XML : un format d'échange de données



XML Terminologie

Prologue

```
<?xml version="1.0" encoding="UTF-8"?>
```

Element

```
<entree>
    <nom>Durand</nom>
    <telephone>0323425342</telephone>
</entree>
```

Balise ouvrante

```
<entree><repertoire creation="2014-12-01">
```

Balise fermante

```
</entree>
```

Balise ouvrante-fermante

```
<repertoire creation="2014-12-01"   />
```

Attributs

creation

Commentaires

```
<!-- le répertoire de Dupond -->
```

SAX : parseur de flux Xml

SAX
Simple API for XML

Le parseur SAX génère des événements
au fur et à mesure de la lecture du flux XML.

Un objet (ContentHandler) est à l'écoute de ces événements

Exemple d'événements :
- une balise ouvrante
- une balise fermante
- du texte
- début de document
- fin de document
- …

SAX : l'interface org.xml.sax.ContentHandler

Les 11 méthodes de l'interface

- Les plus utilisées

```
// une balise ouvrante
void startElement(String uri, String localName, String rawName, Attributes atts) throws SAXE
// une balise fermante
void endElement(String uri, String localName, String rawName) throws SAXException;
// des caractères
void characters(char[ ] ch, int start, int length) throws SAXException;
// début du document
void startDocument( ) throws SAXException;
// fin du document
void endDocument( ) throws SAXException;
```

- Les autres

```
void setDocumentLocator(Locator locator);
void ignorableWhitespace(char[ ] ch, int start, int length) throws SAXException;
void processingInstruction(String target, String data) throws SAXException;
void skippedEntity(String name) throws SAXException;

void startPrefixMapping(String prefix, String uri) throws SAXException;
void endPrefixMapping(String prefix) throws SAXException;
```

un exemple de handler

```
class RepHandlerDemo extends DefaultHandler {
  public void startDocument( ) {
    System.out.println("Debut");
  }

  public void endDocument( ) {
    System.out.println("Fin");
  }

  public void startElement(String uri, String localName, String rawName, Attributes atts) {
    System.out.println("Debut de balise "+rawName);
  }

  public void endElement(String uri, String localName, String rawName) {
    System.out.println("Fin de balise "+rawName);
  }

  public void characters(char[ ] ch, int start, int length) {
    System.out.println("Des caractères :"+
              String.copyValueOf(ch, start,length).trim( )+":");
  }
}
```

Un exemple d'utilisation du parseur

```
public static void main(String [ ] args) throws Exception {
  // création d'une fabrique de parseurs
  SAXParserFactory spf = SAXParserFactory.newInstance( );

  // création d'un parseur
  SAXParser saxParser = spf.newSAXParser( );

  // création d'un handler
  RepHandlerDemo chd = new RepHandlerDemo( );

  // parsing du fichier rep.xml
  saxParser.parse(new File("rep.xml"),chd);

}
```

Sortie avec RepHandlerDemo

```
Debut
Debut de balise repertoire
Des caractères ::
Debut de balise entree
Des caractères ::
Debut de balise nom
Des caractères :Durand:
Fin de balise nom
Des caractères ::
Debut de balise telephone
Des caractères :0323425342:
Fin de balise telephone
Des caractères ::
Fin de balise entree
Des caractères ::
Fin de balise repertoire
Fin
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- le répertoire de Dupond -->
<repertoire creation="2014-12-01">
  <entree>
    <nom>Durand</nom>
    <telephone>0323425342</telephone>
  </entree>
</repertoire>
```

SAXParser : la méthode parse

```
/** Parse the content of the file specified as XML
    * using the specified DefaultHandler
    * @params   file   the file containing the XML to parse
    * @params handlerthe SAX DefaultHandler to use.
    * @throws IOException If any IO errors occur.
    * @throws IllegalArgumentException - If the File object is null
    * @throws SAXException If the underlying parser
                          throws a SAXException while parsing.
    */

public void parse(File file, DefaultHandler handler)
                          throws SAXException, IOException
```

SAX : la classe DefaultHandler

public class DefaultHandler
implements EntityResolver, DTDHandler, ContentHandler, ErrorHandler

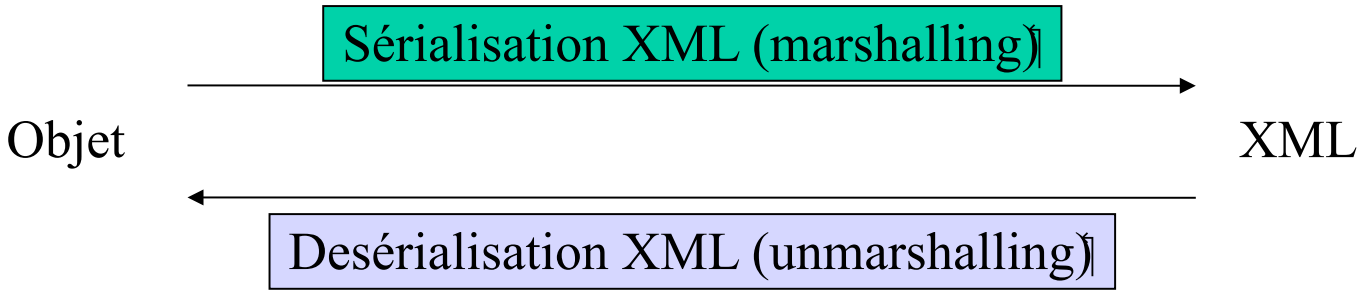
La classe DefaultHandler définit les méthodes de ces 4 interfaces comme ne faisant rien (Exceptée fatalError)

ContentHandler

ErrorHandler pour effectuer un traitement lors d'apparition d'erreurs

```
public void fatalError(SAXParseException e){…}
public void error(SAXParseException e) {…}
public void warning(SAXParseException e) {…}
```

XML ←→ OBJET



Exemple :

obtenir un fichier xml à partir d'un objet répertoire

obtenir l'objet repertoire correspondant à un objet xml

La classe Repertoire

```
public class Repertoire {
private String proprietaire, creationDate;
private Map<String,String> tels = new HashMap<String, String>( );

public Repertoire(String creationDate) {this.creationDate = creationDate;}

public void ajouterEntree(String nom,String tel){this.tels.put(nom,tel);}

public String getTel(String nom){return this.tels.get(nom);}

public void setProprietaire(String name){this.proprietaire=name;}
public String getProprietaire ( ) {return this.proprietaire;}
public String getCreationDate( ) {return this.creationDate;}
```

XML Répertoire

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- le répertoire de Dupond -->
<repertoire creation="2014-12-01">
    <proprietaire>Louis Dupond</proprietaire>
    <entree>
        <nom>Durand</nom>
        <telephone>0323425342</telephone>
    </entree>
    <entree>
        <nom>Marie</nom>
        <telephone>0987654321</telephone>
    </entree>
    <entree>
        <nom>Jean</nom>
        <telephone>0123456789</telephone>
    </entree>
</repertoire>
```

Serialisation

```
// une méthode d'instance dans la classe Repertoire
public void toXml(Writer writer)throws IOException{
  BufferedWriter bw = new BufferedWriter(writer);
  bw.write("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
  bw.write("<repertoire creation=\""+this.getCreationDate( )+"\">\n");
  bw.write("     <proprietaire>"+this.getProprietaire( )+"</proprietaire>\n");
  for (String name:tels.keySet( )){
      bw.write("      <entree>\n");
      bw.write("      <nom>"+name+"</nom>\n");
      bw.write("      <telephone>"+tels.get(name)+"</telephone>\n");
      bw.write("      </entree>\n");
          }
bw.write("</repertoire>\n");
bw.flush( );
```

Une technique simple de sérialisation …

Desérialisation : XML → Repertoire

Le répertoire Java va être créé au fur et à mesure de la lecture du flux xml

<repertoire creation="2014-12-01">

créer l'objet Repertoire

<proprietaire>Louis Dupond

enregistrer le nom de son propriétaire

</proprietaire>

<entree>

<nom>

dans la balise nom

Durand

noter le nom

</nom> <telephone>

dans la balise telephone

0323425342

noter le téléphone

</telephone>

</entree>

enregistrer la nouvelle entrée

XML → OBJET

public void characters(char[ ] ch, int start, int length) {}
Cette méthode ne dit pas dans quelle balise on se trouve. On va devoir le gérer.

private boolean `inProprio,inNom,inTelephone;`

`mises à true dans startElement et à false dans endElement`

public void startElement(String uri, String localName, String rawName, Attributes atts) {}

Le repertoire sera créé dans startElement lors de l'arrivée de la balise repertoire

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

Si on est dans une balise nom ou telephone, il faut mémoriser les valeurs

si c'est la balise proprietaire, on peut enregistrer le nom du proprietaire

private String nom, telephone;

public void endElement(String uri, String localName, String rawName) {}

si c'est la balise entree, on peut ajouter la nouvelle entree

Le handler startElement

```
class RepHandler extends DefaultHandler {
    private Repertoire rep;
    boolean inProprio,inNom,inTelephone;
    private String nom,tel;

public void startElement(String uri, String localName, String rawName, Attributes atts) {
if (rawName.equals("repertoire")){
     String creationDate = atts.getValue("creation");
    rep=new Repertoire(creationDate);
    }
else if (rawName.equals("proprietaire"))    inProprio=true;
else if (rawName.equals("telephone"))      inTelephone=true;
else if (rawName.equals("nom"))            inNom=true;
}
```

Le handler : characters

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

String value=String.copyValueOf(ch, start, length).trim();
if (inProprio)          rep.setProprietaire(value);
else if (inNom)         nom=value;
else if (inTelephone)tel=value;

}
```

Le handler : endElement

```
public void endElement(String uri, String localName, String rawName) {
if (rawName.equals("proprietaire"))       inProprio=false;
else if (rawName.equals("telephone"))     inTelephone=false;
else if (rawName.equals("nom"))           inNom=false;
else if (rawName.equals("entree"))
            rep.ajouterEntree(nom, tel);
}
```

```
public Repertoire getRepertoire() {
    return this.rep;
}
```

Utilisation de RepHandler

```
public static void main(String[] args)throws Exception {

SAXParserFactory spf = SAXParserFactory.newInstance();
SAXParser pf = spf.newSAXParser();

RepHandler handler = new RepHandler();

pf.parse(new File("rep.xml"),handler);
Repertoire repertoire = handler.getRepertoire();

repertoire.toXml(new OutputStreamWriter(System.out));
}
```

Xml et espaces de noms

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- le répertoire de Dupond -->
<rep:repertoire creation="2014-12-01"
   xmlns:rep="http://www.math-info.univ-paris5.fr/~pary">
   <rep:proprietaire>Louis Dupond</rep:proprietaire>
   <rep:entree>
     <rep:nom>Durand</rep:nom>
     <rep:telephone>0323425342</rep:telephone>
   </rep:entree>
</rep:repertoire>
```

Espace de noms : pour éviter les conflits de noms

xmlns : Xml NameSpace

Un espace de nom est caractérisé par une **URI (Uniform Resource Identifier)**

Xml et espaces de noms

```
class RepHandlerDemo extends DefaultHandler {
…
public void startElement(String uri, String localName, String rawName, Attributes atts){
System.out.println("Debut de balise uri="+uri+" ln="+localName+" rn="+rawName);
    }
…}
```

```
…
Debut de balise uri=http://www.math-info.univ-paris5.fr/~pary ln=telephone rn=rep:telephone
…
```

Si les espaces de noms sont utilisés, il faut utiliser uri et localName et non pas rawName.

```
SAXParserFactory spf = SAXParserFactory.newInstance();
spf.setNamespaceAware(true); // pour préciser que l'on gère les espaces de noms
SAXParser pf = spf.newSAXParser();
```

Schéma xml des répertoires (fichier xsd)

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="repertoire" >
  <xs:complexType>
    <xs:sequence>
      <xs:element name="proprietaire"  type="xs:string"/>
      <xs:element name="entree"  maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="nom"  type="xs:string"/>
            <xs:element name="telephone"  type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="creation" type="xs:string"/>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Schéma xml des répertoires : variante

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="entree" >
      <xs:complexType>
         <xs:sequence>
            <xs:element name="nom"  type="xs:string"/>
            <xs:element name="telephone"  type="xs:string"/>
         </xs:sequence>
      </xs:complexType>
 </xs:element>
 <xs:element name="repertoire" >
  <xs:complexType>
      <xs:sequence>
        <xs:element name="proprietaire"  type="xs:string"/>
        <xs:element ref="entree" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="creation" type="xs:string"/>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Validation d'un document xml

```
    static final String JAXP_SCHEMA_LANGUAGE =
            "http://java.sun.com/xml/jaxp/properties/schemaLanguage";
    static final String W3C_XML_SCHEMA =
            "http://www.w3.org/2001/XMLSchema";
    static final String JAXP_SCHEMA_SOURCE =
            "http://java.sun.com/xml/jaxp/properties/schemaSource";
String xsdFile="rep.xsd";
String xmlFile="rep2.xml";
SAXParserFactory spf = SAXParserFactory.newInstance();
// le document va être validé
spf.setValidating(true);
SAXParser sp = spf.newSAXParser();
// schéma xml w3c
sp.setProperty(JAXP_SCHEMA_LANGUAGE, W3C_XML_SCHEMA);
// localisation du fichier xsd
sp.setProperty(JAXP_SCHEMA_SOURCE, xsdFile);
sp.parse(xmlFile, new RepHandler( ));
```

Schéma xml des répertoires avec NameSpace

```
<?xml version="1.0" encoding="UTF-8" ?>
 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"

      <!--  le nameSpace des éléments du fichier xml -->
       targetNamespace="http://www.mi.parisdescartes.fr/~pary"
       <!--  par défaut les éléments fournis ci-dessous sont qualifiés -->
       elementFormDefault="qualified"
       >
<xs:element name="repertoire" >
…
```

DOM

Création en mémoire de l'arborescence correspondant au document XML

- Document
- Node
- Element
- Attributes

Avantages Sur SAX

Accès possible aux divers nœuds du fichier xml

L'arborescence peut être modifiée

Le document xml correspondant peut être généré

Inconvénients

Tout le document est en mémoire

DOM : Interface Node

```
/**A code representing the type of the underlying object
  ELEMENT_NODE, ATTRIBUTE_NODE, TEXT_NODE , COMMENT_NODE , DOCUMENT_NODE ...*/
public short getNodeType( );

/** The name of this node, depending on its type*/
public String getNodeName();

/** The value of this node, depending on its type */
public String getNodeValue( )  throws DOMException;
/**Modify the value of this node*/
public void setNodeValue(String nodeValue)throws DOMException;

   /** The parent of this node.     */
public Node getParentNode();

/** A NodeList that contains all children of this node*/
public NodeList getChildNodes();

/** A NamedNodeMap containing the attributes of this node (if it is an Element)
      or null otherwise */
public NamedNodeMap getAttributes( );
/** returns the text content of this node and its descendants */
public String getTextContent( ) throws DOMException;
```

DOM : Interface Document

```
public interface Document extends Node {
/** rend l'élément racine de ce document.*/
    public Element getDocumentElement();

    public Element createElement(String tagName)throws DOMException;
    public Text createTextNode(String data);
    public Comment createComment(String data);
    public CDATASection createCDATASection(String data) throws DOMException;
…
}
```

DOM : Interface NodeList et NamedNodeMap

```
public interface NodeList {
    public Node item(int index);
    public int getLength();
}

public interface NamedNodeMap {
     public Node getNamedItem(String name);
     public Node item(int index);
     public int getLength();
…
}
```

DOM : Interface Element

```
public interface Element extends Node {
  /**
    * Returns a <code>NodeList</code> of all descendant <code>Elements</code>
    * with a given tag name, in document order.
    * @param name The name of the tag to match on.
    *                    The special value "*" matches all tags.
    * @return A list of matching <code>Element</code> nodes.
    */
public NodeList getElementsByTagName(String name);

public String getAttribute(String name);
public void setAttribute(String name,String value)throws DOMException;
public void removeAttribute(String name)throws DOMException;
public Attr getAttributeNode(String name);
public NodeList getElementsByTagNameNS(String namespaceURI,String localName) throws DOMException;
...
}
```

Parcours de l'arborescence DOM

```
private static void parcourir(Node node,String marge) {
    afficherElement(node,marge);
    afficherFils(node,marge);
}

private static void afficherFils(Node node,String marge) {
   NodeList child=node.getChildNodes();
   for (int i=0;i<child.getLength();i++)
       parcourir(child.item(i),marge+"   ");
}
private static void afficherElement(Node node,String marge) {
  if (node.getNodeType()==Element.TEXT_NODE){
    if (node.getNodeValue().trim().length()>0)
       System.out.println("Texte="+node.getNodeValue());
    }
else if (node.getNodeType()==Element.ELEMENT_NODE)
   {System.out.print("Element="+node.getNodeName());
    afficherAttributs(node,marge);
}}
```

Parcours de l'arborescence DOM

```
private static void afficherAttributs(Node node,String marge) {
   NamedNodeMap nnm=node.getAttributes();
  for (int i=0;i<nnm.getLength();i++)
     System.out.print(nnm.item(i));
  System.out.println();
}
```

```
Element=repertoire  creation="2014-12-01"
Element=proprietaire
Texte=Louis Dupond
Element=entree
Element=nom
Texte=Durand
Element=telephone
Texte=0323425342
Element=entree
Element=nom
Texte=Marie
Element=telephone
Texte=0987654321
```

DOM : --> OBJET

```
private static Repertoire getRepertoire(Document document){
    Element racine =document.getDocumentElement( );
    Repertoire rep = new Repertoire(racine.getAttribute("creation"));
          NodeList listTel = racine.getElementsByTagName("proprietaire") ;
    rep.setProprietaire(listTel.item(0).getTextContent( ));
    NodeList listEntree =racine.getElementsByTagName("entree");
    for (int i=0;i<listEntree.getLength();i++){
      Element entree = (Element)listEntree.item(i);
            NodeList listNom = entree .getElementsByTagName("nom");
            String nom= listNom.item(0).getTextContent( );
            NodeList listTel = entree.getElementsByTagName("telephone");
            String tel=listTel.item(0).getTextContent( );
            rep.ajouterEntree(nom, tel);
            }
  return rep;}
```

OBJET → DOM (1)

```
public static Document getDocument(Repertoire repertoire) throws ParserConfigurationException{

// recuperation d'une fabrique de constructeur de document
      DocumentBuilderFactory fabrique = DocumentBuilderFactory.newInstance();

// recuperation d'un constructeur de document
      DocumentBuilder builder = fabrique.newDocumentBuilder();

// création d'un nouveau document
Document document = builder .newDocument();
// création du nœud principal
Element nodeRep = document.createElement("repertoire");
// ajout de l'attribut creation au nœud principal
nodeRep.setAttribute("creation", repertoire.getCreationDate());
// ajout du noeud au document
document.appendChild(nodeRep);
// on crée ici les autres nœuds (voir slide suivant)
…
// on rend le document DOM ainsi créé
return document;
}
```

OBJET → DOM (2)

```
public static Document getDocument(Repertoire repertoire) throws ParserConfigurationException{
DocumentBuilderFactory fabrique = DocumentBuilderFactory.newInstance();
DocumentBuilder builder= fabrique.newDocumentBuilder();
Document document = builder.newDocument();
Element nodeRep = document.createElement("repertoire");
nodeRep.setAttribute("creation", repertoire.getCreationDate());
document.appendChild(nodeRep);

Element nodeProp = document.createElement("proprietaire");
nodeProp.appendChild(document.createTextNode(repertoire.getName()));
nodeRep.appendChild(nodeProp);
for (String name : repertoire.getNames()){
        Element nodeEntree = document.createElement("entree");
        Element nodeNom = document.createElement("nom");
        nodeNom.appendChild(document.createTextNode(name));
        nodeEntree.appendChild(nodeNom);
        Element nodeTelephone = document.createElement("telephone");
        nodeTelephone.appendChild(document.createTextNode(repertoire.getTel(name)));
        nodeEntree.appendChild(nodeTelephone);
        nodeRep.appendChild(nodeEntree);
    }
return document;
```

Xml --> Document

```
/** rend le document DOM correspondant à un fichier xml */
public static Document getDocumentFromXmlFile(File xmlFile)
                                        throws ParserConfigurationException, SAXException{

    // recuperation d'une fabrique de constructeur de document
    DocumentBuilderFactory fabrique = DocumentBuilderFactory.newInstance();

   // recuperation d'un constructeur de document
    DocumentBuilder builder = fabrique.newDocumentBuilder();

    // parsing du fichier avec le documentBuilder
    Document document = builder.parse(xmlFile);

return document;
}
```

Document --> Xml

```
/** met dans streamResult le contenu xml correspondant au document */
public static void documentToXml(Document document,StreamResult sortie)
                                                        throws TransformerException{
// Récupération d'une fabrique de transformeur
TransformerFactory tfabrique = TransformerFactory.newInstance();
// Récuperation d'un transformeur
Transformer transformeur = tfabrique.newTransformer();

// paramétrage du transformeur
    Properties proprietes = new Properties();
    proprietes.put("encoding", "UTF-8");
    proprietes.put("indent", "yes");
    transformeur.setOutputProperties(proprietes);

// création de la source
    Source entree = new DOMSource(document);
// utilisation du transformeur pour mettre le document sous forme xml dans sortie
    transformeur.transform(entree,sortie);
}
```

Document --> Xml

```
public class Main {

 public static void main(String[] args) throws Exception {
   File dir = new File("src/up5/mi/pary/jc/xml/dom");

   Document document = Util.getDocumentFromXmlFile(new File(dir,"repertoire.xml"));

   Util.parcourir(document, "");

   Util.documentToXml(document, new StreamResult(new File(dir,"sortie.xml")));

   Repertoire rep = Repertoire.getRepertoire(document);

     try (Writer writer =
           new OutputStreamWriter(new FileOutputStream(new File(dir,"sortie2.xml"))))
       {rep.toXml(writer);}
 }
}
```