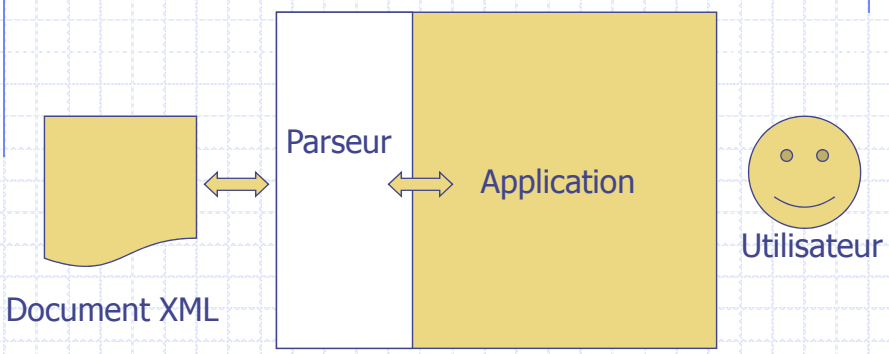
Développer des projets en XML

Les parseurs XML

Rappel sur les parseurs

- Un parseur est un composant logiciel exploité par une application pour consommer ou produire des documents XML:
  - navigateur
  - éditeur
  - tout programme exploitant technologies XML
- Egalement, parseur utilisable comme programme autonome en ligne de commande.

Schéma application et parseur XML



Rappel sur les parseurs

- Caractéristiques des parseurs:
  - DOM et/ou SAX
  - Support de DTD et/ou XML Schema
  - Respect des normes
  - Validant ou non
  - Support de namespace
  - Performance
  - Support dernières technologies

Liste non-exhaustive de parseurs pour Java

- xerces du groupe Apache
  - Initialement don d'IBM
  - XML1.0, SAX 1.0, SAX 2.0
  - DOM Level 1, DOM Level 2
  - Validant et non-validant
  - Gratuit sous license Apache software
  - Version >=1.3 support de XML Schema
  - Disponible en JAVA, Perl et C++
  - xerces 2 en cours de développement : plus performant, plus modulaire
  - Lien : http://xml.apache.org/xerces-j/index.html

Liste non-exhaustive de parseurs pour Java

- Crimson
  - parseur fourni avec JAXP comme exemple d'implémentation intégré désormais dans xerces 2
- Parseur pour Java d'Oracle
- XML4C et XML4J de IBM (AlphaWorks, origine de xerces)
- Parseur AELfred : Java, utilisation applet

Parseur Microsoft

- MS XML >=3.0
- Gratuit, installé avec IE ou séparément
- Composant Active X avec parseur XML validant ou non, et processeur XSL-T
- >= 4.0 support de XML Schema

Liste des parseurs pour le langage C

- Site intéressant pour une liste de parseurs:
  - http://wdvl.internet.com/Software/XML/parsers.html
  - http://www.multimania.com/pensarguet/XML/XML-parser.htm
- Parser en C, C++ :
  - xerces (C++): http://xml.apache.org/xerces-j/index.html
  - xmlbooster : http://www.xmlbooster.com/
  - expat : http://sourceforge.net/projects/expat/
  - LT XML : http://www.ltg.ed.ac.uk/software/xml/

Liste des parseurs pour le langage C

- Parser en C, C++ (suite) :
  - MS XML (VC++) : http://www.microsoft.com/workshop/xml/parser/jparser.asp
  - XML parser for C (et un autre pour C++) de Oracle : http://technet.oracle.com/tech/xml/parser_cpp2/

Liste des parseurs pour le langage PHP

- Parser en PHP:
  - extension à PHP (libexpat.a) : http://www.zend.com/manual/ref.xml.php ; http://dev.nexen.net/docs/php/annotee/manuel_xml.install.php
  - PHP XML Parser de Manuel Lemos : http://www.phpclasses.upperdesign.com/browse.html/class/4

API XML: SAX

- SAX : Simple API for XML parsing
- Né du besoin d'une interface commune de programmation pour exploiter différents parseurs du marché
- En lieu et place des interfaces propriétaires des parseurs
- Développé et maintenu par un groupe de développeurs XML-DEV (nom d'une mailing liste au sujet des technologies XML)
- SAX 1.0 en 1998, SAX 2.0 en décembre 2000, avec, entre autres, support des namespace
- Site de référence : http://www.megginson.com/SAX et http://www.saxproject.org/

API XML: SAX

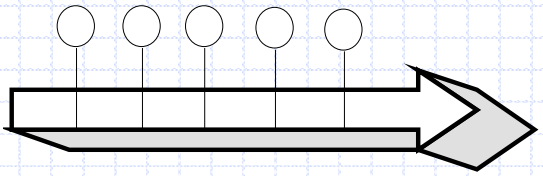
- L'arbre XML n'est pas vu comme une structure de données mais comme une suite d'événements générés par le parseur
- Les sortes d'événements sont:
  - début du document
  - fin du document
  - début d'élément
  - fin d'élément
  - données textes rencontrées
  - instruction processeur
  - ...

API XML: SAX

- SAX lit le document XML du début à la fin, en séquence
- Chaque événement invoque une méthode implémentée dans le programme
- Un arbre XML peut être construit en réponse par l'application (parseur DOM souvent basé sur parseur SAX)
- En général, nécessite l'implémentation d'une machine à état pour assurer le traitement nécessaire

API XML: SAX

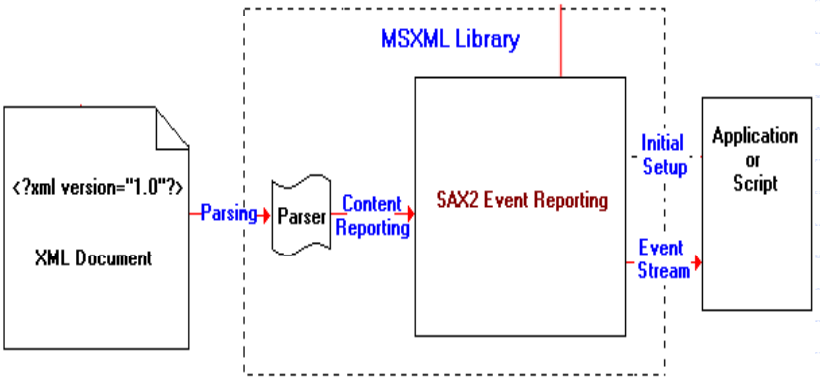
- événementiel
  - "début de l'élément produit, texte de l'élément, fin de l'élément produit,..."



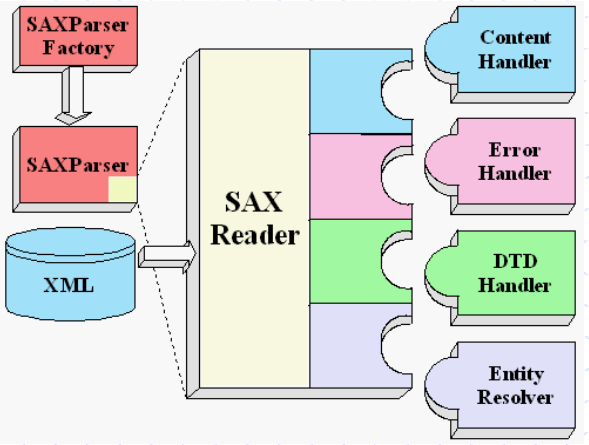
Collaboration parseur SAX - application

- Mode de fonctionnement :
  - Objet parseur et objets listener
  - Les objets listener s'enregistrent auprès de l'objet parseur comme "gestionnaires d'événements"
  - Les objets listener implémentent une interface que l'objet parseur connaît.
  - Evénements sont rapportés via principe de "callback" aux listener
  - A chaque événement correspond une méthode de l'interface qu'implémentent les objets listener
  - Semblable au modèle de délégation d'événements de AWT

Fonctionnement d'un parser avec l'interface SAX



Parseur SAX et gestionnaire d'événements



SAX exemple

```
import java.io.FileReader;
import org.xml.sax.XMLReader;
import org.xml.sax.Attributes;
import org.xml.sax.InputSource;
import org.xml.sax.helpers.DefaultHandler;
import org.apache.xerces.parsers.SAXParser;

public class MySAXApp extends DefaultHandler {
    Locator locator;
    public static void main (String args[]) throws Exception {
        XMLReader xr = new SAXParser();
        MySAXApp handler = new MySAXApp();
```

SAX exemple

```
        xr.setFeature(
                "http://xml.org/sax/features/validation", false);
        xr.setContentHandler(handler);
        xr.setErrorHandler(handler);
        for (int i = 0; i < args.length; i++) {
                FileReader r = new FileReader(args[i]);
                xr.parse(new InputSource(r));
        }
}
public MySAXApp () {
        super();
}
```


SAX exemple

```
public void setDocumentLocator(Locator _locator) {
        locator = _locator;
}
public void startDocument () {
        System.out.println("Start document");
}
public void endDocument () {
        System.out.println("End document");
}
```

SAX exemple

```
public void startElement
    (String uri, String name, String qName, Attributes atts) {
    if ("".equals (uri))
            System.out.println("Start element: " + qName);
    else
            System.out.println("Start element: {" + uri + "}" + name);
    System.out.println(">>> Line : " + locator.getLineNumber() + ",
             column : " + locator.getColumnNumber());
    for (int i = 0 ; i < atts.getLength() ; i++) {
            System.out.println("\tAttribute: " + atts.getLocalName(i));
            System.out.println("\t\tValue: " + atts.getValue(i));
    }
}
```


SAX exemple

```
    public void endElement (String uri, String name, String qName){
            if ("".equals (uri))
              System.out.println("End element: " + qName);
            else
              System.out.println("End element:   {" + uri + "}"
                      + name);
    }
    public void characters (char ch[], int start, int length) {
            String cString = new String(ch, start, length);
            System.out.println("Characters:    " + cString);
    }
}
```

Parseur SAX

- Avantage :
  - Plus performant
  - Composant plus léger (ex.: parseur Aelfred pour applet)
  - moins gourmand en mémoire car seule une faible partie du document est stockée à chaque moment.
  - Dans certain cas, la seule solution exploitable.

Parseur SAX

- Désavantages :
  - Fonctionnalités de bas niveau
  - Le parseur ne maintient aucun contexte
  - Rien de prévu pour la modification du contenu de l'arbre XML, nécessité de recréer une nouvelle arborescence
  - Impossible, via le parseur, de connaître les relations parent/enfant entre les éléments, l'application doit s'en charger
  - Selon les objectifs, l'application est plus complexe à réaliser.

Cas d'utilisation de parseur SAX

- Application exploitant parties de document XML
- Application embarquée
- Application serveur nécessitant une exploitation minimum de ressources

DOM, Document Object Model

- DOM regroupe les spécifications d'interfaces définies et normalisées par le W3C
- Interfaces implémentées par des concepteurs de parseurs permettant d'accéder à un document XML comme une structure d'objets
- API DOM permet d'accéder, de manipuler et de gérer les documents XML:
  - Parcourir les éléments et naviguer dans l'arborescence
  - Créer et modifier un document XML
  - Créer, modifier, détruire, copier des éléments/attributs

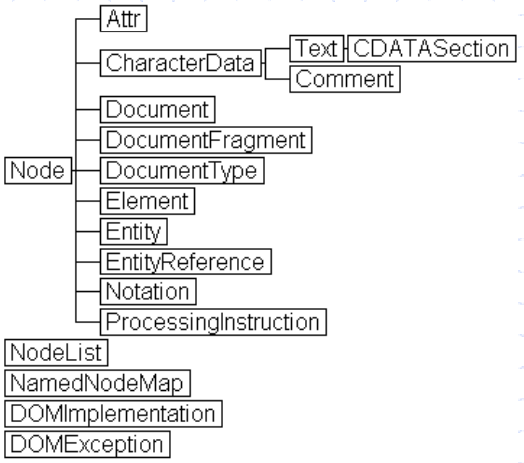
DOM, Document Object Model

- À l'origine le concept de DOM avait été imaginé pour représenter les éléments d'un page Web sous forme d'une collection d'objets
- DOM a été généralisé pour les besoins de XML
- Spécification indépendante de tout langage et exprimée en IDL (Interface Definition Language de l'OMG)

DOM, Document Object Model

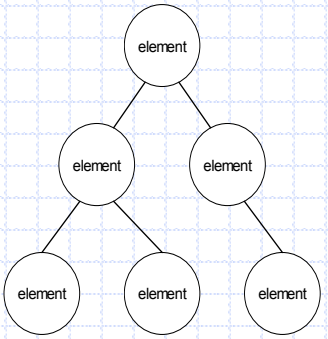
- DOM Level 1, août 98 :
  - "core" des spécifications pour XML
  - plus extensions spécifiques pour HTML
- DOM Level 2, novembre 2000 :
  - Support namespace
  - Modularisation des spécifications : core, events, styles, …
- DOM Level 3 en cours.

Extrait de la hiérarchie des interfaces du package org.w3c.dom



API XML: DOM

- Avec l'API DOM, le document XML est vu comme une structure hiérarchique d'objets
  - "un élément facture contient un élément produit qui contient un élément…"

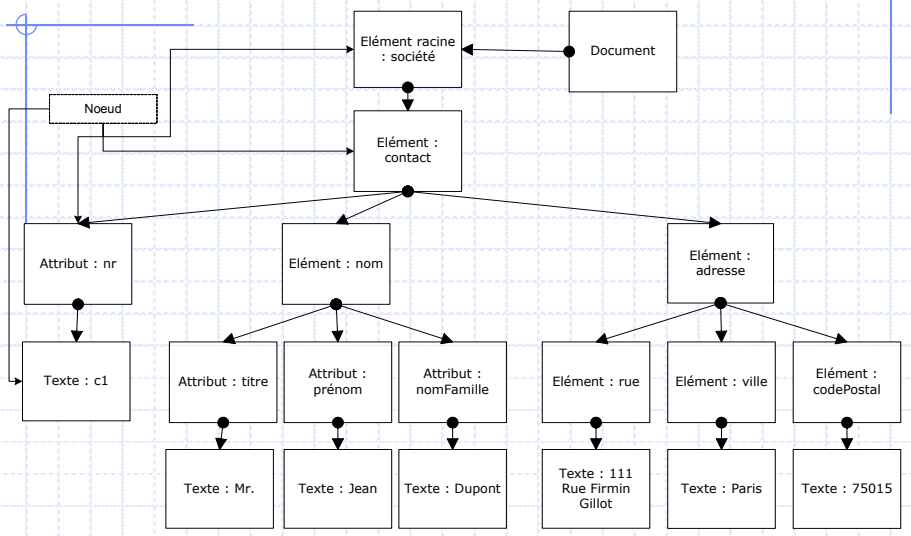


Structure hiérarchique d'un document XML et concept de noeud

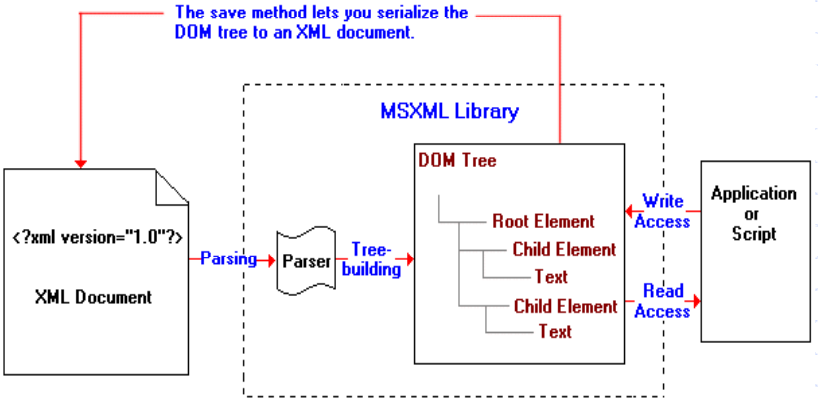
- Un document XML est une structure hiérarchique de nœuds
- Tout est noeud

```
<?xml version="1.0" encoding="UTF-8" ?>
   <société>
        <contact nr="c1">
          <nom titre="Mr."
                prénom="Jean" nomFamille="Dupont" />
          <adresse>
                <rue>111 Rue Firmin Gillot</rue>
                <ville>Paris</ville>
                <codePostal>75015</codePostal >
          </adresse>
        </contact>
   </société>
```


Structure hiérarchique et nœuds



Fonctionnement d'un parser avec l'interface DOM



Exemple de programme

```
import org.w3c.dom.*;
import javax.xml.parsers.*;
import javax.xml.transform.*;
import javax.xml.transform.stream.*;
import javax.xml.transform.dom.*;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.net.URL;
```

Exemple de programme

```
public class ReadXML {
    public static void main( String[] args ) {
        Document doc = null;
        boolean validation = true;

        if( args.length == 0 ) {
            String usage = "Usage:\tReadXML filename.xml
                [true|false]\n";
            usage += "\tThe second parameter switches validation,
                default is true (on)";
            System.out.println( usage );
            return;
        }
```


Exemple de programme

```
        String source = args[0];
        if( args.length == 2 ) {
            validation = Boolean.valueOf(args[1]).booleanValue();
        }

        try {
            System.out.println( "Reading from "+source+"
                with"+(validation?"":"out")+" validation." );
            DocumentBuilderFactory dbf =
                DocumentBuilderFactory.newInstance();
            dbf.setValidating(validation);
            DocumentBuilder db = dbf.newDocumentBuilder();
            doc = db.parse(source);
            Node root = (Node)doc.getDocumentElement();
```

Exemple de programme

```
Node root = (Node)doc.getDocumentElement();
System.out.println("Done, here's the same XML re-output: -\n");
TransformerFactory tFactory =
        TransformerFactory.newInstance();
Transformer transformer = tFactory.newTransformer();
transformer.transform(new DOMSource(doc), new
        StreamResult(System.out));
}
catch( FileNotFoundException fnfEx ) {
        System.out.println(source+" was not found");
}
catch( Exception ex ) {
        ex.printStackTrace();
        System.out.println( ex.getMessage() );
} } }
```


Utilisation de XSLT pour la sortie

- Utilisation d'un "transformer" (processeur XSLT) pour projeter un document dans un flux résultat (standard de sortie ou fichier):

```
TransformerFactory tFactory =
        TransformerFactory.newInstance();
Transformer transformer =
        tFactory.newTransformer();
transformer.transform(new DOMSource(doc),
        new StreamResult(System.out));
```

Analyser la source du document XML

- Méthode parse() de DocumentBuilder
- Paramètres:
  - URL
  - File
  - InputStream
  - org.xml.sax.InputSource
- Valeur retour :
  - objet de la classe Document
  - Représente la structure arbre construite en mémoire centrale après analyse du document

Obtenir l'élément racine du document

- La classe Document fournit la méthode getDocumentElement()
- Cette méthode retourne un objet Element référençant l'élément racine du document
- La classe Element est une sous-classe de Node, donc la conversion explicite est possible

Les nœuds et les éléments

- La classe Node est la classe de base de toutes les autres classes:
  - Document, Element, Attribute, Text, CDATA, Entity, ProcessingInstruction, …
- En DOM, tout est Node et le nœud à un type
- noeud.getNodeType() permet de connaître le type du nœud :
  - ELEMENT_NODE
  - ATTRIBUTE_NODE
  - TEXT_NODE

Méthodes de navigation

- Class Node prévoit les méthodes suivantes pour la navigation:
  - getFirstChild(), getLastChild(), getParentNode(), getPreviousSibling(), getNextSibling() : retournent un autre Node
  - getChildNodes() : retourne une NodeList
- Class NodeList prévoit les méthodes:
  - getLength() : nombre d'item de la liste
  - item(index) : item de la liste, retourne un Node
- Class Document prévoit: getElementsByTagName() qui retourne une NodeList des éléments qui correspondent au nom donné ("*" retourne tous les nœuds).

Méthodes d'information sur les noeuds

- Class Node prévoit :
  - hasChildNodes() : retourne true si le nœud a des enfants
  - getNodeName() : retourne le nom du nœud
  - getNodeValue(), setNodevalue() : consulte et modifie le contenu du nœud, surtout utile avec les nœuds de type Attr, Text et CDATASection
  - getAttributes() : retourne les attributs dans une collection de la classe NamedNodeMap()
- Class NamedNodeMap prévoit :
  - getLength(), item(), removeNamedItem(), getNamedItem(), setNamedItem()

Méthodes de création

- Class DocumentBuilder prévoit les méthodes:
  - newDocument() :
- Class Document prévoit les méthodes:
  - createElement() : création d'un élément avec un certain nom
  - createTextNode() : création d'un nœud de type Text
  - createAttribute() : création d'un nœud de type Attr (attribut)
  - createComment() : création d'un nœud de type Comment
- Ces nouveaux nœuds ne sont pas rattachés à l'arbre, il faudra les rattacher grâce aux méthodes de manipulation de nœuds.

Méthodes de manipulation des noeuds

- Class Node prévoit:
  - insertBefore() : insertion d'un nœud avant un autre
  - replaceChild() : remplace un noeud enfant par un autre
  - removeChild() : retirer un nœud enfant
  - appendChild() : ajouter un nouveau nœud enfant à la fin de la liste de enfants actuels

Méthodes de la classe Element

- Un nœud de type ELEMENT_NODE peut être converti en un objet Element
- La classe Element prévoit:
  - getTagName() : nom de l'élément
  - getAttribute() : retourne la valeur d'un certain attribut
  - setAttribute() : modifie la valeur d'un certain attribut
  - removeAttribute() : retire un certain attribut
  - getElementsByTagName() qui retourne une NodeList des éléments qui correspondent au nom **à partir de l'élément** ("*" retourne tous les nœuds).

Exemple de programme Java utilisant le DOM pour la création d'un document

```
import org.w3c.dom.*;
import javax.xml.parsers.*;
import javax.xml.transform.*;
import javax.xml.transform.stream.*;
import javax.xml.transform.dom.*;
import java.io.PrintWriter;
import java.io.IOException;
public class CreatePresident {
    public static void main( String[] args ) {
        Document doc;
        Element president;
        Element person;
        Element firstName;
        Element surname;
```


Exemple de programme Java utilisant le DOM

```
        try {
            DocumentBuilderFactory dbf =
                DocumentBuilderFactory.newInstance();
            DocumentBuilder db = dbf.newDocumentBuilder();
            doc = db.newDocument();
            person = doc.createElement("Personne");
            firstName = doc.createElement("Prénom");
            firstName.appendChild( doc.createTextNode("Michel") );
            person.appendChild(firstName);
            surname = doc.createElement("NomFamille");
            surname.appendChild( doc.createTextNode("COLUCCI")
    );
            person.appendChild(surname);
            president = doc.createElement("Président");
```

Exemple de programme Java utilisant le DOM

```
                president.setAttribute("Pays","FR");
                president.appendChild( person );
                doc.appendChild( president );
                TransformerFactory tFactory =
                        TransformerFactory.newInstance();
                Transformer transformer = tFactory.newTransformer();
                transformer.transform(new DOMSource(doc), new
                        StreamResult(System.out));
        }
        catch( ParserConfigurationException pcEx ) {
                System.out.println("ParserConfigurationException:
                        "+pcEx.getMessage());
                pcEx.printStackTrace();
        }
```


Exemple de programme Java utilisant le DOM

```
        catch( TransformerConfigurationException tcEx ) {
                System.out.println("TransformerConfigurationException:
                        "+tcEx.getMessage());
                tcEx.printStackTrace();
        }
        catch( TransformerException tEx ) {
                System.out.println("TransformerException:
                        "+tEx.getMessage());
                tEx.printStackTrace();
        }
    }
}
```