

Une brève introduction à XML

et à son utilisation en Java

```
<Content id="" Class="SimpleText" DefaultType="Text"
  <P1><![CDATA[]]></P1>
  <P2 x="" y="" RelativeGap="" />
</Content>
<Content id="" Class="SimpleText" DefaultType="Text"
  <P1><![CDATA[]]></P1>
  <P2 x="" y="" RelativeGap="" />
</Content>
<Content id="" Class="SimpleText" DefaultType="Text"
  <P1><![CDATA[]]></P1>
  <P2 x="" y="" RelativeGap="" />
</Content>
```

Module Structures de données

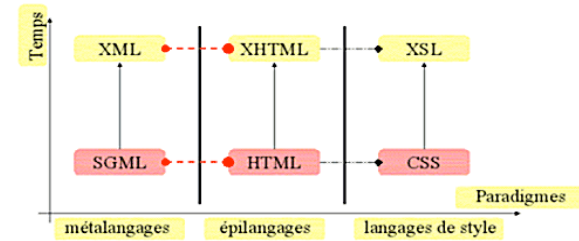
Master IC

V. Berry

I - La syntaxe XML

Définition de XML :

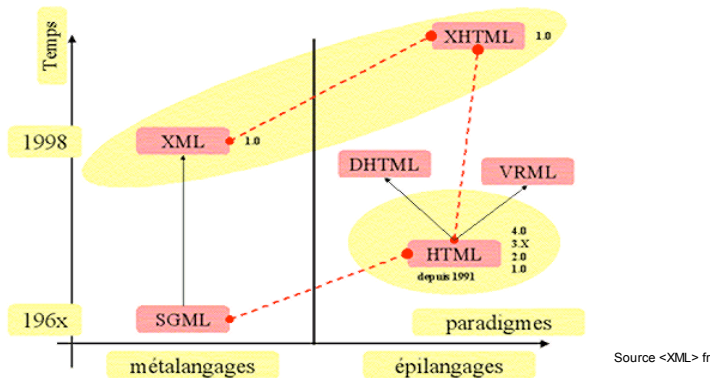
- XML = eXtensible Markup Language
- c-a-d un langage de balisage extensible. En fait XML est plus un métalangage (c-a-d un langage permettant de décrire des langages).
- XML est un sous-ensemble récent du SGML



Source <XML> fr

Définition de XML

XML et HTML donnent ensemble le XHTML qui doit être compréhensible par les navigateurs actuels tout en respectant les contraintes imposées par XML.



Utilisation de XML

- En une phrase, XML est un format pour stocker des données structurées.
- Que sont des données structurées ?
 - Un carnet d'adresses
 - Un fichier de configuration
 - Des transactions financières
 - Des coordonnées géographiques
 - Un réseau métabolique
 - Une ontologie
 - ...

Un aperçu de fichier XML

```
<?xml version="1.0"?>
<addressBook>
  <person>
    <name> <family>Eastwood</family> <given>Clint</given> </name>
    <email>clint@laposte.fr</email>
  </person>

  <person>
    <name> <family>Champignon</family> <given>Pierre</given> </name>
    <email>champignon@lirmm.fr</email>
  </person>
</addressBook>
```

Un aperçu de fichier XML

```
<?xml version="1.0"?>
<recipe>
  <author>Carol Schmidt</author>
  <recipe_name>Chocolate Chip Bars</recipe_name>
  <meal>Dinner
    <course>Dessert</course>
  </meal>

  <ingredients>
    <item>2/3 C butter</item>    <item>2 C brown sugar</item>
    <item>1 tsp vanilla</item>   <item>1 3/4 C unsifted all-purpose flour</item>
  </ingredients>

  <directions>
    Preheat oven to 350 degrees. Melt butter; combine with brown sugar and vanilla in large mixing
    bowl. Set aside to cool. Combine flour. Spread in greased 13-by-9-inch pan. Bake for 25 to 30
    minutes until golden brown; cool.
  </directions>
</recipe>
```

Avantages de XML

- Permet d'échanger des données entre applications et plateformes **diverses**.
- Inclut les informations de structure en plus du contenu en données, ce qui simplifie les **traitements automatiques** (parsing, génération de rapports, etc).
- Format proche du texte, donc **lisible par les humains** dans certaines limites.
- Format qui tend à s'imposer partout : la plupart des **langages de programmation** offrent des **outils** pour lire/écrire en XML. La plupart des **SGBD** permettent d'exporter en XML, ce qui facilite le **portage** de leurs données à un autre système.

Désavantages ?

- XML est **verbeux** : les règles de composition de documents qu'il impose, rendent ces documents longs.
Relativisons :
 - il s'agit de texte (facilement compressible), et l'espace disque est une ressource peu coûteuse actuellement
 - il n'est pas vraiment fait pour être lu par les humains (plutôt par les applications), mais il peut quand même être lu par eux en cas de besoin.

Premières remarques

- Un fichier XML est un simple fichier texte, contenant des balises.
- Le format XML ressemble à du code HTML comme affiché par les navigateurs web.
- Le balisage est toutefois plus strict : toute balise ouvrante doit être fermée, de même que toute apostrophe, guillemets, etc.
- La particularité de XML est qu'aucune balise n'est prédéfinie : c'est au concepteur de définir les balises qui ont du sens pour lui.

Les composants du XML

Essentiellement :

- **Element** : désigne un composant logique d'un document, normalement composé d'une balise d'ouverture et d'une de fermeture encadrant un contenu.
`<STREET> 4296 Rue du razeur </STREET>`
- **Attribut** : un élément peut avoir des attributs sous la forme de nom/valeur placés après le nom de balise ouvrante.
`<PROGRAM length="10" name="affichage">`
- **Commentaire** : permet d'insérer une description qui sera ignorée par un processeur XML.
`<!-- Ceci est un commentaire -->`
- **Instruction de traitement** : permet de passer des informations à l'application traitant le fichier XML.
`<? Application data ?>`

DTD : Document Type Definition

Une DTD est:

- Une **grammaire** (ou jeu de règles) qui :
 - définit la listes des éléments (balises utilisables) ainsi que leurs attributs
 - et qui spécifie leur possible imbrication (relation).
- Chaque balise du document est décrite une fois comme un **élément**, on précise aussi ses « sous-balises » (d'autres éléments) et ses **attributs**.
- La DTD peut être référencée par un URI ou incluse directement dans le document XML
- Ils existent d'autres types de grammaires comme *XML Schema (XSD)*, *Relax NG*, dont la puissance sémantique est plus élevée (c.à.d que l'on peut exprimer plus de contraintes).
 - DTD est la grammaire la plus répandue
 - XML Schema est utilisé par ex. pour formaliser des langages "webservices", par ex. SOAP

Exemple de DTD

```
<!ELEMENT liste_livres (livre+)>
<!ELEMENT livre (titre,auteur+,éditeur,description?,prix)>
<!ATTLIST livre ISBN ID #REQUIRED>

<!ELEMENT titre (#PCDATA)>
<!ELEMENT auteur (#PCDATA)>
<!ELEMENT éditeur (#PCDATA)>
<!ELEMENT prix (#PCDATA)>

<!ELEMENT couverture (#PCDATA)>
<!ATTLIST couverture
    resume (oui | non ) #REQUIRED
    epaisseur CDATA #IMPLIED
    code IDREF #REQUIRED>
```

Attachement d'une DTD à un document XML

Il existe 4 façons d'utiliser une DTD :

1. On ne déclare pas de DTD (dans ce cas le fichier est juste "bien formé")
2. On déclare la DTD et on y ajoute les définitions dans le même fichier (*DTD interne*).
On parle dans ce cas d'un XML "standalone" (le fichier XML se suffit à lui-même) :

```
<!DOCTYPE rootelement [  
  [ELEMENT...  
]>
```
3. On déclare la DTD en tant que *DTD "privée"*, la DTD se trouve quelque part dans votre système ou sur Internet (usage répandu pour les DTDs « faites maison »)

```
<!DOCTYPE hello SYSTEM "hello.dtd">
```
4. On déclare une *DTD "public"*, c.a.d. on utilise un nom officiel pour la DTD (usage répandu pour les DTDs connues comme XHTML, SVG, MathML, etc).

Où indiquer la DTD ?

Lieu de la déclaration :

- La DTD est déclarée entre la déclaration de XML et le document lui-même.

```
<?xml version="1.0" standalone="yes"?>  
<!DOCTYPE hello [  
  <ELEMENT hello (#PCDATA)>  
>  
<hello> Hello XML et hello chère lectrice ! </hello>
```
- La déclaration de XML et celle de la DTD font parti du prologue (qui peut contenir d'autres éléments comme les *processing instructions*)
- Attention: l'encodage de la DTD doit correspondre à celui des fichiers XML !

Notion de document XML « correctement formé »

- Le document commence par une *déclaration XML* (l'attribut version est obligatoire)

```
<?xml version="1.0"?>
```
- Il a une *structure hiérarchique*:
 - begin-tags (balises d'ouverture) et end-tags (balises de fermeture) doivent correspondre, c-a-d pas de croisements de type

```
<i>...<b>...</i> .... </b>
```
- Il est *sensible à la « casse »*, donc "Li" n'est pas égal à "li" par exemple.
- Balises de type EMPTY (balises sans balises de fermeture ...). Ces balises sans contenu utilisent la syntaxe XML « auto-fermante », par ex..

```
<br/>
```
- Les *valeurs* d'attributs sont *entre guillemets* ou simples quotes :
(par ex..

```
<a href= " http://tecfu.unige.ch:8080/xml.html " >
```

)
- *Un seul élément racine* (root):
 - L'élément root ne peut apparaître qu'une fois
 - Le root ne doit pas apparaître dans un autre élément (comme

```
<html>
```

)
- *Caractères spéciaux (!)* : <, &, >, ", '
 - Utilisez

```
&lt;
```

,

```
&amp;
```

,

```
&gt;
```

,

```
&quot;
```

,

```
&apos;
```

 à la place de: <, &, >, ", '

Notion de document XML « valide »

Un document est *valide* si :

- Il est *correctement formé* (*well-formed*)
- Il est *associé à une DTD* (ou une autre grammaire)
- Il est *conforme à cette DTD* (ou un autre type de grammaire).

Exemples

Supposons la DTD suivante :

```
<!ELEMENT addressBook (person)+>
<!ELEMENT person (name,email*)>
<!ELEMENT name (family,given)>
<!ELEMENT family (#PCDATA)>
<!ELEMENT given (#PCDATA)>
<!ELEMENT email (#PCDATA)>
```

Exemple de document non valide

Document XML bien-formé mais **non-valide** :

```
<addressBook>
  <address>Derrière le Salève</address>
  <person>
    <name>
      <family>Schneider</family> <firstName>Nina</firstName>
    </name>
    <email>nina@dks.com</email>
  </person>
  <name>
    <family> Muller </family> </name>
</addressBook>
```

Pouvez-vous trouver les 4 erreurs ?

Exemple de document valide

Document XML **valide** :

```
<addressBook>
  <!-- <address>Derrière le Salève</address> -->
  <person>
    <name>
      <family>Schneider</family> <given>Nina</given>
    </name>
    <email>nina@dks.com</email>
  </person>
  <person>
    <name> <family> Muller </family> <given>??</given> </name>
  </person>
</addressBook>
```

(le jeu des 4 erreurs)

II - Analyse de documents XML

- L'analyse (*parsing*) est le processus qui décompose le contenu d'un texte en ses composants individuels.
- L'outil qui réalise l'analyse d'un document XML s'appelle un **parseur XML** ou un **processeur XML**.
- Ceux-ci peuvent être **validant** ou **non-validants**. Dans les deux cas ils doivent informer l'utilisateur quand le document analysé n'est pas correctement formé.

Parseurs XML

Un parseur XML a pour but de charger le contenu d'un document XML en mémoire.

- **SAX**, Simple API for XML, définit une interface pour les parseurs XML. Il a été proposé par les gens de la liste de discussion XML-DEV.
- SAX est une **interface événementielle** : les applications doivent définir ce qu'elles font pour chaque type d'événements qui les intéresse et qui peut être rencontré à la lecture d'un document XML.
- Des événements existent pour le début et la fin du document, pour le début et la fin d'un élément, pour des données caractères, pour des instructions de traitement, etc

Parseur SAX

En Java, SAX est implémenté par une série d'interfaces, la plus importante étant `org.xml.sax.ContentHandler` qui dispose des méthodes `startDocument()`, `startElement()`, `characters()`, `endElement()` et `endDocument()`.

Par exemple, la lecture de :

```
<first>George</first>
```

déclenchera successivement un appel à la méthode `startElement()` puis `characters()` et enfin `endElement()`

Utiliser SAX

Après les `import` de rigueur, votre application doit déclarer implémenter l'interface `ContentHandler` et définir chacune des méthodes de cette interface.

Dans ces méthodes, vous créez généralement des objets java de classes qui sont propres à votre application (classes métiers).

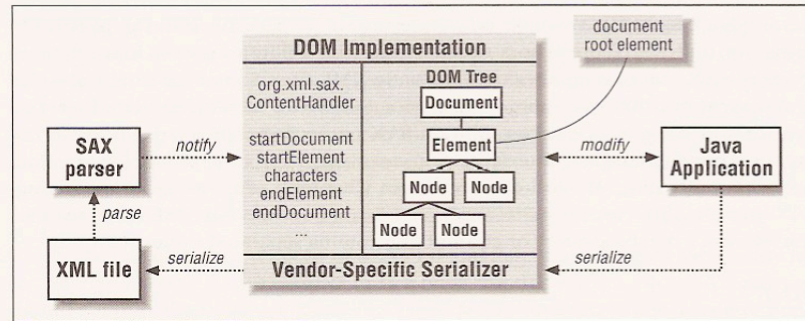
Ainsi, progressivement, l'analyse SAX du document crée une **représentation mémoire** de son contenu sous **forme d'objets**.

Utilisé seul, SAX a l'avantage de permettre de ne charger en mémoire que la partie du document XML qui vous intéresse

DOM : Document Object Model

- DOM est une API qui permet aux programmes de manipuler la structure sous-jacente d'un document XML.
- DOM est une recommandation du W3C exprimée en CORBA IDL, ie indépendamment de toute implémentation.
- La représentation mémoire d'un document XML est souvent appelée arbre DOM, en raison de la forme de sa structure :
 - La racine de cet arbre représente le document XML lui-même (interface `org.w3c.dom.Document`)
 - Les nœuds de l'arbre (interface `org.w3c.dom.Node`) représentent les éléments XML du document (dont l'élément racine du document, par ex `address_book`).
- DOM ne définit pas la façon d'acquérir l'arbre ou de sauver dans un document XML. Dans le cas de Java, on se sert généralement d'un parseur SAX pour obtenir l'arbre.

Interaction de DOM et SAX en Java



Source : Java and XSLT (E.M. Burke, O'Reilly)

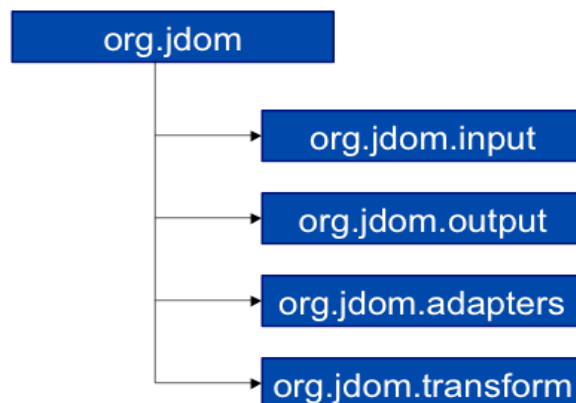
III - JDOM : une API Java-only pour l'analyse de documents XML

www.jdom.org

JDOM est un modèle de programmation pour représenter un document XML. Il est similaire à DOM, mais construit indépendamment.

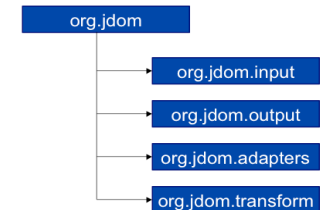
- JDOM est à la fois **centré-Java** et **optimisé pour Java** (maintenance de code). Par exemple, il utilise les **collections** Java, et il a une API naturelle pour les utilisateurs de Java.
- Il fournit un point d'entrée **simple** pour utiliser XML en Java (bien plus que le JAXP de Sun) : il permet d'analyser un document XML, de le convertir en arbre (de façon similaire à DOM), de parcourir/modifier l'arbre, de sauver le document résultant en XML, et tout ça, en peu d'instructions.
- JDOM fournit plutôt des moyens simples de mettre en œuvre les standards SAX et DOM, sans être une couche d'abstraction supplémentaire.
- Il s'intègre facilement avec les implémentations de SAX et DOM déjà présentes dans Java.

Structures des packages JDOM



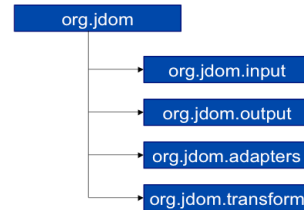
Contenu des classes JDOM

- The **org.jdom** Package
 - Attribute
 - Comment
 - Document
 - EntityRef
 - Text
 - CDATA
 - DocType
 - Element
 - Namespace
 - ProcessingInstruction
- The **org.jdom.transform** Package
 - JDOMSource
 - JDOMResult

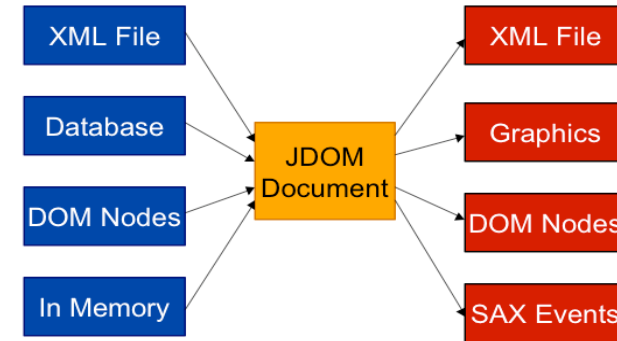


Contenu des classes JDOM

- The `org.jdom.input` Package
 - SAXBuilder
 - DOMBuilder
 - ResultSetBuilder
- The `org.jdom.output` Package
 - XMLOutputter
 - SAXOutputter
 - DOMOutputter
 - JTreeOutputter (widget de GUI)



Utilisation standard de JDOM



Création d'un document JDOM

- Documents are represented by `org.jdom.Document`
- They may be constructed from scratch:

```
Document doc = new Document(new Element("root"));
```

- Or built from a file, stream, system ID, URL:

```
SAXBuilder builder = new SAXBuilder();  
builder.setValidation(true); // attention tps  
builder.setIgnoringElementContentWhitespace(true);  
Document doc = builder.build(url);
```

// ou

```
Document doc = builder.build(new File("/tmp/fic.xml"));
```

Export d'un document JDOM

On déclare d'abord un objet dédié :

```
Document doc = new Document(...); XMLOutputter  
outp = new XMLOutputter();
```

// Raw output

```
outp.output(doc, fileOutputStream);
```

// Compressed output

```
outp.setTextTrim(true);  
outp.output(doc, socket.getOutputStream());
```

// Pretty output

```
outp.setIndent(" "); outp.setNewlines(true);  
outp.output(doc, System.out);
```

Navigation dans l'arbre JDOM

On déclare d'abord un objet dédié :

// Obtenir l'élément racine

```
Element root = doc.getRootElement();
```

// Obtenir la liste des fils

```
List allChildren = root.getChildren();
```

// Obtenir les éléments avec un nom particulier :

```
List namedChildren = root.getChildren("name");
```

// Obtenir le premier élément ayant le nom voulu :

```
Element child = root.getChild("name");
```

(List est la collection java que vous connaissez : itérateur, etc).

Ajout de nœuds dans l'arbre JDOM

```
List allChildren = root.getChildren();
```

// Remove the fourth child

```
allChildren.remove(3);
```

// Remove children named "jack"

```
allChildren.removeAll(root.getChildren("jack"));
```

```
root.removeChildren("jack"); // ou aussi
```

// Add a new child

```
allChildren.add(new Element("jane"));
```

```
allChildren.add(0, new Element("first"));
```

// Moving elements

```
Element elem = new Element("movable");  
parent1.addContent(elem); // place geulque part  
parent1.removeContent(elem); // le décroche  
parent2.addContent(elem); // l'accroche ailleurs  
parent3.addContent(autreElem.detach()); //encore mieux !
```

Note: ces méthodes vérifie au fur et à mesure des modifications de la structure de l'arbre, que celui-ci reste bien ...un arbre (1 racine, pas de cycles, etc).

Accéder aux attributs des balises

```
<table width="100%" border="0"> </table>
```

• // Obtenir la valeur d'un attribut :

```
String width = table.getAttributeValue("width");  
int border = table.getAttribute("width").getIntValue();
```

• // Positionner un attribut :

```
table.setAttribute("vspace", "0");
```

• // Enlever un attribut ou tous :

```
table.removeAttribute("vspace");  
table.getAttributes().clear();
```

Contenu d'un élément

```
<description> A cool demo </description>
```

// The text is directly available: returns "\n A cool demo\n"
String desc = element.getText();

// There's a convenient shortcut: returns "A cool demo"
String desc = element.getTextTrim();

// Text content can be changed directly
element.setText("A new description");

Contenus de natures différentes

- Elements may contain many things

```
<table>
  <!-- Some comment -->
  Some text
  <tr>Some child element</tr>
</table>
```

Obtention de l'élément (objet java) correspondant à la balise `tr` située à l'intérieur de l'élément `table` :

```
Element tr = table.getChild("tr");
```

Contenus de natures différentes

```
<table>
  <!-- Some comment -->
  Some text
  <tr>Some child element</tr>
</table>
```

- On a aussi accès à ce qui est en dehors des balises :

```
List mixedCo = table.getContent();
Iterator itr = mixedCo.iterator();
while (itr.hasNext()) {
  Object o = i.next();
  if (o instanceof Comment) { ... }
}
```

On peut aussi tester l'appartenance aux classes *Element*, *Text*, *CDATA*, *ProcessingInstruction* et *EntityRef*

```
// Enlève le commentaire détecté : pourquoi 1 ci-dessous ?
mixedCo.remove(1);
```

Merci

- Sources Web :

- Site <XML> fr
- <http://cynober.developpez.com/tutoriel/java/xml/jdom/>
- Daniel.Schneider@tecfa.unige.ch

- Livres :

- Java and XSLT, E.M. Burke, O'Reilly, 2001.
- Développer en XML avec Java 2, M.C. Daconta & A. Saganich, CampusPress, 2001.
- 8 solutions contrôlées avec XML et Java, B. Marchal, Dunod, 2001.