

Langages Avancés pour les Bases de Données

Marius Bilasco
Bureau 336 - M3.ext
marius.bilasco@univ-lille1.fr

Master Info M1 2017-2018

Évaluation

Trois notes sont attribuées à chaque étudiant durant le semestre :

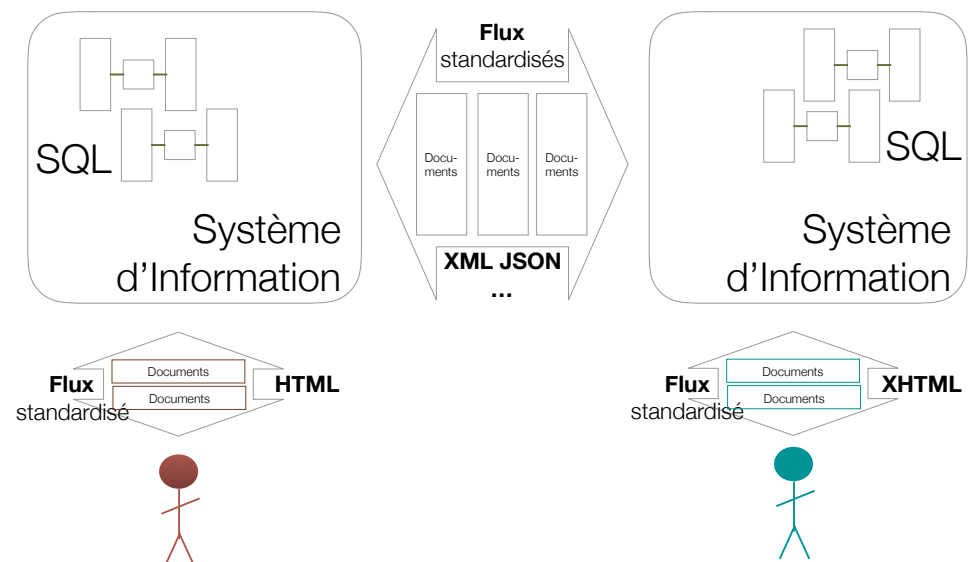
- Une note **INT** sur 20 d'interrogation écrite en amphi en milieu de semestre
- Une note **TP** sur 20 synthétisant l'ensemble des travaux pratiques rendus pendant le semestre : tout **TP** non rendu **-2** note **TP**
- Une note **EX** sur 20 de contrôle écrit terminal

$$\text{LABD} = \frac{\text{TP} + 3 \times \sup\left(\text{EX}, \frac{2 \times \text{EX} + \text{INT}}{3}\right)}{4}$$

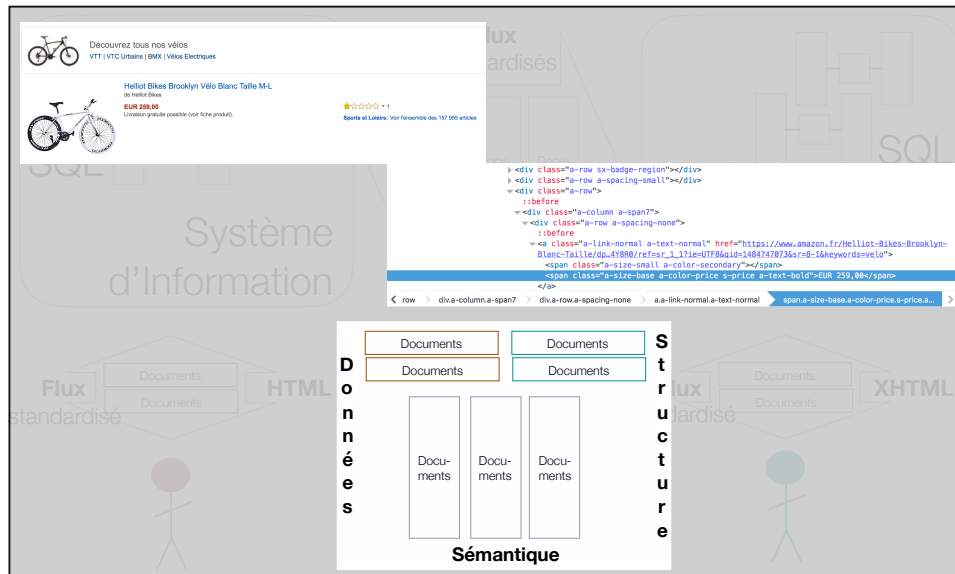
Format pédagogique

- Equipe pédagogique : **Marius Bilasco** et **Yves Roos**
- Cours-TD hebdomadaire : 1h30
- TD sur machine hebdomadaire : 2h

Bases de données, Systèmes d'Informations et Documents



Bases de données, Systèmes d'Informations et Documents

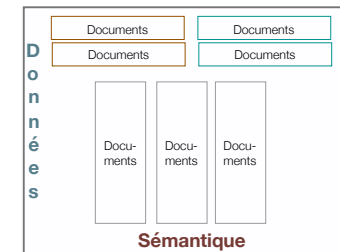


Langages Avancés pour les Bases de Données

Définir, Typier
Naviguer, Parcourir
Interroger
Transformer

DTD XML Schema
XPath 1.0/2.0
XQuery
XSLT

1. Technologies XML



Décrire, Définir
Raisoner

RDF / RDFS
SPARQL

2. Web Sémantique

XML est *human readable*

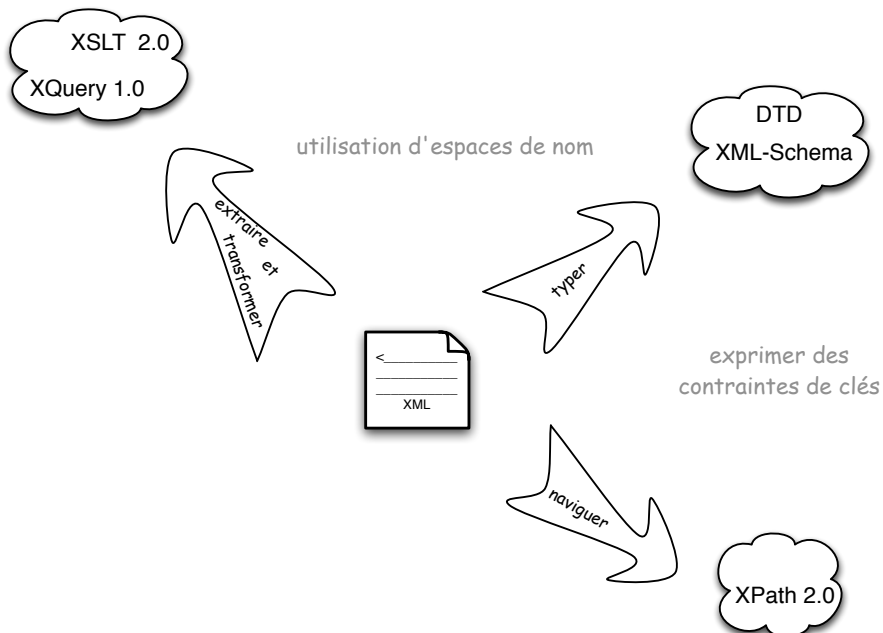
```
<?xml version="1.0" encoding="UTF-8"?>
<library>
  <items>
    <bd uid="8B5F233B-0B45-4907-A106-AA72840F639B">
      <illustrator>Roba</illustrator>
      <numberInSeries>5</numberInSeries>
      <pages>60</pages>
      <publisher>Dupuis</publisher>
      <series>Boule et Bill -1</series>
      <title>60 gags de Boule et Bill n°5</title>
    </bd>
    <bd uid="28F20F96-C938-41BA-96AB-BAFADC22D80E">
      <illustrator>Mezières</illustrator>
      <numberInSeries>12</numberInSeries>
      <pages>48</pages>
      <publisher>Dargaud</publisher>
      <series>Valérien</series>
      <title>Les foudres d'Hypsis</title>
    </bd>
    <bd uid="DCA1F83F-ADA9-45BD-AB19-EA991120DF53">
      <illustrator>Magniaux</illustrator>
      <pages>48</pages>
      <publisher>Editions Williams</publisher>
      <series>Charlot (Williams)</series>
      <title>La ruée vers l'or</title>
    </bd>
  </items>
</library>
```

LABD

1. Technologies XML

XML est *human readable*

```
<?xml version="1.0" encoding="UTF-8"?> <library> <items> <book uuid="8B5F233B-0B45-4907-A106-AA72840F639B"
boxWidthInInches="40E-1" boxHeightInInches="1.17E1" used="0" boxLengthInInches="8.27E0" illustrator="Roba"
fullTitle="60 gags de Boule et Bill n°5" numberInSeries="5" rating="00E0" published="1969-01-01T11:00:00"
series="Boule et Bill -1-" location="BouleEtBill5w_30012005.jpg" hasExperienced="0" minutes="0" listened="0"
boxWeightInPounds="00E0" mediaccount="1" signed="0" read="0" country="fr" watched="0" netrating="00E0" rare="0"
title="60 gags de Boule et Bill n°5" pages="60" edition="Dupuis" upc="0000000008020" played="0" players="0"
lastLookupTime="286237024"></book> <book uuid="D1DDD65E-7E7C-4E51-9B76-5BFCAB88EEFB"
boxWidthInInches="40E-1"
boxHeightInInches="1.17E1" used="0" boxLengthInInches="8.27E0" illustrator="Collectif" fullTitle="Nous,
Tintin" numberInSeries="0" rating="00E0" published="1987-01-01T11:00:00" series="Tintin - Divers"
location="NousTintin.jpg" hasExperienced="0" minutes="0" listened="0" boxWeightInPounds="00E0" mediaccount="1"
signed="0" read="0" country="fr" watched="0" netrating="00E0" rare="0" title="Nous, Tintin" pages="0"
edition="Les éditions du Lion" upc="0000000007566" played="0" players="0" lastLookupTime="286236999"></book>
<book uuid="28F20F96-C938-41BA-96AB-BAFADC22D80E" boxWidthInInches="40E-1" boxHeightInInches="1.17E1"
used="0"
boxLengthInInches="8.27E0" illustrator="Mezières" fullTitle="Les foudres d'Hypsis" numberInSeries="12"
rating="00E0" published="1985-01-10T11:00:00" series="Valérien" location="valeriancouv12.jpg"
hasExperienced="0" minutes="0" listened="0" boxWeightInPounds="00E0" mediaccount="1" signed="0" read="0"
country="fr" watched="0" netrating="00E0" rare="0" title="Les foudres d'Hypsis" pages="48" edition="Dargaud"
upc="9782205030327" played="0" players="0" lastLookupTime="286236581"></book> <book
uuid="DCA1F83F-ADA9-45BD-AB19-EA991120DF53" boxWidthInInches="40E-1" boxHeightInInches="1.17E1" used="0"
boxLengthInInches="8.27E0" illustrator="Magniaux" fullTitle="La ruée vers l'or" numberInSeries="2"
rating="00E0" published="1974-01-04T11:00:00" series="Charlot (Williams)"
location="charlotwilliamsruéeverslor.jpg" hasExperienced="0" minutes="0" listened="0" boxWeightInPounds="00E0"
mediaccount="1" signed="0" read="0" country="fr" watched="0" netrating="00E0" rare="0" title="La ruée vers
l'or" pages="48" edition="Editions Williams" upc="0000000007795" played="0" players="0"
lastLookupTime="286237014"></book>
```



XML est *human readable*

```
<?xml version="1.0" encoding="UTF-8"?>
<library>
  <items>
    <bd uuid="8B5F233B-0B45-4907-A106-AA72840F639B">
      <illustrator>Roba</illustrator>
      <numberInSeries>5</numberInSeries>
      <pages>60</pages>
      <publisher>Dupuis</publisher>
      <series>Boule et Bill -1-</series>
      <title>60 gags de Boule et Bill n°5</title>
    </bd>
    <bd uuid="28F20F96-C938-41BA-96AB-BAFADC22D80E">
      <illustrator>Mezières</illustrator>
      <numberInSeries>12</numberInSeries>
      <pages>48</pages>
      <publisher>Dargaud</publisher>
      <series>Valérien</series>
      <title>Les foudres d'Hypsis</title>
    </bd>
    <bd uuid="DCA1F83F-ADA9-45BD-AB19-EA991120DF53">
      <illustrator>Magniaux</illustrator>
      <pages>48</pages>
      <publisher>Editions Williams</publisher>
      <series>Charlot (Williams)</series>
      <title>La ruée vers l'or</title>
    </bd>
```

LABD

Master Info M1 2014-2015

1. Technologies XML

Cours 1 : XML et DTD

Les principes de base de XML

- **XML** permet de décrire des documents **structurés hiérarchiquement**, utilisant des **balises**.
- XML = e**X**tensible **M**arkup **L**anguage. eXtensible signifie qu'on définit soit même le langage des balises.
- **Format texte** : facile à modifier, à échanger. Est devenu de fait un des **principaux formats d'échange** entre applications et sur le web.

Document bien formé et document valide

- **Bien formé** = suit les règles syntaxiques de XML
 - Bon **parenthésage** des balises ouvrantes et fermantes
 - Un élément **racine** contient tous les autres (on parle d'arbre d'éléments)
- **Valide** = bien formé + conforme à un schéma (DTD, XML-Schema,...)
 - La validité n'est pas requise
 - Définir un schéma permet de manipuler plus facilement des documents, de les traiter par des programmes
 - Il existe plusieurs langages qui permettent de définir des schémas ou types de documents : Document Type Definition (**DTD**), **XML-schema**, **Relax-NG**

Historique

- **1979-1986** : **SGML**, description de documents techniques
- **1991** : **HTML**, inventé pour le web
- **1996** : création d'un groupe de travail du **W3C** dont les objectifs sont de définir un langage plus facile que SGML et plus générique que HTML i.e. qui permet de définir plusieurs familles de langages de balises.
- **1998** : **XML 1.0**. Version simplifiée de SGML et plus adaptée au Web (e.g. support natif des différents codages internationaux).

Exemple

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- exemple de fichier xml -->
<lettre>
  <en-tete date="10 janvier 2018">
    <expediteur>
      marius.bilasco@univ-lille1.fr
    </expediteur>
    <destinataire>
      romain.rouvoy@univ-lille1.fr
    </destinataire>
    <objet>LABD</objet>
  </en-tete>
  <salutation>Romain</salutation>
  <corps>
    <para>
      au fait, peux-tu me rappeler quand commence LABD ?
    </para>
    <para>Merci.</para>
  </corps>
  <signature>Marius</signature>
</lettre>
```

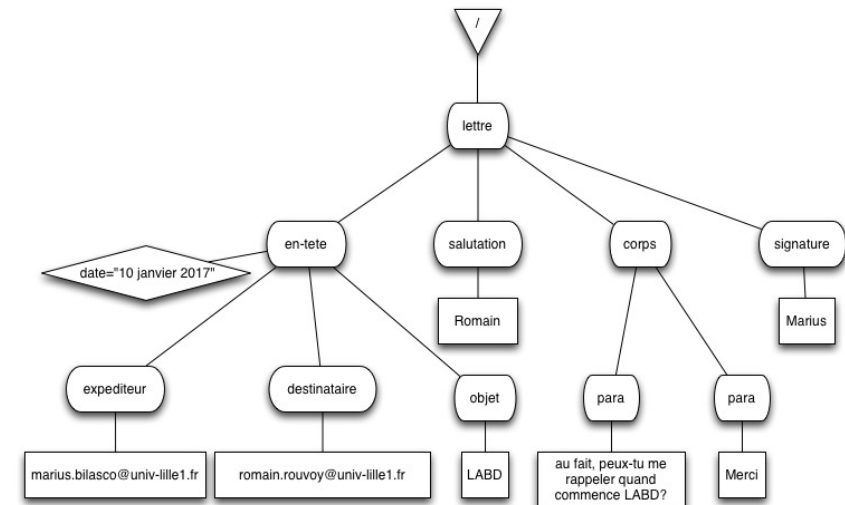
Structure d'un document XML

- un **prologue** (facultatif mais conseillé) déclaration de type de document
`<?xml version="1.0" encoding="UTF-8" ?>`
Par défaut, le codage des caractères est `utf-8`.
- un **arbre d'éléments**. C'est le contenu du document. Les éléments peuvent disposer d'**attributs**.
- des **commentaires** : `<!-- commentaire ... -->`
- des **instructions de traitement** qui peuvent apparaître dans le prologue et dans l'arbre d'éléments. Les instructions de traitement permettent aux documents de contenir des instructions destinées aux applications.
`<?php . . . ?>`
`<?xml-stylesheet href="courrier.css" type="text/css" ?>`

Les éléments

- élément** = **balise d'ouverture** + contenu + **balise de clôture**
`<destinataire>romain.rouvoy@univ-lille1.fr</destinataire>`
- cas particulier : un **élément vide** contient ces trois choses en une seule balise.
`<eltVide/>`
- nom** des éléments :
 - il est sensible à la casse : `<Hello>` ≠ `<hello>`
 - il peut comporter des lettres, des chiffres, des caractères `- _ . :` et il ne peut commencer par un chiffre ou par les caractères `-` et `.` Le caractère `:` ne devrait être utilisé que pour séparer les espaces de noms. Les noms commençant par `xml` sont réservés.

Arbre d'éléments



Les éléments : attributs

- Dans la **balise d'ouverture** d'un élément, on peut définir des **attributs** qui définissent des **propriétés** de l'élément.
`<rapport langue="FR" date-modif="2017-01-07">`
- la valeur d'un attribut est une **chaîne**
`date-modif="2017-01-07" ou date-modif='2017-01-07'`
- les attributs **ne sont pas ordonnés**
`<rapport langue="FR" date-modif="2016-01-07">`
`<rapport date-modif="2016-01-07" langue="FR">`
- pour un élément donné, **chaque nom d'attribut est unique**
`<rapport langue="FR" langue="EN" >`

Conventions de nommage

Pas réellement de convention...mais il est conseillé

- d'utiliser des **minuscules** pour les noms d'élément et d'attribut
- d'**éviter les accents** dans les noms d'élément et d'attribut
- pour les noms composés, d'utiliser comme séparateurs -, _, . ou des **majuscules** (à la java)
- de préférer les guillemets " pour délimiter les valeurs des attributs

Les éléments : contenu

- **section littérale** :

```
<exemple>
  <![CDATA[<auteur>Dupond & al.</auteur>]]
</exemple>
```

Une section CDATA peut contenir n'importe quelle chaîne sauf]]

Les éléments : contenu

- Un élément peut contenir d'**autres éléments**, des **données**, des **instructions de traitement**,...
- Une donnée est un flot de caractères qui **ne contient pas** les caractères <, > et &
- On peut mettre dans la donnée des **entités** prédéfinies :

| Entité | Caractère |
|--------|-----------|
| < | < |
| > | > |
| & | & |
| " | " |
| ' | ' |

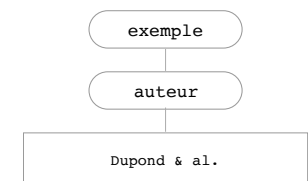
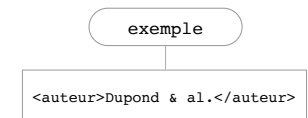
Les éléments : contenu

- **section littérale** :

```
<exemple>
  <![CDATA[<auteur>Dupond & al.</auteur>]]
</exemple>
```

vs

```
<exemple>
  <auteur>Dupond & al.</auteur>
</exemple>
```



Élément ou attribut ?

Comment choisir entre élément ou attribut :

```
<poids unite="mg" valeur="175"/>
```

```
<poids unite="mg">175</poids>
```

```
<poids>
  <unite>mg</unite>
  <valeur>175</valeur>
</poids>
```

```
<poids>
  <unite>mg</unite>
  175
</poids>
```

Élément ou attribut ?

```
<etudiant nom="Dupond"
           prenom="Mathieu"
           formation="Master INFO"
           annee="M1"/>
```

```
<etudiant>
  <nom>Dupond</nom>
  <prenom>Mathieu</prenom>
  <formation>Master INFO</formation>
  <annee>M1</annee>
</etudiant>
```

```
<etudiant>
  <nom>Dupond</nom>
  <prenom>Mathieu</prenom>
  <formation annee="M1">Master INFO</formation>
</etudiant>
```

Élément ou attribut ?

pas de règle absolue mais...

- **élément** = contenu complexe
- les éléments sont **ordonnés**, les attributs non
- attribut = **valeur simple**
- attribut = **assez indépendant** du contenu ou même **méta-donnée**

```
<document lang="FR" encoding="UTF8">...
```

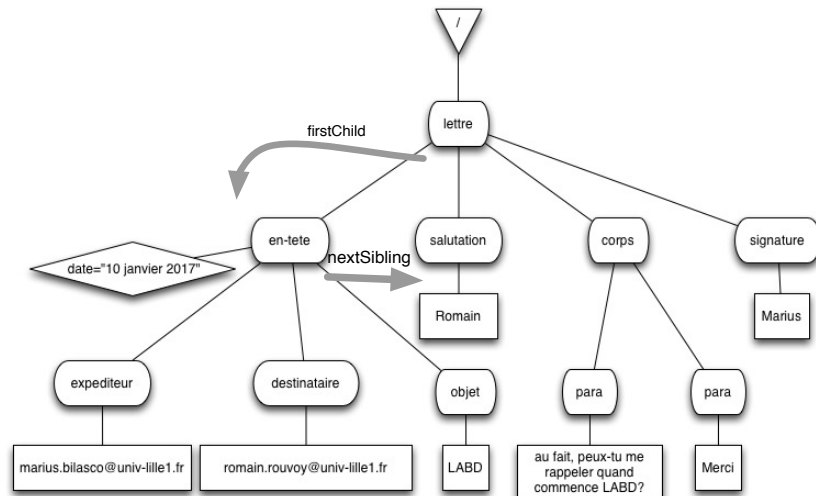
```
<poids unite="mg">...
```

Lecture de fichier XML : APIs java

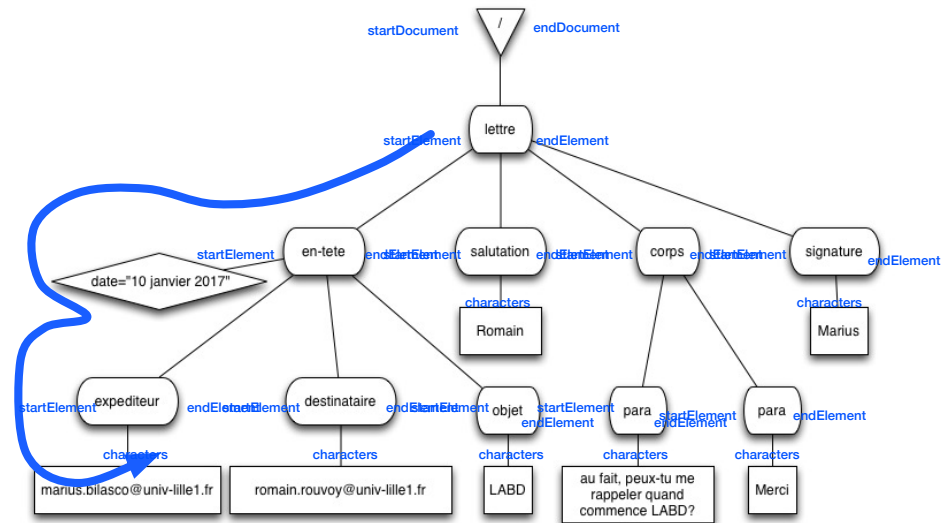
- **DOM**

- **SAX**

Document Object Model



Simple Api for XML



Simple Api for XML

[org.w3c.dom.bootstrap](#)
[org.w3c.dom.events](#)
[org.w3c.dom.ls](#)
[org.xml.sax](#)
[org.xml.sax.ext](#)
[org.xml.sax.helpers](#)

[org.xml.sax](#)
Interfaces
[AttributeList](#)
[Attributes](#)
[ContentHandler](#)
[DocumentHandler](#)
[DTDHandler](#)
[EntityResolver](#)
[ErrorHandler](#)
[Locator](#)
[Parser](#)
[XMLFilter](#)
[XMLReader](#)
Classes
[HandlerBase](#)
[InputSource](#)
Exceptions
[SAXException](#)
[SAXNotRecognizedException](#)
[SAXNotSupportedException](#)
[SAXParseException](#)

Method Summary

| | | |
|------|--|---|
| void | characters (char[] ch, int start, int length) | Receive notification of character data. |
| void | endDocument () | Receive notification of the end of a document. |
| void | endElement (String uri, String localName, String qName) | Receive notification of the end of an element. |
| void | endPrefixMapping (String prefix) | End the scope of a prefix-URI mapping. |
| void | ignorableWhitespace (char[] ch, int start, int length) | Receive notification of ignorable whitespace in element content. |
| void | processingInstruction (String target, String data) | Receive notification of a processing instruction. |
| void | setDocumentLocator (Locator locator) | Receive an object for locating the origin of SAX document events. |
| void | skippedEntity (String name) | Receive notification of a skipped entity. |
| void | startDocument () | Receive notification of the beginning of a document. |
| void | startElement (String uri, String localName, String qName, Attributes atts) | Receive notification of the beginning of an element. |
| void | startPrefixMapping (String prefix, String uri) | Begin the scope of a prefix-URI Namespace mapping. |

Simple Api for XML

```
package labd ;

import org.xml.sax.*;
import org.xml.sax.helpers.* ;
import java.io.IOException;

/**
 * @author yves.roos
 *
 * Exemple d'implementation d'un ContentHandler.
 */
public class LABDHandler extends DefaultHandler {

    /**
     * Evenement envoye au demarrage du parse du flux xml.
     * @throws SAXException en cas de probleme quelconque ne permettant pas de
     * se lancer dans l'analyse du document.
     * @see org.xml.sax.ContentHandler#startDocument()
     */
    public void startDocument() throws SAXException {
        System.out.println("Debut du document");
    }
}
```


Simple Api for XML

```
/**
 * Evenement reçu a chaque fois que l'analyseur rencontre une balise xml ouvrante.
 * @param namespaceURI l'url de l'espace de nommage.
 * @param localName le nom local de la balise.
 * @param rawName nom de la balise en version 1.0 <code>namespaceURI + ":" + localName</code>
 * @throws SAXException si la balise ne correspond pas a ce qui est attendu,
 * comme par exemple non respect d'une dtd.
 */
public void startElement(String namespaceURI, String localName, String rawName, Attributes attributs)
throws SAXException {
    System.out.println("Ouverture de la balise : " + localName);
    if (attributs.getLength() != 0) System.out.println(" Attributs de la balise : ");
    for (int index = 0; index < attributs.getLength(); index++) { // on parcourt la liste des attributs
        System.out.println("    - " + attributs.getLocalName(index) + " = " + attributs.getValue(index));
    }
}
```

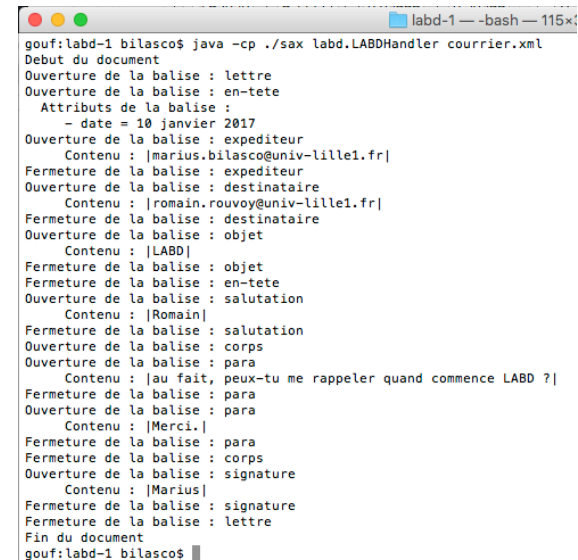
Simple Api for XML : exemple

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- exemple de fichier xml -->
<lettre>
  <en-tete date="10 janvier 2017">
    <expediteur>
      marius.bilasco@univ-lille1.fr
    </expediteur>
    <destinataire>
      romain.rouvoy@univ-lille1.fr
    </destinataire>
    <objet>LABD</objet>
  </en-tete>
  <salutation>Romain</salutation>
  <corps>
    <para>
      au fait, peux-tu me rappeler quand commence LABD ?
    </para>
    <para>Merci.</para>
  </corps>
  <signature>Yves</signature>
</lettre>
```

Simple Api for XML

```
public static void main(String[] args) {
    try {
        XMLReader saxReader = XMLReaderFactory.createXMLReader();
        saxReader.setContentHandler(new LABDHandler());
        saxReader.parse(args[0]);
    } catch (Exception t) {
        t.printStackTrace();
    }
}
```

Simple Api for XML : exemple



```
gouf:labd-1 bilasco$ java -cp ./sax labd.LABDHandler courrier.xml
Debut du document
Ouverture de la balise : lettre
Ouverture de la balise : en-tete
  Attributs de la balise :
    - date = 10 janvier 2017
Ouverture de la balise : expediteur
  Contenu : |marius.bilasco@univ-lille1.fr|
Fermeture de la balise : expediteur
Ouverture de la balise : destinataire
  Contenu : |romain.rouvoy@univ-lille1.fr|
Fermeture de la balise : destinataire
Ouverture de la balise : objet
  Contenu : |LABD|
Fermeture de la balise : objet
Fermeture de la balise : en-tete
Ouverture de la balise : salutation
  Contenu : |Romain|
Fermeture de la balise : salutation
Ouverture de la balise : corps
Ouverture de la balise : para
  Contenu : |au fait, peux-tu me rappeler quand commence LABD ?|
Fermeture de la balise : para
Ouverture de la balise : para
  Contenu : |Merci.|
Fermeture de la balise : para
Fermeture de la balise : corps
Ouverture de la balise : signature
  Contenu : |Marius|
Fermeture de la balise : signature
Fermeture de la balise : lettre
Fin du document
gouf:labd-1 bilasco$
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- exemple de fichier xml -->
<lettre>
  <en-tete date="10 janvier 2017">
    <expediteur>
      marius.bilasco@univ-lille1.fr
    </expediteur>
    <destinataire>
      romain.rouvoy@univ-lille1.fr
    </destinataire>
    <objet>LABD</objet>
  </en-tete>
  <salutation>Romain</salutation>
  <corps>
    <para>
      au fait, peux-tu me rappeler quand ...
    </para>
    <para>Merci.</para>
  </corps>
  <signature>Yves</signature>
</lettre>
```

Simple Api for XML : exercice

```
...
public class InterroHandler extends DefaultHandler {
    private int n ;
    private boolean fermante ;

    public void startDocument() {this.n = 0 ; this.fermante = false ;}

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

    public void startElement(String namespaceURI,...){
        if (this.fermante) System.out.print(" , " ) ;
        System.out.print(localName + "-" + this.n + "( " ) ;
        this.n++ ; this.fermante = false ;
    }

    public void endElement(String namespaceURI, ...){
        System.out.print(" )") ;
        this.n++ ; this.fermante = true ;
    }
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<a><b/><c><d/><e><f/></e></c><g/></a>
```

Comment lier une DTD à un document XML

Une DTD peut être associée de 3 façons à un document XML :

- 1.**DTD interne** : toutes les règles sont dans le fichier XML.
- 2.**DTD mixte** : certaines règles sont décrites dans un fichier spécifique et certaines règles sont dans le fichier XML.
- 3.**DTD externe** : toutes les règles à respecter sont décrites dans un fichier spécifique.

Typage avec des DTD

DTD

= **grammaire** pour la structure des documents

= un ensemble de **règles**,

chacune d'entre-elles décrivant le **contenu autorisé** d'un élément ou l'ensemble des attributs existant pour un élément.

DTD externe

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE bonjour SYSTEM "bonjour.dtd">
<bonjour>Hello world!</bonjour>
```

DTD externe

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE bonjour SYSTEM
  "http://www.chez-moi.fr/dtd/bonjour.dtd">
<bonjour>Hello world!</bonjour>
```

DTD interne

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE bonjour
  [ <!ELEMENT bonjour (#PCDATA)> ]>
<bonjour>bonjour tout le monde</bonjour>
```

DTD externe

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11-strict.dtd">
<html>
  <head>
```

...

DTD mixte

```
<?xml version="1.0"?>
<!DOCTYPE bonjour SYSTEM "bonjour.dtd"
  [<!ELEMENT bonjour (#PCDATA)>]>
<bonjour>bonjour tout le monde</bonjour>
```

Validation d'un fichier XML

A partir du moment où une DTD est associée au document à valider, on peut **valider** à l'aide :

- d'un **logiciel spécialisé** dans le traitement des documents XML (*XMLSpy, Editix, Eclipse, . . .*)
- par **programme** en utilisant les bibliothèques de traitement de XML disponibles dans beaucoup de langages (*java, php, perl, ...*)

Structure d'une DTD

Une DTD contient :

- des déclarations d'**éléments**,
- des déclarations d'**attributs**,
- des déclarations d'**entités**,
- des **commentaires**

<!-- comme dans les documents XML -->

Le fichier `XMLParser.java`

```
package labd ;

import org.xml.sax.XMLReader ;
import org.xml.sax.helpers.XMLReaderFactory ;

/**
 * Analyseur XML pour une validation par rapport a une DTD.
 * La validation se fait à la volée, en lisant le document.
 * C'est un analyseur SAX -> On ne construit pas l'arbre DOM du document.
 */
public class XMLParser {

    /**
     * Methode de validation : Executer "java XMLParser leDocumentAValider.xml"
     * On vérifie que le document est conforme a la DTD qui lui est liée
     *
     * @param args      ligne de commande = le nom du fichier XML à valider
     * @exception Exception Si probleme lors de la creation des objets.
     */
    public static void main(String[] args) {
        try {
            XMLReader saxReader = XMLReaderFactory.createXMLReader(); //comme avant
            saxReader.setFeature("http://xml.org/sax/features/validation", true); // c'est la nouveauté
            //saxReader.setContentHandler(new MonHandlerAMoi()); // si on veut
            saxReader.parse(args[0]);
        } catch (Exception t) {
            t.printStackTrace();
        }
    }
}
```

Déclaration d'élément

<!ELEMENT nom modèle>

- **ELEMENT** (en **majuscule**) est un mot clef,
- **nom** est un nom **valide** d'élément,
- **modèle** est le **modèle de contenu** de l'élément.
 - vide** l'élément n'a pas de contenu (mais peut avoir des attributs)
 - libre** le contenu de l'élément est un contenu quelconque bien formé
 - données** l'élément contient du texte
 - éléments** l'élément est composé d'autre éléments (ses fils)
 - mixte** l'élément contient un mélange de texte et de sous-éléments

Déclaration d'élément

`<!ELEMENT nom EMPTY>`

- `ELEMENT` (en majuscule) est un mot clef,
- `nom` est un nom valide d'élément,
- `modèle` est le **modèle de contenu** de l'élément.
 - vide** l'élément n'a pas de contenu (mais peut avoir des attributs)
 - libre** le contenu de l'élément est un contenu quelconque bien formé
 - données** l'élément contient du texte
 - éléments** l'élément est composé d'autre éléments (ses fils)
 - mixte** l'élément contient un mélange de texte et de sous-éléments

Déclaration d'élément

`<!ELEMENT nom (#PCDATA)>`

- `ELEMENT` (en majuscule) est un mot clef,
- `nom` est un nom valide d'élément,
- `modèle` est le **modèle de contenu** de l'élément.
 - vide** l'élément n'a pas de contenu (mais peut avoir des attributs)
 - libre** le contenu de l'élément est un contenu quelconque bien formé
 - données** l'élément contient du texte
 - éléments** l'élément est composé d'autre éléments (ses fils)
 - mixte** l'élément contient un mélange de texte et de sous-éléments

Déclaration d'élément

`<!ELEMENT nom ANY>`

- `ELEMENT` (en majuscule) est un mot clef,
- `nom` est un nom valide d'élément,
- `modèle` est le **modèle de contenu** de l'élément.
 - vide** l'élément n'a pas de contenu (mais peut avoir des attributs)
 - libre** le contenu de l'élément est un contenu quelconque bien formé
 - données** l'élément contient du texte
 - éléments** l'élément est composé d'autre éléments (ses fils)
 - mixte** l'élément contient un mélange de texte et de sous-éléments

Déclaration d'élément

`<!ELEMENT nom modèle>`

- `ELEMENT` (en **majuscule**) est un mot clef,
- `nom` est un nom **valide** d'élément,
- `modèle` est le **modèle de contenu** de l'élément.
 - vide** l'élément n'a pas de contenu (mais peut avoir des attributs)
 - libre** le contenu de l'élément est un contenu quelconque bien formé
 - données** l'élément contient du texte
 - éléments** l'élément est composé d'autre éléments (ses fils)
 - mixte** l'élément contient un mélange de texte et de sous-éléments

Modèle de contenu d'élément

On définit le contenu à l'aide d'une [expression régulière](#) de sous-éléments :

- [séquence](#)

```
<!ELEMENT chapitre (titre,intro,section)>
```

- [choix](#)

```
<!ELEMENT chapitre (titre,intro,(section|sections))>
```

- [indicateurs d'occurrence](#) * (0-n) + (1-n) ? (0-1)

```
<!ELEMENT chapitre (titre,intro?,section+)>
```

```
<!ELEMENT section (titre-section,texte-section)+>
```

```
<!ELEMENT texte-section (p|f)*>
```

Contenu mixte

Une seule façon de mélanger texte `#PCDATA` et des sous-éléments est acceptée : `#PCDATA` doit être le premier membre d'un choix placé sous une étoile.

```
<!ELEMENT p (#PCDATA | em | exposant | indice | renvoi ) *>
```

Modèle de contenu d'élément

La syntaxe précise des expressions régulières de sous-éléments est :

- `cp ::= (Name | choice | seq) ('?' | '*' | '+')?`

- `seq ::= '(' cp (',' cp)* ')'`

- `choice ::= '(' cp ('|' cp)+ ')'`

Exemple

```
<!ELEMENT catalogue ( stage )*>
<!ELEMENT stage ( intitule , prerequis ?)>
<!ELEMENT intitule(#PCDATA)>
<!ELEMENT prerequis (#PCDATA | xref )*>
<!ELEMENT xref EMPTY>
```

Exemple

```
<catalogue>
  <stage>
    <intitule>XML et les bases de données</intitule>
    <prerequis>
      connaitre les langages SQL et HTML
    </prerequis>
  </stage>
  <stage>
    <intitule>XML programmation</intitule>
    <prerequis>
      avoir suivi le stage de XML et les bases de données
    </prerequis>
  </stage>
</catalogue>
```

```
<!ELEMENT catalogue ( stage )*>
<!ELEMENT stage ( intitule , prerequis ?)>
<!ELEMENT intitule (#PCDATA)>
<!ELEMENT prerequis (#PCDATA | xref )*>
<!ELEMENT xref EMPTY>
```

Exemple

```
<catalogue>
  <stage>
    <intitule>XML et les bases de données</intitule>
    <prerequis>
      connaitre les <xref/> et <xref/>
    </prerequis>
  </stage>
  <stage>
    <intitule>XML programmation</intitule>
    <prerequis>
      avoir suivi le stage de XML et les bases de données
    </prerequis>
  </stage>
</catalogue>
```

```
<!ELEMENT catalogue ( stage )*>
<!ELEMENT stage ( intitule , prerequis ?)>
<!ELEMENT intitule (#PCDATA)>
<!ELEMENT prerequis (#PCDATA | xref )*>
<!ELEMENT xref EMPTY>
```

Exemple

```
<catalogue>
  <stage>
    <intitule>XML et les bases de données</intitule>
  </stage>
  <stage>
    <intitule>XML programmation</intitule>
    <prerequis>
      avoir suivi le stage de XML et les bases de données
    </prerequis>
  </stage>
</catalogue>
```

```
<!ELEMENT catalogue ( stage )*>
<!ELEMENT stage ( intitule , prerequis ?)>
<!ELEMENT intitule (#PCDATA)>
<!ELEMENT prerequis (#PCDATA | xref )*>
<!ELEMENT xref EMPTY>
```

Exemple

```
<catalogue>
  <stage>
    <intitule>XML et les bases de données</intitule>
    <prerequis>
      connaitre les <xref> langages SQL et HTML </xref>
    </prerequis>
  </stage>
  <stage>
    <intitule>XML programmation</intitule>
    <prerequis>
      avoir suivi le stage de XML et les bases de données
    </prerequis>
  </stage>
</catalogue>
```

```
<!ELEMENT catalogue ( stage )*>
<!ELEMENT stage ( intitule , prerequis ?)>
<!ELEMENT intitule (#PCDATA)>
<!ELEMENT prerequis (#PCDATA | xref )*>
<!ELEMENT xref EMPTY>
```

Exemple

```
<!ELEMENT catalogue ( stage )*>
<!ELEMENT stage ( intitule , prerequis ? )>
<!ELEMENT intitule (#PCDATA)>
<!ELEMENT prerequis (#PCDATA | xref )*>
<!ELEMENT xref EMPTY>
```

```
<catalogue>
</catalogue>
```

Déclarations d'attributs

```
<!ATTLIST element nom-attribut1 type1 default1
                  nom-attribut2 type2 default2
                  ...>
```

La déclaration par défaut peut prendre quatre formes :

- la valeur par défaut de l'attribut,
- **#REQUIRED** indique que l'attribut est obligatoire,
- **#IMPLIED** indique que l'attribut est optionnel,
- **#FIXED valeur** indique que l'attribut prend toujours la même valeur, dans toute instance de l'élément si l'attribut y apparaît.

Déclarations d'attributs

```
<!ATTLIST element nom-attribut1 type1 default1
                  nom-attribut2 type2 default2
                  ...>
```

Le type d'un attribut définit les valeurs qu'il peut prendre

- **CDATA** : valeur chaîne de caractères,
- **ID**, **IDREF**, **IDREFS** permettent de définir des références à l'intérieur du document,
- Une **liste de choix** possibles parmi un ensemble de noms symboliques.

Exemples de déclarations d'attributs

```
<!ATTLIST document version CDATA "1.0">
```

```
<document version="1.0" >
...
</document>
```

OK

Exemples de déclarations d'attributs

```
<!ATTLIST document version CDATA "1.0">
```

```
<document version="2.0" >  
...  
</document>
```

OK

Exemples de déclarations d'attributs

```
<!ATTLIST document version CDATA "1.0">
```

```
<document>  
...  
</document>
```

OK

Exemples de déclarations d'attributs

```
<!ATTLIST document version CDATA #FIXED "1.0">
```

```
<document version="1.0" >  
...  
</document>
```

OK

Exemples de déclarations d'attributs

```
<!ATTLIST document version CDATA #FIXED "1.0">
```

```
<document version="2.0" >  
...  
</document>
```

KO

Exemples de déclarations d'attributs

```
<!ATTLIST document version CDATA #FIXED "1.0">
```

```
<document>
...
</document>
```

OK

Exemples de déclarations d'attributs

```
<!ATTLIST nom
  titre (Mlle|Mme|M.) #REQUIRED
  nom-epouse CDATA #IMPLIED
>
```

```
<nom titre="Mme" nom-epouse="Lenoir">
  Martin
</nom>
```

OK

Exemples de déclarations d'attributs

```
<!ATTLIST nom
  titre (Mlle|Mme|M.) #REQUIRED
  nom-epouse CDATA #IMPLIED
>
```

```
<nom titre="M." nom-epouse="Lenoir">
  Martin
</nom>
```

OK

Exemples de déclarations d'attributs

```
<!ATTLIST nom
  titre (Mlle|Mme|M.) #REQUIRED
  nom-epouse CDATA #IMPLIED
>
```

```
<nom titre="M.">
  Martin
</nom>
```

OK

Exemples de déclarations d'attributs

```
<!ATTLIST nom
  titre (Mlle|Mme|M.) #REQUIRED
  nom-epouse CDATA #IMPLIED
>
```

```
<nom titre="Madame" nom-epouse="Lenoir">
  Martin
</nom>
```

KO

Exemples ID, IDREF, IDREFS

```
<!ELEMENT document (personne*,livre*)>
<!ELEMENT personne (nom , prenom)>
  <!ATTLIST personne id ID #REQUIRED>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT livre (#PCDATA)>
  <!ATTLIST livre auteur IDREF #IMPLIED>
```

document.dtd

Attributs ID, IDREF, IDREFS

- Un attribut **ID** sert à référencer un élément, la valeur de cette référence pouvant être rappelée dans des attributs **IDREF** ou **IDREFS**.
- Un élément ne peut avoir au plus qu'un attribut **ID** et la valeur associée doit être unique dans le document XML. Cette valeur doit être un **nom** XML (donc pas un nombre).
- La valeur de défaut pour un attribut **ID** est obligatoirement **#REQUIRED** ou **#IMPLIED**.
- Une valeur utilisée dans un attribut **IDREF** ou **IDREFS** doit obligatoirement correspondre à celle d'un attribut **ID**.

```
<!ELEMENT document (personne*,livre*)>
<!ELEMENT personne (nom , prenom)>
  <!ATTLIST personne id ID #REQUIRED>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT livre (#PCDATA)>
  <!ATTLIST livre auteur IDREF #IMPLIED>
```

Exemples ID, IDREF, IDREFS

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE document SYSTEM "document.dtd">
<document>
  <personne id="id-1">
    <nom>Dupond</nom>
    <prenom>Martin</prenom>
  </personne>
  <personne id="id-2">
    <nom>Durand</nom>
    <prenom>Helmut</prenom>
  </personne>
  <livre auteur="id-1">Ma vie, mon oeuvre</livre>
</document>
```

OK

Exemples ID, IDREF, IDREFS

```
<!ELEMENT document (personne*,livre*)>
<!ELEMENT personne (nom , prenom)>
  <!ATTLIST personne id ID #REQUIRED>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT livre (#PCDATA)>
  <!ATTLIST livre auteur IDREF #IMPLIED>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE document SYSTEM "document.dtd">
<document>
  <personne id="id-1">
    <nom>Dupond</nom>
    <prenom>Martin</prenom>
  </personne>
  <personne id="id-1">
    <nom>Hardailepick</nom>
    <prenom>Helmut</prenom>
  </personne>
  <livre auteur="id-1">Ma vie, mon oeuvre</livre>
</document>
```

KO

Exemples ID, IDREF, IDREFS

```
<!ELEMENT document (personne*,livre*)>
<!ELEMENT personne (nom , prenom)>
  <!ATTLIST personne id ID #REQUIRED>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT livre (#PCDATA)>
  <!ATTLIST livre auteur IDREF #IMPLIED>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE document SYSTEM "document.dtd">
<document>
  <personne id="id-1">
    <nom>Gaisellepick</nom>
    <prenom>Elmer</prenom>
  </personne>
  <personne id="id-2">
    <nom>Hardailepick</nom>
    <prenom>Helmut</prenom>
  </personne>
  <livre auteur="id-3">Ma vie, mon oeuvre</livre>
</document>
```

KO

Exemples ID, IDREF, IDREFS

```
<!ELEMENT document (personne*,livre*)>
<!ELEMENT personne (nom , prenom)>
  <!ATTLIST personne id ID #REQUIRED>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT livre (#PCDATA)>
  <!ATTLIST livre auteur IDREF #IMPLIED>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE document SYSTEM "document.dtd">
<document>
  <personne id="id-1">
    <nom>Gaisellepick</nom>
    <prenom>Elmer</prenom>
  </personne>
  <personne id="id-2">
    <nom>Hardailepick</nom>
    <prenom>Helmut</prenom>
  </personne>
  <livre auteur="id-1 id-2">Ma vie, mon oeuvre</livre>
</document>
```

KO

Exemples ID, IDREF, IDREFS

```
<!ELEMENT document (personne*,livre*)>
<!ELEMENT personne (nom , prenom)>
  <!ATTLIST personne id ID #REQUIRED>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT livre (#PCDATA)>
  <!ATTLIST livre auteur IDREFS #IMPLIED>
```

document.dtd

Exemples ID, IDREF, IDREFS

```
<!ELEMENT document (personne*,livre*)>
<!ELEMENT personne (nom , prenom)>
  <!ATTLIST personne id ID #REQUIRED>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT livre (#PCDATA)>
  <!ATTLIST livre auteur IDREFS #IMPLIED>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE document SYSTEM "document.dtd">
<document>
  <personne id="id-1">
    <nom>Gaisellepick</nom>
    <prenom>Elmer</prenom>
  </personne>
  <personne id="id-2">
    <nom>Hardailepick</nom>
    <prenom>Helmut</prenom>
  </personne>
  <livre auteur="id-1 id-2">Ma vie, mon oeuvre</livre>
</document>
```

OK

Exemple d'entités

```
<!-- entite externe pour importer les entites -->
<!-- representant les caracteres accentues -->
<!ENTITY % HTMLlat1 PUBLIC
  "-//W3C//ENTITIES Latin 1 for XHTML//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml-lat1.ent">
%HTMLlat1; <!-- c'est comme un import -->
<!-- entite parametrique -->
<!ENTITY % elt "(#PCDATA|elt1)*" >
<!ELEMENT racine %elt;>
<!ELEMENT elt1 (#PCDATA)>
<!--entites interne -->
<!ENTITY euro "&#8364;";>
<!ENTITY LILLE1 "Universit&eacute; Lille 1">
<!--l'utilisation du &eacute; est possible parce que -->
<!--c'est une entité externe importée à l'aide de %HTMLlat1 -->
```

Déclarations d'entités

- Les **entités internes** : «macros exportées» qui sont utilisées dans le document XML validé par la DTD.
`<!ENTITY euro "€";>`, utilisation `€`
- Les **entités paramétriques** : «macros non exportées» qui sont utilisées ailleurs dans la DTD.
`<!ENTITY % editeur "O'Reilly">`, utilisation `%editeur;`
- Les **entités externes** : «macros importées» définies dans un autre document, utilisables dans la DTD elle-même ou dans tout document XML valide pour la DTD.
`<!ENTITY % HTMLlat1 PUBLIC
 "-//W3C//ENTITIES Latin 1 for XHTML//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml-lat1.ent">`

Exemple de document valide pour cette DTD

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE racine SYSTEM "./entites.dtd">
<racine>
  blabla
  <elt1>
    Universit&eacute; ; : &LILLE1;
  </elt1>
  <elt1>
    10000 &euro;
  </elt1>
  c'est fini !
</racine>
```