NFP121, Cnam/Paris
Sérialisation
XML et Java
SAX/JDOM

jean-michel Douin, douin au cnam point fr
version : 16 Décembre 2016

Notes de cours

Bibliographie

- SAX
  - http://www.ibm.com/developerworks/xml/tutorials/x-usax/section4.html
- JDOM
  - www.**jdom**.org/dist/docs/presentations/mtvjug04262000.**ppt**
  - **http://www.dom4j.org/dom4j-1.6.1/**

Sérialisation

- **Sommaire**
  - **Au format Java**
    - **Implements java.io.Serializable**
    - **writeObject, readObject**
  - **Au format XML**
    - **API JDOM, SAX**
  - **Au format JSON**

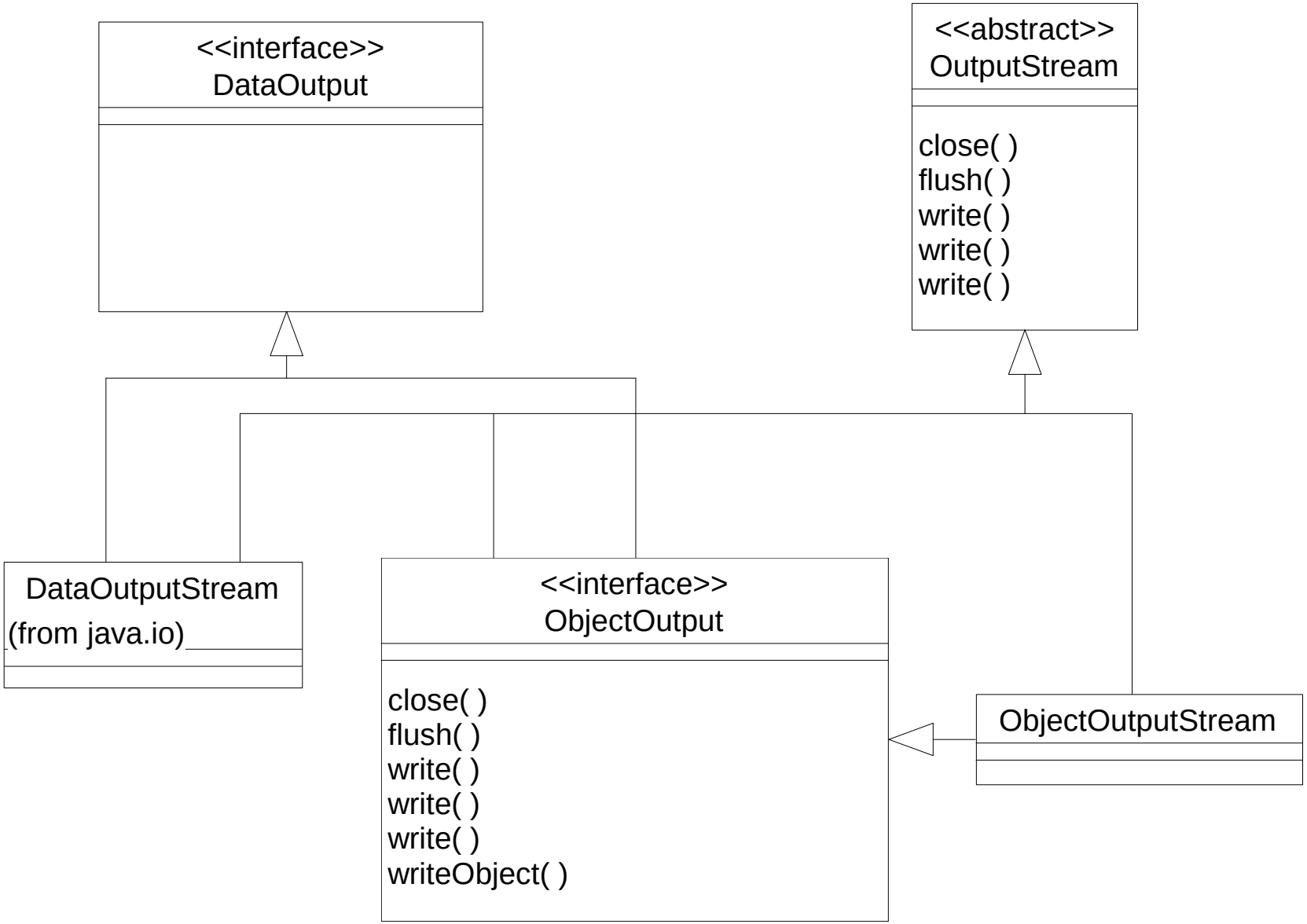
Persistance et Sérialisation

Entrées/Sorties **d'objets quelconques**.

->la "*Serialization*" qui assure ainsi la persistance de l'état d'un objet.

-> copie par valeur, quelque soit la complexité de l'instance

Serialisation



Sauvegarde d'un objet sur fichier

```
// une décoration
ObjectOutputStream out =
        new ObjectOutputStream(
                new FileOutputStream(fileName)
        );

out.writeObject(obj);
```

Exemple

```
class AuditeurCnam implements Serializable{
    private String nom;
    private List<Unité> liste = …

    public String toString(){
       return "nom :"+nom+"  liste: "+liste;
    }
}
```

Instance non sérialisable

Les objets dont l'état dépend de celui de l'environnement d'exécution ne sont pas sérialisables tels quels

exemples : identificateur de thread ou de process, descripteur de fichiers, socket réseau, ...

```
private transient Thread t;

private transient Password pass;
```

ObjectOutputStream

Seules des classes implémentant Serializable ou Externalizable permettent la sérialisation de leurs objets.

Le fichier contient

le nom de cette classe,

sa signature ainsi que la valeur de tous les champs non 'static' et non 'transient' ,

ceci récursivement pour tous les objets sérialisables définis dans les attributs.

Démonstration

```
Unité nfp121 = new Unité("NFP121",6);
nfp121.inscrire(new Auditeur("bbb","1234"));
nfp121.inscrire(new Auditeur("aaa","321"));
nfp121.inscrire(new Auditeur("ccc","456"));
nfp121.inscrire(new Auditeur("zzz","888"));
nfp121.inscrire(new Auditeur("yyyy","888"));
```

```
ObjectOutputStream oos=
    new ObjectOutputStream(new FileOutputStream(“nfp121.ser”));
```

```
oos.writeObject(nfp121);
```

```
oos.close
```

Compatibilité ascendante ?

- **Un scénario :**
  1. **Persistance d'une instance dans un fichier**
  2. **Modification de la classe de cette instance,**
     - **du source Java (ajout d'un attribut)**
  3. **Lecture de ce fichier, création d'une instance**
     - **Compatibilité ascendante ? Valeur du nouvel attribut ?**
     - **Exception ! Mauvaise version de la classe, sauf si …**

- **>serialver –show**

- **Une estampille de votre .class**
  - **private static final serialVersionUID = xxxx;**
  - **À installer dans le source de votre classe**

  – Démonstration

ObjectOutputStream

La sauvegarde par défaut ne convient pas

Sécurité, données cryptées, …

Votre classe propose ces deux méthodes **privées**

```
private Object readObject(ObjectInputStream stream))
    throws IOException, ClassNotFoundException

private void writeObject(ObjectOutputStream stream))
    throws IOException
```

… oos.writeObject(et4101e);

déclenche la méthode privée (writeObject) de la classe Unité

Par introspection à l'exécution …

interface Externalizable

hérite de Serializable

permet de contrôler la totalité de la sauvegarde, y compris de celle des super classes (seule l'identification de la classe de l'objet est traitée automatiquement) void readExternal(ObjectInput oi);

void writeExternal(ObjectOutput oo);

Cette interface ne supporte pas automatiquement la gestion des versions de classes.

Transmission sur sockets

```
Socket s = new Socket(remoteHost, remotePort);
OutputStream os = s.getOutputStream();
ObjectOutputStream out = new ObjectOutputStream(os);
```

out.writeObject(obj);

Démonstration…

En TCP le Client, protocole « java » (le serveur:jfod)

```
public class Client{

  public static void main(String[] args) throws Exception{
    // ouverture d'une connexion TCP
    Socket socket = new Socket("jfod.cnam.fr", 5000);
    ObjectOutputStream oos= new ObjectOutputStream( socket.getOutputStream());



    // envoi vers le serveur de cette « requête »
    SortedSet<String> l = new TreeSet<String>();
    l.add("TCP");l.add("essai");
    oos.writeObject( l);



    // lecture de la réponse retournée
    ObjectInputStream ois= new ObjectInputStream(
   socket.getInputStream());
    System.out.println("le serveur retourne : " + ois.readObject());

    socket.close();
    }
}
```

En TCP un serveur (jfod), protocole « java »

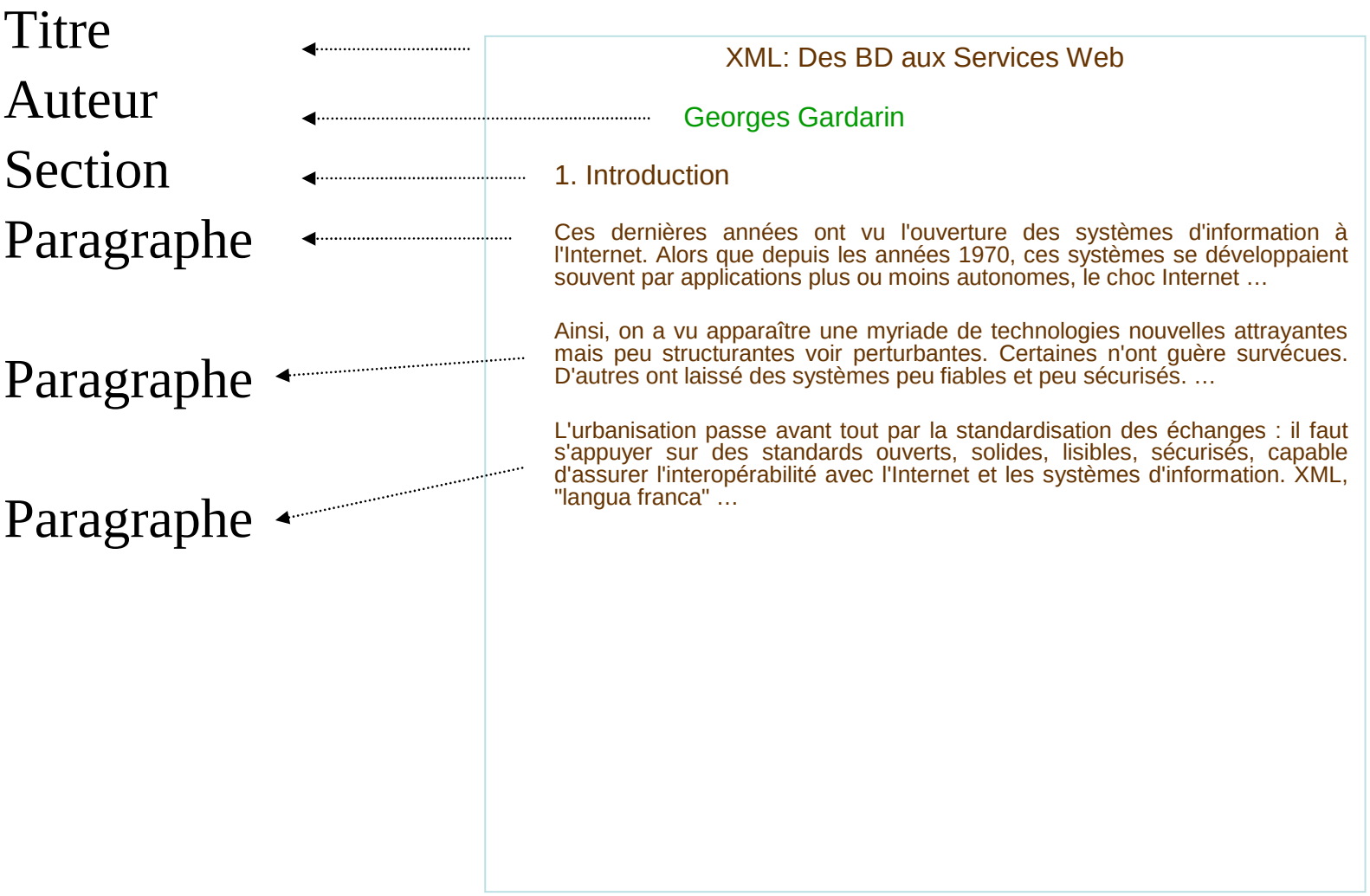
```
public class Serveur{

public static void main(String[] args) throws Exception{
  ServerSocket serveur = new ServerSocket(5000);

  while(true) {
    Socket socket = serveur.accept(); // attente active d'un client

// requête
   ObjectInputStream ois= new ObjectInputStream(socket.getInputStream());
   Object obj = ois.readObject();


// réponse
ObjectOutputStream oos= new ObjectOutputStream(socket.getOutputStream());
oos.writeObject(obj.toString());

     socket.close();
    }
  }
}
```

Démonstration

Sommaire Sérialisation XML

- **XML**
  - Comme eXtended Markup Language
- **SAX**
  - Simple API for XML
- **DOM / JDOM**
  - Document Object Model / Java DOM
- **Structures arborescentes**
  - Le patron Composite et une représentation en XML
- **Un bean et sa représentation XML**

Structuration des données



Vue Balisée en XML

<Livre>

<Titre> XML : Des BD aux Services Web </Titre>

<Auteur>Georges Gardarin</Auteur>

<Section titre = "Introduction">

<Paragraphe>Ces dernières années ont vu l'ouverture des systèmes d'information à l'Internet. Alors que depuis les années 1970, ces systèmes se développaient souvent par applications plus ou moins autonomes, le choc Internet … </Paragraphe>

<Paragraphe>Ainsi, on a vu apparaître une myriade de technologies nouvelles attrayantes mais peu structurantes voir perturbantes. Certaines n'ont guère survécues. D'autres ont laissé des systèmes peu fiables et peu sécurisés. ...</Paragraphe>

<Paragraphe>L'urbanisation passe avant tout par la standardisation des échanges : il faut s'appuyer sur des standards ouverts, solides, lisibles, sécurisés, capable d'assurer l'interopérabilité avec l'Internet et les systèmes d'information. XML, "langua franca" … </Paragraphe>

</Section>

</Livre>

Définir un langage

```
<?xml version="1.0"?>
<!– ANT build file -->
<project name="tutorial" default="build" basedir=".">
    <target name="build">
        <javac srcdir="." />
    </target>
</project>
```

Définir une configuration ici "pour Android"

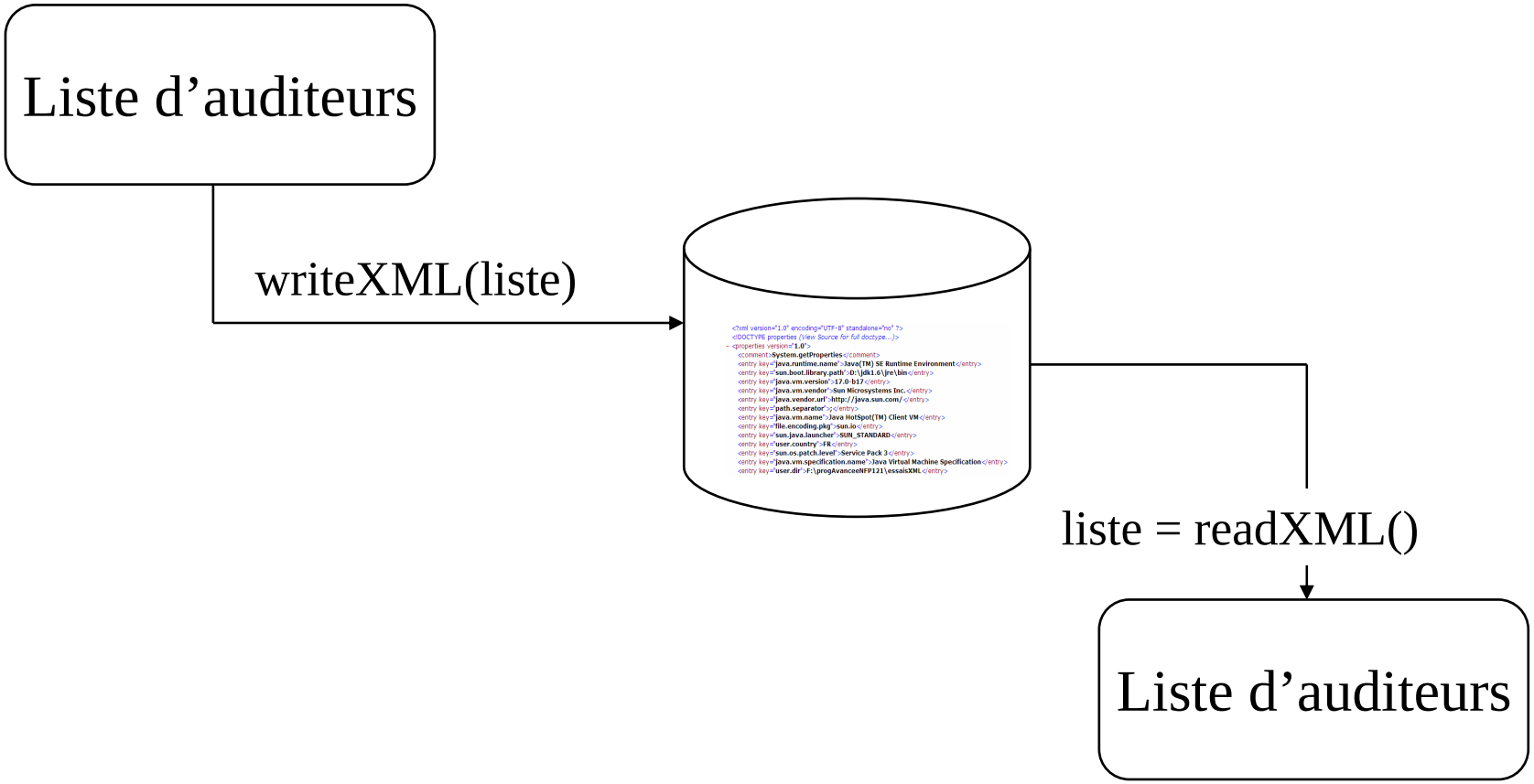
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="test.biblio"
      android:versionCode="1"
      android:versionName="1.0">

      <uses-permission android:name="android.permission.INTERNET" />

    <application android:icon="@drawable/icon"
     android:label="@string/app_name">
        <activity android:name=".BrowserDemo"
                android:label="@string/app_name">

            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

XML pourquoi faire ?

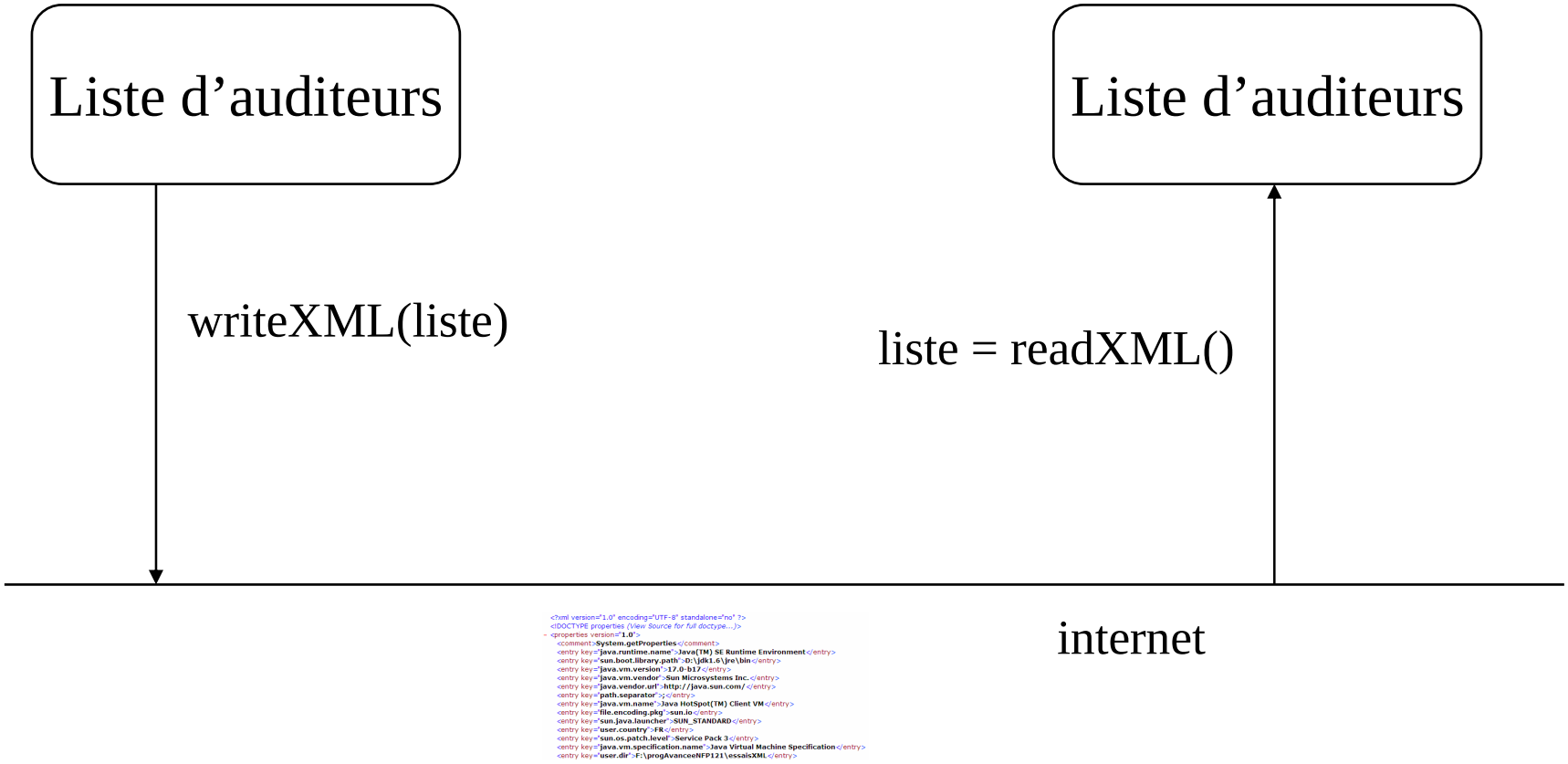
Définir une IHM ici pour Android

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical" android:layout_width="fill_parent"
  android:layout_height="fill_parent" android:gravity="center_horizontal">

  <TextView android:layout_width="fill_parent"
   android:layout_height="wrap_content" android:text="@string/hello" />

   <LinearLayout android:id="@+id/LinearLayout02"
     android:layout_width="wrap_content" android:layout_height="wrap_content"
     android:gravity="center_horizontal">
     <Button android:text="left" android:id="@+id/buttonLeft"
       android:layout_width="wrap_content" android:layout_height="wrap_content"
       android:onClick="onClick" android:width="80px"></Button>
   …
   </LinearLayout>

 <LinearLayout android:id="@+id/LinearLayout01"
   android:layout_width="wrap_content" android:layout_height="wrap_content">
   <View android:id="@+id/touchView" android:layout_width="wrap_content"
     android:layout_height="wrap_content"></View>
  </LinearLayout>
</LinearLayout>
```

Persistance d'un objet et "lisible"



Format d'échange



XML, « parser », interpréteur

- **3 exemples**
  - **L'outil ant**
  - **Environnement de développement**
  - **Une persistance d'objet**

Exemple 1:
Le document ANT est 'parsé' et interprété

```
<?xml version="1.0"?>

<!– ANT build file -->

<project name="tutorial" default="build" basedir=".">

        <target name="build">

                <javac srcdir="." />

        </target>

</project>
```

>ant

Buildfile: build.xml

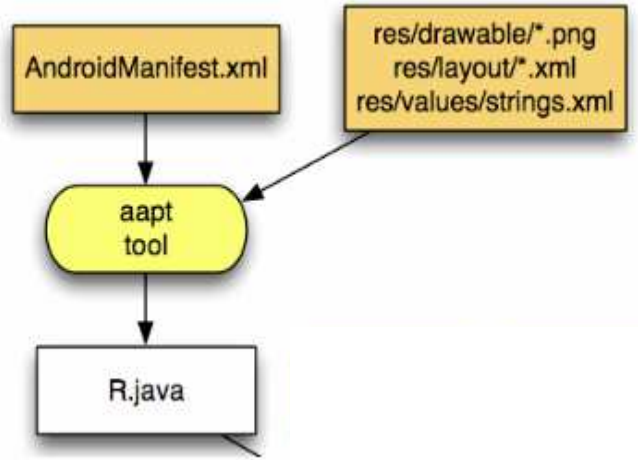
build: [javac] Compiling 1 source file

BUILD SUCCESSFUL Total time: 3 seconds

Exemple 2 le fichier de configuration Android AndroidManifest.xml,… → « R.java »

- **res/layout/main.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
  …
</LinearLayout>
```



- **/test/biblio/R.java** **AUTO-GENERATED FILE. DO NOT MODIFY.**

```
package test;
public final class R {
    …
    public static final class layout {
        public static final int main=0x7f030000;
    }}
```

Exemple 3 : Properties de java.util

Écriture

```
Properties props = System.getProperties();
props.storeToXML(new FileOutputStream(new File("props.xml")),"System.getProperties");
```

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE properties (View Source for full doctype...)>
- <properties version="1.0">
    <comment>System.getProperties</comment>
    <entry key="java.runtime.name">Java(TM) SE Runtime Environment</entry>
    <entry key="sun.boot.library.path">D:\jdk1.6\jre\bin</entry>
    <entry key="java.vm.version">17.0-b17</entry>
    <entry key="java.vm.vendor">Sun Microsystems Inc.</entry>
    <entry key="java.vendor.url">http://java.sun.com/</entry>
    <entry key="path.separator">;</entry>
    <entry key="java.vm.name">Java HotSpot(TM) Client VM</entry>
    <entry key="file.encoding.pkg">sun.io</entry>
    <entry key="sun.java.launcher">SUN_STANDARD</entry>
    <entry key="user.country">FR</entry>
    <entry key="sun.os.patch.level">Service Pack 3</entry>
    <entry key="java.vm.specification.name">Java Virtual Machine Specification</entry>
    <entry key="user.dir">F:\progAvanceeNFP121\essaisXML</entry>
```

Le fichier **props.xml** ici lu par un Navigateur

Lecture

```
Properties props = new Properties();
props.loadFromXML(new FileInputStream(new File("props.xml")));
assertTrue(System.getProperties().equals(props));
```

Démonstration

XML : la famille

- **Né : fin 96**
- **Père : W3C**
- **Petit-fils de SGML (ISO-1986)**
- **Cousin d'HTML**
- **Reconnu le : 10/02/98 – version 1.0**
- **Descendance – XHMTL, MathML, ANT…**

X comme eXtensible

- **HTML : nombre fini de balises**
- **XML : possibilité de définir les balises**

- **HTLM : balises pour formater**
- **XML : balises pour structurer**

- **DTD ou Schéma pour définir les balises**

Règles syntaxiques

- **Commencer par une déclaration XML**
- **Balisage sensible à la casse**
- **La valeur des attributs doit être quotée**
- **Balises non vides appariées** **
</br>**
- **Balises vides fermées** **
**
- **Les élèments ne doivent pas se chevaucher**
  - <jour> <mois> </jour> </mois> **interdit**
- **Un élément doit encapsuler tous les autres**
- **Ne pas utiliser les caractères < et & seuls**

Le Prologue

- **Une déclaration XML**

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
```

- **Instructions de traitement (PI Processing Instruction)**
  - **Une indication de traitement est destinée aux applications qui manipulent les documents XML**

- **Une déclaration de type de document**
  - **indique le type de document auquel se conforme le document en question (ex. DTD)**

```
<!DOCTYPE rapport SYSTEM "rapport.dtd">
```

Élément

- **Composant de base**
- **Identifié par un nom**
- **Délimité par une balise ouvrante et une balise fermante à ce nom**

```
<AUTEUR> Victor Hugo </AUTEUR>
```

- **Ou élément vide**

```
<PHOTO Source= "victor.gif"  />
```

- **Contenu textuel, éléments ou mixte**

Les attributs

- **Inclus dans la balise ouvrante d'un élément**
- **Composé d'un nom et d'une valeur**

```
<AUTEUR NE="1802" MORT="1885" >
    Victor Hugo
</AUTEUR>
```

Données

- **Constituées par un flot de caractères**
  - **tous les caractères sont acceptés sauf le caractère « & » et le caractère « < »**
  - **Exemple :** `<auteurs>Victor Hugo<auteurs>`

- **Si l 'on souhaite insérer des caractères « spéciaux », il est préférable d'utiliser une section littérale ou CDATA**

```
<![CDATA[<auteurs>S. Fleury &amp; al.</auteurs>]]>
```

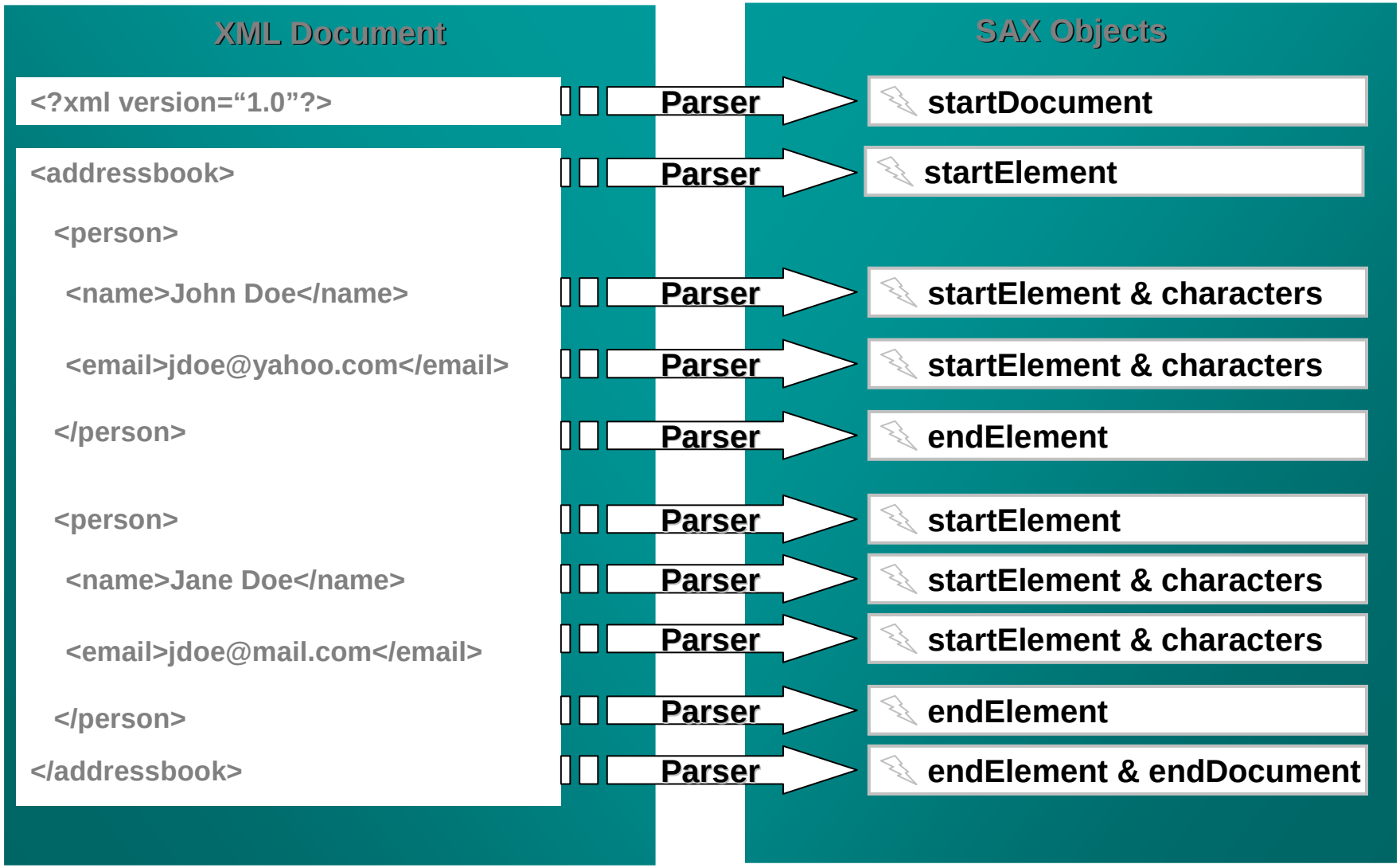
qui se traduit en :

```
<auteurs>S. Fleury &amp; al.</auteurs>
```

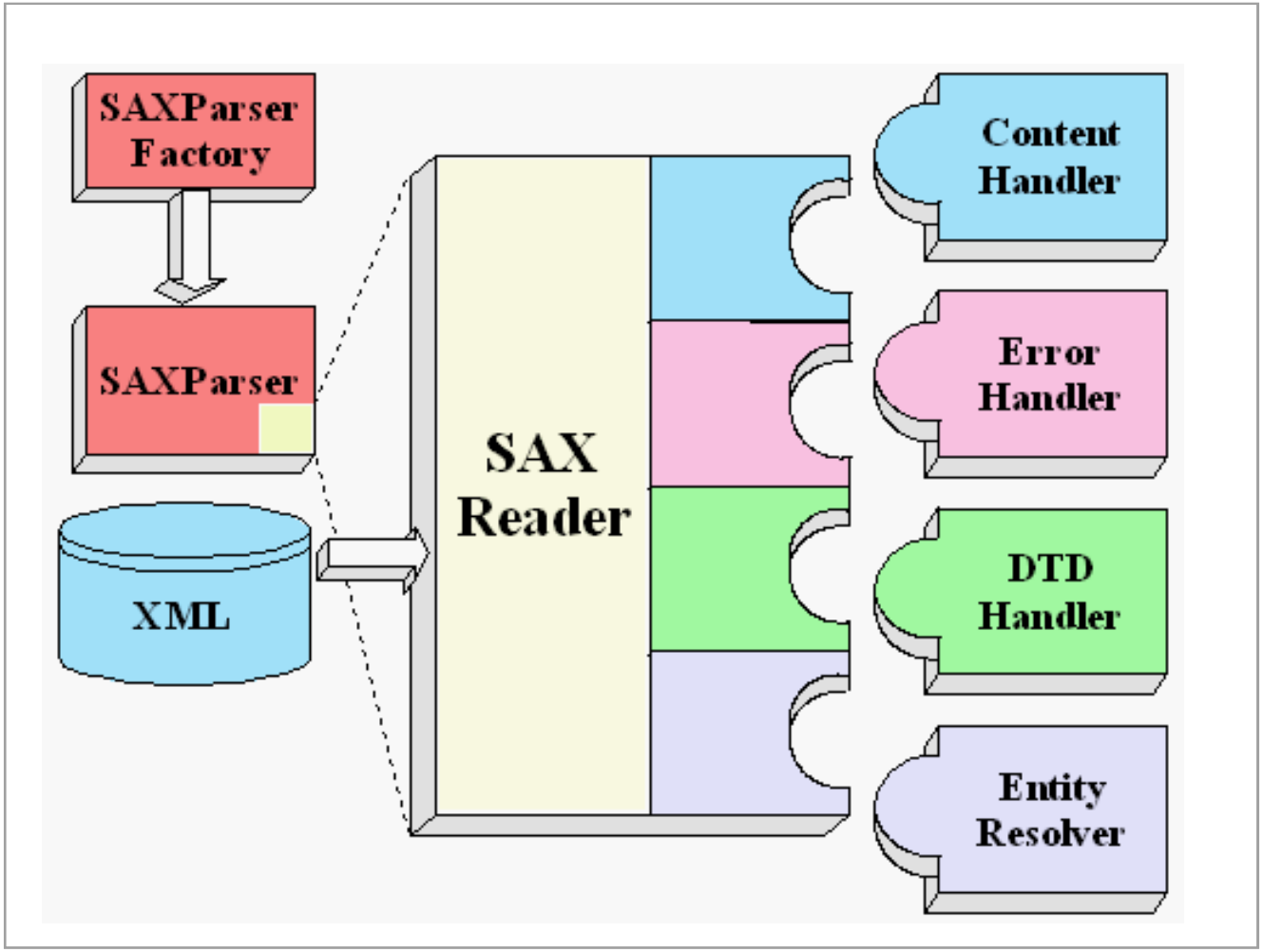
SAX Simple Api for Xml

- Début des travaux Dec, 1997
- SAX 1.0 Mai, 1998
  - Tim Bray
  - David Megginson
  - …
- SAX 2.0 Mai, 2000
- SAX 2.0.2 *27-April 2004:*
- ….

SAX Comment ?


XML Document
SAX Objects
<?xml version="1.0"?>
<addressbook>
<person>
<name>John Doe</name>
<email>jdoe@yahoo.com</email>
</person>
<person>
<name>Jane Doe</name>
<email>jdoe@mail.com</email>
</person>
</addressbook>
Parser
startDocument
startElement
startElement & characters
startElement & characters
endElement
startElement
startElement & characters
startElement & characters
endElement
endElement & endDocument

Implémenter les Handlers d'évènements du parseur



DefaultHandler

Il implémente ces différents Handler avec des méthodes vides, de sorte que l'on peut surcharger seulement celles qui nous intéressent.

Structure du main

```
import org.xml.sax.helpers.DefaultHandler;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import org.xml.sax.XMLReader;
public class SurveyReader extends DefaultHandler{
    public static void main (String args[]) {
        XMLReader xmlReader = null;
        try { SAXParserFactory spfactory = SAXParserFactory.newInstance();
            SAXParser saxParser = spfactory.newSAXParser();
            xmlReader = saxParser.getXMLReader();
            xmlReader.setContentHandler(new SurveyReader());
            InputSource source = new InputSource("surveys.xml");
            xmlReader.parse(source);
        } catch (Exception e) { System.err.println(e); System.exit(1);
        }
    }}
```

org.xml.sax.ContentHandler

- Toutes les applications SAX doivent implanter un ContentHandler

- Méthodes :
  - public void **startDocument**() throws SAXException
  - public void **endDocument**() throws SAXException
  - public void **startElement**(String nspURI, String localName, String qName, Attributes atts) throws SAXException
  - public void **characters**(char[] ch, int start, int length) throws SAXException
  - …

Récupérer le début d'analyse de chaque élément

```
...
import org.xml.sax.Attributes;
public class SurveyReader extends DefaultHandler {
    String thisElement = "", thisQuestion = "";
    public void startDocument() throws SAXException {
        System.out.println("Tallying survey results...");
    }
    public void startElement( String namespaceURI, String localName,
                              String qName, Attributes atts) throws SAXException {
        System.out.print("Start element: ");
        System.out.println(qName);
        thisElement = qName;
        if(qname.equals("question")){thisQuestion = atts.getValue ("subject");}
    }...}
```

Exemple de traitement SAX: startElement(…)

```
<?xml version="1.0"?>
<surveys>
<response username="bob">
<question subject="appearance">A</question>
<question subject="communication">B</question>
<question subject="ship">A</question>
<question subject="inside">D</question>
<question subject="implant">B</question>
</response>
<response username="sue">
<question subject="appearance">C</question>
<question subject="communication">A</question>
<question subject="ship">A</question>
<question subject="inside">D</question>
<question subject="implant">A</question>
</response>
…. </surveys>
```

```
Tallying survey results...
Start element: surveys
Start element: response
Start element: question
Start element: question
Start element: question
Start element: question
Start element: question
Start element: response
Start element: question
Start element: question
Start element: question
Start element: question
Start element: question
Start element: response
Start element: question
Start element: question
Start element: question
Start element: question
Start element: question
```

Récupérer les données

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

    if (thisElement.equals("question“)) {
    System.out.print(thisQuestion + ": ");
    System.out.println(new String(ch, start, length));
}
}
 . . .
```

exemple

```
Tallying survey results...
User: bob
     appearance: A
     communication: B
     ship: A
     inside: D
     implant: B
User: sue
     appearance: C
     communication: A
     ship: A
     inside: D
     implant: A
User: carol
     appearance: A
     communication: C
     ship: A
     inside: D
     implant: C
```

Un autre exemple les stations Vélib

- **http://www.velib.paris.fr**
- **http://www.velib.paris.fr/service/carto**
- **http://www.velib.paris.fr/service/stationdetails/{number}**

http://www.velib.paris.fr/service/carto

```
<carto>
 <markers>
   <marker name="00901 - STATION MOBILE 1" number="901"
     address="ALLEE DU BELVEDERE PARIS 19 - 0 75000 Paris
     -" fullAddress="ALLEE DU BELVEDERE PARIS 19 - 0 75000
     Paris - 75000 PARIS" lat="48.892745582406675"
     lng="2.391255159886939" open="1" bonus="0"/>


   <marker name="03011 -
     TURBIGO" number="3011" address="55 RUE TURBIGO -
     " fullAddress="55 RUE TURBIGO - 75003
     PARIS" lat="48.86558781525867"lng="2.356094545731025"
      open="1" bonus="0"/>
```

Analyse des attributs

de la balise `marker`

en SAX -> méthode startElement

http://www.velib.paris.fr/service/stationdetails/3011

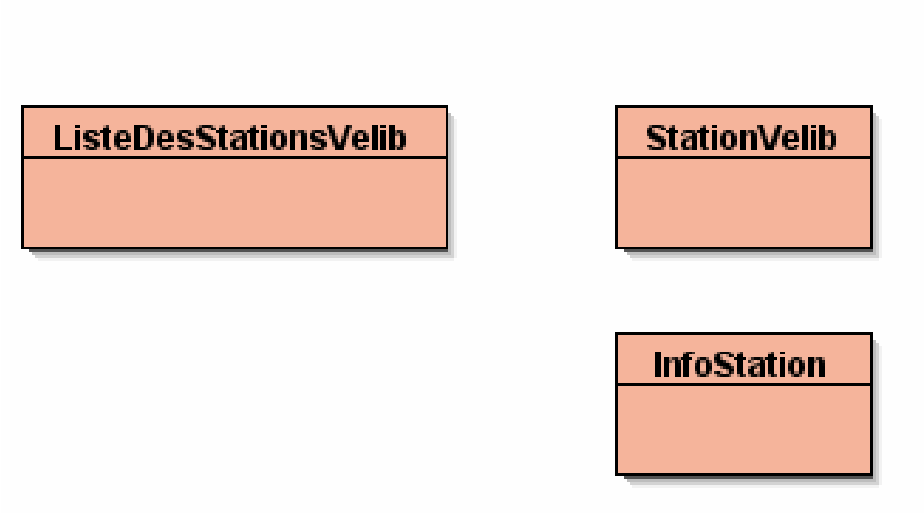
```
<station>
    <available>21</available>
    <free>10</free>
    <total>31</total>
    <ticket>1</ticket>
</station>
```

Analyse du contenu

de la balise `station`

en SAX -> des méthodes startElement, endElement, characters

Un autre exemple les stations Vélib



- Les Classes, *un premier découpage*
  - **StationVelib,**
    - **toutes les infos d'une station, (adresse, longitude, latitude,…)**
  - **InfoStation,**
    - **les informations comme le nombre de vélo et d'emplacements disponibles,...**
  - **ListeDesStationsVelib**
    - **La gestion de la liste des stations**

  - **http://www.velib.paris.fr/service/carto**
  - **http://www.velib.paris.fr/service/stationdetails/{number}**

Initialisation du « parser »

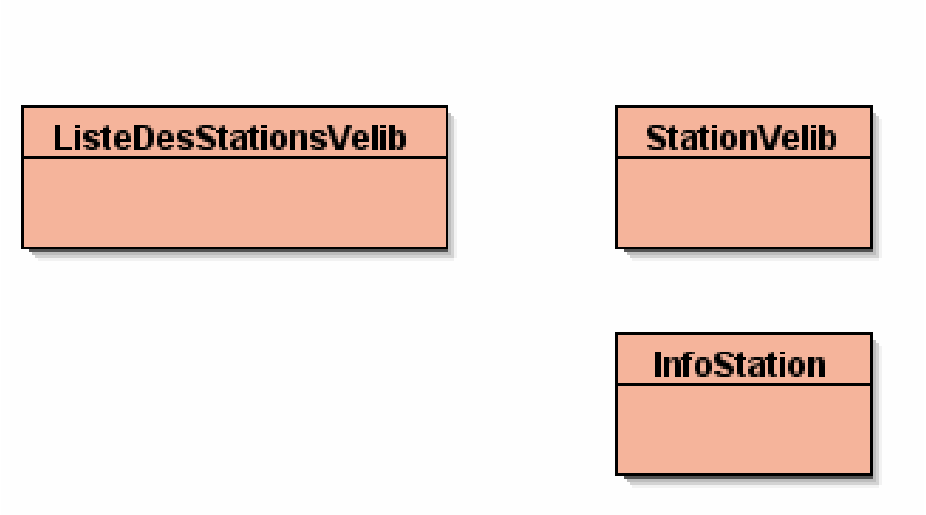
```
class ParserXML  extends DefaultHandler {

  public ParserXML(InputStream in)
                              throws Exception{
    SAXParserFactory spf =
              SAXParserFactory.newInstance();
    SAXParser sp = spf.newSAXParser();

    XMLReader xr = sp.getXMLReader();
    xr.setContentHandler(this);
    xr.parse(new InputSource(in));
    }
```

startElement un extrait

// Création d'une instance de la classe StationVelib
// depuis XML en Java

```
public void startElement(String uri, String localName, String
   qName,Attributes attributes) throws SAXException {

   super.startElement(uri, localName, qName, attributes);


   if(qName.equals("marker")){
     StationVelib station = new StationVelib();

     station.setName(attributes.getValue("name"));
     station.setNumber(Integer.parseInt(attributes.getValue("number")));
     station.setAddress(attributes.getValue("address"));
     station.setLatitude(Double.parseDouble(attributes.getValue("lat")));
     station.setLongitude(Double.parseDouble(attributes.getValue("lng")));

   }
```

Démonstration



Une Info à chaque Station

```
class ParserXML extends DefaultHandler {

   private StringBuffer current; // la valeur


   public ParserXML(int ID){
     URL url = new URL(URL_VELIB_INFO + ID);

     SAXParserFactory spf = SAXParserFactory.newInstance();
     SAXParser sp;
     sp = spf.newSAXParser();
     XMLReader xr = sp.getXMLReader();
     xr.setContentHandler(this);
     xr.parse(new InputSource(url.openStream()));

}
```

A chaque noeud

```
public void startElement (String uri, String localName, String qName,
                Attributes attributes) throws SAXException {
            super.startElement(uri, localName, qName, attributes);
            current = new StringBuffer();
     }


     public void characters (char[] ch, int start, int length) throws SAXException {
            super.characters(ch, start, length);
            current.append(new String(ch, start, length));
     }


   public void endElement (String uri, String localName, String qName)
          throws SAXException {
          super.endElement(uri, localName, qName);
          if(qName.equals("available")){
              available = Integer.parseInt(current.toString());
           …
   }
```

Démonstration …

- **Un vélo est-il disponible ?**
  - **place d'Italie**
    - **N° 13008, 13010**
  - **55 rue Turbigo**
    - **N° 3011**
- **Combien de vélos en circulation en ce moment ?**

Combien de vélos ?

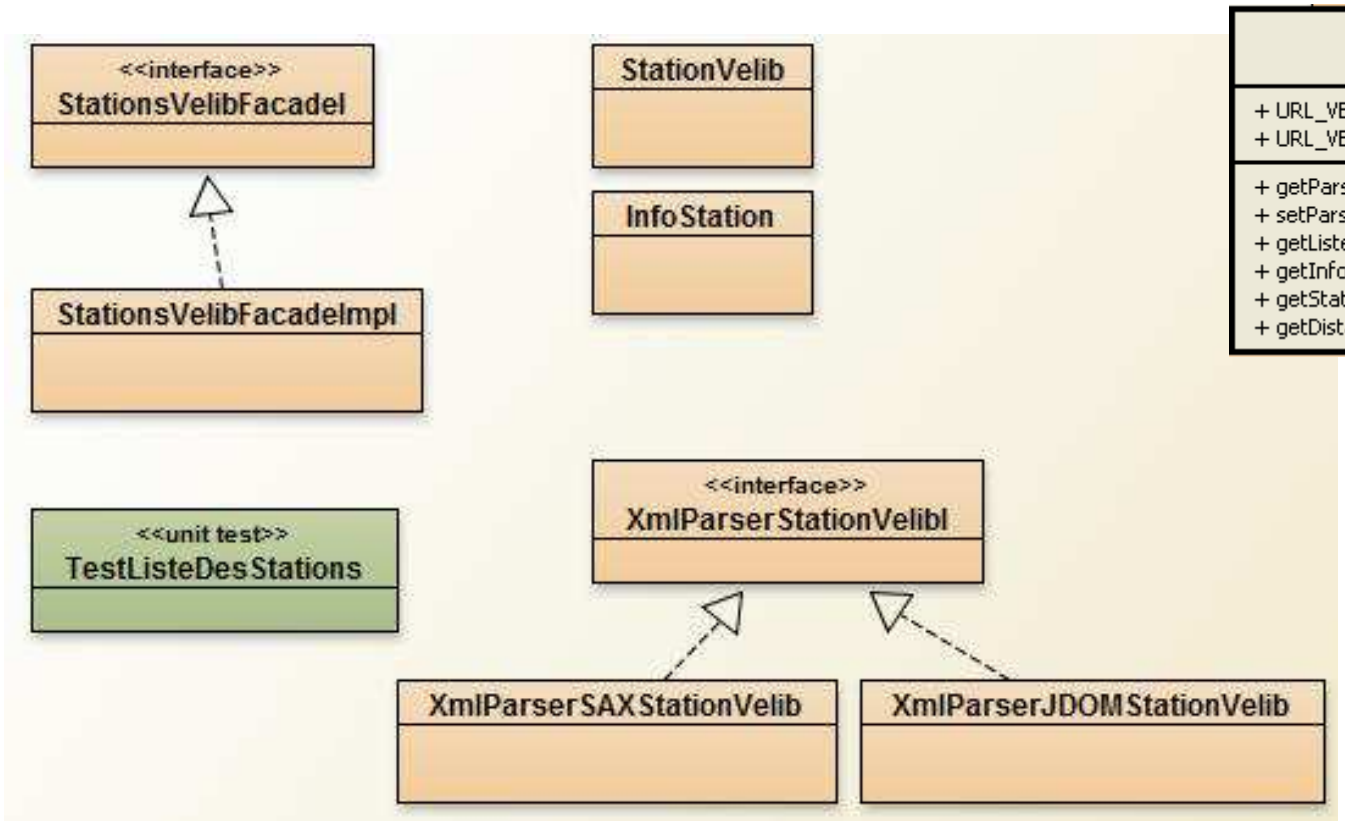
```
URL url = new URL("http://www.velib.paris.fr/service/carto");
ListeDesStationsVelib lesStations = new ListeDesStationsVelib(url.openStream());

//System.out.println(lesStations);
int slotsDisponibles = 0;
int vélosDisponibles = 0;
int compteur = 0;
for (StationVelib s : lesStations){
  if(s.getOpen()){
    compteur++;
    InfoStation info = lesStations.info(s);
    slotsDisponibles = slotsDisponibles + info.getFree();
    vélosDisponibles = vélosDisponibles + info.getAvailable();
  }
}

System.out.println(compteur + " stations");
System.out.println(" slots disponibles :  " + slotsDisponibles);
System.out.println(" vélos disponibles :  " + vélosDisponibles);
```

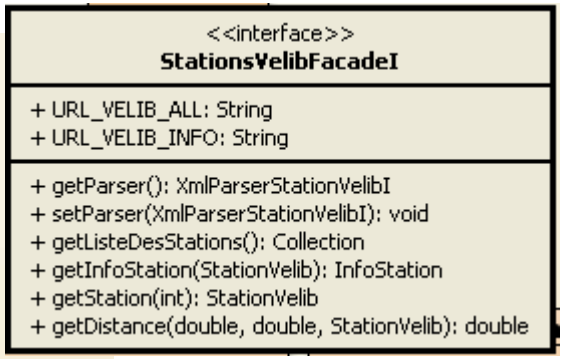
- **Plus de 1400 requêtes …**

Démonstration

- **Aujourd'hui**
  - **. stations**
  - **slots disponibles : ?**
  - **vélos disponibles : ?**
  - **Chiffres à valider …**
- **En direct http://www.parisavelo.net/**

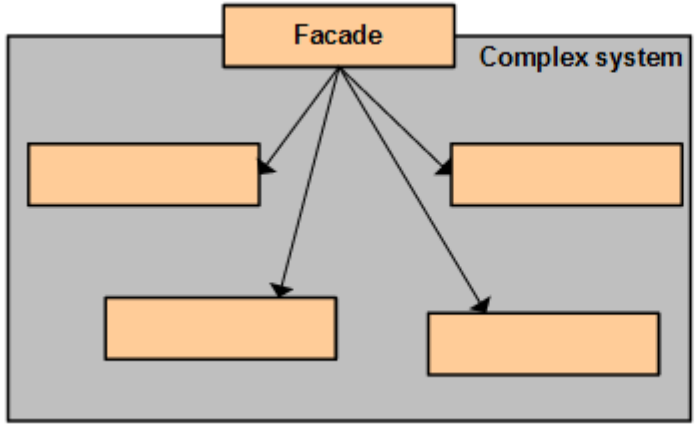
Architecture Vélib refactoring





- **Interface et couplage faible…**
- **DAO, Discussion**

Le patron Façade



Facade

Type: Structural

What it is:
Provide a unified interface to a set of interfaces in a subsystem. Defines a high-level interface that makes the subsystem easier to use.

- **Une mise en œuvre cf. le site de l'unité**

Valider un fichier XML, une DTD ?

```
...
public static void main (String args[]) {
    XMLReader xmlReader = null;
    try {
        SAXParserFactory spfactory =
        SAXParserFactory.newInstance();
        spfactory.setValidating(true);
        SAXParser saxParser = spfactory.newSAXParser();
        xmlReader = saxParser.getXMLReader();
    } catch (Exception e) {
    System.err.println(e); System.exit(1);
    }
}
```

Qu'est qu'une DTD ?

D'après *F. Nolot*

- Elle permet de vérifier qu'un document XML est conforme à une syntaxe donnée (à une grammaire)

DTD comme Document Type Definition

- On distingue 2 types de conformité
  - Les documents valides : les documents XML avec une DTD
  - Les documents bien formés : les documents XML ne comportant pas de DTD mais répondant aux règles de base du XML

- Une DTD peut être définie de 2 façons
  - Sous forme interne, incluant la grammaire dans le document
  - Sous forme externe, soit en appelant un fichier contenant la grammaire à partir d'un fichier local ou bien en y accédant par son URL

Décalaration d'une DTD interne

- <?xml version="1.0" ?>
  <!DOCTYPE purchase_order [
  <!ELEMENT purchase_order (customer)>
  <!ELEMENT customer (account_id, name)>
  <!ELEMENT account_id (#PCDATA)>
  <!ELEMENT name (first, mi, last)>
  <!ELEMENT first (#PCDATA)>
  <!ELEMENT mi (#PCDATA)>
  <!ELEMENT last (#PCDATA)>
  ]>

- <purchase_order>
  <customer>
  <account_id>10-487</account_id>
  <name>
  <first> William </first>
  <mi> R </mi>
  <last> Stanek </last>
  </name>
  </customer>
  </purchase_order>

Déclaration d'une DTD externe

- **Dans un document XML, dans le cas de l'utilisation d'une DTD externe, on doit alors avoir :standalone="no"**
- **Puis l'élément <!DOCTYPE elt_racine SYSTEM "filename.dtd">**

- <?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
- <!DOCTYPE carnet SYSTEM "../Cours5-Solution/Exo1.dtd">
- <carnet>
- <NomPersonne>
- <M/>
- <Prenom>George</Prenom>
- <Nom>FILOCHE</Nom>
- </NomPersonne>
- <NomPersonne>
- <Mlle/>
- <Prenom>Martine</Prenom>
- <Prenom2>Yvonne</Prenom2>
- <Nom>GETUDAVE</Nom>
- </NomPersonne>
- </carnet>

Déclaration d'une DTD

- **<!DOCTYPE elt_racine SYSTEM|PUBLIC emplacement1 emplacement2>**
- **SYSTEM s'utilise avec une DTD externe dont l'emplacement est**
  - **Soit une URL http://www.....**
  - **Soit une URI file:///home/user/DTD/.....**
- **PUBLIC permet l'utilisation d'une référence générique à la DTD par l'intermédiaire d'un URI, voire un second URI**
- **Plutôt utiliser avec des DTD normalisées disponibles au public**
- **Exemple : validité d'un document HTML4**
- **<!doctype html public '-//W3C//DTD HTML 4.0//EN' 'http://www.w3.org/TR/REC-html40/strict.dtd'>**

Exemple de DTD

- **Définissons la conformité d'un fichier XML qui contient des informations concernant une personne**

- **les informations suivantes sont supposées nécessaires**
  - **personne:**
  - **Nom**
  - **Prénom**
  - **Téléphone**

  - **L'élément email est optionnel**

- <!ELEMENT personne (nom,prenom,telephone,email?) >
- <!ELEMENT nom (#PCDATA) >
- <!ELEMENT prenom (#PCDATA) >
- <!ELEMENT telephone (#PCDATA) >
- <!ELEMENT email (#PCDATA) >

Explication de la syntaxe

- **Déclaration d'un élément**
  - <! ELEMENT nom modèle>
- **Le paramètre modèle représente soit un type de données prédéfinies, soit une règle d'utilisation de l'élément**
- **Les types prédéfinis utilisables sont les suivants :**
  - ANY **: l'élément peut contenir tout type de données. A utiliser avec précaution car il supprime quasiment tout contrôle de validité**
  - EMPTY **: l'élément ne contient pas de données spécifiques**
  - #PCDATA **: l'élément doit contenir une chaîne de caractère**
- **Un élément (produit) qui ne doit contenir que des caractères (#PCDATA), sera défini de la façon suivante :**
  - <!ELEMENT produit (#PCDATA)>

Spécifications éléments

(#PCDATA)	Parsed Character DATA
(ELT)	1 fois ELT
(ELT1,ELT2)	Séquence
(ELT1|ELT2|…)	Choix
ELT?	0 ou 1 fois ELT
ELT+	au moins 1 fois ELT
ELT*	0 ou plusieurs fois ELT
()	groupe de sous éléments
ANY	n'importe quoi
EMPTY	rien

Exemple

- Un élément NomPersonne est composé
  - Soit d'un sigle M, Mme, Mlle
  - D'un prénom
  - D'un 2ième prénom
  - Et d'un nom de famille

Ce qui donne

- <!ELEMENT NomPersonne ((M | Mme | Mlle), Prenom, Prenom2, Nom) >
- <!ELEMENT M EMPTY>
- <!ELEMENT Mme EMPTY>
- <!ELEMENT Mlle EMPTY>
- <!ELEMENT prenom (#PCDATA) >
- <!ELEMENT prenom2 (#PCDATA) >
- <!ELEMENT nom (#PCDATA) >

Le document suivant est donc conforme

- <NomPersonne>
- <M/>
- <Prenom>John</Prenom>
- <Prenom2>Edouard</Prenom2>
- <Nom>Martin</Nom>
- </NomPersonne>

Autre exemple de DTD

- **Java.util.Properties**
- <!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">

```
<!--
  Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
 -->

<!-- DTD for properties -->

<!ELEMENT properties ( comment?, entry* ) >

<!ATTLIST properties version CDATA #FIXED "1.0">

<!ELEMENT comment (#PCDATA) >

<!ELEMENT entry (#PCDATA) >

<!ATTLIST entry key CDATA #REQUIRED>
```

XML et DTD

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE properties (View Source for full doctype...)>
- <properties version="1.0">
    <comment>System.getProperties</comment>
    <entry key="java.runtime.name">Java(TM) SE Runtime Environment</entry>
    <entry key="sun.boot.library.path">D:\jdk1.6\jre\bin</entry>
    <entry key="java.vm.version">17.0-b17</entry>
    <entry key="java.vm.vendor">Sun Microsystems Inc.</entry>
    <entry key="java.vendor.url">http://java.sun.com/</entry>
    <entry key="path.separator">;</entry>
    <entry key="java.vm.name">Java HotSpot(TM) Client VM</entry>
    <entry key="file.encoding.pkg">sun.io</entry>
    <entry key="sun.java.launcher">SUN_STANDARD</entry>
    <entry key="user.country">FR</entry>
    <entry key="sun.os.patch.level">Service Pack 3</entry>
    <entry key="java.vm.specification.name">Java Virtual Machine Specification</entry>
    <entry key="user.dir">F:\progAvanceeNFP121\essaisXML</entry>
```

<!ELEMENT properties (comment?, entry*) >
<!ATTLIST properties version CDATA #FIXED "1.0">
<!ELEMENT comment (#PCDATA) >
<!ELEMENT entry (#PCDATA) >
<!ATTLIST entry key CDATA #REQUIRED>

Déclaration de Listes d'attributs

```
<!ATTLIST NomElement
        NomAttribut1 TypeAttribut1 Mode1
        ...
        NomAttributN TypeAttributN ModeN>
```

- **Type**
  - `CDATA` **: chaîne de caractères prise telle quelle**
  - `ID` **ou** `IDREF` **: clé ou référence à clé**
  - `NMTOKEN` **ou** `NMTOKENS` **: noms symboliques formés de caractères alphanumériques (1 ou plusieurs)**
  - (valeurA |valeurB | ...) **: valeurs de l 'attribut au choix dans la liste**

- **Mode (précise la caractère obligatoire ou non de l 'attribut)**
  - `"valeur"` **: valeur par défaut si non spécifié**
  - `#REQUIRED` **: attribut obligatoirement présent**
  - `#IMPLIED` **: présence de l'attribut facultative**
  - `#FIXED "valeur"` **: l'attribut prend toujours cette valeur**

Exemple

```
<!ELEMENT personne (nom, prenom+, tel?, adresse) >
<!ATTLIST personne
                    num ID,
                    age CDATA,
                    genre (Masculin|Feminin)  "Masculin">

<!ELEMENT auteur (#PCDATA) >
<!ELEMENT editeur (#PCDATA) >
<!ATTLIST auteur
                   genre (Masculin|Feminin ) #REQUIRED
                   ville CDATA #IMPLIED>
<!ATTLIST editeur
                    ville CDATA #FIXED "Paris">
```

DTD et Annales

- http://jfod.cnam.fr/NFP121/annales/2011_fevrier/ExamenFevrier2011.pdf

Le visiteur ***VisiteurToXML*** permet d'obtenir une représentation XML du composite, afin par exemple de rendre persistante une instance du composite, les arbres XML engendrés respectent la DTD suivante :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE taches SYSTEM "taches.dtd">
<!ELEMENT taches (taches|tache)*>
<!ATTLIST taches
            nom  CDATA #REQUIRED
            cout CDATA #REQUIRED>
<!ELEMENT tache (nom,cout)>
<!ELEMENT nom  (#PCDATA) >
<!ELEMENT cout (#PCDATA)>
```

- http://jfod.cnam.fr/NFP121/annales/2013_avril/ExamenAvril2013.pdf

Les arbres XML engendrés par le visiteur ***VisiteurToXML*** doivent respecter la DTD suivante :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE formation SYSTEM "formation.dtd">
<!ELEMENT diplome (UE)*>
<!ATTLIST classe code CDATA #REQUIRED>
<!ATTLIST classe intitule CDATA #REQUIRED>
<!ATTLIST classe nbCredits CDATA #REQUIRED>
<!ELEMENT UE (code,intitule,nbCredits)>
<!ATTLIST code type CDATA #REQUIRED>
<!ATTLIST intitule type CDATA #REQUIRED>
<!ATTLIST nbCredits type CDATA #REQUIRED>
```

Namespaces XML

espace de noms
modularité

Combiner deux documents xml ?

```
<?xml version="1.0" ?>
 <Address>
     <Street>10st.</Street>
     <City>Orange_County</City>
     <State>CA</State>
     <Country>States</Country>
     <PostalCode> CA92657 </PostalCode>
</Address>
```

et:

```
<?xml version="1.0" ?>
<Server>
<Name>OurWebServer</Name>
 <Address>123.45.67.8</Address>
</Server>
```

Solution au conflit

- **Renommer les deux Address dans des noms différents**
  - **Pas général**
  - **Peut poser des problèmes au code déjà écrit**
- **Assigner au deux documents des espaces de nommage différents (modules)**

Solution

```
<addr:Address xmlns:addr="http://www.server.Addresses.com">
    <addr:Street>10st</addr:Street>
    <addr:City>OrangeCounty</addr:City> <addr:State>CA</addr:State>
    <addr:Country>States</addr:Country>
     <addr:PostalCode>CA92657</addr:PostalCode>
</addr:Address>

<serv:Server xmlns:serv="http://www.server.com">
    <serv:Name>OurWebServer</serv:Name>
   <serv:Address> 123.45.67.8 </serv:Address>
 </serv:Server>
```

Declarer les Namespaces

```
<element xmlns:prefix=”Namespace URI”>
```

Optional

Element Attribute Prefix Namespace name

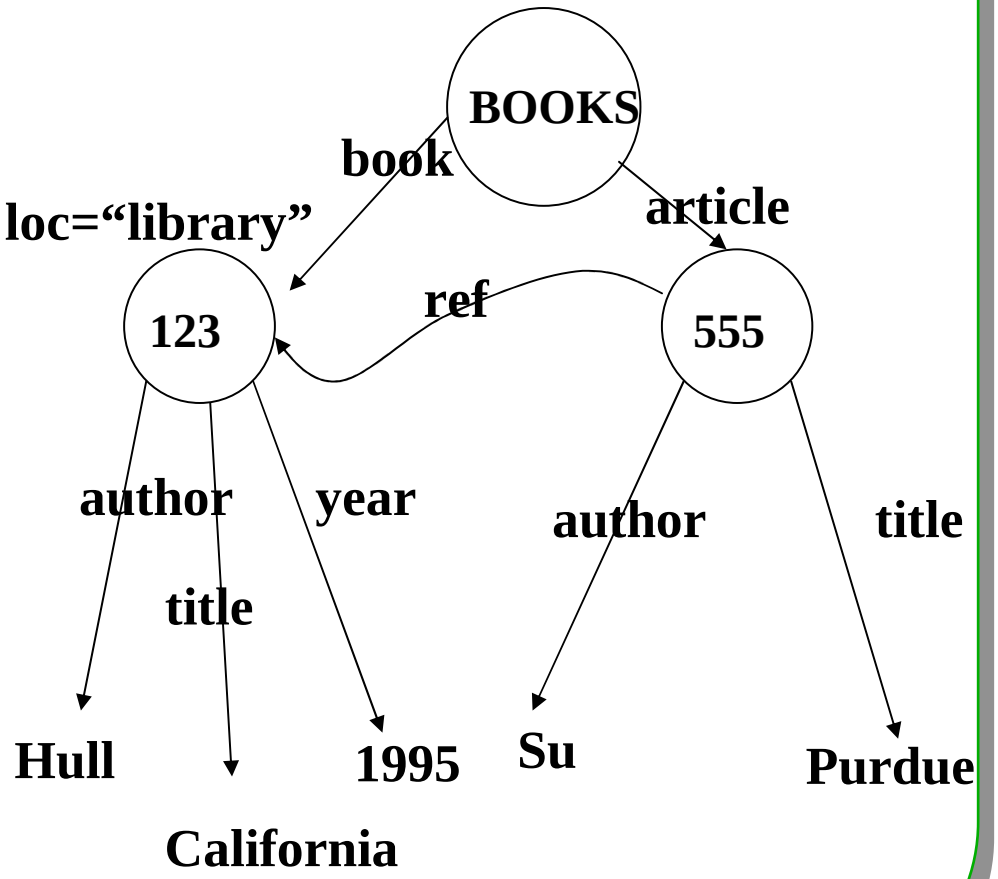
```
<Micro:A xmlns:Micro="http://www.Microsoft.com/">
   <Micro:B>abcd</Micro:B>
 </Micro:A>
```

JDOM

JDOM manipule les éléments d'un Document Object Model spécifique (créé grâce à un constructeur basé sur SAX).

JDOM
Représentation Arborescente des documents XML

```
<BOOKS>
<book id="123" loc="library">
   <author>Hull</author>
   <title>California</title>
   <year> 1995 </year>
</book>
<article id="555" ref="123">
   <author>Su</author>
   <title> Purdue</title>
</article>
</BOOKS>
```



De DOM à JDOM

DOM est l'acronyme de ***Document Object Model***.

API pour modéliser, de parcourir et de manipuler un document XML

DOM fournit une représentation mémoire d'un document XML
sous la forme d'un arbre d'objets et permet la manipulation
(parcours, recherche et mise à jour).

DOM est défini pour être indépendant du langage dans lequel il sera implémenté. DOM n'est donc pas spécifique à Java.

JDOM permet donc de construire des documents, de naviguer dans leur structure, d'ajouter, de modifier, ou de supprimer leur contenu.

Créer un fichier XML avec JDOM

```
<personnes>
   <etudiant classe="P2">
      <nom>CynO</nom>
   <etudiant>
<personnes>
```

JDOM1.java

```
import java.io.*;
import org.jdom.*;
import org.jdom.output.*;
public class JDOM1{
    static Element racine = new Element("personnes");
    static org.jdom.Document document = new Document(racine);

    public static void main(String[] args){
        Element etudiant = new Element("etudiant");
        racine.addContent(etudiant);
        Attribute classe = new Attribute("classe","P2");
        etudiant.setAttribute(classe);
        Element nom = new Element("nom");
        nom.setText("CynO");
        etudiant.addContent(nom);
        affiche();
        enregistre("Exercice1.xml");
    }
}
```

Element

Element

<<create>>+Element(name: String)
+getName(): String
+setName(name: String): Element
+getValue(): String
+isRootElement(): boolean
+getContentSize(): int
+indexOf(child: Content): int
+getText(): String
+getChildText(name: String): String
+setText(text: String): Element
+getContent(): List
+getContent(filter: Filter): List
+removeContent(): List
+removeContent(filter: Filter): List
+setContent(newContent: Collection): Element
+setContent(index: int, child: Content): Element
+setContent(index: int, collection: Collection): Parent
+addContent(str: String): Element
+addContent(child: Content): Element
+addContent(collection: Collection): Element
+addContent(index: int, child: Content): Element
+addContent(index: int, c: Collection): Element
+cloneContent(): List
+getContent(index: int): Content
+removeContent(child: Content): boolean
+removeContent(index: int): Content
+setContent(child: Content): Element
+isAncestor(element: Element): boolean
+getAttributes(): List
+getAttribute(name: String): Attribute
+getAttributeValue(name: String): String
+getAttributeValue(name: String, def: String): String
+setAttributes(newAttributes: List): Element
+setAttribute(name: String, value: String): Element
+setAttribute(attribute: Attribute): Element
+removeAttribute(name: String): boolean
+removeAttribute(attribute: Attribute): boolean
+getDescendants(): Iterator
+getDescendants(filter: Filter): Iterator
+getChildren(): List
+getChildren(name: String): List
+getChild(name: String): Element
+removeChild(name: String): boolean
+removeChildren(name: String): boolean

Créer un élément

Lui affecter un texte

Ajouter un fils

Ajouter un attribut

```
<personnes>
        <etudiant classe="P2">
            <nom>CynO</nom>
        <etudiant>
<personnes>
```

Document

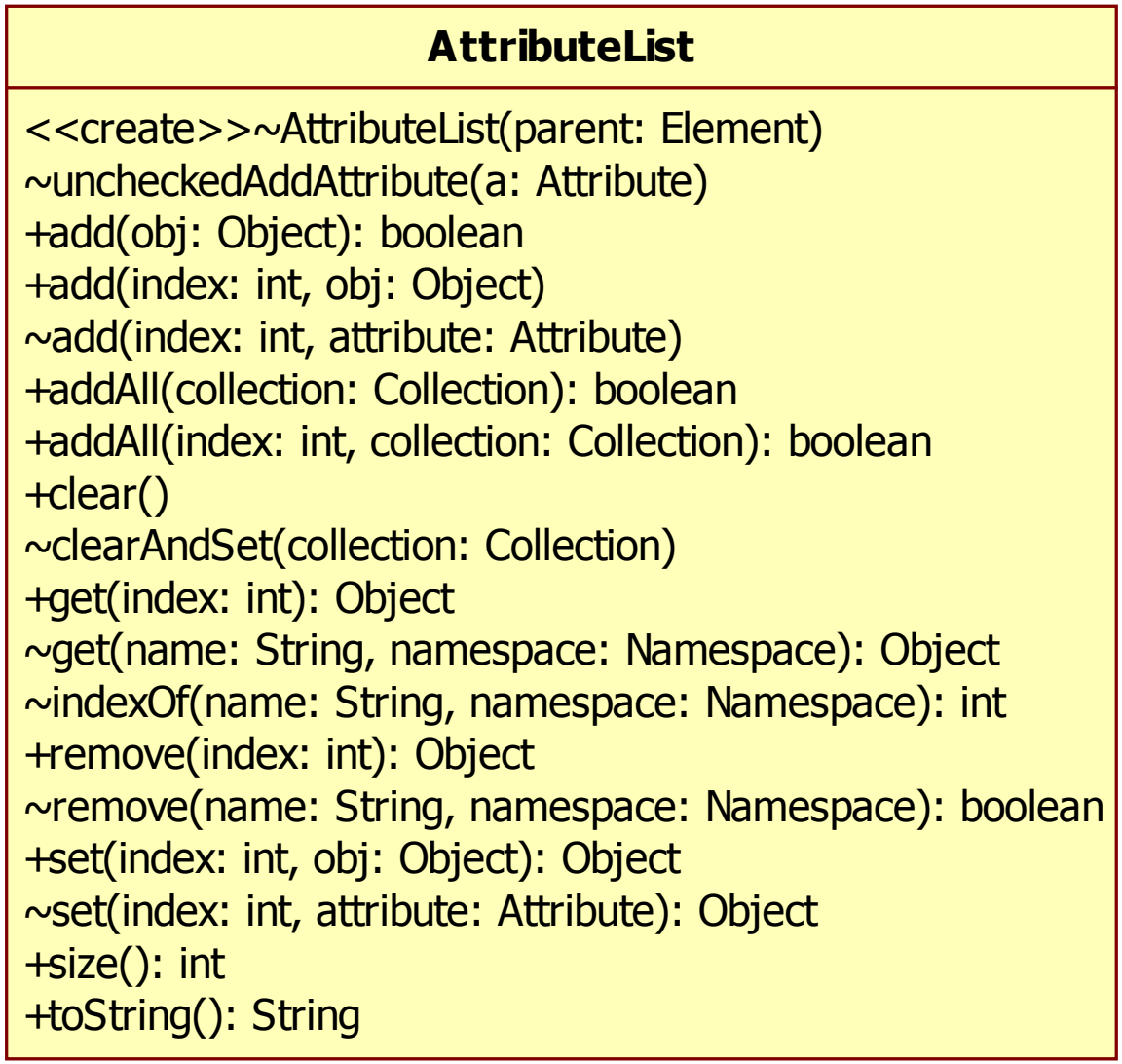
Document

<<create>>+Document()
<<create>>+Document(rootElement: Element, docType: DocType, baseURI: String)
<<create>>+Document(rootElement: Element, docType: DocType)
<<create>>+Document(rootElement: Element)
<<create>>+Document(content: List)
+getContentSize(): int
+indexOf(child: Content): int
+hasRootElement(): boolean
+getRootElement(): Element
+setRootElement(rootElement: Element): Document
+detachRootElement(): Element
+getDocType(): DocType
+setDocType(docType: DocType): Document
+addContent(child: Content): Document
+addContent(c: Collection): Document
+addContent(index: int, child: Content): Document
+addContent(index: int, c: Collection): Document
+cloneContent(): List
+getContent(index: int): Content
+getContent(): List
+getContent(filter: Filter): List
+removeContent(): List
+removeContent(filter: Filter): List
+setContent(newContent: Collection): Document
+setBaseURI(uri: String)
+getBaseURI(): String
+setContent(index: int, child: Content): Document
+setContent(index: int, collection: Collection): Document
+removeContent(child: Content): boolean
+removeContent(index: int): Content
+setContent(child: Content): Document
+toString(): String
+equals(ob: Object): boolean
+hashCode(): int
+clone(): Object
+getDescendants(): Iterator
+getDescendants(filter: Filter): Iterator
+getParent(): Parent
+getDocument(): Document
+setProperty(id: String, value: Object)
+getProperty(id: String): Object

Attribute

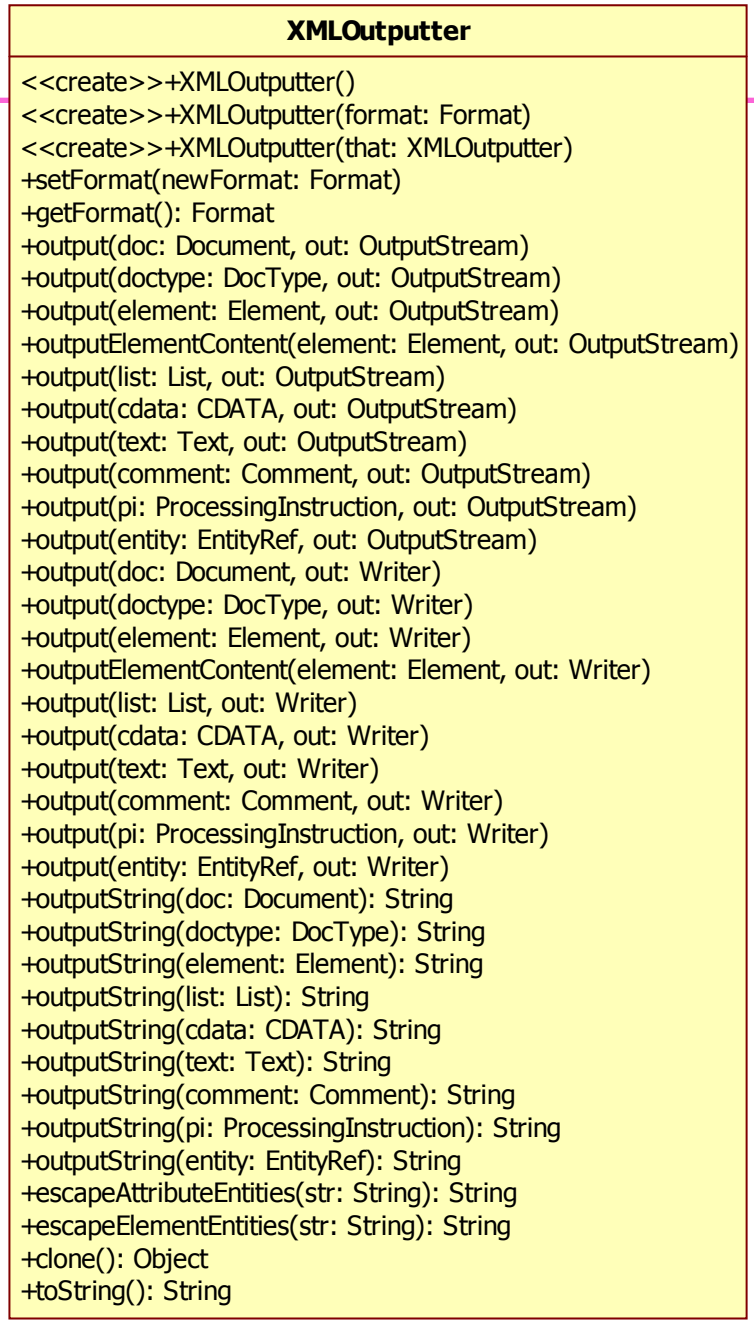
Attribute

<<create>>+Attribute(name: String, value: String, namespace: Namespace)
<<create>>+Attribute(name: String, value: String, type: int, namespace: Namespace)
<<create>>+Attribute(name: String, value: String)
<<create>>+Attribute(name: String, value: String, type: int)
+getParent(): Element
+getDocument(): Document
+detach(): Attribute
+getName(): String
+setName(name: String): Attribute
+getQualifiedName(): String
+getNamespacePrefix(): String
+getNamespaceURI(): String
+getNamespace(): Namespace
+setNamespace(namespace: Namespace): Attribute
+getValue(): String
+setValue(value: String): Attribute
+getAttributeType(): int
+setAttributeType(type: int): Attribute
+toString(): String
+equals(ob: Object): boolean
+hashCode(): int
+clone(): Object
+getIntValue(): int
+getLongValue(): long
+getFloatValue(): float
+getDoubleValue(): double
+getBooleanValue(): boolean


AttributeList
<<create>>~AttributeList(parent: Element)
~uncheckedAddAttribute(a: Attribute)
+add(obj: Object): boolean
+add(index: int, obj: Object)
~add(index: int, attribute: Attribute)
+addAll(collection: Collection): boolean
+addAll(index: int, collection: Collection): boolean
+clear()
~clearAndSet(collection: Collection)
+get(index: int): Object
~get(name: String, namespace: Namespace): Object
~indexOf(name: String, namespace: Namespace): int
+remove(index: int): Object
~remove(name: String, namespace: Namespace): boolean
+set(index: int, obj: Object): Object
~set(index: int, attribute: Attribute): Object
+size(): int
+toString(): String

Éditer le document

```
static void affiche(){
    try{
        XMLOutputter sortie = new XMLOutputter(Format.getPrettyFormat());
        sortie.output(document, System.out);
    }catch (java.io.IOException e){}
}

static void enregistre(String fichier){
    try{
        XMLOutputter sortie = new XMLOutputter(Format.getPrettyFormat());
        //Remarquez qu'il suffit simplement de créer une instance de
        // FileOutputStream avec en argument le nom du fichier
        // pour effectuer la sérialisation.
        sortie.output(document, new FileOutputStream(fichier));
    }catch (java.io.IOException e){}
}
```


XMLOutputter
<<create>>+XMLOutputter()
<<create>>+XMLOutputter(format: Format)
<<create>>+XMLOutputter(that: XMLOutputter)
+setFormat(newFormat: Format)
+getFormat(): Format
+output(doc: Document, out: OutputStream)
+output(doctype: DocType, out: OutputStream)
+output(element: Element, out: OutputStream)
+outputElementContent(element: Element, out: OutputStream)
+output(list: List, out: OutputStream)
+output(cdata: CDATA, out: OutputStream)
+output(text: Text, out: OutputStream)
+output(comment: Comment, out: OutputStream)
+output(pi: ProcessingInstruction, out: OutputStream)
+output(entity: EntityRef, out: OutputStream)
+output(doc: Document, out: Writer)
+output(doctype: DocType, out: Writer)
+output(element: Element, out: Writer)
+outputElementContent(element: Element, out: Writer)
+output(list: List, out: Writer)
+output(cdata: CDATA, out: Writer)
+output(text: Text, out: Writer)
+output(comment: Comment, out: Writer)
+output(pi: ProcessingInstruction, out: Writer)
+output(entity: EntityRef, out: Writer)
+outputString(doc: Document): String
+outputString(doctype: DocType): String
+outputString(element: Element): String
+outputString(list: List): String
+outputString(cdata: CDATA): String
+outputString(text: Text): String
+outputString(comment: Comment): String
+outputString(pi: ProcessingInstruction): String
+outputString(entity: EntityRef): String
+escapeAttributeEntities(str: String): String
+escapeElementEntities(str: String): String
+clone(): Object
+toString(): String

Parser et valider un fichier xml

```
import org.jdom.*;

public class  saxValidate{
  public static void main(String[] args){
    org.jdom.input.SAXBuilder builder =
        new org.jdom.input.SAXBuilder(true);
    try{  Document foo = builder.build("rootBeer.xml");
          System.out.println("XML loaded fine!");
    }
    catch (org.jdom.JDOMException e){
      System.out.println("Error loading XML: " + e.getMessage() );
    }
  }
}
```

exemple

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rootBeer SYSTEM "rootBeer.dtd">
<rootBeer>
  <Foo date="20011021" city="Scottsdale">Look at me.</Foo>
  <Bar>Look at me.</Bar>
</rootBeer>
```

```
C:\dev\topxml\jdom\testcode>java saxValidate
Error loading XML:
Error on line 2 of document file:/C:/dev/topxml/jdom/testcode/roo
tBeer.xml:
External entity not found: "file:/C:/dev/topxml/jdom/testcode/root
Beer.dtd".
```

Avec la DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rootBeer SYSTEM "rootBeer.dtd">
<rootBeer>

    <Foo date="20011021" city="Scottsdale">Look at me.</Foo>
    <Bar>Look at me.</Bar>

</rootBeer>
```

```
<!ELEMENT rootBeer (Foo, Bar) >
<!ELEMENT Foo (#PCDATA) >
<!ELEMENT Bar (#PCDATA) >
<!ATTLIST Foo date CDATA #REQUIRED >
<!ATTLIST Foo city CDATA #IMPLIED >
```

Autres erreurs

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rootBeer SYSTEM "rootBeer.dtd">
<rootBeer>
  <Foo date="20011021" city="Scottsdale">Look at me.</Foo>
  <Bar>Look at me.</Bar>
  <Blug>This is wrong</Blug>
</rootBeer>
```

```
C:\dev\topxml\jdom\testcode>java saxValidate
Error loading XML:
Error on line 6 of document  file:/C:/dev/topxml/jdom/testcode/rootBeer.xml:
Element "rootBeer" allows no further input; "Blug" is not allowed.
```

Modifier un fichier xml

```
<?xml version="1.0" encoding="UTF-8"?>
<personnes>
    <etudiant classe="P2">
        <nom>CynO</nom>
        <prenoms>
            <prenom>Nicolas</prenom>
            <prenom>Laurent</prenom>
        </prenoms>
    </etudiant>
    <etudiant classe="P1">
        <nom>Superwoman</nom>
    </etudiant>
    <etudiant classe="P1">
        <nom>Don Corleone</nom>
    </etudiant>
</personnes>
```

JDom3.java

```
import java.io.*;
…
import java.util.Iterator;
public class JDom3{
static org.jdom.Document document;
static Element racine;
public static void main(String[] args){
      try{
            lireFichier("Exercice 2.xml");
            supprElement("prenoms");
            enregistreFichier("Exercice 2.xml");
      }catch(Exception e){}
}
//On parse le fichier et on initialise la racine de notre arborescence
static void lireFichier(String fichier) throws Exception{
      SAXBuilder sxb = new SAXBuilder();
      document = sxb.build(new File(fichier));
      racine = document.getRootElement();
}
```

suite

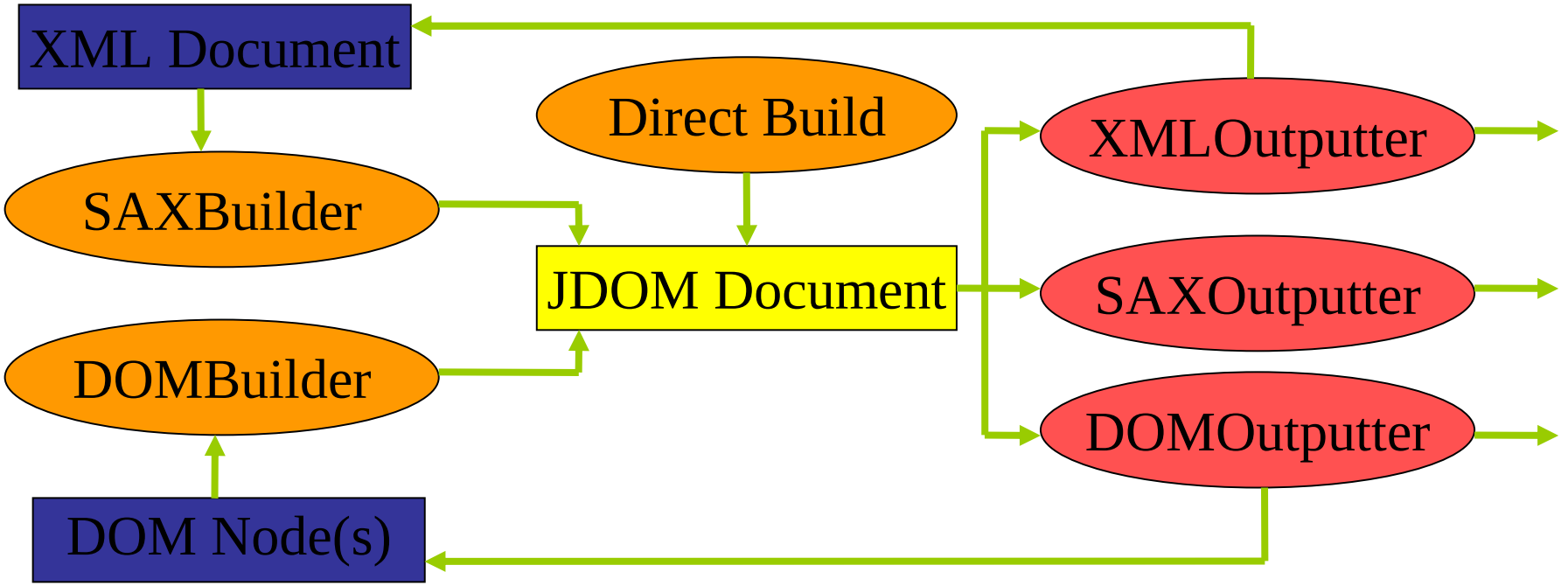
```
static void supprElement(String element){
        List listEtudiant = racine.getChildren("etudiant");
        Iterator i = listEtudiant.iterator();
        while(i.hasNext()){ //On parcours la liste grâce à un iterator
                Element courant = (Element)i.next();
                if(courant.getChild(element)!=null){
                        courant.removeChild(element);
                        courant.setName("etudiant_modifie");
                }
        }
}

static void enregistreFichier(String fichier) throws Exception {
        XMLOutputter sortie = new XMLOutputter(Format.getPrettyFormat());
        sortie.output(document, new FileOutputStream(fichier));
}
}
```

résultat

```
<?xml version="1.0" encoding="UTF-8"?>
<personnes>
    <etudiant_modifie classe="P2">
        <nom>CynO</nom>
    </etudiant_modifie>
    <etudiant classe="P1">
        <nom>Superwoman</nom>
    </etudiant>
    <etudiant classe="P1">
        <nom>Don Corleone</nom>
    </etudiant>
</personnes>
```

Structure générale

- **XML Document -> SAXBuilder -> XMLOutputter**



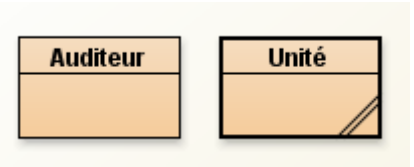
JDOM : Un Document XML

- **Création**
  - **Element e = new Element(“addressbook");**
  - **e.setAttribute(“lang“, “eng");**
  - **Document doc = new Document(e);**

- **Lecture**
  - **Builder builder = new SAXBuilder();**
  - **Document doc = builder.build(url);**

Un exemple … une unité du Cnam, 3 auditeurs

```
public class Unité{


  private String intitulé;          // Attribut JDOM
  private int nombreDeCrédits;      // Attribut JDOM
  private SortedSet<Auditeur> inscrits; // Element(s) JDOM
….

public Element toXML(){
    Element unité = new Element("unite");
    unité.setAttribute("intitule",this.intitulé);
    unité.setAttribute("nombreDeCredits",""+nombreDeCrédits);
    Element liste = new Element("inscrits");
    for(Auditeur a : inscrits){
      liste.addContent(a.toXML());
    }
    unité.addContent(liste);
    return unité;
  }
```



Un exemple … une unité du Cnam, 3 auditeurs

```
public class Auditeur implements Comparable<Auditeur>{
  private String nom;
  private String matricule;

…

  public Element toXML(){
    Element nom = new Element("nom");
    nom.setText(this.nom);
    Element matricule = new Element("matricule");
    matricule.setText(this.matricule);

    Element elt = new Element("auditeur");
    elt.addContent(nom);
    elt.addContent(matricule);
    return elt;
  }
```

Le fichier créé

```
Unité nfp121 = new Unité("NFP121",6);
nfp121.inscrire(new Auditeur("bbb","1234"));
nfp121.inscrire(new Auditeur("aaa","321"));
nfp121.inscrire(new Auditeur("ccc","456"));

Document document = new Document(nfp121.toXML());
XMLOutputter out = new XMLOutputter(Format.getPrettyFormat());
out.output(document,System.out);
out.output(document,new FileOutputStream(new File("nfp121.xml")));
```

```
<?xml version="1.0" encoding="UTF-8" ?>
- <unite intitule="NFP121" nombreDeCredits="6">
  - <inscrits>
    - <auditeur>
        <nom>aaa</nom>
        <matricule>321</matricule>
      </auditeur>
    - <auditeur>
        <nom>bbb</nom>
        <matricule>1234</matricule>
      </auditeur>
    - <auditeur>
        <nom>ccc</nom>
        <matricule>456</matricule>
      </auditeur>
  </inscrits>
</unite>
```

Lecture du fichier, reconstruction de l'arbre

```
SAXBuilder sxb = new SAXBuilder();
Document documentLu = sxb.build(new File("nfp121.xml"));
```

Principe :

La racine est un Element

Element racine = documentLu.getRootElement();

et possède des « enfants »

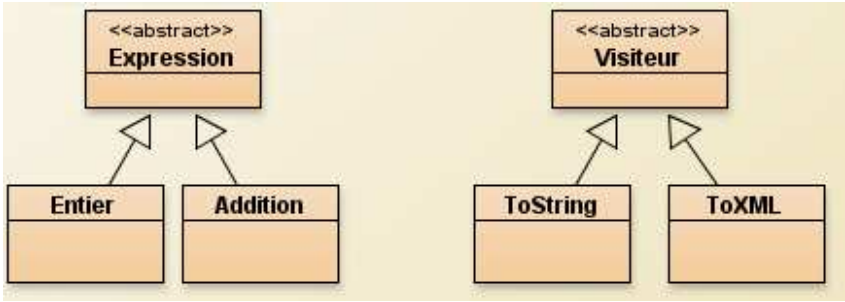
```
   Element inscrits = racine.getChild("inscrits");
   for(Object o :  inscrits.getChildren("auditeur")){
    …
```

Au Complet

```
SAXBuilder sxb = new SAXBuilder();
  Document documentLu = sxb.build(new File("nfp121.xml"));

  out = new XMLOutputter(Format.getPrettyFormat());
  System.out.print("document lu : " );out.output(documentLu, System.out);

   Element racine = documentLu.getRootElement();
   Unité unité = new Unité(racine.getAttribute("intitule").getValue(),
    Integer.parseInt(racine.getAttribute("nombreDeCredits").getValue()));


   Element inscrits = racine.getChild("inscrits");
   for(Object o :  inscrits.getChildren("auditeur")){
     Element elt = (Element)o;
     unité.inscrire(new Auditeur(elt.getChild("nom").getValue(), elt.getChild("matricule").getValue()));
   }
```

Démonstration

Composite + Visiteur




```
public class ToXML extends Visiteur<Element>{
   public Element visiter(Entier e){
      Element elt = new Element("Entier");
      elt.setAttribute(new Attribute("class",e.getClass().getName()));
      elt.addContent(Integer.toString(e.getValeur()));
      return elt;
  }

  public Element visiter(Addition a){
   Element elt = new Element("Addition");
   elt.setAttribute(new Attribute("class",a.getClass().getName()));
   elt.addContent(new Comment("addition et visiteur XML, NFP121"));
   elt.addContent(a.getOp1().accepter(this));
   elt.addContent(a.getOp2().accepter(this));
   return elt;
  }}
```

ici le nom de la classe est un attribut

Composite + Visiteur

- Génération du document

```
Document document = new Document(add.accepter(new toXML()));
XMLOutputter sortie = new XMLOutputter(Format.getPrettyFormat());
sortie.output(document, System.out);
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Addition class="expression.Addition">
  <Addition class="expression.Addition">
    <Entier class="expression.Entier">3</Entier>
    <Entier class="expression.Entier">2</Entier>
  </Addition>
  <Addition class="expression.Addition">
    <Addition class="expression.Addition">
      <Entier class="expression.Entier">3</Entier>
      <Entier class="expression.Entier">2</Entier>
    </Addition>
    <Entier class="expression.Entier">5</Entier>
  </Addition>
</Addition>
```

XMLEncoder, XMLDecoder

- **J2SE et XML**
  - **Classes XMLEncoder et XMLDecoder**
    - **Du paquetage java.beans**
    - **Un Bean, un composant logiciel avec certaines règles d'écriture**
    - **Exemple : Une instance de la classe JFrame et un Bean**

Les Beans et XML

```
public void testXMLBean() throws Exception{
    JFrame b = new JFrame();
    b.setTitle("titre de la JFrame");
    b.setAlwaysOnTop(true);


   XMLEncoder enc = new XMLEncoder(new FileOutputStream("bean.xml"));
    enc.writeObject(b);
    enc.close();

    b = null;
   XMLDecoder dec = new XMLDecoder(new FileInputStream("bean.xml"));
    b = (JFrame)dec.readObject();
    dec.close();
    System.out.println(" b : " + b);
  }
```

Le fichier beans.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
- <java version="1.6.0_02" class="java.beans.XMLDecoder">
  - <object class="javax.swing.JFrame">
    - <void property="alwaysOnTop">
        <boolean>true</boolean>
      </void>
    - <void property="name">
        <string>frame0</string>
      </void>
    - <void property="title">
        <string>titre de la JFrame</string>
      </void>
    </object>
  </java>
```

http://java.sun.com/products/jfc/tsc/articles/persistence4/

Si ce n'est pas un Bean

Pour devenir un Bean, Il faut :

- **Un constructeur par défaut**
- **La classe doit implémenter l'interface Serializable**
- **Toutes les propriétés (attributs) doivent avoir des accesseurs (get/set)**
- **Être une classe publique**

- **Pas de constructeur par défaut ?**
  - **Voir en annexe, utilisation de la classe PersistentDelegate**

Conclusion

- XML incontournable
- XML comme langage
- XML comme configuration
- XML uniforme, description arborescente
- JAXB, et XML est oublié !
  - Java Architecture for XML Binding
  - https://jaxb.dev.java.net/
- JSON …

Annexes

- **XMLEncoder pour des Beans qui n'en sont pas**
- **Une introduction à JAXB**

Une Expression 3+2, XMLEncoder

```
public static void testXMLEncoder(Expression expr,String nomDuFichier) throws Exception{

    XMLEncoder e = new XMLEncoder(new FileOutputStream(nomDuFichier));

    e.setPersistenceDelegate(Entier.class,
                        new PersistenceDelegate() {
                          protected java.beans.Expression instantiate(Object oldInstance,Encoder out) {
                            return new java.beans.Expression(oldInstance, oldInstance.getClass(), "new",
                                                    new Object[]{ Integer.parseInt(oldInstance.toString()) });
                          }
                        });
    e.setPersistenceDelegate(Addition.class,
                        new PersistenceDelegate() {
                          protected java.beans.Expression instantiate(Object oldInstance,Encoder out) {
                            return new java.beans.Expression(oldInstance, oldInstance.getClass(), "new",
                                new Object[]{ ((Addition)oldInstance).op1(), ((Addition)oldInstance).op2() });
                          }
                        });
    e.writeObject(expr);

    e.close();
   }
```

Peu digeste

Une Expression, XMLDecoder

```
public static Expression testXMLDecoder(String nomDuFichier) throws Exception{
    XMLDecoder e = new XMLDecoder(new
                                  FileInputStream(nomDuFichier));
    Object o = e.readObject();
    e.close();
    return ((Expression)o);
  }
```

- **Le test :**

```
testXMLEncoder( new Addition(new Addition(new Entier(3),new Entier(2)),new Entier(5)), "Test3_2.xml");

System.out.println(testXMLDecoder("Test3_2.xml").accepter(new VisiteurToString()));
```

Introduction à JAXB

- **Manipuler du XML en générant automatiquement des classes Java**
  - **SAX ou DOM deviennent des API de bas niveau**
    - **SAX approche évènements**
    - **DOM arborescence**

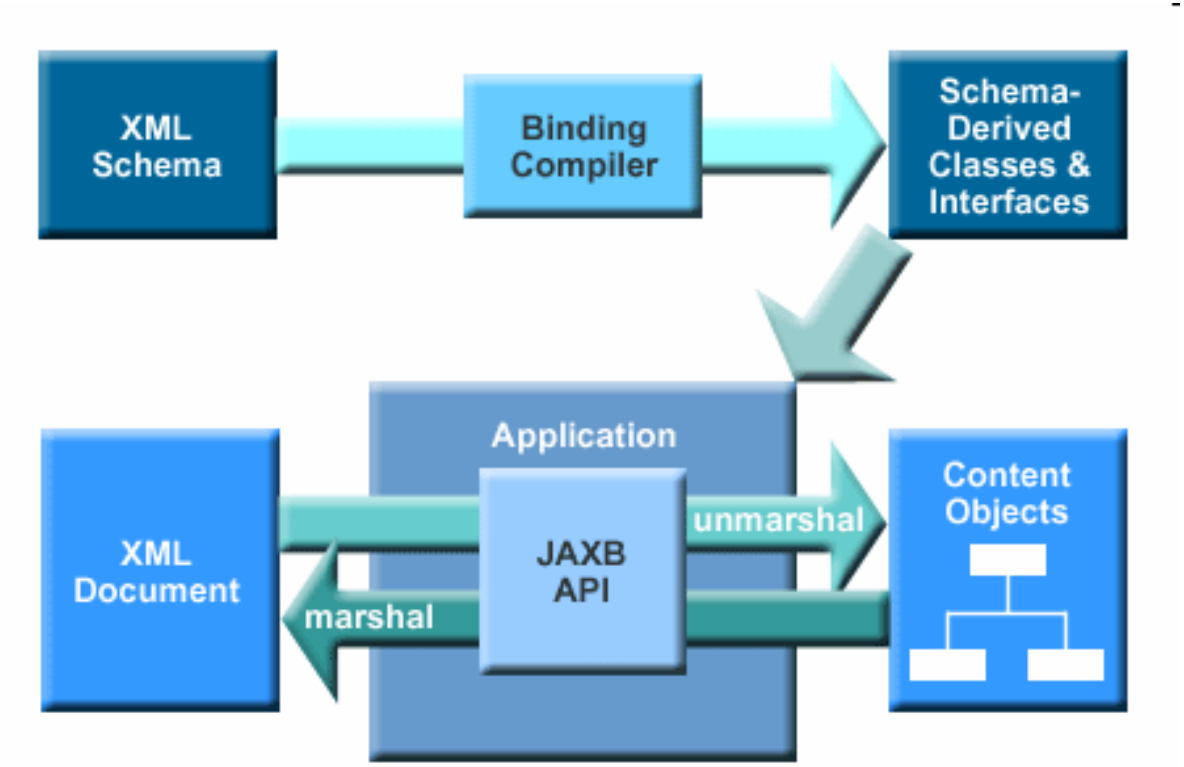
- **Avec JAXB le programmeur Java n'a pas besoin de connaître XML**

- **En standard dans le J2SE 1.6**

- **Adéquation document XML <-> Classes Java**
  - **Sérialisation, désérialisation inclues**
    - **marshal, unmarshall**
  - **Comment : à partir d'un schéma XML,**
    - **une grammaire qui est en XML, plus riche et plus fine que la DTD**

Outils du J2SE6, xjc schema.xsd

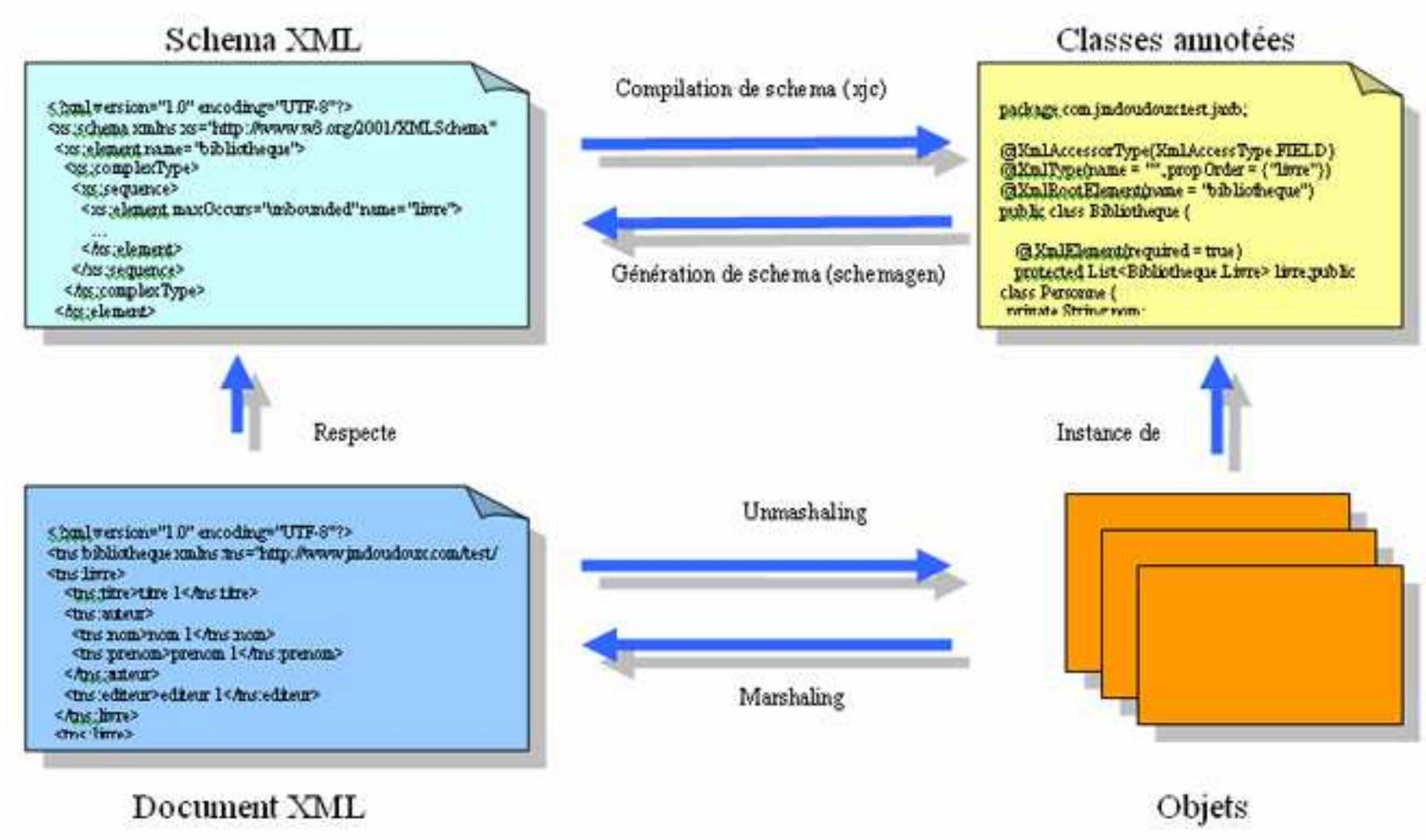
xjc



Extrait de http://www.oracle.com/technetwork/articles/javase/index-140168.html#xmp1

JAXB en standard avec le J2SE, compilateur xjc inclus

Adéquation



Extrait de http://www.jmdoudoux.fr/java/dej/chap037.htm

Exemple 3 + (2 + 5)

- Usage d'un outil sur le web génère un fichier xsd à partir d'un arbre XML, juste pour l'exemple JAXB
  - http://code.google.com/p/jing-trang/
  - java -jar trang-20091111/trang.jar -I xml -O xsd add.xml expr.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<Addition>
  <Entier>3</Entier>
  <Addition>
    <Entier>2</Entier>
    <Entier>5</Entier>
  </Addition>
</Addition>
```

xjc fichier.xsd


```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xs:element name="Addition">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="Addition"/>
        <xs:element ref="Entier"/>
      </xs:choice>
    </xs:complexType>
  </xs:element>
  <xs:element name="Entier" type="xs:integer"/>
</xs:schema>
```

xjc expr.xsd

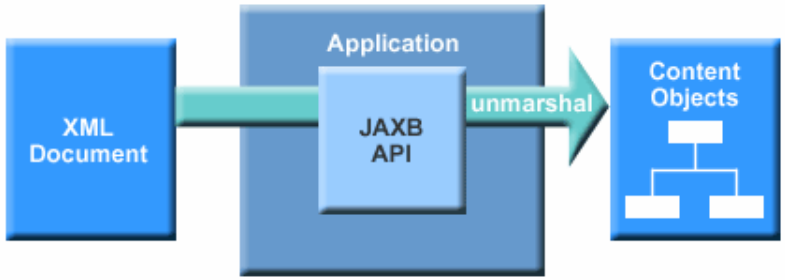
génération de fichiers Java issus de cette grammaire

generated/ObjectFactory.java
generated/Addition.java

Classes Java Annotées, ici Addition.java

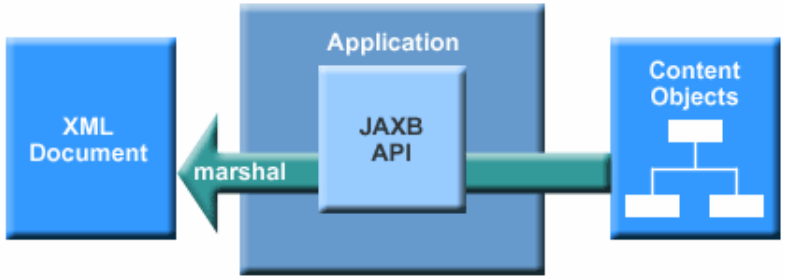

```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "", propOrder = {
    "additionOrEntier"
})
@XmlRootElement(name = "Addition")

public class Addition {

    @XmlElements({
        @XmlElement(name = "Entier", type = BigInteger.class),
        @XmlElement(name = "Addition", type = Addition.class)
    })
    protected List<Object> additionOrEntier;

  public List<Object> getAdditionOrEntier() {
        if (additionOrEntier == null) {
            additionOrEntier = new ArrayList<Object>();
        }
        return this.additionOrEntier;
    }

}
```

unmarshal : instance en XML




```
import javax.xml.bind.*;
import java.io.*;
import java.util.*;
import generated.*;

public class TestJAXB {

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

    JAXBContext jc = JAXBContext.newInstance("generated");
    Unmarshaller unmarshaller = jc.createUnmarshaller();


    Addition expr =
          (Addition)unmarshaller.unmarshal(new File("add.xml"));

    for( Object o : expr.getAdditionOrEntier()){
      System.out.println("o : " + o);
    }
  }
}
```

marshal




```
public class TestJAXB {


public static void main(String[] args) throws Exception{
    JAXBContext jc = JAXBContext.newInstance("generated");
    Unmarshaller unmarshaller = jc.createUnmarshaller();
    Addition expr =
          (Addition)unmarshaller.unmarshal(new File("add.xml"));



   jc = JAXBContext.newInstance("generated");
    Marshaller marshaller = jc.createMarshaller();
    marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, new
  Boolean(true));
    marshaller.marshal(expr,new FileOutputStream("jaxbOutput.xml"));
   }
}
```

Bibliographie XML Schema/JAXB

- **XML Schema Specification**
  - **Part 0: http://www.w3.org/TR/xmlschema-0/**
  - **Part 1: http://www.w3.org/TR/xmlschema-1/**
  - **Part 2: http://www.w3.org/TR/xmlschema-2/**
- **XML.com articles**
  - **http://www.xml.com/pub/a/2001/06/06/schemasimple.html**
- **JAXB articles**
  - http://www.jmdoudoux.fr/java/dej/chap037.htm
  - http://www.oracle.com/technetwork/articles/javase/index-140168.html