SAX et XML

jean-michel Douin, douin au cnam point fr
version : 28 Novembre 2011

Notes de cours

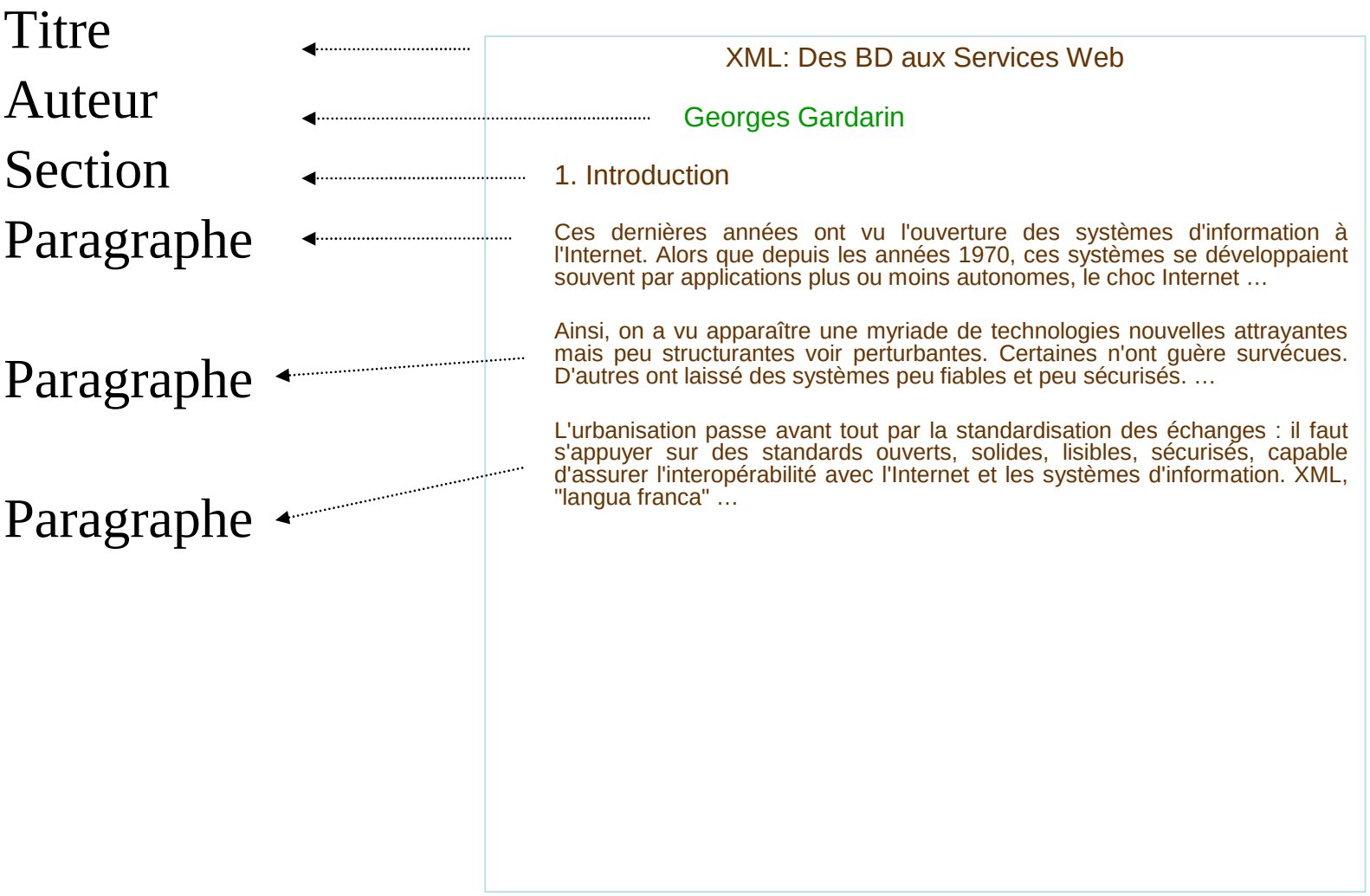
Bibliographie

- Ces notes de cours ont été reprises du cours de J-L Dewez GLG203/2009
- SAX
  - http://www.ibm.com/developerworks/xml/tutorials/x-usax/section4.html

Sommaire

- **XML**
  - **Comme eXtended Markup Language**
  - **SAX**

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>

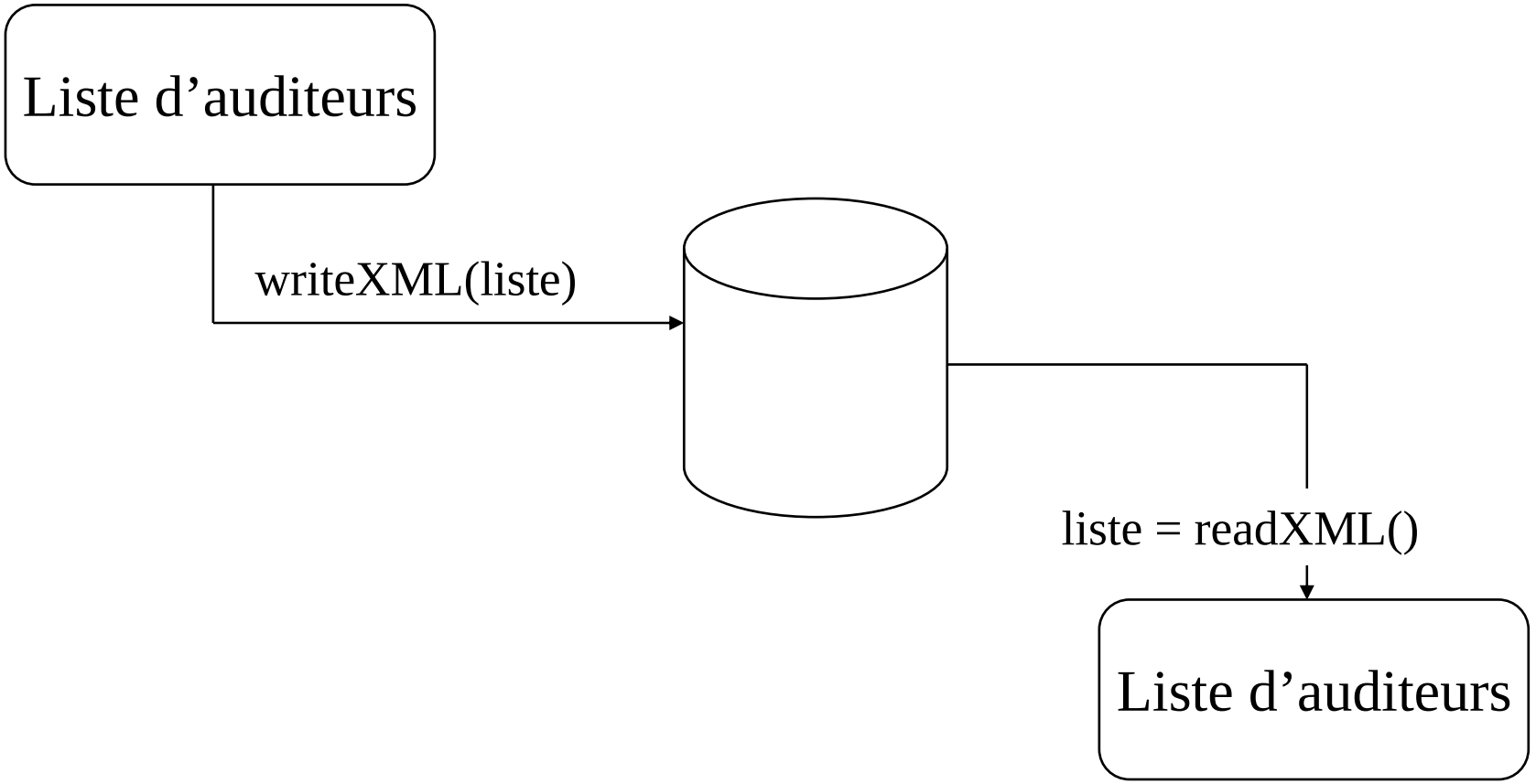
Définir un langage

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

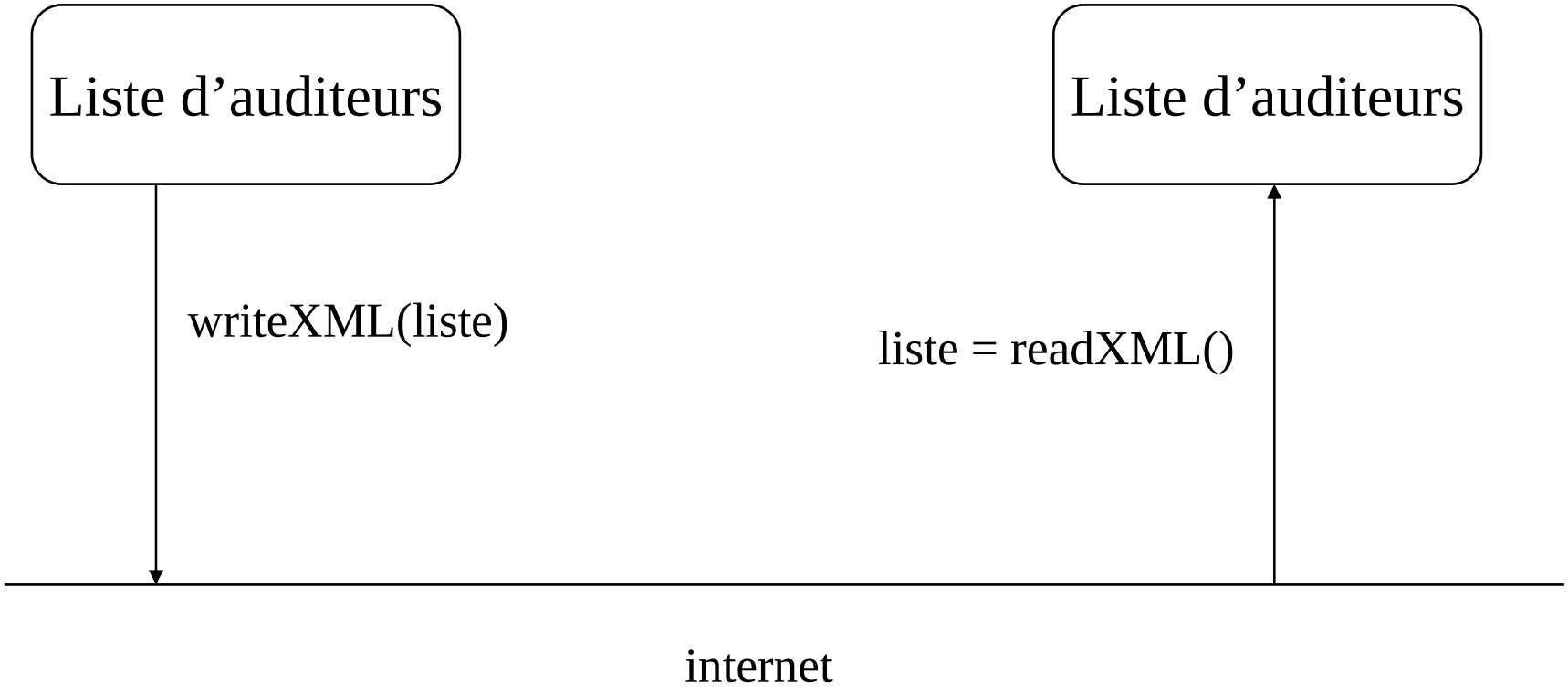
Définir une configuration ici pour Android

```
<?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>
```

Persistance d'un objet et "lisible"



Format d'échange



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>
```

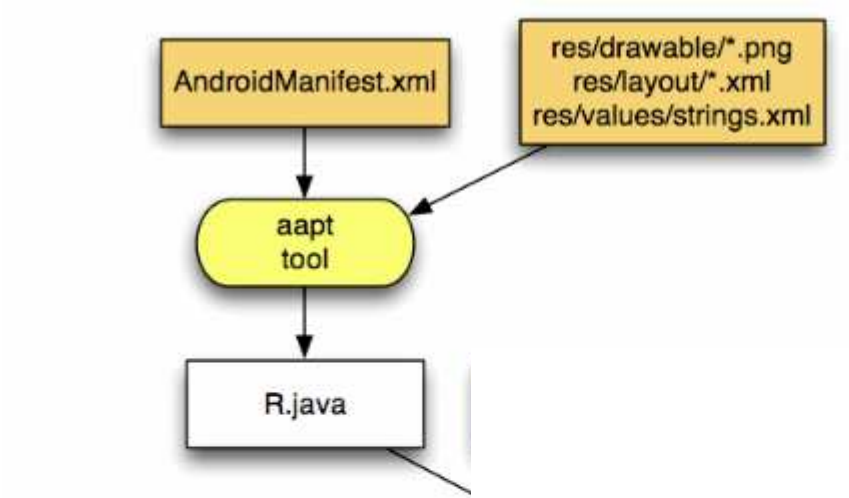
>ant

Buildfile: build.xml

build: [javac] Compiling 1 source file

BUILD SUCCESSFUL Total time: 3 seconds

Exemple 2 : le fichier XML de configuration Android

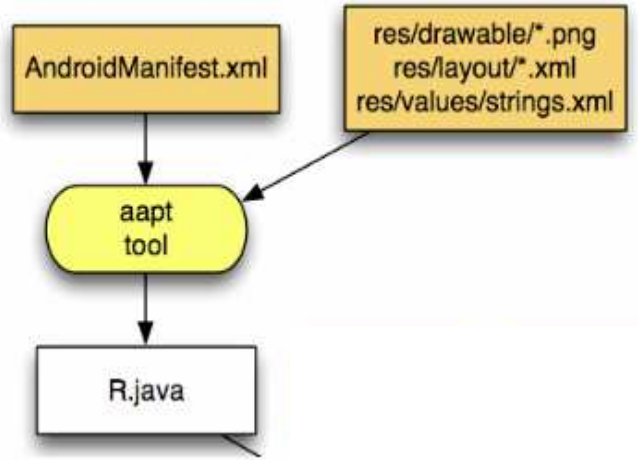


génére un source Java

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** lu par un Navigateur ici ie

Lecture

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

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 (ex. XSL) 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>]]>
```

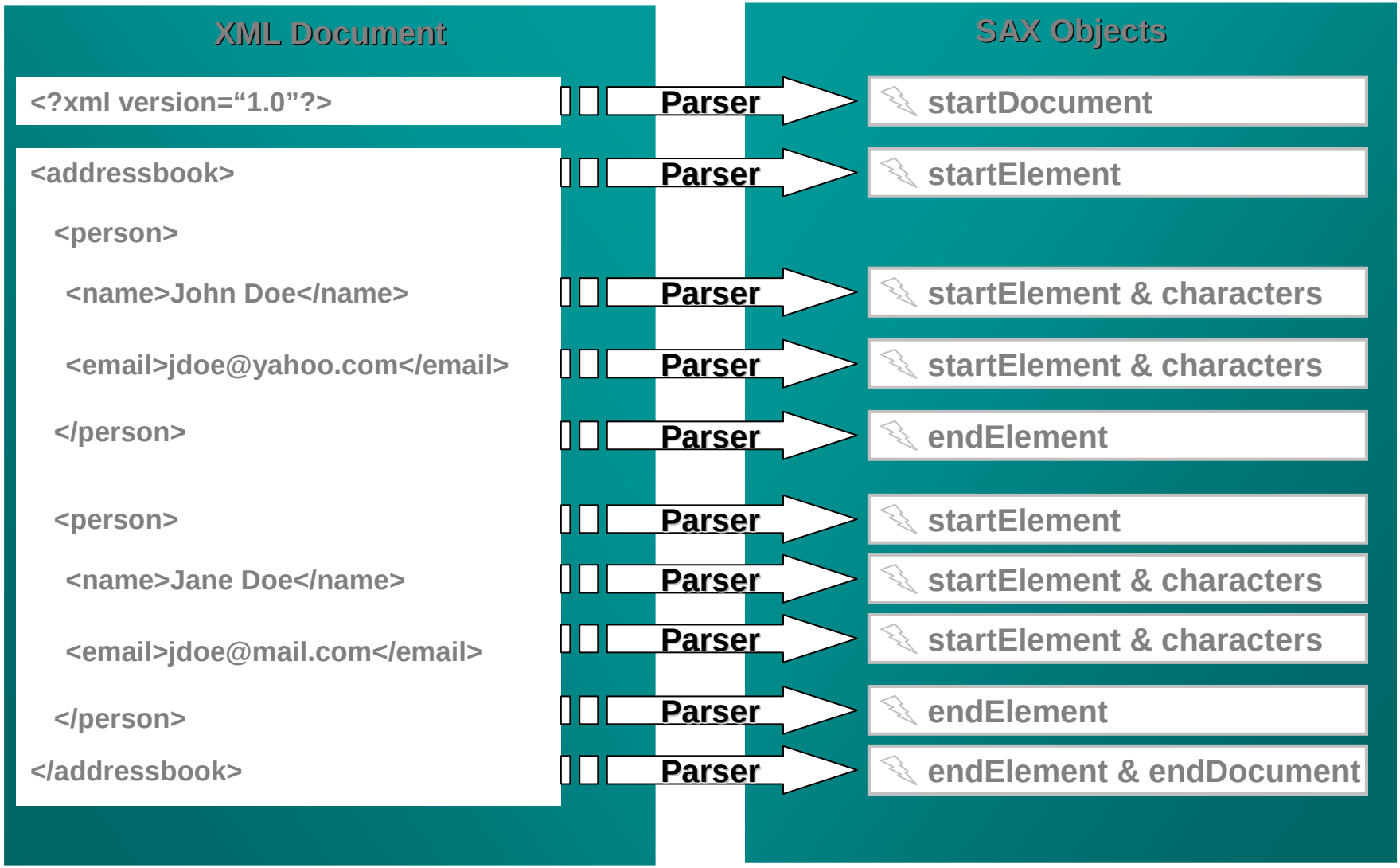
qui se traduit en :

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

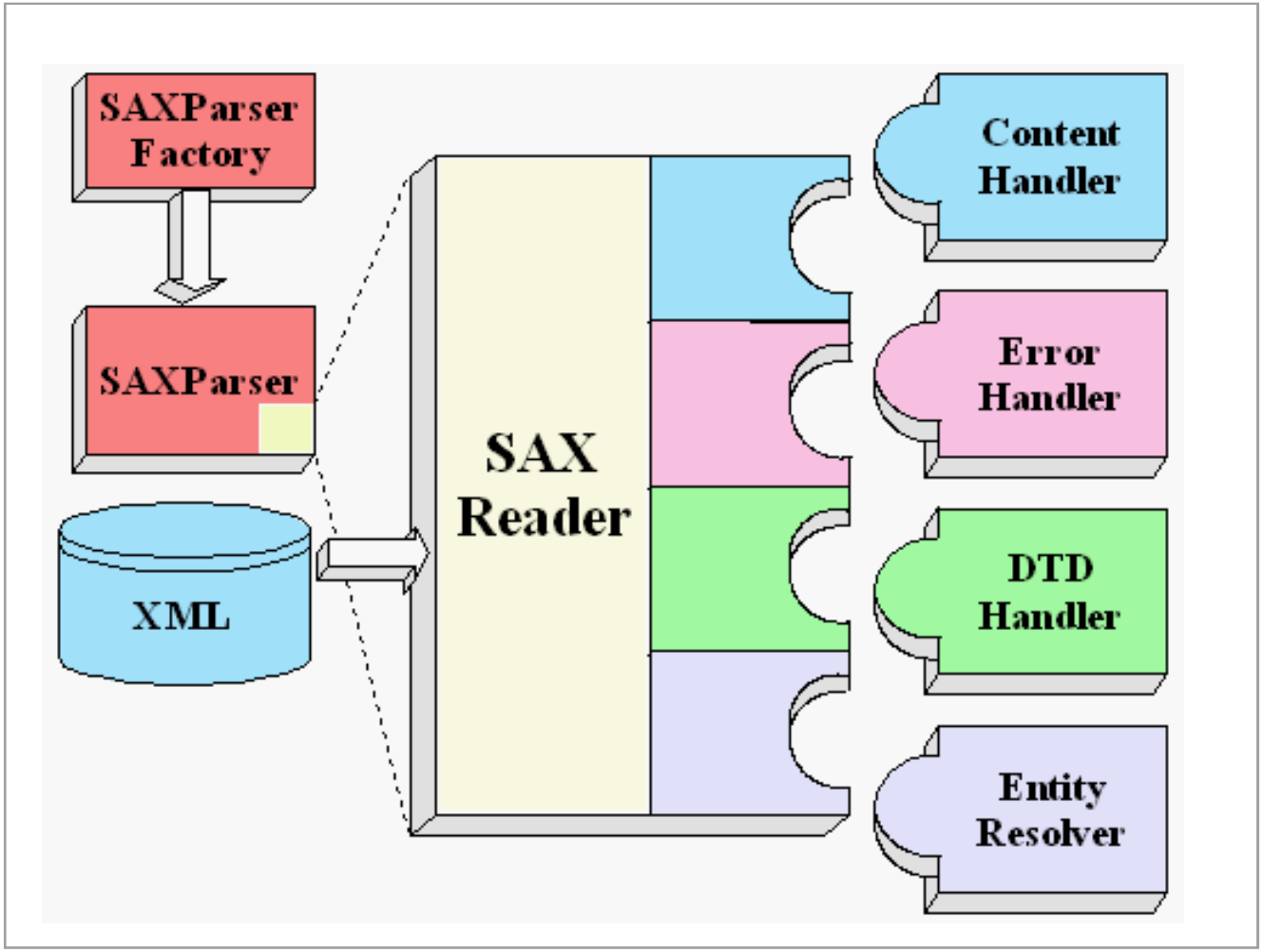
SAX Simple Api for Xml

- 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 implémenter un ContentHandler

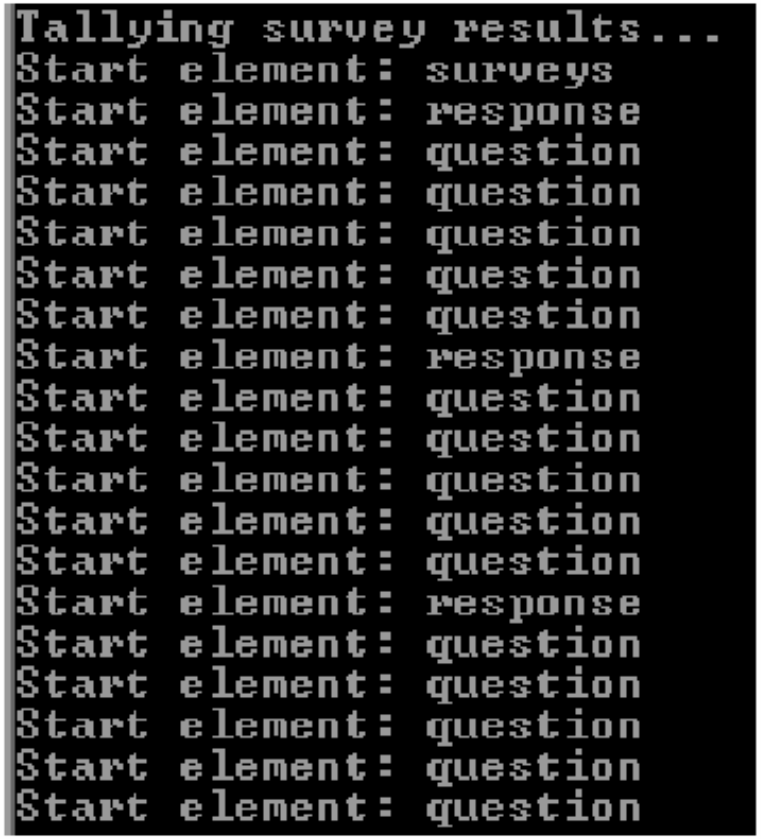
- Méthodes :
  - public void **startDocument**() throws SAXException
  - public void **endDocument**() throws SAXException
  - public void **startElement**(String nspURI, String localName, String qName, Attributes atts) throws SAXException
  - public void **characters**(char[] ch, int start, int length)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



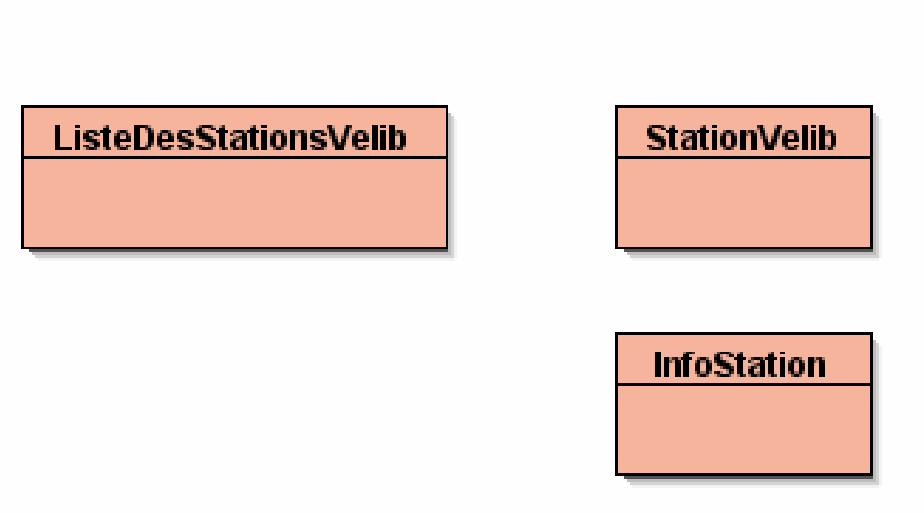
Récupérer les données

```
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 Velib



- **Les présentations**
  - **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}**

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`

surcharge de la méthode startElement

Création d'une instance de la classe StationVelib

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`

surcharge des méthodes startElement, endElement, characters

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")));

   }
```

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 au 55 rue de Turbigo ?**
- **…**

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)
- 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>

- #PCDATA Comme **chaîne de caractère**

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 utilisé 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

Contenu mixte

- **Si un élément peut contenir à la fois des caractères et d'autres éléments Le mot-clé #PCDATA doit être en début de la liste des éléments**

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

- **Les modèles de contenu mixte n'accepte ni de suites d'éléments enfants, ni de choix d'opérateurs de cardinalité. L'ordre utilisé n'a donc pas d'importance sur les éléments suivants #PCDATA**

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>

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**

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>
```

Conclusion