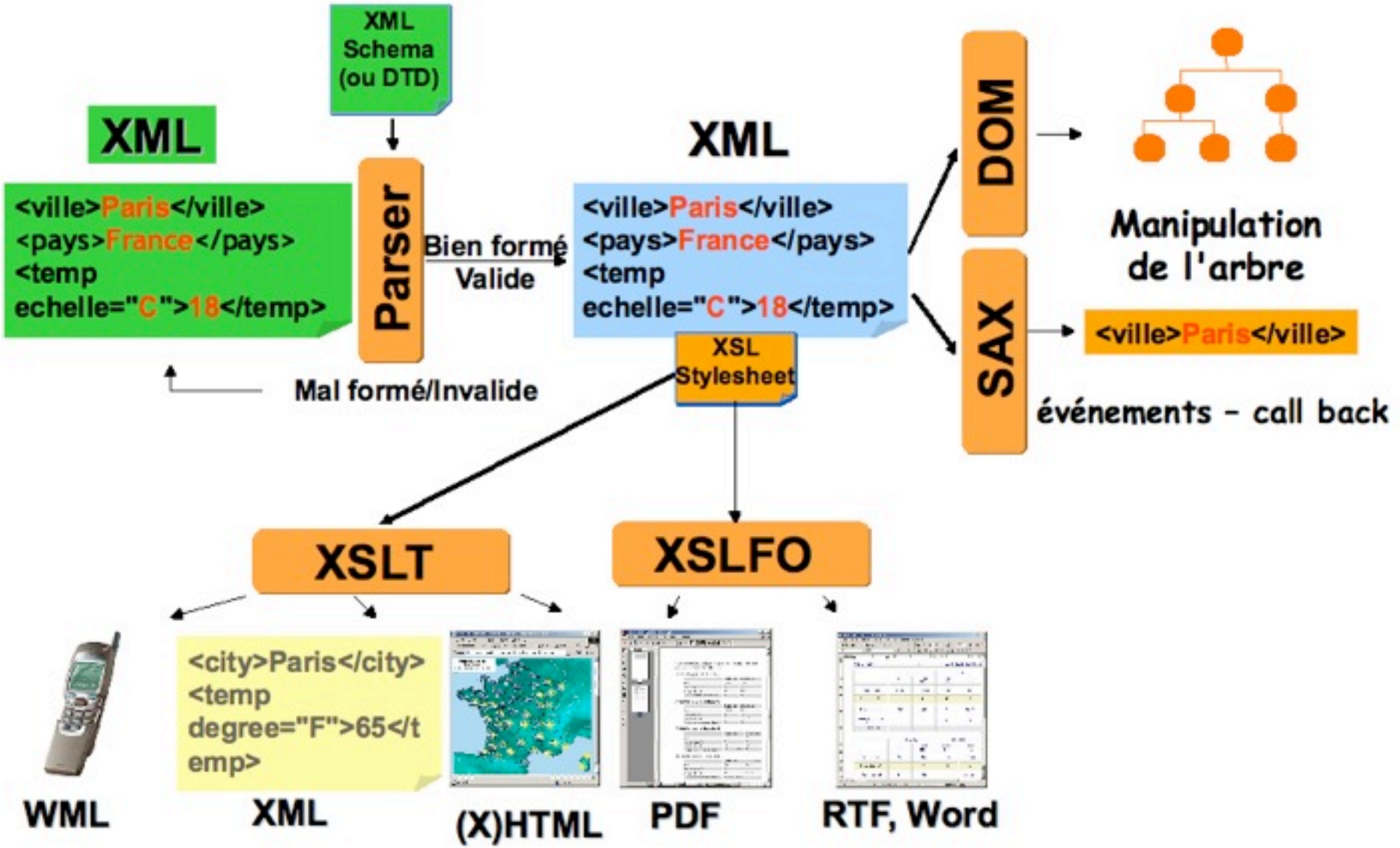
API Java pour XML

XML : traitement par un programme



Un peu de culture

- XML a été développé par le XML Working Group formé sous les auspices du W3C en 1996.
  Il était présidé par Jon Bosak de Sun Microsystems.

  Il y a plus que des liens de parenté entre Java et XML!

- Les classes sax.Counter dans Xerces et Validate dans Xalan sont des exemples (de démonstration) de librairies permettant de parser et valider des documents XML

Aujourd'hui

- Présentation de SAX : une méthode pour lire, parser et tester la validité des documents. C'est une initiative issue de la communauté des programmeurs Java: org.xml.sax

- Présentation de DOM : une spécification W3C permettant de modéliser, parcourir et transformer un document XML

- Data Binding : JAXB, une technique permettant de lier Schémas XML et classes Java.

Intégration : API JAXP

- Java API for XML Parsing
  - JAXP est une API standard J2EE permettant d'effectuer des traitements XML qui englobe :
    - SAX (SAX 2.0)
    - DOM (DOM Level 2)
    - la manipulation de transformations XSLT
- Objectif
  - Fournir une couche d'abstraction permettant d'utiliser n'importe quel analyseur SAX/DOM ou processeur XSL
- JAXP est inclus dans le JDK de Sun

Quelques liens utiles

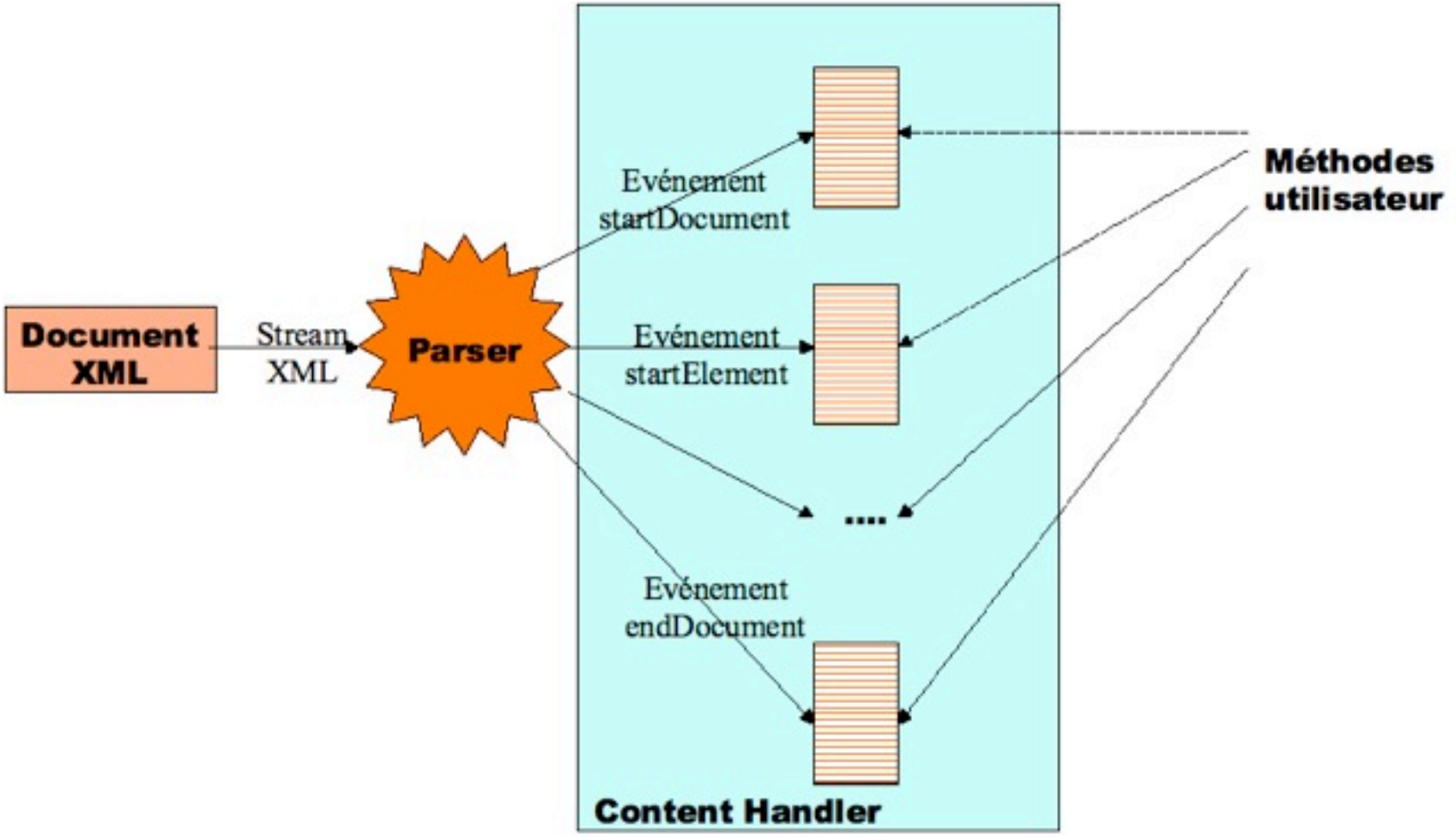
- https://jaxp.dev.java.net/
- www.sun.com/xml
- http://java.sun.com/javase/technologies/core/xml/
- Tutoriel Java :
  http://java.sun.com/j2ee/1.4/docs/tutorial/doc/JAXPIntro.html
- Des exemples Xalan :
  http://xml.apache.org/xalan-j/samples.html
- Un cours en français :
  http://www.jmdoudoux.fr/java/dej/partie4.htm

SAX =
Simple API for XML

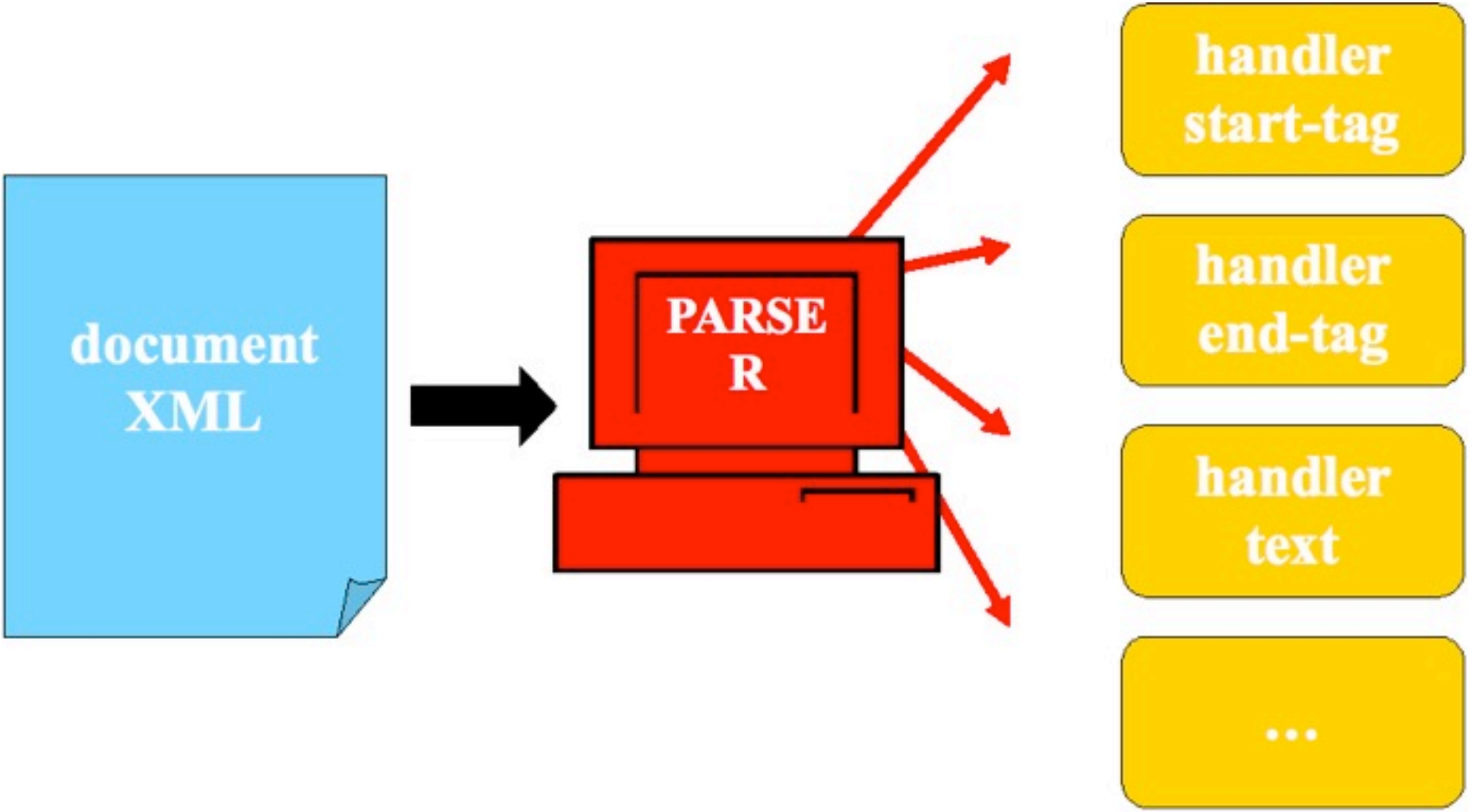
L'Interface SAX

- SAX (Simple API for XML)
  - modèle simplifié d'événements
  - développé par un groupe indépendant.
- Types d'événement :
  - début et fin de document ;
  - début et fin d'éléments ;
  - attributs, texte, … .
- Nombreux parseurs publics
  - XP de James Clark, Aelfred, Saxon
  - MSXML3 de Microsoft
  - Xerces de Apache
  - JAXP de SUN

Principe de fonctionnement



Aperçus de SAX



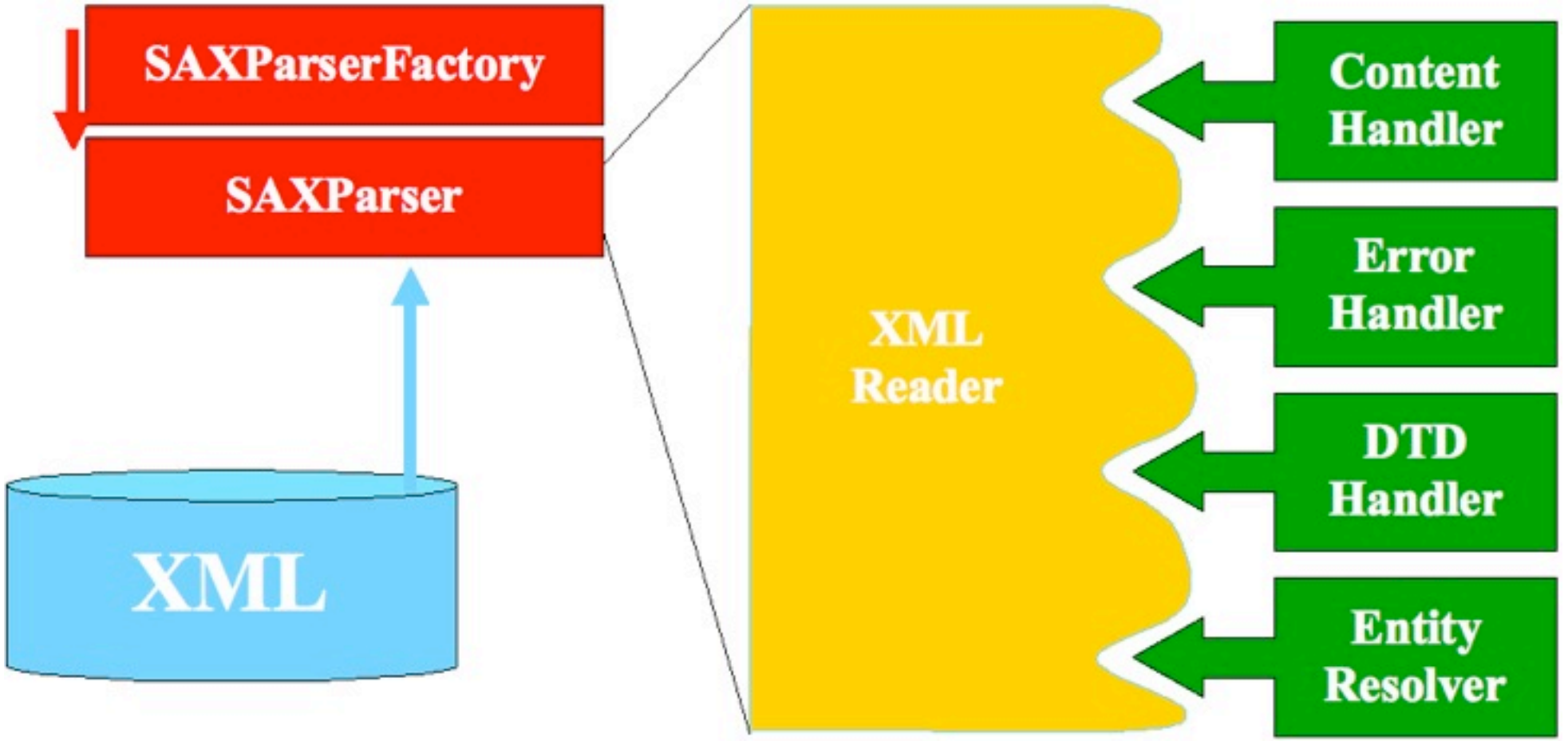
SAX

- Particularité du parser SAX:
  - Event-driven: on a besoin de fournir les fonctions répondant à l'envoi d'évènements
  - Le parcours des documents est séquentiel (sequential read-only access)
  - On accède une seule fois à chaque partie d'un document (one-time access)
- C'est une approche rapide et simple:
  - On n'a pas besoin de charger un document entièrement en mémoire pour travailler
  - Par contre on ne peut pas modifier un document dynamiquement

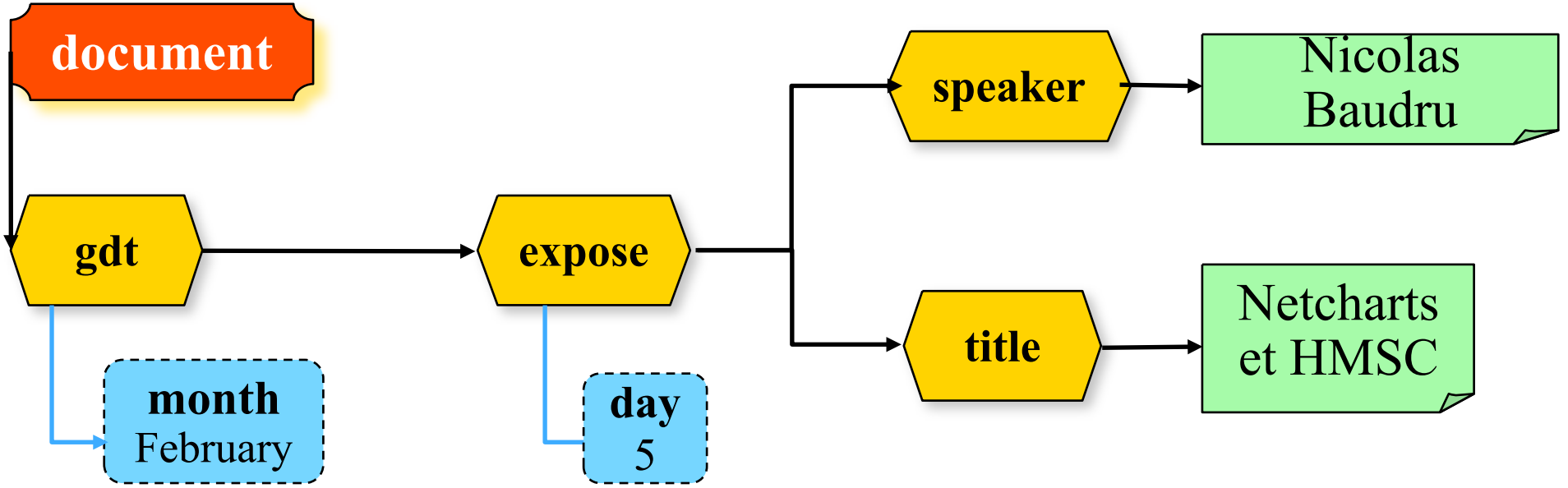
Les méthodes essentielles

- XMLReader
  - setContentHandler
  - setErrrorHandler
  - parse
- ContentHandler
  - startDocument
  - endDocument
  - startElement
  - endElement
  - characters
- ErrorHandler
  - fatalError
  - error
  - warning

Les composants clefs de l'API



Déclenchement des évènements



start document ; start element gdt ; attribute[0] month:February
; start element expose ; attribute[0] day:5
; start element speaker ; characters: Nicolas Baudru
; end element speaker ; …
; end element expose ; …
; end element gdt ; end document

Où retrouve-t-on SAX ?

- Sun's JAXP 1.1 : JAXP supporte SAX 2.0 et le package SAX2-ext utilise Crimson (le “parseur” Java) comme parseur XML par défaut
- Le projet Xerces de la fondation Apache XML inclus un driver SAX 2.0 natif
- SAXON inclus une version modifiée de AElfred (un autre parser XML) qui est aussi conforme à SAX 2.0
- Oracle's XDK

Liste des parseurs les plus utilisés

- **Apache Xerces**: http://xml.apache.org
- IBM XML4J: http://alphaworks.ibm.com/tech/xml4j
- James Clark's XP: http://www.jclark.com/xml/xp
- Oracle XML Parser: http://technet.oracle.com/tech/xml
- Sun Microsystems Crimson: http://xml.apache.org/crimson
- The Mind Electric's Electric XML: http://www.themindelectric.com/products/xml/xml.html
- Microsoft's MXSML Parser: http://msdn.microsoft.com/xml/default.asp

- Chacun de ces parseurs fournit une implémentation de SAX

Packages

- Les classes et interfaces de SAX sont définies dans les packages suivants:
  - javax.xml.parsers
  - org.xml.sax
  - org.xml.sax.helpers

Package javax.xml.parsers

- C'est la librairie qui contient les classes permettant de manipuler les documents XML. On peut les « connecter » à deux types possibles de parseurs :
  - SAX (Simple API for XML)
  - DOM (Document Object Model)
- SAXParserFactory : définit une « usine » qui permet aux applications de configurer un parseur SAX
- SAXParser : définit l'API. Encapsule l'implémentation de la classe XMLReader

- DocumentBuilderFactory : définit une usine qui permet aux applications d'obtenir un parser qui produit une représentation sous forme DOM d'un document
- DocumentBuilder : définit les classes permettant d'obtenir une instance DOM à partir d'un document XML

Un premier exemple de programme

```
$> java -cp . SimpleSaxParser wg.xml
Start Document: file:…/wg.xml
debut element: gdt
 attribut[0]: month=February
 attribut[1]: year=2004
debut element: expose
 attribut[0]: day=5
debut element: speaker
texte: --Nicolas Baudru--
fin element: speaker
debut element: title
texte: --Netcharts et HMSC--
fin element: title
…
```

Exemple de programme

```
import javax.xml.parsers.SAXParserFactory;
import javax.xml.parsers.SAXParser;
import org.xml.sax.XMLReader;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;
import org.xml.sax.ErrorHandler;
import org.xml.sax.Attributes;
import org.xml.sax.helpers.DefaultHandler;
import java.util.*;
import java.io.*;

public class SimpleSaxParser extends DefaultHandler {
 private String currentElement = new String();
 private static String urlToParse;

 static public void main(String[] args){
  try{
// 1. Creation d'un JAXP SAXParserFactory et configuration
    SAXParserFactory factory = SAXParserFactory.newInstance();
// si on veut valider le document par rapport a
// sa declaration de type :
// factory.setValidating(true);

// 2. Creation d'un JAXP SAXParser
    SAXParser saxParser = factory.newSAXParser();

// 3. Obtention du XMLReader
    XMLReader xmlReader = saxParser.getXMLReader();

// 4. Affectation du ContentHandler pour le XMLReader.
// Remarque: on utilise la classe courante
    xmlReader.setContentHandler(new SimpleSaxParser());

// 5. Affectation du ErrorHandler avant parsing
    xmlReader.setErrorHandler(new MyErrorHandler(System.err));
```

```
// 6. On parse le document (en donnant son URL) en utilisant
// le XMLReader
urlToParse = convertToFileURL(args[0]);
xmlReader.parse(urlToParse);
} catch (Throwable t) { t.printStackTrace(); }
   System.exit(0);
}
// debut du document
public void startDocument() throws SAXException {
   System.out.println("Start Document: "+urlToParse);
 }

// debut de l'element
 public void startElement(String namespaceURI,
   String localName, // local name
   String qName,   // qualified name
   Attributes atts) throws SAXException {
// recuperation du nom de l'element
   String eltName = qName;
   System.out.print("debut element: "+ eltName+"\n");
   for (int i=0; i < atts.getLength(); i++) {
    // recuperation du nom de l'attribut et de sa valeur
     String attName = atts.getQName(i);
     if ("".equals(attName)) attName = atts.getQName(i);
     System.out.print("\tattribut["+i+"]: "+ attName + "=" +
                        atts.getValue(i)+"\n");
    }
  }
 // Pour les noeuds textes
 public void characters (char[] ch, int start, int length)
  {
        String text = new String (ch, start, length);
        System.out.print("texte: --"+text+"--\n");
  }
etc.
```

Exemple de programme

```
import javax.xml.parsers.SAXParserFactory;
import javax.xml.parsers.SAXParser;
import org.xml.sax.XMLReader;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;
import org.xml.sax.ErrorHandler;
import org.xml.sax.Attributes;
import org.xml.sax.helpers.DefaultHandler;
import java.util.*;
import java.io.*;

public class SimpleSaxParser extends DefaultHandler {
 private String currentElement = new String();
 private static String urlToParse;
```

```
 static public void main(String[] args){
  try{
// 1. Creation d'un JAXP SAXParserFactory et configuration
    SAXParserFactory factory = SAXParserFactory.newInstance();
// si on veut valider le document par rapport a
// sa declaration de type :
// factory.setValidating(true);

// 2. Creation d'un JAXP SAXParser
    SAXParser saxParser = factory.newSAXParser();

// 3. Obtention du XMLReader
    XMLReader xmlReader = saxParser.getXMLReader();

// 4. Affectation du ContentHandler pour le XMLReader.
// Remarque: on utilise la classe courante
    xmlReader.setContentHandler(new SimpleSaxParser());

// 5. Affectation du ErrorHandler avant parsing
    xmlReader.setErrorHandler(new MyErrorHandler(System.err));
```

```
// 6. On parse le document (en donnant son URL) en utilisant
// le XMLReader
urlToParse = convertToFileURL(args[0]);
xmlReader.parse(urlToParse);
} catch (Throwable t) { t.printStackTrace(); }
   System.exit(0);
}
```

```
// debut du document
public void startDocument() throws SAXException {
   System.out.println("Start Document: "+urlToParse);
 }
```

```
// debut de l'element
 public void startElement(String namespaceURI,
   String localName, // local name
   String qName,   // qualified name
   Attributes atts) throws SAXException {
// recuperation du nom de l'element
   String eltName = qName;
   System.out.print("debut element: "+ eltName+"\n");
   for (int i=0; i < atts.getLength(); i++) {
    // recuperation du nom de l'attribut et de sa valeur
     String attName = atts.getQName(i);
     if ("".equals(attName)) attName = atts.getQName(i);
     System.out.print("\tattribut["+i+"]: "+ attName + "=" +
                      atts.getValue(i)+"\n");
    }
  }
```

```
 // Pour les noeuds textes
 public void characters (char[] ch, int start, int length)
  {
        String text = new String (ch, start, length);
        System.out.print("texte: --"+text+"--\n");
  }
```

etc.

Squelette du programme: main()

```
// 1. Creation d'un SAXParserFactory et configuration
SAXParserFactory usine = SAXParserFactory.newInstance();
// 2. Creation d'un JAXP SAXParser
SAXParser saxParser = usine.newSAXParser();
// 3. Obtention du XMLReader
XMLReader xmlReader = saxParser.getXMLReader();
// 4. Affectation du ContentHandler pour le XMLReader
xmlReader.setContentHandler(new SimpleSaxParser());
// 5. Affectation du ErrorHandler avant parsing
xmlReader.setErrorHandler(new MyErrHand(System.err));
// 6. On parse le document
xmlReader.parse(url-du-document);
```

La classe SAXParserFactory

- javax.xml.parsers.SAXParserFactory est la classe qui permet de configurer et d'obtenir un parseur SAX

- Méthodes importantes de la classe :
  - `SAXParserFactory newInstance()` : permet de créer une nouvelle instance, pouvant se baser sur les "properties" de la machine
  - `SAXParser newSAXParser()` : crée un parseur à partir du factory en utilisant les paramètres courants
  - `void setNamespaceAware(boolean awareness)` : spécifie que le parseur fournit un support pour l'utilisation de namespaces
  - `void setValidating(boolean validating)` : spécifie que le parseur doit valider le document XML

La classe SAXParser

- javax.xml.parsers.SAXParser est la classe qui encapsule la création des objets XMLReader

- Méthodes importantes de la classe :
  - `XMLReader getXMLReader()` : retourne l'objet tant convoité
  - `boolean isValidating()` : spécifie si le parseur doit valider le document XML
  - `void parse(java.io.File f, DefaultHandler dh)` : parse le contenu du fichier (nécessite la description d'un DefaultHandler)

Le package org.xml.sax

- `ContentHandler`
- `DTDHandler`
- `EntityResolver`
- `ErrorHandler`
- `XMLFilter`
- `XMLReader`

L'interface XMLReader (org.xml.sax)

- C'est l'interface qui décrit la méthode pour parcourir les documents XML en utilisant des callbacks
- Deux améliorations par rapport à l'interface Parser de SAX 1.0 (deprecated)
  - elle offre un cadre plus uniforme pour interroger et modifier les propriétés du parser
  - elle permet de gérer les namespaces
- Une classe qui implémente XMLReader doit fournir les méthodes pour enregistrer les callbacks, initier le parsing, configurer les propriétés (system properties)…

Features

- General Features
  - http://xml.org/sax/features/validation
  - http://xml.org/sax/features/namespaces
  - http://apache.org/xml/features/validation/schema
  - http://apache.org/xml/features/continueafter-fatal-error
- SAX Features
  - http://xml.org/sax/features/namespaceprefixes
- Par exemple:

```
XMLReader  reader =  saxParser.getXMLReader();
reader.setFeature("http://xml.org/sax/features/validation", true);
```

L'interface ContentHandler (org.xml.sax)

- C'est l'interface principale qui est implémentée par la plupart des applications utilisant SAX :

  dans notre exemple la classe `SimpleSaxParser` étend la classe `DefaultHandler` qui implémente l'interface `ContentHandler`

- C'est cette interface qui reçoit notification du contenu logique des documents

- **Remarque**: il existe une classe `ContentHandler` dans le package java.net, il vaut mieux donc éviter d'importer ce package en même temps que org.xml.sax

org.xml.sax.ContentHandler

- `void startDocument()` : callback pour le début du document
- `void endDocument()`
- `void startElement(String namespaceURI, String localName, String qName, Attributes atts)`
- `void endElement(`…*idem*…`)`
- `void characters(char[] ch, int start, int length)`
- `endDocument(), processingInstruction(), setDocumentLocator(), skippedEntity(), ignorableWhitespace()`

Exemple 1

```
<?xml version="1.0“
      encoding="ISO-8859-1"?>
<?xml-stylesheet type='text/css'
      href='person.css'?>
<!DOCTYPE person SYSTEM "person.dtd">

<person xmlns="http://xml.ns.org/">
<n:name xmlns:n="http://xml.ns.org/n">
   <n:first>Sydney</n:first>
   <n:last>Lee</n:last>
</n:name>
<assignment projid="p2"/>
</person>
```

```
setDocumentLocator(Locator
   locator)

locator: org.apache.xerces.
   readers.DefaultEntityHandle
   r@1f953d
```

Exemple 2

```
<?xml version="1.0"
      encoding="ISO-8859-1"?>
<?xml-stylesheet type='text/css'
      href='person.css'?>
<!DOCTYPE person SYSTEM "person.dtd">

<person xmlns="http://xml.ns.org/">
<n:name xmlns:n="http://xml.ns.org/n">
   <n:first>Sydney</n:first>
   <n:last>Lee</n:last>
</n:name>
<assignment projid="p2"/>
</person>
```

```
startDocument( )
```

Exemple 3

```
<?xml version="1.0"
      encoding="ISO-8859-1"?>
<?xml-stylesheet type='text/css'
      href='person.css'?>
<!DOCTYPE person SYSTEM "person.dtd">

<person xmlns="http://xml.ns.org/">
<n:name xmlns:n="http://xml.ns.org/n">
   <n:first>Sydney</n:first>
   <n:last>Lee</n:last>
</n:name>
<assignment projid="p2"/>
</person>
```

```
processingInstruction(
   String target,
   String data)

target: "xml-stylesheet"

data: "type='text/css'
   href='person.css'"
```

Exemple 4

```
<?xml version="1.0“
      encoding="ISO-8859-1"?>
<?xml-stylesheet type='text/css'
      href='person.css'?>
<!DOCTYPE person SYSTEM "person.dtd">

<person xmlns="http://xml.ns.org/">
<n:name xmlns:n="http://xml.ns.org/n">
   <n:first>Sydney</n:first>
   <n:last>Lee</n:last>
</n:name>
<assignment projid="p2"/>
</person>
```

```
startPrefixMapping(
   String prefix,
   String namespaceURI)

prefix: ""

namespaceURI:
   "http://xml.ns.org"
```

Exemple 5

```
<?xml version="1.0"
      encoding="ISO-8859-1"?>
<?xml-stylesheet type='text/css'
      href='person.css'?>
<!DOCTYPE person SYSTEM "person.dtd">

<person xmlns="http://xml.ns.org/">
<n:name xmlns:n="http://xml.ns.org/n">
   <n:first>Sydney</n:first>
   <n:last>Lee</n:last>
</n:name>
<assignment projid="p2"/>
</person>
```

```
startElement(
   String namespaceURI,
   String localName,
   String qualifiedName,
   Attributes atts)

namespaceURI:
   "http://xml.ns.org"

localName: "person"

qualifiedName: "person"

atts: {} (no attributes, an
   empty list)
```

Exemple 6

```
<?xml version="1.0“
      encoding="ISO-8859-1"?>
<?xml-stylesheet type='text/css'
      href='person.css'?>
<!DOCTYPE person SYSTEM "person.dtd">

<person xmlns="http://xml.ns.org/">
<n:name xmlns:n="http://xml.ns.org/n">
   <n:first>Sydney</n:first>
   <n:last>Lee</n:last>
</n:name>
<assignment projid="p2"/>
</person>
```

```
ignorableWhitespace(
  char[] text,
  int start,
  int length)

text: <?xml … </person>

start: 181

length: 3
```

Exemple 7

```
<?xml version="1.0“
      encoding="ISO-8859-1"?>
<?xml-stylesheet type='text/css'
      href='person.css'?>
<!DOCTYPE person SYSTEM "person.dtd">

<person xmlns="http://xml.ns.org/">
<n:name xmlns:n="http://xml.ns.org/n">
   <n:first>Sydney</n:first>
   <n:last>Lee</n:last>
</n:name>
<assignment projid="p2"/>
</person>
```

```
startPrefixMapping(
   String prefix,
   String uri)

prefix: "n"

uri: "http://xml.ns.org/n"
```

Exemple 8

```
<?xml version="1.0"
      encoding="ISO-8859-1"?>
<?xml-stylesheet type='text/css'
      href='person.css'?>
<!DOCTYPE person SYSTEM "person.dtd">

<person xmlns="http://xml.ns.org/">
<n:name xmlns:n="http://xml.ns.org/n">
   <n:first>Sydney</n:first>
   <n:last>Lee</n:last>
</n:name>
<assignment projid="p2"/>
</person>
```

```
startElement(
   String namespaceURI,
   String localName,
   String qualifiedName,
   Attributes atts)

namespaceURI:
   "http://xml.ns.org/n"

localName: "name"

qualifiedName: "n:name"

atts: {} (no attributes, an
   empty list)
```

Exemple 9

```
<?xml version="1.0“
      encoding="ISO-8859-1"?>
<?xml-stylesheet type='text/css'
      href='person.css'?>
<!DOCTYPE person SYSTEM "person.dtd">

<person xmlns="http://xml.ns.org/">
<n:name xmlns:n="http://xml.ns.org/n">
   | <n:first>Sydney</n:first>
   <n:last>Lee</n:last>
</n:name>
<assignment projid="p2"/>
</person>
```

```
ignorableWhitespace(
  char[] text,
  int start,
  int length)

text: <?xml … </person>

start: 236

length: 5
```

Exemple 10

```
<?xml version="1.0"
      encoding="ISO-8859-1"?>
<?xml-stylesheet type='text/css'
      href='person.css'?>
<!DOCTYPE person SYSTEM "person.dtd">

<person xmlns="http://xml.ns.org/">
<n:name xmlns:n="http://xml.ns.org/n">
   <n:first>Sydney</n:first>
   <n:last>Lee</n:last>
</n:name>
<assignment projid="p2"/>
</person>
```

```
startElement(
   String namespaceURI,
   String localName,
   String qualifiedName,
   Attributes atts

namespaceURI:
   http://xml.ns.org/n

localName: "first"

qualifiedName: "n:first"

atts: {} (no attributes, an
   empty list)
```

Exemple 11

```
<?xml version="1.0“
      encoding="ISO-8859-1"?>
<?xml-stylesheet type='text/css'
      href='person.css'?>
<!DOCTYPE person SYSTEM "person.dtd">

<person xmlns="http://xml.ns.org/">
<n:name xmlns:n="http://xml.ns.org/n">
   <n:first>Sydney</n:first>
   <n:last>Lee</n:last>
</n:name>
<assignment projid="p2"/>
</person>
```

```
characters(
   char[] text,
   int start,
   int length)

text: <?xml … </person>

start: 253

length: 6
```

Exemple 12

```
<?xml version="1.0"
      encoding="ISO-8859-1"?>
<?xml-stylesheet type='text/css'
      href='person.css'?>
<!DOCTYPE person SYSTEM "person.dtd">

<person xmlns="http://xml.ns.org/">
<n:name xmlns:n="http://xml.ns.org/n">
   <n:first>Sydney</n:first>
   <n:last>Lee</n:last>
</n:name>
<assignment projid="p2"/>
</person>
```

```
endElement(
   String namespaceURI,
   String localName,
   String qualifiedName)

namespaceURI:
   "http://xml.ns.org/"

localName: "first"

qualifiedName: "n:first"
```

Exemple 13 … Exemple 18

```
<?xml version="1.0"
      encoding="ISO-8859-1"?>
<?xml-stylesheet type='text/css'
      href='person.css'?>
<!DOCTYPE person SYSTEM "person.dtd">

<person xmlns="http://xml.ns.org/">
<n:name xmlns:n="http://xml.ns.org/n">
   <n:first>Sydney</n:first>
   <n:last>Lee</n:last>
</n:name>
<assignment projid="p2"/>
</person>
```

```
endElement(
   String namespaceURI,
   String localName,
   String qualifiedName)

namespaceURI:
   "http://xml.ns.org/"

localName: "name"

qualifiedName: "n:name"
```

Exemple 19

```
<?xml version="1.0“
      encoding="ISO-8859-1"?>
<?xml-stylesheet type='text/css'
      href='person.css'?>
<!DOCTYPE person SYSTEM "person.dtd">

<person xmlns="http://xml.ns.org/">
<n:name xmlns:n="http://xml.ns.org/n">
   <n:first>Sydney</n:first>
   <n:last>Lee</n:last>
</n:name>
<assignment projid="p2"/>
</person>
```

```
endPrefixMapping(
   String prefix)

prefix: "n"
```

Exemple 19 … Exemple 26

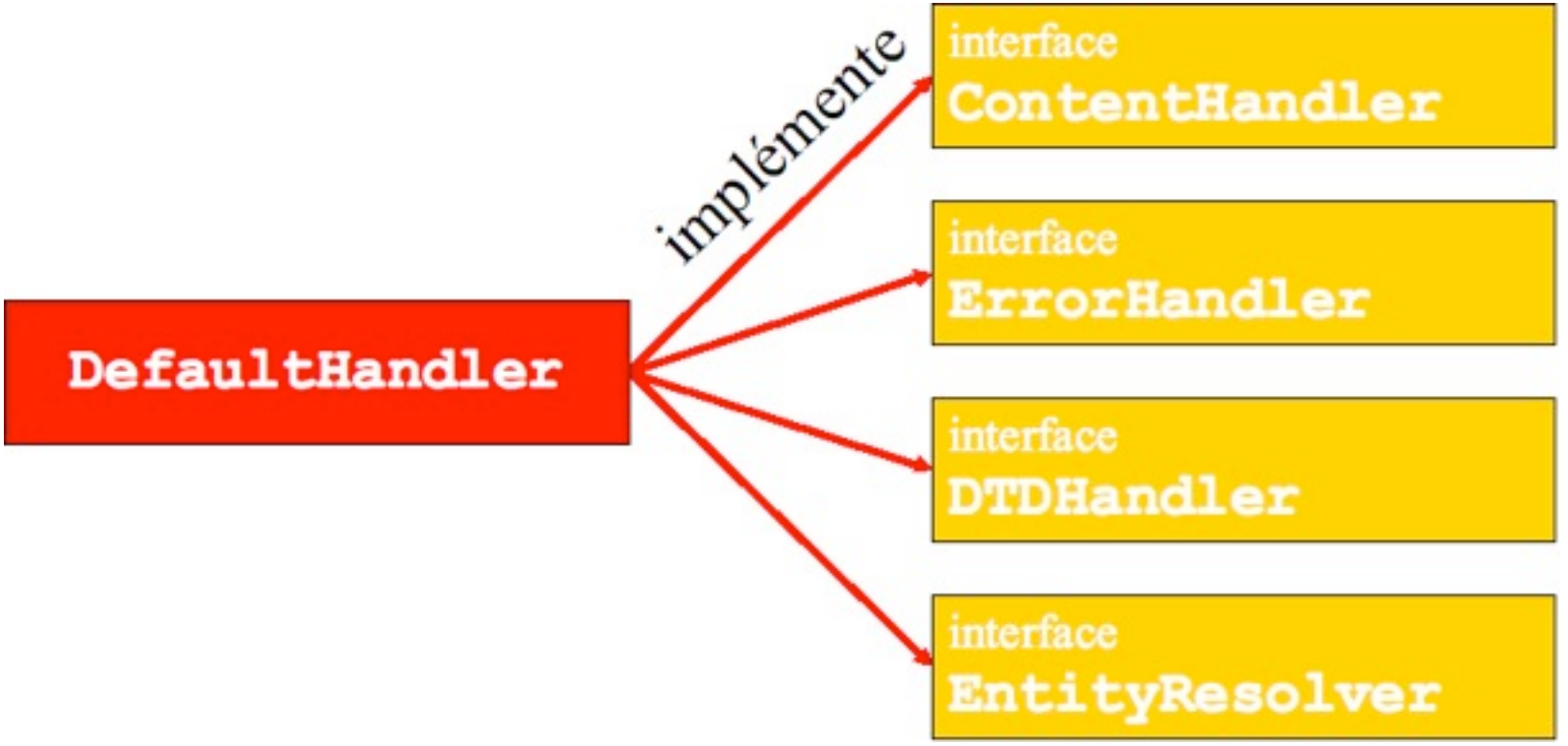
```
<?xml version="1.0“
      encoding="ISO-8859-1"?>
<?xml-stylesheet type='text/css'
      href='person.css'?>
<!DOCTYPE person SYSTEM "person.dtd">

<person xmlns="http://xml.ns.org/">
<n:name xmlns:n="http://xml.ns.org/n">
   <n:first>Sydney</n:first>
   <n:last>Lee</n:last>
</n:name>
<assignment projid="p2"/>
</person>
```

```
endDocument( )
```

Le package org.xml.sax.helpers

- Package qui regroupe les implémentations par défauts des classes se trouvant dans les packages `org.xml.sax.*`, comme par exemple `DefaultHandler`



Le package org.xml.sax.helpers (2)

- On y trouve aussi la classe XMLReaderFactory :
- XMLReader xmlReader = XMLReaderFactory.createXMLReader("org.apache.xerces.parsers.SAXParser ");
- Permet d'instancier un XMLReader à partir d'une implémentation donnée de SAXParser.

Quelques classes et attributs à connaître

- javax.xml.parsers : Classes SAXParser, SAXParserFactory

- org.xml.sax : Interfaces XMLReader, DTDHandler, ErrorHandler, ContentHandler
- org.xml.sax : Interface Attributes
- org.xml.sax.helpers : Classe DefaultHandler

Autres méthodes pour parser

- Directement à partir de la classe `XMLReaderFactory` dans le package `org.xml.sax.helpers`:

```
try {
  XMLReader parser =
    XMLReaderFactory.createXMLReader( );
  // parse the document
... } catch (SAXException e) { ... }
```

- Directement à partir de la classe `SAXParser`:

```
try {
 SAXParser saxParser = factory.newSAXParser();
 saxParser.parse(inputFile, handler);
... } catch (SAXException e) { ... }
```

Bénéfice de SAX

- C’est ce qu’on fait de plus simple!
- C’est un outil rapide qui nécessite une faible empreinte mémoire (indépendante de la taille du document initial)
- Il est utile lorsqu’on doit utiliser des types de données spécialisés pour modéliser les documents XML
- Pratique lorsqu’on a besoin de travailler uniquement sur un petit sous-ensemble d’un document

Inconvénients de SAX

- On ne peut pas modifier directement un document (accès read-only)
- On ne peut pas accéder directement à un endroit précis d'un document (no random-access)
- La recherche de documents n'est pas facile

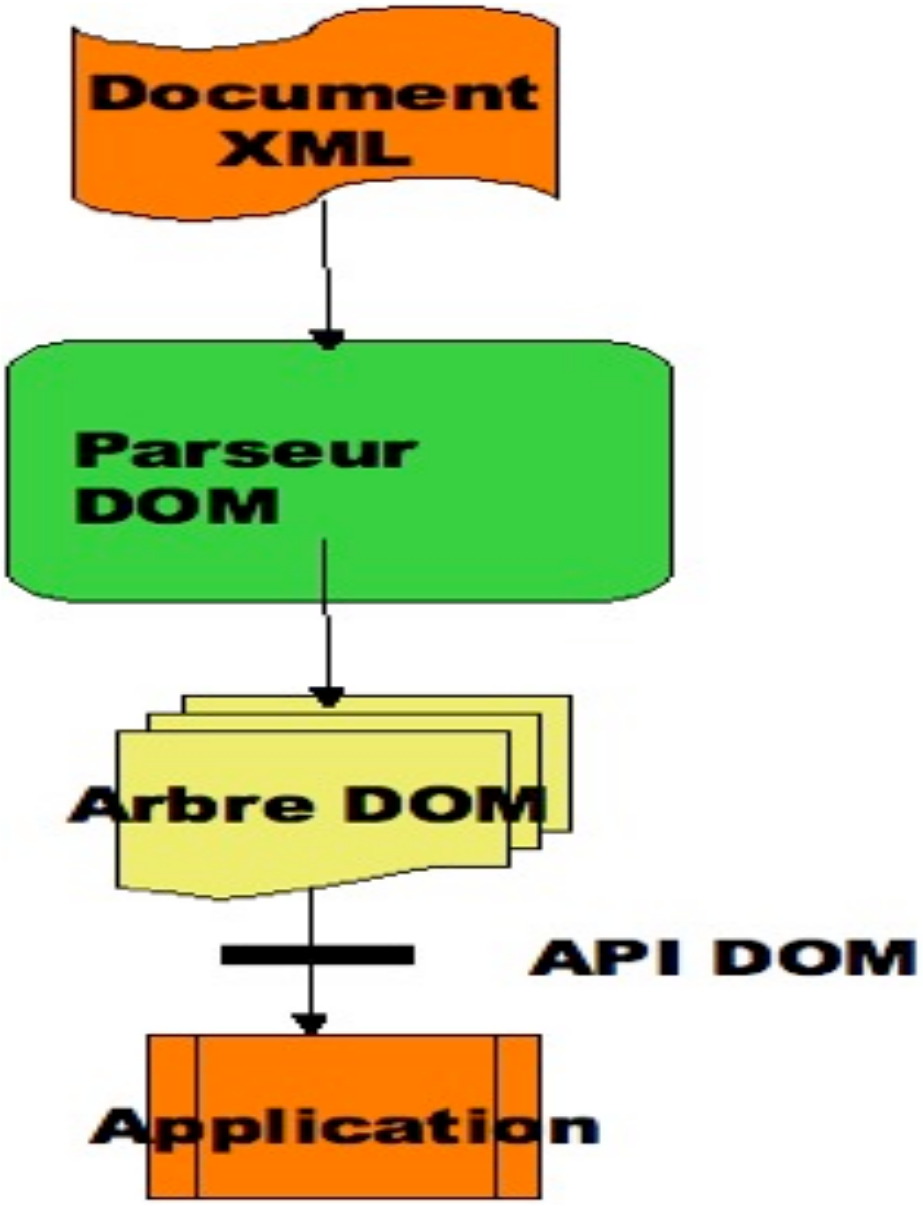
DOM =
Document Object Model

Présentation de DOM

- DOM est une API pour la manipulation de documents XML bien-formés et valides
- DOM Level 2 Core Specification, recommandation W3C depuis 2000

  http://www.w3.org/TR/DOM-Level-2-Core/
- Permet de définir la structure logique des documents mais également de quelle manière les documents peuvent être parcourus et modifiés
  - il est possible de construire de nouveaux documents, de naviguer dans un document, de créer modifier et détruire des éléments
  - comme avec SAX, les documents sont manipulés sous leur forme arborescente, mais il est possible de parcourir l'arbre dans toutes les directions (même en arrière)

Aperçu de DOM



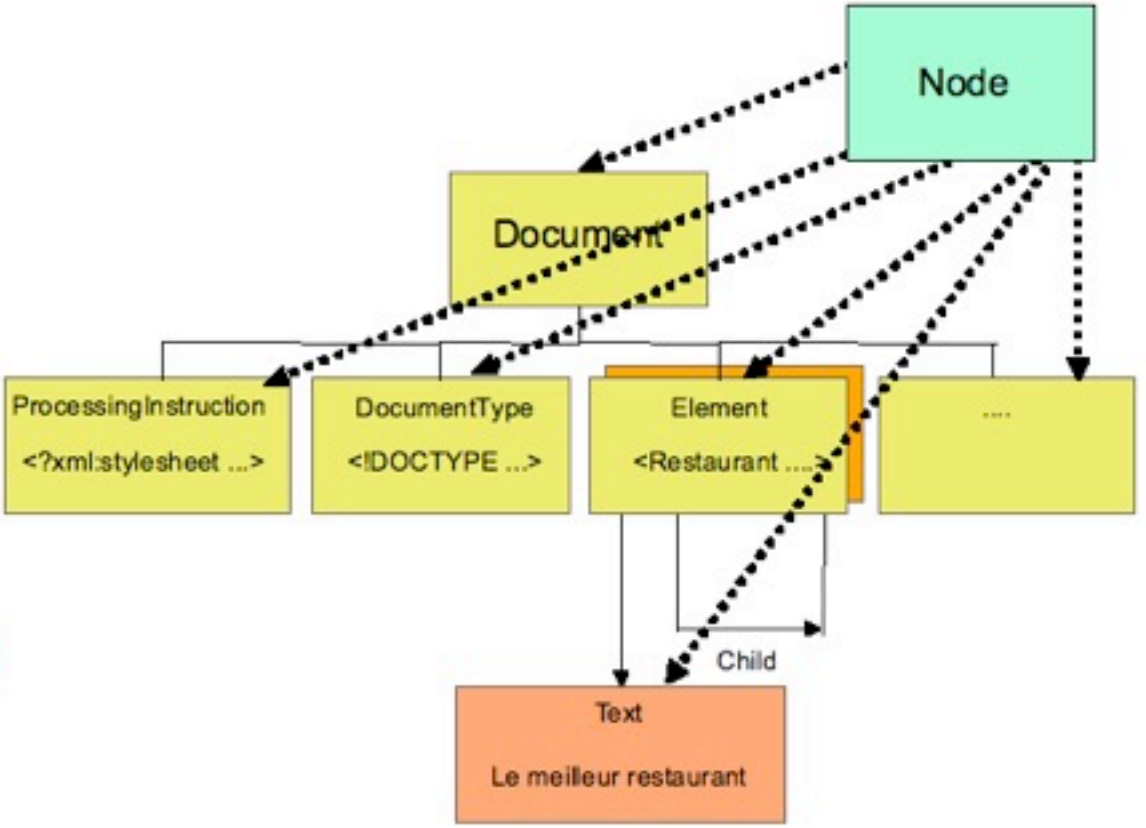
org.w3c.dom Interface Node

- L'interface Node est le type de base qui correspond à un des noeuds de l'arbre DOM
- Tous les objets implémentant l'interface Node disposent de méthodes pour obtenir le père, les frères, les fils, … du noeud
- **Sous-Interfaces:**

```
Attr, CDATASection, Comment, Document,
DocumentFragment, DocumentType, Element,
Entity, EntityReference, Notation,
ProcessingInstruction, Text
```

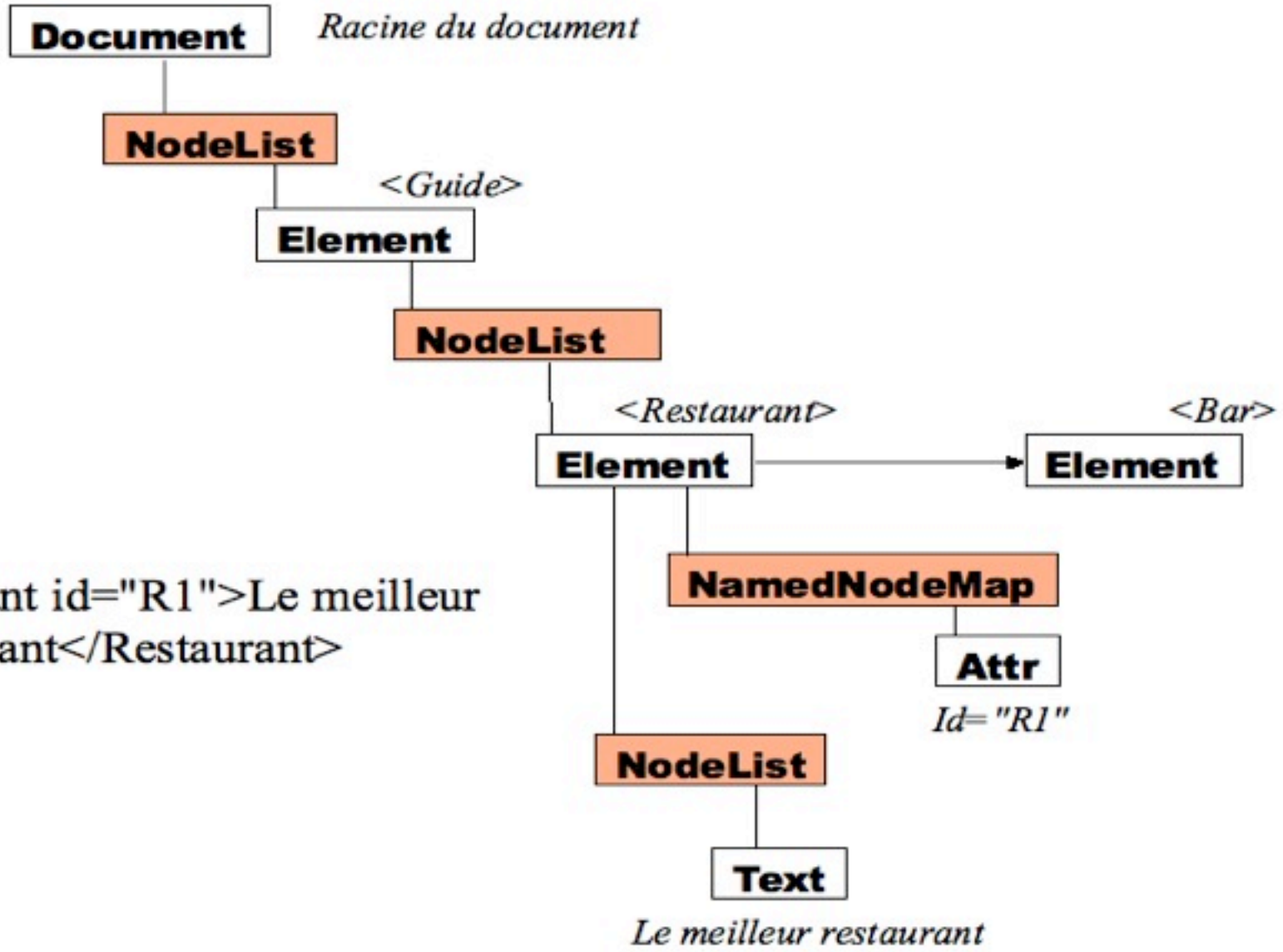
Les Arbres DOM

- Navigation via un arbre générique de nœuds
  - Node
  - NodeList (Parent/Child)
  - NamedNodeMap



- Tout nœud hérite de Node

Exemple d'arbre DOM




```
<Guide>
    <Restaurant id="R1">Le meilleur
      restaurant</Restaurant>
    <Bar/>
</Guide>
```

Node	node Name	node Value	node Type
Attr	name	value	2
CDATAsection	#cdata-section	content	4
Comment	#comment	content	8
Document	#document	null	9
DocumentFragment	#document-fragment	null	11
DocumentType	doc type name	null	10
Element	tag name	null	1
Entity	entity name	null	6
EntitReference	name of entity referenced	null	5
Notation	notation name	null	12
ProcessingInstruction	target	content	7
Text	#text	content	3

Un exemple

```
//  JAXP APIs qu'on utilise
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.FactoryConfigurationError;
import javax.xml.parsers.ParserConfigurationException;
// les exceptions pouvant etre levees
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;
// inputs/outputs
import java.io.File;
import java.io.IOException;
// les definitions W3C pour DOM et les exceptions DOM
import org.w3c.dom.*;

public class SimpleDomParser {
    static final String[] typeName = {
        "none",
        "Element",
        "Attr",
        "Text",
        "CDATA",
        "EntityRef",
        "Entity",
        "ProcInstr",
        "Comment",
        "Document",
        "DocType",
        "DocFragment",
        "Notation",
    };

    public static void main(String argv[])
    {
        DocumentBuilderFactory factory =
            DocumentBuilderFactory.newInstance();
    try {
        DocumentBuilder builder = factory.newDocumentBuilder();
        Document document = builder.parse(new File(argv[0]));
        walk_the_tree(document, 0);
     } catch (SAXException sxe) {
        Exception  x = sxe;
        if (sxe.getException() != null)
            x = sxe.getException();
        x.printStackTrace();
       } catch (ParserConfigurationException pce) {
            pce.printStackTrace();
       } catch (IOException ioe) {
        ioe.printStackTrace();
       }
     }

    ////////////////////////////////////////////////////

public static void walk_the_tree(Node domNode, int depth) {
     String s = typeName[domNode.getNodeType()];
     String nodeName = domNode.getNodeName();
     if (! nodeName.startsWith("#")) {
        s += "  " + nodeName;
     }
      [....]
     System.out.println(s);
      [....]
     NodeList child = domNode.getChildNodes();
     for (int i = 0; i < child.getLength(); i++)
        walk_the_tree(child.item(i), depth+1);
     }
}
```

Un exemple (2)

```
$> java -cp . SimpleDomParser wg.xml
Document
-DocType  gdt
-Element  gdt
--Attr  month = February
---Text = February
--Attr  year = 2004
---Text = 2004
--Element  expose
---Attr  day = 5
----Text = 5
---Element  speaker
----Text = Nicolas Baudru
...
```

Quelques interfaces de base

- Document
  - createElement (Nom_Element):
    - créer un élément avec le nom spécifié en paramètre.
  - createComment (commentaire):
    - créer une ligne de commentaires dans le document.
  - createAttribute (Nom_Attribut):
    - créer un attribut avec le nom pris en paramètre.
  - getElementsByTagName (Nom_Tag):
    - retourne tous les descendants des éléments correspondants au Nom_Tag.
- Nœud
  - insertBefore (Nouveau_Noeud, Noeud_Reference):
    - insère un nouveau nœud fils avant le " nœud référence" déjà existant.
  - replaceChild (Nouveau_Noeud, Ancien_Noeud):
    - remplace le nœud "Ancien_Noeud" par le nœud "Nouveau_Noeud".
  - removeChild (Nœud):
    - supprime le nœud entré en paramètre de la liste des nœuds.
  - appendChild (Nouveau_Noeud):
    - Ajoute un nouveau nœud a la fin de la liste des nœuds.
  - hasChildNodes():
    - Retourne vrai si le nœud possède un enfant

Normalisation des nœuds textes

- Les noeuds textes peuvent contenir des caractères “retour chariot” ou être découpés en sous-textes adjacents
- La normalisation réduit les fins de lignes en un seul CR / réduit plusieurs espaces en un seul / s’assure qu’il n’y a pas de noeuds textes adjacents
- Cette opération est utile pour comparer des documents ou bien pour se débarrasser des espaces inutiles

```
Node node = jtree.getNode(treeNode);
node.normalize();
```

Bénéfices de DOM

- Permet un accès direct (random-access) à un document XML.

- On peut créer un DOM à partir de rien ou encore modifier un document contenu dans un fichier déjà existant.

Inconvénients de DOM

- Cette approche n'est pas adaptée à la manipulation de grands documents car le document entier doit être chargé en mémoire

DOM Level 3 (07 April 2004)
http://www.w3.org/TR/DOM-Level-3-Core/

- XPath
  - Support direct de XPath
  - Définition d'un XPath Evaluator
- Style
  - Accès aux styles
  - Mapping complet
- Load and Save
  - Stockage de l'arbre
  - Lecture partielle d'un document XML
- Utilisation d'annotations "non-XML"

DOM et SAX

- DOM utilise SAX pour la construction de l'arbre d'un document XML

Aller plus loin...

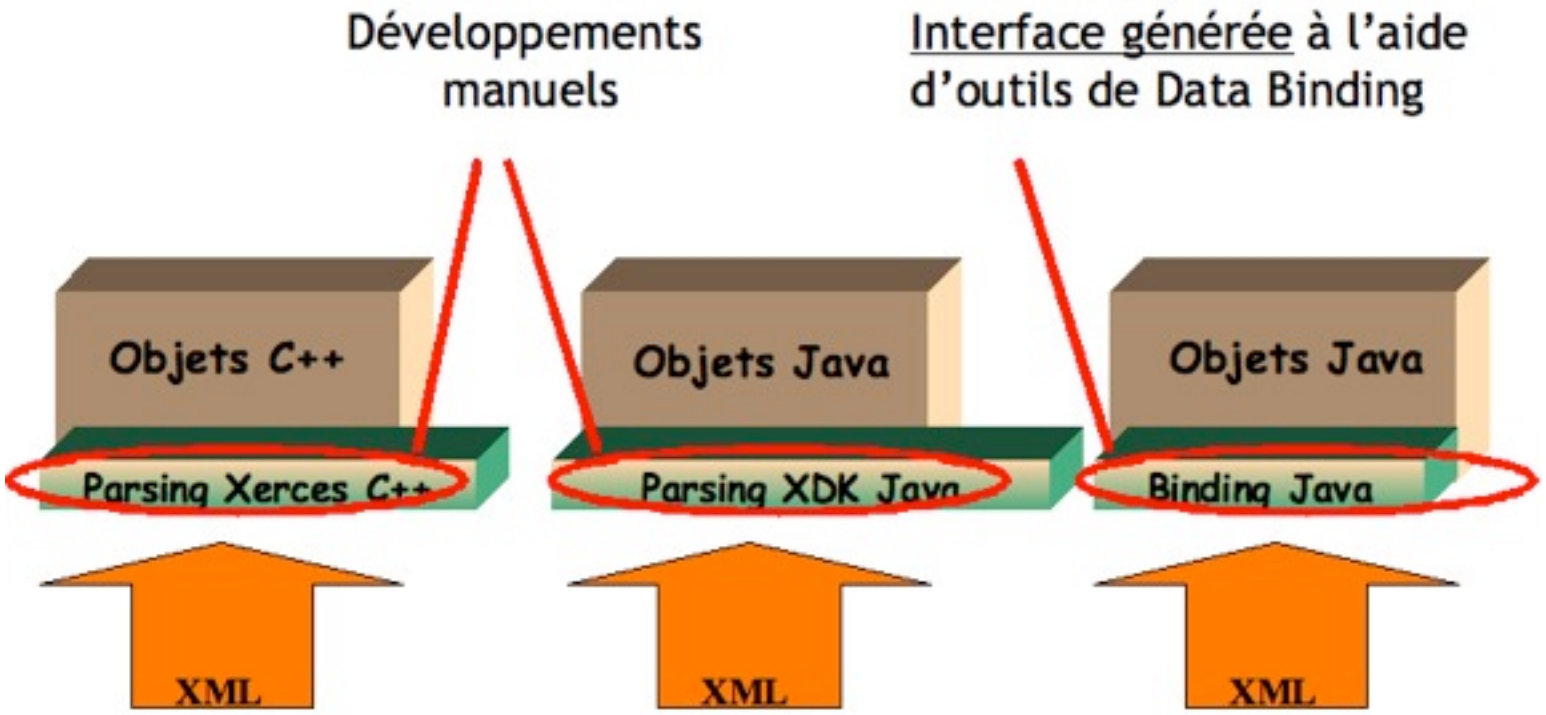
Les librairies implémentant DOM proposent de nombreuses autres fonctionnalités :

- Sélection à l'aide de formules XPath
- Transformations XSLT
- Production du document XML dans un flux de sortie
- Requêtes XQuery
- ...

JAXB =
Java Architecture for XML Binding

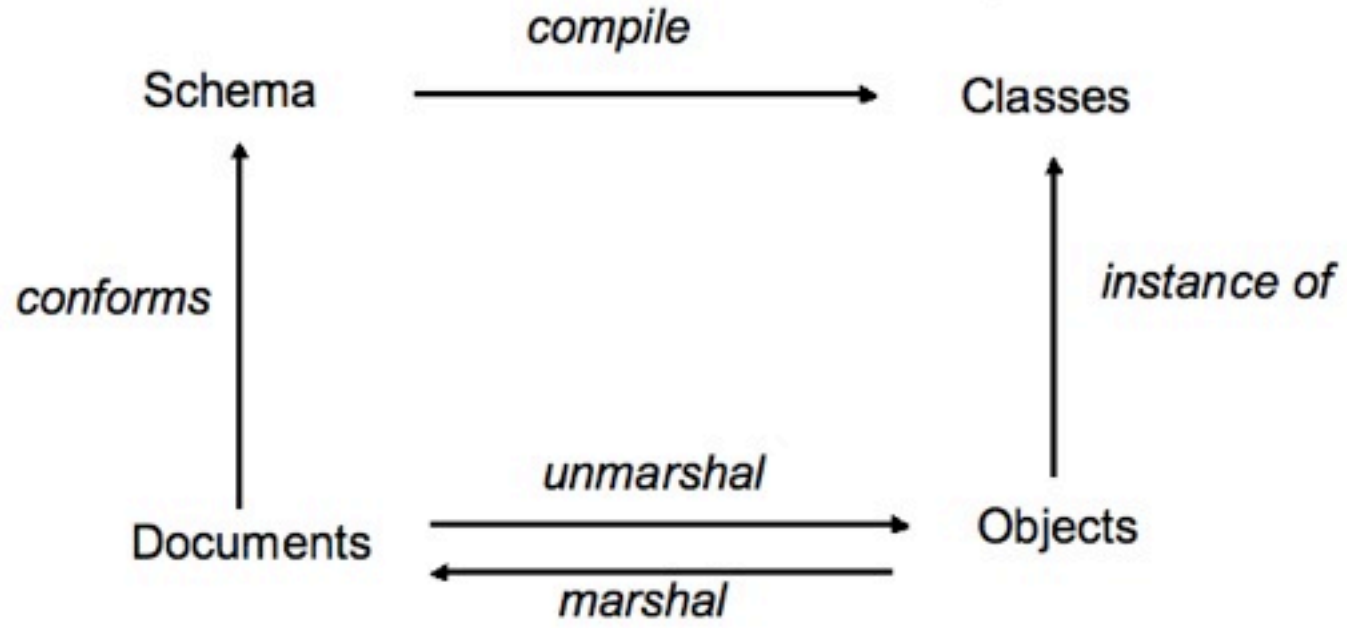
Le mapping objet (Data Binding)

- Définition
  - Mapping entre un document XML et un graphe d'objets métier C++ ou Java



Principes

- Compilation du schéma en classes Java
- Traduction des documents en objets



- Possibilité de publier les objets en XML

Avantages

- L’activité de développement "parsing" est réduite
  - Gain en coût de développement des interfaces
  - Les applications peuvent se concentrer sur le fonctionnel
  - Les développeurs n'ont pas à se soucier du mapping XML
- Les Systèmes Techniques sont simplifiés
  - Capitalisation sur l’interface (utilisation, évolution, …)
  - Gain en termes de fiabilité (fin de l’utilisation de plusieurs parseurs, parfois écrits à la main)
  - Les performances de l’interface sont optimisées
- Les outils de mapping sont intégrés aux serveurs
  - J2EE = JAXB
  - .NET = mapper

Conclusion

- SAX : parsing “à la volée” d’un document XML. Une seule lecture (économe en mémoire), mais pas de modifications possibles.

- DOM : construction d’un arbre représentant le document. Toutes les transformations sont ensuite envisageables...

- JAXB : traduction automatique d’une structure de Schéma XML en graphe de classes Java.

En cas de doute :

Allez voir la Java Doc !

Google : javadoc +

xmlreader / contenthandler / Documentbuilder...