SAX & DOM

Manipulation de données XML

Frédéric Deshaies
frederic.deshaies@c-s.fr

Mai 2004

Manipulation de données XML

Il existe différents **parseurs XML** avec différents niveaux de fonctionnalités

SAX (Simple API for XML)

La lecture du document est linéaire

DOM (Document Object Model)

Le document est chargé en intégralité en mémoire

Objectifs du cours

Vous allez apprendre :

- Les normes SAX
- Les normes DOM
- A utiliser SAX et DOM pour manipuler des messages XML
- A justifier l'emploi de SAX ou de DOM dans une application

Introduction

XML apporte une standardisation de formalisme pour représenter les informations.

XML n'est pas une solution miracle mais facilite la réalisation de systèmes communicants:

- Facilite l'analyse des messages
- Facilite la circulation de l'information
- Facilite la compréhension des messages échangés

Les traitement liés à XML sont standardisés et se basent sur deux normes : **SAX et DOM**.

SAX–Présentation

SAX est née de la collaboration entre développeurs de parseurs XML sur internet

SAX se base sur un modèle de parsing au fil de l'eau

- Pratique si l'on veut rechercher des informations dans un document
- Peut servir de base aux autres modèles de parsing (par exemple DOM)

SAX définit une interface standard d'accès de bas niveau aux parseurs XML

- Modèle événementiel
- Permet de récupérer les informations lues par le parseur
- Le stockage de ces informations en mémoire, si nécessaire, est laissé à la charge de l'application qui utilise le parseur

SAX-Normes

Il existe deux normes SAX, SAX1 à l'heure actuelle répandue et SAX 2, une extension de SAX.

SAX 1 ne propose pas le support des espaces de noms.

Tous les parseurs XML écrits en Java supportent SAX 1

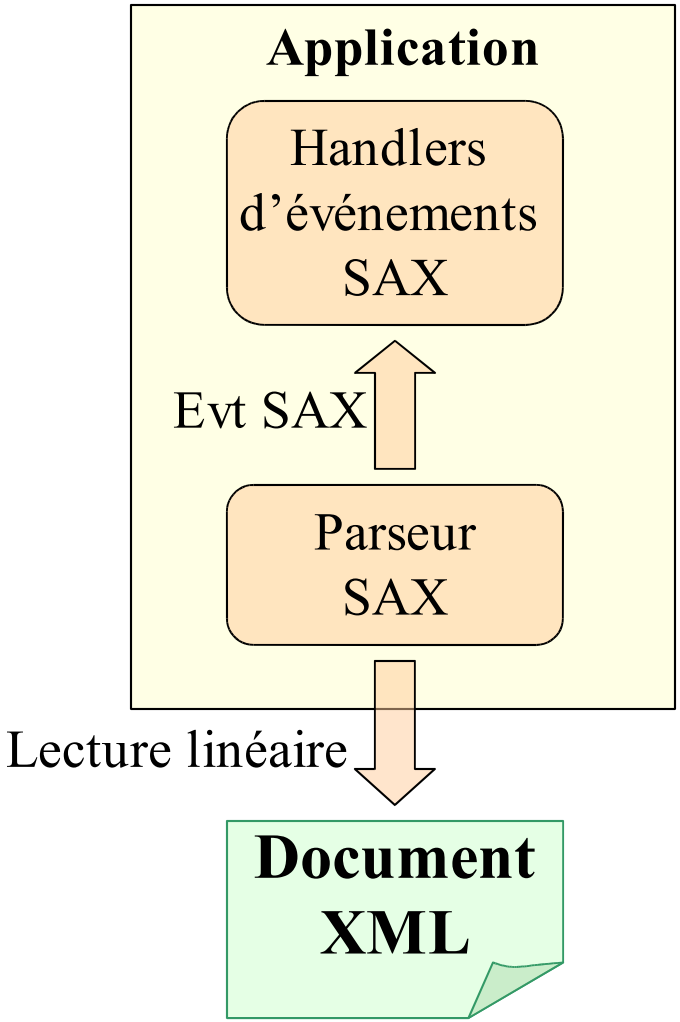
SAX 2 apporte :

- Le support des espaces de noms
- La standardisation des mécanismes de configuration du parseur SAX

SAX 2 n'est pour l'instant supporté que par certains parseurs (Xerces entre autres).

XML

SAX-fonctionnement



L'application utilise des données XML :

- Le document XML est lu séquentiellement
- Les handlers d'événements SAX reçoivent les événements SAX et éventuellement mémorisent les données reçues et/ou effectuent les traitements spécifiques de l'application

SAX-Le parseur SAX

Interfaces

- SAX 1 : org.xml.sax.Parser
- SAX 2 : org.xml.sax.XMLReader

Une application analyse un document en créant une instance de parseur et en utilisant les méthodes parse ().

C'est par le parseur que sont positionnés les handlers d'événements.

SAX-Les handlers d'événements

Le parseur SAX référence un ensemble d'instances pour capturer les événements.

Les événements SAX sont séparés en plusieurs catégories :

- Les événements liés à lecture du document
- Les événements liés à la signalisation des erreurs
- Les événements liés à la DTD
- Les événements liés aux entités externes

Tout comme pour le parseur, les implémentations liées à ces handlers sont différentes selon la version SAX utilisée.

XML

SAX-Handler d'événements de lecture

Interfaces :

- SAX 1 : org.xml.sax.DocumentHandler
- SAX 2 : org.xml.sax.ContentHandler

Les classes implémentant cette interface reçoivent des événements levés lors de la lecture du document XML, entre autres :

- startDocument() : début de lecture du document
- endDocument() : fin de lecture du document
- startElement(String name, AttributeList atts) : lecture d'un élément (balise ouvrante)
- endElement(String name): lecture d'un élément (balise fermante)

SAX-Handler d'événements d'erreur

Interfaces :

- SAX 1 & 2 : org.xml.sax.ErrorHandler

Les classes implémentant cette interface reçoivent des événements signalant une détection d'erreur pendant la lecture du document XML, entre autres :

- warning (…) : notification d'avertissement
- error(…) : notification d'erreur récupérable
- fatalError(…) : notification d'erreur non-récupérable

SAX-Les autres Handlers d'événements

Les événements liés à la DTD

- SAX 1 & 2 : org.xml.sax.DTDHandler
- Cette interface permet à SAX d'accéder aux notations et entités non analysables situées dans la DTD et référencées dans le corps du document.

Les événements liés aux entités externes

- SAX 1 & 2 : org.xml.sax.EntityResolver
- Si le document contient des références à des entités externes, l'URL est normalement analysée automatiquement par le parseur : le fichier est localisé et analysé à l'endroit adéquat. Cette interface permet à une application d'annuler ce comportement.

SAX-Exemple représentatif

```
<Debut>
    <ici> Voici du texte </ici>
    <la/>
</Debut>
```

Événements	Renvoie	Valeur
startDocument	-	-
startElement	LocalName	« Debut »
startElement	LocalName	« ici »
characters	Text	« Voici du texte »
endElement	LocalName	« ici »
startElement	LocalName	« la »
endElement	LocalName	« la »
endElement	LocalName	« Debut »
endDocument	-	-

XML

SAX-Quiz

Que produira l'application qui utilisera le handler de document de la page suivante avec le document XML ci-contre ?

NB : seul les paramètres utiles des méthodes du handler sont représentés.

```
<?xml version="1.0"?>
<B>
    <A>
        <C> 4 </C>
        <C> 2 </C>
    </A>
    <C> 3 </C>
</B>
```

SAX-Quiz

Variables globales :

op[…] **tableau de String**

temp[…] **tableau d'entier**

p **entier**

```
endElement (tagName){
    SI tagName != C ET p-1>=0 ALORS
        p = p-1
        SI op[p-1] =A ALORS
            temp[p-1]=temp[p-1]+temp[p]
        SI op[p-1] =B ALORS
            temp[p-1]=temp[p-1]* temp[p]
}

character (text) {
    n = convertir_entier (text);
    SI op[p-1]=A ALORS temp[p-1]=temp[p-1]+n
    SI op[p-1]=B ALORS temp[p-1]=temp[p-1]*n
}
```

```
startDocument () {
    p = 0
}

endDocument (){
    afficher temp[0]
}

startElement (tagName){
    SI tagName != C
        op[p] = tagName
        SI tagName = B ALORS
            temp[p] = 1
        SI tagName = A ALORS
            temp[p] = 0
        p = p+1
}
```

SAX–Utilisation avec Java

Créer le parseur SAX en appelant la factory ParserFactory du package org.xml.sax.helpers

```
Parser p = ParserFactory.makeParser ("com.xxx.ASAXParser")
```

Créer des handlers pour les événements traités par l'application

- Soit en implémentant les différentes interfaces
- Soit en héritant de la classe HandlerBase (SAX 1) qui implémente toutes les interfaces handlers avec des méthodes vides. On redéfinit juste les méthodes correspondant aux événements utilisés par l'application.

Enregistrer les handlers d'événements dans le parseur

Il y a une méthode pour enregistrer un handler pour chaque type d'événements : setDocumentHandler(…), setDTDHandler(…)

```
p.setContentHandler (new MyContentHandler ())
```

Lancer la lecture du document XML avec la méthode parse() :

```
p.parse (« /doc/document.xml »)
```

SAX-Exemple d'utilisation SAX 1 en Java

```
import org.xml.sax.HandlerBase;
import org.xml.sax.AttributeList;
import org.xml.sax.Parser;
import org.xml.sax.helpers.ParserFactory;

Class MonHandler extends HandlerBase {
        public void startElement (String name, AttributeList attl){
                System.out.println ("<" + name +  >");
        }
}
Class MonApplication {
        public static void main (String[] args){
                Parser p = ParserFactory.makeParser ("com.xxx.ASAXParser");
                p.setDocumentHandler (new MonHandler ());
                p.parse ("/doc/document.xml");
        }
}
```

SAX 2-Exemple d'utilisation en Java

```
import org.xml.sax.helpers.DefaultHandler;
import org.xml.sax.Attributes;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.XMLReaderFactory;
Class MonHandler extends DefaultHandler {
        public void startElement(String nameSpaceURI,String localName,
                                 String qName,Attributes atts){
            System.out.println ("<" + localName + ">");
        }
}
Class MonApplication {
        public static void main (String[] args){
            XMLReader p = XMLReaderFactory.createXMLReader("com.xxx.ASAXParser");
            p.setContentHandler (new MonHandler ());
            p.parse ("/doc/document.xml");
        }
}
```

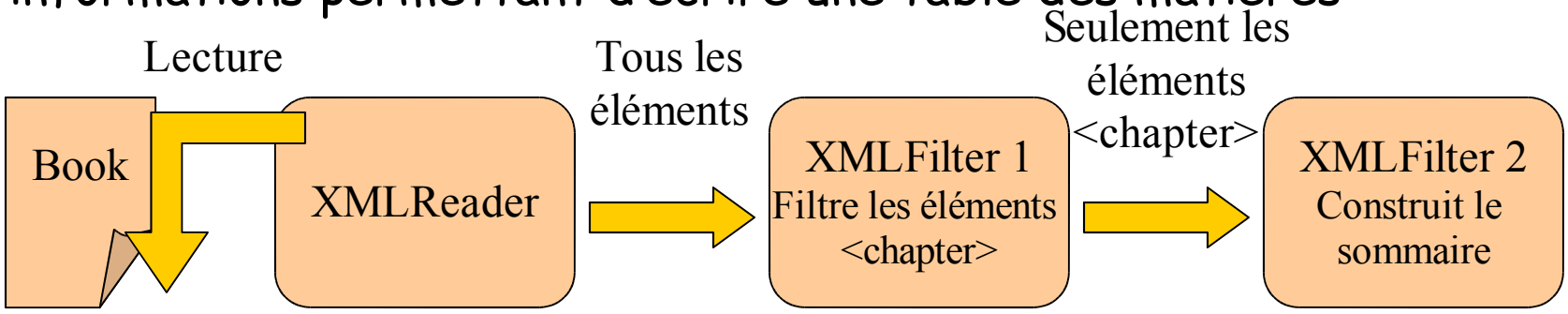
SAX 2-Interface XMLFilter

XMLFilter permet de sélectionner les événements qui sont passés à l'application

- XMLFilter hérite de XMLReader
- Un filtre est construit à partir d'un XMLReader

On peut enchaîner les filtres pour sélectionner de plus en plus finement les événements

Exemple : schéma de fonctionnement pour récupérer les informations permettant d'écrire une table des matières



SAX 2-Configuration du parseur

La classe XMLReader peut être configurée de façon standard

Une feature a une valeur booléenne et permet de vérifier si une fonctionnalité est implémentée ou d'activer une fonctionnalité

- http://xml.org/sax/features-validation permet de spécifier si le parseur doit reporter les erreurs de validation
- ...

Une property peut avoir comme valeur un objet Java quelconque

- http://xml.org/sax/properties/xml-string est accessible en lecture et représente la partie du document qui a donné lieu à l'événement en cours de traitement.
- ...

SAX–Les parseurs du commerce

Nom	URL	Langage
Aelfred	http://home.packbell.net/david-b/xml#utilities	Java
Saxon	http://users.iclway.co.uk/mhkay/saxon,/index.html	Java
MSXML3	http://msdn.microsoft.com/downloadsdefault.asp	C++, VB et COM
Xerces C++	http://xml.apache.org/xerces-c/index.html	C++
Xerces Java	http://xml.apache.org/xerces-j/index.html	Java
JAXP	http://java.sun.com/xml/download.html	Java
XP	http://www.jclark.com/xml/xp/index.html	Java

XML

DOM–Présentation 1/2

DOM permet à une application :

- De modifier un document
- De naviguer dans un document
- D'interagir avec un document (DOM level 2)

Pour les documents HTML :

- Permet de manipuler un document HTML affiché dans un navigateur
- Similaire à Dynamic HTML
- Utilisation avec un langage de script type Javascript ou VBscript

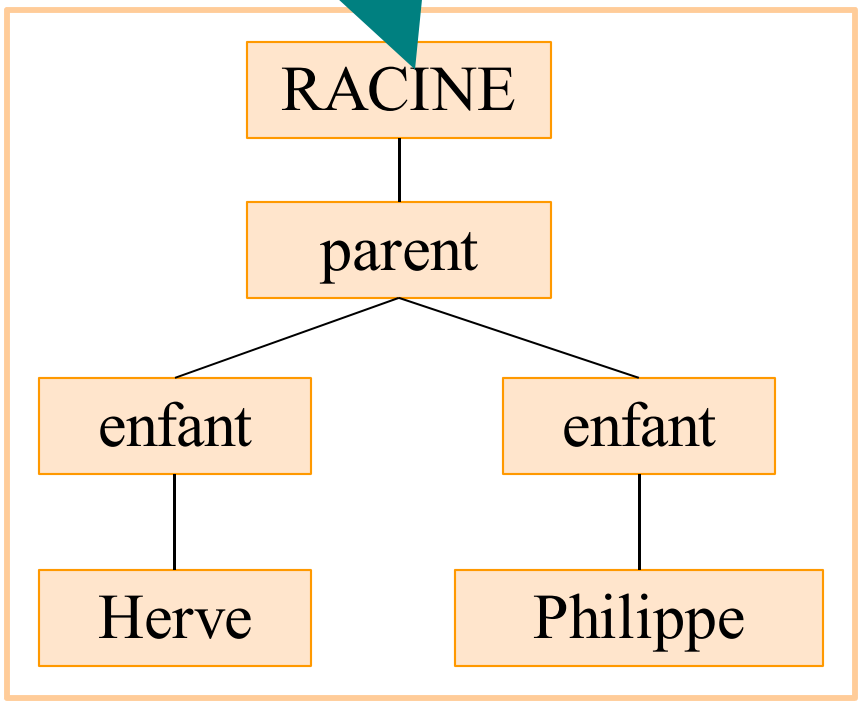
XML

DOM-Présentation 2/2

DOM (Document Object Model) définit la structure des documents XML comme un arbre d'objets.

Construction du DOM

```
<parent>
    <enfant>
        Hervé
    </enfant>
    <enfant>
        Philippe
    </enfant>
</parent>
```



XML

DOM-Normes

DOM est normalisé par le W3C. Cette norme décrit la spécification complète des API DOM.

DOM définit les moyens d'accès et de manipulation des documents

- Définition de l'interface des objets

Les interfaces doivent être respectées pour faciliter l'interopérabilité mais l'implémentation est libre:

- Libre choix de la représentation des objets mémoire
- Libre choix des algorithmes utilisés

Indépendant des langages ou des systèmes

- Décrit en OMG-IDL

XML

DOM–Organisation de la norme

La spécification DOM existe en deux versions et s'articule en plusieurs parties :

DOM level 1

- DOM Core : accès aux éléments d'un document XML ou HTML
- DOM HTML: facilite la manipulation des documents HTML

DOM level 2 : construit sur DOM level 1

- Prise en charge des espaces de noms dans la partie Core
- Style sheets, Cascading style sheets
- Autres fonctionnalités intéressantes
  - ✓ Modèle d'événement
  - ✓ Itérateur, filtre sur ces itérateurs
  - ✓ Notion de vue d'un document

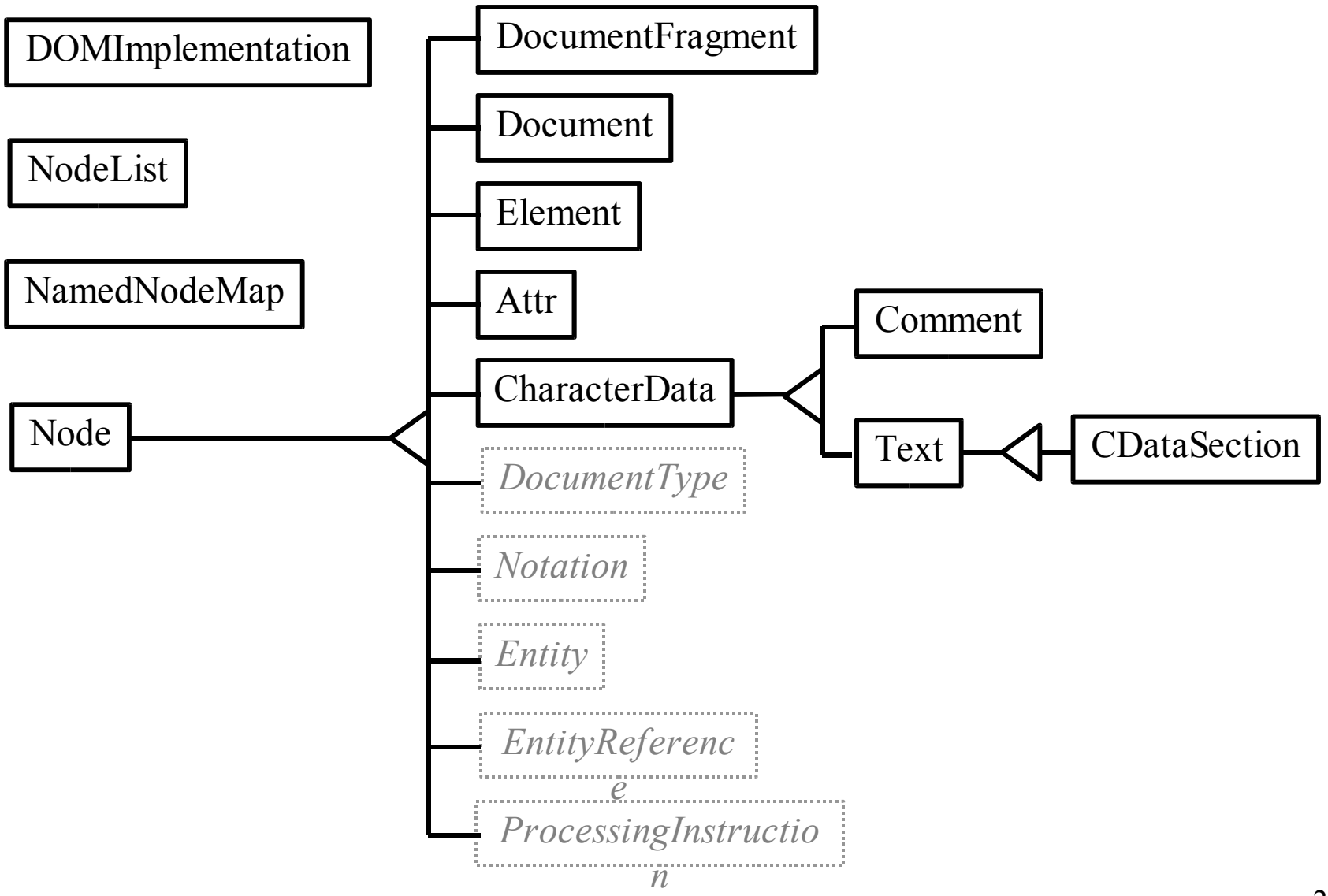
XML

DOM–Interfaces fondamentales

Les API DOM se décomposent en deux types de modules :

- un module commun
  - le **DOM core** pour les documents de base
- des modules optionnels pour d'autres documents tels que :
  - **DOM VIEWS** : gère le contenu de la représentation d'un document
  - **DOM Events** : offre un système générique d'événements
  - **DOM HTML** : gère le contenu d'un document HTML
  - **DOM CSS** : gère le contenu et la structure de feuilles de style

DOM-DOM core


DOMImplementation
NodeList
NamedNodeMap
Node
DocumentFragment
Document
Element
Attr
CharacterData
Comment
Text
CDataSection
DocumentType
Notation
Entity
EntityReference
ProcessingInstruction

DOM–Interface Node 1/2

`Node` représente un nœud du document.

`Node` est l'interface la plus importante du DOM. Presque tous les objets abordés ici étendent les fonctionnalités de cette interface puisque tout élément d'un document XML est un nœud.

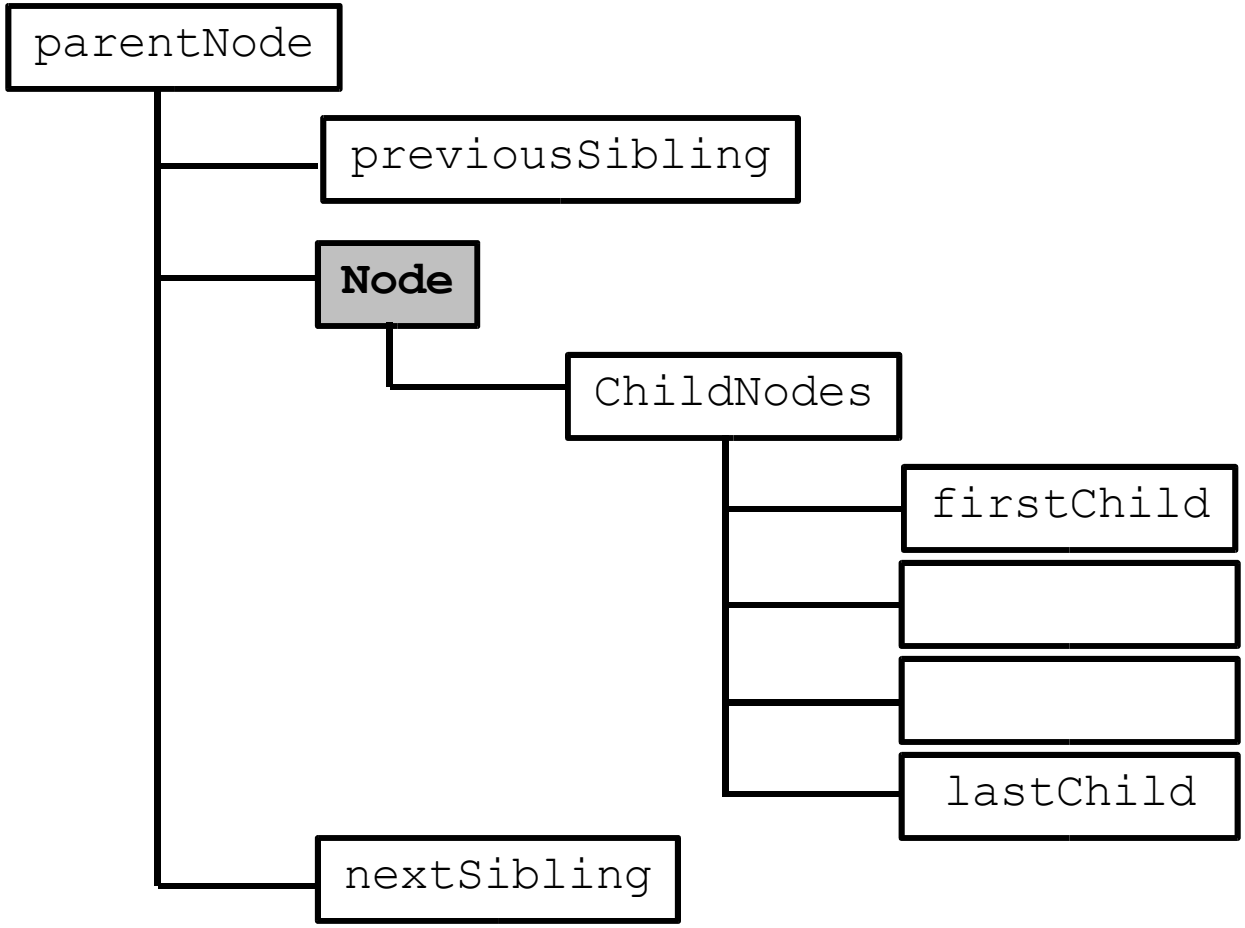
L'objet `Node` permet de réaliser trois actions clés :

- Traverser l'arbre
- Extraire des informations sur le nœud
- Ajouter, supprimer et actualiser des nœuds

XML

DOM-Interface Node 2/2

La navigation dans l'arbre s'effectue principalement avec les propriétés définies dans Node.



DOM–Interface NodeList

L'interface NodeList permet d'atteindre une liste de nœuds. Ce sont principalement les fonctions de navigation dans l'arbre qui renvoient ce type d'objet.

Cette interface comporte une propriété et une méthode :

- La propriété length renvoie le nombre d'éléments dans la liste
- La méthode item renvoie l'élément repéré par son index dans la liste.

Chaque objet de type Node possède une NodeList.

DOM–Interface Document

La racine de l'arbre est de type `Document`

- Accès au type du document (Document Type) et à sa DTD
- Sert de factory pour les autres objets du DOM
- Contient 1 seul objet fils de type `Element` (élément racine du document XML)

L'interface `Document` étend `Node` et lui apporte des fonctionnalités de navigation supplémentaires :

- Trouver des éléments dans le document d'après leur nom
- Identifier des éléments d'après leur attribut ID

XML

DOM–Interface DocumentFragment

Rappel : un document XML ne peut comporter qu'un élément racine.

Contrairement à l'interface Document, l'interface DocumentFragment représente une portion de document XML.

Ceci permet de disposer de fragments de code plus ou moins bien formés dans un emplacement temporaire.

DOM-Interface Element

L'interface `Element` représente un élément du document XML

Elle étend l'interface `Node` et apporte des fonctionnalités supplémentaires :

- Accès au nom de l'élément (méthode `getTagName`)
- Accès aux éléments fils (méthode `getElementsByTagName`)
- Accès aux attributs de l'élément
- Accès à l'élément racine du document (méthode `getDocumentElement()`)

DOM–Interface Attr 1/2

L'interface Attr étend les possibilités de l'interface Node.

Attention à la différence entre les attributs et les autres éléments du document :

> **Les attributs ne font pas directement partie de la structure arborescente d'un document, ils ne sont pas des enfants d'objets de type Element mais simplement des propriétés.**

Ainsi, les propriétés parentNode, previsousSibling et nextSibling d'un attribut renverront toujours **null** !

Pour atteindre l'élément auquel appartient l'attribut, il faut utiliser la propriété ownerElement.

XML

DOM-Interface Attr 2/2

L'accès aux attributs s'effectue depuis l'interface Element :

- getAttribute (…) : lecture de la valeur d'un attribut selon son nom
- setAttribute (…) ajout ou remplacement d'un attribut
- getAttributes (…) retourne une liste de couples associant un nom d'attribut à un nœud Attr
- getAttributeNode (…) lit un attribut
- setAttributeNode (…) modifie un attribut
- removeAttribute (…) supprime un attribut

DOM–Interface NamedNodeMap

L'interface NamedNodeMap permet de représenter une liste non ordonnée de nœuds dont les éléments sont extraits à l'aide de leur nom.

Cette interface définit des méthodes

- D'accès aux nœuds du document
- De gestion de la liste de nœuds (insertion, suppression)

Les attributs d'un élément sont rangés à l'aide d'un objet de type NamedNodeMap.

DOM-Interface Text

Les nœuds de type Text représentent le texte contenu dans un élément

<commande>liste de livres sur XML</commande>

et permettent :

- La récupération du texte sous forme de chaînes de caractères : méthode getData ()
- Des opérations sur les chaînes de caractères
  - Méthode setData (…) : modification de la chaîne
  - Méthode getLength () : retourne la longueur du texte

Attention ! Un parseur DOM peut interpréter les sauts de lignes comme des séparations entre des nœuds Text

DOM–Interface DOMException

Les opérations DOM lancent une exception lorsqu'une opération demandée ne peut pas être effectuée :

- Lorsque les données sont perdues
- Lorsque l'implémentation est devenue instable

Certains langages et systèmes objets exigent que les exceptions soient capturées (Java) tandis que d'autres ne connaissent pas le concept d'exception (VB, C). Dans ce dernier cas, les mécanismes natifs de rapport d'erreur sont utilisés.

DOM-Exercice

Quelle sera l'arborescence des objets DOM après lecture de ce document ?

```
<Debut>
   <ici id="toto" >
      Voici du texte
   </ici>
</Debut>
```

XML

DOM–Utilisation d'un parseur DOM

Création du parser

DOM ne définit pas de factory. Il faut instancier explicitement le parseur.

```
DOMParser p = new DOMParser ();
```

Lecture du document XML avec la méthode `parse ()`

- Le parseur construit alors en mémoire l'arbre DOM

```
p.parse ("doc/document.xml");
```

- On obtient une instance de type Document sur laquelle on peut appliquer les méthodes DOM

```
Document doc = parser.getDocument ();
```

- Cette partie est spécifique à chaque parseur. L'exemple ci-dessus fonctionne avec Xerces.

DOM-Exemple d'utilisation en Java

```
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.apache.xerces.parsers.DOMParser;

Public static void main (String[] args) {
        DOMParser parser = new DOMParser ();
        parser.parse ("book.xml");

        Document d = parser.getDocument ();
        Element root = d.getDocumentElement ();
        String rootTagName= root.getTagName ();
        System.out.println ("Balise racine : " + rootTagName);
        NodeList l = root.getElementByTagName ("chapter");
        System.out.println ("Il y a " + l.getLength () + " chapitres");
}
```

DOM 2-Présentation

DOM Level 2 apporte plusieurs évolutions importantes :

- Support des espaces de noms dans DOM Core
- Support des événements de type « User Interface »
- Support de nouveaux modes de parcours de l'arbre DOM :
  - ✓ NodeIterators
  - ✓ NodeFilters
  - ✓ TreeWalker

On peut vérifier qu'une implémentation de DOM supporte une fonctionnalité avec la méthode hasFeature() de DOMImplementation()

XML

DOM 2-Support des espaces de noms 1/2

L'interface Node est modifiée pour pouvoir gérer les données associées à l'espace de noms :

- namespaceURI est l'URI associée à l'espace de noms du nœud
- prefix est le prefixe associé à l'espace de noms du nœud
- localName est le nom du nœud

L'attribut nodeName (ou qName) contient le nom qualifié prefix+localName

Ces informations ne sont pertinentes que pour les objets de type Element et Attr.

XML

DOM 2-Support des espaces de noms 2/2

Dans l'interface Element

- Pour ajouter un attribut appartenant à un espace de noms particulier :
  Void setAttributeNS (String nsURI,
  String qualifiedName,String value)
- Pour obtenir la valeur d'un attribut, on précise l'URI mais pas le préfixe de l'espace de noms :
  String getAttributeNS (String nsURI, String localName)

Dans l'interface Document

Element createElementNS (String nsURI, String qName)

Attr createAttributeNS (String nsURI, String qName)

NodeList getElementByTagNS (String nsURI, String localName)

Les méthodes ne traitant pas les espaces de noms sont toujours présentes dans DOM 2

DOM 2–Les événements DOM

Objectifs :

- Fournir un modèle d'événements génériques
- Standardiser les événements pour les navigateurs avec DOM HTML

Deux types d'événements

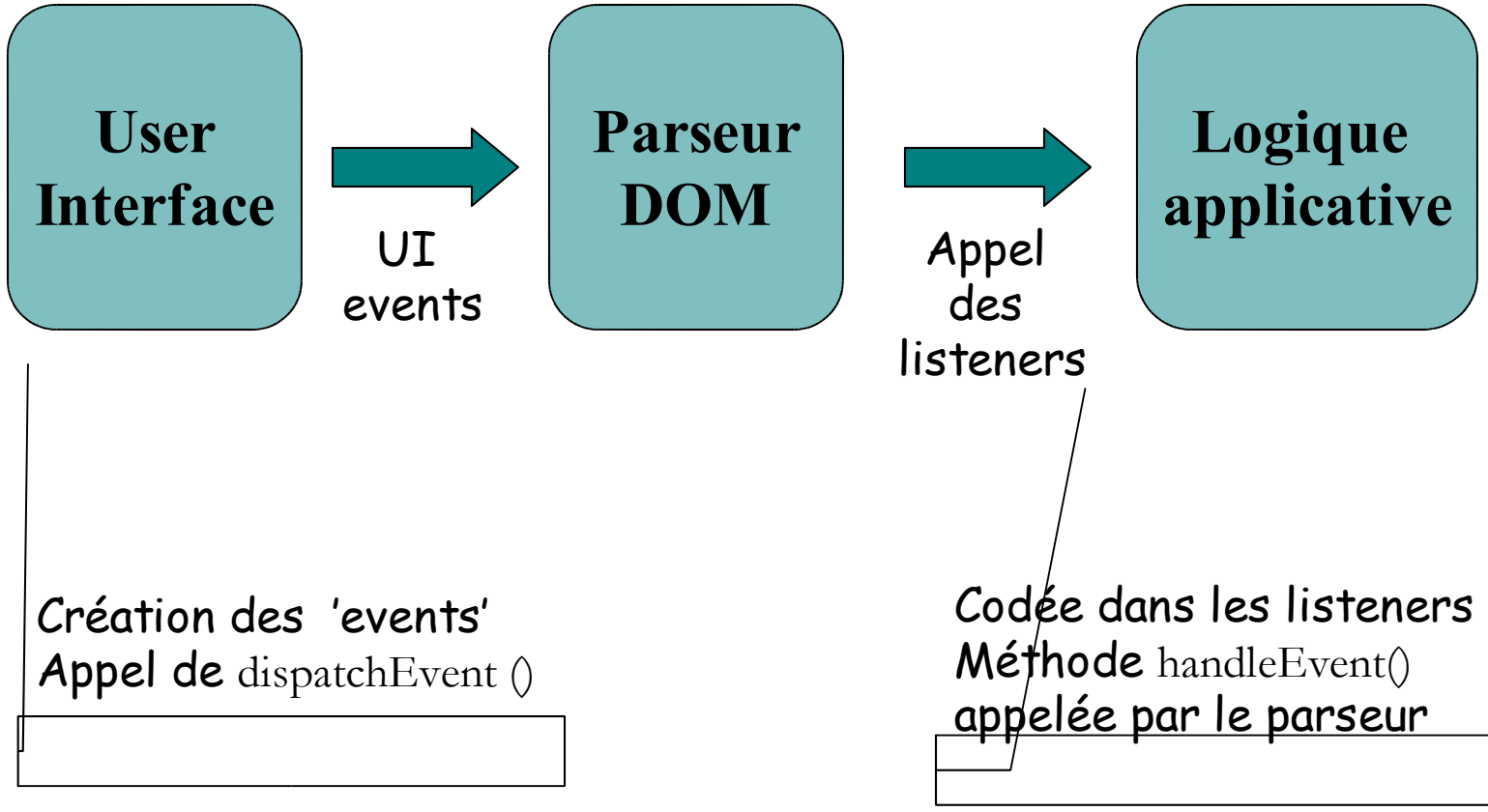
- User Interface events
- User Interface Logical events

Le modèle d'événement est adapté à la structure arborescente du DOM

- Bubling : propagation vers le haut de l'arbre
- Capturing : interception par un ancêtre du destinataire

XML

DOM 2-Principe des événements DOM



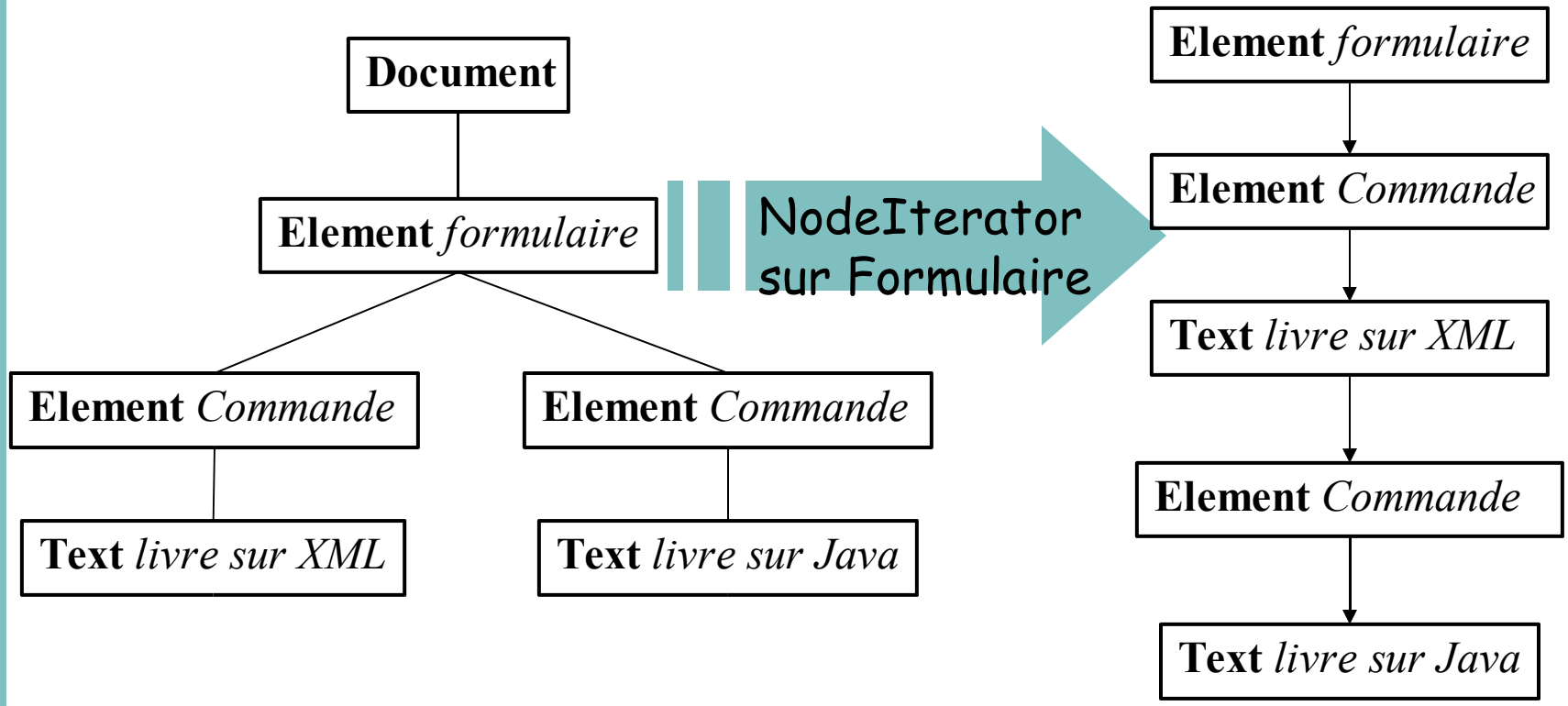
DOM 2-Les Mutations Events

Les mutations events permettent de mettre en place des notifications de modification :

- DOMSubTreeModified
- DOMNodeInserted
- DOMNodeRemoved
- DOMAttrModified
- DOMCharacterDataModified

Une modification du document peut entraîner l'émission de plusieurs événements

DOM 2-NodeIterator



Méthode de l'interface NodeIterator

- Node nextNode() et Node previousNode() pour le parcours
- Node getRoot() permet d'accéder à la racine du sous-arbre

DOM 2-TreeWalker

L'interface TreeWalker reproduit les méthodes de parcours

- De Node
- De NodeIterator

On peut modifier explicitement la position d'un TreeWalker avec void setCurrentNode (Node node) et parcourir l'arborescence à partir de ce nœud.

XML

DOM 2-DOMTraversal et NodeFilter

DOMTraversal permet de créer des instances de type NodeIterator et TreeWalker :

```
NodeIterator createNodeIterator(  Node node, int whatToShow,
                                  NodeFilter filter, boolean entityRefExpansion )
NodeIterator createNodeIterator(  Node node, int whatToShow,
                                  NodeFilter filter,boolean entityRefExpansion )
```

- whatToShow permet de sélectionner des types de nœud
- filter permet de sélectionner plus finement les nœuds

Le NodeFilter teste les caractéristiques du nœud et retourne :

- FILTER_ACCEPT : sélectionne le nœud
- FILTER_REJECT rejette le nœud et tous ses fils (pour les TreeWalker)
- FILTER_SKIP rejette le nœud mais examine ses fils

DOM–Les parseurs du commerce

Nom	Langage
Xerces	Java
4DOM	Python
ActiveDOM	Active X
SDK DOM Docuverse	Java
PullDOM	Python
TclDOM	Script TCL

XML

SAX vs DOM-Avantages de SAX

Gérer de gros documents

Un document analysé par le DOM occupe une place mémoire pouvant atteindre jusqu'à 10 foix la taille du fichier de départ !

Créer des documents comme sous ensemble

SAX peut être utilisée pour créer un document de plus petite taille ne contenant que les éléments nécessaires à l'application

Filtrer les documents dans un flux d'événements

La définition d'une chaîne de filtre permet d'obtenir des transformations de document à différents niveaux.

Interrompre le traitement

Le traitement SAX peut être interrompu depuis n'importe quel gestionnaire d'événement dès que l'application a obtenu l'information depuis la source de donnée

XML

SAX vs DOM–Avantages de DOM

Modifier et enregistrer le document original

SAX ne peut réellement modifier un document XML.

Modifier la structure du document

Le DOM permet de modifier explicitement la structure d'un document.

Procéder à un accès aléatoire : problème de contexte

DOM permet de se déplacer dans le document de n'importe quelle manière alors que SAX ne fait qu'avancer dans le document.

XML

SAX vs DOM–Que choisir ?

Situations mieux adaptées pour DOM

- Pour modifier un document XML fondé sur une saisie utilisateur
- Pour toute structure complexe devant être accédée en mémoire

Situations mieux adaptées pour SAX

- Pour manipuler un gros document ne comportant qu'une faible portion pertinente
- Dans toutes les situations qui ne nécessitent qu'une passe de lecture
- Lorsqu'il faut quitter le document dès qu'une valeur précise a été récupérée
- Lorsque le serveur analysant le document possède des ressources limitées

XML