

TECHNOLOGIES
XML
EXTENSIBLE MARKUP LANGUAGE
PARTIE 4

DOM & SAX

DOCUMENT OBJECT MODEL
&
SIMPLE API FOR XML

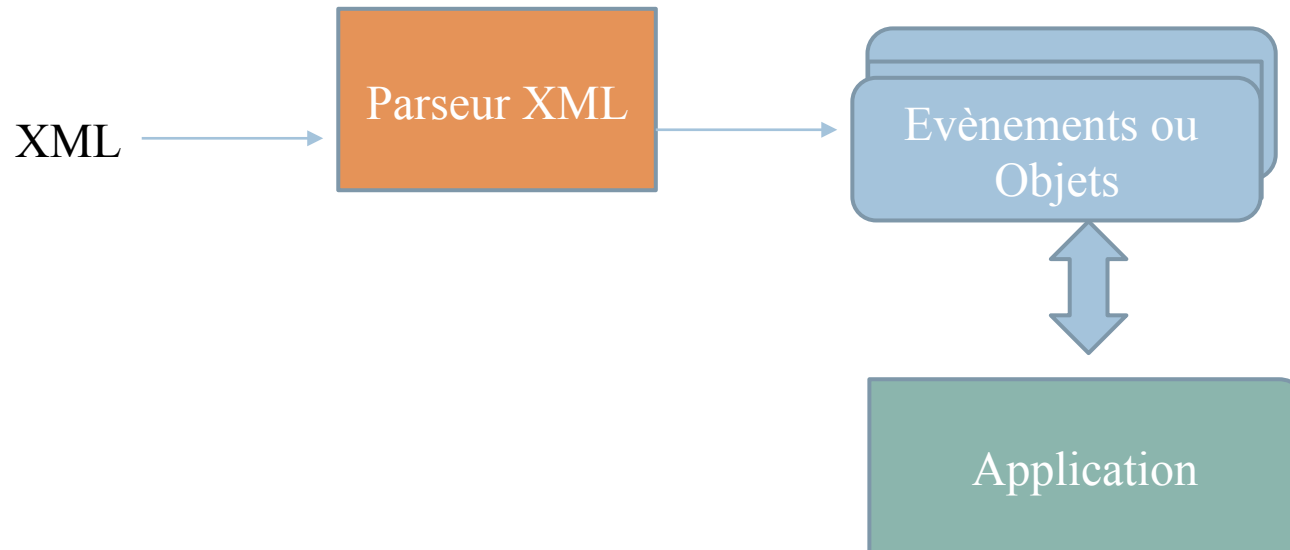
Besoin de parseur

3

- XML est un métalangage qui permet de représenter les données.
- Pour traiter ces données on a besoin du parseur
- Deux types de parseurs existent:
 - ❑ SAX (simple API for XML)
 - Produit un flux d'événement
 - ❑ DOM (Document Object Model)
 - Produit un graphe d'objet en mémoire

Traitement d'un document XML par une application

4



DOM

5

- Standard du W3C (<http://www.w3c.org/DOM/>)
- Parseur qui génère un arbre d'objets reliés entre eux
 - Chaque objet possède des propriétés et des méthodes
- C'est une API (Application Programming Interface) standard disponible dans la plupart des langages d'objet:
 - Java, C++, C#, VB, Python, PHP...
 - Exemple: Javascript utilise DOM pour naviguer dans un document HTML pour récupérer le contenu d'un formulaire, le modifier,...

Versions DOM

6

- DOM Level 1 Recommendation (Oct, 1998)
 - ❓ Représentation d'un document sous la forme d'un arbre. Elle définit deux catégories :
 - Core : spécification générale pour tous les documents XML
 - L'ensemble d'objets et d'interfaces fondamentales pour accéder et manipuler des objets documentaires (document, DocumentFragment, Element, attr, Node, NodeList, etc)
 - L'ensemble d'interfaces étendues spécifiques à la manipulation de documents XML (traitant par exemple les CDATA ou les Processing Instructions, Entity, Notation etc.)
 - HTML: spécification retenant les méthodes applicables à HTML
 - les objets et les méthodes spécifiques aux documents HTML non définis dans le noyau
 - DOM Level 2 Candidate Recommendation (November 2000)
 - ❓ Evolutions du Core
 - ❓ ajoute de nouvelles fonctionnalités tel que la prise en compte des feuilles de style CSS
 - DOM Level 3 Draft (Sept, 2002)
 - ❓ Support de [XPath](#), la gestion d'événements clavier, et une interface de sérialisation de documents XML
 - DOM Level 4 (October 2015 <http://www.w3.org/TR/dom/>)
 - ❓ Amélioration de DOM 3
 - Remplacer les événements de mutation (synchrone et interrompent l'exécution normale du code pour informer votre application des mutations) par les observateurs de mutation (regroupent les enregistrements de mutation pour éviter de submerger l'application par des événements...)
- [https://msdn.microsoft.com/fr-fr/library/dn265032\(v=vs.85\).aspx](https://msdn.microsoft.com/fr-fr/library/dn265032(v=vs.85).aspx)
- <https://www.w3.org/TR/dom/#introduction-to-dom-events>

caractérise la balise. Il contient des primitives pour obtenir les nœuds fils et le parent. il donne également accès aux attributs, soit directement, soit par l'intermédiaire de nœuds de type Attr

représente l'ensemble du document. Il contient une référence (un objet de type Element) vers l'élément racine

Son rôle consiste à stocker un ou plusieurs nœuds

il s'agit d'un attribut d'élément avec un nom et une valeur

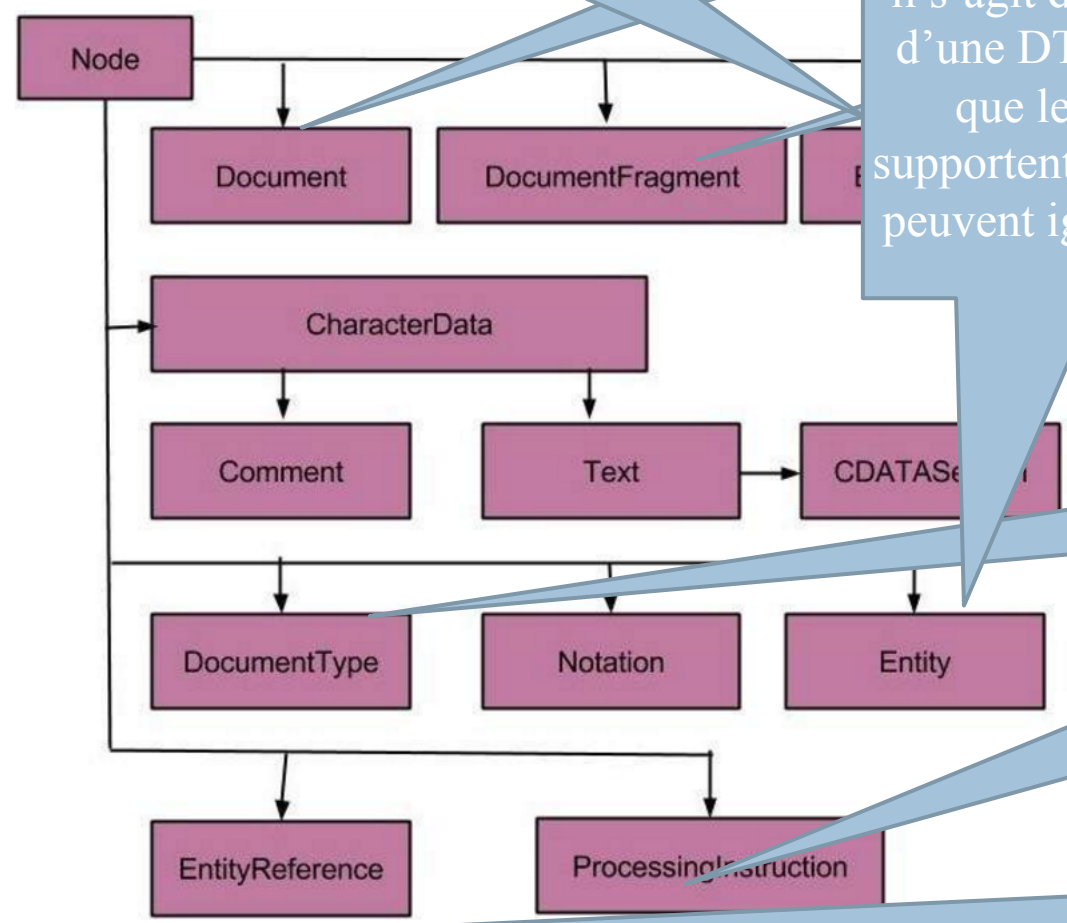
il s'agit d'une entité issue d'une DTD. Il est à noter que les parseurs ne supportent pas la validation peuvent ignorer ce type de nœud

lié à une DTD et caractérise les entités internes ou externes du document

il s'agit d'une instruction de traitement. Il n'y a pas d'analyse ou de manipulation de l'instruction, seule la forme brute est disponible

Elle est positionnée dans un texte et est liée à un objet Entity

DOM défini plusieurs types de nœuds



Exemple

8

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<?xml-stylesheet type="text/xsl" href="test.xsl"?>
```

```
<!-- Ceci est un exemple -->
```

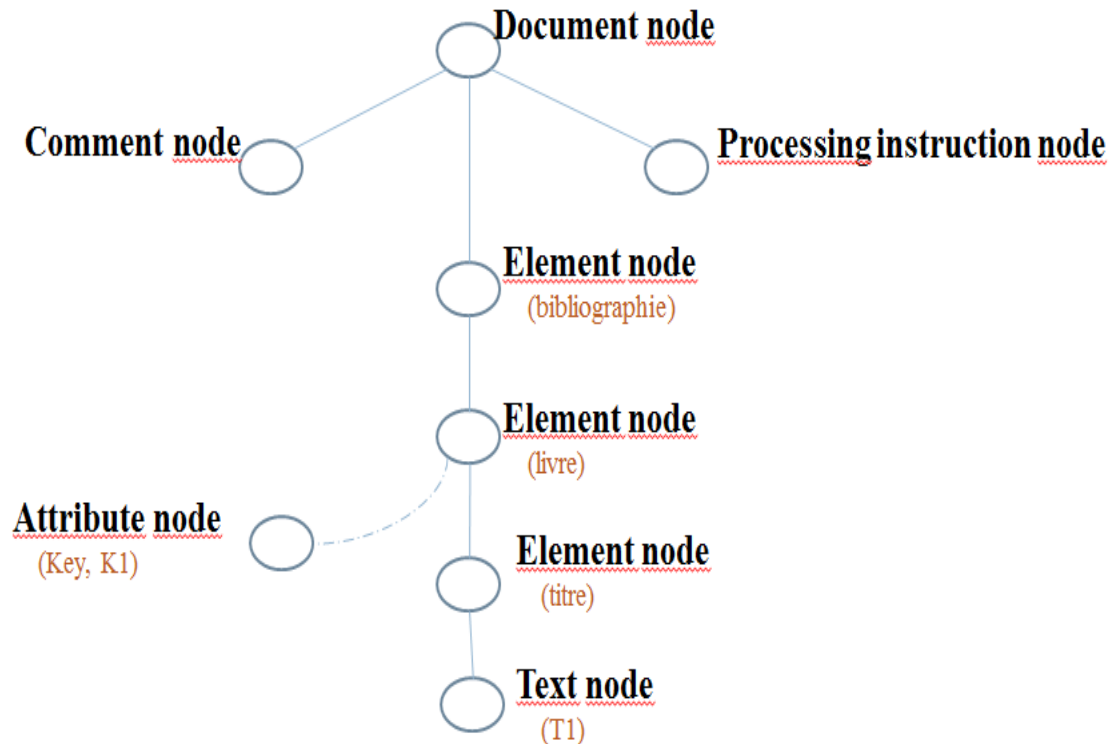
```
<bibliographie >
```

```
  <livre key= " K1" >
```

```
    <titre>T1</titre>
```

```
  </livre>
```

```
</bibliographie>
```



Exemple Programme DOM

9

```
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
```

```
public class Test {
    public static void main(String args[]) {
        try {
            //création du parseur en appelant la méthode statique factory
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            //chargement du fichier XML
            Document document = builder.parse("src/Test.xml");
            //récupérer le contenu de l'élément racine
            Element racine = document.getDocumentElement();
            Node textNode = racine.getFirstChild();
            System.out.println(textNode.getNodeValue());
        }
        catch (Exception e) {
            e.printStackTrace(System.out);
        }
    }
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<affichage>Bonjour tout le monde!</affichage>
```

Interface racine: Document

10

- Caractérise l'ensemble du document. Parmi les méthodes les plus utilisées de cette interface:
 - ❑ `getDocumentElement()`: retourne l'élément racine
 - ❑ `getElementById(String elementId)`: retourne l'élément dont un attribut de type ID contient la valeur `elementId`
 - ❑ `getElementsByTagName(String tagname)`: Retourne la liste des éléments ayant pour nom celui passé en argument
 - ❑ + méthodes de création des nœuds
 - `createElement`, `createTextNode`, `createAttribute...`

Opération sur les Nœuds

11

- Trois méthodes courantes sont utilisées:
 - ❑ getNodeName: retourne selon le type de nœud:
 - Nom de l'élément pour le type élément
 - Nom de l'attribut pour le type attribut
 - Nom de l'entité pour le type entité ou référence d'entité
 - Nom de IT pour le type instruction de traitement
 - ❑ getNodeValue: retourne selon le type du nœud:
 - Valeur de l'attribut pour le type attribut
 - Text lié au nœud Text ou CDATA)
 - ❑ getAttributes: retourne pour le nœud élément l'ensemble des attributs associés via l'objet NameNodeMap

Opération de lecture

12

- Parmi les opérations de lecture de l'arbre DOM:

Opération	Description
getChildNodes	Liste des nœuds fils.
getFirstChild	Le premier nœud fils
getLastChild	Le dernier nœud fils
getNamespaceURI	L'URI, si un espace de noms est utilisé
getNextSibling/getPreviousSibling	Le nœud adjacent (avant ou après)
getOwnerDocument	Le propriétaire du nœud
getParentNode	Le nœud parent

Opération de modification

13

- Parmi les opérations de modification :

Opération	Description
appendChild	Ajout d'un fils
cloneNode	retourner un clone d'un nœud
insertBefore	Insertion d'un nouveau nœud avant le fils
removeChild	suppression d'un nœud fils
replaceChild	remplacement d'un nœud fils par un nouveau

Interface Element

14

- Offre des méthodes pour:
 - ❑ récupérer le nom de l'élément
 - `getTagName`
 - ❑ récupérer les éléments descendants...
`getElementsByTagName`
 - ❑ Manipuler les attributs
 - `getAttribute`, `setAttribute`, `removeAttribute`

Interface Attr

15

- Parmi les méthodes de cette interface:
 - ❑ getName : Nom de l'attribut
 - ❑ getOwnerElement: Élément attaché
 - ❑ getValue : Valeure de l'attribut
 - ❑ setValue : Changement de la valeur de l'attribut

DOM

16

- Le parseur lit le document XML et le représente en mémoire sous forme d'un arbre.
- Pour un grand document, l'analyse prendra beaucoup de temps:
 - ❓ Besoin d'un thread séparé pour assurer l'interaction avec l'application



Simple API for XML: SAX

Simple API for XML: SAX

18

- Sax est une interface standard pour le parsing XML basé sur des déclencheurs (événements/action)
 - <http://www.saxproject.org/>
- L'API SaX est disponible pour la majorité de langage de programmation: C++, C#, Java, Pascal, Perl, PHP...
- Pour écrire une application SAX il faut:
 - ❑ un analyseur XML gérant SAX,
 - ❑ une version de SAX (ensemble de classes et interfaces),
 - ❑ un gestionnaire d'événements.

À l'aide de cette classe on crée
Une instance de parseur

Le gestionnaire par défaut
implémente les méthode de
rappel

contient toutes les
méthodes relatives
aux événements
déclenchés pendant
la lecture d'un
fichier

SaxParser
Factory

SAX
Parser

XML
READER

Content
Handler

Error
Handler

DTD
Handler

Entity
Resolver

DE
FA
UL
T

HA
ND
LE
R

Traitement
des erreurs

XML

Pour les DTD

Pour
résoudre les
références
externes

L'utilisateur fournit juste le code
des méthodes de rappel qu'il veut
implémenter par surcharge au
niveau des gestionnaires
spécifiques

Interfaces XMLReader & ContentHandler

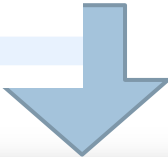
20

- L'interface XMLReader offre les méthodes d'initialisation des gestionnaires et de lancement du parseur
 - ❑ setContentHandler()
 - ❑ Parser()
- L'interface ContentHandler offre des méthodes de rappel pour la surcharge:
 - ❑ startDocument / end Document : notifiant début de document / fin de document
 - ❑ startElement /endElement : notifiant début de document / fin de document
 - ❑ Caracteres : pour recevoir une chaine de caractère

Exemple SAX 1 & Java & XML

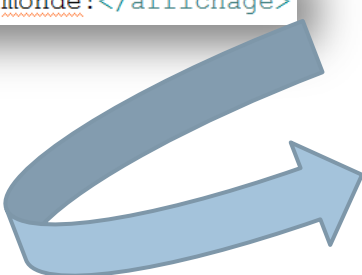
21

```
public class saxParser {  
  
    public static void main(String[] args) throws ParserConfigurationException {  
        // TODO Auto-generated method stub  
  
        try {  
            // creates and returns new instance of SAX-implementation:  
            SAXParserFactory factory = SAXParserFactory.newInstance();  
  
            // create SAX-parser...  
            SAXParser parser = factory.newSAXParser();  
            // .. define our handler:  
            TestSax1Handler handler = new TestSax1Handler();  
  
            // and parse:  
            parser.parse("src/Test.xml", handler);  
  
        } catch (Exception ex) {  
            ex.printStackTrace(System.out);  
        }  
    }  
}
```



```
<?xml version="1.0" encoding="UTF-8"?>  
<affichage>Bonjour tout le monde!</affichage>
```

```
public class TestSax1Handler extends DefaultHandler {  
  
    public void startDocument() throws SAXException {  
        System.out.println("Document processing started");  
    }  
  
    // notifies about finish of parsing:  
    public void endDocument() throws SAXException {  
        System.out.println("Document processing finished");  
    }  
  
    // we enter to element 'qName':  
    public void startElement(String uri, String localName,  
        String qName, Attributes attrs) throws SAXException {  
  
        System.out.println("** Début de l'élément: " + qName);  
    }  
  
    // we leave element 'qName' without any actions:  
    public void endElement(String uri, String localName, String qName)  
        throws SAXException {  
        // do nothing;  
        System.out.println("** Fin de l'élément: " + qName);  
    }  
  
    public void characters(char[] ch, int start, int length) throws SAXException {  
        String value = new String(ch, start, length).trim();  
  
        System.out.println("contenu: " + value);  
    }  
}
```



```
Document processing started  
** Début de l'élément: affichage  
contenu: Bonjour tout le monde!  
** Fin de l'élément: affichage  
Document processing finished
```

SAX 2

22

```
public class SAXPaeser2 {  
  
    public static void main(String[] args) throws SAXException, IOException {  
        // TODO Auto-generated method stub  
        // Création d'un parseur  
        XMLReader parser = XMLReaderFactory.createXMLReader();  
  
        TestSAX2Handler handler= new TestSAX2Handler();  
  
        // Association d'un gestionnaire de contenu et d'erreurs.  
        parser.setContentHandler(handler); //affecter le handler au parseur grâce  
        parser.setErrorHandler(handler);  
  
        //Pareser le document  
        parser.parse("src/Test.xml");  
    }  
}
```

```
public class TestSAX2Handler extends DefaultHandler{  
  
    public void startDocument() throws SAXException {  
        System.out.println("Document processing started");  
    }  
  
    // notifies about finish of parsing:  
    public void endDocument() throws SAXException {  
        System.out.println("Document processing finished");  
    }  
  
    // we enter to element 'qName':  
    public void startElement(String uri, String localName,  
        String qName, Attributes attrs) throws SAXException {  
  
        System.out.println("*** Début de l'élément: " + qName);  
    }  
  
    // we leave element 'qName' without any actions:  
    public void endElement(String uri, String localName, String qName)  
        throws SAXException {  
        // do nothing;  
        System.out.println("*** Fin de l'élément: " + qName);  
    }  
  
    public void characters(char[] ch, int start, int length) throws SAXException  
    {  
        String value = new String(ch, start, length).trim();  
  
        System.out.println("contenu: " + value);  
    }  
}
```

Document processing started
** Début de l'élément: affichage
contenu: Bonjour tout le monde!
** Fin de l'élément: affichage
Document processing finished

Comparaison DOM & SAX

23

	Forts	Faibles
DOM	<ul style="list-style-type: none">- parcours libre de l'arbrepossibilité de modifier la structure et le contenu de l'arbre	<ul style="list-style-type: none">- Un processus qui utilise le DOM ne peut traiter l'arbre qu'après la lecture entière du document- Gourmand en mémoire
SAX	<ul style="list-style-type: none">- Peut gourmand en ressources mémoire- Plus rapide pour les documents volumineux- Permet de ne traiter que les données utiles	<ul style="list-style-type: none">- Traite les données séquentiellement- plus difficile à programmer (nécessité de sauvegarder des informations pour les traiter par la suite)- Impossibilité de pratiquer des requêtes par XPath/Xquery nécessitant une représentation globale en mémoire

Quand utiliser SAX ou DOM ?

24

- DOM:

- ❑ Toutes les données doivent être utilisées

- ❑ Abondance de mémoire

- SAX:

- ❑ Traitement d'une partie des données

XQUERY XML QUERY LANGUAGE

XQuery

26

- XQUERY est un langage d'interrogation de documents XML développé par W3C
 - ❑ sert à interroger des arbres
 - ❑ sur-ensemble de Xpath
 - chaque expression XPath valide est également une expression XQuery valide
 - ❑ Permet de construire des morceaux de document
 - créer des nœuds et construire de nouveaux arbres

XQuery

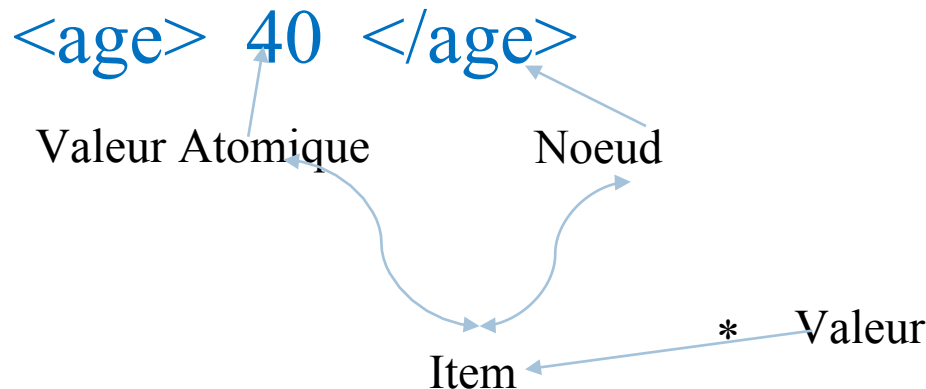
27

- XQuery permet :
 - ❑ Sélection d'arbres et de sous-arbres en utilisant des prédicats sur les valeurs des feuilles,
 - ❑ Utilisation de variables dans les requêtes pour mémoriser un arbre ou pour itérer sur des collections d'arbres,
 - ❑ Combinaison des arbres extraits en utilisant
 - des jointures d'arbres,
 - Ré-ordonnancement des arbres,
 - Imbrication de requêtes,
 - Calculs d'agrégats
 - ❑ Utilisation possible de fonctions définies par l'utilisateur

Modèle de données de XQuery

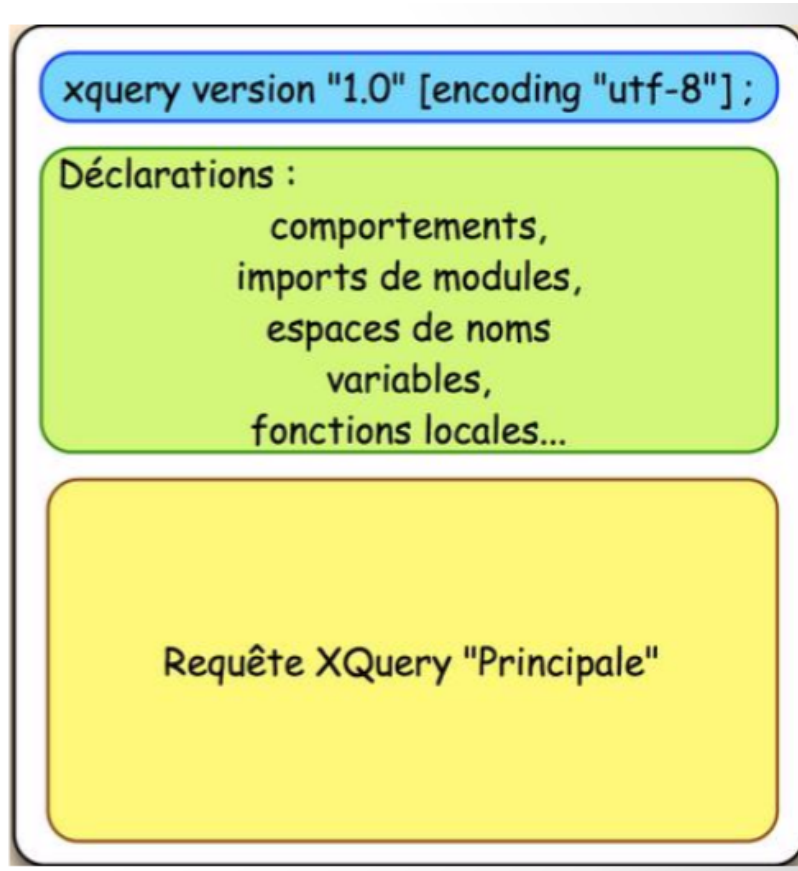
28

- Une valeur retournée par une expression XQuery est une séquence ordonnée d'items.
- Un item est un noeud ou une valeur atomique.



Forme d'une requête XQuery

29



Expression XQuery

30

- L'élément de base du langage XQuery est l'*expression*
- Une requête est **une composition d'expressions**
 - ❓ qui li une séquence de fragments XML ou de valeurs atomiques.
 - ❓ Retourne une séquence de fragments XML ou de valeurs atomiques.
- Une expression peut être:
 - ❓ Simple (valeurs atomiques, variable, opérateurs)
 - ❓ Complexe
 - XPATH : //autor
 - FLWR : For-Let-Where-Return
 - Tests : IF-Then-return-else-return
 - Fonctions : Racine, prédéfini, à définir
- Une requête prend généralement en entrée un document ou une collection de documents
 - ❓ doc("nomDocument.xml") →renvoie le nœud racine du document.
 - ❓ collection("nom") permet de récupérer une forêt de documents XML nommée nom et mémorisée dans un **SGBD-XML** et renvoie une séquence composées des nœuds racines des documents de la collection

Expressions de chemin

31

□ Expression de chemin

❓ Toutes les expressions XPath sont des expressions de Xquery et Utilisent les sélecteurs XPath

Selector	Selected nodes
/	Document root
//	Any sub-path
*	Any element
name	Element of tag name
@*	Any attribute
@name	Attribute of name name
text()	Any text node
processing-instruction('name')	Processing instruction of given name
comment()	Any comment node
node()	Any node
id('value')	Element of id value

Exemple de requête XQuery

32

XQuery Requête

```
<bib>
  <book year="1994">
    <title>TCP/IP Illustrated</title>
    <author><last>Stevens</last><first>W.</first></author>
    <publisher>Addison-Wesley</publisher>
    <price>65.95</price>
  </book>
  <book year="1992">
    <title>Advanced Programming in the Unix environment</title>
    <author><last>Stevens</last><first>W.</first></author>
    <publisher>Addison-Wesley</publisher>
    <price>65.95</price>
  </book>
  <book year="2000">
    <title>Data on the Web</title>
    <author><last>Abiteboul</last><first>Serge</first></author>
    <author><last>Buneman</last><first>Peter</first></author>
    <author><last>Suciu</last><first>Dan</first></author>
    <publisher>Morgan Kaufmann Publishers</publisher>
    <price>39.95</price>
  </book>
</bib>
```

```
for $h in collection("test1")//book
return $h/title
```

doc("biblio.xml")//author

Résultat

```
<author><last>Stevens</last><first>W.</first></author>
<author><last>Stevens</last><first>W.</first></author>
<author><last>Abiteboul</last><first>Serge</first></author>
<author><last>Buneman</last><first>Peter</first></author>
<author><last>Suciu</last><first>Dan</first></author>
```

```
let $book := doc("biblio.xml")//book
return $book/title
```

```
<title>TCP/IP Illustrated</title>
<title>Advanced Programming in the Unix environment</title>
<title>Data on the Web</title>
```

```
<title>TCP/IP Illustrated</title>
<title>Advanced Programming in the Unix environment</title>
<title>Data on the Web</title>
<title>TCP/IP Illustrated</title>
<title>Advanced Programming in the Unix environment</title>
<title>Data on the Web</title>
<title>The Economics of Technology and Content for Digital TV</title>
```


Expression XQuery

33

□ Construction d'élément

❑ 1^{er} situation : Le nom de l'élément est connu et le contenu sera construit par une expression

■ Requête

```
<auteur>
{doc("biblio.xml")//book[3]/author/last}
</auteur>
```

■ Résultat:

```
<auteur>
<last>Abiteboul</last>
<last>Buneman</last>
<last>Suciu</last>
</auteur>
```

Les accolades sont obligatoires sinon l'expression est prise pour du texte.

Expression XQuery

34

□ Construction de

❓ 2ème situation :

■ Constructeurs

- element {expr-nom} {expr-contenu}
- attribute {expr-nom} {expr-contenu}

■ Requête

```
element { doc("biblio.xml")//book[1]/name(*[1]) } {  
  attribute { doc("biblio.xml")//book[1]/name(@*[1]) }  
    { doc("biblio.xml")//book[1]/@*[1] },  
  doc("biblio.xml")//book[1]/*[1]/text()  
}
```

■ Résultat:

```
<title year="1994">TCP/IP Illustrated</title>
```

```
<bib>  
  <book year="1994">  
    <title>TCP/IP Illustrated</title>  
    <author><last>Stevens</last><first>W.</  
first></author>  
    <publisher>Addison-Wesley</publisher>  
    <price>65.95</price>  
  </book>  
</bib>
```

Expression FLWOR

35

□ Une expression FLWOR

- Itère sur des séquences (**F**or)
 - for \$v in expr
 - Génère un flux composé de variables liées aux valeurs prises dans l'ensemble résultant de l'évaluation de expr
- Affecte des valeurs à une variable (**L**et)
 - let \$v := expr
 - Génère un flux composé d'une variable liée à une seule valeur (qui peut être une séquence). la variable \$x prend la valeur de la séquence retournée par Expr.
- Applique des filtres - restriction (**W**here)
 - where condition
 - Filtre le flux
- Trie les résultats (**O**rders by)
- Retourne la valeur de la requête (**R**eturn)

Exemple: Expression FLWOR à changer cet ex

36

```
for $d in doc("bib.xml")//book
let $e := doc("biblio.xml")//book[. = $d]
where
  $e/price[.>40]
order by $e/title[.] ascending
return
  | $e/title
```



```
<title>Advanced Programming in the Unix environment</title>
|<title>TCP/IP Illustrated</title>
```

Requêtes motif d'arbre

37

```
xquery version "1.0";  
for $b in doc("bib.xml")//book, $a in $b/author  
where $a/last="Stevens"  
return  
$b/title|
```

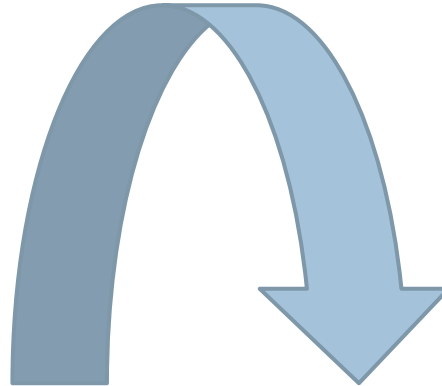


```
<title>TCP/IP Illustrated</title>  
|<title>Advanced Programming in the Unix environment</title>
```

Tests: if-then-else

38

```
<livres>
{ for $b in doc("bib.xml")//book
  where $b/author/last = "Stevens"
  return if ($b/@year >= 1994)
then <livre recent="true">
  {$b/title}
</livre>
else <livre> {$b/title} </livre>
}
</livres>
```



```
<livres>
<livre recent="true">
  <title>TCP/IP Illustrated</title></livre>
<livre>
  <title>Advanced Programming in the Unix environment</title>
</livre>
</livres>
```

Quantification

39

- `some $var in expr1 satisfies expr2`
 - ❓ il existe au moins un noeud retourné par l'expression `expr1` qui satisfait l'expression `expr2` (quantification existentielle).
- `every $var in expr1 satisfies expr2`
 - ❓ tous les nœuds retournés par l'expression `expr1` satisfont l'expression `expr2` (quantification universelle)

Example : Jointure

40

```
for $b in doc("bib.xml")//book
return element livre {
  attribute titre {$b/title},
  for $a in $b/author
  return element auteur {
    attribute nom {$a/last},
    for $p in doc("adresse.xml")//person
    where $a/last = $p/name
    return attribute institut {$p/institution}
  }
}
```

```
<livre titre="TCP/IP Illustrated"><auteur nom="Stevens" institut="New Haven"/></livre>
<livre titre="Advanced Programming in the Unix environment">
<auteur nom="Stevens" institut="New Haven"/></livre>
<livre titre="Data on the Web"><auteur nom="Abiteboul" institut="INRIA"/>
<auteur nom="Buneman" institut="LFCS"/>
<auteur nom="Suciu"/>
</livre>
<livre titre="The Economics of Technology and Content for Digital TV"/>
```



```
<addresses>
<person>
  <name>Stevens</name>
  <country>American</country>
  <institution>New Haven</institution>
</person>
<person>
  <name>Abiteboul</name>
  <country>France</country>
  <institution>INRIA</institution>
</person>
<person>
  <name>Buneman</name>
  <country>UK</country>
  <institution>LFCS</institution>
</person>
</addresses>
```


Opérations

41

- Opérateurs arithmétiques: $1+2$, $3.4-6.5$, $\$y \bmod 2$
- Opérations sur les séquences :
 - concaténation
 - union, intersection, différence de séquences de noeuds
- Opérateurs de comparaison pour valeurs atomiques, noeuds et séquences
- Opérations booléennes: $\$x=2$ and $\$y=4$ or not($\$z$)

Xquery : Fonction

42

□ Fonctions Prédéfinies

- ❓ fonctions min, max, count, sum et avg analogues a celles de SQL.
- ❓ fonctions numériques comme round, floor (plus grand entier à l'expression numérique spécifiée) et ceiling (plus petit entier supérieur ou égal à l'expression numérique spécifiée)
- ❓ fonctions des chaînes de caractères comme concat, string-length, starts-with, end-with, substring, upper-case, lower-case.
- ❓ Autres : distinct-values, doc, not, empty, exists, except...

Exemples de fonctions prédéfinies

43

```
let $b := doc("bib.xml")//book  
let $avg := avg( $b//price )  
return $b[price > $avg]
```

livres sont plus chers que la moyenne

```
for $b in doc("books.xml")//book where  
not(some $a in $b/author satisfies $a/  
last="Stevens") return $b
```

livres dont aucun auteur s'appelle Stevens

```
for $b in doc("books.xml")//book where  
not(empty($b/author)) return $b
```

livres qui ont au moins un auteur

XQuery : Fonction

44

□ Fonctions définies par l'utilisateur

```
declare function prefix:function_name($parameter as  
datatype)  
as return_Datatype  
{  
  ...function code here...  
};
```

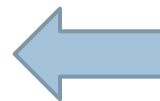
local:reverse ((1, 2, 3, 4, 5))

Local="http://www.w3.org/2005/xquery-local-functions" → par défaut

<result>54321</result>

```
declare namespace local  
= "http://www.irisa.fr/  
bekkers/test";
```

```
declare function local:reverse  
($items xs:integer+) {  
  let $count := count($items as  
item()+)  
  for $i in 0 to $count  
  return $items[$count - $i]  
};  
<result> {  
  local:reverse(1 to 5)  
}  
</result>
```



NB: Le nom de la fonction doit être préfixé. Il existe un préfixe créé par défaut : local.

XML et SGBD

45

□ XML est-il une base de données ?

- ❑ Le stockage (les documents XML)
- ❑ les schémas (DTD, XML Schemas....)
- ❑ des langages de requête (XQuery, XPath, ...)
- ❑ des interfaces de programmation (SAX, DOM)
- ❑ ...

Mais on a des limitations

- ❑ Le stockage efficace, les index, la sécurité, les transactions et l'intégrité des données, l'accès multi-utilisateur, les déclencheurs [*triggers*]...

Types de bases de données XML

46

□ SGBD relationnelle étendu (SGBDR \leftrightarrow XML):

Depuis 2000 la plupart des moteurs relationnels ont intégré le support de l'XML « XML-Enabled » pour transformer les données XML et les mémoriser avec un SGBD relationnel (tables classiques)

- Définition d'un schéma relationnel pour stocker des documents XML
- Interrogation avec SQL

XML et SGBD

47

- Deux solutions pour stocker les données XML
 - ❑ les données sont transformées (divers degrés de transformation sont possibles) et mémorisées avec un SGBD relationnel
 - ❑ Les données sont intégrées directement en XML

BD XML Natives

48

- Les bases de données XML natives (en anglais NXD pour « Native XML Database ») sont des bases de données réalisées pour le stockage de données XML.
- Elles offrent à l'utilisateur une vision logique des données en accord avec le modèle XML (organisation hiérarchique des informations...)
 - de la même façon que les bases de données relationnelles présentent une vision logique des données conforme au modèle relationnel (organisation des informations sous forme de tables).
- Cette vision logique conforme au modèle XML permet d'envisager aisément d'utiliser les standards définis autour de XML (XQuery, XPath, XSLT, XUpdate) pour accéder et traiter les données de la base.

XML et SGBD

49

□ Plusieurs SGBD XML existent:

- ❑ Le SGBD-XML **eXist** est un système open-source sous licence GNU. Il est utilisable sur toutes les plate-formes courantes (Linux, Mac OS ou Windows). Il exploite de nombreux standard tels que XQuery, XSLT, XPath, XUpdate, etc. Facile à installer, il supporte l'accès concurrent et optimise l'accès aux données par une indexation automatique des données.
- ❑ **Sedna** est un SGBD-XML open-source sous licence Apache License 2.0 et développé en C/C++. elle possède des API pour de très nombreux langages (PHP, Java, C, Scheme, Python, etc.). Plus ouverte en API, elle ne permet pas de construire aussi facilement des applications Web comme la précédente, même si elle peut s'intégrer sur un serveur Apache.
- ❑ **BaseX** SGBD-XML open-source en licence BSD(*Berkeley Software Distribution License*) développé en Java et utilisable sur toutes les plate-formes (Linux, Mac OS ou Windows). Il possède des API aussi bien propriétaires que "classiques" (REST, XML:DB) et respecte de nombreux standard XML récents dont XQuery Full-Text et XQuery Update Facility.

Quel choix de stockage ?

50

Vous voulez stocker des *données* ou des *documents*??

❓ Document centré donnée (data centric document). XML est utilisé simplement en tant que vecteur de données entre la base et une application (qui peut d'ailleurs ne pas être XML).

Ou bien

❓ document centré contenu (document centric document). XML est-il exploité intégralement?

Webographie

51

- Livres

Livre XML cours et Exercice :

<https://campusbruxelles.files.wordpress.com/2014/02/2007-eyrolles-xml-cours-et-exercices.pdf>

- Espace de nom :

<http://deptinfo.unice.fr/twiki/pub/Minfo03/ServletEtXml/01c-xml-namespaces.pdf>

- DTD :

http://www2.amk.fi/digma.fi/www.amk.fi/material/attachments/vanhaamk/digma/5h5F5r4b3/DTD_0.7.swf

- XPATH:

<http://www.w3.org/TR/xpath/>

<http://www.loria.fr/~abelaid/Enseignement/miage-m1/XSL-Xpath.pdf>

- XSLT:

http://miage.univ-nantes.fr/miage/D2X1/chapitre_xslt/section_regles.htm#22

- DOM et SAX

<http://deptinfo.unice.fr/twiki/pub/Minfo03/ServletEtXml/06-xml-dom-sax.pdf>

<https://www.lri.fr/~benzaken/documents/sldomsax.pdf>

Webographie

52

- Livre XML cours et Exercice : <https://campusbruxelles.files.wordpress.com/2014/02/2007-eyrolles-xml-cours-et-exercices.pdf>
 - <http://www.telug.ca/inf6450/mod2/chapitre6.xml>
 - Espace de nom : <http://deptinfo.unice.fr/twiki/pub/Minfo03/ServletEtXml/01c-xml-namespaces.pdf>
 - Dtd : http://www2.amk.fi/digma.fi/www.amk.fi/material/attachments/vanhaamk/digma/5h5F5r4b3/DTD_0.7.swf
 - DOM et SAX <http://deptinfo.unice.fr/twiki/pub/Minfo03/ServletEtXml/06-xml-dom-sax.pdf>
 - ? <https://www.lri.fr/~benzaken/documents/sldomsax.pdf>
 - ? <http://www-lium.univ-lemans.fr/~lehuen/master1/xml/cours/xmldiapo3.pdf>
- XQUERY
 - ? Chapitre 6 de base de donn es avanc es XML-Requ te Xquery, Yannik Benezeth
<http://ilt.u-bourgogne.fr/benezeth/teaching/DUTIQ/BDDS3/CM6.pdf>
 - ? Xquery : Une extension de Xpath   la mode SQL Un concurrent d'XSLT ?, Yves Bekkers
https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=18&cad=rja&uact=8&ved=0ahUKEwjFw83x6NfQAhXGbRQKHVG2DQ04ChAWCE0wBw&url=http%3A%2F%2Fwww.irisa.fr%2FLIS%2FMembers%2Fbekkers%2Fxmldossier%2Fxquery%2Fattachment_download%2Ffile&usg=AFQjCNE1Ih2QwPJ21fNx68jdzdZtpuOdQw&sig2=63NIKIMEdlwb4dPffFxe5w
 - ? https://www.ibisc.univ-evry.fr/~serena/coursBDA0910_4.pdf
- XML et SGBD
 - ? http://miage.univ-nantes.fr/miage/D2X1/chapitre_bdxml/section_sgbd_xml.htm
 - ? <http://peccatte.karefil.com/software/rbouret/xmlbd.htm>
 - ? <http://code.ulb.ac.be/dbfiles/Ver2006amastersthesis.pdf>