<*XML num="3"*>Langage Web 2</*XML*>

- Emilien Bondu (Airbus Defence and Space)
- emilien.bondu@cassidian.com

Sommaire

- Cours 1 : Introduction à XML et validation (XSD)
- Cours 2 : Validation (DTD, Relax NG, Schematron) et transformation (XPath, XSL; XSL-FO)
- Cours 3 : Recherche XML (XQuery), Base de données XML, Liens XML, Manipulation XML en Java: Dom, Sax
- Cours 4 : Manipulation XML en Java : StAX, Data-binding, JavaEE : tomcat
- Cours 5 : Manipulation XML en JavaEE: servlet, JSP, ExpressionLanguage, TagLib
- Cours 6 : TagLib (suite), Spring, JSF, AJAX

Rappels

- XPath est :
  - un format basé sur XML
  - une syntaxe pour désigner des parties de documents XML
  - un standard W3C
  - utilisé dans XSL

Rappels

- XPath permet :
  - de récupérer une valeur d'attribut
  - de récupérer un contenu d'élément
  - de récupérer un document dans une collection XML
  - de récupérer un sous arbre XML

Rappels

- XSL est utilisé pour :
  - transformer des fichiers XML
  - produire des fichiers XHTML
  - produire des fichiers XML
  - produire des fichiers PDF

Rappels

- XSL permet de :
  - copier des bouts de fichier XML
  - définir des conditions, des boucles
  - définir des entités XML
  - définir des templates de traitement

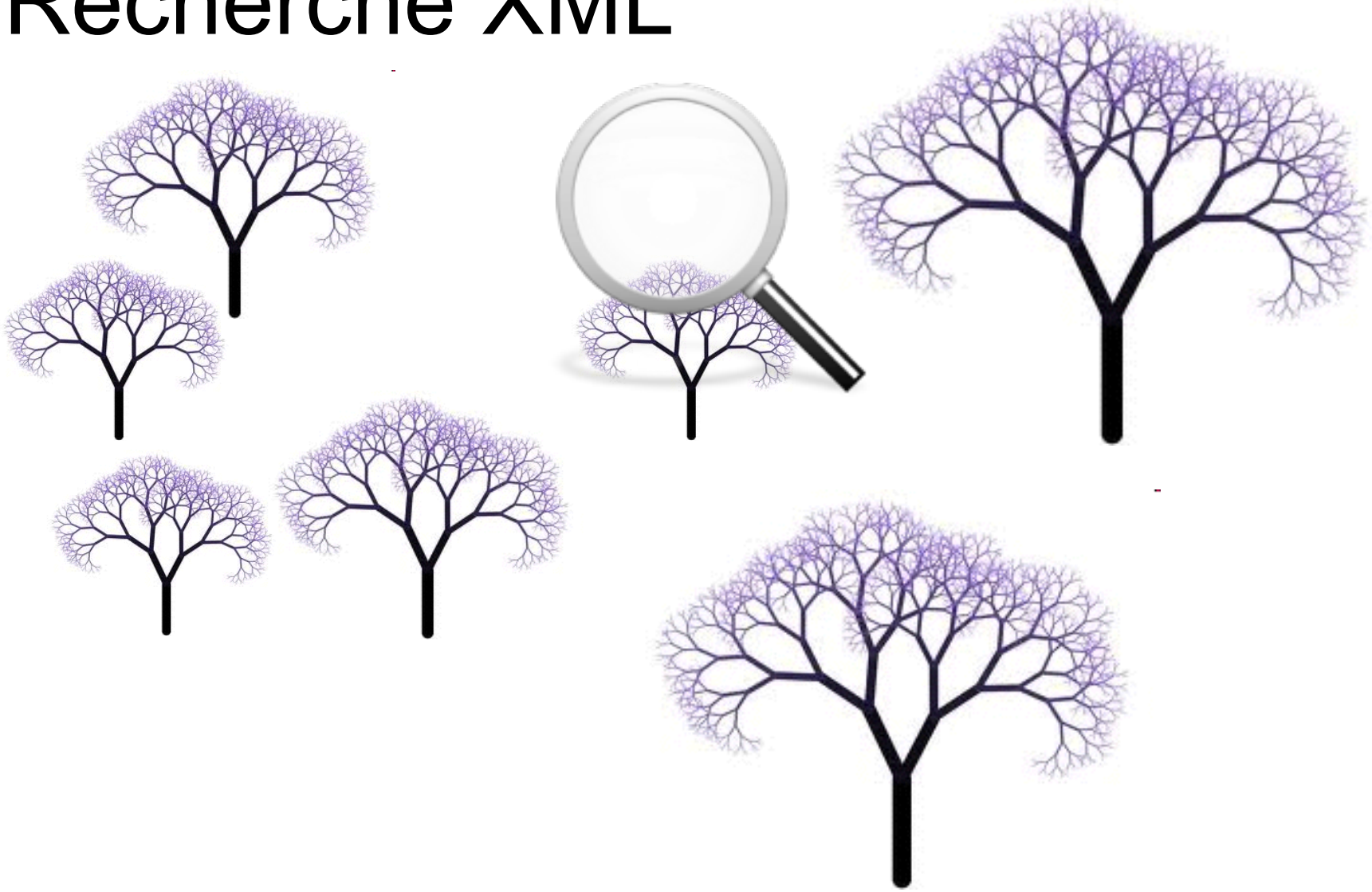
Rappels

- XSL-FO est utilisé pour :
  - transformer des fichiers XML
  - produire des fichiers PNG
  - produire des fichiers XML
  - produire des fichiers PDF

Rappels

- XSL-FO permet :
  - de définir des modèles de présentation
  - de définir quelles informations sont affichées avec un modèle de présentation donné
  - de créer des objets graphiques (table)
  - de définir des calques pour l'affichage des informations

Recherche XML



Recherche XML

- Objectif :
  - Retrouver des documents XML
  - Stocker des documents XML
- Standards associés
  - XQuery

XQuery

- XQuery est :
  - le SQL du XML
  - utile à la fois pour interroger des données structurées ou non
  - indépendant des protocoles propriétaires
  - un langage fonctionnel
  - fortement typé (offre des possibilités d'optimisation et une meilleure détection des erreurs)

« The mission of the XML Query project is to provide flexible query facilities to extract data from real and virtual documents on the World Wide Web, therefore finally providing the needed interaction between the Web world and the database world. Ultimately, collections of XML files will be accessed like databases »

J. Robie

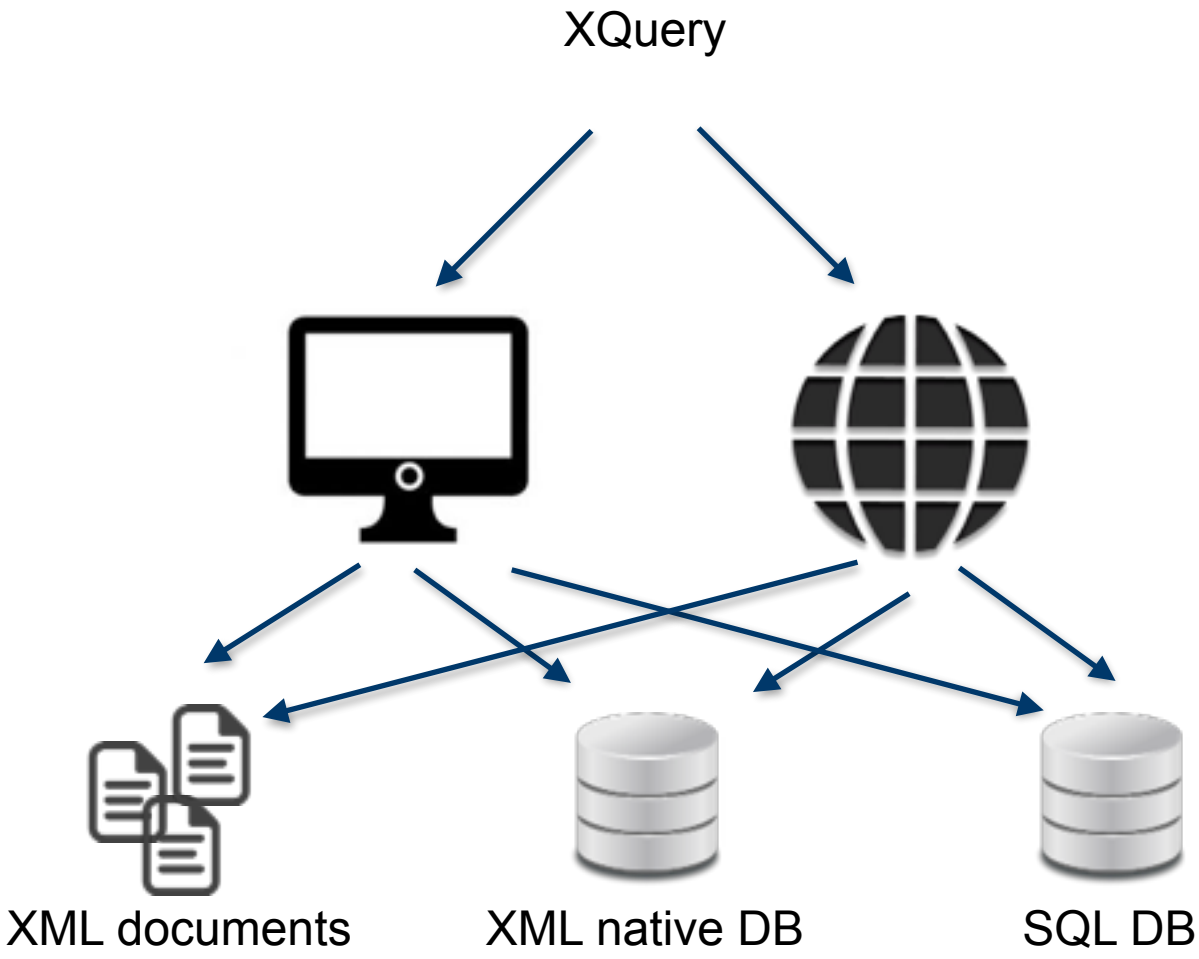
XQuery

- XQuery pour bénéficier des avantages des SGBD dans le cas du XML
  - performance
  - transactions
  - sécurité

« The mission of the XML Query project is to provide flexible query facilities to extract data from real and virtual documents on the World Wide Web, therefore finally providing the needed interaction between the Web world and the database world. Ultimately, collections of XML files will be accessed like databases »

J. Robie

XQuery

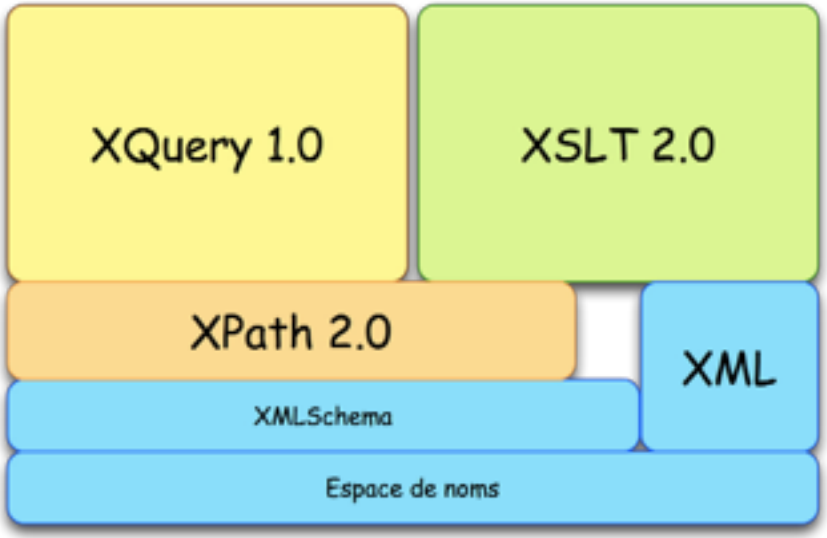


XQuery VS XPath

- XPath est :
  - utile pour naviguer dans des arbres XML
  - un sous-ensemble de XQuery
- XQuery est :
  - utile pour requêter des arbres XML
  - une extension de Xpath
  - dédié (optimisé) pour de grands ensembles de documents

XQuery VS XPath / XSLT

- XPath/XSLT et XQuery permettent :
  - d'extraire des informations de documents XML
  - d'effectuer des calculs sur les informations extraites
  - de reconstruire des documents ou fragments XML

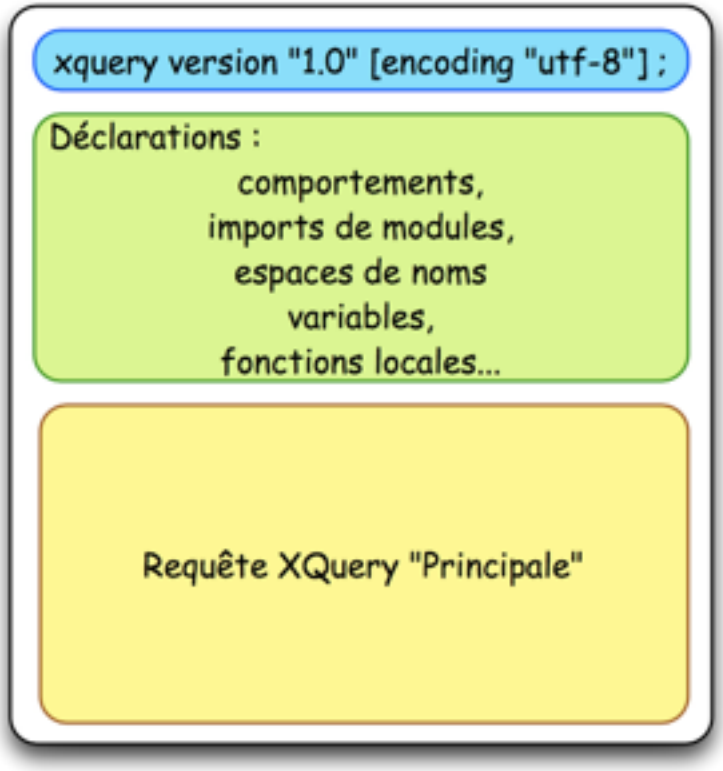


XQuery VS XPath

- XPath et XQuery = même modèle de données
- les valeurs (items) manipulées sont des séquences
- les items sont soient :
  - noeuds XML : document, élément, attribut, texte, namespace, instruction (processing-instruction), commentaires
  - types de base : entier, décimaux, etc.

XQuery « Building Blocks »

- Parties d'une requête



XQuery « Building Blocks »

- Séquences
  - (item1, item2, item3)

```
(1,2,4) (47, <title/>, "toto")
```

  - générateurs (fonctions)
    - to(xs:integrer, xs:integer)

```
(1 to 5)
```

    - doc(xs:string)

```
doc("book.xml")
```

    - collection(xs:string)

```
collection("/db/books")
```

XQuery « Building Blocks »

- Expressions
  - même modèle que XPath
  - même fonctions que XPath

```
xquery version "1.0";

declare namespace livre="http://livre.org";

collection('livres')/livre:livre/livre:titre/text()
```

livre1.xml

```
<livre xmlns="http://livre.org"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="http://livre.org livre.xsd">

        <titre>Apprendre le XQuery</titre>

        <auteur>
                <nom>Lee</nom>
                <prenom>Tom</prenom>
        </auteur>
</livre
```

livre2.xml

```
<livre xmlns="http://livre.org"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="http://livre.org livre.xsd">

        <titre>Apprendre le XML</titre>

        <auteur>
                <nom>Bondu</nom>
                <prenom>Emilien</prenom>
        </auteur>
</livre
```

```
Apprendre le XMLApprendre le XQuery
```

XQuery « Building Blocks »

- Opérateurs (identique à XPath)
  - (=, >, <, +, -, /, …)
  - **union, intersect, except**

```
xquery version "1.0";

declare namespace livre="http://livre.org";

collection('livres')/livre:livre[livre:auteur/livre:nom = 'Lee']/livre:titre
```

```
<titre xmlns="http://livre.org" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">Apprendre le XQuery</titre>
```

XQuery « Building Blocks »

- Constructeur direct
  - {expression}

```
xquery version "1.0";

declare namespace livre="http://livre.org";

<livres total="{count(collection('livres'))}">
    {
        collection('livres')/livre:livre[livre:auteur/livre:nom = 'Lee']/livre:titre/text()
    }
</livres>
```

```
<?xml version="1.0" encoding="UTF-8"?><livre total="2">Apprendre le XQuery</livre>
```

XQuery « Building Blocks »

- Constructeur calculé
  - element name {expression}
  - attribute name {expression}

```
xquery version "1.0";

declare namespace livre="http://livre.org";

element livres {
    attribute total {count(collection('livres'))},

    collection('livres')/livre:livre[livre:auteur/livre:nom = 'Lee']/livre:titre/text()

}
```

```
<?xml version="1.0" encoding="UTF-8"?><livre total="2">Apprendre le XQuery</livre>
```

XQuery « Building Blocks »

- Syntaxe de type FLWOR
  - **F**or – **L**et – **W**here – **O**rder – **R**eturn

```
FOR $<var> in <sequence>,          // itération
LET $<var2> := <subtree>           // assignation
WHERE <condition>                  // élagage
ORDER <ordering-condition>         // ordonnancement
RETURN <result>                    // construction
```

XQuery « Building Blocks »

- Syntaxe de type FLWOR
  - **F**or – **L**et – **W**here – **O**rder – **R**eturn

```
xquery version "1.0";

declare namespace livre="http://livre.org";

<titres>
  {
    for $livre in collection('livres')/livre:livre
    where $livre/livre:auteur/livre:nom = 'Lee'
    return $livre/livre:titre
  }
</titres>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<titres>
       <titre xmlns="http://livre.org" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">Apprendre le XPath</titre>
</titres>
```

XQuery « Building Blocks »

- Syntaxe de type FLWOR
  - **F**or – **L**et – **W**here – **O**rder – **R**eturn

```
xquery version "1.0";

declare namespace livre="http://livre.org";

<titres>
    {
    for $livre in collection('livres')/livre:livre
    let $nbLivre := count(collection('livres')/livre:livre)
    where count($livre/livre:chapitre) >= 2
    return <livre total="{$nbLivre}" >{$livre/livre:titre/text()} ({count($livre/livre:chapitre)} Chapitres) </livre>
    }
</titres>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<titres>
      <livre total="4">Apprendre le XML (3 Chapitres) </livre>
      <livre total="4">Apprendre le XPath (2 Chapitres) </livre>
      <livre total="4">Apprendre le XQuery (5 Chapitres) </livre>
</titres>
```

XQuery « Building Blocks »

- Syntaxe de type FLWOR
  - **F**or – **L**et – **W**here – **O**rder – **R**eturn

```
xquery version "1.0";

declare namespace livre="http://livre.org";

<titres>
    {
    for $livre in collection('livres')/livre:livre
    let $nbLivre := count(collection('livres')/livre:livre)
    where count($livre/livre:chapitre) >= 2
    order by count($livre/livre:chapitre) descending
    return <livre total="{$nbLivre}" >{$livre/livre:titre/text()} ({count($livre/livre:chapitre)} Chapitres) </livre>
    }
</titres>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<titres>
      <livre total="4">Apprendre le XQuery (5 Chapitres) </livre>
      <livre total="4">Apprendre le XML (3 Chapitres) </livre>
      <livre total="4">Apprendre le XPath (2 Chapitres) </livre>
</titres>
```

XQuery « Building Blocks »

- Conditions
  - if … then … else

```
<titres>
    {
    for $livre in collection('livres')/livre:livre
    where count($livre/livre:chapitre) >= 2
    order by count($livre/livre:chapitre) descending
    return if (count($livre/livre:chapitre) = 2)
           then <livreDeuxChapitre>{$livre/livre:titre/text()}</livreDeuxChapitre>
           else <livrePlusDeuxChapitre>{$livre/livre:titre/text()}</livrePlusDeuxChapitre>
    }
</titres>
```

```
<titres>
     <livrePlusDeuxChapitre>Apprendre le XQuery</livrePlusDeuxChapitre>
     <livrePlusDeuxChapitre>Apprendre le XML</livrePlusDeuxChapitre>
     <livreDeuxChapitre>Apprendre le XPath</livreDeuxChapitre>
</titres>
```

XQuery « Building Blocks »

- Conditions
  - typeswitch … case … default

```
<titres>
    {
    for $noeud in collection('livres')/livre:livre/*
    return typeswitch($noeud)
        case $a as element (livre:auteur) return <livreAuteur>{string($a/livre:prenom)}</livreAuteur>
        case $a as element (livre:titre) return <livreTitre>{string($a)}</livreTitre>
        case $a as element (livre:chapitre) return <livreChapitre>{string($a/@num)}</livreChapitre>
        default return <inconnu></inconnu>
    }
</titres>
```

```
<titres>
        <livreTitre>Apprendre le XML</livreTitre>
        <livreAuteur>Emilien</livreAuteur>
        <livreChapitre>1</livreChapitre>
        <livreChapitre>2</livreChapitre>
        <livreChapitre>3</livreChapitre>
        <livreTitre>Apprendre le XPath</livreTitre>
        <livreAuteur>Tom</livreAuteur>
        <livreChapitre>1</livreChapitre>
        <livreChapitre>2</livreChapitre>
        <livreTitre>Apprendre le XQuery</livreTitre>
        <livreAuteur>Tom</livreAuteur>
        <livreChapitre>1</livreChapitre>
</titres>
```

XQuery « Building Blocks »

- Requêtes imbriquées

```
declare namespace livre="http://livre.org";

<titres>
{
    for $livre in collection('livres')/livre:livre
    where count($livre/livre:chapitre) >= 2
    order by count($livre/livre:chapitre) descending
    return
    <chapitres>
    {
        for $chapitre in $livre/livre:chapitre
        return <chapitre>{$chapitre/livre:titre}</chapitre>
    }
    </chapitres>
}</titres>
```

```
<titres>
       <chapitres>
              <chapitre>Les Bases d'XML</chapitre>
              <chapitre>La transformation XML</chapitre>
              <chapitre>La transformation XML</chapitre>
       </chapitres>
       <chapitres>
              <chapitre>Les Bases d'XML</chapitre>
              <chapitre>La transformation XML</chapitre>
       </chapitres>
</titres
```

XQuery « Building Blocks »

- Jointures

```
declare namespace livre="http://livre.org";
declare namespace annuaire="http://annuaire.org";

declare function livre:resume($chap as element(livre:chapitre)) as xs:string {
      concat($chap/livre:titre/text(), ":", substring($chap/livre:paragraph/livre:corps/text(),0, 15), "...")
};

<auteurs>
{
for $livre in collection("livres")/livre:livre, $personne in collection("personnes")/annuaire:personne
where  $livre/livre:auteur/livre:nom = $personne/annuaire:nom
and $livre/livre:auteur/livre:prenom = $personne/annuaire:prenom
return
      <auteur>
          {$personne/annuaire:nom}
          {$personne/annuaire:telephone}
          {$personne/annuaire:email}
      </auteur>
}
</auteurs>
```

XQuery « Building Blocks »

- Fonctions

```
declare function prefix:function_name($parameter as datatype) as returnDatatype
{
 code de la fonction + retour
}
```

```
declare namespace livre="http://livre.org";

(: fonction définie par l'utilisateur :)
declare function livre:resumeChap($chap as element(livre:chapitre)) as xs:string {
    concat($chap/livre:titre/text(), ":", substring($chap/livre:paragraphe[1]/livre:corps/text(), 1, 15), "...")
};

<titres>
    {
    for $noeud in collection('livres')/livre:livre/*
    return typeswitch($noeud)
        case $a as element (livre:auteur) return <livreAuteur>{string($a/livre:prenom)}</livreAuteur>
        case $a as element (livre:titre) return <livreTitre>{string($a)}</livreTitre>
        case $a as element (livre:chapitre) return <livreChapitre>{livre:resumeChap($a)}</livreChapitre>
        default return <inconnu></inconnu>
    }
</titres>
```

XQuery : Fonctions intégrées

- Fonctions sur les chaînes :substring, contains, matches, concat, normalize-space, tokenize
- Fonctions sur les dates : current-date, month-from-date, adjust-time-to-timezone
- Fonctions sur les nombres : round, avg, sum, ceiling
- Fonctions sur les séquences : index-of, insert-before, reverse, subsequence, distinct-values,
- Fonctions sur les noeuds : data, empty, exists, id, idref
- Fonctions sur les noms d'élément : local-name, in-scope-prefixes, QName, resolve-QName
- Rattrapage d'erreur et manipulation d'erreur : error, trace, exactly-one
- Fonctions définie par l'utilisateur

Extensions : XQuery Full-Text

- Recherche sur mot-clés
- Recherche de phrase
- Support des mots de liaison
- Recherche sur préfix, suffix, infix
- Normalisation des mots, accents, capitales, …
- Recherche par proximité (unité = mots)
- Spécification de l'ordre des mots
- Combinaison logic avec AND, OR , NOT
- Recherche par similarité
- Tri des résultats par pertinence

Extensions : XQuery Full-Text

```
<livres>
{
    for $livre in collection('livres')/livre:livre
    where $livre/livre:chapitre/livre:paragraphe/livre:corps/text() contains text "XML" occurs at least 2 times
    return
    <livre>{$livre/livre:titre/text()}</livre>
}
</livres>
```

Extensions : XQuery Update

- Syntaxe
  - update insert *xml_content* into *xpath (or variable)*
  - update replace *xpath* with *xml_content*
  - update value *xpath* with *value*
  - update delete *xpath*

```
for $livre in collection('livres')/livre:livre

   return
   update value $livre/livre:auteur/livre:nom[text() = "Andrew"] with 'Andy'
```

À VOUS DE JOUER!

XQuery - Exercice

- Tous les *films* (dans l'ordre alphabétique)

- Tous les noms de *films* sortis en 2011

- Tous les *films* avec plus d'une séance (à Rouen)

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<catalogue>
 <cinemas>
  <cinema>
   <nom>UGC</nom>
   <adresse>
    <ville>Rouen</ville>
    <lieu>Saint Sever</lieu>
   </adresse>
   <seance heure="16h" ref_film="1"/>
   <seance heure="20h" ref_film="1"/>
   <seance heure="20h" ref_film="3"/>
   <seance heure="20h" ref_film="4"/>
   <seance heure="20h" ref_film="5"/>
  </cinema>
  <cinema>
   <nom>Pathe</nom>
   <adresse>
    <ville>Rouen</ville>
    <lieu>Dock de Rouen</lieu>
   </adresse>
   <seance heure="18h" ref_film="1"/>
   <seance heure="20h" ref_film="1"/>
   <seance heure="21h" ref_film="2"/>
   <seance heure="22h" ref_film="4"/>
   <seance heure="20h" ref_film="5"/>
   <seance heure="23h" ref_film="5"/>
  </cinema>
 </cinemas>
 <films>
  <film film_id="1">
   <titre>Titeuf, le film</titre>
   <année>2011</année>
  </film>
  <film film_id="2">
   <titre>Scream 4</titre>
   <année>2011</année>
  </film>
  <film film_id="3">
   <titre>Source Code</titre>
   <année>2011</année>
  </film>
  <film film_id="4">
   <titre>Rango</titre>
   <année>2011</année>
  </film>
  <film film_id="5">
   <titre>Sucker Punch</titre>
   <année>2011</année>
  </film>
 </films>
</catalogue>
```

XQuery - Corrigé

- Tous les *films* (dans l'ordre alphabétique)

```
for $x in doc("catalogue.xml")//film
order by $x/titre
return $x
```

- Tous les *films* sortis en 2011

```
for $x in doc("catalogue.xml")//film
where $x/année="2011"
return $x/titre
```

- Tous les *films* avec plus d'une séance (à Rouen)

```
for $x in doc("catalogue.xml")//film
where count(//seance[@ref_film=$x/@film_id])>1
return $x
```

```
<?xml version="1.0" encoding="UTF-8" standalone=
<catalogue>
 <cinemas>
  <cinema>
   <nom>UGC</nom>
   <adresse>
    <ville>Rouen</ville>
    <lieu>Saint Sever</lieu>
   </adresse>
   <seance heure="16h" ref_film="1"/>
   <seance heure="20h" ref_film="1"/>
   <seance heure="20h" ref_film="3"/>
   <seance heure="20h" ref_film="4"/>
   <seance heure="20h" ref_film="5"/>
  </cinema>
  <cinema>
   <nom>Pathe</nom>
   <adresse>
    <ville>Rouen</ville>
    <lieu>Dock de Rouen</lieu>
   </adresse>
   <seance heure="18h" ref_film="1"/>
   <seance heure="20h" ref_film="1"/>
   <seance heure="21h" ref_film="2"/>
   <seance heure="22h" ref_film="4"/>
   <seance heure="20h" ref_film="5"/>
   <seance heure="23h" ref_film="5"/>
  </cinema>
 </cinemas>
 <films>
  <film film_id="1">
   <titre>Titeuf, le film</titre>
   <année>2011</année>
  </film>
  <film film_id="2">
   <titre>Scream 4</titre>
   <année>2011</année>
  </film>
  <film film_id="3">
   <titre>Source Code</titre>
   <année>2011</année>
  </film>
  <film film_id="4">
   <titre>Rango</titre>
   <année>2011</année>
  </film>
  <film film_id="5">
   <titre>Sucker Punch</titre>
   <année>2011</année>
  </film>
 </films>
</catalogue>
```

Base de données XML

Base de données Native XML

- A pour modèle de données du XML
- S'interroge avec du XQuery
- A comme unité de stockage logique un document XML.
- Peut avoir n'importe quel modèle physique de stockage (fichiers XML, BD relationnelle, fichiers plats et structure d'index, etc.)

Quelques BD XML natives

Name	License	Native Language	XQuery 3.0	XQuery Update	XQuery Full Text	EXPath Extensions	EXQuery Extensions	XSLT 2.0
BaseX	BSD License	Java	Yes	Yes	Yes	Yes	Yes	Yes
eXist	LGPL License	Java	Partial	Proprietary[11]	Proprietary	Yes	Yes	Yes
MarkLogic Server	Commercial	C++	Partial	Proprietary	Proprietary	No	No	Yes
Sedna	Apache License	C++	No	Yes	Yes	No	No	No

Name	XQJ	XML:DB	RESTful	RESTXQ	WebDAV
BaseX	Yes	Yes	Yes	Yes	Yes
eXist	Yes	Yes	Yes	Yes	Yes
MarkLogic Server	Yes	No	Yes	Yes	Yes
Sedna	Yes	Yes	No	No	Yes

D’autres outils pour XQuery

- Oracle XML DB
- xDB
- Saxon parser
- Galax
- Xindice (inactif)
- XQDT (eclipse)

eXist Native XML Database

- eXist
  - Java
  - Créé en 2004
- Support :
  - XQuery
  - XUpdate
  - XSLT
  - Recherche plein texte
  - Extension XQuery
- Structure de stockage
  - Documents (fichiers) dans des collections (Répertoire)
  - Structure d'Arbre B+ pour le stockage des fichiers avec indexs
  - Resources non XML (XQuery, CSS, JPEG ..), stockée en tant que binaire
- Multiple mode de déploiement
  - Librairie dans une application Java
  - A déployer dans un Apache/Tomcat
  - Version embarquée dans un Jetty
- Multiple Interfaces
  - **API JAVA WQJ**
  - **REST**
  - SOAP
  - XML-RPC

Conclusion Persistance et Recherche XML

- XQuery, mécanisme de requête adapté à XML
- XUpdate, extension pour l'ajout, suppression, modification de données XML
- Base de données XML native dédiées au stockage et à la manipulation de données XML.

Liens XML

XLink

XLink : XML Linking Language

- Recommendation W3C du 27 Juin 2001

- Etendre les liens hypertexte de HTML
- Inspiré de TEI
- Associer de la sémantique aux liens et aux ressources
- Liens externes
- Liens bidirectionnels
- Liens n-aires

Liens XML : Fondements

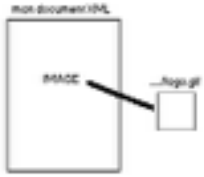
- N'importe quel élément XML peut être un lien.
- Les liens XLink précise les roles, titres et sens des liens;
- Ces liens peuvent relier plus de deux ressources, être bi-directionnels, multi-directionnels et externes aux documents liés.
- XLink peut préciser le comportement des applications par rapport aux liens, notamment ouvrir un lien dans une nouvelle fenêtre.
- Les liens peuvent être définis dans un document à part, séparant le contenu des documents de la "navigation".

- XPointer permet d'adresser/localiser un endroit à l'intérieur d'un document XML, sans recours aux "ancres", et adresse même des parties d'un document.

XLink : Exemples

```
<CREATEUR xlink:type="simple"
                xlink:href="http://www.sun.com/"
                xlink:actuate="onRequest"/>
      SUN microsysteme
</CREATEUR>
```

```
<IMAGE xlink:type="simple"
         xlink:href="linux/triomphant/logo.gif"
         xlink:show="embed"
         xlink:actuate="onLoad"
         xlink:role="logo" />
```



```
<!-- Premier chapitre du livre -->
<chapitre id="chap1">
       <titre>Les Bases d'XML</titre>
       <paragraphe>
              <titre>Premier paragraphe</titre>
              <corps>Contenu du premier paragraphe avec
                     un <lien xlink:type="simple" xlink:href="#chap2">lien</lien> vers ....</corps>
       </paragraphe>
</chapitre>
<!-- Premier chapitre du livre -->
<chapitre id="chap2">
       <titre>Les utilisations d'XML</titre>
       <paragraphe>
              <titre>Premier paragraphe</titre>
              <corps>Contenu du premier paragraphe...</corps>
       </paragraphe>
</chapitre>
```

Pointeurs XML

XPointer

XPointer Framework

- Objectif : Pointer une partie d'un fichier XML
- La forme générale d'un lien est uri#xpointer(pointage), où uri est un universal ressource identifier et pointage une combinaison de spécifications XPointer et d'expressions XPath.
- L'ancrage html s'implémente ainsi :
  - dans XML, tout élément peut avoir un attribut id à valeur unique et exclusive :
    ```
    <chapitre id="chap1">…</chapitre>
    ```
  - le lien pourra adresser xpointer(id("chap1"))
- XPointer permet de créer un lien vers n'importe quel endroit d'une page, sans avoir besoin d'id:
  - xpointer(corps[text() = "Chapitre 1"]) ici l'élément corps dont le texte est « Chapitre 1 »

XPointer Framework

- XPointer permet de pointer vers plusieurs éléments XML précis, un texte, ou encore vers une partie d'un document XML. :
  - xpointer(chapitre/@num="1" to chapitre/@num="2") sélectionne tous les renseignement dans l'arbre XML depuis l'attribut num="1" jusqu'à l'attribut num="2"
  - ce mécanisme définit une région/range entre 2 points dans l'arborescence du document

- XPointer dispose de fonctions de manipulations des localisations (point, noeud, region) et des ensembles de celles-ci :
  - xpointer(string-range(livre[chapitre/@num="1"]/corps, "Chapitre") [position()=last]) sélectionne la dernière position des localisations de la chaine "Chapitre" dans les chapitre ayant pour numéro de section le numéro 1

XLink et XPointer : utilisations

- Adoption très limitée
- Peu d’outils supportant ces « standards »
- Pas retenu pour XHTML…

Standards W3C

… pour XML

Parseurs XML

- Parseur évènementiel => Sax
- Parseur objet générique => Dom
- Parseur objet métier => Data Binding

En Java

- Depuis JDK 1.4, Java permet la gestion des documents XML.
- JAXP, API regroupant SAX et DOM (et XSL)
  - Implémentations nécessaires (voire plus loin)
- JAXB, API pour le Data Binding

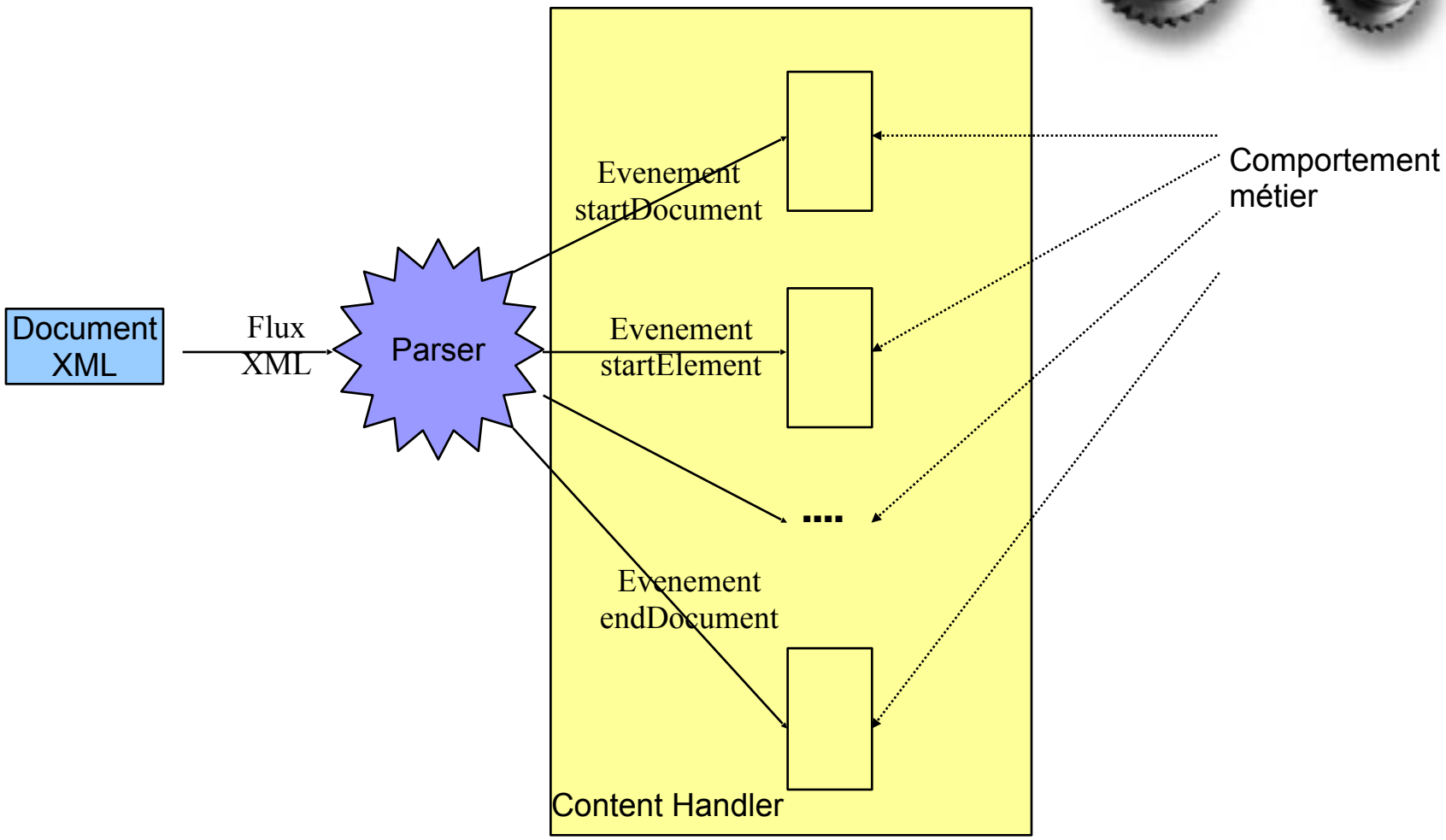
SAX

- SAX : **S**imple **A**PI for **X**ML
- est basé sur un modèle événementiel : l'analyseur appelle automatiquement une méthode lorsqu'un événement est détecté dans le fichier XML
- A chaque événement (début du doc, début d'un élément, texte…) un *callback* est appelé
- Le gestionnaire du contenu effectue les traitements appropriés au *callback* appelé

SAX : Implémentations

- Langages de programmation
  - Java, C++
  - C#, VB
  - Python, PHP
- Quelques implémentations
  - Saxon
  - Xerces de Apache
  - MSXML3 de Microsoft
  - JAXP de Sun

SAX : Fonctionnement



SAX Exemple

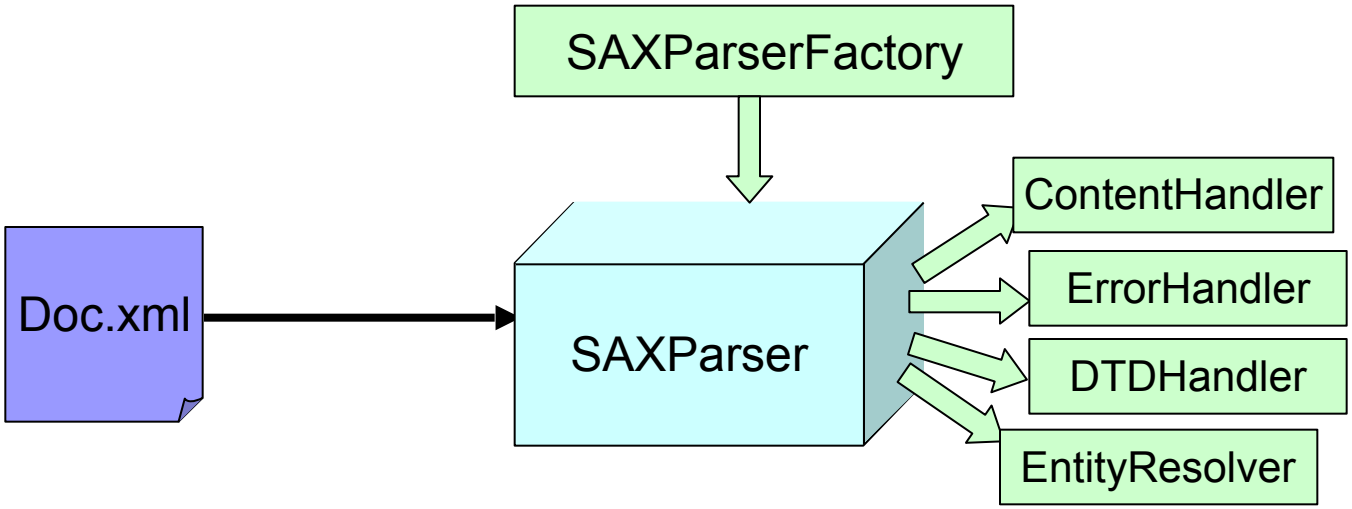
```
public class SaxExample extends DefaultHandler {

        public void startDocument() {
                System.out.println("Racine du doc");
        }

        public void endDocument() {
                System.out.println("Fin du doc");
        }

        public void startElement(String uri, String localName, String tag, Attributes attributes) throws SAXException {
                if (tag.equals("livre")) {
                        System.out.println("This is a 'Livre' node (start)");
                } else if (tag.equals("auteur")) {
                        System.out.println("This is an 'Auteur' node (start)");
                } else if (tag.equals("titre")) {
                        System.out.println("This is an 'Titre' node (start)");
                } else if (tag.equals("paragraphe")) {
                        System.out.println("This is an 'Paragraphe' node (start)");
                } else if (tag.equals("corps")) {
                        System.out.println("This is an 'Corps' node (start)");
                }
        }

        public void endElement(String namespaceUri, String localName, String qualifiedName) throws SAXException {
                if (qualifiedName.equals("livre")) {
                        System.out.println("This is a 'Livre' node (end)");
                } else if (qualifiedName.equals("auteur")) {
                        System.out.println("This is an 'Auteur' node (end)");
                } else if (qualifiedName.equals("titre")) {
                        System.out.println("This is an 'Titre' node (end)");
                } else if (qualifiedName.equals("paragraphe")) {
                        System.out.println("This is an 'Paragraphe' node (end)");
                } else if (qualifiedName.equals("corps")) {
                        System.out.println("This is an 'Corps' node (end)");
                }
        }

        public void characters(char[] chars, int startIndex, int endIndex) {
                if (!new String(chars, startIndex, endIndex).trim().equalsIgnoreCase("")) {
                        System.out.println("Text content : '"+ new String(chars, startIndex, endIndex)+"'");
                }
        }
}
```

SAX Exemple

```
public void test() throws ParserConfigurationException,
 SAXException, IOException {

SaxExample handler = new SaxExample();
SAXParser parser = null;

parser = SAXParserFactory.newInstance().newSAXParser();
File in = new File("src/livres/livre.xml");

parser.parse(in, handler);
}
```

```
Racine du doc
This is a 'Livre' node (start)
This is an 'Titre' node (start)
Text content : 'Apprendre le XML'
This is an 'Titre' node (end)
This is an 'Auteur' node (start)
Text content : 'Bondu'
Text content : 'Emilien'
This is an 'Auteur' node (end)
This is an 'Titre' node (start)
Text content : 'Les Bases d'XML'
This is an 'Titre' node (end)
This is an 'Paragraphe' node (start)
This is an 'Titre' node (start)
Text content : 'Premier paragraphe'
This is an 'Titre' node (end)
This is an 'Corps' node (start)
Text content : 'Contenu du premier paragraphe...'
This is an 'Corps' node (end)
This is an 'Paragraphe' node (end)
This is an 'Paragraphe' node (start)
This is an 'Titre' node (start)
Text content : 'Deuxième paragraphe'
This is an 'Titre' node (end)
This is an 'Corps' node (start)
Text content : 'Contenu du deuxième paragraphe...'
This is an 'Corps' node (end)
This is an 'Paragraphe' node (end)
This is an 'Titre' node (start)
Text content : 'La transformation XML'
This is an 'Titre' node (end)
This is an 'Paragraphe' node (start)
This is an 'Titre' node (start)
Text content : 'Premier paragraphe'
This is an 'Titre' node (end)
This is an 'Corps' node (start)
Text content : 'Contenu du premier paragraphe...'
This is an 'Corps' node (end)
This is an 'Paragraphe' node (end)
This is a 'Livre' node (end)
Fin du doc
```

SAX : Constituants



L’interface ContentHandler

- startDocument() Notification du début du document
- endDocument() Notification de la fin du document
- startElement(String namespaceURI, String localName, String qName, Attributes atts) Notification du début d'un élément
- endElement(String namespaceURI, String localName, String qName) Notification de la fin d'un élément
- characters(char[] ch, int start, int length) Notification de données caractères
- ignorableWhitespace(char[] ch, int start, int length) Notification d'espaces blancs ignorables dans le contenu d'un élément
- startPrefixMapping(String prefix) Notification du début de la portée d'un espace de nommage
- endPrefixMapping(String prefix) Notification de la fin de la portée d'un espace de nommage
- processingInstruction(String target, String data) Notification d'une instruction de traitement
- skippedEntity(String name) Notification d'une entité non résolue
- setDocumentLocator(Locator locator) Permet d'enregistrer un Locator qui délivre des informations sur la localisation d'un événement SAX (numéro de ligne, colonne…)

Quand utiliser SAX?

- Document XML volumineux
- Lecture rapide et efficace
- Économie de mémoire car SAX ne construit pas une représentation en mémoire de la structure en arborescence des fichiers XML lus

- Gestion en flux de données, pas de retour en arrière ou saut à une position dans le fichier.

SAX Exercice

À VOUS DE JOUER!

```xml
<?xml version="1.0" encoding="UTF-8"?>
<livre xmlns="http://livre.org"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="http://livre.org livre.xsd">

        <titre>Apprendre le XML</titre>

        <auteur>
                <nom>Bondu</nom>
                <prenom>Emilien</prenom>
        </auteur>

        <!-- Premier chapitre du livre -->
        <chapitre num="1">
                <titre>Les Bases d'XML</titre>
                <paragraphe>
                        <titre>Premier paragraphe</titre>
                        <corps>Contenu du premier paragraphe...</corps>
                </paragraphe>
                <paragraphe>
                        <titre>Deuxième paragraphe</titre>
                        <corps>Contenu du deuxième paragraphe...</corps>
                </paragraphe>
        </chapitre>


        <!-- Premier chapitre du livre -->
        <chapitre num="2">
                <titre>La transformation XML</titre>
                <paragraphe>
                        <titre>Premier paragraphe</titre>
                        <corps>Contenu du premier paragraphe...</corps>
                </paragraphe>
        </chapitre>
</livre>
```

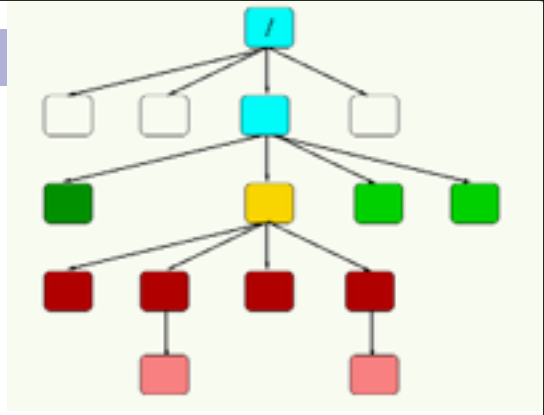
```java
public class SaxExample extends DefaultHandler {

        private static final Object TAG_1 = "livre";
        private static final Object TAG_2 = "paragraphe";
        private static final Object TAG_3 = "titre";

        private boolean inElement= false;

        public void startDocument() {
                System.out.println("Racine du doc");
        }

        public void endDocument() {
                System.out.println("Fin du doc");
        }

        public void startElement(String uri,
                                 String localName,
                                 String tag,
                                 Attributes attributes)
        throws SAXException {
                if (tag.equals(TAG_1)) {
                        inElement = false;
                } else if (tag.equals(TAG_2)) {
                        inElement = false;
                } else if (tag.equals(TAG_3)) {
                        inElement = true;
                }
        }

        public void endElement(String namespaceUri,
                               String localName,
                               String qualifiedName)
        throws SAXException {
                inElement = false;
        }

        public void characters(char[] chars, int startIndex, int endIndex) {
                if (inElement) {
                        System.out.println(
                        new String(chars, startIndex, endIndex)
                        );
                }
        }
}
```

SAX Exercice

Il produit la sortie ci-dessous…

```
Racine du doc
1)Apprendre le XML
2)Les Bases d'XML
3)Premier paragraphe
4)Deuxième paragraphe
5)La transformation XML
6)Premier paragraphe
Fin du doc
```

DOM

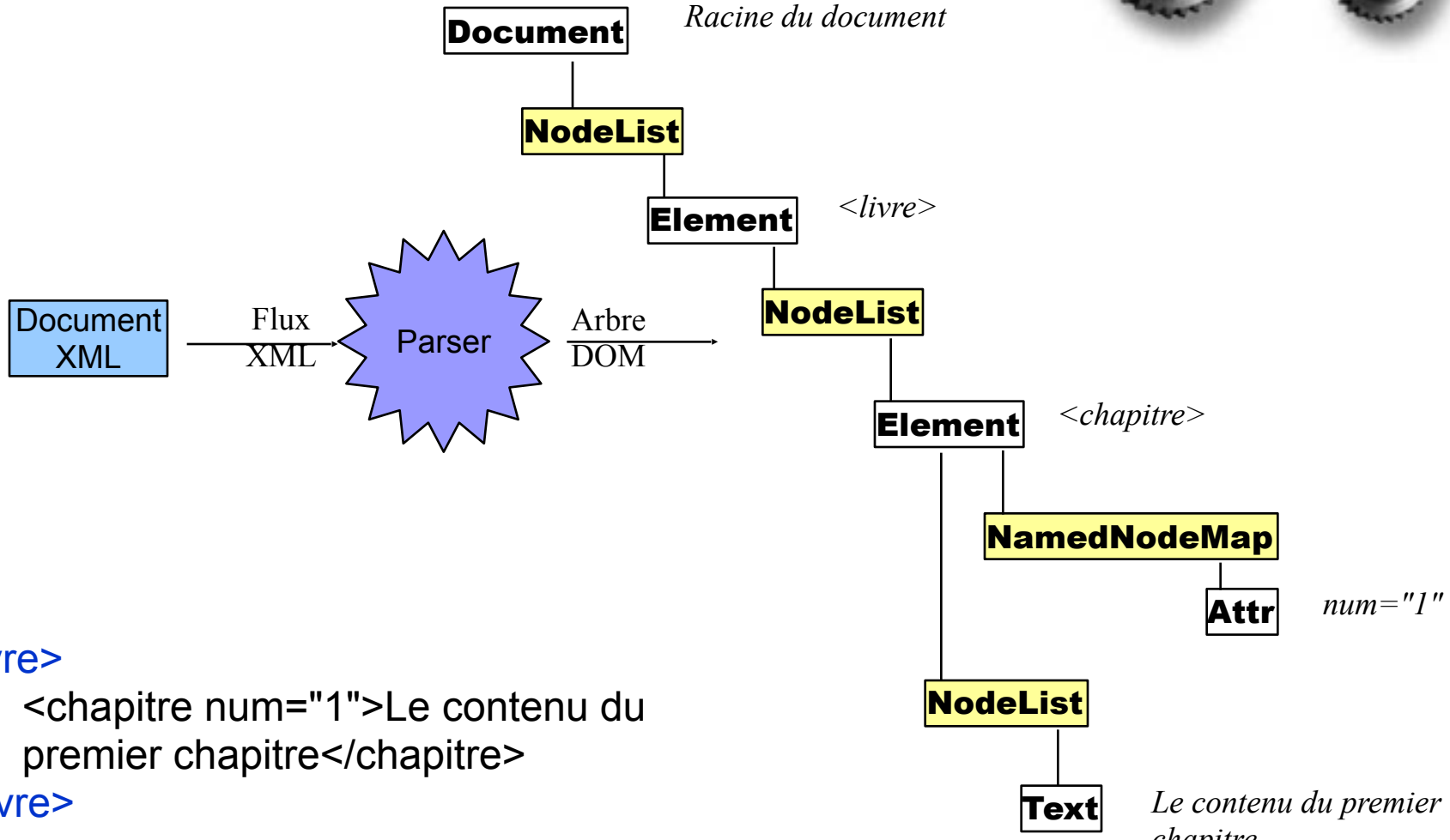


- DOM : **D**ocument **O**bject **M**odel
- offre toute une panoplie d'outils destinés à construire et manipuler un document XML. Pour cela, DOM met à disposition des interfaces, des méthodes et des propriétés permettant de gérer l'ensemble des composants présents dans un document XML.
- spécifie diverses méthodes et propriétés permettant notamment, de créer (*createNode*…), modifier (*replaceChild*…), supprimer (*remove*…) ou d'extraire des données (*get*…) de n'importe quel élément ou contenu d'un document XML.

DOM : Implémentations

- Langages de programmation
  - Java, C++
  - C#, VB
  - Python, PHP
- Quelques implémentations
  - Saxon
  - Xerces de Apache
  - MSXML3 de Microsoft
  - JAXP de Sun

DOM : Fonctionnement




```
<livre>
    <chapitre num="1">Le contenu du
    premier chapitre</chapitre>
</livre>
```

DOM Exemple

```
public class TestDOM {

	@Test
	public void test() {
		try {
			File fXmlFile = new File("src/livres/livre.xml");
			DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
			DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
			Document doc = dBuilder.parse(fXmlFile);

			System.out.println("Root element :" + doc.getDocumentElement().getNodeName());
			System.out.println("----------------------------");

			for (int i = 0; i < doc.getDocumentElement().getChildNodes().getLength(); i++) {
				Node nNode = doc.getDocumentElement().getChildNodes().item(i);

				if (nNode.getNodeType() == Node.ELEMENT_NODE) {
					System.out.println("Current Element :" + nNode.getNodeName());
					Element eElement = (Element) nNode;
					if (eElement.getNodeName().equalsIgnoreCase("titre")) {
						System.out.println("Book title : " + eElement.getTextContent());
					} else if (eElement.getNodeName().equalsIgnoreCase("auteur")) {
						System.out.println("Book author name : " +
								eElement.getElementsByTagName("prenom").item(0).getTextContent());
					}
				}
			}
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
}
```

DOM Exemple

```
Root element :livre
----------------------------
Current Element :titre
Book title : Apprendre le XML
Current Element :auteur
Book author name : Emilien
Current Element :chapitre
Current Element :chapitre
```

DOM Exemple

```
public void test2() throws TransformerException, ParserConfigurationException {
                DocumentBuilderFactory docFactory = DocumentBuilderFactory.newInstance();
                DocumentBuilder docBuilder = docFactory.newDocumentBuilder();

                // root element
                Document doc = docBuilder.newDocument();

                Element rootElement = doc.createElementNS("http://livre.org/", "livre");
                doc.appendChild(rootElement);

                // elements
                Element titre = doc.createElementNS("http://livre.org/", "titre");
                titre.setTextContent("Apprendre le XML");
                rootElement.appendChild(titre);

                Element chap = doc.createElementNS("http://livre.org/", "chapitre");
                Attr attr = doc.createAttribute("num");
                attr.setValue("1");
                chap.setAttributeNode(attr);
                rootElement.appendChild(chap);

                Element para = doc.createElementNS("http://livre.org/", "paragraphe");
                chap.appendChild(para);

                Element corps = doc.createElementNS("http://livre.org/", "corps");
                corps.setTextContent("Ceci est le contenu du premier paragraphe");
                para.appendChild(corps);


                // write the content into xml file
                TransformerFactory transformerFactory = TransformerFactory.newInstance();
                Transformer transformer = transformerFactory.newTransformer();
                DOMSource source = new DOMSource(doc);
                StreamResult result = new StreamResult(new File("src/livres/testDom.xml"));

                // Output to console for testing
                // StreamResult result = new StreamResult(System.out);

                transformer.transform(source, result);

                System.out.println("File saved!");
        }
```

DOM Exemple

```
<?xml version="1.0" encoding="UTF-8"?>
<livre xmlns="http://livre.org/">
      <titre>Apprendre le XML</titre>
      <chapitre num="1">
            <paragraphe>
                  <corps>Ceci est le contenu du premier paragraphe</corps>
            </paragraphe>
      </chapitre>
</livre>
```

DOM : Building blocks

- **Document** : Représente le document XML
- **Propriétés** : doctype, documentElement, implementation, attributes, childNodes, firstChild, lastChild, localName, namespaceURI, nextSibling, nodeName, nodeType, nodeValue, ownerDocument, parentNode, prefix, previousSibling
- **Méthodes** : createAttribute, createAttributeNS, createCDATASection, createComment, createDocumentFragment, createElement, createElementNS, createEntityReference, createProcessingInstruction, createTextNode, getElementById, getElementsByTagName, getElementsByTagNameNS, importNode, appendChild, cloneNode, hasAttributes, hasChildNodes, insertBefore, isSupported, normalize, removeChild, replaceChild

DOM : Building blocks

- **Node** : Représente un nœud du document
- **Propriétés** :attributes, childNodes, firstChild, lastChild, localName, namespaceURI, nextSibling, nodeName, nodeType, nodeValue, ownerDocument, parentNode, prefix, previousSibling
- **Méthodes** : appendChild, cloneNode, getNodeType, hasAttributes, hasChildNodes, insertBefore, isSupported, normalize, removeChild, replaceChild
- **NodeList :** Collection ordonnée de nœuds

DOM : Building blocks

- **Element** : Représente un élément dans un document XML
- **Propriétés** : tagName, attributes, childNodes, firstChild, lastChild, localName, namespaceURI, nextSibling, nodeName, nodeType, nodeValue, ownerDocument, parentNode, prefix, previousSibling
- **Méthodes** : getAttribute, getAttributeNS, getAttributeNode, getAttributeNodeNS, getElementsByTagName, getElementsByTagNameNS, hasAttribute, hasAttributeNS, removeAttribute, removeAttributeNS, removeAttributeNode, setAttribute, setAttributeNS, setAttributeNode, setAttributeNodeNS
- Spécialiser si nécessaire

```
if ( node.getNodeType() == Node.ELEMENT_NODE ) {
        (Element) el = (Element) node;
        Attr fooAttr = el.getAttributeNode("num"); }
```

DOM : Building blocks

- **Attr** : représente un attribut dans l'arbre XML
- **Propriétés** : name, ownerElement, specified, value, ownerDocument, parentNode, prefix, previousSibling…
- **Méthodes** : appendChild, cloneNode, hasAttributes, hasChildNodes, insertBefore, isSupported, normalize, removeChild, replaceChild
- Contrairement à l'arbre XML, les attributs sont des nœuds de l'arbre de DOM mais
  - ils ne sont pas connectés à l'arbre par une dépendance hiérarchique.
  - element.childNode ne retourne pas d'attributs, il faut utiliser element.attributes

DOM vs SAX

- Utilise le parseur Sax
- Accès non linéaire
- Programmation simple
- Richesse en fonctionnalités
- Problèmes de performance : lecture de tout le fichier et chargement en mémoire

Références

- Cours de référence
  - Jenny Benois-Pineau, Benjamin N-Guyen, Georges Gardarin
- Livres
  - XML in a Nutshell, Third Edition Ed O'Reilly
- En ligne
  - http://www.w3.org/
  - http://www.w3schools.com/xml/default.asp