Introduction à XML

Olivier Caron

Polytech Lille
Avenue Paul Langevin Cité Scientifique Lille 1
59655 Villeneuve d'Ascq cedex

http://ocaron.plil.fr
Olivier.Caron@polytech-lille.fr

- Merci à Anne Etien pour l'aide apportée à l'élaboration de ce cours.

- Merci à Anne Etien pour l'aide apportée à l'élaboration de ce cours.
- Merci à `http://w3schools.com/xquery`

XML

- EXtensible Markup Language

XML

- EXtensible Markup Language
- Issu du langage SGML (comme HTML)

XML

- EXtensible Markup Language
- Issu du langage SGML (comme HTML)
- Permet de décrire des structures logiques de documents principalement textuels

XML

- EXtensible Markup Language
- Issu du langage SGML (comme HTML)
- Permet de décrire des structures logiques de documents principalement textuels
- Langage de balises structurées

Distinctions avec HTML

- L'ensemble des balises n'est pas figé.

Distinctions avec HTML

- L'ensemble des balises n'est pas figé.
- Les balises ne servent qu'à décrire les données, pas de balises de mise en page

Distinctions avec HTML

- L'ensemble des balises n'est pas figé.
- Les balises ne servent qu'à décrire les données, pas de balises de mise en page
- Nom de balises : distinction majuscules et minuscules

Distinctions avec HTML

- L'ensemble des balises n'est pas figé.
- Les balises ne servent qu'à décrire les données, pas de balises de mise en page
- Nom de balises : distinction majuscules et minuscules
- Langage rigoureux

XML : un exemple

```
<lettre>
  <en-tete>
    <nom-ent>E.P.U.L.</nom-ent>
    <adresse>
      <rue>Avenue Paul Langevin</rue><numero>12</numero>
      <ville>Villeneuve d'Ascq</ville><cp>59655</cp>
    </adresse>
  </en-tete>
  <date>2002-04-08</date>
  <objet>Inscription ....</objet> <salut> Monsieur</salut>
  <corps> .... </corps>
</lettre>
```

Un document XML

- Un document XML est un document texte :

Un document XML

- Un document XML est un document texte :
  - éditable par n'importe quel éditeur de texte

Un document XML

- Un document XML est un document texte :
  - éditable par n'importe quel éditeur de texte
  - répond à une structure sous formes de balises.

Un document XML

- Un document XML est un document texte :
  - éditable par n'importe quel éditeur de texte
  - répond à une structure sous formes de balises.
  - peut être généré par des outils

Un document XML

- Un document XML est un document texte :
  - éditable par n'importe quel éditeur de texte
  - répond à une structure sous formes de balises.
  - peut être généré par des outils
  - peut être transmis par le réseau

Un document XML

- Un document XML est un document texte :
  - éditable par n'importe quel éditeur de texte
  - répond à une structure sous formes de balises.
  - peut être généré par des outils
  - peut être transmis par le réseau
  - est compatible avec les navigateurs webs

Terminologie XML

- Une paire de balises ouvrante et fermante et le fragment qu'elles entourent constituent un **élément XML**

Terminologie XML

- Une paire de balises ouvrante et fermante et le fragment qu'elles entourent constituent un **élément XML**
- Le nom de la balise est le type de l élément

Terminologie XML

- Une paire de balises ouvrante et fermante et le fragment qu'elles entourent constituent un **élément XML**
- Le nom de la balise est le type de l élément
- Le contenu d un élément XML est obtenu en enlevant les balises qui l'entourent
  Elément XML :<**date**>**2002-04-08** </**date**>

Quelques remarques sur l'exemple

- Document lisible

Quelques remarques sur l'exemple

- Document lisible
- L'indentation n'a aucune incidence

Quelques remarques sur l'exemple

- Document lisible
- L'indentation n'a aucune incidence
- Pas d'indication de mise en page (texte justifié, en-tête à droite, . . .)

Quelques remarques sur l'exemple

- Document lisible
- L'indentation n'a aucune incidence
- Pas d'indication de mise en page (texte justifié, en-tête à droite, . . .)
- la date est indiquée sous le format '2002-04-08' mais pourra être affichée sous la forme 'lundi 08 avril 2002'.

Documents bien formés et documents valides

- Un document est *bien formé* lorsqu'il obéit aux règles syntaxiques du langage XML.
  Il sera traité avec succès par un processeur adéquat et notamment par un parseur XML.

Documents bien formés et documents valides

- Un document est *bien formé* lorsqu'il obéit aux règles syntaxiques du langage XML.
  Il sera traité avec succès par un processeur adéquat et notamment par un parseur XML.
- Un document est valide lorsqu'il obéit à une structure type définie explicitement dans une DTD (Document Type Definition)

Structure d'un document XML

- Un document XML se décompose :

Structure d'un document XML

- Un document XML se décompose :
  - D'un prologue facultatif mais conseillé.

Structure d'un document XML

- Un document XML se décompose :
  - D'un prologue facultatif mais conseillé.
  - D'un arbre d'éléments qui forme le contenu propre du document

Structure d'un document XML

- Un document XML se décompose :
  - D'un prologue facultatif mais conseillé.
  - D'un arbre d'éléments qui forme le contenu propre du document
  - De commentaires

```
<!- - un commentaire fort utile - ->
```

Structure d'un document XML

- Un document XML se décompose :
  - D'un prologue facultatif mais conseillé.
  - D'un arbre d'éléments qui forme le contenu propre du document
  - De commentaires

    ```
    <!- - un commentaire fort utile - ->
    ```
  - d'instructions de traitement(facultatifs)

    ```
    <?  application instruction ?>
    ```

Le prologue

- La déclaration du document a la forme :
  `<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>`
  `standalone` permet de savoir si le document contient toutes les informations pour traiter le document.

Le prologue

- La déclaration du document a la forme :
  `<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>`
  `standalone` permet de savoir si le document contient toutes les informations pour traiter le document.
- la déclaration du type de document :
  `<!DOCTYPE rapport SYSTEM="rapport.dtd" [ déclarations ]>`
  Cette déclaration indique le document est de type rapport dont la définition se trouve dans le document `rapport.dtd`.

Le prologue

- La déclaration du document a la forme :
  `<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>`
  `standalone` permet de savoir si le document contient toutes les informations pour traiter le document.
- la déclaration du type de document :
  `<!DOCTYPE rapport SYSTEM="rapport.dtd" [ déclarations ]>`
  Cette déclaration indique le document est de type rapport dont la définition se trouve dans le document `rapport.dtd`.
- L'attribut `SYSTEM` est spécifié par une adresse URL (partage de DTD).

Structures types de document

- DTD : mécanisme par lequel les structures sont spécifiées.

Structures types de document

- DTD : mécanisme par lequel les structures sont spécifiées.
- La DTD peut directement se trouver dans le document

Structures types de document

- DTD : mécanisme par lequel les structures sont spécifiées.
- La DTD peut directement se trouver dans le document
- Nécessaire pour vérifier la *validité* du document

Structures types de document

- DTD : mécanisme par lequel les structures sont spécifiées.
- La DTD peut directement se trouver dans le document
- Nécessaire pour vérifier la *validité* du document
- Une DTD est appliquable à plusieurs documents XML.

Structures types de document

- DTD : mécanisme par lequel les structures sont spécifiées.
- La DTD peut directement se trouver dans le document
- Nécessaire pour vérifier la *validité* du document
- Une DTD est appliquable à plusieurs documents XML.
- Définition de nouveaux languages !

DTD : déclaration d'éléments

- Une déclaration d'élément est de la forme :

  `<!ELEMENT nom modèle>`

DTD : déclaration d'éléments

- Une déclaration d'élément est de la forme :

  `<!ELEMENT nom modèle>`
- Le modèle peut autoriser :

DTD : déclaration d'éléments

- Une déclaration d'élément est de la forme :

  `<!ELEMENT nom modèle>`
- Le modèle peut autoriser :
  - Un ou des éléments fils spécifiés

DTD : déclaration d'éléments

- Une déclaration d'élément est de la forme :

  `<!ELEMENT nom modèle>`
- Le modèle peut autoriser :
  - Un ou des éléments fils spécifiés
  - Des données représentées par un flot de caractères

DTD : déclaration d'éléments

- Une déclaration d'élément est de la forme :

  `<!ELEMENT nom modèle>`
- Le modèle peut autoriser :
  - Un ou des éléments fils spécifiés
  - Des données représentées par un flot de caractères
  - Un mélange des deux précédents

DTD : déclaration d'éléments

- Une déclaration d'élément est de la forme :

  `<!ELEMENT nom modèle>`
- Le modèle peut autoriser :
  - Un ou des éléments fils spécifiés
  - Des données représentées par un flot de caractères
  - Un mélange des deux précédents
  - Un élément vide

DTD : éléments fils

- Les éléments fils peuvent avoir un ordre imposé
  liste séparés par des virgules
  Ex : `<!ELEMENT chapitre (titre, intro, section)>`

DTD : éléments fils

- Les éléments fils peuvent avoir un ordre imposé
  liste séparés par des virgules
  Ex : `<!ELEMENT chapitre (titre, intro, section)>`
- Les élements fils peuvent être en ordre libre
  éléments séparés par '|'
  Ex : `<!ELEMENT paragraphes (p | br)*>`

DTD : éléments fils

- Les éléments fils peuvent avoir un ordre imposé
  liste séparés par des virgules
  Ex : `<!ELEMENT chapitre (titre, intro, section)>`
- Les élements fils peuvent être en ordre libre
  éléments séparés par '|'
  Ex : `<!ELEMENT paragraphes (p | br)*>`
- Les éléments peuvent être suffixés par un indicateur d'occurrence : `x?` : l'élément x peut apparaître 0 ou 1 fois. `x*` : l'élément x peut apparaître 0 ou plusieurs fois `x+` : l'élément x peut apparaître 1 ou plusieurs fois

DTD : un exemple

Avec la DTD suivante :

```
<!ELEMENT chapitre (titre, intro,(titre-sec,corps-sec)+)>
```

il est possible d'avoir le document valide :

```
<chapitre>
  <titre> ... </titre>
  <intro> ... </intro>
  <titre-sec> ... </titre-sec><corps-sec> ... </corps-sec>
  <titre-sec> ... </titre-sec><corps-sec> ... </corps-sec>
</chapitre>
```

DTD : élément donnée

- La présence de données dans le contenu d'un élément est indiquée par le mot-clé `#PCDATA`
  abréviation de *Parsed Character Data*

DTD : élément donnée

- La présence de données dans le contenu d'un élément est indiquée par le mot-clé `#PCDATA`
  abréviation de *Parsed Character Data*
- Ex : `<!ELEMENT p (#PCDATA)>`

DTD : élément donnée

- La présence de données dans le contenu d'un élément est indiquée par le mot-clé `#PCDATA`
  abréviation de *Parsed Character Data*
- Ex : `<!ELEMENT p (#PCDATA)>`
- Il ne peut y avoir de balises au sein d'un élément XML dont le modèle est `#PCDATA`

DTD : élément donnée

- La présence de données dans le contenu d'un élément est indiquée par le mot-clé `#PCDATA`
  abréviation de *Parsed Character Data*
- Ex : `<!ELEMENT p (#PCDATA)>`
- Il ne peut y avoir de balises au sein d'un élément XML dont le modèle est `#PCDATA`
- On peut mixer élément de données et élément fils :
  Ex : `<!ELEMENT chaine ( guillemet, #PCDATA, guillemet)>`

DTD : élément vide et contenu libre

- Exemple élément vide : `<!ELEMENT essai EMPTY>`
  Notation XML élément vide : `<essai />`
  Pour des raisons de compatibilité SGML

DTD : élément vide et contenu libre

- Exemple élément vide : `<!ELEMENT essai EMPTY>`
  Notation XML élément vide : `<essai />`
  Pour des raisons de compatibilité SGML
- Un élément dont le modèle est `#PCDATA` peut être vide

DTD : élément vide et contenu libre

- Exemple élément vide : `<!ELEMENT essai EMPTY>`
  Notation XML élément vide : `<essai />`
  Pour des raisons de compatibilité SGML
- Un élément dont le modèle est `#PCDATA` peut être vide
- Exemple élément libre : `<!ELEMENT essai ANY>`
  Utile pour la création et la mise au point de documents XML complexes

DTD : déclaration d'attributs

- Syntaxe :

  ```
  <!ATTLIST nom-élém.  nom-attr.  type-attr.
  décl-défaut>
  ```

DTD : déclaration d'attributs

- Syntaxe :
  `<!ATTLIST nom-élém. nom-attr. type-attr. décl-défaut>`
- Quelques types d'attributs :
  `CDATA` : signifie que la valeur de l'attribut est une chaîne de caractères.

DTD : déclaration d'attributs

- Syntaxe :
  `<!ATTLIST nom-élém. nom-attr. type-attr. décl-défaut>`
- Quelques types d'attributs :
  `CDATA` : signifie que la valeur de l'attribut est une chaîne de caractères.
- `ID` ou `IDREF` : permet de définir des renvois à l'intérieur de document.

DTD : déclaration d'attributs

- Syntaxe :
  `<!ATTLIST nom-élém. nom-attr. type-attr. décl-défaut>`
- Quelques types d'attributs :
  `CDATA` : signifie que la valeur de l'attribut est une chaîne de caractères.
- `ID` ou `IDREF` : permet de définir des renvois à l'intérieur de document.
- Quelques déclarations de défaut :
  simplement une valeur par défaut `#REQUIRED` : attribut obligatoire `#IMPLIED` : attribut facultatif `#FIXED 'valeur'` : l'attribut prend toujours la même valeur fixée

Insuffisance des DTD

- Pas de type de données à part du texte (#PCDATA)

Insuffisance des DTD

- Pas de type de données à part du texte (#PCDATA)
- Expression de cardinalités limitée (’?’, ’*’, ’+’)

Insuffisance des DTD

- Pas de type de données à part du texte (#PCDATA)
- Expression de cardinalités limitée (’?’, ’*’, ’+’)
- Syntaxe spécifique (pas XML)

Insuffisance des DTD

- Pas de type de données à part du texte (#PCDATA)
- Expression de cardinalités limitée (’?’, ’*’, ’+’)
- Syntaxe spécifique (pas XML)
- Une alternative : XML-schema du W3C

XML Schema

- définit des documents (avec typage des données)

XML Schema

- définit des documents (avec typage des données)
- spécifié en XML (pas une nouvelle syntaxe)

XML Schema

- définit des documents (avec typage des données)
- spécifié en XML (pas une nouvelle syntaxe)
- Permet de définir des espaces de noms et utilise lui-même un espace de nom `xs:` (ou `xsd:`)

XML Schema

- définit des documents (avec typage des données)
- spécifié en XML (pas une nouvelle syntaxe)
- Permet de définir des espaces de noms et utilise lui-même un espace de nom `xs:` (ou `xsd:`)
- analysable par un parseur XML

Les types XML

- La base d'un schéma XML : l'élément

```
<xs:element name="..." type="..." >
```

Les types XML

- La base d'un schéma XML : l'élément
  `<xs:element name="..." type="..." >`
- Un élément peut avoir un type :

Les types XML

- La base d'un schéma XML : l'élément
  `<xs:element name="..." type="..." >`
- Un élément peut avoir un type :
  - *Simple* si sa valeur a un type prédéfini en XML-SCHEMA (`xs:string`, `xs:int`, `xs:decimal`, `xs:double`, `...`) ou une extension de ces types.

Les types XML

- La base d'un schéma XML : l'élément
  `<xs:element name="..." type="..." >`
- Un élément peut avoir un type :
  - *Simple* si sa valeur a un type prédéfini en XML-SCHEMA (`xs:string`, `xs:int`, `xs:decimal`, `xs:double`, `...`) ou une extension de ces types.
  - *Complexe* s'il contient des sous-éléments ou s'il comporte un attribut :

Les types XML

- La base d'un schéma XML : l'élément
  `<xs:element name="..." type="..." >`
- Un élément peut avoir un type :
  - *Simple* si sa valeur a un type prédéfini en XML-SCHEMA (`xs:string`, `xs:int`, `xs:decimal`, `xs:double`, `...`) ou une extension de ces types.
  - *Complexe* s'il contient des sous-éléments ou s'il comporte un attribut :
    - `xs:all` tous les éléments doivent exister (ordre quelconque)

Les types XML

- La base d'un schéma XML : l'élément
  `<xs:element name="..." type="..." >`
- Un élément peut avoir un type :
  - *Simple* si sa valeur a un type prédéfini en XML-SCHEMA (`xs:string`, `xs:int`, `xs:decimal`, `xs:double`, `...`) ou une extension de ces types.
  - *Complexe* s'il contient des sous-éléments ou s'il comporte un attribut :
    - `xs:all` tous les éléments doivent exister (ordre quelconque)
    - `xs:choice` un des éléments doit exister

Les types XML

- La base d'un schéma XML : l'élément
  `<xs:element name="..." type="..." >`
- Un élément peut avoir un type :
  - *Simple* si sa valeur a un type prédéfini en XML-SCHEMA (`xs:string`, `xs:int`, `xs:decimal`, `xs:double`, `...`) ou une extension de ces types.
  - *Complexe* s'il contient des sous-éléments ou s'il comporte un attribut :
    - `xs:all` tous les éléments doivent exister (ordre quelconque)
    - `xs:choice` un des éléments doit exister
    - `xs:sequence` tous les éléments doivent exister dans l'ordre spécifié

Un exemple XML-Schema

```
<xs:complexType name="AddresseFR">
  <xs:sequence>
    <xs:element name="nom" type="xs:string"/>
    <xs:element name="rue" type="xs:string"/>
    <xs:element name="ville" type="xs:string"/>
    <xs:element name="codep" type="xs:decimal"/>
  </xs:sequence>
  <xs:attribute name="pays" type="xs:NMTOKEN" fixed="FR"/>
</xs:complexType>
```

Les occurences

- Une bibliothèque contient au moins un livre

```
<xs:element name="biblio">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="livre"
                  minOccurs="1" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Les attributs (1/2)

- Les éléments à contenu simple avec attributs

```
<xs:complexType name="titleWithId">
 <xs:simpleContent>
   <xs:extension base="xs:string">
     <xs:attribute name="id" type="xs:ID"/>
   </xs:extension>
 </xs:simpleContent>
</xs:complexType>
```

- Utilisation :

```
<titre id="RF525">La bible de XML</titre>
```

Les attributs (2/2)

- Les éléments à contenu complexe avec attributs

```
<xs:element name="traduction" minOccurs="0"
                              maxOccurs="unbounded">
 <xs:complexType>
   <xs:sequence>
    <xs:element name="traducteur" type="xs:string"
                minOccurs="1" maxOccurs="unbounded" />
   </xs:sequence>
   <xs:attribute name="langue" use="required" type="xs:string" />
   <xs:attribute name="dateTraduction" use="optional" type="xs:date"
 </xs:complexType>
```

- Utilisation :

```
<traduction langue="allemand" dateTraduction="2003-12-01">
  <traducteur>Michael</traducteur>
</traduction>
```

Grouper des éléments

```
<xs:group name="TitreAuteurISBN">
  <xs:sequence>
    <xs:element name="Titre" type="xs:string"/>
    <xs:element name="Auteur" type="xs:string"/>
    <xs:element name="ISBN" type="num5"/>
  </xs:sequence>
</xs:group>
<xs:element name="livre">
  <xs:complexType>
    <xs:sequence>
      <xs:group ref="TitreAuteurISBN"/>
      <xs:element ref="traduction"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Référence à un schéma XML

- Référence sans espace de noms :

```
<Livre xmlns:xsi="http://w3org/2001/XMLSchemainstance"
       xsi:noNamespaceSchemaLocation="Livre.xsd">
<Titre>Les reseaux</Titre> ...
```

- Référence avec espace de noms :

```
<bib:Livre xmlns:xsi="http://w3org/2001/XMLSchemainstance"
    xmlns:bib="http://www.cpe.fr/ns/bib"
    xsi:SchemaLocation="␣http://www.cpe.fr/ns/bibLivre.xsd">
<bib:Titre>Les reseaux</bib:Titre>
```

Bilan DTD et Schéma

- Les deux technologies cohabitent

Bilan DTD et Schéma

- Les deux technologies cohabitent
- DTD plus simple et rapide

Bilan DTD et Schéma

- Les deux technologies cohabitent
- DTD plus simple et rapide
- Schémas plus complet mais verbeux

L'industrie florissante XML

- MathML, VoiceML, ChemicalML, XMI...

L'industrie florissante XML

- MathML, VoiceML, ChemicalML, XMI…
- De nombreux outils d'édition graphiques ou textes

L'industrie florissante XML

- MathML, VoiceML, ChemicalML, XMI...
- De nombreux outils d'édition graphiques ou textes
- De nombreux outils de visualisation de documents XML (IE, ...)

L'industrie florissante XML

- MathML, VoiceML, ChemicalML, XMI...
- De nombreux outils d'édition graphiques ou textes
- De nombreux outils de visualisation de documents XML (IE, ...)
- De nombreux outils de parsing
  API normalisées SAX et DOM
  API existent sur plusieurs langages (Java, C, ...)

L'industrie florissante XML

- MathML, VoiceML, ChemicalML, XMI...
- De nombreux outils d'édition graphiques ou textes
- De nombreux outils de visualisation de documents XML (IE, ...)
- De nombreux outils de parsing
  API normalisées SAX et DOM
  API existent sur plusieurs langages (Java, C, ...)
- Les moteurs de transformation XSLT (XML $\rightarrow$ HTML)

Une démo !

Bases de données relationnelles et XML

- Outil d'échange des données entre SGBD

Bases de données relationnelles et XML

- Outil d'échange des données entre SGBD
- Plusieurs constructeurs proposent déjà de tels outils d'extraction des données (Oracle, . . .)

Bases de données relationnelles et XML

- Outil d'échange des données entre SGBD
- Plusieurs constructeurs proposent déjà de tels outils d'extraction des données (Oracle, . . .)
- Langage intermédiaire d'affichage des données vers différents supports : ordinateurs de bureau , portables GSM, PDA, . . .

Traduction d'une base relationnelle en XML

- Une table devient un élément XML

Traduction d'une base relationnelle en XML

- Une table devient un élément XML
- Une colonne devient un sous-élément XML

Traduction d'une base relationnelle en XML

- Une table devient un élément XML
- Une colonne devient un sous-élément XML
- Les associations XML sont traduites à l'aide des attribus standards id et idRef

Références d'entités (1/2)

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT addressBook (person)+>
<!ELEMENT person (email*)>
<!ATTLIST person id ID #REQUIRED>
<!ATTLIST person gender (male|female) #IMPLIED>
<!ELEMENT email (#PCDATA)>
<!ELEMENT link EMPTY>
<!ATTLIST link manager IDREF #IMPLIED subordinates
          IDREFS #IMPLIED>
```

Références d'entités (2/2)

```
<!DOCTYPE addressBook SYSTEM "ab.dtd">
<addressBook>
  <person id="B.WALLACE" gender="male">
    <email>bwallace@megacorp.com</email>
    <link manager="C.TUTTLE" />
  </person>
  <person id="C.TUTTLE" gender="female">
    <email>ctuttle@megacorp.com</email>
    <link subordinates="B.WALLACE" />
  </person>
</addressBook>
```

XQuery

- ”XQuery est à XML ce que SQL est aux bases de données relationnelles”

XQuery

- ”XQuery est à XML ce que SQL est aux bases de données relationnelles”
- Un langage de requêtes pour XML

XQuery

- ”XQuery est à XML ce que SQL est aux bases de données relationnelles”
- Un langage de requêtes pour XML
- Langage d’interrogation (et pas langage de manipulation)

XQuery

- ”XQuery est à XML ce que SQL est aux bases de données relationnelles”
- Un langage de requêtes pour XML
- Langage d’interrogation (et pas langage de manipulation)
- Basé sur le système de types de XML Schéma

XQuery

- ”XQuery est à XML ce que SQL est aux bases de données relationnelles”
- Un langage de requêtes pour XML
- Langage d’interrogation (et pas langage de manipulation)
- Basé sur le système de types de XML Schéma
- Compatible XPath, XSLT du W3C

Requêtes XQuery

- Requête formée d'expressions simples et composées combinées par des opérateurs et mots-clés

Requêtes XQuery

- Requête formée d'expressions simples et composées combinées par des opérateurs et mots-clés
- Sensible à la casse

Requêtes XQuery

- Requête formée d'expressions simples et composées combinées par des opérateurs et mots-clés
- Sensible à la casse
- Résultat d'une expression : séquence hétérogène

Requêtes XQuery

- Requête formée d'expressions simples et composées combinées par des opérateurs et mots-clés
- Sensible à la casse
- Résultat d'une expression : séquence hétérogène
- Commentaires entre (: et :)

la base XML : books (1/2

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
<book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year> <price>30.00</price>
</book>
<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year> <price>29.99</price>
</book>
```

la base XML : books (2/2

```
<book category="WEB">
  <title lang="en">XQuery Kick Start</title>
  <author>James McGovern</author>
  <author>Per Bothner</author>
  <author>Kurt Cagle</author>
  <author>James Linn</author>
  <author>Vaidyanathan Nagarajan</author>
  <year>2003</year> <price>49.99</price>
</book>
<book category="WEB">
  <title lang="en">Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year> <price>39.95</price>
</book>
</bookstore>
```

Expressions de base: XQuery (1/2)

- La fonction `doc`, permet d'extraire un document XML
  `doc("books.xml")`
- Path Expressions, permet de naviguer entre éléments XML

```
doc("books.xml")/bookstore/book/title
Fournit :
<title lang="en">Everyday Italian</title>
<title lang="en">Harry Potter</title>
<title lang="en">XQuery Kick Start</title>
<title lang="en">Learning XML</title>
```

- "/nomElement" : recherche des éléments `nomElement` dans le niveau inférieur
- "//nomElement" : recherche des des éléments `nomElement` dans le sous-arbre
  `doc("books.xml")//title` donne le même résultat

Expressions de base: XQuery (2/2)

- Les prédicats: `nomElement[clause]`

```
doc("books.xml")/bookstore/book[price<30]
Fournit :
<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year> <price>29.99</price>
</book>
```

- Expressions arithmétiques :$2 + 2$, booléennes
- Chaînes de caractères : "*HelloWorld*!"
- bibliothèque de fonctions
- Accès à un attribut : nomElement/@nomAttribut

XQuery FLWOR Expressions

- ”F”or, ”L”et, ”W”here, ”O”rder by, ”R”eturn

```
for $book in doc("books.xml")/bookstore/book
  let $bookPrice := $book/price
where $bookPrice>30
order by $book/title
return $book/title
```

```
<title lang="en">Learning XML</title>
<title lang="en">XQuery Kick Start</title>
```

- Les clauses `let`, `where` et `order by` sont facultatives

De XML à HTML (1/2)

```
<ul>
{
for $x in doc("books.xml")/bookstore/book/title
order by $x
return <li>{$x}</li>
}
</ul>
```

- Fournit le résultat :

```
<ul>
<li><title lang="en">Everyday Italian</title></li>
<li><title lang="en">Harry Potter</title></li>
<li><title lang="en">Learning XML</title></li>
<li><title lang="en">XQuery Kick Start</title></li>
</ul>
```

De XML à HTML (2/2)

```
<ul>
{
for $x in doc("books.xml")/bookstore/book/title
order by $x
return <li>{data($x)}</li>
}
</ul>
```

- Fournit le résultat :

```
<ul>
<li>Everyday Italian</li>
<li>Harry Potter</li>
<li>Learning XML</li>
<li>XQuery Kick Start</li>
</ul>
```

Conclusion

- Demain ? :

Conclusion

- Demain ? :
  - Des bases de données au format XML
    Plusieurs prototypes existent

Conclusion

- Demain ? :
  - Des bases de données au format XML
    Plusieurs prototypes existent
  - Des langages de requêtes pour XML : OQL, XQuery, XPath, ...

Conclusion

- Demain ? :
  - Des bases de données au format XML
    Plusieurs prototypes existent
  - Des langages de requêtes pour XML : OQL, XQuery, XPath, ...
  - Le langage universel ? (exemple OASIS)