XQuery

Plan

- Introduction
- Syntaxe XQuery
- L'expression FLOWR
- Expressions conditionnelles
- Fonctions XQuery
- Les fonctions utilisateurs
- XQUF : XQuery Update Facility

Introduction

- Xquery
  - Un langage de requête XML fortement typé, à base d'expression de chemins, de boucles, de tests et d'éléments de construction de documents XML.
  - Compatible avec plusieurs normes du W3C tels que XML, espaces de noms, XSLT, XPath et XML Schema.
  - XQuery XSLT et XPath partagent le même modèle de données et supportent les mêmes fonctions et opérateurs.
  - Une Recommandation du W3C
  - Version 1.0 : Janvier 23, 2007.
  - Version 2.0 : 17 Mars 2011.
  - Version 3.0 : 17 Mars 2014.

XQuery

- Une requête XQuery est bâtie sur une expression XPath.
- Il offre deux services :
  - interroger des documents XML
  - construire les résultats en format semi-structuré.
- La génération des résultats s'effectue sous forme de fragments XML.
- Certaines règles de base:
  - Une variable XQuery est défini avec $
  - Les commentaires XQuery sont délimitées par
    - (: XQuery Commentaire:)

Syntaxes XQuery

- La fonction doc() est utilisée pour accéder au fichier de données → doc ("file.xml")
- Usage de Xpath pour naviguer à travers des éléments dans un document XML.
  - Exemple : doc("books.xml")/bookstore/book/title
  - Permet de sélectionner tous les éléments **title** dans le fichier "books.xml":
- Usage des prédicats pour raffiner l'extraction de données.
  - doc("books.xml")/bookstore/book[price<30]

Syntaxe Xquery : FLOWR

- Une requête est basée sur l'instruction FLOWR = "For-Let-Where-Order-Return".
- Chaque expression FLOWR est délimitée par { ... }.
- La forme générale d'une requête FLOWR est la suivante :
  - for *$variable* in *expression_recherche_Xpath*
  - let *$variable* := *expressions_Xpath*
  - where *expression_logique*
  - order by *$variable*
  - return *expression*

FLOWR

- Exemple d'expression FLWOR :
  - for $x in doc("books.xml")/bookstore/book
    where $x/price>30
    order by $x/title
    return $x/title
  - for - (facultatif) lie une variable à chaque élément retourné par l'expression
  - let - (facultatif) pour donner un autre nom à la variable
  - where - (facultatif) spécifie un critère de sélection
  - order by - (optionnel) spécifie le type d'ordre du résultat
  - return - spécifie ce que retourne le résultat

FLOWR

- Les conditions
  - doc("books.xml")/bookstore/book[price>30]/title
- L'expression FLWOR équivalente est :
  - for $x in doc("books.xml")/bookstore/book
    where $x/price>30
    return $x/title
  - for : sélectionne tous les éléments book des éléments bookstore dans une variable appelée $x.
  - where : filtre sur les éléments book avec price>30
  - order by : pour le trie
  - return : spécifie ce que retourne la requête.

La clause FOR

- Permet d'itérer sur une liste de fragments XML.
- Elle associe à chaque *$variable* un fragment de chemin XML défini par Xpath.
- Une seule clause for est insérée dans une expression FLWOR.
- Pour boucler un nombre fini de fois avec FOR, utiliser to :
  - for $x in (1 to 5)
    return <test>{$x}</test>

Ce qui donne:

```
<test>1</test>
<test>2</test>
<test>3</test>
<test>4</test>
<test>5</test>
```

La clause FOR

- Permet de mettre plusieurs expressions de chemins dans une clause FOR.
- Utilisez une virgule pour séparer les expressions de chemins :
  - for $x in (10,20), $y in (100,200)
    return <test>x={$x} and y={$y}</test>
- Ce qui donne:
  - <test>x=10 and y=100</test>
    <test>x=10 and y=200</test>
    <test>x=20 and y=100</test>
    <test>x=20 and y=200</test>

La clause FOR

- Le mot clé **at** est utilisé pour compter le nombre d'itérations:
  - for $x at $i in doc("books.xml")/bookstore/book/title
    return <book>{$i}. {data($x)}</book>
- Ce qui donne:
  - <book>1. Concepts Client/Serveur</book>
    <book>2. Services Web</book>
    <book>3. Relationnel DataBases</book>
    <book>4. Learning XML</book>

La clause let

- La clause *let* est optionnelle, elle permet d'associer le résultat d'une expression Xpath à une variable.
- Objectif : éviter de répéter la même expression.
- let ne se traduit pas par une itération.
  - let $x := (1 to 5)
    return <test>{$x}</test>
- Ce qui donne :
  - <test>1 2 3 4 5</test>

La clause where

- La clause *where* permet de définir une condition de sélection pour construire la réponse.
- La sélection se fait par une expression logique de prédicats élémentaires.
  - where $x/price>30 and $x/price<100
  - where contains($x/title,"XML") or not contains($x/author,"zellou")

La clause order by

- La clause *order* permet de trier les résultats (croissant au décroissant).
- Pour ordonner le résultat par category et title:
  - for $x in doc("books.xml")/bookstore/book
    order by $x/@category, $x/title
    return $x/title
- Ce qui donne:
  - <title lang="fr"> Concepts Client/Serveur </title>
    <title lang="fr"> Services Web </title>
    <title lang="en"> Relationnel DataBases </title>
    <title lang="en"> Learning XML </title>

La clause return

- Spécifie ce qui retourne la requête.
- Chaque itération doit retourner un seul fragment XML (pas une collection).
  - for $x in doc("books.xml")/bookstore/book
    return $x/title
- Ce qui donne :
  - <title lang="fr"> Concepts Client/Serveur </title>
    <title lang="fr"> Services Web </title>
    <title lang="en"> Relationnel DataBases </title>
    <title lang="en"> Learning XML </title>

Expressions conditionnelles

- Avec if/else.
  - for $x in doc("books.xml")/bookstore/book
    return if ($x/@category="web")
    then <web>{data($x/title)}</web>
    else <DB>{data($x/title)}</DB>
- Ce qui donne:
  - <DB> Concepts Client/Serveur </DB>
    <web> Services Web </web>
    <DB> Relationnel DataBases </DB>
    <web> Learning XML </web>

Généralités

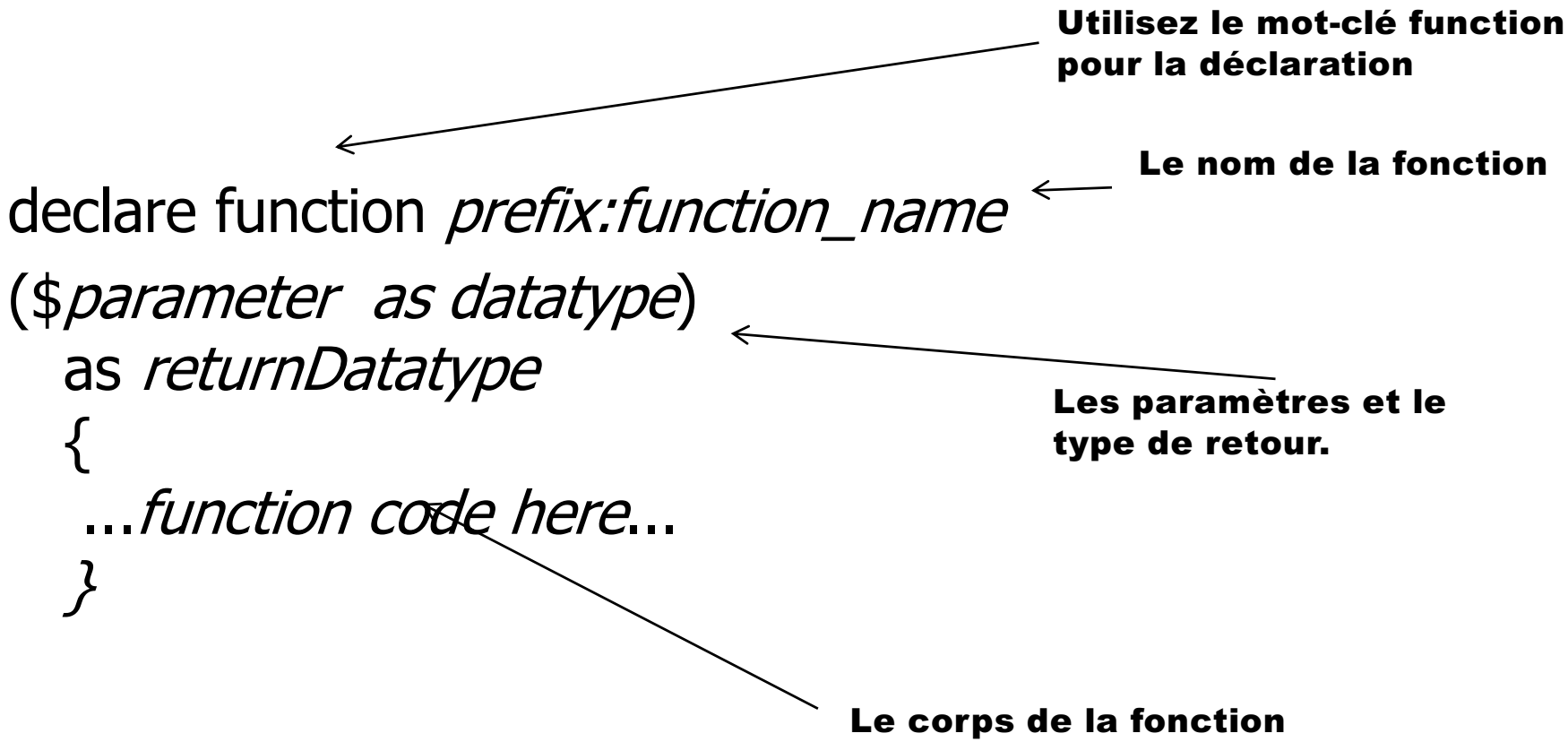
- Les expressions de requêtes peuvent être arbitrairement imbriquées.
- Il est possible d'imbriquer des requêtes au niveau de *for*, de *where*, et de return.
- Fonctions d'agrégations : count, min, max...
- Recherche textuelle : = , *contains* .

Fonctions XQuery

- XQuery, XPath et XSLT partagent la même bibliothèque de fonctions.
- Exemple 1: dans le prédicat d'une expression de chemin »
  - doc("books.xml")/bookstore/book[substring(title,1,3) ='XML']
- Exemple 2: dans une clause let
  - let $name := (max($price))
- Exemple 3: dans un élément
  - <name>{upper-case($booktitle)}</name>

Fonctions Utilisateurs

- Définir des fonctions avec XQuery.

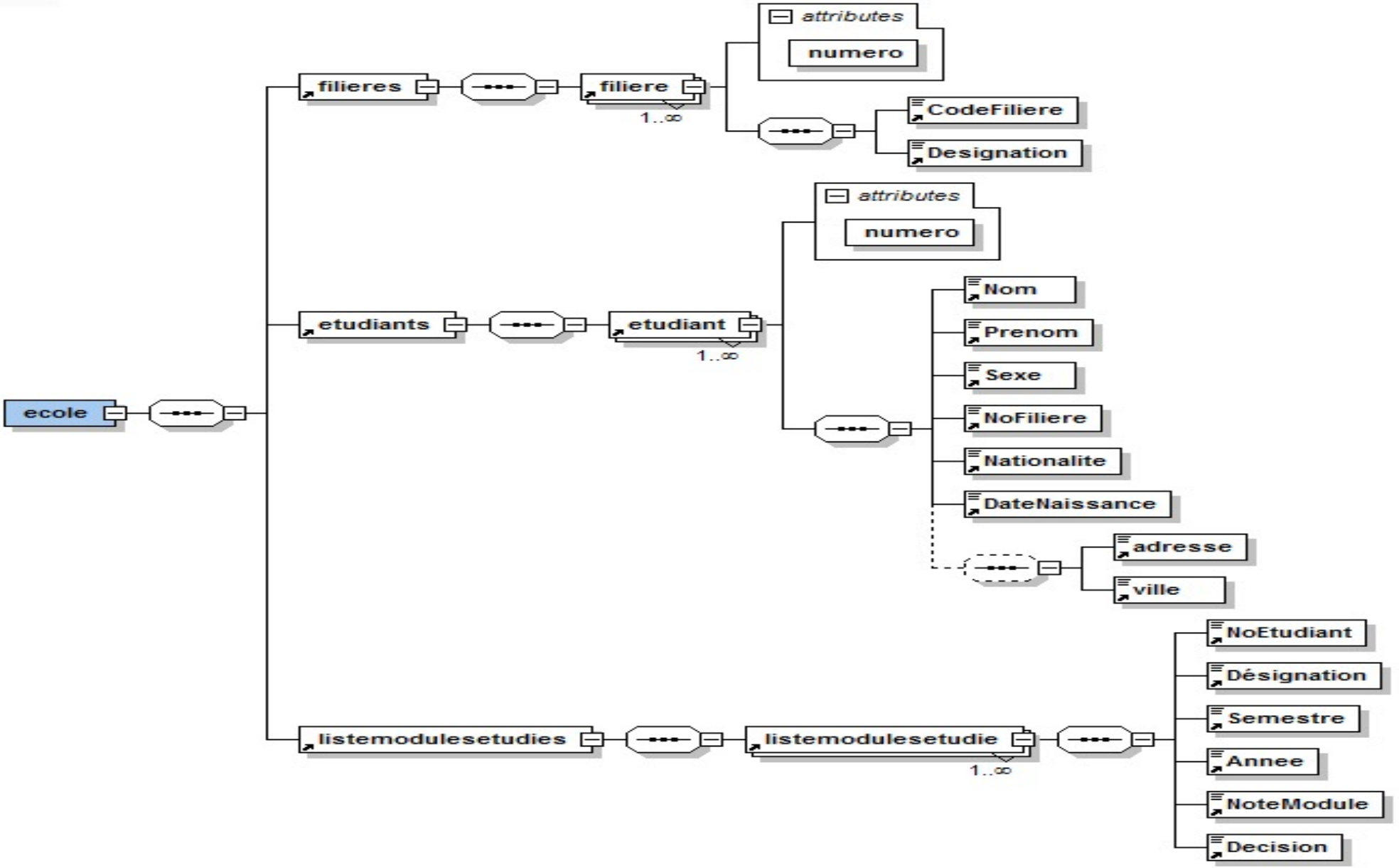


Fonctions Utilisateurs

- Exemple :
  - declare function local:minPrice($a as xs:decimal,$b as xs:decimal)
    AS xs:decimal
    {
        if ($a > $b) let $res := $b
            else let $res := $a
        return ($res)
    }
- L'appel à la fonction :
  - <minPrice>{local:minPrice($book/price,$book/discount)}</minPrice>

Exemples

Exemple : Schéma d'étude



Exemple : requête simple

- Les noms des étudiants:

```
xquery version "1.0";
for $x in doc("etudiants.xml")/ecole/etudiants/etudiant
return $x/Nom
```

- Les noms et prénoms des étudiants:

```
xquery version "1.0";
for $x in doc("etudiants.xml")/ecole/etudiants/etudiant
return <res>{$x/Nom}{$x/Prenom}</res>
```

Exemple : sélection 1/2

- Les conditions :

```
xquery version "1.0";
for $x in
   doc("etudiants.xml")/ecole/etudiants/etudiant[NoFilier
   e=14]
return <res>{$x/Nom}{$x/Prenom}</res>
```

- Les conditions :

```
xquery version "1.0";
for $x in doc("etudiants.xml")/ecole/etudiants/etudiant
where $x/NoFiliere=14
return <res>{$x/Nom}{$x/Prenom}</res>
```

Exemple : sélection 2/2

- Plusieurs conditions :

```
xquery version "1.0";
for $x in doc("etudiants.xml")/ecole/etudiants/etudiant
where $x/NoFiliere=14 and  $x/Sexe="F"
return <res>{$x/Nom}{$x/Prenom}</res>
```

- Une condition sur l'attribut :

```
xquery version "1.0";
for $x in doc("etudiants.xml")/ecole/etudiants/etudiant
where $x/@numero=71034
return <res>{$x/Nom}{$x/Prenom}</res>
```

Exemple : Imbrication 1/2

- Au niveau de where :

```
xquery version "1.0";
for $x in doc("etudiants.xml")/ecole/etudiants/etudiant
where $x/@numero in
    for $y in
    doc("etudiants.xml")/ecole/listemodulesetudies/listemodulesetudie
    where $y/Semestre="S1"
    return {$y/NoEtudiant/text()}
return <res>{$x/Nom}{$x/Prenom}</res>
```

Exemple : Imbrication 2/2

- Au niveau de return avec let :

```
xquery version "1.0";
for $x in doc("etudiants.xml")/ecole/etudiants/etudiant
let $y := $x/@numero
where $x/@numero=70288
return <res>
   {
   for $z in
   doc("etudiants.xml")/ecole/listemodulesetudies/listemodulesetudie
   where $z/Semestre="S1" and $z/NoEtudiant=$y
   return $z/Désignation
   }
</res>
```

Exemple : Jointure 1/2

- Sur un même fichier :

```
xquery version "1.0";
for $x in doc("etudiants.xml")/ecole/etudiants/etudiant
for $y in
   doc("etudiants.xml")/ecole/listemodulesetudies/listemo
   dulesetudie
where $x/@numero=$y/NoEtudiant
      return <res>
                  { $x/Nom}{ $y/Désignation}
                  { $y/NoteModule}{ $y/Decision}
            </res>
```

Exemple : Jointure 2/2

- Sur plusieurs fichiers :

```
xquery version "1.0";
for $x in doc("etudiants.xml")/ecole/etudiants/etudiant
for $y in doc("notes.xml")/ecole/listemodulesetudie
where $x/@numero=$y/NoEtudiant
    return <res>
                { $x/Nom}{ $y/Désignation}
                { $y/NoteModule}{ $y/Decision}
            </res>
```

Exemple : Agrégat simple

- Count (Pas de for) :

```
 xquery version "1.0";
let $x := doc("etudiants.xml")/ecole/etudiants/etudiant
return
<Nombreetudiants> {count($x)} </Nombreetudiants>
```

- Max (Pas de for) :

```
 xquery version "1.0";
let $x :=
doc("etudiants.xml")/ecole/listemodulesetudies/listemodul
esetudie/NoteModule
return
<Notemaximal> {max($x)} </Notemaximal>
```

Exemple : Agrégat partitionné

- Avg :

```
xquery version "1.0";
for $x in doc("etudiants.xml")//modules/Désignation
let $y :=
   avg(doc("etudiants.xml")//listemodulesetudie[Désignati
   on = $x]/NoteModule)
return
  <resultat>
       {$x/Désignation}
    <notemoyenne>{$y}</notemoyenne>
  </resultat>
```

Exemple : Recherche

- contains :

```
xquery version "1.0";
for $x in doc("etudiants.xml")//etudiant
   where contains ($x/Nom,'AD')
                or contains ($x/Prenom, 'YASS')
return <res>{$x/Nom}{$x/Prenom}</res>
```

Exemple : ordre

- contains (Pas de for) :

```
xquery version "1.0";
for $x in doc("etudiants.xml")//etudiant
   where contains ($x/Nom,'AD')
                or contains ($x/Prenom, 'YASS')
return <res>{$x/Nom}{$x/Prenom} sort by {$x/Nom
   descending}</res>
```

XQuery Update

- XQuery Update est une extension de XQuery.
- C'est une recommandation W3C du 17 Mars 2011.
- Il permet de mettre à jour un fichier XML.
- L'Extensions XQuery Update : insert, replace, value, delete et rename
- Une déclaration update peut se faire dans n'importe quelle partie d'une requête XQuery.

XQuery Update

- XQUF permet de :
  - supprimer un ou plusieurs éléments ;
  - insérer un ou plusieurs éléments, avant/après/à l'intérieur d'un élément ;
  - remplacer un élément (avec tout son sous-arbre) par une séquence d'éléments ;
  - remplacer les fils d'un élément par une séquence d'éléments ;
  - remplacer la valeur d'un élément par une valeur textuelle ;
  - renommer un élément.

1. Insert

- Syntaxe :
  - insert (node | nodes) items into expr
  - insert (node | nodes) items as first into expr
  - insert (node | nodes) items as last into expr
  - insert (node | nodes) items before expr
  - insert (node | nodes) items after expr
- expr doit désigner un élément cible.

1. Insert

- Exemple 1:
  - for $x in /biblio/livre
    where contains ($x/titre,'XML')
    return insert node
    <resume>un doc XML</resume>
    into $x
- Exemple 2 :
  - for $x in //livre[DateEdition = 1998]
    return insert node
    <reduction> 10% </reduction>
    after (or befor) $x

2. Replace

- Syntaxe :
  - replace node expr with items
- expr peut être un élément, un attribut ou du texte.
- Exemple 1:
  - for $x in //livre/resume

    return replace node $x

    with <RESUME> {data($x)} </RESUME>

2. Replace

- return replace node $x
- Exemple 2:
  - for $x in //livre/RESUME
    return replace node $x
    with <summary>XQUF</summary>
- Exemple 3:
  - for $x in /biblio/livre[2]/prix
    return replace node $x
    with <prix> {data($x) *0.8 }</prix>

3. Replace Value

- Syntaxe
  - replace value of node expr with exprSingle
- Mette à jour le contenu de tous les nœuds expr avec les éléments de exprSingle.
- Exemple 1:
  - for $x in /biblio/livre/prix
    return replace value of node $x
    with 30

4. delete

- Syntaxe
  - delete (node | nodes) expr
- Supprime tous les nœuds dans expr du document.
- Exemple:
  - for $x in /biblio/livre
    return delete node $x/prix

  - for $x in //livre[prix >30]
    return delete node $x

5. Rename

- Syntaxe
  - rename node expr as exprSingle
- Renomme les nœuds dans expr en utilisant la valeur de exprSingle.
- expr peut être un ensemble d'éléments ou d'attributs.
- Exemple:
  - for $x in /biblio/livre/resume
    return rename node $x
    as "abstract"

Remarques

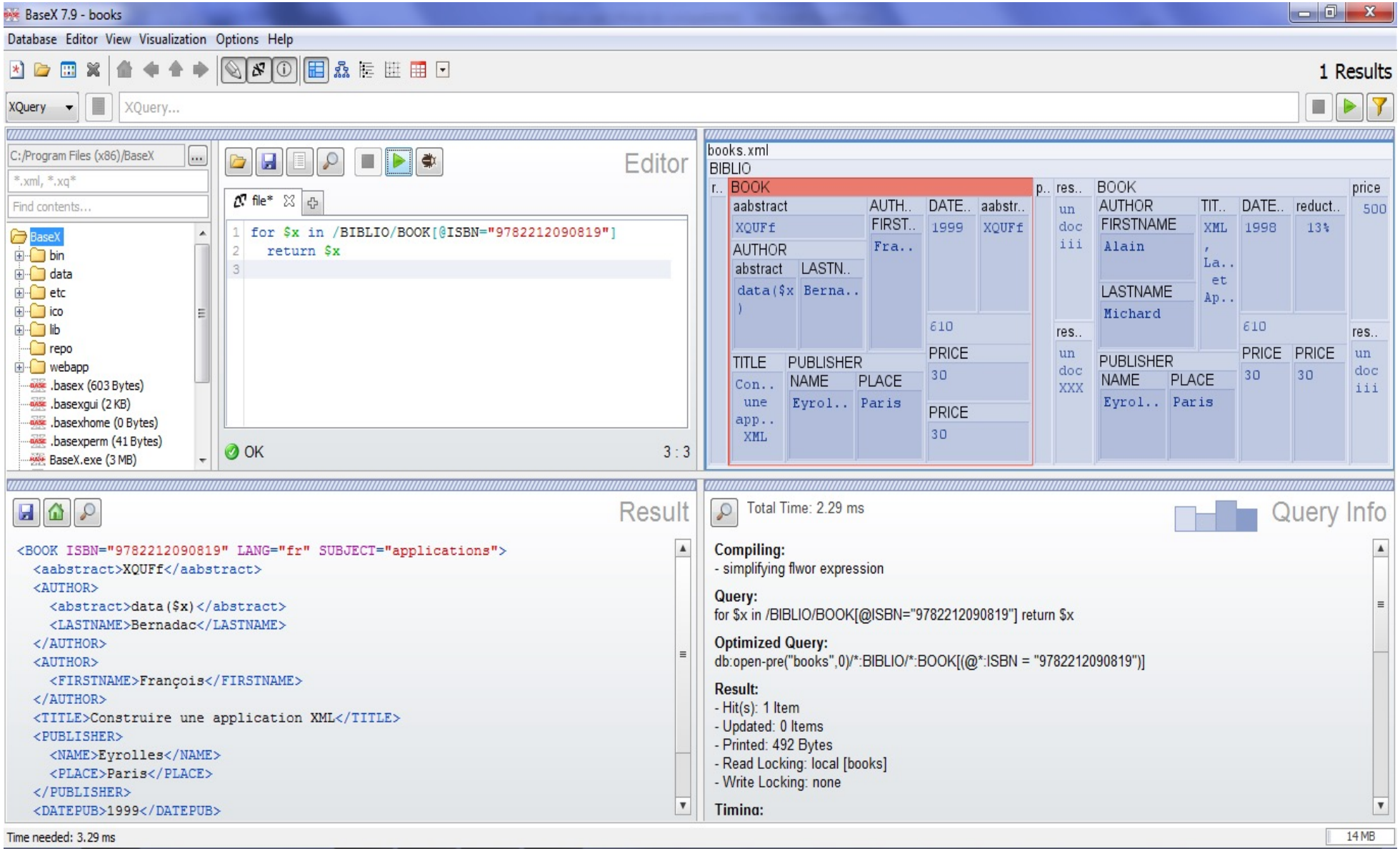
- Il y a 2 façons d'utiliser XQUF :
  - Pour mettre à jour une base de données : insert, delete, rename, replace
  - Sans effet sur la base, avec fabrication d'un arbre XML : copy ... modify ... return.

- Quelques implémentations: Monet DB (CWI), Qizx (XMLmind), Oracle Berkeley DB XML (Oracle), Xqilla, baseX, eXist, altova 2018, …

BaseX

- BaseX est un SGBD XML native
- SGBDS représentatif, open-sources, multi-plate-formes, et support des standard.
- Plusieurs modes de visualisation : tableau, arbre, 3D, etc.
- Moteur de recherche en texte intégral
- Evaluation des requêtes Xquery et XUF
- BaseX ne permet pas l'édition du fichier XML source en cours d'exploitation.
- API supportées : XQuery for PHP (XQP), XQuery for Java (XQJ) and XML:DB (XAPI).

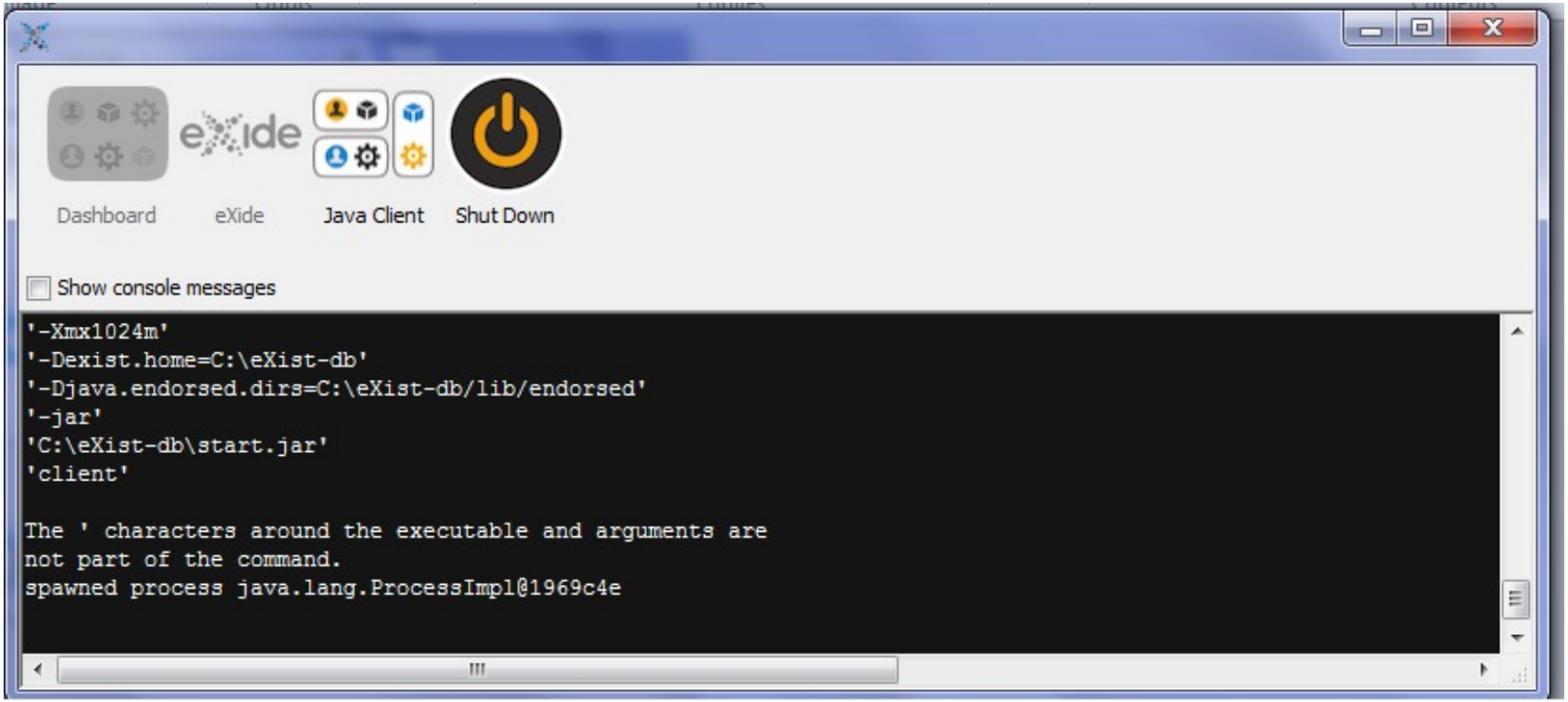
BaseX


BaseX 7.9 - books
Database Editor View Visualization Options Help
1 Results
XQuery
XQuery...
C:/Program Files (x86)/BaseX
*.xml, *.xq*
Find contents...
BaseX
bin
data
etc
ico
lib
repo
webapp
.basex (603 Bytes)
.basexgui (2 KB)
.basexhome (0 Bytes)
.basexperm (41 Bytes)
BaseX.exe (3 MB)
Editor
file*
1 for $x in /BIBLIO/BOOK[@ISBN="9782212090819"]
2 return $x
3
OK
3 : 3
books.xml
BIBLIO
Result
<BOOK ISBN="9782212090819" LANG="fr" SUBJECT="applications">
<aabstract>XQUFf</aabstract>
<AUTHOR>
<abstract>data($x)</abstract>
<LASTNAME>Bernadac</LASTNAME>
</AUTHOR>
<AUTHOR>
<FIRSTNAME>François</FIRSTNAME>
</AUTHOR>
<TITLE>Construire une application XML</TITLE>
<PUBLISHER>
<NAME>Eyrolles</NAME>
<PLACE>Paris</PLACE>
</PUBLISHER>
<DATEPUB>1999</DATEPUB>
Total Time: 2.29 ms
Query Info
Compiling:
- simplifying flwor expression
Query:
for $x in /BIBLIO/BOOK[@ISBN="9782212090819"] return $x
Optimized Query:
db:open-pre("books",0)/*:BIBLIO/*:BOOK[(@*:ISBN = "9782212090819")]
Result:
- Hit(s): 1 Item
- Updated: 0 Items
- Printed: 492 Bytes
- Read Locking: local [books]
- Write Locking: none
Timing:
Time needed: 3.29 ms
14 MB

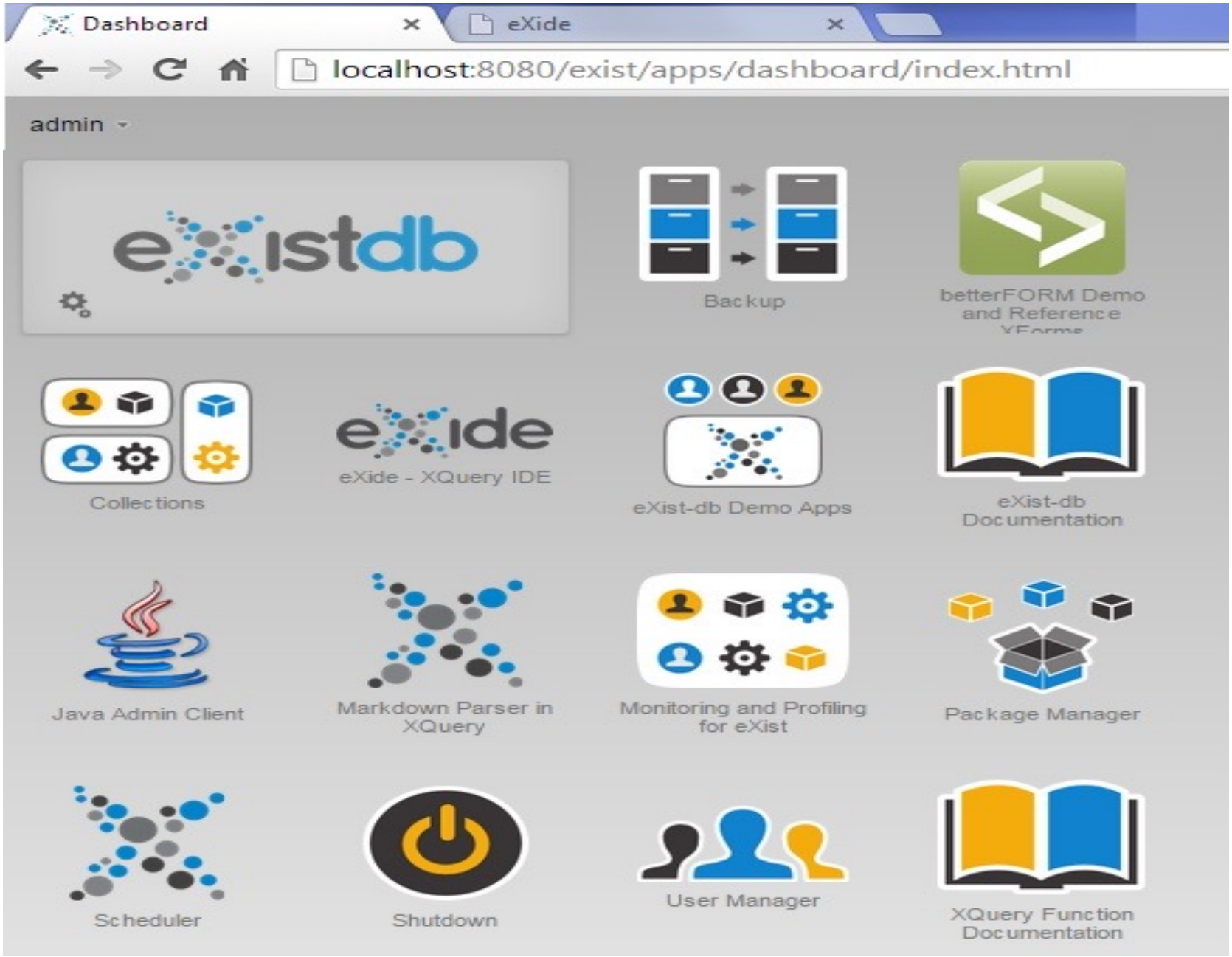
eXist

- eXist est un SGBD open source, Multi plates-formes avec usage format Desktop ou Web.
- Se déploie comme une application Web sous le contrôle d'un moteur de Servlets : Tomcat ou autre.
- Possibilités d’administration avancée (triggers, indexation, etc.)
- Plusieurs API (Java, PHP, C#, etc.) pour interrogation à distance
- Support des standards (Xquery et XUF)

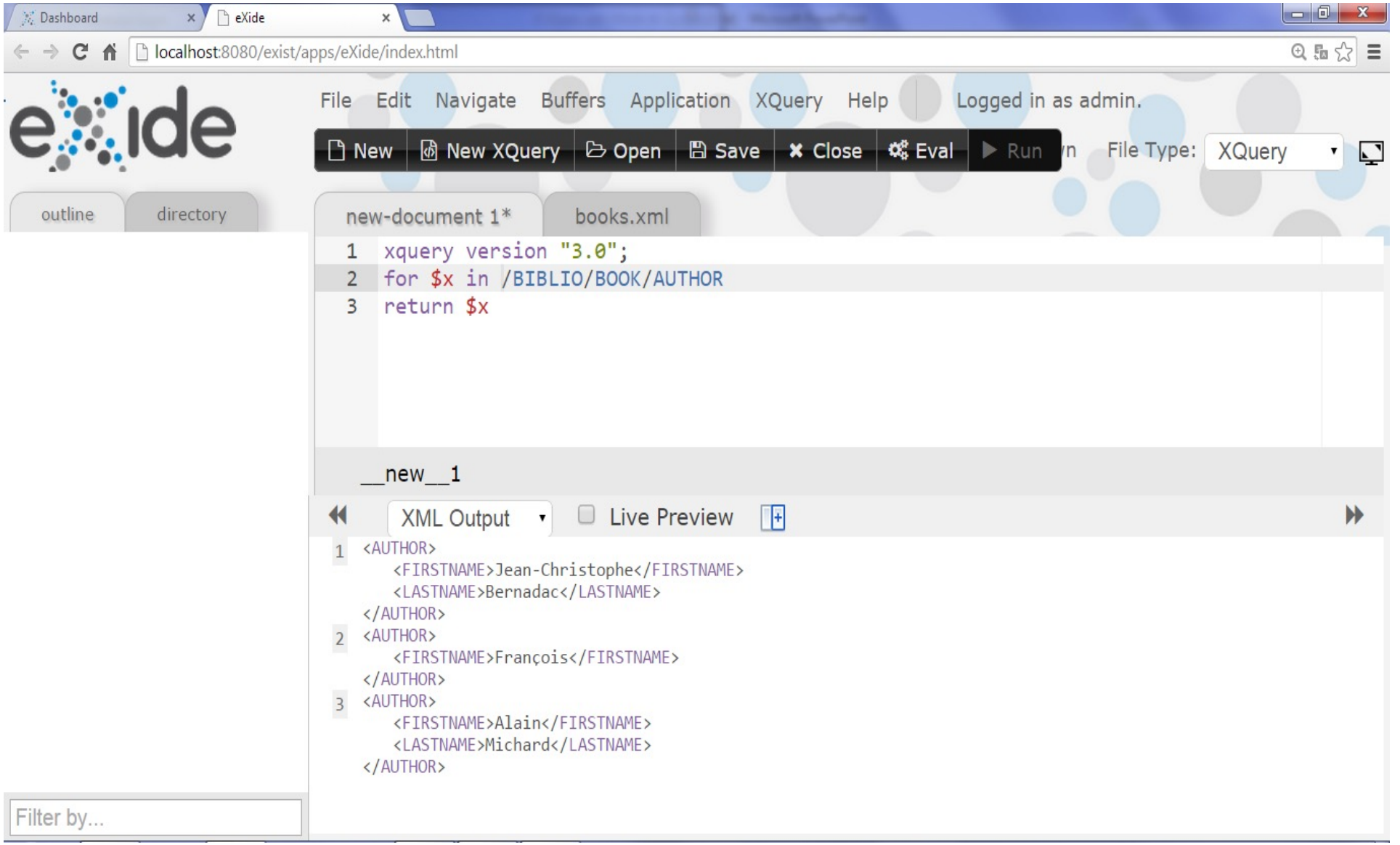
eXist


Dashboard
eXide
Java Client
Shut Down
Show console messages
'-Xmx1024m'
'-Dexist.home=C:\eXist-db'
'-Djava.endorsed.dirs=C:\eXist-db/lib/endorsed'
'-jar'
'C:\eXist-db\start.jar'
'client'
The ' characters around the executable and arguments are
not part of the command.
spawned process java.lang.ProcessImpl@1969c4e

eXist


Dashboard
eXide
localhost:8080/exist/apps/dashboard/index.html
admin
Backup
betterFORM Demo and Reference
Collections
eXide - XQuery IDE
eXist-db Demo Apps
eXist-db Documentation
Java Admin Client
Markdown Parser in XQuery
Monitoring and Profiling for eXist
Package Manager
Scheduler
Shutdown
User Manager
XQuery Function Documentation

eXist


Dashboard
eXide
localhost:8080/exist/apps/eXide/index.html
File Edit Navigate Buffers Application XQuery Help
Logged in as admin.
New
New XQuery
Open
Save
Close
Eval
Run
File Type:
XQuery
outline
directory
new-document 1*
books.xml
1 xquery version "3.0";
2 for $x in /BIBLIO/BOOK/AUTHOR
3 return $x
__new__1
XML Output
Live Preview
1 <AUTHOR>
<FIRSTNAME>Jean-Christophe</FIRSTNAME>
<LASTNAME>Bernadac</LASTNAME>
</AUTHOR>
2 <AUTHOR>
<FIRSTNAME>François</FIRSTNAME>
</AUTHOR>
3 <AUTHOR>
<FIRSTNAME>Alain</FIRSTNAME>
<LASTNAME>Michard</LASTNAME>
</AUTHOR>
Filter by...