



Bases de données XML

Cahier de TPs

Copyright - OXIANE, 98 avenue du Gal Leclerc, 92100 Boulogne. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de OXIANE et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de OXIANE.

OXIANE, le logo OXIANE sont des marques de fabrique ou des marques déposées, ou marques de service, de OXIANE en France et dans d'autres pays.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REpondre A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.

Copyright - OXIANE, 98 avenue du Gal Leclerc, 92100 Boulogne. All rights reserved .

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of OXIANE and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from OXIANE suppliers.

OXIANE, the OXIANE logo are trademarks, registered trademarks, or service marks of OXIANE in France and other countries.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

TP 1 - Premières expressions XQuery

Dans ce TP, nous allons écrire les premières expressions XQuery permettant de produire du contenu. Pour exécuter ces expressions, nous utiliserons Saxon-HE, de Saxonica. Saxon-HE est la version Open Source - Home Edition - de Saxon, qui est un processeur XSLT et XQuery écrit en Java.

Dans les ressources de TP, Saxon-HE est fourni. Pour le lancer, en ligne de commande :

> sous UNIX: `./saxon/bin/saxon -q:query.xq -s:sourceFile.xml`

> sous windows: `saxon\bin\saxon -q:query.xq -s:sourceFile.xml`

Où :

> **query.xq** est le fichier qui contient l'expression XQuery

> **sourceFile.xml** est le fichier source sur lequel appliquer la requête. Ce paramètre est optionnel

TP1.1 : date courante

> Ecrire une expression permettant de renvoyer un document XML contenant la date courante.

> Exécuter cette expression de la façon suivante :

```
./saxon/bin/saxon -q:TP1/today.xq
```

Correction

```
<aujourd'hui>{current-date()}</aujourd'hui>
```

TP1.2 : description d'une formation

> En utilisant le fichier **catalogue.xml** fourni dans l'init du TP1, afficher la description de la formation dont l'identifiant est **XS-XSL2**.

> Attention de ne pas restituer la balise description, mais uniquement son contenu.

Correction

```
declare namespace ox = "com:oxiane:formation";

<Formation-XSL>
  {doc('../..//TP1/catalogue.xml')/ox:formations/ox:formation[@id eq
  'XS-XSL2']/description/string()}
</Formation-XSL>
```

TP 2 : Expressions FLOWR simples

TP2.1 : identifiants des formations

- > A partir du catalogue du TP1, écrire une expression XQuery permettant d'afficher l'identifiant de toutes les formations.

Correction

```
declare namespace ox = "com:oxiane:formation";

for $form in
doc('../..//TP1/catalogue.xml')/ox:formations/ox:formation
return <formation>{$form/@id}</formation>
```

TP 2.2 : trier par titre

- > Enrichir cette expression pour trier les formations par titre

TP 2.3 : filtrer

- > A l'aide d'une clause WHERE, n'afficher que les formations dont l'identifiant contient 'X'

Correction

```
declare namespace ox = "com:oxiane:formation";

for $form in
doc('../..//TP1/catalogue.xml')/ox:formations/ox:formation
order by $form/titre/text() ascending
where $form/contains(@id, 'X')
return <formation>{$form/@id}</formation>
```

TP 3 : Jointures

Dans ce TP, nous allons réaliser des jointures entre plusieurs sources de données, de façon progressive.

TP 3.1 : Jointure Simple

- > Produire, à partir de deux séquences d'entiers de 1 à 10, la liste des produits des entiers.
- > Produire une séquence d'éléments td contenant le résultat de la multiplication. On devrait obtenir 100 éléments td.

Correction

```
for $x in (1, 2, 3, 4, 5, 6, 7, 8, 9, 10),  
    $y in (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)  
return <td>{$x * $y}</td>
```

TP 3.2 Mettre les résultats en lignes

- > Modifier sensiblement l'expression XQuery de façon à grouper les cellules en lignes.
- > Ajouter une colonne à gauche pour afficher le multiplicateur.

Correction

```
for $y in (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)  
return  
  <tr><th>{$y}</th> {  
    for $x in (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)  
    return <td>{$x * $y}</td>  
  }</tr>
```

TP 3.3 Produire un HTML visualisable

- > Ajouter une ligne d'entête avec les opérandes.
- > Puis ajouter l'enveloppe html. Enregistrer la sortie dans un fichier, et regarder le fichier produit dans un navigateur.

Correction

```
<html>  
  <head>  
    <title>Tricher pour ses tables de multiplication</title>  
  </head>  
  <boby>  
    <table>{  
      (  
        <tr><th/>{for $x in (1, 2, 3, 4, 5, 6, 7, 8, 9, 10) return  
        <th>{$x}</th>}</tr>,&br/>      )  
    }</table>
```

```
for $y in (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
return
  <tr><th>{$y}</th> {
    for $x in (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
      return <td>{$x * $y}</td>
    }
  </tr>
}</table></boby>
</html>
```

TP 3.4 Jointures multiples

- Etre attentif a bien lister tous les auteurs de chaque livre.

Correction

```
declare namespace li = "com:oxiane:formation:xml:livres";
declare namespace at = "com:oxiane:formation:xml:auteurs";

for $li in doc("../..//TP3/livres.xml")/li:livres/li:livre
return <livre>
<titre>{$li/li:titre/string()}</titre>
{
  for $at in doc("../..//TP3/auteurs.xml")/at:auteurs/at:auteur[at:id
= $li/li:auteurs/li:auteur/@refid]
    return <auteur>{$at/string-
join((at:prenom/string(),at:nom/string()),' ')}</auteur>
}
</livre>
```

TP 3.5 Jointures ouvertes

- > Puis afficher la liste des auteurs et de leurs éventuels livres. Veiller à ce que tous les auteurs soient bien listés, y compris ceux pour lesquels aucun livre n'existe.

Corrections

```
declare namespace li = "com:oxiane:formation:xml:livres";
declare namespace at = "com:oxiane:formation:xml:auteurs";

for $at in doc("../..//TP3/auteurs.xml")/at:auteurs/at:auteur
for $li allowing empty in
doc("../..//TP3/livres.xml")/li:livres/li:livre[li:auteurs/li:auteur/
@refid = $at/at:id/string()]
return
<auteur>
  <nom>{$at/string-join((at:prenom/string(),at:nom/string()),'
')}</nom>
  {if(not(empty($li))) then $li/li:resume/string() else ()}
</auteur>
```

TP 4 - Groupage

Chaque livre de notre collection a un ISBN, **International Standard Book Number**. En 2007, le code ISBN est passé de 10 à 13 chiffres, parce que la norme des codes barres, basée sur GTIN-13, est sur 13 caractères. Ces 10 ou 13 chiffres peuvent être séparés par des tirets

TP 4.1

- > Produire, à l'aide d'une expression de groupage, un document XML listant les vieux livres, d'avant 2007, et les plus récents. Seuls les titres et le code ISBN sont intéressants.
- > Afin de déterminer si un livre est ancien ou récent, on utilisera le nombre de caractères numériques contenus dans le code ISBN. Pour cela, on supprimera tous les caractères non numériques par rien avec la fonction **replace** et on utilisera la fonction **string-length** pour connaître la longueur de la chaîne de caractères obtenus.
- > L'expression régulière permettant d'obtenir un caractère non-numérique est '**[^0-9]**'.
- > On peut utiliser l'**arrow-operator** qui permet de chaîner des fonctions de façon assez lisible. Dans ce cas, on omet le premier argument de la fonction. '**test**'=>**string-length()** Ceci renvoie 4.

Corrections

```
declare namespace li = "com:oxiane:formation:xml:livres";

<livres>{
  for $livre in doc("../TP3/livres.xml")/li:livres/li:livre
  let $isbnLength := $livre/li:isbn=>replace('[^0-9]', '')=>string-length()
  group by $isbnLength
  return
    element {if($isbnLength eq 10) then 'vieux' else 'jeune'}}{for $l
in $livre return

  <livre><isbn>{$l/li:isbn/text()}</isbn><titre>{$l/li:titre/text()}</
titre></livre>
}
}</livres>
```

TP 4.2

- > Faire en sorte d'afficher d'abord les livres récents, puis les livres anciens.

Correction

```
declare namespace li = "com:oxiane:formation:xml:livres";

<livres>{
  for $livre in doc("../TP3/livres.xml")/li:livres/li:livre
  let $isbnLength := $livre/li:isbn=>replace('[^0-9]', '')=>string-length()
  group by $isbnLength
  return
    element {if($isbnLength eq 10) then 'vieux' else 'jeune'}}{for $l
in $livre return

  <livre><isbn>{$l/li:isbn/text()}</isbn><titre>{$l/li:titre/text()}</
titre></livre>
}
}</livres>
```

```
group by $isbnLength
order by $isbnLength descending
return
  element {if($isbnLength eq 10) then 'vieux' else 'jeune'}{for $l
in $livre return
  <livre><isbn>{$l/li:isbn/text()}</isbn><titre>{$l/li:titre/text()}</
titre></livre>
}
}</livres>
```

Dans ces deux exercices, on constate que l'ordre des éléments dans le document source n'est pas conservé dans la sortie. En effet, deux groupes sont constitués, et à l'intérieur de chacun des groupes, les éléments sont ordonnés suivant l'ordre du document, mais sur l'ensemble de la sortie, les livres ne sont pas strictement dans l'ordre où ils apparaissaient dans le document.

TP5 - Fenêtrage

Au TP4, nous avons groupé les livres en deux groupes distincts, mais sans conserver l'ordre des éléments dans le document original.

L'objectif, maintenant, est de conserver l'ordre des livres tels qu'ils sont dans le document original, tout en les groupant dans des balises **jeune** ou **vieux**.

L'idée est d'obtenir quelque chose ressemblant à :

```
<livres>
  <vieux>
    <livre><isbn>1-234-5678-90</isbn><livre>
  </vieux>
  <jeune>
    <livre><isbn>1-234-56-789012-3</isbn></livre>
    <livre><isbn>2-345-67-890123-1</isbn></livre>
  </jeune>
  <vieux>
    <livre><isbn>2-345-6789-01</isbn><livre>
  </vieux>
</livres>
```

TP 5.1 Groupage par le début

- > Ecrire une expression avec fenêtrage adjacent, où chaque fenêtre commence soit lorsque la position de l'item est 1, soit lorsque sa longueur est différente de la longueur de la précédente.

Corrections

```
declare namespace li = "com:oxiane:formation:xml:livres";

<livres>{
  for tumbling window $win in
  doc("../TP3/livres.xml")/li:livres/li:livre
  start
    $s at $s-pos
    previous $p
    when
      $s-pos eq 1 or
      ($s/li:isbn=>replace('[^0-9]', '')=>string-length() !=
      $p/li:isbn=>replace('[^0-9]', '')=>string-length())
    return
      element {if($win[1]/li:isbn=>replace('[^0-9]', '')=>string-
length() eq 10) then 'vieux' else 'jeune'}}{for $l in $win return

<livre><isbn>{$l/li:isbn/text()}</isbn><titre>{$l/li:titre/text()}</
titre></livre>
}
}</livres>
```

TP 5.2 Mise en forme de la sortie

Avec XQuery, on peut décider de la mise en forme du résultat, comme en XSL avec **s:output**. Des options du prologue permettent de définir le format de sortie que l'on souhaite. Il y a juste une méthode qui existe dans XQuery et qui n'existe pas avec XSL : **adaptative**. Elle permet d'adapter l'une des autres méthodes à l'item. Parfois, tous les items ne sont pas du même type et ne nécessitent pas les mêmes options de sérialisation.

Ajouter le prologue suivant à la précédente requête, et observer le résultat :

```
declare namespace output = "http://www.w3.org/2010/xslt-xquery-serialization";  
declare option output:method "xml";  
declare option output:indent "yes";  
declare option output:omit-xml-declaration "yes";
```

Attention de bien utiliser le bon espace de nommage ; si il y a une erreur dedans, les options ne seront pas reconnues, et la mise en forme ne se fera pas.

TP 6 Fonctions

L'utilisation de fonctions permet de factoriser le code, de le réutiliser, et d'écrire des tests unitaires. Nous allons ici réutiliser les exercices précédents et factoriser le code en fonctions.

TP 6.1 Ecrire une fonction calculant le nombre de caractères de l'ISBN

L'ISBN est une suite de 10 ou 13 chiffres, de 0 à 9. Il est souvent découpé en groupes de chiffres, soit avec des tirets, soit avec des espaces. Ces tirets ou espaces n'ont aucune signification dans notre utilisation pour connaître l'ancienneté d'un ouvrage. Nous allons donc écrire une fonction permettant de calculer le nombre de chiffres dans une chaîne de caractères. Comme cette fonction n'est pas liée au métier de l'ISBN, nous l'appellerons **digitLength**.

Copier la requête écrite au TP 5.2 dans un nouveau fichier TP6.1.xq. Déclarer un nouvel espace de nommage, qui vous est propre, ou utiliser **com:oxiane:formation:xq:common:string**. Entre le prologue et la requête, déclarer une fonction renvoyant un entier (**xs:integer**) et ayant un paramètre de type chaîne de caractères (**xs:string**). Récupérer l'expression de calcul de longueur de l'ISBN et l'adapter pour la mettre dans la fonction.

Enregistrer, exécuter, rien ne doit changer.

Remplacer dans la requête les trois expressions de calcul de longueur par un appel de la fonction créée, et vérifier que le résultat est identique. Observer la lisibilité de l'expression XQuery.

Correction

```
declare namespace li = "com:oxiane:formation:xml:livres";
declare namespace output = "http://www.w3.org/2010/xslt-xquery-serialization";
declare namespace str = "com:oxiane:formation:xq:common:string";

declare option output:method "xml";
declare option output:indent "yes";
declare option output:omit-xml-declaration "yes";

declare function str:digitLength($s as xs:string) as xs:integer {
  $s=>replace('[^0-9]', '')=>string-length()
};

<livres>{
  for tumbling window $win in
  doc("../TP3/livres.xml")/li:livres/li:livre
  start
    $s at $s-pos
    previous $p
    when
      $s-pos eq 1 or
      ($s/li:isbn=>str:digitLength() != $p/li:isbn=>str:digitLength())
  return
    element {if($win[1]/li:isbn=>str:digitLength() eq 10) then
  'vieux' else 'jeune'}}{for $l in $win return

<livre><isbn>{$l/li:isbn/text()}</isbn><titre>{$l/li:titre/text()}</
titre></livre>
```

```
}  
</livres>
```

TP 7 Librairie de fonctions sur l'ISBN

L'ISBN est codifié. Le découpage à l'aide de tirets permet de constituer des groupes de chiffres ayant une signification.

Dans l'ISBN 10, le premier groupe de chiffres est le Domaine ISBN, ou zone de chalandise, ou langue. Le second groupe est le code éditeur, qui peut être sur 2, 3, 4, 5, 6 ou 7 caractères, en fonction du nombre de titres disponibles dans le catalogue de l'éditeur. Le troisième groupe est le numéro de publication de l'éditeur. Enfin, le dernier groupe est une clé de contrôle.

Dans l'ISBN 13, les trois premiers chiffres indiquent que c'est un livre ; 978 ou 979 quand 978 sera épuisé. L'ensemble de la codification est régie par l'ISO-2108.

TP 7.1 : isoler la fonction `digitLength` dans un module

Créer un nouveau fichier **strings.xqm** qui contiendra la fonction **str:digitLength** créée au précédent TP.

Correction

```
module namespace str = "com:oxiane:formation:xq:common:string";

declare function str:digitLength($s as xs:string) as xs:integer {
    $s=>replace('[^0-9]', '')=>string-length()
};
```

Ce fichier n'est pas exécutable. En effet, il ne contient qu'une fonction, et ne peut pas être lancé directement.

TP 7.2 Librairie ISBN

- > Créer un fichier **isbn.xqm**, qui contiendra l'ensemble des fonctions liées à l'ISBN. Y déclarer un espace de nommage **com:oxiane:formation:xq:libs:isbn**.
- > Importer dans ce fichier le module **strings.xqm**, avec son espace de nommage.

```
import module namespace str =
    "com:oxiane:formation:xq:common:string" at "strings.xqm";
```

- > Créer une première fonction qui vérifie que le code passé est bien un livre (donc qui commence bien par **'978'** ou par **'979'**). Cette fonction prend une chaîne de caractères en paramètre et renvoie un boolean. Les commentaires en XQuery sont encadrés par (: et :). Ce n'est pas très joli, mais cela permet de documenter les fonctions. Ajouter la documentation de la fonction. De façon générale, toute fonction publique doit être documentée, avec une description claire de ce qu'elle fait d'un point de vue métier, une description de ses paramètres et de ce qu'elle renvoie.
- > Créer une deuxième fonction qui 'normalise' le code ISBN, en lui enlevant les caractères de tête si c'est un code sur 13 caractères.
- > Créer une troisième fonction qui prend en paramètre une code ISBN, et renvoie une séquence de chaînes de caractères, une par groupe du code ISBN 10. La fonction **tokenize(string,**

pattern) permet de découper une chaîne de caractères suivant un pattern (expression rationnelle).

- > Créer une fonction privée qui prend en paramètre 4 chaînes de caractères et qui renvoie le domaine ISBN.
- > Créer une fonction privée qui prend en paramètre 4 chaînes de caractères et qui renvoie le code éditeur.
- > Créer une fonction privée qui prend en paramètre 4 chaînes de caractères et qui renvoie le numéro d'édition.
- > Créer une fonction privée qui prend en paramètre 4 chaînes de caractères et qui renvoie le code de contrôle.
- > Créer 4 fonctions qui prennent en paramètre une chaîne de caractères et qui renvoient respectivement le domaine ISBN, le code éditeur, le numéro de publication, et le code de contrôle. Ces 4 fonctions utiliseront les précédentes.

Correction

```
module namespace isbn = "com:oxiane:formation:xq:isbn";
import module namespace str =
"com:oxiane:formation:xq:common:string" at "strings.xqm";

(:
  Cette fonction vérifie que le code passé correspond bien à un
  ISBN.
  Soit le nombre de chiffres dans le code est 10, soit il commence
  par
  978 ou 979.
  @param $code Le code ISBN à tester
  @return true si c'est ISBN, false sinon.
: )
declare function isbn:isLivres($code as xs:string) as xs:boolean {
  let $length as xs:integer := str:digitLength($code)
  return
    if($length eq 10) then true()
    else matches('^97[89].*$',$code)
};

(:
  Renvoie à partir d'un code ISBN valide la version en ISBN 10
  @param $code L'ISBN à normaliser
  @return La version sur 10 digits, avec les séparateurs de groupe.
: )
declare function isbn:normalize($code as xs:string) as xs:string {
  if(str:digitLength($code) eq 10) then $code
  else $code=>substring(5)
};

(:
  Cette fonction renvoie les différents groupes de l'ISBN sous la
  forme d'une séquence.
  @param $code L'ISBN à découper
  @return Une séquence de 4 chaînes de caractères.
: )
```

```
:)
declare function isbn:split($code as xs:string) as xs:string+ {
    isbn:normalize($code)=>tokenize('[^0-9]')
};

declare %private function isbn:getDomain($domain as xs:string,
$editor as xs:string, $publi as xs:string, $control as xs:string) as
xs:string {
    $domain
};

declare %private function isbn:getEditor($domain as xs:string,
$editor as xs:string, $publi as xs:string, $control as xs:string) as
xs:string {
    $editor
};

declare %private function isbn:getPublication($domain as xs:string,
$editor as xs:string, $publi as xs:string, $control as xs:string) as
xs:string {
    $publi
};

declare %private function isbn:getControl($domain as xs:string,
$editor as xs:string, $publi as xs:string, $control as xs:string) as
xs:string {
    $control
};

(:
    Renvoie le domaine ISBN à partir d'un code ISBN sur 10 ou 13
    caractères
    @param $code L'ISBN à traiter
    @return Le code domain de l'ISBN
: )
declare function isbn:getDomain($code as xs:string) as xs:string {
    let $seq as xs:string+ := isbn:split(isbn:normalize($code))
    return isbn:getDomain($seq[1], $seq[2], $seq[3], $seq[4])
};

(:
    Renvoie le code éditeur à partir d'un code ISBN sur 10 ou 13
    caractères
    @param $code L'ISBN à traiter
    @return Le code éditeur de l'ISBN
: )
declare function isbn:getEditor($code as xs:string) as xs:string {
    let $seq as xs:string+ := isbn:split(isbn:normalize($code))
    return isbn:getEditor($seq[1], $seq[2], $seq[3], $seq[4])
};

(:
    Renvoie le numéro de publication à partir d'un code ISBN sur 10 ou
    13 caractères
    @param $code L'ISBN à traiter
    @return Le numéro de publication de l'ISBN
: )
```

```
:)
declare function isbn:getPublication($code as xs:string) as
xs:string {
  let $seq as xs:string+ := isbn:split(isbn:normalize($code))
  return isbn:getPublication($seq[1], $seq[2], $seq[3], $seq[4])
};

(:
  Renvoie le code contrôle à partir d'un code ISBN sur 10 ou 13
  caractères
  @param $code L'ISBN à traiter
  @return Le code contrôle de l'ISBN
:~)
declare function isbn:getControl($code as xs:string) as xs:string {
  let $seq as xs:string+ := isbn:split(isbn:normalize($code))
  return isbn:getControl($seq[1], $seq[2], $seq[3], $seq[4])
};
```

TP 7.3 Lister les livres, avec les informations détaillées issues de l'ISBN

On pourra pour cet exercice, renvoyer une table HTML, ou toute autre forme lisible, avec le titre, le code ISBN, et séparément, en utilisant la librairie créée, le code domaine, le code éditeur, le numéro de publication et le code de contrôle. Si on avait eu un référentiel des éditeurs, on aurait pu l'utiliser pour afficher le nom de l'éditeur plutôt que son code.

Correction

```
declare namespace li = "com:oxiane:formation:xml:livres";
declare namespace output = "http://www.w3.org/2010/xslt-xquery-
serialization";
import module namespace isbn = "com:oxiane:formation:xq:isbn" at
"isbn.xqm";
declare option output:indent "yes";

<livres>{
for $li in doc("../TP3/livres.xml")/li:livres/li:livre
return
  <livre>
    <titre>{$li/li:titre/data()}</titre>
    <isbn>{$li/li:isbn/data()}</isbn>
    <domaine>{$li/li:isbn=>isbn:getDomain()}</domaine>
    <editeur>{$li/li:isbn/data()=>isbn:getEditor()}</editeur>

    <publication>{$li/li:isbn/data()=>isbn:getPublication()}</publicatio
n>
    <controle>{$li/li:isbn/data()=>isbn:getControl()}</controle>
  </livre>
}</livres>
```


TP 8 - Collections

En fonction du client qu'on utilise, la fonction **collection** ne fonctionne pas de la même façon. En effet, dans la spécification XPath 3.1, il n'est pas précisé quels nœuds sont renvoyés lors de l'évaluation de l'URI passée en paramètre. <https://www.w3.org/TR/xpath-functions-30/#func-collection>

Saxon définit une syntaxe d'URI permettant de rechercher des fichiers dans des répertoires.

TP 8.1 : noms des éléments racine

- > Ecrire une expression permettant d'afficher les noms des éléments racine de tous les fichiers XML trouvés dans le répertoire d'init des TPs. Cela doit concerner tous les fichiers XML, y compris dans les sous répertoire.
- > On consultera la documentation saxon, et on prêtera attention aux paramètres **recurse** et **select**.
- > <http://saxonica.com/documentation9.9/index.html#!sourcedocs/collections>

Corrections

```
xquery version "3.1";

let $col := collection("file:///Users/cmarchand/devel/formation/AE-DBX/src/main/tp-resources?recurse=yes&select=*.xml") /*
return
  if(empty($col)) then 'Aucun fichier trouvé'
  else
    for $n in $col
    return name($n)
```

TP 9 - XQuery Update Facility

Pas de TP, on n'utilisera pas BaseX

TP 10 - Exist-DB XQuery Update Extension

Pour ce TP, nous allons utiliser eXist-db. Télécharger eXist-db depuis <https://bintray.com/existdb/releases/exist/5.2.0/view> et choisir une version adaptée à votre environnement. Etant donné qu'on installe ici une version pour une formation, et pas un serveur de production, un .tar.gz ou un .zip est le plus adapté. Exist-DB nécessite d'avoir Java 8 installé, et correctement configuré. Avoir un JDK est mieux, mais pas indispensable.

Nous noterons ici **\$EXIST_HOME** le répertoire dans lequel est décompressé eXist. Sur la machine du formateur, **\$EXIST_HOME** vaut **~/applications/exist-distribution-5.1.2**.

Lancer eXist-db, avec **\$EXIST_HOME/bin/startup.sh** ou **\$EXIST_HOME\bin\startup.bat**. Ouvrir un navigateur, et aller sur <http://localhost:8080/exist>. Depuis le Dashboard, aller dans Exide, qui est l'IDE d'eXist-db. Un environnement de travail s'ouvre, avec un éditeur XQuery.

TP 10.1 : Créer une nouvelle collection, import de fichiers

Les collections définissent l'organisation métier des données. Dans eXist-db, il y a un concept complémentaire, qui est l'application.

Une application regroupe en général des données, et du code. eXist-db propose d'organiser les collections par applications. Certaines données peuvent être utilisées par plusieurs applications.

- > Dans eXide, Aller dans le menu File / Manage. Si on vous demande de vous authentifier, utiliser **admin** comme login, et laisser le mot de passe vide.
- > Le Database Manager s'ouvre, il permet de manipuler les collections. Cliquer sur le bouton "Dossier" pour créer une nouvelle collection, et saisir **data/shared**.
- > Double-cliquer sur la ligne data (pas dans la colonne de gauche), puis sur la ligne shared..
- > Cliquer sur le bouton Upload (le nuage), puis sur le bouton Upload Files. Sélectionner le fichier **library.zip** dans le répertoire **TP9** des inits de TP.
- > Dans l'éditeur XQuery, saisir l'expression suivante, et l'exécuter (**Ctrl+ENTER**)

```
xquery version "3.1";

let $collection-uri := "/db/data/shared/"
let $zip := util:binary-doc($collection-uri || "library.zip")

return compression:unzip($zip, compression:no-filter#3, (),
function($path, $dt, $px){$collection-uri || $path}, ())
```

Si il y a une erreur lors de l'exécution, vérifier dans l'onglet directory l'emplacement du fichier **library.zip** et ajuster la déclaration de la variable **\$zip**.

Ceci fait, dans la partie gauche de eXide, cliquer sur l'onglet **directory**, déplier **db/data/shared** et vérifier qu'il y a bien les deux fichiers XML, ainsi que les grammaires associées.

TP 10.2 Afficher le contenu d'un fichier

Dans l'éditeur XQuery, écrire une requête qui permet d'afficher le contenu du fichier **livres.xml**.

Correction

```
xquery version "3.1";  
  
doc('/db/data/shared/livres.xml')/*
```

TP 10.3 Corriger les ISBN sur 10 pour les passer sur 13

Afin d'uniformiser tous les ISBN et les avoir tous au format ISBN 13, écrire une expression qui viendra modifier tous les ISBN 10 pour les transformer en ISBN 13. Utiliser pour ce faire **update value**. On pourra réutiliser les fonctions créées précédemment.

- > Commencer par écrire une expression qui affiche toutes les balises ISBN 10.
- > Puis enrichir cette expression pour afficher l'ancienne version et la nouvelle
- > Enfin, écrire l'expression de mise à jour.
- > Réafficher le fichier pour vérifier que tous les ISBN sont bien maintenant au format ISBN 13

Corrections

```
xquery version "3.1";  
  
declare namespace li="com:oxiane:formation:xml:livres";  
declare namespace str="com:oxiane:formation:xq:string";  
declare namespace local="com:oxiane:formation:xq:livres";  
  
declare function str:digitLength($s as xs:string) as xs:integer {  
    $s=>replace('[^0-9]', '')=>string-length()  
};  
  
doc('/db/data/shared/livres.xml')//li:isbn[str:digitLength(.) eq 10]  
  
xquery version "3.1";  
  
declare namespace li="com:oxiane:formation:xml:livres";  
declare namespace str="com:oxiane:formation:xq:string";  
declare namespace local="com:oxiane:formation:xq:livres";  
  
declare function str:digitLength($s as xs:string) as xs:integer {  
    $s=>replace('[^0-9]', '')=>string-length()  
};  
  
for $i in  
doc('/db/data/shared/livres.xml')//li:isbn[str:digitLength(.) eq 10]  
return (  
    $i,  
    <li:isbn>{concat('978-', data($i))}</li:isbn>)  
  
xquery version "3.1";  
  
declare namespace li="com:oxiane:formation:xml:livres";  
declare namespace str="com:oxiane:formation:xq:string";  
declare namespace local="com:oxiane:formation:xq:livres";
```

```
declare function str:digitLength($s as xs:string) as xs:integer {  
    $s=>replace('[^0-9]', '')=>string-length()  
};  
  
for $i in  
doc('/db/data/shared/livres.xml')//li:isbn[str:digitLength(.) eq 10]  
return update value $i with concat('978-', data($i))
```

TP 11 - Application Web avec Exist-DB

eXist-db propose plusieurs moyens pour permettre de requêter au travers de http :

- > la possibilité d'enregistrer des expressions XQuery et de les appeler au travers d'URL
- > la mise en place d'applications complètes, avec templating HTML, et intégration des résultats de requêtes
- > RestXQ

La première méthode est la plus simple, mais il est complexe d'y passer des paramètres. Pas que ce soit impossible, mais il faut utiliser des fonctions du module HTTP, et donc connaître assez bien HTTP.

La deuxième est la plus complète, elle permet de mettre en place une application complète, et la possibilité de configurer finement la façon dont les données sont indexées.

La dernière, RestXQ, permet de créer des services Rest de façon concise, mais ne propose pas de moyen de réaliser une application complète. Il faut ensuite écrire une application capable d'exploiter ces services Rest exposés. Malheureusement, la mise en place de RestXQ est complexe, alors que l'écriture des services est simple ; probablement une volonté de pousser à utiliser la solution "maison"...

Dans ce TP, nous allons étudier les deux premières méthodes pour proposer un frontal sur notre librairie.

TP 11.1 Utilisation de requête stockée sur le serveur

Cliquer sur le bouton **New XQuery** de la toolbar.

Dans l'éditeur XQuery, créer une requête qui renvoie tous les auteurs, sous forme de liste. Cette requête doit renvoyer une page HTML correcte. L'idée est d'obtenir quelque chose comme cela :

```
<html>
  <head>
    <title>Auteurs de la librairie</title>
  </head>
  <body>
    <h1>Nos auteurs</h1>
    <ul>
      <li>Marek Halter</li>
      <li>Pierre Boulle</li>
      <li>Platon</li>
      ...
    </ul>
  </body>
</html>
```

Cliquer sur le bouton **Save** pour enregistrer cette requête dans **/db/apps/auteurs.xq**.

Save Document As ...

↺

📁

🗑️

ℹ️

/db/apps/book-collection/

Name	Permissions	Owner	Group	Last Modified
..				
modules	crwxrwxr-x	admin	dba	03/26/2020 18:29:58
resources	crwxrwxr-x	admin	dba	03/26/2020 18:29:58
templates	crwxrwxr-x	admin	dba	03/26/2020 18:29:58
build.xml	-rw-r--r--	admin	dba	03/26/2020 18:29:59
collection.xconf	-rw-rw-r--	admin	dba	03/26/2020 18:29:59
controller.xql	-rwxrwxr-x	admin	dba	03/26/2020 18:29:59
error-page.html	-rw-rw-r--	admin	dba	03/26/2020 18:29:59
expath-pkg.xml	-rw-rw-r--	admin	dba	03/26/2020 18:29:59
index.html	-rw-rw-r--	admin	dba	03/26/2020 18:29:59
pre-install.xql	-rwxrwxr-x	admin	dba	03/26/2020 18:29:59

Name:

auteurs.xq

Cancel

Save

Cette requête est maintenant enregistrée dans eXist, et peut être utilisée.

Dans le navigateur, ouvrir un nouvel onglet et saisir l'adresse suivante :

<http://localhost:8080/exist/rest/db/apps/auteurs.xq>. Vous devriez voir apparaître la liste des auteurs.

Si la page affiche le HTML comme du XML, c'est que la définition du format de sortie n'est pas bonne. Il faut déclarer que cette requête produit du HTML5. Cela se fait en déclarant des options de sérialisation :

- > **output:method "html5"**
- > **output:media-type "text/html"**
- > Ces deux options sont dans l'espace de nommage **<http://www.w3.org/2010/xslt-xquery-serialization>**

Correction

```
xquery version "3.1";

declare namespace li="com:oxiane:formation:xml:livres";
declare namespace at="com:oxiane:formation:xml:auteurs";
```

```
declare namespace output="http://www.w3.org/2010/xslt-xquery-serialization";

declare option output:method "html5";
declare option output:media-type "text/html";

<html>
  <head>
    <title>Librairie Oxiane</title>
  </head>
  <body>
    <h1>Nos auteurs référencés</h1>
    <ul>
      {
        for $a in collection('/db/data/shared')//at:auteur
        return <li>{string-join(($a/at:prenom!data(), $a/at:nom!data()), '
')}</li>
      }
    </ul>
  </body>
</html>
```

TP 11.2 Application Web dans eXist-db

Dans le menu, cliquer sur **Application/New application**. Puis remplir le formulaire comme suit (vous pouvez changer le nom de l'application mais ce doit être une URI ; il est cependant recommandé, pour la clarté des exercices suivants de conserver les mêmes valeurs) :

Deployment Editor



Package collection: /db/apps//

Application Properties

Template: eXist-db Design

Type of the package: Application

Target collection: book-collection

Name: http://oxiane.com/apps/book-collection

Abbreviation: book-collection

Title: Book Collection|Web app

Version: 0.1

Status: Alpha

Pre-install XQuery: pre-install.xql

Post-install XQuery: post-install.xql

Back

Next

Cancel

Done

Plier / déplier l'arbre de la vue **directory**, retrouver dans **apps/book-collection** le fichier **index.html**. L'ouvrir et regarder son contenu.

A la ligne 6, on voit un lien `<a data-template="templates:load-source" href="index.html">...`. Ce lien pointe vers **index.html** et l'attribut indique qu'il faut utiliser la fonction XQuery **templates:load-source**.

A la ligne 17, il y a un `<div data-template="app:test"/>`. Ce div sera remplacé par le mécanisme de tempating par ce que renvoie la fonction **app:test**. Cette fonction est définie dans le fichier **modules/app.xql**. Ouvrir ce fichier et regarder ce que renvoie la fonction.

Ouvrir un nouvel onglet de navigateur à l'adresse <http://localhost:8080/exist/apps/tutorial/index.html> ; c'est l'application. Comparer le lien **This** qui permet de recharger la page, et le texte renvoyé par la fonction.

On a donc moyen, depuis une page HTML, d'appeler directement des fonctions XQuery.

Réaliser les actions suivantes :

- > Dans le fichier **modules/app.xql**, créer une nouvelle fonction **app:auteurs** qui renvoie un **** contenant la liste des auteurs. **Attention** : toutes les fonctions appelées par le templating HTML prennent au minimum deux paramètres, **\$node as node()**, **\$model as map(*)**. Elles peuvent prendre d'autres paramètres qui seront passés par le contexte de la requête HTTP.
- > Modifier la page **index.html**, supprimer le **div** avec la class **alert alert-success**. A l'intérieur du premier **div** ayant pour class **col-md-6**, remplacer le **p** par **<div data-template="app:auteurs"/>**.
- > Enregistrer, et vérifier que la page d'accueil de l'application affiche bien la liste des auteurs.

Correction

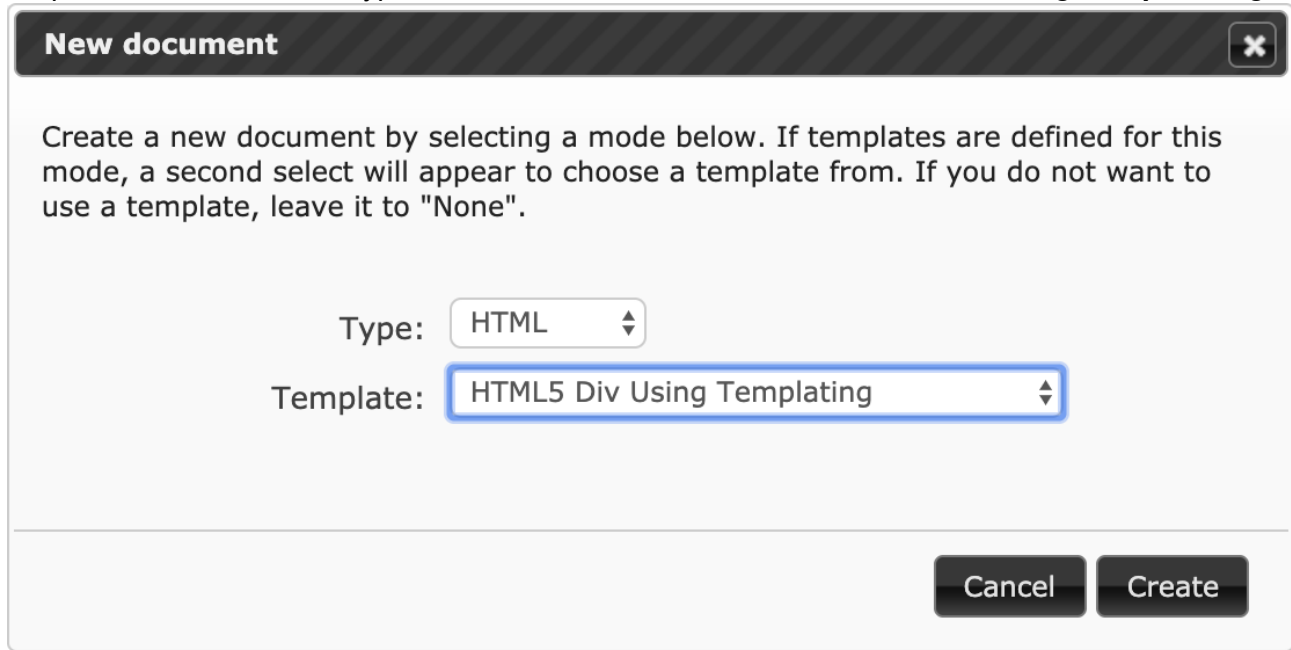
```
index.html
-----
<div xmlns="http://www.w3.org/1999/xhtml" data-
template="templates:surround" data-template-
with="templates/page.html" data-template-at="content">
  <div class="col-md-9">
    <h1 data-template="config:app-title">Generated page</h1>
    <div class="row">
      <div class="col-md-6">
        <h2>Naviguer parmi nos auteurs</h2>
        <div data-template="app:auteurs"/>
      </div>
    </div>
  </div>
  <div class="col-md-3">
    <h2>Application Info</h2>
    <div data-template="config:app-info"/>
  </div>
</div>

app.xql
-----
declare function app:auteurs($node as node(), $model as map(*)) {
  <ul>
  {
    for $a in collection('/db/data/shared')//at:auteur
    return <li><a data-template="templates:load-source"
href="livres.html?auteurId={$a/at:id/data()}">{string-
join(({$a/at:prenom!data()}, $a/at:nom!data()), ' ')}</a></li>
  }
  </ul>
};
```

Nous allons maintenant créer une page permettant d'afficher tous les ouvrages de l'auteur spécifié, et le résumé de chacun de ses ouvrages. Si l'auteur n'a pas d'ouvrage, on affichera un message indiquant qu'il n'a pas d'ouvrage. Si un ouvrage n'a pas de résumé, on affichera un message indiquant qu'il n'y a pas de résumé pour cet ouvrage.

Créer dans **modules/app.xql** une nouvelle fonction **app:nomAuteur** ayant 3 paramètres, les deux obligatoires, et un **\$auteurId** de type **xs:string**.

Cliquer sur New, choisir le type HTML et sélectionner ensuite **HTML5 Div using Templating**.



New document

Create a new document by selecting a mode below. If templates are defined for this mode, a second select will appear to choose a template from. If you do not want to use a template, leave it to "None".

Type: HTML

Template: HTML5 Div Using Templating

Cancel Create

Enregistrer cette nouvelle page sous **livres.html** à la racine de l'application (dans **/db/apps/book-collection/**). Ajouter un titre **h1** :

```
<h1>Livres de <span data-template="app:nomAuteur"/></h1>
```

Enregistrer. Revenir dans la fonction **app:auteurs**, et ajouter un lien vers la page **livres.html** permettant d'afficher les livres de l'auteur :

```
<a data-template="templates:load-source" href="livres.html?
auteurId={$a/at:id/data()}">{string-join(({$a/at:prenom!data(),
$a/at:nom!data()), ' ')}</a>
```

Enregistrer. Retourner dans l'application Web, rafraichir, constater qu'on a maintenant pour chaque auteur un lien, et vérifier que quand on clique sur le lien, on affiche bien une page avec le nom de l'auteur.

Correction

```
livres.html
-----
<div xmlns="http://www.w3.org/1999/xhtml" data-
template="templates:surround"
data-template-with="templates/page.html" data-template-
at="content">
```

```

    <h1>Livres de <span data-template="app:nomAuteur"/></h1>
    <div data-template="app:livres"/>
</div>

app.xql
-----
declare function app:auteurs($node as node(), $model as map(*)) {
    <ul>
    {
        for $a in collection('/db/data/shared')//at:auteur
        return <li><a data-template="templates:load-source"
href="livres.html?auteurId={$a/at:id/data()}">{string-
join(($a/at:prenom!data(), $a/at:nom!data()), ' ')}</a></li>
    }
    </ul>
};

declare function app:nomAuteur($node as node(), $model as map(*),
$auteurId as xs:string) {
    collection('/db/data/shared')//at:auteur[at:id eq
$auteurId]/string-join((at:prenom/data(), at:nom/data()), ' ')
};

```

Enfin, dernière étape, créer dans **app.xql** une nouvelle fonction qui affiche les livres de l'auteur, et leur résumé. Ajouter dans la page **livres.html** un div utilisant cette fonction.

Correction

```

app.xql
-----
declare function app:livres($node as node(), $model as map(*),
$auteurId as xs:string) {
    let $livres :=
collection('/db/data/shared')//li:livre[li:auteurs/li:auteur/@refid
eq $auteurId]
    return if($livres) then
        for $li in $livres
        return (
            <h2>{$li/li:titre/data()}</h2>
            ,
            if ($li/li:resume) then
                (<h3>Résumé</h3>,
                for $p in $li/li:resume/*
                return <p>{$p/data()}</p>)
            else <p>Pas de résumé pour ce titre</p>
        )
    else
        <p>Pas de livre pour cet auteur</p>
};

livres.html
-----
declare function app:livres($node as node(), $model as map(*),
$auteurId as xs:string) {

```

```
let $livres :=  
collection('/db/data/shared')//li:livre[li:auteurs/li:auteur/@refid  
eq $auteurId]  
return if($livres) then  
  for $li in $livres  
  return (  
    <h2>{$li/li:titre/data()}</h2>  
    ,  
    if ($li/li:resume) then  
      (<h3>Résumé</h3>,  
      for $p in $li/li:resume/*  
      return <p>{$p/data()}</p>)  
    else <p>Pas de résumé pour ce titre</p>  
  )  
else  
  <p>Pas de livre pour cet auteur</p>  
};
```

eXist-db propose plusieurs moyens de restituer les données stockées dans la base. Il faut choisir entre ces différentes méthodes en fonction du type d'application qu'on veut réaliser. Si le type d'application que vous souhaitez réaliser n'est pas une application Web, mais plutôt une application de traitement en Java, il est possible d'utiliser le client Java au travers de l'API XQJ (XQuery for Java), ou d'appeler les XQuery stockées dans eXist-db, au travers d'appels REST. De nombreuses possibilités sont ouvertes.

eXist-db a cependant plusieurs lacunes qui le limite plutôt à des usages de publication. eXist-DB n'est pas transactionnel, et ne devra pas être utilisée pour des applications où les modifications sont prépondérantes, comme un éditeur de documents, par exemple.