XQUERY TIPS & TRICKS

YOU'VE GOT QUESTIONS... WE'VE GOT ANSWERS

Welcome to XQuery Tips and Tricks. We've answered a lot of questions about how you can get the most bang for your XQuery buck by using inventive XQuery coding solutions and, of course, the power and added functionality available from the DataDirect XQuery processor.

TABLE OF CONTENTS

WHY DOES MY RELATIVE PATH FAIL EVEN IF I SET THE BASE-URI PROPERTY?

My XQuery looks like this:

```
declare base-uri "file:///C:/mydocuments/xml";
doc("items.xml")//ITEM[@id='145']
```

My *items.xml* file is indeed in my *c:\mydocuments\xml* folder, as shown in the base-uri, but when I run the XQuery, I get this error:

```
[DataDirect][XQuery][err:FODC0005]java.io.FileNotFoundException:
C:\mydocuments\items.xml (The system cannot find the file specified)
```

Why is the XQuery processor looking for *items.xml* in the *c:\mydocuments* folder instead of *c:\mydocuments\xml* as specified in the base-uri property?

As strange as it sounds, the XQuery processor is adhering to what the specifications dictate. Take a look at RFC 2396 - Uniform Resource Identifier (URI): Generic Syntax. Section 5.2. Resolving Relative References to Absolute Form, subsection 6a contains the following:

All but the last segment of the base URI's path component is copied to the buffer. In other words, any characters after the last (right-most) slash character, if any, are excluded.

You can fix your XQuery quite easily, by adding a final slash to the declared base-uri:

```
declare base-uri "file:///C:/mydocuments/xml/";
doc("items.xml")//ITEM[@id='145']
```

IS THERE A WAY TO NUMBER RECORDS EXTRACTED FROM A DATABASE?

I need to extract data from my relational database as XML, and I need to assign a number to each record that I extract. I wrote the following XQuery...

```
<myRecords> {
   let $i := 0
   for $Territories in collection("Northwind.dbo.Territories")/Territories
   let $i := $i + 1
   return
      <territory rec_count="{$i}">
         {$Territories/TerritoryDescription/text()}
      </territory>
 } </myRecords>
```

... I don't get the result I expected:

```
<myRecords>
   <territory rec_count="1">Westboro</territory>
```

```
      <territory rec_count="1">Bedford</territory>
      <territory rec_count="1">Georgetown</territory>
...
```

The record number isn't incremented. Why?

XQuery is a functional language without side effects — you can't really create a notion of state in XQuery as you might in Java or C#, for example. In some cases, you can simulate state using a recursive function, but to solve this problem there is a much simpler solution: you can use a *positional variable*. Here's the XQuery:

```
<myRecords> {
    for $Territories at $i in collection("Northwind.dbo.Territories")/Territories
    return
        <territory rec_count="{$i}">
            {$Territories/TerritoryDescription/text()}
        </territory>
 } </myRecords>
```

And here's the (hoped-for) result:

```
<myRecords>
    <territory rec_count="1">Westboro</territory>
    <territory rec_count="2">Bedford</territory>
    <territory rec_count="3">Georgetow</territory>
```

CAN I USE FN:COLLECTION() TO RETRIEVE THE URL OF A DOCUMENT IN A FOLDER?

Suppose I have this XQuery:

```
<documents> {
    for $doc in fn:collection("file:///c:/xml?select=*.xml")
    return
        <doument URL="theDocURL">
            {$doc}
        </doument>
 } </documents>
```

How do I assign the proper value to the URL= attribute?

You can assign values to the URL= attribute using the fn:document-uri function, shown here:

```
<documents> {
    for $doc in fn:collection("file:///c:/xml?select=*.xml")
    return
        <doument URL="{$doc/document-uri($doc)}">
            {$doc}
        </doument>
 } </documents>
```

WHAT'S THE EASIEST WAY TO EXTRACT THE FILE NAME FROM A URL?

Probably like this:

```
tokenize($uri, "/")[last()]
```

And if you want to replace the extension of the file name with something different, you could use this simple function:

```
declare function local:generate-file-name($uri as xs:string, $ext as xs:string) {
  let $fileName := tokenize($uri, "/")[last()]
  let $fileName := replace($fileName,'^(.*)\..*','$1')
  let $fileName := concat($fileName, ".", $ext)
  return $fileName
 };
```

CAN I INSTRUCT XQUERY TO NOT USE THE MINIMIZED XML TAG FORMAT?

My XQuery is this:

```
<root><tag></tag></root>
```

And the result is this:

```
<root><tag/></root>
```

How can I force the XQuery processor to stop using the minimized tag format for empty elements?

From an XML InfoSet point of view, there is no difference between <tag/> and <tag></ tag >, which is why the serialization specifications for XQuery 1.0 and XSLT 2.0 don't say anything about minimized tags when generating XML.

The only case in which the serialization specifications address the minimized XML tag format is when XHTML is used as the output method (to maintain compatibility with HTML browser behavior). Using DataDirect XQuery, you can change your XQuery as follows to avoid minimized XML tags:

```
declare option ddtek:serialize "method=xhtml";
<root><tag></tag></root>
```

The XQuery processor will serialize the result of that XQuery as follows:

```
<root><tag></tag></root>
```

Switching to the XHTML output method is a viable work-around when using any compliant XQuery processor. That said, using the XHTML output method can cause other, undesired side effects, so

DataDirect XQuery allows you to specifically turn off the minimized tag format in result serialization using the propietary "empty-element-tags" option:

```
declare option ddtek:serialize "method=xml,{http://www.datadirect.com/xquery}empty-
element-tags=no";
<root><tag></tag></root>
```

The result in this case is what you would expect:

```
<root><tag></tag></root>
```

HOW CAN I LOAD XML DATA INTO MY RELATIONAL DATABASE?

What you are looking for is the ability of an XQuery implementation to handle updates to a relational database. DataDirect XQuery supports relational updates, and you can find a detailed discussion of this functionality here.

Because DataDirect XQuery is an easily embeddable Java component, it's correct to infer that you can easily add the ability to load XML data into your relational database from your Java application, or even to expose it as a Web service on the application server of your choice.

CAN I GROUP ELEMENTS BASED ON THEIR RELATIVE POSITIONS?

My XML is structured this way:

```
<books>
   <title>title 1</title>
   <price>100</price>
   <date>01012007</date>
   <title>title 2</title>
   <price>90</price>
   <publisher>pub 1</publisher>
 </books>
```

The XML file contains a long list of books, about which only the <title> element is guaranteed to be always present.

How can I modify the XML document using XQuery to group each book's properties under a single element, like this:

```
<books>
   <book>
      <title>title 1</title>
      <price>100</price>
      <date>01012007</date>
   </book>
   <book>
      <title>title 2</title>
```

```
                <price>90</price>
                <publisher>pub 1</publisher>
            </book>
        </books>
```

The simplest solution we can think of is this:

```
    <books> {
        for $title in /books/title
        return
            <book>  {
                $title,
                let $nextTitle := $title/following-sibling::title[1]
                for $prop in $title/following-sibling::*[local-name() != "title"]
                where empty($nextTitle) or $prop << $nextTitle
                return $prop
            } </book>
    } </books>
```

In this solution, we basically iterate over all the <title> elements. For each of them we find the next <title> element (if any), and then we iterate through all the elements contained in between the two <title> elements.

The code can become a bit easier to read and more resuable if we introduce a function, like this:

```
    declare function local:getRelatedSiblings($item) {
        let $nextItem := $item/following-sibling::*[local-name()=$item/local-name()][1]
        for $related in $item/following-sibling::*[local-name()!=$item/local-name()]
        where empty($nextItem) or $related << $nextItem
        return $related
     };

     <books> {
        for $title in /books/title
        return
            <book>  {
                $title,
                for $prop in local:getRelatedSiblings($title)
                return $prop
            } </book>
    } </books>
```

WHAT'S THE PROPER WAY TO USE THE NODE SEQUENCE (<>) OPERATORS?

This is my XQuery:

```
let $referenceBook := /books/book[2]
        let $book := /books/book
        where $book/title << $referenceBook
        return $book
```

This looks ok to me, but when I run it, I get the following error. Can you tell me why?

[DataDirect][XQuery][err:XPTY0004]A sequence of more than one item is not allowed as the first operand of '<<' (<title/>, <title/>, ...)

The "node before" (<<) and "node after" (>>) operators can only compare the document position of a single node to the position of another node. In your case, the let $book := /books/book instruction is assigning all the <book> elements in the input document to the $book variable, which means that $book/title corresponds to a sequence of many elements.

Perhaps what you mean to do is this:

```
let $referenceBook := /books/book[2]
  for $book in /books/book
  where $book/title << $referenceBook
  return $book
```

CAN I INSTRUCT XQUERY TO USE NUMBERS AND NOT SCIENTIFIC NOTATION IN QUERY RESULTS?

My database has a "Quantity" column defined as type "real". When I run this XQuery:

```
<bigOnes> {
      for $Invoices in collection("Invoices")/Invoices
      where $Invoices/Quantity > 1000
      return $Invoices/Quantity
  } </bigOnes>
I get this result:
<bigOnes>
    <Quantity>1.0E7</Quantity>
</bigOnes>
```

Considering that the "Quantity" column never uses decimals, it doesn't really make sense to display my results this way. How can I instruct the XQuery processor to *not* use scientific notation?

The simplest way is probably to force "Quantity" to be interpreted as an xs:integer type, which you can do like this:

```
<bigOnes> {
      for $Invoices in collection("Invoices")/Invoices
      where $Invoices/Quantity > 1000
      return <Quantity>{xs:integer($Invoices/Quantity)}</Quantity>
  } </bigOnes>
This will get you the result you're looking for:
<bigOnes>
   <Quantity>10000000</Quantity>
</bigOnes>
```

WHY DO I GET AN ERROR TRYING TO EXTRACT VALUES FROM A FUNCTION RETURNING XML?

My XQuery uses a function to access an input XML file:

```
        declare function local:input() {
            "<books><book><title>title  1</title></book><book><title>title 
2</title></book></books>"
         };
          local:input()//book[1]
```

When I try to access the result through an XPath expression, I get this error:

[DataDirect][XQuery][err:XPTY0019]Error at line 5, column 1. It is a type error if the result of a step (other than the last step) in a path expression contains an atomic value.

Can you tell me why?

The local:input() function is really meant to return a string, rather than a real XML fragment. Whether or not that's a mistake depends on what you are trying to do.

If it's *not* a mistake, and you really intend for local:input() to reuturn a string containing an XML fragment, then you need to instruct the XQuery processor to *parse* that string before the XML can be accessed through XPath. You can provide this instruction using the DataDirect XQuery ddtek:parse() function, shown here:

```
        declare function local:input() {
            "<books><book><title>title  1</title></book><book><title>title 
2</title></book></books>"
         };
          ddtek:parse(local:input())//book[1]
```

On the other hand, if the error is that the local:input() function shouldn't be returning a string in the first place — you want a node — the you need to fix the function itself, as shown here:

```
        declare function local:input() {
            <books><book><title>title 1</title></book><book><title>title 2</title></book></books>
         };
          local:input()//book[1]
```

CAN I USE DYNAMIC TEXT AS THE NAME OF A RESULT NODE?

Yes, you can use *computed constructors* (http://www.w3.org/TR/xquery/#id-computedConstructors):

```
        let $r := "some-dynamic-value"
```

```
  return
      element {$r} {'something'}
to return...
<some-dynamic-value>something</some-dynamic-value>
```

HOW CAN I RETURN THE RESULT OF MULTIPLE EXPRESSIONS?

I have the following XQuery:

```
<root>
    {
      for $a in (1 to 10)
      return
          <number>{  $a }</number>
          <number-plus-100>{ $a + 100 }</number-plus-100>
    }
  </root>
```

But the syntax coloring is "broken," and executing this XQuery returns this error:

[DataDirect][XQuery][err:XPST0003]Error at line 6, column 19. Expected "}", but encountered ">".

If you want to return multiple expressions (the <number> and <number-plus-100> elements in this case), you need to return a sequence containing multiple expressions; to do that, you use the comma operator (http://www.w3.org/TR/xquery/#dt-comma-operator). NOTE: the comma operator is the operator with the lowest precedence in XQuery; so writing something like:

```
…
    return
          <number>{  $a }</number>,
          <number-plus-100>{  $a + 100 }</number-plus-100>
```

... will trigger another error:

[DataDirect][XQuery][err:XPST0008] Undeclared variable $a;

Instead you need to change the XQuery into:

```
<root>
    {
      for $a in (1 to 10)
      return (
          <number>{  $a }</number>,
          <number-plus-100>{  $a + 100 }</number-plus-100>
      )
    }
  </root>
```

WHY DOES MY XQUERY FAIL WHEN I OVERRIDE THE DEFAULT NAMESPACE?

The following XQuery:

```
let $doc := <root><book>1</book></root>
  return
      <books> {
          for $book in $doc//book
          return <bookid>{$book/text()}</bookid>
      } </books>
```

... works fine for me and returns:

```
<books><bookid>1</bookid></books>
```

I need to specify that the result fragment is defined in a different namespace; but when I change the XQuery into:

```
let $doc := <root><book>1</book></root>
  return
      <books xmlns="myNamespace">  {
          for $book in $doc//book
          return <bookid>{$book/text()}</bookid>
      } </books>
```

...then no <bookid>'s are returned. Why?

The problem is caused by the fact that in XQuery the selection of a new default namespace (xmlns="myNamespace") also affects the XPath expressions contained in that scope; so, your XQuery is selecting all <book> elements in the “myNamespace” namespace, rather than in the (default) one used to define the input document. This mechanism usually makes life easier for XQuery developers; but in this case it causes some headaches. The cleanest solution is probably to avoid using the default namespace and use a prefixed one instead, like in this example:

```
let $doc := <root><book>1</book></root>
  return
      <ns:books xmlns:ns="myNamespace">  {
          for $book in $doc//book
          return <ns:bookid>{$book/text()}</ns:bookid>
      } </ns:books>
... which returns:
<ns:books xmlns:ns="myNamespace"><ns:bookid>1</ns:bookid></ns:books>
```

CAN I CREATE A LIBRARY OF XQUERY FUNCTIONS?

Yes! In fact, a library of external XQuery functions is a great way to avoid having to redefine functions in the prologs of every query you write.

You can use the “import module” directive in XQuery; for example, suppose you have defined a library of functions like myFunctions.xquery:

```
module namespace utilities = "myUtilities";

 declare function utilities:getName($book) as xs:string {
  …
 };

 declare function utilities:getPublisher($book) as xs:string {
  …
 };
```

You can then use the functions defined in that module in any other XQuery doing:

```
import module namespace utilities = "myUtilities" at "myFunctions.xquery";
 for $aBook in doc("books.xml")//book
 return utilities:getName($aBook)
```

HOW CAN I QUERY ALL DOCUMENTS IN A FOLDER?

You can use the fn:collection() function to query all documents in a folder. DataDirect XQuery™ allows you to specify regular expressions to match file names, provides the possibility to fetch all documents recursively in nested folders, and the possibility to sort the result. The syntax that DataDirect XQuery™ accepts for the fn:collection() function argument when dealing with retrieving files from folders is this:

collection("*directory_url*(?*option*(;*option*)*)?")
where:

> *directory_url* is a URL referencing a directory. The URL must use the file:// scheme.

> *option* is {(select="REGEX") | recurse={yes | no} | (sort=[a,t,r]+) | (xquery-regex=(yes|no))}

In the following example, suppose you have a number of XML files stored in the directory named *books*. Each of the files contains information about one book, and you want to create a single XML document that contains a list of all your books.

```
<books> {
 for $book in collection("file:///c:/books?select=*.xml ")
 return
   <myBook>{$book/book/title}</myBook>
 } </books>
```

The result would look something like this:

```
<books>
  <myBook>
    <title>Emma</title>
  </myBook>
  <myBook>
    <title>Pride and Prejudice</title>
  </myBook>
  . . .
 </books>
```

If the *books* directory contained sub-folders with other XML documents, then you would need to change the XQuery to:

```
<books> {
 for $book in collection("file:///c:/books?select=*.xml;recurse=yes")
 return
   <myBook>{$book/book/title}</myBook>
 } </books>
```

IN WHAT ORDER DOES FN:COLLECTION() RETURN DOCUMENTS?

In DataDirect XQuery™, the order of documents returned by the fn:collection() depends on which function options you specify to fetch documents from folders. If you don't select any "sort" options (collection("file:///c:/books?select=*.xml"), for example), the order in which documents are returned is not guaranteed; if you use any "sort" option, is specified (collection("file:///c:/books?select=*.xml";sort=a), for example), then documents are returned in the requested order.

WHAT IS A CONTEXT ITEM TYPE? AND HOW DO I SET IT?

You might wonder why you get the following error when running a simple XQuery, like Hello, for example, using DataDirect XQuery:

The context item static type has not been set

Most likely what you are trying to do is to write an XQuery that returns the character string "Hello"; but the way that XQuery is written means: return all children elements of the context item whose name matches "Hello". The correct XQuery is:

vHello"

The reported error is actually an XQuery compilation error, not even a runtime one; what's happening here is that DataDirect XQuery™ is not finding information about the type for the context item (it's not complaining about the lack of data yet). If you are using the DataDirect

XQuery command line, no information (type or data) about the context item is specified by it unless you provide the –s option. If you use the DataDirect XQuery XQJ implementation, you can set the context item type by doing the following:

```
 XQStaticContext cntxt = xqConnection.getStaticContext();
cntxt.setContextItemStaticType(xqConnection.createDocumentElementType(xqConnectio
n.createElementType(null, XQItemType.XQBASETYPE_UNTYPED)));
```

... for example (assuming you are planning to bind a generic document to the context item).

If you specify the static type for the context item, but you bind no value for it, you will get a different error trying to run the XQuery described at the beginning of this section:

Error XPDY0002: The context item for axis step child::element(Hello) is undefined;

To both set the static type and bind a document to the context item, you can do:

```
 XQStaticContext cntxt = xqConnection.getStaticContext();
cntxt.setContextItemStaticType(xqConnection.createDocumentElementType(xqConnectio
n.createElementType(null, XQItemType.XQBASETYPE_UNTYPED)));
 xqExpression = xqConnection.createExpression(cntxt);
 xqExpression.bindDocument(XQConstants.CONTEXT_ITEM, new
FileInputStream(vmyDocument.xml"));
```

There are of course different ways to reference the XML document to be bound, including StAX or SAX streams and DOM trees.

CAN I INCLUDE CDATA SECTIONS IN MY XQUERY RESULTS?

I have this XQuery:

```
let $v := "<e/>"
  return <e>{ $v  }</e>
If I run it, I get the following result:
<e>&lt;e/&gt;</e>
```

… but what I really want is a CDATA section as the value of <e> which prevents the need of escaping characters:

```
<e><![CDATA[<e/>]]></e>
```

Both results represent the same XML document; CDATA sections are nothing more than syntactical sugar. Neither the XML Info Set nor the XQuery 1.0 Data Model know about the concept of CDATA sections. So what you have here is an <e> element, with a single text node with content <e>.

Someone might be tempted to solve the problem by changing the XQuery to:

```
let $v := "<e/>"
  return <e>{  fn:concat("<![CDATA[", $v, "]]>") }</e>
… but that won't help much, as the result becomes:
<e>&lt;![CDATA[&lt;e/&gt;]]&gt;</e>
```

DataDirect XQuery™ allows you to control the serialization parameters through either its XQJ API or through the XQuery itself. One of these serialization parameters (cdata-section-elements) allows you to specify that the value of the listed elements must be serialized in CDATA sections, rather than as escaped text. That means I can change the XQuery from which we started to this:

```
declare option ddtek:serialize "cdata-section-elements=e";
  let $v := "<e/>"
  return <e>{ $v  }</e>
… and I get the output in the format I want:
<e><![CDATA[<e/>]]></e>
```

More in general, the parameter value of cdata-section-elements is a list of qnames, separated by semi-colons; namespaces can be specified using the well known compact notation by James Clark, "{"+namespace uri+"}"localname.

For example, this XQuery:

```
declare namespace p1 = "uri1";
  declare namespace p2 = "uri2";
  declare namespace p3 = "uri3";
  declare option ddtek:serialize "indent=yes,cdata-section-elements={uri1}e;e;{uri2}e";
  <result>
    <e>aaa</e>
    <p1:e>bbb</p1:e>
    <p2:e>ccc</p2:e>
    <p3:e>ddd</p3:e>
  </result>
```

…generates this output:

```
<result>
    <e><![CDATA[aaa]]></e>
    <p1:e xmlns:p1="uri1"><![CDATA[bbb]]></p1:e>
    <p2:e xmlns:p2="uri2"><![CDATA[ccc]]></p2:e>
    <p3:e xmlns:p3="uri3">ddd</p3:e>
  </result>
```

WHY DO I GET THE WRONG RESULTS LOOKING FOR THE MY BOOK'S FIRST AUTHOR?

My XML document is structured like this:

```
<root>
   <book>
      <info>
         <author>a1</author>
      </info>
      <info>
         <author>a2</author>
      </info>
   </book>
 </root>
```

… and I'm trying to select only the first author of the book. But when I run this XQuery:

```
        let $doc :=
<root><book><info><author>a1</author></info><info><author>a2</author></info></book></root>
         return
            $doc//book//author[1]
```

… I get back all the authors.

The problem here is that [1] (equivalent to [position()=1]) applies to the <author> position, which is always in first position as child of the <info> element. The expanded XPath syntax corresponding to the abbreviated one used in the query is:

```
$doc/descendant-or-self::node()
   /child::book
      /descendant-or-self::node()
         /child::author[position()=1]
```

To correct the problem, I can use the descendant:: axis:

```
        let $doc :=
<root><book><info><author>a1</author></info><info><author>a2</author></info></book></root>
        return
           $doc//book/descendant::author[1]
```

CAN I USE VARIABLES IN MY XPATH EXPRESSIONS?

If we run the following XQuery:

```
let $xml := <root><book>book1</book><video>video1</video></root>
 for $el in ("book", "video")
```

```
return $xml/$el/text()
```

... we get this error:

It is a type error if the result of a step (other than the last step) in a path expression contains an atomic value.

The problem is that $el is being inserted in the XPath expression as a string literal, not as the name of an element; it as if you tried to run an XPath expression like $xml/"book"/text().

What you typically are trying to do is:

```
let $xml := <root><book>book1</book><video>video1</video></root>
  for $el in ("book", "video")
  return $xml/*[local-name() eq $el]/text()
```

HOW DO I CHOOSE THE RIGHT COMPARISON OPERATOR?

I have this XQuery, which works as I would expect:

```
let $doc :=
      <books>
         <book>
            <title>book1</title>
            <authors>
               <author>author1</author>
            </authors>
         </book>
      </books>
  return
      $doc//book[.//author eq "author1"]
```

… in that it returns the "book1" book authored by "author1"; but if I changed the XQuery to this…

```
let $doc :=
      <books>
         <book>
            <title>book1</title>
            <authors>
               <author>author1</author>
            </authors>
         </book>
         <book>
            <title>book2</title>
            <authors>
               <author>author1</author>
```

```
                <author>author2</author>
            </authors>
        </book>
    </books>
return
    $doc//book[.//author eq "author1"]
```

… then I get a type error, like the following:

A sequence of more than one item is not allowed as the first operand of 'eq'"

The problem is with the use of the comparison operator "eq"; XQuery has a variety of comparison operators: value, general, node and order. It is easy to miss the difference between **value** (eq, ne, lt, le, gt, ge) and **general** (=, !=, <, <=, >, >=) comparison operators.

> **Value** comparison operators (new in XPath 2.0) can only compare two single atomic values, where a single atomic value can be a string, a number, a date or also a single node that contains a single atomic value, or also the empty sequence. So, 5 gt 4 = false; "first" ne "second" = true; <el>4</el> lt <el>6</el> = true; but (2,3) eq 2 or (2,3) eq (2,3) both trigger type errors. It's also worth noting that if $a eq $b is true, the $b eq $a is true too; the same assumption is not correct using general comparison operators.

> **General** comparison operators (introduced originally in XPath 1.0) too work against atomic type operands, but instead of requiring each operand to be single atomic values, they also work against *sequences* of atomic values. The general comparison returns true if any value on the left matches any value on the right, using the appropriate comparison.

There are other more subtle differences about value and general comparison operators, especially when dealing with untyped data; The XPath 2.0 specifications describe those differences in full details.

From a performance point of view, it is usually good practice to use value comparison operators when you know that you are comparing exactly two items.

Going back to the example above, if what the query is trying to find are all the books for which "author1" is listed as an author, then you can just change it to use a general comparison operator:

```
let $doc :=
    <books>
        <book>
            <title>book1</title>
            <authors>
                <author>author1</author>
            </authors>
        </book>
        <book>
            <title>book2</title>
            <authors>
                <author>author1</author>
                <author>author2</author>
            </authors>
        </book>
    </books>
return
    $doc//book[.//author = "author1"]
```

CAN I GROUP RESULTS BASED ON VALUES?

I have a list of books in XML, and I would like to generate a different XML structure that groups all books per subject; this is my source XML:

```
<books>
    <book bookid="1">
        <title>Java Web Services</title>
        <subject>XML</subject>
    </book>
    <book bookid="2">
        <title>XML Applications</title>
        <subject>XML</subject>
    </book>
    <book bookid="4">
        <title>Beginning Visual C++ 6  Database Programming</title>
        <subject>Database</subject>
    </book>
    <book bookid="5">
        <title>Beginner&apos;s  Guide to Access 2.0</title>
        <subject>Database</subject>
    </book>
 </books>
```

I know in XSLT 2.0 has a way to do that easily using xsl:for-each-group, like this:

```
<subjects>
    <xsl:for-each-group select="/books/book" group-by="subject">
        <subject name="{subject}">
            <xsl:for-each select="current-group()">
                <book-title><xsl:value-of select="title"/></book-title>
            </xsl:for-each>
        </subject>
    </xsl:for-each-group>
 </subjects>
```

How can I do the same using XQuery 1.0?

Even if XQuery 1.0 doesn't have the explicit language support for value-based grouping that XSLT 2.0 has, you can easily implement grouping using the fn:distinct-values() function and the other XQuery language structures. Your example can be re-written in XQuery this way:

```
<subjects> {
   for $subject in distinct-values(/books/book/subject)
   let $books-in-group := /books/book[subject=$subject]
   return
      <subject name="{$subject}"> {
         for $book in $books-in-group
         return
            <book-title>{$book/title/text()}</book-title>
      } </subject>
 } </subjects>
```

See the XQuery W3C specifications for more details on this and other grouping functions.

WHEN SHOULD I USE /TEXT() AT THE END OF MY XPATH EXPRESSION?

You have probably seen a lot of examples that use /text() at the end of XPath expressions, and you might be wondering, *Should I do the same? Are there any drawbacks?*

There are cases in which you do want to append /text() to your XPath expression to force your XQuery to consider the text value of your XPath expression; the typical case is when you want to dynamically create the value of a result element; something like this:

```
<title>{$doc//book[1]/book-title}</title>
… generates this output:
 <title>
    <book-title>La Divina  Commendia</book-title>
 </title>
```

However, if what you are trying to create is actually this:

```
<title>La Divina Commendia</title>
```

… then you need to instruct XQuery to use the text value of the <book-title> element, and an easy way to do that is appending /text() to the XPath expression:

```
<title>{$doc//book[1]/book-title/text()}</title>
```

But often XQuery developers use /text() too frequently, in cases similar to this, for example:

```
for $book in $doc//book[title/text() = "La Divina Commendia"]
return …
```

What are the drawbacks of using /text() when it is not needed? It turns out that there are several:

If the data is typed, the type information is thrown away. Beside the possibly unnecessary type conversions, this also makes the query less type safe. This can also cause significant performance disadvantages when dealing with relational data sources.

It causes the query to fail — or even produce wrong results — if the element contains comment nodes. There are cases in which this is not true, but in general queries should be written so they aren't affected by comments nodes. Consider the following XQuery, for example:

```
let $doc := <root><val>foo<!-- comment -->bar</val></root>
    return
       ($doc//val = "foobar",
        $doc//val/text() = "foobar")
```

It returns (true, false) because the result of $doc//val/text() is actually the sequence (“foo”, “bar”) and not “foobar” as you would expect.

It makes your queries more verbose than needed.