# Chapter 9: XQuery, the XML Query Language



## Chapter 9 Objectives

- Why XQuery was created to complement languages such as SQL and XSLT
- How to get started with XQuery using the XQuery tools which are already available
- How to query an XML document using XQuery and how to create new elements in the result using element constructors
- About the XQuery data model and how to use the different types of expression in XQuery, including the important FLWOR (for, let, where, order by, return) expressions
- How to use some XQuery functions
- What further developments are likely in future versions of XQuery including full text searching and update functionality

# Why XQuery?

- **Historical Factors**
  - In development, rough functionality
- **Technical Factors**
  - Embrace newer technologies such as binary data storage models as opposed to relational database models
- **Current Status**
  - Developing XQuery, XSLT 2.0, XPath 2.0
  - Using XSLT and XQuery
  - Comparing XSLT, XPath, XQuery



# XQuery Example Using: doc()

**SimpleBooks.xml**

```
<?xml version="1.0"?>
<Books>
 <Book>Beginning XML, 4rd Edition</Book>
 <Book>Beginning XML Databases</Book>
 <Book>Professional Web 2.0 Programming</Book>
</Books>
```

**collection()**: process more than one documents

**SimpleBooks.xquery**

```
doc("SimpleBooks.xml")/Books/Book
```

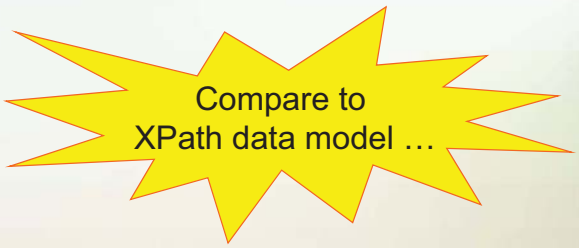
**To have Saxon run the query and display the output to the command window, enter:**

```
java net.sf.saxon.Query SimpleBooks.xquery
```

**Or use Kernow (Standalone tab)**

# Retrieving Nodes

- Using XPath expression to retrieve nodes
- Lacking axis support for
  - namespace *
  - ancestor
  - ancestor-or-self
  - following
  - following-sibling
  - preceding
  - preceding-sibling




**Example** BibAdapted.xml
BibQuery1.xquery: doc("BibAdapted.xml")/bib/book
(Try some more: BibQuery1a.xquery & BibQuery1b.xquery)



# Problem – Output Not Well-Formed

- Solution: element constructor
- Add literal start- and end-tags for root element
- Enclose query in { }

**Example**
SimpleBooks.xquery: <Books>
{doc("SimpleBooks.xml")/Books/Book}
</Books>
BibQuery2.xquery: <myNewBib>
{doc("BibAdapted.xml")/bib/book}
</myNewBib>

# The XQuery Prolog

- **XQuery Version Declaration**

```
xquery version "1.0";
```

- **XQuery Modules**

```
module namespace WROX =
  "http://www.wrox.com/XQuery/Books";
```

Note:

1. The *version declaration*, if present, must always come first.
2. Next is the *module declaration* (if there is one).
3. Then comes the rest of the prolog.



# Declarations

- **The base-url Declaration**
- **The namespace Declaration**
- **Default namespace Declarations**
- **Schema Imports**
- **Variable Declarations**
- **Validation Declarations**
- **The boundary-space Declaration**

Examples are in the book

# Computed Constructors

- **Allows elements and attributes to be constructed at runtime.**
- **Straightforward**
- **Use , to separate sequence of elementss**
- **Example: Library.xquery**

```
element library{
 element book {
  attribute year {2007},
  element title {
   "Beginning XML, 4th Edition"
   }
  },
 element book {
  attribute year {2006},
  element title {
   "Beginning XML Databases"
   }
  },
 element book {
  attribute year {2006},
  element title {
   "Professional Web 2.0
     Programming"
   }
 }
}
```



# Syntax (there are more …)

## XQuery Comments

(: After the scowl, we smile when the comment ends. :)

## Delimiting Strings

element Paragraph {
"Some content contained in paired double quotes"
}

*Or*

element Paragraph {
'Some content contained in paired apostrophes.'
}

# The XQuery Data Model

- Shared Data Model with XPath 2.0 and XSLT 2.0
  - Treelike hierarchy
- Node Kinds
  - Root is called document node, not root node
- Sequences of Node-Sets
  - (item 1, item 2, …, item n)
- Document Order
- Comparing Items and Nodes
  - Sequence can include nodes and atomic values
- Types in XQuery
  - XML Schema type
- Axes in XQuery



# FLWOR Expressions

for
let
where
order by
return

Must have either a for or a let

# FLWOR Expressions

## *for* expression

```
<items>
{for $i in (1,2,3,4) return <item>{$i}</item>}
(: or “for $i in 1 to 4 return <item>{$i}</item>” :)
</items>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<items>
  <item>1</item>
  <item>2</item>
  <item>3</item>
  <item>4</item>
</items>
```

ForIn.xquery



# FLWOR Expressions

## *for* expression: mixing atomic values and nodes

```
<items>
{for $i in (1,2, doc("Products.xml")/Products/Product/text(), 3, 4)
   return <item>{$i}</item>}
</items>
```

**Products.xml**

```
<?xml version="1.0"?>
<Products>
 <Product>Widget</Product>
 <Product>Gadget</Product>
 <Product>Knife</Product>
 <Product>Spoon</Product>
</Products>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<items>
  <item>1</item>
  <item>2</item>
  <item>Widget</item>
  <item>Gadget</item>
  <item>Knife</item>
  <item>Spoon</item>
  <item>3</item>
  <item>4</item>
</items>
```

ForIn2.xquery

# FLWOR Expressions

## *for* expression

```
<items>
{for $i in (1 to 5, 7, 8) return
 <group>{ for $a in (1 to ($i - 2)) return<item>{$a}</item>}
 </group>
 }
</items>
```


ForNested.xquery



# FLWOR Expressions

## Filtering with the *where* clause

```
<books>{
for $book in doc("BibAdapted.xml")/bib/book
        where $book/publisher = "Wrox Press" return
element book {
 attribute year {$book/@year},
 element title {$book/title/text()}
 }
}
</books>
```

Publisher.xquery

# FLWOR Expressions

## Sorting using the *order by* clause

```
<books>{
  for $book in doc("BibAdapted.xml")/bib/book
  let $t := $book/title/text() order by $t return
  <book><title>{$t}</title></book>
  }
</books>
```

## Sorting by descending order

**order by** $t **descending**

OrderByTitle.xquery



# FLWOR Expressions

## Conditional expressions

```
<MultiAuthor>
{for $book in doc("BibAdapted.xml")/bib/book
return  if (count($book/author) gt 2)
  then <book>
         <title>{$book/title/text()}</title>
         <NumberOfAuthors>{count($book/author)}</NumberOfAuthors>
        </book>
  else ()
}
</MultiAuthor>
```

Can also be done using **where**

MultiAuthor.xquery

# XQuery Functions

## The concat() Function: string concatenation

```
<ASaying>{
for $a in doc("Parts.xml")/Parts/Part[1]
  for $b in doc("Parts.xml")/Parts/Part[2]
  return  concat($a, " ", $b)
}</ASaying>
```



# XQuery Functions

## The count() Function: #nodes in node-set

```
<library
   count="{count(doc("SimpleBooks.xml")/Books/Book)}"
   >
 { for $b in doc("SimpleBooks.xml")/Books/Book return
   <book>{$b/text()}</book>
 }
</library>
```

# Using Parameters with XQuery

## Passing parameters from outside

**ParameterExample.xquery:**

```
declare variable $input as xs:string external;
<output>
 {$input}
</output>
```

```
java net.sf.saxon.Query ParameterExample.xquery
   input="Hello, World!"
```



# User-Defined Functions

```
declare namespace math =
   "http://wrox.com/namespaces/xquery/math";

declare function math:add($op1 as xs:integer, $op2 as
   xs:integer) as xs:integer
{
  $op1 + $op2
};

declare variable $op1 as xs:integer := 1;
declare variable $op2 as xs:integer := 2;
<add>
  <op1>{$op1}</op1>
  <op2>{$op2}</op2>
  <result>{math:add($op1, $op2)}</result>
</add>
```

# User-Defined Functions

```
declare namespace math =
   "http://wrox.com/namespaces/xquery/math";

declare variable $n as xs:integer external;

declare function math:factorial($integer as xs:integer)
   as xs:double
{
  if ($integer gt 1) then $integer * math:factorial($integer
   - 1) else 1
};

concat($n, "! = ", math:factorial($n))
```



# Looking Ahead

- **Update Functionality (as Part of XQuery)**
  - **XQuery 1.0 can't do insert, delete, update!**

  *www.w3.org/TR/xqupdate/*
- **Full-Text Search**

  *www.w3.org/TR/xquery-full-text/*