P.PORTO

ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

XPATH e XQUERY

Bruno Oliveira (bmo@estg.ipp.pt)

PEI Processamento Estruturado de Informação

Agenda

- XPath
  - Terminologia
  - Sintaxe
  - Steps: eixos, nodos de teste e predicados
  - Funções
  - Operadores
- Introdução a XQuery

XML e XPath

- O XPath (XML Path Language) trata um documento XML como uma árvore constituída por nodos.
- Existem diferentes tipos de nodos, como nodos de elementos, de atributos, de comentários e de texto.
- As expressões permitem navegar através dos nodos de um documento XML, possibilitando a localização de pontos específicos do documento.
- Uma expressão XPath devolve uma sequência de itens, como valores, elementos e atributos, pela mesma ordem com que surgem no documento XML.

Diferentes versões do XPath

- XPath versão 1.0: Tornou-se uma recomendação W3C em 16 de Novembro de 1999;
- XPath versão 2.0: Tornou-se uma recomendação da W3C em 23 de Janeiro de 2007;
  - É uma subconjunto do XQuery 1.0;
  - Disponibiliza uma biblioteca de funções e operadores mais poderosa;
- XPath versão 3.0:
  - Tornou-se uma recomendação da W3C em 8 Abril de 2014 e a versão 3.1 em 21 de Março de 2017;
  - O XPath 3.0 é um subconjunto do XQuery 3.0;

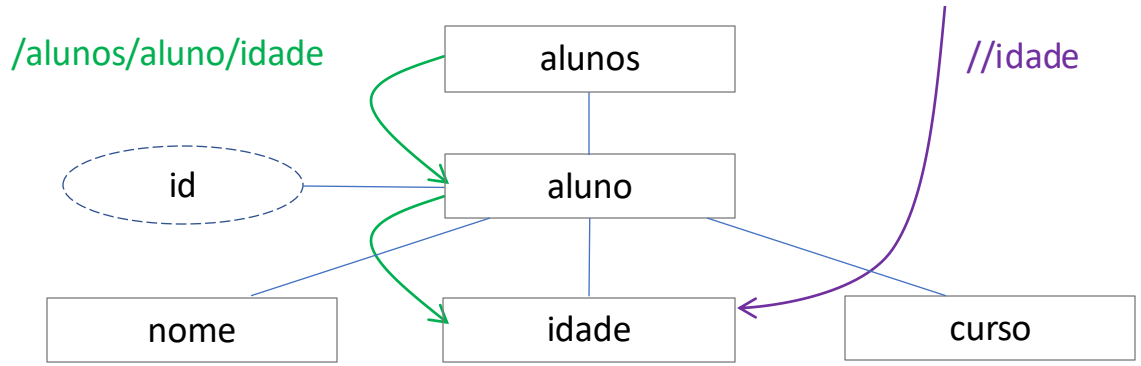
Diferentes versões do XPath

- O XPath usa expressões de caminho (path) para selecionar nodos ou conjuntos de nodos num documento XML.
- Uma expressão path consiste em um, ou mais, passos (steps) separados por “/” ou “//”.
- Path:
  - Absoluto: /Step1/Step2/…/StepN
  - Relativo: //Step1/Step2/…/StepN
- Steps:
  - eixo::nodoteste predicado1 predicado2 …

XML e XPath

- Quando a expressão XPath se inicia por "/", representa um caminho que começa no nodo raiz;
- Quando a expressão XPath se inicia por "//", representa os caminhos que fazem "match" com a expressão, independentemente da posição na árvore em que se encontram.
- A "/" permite descer um nível na árvore XML.
- A "//" possibilita "saltar" vários níveis na árvore.

XML e XPath



Expressões XPath

- Exemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<pessoas>
  <pessoa genero="M">
    <primeiroNome>Pedro</primeiroNome>
    <ultimoNome>Pinto</ultimoNome>
    <idade>28</idade>
  </pessoa>
  <pessoa genero="F">
    <primeiroNome>Fernando</primeiroNome>
    <ultimoNome>Santos</ultimoNome>
    <idade>30</idade>
  </pessoa>
</pessoas>
```

Expressão: /pessoas/pessoa/idade

Resultado: <idade>28</idade>

<idade>30</idade>

Expressão: //idade

Resultado: <idade>28</idade>

<idade>30</idade>

Expressões XPath

- Os steps são constituídos por eixos (axis), nodos de teste e 0 ou mais predicados.
- Um eixo é usado para localizar nodos relativamente ao nodo atual e especifica uma direção de navegação.
- Um nodo de teste seleciona nodos com base no seu nome ou no seu tipo.
- Um predicado consiste em condições, que são expressas entre parênteses retos, e que funcionam como filtros que restringem uma expressão Xpath.
- Exemplo:

Nodos de teste e Predicados

- Os nodos de teste podem ser, por exemplo:
  - nome do nodo/atributo: seleciona os elementos/atributos do nome indicado
  - * : seleciona todos os elementos/atributos do eixo especificado. Exemplo: //pessoa/parent::*
  - comment() : seleciona os nodos de comentário relativos ao eixo especificado.
  - text(): seleciona qualquer nodo de texto relativo ao eixo especificado. Exemplo: //pessoa/primeiroNome/text()

Nodos de teste e Predicados

- Os predicados podem restringir os nodos selecionados:
  - //aluno[@id='123']/nome – retorna o nome do aluno com o id "123"
  - //aluno[idade>18] – retorna os alunos que têm mais de 18 anos

Eixos

- O node test e o predicado são utilizados para filtrar os nodos especificados pelo axis;
- O XPath pode ser expresso de uma forma abreviada (/A/B/C) ou não abreviada (/child::A/child::B/child::C);
- Exemplo: Todas as pessoas com idade menor que 20:

```
/pessoas/child::*[idade<20] (equivalente a: /pessoas/pessoa[idade<20])
```

Eixos

Eixo	Significado
ancestor	Seleciona todos os nodos ancestrais do nodo atual
ancestor-or-self	Seleciona todos os nodos ancestrais do nodo atual e também o nodo atual
attribute	Seleciona todos os atributos do nodo atual. *attribute:: pode ser abreviado por "@"*
child	Seleciona todos os filhos do nodo atual. *child:: pode ser omitido*
descendant	Seleciona todos os descendentes (filhos, netos, etc.) do nodo atual
descendant-or-self	Seleciona todos os descendentes do nodo atual e também o nodo atual
following	Seleciona todos os nodos que estão a seguir ao nodo atual, exceto os descendentes
following-sibling	Seleciona todos os irmãos que estão a seguir ao nodo atual
parent	Seleciona o pai do nodo atual. *parent:: pode ser abreviado por ".."*
preceding	Seleciona todos os nodos que precedem o nodo atual, exceto os ancestrais
preceding-sibling	Seleciona todos os irmãos que estão antes do nodo atual
self	Seleciona o nodo atual. *self:: pode ser abreviado por "."*

Eixos

- Os caminhos XPath definem uma localização de um elemento utilizando caminhos absolutos e relativos;
- Os Axes são utilizados para identificar elementos pelo seu relacionamento;
- Exemplo: Selecionar todos os irmãos que sucedem ao elemento atual:
  - //pessoa[@genero="M"]/following-sibling::*
- Exemplo: Todos os filhos (elementos) do elemento atual:
  - //pessoa/child::*

Mais informação: https://www.w3schools.com/xml/xpath_axes.asp

Funções

- O XPath dispõe de uma biblioteca de funções como:

Função	Descrição	Exemplo
count()	Devolve a contagem de nodos	count(//aluno)
min()	Devolve o argumento com o menor valor	min((18,20))
max()	Devolve o argumento com o maior valor	max((18,20))
avg()	Retorna o valor médio dos argumentos	avg(//idade)
sum()	Retorna a soma dos valores especificados	sum(//idade)
last()	Retorna o último item do nodo especificado	//aluno[last()]
position()	Refere-se aos itens na posição indicada	//aluno[position() > 1]
concat()	Concatena as *strings* indicadas	//aluno/concat(nome, ', ', idade)
contains()	Retorna "verdade" se a 2ª string estiver contida na 1ª string	//nome[contains(., 'o')]
starts-with()	Retorna "verdade" se a 1ª string começa pela 2ª string	//nome[starts-with(.,'F')]
ends-with()	Retorna "verdade" se a 1ª string termina pela 2ª string	//nome[ends-with(.,'z')]
distinct-values()	Retorna apenas os valores que são diferentes	distinct-values(//curso)

Exemplos

- Exemplo: Selecionar o primeiro elemento pessoa que é filho do elemento pessoas:
  - /pessoas/pessoa[1]
- Exemplo: Selecionar o último elemento pessoa que é filho do elemento pessoas:
  - /pessoas/pessoa[last()]
- Exemplo: Selecionar os primeiros 2 elementos do tipo pessoa que são filhos do elemento pessoas:
  - /pessoas/pessoa[position()<3]

Exemplos

- Exemplo: Selecionar todos os elementos que têm um atributo com o nome genero:
  - //pessoa[@genero]
- Exemplo: Selecionar todos os elementos que têm um atributo com o nome genero com o valor: “M”:
  - //pessoa[@genero="M"]
- Exemplo: Selecionar os nomes das pessoas que têm mais do que 25 anos:
  - //pessoa[idade>25]/primeiroNome

Exemplos

- Selecionar os elementos filho do elemento pessoas:
  - /pessoas/* (* significa “todos”)
- Todos os elementos do documento:
  - //*
- Selecionar todas as pessoas que têm pelo menos um atributo:
  - /pessoas/pessoa[@*]

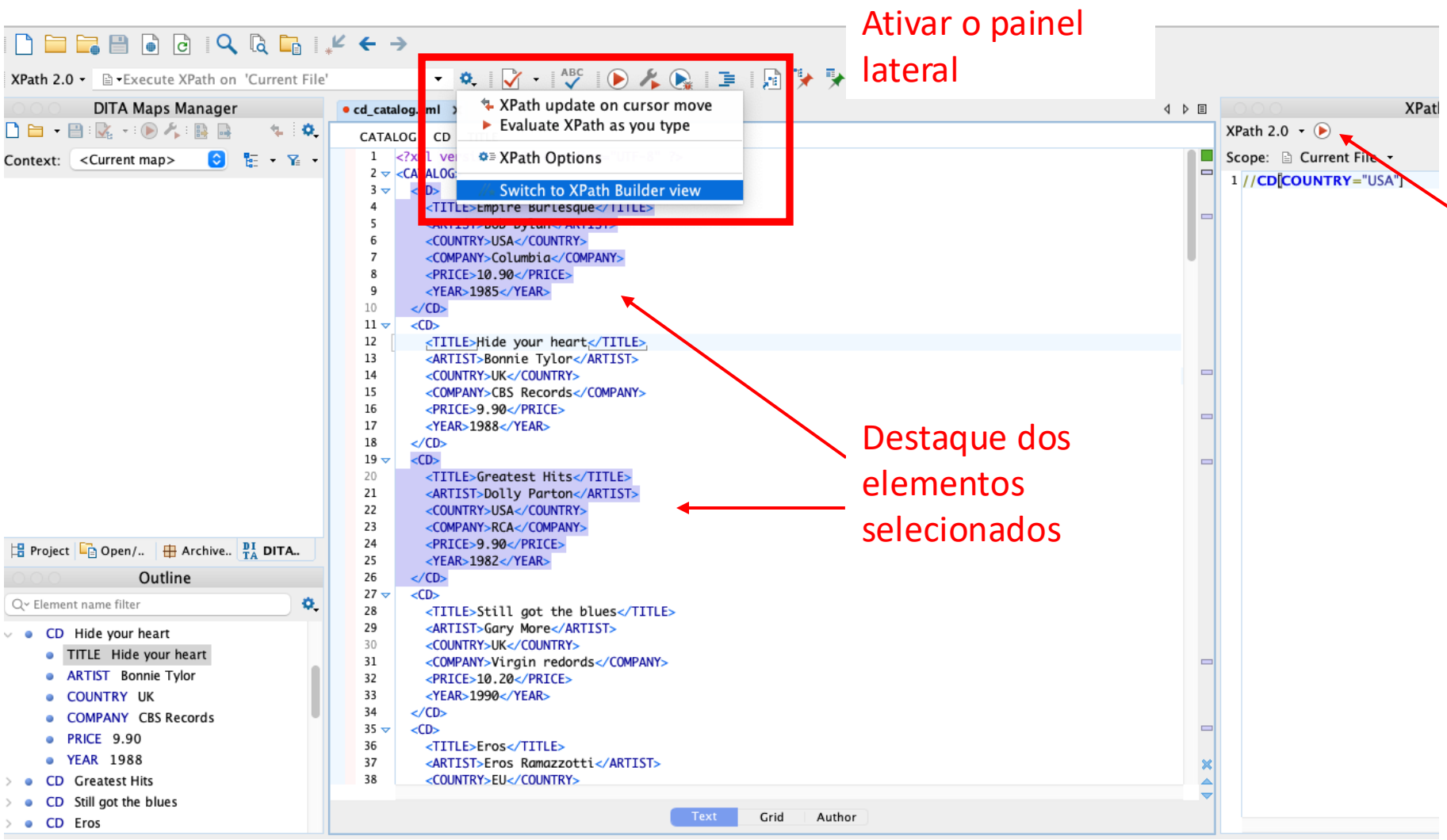
Exemplos

- Exemplos:
  - /pessoas/pessoa/idade -> retorna o elemento idade (<idade>17</idade>)
  - /pessoas/pessoa/idade/text() -> retorna o conteúdo do elemento (17)

Operadores XPath

- Alguns dos operadores que podem ser utilizados:
- |
  - Exemplo: //aluno/nome | //aluno/idade -seleciona vários caminhos
- + - * div
  - Exemplo: //aluno[1]/idade + 1 -retorna a idade + 1 do primeiro aluno
- = < <= > >= !=
  - Exemplo: /alunos/aluno[idade != 20]/nome -retorna o nome dos alunos que têm uma idade diferente de 20 anos
- And/or
  - Exemplo: //aluno[@id > 100 and idade < 20] -retorna os alunos com id maior que 100 e idade inferior a 20 anos

XPath utilizando Oxygen



XQuery

- XQuery foi desenhado para explorar o conteúdo de um documento XML;
- O XQuery incorpora o XPath;
- XQuery é uma recomendação da W3C;
- Com o XQuery podemos:
  - Aplicar transformações a documentos XML;
  - Selecionar informação relevante nos documentos.

Especificação oficial: https://www.w3.org/TR/xquery-30/

XQuery

- Existem 3 versões do XQuery:
  - 1.0 – Recomendação W3C em 2007
  - 3.0 – Recomendação W3C em 2014 – Versão estendida da XQuery 1.0 onde se destacam novas funções como: group by, expressão try-catch, etc
  - 3.1 – Recomendação W3C em 2017

O modelo de dados XQuery

- No modelo de dados XQuery, cada documento é representado como uma árvore de nodos (XML);
- Cada nodo tem uma identidade única!
- O XQuery baseia-se os tipos de dados standard XML (valores numéricos, strings, etc) para suportar a aplicação a linguagem;
- Como tal, também suporta o tipo de dados: node, que pode ser um element, atribute ou text-node.

Funções de input

- O XQuery utiliza funções de input para identificar a fonte de dados a processar. Existem duas funções de input:
  - doc() retorna um documento, identificando-o através de um Universal Resource Identifier (URI). Retorna um novo do tipo: document node;
- Exemplo:

```
doc("http://www.functx.com/input/order.xml")

doc("order.xml")
```

Funções de input

- collection() retorna uma coleção, que representa a sequência de nodos associados a um URI. É frequentemente utilizado para identificar a base de dados utilizada na consulta.
- Exemplo:

```
collection("bookstoreFiles")/bookstore/book/title
```

XQuery e Xpath - Exemplo

- Exemplo: Selecionar todos os elementos: Title do documento bookstore.xml:

```
doc("bookstore.xml")/bookstore/book/title
```

- Com um conjunto de documentos, podemos fornecer o caminho para a pasta:

```
collection(”./bookstoreFiles")/bookstore/book/title
```

XQuery e XPath

- Naturalmente, em XQuery também podemos utilizar os predicados (assim como qualquer elemento/funções das expressões XPath) para limitar os dados extraídos a partir de documentos XML;
- Exemplo: Selecionar todos os livros com um preço menor do que 30:

```
doc("bookstore.xml")/bookstore/book[price<30]
```

Funções e expressões

- As expressões FLWOR são um acrónimo de: "For, Let, Where, Order by, Return".
  - For – Seleciona uma sequência de nodos;
  - Let – Associa uma sequência a uma variável
  - Where – filtra os nodos
  - Order by – ordena os nodos
  - Return – retorno avaliado uma vez para cada nodo

Funções e expressões

- Uma expressão FLWOR começa, obrigatoriamente, por uma ou mais cláusulas for ou let.
- Após a definição das cláusulas for/let segue-se a cláusula opcional where.
- De seguida, pode utilizar-se (opcionalmente) a cláusula order by.
- No final, obrigatoriamente, é preciso definir a cláusula return.
- As cláusulas criam for e let criam tuplos e cada uma das restantes cláusulas “trabalha” sobre esses tuplos.

FLWOR

- Exemplo: Selecionar todos os livros com um preço menor do que 30:

```
for $x in  doc("bookstore.xml")/bookstore/book
where $x/price>30
order by $x/title
return $x/title
```

Seleciona todos os elementos “book” para a variável $livro

Ordena por “title”

Filtra por “price”

Define o que será retornado

```
30 Dias para Mudar de Vida, Detox Paleo
Harry Potter
I Wish You More
XML In A Nutshell
```

Nota: A consulta XQuery não retorna um documento XML válido

Cláusula return

- O resultado de uma expressão XQuery é sempre um documento XML que deverá ser válido;
- A cláusula return suporta element constructors que são bastante comuns para a aplicação de transformações;
- Podemos utilizar os element constructors para criar um novo documento (aplicar transformação) a partir de dados de um documento de origem.

Cláusula return

- Por exemplo, podemos criar uma lista de preços utilizando um element constructor:

```
for $b in doc("bookstore.xml")//book
return <quote>{ $b/title, $b/price }</quote>
```

Element construtor para criar um elemento quote. Representa a parte estática

Enclosed expression que utiliza chavetas para distinguir texto estático da expressão que será avaliada e substituída pelo seu resultado

Clausulas for e let

- Cada cláusula de uma expressão FLWOR é definida em termos de tuplos;
- As cláusulas for e let criam tuplos;
- Por isso, cada expressão FLWOR deve ter pelo menos uma cláusula let ou for;

Clausulas for e let

- Exemplo de tuplos:

```
for $x in
doc("bookstore.xml")/bookstore/book
where $x/price>30
return $x/title
```

tuplos

$x
Harry Potter
Lord of the Ring
Game of Thrones

- A variável x é associada a cada livro, um de cada vez, para criar uma série de tuplos;
- Cada tuplo contém uma associação com a variável x para um livro.

Clausulas for e let

- Exemplo utilizando let:

```
let $x := (1 to 5)
return <test>{$x}</test>
```

- A cláusula let associa uma variável a um resultado de uma expressão (não resulta em iteração);
- Quando não existem cláusulas for numa expressão FLWOR, apenas um tuplo é criado.

Nota: A consulta XQuery não retorna um documento XML válido

Cláusulas for e let

- Se uma cláusula let é utilizada numa expressão FLWOR que tem uma ou mais cláusulas for, as associações das cláusulas let são adicionadas aos tuplos gerados pelas cláusulas for:

```
for $i in (1, 2, 3)
let $j := (1, 2, 3)
return <tuple><i>{ $i }</i><j>{ $j }</j></tuple>
```

Cláusulas for e let

- Exemplo de consulta que combina cláusulas for e let:

```
for $b in doc("bookstore.xml")//book
let $c := $b//author
return <book> { $b/title, <count>{ count($c) }</count>} </book>
```

Variável posicional : at

- A cláusula for suporta variáveis posicionais (at) que identificam a posição de um determinado item na expressão que o gerou;
- Por exemplo, podemos retornar o título dos livros com a numeração do livro:

```
for $t at $i in doc("bookstore.xml")//title
return <title pos="{$i}">{data($t)}</title>
```

A função data é utilizada para extrair o valor atómico do elemento (semelhante ao node test do xPath: /text()

Clausula order by

- A cláusula order by ordena os tuplos antes da aplicação da clausula return ser avaliada de forma a alterar a ordem dos resultados;
- Exemplo da ordenação dos livros por ordem alfabética (ascending ou descending):

```
for $t in doc("bookstore.xml")//title
order by $t ascending
return $t
```

If - then - else

- Em XQuery, expressões if-then-else são permitidas:

```
for $x in doc("bookstore.xml")/bookstore/book
return if ($x/@category="CHILDREN")
          then <child>{data($x/title)}</child>
          else <adult>{data($x/title)}</adult>
```

XQuery e namespaces

- Com declaração de prefixo (útil quando temos mais do que um namespace):

```
declare namespace ns = "urn:OECD:StandardAuditFile-Tax:PT_1.04_01";
for $b in doc("SAFT.xml")/ns:AuditFile/ns:MasterFiles/ns:Product
return $b
```

XQuery e namespaces

- Com declaração de default namespace:

```
declare default element namespace "urn:OECD:StandardAuditFile-
Tax:PT_1.04_01";
for $b in doc("SAFT.xml")/AuditFile/MasterFiles/Product
return $b
```

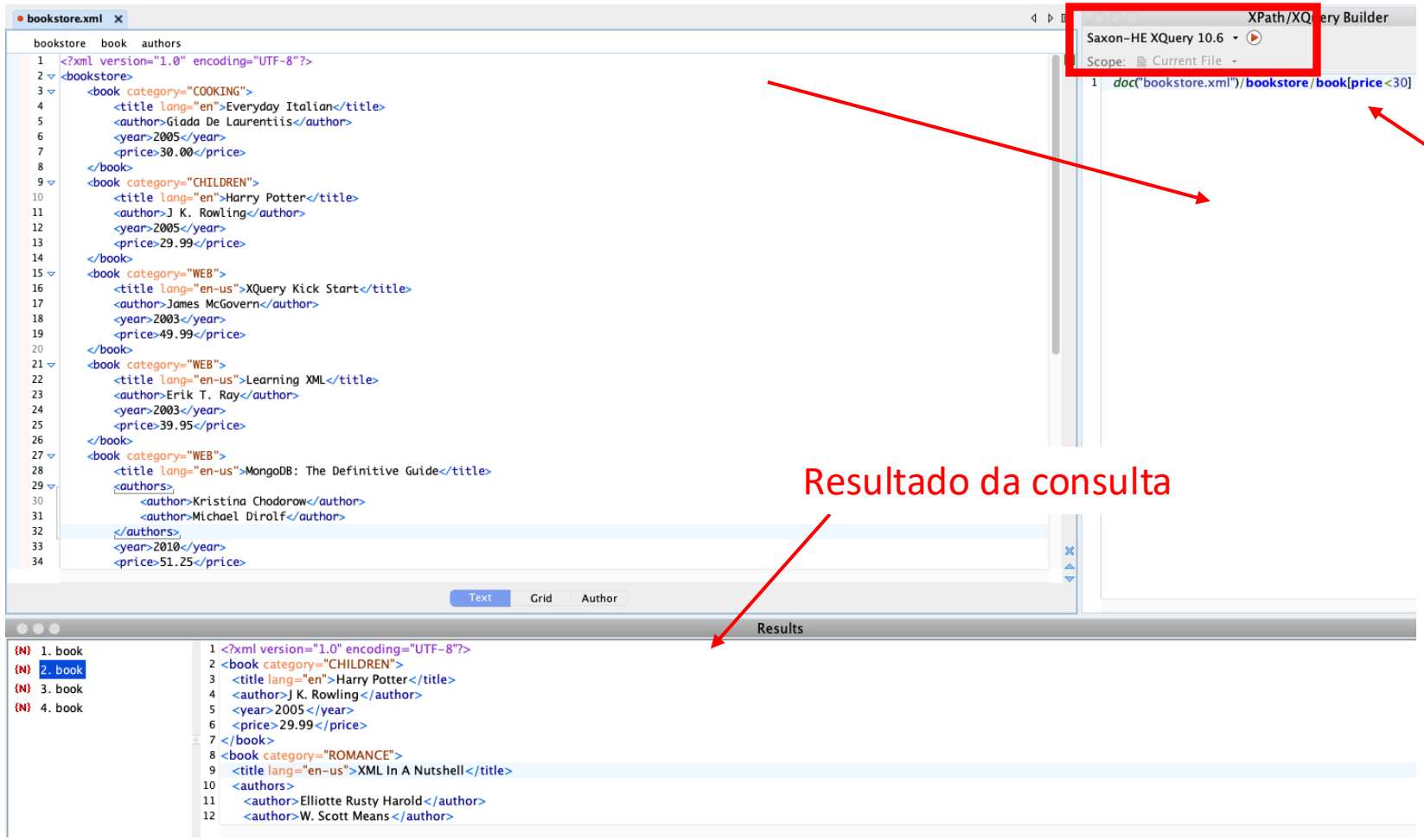
Comentários XQuery e tipos de dados

- O XQuery utiliza “smiles” para definição de comentários:

```
(: This is a comment! :)
```

- O XQuery suporta os tipos de dados utilizados no XSD, como tal reconhece tipos simples como: xs:integer, xs:decimal, ou xs:double;
- Por exemplo, qualquer número que contém apenas dígitos é considerado um inteiro, se tiver casa decimal é considerado como decimal.

XQuery utilizando Oxygen



Bibliografia/referências

Referências Web:

- https://www.w3schools.com/xml/xpath_intro.asp
- https://www.progress.com/tutorials/xquery/tour
- https://www.altova.com/training/xquery3/constructors

P.PORTO

ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

XPATH e XQUERY

Bruno Oliveira (bmo@estg.ipp.pt)

PEI Processamento Estruturado de Informação