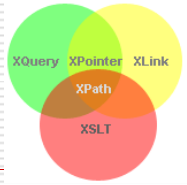
XPath

Nguyễn Hồng Phương
Email: phuong.nguyenhong@hust.edu.vn
Site: http://is.hut.edu.vn/~phuongnh
Bộ môn Hệ thống thông tin
Viện Công nghệ thông tin và Truyền thông
Đại học Bách Khoa Hà Nội

Nội dung

1. Giới thiệu

- Xpath được sử dụng để duyệt qua các phần tử và thuộc tính trong một tài liệu XML
- Cùng với XQuery và XPointer, Xpath là phần tử chính trong chuẩn XSLT của W3C.



2. Khái niệm XPath

- Là cú pháp để định nghĩa các phần của tài liệu XML
- Sử dụng biểu thức đường dẫn để duyệt tài liệu XML
- Chứa một thư viện các hàm chuẩn
- Là thành phần chính trong XSLT
- Là khuyến cáo của W3C từ 11/1999

2. Khái niệm Xpath (tiếp)

- Biểu thức đường dẫn
  - Sử dụng biểu thức đường dẫn để chọn lựa các nút/tập nút trong tài liệu XML
  - Khá giống với hệ thống file máy tính truyền thống
- Hàm chuẩn
  - Trên 100 hàm chuẩn được tích hợp sẵn: các hàm giá trị chuỗi, giá trị số, so sánh date time, Boolean,…

3. Nút

- Có 7 loại nút
  - phần tử
  - thuộc tính
  - text
  - namespace
  - chỉ thị xử lý
  - comment
  - nút tài liệu
- Coi tài liệu XML có cấu trúc cây

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<bookstore>
 <book>
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
 </book>
</bookstore>
```

- Nút gốc: <bookstore>
- Nút phần tử: <author>J K. Rowling</author>
- Nút thuộc tính: lang="en"

7

- Giá trị nguyên tố: nút không có nút cha và nút con
  - Ví dụ: J K. Rowling, "en"
- Item: là giá trị nguyên tố hoặc nút
- Quan hệ giữa các nút:
  - Cha: mỗi phần tử và thuộc tính đều có nút cha (trừ nút gốc)
  - Con: mỗi nút có thể có 0, 1 hoặc nhiều nút con

8

- Nút anh em: có cùng nút cha
- Tổ tiên: nút cha, nút cha của cha,…
- Hậu duệ: Nút con, nút con của con,…

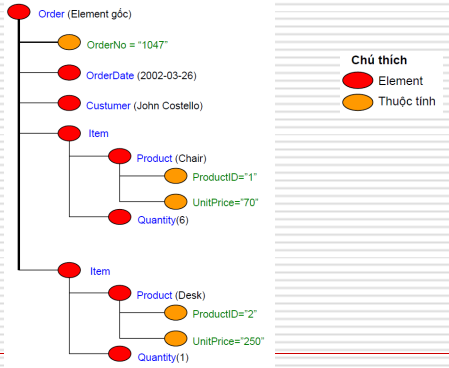
9

Ví dụ:

```
<?xml version="1.0"?>
<Order OrderNo="1047">
  <OrderDate>2002-03-26</OrderDate>
  <Customer>John Costello</Customer>
  <Item>
        <Product ProductID="1" UnitPrice="70">Chair</Product>
        <Quantity>6</Quantity>
  </Item>
  <Item>
        <Product ProductID="2" UnitPrice="250">Desk</Product>
        <Quantity>1</Quantity>
  </Item>
</Order>
```

10

Dạng cây



11

4. Cú pháp

- 4.1. Đường dẫn tuyệt đối
- 4.2 Đường dẫn tương đối
- 4.3. Kí tự đại diện
- 4.4. Vị từ
- 4.5. Một số hàm
- 4.6. Một số toán tử

12

4.1. Đường dẫn tuyệt đối

- Nếu đường dẫn XPath bắt đầu bởi dấu / thì có nghĩa đây là một đường dẫn tuyệt đối bắt đầu từ phần tử gốc.
- Ví dụ: muốn chọn nút Order ta viết như sau
  - Cú pháp đầy đủ: **/child::Order**
  - Cú pháp tắt: **/Order**
- Đi ra nhánh con Custumer bằng XPath như sau:
  - Cú pháp đầy đủ: **/child::Order/child::Custumer**
  - Cú pháp tắt: **/Order/Custumer**

13

- Muốn đi đến thuộc tính của nút, cần phải chỉ rõ từ khóa Attribute trong cú pháp đầy đủ hoặc @ trong cú pháp tắt.
- Để lấy thuộc tính OrderNo của nút Order ta dùng cú pháp Xpath:
  - Cú pháp đầy đủ: **/child::Order/Attribute::OrderNo**
  - Cú pháp tắt: **/Order/@OrderNo**

14

4.2. Đường dẫn tương đối

- Khi muốn trích một phần tử nào đó mà chỉ biết tên của phần tử này, không biết là phần tử này nằm ở vị trí nào thì có thể dùng đường dẫn tương đối.
- Dùng dấu // để chỉ cho trình phân tích biết đây là đường dẫn tương đối.
- Ví dụ: để trích các phần tử có tên là Product
  - Cú pháp đầy đủ: **//child::Product**
  - Cú pháp tắt: **//Product**

15

4.3. Ký tự đại diện

- Bất kỳ nút nào: ký tự đại diện *****
  - để lấy tất cả các phần tử con của phần tử Order, ta viết như sau:
    - Cú pháp đầy đủ: **/child::Order/child::***
    - Cú pháp tắt: **/Order/***
- Bất kỳ nút thuộc tính nào: @*
- Bất kỳ nút của bất kỳ kiểu nào: node()
- Ví dụ

/bookstore/*	Chọn tất cả nút con của phần tử bookstore
//*	Chọn tất cả phần tử trong tài liệu
//title[@*]	Chọn tất cả phần tử title có bất kỳ thuộc tính

16

4.4. Vị từ

- Được sử dụng để tìm một nút xác định hoặc một nút chứa một giá trị xác định
- Được đặt trong cặp ngoặc vuông []
- Ví dụ:
  - chọn phần tử book đầu tiên là con của phần tử bookstore: **/bookstore/book[1]**
  - chọn phần tử book cuối cùng là con của phần tử bookstore: **/bookstore/book[last()]**
  - chọn 2 phần tử book đầu tiên là con của phần tử bookstore: **/bookstore/book[position()<3]**

17

Vị từ: ví dụ

- lấy tất cả phần tử title có thuộc tính tên là lang: **//title[@lang]**
- lấy tất cả phần tử title có thuộc tính lang nhận giá trị 'eng': **//title[@lang='eng']**
- lấy ra phần tử book là con của bookstore có phần tử price với giá trị lớn hơn 35: **/bookstore/book[price>35.00]**
- **/bookstore/book[price>35.00]/title**

18

Vị từ: ví dụ

- lấy mọi phần tử Product có thuộc tính UnitPrice > 70
  - Cú pháp đầy đủ: **//child::Product[Attribute::UnitPrice>70]**
  - Cú pháp tắt: **//Product[@UnitPrice>70]**
- lấy những phần tử Item có phần tử con là Product và có thuộc tính ProductID=1:
  - Cú pháp đầy đủ: **//child::Item[child::Product/Attribute::ProductID=1]**
  - Cú pháp tắt: **//Item[Product/@ProductID=1]**

19

4.5. Một số hàm

Tên hàm	Mô tả	Ví dụ
count()	Hàm lấy tổng số nút con của một phần tử nào đó	**//Item[count(*)=2]** Chọn tất cả các phần tử Item có số phần tử con là 2
name()	Lấy tên của phần tử	**/Order/*[name()='Item']** Chọn tất cả các phần tử con của Order có tên là Item
not()	Hàm phủ định	//**Item/*[not(@*)]** Chọn tất cả các phần tử con của Item không chứa thuộc tính nào
normalize-space(str)	Hàm bỏ khoảng trắng	**//Item/*[normalizespace(@ProductID)='abc']** Chọn tất cả các phần tử con của Item có thuộc tính ProductID=abc (không phân biệt khoảng trắng)

20

4.5. Một số hàm (tiếp)

Tên hàm	Mô tả	Ví dụ
startswith(str,substr)	Hàm kiểm tra xem chuỗi str có chứa chuỗi substr (tính từ vị trí đầu tiên) hay không	**//item/*[starts-with(name(),'P')]** Chọn tất cả các phần tử con của Item có tên bắt đầu bởi ký tự P
contains(str,substr)	Kiểm tra một chuỗi str có chứa chuỗi con substr hay không	**//item/*[contains(name(),'u')]** Chọn tất cả các phần tử con của phần tử Item mà tên của các phần tử con này có chứ ký tự u
string-length(str)	Hàm lấy chiều dài của 1 chuỗi	**//Item/*[string-length(name())=5]** Chọn tất cả các phần tử con của Item mà độ dài tên của các phần tử con này là 5

21

4.5. Một số hàm (tiếp)

Tên hàm	Mô tả	Ví dụ
position()	Cho biết vị trí hiện tại của phần tử	**//Item[position()=5]** Chọn phần tử Item có vị trí là 5
floor()	Lấy giá trị nhỏ nhất gần với giá trị chỉ định	
ceiling()	Lấy giá trị lớn nhất gần với giá trị chỉ định	
last()	Vị trí nút cuối cùng	**//Item[last()]** Chọn phần tử Item cuối cùng

22

4.6. Một số axe

Tên	Mô tả	Ví dụ
|	Toán tử hoặc dùng để chọn ra một lần nhiều phần tử có điều kiện khác nhau	**//Item/*[starts-with(name(),'U') | startswith(name(),'Q')]** Chọn tất cả các phần tử là con của Item có có tên bắt đầu bởi ký tự P hoặc Q
descendant	Chọn phần tử con của phần tử chỉ định	**/Order/Item/Product/descendant::*** Chọn tất cả các phần tử là con của /Order/Item/Product
ancestor	Chọn phần tử cấp trên	**/Order/Item/Product/ancestor::*** chọn 2 phần tử Item và phần tử Order

23

4.6. Một số axe (tiếp)

Tên	Mô tả	Ví dụ
following-sibling	Chọn phần tử cùng cấp kế tiếp	**/Order/OrderDate/following-sibling::*** chọn các phần tử Custumer và hai phần tử Item theo sau và cùng cấp với phần tử OrderDate
preceding-sibling	Chọn phần tử cùng cấp trước đó	**/Order/Custumer/preceding-sibling::*** chọn phần tử OrderDate
following	Chọn phần tử theo sau phần tử chỉ định	**/Order/OrderDate/following::*** chọn phần tử Custumer và 2 phần tử Item và các phần tử con của Item

24

4.6. Một số axe (tiếp)

Tên toán tử	Mô tả	Ví dụ
preceding	Chọn các phần tử đứng trước phần tử chỉ định	**/Order/Custumer/preceding::*** chọn tất cả các phần tử đi trước phần tử Custumer
descendant-or-self	Chọn phần tử cấp dưới và phần tử chỉ định	**/Order/Item/descendant-or-self::*** Chọn tất cả các phần tử Item và các phần tử con của phần tử này
ancestor-or-self	Chọn phần tử cấp trên và phần tử chỉ định	**/Order/Item/product/ancestor-orself::*** chọn 2 phần tử product, 2 phần tử Item và phần tử Order

4.7. Một số toán tử

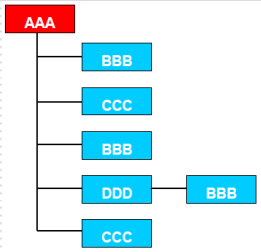
Toán tử	Mô tả	Ví dụ	Trả về
|	Tính toán hai tập nút	//book | //cd	Trả về tập nút với phần tử là book và cd
+	Cộng	6 + 4	10
-	Trừ	6 - 4	2
*	Nhân	6 * 4	24
div	Chia	8 div 4	2
=	Bằng	price=9.80	đúng nếu price là 9.80 sai nếutprice is 9.90
!=			

Toán tử	Mô tả	Ví dụ	Trả về
<	Nhỏ hơn	price<9.80	true nếu price >=9.00 false nếu price là 9.80
<=	Nhỏ hơn hoặc bằng	price<=9.80	
>	Lớn hơn	price>9.80	
>=	Lớn hơn hoặc bằng	price>=9.80	
or	or	price=9.80 or price=9.70	
and	and	price>9.00 and price<9.90	
mod	chia lấy số dư	5 mod 2	

5. Một số ví dụ

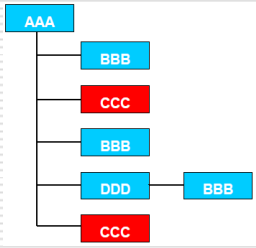
- Chọn phần tử gốc AAA: /AAA

```
<AAA>
    <BBB/>
    <CCC/>
    <BBB/>
    <BBB/>
    <DDD>
      <BBB/>
    </DDD>
    <CCC/>
</AAA>
```



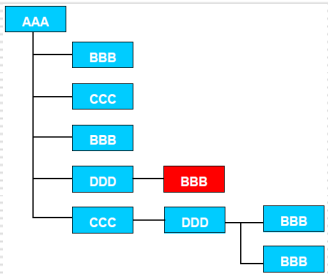
- Chọn phần tử CCC là con của AAA : /AAA/CCC

```
<AAA>
    <BBB/>
    <CCC/>
    <BBB/>
    <BBB/>
    <DDD>
    <BBB/>
    </DDD>
    <CCC/>
</AAA>
```

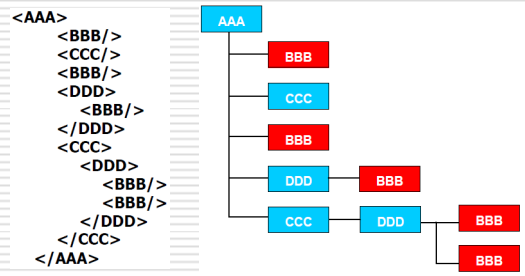


- Chọn tất cả các phần tử BBB là con của DDD mà DDD là con của AAA: /AAA/DDD/BBB

```
<AAA>
    <BBB/>
    <CCC/>
    <BBB/>
    <BBB/>
    <DDD>
      <BBB/>
    </DDD>
    <CCC/>
</AAA>
```

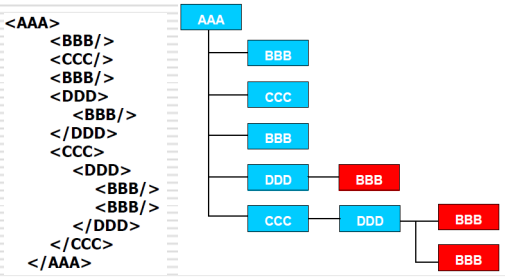


- Chọn tất cả các phần tử BBB: //BBB



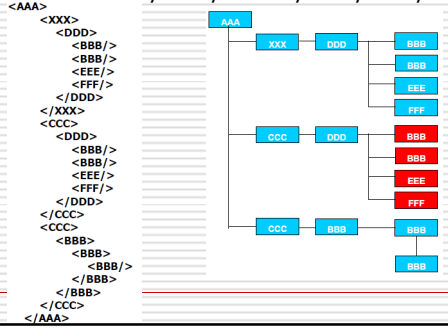
31

- Chọn tất cả các phần tử BBB là con của DDD: //DDD/BBB



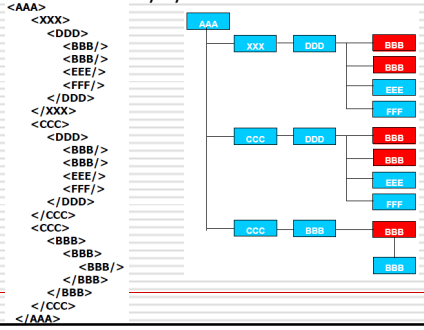
32

- Chọn tất cả các phần tử mà dòng họ của nó là /AAA/CCC/DDD: /AAA/CCC/DDD/*



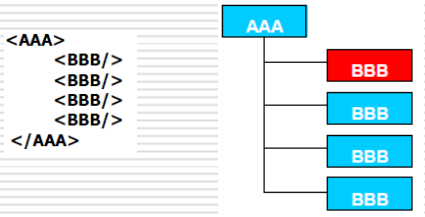
33

- Chọn tất cả các phần tử BBB mà nó có 3 cấp cha: /*/*/*/BBB



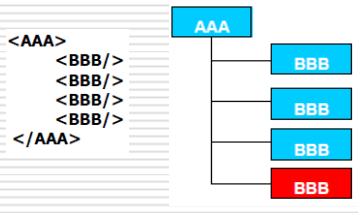
34

- Chọn phần tử BBB đầu tiên là con của AAA: /AAA/BBB[1]



35

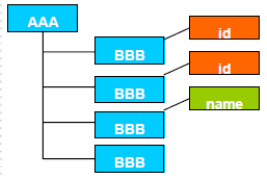
- Chọn phần tử BBB cuối cùng là con của AAA: /AAA/BBB[last()]



36

- Chọn tất cả các thuộc tính có tên id: //@id

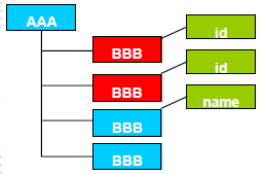
```
<AAA>
  <BBB id = "b1"/>
  <BBB id = "b2"/>
  <BBB name = "bbb"/>
    <BBB/>
</AAA>
```



37

- Chọn tất cả các phần tử BBB có thuộc tính tên là id: //BBB[@id]

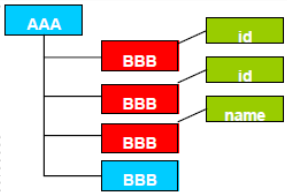
```
<AAA>
  <BBB id = "b1"/>
  <BBB id = "b2"/>
  <BBB name = "bbb"/>
  <BBB/>
</AAA>
```



38

- Chọn tất cả các phần tử BBB có tên thuộc tính: //BBB[@*]

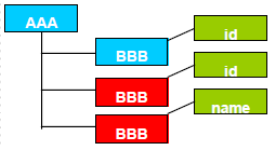
```
<AAA>
  <BBB id = "b1"/>
  <BBB id = "b2"/>
  <BBB name = "bbb"/>
  <BBB/>
</AAA>
```



39

- Chọn tất cả các phần tử BBB có tên thuộc tính là bbb, không phân biệt khoảng trắng: //BBB[normalize-space(@name)='bbb']

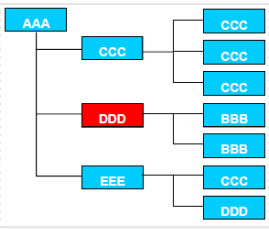
```
<AAA>
  <BBB id = "b1"/>
  <BBB name=" bbb "/>
  <BBB name = "bbb"/>
</AAA>
```



40

- Chọn tất cả các phần tử có chứa các phần tử mà trong đó có 2 phần tử con tên là BBB: //*[count(BBB)=2]

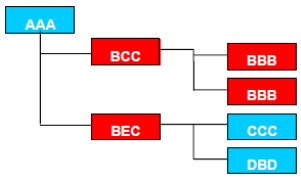
```
<AAA>
    <CCC>
      <BBB/>
      <BBB/>
      <BBB/>
    </CCC>
    <DDD>
      <BBB/>
      <BBB/>
    </DDD>
    <EEE>
      <CCC/>
      <DDD/>
    </EEE>
</AAA>
```



41

- Chọn tất cả các phần tử mà tên của nó bắt đầu là ký tự B: //*[starts-with(name(),'B')]

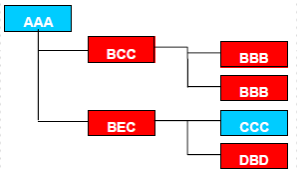
```
<AAA>
    <BCC>
      <BBB/>
      <BBB/>
    </BCC>
    <BEC>
      <CCC/>
      <DBD/>
    </BEC>
</AAA>
```



42

- Chọn tất cả các phần tử mà tên của nó có chứa ký tự B: //*[contains(name(),'B')]

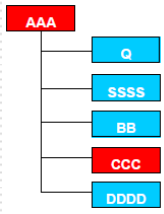
```
<AAA>
    <BCC>
        <BBB/>
        <BBB/>
    </BCC>
    <BEC>
        <CCC/>
        <DBD/>
    </BEC>
</AAA>
```



43

- Chọn tất cả các phần tử mà tên của nó có độ dài là 3: //*[string-length(name())=3)]

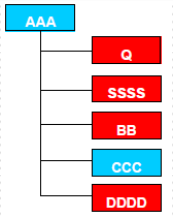
```
<AAA>
    <Q/>
    <SSSS/>
    <BB/>
    <CCC/>
    <DDDD/>
</AAA>
```



44

- Chọn tất cả các phần tử mà tên của nó có độ dài khác 3: //*[string-length(name())!=3)]

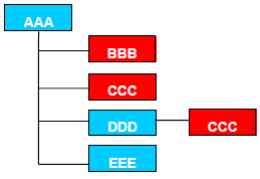
```
<AAA>
    <Q/>
    <SSSS/>
    <BB/>
    <CCC/>
    <DDDD/>
</AAA>
```



45

- Chọn tất cả các phần tử mà tên của nó là CCC hoặc BBB: //*[name()='CCC']| //*[name()='BBB']

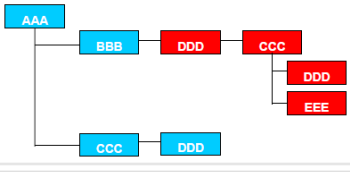
```
<AAA>
    <BBB/>
    <CCC/>
    <DDD>
        <CCC/>
    </DDD>
    <EEE/>
</AAA>
```



46

- Chọn tất cả các phần tử là con của AAA/BBB: /AAA/BBB/descendant::*

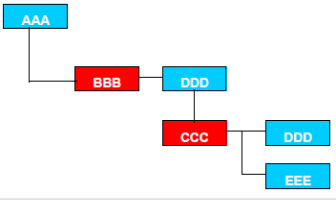
```
<AAA>
    <BBB>
        <DDD>
            <CCC>
                <DDD/>
                <EEE/>
            </CCC>
        </DDD>
    </BBB>
    <CCC>
        <DDD/>
    </CCC>
</AAA>
```



47

- Chọn tất cả các phần tử là cha của phần tử DDD: //DDD/parent::*

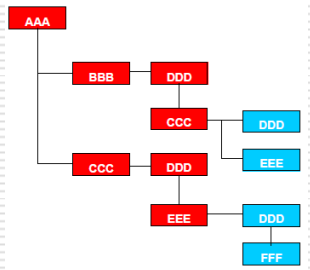
```
<AAA>
    <BBB>
        <DDD>
            <CCC>
                <DDD/>
                <EEE/>
            </CCC>
        </DDD>
    </BBB>
</AAA>
```



48

- Chọn tất cả các phần tử là tổ tiên của phần tử DDD: //DDD/ancestor::*

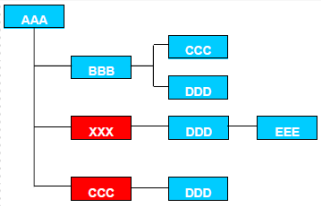
```
<AAA>
  <BBB>
    <DDD>
      <CCC>
        <DDD/>
        <EEE/>
      </CCC>
    </DDD>
  </BBB>
  <CCC>
    <DDD>
      <EEE>
        <DDD>
          <FFF/>
        </DDD>
      </EEE>
    </DDD>
  </CCC>
</AAA>
```



49

- Chọn tất cả các phần tử cùng cấp đi sau phần tử BBB: //BBB/followingsibling::*

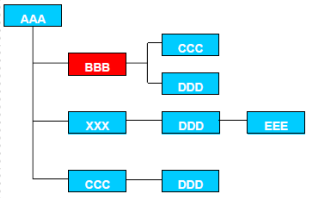
```
<AAA>
  <BBB>
    <CCC/>
    <DDD/>
  </BBB>
  <XXX>
    <DDD>
      <EEE/>
    </DDD>
  </XXX>
  <CCC>
    <DDD/>
  </CCC>
</AAA>
```



50

- Chọn tất cả các phần tử cùng cấp đi trước phần tử XXX: //XXX/precedingsibling::*

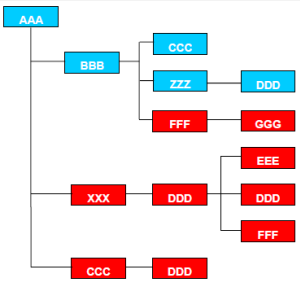
```
<AAA>
  <BBB>
    <CCC/>
    <DDD/>
  </BBB>
  <XXX>
    <DDD>
      <EEE/>
    </DDD>
  </XXX>
  <CCC>
    <DDD/>
  </CCC>
</AAA>
```



51

- Chọn tất cả các phần tử đi sau phần tử ZZZ: //ZZZ/following::*

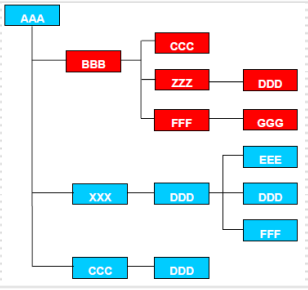
```
<AAA>
  <BBB>
    <CCC/>
    <ZZZ>
      <DDD/>
    </ZZZ>
    <FFF>
      <GGG/>
    </FFF>
  </BBB>
  <XXX>
    <DDD>
      <EEE/>
      <DDD/>
      <FFF/>
    </DDD>
  </XXX>
  <CCC>
    <DDD/>
  </CCC>
</AAA>
```



52

- Chọn tất cả các phần tử đi trước phần tử XXX ngoại trừ những phần tử gốc: //XXX/preceding::*

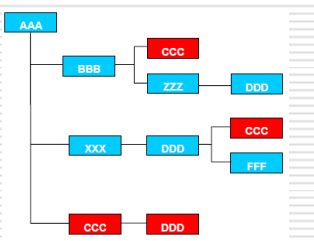
```
<AAA>
  <BBB>
    <CCC/>
    <ZZZ>
      <DDD/>
    </ZZZ>
    <FFF>
      <GGG/>
    </FFF>
  </BBB>
  <XXX>
    <DDD>
      <EEE/>
      <DDD/>
      <FFF/>
    </DDD>
  </XXX>
  <CCC>
    <DDD/>
  </CCC>
</AAA>
```



53

- Chọn tất cả các phần tử CCC và con của nó: //CCC/descendant-or-self::*

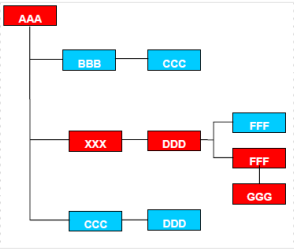
```
<AAA>
  <BBB>
    <CCC/>
    <ZZZ>
      <DDD/>
    </ZZZ>
  </BBB>
  <XXX>
    <DDD>
      <CCC/>
      <FFF/>
    </DDD>
  </XXX>
  <CCC>
    <DDD/>
  </CCC>
</AAA>
```



54

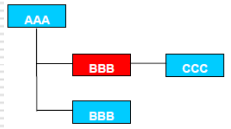
- Chọn tất cả các phần tử GGG và tổ tiên của nó: //GGG/ancestor-or-self::*

```
<AAA>
  <BBB>
    <CCC/>
  </BBB>
  <XXX>
    <DDD>
      <FFF/>
      <FFF>
        <GGG/>
      </FFF>
    </DDD>
  </XXX>
  <CCC>
    <DDD/>
  </CCC>
</AAA>
```



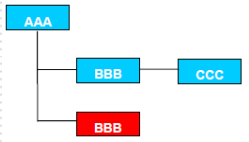
- Chọn phần tử BBB đầu tiên: //BBB[floor(1.2)]

```
<AAA>
  <BBB>
    <CCC/>
  </BBB>
  <BBB/>
</AAA>
```



- Chọn phần tử BBB thứ hai: //BBB[ceiling(1.2)]

```
<AAA>
  <BBB>
    <CCC/>
  </BBB>
  <BBB/>
</AAA>
```



6. Ví dụ tổng hợp

- Quan sát lại file book.xml:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<bookstore>

<book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
</book>

<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
```


File book.xml (tiếp)

```
<book category="WEB">
  <title lang="en">XQuery Kick Start</title>
  <author>James McGovern</author>
  <author>Per Bothner</author>
  <author>Kurt Cagle</author>
  <author>James Linn</author>
  <author>Vaidyanathan Nagarajan</author>
  <year>2003</year>
  <price>49.99</price>
</book>

<book category="WEB">
  <title lang="en">Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>

</bookstore>
```


- Nạp tài liệu XML
  - IE5, IE6:

```
var xmlhttp=new ActiveXObject("Microsoft.XMLHTTP")
```

  - Trình duyệt khác:

```
var xmlhttp=new XMLHttpRequest()
```

- Chọn nút
  - IE sử dụng phương thức selectNodes()

```
xmlDoc.selectNodes(xpath);
```

  - Trình duyệt khác sử dụng phương thức evaluate()

```
xmlDoc.evaluate(xpath, xmlDoc, null, XPathResult.ANY_TYPE,null);
```


Chọn tất cả các title

- /bookstore/book/title

```
<!DOCTYPE html>
<html>
<body>
<script>
function loadXMLDoc(dname){
if (window.XMLHttpRequest){
  xhttp=new XMLHttpRequest();
  }else{
  xhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xhttp.open("GET",dname,false);
xhttp.send("");
return xhttp.responseXML;
}

xml=loadXMLDoc("books.xml");
path="/bookstore/book/title"
```


Chọn tất cả các title (tiếp)

```
// code for IE
if (window.ActiveXObject)
{
var nodes=xml.selectNodes(path);

for (i=0;i<nodes.length;i++)
  {
  document.write(nodes[i].childNodes[0].nodeValue);
  document.write("<br>");
  }
}
// code for Mozilla, Firefox, Opera, etc.
else if(document.implementation &&
document.implementation.createDocument)
{
var nodes=xml.evaluate(path, xml, null, XPathResult.ANY_TYPE, null);
var result=nodes.iterateNext();
```


Chọn tất cả các title (tiếp)

```
while (result)
  {
  document.write(result.childNodes[0].nodeValue);
  document.write("<br>");
  result=nodes.iterateNext();
  }
}
</script>

</body>
</html>
```

- Kết quả:

Everyday Italian
Harry Potter
XQuery Kick Start
Learning XML

Chọn title của book đầu tiên

- /bookstore/book[1]/title

```
<!DOCTYPE html>
<html>
<body>
<script>
function loadXMLDoc(dname){
if (window.XMLHttpRequest)  {
  xhttp=new XMLHttpRequest();
  }
else  {
  xhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xhttp.open("GET",dname,false);
xhttp.send("");
return xhttp.responseXML;
}
```


Chọn title của book đầu tiên (tiếp)

```
xml=loadXMLDoc("books.xml");
path="/bookstore/book[1]/title";
// code for IE
if (window.ActiveXObject)
{
var nodes=xml.selectNodes(path);

for (i=0;i<nodes.length;i++)
  {
  document.write(nodes[i].childNodes[0].nodeValue);
  document.write("<br>");
  }
}
// code for Mozilla, Firefox, Opera, etc.
else if (document.implementation &&
document.implementation.createDocument)
```

Chọn title của book đầu tiên (tiếp)

```
{
var nodes=xml.evaluate(path, xml, null, XPathResult.ANY_TYPE,null);
var result=nodes.iterateNext();

while(result)
  {
  document.write(result.childNodes[0].nodeValue);
  document.write("<br>");
  result=nodes.iterateNext();
  }
}
</script>
</body>
</html>
```

- Kết quả Everyday Italian

67

Chọn title của book đầu tiên (tiếp)

- Chú ý: IE5 trở về sau cho rằng, phần tử đầu tiên bắt đầu từ 0
- Giải quyết vấn đề này:

```
xml.setProperty("SelectionLanguage","XPath");
xml.selectNodes("/bookstore/book[1]/title");
```

```
// code for IE
if (window.ActiveXObject){
xml.setProperty("SelectionLanguage","XPath");
var nodes=xml.selectNodes(path);
for (i=0;i<nodes.length;i++)
  {
  document.write(nodes[i].childNodes[0].nodeValue);
  document.write("<br>");
  }
}
// code for Mozilla, Firefox, Opera, etc.
.....
```

68

Chọn text của tất cả nút price

- /bookstore/book/price/text()

```
<!DOCTYPE html>
<html>
<body>
<script>
function loadXMLDoc(dname)
{
if (window.XMLHttpRequest)
  {
  xhttp=new XMLHttpRequest();
  }
else
  {
  xhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xhttp.open("GET",dname,false);
xhttp.send("");
return xhttp.responseXML;
}
```

69

Chọn text của tất cả nút price (tiếp)

```
xml=loadXMLDoc("books.xml");
path="/bookstore/book/price/text()"
// code for IE
if (window.ActiveXObject)
{
var nodes=xml.selectNodes(path);

for (i=0;i<nodes.length;i++)
  {
  document.write(nodes[i].nodeValue);
  document.write("<br>");
  }
}
// code for Mozilla, Firefox, Opera, etc.
```

70

Chọn text của tất cả nút price (tiếp)

```
else if (document.implementation &&
document.implementation.createDocument)
{
var nodes=xml.evaluate(path, xml, null, XPathResult.ANY_TYPE,null);
var result=nodes.iterateNext();

while (result)
  {
  document.write(result.nodeValue + "<br>");
  result=nodes.iterateNext();
  }
}
</script>
</body>
</html>
```

30.00
29.99
49.99
39.95

71

72

Lời hay ý đẹp

"Một vết chém của con dao có thể chữa khỏi, nhưng một vết chém của lưỡi thì khó lòng chữa được"

Ngạn ngữ Tây Ban Nha

73