PB138 — XPath

XPath basic principles

- XPath is a syntax used to specify parts of XML documents (primitive values, nodes, sequences of values or nodes
- XPath does not allow to specify parts of text nodes.
- Its name is derived from *path expression* providing a means of hierarchic addressing of the nodes in an XML tree.
- XPath uses syntax similar to *file system path*.
- XPath offers *standard functions library*, as well as user defined functions in either some XPath 2.0 or even XPath 1.x processors.
- XPath does not use XML syntax (it would be too long).

XPath specifications

- XPath 1.0 (revised Sep 7, 2015),
- XML Path Language (XPath) 2.0 and
- XML Path Language (XPath) 3.0 are W3C Recommendations (Apr 08, 2014)
- XPath 3.1 is a Recommendation (Mar 21, 2017)
- Backward compatibility: *nearly all* XPath 1.0 expressions continue to deliver the same result with XPath 3.0 (for exceptions see http://www.w3.org/TR/xpath-30/#id-backwards-compatibility)

XPath in other XML technologies

- XPath is used as a base for XSLT since version 1.0 and
- in XQuery since XPath version 2.0.

Crucial Learning Resources

- XPath Tutorial @W3Schools
- Zvon XPath 1.0 Tutorial to learn step by step (by Miloslav Nič)
- PathEnq XPath 2.0 online evaluator — nice for try&see
- Online XPath Tester - Evaluator by ExtendsClass

XPath domain: Advanced XML Data navigation

```
<?xml version="1.0"?>
<a>
   <b/> <!-- this is the //b[count(./*)=0] -->
   <b> <!-- this is the //b[./c] -->
     <c/>
   </b>
   <b> <!-- this is the //b[3] -->
       <!-- and also //b[./c] -->
     <c/>
   </b>
</a>
```

- Select the 3rd node *b*: `//b[3]`
- Select a node "b", which has a child node "c": `//b[./c]`
- Select an empty (eg. no child elements) node b: `//b[count(./*)=0]`

XPath domain: Transformation (XSLT)

- Select nodes that have to be processed next: `<xsl:apply-templates match="para"/>`
- Select value: `<xsl:value-of select="para/@id"/>`

XPath domain: Selection parts in XQuery

- (F)or part, eg. `for $para in $doc//para` selects all `para` in the document `doc`
- (L)et part, eg. `let $mypara := $doc//para[@owner='myself']`
- (W)here part, eg. `where $para[@class='task']`
- (O)rder part, eg. `order by $para/@created`

XPath domain: Modeling languages

- Schematron
- XML Schema

XPath paths and locations

Path describes (or "navigates" to) an XML document location. Paths syntax is constructed a similar way to paths in file systems, i.e.:

relative

related to a *context node* (CN), see further, or

absolute

related to the *root element* but predicates are evaluated in relation to CN.

XPath data types

- Since XPath 3.0 unified with the XML Schema and XQuery datatypes
- XQuery and XPath Data Model 3.0, W3C Recommendation 08 April 2014

Axes

- **Axes** (singular axis, plural axes) are sets of document elements, related to (usually relatively) to context.
- **Context** is formed by a *document* and the current (*context*) node (CN).

List of Axes (1)

`child`

contains direct child nodes of CN

`descendant`

contains all descendants of CN except attributes.

`parent`

contains the CN parent nod (if it exists)

`ancestor`

contains all ancestors of CN - means parents, grandparents, etc to a root element (if the CN is not the root element itself)

`following-sibling`

contains all following siblings of CN (the axis is empty for NS and attributes)

`preceding-sibling`

dtto, but it contains the preceding sibling.

List of Axes (2)

`following`

contains all nodes following the CN (except the attributes, child nodes and NS nodes)

`preceding`
: dtto, but contains preceding nodes (except ancestors, attributes, NS)

`attribute`
: contains attributes (for elements only)

`namespace`
: contains all NS nodes of CN (for elements only)

`self`
: the CN itself

`descendant-or-self`
: contains the union of descendant and self axes

`ancestor-or-self`
: contains the union of ancestor and self axes

XPath online testers

- It is possible to try evaluation of XPath expressions upon a provided XML document by using many online testers without the need of (local PC) installation.
- Such as http://codebeautify.org/Xpath-Tester or
- PathEnq XPath 2.0 online evaluator
- XPath online tester also allows to evaluate XPath against an XML document

Example `//b/child::*`

```
<?xml version="1.0"?>
<a>
  <b/>
  <b>
      <c/> <!-- this "c" will be selected -->
  </b>
  <b>
      <c/> <!-- and this "c" too -->
  </b>
</a>
```

Example `//b/descendant::*`

```
<?xml version="1.0"?>
<a>
  <b/>
  <b>
       <c> <!-- everything "under b" will be selected -->
           <d/> <!-- i.e. this "d" too -->
       </c>
  </b>
  <b>
       <c/> <!-- and this "c" too -->
  </b>
</a>
```

Example //d/parent::*

```
<?xml version="1.0"?>
<a>
  <b/>
  <b>
         <c> <!-- this "c" is the parent of "d" -->
            <d/>
         </c>
  </b>
  <b>
         <c/>
  </b>
</a>
```

Example //d/ancestor::*

```
<?xml version="1.0"?>
<a> <!-- this "a" is ancestor of "d" -->
   <b/>
   <b> <!-- this "b" is ancestor of "d" -->
       <c> <!-- this "c" is ancestor of "d" -->
           <d/>
       </c>
   </b>
   <b>
        <c/>
   </b>
</a>
```

Example //b/following-sibling::*

```
<?xml version="1.0"?>
<a>
  <b/> <!-- every child of "a" after this "b" is following-sibling -->
  <b> <!-- this "b" too -->
     <c>
        <d/>
     </c>
  </b>
  <b> <!-- this "b" too -->
      <c/>
  </b>
</a>
```

Example //b/preceding-sibling::*

```
<?xml version="1.0"?>
<a>
  <b/> <!-- this "b" too -->
  <b>
       <c>
           <d/>
       </c>
  </b> <!-- this "b" is preceding-sibling -->
  <b> <!-- every child of "a" before this "b" is preceding-sibling -->
       <c/>
  </b>
</a>
```

Example /a/b/c/following::*

```
<?xml version="1.0"?>
<a>
     <b/>
     <b>
         <c>
              <d/>
         </c> <!-- every element starting after "c" is following -->
         <e/> <!-- such as this "e" -->
     </b>
     <b> <!-- this "b" too -->
         <c/> <!-- this "c" too -->
     </b>
</a>
```

Example /a/b/e/preceding::*

```
<?xml version="1.0"?>
<a>
     <b/> <!-- this "b" too -->
     <b> <!-- this "b" too -->
         <c> <!-- this "c" too -->
             <d/> <!-- this "d" too -->
         </c>
     </b>
```

Example /a/b/e/preceding::*

```
     <b>
         <d/> <!-- such as this "d" -->
         <e/> <!-- every element starting before "e" is preceding -->
     </b>
</a>
```

Predicates

- Figure: `/article/para[3]` — selects the 3rd paragraph (element `para`) of article (element `article`)
- Simplest predicate expression is proximity position specification — see `preceding`.
- Attention at reverse axes (`ancestor`, `preceding`, ...) - position is numbered always from the context node, means opposite to document physical location directions.
- Position specification 3 can be replace by the expression `position()=3`.

Expressions

- Used in *predicates* for calculations. Expressions may contain XPath functions. Expressions may operate on:
  - text strings
  - numbers (floating-point numbers)
  - logical values (boolean)
  - nodes
  - sequences.

Short notation — examples 1

`para`
: selects all child nodes of context node with name `para`

`*`
: selects all element children of the context node

`text()`
: selects all text node children of the context node

`@name`
: selects the `name` attribute of the context node

`@*`
: selects all the attributes of the context node

`para[1]`
: selects the first `para` child of the context node

`para[last()]`
: selects the last `para` child of the context node

`*/para`
: selects all `para` grandchildren of the context node

Short notation — examples 2

`/doc/chapter[5]/section[2]`
: selects the second `section` of the fifth `chapter` of the `doc`

`chapter//para`
: selects all descendants of element `chapter` with name `para`

`//para`
: selects all elements `para` in the document

`//olist/item`
: selects all elements `item` with parent element `olist`

`.//para`
: selects all descendant nodes of the context node with name `para`

`..`
: selects the parent node of the context node

`../@lang`
: selects a `lang` attribute of the context node parent node

XPath — short notation (2)

Most common used short notation is at child axis

- we use article/para instead of `child::article/child::para`.
- at attribute:we use `para[@type="warning"]` instead of `child::para[attribute::type="warning"]`
- The next used short notation is `//` instead of `/descendant-or-self::node()/`
- and of course shortcuts `.` and `..`

For clarity, we keep sometimes the longer form: Do not fight it at all costs!

XPath 2.0

- Final specification available at http://www.w3.org/TR/xpath20/
- Different point of view on return values of XPatch expressions: everything is a sequence (even containing a single element) → removes the set node order problems
- Introduces conditional expressions and cycles.

XPath 2.0

- Introduces user-defined functions (dynamically evaluate XPath expressions)
- Users can uses general and existential quantifiers, for example `exist student/name="Fred"`, `all student/@id`
- For more details see http://www.saxonica.com/, pages contains the XPath/XSLT/XQuery processor Saxon as well.

XPath 2.0 - examples

String functions

http://www.fi.muni.cz/~tomp/xml03/xpath20/string.html

Numeric functions

http://www.fi.muni.cz/~tomp/xml03/xpath20/numeric.html

Sequence functions

http://www.fi.muni.cz/~tomp/xml03/xpath20/sequence.html

Boolean functions

http://www.fi.muni.cz/~tomp/xml03/xpath20/boolean.html

Other resources on XPath

- Programming in XPath 3.0 (D. Novatchev)
- XPath functions (Mozilla)