Navigating the XML Tree

Syd Bauman
Navigating the XML Tree

XML representations

```
<?xml version="1.0" encoding="UTF-8"?>
<book>
  <introduction>Blah blah blah ... </introduction>
  <chapter>
    <heading>Wines</heading>
    <section>White wines ... </section>
    <section>Red wines ... </section>
  </chapter>
  <chapter>
    <heading>Beers</heading>
    <section>Ales ... </section>
    <section>Lagers ... </section>
  </chapter>
  <index> stuff ... </index>
</book>
```

Note that pressing Control-minus and Control-plus will make the text larger or smaller, allowing the images to grow or shrink too.
Navigating the XML Tree

Sample document fragment

<?xml version="1.0" encoding="UTF-8"?>
<lg type="limerick" rhyme="aabba" n="3">
  <head>Warp Speed, Ms Bright!</head>
  <l>There was a young lady named <rhyme label="a">Bright</rhyme>,</l>
  <l>Who travelled much faster than <rhyme label="a">light</rhyme>,</l>
  <l>She departed one <rhyme label="b">day</rhyme>,</l>
  <l>In a <term xml:id="t17">relative</term> way <rhyme label="b">way</rhyme>,</l>
  <l>And returned on the previous <rhyme label="a">night</rhyme>.</l>
  <note target="#t17">See</note>

Sample portion of TEI tree

Grossly simplified sub-tree representation of a TEI document.
There was a young lady named Bright, Whose speed was far faster than light; She started one day In a relative way, sAnd returned on the previous night.


XSLT view

Same fragment, but adding text and attribute nodes
Element nodes have just the element’s local name; attribute nodes have just the name of the attribute; text nodes have just the word text.

The context node

- Important definition: The **context node** is where we are now in the XML tree.
- In XPath, and therefore in XSLT, the processor is always somewhere in the tree.
- From the context node, you can travel anywhere else in the tree.
- We do that by travelling along **XPath axes**.
Axes: `self::`
Navigating the XML Tree

Descendants (child:: and descendant::)
Navigating the XML Tree

Ancestors (parent:: and ancestor::)
This is where the analogy with genealogical trees breaks down. The preceding:: axis includes a set of nodes which do not match any familial group. It's basically all the nodes which both started and finished before the context node.
The following axis contains only elements which start after the context node has finished.
Axes: the "XPath Butterfly"

... lets us (among other things) select nodes in the tree.
... is used by XSLT, XQuery, Schematron; XLink and XPointer, too.
Navigating the XML Tree

Try it!

- Download the example short David Copperfield XML file and open it in Oxygen: http://web.uvic.ca/~mholmes/dhoxss2013/examples/copperfield.xml
- In the upper left corner there is a text-entry field (looks like a search box). Ensure the box is labelled XPath 2.0. Then type in /TEI/teiHeader/fileDesc/titleStmt/author.

Basic filepath-like path expressions

A bare-bones XPath expression is similar to filesystem addressing: if the path starts with a solidus (/ aka "forward slash"), then it represents a path from the root; if it does not start with a solidus then it represents a path from "here".

/TEI/teiHeader/fileDesc/titleStmt/title

list/item/label
What's another way of saying "here"?

### XPath axes: me, myself, I

<table>
<thead>
<tr>
<th>Short</th>
<th>Long</th>
<th>Means</th>
<th>Try it!</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>self::node()</td>
<td>me, whatever I happen to be (element, attribute, comment, processing instruction, document root, or even text)</td>
<td>oXygen shows node cursor is in</td>
</tr>
</tbody>
</table>

(Kay, chapter 9)

### XPath axes: my children

<table>
<thead>
<tr>
<th>Short</th>
<th>Long</th>
<th>Means</th>
<th>Try it!</th>
</tr>
</thead>
<tbody>
<tr>
<td>head</td>
<td>child::head</td>
<td>my <code>&lt;head&gt;</code> children</td>
<td>List all <code>&lt;head&gt;</code> children for <code>&lt;div&gt;</code> children of the body</td>
</tr>
</tbody>
</table>
| *     | child::* | my `element` children, whatever they are | List all elements which are children of `<div>`.

(Kay, chapter 9)

/TEI/text/body/div/head, or /TEI/text/body/div/child::head

### XPath axes: my descendants

<table>
<thead>
<tr>
<th>Short</th>
<th>Long</th>
<th>Means</th>
<th>Try it!</th>
</tr>
</thead>
<tbody>
<tr>
<td>//div</td>
<td>descendant::div</td>
<td>my <code>&lt;div&gt;</code> descendants</td>
<td>List all the <code>&lt;head&gt;</code> elements</td>
</tr>
</tbody>
</table>

(note: long and short are not exactly equivalent)
Navigating the XML Tree

(Kay, chapter 9)

/TEI/descendant::head but compare //head with /TEI/text/body//head; even when the cursor is in the <body> tag, the // takes us back to the root.

XPath axes: my parent

short   long       means                                           try it!
..       parent::node()                                        List the parents of
[none]   parent::author my parent, whatever it happens to be (element or root) <note> elements

(Kay, chapter 9)

Contrast the use of //p/parent::div with //p/parent::node().

XPath axes: my ancestors

short   long       means                                           try it!
[none]   ancestor::* all my ancestors                          List the ancestors of

(Kay, chapter 9)

Note that the ancestor axis ultimately includes everything.
XPath axes: following and following-sibling

<table>
<thead>
<tr>
<th>short</th>
<th>long</th>
<th>means</th>
<th>try it!</th>
</tr>
</thead>
<tbody>
<tr>
<td>[none]</td>
<td>following::p</td>
<td>all the &lt;p&gt;s that come after me</td>
<td>Find all the &lt;p&gt; elements that</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>occur after the &lt;publicationStmt&gt;</td>
</tr>
<tr>
<td>[none]</td>
<td>following-sibling::*</td>
<td>the children of my parent that come after me</td>
<td>Find all the children of &lt;div&gt; that follow &lt;head&gt;</td>
</tr>
</tbody>
</table>

(Kay, chapter 9)

1) /TEI/teiHeader/fileDesc/publicationStmt/following::p or //publicationStmt/following::p. Note how it doesn't select the <p> child of <publicationStmt>.

2) //head/following-sibling::* . Note that <note> elements are not included -- why?

XPath axes: preceding and preceding-sibling

<table>
<thead>
<tr>
<th>short</th>
<th>long</th>
<th>means</th>
<th>try it!</th>
</tr>
</thead>
<tbody>
<tr>
<td>[none]</td>
<td>preceding::p</td>
<td>the &lt;p&gt;s that come before me</td>
<td>put your cursor in the last &lt;p&gt; tag in the document, and find all the preceding &lt;p&gt; tags.</td>
</tr>
<tr>
<td>[none]</td>
<td>preceding-sibling::p</td>
<td>the &lt;p&gt; children of my parent that come before me</td>
<td>How many preceding-sibling &lt;p&gt; tags are there for the last &lt;p&gt; tag in the document?</td>
</tr>
</tbody>
</table>

(Kay, chapter 9)
XPath axes: attributes

<table>
<thead>
<tr>
<th>short</th>
<th>long</th>
<th>means</th>
<th>try it!</th>
</tr>
</thead>
<tbody>
<tr>
<td>@target</td>
<td>attribute::target</td>
<td>list all the @target attributes</td>
<td>list all the @target attributes</td>
</tr>
<tr>
<td>@*</td>
<td>attribute::*</td>
<td>all of my attributes</td>
<td>How many attributes are there in the document?</td>
</tr>
</tbody>
</table>

(Kay, chapter 9)

Note that the attributes don't include the xmlns declaration.

XPath axes: summary

- self::
- child::, descendant::
- parent::, ancestor::
- following::, following-sibling::
- preceding::, preceding-sibling::
- attribute::