XPath

Mario Alviano

University of Calabria, Italy

A.Y. 2018/2019

Outline

Why XPath?

- How to access information stored in an XML document?
- XPath is a query language for XML
  - Useful to select parts of an XML document
  - Allows to check whether given conditions are satisfied
  - Thought for the hierarchical structure of XML documents
- XPath has been successful in the XML community
- It is used by many languages
- It will be used for many years! (maybe)

Example. XPath expression

```
/company/employee[@id="123"]/salary
```

XPath expressions

- The main concept in XPath is that of **expression**
- There are three types of expressions
  1. **Value expression:** allow to define propositions on values
  2. **Path expression:** allow to select parts of the XML document
  3. **Node set expression:** allow to combine results of several expressions

Path expressions

- Path expressions can be relative or absolute (if start with `/`)
- Are made of a sequence of **steps** separated by `/`
- Each step has three parts
  1. an **axis**
  2. a **node-test**
  3. zero o more **predicates**

```
axis::node-test[predicates]
```

At each step, the context (i.e., the result) is modified following the axis, then restricted to the elements with name equal to `node-test`, and eventually filtered by predicates.

Path expressions: axis

- Axis modify the context
- Valid axis are
  - `child` (default) and `descendant` (roughly `/`)
  - `parent` (`..`) and `ancestor`
  - `following-sibling` and `preceding-sibling`
  - `self` (`.`), `descendant-or-self` and `ancestor-or-self`
  - `following` and `preceding`
  - `attribute` (`@`) and `namespace`

Examples

- All the ancestors of professors that were professors
  `//professor/ancestor::professor`
  Note: we used the `/` abbreviation. It is similar to `descendant`, but according to the W3C specification "`//` is short for `/descendant-or-self::node()/`"
- Titles of movies
  `movies/title`

Path expressions: node-tests

- Node-tests restrict the context
- Valid node-tests are
  - The name of an element
  - The wildcard `*`
  - `comment()`
  - `text()`
  - `processing-instruction()`
  - `node()`

Warning!

The node-test `*` restricts the context to elements only, while `node()` restricts the context to elements and attributes only.

Path expressions: predicates

- Predicates allow to further filter the context
- Any value expression can be used (see the next slide)
- Non-boolean expressions are interpreted as follows:
  - integers: equivalent to `position() = the-result-of-the-expression`
  - strings: true if the string is nonempty
  - node sets: true if the node set is nonempty

Value expressions

- Value expressions include
  - Strings and numeric literals: `"home", 42, 1.23, ...`
  - Variable references: `$x, $y, $myVar, ...`
  - Function invocation: `fn:upper-case($x), ...`
  - Logic and arithmetic comparison: `$x < 10, $x > 5 or $x < 2, ...`
  - Path expressions

Value expressions: functions

- Value expressions can use several functions
- XPath functions are defined in the namespace `http://www.w3.org/2005/xpath-functions`
  - `last()`: position of the last element in the context
  - `position()`: position of each element in the context
  - `string(arg)`, `concat(s1,s2,...)`, `starts-with(s1,s2)`, `contains(s1,s2)`, `substring(s, start, len?)`, `string-length(s)`, `normalize-space(s)`
  - `Boolean(arg)`, `not(arg)`, `true()`, `false()`
  - `lang(lang)`: true if the language of the current node is that given in the argument
  - `number(arg)`, `floor(number)`, `ceiling(number)`, `round(number)`
  - `count(arg,arg,...)`, `sum(arg,arg,...)`, `min(arg,arg,...)`, `max(arg,arg,...)`, `avg(arg,arg,...)`
- **Convention:** use the prefix `fn`

Node set expressions

- Allow to combine contexts obtained by several expressions
  - `|` (o `union`)
  - `intersect`
  - `except`
  - `,` (concatenation)

```
<Library title="Alpha">
  <Book title="Bravo">
    <Chapter title="Charlie">Traveling with a poodle</Chapter>
    <Chapter title="Delta">Mouth of the Mississippi</Chapter>
  </Book>
  <Book title="Echo">
    <Chapter title="Foxtrot">Dance to four-quarters time</Chapter>
    <Part title="Golf">
      <Chapter title="Hotel">Check in, but not out</Chapter>
      <Chapter title="India">Indus to the Ganges</Chapter>
    </Part>
  </Book>
  <Book title="Juliet">
    <Part title="Kilo">
      <Chapter title="Lima">Peru is here too</Chapter>
      <Chapter title="Mike">Decorated Sistine Chapel</Chapter>
    </Part>
    <Part title="November">
      <Chapter title="Oscar">Academy Awards</Chapter>
      <Chapter title="Papa">To me he was so wonderful</Chapter>
    </Part>
  </Book>
</Library>
```

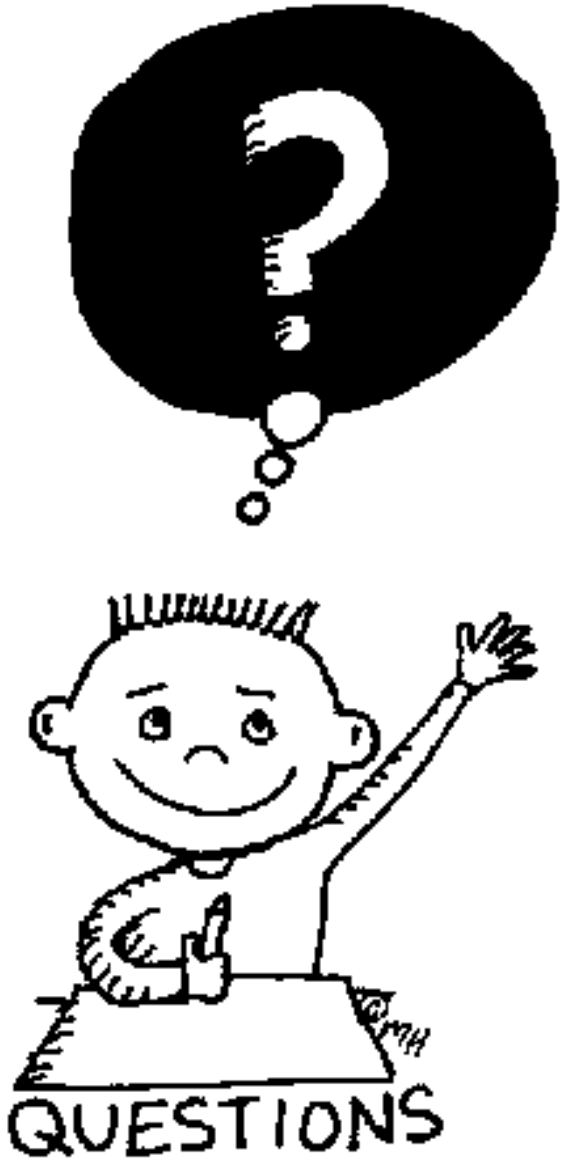
XPath expression:

```
/Library/Library/Book/Chapter//Chapter//Part[@title="Kilo"]/Chapter/de
```

Exercises

- XPath with libxml
  - `xmllint -xpath XPathExpression XMLfile`
  - There is also an interactive shell
    `xmllint -shell`
- XPath with Eclipse EE
  - XPath expressions are applied from the element on which the cursor is placed

1. Try the examples on **library.xml**
2. There are several, interesting exercises at
   `http://learn.onion.net/language=en/35426/w3c-xpath`



END OF THE LECTURE