XPATH

- XML is often compared to a database because of the way it packages information for easy retrieval. Element names and attributes put handles on data, just as SQL tables use table and column names. You can locate and extract any piece of information using the element names.
- First you can locate specific data from a known location (called a path) in a particular document.
- You can also use this path information to get really specific about processing a class of documents.

XPATH

- To express path information in a standard way, the W3C recommends the XML Path language (also known as XPath).
- XPath facilitates other technologies such as XSLT.
- In eclipse we can see the tree structure of an XML document, it comprises of nodes which are elements. Each elements children can be seen.
- There is a unique path from the root to any other point.
- XPath simply describes how to climb the tree in a series of steps to arrive at a destination.

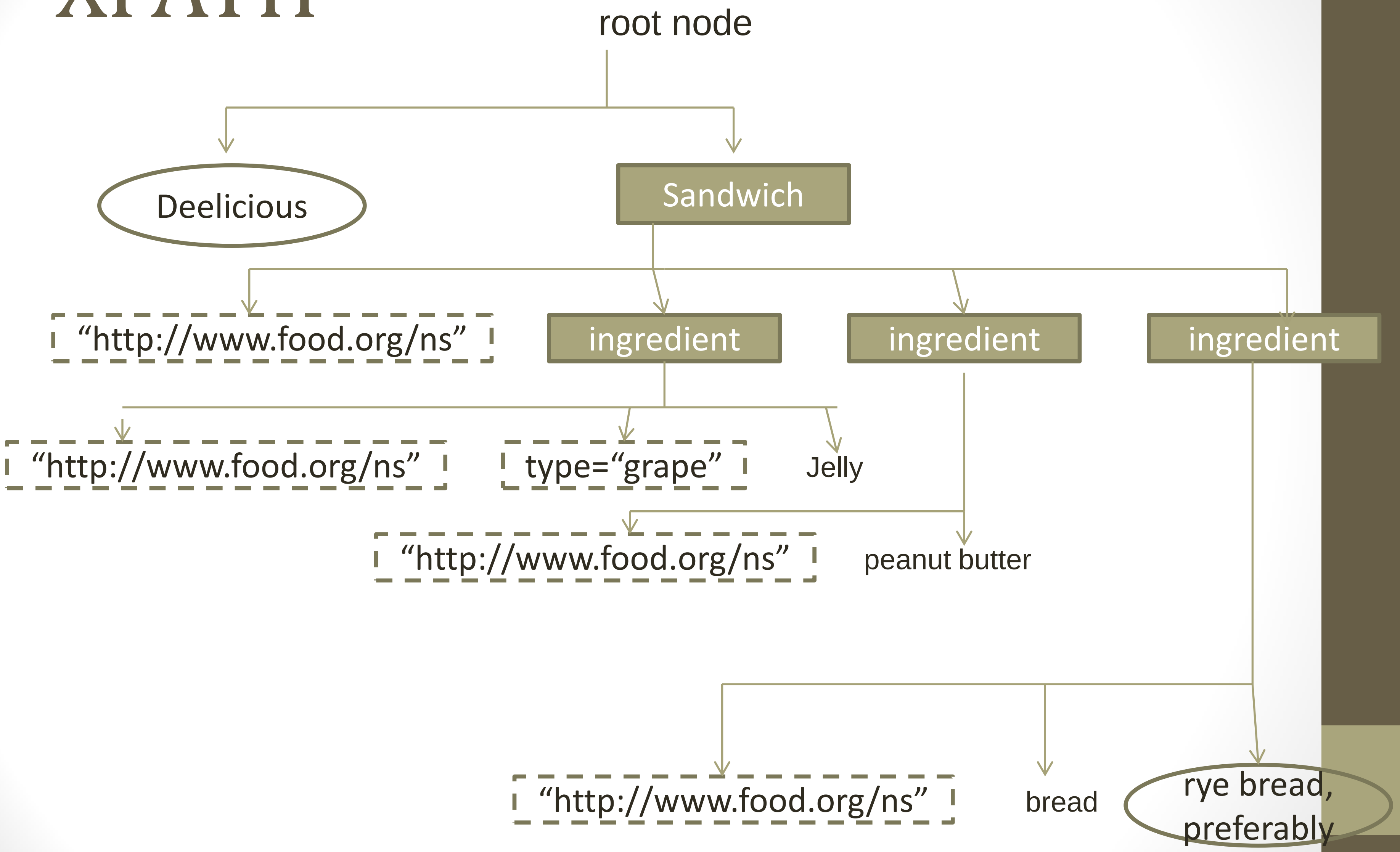
XPATH

- node types:
  - Root – contains the document element and any comments or processing instructions that surround the document element.
  - Element – Elements and the root node alone can contain other nodes.
  - Attribute – XPath treats attributes as seaparate nodes from their element host.
  - Text – lead node, always the child of an element.
  - Comment – also a node.
  - Namespace – also a node.

XPATH

```
<!-- Deelicious -->
<sandwich xmlns=http://www.food.org/ns>
  <ingredient type="grape"> Jelly </ingredient>
  <ingredient>peanut butter</ingredient>
  <ingredient>bread<!--rye bread, preferably --> </ingredient>
</sandwich>
```

XPATH


root node
Deelicious
Sandwich
"http://www.food.org/ns"
ingredient
ingredient
ingredient
"http://www.food.org/ns"
type="grape"
Jelly
"http://www.food.org/ns"
peanut butter
"http://www.food.org/ns"
bread
rye bread, preferably

XPATH

- Finding nodes:

  XPath uses a chain of steps in the XML tree.

  A location path is a chain of location steps that get you from one point in a document to another.

  If a location path begins with an absolute position (say the root node), then we call it an absolute path.

  Otherwise it is called a relative path because it starts from a place not yet determined.

XPATH

A location step has three parts: an axis that describes the direction to travel, a node test that specifies what kinds of nodes are applicable, and a set of optional predicates that use Boolean (true/false) tests to filter down the candidates further.

The axis is a keyword that specifies a direction you can travel from any node. You can go up through ancestors, down through descendants or linearly through siblings.

XPATH

Node Axes

Axis type	**Matches**
Ancestor	all nodes above the context, up to root.
Ancestor-or-self	ancestor node plus context node.
Attribute	attributes of the context node.
Child	children of the context node.
Descendant	children of the context plus descendents
Descendant-or-self	descendent nodes plus context node.
Following	nodes that follow the context node (not including context node descendents).
Following-sibling	nodes that follow the context node at the same level (have the same parent).

XPATH

Node Axes

Axis type	Matches
Namespace	All the namespace nodes of an element
Parent	the parent of the context node.
Preceding level in	nodes that occur before the context node at any the document (including preceding siblings).
Preceding-sibling	nodes that occur before the context node at the same level
Self	context node itself.

XPATH

After the axis comes the node test parameter, joined to the axis by a double colon (::)

Term	**Matches**
/	the root node (contains root element)
node()	matches any node
*	any attribute/namespace/element
crabcake	name of attribute/namespace/element
text()	any text node
processing-instruction()	any processing instruction
processing-instruction('for-web')	any instruction with target 'for web'
comment()	any comment node.

XPATH

Location paths are chained together using the slash (/) character. Each step moves you closer to the node you wish to locate.

For example to get from the root node to a para element inside a section inside a chapter inside a book, a path might look like this:

book/chapter/section/para

Or

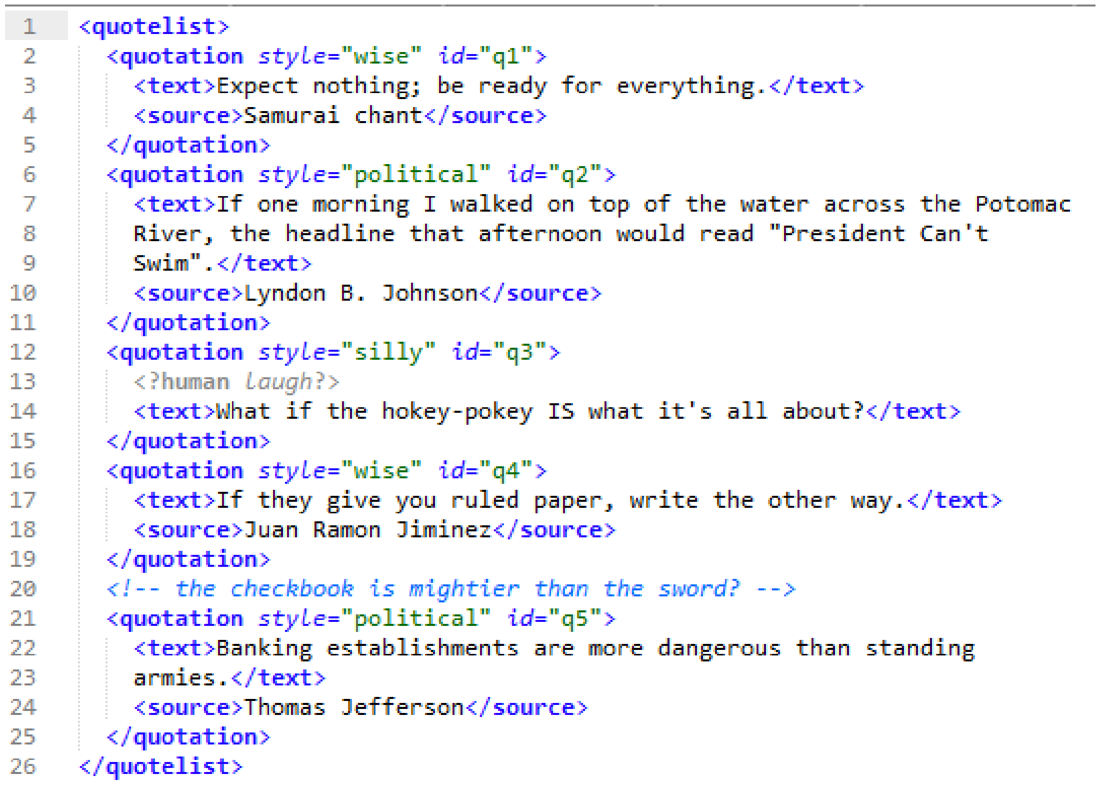
book/chapter/section/child::node()

XPATH

XPATH has some handy shortcuts:

@role	Matches an attribute named role (equivalent to attribute::role)
.	The context node, same as self::node()
/*	Matches doc element and then any element
..	Matches the parent node. Same as parent::node()
.//para	any element of type para that is descendent from current node.
//para	any para descending from root node

XPATH



```xml
<quotelist>
  <quotation style="wise" id="q1">
    <text>Expect nothing; be ready for everything.</text>
    <source>Samurai chant</source>
  </quotation>
  <quotation style="political" id="q2">
    <text>If one morning I walked on top of the water across the Potomac
    River, the headline that afternoon would read "President Can't
    Swim".</text>
    <source>Lyndon B. Johnson</source>
  </quotation>
  <quotation style="silly" id="q3">
    <?human laugh?>
    <text>What if the hokey-pokey IS what it's all about?</text>
  </quotation>
  <quotation style="wise" id="q4">
    <text>If they give you ruled paper, write the other way.</text>
    <source>Juan Ramon Jiminez</source>
  </quotation>
  <!-- the checkbook is mightier than the sword? -->
  <quotation style="political" id="q5">
    <text>Banking establishments are more dangerous than standing
    armies.</text>
    <source>Thomas Jefferson</source>
  </quotation>
</quotelist>
```

/quotelist/child::node()

All the quotation elements and the XML comment

/quotelist/quotation

All the quotation elements

/*/*

All the quotation elements

//comment()/following-sibling::*/style

Style attribute of the last quotation element

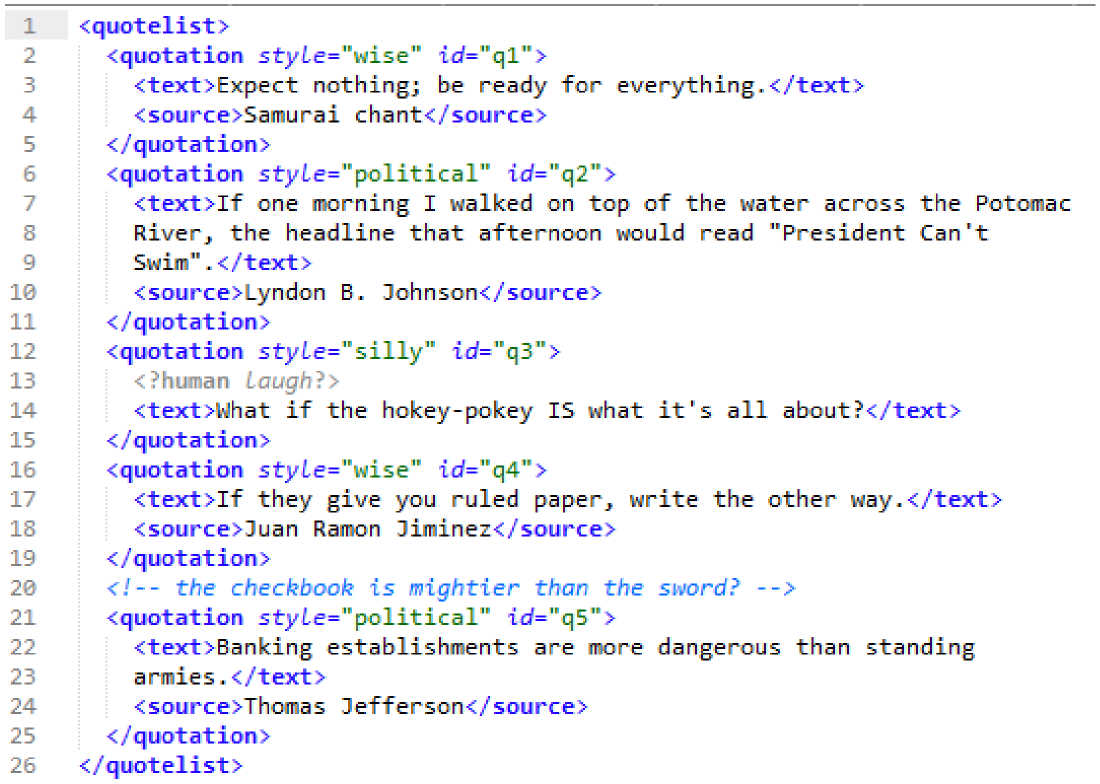
id('q1')/ancestor-or-self::*

The document element and the first quotation element

XPATH

- If the axis and node type aren’t sufficient to narrow down the selection, you can use one or more predicates. A predicate is a boolean expression enclosed within square brackets []
- Nodes that pass the test is included in the final node set, nodes that fail the test are not include.

XPATH



```
<quotelist>
  <quotation style="wise" id="q1">
    <text>Expect nothing; be ready for everything.</text>
    <source>Samurai chant</source>
  </quotation>
  <quotation style="political" id="q2">
    <text>If one morning I walked on top of the water across the Potomac
    River, the headline that afternoon would read "President Can't
    Swim".</text>
    <source>Lyndon B. Johnson</source>
  </quotation>
  <quotation style="silly" id="q3">
    <?human laugh?>
    <text>What if the hokey-pokey IS what it's all about?</text>
  </quotation>
  <quotation style="wise" id="q4">
    <text>If they give you ruled paper, write the other way.</text>
    <source>Juan Ramon Jiminez</source>
  </quotation>
  <!-- the checkbook is mightier than the sword? -->
  <quotation style="political" id="q5">
    <text>Banking establishments are more dangerous than standing
    armies.</text>
    <source>Thomas Jefferson</source>
  </quotation>
</quotelist>
```

//quotation[@id="q3"]/text

Text element in the third quote

//quotation[source]

All quote elements with a source

//quotation[not source]

The third quote element

//quotation[position()!=2]

All quotes but the second one

//quotation[4]

The fourth quote