Semi-structured Data

8 - XPath
Outline

• XPath Terminology

• XPath at First Glance

• Location Paths (Axis, Node Test, Predicate)

• Abbreviated Syntax

• Further Examples
What is XPath?

• A language for **extracting parts** of an XML document

• A basic **query language for XML** - plays the same role as the SQL SELECT statement plays for relational databases

• An important component of other XML-related technologies (such as XSD, XQuery and XSLT)

• As expected, XPath is a W3C standard
XPath Terminology

- XML documents are treated as *trees* of nodes - slightly different than DOM trees

- There are *seven kinds* of nodes:
  - Document nodes
  - Element nodes
  - Attribute nodes
  - Text nodes
  - Namespace nodes
  - Processing-instruction nodes
  - Comment nodes
XPath Terminology - Nodes

<?xml version="1.0"?>
<!-- DBAI -->
<?xml-stylesheet href="course_style.css" type="text/css"?>
<courses>
  <course semester="Summer">
    <title> Semi-structured Data (SSD) </title>
    <day> Thursday </day>
    <time> 09:15 </time>
    <location> HS8 </location>
  </course>
</courses>
Relationships Among Nodes

- The terms parent, child, sibling, ancestor and descendant are describing the relationships among nodes.

- In an XML tree:
  - The top node is the root
  - Every node has exactly one parent (except the root)
  - A node can have an unbounded number of children
  - A leaf node has no children
  - Siblings have the same parent
Relationships Among Nodes
Relationships Among Nodes

Element: `<courses>`

Element: `<course>`

Element: `<title>`

Element: `<day>`

Element: `<time>`

Element: `<location>`

child nodes to `<course>`

sibling nodes to each other
Relationships Among Nodes

Element: 
<courses>

Element: 
<course>

Element: 
<title>

Element: 
<day>

Element: 
<time>

Element: 
<location>

descendant nodes to <courses>
Relationships Among Nodes

Element: `<courses>`

Element: `<course>`

Element: `<title>`

Element: `<day>`

Element: `<time>`

Element: `<location>`

ancestor nodes to `<title>`, `<day>`, `<time>` and `<location>`
XPath at First Glance

ROOT (/)

Element:
<title>

Comment: DBAI

Processing instruction:
xml-stylesheet ...

Element:
<courses>

Attribute:
semester="Summer"

Element:
<course>

Element:
<title>
Text:
Semi-structured Data (SSD)

Element:
<day>
Text:
Thursday

Element:
<time>
Text:
09:15

Element:
<location>
Text:
HS8
XPath at First Glance

ROOT (/)

Comment: DBAI

Processing instruction: xml-stylesheet ...

Element: <courses>

Attribute: semester="Summer"

Element: <course>

Element: <title>

Text: Semi-structured Data (SSD)

Element: <day>

Text: Thursday

Element: <time>

Text: 09:15

Element: <location>

Text: HS8

/child::courses
XPath at First Glance

(ROOT (/))

- Comment: DBAI
- Processing instruction: xml-stylesheet ...
- Element: <courses>
  - Attribute: semester="Summer"
  - Element: <course>
    - Element: <title>
      - Text: Semi-structured Data (SSD)
    - Element: <day>
      - Text: Thursday
    - Element: <time>
      - Text: 09:15
    - Element: <location>
      - Text: HS8

/child::courses/child::course
XPath at First Glance

ROOT (/)

Comment: DBAI

Processing instruction: xml-stylesheet ...

Element: <courses>

Attribute: semester="Summer"

Element: <course>

Element: <title>

Text: Semi-structured Data (SSD)

Element: <day>

Text: Thursday

Element: <time>

Text: 09:15

Element: <location>

Text: HS8

/child::courses/child::course/child::title
XPath at First Glance

(ROOT (/)

- Comment: DBAI
- Processing instruction: xml-stylesheet...
- Element: <courses>
  - Attribute: semester="Summer"
  - Element: <course>
    - Element: <title>
      - Text: Semi-structured Data (SSD)
    - Element: <day>
      - Text: Thursday
    - Element: <time>
      - Text: 09:15
    - Element: <location>
      - Text: HS8

/descendant::course/child::title
XPath at First Glance

/descendant::course/child::*
XPath at First Glance

/root (/)

Comment: DBAI

Processing instruction: xml-stylesheet ...

Element: <courses>

Attribute: semester="Summer"

Element: <course>

Element: <title>

Text: Semi-structured Data (SSD)

Element: <day>

Text: Thursday

Element: <time>

Text: 09:15

Element: <location>

Text: HS8

/descendant::course/descendant::node()
XPath at First Glance

ROOT (/)

Comment: DBAI

Processing instruction: xml-stylesheet...

Element: <courses>

Attribute: semester="Summer"

Element: <course>

Element: <title>
Text: Semi-structured Data (SSD)

Element: <day>
Text: Thursday

Element: <time>
Text: 09:15

Element: <location>
Text: HS8

/descendant::course/descendant::text()
XPath at First Glance

Comment: DBAI
Processing instruction: xml-stylesheet...
Element: <courses>
Attribute: semester="Summer"
Element: <course>
Element: <title>
Text: Semi-structured Data (SSD)
Element: <day>
Text: Thursday
Element: <time>
Text: 09:15
Element: <location>
Text: HS8

/child::courses/child::course/attribute::semester
XPath at First Glance

ROOT (/)

Comment: DBAI

Processing instruction: xml-stylesheet …

Element: <courses>

Attribute: semester="Summer"

Element: <course>

Element: <title>
  Text: Semi-structured Data (SSD)

Element: <day>
  Text: Thursday

Element: <time>
  Text: 09:15

Element: <location>
  Text: HS8

/descendant::course/attribute::semester
Up to Now

• XPath Terminology

• XPath at First Glance

• Location Paths (Axis, Node Test, Predicate)

• Abbreviated Syntax

• Further Examples
Location Paths

- XPath uses **location paths** to select nodes in a tree.
- A location path is a series of **location steps** separated by the symbol `/`.
- Each location step has the form
  
  \[
  \text{axis::node-test[expression-1][expression-2]}...
  \]

  - Defines the relationship to be followed.
  - Defines what kind of nodes must be selected.
  - Zero or more predicates, which filter the selected nodes according to arbitrary selection criteria.
The Anatomy of a Location Path

child::courses/child::course[position() = 1]

-axis node-test axis node-test predicate

- location step
  - location path

**ATTENTION:** The first location step does not have a predicate
Axes

- XPath defines 13 axes:
  - ancestor
  - ancestor-or-self
  - attribute
  - child
  - descendant
  - descendant-or-self
  - following
  - following-sibling
  - namespace
  - parent
  - preceding
  - preceding-sibling
  - self
Axes

- XPath defines 13 axes:
  - ancestor
  - ancestor-or-self
  - attribute
  - child
  - descendant
  - descendant-or-self
  - following
  - following-sibling
  - namespace
  - parent
  - preceding
  - preceding-sibling
  - self

- Selects all the nodes that are ancestors of the origin node
- The first node on the axis is the parent of the origin, the second is its grandparent, and so on
- The last node on the axis is the root of the tree
Axes

- XPath defines 13 axes:
  - ancestor
  - \textbf{ancestor-or-self}
  - attribute
  - child
  - descendant
  - descendant-or-self
  - following
  - following-sibling
  - namespace
  - parent
  - preceding
  - preceding-sibling
  - self

- Selects the same nodes as the ancestor axis
- … but starting with the origin node (instead of the parent of the origin node)
Axes

- XPath defines 13 axes:
  - ancestor
  - ancestor-or-self
  - attribute
  - child
  - descendant
  - descendant-or-self
  - following
  - following-sibling
  - namespace
  - parent
  - preceding
  - preceding-sibling
  - self

- If the origin is an element node, then this axis selects all its attribute nodes; otherwise, it selects nothing (empty sequence)

- The attributes will not necessarily be in the order in which they appear in the document

- Namespace nodes are not selected
Axes

- XPath defines 13 axes:
  - ancestor
  - ancestor-or-self
  - attribute
  - child
  - descendant
  - descendant-or-self
  - following
  - following-sibling
  - namespace
  - parent
  - preceding
  - preceding-sibling
  - self

- Selects all the children of the origin in document order

- If the origin is other than a document or element node, then this axis selects nothing

- The children of an element node do not include attribute or namespaces
Axes

- XPath defines 13 axes:
  - ancestor
  - ancestor-or-self
  - attribute
  - child
  - descendant
  - descendant-or-self
  - following
  - following-sibling
  - namespace
  - parent
  - preceding
  - preceding-sibling
  - self

- Selects all the children of the origin, and their children, and so on recursively in document order
Axes

- XPath defines 13 axes:
  - ancestor
  - ancestor-or-self
  - attribute
  - child
  - descendant
  - descendant-or-self
  - following
  - following-sibling
  - namespace
  - parent
  - preceding
  - preceding-sibling
  - self

- Selects all the children of the origin, and their children, and so on recursively in document order
Axes

- XPath defines 13 axes:
  - ancestor
  - ancestor-or-self
  - attribute
  - child
  - descendant
  - descendant-or-self
  - following
  - following-sibling
  - namespace
  - parent
  - preceding
  - preceding-sibling
  - self

- Selects the same nodes as the descendant axis, except that the first node selected is the origin
Axes

- XPath defines 13 axes:
  - ancestor
  - ancestor-or-self
  - attribute
  - child
  - descendant
  - descendant-or-self
  - following
  - following-sibling
  - namespace
  - parent
  - preceding
  - preceding-sibling
  - self

- Selects all the nodes that appear after the origin in document order, excluding the descendants of the origin

- The following axis will never contain attributes or namespaces
Axes

- XPath defines 13 axes:
  - ancestor
  - ancestor-or-self
  - attribute
  - child
  - descendant
  - descendant-or-self
  - following
  - following-sibling
  - namespace
  - parent
  - preceding
  - preceding-sibling
  - self

- Selects all the nodes that follow the origin in document order, and that are children of the same parent

- For document, attribute and namespaces, this axis is empty
Axes

- XPath defines 13 axes:
  - ancestor
  - ancestor-or-self
  - attribute
  - child
  - descendant
  - descendant-or-self
  - following
  - following-sibling
  - namespace
  - parent
  - preceding
  - preceding-sibling
  - self

- If the origin is an element node, then this axis selects all the namespace nodes that are in the scope of that element; otherwise, it is empty.

- The namespaces will not necessarily be in the order in which they appear in the document.
Axes

- XPath defines 13 axes:
  - ancestor
  - ancestor-or-self
  - attribute
  - child
  - descendant
  - descendant-or-self
  - following
  - following-sibling
  - namespace
  - parent
  - preceding
  - preceding-sibling
  - self

- Selects the parent of the origin node (i.e., a single node)

- If the origin node does not have a parent, then the parent axis is empty
Axes

- XPath defines 13 axes:
  - ancestor
  - ancestor-or-self
  - attribute
  - child
  - descendant
  - descendant-or-self
  - following
  - following-sibling
  - namespace
  - parent
  - preceding
  - preceding-sibling
  - self

- Selects all the nodes that appear before the origin, excluding the ancestors of the origin node

- The preceding axis will never contain attributes or namespaces
Axes

- XPath defines 13 axes:
  - ancestor
  - ancestor-or-self
  - attribute
  - child
  - descendant
  - descendant-or-self
  - following
  - following-sibling
  - namespace
  - parent
  - preceding
  - preceding-sibling
  - self

- Selects all the nodes that precede the origin, and that are children of the same parent.

- For document, attribute and namespace nodes, this axis is empty.
Axes

- XPath defines 13 axes:
  - ancestor
  - ancestor-or-self
  - attribute
  - child
  - descendant
  - descendant-or-self
  - following
  - following-sibling
  - namespace
  - parent
  - preceding
  - preceding-sibling
  - self

- Selects the origin node

- This axis is always non-empty

- Usually, this axis is used in a node-test in order to test whether the current node pass that node-test
Location Paths

- XPath uses location paths to select nodes in a tree.
- A location path is a series of location steps separated by the symbol `/`.
- Each location step has the form `axis::node-test[expression-1][expression-2]…` where:
  - The first part defines the relationship to be followed.
  - The second part defines what kind of nodes must be selected.
  - Zero or more predicates, which filter the selected nodes according to arbitrary selection criteria.
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>node()</td>
<td>selects all nodes</td>
</tr>
<tr>
<td>text()</td>
<td>selects only text nodes</td>
</tr>
</tbody>
</table>
| name    | selects only elements nodes with tag “name”  
|         | …but, if it is used with the attribute axis (attribute::name), then it selects the “name” attribute nodes  
|         | …and if it is used with the namespace axis (namespace::name), then it selects the namespace nodes with prefix “name” |
| *       | selects all element nodes  
|         | …but, if it is used with the attribute axis (attribute::*), then it selects all the attribute nodes  
|         | …and if it is used with the namespace axis (namespace::*), then it selects all the namespace nodes |
Location Paths

- XPath uses location paths to select nodes in a tree.
- A location path is a series of location steps separated by the symbol `/`.
- Each location step has the form

\[\text{axis}::\text{node-test}[\text{expression-1}][\text{expression-2}]\ldots\]

- Defines the relationship to be followed.
- Defines what kind of nodes must be selected.
- Zero or more predicates, which filter the selected nodes according to arbitrary selection criteria.
Predicates

- A **qualifying expression** used to select a subset of the nodes in a sequence

- May be any XPath expression written in square brackets

- Each node of a sequence is kept only if the evaluation of the qualifier for this node returns true
Predicates: Examples

ROOT (/)

Comment: DBAI

Processing instruction: xml-stylesheet ...

Element: <courses>

Attribute: semester="Summer"

Element: <course>

Element: <title>
Text: Semi-structured Data (SSD)

Element: <day>
Text: Thursday

Element: <time>
Text: 09:15

Element: <location>
Text: HS8

/child::courses/child::course[position() = 1]
Predicates: Examples

```xml
<courses>
  <course semester="Summer">
    <title>Semi-structured Data (SSD)</title>
    <day>Thursday</day>
    <time>09:15</time>
    <location>HS8</location>
  </course>
</courses>
```

/child::courses/child::course[position() = last()]
Predicates: Examples

ROOT (/)

Comment: DBAI

Processing instruction: xml-stylesheet ...

Element: <courses>

Attribute: semester="Summer"

Element: <course>

Element: <title>

Text: Semi-structured Data (SSD)

Element: <day>

Text: Thursday

Element: <time>

Text: 09:15

Element: <location>

Text: HS8

/child::courses/child::course[position() = last()-1]

empty!!!
Predicates: Examples

```
/child::courses/child::course[position() < 3]
```
/child::courses/child::course[attribute::semester]
Predicates: Examples

/child::courses/child::course[attribute::semester = "Summer"]
Predicates: Examples

(ROOT (/)

Comment: DBAI

Processing instruction: xml-stylesheet …

Element: <courses>

Attribute: semester="Summer"

Element: <course>

Element: <title>

Text: Semi-structured Data (SSD)

Element: <day>

Text: Thursday

Element: <time>

Text: 09:15

Element: <location>

Text: HS8

empty!!!

/child::courses/child::course[attribute::semester = “Winter”]
Predicates: Examples

/child::courses/child::course[position() = 1][attribute::semester = "Summer"]
Semi-structured Data (SSD)

Thursday

09:15

HS8

/child::courses/child::course[attribute::*]
Predicates: Examples

RootElement (/)

Comment: DBAI

Processing instruction: xml-stylesheet ...

Element: <courses>

Attribute: semester="Summer"

Element: <course>

Element: <title>
  Text: Semi-structured Data (SSD)

Element: <day>
  Text: Thursday

Element: <time>
  Text: 09:15

Element: <location>
  Text: HS8

/child::courses/child::course[child::day = "Thursday"]
/child::courses/child::course[child::day = "Monday" or child::day = "Thursday"]
## XPath Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>`</td>
<td>`</td>
<td>Computes two node-sets</td>
</tr>
<tr>
<td><code>+</code></td>
<td>Addition</td>
<td><code>6 + 4</code></td>
</tr>
<tr>
<td><code>-</code></td>
<td>Subtraction</td>
<td><code>6 - 4</code></td>
</tr>
<tr>
<td><code>*</code></td>
<td>Multiplication</td>
<td><code>6 * 4</code></td>
</tr>
<tr>
<td><code>div</code></td>
<td>Division</td>
<td><code>8 div 4</code></td>
</tr>
<tr>
<td><code>mod</code></td>
<td>Modulus (division remainder)</td>
<td><code>5 mod 2</code></td>
</tr>
<tr>
<td><code>=</code></td>
<td>Equal</td>
<td><code>A = 9.80</code></td>
</tr>
<tr>
<td><code>!=</code></td>
<td>Not equal</td>
<td><code>A != 9.80</code></td>
</tr>
<tr>
<td><code>&lt;</code></td>
<td>Less than</td>
<td><code>A &lt; 9.80</code></td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>Less than or equal to</td>
<td><code>A &lt;= 9.80</code></td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>Greater than</td>
<td><code>A &gt; 9.80</code></td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>Greater than or equal to</td>
<td><code>A &gt;= 9.80</code></td>
</tr>
<tr>
<td><code>or</code></td>
<td>Logical OR</td>
<td><code>A = 9.80 or A = 9.70</code></td>
</tr>
<tr>
<td><code>and</code></td>
<td>Logical AND</td>
<td><code>A &gt; 9.00 and A &lt; 9.90</code></td>
</tr>
</tbody>
</table>
Location Paths

- XPath uses location paths to select nodes in a tree
- A location path is a series of location steps separated by the symbol /
- Each location step has the form

```
axis::node-test[expression-1][expression-2]...
```

- Defines the relationship to be followed
- Defines what kind of nodes must be selected
- Zero or more predicates, which filter the selected nodes according to arbitrary selection criteria
Up to Now

- XPath Terminology
- XPath at First Glance
- Location Paths (Axis, Node Test, Predicate)
- Abbreviated Syntax
- Further Examples
Abbreviated Syntax

- The most commonly used location steps can be in an abbreviated syntax
- Simplify XPath expressions

| Path Expression                  | Abbreviation
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/descendant-or-self::node()</code></td>
<td>//</td>
</tr>
<tr>
<td><code>self::node()</code></td>
<td>.</td>
</tr>
<tr>
<td><code>parent::node()</code></td>
<td>..</td>
</tr>
<tr>
<td><code>child::</code></td>
<td></td>
</tr>
<tr>
<td><code>attribute::</code></td>
<td>@</td>
</tr>
<tr>
<td><code>position() = n</code></td>
<td>n</td>
</tr>
</tbody>
</table>
Abbreviated Syntax: Examples

/child::courses/child::course[position() = 1]

/courses/child::course[position() = 1]

/courses/course[position() = 1]

/courses/course[1]
Abbreviated Syntax: Examples

/child::courses/child::course[attribute::semester]

/courses/child::course[attribute::semester]

/courses/course[attribute::semester]

/courses/course[@semester]
Abbreviated Syntax: Examples

/child::courses/child::course[position() = 1][attribute::semester = "Summer"]

/courses/child::course[position() = 1][attribute::semester = "Summer"]

/courses/course[position() = 1][attribute::semester = "Summer"]

/courses/course[1][attribute::semester = "Summer"]

/courses/course[1][@semester = "Summer"]
Abbreviated Syntax: Examples

/\texttt{descendant-or-selfnode::node()}/\texttt{child::course[position() = 1]}
\begin{itemize}
  \item [\texttt{attribute::semester = "Summer"}]
\end{itemize}

//\texttt{child::course[position() = 1][attribute::semester = "Summer"]}

//\texttt{course[position() = 1][attribute::semester = "Summer"]}

//\texttt{course[1][attribute::semester = "Summer"]}

//\texttt{course[1][@semester = "Summer"]}
Up to Now

- XPath Terminology
- XPath at First Glance
- Location Paths (Axis, Node Test, Predicate)
- Abbreviated Syntax
- Further Examples
Further Examples

//box/item
Further Examples

//box/item

items in a box
Further Examples

//box/item[@color="yellow"]
Further Examples

//box/item[@color="yellow"]

yellow items in a box
Further Examples

//box[item]
Further Examples

//box[item]  note the difference with //box/item

boxes with items
Further Examples

//box[item[@color="yellow"]]
Further Examples

//box[item[@color="yellow"]]

boxes with yellow items
Further Examples

```html
//item[1]
```
Further Examples

//item[1]

items in a box that appear first
Further Examples

/descendant::item[1]
Further Examples

```
/ancestor::box[1]  /* Note the difference with //box[1] */
```

the first item in the tree
Further Examples

```
(ROOT
  boxes
    box
      item color="blue"
      item color="green"
    box
      item color="yellow"
    box
      item color="blue"
      item color="yellow"
)

//item[1][@color="blue"]
```
Further Examples

//item[1][@color="blue"]

items in a box that appear first and are also blue
Further Examples

//item[@color="blue"][1]
Further Examples

//item[@color="blue"][1]

first blue items in a box

note the difference with

//item[1][@color="blue"]
Sum Up

• XPath Terminology

• XPath at First Glance

• Location Paths (Axis, Node Test, Predicate)

• Abbreviated Syntax

• Further Examples