Intelligent Internet Technologies

Lecture 11.

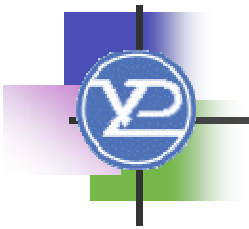
Other XML Technologies:

XQuery, Xlink, XPointer, XInclude

Alexandra V. Vitko

KNURE, AI Department, alexandra_vitko@yahoo.com

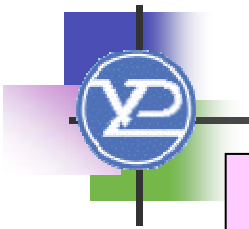
XML Query Laguage



- XML Query, XQuery

"The mission of the XML Query project is to provide flexible query facilities to extract data from **real and virtual documents** on the World Wide Web, thereby **finally providing the needed interaction between the Web world and the database world**. The ambitious task of the XML Query (XQuery) Working Group is to develop the first world standard for querying Web documents..."

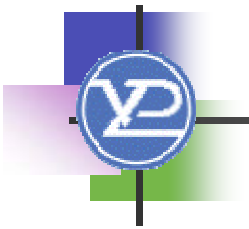
- W3C Candidate Recommendation
  http://www.w3.org/XML/Query



XML Query

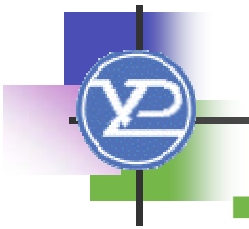
- XQuery is to XML what SQL is to databases

- Built on XPath expressions
  - XPath is a syntax for defining parts of an XML document: Path expressions, Axes and Node Tests, and Predicates
  - XPath defines a library of standard functions: Node Set Functions, String Functions, Number Functions, Boolean Functions
  - XPath supports numerical, equality, relational, and Boolean expressions.



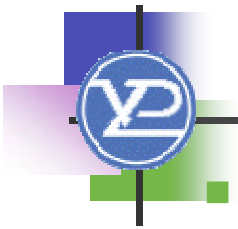
Overview of XQuery

- Purely functional language
  - may access elements from documents, may construct new values (elements), but cannot modify data
  - any expression is a query
  - query nesting is allowed at any place and on any level
- Strongly and statically typed
  - both type checking and type inference
- Has formal semantics based on the XML abstract data model



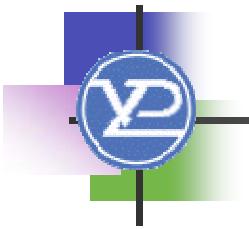
XQuery Expressions

- Element construction:
  - <tag> ... </tag>
- XPath expressions start from a root: document("URL")
  - document("book.xml")/chapter[10]
- Arithmetic operators; comparisons; set operators: union, intersect, except; aggregation: count, sum, avg, min, max, etc.
- You can define functions



Basic Query

- FLWR (pronounced "flower") expression is the analogue of the SELECT-FROM-WHERE construction in SQL:
  - **FOR**-clause: binds one or more variables to a sequence of values returned by another expression (usually a path expression) and iterates over the values.
  - **LET**-clause: also binds one or more variables but without iterating.
  - **WHERE**-clause: contains one or more predicates that filters or limits the set of nodes as generated by the FOR/LET-clauses.
  - **RETURN**-clause: generates the output of the FLWR expression. The RETURN-clause usually contains one or more element constructors and/or references to variables and is executed once for each node-reference that is returned by the FOR/LET/WHERE-clauses.



A Simple Query

- Example Basic Query:

```
FOR $b IN document("books.xml")//book
   WHERE $b/publisher/text() = "Springer"
   AND $b/@year = "2006"
RETURN $b/title
```

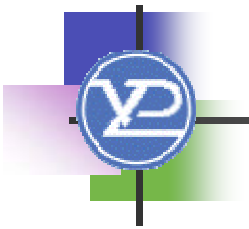
A Simple Example

```
<books>{
    for $b in document('books.xml')//book
    where $b/author/firstname = 'John'
      and $b/author/lastname = 'Smith'
    return <book>{
              $b/title,
              $b/price
           }</book>
}</books>
```

May return:

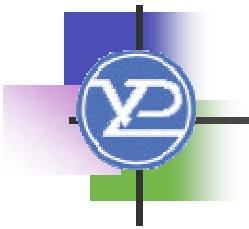
```
<books>
  <book><title>XML</title><price>29.99</price></book>
  <book><title>DOM and SAX</title><price>40</price></book>
</books>
```

Another Example



```
<best_students>{
  let $d := document('depts.xml')//department[name='cs']
  for $s in $d//gradstudent
  where $s/gpa > 3.5
  return <student>{
            $s/name,$s/gpa,
            count($d//gradstudent)
         }</student>
}</best_students>
```

XML Linking Language (XLink)



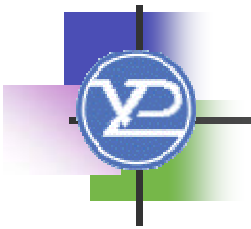
- **Description**

  allows elements to be inserted into XML documents in order to create and describe links between resources. It uses XML syntax to create structures that can describe the simple unidirectional hyperlinks of today's HTML, as well as more sophisticated links.

- **W3C Status**

  Recommendation 1.0, June 27th 2001

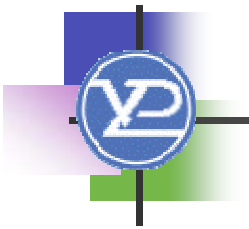
  http://www.w3.org/TR/xlink/



Xlink Overview

- A namespace (usually prefixed as "xlink") with a set of elements and attributes anchored in that namespace http://www.w3.org/1999/xlink/namespace/
- Two kinds of links:
  - Simple link elements (HTML like)
  - Extended links
- Well-Formed XML

XLink attributes



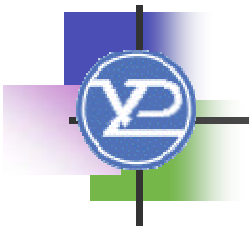
- **XLinks described with attributes**

```
<!ELEMENT Authors>
    <!ATTLIST Authors
        xmlns:xlink     CDATA #FIXED http://www.w3c.org/1999/xlink
        xlink:type      (simple | extended | locator | arc | resource | title) “simple”
        xlink:href      CDATA  #REQUIRED
        xlink:role      CDATA #IMPLIED
        xlink:title     CDATA #IMPIED
        xlink:show      (new | replace | embed | other | none ) “replace”
        xlink:actuate   (onLoad | onRequest | other | none ) “onRequest” >
```

XLink attributes

Category	Attribute	Description
	type	Describes the type of link. Possible values are simple, extended, locator, resource, arc, title, none.
	href	Supplies the location (URI) to allow the XLink application to find a remote resource
Semantic	role	URI reference describes the intended property of the XLink
	arcrole	URI reference describes the intended property of the arc
	title	Describes the meaning of the link in human readable terms
Behavioral	show	Describes the desired presentation of the data, possible values are new, replace, embed, other, or none.
	actuate	Describes the desired timing of the link. Possible values are onLoad, onRequest, other, or none.
Traversal	label	Provides and identifier that can be used to identify how different types of resources and locators are connected within a link
	from	Identifies the origin, using a label value from one of the locator or resource elements of an extended link
	to	Identifies the destination, using a label value from one of the locator or resource elements of an extended link

“type” Attribute

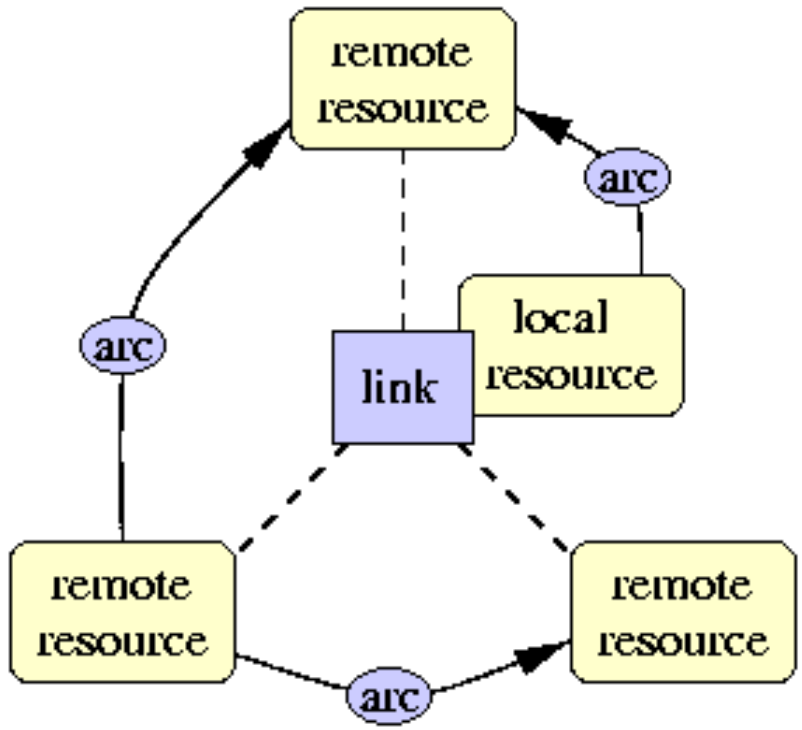


- Type (required) defines the type of link:
  - simple (Similar to HTML <A HREF=....>)
  - extended (allows multiple links)

- Sub-elements to extended links
  - locator Locations to participate in an extended link
  - arc Define navigable connections between locations resources
  - resource Define participants in a link

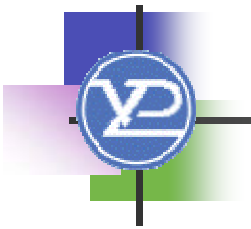
XML Linking Model

- XLink allows definition of Links
- XLink is simply an association between a number of resources (Locators)
- Link does not imply traversal
- Given link may contain a number of different arcs



XLink separates associating resources (using a link) from traversal amongst resources (using arcs)

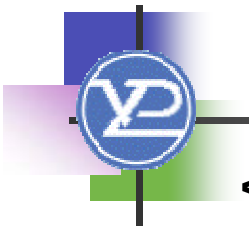
Association



```
<siblings xlink:type="extended">
 <child xlink:type="locator"
 xlink:href="people.xml#xpointer(id('anna'))"
 xlink:title="Anna"/>
 <child xlink:type="locator"
 xlink:href="people.xml#xpointer(id('bill'))"
 xlink:title="Bill"/>
 <child xlink:type="locator"
 xlink:href="people.xml#xpointer(id('carl'))"
 xlink:title="Carl"/>
</siblings>
```

- Three locator elements, each of which uses an XPointer as part of the locator
- No traversal between these elements specified.

Multiple Resources - Multiple Traversal

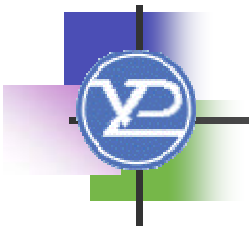


```
<extendedlink xlink:type="extended">
    <loc xlink:type="locator" xlink:href="a.xml" xlink:label="x"/>
    <loc xlink:type="locator" xlink:href="b.xml" xlink:label="y"/>
    <loc xlink:type="locator" xlink:href="c.xml" xlink:label="z"/>
    <loc xlink:type="locator" xlink:href="d.xml" xlink:label="z"/>
    <loc xlink:type="locator" xlink:href="e.xml" xlink:label="z"/>

    <go xlink:type="arc" xlink:from="x" xlink:to="y"/>
    <go xlink:type="arc" xlink:from="y" xlink:to="z"/>
</extendedlink>
```

- Two arc specification
- Second arc specification actually creates three arcs
- End up with one arc from a.xml (to b.xml), and three arcs from b.xml (to c.xml, d.xml, and e.xml).

Multi-Ended

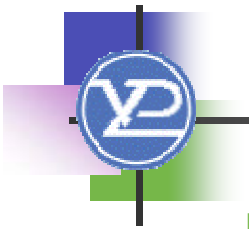


```
<family xlink:type="extended">
        <loc xlink:type="locator" xlink:label="parent" xlink:href="Ann.xml"/>
        <loc xlink:type="locator" xlink:label="parent" xlink:href="Bob.xml"/>
        <loc xlink:type="locator" xlink:label="child" xlink:href="Gina.xml"/>
        <loc xlink:type="locator" xlink:label="child" xlink:href="Hank.xml"/>
        <loc xlink:type="locator" xlink:label="child" xlink:href="Irma.xml"/>

        <go xlink:type="arc" xlink:from="parent" xlink:to="child"/>
</family>
```

- Five participating resources
- Single arc specification which results in six arcs (Ann–Gina, Ann–Hank, Ann–Irma, Bob–Gina, Bob–Hank, Bob–Irma).

XML Pointer Language (XPointer)

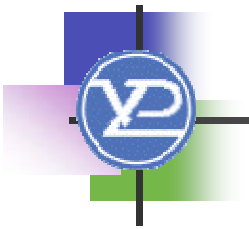


- **Description**
  - the language to be used as a fragment identifier for any URI-reference that locates a resource of Internet media type text/xml or application/xml.

- **W3C Status**

  Recommendation, published 25 March 2003

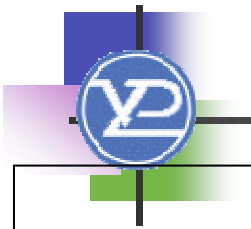
  http://www.w3.org/TR/xptr/



What is XPointer?

- an extension of XPath suited for linking
- specifies connection between XPath expressions and URIs
- Related to, but much more powerful than location specification of HTML.

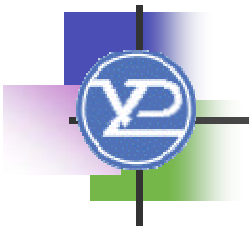
Xpointer example



xlink:href="books.xml#xpointer(/chapter[3]/topic[@name="XPath"])"

Match: <chapter>
…
</chapter>
<chapter>
…
</chapter>
<chapter>
<topic name="XPath"> …</topic>
</chapter>

XML Inclusions (XInclude)



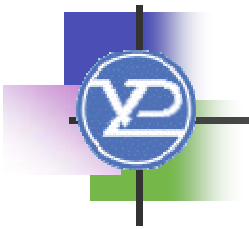
- **Description**

  a processing model and syntax for general purpose inclusion of XML documents.

- **W3C Status**

  Recommendation 15 November 2006

  http://www.w3.org/TR/xinclude/

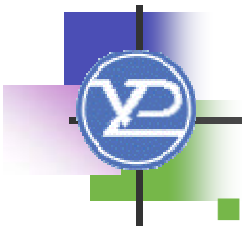


Elements and attributes

- XInclude standard defines two elements - **include** and **fallback**, and six attributes - **href**, **parse**, **xpointer**, **encoding**, **accept**, and **accept-language**.

- The elements are from the namespace **http://www.w3.org/2001/XInclude**.

Attribute href



- **Attribute href**: The **href** attribute of the **include** element tells the processor which document fragment(s) should be included.

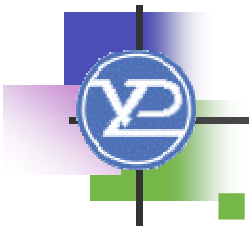
The file **a.xml** includes whole file **b.xml** as XML.

File **a.xml**:

```
<a>
   <xi:include href="b.xml"
   xmlns:xi="http://www.w3.org/2001/XInclude"/>
</a>
```

File **b.xml**: **<b/>**

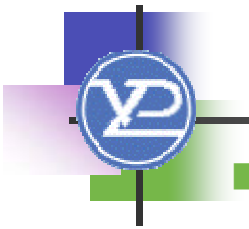
File **a.xml** after processing: <a> **<b/>**



Attribute parse = xml

- Indicates whether to include the resource as parsed XML or as text.
- If the **parse** attribute of the **include** element is set to 'xml', the included file (or its fragment) will be parsed (and included) as XML.

- See previous example



Attribute parse = text

- If the **parse** attribute of the **include** element is set to 'text', the included file (or its fragments) will be included as sequence of characters.

The file **a.xml** includes whole file **b.xml**.

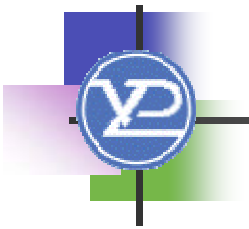
File **a.xml**:

```
<a>
<xi:include href="b.xml" parse="text"
   xmlns:xi="http://www.w3.org/2001/XInclude"/>
</a>
```

File **b.xml**: **<b/>**

File **a.xml** after processing: <a> **<b/>**

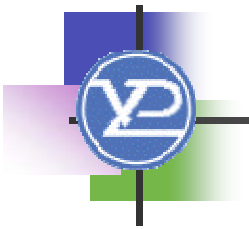
Attribute encoding and parse="text"



- If the **parse** attribute of the **include** element is set to 'text', you may sometimes need to specify the encoding of the included document.

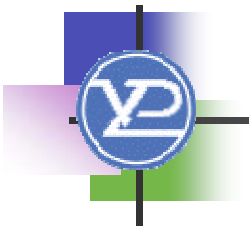
```
<a>
<xi:include href="b.txt" parse="text"
   encoding="iso-8859-1"
   xmlns:xi="http://www.w3.org/2001/XInclude"/>
</a>
```

Attribute encoding and parse="xml"



- In this case, the attribute **encoding** has no effect. XML files carry their encoding with them.

Including only a part of the file (parse = xml) - attribute xpointer



- We can also include only a fragment of the file, using XPointer

The file **a.xml** includes a paragraph with attribute id = 'p1' from file **b.xml**.

File **a.xml**:

```
<a> <xi:include href="b.xml" xpointer="p1" parse="xml"
    xmlns:xi="http://www.w3.org/2001/XInclude"/>
</a>
```

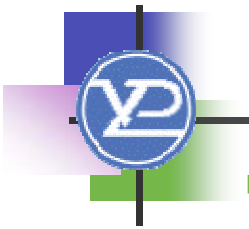
File **b.xml**:

```
<b> <p id="p1">aaaa</p> <p id="p2">bbbb</p> </b>
```

File **a.xml** after processing:

```
<a> <p id="p1">aaaa</p> </a>
```

Element fallback



- Element **fallback** tells the processor, what to do in case of **resource error** (a failure of an attempt to fetch a resource from a URL) - non-existent file, bad connection, etc. When this happens, **include** element is replaced with the contents of the **fallback** element. The **fallback** must be a child of the **include** element. It can contain anything.

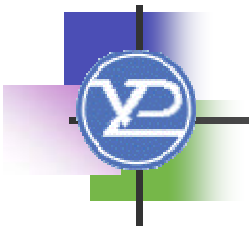
The file **a.xml** tries to include non-existent file.

File **a.xml**:

```
<a> <xi:include href="non-existent.xml"
   xmlns:xi="http://www.w3.org/2001/XInclude">
   <xi:fallback>There are no files today ...
   </xi:fallback> </xi:include> </a>
```

File **a.xml** after processing:

<a> **There are no files today ...**



Read More in

World Wide Web Consortium

http://www.w3.org

www.xmlsoftware.com