Application Design Using Java

Lecture 16

White Space in XML

- Characters
  - Carriage return (\r or \u000d)
  - Newline (\n or \u000a)
  - Tab (\t or \u0009)
  - Space (‘ ‘ or \u0020)
- Types
  - Significant
  - Insignificant

What Is an XML Vocabulary?

- Problem: two XML sources cannot be easily combined

```
<Inventory>
  <Product SKU="1">Goo</Product>
</Inventory>
```

+

```
<StockListing>
  <Item Number="2">Glob</Item>
</StockListing>
```

= ?

- Solution: each XML source shares the same structure, naming, and case

```
<Inventory>
  <Product SKU="1">Goo</Product>
</Inventory>
```

+

```
<Inventory>
  <Product SKU="2">Glob</Product>
</Inventory>
```

=

```
<Inventory>
  <Product SKU="1">Goo</Product>
  <Product SKU="2">Glob</Product>
</Inventory>
```

Guidelines for Creating Vocabularies

- Determine if a vocabulary already exists
- Make your XML easy to read and understand
- Use title case for elements and attribute names
- Avoid abbreviations

```
<docprops>
  <create>2012-07-24T22:39:55Z</create>
  <lsavd>2012-08-10T16:13:07Z</lsavd>
  <comp>Rensselaer Polytechnic Institute</comp>
  <ver>10.2625</ver>
</docprops>
```

```
<!-- Which of these makes better sense? -->
```

```
<DocumentProperties>
  <Created>2012-07-24T22:39:55Z</Created>
  <LastSaved>2012-08-10T16:13:07Z</LastSaved>
  <Company>Rensselaer Polytechnic Institute</Company>
  <Version>10.2625</Version>
</DocumentProperties>
```

What Is a Namespace?

- A namespace is a collection of element names identified by a unique reference
- Namespaces prevent confusion when combining data from multiple XML sources

What is the problem here?

```
<Order>
  <Employee>
       <Name>Jane Doe</Name>
       <Title>Developer</Title>
  </Employee>
  <Product>
       <Title>The Joshua Tree</Title>
       <Artist>U2</Artist>
  </Product>
</Order>
```

How to Use Default Namespaces

- A default namespace associates a URI to an element and all child elements

```
<ElementName xmlns="URI">
```

- Default namespaces allow you to combine XML fragments from separate sources without changing the XML structure

```
<Order>
 <Employee xmlns="http://hrweb">
     <Name>Jane Doe</Name>
     <Title>Developer</Title>
 </Employee>
  <Product xmlns="http://market">
     <Title>The Joshua Tree</Title>
     <Artist>U2</Artist>
  </Product>
</Order>
```

These names belong to the http://hrweb namespace.

These belong to the http://market namespace.

How to Use Explicit Namespaces

- An explicit namespace associates a prefix with a URI
- You can then use that prefix to mark specific elements as belonging to that namespace

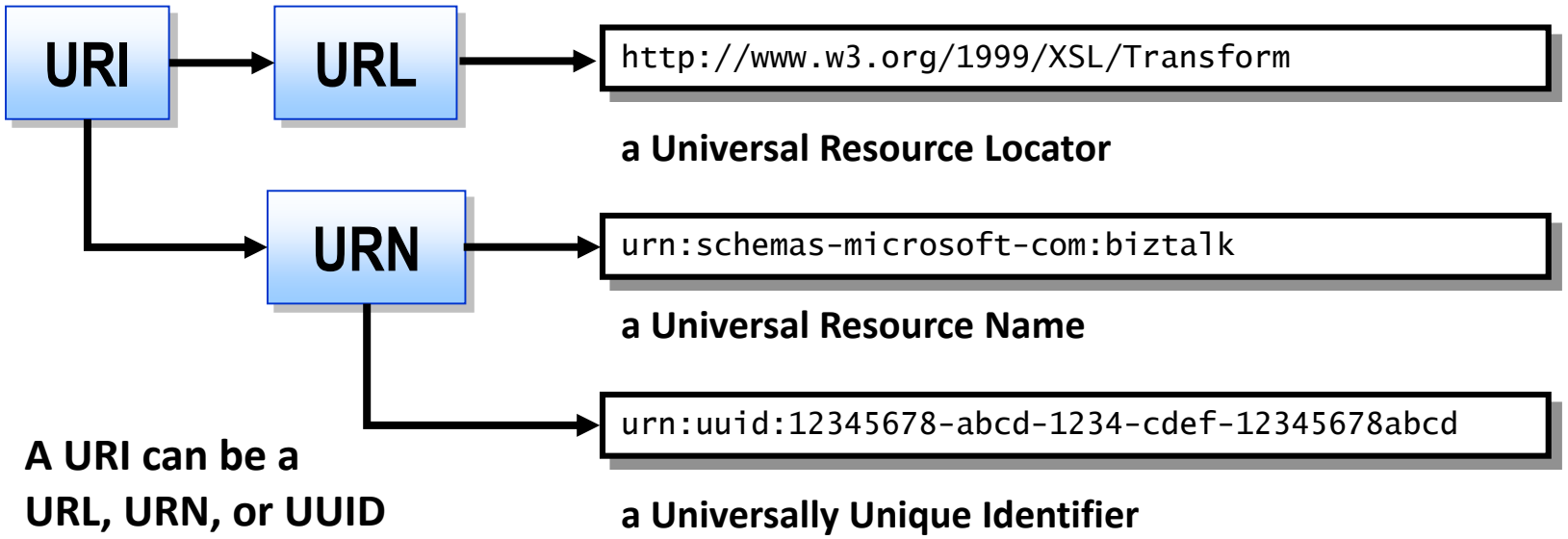
```
<ElementName xmlns:Prefix="http://contoso.msft/namespace_for_examples">
        <Prefix:AnyElement>Some Data</Prefix:AnyElement>
        <AnotherElement>More Data</AnotherElement>
```

- Explicit namespaces allow you to combine XML fragments into a new XML structure

```
<Order xmlns:hr="http://hrweb" xmlns:mkt="http://market">
  <hr:Name>Jane Doe</hr:Name>
  <hr:Title>Developer</hr:Title>
  <mkt:Title>The Joshua Tree</mkt:Title>
  <mkt:Artist>U2</mkt:Artist>
</Order>
```

Namespace URIs

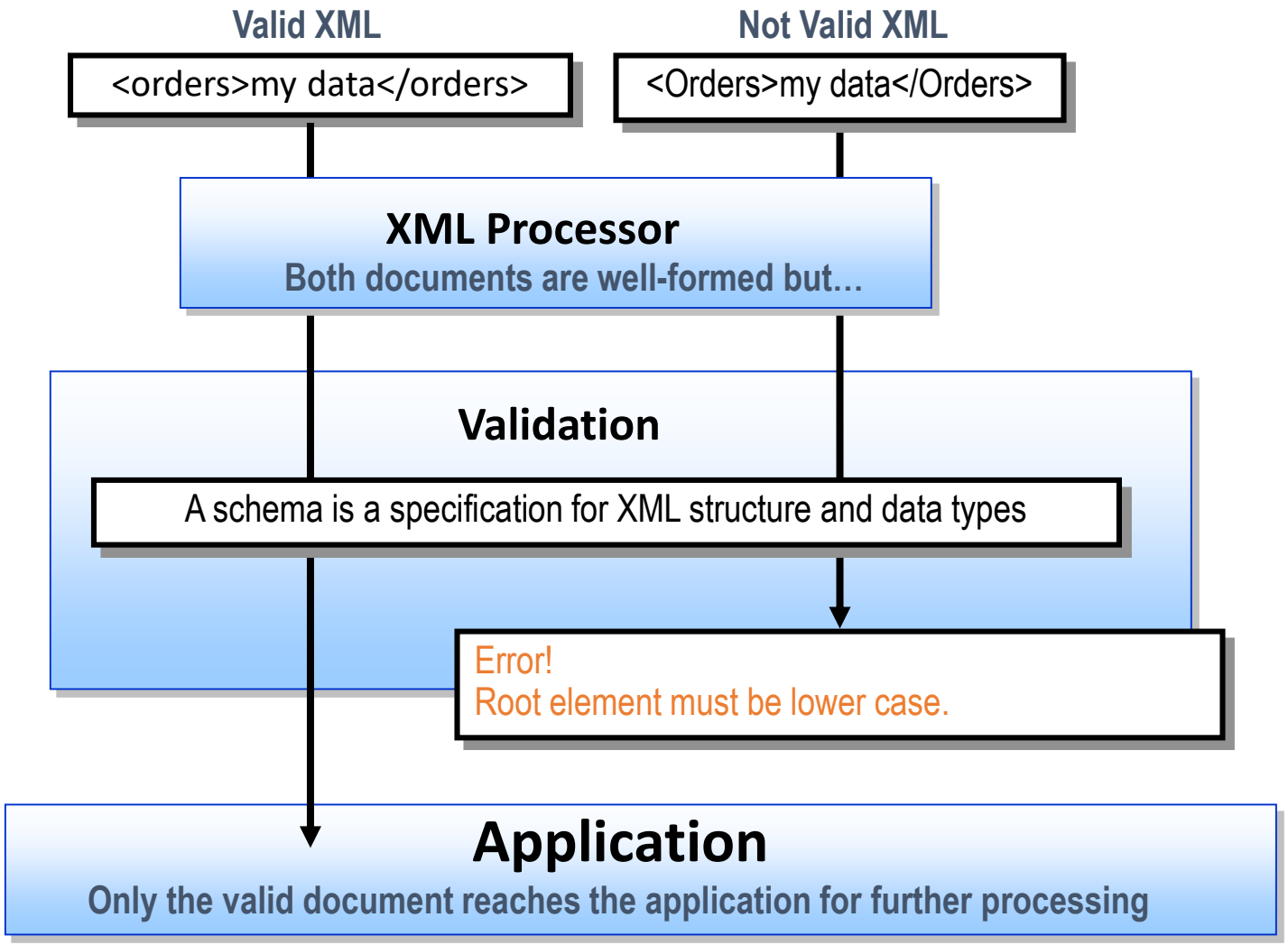
- The Uniform Resource Identifier (URI) uniquely identifies the namespace
- The XML processor does not verify the uniqueness or independent existence of a URI



Guidelines for Choosing a Namespace URI

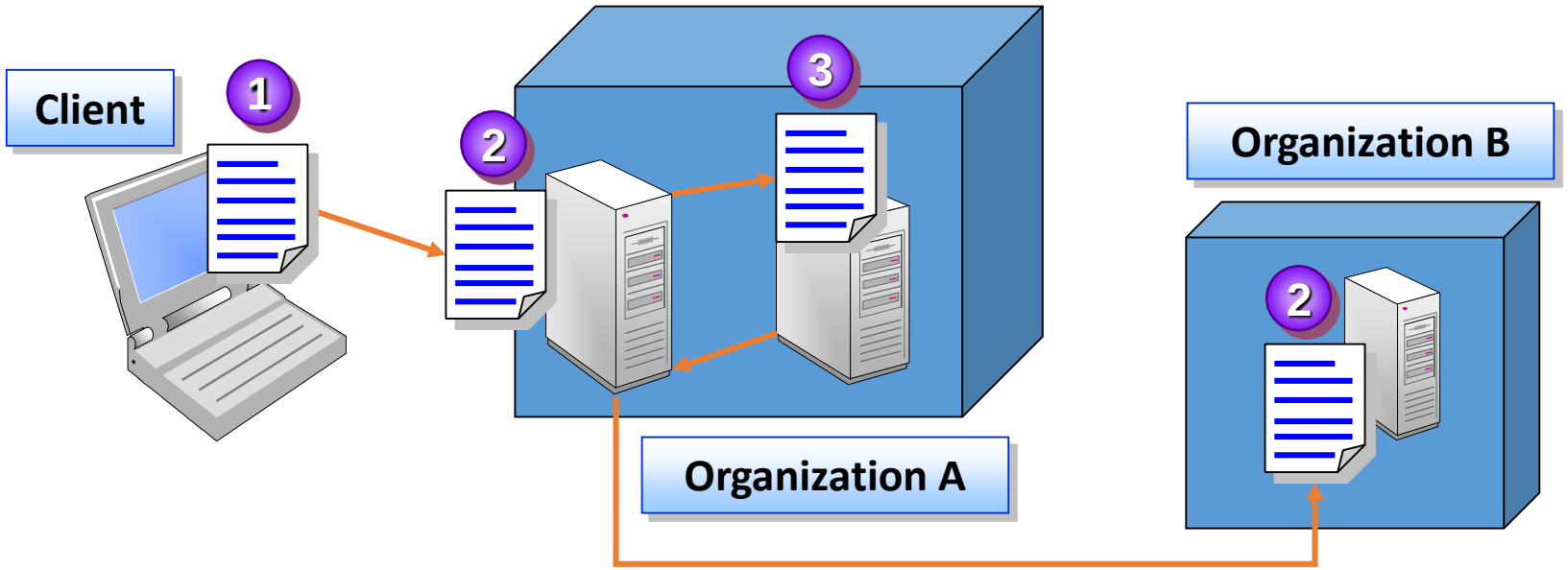
- Use URIs that you control
- Use URIs that are persistent
- Use URIs that consistently point to the same location
- Identify and describe the URI in the documentation for your XML vocabulary

What Is Valid XML?



Validation Scenarios

- Data transferred between systems requires validation



1. Validate user input prior to uploading.
2. Validate incoming XML against a required vocabulary.
3. Validate details against business logic prior to processing.

How to Recognize a Document Type Definition

- DTDs are superceded by XSD schemas, but you might still need to work around them
- A reference to an external DTD

```
<?xml version='1.0'?><!DOCTYPE bookstore
   SYSTEM "books.dtd">
```

- An external DTD

```
<!ELEMENT bookstore (book)*>
<!ELEMENT book (title,author*,price)>
<!ATTLIST book genre CDATA #REQUIRED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (name | (first-name,last-name))>
<!ELEMENT price (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT first-name (#PCDATA)>
<!ELEMENT last-name (#PCDATA)>
```

- An inline DTD

```
<?xml version='1.0'?>
<!DOCTYPE bookstore [
<!ELEMENT bookstore (book)*>
```

How to Recognize an XDR Schema

- External XDR schema

```
<Schema name="MySchema" xmlns="urn:schemas-microsoft-com:xml-data"
   xmlns:dt="urn:schemas-microsoft-com:datatypes">
```

- Inline XDR schema

```
<root xmlns:data="x-schema:#inLineSchema">
   <Schema name ="inLineSchema“ xmlns="urn:schemas-microsoft-com:xml-data">
   <ElementType name="myelement"/>
   </Schema>
   <data:myelement>inline example</data:myelement>
</root>
```

- Reference to an XDR schema in a source document

```
<ElementName xmlns="x-schema:yourschema.xml"> </ElementName>
```

Parts of an XSD Schema

- XSDs reference the W3C XML Schema namespace

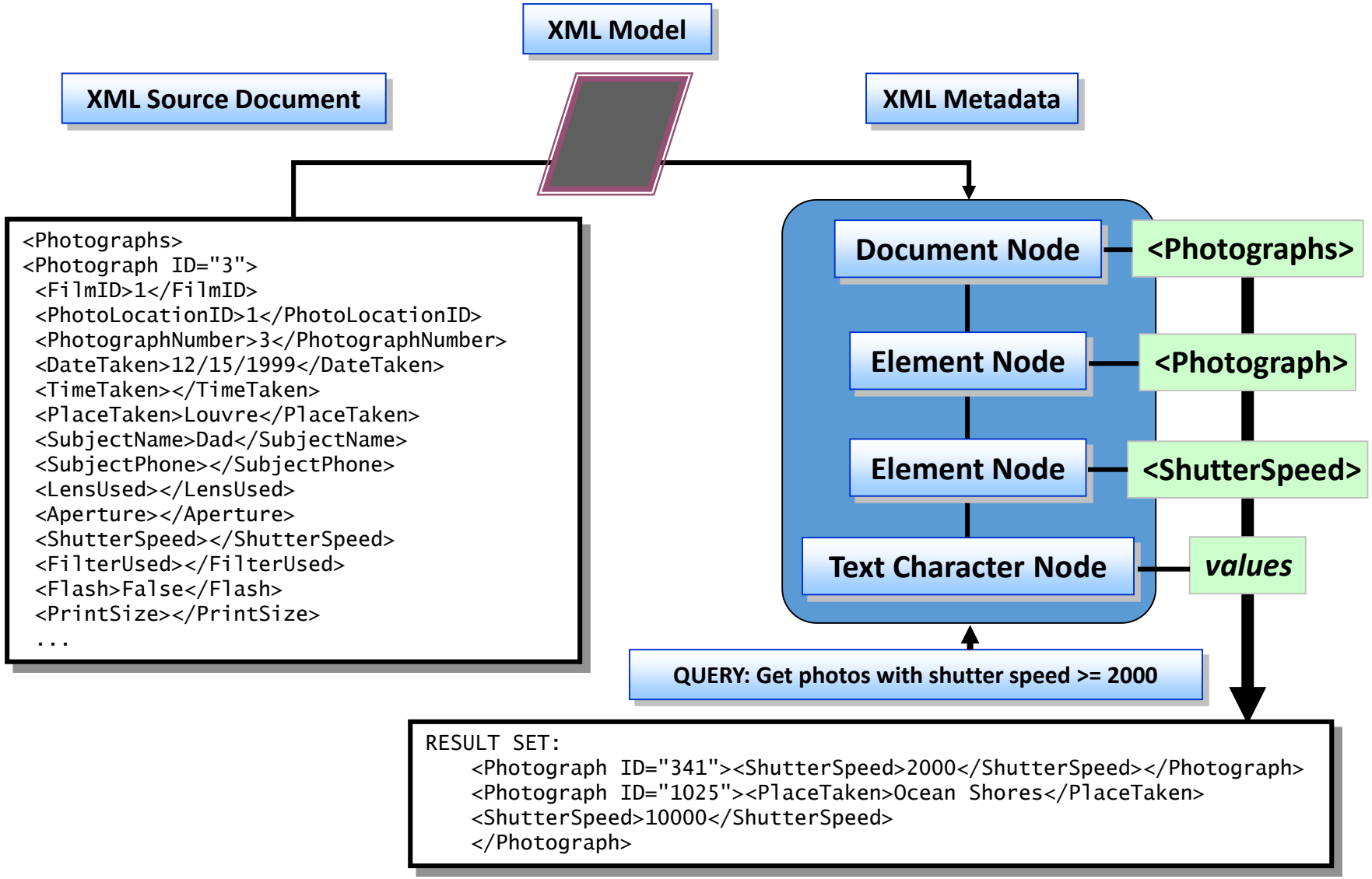
```
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

  - XSD names are prefixed with xsd:
- Element and attribute declarations
- Data types definitions
  - Use simple type for Boolean, integer, string, and so on
  - Use complex type for element relationships

What Is XML Metadata?

- Metadata is data about data
- Humans use lexical metadata to understand the markup
  - Consists of the meaning of the element and attribute names
- Software uses structural metadata
  - Consists of the element hierarchy and the presence of attributes, comments, or CDATA sections
- DOM and XPath model structural metadata as a tree

How to Use XML Metadata

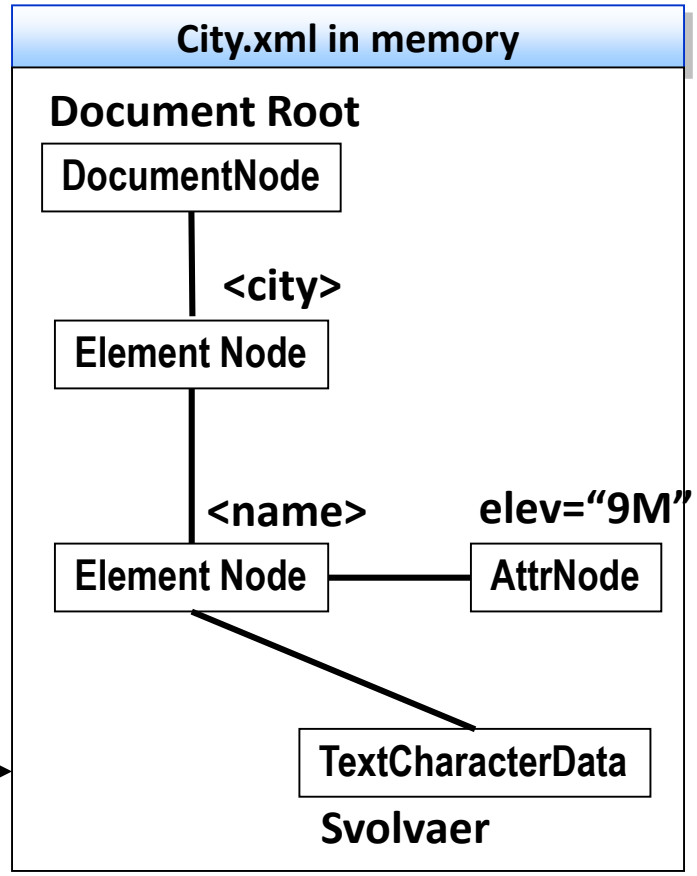


What Is the XML Document Object Model?

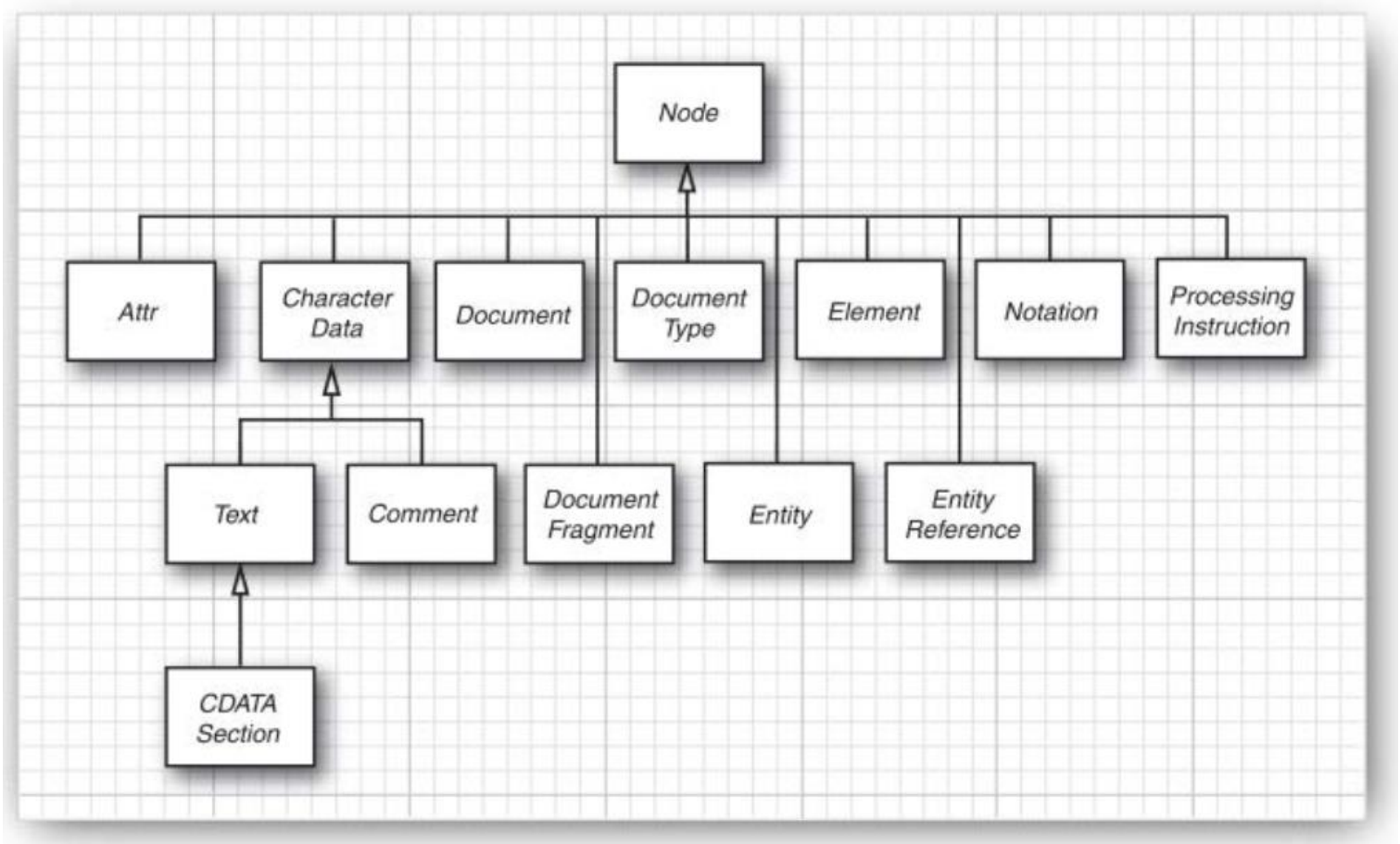
- DOM is the W3C programming interface for XML
- DOM models an XML source as a tree of nodes in memory
- DOM parser reads an entire XML document into a tree structure
- You can use DOM to:
  - Navigate and search
  - Add and delete content

City.xml

```
<city>
<name
elev="9M">Svolvaer</name>
</city>
```



DOM using Java



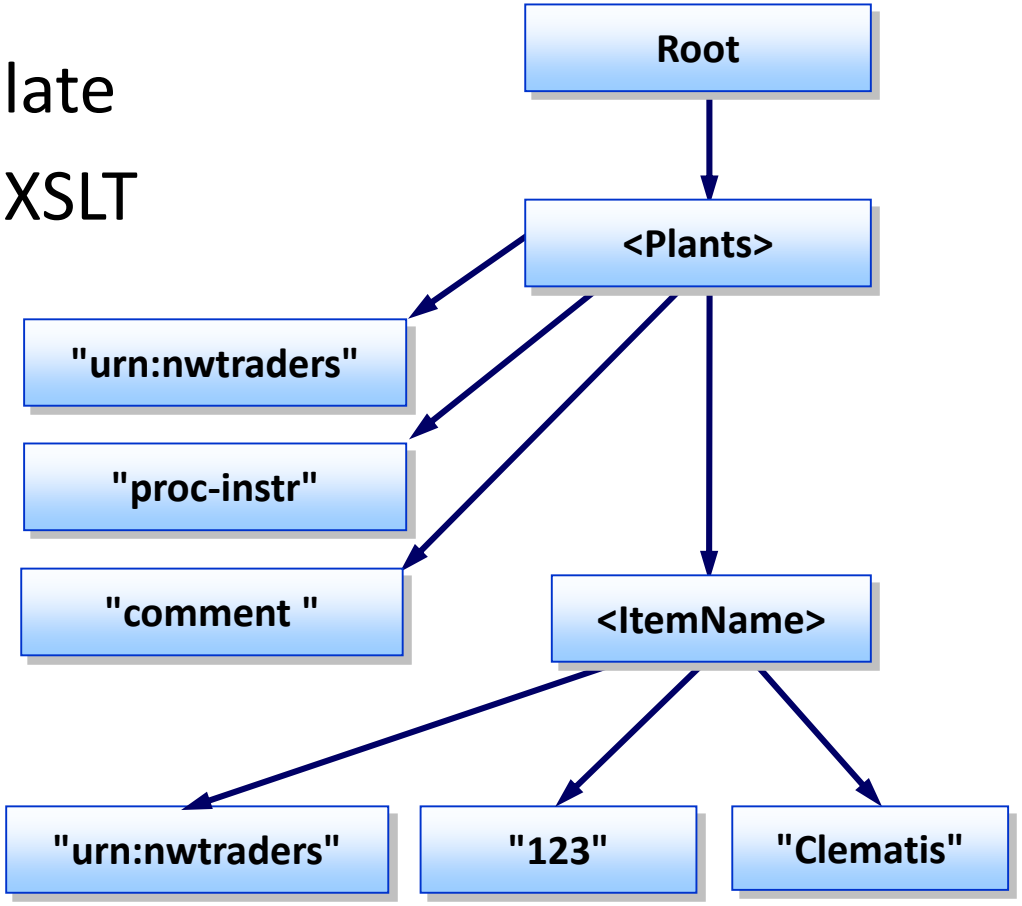
Simple API for XML (SAX)

- SAX parser generates events as it reads an XML document
- Does not store the document in any way
- Need a handler that defines the event actions
- StAX is a “pull parser”
  - Do not install event handlers
  - Iterate through the events

What Is XPath?

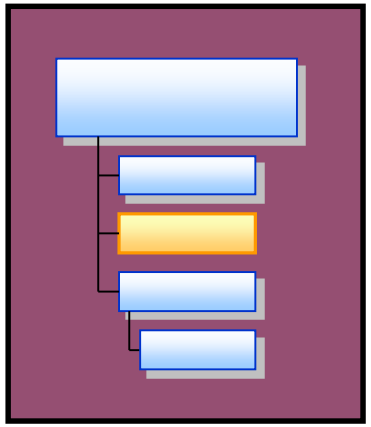
- XPath is the W3C XML Path Language
- Use XPath to navigate, search, manipulate
- XPath is used with DOM, XQuery, XSL, XSLT
- XPath models XML as a tree of nodes

```
<Plants
 xmlns="urn:nwtraders">

 <?proc instr?>
 <!--comment-->

 <ItemName code="123">
      Clematis
 </ItemName>

   </Plants>
```



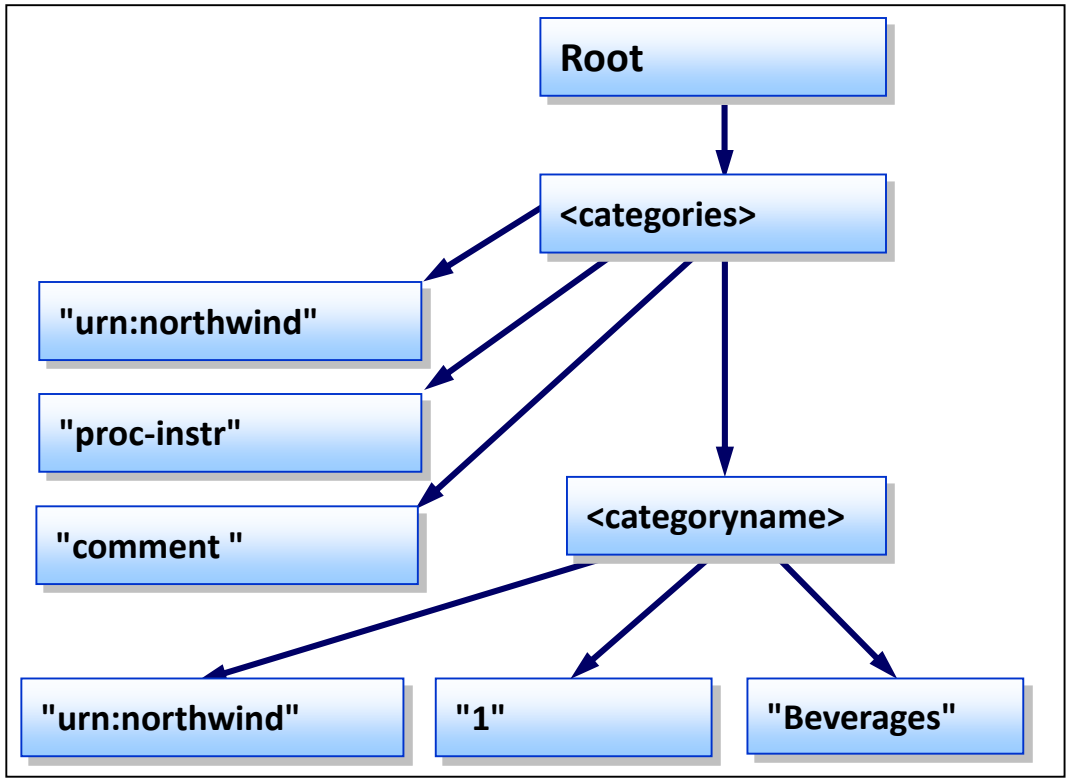
Language for Document Addressing

- Use XPath to address parts of an XML source
- Use XPath to match patterns of content or structure
- XPath includes an object model for XML that maps XML element to a tree of node types



What Are the Node Types in XPath?

- Root
- Element
- Attribute
- Namespace
- Processing Instruction
- Comment
- Text
- Whitespace




```
<categories xmlns="urn:northwind">
<?proc instr?>
 <!--comment-->
 <categoryname id="1">Beverages</categoryname>
</categories>
```

What Is a Location Path?

- Composed of one or more location steps
- Read from left to right
- Location step syntax (unabbreviated): `axis::node-test[predicate]`
- Axis types:

parent::	**following::**
child::	**preceding::**
ancestor::	**ancestor-or-self::**
self::	**descendant-or-self::**
descendant::	
namespace::	**following-sibling::**
attribute::	**preceding-sibling::**

- **Node-test parameters:**

by node name or by node type

- **Predicate options:**

filter by position
filter by value
filter by presence

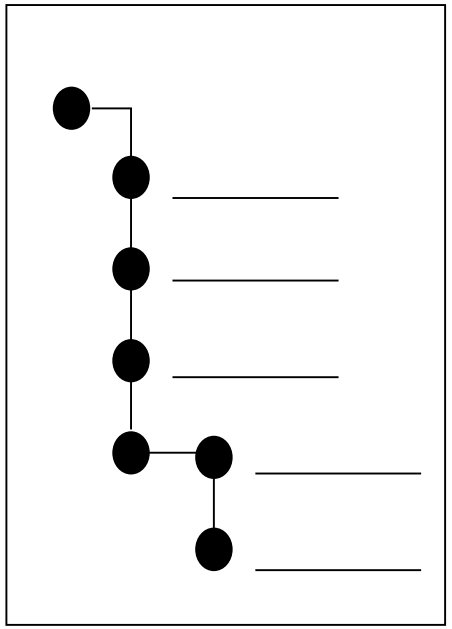
Location Path Syntax

- Choose between unabbreviated and abbreviated syntax when specifying the axis

Unabbreviated syntax	Abbreviated syntax
`attribute::`	`@`
`/descendant-or-self::node()`	`//`
`self::node`	`.`
`parent::node`	`..`
`node tree root`	`/`
`ancestor-or-self::`	`not available`

How to Construct a Location Path

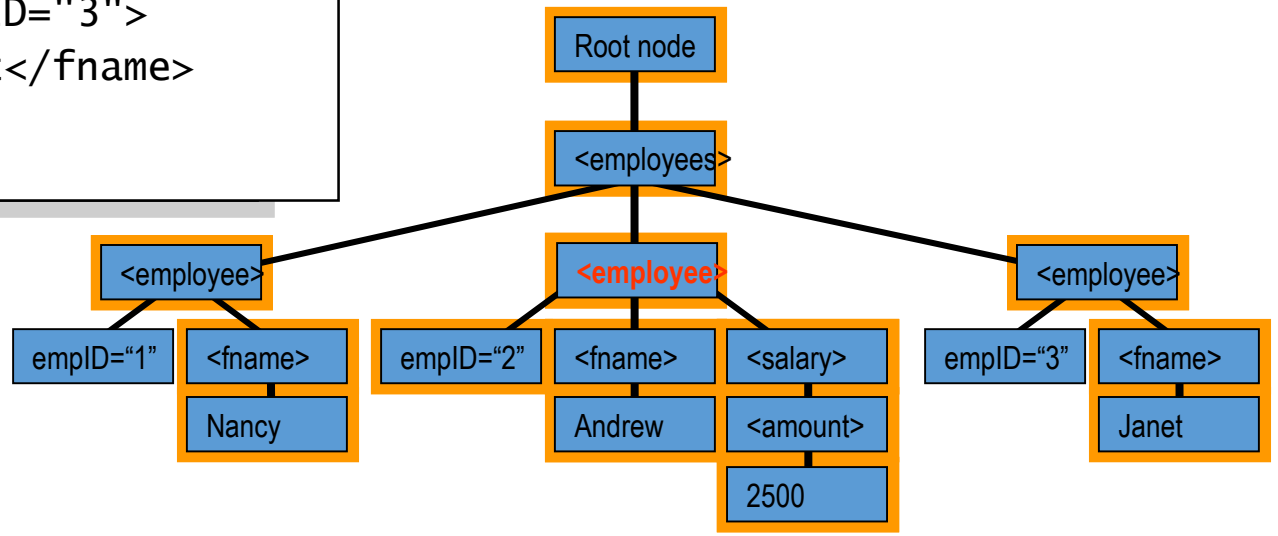
1. Analyze the XML source to be used
2. Define your search criteria
3. Determine what to locate in the document:
   How deeply is the data nested?
   How will the source vary?
   Will the source be validated?
4. Build, test, and refine the location path



How to Define the Axis

```
<employees>
  <employee empID="1">
    <fname>Nancy</fname>
  </employee>
  <employee empID="2">
    <fname>Andrew</fname>
    <salary>
      <amount>2500</amount>
    </salary>
  </employee>
  <employee empID="3">
    <fname>Janet</fname>
  </employee>
</employees>
```

- self
- child
- parent
- attribute
- descendant
- descendant-or-self
- ancestor
- ancestor-or-self
- following
- following-sibling
- preceding
- preceding-sibling



Operators and Functions

- Union operator (|) allows you to merge node-sets
- Node-set functions:

```
cars[position() = last()]

count(car[@fueltank > 25])

id("quidID")

*[namespace-uri()= " www.litware.com"
```

```
position
last
count
id
local-name
namespace-uri
name
```

Boolean Operators and Functions

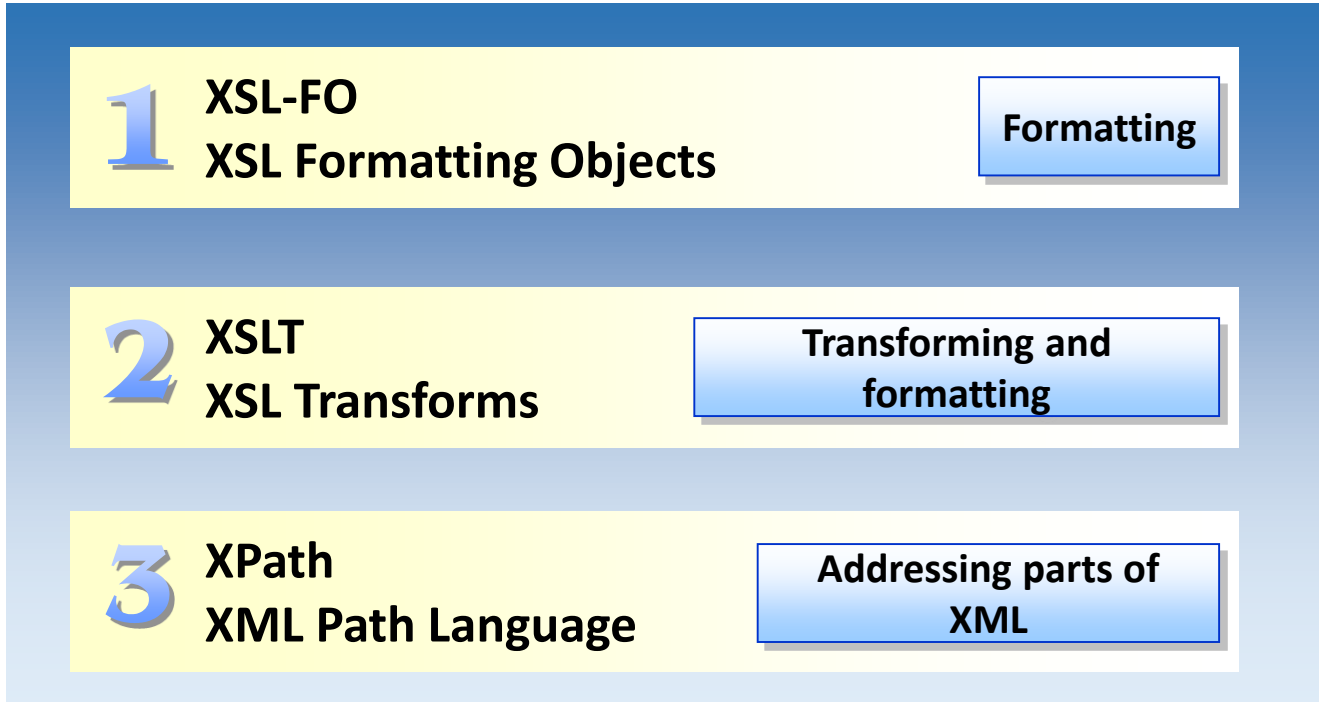
Boolean Operators	Boolean Functions
= or =!	True
>, >=, <, <=	False
and	boolean
or	not
	lang

Numeric and String Operators and Functions

Number operators	Number functions	String functions	String functions
- (unary)	number	string	substring-before
+	floor	string-length	substring-after
- (subtraction)	ceiling	concat	normalize-space
*	round	starts-with	translate
div	sum	contains	
mod		substring	

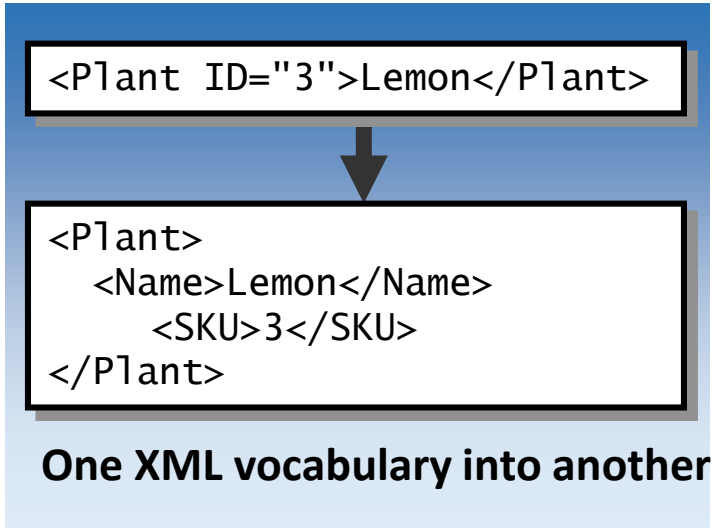
What Is XSL?

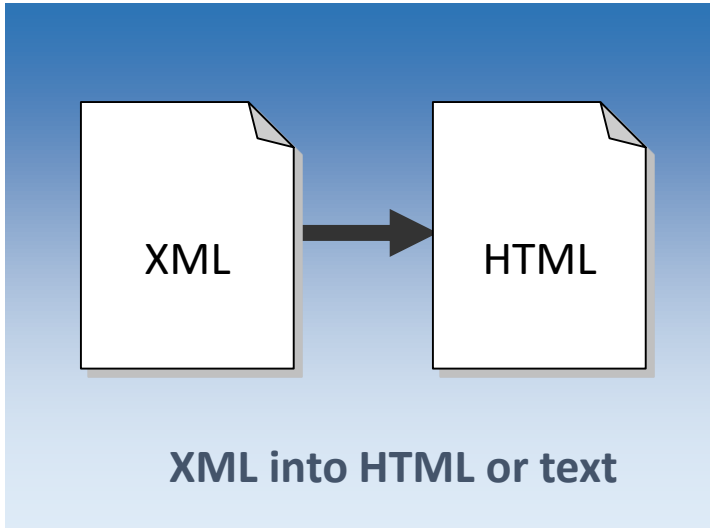
- Extensible Stylesheet Language (XSL) is a family of W3C recommendations for defining XML document transformation and presentation
- XSL has resulted in three language Recommendations:



What Is XSLT?

- XSLT is the W3C XSL Transformations language
- Used with XPath
- Use it to transform format and XML vocabularies





How Are XSLT and XPath Related?

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="/">
    <xsl:apply-templates
  select="employees/employee">
      <xsl:sort select="name" />
    </xsl:apply-templates>
  </xsl:template>

</xsl:stylesheet>
```

Starting at the document element…

List <employee> children of <employees>

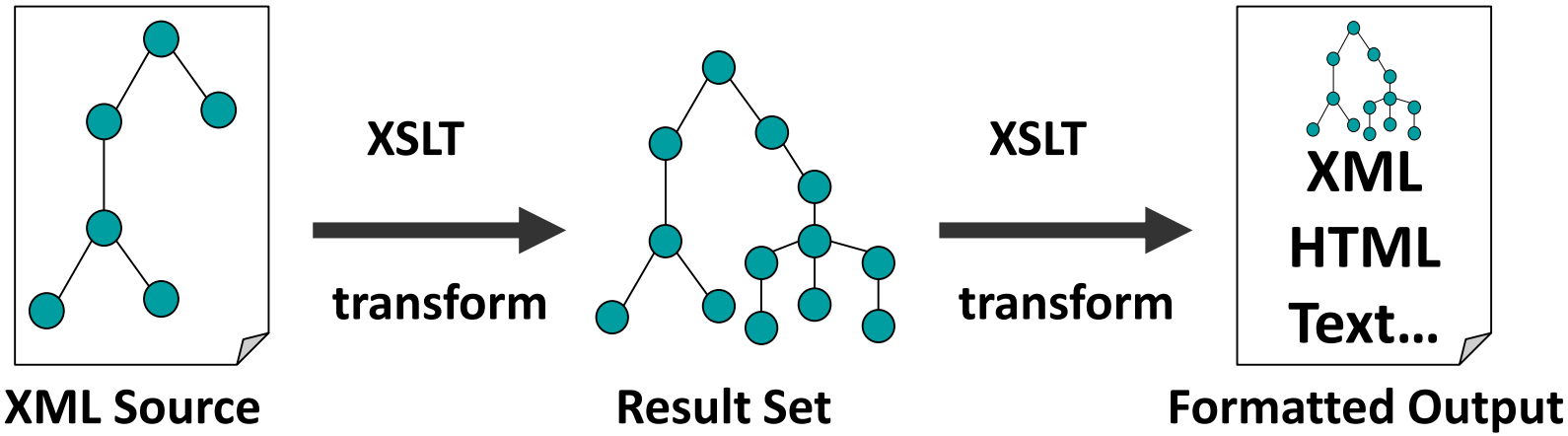
Sort the list by content in <name>

XPath

From the root node, create a list of employee names.

What Is XSLT?

- XSLT uses templates to transform XML
- Templates define transformations
- XSLT uses XPath to fix locations and for conditional processing
- XSLT is a declarative language
- Transformations are applied recursively and independently of the sequence in which they appear in the style sheet



Parts of an XSLT Style Sheet

- Identify this as an XSLT style sheet

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">
```

- Match a pattern

```
<xsl:template match="">
```

- Apply a transformation

```
<xsl:apply-templates select="">
```

- Create output

```
<xsl:value-of select="">
```

```
<xsl:copy-of select="">
```

What Can You Do with XSLT?

- Transforming XML
- Using templates and matching
- Filtering and sorting

//TODO before next lecture:

- Homework 3 due on 3/23 at 11:59 pm EDT. Must be submitted on Submitty.
- Final Project proposal due on 3/26 at 11:59 pm EDT. Must be submitted on Submitty.
- Java puzzler (posted on Submitty Forum).