XML Technologies

Doc. RNDr. Irena Holubova, Ph.D.

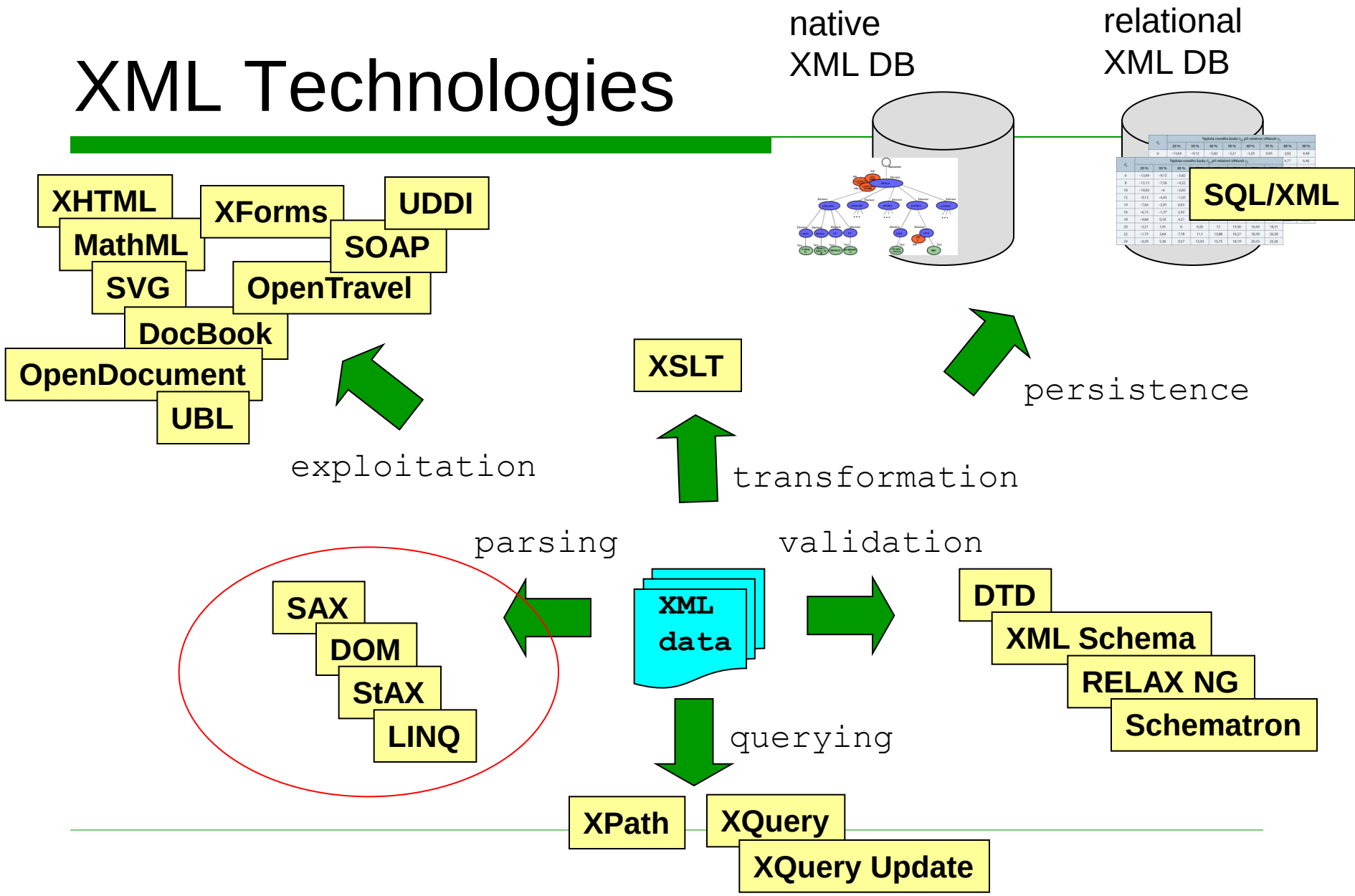
holubova@ksi.mff.cuni.cz

Web pages:
MFF: http://www.ksi.mff.cuni.cz/~holubova/NPRG036/
FEL: http://www.ksi.mff.cuni.cz/~holubova/A7B36XML/

Outline

- Introduction to XML format, overview of XML technologies
- DTD
- XML data models
- Interfaces for XML data
- XPath
- XSLT
- XQuery, XQuery Update
- XML schema languages
- SQL/XML
- An overview of standard XML formats
- XML data persistence

XML Technologies



Namespaces

Namespaces

Today we learn what it is, later (when talking about XML Schema) we learn how to create and use it.

- Problem: We need to distinguish the same names of elements and attributes in cases when a conflict may occur.
  - The application needs to know which elements/attributes it should process
  - e.g. name of a book vs. name of a company
- Idea: expanded name of an element/attribute = ID of a namespace + local name
  - The namespace is identified by URI
- URI is too long → shorter version
  - Namespace declaration = prefix + URI
  - Qualified name (QName) = prefix + local name of an element/attribute
- Note: DTD does not support namespaces (it considers them as any other element/attribute names)
  - XML Schema is conversely based on namespaces

Ex. Namespace

Namespace declaration → the area of validity

```
<pricelist:offer
    xmlns:pricelist="http://www.eprice.cz/e-pricelist">
  <pricelist:item tax="22%">
    <pricelist:name>
      <bib:book xmlns:bib="http://www.my.org/bib">
        <bib:author>Mark Logue</bib:author>
        <bib:name>The King's Speech: How One Man Saved
the British Monarchy</bib:name>
      </bib:book>
    </pricelist:name>
    <pricelist:price curr="CZK">259</pricelist:price>
  </pricelist:item>
</pricelist:offer>
```

Namespace usage

Ex. Implicit Namespace

```
<offer
    xmlns="http://www.eprice.cz/e-pricelist">
  <item tax="22%">
    <name>
      <bib:book xmlns:bib="http://www.my.org/bib">
        <bib:author>Mark Logue</bib:author>
        <bib:name>The King's Speech: How One Man
Saved the British Monarchy </bib:name>
      </bib:book>
    </name>
    <price curr="CZK">259</price>
  </item>
</offer>
```

Namespace

- A set of non-conflicting identifiers
- A namespace consists of disjoint subsets:
  - All element partition
    - A unique name is given by namespace identifier and element name
    - I.e. all elements have unique names
  - Per element type partitions
    - A unique name is given by namespace identifier, element name and local name of attribute
    - I.e. attributes have names unique within element declarations
  - Global attribute partition
    - A unique name is given by namespace identifier and attribute name
      - This kind of attribute can be defined in XML Schema
    - I.e. a special type of attributes having unique names among all attributes

See XML Schema for explanation

Ex. Parts of Namespaces

```
<offer
    xmlns="http://www.eprice.cz/e-pricelist"
    xmlns:bib="http://www.my.org/bib">
  <item tax="22%">
    <name>
      <bib:book>
        <bib:author>Mark Logue</bib:author>
        <bib:name xml:lang="cs">The King's Speech:
How One Man Saved the British  archy </bib:name>
      </bib:book>
    </name>
    <price curr="CZK">259</price>
  </item>
</offer>
```

Element from namespace bib

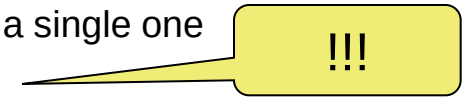
???

Global attribute from namespace xml

Attribute of element price, from (implicit) namespace http://www.eprice.cz/e-pricelist

Namespace XML

- Each XML document is assigned with namespace XML
  - URI: http://www.w3.org/XML/1998/namespace
  - Prefix: xml
  - It does not have to be declared
- It involves global attributes:
  - xml:lang – the language of element content
    - Values are given by the XML specification
  - xml:space – processing of white spaces by the application
    - preserve
    - default = use application settings
      - Usually replaces multiple white spaces with a single one
  - xml:id – unique identifier (of type ID)
  - xml:base – declaration of base URI, others can be defined relatively
    - E.g. in XML technology XLink



More on Namespaces

- W3C specification:
  - http://www.w3.org/TR/REC-xml-names/
- Lectures on XML Schema
  - Later

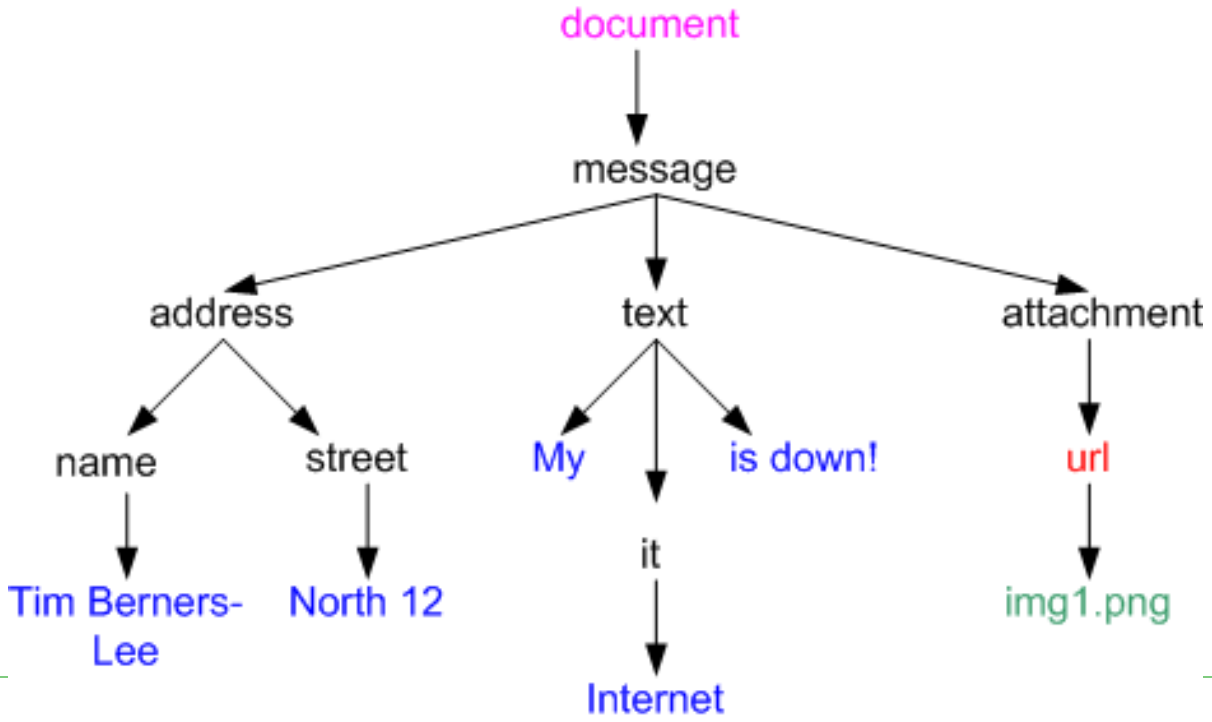
XML Data Model

XML Infoset

- A well formed XML document → hierarchical tree structure = XML Infoset
  - Abstract data model of XML data
- Information set = the set of information (in the XML document)
- Information item = a node of the XML tree
  - Types of items: document, element, attribute, string, processing instruction, comment, notation, DTD declaration, …
  - Properties of items: name, parent, children, content, …
- It is used in other XML technologies
- DTD (in general XML schema) can „modify“ Infoset
  - E.g. default attribute values

Ex.

```
<message>
  <address>
    <name>Tim Berners-Lee</name>
    <street>North 12</street>
  </address>
  <text>My <it>Internet</it> is down!</text>
  <attachment url="img1.png"/>
</message>
```



Ex. Element Information Item

- [namespace name]
  - (Possibly empty) name of namespace
- [local name]
  - Local part of element name
- [prefix]
  - (Possibly empty) prefix of namespace
- [children]
  - (Possibly empty) sorted list of child items
    - Document order
    - Elements, processing instructions, unexpanded references to entities, strings and comments
- [attributes]
  - (Possibly empty) unsorted set of attributes (Attribute Information Items)
    - Namespace declarations are not included here
  - Each item (attribute) is declared or given by the XML schema
    - Attributes with default values

Ex. Element Information Item

- [namespace attributes]
  - (Possibly empty) unsorted set of declarations of namespaces
- [in-scope namespaces]
  - Unsorted set of namespaces which are valid for the element
  - It always contains namespace XML
  - It always contains items of set [namespace attributes]
- [base URI]
  - URI of the element
- [parent]
  - Document/Element Information Item to whose property [children] the element belongs

- For other items see http://www.w3.org/TR/xml-infoset/

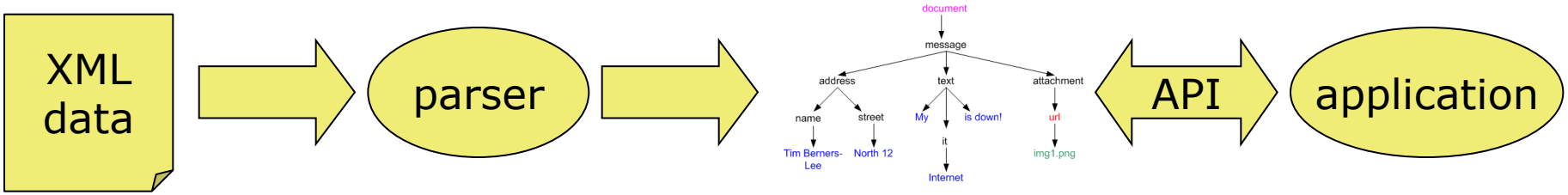
Post Schema Validation Infoset (PSVI)

- Typed Infoset
- It results from assigning data types on the basis of validation against an XML schema
  - We can work directly with typed values
  - Without PSVI we have only text values
    - DTD: minimum of data types
    - XML Schema: int, long, byte, date, time, boolean, positiveInteger, …
- Usage: in query languages (XQuery, XPath)
  - E.g. We have functions specific for strings, numbers, dates etc.

XML parsers

XML Parsers

- Problem: We want to process XML data
  - Read it in a particular SW
- XML document = text document → we can read the document as a text
  - Demanding, user-unfriendly, inefficient,…
- Solution: While processing XML data, we need to know what is element, attribute, text, comment, … → we are interested in Infoset of the XML document
- XML parser = SW which provides an application with an interface to the Infoset of input XML data

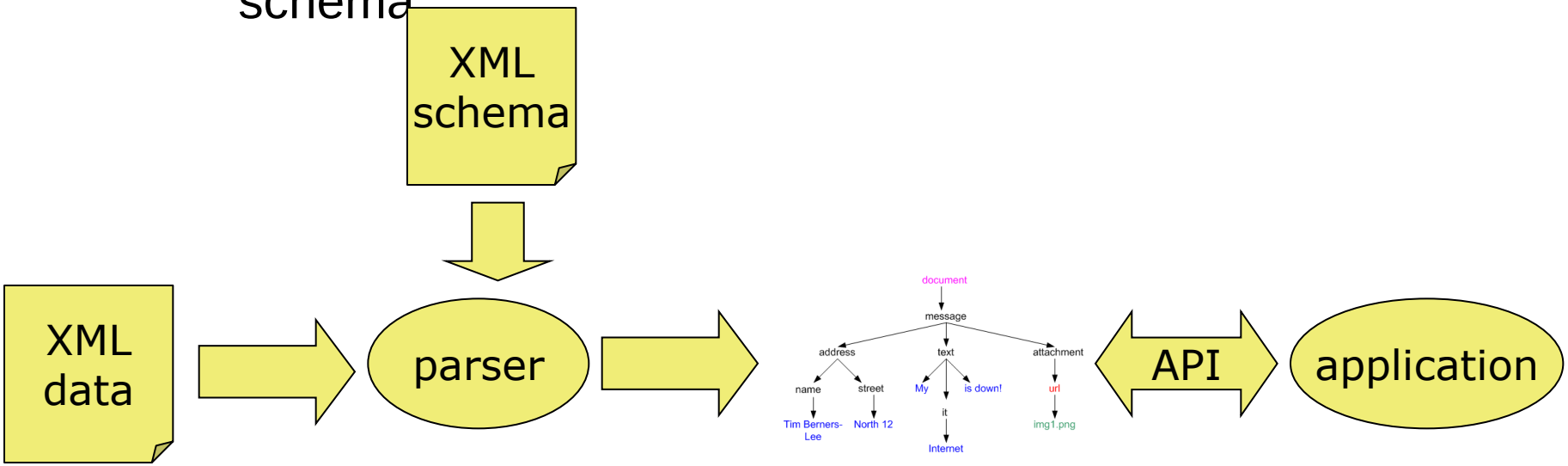


Types of Parsers (1)

- Sequential
  - Fast, require less memory
  - A single linear traversal of the data
  - Push vs. pull parser
    - A stream of events vs. reading when required
- Tree representations
  - The whole document is read into memory
  - Repeatable and non-sequential traversal
  - Memory requirements, inefficient

Types of Parsers (2)

- Validating × non-validating
  - Can check validity of the data against an XML schema

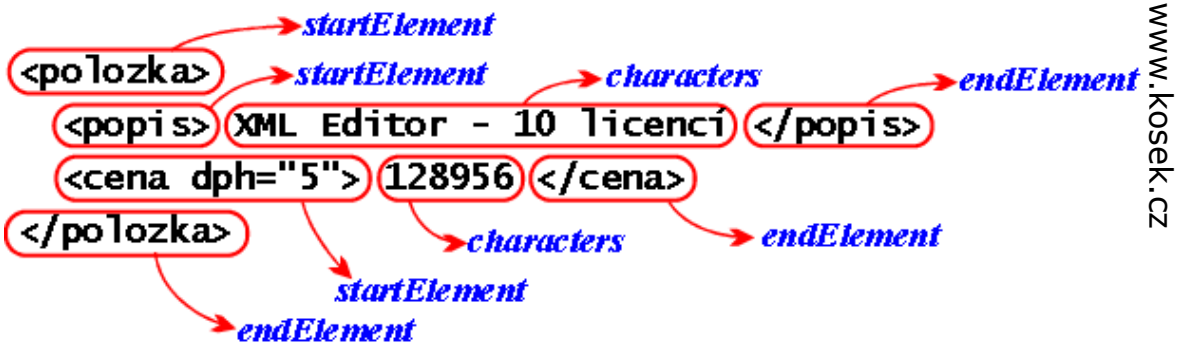


- With(out) support for PSVI

Interface SAX

SAX

- SAX = Simple API for XML
- Reading a part of document = event
  - We can define a handler
- Key events:



  - Attributes are a part of parameters of startElement ()

Java: Interface ContentHandler

- void startDocument ()
- void endDocument ()
- void startElement (String uri, String localName, String qName, Attributes atts)
- void endElement (String uri, String localName, String qName)
- void characters (char[] ch, int start, int length)
- void processingInstruction (String target, String data)
- void ignorableWhitespace (char[] ch, int start, int length)
- void startPrefixMapping (String prefix, String uri)
- void endPrefixMapping (String prefix)
- void skippedEntity (String name)
- void setDocumentLocator (Locator locator)

ContentHandler: startElement ()

- String uri
  - URI of element namespace
- String localName
  - Local name
- String qName
  - Qualified name
- Attributes atts

```
for (int i = 0; i < atts.getLength (); i++ ) {
  System.out.println (atts.getQName (i));
  System.out.println (atts.getValue (i));
  }
```

Interface Attributes (1)

- int getLength ()
  - Returns the number of attributes in the list of attributes
- int getIndex (String qName)
  - Returns the index of attribute with the given (qualified) name
- int getIndex (String uri, String localName)
  - Returns the index of attribute with the given local name and URI of namespace
- String getLocalName (int index)
  - Returns the local name of attribute with the given index
- String getQName (int index)
  - Returns the qualified name of attribute with the given index
- String getURI (int index)
  - Returns the URI of attribute with the given index

Interface Attributes (2)

```
E.g.
CDATA
ID
IDREF
IDREFS
NMTOKEN
...
```

- String getType (int index)
  - Returns the type of attribute with the given index
- String getType (String qName)
  - Returns the type of attribute with the given (qualified) name
- String getType (String uri, String localName)
  - Returns the type of attribute with the given local name and URI of namespace
- String getValue (int index)
  - Returns the value of attribute with the given index
- String getValue (String qName)
  - Returns the value of attribute with the given (qualified) name
- String getValue (String uri, String localName)
  - Returns the value of attribute with the given local name and URI of namespace

ContentHandler: characters ()

- SAX parser can buffer the character data arbitrarily → we cannot rely on getting the whole text in a single call of the function
- char[] ch
  - An array where the character data are stored
- int start
  - Starting position of the characters in the array
- int length
  - Number of characters in the array

ContentHandler: ignorableWhitespace ()

- Ignorable white spaces
- char[] ch
  - An array where the character data are stored
- int start
  - Starting position of the characters in the array
- int length
  - Number of characters in the array

ContentHandler: setDocumentLocator ()

```
class myContentHandler implements ContentHandler {
  Locator locator;

  public void setDocumentLocator (Locator locator) {
    this.locator = locator;
    }
  ...
```

- Targeting the place in the document where the event occurred
- Interface Locator
  - int getColumnNumber () – column number
  - int getLineNumber () – row number
  - String getPublicId () – public identifier of the document (if exists)
  - String getSystemId () – system identifier of the document (if exists)

Initialization of the Processing

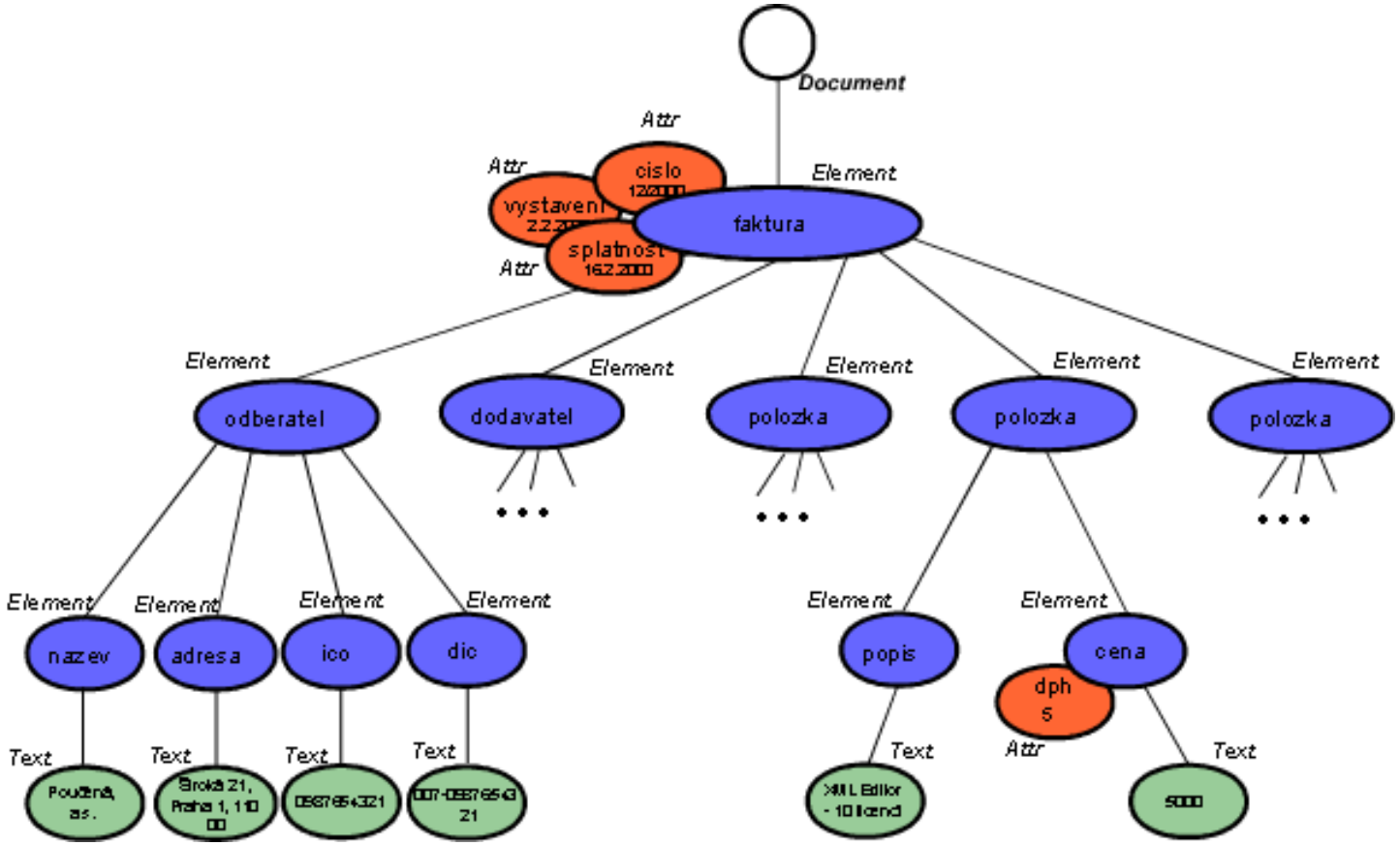
```
// Creating of an instance of the parser
XMLReader parser = XMLReaderFactory.createXMLReader ();

// Creating of input stream of data
InputSource source = new InputSource ("myDocument.xml");

// Setting our own content handler for processing of events
parser.setContentHandler (new myContentHandler ());

// Processing of the data
parser.parse (source);
```

Interface DOM

DOM

- DOM = Document Object Model
- W3C standard
  - Versions: Level (0), 1, 2, 3
    - Level 0 = DOM-like technologies before standardization by W3C
  - http://www.w3.org/DOM/DOMTR
- The whole document is loaded into memory
- Tree representation
- Nodes of the tree are represented as objects
  - Document, document fragment, DTD declaration, element, attribute, text, CDATA section, comment, entity, entity reference, notation, processing instruction
  - Methods of objects are given by the DOM specification
  - Child nodes of objects are given by XML Infoset

Example: DOM Tree



Java: Building DOM Tree

```
// DocumentBuilderFactory creates DOM parsers
DocumentBuilderFactory dbf =
  documentBuilderFactory.newInstance ();

// we do not want to validate (and other parameters can be set)
dbf.setValidating (false);

// we create a DOM parser
DocumentBuilder builder =
  dbf.newDocumentBuilder ("myDocument.xml");

// the parser processes the documents and builds the tree
Document doc = builder.parse ();

// we process the DOM tree
processTree (doc);
```

Document doc

Java: Storing DOM Tree

```
// TransformerFactory creates DOM serializers
TransformerFactory tf = TransformerFactory.newInstance ();

// Transformer serializes DOM trees
Transformer writer = tf.newTransformer ();

// we set encoding
writer.setOutputProperty
  (OutputKeys.ENCODING, "windows-1250");

// we start transformation of DOM tree into a document
writer.transform
  (new DOMSource (doc),
   new StreamResult (new File ("outputDocument.xml")));
```

Java Classes (1)

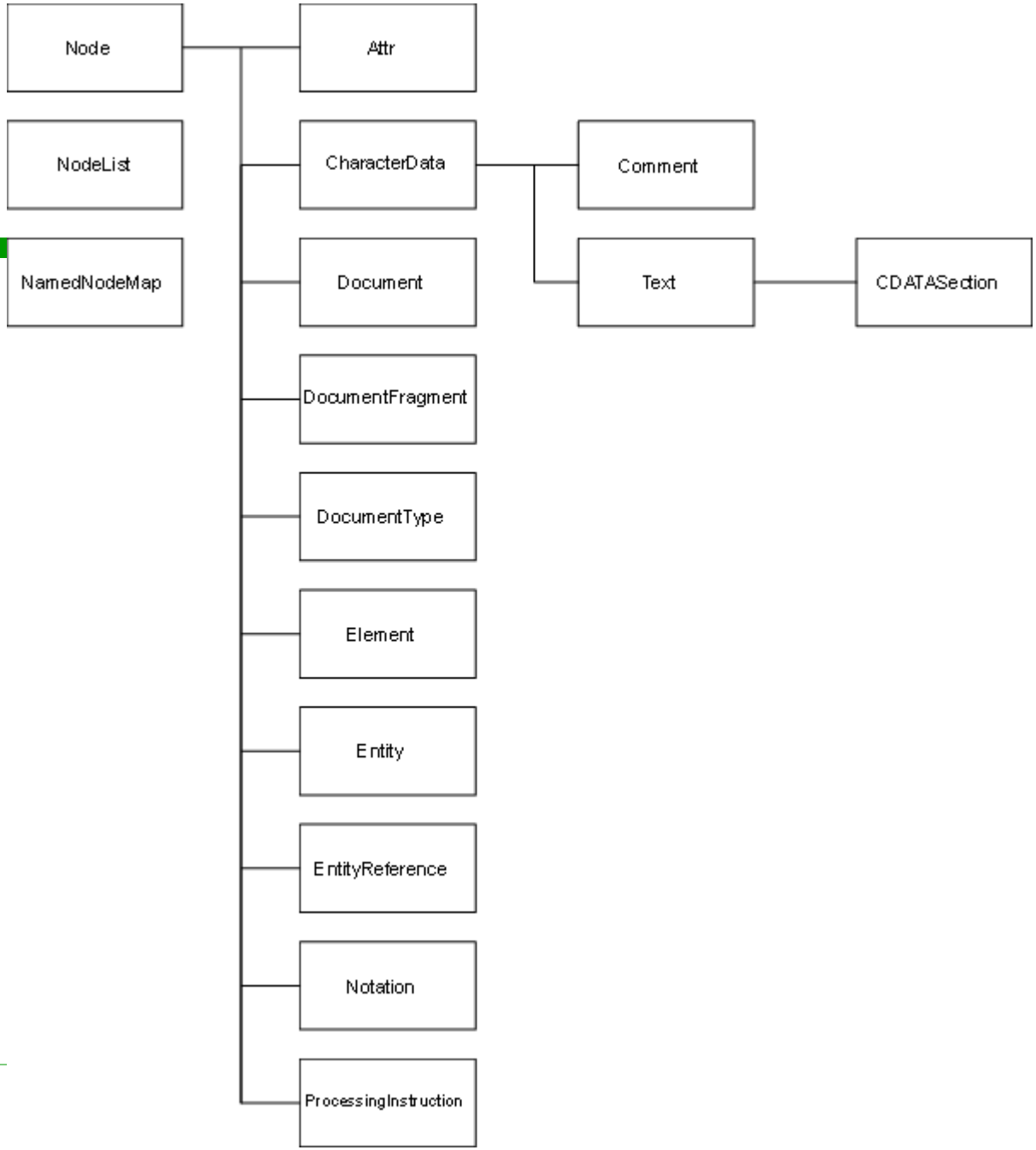
- Node – basis for other interfaces representing further nodes of tree

Node	Child Nodes
Document	Element (at most one), ProcessingInstruction, Comment, DocumentType (at most one)
DocumentFragment	Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
Element	Element, Text, Comment, ProcessingInstruction, CDATASection, EntityReference
Attr	Text, EntityReference
Text	-
CharacterData	-

Java Classes(2)

Node	Child Nodes
ProcessingInstruction	-
Comment	-
CDATASection	-
Entity	Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
EntityReference	Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
Notation	-
DocumentType	-

Hierarchy of Classes



Interface Node (1)

- String getNodeName ()
- short getNodeType ()
- String getNodeValue ()

- String getBaseURI ()
- String getPrefix ()
- void setPrefix (String prefix)
- String getLocalName ()
- String getNamespaceURI ()
- String lookupNamespaceURI (String prefix)
- String lookupPrefix (String namespaceURI)
- boolean isDefaultNamespace (String namespaceURI)

namespace information

- boolean hasAttributes ()
- boolean hasChildNodes ()

checking structure

Interface Node (2)

- Node getParentNode ()
- Node getPreviousSibling ()
- Node getNextSibling ()
- NodeList getChildNodes ()
- Node getFirstChild ()
- Node getLastChild ()

traversing the tree

- NamedNodeMap getAttributes ()
- String getTextContent ()
- Document getOwnerDocument ()
  - Returns Document associated with the node

- Node removeChild (Node oldChild)
- Node replaceChild (Node newChild, Node oldChild)
- Node appendChild (Node newChild)
- Node insertBefore (Node newChild, Node refChild)

modifying the tree

Interface Node (3)

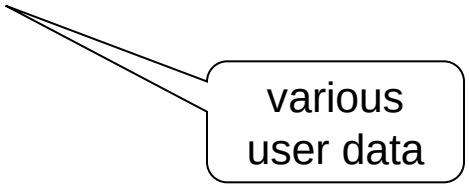
- Node cloneNode (boolean deep)
- void setNodeValue (String nodeValue)
- void setTextContent (String textContent)

- void normalize ()
  - Normalizes all text sub-nodes, i.e. merges text contents and eliminates the empty ones

- boolean isEqualNode (Node arg)
- boolean isSameNode (Node other)
- short compareDocumentPosition (Node other)
  - Compares positions of Nodes in the document

Interface Node (4)

- Object getUserData (String key)
  - Returns an Object associated with key
- Object setUserData (String key, Object data, UserDataHandler handler)
  - Associates an Object with a key
  - Handler is a callback which will be called any time the node is being cloned, imported, renamed, as well as if deleted or adopted
- Object getFeature (String feature, String version)
  - Returns an Object with a given feature with a given version
- boolean isSupported (String feature, String version)
  - Tests if the implementation supports the given feature with the given version



Interface	nodeName	nodeValue	attributes
Attr	like Attr.name	like Attr.value	null
CDATA-Section	"#cdata-section"	like CharacterData.data, content of CDATA section	null
Comment	"#comment"	like CharacterData.data, content of comment	null
Document	"#document"	null	null
Document-Fragment	"#document-fragment"	null	null
Document-Type	like DocumentType.name	null	null
Element	like Element.tagName	null	Named-NodeMap
Entity	name of entity	null	null
Entity-Reference	name of referenced entity	null	null
Notation	name of notation	null	null
Processing-Instruction	like ProcessingInstruction.target	like ProcessingInstruction.data	null
Text	"#text"	like CharacterData.data, content of text node	null

Ex. Child Nodes vs. Attributes

```
for (Node child = n.getFirstChild();
     child is down!= null;
     child = child.getNextSibling())
  {
  processChildNode(child);
  }
```

```
NamedNodeMap atts = n.getAttributes();
for (int i = 0; i < atts.getLength(); i++)
  {
  Node att = atts.item(i);
  processAttribute(att);
  }
```

Interface Document (1)

- Attr createAttribute (String name)
- Attr createAttributeNS (String namespaceURI, String qualifiedName)
- CDATASection createCDATASection (String data)
- Comment createComment (String data)
- DocumentFragment createDocumentFragment ()
- Element createElement (String tagName)
- Element createElementNS (String namespaceURI, String qualifiedName)
- EntityReference createEntityReference (String name)
- ProcessingInstruction createProcessingInstruction (String target, String data)
- Text createTextNode (String data)

creating new nodes

- Element getElementById (String elementId)
  - Returns an element with a given value of attribute of type ID
- NodeList getElementsByTagName (String tagname)
- NodeList getElementsByTagNameNS (String namespaceURI, String localName)

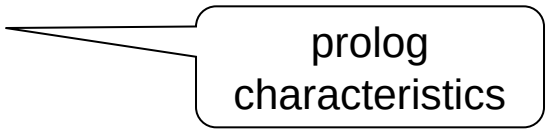
a kind of a query

Interface Document (2)

- Element getDocumentElement ()
- DocumentType getDoctype ()

- Node renameNode (Node n, String namespaceURI, String qualifiedName)
- Node adoptNode (Node source)
  - Appends Node to current document
- Node importNode (Node importedNode, boolean deep)
  - Imports a node to current document, i.e. creates its copy

- String getInputEncoding ()
  - Returns encoding used when parsing
- String getXmlEncoding ()
- DOMImplementation getImplementation ()
  - Returns implementation (DOMImplementation) associated with the document
- DOMConfiguration getDomConfig ()
  - Returns configuration for normalization of nodes



Interface Document (3)

- boolean getXmlStandalone ()
- String getXmlVersion ()
- String getDocumentURI ()
- void setXmlStandalone (boolean xmlStandalone)
- void setXmlVersion (String xmlVersion)
- void setDocumentURI (String documentURI)

prolog characteristics

- void normalizeDocument ()
  - Normalizes XML document, i.e. replaces all references to entities and normalizes text values
- boolean getStrictErrorChecking ()
  - Checks whether error checking is given by the specification or depends on the implementation
- void setStrictErrorChecking (boolean strictErrorChecking)
  - Sets whether error checking is given by the specification or depends on the implementation

Interface Element (1)

- String getTagName ()
  - Returns the element name
- NodeList getElementsByTagName (String name)
  - Returns the NodeList of all child elements with the given name
- NodeList getElementsByTagNameNS (String namespaceURI, String localName)
  - Returns the NodeList of all child elements with the given local name and URI
- String getAttribute (String name)
  - Returns the value of attribute with the given name
- Attr getAttributeNode (String name)
  - Returns the attribute with the given name
- Attr getAttributeNodeNS (String namespaceURI, String localName)
  - Returns the attribute with the given local name and URI
- String getAttributeNS (String namespaceURI, String localName)
  - Returns the value of attribute with the given local name and URI

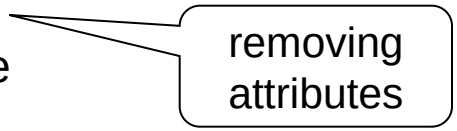
a kind of query

getting attributes

Interface Element (2)

- boolean hasAttribute (String name)
  - true = the element has an attribute wit the given name
- boolean hasAttributeNS (String namespaceURI, String localName)
  - true = the element has and attribute with the given local name and URI

- void removeAttribute (String name)
  - Removes attribute with the given name
- Attr removeAttributeNode (Attr oldAttr)
  - Removes the given attribute node
- void removeAttributeNS (String namespaceURI, String localName)
  - Removes attribute with the given local name and URI



- TypeInfo getSchemaTypeInfo ()
  - Type information for the given element

Interface Element (3)

- void setAttribute (String name, String value)
  - Adds a new attribute with the given name and value
- Attr setAttributeNode (Attr newAttr)
  - Adds a new attribute node, replaces if it exists
- Attr setAttributeNodeNS (Attr newAttr)
  - Adds a new attribute node, replaces if it exists and takes into account also namespaces
- void setAttributeNS (String namespaceURI, String qualifiedName, String value)
  - Adds a new attribute with the specified parameters

setting attributes

- void setIdAttribute (String name, boolean isId)
  - Changes attribute type from/to ID
- void setIdAttributeNode (Attr idAttr, boolean isId)
  - Changes attribute type from/to ID
- void setIdAttributeNS (String namespaceURI, String localName, boolean isId)
  - Changes attribute type from/to ID

Ex. Creating an Element

```
public Node createEmployee(Document document) {
  Element firstName = document.createElement("FirstName");
  firstName.appendChild(document.createTextNode("Shawn"));

  Element lastName = document.createElement("LastName");
  lastName.appendChild(document.createTextNode("Michaels"));

  Attr genderAttribute = document.createAttribute("gender");
  genderAttribute.setValue("M");

  Element employee = document.createElement("Employee");
  employee.setAttributeNode(genderAttribute);
  employee.appendChild(firstName);
  employee.appendChild(lastName);

  return employee;
  }
```

Interface Attr

```
NamedNodeMap attrs =
  node.getAttributes();
Attr attr = (Attr)attrs.item(0);
System.out.print(
  attr.getNodeName() + "=\"" +
  attr.getNodeValue() + "\"");
```

- String getName ()
  - Returns attribute name
- String getValue ()
  - Returns attribute value
- void setValue (String value)
  - Sets attribute value
- Element getOwnerElement ()
  - Returns the element node of the attribute
- TypeInfo getSchemaTypeInfo ()
  - Returns information on attribute type
- boolean getSpecified ()
  - true = the attribute was explicitly specified in the document
- boolean isId ()
  - true = the attribute is of type ID

Interface CharacterData

- String getData ()
  - Returns the text data
- int getLength ()
  - Returns the length of the data
- String substringData (int offset, int count)
  - Returns the required substring of the data
- void setData (String data)
  - Sets the text data
- void insertData (int offset, String arg)
  - Inserts a part of the data at the specified index
- void appendData (String arg)
  - Appends a new part of the data at the end
- void deleteData (int offset, int count)
  - Removes the specified part part of the data
- void replaceData (int offset, int count, String arg)
  - Replaces the specified part of the data

Interface Text

- Methods of CharacterData
- String getWholeText ()
  - Returns the text content of all logically neighbouring text child nodes connected into one
- Text replaceWholeText (String content)
  - Replaces textual content of all logically neighbouring text child nodes
- boolean isElementContentWhitespace ()
  - true = the text node contains insignificant white spaces
- Text splitText (int offset)
  - Splits the text at the given position into two

Interface ProcessingInstruction (PI)

- String getData ()
  - Returns the text content of PI
- void setData (String data)
  - Sets text content of PI
- String getTarget ()
  - Returns the target (i.e., the first part) of PI

Interface Notation

- String getPublicId ()
  - Returns public identifier of notation
- String getSystemId ()
  - Returns system identifier of notation

Interface Entity

- String getNotationName ()
  - Returns the name of notation associated with entity
- String getPublicId ()
  - Returns public identifier of the entity
- String getSystemId ()
  - Returns system identifier of the entity
- String getXmlVersion ()
  - Returns XML version of external entity
- String getXmlEncoding ()
  - Returns encoding of external entity
- String getInputEncoding ()
  - Returns encoding of external entity used for parsing

Interface DocumentType

- String getName ()
  - Returns root element name of DTD
- String getPublicId ()
  - Returns public identifier of DTD
- String getSystemId ()
  - Returns system identifier of DTD
- String getInternalSubset ()
  - Returns DTD declarations as a string
- NamedNodeMap getEntities ()
  - Returns the list of declared entities
- NamedNodeMap getNotations ()
  - Returns the list of declared notations

Other Interfaces

- Interface DocumentFragment
  - Just methods of Node
- Interface EntityReference
  - Just methods of Node
- Interface CDATASection
  - Methods of Node, Text and CharacterData
- Interface Comment
  - Methods of Node and CharacterData

DOM has lots of other classes…

- E.g. DOMLocator
  - DOM Level 3
  - Similar to SAX locator
  - Attributes: lineNumber, columnNumber, relatedNode, …
  - One of properties of class DOMError
    - Parameter of method handleError of class DOMErrorHandler which is a
      - property of class DOMConfiguration which is a
        - property of class Document (but from Level 3)

Interface StAX

StAX

- Streaming API for XML
  - http://stax.codehaus.org/Home
- Advantages:
  - DOM: Data traversal is driven by the application; support for data modification
  - SAX: Low memory requirements
  - StAX: Both
- General properties:
  - Pull parser
    - The application does not have to save the context, it decides when to move further
  - Idea: cursor which we can move through the data
    - Raw vs. object-based data access

Java: XMLEventReader

```
// we create XMLInputFactory
XMLInputFactory factory = XMLInputFactory.newInstance();

// we set properties
factory.setProperty
  (XMLInputFactory.IS_NAMESPACE_AWARE, true);

// we create a parser
XMLEventReader eventReader = factory.createXMLEventReader
  (new FileReader("myData.xml"));
```

Java: XMLEventReader

```
while (eventReader.hasNext()) {

  XMLEvent event = eventReader.nextEvent();

  if (event.getEventType() == XMLStreamConstants.START_ELEMENT)
    {
    StartElement startElement = event.asStartElement();
    System.out.println(startElement.getName().getLocalPart());
    }
}
```

- Events: ATTRIBUTE, CDATA, CHARACTERS, COMMENT, DTD, END_DOCUMENT, END_ELEMENT, ENTITY_DECLARATION, ENTITY_REFERENCE, NAMESPACE, NOTATION_DECLARATION, PROCESSING_INSTRUCTION, SPACE, START_DOCUMENT, START_ELEMENT

Java: XMLEventReader

- XMLEvent
  - Reads XML data
  - Knows where we are in the document
    - Column, row
  - Can be transformed to particular (XML) object:
    - asStartElement – element name, attribute, namespaces
    - asEndElement – element name, namespaces
    - asCharacters – text data
      - CDATA sections, white spaces, ignorable white spaces, …

Java: XMLEventWriter

```
// we create XMLOutputFactory
XMLOutputFactory factory = XMLOutputFactory.newInstance();

// we create serializer of events
XMLEventWriter writer =
  factory.createXMLEventWriter
    (new FileWriter("myData2.xml"));

// we create XMLEventFactory to create events
XMLEventFactory eventFactory =
  XMLEventFactory.newInstance();
```

```
XMLEvent event = eventFactory.createStartDocument();
writer.add(event);

event = eventFactory.createStartElement
  ("employee", "http://mynamespace.com", "mns");
writer.add(event);

event = eventFactory.createNamespace
  ("mns", "http://mynamespace.com");
writer.add(event);

event = eventFactory.createAttribute
  ("number", "1234");
writer.add(event);

event = eventFactory.createEndElement
  ("employee", "http://mynamespace.com", "mns");
writer.add(event);

writer.flush();
writer.close();
```

Java: XMLStreamReader

```
// we create XMLInputFactory
XMLInputFactory factory = XMLInputFactory.newInstance();

// we set properties
factory.setProperty(XMLInputFactory.IS_NAMESPACE_AWARE,
true);

// we create parser
XMLStreamReader streamReader =
  factory.createXMLStreamReader
    (new FileReader("myData.xml"));
```

Java: XMLStreamReader

```
while (streamReader.hasNext())
  {
  streamReader.next();

  if (streamReader.getEventType() ==
      XMLStreamReader.START_ELEMENT) {
    System.out.println(streamReader.getLocalName());
    }
  }
```

- Main difference: When we move cursor (next()), we loose information on the previous event
  - XMLEventReader returns the event as an object – we can store it

Java: XMLStreamWriter

```
XMLOutputFactory factory =
  XMLOutputFactory.newInstance();
XMLStreamWriter writer =
  factory.createXMLStreamWriter( new
FileWriter("myData2.xml"));

writer.writeStartDocument();
writer.writeStartElement("employee");
writer.writeStartElement("data");
writer.writeAttribute("number", "1234");
writer.writeEndElement();
writer.writeEndElement();
writer.writeEndDocument();
writer.flush();
writer.close();
```

JAXP

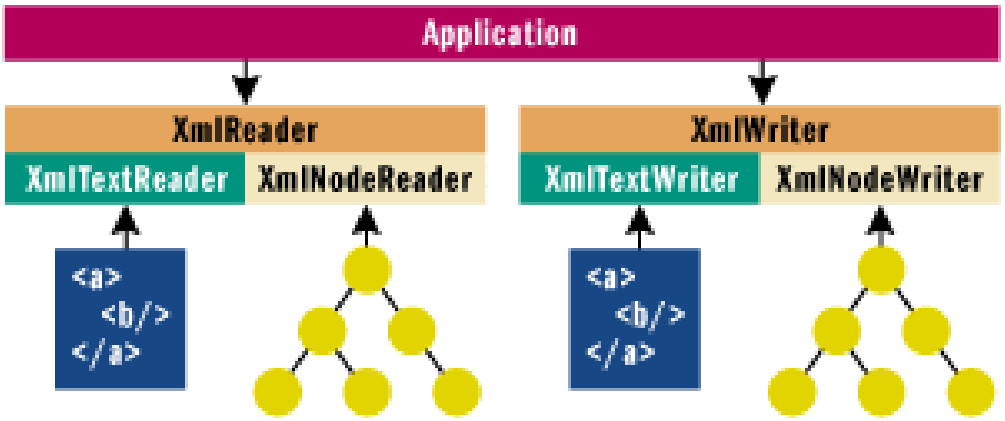
JAXP

- Java API for XML Processing
  - https://jaxp.java.net/
  - http://java.sun.com/j2ee/1.4/docs/tutorial/doc/
- JAXP 1.3 is a part of J2SE 5.0
- JAXP 1.4 is a part of Java SE 6.0.
  - Corrected errors in JAXP 1.3 + support for StAX
- Parsing, validation, transformation
  - XML 1.0, XML 1.1
  - SAX 2.0.2
  - StAX 1.0
  - DOM Level 3 Core, DOM Level 3 Load and Save
  - XInclude 1.0, W3C XML Schema 1.0, XSLT 1.0, XPath 1.0

XML and .NET

XML and .NET

- .NET Framework XML Classes
  - http://msdn.microsoft.com/en-us/magazine/cc302158.aspx
- XMLReader, XMLWriter – abstract classes
  - Implementation of XMLReader:
    - XmlTextReader, XmlNodeReader
  - Implementation of XMLWriter:
    - XmlTextWriter, XmlNodeWriter



XML and .NET

- XmlTextReader, XmlTextWriter
  - Similar to StAX, i.e. pull parser
- XmlNodeReader, XmlNodeWriter
  - Reading of parts of DOM tree node by node

XmlTextReader

```
XmlTextReader reader = new XmlTextReader("MyFile.Xml");

while (reader.Read()) {
  switch (reader.NodeType) {
    case XmlNodeType.XmlDeclaration:
      Console.Write("<?xml version='1.0'?>");
      break;
    case XmlNodeType.Element:
      Console.Write("<" + reader.Name + ">");
      break;
    case XmlNodeType.Comment:
      Console.Write("<!--" + reader.Value + "-->");
      break;
    case XmlNodeType.EndElement:
      Console.WriteLine("</" + reader.Name + ">");
      break;
      }
  }
reader.Close();
```

XmlTextWriter

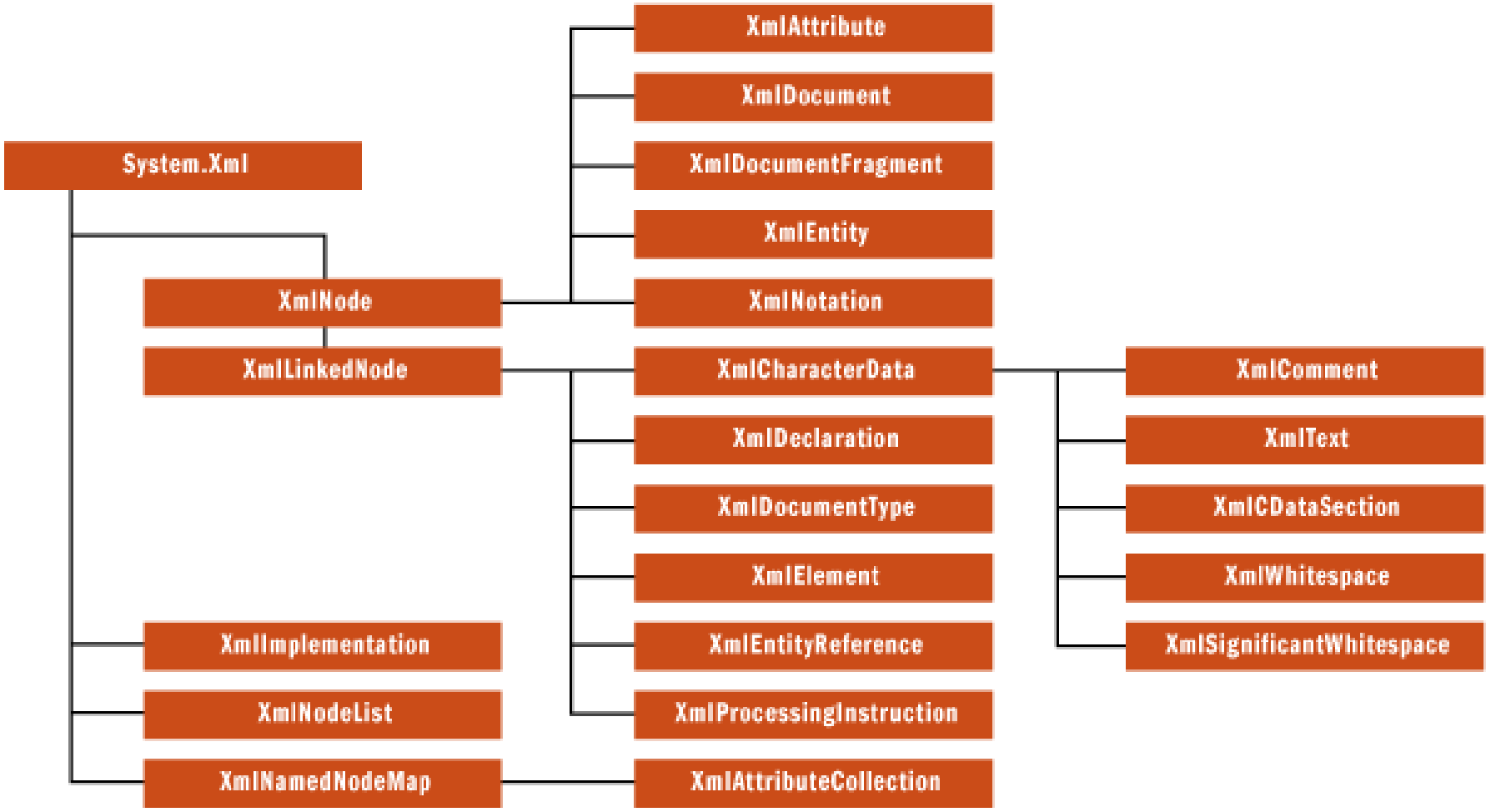
```
XmlTextWriter writer = new XmlTextWriter
("C:\\temp\\xmltest.xml", null);

writer.WriteStartDocument ();
writer.WriteComment ("sample person document");
writer.WriteProcessingInstruction ("hack", "on person");
writer.WriteStartElement ("p", "person", "urn:person");
writer.WriteElementString ("name", "joebob");
writer.WriteStartElement ("age", "");
writer.WriteAttributeString ("unit", "year");
writer.WriteString ("28");
writer.WriteEndElement ();
writer.WriteEndElement ();
writer.WriteEndDocument ();
writer.Close();
```

```
<?xml version="1.0"?>
<!--sample person document-->
<?hack on person?>
<p:person
xmlns:p="urn:person">
  <name>joebob</name>
  <age unit="year">28</age>
</p:person>
```

DOM

- Support for DOM Level 2 Core
  - Slightly modified names of classes
- Uses XmlReader and XmlWriter

DOM Hierarchy of Classes



DOM Processing

```
XmlDocument document = new XmlDocument();
document.Load(TestFileName);

List<XmlNode> nodes = new List<XmlNode>();
nodes.Add(document);

while (nodes.Count > 0) {
  XmlNode node = nodes[0];
  nodes.RemoveAt(0);
  Console.WriteLine ("{0}: {1}={2}",
    node.NodeType, node.Name, node.Value);
  if (node.Attributes != null)
    foreach (XmlNode n in node.Attributes) nodes.Add(n);
  if (node.ChildNodes != null)
    foreach (XmlNode n in node.ChildNodes) nodes.Add(n);
  }
```

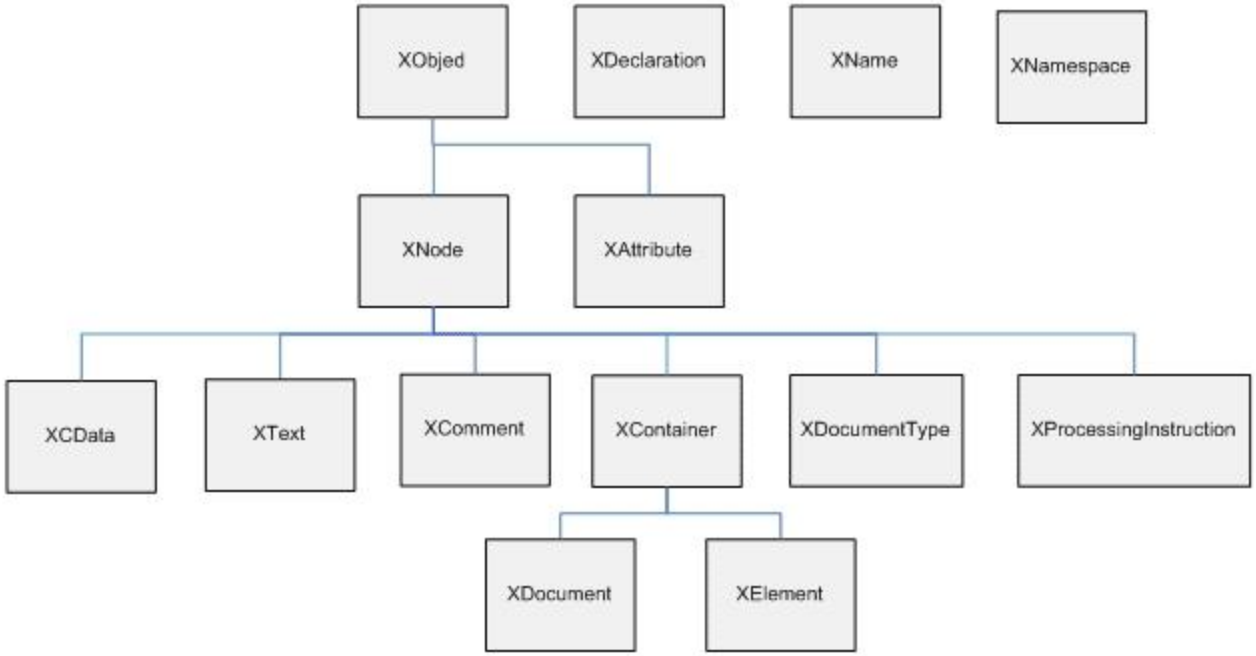
DOM Serialization

```
XmlDocument doc = new XmlDocument();

XmlNode node = doc.CreateComment("sample person document");
doc.AppendChild(node);
node = doc.CreateProcessingInstruction("hack", "on person");
doc.AppendChild(node);

node = doc.CreateElement("p", "person", "urn:person");
doc.AppendChild(node);
node = doc.CreateElement("name");
node.InnerText = "joebob";
doc.DocumentElement.AppendChild(node);
node = doc.CreateElement("age");
node.InnerText = "28";
doc.DocumentElement.AppendChild(node);

XmlTextWriter tw = new XmlTextWriter(Console.Out);
tw.Formatting = Formatting.Indented;
doc.Save(tw);
```

LINQ to XML

- LINQ = Language INtegrated Query
- LINQ to XML
  - http://msdn.microsoft.com/en-us/netframework/aa904594.aspx
- A set of .NET functions for processing of XML data
  - Reading of XML data from files and streams
  - Serialization of XML data into files and streams
  - Construction of XML data
  - Manipulation of XML data
    - Add, Remove, ReplaceWith, SetValue
  - Querying, manipulation, …

LINQ Hierarchy of Classes



LINQ – Construction, Serialization

```
XNamespace p = "urn:person";

XDocument document = new XDocument (
  new XComment ("sample person document"),
  new XProcessingInstruction ("hack", "on person"),
  new XElement ( p + "person",
    new XAttribute (XNamespace.Xmlns + "p", p),
    new XElement ("name", "joebob"),
    new XElement ("age", "28",
      new XAttribute ("unit", "year")
      )
    )
  );

document.Save(TestFileName);
```

LINQ – Reading, Querying, Manipulation

```
XDocument doc = XDocument.Load(@"c:\temp\books.xml");

foreach (XElement book in
  doc.Element("catalog").Elements("book"))
  {
  if (book.Element("genre").Value == "Computer") {
    book.Element("price").AddAfterSelf
      (new XElement("discount", "25%") );
    }
  }

Console.WriteLine(doc);
```

```
<?xml version="1.0"?>
<catalog>
   <book id="bk101">
      <author>Gambardella, Matthew</author>
      <title>XML Developer's Guide</title>
      <genre>Computer</genre>
      <price>44.95</price>
      <discount>25%</discount>
      <publish_date>2000-10-01</publish_date>
  </book>
  …
</catalog>
```

The End