XML and Semantic Web Technologies

XML Processing Models

Karen Tso
Computer based New Media Group
Institute for Computer Science
Albert-Ludwigs-University of Freiburg

7 June 2005

Overview

- XML parsers
- XML processing models
  - Text-based processing
  - Event-based processing
  - Tree-based processing
- SAX (Simple API for XML)
- DOM (Document Object Model)

XML Parsers

- Read in text file
- Check syntax
- Vaildate
- Convert document content to appropriate data structure
- Pass data structure to application programmer →API

XML API

- API (Application Programming Interface) for XML
  - Provide methods to:
    - Read and write xml documents
    - Extracting information from xml documents
    - Navigate documents in a tree structure
    - Edit xml documents

XML Processing Models

- Text-based
  - View XML document as text
- Event-based
  - XML structure generates events
  - XML processing as event handling
- Tree-based
  - XML document as a tree

Text-based Model

- Treat XML as text
- Does not have to use XML tools
- Does not make use of XML Document Structure

Event-based Model (I)

- Reacts on specific “events”
  - Start of document
  - End of document
  - Element Start tag e.g. <book>
  - Element End Tag eg. </book>
  - Attributes
  - Text
  - ……
- Parser returns the “events” information to application (Call-back mechanism)
- Process as document being read

Event-based Model (II)

- + only need to implement application-specific part
- + fast & least memory-intensive (more efficient for large document)
- + Process can be stop before end of documents
- -requires a lot more programming than Tree-based model (e.g. need to build you own data structure)
- -can not randomly access a document

Event-Based Example: SAX

```
<?xml version="1.0"?>
<books>
    <book number = “1”>
          <author>R.E.</author>
          <author>S.E.</author>
          <title lang="en">XML und DM</title>
    </book>
    <book number = “2”>
          <author>E.R.</author>
          <title lang="de">Learning XML</title>
    </book>
    <book number = “3”>
          <author>N.W.</author>
          <author>L.M.</author>
          <title lang="de">DocBook</title>
    </book>
</books>
```

1. Start document
2. Start element **books**
3. Start element **book**, Attributes **number**, value **1**
4. Start element **author**
5. Text **R.E.**
6. End element **author**
7. Start element **author**
8. Text **S.E**.
9. End element **author**
10. Start element **title**, Attributes **lang**, value **en**
11. Text **XML und DM**
12. End element **title**
13. End element **book**
14. …….
15. etc

Tree-based Model (I)

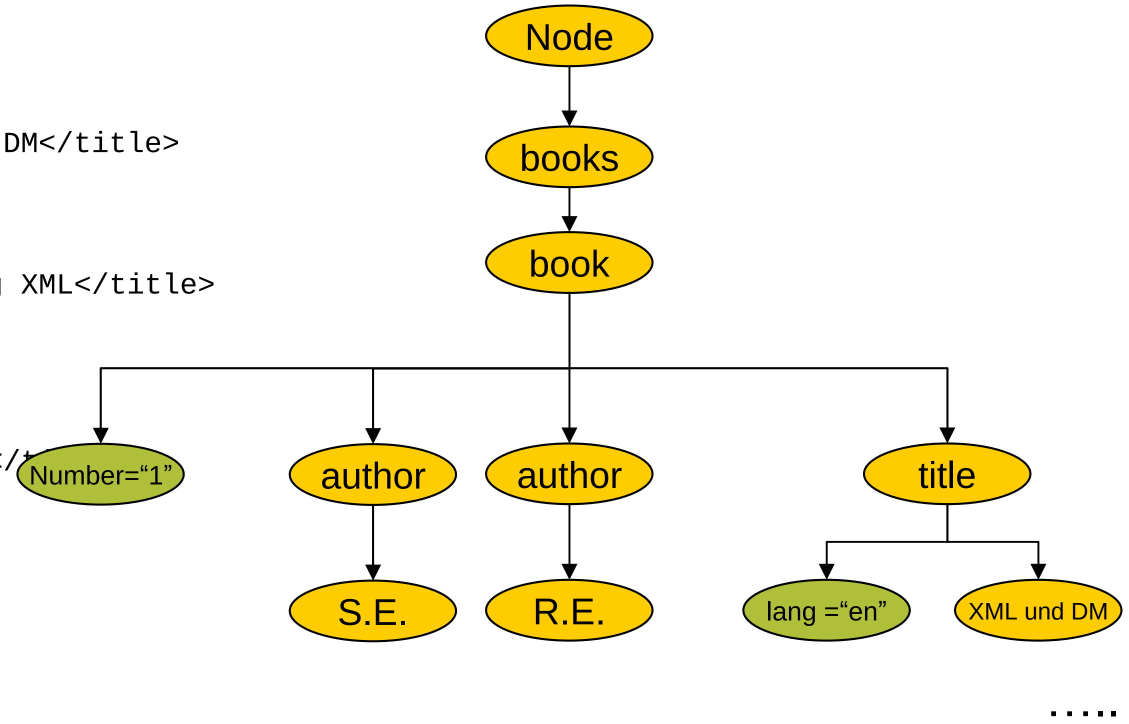
- Whole document is read and stored in memory as tree structure
- Create nodes, modify them, delete and rearrange them

Tree-based Model (II)

- + allow modification to documents
- + random access possible
- + whole document information at once
- - memory intensive (inefficient for large documents)

Tree-Based Example: DOM

```
<?xml version="1.0"?>
<books>
    <book number = “1”>
          <author>R.E.</author>
          <author>S.E.</author>
          <title lang="en">XML und DM</title>
    </book>
    <book number = “2”>
          <author>E.R.</author>
          <title lang="de">Learning XML</title>
    </book>
    <book number = “3”>
          <author>N.W.</author>
          <author>L.M.</author>
          <title lang="de">DocBook</t
    </book>
</books>
```



SAX- Simple API for XML

- Standard API for event-driven processing of XML data
- Allow parsers deliver XML info to application in little chunks
- Developed primarily to work with Java, now also with other programming languages
- SAX is NOT a parser, but implemented by many XML parsers (e.g. Xerces)

- http://xml.apache.org/dist/xerces-j/
  - Download the latest binary files
  - xercesImpl.jar

SAX- Simple API for XML

1. Main Application
2. Instantiates a parser object by a parser developer (e.g. “Xerces”)
3. Pass XML document to Parser
4. Parse the document
5. Processing document detect events like “start tags”, “end tags”…etc
6. When an event occurs, parser returns info to main application (Call-back)
7. Then calls an appropriate method to “handle” this event

- Document is processed as it is being parsed

SAX Interfaces

- XMLReader
- ContentHandler
- Attributes
- DefaultHandler (Helper Class)

Interface XMLReader

- SAX Parser
  - Reads the document
  - Generate events
  - Calls callback methods for each event
- Interface for SAX parser
  - public void parse(String uri);
    - parses the document located at uri
    - calls callback methods as it processes events
  - public void setContentHandler(ContentHandler h);

XMLReader–create SAXParser (I)

- SAX factory class
  - `org.xml.sax.helpers.XMLReaderFactory`
- `XMLReader reader = XMLReaderFactory.createXMLReader("org.apache.xerces.parsers.SAXParser");`
- `createXMLReader(parserClass)` creates an XML parser of class *`parserClass`*

XMLReader–create SAXParser (II)

- JAXP factory class
  - `javax.xml.parsers.SAXParserFactory`
    - creates new SAXParser
    - SAXParser class wraps an implementation of XMLReader

-
  ```
  SAXParserFactory f =
  SAXParserFactory.newInstance();
  SAXParser parser = f.newSAXParser();
  parser.parse( uri,handler );
  ```

XMLReader–create SAXParser (III)

- Xerces parser implementation
  - `org.apache.xerces.parsers.SAXParser`
- implements the XMLReader interface
  - `XMLReader reader = new SAXParser();`
- Not the same as `javax.xml.parsers.SAXParser`

Interface ContentHandler

- `interface ContentHandler {`

  -
    ```
    /* start / end of document */
    public void startDocument()
    public void endDocument()
    ```

  -
    ```
    /* start / end of element */
    public void startElement(String uri, String local, String q,
    Attributes atts)
    public void endElement(String uri,String local, String q)
    ```

  -
    ```
    /* text */
    public void characters(char[] ch, int start,int len)
    ```

  -
    ```
    /* processing instruction */
    public void processingInstruction(String target, String data)
    ...
    }
    ```

Example - ContentHandler

-
  ```
  public void startDocument()
  {
      System.out.println("Start of Document");
  }
  ```

-
  ```
  public void endDocument()
  {
      System.out.println(〝End of Document");
  }
  ```

Interface Attributes

- Return at method startElement
- List of attributes
- Methods to access
  - Attributes by index
  - Attribute values by name

- int getLength()
- String getQName(int index)
- String getValue(int index)
- ……

Example - Attributes

- Example of Callback Methods

```
<?xml version="1.0"?>
<book number ="1" lang="en"
   Title="XML und DM">
        <author>R.E.</author>
        <author>S.E.</author>

</book>
```

```
public void startElement(String uri, String
   local, String q, Attributes attrs)
   {
        System.out.println("Element "+ q);
        System.out.print("Attributes: ");
        for(int i=0; i<attrs.getLength();i++)
        {
        System.out.println(attrs.getQName(i)
   + " = " + attrs.getValue(i)+ ";");
        }


   }
```

Output:

```
Element book
Attributes: number = 1; lang = en; Title = XML und DM;
```

Interface DefaultHandler

- Helper class (not all methods of an interface had to be implemented)
- avoid setting up all methods for a specified event handler
- implements the interfaces
  - ContentHandler
  - EntityResolver
  - ErrorHandler
  - DTDHandler

Example

```
<?xml version="1.0"?>
<books>
    <book number = “1”>
          <author>R.E.</author>
          <author>S.E.</author>
          <title lang="en">XML und
    DM</title>
    </book>
    <book number = “2”>
          <author>E.R.</author>
          <title lang="de">Learning
    XML</title>
    </book>
    <book number = “3”>
          <author>N.W.</author>
          <author>L.M.</author>
          <title
    lang="de">DocBook</title>
    </book>
</books>
```

- We want to find the book title of author E.R.

Example:

```
//class implements callback methods(DefaultHandler)
public class Book extends DefaultHandler
{
   private String theName;
   private boolean found =false;        // indicates if name is found
   or not
   private String text = "";            // saves the contents of text
   nodes

   private XMLReader parser;

   //Constructor initialize the parser and the theName field
   public  Book(String n)       {
        theName=n;
        parser = new SAXParser();
        parser.setContentHandler(this);
……..

   }
```

Example:

```
public void characters(char[] ch, int s, int l)
{
     text = new String(ch, s, l);
}

public void endElement(String uri, String l, String q)
{
     if(q.equals("author"))
     {
             if(text.equals(theName))
                     found =true;
             else
                     found =false;
     }
     else
             if(q.equals("title") && found)
                     System.out.println(theName + ": " + text);


     text = "";
}
```

Example

```
public void parse(String uri)
   {
        try{
                System.out.println("uri is "+ uri);
                parser.parse(uri);
        }
        catch(SAXException se)
        {
                System.err.println("Parse error");
        }
        catch(IOException ioe)
        {
                System.err.println("I/O error");
        }
   }
```

Example:

```
public static void main(String[] args)
   {
        String document = args[0];      //book.xml
        String name =args[1];           //E.R.
        Book book = new Book(name);
        book.parse(document);
   }
```

Output:

```
E.R.: Learning XML
```

Overview

- XML parsers
- XML processing models
  - Text-based processing
  - Event-based processing
  - Tree-based processing
- SAX (Simple API for XML)
- DOM (Document Object Model)

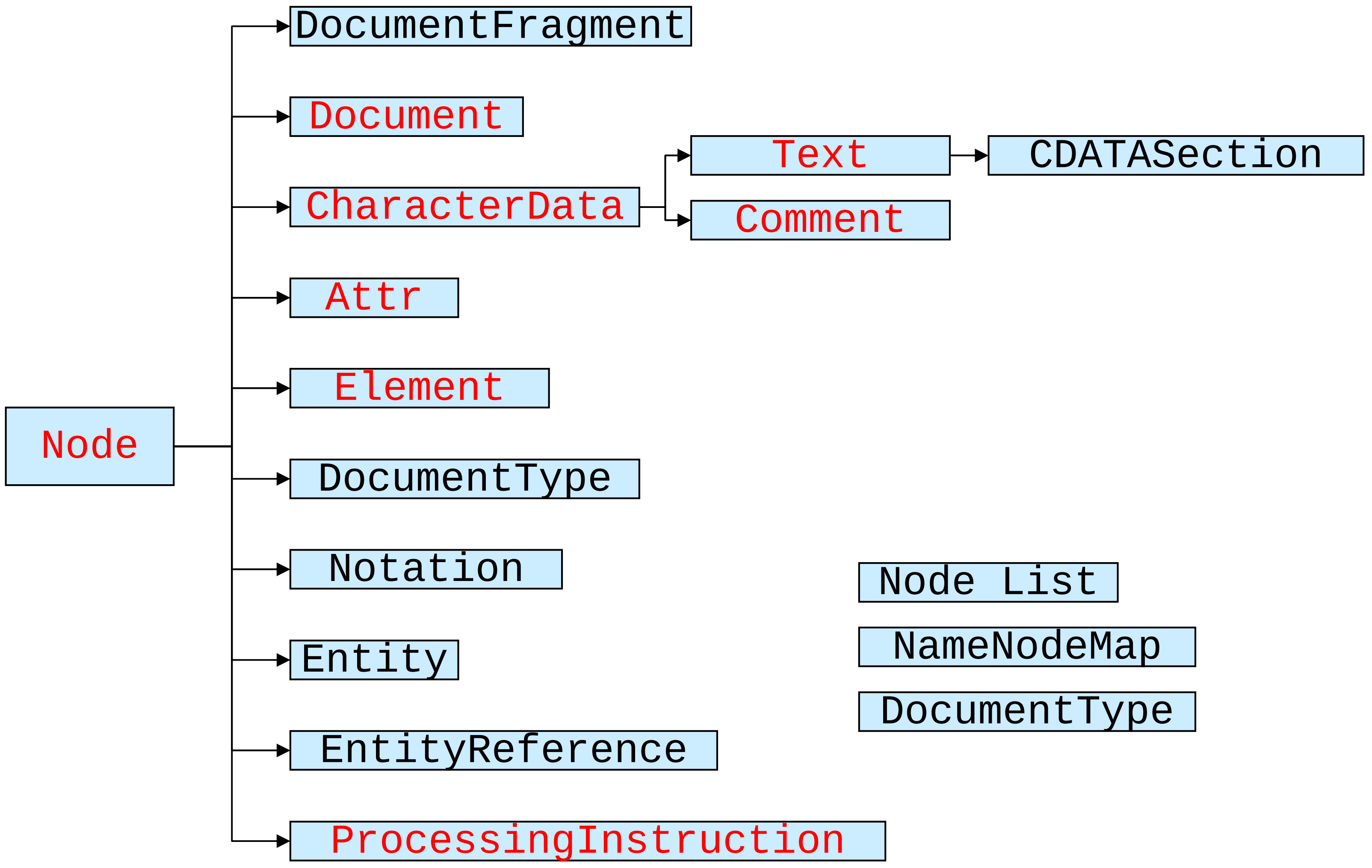
DOM (Document Object Model)

- DOM
  - Class structure
  - Methods
  - Example
  - DOM Parser: Xerces, JAXP

DOM

- Tree-structure
- Each node contains one component from an XML structure (element, attributes, text…etc)
- Object-oriented
- Language independent
  - We talk about Java API for DOM here

Interface Overview



Interface Node

- Heart of DOM scheme
- Methods Categories
  - Characteristics (type, name and value)
  - Relative location in document tree (parent, siblings and children)
  - Capability to modify its content

Node Characteristics

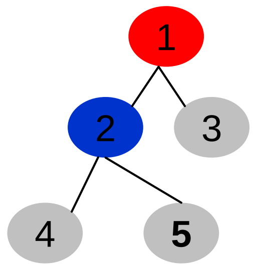
- short getNodeType();
- String getNodeName();
- String getNodeValue() throws DOMException;
- void setNodeValue(String nodeValue) throws DOMException;
- boolean hasChildNodes();
- NamedNodeMap getAttributes();
- Document getOwnerDocument();

Node Characteristics

Type	Type Number	Interface	Name	Value
ELEMENT_NODE	1	Element	Tag name	NULL
ATTRIBUTE_NODE	2	Attr	Attribute name	Attribute value
TEXT_NODE	3	Text	#text	Text string
CDATA_SECTION_NODE	4	CDATASection	#cdata-section	CDATA content
ENTITY_REFERENCE_NODE	5	EntityReference	Entity name	NULL
ENTITY_NODE	6	Entity	Entity name	NULL
PROCESSING_INSTRUCTION_NODE	7	ProcessingInstruction	Target string	Content string
COMMENT_NODE	8	Comment	Entity name	Content string
DOCUMENT_NODE	9	Document	#document	NULL
DOCUMENT_TYPE_NODE	10	DocumentType	DOCTYPE	NULL
DOCUMENT_FRAGMENT_NODE	11	DocumentFragment	#document-fragment	NULL
NOTATION_NODE	12	Notation	Notation name	NULL

Node Navigation

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



Parent: 1

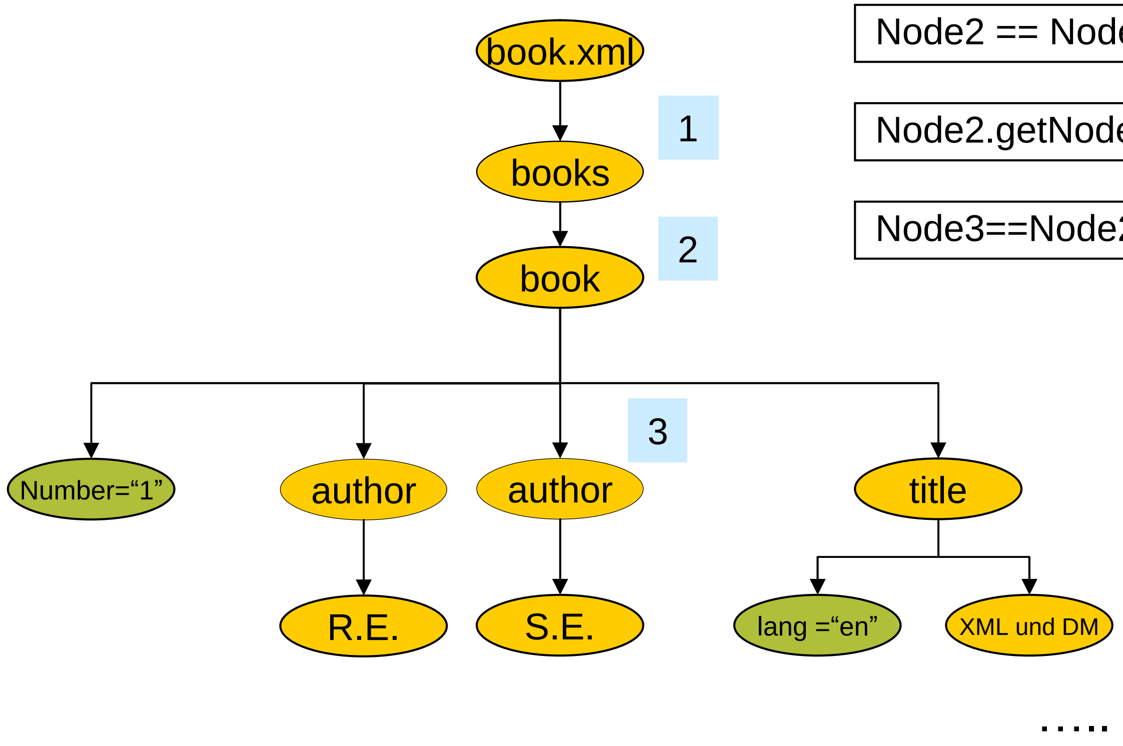
Parent’s Children: 2,3

Sibling of 2: 3

Children of 2: 4,5

Parent of 4: 2

Example:



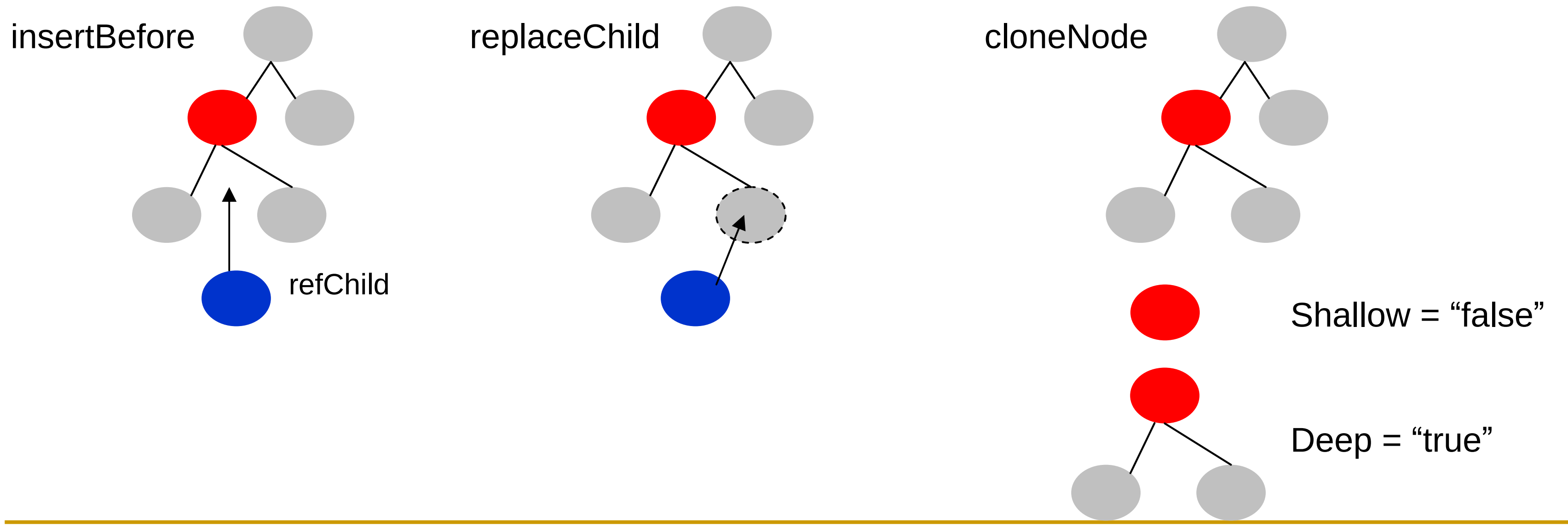
Node2 == Node1.getChildNodes().item(0)

Node2.getNodeName()=="book"

Node3==Node2.getChildNodes().item(1)

Node Manipulation

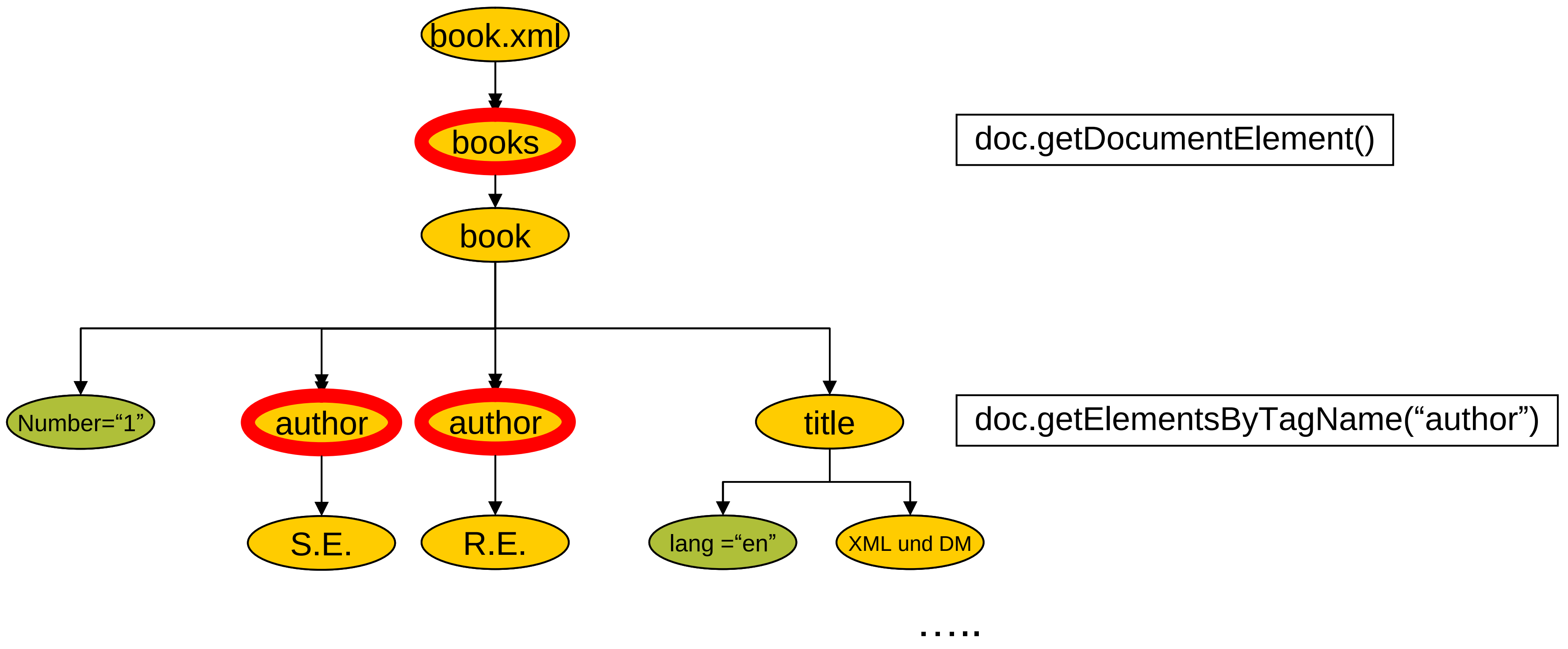
- Node removeChild(Node oldChild) throws DOMException;
- Node insertBefore(Node newChild, Node refChild) throws DOMException;
- Node appendChild(Node newChild) throws DOMException;
- Node replaceChild(Node newChild, Node oldChild) throws DOMException;
- Node cloneNode(boolean deep);



Interface Document

- public Element getDocumentElement()
- public NodeList getElementsByTagName(String tagname)
- public DocumentType getDoctype()
- public DOMImplementation getImplementation()
- public Element createElement(String tagName)
- public DocumentFragment createDocumentFragment()
- public Text createTextNode(String data)
- public Comment createComment(String data)
- public CDATASection createCDATASection(String data)
- public ProcessingInstruction createProcessingInstruction(String target, String data)
- public Attr createAttribute(String name)
- public EntityReference createEntityReference(String name)

Example:


book.xml
books
book
Number="1"
author
author
title
S.E.
R.E.
lang ="en"
XML und DM
doc.getDocumentElement()
doc.getElementsByTagName("author")
…..

Interface Node Lists

- Implements a list of Nodes
- int getLength()
  - Return the number of nodes in the list
- Node item(int index)
  - Returns the node at position `index` in the list
  - Note: the first position is 0 and last is getLength()-1

```
Node child;
NodeList children = element.getChildNodes();

for (int i = 0; i < children.getLength(); i++)
{
        child = children.item(i);
        if (child.getNodeType() == Node.ELEMENT_NODE)
        System.out.println(child.getNodeName());
}
```

Interface NamedNodeMap

- Implements a list of attribute Nodes
- int getLength()
  - Return the number of nodes in the list
- Node item(int index)
  - Returns the node at position `index` in the list
- Node getNamedItem(String name)
  - Returns the attribute node named `name`

Example - NamedNodeMap

```
<?xml version="1.0"?>
<book number ="1" lang="en"
   Title="XML und DM">
        <author>R.E.</author>
        <author>S.E.</author>

</book>
```

```
NamedNodeMap map = rootElement.getAttributes();
for(int j=0; j<map.getLength(); j++)
            System.out.print(map.item(j));
```

Output:

```
Title="XML und DM"
lang="en"
number="1"
```

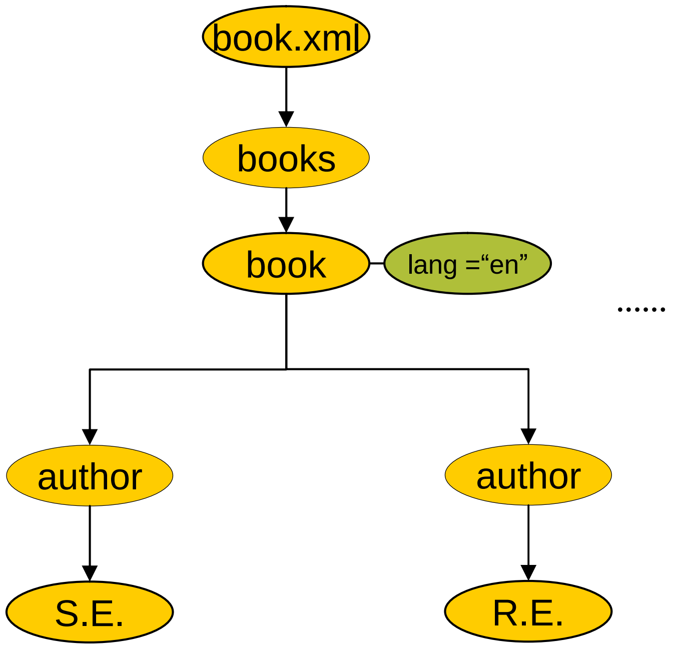
Interface Attribute

- Sub-interface of Node
- Attr nodes:
  - Not children of other nodes!
  - Associated with an element, not a separate node

Example

```
<?xml version="1.0"?>
<book number ="1" lang="en"
   Title="XML und DM">
        <author>R.E.</author>
        <author>S.E.</author>

</book>
```



Example of Attributes

- String getName();
- String getValue();
- String setValue(String value);

```
Eg. myAttrNode.setValue( “My new value“ );
```

```
Node attr = attrs.item(i);
if(attr.getNodeType() == Node.ATTRIBUTE_NODE)
System.out.println(attr.getName()+ " = " +
        attr.getValue());
```

DOM Parser

- Reads xml document
- Stores document as DOM tree (Document object consisting of Node objects)
- Objects have methods for accessing all parts of the document
- Xerces parser

DOM Parser – Xerces Parser

```
import org.w3c.dom.*;
import org.apache.xerces.parsers.DOMParser;

public static void main(String[] args)
   {
        DOMParser parser = new DOMParser();
        parser.parse(file.xml);
        Document myDoc = parser.getDocument();
        ....
   }
```

DOM Parser – JAXP

```
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

public static void main(String[] args)
   {
       DocumentBuilderFactory factory =
       DocumentBuilderFactory.newInstance();
       DocumentBuilder docBuilder = factory.newDocumentBuilder();
       Document doc = docBuilder.parse("file.xml");
       ……
}
```

DOM vs. SAX

- DOM
  - Tree-based
  - Document is stored as a tree data structure
  - Methods to access and modify tree
  - Whole document accessible at all times
- SAX
  - Event-based
  - Events are generated while parsing
  - Programmer implements event handlers /callback methods
  - Only the current node is accessible
    - Programmer implement their data structure to store info