Parsers

SAX, Parsing, TREE BASED PARSERS: DOM, JavaScript examples of DOM , More example in Java

SAX 2.0.2

Event-based APIs

SAX 2.0.2

- SAX is the *Simple API for XML*
- SAX was the first widely adopted API for XML in **Java**, and is a "de facto" standard
- Can be downloaded from: http://sourceforge.net/projects/sax/files/sax/
- You can write very fast SAX parsers with SAX
  - No memory to allocate, data structures to link
  - "Fire and forget"
  - It is useful for large documents
    - Loading the whole document into memory is prohibitive
  - It is easy to use

SAX 2.0.2

- Example:

```
<?xml version="1.0"?>
<doc>
        <para>Hello, world!</para>
</doc>
```

- An event-based interface will break the structure of this document down into a series of linear events, such as these:

```
start document
start element: doc
start element: para
characters: Hello, world!
end element: para
end element: doc
end document
```

  - An application handles these events just as it would handle events from a graphical user interface (GUI)

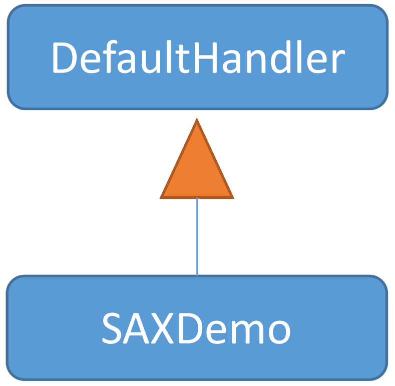
- there is no need to cache the entire document in memory

SAX 2.0.2

- Event-Based Parsing
- Document
  - <students>
    - <student id="0001“></student>
  - </students>
- ContentHandler
  - startDocument(); (1st event)
  - startElement("students", {});
  - characters("\n ");
  - startElement("student", {("id", "0001")};
  - endElement("student");
  - endElement("students");
  - endDocument();

- Notes:
  - SAX Allows you to read XML file and each time it encounters a tag it triggers an EVENT and you write code to HANDLE the event
  - Your java class will look like this:

public class SAXDemo extends DefaultHandler



Exact output

```
startDocument();
startElement("studentsRecords", {});
characters("
                    ");
startElement("students", {});
characters("
                                        ");
startElement("student", {});
characters(" id="0001" ");
endElement("student");
characters("
                    ");
endElement("students");
characters("
");
endElement("studentsRecords");
endDocument();
```

books.xml

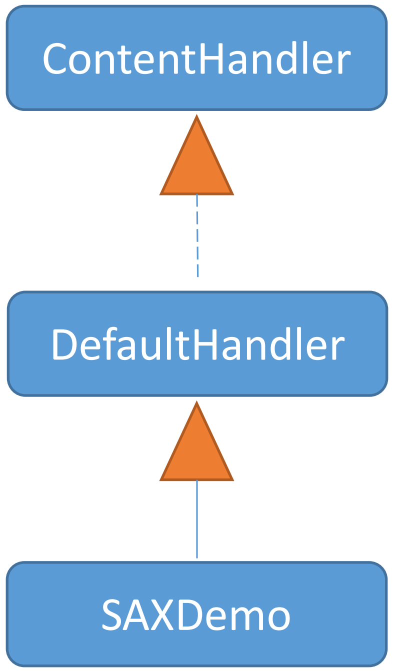
D:\
ing\ITM-466\SAX

SAX Code

ContentHandler is an INTERFACE and DefaultHandler is available as a convenience base class for SAX2 applications: it provides default implementations for all of the callbacks in the four core SAX2 handler classes:

EntityResolver
DTDHandler
ContentHandler
ErrorHandler

API:
http://www.saxproject.org/apidoc/org/xml/sax/**ContentHandler.html**
http://www.saxproject.org/apidoc/org/xml/sax/helpers/**DefaultHandler.html**



PASRSING

Parsing

- XML, parsing means:
  1. reading an XML document,
  2. identifying the various components, and
  3. making it available to an application

- In order to parse a document , you need to be able to specify exactly what it contains
- XML specification does this for XML using a grammar in Backus-Naur Form (BNF)
  - “BNF (Backus Normal Form or Backus–Naur Form) is a notation technique for **context-free grammars**, often used to describe the syntax of languages used in computing, such as computer programming languages, **document formats**, instruction sets and communication protocols”

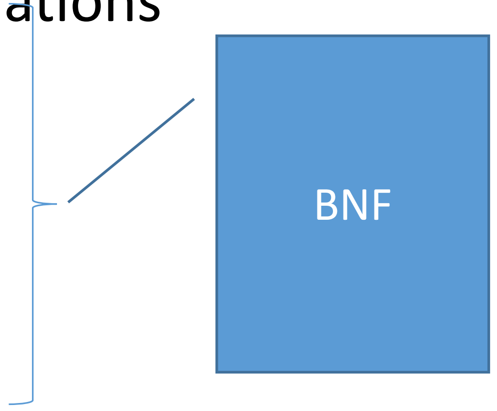
Parser

- A grammar describes a language through a series of rules
  - A rule describes how to produce a something (e.g., a start tag) by assembling characters and other non-terminal symbols

- Made up of
  - **non-terminal** symbols
  - **terminal symbols** (data that is taken literally)

Arithmetic Parser

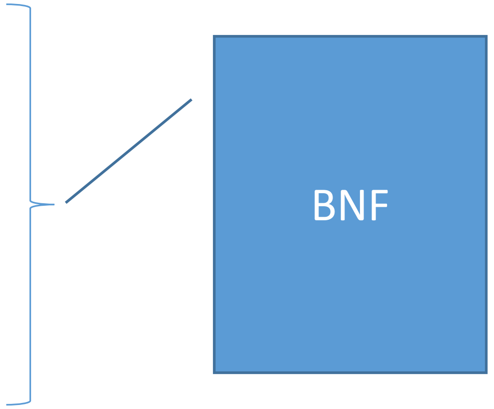
- Arithmetic Parser: A grammar for arithmetic equations
  - Eqn ::= Term '=' Term
  - Term ::= '(' Term Op Term ')' | Value
  - Op ::= '+' | '-' | '/' | '*'
  - Value ::= <any number>



- Produces
  - (4 + 3) = 7
  - (1 + 2) = (3 – 0)
  - ((10 / 2) + 1) = (3 * 2)
  - 4 = 5
  - …

XML Parser

- A (much simplified) grammar for XML
  - element ::= STag content Etag
  - content ::= (element | CharData)*
  - STag ::= '<' Name '>'
  - ETag ::= '<' '/' Name '>'



- where
  - **Name** is one or more characters excluding > and **CharData** is zero or more characters excluding <.

XML Parser

- Tokenizing and Recognizing

- Tokenizing
  - Creates tokens from the character stream
  - Element name, equal sign, start tag

- Recognizing
  - Understands the syntax of the document and checks for correctness
  - Builds a syntax tree

PARSERS

TREE BASED PARSERS: DOM

DOM - *Document Object Model*

- *"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document.“*
- The DOM is separated into 3 different parts / levels:
  - Core DOM - standard model for any structured document
    - defines the **objects and properties** of all document elements, and the **methods** (interface) to access them
  - XML DOM - standard model for XML documents
    - The XML DOM defines a standard way for accessing and manipulating XML documents.
  - HTML DOM - standard model for HTML documents
    - The HTML DOM defines the **objects and properties** of all HTML elements, and the **methods** (interface) to access them.

XML DOM

- The XML DOM defines the **objects and properties** of all XML elements, and the **methods** (interface) to access them.
- The XML DOM is:
  - A standard object model for XML
  - A standard programming interface for XML
  - Platform- and language-independent
  - A W3C standard
- **The XML DOM is a standard for how to get, change, add, or delete XML elements**

XML Node

- The XML DOM
  - everything in an XML document is a node
    - The entire document is a **document node**
    - Every XML element is an **element node**
    - The text in the XML elements are **text nodes**
    - Every attribute is an **attribute node**
    - Comments are **comment nodes**

- XML Node presents an XML document as a **tree-structure**

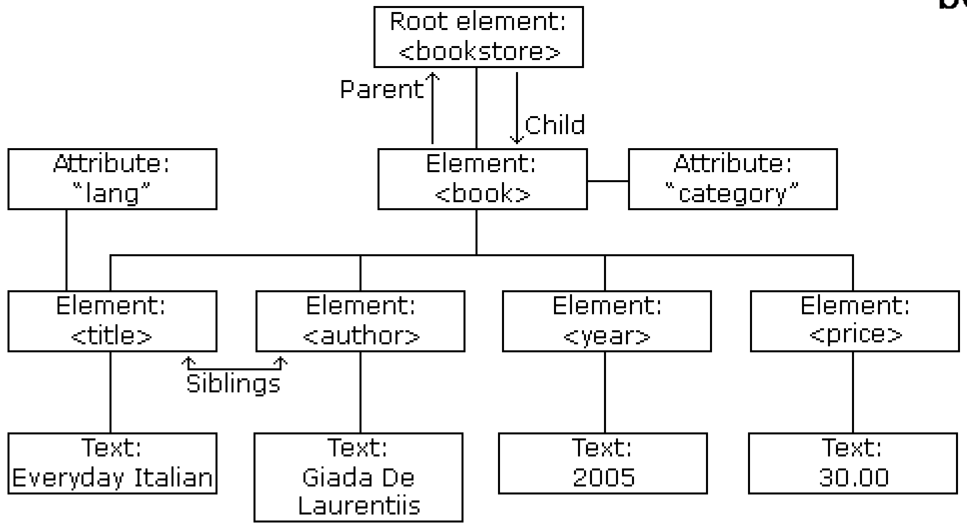
XML DOM Tree Example: books.xml

- <?xml version="1.0" encoding="UTF-8"?>
- <**bookstore**>
  - <book category="cooking">
    - <title lang="en">Everyday Italian</title>
    - <author>Giada De Laurentiis</author>
    - <year>2005</year>
    - <price>30.00</price>
  - </book>
  - <book category="children">
    - <title lang="en">Harry Potter</title>
    - <author>J K. Rowling</author>
    - <year>2005</year>
    - <price>29.99</price>
  - </book>
  - <book category="web">
    - <title lang="en">XQuery Kick Start</title>
    - <author>James McGovern</author>
    - <author>Per Bothner</author>
    - <author>Kurt Cagle</author>
    - <author>James Linn</author>
    - <author>Vaidyanathan Nagarajan</author>
    - <year>2003</year>
    - <price>49.99</price>
  - </book>
  - <book category="web" cover="paperback">
    - <title lang="en">Learning XML</title>
    - <author>Erik T. Ray</author>
    - <year>2003</year>
    - <price>39.95</price>
  - </book>
- </**bookstore**>

- The root node in the XML above is named <**bookstore**>.
  - All other nodes in the document are contained within <bookstore>
- The root node <bookstore> holds four **<book>** nodes.
  - The first <book> node holds four nodes:
    - <title>,
    - <author>,
    - <year>, and
    - <price>,
  - which contains one text node each,
    - *"Everyday Italian",*
    - *"Giada De Laurentiis",*
    - *"2005", and*
    - *"30.00".*

XML DOM Tree Example: books.xml

- <?xml version="1.0" encoding="UTF-8"?>
- <**bookstore**>
  - <book category="cooking">
    - <title lang="en">Everyday Italian</title>
    - <author>Giada De Laurentiis</author>
    - <year>2005</year>
    - <price>30.00</price>
  - </book>
  - <book category="children">
    - <title lang="en">Harry Potter</title>
    - <author>J K. Rowling</author>
    - <year>2005</year>
    - <price>29.99</price>
  - </book>
  - <book category="web">
    - <title lang="en">XQuery Kick Start</title>
    - <author>James McGovern</author>
    - <author>Per Bothner</author>
    - <author>Kurt Cagle</author>
    - <author>James Linn</author>
    - <author>Vaidyanathan Nagarajan</author>
    - <year>2003</year>
    - <price>49.99</price>
  - </book>
  - <book category="web" cover="paperback">
    - <title lang="en">Learning XML</title>
    - <author>Erik T. Ray</author>
    - <year>2003</year>
    - <price>39.95</price>
  - </book>
- </**bookstore**>



root node <bookstore>

The tree structure is called a **node-tree.**

- Every node, except the root, has exactly one parent node
- A node can have any number of children
- A **leaf** is a node with no children
- **Siblings** are nodes with the same parent

XML DOM Parser

- Because the XML data is structured in a tree form, it can be traversed without knowing the exact structure of the tree and without knowing the type of data contained within

- **XML DOM Parser** is used to parse the document:

XML DOM Parser

- The XML DOM parser converts XML into an **XML DOM object** that can be accessed with JavaScript, Java, .Net, etc.
  - Before an XML document can be accessed and manipulated, it must be loaded into an XML DOM object
  - The XML DOM contains methods to traverse XML trees:
    - access,
    - insert, and
    - delete nodes.
  - Most browsers have a built-in XML parser.

Load an XML Document

- JavaScript

```
function loadXMLDoc(filename) {
    if (window.XMLHttpRequest)   {
      xhttp = new XMLHttpRequest();
    }

    xhttp.open("GET","books.xml",false);
    xhttp.send();                                   //load it
    xmlDoc = xhttp.responseXML;                      //get it in local object
}
```

The function above can be stored in the <head> section of an HTML page, and called from a script in the page.

JavaScript or other programming languages

DOM XML parser

- DOM parser parses the entire XML document and **loads it into memory**; then models it in a “**TREE**” structure for easy traversal or manipulation.
  - **<?xml** version="1.0"**?>**
  - **<company>**
    - **<staff** id="1001"**>**
      - **<firstname>**yong**</firstname>**
      - **<lastname>**mook kim**</lastname>**
      - **<nickname>**mkyong**</nickname>**
      - **<salary>**100000**</salary>**
    - **</staff>**
    - **<staff** id="2001"**>**
      - **<firstname>**low**</firstname>**
      - **<lastname>**yin fong**</lastname>**
      - **<nickname>**fong fong**</nickname>**
      - **<salary>**200000**</salary>**
    - **</staff>**
  - **</company>**

JavaScript examples of DOM

- The DOM models XML as a set of node objects. The nodes can be accessed with JavaScript or other programming languages
- **Properties** are often referred to as something that is (i.e. node name is "book").
- **Methods** are often referred to as something that is done (i.e. delete "book").

JavaScript examples of DOM

- XML DOM Properties
  - x.nodeName - the name of x
  - x.nodeValue - the value of x
  - x.parentNode - the parent node of x
  - x.childNodes - the child nodes of x
  - x.attributes - the attributes nodes of x

  - Note: In the list above, x is a node object.

JavaScript examples of DOM

- XML DOM Methods
- x.getElementsByTagName(*name*)
  - get all elements with a specified tag name
- x.appendChild(*node*)
  - insert a child node to x
- x.removeChild(*node*)
  - remove a child node from x

- Note: In the list above, x is a node object.

JavaScript examples of DOM

```
<book category="cooking">
        <title lang="en">Everyday Italian</title>
        <author>Giada De Laurentiis</author>
        <year>2005</year>
        <price>30.00</price>
</book>
```

- Example
- The JavaScript code to get the text from the first <title> element in books.xml:
  - txt=**xmlDoc**.getElementsByTagName("title")[0].childNodes[0].nodeValue

    - **xmlDoc** - the XML DOM object created by the parser.
    - **getElementsByTagName("title")[0]** - the first <**title**> element
    - **childNodes[0]** - the first child of the <title> element (the text node)
    - **nodeValue** - the value of the node (the text itself)
  - After the execution of the statement, txt will hold the value "Everyday Italian“

  - See http://www.w3schools.com/dom/dom_nodes_access.asp for a complete example on how to use JavaScript to parse XML DOM docs

Example in Java

SAX Parsers

SAXDemo.java

JAVA DOM Parsers

ReadXMLFile1.java

ReadXMLFile2.java

XML files

staff.xml

students.xml

books.xml