

Computer Science E-259

XML with J2EE

Lecture 2: XML 1.1 and SAX 2.0.2

9 February 2006

David J. Malan
`malan@post.harvard.edu`

1

Copyright © 2006, David J. Malan <malan@post.harvard.edu>. All Rights Reserved.

Computer Science E-259

Last Time

- Bingo Destinations
- Computer Science E-259
- J2EE
- XML
 - What
 - Who
 - When
 - How
 - Why
- Computer Science E-259

2

Copyright © 2006, David J. Malan <malan@post.harvard.edu>. All Rights Reserved.

Computer Science E-259

This Time

- XML 1.1
- SAX 2.0.2
- JAXP 1.3 and Xerces 2.7.1
- Parsing
- My First XML Parser

3

Copyright © 2006, David J. Malan <malan@post.harvard.edu>. All Rights Reserved.

XML 1.1

A Representative Document

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE students SYSTEM "student.dtd">

<!-- This is an XML document that describes students -->
<?studentdb displaydesc="true"?>
<students>
  <student id="0001">
    <name>Jim Bob</name>
    <status>graduate</status>
    <dorm/>
    <major>Computer Science &amp; Music</major>
    <description>
      <![CDATA[ <h1>Jim Bob!</h1>
        Hi my name is jim. I look like
         ]]>
    </description>
  </student>
  <student id="0002">
    ...
  </student>
</students>
```

4

Copyright © 2006, David J. Malan <malan@post.harvard.edu>. All Rights Reserved.

XML 1.1

XML Declaration

```
<?xml version="1.0" encoding="UTF-8"?>
```

- Optional
- Must appear at the very top of an XML document
- Used to indicate the version of the specification to which the document conforms (and whether the document is “standalone”)
- Used to indicate the character encoding of the document
 - UTF-8
 - UTF-16
 - iso-8859-1
 - ...

5

Copyright © 2006, David J. Malan <malan@post.harvard.edu>. All Rights Reserved.

XML 1.1

DOCTYPE

```
<!DOCTYPE students SYSTEM "students.dtd">
```

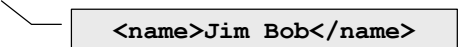
- References a Document Type Definition (DTD)
- Can refer to an external DTD file or include some DTD information within the tag itself
- DTD is the original mechanism for specifying the schema of an XML document
 - Inherited in part from SGML
 - Arcane syntax
 - Limited expressive functionality
- More in Lecture 8...

6

Copyright © 2006, David J. Malan <malan@post.harvard.edu>. All Rights Reserved.

XML 1.1

Elements



```
<name>Jim Bob</name>
```

- Main structure in an XML document
- Only one root element allowed
- Start Tag
 - Allows specification of zero or more attributes
`<student id="0001" ...>`
- End Tag
 - Must match name, case, and nesting level of start tag
`</student>`
- Name must start with letter or underscore and can contain only letters, numbers, hyphens, periods, and underscores


XML 1.1

Element

- Element Content
`<student>
 <status>...</status>
</student>`
- Parsed Character Data (aka PCDATA, aka Text)
`<name>Jim Bob</name>`
- Mixed Content
`<name>Jim <initial>J</initial> Bob</name>`
- No Content
`<dorm/>`

XML 1.1

Attributes




```
<student id="0001">
```

- Name
 - Must start with letter or underscore and can contain only letters, numbers, hyphens, periods, and underscores
- Value
 - Can be of several types, but is almost always a string
 - Must be quoted
 - `title="Lecture 2"`
 - `match='item="baseball bat"'`
 - Cannot contain `<` or `&` (by itself)

XML 1.1

PCDATA



```
Jim Bob
```

- Text that appears as the content of an element
- Can reference entities
- Cannot contain `<` or `&` (by itself)

XML 1.1

Entities

↙ `&`

- Used to “escape” content or include content that is hard to enter or repeated frequently
 - Somewhat like macros
- Five pre-defined entities
 - `&`; `<`; `>`; `'`; `"`;
- Character entities can refer to a single character by unicode number
 - `©` ≡ ©
- Must be declared to be legal
 - `<!ENTITY nbsp " ">`
- Cannot refer to themselves

11

Copyright © 2006, David J. Malan <malan@post.harvard.edu>. All Rights Reserved.

XML 1.1

CDATA

↙ `<![CDATA[<h1>Jim Bob!</h1> ...]]>`

- Parsed in “one chunk” by the XML parser
- Data within is not checked for subelements, entities, *etc.*
- Allows you to include badly formed markup or character data that would cause a problem during parsing
- Examples
 - Including HTML tags in an XML document
 - Used in XSLT to write out non-XML text

12

Copyright © 2006, David J. Malan <malan@post.harvard.edu>. All Rights Reserved.

XML 1.1

Comments

```
<!-- This is ... -->
```

- Can include any text inside a comment to make it easier for human readers to understand your document
- Generally not available to applications reading the document
- Always begin with `<!--` and end with `-->`
- Cannot contain `--`

13

Copyright © 2006, David J. Malan <malan@post.harvard.edu>. All Rights Reserved.

XML 1.1

Processing Instructions

```
<?studentdb displaydesc="true"?>
```

- “Sticky notes” to applications processing an XML document that explain how to handle content
- The target portion (e.g., `studentdb`) of a PI indicates the application that is to process this instruction; cannot start with “xml”
- The remainder of the PI can be any text that gives instructions to the application
- Examples
 - Instructions to an application to display different versions of an image
 - Instructions to an application to suppress display of certain content
 - ...

14

Copyright © 2006, David J. Malan <malan@post.harvard.edu>. All Rights Reserved.

SAX 2.0.2

A Sample Document

```
<students>
  <student id="0001"/>
</students>
```

15

Copyright © 2006, David J. Malan <malan@post.harvard.edu>. All Rights Reserved.

SAX 2.0.2

Event-Based Parsing

Document

```
<students>
  <student id="0001"/>
</students>
```

ContentHandler

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

16

Copyright © 2006, David J. Malan <malan@post.harvard.edu>. All Rights Reserved.

JAXP 1.3 and Xerces 2.7.1

SAXDemo

```
javax.xml.parsers.SAXParserFactory
    javax.xml.parsers.SAXParser
        org.xml.sax.*
            org.xml.sax.helpers.*
                ...
```

17

Copyright © 2006, David J. Malan <malan@post.harvard.edu>. All Rights Reserved.

Parsing

Definition

- In linguistics, to divide language into small components that can be analyzed. For example, parsing this sentence would involve dividing it into words and phrases and identifying the type of each component (*e.g.*, verb, adjective, or noun)
- For XML, parsing means reading an XML document, identifying the various components, and making it available to an application

18

Copyright © 2006, David J. Malan <malan@post.harvard.edu>. All Rights Reserved.

Parsing

Grammars in Backus-Naur Form

- 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)
- 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)

19

Copyright © 2006, David J. Malan <malan@post.harvard.edu>. All Rights Reserved.

Parsing

Arithmetic

- A grammar for arithmetic equations

```
Eqn    ::= Term '=' Term
Term   ::= '(' Term Op Term ')' | Value
Op     ::= '+' | '-' | '/' | '*'
Value  ::= <any number>
```
- Produces
 - $(4 + 3) = 7$
 - $(1 + 2) = (3 - 0)$
 - $((10 / 2) + 1) = (3 * 2)$
 - $4 = 5$
 - ...

20

Copyright © 2006, David J. Malan <malan@post.harvard.edu>. All Rights Reserved.

Parsing

XML

- 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 <.

21

Copyright © 2006, David J. Malan <malan@post.harvard.edu>. All Rights Reserved.

My First 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
- In **mf.XMLParser**, there will be no clear distinction between tokenizing and recognizing

22

Copyright © 2006, David J. Malan <malan@post.harvard.edu>. All Rights Reserved.

My First XML Parser

Recursive Descent Parsing

- XML's grammar works well with a parsing technique known as recursive descent parsing
- Basically:
 - You write a function that is responsible for parsing every non-terminal in the grammar
 - You assume that the document matches the grammar
 - The correct alternation in a rule can be determined by examining a few tell-tale starting characters (lookahead)
 - You recursively parse the document, calling each non-terminal parsing function as dictated by the grammar
 - Use exception handling to handle errors when they occur deep in the recursive call tree

23

Copyright © 2006, David J. Malan <malan@post.harvard.edu>. All Rights Reserved.

My First XML Parser

Source Code

```
cscie259.project1.mf.*
```

24

Copyright © 2006, David J. Malan <malan@post.harvard.edu>. All Rights Reserved.

Computer Science E-259

Next Time

- The SAX API has a number of important advantages...
 - You can write very fast SAX parsers
 - 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
- ...but it doesn't solve every problem
 - Need to have an internal data structure for some applications
 - To follow links in information (especially backwards ones)
 - To perform operations that require having multiple pieces of the document at the same time
- Enter the Document Object Model (DOM)...

25

Copyright © 2006, David J. Malan <malan@post.harvard.edu>. All Rights Reserved.

Computer Science E-259

XML with J2EE

Lecture 2: XML 1.1 and SAX 2.0.2

9 February 2006

David J. Malan

malan@post.harvard.edu

26

Copyright © 2006, David J. Malan <malan@post.harvard.edu>. All Rights Reserved.