CS122B: Projects in Databases and Web Applications
Spring 2018

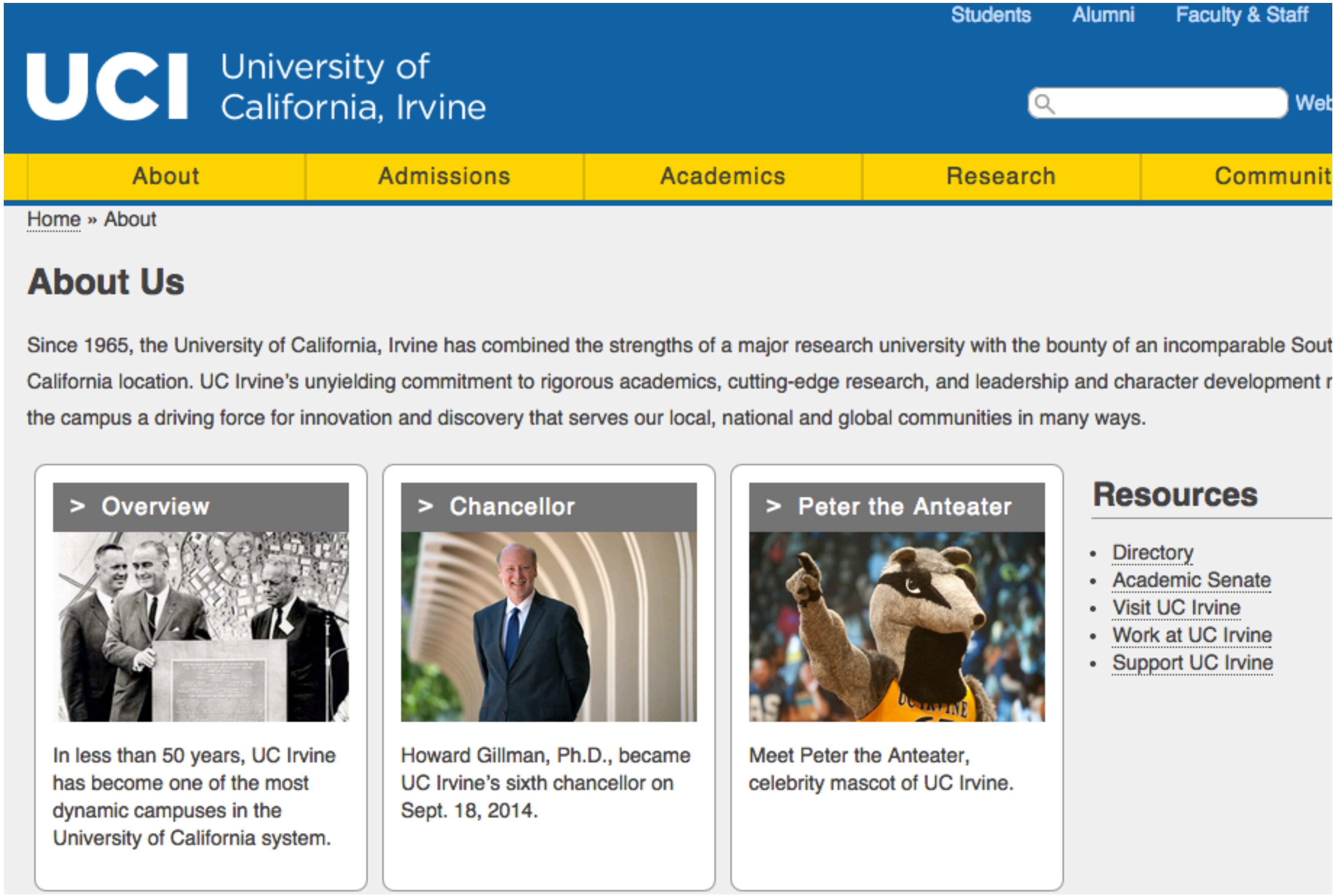
Notes 11: XML

Professor Chen Li
Department of Computer Science
UC Irvine

Outline

- XML basics
- DTD
- Parsing XML (SAX/DOM)

An HTML document



Students Alumni Faculty & Staff

University of California, Irvine

About Admissions Academics Research Communit

Home » About

About Us

Since 1965, the University of California, Irvine has combined the strengths of a major research university with the bounty of an incomparable Sout California location. UC Irvine's unyielding commitment to rigorous academics, cutting-edge research, and leadership and character development r the campus a driving force for innovation and discovery that serves our local, national and global communities in many ways.

> Overview

In less than 50 years, UC Irvine has become one of the most dynamic campuses in the University of California system.

> Chancellor

Howard Gillman, Ph.D., became UC Irvine's sixth chancellor on Sept. 18, 2014.

> Peter the Anteater

Meet Peter the Anteater, celebrity mascot of UC Irvine.

Resources

- Directory
- Academic Senate
- Visit UC Irvine
- Work at UC Irvine
- Support UC Irvine

HTML code

```
<nav id="nav-main" role="navigation">
<ul class="content-container clearfix">
<li class="nav-main-about">
<a class="nav-main-item" href="/about/index.php">About</a>
<ul class="nav-main-menu">
<li>
<ul class="submenu">
<li><a class="nav-main-item" href="/facts/index.php">Overview</a></li>
<li><a class="nav-main-item"
    href="/distinctions/index.php">Distinctions</a></li>
<li><a class="nav-main-item" href="/facts/campus-data.php">Campus
    Data</a></li>
<li><a class="nav-main-item"
    href="http://www.strategicplan.uci.edu/">Strategic Plan</a></li>
<li><a class="nav-main-item" href="/innovation/index.php">Spotlights on
    Innovation</a></li>
</ul>
</li>
```

What is the problem?

- To do more fancy things with documents:
  - need to make their logical structure explicit.
- Otherwise, software applications
  - do not know what is what
  - do not have any handle over documents.

An XML document

```
<?xml version="1.0" ?>
<bib>
  <vendor id="id3_4">
   <name>QuickBooks</name>
   <email>booksales@quickbooks.com</email>
   <phone>1-800-333-9999</phone>
   <book>
    <title>Inorganic Chemistry</title>
      <publisher>Brooks/Cole Publishing</publisher>
      <year>1991</year>
    <author>
     <firstname>James</firstname>
      <lastname>Bowser</lastname>
    </author>
   <price>43.72</price>
   </book>
  </vendor>
</bib>
```

What is XML?

- eXtensible Markup Language
- Data are identified using tags (identifiers enclosed in angle brackets: <...>)
- Collectively, the tags are known as "markup"
- XML tags tell you what the data *means*, rather than how to display it

XML versus relational

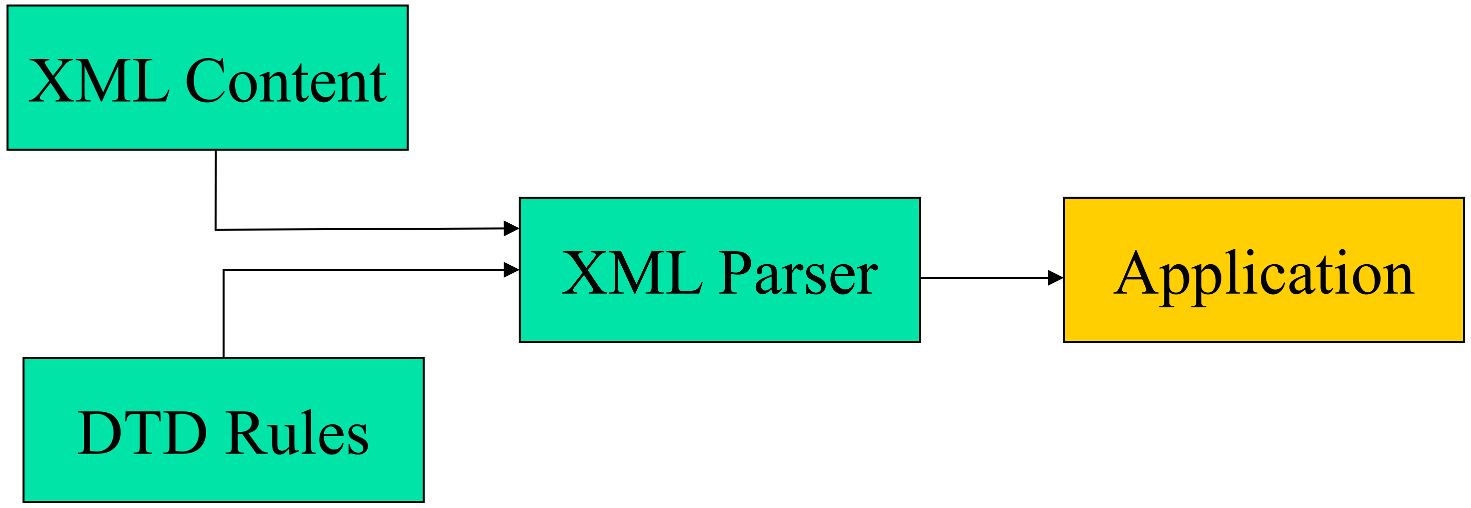
- Relational: structured
- XML: semi-structured
- Plain text file: unstructured

How does XML work?

- XML allows developers to write their own Document Type Definitions (DTD)
- DTD is a markup language's rule book that describes the sets of tags and attributes that is used to describe specific content
- If you want to use a certain tag, then it must first be defined in DTD

Key Components in XML

- Three generic components, and one customizable component



Meta Markup Language

- Not a language – but a way of specifying other languages
- Meta-markup language – gives the rules by which other markup languages can be written
- Portable - platform independent

Markup Languages

- Presentation based:
  - Markup languages that describe information for presentation for human consumption
- Content based:
  - Describe information that is of interest to another computer application

HTML and XML

- HTML tag says "display this data in bold font"
  - `<b>...</b>`
- XML tag acts like a field name in your program
- It puts a label on a piece of data that identifies it
  - `<message>...</message>`

Simple Example

- XML data for a messaging application:

```
<message>
   <to>you@yourAddress.com</to>
   <from>me@myAddress.com</from>
   <text>
   Why is it good? Let me count the ways...
   </text>
</message>
```

Element

- Data between the tag and its matching end tag defines an element of the data
- Comment:
  - `<!-- This is a comment -->`

Example

```
<!-- Using attributes   -->
<message to="you@yourAddress.com"
  from="me@myAddress.com">
<text>Why is it good? Let me count the
  ways...</text>
</message>
```

Attributes

- Tags can also contain attributes
- Attributes contain additional information included as part of the tag, within the tag's angle brackets
- Attribute name is followed by an equality sign and the attribute value

Other Basics

- White space is essentially irrelevant
- Commas between attributes are not ignored - if present, they generate an error
- Case sensitive: “message” and “MESSAGE” are different

Well Formed XML

- Every tag has a closing tag
- XML represents hierarchical data structures having one tag to contain others
  - Tags have to be completely nested
- Correct:
  - `<message>..<to>..</to>..</message>`
- Incorrect
  - `<message>..<to>..</message>..</to>`

Empty Tag

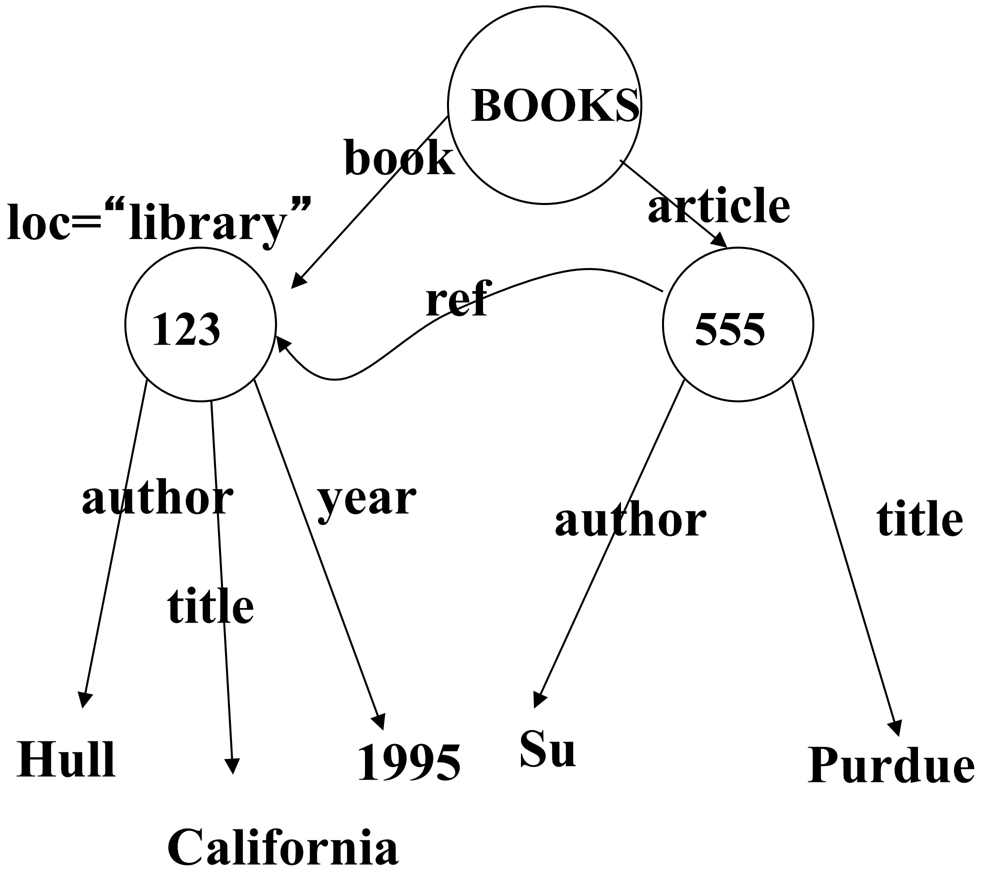
- Empty tag is used when it makes sense to have a tag that stands by itself and doesn't enclose any content - a "flag"
  - You can create an empty tag by ending it with />
  - **<flag/>**

Example

```
<message to="you@yourAddress.com"
  from="me@myAddress.com"
  subject="XML is good"> <flag/>
  <text> Why is it good? Let me count the
  ways...
  </text>
</message>
```

Tree representation

```
<BOOKS>
<book id="123" loc="library">
   <author>Hull</author>
   <title>California</title>
   <year> 1995 </year>
</book>
<article id="555" ref="123">
   <author>Su</author>
   <title> Purdue</title>
</article>
</BOOKS>
```



Special Characters

- Some characters need to be escaped because they have special significance:
  - < &lt;
  - > &gt;
  - & &amp;
  - ' &apos;
  - " &quot;
- If they were not escaped - would be processed as markup by XML engine

Prolog in XML Files

- XML file always starts with a prolog
- The minimal prolog contains a declaration that identifies the document as an XML document:

```
<?xml version="1.0"?>
```

- The declaration may also contain additional information
  - **version** - version of the XML used in the data
  - **encoding** - Identifies the character set used
  - **standalone** - whether the document references an external entity or data type specification

An Example

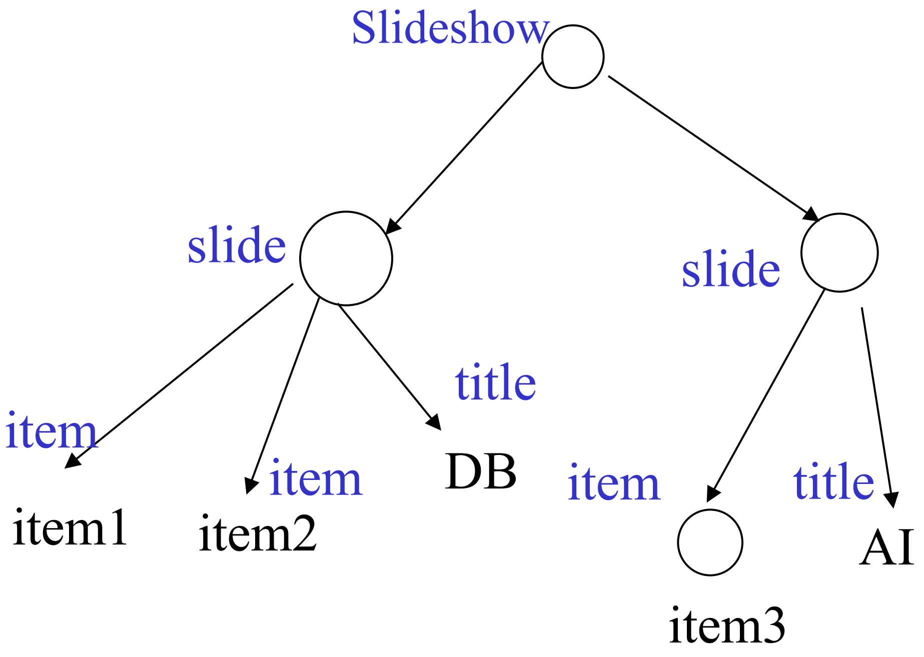
```
<?xml version="1.0" encoding="us-ascii" ?>
<!-- A SAMPLE set of slides   -->
<slideshow title="Sample Slide Show">
   <!-- TITLE SLIDE   -->
   <slide type="all">
       <title>Introduction to CML</title>
   </slide>
   <!-- OVERVIEW   -->
   <slide type="all">
    <title>Overview</title>
    <item>Why is XML great?</item>
    <item />
  </slide>
</slideshow>
```

Next: Data Type Definition (DTD)

Data Type Definition (DTD)

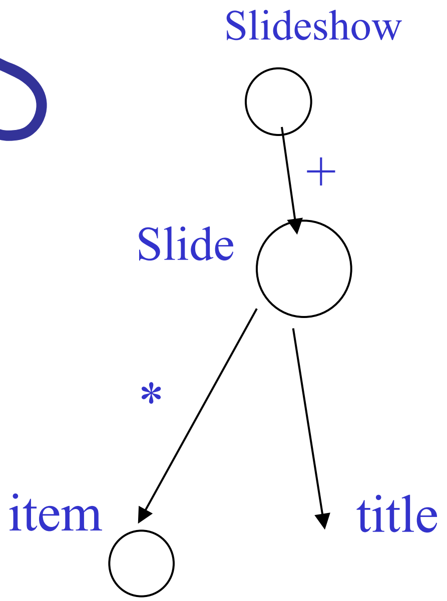
- DTD specifies the types of tags that can be included in the XML document
  - it defines which tags are valid, and in what arrangements
  - where text is expected, letting the parser determine whether the whitespace it sees is significant or ignorable
- An optional part of the document prolog

XML document and DTD



XML Document

XML DTD



```
<?xml version='1.0' encoding='us-ascii'?>
<!-- DTD for a simple "slide show".-->
<!ELEMENT slideshow (slide+)>
<!ELEMENT slide (title, item*)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT item (#PCDATA) >
```

Qualifiers

Qualifier	Meaning
?	Optional (zero or one)
*	Zero or more
+	One or more

Defining Text

- PCDATA:
  - Parsed Character DATA (PCDATA)
  - "#" that precedes PCDATA indicates that what follows is a special word, rather than an element name
- CDATA:
  - Unparsed character data
  - Normally used for embedding scripts (such as Javascript scripts).

Attribute Types

Attribute Type	Specifies...
CDATA	"Unparsed character data" = a text string.)
ID	A name that no other ID attribute shares.
IDREF	A reference to an ID defined elsewhere in the document.
IDREFS	A space-separated list containing one or more ID references.
ENTITY	The name of an entity defined in the DTD.
ENTITIES	A space-separated list of entities.
NMTOKEN	A valid XML name composed of letters, numbers, hyphens, underscores, and colons.
NMTOKENS	A space-separated list of names.
NOTATION	The name of a DTD-specified notation, which describes a non-XML data format, such as those used for image files

Next: Parsing XML (SAX/DOM)

What is an XML Parsing API?

- Programming model for accessing an XML document
- Sits on top of an XML parsing engine
- Language/platform independent

Java XML Parsing Specification

- The Java XML Parsing Specification is a request to include a standardised way of parsing XML into the Java standard library
- The specification defines the following packages:
  - javax.xml.parsers
  - org.xml.sax
  - org.xml.sax.helpers
  - org.w3c.dom
- The first is an all-new plugability layer, the others come from existing packages

Two ways of using XML parsers: SAX and DOM

- The Java XML Parsing Specification specifies two interfaces for XML parsers:
  - Simple API for XML (SAX) is a flat, event-driven parser
  - Document Object Model (DOM) is an object-oriented parser which translates the XML document into a Java Object hierarchy

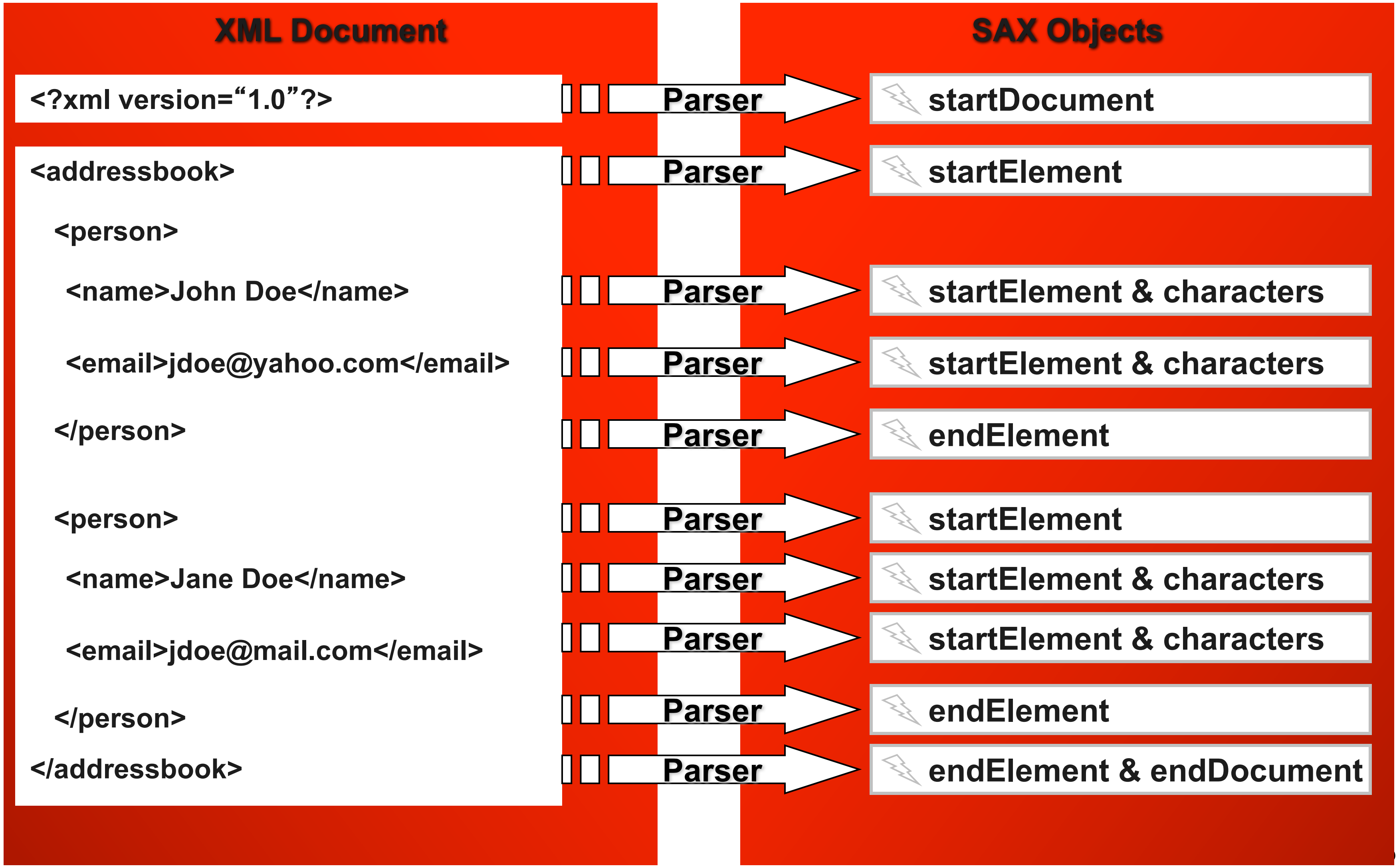
SAX

- Simple API for XML
- Event-based XML parsing API
- Not governed by any standards body
  - Guy named David Megginson basically owns it…
- SAX is simply a programming model that the developers of individual XML parsers implement
  - SAX parser written in Java would expose the equivalent events
  - "serial access" protocol for XML

SAX (cont)

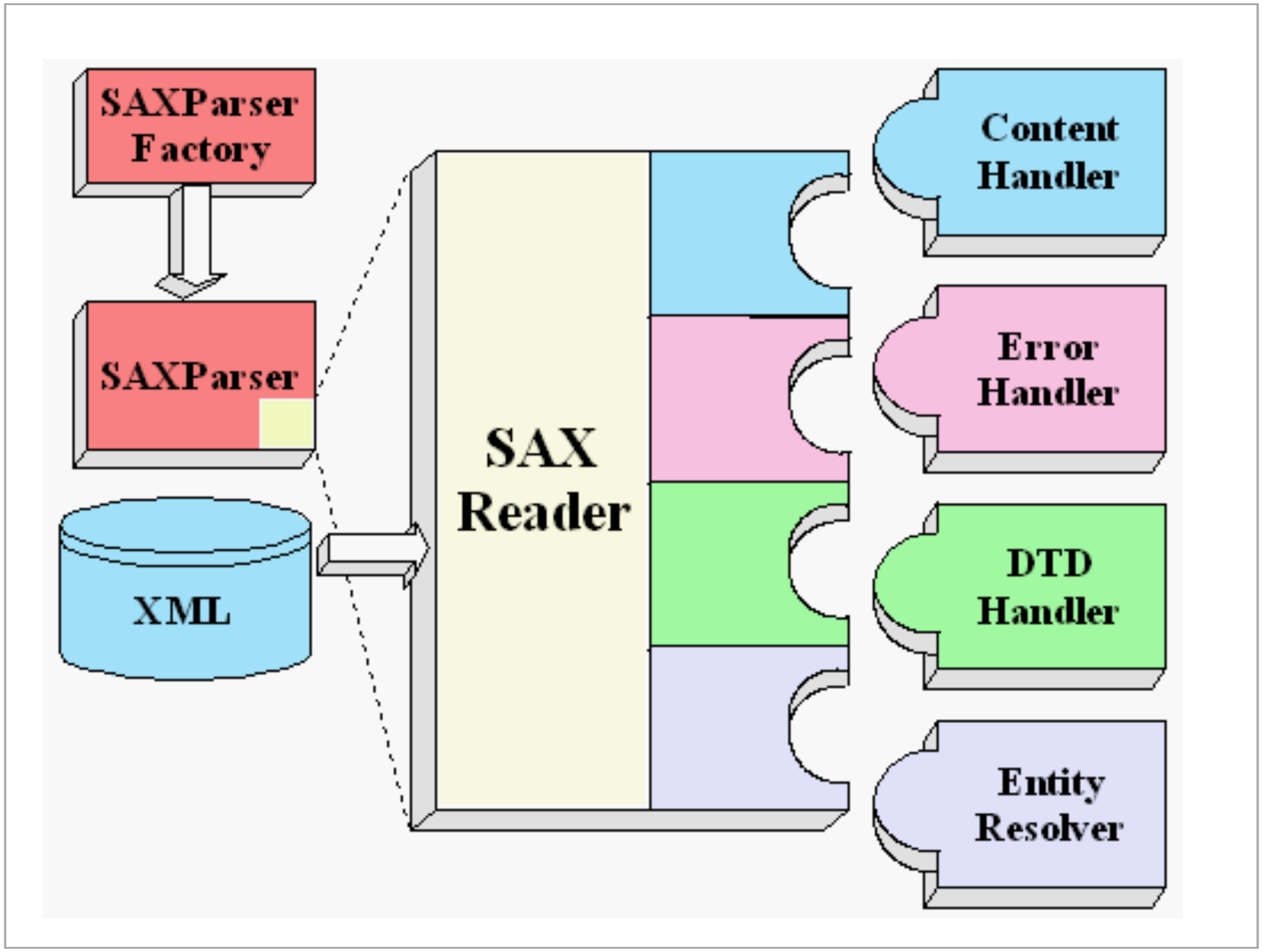
- A SAX parser reads the XML document as a stream of XML tags:
  - starting elements, ending elements, text sections, etc.
- Every time the parser encounters an XML tag it calls a method in its HandlerBase object to deal with the tag.
- The HandlerBase object is usually written by the application programmer.
- The HandlerBase object is given as a parameter to the parse() method in the SAX parser. It includes all the code that defines what the XML tags actually "do".

How Does SAX work?



XML Document		SAX Objects
`<?xml version="1.0"?>`	Parser	startDocument
`<addressbook>`	Parser	startElement
`<person>`		
`<name>John Doe</name>`	Parser	startElement & characters
`<email>jdoe@yahoo.com</email>`	Parser	startElement & characters
`</person>`	Parser	endElement
`<person>`	Parser	startElement
`<name>Jane Doe</name>`	Parser	startElement & characters
`<email>jdoe@mail.com</email>`	Parser	startElement & characters
`</person>`	Parser	endElement
`</addressbook>`	Parser	endElement & endDocument

SAX structure



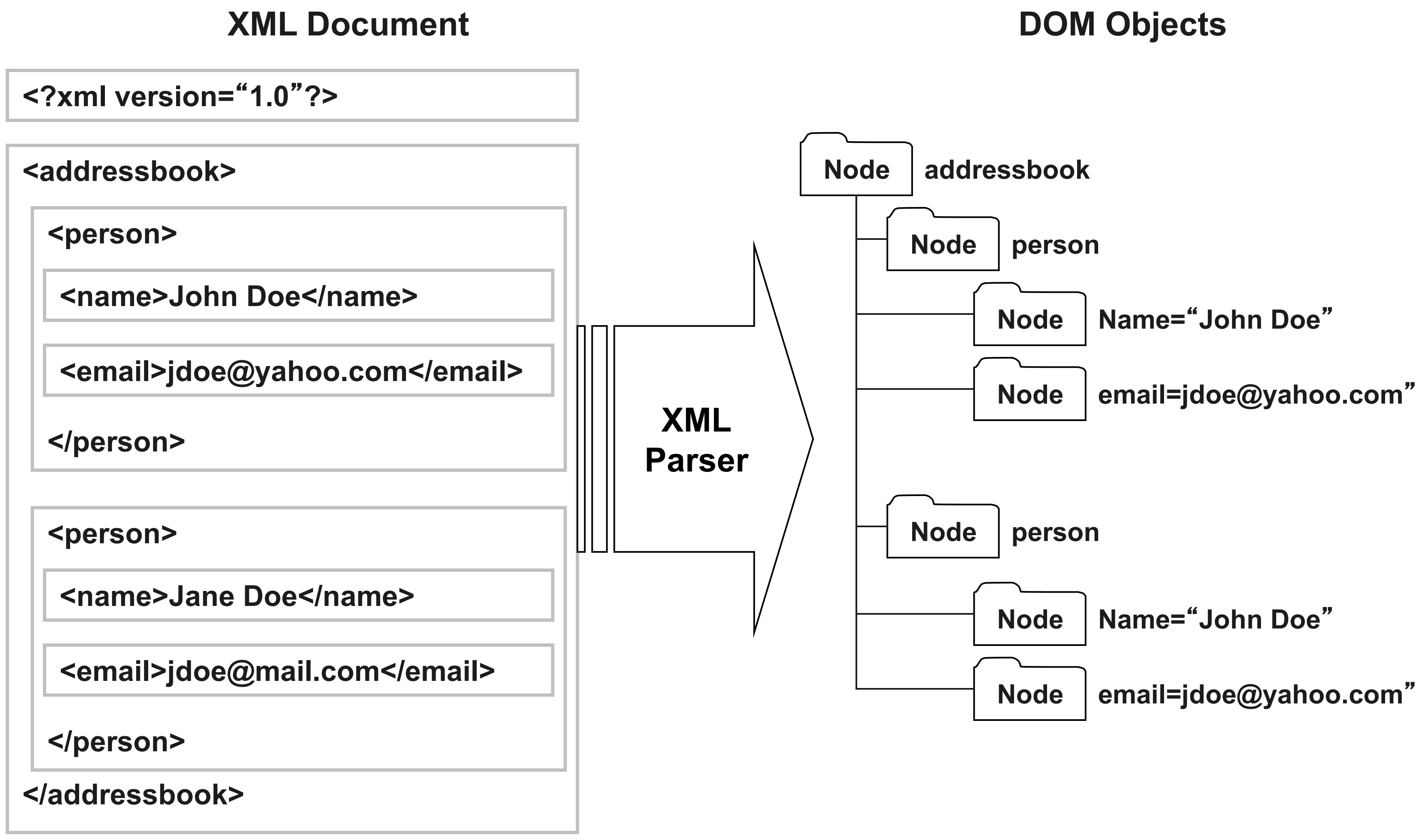
Document Object Model (DOM)

- Most common XML parser API
- Tree-based API
- W3C Standard
- All DOM compliant parsers use the same object model

DOM (cont)

- A DOM parser is usually referred to as a document builder. It is not really a parser, more like a translator that uses a parser.
- In fact, most DOM implementations include a SAX parser within the document builder.
- A document builder reads in the XML document and outputs a hierarchy of Node objects, which corresponds to the structure of the XML document.

How Does DOM work?



```
<?xml version="1.0"?>
<addressbook>
  <person>
    <name>John Doe</name>
    <email>jdoe@yahoo.com</email>
  </person>
  <person>
    <name>Jane Doe</name>
    <email>jdoe@mail.com</email>
  </person>
</addressbook>
```

DOM Structure Model and API

- hierarchy of Node objects:
  - document, element, attribute, text, comment, ...
- language independent programming DOM API:
  - get... first/last child, prev/next sibling, childNodes
  - insertBefore, replace
  - getElementsByTagName
  - ...
- Alternative event-based SAX API (Simple API for XML)
  - does not build a parse tree (reports events when encountering begin/end tags)
  - for (partially) parsing very large documents

A few functions

- Create a DOM document (tree)

```
Document doc = builder.parse( new File(argv[0]) );
```

- Remove those text nodes from those XML formatting spaces

```
doc.normalize();
```

- Generate a list of nodes for those “link” elements

```
NodeList links = doc.getElementsByTagName("link");
```

- W3C treats the text as a node

```
…getFirstChild().getNodeValue()
```