

Introduction to XML

*University of California, Santa Cruz Extension
Computer and Information Technology*



Introduction

- Answer the question “What is XML?”
- Cover fundamentals of XML.
- Practical demonstration using XML.
- Next steps – where to learn more about XML.



Presenter Background

- Software Architect – CPU Technology, Inc.
 - Responsible for design and development of digital circuit simulation tool suite.
- Founder – iDevelopSoftware, Inc.
 - Contract software development and training.
- Instructor – UC Santa Cruz Extension
 - Develop and teach Windows, .Net, and XML programming classes.
- Background in data warehousing, high-volume transaction processing systems, knowledge management, simulations, and embedded systems development.
- Worked at numerous startups, a few big companies, and as an independent consultant.



Lesson 1: Objectives

- Clearly articulate what XML is and what it is not.
- Identify potential uses for XML technologies.
- Select appropriate tools to manipulate XML documents.
- Understand relationship between XML standards.



What is XML?

- XML stands for eXtensible Markup Language.
- XML is a **text-based markup language**.
- XML is used to describe **semi-structured data**.
- XML is **self describing**.
- XML uses a **DTD** or **Schema** to describe the data.



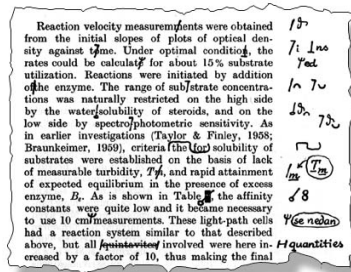
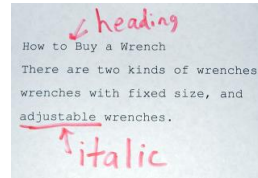
What XML is Not

- XML is not a programming language.
- XML is not a network transport protocol.
- XML is not a database.
- XML is not a silver bullet!



What is a Markup Language?

- A set of symbols and rules for their use when doing a markup of a document.
- A way to add computer-understandable information to text files. Certain parts of the text file are interpreted as markup instead of content. This markup may contain instructions for the computer. The interpretation of those instructions is defined by the semantics of a particular markup language. HTML, MathML, and SMIL are examples of markup languages.



What can I do with XML?

- XML can keep data separated from your HTML.
- XML can be used to store data inside HTML documents.
- XML can be used as a format to exchange information.
- XML can be used to store data in files or in databases.



How XML differs from HTML

- HTML is a markup language focused on presentation of information.
- XML is a markup language focused on representing information in a meaningful, reusable way.



HTML Stock Portfolio

```
<html>
<head>
  <meta http-equiv="Content-Type"
        content="text/html; charset=windows-1252"/>
  <title>Stock Portfolio</title>
</head>
<body>
  <h1>Stock Portfolio for: Bill Gates</h1>
  <h2>Close of Trade Day Summary: 04/17/2001</h2>
  <table border="1">
    <tr><th>Symbol</th><th>Corporate Name</th><th>Price at Close</th></tr>
    <tr><td>INTC</td><td>INTEL CORP</td><td>29.22</td></tr>
    <tr><td>ORCL</td><td>ORACLE CORP</td><td>17.35</td></tr>
    <tr><td>CSCO</td><td>CISCO SYSTEMS</td><td>17.9</td></tr>
  </table>
</body>
</html>
```



XML Stock Portfolio

```
<?xml version="1.0" encoding="UTF-8"?>
<portfolio>
  <investor>Bill Gates</investor>
  <trade_day>20010417</trade_day>
  <securities>
    <stock>
      <symbol>INTC</symbol>
      <name>INTEL CORP</name>
      <close_price>29.22</close_price>
    </stock>
    <stock>
      <symbol>ORCL</symbol>
      <name>ORACLE CORP</name>
      <close_price>17.35</close_price>
    </stock>
    <stock>
      <symbol>CSCO</symbol>
      <name>CISCO SYSTEMS</name>
      <close_price>17.9</close_price>
    </stock>
  </securities>
</portfolio>
```



Goals of XML

1. XML shall be straightforwardly usable over the Internet.
2. XML shall support a wide variety of applications.
3. XML shall be compatible with SGML.
4. It shall be easy to write programs that process XML documents.
5. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.



Goals of XML (cont.)

6. XML documents should be human-legible and reasonably clear.
7. The XML design should be prepared quickly.
8. The design of XML shall be formal and concise.
9. XML documents shall be easy to create.
10. Terseness in XML markup is of minimal importance.



Key Standards

- Extensible Markup Language (XML) 1.0 Spec.
- Namespaces in XML
- XML Inclusions (XInclude)
- XML Path Language (XPath) Version 1.0
- XML Schema
- XML Transformations (XSLT) Version 1.0

There are more – see <http://www.w3.org>.

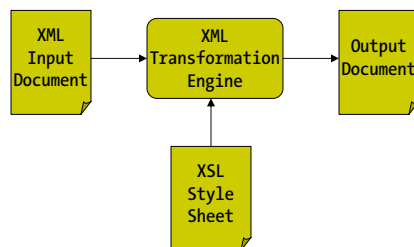


Terminology

- Parser
 - A piece of code that can parse text files according to the XML 1.0 Standard.
- Processor
 - A program designed to manipulate XML content in some predetermine way. Processors will use parsers to read/write XML documents.
- Vocabulary
 - A specific set of markup (tags, elements, attributes) used to define a set of related data. A vocabulary is sometimes referred to as an XML Application. Vocabularies are defined using a DTD or Schema.
- Transformation
 - Applying a set of rules to convert one XML document into another. Transformations are expressed using the XSLT standard.



XML-XSL Transformation Process



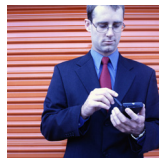
- Apply transformation to convert input content into some other format.
- Input document is XML.
- Output document can be
 - XML, HTML, CSV, TXT
- Style Sheet describes transformation process.
- Transformation engine is built into many popular XML parsers.



B2C Web Applications using XML



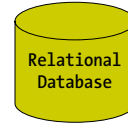
Browser Access
(HTML + CSS)



Wireless Access
(WML)



Web Server Farm

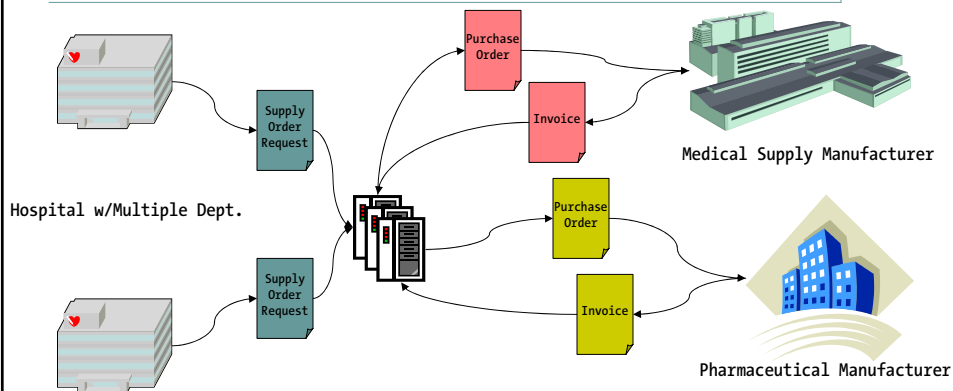


Relational Database

- Customer, product, and order data stored in relational database.
- Web application makes content available in appropriate format (HTML, WML) for each customer.
- Consistent look and feel for web application maintained using XSL style sheets.



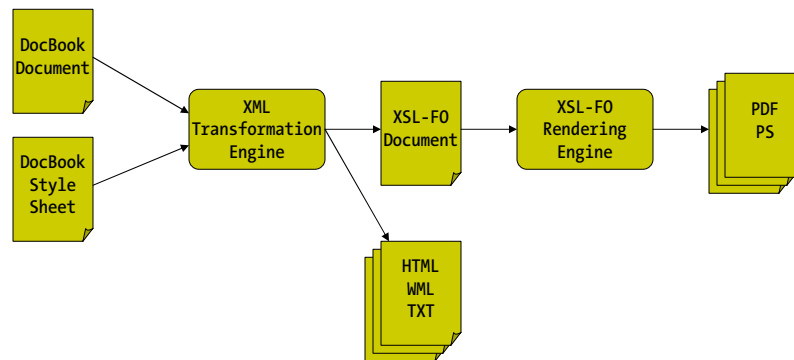
B2B Data Exchange using XML



- Departments submit order requests in an internal XML format.
- IT department collects requests into groups, transforms into purchase order XML format and delivers POs to manufacturer via network (SOAP, Web Services).
- Manufacturers fill orders and return invoices in XML format.
- IT department transforms invoices back into internal format for storage and payment.



Structured Documents using XML



<http://www.docbook.org>

XML Parsers

- Libxml (C/C++; Windows, Unix, Linux)
- MSXML (COM; Windows)
- System.Xml (Microsoft .Net Framework)
- SAXON (Java; Windows, Unix, Linux)
- Xerces (Java, C++, Perl; Windows, Unix, Linux)
- eXpat (C; Windows, Unix, Linux)

Other parsers exist for Perl, Python, PHP, etc.

XML Editors

- XmlSpy
- Serna
- XMLmind
- oXygen
- XMLWriter
- Xeena
- Notepad
- Vi
- Emacs



Lesson 1: Wrap Up

- XML is an “extensible” markup language used to define vocabularies for specific problem domains.
- XML can be applied to many areas in application development.
 - B2C, B2B, Technical Documentation, etc.
- Parsers and tools to implement XML solutions are readily available on all popular platforms.

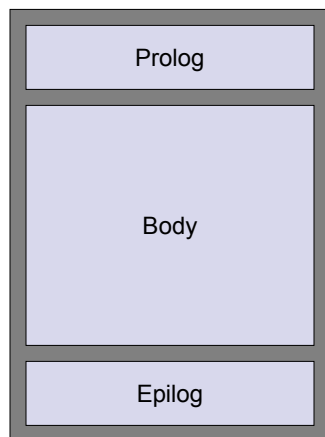


Lesson 2: Objectives

- Learn about the components of an XML document.
- Learn what an XML application is and how to define one.
- Learn how to create an XML document.



General XML Document Structure



- Prolog
 - Processing Instructions
 - Data Type Definitions
 - Comments
- Body
 - Root element
 - Nested Elements
 - Comments
- Epilog
 - Processing Instructions
 - Comments



Formatting Conventions

- Case Matters – these are not the same:
 - purchaseOrder, purchaseorder, PURCHASEORDER, Purchaseorder
- Naming Things
 - NameChar ::= Letter | Digit | ‘.’ | ‘-’ | ‘_’ | ‘:’
 - Name ::= (Letter | ‘_’ | ‘:’) (NameChar)*
 - Names must **not** begin with “XML” in any combination of upper and lower case letters.
- ISO/IEC 10646 Character Set provides International Support
- End-of-Line Handling
 - All lines must end with a single new-line.
 - Carriage returns are optional and will be removed by the parser.



Elements

- XML documents contain one or more elements.
- Elements have both a name and a value.
- The element name must follow the naming conventions mentioned earlier.
- The value of an element is referred to as content.
- Elements may contain nested elements.



Elements (cont.)

- Elements are delimited by start and end tags.
- Start and end tags are enclosed in '<' and '>' signs.
- Empty elements may omit the end tag by using an empty-element tag.
- Elements with content **must have** end tags. (This is different than HTML!)



Element Examples

- General case:
 - `<message>Hello, World!</message>`
 - `<message>
Hello, World!
</message>`
- Empty-element:
 - `<message></message>`
 - `<message/>`



Element Content

- Content is the text between start and end tags.
- Characters not allowed to appear in content:
 - Less-than character – ‘<’
 - Ampersand character – ‘&’
- Spaces between words are left intact.
- Additional white space characters (space, tab, carriage return, line feed) are ignored.
- Use entity references or CDATA sections to get around restrictions.



The Root Element

- Every document contains exactly one root element.
- Represents the data of the document.
- The root element has no sibling elements.
- All other elements must be child elements of the root element.



Child Elements

- Permitted within content area of any element.
- Nested elements **must not overlap**.



```
<root>
  <a>
    <b>
    </b>
  </a>
  <a>
    <b>
    </b>
  </a>
</root>
```



```
<root>
  <a>
    <b>
  </a>
  </b>
  <a>
    <b>
    </b>
  </a>
</root>
```

} Incorrect nesting



Invalid Content – Missing End Tags

Incorrect

```
<messages>
  <message>Hello World
  <message>Goodbye
</messages>
```

These message elements
must have closing tags.

Corrected

```
<messages>
  <message>Hello World</message>
  <message>Goodbye</message>
</messages>
```



Invalid Content – Special Characters

```
<sample lang="c++">
```

```
<description>
```

```
Example of using a  
conditional expression.
```

```
</description>  
<code>
```

```
if (x < 10)
```

```
{
```

```
  x = 5;
```

```
}
```

```
else
```

```
{
```

```
  x = x & y;
```

```
}
```

```
</code>
```

```
</sample>
```

The '<' character is not permitted in the content of an element.

The '&' character is not permitted in the content of an element.



Corrected Content – Entity Ref.

```
<sample lang="c++">
```

```
<description>
```

```
Example of using a  
conditional expression.
```

```
</description>  
<code>
```

```
if (x &lt; 10)
```

```
{
```

```
  x = 5;
```

```
}
```

```
else
```

```
{
```

```
  x = x &amp; y;
```

```
}
```

```
</code>
```

```
</sample>
```

The '<' character has been replaced with the < entity reference.

The '&' character has been replaced with the & entity reference.



Corrected Content – CDATA Section

```
<sample lang="c++">
```

```
<description>
```

```
Example of using a  
conditional expression.
```

```
</description>
```

```
<code><![CDATA[
```

```
if (x < 10)
```

```
{
```

```
  x = 5;
```

```
}
```

```
else
```

```
{
```

```
  x = x & y;
```

```
} ]]>
```

```
</code>
```

```
</sample>
```

Begin CDATA section using
“<![CDATA[” syntax.

End CDATA section using
“]]>” syntax.



Attributes

- An attribute is a name-value pair attached to an element's start tag.
- Names are separated from values by an equal sign.
- Values are enclosed in single or double quotes.
- Attribute order is not significant.
- Attributes are only permitted to be listed in an element tag once.



Attribute or Element – When to Use?

- Attributes and Elements both contain text, so really you may use either.
- Think of elements as “nouns” and attributes as “adjectives”.



Attribute Example

```
<messages>
  <message language="English">
    Goodbye
  </message>
  <message language="French">
    Au revoir
  </message>
  <message language="Italian">
    Arrivederci
  </message>
</messages>
```



Comments

- Comments are just like HTML.
- Syntax:
 - `<!-- Comment text goes here. -->`
- XML documents **must not** begin with a comment.
- Comments **must not** appear within the tag portion of an element.
- A double dash “--” **must not** appear within a comment.
- Take care when commenting out XML elements that you do not introduce a nesting error.



Processing Instructions

- Mechanism to pass information to an “XML Processor” that may read the document.
- Processors identified by target name.
- Syntax:
 - `<?target text?>`
- Target is a single word identifying the xml processor.
- Text is any text that should be made available to the processor.
- Examples:
 - `<?xml version="1.0" encoding="UTF8" standalone="yes"?>`
 - `<?xml-stylesheet href="common.css" type="text/css"?>`



Schema

- A set of rules describing the allowable content of a particular vocabulary or XML application.
- Schema languages include the following:
 - DTD – Part of core XML standard, very rudimentary.
 - W3C Schema – Powerful, difficult to master.
 - Others: RELAX NG, Schematron, Hook



Well-Formed and Valid XML Docs.

- Well-Formed
 - An XML document that follows the syntax rules described in the XML standard is said to be well-formed.
 - All XML documents **must** be well-formed.
 - All XML parsers perform checks to ensure documents are well-formed.
- Valid
 - An XML document that contains elements and attributes from a particular XML application is considered valid if and only if it follows the rules set forth by the XML application schema.
 - Validating Parsers (a special class of parser) are able to check for document validity.



Sample XML Vocabularies

- RSS – a format for syndicating news and the content of news-like sites, including major news sites like Wired, news-oriented community sites like Slashdot, and personal weblog.
- XBEL – an Internet "bookmarks" interchange format.
- OPML – an XML-based format that allows exchange of outline-structured information between applications running on different operating systems and environments.
- Check <http://www.xml.org> for a large collection of industry sponsored DTD and Schema definitions.



Lesson 2: Wrap Up

- XML files are made up of elements and attributes.
- DTD and Schema can be used to define an XML vocabulary.
- XML documents must be well-formed.
- XML documents may be valid according to some schema.



Advanced Example: XSL Transformations

- XML file contains course description.
- XSL transformation used to render as HTML
 - Course Overviews
 - Catalog Entries



Resources

- Slides and examples available via e-mail
 - bennettsmith@idevelopsoftware.com
- <http://www.w3.org>
 - Read the standards!
- <http://www.w3school.org>
 - Quick tutorials on web technologies.
- UC Extension XML Courses!



Thank You and Good Night!

Request copies of slides or examples via e-mail.

bennettsmith@idevelopsoftware.com

