DT228/2 Web Development

Introduction to XML

Definition

XML is a cross-platform, software and hardware independent tool for transmitting information.

“XMl is going to be everywhere” – W3C

Introduction

- XML is an important technology used throughout web applications
- XML stands for **EX**tensible **M**arkup **L**anguage
- XML is a **markup language** much like HTML
- XML was designed to **describe data**
- XML tags are not predefined in XML. You must **define your own tags**

Introduction

- XML uses a **Document Type Definition** (DTD) or an **XML Schema** to describe the data
- XMl documents are human readable
- XMl documents end with .xml   e.g. note.xml

Introduction

The Web was created to publish information for people

- – "eyes-only" was dominant design perspective
- – Hard to search
- – Hard to automate processing

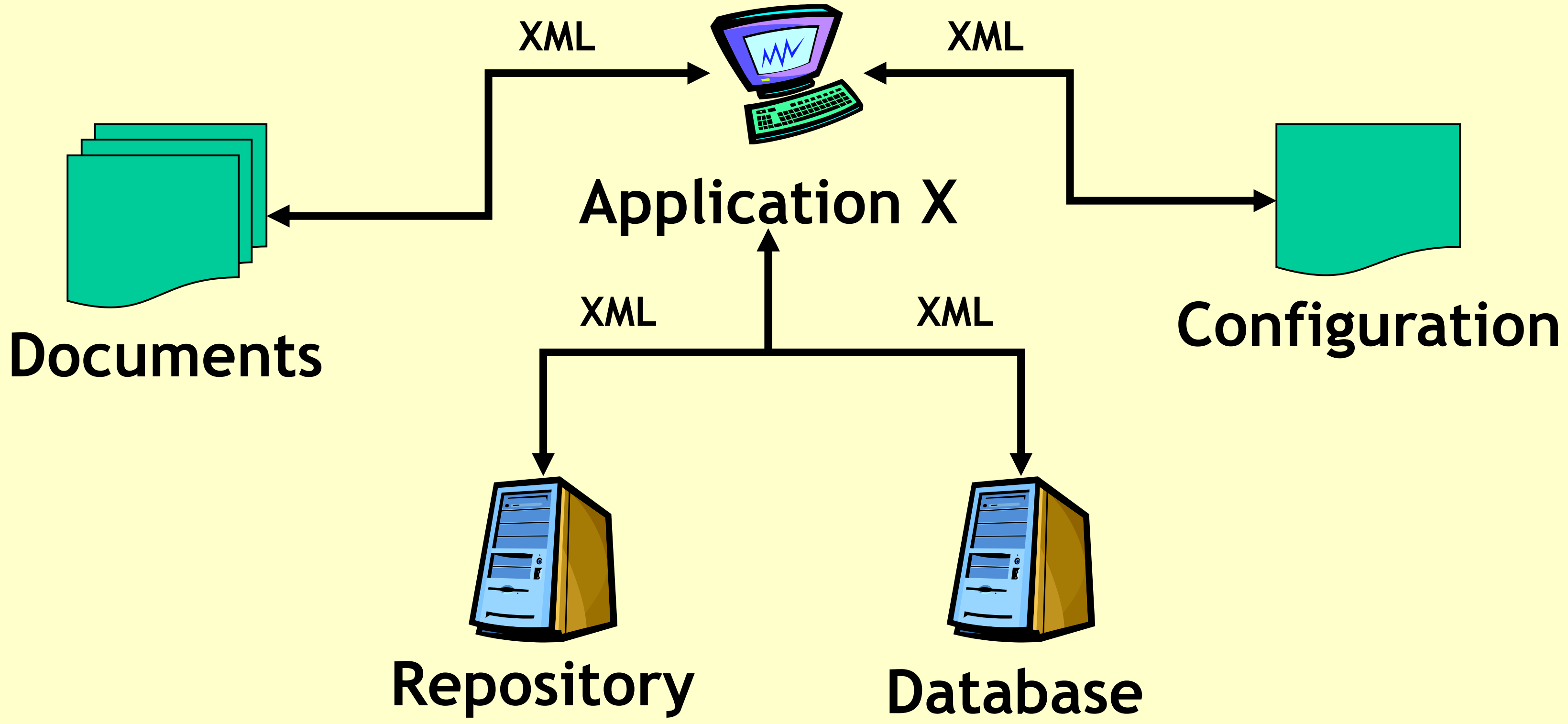
Introduction

The Web is using XML to become a platform for information exchange between <u>computers </u>(and people)

- – Overcomes HTML’s inherent limitations
- – Enables the new business models of the network economy

What is XML?

- XML is a "use everywhere" data specification



eXtensible Markup Language

- Instead of a fixed set of format-oriented tags like HTML, XML allows you to create whatever set of tags are needed for your type of information.

- This makes any XML instance “self-describing” and easily understood by computers and people.

- XML-encoded information is smart enough to support new classes of Web and e-commerce applications.

Why XML?

- XML plays an important role in many different IT systems.
- XML is often used for distributing data over the Internet.
- It is important (for all types of software developers!) to have a good understanding of XML

Why XML?

- **Sample Catalog Entry in HTML**

```
<TITLE> Laptop Computer </TITLE>
   <BODY>
      <UL>
            <LI> 15-inch MacBook Pro
            <LI>2.7 GHz
            <LI> 4 Gb
            <LI>512 Gb
            <LI> 2.54 Kg
            <LI> €2349
    </UL>
 </BODY>
```

- How can I parse the content? E.g. price?
- Need a more flexible mechanism than HTML to interpret content.

Why XML?

Sample Catalog Entry using XML

```
<COMPUTER TYPE=“Laptop”>
    <MANUFACTURER>Apple</MANUFACTURER>
    <LINE> MacBook Pro </LINE>
    <MODEL> 15-inch </MODEL>
    <SPECIFICATIONS>
        <SPEED UNIT = “GHz”>27</SPEED>
        <MEMORY UNIT=“GB”>4</MEMORY>
        <DISK UNIT=“GB”>512</DISK>
        <WEIGHT UNIT=“Kg”>2.54</WEIGHT>
        <PRICE CURRENCY=“Euro”>2349</PRICE>
    </SPECIFICATIONS>
</COMPUTER>
```

Smart processing using XML

- <COMPUTER> and <SPECIFICATIONS> provide logical containers for extracting and manipulating product information as a unit
- – Sort by <MANUFACTURER>, <SPEED>,

<WEIGHT>, <PRICE>, etc.

- Explicit identification of each part enables

its automated processing

– Convert <PRICE> from “Euro” to USD, Yen,etc.

Document exchange

- Use of XML allows companies to exchange information that can be processed automatically without human intervention e.g.
  - Purchase orders
  - Invoices
  - Catalogues etc

Difference between HTML and XML?

- **XML was designed to carry data.**

- XML is not a replacement for HTML.

- XML and HTML were designed with different goals:
  - XML was designed to describe data and to focus on what data is.
    HTML was designed to display data and to focus on how data looks.
  - HTML is about displaying information, while XML is about describing information.

XML is free and extensible

- XML tags are not predefined. You must "invent" your own tags.

- The tags used to mark up HTML documents and the structure of HTML documents are predefined. The author of HTML documents can only use tags that are defined in the HTML standard (like <p>, <h1>, etc.).

- XML allows the author to define his own tags and his own document structure.

- The tags in the example above (like <to> and <from>) are not defined in any XML standard. These tags are "invented" by the author of the XML document.

Documents vs. Data

- XML is used to represent two main types of things:
  - Documents
    - Lots of text with tags to identify and annotate portions
      of the document
  - Data
    - Hierarchical data structures

XML and Structured Data

- Pre-XML representation of data:

```
"PO-1234","CUST001","X9876","5","14.98"
```

- XML representation of the same data:

```
<PURCHASE_ORDER>
  <PO_NUM> PO-1234 </PO_NUM>
  <CUST_ID> CUST001 </CUST_ID>
  <ITEM_NUM> X9876 </ITEM_NUM>
  <QUANTITY> 5 </QUANTITY>
  <PRICE> 14.98 </PRICE>
</PURCHASE_ORDER>
```

Benefits of XML

- Open W3C standard
- Representation of data across heterogeneous environments
  - Cross platform
  - Allows for high degree of interoperability
- Strict rules
  - Syntax
  - Structure
  - Case sensitive

Who Uses XML?

- Submissions by
  - Microsoft
  - IBM
  - Hewlett-Packard
  - Fujitsu Laboratories
  - Sun Microsystems
  - Netscape (AOL), and others…
- Technologies using XML
  - SOAP, ebXML, BizTalk, WebSphere, many others…

Components of an XML Document

- Elements
  - Each element has a beginning and ending tag
    - `<TAG_NAME>...</TAG_NAME>`
  - Elements can be empty (`<TAG_NAME />`)
- Attributes
  - Describes an element; e.g. data type, data range, etc.
  - Can only appear on beginning tag
- Processing instructions
  - Encoding specification (Unicode by default)
  - Namespace declaration
  - Schema declaration

Components of an XML Document

```
<?xml version=“1.0” ?>
<?xml-stylesheet type="text/xsl” href=“template.xsl"?>
<ROOT>
   <ELEMENT1><SUBELEMENT1 /><SUBELEMENT2 /></ELEMENT1>
   <ELEMENT2> </ELEMENT2>
   <ELEMENT3 type=‘string’> </ELEMENT3>
   <ELEMENT4 type=‘integer’ value=‘9.3’> </ELEMENT4>
</ROOT>
```

Elements with Attributes

Elements

Prologue (processing instructions)

Rules For Well-Formed XML

- There must be one, and only one, root element
- Sub-elements must be properly nested
  - A tag must end within the tag in which it was started
- Attributes are optional
  - Defined by an optional schema
- Attribute values must be enclosed in “” or ‘ ’
- Processing instructions are optional
- XML is case-sensitive
  - `<tag>` and <TAG> are not the same type of element

Well-Formed XML?

- No, CHILD2 and CHILD3 do not nest properly

```
<xml? Version="1.0" ?>
<PARENT>
<CHILD1>This is element 1</CHILD1>
<CHILD2><CHILD3>Number
3</CHILD2></CHILD3>
</PARENT>
```

Well-Formed XML?

- No, there are two root elements

```
<xml? version="1.0" ?>
<PARENT>
<CHILD1>This is element 1</CHILD1>
</PARENT>
<PARENT>
<CHILD1>This is another element
1</CHILD1>
</PARENT>
```

Well-Formed XML?

- Yes

```
<xml? Version=“1.0” ?>
<PARENT>
<CHILD1>This is element 1</CHILD1>
<CHILD2/>
<CHILD3></CHILD3>
</PARENT>
```

An XML Document

```
<?xml version='1.0'?>
<bookstore>
<book genre='autobiography' publicationdate='1981'
ISBN='1-861003-11-0'>
<title>The Autobiography of Benjamin Franklin</title>
<author>
<first-name>Benjamin</first-name>
<last-name>Franklin</last-name>
</author>
<price>8.99</price>
</book>
<book genre='novel' publicationdate='1967' ISBN='0-201-63361-2'>
<title>The Confidence Man</title>
<author>
<first-name>Herman</first-name>
<last-name>Melville</last-name>
</author>
<price>11.99</price>
</book>
</bookstore>
```

Namespaces: Overview

- Part of XML's extensibility
- Allow authors to differentiate between tags of the same name (using a prefix)
  - Frees author to focus on the data and decide how to best describe it
  - Allows multiple XML documents from multiple authors to be merged
- Identified by a URI (Uniform Resource Identifier)
  - When a URI is used, it does NOT have to represent a live server

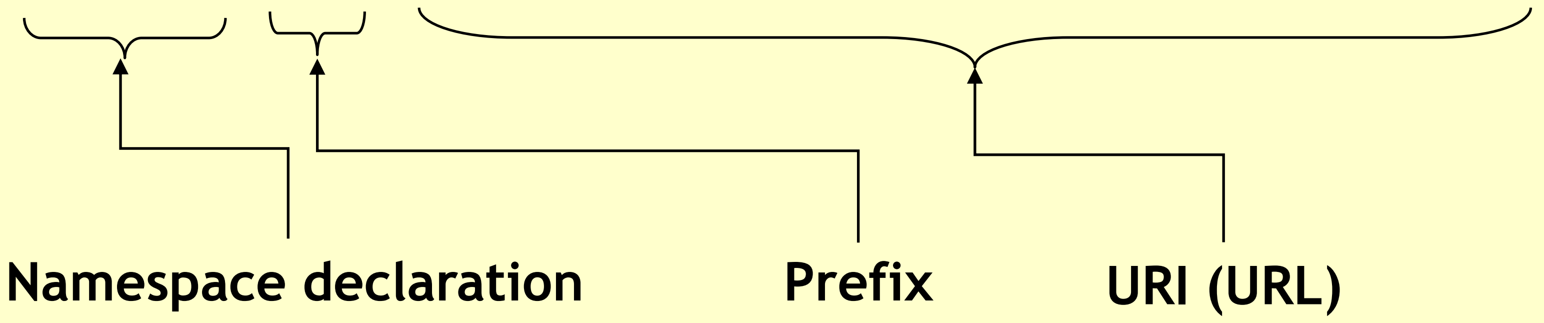
Namespaces: Declaration

Namespace declaration examples:

```
xmlns: bk = “http://www.example.com/bookinfo/”
```

```
xmlns: bk = “urn:mybookstuff.org:bookinfo”
```

```
xmlns: bk = “http://www.example.com/bookinfo/”
```



Namespaces: Examples

```
<BOOK xmlns:bk="http://www.bookstuff.org/bookinfo">
          <bk:TITLE>All About XML</bk:TITLE>
         <bk:AUTHOR>Joe Developer</bk:AUTHOR>
    <bk:PRICE currency='US Dollar'>19.99</bk:PRICE>
```

```
<bk:BOOK xmlns:bk="http://www.bookstuff.org/bookinfo"
             xmlns:money="urn:finance:money">
          <bk:TITLE>All About XML</bk:TITLE>
         <bk:AUTHOR>Joe Developer</bk:AUTHOR>
        <bk:PRICE money:currency='US Dollar'>
                    19.99</bk:PRICE>
```

Namespaces: Default Namespace

- An XML namespace declared without a prefix becomes the default namespace for all
  sub-elements
- All elements without a prefix will belong to the default namespace:

```
<BOOK xmlns=“http://www.bookstuff.org/bookinfo”>
           <TITLE>All About XML</TITLE>
          <AUTHOR>Joe Developer</AUTHOR>
```

Namespaces: Scope

- Unqualified elements belong to the inner-most default namespace.
  - BOOK, TITLE, and AUTHOR belong to the default book namespace
  - PUBLISHER and NAME belong to the default publisher namespace

```
<BOOK xmlns=“www.bookstuff.org/bookinfo”>
        <TITLE>All About XML</TITLE>
       <AUTHOR>Joe Developer</AUTHOR>
 <PUBLISHER xmlns=“urn:publishers:publinfo”>
         <NAME>Microsoft Press</NAME>
               </PUBLISHER>
                </BOOK>
```

Namespaces: Attributes

- Unqualified attributes do NOT belong to any namespace
  - Even if there is a default namespace
- This differs from elements, which belong to the default namespace

Entities

- Entities provide a mechanism for textual substitution, e.g.

Entity	Substitution
`&lt;`	<
`&amp;`	&

- You can define your own entities
- Parsed entities can contain text and markup
- Unparsed entities can contain any data
  - JPEG photos, GIF files, movies, etc.

To manually create or write an XML document:

- Figure out what the overall document *is* (-> root tag)
- Figure out what the key fields of information.. (-> tag names)
- Figure out which information is data (-> tag contents)

To manually create or write an XML document:

Example: Write an XML document for the following

Address book

Name	Telephone	Address
Jeremy Cannon	0098837	22, Marlboro Ct
Naomi Murphy	992887	39, Alma Road
Sheila Zheng	999287	Apt 4, Hyde Road

To manually create or write an XML document:

- Figure out what the overall document *is* (-> root tag)
 --- Address book

- Figure out what the key fields of information.. (-> tag names)
--- Information is Name (which can be broken into first name, surname), Telephone, Address (which can be broken down into house number, street name)

- Figure out which information is data (= tag contents)
number, street name)

To manually create or write an XML document: example

```
<addressbook>
   <person>
       <name>
              <firstname> Jeremy </first name>
              <surname> Cannon</surname>
        </name>
        <telephone> 0098837</telephone>
         <address>
              <housenumber>22</house number>
             <street> Marlboro Ct</street>
           </address>
     </person>
     <person>
         <name>
             <firstname> Jeremy </first name>
       … etc

     </person
</addressbook>
```

To check your document..

- Save it as .xml file

- Open it in a browser – and see if any errors are produced

  OR

- go to a specialist XML validator for better error diagnosis e.g. http://www.w3schools.com/dom/dom_validate.asp

Valid XML documents

- XML itself is fairly simple
- Most of the learning curve is knowing about
  all of the related technologies

Valid XML documents

- DTD (Document Type Definitions)
  - Not written in XML
  - No support for data types or namespaces
- XSD (XML Schema Definition)
  - Written in XML
  - Supports data types
  - Current standard recommended by W3C

Valid XML documents

- Define the “rules” (grammar) of the document
  - Data types
  - Value bounds
- A XML document that conforms to a schema is said to be valid
  - More restrictive than well-formed XML
- Define which elements are present and in what order
- Define the structural relationships of elements

Schemas: DTD Example

- XML document:

```
<BOOK>
<TITLE>All About XML</TITLE>
<AUTHOR>Joe Developer</AUTHOR>
</BOOK>
```

- DTD schema:

```
<!DOCTYPE BOOK    [
<!ELEMENT BOOK    (TITLE+, AUTHOR) >
<!ELEMENT TITLE   (#PCDATA) >
<!ELEMENT AUTHOR  (#PCDATA) >
]>
```

Document Type Definitions

The DTD (note.DTD) for XMl document Note: is

```
<!DOCTYPE note [

 <!ELEMENT note (to,from,heading,body)>
 <!ELEMENT to (#PCDATA)>
 <!ELEMENT from (#PCDATA)>
 <!ELEMENT heading (#PCDATA)>
 <!ELEMENT body (#PCDATA)> ]>
```

Lists the document type (note) and the valid elements, and the type of content they can accept

#PCDATA means parse-able text data.

XML Schema

XML Schema is an XML based alternative to DTD.

An XML schema describes the structure of an XML document.

XML Schemas will probably be used in most Web applications as a replacement for DTDs because:

- XML Schemas are supported by the W3C
- XML Schemas are richer and more useful than DTDs
- XML Schemas are written in XML
- XML Schemas support data types
- XML Schemas are extensible to future additions
- XML Schemas support namespaces

Schemas: XSD Example

- XML document:

```
<CATALOG>
   <BOOK>
      <TITLE>All About XML</TITLE>
      <AUTHOR>Joe Developer</AUTHOR>
   </BOOK>
   …
</CATALOG>
```

Schemas: XSD Example

```
<xsd:schema id="NewDataSet“ targetNamespace="http://tempuri.org/schema1.xsd"
   xmlns="http://tempuri.org/schema1.xsd"
   xmlns:xsd="http://www.w3.org/1999/XMLSchema"
   xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
  <xsd:element name="book">
    <xsd:complexType content="elementOnly">
      <xsd:all>
        <xsd:element name="title" minOccurs="0" type="xsd:string"/>
        <xsd:element name="author" minOccurs="0" type="xsd:string"/>
      </xsd:all>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name=“Catalog" msdata:IsDataSet="True">
    <xsd:complexType>
      <xsd:choice maxOccurs="unbounded">
        <xsd:element ref="book"/>
      </xsd:choice>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

XML Parser

```
<!DOCTYPE html>
<html>
<body>

<h1>DT228/2 Internal Note</h1>
<div>
<b>To:</b> <span id="to"></span><br>
<b>From:</b> <span id="from"></span><br>
<b>Message:</b> <span id="message"></span>
</div>
```

XML Parser

```
<script>
var txt, parser, xmlDoc;
txt = "<note>" +
"<to>Tove</to>" +
"<from>Jani</from>" +
"<heading>Reminder</heading>" +
"<body>Don't forget me this weekend!</body>" +
"</note>";
```

XML Parser

```
parser = new DOMParser();
xmlDoc = parser.parseFromString(txt,"text/xml");

document.getElementById("to").innerHTML =
xmlDoc.getElementsByTagName("to")[0].childNodes[0].nod
eValue;
document.getElementById("from").innerHTML =
xmlDoc.getElementsByTagName("from")[0].childNodes[0].
nodeValue;
document.getElementById("message").innerHTML =
xmlDoc.getElementsByTagName("body")[0].childNodes[0].
nodeValue;
</script>
</body>
</html>
```

XML-Based Applications

- Microsoft SQL Server
  - Retrieve relational data as XML
  - Query XML data
  - Join XML data with existing database tables
  - Update the database via XML Updategrams
  - New XML data type in SQL 2005
- Microsoft Exchange Server
  - XML is native representation of many types of data
  - Used to enhance performance of UI scenarios (for example, Outlook Web Access (OWA))