XML Concept

- 1. XML Concept
- 2. XML: Defining Well-formed documents
- 3. XSD : Defining Valid documents, Introduction to Schema

Instructor: Professor Izidor Gertner,
  e-mail: csicg@csfaculty.engr.ccny.cuny.edu

Teaching Assistant: MinHua Huang
  e-mail: minhuahuang2003@yahoo.com

What is XML

1. XML is a markup language much like HTML, but XML is not replacement for HTML (Hyper Text Markup Language).

2. XML was designed to describe data (what data is), but HTML design to display data (how data looks).

3. XML is created as a way to structure, store and send information. In other words, main goal of XML is carrying data, not displaying; Content and presentation are separate.

4. XML tags are not predefined in XML. Then, you must invent your own tags, unlike HTML tags, which is predefined. So, we call it "Extensible" Markup Language or Meta-language.

5. XML uses a XML Schema or DTD (Document Type Definition) to describe the data. You need the schema since you define your tags. Schemas are very much the same concept as grammar in language systems. In other words, the purpose of a Schema is to define the legal building blocks of an XML.

6. XML with a XML Schema is designed to be self-descriptive. It defines elements, attributes, values, and relations in a document. And notice that XML Schema is a alternative of DTD (Document Type definition) by W3C recommendation on May 2, 2001.

Extensible Markup Language (XML): http://www.w3.org/XML/

Examples of XML documents

Note: Internet Explore 5.0 above and Netscape 6.0 above support XML, but if you want to see the tree structure of XML documents, use Internet Explore. And notice + and - sign and click on it, then you will find the tree structure of XML documents.

XML documents should be Well-Formed or Valid Documents.

1. Invalid Documents: Documents that don't follow the tag rules. Also, if a document has a XML Schema or DTD, and it doesn't follow the rules defined in its DTD, that document is invalid.

2. Well-Formed Documents: Documents that follow the XML tag rules, but don't have a XML Schema or DTD.

3. Valid Documents: Documents that follow both the XML tag rules and the rules defined
Invalid Document

http://java.sun.com/xml/jaxp/dist/1.1/docs/tutorial/sax/samples/slideSampleBad1.xml

```xml
<?xml version='1.0' encoding='utf-8'?>
<!-- Slides with a fatal error -->
<slideshow
    title="Sample Slide Show"
    date="Date of publication"
    author="Yours Truly"
>
<!-- TITLE SLIDE -->
<slide type="all">
    <title>Wake up to WonderWidgets!</title>
</slide>

<!-- OVERVIEW -->
<slide type="all">
    <title>Overview</title>
    <item>Why <em>WonderWidgets</em> are great</item>
    <item>Who <em>buys</em> WonderWidgets</item>
</slide>
<slideshow>
```

Well-Formed Document

http://java.sun.com/xml/jaxp/dist/1.1/docs/tutorial/dom/samples/slideSample01.xml

```xml
<?xml version='1.0' encoding='utf-8'?>
```
<!-- A SAMPLE set of slides -->
<slideshow
title="Sample Slide Show"
date="Date of publication"
author="Yours Truly">
<!-- TITLE SLIDE -->
<slide type="all">
<title>Wake up to WonderWidgets!</title>
</slide>

<!-- OVERVIEW -->
<slide type="all">
<title>Overview</title>
<Item>Why <em>WonderWidgets</em> are great</Item>
<Item>Who <em>buys</em> WonderWidgets</Item>
</slide>
</slideshow>

Valid Document
http://java.sun.com/xml/jaxp/dist/1.1/docs/tutorial/dom/samples/slideSample10.xml
XML Concept
What is XML

<?xml version='1.0' encoding='utf-8'?>
<!-- Slides valid document -->
<!-- http://java.sun.com/xml/jaxp/dist/1.1/docs/tutorial/dom/samples/slideSample10.xml -->
<!-- A SAMPLE set of slides -->
<!DOCTYPE slideshow SYSTEM "slideshow3.dtd" [
  <!ENTITY product  "WonderWidget">  
  <!ENTITY products "WonderWidgets">  <!-- FOR WALLY / WALLIES -->  
  <!ENTITY copyright SYSTEM "copyright.xml">  
]>  

<!-- SUBSTITUTIONS WORK IN ATTRIBUTES, TOO -->
<slideshow  
  title="&product; Slide Show"  
  date="Date of publication"  
  author="Yours Truly"  
>

<!-- PROCESSING INSTRUCTION -->
<?my.presentation.Program QUERY="exec, tech, all"?>

<!-- TITLE SLIDE -->
<slide type="all">  
  <slide-title>Wake up to &products;!</slide-title>  
</slide>

<!-- TITLE SLIDE -->
<slide type="all">  
  <item>&copyright;</item>  
</slide>

<!-- OVERVIEW -->
<slide type="all">  
  <slide-title>Overview</slide-title>  
  <item>Why <em>&products;</em> are great</item>  
  <item>Who <em>buys</em> &products;</item>  
</slide>

<slide type="tech">  
  <slide-title>How it Works</slide-title>  

<item>First we fozzle the frobmorten</item>
<item>Then we framboze the staten</item>
<item>Finally, we frenzle the fuznaten</item>
<item><![CDATA[Diagram:

frobmorten <------------- fuznaten
    |                      ^ <3>
    | <1> = fozzle
    |<2> = framboze
staten------------------------+ <3> = frenzle
    <2>
]]></item>
</slide>
</slideshow>

How To Use It?

XML can be deployed on the client side or on the server side as well as a data for data storage. It is important to know that XML was designed to store, carry and exchange data. You will need it when you need to send self-describing data to another machine or application.

Examples of usage of XML documents

0. To use and store XML documents for a browser: HTML

1. To utilize data stored inside HTML pages Script languages: Java Script, Visual Basic Script.

2. To use XML document to describe transformation of another XML document into something your browser can handle. Style Sheet Languages: XSL (Extensible Style Sheet Language), CSS (Cascade Style Sheet Language).

3. To interpret data and build models: DOM (Document Object Model), SAX (Simple API for XML)

XML can Separate Data from HTML

Your data can be stored outside your HTML using XML. When HTML is used to display data, the data is stored inside your HTML. With XML, data can be stored in separate XML files. This way you can concentrate on using HTML for data layout and display, and be sure that changes in the underlying data will not require any changes to your HTML. XML data can also be stored inside HTML pages as "Data Islands". You can utilize the data using Script languages such as Java Script, and use HTML only for formatting and displaying the data. Moreover, you can use XSL based on the browser type to form data into HTML style. This allows you to create different views of an XML document.
XML Concept

Defining a well-formed XML document

XML is used to Exchange Data

The strongest point of XML is its ability to interchange data since data can be exchanged between incompatible systems using XML. In the real world, computer systems and databases contain data in incompatible formats, so developers are required a significant amount of time-consuming to communicate between such systems over the Internet. Converting the data to XML can greatly reduce this complexity and create data that can be read by many different types of applications. XML makes it easy to send structured data across the web.

XML and B2B

From the advantage of Data Exchange using XML, financial information can be exchanged over the Internet. XML is going to be the main language for exchanging financial information between businesses over the Internet. A lot of interesting B2B applications are under development. For example, when using XML, I can receive XML data from your system, and you can receive XML data from mine. They are not required to know each other's system organization. They only need to share XML Schema.

XML can be used to Share Data

With XML, plain text files can be used to share data. Since XML data is stored in plain text format, XML provides a software- and hardware-independent way of sharing data. This makes it much easier to create data that different applications can work with. It also makes it easier to expand or upgrade a system to new operating systems, servers, applications, and new browsers.

XML can be used to Store Data

With XML, plain text files can be used to store data. XML can also be used to store data in files or in databases. Applications can be written to store and retrieve information from the store, and generic applications can be used to display the data. They simply require to follow the rules defined in the XML Schema.

XML can make your Data more Useful

Your data is available to more users due to the platform independency. You can render your data available to more than only standard HTML browsers. Other clients can access your XML files as data sources, like they are accessing databases. Your data can be made available to all kinds of agents or machines, such as palms or mobile phone. In order words, XML data can be utilized by any form of agents through their intelligent interpretations of information.

Defining a well-formed XML document

Most HTML browsers will attempt to display anything, no matter how poorly formed the HTML markup maybe. However, XML processor are required to generate a fatal error when the documents does not follow the rules of XML (not well-formed). XML processor is a software module that provides applications to check whether the document confirms to the rules of XML and the structures of the document against the rules specified in Schema.

An XML document can be divided into two categories: markup and character data.
XML Concept
Defining a well-formed XML document

**Markup:** everything begins with `<` or `&` and ends with `>` or `;`

**Character data:** everything that is not markup.

```xml
<?xml version="1.0" standalone="yes"?>
<book>
<title>Distributed Systems</title>
<author><subchild>Tim Kindberg</subchild></author>
<publication></publication>
<year>2000</year>
<img src="goodbook.gif" width="100" height="200" /> 
</book>
```

The first line is the XML declaration. It shows the version of XML document. Also, standalone="yes" attribute means this document does not require a XML Schema. It will increase performance in case where structure and reusability are not important.

**Elements of an XML document**

- **Elements:** e.g. `<title>Distributed Systems</title>`
- **Element name begins with a letter,** case sensitive element needs to step in at this point.
  - `<book></book>, <title></title>, <author></author>, <year></year>` (starting and ending tags) are elements
  - book, title, author, subchild, publication, year are generic identifiers, Type
  - Distributed Systems, Tim Kindberg, 2000 are **Content (value)**
  - `<publication></publication>` an element has no content empty element

**Attributes**

Provide additional descriptive information about an element.

Attributes are specified by name, followed by `=` sign to the attribute “value” within quotation marks.

```xml
<img src="goodbook.gif" width="100" height="200" /> 
```

elements can have attributes "goodbook.gif", "100", "200" (quoted ones inside attribute)

**XML Syntax Rules and characters**

**Note:** XML document must be well-formed
XML Concept

Defining a well-formed XML document

---

**Every XML Document Must Have Root Element**

```xml
<book>

..............

</book>
```

---

**Every Element Must Have Closing Tag**

```xml
<book>
<title>Distributed Systems</title>

.....

.......
</book>
```

---

**Element Must Be Properly Nested**

```xml
<book>
<title>Distributed Systems</title>
<author>
<subchild>Tim Kindberg</subchild>
</author>
<p>ublication>
</publication>
<year>2000</year>
</book>
```

---

**Attributes Must Be Enclosed Within Quotation Marks**

```xml
<img src="goodbook.gif" width="100" height="200" />
```

---

**Checking Well-Formedness**

If the document is not Well-Formed the application e.g. IE will give an error

The XML page cannot be displayed

Cannot view XML input using XSL style sheet. Please correct the error and then click the Refresh button, or try again later.

End tag 'slide' does not match the start tag 'item'. Error processing resource 'file:///G:/MyData/CCNYCOURSES/Cs311Fall2005...

</slide>

-----^--

---

**Declaring XML Version**

```xml
<?xml version="1.0" standalone="yes"?>
```
XML Comments

<!-- Ship Order -->

Whitespace

XML processors ignore whitespace (extra spaces, line breaks, tabs, etc...)

Entities

Entities are used to substitute frequently occurring text strings (like MACRO).
XML includes five default Entities:
Ent amp &
Ent gt >
We need Entities because XML Parser can confuse “greater” with “opening tag delimiter” and other delimiters...
Custom Entities definition
<!ENTITY products "WonderWidgets">
Usage
<slide-title>Wake up to &products;!</slide-title>

Character Data (CDATA)

Is used when you want to include plain text in XML document- XML parser will not process it.

• Do not use Entities in CDATA
• Do not nest CDATA

Summary of the example above wellformedbook-xml.xml

• All documents must have open and closing tags.
(If case of HTML codes that have one tag to be compatible with XML, put space before a back slash) <img src.... /> or <br />
Defining Valid XML documents

XML Concepts define classes of XML documents and it is now W3C official recommendation.

It can be found www.w3.org/XML/Schema.html.
XML Schemas divide **elements** into two types: simple and complex.

**Simple types**: elements that only contain character data.

**Complex types**: elements that contain other elements or attributes as well as character data.

**Note**: Attributes always have simple types.

---

**An XML Schema**

- defines elements that can appear in a document.
- defines attributes that can appear in a document.
- defines which elements are child elements.
- defines the sequence in which the child elements can appear.
- defines the number of child elements.
- defines whether an element is empty or can include text.
- defines default values for attributes.

---

**shiporder-xml.xml XML example**

```xml
<?xml version="1.0"?>
<shipOrder>
  <shipTo>
    <name>Jane Meyers</name>
    <street>138th Street</street>
    <address>Convent Avenue</address>
    <country>US</country>
  </shipTo>
  <items>
    <item>
      <title>Distributed Systems</title>
      <isbn>0-201-61918-0</isbn>
      <quantity>1</quantity>
      <price>50.95</price>
    </item>
    <item>
      <title>The Nature of Mind</title>
      <isbn>0-534-94566-x</isbn>
      <quantity>1</quantity>
      <price>45.90</price>
    </item>
  </items>
</shipOrder>
```
The order above consists of a

**root element** `<shipOrder>`, with

**two child elements** `<shipTo>` and `<items>`.`

**Note:** The `<items>` element contains `<item>` elements.

**Note:** An `<item>` element contains `<title>`, `<quantity>`, and `<price>` elements.

**shiporder-xsd shiporder-xml's XML Schema example**

```xml
<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema">

<xsd:element name="shipOrder" type="order"/>

<xsd:complexType name="order">
    <xsd:element name="shipTo"    type="shipAddress"/>
    <xsd:element name="items"     type="bookItems"/>
</xsd:complexType>

<xsd:complexType name="shipAddress">
    <xsd:element name="name" type="xsd:string"/>
    <xsd:element name="street" type="xsd:string"/>
    <xsd:element name="address" type="xsd:string"/>
    <xsd:element name="country" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="bookItems">
    <xsd:element name="item" type="bookItem"/>
</xsd:complexType>

<xsd:complexType name="bookItem">
    <xsd:element name="title" type="xsd:string"/>
    <xsd:element name="isbn" type="xsd:string"/>
    <xsd:element name="quantity" type="xsd:positiveInteger"/>
    <xsd:element name="price" type="xsd:decimal"/>
</xsd:complexType>

</xsd:schema>
```

**RULE:** If you create tag names, you have to define them all from a root to leaves until you reach all known data types.
Elements with **simple types** have no attributes and do not contain any other elements. XSD (XML Schema Definition language) contains a set of built-in simple types, such as **string**, **boolean**, **double**, **float**, and so on. Additional simple types can be derived from built-in types, which is called "restriction". XML Schemas have great flexible occurrence constraints and allow you to specify a minimum or a maximum, a value and even more complicated constraints.

Elements that have **complex types** are defined using the **complexType element**. The elements and attributes contained within complex are defined using **element and attribute elements**, respectively. The XML Schema above defines the element `<shipOrder>` to be of the type order. Order is a complex type element consisting of the elements `<shipTo>` and `<items>`. The `<shipTo>` element is of the type shipAddress - a complex type element consisting of the elements `<name>`, `<street>`, `<address>`, and `<country>`. The `<items>` element is of the type bookItems - a complex type element consisting of `<item>` elements. The `<item>` element is of the type bookItem - a complex type element consisting of `<title>`, `<isbn>`, `<quantity>`, and `<price>` elements. The `<title>` element is a normal element of the type string. If you read the following examples once more, you'll soon understand it.

---

**catalog-xml and its XML Schema catalog-xsd**

**Catalog-Xml.xml**

```xml
<?xml version='1.0' encoding='utf-8'?>
<!-- Catalog.xml document -->
<catalog>
  
  <product partNum="ABC-AMA355" onSaleDate="12/12/2001">
    <productName>Panasonic 1000</productName>
    <quantity_in_stock>39</quantity_in_stock>
    <price>250.95</price>
    <description>for the beginning photographers</description>
  </product>

  <product partNum="ABC-MON300" onSaleDate="12/13/2001">
    <productName>Sony SyncMaster 2K</productName>
    <quantity_in_stock>80</quantity_in_stock>
    <price>250.95</price>
    <description>Flat Screen, huge viewing size</description>
  </product>

  <product partNum="ABC-SCN255" onSaleDate="12/14/2001">
    <productName>Hp Scanner 5000</productName>
    <quantity_in_stock>55</quantity_in_stock>
    <price>250.95</price>
    <description>High resolution vivid quality</description>
  </product>

</catalog>
```

**and the schema catalog-xsd.xsd**

```xml
<?xml version='1.0' encoding='utf-8'?>
<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema">
```
<xsd:element name="Catalog" type="CatalogType"/>

<xsd:complexType name="CatalogType">
  <xsd:element name="product" type="ProductType"/>
  <xsd:attribute name="partNum" type="PartType"/>
  <xsd:attribute name="onSaleDate" type="xsd:date"/>
</xsd:complexType>

<xsd:complexType name="ProductType">
  <xsd:element name="productName" type="xsd:string"/>
  <xsd:element name="quantity_in_stock">
    <xsd:simpleType base="xsd:positiveInteger">
      <xsd:maxExclusive value="500"/>
    </xsd:simpleType>
  </xsd:element>
  <xsd:element name="price" type="xsd:decimal"/>
  <xsd:element name="description" type="xsd:string" minOccurs="0"/>
</xsd:complexType>

<xsd:simpleType name="PartType" base="xsd:string">
  <xsd:pattern value="[A-Z]{3}-[A-Z]{3}d{3}"/>
</xsd:simpleType>

purchase-xml and its XML Schema purchase-xsd (from W3C)

Purchase-XML.XML
defined without schema
?xml version='1.0' encoding='utf-8'?>
<!-- Purchase Order without using schema -->
<purchaseOrder orderDate="1999-10-20">
  <shipTo country="US">
    <name>Alice Smith</name>
    <street>123 Maple Street</street>
    <city>Mill Valley</city>
    <state>CA</state>
    <zip>90952</zip>
  </shipTo>
  <billTo country="US">
    <name>Robert Smith</name>
    <street>8 Oak Avenue</street>
    <city>Old Town</city>
    <state>PA</state>
    <zip>95819</zip>
  </billTo>
  <comment>Hurry, my lawn is going wild!</comment>
</purchaseOrder>

<items>
<item partNum="872-AA">
  <productName>Lawnmower</productName>
  <quantity>1</quantity>
  <USPrice>148.95</USPrice>
  <comment>Confirm this is electric</comment>
</item>

<item partNum="926-AA">
  <productName>Baby Monitor</productName>
  <quantity>1</quantity>
  <USPrice>39.98</USPrice>
  <shipDate>1999-05-21</shipDate>
</item>

**Purchase-XSD.XSD Schema**

```xml
<?xml version='1.0' encoding='utf-8'?>
<!--  Purchase Order  shcema  -->
<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Purchase order schema for Example.com.
      Copyright 2000 Example.com. All rights reserved.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:element name="purchaseOrder" type="PurchaseOrderType"/>
  <xsd:element name="comment" type="xsd:string"/>
  <xsd:complexType name="PurchaseOrderType">
    <xsd:sequence>
      <xsd:element name="shipTo" type="USAddress"/>
      <xsd:element name="billTo" type="USAddress"/>
      <xsd:element ref="comment" minOccurs="0"/>
      <xsd:element name="items" type="Items"/>
    </xsd:sequence>
    <xsd:attribute name="orderDate" type="xsd:date"/>
  </xsd:complexType>
  <xsd:complexType name="USAddress">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="street" type="xsd:string"/>
      <xsd:element name="city" type="xsd:string"/>
      <xsd:element name="state" type="xsd:string"/>
      <xsd:element name="zip" type="xsd:decimal"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```
Defining Valid XML documents

```xml
<shipTo country="US">
  <item minOccurs="0" maxOccurs="unbounded">
    <complexType>
      <sequence>
        <element name="productName" type="xsd:string"/>
        <element name="quantity">
          <xsd:complexType>
            <xsd:restriction base="xsd:positiveInteger">
              <xsd:maxExclusive value="100"/>
            </xsd:restriction>
          </xsd:complexType>
        </element>
        <element name="USPrice" type="xsd:decimal"/>
        <element ref="comment" minOccurs="0"/>
        <element name="shipDate" type="xsd:date" minOccurs="0"/>
      </sequence>
      <attribute name="partNum" type="SKU" use="required"/>
    </complexType>
  </item>
  <!--
  Stock Keeping Unit, a code for identifying products
  -->
  <SKU/>
</shipTo>
```

**XML File with Schema**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<purchaseOrder orderDate="1999-10-20" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="Purchase-XSD.xsd">
  <shipTo country="US">
```

<name>Alice Smith</name>
<street>123 Maple Street</street>
<city>Mill Valley</city>
<state>CA</state>
<zip>90952</zip>
</shipTo>

<billTo country="US">
<name>Robert Smith</name>
<street>8 Oak Avenue</street>
<city>Old Town</city>
<state>PA</state>
<zip>95819</zip>
</billTo>

<comment>Hurry, my lawn is going wild!</comment>

<items>

=item partNum="872-AA">
<productName>Lawnmower</productName>
<quantity>1</quantity>
<USPrice>148.95</USPrice>
<comment>Confirm this is electric</comment>
</item>

=item partNum="926-AA">
<productName>Baby Monitor</productName>
<quantity>1</quantity>
<USPrice>39.98</USPrice>
<shipDate>1999-05-21</shipDate>
</item>
</items>

<comment>XML File with absolute Schema location specification</comment>

<?xml version="1.0" encoding="UTF-8"?>
<purchaseOrder orderDate="1999-10-20" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:MYNamespaceSchemaLocation="file://G:/MyData/CCNYCOURSES/Cs311Fall2005/Lec01_Aug29_2005/Examples/Purchase-XSD.xsd">
<shipTo country="US"/>
XML Concept
Defining Valid XML documents

For more details, please look at W3C Recommendation, 2 May 2001

XML Schema Part 0: Primer www.w3.org/TR/2001/REC-xmlschema-0-20010502/
