XML Structures

SET09103 Advanced Web Technologies

School of Computing
Napier University, Edinburgh, UK
Module Leader: Uta Priss

2008

Outline

What is XML?

XML means **Extensible Markup Language**.

- What is "markup"?
- What is "extensible"?

Invented in 1996 and became a W3C Recommendation in 1998.

XML is a simplified subset of SGML (Standard Generalised Markup Language).

Extensibility

Users can define their own XML elements.

This means that XML is a very general, multi-purpose format language.

It contains very few rules and constraints.

Why XML?

XML's purpose is to facilitate

- sharing of structured data (especially on the WWW).
- data serialisation.

Data serialisation

Before XML, every program had different configuration files.

Unix password file:

```
nobody:*:-2:-2:Unprivileged User:/:/usr/bin/false
root:*:0:0:System Administrator:/var/root:/bin/sh
```

Unix hosts file:

```
127.0.0.1 localhost
255.255.255.255 broadcasthost
```

Email configuration:

```
smtp-server=esmtp.napier.ac.uk:587
personal-name=John Doe
```

XML-based data serialisation

Mac OS X Info.plist file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-DTD PLIST 1.0//EN" "PropertyLi

<plist version="1.0">
<dict>
<key>CFBundleDevelopmentRegion</key>
<string>English</string>
<key>CFBundleExecutable</key> ...
```

Which types of files (XML or non-XML) are

- easier to read and write for people?
- easier to parse using software?
- easier to access using different software tools?
- usable in different operating systems (PC, Mac, Linux)?
- usable in other languages (Chinese, Arabic)?
- requires less storage space?

Just to make it really clear:

XML does not do anything!

- XML is not a programming language.
- XML doesn't solve any modelling problems.
- XML is a storage format.
- XML is used for data sharing and serialisation.
- With current technology, it is necessary to use XML.
- Ideally, tools should be used to read and write XML.

Syntax and Semantics

Syntax: rules for arranging signs (words, strings, terms, ...).
Syntactically, XML documents must be **well-formed**.
E.g.: "The force with you may be."

Semantics: the meaning of signs.
Semantically, XML documents must be **valid**.
E.g.: "My birthday is July 3, 1887."

Pragmatics: the use of signs.
E.g.: "This website is fun to use."

Well-formed XML documents

The following slides show the conditions for **well-formed** documents.

XML declaration

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

- The declaration is optional.
- If it exists, it must be at the start of the document.
- The document must comply with its declared character encoding.

A tree with one root

```
<html>
      <head> ...</head>
      <body> ...</body>
</html>
```

This is NOT well-formed:

```
<person>
      <name> ... </name> <address> ...</address>
</person>
<person>
      <name> ... </name> <address> ...</address>
</person>
```

Comments and special characters

Comments: (cannot contain - - itself.)

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

Special characters:

The characters <, >, & must be escaped: < > &

Elements and attributes

```
<element attribute="value">content</element>
```

- An element must have a start-tag and an end-tag.
- Elements can have several (differently named) attributes.
- The content of an element can contain text and further XML elements.
- Attribute values must be quoted (single or double quotes).
- All names and tags are case-sensitive.

```
<element attribute1="value" attribute2="value">
      <something> ... </something>
</element>
```

Proper nesting!

This is well-formed:

```
<h3> <i> ... </i> </h3>
```

This is NOT well-formed:

```
<h3> <i> ... </h3> </i>
```

Empty elements may contain attributes

Empty elements:

```
<br></br>
<br />
<br/>
```

With attributes:

```
<br clear="left"></br>
<br clear="left"/>
```

Entities

Entity references are like constant variables or placeholders.
Named character references are pre-declared or declared in a DTD.

- 5 pre-declared named character references:
  & (ampersand &)
  < (less than <)
  > (greater than >)
  ' (apostrophe ')
  " (quotation mark ")
- Numeric character references:
  ♣ (♣)
  Ω (Ω)
  ...

Semantics: valid documents

A schema or DTD provides a set of constraints about element and attribute names, their containment hierarchy and data types.

An XML document is **valid** if it is well-formed and complies with a schema/DTD.

Document Type Definition (DTD)

DTDs are the old schema format that was inherited from SGML.

- Advantages: DTDs are supported by all XML tools (because they are part of the XML 1.0 standard).
  They are easy to read and write.
- Disadvantages: DTDs don't support newer features and more complicated expressions.
  They don't use XML syntax themselves.

Public DTDs

For HTML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

For SVG:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
```

DTD system identifiers

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

<!DOCTYPE sometype [ this_is_where_the_definitions_go ]>
```

DTD elements

```
<!ELEMENT personlist (person*)>
<!ELEMENT person (name, birthdate?, address?)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT birthdate (#PCDATA)>
<!ELEMENT address (#PCDATA)>
```

Corresponds to

```
<personlist>
<person>
     <name>John Doe</name>
     <birthdate>11.1.2001</birthdate>
</person>
</personlist>
```

DTD attributes and entities

```
<!ATTLIST person idnumber CDATA #REQUIRED>

<!ENTITY abbrev "This is too long and needs to be abbreviat
```

Correspond to

```
<person idnumber="274">...
... &abbrev; ...
```

XML Schema

The newer **XML schema** language is the successor of DTDs.

Consists of XML Schema instances and XSD (XML Schema Definition).

Other schema languages exist: RELAX NG, ISO DSDL, ...

XML Schema example

```
<xs:element name="person">
<xs:complexType>
    <xs:sequence>
       <xs:element name="name" type="xs:string"/>
       <xs:element name="birthdate" type="xs:string"/>
       <xs:element name="address" type="xs:string"/>
    </xs:sequence>
</xs:complexType>
</xs:element>
```

XML versus HTML

HTML	XML
case insensitive	case sensitive
white space reduced	white space preserved
attributes need not be quoted	must be quoted
some tags are not closed	all tags must be closed
... is ok	is not ok
  etc mean something	 etc mean nothing
lots of named character references	only: &, <, >, ', "

Note:
there is a version of HTML that is as strict as XML.
It is called XHTML.

Elements versus attributes

Which of the following is better?

```
<person>
<id>1</id>
<name>Mary</name>
<address>10 Colinton Road</address>
</person>
```

Or this:

```
<person id='1' name='Mary' address='10 Colinton Road'>
</person>
```

Or this:

```
<person id='1'>
<name>Mary</name>
<address>10 Colinton Road</address>
</person>
```

Elements versus attributes

One strategy is to use attributes for **metadata** and elements for **data**:

```
<person id='1'>
<name>Mary</name>
<address>10 Colinton Road</address>
</person>
```

Advantages and Disadvantages of XML

The following list is adapted from
http://en.wikipedia.org/wiki/XML

Advantages

- text-based and extensible
- Unicode and international standards
- simple, efficient, and consistent parsing algorithms
- widely used, usable for all common data structures
- platform independent
- allows validation using schema languages

Disadvantages

- text-based, very verbose (compared to binary formats)
- higher storage, transmission and processing costs
- not really human readable
- tree-hierarchical data structure can be problematic
- distinction between content and attributes in XML seems unnatural