웹 기술 및 응용

XML DTD
(Document Type Definition)

2018년 2학기

Instructor: Prof. Young-guk Ha
Dept. of Computer Science & Engineering

목차

- XML Document Model
- Element Declaration
- Attribute Declaration
- Notation Declaration
- DTD vs. XML Schema

Document Model

❑ Special kind of document

- ➢ Written in a syntax designed to describe XML languages
- ➢ Explicitly lays out the grammar and vocabulary for XML languages

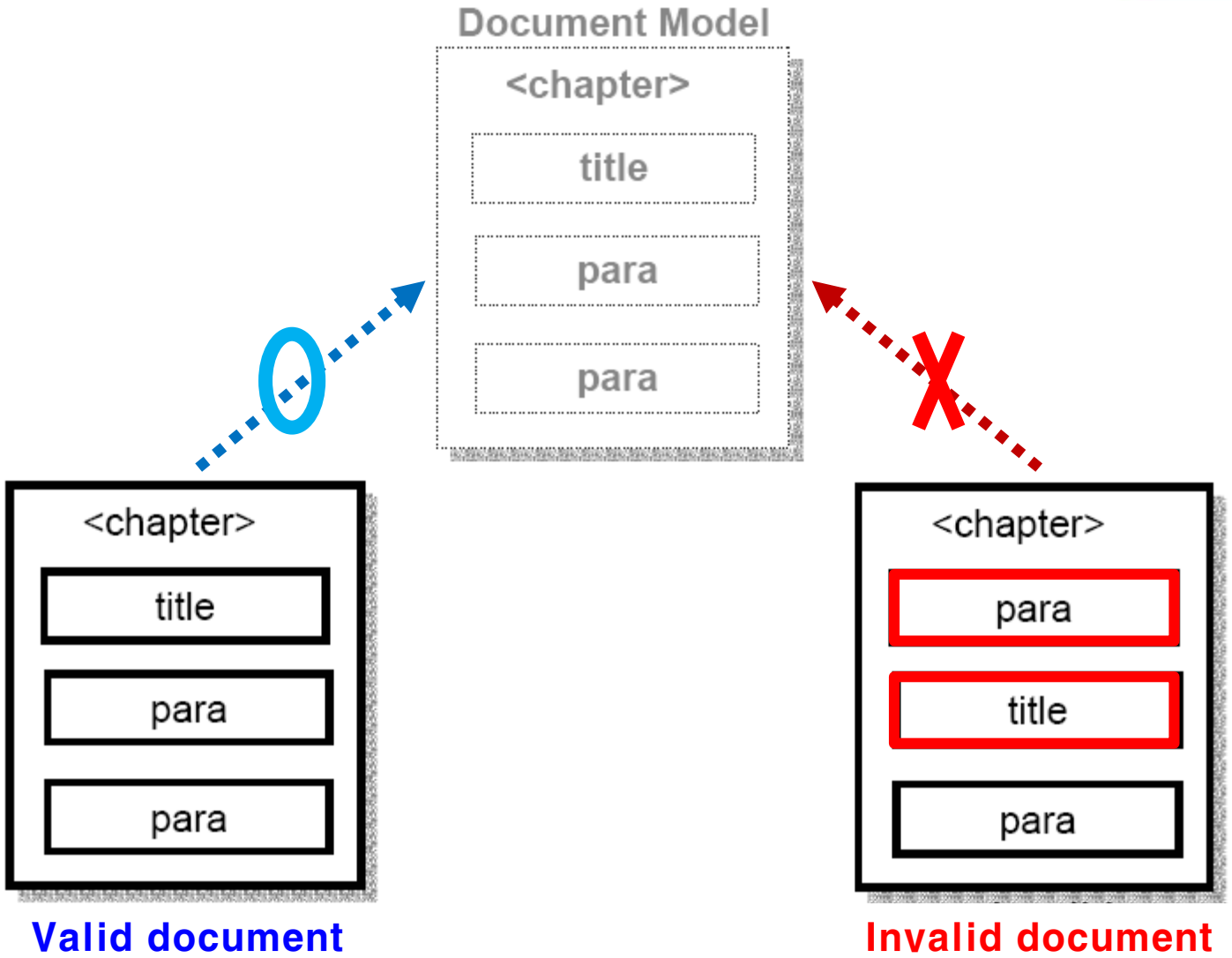
E.g.) Element "**letter**" declared in DTD

```
<!ELEMENT letter (to, subject, message, from)>
```

❑ Determines which document conforms to the XML language

- ➢ If document conforms, called ***valid*** document
- ➢ If document doesn't conform, called ***invalid*** document

Document Model (cont'd)

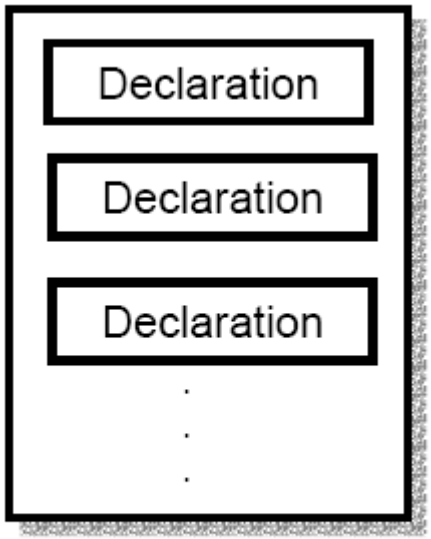


Document Model (cont'd)

- Pros
  - Document model guarantees the quality of document
  - Document model restricts the pattern to a predictable form
    - Makes XML document processing much easier
    - Reduces the possibility of errors
- Cons
  - Document model can be inconvenient to maintain
  - Document model can slow down processing
    - Document models need to be downloaded over network prior to parse XML documents
- Two ways to model XML documents
  - Document Type Definition (DTD)
  - XML Schema (to be discussed later)

DTD (Document Type Definition)

- DTD consists of a set of declarations
  - ***Element*** declaration
  - ***Attribute*** declaration
  - ***Entity*** declaration
  - ***Notation*** declaration



▲ DTD Syntax

- If redundant declarations occur
  1) The first one takes precedence
  2) Internal subset takes precedence
  3) All others are ignored

❑ Element declaration

```
<!ELEMENT name content-model>
```

- ➢ DTD content models
  - o Empty (defines empty elements)
  - o Only character data
  - o Only elements
  - o Mixed content
  - o Any content

Declarations

❑ Element declaration

➢ DTD content models

- Empty (defines empty elements)

```
<!ELEMENT para EMPTY>
```

E.g.)

```
<para/>
```

- Only character data

```
<!ELEMENT para (#PCDATA)>
```

- **PCDATA**: parsed character data (XML parser checks entity references, cf., **CDATA**)

E.g.)

```
<para>File type is &quot;JPG&quot;</para>
```

Declarations

- DTD content models (cont'd)
  - Only elements

```
<!ELEMENT chapter (para*)>
E.g.) <chapter>
        <para>Paragraph content</para>
      </chapter>

<!ELEMENT chapter (title?, (para | sect)+)>
E.g.) <chapter>
        <title>Title of document</title>
        <para>Paragraph content</para>
        <sect>Section 1 content</sect>
        <sect>Section 2 content</sect>
        <para>Paragraph content</para>
      </chapter>

      <chapter>
        <sect>Section 1 content</sect>
      </chapter>
```

o Structural constraints
- ( ) : group
- , : sequence
- | : choice

o Cardinality constraints
- *e* : mandatory
- *e*? : optional
- *e** : zero or more
- *e*+ : one or more

Declarations

- DTD content models (cont'd)
  - Only elements

```
<!ELEMENT chapter (title, subtitle*, (para | sect1)+)>
```

E.g.)

```
<chapter>
  <title>Title of document</title>
  <subtitle>Subtitle 1</subtitle>
  <subtitle>Subtitle 2</subtitle>
  <sect>Section 1 content</sect>
  <sect>Section 2 content</sect>
  <para>Paragraph content</para>
</chapter>

<chapter>
  <title>Title of document</title>
  <sect>Section 1 content</sect>
  <sect>Section 2 content</sect>
  <sect>Section 3 content</sect>
</chapter>
```

Declarations (cont'd)

- DTC content models (cont'd)

  - Mixed content

```
<!ELEMENT para (#PCDATA | emphasis | person)*>
```

E.g.)

```
<para>
        His name is <person>Tomas</person>
        He noticed a strong smell of gasoline,
        and realized he was
        <emphasis>really</emphasis>
        in trouble !
</para>
```

  - Any content

```
<!ELEMENT note ANY>
```

Declarations (cont'd)

❑ Attribute declaration (1)

```
<!ATTLIST elename
    attname1 atttype1 attdesc1?
    attname2 atttype2 attdesc2?
    ...
>
```

E.g.)
```
<!ATTLIST memo
          id          ID                  #REQUIRED
          security    (high | low)        "high"
          keywords    NMTOKENS            #IMPLIED >

<memo id="m1" security="high" keywords="key1 key2">
  This is a confidential idea.
</memo>
<memo id="m2" security="low"/>
```

Declarations (cont'd)

❑ Attribute declaration (2)

- Attribute types
  - `CDATA`
  - `ID`
  - `IDREF`, `IDREFS`
  - `NMTOKEN`, `NMTOKENS`
  - `ENTITY`, `ENTITIES`
  - Enumerated value list
- Attribute description
  - `#REQUIRED`: attribute should exist
  - `#IMPLIED`: optional and has no default value
  - `#FIXED "certainValue"`: explicitly specified as certain value
  - "*defaultValue*": optional and has default value

Declarations (cont'd)

❑ Attribute declaration (3)

➢ CDATA type (Character data type)

E.g.)

```
<!ATTLIST element1 attr1 CDATA #REQUIRED>
<element1 attr1="This is CDATA example">

<!ATTLIST element2 attr2 CDATA #IMPLIED>
<element2>
<element2 attr2="This is CDATA example">

<!ATTLIST element3 attr3 CDATA #FIXED "certainValue">
<element3 attr3="certainValue">

<!ATTLIST element4 attr4 CDATA "certainDefault">
<element4>
```

→ parsed as `<element4 attr4="certainDefault">`

Declarations (cont'd)

PCDATA vs. CDATA

PCDATA (Parsed Character Data) in element declarations

```
<!ELEMENT para (#PCDATA)>
<para>File type is &quot;JPG&quot;</para>
<para>if (a < b && a < 0) return 1;</para>
```

CDATA (Unparsed Character Data)

- When used in XML documents (CDATA section)

  → tags inside the text will *not* be treated as markup and entities will *not* be expanded

```
<para>
  <![CDATA[
    if (a < b && a < 0) return 1;
    &quot;This is CDATA example&quot;
  ]]>
</para>
```

- When used in DTD attribute declarations

  → attribute values must follow the same rules as normal textual content (same as PCDATA)

```
<!ATTLIST element1 attr1 CDATA #REQUIRED>
<element1 attr1="&quot;This is CDATA example&quot;">
```

Declarations (cont'd)

Attribute declaration (4)

- **ID type:** should be used with #REQUIRED
  (an ID value must be *unique* throughout the document)
- **IDREF type:** should be used with #REQUIRED
- **IDREFS type:** should be used with #REQUIRED
  - IDREFS ::= IDREF (' ' IDREF)*

E.g.)

```
<!ELEMENT part          (compatible | compaMany)*>
<!ATTLIST part          id      ID      #REQUIRED>
<!ATTLIST compatible    idref   IDREF   #REQUIRED>
<!ATTLIST compaMany     idrefs  IDREFS  #REQUIRED>
```

```
<part id="bolt-100"/>
<part id="bolt-101"/>
<part id="bolt-102"/>
<part id="nut-100">
   <compatible idref="bolt-100"/>
   <compaMany idrefs="bolt-100 bolt-1001 bolt-1002"/>
</part>
```

Declarations (cont'd)

❑ Attribute declaration (5)

➢ NMTOKEN type

- o May consist of letters, digits, periods (.), hyphens (-), underscores (_), and colons (:)
- o May start with any of the above characters
- o Any leading or trailing whitespace will be removed
- o No whitespace may appear within the value itself

➢ NMTOKENS type

- o NMTOKENS ::= NMTOKEN (' ' NMTOKEN)*

E.g.)

```
<!ATTLIST letter attachment  NMTOKEN  #IMPLIED>
<!ATTLIST letter attachments NMTOKENS #IMPLIED>

<letter attachment="readme.txt"> ... </letter>
<letter attachment="read me.txt"> ... </letter>

<letter attachments="readme.txt v3-4.exe"> ... </letter>
<letter attachments="readme.txt #v1-2.exe"> ... </letter>
```

Declarations (cont'd)

❑ Attribute declaration (6)

- ➢ ENTITY type
  - o The value should be general entity name
- ➢ ENTITIES type
  - o ENTITIES ::= ENTITY (' ' ENTITY)*

E.g.)

```
<!ENTITY   blueDot   SYSTEM "./bluedot.png" NDATA PNG>
<!ENTITY   redDot    SYSTEM "./reddot.png"  NDATA PNG>
<!ATTLIST  button    icon   ENTITY    #IMPLIED>
<!ATTLIST  button    icons  ENTITIES  #IMPLIED>

<button icon="blueDot"> ... </letter>
<button icons="blueDot redDot"> ... </letter>
```

Declarations (cont'd)

❑ Attribute declaration (7)

➢ Enumerated value list type

- o Attribute can have only one of the values in the list

E.g.)

```
<!ATTLIST part instock (true | false) #IMPLIED>
<part instock="true"/>   : valid
<part instock="1"/>      : invalid

<!ATTLIST schedule day
      (mon | tue | wed | thu | fri | sat | sun) #REQUIED>
<schedule day="thu"/>

<!ATTLIST shape type (circle | square | triangle) "square">
<shape/>
```

→ is parsed as `<shape type="square"/>`

- If user doesn't supply value for **type** attribute, default value "**square**" is applied

Declarations (cont'd)

❑ Notation declaration

<!NOTATION *name* SYSTEM *identifier*>

- ➢ Tells the XML processor what kind of data it's looking at
  - o Used with *unparsed entities*
- ➢ Exactly, what the meaning of notation is depends on the application
  - o Identifiers can be anything (URL or MIME type)
  - o MIME (Multi-purpose Internet Mail Extensions)
    - » Forms a standard way of classifying file types on the Internet (HTML: **text/html**, GIF: **image/gif**, MPEG: **video/mpeg**, …)

E.g.)

```
<!NOTATION GIF SYSTEM "image/gif">
<!ENTITY mypic SYSTEM "./erik.gif" NDATA GIF>
```

XML Schema: An Alternative to DTD

- DTD has its own strength
  - Compact size and simplicity
- DTD is not expressive enough for some needs
  - XML Schema provides much more control over data types and content models, for example
    - Elements can have enumerated value as well
    - Provides more complex patterns for content models
    - Inherits the properties of other declarations in an object-oriented way
- DTD uses an old and inflexible syntax
  - DTD syntax is different from XML syntax
  - Whereas XML Schema itself is a well-formed XML
    - XML Schema itself is defined by an XML document model (i.e., there are XML Schema DTD and XML Schema schema)

References

- XML 1.0 (Fifth Edition)
  - W3C Recommendation 26 Nov. 2008
  - http://www.w3.org/TR/xml

W3C WORLD WIDE WEB consortium Ubiquitous Web domain